The Evolution of Authenticated Encryption

Phillip Rogaway
University of California, Davis, USA

Includes joint work with
Mihir Bellare, John Black, Ted Krovetz, and Tom Shrimpton

DIAC — Directions in Authenticated Ciphers
05 July 2012
Stockholm, Sweden
Today

Historically ordered

1. Introduction
   - The recognition of AE as a useful “thing”
   - Modes that don’t work

2. Definitions and constructions
   - Defining AE
   - Generic composition
   - RPC, XCBC$, IAPM, OCB
   - Defining nonce-based AEAD
   - CCM
   - GCM
   - OCB, again
   - Defining MRAE
   - SIV

3. Discussion
   - Taxonomy
   - Patents
   - Suggestions
   - Sample research questions
Authenticated Encryption (AE)

Promises **two benefits**

1. An easier-to-correctly-use abstraction boundary
2. More efficient realizations

Begins with two **realizations** regarding symmetric encryption

1. “Integrity” /“authenticity” is routinely needed
2. “Standard” privacy mechanisms don’t provide it
Check / insert redundancy

CBC

IV $\rightarrow P_1 \rightarrow E_K \rightarrow C_1$

$\rightarrow P_2 \rightarrow E_K \rightarrow C_2$

$\rightarrow P_3 \rightarrow E_K \rightarrow C_3$

$\rightarrow \Sigma \rightarrow E_K \rightarrow C_4$

1. Introduction
Add more arrows

PCBC

See: Yu, Hartman, Raeburn 2004
“The Perils of Unauthenticated Encryption: Kerberos Version 4”
Still more arrows/operations

Promtly broken by Jutla (1999) and by Ferguson, Whiting, Kelsey, Wagner (1999)
Emerging understanding that:
- **Beyond IND-CPA privacy** was often desirable
- Didn’t come with **standard** encryption methods
- Simple ways to try to get it *cheaply* don’t work

~2000

**Similar realizations in the public-key world ...**

- [Bleichenbacher 1998] – “A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1”
- Reaction was that IND-CPA security was not enough
  - **CCA1** security (Naor-Yung 1990)
  - **CCA2** security (Rackoff-Simon 1991)
  - **Non-malleability** (Dolev-Dwork-Naor 1991)
AE Defined

[Bellare, Rogaway 2000] – “Encode-then-encipher encryption: how to exploit nonces or redundancy in plaintexts for efficient cryptography”


- **Conventional privacy** [BDJR97]: Indistinguishability / semantic security.
- **Authenticity**: The only ciphertexts $C$ that will decrypt to something valid are those previously obtained by an $Enc(\cdot)$ call.
AE Defined

\[
\text{Adv}^\text{priv}_\Pi(A) = \Pr[A \overset{\text{Enc}_K(\cdot)}{\rightarrow} 1] - \Pr[A \overset{\text{Enc}_K(0^{|\cdot|})}{\rightarrow} 1]
\]

2. Definitions and constructions
AE Defined

\[ \text{Adv}^{\text{priv}}_{\Pi}(A) = \Pr[A^{\text{Enc}_K(\cdot)} \to 1] - \Pr[A^{\text{Enc}_K(0\mid\cdot)} \to 1] \]

\[ \text{Adv}^{\text{auth}}_{\Pi}(A) = \Pr[A^{\text{Enc}_K(\cdot)} \to C^*: \text{no query returned } C^* \text{ and } \text{Dec}_K(C^*) \neq \bot] \]
The Strength of AE

- Implies **IND-CCA$_2$** security
- Implies **NM-CCA$_2$** security
Generic Composition
of an IND-CPA encryption scheme and a PRF

2. Definitions and constructions

[BN 2000]
The Cost of Generic Composition

Cost(AE) = Cost(Enc) + Cost(MAC)

Example cases:
Enc = CTR, CBC
MAC = CMAC, HMAC, PMAC, UMAC

⇒ Generic composition can be pretty cheap – if you use a cheap MAC
2. Definitions and constructions
XCBC$ Mode

Illustration from
Gligor-Donescu

2. Definitions and constructions
IAPM Mode

Illustration from [Jutla 2001]

