

Combining Message Authentication and Encryption

Greg Rose

QUALCOMM Australia

ggr@qualcomm.com

What I'm Going To Say...

- **What are Encryption and Message Authentication?**
- **Why would you want to combine them?**
- **Some obvious ways**
 - ... that don't work
- **One pass block cipher methods**
- **Two pass block cipher methods**
- **Combined stream ciphers**
- **Summary**

What is Encryption?

- **Encryption scrambles the data**
 - Under control of a key
- **So that the attacker can't figure out what the data was from looking at it**
- **Two general types:**
 - Block ciphers operate on large chunks
 - Stream ciphers operate (effectively) on single bits
- **Two types of attackers:**
 - Passive “eavesdroppers”
 - Active “modifiers”

What is Message Authentication?

- **Also called “message integrity”**
- **Provides assurance that the message:**
 - **Came from who you thought it came from**
 - **Wasn’t changed along the way.**
- **Usually done by calculating a “keyed hash”, called a MAC (Message Authentication Code)**
 - **Sometimes called “tag”**

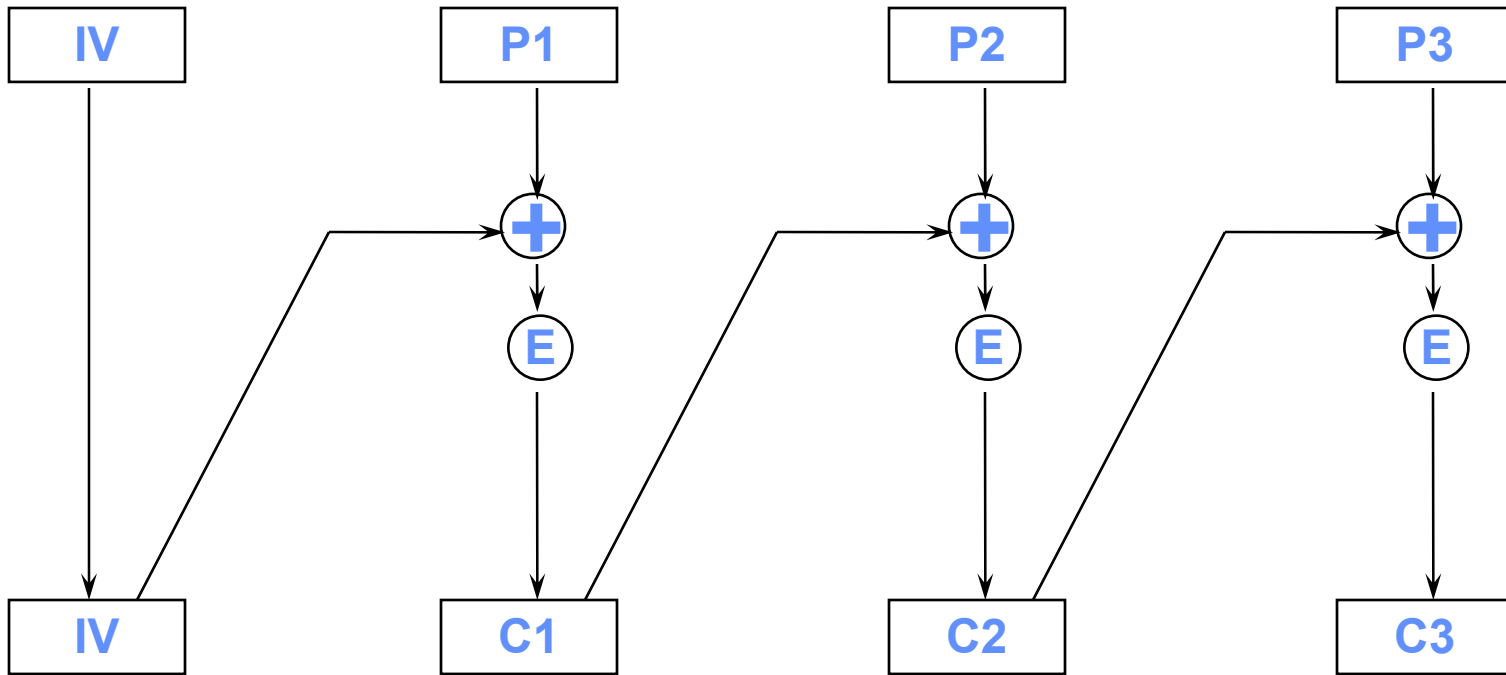
Voodoo

- **I thought I'd better just explain some of these acronyms, before going too far**
- **MAC – Message Authentication Code**
- **CBC – Cipher Block Chaining**
- **Counter Mode**
- **CBC-MAC – CBC mode used as a MAC**
- **HMAC – Hash based MAC**
- **Nonce – a Number used only ONCE**
- **IV – Initialization vector**

