정수계획법에 기반한 공개키 암호 알고리즘의 설계
(Design of Public-Key Cryptographic Algorithm based on Integer Programming)

용 승 림 † 조 태 남 † 이 상 호 ††
(Seung-Lim Yong)(Tae-Nam Cho)(Sang-Ho Lee)

요 약 공개키 암호 알고리즘은 암호화에 사용되는 공개키와 복호화에 사용되는 비밀키가 서로 다르며, 공개키는 공개되고 비밀키는 비밀로 유지되어 소유자만이 알고 있다. 이러한 알고리즘의 암호화 함수는 한 방향으로의 계산은 매우 쉬우나, 특별한 정보 없이 반대 방향으로 계산하는 것은 매우 어려운 성질이 있도록 하기 위하여 계산상 풀기 어려운 문제에 기반하여 연구되고 있다.

본 논문에서는 정수계획법에 기반한 계산상 풀기 어려운 문제를 이용하여 새로운 공개키 암호 알고리즘을 제안한다. 먼저 정수 계획법에 대하여 소개하고 비밀키와 공개키의 생성 과정을 보인다. 공개키로 이용되는 행렬을 평문에 곱하여 암호문을 만들고 공개키와 비밀키를 이용한 복호화 행렬을 이용하여 평문을 복원한다. 이 알고리즘의 키 생성 방식은 기존의 배낭꾸리기 암호 시스템의 방식과 유사하지만 배낭꾸리기 시스템의 비밀키에서 나타나는 취약점을 보완하였다.

Abstract The public-key cryptosystem uses two different keys for encryption and decryption. Since the encryption function of a public-key cryptosystem has to have properties of trapdoor and one-wayness, it is based on computationally intractable problems.

We propose a new public-key cryptographic algorithm based on integer programming. We introduce the integer programming problem, and then show the scheme of encryption and decryption using integer programming problem. Our algorithm efficiently encrypts and decrypts in polynomial time. Although the proposed algorithm is similar to knapsack public-key cryptosystem, it is more secure on attacks because it complements the weakness of private key of the knapsack cryptosystem.

1. 서 론

암호 기술이란 정당한 사용자는 정보를 알 수 있지만 정당하지 못한 사용자는 정보를 알 수 없도록 비밀성을 보장하기 위하여 암호 알고리즘을 이용하여 평문을 암호문으로 바꾸고 정당한 사용자만이 암호문을 평문으로 복호화할 수 있도록 하는 방법이다. 이러한 암호 기술은 평문을 암호화할 때 사용하는 키의 분배나 사용 방법에 따라 비밀키 암호 시스템과 공개키 암호 시스템으로 분류한다.

비밀키 암호 시스템은 안전하게 통신을 하고자 하는 두 사람이 서로 같은 키 혹은 서로 쉽게 유도 가능한 키를 공유하고, 이를 이용하여 평문을 암호화하며 암호화된 문서를 복호화한다. 일반적으로 비밀키 암호 시스템은 주어진 메시지를 단순한 비트 연산의 반복을 통하여 암호화하기 때문에 비교적 암호화와 복호화 속도가 빠르다. 그러나 암호화할 때와 복호화할 때 같은 키를 사용하기 때문에 암호화 과정에 사용되는 키를 안전하게 분배해야 하고, 여러 사람과 통신을 하기 위해서 통신하고자 하는 사용자 쌍마다 각각의 키를 가지고 있어야 하는 문제점이 있다. 이러한 문제를 해결하기 위하여 1976년 Diffie Hellman이 새로운 암호 방식인 공개키 암호 시스템을 제안하였다[1].

공개키 암호 시스템은 암호화에 사용하는 공개키와 복호화에 사용하는 비밀키를 다르게 만들어 공개키는

· 이 논문은 과학기술부 여자대학교 연구기반 확충사업(KISTEP)의 연구비 지원에 의한 연구 결과임.
† 학생회원 : 이화여자대학교 컴퓨터학과
dragon@ewha.ac.kr
tncho@ewha.ac.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
shlee@ewha.ac.kr
논문접수 : 2000년 2월 1일
심사완료 : 2000년 7월 25일

공개하고 비밀키는 안전하게 유지한다. 따라서 비밀키 암호 시스템처럼 비밀키를 분배하거나 통신자 각각의 키를 모두 가지고 있을 필요가 없다. 또한 비밀키 암호 시스템에서는 송신자의 신원을 확인할 수 없으나, 공개키 암호 시스템은 송신자의 신원을 확인할 수 있는 전자서명에도 이용할 수 있기 때문에 문서를 비밀로 전송하는 동시에 송신자의 신원도 확인할 수 있도록 해 준다. 공개키 암호 시스템은 사용되는 수학적 계산의 복잡도가 비밀키 암호 시스템에 비해 높기 때문에 암호화의 속도는 느리지만 안전성이나 응용성이 좋다.

