History of message integrity techniques

Chris Mitchell
17th January 2008

Contents of talk

1. CBC-MACs
2. Standardised CBC-MACs
3. EMAC and ARMAC
4. New CBC-MAC schemes
5. RMAC
6. The XCBC family
7. Other schemes
8. Conclusions
Scope of talk

• We are concerned in this talk with message integrity techniques based on shared secret keys, i.e. the class of symmetric cryptosystems known as MACs.
• Digital signatures (examples of asymmetric cryptography) can also be used to provide message integrity, but we do not discuss them.

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. CBC-MACs are the main focus of this talk.

• 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 $K$.

Simple CBC-MAC operation

• Divide and pad data to be MACed into $n$-bit blocks $D_1, D_2, \ldots, D_q$ ($n$ 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 $i = 2, 3, \ldots, q$: put $H_i = e_K(D_i \oplus H_{i-1})$.

• $H_q$ is then truncated to $m$ bits to give the MAC.
Simple CBC-MAC calculation

\[ D_1 \leftarrow e_K \\
H_1 \ (n \text{ bits}) \]

\[ D_2 \leftarrow e_K \\
H_2 \ (n \text{ bits}) \]

\[ \ldots \quad H_{q-1} \leftarrow e_K \\
H_q \ (n \text{ bits}) \]

Optional truncate

\[ H_q - 1 \]

MAC \ (m \text{ bits})

Background

- Idea dates (at least) back to 1970s.
- Simple CBC-MACs have been used since that time, most widely with DES and (more recently) triple DES.
- First appeared in a standard in 1980.
- Padding method needed – originally done simply by adding the minimum number of zeros necessary.
Attack types

- There are two main types of attack on a MAC scheme:
  - **Forgery attacks**, in which an attacker is able to generate a new valid (message, MAC) pair;
  - **Key recovery attacks**, where an attacker can learn the secret key in use (of course, a successful key recovery attack enables arbitrary numbers of forgeries).

- There are also variants of the basic attacks, including **chosen message forgery attacks**.

Contents of talk

1. CBC-MACs
2. **Standardised CBC-MACs**
3. EMAC and ARMAC
4. New CBC-MAC schemes
5. RMAC
6. The XCBC family
7. Other schemes
8. Conclusions
Evolution of CBC-MACs

- The Simple CBC-MAC (SMAC) was the first used MAC scheme, and is still in use today.
- SMACs are provably secure if the message length is fixed.
- However, there are problems both with the method used to generate the MAC and also with the padding method if message lengths are variable.

Enhanced CBC-MAC operation

- Divide and pad data to be MACed into $n$-bit blocks $D_1, D_2, \ldots, D_q$ ($n$ 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 $i = 2, 3, \ldots, q$: 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

\[ D_1 \xrightarrow{e_K} H_1 \]  
\[ D_2 \xrightarrow{e_K} H_2 \]  
\[ \ldots \xrightarrow{H_{q-1}} \]  
\[ D_q \xrightarrow{e_K} H_q \]

Optional process

Optional truncate

\[ MAC (m \text{ bits}) \]

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:
  $$\text{MAC}_1 = e_K(D_1), \quad \text{MAC}_2 = e_K(D_2).$$
• Then MAC$_2$ is a valid MAC on the two block message:
  $$D_1 \ || \ D_2 \oplus \text{MAC}_1.$$
• Need to add optional process (or padding method 3) to avoid this ‘cut and paste’ Forgery attack.
Contents of talk

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

Optional processes

• Two well-known optional processes:
  – choose a key $K_1$ and compute:
    $$H_q'' = e_{K_1}(d_{K_1}(H_q)),$$
  – 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 (1999)

- 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).

Forgery attacks

- Both EMAC and ARMAC are subject to possible forgery attacks if the attacker has access to \( 2^{n/2} \) (message, MAC) pairs.
- Relies on the fact that it is likely that two of these pairs will have the same MAC.
- This will arise because of a ‘birthday probability’ internal collision.
- A pair of messages with the same MAC can then be used to construct forgeries.
Contents of talk

1. CBC-MACs
2. Standardised CBC-MACs
3. EMAC and ARMAC
4. New CBC-MAC schemes
5. RMAC
6. The XCBC family
7. Other schemes
8. 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?
New standards

• NIST has recently produced three new ‘modes’ standards for AES.
  B. CBC-MAC standard (SP800-38B, May 2005).
  C. Combined encryption + integrity mode (SP800-38C, May 2004) – contains CCM.

