# Basic comparison of Modes for Authenticated-Encryption (IAPM, XCBC, OCB, CCM, EAX, CWC, GCM, PCFB, CS)

Petr Švenda
Masaryk University in Brno, Faculty of Informatics
<xsvenda@fi.muni.cz>

## 1 Introduction

An authentication-encryption (AE) scheme is an encryption scheme with a pre-shared key providing both data privacy and authenticity. Basic AE scheme can be constructed by a naive (serial) combination of some existing encryption mode together with a message authentication code (MAC). The computation cost of such approach equals to cost of the encryption plus the cost of the MAC. There are several reasons why to design a dedicated mode. An improper combination of encryption and authentication mode can make whole scheme insecure. A random nonce (initialization vector) is often required, what is hard to achieve. Random and secret value of the nonce serves as "native" integrity protection of the nonce value and thus preventing predictable changes to a plaintext by attacker. A data encryption modes or MAC calculations are often not parallelizable and thus unsuitable for a high-speed environments.

Dedicated modes for authenticated-encryption try to solve at least some from the above-mentioned problems and also some others. The authentication significantly cheaper than calculating MAC alone can be provided. A mode security based on the security of the underlying cipher can be proven. A design as simple as possible to lower probability of misuse should be proposed. Need for the random nonce value can be avoided covering this value by an integrity protection. At least partial parallelizability can be enabled if a backward dependency is avoided. The resistance of known attacks against common constructions (CBC and CBC-MAC) like CBC forgery attack should be provided. In the practice, plaintext data like a packet headers must be often authenticated too. Some modes offer this feature, known as an authenticated-encryption with associated-data (AEAD), in advance. Note, that not all proposed modes achieves all from this features.

This article describes all AE modes proposed to National Institute of Standards and Technology (NIST) at present time available at [2]. Compares its main characteristics and discusses suitability for a computation environments with different conditions, focusing especially on the suitability for use with an obfuscated ciphers.

This text is organized as follows: Chapter 2 gives a brief overview of all AE modes proposed to NIST. In chapter 3 common characteristics for all modes are listed and compared. Chapter 4 presents performance of the modes with available source codes on a personal computer with AMD Barton 2500+ processor. Suitability for an obfuscated ciphers and environments with different potency to parallelism and amount of an available memory is discussed in chapter 5. Conclusions are given in chapter 6.

The following notation will be used in the text:
A plaintext message $M$ is divided into $m$ parts $P_1, \ldots, P_m$, encrypted into a ciphertext $C$. A non-secret associated data covered by authentication $A$ is divided into $r$ parts $A_1, \ldots, A_r$. A unique nonce $N$ for each message encrypted with a same key $K$ ($K_1, K_2$) for underlying cipher with a block size of $n$-bits. A message authentication is verified using an authenticator $T$ with length $\tau$. Exclusive-OR binary operation is notated as $\oplus$ or simply "xor" operation.

## 2 Authenticated-Encryption modes overview

### 2.1 IAPM

Integrity Aware Parallelizable Mode [7] (IAPM) was proposed by Ch. Jutla as first provable secure mode for authenticated encryption. Requires two independent keys $K_0, K_1$ with the same length as an underlying block cipher. Both an encryption and a decryption direction of a block cipher are used. An unique nonce $r$ (for each message using same keys $K_0, K_1$) is used to generate an "offsets", a pair wise differentially uniform vectors[1] with m + 1 members. This generation requires a one block cipher invocation using a key $K_0$, an integer additions over the Galois field using n-bits prime $p$ ($GF_p$) respectively only by the xor operations with a penalty of approximately $log_m$ extra block cipher invocations [7, pp. 4]. The nonce $N$ is communicated as part of a ciphertext. The rest of encryption process is shown on the picture 1 and consists from m block cipher invocations using key $K_1$ and the integer additions over $GF_p$ respectively xor operations. *Checksum* denotes to $P_1 \oplus P2 \oplus \ldots \oplus P_{m-1}$. During the decryption process, same "offsets" using $K_0$ are generated, and the encryption process is simply reversed. After the decryption, message integrity is verified by checking if $P_m = P_1 \oplus P2 \oplus \ldots \oplus P_{m-1}$. For the detailed specification refer to [7]. Only the version of IAPM substituting integer additions by xor operations is discussed in the following text.

Jutla also proposed another one mode called Integrity Aware Cipher Block Chaining Mode (IACBC) [7]. This mode is like IAPM, but instead of xoring the offset $s_i$ to $P_i$ (before block cipher invocation), ciphertext of the previous block is used (as in the CBC mode). Mode IACBC isn't parallelizable and due to missing test vectors, an additional documentation etc. seems to be less preferred by the author than IAPM.

IAPM is patented, fully parallelizable, provable secure mode without AEAD feature overcome by latter OCB mode.

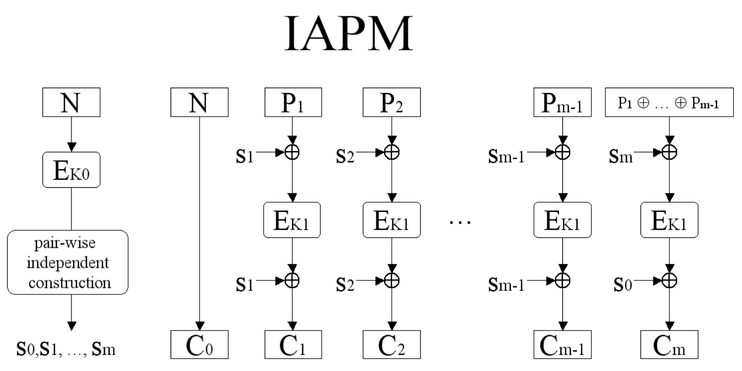

**Figure 1. Integrity Aware Parallelizable Mode scheme.**

### 2.2 XCBC

Extended Cipher Block Chaining Encryption [16](XCBC) is proposed in a three versions: stateless, stateful-sender and stateful-both. Only the stateful-sender version is considered here, because of having the same nonce requirement as other proposed modes. Using keys $K_1$ and $K_2$, values $R$ and $Z_0$ are created. $Z_0$ serves as an initial vector for CBC-like encryption, $R$ is used to post-whitening each ciphertext block $C_i$ by $i * R$. The authenticator $T$ is calculated from an additional plaintext block consists from $Z_0 \oplus P_1 \oplus P_2 \ldots P_m$. Whole process is shown on the picture 2. XCBC is quite similar to Jutla's IACBC mode, which was published little bit sooner. For a detailed specification refer to [16].

Basic XCBC mode contains backward dependency as in CBC mode and thus no parallelizability is possible. A modification for support to parallel execution is also proposed as ipXCBC mode. The message is divided into $L$ independent segments.

