ADDENDUM
XCBC Encryption with Authentication and XECB Authentication Modes

Submitter:
VDG Inc.
6009 Brookside Drive
Chevy Chase, Maryland 20815

tel. (301) 657-1959
fax. (301) 657-9021

Inventors:
Virgil D. Gligor and Pompiliu Donescu

Owner:
VDG Inc.
6009 Brookside Drive
Chevy Chase, Maryland 20815
March 28, 2001

In this Addendum, we present several of the modes discussed in the main submission paper "Fast Encryption and Authentication: The XCBC Encryption and XECB Authentication Modes," which can be found at *http://csrc.nist.gov/encryption/modes/*.) In particular, we summarize the properties of these modes in the format suggested by NIST. While this Addendum contains additional information regarding the proposed modes, it is not a substitute for the main submission paper, which also contains a detailed discussion of, and motivation, for the proposed modes.

1 Specification of the Stateless Encryption with Authentication XCBC-XOR Mode (XCBC$-XOR)

The encryption and decryption functions of the one-key stateless mode providing secrecy and authenticity, $\mathcal{E}{-}XCBC\$ - XOR^{F_K}(x)$ and $\mathcal{D}{-}XCBC\$ - XOR^{F_K}(y)$, are defined as follows.

```
function $\mathcal{E}$−XCBC$-XOR$^f(x)$
$r_0 \leftarrow \{0,1\}^l$
$y_0 = f(r_0);\ z_0 = f(r_0 + 1)$
if $|x_n| = l$ then {
    $Z = \overline{z_0}$ and $P = x$ }
else {
    $Z = z_0$ and $P = Pad(x)$ }
$P_{n+1} = Z$
for $i = 1, \cdot\ \cdot, n$ do {
    $P_{n+1} = P_{n+1} \oplus P_i$ }
for $i = 1, \cdot\ \cdot, n+1$ do {
    $z_i = f(P_i \oplus z_{i-1})$
    $y_i = z_i + i \times r_0$ }
return $y = y_0||y_1 y_2 \cdot\ \cdot y_n y_{n+1}$
```

```
function $\mathcal{D}$−XCBC$-XOR$^f(y)$
Parse $y$ as $y_0||y_1 \cdot\ \cdot y_n y_{n+1}$
$r_0 = f^{-1}(y_0);\ z_0 = f(r_0 + 1)$
for $i = 1, \cdot\ \cdot, n+1$ do {
    $z_i = y_i - i \times r_0$
    $P_i = f^{-1}(z_i) \oplus z_{i-1}$ }
$g = 0$
for $i = 1, \cdot\ \cdot, n$ do {
    $g = g \oplus P_i$ }
if $g \oplus \overline{z_0} = P_{n+1}$ {
    $x = P$ and
    return $x = x_1 x_2 \cdot\ \cdot x_n$ }
else {
    if $g \oplus z_0 = P_{n+1}$ {
        $x = Unpad(P)$ and
        return $x = x_1 x_2 \cdot\ \cdot x_n$ }
    else
        return $Null$ },
```

where $Null$ is an authenticity failure indicator. In this specification, $Pad(x) = x||10^i$ where $i = l - 1 - |x| \bmod l$ and $||$ is the concatenation operator; $Unpad(x)$ means removing the pad of the form 10^i from the last block of string x. It is clear that the computation of P_{n+1} at encryption and of g at decryption can proceed in the same loop as the enciphering, or correspondently, deciphering of the first n blocks.

An extended discussion of the properties of this mode, including its proof for the two-key variant, can be found at: *http://csrc.nist.gov/encryption/modes/*.)

1.1 Summary of Properties:

1. **Security Function**

   Authenticated encryption.

2. **Error Propagation**

The XCBC$-XOR mode does not allow the decryption, with non-negligible probability, of any ciphertext string where any bit change has been made. Without the authentication mechanism, the XCBC$ mode has similar error propagation characteristics to those of CBC (viz., section 10 below).

3. **Synchronization**

   The XCBC$-XOR mode does not allow the decryption, with non-negligible probability, of any ciphertext string where any bit change has been made. Without the authentication mechanism, the XCBC$ mode has similar synchronization characteristics to those of CBC (viz., section 10 below).

4. **Parallelizability**

   This mode is primarily intended for sequential operation. However, as specified in the main paper, this mode can be used in an interleaved-parallel manner. This enables incremental updates of encrypted data and out-of-order processing of different plaintext and ciphertext segments (i.e., specified sets of sequential blocks), each segment potentially having a different length.

5. **Keying Material**

   This mode requires a single key. However, in the main paper we illustrate a two-key version also. If the block cipher is AES, then $l = 128$ and the key length $|K| \in \{128, 192, 256\}$.

6. **Counter/IV/Nonce requirements**

   For its intended use, this mode requires an unpredictable IV, z_0, that is generated anew for each message encryption, whose value is not exported outside the mode (i.e., it is not output). This IV is generated by enciphering a variant of a per-message secret random value r_0 (i.e., enciphering $r_0 + 1$) with the same key as that used for plaintext block enciphering. Both the secret random value and the IV are of the same length as that of the block cipher (i.e., 128 bits for AES).

