임계 암호시스템 구현을 위한 능동적 비밀 분산에서의 공유 갱신 방법

(Share Renewal Scheme in Proactive Secret Sharing for Threshold Cryptosystem)

이 윤 호 † 김 희 열 † 정 병 천 †† 이 재 원 †† 윤 현 수 †††
(Youn-Ho Lee) (Heeyoul Kim) (Byung-Chun Chung) (Jaewon Lee) (Hyunsoo Yoon)

요 약 비밀 분산(Secret Sharing)은 임계 암호시스템(Threshold Cryptosystem)의 기본 개념이며, 현대 암호학에서 중요한 한 축을 이루는 암호학의 한 분야이다. 1995년 Jarecki는 이동 공격자 모델에 대하여 비밀 분산 프로토콜의 안전성을 유지할 수 있는 능동적 비밀 분산(Proactive Secret Sharing) 개념을 제안하였고, 이와 함께 (k, n) threshold scheme에서 능동적 비밀 분산을 적용하기 위하여 공유 갱신 프로토콜을 제안하였다. Jarecki가 제안한 공유 갱신 프로토콜은 전체 참여자가 n명일 경우 각 참여자당 $O(n^2)$의 모듈라 멱승 연산을 수행하여야 한다. 이것은 매우 큰 연산량으로 참여자가 큰 대규모의 비밀 분산 수행시 계산 비용의 증가로 Jarecki의 프로토콜은 사용하기 어렵게 된다. 본 논문에서는 (k, n) threshold scheme에서의 능동적 비밀 분산을 위한 효율적인 공유 갱신 방법을 제안한다. 제안 방법은 이전 방법과는 달리 각 참여자당 $O(n)$의 모듈라 멱승 연산만으로 공유 갱신이 가능하다. 이와 함께 본 논문에서는 $k<\frac{1}{2}n-1$ 인 경우에 대하여 제안 방법의 안전함을 증명한다.

키워드 : 임계 암호시스템, 비밀 분산 , 능동적인 비밀 분산

Abstract The secret sharing is the basic concept of the threshold cryptosystem and has an important position in the modern cryptography. At 1995, Jarecki proposed the proactive secret sharing to be a solution of existing the mobile adversary and also proposed the share renewal scheme for (k, n) threshold scheme. For n participants in the protocol, his method needs $O(n^2)$ modular exponentiation per one participant. It is very high computational cost and is not fit for the scalable cryptosystem. In this paper, we propose the efficient share renewal scheme that need only $O(n)$ modular exponentiation per participant. And we prove our scheme is secure if less than $\frac{1}{2}n-1$ adversaries exist and they are static adversary.

Key words : Secret Sharing, Proactive Secret Sharing , Threshold Cryptography, (k,n) threshold scheme

1. 서 론

정보 통신 기술의 급속한 발달 및 인터넷의 보급은 가정 또는 원격지에서 많은 업무를 처리할 수 있게 되었다. 이러한 업무의 대표적인 예가 인터넷 뱅킹과 전자상거래 등이다. 이와 같은 업무를 수행할 때 사용자는 자신의 비밀 정보를 기밀성을 유지한 채로 다른 사용자 또는 기관에게 전송하는 경우가 생기게 되었다. 이러한 정보에 대한 기밀성 유지와 사용자에 대한 인증을 위해서 공개키 암호 시스템이 개발되었으며 현재 인터넷 뱅킹등 고도의 보안을 요구하는 서비스에서 널리 사용되고 있다.

일반적인 공개키 암호시스템은 하나의 서버에 자신의

· 본 연구는 첨단 정보기술 연구 센터를 통하여 과학재단의 지원을 받았고 대학 IT연구센터 육성·지원 사업의 연구 결과로 수행되었음.
† 학생회원 : 한국과학기술원 전자전산학과
yhlee@camars.kaist.ac.kr
hykim@camars.kaist.ac.kr
†† 비 회 원 : 한국과학기술원 전자전산학과
bcchung@camars.kaist.ac.kr
jaewon@camars.kaist.ac.kr
††† 종신회원 : 한국과학기술원 전자전산학과
hyoon@camars.kaist.ac.kr
논문접수 : 2002년 7월 19일
심사완료 : 2003년 1월 24일

개인키를 갖고 서명, 암호화등 암호학적 작업을 수행한다. 만약 서버의 개인키가 노출될 경우 공개키 암호시스템이 깨어지게 되며 시스템 공격자들은 이러한 개인키가 저장된 단 하나의 서버만 공격하면 되는 것이다.

임계 암호시스템(Threshold Cryptography)은 이러한 키의 집중에 관한 문제를 해결할 수 있는 한가지 방법이 될 수 있다. 이것은 일반적인 암호 시스템에서 사용하는 하나의 키를 여러 서버에게 분산하여 보호할 수 있는 기술이다. 예를 들어 공인 인증기관의 개인키와 같은 중요한 키를 하나의 서버에 저장하여 사용하였을 경우 개인키를 갖고 있는 서버만을 공격하면 개인키가 유출된다. 그러나 임계 암호시스템에서는 하나의 개인키를 여러 개의 키에 대한 공유(share)로 나누어서 다수의 서버에게 공유를 나누어 주게 되며, 원래의 하나의 개인키로써 수행하던 작업을 다수의 공유를 이용하여 수행한다. 따라서 공격자는 다수의 서버를 공격하여 일정한 수 이상의 공유를 모을 경우에만 원래의 개인키를 알 수 있게 되는 어려움이 있게 된다[1].

임계 암호시스템의 방법은 암호학의 한 분야인 비밀 분산을 근거로 한다. 비밀 분산은 다수가 어떤 비밀 정보를 분산, 공유하여 일정한 조건을 만족해야만 원래의 비밀 정보를 복구할 수 있는 것을 목표로 하는 안전한 프로토콜을 찾는 분야이다. 이 중 (k,n) threshold scheme은 비밀 정보를 n개의 공유로 나누어서 그 중 k개 이상의 공유가 모일 경우 비밀 정보를 복구할 수 있고 그 미만의 공유가 모일경우 비밀 정보에 관하여 어떠한 정보도 알 수 없도록 하는 비밀 분산의 한 분야이다. 이 분야는 1979년 Shamir가 제안하였으며 다항식 보간법(polynomial interpolation)을 이용한 (k,n) threshold scheme 방법도 동시에 제안하였다[2].

비밀 분산은 방법의 안전성의 증가를 위하여 많은 발전이 있어왔으며 검증 가능한 비밀 분산(Verifiable Secret Sharing)[3-5], 능동적 비밀 분산 (Proactive Secret Sharing)[6]등으로 발전하여 왔다. 1995년 Jarecki는 능동적 비밀 분산을 위한 (k,n) threshold scheme에서의 공유 갱신 방법을 제안하였다. 이 방법은 각 참여자 당 $O(nk)$의 모듈라 멱승을 필요로 한다. 일반적으로 k값은 $O(n)$에 해당하는 값이므로 결국 각 참여자는 공유 갱신 시 약 $O(n^2)$의 연산을 필요로 한다. 이것은 상당한 연산량이며 실용적으로 쓰이기에는 부적합하다.

