OCB Mode

Phillip Rogaway

Department of Computer Science
UC Davis + CMU

rogaway@cs.ucdavis.edu
http://www.cs.ucdavis.edu/~rogaway
+66 1 530 7620 +1 530 753 0987

802.11 Presentation – Security (TG i) - Portland, Oregon

What am I?

- A cryptographer (MIT → IBM → UCD)
- Practice-oriented provable security – 1993 → present. Research program jointly envisioned with M. Bellare
- Approach applied to many cryptographic problems
- Work picked up in various standards & draft standards:
  (OEAP, DHIES, PSS, PSS-R) by (ANSI, IEEE, ISO, PKCS, SECG)

What am I **not** ?

- An expert in networking
- A businessman
- An attorney

Two Cryptographic Goals

Privacy What the Adversary sees tells her nothing of significance about the underlying message M that the Sender sent

Authenticity The Receiver is sure that the string he receives was sent (in exactly this form) by the Sender

---

Authenticated Encryption Achieves both privacy and authenticity

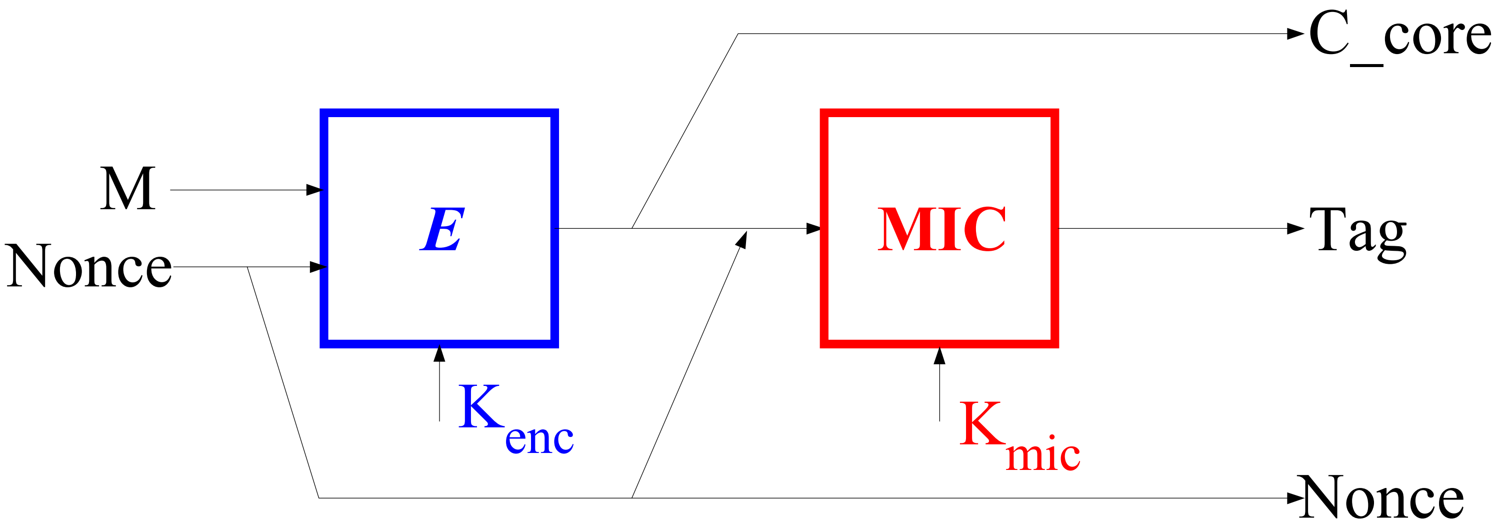
Authenticity is Essential

- You may or may not care about privacy, but you almost certainly care about authenticity: without it, an adversary can completely disrupt the operation of the network.
- The authenticity risk is higher in a wireless environment, as the adversary can easily inject her own packets.
- Standard privacy methods do not provide authenticity, and simple ways to modify them (eg, “add redundancy then encrypt”) don't work

Folklore approach. See [Bellare, Namprempre] and [Krawczyk] for analysis.