7. **Memory Requirements**

   The XCBC$-XOR mode requires six blocks for holding the information specific to this scheme, and the necessary memory for the block cipher with one key. Namely, the mode requires:

   - one block R for holding r_0;
   - one block E for holding the summation term $E = i \times r_0$;
   - three blocks for performing the enciphering of plaintext block P_i, namely, one block X for P_i, one block Z_OLD for z_{i-1} and one block Z_NEW for z_i. In this case, the algorithm for enciphering block P_i may proceed as follows:

     $$\begin{aligned} &Z\_NEW \quad F_K(X \oplus Z\_OLD); \\ &Z\_OLD \quad Z\_NEW; \\ &Z\_NEW \quad Z\_OLD + E; \end{aligned}$$

     i.e., after the execution of this part, block Z_NEW contains the value of the ciphertext block y_i that is further output. These blocks can also be used at the initialization phase as follows:

     $$\begin{aligned} &Z\_OLD \quad R + 1; \\ &Z\_NEW \quad F_K(Z\_OLD); \\ &Z\_OLD \quad Z\_NEW; \\ &Z\_NEW \quad F_K(R), \end{aligned}$$

where in the first line we compute $r_0 + 1$ and store it in block Z_OLD, then compute $z_0 = F_K(r_0 + 1)$ in block Z_NEW, then save z_0 in Z_OLD, and use Z_NEW for $y_0 = F_K(r_0)$ that is further output;

- one block W for the value of function $z_0 \oplus x_1 \oplus \quad \oplus x_{n+1}$.

The estimates provided herein are conservative, in the sense that standard optimization can be applied to reduce memory register requirements.

8. **Pre-processing Capability**

This mode requires a random value generator for creating the random value r_0, and a block cipher invocation for generating the IV (z_0). A block cipher invocation is also required for enciphering r_0 (i.e., to obtain the ciphertext y_0 transmitted to the message receiver).

9. **Message Length Requirements**

This mode requires padding. A padding function that does not require an extra block cipher-invocation for plaintext messages comprising an integer number of block, does not require an extra enciphering key, and maintains the unpredictability of the redundancy function $g(x)$ representing a mode-appended plaintext block, is specified.

10. **Ciphertext Expansion**

The XCBC\$-XOR outputs $n+2$ ciphertext blocks, where n is the number of plaintext blocks obtained after padding.

11. **Other Characteristics**

- Alternatively, the computation of the ciphertext blocks y_i can be done as follows:

$$y_i = z_i \oplus (i \times r_0).$$

- The related stateless encryption-only mode XCBC\$ may also be considered, where the encryption and decryption functions of the one-key stateless mode, $\mathcal{E}-XCBC\$^{F_K}(x)$ and $\mathcal{D}-XCBC\$^{F_K}(y)$, are defined as follows.

```
**function** $\mathcal{E}$−XCBC\$$^f(x)$
$r_0 \quad 0, 1\}^l$
$y_0 = f(r_0);\; z_0 = f(r_0 + 1)$
**for** $i = 1, \quad , n$ **do** {
$z_i = f(x_i \oplus z_{i-1})$
$y_i = z_i + i \times r_0$ }
**return** $y = y_0||y_1 y_2 \quad y_n$
```

```
**function** $\mathcal{D}$−XCBC\$$^f(y)$
Parse $y$ as $y_0||y_1 \quad \leftarrow \{y_n$
$r_0 = f^{-1}(y_0);\; z_0 = f(r_0 + 1)$
**for** $i = 1, \quad , n$ **do** {
$z_i = y_i - i \times r_0$
$x_i = f^{-1}(z_i) \oplus z_{i-1}$ }
**return** $x = x_1 x_2 \quad x_n$
```

Error Propagation If used exclusively for encryption, namely without authentication, this mode has similar error propagation properties to those of the basic CBC mode. At *decryption*, a single bit error in ciphertext block y_i causes the of the corresponding hidden ciphertext block z_i to contain one or more bit errors thereby affecting the deciphering of both ciphertext blocks y_i and y_{i+1}. Plaintext block x_i recovered from hidden ciphertext block z_i is totally random (depending on the properties of the block cipher), whereas the recovered plaintext x_{i+1} contains the bit errors exactly in the same bit positions where z_i did.

At *encryption*, a single bit error in plaintext block $x_i, i \geq 1$ causes alterations in all subsequent ciphertext blocks.

Synchronization If used exclusively for encryption, namely without authentication, this mode is self-synchronizing in the sense that if a bit error occurs (or bit errors occur) in ciphertext block y_i and not in y_{i+1}, ciphertext block y_{i+2} is correctly deciphered to x_{i+2}, and all further ciphertext blocks are deciphered correctly.

The XCBC$ is primarily intended for sequential operation, but can be used in a parallel-interleaved manner, in a similar way to XCBC$-XOR.

The XCBC$ mode requires 5 blocks (i.e., the blocks needed by XCBC$-XOR without the one storing the value of function $z_0 \oplus x_1 \oplus \quad \oplus x_{n+1}$.

The XCBC$ mode requires padding. Industry-standard padding schemes can be used.

1.2 Test Vectors

Please see discussion in Appendix 1.

1.3 Performance estimates

The XCBC$-XOR mode requires $n + 3$ block cipher invocations, where n is the number of plaintext blocks after padding.

1.4 Intellectual Property Statement

Presented in Appendix 2.