대부분의 공개키 암호 시스템은 수학분야의 풀기 어려운 문제나 계산 복잡도 이론상으로 풀기 어려운 문제를 기반으로 한다. 수학분야의 풀기 어려운 문제를 기반으로 한 시스템들은 암호화와 복호화의 속도가 느리다는 단점이 있지만 아직까지 효율적인 공격법이 알려져 있지 않아 현재까지 실용화되어 이용되고 있다. 계산 복잡도 이론상으로 풀기 어려운 문제인 NP-완전 문제 부류의 문제를 이용한 공개키 암호 알고리즘은 암호문의 길이가 평문의 길이보다 훨씬 더 커지거나, 비밀키 설정상의 특성으로 인한 취약성에 대하여 많은 암호학적 공격이 알려져 있다.

아직까지 계산상 풀기 어려운 많은 문제들이 공개키 암호 시스템에 이용될 수 있는 가능성을 가지고 있다. 본 논문에서는 정수 계획법이라는 문제의 특수한 형태를 기반으로 한 새로운 공개키 암호 알고리즘을 제안한다. 이 알고리즘은 계산상 풀기 어려운 부정방정식의 해를 구하는 문제를 이용한 것으로서, 기존의 배낭꾸리기(knapsack) 암호 시스템의 방식과 유사하지만 비밀키의 특성으로 인한 취약점을 보완하였기 때문에, 이러한 취약점을 이용한 암호학적 공격 방법에 대해 안전하다. 또한 메시지를 암호화하고 복호화하는 시간이 짧아 효율적이고 시스템 구현이 간단하다는 장점이 있다.

2. 관련연구

공개키 암호 시스템은 공개키를 알고 있다 하더라도 유용한 시간 내에 수학적인 계산이나 유추에 의해서 공개키로부터 비밀키를 알아내는 것이 불가능해야 한다. 이러한 성질을 만족하기 위해서 암호화 함수는 한 방향으로의 계산은 매우 쉬우나 반대 방향으로의 계산은 매우 어려운 일방향의 성질을 만족해야 한다. 또한 특별한 정보를 가지고 있는 경우에는 반대 방향으로의 계산을 쉽게 할 수 있는 트랩도어(trapdoor)의 성질도 있어야 한다. 지금까지 제안되어 온 공개키 암호 시스템은 특별한 정보 없이 암호문을 복호화하는 것이 비효율적이도록 하기 위해서, 풀기 어렵다고 생각되는 수학 문제나 NP-완전 문제를 이용하여 개발되었다.

1978년 Ron Rivest, Adi Shamir 그리고 Leonard Adleman은 RSA라는 공개키 암호 시스템을 제안하였다[8]. RSA 공개키 암호 시스템은 합성수를 소인수 분해하는 것이 어렵다는 점에 기반하여 개발되었다. 즉, 두 개의 큰 소수 p, q를 알고 있을 때 $n=p\cdot q$인 합성수 n을 계산하기는 쉽지만, n을 소인수 분해하여 소인수 p, q를 알아내기는 어렵다는 사실을 이용하여 만들어진 시스템이다. 수학 문제의 어려움에 기반한 또 하나의 대표적인 시스템으로 ElGamal의 공개키 암호 시스템이 있다[2]. 이 암호 시스템은 이산대수 문제를 근간으로 만들어졌다. 이산대수 문제란 $y \equiv g^x \bmod p$인 g, p와 y를 알고 있을 때 x의 값을 알아내는 문제인데 ElGamal의 공개키 암호 시스템은 g, p와 y의 값을 알더라도 x를 구하기 어렵다는 사실을 근거로 만들어진 시스템이다. 이 외에 합성수의 제곱근을 구하는 문제에 기반한 Rabin의 알고리즘, 타원곡선 상의 이산대수 문제를 이용한 타원곡선 공개키 암호 알고리즘 등이 있다[3,7].

NP-완전문제를 기반으로 한 공개키 암호 시스템은 Merkle-Hellman의 배낭꾸리기(knapsack) 공개키 암호 시스템과 McEliece의 에러 수정 코드(error correcting code) 시스템 등이 있다[5, 6]. 배낭꾸리기 암호 시스템은 NP-완전문제를 기반으로 한 최초의 공개키 암호 시스템으로서 계산상 풀기 어려운 문제로 알려진 부분합 문제를 기반으로 고안되었다. 배낭꾸리기 문제란 서로 다른 값을 갖는 원소로 이루어진 집합 $M=\{M_1, M_2, \ldots, M_n\}$과 어떤 특정한 값 S가 주어졌을 때, 합계가 S가 되는 집합 M의 부분집합이 존재하는지를 결정하는 문제이다. Merkle-Hellman은 일반 배낭꾸리기 문제가 풀기 어려운 성질을 이용하여 일방향성을 만족하였고, 특별한 정보를 이용하여 초증가 수열(superincreasing sequence)의 부분합 문제로 변형하여 쉽게 답을 구함으로써 트랩도어 성질도 만족시켰다. 초증가 수열이란 수열의 각 항이 모든 이전 항들의 합계보다 더 큰 수열을 말한다. 즉, 수열 $B=(b_1, b_2, \ldots, b_n)$의 각 원소가 $b_i > \sum_{j=1}^{i-1} b_j$ 를 만족하는 양의 정수이면 B를 초증가 수열이라 한다. 배낭꾸리기 공개키 암호 알고리즘에서는 이러한 초증가 수열을 비밀키로 하고, 초증가 수열을 변형하여 생성한 일반적인 수열은 공개키로 한다. 공개키는 $N > \sum_{i=1}^{n} b_i$를 만족하는 N과 $\gcd(w, N)=1$을 만족하는 w $(1<w<N)$를 선택한 후,

