2009-09

Cube-type algebraic attacks on wireless encryption protocols

Petrakos, Nikolaos

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CUBE-TYPE ALGEBRAIC ATTACKS ON WIRELESS ENCRYPTION PROTOCOLS

by

Nikolaos Petrakos

September 2009

Thesis Co- Advisors: George Dinolt
James Bret Michael
Pantelimon Stanica

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2009	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Cube-type Algebraic Attacks on Wireless Encryption Protocols	5. FUNDING NUMBERS
6. AUTHOR(S) Nikolaos Petrakos	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

In this study, we investigated an algebraic-type attack, known as the cube attack, against wireless networks. We implemented the cube attack in a wireless system, namely Bluetooth. We formally modeled the encryption function of E0 Bluetooth key generator and automated the process of the cube attack on E0 of the factorization process (preprocessing phase). In this phase, an attacker finds as many maxterms (a term of the encryption function such that its co-factor is a linear nonconstant polynomial) as possible. In the actual attacking phase, the attacker solves the system of linear equations through a chosen plaintext attack and reveals useful information about the cryptosystem. The number of operations needed in the computational process is $2^{21.1}$ and is considerably less than that of similar algebraic types of attacks, but it is limited to the output of the LFSRs at any clock cycle. The results of our analysis indicate that if an attacker is an unauthorized participant of the security protocol, then by manipulating some of the output bits of the LFSRs of two arbitrary clock cycles and intercepting the output bits of the entire machine the attacker then succeeds in finding the output bits of the LFSRs at any clock tick.

14. SUBJECT TERMS Wireless Security, Cryptanalysis, Boolean Functions, Algebraic Attacks, Correlation Attacks, Cube Attacks, Bluetooth, Security Protocols.	15. NUMBER OF PAGES 99
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

CUBE-TYPE ALGEBRAIC ATTACKS ON WIRELESS ENCRYPTION PROTOCOLS

Nikolaos Petrakos
Lieutenant, Hellenic Navy,
B.A., Hellenic Naval Academy, 1996

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF COMPUTER SCIENCE
and
MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author:	Nikolaos Petrakos
Approved by:	George Dinolt Thesis Co-Advisor
	James Bret Michael Thesis Co-Advisor
	Pantelimon Stanica Thesis Co-Advisor
	Peter J. Denning Chairman, Department of Computer Science
	Carlos F. Borges Chairman, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this study, we investigated an algebraic-type attack, known as the cube attack, against wireless networks. We implemented the cube attack in a wireless system, namely Bluetooth. We formally modeled the encryption function of E0 Bluetooth key generator and automated the process of the cube attack on E0 of the factorization process (preprocessing phase). In this phase, an attacker finds as many maxterms (a term of the encryption function such that its co-factor is a linear nonconstant polynomial) as possible. In the actual attacking phase, the attacker solves the system of linear equations through a chosen plaintext attack and reveals useful information about the cryptosystem. The number of operations needed in the computational process is $2^{21.1}$ and is considerably less than that of similar algebraic types of attacks, but it is limited to the output of the LFSRs at any clock cycle. The results of our analysis indicate that if an attacker is an unauthorized participant of the security protocol, then by manipulating some of the output bits of the LFSRs of two arbitrary clock cycles and intercepting the output bits of the entire machine the attacker then succeeds in finding the output bits of the LFSRs at any clock tick.

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

In this study, we investigated an algebraic-type attack, known as the cube attack, against wireless networks. We implemented the cube attack in a wireless system, namely Bluetooth. We formally modeled the encryption function of the E0 Bluetooth key generator and automated the process of the cube attack on E0 of the factorization process (preprocessing phase). In this phase, an attacker finds as many maxterms (a term of the encryption function such that its co-factor is a linear nonconstant polynomial) as possible. In the actual attacking phase, the attacker solves the system of linear equations through a chosen plaintext attack and reveals useful information about the cryptosystem. The number of operations needed in the computational process is $2^{21.1}$ and is considerably less than that of similar algebraic types of attacks, but it is limited to the output of the LFSRs at any clock cycle. The main contribution of this thesis is that if the attacker is an unauthorized participant of the security protocol, then by manipulating some of the output bits of the LFSRs of two arbitrary clock cycles and intercepting the output bits of the entire machine the attacker then succeeds in finding the output bits of the LFSRs at any clock tick. The most important question that needs to be answered next is how one can recover the encryption key of E0 after knowing the output bits of every LFSR at any clock that this study provides.

Building on these results, the next stage of the research is to validate our integration of the cube-type attack into the Bluetooth encryption protocol. As demonstrated in this and other research we cited in this thesis, one needs to understand and formally evaluate the strength of a given cryptosystem, be able to evaluate its implementation to ensure that there are no flaws at that stage. The cryptosystem and the protocol it uses may be good but if poorly implemented will most likely be untrustworthy.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my thesis advisors Dr. Pantelimon Stanica, Dr. George Dinolt, and Dr. Bret James Michael for their invaluable advice, wisdom, time and effort plentifully devoted to me for this thesis. I would like also to thank Dr. David Canright for sharing with me his expertise in programming in the Maple environment. I would like further to thank all the professors of the Naval Postgraduate School who devoted time and effort to teach me with all their valuable skills and academic knowledge.

I want to thank whole-heartedly my wife Maria and my son Ioannis-Stephanos for all their love, patience, understanding, and support they have always given to me.

I would like to relay my sincere appreciation to the Hellenic Navy General Staff for affording me the opportunity to attend the Naval Postgraduate School and pursue a dual degree.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Nowadays, there is great interest from the United States Department of Defense to move from wired communication systems to wireless systems. How to secure wireless cryptosystems, which are known to have suffered malicious attacks, is a question this thesis is attempting to answer. Sun-Tzu stated (400–320 BC, translated Giles, 1910) “If you know the enemy and know yourself, you need not fear the result of a hundred battles.” As in that saying, there is a need to see and understand the mathematical theory hidden in modern types of attacks, and know how effective they are compared to the traditional exhaustive key searches in wireless security protocols (e.g., Bluetooth, Wi-Fi, Wi-Max). Bluetooth is a well-established wireless communications standard (IEEE 802.15.1) between different devices (e.g., personal computers, laptops, mobile phones) that operates over a short range and at low power. For efficiency reasons, such as speed, size and power consumption, the system uses a stream cipher encryption (E0) instead of the widely-used block ciphers. Four linear feedback shift registers[1] (LFSRs) are used in the algorithm, and a nonlinear Boolean function combines their output. The plaintext is then combined with the output key stream using an exclusive OR (XOR) producing the ciphertext. Wired Equivalency Privacy (WEP) IEEE 802.11 is another security protocol for Wi-Fi networks. It provides authentication and encryption. The key component of this protocol is the commonly used stream cipher RC4. IEEE 802.11, which has questionable functionality due to the wireless packet network structure, provides relatively weak encryption and a single-way authentication, and has no key-distribution mechanisms. IEEE 802.11i updated the previous protocol and

[1] In digital circuits, a shift register is a type of sequential logic circuit mainly for storage of digital data, set up in a linear fashion, which has its inputs connected to the outputs in such a way that the data shifts down the line when the circuit activates. A linear feedback shift register is a shift register whose input bit is the output of a linear function of two or more of its previous states (from [23], p.19).

underwent final ratification, providing much stronger forms of encryption, an extensible set of authentication mechanisms, and key distribution capabilities. It includes an Advanced Encryption Standard (AES) - based encryption scheme. World Interoperability for Microwave Access (Wi-Max) is a family of IEEE 802.16 standards that aims to deliver wireless data to a large number of users over a wide area at rates that rival those of cable modems. There are two schemes for data encryption supported in the 802.16 standard, the Advanced Encryption Standard (AES) and Triple Data Encryption Standard (3DES). Both of these schemes are block ciphers that operate on one block or chunk of data at a time, whereas stream ciphers can act on a single bit. AES handles a 128-bit block of data at a time, and has been shown to be very fast and easy to implement.

This thesis will investigate from a theoretical perspective the effectiveness of several promising attacks against linear shift feedback registers (LSFRs)-based ciphers, precisely we will look at correlation, algebraic, and cube attacks implemented in Bluetooth encryption (128-bit key size).

Correlation attacks deal with distinguishing and recovering keys against mainly stream ciphers. That means that there is a statistically biased relation between the produced keystream and the output of certain LFSR sequences. Using the notion of correlation, there is a direct relation between the output state of an individual LFSR in the keystream generator and the output of the Boolean function that combines the output state of all LFSRs. Therefore, partial knowledge of the keystream (derived from the partial knowledge of the plaintext) is needed. In 2004, Lu and Vaudenay used a correlation attack and implemented it on an E0 Bluetooth keystream generator by applying a novel maximum decoding algorithm based on the Walsh transform (a feature of the Boolean functions), and succeeded in having key recovery of 2^{39} operations after 2^{37} operations for precomputation [1]. One year later, Lu, Meier and Vaudenay proposed the use of conditional correlation attacks. The term “conditional correlation” describes the linear correlation of the inputs conditioned on a given sort output pattern of a nonlinear function with small input size. Their attack

implemented in output of the same key generator E0 of Bluetooth and disclosed the encrypted key in 2^{38} operations using the first 24 bits of $2^{23.8}$ frames, thus improving the previous results of two of them [2]. One can also use algebraic attacks against LFSR-based stream ciphers. Algebraic attacks consist of expressing the whole cipher as a large system of multivariate algebraic equations that can be solved to recover the secret key. The unknowns in these equations occasionally represent the bits of the secret key. A major parameter that influences the complexity of such attacks is the degree of the underlying algebraic system. When the transition is linear, any keystream bit can be expressed as a function of degree $\deg(f)$ in the initial state bits. However, despite the high degree of the filtering Boolean function that is used in the keystream generator, such an attack can be applied as soon as there are relations of low degree between the output and the inputs of the Boolean function. Armknecht proposed a scheme that solved the E0 cryptosystem in $2^{54.51}$ operations [3].

Dinur and Shamir described a type of algebraic attack called the cube attack [4]. The active assault on a cryptosystem requires the attacker to extract useful information from the bit stream. By skillfully choosing some publicly settable bits, the attacker may be able to replace the polynomial that represents the encryption function by a system of linear equations. Shamir and Dinur used this approach on the Trivium cipher and recovered the encryption key in 2^{19} bit operations, which is the best result in the literature so far. Zhang et al. extended Shamir and Dinur's approach to other polynomials f from where they could find a lower degree polynomial g, so that the product fg also has a lower degree. They applied this attack on the Toyocrypt cipher with re-synchronization, breaking the stream cipher in a few milliseconds on an ordinary PC [5].

All of the above-mentioned attacks are based on the cryptographic features of Boolean functions that have been an object of study in modern cryptography for about the last thirty-five years.

B. THESIS OUTLINE

The thesis consists of seven chapters. In Chapter I, the author gives a general outline of the work, describes the motivation for this research, and defines the problem that will be investigated. In Chapter II, the author describes the mathematical background necessary for the reader to understand the material that follows, the tools the author will use (Boolean functions, security protocol of E0, etc.), and the basic definitions of cryptosystems and wireless security. In Chapter III, the author examines the correlation and algebraic attacks and their theoretical background. In Chapter IV, the author details the cube attack concept and, in Chapter V, he models the Bluetooth keystream generator E0. In Chapter VI, the author details the tool he created in order to automate the cube attack and analyzes the results. The author ends this thesis with the conclusions reached from the research and provides future recommendations.

C. THE PROBLEM

In recent years, there has been great interest from the Department of Defense on substituting ground-wired networks (LANs) with short-range (Bluetooth) or medium-range (Wi-Fi) wireless networks. Several types of attacks have been successful at defeating the cryptosystems used by IEEE 802.11 and 802.16 technologies, leading one to ask the question: how much trust should we place in the wireless encryption protocols?

D. ACCOMPLISHMENTS OF THIS STUDY

We formally modeled the encryption function of E0 Bluetooth key generator and automated the factorization process (preprocessing phase) of cube attack on E0. We applied the cube-type attack and reduced the search space for the output of the LFSRs of E0, a hard task since Bluetooth E0 uses a more complex encryption algorithm than the ciphers implemented so far. The main contribution of this thesis is that under the assumption that the attacker is an unauthorized participant of the security protocol, then by manipulating some of the output bits of the LFSRs of two arbitrary clock cycles and intercepting the

output bits of the entire encryption machine the attacker then succeeds in revealing the output bits of the LFSRs at any clock cycle.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. COMPUTER SCIENCE

1. Security Protocol

Definition 2.1: "A security protocol is a sequence of messages between two or more parties in which encryption is used to provide authentication or to distribute cryptographic keys for new conversations." [6]

The majority of the security protocols in computer networks are based on cryptography, which is why they are also called cryptographic protocols. In order to establish a secure communication there are a sequence of steps the participating parties must perform. These steps include the transmission of a message, possibly encrypted, participating names, cryptographic keys, random numbers, timestamps, ciphertexts and concatenation of these components. A security protocol aims to achieve certain goals upon its completion, like verifying the authenticity of the sender, ensuring the integrity of the transmitted message, protecting the confidentiality of the header and contents of the message, and providing for nonrepudiation. A security protocol is said to be flawed if it fails to achieve its claimed goals [7].

2. Wireless Security

Security is an important concern in wireless networks because the radio frequency (RF) transmissions can be monitored by malicious people. A cryptosystem is a system used to encrypt a plaintext into ciphertext and at the other end to decrypt a ciphertext into plaintext. The cryptosystem is also used to ensure the four main goals of information security: confidentiality, integrity, authenticity and norepudiation.

3. Cryptosystem

Definition 2.2: "A cryptosystem is a five-tuple (P, C, K, E, D), where the following conditions are satisfied:

1. P is a finite set of possible plaintexts.

2. C is a finite set of possible ciphertexts.

3. K is the keyspace, which is a finite set of possible keys.

4. For each $\mathbb{k} \in K$ (i.e., for each bit that belongs to the keyspace), there is an encryption rule $e_{\mathbb{k}} \in \mathrm{E}$ and a corresponding decryption rule $d_{\mathbb{k}} \in D$. Each $e_{\mathbb{k}} : P \to C$ and $d_{\mathbb{k}} : C \to P$ are functions such that $d_{\mathbb{k}}(e_{\mathbb{k}}(x)) = x$ for every plaintext element $x \in P$." [8]

The main property of all the above is the fourth property, where if a plaintext x is encrypted using an encryption key $e_{\mathbb{k}}$, the resulting ciphertext will be decrypted using a decryption key $d_{\mathbb{k}}$, revealing the original plaintext x.

For our work, we choose $P = C = \mathbb{Z}_2^m$ where *m* is the length of the plaintext to be enciphered and $\mathbb{Z}_2$ is the set of remainders when dividing integers by 2. Thus, $\mathbb{Z}_2$ has two elements $\{0,1\}$ and is called the set of integers modulo 2. $\mathbb{Z}_2[X]$ is the set of polynomials whose coefficients are integers modulo 2.

4. Wireless Threats

In common terms, a hacker is a person who legally or illegally gains access to a computer system to make changes to the system or to reveal security flaws [9, p. 379].

We consider three types of hackers. The whitehat hacker is a person that is hired from a company to find the flaws in a computer system. A blackhat hacker is a person who illegally accesses a computer system. There are also greyhat hackers, namely something in the middle, persons who access a computer system without authorization to make changes mostly for publicity purposes and to gain popularity [9, p. 393].

Some common types of attacks on wireless systems are discussed below [10]. In traffic analysis or passive eavesdropping, an adversary intercepts the traffic in a wireless local area network (WLAN). Active eavesdropping occurs

when the adversary inserts a message into the network, and from the response of the system derives useful information about the system such as response time. There is also message deletion on a network, which implies full control of the network by the attacker. Next is session hijacking, where the adversary might hijack a valid session and put authentication between legitimate users in dispute. There is also the man-in-the-middle attack, where the adversary must participate in the communication between the target parties. Before this happens, the adversary spoofs the authentication process of both parties and then breaks the connection between the two parties. The adversary pretends that he is the legitimate one of the two associated users.

The Diffie-Hellman algorithm is vulnerable to the man-in-the-middle-attack, because no authentication occurs before the two parties exchange the secret keys [11]. Finally, denial-of-service (DoS) attacks have as a goal to deny the services that the target system provides. Denial-of-service (DoS) attacks may be launched over the Internet to target routers, servers, and firewalls. This makes them rapidly use all of their resources and unable to provide further services. There are policies and enforcement mechanisms that can be put in place to guard against such attacks, but consideration of these is outside the scope of this thesis.

From a cryptanalysis point of view, the most common models of attack are as follows:

1. Ciphertext-only attack: The adversary possesses a ciphertext, possibly by intercepting traffic.

2. Known-plaintext attack: The adversary possesses a plaintext and its corresponding ciphertext.

3. Chosen-plaintext attack: The adversary has access to the encryption cipher and he can choose a plaintext and construct the corresponding ciphertext, and he can repeat this process as many times as he likes.

4. Chosen-ciphertext attack: The adversary has access to the decryption cipher and he can choose a ciphertext and construct the corresponding plaintext, and he can repeat this process as many times as he likes.

Here, the goal of the adversary is to determine the secret key that has been used by the cipher. Correlation, algebraic and cube attacks, the foundations of our results, are detailed in the following chapters.

B. MATHEMATICAL THEORY

The attack we have developed is based on several mathematical concepts. Below we provide a description of these. We assume that the reader has some familiarity with the concepts from Abstract Algebra and Boolean functions.

At a very high level, a Boolean function outputs a single bit result (0 or 1) for each possible combination of values from many Boolean variables. The algebraic environment of Boolean functions is a vector space (defined below) of dimension *n* over the binary field. The Boolean output consists of the bit values {0,1}, with “XOR” as addition and “AND” as multiplication.

1. Vector Space

A field is a set endowed with two operations, satisfying a plethora of conditions. We will use mostly the binary field $\mathbb{F}_2$ whose addition and multiplication operations are defined as follows:

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$
$$0 \cdot 0 = 0$$
$$1 \cdot 0 = 0 \cdot 1 = 0$$
$$1 \cdot 1 = 1$$

Definition 2.3: *Let $\mathbb{F}$ be an algebraic field. A vector space over $\mathbb{F}$ (or $\mathbb{F}$-vector space) consists of an abelian (commutative) group V under addition together with an operation of scalar multiplication of each element of V by each element of $\mathbb{F}$ on the left, such that for all $a,b \in \mathbb{F}$ and $\alpha,\beta \in V$ the following conditions are satisfied:*

- $a\alpha \in V$.
- $a(b\alpha) = (ab)\alpha$.
- $(a+b)\alpha = (a\alpha) + (b\alpha)$.
- $a(\alpha+\beta) = (a\alpha) + (\alpha\beta)$.
- $1\alpha = \alpha$.