본 논문에서는 (k,n) threshold scheme에서 각 참여자의 연산량이 $O(n)$을 갖는 새로운 방법을 제안한다. 이것은 이전 방법이 각 참여자당 자신만이 알고 있는 개인값이 k개인 반면, 제안 방법은 상수개인 것에 근거한다. 또한 제안 프로토콜의 안전성은 k-1 또는 그 이하의 공격자가 프로토콜에 참여해도 다른 참여자가 갖게 되는 개인값을 알아낼 수 없다는 방법을 사용하여 증명하며 이 때 이러한 k값이 $\frac{1}{2}n-1$ 보다 작을 경우, 프로토콜이 안전함을 증명한다. 증명 방식의 모델은 이전의 연구 논문을 참조한다[7-8].

본 논문의 구성은 다음과 같다. 2장에서는 Pederson이 제안한 검증 가능한 비밀 분산 및 그것을 이용하여 구현한 Jarecki의 능동적 비밀 분산 방법에 대해 살펴본다. 3장에서는 본 논문에서 제안한 새로운 공유갱신 프로토콜에 대하여 설명하며 4장에서는 제안 프로토콜 및 이전 프로토콜의 계산 복잡도 및 망 사용량 복잡도를 분석하고, 전체 프로토콜을 실제로 구현하여 수행시간을 측정한 결과를 기술한다. 마지막으로 5장에서는 결론을 내린다.

2. 관련 연구

2.1 Pederson의 검증 가능한 비밀 분산 (Pederson's Verifiable Secret Sharing)

검증 가능한 비밀 분산은 공유를 분배받는 각 참여자에게 자신이 받는 공유가 올바른 값이라는 검증 작업을 할 수 있도록 제안된 방법이다. 검증 가능한 비밀 분산에서 딜러는 각 참여자에게 자신이 분배한 공유에 대한 검증값을 동보하며 공유를 받은 각 참여자는 딜러가 동보한 검증값을 이용하여 자신이 받은 공유가 올바른 값인지를 검증한다.

본 절에서는 검증 가능한 비밀 분산을 구현한 방법들 중, 본 논문과 직접 연관이 있는 이산 대수 문제를 이용하여 검증값을 제공하는 프로토콜에 대하여 기술한다. 검증 가능한 프로토콜에서 이산 대수 문제를 이용한 검증값을 제안한 대표적인 논문은 Feldman이 1987년 제안한 논문이 있다[4]. 그러나 이 논문에서 제안한 방법은 검증값에 대하여 정보의 누출이 일어나게 되어 공격 방법이 존재한다[1,9,10]. 이러한 공격 방법에 대한 약점을 개선한 방법이 1991년 Pederson에 의해 제안되었다[5,6]. 이 방법은 최근에도 참조되고 있는 방법이며 이것을 확장한 많은 프로토콜이 제안되었다[1,9,10,11].

Pederson의 프로토콜은 특정한 비밀 정보가 정해져 있지 않고 분산 키 생성과 같이 각 참여자의 비밀값을 합친 새로운 비밀값에 대한 비밀 분산 작업을 수행할 경우 믿을 수 있는 제 3자이며 공유 값을 다른 참여자에게 나누어주는 역할을 하는 딜러가 존재하지 않는 프로토콜로 확장할 수 있다. 이 때에 각 참여자 모두는 프

로토콜에서 딜러의 역할을 수행한다. 따라서 각 참여자는 다른 모든 참여자가 생성한 자신의 공유에 대하여 검증을 수행해야 한다. 그러므로 자신을 제외한 n-1명이 생성한 k-1차 다항식의 계수를 검증해야 한다. 각 참여자는 다른 하나의 참여자의 다항식을 검증하기 위해 k+1번의 모듈라 멱승 연산을 수행해야 하므로 각 참여자는 최종적으로 $O(kn)$번의 모듈라 멱승 연산을 수행해야 한다.

2.2 능동적 비밀 분산(Proactive Secret Sharing)

능동적 비밀 분산은 기존의 공격자 모델보다 좀 더 강력한 공격자 모델인 이동 공격자(Mobile Adversary) 모델에 대하여 비밀 분산 작업을 안전하게 수행하게 하기 위해 제안된 개념이다. 이동 공격자 모델은 공격자가 특정 주기를 두어 매 주기마다 프로토콜에 침입하는 것이다[12]. 따라서 프로토콜 내에서 특정 주기 내에 현재 주기 안에 있는 공격자를 고려해 주지 않는다면, 다음 주기에서의 공격자들은 이전 주기에서 수집한 프로토콜에 관한 정보를 이용하여서 좀 더 강한 공격을 할 수 있다.

능동적 비밀 분산은 이러한 이동 공격자가 존재하는 상황에서 안전한 비밀 분산을 수행할 수 있도록 제안된 개념이다. 1995년 Jarecki는 능동적인 비밀 분산을 이루기 위해서 주기적인 공유의 갱신을 통한 해결 방법을 제시하였다[7]. 이것은 임의의 비밀 정보에 대하여 이미 비밀 분산을 수행하여 공유들을 분배받은 상태에서 수행하는 것으로 새로운 공유 갱신값을 더하여 기존 공유에 대한 안전도를 높이는 방법이다. 공유 갱신 방법을 간단히 요약하면 딜러가 없는 Pederson의 비밀 분산 프로토콜에서 각 참여자가 만드는 다항식의 상수항을 0으로 하는 것이다.

만약 공유 갱신 이전의 비밀 분산에 사용된 다항식을 $f_{old}(x)$라 한다면, 공유 갱신의 결과로 생긴 새로운 공유는 결과적으로 다음의 다항식 $f_{new}(x)$에서 공유를 분배한 결과와 같다.

$$f_{new}(x)=f_{old}(x)+\sum_{i\in QUAL} f_i(x)$$

이 때 새로 더한 $f_i(x)$의 계수의 합은 프로토콜에 참여한 모든 참여자가 각각 새로 만든 다항식의 계수를 알아야만 알 수 있는 값이므로 $f_{old}(x)$에 관한 정보를 갖고 있는 공격자도 새로운 다항식 $f_{new}(x)$에 관하여 정보를 알 수 없게 된다. 따라서 공격자는 새로운 공유에 대한 정보를 얻을 수 없다.

Jarecki의 공유 갱신 프로토콜은 Pederson의 검증 가능한 비밀 분산 프로토콜에서 전체 참여자가 모두 딜러의 역할을 한번씩 수행한 것과 연산량이 갖게 된다. 따라서 위에서 설명한 바와 같이 각 참여자는 공유 갱신을 위해 전체 $O(kn)$번의 모듈라 멱승 연산을 수행해야 한다. 일반적으로 k는 $O(n)$의 값이므로 결국 $O(n^2)$의 모듈라 멱승 연산을 수행해야 한다. 이것은 매우 많은 연산량이며 참여자가 많아지게 될 경우 공유 갱신 프로토콜을 수행하기 위하여 많은 수행시간이 소요된다.

