Analysis of Algorithms Related to Multivariate Public Key Cryptosystems and the Transport Layer Security Protocol

（多変数公開鍵暗号及び
トランスポート層セキュリティプロトコル
に関するアルゴリズムの解析）

2023 年

黒川　貴司

Analysis of Algorithms Related to Multivariate Public Key Cryptosystems and the Transport Layer Security Protocol

Takashi Kurokawa

Division of
Electronic and Computer Systems Engineering,
Department of
Electrical, Electronic and Computer Systems Engineering,
Graduate School of
Engineering and Resource Science,
Akita University

2023

Contents

Abstract

In recently, the development in quantum computing has been progressing at a steady pace. Furthermore, there is a growing concern that Rivest-Shamir-Adleman and elliptic curve cryptosystems, which have been widely used for a long time, may be vulnerable to compromise in the future because of Shor's algorithm. Although it is difficult to predict when a large-scale quantum computer with error correction will become widely available, it will take significant time to deploy cryptographic technology in society, and if it is not adequately secure, various adverse effects will occur.

The post-quantum cryptography standardization project is currently led by the National Institute of Standards and Technology. Several public-key cryptosystems were proposed for the project. Of these, there are public-key cryptosystems that rely on the hardness of solving a system of multivariate quadratic (MQ) polynomial equations over finite fields, known as multivariate public-key cryptosystems (MPKCs). By the end of 2022, the project has moved to a fourth round of the evaluation process, and no MPKCs remain. However, as the project is currently soliciting applications for digital signature schemes featuring a short signature length and fast signature verification, it is expected that MPKCs with such characteristics will reemerge. To assess the security of MPKCs, it is crucial to evaluate the difficulty of solving a system of MQ polynomial equations, which is referred to as the MQ problem.

There are several algorithms for solving MQ problems, including the Gröbner basis algorithm and its variant, the F4 algorithm. In the Gröbner basis algorithm, two polynomials are reduced by canceling their leading monomials. The two polynomials and the monomials utilized to cancel their leading monomials are referred to as a critical pair. The F4 algorithm constructs a matrix known as the Macaulay matrix from the critical pairs and applies Gaussian elimination to reduce multiple polynomials simultaneously. However, it has the disadvantage that the computational cost increases as the matrix size increases. To overcome this disadvantage, ongoing research aims to divide the set of critical pairs into subsets; perform Gaussian elimination of the Macaulay matrix for each subset; and ignore the remaining subsets when a zero polynomial is generated.

We propose several methods for dividing the set of critical pairs and demonstrate through computer experiments that they can reduce the processing time compared with the F4 algorithm, which is implemented using the OpenF4 library. The method of generating the sample MQ problems utilized in the experiments is the same as that used in the Fukuoka MQ Challenge. The experiments also confirmed that the failure rate is minimal, even when the remaining critical pairs are omitted during Gaussian elimination of the Macaulay matrix. Additionally, the experiments indicated that the minimum number of critical pairs

required to lower the Macaulay matrix's rank is almost constant. However, further study is needed to theoretically support such a property.

The Secure Sockets Layer/Transport Layer Security (SSL/TLS) protocol, which is a cryptographic communication protocol, can be widely used on the internet services, such as online shopping. It includes not only an encryption function to provide confidentiality but also a function to authenticate the entities. If these functions are tampered with, secure communication services cannot be utilized. In recent years, a variety of attack methods against SSL/TLS, including Compression Ratio Info-leak Made Easy (CRIME), LUCKY THIRTEEN, Padding Oracle On Downgraded Legacy Encryption (POODLE), and Browser Exploit Against SSL/TLS (BEAST) have been proposed, and instances of their practical implementation have been made public. Specifically, the BEAST attack is an attack that can be applied to a particular block cipher mode of operation, known as the Cipher Block Chaining (CBC) mode in TLS 1.0. The CBC mode of TLS 1.0 has two vulnerabilities: the padding scheme and the method of selecting the initialization vector (IV). These vulnerabilities can be exploited if a security bug is contained in a web browser.

As a countermeasure against the BEAST attack, the "$1/n-1$ record splitting" patch was proposed, which is a method of dividing the transmitted data into the first byte and the remainder, as opposed to transmitting all the data at once. The patched CBC mode proved secure against chosen-plaintext attacks. In the CBC mode, certain information is appended when encrypting data. Specifically, the bit string *seq_num*, which serves as a counter, is appended, and the message authentication code is then applied. Because of such operations, the output of the IV becomes unpredictable when the first byte is processed. This is likely the reason why the "$1/n-1$ record splitting" patch is an effective countermeasure.

Chapter 1

Introduction and Background

1.1 Post-quantum cryptography and NIST PQC Standardization

In recent years, research and development of quantum computers have progressed rapidly. For example, noisy intermediate-scale quantum computers are already in practical use. Shor [65] demonstrated that solving both the integer factorization problem (IFP) and the discrete logarithm problem (DLP) is theoretically tractable in polynomial time. Rivest-Shamir-Adleman (RSA) and elliptic curve cryptosystems are widely used in applications, and their security depends on the IFP and DLP, respectively. If large-scale reliable quantum computers were made available, many existing information and communication systems would become insecure. Although it is unknown when such computers will be available, research and development of secure cryptosystems against both quantum and conventional computers are urgent issues, where such systems are referred to as post-quantum cryptography (PQC). Research, development, and standardization projects for PQC are ongoing against this background.

The PQC Standardization Process was started by the National Institute of Standards and Technology (NIST) in 2016. Several cryptosystems have been proposed for the NIST PQC project, including lattice-based cryptosystems and code-based cryptosystems. The multivariate public key cryptosystem (MPKC) is one of the cryptosystems proposed for the NIST PQC project.

Several MPKCs were also proposed for the NIST PQC project [51], e.g., the G*e*MSS [15], LUOV [12], MQDSS [16], and Rainbow [23] MPKCs. In the fourth round of the NIST PQC project, NIST issued a new request for proposals of digital signature schemes with short signatures and fast verification [54]. MPKCs are frequently more efficient than other public key cryptosystems, primarily digital signature schemes; researching the security of MPKCs is still essential.

1.2 MPKC

Research into MPKCs began in the 1980s. One of the most well-known public key cryptosystems based on multivariate polynomials over a finite field was proposed by Matsumoto and Imai [46]. Patarin [56] later demonstrated that the Matsumoto-Imai cryptosystem is insecure. At the same time, Patarin proposed the hidden field equations (HFE) public key cryptosystem by repairing their cryptosystem[57] and did the Oil and Vinegar scheme (OV) [55]. Note that there are several variations of the HFE and OV schemes, e.g., Kipnis, Patarin, and Goubin proposed the unbalanced OV scheme (UOV) [38]. In addition, Patarin et al. introduced the isomorphisms of polynomials problems and the morphism of polynomials problem [57, 58], and the security of MPKC depends on these problems. Attacks against the basic HFE were developed by Kipnis and Shamir [40], Courtois [18], and Faugère and Joux [30], and an attack against the OV scheme was developed by Kipnis and Shamir [39].

1.3 MQ problems

The security of an MPKC is highly dependent on the hardness of solving a system of multivariate quadratic (MQ) polynomial equations over finite fields because multivariate polynomial equations are transformed into MQ polynomial equations by increasing the number of variables and equations. Solving such systems is referred to as the MQ problem. It is also fundamental for breaking a cryptosystem that we translate its underlying algebraic structure into a system of multivariate polynomial equations. There are three well-known algebraic approaches to solving the MQ problem: the XL algorithm proposed by Courtois et al. [17] and the F4/F5 algorithm proposed by Faugère [27, 28].

1.3.1 The Fukuoka MQ Challenges

In addition to theoretical evaluations of the computational complexity of the algorithms, practical evaluations are also crucial in the research of cryptography, e.g., such as many efforts against the RSA challenge [2], the ECC challenge [1], and the Lattice challenge [3]. The Fukuoka MQ challenge project [71, 70] was started in 2015 to evaluate the security of the MQ problem. In the Fukuoka MQ challenge project, the MQ problems are classified into encryption and digital signature schemes. Each scheme in this project is then classified into three categories according to the number of quadratic equations (m), the number of variables (n), and a characteristic of the finite field. The encryption schemes are classified into types I to III, which correspond to the condition where $m = 2n$ over $\mathbb{F}_2$, $\mathbb{F}_{256}$, and $\mathbb{F}_{31}$, respectively. The digital signature schemes are classified into types IV to VI, which corresponds to the condition where $n \approx 1.5m$ over $\mathbb{F}_2$, $\mathbb{F}_{256}$, and $\mathbb{F}_{31}$, respectively. At the time of writing, all best records in the Fukuoka MQ challenge except type IV are set by variant algorithms of both the XL and the F4 algorithm. For example, we improved the F4-style algorithm and set new records of both type II and III, as described below, but a variant of the XL algorithm later updated the record of type III.

1.4 SSL 1.0 and TLS 1.0

SSL 3.0/TLS 1.0 was formerly one of the most widely used cryptographic protocols in the internet services. SSL 3.0 [31] was released by Netscape Communications in 1996 and then TLS 1.0 [19] was released by Internet Engineering Task Force (IETF) in 1999. They were then updated and TLS 1.1 [20], 1.2 [21], and 1.3 [60] were released in 2006, 2008, and 2018, respectively.

SSL 3.0/TLS 1.0 was deployed to perform almost all the popular network services securely; for example, online shopping and online banking. At the same time, many cryptographic attacks against SSL 3.0 and TLS 1.0 were found, e.g., Compression Ratio Info-leak Made Easy (CRIME) [61], Lucky Thirteen [5], Browser Exploit Against SSL/TLS (BEAST) [24], Padding Oracle On Downgraded Legacy Encryption (POODLE) [49] and RC4 biases attacks [4, 34].[1]

In SSL 3.0/TLS 1.0, many cryptographic primitives are employed, e.g., RSA [50], DH(E)[2] [22], AES [52], RC4 [64], CBC mode [25], and HMAC [53]. Herein, we will focus on the *CBC mode* in SSL 3.0/TLS 1.0, which can cause security issues in SSL 3.0/TLS 1.0.

According to the SSL Pulse data [68], TLS 1.0 was the most widely used protocol version through SSL/TLS[3], and the CBC mode was included in many cipher suites of them, at the time of writing the paper [42].[4]. Although a software patch was released for the CBC mode, it is unclear whether the patched CBC mode is secure against BEAST attacks or how secure it is.

1.4.1 Overview of the Record Layer

SSL 3.0 and TLS 1.0 consist mainly of the *Record*, *Handshaking*, *Change Cipher Spec*, and *Alert Protocol.* Herein, we explain only the record protocol. It divides the data passed by higher layers into information blocks. One of the specified information blocks is the *SSLCiphertext* or the *TLSCiphertext* as shown in Listing 1.1, 1.2. If the block cipher encryption is chosen by the *CipherSuite* in the SSL/TLS, the *GenericBlockCipher* is selected as the *fragment.*

Listing 1.1: SSLCiphertext Structure in SSL 3.0

```
struct {
    ContentType type;
    ProtocolVersion verions;
    uint16 length;
    select (CipherSpec.cipher_type) {
        case stream: GenericStreamCipher;
        case block: GenericBlockCipher;
    } fragment;
} SSLCiphertext;

block-ciphered struct {
    opaque Content[SSLCompressed.length];
    opaque MAC[CipherSpec.hash_size];
    uint8 padding[GenericBlockCipher.padding_length];
    uint8 padding_length;
} GenericBlockCipher;
```

[1]For the overview of these attacks, see [63].
[2]DHE denotes ephemeral Diffie-Hellman key exchange.
[3]As of December 2015, 98.8% of the sites surveyed by the SSL Pulse support TLS 1.0.
[4]As of December 2015, 0.3% of the sites surveyed by the SSL Pulse support only RC4 cipher suites.

Listing 1.2: TLSCiphertext Structure in TLS 1.0

```
struct {
    ContentType type;
    ProtocolVersion verions;
     uint16 length;
     select (CipherSpec.cipher_type) {
         case stream: GenericStreamCipher;
         case block: GenericBlockCipher;
    } fragment;
} TLSCiphertext;

block-ciphered struct {
    opaque Content[TLSCompressed.length];
    opaque MAC[CipherSpec.hash_size];
    uint8 padding[GenericBlockCipher.padding_length];
    uint8 padding_length;
} GenericBlockCipher;
```

The selected block cipher encrypts the *GenericBlockCipher* in combination with the CBC mode of operation. To force the total length of this block structure to be an integral multiple of the block size of the chosen block cipher, the *padding* is appended just after the *MAC*.

In SSL 3.0, each byte of the *padding* is indefinite, and the *padding_length* is less than the cipher block size or zero. In TLS 1.0, each byte of the *padding* is filled with the *padding_length* value, and the padding length is up to 255 bytes.

The *MAC* is generated using the computation in Listing 1.3, 1.4. Note that the *seq_num* denotes the sequence number of the record.

Listing 1.3: MAC in SSL 3.0

```
hash(MAC_write_secret + pad_2 +
     hash(MAC_write_secret + pad_1 + seq_num +
          SSLCompressed.type + SSLCompressed.length +
          SSLCompressed.fragment));
where "+" denotes concatenation.
```

Listing 1.4: MAC in TLS 1.0

```
HMAC_hash(MAC_write_secret, seq_num + TLSCompressed.type +
              TLSCompressed.version + TLSCompressed.length +
              TLSCompressed.fragment));
where "+" denotes concatenation.
```

1.4.2 The CBC Mode in SSL 3.0 and TLS 1.0

In the SSL/TLS, a plaintext is "tagged" before the encryption, and the tag is computed as the *MAC*, as described above. In other words, before encrypting a plaintext M, the tag t is firstly generated and secondly appended to M. Then the message

$$M' = M \| t$$

is encrypted using the CBC mode, as shown in Figure 1.1. Then, the ciphertext of the (tagged) message

$$M' = (M'[0], M'[1], \ldots, M'[m-1])$$

is represented as

$$\mathcal{F}_K(\texttt{IV} \oplus M'[0]), \ldots, \mathcal{F}_K(C[m-2] \oplus M'[m-1] \| \texttt{padding} \| \texttt{padding\_length}), \tag{1.1}$$

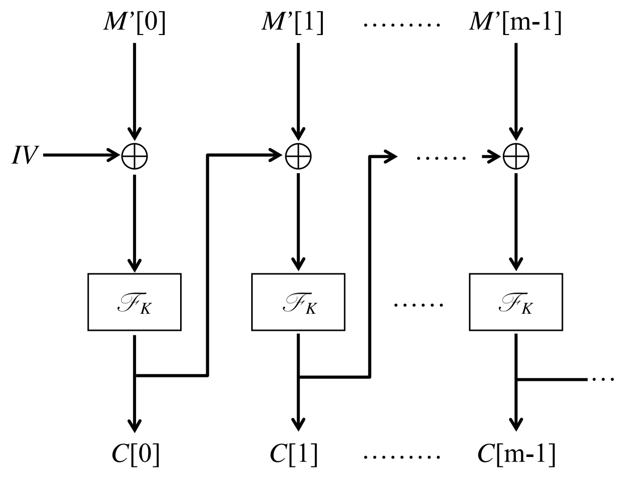


Figure 1.1: The CBC Mode in SSL 3.0 and TLS 1.0

where $\mathcal{F}_K : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a secure block cipher which can be modeled as the pseudorandom permutation (See Section 2.7), λ is the block length, `IV` is an initialization vector, `padding` is the padding of the GenericBlockCipher structure, `padding_length` is the length of `padding`,

$$C[i] = \begin{cases} \mathcal{F}_K(\texttt{IV} \oplus M'[0]) & \text{for } \ i = 0, \\ \mathcal{F}_K(C[i-1] \oplus M'[i]) & \text{for } \ 1 \le i \le m-2, \\ \mathcal{F}_K(C[m-2] \oplus M'[m-1] \| \texttt{padding} \| \texttt{padding\_length}) & \text{for } \ i = m-1. \end{cases}$$

The `IV` for the first record is generated from the *SecurityParameters* in the SSL/TLS, and the `IV` of the next record is set from a ciphertext block of a previous record.

The CBC mode in the SSL/TLS has two potential weaknesses: one is in the padding scheme, and the other is in handling the `IV` [48].

Padding: In the encryption of the form (1.1), which follows the Mac-then-Enc paradigm [33], the MAC is not applied to the padding. In other words, the padding is appended after the tag generation. When catching an error in this form, we can consider two cases: the error of the padding scheme and that of the MAC. If an adversary can distinguish these two cases, an attack known as the *padding oracle attack* [69] may work. For example, a timing analysis [14] enables the adversary to distinguish these two cases. Other side-channel information may also be exploited to attack the CBC mode. In fact, Möller et al. [49] found a practical attack against the CBC mode in SSL 3.0, named the POODLE attack (See Section 1.5). Because TLS and SSL employ a different padding schemes, this attack cannot be applied directly to the CBC mode in TLS. However, some implementations of TLS were affected by the POODLE attack [44].

Choice of `IV`: In SSL 3.0 and TLS 1.0, the `IV` after the first record is chosen from the last block of the ciphertext; therefore, an adversary who can eavesdrop on the ciphertexts knows the `IV` before the next plaintext is encrypted [62]. Thus, the `IV` is predictable from the adversary's point of view, so the CBC mode in SSL 3.0 and TLS 1.0 does not satisfy indistinguishability. This fact does not immediately imply that the adversary can recover the whole plaintext, and it would be expected that the time complexity of recovering the plaintext would be $O(2^{\lambda})$ for one block of ciphertexts. However, Duong and Rizzo demonstrated the BEAST attack [24], whose time complexity is $O(\lambda)$.

In TLS 1.1 [20] and TLS 1.2 [21], the subsequent `IV` is securely sent from one to the other by the following structure (See Listing 1.5).

Listing 1.5: GenericBlockCipher Structure in TLS 1.1 and TLS 1.2

```
block-ciphered struct {
      opaque IV[CipherSpec.block_length];
      opaque content[TLSCompressed.length];
      opaque MAC[CipherSpec.hash_size];
      uint8 padding[GenericBlockCipher.padding_length];
      uint8 padding_length;
} GenericBlockCipher;
```

1.5 Attacks against SSL 3.0 and TLS 1.0

In this section, we give outlines of the BEAST, POODLE, and RC4 bias attacks by comparison.

Web browsers and web application programming interfaces adopt the *Same-Origin Policy* (SOP) [9] to restrict interactions between different domains. Therefore, a security flaw in SOP involves substantial risk of data theft, such as secret information typed into legitimate websites and HTTP cookies [8] by an attacker.

In the attacks cited below, we implicitly assume that a target of an attacker knows a vulnerability of SOP in a target's web browser, and so on.

Let us suppose that the attacker can control the following actions of the target, as shown in Figure 1.2:

S1. To direct the target to a rogue web server that the attacker can control and access to it.

S2. To infect the target with a malicious code inserted at a web page on a rogue web server.

S3. To direct the target to a legitimate web server and access it.

S4. To make the target send private secret information to a legitimate web server.

The BEAST Attack

The BEAST attack[5] [24] is the chosen-plaintext attack against SSL 3.0 and TLS 1.0 and is mainly divided into two phases, as shown in Figure 1.3:

B1. Chosen Boundary Phase:

   (a) An attacker generates a string of a certain length and sends it to a target.

[5]BEAST stands for "Browser Exploit Against SSL/TLS".

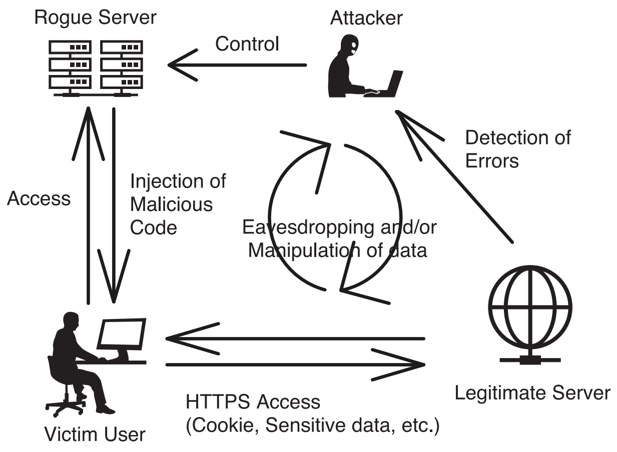

Rogue Server
Attacker
Control
Access
Injection of Malicious Code
Eavesdropping and/or Manipulation of data
Detection of Errors
Victim User
HTTPS Access (Cookie, Sensitive data, etc.)
Legitimate Server

Figure 1.2: Schematic View of Attack Scenarios

   (b) The target (unknowingly) prepends it to a plaintext to be encrypted, encrypts the resulting text in CBC mode and sends a ciphertext to a server.
   (c) The attacker knows the ciphertext.