$a_i \equiv w \cdot b_i \bmod N$ 의 계산 과정을 거쳐 수열 $A=(a_1, \cdots, a_n)$로 변형하여 생성한다. 평문을 암호화하기 위해서는 먼저 평문을 이진화하여 벡터 $X=(x_{1, \cdots, x_n})$로 만든다. 평문 X에 대한 암호문 S는 평문과 공개키를 이용하여 $S=\sum A \cdot X$를 계산한 부분합이 된다. 암호문을 복호화하기 위해서는 $w \cdot w^{-1} \equiv 1 \bmod N$이 되는 w^{-1}을 구하고 $w^{-1} \cdot S \bmod N=\sum B \cdot X$가 되는 벡터 $X=(x_1, \cdots, x_n)$를 계산해 낸다.

이 암호 시스템은 초증가 수열을 변형하여 일반적인 수열로 만들었는데 초증가 수열의 2^{i} 항은 모든 이전 항들의 합보다 커야 하며 공개키인 일반 수열의 각 항이 비밀키인 초증가 수열의 각 항에 일대일로 대응하는 특성을 가지게 된다. 이러한 특성이 약점이 되어 이에 대한 여러 가지 공격법이 제안되었고 제안된 공격법들에 의하여 안전하지 못함이 증명되었다[4,10,11]. 본 논문에서 제안한 알고리즘은 배낭꾸리기 공개키 암호 시스템과 유사성을 가지고 있지만 전술한 바와 같은 배낭꾸리기 암호 시스템의 약점을 보완하였다.

3. 알고리즘

본 장에서는 정수 계획법에 대하여 소개하고 본 논문에서 제안하고자 하는 정수 계획법에 기반한 공개키 암호 알고리즘을 기술한다.

정수 계획법이란 양의 정수 m, n과 정수의 계수를 갖는 $m \times n$ 크기의 행렬 A, 정수 계수를 갖는 벡터 B, C, 그리고 정수 d가 주어졌을 때 $C \cdot X \geq d$이면서 $A \cdot X \leq B$를 만족하는 정수 계수의 벡터 X가 존재하는지의 여부를 결정하는 문제이다. 또한 $C \cdot X \geq d$이면서 등식 $A \cdot X = B$을 만족하는 정수 계수의 벡터 X가 존재하는지의 여부를 결정하는 문제 역시 정수 계획법이라 한다[9].

본 논문에서 제안한 공개키 암호 알고리즘은 정수 계수의 벡터 C의 모든 원소를 1로, 정수 d를 0으로 고정시킨 특수한 형태의 정수계획법을 이용하였다. 이 알고리즘은 공개키 암호 알고리즘이 만족해야 하는 일방향성과 트랩도어의 성질을 만족한다. 정수 계획법 정의에서의 행렬 A를 공개키로 이용하고, 암호화하고자 하는 평문을 X로, 두 행렬의 곱으로 얻어진 벡터 B를 암호문으로 이용한다. 행렬 A와 벡터 X를 알고 있을 때 $A \cdot X = B$를 계산하는 것은 쉽지만, 행렬 A와 B를 알고 있을 때 벡터 X를 구하는 문제는 부정방정식의 해를 구하는 문제가 되므로 일방향성을 만족한다. 특별한 정보를 알고 있는 사람은 쉽게 해를 구할 수 없는 부정방정식의 문제를 알고 있는 정보를 이용하여 일반적인 행렬의 곱셈문제로 변형하여 쉽게 해를 구할 수 있다. 따라서 트랩도어의 성질도 만족한다. 정수 계획법을 이용한 공개키 암호 알고리즘은 다음과 같은 단계를 가지고 있다.

키 생성 단계에서는 행렬 크기 등의 파라미터 값을 결정하고 결정된 파라미터 값을 이용하여 공개키와 비밀키를 생성한다. 암호화 단계에서는 평문을 일정한 크기의 메시지 블록으로 나누고 키 생성 단계에서 생성된 공개키를 이용하여 암호화한다. 복호화 단계에서는 키 생성 단계에서 생성된 비밀키와 암호화 단계에서 생성된 암호문을 이용하여 평문을 복호화한다. 각 단계별 상세 알고리즘은 다음 절에서 기술한다.

3.1 키 생성

키 생성 단계에서는 공개키와 비밀키를 생성한다. 먼저, 공개키와 비밀키의 생성에 필요한 키의 크기 파라미터와 메시지 블록의 길이 파라미터를 설정한다. 설정한 키의 크기 파라미터를 이용하여 비밀키를 생성하고, 비밀키와 메시지 블록의 길이 파라미터를 이용하여 공개키를 생성한다.

3.1.1 파라미터 설정 단계

먼저, 공개키와 비밀키의 크기 파라미터를 정한다. 공개키는 $m \times n$ 크기의 행렬이고, 비밀키는 $(n-m) \times n$ 크기의 행렬과 몇 개의 변수로 이루어져 있다. 공개키 행렬의 행수가 비밀키 행렬의 행수보다 많아야 하기 때문에 $n-m \leq m < n$을 만족하도록 양의 정수 m과 n을 설정한다.

