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

Round 2 Comments
[in response to a notice in the September 15, 1999 Federal Register
(Volume 64, Number 178; pages 50058-50061)]

Updated comments will be posted at http://csrc.nist.gov/encryption/aes/round2/pubcmnts.htm.

From: "John Savard" <sewardconsulting@v-wave.com>
To: <AESRound2@nist.gov>
Subject: Comments on MARS
Date: Tue, 10 Aug 1999 13:53:39 -0600
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3155.0
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3155.0

10245-151st Street
Edmonton, Alberta
CANADA
T5P 1T6

(780) 444-3599

The choice of MARS as one of the five candidate ciphers to proceed to Round 2 of the AES process has surprised a number of people, as they have viewed that particular block cipher as inelegant or unwieldy.

I do not find the choice of MARS as a finalist surprising, as I consider security to be the most important characteristic that the block cipher chosen as the AES must have, and I think the unique design of MARS, with its unkeyed forward mixing and unkeyed reverse mixing stages will indeed help to protect MARS against future attacks.

I also note that several candidates, of which MARS is the only one to proceed to Round 2, had minor changes proposed for them. The changes proposed to MARS are claimed to make it harder to invert the key schedule and determine the original key from the subkeys. This will be useful in connection with what follows.

An obvious naive criticism of MARS is that, since the forward mixing and reverse mixing stages of MARS are unkeyed, they do not seem to contribute as much to its security as they should for the amount of processing they involve.

The rationale behind these stages being unkeyed, is, of course, a reasonable and understandable one. If the subkeys used for keying these stages contain information about the subkeys used for the cryptographic core stages of MARS, they could, however unlikely it may seem, provide a basis for an attack on MARS. At least, the presence of such subkeys would make the analysis of MARS much more complicated, creating uncertainty about its security.

If the mixing stages were keyed, I envisage that they would be keyed as follows: the copy of the 32-bit subblock used to index into the four S-boxes, before being used for that purpose, would be replaced by itself XOR a subkey. This would require 16 additional subkeys. Let us designate them as subkeys 40 through 55, with subkey 40 used in the first forwards mixing round, and so on (i.e., subkey 48 would be used in the first reverse mixing round).

To remove the concerns associated with placing the mixing stages under the control of the key, it would be sufficient to ensure that there is no way to determine the other 40 subkeys from the 16 subkeys used for that purpose.

I propose that the subkey generation process of MARS (as it now stands, after the "tweak") be modified as follows to achieve this.

After the generation of subkeys 30 through 39, the elements of the temporary array with 15 elements (T[0] through T[14]) are to be modified as follows:

T[0] = T[0] xor subkey 6
T[1] = T[1] xor subkey 8
.
.
.
T[11] = T[11] xor subkey 28

and

T[12] = T[12] xor subkey 1
T[13] = T[13] xor subkey 2
T[14] = T[14] xor subkey 3

This uses only subkeys generated in iterations preceding the final one, and excludes the subkeys modified for use in multiplication, and provides additional non-invertibility. (This is only one of several possible alternatives. One could perform no extra step, and rely on the non-invertibility existing in the present key generation algorithm, or one could seek to achieve greater non-invertibility through more elaborate measures, for example involving performing a MARS encryption of a constant with the subkeys generated so far, the other new subkeys being zero.)

Then, two additional iterations of the process used to generate subkeys in groups of 10 will be performed (the first one with j=4, the second with j=5; it may be advisable to change "4i+j" to "8i+j" in the description of the key generation process to accomodate this) but each time only the first 8 additional subkeys will be used.

If this key schedule is seen as adequate to avoid introducing additional attacks, this modification of MARS would ensure that all the steps in a MARS encryption make their full contribution to its security.

John J. G. Savard

From: "John Savard" <sewardconsulting@v-wave.com>
To: <AESRound2@nist.gov>
Subject: MARS
Date: Tue, 10 Aug 1999 16:22:03 -0600
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3155.0
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3155.0

10245-151st Street
Edmonton, Alberta
CANADA
T5P 1T6

(780) 444-3599

In a previous communication, I suggested the following:

(begin quote)
I propose that the subkey generation process of MARS (as it now stands, after the "tweak") be modified as follows to achieve this.

After the generation of subkeys 30 through 39, the elements of the temporary array with 15 elements (T[0] through T[14]) are to be modified as follows:

T[0] = T[0] xor subkey 6
T[1] = T[1] xor subkey 8

To: AESround2@nist.gov
Date: Wed, 18 Aug 1999 05:19:42 -0700
From: " " <csybrandy@my-deja.com>
X-Sent-Mail: off
X-Mailer: MailCity Service
Subject: AES Algorithm Selection
X-Sender-Ip: 147.160.10.21
Organization: My Deja Email (http://www.my-deja.com:80)

One thing that should be considered is how well an algorithm is suited for Palm Pilots and such. One question could be which is cheaper, memory or hardware multipliers? If memory is cheaper, Rijndeal or Twofish may be very suitable. If hardware multipliers are cheaper, then RC6 is your best bet. As for MARS and Serpent, I am not sure about either along these lines. I have not looked at Serpent in depth and MARS requires both a multiplier and memory.

Casey Sybrandy

From: jmm@elegant.com (John Macdonald)
Subject: AES criteria
To: AESround2@nist.gov
Date: Wed, 1 Sep 1999 18:52:42 -0400 (EDT)
Cc: jmm@elegant.com
X-Mailer: ELM [version 2.4 PL23]

It has been suggested by Bruce Schneier in his Crypto-Gram newsletter that comments on cryptographic requirements from users is as important for your consideration as the mathematical analysis of the particular algorithms that are being considered.

I work for a small company (<20 people, < $10M annual sales) that licenses distributed system adminstration programs. This is a natural for using cryptography:

- protect info in transit that deals with security repated issues
- authorize remote control and update of the distributed facets

For our purposes, there are 3 criteria that are important. The first two are relevant to your contest:

(1) encryption must be in software: our code runs on many different Unix platforms, we don't want to deal with providing hardware interfaces to those many different platforms and we can't be requiring customers to be adding and maintaining hardware to their systems

(2) it must be reasonably efficient on workstation-class machines

These two requirements are ones that you are seem to be targetting well with your current process. Please continue to treat them as important. Our third criteria is unfortunately outside your mandate:

(3) we can't afford to treat foreign sales differently, or to have any extra paperwork. Export controls mean that we use trivial encryption instead of any serious encryption. We cannot afford to have a secure US/Canada release of our software and a separate unsecured release for the rest of the world. Applying for export licenses for each foreign sale would be even worse.

While this final requirement is not within your mandate, it means that we, like most other small companies, will be unable to make use of the winning candidate.

--
objects: | John Macdonald
Think of them as data with an attitude. | jmm@elegant.com

X-Sender: doshel@soli.inav.net
Date: Thu, 23 Sep 1999 22:01:28 -0500
To: AESRound2@nist.gov
From: "David C. Oshel" <dcoshel@pobox.com>
Subject: I am pro-Rijndael

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

I tend to be pro-Rijndael because it is royalty-free, Belgian (i.e., it has a politically neutral provenance, unlike the U.S. and Euro corporate or U.S., English, Israeli, etc. national contributions), appears extremely competent and is frankly very easy to implement. I intend to consider Rijndael seriously in those few applications I may work on that benefit from competent obfuscation, regardless of how the AES competition turns out. Although the competition is interesting, I feel the practice will tend to be based on less formal, indeed, even irrelevant, considerations than the cryptology has been so far.

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_

-----BEGIN PGP SIGNATURE-----
Version: PGPfreeware 6.0.2i for non-commercial use

iQA+AwUBN+rplvxWvgP9KeXyEQLsaQCRAWejb9yWlTAXzWbgYSk5ttVecACcDl9Z
occpqPJFGEZcxwrNBgLuv2U=
=fCeo
-----END PGP SIGNATURE-----

From: certifyit@alltel.net
Date: Mon, 27 Sep 1999 09:37:34 -0500
X-Mailer: Mozilla 4.61 [en] (WinNT; U)
X-Accept-Language: en
To: AESround2@nist.gov
Subject: Intellectual property concerns with AES

To: NIST

I recently went to look at some software concerning one of the finalist candidates for the AES. From reading what was on the NIST web site, I assumed that such software would be in the public domain. What I found was that the software I downloaded contained copyright notices. I sent an e-mail message asking about this to an author of the algorithm and got back a one sentence response saying simply "I don't know". This did not seem to me to be a very adequate response to my inquiry.

What concerns me is that the finally decided upon algorithm, while itself might be free of patent/copyright constraints as per NIST requirements, might have "efficient"/"optimal" software/hardware implementations still protected by such patent/copyright holders. I believe this should be of concern to NIST also.

What I suggest is that NIST formulates an approach similar to the GNU copyleft concept which is covering more and more software these days. (See http://www.fsf.org/ and more specifically http://www.fsf.org/copyleft/copyleft.html for more information on this type of protection.) This type of protection basically says that any implementation derived from any other implementation is still under the constraint of being fully open software.

It seems to me that it would be beneficial for NIST to look into this matter more carefully to insure that after an AES algorithm is decided upon, it still remains free in all implementations for anyone to use.

Sincerely,

John Eichler
CertifyIt@Alltel.net

From: "Sybrandy, Cornelius" <sybrandc@ctc.com>
To: "'AESround2@nist.gov'" <AESround2@nist.gov>
Subject: Adding additional rounds to make a cipher more secure
Date: Tue, 28 Sep 1999 16:21:35 -0400
X-Mailer: Internet Mail Service (5.5.2448.0)

There are circumstances where the addition of an extra round within a cipher is necessary for it's security. For example, if rc6 were suceptable to an attack that could break all 20 rounds, the addition of a 21st or 22nd round may be an option. However, because of the different designs of the algorithms, some have more rounds than others. Let's say that two algorithms, A and B run at the same speed. Algorithm A uses 20 rounds, while Algorithm B uses 8. Overall they run at the same speed, however each round of B takes much longer than a round of A. This means that if an extra round were added to B, it would have a more significant effect on the speed of the cipher than if one round were added to A. Since this is a simple way to make a cipher a bit more secure, this may be a point of consideration when deciding on an AES finalist.

Cornelius A Sybrandy
Concurrent Technologies Corporation
100 CTC Drive
Johnstown, PA 15904
Phone: (814) 269-6587
"Imagination is more important than knowledge. Knowledge is limited.
Imagination encircles the world."
Albert Einstein

From: "Sybrandy, Cornelius" <sybrandc@ctc.com>
To: "'AESround2@nist.gov'" <AESround2@nist.gov>
Subject: AES and Palm Pilots or hardware
Date: Tue, 28 Sep 1999 16:35:11 -0400
X-Mailer: Internet Mail Service (5.5.2448.0)

One of the criteria for AES is that the selected algorithm performs well in smart-cards and other hardware solutions. One aspect of this is the cost of the hardware to accomodate the cipher. One example could be RC6 vs. Twofish. RC6 relies heavily on multiplications, which require relatively expensive multipliers. Twofish, on the other hand, uses two tables. If memory is more expensive, than RC6 should be used. If a multiplier is more expensive, than Twofish is used. Simply put, it's nice to see which algorithm is the fastest in hardware, but which one is the cheapest to implement in hardware or in products such as Palm Pilots.

Cornelius A Sybrandy
Concurrent Technologies Corporation
100 CTC Drive
Johnstown, PA 15904
Phone: (814) 269-6587
"Imagination is more important than knowledge. Knowledge is limited.
Imagination encircles the world."
Albert Einstein

From: "Patterson Programming" <stealth@scotlandmail.com>
To: <AESround2@nist.gov>
Subject: Re: Twofish
Date: Wed, 29 Sep 1999 09:33:05 -0400
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3110.1
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3

To whom it may concern,

I write to inform you of an issue regarding Twofish, one of the AES candidates. Having reviewed the paper by Counterpane Systems, I find that the key element used in the algorithm Twofish is virtually identical to the one I used in an algorithm of my own design. It is relevant to me because I provided a paper copy of a reference implementation of my algorithm LeapFrog to Mr. Schneier in the Fall of 1996. Note that the said algorithm was provided to him prior to the AES call for papers. The algorithm is not patented, but the reference implementation is registered with the U.S. Copyright Office. Ref: http://lcweb.loc.gov/copyright Registration numbers: TXu 763-698 and TXu 796-545. Reputable people would attest to the fact that deposits were sent to Counterpane Systems prior to the call for papers. As stated, the algorithm is not patented, I object to the fact that any reference to my design was, I believe, intentionally omitted from his papers. Note that while the two algorithms are not identical, and Twofish uses some elements not employed in my design, an unbiased look at the two designs (knowing that my design was seen by the designers of Twofish) should lead to questions. A simple example follows:

From the S-box function in Twofish:

```
y0 = q1[q0[q0[y2;0] Xor l1;0] Xor l0;0]
y1 = q0[q0[q1[y2;1] Xor l1;1] Xor l0;1]
y2 = q1[q1[q0[y2;2] Xor l1;2] Xor l0;2]
y3 = q0[q1[q1[y2;3] Xor l1;3] Xor l0;3]
```

(q0 and q1 are fixed permutations on 8-bit values)

Ref: http://www.counterpane.com/twofish-paper.html

From the main function in LeapFrog:

```
* get the T variables
T1 = A1(p1), T2 = A2(p2), T3 = A3(p3), T4 = A4(p4)

* note the T variable order
 p5 = p5 Xor A1(A1(T2 Xor T3) Xor T4)
 p6 = p6 Xor A2(A2(T1 Xor T4) Xor T3)
 p7 = p7 Xor A3(A3(T4 Xor T2) Xor T1)
 p8 = p8 Xor A4(A4(T3 Xor T1) Xor T2)
```

p5 = (p5 + T1) And FF
p6 = (p6 + T2) And FF
p7 = (p7 + T3) And FF
p8 = (p8 + T4) And FF

I find it interesting that the significant element of my design was employed in Twofish for both the key-expansion code and the main function. Also, the Twofish paper notes that using only random S-boxes would be about as secure as optimized boxes. And there are other references that lead me to believe that he was mentally trying to distinguish Twofish from LeapFrog. I may inform many individuals in the cryptographic community of my concerns and let them judge for themselves. I guess I should rename my algorithm Onefish, as it looks more like Twofish than Blowfish does. Please reply if you have any questions.

Regards,
jp

Patterson Programming

From: certifyit@alltel.net
Date: Wed, 29 Sep 1999 13:21:30 -0500
X-Mailer: Mozilla 4.61 [en] (WinNT; U)
X-Accept-Language: en
To: AESround2@nist.gov
Subject: Follow-up to Intellectual property concerns with AES

To NIST,

The following scenarios might indicate possible problems that could arise in the future.

Scenario 1: A new Motorola microprocessor comes out with a feature that permits microprogramming an instruction which combines the exclusive OR operation with a shift such that I can effectively make a new "custom" instruction which then my software can take advantage of. I copyright my new "more efficient" code for the AES and offer to license it to anyone (for a fee, of course) who is interested in doing the job faster using that processor.

Scenario 2: Philips introduces a new programmable logic array that is an eight-pin DIP chip which permits me to implement the AES algorithm in such a manner that I can process over 500,000,000 bytes per second. The chip sells for $1.95 in quantities of 100. Being astute, I want to patent my implementation and then sell the chip for $50.00 each to makers of all computers to incorporate on their motherboards.

Scenario 3: The year is 2020 and the patent on the AES algorithm has long since run out. Two years ago the new quanta/quantum (i.e., light combined with certain a quantum-mechanic effects) process had been perfected. The new technology allows, depending on how many parallel units on the chip that are utilized, up to 5 trillion operations per second with a theoretical of 87.6 trillion operations per second possible. (Moore's Law just flew south!) NIST has openly admitted that the original AES algorithm is likely to fall with 2 years or sooner. People have started turning to "quad-AES" as an alternative that is predicted to last for at least another 25 years. The corporation that came up with the quad-AES scheme has patented their new scheme because of a unique twist (in a "stroke of genius") in the technique they use but under the covers, it still uses the old original AES encryption method just done multiple times.

In scenarios like the above, the approach to just put the AES algorithm into the public domain may not cover the future use(s) of this algorithm properly. It is rather suggested that NIST licenses its use under something similar to the GNU licensing agreement where any further developments must, because they are based upon a licensing agreement, be done with making such developments available to everyone else under the same licensing agreement.

The main point here is that either a patent and/or copyright may be used advantageously via the proper licensing agreement to insure the openness of use of the AES algorithm. Therefore it is suggested that NIST actually license this technology (in a similar manner to how NASA licenses its technology) to anyone who desires to utilize it. This approach, it is believed, better fulfills the "spirit" of what NIST is trying to accomplish.

Sincerely,

John Eichler
CertifyIt, Inc.

From: "Patterson Programming" <stealth@scotlandmail.com>
To: <AESround2@nist.gov>
Subject: Clarification of previous message.
Date: Thu, 30 Sep 1999 09:46:26 -0400
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3110.1
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3

Hello,

I am writing to clarify one item regarding the message I sent you on 9/29/99 in reference to Twofish. Clarification: the only implementation of LeapFrog that was published was in: TXu 876-599. The original implementation (identified in my previous message) was used only to test the round function. The one listed in this message employs alterations to the key-expansion routine, as well as whitening. (to create a secure implementation) I am sure Counterpane Systems was sent a copy of the original papers, but I cannot verify that the revised implementation was sent. However, it is not relevant because the round function from LeapFrog that may have been generalized to Twofish has not been revised since 1996. It may not be important, but I wanted to advise you.

Regards,

JP
Patterson Programming

From: "Patterson Programming" <stealth@scotlandmail.com>
To: <AESround2@nist.gov>
Subject: Twofish
Date: Fri, 1 Oct 1999 08:29:59 -0400
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3110.1
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3

X-MIME-Autoconverted: from 8bit to quoted-printable by email.nist.gov id IAA08950

Regarding Twofish and LeapFrog:
Re-reading my posts regarding this issue, I suspect that some could make incorrect inferences from my statements. I may have reacted too strongly. I only wanted to point out that Twofish's substitution stage using a parity function and compound look-ups is *not* a novel idea. (If that is what they meant in the Twofish paper) Cryptography is not my main interest, so please take my comments for what they are worth.

Regards,
JP

Patterson Programming

X-Sender: foti@csmes.ncsl.nist.gov
X-Mailer: QUALCOMM Windows Eudora Pro Version 3.0.5 (32)
Date: Tue, 05 Oct 1999 12:23:35 -0400
To: AESround2@nist.gov
From: DWHITING@hifn.com (by way of Jim Foti <jfoti@nist.gov>)
Subject: RE: Twofish

Jim, could you please add this to the AES public comments for Twofish?

Bruce, could you post it on sci.crypt?

As a member of the Twofish design team, I (Doug Whiting) wanted to respond to a recent postings on sci.crypt about similarities between Twofish and "Leapfrog" (from JP of Patterson Programming). I am not trying to comment at all on whether Leapfrog is a good or bad idea: indeed, I have far too little information (or time) to make such a determination. However, it did not have any influence on the design of Twofish.

The Twofish design borrowed several ideas from previous ciphers. In each case, where it was done consciously, we openly gave credit to the previous inventors. Most advances in the field of cryptology come by building on the previous work of others.

In fact, I was the one who came up with the 8-bit S-box construction in Twofish. Bruce Schneier was an important part of the team, but he had no influence in that part of the design. Until I saw learned of the posting last week, I had never heard of Leapfrog nor seen any of its design elements. Thus, while there may be similarities (and I will show below that there are very significant differences) between this Twofish element and the Leapfrog construction, they are totally coincidental at most.

Last week, after seeing the sci.crypt posting, I queried Bruce about the Leapfrog paper that was sent to him several years ago. He has no recollection of it at all. Bruce routinely receives such unsolicited cipher descriptions, and he almost always discards them unread. This fact is not a criticism of Leapfrog, merely as a measure of how busy Bruce is with "paying customers". Even if he had read it, he never told me anything about it, so it could not have influenced my design.

Further, let's look at the elements in question. Here is the equation for an 8-bit Twofish S-box, in the 128-bit key case:

    y0 = q1[q0[q0[y[2,0]] Xor L[1,0]] Xor L[0,0]]

The L[i,j] values are key material bytes that DO NOT CHANGE for a given key. Thus, this mapping from a single byte of y2 to y0 is an 8-bit to 8-bit nonlinear mapping (S-box), fixed for a given key. In other words, this mapping can be precomputed as a small (256-byte) lookup table for each key. This S-box precomputation is used to speed up certain Twofish implementations.

Now, contrast this with the Leapfrog excerpts provided:

    * get the T variables
    T1 = A1(p1), T2 = A2(p2), T3 = A3(p3), T4 = A4(p4)

```
* note the T variable order
p5 = p5 Xor A1(A1(T2 Xor T3) Xor T4)
p6 = p6 Xor A2(A2(T1 Xor T4) Xor T3)
p7 = p7 Xor A3(A3(T4 Xor T2) Xor T1)
p8 = p8 Xor A4(A4(T3 Xor T1) Xor T2)

p5 = (p5 + T1) And FF
p6 = (p6 + T2) And FF
p7 = (p7 + T3) And FF
p8 = (p8 + T4) And FF
```

This is a Feistel-like structure, in which p1..p4 are used to modify p5..p8. Note that the chained table lookup in question is data dependent, not key dependent. This is an inference I draw from the usage, since, if p1..p4 were instead fixed key material, the equation reduces to a simple xor/add of a key-dependent constant. Thus, the modification to p5..p8 cannot be precomputed in any reasonably sized table (unless 16MBytes is reasonable). The change to p5..p8 is very linear, as is common in Feistel structures.

In other words, Leapfrog apparently uses this construction as a diffusion mechanism. This may be a very nice idea, but it is dramatically different construction than Twofish, which uses the S-box and the MDS matrix for diffusion. More abstractly, Twofish uses chained S-boxes as follows:

```
  y = f(x,Key)                // x,y are 8 bits each
```

while Leapfrog does

```
  y = y ^ F(p1,p2,p3,p4).     // y, p1..p4 are 8 bits each
```

That f and F are somewhat similar is not surprising -- block ciphers are almost always built out of running things through S-boxes and then combining with xor or add. SAFER+ does something very similar, as do E2 and LOKI97. In fact, looking across rounds (as opposed to within a round), almost every block cipher looks something like this, including DES.

The fact is that Twofish uses 8x8 key-dependent S-boxes. The Leapfrog construction is nothing of the sort. Key-dependent S-boxes are a crucial design element to add security to Twofish. Chaining a fixed S-box with key material xors is a very simple way to achieve this, although there may be other or better methods.

In any case, as noted in the sci.crypt posting, there is no patent involved, so Twofish is still available as a public domain cipher. Further, since I never had access to any information about Leapfrog, there is no possible copyright issue either.

We are happy to give credit where credit is due, but it seems like the inventor of Leapfrog has dramatically misinterpreted both Twofish and its design team.

Sincerely,

Doug Whiting, CTO, Hi/fn
dwhiting@hifn.com

From: DJohn37050@aol.com
Date: Fri, 15 Oct 1999 14:49:33 EDT
Subject: AES, ANSI X9F1, and future resiliency
To: AESround2@nist.gov
X-Mailer: Windows AOL sub 41

NIST,
When I presented my submission on "Future Resiliency: A Possible New AES Criterion" to ANSI X9F1, there was a general consensus (we took an informal vote) that NIST should include this criterion in its selection process. Participants include representatives from banks, auditors, vendors and government.
Don Johnson

From: matthew.fisher@convergys.com
X-Lotus-FromDomain: CVG
To: vincent.rijmen@esat.kuleuven.ac.be, daemen.j@protonworld.com,
rja14@cl.cam.ac.uk, biham@cs.technion.ac.il, lars.knudsen@ii.uib.no,
kelsey@counterpane.com, schneier@counterpane.com, daw@cs.berkeley.edu
cc: aesround2@nist.gov
Date: Tue, 26 Oct 1999 12:40:05 -0400
Subject: Weakness in Rijndael/Serpent key schedule?

Sirs,

Is the fact that a Rijndael key can be recovered via the XOR of two Cipher Keys and the XOR of the Round Keys important?

'Serpent' also seems to have this property. Given the XOR of two keys K1,K2 then the XOR of the prekeys w[0..131] could be determined. Now if an attacker knows the XOR of {k0,k1, ...,k131 } for the round keys, it is straight forward to determine K1 and K2.

This property doesn't seem relevant to DES.

This property seems to violate the 'Prudent Rules of Thumb for Key-Schedule Design' in 'Related Key Cryptanalysis of 3-WAY, Biham-DES,CSA,DES-X, NEWDES, RC2 and TEA' by Kelsey, Schneier and Wagner. That paper states that 'Key schedules should be specifically designed to resist differential related-key attacks.'

I have not been able to create a practical attack based on this analysis. How an attacker could get the XOR of the round keys but not the keys themselves is certainly a mystery.

thanks,
--Matthew

I posted the following to sci.crypt. I was hoping for commets on the idea.

A comment on the key schedule for Rijndael.

Hello sci.crypt,

I believe I have found a weakness in the key schedule for Rijndael. I turn to this august body for confirmation or harsh refutation of my theory. A quick note on my credentials; I am only an amateur cryptographer but a long time professional computer scientist. This post is a follow up to the ‘Curious Keys in Rijndael’ post.

Abstract

In this paper the block cipher Rijndael is analyzed. Rijndael is submitted as a candidate for the Advanced Encryption Standard. The cipher has variable key and block length. This paper focuses on the key length of 128 bits with the block length of 128 bits. The Rijndael cipher is an iterated ten round block cipher for this case. Here it is shown, the Cipher Key can be recovered if an attacker has the 'exclusive or' of several Round Keys. Moreover, one byte of key recovery can be mounted on Rijndael for certain key relationships and strange chosen cipher texts.

This note is just the beginning of the full paper I am working on. If it turns out that I goofed up then this posting will save me the work

I will use the notation adopted in 'The Rijndael Block Cipher' by Joan Daemen and Vincent Rijmen. To understand the attack, you will have to read 'The Rijndael Block Cipher.'

To begin with, it can be easily shown that if an attacker can gain the XOR of several Round Keys the Cipher Key may be obtained for many cases.

Here is an example of recovering a Cipher Key by knowing the XOR of the Round Keys. I have abbreviated the example to the last four bytes of the Cipher Key. The rest may be obtained by having a more Round Key XOR. A total of four sequential Round Key XOR is enough to obtain the entire Cipher Key.

Here are the keys and Round Key 0 and the first four bytes of Round Key 1. The Round Keys are stored in a structure called W. W has four bytes per entry. The W[0..3] structure would be the first Round Key for Nk =4 and Nb = 4.

Key K1 = c6 81 4b e2 54 bd 7c e1 01 31 56 0f 67 87 aa 3c
Key K2 = 7e 6b fb fb f6 df 1c 87 bf de 72 47 66 59 88 59

W Byte K1 K2 Xor

00 00 198 126 184
00 01 084 246 162
00 02 001 191 190
00 03 103 102 001

01 00 129 107 234
01 01 189 223 098
01 02 049 222 239
01 03 135 089 222

02 00 075 251 176
02 01 124 028 096
02 02 086 114 036
02 03 170 136 034

03 00 226 251 025
03 01 225 135 102
03 02 015 071 072
03 03 060 089 101

04 00 063 104 087
04 01 034 086 116
04 02 234 116 158
04 03 255 105 150

It can be seen from the Rijndael key schedule that the bytes of W[4] are a combination of the bytes of W[0] and bytes of W[3]. The formula is:

W[4] = W[0] ^
SubByte(RotByte(W[3]) ^ Rcon[1]

Where ^ is XOR and Rcon is a fixed set of round constants. Thus due to the RotByte the formula for the first byte of W[4][0] is

W[4][0] = W[0][0] ^ SubByte(W[3][1]) ^ Rcon[1]

Now for two keys K1 and K2 the XOR of W[4][0] is

W1[4][0] ^ W2[4][0] =
W1[0][0] ^ W2[0][0] ^
SubByte(W1[3][1]) ^ SubByte(W2[3][1])

The Rcon is canceled by the XOR.

In our example, the attacker knows

W1[0][0] ^ W2[0][0] = 184

And

W1[4][0] ^ W2[4][0] = 87

And thus

W1[0][0] ^ W2[0][0] ^ W1[4][0] ^ W2[4][0] =
184 ^ 87 = 239

And thus

239 = SubByte(W1[3][1]) ^ SubByte(W2[3][1])

Also the attacker know

W1[3][1] ^ W2[3][1] = 102

Given the XOR of the inputs, 102, and the XOR of the outputs, 239, into a bijective S-box, it is trivial to calculate the possible inputs. For the example, the only possible inputs for the Rijndael S-box are

225 and 135

225 ^ 135 = 102
S(225) ^ S(135) = 248 ^ 23 = 239

Now the attacker knows that the W1[3][1] must be a 225 or a 135. Since, the Cipher Key is copied into W[0..3], the attack knows one of the key bytes. All of the other bytes can be obtained in a similar manor. To get W[0..2], more Round Key XOR are needed.

