Comparison of CBC MAC Variants and Comments on NIST's Consultation Paper

Tetsu Iwata

Department of Computer and Information Sciences,
Ibaraki University
4–12–1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan
iwata@cis.ibaraki.ac.jp

May 5, 2003

1 Summary

In this note, we present a comparison of the following CBC MAC variants:

- RMAC by Jaulmes, Joux and Valette [9, 10, 5, 6],
- EMAC from ISO 9797-1 [1, 13],
- XCBC by Black and Rogaway [3, 4],
- TMAC by Kurosawa and Iwata [12], and
- OMAC by Iwata and Kurosawa [7].

We consider two RMACs. One is RMAC defined in NIST's draft [5], which we write RMAC1, with parameter set IV or V, where AES is used as the underlying block cipher, and uses a nonce R. The other one is RMAC mode 2 stated in NIST's consultation paper [6], which we write RMAC2, where AES128 is used to compute the CBC MAC tag, AES256 is used to encrypt it, and uses a nonce R. We write RMAC to mean both RMAC1 and RMAC2 [1].

2 Security Comparison

In this section, we show a security comparison.

- Let $F[E]$ denote a MAC F with E as the underlying block cipher, where F is RMAC, EMAC, XCBC, TMAC, or OMAC, and E is TDES112, TDES168, AES128, AES192, or AES256. We write TDES to mean both TDES112 and TDES168, and we write AES to mean AES128, AES192 and AES256. RMAC[TDES] is considered just for completeness.
- $\mathtt{Adv}^{\mathsf{mac}}_{F[E]}(t, q, L)$ is the maximum forgery probability, where the maximum is over all adversaries who run in time at most t, make at most q queries, and the total length of all queries are at most L blocks.
- $\mathtt{Adv}^{\mathsf{prp}}_{E}(t', q')$ is the the maximum distinguishing probability between the block cipher E and a randomly chosen permutation, where the maximum is over all adversaries who run in time at most t', and make at most q' queries.

[1] We do not consider RMAC with randomly chosen R in the original submission [9, 10], since a nonce is allowed in [5, 6].

- $\mathtt{Adv}^{\mathsf{prp-rka}}_{\Phi_k{}^{\oplus},E}(t'',q'')$ is the the maximum distinguishing probability between the block cipher E and a family of randomly chosen permutations (ideal block cipher) in the related key attack scenario, where the maximum is over all adversaries who run in time at most t'', and make at most q'' queries.
- m denotes the length of the tag. If there is no output truncation, then $m = 64$ for TDES and $m = 128$ for AES.

2.1 Security Bounds with TDES

- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{RMAC[TDES]}}(t,q,L) \le \dfrac{3L^2}{2^{64}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{TDES}}(t',q') + \mathtt{Adv}^{\mathsf{prp-rka}}_{\Phi_k{}^{\oplus},\mathrm{TDES}}(t'',q'')$,
  where $t' = t + O(L)$, $q' = L$, $t'' = t' + O(L)$, and $q'' = q$ [11, p .2].
- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{EMAC[TDES]}}(t,q,L) \le \dfrac{3L^2}{2^{64}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{TDES}}(t',q') + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{TDES}}(t'',q'')$,
  where $t' = t + O(L)$, $q' = L$, $t'' = t + O(L)$ and $q'' = q$ [13, Theorem 1].
- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{XCBC[TDES]}}(t,q,L) \le \dfrac{3L^2}{2^{64}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{TDES}}(t',q')$,
  where $t' = t + O(L)$ and $q' = L$ [8, Theorem 3.1].
- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{TMAC[TDES]}}(t,q,L) \le \dfrac{3L^2}{2^{64}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{TDES}}(t',q')$,
  where $t' = t + O(L)$ and $q' = L$ [8, Theorem 3.1].
- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{OMAC[TDES]}}(t,q,L) \le \dfrac{4L^2}{2^{64}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{TDES}}(t',q')$,
  where $t' = t + O(L)$ and $q' = L + 1$ [8, Theorem 3.1].

