情報保護學會論文誌
第15券 第1號, 2005. 2

소프트웨어 구현에 적합한 스트림 암호의 대수적 공격에 대한 안전성

성 재 철,[1†] 문 덕 재,[2‡] 임 흥 수,[3] 지 성 택,[2] 이 상 진[3]

[1]서울시립대학교 수학과, [2]국가보안기술연구소, [3]고려대학교 정보보호기술연구센터

Security Analysis of Software-Oriented Stream Ciphers against Algebraic Attacks

Jaechul Sung,[1†] Dukjae Moon,[2‡] Hung-su Im,[3] Seongtaek Chee,[2] Sangjin Lee[3]

[1]University of Seoul, [2]National Security Research Institute, [3]Korea University

요 약

본 논문에서는 최근 제안된 소프트웨어 구현에 적합한 (블록 단위) 스트림 암호인 HELIX, SCREAM, MUGI, PANAMA 알고리즘들의 대수적 공격에 대한 안전성을 분석한다.[14,16,24,10] 대수적 공격은 알려진 입출력 쌍을 가지고 내부 알고리즘의 기본 대수 방정식을 이용하는 방법으로, 과포화된 다변수 연립 방정식을 통하여 변수의 값을 얻고 이를 이용하여 키를 복구해내는 방법이다.[3,4,7,8,18] 이 분석법은 초기 대수적 특성을 지닌 블록 암호의 분석에 적용되었으나, 이후 블록 암호보다는 스트림 암호의 분석에 보다 용이하게 적용되었다.[1,5,6] 그러나 일반적인 알고리즘을 이 분석법을 그대로 적용하기에는 많은 어려움이 존재한다. 따라서 본 논문에서는 최근 제안된 워드 단위의 소프트웨어 구현에 적합한 스트림 암호의 각 알고리즘에 대해 대수적 방정식과 변수의 수를 비교 및 분석함으로써 각 알고리즘의 대수적 공격에 대한 내성을 살펴본다. 또한 이러한 분석을 통해 대수적 공격에 안전한 소프트웨어 구현에 적합한 스트림 암호의 설계 시 고려해야 할 세 가지 설계 고려사항을 제시한다.

ABSTRACT

In this paper we consider the security of recently proposed software-oriented stream cipher HELIX, SCREAM, MUGI, and PANAMA against algebraic attacks.[14,15,24,10] Algebraic attack is a key recovery attack by solving an over-defined system of multi-variate equations with input-output pairs of an algorithm.[3,4,7,8,18] The attack was firstly applied to block ciphers with some algebraic properties and then it has been more usefully applied to stream ciphers.[1,5,6] However, it is difficult to obtain over-defined algebraic equations for a given cryptosystem in general. Here we analyze recently proposed software-oriented stream ciphers by constructing a system of equations for each cipher. Furthermore we propose three design considerations of software-oriented stream ciphers.

Keywords : *Software-Oriented Stream ciphers, Algebraic attacks, HELIX, SCREAM, MUGI, PANAMA*

접수일 : 2004년 10월 1일 ; 채택일 : 2005년 1월 24일

* 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성·지원사업의 연구결과로 수행되었음

† 주저자, jcsung@uos.ac.kr

‡ 교신저자, djmoon@etri.re.kr

Ⅰ. 서 론

스트림 암호는 짧은 길이의 난수열을 이용하여 긴 길이의 의사-난수열인 키 스트림을 생성하는 대칭키 암호이다. 스트림 암호는 평문과 키 스트림의 연관성 유무에 따라 동기식(synchronous)과 비동기식(asynchronous)으로 구분된다.

DES와 같은 블록 암호는 개발 당시 하드웨어에 구현이 용이하도록 설계되었으나, 1990년대 이후 암호 기술이 일반화되면서 AES 등과 같이 소프트웨어적으로 구현이 용이한 알고리즘이 주로 설계되고 있다. 스트림 암호의 경우에도 과거에는 주로 하드웨어 구현이 용이한 LFSR(Linear Feedback Shift Register)에 기반을 둔 알고리즘이 주로 설계되어 왔으나, 근래에는 다양한 응용 환경의 개발과 인터넷 서비스에서 스트리밍 기법이 다수 개발되면서 블록 암호보다 고속 동작이 가능한 소프트웨어 구현에 적합한 스트림 암호 개발이 증가하고 있다. 일반적으로 스트림 암호는 블록 암호보다 5~10배 정도 빠르게 구현되는 장점을 가지고 있다.

1970년대 유럽을 중심으로 LFSR 기반의 스트림 암호에 대한 개발 및 연구가 본격화되었다. LFSR 자체는 선형이기 때문에 비선형적 논리와 결합하여 스트림 암호를 설계한다. 사용된 비선형적 논리에 따라 LFSR 기반의 스트림 암호는 비선형 필터 생성기, 비선형 결합 생성기, 시각 제어 생성기로 분류된다.

하드웨어 구현이 용이한 LFSR 기반의 스트림 암호와는 달리, RC4[20]나 SEAL[21,22]와 같은 스트림 암호는 소프트웨어 구현이 용이하도록 개발되었다. 즉, 이러한 알고리즘들은 비트별 연산을 수행하는 LFSR 기반의 스트림 암호와 달리 8-비트나 32-비트 등의 워드별 연산을 수행한다. 이러한 스트림 암호를 일반적으로 소프트웨어 구현에 적합한(Software-Oriented) 스트림 암호라 한다.

스트림 암호의 공격은 그 공격의 목표에 따라 구별공격(Distinguishing Attack), 예측 공격(Prediction Attack), 키복구 공격(Key Recovery Attack)으로 분류된다. 이러한 공격을 위해 주로 사용되는 분석법으로는 상관 공격(Correlation Attack), 분할 정복 공격(Divide-and-Conquer Attack), 추측 결정 공격(Guess-and-Determine Attack), Time-Memory Trade-Off 공격 등이 있다.

소프트웨어 구현에 적합한 스트림 암호는 기존의 LFSR 기반의 스트림 암호의 안전한 설계 논리와 블록 암호나 해쉬 함수 등에서의 효율적이고 안전한 암호 논리를 결합하여 설계된다. 따라서 이러한 최근 제안되고 있는 소프트웨어 구현에 적합한 스트림 암호는 이러한 전통적인 스트림 암호의 분석 방법으로는 효과적으로 분석되지 않는다.

소프트웨어 구현에 적합한 스트림 암호의 효과적인 분석법으로 최근 주목받고 있는 방법은 대수적 공격(Algebraic Attack)이다.[3,18] 대수적 공격은 내부의 미지의 변수와 알려진 입출력과의 관계인 대수적 방정식을 만들고, 이를 분석하여 대수적 방정식을 선형화(Linearization)에 의해 내부 변수를 풀어냄으로써 공격하는 방법이다.

