Specifications Overview for Counter Mode of Operation. Security Aspects in Case of Faults

R. Tirtea and G. Deconinck
K. U. Leuven/ESAT, Leuven, Belgium

***Abstract*—In 2001, after a selection process, NIST added the Counter Mode of operation to be used with the Advanced Encryption Standard (AES) [1]. In the NIST recommendation [1] a Standard Incrementing Function is defined for generation of the counter blocks which are encrypted for each plaintext block. IPsec Internet Draft [2] and ATM Security Specifications [3] contain implementation specifications for Counter Mode Standard Incrementing Function. In this paper we present those specifications. We analyze the probability to reveal useful information in case of faults in Standard Incrementing Function described in NIST recommendation. The confidentiality of the mode can be compromised with the fault model presented in this paper. We recommend another solution to be used in generation of the Standard Incrementing Function in the context of the Counter Mode.**

I. INTRODUCTION

One of the conditions for a secure encryption requires that the plaintext contains no pattern, as these will leak to the ciphertext. For this condition to be satisfied independent of the plaintext to be encrypted, the ciphers are used in a special way, called modes of operation. In 1980 four modes of operation were standardized in [4]:

- ECB (Electronic Code Block) mode;
- CBC (Cipher Block Chaining) mode;
- CFB (Cipher FeedBack) mode;
- OFB (Output FeedBack) mode.

In 2001 the U.S. National Institute of Standards and Technology (NIST) added a fifth mode, called Counter Mode (CTR), all of them recommended as modes of operation to be used with AES [1]. The Counter Mode has efficiency advantages over the previous modes of operation (ECB, CBC, CFB, and OFB) [5]. Used with large block size ciphers as AES, CTR mode is not weakening the security of the algorithm.

The paper contains, beside the NIST Recommendations, two examples of using Counter Mode: one in an Internet Draft regarding IPsec [2] and a second on the ATM Security Specifications [3].

In this paper we analyze the effect of faults in certain implementations of the CTR mode. Attacks based on random faults were announced by Bohen, DeMillo and Lipton [6] in 1997. Their attack was applicable only to public key cryptosystems, but, following year, starting from this work, Eli Biham and Adi Shamir proposed new attacks, based on transient faults and permanent faults targeting also secret key cryptosystems [7].

Faults can reduce the overall confidentiality of the cryptosystem. We noticed that in case of the Counter Mode, if faults are affecting one or more bits of the encrypting sequence (called counter blocks in CTR mode) then, consecutive plaintext blocks are XOR-ed with the same counter block to generate the ciphertext blocks and this, independent of the key value.

II. COUNTER MODE

Diffie and Hellman introduced Counter Mode in 1979 [8]. Different institutions or consortiums such as NIST [1] or the ATM Forum [3] standardized Counter Mode. The advantages [5] of this mode compared with others are:

- high speed implementations. CTR is fully parallelizable; Also pre-processing can be used to increase speed;
- the low rate of error propagation;
- arbitrary length of the messages;

all those without weakening the security.

The encryption and decryption processes using counter mode of operation are presented in Fig. 1. We use the following notation:

- n - number of bits of the encryption/decryption block,
- l - length of a message encrypted with the same key K,
- m is l/n rounded up to the nearest integer,
- u - value smaller than n so that

$$l=n*(m-1)+u$$

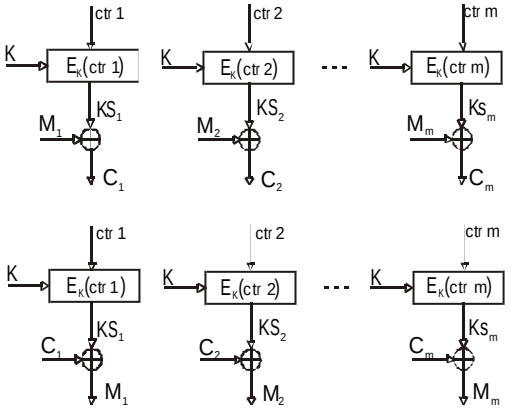

Figure 1. Counter mode. Encryption and decryption.