2.2 Security Bounds with AES

- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{RMAC[AES]}}(t,q,L) \le \dfrac{3L^2}{2^{128}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{AES}}(t',q') + \mathtt{Adv}^{\mathsf{prp-rka}}_{\Phi_k{}^{\oplus},\mathrm{AES}}(t'',q'')$,
  where $t' = t + O(L)$, $q' = L$, $t'' = t' + O(L)$, and $q'' = q$ [11, p .2].
- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{EMAC[AES]}}(t,q,L) \le \dfrac{3L^2}{2^{128}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{AES}}(t',q') + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{AES}}(t'',q'')$,
  where $t' = t + O(L)$, $q' = L$, $t'' = t + O(L)$ and $q'' = q$ [13, Theorem 1].
- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{XCBC[AES]}}(t,q,L) \le \dfrac{3L^2}{2^{128}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{AES}}(t',q')$,
  where $t' = t + O(L)$ and $q' = L$ [8, Theorem 3.1].
- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{TMAC[AES]}}(t,q,L) \le \dfrac{3L^2}{2^{128}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{AES}}(t',q')$,
  where $t' = t + O(L)$ and $q' = L$ [8, Theorem 3.1].
- $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{OMAC[AES]}}(t,q,L) \le \dfrac{4L^2}{2^{128}} + \dfrac{1}{2^m} + \mathtt{Adv}^{\mathsf{prp}}_{\mathrm{AES}}(t',q')$,
  where $t' = t + O(L)$ and $q' = L + 1$ [8, Theorem 3.1].

2.3 Comments

- For RMAC[TDES], we added $0.5L^2/2^{64} + 0.5q^2/2^{64} \le L^2/2^{64}$ to the bound $2L^2/2^{64}$ stated in [11], since we consider a random permutation rather than a random function. Also, we added $1/2^m$ since we consider a forgery probability rather than a distinguishing probability. The same comment goes for EMAC[TDES], RMAC[AES], and EMAC[AES].
- Bounds for XCBC, TMAC and OMAC are formally proved in [8].

2.4 Discussions

In NIST's consultation paper [6, p. 2, Sec. Security, Proof bounds with AES], it is stated that the security bound of RMAC[AES] is $(517L + t)/2^{128}$. However:

- The bound requires randomly chosen R. Thus, this bound has nothing to do with RMAC with nonce R.
- The corresponding proof [9, Sec. 4.1] is incorrect as was suggested by Rogaway [14, p. 3, Sec. 2.2].

Thus, it is not appropriate to use the bound $(517L + t)/2^{128}$ for a comparison, and we believe that the security bound of RMAC[AES] should be understood as the bound stated in [11], which is apparently *worse* than that of EMAC as discussed below.

In RMAC submitter's comment [11] and NIST's consultation paper [6], it is stated that RMAC with nonce R is as secure as EMAC. However, it seems to be false.

Rogaway's comment [14] already illustrates this. In [14, p. 3, Sec. 2.2], it is shown that assuming the underlying block cipher to be a pseudorandom permutation is not sufficient for proving the security of RMAC. This fact does not change even if the RMAC submitters provide the reduction based security bound.

It may be interpreted as follows: Suppose that the underlying block cipher is a pseudorandom permutation (that is, $\mathtt{Adv}^{\mathsf{prp}}_{E}(t', q')$ is sufficiently small). Then EMAC is a secure MAC, while RMAC may not (since $\mathtt{Adv}^{\mathsf{prp-rka}}_{\Phi_k{}^{\oplus},E}(t'', q'')$ may be large). Therefore, the security bound of EMAC is better than that of RMAC.

Here is another example: Suppose that we have found a related key attack against AES. That is, $\mathtt{Adv}^{\mathsf{prp-rka}}_{\Phi_k{}^{\oplus},\mathrm{AES}}(t'', q'')$ is no more negligible. Then the security bound of RMAC becomes meaningless, while this fact has nothing to do with the security bounds of EMAC.

Although a related key attack against AES may not be found, this clearly illustrates the security gap between EMAC and RMAC, and, at least, we may conclude that the security bound of EMAC is better than that of RMAC with nonce R.

NIST's consultation paper states that the security bound of EMAC is better than that of XCBC. However, in [8], we have proved that XCBC (and TMAC and OMAC) are as secure as EMAC [2]. In fact, for both TDES and AES, we see that there is no significant difference among the security of EMAC, XCBC, TMAC and OMAC. These four MACs are equally secure.

3 Efficiency Comparison

In this section, we show an efficiency comparison.

In what follows:

- "K len." denotes the key length.
- "$\#K$ sche." denotes the number of block cipher key schedulings. It is desirable to minimize this, since TDES and AES have non-trivial key scheduling cost. For RMAC, it requires one block cipher key scheduling each time generating a tag.
- "$\#M$" denotes the number messages which the sender has MACed.
- "$\#E$ invo." denotes the number of block cipher invocations to generate a tag for a message M, assuming $|M| > 0$.
- "$\#E$ pre." denotes the number of block cipher invocations during the pre-processing time. These block cipher invocations can be done without the message.
- "+`kst`" means that the key separation technique is used. OMAC does not need the key separation technique since its key length is optimal in its own form.
- For RMAC2, we assume that AES128 is used to compute the CBC MAC tag, and AES256 is used to encrypt it.