2. Definitions and constructions
**OCB Mode** (later “OCB1”)  
Like IAPM but highly optimized. Motivated by NIST’s modes call.

\[
Z[i] = R \oplus \gamma_i \cdot L  \\
\text{Checksum} = M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]0^* \oplus Y[m]
\]

- Arbitrary-length messages
- Efficient offset calculations
- \( m + 2 \) blockcipher calls, \( m = \lceil |M|/n \rceil \)
- Single blockcipher key
- Cheap key setup (one blockcipher call)

2. Definitions and constructions
Two important players: NIST and IEEE 802.11i

- WiFi standard ratified in 1999
  Uses WEP security

- Fatal attacks soon emerge:
  - [Fluhrer, Mantin, Shamir 2001]
    Weaknesses in the key scheduling algorithm of RC4
  - [Stubblefield, Ioannidis, Rubin 2001]
    Using the Fluhrer, Mantin, Shamir attack to break WEP
  - [Borisov, Goldberg, Wagner 2001]
    Intercepting mobile communications: the insecurity of 802.11
  - [Cam-Winget, Housley, Wagner, Walker 2003]
    Security flaws in 802.11 data links protocols

- WEP → TKIP → WPA → WPA2
  - Draft solutions based on OCB
  - Politics and patent-avoidance:
    [Whiting, Housley, Ferguson 2002] develop CCM (=CCMP)
  - CCM standardized for 802.11, then NIST
Before describing CCM ...
Back to the definitional story

1) Move the coins “out” and make a “nonce” sufficient \[\text{[RBBK01]}\]

2) Add in “associated data” \[\text{[R02]}\]

- Random values routinely aren’t
- Many application have an available nonce
- Weaker user requirement; less misuse

• Requirement from Cam-Winget, Kaliski, Walker
• AD is authenticated but not encrypted
• Failure to provide same AD on decryption results in ⊥
AEAD

Also: (1) Ask for indistinguishability from random bits [RBBK00]
(2) All-in-one definition [R, Shrimpton 2006]

\[
\text{Adv}^{\text{aead}}_{\Pi}(A) = \Pr[A \xrightarrow{\text{Enc}_K \text{ Dec}_K} 1] - \Pr[A \xrightarrow{\bot} 1]
\]

\[\text{A may not: repeat an } N\text{-value in an enc query; or ask a dec query } (N, AD, C) \text{ after } C \text{ is returned by an } (N, AD, \cdot) \text{ enc query}\]
2. Definitions and constructions

CCM Mode

Roughly
MAC-then-Encrypt
Functions \textbf{FORMAT and COUNT}

\[ \text{COUNT}_q(N, m) = N_1 \parallel N_2 \parallel \cdots \parallel N_m \text{ where} \]
\[ N_i = 0^5 \parallel [q-1]_3 \parallel N \parallel [i]_{8q} \]

\[ \text{FORMAT}_{q,t}(N, A, P) = \]
\[ 0 \parallel \text{if } A = \varepsilon \text{ then } 0 \text{ else } 1 \text{ endif} \parallel [t/2-1]_3 \parallel [q-1]_3 \parallel \]
\[ N \parallel [|P|_8]_{8q} \parallel \]
\[ \text{if } A = \varepsilon \text{ then } \varepsilon \text{ elseif } \]
\[ |A|_8 < 2^{16} - 2^8 \text{ then } [|A|_8]_{16} \]
\[ \text{elseif } |A|_8 < 2^{32} \text{ then } 0xFFFE \parallel [|A|_8]_{32} \text{ else } 0xFFFF \parallel [|A|_8]_{64} \text{ endif} \parallel \]
\[ A \parallel \]
\[ \text{if } A = \varepsilon \text{ then } \varepsilon \text{ elseif } |A|_8 < 2^{16} - 2^8 \text{ then } (0x00)^{14-|A|_8 \text{ mod } 16} \]
\[ \text{elseif } |A|_8 < 2^{32} \text{ then } (0x00)^{10-|A|_8 \text{ mod } 16} \text{ else } (0x00)^{6-|A|_8 \text{ mod } 16} \text{ endif} \parallel \]
\[ P \parallel \]
\[ (0x00)^{-(|M|_8 \text{ mod } 16)} \]
CCM Mode