3. 빠른 공유 갱신 프로토콜

본 장에서는 Jarecki의 공유 갱신 프로토콜의 수행 속도 문제를 개선한 새로운 프로토콜을 제안한다. 이전 방법은 각 참여자가 새로운 다항식을 만들어 그 계수값을 숨기는 방법을 사용한다. 따라서 각 참여자는 다른 참여자가 숨기는 계수의 값을 검증하기 위하여 다른 한 명의 참여자에 대해 $O(k)$번의 모듈라 멱승을 수행하여야 한다. 따라서 최종적으로 자신을 제외한 n-1명의 참여자에 대하여 검증작업을 수행하므로 전체 $O(kn)$의 모듈라 멱승을 수행한다. 이에 반해 제안 프로토콜을 공개된 다항식에 자신만이 알고 있는 하나의 값을 다항식의 함수 값에 곱하는 방법으로 각 참여자마다 검증 받아야 하는 값을 다항식의 차수에 독립적으로 만들었다. 따라서 각 참여자는 최종적으로 다른 한명의 참여자에 대해 상수번의 모듈라 멱승을 수행하여 최종적으로 $O(n)$번의 모듈라 멱승만으로 검증이 가능하다.

3.1 프로토콜의 가정

3.1.1 통신 채널 모델 가정

프로토콜의 각 참여자는 다른 참여자와 암호학적으로 안전한 전용선으로 연결되어 있다고 가정하며, 이러한 안전한 전용선을 이용하는 비용이 모듈라 멱승 연산 보다 작다고 가정한다. 또한 각 참여자 사이의 전용선은 동기화된 채널(partially synchronous channel)이라고 가정한다[10]. 이것은 각 참여자가 현재 수행되는 프로토콜의 단계에서 받을 수 있는 정보를 자신이 다른 참여자에게 정보를 보내기 이전에 알 수 있다고 가정한다. 마지막으로 전용 동보 채널이 있다고 가정한다.

3.1.2 공격자 모델

프로토콜에 참여한 공격자 A는 k가 임계값일 경우 k-1명까지 될 수 있다고 가정한다. 또한 $k<\frac{n}{2}-1$이라 가정하며, 공격자들은 자신들만의 도청 불가능한 채널이 있어서 공격자들의 정보를 공유한다. 따라서 공격자는 프로토콜의 각 단계 수행시 자신이 다른 정직한 참여자로부터 받은 정보 및 다른 공격자들이 갖고 있는 모든 정보를 알 수 있다. 또한 공격자는 프로토콜 시작시 정

적으로 결정된다고 가정한다.

3.2 제안 프로토콜

본 절에서는 제안 프로토콜에 대한 일련의 수행 과정에 대해 논의한다. 제안 프로토콜은 두 개의 세부 프로토콜로 이루어져 있다. 하나는 확장 분산 난수 분배(Extended Distributed Coin Flip) 프로토콜로서 실제 공유 갱신 프로토콜에서 참여자가 생성한 비밀값을 숨기기 위하여 사용되는 값들을 생성하는 프로토콜이며, 다른 하나는 실제로 공유 갱신 작업을 수행하는 프로토콜이다.

3.2.1 용어 및 기호의 정의

프로토콜에 사용되는 일련의 용어 및 기호의 정의는 아래와 같다. 또한 프로토콜에서 사용되는 간단한 알고리즘에 대하여 설명한다.

- p, q : 연산의 기본이 되는 소수이다. $q = mp + 1$ ($p \gg m$)을 만족하며 프로토콜에서 일반 덧셈 및 곱셈 연산은 Z_p상에서 수행되며 지수승 연산은 Z_p^*상에서 수행된다.
- g, h : Z_p^*상에서의 위수 p를 갖는 생성자이며 $g = h^c$를 만족한다. 여기서 c값은 알려지지 않은 값이다.
- ID_i : 참여자 i의 ID를 나타낸다. ($i = 1, \cdots, n$)
- k, n : (k, n) threshold scheme에서의 임계값 및 전체 참여자를 나타낸다.
- y_i : 새로운 공유 갱신 프로토콜에 사용되는 참여자 i가 생성하는 값으로써 참여자 i만이 알고 있는 값이다. 공유 갱신에 사용되는 새로운 다항식을 정의하는 값으로서 다른 참여자에게 어떠한 정보도 알려져서는 안되는 값이다.
- d_i, e_i : 참여자 i가 생성하는 값으로서 y_i값을 숨기기 위해 사용된다. y_i와 마찬가지로 다른 참여자에게 어떠한 정보도 알려져서는 안되는 값이다.
- $Alg(i, j, k, n)$: i보다 크거나 같고, j보다 작거나 같은 수들 중에서 임의로 k개의 수를 뽑아 수열을 생성하는 작업을 n회 반복하여 결과값인 수열들을 돌려주는 알고리즘이다. ($i-j \le kn$)일 경우 뽑힌 수열들에는 i와 j 사이에 있는 모든 수가 포함되어 있으며 또한 뽑힌 수열 중에서 수열을 이루는 수들이 모두 같은 두 개 이상의 수열이 존재하지 않는다. ($j - i \ge n$, $k < \frac{n}{2} - 1$)을 가정한다.
- $HashExpansion(a, k)$: a의 값을 입력받아서 a의 비트 길이의 k배의 비트 길이를 갖는 임의의 값을 돌려준다. 일반적으로 쓰이는 암호학적 해쉬 함수를 이용하여 구현한다.

3.2.2 확장 분산 난수 분배 프로토콜

분산 난수 분배 프로토콜은 각 참여자가 일련의 수를 생성하여 그것을 다른 참여자들과 교환하여 각 참여자들이 갖게 되는 공유들을 더할 경우 결과값이 특정 값이 되는 공유를 각 참여자들이 나누어 갖는 방법이다[11]. 이 때에 각 참여자들은 자신의 공유에 대하여만 알 수 있으며 다른 참여자들의 공유에 관하여는 전혀 알 수 없어야 한다.

제안 프로토콜을 이와 같은 분산 난수 분배 프로토콜을 확장하여 상수번의 정보 교환으로 분산 난수 분배 프로토콜을 n번 수행한 결과가 되며 그 공유들의 합이 0으로 되는 특정한 프로토콜을 제안한다. 프로토콜의 각 참여자는 최종적으로 n개의 공유를 갖게 된다. 구체적인 방법은 그림 1과 같다.

확장 분산 난수 분배 프로토콜의 결과값은 이후의 공유 갱신 프로토콜에 사용된다. 확장 분산 난수 분배 프로토콜에 대한 안전성은 본 장의 마지막 절에서 증명한다.

3.2.3 공유 갱신 프로토콜

본 세부 절에서는 제안 공유 갱신 프로토콜에 대하여 논의한다. 프로토콜의 대략적인 수행 과정은 각 참여자는 공유 갱신에 사용되어 새로운 다항식을 만드는 곳에 사용되는 핵심값(이제부터 이 값을 개인값(private value)이라 부르기로 한다.)을 생성하고, 그것을 숨기기 위한 은닉값을 더하여 다른 참여자에게 전달한 후, 그 값에 대한 검증 작업을 수행한다. 이 후 모든 참여자는 다른 참여자들의 은닉값으로 사용된 값을 빼어 줌으로써 프로토콜이 종료된다. 자세한 과정은 아래의 그림 2와 같다.