2 Specification of the Stateful-Sender Encryption with Authentication XCBC-XOR Mode (XCBCC-XOR)

The encryption and decryption functions of the one-key stateful-sender mode, providing secrecy and authenticity, $\mathcal{E}{-}XCBCC^{F_K}(x, ctr)$ and $\mathcal{D}{-}XCBCC^{F_K}(y)$, are defined as follows.

```
function $\mathcal{E}-\mathrm{XCBCC}-XOR^f(x, ctr)$
$r_0 = f(ctr);\ z_0 = f(r_0 + 1)$
if $|x_n| = l$ then {
    $Z = \overline{z_0}$ and $P = x$ }
else {
    $Z = z_0$ and $P = Pad(x)$ }
$P_{n+1} = Z$
for $i = 1, \quad , n$ do {
    $P_{n+1} = P_{n+1} \oplus P_i$ }
for $i = 1, \quad , n + 1$ do {
    $z_i = f(P_i \oplus z_{i-1})$
    $y_i = z_i + i \times r_0$ }
$ctr' \quad ctr + 1$
$y = ctr||y_1 y_2 \quad y_n y_{n+1}$
return $y$
```

```
function $\mathcal{D}-\mathrm{XCBCC}-XOR^f(y)$
Parse $y$ as $ctr||y_1 \quad y_n y_{n+1}$
$r_0 = f(ctr);\ z_0 = f(r_0 + 1)$
for $i = 1, \quad , n$ do {
    $z_i = y_i - i \times r_0$
    $P_i = f^{-1}(z_i) \oplus z_{i-1}$ }
$g = 0$
for $i = 1, \quad , n$ do {
    $g = g \oplus P_i$ }
if $g \oplus \overline{z_0} = P_{n+1}$ {
    $x = P$ and
    return $x = x_1 x_2 \quad x_n$ }
else {
    if $g \oplus z_0 = P_{n+1}$ {
        $x = Unpad(P)$ and
        return $x = x_1 x_2 \quad x_n$ }
    else
        return $Null$ },
```

where $Null$ is an authenticity failure indicator. In this specification, $Pad(x) = x||10^i$ where $i = l - 1 - |x| \bmod l$ and $||$ is the concatenation operator; $Unpad(x)$ means removing the pad of the form 10^i from the last block of string x. It is clear that the computation of P_{n+1} at encryption and of g at decryption can proceed in the same loop as the enciphering, or correspondently, deciphering of the first n blocks.

An extended discussion of the properties of this mode, for the two-key variant, can be found at: *http://csrc.nist.gov/encryption/modes/*.)

2.1 Summary of Properties:

1. **Security Function**

   Authenticated encryption.

2. **Error Propagation**

   Same as for the stateless XCBC-XOR mode (XCBC$-XOR).

3. **Synchronization**

   Same as for the stateless XCBC-XOR mode (XCBC$-XOR).

4. **Parallelizability**

   Same as for the stateless XCBC-XOR mode (XCBC$-XOR).

5. **Keying Material**

   This mode requires a single key. However, in the main paper we illustrate a two-key version also. If the block cipher is AES, then $l = 128$ and the key length $|K| \in \{128, 192, 256\}$.

6. **Counter/IV/Nonce requirements**

   The sender maintains a counter *ctr* that is increased by one for each plaintext encryption.

7. **Memory Requirements**

   This mode requires seven blocks for holding the information specific to this scheme, and the necessary memory for the block cipher with one key. This mode uses the six blocks shown for the XCBC\$-XOR mode and a block C for the counter.

   If the counter is read and output immediately, then the memory requirements are the same as for the stateless XCBC-XOR mode (XCBC\$-XOR). To see this, we show an implementation of the initialization phase, where the counter is first stored in Z_OLD:

$$\begin{array}{ll} R & F_K(Z\_OLD); \\ Z\_OLD & R+1; \\ Z\_NEW & F_K(Z\_OLD); \\ Z\_OLD & Z\_NEW; \\ Z\_NEW & F_K(R), \end{array}$$

   where in the first line we compute $r_0 = F_K(ctr)$, in the second line we compute $r_0 + 1$ and store it in block Z_OLD, then compute $z_0 = F_K(r_0 + 1)$ in block Z_NEW, then save z_0 in Z_OLD, and use Z_NEW for $y_0 = F_K(r_0)$. In this implementation, the counter that is initially in Z_OLD is output immediately after r_0 is computed in R, and hence, block Z_OLD can be reused.

   The estimates provided herein are conservative, in the sense that standard optimization can be applied to reduce memory register requirements.

8. **Pre-processing Capability**

   This mode requires maintaining a counter *ctr* and a block cipher invocation to create the random value r_0; also, a block cipher invocation is necessary for generating the IV (z_0).

9. **Message Length Requirements**

   Same as for the stateless XCBC-XOR mode (XCBC\$-XOR).

10. **Ciphertext Expansion**

    The XCBCC-XOR mode outputs the value of the counter *ctr* and $n + 1$ ciphertext blocks, where n is the number of plaintext blocks obtained after padding.

11. **Other Characteristics**

    - Alternatively, the computation of the ciphertext blocks y_i can be done as follows:

$$y_i = z_i \oplus (i \times r_0).$$

    - The related stateful-sender encryption mode XCBCC may also be considered, where the encryption and decryption functions of the one-key stateful-sender mode, $\mathcal{E}{-}XCBCC^{F_K}(x, ctr)$ and $\mathcal{D}{-}XCBCC^{F_K}(y)$, are defined as follows.

```
**function** $\mathcal{E}-\text{XCBCC}^f(x, ctr)$
$r_0 = f(ctr);\ z_0 = f(r_0 + 1)$
**for** $i = 1, \quad , n$ **do** {
    $z_i = f(x_i \oplus z_{i-1})$
    $y_i = z_i + i \times r_0$ }
$ctr' \quad ctr + 1$
$y = ctr||y_1 y_2 \quad y_n$
**return** $y$
```

```
**function** $\mathcal{D}-\text{XCBCC}^f(y)$
Parse $y$ as $ctr||y_1 \quad y_n$
$r_0 = f(ctr);\ z_0 = f(r_0 + 1)$
**for** $i = 1, \quad , n$ **do** {
    $z_i = y_i - i \times r_0$
    $x_i = f^{-1}(z_i) \oplus z_{i-1}$ }
**return** $x = x_1 x_2 \quad x_n$
```

This mode provides encryption only.

2.2 Test Vectors

Please see discussion in Appendix 1.

