Collision attacks on OCB

Niels Ferguson[*]

February 11, 2002

Abstract

We show that collision attacks are quite effective on the OCB block cipher mode. When a collision occurs OCB loses its authentication capability. To keep adequate authentication security OCB has to be limited in the amount of data it processes. This restriction is relevant to real-life applications, and casts doubt on the wisdom of using OCB.

Keywords: OCB, block cipher mode, collision attack

1 Introduction

OCB is a block cipher mode that provides both encryption and authentication [RBBK01b, RBBK01a]. It has been submitted to NIST as a proposal for standardisation, and is being considered for inclusion in the IEEE 802.11 wireless network standard. The main advantage of OCB is that it provides both authentication and encryption at about the same cost as CBC encryption. This makes OCB significantly faster then a combination of a classical encryption mode and an authentication mode. OCB comes with a proof of security [RBBK01a].

Since its publication OCB has received some attention, but very little cryptanalysis. We believe this has several reasons. Firstly, a proof of security seems to imply that cryptanalysis is useless. Secondly, the proof is quite complicated and analysis of the proof details is restricted to those people well-versed in these type of proof techniques. Thirdly, the OCB mode has been patented. This last reason has been the main reason for the author,

[*]MacFergus cryptography consulting, `niels@ferguson.net`

and several other reputable cryptanalysts he has spoken to, not to spend any time on OCB. Spending time on OCB will only help the patent-holders sell their licenses without any further compensation to the cryptanalyst. There is a significant cost, both direct and indirect, associated with using a patented algorithm. Given that OCB's computational advantage over the patent-free modes is at most a factor of 2, there seem to be only a few situations in which this cost is justified. We therefore expect OCB only to be used in niche applications where the performance advantage is critical. Thus the gain to the general community of cryptanalytical work on OCB is limited. All these factors lead to cryptanalysts spending their time on other interesting problems.

OCB is being considered for inclusion in IEEE 802.11 wireless network standard. The author was hired to advise on the security aspects of this standard, and in the course of that work had a look at OCB. The attacks in this paper were developed in the course of about a day. The fact that such simple attacks were found in such a short time is indicative that little, if any, cryptanalysis of OCB has been done up to now. This fact alone casts doubts on the wisdom of using OCB in any new design.

We present a two versions of a collision attack on OCB. If a particular collision on 128-bit values occurs, then an attacker can modify the message without being detected by the OCB authentication function. This implies that users who want to limit the probability of a successful forgery attempt to less than 2^{-64} cannot use OCB for more than 2^{32} blocks of data.

2 OCB

We first give a short description of OCB using the notation from [RBBK01a] with some of our own extensions. OCB encrypts a message M using a key K and a nonce N. Each nonce value should be used at most once for any given key.

We define

$$\begin{aligned} L &:= E_K(0) \\ R &:= E_K(N \oplus L) \end{aligned}$$

Both L and R are used to derive the various offsets. Let γ_i be the representation of the integer i in the canonical Gray code.[1] The offset Z_i is defined

[1]The Gray code is an alternative bijective mapping of the bit strings $\{0,1\}^n$ to the

by

$$Z_i := \gamma_i \cdot L \oplus R$$

where the multiplication $\cdot$ is done in the field GF(2^{128}) using the canonical conversion from bit strings to polynomials over GF(2). The primitive polynomial used to define the field is $x^{128} + x^7 + x^2 + x + 1$.

The message M is split into message blocks $M_1, \ldots, M_m$ where all but the last block are 128 bits long. The blocks $i = 1, \ldots, m - 1$ are encrypted as follows:

$$\begin{aligned} X_i &:= M_i \oplus Z_i \\ Y_i &:= E_K(X_i) \\ C_i &:= Y_i \oplus Z_i \end{aligned}$$

where C_i is the ciphertext block corresponding to M_i. The last block M_m is treated differently, because it might be shorter than a full block. Let μ be the length of M_m in bits, encoded as a 128-bit integer.

$$\begin{aligned} X_m &:= \mu \oplus x^{-1} \cdot L \oplus Z_m \\ Y_m &:= E_K(X_m) \\ C_m &:= M_m \oplus \text{first-}\mu\text{-bits}(Y_m) \end{aligned}$$

The authentication is provided by a tag that is computed in two steps:

$$\begin{aligned} S &:= M_1 \oplus \cdots \oplus M_{m-1} \oplus C_m 0^* \oplus Y_m \\ T &:= \text{first-}\tau\text{-bits}(E_K(S) \oplus Z_m) \end{aligned}$$

where $C_m 0^*$ is the last ciphertext block padded with zeroes to the full 128 bit length. (The value S is called "Checksum" in the specification.) The parameter τ specifies the number of bits in the authentication tag, allowing a tradeoff between the size of tag to be transmitted and the authentication security level. The OCB documentation suggests to use a τ in the range $32 \ldots 80$.

