이학 박사 학위논문

Analytic Tools for White-box and Lattice Cryptography

(화이트 박스 및 격자 암호 분석 도구)

2016년 2월

서울대학교 대학원

수리과학부

백충훈

Analytic Tools for White-box and Lattice Cryptography

(화이트 박스 및 격자 암호 분석 도구)

지도교수 김명환

이 논문을 이학 박사 학위논문으로 제출함

2015년 10월

서울대학교 대학원

수리과학부

백충훈

백충훈의 이학 박사 학위논문을 인준함

2015년 12월

위원장 이인석 (인)

부위원장 김명환 (인)

위원 이향숙 (인)

위원 오병권 (인)

위원 천정희 (인)

Analytic Tools for White-box and Lattice Cryptography

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Chung Hun Baek

Dissertation Director : Professor Myung-Hwan Kim

Department of Mathematical Sciences
Seoul National University

February 2016

Abstract

Analytic Tools for White-box Cryptography and Lattice Based Cryptography

Chung Hun Baek
Department of Mathematical Sciences
The Graduate School
Seoul National University

In crypto world, the existence of analytic toolbox which can be used as the measure of security is very important in order to design cryptographic systems. In this thesis, we focus on white-box cryptography and lattice based cryptography, and present analytic tools for them.

White-box cryptography presented by Chow et al. is an obfuscation technique for protecting secret keys in software implementations even if an adversary has full access to the implementation of the encryption algorithm and full control over its execution platforms. Despite its practical importance, progress has not been substantial. In fact, it is repeated that as a proposal for a whitebox implementation is reported, an attack of lower complexity is soon announced. This is mainly because most cryptanalytic methods target specific implementations, and there is no general attack tool for white-box cryptography. In this thesis, we present an analytic toolbox on white-box implementations of the Chow et al.'s style using lookup tables. Our toolbox could be used to measure the security of white-box implementations.

Lattice based cryptography is very interesting field of cryptography nowadays. Many hard problems on lattice can be reduced to some specific form of the shortest vector problem or closest vector problem, and hence related to problem of finding a short basis for given lattice. Therefore, good lattice reduction algorithm can play a role of analytic tools for lattice based cryptography. We proposed an algorithm for lattice basis reduction which uses block reduction. This provides some trade-off of reduction time and quality. This can gives a guideline for the parameter setting of lattice based cryptography.

Key words: White-box cryptography, SPN structure, Lattice based cryptography, Lattice reduction algorithm, Block LLL algorithm
Student Number: 2007-20276

Contents

List of Figures

Chapter 1

Introduction

Traditionally, the security of cryptographic algorithms is studied in the black-box model—the end points are trusted and the attacker only has access to the input/output of the algorithm. Under this model, cryptographic schemes are designed to prevent attackers from obtaining secret information using only the input/output values of algorithm without any knowledge of its internal information. In the real world, however, untrusted hosts may access unapproved contents illegally, malicious software in user devices may access the memory used to execute a cryptographic algorithm, or internal information may be leaked during the process of communication. Actually, many attacks have been proposed, such as side channel attacks [GMO01, Koc96, KJJ99, Nov02, QS01], which extract secret information by access to the internal states in the implementation of algorithm. The concept of white-box cryptography has been proposed to enhance security of cryptosystems under such hostile environment.

The white-box cryptography is defined as an obfuscation technique which gives a secure software implementation, by Chow *et al.* in 2002. Its goal is to prevent attackers, who have full access to the implementation, from extracting secret key information. In the past, hardware such as smart cards and trusted platform modules were used to protect internal information. Such hardware is costly and difficult to be replaced by a new one when a flaw is discovered. White-box cryptography is a means of protecting the internal information of the software implementation, and hence, considered one of the tools to supplement the security of devices.

Many commercial products can use white-box cryptography. One of the main applications is in the digital rights management on the commercial devices, such as a PC, a mobile device, or a set-top box for video-on-demand. Nowadays, unlike the past when that commercial contents were delivered in the material form such as CD or DVD, the contents are transmitted to personal devices through the network due to the advance of the communication technology. In this environment, illegal access to the contents and leakage of them are much easier and hence the copyright protection of the contents become more important. The contents should be provided in encrypted form with decryption process which allows to be decrypted in only permitted devices. If an illegal user access them on the network and obtains the decryption key for the content, she can use it in other devices and distribute illegal copies of the content. White-box cryptography aims to prevent attackers from obtaining the decryption key even though attackers have full access decryption process.

The first proposals to implement cryptographic primitives in white-box cryptography were made by Chow *et al.*, who presented a white-box AES implementation [CEJO03] and a white-box DES implementation [CEJvO03] in 2002. They are based on the basic strategy: the whole cipher is decomposed into round functions and the round functions are represented by summation of lookup tables with small size. Although Chow *et al.*'s implementations have been broken with complexity 2^{14} for DES [WMGP07] and 2^{22} for AES [LRM$^+$14], their strategy provided a framework, called "CEJO framework", for designing white-box implementation of using table lookups. Most white-box implementations after Chow *et al.*'s proposal follow the CEJO framework: Xiao and Lai [XL09] proposed white-box AES implementation using wider linear encodings than Chow *et al.*'s. Karroumi [Kar11] modified the algebraic operations in each AES round function using dual representations of the AES cipher and presented a white-box AES implementation. However all of these have been broken in the sense that the secret key can be recovered in the lower complexity than their claimed security when the full lookup tables are given (complexity of 2^{32} and 2^{22}, respectively [MRP13, LRM$^+$14]).

On the other hand, research for white-box cryptography has been proceeded in various ways: Some security notions for white-box cryptography have been studied in [DLPR14, SWP09, Wys09]. Independently, Biryukov *et al.* [BBK14] proposed a new symmetric ASASA-based block cipher with secret S-boxes satisfying white-box security notion, whereas previous works focused on proposing white-box implementation of the existing cipher which is well-known and secure.

As we can see from previous implementations, it is very difficult to design a white-box implementation with a security level similar to the black-box model. Hence, the practical objective of white-box implementations is to increase the complexity of cryptanalysis. All of the implementations mentioned above suffered unpredicted attacks soon after their designs were announced. This is mainly because there are no standard attack tools such as differential cryptanalysis and linear cryptanalysis for block ciphers.

Lattice based cryptography is very interesting field of cryptography nowadays. There are many hard problems on lattices which have been used as a based problem for public key cryptography such as Ajtai-Dwork cryptosystem [AD97], Goldreigh-Goldwasser-Halevi cryptosystem [GGH97], NTRU [HPS98], LWE based cryptosystem and Gentry's fully homomorphic encryption [Gen09]. Lattice is also used as a cryptanalytic tool for public key cryptosystem such as lattice attacks on knapsack cryptosystem and Coppersmith' method for RSA [Gal12]. Additionally, Lattice is used for many applications or security analysis.

Many hard problems on lattice can be reduced to some specific form of the shortest vector problem or closest vector problem, and hence related to problem of finding a “good” basis for given lattice. The meaning of “good” basis varies with the use of the basis, but it usually means short and close to orthogonal. Lattice basis reduction algorithms provide solutions that are required for the shortest vector problem or closest vector problem with some approximate factor, and the relation between running time of the algorithm and approximate factor is used for security analysis of lattice based cryptography. Therefore, good lattice reduction algorithm can play a role of analytic tools for lattice based cryptography.

There are many lattice basis reduction algorithms. The best known lattice basis reduction algorithms are LLL algorithm , HKZ algorithm and BKZ algorithm. The LLL lattice basis reduction algorithm [LLL82] is a polynomial time lattice reduction algorithm invented by Arjen Lenstra, Hendrik

Lenstra and László Lovász in 1982. LLL algorithm is well-analyzed, but the quality of output of the algorithm is not high and hence is not enough to be used widely as an analytic tools of lattice based cryptography. HKZ lattice basis reduction algorithm is an exponential time lattice reduction algorithm proposed by Korkine and Zolotarev. HKZ algorithm provides the exact solution, but the running time is too long and the complexity is high, and hence we can use this algorithm for high dimensional lattice. BKZ lattice basis reduction algorithm [Sch87] proposed by Schnorr in 1987 algorithm provides some trade off between the approximate factor and the running time of the algorithm. The outputs of high quality and various estimation results with various block size and dimension. However, The estimation results are from several experiments or many assumptions and hence does not give theoretic bounds. Therefore, we need to get a new lattice basis reduction algorithm such that is well-analyzed and provides various outputs of high-quality according to the conditions.

1.1 Contributions

Throughout this paper, we focus on white-box implementations of substitution-linear transformation (SLT) ciphers following CEJO framework. Let $E = M \circ S$ be the round function of an SLT cipher on n bits, where M is an invertible linear map and S is a concatenation of S-boxes on m bits with a fixed key. We define the input encoding as $f = A \circ P$, where A is an invertible linear map and P is a concatenation of small nonlinear permutations. If we let g be the input encoding of the next round, then the encoded round function F of E is of the form $F = g^{-1} \circ E \circ f = QBSAP$, where B is an invertible linear map and Q is a concatenation of small nonlinear permutations.

Our contributions are as follow. We present an **analytic toolbox** for white-box implementations of SLT ciphers in the CEJO framework. Our toolbox consists of several algorithms to recover nonlinear and affine encodings used in this model.

First, by adopting the Biryukov–Shamir technique [BS01], we show that the nonlinear part Q can be removed up to an affine transformation in $O\left(\frac{n}{m_Q}2^{3m_Q}\right)$ when $Q = (Q_1, \cdots, Q_{n/m_Q})$ and each Q_i is a nonlinear bijection on m_Q bits. For example, the nonlinear encoding in the Chow *et al.*'s implementation can be removed in 2^{18} bit operations, whereas it takes 2^{29} bit operations using Billet *et al.*'s attack [BGEC05]. While Billet *et al.*'s method is only available when the input size of the S-boxes is the same as the input size of the encodings, ours can be efficiently applied when $m \neq m_Q$.

Second, when $F = B \circ S \circ A$ for affine mappings A, B, it is affine equivalent to S. Hence we can apply the affine equivalence algorithm in [BCBP03], which has a complexity of $O(n^3 2^{2n})$. We improve this algorithm for the case where S consists of small S-boxes of size m. According to our specialized affine equivalence algorithm (SAEA), if the F^{-1} oracle is given, we can find A and B in $O\left(\frac{n}{m} \cdot {m_A}^3 2^{3m}\right)$, where m_A is the smallest integer p such that A (or its similar matrix obtained by permuting rows and columns) is a block diagonal matrix with $p \times p$ matrix blocks. In fact, m_A is the minimal block size when considering A as a block diagonal mapping. When F^{-1} oracle is not given, SAEA requires $O\left(\min\left\{\frac{n}{m} \cdot {m_A}^{m+3} \cdot 2^{2m}, n \cdot \log m_A \cdot 2^{m_A/2}\right\}\right)$, including the complexity of inverting F, to recover the affine encodings.

Our attack is universal in the sense that all known implementations based on the CEJO framework are susceptible to them. Furthermore, they could play a role of estimating the security of possible white-box implementation designs.

We propose a **new design for a white-box implementation** whose security level is close to that of the original cipher. Most variants of Chow *et al.*'s implementation [Kar11, XL09] attempted to increase the security by introducing new affine encodings. According to our toolbox, however, for any affine encoding the complexity for finding the secret key is upper bounded by the minimum of $O\left(\frac{2^{2m}}{m} \cdot n^{m+4}\right)$ and $O\left(n \log n \cdot 2^{n/2}\right)$, which is much lower than 2^n. This provides a negative perspective on secure white-box implementations of SLT ciphers using table lookups.

Our new approach is to use the encryption of multiple plaintexts: For AES-128, we consider the concatenation of two AES-128 ciphers. Let E be a round function of AES-128 and $F = g^{-1} \circ (E, E) \circ f$ be the encoded round function on 256 bits. Then we can take $m_A = 2n > n$ and hence accomplish higher security, up to 2^{110} for $m_A = 256$ and $m = 8$. This approach can be applied to any SLT cipher with $m_A = tn$ for suitable $t \in \mathbb{N}$ and then the security level is large up to $\left(\frac{2 \cdot 4^m}{m} n^{m+4}\right) \cdot t^{m+4}$. Therefore, this provides a new approach for the design of a secure white-box implementation, regardless of the block length of the original cipher. One shortcoming of this approach is its large storage requirement. However, this is compensated by the use of special sparse encodings. We give an instance with storage requirements of about 16 MB and 64 MB for a single round when $m_A = 128$ and 256, respectively, in Section 3.4. Our design does not have a security reduction to well known problems and needs to be scrutinized to get a confidence. However, it is still worthy in that it explains why the previous design trials have been failed and how to overcome this barrier in the current state. We expect our work inspires further research to design a secure white-box implementation.

On the other hands, the goal of lattice basis reduction is to find a good basis from a given lattice basis. The meaning of "good" basis varies with the use of the basis, but it usually means short and close to orthogonal. To obtain a new lattice basis reduction algorithm, we need to make the purpose of it clear. Our purpose of lattice basis reduction is to get cryptanalysis of lattice based cryptosystems and analyze them using the reduction.

We try to get a new lattice basis reduction algorithm that is well-analyzed and provides various outputs of high-quality according to the conditions. We focus on the Babai's nearest plane algorithm for closest vector problem. We proposed an algorithm for lattice basis reduction which uses block reduction. We can consider β-dimensional projection instead of 1-dimensional projection in Babai's nearest plane algorithm. For this we use the concept of Voronoi cell of the lattice and the covering radius of the lattice.

This provides some trade-off of reduction time and quality. If $n = \beta k$, β-

block LLL reduction algorithm gives the solution with approximate factor of $2^{\frac{k}{2}}$, which is better than LLL reduction algorithm with approximate factor of $2^{\frac{n}{2}}$. We can get various outputs of high quality according to the block size β as BKZ reduction algorithm. Furthermore, we can follow the way of analysis of the LLL reduction algorithm and so our algorithm is more easy to analyze than BKZ reduction algorithm. This can give a guideline for the parameter setting of lattice based cryptography.

1.2 Organization

In Chapter 2, we introduce the basic concept of white-box cryptography and previous works. Also we introduce basic lattice theory and LLL algorithm. We propose attack tools that can be applied to a white-box implementation in Chapter 3. An approach to the design of a white-box implementation based on the result of our toolbox and some instance of the design approach are also given there. Our new lattice basis reduction algorithm using block reduction is presented in Chapter 4. Some applications and comparisons will be provided there. We conclude the paper in Chapter 5.

Chapter 2

Preliminaries

2.1 SLT Cipher

A substitution-linear transformation (SLT) cipher defined in [MGH09] is a type of iterated cipher with a couple of substitution layers and linear transformation layers. SLT cipher can be considered a general form of cipher using substitution-permutation network (SPN)[*], because permuting bits is a linear transformation. More precise definition of SLT cipher is as follows.

Definition 2.1.1. The **SLT cipher** $\mathcal{E}$ is defined as follows. It consists of R rounds for some $R \geq 1$. For each $r = 1, \cdots, R$, the r-th round function $E^{(r)}(x_1, \cdots, x_k)$ is a bijective function on n bits, where $n = k \cdot m$ and x_j is an m-bit value for each j and consists of the following three operations:

1. **XOR-ing round key** XOR the r-th round key $K^{(r)} = (K_1^{(r)}, \cdots, K_k^{(r)})$ of n bits to the input $(x_1, \cdots, x_k)$. This outputs $y_i = x_i \oplus K_i^{(r)}$ for all $i = 1, \cdots, k$.