메시지 블록은 암호화를 위하여 적당한 길이로 자른 평문의 일부로서 하나의 블록은 $k \times n$ 비트가 된다. 파라미터 k는 메시지 블록의 크기를 결정해 주는 요소이며, n은 비밀키 및 공개키 행렬의 크기 파라미터와 동일한 값을 사용한다. 이 단계에서 설정된 파라미터들은 비밀키를 생성하고 공개키를 변형하는데 이용된다.

3.1.2 비밀키 생성 단계

비밀키는 행렬 S, 변수 w, N, 교환순열(permutation) π로 이루어진다. 먼저, 0보다 큰 정수를 임의로 선택하여 $(n-m) \times n$ 크기의 비밀키 행렬 S를 만든다. 비밀키 행렬 S는 평문을 복호화할 때 이용되며 다음과 같이 표현된다.

$$S=\begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ & \vdots & & \\ s_{n-m,1} & s_{n-m,2} & \cdots & s_{n-m,n} \end{pmatrix} \text{ s.t. } s_{i,j} \in_R \mathbb{N} \ (\in_R : \text{무작위 선택})$$

w와 N은 비밀키 행렬 S와 파라미터 k를 이용하여

다음과 같은 조건을 만족하도록 설정한다. w와 N은 비밀키를 변형하여 공개키를 만들 때와 암호문을 평문으로 복호화할 때 이용된다.

$$N : N > \max\left\{ \sum_{j=1}^{n} s_{ij} \cdot (2^k-1) \mid i=1,2,\ldots,n-m \right\}$$

w : $1 < w < N$ 이고 $\gcd(w,N)=1$ 인 정수

마지막으로, 공개키 생성시 행의 교환에 사용될 교환순열 π를 설정한다. 교환순열 π는 집합 $I=\{1,2,\ldots,l\}$에서 그 자체의 집합으로 일대일 대응되는 순열이다. 이와 같이 생성한 (S,w,N,π)가 비밀키가 된다.

3.1.3 공개키 생성 단계

공개키는 $m\times n$ 크기의 공개키 행렬 P로 이루어진다. 공개키 행렬은 비밀키 행렬 S를 변형하여 $m\times n$ 행렬 T를 만들고, 교환순열을 적용하여 생성한다. T의 처음 $n-m$ 행의 각 원소들은 S의 대응되는 원소값에 w를 곱하고 $\bmod N$ 연산을 하여 구한다. 나머지 $2m-n$ 행은 N보다 작은 임의의 양의 정수로 선택한다. 이 때, $2m-n$ 행의 원소들은 3.3.1절에서의 행렬 B가 역행렬이 존재하도록 선택한다. 행렬 T의 생성 과정은 다음과 같다.

$T=(t_{ij})$

$$t_{ij}=\begin{cases} w\cdot s_{ij} \bmod N, & \text{if } 1\le i\le n-m,\ 1\le j\le n \\ r_{ij}\in_R \mathbf{N} \bmod N, & \text{if } n-m<i\le m,\ 1\le j\le n \end{cases}$$

위에서 생성한 행렬 T의 각 행에 교환순열 π를 적용하여 공개키 행렬 P를 만들어 낸다. 공개키 행렬 P는 다음과 같이 표현된다.

$$P=\begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ & \vdots & & \\ p_{m,1} & p_{m,2} & \cdots & p_{m,n} \end{pmatrix}=\begin{pmatrix} t_{\pi(1),1} & t_{\pi(1),2} & \cdots & t_{\pi(1),n} \\ & \vdots & & \\ t_{\pi(m),1} & t_{\pi(m),2} & \cdots & t_{\pi(m),n} \end{pmatrix}$$

3.2 암호화

대부분의 암호 알고리즘에서와 마찬가지로 메시지를 암호화하기 위해서는 암호화할 평문을 일정한 길이의 블록으로 분할하고, 분할된 각 메시지 블록에 대해 암호화 알고리즘을 적용하여 각 메시지 블록에 대응하는 암호문 블록을 만든다.

3.2.1 평문 분할 단계

평문 M은 파라미터 설정 단계에서 결정된 파라미터 k과 n을 이용하여 $k\times n$ 비트의 블록으로 분할한다. 먼저, 암호화하고자 하는 평문을 이진화한다. 만일 이진화된 평문 M의 길이가 $k\times n$의 정수배가 아닐 경우 $k\times n$의 정수배가 되도록 길이를 조정한다. 이 때, 실제 메시지 길이와 패딩을 구별하기 위해서 메시지 블록의 처음 부분에 패딩이 차지하는 길이를 알려주는 패딩 길이 필드 PLF(Padding Length Field)를 첨가한다. PLF의 길이($|PLF|$)는 패딩의 길이가 최대 $k\times n$ 비트를 넘지 않으므로 $\lfloor \log(k\cdot n-1) \rfloor +1$로 고정한다. 평문 M의 길이와 PLF의 길이의 합이 $k\times n$의 정수배가 되면 패딩을 하지 않지만 그렇지 않은 경우는 임의의 값으로 패딩을 함으로써 길이를 맞춘다. 길이가 조정된 평문을 X라 할 때, 패딩의 길이는 X의 길이에서 평문 M과 PLF가 차지하는 길이를 뺀 $|X|-|M|-|PLF|$로서 PLF의 값과 같다. 평문 M에 패딩과 PLF가 추가된 메시지 X가 가지는 $k\times n$길이 블록의 수를 u라 할 때 $u=\lceil |X|/(k\cdot n) \rceil$이다. 이렇게 생성된 메시지 X는 그림 1과 같다.