The elements of V are vectors and the elements of the algebraic field F are scalars. When only one field $\mathbb{F}$ is under discussion, the reference to $\mathbb{F}$ is dropped and instead refers to a vector space [12]. Specifically, let V_n be the vector space of dimension n over the two-element field $\mathbb{F}_2$. For two vectors in V_n, say $a = (a_1,...,a_n)$ and $b = (b_1,...,b_n)$, the scalar product is defined as $a \cdot b = a_1 b_1 \oplus ... \oplus a_n b_n$, where the multiplication and the addition $\oplus$ are over $\mathbb{F}_2$ (This operation should not be confused with the direct product of vector spaces). The operation $*$ on vectors is defined by $a * b = (a_1 b_1,...,a_n b_n)$.

When one is dealing with the vector space $V_n = \mathbb{F}_2^n$ (where $\mathbb{F}_2^n = \underbrace{\mathbb{F}_2 \times \mathbb{F}_2 \times ... \times \mathbb{F}_2}_{n\text{-times}}$ represents the set of all *n*-tuples of 0's and 1's) then the following operations apply:

- **Addition**

  $(v_1, v_2, v_3, ..., v_n) \oplus (w_1, w_2, w_3, ..., w_n) = (v_1 \oplus w_1, v_2 \oplus, v_3 \oplus w_3, ..., v_n \oplus w_n)$

- **Multiplication**
  - **Scalar Multiplication**

$$(v_1, v_2, v_3, ..., v_n) \cdot (w_1, w_2, w_3, ..., w_n) = v_1 w_1 \oplus v_2 w_2 \oplus v_3 w_3 \oplus ... \oplus v_n w_n$$

- **Vector Intersection**

$$(v_1, v_2, v_3, ..., v_n) * (w_1, w_2, w_3, ..., w_n) = (v_1 w_1, v_2 w_2, v_3 w_3, ..., v_n w_n)$$

2. Vector Space and Correspondence of the Finite Field

In abstract algebra, a finite field is any field with a finite number of elements. For every prime *p* and positive integer *n* there is exactly one finite field (up to isomorphism) of order p^n. The field $GF(2^n)$ is usually referred as the Galois field of order 2^n [12, p. 300].

Definition 2.4: *A polynomial is primitive if it is the minimal polynomial of a primitive element of the finite extension field* $GF(p^n)$. *In other words, a polynomial* $P(X)$, *with coefficients in* $GF(p) = \mathbb{Z}/p\mathbb{Z}$, *is a primitive polynomial, if it has a root α in* $GF(p^n)$ *such that* $\left\{0, 1, a, a^2, a^3, ..., a^{p^n-2}\right\}$ *is the entire field* $GF(p^n)$ *and* $P(X)$ *is the smallest degree polynomial having α as root in* $GF(p^n)$.

Any finite field of dimension n over $GF(p)$ can be constructed by taking a primitive polynomial p which is of degree n (p is primitive and $\deg P(X) = n$).

For the Galois field GF(2) we have the correspondence $GF(2^n) \equiv \mathbb{F}_2^n$:

$$GF(2^n) = \frac{\mathbb{F}_2[X]}{< P >} = \left\{a_0 + a_1 X + ... + a_{n-1} X^{n-1}\right\}, a_i \in \mathbb{F}_2.$$

Given such a representation of $GF(2^n)$ by a polynomial *P*, to every element $a_0 + a_1 X + ... + a_{n-1} X^{n-1}$ we associate the vector $(a_0, a_1, ..., a_{n-1}) \in \mathbb{F}_2^n = V_n$.

This does not mean that both structures are the same; rather, it means that there is a bijective correspondence between those two structures.

Example 2.5:

Assume one is working in $GF(2^3)$, thus $GF(2^3) = \frac{\mathbb{F}_2[X]}{< P >} = \left\{0,1,...a^{2^3-1-1}\right\}$.

One has to use a primitive polynomial of degree 3, say $p = x^3 + x + 1$.

$GF(2^3)$	V_3
0	000
$1 = a^0$	001
a	010
a^2	100
$a^3 = a + 1$ **(1)**	$011 = 010 + 001$
$a^4 = a^2 + a$ **(2)**	$110 = 100 + 010$
$a^5 = a^2 + a + 1$	$111 = 100 + 010 + 001$
$a^6 = a^2 + 1$	$101 = 100 + 001$
$a^7 = 1$	001

Table 1. Correspondence between Finite Fields and Vector Spaces

Observations:

(1) $a^3 + a + 1 = 0 \Rightarrow a^3 = a + 1$ since a is a primitive element.

(2) $a^4 = a(a^3) = a(a+1) = a^2 + a$; continue in that fashion up to the element where there is repetition (a^7).

3. Boolean Function

Definition 2.6: *A Boolean function f in n variables is a map from a vector space V_n of dimension n over F_2 to the two-element field $\mathbb{F}_2$. The (0,1) sequence generated by the Boolean function f is defined by $(f(v_0), f(v_1),..., f(v_{2^n-1}))$ and is called the truth table of f, where $v_0 = (0,...,0,0), v_1 = (0,...,0,1),...v_{2^n-1} = (1,...,1,1)$, ordered lexicographical. The (1,-1) sequence of f is defined as $((-1)^{f(v_0)},(-1)^{f(v_1)},...,(-1)^{f(v_{2^n-1})})$.*

Any function that is defined in a vector space over a finite field, in particular in $\mathbb{F}_2$ is in fact a polynomial [13]. The idea is that if one defines a function that takes any vector into an output, then by taking the degree of the polynomial high enough, one can find appropriate coefficients so that particular polynomial will match the dataset.

"A Boolean function on V_n can be expressed as a polynomial in $\mathbb{F}_2[x_1,...,x_n]/(x_1^2 - x_1,...,x_n^2 - x_n)$; the algebraic normal form (ANF) is

$f(x) = \sum_{a \in V_n} c_a x_1^{a_1} \cdots x_n^{a_n}$, *where* $c_a \in \mathbb{F}_2$ *and* $a = (a_1,...,a_n)$. *Moreover,* $c_a = \sum_{x \le a} f(x)$

where $x \le a$ means that $x_i \le a_i$ for all $1 \le i \le n$. The algebra of all Boolean functions on V_n will be called B_n" [13, pp. 5–6].

The simplest Boolean functions are the constant functions 0 and 1.

Example 2.7:

Assume $n = 3$, thus working on V_3.

Let $f: V_3 \to \mathbb{F}_2 : f(x_1, x_2, x_3) = x_1 x_2 \oplus x_3$ be the Algebraic Normal Form (ANF) of a Boolean function f.

V_3 *(Lexicographical Order) Labeling of values:* $x_3x_2x_1$	f
000	**0**
001	**0**
010	**0**
011	**1**
100	**1**
101	**1**
110	**1**
111	**0**

Table 2. Truth Table of f

Thus, the Boolean function has the following truth table (Table 2): $f = 00011110$. One can infer the ANF of f having the sequence of bits of that Boolean function and vice versa.

Definition 2.8: *An affine function* $l_{a,c}$ *on* V_n *is a function that takes the form:* $l_{a,c}(x) = a \cdot x \oplus c = a_1x_1 \oplus ... \oplus a_nx_n \oplus c$,

where $a = (a_1, a_2, ..., a_n) \in V_n, c \in \mathbb{F}_2$. *If* $c = 0$, *then* $l_{a,0}(= l_a)$ *is a* linear function [13, p. 6].

Definition 2.9: *Let* A *be a set. If there are exactly* n *distinct elements in* A *where* n *is a nonnegative integer, we say that* A *is a finite set and* n *is the cardinality of* A. *The cardinality of* A *is denoted by* $|\mathrm{A}|$ [14].

Lemma 2.10: *The number of all affine functions in n variables is* $|A_n| = 2^{n+1}$**.**

Proof: By definition, an affine function depends on $n+1$ parameters $a_1, a_2, ..., a_n, c$ each of which taking values $\{0,1\}$. Therefore, the number of such choices is 2^{n+1} The set of all affine functions is a small class of Boolean functions.

Additionally, one should note that the set of all linear functions L_n *has* $|L_n| = 2^n$, since *c*=0.

■

Lemma 2.11: *The number of all Boolean functions in n variables is* $|B_n| = 2^{2^n}$.

Proof: By definition, a Boolean function f is a mapping: $f : X^{V_n = \mathbb{F}_2^n} \to Y^{\mathbb{F}_2}$.
Since the cardinality of the set for all linear functions is 2^n, the following assertion holds for all functions:
$|functions| = |Y|^{|X|} = 2^{2^n}$ and so $|B_n| = 2^{2^n}$.

■

Example 2.12:

For $n = 4,$ the number of Boolean functions is $2^{2^4} = 2^{16}$. For $n = 6,$ the number of Boolean functions is $2^{2^6} = 2^{64}$. As can be seen from these examples, the class of Boolean functions becomes extremely large. From a cryptographic point of view, one wants to count the elements of such a set because if the set is small, then one can implement an exhaustive approach and do whatever analysis one wants to do.

4. Hamming Weight and Distance

In coding theory, the Hamming distance between (two) bit strings of the same size is the number of bits where they differ. The Hamming distance is a metric and represents the minimum number of necessary substitutions to transform a bit string into another.

For example, if $f = 101001101$ and $g = 011011100$, then their Hamming distance is $d(f,g) = 4$. The Hamming weight of the string is the number of 1's it has, its distance from the 0-vector. Thus, in the previous example $wt(f) = 5, wt(g) = 5$.

The Hamming weight of a Boolean function f is the number of 1's in the truth table of f. More formally:

Definition 2.13: *The Hamming weight of a vector $x \in V_n$, denoted by $wt(x)$, is the number of 1's in the vector x. For a Boolean function on V_n, let $\Omega_f = \{x \in V_n : f(x) = 1\}$ be the support of f. The Hamming weight of a function f is the Hamming weight of its truth table, that is the cardinality of $f^{-1}(1)$ or equivalently $wt(f) = |\Omega_f|$. The Hamming distance between two functions $f, g : V_n \to \mathbb{F}_2$, denoted by $d(f,g)$ is defined as:*

$$d(f,g) = wt(f \oplus g)$$

5. Walsh Transform

The Walsh or Handamard transform is a type of discrete Fourier transform of a Boolean function. Using the Walsh transform, correlations in combining functions may be identified.

Definition 2.14: *"The Walsh transform of a function f on a vector space V_n of dimension n over F_2 (with the values of f taken to be real numbers 0 and 1) is the map $W(f) : V_n \to R$, defined by*

$$W(f)(w) = \sum_{x \in V_n} f(x)(-1)^{w \cdot x} \qquad (2.1)$$

This defines the coefficients of f *with respect to the orthogonal basis of the group characters* $Q_w(x) = (-1)^{w \cdot x}$*;* f *can be recovered by the inverse Walsh transform:*

$$f(x) = 2^{-n} \sum_{x \in V_n} W(f)(w)(-1)^{w \cdot x} \qquad (2.2)$$

The Walsh spectrum of f *is the list of* 2^n *Walsh coefficients given by (2.1) as* w *varies"* [13, p. 8].

III. CORRELATION AND ALGEBRAIC ATTACKS

A. INTRODUCTION

In recent years where communication, computer-based systems have been commonly used in both commercial and military environments, stream ciphers remain dominant since a stream cipher provides speed to the encryption process and allows synchronization between data and voice in broadband channels. Short-range (Bluetooth) and medium-range (Wi-Fi) wireless networks use stream ciphers to provide authentication and data encryption between a host and wireless access points. Bluetooth uses an E0 stream cipher and WEP uses RC4 stream cipher that provides weak encryption. Wi-Fi uses the IEEE 802.11i (Wide Protected Access 2- WPA2) protocol for encryption. WPA2 uses the block cipher advanced encryption standard (AES). World interoperability for microwave access (Wi-Max) is an IEEE 802.16 standard that aims to deliver wireless data fast and over a long range. Wi-Max uses a combination of AES and 3DES (data encryption standard). In this chapter, we present the foundations of correlation and algebraic attacks. We review the basic features of these attacks and discuss the results of the implementation of these attacks on stream ciphers used in a wireless environment such as Bluetooth.

B. PROPERTIES OF BOOLEAN FUNCTIONS

The Boolean functions are polynomials of n variables and bit output, are used in several cryptographic applications in wireless systems and must satisfy several cryptographic criteria. Although the quality of these properties depends on the specific cryptosystem that is implemented, the properties that a Boolean function must focus on are balance, nonlinearity, correlation immunity, and high algebraic degree, just to mention a few.

1. Balance of Boolean Functions

A Boolean function is *balanced* if its output is equally distributed, which means that its Hamming weight is 2^{n-1}, where *n* is the number of variables.

2. Nonlinearity

The *nonlinearity* of a Boolean function f, N_f, is defined as the minimum Hamming distance between the function itself and every single function that belongs to the set of the affine Boolean functions. Thus,

$$\mathrm{N}_f = \min_{\phi \in \mathrm{A}_n} d(f, \Phi),$$

where A_n is the class of all affine functions on vector space V_n [13, p. 7].

3. Correlation and Algebraic Immunity

A Boolean function f has *correlation immunity* of order *k* if its values are statistically independent of any subset of *k* input variables. Correlation is a useful concept in cryptanalysis, because it may reveal to an attacker how an encryption function f behaves if one slightly changes the input. Furthermore, a Boolean function with low-order degree of correlation immunity is more susceptible to attacks on the system than a Boolean function of high-order degree with correlation immunity. Siegenthaler in [15] showed that a high-algebraic degree will restrict the maximum possible correlation immunity when the correlation immunity *k* of a Boolean function f of degree d and n variables for a given set of input variables satisfies the relation $k + d \leq n$.

Definition 3.1: *An annihilator* of a polynomial f is a nonzero polynomial g, such that $fg = 0$.

The above definition motivates the concept of algebraic immunity $AI(f)$ of a Boolean function f of degree d and of n variables. $AI(f)$ is the least value of

d such that either f or $f \oplus 1$ has an annihilator of degree d. In other words, given f and g of minimum degree d, such that $fg = 0$ or $(f \oplus 1)g = 0$, then the algebraic immunity is d.

Example 3.2:

Assume $f(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4$ and $g(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$, then $fg = x_1x_2x_3x_4 \oplus x_1x_2x_3x_4 \oplus x_1x_2x_3x_4 \oplus x_1x_2x_3x_4 = 0$, since $x_1x_1 = x_1, x_2x_2 = x_2, x_3x_3 = x_3, x_4x_4 = x_4$.

Notice that f is of degree 4 with four variables whereas g is of degree 1.

■

C. CORRELATION ATTACKS

Correlation and fast or conditional correlation attacks [1], [2] use a biased relation between keystream and certain LFSR output sequences that have to be found. A correlation attack is a probabilistic approach of attacking. When an attacker has access to the output of the LFSRs of a cipher of a cryptosystem and the output of a Boolean function that combines the outputs of all the LFSRs, then he may find the initial values of the LFSRs by simply guessing the initial values. The following example illustrates the correlation attack process.

Example 3.3:

Suppose that a keystream generator consists of three LFSRs, say x, y, z, of lengths three, four, and five respectively. Assume that the combiner Boolean function is of the form:

$$f(x, y, z) = xy \oplus yz \oplus z$$

Then, the initial value of the key must be $12 = 3+4+5$ bits long.

Suppose that the initial values of the LFSRs are $x = 011, y = 0101, z = 11100$, and for bits $i = 0, 1, 2, ...23$ the following evaluations hold:

$$x_i = 011100101110010111001011$$
$$y_i = 010110010001111010110010$$
$$z_i = 111000110111010100001001$$
$$k_i = 111100100110010110001011$$

where k_i is the keystream.

The truth table of the combined Boolean function f is of the following form:

x	*y*	*z*	*f*
0	**0**	**0**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**0**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	**1**	**1**	**1**

where $f = 01000111$.

By comparing the columns of variables *x, y* with *f* one can easily observe that $f(x, y, z) = x$ with probability $P(f = x) = 3/4$ and $f(x, y, z) = z$ with probability $P(f = z) = 3/4$. Assume that the attacker has access to the following keystream table:

$$k_i = 111100100110010110001011$$

The attacker is trying to find the initial values of the LFSRs and he guesses that $x = 111$, and he then generates the first 24 bits of x and compares it to k_i as follows:

$$x_i = 111001011100101110010111$$
$$k_i = 111100100110010110001011$$

Comparison of the two shows that only 12 out of 24 bits match exactly, so the question is this: can an attacker make a better guess? If the attacker guesses $x = 011$ and he then generates the first 24 bits of x and compares it to k_i, he will find 21 out of 24 bits, which is a better match, so the attacker has found the initial values of x as seen below:

$$x_i = 011100101110010111001011$$
$$k_i = 111100100110010110001011$$

If the *n* LFSRs have lengths $n_0, n_1, ..., n_{n-1}$, then the correlation attack needs $2^{n_0-1} + 2^{n_1-1} + ... + 2^{n_{n-1}-1}$ effort, which is much less than the work required for the exhaustive key search that is $2^{n_0+n_1+...+n_{n-1}-1}$.

The main derivatives of correlation attacks are fast correlation attacks and conditional correlation attacks. Lu and Vaudenay [1] in 2004 introduced a *fast correlation attack* and implemented it in a Bluetooth E0 keystream generator (Chapter V details an E0 keystream generator). Despite the fact that correlations of E0 have been discussed but only for a short sequence of bits, Lu and Vaudenay formulated a powerful computation method of correlations using a recursive expression based on the maximum likelihood decoding (MLD) algorithm by means of a fast Walsh transform (FWT). In order for their attack to succeed, they built a distinguisher for E0 based on the largest bias they found. Their best result, as it concerns E0, is limited to 2^{37} operations for precomputation and 2^{39} operations for the actual keysearch.

The *conditional correlation attack* takes advantage of the linear correlation of the inputs conditioned on a known output pattern of a particular nonlinear function and was proposed by Lu, Meier and Vaudenay in 2005. The best result that they obtained on a Bluetooth E0 keystream generator was in 2^{38} operations required the first 24 bits of $2^{23.8}$ frames [2].