A set of input blocks (of length n), called *counter blocks* ($ctr\ 1, \ldots, ctr\ m$), are encrypted using the *key K* to produce a sequence of output blocks, called *key stream blocks* ($KS_1, \ldots, KS_m$), which are XOR-ed with the *plaintext blocks* ($M_1, \ldots, M_m$) to produce the *ciphertext blocks* ($C_1, \ldots, C_m$). With $E_K(ctr\ j)$ is denoted encryption of

the counter block *ctr j*, with the given key *K*. For decryption, the ciphertext is XOR-ed with the key stream to produce the plaintext. In this way the same function is used for encryption and decryption process, only the inputs are different. This represents an advantage for hardware implementation - the same hardware can be used for encryption and for decryption even if the algorithm is not identical for both operation (encryption/decryption) - such as in Rijndael (the AES selected standard) [9], [10].

Equations (1) and (2) contain the formulas for encryption and decryption respectively, where E_K represents encryption function with the key *K*.

$$KS_j = E_K(ctr\ j), \quad for\ j=1, 2, \ldots m;$$
$$C_j = M_j \text{ Å } KS_j, \quad for\ j=1, 2, \ldots m-1;$$
$$C_m = M_m \text{ Å } MSB_u(KS_m). \qquad (1)$$

$$KS_j = E_K(ctr\ j), \quad for\ j=1, 2, \ldots m;$$
$$M_j = C_j \text{ Å } KS_j, \quad for\ j=1, 2, \ldots m-1;$$
$$M_m = C_m \text{ Å } MSB_u(KS_m). \qquad (2)$$

If the last block of the message encrypted with the same key has a length *u* smaller than the block size *n*, then only the first *u* bits are XOR-ed with the first *u* bits of the key stream KS_m, the rest of them being discarded.

From Fig. 1 and from the equations can be noticed that the encryption function can be executed for the counters before the plaintext is available for producing the ciphertext.

The sequence of counter blocks must have the property that each block is different from others while the same given key is used. If this requirement is not satisfied, then, the confidentiality of all the plaintext blocks encrypted with the same counter block may be compromised [5].

So, the security of the encryption can be reduced in case that more plaintext blocks are encrypted with the same counter block.

A. Standard Incrementing Function in NIST Recommnendations

The function used for generation of the counter blocks has to satisfy the uniqueness requirements, meaning that for the same key, all counter blocks should be different.

Starting from an initial counter block *ctr1*, the successive counter blocks are derived by applying an *incrementing function*.

In [1] this is called *Standard Incrementing Function*. The Standard Incrementing Function can be applied to entire block or to part of a block. If *p* is the number of bits in the part to be incremented $(p < n)$, and $x < 2^p$ a positive integer, then the function takes $[x]_p$ (the binary representation of the last *p* bits of the integer *x*) and returns $[x+1 \bmod 2^p]_p$.

An example with small values of $p=5$ and $n=8$ is given in [1]. The symbol * represents an unknown bit in the example, and ***11110* is the initial value, which is incremented to generate the rest of the counter blocks. After four application of the Incrementing Function the output is the following:

** * * 1 1 1 1 0*
** * * 1 1 1 1 1*
** * * 0 0 0 0 0*
** * * 0 0 0 0 1*
** * * 0 0 0 1 0.*

This function satisfies the uniqueness requirements in case of $m <= 2^p$ blocks encrypted with the same key. The recommendations are not restrictive. There is also mentioned that in case of a non–zero initial string, a linear feedback shift register can be used.

In the next sections there are presented two modes for implementation of the Incrementing Function for the counter block. The first one is using 128 bits blocks and the AES encryption algorithm in an Internet Draft concerning IPsec [2] and the second one, is the one used in the ATM Security Specifications [3].

