PRF, Block Ciphers
MAC
Lecture 6
One-time CPA-secure SKE with a Stream-Cipher

- One-time Encryption with a stream-cipher:
  - Generate a one-time pad from a short seed
  - Can share just the seed as the key
  - Mask message with the pseudorandom pad
- Decryption is symmetric: plaintext & ciphertext interchanged
- SC can spit out bits on demand, so the message can arrive bit by bit, and the length of the message doesn't have to be a priori fixed
- Security: indistinguishability from using a truly random pad
Beyond One-Time Encryption
Beyond One-Time Encryption

Need to make sure the same part of the one-time pad is never reused
Beyond One-Time Encryption

- Need to make sure the same part of the one-time pad is never reused

- Sender and receiver will need to maintain state and stay in sync (indicating how much of the pad has already been used)
Beyond One-Time Encryption

Need to make sure the same part of the one-time pad is never reused

Sender and receiver will need to maintain state and stay in sync (indicating how much of the pad has already been used)

Or only sender maintains the index, but sends it to the receiver. Then receiver will need to run the stream-cipher to get to that index.
Beyond One-Time Encryption

- Need to make sure the same part of the one-time pad is never reused
- Sender and receiver will need to maintain state and stay in sync (indicating how much of the pad has already been used)
- Or only sender maintains the index, but sends it to the receiver. Then receiver will need to run the stream-cipher to get to that index.
- A PRG with direct access to any part of the output stream?
Beyond One-Time Encryption

- Need to make sure the same part of the one-time pad is never reused

- Sender and receiver will need to maintain state and stay in sync (indicating how much of the pad has already been used)

- Or only sender maintains the index, but sends it to the receiver. Then receiver will need to run the stream-cipher to get to that index.

- A PRG with direct access to any part of the output stream?

Pseudo Random Function (PRF)
Pseudorandom Function (PRF)
Pseudorandom Function (PRF)

A compact representation of an exponentially long (pseudorandom) string
Pseudorandom Function (PRF)

- A compact representation of an exponentially long (pseudorandom) string
- Allows “random-access” (instead of just sequential access)
Pseudorandom Function (PRF)

- A compact representation of an exponentially long (pseudorandom) string
- Allows “random-access” (instead of just sequential access)
- A function $F(s; i)$ outputs the $i^{th}$ block of the pseudorandom string corresponding to seed $s$
Pseudorandom Function (PRF)

- A compact representation of an exponentially long (pseudorandom) string
- Allows “random-access” (instead of just sequential access)
- A function $F(s; i)$ outputs the $i^{th}$ block of the pseudorandom string corresponding to seed $s$
- Exponentially many blocks (i.e., large domain for $i$)
Pseudorandom Function (PRF)

- A compact representation of an exponentially long (pseudorandom) string
- Allows “random-access” (instead of just sequential access)
- A function $F(s; i)$ outputs the $i^{th}$ block of the pseudorandom string corresponding to seed $s$
- Exponentially many blocks (i.e., large domain for $i$)
- Pseudorandom Function
Pseudorandom Function (PRF)

A compact representation of an exponentially long (pseudorandom) string

- Allows “random-access” (instead of just sequential access)

- A function $F(s;i)$ outputs the $i^{th}$ block of the pseudorandom string corresponding to seed $s$

- Exponentially many blocks (i.e., large domain for $i$)

Pseudorandom Function

- Need to define pseudorandomness for a function (not a string)
Pseudorandom Function
(PRF)
Pseudorandom Function (PRF)

\[ F: \{0,1\}^k \times \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)} \] is a PRF if all PPT adversaries have negligible advantage in the PRF experiment.
Pseudorandom Function (PRF)

$F: \{0,1\}^k \times \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)}$ is a PRF if all PPT adversaries have negligible advantage in the PRF experiment.

