Cryptography

Academic Year 2025-2026

Homework 2

Francesco Testa, ID 1179083

November 9, 2025

Exercise 1.

Let's prove that deleting one of 3 phases makes the block cipher easily distinguishable from a random function. So we can introduce the three following block ciphers:

- SPN_k: SPN without the mixing with the key;
- SPN_S: SPN without the S-BOX;
- SPN_p: SPN without the permutation.

For each one, let's create a distinguisher that distinguishes the cipher from a random function.

SPN_k Let's build the distinguisher D_{SPN_k} as in Algorithm 1. Let's call R the number of rounds.

Algorithm 1 $D_{SPN_k}(1^{64})$ # We have access to Oracle O for SPN_K or f

$y \leftarrow O(0^{64})$
$z \leftarrow 0^{64}$
for $i = 1$ to R **do** # R is the number of Rounds, fixed
  $z_1, ..., z_8 = split(z)$ # Split in 8 bytes
  $z \leftarrow P(S_1(z_1)||...||S_8(z_8))$ # S-BOX and Permutation are known
end for
if y==z **then**
  return 1
end if
return 0

This algorithm works thanks to Kerkhoff's principle. In fact, the secrecy of a cipher is obtained obscuring the key, and not obscuring the algorithm itself. Therefore, in this case, we can have access to S-BOX and Permutation, so we can use them to build the distinguisher.

Let's now analyze the probability:

$$|Pr(D_{SPN_k}^{SPN_k(\cdot)}(1^{64}) = 1) - Pr(D_{SPN_k}^{f(\cdot)}(1^{64}) = 1)| = 1 - \frac{1}{2^{64}}$$

which is not negligible. This means that D_{SPN_k} is a distinguisher for SPN_k, therefore isn't pseudorandom.

SPN_S Let's build the distinguisher SPN_S as in Algorithm 2. Let's call R the number of rounds used. This algorithm exploits the linearity of the cipher due to absence of the S-BOX.

Before reading the algorithm, let's observe the idea behind it. In every round, the state is:

$$S_1 = P(m \oplus k_1)$$
$$S_i = P(S_{i-1} \oplus k_i) \text{ for } i = 2, ..., R$$

So, at the end, we have that:

$$S_R = P(\ldots P(P(m \oplus k_1) \oplus k_2) \oplus ... \oplus k_R)$$

Due to linearity of the permutation and XOR, we can rewrite the equation as:

$$S_R = P^R(m) \oplus P^R(k_1) \oplus P^{R-1}(k_2) \oplus ... \oplus P(k_{R-1}) \oplus k_R$$

where P^i is the permutation applied i times.

So we can observe that the message and the key are separated, so we can consider all the second part as the key.

$$K = P^R(k_1) \oplus P^{R-1}(k_2) \oplus ... \oplus P(k_{R-1}) \oplus k_R$$

Therefore the cipher can be rewritten as:

$$S_R = P^R(m) \oplus K$$

So using a CPA attack we can easily find the key K and then test it on a new plaintext.

Algorithm 2 $D_{SPN_S}()$ # We have access to Oracle O for SPN_S or f

$x \leftarrow O(0^{64})$
$y \leftarrow 0^{64}$
for $i = 1$ to R **do** $y \leftarrow P(y)$ # Compute $P^R(0^{64})$
end for
$K = x \oplus y$ # Extract the key
$w = 1^{64}$
$z = O(w)$
for $i = 1$ to R **do** $w \leftarrow P(w)$ # Compute $P^R(1^{64})$
end for
if $z == (w \oplus K)$ **then**
  return 1
end if
return 0

Analyzing the probability, we have that:

$$|Pr(D_{SPN_S}^{SPN_S(\cdot)}(1^{64}) = 1) - Pr(D_{SPN_S}^{f(\cdot)}(1^{64}) = 1)| = 1 - \frac{1}{2^{64}}$$

which is not negligible. This means that D_{SPN_S} is a distinguisher for SPN_S.

SPN_p Without the permutation, the Avalanche effect does not hold. Indeed, changing a single input bit affects only the output byte corresponding to the S-box where that bit is processed.

Using this observation, let's build the distinguisher for SPN_p as in Algorithm 3.

Analyzing the probability, we have that:

$$|Pr(D_{SPN_p}^{SPN_p(\cdot)}(1^{64}) = 1) - Pr(D_{SPN_p}^{f(\cdot)}(1^{64}) = 1)| = 1 - \frac{2^8}{2^{64}} = 1 - \frac{1}{2^{56}}$$

Algorithm 3 $D_{SPN_p}()$ # We have access to Oracle O for SPN_S or f

$y \leftarrow O(0^{64})$
$w \leftarrow O(0^{63}||1)$
$y', \_ \leftarrow split(y)$ # Split the first 7 byte from the last one
$w', \_ \leftarrow split(w)$ # Split the first 7 byte from the last one
if y'==w' **then**
  return 1
end if
return 0

which is not negligible. This means that D_{SPN_p} is a distinguisher for SPN_p. The number of rounds is irrelevant in this case, as the absence of a permutation layer prevents any propagation of bit differences between rounds.