B. Counter Mode and IPsec

In [2] an Internet Draft of Internet Engineering Task Force is presented, and it describes the use of AES Counter Mode of operation (called here AES-CTR). It contains also the explicit initialisation vector as an IPsec Encapsulating Security Payload (ESP) confidentiality mechanism.

It also shows the need for a unique combination of initial value and key, and the requirement that the same counter block is not repeated during the use of a key.

Counter Block Format. The counter block contains 128 bits. The components of the counter block are as follows:

- *nonce*, a single use value field of 32 bits. It is assigned at the beginning of the security association;
- *initial vector*, a field of 64 bits, chosen only once for a given key;
- *block counter*, the last 32 bits of the counter block, starts with a value of one and is incremented to generate the next counter blocks.

The *block counter* field starts with the value of one and is incremented to generate the subsequent field of the counter block.

This assures $2^{32}–1$ distinct counter blocks, or 4,294,967,295 blocks, which is considered to be sufficient to handle IPv6 requirements.

The Internet Draft [2] contains also 9 test vectors. The test vectors contains maximum the first 3 consecutive counter blocks.

Every time when a security association is established or a key is changed (meaning new nonce or initial vector are established between parties) the initial value for the least significant 32 bits is set to 1 and then it is incremented till a new association or key is established.

C. Counter Mode in ATM Security Specifications

ATM Security Specifications [3] presents the utilization model for the counter mode with any 64-bit block encryption algorithm. Version 1 of the specification was published in 1999 and Version 1.1 in 2001. The part relevant for our paper has no major changes.

In [3] the counter mode is considered the most efficient mode of operation in ATM encryption due to the parallel encryption capabilities.

State Vectors Fields. The counter blocks are called in [3] *State Vectors* (*SVs*). In order to ensure unique key stream value for each block that is encrypted with the same key, each State Vector contains fields with various counters and a Linear Recurring Sequence.

The State Vector has 64 bits belonging to five fields. The fields of the State Vector are as follows:

- *Galois LFSR*, 21 bits;
- *Initiator/Response (I/R)* bit;
- *Sequence Number*, 4 bits;
- *Segment Number*, 3 bits;
- *Jump Number*, 35 bits.

The first field is composed of a *Galois Linear Feedback Shift Register* (LFSR). The *Galois* implementation of the shift register is used. The LFSR is pre-set back to initial value at each resynchronization of the communication pairs. The maximum interval of time between resynchronizations determinates the selection of a 21-bit size LFSR, generating 2^{21}-1 values, meaning 2,097,151 distinct blocks.

The *I/R bit* is used to avoid the same cipher text to be used with the original message and with the response message in case of the same key and SV used in duplex connections. This determines that the responder's key stream to be different, so enclosing the original plaintext would produce different ciphertext.

The *Sequence Number* bits are set to different values depending on the context of use (AAL1 connections, AAL3/4 connections or other connections).

The *Segment Number* is a 3-bit field that defines which 64-bit segment within the payload is encrypted/decrypted. (The 384-bit ATM cell payload is segmented into 6 segments of 64 bits for encryption and decryption). The LFSR is constant for the entire cell payload.

The *Jump Number* starts from all zeros and it is incremented each time a resynchronization occurs or in case of AAL5 with each end-of-message cell.

The 35-bit field allows 2^{35}-1 resynchronizations without repetition and is incremented as binary counter.

From all 64 bits of State Vector, only the Jump number requires to be transmitted to the receiver during a resynchronization or key changeover. The other fields are preset to their default values.

The generation of the next counter block between resynchronizations is mainly based on the output of the LFSR. Even if at the resynchronization the LFSR has the same initial value, the Jump number (incremented at every resynchronization) gives the differentiation. The specifications contain also a requirement. The jump number should be always greater than the previous jump number. If the new Jump number is less or equal to the previous Jump number, then this is rejected and considered as an error condition.

III. SECURITY ASPECTS IN CASE OF FAULTS

As mentioned in section II, the main concern regarding the generation of the counter is to have a unique value for all messages encrypted with the same key. In this section, the security of the Standard Incrementing Function presented in section II is analyzed.