2. **Substitution** Compute $z_i = S_i^{(r)}(y_i)$ for all $i = 1, \cdots, k$, where each $S_i^{(r)}$ is an invertible S-box on m bits in the i-th round.

3. **Linear transformation** For $z = (z_1, \cdots, z_k)$, compute $M^{(r)}z$ where $M^{(r)}$ is an $n \times n$ invertible matrix over GF(2). This n-bit value is the output of the r-th round function.

Note that operation 1,2 realize confusion and operation 3 realizes diffusion.

[*] A SPN is a type of iterated cipher with a couple of substitutions and permutation on bits.

2.2 White-box Implementations

In the black-box model, it is assumed that the encryption algorithm is executed in trusted platforms. Hence, an adversary cannot observe the internal behavior of the encryption process, but can only the external values, such as the plaintext/ciphertext of the encryption algorithm. However, these models are theoretical, and the leakage of secret information can occur in practical implementations. In gray-box models, adversaries can access more information about the internal details of the encryption algorithm. This information includes side channel information related to runtime, power consumption, and fault analysis, which can be leaked by partial access.

In the white-box model, however, it is assumed that the adversary has full access to the implementation of the encryption algorithm and full control over its execution platforms. In this context, the main objective of the adversary is to extract the secret key. That is, the purpose of secure white-box implementations is to prevent the encryption key from being revealed even when internal algorithm details are completely visible in the untrusted platform, and the adversary has full access to the execution of the encryption algorithm.

One approach for secure white-box implementation of a block cipher is to give a table of all input/output values of the encryption. In this case, the security of the implementation of an algorithm is equivalent to the security of the encryption in the black-box model, and hence depends on the security of the encryption scheme itself, regardless of implementations. Unfortunately, such an implementation is not practical, because the storage requirements of the table are prohibitive. For example, the size of the input/output table of AES-128 is $2^{128} \times 128 = 2^{102}$ GB. Chow et al. suggested a white-box implementations with an implementable table size for AES [CEJO03].

2.2.1 Chow et al.'s implementation

In the Chow et al.'s implementation, the basic approach for reducing the table size is to decompose the table into small tables with a composition that composition is equivalent to the original input/output table. The most important factor in the table size is the size of the input affecting each S-box, because the S-boxes cannot be decomposed into smaller parts. In AES, one S-box in a single round is influenced by only 8 input bits, but in more than two rounds, each S-box is influenced by all input bits. Hence, Chow et al. decomposed the whole AES cipher into round functions, and represented these as the composition of small tables whose inputs are those corresponding to each S-box.

Because the round key can be exposed if the input/output values of a single round are provided, the input/output tables of each round must be obfuscated by input/output encoding functions. For equivalence with the original AES, the input encoding of the i-th round is offset by the output encoding of the previous round, as in Fig 2.1 (where $E^{(i)}$ is the i-th round function, $f^{(i)}$ is an input encoding function of the i-th round, and M_{in}/M_{out} are external input/output encodings for security supplement).

The strategy used in the Chow et al.'s implementation can be summarized as follows:

1. The cipher is decomposed into round functions and the round functions are obfuscated by input/output encodings.

2. Each round function is decomposed into a network of lookup tables whose inputs are those corresponding to each S-box.

This strategy provided a framework for designing white-box implementation of block cipher using table lookups. We call it "CEJO framework".

In Chow et al.'s implementation, the encodings composed of nonlinear mappings and linear mappings are used. To prevent an increase in the size of the input that affects each S-box, Chow et al. used 8-bit encodings whose size are the same as that of the size of the S-boxes. A more precise description of

$$\underbrace{M_{out} \circ E^{(r)} \circ f^{(r)}}_{\text{table}} \circ \underbrace{(f^{(r)})^{-1} \circ E^{(r-1)} \circ f^{(r-1)}}_{\text{table}} \circ \cdots \circ \underbrace{(f^{(1)})^{-1} \circ M_{in}}_{\text{table}}$$
$$= M_{out} \circ E^{(r)} \circ \cdots \circ E^{(2)} \circ E^{(1)} \circ M_{in}$$

Figure 2.1: The basic strategy of in the CEJO framework

the encoded round function is as follows. Each encoded round function on 128 bits is composed of four parallel subround functions on 32 bits. In the Chow et al.'s implementation, the subround function F on 32 bits has the form $F = QBMSAP$, where P, Q are concatenations of 4-bit nonlinear permutations, A, B are block diagonal linear mappings with block size 8, S is the bytewise operation of S-boxes, and M is the Mixcolumns operation on 32 bits. Note that an AddRoundKey operation can be merged to the nonlinear encoding P. Because the block size of the encodings is 8 which is the same as the input size of the S-boxes, the ShiftRows operation can be omitted in the round function. (Thus, we consider the encoded round function as a concatenation of four parallel subround functions.) Hence, F can be represented by the summation of four 8-bit to 32-bit lookup tables. For the "summation" of these tables, twenty-four 8-bit to 4-bit XOR tables are required additionally.[†]

2.2.2 BGE Attack

BGE attack [BGEC05] exploited that the input encoding size is the same as that of the S-box in the Chow et al.'s implementation. It consists of three steps. First, they recover the nonlinear parts of the encodings. As the Chow et al.'s implementation only uses input encoding on 8 bits (composition of 8-bit mixing bijection and two 4-bit nonlinear encodings), it is easy to obtain the bijective subfunction of F on 8 bits by fixing three bytes of the input. Using this property, the BGE attack can recover nonlinear parts of the encodings (up to affine) in 2^{24} time. In the second step of the attack, the

[†] As each output value is transformed by a nonlinear encoding, the output values cannot be added directly. Therefore, we need an "XOR table" to perform decoding-XOR-reencoding.

relations between input/output of the table are found using a property of the Mixcolumns operation. Finally, the round key can be found using the result of the second step. The dominant part of this attack's complexity is in the first step, and the total complexity of recovering a 128-bit AES key is 2^{30}.

2.2.3 Michiels et al.'s Cryptanalysis for SLT cipher

The CEJO framework can be applied for designing white-box implementation of any other ciphers, such as a generic class of substitution-linear transformation (SLT) ciphers. Michiels et al. [MGH09] considered the white-box implementation of SLT ciphers based on the CEJO framework and presented the associated cryptanalysis.

Michiels et al.'s use input encodings whose input size is the same as the input size of the S-boxes, as for the original Chow et al.'s implementation. This means the first step of the BGE attack is available to recover the nonlinear parts of the encodings. However, because Michiels et al.'s setting is not only defined on AES, but on any SLT cipher, the other steps of the BGE attack that use the property of AES are not available. Instead, Michiels et al. transformed the encoded round function into a block diagonal mapping whose block size is the same as that of the S-boxes, and recovered the affine encoding of each block using an affine equivalence algorithm [BCBP03]. The reason for transforming the encoded round function into a block diagonal mapping is that the input encodings still have an input size that is the same as that of the S-boxes.

2.3 Lattice Basis Reduction

Lattice is one of the most important primitives of modern cryptography. In this section, we introduce basic lattice theory and some important results of lattice theory related to cryptography. In particular, we introduce the **Lenstra-Lenstra-Lovasz (LLL) Algorithm**, which is one of the most important algorithms dealing with the geometry of numbers, and has applications to cryptanalysis, complexity, and number theory.

2.3.1 Lattice

Lattices are discrete subgroups of $\mathbb{R}^m$. A lattice L is represented by a basis, *i.e.,* a set of linearly independent vectors $b_1, \cdots, b_n$ in $\mathbb{R}^m$ such that L is equal to the set $L(b_1, \cdots, b_n) = \{\sum_{i=1}^n x_i b_i \mid x_i \in \mathbb{Z}\}$ of all integer linear combinations of the b_i's.

Definition 2.3.1. A **lattice** $L \subseteq \mathbb{R}^m$ is a discrete additive group. A set L is

- **discrete** if $\forall x \in L\ \exists \delta > 0$ such $B(x, \delta) \cap L = \{x\}$;
- **additive group** if $x, y \in L \implies x + y, x - y \in L$.

Definition 2.3.2. A **basis** for L is a set $B = \{b_1, \cdots, b_n\} \subseteq \mathbb{R}^{m \times n}$ of linearly independent vectors such that $L(B) = \{Bx \mid x \in \mathbb{Z}^n\}$. The integer n is called the **dimension** of L.

There are some propositions about lattice basis.

Proposition 2.3.1.

1. *L is a lattice if and only if $L = L(B)$ for some basis B.*

We define an important invariant of lattices.

Definition 2.3.3. The **determinant** (or index) of a lattice L is $|\det(B)|$ for any basis B of L.

This is well-defined since changing basis requires elementary row operations on the matrix B, the composition of which is a unimodular matrix with determinant ±1.

Problems related to lattices. An easy problem is the following: given x and L, is $x \in L$? This can be solved easily using linear algebra. Once we start asking about the geometry of our lattice, questions get harder. A hard question is: how close is x to L? Or, find the closest point in L to x. Another hard problem is the **smallest vector problem (SVP)**.

Definition 2.3.4. The **Shortest Vector Problem(SVP)** is the computational problem: given a basis B, find $v \in L(B) - \{0\}$ which minimizes $\|v\|_2$.

Definition 2.3.5. The **Closest Vector Problem(CVP)** is the computational problem: given a basis B and a vector x, find $v \in L(B)$ which minimizes $\|v - x\|_2$.

2.3.2 LLL Algorithm

The goal of lattice reduction is to find bases consisting of reasonably short and nearly orthogonal vectors. Lattice reduction algorithms have many applications, notably public key cryptanalysis where they have been used to break special cases of RSA, DSA, and many lattice based encryptions. There are roughly two types of lattice reduction algorithms:

The **LLL algorithm** is an approximation algorithm for the SVP. It finds $v' \in L(B) - \{0\}$ that satisfies

$$\|v'\|_2 \leq 2^n \|v\|_2$$

in polynomial time, where v is the smallest vector. There is a randomized algorithm that approximates within a factor of $2^{n/\log n}$. Solving the SVP exactly is NP-hard. In fact, finding a $2^{\log^{1-\varepsilon} n}$-approximation is NP-hard. The problem of $\sqrt{n}$-gap SVP (a promise problem where there is a "gap" between two given lattices, and you must distinguish between the two) is in the intersection $\mathbf{NP} \cap \mathrm{co}\mathbf{NP}$, which suggests that it may be easier than the exact SVP problem. Many cryptography applications assume that n^c-

approximation for the gap-SVP requires super-polynomial time, for $c > 1$ (say, $c = 10$).

Before we cover the actual LLL algorithm, we show Gauss' algorithm for solving the SVP in dimension 2.

Gauss' Algorithm. Input: $a, b \in \mathbb{Z}^2$

1. $i \leftarrow \arg\min_j \|b - ja\|$
2. $b \leftarrow b - ia$
3. If $\|b\| \leq \frac{1}{\sqrt{3}}\|a\|$, swap a and b and go to 1, else output $\arg\min\{\|a\|, \|b\|\}$.

The running time is polynomial in the bit length of (a, b) since each swap reduces the sum of the bit lengths of a and b by a constant. Now we show that the algorithm gives us the correct answer.

Proof. Let $v = ia + jb$ be the smallest vector. Write $b = b^* + \alpha \cdot a$, for $\alpha \in \mathbb{R}$, where $b^* \perp a$ and $|\alpha| < \frac{1}{2}$. Then

$$\|v\|^2 = j^2\|b^*\|^2 + (i + \alpha j)^2\|a\|^2 \geq j^2\|b^*\|^2$$

and so $\|v\| \geq j\|b^*\|$. We claim that $\|b^*\| > \frac{1}{2}\|b\|$. This implies that $\|v\| > \frac{j}{2}\|b\|$ which implies that $j < 2$, so $j \in \{0, 1\}$ If $j = 0$, then $i = 1$ and $v = a$, otherwise $j = 1$ and $i = 0$ so $v = b$. Now we prove the claim. We have that $\|b\| > \frac{1}{\sqrt{3}}\|a\|$ and $\|b\|^2 = \|b^*\|^2 + \alpha^2\|a\|^2$ with $|\alpha| < \frac{1}{2}$. Therefore,

$$\begin{aligned}
\|b^*\|^2 &= \|b\|^2 - \alpha^2\|a\|^2 \\
&> \|b\|^2 - 3\alpha^2\|b\|^2 \\
&> \frac{1}{4}\|b\|^2.
\end{aligned}$$

□

The LLL algorithm is similar to Gauss' algorithm. The motivation is to define b_i^* to be b_i minus the projection of b_i to $\text{span}(b_1, \ldots, b_{i-1})$, so $\prod_i b_i^* = \det(B)$. Let $\mu_{ij} \in \mathbb{R}$ be the coefficients such that $b_i = \sum_{j \leq i} \mu_{ij} b_j^*$, $\mu_{ii} = 1$.

LLL Algorithm.

1. "Orthogonalize": make sure $|\mu_{ij}| \leq \frac{1}{2}$, for $j < i$ (takes time $\binom{n}{2}$)

2. "Swap": if there is i such that swapping b_i and b_{i+1} reduces b_i^* by a $\frac{3}{4}$ factor, then swap, else return b_1.

Claim 2.3.1. $\|b_1\| \leq 2^n \min_j \|b_j^*\|$, and therefore $\|b_1\| \leq 2^n \|\text{SVP}\|$ since $\|\text{SVP}\| \geq \min_j \|b_j^*\|$.

Proof. Since $\|b_{i+1}^*\| \geq \frac{1}{2}\|b_i^*\|$, by induction $\|b_1\| = \|b_1^*\| \leq 2^n\|b_i\|$ for any i. □

Claim 2.3.2. The running time of LLL is polynomial in n and the bit lengths of the b_i.

Proof. (Sketch) Define

$$\phi_t = \sum_{i=1}^{n} (n+1-i) \log \|b_1^*\| \text{ after } t \text{ swaps}$$

and it happens that $\phi_{t+1} \leq \phi_t - C$ for some constant $C > 0$. □

Algorithm 1 LLL Algorithm

```
Input: Given basis $b_1, \cdots, b_n$.
Output: LLL reduced basis $b_1, \cdots, b_n$.
Compute $b_1^*, \cdots, b_n^*$.
$i \leftarrow 2$
  while $i \leq n$ do
    for $j = i - 1$ to 1 do
      size reduce $b_i$ w.r.t. $b_j$
    end for
    if $2\|b_i^*\|^2 \geq \|b_{i-1}^*\|^2$ then
      $i = i + 1$
    else
      Swap $b_i$ with $b_{i-1}$ and size reduce for changed part.
      $i = \max\{2, i - 1\}$
    end if
  end while
```

Chapter 3

Analytic Tools for White-box Cryptography

3.1 General Model for CEJO framework

In the Chow et al.'s implementation and Michiels et al.'s modification, the reason for using the input encodings whose input size is the same as that of the S-boxes is to maintain the number of input bits affecting the S-boxes. However, this leads to weakness against BGE attack and Michiels et al.'s cryptanalysis. Therefore, the next step is to extend the form of the encodings for CEJO framework to satisfy both practical and security aspects.

Consider a white-box implementation of an SLT cipher, which follows the CEJO framework. Let $E = M \circ S$ be a round function of the SLT cipher on $n = km$ bits, where M is a linear layer of the SLT cipher and S is a concatenation of S-boxes $S_1, \cdots, S_k$ on m bits.[*] Let f and g be input and output encoding functions, respectively, which are bijections on n bits. Clearly, g is the inverse function of the input encoding function of the next round. Thus, the encoded round function F is defined as $F = g \circ E \circ f$. We first consider an extended form of the encoding at $f = A \circ P$, where A is an invertible linear map on n bits and P is a nonlinear permutation. In the CEJO framework, the table size is mainly determined by the size of input affecting each S-box. Therefore, if A and P are arbitrary bijective linear and nonlinear mappings, respectively, the table size would be huge. Hence, we consider a special mappings that ensure the white-box implementation with reasonable size.