[1]A sequence of uniformly distributed n-bit random numbers $s_1, s_2, \ldots, s_m$, is called pair-wise independent if for every pair i, j, $i \neq j$, and every pair of n bit constants c1 and c2, a probability that $s_i = c_1$ and $s_j = c_2$ is $2^{-2n}$ [7, pp. 2]. In other words, $s_i$ is random and has no special relation to $s_j$.

Instead of use one value $z_0$, new values $z_0^s$ are computed for each segment: $z_0^s = E_K(N+s), s \in \{1,\dots,L\}$. The authenticator $T_s$ is calculated for each segment independently and appended to the ciphertext.

XCBC is patented, non-parallelizable, provable secure mode without AEAD feature. ipXCBC is patented, costly parallelizable, provable secure mode without AEAD feature.

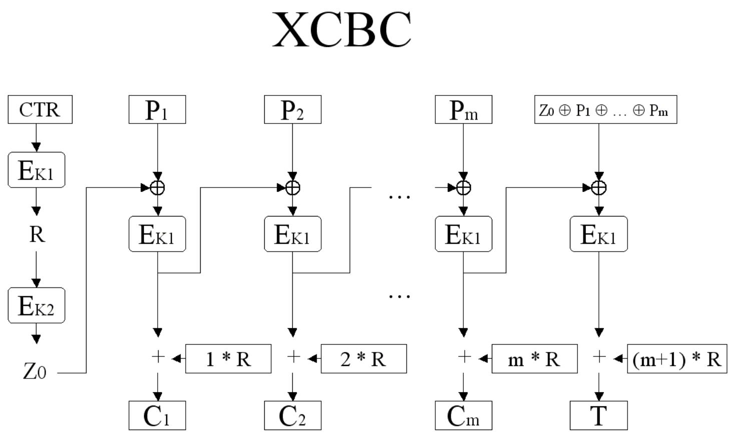

**Figure 2. Extended Cipher Block Chaining Encryption mode scheme.**

### 2.3 OCB

Offset CodeBook mode [8](OCB) is based on IAPM mode, but some new characteristics are added. Only one underlying key $K$ is needed for the "offset" calculations and the data encryption. More efficient way for the "offset" calculations is employed. Message integrity is verified using authentication tag $T$ with optional length $\tau$ (up to $n$-bits), varying depending on an application needs. The input message can be an arbitrary length[2] and is preserved in the output (except length of the authentication tag).

Both the encryption and the decryption direction of the block cipher are used. For every new message, non-repetitive nonce is used. For the key value $K$ value $L \leftarrow E_K(0^n)$ is computed (only once if $L$ can be stored for a future invocations). For each new nonce $N$, one block cipher invocation is used to create the intermediate value $R \leftarrow E_K(N \oplus L)$. Using the Gray codes $\gamma$, $L$ and $R$, the "offsets" $z_i$ are generated $z_i \leftarrow \gamma_i.L \oplus R$ [3]. The rest of the encryption process is shown on the picture 3 and consists from $m$ block cipher invocations using the key $K$ and the xor operations. $p_n^{-1}$ denotes to the inverse of (chosen) irreducible polynomial $p_n$[4] with a degree equal to the underlying cipher block length. *Checksum* denotes to $P_1 \oplus P2 \oplus \ldots \oplus P_{m-1} \oplus C_m 0^* \oplus Y_m$. For the detailed specification refer to [8].

OCM is patented, fully parallelizable, provable secure, flexible and very efficient mode with small memory requirements without AEAD feature.

### 2.4 CCM

Counter with CBC-MAC mode [10] (CCM) was designed as non-patented alternative to OCB. Mode combines *CBC-MAC* scheme for a data authentication and *CTR mode* [6] for a data encryption. Basically uses "mac-then-encrypt" approach, but some modifications are proposed: Only one underlying key $K$ is used for both the authentication and the encryption part. An additional plaintext data $A$ can be authenticated. A variable length nonce $N$, unique for each message, is combined with flags and counter ad serves as an input for a key stream generation in *CTR mode*. The message integrity is verified using

[2]No multiple of underlying the cipher block length or the padding is required.
[3]Precalculation of $\gamma_i.L$ for the given key is possible.
[4]E.g. for AES (128-bits block) $p_{128}(x) = x^{128} + x^7 + x^2 + x + 1$, $p_{128}(x)^{-1} = x^{127} + x^6 + x + 1$.

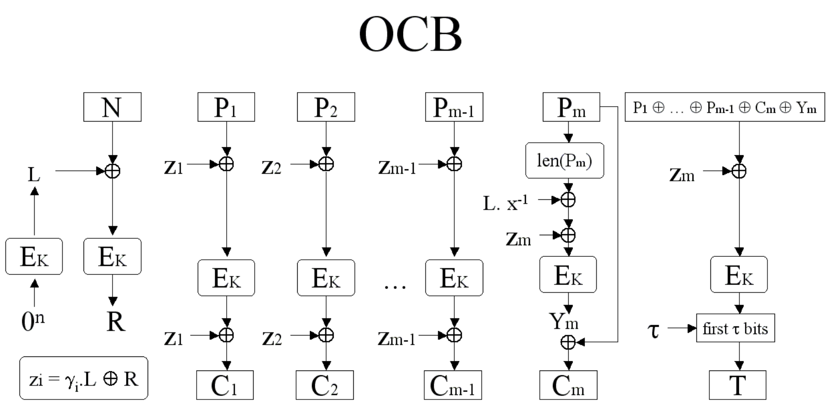

**Figure 3. Offset CodeBook mode scheme.**

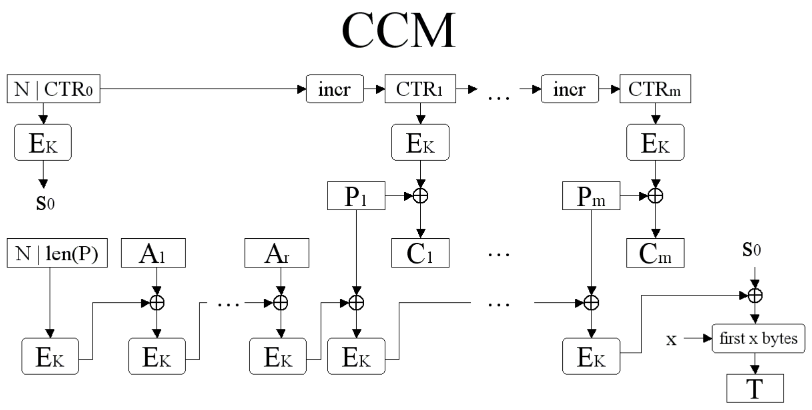

**Figure 4. Counter with CBC-MAC mode scheme.**

the authentication tag $T$ with variable length of 4, 6, 8, 10, 12, 14 or 16 bytes. CCM mode is approved specially for the ciphers with 128 bits block length. A whole process is shown on the picture 4 and consists from $2 * m + 2 + r$ block cipher invocations using key $K$ and the xor operations. For a detailed specification refer to [10].