D. ALGEBRAIC ATTACKS

At a very high level, algebraic attacks on stream ciphers based upon LFSRs recover the secret key by solving an over-defined system of multivariable algebraic equations. One successfully does so by exploiting multivariable relations involving keybits and output bits, this process becomes more efficient once relations of low degrees can be found. The idea of algebraic attacks is based on the capability of an attacker to solve a system of nonlinear multivariable equations of low degree. Courtois and Meier introduced algebraic attacks [16] in 2003. Algebraic attacks have been successful in breaking some keystream generators like Toyocrypt and LILI 128 by drastically reducing the computation time needed. The key idea is to generate low-degree equations by multiplying the initial equations by well-chosen multivariable polynomials. The basic methods used to solve the derived system of equations are the Gröbner basis algorithm or linearization methods like extended linearization (XL) [17].

Courtois and Meier introduced three scenarios (S3a, S3b and S3c) under which low-degree relations may exist in order to implement algebraic attacks [18].

- S3a - assume that there is a function *g* of low degree such that $fg \neq 0$ and fg is a low-degree function, where *f* is a Boolean encryption function
- S3b - assume that there is a function *g* of low degree such that $fg = 0$, where *f* is a Boolean encryption function
- S3c. assume that there is a function *g* of high degree and f is of high degree, such that $fg \neq 0$ and fg is of a low-degree function, where *f* is a Boolean encryption function

Meier, Pasalic and Carlet [19] described a method to find all possible annihilators of a given Boolean function *f* and an algorithm which determines whether a Boolean function of *n* variables has low algebraic immunity.

Several algorithms have been introduced that assist in reducing the complexity of solving systems of multivariable equations, but there is no silver bullet, since Garey and Johnson [20] indicate that solving such systems of multivariate polynomial equations is a nonpolynomial (NP)-hard problem. The classical algorithm for solving such a system of equations is Buchberger's algorithm, which transforms the polynomial equations to a Gröbner basis [21]. A Gröbner basis is a set of multivariate polynomials that has the property of Gaussian elimination (one may solve one variable at a time). Every set of polynomials can be transformed into a Gröbner basis. The solution to a Gröbner basis is the same as for the original equation. The linearization algorithms, like XL, have the following steps:

- Find an over-defined equation
- Replace each monomial with a new variable
- Solve the new system of equations as a linear system

Example 3.4:

Assume the following system of equations :

$$x_1 \oplus x_2 \oplus x_3 = 0$$
$$x_3^2 \oplus x_1 x_2 \oplus 1 = 0$$
$$x_1 x_2 \oplus x_1 = 0$$
$$x_1^2 \oplus x_1 x_2 \oplus x_3^2 = 0$$
$$x_3^2 \oplus x_1^2 \oplus x_2 = 0$$
$$x_1^2 \oplus x_2 = 0$$

By substitution, $u_1 = x_3^2, u_2 = x_1 x_2, u_3 = x_1^2,$

The following system of linear equations is then obtained:

$$x_1 \oplus x_2 \oplus x_3 = 0$$
$$u_1 \oplus u_2 \oplus 1 = 0$$
$$u_2 \oplus x_1 = 0$$
$$u_3 \oplus u_2 \oplus u_1 = 0$$
$$u_1 \oplus u_3 \oplus x_2 = 0$$
$$u_3 \oplus x_2 = 0$$

which is easy to solve.

In 2003, Armchnecht and Krause [22] applied algebraic attacks in wireless systems like Bluetooth E0 in which the key could be recovered in $2^{68.48}$ operations after the adversary had knowledge of $2^{23.07}$ keystream bits. Armchnecht in 2004, by using a precomputation step, reduced the complexity to $2^{54.51}$ operations after the adversary had knowledge of $2^{23.44}$ keystream bits [23].

E. CONCLUSION

In this chapter, the author reviewed some of the recent types of attacks on wireless systems, namely correlation and algebraic attacks. It seems that correlation attacks are faster in the computational process in wireless encryption systems, like Bluetooth, which use stream ciphers, yet algebraic attacks require less data during the preprocessing phase. In the following chapters, the author will investigate a recently introduced type of algebraic attack, the cube attack, which will be applied on the E0 keystream generator.

IV. CUBE ATTACK

A. INTRODUCTION

At Crypto Conference 2008, Shamir described a new type of algebraic attack, the cube attack. In September 2008, Dinur and Shamir published a paper on *eprint* [4] entitled “Cube Attacks on Tweakable Black Boxes Polynomials” describing their approach. The cube attack is a generic attack that may be applied to block ciphers, stream ciphers, or even keyed hash functions without necessarily having knowledge of the internal structure of the cipher, as long as at least one output bit can be represented by a polynomial of low degree of the secret and public variables. Their approach is based on the basic algebraic cryptanalysis concept, which attempts to lower the degree of the polynomial equations that represent a cryptosystem by polynomials of lower degree. The polynomial equations used to describe a cryptosystem are variants derived from a master polynomial by setting some variables to any possible value (0 or 1) and then summing the results. They call this attack the cube attack

“...since it sets some public variables to all their possible values in n, $(d-1)$-dimensional Boolean cubes, and sums the results in each cube, where d represents the degree of the polynomial and n is the number of variables." [4, p. 5]

The mathematical concepts we use in this chapter are Boolean functions (polynomials of *n* variables and bit output), factorization of multivariable equations to reveal linear co-factors called superpolys, and solving a system of linear equations.

B. BACKGROUND/KEY OBSERVATIONS ON THE CUBE ATTACK

Actually, the idea of the cube attack is not new. Variations of this attack have been proposed in [24], [25], [26]. These approaches are mostly based on the use of heuristics that sum the output values of Boolean cubes of publicly-

known variables. They are referred to as chosen-IV statistical attacks and are mainly applicable against stream ciphers. However, the cube attack has a more wide range of targets and may be applied to block ciphers.

In the cube attack, when the master polynomial is random one may eliminate with high probability all of the nonlinear terms by using, for example, a chosen plaintext attack, thus reducing the complexity from polynomial time to a system of linear equations that is (relatively) easy to solve. Dinur and Shamir implemented their cube attack on the Trivium stream cipher and recovered the encryption key in 2^{19} bit operations. The previous best-known attempt was made by Fischer, Khazaei and Meier in [27], using a chosen-IV statistical analysis. They succeeded in key recovery of 2^{55} bit operations. The master polynomial was in algebraic normal form (ANF), which means that it must be in sum of products of variables.

The following theorem expresses the concept of the cube attack.

Theorem 4.1: [from 5] *Let $f(x)$ be a polynomial in n variables of degree d. Suppose $0 < k \le d$ and t is the monomial $x_0 x_1 ... x_{k-1}$. Suppose f can be written in the following form:*

$$f(x) = tP_t(x) \oplus Q_t(x), \tag{4.1}$$

where none of the terms in $Q_t(x)$ is divisible by t. Note that $\deg(P_t) \le d - k$.

Then, the sum f over all $(x_0, ..., x_{k-1}) \in F_2^k$, $\sum_{(x_0,...,x_{k-1}) \in F_2^k} f$, considered as a polynomial in k, equals

$$P_t(\overbrace{1, ..., 1}^{k}, x_k, x_{k+1}, ..., x_{n-1})$$

and hence is a polynomial of degree at most $d - k$.

Proof: Consider the following equality: $f = tP_t \oplus Q_t$.

Then,

$$\sum_{(x_0,...,x_{k-1})\in F_2^k} tP_t = P_t(1,...,1,x_k,x_{k+1},...,x_{n-1}) \oplus \sum_{(x_0,...,x_{k-1})\in F_2^k} Q_t.$$

However, $\sum_{(x_0,...,x_{k-1})\in F_2^k} tP_t = 0$ since in order for the summation to be different from 0, all $x_{0,}...,x_{k-1} = 1$, hence

$$\sum_{(x_0,...,x_{k-1})\in F_2^k} tP_t = P_t(1,...,1,x_k,x_{k+1},...,x_{n-1}).$$

Furthermore, Q_t is a sum of monomials that are not divisible by *t*. Let *m* be any one of these monomials. Since *m* is not divisible by *t*, then x_i is excluded for $0 \le i \le k-1$.For instance, if x_0 is excluded, then the sum across all $(x_{0,}...x_{k-1}) \in F_2{}^k$ can be further split into two sums: the sum for $x_0 = 0$ and the sum for $x_0 = 1$ .These two sums are equal since x_0 does not appear in *m*.

Therefore,

$$\sum_{(x_0,...,x_{k-1})\in F_2^k} m = 0 \Rightarrow \sum_{(x_0,...,x_{k-1})\in F_2^k} Q_t = 0. \blacksquare$$

The polynomial f written in the form of Theorem 4.1 is called a master polynomial.

The following example illustrates Theorem 4.1.

Example 4.2:

Consider given a master polynomial f of degree *d* = 3 and of four variables, two known variables (x_1, x_2) and two unknown or secret variables (x_3, x_4). Suppose f has the following algebraic normal form (ANF):

$$f(x_1, x_2, x_3, x_4) = x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_2x_3x_4 \oplus x_1x_3x_4 \oplus x_1x_2 \oplus x_1 \oplus x_1x_3 \oplus x_4 \oplus x_3 \oplus 1, \quad (4.2)$$

Third-degree polynomials with four variables may have $\binom{4}{3}+\binom{4}{2}+\binom{4}{1}+\binom{4}{0}=15$ possible terms. From these 15 terms, five terms are going to be linear and the remaining ten terms are going to be nonlinear. To eliminate all the nonlinear terms using Gaussian elimination, and in order to eliminate all the nonlinear terms, at least ten such polynomials of the total 2^{10} possible terms, over $GF(2)$, are needed. If the two known variables x_1, x_2 are set in all their possible values (0 or 1), then one can construct $2^2 = 4$ derived polynomials, which may not be sufficient.

x_1	x_2	*Derived Polynomials from* f	*Formal Sum over all values of* (x_1, x_2)
0	**0**	$x_4 \oplus x_3 \oplus 1$	$\sum_{(x_1,x_2)\in\{0,1\}^2} f(x_1, x_2, x_3, x_4) = x_3 \oplus x_4 \oplus 1$,[1]
0	**1**	$x_3x_4 \oplus x_4 \oplus x_3 \oplus 1$	
1	**0**	$x_3x_4 \oplus x_4$	
1	**1**	$x_3 \oplus 1$	

Table 3. Formal sum of known variables

The points (0,0),(0,1),(1,0),(1,1) can be viewed as a corner of a square of two dimensions (Figure 4.1).

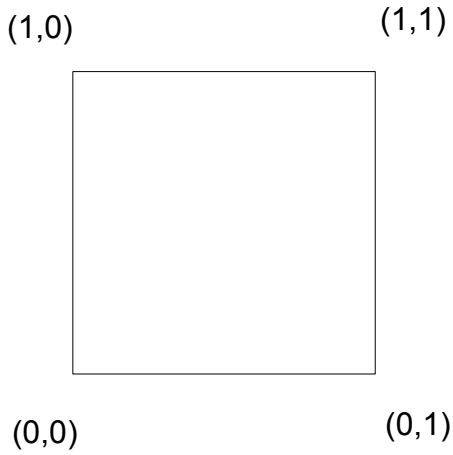

Figure 1. Square of Two Dimensions

This concept may scale to more than two variables. For example, if there are three variables then the evaluation will be for eight points, and these correspond to the corners of a cube in three dimensions, which is why Dinur and Shamir called their process the cube attack (Figure 4.2).

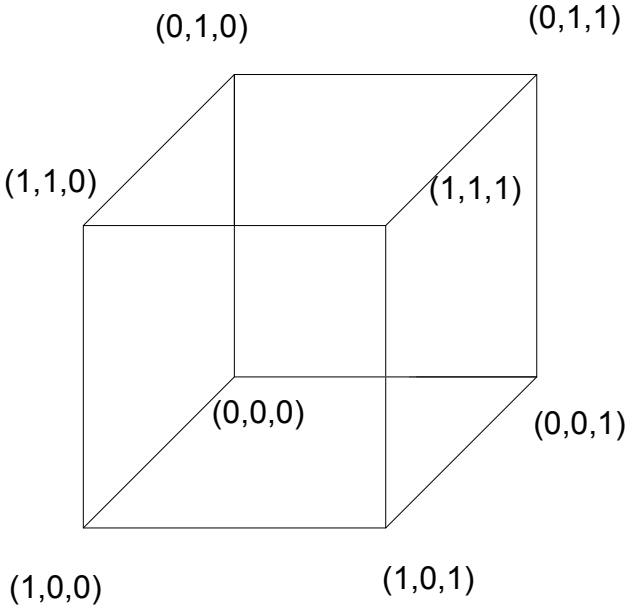

Figure 2. Cube of Three Dimensions

In a similar fashion, once the function f is factored with respect to coefficients x_1, x_2

$$f(x_1, x_2, x_3, x_4) = x_1 x_2 (x_3 \oplus x_4 \oplus 1) \oplus (x_2 x_3 x_4 \oplus x_1 x_3 x_4 \oplus x_1 \oplus x_1 x_3 \oplus x_4 \oplus x_3 \oplus 1), \quad (4.3)$$

where: $t_I = x_1 x_2$ is the maxterm

$P_{t_I}(x) = x_3 \oplus x_4 \oplus 1$ is the superpoly, a linear-cofactor or linear nonconstant polynomial

$Q_{t_I}(x) = x_2 x_3 x_4 \oplus x_1 x_3 x_4 \oplus x_1 \oplus x_1 x_3 \oplus x_4 \oplus x_3 \oplus 1$ is the remainder

The maxterms of the polynomial f are indexed by $I = \{1, 2\}$, a subset of size 2, where $I \subseteq \{1, 2, ..., n\}$ is the index set of the variables that are multiplied together.

Theorem 4.1 is a basic theorem and is the tool used below to cryptanalyze the Bluetooth E0 keystream generator.

Definition 4.3 [from 4]:. *A maxterm of f is a term t_I or cube such that the degree of the superpoly $\deg(P_{t_I}) \equiv 1$, where P_{t_I} is a linear nonconstant polynomial.*

Based on Theorem 4.1 and illustrated in Example 4.4, the sum of the 2^k polynomials derived from the initial polynomial f by assigning all possible values to the *k* variables eliminates all terms, except those that are contained in the superpoly in f.

Observation 4.4: *Using the process described in Theorem 4.1, the monomial coefficients can be computed once all the values of the corresponding variables are summed.*

Example 4.5:

Let f be the following monomial:

$$f(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3 \oplus x_5 \oplus x_3x_4 .$$

Then all values of x_1, x_2, x_3, x_4, x_5 are summed as follows:

$$f(0,0,0,0,0) \oplus f(0,0,0,0,1) \oplus f(0,0,0,1,0) \oplus f(0,0,1,0,0) \oplus ... \oplus f(1,1,1,1,1) = 0 .$$

The value of the expression above represents the coefficient of the monomial $x_1x_2x_3x_4x_5$. Thus,

$$f(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3 \oplus x_5 \oplus x_3x_4 \oplus 0 \cdot x_1x_2x_3x_4x_5 .$$

Observation 4.4 may be generalized. Assume that the encryption function is of the form:

$$z = f(x, v), \tag{4.4}$$

Equation (4.4) actually represents the encryption function of a stream cipher that takes as input *n*-secret bits x and *m*-known bits v of initialization vector (IV) and outputs a keystream bit z.

Initially, the initialization vector bits v are fixed over F_2, and *T* is the set of all possible values of v, so $|T| = 2^m$.

If $f(x, v)$ is summed over $v \in T$, then we can write:

$$\sum_{v \in T} f(x, v) = L(x), \tag{4.5}$$

In accordance with Theorem 4.1, if $L(x) \neq 0$ then a maxterm can be found and therefore one linear relation of the key bits is obtained. Therefore, in order to obtain $n-1$ relations one needs to use the same f with different maxterms. Since there are n such linearly independent relations of the key bits, the secret key can be found by using Gaussian elimination or a chosen plaintext attack.

The cube attack may be completed in two phases: the preprocessing phase where the attacker finds as many maxterms as possible, and the actual attacking phase where the attacker solves the system of linear equations.

C. PREPROCESSING AND ONLINE PHASE

1. Preprocessing Phase

Assume that the following relation represents an encryption function of a cipher represented in accordance to theorem 4.1

$$f(x_1,...,x_n) \equiv t_I P_{t_I}(x) \oplus Q_{t_I}(x_1,...,x_n), \qquad (4.6)$$

and let C_I represent the summation cube of a set of variables with index I.

Then, if t_I is a maxterm of the encryption function f in (4.6), then the attacker may compute the free term of $P_{t_I}(x)$ by summing all the values of $f(x)$ over all variables modulo 2 that are zero except those that appear in C_I,

$$\sum_{(x_0,...,x_{k-1}) \in F_2^k} f = P_t(1,...,1,x_k,x_{k+1},...,x_{n-1})$$

Then the attacker can compute the coefficient of x_i in the linear expression $P_{t_I}(x)$ by summing modulo 2 all values of $f(x)$ for input vectors equal to 0 except at x_i which is 1, as detailed in the proof of Theorem 4.1. [4]

In the preprocessing phase, the attacker is trying to find as many maxterms $(v_1,...,v_n)$ as possible and their corresponding superpolys $(x_1,...,x_n)$, in the following manner:

$$f(x,v) = v_1v_2v_3(x_1 \oplus x_2 \oplus x_3) \oplus ...$$

$$f(x,v) = v_4v_5v_6(x_2 \oplus x_4) \oplus ...$$

$$f(x,v) = v_3v_5(x_4 \oplus x_6) \oplus ...$$

.....

$$f(x,v) = v_2v_4(x_5 \oplus x_6) \oplus ...$$

When the attacker has no information about the structure of the encryption function, then it can be considered as a blackbox polynomial. The attacker can reconstruct the superpolys using linearity tests. All he can do is query the function f, meaning that he can pass in a value x of and get a value of $f(x)$. Because in a linear expression the coefficient of any variable x_i is 1 if and only if changing the value of x_i changes the value of the expression, the free term may be computed by setting all variables to 0.

2. Online Phase

In this phase, the attacker has to solve a system of linear equations where each linear equation is the co-factor P_{t_I} of the maxterm t_I. The attacker simply applies a chosen plaintext attack on the cipher. The attacker has to find as many linear relations as possible in order to solve the system of linear equations.