그림 1 메시지 구성도

3.2.2 암호문 생성 단계

평문에 패딩을 첨가한 메시지 X에 대한 암호문 C는 메시지의 각 블록을 암호화한 암호문 블록 $C_1, C_2, \cdots, C_u$의 접합 $C=C_1 \| C_2 \| \cdots \| C_u$으로 구성된다. 암호문 블록 C_i $(1\le i\le u)$는 공개키 행렬 P에 메시지 블록 X_i를 곱하여 생성된 $m\times 1$ 행렬이다. 즉, $k\times n$ 비트 길이의 메시지 블록 X_i를 k 비트 길이를 가지는 n개의 원소로 구성된 $n\times 1$ 행렬로 표현하고, 암호문 블록 C_i를 다음과 같이 계산한다.

$$C_i=P\cdot X_i=\begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ & \vdots & & \\ p_{m,1} & p_{m,2} & \cdots & p_{m,n} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}=\begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix},\quad 1\le i\le u$$

3.3 복호화

암호문으로부터 평문을 복호화하는 단계에서는 공개키와 비밀키를 이용하여 복호화 행렬을 생성하고 암호문을 확장하여 평문을 복원한다.

3.3.1 복호화 행렬 생성 단계

복호화 행렬은 $n\times n$ 정방 행렬 A로서 공개키 행렬 P와 비밀키 행렬 S로부터 생성된다. 먼저, 공개키 행렬 P를 생성하기 위해 사용되었던 교환 순열 π의 역순열 π^{-1}을 계산하고, 이를 행렬 P에 적용하여 행렬 T를 복구한다. 즉, $T_i = P_{\pi^{-1}(i)}$ 이다. 행렬 T와 비밀키 행렬 S를 접합하여 $n\times n$ 정방 행렬 B를 만든다.

$B=(b_{ij})$

$$b_{ij}=\begin{cases} t_{ij}, & \text{if} \quad 1\le i\le m,\ 1\le j\le n \\ s_{i-m,j}, & \text{if} \quad m<i\le n,\ 1\le j\le n \end{cases}$$

위의 방식으로 구한 행렬 B의 역행렬을 계산함으로써 복호화 행렬 A를 구한다.

$$B=\begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ \vdots & & & \\ t_{m,1} & t_{m,2} & \cdots & t_{m,n} \\ s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ \vdots & & & \\ s_{n-m,1} & s_{n-m,2} & \cdots & s_{n-m,n} \end{bmatrix},\quad A=B^{-1}=\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

3.3.2 암호문 확장 단계

이 단계에서는 공개키 행렬 P에 의해 암호화된 $m\times 1$ 크기의 암호문 블록들을 변형하고 확장하여 $n\times 1$ 크기의 확장된 암호문 블록 $C'_i=(c'_1, c'_2, \ldots, c'_n)$으로 만든다. 행렬 C'_i의 원소값 c'_1부터 c'_m은 C_i의 원소값에 역순열 π^{-1}을 적용하여 구하고, c'_{m+1}부터 c'_n은 각각 $c_{\pi(1)}$ 부터 $c_{\pi(n-m)}$ 값에 w^{-1} 을 곱하고 mod N을 취하여 생성한다 (단, w^{-1}는 $w\cdot w^{-1}\equiv 1 \bmod N$을 만족한다. $\gcd(w,N)=1$이기 때문에 w^{-1}는 유일하게 결정된다).

$C'_i=(c'_j)$

$$c'_j=\begin{cases} c_{\pi(j)}, & \text{if} \quad 1\le j\le m \\ w^{-1}\cdot c_{\pi(j-m)} \bmod N, & \text{if} \quad m<j\le n \end{cases}$$

위의 방식으로 구한 변형된 암호문은 다음과 같이 표현된다.

$$C'_i=\begin{pmatrix} c'_1 \\ \vdots \\ c'_m \\ c'_{m+1} \\ \vdots \\ c'_n \end{pmatrix}=\begin{pmatrix} c_{\pi(1)} \\ \vdots \\ c_{\pi(m)} \\ w^{-1}\cdot c_{\pi(1)} \bmod N \\ \vdots \\ w^{-1}\cdot c_{\pi(n-m)} \bmod N \end{pmatrix}$$

3.3.3 평문 복호화 단계

메시지 블록 X_i는 복호화 행렬 A에 확장된 $n\times 1$ 암호문 블록 C'_i를 곱하여 복원한다.

