Security Flaws Induced by CBC Padding Applications to SSL, IPSEC, WTLS...

Serge Vaudenay

Swiss Federal Institute of Technology (EPFL)
Serge.Vaudenay@epfl.ch

Abstract In many standards, e.g. SSL/TLS, IPSEC, WTLS, messages are first pre-formatted, then encrypted in CBC mode with a block cipher. Decryption needs to check if the format is valid. Validity of the format is easily leaked from communication protocols in a chosen ciphertext attack since the receiver usually sends an acknowledgment or an error message. This is a side channel.
In this paper we show various ways to perform an efficient side channel attack. We discuss potential applications, extensions to other padding schemes and various ways to fix the problem.

1 Introduction

Variable input length encryption is traditionally constructed from a fixed input length encryption (namely a block cipher) in a special mode of operation. In RFC2040 [2], the RC5-CBC-PAD algorithm is proposed, based on RC5 which enables the encryption of blocks of $b = 8$ words where words are bytes. Encryption of any word sequence with an RC5 secret key K is performed as follows.

1. Pad the word sequence with n words, all being equal to n, such that $1 \leq n \leq b$ and the padded sequence has a length which is a multiple of b.
2. Write the padded word sequence as a block sequence $x_1, \ldots, x_N$ in which each block x_i consists of b words.
3. Encrypt the block sequence in CBC mode with a (either fixed or random or secret) IV with a permutation C defined by RC5 with key K: get

$$y_1 = C(\mathrm{IV} \oplus x_1), \quad y_i = C(y_{i-1} \oplus x_i); i = 2, \ldots, N \tag{1}$$

   where $\oplus$ denotes the XOR operation.

The encryption of the message is the block sequence $y_1, \ldots, y_N$.

Although decryption is not clearly defined in RFC2040 [2], it makes sense to assume that the receiver of an encrypted message first decrypts in CBC mode, then checks if the padding is correct and finally removes it. The question is: how must the receiver behave if the padding is not correct? Although the receiver should not tell the sender that the padding is not correct, it is meaningful that non-procession of a decrypted message ultimately leaks this bit of information.

This leads to an attack that uses an oracle for which any block sequence tells if the padding of the corresponding CBC-decrypted sequence is correct according to the above algorithm. The attack works within a complexity of $O(NbW)$ in order to decrypt the message where W is the number of possible words (typically $W = 256$).

A similar attack model was used by Bleichenbacher against PKCS#1 v1.5 [5] and by Manger against PKCS#1 v2.0 [13]. This paper shows that similar attacks are feasible in the symmetric key world.

The paper is organized as follows. We first recall some well known properties and security issues for the CBC mode. We describe several attacks against RC5-CBC-PAD and we introduce the notion of *bomb oracle*. We then discuss extensions to other schemes: ESP, random padding, ... and applications in real life such as SSL, IPSEC, WTLS, SSH2. Next we present some possible fixes which do not actually work like replacing the CBC mode by a double CBC mode, the HCBC mode or other modes which were proposed by the standard process run by NIST. We further propose a fix which does work.

2 CBC Properties

Several security properties of the CBC mode are already known. We think it is useful to recall them in order to remind ourselves of the intrinsic security limits of the CBC mode.

2.1 Efficiency

CBC mode is efficient in practice because we can encrypt or decrypt a stream of infinite length with a constant memory in linear time. Efficiency is comparable to the Electronic Code Book (ECB) mode where each block is encrypted in the same way. The difference between ECB and CBC is a single exclusive or operation. Since the ECB mode is not suitable in most applications because of ciphertext manipulation attacks, and lack of increased message entropy, we prefer to use CBC mode. (See e.g. [14, p. 230] for more details.)

Exhaustive search against CBC mode is related to the length of the secret key. We have yet other bounds related to the block length. First of all, the electronic code book attack has a complexity of W^b. We have other specific attacks related to the intrinsic security of the CBC mode no matter which block cipher is used. These are detailed in following sections.

2.2 Confidentiality Limits

Confidentiality has security flaws. Obviously, when using a fixed IV, one can easily see when two different messages have a common prefix block sequence by just looking at the two ciphertexts.