B2. Blockwise Phase:

   (a) The attacker XORs a certain block of the plaintext, a certain block of the ciphertext, and the last block of ciphertext, and sends it to the target.
   (b) The target (unknowingly) encrypts the resulting text and sends the ciphertext to the server.
   (c) The attacker repeats the above procedures until a certain condition is satisfied.

We refer to such an attack as a BEAST type of attack. Note that, in the first phase, it is crucial to make a first unknown byte put as the last byte of a certain (but not first) block of plaintext. In the second phase, the attacker can learn the `IV` of the next record as the last block of the last ciphertext despite not knowing the first `IV`. Moreover, the attacker can learn the unknown byte $m[0]$ if he continues to encrypt forged blocks $C^*[0] \oplus C^*[n-1] \oplus (r||i)(i = 0, \ldots, 255)$ until the resulting encrypted block $C^*[n]$ equals to a former block $C^*[1]$, as shown in Figure 1.3.

At the second block of the first record, we have

$$\mathcal{F}_K^{-1}(C^*[1]) = C^*[0] \oplus P^*[1] = C^*[0] \oplus (r||m[0]),$$

and at the first block of the second record, thanks to the cancelation of $C^*[n-1]$, we have

$$\mathcal{F}_K^{-1}(C^*[n]) = C^*[n-1] \oplus (C^*[0] \oplus C^*[n-1] \oplus (r||i)) = C^*[0] \oplus (r||i).$$

Then $m[0] = i$ when the identity $C^*[n] = C^*[1]$ holds.

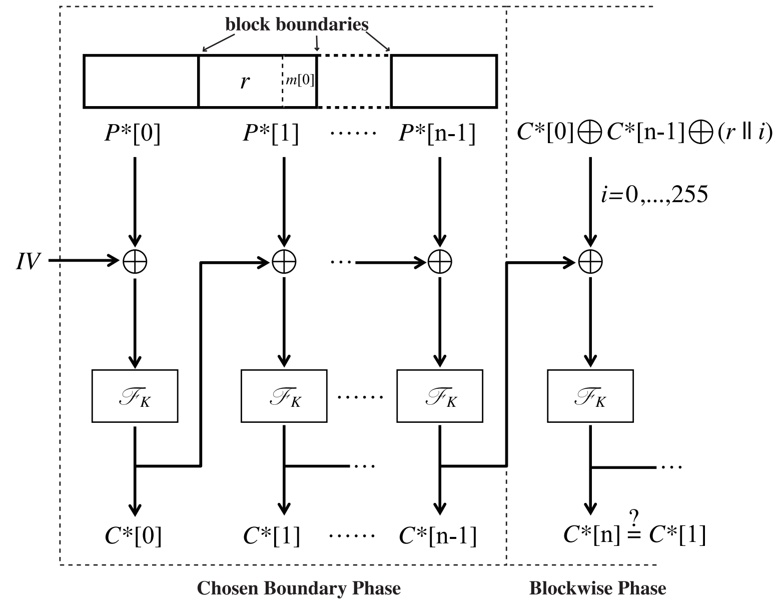


Figure 1.3: Schematic View of the BEAST Attack

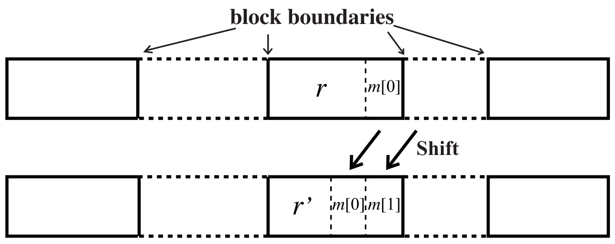


Figure 1.4: Sequential Alignment on the Chosen Boundary Phase in the BEAST Attack

The other unknown bytes of plaintext can be decrypted if the attacker adjusts a length of a prepending string in the manner mentioned above (Fgure 1.4).

To mount the BEAST attack, two underlying conditions are necessary. One is that there exists a vulnerability to bypass the SOP in the browser, and the other is the predictability of IV, which is the case of the CBC mode in SSL 3.0/TLS 1.0. The attack had a significant impact since Duang and Rizzo presented the software bug on SOP in Java. However, many software bugs other than SOP in Java may exist. Hence, browser vendors such as Google, Microsoft, and Mozilla released a software patch for the CBC mode in addition to the patch for Java [63].

The POODLE Attack

The POODLE attack [49] is the man-in-the-middle attack against CBC mode in SSL 3.0, which exploits the weakness of the validity check of the padding scheme, and is mainly divided into two phases described below (Figure 1.5):

P1. Chosen Boundary Phase:

(a) An attacker generates strings and sends them to a target.

(b) The target (unknowingly) inserts them into a plaintext so that only an unknown byte is put as the last byte of a certain block, and a boundary between the rightmost byte of the MAC and the leftmost byte of the padding becomes a block boundary.

(c) The target encrypts the resulting text in CBC mode and sends a ciphertext to a server.

P2. Blockwise Phase:

(a) The attacker substitutes a certain block of ciphertext, including the unknown byte for the last block of ciphertext.

(b) The attacker continues the above steps until the server can decrypt the manipulated ciphertext without errors.

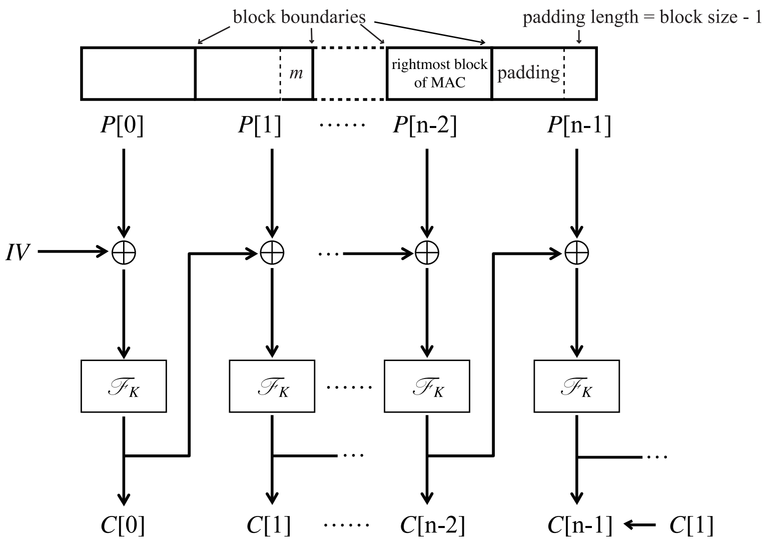


Figure 1.5: Schematic View of the POODLE Attack

The attacker can learn the unknown byte m when the server decrypts the ciphertext without errors (256 times on average). At the second and the last block, we have

$$\mathcal{F}_K^{-1}(C[1]) = C[0] \oplus P[1] \quad \text{and} \quad \mathcal{F}_K^{-1}(C[n-1]) = C[n-2] \oplus P[n-1],$$

and under the condition that no error occurs when decrypting, we have

$$P[1] = C[0] \oplus \mathcal{F}_K^{-1}(C[1]) = C[0] \oplus \mathcal{F}_K^{-1}(C[n-1]) = C[0] \oplus C[n-2] \oplus P[n-1].$$

Then

$$\begin{aligned} m &= \text{the last byte of } P[1] = \text{the last byte of } (C[0] \oplus C[n-2] \oplus P[n-1]) \\ &= \text{the last byte of } (C[0] \oplus C[n-2]) \oplus (\text{block size} - 1), \end{aligned}$$

because the server only checks whether the last byte of $(\mathcal{F}_K^{-1}(C[1]) \oplus C[n-2])$ equals the padding length when decrypting.

To avoid the POODLE attack, SSL 3.0 was disabled by default in some web browsers [43, 6, 47] and was then deprecated by IETF RFC 7568 [7].

RC4 Bias Attacks

RC4 bias attacks are plaintext recovery attacks in the broadcast setting where the same plaintext is encrypted with different keys, as shown in Figure 1.6. These attacks are not related to the CBC mode. Several biases of the RC4 keystream have already been found ([34, 4]).

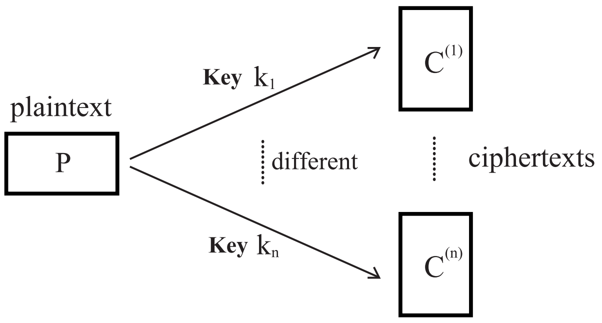


Figure 1.6: Schematic View of the Broadcast Setting

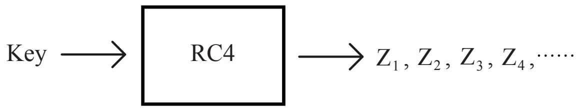


Figure 1.7: The RC4 Keystream

For example, in the broadcast setting, the second byte of the RC4 keystream (Figure 1.7) is biased to zero [45]. In other words, $Z_2 = 0$ occurs with twice the expected probability. So most frequent value of a ciphertext $C_2 (= P_2 \oplus Z_2)$ can be regarded as a plaintext P_2 in this case. Given 2^{32} ciphertexts, the first 257 bytes of the RC4 keystream $Z_1, Z_2, \ldots, Z_{257}$ can be extracted with the probability more than 0.5 [35]. Note that an attacker does not need multiple receivers and the broadcast setting can be feasible in attack scenarios similar to the other attacks.

Because of RC4 bias attacks, the use of the RC4 cipher suites in TLS was prohibited by IETF RFC 7465 [59].

Chapter 2

Preliminaries and Notations

2.1 Monomials and Terms

Here, let $\mathbb{N}$ the set of all natural numbers, let $\mathbb{Z}$ be the set of all integers, let $\mathbb{Z}_{\geq 0}$ be the set of all nonnegative integers, and let $\mathbb{F}_q$ be a finite field with q elements. $\mathcal{R}$ denotes a polynomial ring of n variables over $\mathbb{F}_q$, i.e., $\mathcal{R} = \mathbb{F}_q[x_1, \ldots, x_n] = \mathbb{F}_q[x]$.

A monomial x^a is defined by a product $x_1^{a_1} \cdots x_n^{a_n}$, where the $a = (a_1, \ldots, a_n)$ is an element of $\mathbb{Z}^n_{\geq 0}$. In addition, $\mathcal{M}$ denotes the set of all monomials in $\mathcal{R}$, i.e., $\mathcal{M} = \{x_1^{a_1} \cdots x_n^{a_n} \mid a_1, \ldots, a_n \in \mathbb{Z}_{\geq 0}\}$. For $c \in \mathbb{F}_q$ and $u \in \mathcal{M}$, we call the product cu a term and c the coefficient of u. $\mathcal{T}$ denotes the set of all terms, i.e., $\mathcal{T} = \{cu \mid c \in \mathbb{F}_q, u \in \mathcal{M}\}$. For a polynomial $f = \sum_i c_i u_i \in \mathcal{R}$ for $c_i \in \mathbb{F}_q \backslash \{0\}$ and $u_i \in \mathcal{M}$, $\mathcal{T}(f)$ denotes the set $\{c_i u_i\}$, and $\mathcal{M}(f)$ denotes the set $\{u_i\}$.

The total degree of x^a is defined by the sum $a_1 + \cdots + a_n$, which is denoted by $\deg(x^a)$. The total degree of f is defined by $\max\{\deg(u) \mid u \in \mathcal{M}(f)\}$ and is denoted by $\deg(f)$.

The degree reverse lexicographical order $\prec$ is fixed throughout this article as a monomial order. For a polynomial $f \in \mathcal{R}$, $\mathrm{LM}(f)$ denotes the leading monomial of f, i.e., $\mathrm{LM}(f) = \max_\prec \mathcal{M}(f)$, and $\mathrm{LT}(f)$ denotes the leading term of f, i.e., $\mathrm{LT}(f) = \max_\prec \mathcal{T}(f)$. In addition, $\mathrm{LC}(f)$ denotes the corresponding coefficient for $\mathrm{LT}(f)$. A polynomial f is called monic if $\mathrm{LC}(f) = 1$.

For a subset $F \subset \mathcal{R}$, $\mathrm{LM}(F)$ denotes the set of leading monomials of polynomials $f \in F$, i.e., $\mathrm{LM}(F) = \{\mathrm{LM}(f) \mid f \in F\}$.

For two monomials $x^a = x_1^{a_1} \ldots x_n^{a_n}$ and $x^b = x_1^{b_1} \ldots x_n^{b_n}$ where $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n) \in \mathbb{Z}^n_{\geq 0}$, their corresponding least common multiple (LCM) and the greatest common divisor (GCD) are defined as $\mathrm{LCM}(x^a, x^b) = x^c$ where $c = (\max(a_1, b_1), \ldots, \max(a_n, b_n))$, and $\mathrm{GCD}(x^a, x^b) = x^c$ where $c = (\min(a_1, b_1), \ldots, \min(a_n, b_n))$.

For two subsets A and B, $A \prec B$ is defined if $a \prec b$ holds for $a \in A$ and $b \in B$.

2.2 Gröbner bases

The concept of Gröbner bases was introduced by Buchberger [13] in 1979. Computing Gröbner bases is a standard tool to solve simultaneous equations. This section presents the definitions and notations used in Gröbner bases. Note that methods to compute Gröbner bases are explained in Section 2.4.

Here $\langle G \rangle$ denotes an ideal generated by a subset $G \subset \mathcal{R}$. $G \subset I$ is called a basis of an ideal I if $I = \langle G \rangle$ holds. We refer to G as Gröbner bases of I if for all $f \in I$ there exists $g \in G$ such that $\mathrm{LM}(g) | \mathrm{LM}(f)$. To compute Gröbner bases, we need to compute polynomials called S-polynomials.

Here, let $f \in \mathcal{R}$ and $G \subset \mathcal{R}$. It is said that f is reducible by G if there exist $u \in \mathcal{M}(f)$ and $g \in G$ such that $\mathrm{LM}(g) \mid u$. So we can eliminate cu from f by computing $f - \frac{cu}{\mathrm{LT}(g)} g$, where c is coefficient of u in f. In this case, g is said to be a reductor of u. If f is not reducible by G, then f is said to be a normal form of G. Repeatedly reducing f using a polynomial of G to obtain a normal form is referred to as normalization, and the function normalizing f using G is represented by $\mathrm{NF}(f, G)$.

For example, let $f = x_1x_2 + x_3, g_1 = x_1 - x_3, g_2 = x_2x_3 + 1 \in \mathbb{F}_q[x_1, x_2, x_3]$ and $G = \{g_1, g_2\}$. First, the term x_1x_2 in f is divisible by $\mathrm{LM}(g_1) = x$ and $f - x_2g_1 = f_1$ is obtained. Next, the term x_2x_3 in f_1 is divisible by $\mathrm{LM}(g_2) = x_2x_3$ and $f_1 - g_2 = x_3 - 1 = f_2$ is obtained. Finally, f_2 is the normal form of f by G since f_2 is not reducible by G.

A critical pair of two polynomials (g_1, g_2) is defined by the tuple $(\mathrm{LCM}(\mathrm{LCM}(g_1), \mathrm{LCM}(g_2)), t_1, g_1, t_2, g_2) \in \mathcal{R} \times \mathcal{M} \times \mathcal{R} \times \mathcal{M} \times \mathcal{R}$ such that

$$\mathrm{LM}(t_1g_1) = \mathrm{LM}(t_2g_2) = \mathrm{LCM}(\mathrm{LM}(g_1), \mathrm{LM}(g_2)).$$

For a critical pair p of (g_1, g_2), $\mathrm{GCD}(p)$, $\mathrm{LCM}(p)$, and $\deg(p)$ denote $\mathrm{GCD}(p) = \mathrm{GCD}(\mathrm{LM}(g_1), \mathrm{LM}(g_2))$, $\mathrm{LCM}(p) = \mathrm{LCM}(\mathrm{LM}(g_1), \mathrm{LM}(g_2))$, and $\deg(p) = \deg(\mathrm{LCM}(p))$, respectively.

The S-polynomial, $\mathrm{Spoly}(p)$ (or $\mathrm{Spoly}(g_1, g_2)$), of a critical pair p of (g_1, g_2) is defined as follows:

$$\mathrm{Spoly}(p) = \mathrm{Spoly}(g_1, g_2) = v_1g_1 - v_2g_2,$$
$$v_1 = \frac{\mathrm{LCM}(p)}{\mathrm{LT}(g_1)}, v_2 = \frac{\mathrm{LCM}(p)}{\mathrm{LT}(g_2)}.$$

$\mathrm{Left}(p)$ and $\mathrm{Right}(p)$ denote $\mathrm{Left}(p) = v_1g_1$ and $\mathrm{Right}(p) = v_2g_2$, respectively.

2.3 Targeted MQ problems

Let F be a subset $\{f_1, \ldots, f_m\} \subset \mathcal{R}$, and let $f_j \in F$ be a quadratic polynomial (i.e., $\deg(f_j) = 2$). The MQ problem is to compute a common zero $(x_1, \ldots, x_n) \in \mathbb{F}_q^n$ for a system of quadratic polynomial equations defined by F, i.e.,

$$f_j(x_1, \ldots, x_n) = 0 \text{ for all } j = 1, \ldots, m.$$

The MQ problem is discussed frequently in terms of MPKCs because representative MPKCs, e.g., UOV [38], Rainbow [23], and G*e*MSS [15] use quadratic polynomials. Note that these schemes are signature schemes and employ a system of MQ polynomial equations under the condition where $n > m$.

The computation of Gröbner bases is a fundamental tool to solve the MQ problem. If $n < m$, the system of F tends to have no solution or exactly one solution. If the system of F has no solution, $\langle 1 \rangle$ can be obtained as a Gröbner bases of $\langle F \rangle$. If it has a solution $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_q^n$, $\langle x_1 - \alpha_1, \dots, x_n - \alpha_n \rangle$ can be obtained as a Gröbner bases of $\langle F \rangle$. Thus, it is easy to obtain the solution of the system of F from the Gröbner bases of $\langle F \rangle$.

If $n > m$, it is generally necessary to compute Gröbner bases concerning lexicographic order using a Gröbner basis conversion algorithm, e.g., FGLM [29]. Another method is to convert the system associated with F to a system of multivariate polynomial equations by substituting random values to some variables and then compute its Gröbner bases. The process is repeated with other random values if there is no solution. This method is called the hybrid approach [11] and typically substitutes random values to $n - m + 1$ variables. Hence, it is crucial to solving the MQ problem with $m = n + 1$.

Note that, in our experiments, as described in Section 3.1.1, we generate the MQ problems ($m = n + 1$) with random polynomial coefficients to have at least one solution in the same manner as the Fukuoka MQ challenges [70, Algorithm 2 and Step 4 of Algorithm 1], and we assume that $\mathrm{LC}(f_j) \neq 0$ for all input polynomial f_j $(j = 1, \dots, m)$ because such polynomials are obtained with non-negligible probability for experimental purposes. Taking a change of variables into account, the probability is exactly $1 - \{1 - (1 - 1/q)^m\}^n$. For example, it is close to 1 for $q = 31$ and $(n, m) = (16, 17)$.

2.4 Gröbner bases Computation Algorithms

In this section, we introduce three algorithms to compute Gröbner bases. First, we introduce the Buchberger-style algorithm. Next, we describe the $F4$ algorithm proposed by Faugère. Finally, we described the $F4$-style algorithm proposed by Ito et al., which is the primary focus of this article.