CCM is non-patented, non-parallelizable, provable secure mode with small memory requirements with AEAD feature. The design is less flexible, targeted to the underlying ciphers with 128-bit block length.

## 2.5 EAX

EAX mode [11] is based on the “general composition” approach and uses CTR mode [6] for a data encryption and OMAC [17] hash algorithm for a data authentication. An additional plaintext data $A$ can be authenticated. One key $K$ is used. At first, an initial value $CTR_1$ for CTR mode is created applying $OMAC_K^0$ over the nonce $N$. The plaintext $P$ is than encrypted using standard CTR mode. The authentication tag $T$ is created xoring together $CTR_1$, $OMAC_K^2(C)$ and $OMAC_K^1(A)$ and extracting the first $\tau$ bits. A whole process is shown on the picture 5.

OMAC algorithm is based on CBC-MAC hashing approach, but uses an additional padding method. At first, zero block $(0^n)$ is encrypted with the key $K$ giving a value $L$. Two padding values $2L$ and $4L$ are than derived. If an input data are not aligned to multiple of the underlying cipher block length, padding with 1 and necessary number of the zeros is used. If the padding is used, value $4L$ is than xored at the end of input data, otherwise the value $2L$ is used for the same operation. The result is than processed using the standard CBC-MAC construction. Three slightly personalized versions of OMAC algorithm are used ($OMAC_K^0$, $OMAC_K^1$, $OMAC_K^2$). $OMAC_K^x(DATA)$ is shortage for $OMAC_K(x|DATA)$.

EAX is non-patented, non-parallelizable, provable secure mode with a small memory requirements and AEAD feature, less efficient for shorter messages.

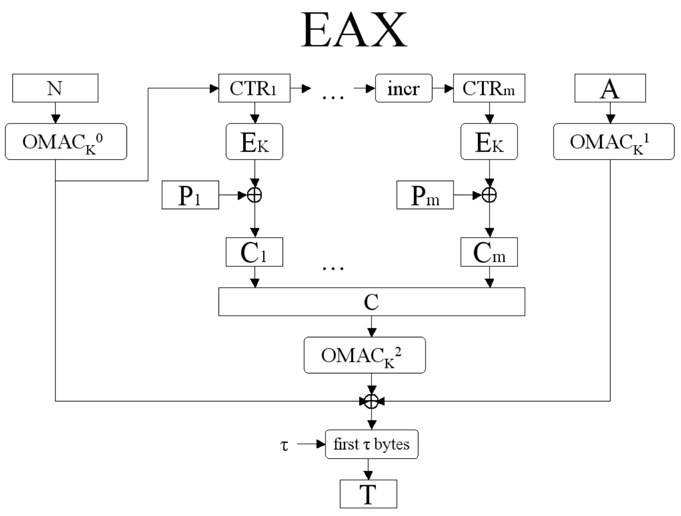

**Figure 5. EAX mode.**

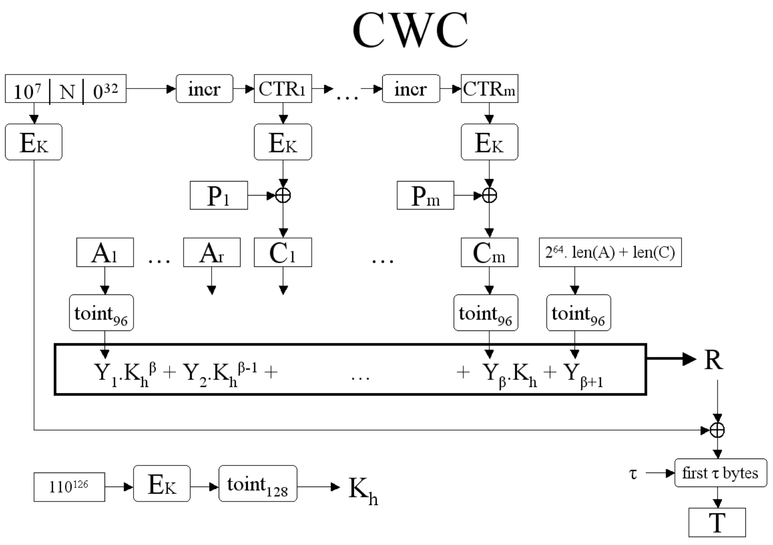

**Figure 6. Counter with Carter-Weigman hash mode scheme.**

## 2.6 CWC

CWC mode [12] combines CTR mode for a data encryption with a Carter-Wegman universal hashing function [4] over $2^{127} - 1$ field for a data authentication. Basically use "encrypt-then-mac" approach with a both steps chosen to be parallelizable. Only one key $K$ is used, but another one $K_h$ is derived from $K$ applying one block cipher invocation with the key $K$ to the fixed constant. The result is converted from binary to integer number representation. At first, the unique nonce $N$ is combined together with an incremental counter and used with the key $K$ in CTR mode to encrypt the blocks $P_1, \ldots P_m$, resulting in the ciphertext $\sigma$. After then, an associated payload data $A$ are bundled together with the ciphertext, $\sigma$, the length of the associated payload data $l_A$ and a length of the ciphertext $L_\sigma$ and padded to the necessary length. The result $Y$ is divided into 96-bits parts $Y_1, \ldots, Y_r, Y_{r+1}$. Each part is converted to an integer number and used as coefficient for the Carter-Weigman polynomial[5] applied to value of $K_h$. Solving of the Carter-Weigman polynomial in a group over residual class with a prime base[6] can be done in parallel way [12, pp. 2]. A whole process is shown on the picture 6. For a detailed specification refer to [12].

CWC is non-patented, parallelizable, provable secure, less efficient mode overcome by GCM mode in the all considered properties.

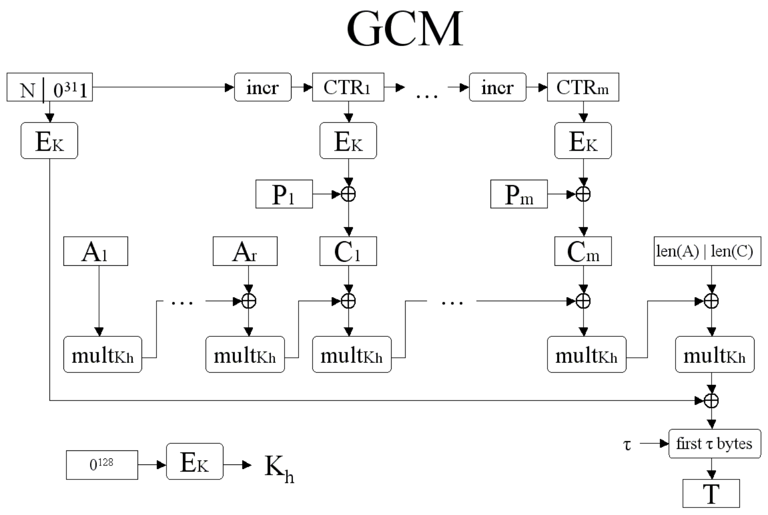

**Figure 7. Galois/Counter Mode scheme.**

## 2.7 GCM

The Galois/Counter Mode of Operation (GCM) [13] refines CWC mode using binary Galois field GF($2^{128}$) instead of 127-bit integer field for the Carter-Wegman polynomial hashing [4] to avoid an expensive integer multiplication operations. Operations over binary fields can be implemented in hardware with smaller circuits depth, saving a die space and offering a higher throughput. The multiplication in GF($2^{128}$) can be efficiently implemented using the shift and the xor operations only. The authentication part of GCM can be effectively accelerated using precomputed arrays with size up to 64 Kb. Other parts remain unchanged regarding to CWC. A whole process is shown on the picture 7. For a detailed specification refer to [13].

