Authentication and Encryption: How to order them?

Debdeep Mukhopadhyay
IIT Kharagpur

Motivation

- Wide spread use of internet requires establishment of a secure channel.
- Typical implementations operate in two stages:
  - first call a key establishment protocol for establishing a secret key,
  - then use this to authenticate and encrypt the transmitted information.
- Common protocols which follow them are SSL, IPSEC, SSH

3 Approaches

- Interestingly 3 different approaches are used by these:
  - SSL: a=Auth(x), C=Enc(x,a), transmit C
    - Authenticate Then Encrypt (AtE)
  - IPSEC: C=Enc(x), a=Auth(C), transmit (C,a)
    - Encrypt Then Authenticate
  - SSH: C=Enc(x), a=Auth(x), transmit(C,a)
    - Encrypt and Authenticate (E&A)

Question

- Q: Are all of them equivalent, or is there any “good” way and “bad” way ?
- Assumptions:
  - We assume that the underlying primitives are strong and developed separately:
    - Enc is secured in the CPA sense’
    - MAC is secured in the chosen message sense.
  - Requirement is for an all-or-nothing approach.
    - ie. we will be satisfied when any combination of a good Enc and a good MAC provides both privacy and integrity.

Encrypt-and-Authenticate

- Independent encryptions and authentication steps:
  - Given a plaintext m, the sender transmits a pair (c,t)

    where,

    $c=Enc_{k1}(m)$, and $t=Mac_{k2}(m)$

  Receiver decrypts c and obtains m. But it outputs, m only when $Vrfy_{k2}(m,t)=1$, otherwise it outputs a NULL.

Authenticate then Encrypt

- Here a MAC tag is first computed, then the message tag and the message are encrypted **together:**
  - **$t=Mac_{k2}(m)$, $c=Enc_{k1}(m||t)$.**
  - **transmit(c)**
- **The receiver decrypts c and then verifies the tag t on m.**
- **It outputs m if $Vrfy_{k2}(m,t)=1$, else NULL.**

Encrypt then Authenticate

- Message is first encrypted and then the tag is computed:
  - $c=Enc_{k1}(m)$, $t=Mac_{k2}(c)$
- Receiver verifies the tag, t and then decrypts c.

Security Definitions

- What is a secure communication channel?

1. The key-generation algorithm Gen' takes input n, runs $\text{Gen}_E(n)$ and $\text{Gen}_M(\text{n})$ to obtain keys k_1 and k_2.
2. The message transmission algorithm EncMac' takes as input the keys (k_1, k_2) and a message m, and outputs a value c, that is derived by some combination of $Enc_{k_1}(.)$ and $Mac_{k_2}(.)$.
3. The decryption algorithm Dec' takes as input the keys (k_1, k_2), and a transmitted value c, and applies some combination of $Dec_{k_1}(.)$ and $Vrfy_{k_2}(.)$. The output of Dec' is either a plaintext m or NULL (if verification fails).

Correctness

$$\mathrm{Dec}'_{k_1,k_2}(EncMac'_{k_1,k_2}(m)) = m$$

Secure Message Transmission Experiment

Secure Message Transmission Experiment $\mathrm{Auth}_{A,\Pi'}(n)$:

1. A random key $k = (k_1, k_2)$ is generated by running Gen'(n).
2. The adversary A is given input n and oracle access to the message transmission algorithm $\mathrm{EncMac}'_k(.)$. The adversary eventually outputs c. Let Q denote the set of all queries that A asked to its oracle.
3. Let $m = Dec'_k(c)$. The output of the experiment is defined to be 1 if and only if 1)m $\neq$ NULL, and 2)m $\notin$ Q

Definition of Security

A message transmission scheme Π' achieves authenticated communication if for all probabilistic polynomial time adversaries A, there exists a negligible function negl, st:

$$\Pr[\text{Auth}_{A,\Pi}(n)=1] \leq negl(n).$$

The definition shows that the adversary's job is slightly easier, since the adversary does not need to know the message m to which its output c corresponds.

Encrypt and Authenticate

$c = Enc_{k_1}(m), t = Mac_{k_2}(m)$

The combination is not necessarily secret. A secure MAC does not necessarily imply privacy.

In particular, if $(\text{Gen}_M, \text{Mac}, \text{Vrfy})$ is a secure MAC, the scheme $(\text{m}, \text{Mac}_k(\text{m}))$ is also secure MAC.

But there is no privacy.

What about CBC-MAC

- **If the MAC is more practical, like CBC-MAC, does the scheme provide secrecy?**

Authenticate-then-encrypt

$t = Mac_{k_2}(m), c = Enc_{k_1}(m \| t)$

The plaintext is often transformed with encodings.
This hides various information to the attacker, like length, side channel information etc.
Let Transform(m) be as follows:

$0 \to 00$

$1 \to 01$ or 10 (arbitrarily)

The inverse transform thus parses the strings as pairs of bits, and then maps 00 to 0, and 01 or 10 as 1.
However, since a 11 can never occur, the result is $\perp$.
Thus, $\text{Transform}^{-1}(0110) = 11$, but $\text{Transformm}^{-1}(1100) = \perp$.

The Encryption function

Define, $\mathrm{Enc}_k(m) = Enc'_k(Transform(m))$, where Enc'_k represents a counter mode encryption.
The discussion holds for any encryption scheme which generates a pesudorandom pad to xor with the message data.
The Enc is a CPA-secure scheme.

Insecurity of the scheme

We show that this scheme is not secure.
The attack works as long as the attacker can check whether a given ciphertext is valid (note an entire decryption is not even needed).
Consider, a challenge ciphertext, $c = Enc'_k(Transform(m \,||\, Mac_{k_2}(m)))$, the attacker simply flips the first two bits of the second block of c.
Note: The first block is the counter value.
Then verifies whether the new ciphertext is valid.
Note, if the first bit of the message is 1, then flipping keeps the ciphertext valid.
However, if the first bit was 0, the flipped bits make the ciphertext invalid.
The attack can be performed on each bit making the scheme leak the entire message.

Authenticate-then-encrypt

- Thus, in general Authenticate-then-encrypt is not secured.
- However certain specific constructs used in SSL are secured.
  - uses CBC mode of encryption
  - OTP for encryption

Encrypt then Authenticate

The message is first encrypted, and then authenticated.

$$c = Enc_{k_1}(m), t = Mac_{k_2}(c).$$

Let Π_E be a CPA-secure private key encryption scheme.
Let Π_M be a secure message authentication code with unique tags. Then the combination (Gen',EncMac',Dec') derived by applying the encrypt-then-authenticate approach to Π_E, Π_M is a secure message transmission scheme.
Proof: CCA-security has been discussed in the last class.
Proof of authentication is left as an exercise.

Secure Message Transmission and CCA-Security

- Secure Message Transmission implies CCA security.
- The opposite is not necessarily true.

Need for independent keys

- Different security goals should always use different keys.
- Thus, if both authentication and privacy are needed we should use different keys.