$$X_i=A\cdot C'_i=\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ & \vdots & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}\cdot\begin{pmatrix} c'_1 \\ \vdots \\ c'_n \end{pmatrix}=\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

확장된 암호문의 확장 부분인 $n-m$행은 암호문 C_i의 $n-m$행을 변형하여 구한다. 확장된 암호문의 변형에 이용되는 암호문 C_i의 $n-m$행은 $C_i=P\cdot X_i=(w\cdot S \bmod N)\cdot X_i$와 같다. 따라서 확장된 암호문의 $n-m$행은 $C'_i=w^{-1}\cdot C_i \bmod N=(w^{-1}\cdot(w\cdot S \bmod N)\cdot X_i) \bmod N=S\cdot X_i$가 된다. 또한 공개키 행렬과 비밀키 행렬을 접합한 행렬은 역행렬이 존재하는데, 임의의 정방행렬에 대한 역행렬은 유일하다. 따라서 $X_i=A\cdot C'_i$가 되며, 복호화된 메시지 블록 X_i는 암호화하기 전의 메시지 블록 X_i와 동일하다.

암호문 블록 C'_i을 메시지 블록으로 복호화 하였을 때 $i=1$인 경우 메시지 블록의 처음 $\lfloor\log(k\cdot n-1)\rfloor+1$ 비트는 패딩 길이 필드(PLF)이고, $i=u$인 경우 마지막 비트로부터 패딩 길이 필드(PLF)의 값에 해당하는 비트 수만큼 패딩이므로 이들을 제거함으로써 원래의 메시지 블록을 얻을 수 있다.

3.4 파라미터 및 변수의 설정 배경

키 생성 이전에 설정했던 변수들이 가져야 하는 조건은 다음과 같은 이유에서이다. 만약 $x\cdot y \bmod N=r$이면 $x\cdot y+N \bmod N=r$이다. 비밀키의 한 행과 평문의 벡터 곱은 최대 $\sum_{j=1}^{n} s_{ij}\cdot(2^k-1)$인데 N이 $N>\max\left\{\sum_{j=1}^{n} s_{ij}\cdot(2^k-1)\ \middle|\ i=1,2,\ldots,n-m\right\}$을 만족하지 않으면 $x\cdot y$과 $x\cdot y+N$이 모두 $\sum_{j=1}^{n} s_{ij}\cdot(2^k-1)$보다 작을 수 있다. 그러므로 확장된 암호문 블록 C'_i를 생성할 때, N의 범위가 위의 조건을 만족하지 않으면 암호문 블록의 서로 다른 두 개의 원소값에 대응되는 확장 암호문의 원소값이 동일할 수 있기 때문에 (즉, $c'_i=c'_j$ s.t. $c_i\neq c_j$) 변수 N은 위와 같은 조건을 만족해야만 한다.

또한 비밀키 (S,w,N,π) 중에서 w가 $1<w<N$ 이고 $\gcd(w,N)=1$인 조건을 만족하지 않을 경우, 공개키의 일부 원소가 되는 $w\cdot s_{ij} \bmod N$ 의 값이 0과 $N-1$ 사이에 고르게 분포하지 않고 몇 개의 정수값에 치중된다[14].

4. 결과 및 분석

본 절에서는 본 논문이 제안한 알고리즘의 안전성을 평가하고 배낭꾸리기 암호 시스템이 받은 공격에 대해 안전함을 보인다. 다음으로 알고리즘의 각 단계별 시간 복잡도와 공간 복잡도를 계산하여 효율성에 대해 평가한다.

4.1 알고리즘의 안전성

공개되어 있는 정보만을 이용하여 하나의 메시지 블록 X_i를 구하려면 $m\times n$ 공개키 행렬 P와 $m\times 1$ 암호문 블록 C_i을 이용하여 $P\cdot X_i=C_i$인 X_i를 구해야 한다. 이것은 n개 미지수를 가지는 m개의 일차식으로 구

성된 연립방정식을 푸는 문제와 같다. 즉,

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ \vdots & & & \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} \Rightarrow \begin{array}{l} p_{11}x_1 + \cdots + p_{1n}x_n = c_1 \\ \vdots \\ p_{m1}x_1 + \cdots + p_{mn}x_n = c_m \end{array}$$

m, n은 부등식 $m<n$을 만족하므로 연립방정식은 식의 수보다 미지수의 수가 더 많은 부정방정식이 된다. 메시지 블록의 크기를 k 비트라 했을 때, 미지수 x_i가 가질 수 있는 값의 범위는 $0 \le x_i \le 2^k-1$이다. n개의 미지수 x_i와 m개의 식이 있을 때 조건을 만족하는 미지수의 값을 결정하기 위해서는 $2^{(n-m)k}$번 값을 대입해 보아야 한다. 따라서 전수조사를 이용하여 하나의 암호문에 대해 대응하는 하나의 평문 블록을 구하는 것은 지수 시간(exponential time)이 걸리기 때문에 계산상 불가능하다.