2.4.1 Buchberger's algorithm

In 1979, Buchberger [13] introduced the concept of Gröbner bases and proposed an algorithm to compute them. He found that Gröbner bases can be computed by repeatedly generating S-polynomials and reducing them. **Algorithm** 2.4.1 describes the Buchberger-style algorithm to compute Gröbner bases. First, we generate a polynomial set G and a set of critical pairs P from the input polynomials F. We then repeat the following steps until P is empty: select one critical pair p from P, generate an S-polynomial s, reduce s to the polynomial h by G, and update G and P from (G, P, h) if h is a nonzero polynomial. The **Update** function (**Algorithm** 2.4.2) is frequently used to update G and P, which works to omit some redundant critical pairs [32]. If a polynomial h is reduced to zero, then G and P are not updated; thus the pair that generates an S-polynomial to be reduced to zero is redundant. Here, a pair selection method that selects the pair with the lowest LCM (referred to as the normal strategy) is frequently employed. If the degree reverse lexicographic order is used as a monomial order, then the pair with the lowest degree is naturally selected under the normal strategy.

Algorithm 2.4.1 Buchberger-style Algorithm

```
Input: $F = \{f_1, \ldots, f_m\} \subset \mathcal{R}$.
Output: A Gröbner bases of $\langle F \rangle$.
1: $(G, P) \leftarrow (\emptyset, \emptyset)$, $i \leftarrow 0$
2: for $f \in F$ do
3:     $(G, P) \leftarrow \mathrm{Update}(G, P, f)$
4: end for
5: while $P \neq \emptyset$ do
6:     $i \leftarrow i + 1$
7:     $p_i \leftarrow$ an element of $P$
8:     $P \leftarrow P \backslash \{p_i\}$
9:     $s_i \leftarrow \mathrm{Spoly}(p_i)$
10:    $h_i \leftarrow \mathrm{NF}(s_i, G)$
11:    if $h_i \neq 0$ then
12:        $(G, P) \leftarrow \mathrm{Update}(G, P, h_i)$
13:    end if
14: end while
15: return $G$
```

Algorithm 2.4.2 Update

```
Input: $G \subset \mathcal{R}$, $P$ is a set of critical pairs, and $h \in \mathcal{R}$.
Output: $G_{\mathrm{new}}$ and $P_{\mathrm{new}}$.
1: $h \leftarrow \frac{h}{\mathrm{LC}(h)}$
2: $C \leftarrow \{(h, g) \mid g \in G\}$, $D \leftarrow \emptyset$
3: while $C \neq \emptyset$ do
4:     $p \leftarrow$ an element of $C$, $C \leftarrow C \backslash \{p\}$
5:     if $\mathrm{GCD}(p) = 1$ or ${}^\forall p' \in C \cup D, \mathrm{LCM}(p') \nmid \mathrm{LCM}(p)$ then
6:         $D \leftarrow D \cup \{p\}$
7:     end if
8: end while
9: $P_{\mathrm{new}} \leftarrow \{p \in D \mid \mathrm{GCD}(p) \neq 1\}$
10: for $p = (g_1, g_2) \in P$ do
11:    if $\mathrm{LM}(h) \nmid \mathrm{LCM}(p)$ or
           $\mathrm{LCM}(\mathrm{LM}(h), \mathrm{LM}(g_1)) = \mathrm{LCM}(p)$ or
           $\mathrm{LCM}(\mathrm{LM}(h), \mathrm{LM}(g_2)) = \mathrm{LCM}(p)$ then
12:        $P_{\mathrm{new}} \leftarrow P_{\mathrm{new}} \cup \{p\}$
13:    end if
14: end for
15: $G_{\mathrm{new}} \leftarrow \{g \in G \mid \mathrm{LM}(h) \nmid \mathrm{LM}(g)\} \cup \{h\}$
16: return $(G_{\mathrm{new}}, P_{\mathrm{new}})$
```

2.4.2 The F4 algorithm

The F4 algorithm [27], which is a representative algorithm to compute Gröbner bases, was proposed by Faugère in 1999, and it reduces S-polynomials simultaneously. In this article, we present an F4-style algorithm with this feature.

Here, let G be a subset of $\mathcal{R}$. A matrix in which the coefficients of polynomials in G are represented as corresponding to their monomials is referred to as a Macaulay matrix of G. G is said to be a row echelon form if $\mathrm{LC}(g_1) = 1$ and $\mathrm{LM}(g_1) \neq \mathrm{LM}(g_2)$ for all $g_1 \neq g_2 \in G$. The F4-style algorithm reduces polynomials by computing row echelon forms of Macaulay matrices. For example, let $f = x_1x_2 + x_3, g_1 = x_1 - z_3, g_2 = x_2x_3 + 1 \in \mathbb{F}_q[x_1, x_2, x_3]$ as in the fourth paragraph of Section 2.2. Note that we use x_2g_1 and g_2 to compute $\mathrm{NF}(f, \{g_1, g_2\}) = x_3 - 1$. The Macaulay matrix M of $\{f, g_1, g_2\}$ is given as follows:

$$\begin{array}{c} \\ f \\ x_2g_1 \\ g_2 \end{array} \begin{array}{c} \begin{array}{cccc} x_1x_2 & x_2x_3 & x_3 & 1 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array} = M.$$

In addition, a row echelon form $\tilde{M}$ of M is given as follows:

$$\tilde{M} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}.$$

We can obtain $x_3 - 1$ from $\tilde{M}$.

Here, the F4-style algorithm is described in **Algorithm** 2.4.3. The main process is described in line 5 to 14, which selects some critical pairs using the **Select** function and reduces the polynomials of the pairs using the **Reduction** function (**Algorithm** 2.4.5). The **Select** function (**Algorithm** 2.4.4) selects critical pairs with the lowest degree based on the normal strategy. In particular, the F4-style algorithm selects all critical pairs with the lowest degree. The **Reduction** function collects reductors to reduce the polynomials and computes the row echelon form of the set of them. In addition, the **Simplify** function (**Algorithm** 2.4.6) determines the reductor whose degree is lowest from the sets of polynomials obtained during the computation of the Gröbner bases.

Algorithm 2.4.3 F4-style Algorithm

```
Input: $F = \{f_1, \ldots, f_m\} \subset \mathcal{R}$.
Output: A Gröbner basis of $\langle F \rangle$.
1: $(G, P) \leftarrow (\emptyset, \emptyset)$, $i \leftarrow 0$
2: for $f \in F$ do
3:   $(G, P) \leftarrow \mathrm{Update}(G, P, f)$
4: end for
5: while $P \neq \emptyset$ do
6:   $i \leftarrow i + 1$
7:   $P_d$, $d \leftarrow \mathrm{Select}(P)$
8:   $P \leftarrow P \backslash P_d$
9:   $L \leftarrow \{\mathrm{Left}(p) \mid p \in P_d\} \cup \{\mathrm{Right}(p) \mid p \in P_d\}$
10:  $(\tilde{H_i}^+, H_i) \leftarrow \mathrm{Reduction}(L, G, (H_j)_{j=1,\ldots,i-1})$
11:  for $h \in \tilde{H_i}^+$ do
12:    $(G, P) \leftarrow \mathrm{Update}(G, P, h)$
13:  end for
14: end while
15: return $G$
```

Algorithm 2.4.4 Select

```
Input: $P \subset \mathcal{R} \times \mathcal{R}$.
Output: $P_d \subset P$ and $d \in \mathbb{N}$.
1: $d = \min\{\deg(\mathrm{LCM}(p)) \mid p \in P\}$
2: $P_d = \{p \mid \deg(p) = d, p \in P\}$
3: return $(P_d, d)$
```

Algorithm 2.4.5 Reduction

```
Input: $L \subset \mathcal{M} \times \mathcal{R}, G \subset \mathcal{R}$, and $\mathscr{H} = (H_j)_{j=1,\ldots,i-1}$, where $H_j \subset \mathcal{R}$.
Output: $\tilde{H}^+$ and $H \subset \mathcal{R}$.
1: $L' \leftarrow \{\mathrm{Simplify}(t, f, \mathscr{H}) \mid (t, f) \in L\}$
2: $H \leftarrow \{t * f \mid (t, f) \in L'\}$
3: Done $= \mathrm{LM}(H)$
4: while Done $\neq \mathcal{M}(H)$ do
5:   $u \leftarrow$ an element of $\mathcal{M}(H) \backslash$ Done
6:   Done $\leftarrow$ Done $\cup \{u\}$
7:   if $^\exists g_1 \in G$ s.t. $\mathrm{LM}(g_1)$ divides $u$ then
8:     $u_1 \leftarrow \frac{u}{\mathrm{LM}(g_1)}$
9:     $(u_2, g_2) \leftarrow \mathrm{Simplify}(u_1, g_1, \mathscr{H})$
10:    $H \leftarrow H \cup \{u_2 g_2\}$
11:  end if
12: end while
13: $\tilde{H} \leftarrow$ row echelon form of $H$
14: $\tilde{H}^+ \leftarrow \{h \in \tilde{H} \mid \mathrm{LM}(h) \notin \mathrm{LM}(H)\}$
15: return $(\tilde{H}^+, H)$
```

Algorithm 2.4.6 Simplify

```
Input: $u \in \mathcal{M}, f \in \mathcal{R}$, and $\mathscr{H} = (H_j)_{j=1,\dots,i-1}$, where $H_j \subset \mathcal{R}$.
Output: $(u_{\text{new}}, f_{\text{new}}) \in \mathcal{M} \times \mathcal{R}$.
 1: for $t \in$ list of divisors of $u$ do
 2:   if $^{\exists}j$ s.t. $tf \in H_j$ then
 3:     $\tilde{H}_j \leftarrow$ row echelon form of $H_j$
 4:     $h \leftarrow$ an element of $\tilde{H}_j$ s.t. $\mathrm{LM}(h) = \mathrm{LM}(tf)$
 5:     if $u \neq t$ then
 6:       return Simplify($\frac{u}{t}, h, \mathscr{H}$)
 7:     else
 8:       return $(1, h)$
 9:     end if
10:   end if
11: end for
12: return $(u, f)$
```

2.4.3 The algorithm proposed by Ito et al.

Redundant critical pairs do not necessarily vanish after processing the **Update** function. Here, we introduce a method [36] to omit many redundant pairs. We assume that the degree reverse lexicographic order is employed as a monomial order, and the normal strategy is used as the pair selection strategy in the Gröbner bases computation. In addition, D_{reg} denotes the highest degree of critical pairs appearing in the Gröbner bases computation. For example, when solving the MQ problem in the Gröbner bases computation, in many cases, the degree d of the critical pairs changes, as described below.

$$d = \overbrace{\underbrace{2, 3, \ldots, D_{\text{reg}} - 1}_{\text{First half}}\, D_{\text{reg}},}^{\text{Ascending part}} \hspace{-2.5em}\underbrace{\phantom{D_{\text{reg}},}\, D_{\text{reg}} - 1, D_{\text{reg}} - 2, \ldots}_{\text{Second half}}$$

In this article, the computation until the degree of the selected pair is D_{reg} is referred to as the first half. In the first half of the computation, many redundant pairs are reduced to zero. When solving the MQ problem, Ito et al. found that if a critical pair of degree d is reduced to zero, all pairs of degree d stored at that time are also reduced to zero with a high probability. Thus, redundant critical pairs can be efficiently eliminated by ignoring all stored pairs of degree d after the critical pairs of degree d are reduced to zero. **Algorithm** 2.4.7 introduces the above method into **Algorithm** 2.4.3. In **Algorithm** 2.4.7, P_d is the set of pairs with the lowest degree d that are not tested. The subset P' contains critical pairs selected by P_d, and H^+ is new polynomials obtained by reducing P'. If the number of new polynomials H^+ is less than the number of selected pairs P', then a reduction to zero has occurred, and at this time, P_d is deleted. We add the process from line 19 to 21 to select pairs with the lowest degree according to the normal strategy.

Note that Ito et al.[36] stated that the proposed method was valid for MQ problems associated with encryption schemes, i.e., of type of $m = 2n$, but other MQ problems, including those of type $m = n + 1$, are not discussed. Moreover, they set the number of selected pairs $|P'|$ to 256 to divide P_d. Hence, they need to guarantee whether this subdividing method is optimal.

2.5 Proposed methods

As mentioned in the section above, the **SelectPd** function serves to select a subset P_d of all the critical pairs at each step for the reduction part of the F4-style algorithm. In addition, Ito et al.[36] proposed a method in which they subdivide P_d into smaller subsets $\{C_1, \ldots, C_k\}$ and perform the **Reduction** function and the **Update** function for each set C_i consecutively during no S-polynomials are reduced to zero at the reduction. On the way, if some S-polynomials reduce to zero at the reduction of a set C_j for the first time, this method ignores the remaining sets $\{C_{j+1}, \ldots, C_k\}$ and removes them from all the critical pairs.

In their paper, it was confirmed that their proposed method was effective for solving the MQ problems of the condition where $m = 2n$ and $k = 256$ only

Algorithm 2.4.7 F4-style Algorithm Proposed by Ito et al.

```
Input: $F = \{f_1, \ldots, f_m\} \subset \mathcal{R}$.
Output: A Gröbner basis of $\langle F \rangle$.
1: $(G, P) \leftarrow (\emptyset, \emptyset)$, $i \leftarrow 0$, $D_{\mathrm{reg}} \leftarrow 0$
2: for $f \in F$ do
3:    $(G, P) \leftarrow \mathrm{Update}(G, P, f)$
4: end for
5: while $P \neq \emptyset$ do
6:    $(P_d, d) \leftarrow \mathrm{Select}(P)$
7:    if $D_{\mathrm{reg}} < d$ then
8:       $D_{\mathrm{reg}} \leftarrow d$
9:    end if
10:   while $P_d \neq \emptyset$ do
11:      $i \leftarrow i + 1$
12:      $P' \leftarrow$ a subset of $P_d$
13:      $(P, P_d) \leftarrow (P \backslash P', P_d \backslash P')$
14:      $L \leftarrow \{\mathrm{Left}(p) \mid p \in P'\} \cup \{\mathrm{Right}(p) \mid p \in P'\}$
15:      $(\tilde{H}_i^+, H_i) \leftarrow \mathrm{Reduction}(L, G, (H_j)_{j=1,\ldots,i-1})$
16:      for $h \in \tilde{H}_i^+ \backslash \{0\}$ do
17:         $(G, P) \leftarrow \mathrm{Update}(G, P, h)$
18:      end for
19:      if $^{\exists} h \in \tilde{H}_i^+ \backslash \{0\}$ s.t. $\deg(h) < d$ then
20:         break
21:      end if
22:      if $|\tilde{H}_i^+| < |P'|$ and $D_{\mathrm{reg}} = d$ then
23:         $(P, P_d) \leftarrow (P \backslash P_d, \emptyset)$
24:      end if
25:   end while
26: end while
27: return $G$
```

and it was not mentioned that other types, especially $m = n + 1$, or other subdividing methods.

2.5.1 Subdividing Methods

To solve the MQ problems, Ito et al.[36] fixed the number of elements of each C_i to 256, i.e., $|C_i| = 256$. Here we propose three types of subdividing methods in the following [41].

SD1: The number of elements in C_i $(i < k)$ is fixed except C_k.
$|C_i| = 128$, 256, 512, 768, 1024, 2048, and 4096 are set in our experiment.

SD2: The number of subdivided subsets is fixed.
$k = 5$, 10, and 15 are set in our experiment.

SD3: The fraction of elements to be processed in the remaining element in P_d is fixed, i.e., $\mid C_1 \mid = \max(\lfloor r \mid P_d \mid \rfloor, 1)$ and $\mid C_i \mid = \max(\lfloor r \mid P_d \backslash \cup_{l=1}^{i-1} C_l \mid \rfloor, 1)$

Algorithm 2.5.1 SubDividePd

```
Input: $P_d \subset P$ and $d \in \mathbb{N}$.
Output: $C_1, \ldots, C_k \subset P_d$
1: $P_d = C_1 \sqcup C_2 \sqcup \cdots \sqcup C_k$ (disjoint union) s.t. $C_i \prec C_j$ for $i < j$
2: return $\{C_1, \ldots, C_k\}$
```

for $i > 1$.
$r = 1/5$, $1/10$, and $1/15$ are set in our experiment.

Furthermore, we propose two subdividing methods based on **SD1** in Section 3.1.2.

2.5.2 Removal method

It is essential to skip redundant critical pairs in the $F4$-style algorithm because it takes extra time to compute reductions of larger matrix sizes. To solve the

Algorithm 2.5.2 F4-style Algorithm Integrating the Proposed Methods

```
Input: $F = \{f_1, \ldots, f_m\} \subset \mathcal{R}$.
Output: A basis of $\langle F \rangle$.
1: $(G, P) \leftarrow (\emptyset, \emptyset)$, $i \leftarrow 0$
2: for $h \in F$ do
3:   $(G, P) \leftarrow$ Update$(G, P, h)$
4: end for
5: while $P \neq \emptyset$ do
6:   $(P_d, d) \leftarrow$ SelectPd$(P)$
7:   $P \leftarrow P \backslash P_d$
8:   while $P_d \neq \emptyset$ do
9:     // Use the method presented in Section 2.5.1
10:    $\{C_1, \ldots, C_k\} \leftarrow$ SubDividePd$(P_d)$
11:    for $l = 1$ to $k$ do
12:      $i \leftarrow i + 1$
13:      $P_d \leftarrow P_d \backslash C_l$
14:      $L \leftarrow \{\mathrm{Left}(p') \mid p' \in C_l\} \cup \{\mathrm{Right}(p') \mid p' \in C_l\}$
15:      $(\tilde{H_i}^+, H_i) \leftarrow$ Reduction$(L, G, (H_j)_{j=1,\ldots,i-1})$
16:      for $h \in \tilde{H_i}^+ \backslash \{0\}$ do
17:        $(G, P) \leftarrow$ Update$(G, P, h)$
18:      end for
19:      // Use the method presented in Section 2.5.2
20:      if $0 \in \tilde{H_i}^+$ then
21:        $P_d \leftarrow \emptyset$
22:        break
23:      end if
24:    end for
25:  end while
26: end while
27: return $G$
```

MQ problems that are defined as systems of m quadratic polynomial equations over n variables, Ito et al.[36] experimentally confirmed that once a reduction to zero occurs for some critical pairs in $P' \subset P$ then nothing but a reduction to zero will be generated for all subsequently selected critical pairs in P in the case of $\mathcal{R} = \mathbb{F}_{256}$ or $\mathbb{F}_{31}$, the number of polynomials $m = 2n$ and the number of variables $n = 16, \ldots, 25$.

We also checked such a hypothesis through computational experiments. We integrated our proposed methods into the F4-style algorithm using OpenF4 version 1.0.1 [37] as defined in **Algorithm** 2.5.2 and checked the hypothesis as described below. The OpenF4 library is an open-source implementation of the F4-style algorithm and is suitable for integrating our proposed methods into the F4-style algorithm.

Hypothesis. *If a Macaulay matrix constructed by critical pairs $p' \in P'$ $(\subset P)$ has some reductions to zero, i.e., $0 \in \tilde{H}$ at line 15 in **Algorithm** 2.5.2 with the normal strategy, then all remaining criitical pairs $p \in P$ s.t.* $\deg(\mathrm{LCM}(p)) = \deg(\mathrm{LCM}(p'))$ *will be reduced to zero with a high probability.*

We will explain the difference between a checking algorithm and a measuring algorithm. In the algorithm for measuring the software performance of the OpenF4 library and our methods, as defined in **Algorithm** 2.5.2, once a reduction to zero occurs, then the remaining critical pairs in P_d are removed. In other words, in such an algorithm, a new next P_d is selected immediately after occurring a reduction to zero. On the other hand, in the algorithm for checking the hypothesis as described above, we need to continue reductions for all remaining critical pairs and monitor that reductions to zero are consecutively generated after occurring a reduction to zero for the first time. However, note that because the behavior of the checking algorithm needs to match that of the measuring algorithm, every internal state just before processing the remaining critical pairs in the checking algorithm is reset to the state immediately after occurring a reduction to zero.