기존의 공유 갱신 방법은 처음 공유 분배에 사용된 k-1차 다항식을 $f_{old}(x)$라 하면, 새로운 공유 갱신 다항식 $f_{new}(x)$는 각 참여자 i가 각기 새로운 상수항이 0인 k-1차 다항식 $f_i(x)$를 생성하여 더한 결과인 $f_{new}(x) = f_{old}(x) + f_1(x) + f_2(x) + \cdots + f_n(x)$이었다[6,7]. 이에 반하여 제안 방법의 새로운 다항식 $f'_{new}(x)$는 공개된 각 참여자의 상수항이 0인 $k-1$차 다항식에 공개하지 않는 각 참여자의 난수값 y_i를 곱하여 모두 더한 다항식이 되어 최종적으로 $f'_{new}(x) = f_{old}(x) + y_1 f_1(x) + y_2 f_2(x) + \cdots + y_n f_n(x)$가 된다. 각 참여자는 이러한 다항식의 함수의 x값이 자신의 ID를 대입하여 새로운 공유를 얻게 되므로 새로운 공유는 상수항이 비밀값인 다항식에서 공유를 분배한 것과 같은 결과를 얻게 된다. 또한 공격자는 갱신에 사용된 다항식 $y_1 f_1(x) + \cdots + y_n f_n(x)$에서 각 참여자의 난수값 $y_1, \cdots, y_n$을 알 수 없으므로 갱신된 새로운 공유에

확장 분산 난수 분배 프로토콜

* 각 참여자 i의 입력값 $(i=1,\ldots,n)$

$\{\alpha_{i1},\alpha_{i2},\ldots,\alpha_{in}\}$ $(\Sigma_{j=1}^{n}\alpha_{ij} \pmod p)=0$, $\alpha_{ij}\in Z_p$ $(j=1,\ldots,n)$),

$\{\beta_{i1},\beta_{i2},\ldots,\beta_{in}\}$ $(\beta_{ij}\in Z_p$ $(j=1,\ldots,n)$

* 공개된 입력값 : p,q,g,h

* 각 참여자 i의 결과값

$\{\delta_{ij}\}_{j=1,\ldots,n}$, $\{\lambda_{ij}\}_{j=1,\ldots,n}$ ($\Sigma_{m=1}^{n}\delta_{mj}=0$, $\delta_{ij}\in Z_p$, $\lambda_{ij}\in Z_p$) , $\{\{\gamma_{mj}\}_{j=1,\ldots,k+1}\}_{m=1,\ldots,n'}$

* 수행과정

1. 각 참여자 i는 α_{ij},β_{ij}를 참여자 j에게 보낸다. $(i\le j\le n$, $j\neq i)$

2. i는 $t_i=\Sigma_{m=1}^{n}\beta_{im} \pmod p$ 를 계산하고 이 결과값을 이용하여 자신의 수열들에 대한 검증값 $g^{\alpha_{ij}}h^{\beta_{ij}} \pmod q$, $h^{t_i} \pmod q$ 를 공개한다. $(1\le j\le n,\ j\neq i)$

3. i는 다른 참여자들의 검증값을 이용하여 다른 참여자에게서 전송 받은 값들을 다음과 같은 수식의 만족여부를 이용하여 검증한다.

$$\Pi_{j=1}^{n}(g^{\alpha_{mj}}h^{\beta_{mj}}) \pmod q \stackrel{?}{=} h^{t_i} \pmod q$$

$$(g^{\alpha_{mi}}h^{\beta_{mi}}) \pmod q \stackrel{?}{=} (g)^{\alpha_{mi}}\times(h)^{\beta_{mi}} \pmod q$$

4. 만약 검증값이 통과하지 않는 값이 발생할 경우 그 값을 받은 참여자 i는 잘못된 값을 보낸 참여자 j가 잘못된 값을 보냈다는 선언 및 참여자 j가 보낸 값을 동보한다. 만약 j가 보낸 값이 잘못된 값이라면, j는 프로토콜에서 제거되며 그렇지 않은 경우 참여자 i가 프로토콜에서 제거된다.

5. 프로토콜에서 제거된 참여자가 생성한 값은 모두 삭제하며 만약 4에서 자신의 수열이 공개되었으나 올바른 참여자 임이 판명된 경우, 다시 수열을 생성하여 다른 참여자에게 분배한다. 또한 제거된 참여자에게 나누어주었던 값을 자신의 값에 모두 더하고 모든 참여자는 그에 해당하는 검증값을 갱신한다.

6. 참여자중 프로토콜에서 제거되지 않은 참여자들을 차례로 $1,\ldots,n'$으로 다시 순서를 짓는다.

7. 각 참여자는 $Alg(1,n',k+1,n')$을 수행하여 결과값이 되는 수열의 값을 $\{\{\gamma_{mj}\}_{j=1,\ldots,k+1}\}_{m=1,\ldots,n'}$ 이라 한다. 모든 참여자는 동일한 수열값을 얻는다.

8. $\delta_{ij}=\Sigma_{m=1}^{k+1}\alpha_{\gamma_{mi}} \pmod p$, $\lambda_{ij}=\Sigma_{m=1}^{k+1}\beta_{\gamma_{mi}} \pmod p$, $(j=1,..,n')$ 을 계산하여 프로토콜의 결과값을 얻는다.

그림 1 확장 분산 난수 분배 프로토콜

대하여 알 수 없다.

이전 프로토콜과 제안 프로토콜에서 각 참여자가 다른 참여자들에게 공개하지 않는 개인값을 비교하면, 이전 방법에서는 각 참여자가 생성한 $k-1$차 다항식의 각 계수, 즉 $(k-1)$개인 것에 비하여 제안 프로토콜에서는 각 참여자의 난수값 y_i값만을 공개하지 않는다. 따라서 검증에 사용되는 연산량은 각 참여자가 공개하지 않는 값의 개수에 비례하므로 제안 프로토콜에서의 검증 연산량이 줄어든다.

3.3 프로토콜의 안전성 증명

본 절에서는 프로토콜의 안전성에 대하여 논의한다. 증명은 영지식 증명 분야 및 적응적 선택 암호문 공격(Adaptive Chosen Ciphertext Attack)에 안전한 공개키 암호 시스템 연구에서 많이 사용하는 시뮬레이터를 이용한 접근 방법을 사용한다. 증명의 세부 방법은 이전의 논문 방법을 참조하였다[1, 8-11].

이후 세부 절에서는 제안 프로토콜에 대하여 본 세부절에서 설명한 방법으로 증명을 수행한다. 각 프로토콜을 이루는 개인값 및 공개된 입력값, 결과값을 설정하고 프로토콜에 적합한 시뮬레이터를 설계하는 방식으로 증

제안 공유 갱신 프로토콜

* 각 참여자 i의 입력값 $(i=1,\ldots,n')$

이전 프로토콜의 결과값 $\{\delta_{ij}\}_{j=1,\ldots,n'}$, $\{\lambda_{ij}\}_{j=1,2,\ldots,n'}$, $\{\{\gamma_{mj}\}_{j=1,\ldots,k+1}\}_{m=1,\ldots,n'}$, 공유 s_i

* 공개된 입력값

이전 프로토콜에서 사용한 검증값 $\{\{g^{a_{ij}}h^{\beta_{ij}}\}_{j=1,\ldots,n'}\}_{i=1,\ldots,n'}$ 및 $ID_i\ (i=1,..,n')$, p, q, g, h

* 각 참여자 i의 결과값 $(i=1,\ldots,n')$: 갱신된 새로운 공유 s_i'

* 수행과정

1. 각 참여자는 임의로 $y_i, d_i, e_i(\in Z_p^*)$를 생성한다. $(i=1,\ldots,n')$

2. 참여자 i는 y_i+d_i 값을 동보한다.