GCM is non-patented, parallelizable, provable secure, flexible and efficient mode (if the precomputed storage can be used) with AEAD feature, comparable to OCB in the most environments.

## 2.8 PCFB j/k

Propagating Cipher Feedback mode [14](PCFB) is based on Cipher Feedback mode [5, pp. 231](CFB). In a contrast to j-bit CFB mode, which use the left shift by $n - j$ bits, PCFB use the left shift by $k - j$ instead, assumes that $k < n$ and introduce additional right shift operation by $j$ bits. PCFB comes in a version without and with message authentication (AREA[7]). In AREA version, a redundancy is introduced by inserting number of message blocks as an additional $P_0$ and

[5] $Y_1 x^r + Y_2 x^{r-1} + Y_3 x^{r-2} + \ldots + Y_r x + Y_{r+1} \; mod \; 2^{127} - 1$
[6] $2^{127}$ - 1 is a prime.
[7] Added Redundancy Explicit Authentication.

$P_{m+1}|\ldots|P_{m+128/j}$ blocks. PCFB claims that redundancy together with error propagation provides message authentication, but no formal proof was proposed. A whole process is shown on the picture 8. For a detailed specification refer to [14].

PCFB is non-parallelizable mode without security proof and with lack of available analysis. Patent status is unknown.

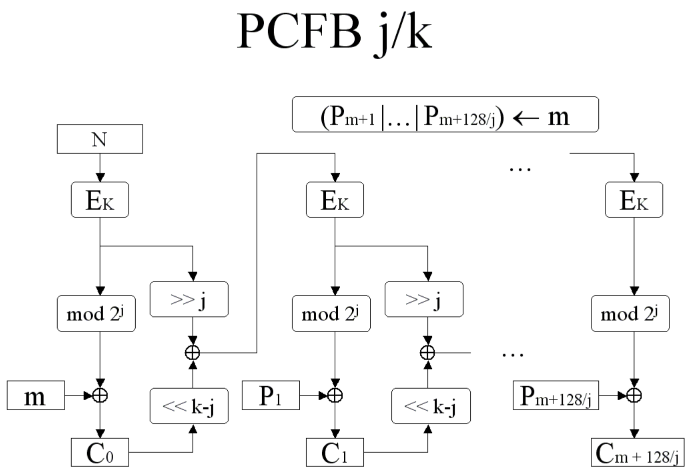

**Figure 8. Propagating Cipher Feedback mode scheme.**

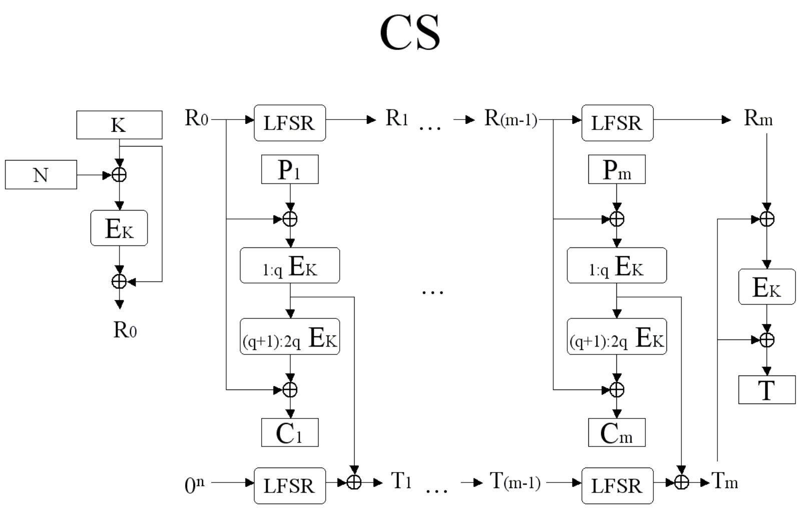

**Figure 9. Cipher-State mode scheme.**

## 2.9 CS

Cipher-State mode [15](CS) uses an internal state of the cipher produced by underlying round-based cipher during a data encryption together with the linear feedback shift registers (LFSR) used to pre-whiten the plaintext and post-whiten the ciphertext. One key $K$ is used. An initial whitening value $R_0$ is created from the nonce $N$ and the key $K$. The plaintext block $p_i$ is pre-whitened using $R_i$, encrypted using the key $K$ and post-whitened with $R_i$. After a half ($q$) of total number of rounds of the underlying block cipher, internal state[8] of cipher is xored with the current pre-authenticator value $T_i$. The values

[8]E.g. current STATE matrix for AES.

$R_i$ and $T_i$ are updated after each step using LFSR with a connection polynomial taken as lexicographically least primitive polynomial with degree $n$. To stop possible information leakage due to using the internal cipher state, a final authenticator $T$ is computed using an addition block cipher invocation: $T \leftarrow E_K(T_i\ xor\ R_i)\ xor\ T_i$. Alternatively, a pre-authenticator value $T_m$ is processed together with $K$ and $R_m$ throw a one-way hash function resulting in the authenticator $T$ with length depending on used hash function. A whole process is shown on the picture 9.

No formal proof was proposed. In my opinion, it will be hard to proof security of mode independently from underlying cipher. If underlying cipher behave as "pseudorandom" generator and $2q$ rounds are needed to ensure this, internal cipher state after only $q$ rounds will probably have weaker properties (or $2q$ rounds are uselessly many). The attacker can distinguish between a random value and the internal cipher state with higher probability than between the random value and the ciphertext. This increases the attacker probability to successfully forge the authenticator value. I can't say, how much usage of LFSR improve "pseudorandom" property of the pre-authenticator values stream. For a detailed specification refer to [15].

CS is non-patented, parallelizable, flexible and very efficient mode with small memory requirements, but without security proof and AEAD feature.

## 3 Characteristics comparison

The characteristics mentioned in this section are taken from the basic list of properties required for NIST proposal and extended by features, which varies across modes. The characteristic same for the all proposed modes (nonce requirement, synchronization) are omitted. The characteristics are summarized in the table 1 (fully-parallelizable modes) and the table 2 (remaining modes). A number of a block cipher invocations of the underlying cipher for a *init* phase (once for each new key) and a *crypt* phase (for every message) are summarized in the table 3.

