情 報 保 護 學 會 誌
第 13 券 第 5 號, 2003. 10

키 생명주기에 따른 단계별 안전성 요구사항 분석

한 종 수*, 안 기 범*, 곽 진*, 양 형 규**, 원 동 호***

요 약

최근 인터넷을 통한 전자상거래와 금융 서비스가 널리 확산됨에 따라 인터넷 상에서 전송되는 정보의 안전성과 신뢰성을 확보하기 위해 필요한 암호기술이 크게 주목을 받고 있다. 이러한 암호기술은 크게 보호하려는 정보와 관련된 키를 관리하는 것으로 나눌 수 있는데, 키에 대해 정당한 사용자만이 접근하여 이를 사용하고 제 3자로부터의 키의 노출 및 손상 등을 방지하기 위한 기술인 키 관리가 매우 중요한 과제로 부각되고 있다. 하지만 키 관리에 대한 관심 및 이용이 증가하고 있는 반면, 이에 대한 안전성 분석과 활용이 미흡한 실정이다. 따라서 본 고에서는 키 관리의 키 생명주기에 따라 안전성을 분석하고 이에 따른 각 단계별 안전성 요구사항을 분석하고자 한다.

Ⅰ. 서 론

최근 유·무선 통신 기술의 발전으로 이를 이용한 뱅킹(Banking) 서비스, 주식 거래, 전자상거래 등 다양한 서비스가 가능하게 되었으며, 또한 일상생활의 유익성과 편리함을 제공하고 있다. 하지만 컴퓨터 해킹, 바이러스, 개인 정보 누출 등의 각종 정보화 역기능으로 인한 새로운 사회적 문제가 야기되고 있다.

따라서, 이러한 정보화 역기능을 예방하고 서비스들을 안전하게 제공하기 위해 정보 시스템 내에서의 정보보호와 통신 상태의 정보보호 및 사용자 인증을 위한 암호기술의 필요성이 증대되었다. 특히, 정보화 사회가 발전함에 따라 제 3자로부터 보호해야할 정보의 급증으로 정보보호를 위한 암호기술의 사용이 요구되고 있다. 이러한 암호기술은 근본적으로 보호하려는 정보와 관련된 키를 관리하는 것으로 축약될 수 있다.

키에 대해 정당한 사용자만이 접근하여 이를 이용하고 제 3자로부터의 키의 노출, 손상 등을 방지하기 위한 기술로 키 관리(Key management)가 매우 중요한 과제로 주목받고 있다. 키 관리 기술은 키 생명주기(Key lifecycle)에 걸쳐 키에 대한 정보를 유지, 관리 등의 모든 절차를 포함한다. 즉, 키 관련 데이터를 생성하는 키 생성, 생성된 키를 제 3자에게 노출시키지 않고 두 객체 사이에 비밀리에 키를 공유하는 키 분배, 키 관리 프로토콜을 수행하는 과정에서 생성되는 키와 키 관련 데이터를 안전하게 보관하기 위한 키 저장, 키 관련 정보의 노출이나 유효기간의 만료 등으로 인한 키 폐기와 합법적인 사용자의 키 분실이나 손상으로 키를 복구하는 경우 수행하는 키 복구 과정으로 구성된다[1~5].

이를 바탕으로 키 관리 서비스의 제공자, 기술 개발자 및 사용자의 수가 증가함에 따라 키 관리 서비스의 이용이 증가하고 있지만, 키 생명주기에 대한 안전성 요구사항을 분석하고 이를 기반으로 하는 연구와 서비스는 미흡한 실정이다.

따라서 본 고에서는 키 생명주기에 따라 가능한 공격모델을 분석하고 이에 따른 각 단계별 안전성 요구사항을 분석하고자 한다.

본 고의 구성은 다음과 같다. 먼저 제 2장에서는 키 생명주기에 대하여 살펴보고, 제 3장에서는 키 생명주기를 단계별로 분류하여 안전성 요구사항을 분석한다. 제 4장에서는 키 생명주기에서 안전한 키 관리를 제공할 수 있는 요구사항을 도출하고, 마지막으로 5장에서 결론을 맺는다.

* 성균관대학교 정보통신공학부 정보통신보호연구실({jshan, gbahn, jkwak}@dosan.skku.ac.kr)
** 강남대학교 컴퓨터미디어 공학부 부교수(hkyang@kns.kangnam.ac.kr)
*** 성균관대학교 정보통신공학부 정교수(dhwon@dosan.skku.ac.kr)

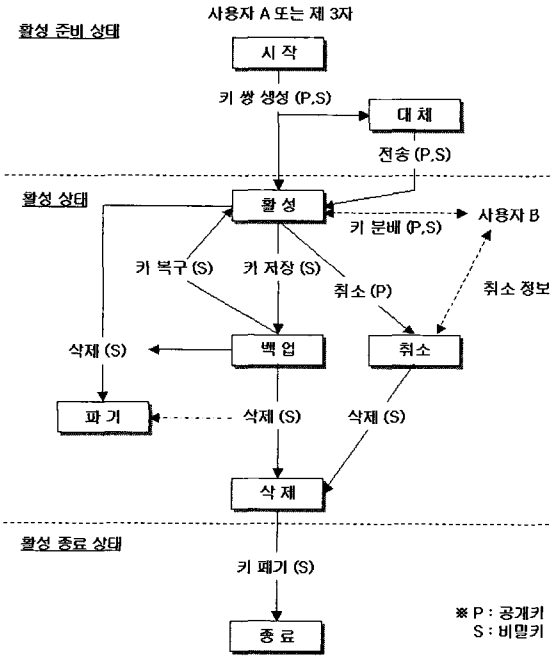
Ⅱ. 키 생명주기

키의 생명주기는 크게 세 가지 단계로 구분할 수 있다. 키가 암호화나 복호화 같은 작업을 수행할 수 없는 활성 준비 상태(pending active)와 키를 사용하여 암호학적으로 처리할 수 있는 활성 상태(active), 그리고 키가 단지 복호화나 검증을 위해서만 사용되는 활성 종료 상태(post active)로 구성된다[6][8].

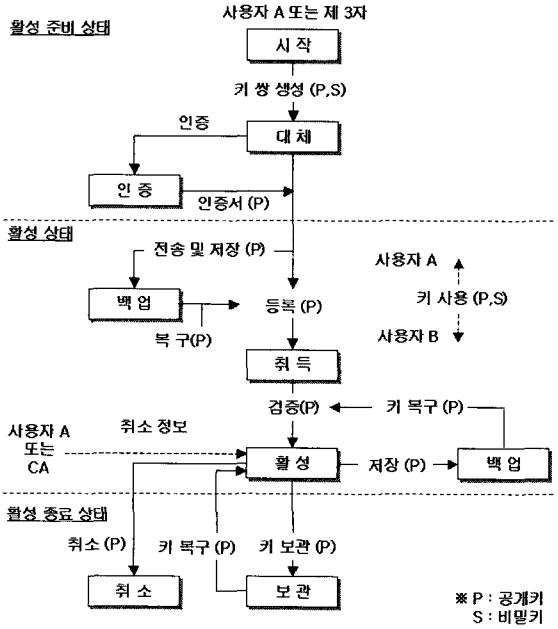
활성 준비 상태는 키를 생성하는 단계와 키에 문제가 있을 경우 또는 새로운 키를 생성하여 기존의 키를 대체하는 단계가 포함된다. 특히 공개키 암호 방식에서는 공개키와 정당한 사용자를 연관시키는 인증 단계가 추가된다. 활성 상태에는 키를 암호학적으로 사용하는 모든 단계가 구성 요소로 포함된다. 즉, 키를 사용하여 임의의 메시지를 암호화하거나 복호화를 수행하는 단계, 서명을 수행하고 검증하는 단계, 그리고 키를 다른 사용자와 분배하는 단계 등이 여기에 속하게 된다. 또한 키를 안전하게 저장하는 단계와 키의 분실이나 특정 이유로 인해 키를 복구하는 과정 등도 키 활성 상태에 포함된다. 활성 종료 상태에는 키가 더 이상 사용되지 않도록 조치하는 키 폐기 단계를 의미한다. 특히 공개키의 경우에는 폐기된 이후에도 폐기된 시점 이전에 암호화나 서명이 된 경우의 복호화 및 검증을 위하여 키를 보관하는 단계가 추가된다[7][9]. 다음 [그림 1], [그림 2]는 비밀키와 공개키의 키 생명주기를 도식화한 것이다.

[그림 1], [그림 2]에서 보는 바와 같이 키의 생명주기는 크게 키 생성, 키 분배, 키 저장, 키 폐기, 키 복구로 나눌 수 있고 각 단계의 특징은 다음과 같다[1][4].

- 키 생성 : 키 생성은 암호학적으로 안전한 키를 생성하는 절차를 의미하며 생성된 키는 예측이 불가능하고 위조할 수 없는 랜덤 수(random number)를 사용해야 하며, 재사용해서는 안 된다.
- 키 분배 : 키 분배는 인가된 객체들 사이에 키 또는 키 재료(key material)가 안전하게 공유되는 것을 의미한다. 공개키 암호 방식에서는 키를 분배하는 특정 메커니즘을 사용하고, 대칭 암호 방식에서는 KTC(Key Translation Center), KDC (Key Distribution Cen- ter)에 의해 분배가 이루어진다.
- 키 저장 : 키 저장은 키를 사용하거나 복구를 위한 백업(backup)을 위해 키를 안전하게 저장하는 것을 의미한다. 이 때, 키는 물리적으로 안전한 장치에 저장하는 것이 바람직하며, 키 저장은 키 생명주기 동안의 모든 상태(활성 준비, 활성, 활성 종료)에서 발생할 수 있다.
- 키 폐기 : 키 폐기는 더 이상 사용될 필요가 없는키의 안전한 폐기를 위해 이루어진다. 키 폐기 과정은 키와 관련된 모든 기록을 제거함으로써 폐기 후에



[그림 1] 비밀키의 키 생명주기



[그림 2] 공개키의 키 생명주기

남아있는 어떤 정보를 가지고도 폐기된 키를 다시 복구할 수 없도록 해야 한다.

· 키 복구 : 키 복구는 정부기관 또는 허가된 사용자 등이 합법적인 수단을 통해 암호문을 복호화 하거나, 사용자가 자신의 비밀키를 분실하였을 경우와 같은 유사시에 정당한 사용자만이 키를 복구할 수 있는 기능을 제공하기 위해 이루어진다.

Ⅲ. 키 생명주기 단계별 분석

본 장에서는 키의 생명주기를 키 생성, 키 분배, 키 저장, 키 폐기, 키 복구로 분류하고, 각각의 단계에서 가능한 공격 모델을 살펴본다. 그리고 분석의 결과를 토대로 각 단계에서 만족해야 하는 안전성 요구사항을 도출한다.

1. 키 생성

암호 시스템의 안전성 및 신뢰성은 키의 안전성에 기반을 두기 때문에, 암호 시스템 설계 및 구현 시 키를 안전하게 생성하는 것은 매우 중요한 일이다. 키 생성은 암호학적인 안전성을 만족하는 키를 생성하는 절차를 의미하며, 키를 생성하기 위해서는 지금까지 알려진 여러 가지 공격 방법들에 대해 최상의 안전성을 확보할 수 있는 파라미터들을 사용한다.

본 절에서는 키 생성 단계의 안전성 요구사항을 분석하기 위해 암호 시스템에 따라 대칭키 암호 방식과 공개키 암호 방식의 키 생성으로 구분하고, 공개키 암호 방식에서의 이산대수 문제(discrete logarithm problem)와 소인수분해 문제(facto-rization problem)를 푸는 알고리즘을 살펴본다. 그리고 키 생성에 사용되는 파라미터들의 유형에 따라 가능한 공격 방법들을 살펴보고 파라미터의 안전성에 대해 살펴본다. 이러한 결과를 토대로 키 생성 단계의 안전성 확보를 위한 안전성 요구사항을 도출한다.

1.1 가능한 공격 모델

1.1.1 대칭키 암호 방식

대칭키 암호 방식은 그 방식에 따라 스트림 암호 방식과 블록 암호 방식으로 구분할 수 있다. 스트림 암호 방식은 임의로 선택한 키를 사용하여 비트 단위로 암호화 및 복호화를 수행하는 방식이고, 블록 암호 방식은 DES, SEED와 같은 상용화된 암호 방식으로 평문을 일정 블록으로 나누어 각 블록을 동일한 키로 암호화하고 같은 방법으로 복호화한다. 이러한 방식들은 모두 일정한 난수열을 생성하여 랜덤한 비트열을 키 자체로 사용하거나, 키 스케줄러(key scheduler)의 입력 시드(seed) 값으로 사용하여 키를 생성하게 된다.

비트 단위로 암·복호화를 수행하는 스트림 암호 방식의 경우 키 생성 과정에서 가능한 공격은 랜덤한 비트열을 임의로 생성하여 시스템에 적합한 키를 구하는 것이다. 이러한 공격은 가능한 모든 비트열을 모두 구해야 하므로 키의 길이가 길 경우 공격자가 정당한 키를 구하여 공격하는 것이 매우 어렵다.

블록 암호 방식에서 사용되는 키 스케줄러는 고정된 길이의 시드 값으로부터 암·복호화 과정에서 필요로 하는 수만큼의 라운드 키(round key)를 여러 가지 연산을 통해서 생성하는 알고리즘으로 랜덤한 라운드 키를 생성해야 한다. 이렇게 생성한 키는 시드 값과 라운드 키 사이에 연관 관계가 존재하지 않도록 하여 생성된 라운드 키로부터 입력 시드 값을 알아내는 것이 불가능해야 한다. 즉, 라운드 키로부터 입력 시드 값을 유추할 수 없도록 하여 라운드 키가 유출되더라도 입력 시드 값에 대한 안전성은 보장되어야 한다. 이는 평문과 암호문의 쌍을 이용하여 키를 찾는 방법인 연관키 공격(related key attack)을 방지하기 위함이다. 이 공격 방법은 특히 블록 암호 방식이 해쉬 함수로 사용될 경우 적용이 용이하다[10].