Nonces versus IVs

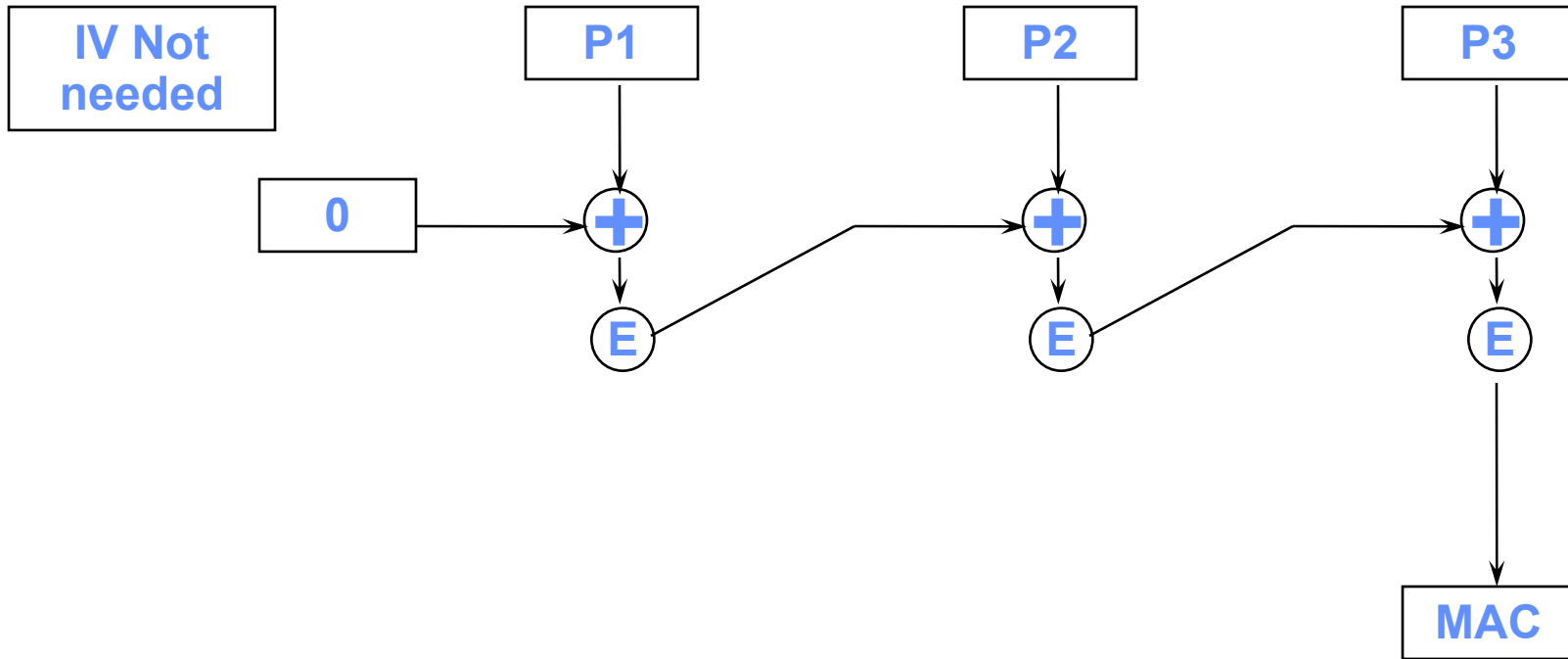
- **Nonces are nice:**
 - Can be smaller than a whole block
 - Only requirement is uniqueness, that is, never use the same one twice
 - Can be a timestamp
 - Can be a counter
 - Can be derived from the LSBs of a small counter
 - Can be derived from the environment somehow
- **IVs are not so easy:**
 - Must be pseudorandom and unpredictable