[2] We believe that the results in [8] are mostly trivial for those who are working in this area.

3.1 Efficiency Comparison without Key Separation Technique

Table 1. Efficiency Comparison with TDES112.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[TDES112]	224	$1+\#M$	$1+\lceil(\lvert M\rvert+1)/64\rceil$	0
EMAC[TDES112]	224	2	$1+\lceil(\lvert M\rvert+1)/64\rceil$	0
XCBC[TDES112]	240	1	$\lceil\lvert M\rvert/64\rceil$	0
TMAC[TDES112]	176	1	$\lceil\lvert M\rvert/64\rceil$	0
OMAC[TDES112]	112	1	$\lceil\lvert M\rvert/64\rceil$	1

Table 2. Efficiency Comparison with TDES168.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[TDES168]	336	$1+\#M$	$1+\lceil(\lvert M\rvert+1)/64\rceil$	0
EMAC[TDES168]	336	2	$1+\lceil(\lvert M\rvert+1)/64\rceil$	0
XCBC[TDES168]	296	1	$\lceil\lvert M\rvert/64\rceil$	0
TMAC[TDES168]	232	1	$\lceil\lvert M\rvert/64\rceil$	0
OMAC[TDES168]	168	1	$\lceil\lvert M\rvert/64\rceil$	1

Table 3. Efficiency Comparison with AES128.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[AES128]	256	$1+\#M$	$1+\lceil(\lvert M\rvert+1)/128\rceil$	0
RMAC2[AES128]	384	$1+\#M$	$1+\lceil\lvert M\rvert/128\rceil$	0
EMAC[AES128]	256	2	$1+\lceil(\lvert M\rvert+1)/128\rceil$	0
XCBC[AES128]	384	1	$\lceil\lvert M\rvert/128\rceil$	0
TMAC[AES128]	256	1	$\lceil\lvert M\rvert/128\rceil$	0
OMAC[AES128]	128	1	$\lceil\lvert M\rvert/128\rceil$	1

Table 4. Efficiency Comparison with AES192.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[AES192]	384	$1+\#M$	$1+\lceil(\lvert M\rvert+1)/128\rceil$	0
EMAC[AES192]	384	2	$1+\lceil(\lvert M\rvert+1)/128\rceil$	0
XCBC[AES192]	448	1	$\lceil\lvert M\rvert/128\rceil$	0
TMAC[AES192]	320	1	$\lceil\lvert M\rvert/128\rceil$	0
OMAC[AES192]	192	1	$\lceil\lvert M\rvert/128\rceil$	1

Table 5. Efficiency Comparison with AES256.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[AES256]	512	$1+\#M$	$1+\lceil(\lvert M\rvert+1)/128\rceil$	0
EMAC[AES256]	512	2	$1+\lceil(\lvert M\rvert+1)/128\rceil$	0
XCBC[AES256]	512	1	$\lceil\lvert M\rvert/128\rceil$	0
TMAC[AES256]	384	1	$\lceil\lvert M\rvert/128\rceil$	0
OMAC[AES256]	256	1	$\lceil\lvert M\rvert/128\rceil$	1

3.2 Efficiency Comparison with Key Separation Technique

Table 6. Efficiency Comparison with TDES112 and Key Separation Technique.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[TDES112] + `kst`	112	$2+\#M$	$1+\lceil(\lvert M\rvert+1)/64\rceil$	4
EMAC[TDES112] + `kst`	112	3	$1+\lceil(\lvert M\rvert+1)/64\rceil$	4
XCBC[TDES112] + `kst`	112	2	$\lceil\lvert M\rvert/64\rceil$	4
TMAC[TDES112] + `kst`	112	2	$\lceil\lvert M\rvert/64\rceil$	3
OMAC[TDES112]	112	1	$\lceil\lvert M\rvert/64\rceil$	1

Table 7. Efficiency Comparison with TDES168 and Key Separation Technique.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[TDES168] + `kst`	168	$2+\#M$	$1+\lceil(\lvert M\rvert+1)/64\rceil$	6
EMAC[TDES168] + `kst`	168	3	$1+\lceil(\lvert M\rvert+1)/64\rceil$	6
XCBC[TDES168] + `kst`	168	2	$\lceil\lvert M\rvert/64\rceil$	5
TMAC[TDES168] + `kst`	168	2	$\lceil\lvert M\rvert/64\rceil$	4
OMAC[TDES168]	168	1	$\lceil\lvert M\rvert/64\rceil$	1