2.3 Performance estimates

When used with its specified authentication function, this encryption mode requires $n + 3$ block cipher invocations, where n is the number of plaintext blocks after padding.

2.4 Intellectual Property Statement

Presented in Appendix 2.

3 Specification of the Stateful Encryption with Authentication XCBC-XOR Mode (XCBCS-XOR)

Let IV be a random and uniformly distributed variable that is part of the keying state shared by the sender and receiver. The encryption and decryption functions of the one-key stateful mode providing secrecy and authenticity, $\mathcal{E}-XCBCS-XOR^{F_K}(x)$ and $\mathcal{D}-XCBCS-XOR^{F_K}(y)$, are defined as follows.

```
function $\mathcal{E}-$XCBCS-XOR$^f(x)$
$r_0 \quad 0,1\}^l$
$y_0 = f(r_0);\ z_0 = IV + r_0$
if $|x_n| = l$ then {
    $Z = \overline{z_0}$ and $P = x$ }
else {
    $Z = z_0$ and $P = Pad(x)$ }
$P_{n+1} = Z$
for $i = 1, \quad , n$ do {
    $P_{n+1} = P_{n+1} \oplus P_i$ }
for $i = 1, \quad , n + 1$ do {
    $z_i = f(P_i \oplus z_{i-1})$
    $y_i = z_i + i \times r_0$ }
return $y = y_0||y_1y_2 \quad y_ny_{n+1}$
```

```
function $\mathcal{D}-$XCBCS-XOR$^f(y)$
Parse $y$ as $y_0||y_1 \quad y_ny_{n+1}$
$r_0 = f^{-1}(y_0);\ z_0 = IV + r_0$
for $i = 1, \quad , n + 1$ do {
    $z_i = y_i - i \times r_0$
    $P_i = f^{-1}(z_i) \oplus z_{i-1}$ }
$g = 0$
for $i = 1, \quad , n$ do {
    $g = g \oplus P_i$ }
if $g \oplus \overline{z_0} = P_{n+1}$ {
    $x = P$ and
    return $x = x_1x_2 \quad x_n$ }
else {
    if $g \oplus z_0 = P_{n+1}$ {
        $x = Unpad(P)$ and
        return $x = x_1x_2 \quad x_n$ }
    else
        return $Null$ },
```

where $Null$ is an authenticity failure indicator. In this specification, $Pad(x) = x||10^i$ where $i = l - 1 - |x| \bmod l$ and $||$ is the concatenation operator; $Unpad(x)$ means removing the pad of the form 10^i from the last block of string x. It is clear that the computation of P_{n+1} at encryption and of g at decryption can proceed in the same loop as the enciphering, or correspondently, deciphering of the first n blocks.

An extended discussion of the properties of this mode can be found at:
http://csrc.nist.gov/encryption/modes/.)

3.1 Summary of Properties:

1. **Security Function**

   Authenticated encryption.

2. **Error Propagation**

   Same as for the stateless XCBC mode (XCBC$).

3. **Synchronization**

   Same as for the stateless XCBC-XOR mode (XCBC$-XOR).

4. **Parallelizability**

   Same as for the stateless XCBC-XOR mode (XCBC$-XOR).

5. **Keying Material**

   This mode requires a single key. If the block cipher is AES, then $l = 128$ and the key length $|K| \in \{128, 192, 256\}$.

6. **Counter/IV/Nonce requirements**

For its intended use, this mode requires an unpredictable z_0, that is generated anew for each message encryption, whose value is not exported outside the mode (i.e., it is not output). z_0 is generated by adding to a per-key random number IV a per-message secret random value r_0. The per-key random number IV is secret and shared between the sender and the receiver, or can be derived from the secret shared key using well-known key separation techniques. Both r_0 and z_0 are of the same length as that of the block cipher (i.e., 128 bits for AES).

7. **Memory Requirements**

   This mode requires seven blocks for holding the information specific to this scheme, and the necessary memory for the block cipher with one key. This mode uses the six blocks shown for the XCBC$-XOR mode and a block C for IV.

   The estimates provided herein are conservative, in the sense that standard optimization can be applied to reduce memory register requirements.

8. **Pre-processing Capability**

   This mode requires a random value generator for creating the random value r_0. A block cipher invocation is also required for enciphering r_0 (i.e., to obtain the ciphertext y_0 transmitted to the message receiver).

9. **Message Length Requirements**

   Same as for the stateless XCBC-XOR mode (XCBC$-XOR).

10. **Ciphertext Expansion**

    Same as for the stateless XCBC-XOR mode (XCBC$-XOR).

11. **Other Characteristics**

    - Alternatively, the computation of the ciphertext blocks y_i can be done as follows:

      $$y_i = z_i \oplus (i \times r_0).$$

    - Let IV be a random and uniformly distributed variable that is part of the keying state shared by the sender and receiver.
      $\mathcal{E}-XCBCS^{F_K}(x)$ and $\mathcal{D}-XCBCS^{F_K}(y)$, are defined as follows.

```
function $\mathcal{E}$−XCBCS$^f(x)$
$r_0 \quad 0,1\}^l$
$y_0 = f(r_0);\ z_0 = IV + r_0$
for $i = 1, \quad ,n$ do {
$z_i = f(x_i \oplus z_{i-1})$
$y_i = z_i + i \times r_0$ }
return $y = y_0||y_1y_2 \quad y_n$
```

```
function $\mathcal{D}$−XCBCS$^f(y)$
Parse $y$ as $y_0||y_1 \quad \leftarrow y_n\{$
$r_0 = f^{-1}(y_0);\ z_0 = IV + r_0$
for $i = 1, \quad ,n$ do {
$z_i = y_i - i \times r_0$
$x_i = f^{-1}(z_i) \oplus z_{i-1}$ }
return $x = x_1x_2 \quad x_n$
```

3.2 Test Vectors

Please see discussion in Appendix 1.