Although OCB can be used with any block cipher, it is clearly designed to be used with AES, or some other 128-bit block cipher. For the rest of this article we will assume that AES is used as the block cipher.

integers $0, \ldots, 2^n - 1$. The main interesting property is that $\gamma_i \oplus \gamma_{i+1}$ has Hamming-weight 1 for all i.

3 An attack on OCB

Our attack proceeds in two steps. First, we wait for a specific type of collision to occur on 128-bit values. Once this collision occurs, we can modify the ciphertext (and thereby the plaintext after decryption) without modifying the authentication tag. In effect, the authentication tag is entirely ineffective at stopping message forgery once the collision has occurred.

Our first attack will use a single large message. We choose random message blocks $M_1, \ldots, M_m$ subject to the restriction $M_4 \oplus M_5 \oplus M_6 \oplus M_7 = 0$. We let the sender encrypt this message giving the ciphertext $C_1, \ldots, C_m, T$. As the last message block is treated differently we will not use it or modify it in our attack.

The collision we want to have is a pair of indices (i, j) such that $X_i = X_j$. We define $\Delta := (\gamma_i \oplus \gamma_j) \cdot L$. When this collision occurs we also have

$$\begin{aligned} M_i \oplus \gamma_i \cdot L &= M_j \oplus \gamma_j \cdot L \\ \Delta &= M_i \oplus M_j \\ Y_i &= Y_j \\ C_i \oplus \gamma_i \cdot L &= C_j \oplus \gamma_j \cdot L \\ M_i \oplus C_i &= M_j \oplus C_j \end{aligned}$$

The last equality allows us to recognise this type of collision. We sort the blocks by $M_i \oplus C_i$ and look for collisions. For any pair (i, j) we have a chance of 2^{-128} that $X_i = X_j$. If $X_i \neq X_j$ then there is a 2^{-128} chance that $M_i \oplus C_i = M_j \oplus C_j$ anyway, so we expect that half the instances where $M_i \oplus C_i = M_j \oplus C_j$ are due to a collision of the X values. For the rest of the attack we assume that the collision is due to the X values. (This implies that when we find a $M_i \oplus C_i = M_j \oplus C_j$ the rest of the attack succeeds with a probability of 50%.)

Given that $X_i = X_j$, we can now recover L by

$$L = (M_i \oplus M_j) \cdot (\gamma_i \oplus \gamma_j)^{-1}$$

where the inversion is of course taken in GF(2^{128}). Knowledge of L allows us to change the message. We will indicate modified values with a prime. Choose any index d and set

$$C'_k := C_d \oplus (\gamma_d \oplus \gamma_k) \cdot L \qquad \text{for } k = 4, \ldots, 7$$

It is easy to see that for $k = 4, \ldots, 7$ we have $Y'_k = Y_d$ and therefore $X'_k = X_d$. We have

$$\begin{aligned}
M'_4 \oplus M'_5 \oplus M'_6 \oplus M'_7 &= X_d \oplus Z_4 \oplus X_d \oplus Z_5 \oplus X_d \oplus Z_6 \oplus X_d \oplus Z_7 \\
&= Z_4 \oplus Z_5 \oplus Z_6 \oplus Z_7 \\
&= (\gamma_4 \oplus \gamma_5 \oplus \gamma_6 \oplus \gamma_7) \cdot L \\
&= (110 \oplus 111 \oplus 101 \oplus 100) \cdot L \\
&= 0 \cdot L \\
&= 0
\end{aligned}$$

The original message had $M_4 \oplus M_5 \oplus M_6 \oplus M_7 = 0$ too, so this modification has not changed the checksum S or the tag T. Therefore, the modified message will be accepted by the receiver, but decrypt to a different message then what was originally sent.

3.1 Workload

The workload of this attack is less than the effort the sender and receiver of the message must expend to encrypt and decrypt the message. The only bulk data handling the attacker has to do is to store the values $M_i \oplus C_i$ in a hash table and detect collisions. All other computations are constant time.