대수적 공격은 초기에 공개키 알고리즘인 HFE에 적용되었고, 이후 SERPENT, AES와 같은 대수적 성질을 가지는 블록 암호의 분석에 적용되었다.[4,7,8] 이 공격법은 이후 블록 암호 보다 적용이 용이한 LILI-128, E0 등의 여러 스트림 암호의 분석에 사용되었다.[5,6]

기존의 대칭키 암호의 기존 분석법은 대부분 확률분포의 비균일성을 이용한 확률론적 공격인 반면, 대수적 공격은 다음과 같은 특성을 가지고 있다.

- 주어진 알고리즘에 대한 대수적 방정식들이 과포화(Overdefined, 방정식의 개수가 변수의 개수가 많은 경우)라면 대수적 공격은 확률 1로 내부 변수(초기 상태 혹은 마스터 키)의 값을 찾을 수 있다.
- 대수적 공격의 복잡도는 오직 내부 변수의 개수에만 의존하고 변수 값을 구하기 위한 복잡도는 변수 개수의 증가에 따라 지수적으로는 증가하지는 않는다.

위와 같은 특성으로 인해 대수적 공격을 효과적으로 적용하기 위한 문제는 크게 두 가지로 생각할 수 있다. 하나는 낮은 차수의 대수적 방정식을 찾는 문제이고, 다른 하나는 연립 방정식을 효율적으로 계산하여 해를 찾는 문제이다.

공격 복잡도가 방정식의 차수에 크게 의존하므로, 낮은 차수의 방정식을 어떻게 효과적으로 많이 찾을 수 있느냐가 대수적 공격의 큰 관건이다. 이러한 낮은 차수의 대수적 방정식을 찾는 방법에 대한 연구는 꾸준히 진행되고 있다.[5] 또한 연립 방정식을 효

율적으로 계산하는 문제는 XL 알고리즘과 Grobner basis 알고리즘 등을 이용하여 연구가 진행되고 있다.

본 논문에서는 최근 제안된 소프트웨어 구현에 적합한 스트림 암호 중 블록 암호의 구조나 워드단위의 LFSR 구조로 현재까지 효과적인 분석법이 발견되지 않고 있는 스트림 암호 HELIX,[14] SCREAM,[16] PANAMA,[10] MUGI[24]에 대해 대수적 공격의 공격 복잡도에 중요한 역할을 하는 대수적 방정식을 찾는 문제를 다룬다. 이를 위해 각 알고리즘에 대해 대수적 방정식을 구성하고, 연립 방정식의 해를 구하는데 영향을 미치는 각 변수들 간의 성질을 살펴보고 분석한다. 또한, 이러한 분석을 바탕으로 대수적 분석에 안전한 스트림 암호의 설계 시 고려되어야 할 세 가지 조건을 제시한다.

Ⅱ. 소프트웨어 구현에 적합한 스트림 암호

암호 기술이 전 세계적으로 일반화된 1990년대부터 소프트웨어 구현이 용이한 스트림 암호가 등장하기 시작하였다. 최초로 알려진 알고리즘이 RSA사에서 개발한 RC4[20]와 IBM에서 개발한 SEAL[21,22]이다. RC4의 경우 최초 개발 당시에는 알고리즘 구조에 대해서는 비밀이었으나, 1990년대 중반 알고리즘에 대한 구조가 알려진 후, 많은 공격법들이 소개되었다. 1993년 개발된 SEAL 알고리즘은 블록 암호와 해쉬 함수에서 사용되는 안전한 암호 논리를 이용하여 설계되었으나, 최초 버전 1.0의 경우 통계 분석을 통해 해독되었다.

RC4와 SEAL의 등장은 소프트웨어 구현에 적합한 스트림암호의 개발 및 분석에 대한 시발점이 되었다. 1990년대 들어 다양한 설계 논리를 바탕으로 한 스트림 암호가 등장하였다. 블록 암호에 비해 소프트웨어 구현에 적합한 스트림 암호는 기존의 블록 암호의 설계 논리, 해쉬 함수의 설계 논리, LFSR의 설계 논리 등을 바탕으로 설계되고 있는 실정이다. 1990년대 말부터 최근까지 개발된 소프트웨어 구현에 적합한 스트림 암호는 약 10 여에 이르나, 이에 대한 안전성 분석은 제대로 이루어지지 않고 있는 실정이다. 또한, 유럽의 NESSIE[18] 프로젝트나 일본의 CRYPTREC[9] 프로젝트에서도 소프트웨어 구현에 적합한 스트림 암호에 대한 공모를 수행하였다. NESSIE 프로젝트에서는 SOBER, SNOW 등의 알고리즘이 제안되었으나, 제안된 모든 알고리즘에 대한 안전성의 결함이 제기되어 어떠한 알고리즘도 선정되지는 못했다. CRYPTREC에서도 다수의 알고리즘들이 제안되었으나, 최종적으로는 MULTI-S01, MUGI, 128-비트 RC4 만이 선정되었다.

최근 제안된 이러한 소프트웨어 구현에 적합한 스트림 암호를 각 알고리즘에 사용된 설계 논리 및 구조의 유사성에 따라 다음과 같이 크게 네 가지로 분류할 수 있다.

- SEAL-계열 : SEAL,[21] TWOPRIME,[11] **SCREAM**,[16] **HELIX**[14]
- SOBER-계열 : SOBER,[17] SSC2,[26] SNOW,[13] TURING[23]
- PANAMA-계열 : **PANAMA**,[10] MULTI-S01,[15] **MUGI**[24]
- 기타 : RC4,[20] BEEPBEEP,[12] RABBIT,[2] VMPC,[27] HC-256[25]

SEAL-계열 스트림 암호는 블록 암호의 라운드 함수의 설계 논리를 기반으로 하여 설계된 알고리즘으로, 기존의 블록 암호에서 안전성의 핵심 논리로 사용되는 S-박스를 사용한다. 또한 확산 효과를 내기위해 기존 블록 암호의 로테이션 등의 연산을 사용한다. SEAL-계열 스트림 암호 중 SEAL과 TWOPRIME의 경우 전수조사보다 효과적인 분석법이 발견되었으나, HELIX와 SCREAM의 경우에는 아직까지 효과적인 분석법은 발견되지 않았다.

SOBER-계열 스트림 암호는 비트 단위로 작동되는 LFSR 기반 논리를 워드 단위로 작동 가능하도록 설계되어진 알고리즘이다. 워드 단위로 효과적으로 작동되는 LFSR 논리를 이용하여 주기 및 선형 복잡도에 대한 안전성을 확보하였고, LFSR의 적당한 워드 값들을 이용하여 비선형 함수를 통과한 후 워드 단위의 키 스트림을 생성한다. 또한, 블록 암호에 사용되는 S-박스 논리도 사용한다.

