Email Comments on Block Cipher Modes of Operation

Updated comments will be posted at http://www.nist.gov/modes

Date: Mon, 27 Dec 1999 16:54:12 +0100
From: Eric Boesch <ebo@dannet.dk>
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),....

From: "Costantini, Frank @ CSE" <fcostant@mail.cse.l-3com.com>
Subject: AES modes of operation
Date: January 26, 2000 16:02

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

– – –

From: "Chip McGrogan" <chip.mcgrogan@L-3COM.com>
Subject: RE: AES modes of operation
Date: Wed, 26 Jan 2000 19:03:53 -0500

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: "Schroeppel, Richard" <rschroe@sandia.gov>
Subject: comment for AES cipher selection
Date: Mon, 15 May 2000 14:18:37 -0600

Excerpts extracted relative to modes (full text may be found in the AES Round 2 comments at http://www.nist.gov/aes):

[SNIP]

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

[SNIP]

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.

[SNIP]

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.

Suggestion from Paulo Barreto
Subject: AES Modes
Date: about 30, May, 2000

NIST will standardize encryption modes of operation for AES and even asked for comments on this subject. However, I have found no mention to *hashing function* modes. I wonder whether the AES standard will specify any hashing function construction (e.g. Davies-Meyer or Matyas-Meyer-Oseas) on top of the AES block cipher?

Date: Mon, 03 Jul 2000 09:26:45 -0600
From: John Myre <jmyre@sandia.gov>
Subject: Some thoughts on the requirements

To the NIST team,
Although NIST asked for input relating to encryption modes as part of the AES comment process, I did not see much on this subject in the published AES comments. I am glad to see NIST taking the initiative to address the subject separately. In this note I will mention a number of general ideas. Here is a quick list of the points raised below:

1. NIST should consider defining modes based on application requirements.
2. Disk (sector) encryption has requirements that are not met just by one of the standard modes.
3. Dropping the requirement for encrypting only in the forward direction may provide methods for enhanced security.
4. Padding methods and ciphertext stealing should be addressed together with encryption modes.
5. If possible, NIST should provide guidance on making choices within the standard.

I will not attempt to advocate any particular mode; this is the job of the cryptographers. I do want to point out, however, that the usual modes (ECB, CBC, OFB, CFB) are all designed for use by communication channels. Thus, all of the modes are used in the "forward" direction, not requiring access to past data. There are, however, many other applications, each with their own requirements. I would hope that NIST address the needs of these other applications.
For example, consider disk encryption. In this case, what we really want is a block cipher with a large block, the size of a disk sector (512 bytes or larger). Various methods have been designed for this; the good ones all seem to require something more complex than choosing one of the standard modes. The main requirements to meet are that no data expansion is allowed, and that data in a location (sector) may often be replaced with small changes.

Other examples of encryption of large data blocks are e-mail and file encryption. Just as for disk encryption, it is correct to assume that we have access to all of the data at once. Although CBC (with a unique, random IV for each encryption) is usually thought adequate for this, one can consider use of other modes to enhance security. The concept of an "all-or-nothing transform" is attractive, when the extra processing is worth it. In any case, NIST should consider defining modes that are appropriate for use in these situations.

Perhaps other users can name other application areas. Clearly NIST can not undertake the job of designing protocols for every application. However, I think that NIST can, and should, go beyond the simple modes already defined. It should be possible to categorize the basic application types, based on their needs for encryption modes.

In addition to the essential encryption mode, often a user must choose a padding method. I think that NIST should endeavor to address this area, as well. The padding mode can have security implications in some cases. Note also the concept of "ciphertext stealing", which is conceptually a "padding" method (that avoids padding) but is really a modification of the encryption mode.

There are cases where this technique is important, and it would be very helpful for NIST to define how to implement it correctly.

Of course there are other places one can go for information. ANSI X9, IEEE P1363, and RSADSI PKCS come to mind. NIST is certainly free to take advantage of work done there, as for example the association of FIPS 117 and X9.17.

Finally, the best service that NIST could provide would be to provide guidance on when each encryption mode is appropriate. Indeed, the same could be said about choosing an encryption algorithm. When, for example, would Skipjack be a better choice than AES? Perhaps, in the uncertain world of cryptography, NIST feels unqualified to document such considerations. Still, many (if not most) implementors would welcome advice of this type.