This proves that given only the XOR of the Round Keys an attacker can recover the entire Cipher Key. How an attacker would gain such information is unknown. A possible method is that power or time analysis would reveal some relative measure but not an actual key. This relative measure might somehow be combined for two keys to create an XOR of the Round Keys without revealing the actual values.

Here we go with the byte recovery attack.

For a chosen cipher text, related key attack the conditions are:

1. An implementation using Nb = 4 and Nk =4 i.e. 128 bit key and 128 bit block
2. Two Cipher Keys in Rijndael such that K1 == K2 except for the first byte i.e. K1[0] != K2[0]
3. A relationship between K and K1 such that the Round Key (RK) W1[36][3] == W2[36][3]
4. Two cipher text, C1, C2 such that C1 XOR RK1(11) == C2 XOR RK2(11). RKy(x) is Round Key number x for key y. C1 will be generated under K1 and C2 will be generated under K2. This requirement is that the cipher texts must be equal before the last round key application in the tenth round of Rijndael.

If the above conditions are met, one byte of the tenth Round Key can be found with high probability usually one half. Since, the Cipher Key can be constructed from any Round Key, the remaining bytes of the tenth Round Key can be guessed. Once a full Round Key is guessed, the Cipher Key can be generated and tested against one known plain text cipher text pair. The guessing and testing process will have to be done on about 2^120 Round Keys. 2^120 is obviously must faster than 2^128 thus the attack is faster than exhaustive search.

The relationship between the keys is quite awkward but is certainly possible. I written a simple program to generated keys matching the above criteria.

Here is the attack on Cipher Key K1.
1. Given the two cipher texts C1 and C2 from above. XOR C1 and C2. The result will be the RK1(11) XOR RK2(11)
2. Due to the awkward relationship between the keys and the key schedule for Nk = 4, information about the RK1(10) can be gained from RK1(11) XOR RK2(11).
3. The Round Key sub structure W1[40] will contain four bytes. The first, A, and fourth, D, bytes of W1[40] are related by the Rijndael S-Box.
4. The relationship is that A is the XOR of inputs into the S box and D is the XOR of the outputs.
5. Typically, only one pair of inputs/outputs meets the criteria. Thus byte one of W1[39] must be one of the pair. This gives a 50% chance of guessing the byte. W1[39] is in RK1(10).

6. Guessing the other 2^120 bits can be done by exhaustive search.

How an attacker could gain the required information is unknown. The above attack appears to be entirely theoretical. It is interesting none the less.

Several possible extensions exist for this attack. Here is one possible extension. It appears that a given XOR input has a limited set of output XOR for the eleventh Round Key. If the set of output XOR is small in comparison to 2^128 then a table look up attack could be mounted. The table would contain all possible XOR outputs and the corresponding keys for a particular input XOR, K1 ^ K2. An attack would then randomly XOR cipher text from two related keys. The results of the cipher text XOR would be looked up in the table and the keys tried. When two cipher texts were equal in the last round before the key XOR, the proper entry would be found and thus the Cipher Key would be found in the table.

Got all that? ;-)

--Matt

Matthew E Fisher
mfisher@magicnet.net

X-Authentication-Warning: apal.ii.uib.no: larsr owned process doing -bs
Date: Wed, 27 Oct 1999 00:02:30 +0200 (MET DST)
From: Lars Ramkilde Knudsen <larsr@ii.uib.no>
To: matthew.fisher@convergys.com
cc: vincent.rijmen@esat.kuleuven.ac.be, daemen.j@protonworld.com,
rja14@cl.cam.ac.uk, biham@cs.technion.ac.il, lars.knudsen@ii.uib.no,
kelsey@counterpane.com, schneier@counterpane.com, daw@cs.berkeley.edu,
aesround2@nist.gov
Subject: Re: Weakness in Rijndael/Serpent key schedule?

> Is the fact that a Rijndael key can be recovered via the XOR of two Cipher
> Keys and the XOR of the Round Keys important?

I don't think so, no.

> 'Serpent' also seems to have this property. Given the XOR of two keys
> K1,K2 then the XOR of the prekeys w[0..131] could be determined. Now if an
> attacker knows the XOR of {k0,k1, ...,k131 } for the round keys, it is
> straight forward to determine K1 and K2.
> This property doesn't seem relevant to DES.

No, but on the other hand, the exor of two DES cipher keys gives immediately the exor of all round keys, which is probably worse. This is not the case for Rijndael and Serpent.

> I have not been able to create a practical attack based on this analysis.
> How an attacker could get the XOR of the round keys but not the keys
> themselves is certainly a mystery.

Assume you find an algorithm which on input a number of plaintext-ciphertext pairs encrypted under two randomly chosen and unknown keys, returns the exor of the round keys (and/or the master keys).
This algorithm can be used to find a single key.
Run the algorithm but where you choose one of the keys yourself at random.
You can then compute plaintext-ciphertext pairs using the key you know and feed these to your algorithm together with pairs encrypted under a secret key K. Then you get the round keys of key K, and you have broken the cipher.

-Lars

From: "Albert Yang" <albert@achtung.com>
To: AESround2@nist.gov
Date: Mon, 1 Nov 1999 16:55:59 -0800
Subject: Home made tweaks
Priority: normal
X-mailer: Pegasus Mail for Win32 (v3.12a)

First, I think you guys are doing a great job. I don't usually say that to any group with a .gov...

As for the multi-algorithm vs. single debate, here's an offshoot of it.

I like Serpent, it's slow but like a vault, if it doesn't get picked, I still will probably use it in most of my personally developed applications. If Mars or Twofish got picked, they are great, but I can't remember all that, and I am afraid that I would screw up the implementation of it. I seriously doubt that any of the 5 are going to be broken, that's not where the weaknesses are going to be. It's going to be implementation, and that's where I screw up.

So for that reason, I like RC6. I can fit it on a napkin, and still wipe my mouth. Easy to remember, and it has a good family history so I'm a bit more confident in it.

Here's the thing though; I'm thinking, if I need a bit more security, I might just use RC6 and make it 384 bits, and up the number of rounds. It's a cipher I know, it's a cipher I trust, and I can just up the rounds and the key size.

Same with Rijndael. 10 rounds is not enough to make me sleep well at night, so if it got picked, I'd crank it up to 16 or 20 rounds. Double like Lars says is good for me.

So is that something you guys are addressing or considered addressing? The homebrewed stuff. I take AES algorithms, and tweak it up myself? I think comparison of speed is not fair for this reason alone. What if the designers of Serpent had picked only 8 rounds? Then it would be fast. But they picked 32, so it's slow. What if Rijndael was 32 rounds?

Right now, I'm thinking, it doesn't matter which one wins, I will probably use Serpent straight out of the bottle, or RC6 with 32 rounds and maybe a beefed up key, or Rijndael with 20 or 32 rounds.

I can't be alone on this line of thinking I suspect...

Regards,
Albert

Date: Tue, 16 Nov 1999 00:34:37 -0800
From: Ted Goldstein <tedg@ricochet.net>
Reply-To: tedg@transactor.net
X-Mailer: Mozilla 4.07 [en] (Win95; I)
To: AESround2@nist.gov, tgoldstein@brodia.com
Subject: Issues for AES

One of the interesting issues regarding AES is multiple utilitiy. DES has been used for many years in authentication schemes as well as encryption schemes. This becomes especially important in smartcards, where the multiple utility is a necessity.

Ted Goldstein
CTO, Brodia.com

X-Originating-IP: [152.163.204.201]
From: "Matt Robshaw" <matt_robshaw@hotmail.com>
To: AESround2@nist.gov
Subject: comment
Date: Mon, 29 Nov 1999 01:21:34 PST

Dear NIST AES team,

As you will no doubt be aware, Lars Knudsen and Willi Meier have written a paper entitled "Correlations in RC6". Since the posting of this paper to the AES discussion forum there appears to have been little discussion on the implications of this excellent work. We would therefore like to take this opportunity to let you know our own thoughts on this paper.

In our report "The Security of RC6" (which was published on August 20, 1998 and is available via www.rsa.com) we provide a thorough investigation of the security of RC6. We primarily used the techniques of differential and linear cryptanalysis, and while we also considered some other techniques, we felt that linear cryptanalysis would likely provide the most successful avenue for the attacker. In particular we gave a clearly reasoned and conservative estimate that an attack on 16 rounds of RC6 using around 2^{119} known plaintexts should be expected. (See Table 15, on page 51 of the report cited above.)

By vulnerable we assumed that not only could RC6 be distinguished from a random permutation using this amount of plaintext but in fact we followed established cryptanalytic practice in assuming that the full encryption key (or its equivalent) could be recovered! Taking this into account, and feeling that the simplicity of RC6 had allowed us to establish a good intuitive and empirical understanding of the cipher during the design phase, we fixed the number of rounds of RC6 to be 20.

What is very interesting to us is that the results of Knudsen and Meier so closely match our own expectations for the security of RC6. Their innovative extension and generalization of the earlier results of Vaudenay et al. give a remarkably similar result to our own: they have concluded that around 2^{119} plaintexts can be used to attack 15 rounds of RC6. While the attack they outline may not be quite as effective as the attack on RC6 that we already believe exists, it does seem to provide excellent confirmation that our own analysis is broadly in line with that conducted independently by other researchers.

We think that there are still some unanswered questions about the style of statistical analysis used for the estimates in their attack, particularly in its extrapolation from very few rounds to a large number of rounds. We hope that this will be an area for continued research. Indeed, many of the same questions are open for the closely-related technique of linear cryptanalysis on which we based our own estimates for the security of RC6. However we are encouraged that, independently, the two techniques give very similar results and they increase our confidence that both sets of estimates for the strength of RC6 are as accurate as current analytical techniques allow.

We look forward to hearing of any more results on the security of RC6 and we encourage others to explore and analyze the cipher. Certainly we believe that we have been successful with our secondary design rationale after security - that of keeping the cipher simple. Only in this way is it possible to encourage independent research, to facilitate analysis by cryptographers other than the designers, and to provide an accurate estimate of the security offered by a cipher. We feel that this is vitally important when the time allowed for such independent analysis during the second round of the AES schedule is so limited.

Yours sincerely,

Ron Rivest
Matt Robshaw
Lisa Yin

From: Jeffrey Streifling <jss@icrossroads.com>
To: AESround2@nist.gov
Subject: Consideration of RC6
Date: 5 Dec 1999 15:02 -0700

Disclaimer: I have just been investigating RC6; I have no special interest in it.

I've been looking at RC6 as one of the most interesting of the AES candidates. It has rather a peculiar set of advantages and disadvantages in regards to implementation issues. It is an extremely concise algorithm and avoids lookup tables and special cases. It makes use of common instructions in most processors. However, it does use multiplication instructions which are hard to emulate in microcontroller environments. If there is to be only one AES algorithm, the use of the multiply would probably mean that it would have to go based on performance and implementation issues in small environments. (As an engineer familiar with the instruction sets on several microcontrollers, I know I would hate the job of producing a small implementation of this.) Here I assume that all remaining algorithms are reasonably sound cryptographically and that the remaining selection procedure will be based on implementation issues.

RC6 turns out to be very useful, almost uniquely useful, though in several environments where none of the other algorithms can apply.

- Suppose I am given the job of producing a small wrapper program for a security sensitive application. The wrapper is going to be put through a very heavy security audit, and as such, needs to keep the number of lines of code to a minimum. RC6 can be embedded in very small programs, facilitating easy audit ("It's obviously correct!") and testing. Other algorithms will need more lines of code to implement specialized logic and lookup tables. It takes a considerable amount of effort to verify the correctness of implementations of the other algorithms as their correctness is not always obvious.

- Suppose I want to implement a one-time password calculator in a programmable calculator (e.g., an HP 48) to take on the road to access home office facilities through untrusted network terminals (think of "Internet Cafes", truck stop facilities, airport "web booths", libraries, and the like). RC6 can be implemented fairly easily in anything that supports multi-base arithmetic; I don't even want to think about trying to squeeze TWOFISH or SERPENT into these environments, even as much as I like them.

- Suppose I want to implement a quick mechanism for disguising some critical piece of information like a credit card number in a script/interpreted program on a machine that doesn't

already have a library for that purpose. RC6 is the natural winner: 10 lines of code will encrypt, another 10 will decrypt, and another 10 will schedule the key if necessary. It all just goes in line and I'm done. I don't want to think of the tedium of keying a kilobyte worth of hexidecimal constants from a book somewhere.

- Suppose I need to verify my implementation quickly in the presence of some new bug that gets discovered down the road. With RC6, I can quickly check any part of the action by hand. The other algorithms require a bit of fiddling -- quick on a computer, but messy by hand.

- In general, any application where the algorithm needs to be implemented quickly or at low cost, or where the code needs to be heavily scrutinized, or where the expense of maintaining security sensitive lines of code is high will benefit greatly from RC6.

I think that if at all possible, algorithm simplicity should be one of the factors in the AES decision. I personally think that RC6 is a hands-down winner in this arena, and if it is posssible to choose more than one AES candidate for the standard, RC6 should definately be one of the chosen algorithms (assuming that some new theoretical result doesn't break it in the interim...) (Even if there was only to be one algorithm, I would still vote for RC6 for the above reasons, but I could understand why it would face opposition from smart-card software authors.)

As for how the requirements for AES compliance should be stated in the presence of multiple approved algorithms, I think that a distinction should be made between "client compliance" and "server compliance" (change the terminology to suit). Servers (things to whom people need to authenticate, messaging hubs, and any kind of software package where it might be possible to implement multiple algorithms) should be required to implement all algorithms; clients (smart cards, hand-held computers/palmtops, calculators, etc) should have their choice of implementing any one algorithm. Then clients and servers could reliably communicate, as well as two servers. (Any kind node in a messaging system where full-mesh connectivity is desired would automatically be subject to the server requirement, including applications for the Internet). I would like to be able to implement the algorithms in question on a programmable calculator (e.g., for testing or maybe to reduce the number of items in my pocket) while a smart card author would like to be able to cram basic functionality into a 50 cent controller. Different algorithms will be suited to our purposes, and it would be nice to permit as wide a variety of implementations as possible.

I think that at this point, with only five candidates left, that the only reason candidates should be eliminated is for cryptographic weakness. It would be straightforward to implement all five algorithms in any loadable computer software (witness today's multigigabyte hard drives!), which would provide the most flexibility for small devices which need to interact with them. Perhaps algorithms could be eliminated if it could be shown that there could be no environment to which they would be uniquely best suited (i.e., they are redundant). RC6 should definately not be eliminated. Multiple

standardized algorithms will provide insurance against future theoretical breakthroughs which might compromise the security of one, maybe two, but not all algorithms.

Once again, if at all possible, please include RC6 in the final AES standard. A simple cipher that can be implemented almost anywhere would provide valuable future flexibility!

Thank you
Jeffrey Streifling
<jss@icrossroads.com>

Date: Sun, 26 Dec 1999 21:20:58 -0500
From: Eva Bozoki <eva@fortresstech.com>
X-Mailer: Mozilla 4.5 [en] (Win95; U)
X-Accept-Language: en
To: AESRound2@nist.gov
CC: aharon <aharon@fortresstech.com>, dennis <dennis@fortresstech.com>
Subject: Comments on Round 2 White Paper

This is a vendor-oriented response on the question of "How many AES algorithms?" Jim Foti raised in his recent email and which is discussed in the Round-2 Issues White Paper.

I assume that out of the final AES candidates there will be more then one that meets the requirements. More then likely, beyond satisfying the basic security requirements, they will have different advantages making them equally good candidates under different circumstances (efficiency, security, margin tradeoff, hardware/software implementability, etc). For that reason, probably the votes of the committee can not be and will not be unanimous. Rather then choosing 1 out of 2 or more almost equally good candidates, why not allow vendors and their customers to make the choice from a pool of approved algorithms according to how they perceive their needs and preferences? (Naturally, if only one algorithm would satisfy all the requirements, my arguments are not valid.)

I don't think that the use of different algorithm would necessarily cause interoperability problems. There are three scenarios:

1. Each encryption product is manufactured to use one and only one algorithm out of the pool. It is ordered and manufactured for a given need and Network, where all the used crypto products has the same communication protocol and the same encryption algorithm.
2. The User of the encryption product (in the role of the Crypto Officer?) is allowed and able to switch between algorithms, thus the same product is portable from one Network, using algorithm-1, to another Network that uses algorithm-2. Also, it makes it possible and easy for a Company that purchased encryption products to switch from one algorithm to another if their need or their confidence changes.
3. The encryption product can automatically recognize the algorithm in the incoming messages and use the same algorithm, much like the IPSEC protocol does.

In none of the above scenarios is the same message encrypted with more then 1 algorithm, thus approving multiple AES algorithm will not increase the probability of breaking any one of them.

Finally, saying that public confidence would be shaken if any of the approved AES algorithms would be broken sounds silly. What if the single AES algorithm is broken? In that case not only the public confidence, but also much more would be broken.

.Eva Bozoki
PATCO
Scientific Advisor to Fortress Technologies

Date: Mon, 27 Dec 1999 16:54:12 +0100
From: Eric Boesch <ebo@dannet.dk>
X-Mailer: Mozilla 4.7 [en] (WinNT; I)
X-Accept-Language: en
To: AESround2@nist.gov
Subject: Modes of operation

1. PCBC is desirable for fast verification of ciphertext integrity.

The need for checksums to verify message integrity is commonplace, even in unencrypted network applications. CBC and CFB MACs authenticate their plaintext only -- they do not authenticate their ciphertext. Using PCBC allows you to simultaneously generate ciphertext and a MAC that can be applied to the ciphertext, at little extra cost.

PCBC's behavior is simple, unforgiving, and often desirable: all errors propagate forward indefinitely.

2. Why output feedback mode instead of counter mode?

Output feedback involves XORing the plaintext with an independent codestream E(IV),E(E(IV)),E(E(E(IV))),.... If the block size is 128 bits and E() is a random one-to-one mapping, then you can expect, by the birthday paradox rules, to have your codestream repeat after about 2^64 blocks on average. If the encryption function is secure, then under no circumstances does output feedback appear to offer significant advantages over an appropriate use of counter mode, where the plaintext is XORed with E(IV),E(IV+1),E(IV+2),....

X-Sender: schlafly@pop.mindspring.com
X-Mailer: QUALCOMM Windows Eudora Light Version 3.0.6 (32)
Date: Tue, 11 Jan 2000 04:37:20 -0800
To: AESround2@nist.gov
From: Roger Schlafly <real@ieee.org>
Subject: AES round 2 comment

Comment on NIST AES process

There have been rumors that NIST is considering multiple winners for the AES. I think this is a very bad idea.

When NIST announced the DES about 25 years ago, it was radical. It introduced to the public the idea that a cipher algorithm could be public and standardized, while all the secrecy was in the keys.

While DES has been intensely attacked over the years, it has held up rather well. It delivers the security it promises, more or less. NIST and IBM are to be congratulated for their outstanding contribution to the science of cryptology, for convincing both academics and the public that cryptography can be an open process, and for a huge positive impact on computer security.

Now that DES needs to be replaced with AES, NIST has taken the process one step farther by using an open competition to find a state-of-the-art cipher. This is admirable. The five finalists are all outstanding, and any one of them should make for a great AES (provided the various concerns are addressed, of course).

But now I am troubled by the possibility that NIST may weaken in its resolve in the face of political pressure. Some people seem to have ideological or business reasons for opposing a standard. Or maybe they still have not accepted the fact a cipher can have an open specification and still be secure.

If NIST chooses multiple winners, then it will be a failure of the AES process. People will assume that no cipher was fully satisfactory, or that NIST could not make a decision. If that happens, a lot of people will continue to use DES or triple-DES. Interoperability will become difficult or impossible. Many resources will be wasted by people who want to implement each algorithm in order to get full compliance. Others will find full compliance too expensive, and end up not complying at all. Some people will think that they have to implement some sort of algorithm selection protocol in order to make full use of AES. While there are some theoretical arguments that such a selection protocol can increase security, it is much more likely that flawed protocols will decrease security. Others will be encouraged to iterate or nest the ciphers on the theory that if one cipher is not good enough for NIST then it is not good enough for them either.

The ANSI X9F1 committee had a resolution in favor of NIST considering future resilience in the AES process, whatever that means. Attempts to gain support for a motion to recommend multiple AES winners apparently failed. I would not assume that the ANSI X9F1 committee wants multiple AES winners unless it actually passes a motion that

recommends multiple AES winners. I would have voted in favor of considering future resiliency, but against adopting multiple standards.

So why would there be resistance to a single AES winner? Here is the opposition that comes to mind.

1. Those who naively think that more ciphers means more security.

2. Those who are ideologically opposed to standards.

3. Those with business interests contrary to a single AES standard.

4. Those who think having both US and foreign designed winners would be more politically acceptable.

It is the large key space that gives the diversity of ciphers. If we needed more ciphers, then you could add a couple of key bits. But 256 key bits is plenty already. If political considerations warrant the choice of a second cipher, then perhaps one could be designated a "runner-up" so that it will be available but not obligatory.

Therefore I strongly recommend sticking to the vision of an encryption standard, and resisting efforts to dilute and weaken it with wishy-washy decisions.

Roger Schlafly

From: Chen Lidong-LCHEN1 <LCHEN1@email.mot.com>
To: '"AESRound2@nist.gov"' <AESRound2@nist.gov>
Subject: Re: Request related to FIPS 140-1 and the AES
Date: Fri, 14 Jan 2000 15:26:46 -0600
X-Mailer: Internet Mail Service (5.5.2650.21)

Jim:

We prefer that one algorithm be selected as the final AES algorithm for the following reasons:

1. It would be best to put efforts into evaluating the security of one algorithm. If more than one algorithm is selected, then the analysis efforts would not be as focused.

2. If multiple AES algorithms are selected, then security issues related to algorithm negotiation protocols will need to be considered.

3. It is more cost-effective to implement a single encryption algorithm than to implement multiple algorithms. Multiple algorithms would require larger, more costly circuits to support multiple algorithms. These circuits would also be more complex, thus reducing production yields and increasing testing requirements. Complex circuits are also more prone to bugs, which can potentially lead to other security issues.

4. Multiple AES algorithms will lead to interoperability issues between different systems. These issues will require further standardization efforts, which may lead to relatively long product deployment delays.

5. Customer confidence will be diminished by choosing multiple AES algorithms. It will be difficult to explain why there are multiple AES algorithms. Also, customers will be bothered to decide which algorithm is better.

6. Choosing multiple AES algorithms, might increase the possibility of intellectual property issues. The more algorithms that are selected, the more likely one of them will have intellectual property issues that will surface at a later date.

Based on the above noted issues, we recommend that NIST select a single algorithm for the AES algorithm.

Regards,

Lily Chen
Motorola Inc.

1501 W. Shure Dr.
Arlington Heights, IL 60004
(847) 632- 3033

Date: Mon, 17 Jan 2000 16:51:36 +0900 (JST)
From: Kaneko Yasuyoshi <kaneko@citron.yokohama.tao.go.jp>
Reply-To: Kaneko Yasuyoshi <kaneko@citron.yokohama.tao.go.jp>
Subject: Some comment
To: AESRound2@nist.gov
X-Mailer: dtmail 1.2.1 CDE Version 1.2.1 SunOS 5.6 sun4u sparc

Dear AES officers

I am Yasuyoshi Kaneko belonging to TAO in Japan.
May I congratulate on the third AES conference, and I hope the final AES algorithm will be selected without accident.
I also have watched the AES selecting process as many participants in the world and now I feel there are some points which NIST had better reconsider and make respond.

1. Estimating Process
The estimating process of AES seems to be more complicated compared with other contests or other engineering. Because cipher designers and cryptanalysts are not so completely distinguished that a person may be sometimes a good or bud designer but other times a bad or good cryptanalyst. Thus the estimation results, which were gotten by, many cryptanalyst seems to be a jumble of wheat and tares. After all judging and deciding is due to NIST, so I ask NIST to continue to be open to present his judging process and conclusion.

2. Simple design criteria
This simple design criterion was presented before the first AES conference by NIST, and most of all designers of AES candidate seem to try to satisfy these criteria. However this idea of simple is against the security because a simple design structure of cipher may give many chance of cryptanalysis and this cryptanalysis results give a weak image to these ciphers without breaking. The idea of simple is difficult to define and it is difficult to judge which cipher is simple or not (With this reason simple design criteria has been not excitedly discussed during AES estimating process). A correct expression may be that clear and united design principle AES algorithm should have. If final selected AES will seem to be not simple what kind of explanation will NIST give?

3. Lifetime criteria
I think, as many people think, the lifetime of AES is the most important thing, which NIST consider and take the responsibility to. NIST seems to consider that the lifetime of AES is 20-30 years, without any clear comment on the maintenance plan of AES specification. Maybe there are many threats, which force AES to change itself. Simple examples are given as follows.
a. Someone finds a week point, which seems to give a chance to break the AES by a new attack.
b. Someone finds new computing processor or technique, which give an enough ability to break AES by ciphertext-only attack.

c. Someone finds new design technique for keeping security and which technique seems to be enough attractive that many engineers abandon to use AES.

Anyway, when the reliability of AES is fallen the most damage is given to users and many companies. Maybe royalty free sounds good for every engineer who consider the design of security system, however if there is no guarantee for long-time using or some guidance for using the specification of AES, using AES seems to be very risky selecting for engineers and his companies. I think that there are 2 things, which NIST should consider and officially announce the guidance of coping with.

1) Making a usual maintenance plane about key-length and block-size to present guidance for using the many kind of key-length and block-size.

2) Making an unusual maintenance plane especially considering the cases a.

The most important thing is 2), because the specification of AES may be dramatically changed and users of AES may also change the security system as soon as possible.

Maybe one practical method of solving risky status is offering some AES algorithms as some people said, however I think many cipher algorithms will give much pleasure to cryptanalyst and attacker so that the results of cryptanalysis contribute to shorten the lifetime of these AES. Moreover if many kind of standard algorithms are used in each security system, interface of each system becomes to be more complicated and user-interface also becomes to be no good.

I think other method is increasing the number of round with an expectation that the speed of hardware and software will be enough faster than now. Maybe this method of increasing the round number does not seem to be attractive way but this method is the most certain way to keep security of ciphers. Because almost all cryptanalysis depend on the round number and if the round number increases enough anyone fail to attack any more.

(I think that if trying to use a block cipher for a long period, as an old proverb says,

There is no royal road to block ciphers. The only thing that we can ask is not "Is AES secure enough?" but "How many round of AES is secure enough for now?"

The only problem is changing also the algorithm of key scheduling, so the specification of key scheduling which makes it possible to increase the number of rounds should be given. Fortunately the five finalists of Round2 seems to satisfy the flexibility of changing the round number and the key-scheduling, so anyway my idea is possible to realize. Moreover it is desirable that the method of this changing specification is parameterized to easily implement and change its algorithm. It is also need that the number of round should be periodically (every several years) reexamined to judge whether the round-number of AES should be changed or not.

I hope that NIST will consider and investigate these three points and will make a public of his decision at the third AES conference. With every best wish for success for AES3.

Yasuyoshi Kaneko,
Telecommunications Advancement Organization of Japan,
E-mail: kaneko@yokohama.tao.go.jp

From: "Costantini, Frank @ CSE" <fcostant@mail.cse.l-3com.com>
To: "'AESround2@nist.gov'" <AESround2@nist.gov>
Cc: "Carter, Matthew @ CSE" <matthew.carter@l-3com.com>,
"Kozak, Taras @ CSE" <tkozak@mail.cse.l-3com.com>,
"McGrogan, Chip @ CSE" <emcgrogan@mail.cse.l-3com.com>,
"Holland, Robert @ CSE" <rholland@mail.cse.l-3com.com>
Subject: AES modes of operation
Date: Wed, 26 Jan 2000 16:01:32 -0500
X-Mailer: Internet Mail Service (5.5.2650.21)

AES Evaluation Group:

While I do not have any specific comments on any of the particular AES candidates, I would like to propose that NIST evaluate the security of the proposed algorithms when employed using counter-mode cryptographic operation. Counter-mode has advantages over OFB, CBC, and ECB modes for high-speed packet-based applications (like ATM), and as well as packet based communications over bandwidth-restricted channels. This mode requires little synchronization overhead and allows the keystream to be calculated in advance of the plaintext (for transmit) and ciphertext (for receive) becoming available. Furthermore, I would like to propose that the counter mode of operation be included in any "modes of operation" standard that is produced as part of the AES process.

Thank you for your consideration.