More generally, when two ciphertext blocks y_i and y_j are equal, one can deduce from Eq. (1) that $y_{i-1} \oplus y_{j-1} = x_i \oplus x_j$.[1] We can then exploit the redundancy in the plaintext in order to recover x_i and x_j from $y_{i-1} \oplus y_{j-1}$. This flaw is however quite negligible: since the ciphertext blocks get a distribution which is usually indistinguishable from a uniform distribution, the probability that two b-words blocks out of N are equal is given by the birthday paradox theorem

$$p \approx 1 - e^{-\frac{1}{2}N^2.W^{-b}}$$

where W is the number of possible words. The attack is efficient when N reaches the order of magnitude of $\sqrt{W^b}$. Therefore, for $b = 8$ and $W = 256$, we need about 2^{35} bytes (32GigaBytes) in order to get a probability of success equal to 39% for this attack which leaks information on 16 Bytes only.

2.3 Authentication Limits

The CBC mode can be used to create message authentication codes (MAC). Raw CBC-MAC (i.e. taking the last encrypted block as a MAC) is well known to have security flaws: with the MAC of three messages m_1, m_2, m_3 where m_2 consists of m_1 augmented with an extra block, we can forge the MAC of a fourth message which consists of m_3 augmented with an extra block. This is fixed by re-encrypting the raw CBC-MAC, but this new scheme still has attacks of complexity essentially $\sqrt{W^b}$. (See [15,16,19].)

3 The Attack

Let b be the block length in words, and W be the number of possible words. (We assume that $W \geq b$ and that all integers between 1 and b can unambiguously be encoded into words in order to make the CBC-PAD scheme feasible.)

We say that a block sequence $x_1, x_2, \ldots, x_N$ has a correct padding if the last block x_N ends with a word string of n words equal to n with $n > 0$: `1`, or `22`, or `333`, ... Given a block sequence $y_1, y_2, \ldots, y_N$, we define an oracle $\mathcal{O}$ which yields 1 if the decryption in CBC mode has a correct padding. Decryption is totally defined by a block encryption function C and IV. Oracle $\mathcal{O}$ is thus defined by C and IV.

3.1 Last Word Oracle

For any block y, we want to compute the last word of $C^{-1}(y)$. We call it the "last word oracle".

Let $r_1, \ldots, r_b$ be random words, and let $r = r_1 \ldots r_b$. We forge a fake ciphertext $r|y$ by concatenating the two blocks r and y. If $\mathcal{O}(r|y) = 1$, then $C^{-1}(y) \oplus r$ ends with a valid padding. In this case, the most likely valid padding is the one which ends with `1`. This means that the last word of $C^{-1}(y)$ is $r_b \oplus$ `1`. If

[1] This property was notably mentioned in [12, p. 43].

$\mathcal{O}(r|y) = 0$, we can try again (by making sure that we pick another r_b: picking the same one twice is not worthwhile).

If we are lucky (with probability W^{-1}), we find the last word with the first try. Otherwise we have to try many r_bs. On average, we have to try $W/2$ values.

Odd cases occur when the valid padding found is not `1`. This is easy to detect. The following program eventually halts with the last words of y: one in the typical case, several if we are lucky.

1. pick a few random words $r_1, \ldots, r_b$ and take $i = 0$
2. pick $r = r_1 \ldots r_{b-1}(r_b \oplus i)$
3. if $\mathcal{O}(r|y) = 0$ then increment i and go back to the previous step
4. replace r_b by $r_b \oplus i$
5. for $n = b$ down to 2 do
   (a) take $r = r_1 \ldots r_{b-n}(r_{b-n+1} \oplus 1)r_{b-n+2} \ldots r_b$
   (b) if $\mathcal{O}(r|y) = 0$ then stop and output $(r_{b-n+1} \oplus n) \ldots (r_b \oplus n)$
6. output $r_b \oplus$ `1`

3.2 Block Decryption Oracle

Now we want to implement an oracle which computes $C^{-1}(y)$ for any y: a "block decryption oracle".