To summarize, I hope that NIST will consider the requirements of different application classes in defining encryption modes. Examples include disk encryption and file encryption. Also, padding methods are sometimes related to encryption modes, and need to be standardized, as well. Finally, the most difficult problem to solve, and therefore the area in which NIST can help the most, is choosing among alternatives.

Thank you for your time -
John Myre
Sandia National Labs
jmyre@sandia.gov
These comments are mine alone, and do not necessarily represent
the position of Sandia Labs or its associated entities.

Date: Thu, 03 Aug 2000 13:58:34 -0600
From: David Scott <dscott@elpasonet.net>
Organization: retired
Subject: Modes of Operation Workshop

Will the use of other than the normal standard modes of chaining, such as "wrapped PCBC" be considered. I would be willing to speak on the use of such secure chaining methods. And on the use of bijective mapping to get files to match the block sizes used in the encryption.
David A. Scott

Date: Mon, 21 Aug 2000 16:03:00 +0200
From: Mats Näslund (ERA) <mats.naslund@era-t.ericsson.se>
Subject: Enquiry

Dear Sirs
We have a new mode of encryption that we feel should be of interest for the AES work. Like OFB it provides a key-stream generator, but in addition, this new mode has provable security properties.

Can you please give some further information concerning how submissions to this workshop is handled (I trust there _will_ be submissions?), deadline etc. Will there be any review process of submissions? Finally, will it be required that one of the authors present the work at the workshop? (I would guess so, and I am sure we can arrange for that in such case).

Kind regards
/Mats

From: Gideon Yuval <gideony@microsoft.com>
Subject: If modes are to be as secure as AES itself,
Date: Fri, 29 Sep 2000 12:37:56 -0700

naive counter-mode is weak: after 2^64 cycles, the absence of birthday-paradox collisions indicates a clear difference between AES & real random.

Adding successive pairs of results, modulo 2^128 (NOT xor!!) will fix it -- remaining nonrandomness is a 2^-128 bias in favor of odd numbers, which needs 2^256 encryptions (!) to detect.

I'm told there are much cheaper fixes, though.

Date: Sun, 01 Oct 2000 08:09:11 -0600
From: David Scott <dscott@elpasonet.net>
Organization: retired
Subject: Modes of operation

I have noticed the old 3 letter modes do not provide for a degree of secutiry for individuals who wish to isolate the individual plaintext ciphertext block pairs from one using chosen plaintext type of attacks. Will there be any consideration for modes that allow a more wide spread diffusion through a file such as "wrapped PCBC".
Thank You
David A. Scott

Date: Mon, 02 Oct 2000 08:30:56 -0400
From: "Brian Gladman" <brg@gladman.plus.com>
Subject: Re: Modes of Operation
In-Reply-To: <000901c02acc$9f6168b0$592a9fd4@fortytwo>

It seems to me that the various modes of operation comments spread throughout the papers and public comments are not really in a form that is easy to deal with for the upcoming Modes of Operation workshop.

I started to go through all the papers with the aim of documenting each proposal but it is a big job just to find them all and this led me to wonder whether it would be better to ask those who have discussed modes of operation to provide short decriptions of their proposals prior to the workshop. Even asking for confirmation that they wish their proposal to be tabled would help sort out those that are real contenders.

Doing this would have the merit of allowing people who won't be there to comment and this in turn will help those at the workshop. Maybe this is all too late but I am slightly worried that the workshop may not succeed unless there is some preparatory activity and my guess is that you folks have had other things on your mind!

As you know, I am happy to help and that was my aim in trying to collect all the proposals together. Unfortunately, however, I just don't have the time to go through the massive volume of public comments to actually find the proposals that have been made. Once they are found its probably easy to document them - finding them is the problem!

And this was the reason for my recent question in the discussion forum. Let me know what you think (when you catch your breath after Monday!).

all the best,

Brian

From: "Phinney, Tom (AZ15)" <tom.phinney@honeywell.com>
Subject: Rijndael - Encryption vs decryption
Date: Wed, 4 Oct 2000 23:53:05 -0700

To: NIST AES post-selection comments
Subject: Rijndael suggestion re encryption vs. decryption

First, I would like to complement the AES selection team on an even-handed assessment of the strengths and weaknesses of the five candidates from the second selection round. The rankings given in the detailed report were clearly supported by the facts of record.