암호 시스템의 안전성을 저하하는 요소 중에 암호 과정의 문제점이외에 키 스케줄러를 통해 생성되는 서브키의 특성에 따라서 키 자체에 문제가 발생할 수 있다. 키 스케줄러의 취약점은 모든 평문에 대해 동일한 암호문을 출력하는 키, 같은 키로 연속하여 두 번 암호화했을 때 평문이 나오게 하는 취약키 (즉, $E_K(E_K(M))=M$이 되는 키), 한 키로 암호화한 것을 다시 다른 키로 암호화했을 때 평문이 나오는 유사 취약키 (즉, $E_{K_0}(E_{K_1}(M))=M$이 되는 키) 등이 있다[11]. 또한 어떠한 키의 특수성에 의해서 각종 공격 방식이 적용되었을 때 쉽게 해독이 될 경우 이 키도 취약키라 할 수 있다. 이러한 키는 IDEA와 같이 산술 연산이 주로 사용되는 경우에 종종 발생한다[12].

또 다른 공격 모델로는 임의의 키를 설정하고 이 키에 대한 특정 비트를 반전시켜 차분 공격에 적용하는 형태의 공격 모델 등이 있다.

1.1.2 공개키 암호 방식

공개키 암호 방식에서 사용하는 키 쌍은 다항식 시

간 내에 풀 수 없는 문제에 기반하여 수학적 연관성을 갖도록 구성되어 있다. 따라서 키 생성 단계의 안전성 요구사항 분석을 위해서는 그 기반 문제 자체의 안전성 평가가 선행되어야 한다. 본 절에서는 크게 이산대수 문제의 어려움과 소인수분해 문제의 어려움을 기반으로 키 생성 단계의 위험 요소를 구분하고, 그 중 이산대수 문제는 유한체 상의 이산대수 문제와 타원곡선 상에서의 이산대수 문제로 나누어 가능한 공격 모델을 분류하여 키 생성 단계의 위험 요소를 분석한다[13].

■ 이산대수 문제

〔유한체 상에서의 이산대수 문제〕

일반적으로 이산대수 문제는 위수(order)가 $p-1$인 순환 그룹(finite cyclic group) G, 그룹 G의 원시원소 g와 G의 원소 β가 주어졌을 때, $g^x=\beta$를 만족하는 (단, $0<x\leq p-1$)를 계산하는 것으로, 이 x를 원시원소 g에 대한 β의 이산대수(discrete logarithm)라 부른다. 지금까지 알려진 유한체 상에서 이산대수를 구하는 알고리즘은 소모적 공격(exhaustive search) 알고리즘, Shanks의 baby-step /giant-step 알고리즘, Pollard's rho 알고리즘, Pollard's lambda 알고리즘, Pohlig-Hellman 알고리즘, 그리고 Index-calculus 알고리즘 등이 있다.

· 소모적 공격 알고리즘

이 방법은 위수가 $p-1$인 순환 그룹 G의 원시원소 g와 G의 원소 β에 대해, $\beta\equiv g^x(\mathrm{mod}\,p)$의 이산대수 x를 구하기 위해서 $g^0, g^1, g^2, \cdots$을 순차적으로 계산하여 만족하는 x 값을 찾는 방식이다. 이 방식은 g의 위수가 n일 경우 $O(n)$번의 곱셈을 요구하므로 n의 크기가 큰 경우에는 비효율적인 알고리즘이라 할 수 있다.

· Baby-step/giant-step 알고리즘

Baby-step/giant-step 알고리즘은 소모적 공격의 시간과 메모리 사이의 trade-off를 고려한 방법이라고 할 수 있다. 즉, 만족하는 이산대수 x를 찾기 위하여 특정 값까지는 사전에 계산하여 테이블을 생성하고, 나머지 부분에 대해서 소모적 공격 방법을 적용하는 알고리즘이다. 따라서 이산대수를 찾는 속도를 빠르게 하기 위해서는 사전에 계산하여 생성하는 테이블을 크게 하여야 하고, 테이블에 저장된 값이 적을수록 이산대수를 구하기 위한 시간이 많이 걸리게 된다. 또한, 그룹의 위수가 n일 때 $O(\sqrt{n})$의 계산량을 가지므로, n이 2^{160} 이상이면 이산대수 문제를 푸는 것이 계산상 불가능하게 된다.

· Pollard's rho 알고리즘

Pollard's rho 알고리즘은 baby-step/giant-step 알고리즘과 같은 수행시간을 가지면서, 아주 적은 양의 메모리를 필요로 하는 알고리즘이다. 또한 이 알고리즘은 Floyd's cycle-finding 알고리즘에 의한 랜덤 증가 함수를 사용하므로 랜덤화된 알고리즘이라 할 수 있다. 이 알고리즘은 같은 시간에 매우 적은 메모리에서 동작하므로 보다 실용적이지만, 이 방식을 적용하기 위해서는 그룹의 위수를 알고 있어야 한다는 조건이 있다.

· Pollard's lambda 알고리즘

Pollard's lambda 알고리즘은 이산대수 x가 $b<x<b+w$ 범위 안에 있다는 것을 알 때, 이산대수 $x(\mathrm{mod}\,p-1)$ (단, $ord(g)=p-1$)를 구하는 알고리즘이다. 앞에서 설명한 Pollard's rho 알고리즘은 임의의 범위에 있는 이산대수 x를 계산하는 반면, 이 알고리즘은 구하고자 하는 이산대수 x의 범위를 알 때 Pollard's lambda 알고리즘을 적용시키는 것이라 할 수 있다.

Pollard's lambda 알고리즘은 구하고자 하는 이산대수 x의 범위($b<x<b+w$)를 알 때, 그룹의 원시원소 g와 원소 β에 대해 i,j 값을 랜덤하게 증가시켜 가며 식 $g^i\cdot\beta\equiv g^j\cdot g^{b+w}\ (\mathrm{mod}\ p)$을 만족하는 i,j 값을 구한다. 단, $i<j+w$ 일 때까지만 반복한다. 만족하는 i,j 값을 구하면, 다음 두 식을 통해 이산대수 x를 계산할 수 있다.

$$\beta \equiv g^j \cdot g^{b+w} \cdot g^{-i}\ (\mathrm{mod}\ p)$$
$$\log_g \beta \equiv j + b + w - i\ (\mathrm{mod}\ p-1)$$

· Pohlig-Hellman 알고리즘

이산대수 문제는 원시원소 g의 위수의 크기에 따라 해를 구하는데 필요한 계산량이 증가하게 된다. 따라서, Pohlig-Hellman 알고리즘의 기본 개념은 소수 p가 $p=\prod_{i=1}^{n} q_i+1$인 경우, $(\beta)^{\frac{(p-1)}{q_i}} \equiv (g^x)^{\frac{(p-1)}{q_i}}$ (mod p)와 같이 양변에 $\frac{(p-1)}{q_i}$ 승을 하여 원시원

소의 g의 위수를 작은 소수 q_i로 낮추어 각각에 대해 이산대수 값(mod q_i)을 구하는 것이다. 이때, 식 $(\beta)^{\frac{(p-1)}{q_i}} \equiv (g^x)^{\frac{(p-1)}{q_i}} \pmod{p}$에서 원시원소의 위수는 작은 소수인 q_i가 되므로 Pollard's rho 알고리즘 등을 이용하여 x_i를 구할 수 있게 된다. 모든 q_i $(1 \le i \le n)$에 대해 이를 적용하여 각각의 x_i 값을 구한 후 중국인의 잉여 정리(CRT: Chinese Remainder Theorem)를 이용하면 구하고자 하는 이산대수 x를 계산할 수 있게된다.

· Index-calculus 알고리즘

Index-calculus 알고리즘은 지금까지 알려진 알고리즘들 중 이산대수 문제를 푸는 가장 효율적인 알고리즘이다. 따라서 이 방법을 적용할 수 있는 그룹에서는 subexponential-time 알고리즘을 제공하는 반면 그룹의 성질에 의존적이므로 모든 그룹에 적용할 수는 없다는 단점을 가진다.

Index-calculus 알고리즘을 사용하여 이산대수 x를 구하기 위해서는 먼저, 그룹의 모든 원소를 효율적으로 나타낼 수 있는 factor base S를 구성해야 한다. 다음으로 factor base의 원소들로 선형 방정식을 구성하여 각각의 이산대수 값을 계산하고 그들로부터 구하고자 하는 이산대수 x 값을 구하는 것이다. 이 알고리즘은 $O(\exp((c+o(1))(\log p)^{1/2}(\log\log p)^{1/2})$의 수행시간을 가진다.

〔타원곡선 상에서의 이산대수 문제〕

타원곡선 상에서의 이산대수 문제는 유한체 $F(q)$위에서 정의된 타원곡선 $E(F(q))$와 소수 위수 Q를 갖는 타원곡선 위의 점 $G \in E(F(q))$으로 생성되는 부분군 $\langle G\rangle$가 주어져 있을 때, 점 $T \in \langle G\rangle$에 대하여 $T = lG$를 만족하는 l, $(0 \le l < Q)$을 찾는 문제이다. 지금까지 알려진 타원곡선 상에서의 이산대수 문제를 풀기 위한 방법은 Pohlig-Silver-Hellman 알고리즘, Frey-Ruck 알고리즘, Me-nezes-Okamoto-Vanstone 알고리즘과 Araki-Satoh, Smart, Ruck 알고리즘 등이 있다.

· Pohlig-Silver-Hellman 알고리즘

이 알고리즘은 $E(F(q))$에서의 이산대수 문제를 위수 Q인 순환 부분군(cyclic subgroup)에서의 이산대수 문제로 줄여서 계산하는 "divide-and-conqure" 방법이다.

· Frey-Ruck 알고리즘과 Menezes-Okamoto-Vanstone 알고리즘

타원곡선 $E(F(q))$의 위수 Q인 순환부분군 $\langle G\rangle$에서의 이산대수 문제를 $F(q)$의 확장체(extension field) K에서의 이산대수 문제로 변환하는 알고리즘이다. 이때, K는 Q가 $\#(K^*)$을 나누게 되는 가장 작은 체(Field)이다. 초특이 타원곡선의 경우에는 확장 차수가 낮아 타원곡선에서의 이산대수 문제를 쉽게 유한체상의 이산대수 문제로 바꿀 수 있다.

· Araki-Satoh, Smart, Ruck 알고리즘

$F(q)$에서 정의된 타원곡선 $E(F(q))$이 비정규 타원곡선 $(\#E(F(q)) = q)$인 경우에 이산대수 문제를 푸는 알고리즘이다.

■ 소인수분해 문제

소인수분해 문제란 양의 정수 n이 주어졌을 때, $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$를 만족하는 n의 소수인 인수(prime factorization)를 찾는 것을 의미한다. 이는 p_i와 e_i을 알고 있는 사람은 n을 계산하는 것이 쉽지만 n만 알고 있는 사람은 n으로부터 p_i와 e_i을 찾는 소인수분해는 어렵다. 지금까지 제안된 소인수분해 알고리즘은 Pollard's rho 알고리즘, Pollard's p-1 알고리즘, Huber's factoring 알고리즘, Random square factoring 알고리즘, Quadratic sieve factoring 알고리즘과 Number field sieve factoring 알고리즘 등이 있다.

· Pollard's rho 알고리즘

Pollard's rho 알고리즘은 임의의 k에 대하여 $\sqrt{k}$개의 수를 선택하면 약 50%의 확률을 가지고 k를 법(modulo)으로 동일한 쌍이 적어도 하나 존재한다는 Birthday Paradox의 원리를 이용한 알고리즘이다. 이 알고리즘의 원리는 $n = p \cdot q$일 때 n을 인수 분해하려면 n을 법으로 한 $[\sqrt{a}]$개를 선택하여 50%의 확률을 가지고 최소한 한 쌍의 수 (x_i, x_j)가 존재하게 되고 식 $x_i = x_j \bmod p$을 만족하게 된다. 즉, $\gcd(x_i - x_j, n) = dp$를 만족하는 d를 찾는 것이 이 알고리즘의 목적이 된다.

· Pollard's p-1 알고리즘

양의 정수 n의 소수인 인수 p에 대하여 $p-1$의 모

든 소인수가 B 이하인 소수일 때, n을 B-smooth 혹은 bound B를 가지는 smooth라 말하는데, 이런 경우 Pollard's p-1 알고리즘을 사용하여 소인수분해를 수행한다.

· Huber's factoring 알고리즘

이 알고리즘은 Klaus Huber가 제안한 인수분해 방법으로 모든 소수 p는 Fibonacci 수열 F_i에 대하여 $F_{p-1} = 0 \bmod p$, 또는 $F_{p+1} = 0 \bmod p$을 만족하게 된다. 즉, 모든 소수 p에 대하여 $p \mid F_i$가 존재한다. 따라서, 이 알고리즘은 n의 소인수 p를 포함하는 Fibonacci 수 F_{u_p}의 인덱스 u_p가 적을 경우 인수분해를 쉽게 할 수 있다.

· Quadratic sieve 알고리즘

Quadratic sieve 인수분해 알고리즘은 random square 방식과는 달리 $a_i^2 \equiv b_i \pmod n$을 만족하는 (a_i, b_i)을 찾기 위하여 quadratic 다항식, $q(x) = (x+m)^2 - n (m = \lfloor \sqrt{n} \rfloor)$을 이용하며, relation을 찾는 속도를 개선하기 위하여 sieving 기법을 사용하고 있다.