Let $a = a_1 \ldots a_b$ be the word sequence of $C^{-1}(y)$. We can get a_b by using the last word oracle. Assuming that we already managed to get $a_j \ldots a_b$ for some $j \leq b$, the following program gets a_{j-1}, so that we can iterate until we recover the whole sequence.

1. take $r_k = a_k \oplus (b - j + 2)$ for $k = j, \ldots, b$
2. pick $r_1, \ldots, r_{j-1}$ at random and take $i = 0$
3. take $r = r_1 \ldots r_{j-2}(r_{j-1} \oplus i)r_j \ldots r_b$
4. if $\mathcal{O}(r|y) = 0$ then increment i and go back to the previous step
5. output $r_{j-1} \oplus i \oplus (b - j + 2)$

We need $W/2$ trials on average. We can thus recover an additional word within $W/2$ trials. Since there are b words per block, we need $bW/2$ trials on average in order to implement the C^{-1} oracle.

3.3 Decryption Oracle

Now we want to decrypt any message $y_1, \ldots, y_N$ with the help of $\mathcal{O}$. It can be done with $NbW/2$ 2-block oracle calls on average. We just have to call the block decryption oracle on each block y_i and perform the CBC decryption.

One problem remains in the case where IV is secret. Here we cannot decrypt the first block. We can however get the first plaintext block up to an unknown constant. In particular, if two messages are encrypted with the same IV, we can compute the XOR of the two first plaintext blocks.

The attack has a complexity of $O(NbW)$. As an example for $b = 8$ and $W = 256$ we obtain that we can decrypt any N-block ciphertext by making $1024N$ oracle calls on average. The attack is thus extremely efficient.

3.4 Postfix Equality Check Oracle

There are reasons which will be made clear for which we can be interested in *bomb oracles* as defined below. A bomb oracle is an oracle which either gives an answer or *explodes* depending on the input. Of course, the bomb oracle is no longer available after explosion. An attack which uses a bomb oracle fails if the oracle explodes. For instance, we are interested in a bomb oracle $\mathcal{O}'$ which either answers 1 when $\mathcal{O}$ answers 1 or explodes when $\mathcal{O}$ answers 0.

Given a ciphertext $y_1, \ldots, y_N$ and a word sequence $w_1 \ldots w_m$, we want to implement a bomb oracle which checks if $w_1 \ldots w_m$ is a postfix of the decryption of $y_1, \ldots, y_N$ by using $\mathcal{O}'$. Let us first consider that $m \leq b$. We perform the following process.

1. pick a few random words $r_1 \ldots r_{b-m}$
2. take $r_{b-m+k} = w_k \oplus m$ for $k = 1, \ldots, m$
3. send $r|y_N$ to the oracle $\mathcal{O}'$ where $r = r_1 \ldots r_b$
4. if $m = 1$ then
   - take $r'_k = r_k$ for $k = 1, \ldots, b-2, b$ and take $r'_{b-1} = r_{b-1} \oplus \mathtt{1}$

   otherwise
   - take $r'_k = r_k$ for $k = 1, \ldots, b-1$ and take $r'_b = w_m \oplus \mathtt{1}$
5. send $r'|y_N$ to the oracle $\mathcal{O}'$ where $r' = r'_1 \ldots r'_b$
6. output 1

The second oracle call is used in order to eliminate odd cases which are not eliminated by the first one, for instance when $w_m \oplus m \oplus 1$ is a postfix. Obviously, this is a bomb oracle which checks whether $w_1 \ldots w_m$ is a postfix or not.

For $m > b$, we can cut the ciphertext and use the above oracle $\lceil \frac{m}{b} \rceil$ times on each block. As will be noticed, some CBC-PAD variants allow to have paddings longer than b (namely at most $W-1$), so we can generalize the previous oracle and check postfixes within a single $\mathcal{O}$ oracle call. This will be used against SSL/TLS in Section 5.1.

4 Other Padding Schemes

In Schneier [17, pp. 190–191], a slightly different padding scheme is proposed: only the last word is equal to the padding length, and all other padded words are equal to zero. The padded sequence is thus $00\ldots0n$ instead of $nn\ldots n$. Obviously, a similar attack holds.