Second, I would like to applaud the selection of Rijndael. I submitted comments in the final comment round, presenting the needs of the U.S. Industrial Automation and Control community relative to the AES. My evaluation showed that Rijndael and Twofish best met those needs, which with respect to the selection process were primarily for an algorithm with limited computational requirements when used on 8-bit and 16-bit microprocessors.

Now NIST has the task of finalizing the FIPS for AES. I would like to suggest, as others did in the public comments, that the preferred roles of encryption and decryption be reversed from those of Rijndael as submitted. This is relevant because Rijndael is asymmetrical with respect to the energy and processing time required for encryption and decryption. The reasons for suggesting this generic role reversal are as follows:

1) In many common applications, each encrypted message will be decrypted many times. This is true in multicast communications where many receivers decode each transmission (e.g., satellite broadcasts or cell-phone reception by multiple base stations). It is also true in storage applications (e.g., encrypted disk partitions or file contents), where reading dominates writing. In this "green" era of energy conservation, and of battery-operated equipment, the more frequently occurring operation (decryption) should be the one with the lower energy requirement.

2) It is easier to anticipate the session key required for transmission than for reception. This occurs because the transmitter usually schedules transmissions, either from a queue or by other means (e.g., dedicated transmission). Receivers, on the other hand, frequently have no ability to anticipate what will be received (e.g., TCP/IP or Ethernet) and must react immediately. When anticipation is possible, the Rijndael-required development of all round subkeys before the commencement of what is now called "decryption" can be concurrent with a prior transmission. Thus the current "decryption" operation should be used for encryption prior to transmission, and the current "encryption" operation for decryption upon reception, since the latter does not require that intermediate-key precomputation.

3) When Rijndael is used for storage access, the time-critical operation is always reading, not writing. Thus reading and decrypting (e.g., of encrypted information on a disk) should be the faster operation. Again this is best provided by using the current "encryption" mode for decryption after reading, and the current "decryption" mode for encryption before writing. This change places the burden of full precomputation of round subkey to the writing mode in those cases where the round subkeys are not readily precomputed, such as when the keying nformation

is address dependent.

Respectfully submitted,
Tom Phinney
US Technical Advisor for IEC/SC 65C (which writes Industrial Automation and Process Control communications standards)
Convenor, IEC/SC 65C/WG 1 and /MT 9 (the Fieldbus standards)

– – –

From: Bill Burr <william.burr@nist.gov>
Date: Thursday, 05 October 2000 06:38
Subject: Re: Rijndael - Encryption vs decryption

Tom,
A couple of questions or observations:

1. Isn't the difference between encryption and decryption, that is the time or computation required to work forward through the key schedule, essentially trivial compared to the actual encryption round times? Moreover, it really matters only for the first block decrypted doesn't it, as long as you save the starting point for the decryption key schedule? Is the difference enough to really worry about?

2. Of our four present modes, two use only encryption (that is OFB and CFB modes). If we add a counter mode, that will do encryption, I would think. So, for these, it makes sense to make encryption the more efficient operation. Or would you argue that we should then redefine OFB, CFB and counter modes to yes the decryption operation?

Regards,
Bill Burr

– – –

From: "Phinney, Tom (AZ15)" <tom.phinney@honeywell.com>
To: "'Bill Burr'" <william.burr@nist.gov>
Subject: RE: Rijndael - Encryption vs decryption
Date: Thu, 5 Oct 2000 14:05:48 -0700

Bill,

First, the choice of e(x) or d(x) could be mode-specific. I agree that OFB, CFB and the hoped-for counter mode would use only encryption, and in those cases the currently-defined encryption mode is preferable. Presumably, that's why Daemen and Rijmen made the e(x) vs. d(x) choice they did.

To my mind, the default choice for ECB is still open; that's really the issue I had in mind when I sent my note. I apologize for my lack of clarity about this; I should have waited until morning to review it before sending it.

The problem with OFB, CFB and counter modes, as we know, is stream synchronization -- both sender and receivers must be able to identify the proper point in the keystream. This sometimes necessitates recovery, or at least counting, of messages which a receiver misses. Such recovery is not a problem with file transfer or browser page rendering, since they require the datastream to be recovered or retransmitted for other reasons.

