AUTHENTICATED ENCRYPTION
We have looked at methods to provide privacy and integrity/authenticity separately:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Primitive</th>
<th>Security notions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data privacy</td>
<td>symmetric encryption</td>
<td>IND-CPA, IND-CCA</td>
</tr>
<tr>
<td>Data integrity/authenticity</td>
<td>MAC</td>
<td>UF-CMA</td>
</tr>
</tbody>
</table>
Authenticated Encryption

In practice we often want both privacy and integrity/authenticity.

**Example:** A doctor wishes to send medical information $M$ about Alice to the medical database. Then

- We want **data privacy** to ensure Alice’s medical records remain confidential.
- We want **integrity/authenticity** to ensure the person sending the information is really the doctor and the information was **not modified** in transit.

We refer to this as **authenticated encryption**.
Authenticated Encryption Schemes

Syntactically, an authenticated encryption scheme is just a symmetric encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where
The notions of privacy for symmetric encryption carry over:

- IND-CPA
- IND-CCA
Adversary’s goal is to get the receiver to accept a “non-authentic” ciphertext $C$.

Integrity of ciphertexts: $C$ is “non-authentic” if it was never transmitted by the sender.
Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and $A$ an adversary.

<table>
<thead>
<tr>
<th>Game INT-CTXT $\mathcal{AE}$</th>
<th>procedure Dec($C$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedure Initialize</td>
<td>$M \leftarrow \mathcal{D}_K(C)$</td>
</tr>
<tr>
<td>$K \leftarrow \mathcal{K}; S \leftarrow \emptyset$</td>
<td>if ($C \not\in S \land M \neq \bot$) then</td>
</tr>
<tr>
<td>procedure Enc($M$)</td>
<td>$\text{win} \leftarrow \text{true}$</td>
</tr>
<tr>
<td>$C \leftarrow \mathcal{E}_K(M)$</td>
<td>return win</td>
</tr>
<tr>
<td>$S \leftarrow S \cup {C}$</td>
<td>procedure Finalize</td>
</tr>
<tr>
<td>return $C$</td>
<td>return win</td>
</tr>
</tbody>
</table>

The int-ctxt advantage of $A$ is

$$\text{Adv}^{\text{int-ctxt}}_{\mathcal{AE}}(A) = \Pr[\text{INT-CTXT}^A_{\mathcal{AE}} \Rightarrow \text{true}]$$
Integrity with privacy

The goal of authenticated encryption is to provide both integrity and privacy. We will be interested in IND-CPA + INT-CTXT.
**Fact:** \( \text{IND-CPA} + \text{INT-CTXT} \Rightarrow \text{IND-CCA} \)

That is if encryption scheme \( \mathcal{AE} \) is IND-CPA and also INT-CTXT secure, then it is automatically IND-CCA secure.
Plain Encryption Does Not Provide Integrity

**Alg** $E_K(M)$

$C[0] \leftarrow \{0, 1\}^n$

For $i = 1, \ldots, m$ do

$C[i] \leftarrow E_K(C[i-1] \oplus M[i])$

Return $C$

**Alg** $D_K(C)$

For $i = 1, \ldots, m$ do

$M[i] \leftarrow E_K^{-1}(C[i]) \oplus C[i-1]$

Return $M$

**Question:** Is CBC$S$ encryption INT-CTXT secure?
Plain Encryption Does Not Provide Integrity

\textbf{Alg} $E_K(M)$
\begin{align*}
C[0] &\leftarrow \{0, 1\}^n \\
\text{For } i = 1, \ldots, m \text{ do} & \\
C[i] &\leftarrow E_K(C[i-1] \oplus M[i]) \\
\text{Return } C
\end{align*}

\textbf{Alg} $D_K(C)$
\begin{align*}
\text{For } i = 1, \ldots, m \text{ do} & \\
M[i] &\leftarrow E_K^{-1}(C[i]) \oplus C[i-1] \\
\text{Return } M
\end{align*}

\textbf{Question:} Is CBC$^\$ encryption INT-CTXT secure?