D. EXTENSIONS OF THE CUBE ATTACK

Zhang et al. in [5] proposed two different variations of the cube attack: the cube attack with annihilators and the cube attack on a vectorial Boolean function finding relations with low degree polynomials.

1. Cube Attack with Annihilators

In cube attacks with annihilators the focus is on stream ciphers. Their method is a combination of the algebraic attack of Courtois and Meier [18], and the cube attack [5]. They adapt the main observation of Courtois and Meier about polynomials: for some polynomial f one may find a polynomial g of lower degree than f, such that $h \equiv fg$.

Assume that there is a stream cipher and the output bit is

$$z = f(x,v), \tag{4.7}$$

where x is the unknown variable and v represents the known variable. Courtois' concept may be applied in the cube attack and one ends up with the following relation [from 5]:

$$\sum_{v \in C} h(x,v) = \sum_{v \in C} f(x,v)g(x,v), \tag{4.8}$$

where $\deg(g) = k$, $\deg(f) = d$ and $k < d$. Then $\deg(h) = l$, where $l < d$ and $l > k$.

In the basic steps of the cube attack with annihilators the attacker, initially uses known algorithms to find g and h. Then, in the preprocessing phase, the attacker computes the polynomial derived from the summation

$$\sum_{v \in C} h(x,v), \tag{4.9}$$

and in the online phase, he calculates through linearization the summation

$$\sum_{v \in C} f(x,v)g(x,v) = \sum_{v \in C} h(x,v), \tag{4.10}$$

Zhang et al. implemented the above attack in a Toyocrypt cipher with re-synchronization, breaking the cipher in a few milliseconds on an ordinary PC [5].

2. Cube Attack on a Vectorial Filter Function with Low Degree

In the cube attack on a vectorial filter function with low degree Zhang et al. in [5] combined the cube attack with annihilators with a low degree on vectorial

equations that are obtained from the computation of the rank of the matrices of some monomials.

Assuming we have the following vectorial filter function:

$$z = f(x, v), \tag{4.11}$$

where x are unknown bits of size n, v are known bits of size m and z is a vector of multiple output bits. A function of $g(x, v, z)$ is found where $\deg(x, v) = k$ such that $h(x, v) \equiv g(x, v, f(x, v))$ is of degree l, with $k < l \leq \deg(f)$.

The attack phases are as follows [from 5]:

Firstly, g, h must be found. Therefore we choose $\sum_{\kappa=0}^{e} \binom{n}{k}$ maxterms, where *e* is the vector where the *k-th* component is 1 and the rest are 0. For each maxterm the summation $\sum_{C} h(x, v)$ is computed by finding the coefficient of every x-monomial.

Finally, in the online phase for each maxterm $\sum_{C} g(x, v, z)$ is computed as a polynomial of x, since z is known.

The cube attack with annihilators may be applied on single-bit output ciphers whereas the cube attack with a filter function may be applied on multi-output stream ciphers.

THIS PAGE INTENTIONALLY LEFT BLANK

V. BLUETOOTH KEY STREAM GENERATOR E0

A. INTRODUCTION

The Bluetooth encryption concept is described in Volume 2, Part C, Chapter 4.2 of the Bluetooth specification document [28]. Bluetooth is the name of a wireless communication protocol used for exchanging data from mobile and fixed devices (laptops, PCs, mobile phones, etc.) at low energy and short range, thus creating personal area networks (PANs). Bluetooth communication ranges (transmitter/receiver) from 1 to 10 meters (approximately 33 feet), and high-energy Bluetooth devices enable ranges up to 100 meters (approximately 328 feet). Bluetooth provides authentication mechanisms and data encryption, ensuring confidentiality of the data using point-to-point or broadcast encryption. [28, p. 935] Bluetooth uses the stream cipher algorithm E0 for encryption, which is a combinatory generator with memory. For the rest of the thesis, the author will concentrate on analyzing the key generation process investigating the cryptographic strength of E0 under a cube attack.

B. BLUETOOTH'S ENCRYPTION APPROACH

Every time two Bluetooth devices want to communicate securely with each other, key exchange protocols are in use. Once both users agree on a shared secret, called *link key*, and authenticate themselves, this link key is used later to generate the encryption key (K_c). Although Bluetooth uses algorithms E21 and E22, which are based on the block cipher Secure and Fast Encryption Routine (SAFER+), to authenticate its users and for key derivation, Bluetooth does not use these algorithms to encrypt information [28, p. 952]. The actual data of the packet are enciphered separately. The encryption algorithm E0 uses the originator's Bluetooth device address, usually called the master device (BD_ADDR), twenty-six bits of the originator's clock time and the encryption key K_C.

K_C is the secret key that is produced by the current link key. A 96-bit encryption offset number called COF, known from the authentication procedure, and a 128-bit random number (EN_RAND) which is a public variable that is transmitted as plaintext, are needed in order to produce this encryption key K_C, as depicted in Figure 3. This process executes in the encryption algorithm E3.

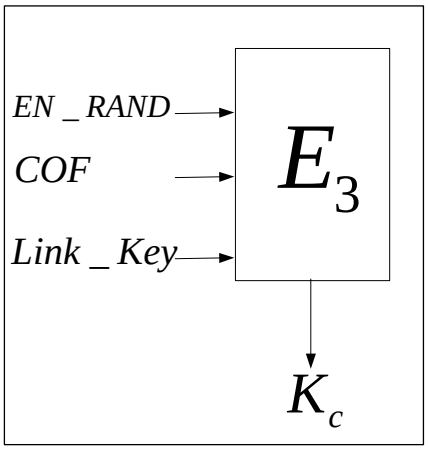

Figure 3. Encryption Algorithm E3 (After [28, p. 953])

Inside E0, the secret key K_C is modified into another key, namely $K_C^{'}$. The $K_C^{'}$ key is used along with the public variables, the originating device's media access control (MAC) address, and the clock value. The clock value changes on each packet (and acts as an “IV”), as is shown in Figure 4.

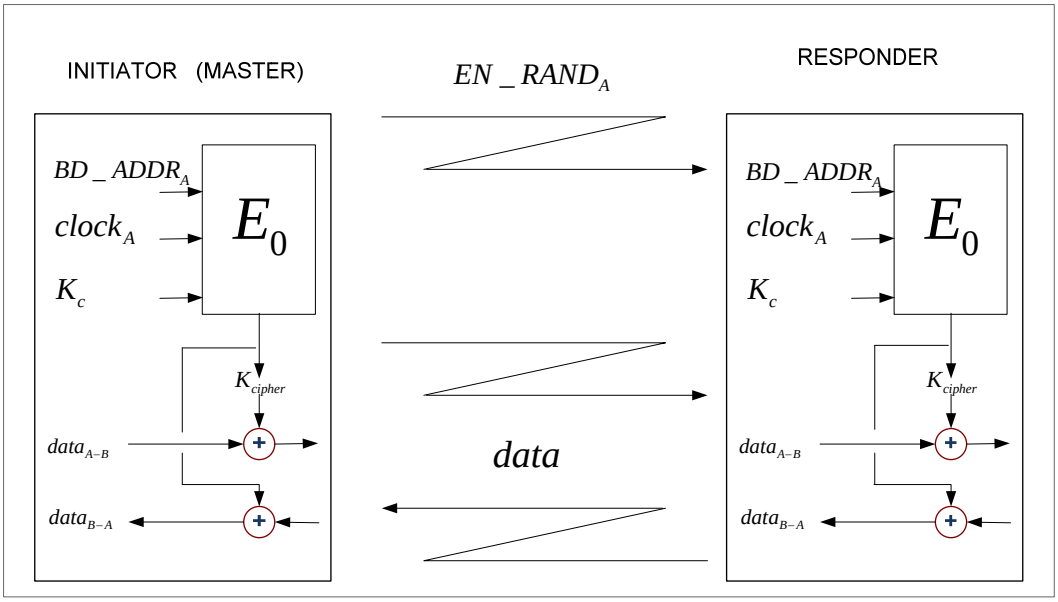

Figure 4. Functional Description of the Encryption Procedure (After [28 p. 937])

The encryption algorithm E0 generates a binary keystream, called K_{Cipher}, which is bitwise XORed with the plaintext. The cipher is symmetric and the decryption will be performed in a similar way, as the receiver generates the same keystream that is then bitwise XORed with the ciphertext to produce the plaintext.

C. STREAM CIPHER E0

Stream cipher E0 is a keystream combination generator with memory. It uses four LFSRs of total length 128 bits and a nonlinear combiner function with memory. A finite state machine, called a summation combiner, with sixteen states, combines the output of the LFSRs. The output of this state machine represents the key sequence, or during the initialization phase is the randomized initial start value. The algorithm uses the encryption key K_C, a 48-bit address, the master clock bits CLK_{26-1}, and a 128-bit random number [28, p. 937–938]. The setup of an E0 keystream generator is depicted in Figure 5.

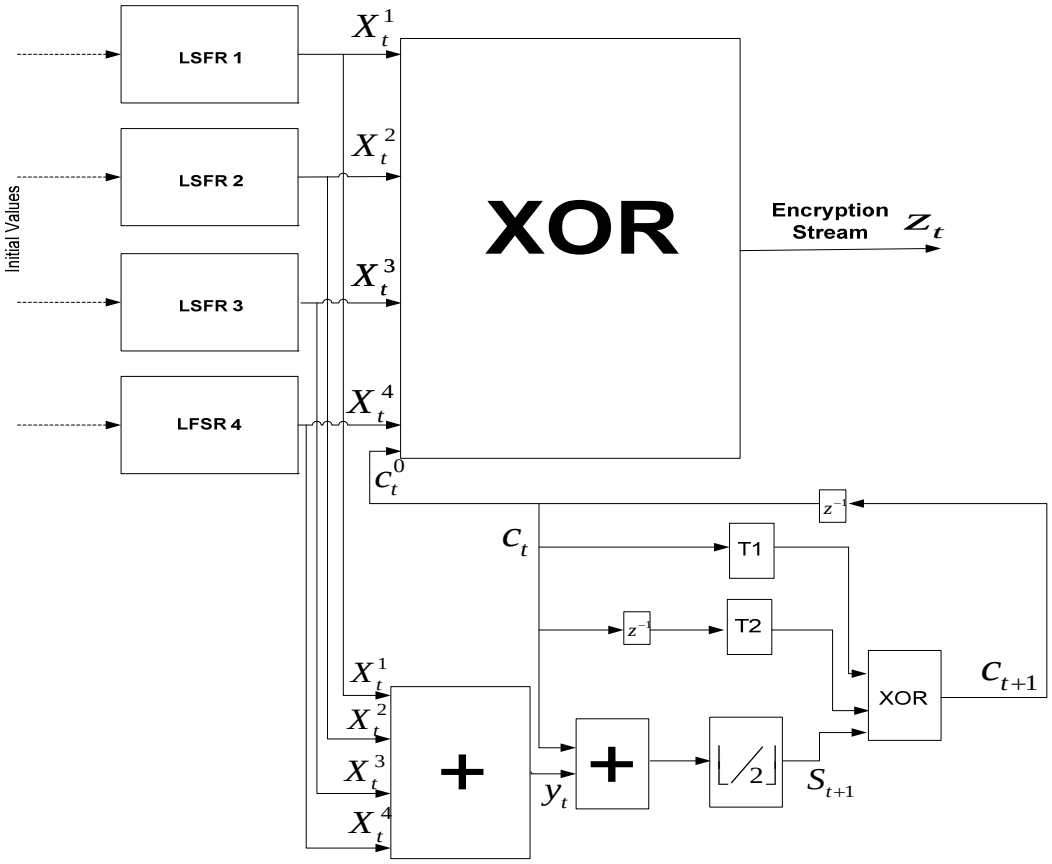

Figure 5. Encryption Procedure (After [39])

The four linear feedback shift registers E0 (LFSR1, LFSR2, LFSR and LFSR4) of E0 have the following lengths:

$$L_1 = 25, L_2 = 31, L_3 = 33, L_4 = 39 .$$

Their corresponding polynomials, which are all primitive, are shown in Table 5.

Primitive Feedback Polynomials of E0			
i	L_i	Primitive Feedback Polynomials $f_i(x)$	Hamming Weight
LFSR1	25	$x^{25} \oplus x^{20} \oplus x^{12} \oplus x^{8} \oplus 1$	5
LFSR2	31	$x^{31} \oplus x^{24} \oplus x^{16} \oplus x^{12} \oplus 1$	5
LFSR3	33	$x^{33} \oplus x^{28} \oplus x^{24} \oplus x^{4} \oplus 1$	5
LFSR4	39	$x^{39} \oplus x^{36} \oplus x^{28} \oplus x^{4} \oplus 1$	5

Table 4. Primitive Feedback Polynomials of E0 (From [28, p. 938])

The Hamming weight of each primitive polynomial is five; therefore, the generated sequences have good statistical properties. On the other hand, they are easy to implement in hardware.

The encryption process of E0 is described below. The LFSRs and the memory bits are initialized with the key, an address, a random number, and clocking bits. The clocking bits ensure that the system will not run numerous times with the same initialization and therefore disclose bits of the key. Let x_t^i denote the output bit of $LFSR^i$ at clock-time *t*. Then we generate the value y_t from the 4th tuple $x_t^1, x_t^2, x_t^3, x_t^4$ by:

$$y_t = \sum_{i=1}^{4} x_t^i \,, \tag{5.1}$$

The summation is over the integers, which means that y_t belongs to $\{0,1,2,3,4\}$. The output of the summation generator can be obtained as follows. [2]

The function f_0 is formed using the XOR operation and one can generate z_t of the keystream:

[2] The glossary of E0 keystream generator can be found in Appendix D.

$$z_t = f_0(x_t, c_t^0) = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0,\ z_t \in \{0,1\} \tag{5.2}$$

The nonlinearity of E0 comes from the function f_1, whose output is a two-bit sequence s_t.

$$s_{t+1} = (s_{t+1}^1, s_{t+1}^0) = f_1(x_t, c_t) = \left\lfloor \frac{y_t + c_t}{2} \right\rfloor \in \{0,1,2,3\} \tag{5.3}$$

The "+" symbol in Equation (5.3) is the usual integer sum. The memory update function is a composition of f_1 and T and is linear with the following form:

$$c_{t+1} = (c_{t+1}^1, c_{t+1}^0) = T(s_{t+1}, c_t, c_{t-1}) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}], \tag{5.4}$$

where $T_1[.]$ and $T_2[.]$ are two different linear bijections over GF(4), summarized in Table 6 [28, p. 939].

E0 Linear Bijections Mapping to Binary Vectors			
$x_0 x_1$	$T_1[x]$	$T_2[x]$	$T_1 : (x_1, x_0) \mapsto (x_1, x_0)$ $T_2 : (x_1, x_0) \mapsto (x_0, x_1 \oplus x_0)$
00	00	00	
01	01	11	
10	10	01	
11	11	10	

Table 5. Mappings of T_1 and T_2

The E0 algorithm must be initialized with a value from the four LFSRs (128 bits in total) and the four bits that specify the values of c_0, c_{-1}. The 132-bit initial value is derived from four inputs using the key stream generator. The input parameters are K_c, a 128-bit random number RAND, a 48-bit Bluetooth device address, and the twenty-six originator's device clock bits CLK_{26-1} [28, p. 940].

D. MODELING ENCRYPTION FUNCTION OF E0

During the author's investigation of the encryption function of the E0 algorithm, he adopted Armknecht and Krause's approach in order to find a function that is not dependent on memory bits and holds for every clock tick. [22]

Let z_t be the keystream bit produced by E0 at clock t, z_{t+1} be the keystream bit produced by E0 at clock $t+1$, etc. These bits are randomly generated. At every clock value, the output of E0 is the bit z_t, which is dependent on the output bits of four LFSRs $x_t = (x_t^1, x_t^2, x_t^3, x_t^4) \in \{0,1\}^4$ and the four memory bits $c_t \in \{0,1\}^4$.

In more detail, the components of $c_t = (c_t^1, c_t^0)$ are as follows:

$$c_t^1 = s_t^1 \oplus c_{t-1}^1 \oplus c_{t-2}^0 \quad , \tag{5.5}$$

$$c_t^0 = s_t^0 \oplus c_{t-1}^0 \oplus c_{t-2}^1 \oplus c_{t-2}^0 , \tag{5.6}$$

The goal of the cryptanalysis is to come up with an equation that describes the encryption of the E0 keystream generator consisting only of the bits of the LFSRs and key stream bits z_t, while eliminating the memory bits c_t. The reason is that the author does not want to use a polynomial of degree n where the system of equations would be unsolvable [23, p. 5].

