Block ciphers

Diego F. Aranha

Institute of Computing
UNICAMP

Introduction

Objectives:

- Visit theoretical formulation of modern block ciphers.
- Discuss attacks on this formulation.

Introduction

Objectives:

- Visit theoretical formulation of modern block ciphers.
- Discuss attacks on this formulation.

Hidden intentions:

- Detect in practice what is **not** a secure block cipher.

Computational security

Kerckhoffs Principle

A cipher should be unbreakable both in theory than in practice.

Example: A cipher should not be breakable with probability lower than 10^{-30} in 200 years in the best computer available.

Computational security

Kerckhoffs Principle

A cipher should be unbreakable both in theory than in practice.

Example: A cipher should not be breakable with probability lower than 10^{-30} in 200 years in the best computer available.

Relaxing perfect secrecy:

- Security is preserved only against *efficient* attacks with a feasible execution time.
- Attacks can have success with small probability.

Different approaches

Concrete approach

A cipher is (t, ϵ)-secure if every adversary with execution time t has success probability upper bounded by ϵ.

Example: Adversary has success $\frac{t}{2^n}$ to break an n-bit key cipher in time t. Time $t = 2^{60}$ in an 1 GHz processor requires 35 years. Using several computers in parallel should reduce this to a few years. Insufficient to many applications.

Different approaches

Concrete approach

A cipher is (t, ϵ)-secure if every adversary with execution time t has success probability upper bounded by ϵ.

Example: Adversary has success $\frac{t}{2^n}$ to break an n-bit key cipher in time t. Time $t = 2^{60}$ in an 1 GHz processor requires 35 years. Using several computers in parallel should reduce this to a few years. Insufficient to many applications.

An event with probability 2^{-60} should occur only once every 100 billions of years (age of the universe is 2^{58} seconds). Hence, a reasonable choice of parameters is $t = 2^{80}$ and $\epsilon = 2^{-48}$, implying $n = 128$.

Different approaches

Concrete approach

A cipher is (t, ϵ)-secure if every adversary with execution time t has success probability upper bounded by ϵ.

Example: Adversary has success $\frac{t}{2^n}$ to break an n-bit key cipher in time t. Time $t = 2^{60}$ in an 1 GHz processor requires 35 years. Using several computers in parallel should reduce this to a few years. Insufficient to many applications.

An event with probability 2^{-60} should occur only once every 100 billions of years (age of the universe is 2^{58} seconds). Hence, a reasonable choice of parameters is $t = 2^{80}$ and $\epsilon = 2^{-48}$, implying $n = 128$.

Limitations: What is the exact computational power of the adversary? What is the implementation? What happens with bigger t?

Different approaches

Alternate relaxations:

- Security parameter n.
- Efficient adversary has computational power polynomial in n, executed in time $O(n^c)$, with $c \in \mathbb{Z}$.
- Honest entities can have polynomial computational power and superpolynomial strategies are ignored.
- Success probability is lower than the *inverse of all polynomials in n*. (negligible).

Asymtoptic approach

A cipher is secure if every PPT adversary (*probabilistic polynomial-time*) has negligible probability.

Need of relaxation

We have that $|\mathcal{K}| < |\mathcal{M}|$, thus perfect secrecy is *impossible* to achieve:

- Given ciphertext c, decrypt c with every possible key $k \in \mathcal{K}$;
- Given ciphertexts c_i from messages m_i, decrypt c_i until you find k such that $\forall i, m_i = Dec_k(c_i)$.
- Given ciphertexts c_i from messages m_i, guess value of k such that $\forall i, m_i = Dec_k(c_i)$.

Brute-force or exhaustive search attack

Adversary has probability of success 1 in time linear to $|\mathcal{K}|$ (exponential in n).

Lucky attack

Adversary has probability of success $1/|\mathcal{K}|$ (negligible in n) with constant execution time.

Need of relaxation

We have that $|\mathcal{K}| < |\mathcal{M}|$, thus perfect secrecy is *impossible* to achieve:

- Given ciphertext c, decrypt c with every possible key $k \in \mathcal{K}$;
- Given ciphertexts c_i from messages m_i, decrypt c_i until you find k such that $\forall i, m_i = Dec_k(c_i)$.
- Given ciphertexts c_i from messages m_i, guess value of k such that $\forall i, m_i = Dec_k(c_i)$.

Brute-force or exhaustive search attack

Adversary has probability of success 1 in time linear to $|\mathcal{K}|$ (exponential in n).