3. 각 참여자에 대한 다항식 $f_i(x)$을 생성한다. $(i=1,\ldots,n')$

$$f_i(x)=c_{ik-1}x^{k-1}+c_{ik-2}x^{k-2}+\ldots+c_{i1}x$$

$$((c_{ik-1}|c_{ik-2}|\ldots.|c_{i1})=HashExpansion(ID_j,k-1))$$

4.참여자 i는 다른 참여자로부터 받은 $y_j+d_j\ (j=1,2,..,i-1,i+1,..,n')$ 및 자신이 생성한 y_i, d_i를 이용하여 다음과 같은 A_i값을 만든다.

$$A_i=\Sigma_{m=1}^{n'}(y_m+d_m)f_m(ID_i)$$

5. 참여자 i는 $d_jf_i(ID_j)+\delta_{ij}(=B_{ij})$및 $e_jf_i(ID_j)+\lambda_{ij}(=C_{ij})$를 생성하여 참여자 j에게 송신한다. $(j=1,\ldots,n')$

6. 참여자 i는 5.에서 보낸 값들의 검증을 위한 검증값 $g^{d_i}h^{e_i}(=V_i)$를 동보한다.

7. 참여자 i는 다른 참여자들이 동보한 값들을 이용 5에서 받은 값들을 검증한다. 검증 방법은 아래의 수식에 대한 만족 여부로 결정된다.

$$g^{B_{ji}}h^{C_{ji}}=?\ V_j^{f_i(ID_j)}\times\Pi_{m=1}^{k}g^{\alpha_{\gamma_{jm}j}}h^{\beta_{\gamma_{jm}j}}\ (j=1,2,..,i-1,i+1,..,n')$$

8. 만약 7에서 검증한 값이 수식의 등식을 만족하지 않는 경우, 검증이 맞지 않은 값을 보낸 참여자 j가 잘못된 값을 보냈다는 선언을 동보한다. 이 때 자신이 5에서 j에게 받은 값을 동보한다.

9. 8에서 선언을 받은 참여자 j는 자신의 비밀값 $y_j, d_j, e_j(\in Z_p^*)$ 및 확장 분산 난수 분배 프로토콜의 결과중 δ_{ji} , λ_{ji}를 동보한다.

10. 다른 모든 참여자는 7~9에서 받은 값을 이용하여 j가 보낸 값을 검증한다. 만약 j가 잘못된 값을 보낸 것으로 판명되면 참여자 j를 프로토콜에서 제거하며 그렇지 않은 경우 참여자 i를 프로토콜에서 제거한다. 또한 모든 참여자는 자신이 공격자에게 분배한 값을 공개하며, 공격자로부터 받은 값도 모두 공개한다.

11. 참여자 i는 7~10의 검증과정에서 A_i에서 제거된 참여자의 값을 제거한다. 이 때 제거된 참여자의 집합을 $DISQUAL$ 이라 하면, 제거한 결과값 A'은 아래와 같이 계산된다.

$$A'_i=A_i-\Sigma_{j\in DISQUAL}(y_j+d_j)f_j(ID_i)$$

12. 참여자 i는 위의 검증 작업을 통과한 참여자를 다시 순서를 지어서 $(1,\ldots,n'')$이라 하면, 각 참여자 i는 11에서 생성한 A'에서 5에서 받은 값 및 공격자와 관련된 값을 빼어주어 공유 갱신에 사용되는 값 D_i를 계산한다.

$$D_i=A'_i-\Sigma_{j=1}^{n''}(B_{ji}-\Sigma_{\gamma_{jm}\in DISQUAL}\alpha_{\gamma_{jm}j}+\Sigma_{\gamma_{jm}=j}(\Sigma_{l\in DISQUAL}\alpha_{jl}))(m=1,\ldots,k\ ,\ l=1,..,n')$$

13. 각 참여자 i는 기존의 공유 s_i에 D_i를 더하여 새로운 공유 s'_i를 생성한 후 s_i를 삭제한다.

그림 2 제안 공유 갱신 프로토콜

명을 수행한다. 본 논문에서 제안된 프로토콜을 다수가 참여하는 프로토콜이므로 시뮬레이터 설계시 공격자의 입력값을 제외한 다른 정직한 참여자의 입력값 모두를 시뮬레이터가 생성한다. 또한 새로운 개인값을 만드는 과정에서 시뮬레이터는 정직한 참여자의 모든 값을 알 수 있다.

3.3.1 확장 분산 난수 분배 프로토콜의 시뮬레이터

본 절에서는 확장 분산 난수 분배 프로토콜의 안전성에 대하여 논의한다. 확장 분산 난수 분배 프로토콜에 대한 시뮬레이터를 생성한 후 그것의 안전성을 증명한다. 본 시뮬레이터의 증명의 의미는 확장 분산 난수 분배 프로토콜의 결과값이 각 참여자가 갖고 있는 난수 생성기에 의존한다는 것을 의미한다. 또한 공격자들이 모든 참여자들이 공개하는 검증 값을 이용하여 정직한 참여자들이 생성한 개인값을 추측해 낼 수 없다는 것을 증명한 것이다. 확장 분산 난수 분배 프로토콜의 시뮬레이터의 구체적인 설계 과정은 그림 3과 같다.

그림 3에서 시뮬레이터의 증명의 의미는 확장 분산 난수 분배 프로토콜의 결과값은 각 참여자가 갖고 있는 난수 생성기에 의존한다는 것을 의미한다. 또한 공격자들이 모든 참여자가 공개하는 검증값을 이용하여 정직한 참여자들의 개인값을 추측해 낼 수 없다는 것을 증

확장 분산 난수 분배 프로토콜의 시뮬레이터

안전성 증명의 대상

정직한 참여자 i의 프로토콜의 결과값 δ_{ij} $(j=1,\ldots,n)$

수행과정

1. 공격자들과 시뮬레이터는 정상적인 확장 분산 난수 분배 프로토콜을 수행한다. 이 때 시뮬레이터는 정직한 참여자 모두의 행동을 수행한다.

2. 공격자들은 자신들의 프로토콜의 결과값과 각 참여자들이 보낸 검증값을 Decryption Oracle에 보내어 정상적인 참여자 i가 생성한 값인 δ_{ij} , λ_{ij} 를 추측해 낸다. 이 추측값을 $\overline{\delta_{ij}}$, $\overline{\lambda_{ij}}$ 라 한다.

3. 공격자가 보낸 추측값을 받은 시뮬레이터는 $Dlog_g h = c$ 를 계산한다.

4. 프로토콜에서 생성되는 δ_{ij} , λ_{ij} 값은 $k+1$명의 참여자의 α_{mi} , β_{mi} $(m \in \{1,2,..,n'\})$ 으로 이루어져 있다. 프로토콜의 공격자는 최대 $k-1$명이므로 δ_{ij} , λ_{ij}값은 최소 2명 이상의 정직한 참여자의 α_{mi} , β_{mi}값으로 이루어져 있다는 것을 알 수 있다. 이 정직한 참여자의 값을 α_{honest} , β_{honest}라 하자. 이 값은 정직한 참여자가 다른 정직한 참여자에게 주는 값이므로 공격자가 알 수 없는 값이다. 따라서 시뮬레이터가 임의로 생성할 수 있는 값이다.