The main characteristics:

- **Patent issues** – Initiatory provably secure modes for the authenticated encryption (IAPM, OCB, XCBC, ipXCBC) are patented. The patent situation is not clear and some patents are probably overlapped. The modes, which come later are trying to be patent aware (CCM, EAX, CWC, GCM, CS). No intellectual property statement was found for PCFB mode.

- **Provably secure** – Some submitted modes come with a proof of security based on an assumption that the underlying block cipher behaves as pseudorandom generator. The message privacy and authenticity while using given mode is proved together with fact, that mode don't significantly increase the attacker possibility to distinguish between a ciphertext and a random stream.

  Provably secure modes are (proofs available in a square bracket reference) IAPM [7], OCB [8], CCM [20], EAX [11], CWC [21], GCM [22], and XCBC and ipXCBC [16].

- **Performance** – Performance benefits (especially in hardware with potency for a parallel execution) are one from a key motivation why to introduce a new mode for a data encryption/authentication rather than use some variant of a standard "encrypt-than-mac" approach. Many factors impacts a result performance: an execution environment conditions (see 5), parallelizability, a number of block cipher invocations, an average length of messages, a frequency of changing keys, a key schedule etc.

  The number of block cipher invocations (see table 3) is partial indicator of a possible mode speed, but more other influences must be considered. For a fast underlying cipher (like AES), this factor is less significant than for a slower ones. A modes using cipher for an authenticator calculation (having more invocations) can be finally faster in a serial environments than a modes using other "non-cipher" based approach. A fewer invocations are more suitable for a high speed parallel processing, because of smaller implementation size of a total implemented cipher engines.

  Chapter 4 discuss a speed comparison for selected modes (with available reference code) on PC in almost serial environment.

- **Parallelizability** – Degree of an each mode parallelizability is important factor for a high-speed environments, where a multiple tasks can be done parallel. Fully parallelizable modes are listed in the table 1 having an encryption and an authentication part parallelizable (denoted as A+E). A parallelizable version of XCBC mode (ipXCBC) is submitted, but

| **Feature** | **IAPM** | **OCB** | **CWC** | **GCM** | **CS** | **ipXCBC** |
|---|---|---|---|---|---|---|
| Patent | yes | yes | no | no | no | yes |
| Parallelizability | E+A | E+A | E+A | E+A | E+A | E+A |
| Provably secure | yes | yes | yes | yes | no | yes |
| Cipher text expansion | $0 \dots n + n$ | $\tau$ | $\tau$ | $\tau$ | $0 \dots n + n$ | $0 \dots n + (L+1) * n$ |
| Keying material | 2 keys | 1 key | 1 key | 1 key | 1 key | 2 keys |
| Online | yes | yes | yes | yes | yes | yes |
| Endian dependent | yes | no | yes | yes | yes | yes |
| Incremental MAC | no | no | no | yes | no | yes |
| Error propagation | no | no | no | no | no | yes |
| Two-pass | no | no | yes | yes | no | no |
| Authenticator length | $n$ | $0 \dots n$ | $0 \dots n$ | $0 \dots n$ | $n$ | $(L+1) * n$ |
| Only encrypt engine | no | no | yes | yes | no | no |
| Associated data auth | no | no | yes | yes | no | no |

**Table 1.** *A summary of basic properties for fully parallelizable modes.*

| **Feature** | **CCM** | **EAX** | **PCFB** | **XCBC** |
|---|---|---|---|---|
| Patent | no | no | N/A | yes |
| Parallelizability | E only | E only | no | no |
| Provably secure | yes | yes | no | yes |
| Cipher text expansion | $16k, k \in \{2, \dots, 8\}$ | $\tau$ | $(128/j) * n$ | $0 \dots n + n$ |
| Keying material | 1 key | 1 key | 1 key | 2 keys |
| Online | no | yes | no | yes |
| Endian dependent | yes | yes | yes | yes |
| Incremental MAC | no | no | no | no |
| Error propagation | no | no | yes | yes |
| Two-pass | yes | yes | no | no |
| Authenticator length | $16k, k \in \{2, \dots, 8\}$ | $0 \dots n$ | $128/j$ | $n$ |
| Only encrypt engine | yes | yes | yes | no |
| Associated data auth | yes | yes | no | no |

**Table 2.** *A summary of basic properties for serial modes.*

expensive due to a ciphertext expansion and an overhead computation. A modes using only an encrypt engine (CWC, GCM) saves a space needed for the decryption engine and thus offer a potency for a higher degree of parallelizability on a same area.

- **Memory requirements** – Total memory used by a specific mode is given by the underlying cipher requirements, number of a required different keys, a space occupied by a scheduled keys, a size of a message fraction needed for processing and an amount of a precalculated arrays necessary for a reasonable execution speed. If a message length is not known in advance, online property can significantly save a used memory. If a mode uses only an encryption engine, the memory needed for a decryption engine (like a precalculated arrays) and a decrypt key schedule is saved.

  Except IAPM and XCBC mode, only one key is required. Mode GCM uses precalculated arrays to significantly speed up an execution. This can be an issue in memory-restricted environments like present smart cards.

- **Associated data authentication (AEAD)** – Possibility to coverage also an unencrypted data (like packet headers) with an integrity protection saves cost for additional mechanisms, which must be otherwise implemented beyond mode itself, simplifies an implementation and reduce a probability of an incorrect implementation. Only IAPM and OCB modes don't have this property.

  Some modes can be probably extended by minor changes to support AEAD feature, but this extension should be properly proved and included in the mode specification. Proposed modes should declare if support for AEAD feature is possible or not.

| Mode | Cipher invocations (init) | Cipher invocations (crypt) |
|---|---|---|
| IAPM | 0 | $\lceil \lvert M\rvert/n\rceil + log_2\lceil \lvert M\rvert/n\rceil + 1$ |
| OCB | 1 | $\lceil \lvert M\rvert/n\rceil + 2$ |
| CCM | 0 | $2\lceil \lvert M\rvert/128\rceil + \lceil \lvert A\rvert/128\rceil + 3$ |
| EAX | 3 | $2\lceil \lvert M\rvert/n\rceil + \lceil \lvert A\rvert/n\rceil + 1$ |
| CWC | 2 | $\lceil \lvert M\rvert/n\rceil + 2$ |
| GCM | 2 | $\lceil \lvert M\rvert/n\rceil + 1$ |
| PCFB j/k | 0 | $\lceil \lvert M\rvert/n\rceil + 128/j + 1$ |
| CS | 0 | $\lceil \lvert M\rvert/n\rceil + 2$ |
| XCBC | 0 | $\lceil \lvert M\rvert/n\rceil + 3$ |
| $ipXCBC_L$ | 0 | $\lceil \lvert M\rvert/n\rceil + 2L + 2$ |

**Table 3.** *A number of the underlying block cipher invocations. The phase (init) is executed only once for an each new key(s). The phase (crypt) is executed repeatedly for an each new message.*

- **Ciphertext expansion** – This property is important especially for a short messages (few blocks) where can even overcome an original message length. All proposed modes expand the message at least by a length of an authentication tag. Moreover, IAPM, CS and XCBC require the message length to be aligned as the underlying cipher block length. Note that the nonce $N$ must be communicated or pre-shared every time and its length isn't calculated into a ciphertext expansion.

