

# On the security of CBC-MAC schemes

Chris Mitchell

Information Security Group

Royal Holloway, University of London

[C.Mitchell@rhul.ac.uk](mailto:C.Mitchell@rhul.ac.uk)

<http://www.isg.rhul.ac.uk/~cjm>

# Contents of talk

1. **CBC-MACs**
2. EMAC and ARMAC
3. New CBC-MAC schemes
4. RMAC
5. The XCBC family
6. Conclusions

# Purpose of MACs

- Used to protect integrity and guarantee origin of data strings.
- Sender and verifier share a secret key (of  $k$  bits).
- Sender inputs data and key to MAC algorithm – output is MAC (short string of bits) which is sent/stored with data.
- Verifier recomputes MAC using received message and secret key and compares.

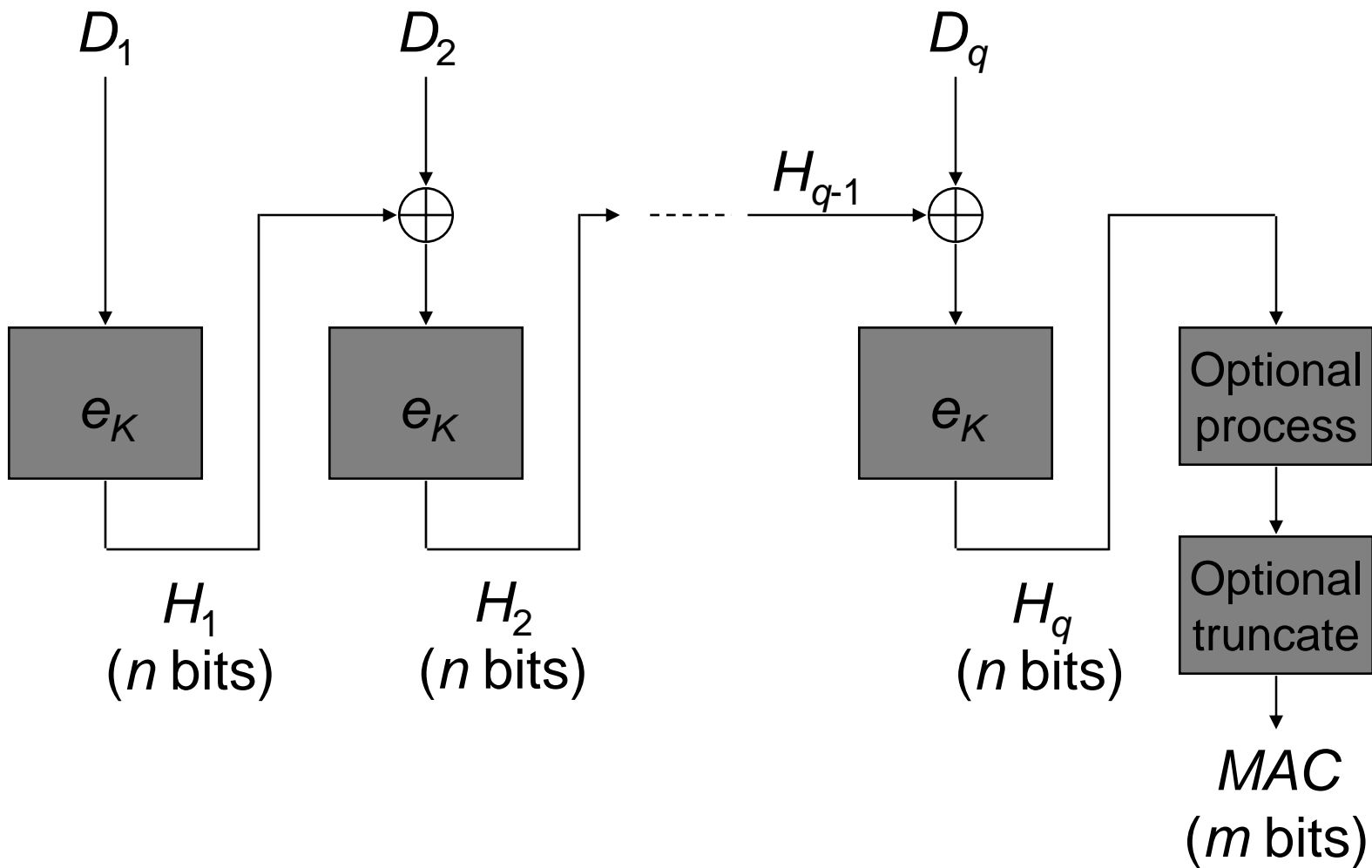
# CBC-MACs

- A CBC-MAC is a particular (very popular) type of MAC.
- Computed using a block cipher in CBC (Cipher Block Chaining) mode.
- Write  $e_K(P)$  for block cipher encryption of block  $P$  ( $n$  bits) using secret key  $K$  ( $k$  bits).
- Similarly, write  $d_K(C)$  for block cipher decryption of block  $C$  using key  $K$ .

# CBC-MAC operation

- Divide and pad data to be MACed into  $n$ -bit blocks  $D_1, D_2, \dots, D_q$  ( $n$  is block length of block cipher, e.g.  $n = 64$  for DES).
- The MAC is computed by:
  - put  $H_1 = e_K(D_1)$ ,
  - for  $i = 2, 3, \dots, q$ : put  $H_i = e_K(D_i \oplus H_{i-1})$ .
- $H_q$  is then subject to an ‘optional process’ and truncated to  $m$  bits to give the MAC.

# CBC-MAC calculation



# Padding

- Three well known padding methods:
  - **Method 1:** add minimum no. of zeros to make a whole number of blocks.
  - **Method 2:** add single one followed by zeros to make a whole number of blocks.
  - **Method 3:** right-pad with zeros as necessary. Left-pad with extra  $n$ -bit block containing binary representation of bit-length of unpadded string.
- Padding not sent with MACed message.