Table 8. Efficiency Comparison with AES128 and Key Separation Technique.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[AES128] + `kst`	128	$2+\#M$	$1+\lceil(\lvert M\rvert+1)/128\rceil$	2
RMAC2[AES128] + `kst`	128	$2+\#M$	$1+\lceil\lvert M\rvert/128\rceil$	2
EMAC[AES128] + `kst`	128	3	$1+\lceil(\lvert M\rvert+1)/128\rceil$	2
XCBC[AES128] + `kst`	128	2	$\lceil\lvert M\rvert/128\rceil$	3
TMAC[AES128] + `kst`	128	2	$\lceil\lvert M\rvert/128\rceil$	2
OMAC[AES128]	128	1	$\lceil\lvert M\rvert/128\rceil$	1

Table 9. Efficiency Comparison with AES192 and Key Separation Technique.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[AES192] + `kst`	192	$2+\#M$	$1+\lceil(\lvert M\rvert+1)/128\rceil$	3
EMAC[AES192] + `kst`	192	3	$1+\lceil(\lvert M\rvert+1)/128\rceil$	3
XCBC[AES192] + `kst`	192	2	$\lceil\lvert M\rvert/128\rceil$	4
TMAC[AES192] + `kst`	192	2	$\lceil\lvert M\rvert/128\rceil$	3
OMAC[AES192]	192	1	$\lceil\lvert M\rvert/128\rceil$	1

Table 10. Efficiency Comparison with AES256 and Key Separation Technique.

Name	K len.	$\#K$ sche.	$\#E$ invo.	$\#E$ pre.
RMAC1[AES256] + `kst`	256	$2+\#M$	$1+\lceil(\lvert M\rvert+1)/128\rceil$	4
EMAC[AES256] + `kst`	256	3	$1+\lceil(\lvert M\rvert+1)/128\rceil$	4
XCBC[AES256] + `kst`	256	2	$\lceil\lvert M\rvert/128\rceil$	4
TMAC[AES256] + `kst`	256	2	$\lceil\lvert M\rvert/128\rceil$	3
OMAC[AES256]	256	1	$\lceil\lvert M\rvert/128\rceil$	1

3.3 Discussions

None of RMAC, EMAC, XCBC, TMAC and OMAC is optimal in *all* efficiency measures: "K len.," "$\#K$ sche.," "$\#E$ invo.," and "$\#E$ pre." There is a tradeoff among the above four measures.

Key length: In Tables 1–5, we see that OMAC gives the best performance. We note that OMAC is as secure as EMAC, XCBC, and TMAC despite of its optimal key length.

In Tables 6–10, the key lengths of RMAC, EMAC, XCBC and TMAC can be reduced to the optimal length. But the cost appears in the number of key schedulings and the number of block cipher invocations during the pre-processing time.

Number of key schedulings: RMAC requires one block cipher key scheduling each time generating a tag. This might be significant since TDES and AES have non-trivial key scheduling cost.

In Tables 1–5, XCBC, TMAC and OMAC give the best performance, while EMAC requires two block cipher key schedulings.

In Tables 6–10, it is obvious that OMAC gives the best performance.

Number of block cipher invocations: In Tables 1–10, RMAC and EMAC requires one or two extra block cipher invocations compared to XCBC, TMAC and OMAC.

This overhead is significant for short messages, which is very common in practice.

Number of block cipher invocations during the pre-processing time: In Tables 1–5, only OMAC requires one block cipher invocation. But this is not very significant since:

- it can be done in an idle time, and
- it is performed infrequently compared to MAC generation. Thus one or two block cipher invocations to generate a tag in RMAC and EMAC is much more significant since it is performed on *each message.*

In OMAC, the gain for this cost is its optimal key length, which *completely eliminates* the need for the key separation technique. We believe this is a very reasonable and desirable tradeoff since:

- key separation technique is a very error-prone process in practice, and
- key separation technique is used in many environment, but if it is used, then other MACs have a significant key setup cost compared to OMAC.

In fact, we see that the performance of OMAC is far better than RMAC, EMAC, XCBC and TMAC in Tables 6–10. Especially, 6 TDES invocations during the pre-processing time in EMAC[TDES168] is a large cost.

We believe that OMAC gives the best performance and tradeoff in efficiency.