A. General Security Considerations

Based on the time and cost involved in breaking the cipher text and finding its key, encryption can be divided into two categories:

- unconditionally secure;
- computationally secure.

An encryption scheme is unconditionally secure if the cipher text generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available.

An encryption scheme is said to be computationally secure if the following criteria are met:

- the cost of breaking the cipher exceeds the value of the encrypted information.
- the time required to break the cipher exceeds the useful lifetime of the information.

The new encryption algorithms are computationally secure, and, usually they are designed to resist to all known attacks. Strength against known attacks is mentioned in the description of most encryption algorithms such as AES (Rijndael) [9] or Camellia [11]. An overview of the proposed attacks for Rijndael algorithm is presented in [12].

B. Attacks based on Faults in Implementation

In 1996, Boneh, Demillo and Lipton [6] presented a theoretical model for breaking various cryptographic schemes by taking advantage of random hardware faults.

The model consists of a black-box containing some cryptographic secret [13]. The box interacts with the outside world by following a cryptographic protocol. The model supposes that from time to time the box is affected by a random hardware fault causing it to output incorrect values. For example, the hardware fault flips an internal register bit at some point during the computation. In [14] was shown that for many digital signatures and identification schemes these incorrect outputs completely expose the secrets stored in the box.

This attack was considered to be applicable to public key cryptosystems and not to secret-key algorithms [7].

Biham and Shamir propose a related attack called *Differential Fault Analysis* (DFA) in [7]. They showed in [7] that DFA was applicable to almost any secret key cryptosystem proposed in the open literature at that moment.

The main criticism against DFA was that the transient fault model that was claimed to be unrealistic. Starting from this, Biham and Shamir decided to develop a more practical fault model based on permanent hardware faults. They showed that their model could be used to break Data Encryption Standard (DES) [7]. They called this *Non-Differential Fault Analysis* (NDFA). For this attack they proposed the cut of a wire or permanent destroy of a single memory cell.

This paper uses the assumptions of faults in case of the Counter Mode of operation. In case of *hardware implementation*, a permanent hardware fault affecting a certain bit (the most vulnerable bit of the counter blocks) can reduce the confidentiality of the mode (more explanations in the next section).

C. Faults in Case of Counter Mode

In context of secure encryption algorithms such as AES, the operation modes should not affect the overall cipher security.

In this section we analyze the Standard Incrementing Function example already mentioned in section II.

We consider the following sequence (adding one more counter block to the initial sequence from section IIA):

*ctr 1 = * … * 1 1 1 1 0*
*ctr 2 = * … * 1 1 1 1 1*

*ctr 3 = * … * 0 0 0 0 0*
*ctr 4 = * … * 0 0 0 0 1*
*ctr 5 = * … * 0 0 0 1 0*
*ctr 6 = * … * 0 0 0 1 1*

It is easy to notice that the Hamming distance between each two pairs of counters *(ctr1, ctr2), (ctr3, ctr4), (ctr5, ctr6)* etc. is 1. This is true for all pairs of (*even, odd*) counter block values.

In this example we consider that the Standard Incrementing Function is applied to the least significant bits of the initial value counter (as it is also in [1] and [2]). If the positioning is different, the example is valid also, but we call the least significant bit – the one situated on the least significant position of segment affected by the incrementing function.