In the industrial real-time control environment which I represent, such recovery is a problem. Our environment uses multi-cast (unconnected) and multi-point connected transmission modes for conveying current-state data. Obsolete state data is not of interest, and no recovery is attempted. Most state-data transmissions are hard scheduled, and there is no communications time available for recovery of previously-missed information.

This distinction between valuing old and new data has been referred to as the "milk vs. wine" problem -- one discards old milk and new wine. The industrial control industries' difficulties with AES use are anticipated to be associated with the "milk", not the "wine".

Since the AES announcement on Monday, I have already initiated a discussion process among the U.S. Technical Advisory Group for IEC/SC 65C -- on industrial communications standards -- focussing on where and how we should plan to use AES. This letter is forcing me to progress that analysis now. (And for that I do give you thanks. :)

Industrial communications protocols will use AES to encrypt only higher-layer parts of messages -- those representing application data or commands. In most cases this information is very small, and will require padding to reach the minimum AES block size. In such cases there is no potential reuse of keying material within a single message encryption or decryption.

The choices of mode for these industrial multi-cast and multi-point communications seem to be either (1) ECB, or a variant of counter mode where the counter is derived from (2) an inferred message sequence number or (3) a quantized shared sense of time. Of these, the best would be (2) because of the potential to precompute keying material during periods of low compute usage, followed by (1). Alternative (3) is the worst because its time-sense sharing will occasionally give rise to the need of receivers to "hunt" for the quantized prior-time instant used by the sender.

A typical plant environment consists of thousands of low-power microcomputers, each monitoring one or a few sensors, or controlling an actuator and associated feedback sensors. The vast majority of these "field devices" are built around simple sensors such as limit switches or silicon temperature sensors. The sensors, microcomputer (typically 8-bit) and communications circuitry are powered from the same twisted pair on which the device communicates. This has been the industry practice for the last 25 years with analog 4-20 mA signaling, and continues with today's various digital fieldbusses.

If we use alternative (2), a message sequence number, for multi-point connections, then we have the difficulty of synchronizing new or recovering receivers with the sender. This is not a severe technical problem, but will necessitate changes in current IEC-standard communications protocols which the other two alternatives do not require. Because certain European markets

have legal requirements for conformance to IEC standards, and because changes in the standards take time and will introduce incompatibilities, the progression of such a solution will be difficult. (Fortunately, I'm the convenor of the standards groups involved here, so at least there is no management impediment.)

If we use alternative (1), ECB, for multi-point connections, then it is practical to save the decryption starting point along with the other state information for that connection. One weakness of ECB is that state data tends to be slowly changing, and we may find it necessary to conceal that change or lack thereof from eavesdroppers. This could be solved by prefixing an unencrypted salt value to the encrypted information field, providing non-repeatability of cyphertext (CT) for unchanging plaintext (PT).

The time-based pseudo-counter mode of (3) solves the CT repeatability problem at the expense of more difficult synchronization. The reality of modern control systems is that they _do_ have a highly synchronized sense of time, usually to within one or a few milliseconds. This is a market >requirement for sequence-of-events recording, and all vendors provide it, at least in part as a CYA measure in anticipation of the need to reconstruct events following a major incident involving loss of life or plant of creation of a new superfund site. Approximate synchronization is easy, but exact synchronization (i.e., all devices incrementing their "counters" in parallel) is not.

Based on the above, I believe I will back alternative (2) for multi-point connections and alternative (1) with a per-message salt for multi-cast connectionless traffic. I will initiate discussion of the needed changes in the IEC Type 1 and Type 5 Fieldbus standards, which are those used by the U.S.-based Fieldbus Foundation (FF), and in the FF specs themselves. (I am also on the FF's Technical Steering Committee and head it's Architectural Control Team, so this I can do.)

To get back to my original message and your considered reply, the only real issue is with ECB. Connectionless communications (what we once called datagrams) will be the main industrial communications use of ECB. In this case, per-message keying agility is required by receivers and reception should be the more rapid operation. One-time precomputation of the starting round key for decryption is not onerous; that can be associated with the other information for a sending information stream. However, storage of all the round keys for each ECB stream would be an imposition. I don't recall whether Rijndael requires that, or whether the round key computation is invertible so that successive decryption round keys can be computed from the starting decryption round key. If the latter, then the choice of d(x) vs. e(x) is not an issue in the industrial markets.