Exercise 2.
We have to prove that:

$$(\mathsf{Gen}, H) \text{ collision-resistant } \implies (\mathsf{Gen}, K) \text{ collision-resistant, where } K^s(x) = H^s(H^s(x))$$

Let's prove it by reduction. Suppose that $\Pi_K = (\mathsf{Gen}, K)$ is not collision-resistant. So exists an adversary A_K such that

$$Pr(\mathsf{HashColl}_{A_K,\Pi_K}(n) = 1) = \eta(n)$$

where $\eta(n)$ is not negligible.
We can use the A_K to build an adversary A_H for (Gen, H), as shown in algorithm 4

Algorithm 4 $A_H(s)$ # We have access to A_K

$x, y \leftarrow A_K(s)$ # $x \neq y \land K^s(x) = K^s(y)$
$w \leftarrow H^s(x)$
$z \leftarrow H^s(y)$
if w==z **then**
  return x,y
end if
return w,z

Let's analyze why the algorithm works in the correct way. First, call the Adversary for K and get $x, y | x \neq y \land K^s(x) = H^s(H^s(x)) = H^s(H^s(y)) = K^s(y)$. Afterwards, let's compute w and z as shown in algorithm. If $w = z$ then the adversary returns x, y. So we have that:

$$x \neq y \text{ (by hypothesis) } \land z = H^s(x) = H^s(y) = w \text{ (if condition)}$$

Otherwise, if $w \neq z$ returns w, z. So:

$$w \neq z \text{ (if condition)} \land H^s(w) = H^s(H^s(x)) = K^s(x) = K^s(y) = H^s(H^s(y)) = H^s(z) \text{ (by hypothesis)}$$

Therefore, let's analyze the probability:

$$Pr(\mathsf{HashColl}_{A_H,\Pi_H}(n) = 1) = Pr(\mathsf{HashColl}_{A_K,\Pi_K}(n) = 1) = \eta(n)$$

which is not negligible, and therefore we have a contradiction. So Π_K is collision-resistant.
We now prove that:

$$(\mathsf{Gen}, H) \land (\mathsf{Gen}, J) \text{ collision-resistant } \implies (\mathsf{Gen}, K) \text{ collision-resistant, where } K^s(x) = H^s(J^s(x))$$

So, using the proof by reduction and De Morgan's laws, we have to prove that:

$$\begin{aligned}\neg(\mathsf{Gen}, K) \text{ collision-resistant } &\implies \neg((\mathsf{Gen}, H) \text{ collision-resistant } \wedge (\mathsf{Gen}, J) \text{ collision-resistant})\\ &\implies \neg(\mathsf{Gen}, H) \text{ collision-resistant } \vee \neg(\mathsf{Gen}, J) \text{ collision-resistant}\end{aligned}$$

In other words, if (Gen, K) is not collision resistant, then at least one of its components $((Gen, H)$ or $(Gen, J))$ must also be not collision resistant.

So let's assume that exists an Adversary A_K for (Gen, K). Let's build an algorithm (Algorithm 5), that works as the Algorithm 4:

Algorithm 5 $A_{HJ}(s)$ # We have access to A_K

$x, y \leftarrow A_K(s)$ # $x \neq y \wedge K^s(x) = K^s(y)$
$w \leftarrow J^s(x)$
$z \leftarrow J^s(y)$
if w==z **then**
  return x,y
end if
return w,z

Let's now analyze the probability, defining two events:

$$\begin{aligned}E_J =& A_{HJ} \text{ breaks } (\mathsf{Gen}, J)\\ E_H =& A_{HJ} \text{ breaks } (\mathsf{Gen}, H)\end{aligned}$$

We knows that when A_K breaks K, then is valid E_J or E_H (disjoint events). So, speaking about the probabilities, we have that:

$$\begin{aligned}Pr(\mathsf{HashColl}_{A_K,\Pi_K}(n) = 1) =& \eta(n)\\ =& Pr(E_J \vee E_H)\\ =& Pr(E_J) + Pr(E_H)\\ =& Pr(\mathsf{HashColl}_{A_{HJ},\Pi_H}(n) = 1) + Pr(\mathsf{HashColl}_{A_{HJ},\Pi_J}(n) = 1)\end{aligned}$$

Therefore, we have that at least one of the two adding probabilities must be non-negligible, otherwise the sum cannot be non-neglibile. So either $Pr(E_J)$ or $Pr(E_H)$ is non negligible, therefore this contradicts the hypothesis that both (Gen, J) and (Gen, H) are collision resistant. In conclusion, this implies that (Gen, K) must be collision resistant.

Exercise 3.

A cyclic and *non*-abelian group cannot exist. Therefore, let's prove that:

$$(\mathbb{G}, \bullet) \text{ cyclic } \implies (\mathbb{G}, \bullet) \text{ abelian}$$

By definition, a group is cyclic if exists a generator, i.e.:

$$\exists g \in \mathbb{G} \text{ such that } \langle g \rangle = \mathbb{G}$$

We have to prove that the group is abelian, i.e.:

$$\forall a, b \in \mathbb{G}, a \bullet b = b \bullet a$$

So, let's consider two arbitrary elements $a, b \in \mathbb{G}$. For the cyclicity of the group, we can write these elements starting from the generator g:

$$\begin{aligned}a =& g^m \text{ for some } m \in \mathbb{Z}\\ b =& g^n \text{ for some } n \in \mathbb{Z}\end{aligned}$$

Now, let's compute $a \bullet b$ and $b \bullet a$:

$$\begin{aligned} a \bullet b =& g^m \bullet g^n = g^{m+n} \\ b \bullet a =& g^n \bullet g^m = g^{n+m} \end{aligned}$$

Due to the commutativity of integer addition, we have that $m + n = n + m$. Therefore $a \bullet b = b \bullet a$. This result holds for every $a, b \in \mathbb{G}$, so the group is abelian. This means that a cyclic and non-abelian group cannot exist.

C.v.d.

The presence of the order of the group does not change the proof, so the result holds in every case.