If a fault occur on the least significant position of the counter, then we have the following sequence:

*ctr 1 = * … * 1 1 1 1 f*
*ctr 2 = * … * 1 1 1 1 f*
*ctr 3 = * … * 0 0 0 0 f*
*ctr 4 = * … * 0 0 0 0 f*
*ctr 5 = * … * 0 0 0 1 f*
*ctr 6 = * … * 0 0 0 1 f*

where *f* can be 0 or 1. Which is the value of *f* is not important, as long as it is the same for consecutive *(ctr j, ctr j+1)* pairs of block counter values (with *j* being odd, and the value of *ctr j* in this context being even). In this case the encryption process is generating the following sequence of ciphertext blocks:

$KS_j=E_K(ctr\ j),\ KS_{j+1}=E_K(ctr\ j+1),\ \ldots$
$M_j \oplus KS_j,\ M_{j+1} \oplus KS_{j+1},\ M_{j+2} \oplus KS_{j+2},\ M_{j+3} \oplus KS_{j+3},\ldots$

with *ctr j=ctr j+1* and further $KS_j=KS_{j+1}$ due to the fault *f*. So, a XOR operation between two consecutive ciphertext blocks will give:

$C_j \oplus C_{j+1} = M_j \oplus M_{j+1}$, with *j* odd;

independent of the key *K*.

This is not revealing directly the plaintext, but if patterns exist, the plaintext could be extracted. And, as mentioned in section 2, the uniqueness requirement of the Incrementing Function generating the counter blocks is not fulfilled any more.

In hardware implementation this fault can be generated cutting the wire connection of the LSB of the Counter Module to the Encryption Module in Fig. 2. The same effect can be obtained due to trap implementation.

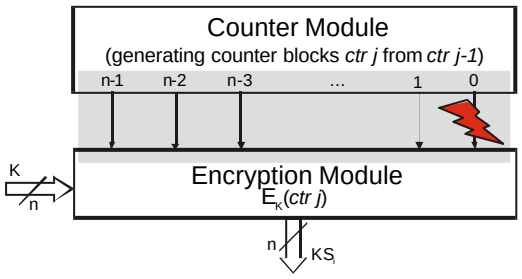

Figure 2. Fault on the LS the Counter module output (n =the size of the encryption/decryption block).

The fault assumption can be extended. If the two least significant bits are faulty, then four consecutive plaintext blocks are XOR-ed with the same key stream block, independent of the key.

We consider that a pair sender/receiver are communicating encrypted data using CTR mode. From the fault model presented before, there are different situations:

- no fault present;
- fault(s) present at one side. In this case the receiver will have one of the following situations after the decryption process:
  – every second block of decrypted data is unintelligible – due to a single fault at the least significant position of the counter segment;
  – every plaintext block is followed by three unintelligible blocks – due to a number of two faults, etc;
- same fault(s) present on both sides. In this case the presence of fault(s) is undetected;
- different faults present at the two sides. At the receiver side, certain (or all) blocks are unintelligible.

From those situations, the most dangerous one is the case of the same fault(s) present at both sides. This can be due to a malicious implementation of the encryption mode of operation if all the parties involved in the secure communication are using the same corrupted implementation.

D. Our Fault Model in Case of Other Operation Modes

In this section we analyze the remaining four modes of operation from [1] in context of the fault model presented in previous sections. Other modes of operation (e.g. Statistical Cipher Feedback (SCFB) mode [15]) are not covered in this section.

Operation modes such as CBC (Cipher Block Chaining), CFB (Cipher FeedBack) and OFB (Output FeedBack) are designed to hide existing patterns in the plaintext. In context of those modes, the blocks that are consecutive encrypted are not consecutive values of a counter module. If the ECB (Electronic Code Block) mode is used, where n-bit blocks are encrypted one by one, the faults would determine incorrect decryption. So, in case of those operation modes our fault model is not causing security concerns and the faults are detectable due to unintelligible blocks at decryption. A more detailed presentation of the effect of error propagation for the operation modes that are mentioned in this paper can be found e.g. in [1].

IV. CONCLUSIONS

In this paper we present the CTR mode and how it is used based on NIST recommendations. In NIST Recommendations and IPsec Specifications the least significant bit of the counter blocks assures the differentiation of the key stream blocks which are XOR-ed with the plaintext blocks to produce the ciphertext blocks. If a fault affects the least significant bit, then two consecutive plaintext blocks are using the same key stream block for encryption. The confidentiality of all the plaintext blocks encrypted with the same counter block can be compromised. The change of the key is not removing the problem.