CBC Diagram

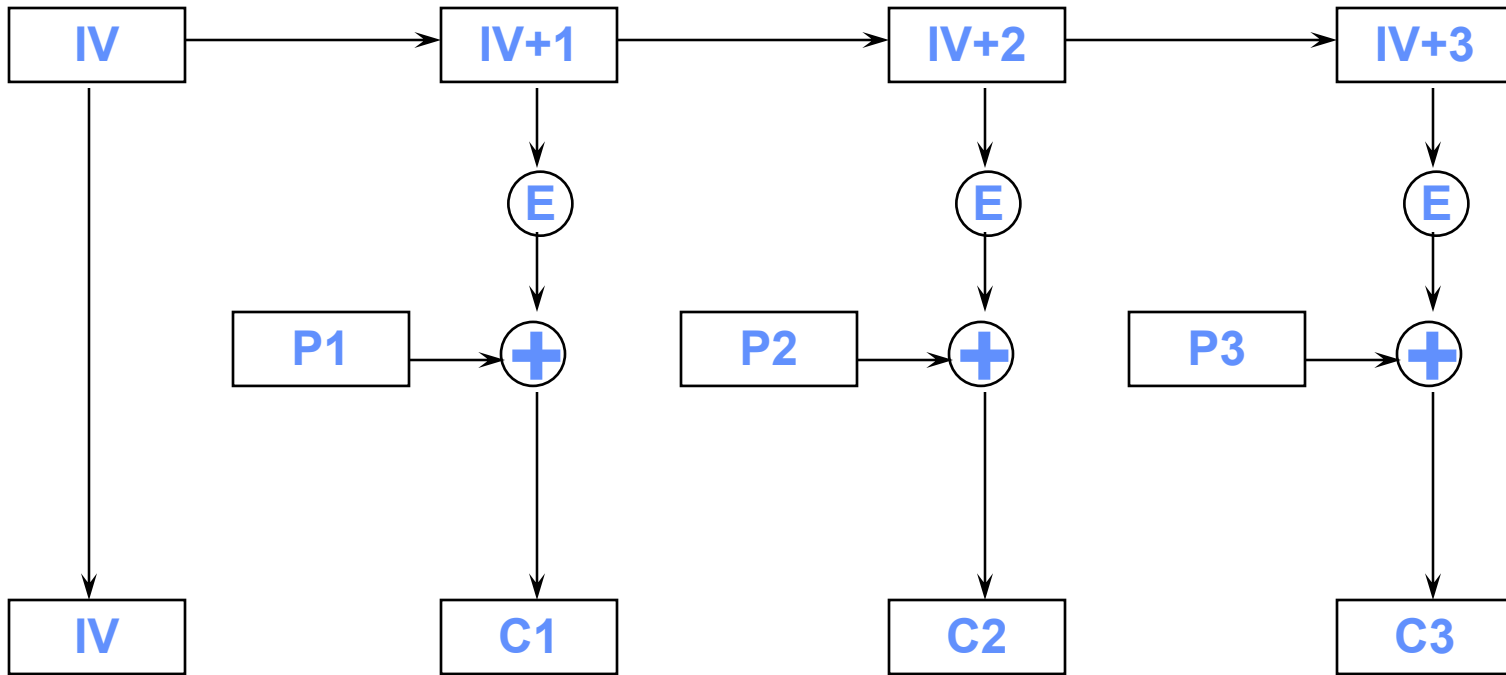


IV used to ensure that same messages encrypt differently.