Adversary given oracle access to either $F$ with a random seed, or a random function $R: \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)}$. Needs to guess which.
Pseudorandom Function (PRF)

F: \{0,1\}^k \times \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)} is a PRF if all PPT adversaries have negligible advantage in the PRF experiment

Adversary given oracle access to either F with a random seed, or a random function R: \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)}. Needs to guess which.
Pseudorandom Function (PRF)

F: \{0,1\}^k \times \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)} is a PRF if all PPT adversaries have negligible advantage in the PRF experiment.

Adversary given oracle access to either F with a random seed, or a random function R: \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)}. Needs to guess which.
Pseudorandom Function (PRF)

F: \{0,1\}^k \times \{0,1\}^{m(k)} \to \{0,1\}^{n(k)} is a PRF if all PPT adversaries have negligible advantage in the PRF experiment.

Adversary given oracle access to either F with a random seed, or a random function R: \{0,1\}^{m(k)} \to \{0,1\}^{n(k)}. Needs to guess which.

Note: Only 2^k seeds for F.

b \leftarrow \{0,1\}
b' = b? \rightarrow \text{Yes/No}
Pseudorandom Function (PRF)

F: \{0,1\}^k \times \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)}\text{ is a PRF if all PPT adversaries have negligible advantage in the PRF experiment}

Adversary given oracle access to either F with a random seed, or a random function R: \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)}. Needs to guess which.

Note: Only $2^k$ seeds for F

But $2^{(n2^m)}$ functions R
Pseudorandom Function (PRF)

F: \{0,1\}^k \times \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)} is a PRF if all PPT adversaries have negligible advantage in the PRF experiment.

Adversary given oracle access to either F with a random seed, or a random function R: \{0,1\}^{m(k)} \rightarrow \{0,1\}^{n(k)}. Needs to guess which.

Note: Only 2^k seeds for F

But 2^{n2^m} functions R

PRF stretches k bits to n2^m bits
Pseudorandom Function (PRF)
Pseudorandom Function (PRF)

A PRF can be constructed from any PRG
A PRF can be constructed from any PRG.
A PRF can be constructed from any PRG.
A PRF can be constructed from any PRG
A PRF can be constructed from any PRG.

G is a length-doubling PRG.
A PRF can be constructed from any PRG

G is a length-doubling PRG
A PRF can be constructed from any PRG.
A PRF can be constructed from any PRG

$G$ is a length-doubling PRG
A PRF can be constructed from any PRG
A PRF can be constructed from any PRG

$G$ is a length-doubling PRG
A PRF can be constructed from any PRG.
Pseudorandom Function (PRF)

A PRF can be constructed from any PRG
A PRF can be constructed from any PRG
Not blazing fast
Pseudorandom Function (PRF)

A PRF can be constructed from any PRG
- Not blazing fast

Faster constructions based on specific number-theoretic computational complexity assumptions
A PRF can be constructed from any PRG
- Not blazing fast
- Faster constructions based on specific number-theoretic computational complexity assumptions
- Fast heuristic constructions
Pseudorandom Function (PRF)

- A PRF can be constructed from any PRG
  - Not blazing fast
- Faster constructions based on specific number-theoretic computational complexity assumptions
- Fast heuristic constructions
- PRF in practice: Block Cipher
Pseudorandom Function (PRF)

- A PRF can be constructed from any PRG
- Not blazing fast
- Faster constructions based on specific number-theoretic computational complexity assumptions
- Fast heuristic constructions

PRF in practice: Block Cipher
Pseudorandom Function (PRF)

- A PRF can be constructed from any PRG
  - Not blazing fast
- Faster constructions based on specific number-theoretic computational complexity assumptions
- Fast heuristic constructions

PRF in practice: Block Cipher
- Extra features/requirements:
Pseudorandom Function (PRF)

A PRF can be constructed from any PRG
- Not blazing fast

Faster constructions based on specific number-theoretic computational complexity assumptions