• NIST activity mirrored in ISO, where:
  A. ISO/IEC 10116 (encryption modes) new version just completed,
  B. ISO/IEC 9797-1 (CBC-MACs) currently being revised, and
  C. ISO/IEC 19772 (Authenticated encryption) being developed.

Candidate schemes

• A number of candidate CBC-MAC schemes were proposed for inclusion in SP800-38B, including:
  – RMAC (Jaulmes, Joux and Valette, 2002),
  – XCBC (Black and Rogaway, 2000), and
Contents of talk

1. CBC-MACs
2. Standardised CBC-MACs
3. EMAC and ARMAC
4. New CBC-MAC schemes
5. RMAC
6. The XCBC family
7. Other schemes
8. 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.

The 2002 draft of SP800-38B

• RMAC was included in the first draft of NIST special publication 800-38B (published in November 2002).
• At that time RMAC was clearly the leading candidate for standardisation.
Reaction to draft SP800-38B

- The release of the 2002 draft of NIST SP 800-38B provoked a large number of negative comments.
- The result was that RMAC was no longer seriously considered for NIST adoption.

A simple observation

- Suppose know one RMAC \((M\text{ say})\) for data \(D\) (using salt \(R\), say).
- Request another MAC \((M'\text{ say})\) for the same data \(D\) (uses salt \(R'\text{ say}\)).
- 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 the 2002 draft of SP 800-38B.
Some attacks on RMAC

- In (Knudsen & Mitchell, 2005) 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. Standardised CBC-MACs
3. EMAC and ARMAC
4. New CBC-MAC schemes
5. RMAC
6. The XCBC family
7. Other schemes
8. 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 + padding 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) 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 standardisation

• In May 2005 NIST published the final version of SP 800-38B.
• This standardises OMAC (which, rather confusingly, NIST calls CMAC).

Partial key recovery attack on TMAC

• Sung, Hong & Lee (2003) described an 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, (and depending on the definition of the term) this is no longer a partial key recovery attack, since $K'$ is not part of the key (but is derived from it) – unlike in 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.
- OMAC (and TMAC) are clearly weaker than EMAC at the ‘proof boundary’, since OMAC (and TMAC) fail catastrophically to trivial forgery attacks.
Modified EMAC I

• We also note that there is a modified version of EMAC which is almost as efficient as OMAC.
• In EMAC the final message block is encrypted twice, once with $K$ and then with a second key $K'$. 
• This can be replaced by a single encryption using $K'$. 

Modified EMAC II

• This is still provably secure.
• It requires the same number of encryption operations as OMAC unless the message is a multiple of the block length (in which case the padding means that one extra encryption is needed).
• Included in the draft revised version of ISO/IEC 9797-1, along with OMAC, EMAC, ARMAC, SMAC (and a modified version of ARMAC called MacDES).
Contents of talk

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

Hash-based MACs

- We have considered only CBC-MACs.
- There are other ways of building MACs.
- With development of credible hash-functions, in 1990s HMAC (a MAC derived from a hash-function) emerged and became popular.
- Note that hashing a concatenation of a key and a message is NOT a good way to generate a MAC – message extension forgeries may be possible.
Novel MAC schemes

- More recently, a new family of MAC functions has emerged with apparently very desirable properties.
- These are based on a family of functions called universal hash-functions.
- A random nonce is needed, which must be different for every message for which a MAC is computed.
- As long as nonces are generated correctly, the schemes are provably secure and also highly efficient.
- Being standardised in ISO/IEC 9797-3.

Contents of talk

1. CBC-MACs
2. Standardised CBC-MACs
3. EMAC and ARMAC
4. New CBC-MAC schemes
5. RMAC
6. The XCBC family
7. Other schemes
8. Conclusions
Where next?

• The main choice right now (for users of CBC-MACs) would appear to be between EMAC and OMAC.
• Both have similar provable security properties.
• OMAC is slightly more efficient.
• However EMAC appears stronger just outside envelope of security proof.
• This may be significant for $n=64$ case, where $2^{n/2}$ is a realisable number of MAC computations.