3.2 Probability of success

If a collision in the X values occurs then this attack succeeds almost certainly. (We only have a problem if there is a second pair of blocks with $M_i \oplus C_i = M_j \oplus C_j$ that is not due to an X value collision. This is very unlikely to happen.)

The probability of having an X collision is about $(m-1)^2 2^{-128}/2 \approx m^2 2^{-129}$. For a message of 2^{32} blocks the probability of the attack succeeding is around 2^{-65}. For a message of 2^{40} blocks the probability of the attack succeeding is around 2^{-49}.

Our detection of the X-collisions is not perfect. We expect twice as many $M_i \oplus C_i = M_j \oplus C_j$ collisions, half of which are X-collisions. This means that once the attacker detects an $M_i \oplus C_i = M_j \oplus C_j$ collision he has a 50% chance of succeeding with this attack.

4 Extension to multiple messages

We now extend this attack to deal with multiple messages. Suppose we have a large number of messages encrypted using the same key but different nonces. We will again ignore the final blocks of each message as they are treated differently. To distinguish the various values in the different messages we add a subscript, or add an extra subscript in front of the existing one. Note that L does not require an extra subscript as it depends only on the key and not on the nonce. Therefore it is the same for all the messages.

We choose the messages with random message blocks subject to the restriction that for any q, $M_{4q} \oplus M_{4q+1} \oplus M_{4q+2} \oplus M_{4q+3} = 0$. It is possible to relax this restriction to cover sums over larger chunks, but this is the easiest variant of the attack to explain. There is a slight problem in that M_0 does not exist, so there is no first group of four. As long as the message sizes are reasonably large ($m > 100$) the chance of encountering this problem is very low, so we will ignore it.

We again look for a collision where $X_{a,i} = X_{b,j}$. If $a = b$ then the collision occurs within a single message and we can use the attack of the previous section. For the rest of this attack we will assume $a \neq b$, and thus $N_a \neq N_b$.

If $X_{a,i} = X_{b,j}$ we have

$$\begin{aligned}
Y_{a,i} &= Y_{b,j} \\
M_{a,i} &= X_{a,i} \oplus Z_{a,i} = X_{a,i} \oplus R_a \oplus \gamma_i \cdot L \\
M_{a,i} \oplus M_{b,j} &= R_a \oplus R_b \oplus (\gamma_i \oplus \gamma_j) \cdot L \\
C_{a,i} \oplus C_{b,j} &= R_a \oplus R_b \oplus (\gamma_i \oplus \gamma_j) \cdot L \\
M_{a,i} \oplus C_{a,i} &= M_{b,j} \oplus C_{b,j}
\end{aligned}$$

so we can use the same detection rule to find our X-collisions.

Unlike in our previous case we cannot recover L from this collision, but we don't have to. The differences between the message blocks gives us $\Delta := R_a \oplus R_b \oplus (\gamma_i \oplus \gamma_j) \cdot L$ which is enough.

Let $d := \lfloor i/4 \rfloor$ and $e := \lfloor j/4 \rfloor$. Let $s(k)$ be the value such that $\gamma_{s(k)} = \gamma_k \oplus \gamma_i \oplus \gamma_j$ for $k \in \{4d, 4d+1, 4d+2, 4d+3\}$. It is clear that $s(i) = j$. As the bit strings $\gamma_{4d}, \gamma_{4d+1}, \gamma_{4d+2}, \gamma_{4d+3}$ only differ in their last two bits we have $\{\, s(k) \mid k = 4d, 4d+1, \ldots, 4d+3 \,\} = \{4e, 4e+1, 4e+2, 4e+3\}$

We modify the message a as follows:

$$C'_{a,k} := C_{b,s(k)} \oplus \Delta \qquad \text{for } k = 4d, \ldots, 4d+3$$

and get

$$\begin{aligned}
C'_{a,k} &= Y_{b,s(k)} \oplus R_b \oplus \gamma_{s(k)} \cdot L \oplus R_a \oplus R_b \oplus (\gamma_i \oplus \gamma_j) \cdot L \\
&= Y_{b,s(k)} \oplus R_a \oplus (\gamma_{s(k)} \oplus \gamma_i \oplus \gamma_j) \cdot L \\
&= Y_{b,s(k)} \oplus R_a \oplus \gamma_k \cdot L \\
Y'_{a,k} &= C'_{a,k} \oplus R_a \oplus \gamma_k \cdot L = Y_{b,s(k)}
\end{aligned}$$