\textbf{Answer:} No, because any string $C[0]C[1] \ldots C[m]$ has a valid decryption.
Plain Encryption Does Not Provide Integrity

\[ \text{Alg } \mathcal{E}_K(M) \]
\[ C[0] \leftarrow \{0, 1\}^n \]
For \( i = 1, \ldots, m \) do
\[ C[i] \leftarrow E_K(C[i-1] \oplus M[i]) \]
Return \( C \)

\[ \text{Alg } \mathcal{D}_K(C) \]
For \( i = 1, \ldots, m \) do
\[ M[i] \leftarrow E_K^{-1}(C[i]) \oplus C[i-1] \]
Return \( M \)

adversary \( A \)
\[ C[0]C[1]C[2] \leftarrow \{0, 1\}^{3n} \]

Then
\[ \text{Adv}^{\text{int-ctxt}}_{\mathcal{E}}(A) = 1 \]

This violates INT-CTXT.

A scheme whose decryption algorithm never outputs \( \perp \) cannot provide integrity!
Suppose $A$ has the CBC encryption $C[0]C[1]$ of a 1-block known message $M$. Then it can create an encryption $C'[0]C'[1]$ of any (1-block) message $M'$ of its choice via

$$
C'[0] \leftarrow C[0] \oplus M \oplus M' \\
C'[1] \leftarrow C[1]
$$
Encryption with Redundancy

Here $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is our block cipher and $h: \{0, 1\}^* \rightarrow \{0, 1\}^n$ is a “redundancy” function, for example

- $h(M[1] \ldots M[m]) = 0^n$
- $h(M[1] \ldots M[m]) = M[1] \oplus \cdots \oplus M[m]$
- A CRC
- $h(M[1] \ldots M[m])$ is the first $n$ bits of SHA1($M[1] \ldots M[m]$).

The redundancy is verified upon decryption.
Let $E: \{0, 1\}^k \times \{0, 1\}^n \to \{0, 1\}^n$ be our block cipher and $h: \{0, 1\}^* \to \{0, 1\}^n$ a redundancy function. Let $SE = (K, E', D')$ be CBC$ encryption and define the encryption with redundancy scheme $AE = (K, E, D)$ via

**Alg $E_K(M)$**

$M[1] \ldots M[m] \leftarrow M$

$M[m + 1] \leftarrow h(M)$

$C \leftarrow \$ $E'_K(M[1] \ldots M[m]M[m + 1])$

return $C$

**Alg $D_K(C)$**

$M[1] \ldots M[m]M[m + 1] \leftarrow D'_K(C)$

if $(M[m + 1] = h(M))$ then

return $M[1] \ldots M[m]$

else return ⊥
The adversary will have a hard time producing the last enciphered block of a new message.
adversary $A$

$M[1] \leftarrow \{0, 1\}^n$ ; $M[2] \leftarrow h(M[1])$


$M[1] \leftarrow \text{Dec}(C[0] C[1] C[2])$

This attack succeeds for any (not secret-key dependent) redundancy function $h$. 
A “real-life” rendition of this attack broke the 802.11 WEP protocol, which instantiated $h$ as CRC and used a stream cipher for encryption [BGW].

What makes the attack easy to see is having a clear, strong and formal security model.
Build an authenticated encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ by combining

- a given IND-CPA symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$
- a given PRF $F : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^n$

<table>
<thead>
<tr>
<th></th>
<th>CBC$$-AES</th>
<th>CTRC-AES</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMAC-SHA1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMAC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMAC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UMAC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\ldots</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Build an authenticated encryption scheme \( \mathcal{AE} = (K, E, D) \) by combining

- a given IND-CPA symmetric encryption scheme \( \mathcal{SE} = (K', E', D') \)
- a given PRF \( F : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^n \)

A key \( K = K_e \parallel K_m \) for \( \mathcal{AE} \) always consists of a key \( K_e \) for \( \mathcal{SE} \) and a key \( K_m \) for \( F \):

\[
\text{Alg } \mathcal{K} \\
K_e \leftarrow K' \; ; \; K_m \leftarrow \{0, 1\}^k
\]

Return \( K_e \parallel K_m \)
The order in which the primitives are applied is important. Can consider:

<table>
<thead>
<tr>
<th>Method</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypt-and-MAC (E&amp;M)</td>
<td>SSH</td>
</tr>
<tr>
<td>MAC-then-encrypt (MtE)</td>
<td>SSL/TLS</td>
</tr>
<tr>
<td>Encrypt-then-MAC (EtM)</td>
<td>IPSec</td>
</tr>
</tbody>
</table>

