Symmetric-Key Encryption: constructions

Lecture 5
PRF, Block Cipher
**PRG**

- **G is a PRG if**\( \{G_k(x)\}_{x \leftarrow \{0,1\}^k} \approx U_{n(k)} \) and \( G \) PPT
- **A PRG can be used to obtain a one-time CPA-secure SKE**
  - Stream cipher: PRG without an a priori bound \( n(k) \) on the output length
  - Security: The pad produced by the PRG is indistinguishable from a truly random pad
  - Hence the scheme is indistinguishable from the one-time pad scheme (which is one-time CPA secure)
- **Question: Multiple-message SKE?**
Beyond One-Time

- Need to make sure that 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)

- 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)
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
A PRF can be constructed from any PRG
Pseudorandom Function (PRF)

- A PRF can be constructed from any PRG
- Not blazing fast: needs $|r|$ evaluations of a PRG
- 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
Suppose Alice and Bob have shared a key (seed) for a block-cipher (or PRF) BC.

For each encryption, Alice will pick a fresh pseudorandom pad, by picking a new 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 new $r$? Pick at random!
Weak PRF

Note: CPA-Security relied on the inputs to the PRF being just distinct (not random)

But if the input is indeed random, a weaker guarantee on PRF suffices

Weak PRF: Similar to PRF, but the inputs to the oracle are chosen randomly

As before, adversary can see both the input and the output

As before, adversary can see as many input-output pairs as it wants

Weak PRF suffices for CPA-secure SKE of single-block messages
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 a PRF (w/o increasing input length)

Output is indistinguishable from t random blocks, provided all the inputs to $F_K$ remain distinct (because F itself is a PRF)
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. But no a priori limit on number of blocks in a message.

- Security from low likelihood of \((r+1, \ldots, r+t)\) running into \((r’+1, \ldots, r’+t’)\)

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