1.2 안전성 요구사항

1.2.1 대칭키 암호 방식

DES와 같은 일반적인 상용 대칭키 암호 방식에서 키를 생성할 때에는 키 스케줄러를 이용하여 키를 생성하는데, 이러한 스케줄러는 주기적인 특성을 갖지 않는 난수열(random sequence)을 발생하게 되고, 이것은 조작될 수 없는 랜덤 시드(random seed)를 포함하게 된다. 이처럼 대칭키 암호 방식에서는 공격자가 예측할 수 없는 난수열을 시드로 사용하므로, 암호 시스템의 안전성과 직접적으로 연관된다. 따라서 대칭키 암호 시스템의 키 생성단계에서는 안전한 난수열을 생성하여 사용하는 것이 가장 중요하다. 안전한 난수열이 될 수 있는 조건은 통계적으로 균등한 분포(statistical uniformity)를 가져야 하고, 다음 비트를 예측할 수 없어야 하며(unpredictability), 이론적으로 증명 가능(theoretical support)해야 한다. 또한 긴 주기 (long period length)를 가져야 하며, 효율적으로 구성(efficiency)되어야 한다.

대칭키 암호 방식에서 사용하는 대표적인 난수 생성 방식으로는 선형 합동 의사 난수 생성기(linear congruential method)를 이용한 방식과 FIPS 140-1에서 정의된 해쉬 함수를 이용한 의사 난수 생성기, ANSI X9.17에 명시된 블록 암호 알고리즘을 이용한 난수 생성기가 있다. 또한 이차 잉여문제를 이용한 Blum Blum Shub의 난수 생성 방식 등이 있다.

1.2.2 공개키 암호 방식

공개키 암호 방식은 일반적으로 어려운 문제라고 알려진 이산대수 문제와 소인수분해 문제의 안전성을 기반으로 연구되고 있다. 이러한 문제들을 완벽하게 풀 수 있는 알고리즘이 아직 발견되지 않았지만, 앞의 절에서 살펴본 바와 같이 특수한 형태의 그룹이나 파라미터 설정에 따라 문제를 풀 수 있는 공격 모델 등이 존재하게 된다. 본 절에서는 공격 모델에 따라 키를 생성하는 과정에서 그룹 혹은 파라미터 설정 시 안전성을 보장할 수 있는 요구사항을 분석한다.

이산대수 문제와 소인수분해 문제는 모두 안전한 소수의 선택이 가장 중요한 요소라 할 수 있다. 대부분의 암호 시스템에서는 한 개 이상의 큰 소수를 필요로 하며, 이러한 시스템의 안전성은 시스템을 구성하는 소수의 크기나 형태에 영향을 받는다. 따라서 안전한 암호 시스템을 구성하기 위해서는 각 암호 방식의 요구사항에 맞는 큰 소수를 생성해야 한다[14].

안전한 소수를 생성하는 알고리즘으로는 일반적으로 Gordon's 알고리즘이 사용되며, 소수 생성 단계는 다음과 같다.

· 비트의 수가 서로 같은 큰 랜덤 소수 s, t 생성한다.
· 정수 i_0 선택한다.
· $2it+1$ (for $i = i_0, i_0+1, i_0+2, \ldots$)에서 첫 번째 소수를 찾고, $r = 2it+1$을 생성한다.
· $p_0 = 2(s^{r-2} \bmod r)s - 1$ 계산한다.
· 정수 j_0 선택한다.
· $p_0 + 2jrs$, for $j = j_0, j_0+1, j_0+2, \ldots$에서 첫번째 소수를 찾고, $p = p_0 + 2jrs$을 생성한다.

또한 생성된 소수 p가 만약 정수 r, s, t로 다음과 같은 식을 만족한다면 p는 안전한 소수(Strong primes)라 할 수 있다[15].

· r: $p-1$은 큰 소인수를 가지고 있다.
$p - 1 = p_0 + 2jrs - 1 \equiv 0 \pmod r$
· s: $p+1$은 큰 소인수를 가지고 있다.

$p+1=p_0+2jrs+1 \equiv 0(\mathrm{mod}\, s)$

· t: $r-1$은 큰 소인수를 가지고 있다.

$r-1=2it \equiv 0(\mathrm{mod}\, s)$

위의 알고리즘에 의해 선택한 소수는 이산대수 문제와 소인수분해 문제 기반의 암호 시스템에 사용되며, 각각 안전성을 보장하기 위해 만족해야 할 요구사항은 〔표 1〕과 같다[13].

〔표 1〕 안전한 파라미터를 선택하기 위한 요구사항

기반문제		요구사항
이산대수	유한체	◦ field prime order q의 비트 길이는 최소한 1024 비트 ◦ subgroup order r의 비트 길이는 최소한 160 비트 ◦ $r \mid q-1$ (for $n \mid m$, $n<m$) : r은 p^m-1로 나누어지면 (○), p^n-1로 나누어지면 (×)
	타원곡선	◦ 타원곡선 위수 $\#E(F(q))$의 가장 큰 소인수인 Q의 비트 길이는 최소한 160 비트 ◦ $q+1-\#E(F(q))$는 p의 배수가 되면 안 됨
소인수분해		◦ 이산대수 요구사항과 같이 안전한 소수 p, q를 선택 ◦ 소수 p, q의 곱인 n의 크기는 최소한 1024 비트 ◦ p, q의 크기는 비슷해야 함

다음 〔표 2〕는 각각의 기반 문제에서 선택된 파라미터가 공격 모델에 대해 안전하기 위한 최소한의 요구사항을 정리한 것이다.

2. 키 분배

키 분배 프로토콜은 설계하는 방식에 따라 다양한 형태의 공격 방법이 존재한다. 키 분배 프로토콜의 안전성은 이산대수 문제와 타원곡선 문제 등을 다항식 시간 내에 계산하는 것이 불가능하고, 키 분배 프로토콜에 사용되는 공개 파라미터들이 모두 안전하게 선택되더라도 프로토콜의 설계 방식에 따라 많은 차이를 가지게 된다. 즉, 사용하는 암호 알고리즘과 파라미터의 안전성뿐만 아니라 설계 방식에 의존하게 된다.

키 분배 프로토콜의 안정성 분석을 위해 가능한 공격 방법은 공격이 가능한 시점에 따라 프로토콜의 수행 전과 수행하는 중간, 그리고 수행한 후의 세 가지 공격 방법으로 나눌 수 있다[16]. 첫째, 키 분배 프로토콜을 수행하기 전에 가능한 공격에는 정당한 두 사용자 사이의 공개키 인증서 발급이나 키 발급 과정에서 공격이 가능한 unknown key share attack이 있으며, Pohlig-Hellman의 decom-position 개념을 이용한 subgroup confinement attack 그리고 사용자가 동일한 키를 여러 프로토콜에 사용하는 경우에

〔표 2〕 키 생성단계에서의 안전성 요구사항

기반문제		공격 모델	안전성 요구사항
이산대수	유한체	exhaustive search	◦ field order의 길이는 1024 비트 이상 ◦ subgroup order 길이는 160 비트 이상
		Baby-step/giant step	◦ 위와 동일
		Pollard's rho	◦ group의 위수가 노출되는 것을 방지
		Pollard's lambda	◦ group의 위수가 노출되는 것을 방지 ◦ 이산대수 x의 범위가 노출되는 것을 방지
		Pohlig-Hellman	◦ group의 위수가 작은 소수들로 이루어진 smooth 그룹으로 되는 것을 방지 ◦ group의 위수인 $p-1$이 최소한 하나 이상의 큰 소수를 인수로 포함하도록 함
		Index-calculus	◦ group의 원소들이 factor base를 이루는 것을 방지
	타원곡선	Pohlig-Silver-Hellman	◦ 타원곡선 위수 $\#E(F(q))$의 가장 큰 소인수인 Q가 160 비트 이상으로 함
		Frey-Ruck	◦ 선택된 타원곡선이 비정규 타원곡선이 아니어야 함
		Menezes-Okamoto-Vanstone	◦ 초특이(super-sigular) 곡선 방지 ◦ $F(q)$를 $F(q^B)$로 변형 가능할 때, B의 크기를 크게 함
		Araki-Satoh, Smart,Ruck	◦ 선택된 타원곡선이 비정규 타원곡선이 아니어야 함
소인수분해		Pollard's rho	◦ n이 작은 인수를 갖는 것을 방지
		Pollard's p-1	◦ n의 인수 p에 대해 $p-1$의 모든 소인수가 일정 크기 이상이 되도록 함

가능한 chosen protocol attack 등이 있다. 둘째, 키 분배 프로토콜을 수행하는 과정 중에 가능한 공격 방법에는 Ciphertext-only attack과 middle-person attack 등이 있다. 가장 초보적인 공격형태인 Ciphertext-only attack은 프로토콜이 진행되는 과정에서 통신망을 통해 전송되는 암호문과 사용자의 공개정보만을 이용하여, 해당 사용자들이 통신에 사용한 세션키를 계산하고 암호문을 해독하는 공격 방법이다. middle-person attack은 키 분배 프로토콜을 수행하는 과정에서 공격자가 불법적으로 참여하여 전송정보를 도청(eavesdrop)·변경(modification) 또는 재전송(replay)하는 공격 방법으로 replay attack, intruder-in-the-middle attack, oracle session attack, interlock attack 등이 있다. 셋째, 키 분배 프로토콜 수행 후 가능한 공격 방법에는 known key attack이 있으며, 고려사항으로는 perfect forward secrecy 보장 문제가 있다. known key attack은 일반적인 암호 시스템에서 known plaintext attack과 유사한 방법으로 공격자가 이전 세션에서 분배된 세션키를 알고 있는 경우 이를 사용하여 현재의 세션키를 알아내거나 암호문을 해독하는 공격 방법이다. 이러한 known key attack에는 known key를 이용하여 해당 세션의 세션키 또는 세션키를 구할 수 있는 정보 등을 계산해내는 known key passive attack과 자신을 합법적인 사용자로 위장하여 정당한 사용자와 세션키를 설정하는 known key impersonation attack, 그리고 known key impersonation attack에서의 두 사용자간에 설정을 3명 이상으로 확장한 known key triangle attack 등으로 나눌 수 있다. 프로토콜 수행 후, perfect forward secrecy 보장 문제는 사용자의 long term key와 같이 비밀키가 노출된 경우라도 두 사용자간에 설정된 세션키는 여전히 안전하도록 보장하는 것으로 키 분배 프로토콜을 설계하는데 고려되어야 할 중요한 요소이다[17].

2.1 가능한 공격 모델

2.1.1 키 분배 프로토콜을 수행하기 전에 가능한 공격

· Unknown key share attack

Unknown key share attack은 합법적인 두 사용자 A, B사이에 실제적인 키 분배 프로토콜이 수행된 적이 없는 경우에도 가능한 공격 방법이다. 이러한 공격 방법의 예로는 공개키 등록 과정 중에 인증기관의 부주의로 인해 두 사용자간에 동일한 공개키가 존재할 경우 공격이 가능하다. Unknown key share attack은 사용자 A가 사용자 B와 세션키를 공유했다고 믿고, B는 공격자 E와 세션키를 공유했다고 여기지만 실제로는 A, B 사이에 세션키가 공유되는 공격으로 크게 다음과 같이 공격자가 사용자들의 공개키를 자신의 공개키로 등록하는 경우와 사용자들의 공개키를 변형하여 공격을 하는 방법 등으로 나누어진다.

· Subgroup confinement attack

소수의 형태에 따라 Z_p^* 상에는 작은 크기의 위수를 가지는 원소들이 있다. 이 원소들은 각기 작은 서브그룹들을 구성되어 공격자는 그러한 서브그룹 상에서 이산대수 문제를 쉽게 풀 수 있다. 따라서 키 분배 프로토콜의 설계 초기 단계에서 서브그룹을 이용하는 공격을 중요하게 고려해야 한다. 이러한 서브그룹을 이용한 공격을 방지하기 위한 가장 근본적인 해결책은 모듈러 p와 기저(base) g의 선택에 있어서 다음과 같은 사항을 고려해야 한다. 모듈러 p의 선택 시 $\psi(n)=p-1$이 작은 소수의 곱들로 이루어지지 않도록 하여 공격자가 서브그룹 상에서 이산대수 문제를 풀 수 없도록 소수 p를 선택하는 것이다. 또한 기저를 선택할 경우에는 소수 위수를 갖는 기저의 선택으로 계산 효율성과 안전성을 함께 유지해야 한다. 이러한 해결책들은 보통 공개키에 대한 검사 과정과 함께 고려되어 위에서 제시한 요구사항들을 만족하고 있는지 확인해야 한다.

· Chosen protocol attack

Chosen protocol attack은 두 개의 다른 프로토콜에 동일한 공개키를 사용할 때 가능한 공격 방법으로 하나의 프로토콜에서 생성된 정보를 이용하여 다른 프로토콜을 공격하는 방식이다. 이 방식은 공격자 E가 정당한 사용자 A, B 사이에서 보안성이 높은 target 프로토콜을 공격하기 위해 target 프로토콜에 대한 공격을 수행할 수 있도록 생성된 tailor-made 프로토콜(chosen 프로토콜)을 만들어서, chosen 프로토콜의 토큰(전송정보)을 이용하여 target 프로토콜을 공격하는 방법이다. 특히, 이러한 공격이 가능한 이유는 암호학적 API(Application Program Interface)가 널리 확산됨에 따라 많은 프로토콜들이 보안을 위한 암호 방식을 사용하고 있고, 인증서의 형태로 발행된 공개키가 여러 프로토콜에 다양하게 사용될 수 있기 때문이다.