We study these following [BN].
Encrypt-and-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

\begin{align*}
\textbf{Alg} & \quad \mathcal{E}_{K_e\|\!K_m}(M) \\
& \quad C' \leftarrow \mathcal{E}'_{K_e}(M) \\
& \quad T \leftarrow F_{K_m}(M) \\
& \quad \text{Return } C'\|T
\end{align*}

\begin{align*}
\textbf{Alg} & \quad \mathcal{D}_{K_e\|\!K_m}(C'\|T) \\
& \quad M \leftarrow \mathcal{D}'_{K_e}(C') \\
& \quad \text{If } (T = F_{K_m}(M)) \text{ then return } M \\
& \quad \text{Else return } \bot
\end{align*}

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td></td>
</tr>
<tr>
<td>INT-CTXT</td>
<td></td>
</tr>
</tbody>
</table>
Encrypt-and-MAC

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

**Alg $\mathcal{E}_{K_e||K_m}(M)$**

$C' \leftarrow \mathcal{E}_K'(M)$

$T \leftarrow F_{K_m}(M)$

Return $C' || T$

**Alg $\mathcal{D}_{K_e||K_m}(C' || T)$**

$M \leftarrow \mathcal{D}_{K_e}'(C')$

If $(T = F_{K_m}(M))$ then return $M$

Else return ⊥

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td>NO</td>
</tr>
<tr>
<td>INT-CTX</td>
<td></td>
</tr>
</tbody>
</table>

Why? $T = F_{K_m}(M)$ is a deterministic function of $M$ and allows detection of repeats.
Encrypt-and-MAC

\( \mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}) \) is defined by

\[
\begin{align*}
\textbf{Alg} & \quad \mathcal{E}_{K_e \| K_m}(M) \\
& \quad C' \leftarrow \mathcal{E}_{K_e}(M) \\
& \quad T \leftarrow F_{K_m}(M) \\
& \quad \text{Return } C' \| T \\
\end{align*}
\]

\[
\begin{align*}
\textbf{Alg} & \quad \mathcal{D}_{K_e \| K_m}(C' \| T) \\
& \quad M \leftarrow \mathcal{D}'_{K_e}(C') \\
& \quad \text{If } (T = F_{K_m}(M)) \text{ then return } M \\
& \quad \text{Else return } \bot
\end{align*}
\]

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td>NO</td>
</tr>
<tr>
<td>INT-CTXT</td>
<td></td>
</tr>
</tbody>
</table>
Encrypt-and-MAC

\[ AE = (K, E, D) \] is defined by

**Algorithm 1:**

- \( C' \leftarrow E'_{K_e}(M) \)
- \( T \leftarrow F_{K_m}(M) \)
- Return \( C' \parallel T \)

**Algorithm 2:**

- \( M \leftarrow D'_{K_e}(C') \)
- If \( T = F_{K_m}(M) \) then return \( M \)
- Else return \( \perp \)

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td>NO</td>
</tr>
<tr>
<td>INT-CTXT</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Why?** May be able to modify \( C' \) in such a way that its decryption is unchanged.
MAC-then-Encrypt

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

$\textbf{Alg}$ $\mathcal{E}_{K_e\|K_m}(M)$
$T \leftarrow F_{K_m}(M)$
$C \leftarrow^$ $\mathcal{E}_{K_e}'(M\|T)$
Return $C$

$\textbf{Alg}$ $\mathcal{D}_{K_e\|K_m}(C)$
$M\|T \leftarrow \mathcal{D}'_{K_e}(C)$
If $(T = F_{K_m}(M))$ then return $M$
Else return $\bot$

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td></td>
</tr>
<tr>
<td>INT-CTXT</td>
<td></td>
</tr>
</tbody>
</table>
MAC-then-Encrypt

$\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by

\[
\begin{align*}
\textbf{ Alg } & \quad \mathcal{E}_{K_e \| K_m}(M) \\
& \quad T \leftarrow F_{K_m}(M) \\
& \quad C \leftarrow^\$ \mathcal{E}'_{K_e}(M \| T) \\
& \quad \text{Return } C
\end{align*}
\]

\[
\begin{align*}
\textbf{ Alg } & \quad D_{K_e \| K_m}(C) \\
& \quad M \| T \leftarrow \mathcal{D}'_{K_e}(C) \\
& \quad \text{If } (T = F_{K_m}(M)) \text{ then return } M \\
& \quad \text{Else return } \bot
\end{align*}
\]

