COMP 8920: Cryptography Winter 2025

Lecture 7 — Jan 28, 2025

Prof. Curtis Bright *Scribe: Aidan Bennett*

1 Overview

In the last lecture we ended in the middle of proving for any cryptosystem that if $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$, then the cryptosystem is perfectly secure if and only if every key is used with equal probability $\left(\frac{1}{|\mathcal{K}|}\right)$, and for each $x \in \mathcal{P}$ and $y \in \mathcal{C}$, there is a unique key k such that $e_k(x) = y$.

In this lecture we finish the proof, give an application with the one-time pad cryptosystem, and begin a discussion of block ciphers.

2 More on perfect security

The other direction of the proof assumes that $\Pr[k] = \frac{1}{|\mathcal{K}|}$, for all $k \in \mathcal{K}$, and for each $x \in \mathcal{P}$ and $y \in \mathcal{C}$, there is a unique key k such that $e_k(x) = y$. In this case, the argument is similar to the proof that the shift cipher is perfectly secure. □

2.1 One-time Pad

The **One-time Pad**, where the length of the key is equal to the length of the plaintext, and $e_k(x) = x \oplus k$ (where $\oplus$ denotes the bitwise XOR operator), is perfectly secure as a result of the previous theorem, assuming that each k is chosen uniformly at random. This is because there is a unique key k for which $e_k(x) = y$ for all $(x, y) \in \mathcal{P} \times \mathcal{C}$, as k can be easily shown to be $x \oplus y$. The One-time Pad was invented in 1917 by Gilbert Vernam. Its major drawback is that $|\mathcal{K}| \geq |\mathcal{P}|$, meaning that the keys are at least as large as the messages sent.

It is also easily broken with a known-plaintext attack. Given plaintext-ciphertext pair (x, y), k can easily be computed to be $x \oplus y$. The same key can also never be used twice, as doing so could potentially reveal information about the key.

3 Block ciphers

Most modern block ciphers use a sequence of permutation and substitution operations. Commonly they use iteration which uses a **round function** and **key schedule** to encrypt one block for one **round**. The full encryption uses N rounds for some fixed N.

Let K be a random binary key of fixed length, used to construct N round keys $k^1, k^2, \ldots, k^N$ that form the key schedule. The k^i are constructed using a known algorithm. The round function g takes

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_s(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

Table 1: Definition for π_s

z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_p(z)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Table 2: Definition for π_p

two inputs: a round key k^r, and a current **state** w^{r-1}. The next state will be $w^r = g(w^{r-1}, k^r)$. State w^0 is the plaintext, and the final state w^N is the ciphertext. The encryption goes as follows:

$$\begin{array}{ll} \text{Round 0:} & w^0 = x \\ \text{Round 1:} & w^1 = g(w^0, k^1) \\ \vdots & \vdots \\ \text{Round } N\text{:} & w^N = g(w^{N-1}, k^N) = y \end{array}$$

Note that g must be injective in N for decryption to be possible. In this case, $g^{-1}(g(w, z), z) = w$. This can only be done if the key is known.

3.1 Substitution-permutation networks (SPNs)

Substitution-permutation networks (SPNs) are a special kind of iterated cipher, whose round function is based on substitutions and permutations.

Suppose $\ell, m \in \mathbb{N}$ and $\mathcal{P} = \mathcal{C} = \mathbb{Z}_2^{\ell m}$, where ℓm is the block length. And SPN built from permutations $\pi_s : \{0,1\}^\ell \to \{0,1\}^\ell$ and $\pi_p : \{1, \ldots, \ell m\} \to \{1, \ldots, \ell m\}$. π_s is called an s-box, and effectively implements a substitution cipher on bitstrings in $\mathbb{Z}_2^\ell$. π_p permutes $\mathbb{Z}_2^{\ell m}$ via permuting the indices of the bits.

We'll apply π_s to m **chunks** of length ℓ. So if $x \in \mathbb{Z}_2^{\ell m}$, we write $x = x_{\langle 1 \rangle} || x_{\langle 2 \rangle} || \ldots || x_{\langle m \rangle}$. SPNs have N rounds, each consisting of the following:

1. The state is XORed with the round key
2. π_s is applied to all m chunks of the state
3. π_p is applied to the indices of the bits of the state to reorder them

Conventionally, the final round skips applying π_p to simplify decryption, and a final XOR is applied (this is called whitening).

Example: Suppose $\ell = m = 4$. We'll use hexadecimal to represent the bitstrings for simplicity (i.e., $0000 = 0, 0001 = 1, \ldots, 0101 = 9, 0110 = A, \ldots, 1111 = F$). Define π_s as seen in Table 1, and define π_p as seen in Table 2. A circuit depiction of this SPN can be seen in Figure 1.