5. 시뮬레이터가 생성한 새로운 α_{honest} , β_{honest}값을 $\alpha_{sim-honest}$, $\beta_{sim-honest}$라 하고, 이전 단계에서 공격자가 추측한 값을 $\overline{\alpha}$, $\overline{\beta}$라 하자. 그러면 $\alpha_{sim-honest}$, $\beta_{sim-honest}$값은 다음과 같은 수식을 만족하는 임의의 값으로 생성할 수 있다.

$$\alpha_{sim-honest} + c*\beta_{sim-honest} = \overline{\alpha} + c*\overline{\beta}$$

6. $\beta_{sim-honest}$에 맞추어 t_l값을 변화시킨다. 이 때의 t_l값은 프로토콜의 핵심 결과값 δ_{ij}값과는 관계 없는 난수 생성값(Random Associator)이므로 변화시킬 수 있는 값이다.

7. 기존에 생성되었던 공격자의 입력값과 새로 시뮬레이터가 생성한 값은 모두 프로토콜의 검증 과정을 통과한다. 그러나 시뮬레이터가 생성한 $\alpha_{sim-honest}$, $\beta_{sim-honest}$값을 이용하여 생성된 δ_{ij} , λ_{ij} 값은 공격자가 추측한 값과 전혀 다른 값을 갖게 된다.

그림 3 확장 분산 난수 분배 프로토콜의 시뮬레이터

명한 것이다.

3.3.2 제안 공유 갱신 프로토콜의 시뮬레이터

본 세부 절에서는 3.2절에서 제안한 공유 갱신 프로토콜의 시뮬레이터를 설계하여 제안 공유 갱신 프로토콜의 안전성을 증명한다. 제안 공유 갱신 프로토콜에서 각 참여자 i가 갖는 개인값은 y_i이다. 프로토콜 상에서 각 참여자는 $y_i + d_i$값을 공개하므로 이 값은 고정된다. 따라서 y_i값을 결정시키는 값인 d_i값에 대한 안전성 증명이 전체 프로토콜의 안전성 증명의 핵심이 된다. 전체의 증명은 각 참여자가 새로운 공유 갱신 값을 프로토콜의 결과값으로 하여 공유 갱신 값에 포함되어 있는 각 y_i값의 구별 불가능성(indistinguishability)를 증명한다. 증명 과정은 그림 4와 같다.

4. 복잡도 분석과 성능 측정

본 장에서는 제안 프로토콜과 이전 프로토콜의 연산량과 망 전송량에 대한 복잡도 분석 및 각 프로토콜을 실제로 구현한 측정 속도에 대하여 논의한다. 연산량 측정시 모든 참여자가 프로토콜에 적법한 수행을 했을 경우를 가정한다.

4.1 복잡도 분석

본 절에서는 프로토콜의 연산량 및 망 사용량에 대한 복잡도 분석을 수행한다. 연산량에 대한 분석은 프로토콜에서 가장 많은 연산 시간을 차지하는 모듈라 멱승의 수행 횟수에 관하여 분석을 하고, 망 사용량은 각 참여자의 관점에서 망으로 전송하는 정보의 양을 분석한다.

4.1.1 연산량 분석

제안 공유 갱신 프로토콜의 수행을 위해서는 확장 분산 난수 분배 프로토콜의 수행은 필수적이다. 따라서 연산량 분석은 Jarecki가 제안한 공유 갱신 프로토콜의 연

제안 공유 갱신 프로토콜의 시뮬레이터

수행 과정

1. 공격자와 시뮬레이터는 정상적인 공유 갱신 프로토콜의 과정을 수행한다.
2. 공격자는 자신에게 주어진 프로토콜의 결과값 및 공개된 검증값을 Decryption Oracle에게 보내게 된다. Decryption Oracle은 이 값을 근거로 임의의 참여자 i의 개인값 y_i및 개인값을 숨기기위한 은닉값인 e_i, d_i를 추측해 낸다. 이 추측값을 $\tilde{y}_i$, $\tilde{d}_i$, $\tilde{e}_i$라 한다.
3. 시뮬레이터는 $Dlog_g h = c$ 값을 계산한다.
4. 시뮬레이터는 위에서 계산한 c값을 이용하여 다음 식을 만족하는 d_{sim} , e_{sim} , y_{sim}을 생성한다.

$$\tilde{d}_i + c * \tilde{e}_i = d_{sim} + c * e_{sim}$$

5. 확장 분산 난수 분배 프로토콜에서 정직한 참여자의 값들의 변화 만으로 임의의 δ_{ij} ,λ_{ij} $(i,j=1,\ldots,n')$값을 변화 시킬 수 있다는 것을 증명했으므로 시뮬레이터는 다음과 같은 성질을 만족하는 δ_{ij}^{sim} ,λ_{ij}^{sim}을 생성한다. $(i,j=1,\ldots,n')$

$$\tilde{d}_i f_i(ID_j) + \delta_{ij} = d_{sim} f_i(ID_j) + \delta_{ij}^{sim}$$

$$\tilde{e}_i f_i(ID_j) + \lambda_{ij} = e_{sim} f_i(ID_j) + \lambda_{ij}^{sim}$$

6. 위의 과정에서 생기는 δ_{ij}^{sim}의 값이 변화하므로 이것을 이전에 수행되었던 프로토콜과 공격자의 관점에서 값의 변화가 없도록 일관성을 맞추어 주어야 한다. 공격자의 관점에서 보았을 때 위와 같은 조건식을 $(n-k)(k+1)$개 받는데 비하여 이 식에 참여하는 정직한 참여자의 변수는 $(n-k)(n-k+1)$개 이상이 된다. 이 때 가정에 의해 $k < \frac{1}{2}n - 1$이므로 공격자의 관점을 변화가 없게 하는 것은 충분하다.
7. 결국 공격자의 관점을 변화시키지 않고 참여자 i의 값 y_i, d_i는 시뮬레이터의 생성값 d_{sim}의 확률 분포에 따라 결정된다. 이것은 공격자에게 노출되는 값이 어떠한 값이라도 실제 개인값 d_i는 임의의 값이 될 수 있다는 것이다. 즉, 공격자는 자신이 알 수 있는 정보만으로 임의의 참여자 i의 개인값 d_i를 구별할 수 없다.

그림 4 제안 공유 갱신 프로토콜의 시뮬레이터

산량과 본 논문에서 제안한 프로토콜의 연산량의 합을 비교한다. 연산량의 비교는 각 프로토콜에서 수행하는 모듈라 멱승의 수행량을 분석함으로써 수행한다. 프로토콜에서 각 참여자가 수행하는 연산중 모듈라 멱승을 제외한 다른 연산량은 모듈라 멱승 연산량에 비해 매우 작은 연산량을 필요로 하는 연산이며 나머지 연산은 참여자의 수및 임계값에 관계없는 상수번의 연산이기 때문이다. 각 프로토콜에서 수행하는 모듈라 멱승의 양을 단계별로 정리하면 아래의 표 1과 같다.