• Provably secure, with OK bounds, if AE if $E$ is a good PRP  [Jonsson 2002]
• Widely used, standardized (eg, in 802.11)
• Simple to implement
• Only forward direction of blockcipher used

• About $2m+2$ blockcipher calls
• Half non-parallelizable
• Word alignment disrupted
• Can’t preprocess static AD
• Not “online” — need to know $m$ in advance
• Complex

• User must specify
  $q \in \{2,3,4,5,6,7,8\}$ – byte length of byte length of longest message which determines nonce length(!) of $\tau = 15 - q$
The issues with CCM aren’t hard to fix

- Generic composition of CTR and CMAC is a good alternative
- EAX is a CCM-like mode intended to fix CCM’s problems
GCM Mode
with 96-bit nonce $N$

See: [R 2011], “Evaluation of Some Blockcipher Modes of Operation”, Ch. 12

[McGrew, Viega 2004]
(Follows CWC
[Kohno, Viega, Whiting 2004])
NIST SP 800-38D:2007
RFC 4106, 5084, 5116, 5288, 5647
ISO 19772:2009
GCM Mode

- Provably secure, with OK bounds for long tags
- Parallelizable, online
- About $m+1$ blockcipher calls, all of them parallelizable
- Very efficient in HW
- Reasonably efficient in SW with AES-NI, PCMULDQ, preprocessing & tables
- Static AD can be preprocessed
- Only forward direction of blockcipher used
- First forgery after $2^{t/2}$ queries
- After, additional forgeries come quickly

- Poor bound if truncate tag too much [Ferguson, 2005] (don’t truncate <96 bits)
- Not that efficient in SW, even with PCMULDQ support
- Timing attacks an issue for table-based realizations (slow setup, too)
- Maximum of $2^{36}$-32 bytes
- “Reflected-bit” convention for representing field points unfortunate
- $|N|\neq 96$ case not handled well
- Published proof is buggy [Iwata, 2012]
OCB Mode

in terms of a tweakable blockcipher [LRW02]

$$\tilde{E}_K^{N_1} M_1 \oplus \tilde{E}_K^{N_2} M_2 \oplus \tilde{E}_K^{N_3} M_3 \oplus \tilde{E}_K^{N_4} M_4 = M_1 \oplus M_2 \oplus M_3 \oplus M_4$$

2. Definitions and constructions
OCB Mode
in terms of a tweakable blockcipher [LRW02]

\[
\begin{align*}
M_1 & \rightarrow \tilde{E}_K^{N1} \rightarrow C_1 \\
M_2 & \rightarrow \tilde{E}_K^{N2} \rightarrow C_2 \\
M_3 & \rightarrow \tilde{E}_K^{N3} \rightarrow C_3 \\
M_* & \rightarrow 0* \\
\text{Checksum} & \rightarrow \tilde{E}_K^{N3*}$
\end{align*}
\]

\[= M_1 \oplus M_2 \oplus M_3 \oplus M_4 0^*\]
OCB Mode
in terms of a tweakable blockcipher [LRW02]
Making the Tweakeable Blockcipher

\[ E^N_i (X) = E_K(X \oplus \Delta) \oplus \Delta \quad \text{with} \quad \Delta = \text{Initial} + \lambda_i L \]

\[ E^N_i^* (X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \text{Initial} + \lambda_i^* L \]

\[ E^N_i^$ (X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \text{Initial} + \lambda_i^$ L \]

\[ E^N_i^{*\$} (X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \text{Initial} + \lambda_i^{*\$} L \]

\[ E^i (X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \lambda_i L \]

\[ E^i^* (X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \lambda_i^* L \]

Nonce = \(0^{127-|N|} 1 \ N\)

Top = Nonce & \(1^{122} 0^6\)

Bottom = Nonce & \(1^{122} 1^6\)

Ktop = \(E_K(\text{Top})\)

Stretch = Ktop \(\parallel (\text{Ktop} \oplus (\text{Ktop} \ll 8))\)