4 Conclusion

In the last line of [6], it is stated that "For applications that desire additional security assurance, RMAC can be retained as an option in parameter sets IV or V, possibly with the "mode 2" construction."

We believe this is not appropriate one to move forward, since:

- RMAC2 is *not* the best choice from a security view point as already pointed out in [14, 2, 15], as well as in this note, and
- it is *not* the best choice from an efficiency view point.

We also claim the following problems:

- RMAC2 uses two block ciphers: AES128 and AES256 (according to Table 1 of [6]), which increases the implementation cost, and
- it makes a restriction on the choice of the underlying block cipher (AES128 can not be used in the last encryption).

We think such a restriction significantly limits the wide spread use of the mode. RMAC1 has already this problem, since TDES is not allowed, we think the limitation in RMAC2 is even worse.

In contrast, there is no such limitations in EMAC, XCBC, TMAC and OMAC.

Recommendation: For security, EMAC, XCBC, TMAC and OMAC are better than RMAC. On the other hand, there is no significant difference among EMAC, XCBC, TMAC and OMAC. These four MACs are equally secure. For efficiency, OMAC gives the best performance. Therefore, our recommendation would be OMAC (especially OMAC1, which simplifies implementations). If it is not possible for some reason, then we recommend either XCBC or TMAC. They are more efficient than EMAC and as secure as EMAC. Again, if it is not possible for some reason, then we recommend to set $r = 0$ in both parameter sets IV and V (thus, choose EMAC). It is not as efficient as OMAC, TMAC and XCBC, but it is more secure than RMAC.

References

[1] A. Berendschot, B. den Boer, J. P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgård, M. Dichtl, W. Fumy, M. van der Ham, C. J. A. Jansen, P. Landrock, B. Preneel, G. Roelofsen, P. de Rooij, and J. Vandewalle. Final Report of RACE Integrity Primitives. *LNCS 1007,* Springer-Verlag, 1995.

[2] J. Black. *Comments on Draft SP 800-38B,* Available at `http://csrc.nist.gov/encryption/modes/`.

[3] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three key constructions. *Advances in Cryptology — CRYPTO 2000, LNCS 1880,* pp. 197–215, Springer-Verlag, 2000.

[4] J. Black and P. Rogaway. Comments to NIST concerning AES modes of operations: A suggestion for handling arbitrary-length messages with the CBC MAC. NIST submission. Available at `http://www.cs.ucdavis.edu/~rogaway/`.

[5] M. Dworkin. Recommendation for block cipher modes of operation: the RMAC authentication mode. *NIST special publication 800-38B,* Available at `http://csrc.nist.gov/encryption/modes/`.

[6] Consultation paper on the selection of a block cipher based MAC algorithm. Available at `http://csrc.nist.gov/encryption/modes/`.

[7] T. Iwata and K. Kurosawa. OMAC: One-Key CBC MAC. Pre-proceedings of *Fast Software Encryption, FSE 2003,* pp. 137–161, 2003. To appear in *LNCS,* Springer-Verlag.

[8] T. Iwata and K. Kurosawa. Stronger security bounds for OMAC, TMAC and XCBC. Manuscript. Available at Cryptology ePrint Archive, Report 2003/082, `http://eprint.iacr.org/`.

[9] É. Jaulmes, A. Joux, and F. Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. *Fast Software Encryption, FSE 2002, LNCS 2365,* pp. 237–251, Springer-Verlag, 2002. Full version is available at Cryptology ePrint Archive, Report 2001/074, `http://eprint.iacr.org/`.

[10] É. Jaulmes, A. Joux, and F. Valette. RMAC a randomized MAC beyond the birthday paradox limit. NIST submission. Available at `http://csrc.nist.gov/encryption/modes/`.

[11] É. Jaulmes, A. Joux, and F. Valette. A remark on the security of RMAC in the related-key security model. Available at `http://csrc.nist.gov/encryption/modes/`.

[12] K. Kurosawa and T. Iwata. TMAC: Two-Key CBC MAC. *Topics in Cryptology — CT-RSA 2003, LNCS 2612,* pp. 33–49, Springer-Verlag, 2003.

[13] E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *J.Cryptology,* vol. 13, no. 3, pp. 315–338, Springer-Verlag, 2000.

[14] P. Rogaway. Comments on NIST's RMAC Proposal. *Comments on Draft SP 800-38B,* Available at `http://csrc.nist.gov/encryption/modes/`.

[15] D. Wagner. Comments on RMAC. *Comments on Draft SP 800-38B,* Available at `http://csrc.nist.gov/encryption/modes/`.