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td>YES</td>
</tr>
<tr>
<td>INT-CTXT</td>
<td></td>
</tr>
</tbody>
</table>

Why? $\mathcal{SE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ is IND-CPA secure.
AE = (K, E, D) is defined by

\[ \text{Alg } E_{K_e||K_m}(M) \]
\[ T \leftarrow F_{K_m}(M) \]
\[ C \leftarrow E'_{K_e}(M || T) \]
Return C

\[ \text{Alg } D_{K_e||K_m}(C) \]
\[ M || T \leftarrow D'_{K_e}(C) \]
If \( T = F_{K_m}(M) \) then return M
Else return ⊥

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td>YES</td>
</tr>
<tr>
<td>INT-CTXT</td>
<td></td>
</tr>
</tbody>
</table>
MAC-then-Encrypt

\( \mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}) \) is defined by

**\textbf{Alg}** \( \mathcal{E}_{Ke \| Km}(M) \)

\( T \leftarrow F_{Km}(M) \)

\( C \leftarrow \$ \mathcal{E}'_{Ke}(M \| T) \)

Return \( C \)

**\textbf{Alg}** \( \mathcal{D}_{Ke \| Km}(C) \)

\( M \| T \leftarrow \mathcal{D}'_{Ke}(C) \)

If \( (T = F_{Km}(M)) \) then return \( M \)

Else return \( \bot \)

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td>YES</td>
</tr>
<tr>
<td>INT-CTXT</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Why?** May be able to modify \( C \) in such a way that its decryption is unchanged.
Encrypt-then-MAC

$AE = (K, E, D)$ is defined by

\[
\begin{align*}
\text{Alg } & E_{K_e||K_m}(M) \\
C' & \leftarrow E_{K_e}(M) \\
T & \leftarrow F_{K_m}(C') \\
\text{Return } & C'\|T
\end{align*}
\]

\[
\begin{align*}
\text{Alg } & D_{K_e||K_m}(C'\|T) \\
M & \leftarrow D'_{K_e}(C') \\
\text{If } (T & = F_{K_m}(C')) \text{ then return } M \\
\text{Else return } & \perp
\end{align*}
\]

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td></td>
</tr>
<tr>
<td>INT-CTXT</td>
<td></td>
</tr>
</tbody>
</table>
Encrypt-then-MAC

\( AE = (\mathcal{K}, \mathcal{E}, \mathcal{D}) \) is defined by

\[
\begin{align*}
\text{Alg } & \mathcal{E}_{K_e||K_m}(M) \\
C' & \leftarrow \mathcal{E}'_{K_e}(M) \\
T & \leftarrow F_{K_m}(C') \\
\text{Return } & C' || T
\end{align*}
\]

\[
\begin{align*}
\text{Alg } & \mathcal{D}_{K_e||K_m}(C' || T) \\
M & \leftarrow \mathcal{D}'_{K_e}(C') \\
\text{If } & (T = F_{K_m}(C')) \text{ then return } M \\
\text{Else return } & \bot
\end{align*}
\]

### Security Achieved?

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td><strong>YES</strong></td>
</tr>
<tr>
<td>INT-CTXT</td>
<td></td>
</tr>
</tbody>
</table>

Why? \( SE' = (\mathcal{K}', \mathcal{E}', \mathcal{D}') \) is IND-CPA secure.
Encrypt-then-MAC

\( \mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}) \) is defined by

\[
\begin{align*}
\textbf{Alg} & \quad \mathcal{E}_{K_e||K_m}(M) \\
C' & \leftarrow^{\$} \mathcal{E}_K'(M) \\
T & \leftarrow F_{K_m}(C') \\
\text{Return } C'\parallel T
\end{align*}
\]

\[
\begin{align*}
\textbf{Alg} & \quad \mathcal{D}_{K_e||K_m}(C'\parallel T) \\
M & \leftarrow \mathcal{D}_K'(C') \\
\text{If } (T = F_{K_m}(C')) \text{ then return } M \\
\text{Else return } \bot
\end{align*}
\]

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td>YES</td>
</tr>
<tr>
<td>INT-CTXT</td>
<td></td>
</tr>
</tbody>
</table>
Encrypt-then-MAC