CBC-MAC Diagram

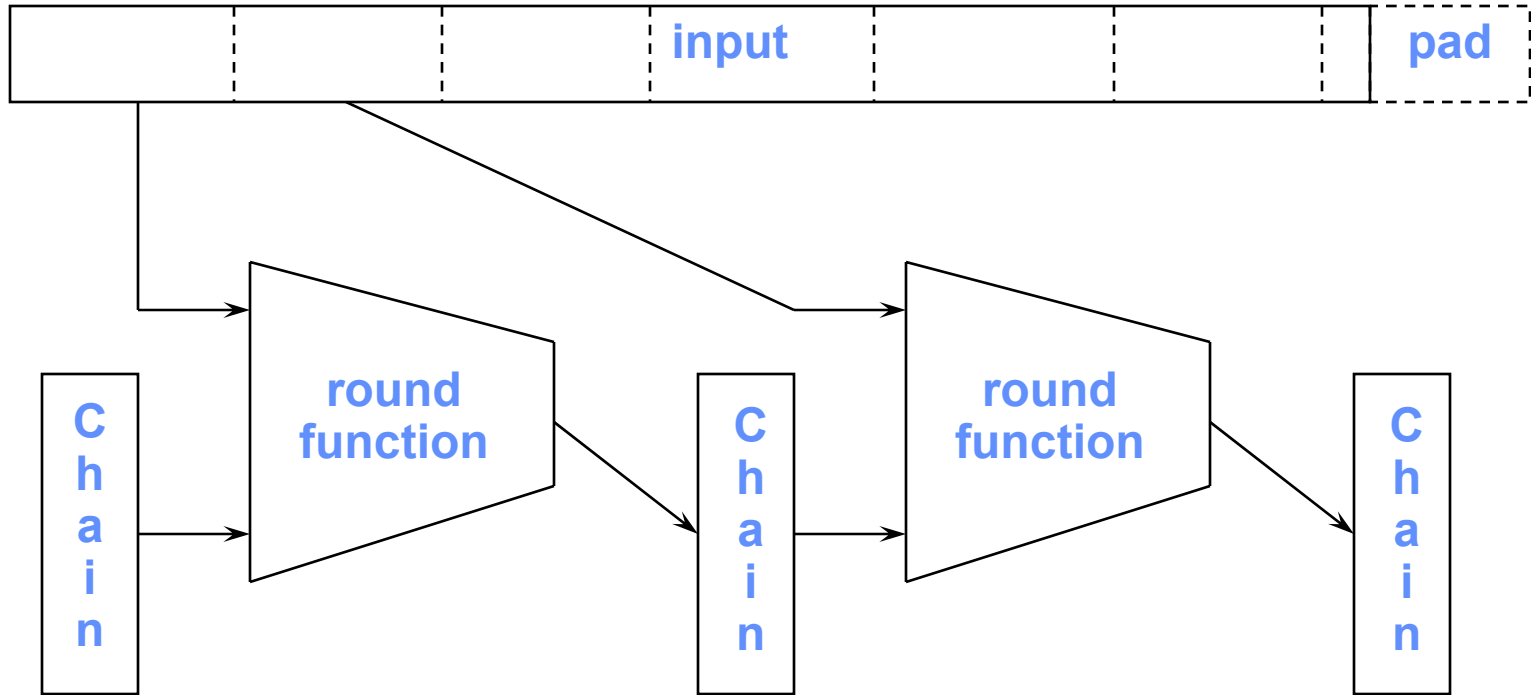


Counter Mode Diagram



IV can be a Nonce, moved left to make room for the counter

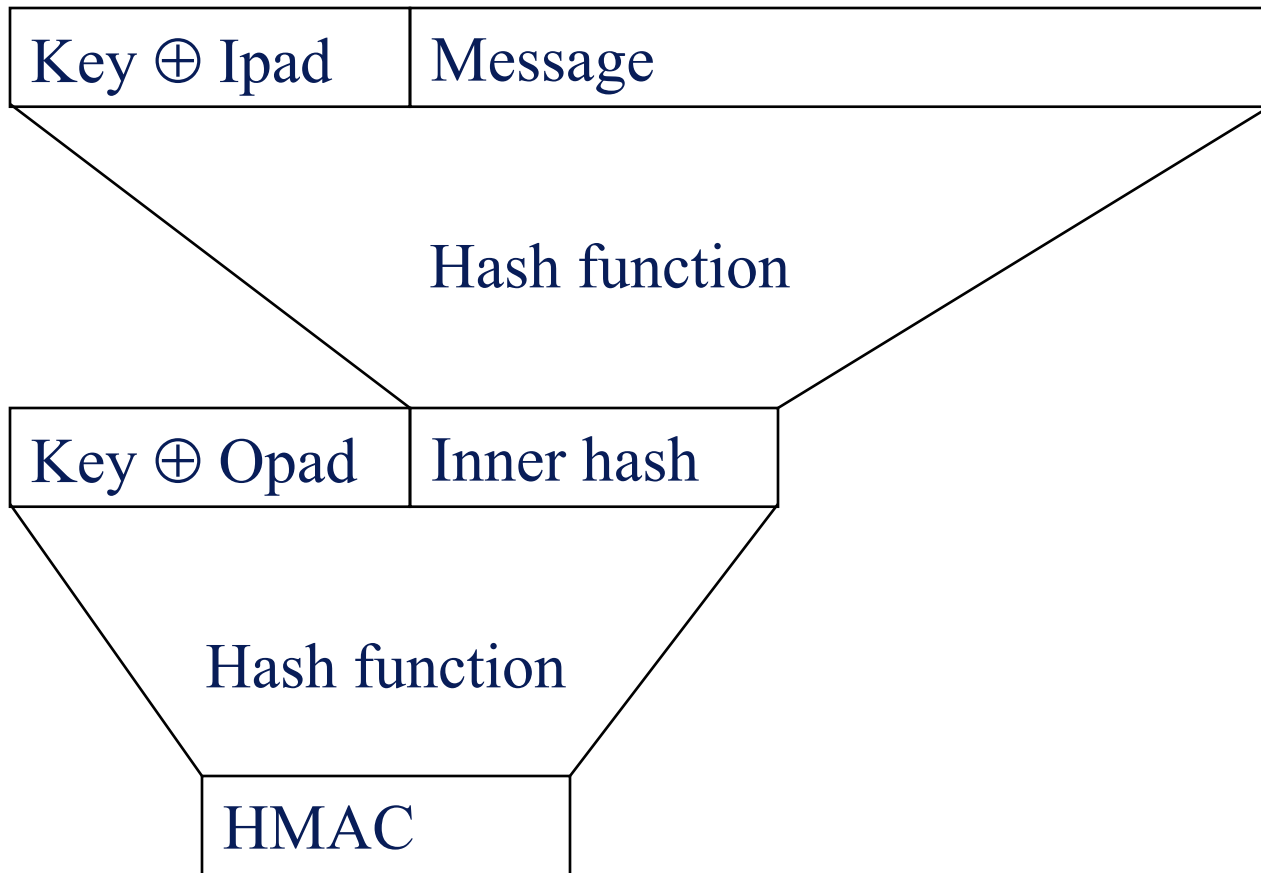
Hash structure diagram



Keyed MAC -- HMAC

- Turning hash into MAC sounds easy, but isn't
 - Extension attacks
- HMAC based on NMAC
 - NMAC “provably secure”
- $MAC = H(k \oplus opad, H(k \oplus ipad, data))$
- opad and ipad just simulate different hash functions
- MD5 has problem but HMAC-md5 avoids it

HMAC diagram



Why combine Encryption and Message Authentication?

- **Some people think that**
 - because you can't understand the message...
 - and if you modify it you don't know what you'll get
 - ... encryption prevents mucking with messages
- ***Wrong!***
 - Bellovin's cut-and-paste attacks on IPsec
 - early ATM attack in England
- **More recently...**

Why? (cont)

- **Some people think that:**
 - because you can't decrypt the data ...
 - ... encryption alone provides data privacy.
- ***WRONG!***
 - Bellare and Namprempre's "Encrypt then Authenticate" paper, and Krawczyk
 - Some of Bellovin's examples
 - 802.11 WEP (see Borisov, Goldberg & Wagner)
 - all are examples where modifying the message allows you to con someone else into decrypting it for you

Still Why?

- To get data confidentiality in the face of an active attack:

You must also have message integrity!

- So, you have to have it. Why a combined mechanism?
 - Too easy to make mistakes with cobbled-together solutions
 - ... again, *Encrypt then Authenticate*
 - possible efficiency gains

Is Message Integrity enough?

- ... of course not, or this slide wouldn't be here
- **Something must prevent replay of valid messages!**