2.1.2 키 분배 프로토콜을 수행 중에 가능한 공격

· Ciphertext-only attack

Ciphertext-only attack은 암호문과 공개정보만

을 이용하여 개인키와 관련된 정보 또는 세션키를 알아내거나 암호문을 해독하는 공격 방법이다. 도청을 통해서 정당한 사용자 사이에 전송되는 정보와 사용자의 공개키와 같은 공개정보만을 이용하여 세션키를 알아내려고 시도하는 Ciphertext-only passive attack과 전송되는 정보를 수정을 통해서 세션키를 알아내고자 하는 Ciphertext-only active attack으로 나눌 수 있다. Ciphertext- only passive attack은 수행하기가 쉬운 공격이나, 안전한 공개 파라미터가 선택되었다면 기존에 나와있는 프로토콜은 모두 이 공격에 안전하다. Ciphertext-only passive attack에 취약한 경우는 대개 공개 파라미터의 잘못된 선택에 의해 발생하게 된다. Ciphertext-only active attack은 Ciphertext-only passive attack과는 다르게 공격자가 두 사용자 사이에 전송된 정보를 도청한 후 수정하여 자신을 합법적인 사용자로 위장하는데 필요한 정보를 만들어 내고 이를 이용하여 프로토콜을 공격한다. 이와 같은 공격을 방지하기 위해서는 안전한 공개 파라미터의 선택이 중요하며 또한 공격자가 유효한 전송정보를 발생시킬 수 없도록 전송정보에 사용자의 비밀키 또는 개인키를 포함시키거나, 세션키 생성함수의 조정 또는 공격자가 정당한 사용자로 위장하지 못하도록 전자서명(digital signa-ture)을 사용하여 예방할 수 있다.

· Replay attack

Replay 공격은 공격자가 자신을 합법적인 사용자로 위장하여 정당한 사용자와 세션키를 설정하기 위해서 이전 프로토콜에서 전송된 메시지를 이용하는 공격이다. Replay attack의 공격 방법은 크게 이전에 전송된 메시지를 재사용하는 방법과 현재 프로토콜에서 전송되는 메시지를 재사용하는 방법이 있다. 또한 재사용된 메시지를 수신하는 사용자에 따라 송신자의 메시지를 다시 송신자에게 되돌리는 reflection attack과 송신자의 메시지가 정당한 수신자에게 전송되지 않고 제3자에게 전송되는 interleaving attack 등이 있다.

일반적으로 각 프로토콜의 특성에 따라 메시지의 재전송 공격을 예방하기 위한 방법은 다양하다. 이산대수에 기반한 키 동의 프로토콜에서는 보통 key freshness를 제공함으로써 공격자가 세션키 전송에 필요한 정보를 재전송하여도 세션키 생성에 영향을 줄 수 없도록 설계하여야 한다.

· Intruder-in-the-middle attack

Intruder-in-the-middle attack은 정당한 사용자 사이의 통신에서 공격자가 중간에 위치하여 전송되는 정보들을 불법으로 도청·변경하여 전송함으로써 두 사용자간의 세션키를 구해내는 공격 방법이다. 공격자가 전송정보를 도청하는 경우 정당한 사용자들은 자신들의 통신 내용이 도청됨을 알 수 없다.

이러한 공격을 예방하기 위해서는 전송정보에 변경되었을 때, 이를 탐지 가능하도록 하거나 정보의 누출이 일어나지 않도록 프로토콜을 설계함으로써 해결할 수 있다. 이산대수 문제에 기반한 키 동의 프로토콜의 경우 전송정보의 수정이 의심될 때에는 전자서명을 이용하여 해결할 수 있고, 정보의 누출이 우려될 때에는 안전한 공개 파라미터를 선택하여 사용함으로써 예방할 수 있다.

· Oracle session attack

Oracle session attack은 공격자가 사용자 A와 정당한 프로토콜 수행을 통해 얻은 정보를 이용하여 사용자 A로 위장하여 사용자 B와 정당한 프로토콜을 수행하고자하는 공격 방법이다. 즉, 공격자가 공격에 필요한 정보들을 얻기 위해 사용자 A를 oracle로 설정하여 수행하는 공격으로 이러한 공격을 oracle session attack이라 한다. 이러한 공격 과정에서 사용자 A는 공격자를 정당한 사용자로 인식하고 세션을 진행하고, 사용자 B는 공격자가 아닌 사용자 A와 세션을 맺는 것으로 인식하게 된다.

이러한 공격은 사용자를 인증하는 전송정보의 내용이 사용자만의 고유한 정보와 연결되어 있지 않기 때문에 발생한다. Oracle session attack을 방지하기 위해서는 사용자의 ID와 같이 사용자만의 고유 정보를 사용자 인증 정보에 포함시켜 인증 조건을 검사할 수 있도록 설계해야 한다.

· Interlock attack

Interlock attack은 Intruder-in-the-middle attack을 방지하기 위해 제안한 프로토콜에서, 상대방이 전송한 정보의 유효성을 검증하지 않을 경우 가능한 공격 방법이다. 이 공격은 전송정보를 주고받을 때마다 매번 전송정보의 유효성을 확인함으로써 방지할 수 있다.

2.1.3 키 분배 프로토콜 종료 후에 가능한 공격

· Known key attack

Known key attack은 공격자가 이전에 프로토콜에서 사용된 세션키를 알 때 현재 세션키를 구하기 위

해 시도하는 공격 방법이다. 이는 이전에 생성된 세션키가 현재의 세션키에 영향을 주기 때문에 발생한다. 이를 예방하기 위해서는 과거에 사용된 세션키를 안전하게 보호해야 하는데 이는 안전하게 유지해야 하는 정보가 프로토콜을 수행함에 따라 계속적으로 증가하게 된다는 문제가 있다. 따라서 known key attack은 키 분배 프로토콜의 설계 과정에서 반드시 고려되어야 할 사항으로써 그 중요성이 부각되고 있다.

Known key attack의 공격 방법에는 크게 known key passive attack과 known key impersonation attack으로 나누어진다.

known key passive attack은 과거의 세션키 및 해당하는 전송정보와 현재의 전송정보를 관측하여 정당한 사용자 사이에 공유하고 있는 현재의 세션키를 획득하고자 하는 방법을 말하며, known key impersonation attack은 공격자가 과거의 세션키 및 해당하는 전송정보를 획득하고 현재에 전송되는 정보를 수정함으로써, 다른 사용자로 위장하여 정당한 사용자와 같은 세션키를 획득하는 공격 방법을 말한다. 특히, known key impersonation attack을 두 사용자에서 3명 이상으로 확장한 경우에는 known key triangle attack이라 한다. Triangle attack은 Yacobi system, Goss system, Günther system, 그리고 COMSET의 키 교환 버전 등이 3자 이상의 객체들 사이에 사용되었을 경우에 적용 가능한 공격 방법으로, 사용자 A로 가장한 공격자가 B와 C 사이의 세션을 도청한 후(passive attack), 각각의 세션을 맺고 이 과정을 통하여 선택된 두 세션키를 획득(known key impersonation attack)함으로써, B와 C 사이의 세션키를 구성하는 키 재료들을 각각 얻을 수 있게 된다. Known key attack은 노출된 이전의 세션키가 공격자에게 현재의 세션키의 생성에 필요한 중요한 정보를 제공할 때에만 의미가 있다. 즉, 세션키가 매 세션마다 보존되는 고정된 비밀정보를 포함하고 있으면, 과거의 세션키를 아는 공격자는 전송정보의 도청이나 조작에 의해 세션키 생성함수로부터 이 고정된 항을 구할 수 있게 된다. 따라서, 공격자는 고정된 항으로부터 원하는 세션키를 직접 구할 수 있거나 향후 수행되는 프로토콜의 실행에서 impersonation attack을 수행하여 정당한 사용자와 세션키를 공유할 수 있게 된다. 따라서, known key attack에 안전한 시스템이 되기 위해서는 매 세션마다 계산되는 세션키에 고정된 항의 비밀정보가 존재하지 않도록 설계해야 한다.

· Perfect forward secrecy

Perfect forward secrecy는 공격자가 사용자의 비밀키(long-term key)를 알게되더라도 이를 통해 생성한 과거의 세션키를 구하는 것은 불가능하도록 하는 것이다. 이는 비밀키가 노출이 되더라도 비밀키를 이용해 생성된 이전의 세션키를 알 수 없게 함으로써 과거에 통신한 암호문들을 보호할 수 있다. 따라서 키 분배 프로토콜 설계 과정에서 중요하게 고려되어야 할 사항이다.

Perfect forward secrecy를 제공하기 위해서는 사용자의 비밀키 노출이 과거의 세션키에 영향을 주지 않도록 설계해야 하는데 이는 각 세션키 생성 과정에 랜덤정보가 포함되게 하며 또한 이 랜덤정보가 사용자의 비밀키와 연결되어 분리할 수 없도록 하여 보장할 수 있다. 즉, 전송정보나 세션키 계산함수가 사용자의 비밀키와 랜덤수가 분리되지 않는 형태로 구성됨으로써 과거의 랜덤정보를 알 수 없는 공격자가 세션키를 구할 수 없도록 하는 것을 의미한다.

2.2 안전성 요구사항

2.2.1 키 분배 프로토콜 수행 전의 공격에 대한 안전성 요구사항

공개키 암호방식에는 안전성에 대한 여러 가지 잠재적인 문제점이 있는데 이는 알고리즘과 공개정보가 공개되어 있고 공개정보와 수학적 연관이 있는 비밀정보만을 비공개 하기 때문에 발생한다. 또한 키 분배를 위한 디렉토리는 누구나 접근이 가능하기 때문에 그 위험성은 더욱 크다. 이를 예방하기 위해서는 하나의 공개키로 수행할 수 있는 프로토콜의 수를 제한하거나 프로토콜의 각 메시지에 대해서 유일성을 가지도록 프로토콜을 설계하여야 한다. [표 3]은 프로토콜 수행전의 공격에 대한 안전성 요구사항을 나타낸 것이다.

2.2.2 키 분배 프로토콜 수행 중의 공격에 대한 안전성 요구사항

Ciphertext-only attack은 공개정보만을 이용하는 수동적인 공격이기 때문에 프로토콜의 안전성을 보장할 수 있는 시스템 파라미터를 사용함으로써 공격의 대부분을 예방할 수 있다. 하지만, 프로토콜의 설계 방식에 따라 시스템의 안전성에 차이를 보이므로 프로토콜 설계 과정에서 이를 충분히 고려하여야 한다. [표 4]는 프로토콜 수행중의 공격에 대한 안전성 요구사항을 나타낸 것이다.

〔표 3〕 프로토콜 수행전의 공격에 대한 안전성 요구사항

공격 모델		안전성 요구사항
Unknown key share attack	동일한 공개키를 사용하는 공격	◦다른 사람과 동일한 키를 등록하는지를 검사 ◦해당하는 개인키를 가지고 있는 지를 확인 ◦인증서와 전송정보에 대한 MAC을 전송
	변형된 공개키를 사용하는 공격	◦해당하는 개인키를 가지고 있는 지를 확인 ◦인증서와 전송정보에 대한 MAC을 전송
Subgroup Confinement attack		◦ $\phi(p)=p-1$가 작은 소수의 곱으로 이루어지지 않도록 선택 ◦소수 위수를 갖는 기저를 선택 ◦공개키 검사
Chosen protocol attack		◦각 키마다 사용할 수 있는 프로토콜의 수를 제한 ◦응용 프로그램, 프로토콜, 버전, 프로토콜 단계마다 고유한 ID를 갖도록 함 ◦메시지의 인증, 서명 시 지정된 위치에 유일한 식별자 삽입 ◦메시지의 암·복호화 시 식별자와 키를 동시에 사용

〔표 4〕 프로토콜 수행중의 공격에 대한 안전성 요구사항

공격 모델		안전성 요구사항
Ciphertext only attack	Ciphertext only passive attack	◦안전한 공개 파라미터 선택
	Ciphertext only active attack	◦전송정보에 사용자의 비밀키 포함 ◦세션키 생성함수의 조정 ◦서명의 사용 ◦메시지의 무결성 확인
Replay attack	Interleaving attack	◦전송되는 정보의 구조를 서로 다르게 함 ◦직접적인 인증을 수행 ◦타임스탬프 사용
	Reflection attack	◦두 사용자가 각각 다른키를 사용하여 암호화 한 데이터를 전송 ◦타임스탬프 사용
Intruder-in-the-middle attack		◦타임스탬프 사용 ◦전자서명의 사용 ◦Challenge-Response 기법 수행
Oracle session attack		◦전송정보에 상대방의 ID 결합하여 전송
Interlock attack		◦메시지를 주고받을 때마다 매번 전송정보의 유효성을 검사

〔표 5〕 프로토콜 수행후의 공격에 대한 안전성 요구사항

공격 모델		안전성 요구사항
Known key attack	Known key passive attack	◦세션키 계산함수에 랜덤수를 사용함으로써, 비밀키로만 이루어진고정된 항이 존재하지 않도록 하여야 함
	Known key impersonation attack	◦공격자가 유효한 전송 정보를 발생시킬 수 없도록 전송정보에 두 사용자의 비밀키를 포함 시키거나, 전자서명 등을 사용함 ◦세션키 계산함수에 랜덤수를 사용
	Known key triangle attack	◦세션키의 형태가 부분키들의 EX-OR나 곱인 시스템에 적용 가능하므로, 그러한 형태를 피하기 위해서 해쉬 값 등을 세션키로 사용
Perfect forward secrecy		◦세션키의 생성에 랜덤 정보가 포함되도록 하고 동시에 세션키 생성함수가 사용자의 비밀키와 랜덤수가 분리되지 않는 형태로 구성

2.2.3 키 분배 프로토콜 수행 후의 공격에 대한 안전성 요구사항

키 분배 프로토콜의 수행후의 공격에 대해 안전성을 확보하기 위해 비밀키가 노출되더라도 이전에 사용된 세션키들을 노출된 비밀키로부터 알아낼 수 없도록 해야 한다. 또한 과거의 세션키가 노출되었을 경우에도 이를 통하여 현재의 세션키를 알 수 없어야 한다. 다음 〔표 5〕는 프로토콜 수행후의 공격에 대한 안전성 요구사항을 나타낸 것이다.