To check the hypothesis, we performed solving the MQ problems of random coefficients over $\mathbb{F}_{31}$ for the condition where $m = n + 1$ and $n = 9, \ldots, 16$ by **Algorithm** 2.5.2 with the **SD1** method. A hundred samples were generated for each problem of $n < 16$. Because of processing times, fifty samples were generated for each problem of $n = 16$. Furthermore, $\mid C_i \mid$ of **SD1** is fixed to 1, 16, 32, 256, and 512 for $n = 9, \ldots, 12$; $n = 13$; $n = 14$; $n = 15$; and $n = 16$, respectively.

Our experiments showed that the hypothesis was valid with about 0.9 probability and neither temporary basis (i.e., an element in G, at line 17 in **Algorithm** 2.5.2) nor a critical pair of higher degree arises from unused critical pairs in omitted subsets with about 0.1 probability. Moreover, all outputs for all problems contain the initial values with no errors.

2.6 The Game-Playing Approach

In the field of cryptography, security proofs are typically formulated as a sequence of rewriting a program, which is called a game [10, 66]. In each game, there exist (at least) two entities: an adversary and a challenger, and in fact, they are probabilistic processes that communicate with each other. In each game, the adversary also calls procedures which are named oracles. These interactions are regarded as games played between the adversary and the challenger. A game G is linked to a specific event S, and security related to game G is evaluated by the probability that event S occurs for all interactions by the adversary. To prove security, we construct a sequence of games: $G_0 \to G_1 \to \cdots \to G_n$ and define event S_i associated with game G_i ($i = 0, \ldots, n$), where $G = G_0$ and $S = S_0$. In the security proof, we evaluate the difference in the probability between two events $|\Pr[S_i] - \Pr[S_{i+1}]|$ at some upper bound ($n = 0, \ldots, n-1$) and show that $\Pr[S_n]$ is equal to some particular probability: typically, either 0 or 1/2. The security is proven if we can show that $\Pr[S]$ is negligibly close to the target probability through a sequence of game transformations. Here, it means that the difference between these probabilities is smaller than the inverse of every polynomial in the security parameter and for all sufficiently large values of the security parameter. The changes by the transition between games G_i and G_{i+1} should be very small in order to minimize $|\Pr[S_i] - \Pr[S_{i+1}]|$ as much as possible.

The following paragraphs contain three standard types of transitions between successive games.

Bridging steps: In such transitions, game G_i is rewritten, and it does not involve the change in the probability; i.e., $\Pr[S_i] = \Pr[S_{i+1}]$.

Transitions based on indistinguishability: In such transitions, the change is very small, but the difference in the probability distributions between two events S_i and S_{i+1} is computationally or statistically indistinguishable by every efficient adversary. Here, it means that the time complexity of the efficient algorithm (adversary) is bounded by a polynomial in the security parameter.

Transitions based on failure events: In such transitions, games G_i and G_{i+1} are identical unless some failure event F occurs, i.e., events $S_i \land \neg F$ and $S_{i+1} \land \neg F$ are the same. Let A, B, F be events. We write the symbol $A \land \neg F \Leftrightarrow B \land \neg F$ if events $A \land \neg F$ and $B \land \neg F$ are the same.

2.6.1 The Difference Lemma

Lemma 2.6.1 (Difference Lemma). *Let A, B, F be events. Suppose that $A \land \neg F \Leftrightarrow B \land \neg F$. Then $|\Pr[A] - \Pr[B]| \le \Pr[F]$.*

Proof. If $A \land \neg F \Leftrightarrow B \land \neg F$ then $\Pr[A \land \neg F] = \Pr[B \land F]$. We have

$$\begin{aligned}
|\Pr[A] - \Pr[B]| &= |\Pr[A \land F] + \Pr[A \land \neg F] - \Pr[B \land F] - \Pr[B \land \neg F]| \\
&= |\Pr[A \land F] - \Pr[B \land F]| \\
&\le \Pr[F].
\end{aligned}$$

□

2.6.2 The pseudorandom permutation/pseudorandom function (PRP/PRF) Switching Lemma

The symbol $A^P \Rightarrow 1$ denotes the event that an adversary A, which is equipped with an oracle P, outputs the bit 1. Let Perm(n) be the set of all permutations on $\{0,1\}^n$. Let Func(n) be the set of all functions from $\{0,1\}^n \to \{0,1\}^n$. The symbol $x \overset{\$}{\leftarrow} X$ denotes an assignment to x of an element uniformly randomly selected from a finite set X.

Lemma 2.6.2 (PRP/PRF Switching Lemma)**.** *Let $n \geq 1$ be an integer. Let A be an adversary that asks at most q oracle queries. Then*

$$|\Pr[A^\pi \Rightarrow 1] - \Pr[A^\rho \Rightarrow 1]| \leq \frac{q(q-1)}{2^{n+1}},$$

where π is a random permutation oracle $\pi \overset{\$}{\leftarrow} \mathrm{Perm}(n)$ and ρ is a random function oracle $\rho \overset{\$}{\leftarrow} \mathrm{Func}(n)$.

Proof. We define two games S_0 and S_1 shown in Figure 2.1. Game S_1 includes the boxed statement, and game S_0 does not include it. The flag *bad* is a Boolean value, **true** or **false**, and is initialized to **false**. The symbol $\overline{\mathrm{image}}(\pi)$ denotes the complement of image(π) relative to $\{0,1\}^n$. The array $\pi[\cdot]$ is initialized to **undefined** everywhere. Note that execution in the procedure $P(X)$ goes back to the beginning when the flag *bad* is set.

```
procedure $P(X)$ :                                                  Game $S_0$
$Y \overset{\$}{\leftarrow} \{0,1\}^n$                                               $\boxed{\text{Game } S_1}$
if $Y \in \mathrm{image}(\pi)$ then $bad \leftarrow$ true, $\boxed{Y \overset{\$}{\leftarrow} \overline{\mathrm{image}}(\pi)}$
return $\pi[X] \leftarrow Y$
```

Figure 2.1: Games Invoked in the Proof on the Switching Lemma

As game S_i progresses, the array $\pi[\cdot]$ is filled with some value in $\{0,1\}^n$. Game S_0 simulates a random function $\rho \overset{\$}{\leftarrow} \mathrm{Func}(n)$. Then $\Pr[A^\rho \Rightarrow 1] = \Pr[A^{S_0} \Rightarrow 1]$. Game S_1 simulates a random permutation $\pi \overset{\$}{\leftarrow} \mathrm{Perm}(n)$. Then $\Pr[A^\pi \Rightarrow 1] = \Pr[A^{S_1} \Rightarrow 1]$. By the Difference Lemma (Lemma 2.6.1), we have

$$\begin{aligned}|\Pr[A^\pi] - \Pr[A^\rho]| &= |\Pr[A^{S_1}] - \Pr[A^{S_0}]| \\ &\leq \Pr[A^{S_0} \text{ sets } bad].\end{aligned}$$

Let C_i be the event that a value $Y \overset{\$}{\leftarrow} \{0,1\}^n$ is already included in image(π) in i-th query. We have

$$\begin{aligned}\Pr[A^{S_0} \text{ sets } bad] &\leq \Pr[C_1] + \Pr[C_2] + \cdots + \Pr[C_q] \\ &\leq \frac{0}{2^n} + \frac{1}{2^n} + \cdots + \frac{q-1}{2^n} \\ &= \frac{q(q-1)}{2^{n+1}}.\end{aligned}$$

□

2.7 PRFs and PRPs

Let λ and τ denote security parameters, each representing the length in a byte. The length λ is considered as the block cipher's block size, and hence λ is a multiple of eight. The negligible function is denoted by $\epsilon(\lambda)$, or simply by ϵ.

PRF and PRP: A PRF $\mathcal{P}$ consists of a pair of algorithms $(\mathcal{K}, \mathcal{F})$:

- The key generation algorithm $\mathcal{K}$ is a probabilistic polynomial-time (PPT) algorithm and generates a key K.
- The evaluation algorithm $\mathcal{F}$ is a deterministic polynomial-time algorithm. It generates $\mathcal{F}(K, x)$ given the key K and a point x.

Definition 2.7.1 (PRF). *We say that $\mathcal{P} = (\mathcal{K}, \mathcal{F})$ is PRF if for any* PPT *algorithm A,*

$$\left|\Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{F}(K,\cdot)} = 1] - \Pr[\mathcal{F}' \xleftarrow{\$} \mathcal{R} : A^{\mathcal{F}'(\cdot)} = 1]\right| \leq \epsilon_{\mathrm{PRF}}(\lambda),$$

where $\mathcal{R}$ is a set of all functions such that both the domain and range are the same as $\mathcal{F}(K, \cdot)$, respectively.

If the function $\mathcal{F}_K(\cdot) := \mathcal{F}(K, \cdot)$ is a permutation, we say that $\mathcal{P}$ is a PRP. In this case, we denote the negligible function by ϵ_{PRP}.

2.8 Symmetric Key Encryptionss and Message Authentication Codes

Symmetric Key Encryption (SKE): The SKE scheme $\mathcal{SE}$ consists of a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$:

- The key generation algorithm $\mathcal{K}$ is a PPT algorithm, which generates a key K.
- The PPT encryption algorithm $\mathcal{E}$ takes a key K and a plaintext M as input, and outputs a ciphertext C. If we consider a *stateful* SKE, then $\mathcal{E}$ has additional input $\mathtt{st}$ as a state, and outputs a new state $\mathtt{st}'$.
- The decryption algorithm $\mathcal{D}$ is a deterministic polynomial time algorithm. This algorithm takes a ciphertext C and a key K as input and outputs a plaintext M or $\perp$ representing an invalid ciphertext. If we consider a stateful SKE, then $\mathcal{D}$ is given a state $\mathtt{st}$ and outputs a new state $\mathtt{st}'$.

The SKE scheme must be "decryptable." That is, for any key K and any plaintext M,

$$\mathcal{D}(K, \mathcal{E}(K, M)) = M$$

holds.

Message Authentication Code: The MAC scheme $\mathcal{MA}$ consists of a triple of algorithms $(\mathcal{K}, \mathcal{T}, \mathcal{V})$.

- The key generation algorithm $\mathcal{K}$ is a PPT algorithm and outputs a key K.

- The tag generation algorithm $\mathcal{T}$ is a deterministic polynomial-time algorithm. This algorithm takes a key K and a plaintext M as input and outputs a tag t of length τ.
- The verification algorithm $\mathcal{V}$ is a deterministic polynomial-time algorithm. This algorithm takes key K, message M, and tag t as input and outputs 0 or 1.

We say that $\mathcal{MA}$ satisfies the completeness if $\mathcal{V}(K, M, t) = 1$ is equivalent to $t = \mathcal{T}(K, M)$. We assume that, for a randomly chosen key K, $\mathcal{T}(K, \cdot)$ is a PRF. The negligible function will be denoted as $\epsilon_{\mathtt{PRF}}$.

2.9 Indistinguishability under Chosen-Plaintext Attack

To define security, we consider the left-or-right oracle $\mathsf{LR}_{K,b}(M_0, M_1) = \mathcal{E}(K, M_b)$, where $b \in \{0, 1\}$.

Definition 2.9.1 (Indistinguishability under Chosen-Plaintext Attack (IND-CPA)). *We say that the SKE $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ satisfies the $(\epsilon_{\mathrm{IND}}, q)$ IND-CPA if for any* PPT *algorithm A,*

$$\mathtt{Adv}_{\mathrm{IND}}(\lambda) = \left|\Pr[K \overset{\$}{\leftarrow} \mathcal{K}, b \overset{\$}{\leftarrow} \{0,1\}, b' \overset{\$}{\leftarrow} A^{\mathsf{LR}_{K,b}(\cdot,\cdot)} \mid b = b'] - \frac{1}{2}\right| \leq \epsilon_{\mathrm{IND}}(\lambda),$$

where q is the number of queries to LR *oracle.*

2.10 The CBC Mode in the TLS 1.0

In the CBC mode of the SSL/TLS, before encrypting the plaintext `Content`, some additional information for maintaining the SSL/TLS session is appended. That is, according to the record layer (Section 1.4.1),

$$\mathtt{Content}, \mathtt{MAC}, \mathtt{padding}, \mathtt{padding\_length}$$

are encrypted simultaneously. Here `padding` is a padding, `padding_length` is the length of the padding, and `MAC` is a tag of

$$\mathtt{MAC\_write\_secret}, \mathtt{seq\_num}, \mathtt{type}, \mathtt{version}, \mathtt{length}, \mathtt{fragment}$$

computed by the keyed-hash MAC (HMAC).

A sequence number `seq_num` is a binary sequence of 64-bit length. This is a counter starting from 0, and it is incremented for each record. This was originally designed to prevent the replay attacks, but we will show later that this counter makes the "patched" CBC mode in TLS 1.0 indistinguishable. There is other information, such as `type` and `versions`, but these are not related to our security analysis, and we will omit them henceforth.

Let λ be a block length of the underlying block cipher (in byte), and let $\|$ be concatenation. Then, for a binary sequence X, we define $X[i]$ as

$$X = \overbrace{X[0]}^{\lambda \text{ byte}} \| \overbrace{X[1]}^{\lambda \text{ byte}} \| \cdots \| \overbrace{X[n-1]}^{\leq\lambda \text{ byte}}, X[i..] = \overbrace{X[i]}^{\lambda \text{ byte}} \| \cdots \| \overbrace{X[n-1]}^{\leq\lambda \text{ byte}}.$$

```
Algorithm $\mathcal{K}_{\mathtt{WeakCBC}}$
    $K \xleftarrow{\$} \mathcal{K}_{\mathtt{PRP}}$
    Output $K$

Algorithm $\mathcal{E}_{\mathtt{WeakCBC}}(K, M; \mathtt{st})$
    $\mathtt{IV} \leftarrow \mathtt{st}$
    $M[0], \ldots, M[n-1] \leftarrow M$
    $C[0] \leftarrow \mathcal{F}_{\mathtt{PRP}}(K, M[0] \oplus \mathtt{IV})$
    For $i = 1$ to $n-1$
        $C[i] \leftarrow \mathcal{F}_{\mathtt{PRP}}(K, M[i] \oplus C[i-1])$
    Output $C = (\mathtt{IV}, C[0], \ldots, C[n-1])$ and $\mathtt{st} = C[n-1]$
```

Table 2.1: The Original CBC Mode in TLS 1.0 (WeakCBC mode)

Hence, except for the last block $X[n-1]$, $X[i]$ is λ byte. Let $X[i]$ be a byte sequence of $\lambda'(\leq \lambda)$ byte. Then we define $X[i][j]$ as

$$X[i] = \overbrace{X[i][0]}^{\text{1 byte}} \| \cdots \| \overbrace{X[i][\lambda'-1]}^{\text{1 byte}}, X[i][j..] = X[i][j] \| \cdots \| X[i][\lambda-1] \| X[i+1..].$$

2.10.1 Weak CBC Mode in TLS1.0

Let $\mathcal{P} = (\mathcal{K}_{\mathtt{PRP}}, \mathcal{F}_{\mathtt{PRP}})$ be a PRP. The CBC mode in TLS 1.0 is implemented as in Table 2.1, where we assume that the length of the message M is a multiple of λ, and the IV is chosen randomly at the beginning. The decryption algorithm $\mathcal{D}_{\mathtt{WeakCBC}}$ is not described since it is trivial.

This version of the CBC mode is called the WeakCBC mode. In this mode WeakCBC mode, since the adversary knows $\mathtt{IV}(= C[n-1])$ in advance, it does not satisfy the IND-CPA security. This is the reason why the original CBC mode (WeakCBC) is vulnerable to the BEAST attack.

2.10.2 Unpatched CBC

```
Algorithm $\mathcal{K}_{\mathtt{WeakTLS1.0}}$
    $K_{\mathtt{WeakCBC}} \xleftarrow{\$} \mathcal{K}_{\mathtt{WeakCBC}}$
    $K_{\mathtt{MA}} \xleftarrow{\$} \mathcal{K}_{\mathtt{MA}}$
    $K \leftarrow (K_{\mathtt{WeakCBC}}, K_{\mathtt{MA}})$
    Output $K$

Algorithm $\mathcal{E}_{\mathtt{WeakTLS1.0}}(K, M; \mathtt{st})$
    Parse $\mathtt{st}$ as $(\mathtt{st}_{\mathtt{WeakCBC}}, c)$
    Parse $K$ as $(K_{\mathtt{WeakCBC}}, K_{\mathtt{MA}})$
    $t \leftarrow \mathcal{T}(K_{\mathtt{MA}}, c \| |M| \| M)$
    $(C, \mathtt{st}_{\mathtt{WeakCBC}}) \leftarrow \mathcal{E}_{\mathtt{WeakCBC}}(K_{\mathtt{WeakCBC}}, \mathsf{Pad}(M \| t); \mathtt{st})$
    Output $(C, (\mathtt{st}_{\mathtt{WeakCBC}}, c + |M|))$

Algorithm $\mathcal{D}_{\mathtt{WeakTLS1.0}}(K, C; \mathtt{st})$
    Parse $\mathtt{st}$ as $c$
    Parse $K$ as $(K_{\mathtt{WeakCBC}}, K_{\mathtt{MA}})$
    $M' \leftarrow \mathcal{D}_{\mathtt{WeakCBC}}(K_{\mathtt{WeakCBC}}, C)$
    $M'' \leftarrow \mathsf{Pad}^{-1}(M')$
    If $M'' \neq \perp$ then parse $M''$ as $M \| t$
    else output $\perp$
    If $\mathcal{T}(K_{\mathtt{MA}}, c \| |M| \| M) = t$,
      output $(M, c + |M|)$
    else output $\perp$
```

Table 2.2: Unpatched CBC (WeakTLS1.0)

In TLS 1.0, the encryption is done by Mac-then-Enc. Hence, the tag is generated before the message is encrypted in the CBC mode. (Table 2.2.) In

Table 2.2, c plays the role of the counter which starts from 0. The counter represents the sequence number `seq_num` in Section 2.10. Other information, such as `type` is not related to our security analysis, and hence we will omit it from this algorithm.

We call the authenticated encryption of Table 2.2 WeakTLS1.0. The algorithm Pad is the padding algorithm which is defined as (1.1), and Pad^{-1} is the algorithm which removes the padding. As defined in Section 2.8, $\mathcal{MA} = (\mathcal{K}_{\mathtt{MA}}, \mathcal{T}, \mathcal{V})$ is the MAC.

Since `IV` is predictable, WeakTLS1.0 does not satisfy the IND-CPA property.

Chapter 3

Contributions

3.1 Selection Strategy of F4-style Algorithm

In this section, we describe the software performance of the proposed methods introduced in Section 2.5. Then, we describe the behavior of the proposed methods in the first half of the computation.

3.1.1 Software performance comparisons

We integrated our proposed methods into the F4-style algorithm using the OpenF4 library version 1.0.1 [37] and compared their software performances, including the original OpenF4. In measuring CPU time, only the functions included in the OpenF4 library were used. We used a non-uniform memory access (NUMA) machine with four nodes of Intel(R) Xeon(R) Platinum 8180 at 2.50 GHz processors and 256 GB RAM each (1TB RAM total).

We benchmarked these implementations on the MQ problems similar to the Fukuoka MQ challenge of Type V over a base field $\mathbb{F}_{256}$ and Type VI over a base field $\mathbb{F}_{31}$ [71]. These problems are defined as MQ polynomial systems of m equations over n variables with $m = n + 1$, based on the hybrid approach [11]. We experimented with ten samples for each parameter : $m = n + 1$ and $n = 9, \ldots, 15$ over $\mathbb{F}_{256}$ and $m = n + 1$ and $n = 9, \ldots, 16$ over $\mathbb{F}_{31}$. Note that, on our machine, we could not run these programs for $n > 16$ over $\mathbb{F}_{31}$ and for $n > 15$ over $\mathbb{F}_{256}$ because the OpenF4 library needs more significant memory than the installed RAM of our machine. Our benchmarking results for $m = n+1$ and $n = 9, \ldots, 15$ over $\mathbb{F}_{256}$ are listed in Table 3.1 and for $m = n+1$ and $n = 9, \ldots, 16$ over $\mathbb{F}_{31}$ are listed in Table 3.2. For the first and second halves of the computation, the top four records by the proposed methods and the record by the OpenF4 library are shown in Figure 3.1a, Figure 3.1b, Figure 3.2a, and Figure 3.2b.