Respectfully,

Frank Costantini
L-3 Communications
Communication Systems-East
1 Federal Street, AE-3C, Camden, NJ 08103
*856-338-3480 Fax: 856-338-3150
email: frank.costantini@L-3COM.com

Reply-To: <emcgrogan@mail.cse.L-3COM.com>
From: "Chip McGrogan" <chip.mcgrogan@L-3COM.com>
To: "'Costantini, Frank @ CSE'" <fcostant@mail.cse.L-3COM.com>,
<AESround2@nist.gov>
Cc: "'Carter, Matthew @ CSE'" <matthew.carter@L-3COM.com>,
"'Kozak, Taras @ CSE'" <tkozak@mail.cse.L-3COM.com>,
"'McGrogan, Chip @ CSE'" <emcgrogan@mail.cse.L-3COM.com>,
"'Holland, Robert @ CSE'" <rholland@mail.cse.L-3COM.com>
Subject: RE: AES modes of operation
Date: Wed, 26 Jan 2000 19:03:53 -0500
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook 8.5, Build 4.71.2377.0
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2314.1300

Frank: Great! Keep me up to date on how NIST reacts.

BTW you forgot to mention one big advantage of Counter Mode, particularly for high data rate applications like ATM - - Since there is no feedback of previous information, parallel (concurrent) engines can be used to increase the data rate above that of a single engine. A 48-byte ATM cell payload can be encrypted by three parallel engines, each encrypting a 16-byte portion of the cell payload in a manner that is interoperable with an implementation using a single engine.

Keep up the initiative!

Chip

From: "Nicholas C. Weaver" <nweaver@CS.Berkeley.EDU>
Subject: Importance of subkey generation time...
To: aesround2@nist.gov
Date: Fri, 18 Feb 2000 14:48:47 -0800 (PST)
X-Mailer: ELM [version 2.5 PL0]

For hardware implementations, subkey generation structures may make a huge difference. There are several applications which would use cryptographic hardware (such as an encrypted backbone packet router, or an encrypting server which may service hundreds or thousands of simultaneous, independently keyed connections) where the key may be different on almost a block-by-block basis. Furthermore, depending on the application, the memory bandwidth requirements for storing and accessing expanded subkeys may be prohibitive, so subkeys would have to be generated on the fly.

The ability to do fast subkey setup really varies from cipher to cipher. MARS and RC6 both have loop carried dependencies and need to traverse the loop multiple times, which implies that one can't build a pipeline to generate subkeys. This also makes it expensive/difficult to do subkey setup concurrent with encryption, as a pipeline capable of encrypting N independent blocks simultaneously would require N copies of the subkey generation hardware to encrypt each block with a unique key.

Rijndael and Serpent are pipelineable but the whole set of subkeys will need to be buffered for decryption because of the reverse order required for decryption subkeys. Thus, for a N block pipeline, this would require the subkey storage * N buffer space, but only a single copy of the subkey generation hardware to do decryption. For encryption, such significant buffering is not required.

Twofish, from a hardware implementor's viewpoint, has the nicest key schedule, since it can be quickly pipelined and, with the subkeys for each round being independently generateable, allows both encryption and decryption pipelines to quickly change subkeys. One would have a pipeline to generate the S-box subkeys, and a second to generate each round's subkeys. This would allow a single copy of the subkey generation hardware to support almost full rate encryption OR decryption, with each block using an independent key.

On another note, as a potential hardware designer, I'd be highly reluctant to support the idea of multiple "winners". Although I defiantly prefer some implementations (Twofish, Rijndael) over others (RC6, Serpent, MARS), there is a significant cost in hardware in having to support multiple ciphers. As a designer, I'd rather support a single, less elegant cipher over two or three different algorithms.

Undoubtedly multiple final selections will result in protocols which involve supporting all candidates. Although not a significant problem from a software point of view, this represents a doubling of the cost from a hardware implementers view if he wishes to support such protocols.

Even if a single winner is less amenable to hardware then one of a group of winners, the extra cost of having to support multiple winners could instead be spent on doubling the hardware available (and doubling the hardware performance) of a single winner.

AES shouldn't suffer from the problem inherent in multiple standards of: "That's the wonderful thing about standards, there are so many to choose from"

--
Nicholas C. Weaver nweaver@cs.berkeley.edu

To: aesround2@nist.gov
cc: craigc@pictel.com
Subject: Comment on VLIW & Instruction Level Parallelism
Reply-To: Craig Partridge <craig@aland.bbn.com>
From: Craig Partridge <craig@aland.bbn.com>
Date: Tue, 14 Mar 2000 10:20:43 -0500
Sender: craig@aland.bbn.com

Hi:

I read Craig Clapp's paper on Instruction-Level Parallelism with interest and had a few comments.

First, I think the Instruction characteristics, while valid for today's slower processors, will not be true soon. Faster VLIW CPUs are leading to more variable load delays and often multi-instruction cycle latencies for more classes of instructions. For instance, on the DEC Alpha a LOAD can take between 2 and 40 some cycles depending on where the data is located and the speed of the external memory. Integer multiply cycles are also growing (as long as 10 to 20 cycles). And, interestingly enough, barrel roller operations (BYTE Extract and shifts) sometimes take more than one cycle.

This suggests that we should think less in terms of particular cycle counts and more in terms of sums of instruction types along critical paths.

Craig Partridge
Chief Scientist, BBN Technologies

E-mail: craig@aland.bbn.com or craig@bbn.com

To: aesround2@nist.gov
cc: Ross.Anderson@cl.cam.ac.uk
cc: biham@cs.technion.ac.il
cc: lars@knudsen.ii.uib.no
cc: brian.gladman@btinternet.com
Subject: comment on Serpent optimized code
Reply-To: Craig Partridge <craig@aland.bbn.com>
From: Craig Partridge <craig@aland.bbn.com>
Date: Tue, 14 Mar 2000 10:39:06 -0500
Sender: craig@aland.bbn.com

Hi folks:

I've been perusing the various AES "optimized" C code solutions and noting that several of them appear to be under-optimized -- that there's space to make them run faster.

This note is to report on some limited success with the Serpent implementation. Note that Serpent appears to be very hard to optimize because it is pretty simple -- a series of simple operations. Very little room for instruction level parallelism or pre-computation. But I've gotten what appears to be a 15% to 20% per-round improvement in performance in Round 0 (which is enough to report on and the results would appear to apply to other rounds).

There are two issues:

- The optimized implementation submitted to NIST uses a huge number of temporary registers and at least some compilers aren't smart enough to realize when the temporaries can be recycled. The code is also not written to minimize register usage. It uses 14 registers when I've been able to rewrite the code to use only 5 registers for round 0. I've also taken Brian Gladman's optimized round 0 code and tuned it to use 6 registers. You get improved performance on the Pentium and the DEC Alpha.

  The utility of this improvement is not just fewer registers in software -- it can mean less data carried along the pipeline in harder.

- The optimized implementation treats three parts of a round, namely the round (RND macros), the transform (another macro) and keying (another macro) as distinct phases. For instance, you don't do the transform until the round is finished. However, you can get a lot of instruction level parallelism by merging the transform and round. Typically some results of the round are available only halfway through a round and they're often enough for the transform stage to start doing useful computation. RND00 appears particularly amenable to this optimization but other rounds should benefit too. For Round 0, this optimization gets a modest win on the Pentium (subject to register interactions) and a big win on the Alpha (which has far more available registers).

> Again, this is not just a software issue -- if you're building hardware the logical thing to do is treat the round and transform as one combined process.

I hope this is useful,

Craig Partridge
Chief Scientist, BBN Technologies

E-mail: craig@aland.bbn.com or craig@bbn.com

Date: Thu, 16 Mar 2000 14:41:26 -0800
From: August Zajonc <azajonc@POMONA.EDU>
Subject: Twofish
To: AESround2@nist.gov
X-MIMEOLE: Produced By Microsoft MimeOLE V5.00.2919.6700
X-Mailer: Microsoft Outlook Express 5.00.2919.6700
X-MSMail-priority: Normal

Developed with care,
usable anywhere.

From: "Manfred Spraul" <manfreds@colorfullife.com>
To: <AESRound2@nist.gov>
Subject: public comment
Date: Mon, 20 Mar 2000 13:45:30 +0100
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.00.2919.6700
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700

a few short notes from a non-cryptographer:

- AES supports key sizes of 128 and 192 bits, but SHA-1 and RIPE-MD 160 produce 160 bit keys. Could you define a standard way how to fold/expand the 160-bits into 128/192 bits? Please remember that many non-cryptographers will use AES.

- I would prefer if one algorithm is choosen as the AES [except as below]: This should increase interoperability and reduce costs.

- Even if Moore's law remains valid for the complete century, 256 bit keys won't be broken during this century by brute force. And there is no guarantee that throwing more bits at an alogorithm actually increases the security. Please consider a Double-AES for the 256-bit key length [e.g split the key into 2 128-bit blocks, and encrypt with 2 different algorithms] Noone will use 256-bit in smart-cards, so memory/size constraints will be less important.

--
Manfred Spraul

From: sesmail@neptune.calstatela.edu
Date: Tue, 21 Mar 2000 20:50:21 -0800 (PST)
X-Authentication-Warning: webmail2.calstatela.edu: nobody set sender to sesmail@neptune using -f
To: webmaster@ams.org, webmaster@entropia.com, info@perfsci.com,
webmaster@eff.org, AESround2@nist.gov
Reply-To: sesmail@neptune.calstatela.edu
User-Agent: IMP/PHP3 Imap webMail Program 2.0.11
Sender: sesmail@neptune.calstatela.edu
X-Originating-IP: 208.149.1.14

Hi this is Simon Esmaili.

I recently read on the internet that quantum computers might make encryption techniques based on the multiplication of two huge prime numbers obsolete.

I know of a simple huge number encryption technique. It may or may not be adequate though. I wouldn't know because I'm fairly ignorant when it comes to encryption.

Step 1: Pick a code.

Step 2: Encode the message.

Step 3: Divide the encoded message into pieces of length n. If the length of the last piece is less than n, then add encoded random characters until the last piece is also of length n (it's probably not a good idea to add encoded blank characters in lieu of encoded random characters).

Step 4: Pick an n by n matrix where each entry in the matrix is a natural number.

Step 5: Multiply each piece/vector of length n by the matrix.

For example,
if
encoded message = "a b c d e f"
n = 3
first row of matrix = "g h i"
second row of matrix = "j k l"
third row of matrix = "m o p"
then
matrix-multiplied encoded message = "ag+bh+ci aj+bk+cl
am+bo+cp dg+eh+fi dj+ek+fl dm+eo+fp".

The matrix acts like a combination lock. If n = 20 and each entry in the matrix is a natural number between 1 and 10^4 inclusive, then there are (10^4)^(20^2) = 10^1600 such matrices. For each matrix, the inverse must be found and multiplied with the intercepted message.

Reply-To: <Gideon@dgsciences.com>
From: "Gideon" <Gideon@dgsciences.com>
To: <AESround2@nist.gov>
Subject: Formalizing a Response to Allegation and Eventuality of breakdown.
Date: Sun, 9 Apr 2000 14:15:59 -0400
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2314.1300

Dear Comments Collectors:

Please add the following comments to the pile:

AES should address both the allegation and the eventuality of any attack which would considerably accelerate the brute force key search.

Allegation: How to handle raised allegations that the AES has been "broken". What procedure should be in place to check out those allegations. Who will check it out, how fast should a response come, and how and where to announce the finding.

Eventuality: In the event that it becomes clear that a productive attack has been substantiated, it is necessary to invoke a damage audit procedure, damage control, if possible, and a credible backup plan.

Thank you for taking these points into account.

Gideon Samid
gideon@dgsciences.com

=-
D&G Sciences -- Virginia Technology Corporation
P.O.Box 1022; 6867 Elm St. Suite 200
McLean Virginia 22101-1022 * U.S.A.
info@dgsciences.com
Ph: 703.385.4144 * Fax: 703.591.0847
=-

From: "Hughes, James P. (SNBG)" <HugheJP@nsc-bridge.network.com>
To: "'AESround2@nist.gov'" <AESround2@nist.gov>
Subject: AES Comments.
Date: Sat, 15 Apr 2000 21:16:52 -0500
X-Mailer: Internet Mail Service (5.5.2650.21)

NIST AES Evaluation comments from Storage Technology Corporation.
Jim Hughes, StorageTek Fellow

Please consider my comments on the following areas: A single winner, FPGA implementation, and a Modes.

- Single winner

I would like to suggest that primary and backup winners will effectively result both algorithms being required by the market, and once implemented will double the vendor's liability from a "patent attack".

1. Because customers will perceive the implementation of both the primary and backup algorithms as a differentiator among competitors, implementing both algorithms will become mandatory. I believe this will be the case that no matter how the standard states that the backup is not mandatory. (I agree that all standards that implement AES should implement algorithm negotiation, and that algorithm agility should be mandated and part of the certification.)

2. Since vendors will be forced by the market to manufacture both the primary and backup algorithms, they will be twice as vulnerable to a patent attack.

- FPGA

StorageTek makes extensive use of Field Programmable Gate Arrays and wishes that the standard algorithm provides the ability to be implemented in a key agile registered pipeline mode. That is, that the algorithm can be effectively implemented in the largest currently available FPGA as a registered pipeline and that the key be changed at least at every 3 blocks without requiring draining the pipe (a requirement to run ATM traffic).

- Modes

To allow use of high performance encryption, StorageTek would like NIST to acknowledge "Counter Mode" as a valid method for AES. In addition, StorageTek would like to highlight security concerns of both Counter mode and CBC-n (n>1) if cryptographic data integrity is not also present.

Thanks

Jim Hughes
StorageTek Fellow
Storage Technology Corporation
Jim@network.com

To: AESround2@nist.gov
Subject: My criterion...
Date: Mon, 17 Apr 2000 20:51:49 -0400
From: Robin Lee Powell <rlpowell@calum.csclub.uwaterloo.ca>

My sole criterion that I would like to see in the AES standard (besides basic security considerations, of course, but I trust NIST can handle those) is that the algorithm be available for public use free of charge. No patents, no licensing, no nothing.

I don't think that's too much to ask, and it's all I really want.

-Robin

--
http://www.csclub.uwaterloo.ca/~rlpowell/ BTW, I'm male, honest.
As a member of the Hans Solo School of Action Before Thought, Welcome, You've Got Male.

From: "Nicholas C. Weaver" <nweaver@CS.Berkeley.EDU>
Subject: Followups to AES conference and my paper
To: aesround2@nist.gov
Date: Mon, 17 Apr 2000 18:13:02 -0700 (PDT)
X-Mailer: ELM [version 2.5 PL0]

This is designed as an addendum, clarification, correction, and expansion on my paper in AES3 ("An analysis of the AES candidates ameniability to FPGA implementation") and other issues from the AES 3 conference.

My opinion has changed slightly. I can now comfortably endorse Serpent as well as Rijndael and Twofish. Although there are some tradeoffs between the three, they are significantly superior to RC6 and FAR superior to MARS from a hardware viewpoint.

A three sided fair coin to decide between Rijndael, Serpent, and Twofish would produce an excellent AES winner.

Only a single winner (and possible backup) should be chosen.

Now for more detailed comments:

When considering hardware performance, one should not care about bandwidth (ECB mode). No matter what algorithm is involved, ANY desirable ECB bandwidth can be achieved by simply devoting suitable hardware resources to the problem at hand.

Instead, the three factors which should be considered are encryption latency, subkey agility, and the ability to construct compact implementations.

Latency, the time to encrypt a single block, dominates performance in CFB encryption. If a paper doesn't discuss latency, CFB encryption bandwidth numbers can be used to classify the algorithms.

For the candidates, Rijndael has an excellent latency, followed closely by Serpent and then Twofish. Even with doubling the number of rounds, Rijndael would still be in the same class, just on the other end of the scale.

RC6 is significantly higher latency, as shown in all the hardware papers presented. This is because multiplication and rotation are comparatively expensive in hardware when compared to the fixed S-boxes of Rijndael and Serpent, and the logic to implement the variable S-box in Twofish.

Finally, all AES hardware papers showed that MARS has an unacceptably high latency. Even the best MARS implementation, by IBM, in a .18uM process, has worst case CFB encryption bandwidth of <500 Mb/s. An FPGA implementation of Rijndael or Serpent should easily reach worst case CFB bandwidth of > 250 Mb/s in a .25 uM process! Using a $150 part (Virtex 300) available today.

Similarly, there appears to be a consensus that subkey agility (the ability to quickly generate both encryption and decryption subkeys) is necessary for at least some important applications (IPsec and other encrypted-packet routers).

Note that subkey agility also affects smartcard implementations, as an agile subkey setup allows for subkey generation to occur concurrently with encryption without taking a large amount of memory to compute the subkeys into.

Twofish obviously has a completely agile subkey generation scheme, since the subkeys are generated independently.

Serpent, as it was pointed out by the creators, can have the subkey generation run in reverse, by starting with the last two elements and running though a reverse pipeline. Generating the last elements could be done directly, but this computation is probably too complicated in hardware.

Instead, the last two elements could be generated once and then stored. This imposes the subkey generation cost once, but thereafter it can be hidden from the application time. It increases storage requirements by 256b per subkey, but does not increase the bandwidth required, because the endpoints would be fetched when decryption is desired, instead of the normal encryption key.

Rijndael has an identical property, although the storage increase is the size of the key used, (128b, 192b, or 256b).

I don't believe the 3 pass trick (store the A and B states for the 2nd and 3rd pass through the array) for RC6 actually works unless the L array has the same number of elements as the S array.

MARS subkey generation is atrocious from a agility viewpoint. Also, the implementation complexity is absurdly difficult.

Although serpent does have the highest initial requirements if performance is desired, it was pointed out to me that it has the lowest cost if performance is NOT a concern, by using one copy of each S-box, a selectable XOR tree, and 4 nibble-loadable shift registers.

Overall, Rijndael, Serpent, and Twofish are all excellent candidates from a hardware perspective. RC6 is a poor candidate, due to its intrinsic latency and lack of subkey agility. MARS is simply atrocious, due to latency and subkey generation issues.

Other Issues: Number of winners.

The general cost of implementing multiple algorithms can best be described as the square of the number of algorithms, due to additional design complexity, verification, silicon costs, and the potential of a successful IP attack.

Furthermore, it does not increase the security of the system. If any of the winners is broken, this would be a significant disaster. Instead, the better solution is "put all your

eggs in one bank vault and GUARD THAT VAULT". We should be confident in our choice, confident to choose a single winner.

Finally, the "Space Probe" Scenario is facetious. We have a long history of flexible software and field updates (Voyager had a bad bit in memory which was programmed around, and the OS of the Mars Rover was updated on mars). With current high density, radiation hard FPGAs, one can design hardware for such environments which is field updatable if necessary.

A winner and a backup is acceptable, as the backup will undoubtedly only be implemented if it is activated.

The panel session of the finalists brought up the recommendation of a 256b standard key length, with 128 and 192b keys being in the spec but "use at your own peril". Although this increases the cost of key storage and management, it does not affect subkey cost and is an idea I would endorse as a potential implementer.

Also, the suggestion was made of having Rijndael's number of rounds increased. Although this would hurt performance, it would still be acceptable in hardware.

Aside: This is a note for those who may be implementing the AES winner in hardware, and an element to consider when evaluating the hardware reports on the candidates. (Keywords: AES hardware implementation techniques)

When implementing a hardware design for one of the AES candidates, one should follow this technique for implementing any candidate. This will maximize the bandwidth, give nearly minimal latency, and the minimum area required to get to the "best" area/performance point.

One should implement a single round (single super round for serpent) and pipeline it internally to run at the desired cycle time. (Formally, this is a C-slow technique, also called inner round pipelining).

This only adds a small amount of latency when compared to an unpipelined single round (limited solely to partitioning effects and the setup and hold time of the flip-flops) a small amount of area (the pipeline registers, which may be free depending on the implementation technology), while greatly improving bandwidth. If CFB is essential and the only concern, then simply remove the internal pipeline registers.

If greater bandwidth is needed, simply duplicate this unit. This gives a linear increase in bandwidth, and the ability to match the desired bandwidth to the optimal silicon area.

NEVER produce an unrolled pipeline. It offers negligible performance benefits (the only improvement would be a slightly lower latency for a fully unrolled but unpipelined approach) and only a very small area benefit (a little less control logic and subkey storage logic). Furthermore, you lose the ability to produce an exact match to the system requirements.

Finally, for evaluation purposes, it offers an increase in performance which is essentially an exact multiple of the increased area requirement, so it conveys no additional information for the evaluation process.

Summary: Single winner. Rijndael, Serpent, Twofish good. RC6 bad. MARS horrible.

And hardware implementers: pipeline within a round if you want bandwidth!

--
Nicholas C. Weaver nweaver@cs.berkeley.edu

To: AESround2@nist.gov
Subject: Why I feel Twofish should be the AES
From: Mark Atwood <mra@pobox.com>
Date: 17 Apr 2000 18:31:46 -0700
Lines: 17
X-Mailer: Gnus v5.5/Emacs 20.3

I support TwoFish to be selected as the AES. It's flexablity in implementation, in software, hardware, and in CHEAP hardware is key, I feel, to making the AES truely ubiquious, in all communication applications.

We are about to enter an era of both exceptionally cheap lowpower wireless devices, and wireless communications *demands* good encryption, and much much faster "landline" bit pipes, which demands *fast* encryption.

I feel the Twofish fills those two niches very well, and while the other contenders may be good at one or the other, they are not good at both, and at software based implementations as well.

-- Mark Atwood
Seattle, WA
206-781-8048

From: Hironobu SUZUKI <hironobu@h2np.net>
To: AESRound2@nist.gov
Reply-To: hironobu@h2np.net
Subject: My comment for AES round 2
Date: Tue, 18 Apr 2000 10:56:47 +0900
Sender: hironobu@mail.h2np.net

In AES3, there were many discussions about "performance" of various algorithms on various CPUs. Some AES developers emphasized good performance on Pentium(pro). Some paper shows performance on specific CPU, sort of 64bit RISC. Aoki and Lipmaa showed how to write fast code with MMX instruction set.

It is true that Serpent which is one of AES finalist is slow on today's 32bit CPU. RC6 and MARS are fast on Pentium Pro. It seems that those algorithm developers target on Penrium Pro.

Now, we must understand that those CPU are today's CPU, not for after days of AES is selected.

Remember why MMX was appeared in CPU market. As far as I knew, what reason of MMX was added to CPU was developed for accelerating multimedia applications and game on desktop computer.

Pentium III has some extensions for streaming data like as jpeg, mp3, mpeg and so on. Apple/Motorola developed G4 velocity engine a.k.a AltiVec technology for same reason. AMD's "3D! NOW" also. Those CPU instruction set and CPU structure was added by market requirement.

After days of AES is selected, each CPU companies will develop and add new instruction for AES algorithm by market requirement, a matter of security. They must say "AES ready!", "AES NOW!" , "AES Velocity!". Also AES algorithm will be integrated into chipset like a Intel 810 chipset.

I'd like say that we DO NOT overemphasize about performance on today's CPU and must focus on security for next century because AES must be selected for 21st century.

Thus, from my point of view, SERPENT is a reasonable algorithm for AES.

--hironobu

Hironobu SUZUKI
Independent Software Consultant
hironobu@h2np.net
URL http://h2np.net

Date: Tue, 18 Apr 2000 07:47:37 -0500
From: Frank <fhburkha@blue-bird.com>
Reply-To: fhburkha@blue-bird.com
Organization: Blue Bird Body Company
X-Mailer: Mozilla 3.01 (Win95; I)
To: AESRound2@nist.gov
CC: fhburkha@blue-bird.com
Subject: AES

I have been following the search for the new AES for several years, and feel it is time to put in my two cents worth.

While I am not very knowledgeable in the area of cryptography, I do use it in several of my programs. I must trust that all 5 candidates are cryptographically secure.

I am writing to you to express my concern about the implementation of AES. Whatever algorithm you choose needs to be fairly fast on all platforms, in hardware and software. To elevate either hardware or software over the other would unduly restrict the choices available to developers. There are (and will continue to be) legitimate needs for both implementations. In hardware, I am speaking not only about smart cards that can be produced (profitably) in the 10 cent range, but about highly specialized plugin cards for computers.

You must choose one and only one AES. To take the easy way out, and choose multiple winners would cause severe fragmentation in the implementation. Interoperability would become difficult or impossible. Lots of money and time would be spent trying to insure that all the AES algorithms are implemented correctly by the developers. This is something you can not waffle on.

I also believe the code for the complete optimal instillation of AES for all common platforms should be patent and royalty free. I would like to see it distributed with a GNU copyleft license, if at all possible.

--
Frank Burkhardt

The above comments are mine, and mine alone.
They may or may not be supported by the company I work for.

Date: Tue, 18 Apr 2000 06:08:59 -0600
From: Rob Neal <rob.neal@lmco.com>
Subject: AES round 2 finalist comments
To: AESround2@nist.gov
Reply-to: rob.neal@lmco.com
X-Mailer: Mozilla 4.7 [en] (WinNT; U)
X-Accept-Language: en

Hello,

I understand this is the e-mail address to send comments on the algorithms being considered for selection as the AES.

I would like to recommend that the algorithm known as 'TWOFISH' be selected.

It would seem to present the best balance between performance and security, and be the best suited for the intended use.

Thank you,

Rob Neal
rob.neal@lmco.com

Sender: Anne.Anderson@East.Sun.COM
Date: Tue, 18 Apr 2000 10:26:04 -0400
From: ANNE ANDERSON <ANNE.ANDERSON@Sun.COM>
Reply-To: aha@acm.org
Organization: Sun Microsystems Laboratories
X-Mailer: Mozilla 4.7 [en] (X11; U; SunOS 5.7 sun4u)
X-Accept-Language: en
To: AESround2@nist.gov
CC: aha@acm.org
Subject: Comment on AES Round 2

Speaking for myself as an implementor of protocols that use cryptography (PKIX, IPSec), and not as an employee of Sun Microsystems:

Interoperability is very important to internet security. If two platforms need to communicate over the internet, but have no cryptographic algorithm in common, they will not be able to communicate securely. Many platforms (smart cards, small devices) do not have room for more than one algorithm, and I expect such devices to be a larger part of the market for security as time passes.

For these reasons, I highly recommend choosing ONE AES algorithm. It should be one that does not have unacceptable performance on smart cards, in software, or in hardware, but need not be the very fastest.

Anne Anderson
--
Anne H. Anderson Email: aha@acm.org
Sun Microsystems Laboratories
1 Network Drive,UBUR02-311 Tel: 781/442-0928
Burlington, MA 01803-0902 USA Fax: 781/442-1692

Date: Tue, 18 Apr 2000 11:19:54 -0400
From: Rob Gerlach <Zen@mail.rit.edu>
Subject: AES Comments...
To: AESround2@nist.gov
Cc: Bruce Schneier <schneier@counterpane.com>
Reply-to: Rob Gerlach <Zen@mail.rit.edu>
Organization: RIT/NTID
X-MIMEOLE: Produced By Microsoft MimeOLE V5.00.2615.200
X-Mailer: Microsoft Outlook Express 5.00.2615.200
X-MSMail-priority: High

Dear Sir/Madam,

I have a comment I'd like to share with you, in regards to the AES Development Effort.

I feel that Twofish, given its speed and currently unbreakable security status, would best suit my needs as both an end-user and a consumer; at least when compared to the other AES candidates. I sure hope you'll agree.

The capacity in which I am telling you this is as a 19-year old Information Technology (IT) Major at Rochester Institute of Technology (RIT), a person interested in Data Security & Encryption, and as a consumer who often purchases items online.

Please feel free to contact me with any comments or questions you may have.

Many thanks,

-Rob Gerlach :-)
http://www.rit.edu/~rmg4048/resume/
(Please note that the above page is somewhat outdated, and is undergoing a move. There is only one "broken" link as far as I can tell. I apologize any inconvenience.)

Sender: patrick@green.wl.vg
Date: Tue, 18 Apr 2000 12:14:21 -0400
From: Patrick Gardella <patrick@whetstonelogic.com>
Organization: Whetstone Logic, Inc.
X-Mailer: Mozilla 4.7 [en] (X11; U; FreeBSD 5.0-CURRENT i386)
X-Accept-Language: en
To: AESround2@nist.gov
Subject: Comments

I have always felt that the biggest detriment to adequate security is the speed of the implementation and the system that it takes to run it.

I would like to see that AES work on all systems from the cheapest smart card, to the highest powered system.

Patrick

Patrick Gardella patrick@whetstonelogic.com
VP-Technology patrick@freebsd.org
Whetstone Logic, Inc. This space intentionally left blank.

Date: Wed, 19 Apr 2000 02:10:34 -0400
From: Kevin Bealer <kbealier@stny.lrun.com>
X-Mailer: Mozilla 4.5 [en] (Win98; I)
X-Accept-Language: en,zh,zh-CN,zh-TW,ca
To: AESround2@nist.gov
Subject: comment regarding AES (unofficial)

I would recommend TwoFish, admittedly influenced by having read Schneier's book, and newsletter.

I would rate the following as the critical important factors in how choice of AES will affect me (and those like me):

1. Fast internet usable software. This means hardware-feasable for high speed. By the time the technology reaches users, it will have influenced PC hardware enough... Hardware solutions will be available before a significant user base is there to "populate" them.