표 1에서 알 수 있듯이 Jarecki의 이전 프로토콜은 $O(kn)$인 것에 비하여 제안 프로토콜은 $O(n)$의 연산량을 갖게 되며, 다른 연산량을 무시할 수 경우 k값이 7 이상의 모든 경우에 제안 프로토콜이 이전 프로토콜보다 적은 수행시간을 갖게 된다.

4.1.2 망 사용량 분석

본 세부 절에서는 이전 프로토콜과 제안 프로토콜의 망 사용량을 비교한다. 소수 p와 q의 비트수의 차이가 무시할 수 있을 정도로 작다고 가정한다. 따라서 모든 연산의 결과의 비트 길이를 log$p$비트라 한다. 또한 동보할 경우의 망 전송량은 각각의 수신자에게 1번씩 전송한 것으로 가정한다. 마지막으로 프로토콜의 참여자 n명은 모두 적법한 수행을 한다고 가정한다.

Jarecki의 프로토콜을 살펴보면 우선 2번 수행과정에서 다른 모든 참여자에게 $\alpha_{ij}=f_i(ID_j)$, $\widetilde{\alpha_{ij}}=f_i(\widetilde{ID_j})$값을 송신하므로 $2(n-1)\log p$비트를 전송한다. 또한 검증값 $C_{im}(m=1,\cdots,k-1)$값을 모든 참여자에게 동보하므로 $n(k-1)\log p$비트의 전송을 수행한다. 따라서 최종적으로 $(nk+n-2)\log p$ 비트 만큼의 정보를 다른 참여자에게 보내게 된다.

이에 비해 제안 프로토콜은 확장 분산 난수 분배 프로토콜의 2번 과정에서 다른 모든 참여자에게 2개의 값을 전송하고, 또한 4번 과정에서 n개의 값을 동보하므로 $(n+2)(k-1)\log p$비트를 전송하게 되고, 또한 제안 공유 갱신 프로토콜에서는 2번 과정에서 1개의 값을 동보하고, 5번 과정에서 두 개의 값을 자신을 제외한 다른 모든 사용자에게 전송하며, 마지막으로 6번 과정에서 1개의 값을 동보하므로 $4(n-1)\log p$비트의 정보를 다른 참여자에게 전송하게 된다.

따라서 log$p$값을 생각할 경우 이전 프로토콜은 각 참여자마다 $O(nk)$의 정보 전송량을 갖는데 비하여, 제안 프로토콜은 $O(n^2)$의 정보를 전송하게 된다. 일반적으로 k값이 $O(n)$값을 갖게 되므로 망 사용량은 제안 프로토콜과 이전 프로토콜이 비슷한 수준임을 알 수 있다.

4.2 성능 측정

본 절에서는 이전 프로토콜과 제안 프로토콜의 수행시간을 직접 측정한 결과를 분석한다. 본 성능 측정의 목적은 전체 프로토콜의 수행시간의 측정을 통하여 전체 프로토콜의 수행시간 중 모듈라 멱승이 차지하는 비율을 살펴봄으로써 모듈라 멱승 연산량을 줄이는 것의 전체 프로토콜 수행시간 단축에 대한 공헌의 정도를 알아보는 것에 있다.

측정 대상은 각 참여자의 계산 시간이었으며 n과 k의 변화에 따른 각 참여자의 수행 시간을 측정하였다. 망 송수신으로 인한 수행시간은 0으로 가정하였다. 또한

표 1 연산량 비교

		단계	연산 수식	모듈라 멱승 연산량	연산 회수	합계
Jarecki의 공유 갱신 프로토콜		2	$C_{im}=g^{c_{im}}h^{c_{im}}$	2	$k-1$	$2(k-1)$
		3	$F_i(x)=\Pi_{m=1}^{k-1}(C_{im})^{x^m}$	$k-1$	$n-1$	$(k-1)(n-1)$
		3	$g^{\alpha_{ij}}h^{\alpha_{ij}}=?F_i(ID_j)$	2	$n-1$	$2(n-1)$
		전체				$(k+1)(n+1)-4$
제안 프로토콜	확장 분산 난수 분배 프로토콜	4	$g^{\alpha_{ij}}h^{\beta_{ij}}$	2	n	$2n$
		4	h^{t_i}	1	1	1
		5	$(g^{\alpha_{mi}}h^{\beta_{mi}})=?(g)^{\alpha_{mi}}\times(h)^{\beta_{mi}}$	2	$n-1$	$2(n-1)$
	제안 공유 갱신 프로토콜	6	$g^{d_i}h^{e_i}(=V_i)$	1	1	1
		7	$g^{B_{ij}}h^{C_{ij}}=?V_j^{f_i(ID_j)}\times\Pi_{m=1}^{k}g^{\alpha_{i,m-1}}h^{\beta_{i,m-1}}$	3	$n-1$	$3(n-1)$
		전체				$7n-3$

Pentium III 450 Mhz 컴퓨터의 Windows 2000 환경에서 각각의 프로토콜을 수행하였다. 또한 소수 p, q는 1024비트 길이의 소수를 사용하였고, 실행 환경에서의 1024비트 모듈라 멱승 1회의 수행시간은 약 135 ms이었다.

우선 그림 5의 경우는 $k=\frac{1}{3}n$의 경우의 계산 시간을 측정한 것이다. $k=\frac{1}{3}n$의 경우이므로 이전 프로토콜의 복잡도는 약 $\frac{1}{3}n^2$이고, 제안 프로토콜은 약 $7n$이 된다.

그림 5의 결과에서 k가 2에서 20으로 10배가 늘어날 경우 이전 프로토콜은 수행시간이 약 70배 늘어났으며 제안 프로토콜은 약 10배가 늘어났으므로 실험 결과는 복잡도 분석의 결과와 일치하며 프로토콜의 계산시간의 거의 대부분을 모듈라 멱승이 차지한다는 것을 알 수 있다.

그림 6은 n이 고정되고 k만 변화할 경우의 수행시간을 조사한 결과이다. 복잡도 분석에 의하면 제안 프로토콜은 k에 독립적으로 수행시간의 변화가 없어야 하며, 이전 프로토콜은 k에 비례하여야 한다. 그림 6의 수행시간의 결과는 이와 정확히 일치한다. 이전 프로토콜은 k 값의 증가에 따라 선형적으로 수행시간이 증가하며, 제안 프로토콜은 k값의 변화에 따른 수행시간의 증가가 없음을 알 수 있다.

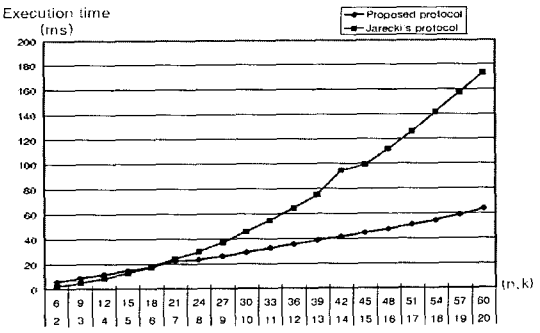

그림 5 수행 시간 I ($n = 3k$)

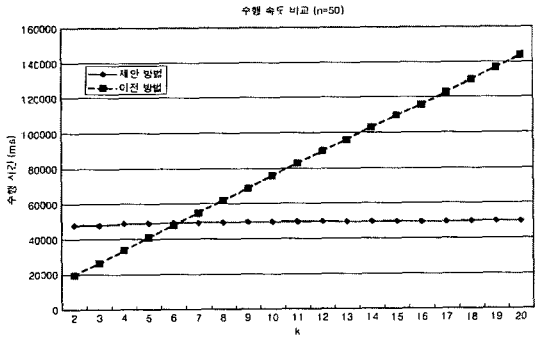

그림 6 수행시간 II ($n = 50$)

결론적으로 전체 프로토콜의 연산량은 모듈라 멱승에 대한 연산 시간이 거의 대부분을 차지하며 전체 프로토콜의 수행시간은 각 프로토콜의 모듈라 멱승의 복잡도 분석의 결과와 일치한다.