만약 비밀키 (S, w, N, π) 중에서 변수 N의 값이 알려지면, w^{-1}를 예측하고 $s_{ij} = w^{-1} \cdot p_{ij} \bmod N$ $(1 \le i \le m, 1 \le j \le n)$을 계산하여 비밀키 행렬 S를 알아낼 수 있다. 그러나 N값을 이용하여 w^{-1}을 찾아낸다 해도 행렬 P의 한 행과 대응하는 행렬 S의 행을 찾기 위해서는 $\binom{m}{n-m} \cdot (n-m)!$번 즉, $\frac{m!}{(2m-n)!}$ 만큼의 비교연산이 필요하기 때문에 이 방법 역시 계산상 불가능하다.

배낭꾸리기 공개키 암호 시스템의 비밀키는 초증가 수열이며 공개키와 비밀키의 원소들은 일대일로 대응된다. Adi Shamir는 이러한 특성을 이용하여 배낭꾸리기 공개키 암호 시스템의 일반적인 공격법을 발표하였다[10,11]. 즉, 공개키 생성 방정식 $a_i \equiv b_i \cdot W \bmod M$으로부터 $a_i U - k_i M = b_i$ $(U \equiv W^{-1} \bmod M)$를 만족하는 정수 $k_1, \ldots, k_n$이 존재한다고 할 때 $|b_i k_1 - b_1 k_i| \le M 2^{i-n}$ 와 같은 부등식 3~4개와 Lenstra의 선형계획법 알고리즘을 통하여 k_i 변수값을 구함으로써 다항식 시간내에 해를 구할 수 있게 되었다[4]. 이러한 공격법은 비밀키가 초증가 수열이라는 특성을 가질 때 가능하다. 본 논문에서 제안한 공개키 암호 알고리즘은 비밀키의 원소값을 임의로 선택하기 때문에 Shamir의 공격법에 대하여 안전하다.

4.2 알고리즘의 효율성

공개키 암호 알고리즘이 실용적이기 위해서는 암호화와 복호화가 빠른 시간에 이루어져야 한다. 본 논문에서 제안한 알고리즘의 시간적, 공간적 측면에서의 효율성을 분석한다.

4.2.1 시간적 측면에서의 효율성

연산의 수행시간은 암호화, 복호화를 할 때 낮은 차수의 다항식 시간 이내에 이루어져야 한다. 암호화 단계에서 암호문 블록 C_i는 $c_j = \sum_{k=1}^{n} p_{jk} \cdot x_k$, $(1 \le j \le m)$ 방법으로 구하면 되므로 암호문 메시지 블록을 만드는 데는 nm번의 곱셈과 nm번의 덧셈 연산을 수행해야 한다. 따라서 곱셈 연산과 덧셈 연산을 기본 연산으로 정했을 때 곱셈 연산 $O(nm)$, 덧셈 연산 $O(nm)$의 시간 복잡도를 가진다.

복호화 단계에서 평문을 구하기 위해서는 복호화 행렬을 생성하고 암호문을 확장하여 복호화 행렬과 확장된 암호문을 곱해야 한다. 복호화 행렬 A는 공개키 행렬과 비밀키 행렬을 접합한 행렬의 역행렬이 된다. 복호화 행렬은 (비밀키, 공개키) 쌍에 대하여 한번만 계산해 놓으면 되므로, 시간 복잡도에는 큰 영향을 미치지 않는다. 확장된 암호문은 $2m-n$번의 $c'_j = w^{-1} \cdot c_j \bmod N$ 연산을 수행하고 평문으로 복호화하는데 $x_j = x_j + a_{jk} \cdot c'_j$을 n^2번 수행한다. 따라서 곱셈 연산 $O(n^2)$, 덧셈 연산 $O(n^2)$의 시간 복잡도를 가진다.

따라서 암호문을 얻는 암호화 단계나 평문을 구하는 복호화 단계 모두 다항식 시간 내에 수행될 수 있다.

4.2.2 공간적 측면에서의 효율성

메시지 블록의 크기를 k 비트라 하고 비밀키 행렬 S의 각 원소를 d 비트라고 가정하면, $(n-m) \times n$ 행렬 S는 최대 $(n-m) \times n \times d$ 비트의 저장공간이 필요하다. $N = \max\left\{ \sum_{j=1}^{n} s_{ij} \cdot (2^k-1) \mid i = 1, 2, \ldots, n-m \right\}$이므로 N값은 최대 $n \cdot (2^d-1) \cdot (2^k-1)$ 값을 가지게 되므로 이를 나타내기 위해 최대 약 $(d+k+\log n)$ 비트가 필요하다. w의 값의 범위는 $1 < w < N$ 이므로 최대 $(d+k+\log n)$ 비트이므로, 비밀키 전체를 저장하는 데는 $(n-m) \times n + 2(d+k+\log n) + (n-m) \times n \times d$ 비트의 공간이 필요하다.

한편, 공개키인 공개키 행렬 P의 한 원소의 크기는 법 N의 최대 크기와 같으므로 최대 $(d+k+\log n)$ 비트가 된다. 따라서 $m \times n$ 크기의 공개키 행렬 P를 저장하기 위해 필요한 최대 저장공간의 크기는 $(d+k+\log n) \times m \times n$ 비트이다.