Generic Composition

Traditional approach to authenticated encryption



Glue together an Encryption scheme + Message Integrity Code (MIC)

Usually called a Message Authentication Code (MAC)

Some Algorithms for Generic Composition

	Good in HW	Fast in SW	Fast key setup	Assurance	Simplicity	Parallelizable
RC4	✓	✓			✓	
CBC-AES	✓	✓	✓	✓	✓	
CTR-AES	✓	✓	✓	✓	✓	✓
CBCMAC-AES	✓	✓	✓	✓	✓	
UMAC-AES		✓		✓		✓
new NMH/MMH MIC		✓		✓	✓	✓

Generic Composition: Conclusion

At this point in time, in this application domain,
CBC-AES / CTR-AES + CBCMAC-AES
is the natural approach for generic composition

Cost of the above, in SW

P3: about: 16 cpb + 16 cpb = 32 cpb
Eg: 54 Mb/s, 1GHz processor ≈ 22 % of processor

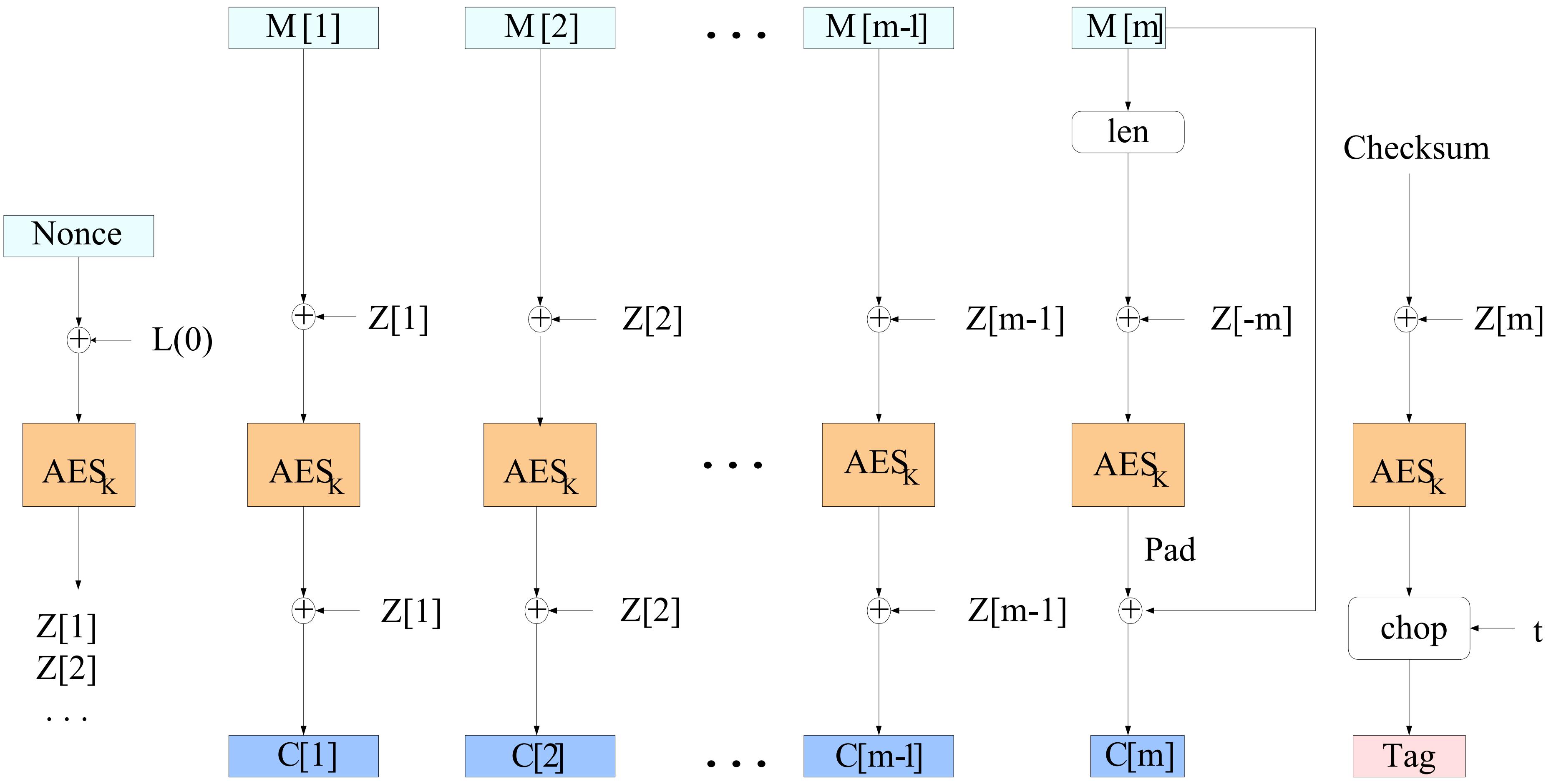
People hate paying 2× the cost to encrypt