3.3 Performance estimates

When used with its specified authentication function, this encryption mode requires $n + 2$ block cipher invocations, where n is the number of plaintext blocks after padding.

3.4 Intellectual Property Statement

Presented in Appendix 2.

4 Specification of the Stateful Encryption with Authentication XECB-XOR Mode (XECBS-XOR)

Let R, R^* be two random, uniformly distributed and independent blocks that are part of the keying state shared by the sender and receiver. The encryption and decryption functions of the one-key stateful mode providing secrecy and authenticity, $\mathcal{E}{-}XECBS-XOR^{F_K}(x)$ and $\mathcal{D}{-}XECBS-XOR^{F_K}(y)$, are defined as follows.

```
**function** $\mathcal{E}$−XECBS-XOR$^f(x)$
**if** $|x_n| = l$ **then** {
$Z = \overline{R}$ and $P = x$ }
**else** {
$Z = R$ and $P = Pad(x)$ }
**for** $i = 1, \quad , n$ **do** {
$z_i = f(P_i + ctr \times R + i \times R^*)$
$y_i = z_i + ctr \times R + i \times R^*$ }
$P_{n+1} = 0$
**for** $i = 1, \quad , n$ **do** {
$P_{n+1} = P_{n+1} \oplus P_i$ }
$z_{n+1} = f(P_{n+1} + ctr \times Z)$
$y_{n+1} = z_{n+1} + ctr \times R + (n+1) \times R^*$ }
$ctr' \quad ctr + 1$
**return** $y = y_1 y_2 \quad y_n y_{n+1}$
```

```
**function** $\mathcal{D}$−XECBS-XOR$^f(y)$
Parse $y$ as $ctr||y_1 \quad y_n y_{n+1}$
**if** $ctr > q_e$ **then return** $Null$
**for** $i = 1, \quad , n$ **do** {
$z_i = y_i - ctr \times R - i \times R^*$
$P_i = f^{-1}(z_i) - ctr \times R - i \times R^*$ }
$z_{n+1} = y_{n+1} - ctr \times R - (n+1) \times R^*$
$P_{n+1} = f^{-1}(z_{n+1}) - ctr \times R$ }
$g = 0$
**for** $i = 1, \quad , n$ **do** {
$g = g \oplus P_i$ }
**if** $g = P_{n+1}$ {
$x = Unpad(P)$ and
**return** $x = x_1 x_2 \quad x_n$ }
**else** {
**if** $g + ctr \times R - ctr \times \overline{R} = P_{n+1}$ {
$x = P$ and
**return** $x = x_1 x_2 \quad x_n$ }
**else**
**return** $Null$ },
```

where $Null$ is an authenticity failure indicator. In this specification, $Pad(x) = x||10^i$ where $i = l - 1 - |x| \bmod l$ and $||$ is the concatenation operator; $Unpad(x)$ means removing the pad of the form 10^i from the last block of string x. The counter ctr is initialized to 1 and increased by 1 on every message encryption up to the maximum allowable number of message encryptions q_e. $\overline{R}$ is the bitwise complement of R.

An extended discussion of the properties of this mode can be found at:
http://csrc.nist.gov/encryption/modes/.)

4.1 Summary of Properties:

1. **Security Function**

   Authenticated encryption.

2. **Error Propagation**

   None.

3. **Synchronization**

   The XECB-XOR mode does not allow the decryption, with non-negligible probability, of any ciphertext string where any bit change has been made.

4. **Parallelizability**

   Fully parallel.

5. **Keying Material**

   This mode requires a single key. If the block cipher is AES, then $l = 128$ and the key length $|K| \in \{128, 192, 256\}$.

6. **Counter/IV/Nonce requirements**

   The sender maintains a counter *ctr* that is increased by one for each plaintext encryption and is initialized to 1. The per-key values R and R^* are secret, random, independent, and are shared by the sender and the receiver, or can be derived from the secret shared key using well-known key separation techniques. Both R and R^* are of the same length as that of the block cipher (i.e., 128 bits for AES).

7. **Memory Requirements**

   This mode requires ten blocks for holding the information specific to this scheme, and the necessary memory for the block cipher with one key, namely:

   - one block for holding *ctr*;
   - one block for holding R;
   - one block for holding R^*;
   - one block for holding Z;
   - two blocks for the terms $ctr \times R$ and $i \times R^*$ (note that the block holding $ctr \times R$ can be used to hold $ctr \times Z$ for the last block encryption);
   - four blocks for performing the enciphering of plaintext block P_i, namely, one block for P_i, one block for $P_i + ctr \times R + i \times R^*$; one block for z_i, and one block for y_i.

   The estimates provided herein are conservative, in the sense that standard optimization can be applied to reduce memory register requirements.

8. **Pre-processing Capability**

   This mode requires maintaining a counter *ctr*.

9. **Message Length Requirements**

   This mode requires padding. A padding function that does not require an extra block cipher-invocation for plaintext messages comprising an integer number of block, and does not require an extra enciphering key, is specified.

10. **Ciphertext Expansion**

    The XCBC$-XOR outputs $n+1$ ciphertext blocks and the counter ctr where n is the number of plaintext blocks obtained after padding.

11. **Other Characteristics**

    Other variants of the stateful XECB-XOR mode can be obtained by using other formulae for the encryption of the redundancy block, such as:

    $$y_{n+1} = f(x_{n+1} + ctr \times Z + (n+1) \times R^*) + ctr \times R,$$

    where $Z = \overline{R}$ for unpadded messages and $Z = R$ for padded messages.