Because the properties of the s function we get

$$\begin{aligned}
\{Y'_{a,4d}, Y'_{a,4d+1}, Y'_{a,4d+2}, Y'_{a,4d+3}\} &= \{Y_{b,4e}, Y_{b,4e+1}, Y_{b,4e+2}, Y_{b,4e+3}\} \\
\{X'_{a,4d}, X'_{a,4d+1}, X'_{a,4d+2}, X'_{a,4d+3}\} &= \{X_{b,4e}, X_{b,4e+1}, X_{b,4e+2}, X_{b,4e+3}\}
\end{aligned}$$

and

$$\begin{aligned}
&M'_{a,4d} \oplus M'_{a,4d+1} \oplus M'_{a,4d+2} \oplus M'_{a,4d+3} \\
&\quad = X'_{a,4d} \oplus X'_{a,4d+1} \oplus X'_{a,4d+2} \oplus X'_{a,4d+3} \oplus (\gamma_{4d} \oplus \gamma_{4d+1} \oplus \gamma_{4d+2} \oplus \gamma_{4d+3}) \cdot L \\
&\quad = X'_{a,4d} \oplus X'_{a,4d+1} \oplus X'_{a,4d+2} \oplus X'_{a,4d+3} \oplus 0 \cdot L \\
&\quad = X_{b,4e} \oplus X_{b,4e+1} \oplus X_{b,4e+2} \oplus X_{b,4e+3} \\
&\quad = M_{b,4e} \oplus M_{b,4e+1} \oplus M_{b,4e+2} \oplus M_{b,4e+3} \\
&\quad = 0
\end{aligned}$$

In other words, the modification we made to the ciphertext has not changed the sum of the four affected message words, so the checksum is not changed. The modified message will be accepted by the receiver, but decrypt to a different message than what the sender sent.

5 Discussion

These attacks on OCB seem rather harmless. Most block cipher modes have collision attacks. Yet when looked at more closely these attacks are worrisome. For most block cipher modes, the consequences of a collision are rather minor. With OCB a collision leads to complete loss of an essential function. The resulting need to keep the probability of a collision down to very low levels means that OCB is incapable of processing large quantities of data with a single key.

Modern systems are designed to withstand an attacker that can perform 2^{128} steps of computation. This is a reasonable requirement for systems that will survive several decades [LV01]. The move to this security level was the major motivation behind the whole AES standardisation process. DES [NIS93] is limited by its 56-bit key, and 3DES [NIS99] is limited by its 64-bit block size. The larger block size and the larger key sizes allow AES-based systems to achieve a far higher security level than DES-based ones.

It turns out to be very difficult to avoid collision attacks. There are, for example, fairly generic pre-computation attacks that effectively perform a collision attack on the encryption key. As stopping all collision attacks is extremely difficult, if not impossible, the obvious solution is to move to 256-bit values.[2] The 256-bit key capability of AES makes it easy to achieve the 2^{128} effort bound.[3] Unfortunately, the 128-bit block size still supports collision attacks with a complexity of 2^{64} steps. Thus, careful design is necessary to achieve the full security required.

5.1 Collision attacks on CBC-MAC

The best known block cipher authentication mode is CBC-MAC. Pure CBC-MAC has many versions of collision attacks [BM01]. Even the addition of various padding rules or sequence numbers do not stop these attacks. For CBC-MAC the critical parameter is not the amount of data but the number of messages authenticated by the sender. CBC-MAC forgery is possible as soon as a collision on the MAC value has occurred. For typical message sizes this implies that CBC-MAC is marginally more secure than OCB, as the collision will occur earlier in OCB.

So far there seems to be little difference between OCB and CBC-MAC. But this is not a fair comparison. OCB is a combined encryption and authentication mode, and it uses a unique nonce. If we look at the authentication function of OCB by itself (without the encryption) it is laughably weak. It is only the combination of the authentication with the encryption that makes OCB secure. For a fair comparison we have to compare OCB with a combination of an encryption mode and an authentication mode.

[2]One of our general design rules: To force an attacker to spend 2^n steps in attacking your system, all cryptographic values should be at least $2n$ bits long. Given the prevalence of collision attacks this is just sound engineering.