3. 키 저장

키 저장은 키 생명주기에서 키를 사용하기 전이나 후, 혹은 사용 중에 키를 안전하게 보관하고 백업하는 과정을 의미한다. 이는 크게 두 가지 방법으로 제공할 수 있다. 첫 번째는 안전한 보안 모듈을 사용하여 키를 평문 형태(plaintext key)로 저장하는 방식이고, 두 번째는 키를 키 암호키(KEK : Key Enciphered Key)로 암호화하여 저장하는 방식으로 구분할 수 있다.

일반적으로 키 저장을 위해 보안 모듈을 사용할 때는 키가 외부와 직접적으로 연결되지 않는 EEPROM (Electrically Erasable Programmable Read-Only Memory) 영역에 저장되고, 키 값은 보안 모듈 내부에서만 읽을 수 있도록 하기 때문에 보안 모듈

내의 IC(Integrated Circuit) 칩의 물리적 공격(Side Channel Attack) 방법이 보안 모듈과 관련된 주요 공격으로 대두되고 있다.

또한 키 암호키를 사용하여 키를 저장하는 방식은 대칭키를 사용하여 암호화하는 방식과 공개키를 사용하여 암호화하는 방식으로 구분할 수 있다. 이러한 방식에 대한 공격 모델은 주로 암호 알고리즘에 대한 공격 방법으로, 대칭키 방식에서는 차분·선형 암호 분석 공격 방법 등이 있고, 공개키 방식에 대해서는 알고리즘의 취약점 등을 이용한 공격 방법이 있다.

3.1 가능한 공격 모델

3.1.1 보안 모듈에 저장

보안 모듈의 칩 공격으로는 크게 시간 공격, 전력 공격, 오류 공격, 전자기 공격, 디캡 공격, 리셋 공격 등이 있다. 이중 전력분석 공격과 오류 공격 방법이 현재로서는 보안 모듈에 관련된 가장 강력한 공격으로 알려져 있다.

· 디캡 공격(Decab attack)

디캡 공격은 염산과 질산을 사용해 IC칩을 감싸고 있는 코팅 물질을 녹임으로써, 내부회로를 찾아내 이 회로에 전기적 신호등을 입력해서 칩 내부의 정보를 얻어내는 방법이다.

· 타이밍 공격(Timing attack)

보안 모듈 내의 IC칩 내부의 암호 알고리즘이 연산을 수행할 때 사용자의 비밀정보와 관련 있는 정보를 처리할 때 걸리는 시간과 일반적인 데이터를 처리할 때 걸리는 시간의 차를 분석해서 비밀정보를 알아내는 방법이다[18].

· 에러 메시지 공격(Error Message attack), 리셋 공격(Reset attack)

에러 메시지 공격과 리셋 공격은 보안 모듈 리더기에 내장되어 사용자를 인증해주는 SAM(Secure Access Module)을 이용하는 방법이다. SAM에서 강제적인 초기화 명령이나 잘못된 정보를 주기적으로 보안 모듈에 보내고 여기에 반응하는 보안 모듈의 메시지를 분석함으로써 내부 정보를 얻는 것이다.

· 보안 모듈의 전력분석 공격

보안 모듈 내에서 암호 시스템 연산 시에 소비 전력 신호의 누출을 이용하여 비밀키와 관련된 정보를 처리할 때 IC칩이 사용하는 전력량과 일반 데이터를 처리할 때 IC칩이 사용하는 전력량을 분석해서 사용자의 비밀키에 대한 정보를 추출하는 공격 방법을 말한다.

- SPA(Single Power Analysis)
  : 보안 모듈 내부의 암호 프로세서의 소비 전력을 직접 관찰하여 수행되는 명령을 구별하여 비밀키를 추출하는 방법
- DPA(Differential Power Analysis)
  : SPA방식에 통계적인 분석을 접목시킨 공격 방법. 데이터를 분석하는 방법은 우선 가능한 키 영역에서 키의 일부를 추측한 후에 분류함수를 이용해서 전력신호 데이터를 분류한다. 양분한 각각의 데이터에 대한 평균을 구하여 차분 신호를 계산하고, 추측한 키를 검증하는 방법
- IPA(Inferential Power Analysis)
  : 기존의 DPA와는 다르게 암호문과 복호문에 대한 정보가 필요하지 않으며, 한번의 분석으로 같은 모듈의 시스템 분석이 가능한 방법

3.1.2 KEK로 암호화하여 저장

■ 대칭키를 사용하여 암호화하는 경우

DES와 같은 일반 블록 알고리즘에 대한 공격 방법은 크게 평문과 암호문 사이의 통계적 특성을 이용하여 키를 찾아내는 통계적 암호분석(Statistical Cryptanalysis)과 알고리즘의 대수적인 성질을 이용하는 대수학적 암호분석(Algebraic Crypt-analysis)로 나눌 수 있다. 전자의 대표적인 예가 지금까지 개발된 블록 암호에 대한 분석법 중 가장 강력한 방법으로 알려진 Biham-Shamir의 차분 암호분석(DC : Differential Cryptanalysis)과 Matsui의 선형 암호분석(LC : Linear Crypt-analysis)이다.

· 차분 암호분석

차분 암호분석은 비선형 함수의 입력 쌍에 대한 차이(input difference)와 해당 출력 쌍에 대한 차이(output difference) 값들의 확률 분포가 균일하지 않다는 사실을 이용하는 선택 평문 공격(chosen plaintext attack)이다. 예를 들어, DES와 같은 Feistel 구조의 r-라운드 블록 암호에 대해 입력 쌍의 차가 P'일 때 (r-2)-라운드에서 출력 쌍의 차가 Q'가 될 확률이 (모든 가능한 평문 및 키 값들 상에서) 평균적으로 p가 된다고 하면 이런 입·출력 차의 쌍 (P', Q')을 (r-2)-라운드 differential pair라고

한다. 이는 $P'=P\oplus P^*$을 만족하는 임의의 평문 쌍 (P, P^*)을 암호화했을 때 (r-2)-라운드에서 해당 출력 차가 Q'이 되는 (Q, Q^*)가 나올 확률이 평균적으로 p가 됨을 의미한다. 차분 공격은 가능한 높은 확률로 발생하는 differential pair를 이용, 충분히 많은 암호문 쌍을 분석하여 마지막 라운드의 라운드 키 K_r을 찾고자 하는 것이다[19].

· 선형 암호분석

선형 암호분석은 임의의 주어진 평문 P와 키 K 및 이에 대응하는 암호문 C에 대하여 확률 $p=\frac{1}{2}+a$로 $P\cdot\Lambda_P\oplus C\cdot\Lambda_C=K\cdot\Lambda_K$ 같은 선형 근사식이 존재할 때 적용 가능하다. 위의 식이 높은 확률로 성립하면, maximum likelihood법을 이용한 알고리즘을 통해 키의 한 비트를 추정할 수 있다. n 라운드의 암호를 공격하기 위해서는 $n-2$ 라운드의 최량 표현을 사용한다. 즉, $1, n$ 라운드는 첫 번째와 n 번째 라운드 함수 F에 입력되는 라운드 키 K_1, K_n만을 이용한다는 점을 이용하여 근사식 중에서 F함수를 넣어 계산한다. 그 결과는 $\frac{1}{2}$의 확률로 식 $P\cdot\Lambda_P\oplus C\cdot\Lambda_C\oplus F_1(P, K_1)\cdot\Lambda_1\oplus F_n(C, K_n)\cdot\Lambda_n=K\cdot\Lambda_K$을 $n-2$ 라운드의 최량 확률로 성립시키게 된다[20].

· 대수학적 암호 분석

고정된 키 K로 암호화된 n개의 평문-암호문 쌍이 주어지면 Lagrange interpolation에 의해 이 polynomial equation을 쉽게 계산할 수 있고, 이 polynomial은 키 K에 의한 암호화 함수와 동치이므로 키 K로 암호화된 임의의 암호문도 해독 가능하다.

■ 공개키를 사용하여 암호화하는 경우

공개키를 사용하여 키를 암호화하여 저장하는 경우에 가능한 공격 모델은 공개키 암호 시스템 자체에 대한 공격이 존재한다. 공개키 암호 시스템은 일반적으로 소인수분해 문제에 기반한 시스템과 이산대수 문제에 기반한 시스템으로 분류할 수 있고 대표적인 방식으로 RSA와 ElGamal 등이 있다. 본 단락에서는 키가 RSA 방식의 암호 시스템으로 저장되었을 경우의 공격 모델에 대해 분석한다[21].

· 고정된 $n=p\cdot q$를 사용하는 경우

모든 사용자들이 n의 인수 p, q는 모르는 채, 같은 n에 대하여 각각 서로 다른 (e, d)를 가지고 있음을 가정하면, 하나의 공개키와 비밀키 쌍을 알고 있는 사용자는 다른 사용자들의 비밀키를 쉽게 구할 수 있다.

$e\cdot d\equiv 1 \bmod \phi(n)$, $\phi(n)=(p-1)(q-1)$에서 $k=ed-1$이라고 하면, 모든 $g\in Z_N^*$에 대하여 $g^k\equiv 1 \bmod n$이 성립한다. $k=2^t r$ (r은 홀수)라고 할 때, $g^{k/2}$는 1의 제곱근, 즉 $\{\pm 1, \pm x\}$중의 하나가 된다. 여기서 x는 $x\equiv 1 \bmod p$, $x\equiv -1 \bmod q$를 만족하는 원소이다. 임의의 g를 선택하여 $g^{k/2}, g^{k/4}, \cdots, g^{k/2^t}$을 차례로 계산하여 ± 1이 나오지 않을 확률은 $\frac{1}{2}$ 이상이 된다. ± 1이 아닌 원소 $g^{k/2^i}$에 대하여 $\gcd(g^{k/2^i}\pm 1, n)$을 구하면 n의 소인수를 구할 수 있다.

· Broadcast attack

작은 공개키를 사용하는 경우의 가능한 공격은 한 사용자가 동일한 평문을 여러 사람들에게 암호화하여 보낼 때 가능한 Broadcast attack이 있다.

사용자 A가 각각의 공개된 n_i보다 작고, 동일한 내용의 평문 m을 여러 사람들 $B_1, B_2, \cdots, B_k$에게 암호화하여 보낸다고 가정한다. 이때, 사용자 A가 m을 인코딩하지 않은 채 그대로 암호화하여 보내고, 공격자가 A의 공개키 e의 개수만큼 암호문을 얻을 수 있으면 원래의 평문 m을 알아내는 것이 가능하게 된다.

· Flanklin-Reiter related message attack

만약 주어진 메시지 m_1, m_2에 대해 $M_1=Enc(m_1)$, $M_2=Enc(m_2)$가 알려진 다항식 $f(x)$에 의해서 $f(M_1)=M_2 \bmod n$라는 관계식이 성립하면 공격자는 $O(\log^2 n)$의 시간 안에 M_1, M_2를 알아낼 수 있다. 예를 들어, 평문 m에 대하여 임의의 k비트 난수 R을 선택하여 $M=Enc(m)=2^k m+R$ 이라고 가정하고, 공개키가 3이라고 하면 다음과 같은 식을 만족한다.

$$c=M^3=(2^k m+R)^3 \bmod n$$

이 때, 동일한 평문 m을 반복해서 암호화하여 여러 명의 상대방에게 보낸다고 할 때, 또 다른 난수 $R'=R+r$을 사용하여 암호화하면, 다음과 같은 식을 구할 수 있다.

$$c'=M'^3=(2^k m+R')^3=(M+r)^3 \bmod n$$

여기서 $f(x)=x^3-c$, $g(x)=(x+r)^3-c'$라고 하면 f, g는 모듈러 n을 공통근으로 갖게 된다. 따라서 f, g의 종결식(resultant) $R(f, g)$는 역시 모듈러 n으로 0이 된다.

$$Resultant(x^3-c, (x+r)^3-c')$$

$$= \text{def} \begin{vmatrix} 1 & 0 & 0 & -c & 0 & 0 \\ 0 & 1 & 0 & 0 & -c & 0 \\ 0 & 0 & 1 & 0 & 0 & -c \\ 1 & 3r & 3r^2 & r^3-c' & 0 & 0 \\ 0 & 1 & 3r & 3r^2 & r^3-c' & 0 \\ 0 & 0 & 1 & 3r & 3r^3 & r^3-c' \end{vmatrix}$$

$$= r^9+(3c-3c')r^6+(3c^2+21cc'+3(c')^2)r^3 +(c-c')^3 \equiv 0 \mod n$$

이것은 r에 대한 일변수함수로 볼 수 있으므로 r을 찾을 수 있다. 따라서 $f(x)=x+r$이라고 할 때, M과 평문 m을 구할 수 있다.