 - Alice tells Bank to transfer \$10 to Mal
 - Mal taps wire, then replays the message a million times
- **“*Freshness*” is required**
 - Something that varies but is checkable by recipient
 - Timestamp, sequence number, response to challenge, ...
 - Not a cryptographic algorithm issue

Enough yet?

- **No, attacker can always prevent delivery of the message**
- **Don't design protocol with "Are you sure?" messages with back consequences**
- **"Abort countdown?" even worse**

Enough why, now *what?*

- **Think of IPsec**
- **want to have message integrity for whole packet**
 - including source, destination, etc.
- **but only want to encrypt the payload**
 - otherwise delivering it gets tricky
- **Want a flexible mechanism that allows message integrity over both plaintext headers and ciphertext**
- **New name: "AEAD", Authenticated Encryption with Associated Data (Rogaway and Wagner)**

Early attempts

- **Block ciphers give "a bit of message integrity"**
 - if two fields fall in one block, hard to modify one without modifying the other
- **Some chaining modes propagate errors**
 - same as above, just a bit bigger
- **Encrypt with CBC, MAC with CBC-MAC**
 - each part is secure
 - put them together, becomes insecure
- **"The Holy Grail" for a while**
 - Many thought it was impossible to do "one pass"

Block Cipher based (1)

- **Suddenly, in 2001:**
 - IAPM (Jutla, IBM)
 - XCBX (Gligor & Donescu)
 - OCB (Rogaway et. al.)
- **The above encrypt/authenticate whole messages**
 - HR mode (Hawkes & Rose) extends to Partial Encryption with Message Integrity
- **High throughput parallelizable modes**
- **All patented**

How do they work?