# Trailing zeros forgeries

- Padding Method 1 allows attacker to add or delete trailing zeros from a message without changing the MAC. **A forgery attack.**
- Arises from fact that Padding Method 1 is not a one-to-one function, i.e. up to  $n$  unpadded messages map to the same padded message.
- Motive for introduction of Method 2.



# Need for optional process

- Suppose a CBC-MAC is computed with no optional process and no truncation (SMAC).
- Suppose we have the MACs for two one-block messages:

$$MAC_1 = eK(D_1), \quad MAC_2 = eK(D_2).$$

- Then  $MAC_2$  is a valid MAC on the two block message:  $D_1 || D_2 \oplus MAC_1$ .
- Need to add optional process (or padding method 3) to avoid this 'cut and paste'  
**Forgery attack.**

# Contents of talk

1. CBC-MACs
- 2. EMAC and ARMAC**
3. New CBC-MAC schemes
4. RMAC
5. The XCBC family
6. Conclusions

# Optional processes

- Two well-known optional processes:
  - choose a key  $K_1$  and compute:

$$H_q'' = e_K(d_{K_1}(H_q)),$$

- choose a key  $K_1$  and compute:

$$H_q' = e_{K_1}(H_q).$$

- First method results in ANSI Retail MAC (ARMAC) when block cipher = DES
- Second method often called EMAC.

# Standard CBC-MACs

- ISO/IEC standard for CBC-MACs (ISO/IEC 9797-1: 1999) contains 6 schemes.
- First three are as follows:
  - Alg. 1 = CBC-MAC with no optional process (SMAC).
  - Alg. 2 = CBC-MAC with optional process as single extra encryption (EMAC).
  - Alg. 3 = CBC-MAC with optional process as extra decryption and encryption (i.e., triple encrypt last block) (ARMAC).

# EMAC security

- EMAC has a proof of security (Petrank & Rackoff, 2000).
- For block ciphers with large enough  $n$  and  $k$  (128 or more), EMAC is sound choice – with padding method 2 or 3.
- For block ciphers with small  $k$  (e.g. DES:  $k=56$ ), EMAC insecure, because of simple meet-in-the-middle key recovery attack.
- Attack complexity:  $O(2^k)$  encryptions with 1 known MAC.