Let $A = \begin{bmatrix} A_1 \\ \vdots \\ A_k \end{bmatrix}$, where A_j is the j-th horizontal strip of size $m \times n$, and let P be a concatenation of nonlinear bijective encodings $P_1, \cdots, P_{k_P}$ on m_P bits, where $n = k_P \cdot m_P$. The output of $f_j = A_j \circ P$ is the input of S_j, and hence the net input[†] size of f_j determines the table size related to S_j. The net input size of f_j is related to the net input size of A_j, and the net input size of A_j is the number of nonzero columns in A_j. Therefore, if we can

[*] For a fixed key, the adding key operation can be merged with the nonlinear permutation or the S-box.

[†] The net input of a function is the part that really affects the output of the function.

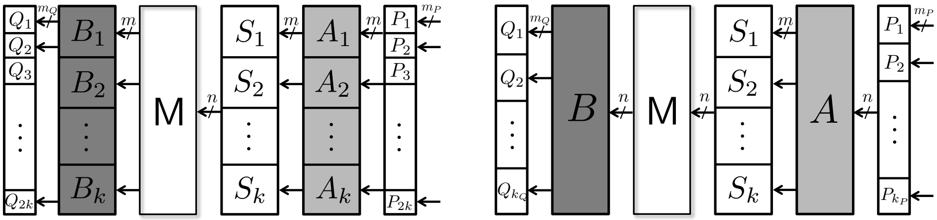

Figure 3.1: Original encodings and extended encodings for the CEJO framework

ensure a small number of nonzero columns in A_j, the table size will be small. Furthermore, since these net input bits are affected by the corresponding P_{j_t}, we should aim for few P_{j_t}, each with a small number of input bits. Therefore, P should be a concatenation of small nonlinear permutations, and A should be an invertible linear map, where each A_j has a small number of nonzero columns.

Since the output encoding is the inverse of the input encoding of the next round, we can write the output encoding g as $g = Q \circ B$, where Q is a concatenation of small nonlinear permutations and B is an invertible linear map. Thus, the encoded round function is of the form $F = QBMSAP$. For simplicity, we write BM as B, because B, M are invertible linear maps, and M is known, *i.e.,* we let $F = QBSAP$.

We may consider the case $f = P \circ A$, where the encoded round function is of the form $F = BQMSPA$. In this case, since the $n \times n$ linear map B follows the Q layer, the XOR tables should decode the encoding BQ, rather than Q. This will make the size of XOR tables huge, and hence we must decompose F into two parts after the Q layer. That is, we let $F = G \circ H$ and make input/output tables of G and H, where $G = B \circ Q_1$ and $H = Q_2MSPA$ with $Q = Q_1 \circ Q_2$. However, if we combine H with G from the previous round, the function is of the form $Q'MSP'$, because the linear mappings A and B (from the previous round) will be canceled out. Since this is covered by the case $f = A \circ P$, we do not consider it further in this paper. Similarly, for a composition of more than two encodings, we just consider the case $f = A \circ P$ as a generalized form of the encoding.

Remark 3.1.1. In practice, m_P cannot be much larger than m, because the use of a nonlinear encoding of size m_P induces the use of a $2m_P$-bit to m_P-bit XOR table. Hence, the choice of m_P is limited. However, we need not be restricted to $m_P \mid m$ or $m \mid m_P$. The nonlinear encodings of the P layer do not need to be aligned with the S-boxes because two different f_j can share some input bits and input encodings. For example, let $n = 192$, $m = 8$, $m_P = 6$, and A be a block diagonal linear mapping with block size 8. For all $j = 1, \cdots, 24$, the number of net input bits of f_j is 12. f_1, f_2 share the same input bits corresponding to P_2 and f_2, f_3 share the same input bits corresponding to P_3. The tables related to each S-box are 12-bit to 192-bit tables and the total table size (including XOR tables) for a round is about 4.4 MB.

Notation

In the remainder of this paper, we define $E = M \circ S$ to be a round function of the SLT cipher with block size n, where M is a linear mapping on n bits and S is a layer of k S-boxes on m bits. We let $F = QBSAP$ be the encoded round function of E, where P, Q are layers of small nonlinear permutations, A, B are layers of linear mappings on n bits, and each A_j has a small number of nonzero columns (A_j is the j-th $m \times n$ horizontal strip of A). Note that B contains M.

We also define variables for the input size of the mappings. For the encoded round function $F = QBSAP$ on n bits, we let P, Q be layers of k_P nonlinear bijective encodings on m_P bits and k_Q nonlinear bijective encodings on m_Q bits, respectively. Furthermore, if A is a block diagonal map consisting of mixing bijections on each block, then we write m_A to denote the size of the blocks and k_A for the number of blocks (*i.e.,* $n = k \cdot m = k_P \cdot m_P = k_Q \cdot m_Q = k_A \cdot m_A$).

3.2 Attack Toolbox for White-Box Implementation

In white-box cryptography, the attacker's objective is to extract the secret key information. Most block ciphers have key schedules, and so cryptanalysis focuses on recovering one round key. In order to extract the secret key, we find the encodings of consecutive two round functions. Using the relation between the output encodings and the input encodings of the consecutive two rounds, the secret key can be extracted efficiently. Therefore, the goal of this section is the extraction of the secret encodings used to obfuscate in the implementation.

We introduce general tools to recover encodings in $F = QBSAP$ as defined in the previous section. We first recover nonlinear parts of encodings up to affine transforms and then we can let $F = B \circ S \circ A$, where A, B are invertible affine maps. Next, we propose attack tools to find A and B in general cases.

3.2.1 Recovering Nonlinear Encodings

Usually, recovering nonlinear parts of encoding is very difficult, but in white-box implementations it is easier because only small nonlinear encodings are used. Billet et al. [BGEC05] presented a method to recover nonlinear parts of the encoding in Chow et al.'s implementation [CEJO03] in 2^{3m} steps. Billet et al. applied this method to only the case that the size of encoding blocks is the same as the size of S-boxes, more precisely, $\mathrm{lcm}(m_P, m_A, m_Q) = m$, where lcm means the least common multiple. Actually, in Chow et al.'s implementation [CEJO03] the size of the S-boxes and the mixing bijections is 8 and the size of the nonlinear encodings is 4. The BGE attack can be easily extended to the case that $\mathrm{lcm}(m_P, m_A, m_Q)$ divides m by regarding $\frac{m}{m_P}$ encodings in layer P, $\frac{m}{m_A}$ mixing blocks in layer A and $\frac{m}{m_Q}$ encodings in layer Q as a single encoding in the P, A, Q layers, respectively.

How about the case that $\mathrm{lcm}(m_P, m_A, m_Q)$ does not divide m? In this case, also the BGE attack can be applied to the implementation if $\mathrm{lcm}(m_P, m_A, m_Q, m) < n$, by considering $\mathrm{lcm}(m_P, m_A, m_Q, m)$ as the size of encodings in the P, A, Q and S layers. The complexity of this attack is $2^{3\mathrm{lcm}(m_P, m_A, m_Q, m)}$, and no longer depend only m. For example, consider the case that $n = 192, m_P = m_Q = 6$ and $m_A = m = 8$, the BGE attack has complexity 2^{75}. This gives the following theorem which is extended version of the BGE attack.

Theorem 3.2.1. *Let $F = QBSAP$ be an encoded round function of white-box implementation as defined in Section 2.2. If $l = lcm(m_P, m_A, m_Q, m) < n$, then one can recover a nonlinear part Q (up to affine transformation) in time $\frac{n}{l} \cdot 2^{3l}$.*

In this subsection, we introduce a more efficient tool to recover nonlinear parts of encodings for the latter case, which is based on the multiset attack of Biryukov and Shamir [BS01]. Using this tool, we can recover nonlinear parts of encodings efficiently even if the size of linear mixing bijections is larger than the size of the S-boxes or the layer of the nonlinear encodings is not aligned with the layer of the S-boxes. This is first approach which provides a link between the technique in [BS01] and cryptanalysis of white-box implementation.

In order to explain this tool, we will use the multiset properties as in [BS01]. For more general attack, we add a subscript to each property symbol to denote the size of input. For a multiset M of m-bit values ($m > 1$), the multiset properties are defined as follows:

- M has property C_m (constant) if it contains only numbers of a single m-bit value.

- M has property P_m (permutation) if it contains all numbers of the 2^m possible values exactly once.

- M has property E_m (even) if each value occurs an even number of times or does not occur.

- M has property B_m (balanced) if the XOR of all the values is 0^m.

We extend this notation to denote combined properties. First, we define a projection map $\pi_I : \{0,1\}^n \to \{0,1\}^\tau$ by $\pi_I(x_1, \cdots, x_n) = (x_{i_1}, \cdots, x_{i_\tau})$, for index set $I = \{i_1, \cdots, i_\tau\} \subseteq \{1, \cdots, n\}$. We say a multiset M of n-bit values has property $\mathsf{P}_{2m}^k \mathsf{C}_{n-2km}$, if $\pi_{\{2im+1,\cdots,2im+2m\}}(M)$ has property P_{2m} for each $i = 0, \cdots, k-1$ and $\pi_{\{2km+1,\cdots,n\}}(M)$ has property C_{n-2km}.

Now let us consider how the multiset properties are transformed by an affine mapping, in the following two lemmas.

Lemma 3.2.1. *Let $A : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ be an affine mapping and $I = \{i_1, \cdots, i_\tau\} \subseteq \{1, \cdots, n\}$ with $\tau \geq m > 1$. For a multiset M of n-bit values, a multiset $A(M)$ has property P_m or E_m if $\pi_I(M)$ has property P_τ and $\pi_{\{1,\cdots,n\}\setminus I}(M)$ has property $\mathsf{C}_{n-\tau}$.*

Proof. We may assume that A is linear, because an addition by a constant preserves property P_m or E_m when M has an even number of elements.

Let $A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$ with column vectors a_i's and $A^* = \begin{bmatrix} a_{i_1} & \cdots & a_{i_\tau} \end{bmatrix}$. Then we have $A(M) = \{A^*(x') + b \mid x' \in \pi_I(M)\}$ for some constant vector $b \in \mathbb{Z}_2^m$. It is enough to show that the multiset $A^*(\pi_I(M))$ has property P_m or E_m.

If $\tau = m$ and A^* has rank m, then the multiset $A^*(\pi_I(M))$ of m-bit values has property P_m. Otherwise, the size of the kernel of A^* is $2^{\tau - \mathrm{rank}(A^*)}$ and hence the number of preimage of $y \in A^*(\pi_I(M))$ is $2^{\tau - \mathrm{rank}(A^*)}$. Since $2^{\tau - \mathrm{rank}(A^*)}$ is even, the multiset $A^*(\pi_I(M))$ has property E_m. It follows that $A^*(\pi_I(M))$ has property P_m or E_m. □

Lemma 3.2.2. *Let $A : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ be an affine mapping. For a multiset M of n-bit values, the multiset $A(M)$ of m-bit values has property B_m if M has property B_n and the size of M is even.*

Proof. We can write $A(x) = L(x) + b$ for some linear mapping L from n bits to m bits and a m bits value b. Then

$$\sum_{y \in A(M)} y = \sum_{x \in M} (Lx + b) = L\left(\sum_{x \in M} x\right) + \sum_{x \in M} b = 0$$

since the multiset M has property B_n and the size of M is even. □

Using these lemmas, we obtain the following theorem, a generalized version of the result in [BS01]. This attack tool which can remove the nonlinearity of encodings is more efficient than Billet et al.'s attack.

For description of the theorem, we provide some definitions. We say a function $f : \mathbb{Z}_2^u \to \mathbb{Z}_2^v$ with $u \geq v$ is balanced if every output occurs 2^{u-v} times. We define $F^{i,\alpha} : \mathbb{Z}_2^{i\cdot m_P} \to \mathbb{Z}_2^n$ as $F^{i,\alpha}(x) := F(\alpha_1, x, \alpha_2)$ where $\alpha = (\alpha_1, \alpha_2)$ and $\alpha_1 \in \mathbb{Z}_2^{t\cdot m_P}, \alpha_2 \in \mathbb{Z}_2^{n-(i+t)\cdot m_P}$ for some $0 \leq t \leq k_P - i$ and $F_j^{i,\alpha} := \pi_j \circ F^{i,\alpha}$, where π_j is a projection onto the j-th block of layer Q. Lastly, we define a set of functions $\Lambda_{i,j} = \{F_j^{i,\alpha} \mid \alpha \in \mathbb{Z}_2^{t\cdot m_P} \times \mathbb{Z}_2^{n-(i+t)\cdot m_P}\text{for some } 0 \leq t \leq k_P - i\}$.

Theorem 3.2.2. *Let $F = QBSAP$ be a round function of white-box implementations and $\Lambda_{i,j}$ be a set of functions defined above where $i = \lceil \frac{m}{m_P} \rceil$. Assume the probability that a function in $\Lambda_{i,j}$ is not balanced is at least $\mathfrak{p} > 0$ for each j. If lcm(m_P, m_A, m_Q) does not divide m, then one can recover nonlinear part Q (up to affine transformation) using $2^{i\cdot m_P + m_Q} \cdot O(1/\mathfrak{p})$ chosen plaintexts in about $O(k_Q \cdot 2^{3m_Q})$ bit operations.*

Proof. Let α be an $(n - i \cdot m_P)$-bit value. For some t, take M_α to be a set with property $\mathsf{C}^t_{m_P}\mathsf{P}_{(i\cdot m_P)}\mathsf{C}^{k_P-(i+t)}_{m_P}$ such that $\pi_{\{1,\cdots,t\cdot m_P,(i+t)m_P+1,\cdots,n\}}(x) = \alpha$ for each $x \in M_\alpha$.

The property $\mathsf{C}^t_{m_P}\mathsf{P}_{(i\cdot m_P)}\mathsf{C}^{k_P-(i+t)}_{m_P}$ is preserved by the layer P, and thus output multiset has also property $\mathsf{C}^t_{m_P}\mathsf{P}_{(i\cdot m_P)}\mathsf{C}^{k_P-(i+t)}_{m_P}$. Since A can be divided into k affine mappings from n-bit to m-bit and $i \cdot m_P \geq m$, this property is transformed by the layer A into the multiset with property $(\mathsf{P}_m \text{ or } \mathsf{E}_m)^k$ by Lemma 3.2.1. Since the property $(\mathsf{P}_m \text{ or } \mathsf{E}_m)^k$ is preserved after layer S, the multiset after layer S has the property B^k_m and this property is equivalent to property B_n. By Lemma 3.2.2, the property B_n is transformed by the layer B into the multiset with property $\mathsf{B}^{k_Q}_{m_Q}$ by dividing B into k_Q affine mappings from n-bit to m_Q-bit.