These experiments observed that there was no failure to compute solutions, including initially selected values and standard variations (σ) of the CPU times were relatively small and the F4-style algorithms integrating our proposed methods were faster than the original OpenF4 library, e.g., up to factor 7.21 in the case of **SD1** under $n = 15$ and $\mid C_i \mid= 512$ over $\mathbb{F}_{256}$ and up to factor 6.02 in the case of **SD1** under $n = 16$ and $\mid C_i \mid= 1024$ over $\mathbb{F}_{31}$. By these results, it could be argued that **SD1** is faster than all other methods. However, if we focus only on the first half of the computation, it is found that the **SD3** method with $r = 15$ may be faster than otherwise in the case of both $\mathbb{F}_{256}$ and $\mathbb{F}_{31}$. This reason will be discussed in the next section. If we distinguish between the first and second halves, we conclude that it is appropriate to apply the **SD3** method with $r = 15$ in the first half and the **SD1** method with $\mid C_i \mid \leq 512$ in the second half. For example, a combination of **SD3** with $r = 1/15$ for the first half and **SD1** with $\mid C_i \mid = 512$ for the second half (i.e., **SD3** followed by **SD1**) is faster than otherwise, e.g., up to factor 7.76 for $(n, m) = (16, 17)$ over $\mathbb{F}_{31}$.

3.1.2 The performance behavior of the proposed method in the first half

In our experiments, we counted CPU time and the number of critical pairs used at each reduction. Our experiments found that the minimum number of critical pairs that generate a reduction to zero for the first time is approximately constant. Note, however, that if more than one of the critical pairs arises, we take the maximum number among them. The number of critical pairs that generate a reduction to zero for the first time for each (n, m) and d is listed in Table 3.3. The symbol Total in Table 3.3 is defined by the number of critical pairs before reducing the Macaulay matrix for each (n, m) and d. The symbol Min in Table 3.3 is defined by the minimum number of critical pairs that generate a reduction to zero for the first time for each (n, m) and d. The ratio of Min to Total is shown in Figure 3.4a. Figure 3.4a shows that the ratios gradually decrease, and the decreasing ratios are not constant. These tendencies likely caused that **SD3** with $r = 1/15$ was handy in the first half.

Here, we propose the subdividing method **SD4** in the first half in the following.

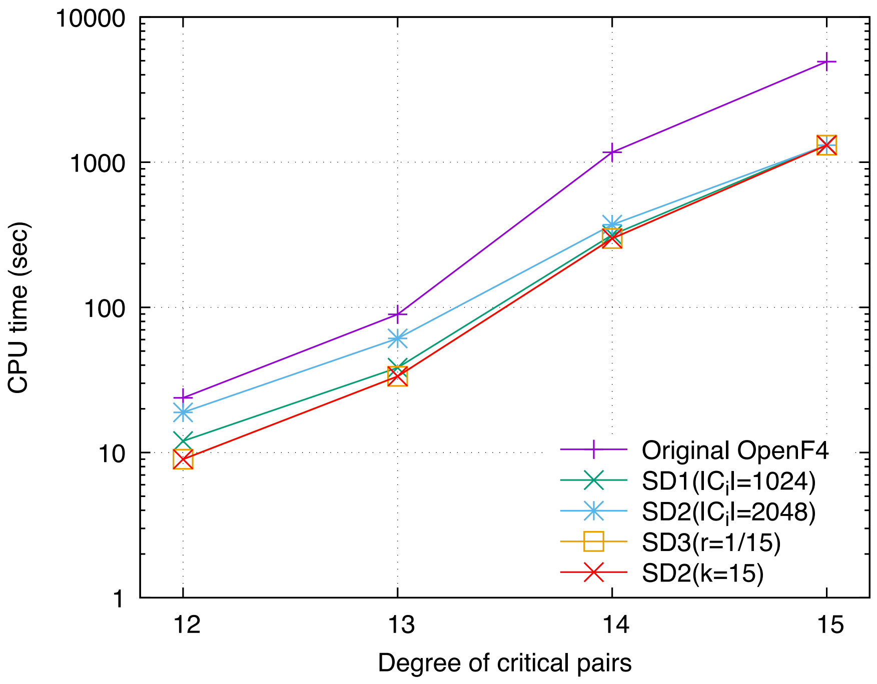
SD4: The number of elements in the first subset $\mid C_1 \mid$ is 1 plus the number Min specified in Table 3.3, regarding **SD1**. $\mid C_i \mid (i > 2)$ is fixed to a small value in place.

We experimented with **SD4** in the first half followed by **SD1** with $\mid C_i \mid = 512$ in the second half (i.e., **SD4** $\Rightarrow$ **SD1**). Our benchmarking results of **SD4** followed by **SD1** are listed in Table 3.1 and 3.2. For the first half of the computation, the record by **SD4** and the record by the OpenF4 library are shown in Figure 3.3a and 3.3b.

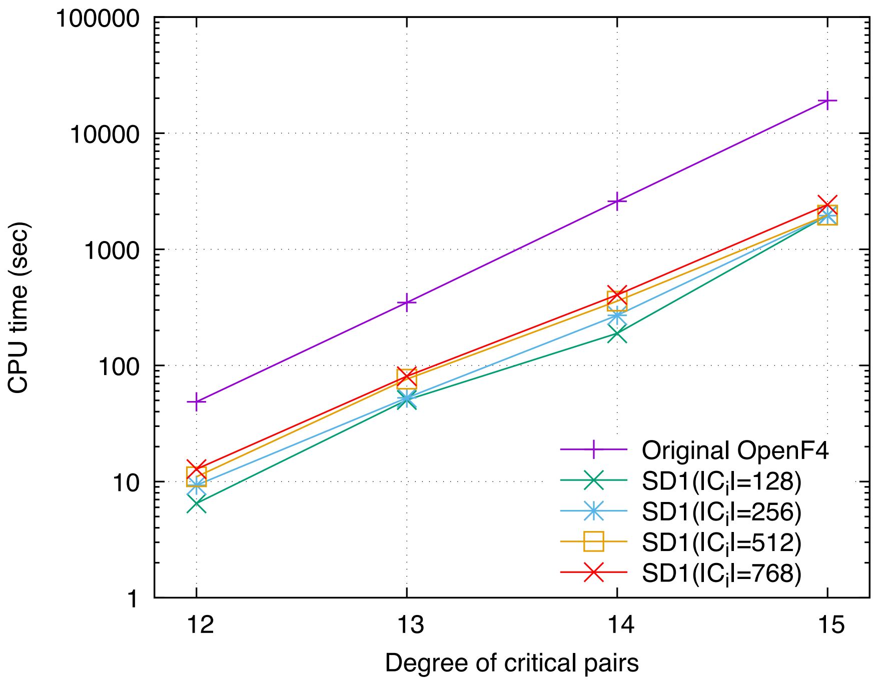
Here, we investigate the approximation of the ratio (r) shown in Figure 3.4a. The **Simplify** function outputs a product $x_i \times p$ such that x_i is a variable and p is a polynomial with a high probability, as stated in the paper [27, Remark 2.6]. So, it seems likely that rows of the Macaulay matrix are composed of products $x_1^{a_1} \cdots x_i^{a_i} \times p$ where $1 \leq i \leq n$ and p is a polynomial with a high probability because of the normal strategy. In addition, matrix elements in the leftmost columns of the Macaulay matrix come from the leading monomials. Hence, it seems reasonable to suppose that the ratio r is approximately related to the number of monomials of degree d:

$$\mid \{x_1^{a_1} \ldots x_n^{a_n} | a_1 + \cdots + a_n = d\} \mid = \binom{d+n-1}{n-1} \approx \frac{d^{n-1}}{(n-1)!} + (\text{lower terms}).$$

So, we assume that the r is proportional to a power of d, i.e., $r \propto d^c$ (c: constant) by ignoring lower terms. We have $\log r = a \log d + b$ (a, b: constant). To see the

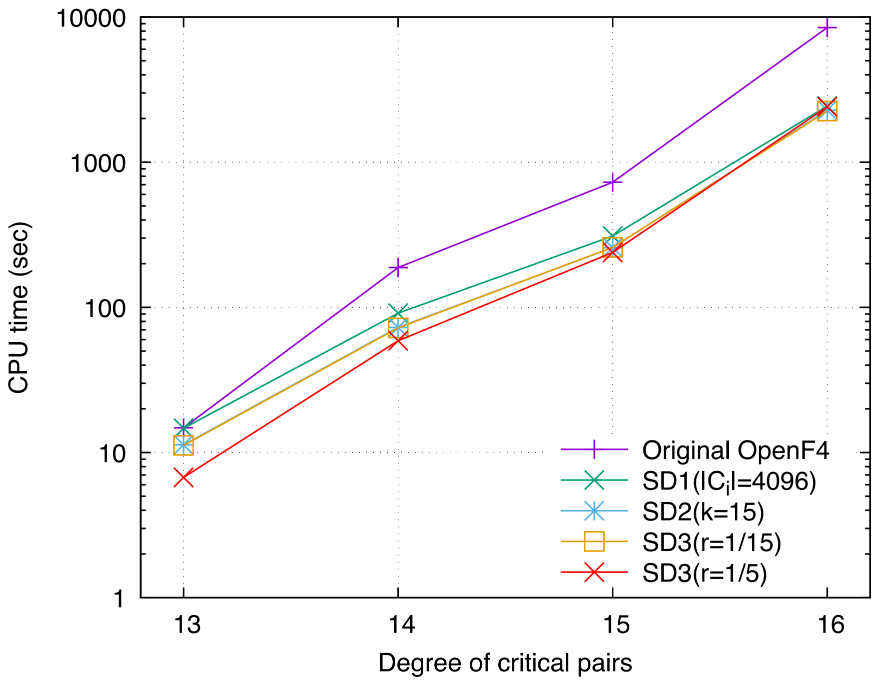


(a) CPU Times in the First Half

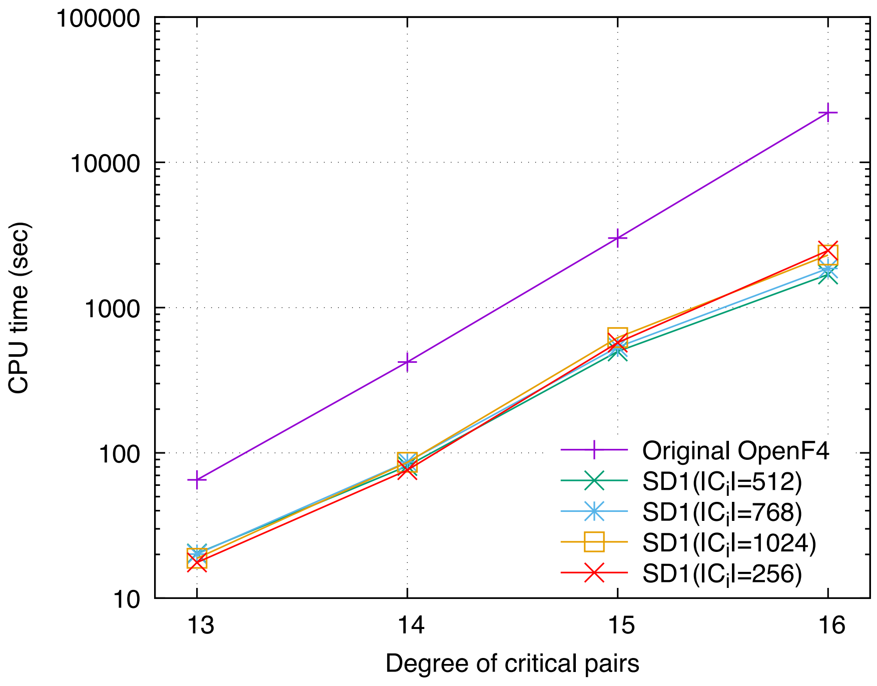


(b) CPU Times in the Second Half

Figure 3.1: Benchmark Results over $\mathbb{F}_{256}$ for $m = n + 1$

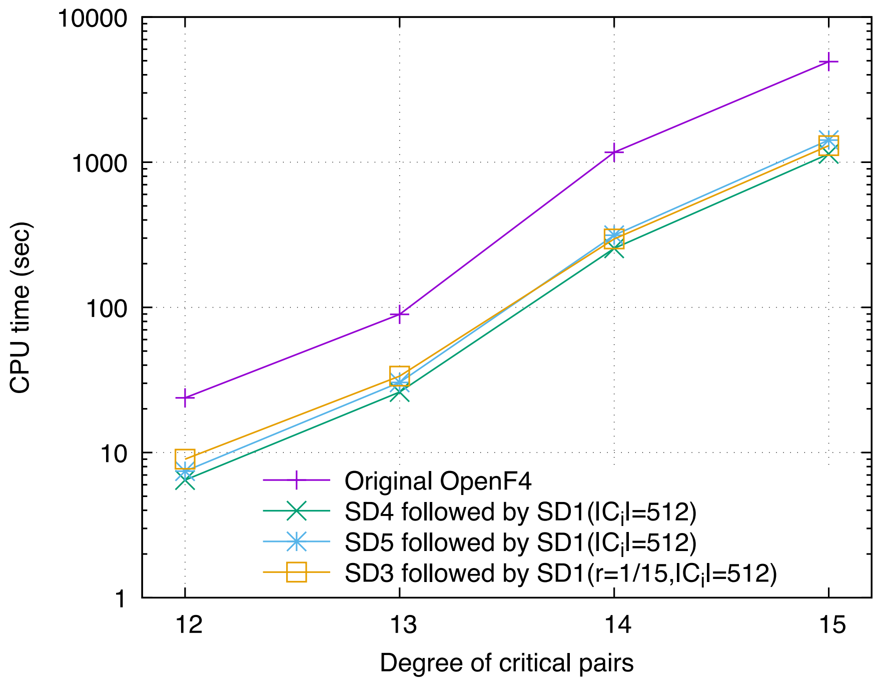


(a) CPU Times in the First Half

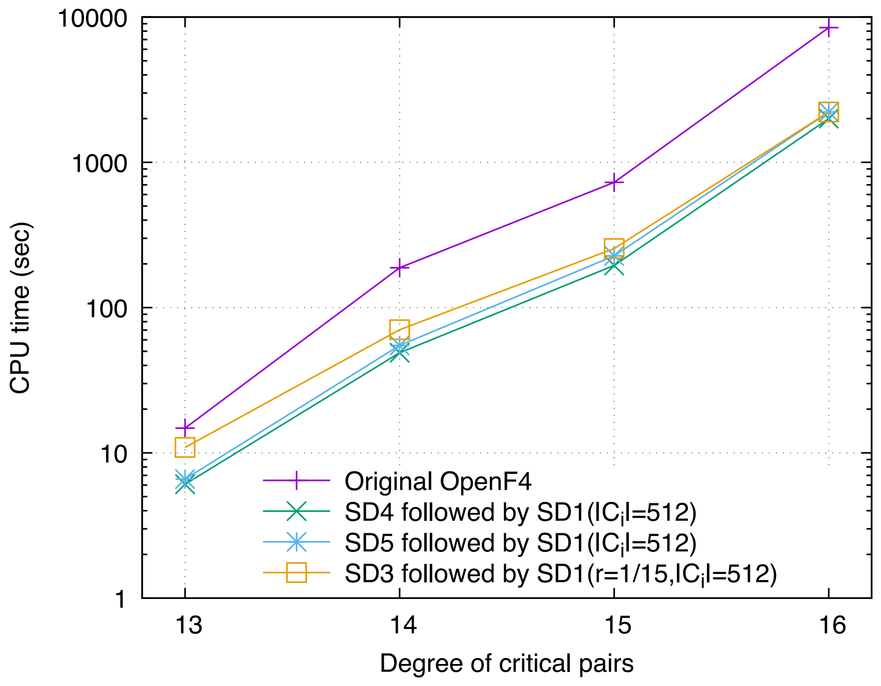


(b) CPU Times in the Second Half

Figure 3.2: Benchmark Results over $\mathbb{F}_{31}$ for $m = n + 1$



(a) CPU Times (sec) over $\mathbb{F}_{256}$ in the First Half



(b) CPU Times (sec) over $\mathbb{F}_{31}$ in the First Half

Figure 3.3: Benchmark Results for $m = n + 1$

relation between d and r, r is displayed in the log-log scale as shown in Figure 3.4b. Furthermore, we assume that linear expressions in n approximate both a and b, and we distinguish between the approximations according to whether n is even and odd. The linear regression analysis of a $(n = 9, \dots, 16)$ shows that

$$a \approx a(n) = \begin{cases} 0.0566\,n - 2.63 & \text{(n is odd)}, \\ 0.0443\,n - 2.38 & \text{(n is even)}. \end{cases}$$

In addition, the regression analysis of b $(n = 9, \dots, 16)$ shows that

$$b \approx b(n) = \begin{cases} -0.0301\,n + 1.15 & \text{(n is odd)}, \\ -0.0236\,n + 1.01 & \text{(n is even)}. \end{cases}$$

Then, we add the constant value 0.0542 as r pass over all points in Figure 3.4b because the expected number of critical pairs should not be less than the required number.

Finally, we propose the subdividing method **SD5** in the first half in the following.

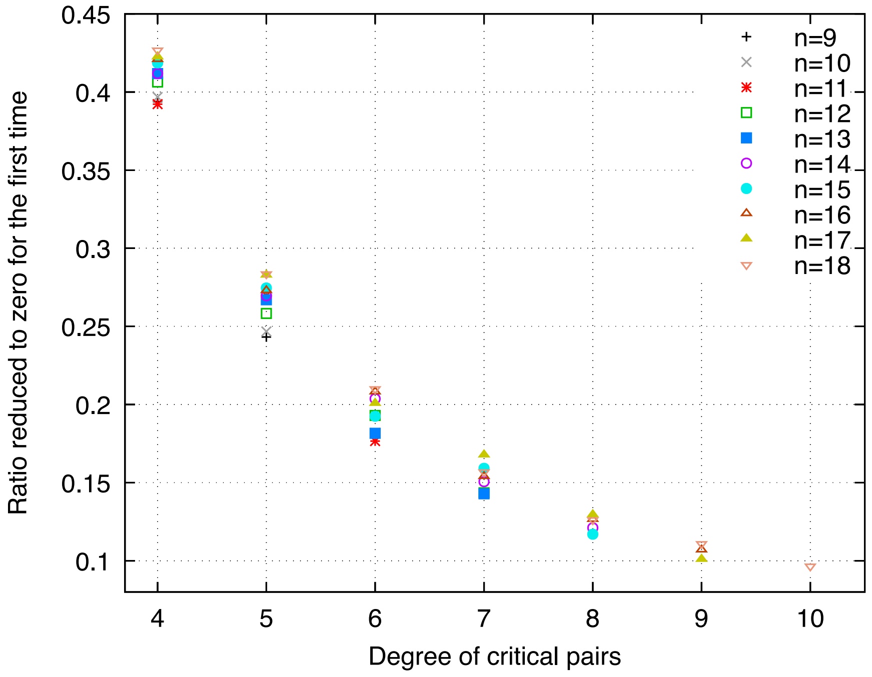
SD5: The number of elements in the first subset $\mid C_1 \mid$ is $r(d, n)$ multiplied by the number Total specified in Table 3.3, regarding **SD1**. $\mid C_i \mid (i > 2)$ is fixed to a small value in place. $r(n, d)$ is defined by the following:

$$r \approx r(n, d) = \begin{cases} 0.0542 + d^{\,0.0566\,n - 2.63}\, 10^{-0.0301\,n + 1.15} & \text{(n is odd)}, \\ 0.0542 + d^{\,0.0443\,n - 2.38}\, 10^{-0.0236\,n + 1.01} & \text{(n is even)}. \end{cases}$$

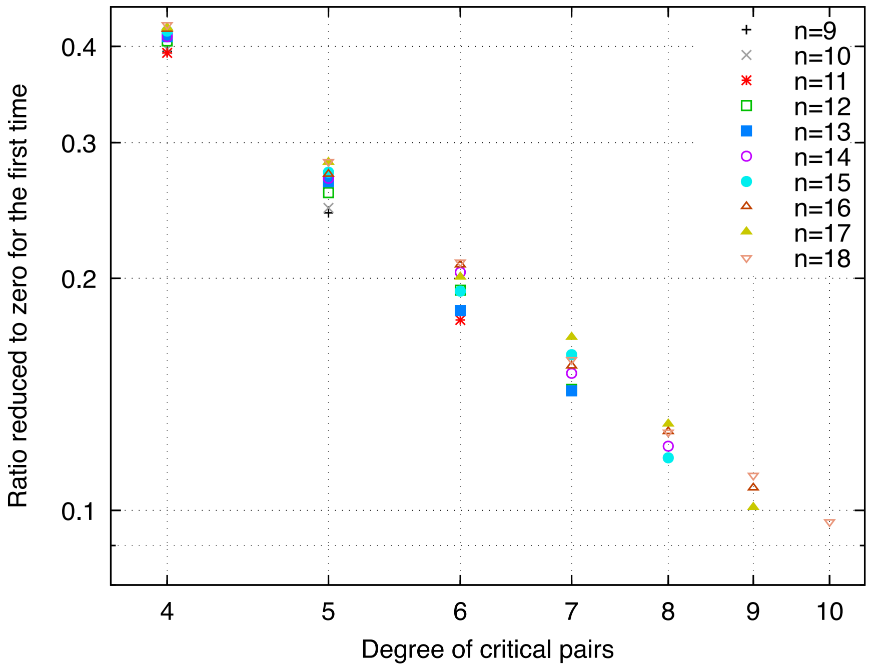
We experimented with **SD5** in the first half followed by **SD1** with $\mid C_i \mid = 512$ in the second half (i.e., **SD5** $\Rightarrow$ **SD1**). Our benchmarking results of **SD5** followed by **SD1** are listed in Table 3.1 and 3.2. For the first half of the computation, the record by **SD5** and the record by the OpenF4 library are shown in Figure 3.3a and 3.3b.
Our benchmark results for MQ problems over $\mathbb{F}_{256}$ for $m = n + 1$ are shown in Table 3.1. Our benchmark results for MQ problems over $\mathbb{F}_{31}$ for $m = n + 1$ are shown in Table 3.2. Experimental results of the number of critical pairs to generate a reduction to zero in the first half for $m = n + 1$ over $\mathbb{F}_{256}$ and $\mathbb{F}_{31}$ are shown in Table 3.3.