IP Encapsulating Security Payload (ESP) [10] uses another slightly different padding: the padded sequence is $\mathtt{1234}\ldots n$ instead of $nn\ldots n$. Obviously, a similar attack holds.

Another padding scheme consists of padding with a non blank word then the necessary number of blank words. This is suggested, for instance by NIST [8, App. A] with $W = 2$ (here the blank word is the bit 0). Obviously, a similar attack holds.

One can propose to have the last word equal to the padding length and all other padded words chosen at random (like SSH2). The attack still enables the decryption of the last word of any block. We also have another security flaw: if the same message is encrypted twice, it is unlikely that the last encrypted blocks are equal, but in the case where the padding is of length one. We can thus guess the padding length when the ciphertexts are equal.

5 The Attack in Real Life

Here we discuss various applications. In most of cases, the attack can be (and is) avoided by using appropriate parameters. However, since this is not carefully specified in the standards, our aim is to warn the users about possible bad configurations.

5.1 SSL/TLS

Like in SSL, TLS v1.0 [7] uses the CBC-PAD scheme with $W = 256$ when using block ciphers (default cipher being the RC4 stream cipher though). The only difference is that the padding length is not necessarily less than b but can be longer (but less than $W - 1$) in order to hide the real length of the plaintext. We can thus expect to use a TLS server like the $\mathcal{O}$ oracle.

TLS v1.0 also provides an optional MAC which failed to thwart the attack: when the server figures out that the MAC is wrong, it yields the `bad_record_mac` error. However, the message padding is performed *after* the MAC algorithm, so the MAC does not preclude our attack since it cannot be checked before the padding in the decryption. The situation is a little different in SSL v3.0 since both wrong MAC both invalid padding return the same error. However, the question whether the client can distinguish the two types of error is debatable.

The reason why the attack is not so practical is because the padding format error (the `decryption_failed` error) is a fatal alert and the session must abort. The server thus stops (or "explodes") as soon as the oracle outputs 0. For this reason we consider the bomb oracle $\mathcal{O}'$. We can thus perform the postfix equality check oracle described in Section 3.4. It can be used in order to decrypt by random trial the last word of a block with a probability of success of W^{-1}, the last two words of a block with a probability of success of W^{-2}, ...

Interestingly, TLS wants to hide the real message length itself. We can easily frustrate this feature by implementing a "length equality check bomb oracle" in a very same way: if we want to check whether or not the padding length is equal to n, we take the last ciphertext block y, and we send $r|y$ to the server where the rightmost word of r is set to $n \oplus 1$ and the others are random. Acceptance by $\mathcal{O}'$ means that the right length is n with probability at least $1 - W^{-1}$. Rejection means that n is not the right length for sure.

Since the padding length is between 1 and W, the above oracle may not look so useful. We can still implement another bomb oracle which answers whether or not the padding length is greater than b, i.e. if the length hiding feature of TLS was used: let y_1 and y_2 be the last two ciphertext blocks. We just send $r|y_1|y_2$ with a random block r to $\mathcal{O}'$. Acceptance means that the padding length is at most b with probability at least $1 - W^{-1}$. Rejection means that the padding length is at least $b + 1$ for sure.

5.2 IPSEC

IPSEC [9] can use CBC-PAD. Default padding scheme is similar, as specified in ESP [10]. Standards clearly mention that the padding should be checked, but the standard behavior in the case of invalid padding is quite strange: the server just discards the invalid message and adds a notification in log files for audit and nothing else. This simply means that errors are processed according to non standard rules or by another protocol layer. It is reasonable to assume that the lack of activity of the receiver in this case, or the activity of the auditor, can be converted into one bit of information. So our attack may be applicable.

IPSEC provides an optional authentication mechanism which could protect against our attack, provided that the authentication check is performed *before* the format check of the plaintext. Although used in most of practical applications, this mechanism still has an optional status in IPSEC. As already recommended by Bellovin [4], authentication should be mandatory. Bellovin actually used a side channel which tells the validity of the TCP checksum. His attack was recently extended to the WEP protocol for 802.11 wireless LANs by Borisov et al. [6].