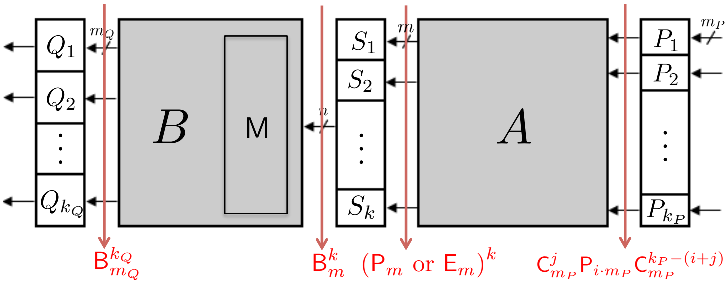

Figure 3.2: The relations between multiset properties on QBSAP

Now, consider the j-th nonlinear bijective encoding Q_j in the layer Q and define $F_j = \pi_j \circ F$, where π_j is a projection onto the j-th block. Then we get a homogeneous equation

$$\sum_{x \in M_\alpha} Q_j^{-1}(F_j(x)) = 0^{m_Q}$$

and since we know the values of $F_j(x)$ for all $x \in M_\alpha$, this equation is a homogeneous equation of the unknowns $Q_j^{-1}(y)$'s for all y through m_Q-bit values, *i.e.*,

$$\sum_y c_{\alpha,y} \cdot Q_j^{-1}(y) = 0^{m_Q}$$

where $c_{\alpha,y}$ is the number of $x \in M_\alpha$ satisfying $F_j(x) = y$.

Since the number of unknowns is 2^{m_Q}, we need more than 2^{m_Q} equations. If we use different constant α at the part correspond to property C from $\mathsf{C}_{m_P}^t \mathsf{P}_{(i \cdot m_P)} \mathsf{C}_{m_P}^{k_P-(i+t)}$, we are likely to get a different homogeneous equation of $Q_j^{-1}(y)$'s. By the assumption, we can obtain 2^{m_Q} equations from $2^{m_Q} \cdot O(1/\mathfrak{p})$ multisets, then we can solve the system of equations by Gaussian elimination. We can do this process for all j's and hence we need $O(k_Q 2^{3m_Q})$ bit operations with $2^{i \cdot m_P + m_Q} \cdot O(1/\mathfrak{p})$ chosen plaintexts to recover the layer Q up to affine transformation. □

The BGE attack take $2^{3\text{lcm}(m_P, m_A, m_Q, m)}$ bit operations, but our attack tool only takes 2^{3m_Q} bit operations. Reconsider example for $n = 192, m_P = m_Q = 6$ and $m_A = m = 8$. In this case, our attack tool reduces the complexity from 2^{75} to 2^{23} to remove the layer Q up to affine transformation. Therefore, our attack tool is useful for removing the non-linear encodings in white-box implementation, whether the nonlinear encodings and the S-boxes are aligned or unaligned.

Remark 3.2.1. To apply the method in Theorem 3.2.2, we require sufficiently many homogeneous equations of the form $\sum_{x \in M_\alpha} Q_j^{-1}(F_j(x)) = 0^{m_Q}$. It is related to the probability $\mathfrak{p}$ because if $F_j^{i,\alpha}$ is balanced, the equation is a trivial equation. By the nonlinearity of S-boxes, if $F_j^{i,\alpha}$ is related to more than 2 S-boxes, $F_j^{i,\alpha}$ is likely to be not balanced. So, we have to take care of choosing multiset of plaintexts, so that $F_j^{i,\alpha}$ is related to more than 2 S-boxes and we note that if $\mathrm{lcm}(m_P, m_A, m_Q)$ does not divide m, $F_j^{i,\alpha}$ is related to more than 2 S-boxes. However, in the case that m_A, m_P and m_Q are equal to m, we can acquire only trivial equation, and hence we cannot use this method. Nevertheless, we can also recover the nonlinear parts of the encodings because the BGE attack can be applied to this case (the BGE attack has same complexity as the method in Theorem 3.2.2). Therefore, the toolbox to recover the nonlinearity of the encodings should include both methods with same complexities, considering all the cases.

Actually, we cannot recover Q exactly because we cannot get a system of equation with full rank of 2^{m_Q}, but we can recover Q up to affine transform. Furthermore, we can recover P by attack for the previous round. Therefore, if we assume $k_P = k_Q$, we can recover all nonlinear part of encoding of a round function in $2k_Q \cdot 2^{3m_Q}$ steps.

Applications

In Chow et al.'s implementation [CEJO03], the input bit size of the linear encodings and the S-boxes is 8 and the size of input/output nonlinear encodings is 4: In our notations, $m = m_A = 8$ and $m_P = m_Q = 4$. Thus, applying the result of Theorem 3.2.2, we can recover the nonlinear encodings in $2k_Q \cdot 2^{3m_Q} = 2 \cdot 32 \cdot 2^{3 \cdot 4} = 2^{18}$ time and the complexity is much less than Billet et al.'s [BGEC05], 2^{29}. The only thing to be careful about is to take multiset of plaintexts. We have to take multiset of plaintexts, so that the

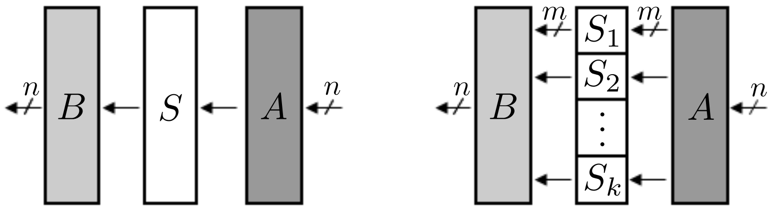

Figure 3.3: Affine equivalence problem and specialized affine equivalence problem

function is related to two S-boxes, for example, the multiset of plaintexts of size 128-bit values that has the property $\mathsf{C}_4\mathsf{P}_8\mathsf{C}_4^{29}$.

3.2.2 Affine Equivalence Algorithm with Multiple S-boxes

We say that two bijections F and S are *linear/affine equivalent* if there exist linear/affine mappings A, B such that $F = B \circ S \circ A$. *The linear/affine equivalence problem* is to find invertible linear/affine mappings A and B such that $F = B \circ S \circ A$ for given nonlinear bijections F and S.

Biryukov et al. [BCBP03] proposed algorithm for solving the linear equivalence problem for arbitrary permutations over $\mathbb{Z}_2^n$ with complexity $O(n^3 2^n)$. For the affine equivalence algorithm, they proposed the concept of the representatives for the linear equivalence classes of permutations and solved the affine equivalence problem in $O(n^3 2^{2n})$ time.

In this subsection, we consider the case that the nonlinear mapping S consists of k invertible S-boxes S_i's which map from $\mathbb{Z}_2^m$ to $\mathbb{Z}_2^m$, where $n = km$, as shown in Fig. 3.3. The problem may be considered to be a specific case of [BCBP03] and so called *the specialized affine equivalence problem.* The following theorem says the problem can be solved more efficiently when compared with the affine equivalence problem.

Theorem 3.2.3. *Let F and S be two permutations on n bits where $S = (S_1, \cdots, S_k)$ with nonlinear permutations S_i on m bits for $i = 1, \cdots, k$. Assume that we can easily access the inversion of F. Then, we can find all affine mappings A and B such that $F = B \circ S \circ A$ in time $O(kn^3 2^{3m})$ if they exist.*

Proof. First, we assume that F and S are linear equivalent. Suppose that A and B are invertible linear mappings over $\mathbb{Z}_2^n$ with $F = B \circ S \circ A$. Let us consider A and B^{-1} to be partitioned into k horizontal strips of size $m \times n$. Denote the i-th strip of A and B^{-1} by A_i and B_i respectively. That is,

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_k \end{bmatrix} \quad \text{and} \quad B^{-1} = \begin{bmatrix} B_1 \\ \vdots \\ B_k \end{bmatrix}. \tag{3.2.1}$$

If one can obtain two sets $\{x_1, x_2, \cdots, x_n\}$ and $\{B_i \circ F(x_1), B_i \circ F(x_2), \cdots, B_i \circ F(x_n)\}$ such that $\{F(x_1), F(x_2), \cdots, F(x_n)\}$ is linearly independent, then one can find B_i from

$$B_i = \begin{bmatrix} B_i \circ F(x_1) & B_i \circ F(x_2) & \cdots & B_i \circ F(x_n) \end{bmatrix} \begin{bmatrix} F(x_1) & F(x_2) & \cdots & F(x_n) \end{bmatrix}^{-1} \tag{3,2.2}$$

where we consider $B_i \circ F(x_j)$ and $F(x_j)$ as column vectors for $1 \leq j \leq n$. Hence, the main strategy is to find two sets $\{x_1, \cdots, x_n\}$ and $\{B_i \circ F(x_1), \cdots, B_i \circ F(x_n)\}$ such that $\{F(x_1), \cdots, F(x_n)\}$ is linearly independent in order to recover B_i.

Suppose that we have two sets $\{x_1, \cdots, x_\ell\}$ and $\{y_1 = B_i \circ F(x_1), \cdots, y_\ell = B_i \circ F(x_\ell)\}$ such that $\{x_1, \cdots, x_\ell\}$ is linearly independent. For any $x = \sum_{j=1}^{\ell} b_j x_j$ ($b_j \in \{0, 1\}$), we can compute $y = B_i \circ F(x)$ from $y_1, \cdots, y_\ell$ by

$$y = S_i \circ A_i(x) = S_i \left(\sum_{j=1}^{\ell} b_j A_i(x_j) \right) = S_i \left(\sum_{j=1}^{\ell} b_j S_i^{-1}(y_j) \right). \tag{3.2.3}$$

Since F is a nonlinear bijection, we can obtain another vector x such that $F(x) \notin \mathbb{Z}_2 F(x_1) + \cdots + \mathbb{Z}_\ell F(x_\ell)$ with high probability. (Assuming F is random bijection, at least one of $\{F(x) \mid x \in \mathbb{Z}_2 x_1 + \cdots + \mathbb{Z}_2 x_\ell\}$ does not belong to $\mathbb{Z}_2 F(x_1) + \cdots + \mathbb{Z}_2 F(x_\ell)$ with probability $1 - \left(\frac{2^{d_\ell}}{2^n}\right)^{2^\ell - \ell}$ where $d_\ell = \dim\langle\{F(x_1), \cdots, F(x_\ell)\}\rangle$.)

On the other hand, suppose that we have two sets $\{F(x_1), \cdots, F(x_\ell)\}$ and $\{y_1 = B_i \circ F(x_1), \cdots, y_\ell = B_i \circ F(x_\ell)\}$ such that $\{F(x_1), \cdots, F(x_\ell)\}$ is linearly independent. For any $x' = F^{-1}(\sum_{j=1}^{\ell} b'_j F(x_j))$ ($b'_j \in \{0,1\}$), we can compute $y' = B_i \circ F(x')$ from $y_1, \cdots, y_\ell$ by

$$y' = B_i \circ F\left(F^{-1}\left(\sum_{j=1}^{\ell} b'_j F(x_j)\right)\right) = \sum_{j=1}^{\ell} b'_j B_i \circ F(x_j) = \sum_{j=1}^{\ell} b'_j y_j. \quad (3.2.4)$$

Since F^{-1} is a nonlinear bijection then we can obtain a new vector x' such that $x' \notin \mathbb{Z}_2 x_1 + \cdots + \mathbb{Z}_2 x_\ell$ with high probability by assuming F^{-1} is random bijection.

Set $x_0 = 0$, $y_0 = B_i \circ F(x_0)$, $x_1 = F^{-1}(0)$ with $F(x_1) = 0$. Then we have $y_0 = S_i \circ A(x_0) = S_i(0), y_1 := B_i \circ F(x_1) = 0$. We need to make an initial guess $y_2 := B_i \circ F(x_2)$ for some $x_2 \in \{0,1\}^n \backslash \{x_0, x_1\}$ to generate another vectors. Note that x_1, x_2 are linearly independent. If we set $x_3 = x_2 + x_1$, then $F(x_3)$ does not belong to $\mathbb{Z}_2 F(0) + \mathbb{Z}_2 F(x_2)$ because F is nonlinear and $x_3 \notin \{x_0, x_1, x_2\}$. By repeating above process in the equation (3.2.3) and (3.2.4) several times, we can successfully obtain n vectors whose F values are linearly independent. For each successful guessing, we get an $m \times n$ linear mapping B_i. We check whether the mapping $S_i^{-1} \circ B_i \circ F$ is linear and reject the incorrect guesses. This process requires n^3 operations for each guessing, and thus the complexity becomes $kn^3 2^m$ to find full matrix B.

Now, let us consider the affine equivalence problem. An affine case is very similar to the linear case. Since an affine mapping is the composition of a linear map and a translation, we can write

$$B_i \circ F(x) + b_i = S_i\left(A_i(x) + a_i\right),$$

for $m \times n$ linear mappings A_i, B_i and the m-bit constant vectors a_i, b_i for $i = 1, \cdots, k$.

For each pair $(a_i, b_i) \in \mathbb{Z}_2^m \times \mathbb{Z}_2^m$, we follow the above process with inputs $F(x)$ and $S_i(x + a_i) + b_i$ and then we can solve the affine equivalence problem. Therefore, the total complexity is $O(kn^3 2^{3m})$ by additionally choosing two m-bit constant vectors. □

We call the algorithm in the Theorem 3.2.3 the *specialized affine equivalence algorithm* (SAEA). While the affine equivalence algorithm has the complexity $O(n^3 2^{2n})$ to find the affine mappings A, B, the SAEA has only complexity $O(kn^3 2^{3m})$. This algorithm gives that the dominant parts of the complexities depend on m, not on n even though A and B are random affine mapping over $\mathbb{Z}_2^n$. Therefore, the SAEA is more efficient whenever S is a concatenation of several S-boxes as in the white-box implementation.

Without the oracle of the invese of F The SAEA requires several evaluations of F^{-1} in equation (3.2.4) and so we can not apply the SAEA directly when the oracle of inversion of F is not given. In that case, we can use only the property in the equation (3.2.3). We have to guess about $\log m_A$ vectors, instead of one vector, to obtain m_A linearly independent vectors, which results in complexity

$$O\left(kn^3 2^{m(\log m_A + 2)}\right) = O\left(kn^{m+3} 2^{2m}\right)$$

for finding the affine encodings. On the other hand, we can use the relation (3.2.4) if we evaluate the required inverse value of F. Using a meet-in-the-middle attack (MITM), one inverse evaluation of F has a time complexity of $O(n2^{n/2})$ and memory requirement of $O(n2^{n/2})$, which can be reduced to $O(n2^{n/4})$ using a dissection-type technique [DDKS12](See the next subsection). Because the SAEA requires about $\log n$ evaluations of F^{-1}, its complexity is

$$O\left(\tfrac{n}{m} \cdot n^3 2^{3m} + \log n \cdot n \cdot 2^{n/2}\right)$$

which is dominated by the complexity of inversion when $n > 6m$. Therefore, the SAEA complexity is

$$O\left(\min\left\{\tfrac{n}{m} \cdot n^{m+3} \cdot 2^{2m}, \tfrac{n}{m} \cdot n^3 2^{3m} + n \cdot \log n \cdot 2^{n/2}\right\}\right)$$

if the oracle of inversion of F is not given.

Meet in the middle attack for inverting F In order to use the SAEA, several number of evaluation of F^{-1} are required, so we need to check the complexity to compute the inverse of F. As in Section 3.2.2, we let F be a bijection on n bits. For simplicity, we assume $F(x) = \sum_{j=1}^{k} F_j(x_j)$ for $F_j : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$, where $x = (x_1, \cdots, x_k)$ and x_j's are m-bit values with $n = km$. A trivial approach to invert F is the exhaustive search, which takes 2^n time complexity. One can improve it using the meet-in-the-middle (MITM) attack: By combining functions, we let $F(x) = G_1(x_1, \cdots x_{\lfloor \frac{k}{2} \rfloor}) + G_2(x_{\lfloor \frac{k}{2} \rfloor + 1}, \cdots, x_k)$. For $y \in \mathbb{Z}_2^n$, one can make a table of $\oplus_y \circ G_1$, sort it by the output values, and compare it with the value of G_2. Then one can evaluate $F^{-1}(y)$ in $O(n2^{n/2})$ time complexity with $O(n2^{n/2})$ memory. The size of required memory for the MITM attack is quite large to implement - for example, 2^{38} GB are required for $n = 128$. We provide another method requiring smaller memory while maintaining asymptotic time complexity.