Fast heuristic constructions

PRF in practice: Block Cipher
- Extra features/requirements:
  - Permutation: input block (r) to output block
Pseudorandom Function (PRF)

- A PRF can be constructed from any PRG
  - Not blazing fast
- Faster constructions based on specific number-theoretic computational complexity assumptions
- Fast heuristic constructions

PRF in practice: Block Cipher
- Extra features/requirements:
  - Permutation: input block (r) to output block
  - Key can be used as an inversion trapdoor
Pseudorandom Function (PRF)

- A PRF can be constructed from any PRG
  - Not blazing fast
- Faster constructions based on specific number-theoretic computational complexity assumptions
- Fast heuristic constructions

PRF in practice: Block Cipher
- Extra features/requirements:
  - Permutation: input block (r) to output block
  - Key can be used as an inversion trapdoor
  - Pseudorandomness even with access to inversion
CPA-secure SKE with a Block Cipher
Suppose Alice and Bob have shared a key (seed) for a block-cipher (PRF) BC.
CPA-secure SKE with a Block Cipher

Suppose Alice and Bob have shared a key (seed) for a block-cipher (PRF) $BC$

For each encryption, Alice will pick a fresh pseudorandom pad, by picking a fresh value $r$ and setting $pad = BC_K(r)$
CPA-secure SKE with a Block Cipher

Suppose Alice and Bob have shared a key (seed) for a block-cipher (PRF) $BC$

For each encryption, Alice will pick a fresh pseudorandom pad, by picking a fresh value $r$ and setting $pad = BC_K(r)$
CPA-secure SKE with a Block Cipher

Suppose Alice and Bob have shared a key (seed) for a block-cipher (PRF) BC.

For each encryption, Alice will pick a fresh pseudorandom pad, by picking a fresh value $r$ and setting $\text{pad} = BC_k(r)$.

Bob needs to be able to generate the same pad, so Alice sends $r$ (in the clear, as part of the ciphertext) to Bob.
CPA-secure SKE with a Block Cipher

Suppose Alice and Bob have shared a key (seed) for a block-cipher (PRF) $BC$

For each encryption, Alice will pick a fresh pseudorandom pad, by picking a fresh value $r$ and setting $pad = BC_K(r)$

Bob needs to be able to generate the same pad, so Alice sends $r$ (in the clear, as part of the ciphertext) to Bob
Suppose Alice and Bob have shared a key (seed) for a block-cipher (PRF) $BC$

For each encryption, Alice will pick a fresh pseudorandom pad, by picking a fresh value $r$ and setting $\text{pad} = BC_K(r)$

Bob needs to be able to generate the same pad, so Alice sends $r$ (in the clear, as part of the ciphertext) to Bob
Suppose Alice and Bob have shared a key (seed) for a block-cipher (PRF) BC.

For each encryption, Alice will pick a fresh pseudorandom pad, by picking a fresh value \( r \) and setting \( \text{pad} = BC_K(r) \).

Bob needs to be able to generate the same pad, so Alice sends \( r \) (in the clear, as part of the ciphertext) to Bob.

Even if Eve sees \( r \), PRF security guarantees that \( BC_K(r) \) is pseudorandom. (In fact, Eve could have picked \( r \), as long as we ensure no \( r \) is reused.)
CPA-secure SKE with a Block Cipher

Suppose Alice and Bob have shared a key (seed) for a block-cipher (PRF) BC.

For each encryption, Alice will pick a fresh pseudorandom pad, by picking a fresh value $r$ and setting $pad = BC_K(r)$.

Bob needs to be able to generate the same pad, so Alice sends $r$ (in the clear, as part of the ciphertext) to Bob.

Even if Eve sees $r$, PRF security guarantees that $BC_K(r)$ is pseudorandom. (In fact, Eve could have picked $r$, as long as we ensure no $r$ is reused.)

How to pick a fresh $r$?
Suppose Alice and Bob have shared a key (seed) for a block-cipher (PRF) BC.