3.1.3 Conclusions and future work

The experimental results in our previous study have demonstrated that the subdividing method **SD1** with $\mid C_i \mid = 256$ and the removal method are valid for solving a system of MQ polynomial equations associated with encryption schemes. In this study, we proposed three basic methods (**SD1**, **SD2**, and **SD3**) and two extra methods (**SD4** and **SD5**) in the subdividing method of the $F4$-style algorithm. Our proposed method considerably improved the performance of the $F4$-style algorithm by omitting redundant critical pairs based on the removal method. Then, our experimental results validated these methods to solve a system of MQ polynomial equations under $m = n + 1$. Furthermore, we found that the number of critical pairs that generate a reduction to zero for the first time was approximately constant under $m = n + 1$ within the range



(a) Experimental Results of the Ratio of Critical Pairs for a Reduction to Zero in the First Half over $\mathbb{F}_{256}$ and $\mathbb{F}_{31}$ for $m = n + 1$



(b) $\log_{10}$-$\log_{10}$ Graph of the Ration of Critical Pairs for a Reduction to Zero in the First Half over $\mathbb{F}_{256}$ and $\mathbb{F}_{31}$ for $m = n + 1$

Figure 3.4: Benchmark Results for $m = n + 1$ over $\mathbb{F}_{256}$

Table 3.1: Benchmark Results for MQ Problems over $\mathbb{F}_{256}$ for $m = n + 1$, Total CPU Time (sec)

(n,m)		$(9,10)$	$(10,11)$	$(11,12)$	$(12,13)$	$(13,14)$	$(14,15)$	$(15,16)$
Original OpenF4								
	Average	2.33×10^{-1}	1.58×10^{0}	9.33×10^{0}	7.24×10^{1}	4.38×10^{2}	3.76×10^{3}	2.40×10^{4}
	σ	9.17×10^{-3}	2.32×10^{-3}	5.25×10^{-3}	1.10×10^{-1}	2.81×10^{-1}	9.77×10^{0}	5.25×10^{2}
	Before D_{reg}	5.04×10^{-2}	5.47×10^{-1}	2.00×10^{0}	2.38×10^{1}	8.95×10^{1}	1.17×10^{3}	4.93×10^{3}
	After D_{reg}	1.80×10^{-1}	1.03×10^{0}	7.33×10^{0}	4.86×10^{1}	3.48×10^{2}	2.59×10^{3}	1.91×10^{4}
SD1								
$\mid C_i \mid = 128$	Average	1.15×10^{-1}	5.74×10^{-1}	2.90×10^{0}	1.54×10^{1}	8.49×10^{1}	5.72×10^{2}	3.68×10^{3}
	σ	9.80×10^{-4}	1.37×10^{-3}	4.09×10^{-3}	1.74×10^{-2}	5.40×10^{-2}	5.51×10^{-1}	3.00×10^{1}
	Before D_{reg}	3.32×10^{-2}	2.82×10^{-1}	9.22×10^{-1}	8.91×10^{0}	3.44×10^{1}	3.82×10^{2}	1.73×10^{3}
	After D_{reg}	7.77×10^{-2}	2.85×10^{-1}	1.96×10^{0}	6.49×10^{0}	5.04×10^{1}	1.89×10^{2}	1.96×10^{3}
$\mid C_i \mid = 256$	Average	1.63×10^{-1}	6.50×10^{-1}	3.05×10^{0}	1.75×10^{1}	8.62×10^{1}	5.85×10^{2}	3.41×10^{3}
	σ	3.53×10^{-3}	9.17×10^{-4}	2.42×10^{-3}	1.41×10^{-1}	5.02×10^{-2}	8.31×10^{-1}	1.44×10^{1}
	Before D_{reg}	4.94×10^{-2}	3.01×10^{-1}	8.80×10^{-1}	8.22×10^{0}	3.36×10^{1}	3.15×10^{2}	1.46×10^{3}
	After D_{reg}	1.10×10^{-1}	3.45×10^{-1}	2.16×10^{0}	9.30×10^{0}	5.26×10^{1}	2.70×10^{2}	1.95×10^{3}
$\mid C_i \mid = 512$	Average	2.18×10^{-1}	9.51×10^{-1}	4.03×10^{0}	1.89×10^{1}	1.11×10^{2}	6.64×10^{2}	3.33×10^{3}
	σ	7.43×10^{-3}	1.86×10^{-3}	4.50×10^{-3}	8.20×10^{-2}	3.60×10^{-2}	8.27×10^{-1}	1.40×10^{1}
	Before D_{reg}	4.99×10^{-2}	4.41×10^{-1}	1.28×10^{0}	7.91×10^{0}	3.46×10^{1}	3.06×10^{2}	1.36×10^{3}
	After D_{reg}	1.65×10^{-1}	5.05×10^{-1}	2.74×10^{0}	1.10×10^{1}	7.62×10^{1}	3.58×10^{2}	1.97×10^{3}
$\mid C_i \mid = 768$	Average	2.27×10^{-1}	1.22×10^{0}	5.11×10^{0}	2.28×10^{1}	1.14×10^{2}	7.10×10^{2}	3.76×10^{3}
	σ	7.76×10^{-3}	1.50×10^{-3}	3.58×10^{-3}	1.24×10^{-2}	3.92×10^{-2}	1.01×10^{0}	7.57×10^{0}
	Before D_{reg}	4.97×10^{-2}	5.43×10^{-1}	1.59×10^{0}	1.00×10^{1}	3.33×10^{1}	3.05×10^{2}	1.33×10^{3}
	After D_{reg}	1.75×10^{-1}	6.70×10^{-1}	3.51×10^{0}	1.28×10^{1}	8.07×10^{1}	4.05×10^{2}	2.42×10^{3}
$\mid C_i \mid = 1024$	Average	2.29×10^{-1}	1.40×10^{0}	6.18×10^{0}	2.68×10^{1}	1.24×10^{2}	7.17×10^{2}	4.40×10^{3}
	σ	8.80×10^{-3}	3.53×10^{-3}	2.80×10^{-3}	4.58×10^{-2}	3.18×10^{-1}	4.52×10^{-1}	1.35×10^{1}
	Before D_{reg}	4.95×10^{-2}	5.44×10^{-1}	1.93×10^{0}	1.20×10^{1}	3.84×10^{1}	3.16×10^{2}	1.32×10^{3}
	After D_{reg}	1.77×10^{-1}	8.50×10^{-1}	4.25×10^{0}	1.47×10^{1}	8.58×10^{1}	4.01×10^{2}	3.07×10^{3}
$\mid C_i \mid = 2048$	Average	2.30×10^{-1}	1.57×10^{0}	8.63×10^{0}	4.17×10^{1}	1.85×10^{2}	9.04×10^{2}	4.91×10^{3}
	σ	9.07×10^{-3}	1.19×10^{-3}	5.04×10^{-2}	2.42×10^{-2}	3.00×10^{-1}	1.80×10^{0}	9.34×10^{0}
	Before D_{reg}	4.98×10^{-2}	5.42×10^{-1}	1.99×10^{0}	1.89×10^{1}	6.11×10^{1}	3.71×10^{2}	1.31×10^{3}
	After D_{reg}	1.77×10^{-1}	1.03×10^{0}	6.64×10^{0}	2.28×10^{1}	1.23×10^{2}	5.33×10^{2}	3.60×10^{3}
$\mid C_i \mid = 4096$	Average	2.30×10^{-1}	1.57×10^{0}	9.35×10^{0}	6.35×10^{1}	2.92×10^{2}	1.34×10^{3}	6.47×10^{3}
	σ	9.19×10^{-3}	6.40×10^{-4}	8.52×10^{-2}	1.12×10^{-2}	2.14×10^{-1}	5.00×10^{-1}	1.73×10^{1}
	Before D_{reg}	4.95×10^{-2}	5.43×10^{-1}	1.99×10^{0}	2.37×10^{1}	8.93×10^{1}	5.80×10^{2}	1.99×10^{3}
	After D_{reg}	1.77×10^{-1}	1.03×10^{0}	7.36×10^{0}	3.97×10^{1}	2.03×10^{2}	7.57×10^{2}	4.49×10^{3}
SD2								
$k = 5$	Average	1.17×10^{-1}	5.86×10^{-1}	3.33×10^{0}	2.21×10^{1}	1.38×10^{2}	1.09×10^{3}	7.25×10^{3}
	σ	1.14×10^{-2}	1.02×10^{-3}	2.54×10^{-2}	9.39×10^{-3}	1.68×10^{-1}	9.22×10^{-1}	5.91×10^{1}
	Before D_{reg}	3.50×10^{-2}	2.35×10^{-1}	8.18×10^{-1}	7.38×10^{0}	3.02×10^{1}	3.41×10^{2}	1.48×10^{3}
	After D_{reg}	6.59×10^{-2}	3.41×10^{-1}	2.50×10^{0}	1.47×10^{1}	1.08×10^{2}	7.50×10^{2}	5.77×10^{3}
$k = 10$	Average	1.27×10^{-1}	5.38×10^{-1}	2.67×10^{0}	1.89×10^{1}	9.63×10^{1}	8.69×10^{2}	5.58×10^{3}
	σ	4.58×10^{-4}	7.81×10^{-4}	2.51×10^{-2}	1.00×10^{-2}	3.96×10^{-2}	1.22×10^{0}	1.03×10^{1}
	Before D_{reg}	3.64×10^{-2}	2.63×10^{-1}	9.39×10^{-1}	8.02×10^{0}	3.19×10^{1}	3.40×10^{2}	1.49×10^{3}
	After D_{reg}	4.74×10^{-2}	2.61×10^{-1}	1.71×10^{0}	1.08×10^{1}	6.44×10^{1}	5.30×10^{2}	4.10×10^{3}
$k = 15$	Average	1.19×10^{-1}	5.23×10^{-1}	2.73×10^{0}	1.66×10^{1}	9.53×10^{1}	7.47×10^{2}	3.77×10^{3}
	σ	3.00×10^{-4}	1.11×10^{-3}	3.87×10^{-3}	8.85×10^{-3}	1.19×10^{-1}	9.83×10^{-1}	8.57×10^{0}
	Before D_{reg}	3.73×10^{-2}	2.71×10^{-1}	1.06×10^{0}	8.64×10^{0}	3.41×10^{1}	3.08×10^{2}	1.33×10^{3}
	After D_{reg}	4.60×10^{-2}	2.25×10^{-1}	1.62×10^{0}	7.85×10^{0}	6.06×10^{1}	4.39×10^{2}	2.44×10^{3}
SD3								
$r = 1/5$	Average	1.12×10^{-1}	5.81×10^{-1}	3.27×10^{0}	2.21×10^{1}	1.35×10^{2}	1.09×10^{3}	7.04×10^{3}
	σ	1.02×10^{-3}	8.72×10^{-4}	4.84×10^{-3}	6.02×10^{-3}	7.38×10^{-2}	6.49×10^{-1}	8.54×10^{1}
	Before D_{reg}	3.49×10^{-2}	2.28×10^{-1}	8.06×10^{-1}	7.35×10^{0}	3.00×10^{1}	3.40×10^{2}	1.48×10^{3}
	After D_{reg}	6.79×10^{-2}	3.40×10^{-1}	2.44×10^{0}	1.47×10^{1}	1.05×10^{2}	7.50×10^{2}	5.56×10^{3}
$r = 1/10$	Average	1.16×10^{-1}	5.36×10^{-1}	2.96×10^{0}	1.92×10^{1}	1.15×10^{2}	8.60×10^{2}	5.88×10^{3}
	σ	5.39×10^{-4}	4.90×10^{-4}	2.23×10^{-3}	7.58×10^{-3}	4.53×10^{-2}	1.09×10^{0}	4.79×10^{1}
	Before D_{reg}	3.60×10^{-2}	2.60×10^{-1}	9.19×10^{-1}	8.37×10^{0}	3.14×10^{1}	3.30×10^{2}	1.47×10^{3}
	After D_{reg}	6.58×10^{-2}	2.62×10^{-1}	2.02×10^{-0}	1.08×10^{1}	8.34×10^{1}	5.30×10^{2}	4.41×10^{3}
$r = 1/15$	Average	1.17×10^{-1}	5.55×10^{-1}	3.19×10^{0}	1.66×10^{1}	9.41×10^{1}	7.44×10^{2}	4.94×10^{3}
	σ	3.00×10^{-4}	7.00×10^{-4}	2.88×10^{-3}	1.19×10^{-1}	1.12×10^{-1}	1.07×10^{0}	9.59×10^{0}
	Before D_{reg}	4.26×10^{-2}	2.87×10^{-1}	1.04×10^{0}	8.99×10^{0}	3.35×10^{1}	2.98×10^{2}	1.31×10^{3}
	After D_{reg}	5.45×10^{-2}	2.48×10^{-1}	2.12×10^{0}	7.55×10^{0}	6.05×10^{1}	4.46×10^{2}	3.63×10^{3}
SD3 followed by SD1 with $r = 1/15$ and $\mid C_i \mid = 512$								
	Average	2.23×10^{-1}	8.20×10^{-1}	3.78×10^{0}	2.01×10^{1}	1.11×10^{2}	6.53×10^{2}	3.24×10^{3}
	σ	6.45×10^{-3}	9.00×10^{-4}	3.04×10^{-3}	1.33×10^{-2}	9.05×10^{-2}	5.78×10^{-1}	1.35×10^{1}
	Before D_{reg}	4.28×10^{-2}	2.89×10^{-1}	1.05×10^{0}	9.00×10^{0}	3.36×10^{1}	2.96×10^{2}	1.30×10^{3}
	After D_{reg}	1.65×10^{-1}	5.13×10^{-1}	2.71×10^{0}	1.11×10^{1}	7.69×10^{1}	3.57×10^{2}	1.94×10^{3}
SD4 followed by SD1 with $\mid C_i \mid = 512$								
	Average	1.91×10^{-1}	7.09×10^{-1}	3.48×10^{0}	1.75×10^{1}	1.03×10^{2}	6.14×10^{2}	3.10×10^{3}
	σ	6.78×10^{-3}	6.00×10^{-4}	3.29×10^{-3}	1.41×10^{-2}	7.21×10^{-2}	1.76×10^{0}	1.11×10^{1}
	Before D_{reg}	2.30×10^{-2}	1.96×10^{-1}	7.25×10^{-1}	6.46×10^{0}	2.61×10^{1}	2.55×10^{2}	1.14×10^{3}
	After D_{reg}	1.64×10^{-1}	5.08×10^{-1}	2.74×10^{0}	1.10×10^{1}	7.68×10^{1}	3.59×10^{2}	1.95×10^{3}
SD5 followed by SD1 with $\mid C_i \mid = 512$								
	Average	1.39×10^{-1}	6.25×10^{-1}	3.33×10^{0}	1.97×10^{1}	1.09×10^{2}	8.09×10^{2}	4.26×10^{3}
	σ	6.64×10^{-3}	2.61×10^{-3}	1.68×10^{-2}	8.88×10^{-3}	5.87×10^{-2}	9.52×10^{-1}	6.62×10^{0}
	Before D_{reg}	2.49×10^{-2}	2.20×10^{-1}	7.96×10^{-1}	7.44×10^{0}	3.04×10^{1}	3.14×10^{2}	1.42×10^{3}
	After D_{reg}	1.10×10^{-1}	4.00×10^{-1}	2.53×10^{0}	1.23×10^{1}	7.83×10^{1}	4.95×10^{2}	2.84×10^{3}

σ stands for a standard deviation.