# ARMAC security

- Problems with EMAC (and SMAC), combined with desire to use DES, motivates design of ARMAC.
- ARMAC seems much more resistant to key recovery attacks than EMAC (no proof however).
- Key recovery attack either requires triple DES break ( $2^k$  encryptions +  $2^k$  storage) or large number ( $2^{n/2}$ ) of known MACs combined with single DES break ( $2^k$  encryptions).

# Contents of talk

1. CBC-MACs
2. EMAC and ARMAC
- 3. New CBC-MAC schemes**
4. RMAC
5. The XCBC family
6. Conclusions

# Rationale

- The standardisation of a block cipher (AES) with larger  $n$  and  $k$ , means that it seems appropriate to re-examine ways in which we use block ciphers.
- Modes of operation and commonly used CBC-MAC schemes are quite 'old' designs.
- Can we do better?



# NIST process

- NIST has an ongoing project to produce new 'modes' standards for DES.
- **Objective:** produce combined encryption + integrity mode (proposal for review in NIST Special Publication 800-38C, September 2003).
- **Objective:** CBC-MAC standard for AES.
- NIST activity mirrored in ISO, where ISO/IEC 9797-1 currently under review, and Data Encapsulation Mechanisms (DEMs) work just starting (DEM = combined encryption/integrity).

# Candidate schemes

- A number of candidate CBC-MAC schemes have been proposed, including:
  - RMAC (Jaulmes, Joux and Valette, 2002),
  - XCBC (Black and Rogaway, 2000), and
  - TMAC and OMAC (Iwata and Kurosawa, 2003).

# Contents of talk

1. CBC-MACs
2. EMAC and ARMAC
3. New CBC-MAC schemes
- 4. RMAC**
5. The XCBC family
6. Conclusions

# RMAC

- RMAC operates as follows.
- Two block cipher keys required ( $K, K_1$ ).
- To generate a MAC first generate a random salt  $R$  (of  $k$  bits).
- Then, using the model previously described, RMAC involves the optional process:

$$H_q' = e_{K_1 \oplus R}(H_q).$$

# Rationale of RMAC

- Typically, a CBC-MAC scheme will be subject to forgery attacks requiring  $O(2^{n/2})$  known/chosen MACs (based on 'birthday paradox' probability).
- For 'short block' block ciphers (e.g. 3DES, IDEA, ... with  $n = 64$ ) this is sometimes a little 'close' to what is possible.
- RMAC objective is to offer greater resistance to 'birthday' forgery attacks.

# NIST draft

- RMAC was included in NIST special publication 800-38B (November 2002) – essentially a draft standard.
- At that time RMAC was clearly the leading candidate for standardisation.

# Reaction to 800-38B

- The release of NIST SP 800-38B provoked a large number of negative comments.
- The result is that RMAC is no longer being seriously considered for NIST adoption.
- The original SP 800-38B and the main comments are available for download at the NIST website.

# A simple observation

- Suppose know one RMAC ( $M$  say) for data  $D$  (using salt  $R$ , say).
- Request another MAC ( $M'$  say) for the same data  $D$  (uses salt  $R'$  say).
- Then immediately know that:  
$$d_{K_1 \oplus R}(M) = d_{K_1 \oplus R'}(M').$$
- Enables exhaustive search for  $K_1$  with complexity  $2^k$  (and just 2 known MACs).
- This contradicts claims in SP 800-38B.



# Some attacks on RMAC

- In (Knudsen & Mitchell, J. Crypt., to appear) a series of *partial key recovery* attacks on RMAC are presented.
- Enable one of the two RMAC keys ( $K_1$ ) to be recovered with much less than  $2^k$  work.
- Once  $K_1$  is known, very simple forgery attacks become possible (based on 'cut and paste' attack).