PANAMA-계열 스트림 암호는 SOBER-계열의 스트림 암호와 비슷한 논리로 설계된 알고리즘들이다. 다만, MUGI를 제외한 PANAMA와 MULTI-S01의 경우 SOBER 계열과는 달리 S-박스를 사용하지 않는다. MULTI-S01의 경우에는 PANAMA 알고리즘을 그대로 사용하여 키 스트림을 생성한다. PANAMA-계열과 SOBER-계열은 크게 워드 단위의 LFSR 구조로 통합하여 분류할 수도 있다.

기타 스트림 암호로는 RC4와 유산한 구조의 HC-256과 VMPC 등이 있다. 다른 계열의 스트림

암호 구조와는 달리 기타 계열의 스트림 암호는 아직까지 구조적인 안전성의 대한 신뢰가 부족한 상태이다. 또한, RC4를 제외한 이러한 스트림 암호는 최근 제안되어 안전성에 대한 분석은 거의 이루어지지 않는 실정이다. 특히 RC4 계열의 Table 변환 과정을 이용한 스트림 암호 VMPC, HC-256의 경우는 대수적 공격 기법을 적용하여 분석하기는 많은 어려움을 가지고 있다.

SEAL-계열 중 SEAL과 TWOPRIME은 구조적인 취약성으로 분석된 상태이며, SOBER-계열은 거의 모든 알고리즘의 취약성이 발견된 상태이다. PANAMA-계열에 대한 알고리즘은 아직까지 뚜렷한 분석 결과는 없는 실정이다.

따라서 본 논문에서는 기타 계열을 제외하고 아직까지 구조적인 약점이 발견되지 않는 나머지 스트림 암호에 대한 대수적 분석을 시도한다. 즉, HELIX, SCREAM, MUGI, PANAMA 알고리즘에 대한 대수적 공격의 내성을 분석한다. 즉, 각 알고리즘에 대한 대수적 방정식의 차수 및 개수와 변수의 수를 조사함으로써 대수적으로 과포화된 상태 여부를 조사한다.

2.1 HELIX, SCREAM, MUGI, PANAMA의 알고리즘 특성

본 절에서는 네 개의 알고리즘에 대해 구조적 특징과 더불어 키 스트림의 데이터 크기, 키의 크기, 초기값의 크기 및 출력 크기 등을 비교한다. 이러한 요소들은 다음 절에 연립 방정식을 구성하는 데 핵심 역할을 하는 구성 요소이다. 다음의 표는 각 스트림 암호의 특성을 표로 구성한 것이다.

표 1. HELIX, SCREAM, MUGI, PANAMA의 특성 비교

	HELIX	SCREAM	MUGI	PANAMA
구 조	일반화된 Feistel 구조	Feistel 변형 구조	워드단위 LFSR 구조	워드단위 LFSR 구조
데이터 크기	32-비트	64-비트	64-비트	32-비트
키 크기	256-비트	128-비트	128-비트	256-비트
초기값 크기	128-비트	128-비트	128-비트	256-비트
출력 크기	32-비트	128-비트	64-비트	256-비트

Ⅲ. 대수적 연립 방정식 구성

본 절에서는 네 개의 각 알고리즘의 대수적 공격을 위한 연립 방정식을 구성하는 방법을 소개한다. 마스터 키를 통해 각 내부 변수를 초기화 하는 과정인 키 초기화 과정 등은 대수적 공격에 그다지 큰 영향을 주는 요소가 아니므로, 내부 변수를 통해 키 스트림을 생성하는 키 스트림 생성 함수(라운드 함수)를 중점 분석한다.

3.1 HELIX에 대한 대수적 연립 방정식 구성

HELIX[14]는 2003년 Counterpane Internet Security에서 개발한 비동기식 스트림 암호이다. HELIX는 256-비트의 마스터 키 길이를 가지며, 128-비트 Nonce를 사용한다. 라운드 함수 F를 통해 32-비트의 키 스트림을 한 라운드 당 생성한다. 또한, 비동기식의 특징을 이용하여 메시지 인증 코드(MAC)의 생성도 가능하다.

HELIX의 라운드 함수 F는 그림 1과 같다. 라운드 함수에서의 비선형의 연산은 법 2^{32}에서의 덧셈 연산이다. 이 연산에 대한 대수 방정식을 구성하

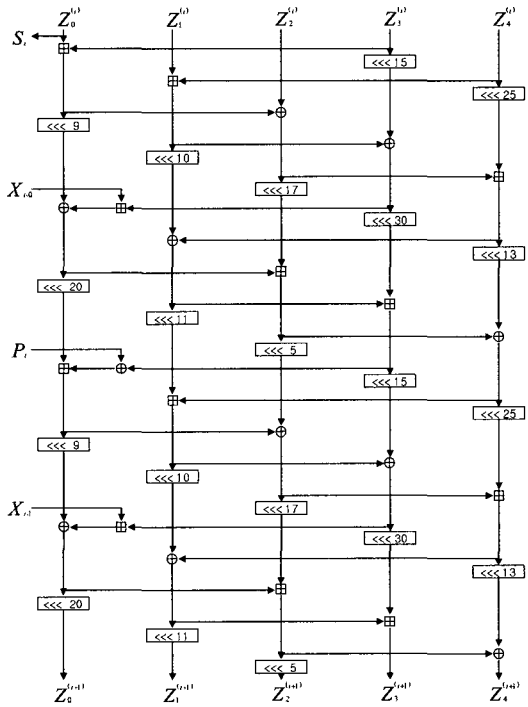


그림 1. HELIX의 라운드 함수 F

면, 31개의 2차 비선형 방정식과 1개의 일차 선형 방정식을 얻을 수 있다. 두 입력 값 A와 B에 대한 덧셈 연산의 값을 $C(=A+B)$라 하면 다음의 방정식을 얻을 수 있다. 여기서 $A=(a_{31},a_{30},\cdots,a_0)$, $B=(b_{31},b_{30},\cdots,b_0)$, $C=(c_{31},c_{30},\cdots,c_0)$이다.

$$
\begin{aligned}
c_0 &= a_0 \oplus b_0,\\
c_1 &= a_1 \oplus b_1 \oplus (a_0 \cdot b_0),\\
c_2 &= a_2 \oplus b_2 \oplus a_1 \oplus b_1 \oplus (a_1 b_1 \oplus a_1 c_1 \oplus b_1 c_1),\\
c_3 &= a_3 \oplus b_3 \oplus a_2 \oplus b_2 \oplus (a_2 b_2 \oplus a_2 c_2 \oplus b_2 c_2),\\
&\vdots\\
c_{30} &= a_{30} \oplus b_{30} \oplus a_{29} \oplus b_{29} \oplus (a_{29} b_{29} \oplus a_{29} c_{29} \oplus b_{29} c_{29}),\\
c_{31} &= a_{31} \oplus b_{31} \oplus a_{30} \oplus b_{30} \oplus (a_{30} b_{30} \oplus a_{30} c_{30} \oplus b_{30} c_{30}),
\end{aligned}
$$

라운드 함수에서 법 2^{32}에서의 덧셈을 제외한 나머지 부분은 선형이다. 또한 각 라운드에서 32-비트 상태 변수 a 중 키 스트림으로 출력되는 하나의 32-비트의 값을 얻을 수 있으므로, 덧셈에 관한 위의 대수적 방정식을 이용하여 방정식을 구성한다. 다음은 HELIX에 대한 대수적 방정식 및 변수 구성 방법을 요약한 것이다.