- **Online message processing** – A possibility to process a message (both encrypt and decrypt direction) without obtaining the whole message is an important property especially in a memory restricted environments. CCM and PCFB mode doesn't have this property and the message length must be known in advance before starting computation of an authentication part. All other modes have online property.

- **Endian dependency** – The endian dependency of mode limits code portability. If the underlying cipher is not endian dependent, endian independent mode should be preferred. All modes, which use the integer addition/multiplication, are the endian dependent, if these operations can't be substituted by the endian independent operations like the xor operation. OCB is only one proposed endian independent mode.

- **Underlying cipher requirements** – No detailed requirements are given by NIST to an authentication encryption mode description. Some of proposed modes are presented in a version for cipher with a concrete block length (typically AES-128), but can be easily extended to other sizes. Except CCM mode, which bundle together information about a data lengths into one block, no low bound for a block size is given. All proposed modes require underlying cipher to be a block cipher.

- **Nonce requirement** – Since generating a truly random nonce is difficult task, all proposed modes are designed to achieve a security only with a non-repeating nonce like a counter or a packet ordinal number. This property simplifies an implementation and makes correct usage easier. A unique value of the nonce is required for each message encrypted by same key.

- **Incremental MAC** – In an applications in which a large and dynamic data set are changing frequently and must be authenticated, such as a remote database, file system, or network storage, recalculating an authenticators for whole data can be an inefficient. Only two modes (GCM, ipXCBC) offer more reliable feature having a computational cost of a new authenticator proportionally to the number of bits that were changed.

- **Authenticator length** – CCM mode prescribes length of authenticator to be 4, 6, 8, 12, 14 or 16 bytes. OCB, EAX, CWC, GCM modes allows variable length of tag up to $n$ bits by truncating an original authenticator value to a size by application needs. Other modes do not propose the truncating of an original authenticator. PCFB j/k use as the authenticator data redundancy presented in additionally added $128/j$ n-bits blocks containing number of blocks of plaintext message.

| **Operation mode** | **Message size (bytes)** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **16 (1)** | **20 (2)** | **40 (3)** | **64 (4)** | **128 (8)** | **256 (16)** | **1024 (64)** | **8192 (512)** |
| CCM | 161.47 (4) | 181.57 (6) | 116.26 (8) | 87.59 (10) | 76.24 (18) | 70.74 (34) | 66.48 (130) | 65.25 (1026) |
| EAX | 227.09 (6) | 235.43 (8) | 145.21 (10) | 105.38 (12) | 85.01 (20) | 75.12 (36) | 67.52 (132) | 65.20 (1028) |
| CWC | 214.66 (3) | 197.53 (4) | 122.19 (4) | 106.27 (6) | 88.47 (10) | 80.92 (18) | 74.46 (66) | 72.48 (514) |
| GCM, 4Kb storage | 166.41 (2) | 188.57 (3) | 119.49 (4) | 90.27 (5) | 79.17 (9) | 79.17 (17) | 69.04 (65) | 67.73 (513) |
| GCM, 8Kb storage | 154.41 (2) | 154.41 (3) | 154.41 (4) | 82.91 (5) | 82.91 (9) | 67.06 (17) | 65.29 (65) | 61.46 (513) |
| GCM, 64Kb storage | 122.34 (2) | 136.78 (3) | 85.29 (4) | 63.26 (5) | 55.68 (9) | 50.68 (17) | 48.21 (65) | 47.07 (513) |
| OCB | 118.91 (3) | 125.63 (4) | 78.29 (5) | 58.87 (6) | 48.83 (10) | 43.85 (18) | 39.87 (66) | 38.77 (514) |

**Table 4.** *AE modes "cycles per byte" comparison on AMD Athlon 2500+ (Barton) with AES as underlying cipher. Value in braces shows number of underlying block cipher invocations.*

## 4 Speed comparison on PC

Comparison of the modes with available optimalized implementation was performed on PC with AMD Barton 2500+ processor, running MS Windows 2000 operating system. B. Gladmann [18] codes for CCM, EAX, CWC and GCM mode were used. T. Krovetz [19] code for OCB mode was used, modified to use same underlying AES cipher implementation and integrated into the Gladmann's test project. Visual Studio C++ 6.0 compilator with an unspecific (blend *) target processor was used.

Tests were performed only in a serialized environment. The advantages of fully parallelizable modes (OCB, CWC, GCM) are not exposed in this test. The main part of execution time for almost all modes is dominated by the underlying block cipher (AES) execution speed. So, improving the cipher speed (optimalization, H/W acceleration, ...) has more significant impact on the modes, which use more cipher invocations like CCM and EAX.

OCB mode offers the best performance over all tested modes for both short and long messages. With 64Kb of a precalculated storage, GCM offers almost a comparable speed to OCB with an advantage of not being patented. Other modes significantly worse performance. From non-tested modes, CS seems to offer also a high performance (see [15, pp. 6]).

The differences between my and GCM inventors measurements [13, pp. 21] (Motorola G4 processor) were detected. On the short messages with 16 bytes length, GCM should be significantly faster (30%) than OCB mode. This result was not confirmed by my tests.

## 5 Suitability for different environments

The advantages and disadvantages of each mode highly depend on the parameters of a target computation environment like potency for parallel execution, available memory and native architecture support for the operands sizes. Three main groups were identified and additionally specific situation for the obfuscated ciphers is discussed.

### 5.1 Powerful, highly parallelizable environments

A computation environment with need for high data throughput with potency for high level of parallelism and sufficient memory for precomputation. The multiple cipher engines can be implemented and executed in a parallel. Field programmable arrays (FPGA) or the specifically designed circuits can be used. An example of such devices is dedicated hardware accelerators used in high speed networks (10Gbs).

Basically, only the fully parallelizable modes listed in table 1(IAPM, OCB, CWC, GCM, CS and ipXCBC) are suitable for this environment. Non-parallelizable modes from table 2 are omitted from discussion in this environment. The modes which use integer addition/multiplication operations (CWC, XCBC) are generally penalized, because of a slower execution time and the bigger circuits depth for these operations. The modes using only an encrypt engine (CWC, GCM) allows more engines on same die space, but necessary space for an authenticator calculation (Carter-Weigman hash) must be considered. CS mode use LFSR, which can be implemented very efficiently in hardware. ipXCBC mode is parallelizable, but with a high computation overhead.

No exact test was made for the highly parallelizable environments. But from the observations mentioned above, suitable modes should be GCM, patented OCB, and alternatively[9] CS.