2. Personal hardware - signatures, financial, and security applications. In a few years, this will provide the security that signatures, credit card numbers supposedly provide now. This may be the most sensitive to the choice made here. Personal financial and identity security is the requirement that prevents most savvy users from stepping up their reliance on technology.

3. High security will be extremely important to me, not just on personal paranoia, but also because errors here will be extremely costly. We need to assume enormous resources, fantasticly high stakes, (NASDAQ is already almost all tech stocks..) DES was immune to media ignorance effects via semi-obscurity, but today the least "bad tech news" can create lots of badly written but influential panic stories in the non-trade press. Crypto/security will be the last technology issue the media/public ever "understands", but they already fret over it.

4. Patriotic reasons. Although I am not a "conspiracy theorist", there is something fundamentally American about these people: only in radically democratic societies is this on the "fiction" shelf. In geek politics, Twofish makes better political sense - it's author "looks" less conspiratorial.

Thanks for your time,
Kevin Bealer
<kbealier@stny.rr.com>

From: "Simson L. Garfinkel" <simsong@vineyard.net>
To: <AESround2@nist.gov>
Subject: Comments on Round 2 encryption candidates
Date: Tue, 18 Apr 2000 23:47:44 -0400
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.00.2615.200
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2615.200

To: NIST AES Round 2 Committee <AESround2@nist.gov>
Subject: Comments on Round 2 encryption candidates
From: Simson L. Garfinkel

To the Selection Committee,

My name is Simson Garfinkel. I am the author and co-author of nine books, most of which are on computer security and/or system design. Most recently, I am the author of the book "Database Nation" (O'Reilly, 2000). I am also notably the co-author of the books "Practical UNIX and Internet Security" (O'Reilly, 1996) and "Web Security and Commerce" (O'Reilly, 1997.)

Although I am not a cryptographer, I do use encryption on a regular basis in my work. I also do a fair amount of consulting for software designers, telling them how they can use encryption in their programs.

I am currently engaged as a consultant by a company that is attempting to build strong encryption into a device that uses the Motorola MC68HC705JB4 microprocessor. This microprocessor has just 176 bytes of RAM and 4672 bytes of ROM. Even though the 6805 architecture is more than 25 years old, it is still a strong force in the industry today. The microprocessor is popular because it can be purchased for less than $2 in relatively small quantities. The application that has been created for this microprocessor leaves just 64 bytes of RAM available and roughly 1000 bytes of ROM.

I do not believe that our restrictions are unique. Indeed, there are many smart card applications that could benefit from strong encryption. It is my experience that system designers usually develop their systems without encryption and then seek to add it as an afterthought. Therefore, it is likely that many other designers will find themselves in the situation that I am now in --- trying to fit relatively large algorithms into a relatively tiny space.

Based on Geoffrey Keating's paper, it would appear that the Rijndael algorithm is the only AES candidate that will fit in our application. We urge you to strongly consider either the Rijndael algorithm as it is or else a strengthened version of this algorithm.

Failing that, we would urge you to adopt a standard that supports multiple algorithms.

Thank you for your time.

Simson Garfinkel

X-EM-Version: 4, 0, 0, 0
X-EM-Registration: #31E3420614450303B930
X-Mailer: My Own Email v3.5
From: "Keyur Mithawala" <keyur@netexecutive.com>
To: AESround2@nist.gov
Subject: Suggestion...
Date: Wed, 19 Apr 2000 10:40:48 -0600

Hello,
I am working on a Kerberos based security system for networks. As it is evident, we all will be using AES standard for our future developments. I will like to explain my viewpoint for chosing AES algo.

1>> Well, the most important part for an encryption algorithm is SECURITY. Its hardware demands dont matter much. The hardware will catch up in one or two years, but this standard will be there atleast for 5 years.

2>> SPEED for LOW AMOUNT DATA. Large level data encryption is useful only for large corporations. Most users use encryption to encrypt small amounts of data like passwords, etc. Smart cards also encrypt tiny amounts of data. As a kerberos programmer, I use encryption to encrypt authentication data.

Thank you,
Keyur

Date: Wed, 19 Apr 2000 17:20:59 +0100
From: "Jim Gopinathan" <jgopinathan@ascic.co.uk>
Organization: ATMEL Smart Card ICs
X-Mailer: Mozilla 4.7 [en] (WinNT; I)
X-Accept-Language: en
To: AESRound2@nist.gov
Subject: Re: AES and Smart Cards

In smartcard applications speed as well as ROM, RAM and EEPROM requirements are important.

Table 8 in AES3 paper titled "Performance Evaluation of AES Finalists on High-End Smart Card", provides a summary of the memory and processing requirements on Toshiba's 8-bit T6N55 smartcard. The information provided is for algorithm implementations not modified for protection against power analysis, timing attacks and other similar forms of attack.

Modifying code on 8-bit smartcards for protection against these forms of attack will significantly increase ROM, RAM and EEPROM requirements. When DES code on 8-bit HC05 smartcard was modified to be resistant to both timing and DPA attacks, both the code size and RAM requirement approximately doubled.

No doubt a similar increase in code size will occur when the AES candidates are modified to be resistant against these forms of attacks on 8-bit smartcards.

The AES candidate eventually selected, when implemented in software on 8-bit smartcards to be resistant against power analysis and other forms of attacks, must not occupy a significant portion of the ROM and throughput must meet the current and future requirements of smartcard applications.

Jim Gopinathan,
ATMEL Smart Card ICs,
East Kilbride,
Scotland.

X-Authentication-Warning: ns1.crl.go.jp: Host crlgw1 [133.243.18.250] claimed to be crlgw1.crl.go.jp
Date: Thu, 20 Apr 2000 13:19:13 +0900 (JST)
From: Tom Holroyd <tomh@po.crl.go.jp>
X-Sender: tomh@holly.crl.go.jp
To: AESround2@nist.gov
Subject: comment from the public

Multiple algorithms. Aside from the political feuding that will be avoided by not picking a single algorithm, multiple algorithms means that implementors will have to create a mechanism whereby new algorithms can be swapped in, allowing for easy upgrades in the future when all of the current AES candidates start showing their age. Mandate flexibility.

IMO,

Dr. Tom Holroyd
"I am, as I said, inspired by the biological phenomena in which
chemical forces are used in repetitious fashion to produce all
kinds of weird effects (one of which is the author)."
-- Richard Feynman, _There's Plenty of Room at the Bottom_

Date: Thu, 20 Apr 2000 09:35:01 -0700 (PDT)
From: Ray Van De Walker <rgvandewalker@yahoo.com>
Subject: Embedded Systems Criteria
To: AESround2@nist.gov

I've designed embedded computer systems for 17 years.

My priority for AES would be:

1. Software efficiency. Processors are now cheap, and in most applications, AES will be a software-only option. I think this is more important than security, as long as the algorithms are not total push-overs.

2. Minimal conforming implementations will include at least two standard algorithms, and all standard algorithms of any previous AES. Security concerns should be addressed, but encryption is a communication technique. Old software often does not advance.

3. Conforming communication implementations should have a notification and negotiation function, so that the most secure mutually available algorithm is selected. The later implementation should notify the earlier of broken algorithms.

4. I strongly favor use of a GNU-style license. Derivative designs will thereby become public, which aids the public good by reducing costs.

5. The license should explicitly permit reverse engineering and cryptanalysis of any application, provided that successful attempts are published.

Ray Van De Walker
R.G. Van De Walker Inc.
9181 Crawford Circle
Huntington Beach, CA, USA, 92646

From: dave-aes@bfnet.com
Sender: dave@bfnet.com
To: AESround2@nist.gov
Subject: Comment regarding the AES competition
Date: 20 Apr 2000 15:14:35 -0700
Lines: 7
User-Agent: Gnus/5.070099 (Pterodactyl Gnus v0.99) Emacs/20.4

I am a software designer specializing in custom network server systems. I have many customers who require real-time encryption in software. I have thoroughly examined the submissions to AES and believe that Twofish is the most suited to my applications due to its great software performance. I think that Twofish gives the best security to performance trade-off of the submissions, and has the most implementation flexibility. So I support Twofish for AES.

Sender: gwyn@ridley.nist.gov
Date: Fri, 21 Apr 2000 10:08:39 -0400
From: "Douglas A. Gwyn" <gwyn@arl.mil>
Organization: U.S. Army Research Laboratory
X-Mailer: Mozilla 4.72 [en] (X11; U; SunOS 5.7 sun4u)
X-Accept-Language: en
To: AESround2@nist.gov
Subject: AES public comment
X-MIMETrack: Itemize by SMTP Server on mail/arl(Release 5.0.3 |March 21, 2000) at 04/21/2000
10:10:51 AM,
Serialize by Router on mail/arl(Release 5.0.3 |March 21, 2000) at 04/21/2000
10:11:57 AM,
Serialize complete at 04/21/2000 10:11:57 AM

My only comment at this stage is that the final standard should specify only *one* of the algorithms, not multiple algorithms. One reason is that that was the original impression some entrants had when agreeing to waive patent rights etc. Another reason is the added expense and operational hassle of selecting and indicating a choice of algorithm. There is no apparent security benefit to multiple algorithms; they are sufficiently similar that any efficient method of cryptanalysis for one of them is likely to apply to another.

From: keb@smrn.com
X-Sender: keb@mail.smrn.com
X-Mailer: QUALCOMM Windows Eudora Version 4.3.1
Date: Sun, 23 Apr 2000 11:12:34 -0800
To: AESround2@nist.gov
Subject: AES candidates

Sirs:

I'm CTO of MyProof.com, a small e-commerce startup with products built upon strong cryptography. I strongly support TwoFish for the final AES choice.

Sincerely,
Kenneth Broll

From: sidners@tsainc.com
To: AESround2@nist.gov
Subject: AES Round 2 public comments
X-Mailer: Lotus Notes Release 5.0.2b (Intl) 16 December 1999
Date: Mon, 24 Apr 2000 13:56:46 -0500
X-MIMETrack: Serialize by Router on lnm002/TSA(Release 5.0.2b (Intl)|16 December 1999) at
24/04/2000 13:56:51,
Serialize complete at 24/04/2000 13:56:51

I am no cryptographer, but my company is a major software vendor in the international banking community. We are the leading vendor of computer systems to process electronic payments. It has been estimated that 40% of the credit and debit card transactions in the world pass through our systems.

Cryptography is becoming more important in electronic payments - witness the success of SSL in Internet communications. We see the next major thrust in payments to be mobile phones, replacing both plastic and cash as the preferred payment mechanism, first outside the US and then domestically. We expect this to happen over the next 3 to 5 years.

Therefore we are looking for a symmetric key algorithm that is
1) easily implemented in mobile phones
2) fast when encrypting 512 to 4096 byte messages
3) exportable
4) with increasing strength based on key size
5) standard - that is, we would like a single algorithm, to which the industry can build optimized, provably secure implementations, that will interoperate globally as well as DES and Triple-DES have.

Like I said, I am no cryptographer, so I am unable to recommend any of the five submittals. However, Bruce Schneier indicated that you would appreciate comments as to what criteria are important.

Thank you for your consideration,

Sid Sidner
Senior Engineer, Internet Division

ACI Worldwide, 330 South 108 Avenue, Omaha, NE 68154-2684
Mobile: **402-850-7092 (new!)**, Other work: 402-778-1851, Fax: 402-778-1840,
sidners@tsainc.com, www.aciworldwide.com
A TSA Company

"Rationality is the servant of intuition."

To: AESround2@nist.gov
dcc:
Subject: comment on AES candidates
Date: Tue, 25 Apr 2000 08:49:57 -0700
From: Dan Stromberg - OAC-DCS <strombrg@nis.acs.uci.edu>

So far, I think twofish should become the AES.

I personally am primarily concerned with speed of encryption in software, but an all-around candidate is best I think.

Sender: bkelly@odo.edwards.af.mil
Date: Wed, 26 Apr 2000 11:10:34 -0700
From: Brent Kelly <bkelly@odo.edwards.af.mil>
X-Mailer: Mozilla 4.51C-SGI [en] (X11; I; IRIX 6.5 IP22)
X-Accept-Language: en
To: AESround2@nist.gov
Subject: AES Finalist Comments

Gentlemen:

I support Twofish for the AES, I believe it's combination of security and relative ease of implementation in either hardwaror software make Twofish the optimum choice. I do not think there should be multiple standards, as may have been suggested.

Thank You,
Brent
--

Cedric Brent Kelly — Computer Sciences Corp.
805-277-7677 voice — Ridley Mission Control
805-277-5377 fax — Edwards A.F.B., Ca. 93523
Clear Skies... — EMAIL: bkelly@odo.edwards.af.mil

Date: Thu, 27 Apr 2000 08:54:31 -0700 (PDT)
From: "Nicholas C. Weaver" <nweaver@cs.berkeley.edu>
To: AESround2@nist.gov
Subject: My last comments
Cc: nweaver@cs.berkeley.edu

This is my LAST feedback note (this time I really mean it).

RC6 keyschedule "hack". It was pointed out to me that the trick is to not just store and fetch the states of A and B, but copies of the L array, for the second and third pass through the S array. This does considerably increase the storage and fetching required (112B instead of 32B for 256b keys), but is still better than fetching all 176B of fully expanded subkeys. Thus, the savings exist but it is still "not that much".

One could also use this for a smartcard where key setup and encryption are done concurrently to save memory, but once again, the savings are "not that much" for a 256b key.

Subkey agaility isn't just for hardware, but for smartcards which lack the resources to store the fully expanded keys.

Also, this is a response of mine on the MARS key schedule, describing how if one is willing to leave the key schedule potentially open to power and timing attacks, one can do the string matching & correction of the multiplicative subkeys without requiring too much resources or time. I use FPGA terms, but ASICs and custom logic also apply.

(Hopefully it will be a moot point, my bias against MARS is clear, but Just in Case):

For hardware, although I didn't consider the stringmatching portion in detail, I believe the cost should be fairly minor if performance is not an issue, and even if performance is a moderate concern. Here is why:

It takes 1 lut to determine if four inputs are all 0 or all 1. (Output is 1 if they are all the same, otherwise 0). If you AND the results together from several of these, this will be a 1 if you have an overlap of one input between two luts. (EG, bits 0, 1, 2, 3 into lut A. 3, 4, 5, 6 into lut B, etc, and then and the outputs all together). Thus, it would only take 6 LUTs to evaluate a 12 bit run. If one wants fine information on when a run ends, it takes 1 lut/bit, which isn't that large.

The slow approach would just be to load the key into a shift register and do a shift & match approach. If faster performance is required, have a couple of additional outputs which would show where the currently examined run ends, so you could skip 4 for each shift.

EG, if first 4 don't match, skip to the last unique one in that 4. If the first 4 do but the next 3 don't, skip 4. If the first 7 do but the next one doesn't, skip 7. The additional logic for this would be very small.

Since the MARS hardware requires a barrel rotator, if you are not doing subkey generation and encryption in the same cycle, you could reuse the block to skip through the keys.

This does leave the subkey generation potentially open to power or timing attacks, however, as the work and time are vastly different based on the bit pattern in the multiplicative keys.

However, the MARS subkey generation in general, even without the string matching step, poses serious problems if fast subkey generation is desired in hardware, due to the sequential dependancies.

--
Nicholas C. Weaver nweaver@cs.berkeley.edu
It is a tale, told by an idiot, full of sound and fury, .signifying nothing

Date: Sat, 29 Apr 2000 11:44:20 -0700
From: David Eppstein <eppstein@ics.uci.edu>
To: AESround2@nist.gov
Subject: Comment on selection of the algorithm for AES
X-Mailer: Mulberry/2.0.0 (MacOS)

Over the expected lifetime of AES, improvements in technology and algorithms will surely make performance considerations less of a concern, and security considerations more of a concern. Therefore, I strongly agree with AES's original charter "The security provided by an algorithm is the most important factor in the evaluation."

To me this means that, since RC6, Rijndael, Serpent, and Twofish all appear to have adequate and roughly similar performance, we should choose one that has been more conservatively overengineered for security, rather than one that has trimmed the number of rounds to the minimum believed-secure level. I.e., Serpent and Twofish should be preferred over Rijndael and RC6. Also, to minimize the possibility that any AES cipher is broken, we should have one and only one algorithm chosen. Taking into account the requirements of simplicity and key agility, I would be mildly in favor of Serpent over Twofish.

--
David Eppstein UC Irvine Dept. of Information & Computer Science
eppstein@ics.uci.edu http://www.ics.uci.edu/~eppstein/

From: "apli" <apli@btamail.net.cn>
To: <AESround2@nist.gov>
Subject: AES
Date: Tue, 2 May 2000 07:47:16 +0800
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3110.5
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3

Dear Sir:

I have known AES for only a few months since my major in mathematics. As we seen, the five candidate algorithms selected for AES have made further endeavors on the security and implements over DES. The progresses as they made, it seems that the design ideas adopted in the most of these algorithms are yet mainly as one in DES. The structure of DES have been intensive studied recent years,and is relatively easy to be attacked. For this it can be seen in the late cryptanalyses papers on these algorithms. So, the best in the finalist there should be some more new ideas and new structures. If you intend to call for additional new algorithms in the future some time, I would like to submit one. Besides, I have an opinion that as the knowledge and technology for computer have been developing very rapidly the servive period drawn up for one algorithm of AES should not be too long. The best maybe NIST should make up one research group to keep research on the algorithms of AES.

Sincerely yours
Anping Li

X-Sender: mike@mail.pcfax.com
X-Mailer: Windows Eudora Pro Version 3.0 (32)
Date: Thu, 04 May 2000 08:58:08 +0100
To: AESround2@nist.gov
From: Mike Lake <mike.lake@wordcraft.co.uk>
Subject: AES comments

I hope that comments from overseas users are acceptable.

My company is involved with encryption usage in software, firmware and hardware and for test purposes we have implemented a number of the candidate algorithms.

Our primary desire would be a standard that can be implemented across a wide range of platforms rather than having multiple standards optimised for each platform. Obviously we would like to see a high level of security, tight code and fast throughput.

For these reasons we are strongly in favour of the Twofish algorithm - primarily because of its suitability in the widest range of applications

Best regards

Mike Lake
Chairman: Wordcraft International Limited
Web site: http://www.wordcraft.co.uk

Office: 01283-731400 (International: +44-1283-731400)
Mobile: 07973-432085 (International: +44-7973-432085)

From: Jeffrey Streifling <jss@icrossroads.com>
To: AESround2@nist.gov
Subject: Last minute notes
Date: 4 May 2000 18:20 -0600

I sent a message earlier suggesting that simplicity be a deciding factor in the AES selection process; my choice for simplest candidate was RC6. After reading the AES papers and comments, it looks like it will probably be discarded along with MARS due to the inefficiency of the multiply in some environments. If there is a secondary AES pick, it would make a good choice (due to the fact that its structure is very different from everything else, and thus unlikely to break simultaneously with the primary pick), but the idea of a secondary AES pick motivated by security reasons seems a bit costly compared to the alternative of requiring compliant implementations to parameterize the number of rounds.

Observe that virtually all modern cryptanalysis of non-linear ciphers breaks at some number of rounds. When we do differential cryptanalysis, we are able to break up to n rounds using a particular technique, whereupon our technique becomes more expensive than brute-force, or otherwise runs out of theoretical gas. Similarly, more rounds disguise linear bias, causing linear cryptanalysis to require more plaintext. Other kinds of cryptanalysis run out of gas after a particular number of rounds; it is exceedingly rare to find a break of a cipher that can simultaneously solve any number rounds of said cipher in general with equal ease. The point to note is that security in all cases has hinged on the number of rounds. Therefore, a simple way to future proof an AES selection is to require implementations to parameterize the number of rounds. If somebody breaks 100 rounds of AES, we give 'em 200. If somebody breaks 150 rounds, crank it to 300! It may be disconcerting and dog slow, but guaranteed to be interoperable with smart-cards and other limited environments where in practice only one AES candidate will be implemented regardless of how many are chosen. (If the plan is to switch to a backup cipher, then we suddenly have the huge problem of redeployiny hardware, smart-cards, etc, that only actually implemented the primary candidate.) Furthermore, compliance would have already been tested; there would be no need to wait through overhead of getting new products out when time may be of the essence (supposing some future break of N rounds proves dangerously practical). Cranking Rijndael or RC6 to thousands of rounds might possibly seem ugly compared to having a backup, but it would be able to function for a few years while a new standard was created. The advantanges would be (1) guaranteed instant switchover -- just poke in the new desired number of rounds and (2) guaranteed ubiquitous availability -- it is reusing the machinery already deployed. Furthermore, creating the replacement cipher after the hypothetical lethal break would permit it to take advantage of the new crypographic techniques suggested by the break, which would make it a better solution than any backup we can propose now.

Therefore, implementations to be certified AES-compliant should be required to let users change the number of rounds (along with the key -- configuration is already required), although it clearly should default to the standard value. Note that this would recommend choosing RC6, Rijndael, or Serpent since MARS and Twofish do not have parameterized round count.

Then again, MARS and Twofish are already the two ciphers which suffer from excessive complexity. My personal order of preferences for the AES2 candidates based on complexity is
(1) RC6
(2) Rijndael
(3) Serpent
(4) Twofish
(5) MARS
(and therefore don't be surprised that I think that at least one of RC6 and Rijndael should be in the standard).

Reflecting on the possibilities, then, of various single or multiple candidate choices, it occurs to me that the reasons the NIST should choose a possible secondary candidate for the AES is not security, which would better be accomplished with a larger number of rounds (to allow instant, ubiquitous switch-over as described above), but rather implementation flexibility to allow AES to exist in some form on more environments that could be addressed by a single cipher and to permit a wider variety of security/performance tradeoffs. The secondary cipher, if chosen, should be RC6 because it tends to excel in the environments that make the other ciphers flop (consider the Java performance, for example). I can't see it becoming the primary AES due to its difficulty working in low-end microcontrollers and the hardware overhead of the multiplier (and possibly the poor performance once the number of rounds are increased as may be indicated by recent cryptanalytic results). Or, if the flexibility argument merits no consideration, mandate just one cipher for the advantage of maximum interoperability . . .

What then, of the primary AES choice? The multiplier required by RC6 and MARS makes them inefficient in a lot of environments; they can probably be dismissed out of hand (particularly MARS, which was practically ignored in the hardware papers). Further, the complexity required for the Twofish key schedule makes it a lousy cipher key agility is required (such as encrypting ATM cells, or hash function constructions), where the code must be kept simple and easy-to-understand (where the cipher is part of a larger program that spans a security boundary and thus must be subjected to extensive, expensive code audits) or where it is otherwise desired to produce a simple implementation for academic or "one-shot" purposes. Should Twofish be chosen as the sole AES standard, its key schedule would raise an annoying if surmountable barrier to simple implementations: you either port existing code from somewhere else, or sit down and do a fair amount of thinking . . .

As for Rijndael versus Serpent, my preference would be for Rijndael given my bias toward simple ciphers, although the number of rounds probably needs to be increased a little bit based on recent cryptanalytic results. I would not say the decision is obvious from performance considerations, though. More rounds will slow Rijndael, and Dag Arne Osvik as significantly sped up Serpent and provided directions for future speed-ups. Furthermore, I think that the AES algorithm, once widely implemented, will be come a sort of benchmark of computer speed (rather than the other way around), and that chip manufacturers will focus on making Serpent run fast on their processors as a marketing tool. For example, if Serpent is chosen as an AES candidate, Biham's bitslice would probably be added to new processors, significantly speeding up Serpent with respect to other candidates. Furthermore, Serpent scales marvelously to 64 bit architectures (and in general, to n-bit architectures in bit-slice implementations), thereby

allowing it to improve with age (it is already second-fastest on current 64-bit benchmarks, and seeing it come out fastest once chipmakers tune their architectures for AES would not surprise me). It would be useful to increase the number of rounds of other ciphers to approximately match the security of serpent, whereupon Serpent would prove quite fast. Serpent has the potential to significantly contribute to the design of modern microprocessors should it become the chosen standard.

If the NIST chooses a single cipher for the AES, it should probably be Rijndael (an opinion that I think was mirrored in the conference response papers); my reason for choosing it would be simplicity. If the NIST chooses two ciphers for AES, it should choose Serpent as the primary and RC6 as the secondary. It should in any case parameterize the chosen ciphers with respect to round counts so they can easily be changed in the field; or, failing that, choose 32-round Serpent as the singular standard for security reasons.

Actually, any combination of the simpler, round-parameterizable, would be useful -- even providing Rijndael as a secondary for Serpent would admit some reasonably simple implementations. Failing all else, it is sufficient to require that the ciphers chosen be round-parameterizable to function as a life-boat cipher in the event of a break (before a new cipher is created); all such ciphers are secure given sufficient rounds, and all AES2 ciphers that meet this criterion happen to be simple enough to admit complexity-cheap implementations.

Jeffrey Streifling
<jss@icrossroads.com

X-Sender: ggr2@127.0.0.1
X-Mailer: QUALCOMM Windows Eudora Pro Version 4.2.2
Date: Sun, 07 May 2000 17:29:08 +1000
To: AESround2@nist.gov
From: Greg Rose <ggr@qualcomm.com>
Subject: Suitability of Rijndael without extra rounds
Cc: ggr@qualcomm.com

At the recent AES3 conference, there was much discussion about the number of rounds of the candidates, particularly Rijndael. There seemed to be a general feeling that Rijndael had too few rounds for an adequate safety margin.

I disagree with that sentiment. To put it more clearly, I think Rijndael's safety margin is perfectly adequate against publicly known or hypothesised attacks. Of course it is possible that some newly discovered or revealed attack might apply to any of the candidates (and perhaps MARS has a slight advantage in this eventuality). If that happens, I find it hard to believe that increasing the number of rounds from, say, 10 to 14 in the 128-bit case, will save Rijndael from a devastating new attack.

As support for my opinion, I offer the following three examples:

DES, in which the resistance to differential cryptanalysis is very close to its key-limited strength, and yet is the most widely deployed and respected encryption algorithm ever.

SKIPJACK, where attacks such as Biham's go right to the edge (31 out of 32 rounds), and yet it is another apparently successful algorithm.

FEAL, which was designed before differential cryptanalysis was publicly known, had to have the number of rounds quadrupled before it approached a level of real security against that "unknown" attack.

Personally, I'd like to see Rijndael, as submitted, adopted as the AES algorithm, although there's very little between the candidates. I don't think committee-based micromanagement, about the number of rounds, should be applied to any of the candidate algorithms (in either direction).

sincerely,
Greg.

Greg Rose INTERNET: ggr@Qualcomm.com
Qualcomm Australia VOICE: +61-2-9181-4851 FAX: +61-2-9181-5470
Suite 410, Birkenhead Point, http://people.qualcomm.com/ggr/
Drummoyne NSW 2047 232B EC8F 44C6 C853 D68F E107 E6BF CD2F 1081 A37C

From: Gideon Yuval <gideony@MICROSOFT.com>
To: "'AESround2@nist.gov'" <AESround2@nist.gov>
Subject: protocol stuff
Date: Mon, 8 May 2000 16:18:42 -0700
X-Mailer: Internet Mail Service (5.5.2651.58)

Brian Snow suggested having AES and double-AES specifiable in protocol negotiations. I would like to add one thing: have the bit-code for AES be 00000..., and the bit-code for double-AES be 11111...; all codes in the middle should crash _before_ shipping any bits out. The codes for AES and 2AES should work all the way (& thus be testable all the way).

Delivered-To: fixup-AESround2@nist.gov@fixme
User-Agent: Microsoft Outlook Express Macintosh Edition - 5.01 (1630)
Date: Tue, 09 May 2000 21:50:17 -0500
Subject: Comments on Rijndael, Serpent, Twofish software implementation
From: "David C. Oshel" <dcoshel@mac.com>
To: <AESround2@nist.gov>

I have incorporated Rijndael, Serpent and Twofish in a small file encryption grinder for Macintosh. This little utility includes a Test routine designed to exhibit simple characteristics of the program's keystream – produced using the algorithms in question.

While not especially interesting as analysis, the routine does give the algorithms a workout. The default, "Lite," routine -- basically a species of PrngXor -- completes the Test routine in about 15 seconds on a Macintosh Powerbook G3 (Bronze). Rijndael typically concludes the task in 21 seconds, compared to 23 seconds for Serpent, and 1 minute 19 seconds for Twofish.

IMHO, this clearly argues against Twofish in purely software implementations.

The code I used was Brian Gladman's as he had it posted near the end of Round 1, in other words, the source which #includes "../std_defs.h", pretty nearly straight "out of the box" and compiled with Metrowerks' C++ compiler.

I did not bother testing RC6 or Mars because there seemed little practical point in using candidates with reservations about their intellectual property status.

--
David C. Oshel mailto:obsidian@kagi.com
Cedar Rapids, IA http://homepage.mac.com/dcoshel/Obsidian.html
``Tension, apprehension and dissension have begun!'' - Duffy Wyg&, in Alfred Bester's _The Demolished Man_

From: DAEMEN.J@protonworld.com
To: AESround2@nist.gov
Subject: Royalty Statement of the Rijndael Team
Date: Wed, 10 May 2000 13:58:07 +0200
X-Mailer: Internet Mail Service (5.5.2650.21)

At the request of the NIST AES Team, here is an updated Royalty Statement for Rijndael:

In the case that NIST decides to adopt as AES an algorithm that can be considered as a variant of Rijndael, the Rijndael team will make no royalty claims whatsoever.