The encryption function G for E0 becomes

$$G(L^t(K), z_t, z_{t+1}, z_{t+2}, z_{t+3}) = 0 \text{ , where } L^t(K) = (x_{1,} x_2, ..., x_{16}) \tag{5.7}$$

More specifically,

$$z_t \oplus z_{t+1} \oplus z_{t+2} \oplus z_{t+3} \oplus \Pi_{t+1}^2(z_t \oplus z_{t+1} \oplus z_{t+2} \oplus z_{t+3}) \oplus \Pi_{t+1}^4 \oplus$$
$$\Pi_{t+1}^1(z_t \oplus z_{t+2} \oplus z_{t+3} \oplus z_{t+1}z_t \oplus z_{t+1}z_{t+2} \oplus z_{t+1}z_{t+3}) \oplus$$
$$\Pi_t^1 \oplus \Pi_t^1\Pi_{t+1}^1(1 \oplus z_{t+1}) \oplus \Pi_t^1\Pi_{t+1}^2 \oplus \Pi_{t+2}^1 z_{t+2} \oplus \Pi_{t+2}^1\Pi_{t+1}^1 z_{t+2}(z_{t+1} \oplus 1) \oplus$$
$$\Pi_{t+2}^1\Pi_{t+1}^2 z_{t+2} \oplus \Pi_{t+2}^2 \oplus \Pi_{t+2}^2\Pi_{t+1}^1(1 \oplus z_{t+1}) \oplus \Pi_{t+2}^2\Pi_{t+1}^2 \oplus$$
$$\Pi_{t+3}^1 \oplus \Pi_{t+3}^1\Pi_{t+1}^1(1 \oplus z_{t+1}) \oplus \Pi_{t+3}^1\Pi_{t+1}^2 \oplus$$
$$\Pi_{t+1}^3 z_{t+1} \oplus \Pi_{t+1}^2 \oplus \Pi_{t+1}^1 = 0, \tag{5.8}$$

where Π_t^i denotes the *i*-th elementary symmetric polynomial in x_t^i.

$$\begin{aligned}
\Pi_t^1 &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \\
\Pi_t^2 &= x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_2x_4 \oplus x_3x_4 \\
\Pi_t^3 &= x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \\
\Pi_t^4 &= x_1x_2x_3x_4
\end{aligned} \tag{5.9}$$

$$\begin{aligned}
\Pi_{t+1}^1 &= x_5 \oplus x_6 \oplus x_7 \oplus x_8 \\
\Pi_{t+1}^2 &= x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8 \\
\Pi_{t+1}^3 &= x_5x_6x_7 \oplus x_5x_6x_8 \oplus x_5x_7x_8 \oplus x_6x_7x_8 \\
\Pi_{t+1}^4 &= x_5x_6x_7x_8
\end{aligned} \tag{5.10}$$

$$\begin{aligned}
\Pi_{t+2}^1 &= x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12} \\
\Pi_{t+2}^2 &= x_9x_{10} \oplus x_9x_{11} \oplus x_9x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{12} \oplus x_{11}x_{12} \\
\Pi_{t+2}^3 &= x_9x_{10}x_{11} \oplus x_9x_{10}x_{12} \oplus x_9x_{11}x_{12} \oplus x_{10}x_{11}x_{12} \\
\Pi_{t+2}^4 &= x_9x_{10}x_{11}x_{12}
\end{aligned} \tag{5.11}$$

$$\begin{aligned}
\Pi_{t+3}^1 &= x_{13} \oplus x_{14} \oplus x_{15} \oplus x_{16} \\
\Pi_{t+3}^2 &= x_{13}x_{14} \oplus x_{13}x_{15} \oplus x_{13}x_{16} \oplus x_{14}x_{15} \oplus x_{14}x_{16} \oplus x_{15}x_{16} \\
\Pi_{t+3}^3 &= x_{13}x_{14}x_{15} \oplus x_{13}x_{14}x_{16} \oplus x_{13}x_{15}x_{16} \oplus x_{14}x_{15}x_{16} \\
\Pi_{t+3}^4 &= x_{13}x_{14}x_{15}x_{16}
\end{aligned} \tag{5.12}$$

and the output bit streams for clock times *t*, *t*+1, *t*+2, *t*+3 are as follows:

$$\begin{aligned} z_t &= a \\ z_{t+1} &= b \\ z_{t+2} &= c \\ z_{t+3} &= d \end{aligned} \tag{5.13}$$

Theorem 5.1: The encryption function of E0 *depends only on the output bits of the four LFSRs and the output keystream bit and holds for every clock tick. Four consecutive clock ticks are needed.*

Proof [from [22]]:

The key stream generator E0 consists of four LFSRs and four memory bits. For every clock time t an output z_t is produced based on the outputs $x_t = (x_t^1, x_t^2, x_t^3, x_t^4)$ of the four LFSRs and the four memory bits $c_t = (q_t, p_t, q_{t-1}, p_{t-1})$. The next memory bits at clock time $t+1$ are $c_{t+1} = (q_{t+1}, p_{t+1}, q_t, p_t)$. The memory bits q_t, p_t appear in both clock times of t and $t+1$. The variable Π_t^i denotes the i-th elementary symmetric polynomial over $x_t = (x_t^1, x_t^2, x_t^3, x_t^4)$, which is the sum of all monomials of length $s \le 4$.

Thus,

$$z_t = \Pi_t^1 \oplus p_t, \tag{5.14}$$

$$c_{t+1} = (q_{t+1}, p_{t+1}, q_t, p_t) \tag{5.15}$$

However, at the same time

$$c_{t+1} = (s_{t+1}^1 \oplus q_t \oplus p_{t-1}, s_{t+1}^0 \oplus q_{t-1} \oplus p_t, q^t, p^t), \tag{5.16}$$

$$s_{t+1} = (s_{t+1}^1, s_{t+1}^0) = \left\lfloor \frac{x_t^1 + x_t^2 + x_t^3 + x_t^4 + 2q_t + p_t}{2} \right\rfloor, \tag{5.17}$$

The contents of the LFSRs and the value of c^1 are set at the beginning. All the other values may be calculated from these.

From Equations (5.15) and (5.16), the following is obtained:

$$c_{t+1} = (q_{t+1}, p_{t+1}, q_t, p_t) = (s^1_{t+1} \oplus q_t \oplus p_{t-1}, s^0_{t+1} \oplus q_{t-1} \oplus p_t, q^t, p^t) \tag{5.18}$$

Assume f_0 and f_1 are two Boolean functions derived from Equations (5.3) and (5.4) such that:

$$s^i_{t+1} = f_i(x^1_t, x^2_t, x^3_t, x^4_t, q^t, p^t) \text{, where } i \in \{0,1\} \tag{5.19}$$

Armknecht [22, p. 173–174] proved that the algebraic normal forms of f_0 and f_1 have the expressions:

$$f_0 = \Pi^2_t \oplus \Pi^1_t p^t \oplus q^t, \tag{5.20}$$

$$f_1 = \Pi^4_t \oplus \Pi^3_t p^t \oplus \Pi^2_t q^t \oplus \Pi^1_t p^t q^t. \tag{5.21}$$

Based on Equation (5.18) we obtain

$$p_{t+1} = s^0_{t+1} \oplus p_t \oplus q_{t-1} \oplus p_{t-1} = \Pi^2_t \oplus \Pi^1_t p_t \oplus q_t \oplus q_{t-1} \oplus p_t \oplus p_{t-1}, \tag{5.22}$$

$$q_{t+1} = s^1_{t+1} \oplus q_t \oplus p_{t-1} = \Pi^4_t \oplus \Pi^3_t p_t \oplus \Pi^2_t q_t \oplus \Pi^1_t p_t q_t \oplus q_t \oplus p_{t-1}, \tag{5.23}$$

The values of p_{t+1} and q_{t+1} depend on $x_t, q_t, q_{t-1}, p_t, p_{t-1}$ and x_t, q_t, p_t, p_{t-1}, respectively.

Equations (5.22) and (5.23) are simplified by using the following equations:

$$\Phi(t) = \Pi^4_t \oplus \Pi^3_t p_t \oplus p_{t-1}, \tag{5.24}$$

$$\Psi(t) = \Pi^2_t \oplus \Pi^1_t p_t \oplus 1. \tag{5.25}$$

Therefore, Equations (5.22) and (5.23) become

$$p_{t+1} = \Psi(t) \oplus 1 \oplus p_{t-1} \oplus p_t \oplus q_{t-1}, \tag{5.26}$$

$$q_{t+1} = \Phi(t) \oplus \Psi(t) q_t , \tag{5.27}$$

From Equation (5.27), the following is obtained:

$$\Psi(t) q_{t+1} = \Psi(t)(\Phi(t) \oplus \Psi(t) q_t) \text{ or}$$

$$\Psi(t)(\Phi(t) \oplus q_t \oplus q_{t+1}) = 0 \text{, since } \Psi(t)\Psi(t) = \Psi(t) . \tag{5.28}$$

Equation (5.26) is then transformed into the following:

$$q_t \oplus q_{t-1} = \Psi(t) \oplus 1 \oplus p_{t-1} \oplus p_t \oplus p_{t+1} . \tag{5.29}$$

Replacing t by $t+1$ in Equation 5.28 and applying Equation 5.29, we have:

$$\Psi(t)(\Phi(t) \oplus \Psi(t+1) \oplus 1 \oplus p_t \oplus p_{t+1} \oplus p_{t+2}) = 0 . \tag{5.30}$$

Applying Equation (5.14) we are now able to derive Equation (5.8) which holds for every clock t and does not have any memory bits in the equation.

$$\begin{aligned}
&z_t \oplus z_{t+1} \oplus z_{t+2} \oplus z_{t+3} \oplus \Pi^2_{t+1}(z_t \oplus z_{t+1} \oplus z_{t+2} \oplus z_{t+3}) \oplus \Pi^4_{t+1} \oplus \\
&\Pi^1_{t+1}(z_t \oplus z_{t+2} \oplus z_{t+3} \oplus z_{t+1}z_t \oplus z_{t+1}z_{t+2} \oplus z_{t+1}z_{t+3}) \oplus \\
&\Pi^1_t \oplus \Pi^1_t\Pi^1_{t+1}(1 \oplus z_{t+1}) \oplus \Pi^1_t\Pi^2_{t+1} \oplus \Pi^1_{t+2}z_{t+2} \oplus \Pi^1_{t+2}\Pi^1_{t+1}z_{t+2}(z_{t+1} \oplus 1) \oplus \\
&\Pi^1_{t+2}\Pi^2_{t+1}z_{t+2} \oplus \Pi^2_{t+2} \oplus \Pi^2_{t+2}\Pi^1_{t+1}(1 \oplus z_{t+1}) \oplus \Pi^2_{t+2}\Pi^2_{t+1} \oplus \\
&\Pi^1_{t+3} \oplus \Pi^1_{t+3}\Pi^1_{t+1}(1 \oplus z_{t+1}) \oplus \Pi^1_{t+3}\Pi^2_{t+1} \oplus \\
&\\
&\Pi^3_{t+1}z_{t+1} \oplus \Pi^2_{t+1} \oplus \Pi^1_{t+1} = 0,
\end{aligned}$$

and in a more generic form:

$$G(x_1, x_2, \ldots x_{16}, z_t, z_{t+1}, z_{t+2}, z_{t+3}) = 0$$

■

Equation (5.8), of degree 4 with twenty variables, can be fully described by the following expression:

$$
\begin{aligned}
0 = {} & a \oplus b \oplus c \oplus d \oplus \\
& (x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8)(a \oplus b \oplus c \oplus d) \oplus x_5x_6x_7x_8 \oplus \\
& (x_5 \oplus x_6 \oplus x_7 \oplus x_8)(a \oplus c \oplus d \oplus ab \oplus bc \oplus bd) \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \\
& (x_1 \oplus x_2 \oplus x_3 \oplus x_4)(x_5 \oplus x_6 \oplus x_7 \oplus x_8)(1 \oplus b) \oplus \\
& (x_1 \oplus x_2 \oplus x_3 \oplus x_4)(x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8) \oplus \\
& (x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12})b \oplus (x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12})(x_5 \oplus x_6 \oplus x_7 \oplus x_8)c(b \oplus 1) \oplus \\
& (x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12})(x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8)c \oplus \\
& (x_9x_{10} \oplus x_9x_{11} \oplus x_9x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{12} \oplus x_{11}x_{12}) \oplus \\
& (x_9x_{10} \oplus x_9x_{11} \oplus x_9x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{12} \oplus x_{11}x_{12})(x_5 \oplus x_6 \oplus x_7 \oplus x_8)(1 \oplus b) \oplus \\
& (x_9x_{10} \oplus x_9x_{11} \oplus x_9x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{12} \oplus x_{11}x_{12})(x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8) \oplus \\
& x_{13} \oplus x_{14} \oplus x_{15} \oplus x_{16} \oplus (x_{13} \oplus x_{14} \oplus x_{15} \oplus x_{16})(x_5 \oplus x_6 \oplus x_7 \oplus x_8)(b \oplus 1) \oplus \\
& (x_{13} \oplus x_{14} \oplus x_{15} \oplus x_{16})(x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8) \oplus \\
& x_5x_6x_7b \oplus x_5x_6x_8b \oplus x_5x_7x_8b \oplus x_6x_7x_8b \oplus x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8 \oplus \\
& x_5 \oplus x_6 \oplus x_7 \oplus x_8.
\end{aligned}
\tag{5.31}
$$

The full expansion of the encryption function of can be found in Appendix A.

VI. AUTOMATED TOOL FOR MODELING CUBE ATTACK

> No matter how correct a mathematical theorem may appear to be, one ought never to be satisfied that there was not something imperfect about it until it also gives the impression of being beautiful.
>
> George Boole (1815–1864)

A. OVERVIEW

In this chapter, the author implemented Dinur and Shamir's cube attack on a Bluetooth E0 keystream generator. In order to do that, he modeled the E0 encryption function of Bluetooth in Chapter V. He then created an automated tool in the Maple 12 environment (http://www.maplesoft.com) that finds all of the maxterms and their corresponding superpolys (linear coefficients) of the encryption function. Then, in the online phase, he used a chosen plaintext attack in order to solve the system of linear equations he found. Eventually, he evaluated the results and investigated the complexity of the process.

B. APPROACH—BASIC ASSUMPTIONS

The most time-consuming work in the computation process, namely finding the maxterms and their corresponding superpolys, was executed in the Maple 12 environment. Maple is a high-level programming language with powerful built-in symbolic algebra, numerical and graphical capabilities. The reasons why the author chose Maple 12 instead of any other programming language like C, C++, Java, or symbolic Python were mainly that he wanted to benefit from the advantages of a high-performance mathematical engine with fully integrated numerals and symbols, especially in algebra. With this in mind, under the guidance of an expert programmer in the Maple environment, Dr. David Canright, Associate Professor of the Department of Applied Mathematics of the Naval Postgraduate School' the author created effective code in a compact and optimal way.

1. Modeling Environment

Maple uses a C-like programming language. It has many of the features that other high-level programming languages have, like loops, conditionals, and functions. Maple does not support classes of objects; however, this feature is overcome by a rich set of packages available for Maple. Maple can generate code in other high-level programming languages like C, Java, Fortran, Visual Basic and Matlab using the CodeGeneration package. The OpenWatcom C compiler is used for the Maple compiler. This allows the user to compile some types of user-written Maple routines to increase code performance.

Maple 12 works on Windows (2000, 2003, XP, Vista), Macintosh, UNIX, Linux and Solaris environments. Developers' system recommendations include the following [29]:

- CPU: AMD X86_64/ 1 GHz/Intel Xeon/ Intel 64
- RAM: 512MB (at least)
- Hard disk: 1 GB

The computational interfaces Maple 12 has available for its users include the *standard worksheet*, which is the environment that the author worked in. The standard worksheet is a full-feature graphical user interface that enables users to create documents, and it displays all the calculations and possible errors in the results. The *standard* interface is written primarily in Java to speed up the computational process and provide portability. The standard worksheet has two modes: the *document mode* and the *worksheet mode*. The main difference between these two modes is that in the first interface the user hides all commands used to perform calculations whereas in the latter interface the user shows all commands. Maple 12 also has other user interfaces such as the *classic worksheet,* which is a basic worksheet environment for computers with limited memory; and the *command line interface,* in which a user may solve large and complex problems without thorough graphical user interface features available.

The *Maplesoft graphing calculator* provides another Maple 12 interface and is available for computers using the Microsoft Windows Operating System only. This graphical user interface contains windows, textbox regions and other visual interfaces that give the user a point–and-click interface to access the computation processor of Maple without using the worksheet. Finally, Maple provides the *Mapletapplication.* It has a graphical calculator interface that the user can use to perform simple computations and create customizable graphs in a windows environment only [30].

2. Basic Assumptions

In part, the cube attack is a chosen plaintext attack: the part that can be manipulated by the attacker. To implement the cube attack, we assume the attacker has the capability to properly send structured packets that the Bluetooth receiver will respond to, thus providing the attacker with access to the encryption machine. This machine behaves like an oracle. If the attacker convinces the oracle it is a legitimate participant, it will be duped into sending data to the attacker or another participant; however, the attacker can observe “over the air” whatever responses the oracle or the user sends back.

For example, the attacker can masquerade as a real user, with sufficient detail to send data to the oracle. The oracle will return encrypted data to the attacker or an authorized user/participant in the communication process, and the attacker will collect this data. The attacker thus gains some knowledge of the output bitstreams for the combiner at clock ticks t, $t+1$, $t+2$, and $t+3$.

The following theorem derived from our investigation:

Theorem 6.1: *The maxterms of E0 encryption function can only be of 2^{nd} or 3^{rd} degree.*

Proof:

Assume that a maxterm could be of degree 4. By Definition 4.3 of the term called maxterm, in order for a maxterm to exist there must be terms in the E0

encryption function of the 5th degree. Since the encryption function being used in this study (Appendix A) is of degree 4, it cannot have a maxterm of degree 4.

Assume that a maxterm could be of degree 1. Then, by the definition of maxterm, since the cofactor must be linear and not constant, one must check all the 2nd degree terms of the encryption function E0 in Equation (5.31). Thus, one may observe there, that the only terms of the 2nd degree derive from the following products:

$\Pi^2_{t+1}(z_t \oplus z_{t+1} \oplus z_{t+2} \oplus z_{t+3}), \Pi^1_t \Pi^1_{t+1}(1 \oplus z_{t+1}), \Pi^1_{t+2}\Pi^1_{t+1} z_{t+2}(z_{t+1} \oplus 1), \Pi^2_{t+2}, \Pi^1_{t+3}\Pi^1_{t+1}(1 \oplus z_{t+1})$

and Π^2_{t+1}.