## 5.2 Powerful, but almost serial environments

The computational environments with a sufficient memory for precomputation, but with no or very little potency for parallelism. An example can be common personal computer with general-purpose RICS processor[10]. No hardware accelerator for the underlying block cipher is assumed.

The modes using an integer addition/multiplication operations (CWC, XCBC) are generally penalized by a slower execution time of these operations (see table 4). CCM and EAX use approximately two times more invocations of underlying cipher and more faster cipher thus increase its speed more significantly. Results for a fast cipher like AES shows that this modes are still defeated by GCM and OCB mode.

A test was made only for the modes with publicly available implementations. The results are discussed in 4. Suitable modes for this environment are GCM and OCB, alternatively CS mode.

## 5.3 Restricted environments

The computation environments with a limited and slow memory and an expensive computation of some operations like integer multiplication or floating point operations with operands longer than architecture supports natively. An example of such device can be smartcards or some small embedded systems like sensor network nodes.

Modes, which use only encryption engine (CCM, EAX, CWC, GCM, PCFB) significantly saves both die space and memory needed for a decryption key schedule. If a message size is not known in advance, support for an online message processing is vital due to a limited memory storage. The modes using an integer addition/multiplication operations (CWC, XCBC) are not acceptable, because of typical architecture support for only 8 or 16 bits operands. The small devices with less efficient processor must have a cryptographic coprocessor for reasonable execution speed of the underlying cipher. This provides advantage to the modes with more cipher invocations (CCM, EAX). For truly memory-limited environments, the modes which use more extensive precomputed storage (GCM) cannot be used.

No exact test was made for the restricted environments. But from observations mentioned above, suitable modes should be EAX, patented OCB, and an alternatively CS if obtaining of cipher internal state is possible.

## 5.4 Environments with obfuscated ciphers

An obfuscated cipher is non-standard implementation of standard cipher algorithm, trying to protect the used keys against attacker, which can observe cipher code and used data during execution. The obfuscated cipher can be constructed applying common code obfuscating technology on standard implementation of cipher, or (better) design a whole cipher in mobile cryptography [23] manners (MC). An example of such approach is white-box attack resistant AES (WAES) and DES [24, 25] implementations. The obfuscated cipher can be utilized in scenarios, where a software application is deployed in possibly hostile environment. Attack from privileged user with access to a cipher execution (memory, processed instructions) or malware is expected. An advantage of using standard cipher algorithm together with a standardized mode can be significant in cases, where hardware accelerators can be employed on at least one side (smartcards or digital rights management server).

A discussion bellow in this section is lead by a general idea that suitable mode should use the advantages of the obfuscated ciphers to make difficult to extract the cipher engines, reveal processed data or make a targeted modification. In an ideal

[9] No security proof for authentication part.

[10] Note that due to current market trends in the processors development, which positioning more execution units into one chip (like P4 with Hyper Threading technology) into common personal computer, parallelizable modes can still have significant advantage than serial ones. This case is not considered in this category, but if two modes have comparable results and requirements in strictly serial environment, parallelizable one should be chosen.

situation, all mode's operations can be implemented in mobile cryptography manner as well as cipher itself. Parallelizability and performance is often not considered as most important feature.

I have identified the following characteristics as being most important for obfuscated cipher usage:

- **No direct access to key value** – A key value is permanently embedded into cipher instance, cannot be directly accessed and is computationally hard to extract an open value. This serves as a protection against an attacker, but limit an usability of the key value. The key must be used in mode as "crypt machine" with embedded key.

  CS mode requires a direct value of a key $K$ to generate $R_0$. For MC design, this should be placed by different construction. Other modes don't need a direct key value.

- **Error propagation** – An error propagation can be used as integrity supporting mechanism robust against modification of an integrity verification process by attacker. If the error is propagated, the decryption of corrupted message will result in unusable plaintext, which causes probably crash if application tries to use it. The error propagation of mode is useful from this point of view, because attacker can easily void comparative operation (during integrity check) against expected authenticator.

  The error propagation is present in PCFB, XCBC and ipXCBC modes, being vital especially for PCFB, which builds authentication part security only on redundancy and error propagation.

- **"Directional" keys** – Separability of an encryption and a decryption engines of obfuscated cipher can be used to introduce "directional" keys for upper laying cryptographic protocol. The different keys $K_{AB}$ and $K_{BA}$ are used for encryption of messages exchanged between A and B. Key $K_{AB}$ is used to encrypt messages coming from A to B and key $K_{BA}$ for messages from B to A. Party A thus require only a decryption engine for $K_{BA}$ and an encryption engine for $K_{AB}$ (inversely for B). Using this, attacker which extracts cipher engines from an application code of party B is not able to create a messages appears to be from A until $K_{AB}$ is known.

  The directional keys can be used with IAPM, OCB, CS, XCBC and ipXCBC modes.

- **No direct access to cipher intermediate values** – The obfuscated cipher execution consist from series of steps with intermediate results protected by MC method, e.g. (random) bijectional mapping.

  Only CS mode requires an internal cipher state and its accessibility depends on implementation of the obfuscated cipher. For example when using WAS cipher, used mapping must be known in advance, but retrieving of state is possible.

- **Input and output encoding** – Idea of protecting intermediate values by the bijectional mappings can be extended onto an input and an output data of a cipher engine. The input data are threaten as being transformed by mapping $IC$ and the output data are finally transformed by $OC$. Until $IC$ and $OC$ are not known to attacker, cipher engine cannot be used as standalone "crypt machine". More, $IC$ can be used as output encoding for direct preceding operation (same for $OC$) and allows integrating cipher engine into surround operations in mobile cryptography manner.

  Manipulations done by mode to the intermediate values should be protect able by mobile cryptography techniques. Currently polynomial and rational function can be protected [**?**]. Also operations, whose arguments can be divided and processed independently per smaller blocks (like xor) are more suitable, because of limited size[11] of bijectional mappings use for an intermediate result protection.

  The modes using bit-level shifts (CS, PCFB) or integer multiplication (CWC, XCBC, ipXCBC) with the arguments bigger than $w$-bits of used mapping are hard to implement, if cannot be substituted by other operation. If parameters $j$ and $k$ for PCFB mode are chosen to satisfy $j = a * w$ and $k - j = b * w \; a, b \in \{1, \ldots\}$, shift operation on a block level do not require direct access to the intermediate value.

None of proposed modes is ideally suitable for MC design. The following ideas and suggestions were conducted:

– A direct key value access must be omitted.

[11] Some mobile cryptography approaches (like WAS does) limit maximum possible wide $w$-bits of bijectional mappings usable for the input and the output data encoding. For n-bit block of underlying cipher, $\lceil n/w \rceil$ different mappings is used dividing block into $\lceil n/w \rceil$ segments.

– An error propagation is important advantage until serves as native protection against modifications done by attacker.

– CTR mode preclude “directional” keys. This important feature of upper laying protocol should be supported in this context.