# Contents of talk

1. CBC-MACs
2. EMAC and ARMAC
3. New CBC-MAC schemes
4. RMAC
- 5. The XCBC family**
6. Conclusions

# XCBC

- XCBC, another CBC-MAC scheme, was proposed by Black & Rogaway in 2000.
- Objective was to define a provably secure CBC-MAC which minimises number of block cipher encryptions/decryptions.
- Address fact that EMAC + pad method 2 can involve 2 'extra' encryptions by comparison with SMAC + pad method 1.

# XCBC operation I

- XCBC does not quite fit the general CBC-MAC model presented earlier.
- Use padding method 2 if data string needs padding; otherwise do not pad.
- Avoid ambiguity problems by computing MAC differently depending on whether or not padding was performed.
- Three keys:  $K$ ,  $K_1$  and  $K_2$  ( $K$  has  $k$  bits, &  $K_1$ ,  $K_2$  have  $n$  bits).

# XCBC operation II

- If no padding then exor  $K_1$  with  $D_q$  (last data block).
- If padding used then exor  $K_2$  with  $D_q$ .
- Then compute SMAC on (modified) data using key  $K$ .

# XCBC properties

- Same number of encryptions as SMAC with padding method 1, yet forgery problems removed.
- Proof of security exists.
- Hence optimally efficient with respect to block cipher operations, BUT largish key (384 bits for AES).

# TMAC

- To reduce key size, Kurosawa and Iwata (2003) proposed TMAC (T for 'two key') using keys  $K$  (of  $k$  bits) and  $K'$  of  $n$  bits.
- Derive  $K_1$  and  $K_2$  from  $K'$  by putting  $K_2 = K'$  and  $K_1 = u.K'$  where multiplication takes place in  $GF(2^n)$ .
- Compute MAC as for XCBC.
- TMAC still has a proof of security.

# OMAC

- Iwata and Kurosawa (2003) have recently proposed OMAC (O for ‘one-key’) using just one key  $K$  (of  $k$  bits).
- Derive  $K'$  from  $K$  by setting  $K' = e_K(0^n)$ .
- Then derive  $K_1$  and  $K_2$  from  $K'$  as for TMAC.
- Finally, compute MAC as for XCBC.
- OMAC again has a proof of security.



# NIST statement

- NIST have not yet published a new draft on CBC-MACs, but have indicated that they are leaning towards OMAC.
- There is also an 'open call' for comments on all CBC-MAC schemes.
- Some comments exist on NIST website.
- Thus, now is the time to provide input to NIST!

# Partial key recovery attack on TMAC

- Sung, Hong & Lee (2003) described attack against TMAC which allows recovery of  $K'$  given  $O(2^{n/2})$  known/chosen MACs and trivial computation (no key search).
- Recovering  $K$  still requires  $2^k$  work, and proof of security not challenged.
- However, knowing  $K'$  does make very trivial forgeries possible.

# OMAC attacks

- The TMAC attack works against OMAC, as does a further (different) attack, both allowing recovery of  $K'$  given  $O(2^{n/2})$  known/chosen MACs.
- As Iwata has pointed out, this is no longer a partial key recovery attack, since  $K'$  is not part of the key (but is derived from it) – unlike TMAC.
- Nevertheless, recovery of  $K'$  would allow very trivial forgeries.

# What does it mean?

- These attacks do not contradict proofs of security for OMAC and TMAC.
- None of the proofs say anything about security once an attacker has  $O(2^{n/2})$  known MACs.
- However, it is arguable that one should still be concerned about what happens at the ‘boundaries’ of the security proof.

# Contents of talk

1. CBC-MACs
2. EMAC and ARMAC
3. New CBC-MAC schemes
4. RMAC
5. The XCBC family
- 6. Conclusions**

# Where next?

- The main choice right now (for NIST) would appear to be between EMAC and OMAC.
- Both have similar provable security.
- OMAC is more efficient.
- However EMAC appears stronger just outside envelope of security proof.
- Views are needed, both for NIST and in near future for ISO.