5. 결 론

비밀 분산은 집중되어 있는 정보를 다수에게 분산시켜 정보를 보호할 수 있는 암호학의 한 분야이다. 이러한 비밀 분산을 이용하여 임계 암호시스템을 생성할 수 있으며 임계 암호시스템은 키를 다수개의 서버로 분산시키기 때문에 키를 알아내기 위해서는 많은 서버를 공격해야 한다. 따라서 임계 암호시스템은 안전성을 증가시킬 수 있는 분산 인증 서버 등 사용될 수 있으며 또한 프로토콜의 각 참여자의 이해가 엇갈리기 때문에 프로토콜의 각 참여자들의 악의적 행위에도 프로토콜이 적법하게 수행되며, 또한 악의적인 참여자를 가려내야 하는 전자 투표 프로토콜 등에 사용될 수 있다.

이러한 비밀 분산의 분야 중 비밀 분산의 안전성을 높이는 방법으로 능동적인 비밀 분산이 제안되었다. 능동적인 비밀 분산은 공격자가 임계값 이하의 공유를 습득하였을 때 이러한 공유를 무효화하기 위하여 제안된 방법이다. 능동적인 비밀 분산 방법의 핵심은 주기적으로 공유를 갱신하는 것이다. 주기적으로 공유를 갱신하면 공격자는 이전까지 습득한 공유에 관한 정보가 무효화되므로, 공격자는 하나의 공유 갱신 주기 이전에 임계값 이상의 공유를 모아야만 공격이 성공하게 된다.

본 논문에서는 (k, n) threshold scheme을 이용한 능동적 비밀 분산 방법에서 공유 갱신을 빠르게 수행할 수 있는 프로토콜을 제안하였다. 이전에 제안된 (k, n) threshold scheme에 대하여 각 참여자가 k개의 개인값을 갖는 프로토콜을 각 참여자가 단지 1개의 개인값을 갖는 프로토콜로 개선하여 각 참여자가 프로토콜 수행시 계산하는 모듈라 멱승의 수에 대한 복잡도를 $O(n^2)$에서 $O(n)$으로 낮추었다. 또한 제안 프로토콜을 Decryption Oracle을 가정한 시뮬레이션 방법을 이용하여 임계값 이하의 공격자에 대하여 제안 프로토콜의 안전함을 증명하였다.

또한 $k=\frac{1}{3}n$ 일 경우의 실제 프로토콜을 구현하여 참여자가 10배로 증가할 경우, 이전 프로토콜은 수행 시

간이 70배로 증가하는데 비하여 제안 프로토콜은 수행 시간이 10배로 증가하는 것을 확인할 수 있었다. 이것은 각 프로토콜의 모듈라 멱승 연산량에 대한 복잡도 분석의 결과와 일치하며 이것으로 제안 프로토콜 및 이전 프로토콜의 연산량의 대부분이 모듈라 멱승의 연산량이며 나머지 연산량은 전체 프로토콜의 수행 속도에 거의 영향을 미치지 않음을 알 수 있었다.

참 고 문 헌

[1] S. Jarecki, "Efficient Threshold Cryptography", Ph.D Thesis in MIT,2001.

[2] A. Shamir, "How to Share a Secret", vol.22, pp. 612-613, Communications of The ACM, 1979.

[3] B. Chor, S. Goldwasser, S. Micalli and B. Awerbuch, "Verifiable secret sharing and achieving simultaneous broadcast", pp.335-344, Proc. of IEEE Fund. of Comp. Sci., 1985.

[4] P. Feldman, "A Practical Scheme for Non-interactive Verifiable Secret Sharing", pp.427-437, Proc. 28th IEEE Symp. on Foundations of Computer Science, Los Angeles, 1987.

[5] T.Pederson, "Non-interactive and information-theoretic secure verifiable secret sharing", pp.129-140, CRYPTO'91-LNCS, 1991.

[6] T.Pederson, "A threshold cryptosystem without a trusted third party", pp.522-526, EuroCrypt'91 - LNCS 1991.

[7] H.K.A.Herzberg, S.Jarecki and M.Yung, "Proactive Secret Sharing Or: How to Cope With Perpetual Leakage", CRYPTO'95 - LNCS, 1995.

[8] S.Goldwasser, S.Micali and C.Rackoff, "The knowledge complexity of interactive proof-system", pp.365-377, STOC. 85, 1985.

[9] R.Canetti, "Adaptive Security for Threshold Cryptosystems", pp.98-116, CRYPTO'99 - LNCS, 1999.

[10] R.Gennero, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems", pp.295-310, Eurocrypt'99 - LNCS, 1999.

[11] S.Jarecki and A.Lysyanskaya, "Adaptively secure threshold cryptography:Introducing concurrency, removing erasures", Eurocrypt'2000 - LNCS, 2000.

[12] R.Ostrovsky and M.Yung, "How to withstand mobile virus attacks", pp.51-61, In Proc. 10th ACM Symp. on Principles of Distributed Computation, 1991.

이 윤 호

2000년 2월 한국과학기술원 전산학과 학사. 2002년 2월 한국과학기술원 전자전산학과 석사. 2002년 3월~현재 한국과학기술원 전자전산학과 박사 과정 재학중. 관심분야는 암호학, 네트워크 보안

김 희 열

2000년 2월 한국과학기술원 전산학과 학사. 2002년 2월 한국과학기술원 전자전산학과 석사. 2002년 3월~현재 한국과학기술원 전자전산학과 박사 과정 재학중. 관심분야는 암호학, 네트워크 보안

정 병 천

1998년 2월 성균관대 정보공학과 졸업 2001년 2월 한국과학기술원 전자전산학과 석사. 2001년 3월~현재 한국과학기술원 전자전산학과 박사과정 재학중. 관심분야는 정보보호, 네트워크 보안

이 재 원

1997년 2월 KAIST 전산학과 학사 1999년 8월 KAIST 전자전산학과 전산학전공 석사. 1999년 9월~현재 KAIST 전자전산학과 전산학전공 박사과정. 관심분야는 타원곡선 암호, 암호 프토로콜, S/W 보호

윤 현 수

1979년 서울대학교 전자공학과 학사. 1981년 한국과학기술원 전산학과 석사. 1981년~1984년 삼성전자 연구원. 1988년 오하이오 주립대학 전산학 박사. 1988년~1989년 AT&T Bell Labs. 연구원. 1989년~현재 한국과학기술원 전산학과 교수 관심분야는 Adhoc망, 네트워크 보안, 암호학, 상호연결 네트워크