3.2 안전성 요구사항

3.2.1 보안 모듈에 저장

보안 모듈의 공격 방법인 시간 공격, 전력 공격, 오류 공격, 전자기 공격, 디캡 공격, 리셋 공격 등은 소프트웨어나 하드웨어로 구현 시 연산의 수행에 걸리는 시간, 데이터에 따른 소모 전력 변화 분석 그리고 외부의 물리적인 자극으로 인한 순간적인 회로상의 이상으로 연산 결과 값의 변화 등을 이용하여 키를 추출해 내는 공격이다. 실제로 이러한 공격들은 특정 부분에서 암호학적인 공격에 비해 훨씬 쉽게 암호를 깰 수 있다. 따라서 보안 모듈을 제작하거나 구현할 때 이러한 공격들에 대한 대비책을 고려하여 설계할 필요가 있다. 보안 모듈에 키를 저장했을 때의 안전성 요구사항은 위의 〔표 6〕과 같다.

〔표 6〕 보안 모듈에 키를 저장했을 때의 안전성 요구사항

공격 모델		안전성 요구사항
디캡 공격		◦IC 칩 설계 시 복잡하게 회로 설계 ◦실제 데이터가 오가는 회로와 거짓 회로를 섞어 설계
타이밍 공격		◦키와 관련된 연산을 수행하는 시간과 일반적인 데이터를 처리할 때 걸리는 시간의 차이를 줄임
에러 메시지 공격 및 리셋 공격		◦보안 모듈 제작 시 잘못된 정보가 들어오면 IC 칩의 작동을 자동으로 중지할 수 있도록 설계
전력 분석 공격	SPA	◦소비전력의 신호를 감소시킴 ◦수행되는 명령의 구분에 의한 키 정보 유출을 차단
	DPA	◦통계적인 분석이 불가능하게 보안 모듈을 제작 ◦차분 소비 전력 신호를 줄임 ◦은닉서명 기법을 이용 ◦혼합 알고리즘을 사용
	IPA	◦DPA의 요구사항과 동일 ◦키에 대한 연산 위치를 복수로 설정 ◦비밀키가 연산되는 시점을 임의로 변화시킴

3.2.2 KEK로 암호화하여 저장

■ 대칭키를 사용하여 암호화하는 경우

DES와 같은 블록 암호는 기본적으로 비선형 변환과 선형 변환의 적절한 조합에 의해 설계된다. 비선형 변환에는 테이블을 이용하는 치환(substi- tution : S-box), 곱셈, 가변 데이터에 의한 회전 이동(data-dependent rotation) 등이 이용된다. 각 연산에 따라 안전도나 프로세서·하드웨어에 따른 효율성 등에 있어서 장·단점이 있으나, 일반적으로 작은 단위의 S-box를 적절히 이용하는 것이 대부분의 응용 환경에서 효율적이라고 알려져 있다. 또한 블록암호가 위의 공격 모델과 같은 통계적 분석 기법들에 대한 안전성을 보장하기 위해서는 비선형 연산 못지 않게 비선형 연산 결과를 가능한 많은 데이터 비트들과 섞어주는 효율적인 선형 변환을 사용하여야 한다[22]. 실제로 가장 강력한 분석 기법인 차분·선형 암호분석에서의 성공 확률은 얼마나 효과적인 선형 변환을 사용하느냐에 의존한다. 따라서 통계적 성질을 통한 공격 모델과 대수적인 성질을 이용한 공격 모델에 안전하기 위해서는 bit permutation, PHT(Pseudo-Hadamard Transform), MDS matrix multiplication 등의 선형 변환을 사용하여 이용 가능한 다양한 연산들을 적절히 결합하여 안전하면서도 효율성을 높일 수 있는 알고리즘을 설계하고 사용해야 한다. 다음 〔표 7〕은 대칭키를 KEK로 사용하여 키를 저장하는 방식에서 만족해야 하는 안전성 요구사항을 정리한 것이다.

〔표 7〕 대칭키로 암호화한 경우의 안전성 요구사항

공격모델	안전성 요구사항
차분 암호 분석	◦효과적인 선형 변환 사용 · bit permutation · PHT · MDS matrix multiplication
선형 암호 분석	◦효과적인 선형 변환 사용 · bit permutation · PHT · MDS matrix multiplication
대수적 암호분석	◦낮은 비선형 차수를 갖는 라운드 함수의 사용 금지

■ 공개키를 사용하여 암호화하는 경우

RSA 암호 시스템에 대한 공격은 일반적으로 암호문으로부터 평문을 알아내거나 혹은 시스템 내부의 비밀키를 알아내는 것으로 분류할 수 있다. 이러한 공격 방법에는 기본적으로 소인수분해 문제를 푸는 알고리즘을 사용하여 기반 문제 자체를 깨는 방법과 공통 모듈러를 사용하는 경우, 그리고 작은 공개키를 사용하는 경우에 대응되는 공격 방법 등이 존재한다[23]. 따라서 RSA 암호 시스템을 사용하는 사용자는 서로 다른 모듈러를 사용해야 하고, 서명 검증의 효율성과 암호화의 시간을 단축시키기 위해 공개키를 3과 같은 값을 사용하는 것을 방지해야 한다. 따라서 공개키의 크기가 적당하면서도 가급적 모듈러 연산의 효율적인 계산을 수행할 수 있는 공개키를 사용하는 것이 바람직하다. 공개키의 크기를 65537(Fermat의 4번째 소수)로 하면 암호화 연산을 수행 시 곱셈 연산 수가 17번 정도로 적당하게 된다. 또한 평문을 암호화할 때는 반드시 적당한 인코딩 함수를 사용하여 변환을 해야하는데, 특히 패딩을 덧붙이는 경우 랜덤한 패딩을 사용해야 하고 패딩의 길이는 적어도 $\log n/9$보다는 길어야 한다. 다음 [표 8]은 RSA를 사용하여 키를 암호화하고 저장하는 방식에 대한 안전성 요구사항을 정리한 것이다.

[표 8] 공개키로 암호화한 경우의 안전성 요구사항

공격 모델		안전성 요구사항
공통 모듈러를 사용한 경우		◦ 사용자마다 다른 n을 사용
작은 공개키 e를 사용한 경우	Broadcast attack	◦ 적당한 크기의 e를 사용 : e=65537를 사용 ◦ 평문을 암호화할 때 인코딩 함수를 사용하여 변환
	Flanklin-Reiter related message attack	◦ 적당한 크기의 e를 사용 : e=65537를 사용 ◦ 평문에 난수로 된 패딩을 추가 : 패딩의 크기는 $log\ n/9$이상

4. 키 폐기

키 폐기는 키 관리 과정에서 사용된 키가 그 고유의 목적으로 사용되지 않거나 더 이상 사용될 필요가 없는 경우 안전하게 폐기하기 위한 절차를 제공한다. 이는 키와 관련된 모든 기록들을 제거함으로써 폐기 후에 남아있는 어떠한 정보를 가지고도 폐기된 키를 다시 복구할 수 없도록 한다. 또한 키를 폐기하기 전에 키들에 의해 보호되어진 자료들이 더 이상 필요 없는지의 여부를 확인해야 한다.

4.1 가능한 공격 모델

4.1.1 키 재료를 이용한 공격

· Key material attack

키에 대해 폐기가 결정된 후, 키를 폐기하는 과정에서 키를 계산하기 위해 필요한 관련된 키 재료를 모두 폐기하여야 한다. 만일 폐기하려는 키와 연관된 키 재료들을 모두 폐기하지 않고 공격자에게 노출된 경우 공격자는 이 정보를 이용하여 키를 계산할 수 있으며, 키를 이용하여 저장되었던 과거 시점의 정보를 획득하여 악의적인 목적으로 활용할 수 있다.

4.1.2 저장 방식에 따른 공격

· Key storage attack

키의 저장 방식은 크게 소프트웨어 방식과 저장 매체를 이용하는 방식으로 나눌 수 있다. 소프트웨어 방식의 경우 키를 사용하기 위해서 키와 키 재료가 RAM (Random Access Memory) 상에 로드되는데 이러한 비밀정보는 키의 사용이 완료된 후 완전히 제거되어야 한다. 또한 키 관련 정보가 EEPROM(Electrically Erasable PROM)에 저장되어 있는 경우 여러 번에 걸쳐 오버라이트(overwrite)해야하며, EPROM(Erasable Programmable Read-Only Memory)이나 PROM (Programmable Read-Only Memory)에 저장되어 있는 경우에는 물리적으로 완전히 파괴하여야 한다. 일반적인 저장 매체인 디스크를 이용하는 경우에도 키를 사용한 후에는 키가 로드된 부분에서 키와 관련된 정보가 노출되어서는 안 된다.

4.1.3 불완전하게 파기된 보안 모듈에 대한 공격

· Key destruction attack

키와 키 재료들이 보안 모듈에 저장되어 있는 경우 이를 물리적으로 파기함으로써 키와 관련된 키 재료들을 모두 폐기할 수 있다. 하지만 보안 모듈을 물리적으로 파기하는 과정에서 키 정보가 저장되어 있는 부분이 완전히 파기되지 않는 경우 키 정보가 저장되어 있는 부분은 물리적인 공격을 받을 수 있다.

4.2 안전성 요구사항

키 폐기 과정에서 키를 안전하게 폐기하기 위해서는 키 폐기 과정에서 발생할 수 있는 키 재료를 이용

한 공격 방법, 불완전하게 파기된 보안 모듈에 대한 공격 방법, 저장 방식에 따른 공격 방법에 대해 다음 〔표 9〕와 같은 안전성 요구사항을 만족해야 한다.

〔표 9〕 키 폐기에서의 안전성 요구사항

공격 모델	안전성 요구사항
키 재료를 이용하는 공격	◦ 폐기하려는 키와 연관된 키 재료를 모두 폐기
저장 방식에 따른 공격	◦ 소프트웨어를 이용한 RAM 키의 경우 키가 로드되었던 메모리 영역을 zeroization 시킴 ◦ EEPROM의 경우 여러 번에 걸쳐 오버라이트 수행 ◦ EPROM이나 PROM의 경우 물리적인 파괴 ◦ 디스크에 저장되어 있는 키의 경우 디스크 전체 영역에 대해서 오버라이트 혹은 포맷을 수행
불완전하게 파기된 보안 모듈의 공격	◦ 키 저장 요구사항 중 보안 모듈에 저장한 경우와 안전성 요구사항 동일

5. 키 복구

키 복구는 법 집행 기관이 합법적인 상황 하에서 암호문을 복호하거나, 사용자가 자신의 비밀키를 분실했을 때 등의 유사시에 정당한 사용자에게 복호화를 가능하게 할 수 있게 하는 능력을 제공하는 절차이다. 즉, 비밀키를 가지고 암호문을 직접 복호화하지 않고 특별한 상황에서 복호화를 가능하게 하는 방법이라고 할 수 있다[24].

현재까지 제시된 키 복구 기술은 데이터를 암호화하고 복호화하는 사용자 보안 구성요소와 키 복구 기관에 의해 관리되는 복구 기관 구성요소, 그리고 암호문에 부가된 데이터 복구 영역에서 얻어지는 정보와 복구 기관 요소에서 획득 가능한 정보로부터 암호화된 데이터를 복구할 수 있는 데이터 복구 구성요소의 특징에 따라 크게 위탁 방식과 캡슐화 방식, 그리고 TTP 기반의 방식으로 나누어 볼 수 있다. 따라서 본 단락에서는 키 복구 단계의 안전성 요구사항 분석을 위해 키 위탁 방식, 캡슐화 방식, TTP 기반 방식으로 분류하여 가능한 공격 모델을 분석하고 이에 대한 안전성 요구사항을 도출한다.

5.1 가능한 공격 모델

5.1.1 키 위탁 방식에 대한 공격

키 위탁 방식은 복구될 사용자의 비밀키, 비밀키의 부분 또는 키 관련 정보를 하나 이상의 신뢰기관에 위탁하는 방식으로 위탁되는 키는 사용자가 오랫동안 사용하게 되는 비밀키(long term 키)이다. 이 방식에서는 사용자의 비밀키가 위탁 기관에 직접 맡겨져야 하므로 개인의 프라이버시가 전적으로 위탁 기관에 의존한다는 문제점을 안고 있다. 그러므로 위탁 기관의 신뢰성이 매우 중요한 문제이며 이를 보장하기 위한 방법으로 두 개 이상의 위탁기관을 이용하는 비밀 분산 개념이 주로 사용되고 있다. 또한 법 집행기관에 의해 복구되는 키가 사용자의 l비밀키가 아닌 일정 기간동안만 사용하는 세션키가 되게 한다면 사용자들의 프라이버시 침해에 대한 문제도 어느 정도 해결이 가능하다. 그러나 위탁되는 키의 유효성에 대한 문제도 해결이 되어야 한다

반면 이러한 키 위탁 방식은 유사시에 키 복구를 확실하게 할 수 있다는 장점이 있으며 위탁 기관의 신뢰성만 보장된다면 편리하고 안전한 키 복구 방식이다. 이 방식에서 실제 키의 위탁은 암호 시스템 설정 시, 또는 비밀 통신의 전송 시에 일어난다.

■ EES에 대한 공격

EES(Escrowed Encryption Standard)는 사용자들의 프라이버시가 침해될 수 있을 뿐만 아니라 비공개 암호 알고리즘인 SKIPJACK를 사용함으로써 사용자들로부터 알고리즘 내에 트랩 도어가 존재할 가능성이 존재한다는 의심을 가지게 할 수 있으며, DES 등에 비해 그 안전성이 경험적으로 증명되지 않았다는 단점이 있다[25]. 또한 일단 한번 사용자의 키를 복구한 법 집행 기관은 법원의 허가 없이도 그 사용자의 암호문을 복호화할 수 있고, EES에는 tamper-resistant 하드웨어의 사용을 필수로 하기 때문에 시스템을 구현하는데 많은 비용이 소모된다. 무엇보다 키 복구를 어렵게 만드는 가장 큰 요인은 법 집행 기관이 키를 복구할 수 없도록 키 복구 기능을 우회할 수 있는 다양한 공격들이 존재한다는 것이다[26]. 이와 같은 EES에 대한 공격을 보면 다음과 같다.