[HELIX의 연립 방정식 구성 방법]

- **변수** : 방정식에 대한 변수는 법 2^{32}에서의 덧셈에 사용된 두 입력 값 및 하나의 출력 값과 XOR 연산의 출력 값을 새로운 변수로 놓는다. Rotation 연산의 경우에는 새로운 변수로 할당하지 않는다.
- **선형 방정식** : 각각의 법 2^{32}에서의 덧셈의 마지막 비트에 대한 연산은 선형이므로 덧셈 연산에서는 한 개의 선형 방정식을 얻을 수 있고, XOR 연산에서는 32 개의 선형 방정식을 얻는다.
- **비선형 방정식** : 라운드 함수에서 유일하게 비선형 연산은 법 2^{32}에서의 덧셈이다. 이 연산에서 마지막 비트는 선형이므로 31 개의 비선형 방정식이 존재한다. 이 31 개의 비선형 방정식에는 96(=32×3)개의 일차 변수와 62 개의 이차 변수가 존재하므로 총 158 개의 변수가 생성된다.

HELIX의 라운드 함수 F에서 우선 5개의 $Z_j^{(i)}$에 변수에 대해 다음 라운드 $Z_j^{(i+1)}$에서 첫 번째 워드인 $Z_0^{(i+1)}$에 대한 값에 대한 식 만을 얻을 수 있다. 각 5개의 Z_i^j변수는 법 2^{32}에서의 덧셈(혹은 XOR 연산)과 rotation 연산을 통해 갱신된다. 이러한 과정을 1 단계라 한다면 총 20 단계로 구성된다. 이 중, $Z_0^{(i+1)}$과 연관성 있는 단계는 총 15 단계이다(15, 17, 18, 19, 20 단계 제외). 또한, 두 번(중간 5 step 후와 15 step 후)의 $X_{r,j}$과의 덧셈 과정이 추가된다. 첫 단계의 $Z_0^{(i)}$는 출력 스트림으로 얻을 수 있는 값이고 P_i는 알려진 값이라고 가정할 수 있으므로, 첫 단계의 연산과 P_i와의 XOR 연산은 상수와의 XOR 연산으로 생각할 수 있다.

한 라운드 과정에서 XOR 연산은 9번이고, 법 2^{32}에서의 덧셈 연산은 10번이다. 10번의 덧셈 연산 중 첫 번째 덧셈 연산은 상수와의 덧셈 연산이다. 상수와의 덧셈 연산은 두 번째 비트에 대한 연산도 선형이므로, 선형 방정식이 2개이고 비선형 방정식이 30개이다. 따라서, 총 선형 방정식은 299(=32×9+10×1+1)개이고, 비선형 방정식은 309(=31×10 - 1)개이다.

3.2 SCREAM에 대한 대수적 연립 방정식 구성

SCREAM[16]은 2002년 미국의 IBM에서 개발되었고, 마스터 키 길이가 128-비트인 동기식 스트림 암호이다. SCREAM은 키 스트림은 세 개의

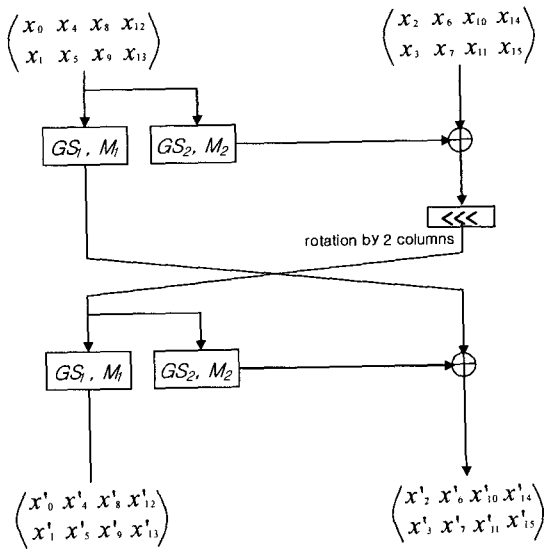


그림 2. SCREAM의 라운드 함수 F

128-비트 내부 상태 변수(x,y,z)를 한 번의 라운드 함수 계산 두 번의 XOR 연산 수행 후, 또 다른 128-비트 변수 W와 XOR 연산 수행 후 출력된다.

SCREAM의 라운드 함수 F는 128-비트의 입출력을 가지는 비선형 함수로, 블록 암호 AES의 라운드 함수의 구조를 이용하여 설계된 (GS_i, M_i)를 사용한다. 함수 (GS_i, M_i)는 64-비트의 입출력 길이를 가지며, 내부에는 8-비트의 입출력을 갖는 8개의 S-박스와 2×2 행렬로 구성되어 있다. 따라서 SCREAM에서의 유일한 비선형 부분은 AES의 8개의 S-박스이다.

이러한 SCREAM 알고리즘의 특성을 이용하면 우리는 다음과 같은 방법으로 SCREAM의 대수적 연립 방정식을 구성할 수 있다.

[SCREAM의 연립 방정식 구성 방법]

- **변수** : 세 개의 128-비트 내부 상태 변수 중 하나의 변수만 비선형 함수 F에 의해 갱신된다. 따라서 방정식에 대한 변수의 함수 F에서의 비선형 부분인 8 개의 S-박스 부분과 선형 연산인 XOR이 수행되는 각 부분에 새로운 변수를 할당한다.
- **선형 방정식** : 함수 F는 4 개의 행렬 연산으로 구성된 선형 변환과 2 번의 XOR 연산으로 구성되어 있다. 또한, 키 스트림 생성에는 128-비트 내부 변수 (x,y,z) 간의 두 번의 XOR 연산 및 변수 W와 한 번의 XOR 연산, 총 3 번의 XOR 연산이 존재한다. 이러한 연산은 비트별로 이루어지므로 이를 이용하여 각각의 선형 방정식을 생성한다.
- **비선형 방정식** : 스트림 생성 함수에서 유일하게 비선형 부분은 라운드 함수의 S-박스 부분이다. 두 개의 S-박스는 AES에 사용된 S-박스를 이용하여 생성된다. S_1-박스는 AES의 S-박스와 동일하며, S_2-박스는 S_1-박스에 선형 변환된 형태이다. 이 함수에 대한 대수적 방정식(총 24개의 이차 방정식)은 널리 알려져 있으므로 이를 이용하여 비선형 방정식을 구성한다.