마지막으로, 평문과 암호문의 크기는 다음과 같이 계산된다. 평문 전체 크기는 하나의 메시지 블록 크기×메시지 블록의 개수이므로 $(k \times n) \times u$ 비트의 공간이 필요하다. 암호문 전체의 크기는 하나의 암호문 블록의 크기

암호문 블록의 개수이므로 $(d+2k+2\log n)\times m\times u$ 비트의 공간이 필요하다.

5. 결 론

본 논문에서는 정수계획법의 특수한 형태를 이용한 새로운 공개키 암호 알고리즘을 제안하였다. 일반적인 전수 조사로 암호문에 해당하는 평문을 얻기 위해서는 $2^{(n-m)k}$번의 지수 시간 연산을 필요로 한다. 공개키를 비밀키로 변형시키는 키 생성의 단계가 기존의 배낭꾸리기 공개키 암호 알고리즘과 유사하지만 배낭꾸리기 공개키 암호 알고리즘에서 보였던 비밀키의 취약점을 보완함으로써 유사한 암호학적 공격을 피할 수 있다. 또한 암호화와 복호화 단계의 시간 복잡도는 각각 $O(nm)$과 $O(n^2)$으로서 다항식 시간 안에 암호화, 복호화가 가능하다. 공간 복잡도 면에서는 $k\times n$크기의 메시지 블록에 대해 $nk+\frac{1}{2}nd+n\log n$ 길이의 암호문이 생성된다. 알고리즘의 안전성을 증가시키면서 암호문의 증가분을 적게 하기 위해서는 키의 크기를 크게 하기보다는 메시지 블록의 크기를 늘리는 것이 더 효율적이다.

제안한 공개키 암호 알고리즘의 변수값들은 안전성과 효율성 면에서 서로 엇갈림 (trade-off)이 있다. 그러므로, 실제적으로 구현되어 이용될 때 가장 안전하면서도 효율적일 수 있는 변수값의 설정과 비밀키와 공개키를 생성할 때 약한 키가 되는 경우와 일반적인 형태의 정수계획법을 이용한 공개키 암호 알고리즘에 대한 향후 연구가 필요하다.

참 고 문 헌

[1] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, Vol. IT-22, pp. 644-654, 1976.

[2] T. ElGamal, "A Public-Key Cryptosystem and A Signature Scheme Based on Discrete Logarithms," IEEE Transaction on Information Theory, Vol. IT-31, pp. 469-472, 1985.

[3] N. Koblitz, "Elliptic Curve Cryptosystems," Mathematics of Computation, Vol. 48, No. 177, pp. 203-209, 1987.

[4] H. W. Lenstra, Jr., "Integer Programming with a Fixed Number of Variables," Math. of Operations Research, Vol. 8, No. 4, pp. 538-548, 1983.

[5] R. J. McEliece, "A Public-Key Cryptosystem Based on Algebraic Coding Theory," Deep Space Network Progress Report, pp. 42-44, 1978.

[6] R. C. Merkle and M. E. Hellman, "Hiding Information and Signature in Trap-door Knapsacks," IEEE Transactions on Information Theory, Vol. IT-24, pp. 525-530, 1978.

[7] M. O. Rabin, "Digital Signatures and Public-Key Functions as Intractable as Factorization," MIT Laboratory for Computer Science, Technical Report, 1979.

[8] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," Comm. ACM, Vol. 21, pp. 120-126, 1978.

[9] S. Sahni, "Computationally Related Problems," SIAM J. Comput., Vol. 3, pp. 262-279, 1974.

[10] A. Shamir, "On the Cryptocomplexity of Knapsack System," Proc. 11th ACM Symp. on Theory of Computing, pp. 118-129, 1979.

[11] A. Shamir, "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem," Proceedings of the 23rd Annual Symposium on the Foundations of Computer Science(IEEE), pp. 145-152, 1982.

[12] Howard Anton, *Elementary Linear Algebra*, Anton Textbooks, 1987.

[13] Michael R. Garey and David S. Johnson, *Computers and Intractability*, Freeman, 1979.

[14] Charles P. Pfleeger, *Security in Computing*, Prentice-Hall, 1997.

[15] Kenneth H. Rogen, *Elementary Number Theory and Its Application*, Addison-Wesley,1993.

[16] Bruce Schneier, *Applied Cryptography*, Wiley, 1996.

용 승 림

1998년 2월 이화여자대학교 공과대학 컴퓨터학과 학사. 2000년 2월 이화여자대학교 공과대학 컴퓨터학과 석사. 2000년 ~ 현재 이화여자대학교 과학기술대학원 박사과정

조 태 남

1986년 이화여자대학교 전자계산학과 이학사. 1988년 이화여자대학교 전자계산학과 이학석사. 1988년 ~ 1996년 한국전자통신연구원 선임연구원. 1998년 ~ 현재 이화여자대학교 컴퓨터학과 박사과정.

이 상 호
1979년 서울대학교 계산통계학과 이학사. 1981년 한국과학기술원 전산학과 이학석사. 1987년 한국과학기술원 전산학과 공학박사. 1990년 미국 일리노이대학교 전산학과 방문교수. 현 이화여자대학교 컴퓨터학과 교수. 관심분야는 알고리즘 설계, 계산기하학, 그래프이론, 알고리즘 애니메이션 등임.