For convenience of notations, we let $F(x) = \sum_{j=1}^{4} F_j(x_j)$ where $x = (x_1, x_2, x_3, x_4)$ and each x_j is an m_0-bit value with $m_0 = \frac{n}{4}$. For a function f whose value is on n bits, $\tilde{f}$ denotes the projection of f on the first m_0 bits. To evaluate $F^{-1}(y)$ for any n-bit value y, we perform the following steps:

1. Guess m_0-bit value $\tilde{z}$ for $\tilde{F}_1(x_1) + \tilde{F}_2(x_2)$.
2. Perform the MITM attack using $\tilde{F}_1(x_1) + \tilde{F}_2(x_2) = \tilde{z}$ and store the list

$$L = \left\{ (x_1, x_2, F_1(x_1) + F_2(x_2) + y) \mid \tilde{F}_1(x_1) + \tilde{F}_2(x_2) = \tilde{z} \right\}$$

   for the result of the MITM attack.

3. Perform the MITM attack using $\tilde{F}_3(x_3) + \tilde{F}_4(x_4) = \tilde{y} + \tilde{z}$, where $\tilde{y}$ is the first m_0-bit of y.
4. For each (x_3, x_4) satisfying $\tilde{F}_3(x_3) + \tilde{F}_4(x_4) = \tilde{y} + \tilde{z}$, compare $F_3(x_3) + F_4(x_4)$ with the values in L.

Both the average number of elements in L and the number of (x_3, x_4) satisfying $\tilde{F}_3(x_3) + \tilde{F}_4(x_4) = \tilde{y} + \tilde{z}$ are $2^{n/4}$. For one guessing of $\frac{n}{4}$-bit value, we perform 3 times of MITM attacks on the sets with $2^{n/4}$ cardinality. Therefore, we can evaluate $F^{-1}(y)$ in $O(n2^{n/2})$ time complexity with $O(n2^{n/4})$ memory. For the case of $n = 128$, the required memory is 64 GB, which is practical to implement.

Comparing an ASA attack [4] with the SAEA In [BS01], Biryukov and Shamir show that the 3-layered ASA scheme in which each S-layer contains invertible S-boxes and each A-layer contains an invertible affine mapping is susceptible to a chosen-ciphertext attack if S-layers consist of smaller S-boxes. One may attempt to apply the ASA attack to the SAE problem since an instance function $F = BSA$ of the SAE problem has the ASA structure. However, there is a difference between ASA attack and our SAEA: The SAEA targets at finding $\{B, A\}$ tuples with $BSA = F$ for given S and F. However, the ASA attack find $\{B, S, A\}$ tuples with $BSA = F$ for a given F. The solution S is only affine equivalent to the original S. The ASA attack cannot be applied for recovering of the encodings in the white-box implementation.
When A is split When we use the SAEA for the white-box implementation, we can reduce the complexity for the case where input encoding A is of some special form.

For convenience, let A and B be linear. Let's consider $A \in (\mathbb{Z}_2)^{n\times n}$ as a $\tilde{A} \in (\mathbb{Z}_2^{m\times m})^{k\times k}$, where $n = km$. If $\tilde{A}$ is block-diagonal map, then we can perform separately the above attack on each block. If the size of the each block of block-diagonal map $\tilde{A}$ is k_i with $\sum_i k_i = k$, $A_i \in \mathbb{Z}_2^{k_i m\times k_i m}$ is i-th block of A and $B_i \in \mathbb{Z}_2^{n\times k_i m}$ is i-th vertical strip of B correspond to A_i, we can find maps of the form $F_i = B_i \circ (S, \cdots, S) \circ A_i$, where $(S, \cdots, S)$ is concatenation of k_i S-boxes. Since image of F_i has rank k_i, we can find a $k_i \times n$ matrix C_i satisfying that $C_i \circ F_i$ is bijective. Then we obtain bijective maps of the following form:

$$\tilde{F}_i = \tilde{B}_i \circ (S, \cdots, S) \circ A_i$$

where $\tilde{B}_i = C_i \circ B_i$. Thus we can recover the encodings in complexity $\sum_i k_i(k_i m)^3 2^{3m}$, less than $kn^3 2^{3m}$.

More generally, if $\tilde{A}$ can be split into two or more bijective map, that is, A is *split* as defined in section 3.2.2, we can apply the above argument to $\tilde{A}$. In detail, in the case that $(\tilde{A})_{i,j} = 0^{m \times m}$ for $(i,j) \in [k_1+1, k_1+k_0] \times ([1,k] \setminus [k_2+1, k_2+k_0])$ or $(i,j) \in ([1,k] \setminus [k_1+1, k_1+k_0]) \times [k_2+1, k_2+k_0]$ fore some k_0, k_1, k_2 with $k_1+k_0, k_2+k_0 \leq k$,
i.e., $\tilde{A}$ is the form of $\left[\begin{array}{c|c|c} * & 0 & * \\ \hline 0 & A^* & 0 \\ \hline * & 0 & * \end{array}\right]$, one can obtain a bijective map on $\mathbb{Z}_2^{k_0 m}$ using small submatrix, A^* in the above.

Applications In Xiao and Lai's implementation [XL09], they use only the linear mappings for input/output encoding. The input bit size of the input encodings is twice of the input bit size of the S-boxes. By fixing input value on all but 2 bytes as a constant, one can obtain the bijection map F on 16 bits of the following form:

$$F = B \circ (S, S) \circ (\oplus_{K'}, \oplus_{K''}) \circ A$$

where A, B are linear invertible maps on 16 bits. Then F is affine equivalent to (S, S) with linear map B and affine map $(\oplus_{k'}, \oplus_{k''}) \circ A$.

By applying the extended affine equivalence algorithm, we can recover one part of the secret encoding in $\frac{n}{m} n^3 2^{2m} = 2^{29}$ steps for $m = 8$ and $n = 16$. This result is coincident with the result of Mulder et al. [MRP13], $2n^3 2^n = 2^{29}$ steps. However, our attack tool has some potential advantages over Mulder et al.'s: (1) First, as n is larger than twice of m, *i.e.,* $n = km$ with $k > 2$, our attack has less complexity than Mulder et al.'s. For the case of $m = 8$ and $n = 4m = 32$, the complexity to recover one part of the secret encoding is $\frac{n}{m} n^3 2^{2m} = 2^{33}$ using our attack tool, while $2n^3 2^n = 2^{48}$ using Mulder et al.'s method. (2) One additional advantage of our attack is that if we set A and B to be affine mappings instead of linear mappings to increase security, our tool can be applicable to the scheme while Mulder et al.'s method cannot. For the affine case with same n and m, one can recover a secret encoding in $\frac{n}{m}n^3 2^{3m} = 2^{37}$ using our tool.

3.3 Approaches for Resisting Our Attack Tools

There have been many proposals for a new white-box implementation, but none appear to require more than 2^{32} complexity to recover the whole secret key. Hence, the urgent subject of white-box cryptography is to design a white-box implementation of higher security with reasonable storage. In this section, we explore why previous white-box implementations can be attacked with low complexity, and investigate several approaches that may overcome this barrier. Note that we consider an SLT-type block cipher of n-bit inputs with m-bit S-boxes.

Recall that m_A is the size of the minimized blocks of block diagonal affine encodings. More precisely, consider the affine encoding of the form $\oplus_a \circ A$, where A is an invertible matrix in $R^{k\times k}$ with $R = \mathbb{Z}_2^{m\times m}$ and a is an n-bit value. Let k_0 be the smallest integer such that there exist two permutation matrices P_1 and $P_2 \in \mathbb{Z}_2^{km\times km}$ satisfying $P_1AP_2 = \left[\begin{array}{c|c} A_1 & \mathbf{0} \\ \hline \mathbf{0} & A_2 \end{array}\right]$ for some $A_1 \in R^{k_0\times k_0}$. We define $m_A = k_0 \cdot m$.

3.3.1 Limitation of White-Box Implementation

Putting the above theorems together, we can summarize our attacks in the following theorem:

Theorem 3.3.1. (Main Theorem) *For $i = 1, 2, 3$, $F^{(i)} = Q^{(i)} \circ B^{(i)} \circ S^{(i)} \circ \oplus_{K^{(i)}} \circ A^{(i)} \circ P^{(i)}$, bijections on n bits and $S^{(i)}$, a concatenation of $\frac{n}{m}$ nonlinear bijections on m bits are given where $K^{(i)}$ are secret keys of n bits, $P^{(i)}$ and $Q^{(i)}$ are concatenations of $\frac{n}{m_Q}$ nonlinear bijection on m_Q bits, and $A^{(i)}$ and $B^{(i)}$ are invertible linear mappings on n bits, satisfying $Q^{(i)} \circ P^{(i+1)} = id = B^{(i)} \circ A^{(i+1)}$.*

Then, one can find $K^{(2)}$ in time

$$O\left(3\frac{n}{max(m_Q, m)} \cdot 2^{3max(m_Q,m)} + 2\frac{n}{m} \cdot lcm(m_A, m_Q)^3 2^{3m}\right)$$

with $O\left(\frac{2n\log(lcm(m_A,m_Q))}{lcm(m_A,m_Q)}\right)$ *calls of* $(F^{(i)})^{-1}$ *oracle, or in time*

$$O\left(3\frac{n}{max(m_Q,m)}\cdot 2^{3max(m_Q,m)}+2\frac{n}{m}\cdot lcm(m_A,m_Q)^{m+3}2^{2m}\right)$$

without using $(F^{(i)})^{-1}$ *oracle, where* m_A *is the size of minimized blocks of* $A^{(i)}$*'s as difined above.*

Proof. Note that $m|m_A$ by definition of m_A. Since $\mathrm{lcm}(m_A,m_Q)|m$ implies $l=\mathrm{lcm}(m_A,m_Q,m)=m$, one can recover $Q^{(i)}$ (up to affine transformation) in time $O(\frac{n}{max(m_Q,m)}\cdot 2^{3max(m_Q,m)})$ by Theorem 3.2.1 and Theorem 3.2.2 and also can recover $P^{(1)}$ and $P^{(2)}$ from $P^{(1)}=(Q^{(0)})^{-1}$ and $P^{(2)}=(Q^{(1)})^{-1}$.

Now, for $i=1,2$, the nonlinear effects of $P^{(i)}$ and $Q^{(i)}$ can be removed in $F^{(i)}$ and hence $F^{(i)}$ can be considered $F^{(i)}=\tilde{B}^{(i)}\circ S^{(i)}\circ\oplus_{K^{(i)}}\circ\tilde{A}^{(i)}$ for some affine mappings $\tilde{A}^{(i)}$ and $\tilde{B}^{(i)}$ on n bits. Note that $\tilde{A}^{(i)}$ can be considered block diagonal affine mappings with block size $l=\mathrm{lcm}(m_A,m_Q)$. Therefore, one can apply SAEA to each block of size l. When SAEA is applied to block of size l, it needs $\log l$ calls of $(F^{(i)})^{-1}$ oracle or to guess about $\log m_A$ vectors, instead of one vector, without using $(F^{(i)})^{-1}$ oracle. It follows that one can recover $\hat{A}^{(i)}=\oplus_{K^{(i)}}\circ\tilde{A}^{(i)}$ and $\tilde{B}^{(i)}$ in time $O(\frac{n}{m}\cdot l^3 2^{3m})$ with $\frac{n}{l}\log l$ calls of $(F^{(i)})^{-1}$ oracle or $O(\frac{n}{m}\cdot l^3 2^{m(\log l+2)})=O\left(\frac{n}{m}\cdot l^{m+3}2^{2m}\right)$ without using $(F^{(i)})^{-1}$ oracle.

From the relation between $P^{(2)}$ and $Q^{(1)}$, one can find $K^{(2)}$ by computing $\hat{A}^{(2)}\circ\tilde{B}^{(1)}(\mathbf{0})=\oplus_{K^{(2)}}\circ A^{(2)}\circ B^{(1)}(0)=K^{(2)}$. □ □

All of the previous white-box implementations have common features: For $n=128, m=8$, (1) they use affine/linear encodings with $m_A\leq 16$, and (2) they do not use nonlinear encodings, or use nonlinear encodings with only $m_Q=4$. In these case, $\mathrm{lcm}(m_A,m_Q)\leq 16$, and so one can easily compute the inverse. By the result of Theorem 3.3.1, all previous implementations can be broken in less than 2^{41} time without using a specific attack.

To increase the complexity, we need to increase $\mathrm{lcm}(m_A,m_Q)$. Increasing m_Q results in large storage requirements for the XOR table; e.g., one XOR table requires 8 for $m_Q = 16$. Another approach is to increase m_A. We may try to increase m_A up to n. When $m_A = n \leq 128$ and $m = 8$, the complexity is at most 2^{50} if the F^{-1} oracle is given, where F is an encoded round function.

3.3.2 Perspective of White-Box Implementation

In reality, however, the oracle of F^{-1} is not provided, and so we focus on this case. By Theorem 3.3.1, when $m_A = n$, the complexity of the SAEA is $O\left(\frac{n}{m} \cdot {m_A}^{m+3} 2^{2m}\right)$, which is a polynomial of m_A with degree $m + 3$. On the other hand, we could also use the SAEA by evaluating the required inverse values of F. Using a meet-in-the-middle attack (MITM), one inverse evaluation of F has a time complexity of $O(m_A 2^{m_A/2})$ and memory requirement of $O(m_A 2^{m_A/2})$, which can be reduced to $O(m_A 2^{m_A/4})$ using a dissection-type technique [DDKS12] (See Appendix **??**). Because the SAEA requires about $\log m_A$ evaluations of F^{-1}, its complexity is

$$O\left(\frac{n}{m} \cdot m_A^3 2^{3m} + \frac{n}{m_A} \cdot \log m_A \cdot m_A \cdot 2^{m_A/2}\right)$$

which is dominated by the complexity of inversion when $m_A > 6m$. Therefore, when $m_A = n = 128$ and $m = 8$, the SAEA complexity is

$$O\left(\min\left\{\frac{n}{m} \cdot {m_A}^{m+3} \cdot 2^{2m}, n \cdot \log m_A \cdot 2^{m_A/2}\right\}\right) = 2^{74}.$$

A security level of 74 bits is higher than that of previous implementations [CEJO03, Kar11, XL09], but is not sufficient considering the security level of the original cipher. This limitation arises because the complexity is heavily dependent on m_A, but m_A cannot exceed n. We must therefore examine another approach to further increase the security level of the white-box implementation.

Let us consider the case whereby we encrypt messages that are longer than the cipher's block length or multiple messages. We investigate an approach in which multiple plaintexts are simultaneously encrypted by one

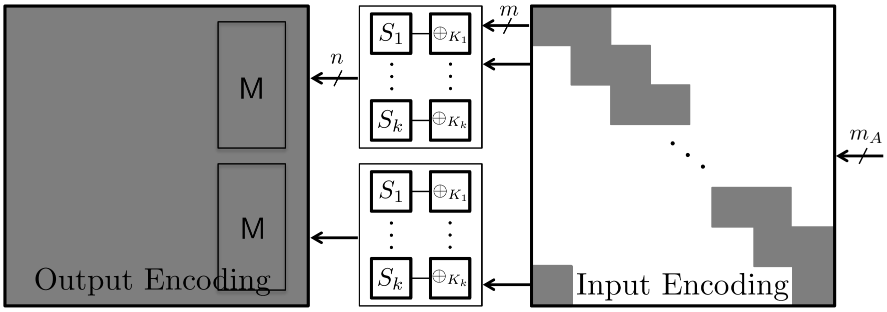

Figure 3.4: The hard-to-invert encoded round function with $m_A = 2n$

white-box implementation. Then, we can take m_A larger than n, such as $m_A = 2n, 3n, \cdots$, and hence the security level can be improved over that stated above. For example, the complexity of the SAEA can be large up to 2^{109} when $m_A = 2n, n = 128$, and $m = 8$. However, this approach may lead to large storage requirement. To overcome this problem, we use special "*sparse unsplit*" encodings, as shown in Fig 3.4. In the next section, we present an instance of this white-box implementation for concatenations of AES-128 using sparse unsplit encodings that are bijections on $m_A = 256$.