Joan Daemen & Vincent Rijmen

From: Carlisle Adams <carlisle.adams@entrust.com>
To: "'AESRound2@nist.gov'" <AESRound2@nist.gov>
Subject: For inclusion in the Round 2 public comments...
Date: Thu, 11 May 2000 12:10:46 -0400
X-Mailer: Internet Mail Service (5.5.2650.21)

Dear NIST AES Selection Committee,

>From reading NIST's Round 1 report, from looking at the public comments received so far in Round 2, and especially from attending the 3rd AES Conference last month, one thing in particular strikes me as relatively clear: Rijndael appears to have the fewest drawbacks of the 5 finalists. There seem to be concerns associated with MARS, RC6, Serpent, and Twofish. Whether the difficulties involve complexity, performance, overall design, key agility, suitability for very constrained environments, suitability for tomorrow's 64-bit (or higher) processors, or some other reason, it seems that the choice of any of these four finalists will end up causing some level of dismay for some group of implementers. Rijndael, by contrast, appears to have very many "plusses" and very few "minuses"; it seems to place at least in the middle-of-the-pack (and often places first or second) on every criterion considered and on every platform proposed. Furthermore - and to me this is significant - it is the algorithm most often chosen as a favorite by the submitters when they are asked to pick a candidate other than their own.

As far as I can tell (from both public and private discussions at last month's FSE/AES conferences), the only minor concern with respect to Rijndael is that its margin of security may not be high enough for the 50 or 100 years that Ross Anderson proposes will be the actual lifetime of the AES cipher. Consequently, I recommend adding 4-6 rounds to this algorithm for each of its three sizes (i.e., going from {10, 12, 14} rounds to {14, 16, 18} or {16, 18, 20} rounds). This should allay any fears regarding security margin without destroying its substantial performance characteristics on known or anticipated platforms (that is, it might no longer be the fastest of the candidates, but neither would it be among the slowest because of its current impressive speed).

In my reading and conversations thus far, I have yet to come across a significant number of people that would seriously object to Rijndael being selected as the AES cipher, especially with the addition of a small number of rounds. I have not found this same level of general acceptance for any of the other candidates. Given that the people following the AES process are evaluating the ciphers from a multitude of different (and sometimes opposing) perspectives, with a wide variety of skill sets, backgrounds, and biases, in my opinion this speaks volumes.

Carlisle Adams

X-Authentication-Warning: korkeik.ii.uib.no: larsr owned process doing -bs
Date: Thu, 11 May 2000 23:37:53 +0200 (MET DST)
From: Lars Ramkilde Knudsen <Lars.Knudsen@ii.uib.no>
To: Jim Foti <jfoti@nist.gov>
cc: AESround2@nist.gov
Subject: Re: AES - RE: a question at AES3

Jim,

Serpent is completely royalty-free. We let the patent lapse. Whether you decree 16 rounds or 64, we can't charge anyone a penny.

Neither can anybody else, as far as we can see. Since Serpent resuses the DES technology (recall that the first version of Serpent even used the DES S-boxes) and since IBM's patent on DES expired a long time ago, it's unlikely in the extreme that some third party will come up with a subterranean patent.

This wasn't a deliberate design feature, just a side effect of our decision to stick to tried and tested primitives.

So we believe that our algorithm is not just the most able to resist attacks involving cryptanalysis, but also attacks involving IPR.

Best regards
Ross Anderson
Eli Biham
Lars Knudsen

From: "Simpson, Sam" <s.simpson@mia.co.uk>
To: aesround2@nist.gov
Subject: Round 2 AES Comments
X-Mailer: Internet Mail Service (5.5.2650.21)
Date: Fri, 12 May 2000 09:46:39 +0100

Dear Selection Committee,

I have watched the AES process with great interest for two years and have finally decided to write to share my thoughts on the selection of the final algorithm(s). I am a user & implementor of encryption and have assisted Dr B.Gladman in significantly improving the original Serpent S-BOX performance [1].

At this late stage I believe that decisions can be made in three key areas, as the previous round appears to have removed all clearly inappropriate ciphers. I believe any of the 5 remaining ciphers will make an adequate AES cipher, but it is my belief that NIST has the opportunity to select an excellent cipher from a choice of two or three.

Security

Security *must* be the primary concern when evaluating these ciphers. An "over-engineered" cipher will merely be slow until hardware is sufficient for the application, but an "under-engineered" cipher would render the cipher obsolete (or at a minimum be subject to the same kind of mistrust as per DES). NIST has declared AES [2]: 'A crypto algorithm for the 21st century' and security will certainly be the metric by which this statement is evaluated.

1) AES should rely on tried and tested security constructs. Several ciphers employ relatively new constructs (MARS & RC6 with data dependent rotations, Twofish & Rijndael with MDS) which will not be adequately assessed in the two-year time frame. The strength of RC6 seems to depend solely on DDR and RC6 should therefore be dropped from further consideration.

2) Algorithms should be clean and easy to evaluate. Again, in the two-year timeframe it is unlikely that the inner workings of a complex algorithm will be fully assessed (see for example [3]). From a cryptanalytic perspective both MARS & Twofish are significantly more complex than the other three algorithms and this should count substantially against the algorithms.

3) Number of rounds should be over-engineered (as per [4]). RC6 appears to be significantly under-engineered whilst MARS & Rijndael are slightly under-engineered. Twofish is over engineered, but Serpent is by far the most over-engineered design.

4) The successful algorithm should not be prohibitively expensive to secure against "operational" attacks on smartcards.

5) The argument that the heterogeneous structure of MARS adds some security is plausible, but the benefit seems relatively small compared to 2) above.

Platform Applicability

Another important concern when selecting an algorithm is whether it runs on all necessary platforms. Clearly, an ideal situation is the selection of an AES cipher that is fully compatible across the full range of potential platforms:

1) My understanding is that MARS & RC6 cannot be implemented on very low-end smart cards [5]. I assert that neither MARS nor RC6 offer anything particularly special over the other 3 candidates (RC6 has a very clean design and MARS has the benefit of heterogeneous structure, but I don't think these benefits are compelling) and should therefore not be considered for AES.

2) The remaining three ciphers appear to be applicable to all platforms. Rijndael appears to have the most uniform speed across all hardware and software platforms (discussed next).

Performance

I believe performance should be used as a metre after the two previous criteria have been suitably satisfied - without either security or pervasiveness the standard will not be widely accepted and deployed.

1) Need to consider the whole range of applicable software and hardware implementations, not just results that suit specific ciphers. Rijndael performs very well across all platforms; Twofish works well across all platforms apart from Java [6]. It should also be noted that Twofish is very slow in software when only encrypting one or two blocks due to the large key-setup overhead.

2) MARS & RC6 require specific support for variable rotations & mod 2^32 multiply and therefore performance is severely impeded on platforms not providing these instructions (not only smartcards but also current platforms such as Sun UltraSPARC [7] and forthcoming platforms such as Intel's IA64 [8]).

3) Does the speed of any of the 5 candidate ciphers prevent it from reasonably being used across platforms? A Pentium Pro 200Mhz "reference platform" can run the slowest candidate at 34Mbits/s (approximately the same as DES [1]), whilst a Pentium II running at 200Mhz obtains 45Mbits/s.

4) Arguably, hardware throughput is most important (for highspeed systems such as IPSEC and ATM for example). Smartcard operations will usually be low bandwidth (e.g. challenge response / passphrase encryption and so on) and will therefore not be unduly affected by a slow cipher.

5) Scope for speed optimisation. Most of the ciphers submitted appear to be near optimal on the NIST test platform, but one cipher Serpent has shown significant and continuing improvement due to S-Box optimisation [1], [9]. It is likely that Serpents' performance could be similarly improved by:

a) Producing more S-Box representations with fewer terms. This would probably improve the performance across most if not all platforms.

b) Rearranging S-Box terms to better exploit both processor level parallelism and "streaming" parallelism (e.g. MMX, SSE, 3Dnow, Altivec and other SIMD units) - see [1] for example.

6) We can't predict how CPU architectures will look in the future, but the current trend is the support for multiple pipelines and streaming (e.g. SIMD) instruction sets (see for example, Pentium III, AMD Athlon and PowerPC Altivec). These types of architectures would appear to benefit Twofish, Rijndael & Serpent more than RC6 and MARS [8].

7) The MARS and Twofish designers have implemented cipher components that are available at little or no cost on the specified test platform but may be expensive on other platforms.

8) Algorithms won't always be implemented in hand optimised assembly code - high-level language speed is also very important. I note [10] that Serpent has the best high-level language to assembly ratio.

AES was always going to be a trade-off between security and speed. I believe NIST correctly laid down the criteria in [11]: "...with a strength equal to or better than that of Triple DES and significantly improved efficiency". The Twofish teams' comments also seemed prudent [12]: "While it is impossible to optimize a cipher design for resisting attacks that are unknown, conservative design and over-engineering can instil some confidence". We need to recall that even the slowest performing R2 AES candidate, Serpent, is approximately three times faster than Triple-DES and (is widely believed to be...) significantly stronger than Triple-DES and the other AES candidate ciphers.

The 5 remaining teams whom submitted AES candidates (predictably...) continue to assert their own algorithms' security benefits and, whilst they are unable to arrive at a modus vivendi, they do all appear to agree that Serpent is the most secure candidate, but then move on to knock it for speed. I suggest, therefore that if NIST is now tied to picking one of the unamended algorithms then Serpent *has* to be the algorithm of choice. If NIST is allowing cipher modifications at this point then Rijndael + 3r is also a strong consideration.

Finally, I must applaud NIST for their openness in selecting AES - this will no doubt ensure that the successful algorithm will not be subjected to the same mistrust as DES.

Regards,

Sam Simpson B.Sc. (Hons.)
IT Operations Manager
MIA Ltd (UK)

Phone : +44 (0)1438 735478
Fax : +44 (0)1438 726069
e-mail : s.simpson@mia.co.uk

References:

[1] B.Gladman, "Serpent", available at http://www.btinternet.com/~brian.gladman/cryptography_technology/serpent/index.html

[2] M.E.Smid, "AES - A Crypto Algorithm for the Twentyfirst century", FSE Workshop, 1998.

[3] M.J.B.Robshaw, Y.L.Yin, "Potential Flaws in the Conjectured Resistance of MARS to Linear Cryptanalysis", 2000.

[4] E.Biham, "A Note on Comparing the AES Candidates", 1999.

[5] B.Schneier, D.Whiting, "A Performance Comparison of the Five AES Finalists", 2000.

[6] J.Dray, "NIST performance Analysis of the final round JAVA AES candidates", 2000.

[7] L.E.Bassham, "Efficiency Testing of ANSI C Implementations of Round 2 Candidate Algorithms for the Advanced Encryption Standard", 2000.

[8] J.Worley, B.Worley, T.Christian, C.Worley, "AES Finalists on PA-RISC and IA-64: Implementations & Performance".

[9] D.A.Osvik, "Speeding up Serpent", 2000.

[10] B.Gladman, Personal communication, 2000.

[11] "ANNOUNCING REQUEST FOR CANDIDATE ALGORITHM NOMINATIONS FOR THE
ADVANCED ENCRYPTION STANDARD (AES)", Federal Register, Friday, September 12,
1997.

[12] B.Schneier, J.Kelsey, D.Whiting, D.Wagner, C.Hall, N.Ferguson, "Twofish: A 128-Bit Block Cipher", 1998.

Date: Fri, 12 May 2000 15:46:33 +0530 (IST)
From: "R.Venkatesh" <venky@cse.iitb.ernet.in>
To: AESround2@nist.gov
Subject: AES Comment.

Hello:

I am R.Venkatesh and I research on cryptographic techniques with Tata Infotech Ltd, India. I attended the AES Round 3 Conference held in New York City. In my position as Senior Research Associate with the said Company, I enclose herewith my comments on the Advanced Encryption Standard.

Thank you.

With warm regards,
for Tata Infotech Ltd.,

(R.VENKATESH)

Script follows ==>>
==========

Comments on the Advanced Encryption Standard

Author : R.Venkatesh
Date : May 09,2000

Standardization of the cipherstream

To obtain a cipherstream from a given plaintext stream, the following steps are carried out.
- Divide the plaintext stream into a number of fixed-size blocks. Each block forms the basic unit of encryption. In case the last block is not filled completely, perform appropriate padding at the end of the plaintext stream to ensure that.
- Encrypt each fixed-size block using the encryption core.
- Concatenate the encrypted block to the previously encrypted block to form the cipherstream.

Suppose the cipherstream alone is received by a decryptor. Then the decryptor will not be in a position to determine the parameters necessary for successful decryption. Specifically, some of the additional information needed by the decryptor may be:
- The algorithm identifier that specifies which algorithm was used in encryption
- The block length used in encryption (this could be fixed in certain ciphers such as RC-6 or user-defined in others such as Rijndael).
- The number of rounds (this could be fixed in certain ciphers such as Serpent or user-defined in others such as RC-6).

- The size of the underlying plaintext. This is necessary to find out the amount of padding done at the end of the plaintext before encryption; this padded string can be discarded accordingly after decryption.
- The initialization vector, incase other modes of encryption are used (such as Cipher Block Chain, Cipher Feedback, Output Feedback).

This information is therefore common to the encryption and decryption processes and could be typically be stored by an encryption process in a descriptor. This descriptor will be made available to the decryptor in addition to the cipherstream.

It is suggested that a standard be evolved in storing and possibly encoding such descriptor content. As a consequence, interoperability of different implementations of the same cipher is ensured. RSA Laboratories has already taken a step in ensuring this at the level of the cipherblock and the public and private keys by enforcing the Public Key Cryptography Standard (PKCS). RSA has also specified encodings by relating the ASN.1 object members for the cipherblock and the keys against their corresponding PKCS formats. Even in public key cryptography, no such standard has been enforced on the ElGamal public-key cryptosystem. And the standardization of the Elliptic Curve Cryptosystem is under way via the PKCS #13.

This standard should be as universal as possible. Then users/developers need not have to write specific interface functions to fill data members in the objects of other application standards such as S/MIME, etc.

This evolution need not be done during the decision-making process of the AES algorithm(s); that process only deals with the encryption core and the basic unit of information to be encrypted. However as mention was made during the introductory session, additional modes of encryption would be taken up for the AES algorithm(s). This could mean one or both of the following:
- New modes similar to the chaining and feedback modes.
- The new standardization scheme suggested above.
(This last suggestion has evolved due to the advice of Dr.Brian Gladman who specifically mentioned that taking up standardization issues during decision-making of the AES core(s) could result in a lot of confusion. Instead this should be put forward as another mode while deciding the AES encryption modes later. Moreover although the encryption core may be very strong, more often it is an implementation of it that could pose security loopholes. Consequently the standard has to be designed and security-audited very carefully before it is put forward).

(Dr.Bruce Schneier has mentioned the suggestion on standardization as an excellent one when put forward during the concluding stages of the AES Round 3 Conference).

Date: Fri, 12 May 2000 13:33:51 +0100 (WEST)
From: Håvard Raddum <haavardr@ii.uib.no>
Reply-To: Håvard Raddum <haavardr@ii.uib.no>
Subject: Something to keep in mind when selecting the AES algorithm
To: AESRound2@nist.gov
X-Mailer: dtmail 1.3.0 @(#)CDE Version 1.4_32 SunOS 5.8 sun4u sparc
X-MIME-Autoconverted: from QUOTED-PRINTABLE to 8bit by email.nist.gov id HAA02846

In NIST's call for candidates it is pointed out that security will be more important than performance when judging the candidates. I would like to stress the point that once the AES algorithm is selected, its security will start getting weaker, and its performance will start getting better. We already saw several examples of this at AES3, with new and improved attacks on some candidates and how to speed up the slower candidates. In other words, the AES standard should be the one with the largest safety margin as long as it's useable in practice. In my opinion Serpent is the safest AES candidate. Serpent is also faster than 3DES (NIST's benchmark), and should therefore be selected as the AES standard.

Håvard Raddum

To: AESRound2@nist.gov
Subject: Why we chose Rijndael
Date: Fri, 12 May 2000 14:00:05 +0200
From: Simon Wigzell <simon@orcsoftware.com>
Reply-To: simon@orcsoftware.com
X-Mailer: by Apple MailViewer (2.106)

A short note from one developer concerning why our company chose Rijndael when we needed encryption

Our program suite, a rather complex one with a server/client environment, uses encryption for all communication. What we needed was something which was as speedy as possible for encrypting small chunks of data (typically a couple of hundred bytes), with no state whatsoever. There is a lot of connections going on; a typical server recieves maybe a thousand of these packages every minute. This implied that the encryption had to be really fast to initialize (setting up keys, since we don't have a state leading to every package having it's own key as far as the server is concerned) and also being more secure than DES since the financial institutes using our product are very concerned about security. As the situation is right now the only candidate that is good enough for us is Rijndael. It is just fast enough and using little enough memory to be possible for our use (though we don't have much margin).
Some of the speakers at the AES conference said we should disregard the speed issue (or at the least not concern ourselves too much with it) since in a few years all computers will be that much speedier, but the problem is that we need something that is good today, not tomorrow.

For our purposes Rijndael was a given, but that might not be true for others. Irregardless of whether NIST chooses Rijndael for AES we will continue using it since it is, without doubt, our best option.

Yours Sincerely,

Simon Wigzell

Simon Wigzell
Cryptologist
Orc Software
http://www.orcsoftware.com

Delivered-To: fixup-AESround2@nist.gov@fixme
User-Agent: Microsoft Outlook Express Macintosh Edition - 5.01 (1630)
Date: Fri, 12 May 2000 16:41:20 -0500
Subject: Re: Comments on Rijndael, Serpent, Twofish software implementation
From: "David C. Oshel" <dcoshel@mac.com>
To: <AESround2@nist.gov>

This is a followup on my earlier comment. What I perceived as a problem with Twofish appears to be a mere documentation issue. Twofish does not like frequent key changes (in common with Blowfish), at least in my naïve implementation. When I revised my code following hints from D. Whiting, the performance discrepancies vis-a-vis Rijndael and Serpent essentially disappeared.

Thanks for your patience,
Dave Oshel

From: David C. Oshel <dcoshel@mac.com>
Date: Tue, 09 May 2000 21:50:17 -0500
To: <AESround2@nist.gov>
Subject: Comments on Rijndael, Serpent, Twofish software implementation

I have incorporated Rijndael, Serpent and Twofish in a small file encryption grinder for Macintosh. This little utility includes a Test routine designed to exhibit simple characteristics of the program's keystream – produced using the algorithms in question.

While not especially interesting as analysis, the routine does give the algorithms a workout. The default, "Lite," routine -- basically a species of PrngXor -- completes the Test routine in about 15 seconds on a Macintosh Powerbook G3 (Bronze). Rijndael typically concludes the task in 21 seconds, compared to 23 seconds for Serpent, and 1 minute 19 seconds for Twofish.

IMHO, this clearly argues against Twofish in purely software implementations.

The code I used was Brian Gladman's as he had it posted near the end of Round 1, in other words, the source which #includes "../std_defs.h", pretty nearly straight "out of the box" and compiled with Metrowerks' C++ compiler.

I did not bother testing RC6 or Mars because there seemed little practical point in using candidates with reservations about their intellectual property status.
--
David C. Oshel mailto:obsidian@kagi.com
Cedar Rapids, IA http://homepage.mac.com/dcoshel/Obsidian.html
``Tension, apprehension and dissension have begun!'' - Duffy Wyg&, in Alfred Bester's _The Demolished Man_

From: "Paulo S. L. M. Barreto" <paulo.barreto@terra.com.br>
To: <AESround2@nist.gov>
Subject: Are the existing AES comparisons fair?
Date: Sun, 14 May 2000 10:01:44 -0300
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.00.2615.200
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2615.200

Are the existing AES comparisons fair?

We argue on the contrary.

1. Some criteria presented by the finalist submitters are obviously subjective, and hence hardly relevant.

For instance, comparing the relative cipher complexities on basis of lines of code, as the MARS submitters propose, is subjected to the characteristics of the notation or programming language in use. Indeed, a simple-looking line containing a 32-bit integer multiplication easily becomes a bunch of hacking on platforms that lack multiplication instructions.

Another example of subjective criterion is the "difficulty to analyze the cipher". The MARS team states, for instance, that the round function of Rijndael is difficult to analyze. This assertion is totally irrelevant, as it only reveals that some cryptanalysts could not find any way to attack the cipher (while others were able to find new results, as is the case for the Twofish team). In fact, should any of the AES finalists be considered "easy" to analyze, it would probably be broken by now.

2. Benchmarks were abused in cipher comparisons.

It's surprising to find that some submitters still claim that their proposal is the fastest on this or that platform, even though the evidence in contrary from all benchmarks available. Most times, such claims do only hold on a very specific platform and for a very specific implementation.

This is the case, for instance, for the MARS team claim that MARS is the fastest cipher when implemented in Java. However, the source of this information is not mentioned. Which implementation was used for the other ciphers? On which underlying hardware were the measurements performed for each cipher? (the bytecode may be portable, but it will certainly execute differently on a Java ring and on a Pentium III).

A curious example is the comparison presented by the Twofish team. Not surprisingly, Twofish is argued to be faster than the other finalists; however, an implementation tailored for a particular key (hardwired into the self-modifying code) is quite unusual to say the least, certainly impractical, and absolutely insecure, as now the security resides in the code itself. Perhaps this should be considered a variant of Twofish over a keyspace of cardinality one, not Twofish itself.

3. Some arguments put forward against candidate ciphers lack any reasonable foundation.

This is the case of the MARS team statement that Rijndael's has "a key schedule that makes it easier to mount power attacks", while no such attack is presented. On the other hand, this very issue is analyzed in detail elsewhere, and no such weakness is reported.

Also, the same team makes assertions on security margins without clearly defining the criteria they used to assess these margins. This contrasts, for instance, with the procedure adopted by the Twofish team, whose criteria are not only precisely stated, but also quite reasonable.

4. Sometimes, truncated or out of context statements are advanced.

This is quite misleading. For instace, the RC6 team agrees that analyses on reduced-round variants are flawed - after quoting a related key attack on reduced-round Rijndael, and omitting the very fact that this weakness is absent from the full cipher.

5. Other issues.

An argument advanced by the RC6 team as a response to the criticism that RC6 is not suitable for implementation on cheap smart cards is that such processors will be obsolete on the long term, so that only more powerful processors should be considered when comparing the AES candidates. Hopely this would put RC6 on an advantageous position. However, existing evidence suggests the contrary: RC6 has the worst performance on the IA64 platform, and is equally bad on the PA-RISC.

The strategy adopted by NIST -- namely, taking *all* existing analyses into account -- is likely to be the only viable way of correcting these problems, and compare the finalists with regard to their intrinsic (as opposed to their advertised) merits.

Best regards,

Paulo S. L. M. Barreto.

From: "Paulo S. L. M. Barreto" <paulo.barreto@terra.com.br>
To: <AESround2@nist.gov>
Subject: On the multiplicity of AES algorithms
Date: Sun, 14 May 2000 10:23:25 -0300
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.00.2615.200
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2615.200

On the multiplicity of AES algorithms.

It hardly makes any sense to speak of a "standard" if several highly dissimilar algorithms are chosen. It's mere illusion to expect that industry might afford the costly effort of implementing a second algorithm just in case something happens to the primary cipher. If the primary cipher is considered effectively strong by all accumulated evidence, there's no need for a backup cipher. If it's *not* considered strong enough, the psychological effect is devastating (who can expect any level of confidence in such an algorithm?). Furthermore, should the primary algorithm be broken there would again be just one AES cipher, unless three or more algorithm are chosen, making full AES compliance even more expensive (if viable at all). In practice, industrial applications are most likely to implement only one of the AES ciphers (interoperability issues dictate that this be the primary one), making the existence of alternative ciphers a theoretical fact irrelevant to ensure protection in the realm of those applications.

The suggestion to define the backup as the *same* algorithm with a larger number of rounds is a far superior idea. It's quickly, easily and cheaply achievable; in fact, it closely relates to the present industrial solution to the obsolescence of DES, namely, using 3-DES instead (with the advantage of doubling rather than tripling the processing time). As a side note, it's interesting that many people recommended using Serpent as secondary cipher because of its allegedly higher security level compared to the other AES finalists. However, this level derives precisely from its large number of rounds: 16-round Serpent is as secure as any other finalist. This provides clear evidence that a backup cipher consisting of the *same* algorithm with *twice* as many rounds would be felt as a good choice.

For the sake of comparison, the recent DSS revision is based on a totally different situation. First, RSA is too widely spread to be ignored; it's a de facto worldwide standard; including it in the revised document is mere acceptance of this. Second, ECDSA has clear advantages over conventional DSA; ideally, it should be a full replacement. But (this is the third point) this is obviously impossible to be achieved except in the very long term, so DSA should be preserved in the standard. Finally, there are not many choices for public key algorithms (as opposed to the proliferation of symmetrical ones), which makes the two standardization processes entirely different.

In conclusion, NIST should select a *single* AES algorithm. If a "backup" cipher is needed, just double the default number of rounds.

On this light, I would like to manifest my opinion that Rijndael seems to be the only cipher that performs consistently well on all platforms for which implementations exist (hardware, smart cards, 32- and 64-bit processors, JVM). Needless to say, its estimated

security seems to at least match that of any other finalist, and it's completely in the public domain.

Best regards,

Paulo S. L. M. Barreto.

Posted-Date: Sun, 14 May 2000 10:34:56 -0500 (CDT)
X-Sender: schneier@mail.visi.,com
X-Mailer: QUALCOMM Windows Eudora Pro Version 4.2.0.58
Date: Sun, 14 May 2000 10:33:32 -0500
To: AESround2@nist.gov
From: Bruce Schneier <schneier@counterpane.com>
Subject: AES Comment: The Hitachi patent

Hitachi has two U.S. patents, numbers 4,982,429 and 5,103,479, each of which is purported to cover "any" encryption system using rotations. These patents were filed in 1988. All of the five AES candidates use some kind of rotation, including Rijndael's ShiftRow operation. However, for what it's worth, it should be noted that Twofish can be implemented as a "straight Feistel cipher plus a final permutation, with rotations applied only within the round function, not in the Feistel XOR path.

The authors of Twofish were unaware of these patents until recently, but the notion that such a broad claim could be valid seems quite ludicrous. he following well-known pieces of prior art would seem to at least dramatically limit the scope (if not completely invalidate) any such claims.

1) FEAL (1987) uses a rotation in its round function.

2) Madryga (1984) uses a variable rotation in its round function.

3) DES (1977) uses a rotation in each round of its key schedule.

4) DES (1977) uses a bit permutation (of which rotations are a special case) in every round.

5) GOST (1989) applies a bit permutation (that is a rotation) in each round after performing its S-box lookup

6) The very words "rotor" and encryption have been linked for a long time (e.g., the Caesar cipher, and Enigma)

The concept of rotation in encryption was clearly neither novel nor unobvious at time these patents were filed. The fact is that EVERY microprocessor opcode has been considered for use in encryption, with rotation being just one example.

This particular example is a counter to the "IP attack" argument espoused by some as a reason to select multiple AES algorithms instead of a single one. It is most likely that IP attacks, if any, will be based on very broad and ambiguous claims (like those of Hitachi) that the patent holder attempts to apply to all encryption systems.

Niels Ferguson
Bruce Schneier
David Wagner
Doug Whiting
****

Bruce Schneier, CTO, Counterpane Internet Security, Inc. Ph: 408-556-2401
3031 Tisch Way, 100 Plaza East, San Jose, CA 95128 Fax: 408-556-0889
Free Internet security newsletter. See: http://www.counterpane.com

From: "Brian Wong" <makemoneyfast2@netzero.com>
To: <AESround2@nist.gov>
Subject: NIST AES Round 2 Comment
Date: Sun, 14 May 2000 16:31:10 -0400
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2314.1300

Comment on NIST AES Process:

While I agree almost entirely with Bruce Schneier's comment regarding Hitachi's asserted patent coverage of the AES candidates, I would like to comment on one specific statment made in that document, that being:

>All of the five AES candidates use some kind of
>rotation, including Rijndael's ShiftRow operation.

Rijndael's ShiftRow operation is more accurately described as a byte-wise permutation on the internal state of the algorithm rather than a rotation or "barrel shift." In most computer architecture contexts, the term rotation is usually used to denote an operation working on the individual bits of a machine word or smaller component thereof. No such operation appears in the specification of the Rijndael algorithm.