SCREAM의 라운드 변환 과정은 우선 변수 128-비트의 변수 W와 (x,y,z)에 의해, 3 번의 128-비트 XOR 연산과 1 번의 라운드 함수 F 변환 과정이다. 최초 라운드의 변수의 수는 128-비트 변수 4 개, 각 XOR 연산 후의 변수 128-비트 변수 3 개, F 함수에서의 64-비트 변수 1 개이다. 따라서 총 1088(= 128×7 + 64×3) 개이다. 선형 방정식의 수는 128-비트 XOR 3 번과 64-비트의 XOR 1 번이므로 448 개이다. 비선형 방정식의 수는 라운드 함수 F에서 총 4 번의 G 함수가 있으므로 총 768(=4×8×24) 개이다.

SCREAM의 경우 최초 라운드 후 다음 라운드에서는 세 개의 128-비트의 변수 중 (x,y,z)가 계속 사용되고 변수 W만이 새로운 변수로 사용된다. 따라서 첫 라운드를 제외한 나머지 라운드에서는 변수의 수가 384개 감소한다. 선형 방정식의 수도 384 개 감소한다. 하지만 비선형 방정식의 수는 변화가 없다.

3.3 MUGI에 대한 대수적 연립 방정식 구성

MUGI[24]는 워드 단위의 LFSR 구조를 사용하는 PANAMA-계열의 알고리즘으로 CRYPTREC 프로젝트에 스트림 선정된 알고리즘이다. 128-비트의 마스터 키 길이와 초기 벡터 128-비트를 사용한다.

각 내부 변수 워드의 사이즈는 64-비트로, 16 개의 워드 단위의 LFSR 구조(선형)로 값이 갱신되는 b변수와 3개의 내부 변수 a가 존재한다. 여기서 함수 F는 유일한 비선형 부분으로 64-비트의 입출력을 갖는다. 함수 F는 키 역할로 입력되는 변수 b와 XOR 연산 후, 8-비트 입출력을 갖는 8 개의 S-박스를 통과한다. 이 후, 64-비트를 두 개의 32-비트로 분리하여 AES의 MDS 행렬 수행 후 출력된다.

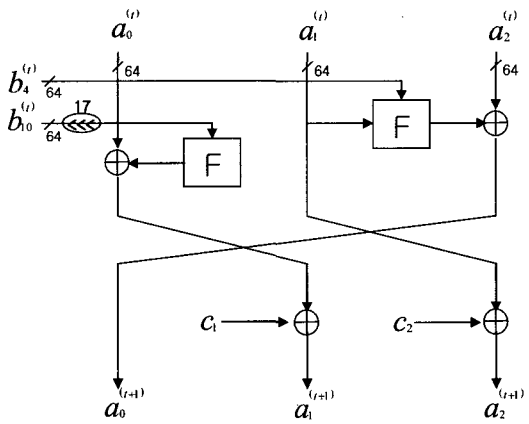


그림 3. MUGI의 라운드 함수 ψ

MUGI에 사용된 S-박스는 AES에 사용된 $GF(2^8)$ 위에서의 x^{-1}의 아핀 변환으로 유일한 비선형 부분이다. 이러한 MUGI 알고리즘의 특성을 이용하면 앞의 방법과 유사한 방법으로 다음과 같이 대수적 연립 방정식을 구성할 수 있다.

[MUGI의 연립 방정식 구성 방법]

- **변수** : 각 방정식에 대한 변수는 선형 연산인 XOR 연산에 대한 변수와 8-비트의 입출력을 갖는 S-박스 입출력에 대한 변수로 구성된다.
- **선형 방정식** : 라운드 함수 ψ는 두 개의 F 함수로 구성되어 있다. 함수 F는 하나의 선형 변환, 한 개의 버퍼의 값과 XOR 연산, S-박스 변환으로 이루어져 있다. 두 개의 F 함수 중 한 개의 F 함수만이 출력 키 스트림과 연관되어 있으므로 이것을 이용하여 64 개의 선형 방정식을 구성할 수 있다.
- **비선형 방정식** : 스트림 생성 함수에서 비선형인 부분은 SCREAM과 마찬가지로 S-박스 뿐이다. 8-비트의 입출력을 갖는 S-박스는 총 24 개의 이차 비선형 방정식이 존재한다.

MUGI의 최초 라운드의 변수의 수는 3 개의 64-비트의 변수 $a_j^{(t)}$와 2 개의 64-비트 F 함수 출력 변수로 총 320(=5×32)이다. 선형 방정식의 수는 2 번의 64-비트 XOR로 구성된 식 128 개이고, 비선형 방정식의 수는 두 번의 F 함수의 S-박스에서 계산된 192(=2×8×24) 개이다. 최초 라운드 후 64-비트 변수의 출력 값을 이용하면 64-비트 변수의 수를 줄일 수 있다.

3.4 PANAMA에 대한 대수적 연립 방정식 구성

PANAMA[24]는 MUGI와 마찬가지로 블록 단위의 LFSR 구조를 이용한 알고리즘으로 마스터 키 사이즈는 256-비트이다. LFSR의 상태변수는 총 32개의 256-비트 값이고 내부 상태 변수(a)는 17 개의 32-비트로 구성되어 있다.

PANAMA는 비선형 함수 ρ를 이용하여 내부 상태 변수 a를 갱신하여 8 개의 32-비트의 값은 키 스트림으로 출력되며 8 개의 32-비트 값은 LFSR 상태 변수를 갱신하는 데 사용된다.

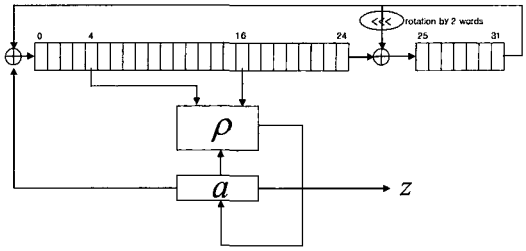


그림 4. PANAMA의 Pull Mode 함수

함수 ρ는 선형 부분인 σ, θ, π 함수와 비선형 부분인 γ 함수로 구성되어 있다. 비선형 함수 γ는 비트 단위의 이차 방정식으로 표현된다. 여기서, 변수 $A^i=(a_{31}^i, a_{30}^i, \cdots, a_0^i)$이고 $C^i=(c_{31}^i, c_{30}^i, \cdots, c_0^i)$이며 $0 \le i \le 16$이다.

$$c_j^i = a_j^{i+1}a_j^{i+2} \oplus a_j^i \oplus a_j^{i+2} \oplus 1 \quad (0 \le j \le 31)$$

위의 방정식과 알고리즘의 특성을 이용하여 다음과 같이 대수적 연립 방정식을 구성한다.