Trying to do Better

- Numerous attempts to make privacy + authenticity cheaper
- One approach: stick with generic composition, but find cheaper privacy algorithm and cheaper authenticity algorithms
- Make authenticity an "incidental" adjunct to privacy within a conventional-looking mode
    - CBC-with-various-checksums (wrong)
    - PCBC in Kerberos (wrong)
    - [Gligor, Donescu 99] (wrong)
    - [Jutla - Aug 00] First correct solution
- Jutla described two modes, IACBC and IAPM
- A lovely start, but many improvements possible
- OCB: inspired by IAPM, but many new characteristics

What is OCB?

- Authenticated encryption: privacy + authenticity in one shot
- Uses any block cipher (you'd use AES)
- Computational cost ≈ cost of CBC
- Good in SW or HW (since AES is)
- Lots of nice characteristics designed in:
  - $\lceil |M| / 128 \rceil + 2$ block-cipher calls to encrypt M
  - Uses any nonce (needn't be unpredictable)
  - Works on messages of any length
  - Creates minimum length ciphertext
  - Uses only a single AES key, each AES keyed with it
  - Quick key setup – suitable for single-message sessions
  - Essentially endian-neutral
  - Fully parallelizable
- Provably secure: if you break OCB-AES you've broken AES


M [1]
M [2]
. . .
M [m-l]
M [m]
len
Checksum
Nonce
L(0)
Z[1]
Z[2]
Z[m-1]
Z[-m]
Z[m]
AES$_K$
AES$_K$
AES$_K$
. . .
AES$_K$
AES$_K$
AES$_K$
Pad
Z[1]
Z[2]
. . .
Z[1]
Z[2]
Z[m-1]
chop
t
C[1]
C[2]
. . .
C[m-l]
C[m]
Tag

Pseudocode for OCB-AES

algorithm OCB-Encrypt $_K$ (Nonce, M)
L(0) = AES_K (**0**)
L(-1) = lsb(L(0))? (L(0) >> 1) ⊕ Const43 : (L(0) >>1)
for i = 1 **to** 7 **do** L(i) = msb(L(i-1))? (L(i) << 1) ⊕ Const87 : (L(i-1) <<1)
Partition M into M[1] ··· M[m] // each 128 bits, except M[m] may be shorter
Offset = AES_K (Nonce ⊕ L(0))
for i=1 **to** m-1 **do**
    Offset = Offset ⊕ L(ntz(i))
    C[i] = AES_K (M[i] ⊕ Offset) ⊕ Offset
Offset = Offset ⊕ L(ntz(m))
Pad = AES_K (len(M[m]) ⊕ Offset ⊕ L(-1))
C[m] = M[m] ⊕ (first |M[m]| bits of Pad)
Checksum = M[1] ⊕ ··· ⊕ M[m-1] ⊕ C[m]0* ⊕ Pad
Tag = first t bits of AES_K(Checksum ⊕ Offset)
return C[1] ··· C[m] || Tag

Assembly Speed

Data from **Helger Lipmaa** www.tcs.hut.fi/~helger helger@tcs.hut.fi

// Best Pentium AES code known. Helger's code is for sale, btw.