\( \mathcal{AE} = (K, E, D) \) is defined by

\[
\begin{align*}
\textbf{Alg } & E_{K_e||K_m}(M) \\
C' & \leftarrow E'(M) \\
T & \leftarrow F_{K_m}(C') \\
\text{Return } C' \| T
\end{align*}
\]

\[
\begin{align*}
\textbf{Alg } & D_{K_e||K_m}(C' \| T) \\
M & \leftarrow D'(C') \\
\text{If } (T = F_{K_m}(C')) \text{ then return } M \\
\text{Else return } \bot
\end{align*}
\]

<table>
<thead>
<tr>
<th>Security</th>
<th>Achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA</td>
<td>YES</td>
</tr>
<tr>
<td>INT-CTXT</td>
<td>YES</td>
</tr>
</tbody>
</table>

Why? If \( C \| T \) is new then \( T \) will be wrong.
Achieving IND-CCA

We saw that

\[
\text{IND- CPA } + \text{ INT-CTXT } \Rightarrow \text{ IND-CCA}. 
\]

So an IND-CCA secure symmetric encryption scheme can be built as follows:

- Take any IND-CPA symmetric encryption scheme \(\mathcal{SE}\)
- Take any PRF \(F\)
- Combine them in Encrypt-then-MAC composition

Example choices of the base primitives:

- \(\mathcal{SE}\) is AES-CBC$
- \(F\) is AES-CMAC or HMAC-SHA1
We have used separate keys $K_e, K_m$ for the encryption and message authentication. However, these can be derived from a single key $K$ via $K_e = F_K(0)$ and $K_m = F_K(1)$, where $F$ is a PRF such as a block cipher, the CBC-MAC or HMAC.

Trying to directly use the same key for the encryption and message authentication is error-prone, but works if done correctly.
## Generic Composition in Practice

<table>
<thead>
<tr>
<th>AE in</th>
<th>is based on</th>
<th>which in general is</th>
<th>and in this case is</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH</td>
<td>E&amp;M</td>
<td>insecure</td>
<td>secure</td>
</tr>
<tr>
<td>SSL</td>
<td>MtE</td>
<td>insecure</td>
<td>insecure</td>
</tr>
<tr>
<td>SSL + RFC 4344</td>
<td>MtE</td>
<td>insecure</td>
<td>secure</td>
</tr>
<tr>
<td>IPSec</td>
<td>EtM</td>
<td>secure</td>
<td>secure</td>
</tr>
<tr>
<td>WinZip</td>
<td>EtM</td>
<td>secure</td>
<td>insecure</td>
</tr>
</tbody>
</table>

**Why?**

- Encodings
- Specific “E” and “M” schemes
- For WinZip, disparity between usage and security model
SSH2 encryption uses inter-packet chaining which is insecure [D, BKN].

RFC 4344 [BKN] proposed fixes that render SSH provably IND-CPA+INT-CTXT secure. Fixes recommended by Secure Shell Working Group and included in OpenSSH since 2003, but became default only in 2009. Fixes also included in PuTTY since 2008.
SSL uses MtE

\[ \mathcal{E}_{K_e\|K_M} = \mathcal{E}'_{K_e}(M\|F_{K_m}(M)) \]

which we saw is not INT-CTXT-secure in general. But \( \mathcal{E}' \) is CBC\$ in SSL, and in this case the scheme does achieve INT-CTXT [K].

\( F \) in SSL is HMAC.

Sometimes SSL uses RC4 for encryption.
The goal has evolved into Authenticated Encryption with Associated Data (AEAD) [Ro].

- Associated Data (AD) is authenticated but not encrypted
- Schemes are nonce-based (and deterministic)

**Sender**

- $C \leftarrow E_K(N, AD, M)$
- Send $(N, AD, C)$

**Receiver**

- Receive $(N, AD, C)$
- $M \leftarrow D_K(N, AD, C)$

Sender must never re-use a nonce.

But when attacking integrity, the adversary may use any nonce it likes.
AEAD Privacy

Let $A\mathcal{E} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Adversary is not allowed to repeat a nonce in its LR queries.