The use of a LFSR is recommended to avoid the vulnerability of a single or multiple bit faults based on the

model presented in section III. We consider that the ATM Security Specifications are more reliable in context of such a fault. The NIST recommendation [1] mentions but not establishes the use of LFSR for Standard Incrementing Function. Another mode to avoid this model of fault is to test if the output of the Counter module (Fig. 2) used for encryption of current plaintext block is different from the one used for previous plaintext block.

ACKNOWLEDGMENT

This project has been partially supported by K.U.Leuven project GOA/2001/04, and by the Fund for Scientific Research - Flanders (Belgium, F.W.O.) through "Krediet aan Navorsers 1.5.148.02".

REFERENCES

[1] NIST Special Publication 800-38A 2001 Edition, Recommendation for Block Cipher Modes of Operation, Methods and Techniques, available at: src.nist.gov/publications/nistpubs/800- 38a/sp800-38a.pdf

[2] R. Housley, IPsec Working Group, Using AES Counter Mode With IPsec ESP, Internet Draft, May 2003, available at: http://www.potaroo.net/ietf/ids/draft-ietf-ipsec-ciph-aes-ctr-04.txt.

[3] The ATM Forum Technical Committee, ATM Security Specification Version 1.1, af-sec-0100.002, March 2001, av. at ftp://ftp.atmforum.com/pub/approved-specs/af-sec-0100.002.pdf.

[4] FIPS81, DES Modes of Operation, Federal Information Processing Standard (FIPS), Publication 81, National Bureau of Standards, US Department of Commerce, Washington D.C., December 1980.

[5] H. Lipmaa, P. Rogaway, D. Wagner, Comments to NIST concerning AES Modes of Operation: CTR-Mode Encryption, Symmetric Key Block Cipher Modes of Operation Workshop, 2000, available at: www.tcs.hut.fi/~helger/papers/lrw00/html/

[6] D. Boneh, R. DeMillo, R. Lipton, On the Importance of Checking Cryptographic Protocols for Faults, *Eurocrypt '97*, LNCS 1233, Springer-Verlag, pp.37-51, 1997.

[7] E. Biham, A. Shamir, Differential Fault Analysis of Secret Key Cryptosystems, *Proceedings of Crypto'97*, 1997 available at: www.cs.technion.ac.il/~biham/publications.html

[8] W. Diffie, M. Hellman, Privacy and Authentication: An Introduction to Cryptography, Proceedings of the IEEE, 67 (1979), pp. 397–427.

[9] Joan Daemen, Vincent Rijmen, The Rijndael block cipher, AES Proposal: Rijndael, available at http://csrc.nist.gov/CryptoToolkit/aes/rijndael.

[10] NIST Report on the Development of the Advanced Encryption Standard (AES), James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, and co., publication date: October 2, 2000.

[11] S. Moriai, Y. L. Yin, S. Okazaki, The Camellia Cipher Algorithm and Its Use With IPsec, available at: http://www.ietf.org/proceedings/02mar/I-D/draft-ietf-ipsec-ciph-camellia-00.txt.

[12] E. Oswald, J. Daemen, V. Rijmen, AES -The State of the Art of Rijndael 's Security, available at: http://www.a-sit.at/technologieb/evaluation/aes_report_e.pdf

[13] D. Boneh, R. DeMillo, and R. Lipton, On the importance of checking cryptographic protocols for faults, *Journal of Cryptology*, Springer-Verlag, Vol. 14, No. 2, pp. 101-119, 2001.

[14] D. Boneh, R. DeMillo, R. J. Lipton, On the Importance of Eliminating Errors in Cryptographic Computations, extended version of the paper Eurocrypt '97, available at: http://crypto.stanford.edu/~dabo/abstracts/faults.html.

[15] H. Heys, Analysis of the Statistical Cipher Feedback Mode of Block Ciphers, *IEEE Tran. On Computers*, Vol. 52, No. 1, Jan. 2003, pp. 77-92.