5.3 WTLS

WTLS [1] (which is the SSL variant for WAP) perfectly implements the oracle $\mathcal{O}$ by sending `decryption_failed` warnings in clear. Actually since mobile telephones have a limited power and CPU resources, key establishment protocols with public key cryptography are limited. So we try to limit the number of session initializations and to avoid breaking them. So seldom errors are fatal alerts. Some implementations of WTLS can however limit the tolerance number of errors within the same session, which can limit the efficiency of the attack. This is however non standard.

In the case of mobile telephones (which is the main application of WTLS), WTLS is usually encapsulated in other protocols which may provide their own encryption protocol, for instance GSM. In this case, the extra encryption layer needs to be bypassed by the attacker.

5.4 SSH2

In SSH2, the MAC is optional. When not used, our attack is feasible, but only recovers one word since the padding is mostly random. When used, the MAC is computed on the padded message. Therefore, it is checked before the padding format, which protects against our attack.

6 Fixes which Do not Work

6.1 Padding Before the Message

One can propose to put the padding in the first block. This only works for CBC modes in which IV is not sent in clear with the ciphertext (otherwise the same attack holds). This also requires to know the total length (modulo b) of the message that we want to encrypt before starting the encryption. When the plaintext is a word stream, this assumption is not usually satisfied. Therefore we believe that this fix is not satisfactory.

6.2 CBCCBC Mode

Another possibility consists of replacing the CBC mode by a double CBC encryption (i.e. by re-encrypting the $y_1, \ldots, y_N$ sequence in CBC mode). We call it the CBCCBC mode.

Unfortunately, a similar attack holds: given y and z we can recover the value of $u = C^{-1}(y) \oplus C^{-1}(y \oplus C^{-1}(z))$ by sending $r|y|z$ trials to the oracle. This is enough in order to decrypt messages: if y is the $(i-1)$th ciphertext block, z is the ith ciphertext block, and if t is the $(i-2)$th ciphertext block, then the ith plaintext block is nothing but $t \oplus u$!

The same attack holds with a triple CBC mode...

6.3 On-Line Ciphers and HCBC Mode

We can look for another mode of operation which "provably" leaks no information. One should however try to keep the advantages of the CBC mode: being able to encrypt a stream without knowing the total length, without having to keep an expanding memory, ... In [3], Bellare et al. presented the notion of on-line cipher. This notion is well adapted for these advantages of the CBC mode.

They also proposed the HCBC mode as a secure on-line cipher against chosen plaintext attacks. The idea consists in replacing Eq. (1) by

$$y_i = C(H(y_{i-1}) \oplus x_i)$$

where H is a XOR-universal hash function which includes part of the secret key. For instance one can propose $H(x) = K_1 x$ in $\mathrm{GF}(W^b)$ where $K_1 \neq 0$ is part of the secret key. (For any fixed a, b, c with $a \neq b$, we have $\Pr[H(a) \oplus H(b) = c] \leq 1/(W^b - 1)$ if K_1 is uniformly distributed, thus H is XOR-universal.)

One problem is that this does not protect against the kind of attack we proposed. For instance we notice that if we get several accepted $r_i|y$ messages with a fixed y, then we deduce that $H(r_i) \oplus x$ ends with a valid padding for an unknown but fixed x. Hence $H(r_i) \oplus H(r_j)$ is likely to end with the word zero. Since this is the last word of $K_1(r_i \oplus r_j)$, we deduce K_1 from several (i, j) pairs. With the knowledge of K_1 we then adapt the attack against the raw CBC. It is even more dramatic here since we indeed recover a part of the secret key.

We outline that with the particular choice of XOR-universal hash function, the claimed security result collapses. Of course, there is no contradiction with the security result since our attack gets extra information from the side channel oracle $\mathcal{O}$, which was not allowed in the security model of [3]: the notion of on-line cipher resistant against chosen plaintext attacks does not capture security against the kind of cryptanalysis that we have proposed.