Initial = (Stretch \(\ll \text{Bottom}) [1..128]\)

\[ L = E_K(0^{128}) \]

\[ \lambda_i = 4 \ a(i) \]

\[ \lambda_i^* = 4 \ a(i)+1 \]

\[ \lambda_i^$ = 4 \ a(i)+2 \]

\[ \lambda_i^{*$} = 4 \ a(i)+3 \]

\[ a(0) = 0 \]

\[ a(i) = a(i-1) \oplus 2^{\text{ntz}(i)} \]

2. Definitions and constructions
Software Performance
Intel Core x86 i7 – “Sandy Bridge”
64-bit OS, using AES/GCM NIs

<table>
<thead>
<tr>
<th>Mode</th>
<th>4KB cpb</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCM</td>
<td>5.14</td>
</tr>
<tr>
<td>GCM</td>
<td>2.95</td>
</tr>
<tr>
<td>OCB</td>
<td>0.87</td>
</tr>
</tbody>
</table>

2. Definitions and constructions
Software Performance
Intel Core x86 i5-650 – “Clarkdale”
64-bit OS, using AES/GCM NIs

<table>
<thead>
<tr>
<th>Mode</th>
<th>4K cpb</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCM</td>
<td>2.09</td>
</tr>
<tr>
<td>GCM</td>
<td>2.46</td>
</tr>
<tr>
<td>OCB</td>
<td>0.21</td>
</tr>
</tbody>
</table>
## Authenticated-Encryption Software Performance: Comparison of CCM, GCM, and OCB

- Click on a Time or Overhead plot to see a larger version of it.
- Click on a Mode (CCM, GCM, OCB, etc) to retrieve the raw data.
- Here OCB means OCB. A comparison webpage compares the performance of OCB variants.
- Further notes can be found on the bottom of this page.