Each term of the 2nd degree is examined as follows:

$$\begin{aligned}
&\Pi^2_{t+1}(z_t \oplus z_{t+1} \oplus z_{t+2} \oplus z_{t+3}) = \\
&x_5x_7a \oplus x_5x_7b \oplus x_5x_7c \oplus x_5x_7d \oplus x_5x_8a \oplus x_5x_8b \oplus x_5x_8c \oplus x_5x_8d \oplus \\
&x_6x_7a \oplus x_6x_7b \oplus x_6x_7c \oplus x_6x_7d \oplus x_6x_8a \oplus x_6x_8b \oplus x_6x_8c \oplus x_6x_8d \oplus \\
&x_7x_8a \oplus x_7x_8b \oplus x_7x_8c \oplus x_7x_8d,
\end{aligned} \tag{6.1}$$

$$\begin{aligned}
&\Pi^1_t\Pi^1_{t+1}(1 \oplus z_{t+1}) = \\
&x_1x_5 \oplus x_1x_6 \oplus x_1x_7 \oplus x_1x_8 \oplus x_2x_5 \oplus x_2x_6 \oplus x_2x_7 \oplus x_2x_8 \oplus \\
&x_3x_5 \oplus x_3x_6 \oplus x_3x_7 \oplus x_3x_8 \oplus x_4x_5 \oplus x_4x_6 \oplus x_4x_7 \oplus x_4x_8 \oplus \\
&x_1x_5b \oplus x_1x_6b \oplus x_1x_7b \oplus x_1x_8b \oplus x_2x_5b \oplus x_2x_6b \oplus x_2x_7b \oplus x_2x_8b \oplus \\
&x_3x_5b \oplus x_3x_6b \oplus x_3x_7b \oplus x_3x_8b \oplus x_4x_5b \oplus x_4x_6b \oplus x_4x_7b \oplus x_4x_8b,
\end{aligned} \tag{6.2}$$

$$\begin{aligned}
&\Pi^1_{t+2}\Pi^1_{t+1} z_{t+2}(z_{t+1} \oplus 1) = \\
&x_9x_5cb \oplus x_9x_6cb \oplus x_9x_7cb \oplus x_9x_8cb \oplus x_{10}x_5cb \oplus \\
&x_{10}x_6cb \oplus x_{10}x_7cb \oplus x_{10}x_8cb \oplus x_{11}x_5cb \oplus x_{11}x_6cb \oplus \\
&x_{11}x_7cb \oplus x_{11}x_8cb \oplus x_{12}x_5cb \oplus x_{12}x_6cb \oplus x_{12}x_7cb \oplus \\
&x_{12}x_8cb \oplus x_9x_5c \oplus x_9x_6c \oplus x_9x_7c \oplus x_9x_8c \oplus x_{10}x_5c \oplus \\
&x_{10}x_6c \oplus x_{10}x_7c \oplus x_{10}x_8c \oplus x_{11}x_5c \oplus x_{11}x_6c \oplus x_{11}x_7c \oplus \\
&x_{11}x_8c \oplus x_{12}x_5c \oplus x_{12}x_6c \oplus x_{12}x_7c \oplus x_{12}x_8c,
\end{aligned} \tag{6.3}$$

$$\Pi^2_{t+2} = x_9x_{10} \oplus x_9x_{11} \oplus x_9x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{12} \oplus x_{11}x_{12}, \tag{6.4}$$

$$
\begin{aligned}
&\Pi^1_{t+3}\Pi^1_{t+1}(1 \oplus z_{t+1}) = \\
&x_{13}x_5 \oplus x_{13}x_6 \oplus x_{13}x_7 \oplus x_{13}x_8 \oplus x_{14}x_5 \oplus \\
&x_{14}x_6 \oplus x_{14}x_7 \oplus x_{14}x_8 \oplus x_{15}x_5 \oplus x_{15}x_6 \oplus \\
&x_{15}x_7 \oplus x_{15}x_8 \oplus x_{16}x_5 \oplus x_{16}x_6 \oplus x_{16}x_7 \oplus \\
&x_{16}x_8 \oplus x_{13}x_5b \oplus x_{13}x_6b \oplus x_{13}x_7b \oplus x_{13}x_8b \oplus \\
&x_{14}x_5b \oplus x_{14}x_6b \oplus x_{14}x_7b \oplus x_{14}x_8b \oplus x_{15}x_5b \oplus \\
&x_{15}x_6b \oplus x_{15}x_7b \oplus x_{15}x_8b \oplus x_{16}x_5b \oplus x_{16}x_6b \oplus \\
&x_{16}x_7b \oplus x_{16}x_8b,
\end{aligned}
\tag{6.5}
$$

$$
\Pi^2_{t+1} = x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8. \tag{6.6}
$$

Notice that *a,b,c, and d* are assumed known bits (0 ,1) because we assume that the attacker can intercept them; therefore, their appearance as terms in the equation does not increase the degree of the equation since they behave as constants.

In the next steps, the author investigates the unknown variables $x_1,...,x_{12}$ that appear in Equations (6.1) through (6.6).

We note that if there is factoring by x_1 (though of as a maxterm) in Equations (6.1) and (6.2) where x_1 appears, then one gets $x_1(x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_5b \oplus x_6b \oplus x_7b \oplus x_8b)$. However, looking in the Equation (5.31), x_1 appears also in the product:

$$
\Pi^1_t\Pi^2_{t+1} = (x_1 \oplus x_2 \oplus x_3 \oplus x_4)(x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8).
$$

That means that the superpoly is not going to be linear but of 2nd degree and based on Definition 4.3, x_1 fails to be a maxterm.

Similarly, the appearance of the product, $\Pi^1_t\Pi^2_{t+1}$ in Equation (5.31), makes the variables $x_2, x_3, x_4, x_5, x_6, x_7, x_8$ fail to be maxterms for the same reason.[3]

[3] Note that variables x_5, x_6, x_7, x_8 fail at being maxterms because $\Pi^4_{t+1} = x_5x_6x_7x_8$ appears in Equation (5.31).

If one factors x_9 from Equations (6.3) and (6.4), one gets the following product: $x_9(x_5cb \oplus x_6cb \oplus x_7cb \oplus x_8cb \oplus x_{10} \oplus x_{11} \oplus x_{12})$, where x_9 fulfills Definition 4.3. However, looking at Equation (5.31) x_9 also appears in the product:

$$\Pi_{t+2}^2\Pi_{t+1}^2 =$$
$$(x_9x_{10} \oplus x_9x_{11} \oplus x_9x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{12} \oplus x_{11}x_{12})(x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8) =$$
$$x_9x_{10}x_5x_6 \oplus x_9x_{10}x_5x_7 \oplus x_9x_{10}x_5x_8 \oplus x_9x_{10}x_6x_7 \oplus x_9x_{10}x_6x_8 \oplus x_9x_{10}x_7x_8 \oplus$$
$$x_9x_{11}x_5x_6 \oplus x_9x_{11}x_5x_7 \oplus x_9x_{11}x_5x_8 \oplus x_9x_{11}x_6x_7 \oplus x_9x_{11}x_6x_8 \oplus x_9x_{11}x_7x_8 \oplus$$
$$x_9x_{12}x_5x_6 \oplus x_9x_{12}x_5x_7 \oplus x_9x_{12}x_5x_8 \oplus x_9x_{12}x_6x_7 \oplus x_9x_{12}x_6x_8 \oplus x_9x_{12}x_7x_8 \oplus$$
$$x_{10}x_{11}x_5x_6 \oplus x_{10}x_{11}x_5x_7 \oplus x_{10}x_{11}x_5x_8 \oplus x_{10}x_{11}x_6x_7 \oplus x_{10}x_{11}x_6x_8 \oplus x_{10}x_{11}x_7x_8 \oplus$$
$$x_{10}x_{12}x_5x_6 \oplus x_{10}x_{12}x_5x_7 \oplus x_{10}x_{12}x_5x_8 \oplus x_{10}x_{12}x_6x_7 \oplus x_{10}x_{12}x_6x_8 \oplus x_{10}x_{12}x_7x_8 \oplus$$
$$x_{11}x_{12}x_5x_6 \oplus x_{11}x_{12}x_5x_7 \oplus x_{11}x_{12}x_5x_8 \oplus x_{11}x_{12}x_6x_7 \oplus x_{11}x_{12}x_6x_8 \oplus x_{11}x_{12}x_7x_8.$$

That means that the superpoly is not going to be linear, but of 2nd degree, and again by the Definition 4.3, x_9 fails at being a maxterm. The appearance of the same product $\Pi_{t+2}^2\Pi_{t+1}^2$ in Equation (5.31), makes variables x_{10}, x_{11}, x_{12} fail at being maxterms for the same reasons x_9 did.[4]

The results detailed in Table 7 of section C of this chapter illustrate that the maxterms of 2nd and 3rd degree do exist.

C. RESULTS

1. Preprocessing Phase

In Table 7, the author has displayed all the maxterms and their corresponding linear coefficients or superpolys of the encryption function found by running the program in the Maple environment.

[4] Note that variables $x_9, x_{10}, x_{11}, x_{12}$ fail at being maxterms because $\Pi_{t+2}^2\Pi_{t+1}^1, \Pi_{t+2}^1\Pi_{t+1}^2$, appear in Equation (5.31).

Superpolys (with Linear Coefficients)	Cube Indexes of Maxterms of the 2nd Degree	Cube Indexes of Maxterms of the 3rd Degree
$x_6 \oplus x_7 \oplus x_8 \oplus b \oplus 1$	{1,5}, {2,5}, {3,5}, {4,5} {5,13}, {5,14}, {5,15}, {5,16}	{5,9,10}, {5,9,11}, {5,9,12}, {5,10,11}, {5,10,12}, {5,11,12}
$x_5 \oplus x_7 \oplus x_8 \oplus b \oplus 1$	{1,6}, {2,6}, {3,6}, {4,6}, {6,13}, {6,14}, {6,15}, {6,16}	{6,9,10}, {6,9,11}, {6,9,12}, {6,10,11}, {6,10,12}, {6,11,12}
$x_5 \oplus x_6 \oplus x_8 \oplus b \oplus 1$	{1,7}, {2,7}, {3,7}, {4,7}, {7,13},{7,14},{7,15}, {7,16}	{7,9,10}, {7,9,11}, {7,9,12}, {7,10,11}, {7,10,12}, {7,11,12}
$x_5 \oplus x_6 \oplus x_7 \oplus b \oplus 1$	{1,8}, {2,8}, {3,8}, {4,8}, {8,13}, {8,14}, {8,15}, {8,16}	{8,9,10}, {8,9,11}, {8,9,12}, {8,10,11}, {8,10,12}, {8,11,12}
$x_9 \oplus x_{10} \oplus x_{11} \oplus c$	-	{5,6,12}, {5,7,12}, {5,8,12}, {6,7,12}, {6,8,12}, {7,8,12}
$x_9 \oplus x_{10} \oplus x_{12} \oplus c$	-	{5,6,11}, {5,7,11}, {5,8,11}, {6,7,11}, {6,8,11}, {7,8,11}
$x_9 \oplus x_{11} \oplus x_{12} \oplus c$	-	{5,6,10}, {5,7,10}, {5,8,10}, {6,7,10}, {6,8,10}, {7,8,10}
$x_{10} \oplus x_{11} \oplus x_{12} \oplus c$	-	{5,6,9}, {5,7,9}, {5,8,9}, {6,7,9}, {6,8,9}, {7,8,9}
$x_5 \oplus b$	-	{6,7,8}
$x_6 \oplus b$	-	{5,7,8}
$x_7 \oplus b$	-	{5,6,8}
$x_8 \oplus b$	-	{5,6,7}

Table 6. Maxterms and Superpolys of the E0 Keystream Generator

The author ended up with twelve superpolys/linear coefficients, depending on the following unknown variables: $x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}$.

Observation 6.2: The author was forced to use variables $x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}$ as unknowns since they are the only variables that appear as variables in the superpolys. By implementing a chosen plaintext attack, the attacker can determine their values.

This is a useful observation, and in addition, the terms that appear in the 2nd and 3rd columns of the table do not have to be assumed known, but rather only need to be manipulatable.

The program was executed several times, for testing purposes, on an Intel Pentium 4 processor with a CPU of 2.80 GHz and 1GB of RAM, and the results were produced in a mean time of 8.03 seconds, consuming 5.25 MB of memory.

2. Online Phase

Using the encryption function formed by the multivariable polynomial (Appendix A) after the processing phase, the attacker obtained all the possible linear co-factors (superpolys). From the specific encryption function of the multivariable polynomial (obtained after the attacker masquerades as an authorized user and gains access to the security protocol) the attacker will eventually succeed in gathering twelve unique and independent equations:

$$x_5 \oplus b = a_1, \tag{6.1}$$

$$x_6 \oplus b = a_2, \tag{6.2}$$

$$x_7 \oplus b = a_3, \tag{6.3}$$

$$x_8 \oplus b = a_4, \tag{6.4}$$

$$x_6 \oplus x_7 \oplus x_8 \oplus b \oplus 1 = a_5, \tag{6.5}$$

$$x_5 \oplus x_7 \oplus x_8 \oplus b \oplus 1 = a_6, \tag{6.6}$$

$$x_5 \oplus x_6 \oplus x_8 \oplus b \oplus 1 = a_7, \tag{6.7}$$

$$x_5 \oplus x_6 \oplus x_7 \oplus b \oplus 1 = a_8, \tag{6.8}$$

$$x_9 \oplus x_{10} \oplus x_{11} \oplus c = a_9, \tag{6.9}$$

$$x_9 \oplus x_{10} \oplus x_{12} \oplus c = a_{10}, \tag{6.10}$$

$$x_9 \oplus x_{11} \oplus x_{12} \oplus c = a_{11}, \tag{6.11}$$

$$x_{10} \oplus x_{11} \oplus x_{12} \oplus c = a_{12}, \tag{6.12}$$

where $a_i \in \{0,1\}$ and $i \in \{1,...,12\}$ are considered known bits.

The above system of equations is an over-defined system of equations on variables x_5, x_6, x_7, x_8. The solution we obtained is:

$$x_5 = a_1 \oplus b, \tag{6.13}$$

$$x_6 = a_2 \oplus b, \tag{6.14}$$

$$x_7 = a_3 \oplus b, \tag{6.15}$$

$$x_8 = a_4 \oplus b, \tag{6.16}$$

$$x_9 = a_9 \oplus a_{10} \oplus a_{11}, \tag{6.17}$$

$$x_{10} = a_9 \oplus a_{10} \oplus a_{12}, \tag{6.18}$$

$$x_{11} = a_9 \oplus a_{11} \oplus a_{12} \oplus c, \tag{6.17}$$

$$x_{12} = a_{10} \oplus a_{11} \oplus a_{12} \oplus c, \tag{6.18}$$

Remark. *It is worth mentioning that even if not all these assumptions are made, it is still possible to use this approach to find useful information about the output bits of the LFSRs.*

D. ANALYSIS OF THE RESULTS

Below is our main contribution in this thesis.

Theorem 6.3: *If an attacker has* unauthorized access *to the encryption protocol and can use the encryption machine as an oracle so that he can manipulate some of the bits of the LFSRs, and by knowing the output bits of the E0 keystream generator he succeeds in recovering the outputs of the LFSRs at any clock tick.*

Proof:

In section C of this chapter we proved that assuming that an attacker has access to the variables of the four LFSRs at clock time *t, t+1, t+2* and *t+3* and the output bit streams of E0 he can compute the output of the four LFSRs at clocks ticks *t*+1 and *t*+2.

By continuing this process in reverse order, it is easy to observe that one can compute the output of the four LFSRs at clock ticks *t* and *t+1*, by only having access and tweaking the variables and the output of E0 at clock tick *t-1*.

Taking a step back in time at another one clock, an attacker may explicitly find that for the output of the LFSRs at clocks *t, t*-1 he only has to have further access and tweak the variables and the output bits of E0 at clock *t*-2, and so on.

The theorem is proved. ■

Further knowledge about the insight of E0 is needed to correlate the output of the LFSRs and the encryption key placed in E0. A difficulty one may have in completely revealing the encryption key is that in accordance with Lu and Vaudenay in [1], the E0 keystream generator produces limited segments of keystream and after 2745 bits, the generator is reinitialized. However, this is not explicitly stated in the Bluetooth core specifications document.

E. COMPLEXITY

The complexity in this section is measured in operations steps.

1. Preprocessing Phase

Let *d* be the degree of the encryption function *f* and *n* be the number of variables of *f*. During the preprocessing phase, an attacker is trying to find as many maxterms as possible. From this phase, an attacker may obtain *n+1* output bits from the LFSRs and some constant terms. The amount of work needed, based on Zhang et al. in [5], is

$$n(n+1)2^{d-1}$$

The attacker also needs to compute the inverse of the matrix of linear relations matrix. This requires approximately n^3 operations and as a result, an upper bound from this phase is:

$$n(n+1)2^{d-1}+n^3$$

2. Online Phase

For the online phase, where one needs to solve the system of linear equations implementing a chosen plaintext attack, $n2^{d-1}$ evaluations of the E0 encryption function are needed, and the matrix multiplication which takes n^2 operations needs to be performed. Again, by drawing on the analysis by Zhang et al. [5], the complexity is of the following form:

$$n2^{d-1} + n^2$$

Therefore, the overall complexity from both phases is:

$$n(n+1)2^{d-1} + n^3 + n2^{d-1} + n^2 = \\ n^2 2^{d-1} + 2n2^{d-1} + n^3 + n^2 \qquad (6.19)$$

which is equivalent to $O(n^2 2^{d-1} + n^3)$.

In the case of Bluetooth, with $n = n_1 + n_2 + n_3 + n_4 = 128$ (where n_1 is the length of the first LFSR, n_2 is the length of the second LFSR, and so on) and *d=4*, we determine that the attack on E0 requires $2246656 \approx 2^{21.1}$ bit operations.

The number of operations needed for the computational process is considerable less than of similar algebraic attack ($2^{54.51}$ bit operations needed [3]) and correlation attack (2^{37} bit operations needed [2]) types, which we described in Chapter III. However, our cube-type attack is limited to the LFSRs' output at any clock tick.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSION

> We can only see a short distance ahead, but we can see plenty there that needs to be done.
>
> Alan Turing (1912–1954)

A. CONTRIBUTION

The main contribution of this thesis is as follows:

If an attacker has unauthorized access *to the encryption protocol, the attacker can use the encryption machine as an oracle so that he can manipulate some of the bits of the LFSRs, and knows the output bits of the E0 keystream generator, he can find the outputs of the individual LFSRs at any clock tick.*

In this study, we investigated the current types of attacks, like correlation and algebraic attacks, used in wireless systems. He focused on a new (introduced in 2008) and promising type of algebraic attack, namely the cube attack. We implemented the cube attack in a wireless system, namely Bluetooth. We modeled the encryption function of E0 and automated the process of the cube attack on E0. This included the factorization process (preprocessing phase) where an attacker finds as many maxterms as possible. In the actual attacking phase, the attacker solves the system of linear equations through a chosen plaintext attack and computes useful information about the cryptosystem. The number of operations needed for the computational process is of order $2^{21.1}$ bit operations and is considerably less than that of similar algebraic types of attacks, but is limited in finding the output of the LFSRs at any clock cycle.

A useful observation is the following. We have all these different types of attackers. Regardless of whether the attacker is a blackhat or greyhat or a whitehat hacker, a sufficient level of sophistication is required for the attacker to succeed on the implementation of the cube-type attack. A mixture of man-in-the-middle attack and a chosen plaintext attack, knowledge of the encryption function

of the target machine, and knowledge of the encryption protocol that is in use, comes to take place, thus increasing the difficulty of the attack.

B. FUTURE DIRECTIONS

Further studies may improve many aspects of this thesis. The most important question that needs to be answered is to determine how an attacker can recover the encryption key of E0 after learning the output bits of every LFSR that this study provides. Further investigation of the structure of E0 given in [28] is required to correlate the internal, initial state of the LFSRs, like the pure key, corresponding address, random number and the clocking bits that feed into the LFSRs during their initialization phase, and the output bits per clock tick.

