CS 6160 Cryptology Lecture 6: Block Ciphers as Pseudorandom Functions

Maria Francis

September 15, 2020

Block Ciphers

- A key ingredient in most symmetric/secret/shared key cryptographic systems.
- Building blocks but security is not assured, depends on the mode of operation
- In this lecture we see an overview (not in depth) of the theoretical foundations of block ciphers.
- The plan over the next 3 lectures :
  - understand pseudorandom permutations used in building block ciphers,
  - see the two standard ways of building block ciphers – SPN and Feistel,
  - look at the two typical block ciphers – DES and AES, and
  - study cryptanalysis of block ciphers
- Note that the design and analysis of block ciphers is an art so some details will remain a mystery!

Block Ciphers

- A block cipher is a function $E : \{0,1\}^n \times \{0,1\}^l \to \{0,1\}^l$.
- E has two inputs : a n-bit string which is the *key* k and a l-bit string *block of plaintext*
- The output of E is a l-bit string called a *ciphertext*
- n,l- *parameters of a block cipher* these values vary from a block cipher to block cipher.
- For each key $k \in \{0,1\}^n$, we define $E_k : \{0,1\}^l \to \{0,1\}^l$ as $E(k, m)$.
- Now E_k is a permutation on $\{0,1\}^l$ one-to-one onto function.
- This implies there is an inverse E_k^{-1}, the inverse block cipher.

Concrete Security of Block Ciphers

- n, l - fixed constants. But theoretically, they are *functions of security parameter*.
- Concrete security vs asymptotic security?
- Concrete says a good cipher resists attacks of time complexity equivalent to brute force search of key.
- So for key length $n = 256$, 2^{128} attack is insecure, even though it is infeasible.
- Asymptotically, secure against $2^{n/2}$ attacks is still secure since it is exponential!
- Concrete security is more stringent since we are looking at actual complexity of attack NOT asymptotic behaviour!

Asymptotic security – Pseudorandom Functions

- Pseudorandom functions are a neat abstraction of block ciphers.
- Generalize the notion of PRGs, instead of random-looking strings, we look at random-looking functions/permutations.
- It is not a fixed function that is pseudorandom but a distribution on functions.
- We have keyed functions E_k with key length($\ell_k(n)$), input length($\ell_{in}(n)$) and output length($\ell_{out}(n)$) all functions in n, the security parameter, i.e. $E_k : \{0,1\}^{\ell_{in}(n)} \rightarrow \{0,1\}^{\ell_{out}(n)}$.
- We assume all are length preserving, $\ell_k(n) = \ell_{in}(n) = \ell_{out}(n) = n$, but not necessarily a permutation!

Pseudorandom Functions

- E_k induces a natural distribution E on functions given by choosing a uniform key $k \in \{0,1\}^n$.
- We call E pseudorandom if the function E_k is indistinguishable from a function f chosen uniformly at random from the set of all functions with the same domain and range (i.e. $f : \{0,1\}^n \to \{0,1\}^n$).
- How to choose a function at random? How big is the space? $|\ \mathrm{Func}\ _n| = 2^{n \cdot 2^n}$.
- Formalizing the idea :
  - Every polynomial time distinguisher D that receives the *description* of pseudorandom function E_k outputs 1 with "almost " same probability as when it is given a description of random function f.
  - But description of f could be exponential since $|\ \mathrm{Func}_n| = 2^{n \cdot 2^n}$, we need lookup table of $n \cdot 2^n$.

Oracle to avoid exponential description

- We give D an access to oracle $\mathcal{O}$ which is either equal to E_k or f.
- Distinguisher queries oracle at any point with x and the oracle returns $\mathcal{O}(x)$.
- The oracle is a black-box but deterministic and gives same output for same input.
- D can only do polynomial number of queries.
- D is not given key k, else distinguishing is trivial.
  - D will query oracle with x, obtain y,
  - Check $E_k(x) = y$ if yes then conclude it was the oracle for E_k, else oracle for f

Pseudorandom Functions

Let $E_k : \{0,1\}^* \to \{0,1\}^*$ be a an efficient length preserving keyed function. E_k is a pseudorandom function if for all PPT distinguishers D, there is a neglible function in n,

$$\begin{aligned}
&|Pr[D(E_k, 1^n) = 1 : k \leftarrow^R \{0,1\}^n] \\
&-Pr[D(f, 1^n) = 1 : f \leftarrow^R \mathrm{Func}_n]| \\
&\leq \mathrm{negl}\,(n).
\end{aligned}$$

NOT a Pseudorandom Function

- Let $E_k(x) = k \oplus x$.
- If k is uniform $E_k(x)$ is also uniformly distributed.
- Consider the following distinguisher D that queries $\mathcal{O}$ on arbitrary, distinct points x_1, x_2 to get $y_1 = \mathcal{O}(x_1)$ and $y_2 = \mathcal{O}(x_2)$.
  - It outputs 1 iff $y_1 \oplus y_2 = x_1 \oplus x_2$
  - If $\mathcal{O} = E_k$, for any k, D outputs 1.
  - For $\mathcal{O} = f$, the probability $f(x_1) \oplus f(x_2) = x_1 \oplus x_2$ is the same as probability $f(x_2) = x_1 \oplus x_2 \oplus f(x_1)$, which is 2^{-n}.
  - The difference is $|1 - 2^{-n}|$, not negligible.