For each encryption, Alice will pick a fresh pseudorandom pad, by picking a fresh value \( r \) and setting \( \text{pad} = \text{BC}_{K}(r) \).

Bob needs to be able to generate the same pad, so Alice sends \( r \) (in the clear, as part of the ciphertext) to Bob.

Even if Eve sees \( r \), PRF security guarantees that \( \text{BC}_{K}(r) \) is pseudorandom. (In fact, Eve could have picked \( r \), as long as we ensure no \( r \) is reused.)

How to pick a fresh \( r \)?

- Pick at random!
CPA-secure SKE with a Block Cipher
CPA-secure SKE with a Block Cipher

How to encrypt a long message (multiple blocks)?
How to encrypt a long message (multiple blocks)?

Chop the message into blocks and independently encrypt each block as before?
CPA-secure SKE with a Block Cipher

How to encrypt a long message (multiple blocks)?

Chop the message into blocks and independently encrypt each block as before?

Works, but ciphertext size is double that of the plaintext (if |m| is one-block long)
How to encrypt a long message (multiple blocks)?

Chop the message into blocks and independently encrypt each block as before?

Works, but ciphertext size is double that of the plaintext (if $|r|$ is one-block long)

Extend output length of PRF (w/o increasing input length)
How to encrypt a long message (multiple blocks)?

Chop the message into blocks and independently encrypt each block as before?

Works, but ciphertext size is double that of the plaintext (if $|r|$ is one-block long)

Extend output length of PRF (w/o increasing input length)
CPA-secure SKE

with a Block Cipher

How to encrypt a long message (multiple blocks)?

- Chop the message into blocks and independently encrypt each block as before?
- Works, but ciphertext size is double that of the plaintext (if |r| is one-block long)
- Extend output length of PRF (w/o increasing input length)
CPA-secure SKE
with a Block Cipher

How to encrypt a long message (multiple blocks)?

- Chop the message into blocks and independently encrypt each block as before?
  - Works, but ciphertext size is double that of the plaintext (if $|r|$ is one-block long)

- Extend output length of PRF (w/o increasing input length)
How to encrypt a long message (multiple blocks)?

Chop the message into blocks and independently encrypt each block as before?

Works, but ciphertext size is double that of the plaintext (if |r| is one-block long)

Extend output length of PRF (w/o increasing input length)

input length slightly decreased, based on an a priori limit on t
CPA-secure SKE with a Block Cipher

How to encrypt a long message (multiple blocks)?

- Chop the message into blocks and independently encrypt each block as before?
- Works, but ciphertext size is double that of the plaintext (if |r| is one-block long)

Extend output length of PRF (w/o increasing input length)

- Output is indistinguishable from t random blocks (even if input to \( F_K \) known/chosen)
CPA-secure SKE with a Block Cipher
CPA-secure SKE with a Block Cipher

Various “modes” of operation of a Block-cipher (i.e., encryption schemes using a block-cipher). All with one block overhead.
CPA-secure SKE with a Block Cipher

Various “modes” of operation of a Block-cipher (i.e., encryption schemes using a block-cipher). All with one block overhead.

**Output Feedback (OFB) mode:** Extend the pseudorandom output using the first construction in the previous slide.
CPA-secure SKE with a Block Cipher

Various “modes” of operation of a Block-cipher (i.e., encryption schemes using a block-cipher). All with one block overhead.

Output Feedback (OFB) mode: Extend the pseudorandom output using the first construction in the previous slide.
CPA-secure SKE with a Block Cipher

Various “modes” of operation of a Block-cipher (i.e., encryption schemes using a block-cipher). All with one block overhead.

Output Feedback (OFB) mode: Extend the pseudorandom output using the first construction in the previous slide.
Various “modes” of operation of a Block-cipher (i.e., encryption schemes using a block-cipher). All with one block overhead.