Building on these results, the next stage of research is to validate our integration of the cube-type attack into the Bluetooth encryption protocol. As demonstrated in this research as well as other research, one needs to be able to understand and formally evaluate the strengths of a given cryptosystem and be able to evaluate the implementation of the cryptosystem to ensure that there are no flaws in the application of the cryptosystem. The cryptosystem and the protocol it uses may be good, but if poorly implemented they will most likely be untrustworthy.

Given the ubiquity of Wi-Fi and emerging adoption of Wi-Max, it is evident that more work needs to be done to understand the trustworthiness of wireless systems in terms of the strength of the underlying encryption protocols. These systems use different encryption algorithms and different ciphers than E0. One could follow our steps to implement the cube-type attack, like modeling the encryption function of these systems, and then execute the preprocessing phase and online phase and observe how effective this attack may be.

APPENDIX A. ENCRYPTION FUNCTION OF E0 IN FULL EXPANSION

From Equation (5.31), after doing the algebraic multiplication and addition, we end up with the detailed encryption function. We did not use any tool to gain the result, since the polynomial was not of high degree and the number of variables was manageable.

$$
\begin{aligned}
0 = {} & a \oplus b \oplus c \oplus d \oplus x_5x_6a \oplus x_5x_6b \oplus x_5x_6c \oplus x_5x_6d \oplus \\
& x_5x_7a \oplus x_5x_7b \oplus x_5x_7c \oplus x_5x_7d \oplus x_5x_8a \oplus x_5x_8b \oplus x_5x_8c \oplus x_5x_8d \oplus \\
& x_6x_7a \oplus x_6x_7b \oplus x_6x_7c \oplus x_6x_7d \oplus x_6x_8a \oplus x_6x_8b \oplus x_6x_8c \oplus x_6x_8d \oplus \\
& x_7x_8a \oplus x_7x_8b \oplus x_7x_8c \oplus x_7x_8d \oplus x_5x_6x_7x_8 \oplus x_5a \oplus x_5c \oplus x_5d \oplus x_5ab \oplus x_5bc \oplus x_5bd \oplus \\
& x_6a \oplus x_6c \oplus x_6d \oplus x_6ab \oplus x_6bc \oplus x_6bd \oplus x_7a \oplus x_7c \oplus x_7d \oplus x_7ab \oplus x_7bc \oplus x_7bd \oplus \\
& x_8a \oplus x_8c \oplus x_8d \oplus x_8ab \oplus x_8bc \oplus x_8bd \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \\
& x_1x_5 \oplus x_1x_6 \oplus x_1x_7 \oplus x_1x_8 \oplus x_2x_5 \oplus x_2x_6 \oplus x_2x_7 \oplus x_2x_8 \oplus \\
& x_3x_5 \oplus x_3x_6 \oplus x_3x_7 \oplus x_3x_8 \oplus x_4x_5 \oplus x_4x_6 \oplus x_4x_7 \oplus x_4x_8 \oplus \\
& x_1x_5b \oplus x_1x_6b \oplus x_1x_7b \oplus x_1x_8b \oplus x_2x_5b \oplus x_2x_6b \oplus x_2x_7b \oplus x_2x_8b \oplus \\
& x_3x_5b \oplus x_3x_6b \oplus x_3x_7b \oplus x_3x_8b \oplus x_4x_5b \oplus x_4x_6b \oplus x_4x_7b \oplus x_4x_8b \oplus \\
& x_1x_5x_6 \oplus x_1x_5x_7 \oplus x_1x_5x_8 \oplus x_1x_6x_7 \oplus x_1x_7x_8 \oplus x_2x_5x_6 \oplus x_2x_5x_7 \oplus x_2x_5x_8 \oplus x_2x_6x_7 \oplus x_2x_7x_8 \oplus \\
& x_3x_5x_6 \oplus x_3x_5x_7 \oplus x_3x_5x_8 \oplus x_3x_6x_7 \oplus x_3x_7x_8 \oplus x_4x_5x_6 \oplus x_4x_5x_7 \oplus x_4x_5x_8 \oplus x_4x_6x_7 \oplus x_4x_7x_8 \oplus \\
& x_9b \oplus x_{10}b \oplus x_{11}b \oplus x_{12}b \oplus x_9x_5cb \oplus x_9x_6cb \oplus x_9x_7cb \oplus x_9x_8cb \oplus \\
& x_{10}x_5cb \oplus x_{10}x_6cb \oplus x_{10}x_7cb \oplus x_{10}x_8cb \oplus x_{11}x_5cb \oplus x_{11}x_6cb \oplus x_{11}x_7cb \oplus x_{11}x_8cb \oplus \\
& x_{12}x_5cb \oplus x_{12}x_6cb \oplus x_{12}x_7cb \oplus x_{12}x_8cb \oplus x_9x_5c \oplus x_9x_6c \oplus x_9x_7c \oplus x_9x_8c \oplus \\
& x_{10}x_5c \oplus x_{10}x_6c \oplus x_{10}x_7c \oplus x_{10}x_8c \oplus x_{11}x_5c \oplus x_{11}x_6c \oplus x_{11}x_7c \oplus x_{11}x_8c \oplus x_{12}x_5c \oplus x_{12}x_6c \oplus x_{12}x_7c \oplus x_{12}x_8c \oplus \\
& x_9x_5x_6c \oplus x_9x_5x_7c \oplus x_9x_5x_8c \oplus x_9x_6x_7c \oplus x_9x_6x_8c \oplus x_9x_7x_8c \oplus \\
& x_{10}x_5x_6c \oplus x_{10}x_5x_7c \oplus x_{10}x_5x_8c \oplus x_{10}x_6x_7c \oplus x_{10}x_6x_8c \oplus x_{10}x_7x_8c \oplus \\
& x_{11}x_5x_6c \oplus x_{11}x_5x_7c \oplus x_{11}x_5x_8c \oplus x_{11}x_6x_7c \oplus x_{11}x_6x_8c \oplus x_{11}x_7x_8c \oplus \\
& x_{12}x_5x_6c \oplus x_{12}x_5x_7c \oplus x_{12}x_5x_8c \oplus x_{12}x_6x_7c \oplus x_{12}x_6x_8c \oplus x_{12}x_7x_8c \oplus \\
& x_9x_{10} \oplus x_9x_{11} \oplus x_9x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{12} \oplus x_{11}x_{12} \oplus \\
& x_9x_{10}x_5 \oplus x_9x_{10}x_6 \oplus x_9x_{10}x_7 \oplus x_9x_{10}x_8 \oplus x_9x_{11}x_5 \oplus x_9x_{11}x_6 \oplus x_9x_{11}x_7 \oplus x_9x_{11}x_8 \oplus \\
& x_9x_{12}x_5 \oplus x_9x_{12}x_6 \oplus x_9x_{12}x_7 \oplus x_9x_{12}x_8 \oplus x_{10}x_{11}x_5 \oplus x_{10}x_{11}x_6 \oplus x_{10}x_{11}x_7 \oplus x_{10}x_{11}x_8 \oplus
\end{aligned}
$$

$$x_{10}x_{12}x_5 \oplus x_{10}x_{12}x_6 \oplus x_{10}x_{12}x_7 \oplus x_{10}x_{12}x_8 \oplus x_{11}x_{12}x_5 \oplus x_{11}x_{12}x_6 \oplus x_{11}x_{12}x_7 \oplus x_{11}x_{12}x_8 \oplus$$
$$x_9x_{10}x_5b \oplus x_9x_{10}x_6b \oplus x_9x_{10}x_7b \oplus x_9x_{10}x_8b \oplus x_9x_{11}x_5b \oplus x_9x_{11}x_6b \oplus x_9x_{11}x_7b \oplus x_9x_{11}x_8b \oplus$$
$$x_9x_{12}x_5b \oplus x_9x_{12}x_6b \oplus x_9x_{12}x_7b \oplus x_9x_{12}x_8b \oplus x_{10}x_{11}x_5b \oplus x_{10}x_{11}x_6b \oplus x_{10}x_{11}x_7b \oplus x_{10}x_{11}x_8b \oplus$$
$$x_{10}x_{12}x_5b \oplus x_{10}x_{12}x_6b \oplus x_{10}x_{12}x_7b \oplus x_{10}x_{12}x_8b \oplus x_{11}x_{12}x_5b \oplus x_{11}x_{12}x_6b \oplus x_{11}x_{12}x_7b \oplus x_{11}x_{12}x_8b \oplus$$
$$(x_9x_{10}x_5x_6 \oplus x_9x_{10}x_5x_7 \oplus x_9x_{10}x_5x_8 \oplus x_9x_{10}x_6x_7 \oplus x_9x_{10}x_6x_8 \oplus x_9x_{10}x_7x_8 \oplus$$
$$x_9x_{11}x_5x_6 \oplus x_9x_{11}x_5x_7 \oplus x_9x_{11}x_5x_8 \oplus x_9x_{11}x_6x_7 \oplus x_9x_{11}x_6x_8 \oplus x_9x_{11}x_7x_8 \oplus$$
$$x_9x_{12}x_5x_6 \oplus x_9x_{12}x_5x_7 \oplus x_9x_{12}x_5x_8 \oplus x_9x_{12}x_6x_7 \oplus x_9x_{12}x_6x_8 \oplus x_9x_{12}x_7x_8 \oplus$$
$$x_{10}x_{11}x_5x_6 \oplus x_{10}x_{11}x_5x_7 \oplus x_{10}x_{11}x_5x_8 \oplus x_{10}x_{11}x_6x_7 \oplus x_{10}x_{11}x_6x_8 \oplus x_{10}x_{11}x_7x_8 \oplus$$
$$x_{10}x_{12}x_5x_6 \oplus x_{10}x_{12}x_5x_7 \oplus x_{10}x_{12}x_5x_8 \oplus x_{10}x_{12}x_6x_7 \oplus x_{10}x_{12}x_6x_8 \oplus x_{10}x_{12}x_7x_8 \oplus$$
$$x_{11}x_{12}x_5x_6 \oplus x_{11}x_{12}x_5x_7 \oplus x_{11}x_{12}x_5x_8 \oplus x_{11}x_{12}x_6x_7 \oplus x_{11}x_{12}x_6x_8 \oplus x_{11}x_{12}x_7x_8) \oplus$$
$$x_{13} \oplus x_{14} \oplus x_{15} \oplus x_{16} \oplus x_{13}x_5 \oplus x_{13}x_6 \oplus x_{13}x_7 \oplus x_{13}x_8 \oplus x_{14}x_5 \oplus x_{14}x_6 \oplus x_{14}x_7 \oplus x_{14}x_8 \oplus$$
$$x_{15}x_5 \oplus x_{15}x_6 \oplus x_{15}x_7 \oplus x_{15}x_8 \oplus x_{16}x_5 \oplus x_{16}x_6 \oplus x_{16}x_7 \oplus x_{16}x_8 \oplus x_{13}x_5b \oplus x_{13}x_6b \oplus$$
$$x_{13}x_7b \oplus x_{13}x_8b \oplus x_{14}x_5b \oplus x_{14}x_6b \oplus x_{14}x_7b \oplus x_{14}x_8b \oplus x_{15}x_5b \oplus x_{15}x_6b \oplus x_{15}x_7b \oplus x_{15}x_8b \oplus$$
$$x_{16}x_5b \oplus x_{16}x_6b \oplus x_{16}x_7b \oplus x_{16}x_8b \oplus x_{13}x_5x_6 \oplus x_{13}x_5x_7 \oplus x_{13}x_5x_8 \oplus x_{13}x_6x_7 \oplus x_{13}x_6x_8 \oplus x_{13}x_7x_8 \oplus$$
$$x_{14}x_5x_6 \oplus x_{14}x_5x_7 \oplus x_{14}x_5x_8 \oplus x_{14}x_6x_7 \oplus x_{14}x_6x_8 \oplus x_{14}x_7x_8 \oplus$$
$$x_{15}x_5x_6 \oplus x_{15}x_5x_7 \oplus x_{15}x_5x_8 \oplus x_{15}x_6x_7 \oplus x_{15}x_6x_8 \oplus x_{15}x_7x_8 \oplus$$
$$x_{16}x_5x_6 \oplus x_{16}x_5x_7 \oplus x_{16}x_5x_8 \oplus x_{16}x_6x_7 \oplus x_{16}x_6x_8 \oplus x_{16}x_7x_8 \oplus$$
$$x_5x_6x_7b \oplus x_5x_6x_8b \oplus x_5x_7x_8b \oplus x_6x_7x_8b \oplus x_5x_6 \oplus x_5x_7 \oplus x_5x_8 \oplus$$
$$x_6x_7 \oplus x_6x_8 \oplus x_7x_8 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8$$

Note: Glossary of E0 keystream generator is provided in Appendix D.

■

APPENDIX B. MAPLE 12

Working in the Maple 12 environment and after running the detailed program, we found twelve superpolys, including the unknown variables of the four LFSRs for two consecutive clock times. The program was executed several times for testing purposes on an Intel Pentium 4 processor with a CPU of 2.80 GHz and 1 GB of RAM, and the results were produced in a mean time of 8.03 seconds, consuming 5.25 MB of memory.

The structure of the program is simple. Using methods *prod2* and *prod3,* we take the integers that represent the variables of the encryption function and concatenate them to create products of variables. The *part* method takes as an input any product of variables and returns its remainder and the cofactor (superpoly). The *ptab* method stores the results in a table. Then we iterate through the table and output every unique linear, nonconstant co-factor and their corresponding products (maxterms).

In order to run this program one has to open a new worksheet in the Maple 12 environment and copy every paragraph that starts with the symbol " >" and ends with symbol ";" of the following Maple code along with its contents and paste it to the worksheet. Then he or she has to press symbol "*!!!*" from the taskbar to compile the code and continually do this process up to the last line of code. Comments starting with the symbol "//" must not be entered in the worksheet as it will cause an error.

MAPLE CODE

// The encryption function of E0 in Algebraic Normal Form in Maple syntax

```
> anf := a + b + c + d + X5*X6*a + X5*X6*b + X5*X6*c +
X5*X6*d + X5*X7*a + X5*X7*b + X5*X7*c + X5*X7*d + X5*X8*a +
X5*X8*b + X5*X8*c + X5*X8*d + X6*X7*a + X6*X7*b + X6*X7*c +
X6*X7*d + X6*X8*a + X6*X8*b + X6*X8*c + X6*X8*d + X7*X8*a +
X7*X8*b + X7*X8*c + X7*X8*d + X5*X6*X7*X8 + X5*a + X5*c +
X5*d + X5*a*b + X5*b*c + X5*b*d + X6*a + X6*c + X6*d +
X6*a*b + X6*b*c + X6*b*d + X7*a + X7*c + X7*d + X7*a*b +
X7*b*c + X7*b*d + X8*a + X8*c + X8*d + X8*a*b + X8*b*c +
```

X8*b*d + X1 + X2 + X3 + X4 + X1*X5 + X1*X6 + X1*X7 + X1*X8
+ X2*X5 + X2*X6 + X2*X7 + X2*X8 + X3*X5 + X3*X6 + X3*X7 +
X3*X8 + X4*X5 + X4*X6 + X4*X7 + X4*X8
+ X1*X5*b + X1*X6*b + X1*X7*b + X1*X8*b + X2*X5*b + X2*X6*b
+ X2*X7*b + X2*X8*b + X3*X5*b + X3*X6*b + X3*X7*b + X3*X8*b
+ X4*X5*b + X4*X6*b + X4*X7*b + X4*X8*b + X1*X5*X6 +
X1*X5*X7 + X1*X5*X8 + X1*X6*X7 + X1*X6*X8 + X1*X7*X8 +
X2*X5*X6 + X2*X5*X7 + X2*X5*X8 + X2*X6*X7 + X2*X6*X8 +
X2*X7*X8 + X3*X5*X6 + X3*X5*X7 + X3*X5*X8 + X3*X6*X7 +
X3*X6*X8 + X3*X7*X8 + X4*X5*X6 + X4*X5*X7 + X4*X5*X8 +
X4*X6*X7 + X4*X6*X8 + X4*X7*X8 + X9*b + X10*b + X11*b +
X12*b + X9*X5*c + X9*X6*c + X9*X7*c + X9*X8*c + X10*X5*c +
X10*X6*c + X10*X7*c + X10*X8*c + X11*X5*c + X11*X6*c +
X11*X7*c + X11*X8*c + X12*X5*c + X12*X6*c +
X12*X7*c + X12*X8*c + X9*X5*c*b + X9*X6*c*b + X9*X7*c*b +
X9*X8*c*b + X10*X5*c*b + X10*X6*c*b + X10*X7*c*b +
X10*X8*c*b + X11*X5*c*b + X11*X6*c*b + X11*X7*c*b +
X11*X8*c*b + X12*X5*c*b + X12*X6*c*b + X12*X7*c*b +
X12*X8*c*b + X9*X5*X6*c + X9*X5*X7*c + X9*X5*X8*c +
X9*X6*X7*c + X9*X6*X8*c + X9*X7*X8*c + X10*X5*X6*c +
X10*X5*X7*c + X10*X5*X8*c + X10*X6*X7*c + X10*X6*X8*c +
X10*X7*X8*c + X11*X5*X6*c + X11*X5*X7*c + X11*X5*X8*c +
X11*X6*X7*c + X11*X6*X8*c + X11*X7*X8*c + X12*X5*X6*c +
X12*X5*X7*c + X12*X5*X8*c + X12*X6*X7*c + X12*X6*X8*c +
X12*X7*X8*c + X9*X10 + X9*X11 + X9*X12 + X10*X11 +
X10*X12 + X11*X12 + X9*X10*X5 + X9*X10*X6 + X9*X10*X7 +
X9*X10*X8 + X9*X11*X5 + X9*X11*X6 + X9*X11*X7 + X9*X11*X8 +
X9*X12*X5 + X9*X12*X6 + X9*X12*X7 + X9*X12*X8 + X10*X11*X5
+ X10*X11*X6 + X10*X11*X7 + X10*X11*X8 + X10*X12*X5 +
X10*X12*X6 + X10*X12*X7 + X10*X12*X8 + X11*X12*X5 +
X11*X12*X6 + X11*X12*X7 + X11*X12*X8 + X9*X10*X5*b +
X9*X10*X6*b + X9*X10*X7*b + X9*X10*X8*b + X9*X11*X5*b +
X9*X11*X6*b + X9*X11*X7*b + X9*X11*X8*b + X9*X12*X5*b +
X9*X12*X6*b + X9*X12*X7*b + X9*X12*X8*b + X10*X11*X5*b +
X10*X11*X6*b + X10*X11*X7*b + X10*X11*X8*b + X10*X12*X5*b +
X10*X12*X6*b + X10*X12*X7*b + X10*X12*X8*b + X11*X12*X5*b +
X11*X12*X6*b + X11*X12*X7*b + X11*X12*X8*b + X9*X10*X5*X6 +
X9*X10*X5*X7 + X9*X10*X5*X8 + X9*X10*X6*X7 + X9*X10*X6*X8 +
X9*X10*X7*X8 + X9*X11*X5*X6 + X9*X11*X5*X7 + X9*X11*X5*X8 +
X9*X11*X6*X7 + X9*X11*X6*X8 + X9*X11*X7*X8 + X9*X12*X5*X6 +
X9*X12*X5*X7 + X9*X12*X5*X8 + X9*X12*X6*X7 + X9*X12*X6*X8 +
X9*X12*X7*X8 + X10*X11*X5*X6 + X10*X11*X5*X7 +
X10*X11*X5*X8 + X10*X11*X6*X7 + X10*X11*X6*X8 +
X10*X11*X7*X8 + X10*X12*X5*X6 + X10*X12*X5*X7 +
X10*X12*X5*X8 + X10*X12*X6*X7 + X10*X12*X6*X8 +

```
X10*X12*X7*X8   +   X11*X12*X5*X6   +   X11*X12*X5*X7   +
X11*X12*X5*X8   +   X11*X12*X6*X7   +   X11*X12*X6*X8   +
X11*X12*X7*X8 + X13 + X14 + X15 + X16 + X13*X5 + X13*X6 +
X13*X7 + X13*X8 + X14*X5 + X14*X6 + X14*X7 + X14*X8 +
X15*X5 + X15*X6 + X15*X7 + X15*X8 + X16*X5 + X16*X6 +
X16*X7 + X16*X8 + X13*X5*b + X13*X6*b + X13*X7*b
+ X13*X8*b + X14*X5*b + X14*X6*b + X14*X7*b + X14*X8*b +
X15*X5*b + X15*X6*b + X15*X7*b + X15*X8*b + X16*X5*b +
X16*X6*b + X16*X7*b + X16*X8*b + X13*X5*X6 +
X13*X5*X7 + X13*X5*X8 + X13*X6*X7 + X13*X6*X8 + X13*X7*X8 +
X14*X5*X6 + X14*X5*X7 + X14*X5*X8 + X14*X6*X7 + X14*X6*X8 +
X14*X7*X8 + X15*X5*X6 +
X15*X5*X7 + X15*X5*X8 + X15*X6*X7 + X15*X6*X8 + X15*X7*X8 +
X16*X5*X6 + X16*X5*X7 + X16*X5*X8 + X16*X6*X7 + X16*X6*X8 +
X16*X7*X8  +  X5*X6*X7*b  +  X5*X6*X8*b  +  X5*X7*X8*b  +
X6*X7*X8*b + X5*X6 + X5*X7 + X5*X8 + X6*X7 + X6*X8 +
X7*X8 + X5 + X6 + X7 + X8;
```