Pseudorandom Permutations

- Let Perm be the set of all permutations (bijections) on $\{0,1\}^n$.
- Just like we had Func_n, we now consider $f \in \mathrm{Perm}_n$.
- What is the size of Perm_n?
  - For the first element in the domain, there are 2^n possible elements in the range. For the second element in the domain, there are only $2^n - 1$ choices and therefore the size is $2^n!$.
- Let E_k be a keyed function, it is a *keyed permutation* if $\ell_{in} = \ell_{out}$ and if for all $k \in \{0,1\}^{\ell_k(n)}$, $E_k : \{0,1\}^{\ell_{in}(n)} \to \{0,1\}^{\ell_{out}(n)}$ is one-one.
- Keyed permutation is efficient if E_k is efficiently computable and efficiently invertible given k.
- For an efficient, keyed permutation to be pseudorandom: is analogous to PRFs.

Pseudorandom Functions and Pseudorandom Permutations

- When we say analogous you can require that E_k needs to be indistinguishable from *a uniform permutation rather than a uniform function*, i.e. $f \in \mathrm{Perm}_n$.
- Turns out that we can still say f a random function.
- For a block length sufficiently long random permutation is indistinguishable from a uniform function.
- For them to be distinguishable means we need to find x and y s.t. $f(x) = f(y)$, however finding such x and y using polynomial number of queries is highly unlikely.

We state the same without proof:

If E is a pseudorandom permutation and $\ell_{in}(n) \geq n$ then E is a pseudorandom function.

Strong Pseudorandom Permutations

- Strongness is introduced to take care of a stronger requirement: the knowledge of E_k^{-1} does not cause a security risk.
- The distinguisher D is now given access to the inverse of the permutation.

Let $E_k : \{0,1\}^* \to \{0,1\}^*$ be a an efficient length preserving keyed permutation. E_k is a strong pseudorandom permutation if for all PPT distinguishers D, there is a neglible function in n,

$$
\begin{aligned}
|Pr[D(E_k, E_k^{-1}, 1^n) = 1 : k \leftarrow^R \{0,1\}^n] \\
-Pr[D(f, f^{-1}, 1^n) = 1 : f \leftarrow^R \mathrm{Perm}_n]| \\
\leq \mathrm{negl}(n).
\end{aligned}
$$

PRFs and PRGs

- PRFs and PRGs are closely related.
  - PRG guarantees that a single output appears random if the input is chosen at random, i.e. $G(x)$ is uniform if x is uniform.
  - PRF guarantees all its outputs appear random regardless of its input provided the function is drawn at random, E_k is chosen by choosing a k at random, not its inputs!
- PRG can be constructed from PRF by simply evaluating it on different inputs.
- PRF from PRG? GGM construction given by Goldreich, Goldwasser, and Micali.

PRFs and PRGs

- A compact representation of an exponentially long pseudorandom string. PRGs always run in poly time and so can only have outputs which are $\mathrm{poly}\ k$, the security parameter.
- PRFs remove the need of the sender and receiver to maintain state and stay in synch to make sure that the pseudorandom pad is not reused.
- PRFs allow for random-access, direct access to any part of the output stream, output of a function $f_k(i)$, ith block of the pseudorandom string with seed k.
- PRFs are a way to achieve random access to a very long pseudorandom string.

Attacks on Block Ciphers

- KPA : attacker is given pairs of inputs/outputs $\{(m_i, E_k(m_i))\}$, with $\{m_i\}$ outside of the attacker's control.
- CPA: attacker is given $\{E_k(m_i)\}$ for inputs $\{m_i\}$ chosen by attacker.
- CCA: attacker is given $\{E_k(m_i)\}$ for inputs $\{m_i\}$ chosen by attacker and $\{E_k^{-1}(c_i)\}$ for chosen $\{c_i\}$.
- Aim:Using above attacks the idea is:
  - Distinguish E_k from a uniform permutation
  - Key-recovery attacks: recover the key k after interacting with E_k.
- *Security against key-recovery is a necessary but NOT sufficient condition for a block cipher.*

Pseudorandom Permutations and Block Ciphers

- A pseudorandom permutation cannot be distinguished from a uniform permutation under a CPA.
- A strong pseudorandom permutation cannot be distinguished even under a CCA. (Note: now the attacker has access to the decryption oracle, i.e. the inverses of E_k.)
- Block ciphers are designed to behave at a minimum as secure instantiations of (strong) pseudorandom permutations/functions with some fixed key length and block length.
- Modeling it as strong pseudorandom permutations allows for proofs of security. E.g: Output of AES is indistinguishable from a random permutation.