### 2. Definitions and constructions

<table>
<thead>
<tr>
<th>Environment (details)</th>
<th>Time (cpl vs. bytes)</th>
<th>Overhead (subtract time for CTR)</th>
<th>Mode (clickable)</th>
<th>Over 4096</th>
<th>Time 4096</th>
<th>IPI (cpl)</th>
<th>Size (bytes)</th>
<th>Init (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel x86 i5-650 “Clarkdale” 64-bit NI</td>
<td><img src="image1" alt="Graph" /></td>
<td><img src="image2" alt="Graph" /></td>
<td>CCM</td>
<td>2.90</td>
<td>5.17</td>
<td>457</td>
<td>912</td>
<td>265</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GCM</td>
<td>2.46</td>
<td>5.37</td>
<td>453</td>
<td>936</td>
<td>327</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OCB</td>
<td>0.21</td>
<td>1.48</td>
<td>187</td>
<td>624</td>
<td>295</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTR</td>
<td>0.21</td>
<td>1.48</td>
<td>187</td>
<td>624</td>
<td>295</td>
</tr>
<tr>
<td>Intel x86 i5-650 “Clarkdale” 32-bit NI</td>
<td><img src="image3" alt="Graph" /></td>
<td><img src="image4" alt="Graph" /></td>
<td>CCM</td>
<td>2.79</td>
<td>1.31</td>
<td>170</td>
<td>512</td>
<td>274</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GCM</td>
<td>2.49</td>
<td>3.31</td>
<td>479</td>
<td>956</td>
<td>365</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OCB</td>
<td>0.20</td>
<td>1.59</td>
<td>204</td>
<td>624</td>
<td>318</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTR</td>
<td>0.20</td>
<td>1.59</td>
<td>204</td>
<td>624</td>
<td>318</td>
</tr>
<tr>
<td>Intel x86 i5-650 “Clarkdale” 64-bit Kasper-Schwabe</td>
<td><img src="image5" alt="Graph" /></td>
<td><img src="image6" alt="Graph" /></td>
<td>GCM</td>
<td>14.7</td>
<td>22.4</td>
<td>26.7</td>
<td>1346</td>
<td>3780</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GCM-256</td>
<td>3.19</td>
<td>10.9</td>
<td>15.2</td>
<td>731</td>
<td>2560</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OCB</td>
<td>0.31</td>
<td>8.05</td>
<td>9.24</td>
<td>321</td>
<td>3430</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTR</td>
<td>0.31</td>
<td>8.05</td>
<td>9.24</td>
<td>321</td>
<td>3430</td>
</tr>
<tr>
<td>ARM Cortex-A8 32-bit OpenSSL</td>
<td><img src="image7" alt="Graph" /></td>
<td><img src="image8" alt="Graph" /></td>
<td>CCM</td>
<td>25.9</td>
<td>51.3</td>
<td>53.7</td>
<td>512</td>
<td>1390</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GCM-256</td>
<td>3.49</td>
<td>28.9</td>
<td>30.9</td>
<td>784</td>
<td>2050</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OCB</td>
<td>0.49</td>
<td>25.4</td>
<td>25.9</td>
<td>244</td>
<td>236</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTR</td>
<td>0.49</td>
<td>25.4</td>
<td>25.9</td>
<td>244</td>
<td>236</td>
</tr>
<tr>
<td>PowerPC 970 64-bit OpenSSL</td>
<td><img src="image9" alt="Graph" /></td>
<td><img src="image10" alt="Graph" /></td>
<td>CCM</td>
<td>38.2</td>
<td>75.7</td>
<td>77.8</td>
<td>966</td>
<td>1510</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GCM-256</td>
<td>16.0</td>
<td>53.5</td>
<td>56.2</td>
<td>966</td>
<td>1030</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OCB</td>
<td>6.0</td>
<td>37.5</td>
<td>19.6</td>
<td>784</td>
<td>2200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTR</td>
<td>0.6</td>
<td>37.5</td>
<td>19.6</td>
<td>784</td>
<td>2200</td>
</tr>
<tr>
<td>UltraSPARC III 64-bit OpenSSL</td>
<td><img src="image11" alt="Graph" /></td>
<td><img src="image12" alt="Graph" /></td>
<td>CCM</td>
<td>25.3</td>
<td>49.4</td>
<td>51.7</td>
<td>912</td>
<td>1280</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GCM-256</td>
<td>15.2</td>
<td>39.3</td>
<td>41.5</td>
<td>912</td>
<td>904</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OCB</td>
<td>6.9</td>
<td>25.0</td>
<td>10.6</td>
<td>784</td>
<td>1170</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTR</td>
<td>0.6</td>
<td>25.0</td>
<td>10.6</td>
<td>784</td>
<td>1170</td>
</tr>
</tbody>
</table>

See the OCB homepage for more platforms and data, or for reference code.
Limitations of OCB

1. Blockcipher used in the forward and backward direction
2. No CTR-like locality in blocks being enciphered
3. Security “only” to birthday bound
4. Not resistant to non-reuse
Misuse-Resistant AE

MRAE

• If the IV is a **nonce**, you get what an AE delivers
• If the IV gets **reused**, all that leaks is repetitions
  - authenticity is undamaged
  - privacy damaged to the extent unavoidable—repetitions of \((AD, M)\) revealed
**Misuse-Resistant AE**

**MRAE**

\[\text{Enc}_K(\cdot,\cdot,\cdot)\]

\[\text{Dec}_K(\cdot,\cdot,\cdot)\]

\[\text{AD, IV, M}\]

\[\text{A}\]

\[\text{C}\]

\[\$ (\cdot,\cdot,\cdot)\]

\[\perp (\cdot,\cdot,\cdot)\]

**A** may not: ask an enc query \((\text{AD}, M, C)\);
ask a dec query of \((\text{AD,IV}, C)\) after an enc query \((\text{AD, IV, M})\) returns \(C\)
Deterministic AE
(DAE)

$\text{Enc}_K (\cdot, \cdot)$

$\text{Dec}_K (\cdot, \cdot)$

Deterministic

vector

$AD, M$

$AD, C$

$\bot (\cdot, \cdot)$

$C$

$C$

$\bot$

A may not: ask a dec query $(AD, C)$ after an enc query $(AD, M)$ returns $C$; repeat a query

$2. \text{Definitions and constructions}$

DAE $\Rightarrow$ MRAE: Regard one component of the $AD$ as the IV

($\approx$ “PRI” – “pseudorandom Injection”)[R, Shrimpton 2006]
SIV

ISO/IEC 19772:2009
RFC 5297

2. Definitions and constructions
A Traditional Taxonomy

Confusion/diffusion: one atomic primitive
  - Helix, SOBER, ...