Brian Wong

From: "Tom Phinney" <tom.phinney@attglobal.net>
To: <AESround2@nist.gov>
Subject: Use of AES in Industrial Field Communications
Date: Sun, 14 May 2000 13:45:46 -0700
Organization:
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.00.2314.1300
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2314.1300

Background
==========
I am the current U.S. Technical Advisor for IEC/SC 65C, whose scope is Industrial Communications. I have been involved in the standardization of industrial communications since 1980, in IEEE 802 and in IEC/SC 65C, both as an editor and as a significant contributor. I am a mathematician by training, have considerable experience in design of software, hardware and low-power digital ICs, and have been peripherally involved in the cypherpunk community.

My comments are with respect to AES use in the context of industrial automation and process control. Industrial networking in the discrete manufacturing, batch and continuous process industries is characterized by large numbers of sensors and actuators (know as "field devices") connected through a hierarchy of concentrators (bridges, switches and intermediary controllers) to centralized control systems supervised by human operators.

The number of sensors and actuators in a large system frequently exceeds 10^4, and sometimes 10^5 devices. Many of these devices are conceptually very simple, such as limit switches, relays and temperature transmitters. Networking considerations include multi-device failure modes and cost of connection; in many cases this is complicated by a potentially hazardous (i.e., explosive) environment which imposes absolute limits on available power and stored charge.

In the most hazardous environment, which is the market-required design-point for most process control equipment, the available device power is on the order of 200 - 300 mW during periods of network transmission, and 30 - 100 mW during the majority of the time when a networked device is not transmitting. This power budget includes that required for the sensor or, less frequently, the actuator as well as any connected microcomputer, memory, communications circuitry, etc. Typical microcomputers in today's devices are 8-bit and 16-bit CPUs with some EEPROM, such as various models of the 6805 and 8051 and related and derivative 16-bit families.

Before the advent of digital networking in this environment, most field devices used pneumatic or near-DC analog point-to-point communications, with the device interconnections on physically-secured premises (e.g. a fenced access-controlled oil refinery). The advent of networked digital communications has led to higher bit-rate transmissions, either wired or now wireless, where eavesdropping from "outside the fence" is feasible. Indeed, one European refiner has reported discovering that arbitrageurs were eavesdropping on the signaled levels of its petroleum storage tanks and were playing the local spot market against the company.

Implications for AES
==========
The following comments are relative to current wired industrial field device networks and their likely extension to wireless regimes.

Industrial field devices use combinations of point-to-point client-server and point-to-multipoint publisher-subscriber connections, as well as point-to-point (unicast) and point-to-multipoint (multicast) multi-source to multi-sink connectionless communications. Most such messages are relatively short, consisting of approximately 10 – 40 bytes. In a wired environment, critical messaging should use source authentication and message integrity checking; in a wireless environment both authentication and MACs are a requirement.

Whether wired or wireless, business-critical information such as inventory levels and critical process measurements requires confidentiality. This is best applied on a presentation layer (field content) basis, with the data values encrypted and the remainder (communications protocol portion) of the message in the clear.

Different connections may require different session confidentiality keys, so a single device may be encrypting and decrypting concurrently under many keys. However, the existence of long-lived connections and associations makes precomputation of round-keys feasible, up to the amount of available RAM in the field device. Most such field devices can be expected to have at least a few kB of unused RAM (today, typically in a separate RAM chip), so this need not be a severe constraint on key setup. Thus key agility has some value in this environment, but less than it would have in a transactional environment where long-term relationships among communicating nodes is not the norm.

Field devices are both clients and servers. Thus assumptions that simple devices are not servers are incorrect . (Cf. letter to NIST on AES by Jeffrey Streifling, "Consideration of RC6", 1999.12.05)

Unlike most commercial battery-operated devices, which have limited energy storage but few restrictions on instantaneous power use, field devices designed for hazardous-area use have a strictly limited power supply. Software design strategies try to average power consumption and minimize peak usage. Encryption modes where the keystream material can be precomputed are a good fit for this power regime.

The AES security timeframe of 20 years would be considered short in the industrial environment. Many industrial systems operating today are well over 20 years old; some are in their 30s. For example, a large percentage of the electric power generation in the U.S. falls into this aged category. Typically, these systems undergo a substantial upgrade every few years, but wholesale replacement or discard of working devices is a seldom occurrence. The multiple key lengths currently specified for AES are a reasonable fit to the spectrum of threats envisaged for such long-lived systems.

Increasingly, field devices have a downloadable code store (e.g., flash PEROM) to handle issues of software protocol upgrade. This was not possible in the days of 12 - 18 V flash programming voltages, but new devices use such low voltages and energies that they can meet design requirements for hazardous areas (a field known as intrinsic

safety). Therefore, if a usable attack on the chosen AES algorithm were to be discovered, such devices could be upgraded with a replacement algorithm.

Other comments
==========
I recommend that NIST use a self-administering FSF-like CopyLeft licensing procedure for the AES to preclude restrictive derivative-use patents. I see this as partly an issue of "internet years" versus human years. We as humans cannot predict the specifics of the hot encryption uses over the next few decades, but the "wired economy" will suffer substantial harm if this public process is coopted for private gain, e.g., by the first person to claim the use of AES for digital cash or micro-payments or E-mail or something else which the patent examiners consider "non-obvious to one skilled in the art."

Some of the letters to NIST indicate concern about implementation errors. This seems unlikely to be a problem for the AES core algorithm, assuming that multiple reference implementations and an adequate set of test vectors both exist (as they do or will). Of much greater concern is the implementation of key management, which is outside the scope of the AES effort. In this latter area all five candidates are equal.

I believe that NIST should choose only one AES algorithm. A backup algorithm should not be specified now. Rather, defer such selection from the other four current candidates, or more likely run a new selection process, only if and when cracks (pun intended) start to appear in the strength of the selected algorithm. This deferral permits the selection of the backup to be informed by continuing research, and motivates that research, without imposing the inevitable "consider implementing the backup algorithm now" debates and security risks which an actual choice would engender.

Specifying a backup candidate now may actually increase the security risks if it is ever employed, because in many cases the implementation will already be in place if it becomes required, but will not have been tested adequately because of the backup algorithm's irrelevance at the time the code or hardware was developed.

My own preferences for the AES selection are TwoFish or Rijndael. The per-byte power requirements for multi-precision software multiplication and shifting of MARS and RC6 seem excessive for the above-described field device environment. Serpent seems excessively cautious.

In closing I would like to applaud the submitters of the 15 initial AES candidates, and the tremendous investment of the participating crypto community in their analysis and in continuing analysis of the five semifinalists. For this we should all be grateful.

The above comments are my own, and do not represent an opinion of my employer or of the U.S. industrial networking standards community.

Tom Phinney
Principal Engineering Fellow
Honeywell
Industrial Automation and Control
Phoenix, AZ USA

Date: Sun, 14 May 2000 23:47:27 +0100
From: David Crick <dacrick@cwcom.net>
X-Mailer: Mozilla 4.61 [en] (Win95; U)
X-Accept-Language: en
To: AESround2@nist.gov
Subject: AES Round 2 Comments

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

My views are:

* Rijndael should be selected as the one and only AES algorithm.

  - it is the only finalist that performs exceptionally well in
    all environments. Fast key setup, support for larger block
    sizes (essential for hashing), and an inherently parallel
    nature further mark it out as an exemplary modern block cipher.

  - safety margin aside (the authors still feel it is adequate),
    the cipher has remained secure throughout the evaluation period.
    Many people have expressed the view that the simple, "clean" and
    indeed elegant structure of Rijndael has proven advantageous in
    analysis and understanding.

  - multiple algorithms introduce complications. Ian Harvey's
    paper concludes that a primary algorithm with optional
    secondaries is the most acceptable of the multiple scenarios.

    Yet there is little difference between this and a single cipher
    when resources only permit one to be implemented, and multiple
    ciphers (to choose from) would in any case be likely in suitable
    environments. (If Rijndael does win, I expect to see Serpent and
    Twofish used in such circumstances.)

    If NIST and the community (who in the AES2 questionnaire were
    in favour of a single winner, with Rijndael and Serpent rating
    highly) cannot decide on a single algorithm, then how are the
    public supposed to? Not to mention all the implementation
    uncertainties that would arise in such a scenario.

* Addressing Rijndael's safety margin, if NIST feels that there is
  enough call to increase it, then they should do so now. I feel
  that an increase to 16/18/20 rounds for 128/192/256-bit key and
  block lengths respectively should satisfy such concerns, while
  still retaining adequate performance.

  For those still not satisfied (and also to counter breaks if they
  occur), perhaps the best situation is to have the number of rounds

as a parameter, so that higher figures still could be used which could nevertheless be decoded by all implementations. (A hard-coded lower limit is crucial here, to stop someone [maliciously] requesting lower than 10 rounds - or even zero!)

* Implementations should use 256-bit keys unless there are serious reasons why this is not possible. Conceivably, technology such as quantum brute-force key searches could come into play during the lifetime of AES.

* Triple-DES should be promoted and run concurrently with AES as a FIPS, so that organisations and individuals who do not feel confident with a relatively new algorithm can go for a more established and trusted one.

* Similarly, 3DES could be used as an emergency replacement should AES be totally broken. This makes most sense as 3DES is already fielded, and implementation expertise, etc. already exists.

* NIST should use the (estimated) one year period between the selection of the AES winner and the FIPS certification to encourage further analysis. The reduction of the fifteen Round 1 candidates to the five for Round 2 facilitated much more detailed analysis on those that were left. Some people have commented that another year's study is needed for the finalists. This year of "dead" time could therefore be used highly profitably in a concentrated effort on a single algorithm, further boosting confidence.

* If AES suffers a critical break (one where data is actually at risk, or if a valid "academic" attack seems like it may become practical in the future lifetime of AES), then NIST obviously needs to take action. Some options I suggest are:

  - increasing the number of rounds if this will stop the break (with Rijndael this is easy to do).

  - patching the cipher if possible, although this will need to be done with care and with further analysis (the move from SHA to SHA-1 is an example of this).

  - replacing the algorithm with one of the other finalists if they are immune (intellectual property is crucial here).

  - replace with a more up-to-date cipher (ideally one that has already been analysed for some time).

- initiate another call for candidates for a new standard (the AES process was initiated when DES was recognised as being inadequate/vulnerable).

* For NIST's Round 2 Status Report, a summary table of attacks for each finalist should be given. This would detail for every attack currently known:

  - the number of rounds broken

  - the key size it relates to

  - the complexity of the attack (in terms of time, data, etc.)

  - the type of attack (e.g. Square, partial sums, related key)

  Such information would need to be compiled from the original candidate submission documents, Round 1 / Round 2 papers and comments, plus other "outside" sources (eg FSE, SCIS2000).

  An example of the type of table I am referring to can be seen in Table 1 of "Improved Cryptanalysis of Rijndael" by the Counterpane team.

* NIST should adopt the GNU copyleft licensing for AES. This would ensure that all implementations, optimisations, etc. are free for anybody and everybody to use, which I believe is the true spirit of AES.

* For the record, my ranking of the finalists, best to worst, is: Rijndael (outstanding); Serpent, Twofish (both very good); RC6, MARS (both have poor performance off the reference platform, and also design concerns).

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

iQEVAwUBOR8q+4VnnSUi1cepAQFmVwf5Aa9bRQpkr9tlhDuIH6eOp4HlaoB6MUll
DoJtozW6u2MvkijZF1gQCUpkIGt5ia+cvWoCTIAhMAS3pTyhKBPzCvzrC+ajjEQk
/FlYkcWr53wxNCyTk6zL1fPabSh4KrPVnheYOckVxQWWsMR/glNZ7R4zVmbIy4i4
FliMaaj1Zp+8dzimICSPOYySCYK19hZDe2hSh0h9h0RIBzdtPBEmVdTSDmCd+81Y
eoqgUTqUcQgyx0rYqThwVethktBscvu1wtZaOlkd+usrw2IdETkr0zKNVzZ0zG/u
prqgw3jG1QJBH/eeWJ863H0FynGopitjwrY50ZxGl9Skg0H6T/JbhA==
=wFcs
-----END PGP SIGNATURE-----

From: "Yaro Charnot" <ycharnot@identikey.com>
To: <AESround2@nist.gov>
Subject: AES Round 2 Finalists
Date: Mon, 15 May 2000 01:08:53 +0100
X-MSMail-Priority: High
X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)
Importance: High
Sensitivity: Company-Confidential
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2314.1300

As a computer security expert, software engineer, software reverse engineer, Master of Cryptology, former computer cracker and hacker with many years of experience, I want to use my opportunity to influence your decision on the next Advanced Encryption Standard.

My points are:

1. The security of the products I have been cracking for years lied mainly on the implementation, not on the number of rounds or possible theoretical attacks on the cipher. After personally evaluating all AES candidates, I can report that only two ciphers qualify as simplest to implement not leaving much room for mistakes: Twofish and RC6, with RC6 being amazingly simple and requiring the least IQ of the programmer to implement it, debug it and deal with it.

2. My personal never published (company's internal) analysis of R1 AES candidates in June 99 using personally designed and publicly available randomness testing tools (like DIEHARD tests) showed that two algorithms stand off the rest. Those are again Twofish and RC6, with Twofish leading in speed on most of the processors. They offer the lowest price (speed and memory requirements) for the same level of security.

3. All AES candidates have different properties, but in software implementation (in 95% of the cases performed by programmers with very little understanding of cryptography) one characteristic plays the second most important role: flexibility of the key size. RC6-like algorithms offer it allowing the entire large output of public key algorithms like 4096-bit Diffie-Hellman to be used as a key without a need to design an extra special "compression" function for this purpose leaving poor software developers on their own in that design. They will most probably use the lower 128 or 256 bit of the common key, which are the least secure as PK cryptographers know. But in some cases this flexibility is irrelevant and other characteristics play more important role:

4. Hardware implementations? Again, Twofish wins, RC6 follows, the rest of the algorithms is behind only confusing developers with their complexity.

5. Pick a 15 y.o. student, show him all the algorithms and ask him which ones he likes the best and would use? He'll point at RC6. Why? It's naturally and geniusly simple and easy to understand, one can hardly make a mistake implementing it.

6. Pick an experienced Unix software developer, show him all the algorithms and ask him which one he likes the best? He'll point at Twofish. He trusts its predecessor

Blowfish and his main concern is inter-platform effectiveness and compatibility. Problems like byte order, absence of rotation functions on different processors, differences in register size, memory alignment requirements, etc. Twofish is fastest on 8-bit processors as it is on 32-bit or 64-bit processors regardless of byte ordering or instruction set.

7. Even if you pick only one algorithm as AES, all the software developers will still want to support both RC6 and Twofish because they are already doing it. Why are they doing it? Because they had a choice and they choose the algorithms easiest to implement, the algorithms most effective, the algorithms most flexible, the algorithms most trusted. Trust is the key to the success of their software. They trust RC5 and they trust Blowfish, so they all trust RC6 and Twofish more than all the other algorithms combined together. I do too.

8. In some countries in the world people will not trust Twofish or RC6 because they are a US product. An encryption algorithm from a country that has been imposing bans on strong encryption for years. On contrary in the US people will most likely favour a US product not trusting much a foreign algorithm to protect their most valuable assets. The USA is the world's leader no matter how good or bad it is or its products are. It's rich and strong. The whole world is watching and is trying to follow. Give the world a choice and they will love you for that. Serpent suits well for it although it lacks the flexibility and simplicity RC6 and Twofish offer.

9. I myself don't have a preference in any of the algorithms. They are all secure enough to protect from a street hacker or even a small or large business trying to hack a network or a financial software. I can implement even DES in such a way that it will be uncrackable for NSA. The security of the algorithms is my least concern. The beauty and ease of implementation and flexibility of the algorithm is. Only two algorithms qualify: RC6 and Twofish. Although if I was to choose a HASH function, I'd stick to Tiger due to its natural simplicity and my trust to Eli Biham's and his colleagues with Russian names cryptanalytic skills. So Serpent would be my next trusted choice.

10. There are other less significant points like one round of RC6 can be and will be used as a beautiful non-linear chaining function replacing simple easy-to-attack XOR, like both algorithms can be and will be combined together. Paranoid programmers will want to encrypt their data first with Twofish and then with RC6 or something else on top just in case one of the algorithms is broken. And they will be right. The skills of the majority of them are very low.

Unfortunately cryptographers and cryptanalysts do not develop security software. I represent the rare 0.001% of those developers and while I personally don't care what the next AES will be (my products will still be more secure than everything else around), the rest 99.999% might get stuck with a big pain for years. Please be careful.

The conclusion is I ask, demand and beg to include both Twofish and RC6 (maybe Serpent as well) in AES for all the software and hardware developers, copy-protectionists, information security advisors, software reverse engineers, white hat hackers and beginners, as someone who is qualified enough to see and exploit the real insecurity of all so-called security software every day of his life, someone from the battlefield.

- Yaro Charnot

Identikey - The Key To Internet Security

Yaro Charnot
Chief Security Advisor
Identikey (Australia) Pty Ltd
143 Coronation Drive
Milton QLD 4064
Phone: +61 7 3236 5050
Fax: +61 7 3236 5850
E-mail: ycharnot@identikey.com
https://www.identikey.com

X-Authentication-Warning: everest.cs.umbc.edu: aselcu1 owned process doing -bs
Date: Sun, 14 May 2000 22:58:35 -0400 (EDT)
From: Ali Aydin Selcuk <aselcu1@cs.umbc.edu>
Reply-To: Ali Aydin Selcuk <aselcu1@cs.umbc.edu>
To: AESround2@nist.gov
Subject: comment on RC6

A general comment on RC6:

I think RC6 is the best publicly analyzed algorithm after DES, considering the amount of public analysis received by RC6 and RC5. Having so much analysis done on such a small piece of code, and still having no significant weakness found, I think, is an important plus for RC6.

Regards,

Ali

X-Lotus-FromDomain: NRTA
From: "Ken Tindell" <ktindell@realogy.com>
To: AESround2@nist.gov
Date: Mon, 15 May 2000 09:41:42 +0100
Subject: Comment on round 2 AES algorithm selection

I wish to comment on the criteria for selecting the AES algorithm. Aside from the issues already well covered and understood, I would like to emphasize a key area:

The efficiency in terms of code space and working RAM must be very high. The smartcard market has been discussed already, but there are other domains that require high memory efficiency: mass-produced embedded devices. It is quite common now for embedded computing in mass-produced devices to be sophisticated, and yet to be implemented in small, low-cost microcontrollers. I give the example of automobiles: there are several new automobiles with more than 20 microcontrollers on-board, each with Flash memory and hence able to be re-programmed. Several manufacturers of components (e.g. engine management systems) wish to keep their software secret when downloading and programming devices, and to ensure that certain data is protected against tampering (e.g. engine calibration tables). This has led to the use of customized low-resource encryption. However, there are considerable risks in adopting proprietary encryption algorithms, which would be avoided by adopting a memory-efficient AES algorithm.

>From an examination of the candidate algorithms it would seem that Twofish is best placed to be implemented in low-resource micros.

Best regards,

Ken Tindell

Sender: contini@maths.usyd.edu.au
Date: Tue, 16 May 2000 01:01:11 +1000
From: Scott Contini <contini@maths.usyd.edu.au>
X-Mailer: Mozilla 4.7 [en] (X11; I; SunOS 5.5.1 sun4u)
X-Accept-Language: en
To: AESround2@nist.gov
Subject: AES comments

Dear NIST,

I am writing to make my recommendations for the choice of the AES. My recommendations are entirely from a security analyst perspective.

I am most supportive of RC6 being chosen for the AES. It is quite clear that RC6 is the easiest cipher to analyze. This analysis is aided by 6 years of public research on RC5. I think many people have overlooked the value of this, and many others just do not have the experience with analyzing algorithms to appreciate how important such results are.

For instance, the analysis of RC6 had the results of Biryukov and Kushilevitz to build upon, which are the best results today for attacking RC5. Their results were based upon earlier work of Knudsen and Meier which was based upon earlier research by Kaliski and Yin. Biryukov and Kushilevitz were able to mount very enlightening attacks against RC5 by using a more general notion of difference than the earlier researchers. The security analysis of RC6 considers their general notion of difference with respect to both subtraction and exclusive or. Similar experience was gained from linear cryptanalysis and other attacks against RC5. I have not seen any other AES submissions with such general analyses as RC6, nor any analyses that are based upon so much public research.

RC6 is a cipher that is so simple that it is inviting to analyze. However, the most significant new result by Knudsen and Meier closely agrees with the security analysis originally submitted by the RC6 design team and myself. The lack of results on such an easy to analyze cipher should be strongly considered, since it suggests that RC6 is the most well understood and has the most accurate security analysis.

I recommend to NIST to avoid new ciphers that are not well understood and are difficult to analyze. The two ciphers that come to mind are MARS and Twofish. While these ciphers may (or may not) appear to offer good security at first, it is important to realize that analyzing such ciphers takes many years (even analyzing a simple cipher like RC5 took a couple years to get strong results) and thus we cannot be confident of the true security that the ciphers offer. It will likely take several years before the research community has a good feeling for how secure these ciphers really are. Ciphers that are based upon years of cryptanalysis experience should weigh more than newly designed, complex ciphers when choosing the AES.

Thank you for taking the time to read my comments,

Scott Contini

From: "Walker, Jesse" <jesse.walker@intel.com>
To: "'AESround2@nist.gov'" <AESround2@nist.gov>
Subject: Feedback to NIST AES Questionaire
Date: Mon, 15 May 2000 08:04:52 -0700
X-Mailer: Internet Mail Service (5.5.2448.0)

This provides feedback from Intel's Network Communication Group to NIST's AES questionaire. Our focus is on the impact of the AES standard on mass-deployment networking markets. If you need any clarification of any of our points, please contact me.

Jesse Walker
Intel Corporation
Network Communications Group
2111 N.E. 25th Avenue
JF3-448
Hillsboro, OR 97214
(503) 712-1849
jesse.walker@intel.com

1. NIST has stated its goal that the AES should specify an algorithm(s) that will provide strong security for protecting sensitive data for 20-30+ years. How would the selection of one versus multiple algorithms affect the likelihood of achieving this goal?

The selection of a single algorithm would contribute far more to the goal than multiple algorithms. The reasons are sociological, not technical.

The first reason is market psychology. If NIST were to select more than one algorithm, the market would ask what's wrong with the lot of them. The public would incorrectly conclude that NIST sufficiently trusts none of the algorithms, or they would have chosen the one most trusted as the winner. This is the "every contest has to have one winner" syndrome, and it is a real phenomenon.

The second reason is the criteria to select which algorithm to employ in a particular circumstance are too arcane for even cryptographers. Compelling a choice would overwhelm most people; ask your great Aunt Nellie when she intends to protect her data with Rijndael and when with Serpent. This is not just rhetoric, because cryptographic tools have flooded far beyond the province of security professionals. And we in private industry have demonstrated limited ability in automating the selection process to the degree needed to win wide-spread acceptance of any security standard with multiple algorithms; for instance, almost all IPsec GUIs ask users to select between HMAC-SHA-1 and HMAC-MD5. However absurd this situation might be, it is likely to be repeated with a multi-algorithm AES standard.

Finally, multiple algorithms would increase security costs. Multiple algorithms would marginally increase hardware, software, and testing costs for equipment vendors. More importantly, the deployment costs for multiple algorithms would also be higher than for equipment based on a single algorithm, because the deploying organization has to elaborate and configure policies selecting the appropriate algorithm, and then debug and maintain their configuration. None of these costs are trivial for deploying organizations,

and they are likely to become only more of a burden as the shortage of IT personnel continues to accelerate each year.

This reasoning leads us to believe that, whatever its technical merits, a multiple algorithm standard would delay AES acceptance and deployment, thus extending the widespread usage of 40-bit RC4 and 56-bit DES. Data protected in this way are far more vulnerable to recovery and compromise than data protected by any of the AES candidates.

Neither the standard grade school cirriculum nor the lessons from everyday life include an education in cryptography, and so we cannot expect the public to reason correctly or even be equipped to make informed choices in this arena. The technologically elite will use multiple algorithms, regardless of what NIST does, because they can, and everyone else will use one algorithm or nothing at all, because they aren't prepared to do anything else.

2. If only one algorithm is selected, how will sensitive data be protected (with AES-comparable security) in the event that the AES algorithm is broken?

History suggests this depends completely on the technical sophistication of the individuals and organizations affected. In the eventuality of a break, the overwhelming majority of people would continue to use the compromised algorithm, and they would resist change until forced to do otherwise (e.g., by an insurance provider refusing to renew a policy without an upgrade). Everyone has known for years that 40-bit RC4 and 56-bit DES do not provide any substantive privacy, but both still enjoy widespread use. On the other hand, the cryptographic literati routinely employ many algorithms, and promptly migrate from old favorites to newer ones with any hint of compromise. Compromise thus leads to interoperability failures and islands governed by competing algorithms, not to needed behavioral change in the user community. Making multiple algorithms available will do nothing for most already encrypted data, because most people will not re-encrypt. Multiple algorithms only make sense in a context of a properly trained and motivated user community, and the larger community is neither so trained nor properly motivated.

There are lots and lots and lots of good symmetric key algorithms. The people and organizations who really care about their data being compromised by cryptanalysis will protect their data by a new algorithm when their first choice might be broken. Most people won't. Selecting two or two thousand algorithms for AES will not change this fact.

3. What type of attack on an AES algorithm would be sufficient to "break" the algorithm? A practical attack or a purely theoretical one? That is, when is an algorithm considered to be broken? Another way to think of this is to consider what sort of attack would cause users to lose confidence in the AES algorithm(s).

Given the continued wide-spread acceptance of 40-bit RC4 and 56-bit DES, it is difficult to imagine a scenario powerful, practical, imminent, disarming, and threatening enough to even make a blip on the public's radar. The cryptographically sophisticated always respond when a potential for liability threatens, but few other members of society care or even notice.

In a world where insurers underwrite losses accrued by organizations attaching today's commercial operating systems to the Internet, raising the question of what will constitute a break in the public's eye seems to miss the real threats. 40-bit RC4 is still in wide-spread use not because it is safe, but because hackers need not bother breaking its keys; simple buffer overrun attacks, not cryptanalysis, are their preferred route. Commercial operating systems cannot become more secure, for the simple reason they are getting more complex, so it seems unlikely the present balance between hackers and their prey will soon change. Thus, while practical attacks are easy against the weak cryptographic algorithms deployed today, by and large this is a concern only to a very narrow portion of the public, because almost all systems offer so many much easier attacks. Those who understand the issues already know lots of algorithms to fall back to in case of a compromise.

Why force everyone to choose among different bank vault doors to install on their yurts? The few people who actually possess vaults already know plenty of vault door suppliers. With the exception of this tiny minority, the community cannot and will not use a standard that makes them choose among options they cannot readily distinguish.

4. If multiple algorithms are selected, what effect would this have on interoperability? Note that there are currently multiple algorithms available which may provide confidentiality and other security services.

Multiple algorithms have always undermined interoperability. History amply testifies to this.

The success of the original single algorithm DES standard is evident. The success of TLS was likewise forged by its reliance on a single weak algorithm, 40-bit RC4. IPsec's comparative lack of success may be attributed in part to its explicit embrace of multiple algorithms. Even though there are many good reasons to promote DSA and EC-DSA as equal partners with RSA, so far the market has adamantly rejected the new-comers, more because of the confusion their deployment would engender than for any apparent cryptographic concerns: RSA got there first.

The market overwhelmingly and emphatically tells us over and over again its wants a simple choice: encryption, YES or NO. People want this whether or not it is the ideal choice cryptographically. People without deep professional training and experience in security realize they are incapable of dealing responsibly with the choices a more complex standard would impose.

With the advent of PCs and the Web, cryptography is no longer the exclusive sandbox of security professionals. As reiterated before, those who are so inclined will use multiple algorithms regardless of whether NIST selects 1 or 1 million algorithms as the AES standard, while even two algorithms will be a deployment barrier for everyone else. Selecting a single algorithm will maximize interoperability by giving the newly enfranchised a tool with which to push back against the small cadre of experts demanding more.

5. If multiple algorithms are selected, how many should there be?

This is the wrong question; there should not be more than one.

The availability of two algorithms is no different than ten thousand for the vast majority of the users, because they have no relevant means to distinguish among more than one. Indeed, a multi-algorithm standard presupposes both well-defined criteria specifying when to use one algorithm over another, and that the community relying on the standard is trained in the criteria. Neither presupposition is valid. Without a mechanism analogous to the driver's license testing, an applicable common school cirriculum, or the like, correct application of the criteria by the wider community is impossible.

Many security professionals will dismiss this kind of argument as specious, because it ignores any examination of "real" security needs. That is exactly the point, however. The diffusion of cryptographic tools to a vast cross section of the public relegates technical arguments to secondary importance. The cryptographic community seems convinced most of the AES finalists are essentially sound, and that any one of them probably provides vastly greater security than any encryption algorithm used commercially today. All the technical arguments needed have thus already been made to winnow the field to five and then rank them. This has resulted in the best judgement technical consideration can render now. Given this situation, arguments maximizing deployment and use have to become the overwhelming test of acceptability, and this argues for selecting just one algorithm.

