Cryptography
CS 555

Topic 15: HMAC, Combining Encryption & Authentication
Outline and Readings

• Outline
  • Hash Family
  • NMAC and HMAC
  • CCA-secure encryption
  • Combining encryption & authentication

• Readings:
  • Katz and Lindell: 4.7, 4.8, 4.9
Hash Family (Called Hash Function in the Textbook)

- A hash family $H$ is a function $K \times X \rightarrow Y$
  - $X$ is a set of possible messages
  - $Y$ is a finite set of possible message digests
  - $K$ is the keyspace
  - For each $s \in K$, there is a hash function $h^s \in H$.
  - Here, it is typically assumed that $s$ is made public
    - Unlike when we analyze a PRF

- Hash functions in practice (SHA-1, SHA-2) can be viewed as hash family, where the IV is viewed as the key
Collision Resistant Hash Family

• A Hash family is collision resistant if no adversary has negligible advantage in the following experiment:
  – A key s is generated.
  – Adversary is given s, and needs to find a collision on $h^s$, that is find $x_1, x_2$ such that $h^s(x_1) = h^s(x_2)$
    • A random hash function is chosen, and the adversary needs to produce a collision on that
• Advantage of using the concept of collision resistant hash family instead of a collision resistant hash function
  – Now it makes sense to assume that there is no adversary algorithm can produce collision.
  – Why it does not make sense to say that there exists no algorithm to produce a collision on a fixed hash function?
Constructing MAC from Collision Resistant Hash Functions

- Let $h$ be a collision resistant hash function

- $\text{MAC}_k(M) = h(k \ || \ M)$, where $\ || $ denote concatenation
  - Okay as fixed-length MAC
  - Insecure when variable-length messages are allowed
  - Because of the Merkle-Damgard construction for hash functions, given $M$ and $t=h(K \ || \ M)$, adversary can compute $M'$ by appending to $M$ some new data blocks, and then $h(K||M')$
Idea of NMAC (Nested MAC)

- Given a compression function \( f \), and a hash function \( h \) constructed with \( f \) using the Merkle-Damgard method, NMAC defines \( \text{MAC}_{k_1,k_2}(m) = f(k_1|| h(k_2||m)) \).
  - Technically, both \( f \) and \( h \) are parameterized by a randomly chosen \( s \), however, we ignore it.

- NMAC is secure if both (1) \( h \) produces no collision, and (2) \( f(k||m) \) is a secure fixed-length MAC.
  - \( f(k||m) \) is a secure MAC means that adversary cannot compute \( f(k||m') \) even after obtaining \( f(k||m_1), f(k||m_2), \ldots \)
    - Not implied by \( f \) being collision resistant, but in general safely assumed to be true for practical hash functions.
  - Proof. A forgery against \( f(k_1|| h(k_2||m')) \) means that either \( h(k_2||m') = h(k_2||m_i) \) for a queried \( m_i \), which means \( h \) is not collision resistant; or one computes \( f(k_1||d= h(k_2||m')) \), for a new value \( d \), which means that \( f \) is not a secure MAC.
HMAC: A Derivative of NMAC

\[ HMAC_K[M] = Hash[(K^+ \oplus opad) \ || \ Hash[(K^+ \oplus ipad)\|\|M)]] \]

- \( K^+ \) is the key padded (with 0) to B bytes, the input block size of the hash function
- \( ipad = \) the byte 0x36 repeated B times
- \( opad = \) the byte 0x5C repeated B times.

- Essentially NMAC. Differs in that NMAC uses independent \( k1 \) and \( k2 \), HMAC uses two keys that are computed from one key
- Proven to be PRF if compression function is PRF.
- If used with a secure hash functions (e.g., SHA-256) and according to the specification (key size, and use correct output), no known practical attacks against HMAC exists
HMAC Overview

\[ K^+ \quad \text{ipad} \]

\[ S_i \quad Y_0 \quad Y_1 \quad \ldots \quad Y_{L-1} \]

\[ K^+ \quad \text{opad} \]

\[ S_o \]

IV \( n \) bits \( \rightarrow \) Hash \( n \) bits

\[ H(S_i \parallel M) \]

pad to \( b \) bits

IV \( n \) bits \( \rightarrow \) Hash \( n \) bits

\[ \text{HMAC}_K(M) \]
Constructing CCA-Secure Encryption

• Construction 4.19. CCA-secure encryption scheme.
  – Uses a CPA-secure encryption scheme, and a secure MAC.
  – In key generation, generates $k_1$ for encryption, and $k_2$ for MAC.
  – To encrypt a message $m$, computes ciphertext $\langle c=\text{Enc}_{k_1}(m), \ t=\text{MAC}_{k_2}(c_1) \rangle$
    • The ciphertext of the scheme is a pair $(c,t)$
  – To decrypt a ciphertext $\langle c, t \rangle$, first check whether $\text{Vrfy}_{k_2}(c, t)=1$; if yes, outputs $\text{Dec}_{k_1}(c)$; if not, outputs $\bot$
    • That is, decline to decrypt if the MAC does not verify

• This is CCA-secure because the adversary gets nothing from the decryption oracle, unless the adversary can break the MAC first
Encryption and Authentication

• Three ways for encryption and authentication
  – Authenticate-then-encrypt (AtE), used in SSL
    • $a = \text{MAC}(x)$, $C = E(x,a)$, transmit $C$
  – Encrypt-then-authenticate (EtA), used in IPSec
    • $C = E(x)$, $a = \text{MAC}(C)$, transmit $(C,a)$
  – Encrypt-and-authenticate (E&A), used in SSH
    • $C = E(x)$, $a = \text{MAC}(x)$, transmit $(C,a)$

• Which way provides secure communications when embedded in a protocol that runs in a real adversarial network setting?
Encryption Alone May Be Insufficient for Privacy

- If an adversary can manipulate a ciphertext such that the observable behavior (such as success or failure of decryption) differs depending on the content of plaintext, then information about plaintext can be leaked.
- To defend against these, should authenticate ciphertext, and only decrypt after making sure ciphertext has not changed.
- Encrypt-then-authenticate (EtA) is secure
  - $C = E(x)$, $a = MAC(C)$, transmit $(C, a)$
Encryption Alone May Be Insufficient for Privacy: An Artificial Example

- Given a secure stream cipher (or even one-time pad) $E$, consider encryption $E^*$
  - $E^*[x] = E[\text{encode}[x]]$
    - $\text{encode}[x]$ replaces 0 with 00, and 1 with either 01 or 10.
  - How to decrypt?
  - $E^*[x]$ is secure

- Using $E^*$ may not provide confidentiality in some usage
  - Consider the case an adversary flips the first two bits of $E^*[x]$
  - When the bits are 01 or 10, flipping results in no change after decrypt
  - When the bits are 00, flipping result in decryption failure
  - Learning whether decryption succeeds reveal first bit
AtE and E&A are insecure

- Authenticate-then-encrypt (AtE) is not always secure
  - $a = \text{MAC}(x)$, $C=E(x,a)$, transmit $C$
  - As first step is decryption, its success or failure may leak information.
  - AtE, however, can be secure for some encryption schemes, such as CBC or OTP (or stream ciphers)

- Encrypt-and-authenticate (E&A) is not secure
  - $C=E(x)$, $a=\text{MAC}(x)$, transmit $(C,a)$
  - MAC has no guarantee for confidentiality
Coming Attractions …

- Private key management and the Public key revolution

- Reading: Katz & Lindell: Chapter 9