**Output Feedback (OFB) mode:** Extend the pseudorandom output using the first construction in the previous slide

**Counter (CTR) Mode:** Similar idea as in the second construction. No a priori limit on number of blocks in a message. Security from low likelihood of \((r+1,...,r+t)\) running into \((r'+1,...,r'+t')\)
Various “modes” of operation of a Block-cipher (i.e., encryption schemes using a block-cipher). All with one block overhead.

- **Output Feedback (OFB) mode:** Extend the pseudorandom output using the first construction in the previous slide.

- **Counter (CTR) Mode:** Similar idea as in the second construction. No a priori limit on number of blocks in a message. Security from low likelihood of \((r+1,...,r+t)\) running into \((r'+1,...,r'+t')\)
CPA-secure SKE
with a Block Cipher

Various “modes” of operation of a Block-cipher (i.e., encryption schemes using a block-cipher). All with one block overhead.

Output Feedback (OFB) mode: Extend the pseudorandom output using the first construction in the previous slide.

Counter (CTR) Mode: Similar idea as in the second construction. No a priori limit on number of blocks in a message. Security from low likelihood of \((r+1,...,r+t)\) running into \((r'+1,...,r'+t')\)

Cipher Block Chaining (CBC) mode: Sequential encryption. Decryption uses \(F_K^{-1}\). Ciphertext an integral number of blocks.

Not a PRF (Why?)
Active Adversary
Active Adversary
Active Adversary

An active adversary can inject messages into the channel
Active Adversary

- An active adversary can inject messages into the channel.
- Eve can send ciphertexts to Bob and get them decrypted.
Active Adversary

An active adversary can inject messages into the channel.

Eve can send ciphertexts to Bob and get them decrypted.

Chosen Ciphertext Attack (CCA)
Active Adversary

- An active adversary can inject messages into the channel.
- Eve can send ciphertexts to Bob and get them decrypted.
  - Chosen Ciphertext Attack (CCA)
- If Bob decrypts all ciphertexts for Eve, no security possible.
Active Adversary

An active adversary can inject messages into the channel.

- Eve can send ciphertexts to Bob and get them decrypted.
- Chosen Ciphertext Attack (CCA)

If Bob decrypts all ciphertexts for Eve, no security possible.

- What can Bob do?
Symmetric-Key Encryption

SIM-CCA Security

\[ \forall \exists \text{s.t.} \]

\[ \text{REAL} \approx \text{IDEAL} \]
Symmetric-Key Encryption

SIM-CCA Security

Invalid ciphertexts are silently ignored

REAL ≈ IDEAL
Symmetric-Key Encryption

IND-CCA Security

Experiment picks $b \leftarrow \{0,1\}$ and $K \leftarrow \text{KeyGen}$

Adv gets (guarded) access to $\text{Dec}_K$ oracle

For as long as Adversary wants

Adv sends two messages $m_0, m_1$ to the experiment

Expt returns $\text{Enc}(m_b, K)$ to the adversary

Adversary returns a guess $b'$

Experiments outputs 1 iff $b' = b$

IND-CCA secure if for all feasible adversaries $\Pr[b' = b] \approx 1/2$
CCA Security
CCA Security

How to obtain CCA security?
CCA Security

How to obtain CCA security?

Use a CPA-secure encryption scheme, but make sure Bob “accepts” and decrypts only ciphertexts produced by Alice.
CCA Security

How to obtain CCA security?

Use a CPA-secure encryption scheme, but make sure Bob “accepts” and decrypts only ciphertexts produced by Alice

i.e., Eve can’t create new ciphertexts that will be accepted by Bob
CCA Security

How to obtain CCA security?

- Use a CPA-secure encryption scheme, but make sure Bob “accepts” and decrypts only ciphertexts produced by Alice.

  i.e., Eve can’t create new ciphertexts that will be accepted by Bob.

- Achieves the stronger guarantee: in IDEAL, Eve can’t send its own messages to Bob.
CCA Security

How to obtain CCA security?