Disk encryption gives rise to a similar analysis. Since SEAL is still patent-protected, AES will be used for disk file encryption, sometimes with address-dependent keying. If there is no address-dependent keying, then the analysis of the preceeding paragraph applies. If the keying is address-dependent, then the suggested encryption mode would seem to be d(x), with the suggested decryption mode being e(x). However, real-world disk transfers involve a large number of AES 16-byte blocks. In this environment, the precomputation cost of the d(x) first round key is negligible, and e(x) vs. d(x) doesn't make much real difference.

The selected algorithm in a driver for disk encryption is clearly an implementer's local choice, but NIST's recommendation will be used without reconsideration by most implementers.

In summary, I believe that NIST should consider recommending use of Rijndael's d(x) function for ECB encryption, and its e(x) for ECB decryption. The advantages are slight (and probably negligible) if the round key computation function is invertible, so that successive decryption round keys can be computed from a stored pre-computed initial decryption round key. Otherwise the advantages could be significant, particularly for short messages such as are common in the industrial real-time control environment.

For all other modes -- OFB, CFB and the various counter modes -- Rijndael's e(x) function should be used for encryption, and decryption need not be used at all.

I hope that this somewhat-rambling discussion clarifies my earlier message. I regret that that message was so poorly thought out and stated, but am glad that I sent it or this interchange would not have occurred.

Please feel free to use this message as you feel appropriate. There is no need for it to appear in a public record, but you are free to so incorporate it if that seems useful. Likewise for my earlier ill-considered message.
>
Thanks again for your considerate response, and for inducing me to the above analysis and decisions,

Best regards,
Tom Phinney
U.S. Technical Advisor for IEC/SC 65C
Convenor, IEC/SC 65C/WG 1 and /MT 9

========== Addendum ==========
It is possible to combine the underlying idea of the message sequence number mode of (2) with the ECB mode of (1). In this case the sequence number would be used to generate a low-computational-cost non-cryptographic-quality message-altering byte stream, rather than as a source of keying information. After XORing the two, ECB would be used to encrypt the modified PT, removing any observable correlation between CTs. The protocol changes required by (2) would still be required, but the proposed e(x) vs. d(x) swap would no longer be an issue.

Tom

– – –

Date: Thu, 05 Oct 2000 17:21:24 -0400
From: Jim Foti <jfoti@nist.gov>
Subject: RE: Rijndael - Encryption vs decryption

>To get back to my original message and your considered reply, the only real issue is with ECB.
>Connectionless communications (what we once called datagrams) will be the main industrial

>communications use of ECB. In this case, per-message keying agility is required by receivers
>and reception should be the more rapid operation. One-time precomputation of the starting
>round key for decryption is not onerous; that can be associated with the other information for a
>sending information stream. However, storage of all the round keys for each ECB stream would
>be an imposition. I don't recall whether Rijndael requires that, or whether the round key
>computation is invertible so that successive decryption round keys can be computed from the
>starting decryption round key. If the latter, then the choice of d(x) vs. e(x) is not an issue in the
>industrial markets.

Since Rijndael has on-the-fly keying for decryption I would think that this would not be an issue...

[snip]

>In summary, I believe that NIST should consider recommending use of Rijndael's d(x) function
>for ECB encryption, and its e(x) for ECB decryption. The advantages are slight (and probably
>negligible) if the round key computation function is invertible, so that successive decryption
>round keys can be computed from a stored pre-computed initial decryption round key.
>Otherwise the advantages could be significant, particularly for short messages such as are
>common in the industrial real-time control environment.

>For all other modes -- OFB, CFB and the various counter modes -- Rijndael's e(x) function
>should be used for encryption, and decryption need not be used at all.

Looking at the Rijndael spec, one interesting thing to note is that in the sections specifying the algorithm itself, the authors refer to the "Cipher" and "Inverse Cipher", instead of encryption and decryption functions. In one place, they refer to "encryption" in terms of the ECB mode: "The cipher input bytes (the 'plaintext' if the mode of use is ECB encryption)..."

So, here's a thought: In the [AES] FIPS, which could really be thought of as "mode-independent", perhaps we should keep the references to "Cipher" and "Inverse Cipher". Then, in the modes [FIPS], we can say specifically that for ECB, encryption uses the Cipher function, etc. Then, we could add another mode, say "Reverse ECB" (RECB), which specifies that RECB encryption uses the Inverse Cipher function, etc. Is such a mode already defined somewhere, possibly under another name?