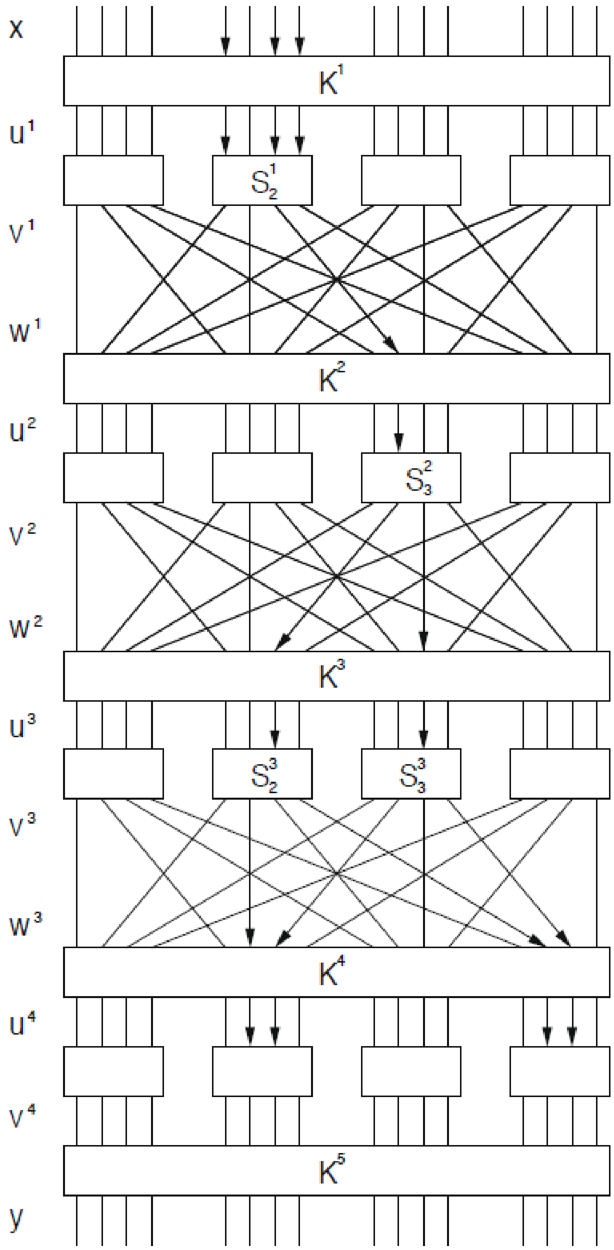

Figure 1: SPN network diagram, from *Cryptography, Theory and Practice*, 4th edition, by Douglas R. Stinson and Maura B. Paterson

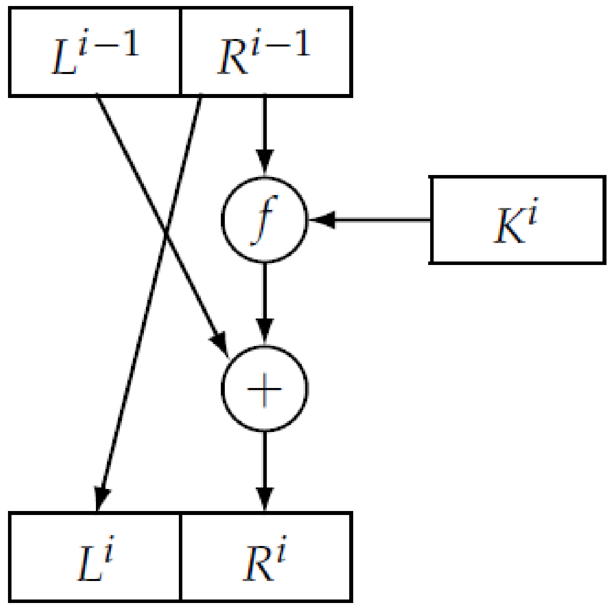

Figure 2: DES round circuit depiction, from *Cryptography, Theory and Practice*, 4th edition, by Douglas R. Stinson and Maura B. Paterson

The encryption process goes as follows.

Suppose $x = 0010$	0110	1011	0111
$k^1 = 0011$	1010	1001	0100
$x \oplus k^1 = 1$	C	2	3
Apply $\pi_s : 4$	5	D	1
Apply $\pi_p : 0010$	1110	0000	0111
$\vdots$			

In practice, s-boxes are implemented via lookup table, so $\pi_s : \mathbb{Z}_2^\ell \to \mathbb{Z}_2^\ell$ needs $2^\ell \cdot \ell$ bits (ℓ bits for each input). As a result, hardware implementations would need to have very small s-boxes.

4 Data Encryption Standard (DES)

In 1973, what is now known as NIST (National Institute of Standards and Technology) solicited a call for a cryptosystem, leading to adapting DES (Data Encryption Standard) as a standard in 1977 after being developed by IBM. It's a type of iterated cipher called a Feistel cipher.

Say state $u^i = L^i || R^i$ (dividing i^i into its left and right halves). The round function g has the form $g(u^{i-1}, k^{i-1}) = u^i = L^i || R^i$, where $L^i = R^{i-1}$ and $R^i = L^{i-1} \oplus f(R^{i-1}, k^i)$ for some function f. f does not need to be invertible, as g will still be invertible via $L^{i-1} = R^i \oplus f(L^i, k^i)$ and $R^{i-1} = L^i$. The circuit depiction can be seen in Figure 2.