Lucky attack

Adversary has probability of success $1/|\mathcal{K}|$ (negligible in n) with constant execution time.

Conclusion: Computational security limits both attacks.

Composition or product of ciphers

Composition $S_1 \times S_2$ can be classified as follows:

- **Commutative**: $S_1 \times S_2 \equiv S_2 \times S_1$.
- **Idempotent**: $S_1 \times S_1 \equiv S_1$.
  Examples: Shift, Vigenére.
- **Non-idempotent**: $S_1 \times S_2 \equiv S_3$.
  Example: Substitution + transposition.

Security:

- Keys chosen independently!
- Iterating an idempotent system does not add security.

Iterated cipher

Definition

An **iterated cipher** is a cipher represented through the repetition of a composition of elementary ciphers. In other words, $\forall Nr \in \mathbb{N}, S^{Nr}$ is an iterated cipher built from the cryptosystem S.

Formalization:

- Nr is the *number of rounds*.
- The *key schedule* is $\langle K^1, K^2, \ldots, K^{Nr} \rangle$.
- K^i is the *round key* for round $1 \leq i \leq Nr$.
- Each round is described by invertible *round function* $g : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$.
- Useful only when S is non-idempotent.

Important: Key $\langle K^1, K^2, \ldots, K^{Nr} \rangle$ is usually derived from key K through a *known* algorithm.

Iterated cipher

Encryption

$$\begin{aligned} w^1 &\leftarrow g(x, K^1) \\ w^2 &\leftarrow g(w^1, K^2) \\ &\dots \\ w^{Nr-1} &\leftarrow g(w^{Nr-2}, K^{Nr-1}) \\ y &\leftarrow g(w^{Nr-1}, K^{Nr}) \end{aligned}$$

Decryption

$$\begin{aligned} w^{Nr-1} &\leftarrow g^{-1}(y, K^{Nr}) \\ &\dots \\ w^1 &\leftarrow g^{-1}(w^2, K^2) \\ x &\leftarrow g^{-1}(w^1, K^1) \end{aligned}$$

Substitution-permutation network

Definition

A **substitution-permutation network** (SPN) is a special case of iterated cipher where $g : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ is represented by the compositition of substitution and transposition ciphers.

Formalization:

- The quantity lm com $l, m \in \mathbb{N}$ is called *block size*.
- $\mathcal{C} = \mathcal{M} = (\mathbb{Z}_2)^{lm}$.
- $\mathcal{K} \subseteq ((\mathbb{Z}_2)^{lm})^{Nr+1}$.
- Substitution of l bits given by $\pi_S : (\mathbb{Z}_2)^l \to (\mathbb{Z}_2)^l$.
- Transposition of lm bits given by $\pi_P : \{1, \ldots, lm\} \to \{1, \ldots, lm\}$.

Important: π_S adds **confusion**, π_P adds **diffusion**.

Substitution-permutation network

Encryption algorithm

Input: $x, \pi_S, \pi_P, \left\langle K^1, K^2, \ldots, K^{Nr}, K^{Nr+1} \right\rangle$.

1 $w^0 \leftarrow x$

2 **for** $r \leftarrow 1$ **to** $Nr - 1$ **do**

  2.1 $u^r \leftarrow w^{r-1} \oplus K^r$

  2.2 **for** $i \leftarrow 1$ **to** m **do** $v^r_{<i>} \leftarrow \pi_S(u^r_{<i>})$

  2.3 $w^r \leftarrow (v^r_{\pi_P(1)}, \ldots, v^r_{\pi_P(lm)})$

3 $u^{Nr} \leftarrow w^{Nr-1} \oplus K^{Nr}$

4 **for** $i \leftarrow 1$ **to** m **do** $v^{Nr}_{<i>} \leftarrow \pi_S(u^{Nr}_{<i>})$

5 $w^{Nr} \leftarrow (v^{Nr}_1, \ldots, v^{Nr}_{lm})$ (no permutation!)

6 **return** $y \leftarrow w^{Nr} \oplus K^{Nr+1}$

Substitution-permutation network

Encryption algorithm

Input: $x, \pi_S, \pi_P, \langle K^1, K^2, \ldots, K^{Nr}, K^{Nr+1} \rangle$.