- **All similar in concept**
- **All are proven correct based on reasonable assumptions about the security of the underlying block cipher**
- **combines a secret counter with plaintext, then encrypts**
- **MAC is a checksum of intermediate results, then encrypted itself**
- **(This is a gross oversimplification!)**

Block Cipher based (2)

- **Because of patents, much of the community has backed off from use of those modes.**
- **Instead, go back to using block cipher in counter mode for encryption**
- **For message integrity, use something either:**
 - **Based on the block cipher, for speed and commonality with the encryption, or...**
 - **Use a very fast universal hash function then encrypt it.**
- **AES-CBC + HMAC-SHA-1 (ie. IPsec, TLS) is secure, just too slow.**
 - **also can't be parallelized in any way**

New "combined" modes

- **CCM**
- **EAX**
- **CWC**

- **Of course there's the old, trusty, combination of CBC mode encryption and HMAC**

CCM

-
- **Whiting, Ferguson and Housley**
 - **"Counter with CBC-MAC"**
 - **Encrypts with counter mode**
 - **Uses CBC-MAC for Message Authentication Code**
 - **Used in 802.11 next generation security**
 - **Prepends header with length of message**
 - **makes it hard to process a stream**
 - **Very "bit fiddly"**

EAX

-
- **Bellare, Rogaway and Wagner**
 - **Encrypt then Authenticate then X(trans)late**
 - **Encryption is Counter Mode**
 - **MAC is just a little more complicated**
 - based on CBC-MAC variant called OMAC
 - length field gets mixed in at the end
 - complicated padding to avoid extending data
 - **(Coined the term AEAD)**

CWC

- **Kohno, Viega, Whiting**
- **"Carter-Wegman and Counter"**
- **Encryption is same old Counter Mode**
- **MAC is Carter-Wegman construct:**
 - **treat the input data as 96-bit coefficients of a polynomial**
 - **evaluate polynomial with $x = \text{key}$**
 - **all modulo $2^{127}-1$ (prime, easy arithmetic)**
 - **encrypt the result**
- **Potentially faster than the others**
 - **big multiplications a bit faster than encryption**

Stream Ciphers

- **Note that all of the above use Counter Mode for the bulk encryption. This just makes the block cipher into a stream cipher**
- **But stream ciphers can be significantly more efficient!**
- **New rules required:**
 - **Never allowed to re-use nonces**
 - **MUST discard messages failing authentication, which sounds obvious, but some people try to use them.**
 - **"A bit in the message says whether MAC is required or optional..."**

Helix

-
- **Ferguson, Whiting, Schneier, Kelsey, Lucks, Kohno**
 - **5 state words, combined using shifts, addition and XOR in a helical pattern**
 - **Each block produces one word of keystream, and accepts one word of plaintext for MAC**
 - **At end of block, run it for a little while longer, then generate keystream for MAC.**

SOBER-128

- **Rose & Hawkes**
- **Based on earlier SOBER ciphers, particularly SOBER-t32.**
- **MAC aspect similar to Helix**
- **17-word LFSR state**
- **5 word nonlinear function, 1 word output**
- **plaintext word added into LFSR using nonlinear function**
- **Generate MAC by mixing some more then output keystream**

Performance

- **Brian Gladman's figures, Pentium 4:**
 - **CCM: $28 * \text{header_length} + 48 * \text{message_length}$**
 - **CWC: $40 * \text{header_length} + 66 * \text{message_length}$**
 - **EAX: $23 * \text{header_length} + 48 * \text{message_length}$**

- **My figures, Centrino @ 1.5Ghz:**
 - **Helix: 30 cycles per byte (49 MB/s)**
 - **SOBER-128: 8.6 cycles per byte (175 MB/s)**

- **Not strictly comparable!**

Conclusion

- **Must use message integrity for secure communication**
- **Wasn't easy to get this right (WEP, SSLv2)**
- **There's now a choice of possible mechanisms**

- **Use one of them!**

URLs

-
- <http://people.qualcomm.com/ggr/...>
 - <...Enc+MAC-Slides.pdf>
 - **...Enc+MAC-Paper.pdf**