6.4 Other Modes of Operation

The first stage of the standardization process on modes of operation launched by NIST also contained problematic proposals.[2] Several of the proposals could be generalized as follows. The CBC mode is modified in order to have a XOR before and after the block cipher encryption, depending on all previous ciphertext blocks and all previous plaintext blocks. We replace Eq. (1) by

$$y_i = C(x_i \oplus f_i(x, y)) \oplus g_i(x, y)$$

with public $f_i(x, y)$ and $g_i(x, y)$ functions which only depend on i and all x_j and y_j for $j = 1, \ldots, i-1$. (Note that HCBC is not an example since f_i is not public.)

Assuming that an attacker knows several (x^j, y^j) plaintext-ciphertext pairs written $x^j = x_1^j|\ldots|x_{\ell_j}^j$ and $y^j = y_1^j|\ldots|y_{\ell_j}^j$, and she wants to compute $C^{-1}(y)$ for some given y, she can submit some $y_1^j|\ldots|y_k^j|(y \oplus \delta)$ ciphertexts where $k \leq \ell_j$, $\delta = g_{k+1}(x^j, y^j)$. Acceptance would mean that the block $C^{-1}(y) \oplus f_{k+1}(x^j, y^j)$ ends with a valid padding. Therefore we can decrypt the rightmost word with W samples, two words with W^2 samples, ...

6.5 CBC-PAD with Integrity Check

One can propose to add a cryptographic checkable redundancy code (crypto-CRC) of the whole padded message (like a hashed value) in the plaintext and encrypt

$$\text{message}|\text{padding}|h(\text{message}|\text{padding}).$$

This way, any forged ciphertext will have a negligible probability to be accepted as a valid ciphertext. Basically, attackers are no longer able to forge valid ciphertexts, so the scheme is virtually resistant against chosen ciphertext attacks.

Obviously it is important to pad before hashing: padding after hashing would lead to the a similar attack. The right enciphering sequence is thus

pad, hash, encrypt

Conversely, the right deciphering sequence consists of decrypting, checking the hashed value, then checking the padding value. Invalid hashed value must abort the decipherment.

[2] See http://csrc.nist.gov/encryption/modes/

There is still a nice security flaw discovered by David Wagner for this scheme in a subtle attack model.[3] We perform a "semi-chosen plaintext attack": we assume that we can convince the sender to send a message consisting of an unknown x (of known length) concatenated by a chosen postfix, and the goal is to get information on x. We can implement a guess check oracle: if g is a guess for x, we ask the sender to concatenate x with $y|h(g|y)$ where y is such that $g|y$ is a valid padded message. The sender then pads $x|y|h(g|y)$ (with a constant block $bb\ldots b$), appends a message digest, and encrypts the whole sequence in CBC mode. The attacker can then truncate the ciphertext after the $h(g|y)$. If the receiver accepts the truncated message, it means that the guess was right!

7 A Fix which May Work

The author of this paper first thought that using authentication in the CBC-PAD in order to thwart the attacks was an overkill. Wagner's attack demonstrates that it actually is not. We thus propose to replace the CBC encryption by a scheme which simultaneously provides authentication and confidentiality. As having padding in between authentication and encryption (as is done in TLS) is not a fortunate idea, the authentication-encryption scheme *must* apply on padded plaintexts:

1. take the cleartext
2. pad the message and take the padded message x
3. authenticate and encrypt x
4. transmit the result y

Similarly, the authentication check and decryption *must* be performed *before* the padding check:

1. decrypt and check the authenticity of y
2. take the plaintext x
3. check the padding of x
4. extract the padding and get the cleartext

(Note: we used the subtle difference between "cleartext" and "plaintext" as specified in RFC 2828 [18]: the cleartext is the original message in clear, and the plaintext is the input of the encryption process.)

The question whether authentication must be done before encryption or not is another problem. As an example, Krawczyk [11] recently demonstrated the security of the authenticate-then-CBC-encrypt scheme. We must however be careful about the meaning of this security result: in this proof, attackers are not assumed to have access to side channel oracles like in our model. Therefore the security result may collapse when using an appropriate oracle as for the HCBC mode. Therefore it is not quite clear how this result extends in a model where we can have side channel. We leave this as an open problem.