· Non-interoperable Rogue Application attack

부당한 두 사용자들 사이에서 법 집행 기관에 수행하는 감시를 피해 암호 통신을 하려는 이 공격은 감청자, 즉 법 집행 기관이 LEAF(Law Enforcement Access Field)에서 원하는 정보를 얻어 낼 수 없도록 하는 LEAF Obscuring 공격과 LEAF 자체를 송신하지 않는 LEAF Feedback 공격으로 나누어진다.

▫ LEAF Obscuring attack
: LEAF Obscuring 공격은 EES 공격 중 가장 단순한 방법으로 암호문을 감청하는 법 집행 기관이 LEAF 또는 암호문을 복호화 할 수 없도록 하는 방식

▫ LEAF Feedback attack
: LEAF Feedback 공격은 법 집행 기관이 암호 통신을 해독하기 위해 필요한 정보인 LEAF를 보내지 않고 단지 암호문만 전송함으로써 법 집행 기관으로 하여금 암호문을 복호화 할 수 없게 하는 공격

· Interoperable Rogue Application attack

이 공격은 부당한 송신자와 정당한 수신자 사이에서 법 집행 기관의 키 복구를 우회하여 암호 통신을 가능케 하는 공격으로 수신측에서는 유효하여 LEAF 검증 과정을 통과하지만 실제로는 암호화된 세션키 KS를 포함하지 않은 LEAF를 생성함으로써 가능한 공격이다. 이 공격을 brute-force LEAF search라고 한다.

수신측에서 이루어지는 LEAF 검증 과정은 LEAF 내에 포함되어 있는 16 비트의 검사합 필드에 전적으로 의존하게 되고 송신측에서 임의로 생성한 128비트의 LEAF′은 1/216의 확률로 LEAF 검증 과정을 통과할 수 있다. 즉, 송신자는 216 정도의 시도를 통하여 수신측의 검증 과정을 통과할 수 있고, 이 때 정당한 KS가 아닌 임의의 KS′을 포함하는 LEAF′를 생성할 수 있다.

· Squeezing attack

Squeezing 공격은 복구 기관이 키 복구를 수행하지 않고는 통신자의 불법 사실을 알아차릴 수 없는 공격으로, 키 복구 기능을 우회하려는 공격자는 자신이 생성한 LEAF 대신에 합법적인 사용자의 LEAF′를 이용하여 암호 통신을 수행하게 된다. 이 공격을 이용하면 복구 기관이 불법 통신 사실을 감지하고, 그 사용자를 추적할 경우에 불법 사용자가 아닌 정당한 사용자가 추적된다.

· Self-Squeezing attack

Self squeezing 공격은 LEAF Feed-back 공격을 발전시킨 것으로써 복구 기관은 키 복구 없이 사용자의 불법 통신 사실을 알 수 없으며, squeezing 공격과 달리 제 3의 합법적인 사용자의 LEAF나 세션키를 필요로 하지 않는다.

이 공격의 내용은 다음과 같다. 먼저, 부정한 두 사용자는 세션키 KS와 그로부터 파생된 $KS' = f(KS)$를 두 사용자 사이에 합의된 일방향 함수 f를 사용하여 생성한다. 송신자는 KS와 KS'를 사용하여 LEAF와 LEAF′를 생성하며, EKS(m)|| LEAF 대신 EKS(m)|| LEAF′를 수신자에게 전송한다. 수신자는 수신한 EKS(m)||LEAF′에서 LEAF′를 LEAF로 교체한 후, EKS(m)을 복호화한다. 나중에 복구 기관이 키를 복구할 경우, 불법 통신 사실이 적발되지만 사용자들은 기기 테스트 중이었다고 주장할 수 있다.

■ FPKC에 대한 공격

FPKC(Fair Public Key Cryptosystem)은 사용자, 다수의 신뢰된 위탁기관, 그리고 인증기관으로 구성되는데, 키를 신뢰된 다수의 위탁기관에 안전하게 위탁하고 위탁된 키의 정당성을 인증기관이 확인하는 동시에, 인증기관이나 N-1개의 위탁기관이 사용자의 비밀정보에 대한 어떠한 부분정보도 알아낼 수 없고 오직 N개의 위탁기관이 모두 모여야만 비밀정보를 복구할 수 있도록 하는 비밀분산 방식을 이용한다. 이 때, 각 위탁기관에 위탁되는 부분정보의 조각이 사용자의 공개키에 대한 정당한 조각임을 확인할 수 있다[27].

FPKC는 기존의 암호 시스템을 공정한 공개키 암호 시스템으로 재구성한 것으로 다음과 같은 공격 방법이 존재한다[28].

· Shadow public-key attack

FPKC의 가장 큰 약점은 사용자가 자신의 비밀키와 공개키 쌍을 스스로 생성한다는 것이다. 이것은 subliminal channel을 사용하는 공격이 가능하며 그 내용은 다음과 같다.

FPKC의 일반 사용자들은 자신의 공개키와 비밀키 쌍 (P, s)를 생성하여 P는 공개하고, 정부 기관이 비밀키 s를 생성할 수 있도록 s의 조각들을 위탁 기관에 위탁한다. 이 때 공격자는 정당한 키 쌍 (P, s)외에 "shadow key"라고 불리는 키 쌍 (P', s')를 다음과 같이 생성한다.

$P' = f(P)$
(단, f는 계산이 쉽고 공개된 함수)

공격자는 일반 사용자와 같은 방법으로 (P, s)를 사

용하지만 s'는 자신의 shadow secret key로 보관한다. 만약 누군가 공격자에게 복구 기관이 복호할 수 없도록 메시지를 보내고자 한다면, 송신자는 공격자의 P를 이용하여 $P'=f(P)$를 계산하고 P'를 사용하여 메시지를 암호화하여 전송한다. 공격자는 비밀로 간직하고 있던 s'를 이용하여 P'로 암호화된 메시지를 복호화한다. 이 공격에서 공격자는 정당한 키 쌍인 (P, s)를 생성하여 합법적인 사용자가 사용하는 것과 같은 방법으로 이용하므로 공격자에 의한 부정 행위가 발각되지 않는다.

· Lack-of-fairness attack

FPKC를 사용하면 범죄자들을 감시하면서도 사용자의 프라이버시를 보호할 수 있다. 하지만 이 공격은 FPKC가 범죄자의 감시와 사용자의 프라이버시 보호라는 두 가지 요구를 동시에 만족하지 않는다. 예를 들어, ElGamal 암호 시스템에 기반한 FPKC의 경우 다음과 같은 공격이 가능하다.

사용자 A는 식 $y_A \equiv g^{x_A} \bmod p$을 만족하는 공개키와 비밀키의 쌍 (y_A, x_A)을 소유한다. 범죄자 B는 사용자 A에게 $K = g^R$, $C \equiv {y_A}^R \cdot M \bmod p$를 계산하여 암호문 (K, C)를 전송한다. 암호문을 수신한 사용자 A는 $M \equiv C \cdot (K^{x_A})^{-1} \bmod p$와 같이 메시지 M을 복호화한다.

이 과정에서 법 집행 기관이 공개된 정보 K와 C로부터 메시지 M을 얻기 위해서는 사용자 A의 비밀키 x_A를 복구해야만 한다. 즉, FPKC에서는 메시지를 송신하는 범죄자의 키가 아니라 범죄자로부터 메시지를 수신하는 사용자의 비밀키가 노출되는 것이다.

5.1.2 캡슐화 방식에 대한 공격

캡슐화 방식은 키 위탁 방식과는 달리 암호문을 생성하는 각 세션마다 키를 복구해 낼 수 있는 정보를 포함하는 필드를 생성하여 해당 암호 메시지에 부가시키는 방식으로 실제적인 키 위탁이 일어나지는 않는다. 법 집행 기관의 키 복구는 복구 기관이 가진 복구 키를 이용하여 암호화된 데이터에 부가된 복구 필드를 복구한 후 목적키를 얻을 수 있다. 그러므로 복구되는 키가 사용자의 long term 키가 아니라 세션키가 되도록 할 수 있기 때문에 도청 기관의 복구 능력을 제한할 수 있게 되어 사용자의 입장에서는 키 위탁 방식보다는 안전에 대한 확신을 가질 수 있다. 또한 기존의 프로토콜에서 확장 필드가 존재한다면 이를 이용하여 복구 필드를 부가시킴으로써 구현 비용의 절감과 높은 호환성이 가능하다는 장점이 있다.

그러나 복구 필드의 생성이 사용자 측에서 일어나므로 이 필드에 대한 사용자의 부정이 충분히 가능하다. 그러므로 복구 필드의 유효성 확인 과정이 반드시 필요하며 복구 기관의 신뢰성도 키 위탁 방식에서와 마찬가지로 보장되어야 한다.

· Blinding Cryptography attack

Binding cryptography 공격은 Binding cryptography를 사용하는 사용자들이 의심을 받지 않고 키 복구 기능을 우회하여 암호 통신을 할 수 있는 공격으로, LEAF obscuring 공격과 비슷하며 다음과 같이 수행된다.

사용자 A는 메시지 M에 대한 암호문 C를 생성한다. 이 때 세션키 KS는 신뢰받는 복구 기관(TRP : Trusted Recovery Party)의 공개키 $Kpub_{TRP}$와 사용자 B의 공개키 $Kpub_B$로 암호화하고, KS를 이용하여 메시지 M을 암호화한다.

$$C = E_{KS}(M) \| E_{Kpub_B}(KS) \| E_{Kpub_{TRP}}(KS) \| binding\ data$$

사용자 A는 암호문 C의 $E_{Kpub_{TRP}}(KS) \| binding\ data$ 부분을 삭제하고 나머지 부분에 수신자가 메시지를 복구할 수 있도록 하는 정보를 부가한 형태의 새로운 메시지로 암호문 C'를 생성하여 사용자 B에게 전송한다.

$$D = (info \| E_{KS}(M) \| E_{Kpub_B}(KS))$$

$$C' = E_{KS'}(D) \| E_{Kpub_B}(KS') \| E_{Kpub_{TRP}}(KS') \| binding\ data$$

사용자 B는 C'로부터 D를 복호화 한 후 info로부터 M을 얻어낸다.

5.2 안전성 요구사항

5.2.1 키 위탁 방식

키 위탁 방식에서는 법 집행 기관이 키를 복구할 수 없도록 키 복구 기능을 우회할 수 있는 공격과 복구 기관이 키 복구를 수행하지 않고는 통신자의 불법 사실을 감지할 수 없도록 하는 공격 등이 있다. 이에 대한 해결책으로 SKIPJACK과 같은 비공개 암호 알고리즘을

사용하는 대신 공개 암호 알고리즘이나 공개키 암호 시스템을 사용하도록 하고, 사용자간 사용하는 세션키를 생성할 때 재사용 공격에 대해 안전하도록 랜덤수를 삽입하도록 한다. 또한 LEAF 생성 시에도 랜덤수나 타임스탬프를 삽입하고, 키 생성 시에는 사용자와 신뢰기관이 함께 참여하도록 한다. 다음 〔표 10〕은 키 위탁 방식에 대한 안전성 요구사항을 나타낸 것이다.

〔표 10〕 키 위탁 방식에 대한 안전성 요구사항

공격 모델			안전성 요구사항
E E S	Non-interoperable Rogue Application attack	LEAF Obscuring	◦공개 암호 알고리즘사용
		LEAF Feedback	◦송신자가 임의의 블록을 메시지에 추가
	Interoperable Rogue Application attack		◦LEAF 내에 포함되어 있는 검사합 필드의 비트 확장
	Squeezing attack		◦세션키 생성 시 랜덤수 포함 ◦LEAF 재사용 금지 ◦LEAF 내에 랜덤수 또는 타임스탬프 삽입
	Self-squeezing attack		◦키 사용 용도 표시
F P K C	Shadow public key attack		◦키 생성 시 사용자와 신뢰기관이 함께 참여
	Lack-of-fairness attack		◦특정 형태의 암·복호화 형식을 갖는 어플리케이션 사용

5.2.2 캡슐화 방식

캡슐화 방식에서는 binding cryptography를 이용하여 키 복구를 우회하는 통신이 가능한데, 이는 송신자로부터 전송 받은 암호문에서 얻은 세션키로 암호문에 포함되어 있는 메시지를 복호화하여 의미 있는 문장이 나오는지 확인함으로써 방지할 수 있다. 다음 〔표 11〕은 캡슐화 방식에 대한 안전성 요구사항을 나타낸 것이다.

〔표 11〕 캡슐화 방식에 대한 안전성 요구사항

공격 모델	안전성 요구사항
Blinding Cryptography attack	◦송신자의 암호문에 포함된 메시지가 의미 있는 문장인지 확인

Ⅳ. 안전성 요구사항

공개키 및 대칭키 암호 방식에 대한 안전성 요구사항은 키의 생명주기에 따라 키 생성, 분배, 저장, 폐기 및 복구 등의 과정에서 각 단계에 적합한 요구사항이 적용되어야 한다.

첫째, 대칭키 암호 방식에서의 세션키 생성 시에는 공격자가 예측할 수 없는 난수열을 생성하여 사용해야 한다. 이를 위해 선형 궤환 시프트 레지스터, 이차 잉여문제를 이용한 Blum Blum Shub의 난수 생성 방식 및 FIPS 140-1에 정의된 해쉬 함수를 이용하여 난수를 생성한다.