It seems to me that this arrangement would still allow us to do the intuitive: use the Cipher to "encrypt" for the majority of the modes. Meanwhile, those apps which absolutely needed the decrypt function to be faster could use RECB mode.

This might be sufficient for Phinney (and would not necessitate a slew of new modes), since he also said that

>For all other modes -- OFB, CFB and the various counter modes -- Rijndael's e(x) function
>should be used for encryption, and decryption need not be used at all.

Date: Tue, 10 Oct 2000 17:57:16 -0400
From: csjutla <csjutla@watson.ibm.com>
Subject: Patent Letter

Hi,
Here is the patent licensing process that IBM intends to follow.
Please see the attached letter.
-Charanjit

– – –

October 4, 2000

Mr. Edward Roback
NIST
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20899-8930

Dear Mr. Roback:

This letter is to inform NIST of IBM's patent licensing practices as related to standardization activities. It is the belief of IBM that the IBM patent application listed in Attachment A is relevant to a mode of operation proposed by IBM for inclusion in Federal Information Processing Standards for Modes of Operation for Symmetric Key Block Ciphers.

In the event the proposed standard is adopted, and the standard can not be practiced without the use of the aforementioned patent, IBM agrees upon request to grant a non-exclusive license under this patent on a nondiscriminatory basis and on reasonable terms and conditions including its then current royalty rates and provided a similar grant under licensee's patents within the scope of the license granted to licensee is made available upon request to IBM. Should other IBM patents be required to implement this standard, they will be licensed under the same non-discriminatory, reasonable terms and conditions basis.

Any requests for license information may be directed to:

Frank Squillante
Director of Licensing
International Business Machines Corporation
North Castle Drive
Armonk, NY 10504

Internet ID: fps@us.ibm.com
Telephone: 914-765-4260
Fax: 914-765-4360

IBM is pleased to make this offer in support of the standardization activities in NIST. If you have any questions, please let me know.

Sincerely,

Stephen M. Matyas, Jr.
for IBM Corporation

Attachment A: SUBJECT PATENT APPLICATIONS

Charanjit Jutla, “Encryption schemes with almost free integrity awareness,” docket number YOR920000194US1. Filed April 14, 2000.

From: "Virgil D. Gligor" <gligor@po4.glue.umd.edu>
Date: Wed, 11 Oct 2000 11:39:28 -0400 (EDT)
Subject: Full disclosure

The proposals made in the paper
``Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes,"
submitted to the NIST Modes of Operation Workshop are the subject of three patent applications VDG Inc has submitted since January 1, 2000. The authors of those applications are Virgil D. Gligor and Pompiliu Donescu.

Although I recall mentioning this in prior conversations, I'd like to make this part of the record in view of NIST's posted disclosure requirement.

Best regards,
Virgil

Date: Wed, 11 Oct 2000 14:55:46 -0700 (PDT)
From: Lars Knudsen <lknudsen@cs.ucsd.edu>
Subject: counter mode

A comment on the second item in email from Eric Boesch from 27.12.99

It is correct what you say, that the expected cycle length of the output feedback mode using a 128-bit random function is 2^64. But a 128-bit block cipher is not a random function, it is bijective and the expected cycle length of the output feedback mode is here 2^127, not 2^64.

You also say that output feedback mode does not offer any advantages over counter mode. However, in counter mode an attacker knows the difference (modulo 2^128) of any two inputs to the block cipher, which is not the case in output feedback mode. I'm not saying that this is a major disadvantage, but it is at least a difference.

Lars
Lars R. Knudsen, Visiting Prof., University of California
San Diego, Dept. of Comp.Science and Engineering, Off. 4101,
Tlph. +1 (858) 534-6265, Facs. +1 (858) 534-7029.

Subject: Intellectual property issues
From: Paul Crowley <paul@cluefactory.org.uk>
Date: 14 Oct 2000 16:42:34 +0100

NIST made the right decision in the AES contest by requiring all those submitting proposed standards to provide a worldwide, royalty-free license to any of their intellectual property when used to implement the AES. The excellent quality of the submissions shows that this was not a problematic barrier to contribution. I hope that the same policy is taken with regard to modes of operation. For example, the very useful IAPM mode proposed by Charanjit S. Jutla is covered by patents held by IBM; it seems plausible that IBM might provide the necessary licenses for this to be used on the same terms as the AES if this were a requirement for becoming a US national standard.