    Stateless architecture-independent parallel modes and stateful-sender architecture-independent parallel modes can be specified in the same manner as those for the XCBC modes; for example, R and R^* can be derived from the l-bit random number number r_0 (e.g., $R = r(r_0+1)$ and $R^* = f(r_0+2)$), and, in the stateful-sender $r_0 = f(ctr)$ where ctr is an l-bit counter initialized to a constant such as -1.

4.2 Test Vectors

Please see discussion in Appendix 1.

4.3 Performance estimates

When used with its specified authentication function, this encryption mode requires $n+1$ block cipher invocations, where n is the number of plaintext blocks after padding, and one block-cipher invocation throughput.

4.4 Intellectual Property Statement

Presented in Appendix 2.

5 Specification of the Stateless XECB Authentication Mode (XECB$-MAC)

The signing and authentication functions of the one-key stateless mode, are as follows:

Stateless XECB-MAC Mode (XECB$-MAC)

```
**function** Sign-XECB$-MAC$^f(x)$
$r_0 \quad 0,1\}^l$
$y_0 = f(r_0), z_0 = f(r_0 + 1)$
**if** $|x_n| = l$ **then**
$Z = \overline{z_0}$ and $P = x$
**else**
$Z = z_0$ and $P = Pad(x)$
$P_{n+1} = Z$
**for** $i = 1, \quad , n + 1$ **do** $\{$
$y_i = f(P_i + i \times y_0)$ $\}$
$w = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$
**return** $(r_0, w)$
```

```
**function** Verify-XECB$-MAC$^f(x, r_0, w)$
$y_0 = f(r_0), \leftarrow z_0 \{= f(r_0 + 1)$
**if** $|x_n| = l$ **then**
$Z = \overline{z_0}$ and $P = x$
**else**
$Z = z_0$ and $P = Pad(x)$
$P_{n+1} = Z$
**for** $i = 1, \quad , n + 1$ **do** $\{$
$y_i = f(P_i + i \times y_0)$ $\}$
$w' = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$
**if** $w = w'$ **then return** $1$
**else return** $0$.
```

In this specification, $Pad(x) = x||10^i$ where $i = l - 1 - |x| \bmod l$ and $||$ is the concatenation operator.

An extended discussion of the properties of this mode for the two-key variant can be found at: *http://csrc.nist.gov/encryption/modes/*.)

5.1 Summary of Properties:

1. **Security Function**

   Authentication.

2. **Error Propagation**

   Not applicable.

3. **Synchronization**

   Not applicable.

4. **Parallelizability**

   This mode is intended to be used for either sequential or parallel implementation.

5. **Keying Material**

   This mode requires a single key. However, in the main paper we illustrate a two-key version also. If the block cipher is AES, then $l = 128$ and the key length $|K| \in \{128, 192, 256\}$.

6. **Counter/IV/Nonce requirements**

   For its intended use, this mode requires an unpredictable z_0, that is generated anew for each message encryption, whose value is not exported outside the mode (i.e., it is not output). This z_0 is generated by enciphering a variant of a per-message secret random value r_0 (i.e., enciphering $r_0 + 1$) with the same key as that used for the enciphering of the randomized plaintext block, namely $P_i + i \times y_0$. Both y_0 and z_0 are of the same length as that of the block cipher (i.e., 128 bits for AES).

7. **Memory Requirements**

   The XECB$-MAC mode requires eight blocks for holding the information specific to this scheme, and the necessary memory for the block cipher with one key. Namely, the mode requires:

- one block for holding r_0;
- one block for holding y_0;
- one block for holding z_0 and later the variable Z;
- one block for holding the term $i \times y_0$;
- three blocks for performing the enciphering of plaintext block P_i, namely, one block for P_i, one block for $P_i + i \times y_0$ and one block for y_i.
- one block for storing the tag $y_1 \oplus \quad y_{n+1}$.

The estimates provided herein are conservative, in the sense that standard optimization can be applied to reduce memory register requirements.

8. **Pre-processing Capability**

   This mode requires a random value generator for creating the random value r_0, and two block cipher invocations for generating y_0 and z_0.

9. **Message Length Requirements**

   This mode requires padding. A padding function that does not require an extra block cipher-invocation for plaintext messages comprising an integer number of block and does not require an extra enciphering key, is specified.

10. **Ciphertext Expansion**

    Not applicable.

11. **Other Characteristics**

    - An alternative to the computation of the tag w uses the addition modulo $2^L - 1$, i.e.,

    $$w = y_1 + \quad + y_n + y_{n+1} \bmod 2^L - 1.$$

    - Another alternative to the computation of the tag w uses the subtraction modulo $2^L - 1$, i.e.,

    $$w = y_1 - \quad - y_n - y_{n+1} \bmod 2^L - 1$$

5.2 Test Vectors

Please see discussion in Appendix 1.

5.3 Performance estimates

This authentication mode requires $n + 3$ block cipher invocations, where n is the number of plaintext blocks after padding. The throughput of the stateless XECB mode (XECB$-MAC) is equivalent to three sequential block-cipher invocations (to compute y_0, z_0 and y_{n+1} that takes as input $P_{n+1} + (n + 1) \times y_0$).

5.4 Intellectual Property Statement

Presented in Appendix 2.