Table 3.2: Benchmark Results for MQ Problems over $\mathbb{F}_{31}$ for $m = n + 1$, Total CPU Time (sec)

(n, m)		$(9, 10)$	$(10, 11)$	$(11, 12)$	$(12, 13)$	$(13, 14)$	$(14, 15)$	$(15, 16)$	$(16, 17)$
Original OpenF4									
	Average	1.22×10^{-1}	5.89×10^{-1}	2.55×10^{0}	1.38×10^{1}	8.00×10^{1}	6.30×10^{2}	3.74×10^{3}	3.04×10^{4}
	σ	2.12×10^{-3}	3.44×10^{-3}	1.01×10^{-1}	1.51×10^{0}	1.78×10^{-1}	2.32×10^{0}	2.54×10^{2}	4.32×10^{2}
	Before D_{reg}	2.82×10^{-2}	2.04×10^{-1}	5.75×10^{-1}	4.39×10^{0}	1.48×10^{1}	1.88×10^{2}	7.28×10^{2}	8.46×10^{3}
	After D_{reg}	8.98×10^{-2}	3.81×10^{-1}	1.97×10^{0}	9.40×10^{0}	6.52×10^{1}	4.42×10^{2}	3.01×10^{3}	2.20×10^{4}
SD1									
$\lvert C_i \rvert = 128$	Average	7.07×10^{-2}	3.12×10^{-1}	1.38×10^{0}	6.88×10^{0}	3.23×10^{1}	2.01×10^{2}	1.32×10^{3}	1.17×10^{4}
	σ	1.00×10^{-3}	1.85×10^{-3}	2.41×10^{-2}	1.42×10^{-1}	2.70×10^{-1}	5.46×10^{-1}	5.33×10^{0}	6.83×10^{1}
	Before D_{reg}	2.11×10^{-2}	1.55×10^{-1}	4.25×10^{-1}	3.83×10^{0}	1.34×10^{1}	1.38×10^{2}	6.09×10^{2}	8.69×10^{3}
	After D_{reg}	4.60×10^{-2}	1.51×10^{-1}	9.46×10^{-1}	3.03×10^{0}	1.89×10^{1}	6.27×10^{1}	7.11×10^{2}	3.02×10^{3}
$\lvert C_i \rvert = 256$	Average	9.17×10^{-2}	3.03×10^{-1}	1.24×10^{0}	6.30×10^{0}	2.79×10^{1}	1.66×10^{2}	9.76×10^{2}	7.81×10^{3}
	σ	9.00×10^{-4}	2.42×10^{-3}	5.33×10^{-3}	1.31×10^{-1}	7.61×10^{-2}	2.76×10^{-1}	1.54×10^{0}	2.45×10^{1}
	Before D_{reg}	2.73×10^{-2}	1.39×10^{-1}	3.43×10^{-1}	2.79×10^{0}	1.03×10^{1}	9.01×10^{1}	4.02×10^{2}	5.35×10^{3}
	After D_{reg}	5.93×10^{-2}	1.60×10^{-1}	8.85×10^{-1}	3.50×10^{0}	1.76×10^{1}	7.61×10^{1}	5.74×10^{2}	2.47×10^{3}
$\lvert C_i \rvert = 512$	Average	1.12×10^{-1}	3.83×10^{-1}	1.35×10^{0}	5.52×10^{0}	2.90×10^{1}	1.52×10^{2}	7.97×10^{2}	5.39×10^{3}
	σ	2.09×10^{-3}	2.24×10^{-3}	2.59×10^{-2}	2.65×10^{-1}	1.03×10^{-2}	1.36×10^{-1}	1.98×10^{0}	1.70×10^{1}
	Before D_{reg}	2.74×10^{-2}	1.72×10^{-1}	4.30×10^{-1}	2.06×10^{0}	8.69×10^{0}	7.10×10^{1}	2.99×10^{2}	3.70×10^{3}
	After D_{reg}	8.08×10^{-2}	2.05×10^{-1}	9.12×10^{-1}	3.45×10^{0}	2.03×10^{1}	8.12×10^{1}	4.99×10^{2}	1.69×10^{3}
$\lvert C_i \rvert = 768$	Average	1.18×10^{-1}	4.57×10^{-1}	1.56×10^{0}	5.91×10^{0}	2.72×10^{1}	1.52×10^{2}	8.02×10^{2}	5.12×10^{3}
	σ	2.18×10^{-3}	1.85×10^{-3}	2.20×10^{-2}	1.89×10^{-1}	9.46×10^{-3}	2.13×10^{0}	1.31×10^{0}	5.24×10^{1}
	Before D_{reg}	2.75×10^{-2}	1.99×10^{-1}	4.91×10^{-1}	2.41×10^{0}	7.07×10^{0}	6.51×10^{1}	2.67×10^{2}	3.25×10^{3}
	After D_{reg}	8.59×10^{-2}	2.53×10^{-1}	1.06×10^{0}	3.49×10^{0}	2.01×10^{1}	8.67×10^{1}	5.35×10^{2}	1.86×10^{3}
$\lvert C_i \rvert = 1024$	Average	1.19×10^{-1}	5.17×10^{-1}	1.81×10^{0}	6.70×10^{0}	2.66×10^{1}	1.49×10^{2}	8.68×10^{2}	5.05×10^{3}
	σ	1.80×10^{-3}	2.37×10^{-3}	1.75×10^{-2}	1.40×10^{-1}	4.72×10^{-2}	1.54×10^{0}	1.77×10^{0}	3.06×10^{1}
	Before D_{reg}	2.72×10^{-2}	2.00×10^{-1}	5.60×10^{-1}	2.75×10^{0}	7.79×10^{0}	6.28×10^{1}	2.45×10^{2}	2.75×10^{3}
	After D_{reg}	8.72×10^{-2}	3.13×10^{-1}	1.24×10^{0}	3.95×10^{0}	1.88×10^{1}	8.61×10^{1}	6.22×10^{2}	2.30×10^{3}
$\lvert C_i \rvert = 2048$	Average	1.18×10^{-1}	5.79×10^{-1}	2.34×10^{0}	8.98×10^{0}	3.42×10^{1}	1.56×10^{2}	8.63×10^{2}	5.20×10^{3}
	σ	1.68×10^{-3}	1.83×10^{-3}	3.16×10^{-2}	1.43×10^{-1}	2.05×10^{-2}	4.88×10^{-1}	1.15×10^{1}	2.37×10^{1}
	Before D_{reg}	2.73×10^{-2}	1.99×10^{-1}	5.79×10^{-1}	3.71×10^{0}	1.07×10^{1}	6.16×10^{1}	2.14×10^{2}	2.46×10^{3}
	After D_{reg}	8.66×10^{-2}	3.76×10^{-1}	1.77×10^{0}	5.26×10^{0}	2.34×10^{1}	9.39×10^{1}	6.48×10^{2}	2.73×10^{3}
$\lvert C_i \rvert = 4096$	Average	1.19×10^{-1}	5.81×10^{-1}	2.54×10^{0}	1.23×10^{1}	5.27×10^{1}	2.16×10^{2}	1.05×10^{3}	6.09×10^{3}
	σ	1.63×10^{-3}	2.54×10^{-3}	9.76×10^{-2}	9.86×10^{-1}	1.15×10^{-1}	8.77×10^{-1}	1.48×10^{1}	4.83×10^{1}
	Before D_{reg}	2.72×10^{-2}	2.00×10^{-1}	5.69×10^{-1}	4.36×10^{0}	1.47×10^{1}	9.12×10^{1}	3.10×10^{2}	2.44×10^{3}
	After D_{reg}	8.72×10^{-2}	3.76×10^{-1}	1.96×10^{0}	7.91×10^{0}	3.80×10^{1}	1.25×10^{2}	7.39×10^{2}	3.65×10^{3}
SD2									
$k = 5$	Average	7.91×10^{-2}	3.01×10^{-1}	1.32×10^{0}	6.13×10^{0}	2.88×10^{1}	1.83×10^{2}	1.22×10^{3}	8.53×10^{3}
	σ	2.12×10^{-3}	1.99×10^{-3}	1.35×10^{-1}	3.98×10^{-1}	2.44×10^{-2}	7.03×10^{-2}	5.35×10^{2}	2.30×10^{2}
	Before D_{reg}	2.64×10^{-2}	1.32×10^{-1}	3.62×10^{-1}	2.07×10^{0}	6.79×10^{0}	5.97×10^{1}	2.22×10^{2}	2.42×10^{3}
	After D_{reg}	4.51×10^{-1}	1.60×10^{-1}	9.22×10^{-1}	4.03×10^{0}	2.20×10^{1}	1.23×10^{2}	1.00×10^{3}	6.11×10^{3}
$k = 10$	Average	1.17×10^{-1}	3.38×10^{-1}	1.37×10^{0}	6.67×10^{0}	2.67×10^{1}	1.65×10^{2}	9.63×10^{2}	7.10×10^{3}
	σ	1.08×10^{-3}	2.05×10^{-3}	8.76×10^{-3}	1.09×10^{0}	2.97×10^{-1}	2.45×10^{-1}	9.38×10^{-1}	9.41×10^{2}
	Before D_{reg}	3.16×10^{-2}	1.74×10^{-1}	4.93×10^{-1}	2.89×10^{0}	9.10×10^{0}	7.15×10^{1}	2.74×10^{2}	2.58×10^{3}
	After D_{reg}	3.94×10^{-2}	1.48×10^{-1}	8.51×10^{-1}	3.72×10^{0}	1.76×10^{1}	9.38×10^{1}	6.89×10^{2}	4.51×10^{3}
$k = 15$	Average	1.13×10^{-1}	3.68×10^{-1}	1.57×10^{0}	7.05×10^{0}	4.29×10^{1}	1.56×10^{2}	8.51×10^{2}	5.73×10^{3}
	σ	8.72×10^{-4}	1.55×10^{-3}	1.29×10^{-2}	5.16×10^{-1}	1.03×10^{-1}	8.86×10^{-2}	1.63×10^{0}	7.39×10^{0}
	Before D_{reg}	3.41×10^{-2}	1.93×10^{-1}	6.35×10^{-1}	3.64×10^{0}	1.13×10^{1}	7.28×10^{1}	2.58×10^{2}	2.28×10^{3}
	After D_{reg}	4.03×10^{-2}	1.47×10^{-1}	8.85×10^{-1}	3.32×10^{0}	3.10×10^{1}	8.36×10^{1}	5.92×10^{2}	3.45×10^{3}
SD3									
$r = 1/5$	Average	8.05×10^{-2}	2.99×10^{-1}	1.25×10^{0}	5.89×10^{0}	2.81×10^{1}	1.80×10^{2}	1.11×10^{3}	8.39×10^{3}
	σ	1.02×10^{-3}	1.61×10^{-3}	4.56×10^{-3}	4.38×10^{-2}	3.32×10^{-2}	1.56×10^{-1}	5.19×10^{1}	6.57×10^{1}
	Before D_{reg}	2.54×10^{-2}	1.30×10^{-1}	3.56×10^{-1}	2.03×10^{0}	6.74×10^{0}	5.89×10^{1}	2.39×10^{2}	2.41×10^{3}
	After D_{reg}	4.41×10^{-2}	1.57×10^{-1}	8.77×10^{-1}	3.84×10^{0}	2.13×10^{1}	1.21×10^{2}	8.75×10^{2}	5.97×10^{3}
$r = 1/10$	Average	9.70×10^{-2}	3.34×10^{-1}	1.44×10^{0}	6.54×10^{0}	2.85×10^{1}	1.63×10^{2}	9.78×10^{2}	6.65×10^{3}
	σ	8.94×10^{-4}	1.73×10^{-3}	6.32×10^{-3}	9.04×10^{-2}	1.42×10^{-2}	1.84×10^{0}	9.16×10^{0}	5.87×10^{1}
	Before D_{reg}	3.32×10^{-2}	1.73×10^{-1}	4.85×10^{-1}	3.19×10^{0}	8.94×10^{0}	7.07×10^{1}	2.69×10^{2}	2.52×10^{3}
	After D_{reg}	5.08×10^{-2}	1.44×10^{-1}	9.27×10^{-1}	3.32×10^{0}	1.96×10^{1}	9.22×10^{1}	7.10×10^{2}	4.13×10^{3}
$r = 1/15$	Average	1.08×10^{-1}	3.84×10^{-1}	1.68×10^{0}	7.02×10^{0}	3.07×10^{1}	1.56×10^{2}	9.08×10^{2}	5.71×10^{3}
	σ	6.00×10^{-4}	1.86×10^{-3}	7.40×10^{-3}	9.72×10^{-2}	2.96×10^{-2}	2.80×10^{-1}	1.67×10^{0}	1.23×10^{1}
	Before D_{reg}	3.92×10^{-2}	2.10×10^{-1}	6.33×10^{-1}	3.97×10^{0}	1.12×10^{1}	7.22×10^{1}	2.59×10^{2}	2.25×10^{3}
	After D_{reg}	4.70×10^{-2}	1.51×10^{-1}	1.02×10^{0}	3.01×10^{0}	1.94×10^{1}	8.39×10^{1}	6.49×10^{2}	3.46×10^{3}
SD3+SD1 $r = 1/15$, $\lvert C_i \rvert = 512$									
	Average	1.36×10^{-1}	4.27×10^{-1}	1.56×10^{0}	7.22×10^{0}	3.12×10^{1}	1.52×10^{2}	7.52×10^{2}	3.92×10^{3}
	σ	1.08×10^{-3}	1.64×10^{-3}	2.84×10^{-2}	5.30×10^{-2}	2.87×10^{-2}	9.05×10^{-2}	1.10×10^{0}	1.23×10^{1}
	Before D_{reg}	3.90×10^{-2}	2.06×10^{-1}	6.19×10^{-1}	3.88×10^{0}	1.09×10^{1}	7.04×10^{1}	2.55×10^{2}	2.22×10^{3}
	After D_{reg}	8.08×10^{-2}	2.02×10^{-1}	9.11×10^{-1}	3.30×10^{0}	2.03×10^{1}	8.10×10^{1}	4.96×10^{2}	1.70×10^{3}
SD4+SD1 $\lvert C_i \rvert = 512$									
	Average	1.03×10^{-1}	3.21×10^{-1}	1.26×10^{0}	5.38×10^{0}	2.73×10^{1}	1.33×10^{2}	7.11×10^{2}	3.78×10^{3}
	σ	1.02×10^{-3}	8.00×10^{-4}	6.52×10^{-2}	2.76×10^{-1}	2.27×10^{-2}	1.56×10^{-1}	7.46×10^{0}	5.81×10^{2}
	Before D_{reg}	1.73×10^{-2}	1.07×10^{-1}	3.20×10^{-1}	1.83×10^{0}	6.08×10^{0}	4.89×10^{1}	1.95×10^{2}	2.00×10^{3}
	After D_{reg}	8.19×10^{-2}	2.08×10^{-1}	9.38×10^{-1}	3.54×10^{0}	2.12×10^{1}	8.43×10^{1}	5.16×10^{2}	1.78×10^{3}
SD5+SD1 $\lvert C_i \rvert = 512$									
	Average	8.34×10^{-2}	2.93×10^{-1}	1.22×10^{0}	5.69×10^{0}	2.63×10^{1}	1.53×10^{2}	8.09×10^{2}	5.89×10^{3}
	σ	9.14×10^{-4}	1.10×10^{-3}	2.47×10^{-2}	2.03×10^{-1}	1.06×10^{-2}	2.30×10^{-1}	1.34×10^{0}	5.16×10^{0}
	Before D_{reg}	1.79×10^{-2}	1.11×10^{-1}	3.20×10^{-1}	1.96×10^{0}	6.60×10^{0}	5.49×10^{1}	2.27×10^{2}	2.22×10^{3}
	After D_{reg}	6.22×10^{-2}	1.76×10^{-1}	8.95×10^{-1}	3.72×10^{0}	1.97×10^{1}	9.83×10^{1}	5.81×10^{2}	3.67×10^{3}

σ stands for a standard deviation.

Table 3.3: Experimental Results of Minimum Number of Critical Pairs for a Reduction to Zero in the First Half for $m = n + 1$ over $\mathbb{F}_{256}$ and $\mathbb{F}_{31}$

(n, m)		$(9, 10)$	$(10, 11)$	$(11, 12)$	$(12, 13)$	$(13, 14)$	$(14, 15)$	$(15, 16)$	$(16, 17)$	$(17, 18)^{\dagger}$	$(18, 19)^{\dagger}$
$d = 2$	Min	9	10	11	12	13	14	15	16	17	18
	Total	9	10	11	12	13	14	15	16	17	18
$d = 3$	Min	20	24	28	32	36	40	45	50	55	60
	Total	20	24	28	32	36	40	45	50	55	60
$d = 4$	Min	39	50	60	76	91	106	126	146	165	189
	Total	99	126	153	187	221	256	301	347	393	445
$d = 5$	Min	63	88	120	156	204	248	318	378	462	550
	Total	259	354	456	604	764	927	1158	1386	1638	1942
$d = 6$	Min	–	132	187	286	364	532	664	901	1089	1424
	Total		737	1059	1432	2004	2612	3449	4331	5443	6780
$d = 7$	Min	–	–	–	429	572	936	1300	1768	2448	3078
	Total				3003	4004	6216	8164	11492	14616	19614
$d = 8$	Min	–	–	–	–	–	1430	2002	3094	4590	5814
	Total						11804	17108	24480	35496	45999
$d = 9$	Min	–	–	–	–	–	–	–	4862	7072	10336
	Total								45526	70176	93024
$d = 10$	Min	–	–	–	–	–	–	–	–	–	16796
	Total										173774

Min stands for a minimum number of critical pairs for a reduction to zero.
Total stands for the total number of critical pairs appearing before a reduction.
† : Another NUMA machine that has 3TB RAM was used.

of the experiments we conducted. It was not possible to estimate the number of critical pairs that generate a reduction to zero for the first time under the condition where $n > 15$ over $\mathbb{F}_{256}$ or $n > 16$ over $\mathbb{F}_{31}$ due to the limitations of our machine and the OpenF4 library. Further research is required to investigate the number of critical pairs that generate a reduction to zero. Instead, we developed **SD5** in the first half of the computation by approximating the ratio Min to Total specified in Table 3.3. It is possible to apply **SD5** to the condition where $n > 15$ over $\mathbb{F}_{256}$ or $n > 16$ over $\mathbb{F}_{31}$. It should be noted that **SD4** can be applied once a phenomenon like Table 3.3 is identified in a given condition. Future work will mainly cover the investigation of the mechanism to generate a reduction to zero.

3.2 Security of the Patched CBC Mode

3.2.1 Patched CBC

To fix security flaws of the CBC mode, some software patches for the Weak-TLS1.0 described in Section 2.10.2 have been released by browser vendors. For example, a countermeasure to prepend an empty plaintext record before sending the actual plaintext records It is no longer used since it is insufficient for practical use because of the lack of interconnectivity [48]. Currently, the software patch named $1/n-1$ *record splitting patch*[67] is widely used, which is implemented as described in Table 3.4 and Figure 3.5.

We name the authenticated encryption scheme described in Table 3.4 as SplTLS1.0. For decryption, the algorithm outputs plaintexts using $\mathcal{D}_{\texttt{WeakTLS1.0}}$ multiple times.

Algorithm $\mathcal{K}_{\texttt{SplTLS1.0}}$	**Algorithm** $\mathcal{E}_{\texttt{SplTLS1.0}}(K, M; \texttt{st})$
$K \xleftarrow{\$} \mathcal{K}_{\texttt{WeakTLS}}$ Output K	$(C_0, \texttt{st}) \leftarrow \mathcal{E}_{\texttt{WeakTLS1.0}}(K, M[0][0]; \texttt{st})$ If M is one byte then output $(C_0, \texttt{st})$ else $(C_1, \texttt{st}) \leftarrow \mathcal{E}_{\texttt{WeakTLS1.0}}(K, M[0][1..]; \texttt{st})$ and output (C_0, C_1) and $\texttt{st}$

Table 3.4: Patched CBC (SplTLS1.0)

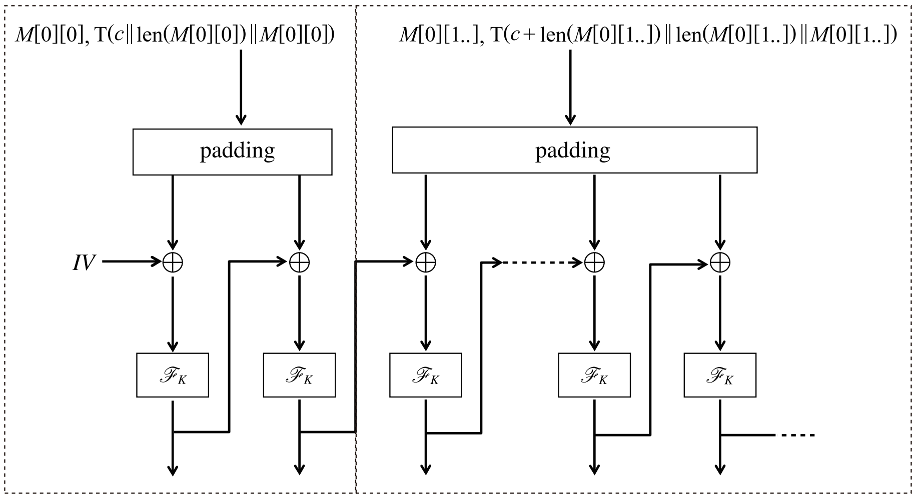

$M[0][0]$, T($c \| \mathrm{len}(M[0][0]) \| M[0][0]$)
$M[0][1..]$, T($c + \mathrm{len}(M[0][1..]) \| \mathrm{len}(M[0][1..]) \| M[0][1..]$)
padding
padding
IV
$\mathcal{F}_K$
$\mathcal{F}_K$
$\mathcal{F}_K$
$\mathcal{F}_K$
$\mathcal{F}_K$

Figure 3.5: Patched CBC: $1/n-1$ Record Splitting Patch Applied WeakTLS1.0 (SplTLS1.0)

In SplTLS1.0, the encryption algorithm for WeakTLS1.0 is invoked two times to encrypt the message M. For the first time, the first byte of the message $M[0][0]$ is encrypted, and for the second time, the remaining message $M[0][1..]$ is encrypted. The security proof of SplTLS1.0 is given as follows.

Theorem 3.2.1 ([42])**.** *If $\mathcal{P}$ is PRP, and $\mathcal{MA}$ is (complete) PRF, then SplTLS1.0 satisfies* $(\epsilon_{\text{IND}}, q)$ *IND-CPA security, where*

$$\epsilon_{\text{IND}} = 2\epsilon_{\texttt{PRF}} + 2\epsilon_{\texttt{PRP}} + \frac{q'(q'-1)}{2^{8\lambda}} + \epsilon_{\text{G4}} + \frac{q'^2}{2^{8\lambda}}.$$

And $8\lambda q'$ *is the bit-length of all the ciphertexts generated by* LR *oracle. For* ϵ_{G4}*,*

- *if* $\lambda - 1 \leq \tau$ *then,* $\epsilon_{\text{G4}} = \frac{q(q-1)}{2^{8\lambda-7}}$;
- else $\epsilon_{\text{G4}} = \frac{q(q-1)}{2^{8\tau-1}}$.