– Carter-Weigmann hashing polynomial over $GF(2^n)$ seems to be well implementable in MC with the precalculated storages with an embedded $K_h$.

– A bit shift operation can be efficiently used, if a shift length corresponds to intermediate values encoding.

## 6 Conclusion

The modes proposed to NIST as the authenticated-encryption scheme were briefly described. Their main characteristic were summarized in the tables 1, 2 and 3. Patented OCB is successor of patented IAPM mode and outclasses him completely. GCM is successor of CWC mode and outclasses him in all environments, where at least 4Kb precalculated storage is possible. EAX is more flexible alternative to CCM with similar design, having advantage of the online message processing, but being slower than CCM on the short messages. Except for “patent issues”, OCB mode seems to be most universal candidate for most scenarios. Except environments with critically limited memory, non-patented GCM offer comparable performance characteristics like OCB and saves space needed for decryption engine. PCFB and CS modes should be studied more deeply.

No straight favorite for all possible environments was found. Three different types of environment were identified, based on its ability to a parallel execution and an available memory. Additionally, specific needs for use of mode together with an obfuscated cipher were discussed too.

In powerful, highly parallelizable environments, modes GCM, OCB and CS were selected as the most suitable. The selection was based on fully-parallelizable design, no use of slow integer addition/multiplication and good performance in PC test (see chapter 4). No exact tests were made and suitability for this environment should be studied more deeply.

In powerful, but almost serial environments like current personal computers, modes GCM, OCB and CS were selected as the most suitable. The selection was based performance tests for modes with available optimalized implementation (CCM, EAX, CWC, GCM and OCB). For GCM mode, implementation with sizes 4Kb, 8Kb and 64Kb of precomputed storage were used. The results shown, that the fastest, but patented mode OCB is nearly followed by unpatented GCM with 64kB storage. Other tested modes are significantly slower. From results for CS mode listed in [15, pp. 6] for CS mode, its performance should be also comparable to GCM and OCB.

For restricted environments with low memory and serial execution like present smartcards, EAX, OCB and CS modes were selected as the most suitable. The selection was based on performance results for PC (see 4), but GCM mode was omitted due to need for large precomputed storage. Modes using integer addition/multiplication with large arguments were omitted due to poor performance amplified by fact, that current smartcards use lower operand sizes that current PC makes this operations even slower. Online feature (EAX, OCB) of mode considered as big advantage, due to limited storage for incoming message. More invocations of underlying block cipher are advantage (EAX, CCM), if the cryptographic co-processor is present. CS mode should have a good performance, but obtaining internal cipher state can be problematic if co-processor is used.

None from the proposed modes is ideally suitable for use with obfuscated ciphers. Performance is often not a main concern, the resistance of implementation of mode together with underlying cipher against observation or modification by attacker is more important. From this point of view, serial combination of CBC mode with MAC is still more suitable. Ideal mode for this environment should have infinite error propagation, use encryption and decryption engines, make no direct access to key value and make only limited manipulations with intermediate values.

My personal favorites are GCM and CS modes, both for absence of patents, high efficiency and parallelizability. GCM mode additionally due to provably secure design and CS mode for small memory requirements and smart idea used for a data authentication.

## References

[1] NIST: Advanced Encryption Standard AES, 2000. Available at http://www.nist.com/aes/ (October 2004)

[2] NIST: Modes of Operation for Symmetric Key Block Ciphers. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/

[3] NIST: Dictionary of Algorithms and Data Structures. Available at http://www.nist.gov/dads/HTML/graycode.html (October 2004)

[4] Wegman, M., Carter, L.: New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences 22, pp. 265-279, 1981.

[5] Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press 1996/2001. Available at http://www.cacr.math.uwaterloo.ca/hac/ (October 2004).

[6] Lipmaa, H., Rogaway, P., Wagner, D.: Counter mode encryption. NIST submission. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/ctr/ctr-spec.pdf (October 2004)

[7] Jutla, Ch.: Parallelizable Encryption Mode with Almost Free Message Integrity, 2000. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/iapm/iapm-spec.pdf (October 2004)

[8] Rogaway, R., Bellare, M., Black, J., Krovetz, T.: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption, 2000 Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/ocb/ocb_full.pdf (October 2004)

[9] Rogaway, R.: OCB Mode: Parallelizable Authenticated Encryption, 2000

[10] NIST Special Publication 800-38C: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, NIST, May 2004. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/ccm/ccm.pdf (October 2004)

[11] Bellare, M., Rogaway, P., Wagner, D.: Conventional Authenticated-Encryption Mode. 2003 Available at http://eprint.iacr.org/2003/069/ (October 2004)

[12] Kohno, T., Viega, J., Whiting, D.: The CWC Authenticated Encryption (Associated Data) Mode, NIST submission, 2004. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/cwc/cwc-spec.pdf (October 2004)

[13] McGrew, A., Viega, J.: The Galois/Counter Mode of Operation, NIST submission, 2004. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf (October 2004)

[14] Hellström, H.: Propagating Cipher Feedback mode, 2001. Available at http://www.streamsec.com/pcfb1.pdf (October 2004)

[15] Schroeppel, R.: Cipher-State Mode of Operation for AES, NIST submission, 2004. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/cs/cs-spec.pdf (October 2004)

[16] Gligor, V., Donescu, P.: eXtended Cipher Block Chaining Encryption, 2000. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/xcbc/xcbc-spec.pdf (October 2004)

[17] Iwata, T., Kurosawa, K.: OMAC: One-key CBC MAC. NIST submission, 2002.
Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/omac/omac-spec.pdf (October 2004)

[18] Gladmann, B.: Authenticated encryption modes source codes, C++. Available at http://fp.gladman.plus.com/AES/modes.zip (October 2004)

[19] Krovetz, T.: OCB-AES-n reference code. Available at http://www.cs.ucdavis.edu/ rogaway/ocb/ocb-ref.tar (October 2004)

[20] Jonsson, J.: On the Security of CTR + CBC-MAC. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/ccm/ccm-ad1.pdf (October 2004)

[21] Kohno, T., Viega, J., Whiting, D.: The CWC authenticated encryption (associated data) mode. Available at http://eprint.iacr.org/2003/106/ (October 2004)

[22] McGrew, D., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation (Full Version). 2004. Available at http://eprint.iacr.org/2004/193/ (October 2004)

[23] Sander, T., Tschudin, Ch.: Protecting Agents From Malicious Hosts. Springer LNCS 1419, Berlin, 1998, pp. 44-60.

[24] Chow, S., Eisen, P., Johnson, H., van Oorschot, P.C.: White-Box Cryptography and an AES implementation. Springer LNCS 2595, Berlin, 2003, pp. 250-270.

[25] Chow, S., Eisen, P., Johnson, H., van Oorschot, P.C.: A White-Box DES Implementation for DRM Applications. Springer LNCS 2696, 2003, pp. 1-15.