6 Specification of the Stateful-Sender XECB Authentication Mode (XECBC-MAC)

The signing and authentication functions of the one-key stateful-sender mode, are as follows:

Stateful-Sender XECB-MAC Mode (XECBC-MAC)

```
function Sign-XECBC-MAC$^f(ctr, x)$
$y_0 = f(ctr), z_0 = f(y_0 + 1)$
if $|x_n| = l$ then
$Z = \overline{z_0}$ and $P = x$
else
$Z = z_0$ and $P = Pad(x)$
$P_{n+1} = Z$
for $i = 1, \quad , n + 1$ do {
$y_i = f(P_i + i \times y_0)$ }
$w = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$
$ctr' \quad ctr + 1$
return $(ctr, w)$
```

```
function Verify-XECBC-MAC$^f(x, ctr, w)$
$y_0 = f(ctr), z_0 = f(y_0 + 1)$
if $|x_n| = l$ then
$Z = \overline{z_0}$ and $P = x$
else
$Z = z_0$ and $P = Pad(x)$
$P_{n+1} = Z$
for $i = 1, \quad , n + 1$ do {
$y_i = f(P_i + i \times y_0)$ }
$w' = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$
if $w = w'$ then return 1
else return 0.
```

Note that ctr' represents the updated ctr value.

In this specification, $Pad(x) = x||10^i$ where $i = l - 1 - |x| \bmod l$ and $||$ is the concatenation operator.

An extended discussion of the properties of this mode, including its proof for the two-key variant, can be found at: *http://csrc.nist.gov/encryption/modes/*.)

6.1 Summary of Properties:

1. **Security Function**

   Authentication.

2. **Error Propagation**

   Not applicable.

3. **Synchronization**

   Not applicable.

4. **Parallelizability**

   This mode is intended to be used for either sequential or parallel implementation.

5. **Keying Material**

   This mode requires a single key. However, in the main paper we illustrate a two-key version also. If the block cipher is AES, then $l = 128$ and the key length $|K| \in \{128, 192, 256\}$.

6. **Counter/IV/Nonce requirements**

The sender maintains a counter *ctr* that is increased by one for each plaintext authentication. The counter is used to generate a per-message secret random value y_0 with the same key as that used for plaintext block enciphering.

For its intended use, this mode requires an unpredictable z_0, that is generated anew for each message encryption, whose value is not exported outside the mode (i.e., it is not output). This z_0 is generated by enciphering a variant of a per-message secret random value y_0 (i.e., enciphering $y_0 + 1$) with the same key as that used for the enciphering of the randomized plaintext block, namely $P_i + i \times y_0$. Both y_0 and z_0 are of the same length as that of the block cipher (i.e., 128 bits for AES).

7. **Memory Requirements**

   The XECBC-MAC mode requires eight blocks for holding the information specific to this scheme, and the necessary memory for the block cipher with one key. Namely, the mode requires:

   - one block for holding *ctr*;
   - one block for holding z_0;
   - one block for holding z_0 and later Z;
   - one block for holding the term $i \times y_0$;
   - three blocks for performing the enciphering of plaintext block P_i, namely, one block for P_i, one block for $P_i + i \times y_0$ and one block for y_i.
   - one block for storing the tag $y_1 \oplus \quad y_{n+1}$.

   The estimates provided herein are conservative, in the sense that standard optimization can be applied to reduce memory register requirements.

8. **Pre-processing Capability**

   This mode requires maintaining a counter *ctr* and and two block cipher invocations for generating y_0 and z_0.

9. **Message Length Requirements**

   This mode requires padding. A padding function that does not require an extra block cipher-invocation for plaintext messages comprising an integer number of block and does not require an extra enciphering key, is specified.

10. **Ciphertext Expansion**

    Not applicable.

11. **Other Characteristics**

    - An alternative to the computation of the tag w uses the addition modulo $2^L - 1$, i.e.,

      $$w = y_1 + \qquad + y_n + y_{n+1} \bmod 2^L - 1.$$

    - Another alternative to the computation of the tag w uses the subtraction modulo $2^L - 1$, i.e.,

      $$w = y_1 - \qquad - y_n - y_{n+1} \bmod 2^L - 1$$

6.2 Test Vectors

Please see discussion in Appendix 1.

6.3 Performance estimates

This authentication mode requires $n+3$ block cipher invocations, where n is the number of plaintext blocks after padding. The throughput of the stateful-sender XECB mode (XECBC-MAC) is equivalent to three sequential block-cipher invocations (to compute y_0, z_0 and y_{n+1} that takes as input $P_{n+1} + (n+1) \times y_0$).

6.4 Intellectual Property Statement

Presented in Appendix 2.

7 Specification of the Stateful XECB Authentication Mode (XECBS-MAC)

The signing and authentication functions of the one-key stateful mode, are as follows:

Stateful XECB-MAC Mode (XECBS-MAC)
Let R, R^* be two random, uniformly distributed and independent blocks that are part of the keying state shared by the sender and receiver.

```
function Sign-XECBS-MAC$^f(ctr, x)$
if $|x_n| = l$ then {
$Z = \overline{R}$ and $P = x$ }
else {
$Z = R$ and $P = Pad(x)$ }
for $i = 1, \quad ,n$ do {
$y_i = f(P_i + ctr \times Z + i \times R^*)$ }
$w = y_1 \oplus \quad \oplus y_n$
$ctr' \quad ctr + 1$
return $(ctr, w)$
```

```
function Verify-XECBS-MAC$^f(x, ctr, w)$
if $ctr > q_s$ then return $0$
if $|x_n| = l$ then {
$Z = \overline{R}$ and $P = x$ }
else {
$Z = R$ and $P = Pad(x)$ }
for $i = 1, \quad ,n$ do {
$y_i = f(P_i + ctr \times Z + i \times R^*)$ }
$w' = y_1 \oplus \quad \oplus y_n$
if $w = w'$ then return $1$
else return $0$.
```

Note that *ctr'* represents the updated *ctr* value. In this specification, $Pad(x) = x||10^i$ where $i = l - 1 - |x| \bmod l$ and $||$ is the concatenation operator. $\overline{R}$ is the bitwise complement of R.