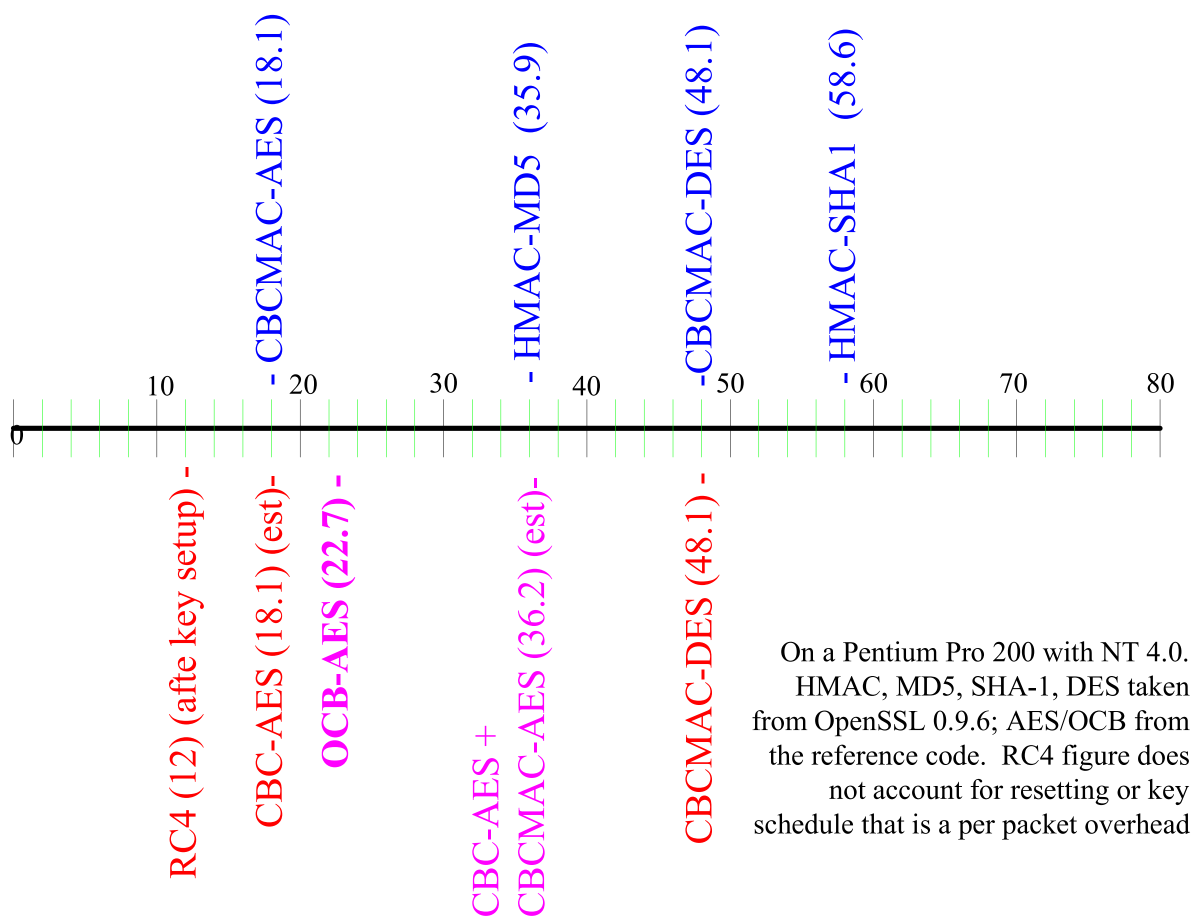
OCB-AES	16.9 cpb	(271 cycles)
CBC-AES	15.9 cpb	(255 cycles)
ECB-AES	14.9 cpb	(239 cycles)
CBCMAC-AES	15.5 cpb	(248 cycles)

6.5 % slower (OCB-AES vs. CBC-AES)

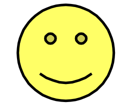
The above data is for 1 Kbyte messages. Code is pure Pentium 3 assembly. The block cipher is AES-128. Overhead so small that AES with a C-code CBC wrapper is slightly more expensive than AES with an assembly OCB wrapper.