[3]The use of 128-bit keys very often leads to more efficient attacks, and always requires careful analysis.

5.2 A suggested CTR-CBC-MAC mode

We give a rough specification of CTR-CBC-MAC: a combination of the CTR encryption mode with the CBC-MAC authentication mode. This is not a complete specification but only intended for comparison to OCB. It is based on [WHF02].

The message M is split into blocks $M_1, \ldots, M_m$, where the last block is padded with 0 bytes if it is not full-size. The nonce N for this mode is 8 bytes long.

First the CBC-MAC is computed by starting with the 128-bit value $V := N \parallel 0 \parallel \ell$ where ℓ is the length of M in bytes and $\parallel$ is the concatenation operator. The MAC is computed by setting $X_0 := V$ and defining $X_i := E_K(X_{i-1}) \oplus M_i$. The tag T is defined as the first τ bits of $E_K(X_m)$. The tag T is appended to the message M, and they are both encrypted together.

Encryption is done by generating a key stream using the blocks $S_i := E_K(N \parallel 1 \parallel i)$. The key stream bytes are xorred with the message and the tag. As this is a stream cipher no padding of the message is necessary; the padding is only used internally by the CBC-MAC part.

There is a proof of security for CTR-CBC-MAC that achieves similar bounds as OCB [Jon02]. As always this proof gives a lower bound on the security. We believe that the resistance to forgery is actually much higher.

All the attacks on CBC-MAC require that the attacker can detect when two MAC values are identical. Encrypting the MAC value in such a way that equal values can no longer be recognised as being equal stops the attacker from performing any of these attacks. As far as we are aware, this stops any of the 2^{64}-effort attacks and requires the attacker to perform 2^{128} operations before the first forgery. (See [JJV02] for a similar solution to this problem.) Just like in OCB, the addition of the encryption function has made the authentication much stronger.

5.3 Comparison

One of the main tools to prevent collision attacks is to limit the amount of data that is ever processed with a single key. For example, for CTR encryption the amount of data encrypted with each key should be limited to 2^{64} blocks, at which point the information-theoretical leakage of information about the plaintext is still below one bit. Limiting the amount of data is relatively easy to do as the implementation can enforce such limits. However,

# blocks data	limit on τ
1	128
2^{16}	96
2^{24}	80
2^{32}	64
2^{48}	32
2^{56}	16
2^{64}	0

Table 1: Limits on the authentication strength of OCB.

low limits can lead to significant complications as they require frequent re-keying or using multiple related keys for one large data item. While limits are necessary, we want them to be so large that they are not an issue for most situations. With current technology a limit on 2^{64} bytes of data is not a problem, but smaller limits can be. Current hard disks can store 2^{36} bytes or more, and we can only expect them to grow. Today, it is not at all unreasonable for someone to want to protect a backup of a database containing a Terabyte (2^{40} bytes) of data. For practical reasons this data might very well be split into several 'messages', but it would be highly desirable to use a single key to secure the whole database. In future databases will only get bigger. The CTR-CBC-MAC mode can be used for any amount of data up to 2^{64} blocks, or 2^{68} bytes.[4] Full authentication strength is maintained throughout.

This is where OCB differs. The collision attacks limit the effective authentication strength of OCB. The parameter τ in OCB allows a trade-off between message expansion and the security level of the authentication. For any τ, it is reasonable to expect that the probability of succeeding in forging a message is around $2^{-\tau}$. For OCB this only holds as long as the amount of information encrypted with a single key is less than $2^{64-\tau/2}$ blocks. Another way to phrase this is that if 2^n blocks of data are processed with OCB, then then τ is effectively limited to $128 - 2n$. The results are shown in table 1. This table can be interpreted in several ways. If 2^{24} blocks of data are to be processed, then τ is limited to 80. A larger τ does not improve the security. The converse is also true. If a design chooses $\tau = 64$ then it should also limit the number of blocks processed per key to 2^{32}. Allowing more data

[4]Users worried about the leakage of even a single bit of information can restrict themselves to a slightly lower bound, such as 2^{64} bytes, which all but eliminates the leakage.

to be processed reduces the authentication strength. The designer is only fooling himself with the value of τ, and could reduce the message expansion by choosing a τ compatible with the amount of data processed.

The choice $\tau = 64$ seems to be a popular one, but as the table shows it implies that only about 64 GB of data can be processed using a single key. This is a severe restriction for modern data-handling systems, certainly if they are still in use 10 or 20 years from now.