Despite the lack of formal security result, we believe that this scheme offers the required security.

[3] Private communication from David Wagner.

8 Conclusion

We have shown that several popular padding schemes which are used in order to transform block ciphers into variable-input-length encryption schemes introduce an important security flaw. Correctness of the plaintext format is indeed a hard core bit which easily leaks out from the communication protocol.

It confirms that security analysis must not be limited to the block cipher but must rather be considered within the whole environment: as was raised by Bellovin [4] and Borisov et al. [6], we can really have insecure standards which use unbroken cryptographic primitives. This was already well known in the public key cryptography world. We have demonstrated that the situation of symmetric cryptography is virtually the same.

Acknowledgments

I would like to thank Pascal Junod for helpful discussions. I also thank my students from EPFL for having suffered on this attack as an examination topic. After this attack was released at the Rump session of CRYPTO'01, several people provided valuable feedback. I would in particular like to thank Bodo Möller, Alain Hiltgen, Wenbo Mao, Ulrich Kühn, Tom St Denis, David Wagner, and Martin Hirt. I also thank the anonymous referees for their extensive and pertinent comments.

References

1. Wireless Transport Layer Security. Wireless Application Protocol WAP-261-WTLS-20010406-a. Wireless Application Protocol Forum, 2001. `http://www.wapforum.org/`
2. R. Baldwin, R. Rivest. The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms RFC 2040, 1996.
3. M. Bellare, A. Boldyreva, L. Knudsen, C Namprempre. Online Ciphers and the Hash-CBC Construction. In *Advances in Cryptology CRYPTO'01*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 2139, pp. 292–309, Springer-Verlag, 2001.
4. S. Bellovin. Problem Areas for the IP Security Protocols. In *Proceedings of the 6th Usenix UNIX Security Symposium*, San Jose, California, USENIX, 1996.
5. D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1. In *Advances in Cryptology CRYPTO'98*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 1462, pp. 1–12, Springer-Verlag, 1998.
6. N. Borisov, I. Goldberg, D. Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, ACM Press, 2001.
7. T. Dierks, C. Allen. The TLS Protocol Version 1.0. RFC 2246, standard tracks, the Internet Society, 1999.
8. M. Dworkin. Recommendation for Block Cipher Modes of Operation. US Department of Commerce, NIST Special Publication 800-38A, 2001.

9. S. Kent, R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, standard tracks, the Internet Society, 1998.
10. S. Kent, R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406, standard tracks, the Internet Society, 1998.
11. H. Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure is SSL?). In *Advances in Cryptology CRYPTO'01*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 2139, pp. 310–331, Springer-Verlag, 2001.
12. L.R. Knudsen. *Block Ciphers — Analysis, Design and Applications*, Aarhus University, 1994.
13. J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS#1 v2.0. In *Advances in Cryptology CRYPTO'01*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 2139, pp. 230–238, Springer-Verlag, 2001.
14. A.J. Menezes, P.C. van Oorschot, S.A. Vanston. *Handbook of Applied Cryptography*, CRC, 1997.
15. E. Petrank, C. Rackoff. CBC MAC for Real-Time Data Sources. *Journal of Cryptology*, vol. 13, pp. 315–338, 2000.
16. B. Preneel, P. C. van Oorschot. Mdx-MAC and Building Fast MACs from Hash Functions. In *Advances in Cryptology CRYPTO'95*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 963, pp. 1–14, Springer-Verlag, 1995.
17. B. Schneier. *Applied Cryptography*, 2nd Edition, John Wiley & Sons, 1996.
18. R. Shirey. Internet Security Glossary. RFC 2828, the Internet Society, 2000.
19. S. Vaudenay. Decorrelation over Infinite Domains: the Encrypted CBC-MAC Case. In *Selected Areas in Cryptography'00*, Waterloo, Ontario, Canada, Lectures Notes in Computer Science 2012, pp. 189–201, Springer-Verlag, 2001. Journal version: *Communications in Information and Systems*, vol. 1, pp. 75–85, 2001.