Composed: ind$-secure symmetric encryption + PRF
  “two-pass” - EtM
  - CCM, GCM, etc.

Integrated: blend privacy/authenticity parts
  “one-pass” - OCB
A Different Taxonomy

Blockcipher

Tweakable blockcipher

Probabilistic enc scheme + PRF

Nonce-based enc scheme + PRF

Stream Cipher + AXU hash

Strong VIL PRP

Permutation

AE Scheme

(de novo constructions)
Patents

- 6,973,187  Gligor and Donescu  2005.12.06
- 6,963,976  Jutla  2005.11.07
- 7,093,126  Jutla  2006.08.15
- 8,107,620  Julta  2007.04.03
- 7,046,802  Rogaway  2006.05.16
- 7,200,227  Rogaway  2007.04.03
- 7,949,129  Rogaway  2011.05.24


- 20080240423. Gueron. Speeding up Galois Counter Mode (GCM) computations.

- 20060126835. Chen and Buckingham. Authenticated encryption method and apparatus.

- 20090310775. Gueron and Kounavis. Using a single instruction multiple data (SIMD) instruction to speed up Galois Counter Mode (GCM) computation.

3. Discussion
Do **Gligor** or **Jutla** Patents Read Against OCB?

I don’t know.

---

3. Discussion

Claim 1 of Gligor-Donescu #6,973,187

1. An encryption method for providing both data confidentiality and integrity for a message, comprising the steps of:

   - receiving an input plaintext string comprising a message and padding it as necessary such that its length is a multiple of 1 bits;
   - partitioning the input plaintext string a length that is a multiple of 1 bits into a plurality of equal-size blocks of 1 bits in length;
   - creating an a Manipulation Detection Code (MDC) block of 1 bits in length that includes the result of applying a non-cryptographic MDC function to the plurality of the equal-size blocks;
   - making one and only one processing pass with a single cryptographic primitive over each of said equal-size blocks and the MDC block to create a plurality of hidden ciphertext blocks each of 1 bits in length; and
   - performing a randomization function over said plurality of hidden ciphertext blocks to create a plurality of output ciphertext blocks each of 1 bits in length.
Do Gligor or Jutla Patents Read Against OCB?

I don’t know.

3. Discussion

Claim 1 of Jutla’s #8,107,620

1. A method for encrypting a sequence of plain-text messages using an n-bit block-cipher, the method comprising:
   choosing first and second secret keys;
   initializing an initial vector;
   initializing a pair-wise differentially uniform sequence generator using the said second secret key and the said initial vector;
   inputting at least one of a plurality of plain-text messages into an encryptor comprising a series of cipher blocks;
   generating a sequence of pair-wise differentially uniform random numbers using the said pair-wise differentially uniform sequence generator;
   updating the pair-wise differentially uniform sequence generator;
   updating the said initial vector;
   processing said at least one of a plurality of plain-text messages, and the said initial vector, and the said pair-wise differentially uniform random numbers, and the said first secret key, in the said encryptor to produce at least one of a plurality of encrypted cipher-text messages with embedded message integrity check, including separating said one of the plain-text messages into a plurality of plain-text blocks, combining each of the plain-text blocks with a respective one of said differentially uniform random numbers to generate a plurality of resultant text blocks and passing the plurality of resultant text blocks concurrently, in parallel through the series of cipher blocks, including passing each of the resultant text blocks through a respective one of the cipher blocks to produce said at least one of a plurality of encrypted cipher-text messages with embedded message integrity check in a single pass of the one of the plain-text messages through said series of cipher blocks; and
   using one or more processing units, executing an encryption program, to perform said processing.
For my part ...

I plan to freely license anything
- open-source
- software – except to the military
- research or non-commercial

The above is not a license. The actual license, in proper legal language, will be dropped to the web in the coming days.