Game Left$_{A\mathcal{E}}$

**procedure** Initialize

$K \leftarrow K$

**procedure** LR$(N, AD, M_0, M_1)$

Return $C \leftarrow \mathcal{E}_K(N, AD, M_0)$

Game Right$_{A\mathcal{E}}$

**procedure** Initialize

$K \leftarrow K$

**procedure** LR$(N, AD, M_0, M_1)$

Return $C \leftarrow \mathcal{E}_K(N, AD, M_1)$

Associated to $A\mathcal{E}$, $A$ are the probabilities

$$\Pr \left[ \text{Left}^A_{A\mathcal{E}} \Rightarrow 1 \right] \quad | \quad \Pr \left[ \text{Right}^A_{A\mathcal{E}} \Rightarrow 1 \right]$$

that $A$ outputs 1 in each world. The (ind-cpa) advantage of $A$ is

$$\text{Adv}^{\text{ind-cpa}}_{A\mathcal{E}}(A) = \Pr \left[ \text{Right}^A_{A\mathcal{E}} \Rightarrow 1 \right] - \Pr \left[ \text{Left}^A_{A\mathcal{E}} \Rightarrow 1 \right]$$
Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Adversary is not allowed to repeat a nonce in its $\text{Enc}$ queries.

\begin{align*}
\text{Game } \text{INTCTXT}_{\mathcal{AE}} \\
\text{procedure Initialize} & \quad K \leftarrow K \\
\text{procedure Enc}(N, AD, M) & \quad C \leftarrow \mathcal{E}_K(N, AD, M) \\
& \quad S_{N,AD} \leftarrow S_{N,AD} \cup \{C\} \\
& \quad \text{return } C \\
\text{procedure Dec}(N, AD, C) & \quad M \leftarrow \mathcal{D}_K(N, AD, C) \\
& \quad \text{if } (C \notin S_{N,AD} \land M \neq \perp) \text{ then} \\
& \quad \quad \text{win} \leftarrow \text{true} \\
& \quad \text{return win} \\
\text{procedure Finalize} & \quad \text{return win}
\end{align*}

The int-ctxt advantage of $A$ is

$$\text{Adv}^{\text{int-ctxt}}_{\mathcal{AE}}(A) = \Pr[\text{INTCTXT}_{\mathcal{AE}}^A \Rightarrow \text{true}]$$
**Generic composition:** E&M, MtE, EtM extend and again EtM is the best.

**1-pass schemes:** IAPM [J], XCBC/XEBC [GD], OCB [RBBK, R]

**2-pass schemes:** CCM [FHW], EAX [BRW], CWC [KVW], GCM [MV]

**Stream cipher based:** Helix [FWSKLK], SOBER-128 [HR]

- 1-pass schemes are fast
- 2-pass schemes are patent-free
- Stream cipher based schemes are fast
Nonce-based symmetric encryption

Worrying for the moment just about privacy, one could build a nonce-based symmetric encryption scheme by

- Using the nonce as IV in CBC mode
- Using the nonce as counter in CTR

Both are insecure, meaning fail to be IND-CPA, but can be fixed.
Nonce-based CBC encryption

Doesn’t work:

\[ M[1] \rightarrow E_K \rightarrow C[1] \]

Nonce-based CBC encryption

Doesn’t work:

Works, and is easily justified under the assumption that $E$ is a PRF:
Nonce-based CTR encryption

Doesn’t work:

\[
\begin{align*}
N + 1 & \quad \quad \quad \quad \quad \quad \quad N + 2 \\
\leftrightarrow & \quad \quad \quad \quad \quad \quad \quad \leftrightarrow \\
E_K & \quad \quad \quad \quad \quad \quad \quad E_K \\
M[1] & \quad \quad \quad \quad \quad \quad \quad M[2] \\
\oplus & \quad \quad \quad \quad \quad \quad \quad \oplus \\
C[1] & \quad \quad \quad \quad \quad \quad \quad C[2] \\
\end{align*}
\]
Nonce-based CTR encryption

 Doesn’t work:

\[ N + 1 \]
\[ \oplus \]
\[ M[1] \]
\[ \rightarrow \]
\[ C[1] \]
\[ E_K \]
\[ N + 2 \]
\[ \oplus \]
\[ M[2] \]
\[ \rightarrow \]
\[ C[2] \]

Works, and is easily justified under the assumption that \( E \) is a PRF:

\[ N \]
\[ \oplus \]
\[ M[1] \]
\[ \rightarrow \]
\[ C[1] \]
\[ E_K \]
\[ R + 1 \]
\[ \oplus \]
\[ M[2] \]
\[ \rightarrow \]
\[ C[2] \]
\[ E_K \]
\[ R + 2 \]
\[ \oplus \]
\[ \ldots \]
Also kind of works:

\[
\begin{align*}
N\|1 & \quad E_K \quad M[1] \oplus C[1] \\
N\|2 & \quad E_K \quad M[2] \oplus C[2] \\
N\|3 & \quad E_K \quad M[3] \oplus C[3] \\
\end{align*}
\]

If maximum message length is \(2^b\) blocks then nonce length is limited to \(n - b\) bits.