Our comparison is for the very simple case when we only consider the probability of successful forgery on the first attempt. If we look at the total amount of work the attacker has to perform the collision attack is slightly less attractive. If each key is used for 2^{32} blocks, the attacker needs to look for collisions for 2^{64} different keys before the first success, for a total work load of 2^{96}. (If each key is used for 2^{48} blocks the attacker's workload is reduced to 2^{80}.) This is well below our target of 2^{128}. Furthermore, the workload of the attacker is still comparable (if not less than) the workload of the user of the system. So if enough data is being processed by OCB to allow this attack, then the attack is practical by definition.

Things become even more difficult when we look at more complicated models where we consider the number of interactions with the sender or receiver that the attacker needs. One generic attack is just guessing the tag value. An attacker could try multiple times to guess a tag value, and each time have the same chance of succeeding. The collision attack on OCB does not allow repeated attempts. On the other hand, an attacker that guesses tag values will alert the receiver that an attack is going on, as receivers expect very few authentication failures in normal operation. The collision attack on OCB never alerts the receiver; there is never an indication of the attack until the receiver gets a forged message. One can argue which one of these attacks is more serious; a discussion that is unlikely to be solved in general terms as it depends heavily on the properties of the rest of the system. We feel that this is an irrelevant discussion as the security of a block cipher mode should not depend on the rest of the system. In other words: the block cipher mode should be secure irrespective of the design of the rest of the system, or the threat model. A mode that has weaknesses in any reasonable model should not be used at all, especially since the rest of the system, including the attack model, is likely to change over time.

If we restrict OCB to its safe operational envelope we get a 2^{-64} probability of message forgery and a limit on 2^{32} blocks of data. This is very similar to the bounds that CTR-CBC-MAC seems to achieve using 3DES. In that sense, OCB does not take advantage of AES, but takes us right back to the problems we had with 3DES.

Collision attacks are much easier when 64-bit block ciphers are used. Therefore, we most strongly advise never to use OCB with a 64-bit block cipher.

5.4 OCB's proof of security

Our attack does not violate the proof of security given in [RBBK01a]. The proof limits the advantage of the attacker to

$$\frac{1.5\,\bar{\sigma}^2}{2^{128}} + \frac{1}{2^\tau}$$

where $\bar{\sigma}$ is the total number of blocks encrypted under a single key plus a few other small terms. This bound clearly allows our attack.

6 Acknowledgements

I would like to thank Intersil for financing this research, and Russ Housley and Doug Whiting for their helpful comments. David Wagner and Tadayoshi Kohno provided valuable feedback on my first draft which greatly improved the presentation.

7 Conclusions

The OCB authenticated encryption mode has a much weaker authentication than could reasonably be expected. This mode is incapable of handling large amounts of data without losing authentication security. We therefore advice against the use of OCB.

References

[BM01] Karl Brincat and Chris J. Mitchell. New CBC-MAC forgery attacks. In V. Varadharajan and Y. Mu, editors, *Information Security and Privacy, ACISP 2001*, volume 2119 of *Lecture Notes in Computer Science*, pages 3–14. Springer-Verlag, 2001.

[JJV02] Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the security of randomized CBC-MAC, beyond the birthday paradox limit: a new construction. In *Fast Software Encryption 2002*, Lecture Notes in Computer Science. Springer-Verlag, 2002. To appear.

[Jon02] Jakob Jonsson. Personal communications, January 2002.

[LV01] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255–293, August 2001.

[NIS93] National Institute of Standards and Technology. *Data Encryption Standard (DES)*, December 30, 1993. FIPS PUB 46-2, available from `http://www.itl.nist.gov/fipspubs/`.

[NIS99] National Institute of Standards and Technology. *Data Encryption Standard (DES)*, 1999. DRAFT FIPS PUB 46-3, available from `http://csrc.ncsl.nist.gov/fips/`.

[RBBK01a] Philip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption, September 2001. Available from `http://www.cs.ucdavis.edu/~rogaway`.

[RBBK01b] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *Eighth ACM Conference on Computer and Communications Security (CCS-8)*, pages 196–205. ACM Press, 2001.

[WHF02] Doug Whiting, Russ Housley, and Niels Ferguson. AES encryption & authentication using CTR mode & CBC-MAC, January 2002. Working document from IEEE 802.11 TGi, doc # 02/001r0.