3.4 A Proposal for a White-Box Implementation of the AES Cipher

We propose a white-box implementation for concatenation of two AES using table lookups. The AES consists of 10 rounds and the round function of AES is made of the four steps: SubBytes(SB), ShiftRows(SR), MixColumns(MC), and AddRoundKey(ARK). Each step is a bytewise operation on the 16 bytes. For efficiency, we set the round function $\mathrm{AES}^{(r)}$ of AES as follows:

$$\mathrm{AES}^{(r)} = \begin{cases} \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB} \circ \mathsf{ARK}, & \text{if } r = 1, \cdots, 9 \\ \mathsf{SR} \circ \mathsf{AK} \circ \mathsf{SB} \circ \mathsf{ARK}, & \text{if } r = 10 \end{cases}.$$

Input and Output Encodings We use *sparse unsplit* encodings as input encodings like in equation (3.4.5) to reduce the storage: Let $A^r \in \mathbb{Z}_2^{256\times256}$ be an invertible linear map such that $A^r_{i,j}$ is the zero matrix for all $(i,j) \neq (i,i),(i,i+1)$ and $(32,1)$, where $A^r_{i,j}$ is (i,j)-th block of A^r when A^r is partitioned into 1024 blocks $A^r_{i,j}$ of size 8×8 for $i,j = 1,\cdots,32$. We define an input encoding $A^{(r)}$ of r-th round of the form $\oplus_{a^r} \circ A^r$ for a 256-bit value $a^r = (a^r_1, \cdots, a^r_{32})$ where a^r_i's are 8-bit values. That is,

$$A^{(r)}(x) = \begin{bmatrix} A^r_{1,1} & A^r_{1,2} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & A^r_{2,2} & A^r_{2,3} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ A^r_{32,1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & A^r_{32,32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{32} \end{bmatrix} \oplus \begin{bmatrix} a^r_1 \\ a^r_2 \\ \vdots \\ a^r_{32} \end{bmatrix} \tag{3.4.5}$$

for $x = (x_1, \cdots, x_{32}) \in \{0,1\}^{256}$ with 8-bit values x_i's.

For the input encoding $A^{(r+1)}$ of the $(r+1)$-th round, we define the output encoding $B^{(r)}$ of r-th round by $B^{(r)} = (A^{(r+1)})^{-1} \circ (\mathsf{MC} \circ \mathsf{SR}, \mathsf{MC} \circ \mathsf{SR})$, for $r = 1, \cdots, 9$, where $(\mathsf{MC} \circ \mathsf{SR}, \mathsf{MC} \circ \mathsf{SR})$ is a concatenation of $\mathsf{MC} \circ \mathsf{SR}$ on 128 bits. In general, the inversion $(A^r)^{-1}$ has no sparse characteristics despite the sparse structure of A^r. (One can easily check it using Gaussian elimination.) Therefore, the output encoding $B^{(r)}$ has no specific structure. Note that nonlinear encodings of small size are used in the white-box implementation in general and the complexity of removing the nonlinear encodings is not higher than the complexity of finding the affine encodings. We have no consideration for the nonlinear encodings in this section.

Then, the encoded round function $F^{(r)}$ of $\mathrm{AES}^{(r)}$ is defined by $F^{(r)} = (A^{(r+1)})^{-1} \circ (\mathrm{AES}^{(r)}, \mathrm{AES}^{(r)}) \circ A^{(r)} = B^{(r)} \circ (S, \cdots, S) \circ \oplus_{(K^r,K^r)} \circ A^{(r)}$ for $r = 1, \cdots, 9$, where S is the S-box function on 8 bits in the SubBytes step and K^r is the secret key of r-th round on 128 bits in the AddRoundKey step. Since the final round of AES is slightly different from the other rounds, the encoded round function of final round will be discussed later.

Construction of Lookup Tables Let B_i^r be a linear mapping from 8 bits to 256 bits for $i = 1, \cdots, 32$ and b^r be a 256-bit value vector such that $B^{(r)}(x) = [B_1^r|\cdots|B_{32}^r](x) \oplus b^r$ for any 256-bit value x. Choose 256-bit random value b_i^r for each $i = 1, \cdots, 31$, and let $b_{32}^r := b^r \oplus b_1^r \oplus \cdots \oplus b_{31}^r$.

Now, each 16-bit to 256-bit table $F_i^{(r)}$, depicted in Fig 3.5, is defined as follows:

$$F_i^{(r)} = \begin{cases} \oplus_{b_i^r} \circ B_i^r \circ S \circ \oplus_{K_i^r \oplus a_i^r} \circ (A_{i,i}^r, A_{i,i+1}^r), & \text{if } 1 \le i \le 16 \\ \oplus_{b_i^r} \circ B_i^r \circ S \circ \oplus_{K_{i-16}^r \oplus a_i^r} \circ (A_{i,i}^r, A_{i,i+1}^r), & \text{if } 17 \le i < 32 \\ \oplus_{b_{32}^r} \circ B_{32}^r \circ S \circ \oplus_{K_{16}^r \oplus a_{32}^r} \circ (A_{32,32}^r, A_{32,1}^r), & \text{if } i = 32 \end{cases}$$

where $K^r = (K_1^r, \cdots, K_{16}^r)$ is the r-th round key for $K_i^r \in \{0,1\}^8$. The affine mapping $\oplus_{a_i^r} \circ (A_{i,i}^r, A_{i,i+1}^r)$ from $\mathbb{Z}_2^{16}$to $\mathbb{Z}_2^8$ is inserted before $S \circ \oplus_{K_i^r}$ and the affine mapping $\oplus_{b_i^r} \circ B_i^r$ from $\mathbb{Z}_2^8$ to $\mathbb{Z}_2^{256}$ is inserted after the S-box part. Then the encoded round function $F^{(r)}$ of $\mathrm{AES}^{(r)}$ can be expressed as a sum of $F_i^{(r)}$'s:

$$F^{(r)}(x_1, x_2, \cdots, x_{32}) = F_1^{(r)}(x_1, x_2) \oplus F_2^{(r)}(x_2, x_3) \oplus \cdots \oplus F_{32}^{(r)}(x_{32}, x_1)$$

for $r = 1, \cdots, 9$ and 8-bit values x_i's. Therefore, the encoded round function $F^{(r)}$ can be implemented using thirty-two 16-bit to 256-bit lookup tables,

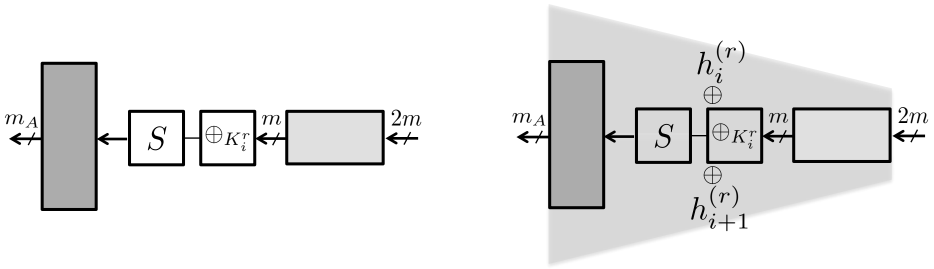

Figure 3.5: The lookup tables of $F_i^{(r)}$ and $T_i^{(r)}$

instead of implementing a 256-bit to 256-bit table of huge size. However, since $F_i^{(r)}(x,0) = \oplus_{b_i^r} \circ B_i^r \circ S \circ \oplus_{K_i^r \oplus a_i^r} \circ A_{i,i}^r(x)$ is a 8 bit-to-128 bit function for 8-bit value x, it can be transformed to a bijection on 8 bits by some projection. Then, it is affine equivalent to S, Hence, the affine equivalent algorithm of [BCBP03] can be applied to each $F_i^{(r)}(x,0)$ and it has only 2^{25} complexity to recover affine mappings. To prevent the individual attack, we modify the functions $F_i^{(r)}$'s. We replace $F_i^{(r)}$ by $T_i^{(r)} : \mathbb{Z}_2^{16} \to \mathbb{Z}_2^{256}$ such that

$$T_i^{(r)}(x,y) = \begin{cases} F_i^{(r)}(x,y) \oplus h_i^{(r)}(x) \oplus h_{i+1}^{(r)}(y), & \text{if } i \neq 32 \\ F_{32}^{(r)}(x,y) \oplus h_{32}^{(r)}(x) \oplus h_1^{(r)}(y), & \text{if } i = 32 \end{cases}$$

for any random function $h_i^{(r)}$ from $\mathbb{Z}_2^8$ to $\mathbb{Z}_2^{256}$ and 8-bit values x, y. The resulting lookup table $T_i^{(r)}$ is depicted in Fig 3.5. Then, if we set $x_{33} := x_1$,

$$\sum_{i=1}^{32} T_i^{(r)}(x_i, x_{i+1}) = \sum_{i=1}^{32} F_i^{(r)}(x_i, x_{i+1}) = F^{(r)}(x_1, \cdots, x_{32})$$

for 8-bit valued x_i's. Furthermore, since the functions $h_i^{(r)}$ have no certain structure unlike $F_i^{(r)}$, we cannot extract encodings from $T_i^{(r)}$'s using the affine equivalence algorithm.

Therefore, we can express the encoded round function $F^{(r)}$ as a sum of 16-bit to 256-bit lookup tables $T_i^{(r)}$'s without revealing $F_i^{(r)}$'s.

External Encoding Tables Let M_{in} and M_{out} be random affine functions on 256 bits. Then, external input encoding function $F^{(0)}$ is defined by $F^{(0)} = (A^{(1)})^{-1} \circ M_{\text{in}}$ and the encoded round function of the final round is defined by $F^{(10)} = M_{\text{out}} \circ (\text{AES}^{(10)}, \text{AES}^{(10)}) \circ A^{(10)}$, where $\text{AES}^{(10)} = \oplus_{K^{11}} \circ \mathsf{SR} \circ (S, \cdots, S) \circ \oplus_{K^{10}}$.

Since the external input encoding function $F^{(0)}$ is an affine function on 256 bits, it is implemented using a matrix of size 256 × 256 and a vector of length 256 bits. The external output encoding function $F^{(10)}$ is split into thirty-two 16-bit to 256-bit lookup tables $T_i^{(10)}$ by the above design technique.

Security Analysis Since the encoded round function $F^{(r)}$ is a bijection on 256 bits, computing the inverse value of $F^{(r)}$ is not an easy task for $r = 1, \cdots, 10$. Therefore, we count the complexity of computing the inverse of $F^{(r)}$ in the cryptanalysis of our proposed white-box implementation using SAEA. The hardness of inverting $F^{(r)}$ for $r = 1, \cdots, 10$, which has sparse unsplit encodings can be considered as a special version of sparse subset sum problem (SSSP) used in [CNT12, GH11, SV14] to design fully homomorphic encryptions.

If A^r is a block diagonal matrix (*i.e.*, all off-diagonal blocks A^r_{ij} $(i \neq j)$ are zero matrices), then $F^{(r)}$ is expressed by the summation of the $F_i^{(r)}$'s, where $F_i^{(r)}$ is a function from $\mathbb{Z}_2^8$ to $\mathbb{Z}_2^{256}$. For a given 256-bit value y and sets $\{(x, F_i^{(r)}(x)) \in \mathbb{Z}_2^8 \times \mathbb{Z}_2^{256} \mid x \in \{0,1\}^8\}_{i=1}^{32}$, computing $(F^{(r)})^{-1}$ of y is to find the 8-bit values x_i such that $y = \sum_{i=1}^{32} F_i^{(r)}(x_i) = F^{(r)}(x_1, \cdots, x_{32})$. It is equivalent to a variant of the SSSP problem to finding the coefficients $\delta_{i,x}$, such that

$$y = \sum_{i=1}^{32} \sum_{x \in \{0,1\}^8} \delta_{i,x} \cdot F_i^{(r)}(x),$$

where

$$\delta_{i,x} \in \{0,1\}, \#\{x \in \{0,1\}^8 \mid \delta_{1,x} = 1\} = \cdots = \#\{x \in \{0,1\}^8 \mid \delta_{32,x} = 1\} = 1.$$

In this case, if we regard $F^{(r)}$ as a bijection on m_A bits, this SSSP can be solved in $\widetilde{O}(2^{m_A/2})$ time with $\widetilde{O}(2^{m_A/4})$ memory using a variant of Schroeppel–Shamir algorithm [SS79]. This is the same complexity as for the proposed MITM attack. However, the presented implementation uses an unsplit encoding that is not a block diagonal mapping. As a result, the 8-bit value x_i is used as the input value of several $T_j^{(r)}$'s, and the computation of $(F^{(r)})^{-1}$ is slightly different from that in the SSSP. Using the unsplit encoding instead of a block diagonal encoding makes the computation of $(F^{(r)})^{-1}$ from the subfunctions $T_j^{(r)}$ more difficult. Therefore, computing $(F^{(r)})^{-1}$ from the subfunctions $T_j^{(r)}$ seems as difficult as in the SSSP, and the subfunctions $T_j^{(r)}$ do not help determine the inverse of $F^{(r)}$.

More generally, if we use sparse unsplit input encodings that are affine mappings on m_A bits, then the complexity of extracting the secret key in the proposed implementation is

$$O\left(\min\left\{{m_A}^{12} \cdot 2^{14}, 2m_A \cdot \log m_A \cdot 2^{m_A/2}\right\}\right)$$

for $m_A = 128, 256, 384, \cdots$. Table 3.1 presents the security level and storage requirements of the proposed implementation for $m_A = 128, 256, 384$. The attack complexity can be up to 2^{110} and 2^{117} when $m_A = 256$ and 384, respectively, which is quite close to the original 128-bit security level. This shows that sparse unsplit input encodings that have a multiple input size of the cipher's block length may be a useful way of designing a secure white-box implementation.

m_A	Security $\min\left\{{m_A}^{12}\cdot 2^{14}, 2m_A\cdot\log m_A\cdot 2^{m_A/2}\right\}$	Storage $\frac{m_A}{8}\cdot m_A\cdot 2^{16}$ bits
128	$2m_A\cdot\log m_A\cdot 2^{m_A/2}=2^{75}$	16 MB × (# of rounds)
256	${m_A}^{12}\cdot 2^{14}=2^{110}$	64 MB × (# of rounds)
384	${m_A}^{12}\cdot 2^{14}=2^{117}$	144 MB × (# of rounds)

Table 3.1: The security and storage of the proposed white-box AES implementation for $m_A = 128, 256, 384$

Chapter 4

New Lattice Basis Reduction Algorithm

CHAPTER 4. NEW LATTICE BASIS REDUCTION ALGORITHM

The goal of lattice basis reduction is to find a good basis from a given lattice basis. The meaning of "good" basis varies with the use of the basis, but it usually means short and close to orthogonal. To obtain a new lattice basis reduction algorithm, we need to make the purpose of it clear. Our purpose of lattice basis reduction is to get cryptanalysis of lattice based cryptosystems and analyze them using the reduction.