We will see this tradeoff in some subsequent AEAD schemes.
A tweakable block cipher is a map

\[ E: \{0, 1\}^k \times \text{TwSp} \times \{0, 1\}^n \rightarrow \{0, 1\}^n \]

such that

\[ E^K_T: \{0, 1\}^n \rightarrow \{0, 1\}^n \]

is a permutation for every \( K, T \), where \( E^K_T(X) = E(K, T, X) \).

With a single key one thus implicitly has a large number of maps

These appear to be independent random permutations to an adversary who does not know the key \( K \), even if it can choose the tweaks and inputs.

Tweakable block ciphers can be built cheaply from block ciphers [R].
OCB [RBBK]


$S = \text{PMAC}_K(AD)$ using separate tweaks.

Output may optionally be truncated.

Some complications (not shown) for non-full messages.

Optional in IEEE 802.11i
Patents on 1-pass schemes

- Jutla (IBM) 7093126
- Gligor and Donescu (VDG, Inc.) 6973187
- Rogaway 7046802, 7200227
2-pass AEAD

- Tailored generic composition of specific base schemes
- Single key

Philosophical questions:
- What is the advantage of one key versus two given that can always derive the two from the one?
- Why not just do specific generic composition of specific base schemes?
MtE-based but single key throughout
CTR-ENC is nonce-based counter mode encryption, and CBC-MAC is the basic CBC MAC. Ciphertext is $C \parallel T$

NIST SP 800-38C, IEEE 802.11i
Critiques of CCM [RW]

- Not on-line: message and $AD$ lengths must be known in advance
- Can’t pre-process static $AD$
- Nonce length depends on message length and the former decreases as the latter increases
- Awkward/unnecessary parameters
- Complex encodings
EtM-based but single key throughout
CTR-ENC is nonce-based counter mode encryption.
Online; can pre-process static $AD$; always 128-bit nonce; simple; same performance as CCM.

ANSI C12.22
CTR-ENC is nonce-based counter mode encryption. CWC-HASH is a AU polynomial-based hash. $K_H$ is derived from $K$ via $E$.

Parallelizable; 300K gates for 10 Gbit/s (ASIC at 130 nanometers); Roughly same software speed as CCM, EAX, but can be improved via precomputation.
CTR-ENC is nonce-based counter mode encryption. GCM-HASH is a AU polynomial-based hash. $K_H$ is derived from $K$ via $E$.

Can be used as a MAC.

NIST SP 800-38D
Let $F$ be a finite field. To data $C = C[0] \ldots C[m-1]$ with $C[i] \in F$ ($0 \leq i \leq m-1$) we associate the polynomial

$$P_C(x) = \sum_{i=0}^{m-1} C[i] \cdot x^i$$

and let $H(K_H, C) = P_C(K_H)$. If $C_1 \neq C_2$, then for $K_H$ chosen at random,

$$\Pr[H(K_H, C_1) = H(K_H, C_2)] = \Pr[(P_{C_1} - P_{C_2})(K_H) = 0] \leq \frac{\max(m_1, m_2) - 1}{|F|},$$

where $m_i$ is the number of blocks in $C_i$.

CWC-HASH works over $F = \mathrm{GF}(p)$ where $p$ is the prime $2^{127} - 1$, and is similar to Poly127 but is parallelizable. GCM-HASH works over $F = \mathrm{GF}(2^{128})$, which they argue is faster.
Critique of GCM [F]

- Message length is at most $2^{36} - 64$ bytes which may not always be enough.

- Performance improvements require large per-key tables, which may be undesirable. (A wireless access point would need 1000 keys, hard for libraries to specify table sizes, tables contain confidential materials, etc.)

- As usual, forgery is possible via a birthday attack, but for some parameters the attacker can get the key.
Performance Comparisons x32

Gladman’s C code
Performance Comparisons x64

Gladman’s C code
Which AEAD scheme should I use?

No clear answer. Ask yourself

- What performance do I need?
- Single or multiple keys?
- Patents ok or not?
- Do I need to comply with some standard?