CSE 207B: Applied Cryptography

Nadia Heninger

UCSD

Fall 2025 Lecture 7.5

Announcements

1. HW 3 is due today!
2. HW 4 is out, due before class in 1 week

Last time: Hash function cryptanalysis and HMAC

This time: Authenticated encryption

Key derivation

Problem: How do we get symmetric keys?

Input: Some data that we want to use to generate a key.

- A password
- A bunch of nonuniform random inputs from the environment
- The result of a public-key agreement (coming soon!)

Desired output: Uniform AES or MAC keys of the right length.

Solutions that work in practice:

- $H(data)$
- $HMAC_0(data)$ (better for Merkle-Damgård functions)

Subkey derivation

For a real protocol, we likely need several keys: encryption keys for each direction, MAC keys.

Once we have derived a master key mk using a hash function, we can use a PRF to to derive subkeys.

Examples:

- $k_{mac} = F_{mk}(\text{"MAC-KEY"})$
- $k_{AB} = F_{mk}(\text{"AB-KEY"})$ for Alice $\rightarrow$ Bob encryption
- $k_{BA} = F_{mk}(\text{"BA-KEY"})$ for Bob $\rightarrow$ Alice encryption

If F is a secure PRF, then these behave like independent keys.

HMAC is often used for this in practice.

HKDF

Standardized HMAC-based key derivation function.

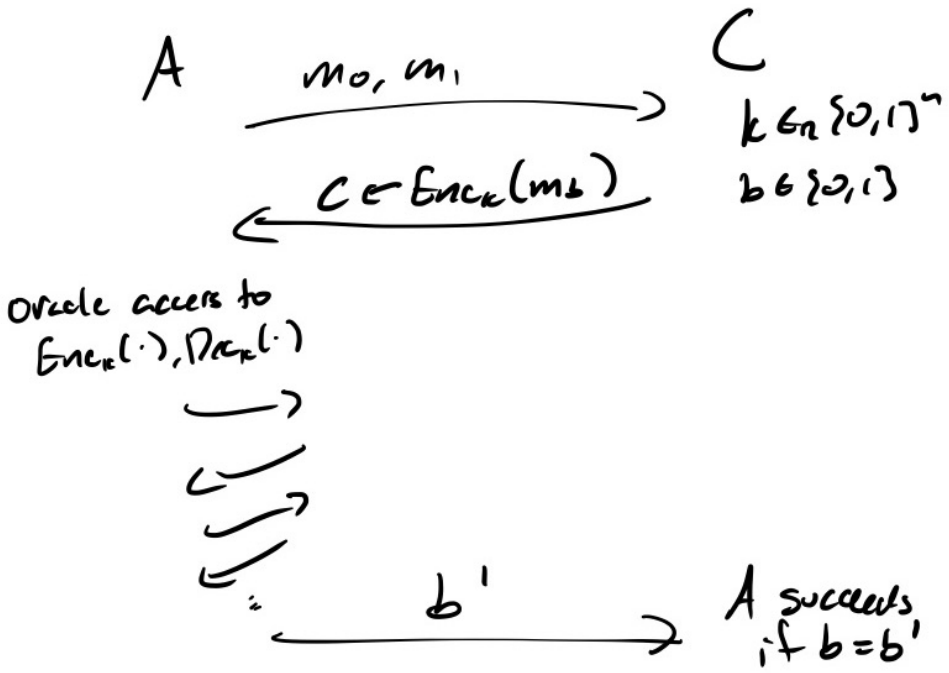
Input: Secret s, optional salt *salt*

Output: L bytes of output

Algorithm:
Use a HMAC function with output length ℓ.

1. $t = HMAC_{salt}(s)$
2. $z_0 =$ empty string.
3. for i from 1 to $\lceil L/\ell \rceil$:
   $z_i = HMAC_t(z_{i-1}||i)$
4. Output L bytes of $z_1||\ldots$