1 $w^0 \leftarrow x$

2 **for** $r \leftarrow 1$ **to** $Nr - 1$ **do**

  2.1 $u^r \leftarrow w^{r-1} \oplus K^r$

  2.2 **for** $i \leftarrow 1$ **to** m **do** $v^r_{<i>} \leftarrow \pi_S(u^r_{<i>})$

  2.3 $w^r \leftarrow (v^r_{\pi_P(1)}, \ldots, v^r_{\pi_P(lm)})$

3 $u^{Nr} \leftarrow w^{Nr-1} \oplus K^{Nr}$

4 **for** $i \leftarrow 1$ **to** m **do** $v^{Nr}_{<i>} \leftarrow \pi_S(u^{Nr}_{<i>})$

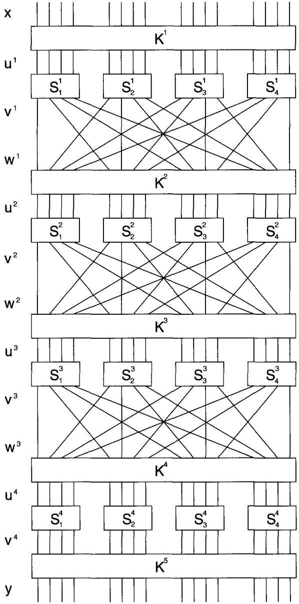
5 $w^{Nr} \leftarrow (v^{Nr}_1, \ldots, v^{Nr}_{lm})$ (no permutation!)

6 **return** $y \leftarrow w^{Nr} \oplus K^{Nr+1}$

Important:

- First and last operations are for *whitening*.
- What is the objective of these operations?
- What are the advantages in the decryption algorithm?

Substitution-permutation network



Substitution-permutation network

Characteristics:

- Efficient both in software and hardware.
- Memory requirements for *substitution boxes* π_S is 2^l bits.
- *Data Encryption Standard:* different π_S for each round.
- *Advanced Encryption Standard:* $l = 8, r \geq 10, lm = 128$.

Linear cryptanalysis

Definition

Linear cryptanalysis is a known-plaintext attack with the objective of recovering bits from the key.

Objective: Find linear approximation of a cipher.

Linear cryptanalysis

Definition

Linear cryptanalysis is a known-plaintext attack with the objective of recovering bits from the key.

Objective: Find linear approximation of a cipher.

Assumptions:

- Possible to find a probabilistic linear relation between plaintext bits and bits in the state immediately before the last round.
- There is a subset of bits such that their addition is biased.
- Attacker knows a large quantity of pairs (x, y) under the key K.

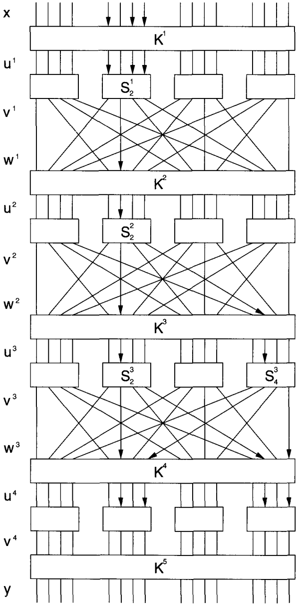
Linear cryptanalysis

Algorithm

1. Choose a small subset of k bits from K^{Nr+1}.
2. Decrypt each $(x \in \mathcal{M}, y \in \mathcal{C})$ using all 2^k combinations of the subset in K^{Nr+1}.
3. For each subkey, compute the state bit and verify if linear relation is still valid.
4. If the relation is valid, increment the frequency counter for that subkey.
5. At the end, the most probable subkey should contain k bits of the key.

Important: In an SPN, approximate substitution boxes and extend to complete cipher!

Linear cryptanalysis



Differential cryptanalysis

Definition

Differential cryptanalysis is a chosen-plaintext attack with the objective of recovering bits from the key.

Objective: Find differences in the input that produce differences in the output.

Differential cryptanalysis

Definition

Differential cryptanalysis is a chosen-plaintext attack with the objective of recovering bits from the key.

Objective: Find differences in the input that produce differences in the output.

Assumptions:

- Possible to find an expected difference in the bits of the state immediately before the last substitution.
- There is a subset of bits which are biased.
- Attacker has a large quantity of (x, x^*, y, y^*) with a chosen difference $x' = x \oplus x^*$ under the same key K.

Differential cryptanalysis

Algorithm

1. Find a small subset of k bits from K^{Nr+1}.
2. Decrypt each (x, x^*, y, y^*), using all 2^k combinations in the subset of bits from K^{Nr+1}.
3. For each subkey, compute k bits of state and verify if difference holds.
4. If the different is valid, increment frequency counter for that subkey.
5. At the end, the most probable subkey should contain the key bits.

Important: In an SPN, find a *differential trail* propagated by the network!

Differential cryptanalysis