[PANAMA의 연립 방정식 구성 방법]

- **변수** : 비선형 함수 γ의 입출력을 새로운 변수로 할당한다.
- **선형 방정식** : 내부 상태 변수를 갱신하는 ρ 함수는 한 개의 선형 변환이 존재하는데, 이 변환은 256 개의 선형 방정식을 구성할 수 있다.
- **비선형 방정식** : 함수 γ에 의해 생성되는 이차 함수로 비선형 방정식을 구성한다.

PANAMA에서 γ에 대해 새로운 변수로 할당하면, 2개의 544 비트이므로 1312 개의 새로운 변수가 할당되고, 비선형 방정식의 수는 544(= 32×17)이다. 하지만, 초기화 과정 후 PANAMA의 버퍼의 비트 길이는 8192(=256×32)이고 내부 상태 변수 a의 길이는 544로, 총 8736 비트 길이이다.

Ⅳ. 각 알고리즘의 대수적 공격의 안전성 비교

표 2는 앞 절에서 구성한 네 개의 각 알고리즘(HELIX, SCREAM, PANAMA, MUGI)에 대한 대수적 연립 방정식의 변수와 방정식의 수를 분석한 것이다. 네 개의 모든 스트림 암호의 대수적 차수는 이차 이므로 일차 변수와 이차 변수만이 존

표 2. 한 라운드에 대한 변수 및 방정식의 수 비교

	HELIX	SCREAM	MUGI	PANAMA
변 수 (Variables)	736	1088	320	1312
선형 방정식 (Linear Eqs)	299	448	128	0
비선형 방정식 (Noninear Eqs)	309	768	192	544
전체 방정식 (Equations)	608	1216	320	544
일차 항 (Linear Terms)	736	1088	320	1312
이차 항 (Quadratic Terms)	849	2048	512	544
전체 항 (Terms)	1585	3136	832	1856
자유 항 (Free Terms)	977	1920	512	1312
변수 대 방정식의 비 (T/E Ratio)	2.61	2.58	2.6	3.41

재하고, 대수적 방정식도 선형 및 이차 방정식만이 존재한다.

HELIX는 모든 상태변수가 각 라운드 당 갱신되며, 이를 제외한 SCREAM, MUGI, PANAMA 알고리즘은 각 라운드 당 내부 상태 변수 중 일부분만 갱신된다. 따라서 HELIX의 경우에는 변수 대 방정식의 비(T/E)가 라운드가 증가하여도 변하지 않는다. 하지만, SCREAM, MUGI, PANAMA의 경우에는 모든 내부 변수의 값이 갱신되지는 않으므로 라운드의 수(i)가 증가함에 따라 변수, 방정식, 항의 수는 다음의 표 3과 같이 변동된다.

표 3. 라운드 증가에 따른 변수, 방정식, 항의 수 비교 (HELIX 제외)

	SCREAM	MUGI	PANAMA
변 수 (Variables)	1088 + 704×i	320 + 256×i	8736 + 288×i
선형 방정식 (Linear Eqs)	448×(i+1)	128×(i+1)	288×i
비선형 방정식 (Noninear Eqs)	768×(i+1)	192×(i+1)	544 + 256×i
일차 항 (Linear Terms)	1088 + 704×i	320 + 256×i	8736 + 288×i
이차 항 (Quadratic Terms)	2048×(i+1)	512×(i+1)	544 + 256×i

표 4. 라운드 수(i) 증가에 따른 변수 대 방정식의 비의 변화

i \ 알고리즘	HELIX	SCREAM	MUGI	PANAMA
10^1	2.61	2.29	2.42	2.46
10^5	2.61	2.26	2.40	1.00
10^9	2.61	2.26	2.40	1.00
10^{13}	2.61	2.26	2.40	1.00
10^{17}	2.61	2.26	2.40	1.00

SCREAM, MUGI, PANAMA 알고리즘은 변수, 방정식, 항의 수가 표 3과 같이 변화함에 따라 변수 대 방정식의 비가 감소할 수 있음을 예측할 수 있다. 다음의 표 4는 라운드 수에 따른 변수 대 방정식의 비의 변화를 나타낸 것이다. 여기서, HELIX의 경우는 그 비가 변동되지 않으며, SCREAM과 MUGI의 경우 그 비가 라운드 수가 약 10^5 정도에서 약간 감소하지만, PANAMA의 경우에는 10^5 정도의 라운드 수에서 그 비가 1이 된 후 유지됨을 알 수 있다.

변수 대 방정식의 수가 1보다 크면 대수적 공격에 안전함을 의미하고, 1보다 작으면 과포화 됨을 의미하므로 공격 가능함을 나타낸다. PANAMA의 경우에는 그 비가 1에 근사하는데(엄밀히 말하면 1보다 약간 큼), 이 경우도 대수적 공격을 직접 적용하여 해를 찾을 수는 없지만, 다른 알고리즘보다 대수적 공격에 대한 내성이 적음을 알 수 있다.

나머지 HELIX, SCREAM, MUGI의 경우에는 그 비가 1보다 크므로 대수적 공격에 내성을 지님을 알 수 있다. 보다 엄밀히 말하면, 변수 대 방정식의 비가 크면 클수록 안전성이 높음을 나타낸다.

V. 대수적 공격에 안전한 소프트웨어 구현에 적합한 스트림 암호의 설계 고려 사항

우리는 지금까지 네 개의 소프트웨어 구현에 적합한 스트림 암호의 대수적 공격의 안전성 분석하였다. 그 결과, 대수적 공격에 안전하기 위해서는 변수 대 방정식의 비가 높아야 하고, 이를 달성하기 위해서는 내부 변수는 높은 비선형 함수를 이용하여 적절히 갱신되어야 함을 얻었다. 또한, 내부 변수의 갱신 비율이 높으면 높을수록 라운드 수가 증가함에 따라 변수 대 방정식의 수가 감소하지 않음을 얻었다.

다음은 이러한 결과를 바탕으로 대수적 공격에 안

전한 소프트웨어 구현에 적합한 스트림 암호의 설계 시 고려해야 할 세 항목으로 나누어 제시한다.