Chosen ciphertext attacks



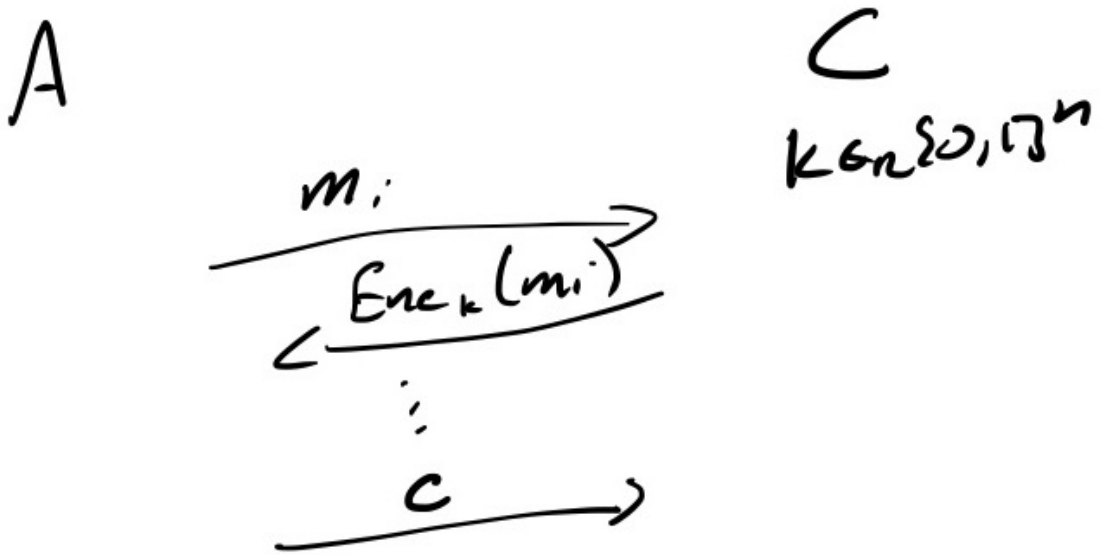
Definition

(Enc, Dec) is CCA-secure if $\forall$ efficient adversaries A,

$$\Pr[A \text{ succeeds}] \leq 1/2 + \epsilon$$

IND-CCA1: Non-adaptive: Decryption oracle only queried prior to c
IND-CCA2: Adaptive: May make further calls to decryption oracle

Ciphertext Integrity



A wins if c is a valid ciphertext and not queried.

Definition

$(\mathsf{Enc}, \mathsf{Dec})$ provides ciphertext integrity if $\Pr[A \text{ succeeds}] =$ negligible.

Authenticated Encryption

Definition

$(\mathsf{Enc}, \mathsf{Dec})$ provides authenticated encryption if it is CPA-secure and provides ciphertext integrity.

Theorem

If $(\mathsf{Enc}, \mathsf{Dec})$ *provides authenticated encryption then it is CCA-secure.*

Constructing Authenticated Encryption

Encrypt-then-MAC

- Encryption: $c = \mathsf{Enc}_{k_e}(m) \quad t = \mathsf{Mac}_{k_m}(c) \quad$ output (c, t)
- Decryption: Input (c, t).
  If $\mathsf{Verify}_{k_m}(c, t) = \texttt{reject}$ then output $\texttt{reject}$
  else output $\mathsf{Dec}_{k_e}(c)$.

Theorem

Encrypt-then-MAC is CCA secure.

Common implementation mistakes:

- Using the same key for encryption and MAC
- Only MACing part of the ciphertext. (e.g. omitting the IV or the data used to derive a deterministic IV)
- Outputting some plaintext before verifying integrity

MAC then Encrypt is not CCA secure

MAC-then-encrypt

- Encryption: $t = \mathrm{Mac}_{k_m}(m) \quad c = \mathrm{Enc}_{k_e}(m||t) \quad$ output c
- Decryption: Input c. Compute $\mathrm{Dec}_{k_e}(c) = (m||t)$
  If $\mathrm{Verify}_{k_m}(m, t) = \texttt{reject}$ then output $\texttt{reject}$
  else output m.

MAC-then-encrypt can fail to be secure even with CPA-secure Enc and secure MAC.

SSL 3.0 vulnerable to POODLE attack.

POODLE Attack Setup

Victim is a web browser.

Victim visits `evil.com`.

`evil.com` contains Javascript causing victim to make cookie-bearing request to `bank.com`.

Man-in-the-middle attacker intercepts encrypted traffic between victim and `bank.com` and modifies ciphertext, using `bank.com` as a decryption oracle.

POODLE Attack Idea

SSL 3.0 uses MAC-then-encrypt with CBC mode.

```
c = Enc(message || MAC tag || pad)
```

To pad p bytes, append $p-1$ arbitrary bytes and then byte $p-1$.
(For 0 bytes, append dummy block of 15 bytes ending in 15.)