The best known lattice basis reduction algorithms are LLL algorithm and BKZ algorithm. LLL algorithm is polynomial time algorithm and well-analyzed, but the quality of output of the algorithm is not high and hence is not enough to be used widely as an analytic tools of lattice based cryptography. BKZ algorithm provides outputs of high quality and various estimation results with various block size and dimension. However, The estimation results are from several experiments or many assumptions and hence does not give theoretic bounds. Therefore, we try to get a new lattice basis reduction algorithm such that is well-analyzed and provides various outputs of high-quality according to the conditions.

To explain our algorithm, we first define some modular operations for lattice.

Definition 4.0.1. Let $v, b, \in \mathbb{R}^n$ be vectors. Then $v \bmod b$ is a vector in $\mathbb{R}^n$ such that $v \bmod b = v - cb$ for some $c \in \mathbb{Z}$ and satisfying $\|v \bmod b\| \leq \|v - xb\|$ for all $x \in \mathbb{Z}$.

More generally, let L be a lattice in $\mathbb{R}^n$. Then $v \bmod L$ is a vector in $\mathbb{R}^n$ such that $v \bmod L = v - w$ for some $w \in L$ and satisfying $\|v \bmod L\| \leq \|v - z\|$ for all $z \in \text{Ł}$.

Here, we also introduce the definition of the Voronoi cell and covering radius.

Definition 4.0.2. Let L be a lattice and $p \in L$ be a lattice point. Voronoi cell $\mathcal{V}(p)$ is the set in span(L) satisfying $\mathcal{V}(p) = \{x \mid \|x - p\| \leq \|x - y\|$ for all $y \in L\}$.

Definition 4.0.3. Let L be a lattice. The covering radius $\rho(L)$ is the maximum distance $dist(x, L)$ where x ranges over the span(L).

$$\rho(L) = \max_{x \in span(L)} \{dist(x, L)\}.$$

By the definition of $\bmod L$ operation, $v \bmod L$ is a vector in the Voronoi cell in L and $\bmod L$ operation transfer v to a point in Voronoi cell in L. Also, by the definition of the covering radius, The maximum length of vectors in the Voronoi cell centered 0 is the covering radius of L. Hence, For any v, $\|v \bmod L\| \leq \rho(L)$.

Remark 4.0.1. There can be more than 1 vector satisfying conditions in the definition. For clear definition, if more than 1 vector satisfy the condition, we choose $v \bmod L$ as follows.

For a vector $v \in \mathbb{R}^n$, let i_v be the coordinate such that i-th coordinate values of v are negative for all $i < i_v$ and i_v-th coordinates value of v is not negative. Also, let j_v be the largest coordinate of v such that i-th coordinate values of v are not negative for all $i_v \leq i \leq j_v$. Then, we choose $v \bmod L$ such that $i_{v \bmod L}$ is minimal. If several candidates have same the minimal i_v, then we choose the vector with maximal j_v from the candidates. Then by the symmetric property of the Voronoi cell of the lattice, $v \bmod L$ is uniquely determined.

4.1 Nearest Plane Algorithm

Let L be a full rank lattice given by an (ordered) basis $\{b_1, \cdots, b_n\}$ and let $\{b_1^*, \cdots, b_n^*\}$ be the corresponding Gram–Schmidt basis. Let $w \in \mathbb{R}^n$. Babai presented a method to inductively find a lattice vector close to w [Bab85]. Note that the vector $v \in L$ output by Babai's method is not guaranteed to be such that $\|w - v\|$ is minimal.

We now describe the method with Figure 4.1. For the induction, we let $L = L_n, w = w_n$. Define $U_{n-1} = span\{b_1, \cdots, b_{n-1}\}$ and let $L_{n-1} = L_n \cap U_{n-1}$ be the sublattice spanned by $\{b_1, \cdots, b_{n-1}\}$. The idea of the nearest plane method is to find a vector $y_n \in L_n$ such that the distance from w_n to the plane $U_{n-1} + y_n$ is minimal. One can easily find that $y_n = \lfloor c_n \rceil b_n^*$ satisfies the condition when $w_n = \sum_{i=1}^n c_n b_n^*$. Then one can sets w_n' to be the orthogonal projection of w onto the plane $U_{n-1} + y_n$. Let $w_{n-1} = w_n' - y_n \in U_{n-1}$. Inductively, One can do this process in the lower dimensional lattice, *i.e.*, one can get $y_i \in L$ for target vector w_i and lattice L_i. The solution to the original instance of the CVP is $v = \sum_{i=1}^n y_i$.

The following two theorems give result about quality of the output of the algorithm.

Theorem 4.1.1. *Let $\{b_1, \cdots, b_n\}$ be LLL-reduced (with respect to the Euclidean norm, and with factor $\delta = \frac{3}{4}$). If v is the output of Babai's nearest plane algorithm on input w then*

$$\|w - v\|^2 \le \frac{2^n - 1}{4} \|b_n^*\|^2.$$

Proof. Since $v = \sum_{i=1}^n y_i$ and $\|w_i' - w_i\| \le \frac{1}{2}\|b_i^*\|$,

$$\begin{aligned} \|w - v\|^2 &= \|w - \sum_{i=1}^n y_i\|^2 = \|w_n - \sum_{i=1}^n (w_i' - w_{i-1})\|^2 \\ &= \|\sum_{i=1}^n w_i - w_i'\|^2 \le \sum_{i=1}^n \frac{1}{4}\|b_i^*\|^2 \end{aligned}$$

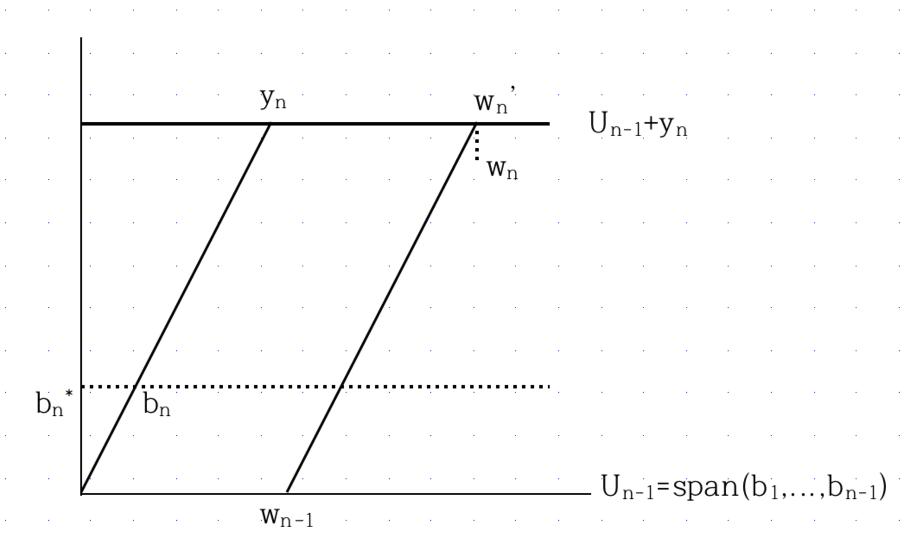

Figure 4.1: Babai's Nearest Plane Algorithm

Since $\{b_1, \cdots, b_n\}$ is LLL-reduced, $\|b_i^*\|^2 \leq 2\|b_{i+1}^*\|^2$, and hence we obtain $\|w-v\|^2 \leq \frac{2^n-1}{4}\|b_n^*\|^2$.

□

Theorem 4.1.2. *If the $\{b_1, \cdots, b_n\}$ is LLL-reduced (with respect to the Euclidean norm, and with factor $\delta = \frac{3}{4}$), then the output of Babai's nearest plane algorithm on input $w \in \mathbb{R}^n$ is a vector v such that*

$$\|v - w\| < 2^{n/2}\|u - w\|$$

for all $u \in L$.

Proof. We prove this result by induction.

For n=1, v is the closest vector and so the result holds.

Let $n \geq 2$ and $u \in L$ be the closest vector to w. We can consider two cases.

If $u \in U_{n-1} + y_n$, then $\|u-w\|^2 = \|u-w_n'\|^2 + \|w_n' - w_n\|^2$ and so u is also the closest vector to w_n'. Also, $u - y_n$ is the closest vector to $w_{n-1} \in U_{n-1}$. Since $\sum_{i=1}^{n-1} y_i$ is the output of the nearest plane algorithm on w_{n-1}, by the induction hypothesis

$$\|\sum_{i=1}^{n-1} y_i - w_{n-1}\| < 2^{(n-1)/2}\|u - y_n - w_{n-1}\|.$$

Therefore,

$$\begin{aligned}\|v-w\|^2 &= \|\sum_{i=1}^{n-1} y_i + (y_n - w_n') + w_n' - w_n\|^2 = \|\sum_{i=1}^{n-1} y_i - w_{n-1}\|^2 + \|w_n' - w_n\|^2 \\ &< 2^{n-1}\|u - w_n'\|^2 + \|w_n' - w_n\|^2 \\ &< 2^n\|u-w\|^2.\end{aligned}$$

Now, Consider the case that $u \notin U_{n-1}+y_n$. Then we have $\|u-w\| \geq \frac{1}{2}\|b_n^*\|$. By Theorem 4.1.2,

$$\|v-w\| \leq \frac{\sqrt{2^n-1}}{2}\|b_n^*\|^2 < 2^{n/2}\|u-w\|.$$

This completes the proof. □

It is known that Babai's nearest plane algorithm provides better output than Babai's rounding technique. Actually, $w-v$ of Babai's rounding technique corresponds to a point in the parallelepiped centered at origin by $b_1, \cdot, b_n$. However, $w-v$ of Babai's nearest plane algorithm corresponds to a point in the parallelepiped centered at origin by $b_1^*, \cdot, b_n^*$, *i.e.,* the product of $(-\frac{1}{2}, \frac{1}{2}]b_i^*$'s. Since the maximum length of vectors in the parallelepiped by $b_1^*, \cdot, b_n^*$ is shorter than that in the parallelepiped by $b_1, \cdot, b_n$, Babai's nearest plane algorithm provides better output than Babai's rounding technique. Note that $w-v$ in exact algorithm for CVP correspond to a point in Voronoi cell.

To obtain better output, the shape of region that $(w-v)$'s correspond to is close to the Voronoi cell $\mathcal{V}(O)$ in L. We can consider β-dimensional projection instead of 1-dimensional projection in Babai's nearest plane algorithm. In other words, To find a vector y in β-dimensional orthogonal projection space

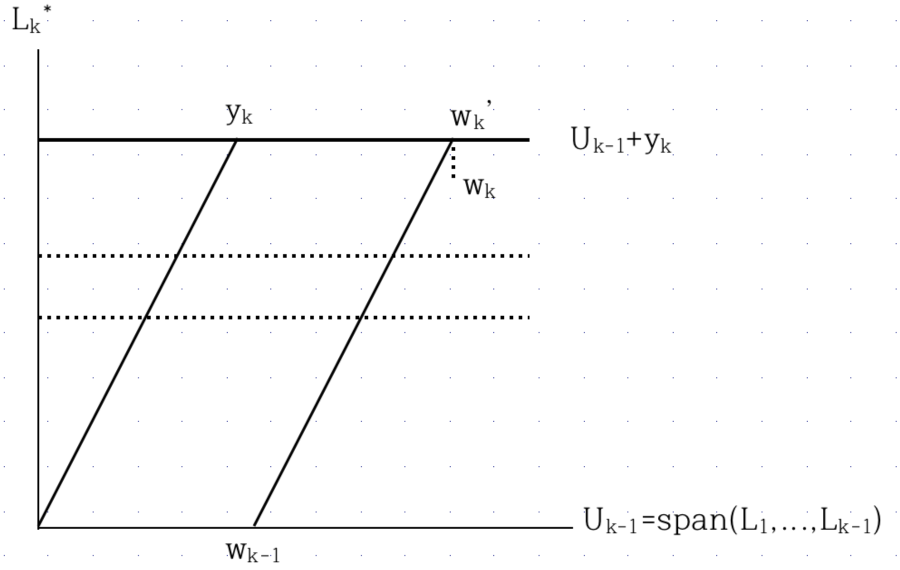

Figure 4.2: β-block Nearest Plane Algorithm

to U such that distance between w and $U + y$ is minimal.

To explain this, we give some notations for sublattice. Let $n = \beta k$ and $L_i = L(b_{(i-1)\beta+1}, \cdots, b_{i\beta})$ for $i = 1, \cdots, k$. For convenience, we let $L = L(L_1, \cdots, L_k) = L(b_1, \cdots, b_n)$. We define the map π_i as the orthogonal projection map onto span$(L_1, \cdots, L_{i-1})$. In particular, we define $L_i^* = \pi_i(L_i)$. It will play a role as b_i^* of Gram-Schmidt algorithm. Lastly, we define $d_i = \|w_i' - w_i\|$ in the nearest plane algorithm.

We now describe the new method with Figure 4.2. For the induction, we let $w = w_k$ and define $U_{k-1} = span\{L_1, \cdots, L_{k-1}\}$. The basic idea of the new algorithm is similar to Babai's nearest plane method. It is to find a vector $y_k \in L_k$ such that the distance from w_k to the plane $U_{k-1} + y_k$ is minimal. Then one can sets w_k' to be the orthogonal projection of w onto the plane $U_{k-1} + y_k$. Let $w_{k-1} = w_k' - y_k \in U_{k-1}$. Inductively, One can do this process in the lower dimensional lattice, *i.e.,* one can get $y_i \in L_i$ for target vector w_i and lattice L_i. The solution to the original instance of the CVP is $v = \sum_{i=1}^{k} y_i$.

For each $i = 1, \cdots, k$, w_i' is the closest vector of w_i in L_i^* and hence $w_i - w_i'$ corresponds to a point in the Voronoi cell centered at 0 in L_i^*. Therefore, $w - v$

in our β-block nearest plane algorithm corresponds to a point in the product of Voronoi cell in L_i^*. (Note that for the $\beta = 1$, $(-\frac{1}{2}, \frac{1}{2}]b_i^*$ is the Voronoi cell in $L(b_i^*)$.) Also, one can easily show that

$$\|w - v\|^2 = \sum_{i=1}^{k} \|w_i - w_i'\|^2 = \sum_{i=1}^{k} d_i^2 \leq \rho(L_i^*)^2.$$

4.2 Blockwise LLL Algorithm

If we use β-block nearest plane algorithm instead of original nearest plane algorithm, we can obtain more close vector v to target vector w than the output of the Babai's nearest plane algorithm. That is because the vectors in k product of β-dimensional Voronoi cells have smaller maximum length than the vectors in n product of 1-dimensional Voronoi cells, that is the product of $(-\frac{1}{2}, \frac{1}{2}]b_i^*$'s.

In the nearest plane algorithm, if we use LLL reduced basis for L with factor $\delta = \frac{3}{4}$, one can obtain a vector v such that $\|v - w\| < 2^{n/2}\|u - w\|$ for all $u \in L$. Then for the case of the β-block nearest plane algorithm, how can we obtain such results? In other words, which basis can we use with β-block nearest plane algorithm to get better factor of quality than $2^{n/2}$? For this, we propose a new lattice basis reduction algorithm which can reduce basis by block.

In the LLL algorithm, LLL reduced basis satisfies two conditions: size reduced and Lovasz condition. These conditions are related to the length of basis vectors. However, in the case of block reduction, it is difficult to compare size of blocks by the lengths of basis vectors. We need some measure for the block reduction instead of the length of the basis vectors.