// prod2 & prod3 take integers and return a product of those X variables

```
> prod2 := (n,m) -> cat(X,n) * cat(X,m);

> prod3 := (n,m,o) -> cat(X,n) * cat(X,m) * cat(X,o);
```

// parts takes a product p and returns a list of 2 parts: remainder and cofactor

```
> parts := proc( p ) global anf; local l, z, t;
l := coeffs( algsubs( p = z, anf ), z, 't' );
if nops([l]) = 1 then [l,0];
else if t[1] = 1 then [ l ];
else [ l[2], l[1] ];
end if; end if; end proc;
```

// set up table "ptab" of these parts, indexed by the integers

```
> ptab := table();

> for i to 15 do for j from i+1 to 16 do
ptab[i,j] :=  parts( prod2(i,j) ) ;
end do; end do;
> for i to 14 do for j from i+1 to 15 do for k from j+1 to
16 do
ptab[i,j,k] :=  parts( prod3(i,j,k) ) ;
end do; end do; end do;
> degree(%);
> degree(%);
> for i in indices(ptab) do
if ( degree( ptab[op(i)][2]) = 1 ) then
```

```
print(i);print(ptab[op(i)][2]);print(ptab[op(i)][1]);    end
if; end do;
> whattype(indices(ptab));
> linfac := select( i -> ( degree( ptab[op(i)][2]) = 1 ),
[indices(ptab)] ):
> nops(linfac);

> ptab[op(linfac[1])][2];

> sort([seq( ptab[op(i)][2], i in linfac)]);


> linfacs := convert(%,set);

> linfacs := convert(linfacs, list);

> nops(linfacs);
> for fac in linfacs do
print(fac);
for i in linfac do
if ( ptab[op(i)][2] = fac ) then print(i); end if;
end do;
end do;
```

Note: In order for one to add comments to the worksheet from the Insert menu of the taskbar, one has to select Paragraph, and then select Before Cursor or After Cursor. A new paragraph is inserted and the cursor is moved to the new blank line. From there, one can enter the paragraph.

APPENDIX C. PROGRAM OUTPUT

Maple 12 works on Windows (2000, 2003, XP, Vista), Macintosh, UNIX, Linux and Solaris environments. The developers' system recommendations include the following:

- CPU: AMD X86_64/ 1 GHz/Intel Xeon/ Intel 64
- RAM: 512MB (at least)
- Hard disk: 1 GB

The program outlined in Appendix B was executed on an Intel Pentium 4 processor with a CPU of 2.80 GHz and 1 GB of RAM in a Windows XP environment. The output of the program is in the following paragraph where the linear term without any bracket represents the superpoly and the terms inside the brackets represent the corresponding index of the variables of the corresponding superpoly. For example, the superpoly $x_5 \oplus b$ has only one maxterm, $x_6x_7x_8$, whereas the superpoly $x_9 \oplus x_{11} \oplus x_{12} \oplus c$ *has as maxterms the terms* $x_5x_6x_{10}, x_6x_7x_{10}, x_5x_7x_{10}, x_6x_8x_{10}, x_5x_8x_{10}, x_7x_8x_{10}$.

OUTPUT

$$X5 + b$$
$$[6, 7, 8]$$
$$X6 + b$$
$$[5, 7, 8]$$
$$X7 + b$$
$$[5, 6, 8]$$
$$b + X8$$
$$[5, 6, 7]$$
$$X11 + X12 + X9 + c$$
$$[5, 6, 10]$$
$$[6, 7, 10]$$
$$[5, 7, 10]$$
$$[6, 8, 10]$$
$$[5, 8, 10]$$
$$[7, 8, 10]$$

$$X11 + X12 + c + X10$$
$$[7, 8, 9]$$
$$[6, 8, 9]$$
$$[5, 6, 9]$$
$$[6, 7, 9]$$
$$[5, 7, 9]$$
$$[5, 8, 9]$$
$$X9 + c + X10 + X11$$
$$[5, 8, 12]$$
$$[5, 6, 12]$$
$$[5, 7, 12]$$
$$[6, 8, 12]$$
$$[7, 8, 12]$$
$$[6, 7, 12]$$
$$c + X10 + X12 + X9$$
$$[6, 7, 11]$$
$$[7, 8, 11]$$
$$[6, 8, 11]$$
$$[5, 8, 11]$$
$$[5, 7, 11]$$
$$[5, 6, 11]$$
$$1 + X6 + X7 + X8 + b$$
$$[1, 5]$$
$$[5, 11, 12]$$
$$[5, 10, 12]$$
$$[5, 16]$$
$$[5, 9, 11]$$
$$[5, 10, 11]$$
$$[5, 13]$$
$$[3, 5]$$
$$[2, 5]$$
$$[5, 9, 12]$$
$$[5, 15]$$
$$[5, 9, 10]$$
$$[4, 5]$$
$$[5, 14]$$
$$1 + X6 + X8 + b + X5$$
$$[7, 10, 12]$$
$$[7, 16]$$
$$[7, 9, 11]$$
$$[7, 9, 12]$$
$$[7, 9, 10]$$
$$[7, 10, 11]$$
$$[7, 11, 12]$$
$$[7, 13]$$

[7, 15]
[4, 7]
[2, 7]
[3, 7]
[7, 14]
[1, 7]
$1 + X7 + X8 + b + X5$
[6, 10, 11]
[1, 6]
[6, 9, 12]
[6, 11, 12]
[6, 16]
[6, 9, 10]
[6, 15]
[2, 6]
[6, 13]
[4, 6]
[3, 6]
[6, 14]
[6, 10, 12]
[6, 9, 11]
$1 + b + X5 + X6 + X7$
[4, 8]
[3, 8]
[8, 10, 12]
[8, 9, 11]
[8, 16]
[1, 8]
[8, 9, 12]
[8, 15]
[8, 14]
[8, 13]
[8, 9, 10]
[8, 10, 11]
[8, 11, 12]
[2, 8]

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. GLOSSARY OF BLUETOOTH KEY STREAM GENERATOR E0

K_c …….. Encryption Key

COF ……Encryption Offset Number

OR……Bitwise OR

XOR…….Bitwise Exclusive OR

LSFR……..Linear Feedback Shift Register

CLK……Master Clock Bits

x_t^i ……Output bit of the $LFSR_i$ at clock-time *t*

$y_t = \sum_{i-1}^{4} x_t^i$ ……Summation outcome (integer) from the output bits of the four LFSRs at clock-time *t*

z_t ……keystream bit produced by E0 at clock-time *t*

z_{t+1} ……..keystream bit produced by E0 at clock-time *t+1*

z_{t+2} …… keystream bit produced by E0 at clock-time *t+2*

z_{t+3} ……..keystream bit produced by E0 at clock-time *t+3*

c_t ……..……Four Memory bits at clock-time *t*

c_t^1 ……...Current two-bit block of Memory bit at clock-time *t*

c_t^0 ……...…Two-bit block of Memory bits at clock-time *t-1*

s_t ……Two-bit sequence

s_t^1 …….....First bit of the two-bit sequence

s_t^0 ……..……Second bit of the two-bit sequence

q_t …….First bit of the current two-bit block of Memory bits at clock time *t*

p_t …...Second bit of the current two-bit block of Memory bits at clock time *t*

q_{t-1} …….First bit of the two-bit block of Memory bits at clock time *t-1*

p_{t-1} …….Second bit of the two-bit block of Memory bits at clock time *t-1*

Π_t^i ……the *i*-th elementary symmetric polynomial in x_t^i

x_1, x_2, x_3, x_4 ...The outputs of the 1st,…,4th LFSR at clock-time *t* respectively.

x_5, x_6, x_7, x_8 ...The outputs of the 1st,…,4th LFSR at clock-time *t+1* respectively.

$x_9, x_{10}, x_{11}, x_{12}$ ...The outputs of the 1st,…,4th LFSR at clock-time *t+2* respectively.

$x_{13}, x_{14}, x_{15}, x_{16}$ ...The outputs of the 1st,…,4th LFSR at clock-time *t+3* respectively.

a……keystream bit produced by E0 at clock-time *t,* z_t

b……keystream bit produced by E0 at clock-time *t+1,* z_{t+1}

c……keystream bit produced by E0 at clock-time *t+2,* z_{t+2}

d……..keystream bit produced by E0 at clock-time *t+3,* z_{t+3}

LIST OF REFERENCES

[1] Y. Lu and S. Vaudenay, “Faster correlation attack on Bluetooth keystream generator E0,” *Lecture Notes in Computer Science,* vol. 3152, pp. 407–425, December 2004.

[2] Y. Lu, W. Meier, and S. Vaudenay, “The conditional correlation attack: A practical attack on Bluetooth encryption,” Lecture Notes in Computer Science, vol. 3621, pp. 97–117, August 2005.

[3] F. Armknecht, July 24–28, 2004, “An algebraic attack on the Bluetooth key stream generator,” Minrank Foundation, *tutorial presented at ECCOMAS*, http://th.informatik.unimannheim.de/people/armknecht/Armknecht_Eccomas.pdf (accessed February 3, 2009).

[4] I. Dinur and A. Shamir, January 26, 2009, “Cube attacks on tweakable black box polynomials, *Cryptology ePrint Archive*, http://eprint.iacr.org/2008/385 (accessed January 7, 2009).

[5] A. Zhang, C.-W. Lim, K. Khoo, W. Lei, and J. Pieprzyk, September 2, 2009, **“**Extensions of the cube attack on low degree annihilators,” *Cryptology ePrint Archive*, http://eprint.iacr.org/2009/049, to appear in proceedings of *Cryptology and Network Security – CANS 2009* , Lecture Notes in Computer Science, Springer–Verlag, 2009. (accessed September 15, 2009).

[6] R. Needham and M. Schroeder, “Using encryption for authentication in large networks of computers,” *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, December 1978.

[7] B. Giampaolo, “Formal Correctness of Security Protocols,” *Springer*, April 2007, p. 2.

[8] D. R. Stinson, “Cryptography Theory and Practice,” *Chapman & Hall/CRC*, 3rd Edition, 2006, p.1.

[9] R. A. Mollin, “Codes: The guide to secrecy from ancient to modern time,” *CRC Press*, May 2005, pp. 379–393.

[10] A. D. Rublin, “White-hat security arsenal: tackling the threats,” *Addison Wesley*, 2001, p. 227–253.

[11] W. Cha, G. Wang, and G. Cho, “A Pai-Wise key Agreement Scheme*,” Lecture Notes in Computer Science*, vol. 3036, pp. 648–651, May 2004.

[12] J. B. Farleigh, “A First Course in Abstract Algebra,” *Addison Wesley*, 7th Edition, 2003, pp. 274–275.

[13] T. W. Cusick and P. Stanica, “Cryptographic Boolean functions and applications,” *Academic Press in-Elsevier*, March 2009, pp. 5–24.

[14] K. H. Rosen, “Discrete mathematics and its applications,” *McGraw Hill*, 6th Edition, 2007, p. 116.

[15] T. Siegenthaler, “Correlation-immunity of nonlinear combining functions for cryptographic applications,” *IEEE Transactions on Information theory*, Vol. IT-30, No. 5, 1984, pp. 776–780.

[16] N. Courtois and W. Meier, “Algebraic Attacks on Stream Ciphers with Linear Feedback,” *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, vol. 2729, pp. 176–194, October 2003.

[17] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, “Efficient Algorithms for Solving an Overdifined Systems of Multivariate Polynomial Equations,” Advances in Cryptology–Eurocrypt 2000, *Lecture Notes in Computer Science*, Springer, vol. 1807, pp. 392–407, Springer, January 2000.

[18] N. Coutrois and W. Meier, “Algebraic Attacks on Stream Ciphers with Linear Feedback,” Eurocrypt 2003, *Lecture Notes in Computer Science*, Springer, vol. 2656, pp. 345–359, Springer, Extended version of the paper at http://www.nicolascourtois.me.uk/ , September 4, 2008, (accessed April 10, 2009).

[19] W. Meier, E. Pasalic, and C. Carlet, “Algebraic Attacks and Decomposition of Boolean functions,” *Lecture Notes in Computer Science,* vol. 3027, Advances in Cryptology–Eurocrypt 2004, pp. 474–471, April 2004.

[20] M.R. Garey and D.S. Johnson, “Computers and Intractability: A guide to the Theory of NP-Completeness,” W.H. Freeman, January 1979.

[21] B. Buchberger, “Gröbner Bases and Applications,” B. Buchberger, F. Winkler (eds.), *London Mathematical Society Lecture Notes Series 251* , Cambridge University Press, March 1998, pp. 535–545.

[22] F. Armknecht and M. Krause, “Algebraic attacks on combiners with memory,” *Adv. in Cryptology – Crypto 2003, Lecture Notes in Computer Science* 2729, pp. 162–175, October 2003.

[23] F. Armknecht, “Algebraic Attacks on Stream Ciphers,” *Proc. European Congress on Computational Methods in Applied Sciences and Engineering*, 2004, http://www.mit.jyu.fi/eccomas2004/proceedings/pdf/509.pdf (accessed February 3, 2009).

[24] S. Fischer, S. Khazaei, and W. Meier, “Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers,” *Progress in Cryptology – AFRICACRYPT 2008, Lecture Notes in Computer Science* 5023, pp. 236–245, May 2008.

[25] A. Joux and F. Muller, “A chosen IV Attack Against Turing,” *Selected areas in Cryptography, Lecture Notes in Computer Science* 3006, pp. 194–207, May 2004.

[26] S. O. Neil, September 23, 2007, “Algebraic Structure Defectoscopy,” *Cryptology ePrint Archive*, Report 2007/738, http://eprint.iacr.org/2007/378 (accessed March 15, 2009).

[27] S. Fischer, S. Khazaei, and W. Meier, “Reduced Complexity Attacks on the Alternating Step Generator,” *Selected areas in Cryptography, Lecture Notes in Computer Science* 4876, pp. 1–16, December 2007.

[28] *Bluetooth Specification* Version 3.0 High Speed Vol2, Bluetooth SIG, adopted 21 April 2009, pp. 934–976, http://www.bluetooth.com/Bluetooth/ Technology/Basics.htm (accessed April 29, 2009).

[29] Maplesoft, a division of Waterloo Maple Inc. 2009, http://www.maplesoft.com/documentation_center/maple12/Install.html#Windows_ System_Requirements, (accessed, April 2, 2009).

[30] Maple 12 User Manual, Maplesoft, a division of Waterloo Maple Inc. 1996–2008. pp. xiii,–xiv, http://www.maplesoft.com/ view. aspx? sid=5883, (accessed, April 4, 2009).

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. George Dinolt
   Department of Computer Science
   Monterey, California

4. Dr. James Bret Michael
   Department of Computer Science
   Monterey, California

5. Dr. Pantelimon Stanica
   Department of Applied Mathematics
   Monterey, California

6. Dr. David Canright
   Department of Applied Mathematics
   Monterey, California

7. Dr. Peter J. Denning
   Chairman, Department of Computer Science
   Monterey, California

8. Dr. Carlos F. Borges
   Chairman, Department of Applied Mathematics
   Monterey, California

9. Hellenic Navy General Staff
   Athens, Greece

10. Hellenic Naval Academy
    Piraeus, Greece

11. LT Nikolaos Petrakos
    Naval Postgraduate School
    Monterey, California