C-Language Speed

Data courtesy of **Jesse Walker**, Intel



On a Pentium Pro 200 with NT 4.0. HMAC, MD5, SHA-1, DES taken from OpenSSL 0.9.6; AES/OCB from the reference code. RC4 figure does not account for resetting or key schedule that is a per packet overhead

Why I like OCB 

- **Ease-of-correct-use**. Reasons: all-in-one approach; any type of nonce; parameterization limited to block cipher and tag length
- **Aggressively optimized**: ≈ optimal in many dimensions: key length, ciphertext length, key setup time, encryption time, decryption time, available parallelism; SW characteristics; HW characteristics; …
- **Simple but sophisticated**
- Ideal setting for **practice-oriented provable security**

More on Provable Security

- Provable security begins with [Goldwasser, Micali 82]
- Despite the name, one doesn't really *prove* security
- Instead, one gives *reductions*: theorems of the form

  If a certain primitive is secure

  then the scheme based on it is secure

  For us:

  If AES is a secure block cipher

  then OCB-AES is a secure authenticated-encryption scheme

  Equivalently:

  If some adversary **A** does a good job at breaking OCB-AES

  then some comparably efficient **B** does a good job to break AES
- Actual theorems quantitative: they measure how much security is "lost" across the reduction.

OCB Theorem
(Informal version)

Suppose there is an adversary **A** that breaks the privacy **or** the authenticity of OCB-E (where E is an n-bit block cipher) with:

time = t   total-number-of-blocks = σ advantage = ε

Then there is an adversary **B** that breaks block cipher E with:

time $\approx$ t   number-of-queries $\approx \sigma$ advantage $\approx \varepsilon - 1.5\,\sigma^2 / 2^n$

- Breaking the privacy of OCB-E: The ability to distinguish OCB-E encrypted strings from random strings.
- Breaking the authenticity of OCB-E: The ability to produce a forged ciphertext.
- Breaking the block cipher E: The ability to distinguish E_K, E_K^{-1} from π, π^{-1}

What Provable Security Does, and Doesn't, Buy You

+ Strong evidence that scheme does what was intended
+ Best assurance cryptographers know how to deliver
+ Quantitative usage guidance

- An absolute guarantee
- Protection from issues not captured by our abstractions
- Protection from usage errors
- Protection from implementation errors

Adoption Issues

- Scheme too new / might be wrong – Largely obviated by provable security
- Stability - OCB (Apr 1) has not and will not change. Good schemes last forever
- NIST does something else – If you care, send mail: EncryptionModes@nist.gov
- Export - Non-issue due to EAR 740.18(b)(4)
- Licensing – Next slides

Do I have a Patent?

- I filed patent applications covering OCB (12 Oct 00, 9 Feb 01)
- I will license the resulting patent(s) under fair, reasonable, non-discriminatory terms
- Letter of Assurance provided to the IEEE (3 May 01)
- My commitment goes well beyond the IEEE policy:
  - Public pricing, public license agreement
  - One-time fee (paid-in-full license)
  - I am committed to making this simple and easy for everyone
  - For further info: see “Licensing” on the OCB web page

Does Anyone **Else** Have a Patent OCB Would Infringe Upon?

Do keep in mind the proviso from slide 2: I'm not a lawyer!

- **At present:** No

In the future: No way to know

- **Jutla / IBM**
  - Has patent filing before me, including IAPM
  - IAPM resembles OCB.
  - But there are major differences which would have made it difficult to make claims for IAPM that read against OCB
  - **My conclusion:** IBM could come to hold a relevant patent, if their attorneys were lucky or insightful

Does Anyone Else Have a Patent, cont.

- **Gligor/VDG**
  - Has patent filings before me and IBM
  - [GD, Aug 00] has an authenticated-encryption scheme, XCBC, but it does not resemble OCB
  - I know of no idea from [GD] that I used in OCB
  - **My conclusion:** I consider it unlikely that Gligor/VDG will come to hold a valid patent that reads against OCB

- **My overall conclusion**
  - A company would be behaving with appropriate diligence to license from me - and no one else - at this time
  - The IEEE would be behaving with appropriate diligence to request patent-assurance letters from IBM and VDG, just in case

For More Information

- OCB web page → www.cs.ucdavis.edu/~rogaway
  Contains FAQ, papers, reference code, assurance letter, licensing info...
- Feel free to call or send email
- Upcoming talks:
  - NIST modes-of-operation workshop (Aug 24, Santa Barbara)
  - MIT TOC/Security seminar (Oct ??, Cambridge)
  - ACM CCS conference (Nov 5-8, Philadelphia)
- Grab me now or at CRYPTO (Aug 20-23)

Anything Else ???