1. 키 스트림의 출력 길이

- 한 라운드 당 SCREAM과 PANAMA의 출력의 길이가 HELIX와 MUGI보다 비교적 크다. 이에 따라, 변수 대 방정식의 비도 출력의 길이가 길면 길수록 감소함을 알 수 있다. 따라서 라운드 당 출력 길이가 작을수록 대수적 공격에 보다 안전할 수 있으므로, 라운드 당 출력 길이의 사이즈를 신중히 고려해야 한다. 보다 구체적으로, 출력 비트의 길이는 내부 상태 변수의 길이의 반 보다 작게 사용하는 것이 안전하다. 만약, 출력 길이가 내부 변수의 길이의 반 이상을 사용하면, 출력 길이가 많아지면 많아질수록 방정식 대 변수의 비율이 점점 줄어들어 과포화 상태 혹은 비율 1의 값에 수렴할 수 있어 대수적 공격에 대한 내성이 작아질 수 있다.

2. 비선형 방정식의 형태

- 대수적 차수가 높으면 높을수록 대수적 공격에 안전하다는 것은 자명한 사실이다. 그러면 다른 조건이 같고 과연 대수적 차수가 같다면 같은 내성을 가질 것인가에 대한 의문을 가질 수 있다. 본 논문에서 살펴본 네 알고리즘은 모두 이차의 대수적 차수를 갖는다. 하지만 PANAMA의 경우는 다른 세 알고리즘에 비해 변수 대 방정식의 비율이 낮다. 물론 구조적인 특성에 의해 기인한 것도 있겠지만 비선형 방정식의 형태가 다른 점이 크게 작용한 것이다. PANAMA의 비선형 방정식은 양함수(Explicit Function)의 형태이나, 다른 세 알고리즘은 음함수(Implicit Function) 형태의 방정식이다. 음함수형태의 방정식은 높은 차수의 변수를 직접적으로 사용하지 않기 위해서는 추가적인 변수가 필요한 반면, 양함수의 경우에는 이러한 추가적 변수가 필요하지 않다. 따라서 대수적 차수가 같더라도 음함수 형태의 방정식이 사용되었다면 양함수에 비해 보다 많은 내부 변수를 갖게 되어 대수적 공격에 내성을 가지게 되므로, 비선형 함수도 신중히 고려하여 설계되어야 한다. 물론, 음함수의 사용이 양함수의 상용보다 절대적으로 안전하다고는 할 수 없다. 이는 단지 같은 대수적 차수를 갖는다면 양함수보다는 음함수가 대수적 공격에 보다 내성을 지님을 의미하는 것이다. 만약 양함수 형태의 함수를 사용한다면, 대수적 차수를 높이거나 다른 암호 논리를 통해 이러한 약점을 보완하여 안전성을 높일 수 있다.

3. 내부 변수의 위치 및 갱신 과정

- HELIX의 경우 내부 변수 모두 비슷한 연산 과정을 거쳐 갱신되지만, SCREAM, PANAMA, MUGI의 경우에는 내부 변수가 모두 갱신되지는 않는다. 또한, 구조에 따라 차이는 있겠지만 내부 변수들이 갱신되어 변하는 위치 등도 대수적 공격의 안전성에 영향을 줌을 알 수 있다. 따라서 알고리즘의 설계 시 내부 변수의 위치 및 갱신 과정을 신중히 선택해야 한다. 구체적으로 설명하자면, 내부 변수를 이용하여 출력 스트림을 생성할 때, 내부 변수의 값을 그대로 출력 할 경우 내부 변수의 값이 그대로 노출되어 대수적 공격에 대한 내성이 작아질 수 있다. 또한 내부 변수를 통해 출력 스트림 생성 시 내부 변수의 위치가 주기적인 간격만을 이용하여 생성하거나 전체 내부 변수 값을 이용하지 않고 특정 부분의 변수만을 이용할 경우, 내부 변수의 주기적 성질을 이용한 공격 등에 취약할 수 있다. 따라서 알고리즘의 설계 시 내부 변수의 위치 및 갱신 과정을 신중히 선택해야 한다.

물론 본 논문에서 제시하는 스트림 암호의 설계 기준은 안전성에 기준을 맞추어 제시한 것으로, 스트림 암호의 효율성 측면에서는 trade-off 관계가 있음을 명심해야 한다. 특히, 블록 단위의 스트림 암호에서는 안전성과 더불어 효율성 역시 알고리즘 설계 요소이므로 앞서 제시한 설계 고려사항을 잘 활용하여 안전성을 평가한 후, 안전성을 크게 저해하지 않는 범위에서 효율성을 극대화할 수 있도록 설계하는 것이 보다 바람직한 방향이다.

Ⅵ. 결 론

본 논문에서는 현재까지 개발된 소프트웨어 구현

에 적합한 스트림 암호의 설계 특성을 살펴보고 설계 논리를 바탕으로 알고리즘 들을 분류한 후, 아직까지 구조적인 취약점이 발견되지 않은 대표적 스트림 암호인 HELIX, SCREAM, PANAMA, MUGI 알고리즘에 대해 대수적 공격의 안전성을 살펴보았다. 또한, 이러한 안전성 분석을 통해 세 가지의 대수적 공격에 안전한 스트림 암호의 설계 시 고려해야 할 항목을 도출하였다. 이러한 결과는 안전한 소프트웨어 구현에 적합한 스트림 설계에 효과적으로 활용될 수 있을 것이다.

참 고 문 헌

[1] F. Armknecht and M. Krause, "Algebraic Attacks on Combiners with Memory," *Advances in Cryptology - CRYPTO 2003*, LNCS 2729, Springer-Verlag, pp. 162-175, 2003.

[2] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, "Rabbit : A New High-Performance Stream Cipher," *The 10th Fast Software Encryption Workshop - FSE 2003*, LNCS 2887, Springer-Verlag, pp. 307-327, 2003.

[3] N. Courtois, "The Security of Hidden Field Equations," *Topics in Cryptology - CT-RSA 2001*, LNCS 2020, Springer-Verlag, pp. 266-281, 2001.

[4] N. Courtois, "Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt," *The 5th International Conference on Information Security and Cryptography - ICISC 2002*, LNCS 2587, Springer-Verlag, pp. 182-199, 2002.

[5] N. Courtois, "Fast Algebraic Attack on Stream Ciphers with Linear Feedback," *Advances in Cryptology - CRYPTO 2003*, LNCS 2729, Springer-Verlag, pp. 176-194, 2003.

[6] N. Courtois, W. Meier, "Algebraic Attacks on Stream Ciphers with Linear Feedback," *Advances in Cryptology - EUROCRYPT 2003*, LNCS 2656, Springer-Verlag, pp. 345-359, 2003.

[7] N. Courtois and J. Patarin, "About the XL Algorithm over GF(2)," *Topics in Cryptology - CT-RSA 2003*, LNCS 2612, Springer-Verlag, pp. 141-157, 2003.

