A Tutorial of White-Box Cryptography Chapter 3 Implementations

Zheng Gong[1,2]
cis.gong@gmail.com

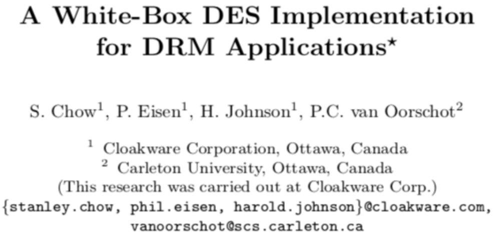
[1]School of Computer Science, South China Normal University
[2]Mobile Applications And Security Engineering Center of Guangdong Province

September 5, 2019

White-box DES

Chow et al.'s white-box DES

- At ACM DRM 2002, Chow et al. proposed a white-box DES implementation for DRM applications.
- The terminology and notation of this implementation are inherited in the following schemes.



A White-Box DES Implementation for DRM Applications*

S. Chow[1], P. Eisen[1], H. Johnson[1], P.C. van Oorschot[2]

[1] Cloakware Corporation, Ottawa, Canada
[2] Carleton University, Ottawa, Canada
(This research was carried out at Cloakware Corp.)
{stanley.chow, phil.eisen, harold.johnson}@cloakware.com,
vanoorschot@scs.carleton.ca

The main concept of encoding

For linear transformation, there are three kinds of encoding terms to obscure the intermediate values of round function.

- encoding
- concatenated encoding
- networked encoding

Encoding

Definition

(encoding) Let X be a transformation from m to n bits. Choose an m-bit bijection F and an n-bit bijection G. Call $X^{'} = G \circ F \circ F^{-1}$ an encoded version of X. F is an input encoding and G is an output encoding.

Concatenated encoding

Definition

(concatenated encoding) Consider bijection F_i of size n_i, where $n_1 + n_2 + \cdots + n_k = n$. Let $||$ denote vector concatenation. The function concatenation $F_1||F_2||\cdots||F_k$ is the bijection F such that, for any n-bit vector $b = (b_1, b_2, \cdots, b_n)$, $F(b) = F_1(b_1, \cdots, b_{n_1})||F_2(b_{n_1+1}, \cdots, b_{n_1+n_2})||\cdots||F_k(b_{n+1} + \cdots + n_{k-1}, \cdots, b_n)$. For such a bijection F, plainly $F^{-1} = F_1^{-1}||F_2^{-1}||\cdots||F_k^{-1}$.

Networked encoding

Definition

networked encoding A network encoding for computing $Y \circ X$ (i.e., transformation X followed by transformation Y) is an encoding of the form:

$$Y' \circ X' = (H \circ Y \circ G^{-1}) \circ (G \circ X \circ F^{-1}) = H \circ (Y \circ X) \circ F^{-1}.$$

Entropy-transference function

${}^{n}_{m}E$ is an *entropy-transference function* such that ${}^{n}_{m}E$ maps m-bit vectors to n-bit vectors, when $m \leq n$ the mapping loses no bits of information, $m > n$ it loses at most $n - m$ bits.

affine transformation (**AT**) function

A vector to vector transformation function P which can be define for all ${}_{m}e$ by

$${}^{n}_{m}P({}_{m}e) =^{n}_{m} M_{m} \cdot e +_{n} d$$

, or

$$P(e) = M \cdot e + d,$$

where M is a constant matrix and d is a constant *displacement/masking/affine* vector.
We consider **ATs** over GF(2). Note if A and B are **ATs**, so $A||B$ and $A \circ B$.

White-box precomputation

- In the black-box model, a cryptographic function is algorithmically implemented beforehand and waits for inputs.
- In the white-box model, the secret key is known first by the white-box tables generator (it might not be needed as server).

Two transformations are required for the white-box cryptosystems:

1. the white-box algorithm: transform the black-box version of algorithm into white-box version
   - affine transformation
   - encoding
2. the white-box key tables: transform the round keys into a series of white-box key tables.

Mixing bijection

A *mixing bijection* is a bijective **AT** which attempts to maximize the dependency of each output bit on all input bits.

- For example, a bit permutation layer P might be sparse in a matrix representation (such as the P layer in DES).
- In order to diffuse more information over more bits, we can represent such a permutation $P = J \circ K$, where $J = P \circ K^{-1}$.

I/O-blocked encoding

An arbitrary function ${}^{n}_{m}P$, where m and n are large, cannot simply be encoded using two arbitrary bijective encodings as $P' = G \circ P \circ F^{-1}$ using a look-up table.

For example, a transformation ${}^{8}_{16}P$ requires $2^8 \times 16 = 2^{12}$ bits (4Kb) storage, whilst ${}^{16}_{64}P$ requires $2^{16} \times 32 = 2^{21}$ bits (2Mb!).

Solution:

- Let $F_P = (F_1 || F_2 || \cdots || Fj) \circ J$ and $G_P = (G_1 || G_2 || \cdots || G_k) \circ K$.
- Concatenated encoding: F_j is ${}^{a}_{a}F$ and G_k. is ${}^{b}_{b}G$.
- $P' = G_P \circ P \circ F_P^{-1}$.

Combined function encoding

For functions P and Q that happened to be evaluated together, we could choose an encoding $P||Q$ such as $G \circ (P||Q) \circ F^{-1}$.

For two 4-bit sboxes $S_1({}_4e)$ and $S_2({}_4e)$, we can combine them together as an 8-bit sbox $(S_1||S_2)({}_8e)$.

By-pass encoding

In general, we want the implementation of each transform to have extra entropy at both the input and output, so that it is more difficult to extract the core.

For a function ${}^{n}_{m}P$, we can put a *by-pass encoding* ${}^{b}_{a}E$ such that ${}^{n+b}_{m+a}P' = G \circ (P||{}^{b}_{a}E) \circ F^{-1}$.

Split-path encoding

Form a function ${}^{n}_{m}P$, we may use an encoding that is really a concatenation of two separate encodings. That is, we define

$$ {}^{n+k}_{m}Q({}_{m}e) = P({}_{m}e)||{}^{k}_{m}R({}_{m}e) $$

The effect is that if P is lossy, Q may lose less information (or no information). This method is used to achieve *local security*.

Output splitting?

We can encode a function P as $k \geq 2$ functions $P_1, P_2, \cdots, P_k$, where the encoded implementation for each part can mix in additional entropy. For example,

$$ {}^{n}_{m}P_2({}_{m}e) = P({}_{m}e) \oplus P_1({}_{m}e) $$

Thanks for your attentions!