6. If multiple algorithms are selected, what sort of guidance or standards would be useful?

The assumption behind the question-that it is technically feasible to assemble reasonable guidelines-is questionable at best; the cryptographic community itself hasn't even reached rough consensus on when to use one of the candidate algorithms instead of another. The nearest thing to a consensus is that the people who attended the AES conferences tend to rank their preferences for algorithms as Rijndael first, then Serpent, followed by Twofish, RC6, and MARS. If professionals cannot agree on a finer scale, what are the amateurs supposed to do? With the present state of knowledge, it is infeasible to issue meaningful guidelines for selecting among the algorithms.

Lacking meaningful guidelines that crisply and easily divide one usage from another, selecting more than one algorithm can do nothing but sow confusion, degrade interoperability, and delay the adoption of AES. Someone always finds a way to break every algorithm, and we don't know when that will happen with any of the AES algorithms. But none of this will matter if people don't adopt and use AES. Multiple algorithms are a deployment barrier for that vast majority of potential users. The threat of a break remains a theoretical attack in the future. The threat of confusion and misapplication that multiple algorithms engender is real today.

Those sophisticated enough to weigh the technical tradeoffs for themselves can and will employ multiple algorithms, regardless of the number of algorithms NIST incorporates into AES. The broader market lacks this sophistication. Indeed, it is implausible that any guidance NIST or anyone else can develop for selecting among multiple algorithms based on existing knowledge would have any practical value or utility outside a narrow community of security professionals.

7. What about the speed versus security margin tradeoff?

We would be more comfortable with four more rounds for Rijndael, or doubling the number of rounds for RC6, but then those of us who bother to encrypt our personal data or use a VPN still employ 3DES and IDEA for these functions, so there is some hypocrisy in our response. Even unchanged, all the candidates already appear stronger than what we use. As time goes on, however, the candidates will only become weaker as they undergo further cryptanalysis, so extra security probably wouldn't hurt.

At least in the networking arena, customer first want to know if we implement a standards-based algorithm, so they know our solutions are interoperable and have undergone adequate technical review. After receiving a satisfying answer, their next three questions are how fast is the solution, when we will make it faster, and when we will deliver something even faster. The market is very clear and precise: speed wins over security margin. We don't agree with that tradeoff, but that's what the market is saying, and it speaks emphatically with its dollars.

The one area where the speed-over-security contingent argument is compelling is in the usability arena. Slow implementations detract from usability as the user community has broadened. Most people are simply not willing to wait on "poor" performance. It does not appear AES will present any problems in this arena, however, except perhaps for some software implementations of Serpent or MARS.

8. How important are low-end smart cards and related environments when selecting the AES algorithm(s)?

We offer no opinions on this topic.

9. What is the relative importance of hardware vs. software performance in the selection of the AES algorithm(s)?

This question misses the point. There will always be a mix of both, so both will be important, and it is difficult to credulously assign a relatively importance of one over the other. If you talk with end users and the vendors who build for them, the answer will be software, because processor performance is growing faster than the demands placed on end-user systems, and end-user performance is a driving usability concern. If you talk to vendors of choke point devices like servers and network plumbing, the answer will be hardware, because network line rates are growing faster than processor speeds. We have to build all the parts of the system, so the answer has to be both are equally important.

10. What modes of operation should be available for the AES algorithm(s)?

The four DES modes should be support, as doing so will simplify the migration from DES to AES. Counter mode is appealing in that it appears to provide marginally better security than CBC, and it is obvious how to parallelize this. A mode combining encryption with data authentication, like Gligor's PCBC mode, would also be attractive.

From: DWHITING@hifn.com
To: AESRound2@nist.gov
Subject: Comments on email from David Oshel
Date: Mon, 15 May 2000 09:35:58 -0700
X-Mailer: Internet Mail Service (5.5.2650.21)

Mr. Oshel has posted a comment noting that Twofish was very slow on his Macintosh PowerPC machine. His numbers made no sense to me, since Twofish was FAR slower even than Serpent in software, which has not been the case on any other platform (even Eli's machine <g>).

After corresponding with him, it was determined that his test application encrypts exactly ONE block before re-keying. However, his initial numbers we taken using the Twofish "full keying" option implemented in Brian Gladman's code. I recommended that he use instead the "zero keying" option of Twofish, which was intended for just such "non-bulk" applications. Within a few hours, he responded to me as follows:

"Thanks for your help on this! After the relevant code changes, Rijndael, Serpent and Twofish all clocked in on the same test with the same start (seed = today's Julian date) at 13 seconds. If I'd been checking 0.1s I might have seen a difference."

Thus, the problem was not a Twofish algorithm problem, but how he was attempting to use it, in a somewhat unrealistic environment. Since he apparently has not yet posted a clarification on this issue, I wanted to make it clear that the issue was resolved to his satisfaction.

X-Lotus-FromDomain: CERTICOM
From: "Don Johnson" <djohnson@certicom.com>
To: AESround2@nist.gov
Date: Mon, 15 May 2000 14:22:39 -0400
Subject: Don Johnson's Final Thoughts on AES

NIST,
Here are my final thoughts on AES, contributed to NIST as grist for the mill:

1. In the AES3 summary poll, many voters indicated they wanted a single winner or possibly a single winner with a backup if patent concerns or security concerns surfaced later. It is essential that NIST listen to all potential users of AES and attempt to make the best decision that meets the needs of ALL members of its constituency. Sometimes a decision needs to be made that is NOT the most currently popular decision; this is the essence of leadership and often takes courage.

It is important that NIST not let any (admittedly unscientific) polls be a large determining factor in its decision. Rather, I suggest the best use of the results of the polls is in recognizing that NIST must explain the rationale behind its final decision as much as possible. Inquiring minds want to know!

2. Each of the summary papers by the inventors of each finalist algorithm give advantages of the algorithm according to certain criteria and tries to put their algorithm in the best light possible. Each paper tries to give good reasons for selecting their algorithm as the finalist. In effect, each summary paper can be seen as contending that the author's rationale should be used by NIST to decide the AES contest. This means that NIST could justify the selection of ANY finalist simply by adopting the criteria and rationale of the corresponding summary paper as its own.

One problem with accepting any particular criterion as critical is that often the best choice under one criterion is not so good under another criterion. Using performance as examples: RC6 seems to be fastest on a modern PC but seems slowest on some processors without support for multiplication. SERPENT seems to be fastest in hardware but is often slowest in software. As another example, for some applications, key agility is very important; for others, the effect of key agility is negligible. How is NIST to make a choice regarding which criteria are more important that others?

Rather than NIST attempting to choose which optimality criterion/criteria is/are MOST important, NIST could instead choose to adopt the anti-pessimal criterion, which says to pick an algorithm which is not bad anywhere. According to the anti-pessimal criterion, RIJNDAEL is a suitable AES winner. This was noted at the conference by Vincent Rijmen using somewhat different words. It is important that the many advantages of NIST adopting the anti-pessimal criterion not be overlooked, hence I am restating it here.

3. If NIST believes that it needs to choose some optimality criteria, my perspective is that suitability for use on constrained devices should be THE critical criterion. This is where the AES decision can make or break the potential for new functionality in a future product or solution. To be specific, on a modern PC acting as a client, it appears that any finalist would be acceptable. On a server, one can add more hardware if

performance is a concern. On a constrained device, the requirement to use an algorithm that is not as suitable as another can make or break the feasibility of an application for the device, raise the price point, etc.

Some say that future advances and Moore's law will overcome any current limitations on processors, but Moore's law has historically manifested itself in two ways (A) faster computation for a fixed price as newer chip designs pack more power and (B) cheaper computation for a fixed time as older chip designs become cheaper. It is important to note that the crypto engine will not be the only code running on a constrained device, applications will also. My concern is that selection of a less suitable algorithm will have a cascading ripple effect in two ways: (A) Internal: The constrained device will need to be larger, more costly and/or less flexible. (B) External: Constrained devices are expected to proliferate and become ubiquitous. As a first approximation, essentially all crypto engines will be on constrained devices. This simply means that any negative consequences of NIST's AES decision in regards to constrained devices packs a double whammy.

>From a constrained device viewpoint, the selection of at least one algorithm from the set of (RIJNDAEL, SERPENT, TWOFISH) as an AES winner seems advantageous for the following reasons:
A) MARS and RC6 appear to require more RAM for initialization.
B) MARS and RC6 require support for multiplication to fun fast. On cheaper processors, this function may not exist and will need to be emulated.
C) MARS and RC6 appear to be less key agile. This becomes of greater relative importance as shorter messages are encrypted or when the cipher algorithm is used for hash purposes.

4. Adi Shamir made the suggestion at AES3 that it would be acceptable to him for NIST to flip a 5-sided coin to determine the winner. This does not seem to me to be the best way for NIST to proceed. However, I suggest that Adi's idea can be used with merit in the following way: NIST (in its collective mind) can PRETEND to flip a coin and contemplate the likely results if any of the five finalists were selected as the/a winner. If, while the coin is in the air (so to speak) NIST discovers that it wants the coin to fall in a certain way, then this can be a way to tap into subconscious and unconscious thoughts that may not at first be apparent to the conscious mind. In other words, if while the coin is in the air, NIST hopes it would come down in a certain way (or at least not come down in a certain way) then I suggest it would be fruitful to explore further why this is so.

When I personally conduct the pretend coin flip exercise above, my results are as follows:
A) At the minimum, NIST should select a primary algorithm and a secondary algorithm.
B) RIJNDAEL becomes the primary algorithm, being the anti-pessimal choice and suitable for constrained devices. As such, it would become the default FIPS symmetric algorithm and used unless there is a GOOD reason for it not to used.
C) SERPENT becomes the secondary algorithm, able to be used by choice for a specific reason, such as super AES (multiple algorithm encryption) or hot backup (if RIJNDAEL should be discovered to have a flaw), speed for hardware link encrypters (as they only talk to themselves), etc. The point is that there would need to be a GOOD reason to use SERPENT; but if there was an explicitly-stated good reason, it would be allowed.

Don B. Johnson, Certicom

From: "Kaliski, Burt" <BKaliski@rsasecurity.com>
To: "'AESround2@nist.gov'" <AESround2@nist.gov>
Cc: "'mrobshaw'" <mrobshaw@supanet.com>
Subject: Patent waiver for modifications to RC6
Date: Mon, 15 May 2000 16:12:19 -0400
X-Mailer: Internet Mail Service (5.5.2448.0)

RSA Security has already waived any license or royalty payments, no matter which algorithm is chosen as the AES. See http://www.rsasecurity.com/rsalabs/aes/rc6_patent.html.

As requested by NIST, we can therefore confirm that this policy would also apply if a version of RC6 were selected with different parameters to those recommended in the AES submission.

-- Burt Kaliski

Burt Kaliski, Chief Scientist and Director
RSA Laboratories - http://www.rsasecurity.com
20 Crosby Drive, Bedford, MA 01730 USA
+1 781 687 7057; fax: +1 781 687 7213; bkaliski@rsasecurity.com

X-Server-Uuid: 7edb479a-fd89-11d2-9a77-0090273cd58c
From: "Schroeppel, Richard" <rschroe@sandia.gov>
To: "'aesround2@nist.gov'" <aesround2@nist.gov>
cc: "Schroeppel, Richard" <rschroe@sandia.gov>,
"'rcs@cs.arizona.edu'" <rcs@cs.arizona.edu>
Subject: comment for AES cipher selection
Date: Mon, 15 May 2000 14:18:37 -0600
X-Mailer: Internet Mail Service (5.5.2650.21)
X-WSS-ID: 153E8415129311-01-01

Here's my analysis of the AES cipher selection.
There are some equations near the end of Appendix D which
look best in a fixed-width font.

Rich Schroeppel rcs@cs.arizona.edu rschroe@sandia.gov

AES Comments

May 15, 2000
Rich Schroeppel
Sandia National Laboratory
University of Arizona
rschroe@sandia.gov

Disclaimers:
As should be abundantly clear, the opinions expressed in this letter
are mine alone, and are unlikely to be official positions of any
organization. I submitted a cipher to Round 1 of the AES process; it
was not selected for Round 2.

Abstract:
I discuss the five AES candidates, and suggest various improvements.
NIST should send all five ciphers back for rework, and hold another
conference in six months to analyze the revised ciphers. Failing
that, I reluctantly recommend Twofish as primary AES and Serpent as
backup AES.

Summary of Most Important Recommendations:

None of the AES candidates is completely satisfactory. NIST should
return all five candidates for curing of deficiencies, with a two
month deadline. Another conference should be held in six months to
review the revised candidates. Deficiencies are listed in Appendix A.

No cipher should be selected with a modified number of rounds or other revisions, without further public review.

Among the present unmodified candidates, Twofish is the best choice as primary AES. Serpent should be an optional backup.

CBC mode must be replaced. I suggest LFSR-counter mode, described in Appendix C.

A full list of recommendations precedes Appendix A.

1. Introduction

NIST is searching for a block cipher to replace DES. [1] Three conferences have been held to evaluate the candidates. Fifteen candidate ciphers have been winnowed down to five. Many algorithms have been attacked, defended, implemented on multiple platforms, and measured and simulated in mushware and hardware. Public comments have been solicited and offered. The time is at hand for a final decision.

This note argues that we now understand the problem better, and should give the cipher teams suggestions for improvements. The final decision should be deferred six months.

Section 2 addresses the question of appropriate security margins, concluding that RC6 and Rijndael have insufficient margin. Section 3 argues that hardware cost has been neglected by most of the teams. Multiplication as an inner loop operation is unacceptable. Key expansion is too difficult for some ciphers, and hard to run in the decryption direction for most candidates. Cipher Block Chaining mode should be replaced. Section 4 discusses other issues, including the AES process. Section 5 contains a list of recommendations. Appendix A lists recommendations for each candidate. Appendix B contains my guesses about future computing power. Appendix C explains an idea for replacing CBC mode. Appendix D discusses my concerns with Rijndael, and offers some suggestions for strengthening it.

2. Security

"Prediction is difficult, especially of the future." -- Niels Bohr

The most important criterion in cipher selection is security. All the candidates claim to be "secure enough", so that secondary considerations such as speed and implementation cost become tie-breaking factors.

None of the ciphers has been broken, in the sense that someone using one today would have reason to fear for the privacy of his traffic. The arguments center around appropriate security margins, and how much

weight to give to infeasible academic attacks against reduced-round versions of the candidates.

The Importance of Academic Attacks

An attack is academic when it represents no real threat to traffic encrypted with the cipher. Examples are attacks that require 2^100 plaintext-ciphertext pairs, or an attack that distinguishes the cipher from a random permutation, or demonstrates a failure of some other theoretical criterion. Most attacks against ciphers are academic. All the reported attacks against the AES candidates are against weakened versions of the ciphers, usually with the number of rounds reduced.

Academic attacks are the best predictors we have of real-world attacks.

The history of cryptography confirms that attacks improve. Progress is so erratic that it's impossible to forecast improvements. When DES was announced, the best attacks could break four rounds. There are now practical attacks on eight rounds, and two different academic attacks on the full sixteen rounds. One of the academic sixteen round attacks has been demonstrated. Progress continues; recently the amount of material required for Matsui's linear attack was reduced by a factor of four. The most important discovery about DES was a structural weakness of Feistel ciphers -- half the rounds could be ignored. Three of the five AES candidates use modified Feistel structures.

The AES is expected to last twenty years, and may be with us for a century. If it's good enough, it may well be permanent. (In spite of known attacks, movement away from DES has been glacial. Even the existence of a key-search machine has not discouraged some folks for proposing DES in new network protocols!)

A successful AES will be a very tempting target, attracting the attention of the best and brightest cryptographers.

Appropriate Security Margins

Briefly review the drawbacks of a busted AES cipher:
(a) A new cipher must be adopted in haste.
(b) Old software that can't be adapted to the new cipher must be abandoned.
(c) Old hardware is toast.
(d) Old data that was thought to be safe is at risk.
(e) People will continue to use the old cipher. Some of their traffic will be broken.
(f) Bad guys may have been secretly reading traffic for years.
(g) New software must use the new cipher, but maintain compatibility to read old encrypted records.
(h) New hardware is in the same boat.

(i) The actual transition time for new software and hardware is several years. Some hardware, such as satellite electronics, has a much longer lifetime.

Now examine the costs of selecting a cipher with extra security margin:
(j) It runs slower in hardware and software, or needs extra area in hardware.
(k) The maximum bandwidth achievable with given amount of logic is less.
(l) A few applications become infeasible or uneconomic because the cost of the encryption becomes too large, or the cipher won't fit on the chip, or crowds out some other functionality.
(m) Some applications that would have used encryption will omit it because it's too slow.

As part of deciding how much security margin is appropriate, we must balance factors a-i against factors j-m. Since the cost of recovering from an insecure cipher is pretty high, it's prudent to add some security margin beyond that required for "can't be broken in the near term".

I agree with Lars Knudsen's suggestion: An AES cipher should use twice as many rounds as the most effective attack that's better than exhaustive key search. This protects against the most likely catastrophe, discovery of a structural weakness such as happened to DES. It also protects against incremental improvements in the attacks that we see every year. It offers some protection against quantum computing, although this is probably a fool's errand. (Appendix C has my guesses about the future of computing.)

This is a conservative criterion -- if we were not trying to plan for an uncertain long term future, we could accept more risk. If we were designing a single application for a short lifetime, we could accept more risk. But the AES will be the world's major cipher for quite a while, and we can't accept more risk.

By this criterion, RC6 and Rijndael have insufficient safety margin while Serpent is overbuilt.

Other Security Issues

Ross Anderson argues that we should routinely use 256-bit keys, and skimp with 192 or 128 only when absolutely necessary. I'll disagree somewhat, and say that we should use 256 bits when it doesn't increase cost much. The current situation is that we don't have infinite hardware, so an application that stores a large number of keys will prefer to use keys that are as short as possible. Some of the ciphers are slower with longer keys.

Most seriously, our key exchange algorithms take unpleasantly long when matched with a 256-bit symmetric key. A Diffie-Hellman key exchange

matched to 128-bit security will use a prime modulus of 3000 bits; matching 256-bit security requires a modulus of 15000 bits, and the key exchange will take twenty-five times as long to compute as the 128-bit case. The situation is somewhat better if elliptic curves are used, since the curves need only use 250-bit or 500-bit fields; but 128-bit matched security will run six times as fast as the 256-bit case.

This suggests an additional requirement for the AES ciphers: Truncated key spaces should still have exhaustive-search complexity. Legacy systems will be using keys of 40, 56, 64, 80, 112, and 168 bits: A standard way of converting such keys to 128 bits (or more) is needed. If I extend an 80-bit key to 128 bits by padding with 0s, and the attacker knows this, breaking the cipher should still require roughly 2^80 effort. Ditto if I extend the key by copying 48 bits, or by xoring two overlapping copies. This requirement is related to the notion of weak keys, but has a different complexion. Weak keys are a lesser concern if they are unlikely to occur in practice; the "truncated key" problem is to show that particular sets of keys which are likely to occur in practice aren't weak.

Except for RC6 and perhaps Mars, all the ciphers have the property that recovering the expanded key will translate into recovering the primary key. More seriously, the key schedules of Rijndael, and to some extent Serpent, allow an attacker who recovers (or guesses) some of the expanded key to compute additional bits of the expanded key. Recall that both differential and linear attacks on DES benefited from replicated subkey bits -- as soon as an attack finds a few subkey bits, the game is over.

Most ciphers execute the same length of time regardless of key length. Rijndael and Twofish vary their encryption effort with the key size, Rijndael by adding rounds, Twofish by complicating the sboxes. The other ciphers might benefit by adjusting the number of rounds to match the security level implied by the key length.

Some effort has been devoted to seeing which ciphers best resist power measurement attacks. Attempting to protect cheap smart cards from their possessors is a losing battle. Fancy test equipment always wins, so any secret worth stealing will be stolen. Protecting the smart card with the holder's cooperation is more reasonable, and benefits from prevention of timing attacks, power attacks, RF leakage, etc. However, some smart cards will be stolen, and their secrets extracted. This should be recognized at the system design stage.

I give no weight whatsoever to the difficulty of implementing a block cipher in software: The criterion is silly. Any serious implementer will test his code against the NIST-required test values, which are sufficient to guarantee good faith code against bugs. Most folks will simply download a certified implementation, or buy software containing

a tested implementation.

An additional argument was made at the AES3 conference, that we don't want to see headlines such as "AES broken" whenever someone discovers how to lower the number of chosen ciphertexts to break Rijndael-256 from 2^254 to 2^253. I don't put a lot of stock in this argument: People will get headlines however they can, and it doesn't much matter what we do.

Security of the AES Candidates

Mars

Mars has adequate margin against the attacks presented so far.
The idea of using different kinds of rounds is a good one.
The symmetrically reduced twelve round attack from AES3, with three of each type of round, is less than half of the actual eight cipher rounds of each type.
The full wrapper plus five core round attack (also from AES3) is only one third of the keyed cryptographic core, so again the margin seems adequate.
The eleven round attack [2] on the Mars core is more than half of the actual sixteen round core, but it isn't a threat because it omits a feature (the sixteen round unkeyed wrapper) specifically included to vitiate the attack.
The Mars key setup is subject to a timing attack, to determine the number of long-bit-block fixups that occurred. This information doesn't seem to help an attacker: even knowing some of the extended key bits doesn't help to learn others, and the extended key is much larger than the primary key.

RC6

There is a fourteen round attack [2] on RC6. There is also a fifteen round attack on RC6-256. Since the actual RC6 has only twenty rounds, the margin is insufficient.

Rijndael

Rijndael has inadequate security margin.
Rijndael-128 (with ten rounds) had a six round break included with its submission, and the FSE2000 and AES3 conferences showed a break of seven round Rijndael-192 (which uses twelve rounds).
AES3 also included a break of seven round Rijndael-128 that is "marginally faster than exhaustive search".
Appendix D addresses some of my concerns with the Rijndael sbox, and suggests a way to defeat the Square attack, which might allow ten round Rijndael to be secure.

Papers at AES3 and FSE2000 pointed out minor weaknesses in the key schedule.
Rijndael varies the number of rounds depending on key length, so a simple timing measurement will determine the key length. Although important, this information is usually available to an attacker from other sources.

Serpent

Serpent (32 rounds) is overengineered; the best attack in only eight rounds, although the Serpent team conjectured the existence of longer differentials. The overengineering comes at a cost: Serpent is typically twice as slow as the other ciphers in software.

Twofish

The best attack against Twofish only gets six rounds, versus sixteen in the actual cipher.
More time is needed to see if Knudsen's unsuccessful attack on nine round Twofish can be salvaged.
Some implementations of Twofish have execution time that depends on key length, so a simple timing measurement will determine the key length. Although important, this information is usually available to an attacker from other sources.

Security Summary

RC6 and Rijndael-(128,192) have insufficient security margin.
Mars and Twofish have adequate margin.
Serpent has too much margin, and pays a speed penalty for it.

3. Hardware Considerations

Of the AES candidates, hardware folks prefer Rijndael and Serpent.

AES is a big step backward for hardware encryption. Measured by throughput/area, no candidate has a clear advantage over 3DES, and several are worse. It's not impossible that high-speed hardware will go its own way if any of the current candidates is adopted without modification.

DES was designed for hardware implementation: The original standard explicitly forbade software implementations. DES uses primitives that are hardware friendly, such as random-looking permutations, shift registers with random-looking taps, tiny fixed sboxes, and nary a carry

in sight. The key schedule is simple, based on a stuttering shift register, which rotates back to the original key after each encryption or decryption, and shifts one way for encryption and the other way for decryption. The cipher hardware is nearly the same for both directions.

Perhaps in reaction, the present AES ciphers are software oriented, and somewhat hardware unfriendly. The ciphers have larger storage requirements, and rampant additions and multiplications. Hardware folks barf at multiplication, and would prefer not to have the long carry chains in 32-bit addition. There is no use in the present candidates of random-looking bit permutations; this is a powerful primitive in DES, but software doesn't support it. Key schedules are more software oriented, with some of the encryption effort being in the key setup, rather than simply moving the bits around as DES does. None of the ciphers has a cycle-back key schedule to ease decryption.

Reading through the AES3 hardware implementation papers, I compiled an implied Hardware Wish/Unwish List:

Key schedule computable from either end, a round at a time, in parallel with encryption/decryption.
Constant time key setup.
No variable rotations.
No additions.
No multiplications.
Small sboxes.
Random looking bit permutations are good.
Encryption and decryption should be virtually identical.
CBC mode must go!

Sandia's high-speed DES board uses a pipeline stage for each round, and can have 16 different encryptions/decryptions in progress with 16 different keys, and freedom to select encryption or decryption for each datum. This is the highest-throughput implementation strategy I know of.

Translating the technology to an AES cipher will run into significant problems: Of course, the circuit complexity (area or gate-count) must increase because of the longer blocksize and keysize. But that's recovered in extra throughput. However: The decryption key schedule is a problem for every cipher except Twofish. The sbox area is much larger for all the ciphers except RC6, which needs a squarer. The round functions are more complicated.

Decryption Key Schedules

Only Twofish has a key schedule that's easy to compute in both directions. The Serpent and Rijndael key schedules can be run backward, if the final state is first computed. This seems to require computing the full key schedule in the forward direction, or equivalent work. Mars and RC6 are worse: the full key schedule must be computed and

stored, and most of the work must be completed before the first round key is ready to use.

Two simple key schedules that are easy to compute in both directions:
(a) Step the key forward with some invertible transformation. Apply a simple map such as Serpent does to generate each round subkey from the stepped key. Arrange for the key to return to the starting state after R (the number of rounds) steps. DES did this with a stuttering shift register, but there are also linear transforms available for any period. If the number of rounds is variable, then some steps are stuttered. Including a stutter in the key schedule is a good idea anyway.
(b) Another way is to use a "tent" pattern: The key is stepped forward for half the rounds, and then stepped backward to return to start. The first half and second half subkey-generation maps must differ so that the subkeys are different in the second half. Some care is required to avoid the DES weak key problem with 0 keys.

One interesting option for using a cipher with unequal encryption and decryption costs is role reversal: Use decryption as the encipherment and encryption as the decipherment. Unfortunately, this cannot be recommended without further analysis. Most of the analysis done so far would carry over to the inverse cipher, but it's unwise to use a cipher in an unintended mode: Something may have been overlooked that would be apparent if the mode had been examined closely from the beginning.

CBC MUST GO!

CBC mode encryption is a problem for any parallel or pipelined hardware system. The fixup is to use lagged CBC, with a lag greater than or equal to the number of pipeline stages. This requires an unreasonable amount of IV, and forces even non-pipelined hardware to remember the intermediate IVs. There are proposals for two or three pipeline stages per cipher round. That's a lot of lag.

Many folks suggested Counter Mode as an alternative. I propose LFSR-Counter Mode. It's as easy to implement as Counter Mode, and allows arbitrary lookahead and so arbitrary parallelism or pipeline depth. The details are in Appendix C. The ATM Forum has a Counter Mode standard. Other modes are also possible; almost anything is better than CBC.

Multiplication in Block Ciphers Considered Harmful

I was very surprised that Mars and RC6 were advanced to the final AES round, since they require a 32-bit multiplier (32x32->low32) and a 30-bit squarer (30^2->low31). This is too much of a hardship for a high-speed hardware implementation, and also a burden on low end smart cards. It also guarantees that when new processor architectures appear, they will be slow on these ciphers, since integer multiplication speed

is never as good as simpler instructions. (An architecture is mature (senescent?) when multiply is nearly as fast as add.)

Multiplication circuits are (1) slow (2) area hogs (3) power hungry (4) hard to test and (5) full of possible patent potholes. IBM-Japan had a good paper at AES3 about minimizing carry-chain propagation delay in the Mars multiplications. There was no mention of patentable circuit optimization techniques in their paper. IBM is a reputable company, and I'm sure they are planning to donate their best Mars circuits to the public domain. But anyone else could have filed a submarine patent we won't hear about for years.

On the bright side, if Mars or RC6 is selected, we will probably see a flowering of new circuit design talent, since there's so much scope for clever optimization. Perhaps our goal of the fastest and best cipher is less important than encouraging this design talent.

Other Hardware Issues

Although Twofish has the most decryption-friendly key schedule, it compensates for this in other ways. Twofish uses key-dependent sboxes. This gives the hardware designer a Hobson's Choice for pipelined encryption with differing keys: Either use substantial real estate to precompute the sboxes and pass them along at each stage of the pipeline, or apply three-to-five levels of sbox at each round.

Having an adjustable number of rounds will cause trouble for pipelined systems. If a cipher with less than the recommended number of rounds is adopted, such as Rijndael-128 with 10 rounds, hardware designers may want to provide for an increase by including a few extra pipeline stages, and a bypass for current operation. Another possibility is to provide contingency circuits for double and triple encryption.

One objection to having a backup cipher in the Standard is that hardware implementations will waste area for the backup cipher. One answer to the "idle hardware" objection is to use both ciphers, with some traffic in each cipher. This changes the backup cipher from a hot spare into extra capacity.