[8] N. Courtois and J. Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined System of Equations," *Advances in Cryptology - ASIACRYPT 2002*, LNCS 2501, Springer-Verlag, pp. 267-287, 2002.

[9] CRYPTREC Project, *Cryptography Research Evaluation Committee*, 2000-2002. Available at http://www.ipa.go.jp/security/enc/CRYPTREC.

[10] J. Daemen and C. Clapp, "Fast Hashing and Stream Cipher with PANAMA" *The 5th Fast Software Encryption Workshop - FSE 1998*, LNCS 1372, Springer-Verlag, pp. 60-74, 1998.

[11] C. Ding, V. Niemi, A. Renvall, and A. Salomaa, "TWOPRIME : A Fast Stream Ciphering Algorithm" *The 4th Fast Software Encryption Workshop - FSE 1997*, LNCS 1372, Springer-Verlag, pp. 82-102, 1997.

[12] K. Discoll, "BEEPBEEP : Embedded Real-Time Encryption" *The 9th Fast Software Encryption Workshop - FSE 2002*, LNCS 2365, Springer-Verlag, pp. 164-178, 2002.

[13] P. Ekdahl and T. Johansson, "A New Version of the Stream Cipher SNOW" *The 9th Annual International Workshop - SAC 2002*, LNCS 2595, Springer-Verlag, pp. 47-61, 2002.

[14] N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks, and T. Kohno, "Helix : Fast Encryption and Authentication in as Single Cryptographic Primitive," *The 10th Fast Software Encryption Workshop - FSE 2003*, LNCS 2887, Springer-Verlag,

pp. 330-346, 2003.

[15] S. Furuya, D. Watanabe, and K. Takaragi, "MULTI-S01 : An Integrity-Aware Block Encryption Based on Cryptographic Pseudorandom Number Generator," *Submitted at CRYPTREC Project*, 2000.

[16] S. Halevi, D. Coppersmith, and C. Jutla, "Scream : A Software-Efficinet Stream Cipher," *The 9th Fast Software Encryption Workshop - FSE 2002*, LNCS 2365, Springer-Verlag, pp. 195-209, 2002.

[17] P. Hawkes and G. G. Rose, "Primitive Specification and Supporting Documentation for SOBER-t32 Submission to NESSIE," *Proceedings of the First Open NESSIE Workshop*, 2000.

[18] A. Kipnis and A. Shamir, "Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization," *Advances in Cryptology - CRYPTO 1999*, LNCS 1666, Springer-Verlag, pp. 19-30, 1999.

[19] NESSIE Project, *New European Schemes for Signatures, Integrity and Encryption*, 2000-2002. Available at http://cryptonessie.org.

[20] R. Rivest, "RC4," *Unpublished Work* (A description of RC4 appears in B. Schneier, Applied Cryptography, 1996).

[21] P. Rogaway and D. Coppersmith, "A Software-Optimized Encryption Algorithm," *The 1st Fast Software Encryption Workshop - FSE 1993*, LNCS 809, Springer-Verlag, pp. 56-63, 1993.

[22] P. Rogaway and D. Coppersmith, "A Software-Optimized Encryption Algorithm," *Journal of Cryptology*, Vol. 11, No. 4, pp. 274-287, 1998.

[23] G. G. Rose and P. Hawkes, "Turing : A Fast Stream Cipher," *The 10th Fast Software Encryption Workshop - FSE 2003*, LNCS 2887, Springer-Verlag, pp. 290-306, 2003.

[24] D. Watanabe, S. Furuya, H. Yoshida, K. Takaragi, and B. Preneel, "A New Keystream Generatior MUGI," *The 9th Fast Software Encryption Workshop - FSE 2002*, LNCS 2365, Springer-Verlag, pp. 179-194, 2002.

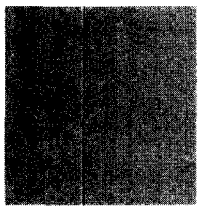
[25] H. Wu, "A New Stream Cipher HC-256," *The 11th Fast Software Encryption Workshop - FSE 2004*, LNCS 3017, Springer-Verlag, pp. 226-244, 2004.

[26] M. Zhang, C. Carroll, and A. Chan, "The Software-Oriented Stream Cipher SSC2," *The 7th Fast Software Encryption Workshop - FSE 2000*, LNCS 1978, Springer-Verlag, pp. 31-48, 2004.

[27] B. Zoltak, "VMPC One-Way Function and Stream Cipher," *The 11th Fast Software Encryption Workshop - FSE 2004*, LNCS 3017, Springer-Verlag, pp. 210-225, 2004.

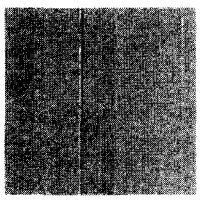
〈著者紹介〉

성 재 철 (Jaechul Sung)
1997년 8월 : 고려대학교 수학과 학사
1999년 8월 : 고려대학교 수학과 석사
2002년 8월 : 고려대학교 수학과 박사
2002년 7월~2004년 1월 : 한국정보보호진흥원 선임연구원
2004년 2월~현재 : 서울시립대학교 수학과 전임강사
〈관심분야〉 대칭키 암호, 해쉬 함수, 메시지 인증 코드

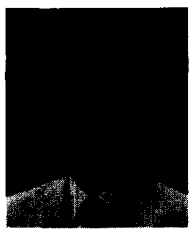


문 덕 재 (Dukjae Moon)
2000년 2월 : 서울시립대학교 수학과 학사
2003년 2월 : 고려대학교 정보보호대학원 석사
2003년 2월~현재 : 국가보안기술연구소 연구원

임 흥 수 (Hung-su Im)
2003년 2월 : 고려대학교 수학과 학사
2003년 3월~현재 : 고려대학교 정보보호대학원 석사과정
〈관심분야〉 블록 암호 및 스트림 암호의 설계 및 분석



지 성 택 (Seongtaek Chee)
1985년 2월 : 서강대학교 수학과 학사
1987년 2월 : 서강대학교 수학과 석사
1999년 2월 : 고려대학교 수학과 박사
1989년 10월~현재 : 국가보안기술연구소 책임연구원



이 상 진 (Sangjin Lee)
1987년 2월 : 고려대학교 수학과 학사
1989년 2월 : 고려대학교 수학과 석사
1994년 2월 : 고려대학교 수학과 박사
1989년 2월~1999년 2월 : 한국전자통신연구원 연구원, 선임연구원
1999년 2월~2001년 8월 : 고려대학교 자연과학대학 조교수
2001년 9월~현재 : 고려대학교 정보보호대학원 부교수
〈관심분야〉 대칭키 암호, 정보은닉이론, 컴퓨터 포렌식