The theorem says that the indistinguishability of the patched CBC mode depends on the tag length. For example, if AES and HMAC-SHA1 are used as $\mathcal{P}$ and $\mathcal{MA}$ respectively then $\lambda = 16$, $\tau = 20$ and hence $\epsilon_{\text{G4}} = q(q-1)/2^{121}$. If the truncated HMAC defined in RFC 6066 [26] is used instead then $\tau = 10$ and hence $\epsilon_{\text{G4}} = q(q-1)/2^{79}$. $\texttt{Adv}_{\text{IND}}$ of SplTLS1.0 may increase in this case.

$\mathcal{SE}_{\texttt{WeakTLS1.0}}$ is not secure against BEAST attacks, as stated in Section 1.5. Now, we pose the question, can we apply BEAST type of attack to the patched CBC mode? We will take a look at a simple example as follows. We will encrypt a plaintext

$$P = r_1||r_2||m \quad \text{where } |r_1| = |m| = 1,\ |r_2| = \lambda - 2$$

by SplTLS1.0 (Figure 3.6).

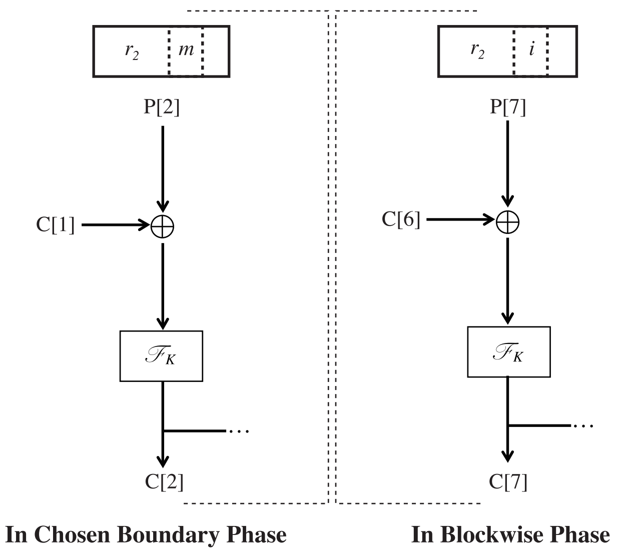


Figure 3.6: An Simple Example of the Record Splitting

At the first block of the second record in the chosen boundary phase, we have

$$C[1] \oplus \mathcal{F}_K^{-1}(C[2]) = P[2] = r_2||m||(\text{a leftmost byte of a tag in P[2]}).$$

Let us suppose that an attacker makes a target encrypt forged blocks $r_1||r_2||i$ (where $i = 0, \ldots, 255$) in the blockwise phase. So, at the first block of the second record in the blockwise phase, we have

$$C[6] \oplus \mathcal{F}_K^{-1}(C[7]) = P[7] = r_2||i||(\text{a leftmost byte of a tag in P[7]}).$$

In this example, we will find the following observations because of the counter c in the tag.

O1. $C[6]$ varies whenever $C[7]$ is computed in the blockwise phase.

O2. Both $P[2]$ and $P[7]$ contain leftmost bytes of different tags respectively.

Because the comparison between $C[2]$ and $C[7]$ is not useful, the patched CBC mode completely defeats all attacker's attempts in the blockwise phase, as was presented in Section 1.5. Therefore, they indicate that it is hard to apply BEAST type of attack to the patched CBC mode without enough oracle access for both $\mathcal{P}$ and $\mathcal{MA}$ and then we can not distinguish plaintexts by these ciphertexts.

3.2.2 Security Proof of Theorem 3.2.1

We define a sequence of games and prove its IND-CPA security. In game G_i, the probability of the adversary A outputting a bit 1 is described by

$$\Pr[A \Rightarrow 1 \mid G_i].$$

Game G_0: In this game, we set $b = 0$ in the definition of IND-CPA. Therefore,

$$\Pr[A \Rightarrow 1 \mid G_0].$$

Game G_1: This is the same as game G_0 except that the PRP $\mathcal{F}$ is replaced with a random permutation. By the definition of PRP,

$$|\Pr[A \Rightarrow 1 \mid G_0] - \Pr[A \Rightarrow 1 \mid G_1]| \leq \epsilon_{\mathtt{PRP}}.$$

Game G_2: This game is the same as game G_1 except that we replace the random permutation $\mathcal{P}$ with a random function. By the PRP/PRF switching lemma (Lemma 2.6.2),

$$|\Pr[A \Rightarrow 1 \mid G_1] - \Pr[A \Rightarrow 1 \mid G_2]| \leq \frac{q'(q'-1)}{2^{8\lambda+1}},$$

where q' is the number of queries to the random permutation. Therefore, this is the total block length of the ciphertexts.

Game G_3: This is the same as game G_2 except that we replace $\mathcal{MA}$ modeled as the PRF with a random function. Since the difference is bounded by the definition of the PRF,

$$|\Pr[A \Rightarrow 1 \mid G_2] - \Pr[A \Rightarrow 1 \mid G_3]| \leq \epsilon_{\mathtt{PRF}}.$$

Game G_4: This game is the same as game G_3 except for the following. Let M_i be the i-th message to be encrypted in LR oracle, and let c_i be its counter. We also define

$$I_i = \mathsf{Pad}(M_i[0][0] \| \mathcal{T}(c_i \| \| M_i[0][0] \| \| M_i[0][0])).$$

In this game, if there exists a pair (i, j) $(i \neq j)$ such that $I_i[0] = I_j[0]$, then LR oracle stops. Let $\texttt{Coll}_{i,j}$ be the event that there exists a pair (i, j) $(i \neq j)$ such that $I_i[0] = I_j[0]$. Then, if for every i, j $(i \neq j)$, $\texttt{Coll}_{i,j}$ does not occur; then, the probabilities that A outputs 1 in games G_3 and G_4 are the same.

Let us estimate the amount of $\Pr[\texttt{Coll}_{i,j}]$. Depending on the length of the tag τ, we consider two cases $\lambda - 1 \leq \tau$, and $\lambda - 1 > \tau$.

Case $\lambda - 1 \leq \tau$ in game G_4: Since $M[0][0]$ is 1 byte which can be controlled by the adversary, and $\lambda - 1 \leq \tau$, the input to $\mathcal{F}_K$ is

$$X = \texttt{IV} \oplus M[0][0] \| t[0][0] \| \cdots \| t[0][\lambda - 2],$$

where $t = \mathcal{T}(c_i \| \| M_i[0][0] \| \| M_i[0][0])$.

Further, since c_i is a counter, the input $c_i \| \| M_i[0][0] \| \| M_i[0][0]$ to $\mathcal{T}$ is not duplicate. Hence, $X[0][1..]$ is random since $\mathcal{T}$ is a random function. Therefore, for every i, j $(i \neq j)$, $\Pr[\texttt{Coll}_{i,j}] \leq 1/2^{8\lambda - 8}$. Taking the union bound, we have

$$|\Pr[A \Rightarrow 1 \mid G_3] - \Pr[A \Rightarrow 1 \mid G_4]| \leq \epsilon_{\mathrm{G4}},$$

where $\epsilon_{\mathrm{G4}} = \frac{q(q-1)}{2^{8\lambda - 7}}$, and q is the number of queries to LR oracle.

Case $\lambda - 1 > \tau$ in game G_4: By a similar discussion as above, we can estimate the difference as

$$|\Pr[A \Rightarrow 1 \mid G_3] - \Pr[A \Rightarrow 1 \mid G_4]| \leq \epsilon_{\mathrm{G4}},$$

where $\epsilon_{\mathrm{G4}} = \frac{q(q-1)}{2^{8\tau - 1}}$.

Game G_5: This game is the same as game G_4 except that $b = 1$. We prove that the difference in the probability that A outputs 1 in games G_4 and G_5 is

$$|\Pr[A \Rightarrow 1 \mid G_4] - \Pr[A \Rightarrow 1 \mid G_5]| \leq \frac{q'^2}{2^{8\lambda}}. \tag{3.1}$$

If the input to the random function $\mathcal{F}_K$ is not duplicate, then a bit b is information-theoretically hidden. Therefore, we estimate the probability that the input to $\mathcal{F}_K$ is duplicate. Let Bad be the event that input to the random function is duplicate. Then, the left-hand side of inequality (3.1) is bounded by $\Pr[\mathsf{Bad}]$.

The oracle LR encrypts M_0 or M_1. From the previous game, we know that the first query $I_i[0]$ to $\mathcal{F}_K$ is not duplicated. Hence, we can estimate the probability that Bad occurs as

$$\Pr[\mathsf{Bad}] \leq \frac{q+1}{2^{8\lambda}} + \frac{q+2}{2^{8\lambda}} + \cdots + \frac{q'}{2^{8\lambda}} \leq \frac{q'^2}{2^{8\lambda}}$$

Game G_6: This game is the same as game G_5 except for the following. Firstly, we replace the random function $\mathcal{T}$ in $\mathcal{MA}$ with a PRF; then, we replace a random function $\mathcal{F}_K$ with a PRP. Since this modification implies reversing the the direction of the game sequence, we have

$$\begin{aligned} &|\Pr[A \Rightarrow 1 \mid G_5] - \Pr[A \Rightarrow 1 \mid G_6]| \\ &\leq \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{PRP}} + \frac{q'(q'-1)}{2^{8\lambda+1}}. \end{aligned}$$

Since game G_0 is $b = 0$ in the game IND-CPA and game G_6 is $b = 1$ in the game IND-CPA, we have

$$\begin{aligned} &|\Pr[K \xleftarrow{\$} \mathcal{K}_{\texttt{SplTLS1.0}}, b \xleftarrow{\$} \{0,1\}, b' \xleftarrow{\$} A^{\mathsf{LR}_{K,b}(\cdot,\cdot)} \mid b = b'] - \frac{1}{2}| \\ &\qquad \leq 2\epsilon_{\mathsf{PRF}} + 2\epsilon_{\mathsf{PRP}} + \frac{q'(q'-1)}{2^{8\lambda}} + \epsilon_{\mathrm{G4}} + \frac{q'^2}{2^{8\lambda}}, \end{aligned}$$

where if $\lambda - 1 \leq \tau$, then $\epsilon_{\mathrm{G4}} = \frac{q(q-1)}{2^{8\lambda-7}}$; $\epsilon_{\mathrm{G4}} = \frac{q(q-1)}{2^{8\tau-1}}$ otherwise. This concludes the proof. $\square$

Bibliography

[1] The Certicom ECC Challenge. https://www.certicom.com/content/dam/certicom/images/pdfs/challenge-2009.pdf.

[2] The RSA Challenge Numbers. https://web.archive.org/web/20010805210445/http://www.rsa.com/rsalabs/challenges/factoring/numbers.html.

[3] TU Darmstadt Lattice Challenge. https://www.latticechallenge.org.

[4] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. On the Security of RC4 in TLS. In *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pages 305–320, 2013.

[5] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 526–540, 2013.

[6] Richard Barnes. The POODLE Attack and the End of SSL 3.0. https://blog.mozilla.org/security/2014/10/14/the-poodle-attack-and-the-end-of-ssl-3-0/, October 2014.

[7] Richard Barnes, Martin Thomson, , Alfredo Pironti, and Adam Langley. Deprecating Secure Sockets Layer Version 3.0. https://tools.ietf.org/html/rfc7568/, June 2015.

[8] Adam Barth. HTTP State Management Mechanism. RFC6265, https://www.rfc-editor.org/rfc/rfc6265, April 2011.

[9] Adam Barth. The Web Origin Concept. https://tools.ietf.org/html/rfc6454/, December 2011.

[10] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Paper 2004/331, 2004. https://eprint.iacr.org/2004/331.

[11] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, 3(3):177–197, 2009.

[12] Ward Beullens and Bart Preneel. Field lifting for smaller UOV public keys. In *Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings*, pages 227–246, 2017.

[13] Bruno Buchberger. A criterion for detecting unnecessary reductions in the construction of Gröbner bases. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 1979.

[14] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password Interception in a SSL/TLS Channel. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 583–599, 2003.

[15] A. Casanova, Jean-Charles Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. GeMSS: A great multivariate short signature. `https://www-polsys.lip6.fr/Links/NIST/GeMSS_specification.pdf`.

[16] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. From 5-pass $\mathcal{MQ}$-based identification to $\mathcal{MQ}$-based signatures. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 135–165, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[17] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 392–407, 2000.

[18] Nicolas T. Courtois. The security of hidden field equations (hfe). In David Naccache, editor, *Topics in Cryptology — CT-RSA 2001*, pages 266–281, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[19] Tim Dierks and Christopher Allen. The TLS Protocol Version 1.0. RFC2246, `https://www.rfc-editor.org/info/rfc2246`, January 1999.

[20] Tim Dierks and Eric Rescorla. The TLS Protocol Version 1.1. RFC4346, `https://www.rfc-editor.org/info/rfc4346`, April 2006.

[21] Tim Dierks and Eric Rescorla. The TLS Protocol Version 1.2. RFC5246, `https://www.rfc-editor.org/info/rfc5246`, August 2008.

[22] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[23] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 164–175, 2005.

[24] Thai Duong and Juliano Rizzo. Here Come The $\oplus$ Ninjas. `http://netifera.com/research/beast/beast_DRAFT_0621.pdf`, May 2011.

[25] Morris Dworkin. NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation, Methods and Techniques. `http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf`, December 2001.

[26] Donald Eastlake. Transport Layer Security (TLS) Extensions: Extension Definitions. `http://tools.ietf.org/html/rfc6066`, January 2011.

[27] Jean-Charles Faugère. A new efficient algorithm for computing gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.

[28] Jean Charles Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f_5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, pages 75–83, New York, NY, USA, 2002. Association for Computing Machinery.

[29] Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *J. Symb. Comput.*, 16(4):329–344, 1993.

[30] Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 44–60, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[31] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC6101, `https://www.rfc-editor.org/rfc/rfc6101`, August 2011.

[32] Rüdiger Gebauer and H. Michael Möller. On an installation of buchberger's algorithm. *Journal of Symbolic Computation*, 6(2):275–286, 1988.

[33] Peter Gutmann. Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). `https://tools.ietf.org/html/rfc7366/`, September 2014.

[34] Takanori Isobe, Toshihiro Ohigashi, Yuhei Watanabe, and Masakatu Morii. Full Plaintext Recovery Attack on Broadcast RC4. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pages 179–202, 2013.

[35] Takanori Isobe, Toshihiro Ohigashi, Yuhei Watanabe, and Masakatu Morii. Comprehensive Analysis of Initial Keystream Biases of RC4. *IEICE Transactions*, 97-A(1):139–151, 2014.

[36] Takuma Ito, Naoyuki Shinohara, and Shigenori Uchiyama. Solving the mq problem using gröbner basis techniques. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E104.A(1):135–142, 01 2021.

[37] Antoine Joux, Vanessa Vitse, and Titouan Coladon. Openf4: F4 algorithm c++ library (gröbner basis computations over finite fields). `https://github.com/nauotit/openf4`, 2015.

[38] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 206–222, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[39] Aviad Kipnis and Adi Shamir. Cryptanalysis of the oil and vinegar signature scheme. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 257–266, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[40] Aviad Kipnis and Adi Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 19–30, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[41] Takashi Kurokawa, Takuma Ito, Naoyuki Shinohara, Akihiro Yamamura, and Shigenori Uchiyama. Selection Strategy of F4-Style Algorithm to Solve MQ Problems Related to MPKC. *Cryptography*, 7(1):10, Feb 2023.

[42] Takashi Kurokawa, Ryo Nojima, and Shiho Moriai. On the security of CBC mode in SSL3.0 and TLS1.0. *J. Internet Serv. Inf. Secur.*, 6(1):2–19, 2016.

[43] Adam Langley. An update on SSLv3 in Chrome. `https://groups.google.com/a/chromium.org/forum/#!topic/security-dev/Vnhy9aKM_l4`, October 2014.

[44] Adam Langley. The POODLE bites again (08 Dec 2014). `https://www.imperialviolet.org/2014/12/08/poodleagain.html`, December 2014.

[45] Itsik Mantin and Adi Shamir. A Practical Attack on Broadcast RC4. In *Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers*, pages 152–164, 2001.

[46] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and Christoph G. Günther, editors, *Advances in Cryptology — EUROCRYPT '88*, pages 419–453, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[47] Microsoft Security Response Center (MSRC). Security Advisory 3009008 updated. `http://blogs.technet.com/b/msrc/archive/2014/10/29/security-advisory-3009008-released.aspx`, October 2014.

[48] Bodo Möller. Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures. `http://www.openssl.org/~bodo/tls-cbc.txt`, May 2004.

[49] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. This POODLE Bites: Exploiting The SSL 3.0 Fallback. `https://www.openssl.org/~bodo/ssl-poodle.pdf`, September 2014.

[50] Kathleen Moriarty, Burt Kaliski, Jakob Jonsson, and Andreas Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, `https://www.rfc-editor.org/info/rfc8017`, November 2016.

[51] National Institute of Standards and Technology. Post-quantum cryptography. `https://csrc.nist.gov/projects/post-quantum-cryptography`.

[52] National Institute of Standards and Technology. Federal Information Processing Standards Publication 197, Specification for the ADVANCED ENCRYPTION STANDARD (AES). `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`, November 2001.

[53] National Institute of Standards and Technology. Federal Information Processing Standards Publication 198-1, The Keyed-Hash Message Authentication Code (HMAC). `http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf`, July 2008.

[54] National Institute of Standards and Technology. New call for proposals: Call for additional: Digital signature schemes for the post-quantum cryptography standardization process. `https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals`, August 2022.

[55] J. Patarin. The oil and vinegar signature scheme. Presented at the Dagstuhl Workshop on Cryptography, September 1997. transparencies.

[56] Jacques Patarin. Cryptanalysis of the matsumoto and imai public key scheme of eurocrypt'88. In Don Coppersmith, editor, *Advances in Cryptology — CRYPT0' 95*, pages 248–261, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[57] Jacques Patarin. Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 33–48, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[58] Jacques Patarin, Louis Goubin, and Nicolas Courtois. Improved algorithms for isomorphisms of polynomials. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 184–200, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[59] Andrei Popov. Prohibiting RC4 Cipher Suites. RFC7465, `https://www.rfc-editor.org/rfc/rfc7465`, February 2015.

[60] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, `https://www.rfc-editor.org/info/rfc8446`, August 2018.

[61] Juliano Rizzo and Thai Duong. The CRIME attack. Ekoparty 2012, `https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/`, September 2012.

[62] Phillip Rogaway. Problems with Proposed IP Cryptography. `http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt`, April 1995.

[63] Pratik G. Sarkar and Shawn Fitzgerald. ATTACKS ON SSL - A Comprehensive Study of BEAST, CRIME, TIME, BREACH, LUCKY 13 & RC4 BIASES. `https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/ssl_attacks_survey.pdf`, August 2013.

[64] Bruce Schneier. *Applied Cryptography - Protocols, Algorithms, and Source Code in C (2nd. edition).* Wiley, 1996.

[65] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[66] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Paper 2004/332, 2004. `https://eprint.iacr.org/2004/332`.

[67] Xuelei Su. Bugzilla Bug 665814 Comment 59. `https://bugzilla.mozilla.org/show_bug.cgi?id=665814#c59`, July 2011.

[68] Trustworthy Internet Movement. SSL Pulse - Survey of the SSL Implementation of the Most Popular Web Sites. `https://www.trustworthyinternet.org/ssl-pulse/`.

[69] Serge Vaudenay. Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ... In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 534–546, 2002.

[70] Takanori Yasuda, Xavier Dahan, Yun-Ju Huang, Tsuyoshi Takagi, and Kouichi Sakurai. MQ challenge: Hardness evaluation of solving multivariate quadratic problems. *IACR Cryptology ePrint Archive*, 2015:275, 2015.

[71] Takanori Yasuda, Xavier Dahan, Yun-Ju Huang, Tsuyoshi Takagi, and Kouichi Sakurai. A multivariate quadratic challenge toward post-quantum generation cryptography. *ACM Commun. Comput. Algebra*, 49(3):105–107, 2015.