Thank you for your consideration,

paul@cluefactory.org.uk
http://www.cluefactory.org.uk/paul/

– – –

From: Paul Crowley <paul@cluefactory.org.uk>
Date: 17 Oct 2000 15:09:30 +0100

Elaine Barker <elaine.barker@nist.gov> writes:
> IBM has supplied a letter. See page 20 of the comments on the modes page (http://www.nist.gov/modes). Elaine

Thank-you! This letter indicates that IBM do not intend to grant a worldwide, royalty-free license as I have hoped. I would still urge NIST not to standardise any mode that cannot be made available on the terms originally demanded for AES itself - the usefulness of nearly halving the computational demands of encryption and MAC would certainly be outweighed by the inconvenience of having to license an IBM patent under the tems they demand. Other approaches to fast MACs, including SHA based approaches, may be more appropriate, though I still hope IBM can be persuaded to change their position.
--
paul@cluefactory.org.uk
http://www.cluefactory.org.uk/paul/

Date: Thu, 19 Oct 2000 22:28:08 +0300 (EET DST)
From: Helger Lipmaa <helger@tml.hut.fi>
Subject: Comment on Hashing Modes

Hi,
here comes a belated comment on the hashing modes that I still hope to be discussed in the Encryption Modes workshop. It's belatedness is purely my fault and is a result of my workaholic lifestyle.

Lately, Eric Young posted a message on the coderpunks mailing list, announcing that the SHA-256 hashing algorithm has throughput performance ~9.7 Mbytes/s. On the other hand, my own implementation of Rijndael, the new AES, has throughput performance of ~15.8 Mbytes/s, which is almost twice the performance of SHA-256. Both measurements were done on a Pentium II 256, and correspond to a C implementation.

Traditionally, customized hash algorithms have been several times faster than the standard block ciphers; however, with the AES and the SHA-256 it now seems to be the other way around. This makes an AES-derived (unkeyed) hash mode now practically more relevant as before.

The well-known benefits of having a block-cipher based standard construction are:
* simplicity - no need to implement two, usually very different, algorithms.
* security - given that block cipher is secure, it _seems_ that for several of the standard constructions, the resultant hash function is also secure. Hence, one does not spend time to cryptanalyze two functions, aassurance we have in a block cipher could be transfered to a hash function.

Now, given that Rijndael is almost twice faster than the SHA-256, and that the time required to expand a 128-bit key is about 60% of the speed required to encrypt a 128-bit block (in my implementation), I propose to study the possibility---this is not equivalent of proposing a standard---of having a standard hash function with the next parameters:

* Underlying block cipher is a 256-bit block length, 256-bit key length Rijndael, with 14 rounds, as specified in the Rijndael original submission document.
* One of the standard block cipher -> hash function conversions is applied. Candidates include: Matyas-Meyer-Oseas, Davies-Meyer, Miyaguchi-Preneel. All three modes require one key scheduling and one run of the Rijndael per block.

(Cautionary) notes:
1. There is not much cryptanalysis of the 256/256-Rijndael, so the number of rounds could be raised to 16 or 18.
2. I do not have an implementation of the 256/256-Rijndael, so I cannot yet measure the speed. Empirically, this variant of Rijndael should have better throughput than the 128/128-Rijndael (for both including the time for key scheduling), and therefore faster than SHA-256.
3. Implementation of both the 128/128-Rijndael and the 256/256-Rijndael would still be more complicated than the implementation of the first. However, in software it is quite easy to

implement both by the same subroutine. (The code, submitted originally with Rijndael proposal is a good, even if not efficient, example.)
4. The key schedule of Rijndael, which some think to be over-simplistic, has been less studied than it's encryption function. More research should be directed in this direction.

But: However, if Rijndael would badly interfere with one of the standard modes, one should reconsider its suitability as an AES.

Based on the first three cautionary notes, another proposal would be to use one of the double-length hash function constructions by Knudsen and Prenel ("Hash Functions Based on Block Ciphers and Quaternary Codes", 1996), built on the 128/128-bit Rijndael. Unfortunately the latter construction would be far less efficient than the first proposal.

I hope to give more feedback (a more concrete proposal) after the workshop, including measurements data of an actual optimized 256/256-Rijndael. I'd also like to thank David Wagner for his---as always---useful comments.

Regards,
Helger