Network Security and Cryptography
Symmetric-key cryptography

Lecture 10: Authenticated encryption

Mark Ryan

So far, we have seen cryptographic algorithms to achieve

- Confidentiality: use AES encryption with CTR mode, and random nonce;
- Authenticity and integrity: use SHA2-HMAC

Want both privacy and authenticity/integrity. There are two ways to achieve this:

- Combining encryption and MAC in appropriate way; or
- Use a new block mode, which guarantees both confidentiality and authenticity.

Several possibilities for combination:

Need two keys, k_1 an k_2.
You can derive the keys from a master key k: e.g., $k_i = MAC_k(i)$.

- Encrypt-then-MAC: encrypt message, then compute MAC of ciphertext: $E_{k_1}(m),\ MAC_{k_2}(E_{k_1}(m))$
- MAC-then-encrypt: First compute MAC, and then encrypt the message-MAC pair: $E_{k_2}(m,\ MAC_{k_1}(m))$
- Encrypt and MAC: Result is pair of ciphertext and MAC: $E_{k_1}(m),\ MAC_{k_2}(m)$.

Does this provide both privacy and integrity if encryption is IND-CPA secure and MAC cannot be forged?

- Encrypt-then MAC: Yes.
  Used in IPSec.
- MAC-then-encrypt: Not in general, but works in specific instances (e.g., if encryption is CBC or CTR mode with random IV).
  Used in SSL with previous ciphertext as new IV – insecure.
- Encrypt and MAC: Not in general, but works in specific instances.
  Used in SSH.

Suppose that, as well as wanting to authentically encrypt some data m, there is some "associated data" a that we want to authenticate as well, but not encrypt.

For example, the associated data might be the subject line of an email. We encrypt just the body of the message, but we authenticate the whole message (body and subject line).

We have a message m to be encrypted and authenticated, and associated data a to be authenticated only.

Derive two keys from our key k:
$k_1 = MAC_k(1)$, and $k_2 = MAC_k(2)$.

Encrypt-then-mac. The result after encrypt-then-mac is (c, t), where:

$c = E_{k_1}(m)$, and $\qquad t = MAC_{k_2}(a, c)$.

The result is (a, c, t).

Verify-and-decrypt. Given the data (a, c, t), here is how to recover the plaintext message m.
Check if $t = MAC_{k_2}(a, c)$. If not, reject the message. If the check succeeds, let $m = D_{k_1}(c)$.

Definition

We define the *authenticated encryption game* between challenger and attacker as follows:

- The challenger picks an encryption key at random
- The attacker does some computations and may send messages $m_1, \ldots, m_n$ to the challenger
- The challenger responds with the ciphertexts $c_1, \ldots, c_n$.
- The attacker does some more computations and submits a putative ciphertext c to the challenger.
- The challenger outputs 1 if $c \neq c_i$ for all i and $D(k, c) \neq \perp$.

The attacker wins this game if the challenger outputs 1.

Definition

An authenticated encryption scheme (E, D) is secure if the following conditions are satisfied:

- it satisfies IND-CPA
- any attacker wins the authenticated encryption game with only negligible probability

Theorem

If (E, D) is a IND-CPA secure encryption scheme and MAC a secure MAC, the authenticated encryption system obtained by first encrypting and then applying the MAC is a secure authenticated encrypted system.

Galois counter mode

Although Encrypt-then-MAC is secure (provided the Encrypt and the MAC are secure), this combination is not very efficient. It requires two passes through the data (once for encrypt, and once for MAC).

There are modes of operation that encrypt the data and compute a MAC with a *single* pass through the data.

- **Input:** plaintext and key
- **Output:** ciphertext and authentication tag

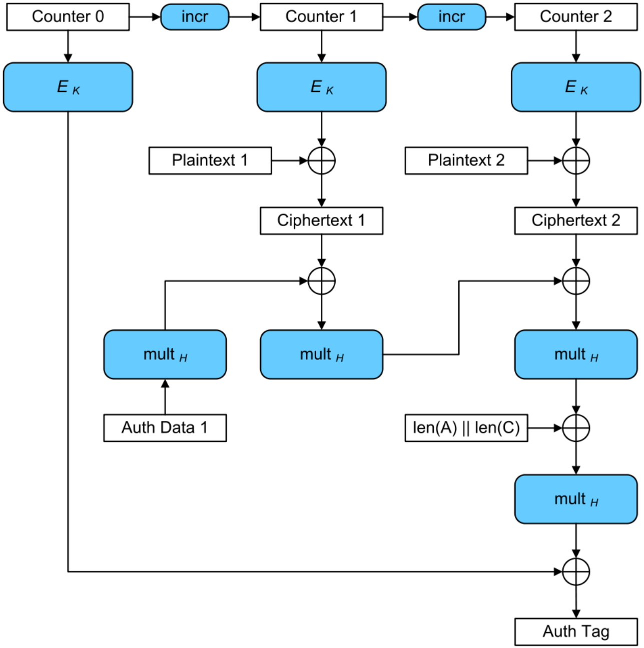
Galois Counter Mode (GCM) is such a mode of operation, and has the advantage of being patent-free.

Galois counter mode (GCM)

GCM works similarly to CTR mode, in that it encrypts a nonce and counter value to produce a key stream which is XOR'd with the plain text. Additionally to CTR mode, it computes an authentication tag on the ciphertext.

GCM works with 128-bit blocks. As well as authenticating the ciphertext, it can authenticate additional data which was not required to be encrypted. This is called "authenticated encryption with associated data (AEAD)".

In the picture on the next slide, the nonce (or IV) that gets included in the encryption and is sent along with the ciphertext is missing :-(


Counter 0
incr
Counter 1
incr
Counter 2
E K
E K
E K
Plaintext 1
Plaintext 2
Ciphertext 1
Ciphertext 2
mult H
mult H
mult H
Auth Data 1
len(A) || len(C)
mult H
Auth Tag

GCM: the authentication tag

To compute the authentication tag, we work within the field GF(2^{128}) whose elements are 128-bit words. The operations $\oplus$ and $\otimes$ are defined on those words: $\oplus$ is bitwise XOR, and $\otimes$ is computed by considering the argument words as polymomials, and multiplying them modulo $x^{128} + x^7 + x^2 + x + 1$.

Let $H = E_K(0^{128})$ be the encryption of 128 zero bits using the encryption key, and let $A_1, \ldots, A_m$ be the blocks of data to be authenticated, and $C_1, \ldots, C_n$ the blocks of ciphertext. The last blocks A_m and C_n are padded with zeros to make 128 bits. Compute:

$$X_i = \begin{cases} 0 & \text{if } i = 0 \\ (X_{i-1} \oplus A_i) \otimes H & \text{if } i = 1, \ldots, m \\ (X_{i-1} \oplus C_{i-m}) \otimes H & \text{if } i = m+1, \ldots, m+n \\ (X_{i-1} \oplus (\text{len}(A) || \text{len}(C))) \otimes H & \text{if } i = m+n+1 \end{cases}$$

The authentication tag is the final of these values, namely X_{m+n+1}, encrypted as the first block (see diagram).

Recommendations

- Don't invent crypto yourself - use standards.
  - For integrity, use use SHA-2 or SHA-3.
  - For authenticity, use HMAC.
  - For confidentiality, use authenticated encryption
    E.g., AES in CTR mode, followed by HMAC
    Or, AES in GCM mode.
- Don't implement crypto yourself - use libraries.