One of natural candidate of measure is the covering radius of sublattices. In fact, $\frac{1}{2}\|b_i^*\|$ is the covering radius of $L(b_i^*)$. Also, size reduction in LLL algorithm can be regarded as a transform to $(-\frac{1}{2}, \frac{1}{2}]b_i^*$, Voronoi cell in 1-dimensional lattice. We define β-LLL reduced basis using the concept of Voronoi cell and covering radius of lattice. Recall the definition of $\bmod L$ in Definition 4.0.1 - $v \bmod L$ is a vector in $\mathbb{R}^n$ such that $v \bmod L = v - w$ for some $w \in L$ and satisfying $\|v \bmod L\| \leq \|v - z\|$ for all $z \in \text{Ł}$.

Definition 4.2.1. We call $b_1, \cdot, b_n$ is β-LLL reduced basis if it satisfies the following conditions.

- (β-size reduced) For all basis of L_i, their L_j^*-component is not larger than$\rho(L_j^*)$ when $1 \leq j < i \leq k$. *i.e.,* if b is a basis of L_i and $b = \sum_{j=1}^{i} b'_j$

where $b'_j \in$ span (L^*_j), then b'_j is in $\mathcal{V}(O)$ in L^*_j.

- (β-Lovasz condition) For $2 \le i \le k$,

$$\rho(L^*_i)^2 \ge \frac{1}{2}\rho(L^*_{i-1})^2.$$

Now we present β-block LLL algorithm which output β-LLL reduced basis for given basis as shown in Algorithm 2.

Now we prove the theorems about the quality of the output of the algorithm.

Theorem 4.2.1. *Let $\{b_1, \cdots, b_n\}$ be β-LLL-reduced where $n = \beta k$. If v is the output of β-nearest plane algorithm on input w then*

$$\|w - v\|^2 \le (2^k - 1)\|\rho(L^*_k)\|^2.$$

Proof. Since $v = \sum_{i=1}^k y_i$ and $\|w'_i - w_i\| \le \|\rho(L^*_i)\|$,

$$\begin{aligned} \|w - v\|^2 &= \|w - \sum_{i=1}^k y_i\|^2 = \|w_k - \sum_{i=1}^k (w'_i - w_{i-1})\|^2 \\ &= \|\sum_{i=1}^k w_i - w'_i\|^2 \le \sum_{i=1}^n \|\rho(L^*_i)\|^2 \end{aligned}$$

Since $\{b_1, \cdots, b_n\}$ is β-LLL reduced, $\|\rho(L^*_i)\|^2 \le 2\|\rho(L^*_{i+1})\|^2$, and hence we obtain $\|w - v\|^2 \le (2^k - 1)\|\rho(L^*_k)\|^2$.

□

Theorem 4.2.2. *Let $\{b_1, \cdots, b_n\}$ is β-LLL reduced where $n = \beta k$. If there is a constant c such that $c\lambda_1(L^*_i) \ge \lambda_\beta(L^*_i)$ for all $i = 1, \cdots, k$, then the output of β-nearest plane algorithm on input $w \in \mathbb{R}^n$ is a vector v such that*

$$\|v - w\| < c\sqrt{\beta}2^{k/2}\|u - w\|$$

for all $u \in L$.

Proof. We prove this result by induction.

For k=1, v is the closest vector and so the result holds.

Let $k \geq 2$ and $u \in L$ be the closest vector to w. We can consider two cases.

If $u \in U_{k-1} + y_k$, then $\|u - w\|^2 = \|u - w'_k\|^2 + \|w'_k - w_k\|^2$ and so u is also the closest vector to w'_k. Also, $u - y_k$ is the closest vector to $w_{k-1} \in U_{k-1}$. Since $\sum_{i=1}^{k-1} y_i$ is the output of the nearest plane algorithm on w_{k-1}, by the induction hypothesis

$$\|\sum_{i=1}^{k-1} y_i - w_{k-1}\| < c\sqrt{\beta}2^{(k-1)/2}\|u - y_k - w_{k-1}\|.$$

Therefore,

$$\begin{aligned} \|v - w\|^2 &= \|\sum_{i=1}^{k-1} y_i + (y_k - w'_k) + w'_k - w_k\|^2 = \|\sum_{i=1}^{k-1} y_i - w_{k-1}\|^2 + \|w'_k - w_k\|^2 \\ &< c\sqrt{\beta}2^{k-1}\|u - w'_k\|^2 + \|w'_k - w_k\|^2 \\ &< c\sqrt{\beta}2^k\|u - w\|^2. \end{aligned}$$

Now, Consider the case that $u \notin U_{k-1} + y_k$. Let u' be a projection of w to the plane containing u. Then $u' - w$ is not in the Voronoi cell in L_k^* and we have

$$\|u - w\| \geq \frac{1}{2}\lambda_1(L_k^*) \geq \frac{1}{2c}\lambda_\beta(L_k^*) \geq \frac{1}{c\sqrt{\beta}}\rho(L_k^*).$$

By Theorem 4.2.1,

$$\|v - w\| \leq \sqrt{(2^k - 1)}\|\rho(L_k^*) < c\sqrt{\beta}2^{k2}\|u - w\|.$$

This completes the proof. □

One can apply the analysis about the number of steps in LLL algorithm to our algorithm. This implies the dominant complexity of this algorithm is 2^β. Compare to the result in Theorem 4.1.2, Our algorithm provides trade-off between reduction complexity and the output quality. Therefore, One can get various analytic results for lattice based cryptosystem using our algorithm with various β.

Algorithm 2 β-block LLL Algorithm

```
Input: Given basis $b_1, \cdots, b_n$.
Output: $\beta$-LLL reduced basis $b_1, \cdots, b_n$.
Let $L_i = L(b_{(i-1)\beta+1}, \cdots, b_{i\beta})$.
$i \leftarrow 2$
  while $i \leq k$ do
      for $j = i - 1$ to 1 do
          $\beta$-size reduce $L_i$ w.r.t. $L_j$
      end for
      if $2\rho(L_i^*)^2 \geq \rho(L_{i-1}^*)^2$ then
          $i = i + 1$
      else
          Swap $L_i$ with $L_{i-1}$ and $\beta$-size reduce for changed part.
          $i = \max\{2, i - 1\}$
      end if
  end while
```

Chapter 5

Conclusions

In this paper, we proposed a general analytic toolbox for white-box implementation, which can efficiently extract the secret encodings used to obfuscate in the implementation when its design follows CEJO framework. With our toolbox, it is very easy to evaluate the asymptotic complexity of any white-box implementation of SLT ciphers which follows CEJO framework, and all previous designs belong to this model. Hence, our toolbox could be used to measure the security of white-box implementations.

Another advantage of our toolbox is that we can remove insecure designs at an early stage, and concentrate on more plausible approaches. We showed that the input size of the encodings is the most important factor in the security of a white-box implementation. In this sense, we presented a white-box implementation that uses sparse unsplit input encodings with an input size that is a multiple of the block length. This not only produces high level of security, but also has reasonable storage requirements.

On the other hand, we proposed an algorithm, block LLL algorithm, for lattice basis reduction which uses block reduction. This provides some trade-off of reduction time and quality of the output. Our algorithm can get various outputs of higher quality than of LLL reduction algorithm according to the block size β as BKZ reduction algorithm. Also, our algorithm is more easy to analyze than BKZ reduction algorithm because one can use the method of analysis in LLL reduction algorithm. This can gives a guideline for the parameter setting of lattice based cryptography.

Bibliography

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 284–293, New York, NY, USA, 1997. ACM.

[Bab85] László Babai. On lovÁsz' lattice reduction and the nearest lattice point problem (shortened version). In *Proceedings of the 2Nd Symposium of Theoretical Aspects of Computer Science*, STACS '85, pages 13–20, London, UK, UK, 1985. Springer-Verlag.

[BBK14] Alex Biryukov, Charles Bouillaguet, and Dmitry Khovratovich. Cryptographic Schemes Based on the ASASA Structure: Black-box, White-box, and Public-key. *IACR Cryptology ePrint Archive*, 2014:474, 2014. To appear in ASIACRYPT 2014.

[BCBP03] Alex Biryukov, Christophe De Canniére, An Braeken, and Bart Preneel. A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 33–50. Springer, 2003.

[BGEC05] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a White Box AES Implementation. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography - SAC 2004*, volume 3357 of *LNCS*, pages 227–240. Springer, 2005.

[BS01] Alex Biryukov and Adi Shamir. Structural Cryptanalysis of SASAS. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 395–405. Springer, 2001.

[CEJO03] Stanley Chow, Philip Eisen, Harold Johnson, and Paul C. Van Oorschot. White-Box Cryptography and an AES Implementation. In Kaisa Nyberg and Howard Heys, editors, *Selected Areas in Cryptography - SAC 2002*, volume 2595 of *LNCS*, pages 250–270. Springer, 2003.

[CEJvO03] Stanley Chow, Phil Eisen, Harold Johnson, and Paul C. van Oorschot. A White-Box DES Implementation for DRM Applications. In Joan Feigenbaum, editor, *Digital Rights Management - DRM 2002*, volume 2696 of *LNCS*, pages 1–15. Springer, 2003.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers. In *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 446–464. Springer, 2012.

[DDKS12] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial search Problems. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *LNCS*, pages 719–740. Springer, 2012.

[DLPR14] Cécile Delerablée, Tancrède Lepoint, Pascal Paillier, and Matthieu Rivain. White-Box Security Notions for Symmetric Encryption Schemes. In Tanja Lange, Kristin Lauter, and Petr Lisoněk, editors, *Selected Areas in Cryptography - SAC 2013*, LNCS, pages 247–264. Springer, 2014.

[Gal12] Steven D. Galbraith. *Mathematics of public key cryptography.* Cambridge University Press, Cambridge, New York, 2012.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pages 112–131, London, UK, UK, 1997. Springer-Verlag.

[GH11] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 129–148. Springer, 2011.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In CetinK. Koc, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, volume 2162 of *LNCS*, pages 251–261. Springer, 2001.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, chapter NTRU: A ring-based public key cryptosystem, pages 267–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[Kar11] Mohamed Karroumi. Protecting White-Box AES with Dual Ciphers. In Kyung-Hyune Rhee and DaeHun Nyang, editors, *Information Security and Cryptology - ICISC 2010*, volume 6829 of *LNCS*, pages 278–291. Springer, 2011.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO 1999*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.

[Koc96] PaulC. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO 1996*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LRM$^+$14] Tancrède Lepoint, Matthieu Rivain, Yoni De Mulder, Peter Roelse, and Bart Preneel. Two Attacks on a White-Box AES Implementation. In Tanja Lange, Kristin Lauter, and Petr Lisoněk, editors, *Selected Areas in Cryptography - SAC 2013*, LNCS, pages 265–285. Springer, 2014.

[MGH09] Wil Michiels, Paul Gorissen, and Henk D. L. Hollmann. Cryptanalysis of a Generic Class of White-Box Implementations. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography - SAC 2008*, volume 5381 of *LNCS*, pages 414–428. Springer, 2009.

[MRP13] Yoni De Mulder, Peter Roelse, and Bart Preneel. Cryptanalysis of the Xiao - Lai White-Box AES Implementation. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography - SAC 2012*, volume 7707 of *LNCS*, pages 34–49. Springer, 2013.

[Nov02] Roman Novak. Spa-based adaptive chosen-ciphertext attack on rsa implementation. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography - PKC 2002*, volume 2274 of *LNCS*, pages 252–262. Springer, 2002.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards. In Isabelle Attali and Thomas Jensen, editors, *Smart Card Programming and Security*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.

[Sch87] C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(2-3):201–224, August 1987.

[SS79] Richard Schroeppel and Adi Shamir. A TcS2 = 0 (2n) time/space tradeoff for certain NP-complete problems. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 328–336, Oct 1979.

[SV14] Nigel P Smart and Frederik R G Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.

[SWP09] Amitabh Saxena, Brecht Wyseur, and Bart Preneel. Towards Security Notions for White-Box Cryptography. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio A. Ardagna, editors, *Information Security*, volume 5735 of *LNCS*, pages 49–58. Springer, 2009.

[WMGP07] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings. In Carlisle Adams, Ali Miri, and Michael Wiener, editors, *Selected Areas in Cryptography - SAC 2007*, volume 4876 of *LNCS*, pages 264–277. Springer, 2007.

[Wys09] Brecht Wyseur. *White-Box Cruptography.* PhD thesis, Katholieke Universiteit Leuven, 2009.

[XL09] Yaying Xiao and Xuejia Lai. A Secure Implementation of White-box AES. In *Computer Science and its Applications - CSA 2009*, pages 1–6. IEEE, 2009.

국문초록

암호학에서 암호 분석 도구의 존재는 암호 시스템 설계 상의 안전성을 측정할 수 있는 척도가 된다는 면에서 매우 중요하다. 본 논문에서는 특별히 화이트 박스 암호와 격자 기반 암호에 대한 분석 도구를 제시한다.

Chow 등에 의해 제안된 화이트 박스 암호 기술는 공격자가 암호화 알고리즘의 구현 과정에 완전히 접근할 수 있고 그 실행 플랫폼을 완전히 제어할 수 있다고 하더라도 소프트웨어 구현의 비밀키를 보호할 수 있는 난독화 기술이다. 화이트 박스 암호 기술은 공격자의 힘이 커지고 다양해지는 현대 사회에 실제적으로 꼭 필요한 암호 기술이지만 그 중요성에 비해 발전 속도는 그리 빠르지 않다. 실제로 화이트 박스 구현이 제안되면 그 때마다 낮은 공격량으로 그것을 공격하는 공격 방법이 바로 제안되는 일들이 반복되어 왔다. 그 이유는 보통 암호 시스템을 설계할 때 기존의 알려진 공격들에 안전하도록 설계를 하는데 화이트 박스 암호에서 제안되는 대부분의 공격 기법들은 일반적인 공격 기법이 아닌 특정 구현에만 적용되는 방법들이기 때문에 설계하는 암호의 안전성을 예측하기 어렵기 때문이다. 본 논문에서는 Chow 스타일의 테이블 검색을 이용한 화이트 박스 암호에 대한 일반적인 공격 방법을 제시한다. 제안되는 공격 기법은 화이트 박스 암호의 안전성을 예측하는 분석도구로 사용될 수 있을 것이다.

격자 기반 암호는 현대 암호학에서 가장 흥미있고 각광받고 있는 암호로서 양자 컴퓨팅 환경의 발전과 더불어 그 중요성이 더욱더 커지고 있다. 대부분의 격자를 기반으로 한 암호학적 난제들은 특정한 형태의 짧은 벡터 문제(Shortest Vector Problem)나 가까운 벡터 문제(Closest Vector Problem)로 환원이 되는데 이 문제들의 어려움은 격자가 주어졌을 때 길이가 짧은 기저를 찾는 문제를 푸는 어려움과 관련이 있다. 따라서 좋은 격자 기저 축소 알고리즘은 격자 기반 암호의 분석도구로서 역할을 할 수 있다. 본 논문에서는 기존의 LLL 알고리즘을 개선한 블록 단위의 격자 기저 축소 알고리즘을 제안한다. 제안되는 알고리즘은 기저 축소 시간과 축소된 기저의 품질 간에 트레이드오프를 제공하며 따라서 격자 기반 암호의 파라미터 설정에 가이드라인으로서의 역할을 할 수 있을 것이라고 예상할 수 있다.

주요어휘: 화이트박스 암호, SPN 구조, 격자 암호, 격자 기저 축소 알고리즘, 블록 LLL 알고리즘

학번: 2007-20276