또한 공개키 암호 방식에서의 공개키·비밀키 쌍은 이산대수 문제와 소인수분해 문제의 안전성에 기반하여 생성하며, 두 기반 문제 모두 각 방식에 맞는 큰 소수를 사용해야 한다. 유한체 상에서의 이산대수 문제 기반의 키 생성에는 1024bits 이상의 소수와 위수가 160bits 이상인 부분군의 원소를 사용해야 한다. 타원 곡선 상에서의 이산대수 문제 기반의 키는 위수의 인수 중 가장 큰 인수가 160bits 이상이 되어야 한다. 그리고 소인수분해 문제 기반에서 역시 1024bits 이상의 소수와 위수가 160bits 이상인 부분군의 원소를 사용해야 하며, 사용되는 두 소수의 크기는 비슷해야 하고 두 소수의 곱은 최소한 1024bits가 되어야 한다[29].

둘째, 두 사용자간 키를 분배하는 과정에서는 키 분배의 시점에 따라 적용되는 안전성 요구사항이 차이를 보이는데, 우선 키 분배 수행 전에는 공개키 디렉토리에 누구나 접근하여 위·변조 할 수 있음을 고려하여 이를 막기 위해 공개키 기반구조를 사용해야 한다. 키 분배 수행 중에는 증명 가능한 안전성을 가진 시스템 파라미터를 사용해야 하고, 메시지의 재전송 공격을 막기 위해 타임스탬프 및 랜덤수를 사용한다. 또한 전송정보에 대한 전자서명, challenge-response 방식을 사용하거나 전송정보들간에 상호 비대칭성을 갖도록 함으로써 명시적인 사용자 인증을 제공해야 한다. 마지막으로 키 분배 수행 후에는 비밀키가 노출되더라도 과거의 세션키들을 알지 못하도록 세션키를 생성할 때 랜덤정보를 포함시켜야 하고 이는 사용자의 비밀정보와 분리되지 않은 형태로 구성되어야 한다. 또한 과거의 세션키가 노출된 경우에도 현재의 세션키는 안전하게 사용할 수 있도록 하기 위해 전자서명, 랜덤수 등을 사용하여 공격자가 유효한 전송정보를 생성할 수 없도록 하거나, 과거의 세션키와 현재의 세션키 간 상호관계를 가지지 않게 하며, 세션키에 고정된 비밀정보로 이루어진 부분이 존재하지 않도록 해야 한다.

셋째, 키를 저장할 때는 보안 모듈에 저장하거나 KEK로 암호화하여 저장한다. 이 때 디캡 공격, 타이

밍 공격, 에러 메시지 공격 및 리셋 공격, 전력분석 공격에 안전하게 설계된 보안 모듈을 사용해야 한다. 또한 대칭키 방식을 이용한 KEK로 암호화하는 경우에는 통계적 성질을 통한 공격 모델과 대수적인 성질을 이용한 공격 모델에 안전하기 위해 bit permutation, PHT, MDS matrix multiplication 등의 선형 변환을 사용하여 이용 가능한 다양한 연산들을 적절히 결합해야 한다. 공개키 방식을 이용하여 KEK로 암호화하는 경우에는 키 생성 부분에서 언급한 바와 같이 이산대수 문제 기반과 소인수분해 문제 기반 모두 안전한 소수를 사용해야 한다[30].

넷째, 키 폐기 과정에서는 키 재료를 이용하는 공격에 안전하기 위해 폐기하려는 키와 연관된 키 재료들을 모두 폐기해야 한다. 또한 키가 RAM에 로드되어 사용된 경우 사용 후에는 모든 메모리 영역이 0으로 셋팅되어야 하고, 키가 디스크에 저장되어 사용된 경우에는 디스크 전체 영역을 오버라이트하거나 포맷해야 한다. 키 저장 시 물리적 공격에 안전한 보안 모듈에 저장함으로써 보안 모듈을 폐기한다해도 저장되어 있는 키에는 접근할 수 없도록 해야 한다.

마지막으로 키 복구 과정에서는 비공개 암호 알고리즘을 사용하는 대신 공개 암호 알고리즘이나 공개키 암호 시스템을 사용해야 하고, LEAF 생성 시 랜덤수나 타임스탬프를 삽입함으로써 재사용을 방지하며, 키를 생성할 때 사용자와 신뢰기관이 함께 참여하도록 한다.

이와 같이 키 관리 모델을 따르는 각 과정에서 안전한 키 관리가 이루어지기 위해서는 키 생명주기의 각 단계에서 필요로 하는 안전성 요구사항을 만족해야 한다.

V. 결 론

정보통신 기술의 발전으로 인해 언제 어디서든 개인, 기업, 정부간의 정보 교환이 원활히 이루어지고 있으며, 이를 이용한 다양한 서비스가 제공되고 있다. 그러나 개인 비밀정보, 기업, 정부의 기밀 정보 등의 누출 및 제 3자에 의한 정보 손상과 같은 정보화 사회의 역기능이 큰 사회 문제로 대두되고 있다.

따라서, 이를 방지하기 위해서 중요 정보와 관련된 키를 안전하게 관리하고 이용하는 키 관리 기술에 대한 관심이 증대되고 있으며, 이를 사용하는 사용자, 기업 및 정부 기관의 수가 증가하고 있다.

이에 키 관리 서비스 제공자의 경우, 안전한 키 관리 서비스를 제공하기 위해서는 키 관리 프로토콜을 구성하는 각 요소 즉, 키 생성, 키 분배, 키 저장, 키 폐기 및 키 복구에 대한 안전성을 분석하고 이를 키 관리 모델의 전체 안전성과 연결하여 안전성 요구 사항을 적용하여야 한다.

본 논문에서를 이를 바탕으로 키 관리를 구성하는 키 생명주기에 따른 안전성을 분석하고 안전성 요구 사항을 도출하였다.

본 연구 결과를 통하여 키 관리 서비스를 이용하는 사용자에게는 안전성 요구 사항을 만족하는 안전한 키 관리 서비스를 제공할 수 있으며, 키 관리 서비스를 제공하는 기업 및 개발자에게는 안전한 키 관리 시스템을 개발하기 위한 지침서로 이용될 수 있다. 또한, 정보화 사회의 사회 문제를 감소시켜 건전한 정보화 사회의 발판을 이룩할 수 있다.

향후 연구 계획으로는 본 논문에서 도출된 안전성 요구 사항을 바탕으로 각기 다른 키 생명주기별 키 관리 표준들 간의 호환성 및 사용자 편의성을 고려하여 안전한 키 관리 서비스 모델에 대한 설계가 필요하며, 유·무선에서 공통으로 사용할 수 있는 키 관리 서비스에 대한 연구 및 개발이 필요할 것으로 사려된다.

참 고 문 헌

[1] ISO/IEC, "Information technology-Security techniques-Key management- Part 1 : Framework", *ISO/IEC 11770-1*, Dec 1996.

[2] ISO/IEC, "Information technology-Security techniques-Key management- Part 2 : Mechanism using symmetric techniques", *ISO/IEC 11770-2*, Apr 1996.

[3] ISO/IEC, "Information technology-Security techniques-Key management- Part 3 : Mechanism using asymmetric techniques", *ISO/IEC 11770-3*, Nov 1999.

[4] ISO, "Banking-Key management(retail) -Part 1 : Introduction to key management", *ISO 11568-1*, Dec 1994.

[5] ISO, "Banking-Key management(retail) -Part 2 : Key management techniques for symmetric ciphers", *ISO 11568-2*, Dec 1994.

[6] ISO, "Banking-Key management(retail) -Part 3 : Key life cycle for symmetric ciphers ", *ISO 11568-3*, Dec 1994.

[7] ISO, "Banking-Key management(retail)

-Part 4 : Key management techniques using public key cryptography", *ISO 11568-4*, Jul 1998.

[8] ISO, "Banking-Key management(retail) -Part 5 : Key life cycle for public key cryptosystems", *ISO 11568-5*, Jul 1998.

[9] ISO, "Banking-Key management(retail) -Part 6 : Key management schemes", *ISO 11568-6*, Jul 1998.

[10] 임채훈, "블록 암호 알고리즘의 분석 기법", *KSIAM 춘계학술회의*, 1999.

[11] J. Kelsey, B. Schneier, D. Wagner," Key-schedule cryptanalysis of IDEA, DES, GOST, SAFER, and triple-DES", *In Advances in Cryptography - CRYPT '96, LNCS 1109*, pp. 237-251, 1996.

[12] E. Biham, "New types of crypt-analytic attacks using related keys", *In Advances in Cryptography - EURO CRYPT'93, LNCS 765*, pp. 229-246, 1994.

[13] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, "Handbook of Applied Cryptography", *CRC Press*, pp. 103-113, 1997.

[14] R. Anderson and S. Vaudenay, "Minding your p's and q's", *In Advances in Cryptography - ASIACRYPT'96, LNCS 1163*, pp. 15-25, 1996.

[15] C.H.Lim, P.J.Lee, "A recovery attack on discrete log-based schemes using a prime order subgroup", *In Advances in Cryptography - CRYPTO'97, LNCS 1294*, pp. 249-263, 1997.

[16] S.M. Bellovin, M. Merritt, "An Attack on the Interlock Protocol When Used for Authentication", *In IEEE Transactions on Information Theory 40:1*, pp. 273-275, 1994.

[17] M. Burmester, "On the risk of opening distributed keys", *In Advances in Cryptography - CRYPT'94, LNCS 839*, pp. 308-317, 1994.

[18] P. Kocher, "Timing attacks on implementation of Diffie-Hellman, RSA, DSS and other systems", *In Advances in Cryptography - CRYPT '96, LNCS 1109*, pp. 104-113, 1996.

[19] E. Biham, A. Shamir, "Differential cryptanalysis of DES-like crypto-systems", *Journal of Cryptology*, pp. 3-72, 1991.

[20] M. Matsui, "Linear cryptanalysis method for DES cipher", *In Advances in Cryptography - EUROCRYPT'93, LNCS 765*, pp. 386-397, 1994.

[21] D. Bleichenbacher, M. Joye, and J.J Quisquater, "A new and optimal chosen-message attack on RSA-Type cryptosystems", *Information and Communications Security, LNCS 1334*, pp. 302-313, 1997.

[22] J. Kelsey, B. Schneier. D. Wagner, "Related-key cryptanalysis of 3-Way, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA", *Proc. of ICICS'97*, pp. 233-246, 1997.

[23] D. Coppersmith, "Finding a small root of a bivariate integer equation : factoring with high bits known", *In Advances in Cryptography - EURO CRYPT'96, LNCS 1070*, pp. 178-189, 1996.

[24] NIST, "Escrowed Encryption Stan- dard", *Federal Information Processing Standards Publication 185*, 1994.

[25] Yair Frankel, Moti Yung, "Escrow Encryption Systems Visited : Attacks, Analysis and Designs", *In Advances in Cryptography - CRYPT'95, LNCS 963*, pp. 223-235, 1995.

[26] Matt Blaze, "Protocol Failure in the Escrowed Encryption Standard", *The 2nd ACM Conference on Computer and Communication Security*, pp. 57-59, 1994.

[27] Sivio Micali, "Fair public-key cryptosystems", *In Advances in Cryptography - CRYPT'92, LNCS 740*, pp. 113-138, 1992.

[28] Joseph Kilian, Tom Leighton, "Fair systems, Revisited", I*n Advances in Cryptography - CRYPT'95, LNCS 963*, pp. 208-220, 1995.

[29] T.M.A. Lomas, L. Gong, J.H. Saltzer, R.M Needham, "Reducing risks from

poorly chosen keys", *Proc. of 12th ACM Symposium on Operation System Principles*, pp. 14-18, 1989.

[30] J. Daemen, V. Rijmen, "Resistance against implementation attacks : a comparative study of the AES propo sals", *Proc. of 2nd AES Candidate Conference*, pp. 122-132, 1999.

〈著者紹介〉

한 종 수 (Jong-su Han)
학생회원

2002년 2월 : 성균관대학교 정보공학과 졸업(공학사)
2002년 3월~현재 : 성균관대학교 정보통신공학부 석사 과정

안 기 범 (Gi-bum Ahn)
학생회원

2002년 2월 : 성균관대학교 시스템경영공학부 졸업(공학사)
2002년 3월~현재 : 성균관대학교 정보통신공학부 석사과정

곽 진 (Jin Kwak)
학생회원

2000년 8월 : 성균관대학교 바이오메카드로닉스 공학과 졸업(공학사)
2001년 3월~2003년 2월 : 성균관대학교 대학원 전기전자 및 컴퓨터공학부 졸업(공학석사)
2003년 2월~현재 : 성균관대학교 정보통신공학부 박사과정

양 형 규 (Dongho Won)
정회원

1983년 2월 : 성균관대학교 전자공학과 졸업(공학사)
1985년 2월 : 성균관대학교 대학원 전자공학과 졸업(공학석사)
1984년 12월~1991년 2월 : 삼성전자 선임 연구원
1995년 2월 : 성균관대학교 대학원 정보공학과 졸업(공학박사)
1995년 3월~현재 : 강남대학교 컴퓨터미디어공학부 부교수

원 동 호 (Dongho Won)
종신회원

성균관대학교 전자공학과(학사,석사, 박사)
한국전자통신연구소 전임 연구원
일본 동경공대 객원연구원
성균관대학교 교학처장, 전기 전자 및 컴퓨터공학부장, 정보통신대학원장, 국무총리실 정보화추진위원회 자문위원
한국정보보호학회 이사, 부회장, 수석부회장, 회장
현재 : 성균관대학교 정보통신공학부 교수
성균관대학교 연구지원처장
한국정보보호학회 명예회장
정통부지정 정보보호인증기술연구센터 센터장