4. Other Issues

The AES Process

There is some sentiment for negotiating changes privately with the developers -- for example, Rijndael has been suggested with 14-18 rounds. This is a bad idea for several reasons:

(a) From the public process viewpoint, it stinks like a three day old fish, no matter how innocent the purpose. One important goal of

the public development process is to avoid the controversy that surrounded DES; this isn't the time to abandon openness to speed up the selection process.

(b) It means the actual selected standard will not have been publicly reviewed. The speed estimates for a revised cipher are simply estimates, as are the area estimates for pipelined chips -- the highest speed chips. In most cases NIST is not particularly qualified to do the estimating: The proper folks to ask are the ones who did the original work.

(c) There's been only desultory discussion of questions such as "How many rounds of Rijndael or RC6 would be secure?" and people haven't had a proper debate with the issues lined up to do battle.

(d) Finally, I suggest revised key schedules for the ciphers to help out the hardware folks, and these must be reviewed if adopted.

There won't be a chance to do it over; let's get it right the first time.

I contend that sending the ciphers back for rework will give us a better crop to choose from, resulting in a better, more widely adopted, standard.

The argument against delay can be summed up as "It's late and getting later; we should have done this five years ago". It comes down to standing still while the meter is running: we'd rather get started designing our hardware and software. Fortunately, encryption is modular; most software can be created with dummy encryption, and the actual AES cipher inserted near the end of the creation phase. (Testing must be done with the actual cipher in place.)

Users of DES-based systems are concerned, since processing power and clock speeds continue to increase, and DES becomes more vulnerable each month. However, DES is not yet breakable without serious resources, either special chips or a thousand PC-years. This will remain true for a couple of more years. Triple DES is not threatened, and is merely slow; there's no urgency to switch, and another six months delay won't hurt.

There are two other timetable related considerations:

(a) The time for analysis has been insufficient. There aren't enough public cryptographers to give full scrutiny to all the ciphers. If NIST feels they must choose a cipher now, then they should designate a provisional candidate(s), to focus further evaluation and criticism. If nothing damaging emerges in a year, the provisional standard can be adopted.

(b) This also reduces the threat of an "IP-attack" with a submarine patent, since such patents can only remain submerged for a limited length of time.

Miscellaneous Comments

CBC mode must be phased out, and replaced with counter mode or some alternative.

NIST (or the cipher developers) should recommend standard method(s) of key extension. There will be folks using 64-bit and other short keys. The two obvious fast methods of key extension to 128 bits, padding with 0s and doubling the key, need significant security analysis for some of the ciphers. Extending a key by "hashing up" (feeding it through SHA), is secure but slows down encryption and requires extra hardware. Longer keys (192 and 256 bits) require two calls to SHA.

There's been no testing of other block sizes; NIST has no basis to recommend any blocksize other than 128 bits or any keysize other than the standard three (128, 192, and 256). This means that the various ciphers' advertised flexibility for other block and key sizes is merely show: It's not safe to use the flexibility without more adverse attention being directed at the variations.

Most ciphers execute for the same length of time regardless of key length. This may indicate an opportunity for optimization.

If encryption and decryption aren't equal time, decryption should be chosen to be faster. This will optimize the situation where an encrypted message is broadcast and decrypted separately by multiple receivers.

It's been pointed out that planning for a cipher transition is cheaper than doing it unplanned; having a backup handy is good sense. Naturally, in the case of cryptographic surprise, the cipher will be abandoned with all deliberate speed, regardless of any official NIST pronouncement.

The benefit of "only one standard" is that the hardware and software can assume that the encryption method is known, with no header bits or protocol negotiations required. But this benefit becomes a liability when the inevitable upgrade to the new system happens, and the application is stuck in the past and cannot even in principle handle the new traffic.

Mars should receive a "certificate of apparent security" so that those who wish to use it have a formal document to cite. It should not be in the standard, becasue it contains a multiplication in its inner loop.

5. Conclusions & Recommendations (You Asked!)

None of the AES candidates is fully satisfactory.
Most can be improved significantly with modest changes.

NIST should return all five candidates for curing of deficiencies, with a two month deadline. (A list of deficiencies is in Appendix A.) Another conference should be held in six months to review the revised candidates. This will push the final decision date back to summer 2001, but we will have a better cipher. In particular, hardware will be faster, and possibly software as well.

In the interim, NIST could designate a front-runner to focus cryptographic attention.

No revised cipher should be considered without several months for public analysis and discussion of the revisions.

NIST should explicitly not endorse any blocksize other than 128 or keysize other than 128, 192, and 256, regardless of a cipher's advertised flexibility.

NIST (or perhaps the cipher developers) should designate a standard method of padding or extending short keys (such as 40, 56, 64, 80, or 112 bits) to 128 bits for use with the selected AES ciphers.

Applications are encouraged to include capability for changing algorithms, or at least, switching to the backup cipher. The closer the backup comes to being a hot spare, or extra capacity, the better.

CBC mode should be replaced with LFSR-counter mode, described in Appendix C, or some alternative mode.

If this process is ever repeated, more consideration should be given to hardware from the start.

If we are doing this again in twenty years, add the criterion that the inverse cipher, using decryption as the cipher, also be secure.

Since there hasn't been sufficient time for adequate cryptographic review, any standard should be considered provisional for a year, and subject to recall if significant adverse information develops.

My choice among the present candidates is Twofish as primary AES, with Serpent as optional but strongly encouraged backup. Mars should receive a "Certificate of Apparent Security" but not be part of the Standard.

Acknowledgments

I would like to thank Tom Cabe, Ed Witzke, Anna Johnston, Susan Landau, Lyndon Pierson, and Craig Wilcox for helpful discussions.

References

[1] Proceedings of the 1st, 2nd, and 3rd AES Conferences.
http://www.nist.gov/~aes
AES CDroms 1, 2, and 3
[2] Preproceedings of FSE2000.

Appendix A: Defect List

Mars
Mars should replace the inner loop multiplication operation with some alternative.
The key schedule should be computable on the fly from either direction.
The variable key setup time, to check for and fix long blocks of 0s and 1s in the round subkeys, is a minor blemish, but it slows down a pipeline implementation.

RC6
RC6 has insufficient security margin. Since 14 rounds have been broken, 28 is the minimum acceptable.
RC6 should replace the inner loop squaring operations with some alternative.
The key schedule should be computable on the fly from either direction.

Rijndael
Rijndael-128 and Rijndael-192 have too few rounds. Since 7 rounds of each has been broken, the minimum should be raised to 14 rounds.
(This might cause Rijndael to lose its speed advantage.)
Change the Shift-Row operation to break up the bytes, to counter the Square attack.
The sbox should be replaced with a non-algebraic sbox.
Encryption is faster than decryption: they should be exchanged.
Papers from FSE2000 and AES3 noted minor weaknesses in the key schedule.
The key schedule should be computable on the fly from either direction.
(The encryption direction is ok now. The decryption direction can be computed backward, if first computed forward, but this requires substantial additional time in hardware, or a software assist.)
See Appendix D for discussion.

Serpent
Serpent has too many rounds, and is consequently slower than it needs to be.
The key schedule should be computable on the fly from both directions.
(The encryption direction is ok, but the decryption direction requires a special setup circuit, or software help.)

Twofish
There's nothing wrong with Twofish, but small hardware implementations will be slow. The key-dependent sboxes make fast pipelined hardware

expensive. Since the other ciphers are being returned for rework, the Twofish team may as well have a chance to improve Twofish.
If Knudsen's nine-round attack fails, the number of rounds in Twofish could be reduced.

Appendix B: The Future of Computing Power -- Speculations

One conspicuous absense in the AES discussions has been the phrase "threat model". The assumption is that a good cipher can trump all threat models with the magic words "128-bit key". During the coffee break, we backtrack to discount academic attacks, insinuating that an attack that requires 2^100 P/C pairs can be safely ignored.

One of the benefits of a powerful cipher is that it can be used and forgotten about: Whatever security problems may occur in our widget or protocol, the cipher won't be the weak point. On the other hand, it never hurts to look around a bit.

How powerful might an adversary be in the near future?

Current limits on world computer power are 2^70-75 encryptions/year and storage around 2^60-70 bits of memory. (Some attacks require real-time available memory, but others can be carried out with massive tape libraries.) For available enciphered material, it's possible that someone has encrypted as much as 2^55 bits (2^48 blocks) of data under one key. This would be a 1GHz channel for a year, or a million backup tapes.

It's reasonable to project another 20-30 bits of computing power from "Moore's Law" and further market penetration of computers. Bandwidth will grow by another 10-30 bits. Presumably demand for processing power and bandwidth will saturate when each of us has a real-time synthetic video channel.

In the far future?

Restricting ourselves to conventional physics, assume every atom in the top kilometer of Earth's crust is a gate, and switching occurs at the speed of light traveling the length of a chemical bond. We are limited to about 2^156 gates (or memory) and 2^61 gate transitions/second. Assume about 15000 gate transitions to encrypt. This translates to about 2^228 encryptions/year. Any attack using more than these resources will remain academic until we can build brains the size of a small planet.

It's too early to tell if Quantum Computing will develop into anything

real. There's a QC algorithm for searching N^2 things in time N; if this can be applied to exhaustive key searching, it would mean that key lengths should be doubled to resist quantum attacks.

There's no proof as yet that NP-complete problems require exponential times with conventional computers, although the betting is that they do.

It's possible that QC will permit "full parallelism", allowing searches of 2^N things with N qubits. This would make NP-complete problems practically solvable, and destroy conventional cryptography. No such general QC algorithm is known. There are QC algorithms known with exponential speedup for solving factoring and discrete logs, but these don't seem to be NP-complete problems.

Appendix C: Replacing Cipher Block Chaining (CBC) Mode

CBC MUST GO!

CBC mode encryption is a problem for any parallel or pipelined hardware system. The fixup is to use lagged CBC, with a lag greater than or equal to the number of pipeline stages. This requires an unreasonable amount of IV, and forces even non-pipelined hardware to remember the lagged IVs.

Counter Mode has been suggested as an alternative. I propose LFSRC (Linear Feedback Shift Register Counter) Mode. It's as easy to implement as Counter Mode, but allows arbitrary lookahead and arbitrary parallelism or pipeline depth.

LFSRC copes with the two main problems of Counter Mode: (1) It's a little too likely that two adjacent plaintext blocks have consecutive values that become equal when the counter is xored with the plaintext, and hence encrypt to the same ciphertext; and (2) CM encrypts a long plaintext block of 0s as the encryption of a sequence of consecutive integers, a worrisome practice if our cipher has an unknown weakness.

In LSFRC Mode, a 128-bit linear feedback shift register is initialized from an IV. The register is xored into the plaintext before each encryption, and is stepped after each block. The step function is to shift the register one bit to the left; if the bit that fell off the end was a 1, we xor 0x87 into the low byte. This corresponds to the polynomial u^128 + u^7 + u^2 + u + 1, the first (mod 2) irreducible polynomial of degree 128. It has maximal period, 2^128-1.

The register can be initialized from a full 128 bit IV, or by a shorter IV of length 32 or 64 bits. The shorter IVs are replicated to make up the full register length. An IV of 0 means ECB Mode.

Appendix D: Problems with Rijndael

(Notation: In this appendix, + is used to mean addition in the GF[256] field used by Rijndael. This is done by xoring the bits representing the field elements. For single bits, this is just xoring the bits. This frees up the symbol ^ to represent powers. I use the letter u to represent the field basis generator. The Rijndael field polynomial is `u^8 + u^4 + u^3 + u + 1`.)

Rijndael has several properties that I find worrisome. These could be easily fixed, with minimal impact on code execution time and circuit size.

Rijndael is a Byte Cipher

Rijndael is really a cipher on bytes, rather than bits. All of the manipulations treat the bytes as whole objects, and Rijndael does nothing to break up the byte structure. This weakness is what allows the Square attack to work. Fortunately, it's easy to fix, by changing the Shift-Row operation to rotate the rows a non-integral number of bytes.

The Fearful Symmetry of the byte movements bothers me. If the Round Constants used in the key expansion were missing, then encryption would be compatible with rotating each word of the plaintext and key by a byte. The Round Constants for the first eight rounds are single bits.

Rijndael is Mostly Linear

A second Rijndael design decision is "not to mix groups". The cipher contains no ordinary arithmetic additions. The only non-linear function in Rijndael is the sbox; the rest of the cipher is pure xors.

One amusing consequence is that the cipher has a number of "alternative implementations" available. Any linear map can be applied to the bits within a byte, with the same map applied to every byte, and other parts of the cipher adjusted, to get the same cipher.

Example: Suppose we map bytes by xoring the low order bit of each byte into the adjacent bit.

```
B1 += B0
```

(Recall that I'm using + to mean xor, so that ^ is free for exponents.) The necessary compensating internal changes are: Plaintext and ciphertext are adjusted to match, as a wrapper around the revised cipher. The input of the sbox is adjusted by conceptually xoring the

address lines: `A1 += A0`. This is equivalent to exchanging two quarters of the sbox table. The output of the sbox is adjusted by xoring S0 into S1.

The Mix-Column operation is modified slightly: Two of the mixed copies of a byte are multiplied by u and u+1, which requires changing two of the three low order bits of the "multiplication by u" circuit. The circuit is still linear.

```
C7...C0  =  u * B7...B0
```

is originally implemented as

```
C7 = B6, C6 = B5, C5 = B4, C4 = B3+B7, C3 = B2+B7, C2 = B1,
C1 = B0+B7, C0 = B7.
```

This is changed to

```
C7 = B6, C6 = B5, C5 = B4, C4 = B3+B7, C3 = B2+B7, C2 = B1+B0,
C1 = B0, C0 = B7.
```

Only `C1` and `C2` are changed.

Multiplication by `u+1` is done by xoring B with C.

Finally, the expanded key must be adjusted to match, by xoring in every byte's low order bit into the adjacent bit. It looks like this can be done by adjusting the actual key, and using the modified sbox. The round constants used for the key schedule stutters are also adjusted to match.

Any nonsingular linear map can be built from similar single-bit-xor primitives. I'm not actually proposing this as an implementation optimization, but it provides an attacker with some interesting working material: He can do a series of manipulations to the sbox, looking for more favorable cryptographic properties such as single-bit-change bytes, or bit-swap bytes. He can even change to one of the other 29 eighth-degree irreducible finite-field defining polynomials, or switch the basis of the finite field from polynomial to normal.

None of the other candidate ciphers has this linearity property. The property is a deliberate design decision. It would be easy to remove, say by combining the round keys and middletext using plain 32-bit addition instead of xoring.

Rijndael's Sbox is Based on Finite Field Reciprocals

Rijndael's algebraic sbox is a terrible idea. The sbox is defined as the finite field reciprocal, followed by an affine linear transformation.

[Math majors, please forgive the recapitulation of finite fields.]
Finite fields have automorphisms, they often have non-trivial subfields,
and they have maps to other finite fields of the same degree. The
automorphism x -> x^2 exists in any finite field of characteristic 2.
In GF[256] the map is of order 8; x -> x^2 -> x^4 -> ... -> x^256 = x.
If x is any finite field element, then the sum of all its morphic
images is simple:

```
Trace(x) = x + x^2 + x^4 + x^8 + x^16 + x^32 + x^64 + x^128 = 0 or 1.
```

The automorphism commutes with polynomials and reciprocals, the inner
function in the Rijndael sbox. So if a,b,c satisfy some equation such
as a^3 = b^2 + b/c, and the automorphism maps a->A,b->B,c->C, then
A,B,C will satisfy the same equation: A^3 = B^2 + B/C. (Constants in
the field must also be mapped, but 0 and 1 always map to themselves.)
The Rijndael reciprocal operator is just x->x^254, and is compatible
with the automorphism.

The affine transform following the inner map is NOT compatible with the
morphism, but it is linear in the field basis, so there's some other
matrix that's equivalent to the morphism applied to the sbox.

We might try the approach used in the Square attack: Apply the morphism
eight times to a plaintext byte, and examine the results. The morphs
of x are {x,x^2,x^4,...,x^128}.

Applying the first half of the sbox, the reciprocal operator, and
summing the results gives

```
1/x + 1/x^2 + 1/x^4 + 1/x^8 + 1/x^16 + 1/x^32 + 1/x^64 + 1/x^128

 = 0 or 1.
```

The second part of the sbox, the affine transform will prevent these
values from continuing to be a set of automorphic images, but will
preserve the property that the sum is one of two values, 0 or 63.
[The value x=0 may cause some of the equations with reciprocals to be
invalid, but many encryptions will never use that sbox entry. A ten
round encryption with a sixteen byte data block will touch the sbox 160
times, with 48% probability of touching the 0 entry.]

GF[256] has unique subfields of order 16, 4, and 2 that are fixed by
the morphism. Elements of GF[2] are fixed by the morphism; elements
of GF[4] are fixed by any even power of the morphism; and elements of
GF[16] are fixed by the fourth power of the morphism.

For any x in the field,

```
  x + x^4 + x^16 + x^64  is in the GF[4] subfield, and
```

```
  =   either 0, 1, u^7+u^5+u^4+u^3+u^2, or u^7+u^5+u^4+u^3+u^2+1.
```

The same is of course true for `1/x + 1/x^4 + 1/x^16 + 1/x^64`; after the affine map is applied, there will be only four possible values.

Similarly, `x+x^16` has one of the values in GF[16]. This subfield is (additively) generated by `1, u^3+u^2, u^6+u^4, and u^7+u^6+u^5`.

The reciprocal of a subfield member is of course in the subfield. Complementing the low-order bit of a byte keeps it in the same subfield, so its reciprocal remains in the subfield.

Rijndael's Sbox has a Relatively Simple Algebraic Formula

The Rijndael documentation notes that the full sbox map is equivalent to the algebraic expresion

```
S(x) =  63 + 8f x^127 + b5 x^191 + 01 x^223 + f4 x^239 +
        25 x^247 + f9 x^251 + 09 x^253 + 05 x^254
```

(Coefficients are hex representations of GF[256] field elements. Calculations are done in the field, so "+" is the xor of the elements.) A typical random sbox would have 250 terms instead of 9.

Although S(x) looks daunting, it isn't quite as bad as it seems: If x is not 0, then `x^255 = 1`, so we can rewrite the right-hand-side as

```
               8f      b5     01     f4    25    f9    09    05
S(x) =  63 + ----- + ---- + ---- + ---- + --- + --- + --- + --
             x^128   x^64   x^32   x^16   x^8   x^4   x^2   x
```

Then the individual denominators are linear in x, allowing relatively simple rational expressions for S(a+b):

```
                    8f             b5                05
S(a+b) =  63 + ------------- + ----------- + ... + -----
               a^128 + b^128   a^64 + b^64         a + b
```

and

```
                           8f                       05
S(a+b+c) =  63 + --------------------- + ... + ---------
                 a^128 + b^128 + c^128         a + b + c
```

and

```
               8f    b5      01          05
S(a^2) =  63 + -- + ----- + ---- + ... + ---
               a    a^128   a^64         a^2
```

and

$$S(a+a^{16}) = 63 + \frac{8f + 25}{a^{128} + a^8} + \frac{b5 + f9}{a^{64} + a^4} + \frac{01 + 09}{a^{32} + a^2} + \frac{f4 + 05}{a^{16} + a}$$

The iteration of several rounds of the cipher will create more complex expressions, but random sboxes would have much more complex expressions.

There is a natural tendency to regard these expressions as much too complex to solve, but we have seen no detailed analysis of the growth of expression size as a function of the number of rounds. If breaking ten round Rijndael-128 merely requires writing the meet-in-the-middle equation

```
R5(plaintext,ExpandedKeyBeg) = R5inv(ciphertext,ExpandedKeyEnd)
```

and substituting in known plaintext-ciphertext pairs reduces the two sides to polynomials with only a million terms, we have trouble. An attacker can collect two million P/C pairs and solve the equations as a large system of LINEAR equations.

There may also be an advantage in using a normal basis representation for the field elements. A normal basis representation uses a field element v and its successive squares `v^2, v^4, v^8, v^16, v^32, v^64,` and `v^128` as the basis instead of `1, u, u^2, u^3, u^4, u^5, u^6, u^7`. We require `Trace(v) = 1`; for the Rijndael polynomial, this means v must have either `u^5 xor u^7` as a component. A good choice for v is `v = u^5`; then the basis is `u^5, u^10, u^20, u^40, u^80, u^160, u^320`, and `u^640`. There's a linear transformation on the bits to convert between the field representations, so the "alternative implementation" offered above works. The advantage of NB representation is that squaring is a rotation of the bits. Suppose that [hgfedcba] is a field element, with each letter representing one bit:

```
[hgfedcba]^2 = (h u^640 + g u^320 + ... + b u^10 + a u^5)^2

             = h u^5 + g u^640 + f u^320 + ... + b u^20 + a u^10

             = [gfedcbah]
```

Reciprocals of rotated values are rotated by the same amount in the same direction, so in the equation for S(x) as a sum of fractions, the denominators will all be rotations of x, and all the reciprocals will be rotations of 1/x.

It's possible that all these sbox speculations will lead to nothing, and that no attack on Rijndael can be built from these ideas. However,

Rijndael is the only AES finalist that's subject to algebraic attack, and the risk can be completely removed by using a new sbox.

There's no extra cost in either hardware or software, since no implementation calculates the sbox fom the definition; all use table lookup. There's even a potential benefit: the substituted sbox can be an involution, which will simplify decryption hardware and make decryption software smaller.

There's no need for the Rijndael sbox to have the reciprocal property. As the authors themselves state, a random sbox with good crypto properies would suffice.

Recommendations for Rijndael

Add more rounds, unless other changes reduce the number of breakable rounds to five or fewer.

Change Shift-Row to break up the bytes and spoil the Square attack, by rotating the rows a non-integral number of bytes.

Replace the sbox with a non-algebraic random sbox having the usual properties of balance, diffusion, etc. This sbox can safely be an involution, which will simplify life for the hardware folks.

Change the key schedule to be easily computable from either end. [Papers at FSE2000 and AES3 noted weaknesses in the Rijndael key schedule which should be addressed.]

Exchange the roles of decryption and encryption, so that decryption is not slower than encryption.

Sender: jworley@hpfctwc.fc.hp.com
Date: Mon, 15 May 2000 15:12:56 -0600
From: John Worley <jworley@fc.hp.com>
Organization: Hewlett Packard Labs, Denver
X-Mailer: Mozilla 4.7 [en] (X11; U; HP-UX B.10.20 9000/785)
X-Accept-Language: en
To: AESround2@nist.gov
Cc: Tom Christian <twc@fc.hp.com>, Bill Worley <worley@hpl.hp.com>,
Ross Anderson <Ross.Anderson@cl.cam.ac.uk>,
Dag Arne Osvik <osvik@ii.uib.no>
Subject: New Serpent Timings for IA-64

By carrying a duplicate of the low 32 bits of the data block in the high 32-bits, it is possible to greatly improve Serpent performance on the IA-64. The advantage is that the fixed rotations can now be accomplished in one cycle. Equally important is the contribution of Dag Arne Osvik's equations tailored to the IA-64 architecture. These provide 4- and 5-cycle software solutions for the s-boxes.

This solution requires a doubled-size key table (1056 bytes), but the keying time decreased from 490 to 340 cycles, while the encryption time decreases from 565 cycles to 419. I think this should be considered more than acceptable software performance for almost any application.

Regards,
John Worley
jworley@fc.hp.com

From: "Mattias Lennartsson" <mattias.lennartsson@telia.com>
To: <AESround2@nist.gov>
Subject: AES
Date: Mon, 15 May 2000 23:24:09 +0200
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3110.5
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3
X-MIME-Autoconverted: from quoted-printable to 8bit by email.nist.gov id RAA08781

I think that the best selection for AES is the safest algorithm, so far I think that it's many that agrees. But witch is than the safest one. As I see it there are two big threats to new algorithms. Either a improvement of the cryptoanalytic methods used today or the discovery of a totoal different way of attacking block chiffers. So the chiffer should be a chiffer that stands greate aganst todays differential attack and also be safe against new ones. First we need a big security margin on the differentials, this could be found on either Serpent or Twofish. And if you add more ronds to Rijndael witch probably should hav had more ronds allready from the begining, a good security margin is found also here.

But it should allso be safe against future attacks. I think that this is achived by a komplex algorithm structure. If you add several different and safe components it seemed resonible to me that it would stop a future attack better. If you use a small number of components and one or more of those component is found less effective it may crack the hole algorithm. But this is not hapening if the chiffer is more complex. So therefore i think that Twofish would be the best algorithm for AES.

And I don't like the idea of selecting two algorithms for AES. It would just be to confusing, I think that selecting one algorithm will do just fine. What is hapening if two AES standards i selected and one is broken. Wouldn't this affect the trust of AES, how do you know witch of the AES you're program is using. I think that we should concentrate on finding the most securest one instead of argumenting.

Mattias Lenartsson (Sweden)

Date: Mon, 15 May 2000 16:34:42 -0600
From: "Ralph S. Hoefelmeyer" <ralph.hoefelmeyer@wcom.com>
Subject: Twofish
To: AESround2@nist.gov
X-MIMEOLE: Produced By Microsoft MimeOLE V5.00.2314.1300
X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)
Importance: Normal
X-MSMail-priority: Normal

Based on my analysis of the options, I support Twofish for AES. It is flexible enough to support the myriad platforms we are going to see in the future, from Java-enable cell phones and PDAs, to smart cards and tokens. It has a reasonable level of assurance of confidentiality for the things one would use it for, such as financial transactions.

This is my opinion, not that of MCI WorldCom.

Ralph S. Hoefelmeyer
Senior Engineer
Team Lead, Next Generation Network Security Services

Date: Tue, 16 May 2000 02:31:16 +0200 (MET DST)
From: Dag Arne Osvik <osvik@ii.uib.no>
To: AESround2@nist.gov

I have some comments on two of the AES finalists

Serpent

- In "The AES second round Comments of the Rijndael", the Rijndael team claims that Serpent needs 80 bytes for encryption, even for 128 bit keys. Now consider the minimum RAM requirements of Serpent:

  - Text block, 16 bytes
  - Key setup state, 32 bytes (independent of key size)
  - Temporary storage, 1 byte (for s-boxes/rotations)
  - Loop counter, 1 byte

  The loop counter and temporary storage might be kept in registers where available. Total RAM requirement is thus 48-50 bytes. Note that this requires application of both forward and inverse s-boxes in the key schedule, increasing code size (inverse s-boxes are required).

- S-boxes can exploit parallelism in processors, as demonstrated by my S-boxes optimized for IA-64 (3 cycles critical path, 4-5 cycles total). John Worley just reported timings using these. Other processors should see similar figures for the s-boxes, depending on their number of ALUs.

Twofish

- There is a tradeoff between key agility and encryption speed, i.e. if an implementation should be optimal for all numbers of blocks, it will have to choose in run time which key setup/encryption combo to use. And of course, all chosen combos must be implemented.

- In "Comments on Twofish as an AES Candidate", the Twofish team claim that "Twofish far surpasses the other four finalists" in terms of security/performance ratio, "normalizing to the largest number of rounds cryptanalyzed". The NSA ASIC implementations indicated that Serpent was nearly 4 times faster than Twofish, and from the cryptanalysis I've seen so far, Serpent seems far more secure than Twofish. Also I expect S-box optimizations to benefit Serpent hardware implementations significantly.

Regarding my personal preferences for the AES, here is my list:

1. Serpent
2. Rijndael (with extra rounds)
3. Twofish
4. RC6
5. MARS

And choose only one.

Regards,

Dag Arne Osvik http://www.ii.uib.no/~osvik

X-Lotus-FromDomain: CERTICOM
From: "Don Johnson" <djohnson@certicom.com>
To: AESround2@nist.gov
Date: Tue, 16 May 2000 16:07:37 -0400
Subject: ANSI X9F1 vote on Future Resiliency

NIST,

I have just reviewed the AES round 2 notes/text comments (the big combined file).

Early in the file I mention that I presented "Future Resiliency" to ANSI X9F1 and that they "voted for it" in some sense. For some reason or another, Roger Schlafly, who was not there, deemed it necessary to claim that this vote was meaningless. This was simply not true, so let me clarify.

I presented the paper "Future Resiliency: A Possible New AES Criterion" to ANSI X9F1, which I had submitted to the first AES conference. The presentation took the better part of an hour, it was not a truncated summary presentation. After I was done, there was some discussion as to what to do, if anything, about the ideas in it. I suggested an official vote, as follows:
A) NIST should incorporate the ideas of "Future Resiliency" in the AES criteria.

This passed with a good majority, there were a few no votes.

Allen Roginsky, IBM, later proposed another official vote, as follows:
B) NIST should consider adding the ideas of "Future Resiliency" to the AES criteria.

This passed with more yes votes, being stated in a weaker form.

These were official ANSI X9F1 votes and were specifically based on the idea that multiple algorithms were a way to address as much as possible the potential for a disastrous crack in one algorithm.

I wanted to let NIST know for the record how the ANSI X9F1 vote should be interpreted.

Don Johnson