An extended discussion of the properties of this mode, including its proof, can be found at:
http://csrc.nist.gov/encryption/modes/.)

7.1 Summary of Properties:

1. **Security Function**

   Authentication.

2. **Error Propagation**

   Not applicable.

3. **Synchronization**

   Not applicable.

4. **Parallelizability**

   This mode is intended to be used for either sequential or parallel implementation.

5. **Keying Material**

   This mode requires a single key. If the block cipher is AES, then $l = 128$ and the key length $|K| \in \{128, 192, 256\}$.

6. **Counter/IV/Nonce requirements**

   The sender maintains a counter *ctr* that is increased by one for each plaintext encryption and is initialized to 1. The per-key values R and R^* are secret, random, independent, and are shared by the sender and the receiver, or can be derived from the secret shared key using well-known key separation techniques. Both R and R^* are of the same length as that of the block cipher (i.e., 128 bits for AES).

7. **Memory Requirements**

   The XECBS-MAC mode requires ten blocks for holding the information specific to this scheme, and the necessary memory for the block cipher with one key. Namely, the mode requires:

   - one block for holding *ctr*;
   - one block for holding R;
   - one block for holding R^*;
   - one block for holding Z;
   - two blocks for the terms $ctr \times Z$ and $i \times R^*$;
   - three blocks for performing the enciphering of plaintext block P_i, namely, one block for P_i, one block for $P_i + ctr \times Z + i \times R^*$; one block for y_i.
   - one block for storing the tag $y_1 \oplus \quad y_{n+1}$.

   The estimates provided herein are conservative, in the sense that standard optimization can be applied to reduce memory register requirements.

8. **Pre-processing Capability**

   This mode requires maintaining a counter *ctr*.

9. **Message Length Requirements**

   This mode requires padding. A padding function that does not require an extra block cipher-invocation for plaintext messages comprising an integer number of block and does not require an extra enciphering key, is specified.

10. **Ciphertext Expansion**

    Not applicable.

11. **Other Characteristics**

    - An alternative to the computation of the tag w uses the addition modulo $2^L - 1$, i.e.,

$$w = y_1 + \quad + y_n + y_{n+1} \bmod 2^L - 1.$$

- Another alternative to the computation of the tag w uses the subtraction modulo $2^L - 1$, i.e.,

$$w = y_1 - \qquad - y_n - y_{n+1} \bmod 2^L - 1$$

7.2 Test Vectors

Please see discussion in Appendix 1.

7.3 Performance estimates

This authentication mode requires n block cipher invocations, where n is the number of plaintext blocks after padding. The throughput of the stateful XECB mode (XECBS-MAC) is equivalent to only one block-cipher invocation.

7.4 Intellectual Property Statement

Presented in Appendix 2.

Appendix 1: Test Vectors

VDG Inc believes that the correct implementation of a mode is essential to its secure operation, and that reasonable assurance of correct implementation should be provided consistent with best industry practices.

The testing of the mode requires that test conditions, test data, and coverage analysis be provided for the systematic exercise of the mode's elements. For AES invocations, specific AES test vectors should be used that are developed specifically for AES. (The test vectors for the modes should be independent of the underlying block cipher.) Nevertheless, test conditions, data, and coverage analysis should be provided to indicate the proper use of the block cipher (i.e., AES) is made.

Specific test conditions and data for the randomness source used should be used for all stateless modes. (These should comply with tests for randomness specified by FIPS 140-1.) Furthermore, test conditions, test data, and coverage analysis should be provided to indicate that the source of randomness used by the mode is protected (e.g., confidentiality, integrity).

Furthermore, test conditions, test data, and coverage analysis is required to verify that the mode implementation protects other elements of the mode that are identified as secret and that cannot be modified from outside the mode itself. (e.g., blocks that are required to remain unpredictable to an adversary, such as the initialization vectors, z_0). The counters used in these modes are output in clear. Nevertheless, test conditions and data should be provided which show that these counters are protected from arbitrary modification or substitution.

If any mode of authenticated encryption or of authentication proposed by VDG Inc is adopted as a standard, VDG Inc agrees to provide reference implementations of this mode on three separate system platforms, and the test conditions, data (i.e., test vectors) and their coverage analysis necessary for specification-compliance testing. Further penetration analysis exercises will be conducted on specific reference implementations.

Appendix 2: Intellectual Property Statement

VDG Inc submitted a patent application covering the XCBC modes on January 31, 2000. Certain parts of this application may be relevant to other modes submitted. These parts include, but are not restricted to, the randomization of the output of an encryption mode to produce other modes that provide authenticated encryption, the nature of the randomization operations, and of the randomization sequences themselves.

VDG Inc submitted another patent application covering the XECB authentication modes on March 31, 2000. Certain parts of this application may also be relevant to other modes submitted. These parts include, but are not restricted to, the randomization of the input of an encryption mode to produce other modes that provide message authentication, the nature of the randomization operations, and of the randomization sequences themselves.

On August 24, 2000, VDG Inc submitted another patent application covering, among other aspects, fully parallelizable modes that provide authenticated encryption using randomization sequences whose elements are not pairwise-independent. Certain parts of this application may be relevant to other modes submitted.

If a mode submitted by VDG Inc is adopted as a standard by NIST, VDG Inc agrees, upon request, to grant non-exclusive license under the scope of the specific patent covering that mode on a nondiscriminatory basis and on reasonable terms and conditions including its then current royalty rates and provided that a similar grant under licensee's patents within the scope of the license granted to licensee is made available, upon request, to VDG Inc.