If adversary intercepts block

$$c = \underbrace{\boxed{c[0]}}_{\text{IV}}\underbrace{\boxed{c[1]}\ \cdots}_{\text{encryption of m}}\underbrace{\boxed{c[\ell-1]}}_{\text{encrypted tag}}\underbrace{\boxed{c[\ell]}}_{\text{encrypted pad}}$$

Then they query decryption oracle with

$$\hat{c} := \boxed{c[0]}\boxed{c[1]}\ \cdots\ \boxed{c[\ell-1]}\underbrace{\boxed{c[1]}}_{\text{encrypted pad?}}$$

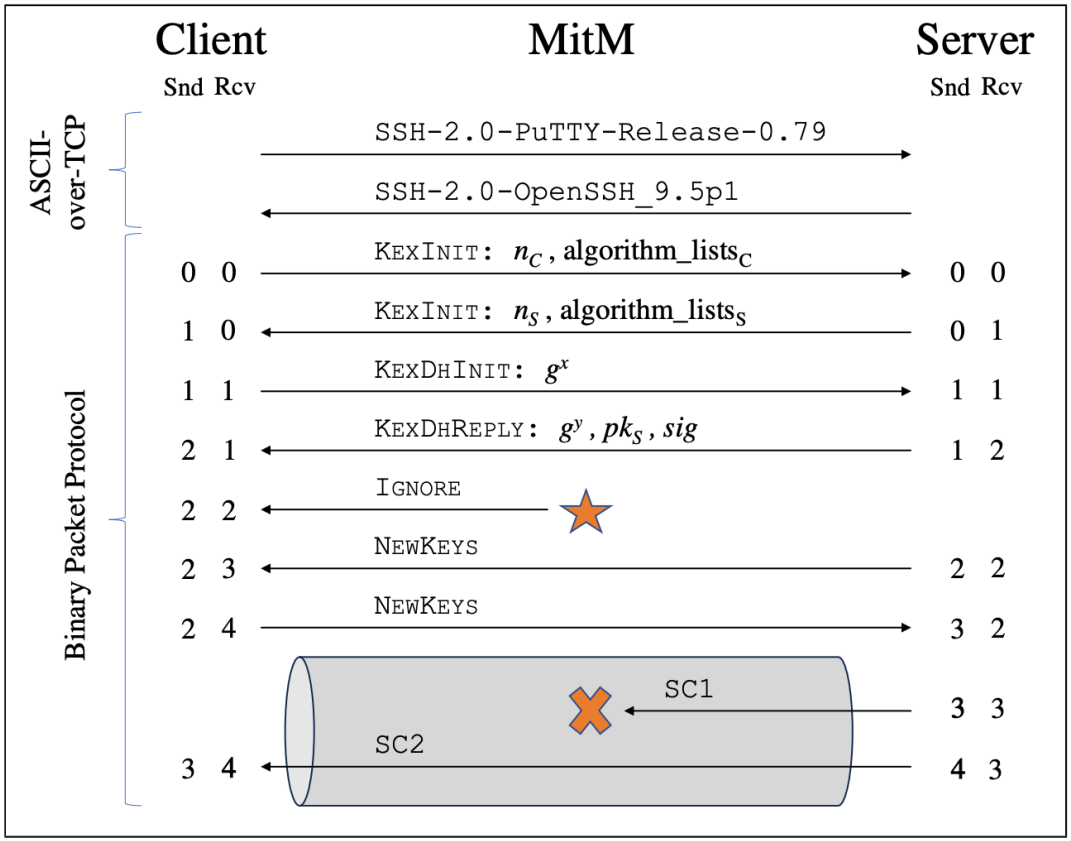
If last byte is 15, decryption valid, otherwise likely reject
$\implies$ learn byte of m. (Same logic as your homework.)

Terrapin Attack: Attacking SSH integrity

[B aumer, Brinkmann, Schwenk 2024]

Observation: A per-packet MAC cannot prevent a MitM attacker from deleting packets.

SSH is supposed to prevent against this with sequence numbers, but the authors identify ways to manipulate sequence numbers.



Authenticated encryption in practice

Fine solution: Use AES-GCM mode.

- TLS 1.3 uses authenticated encryption modes correctly.
- Older versions of TLS use MAC-then-encrypt.
- SSHv2 uses Encrypt-and-MAC, and also offers AEAD ciphers AES-GCM and ChaCha20-Poly1305.

1. HW 3 due today!

2. HW 4 due in 1 week!