I have the **draft language** with me, thanks to Harvard’s Cyberlaw Clinic at the Berkman Center for Internet and Society

This is a **request for comments** on the draft.
Suggestion #1

Don’t underestimate the value of theory for realizing fast and correct AE.
For the past 18 months, the NSA has been developing a high-speed encryption mode for IP packets. The mode that we designed is identical in many aspects to Jutla’s Integrity Aware Parallelizable Mode (IAPM). There is one important difference in our proposal. In the IP world, a large number of packets might arrive out of order. Integrity Aware Parallelizable Mode (IAPM) and the proposed variations incur a large overhead for out of order packets [JU 01]. Each packet requires at least the time to perform a full decryption to obtain an IV before decryption of the cipher can begin. This note describes our solution to this problem.

First, we describe the basic mode and its features. We then describe how to implement this mode for IPSec.

Dual counter mode is a hybrid of ECB mode and counter mode. Let $E$ represent encryption by a codebook of width $W$. Let $P_1, P_2, ..., P_j$ be $j$ blocks of plaintext and let $C_1, C_2, ..., C_j$ be the corresponding ciphertext. Let $f$ be a polynomial of degree $W$ for a primitive linear feedback shift register. Also, let $\{x_i\}$ be the sequence of fills generated by this polynomial. The first fill, $x_0$, is a secret shared between the two peers. This initial fill is most easily derived from the key exchange. Dual counter mode can be described as follows:

$j = \# \text{ of datablocks}$
Ironic Follow-Up

http://www.nsa.gov/research/tech_transfer/fact_sheets/dual_counter_mode.shtml
Suggestion #2

Don’t underestimate the value of implementation for understanding what’s fast or practically desirable.

<table>
<thead>
<tr>
<th>Mode</th>
<th>4K cpb</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCM</td>
<td>4.17</td>
</tr>
<tr>
<td>GCM</td>
<td>3.73</td>
</tr>
<tr>
<td>OCB2</td>
<td>1.80</td>
</tr>
<tr>
<td>OCB1</td>
<td>1.48</td>
</tr>
<tr>
<td>OCB3</td>
<td>1.48</td>
</tr>
<tr>
<td>CTR</td>
<td>1.27</td>
</tr>
</tbody>
</table>

Software Performance
Intel Core x86 i5-650 – “Clarkdale”
64-bit OS, using AES/GCM NIs

[KR11]
3. Discussion

Suggestion #3

Mind the API

and make it incremental

int ae_encrypt(
    ae_ctx *ctx,
    const void *nonce,
    const void *pt,
    int pt_len,
    const void *ad,
    int ad_len,
    void *ct,
    void *tag,
    int final);

int ae_decrypt(
    ae_ctx *ctx,
    const void *nonce,
    const void *ct,
    int ct_len,
    const void *ad,
    int ad_len,
    void *pt,
    const void *tag,
    int final);

If nonce!=NULL then a message is being initiated. If final!=0
then a message is being finalized. If final==0 or nonce==NULL
then the incremental interface is being used. If nonce!=NULL and
ad_len<0, then use same ad as last message.

Returns: if nonnegative, the number of bytes written to ct ...

Challenge the atomicity of
plaintexts, ciphertexts in defs and schemes

[Boldyreva, Degabriele, Paterson, Stam 2012]

The main part of the API from Krovetz’s
implementation of OCB3
Suggestion #4

Standardize

• A few schemes, of different types or characteristic
• The best schemes, irrespective of patents

ISO/IEC 19772:2009

1. CCM
2. EAX
3. GCM
3. SIV
5. OCB2
6. EtM
Suggestion #5

Recognize

The myth of requirements

**English**: what is *necessary*

**Industry**: what *someone* wants

**Tradeoffs** – not requirements

Eg:

- single vs. multiple underlying keys
- vector-valued AD vs. string-valued
- MRAE vs. standard AE vs. online DAE
- BB security vs. something beyond

...
A Few Research Questions

1. Can beyond-birthday-bound security be achieved **cheaply** and **generically**

2. Better **definitions**, and **constructions**, for **online** MRAE (memory usage a parameter)

3. Less atomic, more **API-centric** definitions and constructions

4. A theory useful for making **stream ciphers** into AE schemes with added cost $\ll$ **universal hashing**