Use a CPA-secure encryption scheme, but make sure Bob “accepts” and decrypts only ciphertexts produced by Alice i.e., Eve can’t create new ciphertexts that will be accepted by Bob

Achieves the stronger guarantee: in IDEAL, Eve can’t send its own messages to Bob

CCA secure SKE reduces to the problem of CPA secure SKE and (shared key) message authentication
CCA Security

How to obtain CCA security?

Use a CPA-secure encryption scheme, but make sure Bob “accepts” and decrypts only ciphertexts produced by Alice.

i.e., Eve can’t create new ciphertexts that will be accepted by Bob.

Achieves the stronger guarantee: in IDEAL, Eve can’t send its own messages to Bob.

CCA secure SKE reduces to the problem of CPA secure SKE and (shared key) message authentication.

MAC: Message Authentication Code
Message Authentication Codes
Message Authentication Codes

A single short key shared by Alice and Bob
Message Authentication Codes

- A single short key shared by Alice and Bob
- Can sign any (polynomial) number of messages
Message Authentication Codes

- A single short key shared by Alice and Bob
- Can sign any (polynomial) number of messages
- A triple (KeyGen, MAC, Verify)
Message Authentication Codes

- A single short key shared by Alice and Bob
- Can sign any (polynomial) number of messages
- A triple (KeyGen, MAC, Verify)
- Correctness: For all $K$ from KeyGen, and all messages $M$, $\text{Verify}_K(M, \text{MAC}_K(M)) = 1$
Message Authentication Codes

- A single short key shared by Alice and Bob
- Can sign any (polynomial) number of messages
- A triple (KeyGen, MAC, Verify)
- Correctness: For all $K$ from KeyGen, and all messages $M$, $\text{Verify}_K(M, \text{MAC}_K(M)) = 1$
- Security: probability that an adversary can produce $(M, s)$ s.t. $\text{Verify}_K(M, s) = 1$ is negligible unless Alice produced an output $s = \text{MAC}_K(M)$

\[
\text{Advantage} = \Pr[ \text{Ver}_K(M, s) = 1 \text{ and } (M, s) \notin \{(M_i, s_i)\} ]
\]
CCA Secure SKE
CCA Secure SKE

\[ CCA-Enc_{K_1,K_2}(m) = ( c:= CPA-Enc_{K_1}(m), \ t:= MAC_{K_2}(c) ) \]
CCA Secure SKE

\[ \text{CCA-Enc}_{K_1,K_2}(m) = ( c := \text{CPA-Enc}_{K_1}(m), t := \text{MAC}_{K_2}(c) ) \]

CPA secure encryption: Block-cipher/CTR mode construction
CCA Secure SKE

\[ \text{CCA-Enc}_{K_1,K_2}(m) = (c := \text{CPA-Enc}_{K_1}(m), t := \text{MAC}_{K_2}(c)) \]

- CPA secure encryption: Block-cipher/CTR mode construction
- MAC: from a PRF or Block-Cipher (next time)
CCA Secure SKE

$$\text{CCA-Enc}_{K_1,K_2}(m) = (c := \text{CPA-Enc}_{K_1}(m), t := \text{MAC}_{K_2}(c))$$

- CPA secure encryption: Block-cipher/CTR mode construction
- MAC: from a PRF or Block-Cipher (next time)

**SKE in practice entirely based on Block-Ciphers** (next time)
CCA Secure SKE

\[ \text{CCA-Enc}_{K_1,K_2}(m) = ( c:= \text{CPA-Enc}_{K_1}(m), \ t:= \text{MAC}_{K_2}(c) ) \]

- CPA secure encryption: Block-cipher/CTR mode construction
- MAC: from a PRF or Block-Cipher (next time)

**SKE in practice entirely based on Block-Ciphers** (next time)

In principle, PRFs can be constructed (less efficiently) based on any One-Way Permutation or even any One-Way Function.