[UCI]I804:24011-200000231628

2006년 8월

석사학위논문

SDH를 기반으로 개인 식별에 중심을 둔 그룹 디지털 서명

조선대학교
CHOSUN UNIVERSITY
100355054 2006-09-07

조선대학교 대학원

전산통계학과

김 영 지

SDH를 기반으로 개인 식별에 중심을 둔 그룹 디지털 서명

Group Digital Signature centered on Personal Identification based on SDH

2006년 8월 25일

조선대학교대학원

전 산 통 계 학 과

김 영 지

SDH를 기반으로 개인 식별에 중심을 둔 그룹 디지털 서명

지도교수 배 상 현

이 논문을 이학 석사 학위신청 논문으로 제출함

2006 년 4 월 일

조선대학교대학원

전 산 통 계 학 과

김 영 지

김영지의 석사학위 논문을 인준함

위원장 조 선 대학교 교수 정 채 영

위원 조 선 대학교 교수 이 웅 기

위원 조 선 대학교 교수 배 상 현

2006 년 5 월 일

조선대학교 대학원

목 차

ABSTRACT

I. 서론 4

II. 정보 보안 방법 11

A. 접근통제 기술 11

B. 인증 기술 19

C. 보안 기술 20

D. 감사 기술 21

E. 암호 기술 23

F. 해쉬 함수(Hash Function) 29

G. 디지털 서명(Digital Signature) 41

III. 개인 식별이 가능한 디지털 서명 50

A. 방법 50

B. 예상 응용 분야 59

IV. 결론 및 향후 과제 61

참고문헌

표 목 차

[표 1] 접근통제 판단 기준 12

[표 2] 허용등급 14

[표 3] 접근통제 매트릭스(ACL) 15

[표 4] 객체 A에 대한 ACL 16

[표 5] 사용자 X에 대한 CL 17

[표 6] 정보보호비트(IPB) 18

[표 7] 대칭키 암호시스템과 공개키 암호 시스템의 비교 27

그 림 목 차

[그림 1] 암호기술의 분류 7
[그림 2] 접근통제 19
[그림 3] 감사 기술 21
[그림 4] 암호시스템 23
[그림 5] 암호화 및 복호화 함수 23
[그림 6] 스트림 암호 시스템 개념도 25
[그림 7] 암호학적 해쉬 함수의 분류 32
[그림 8] 해쉬 함수의 일반 모델 33
[그림 9] MAC를 이용한 무결성 확인 34
[그림 10] 메시지의 기밀성을 동반한 MAC의 무결성 확인 34
[그림 11] DES-CBC에 의한 MAC생성 35
[그림 12] MDC 해쉬 함수를 이용한 무결성 확인 36
[그림 13] 메시지의 기밀성을 동반한 무결성 확인 37
[그림 14] MAC의 암호화를 통한 무결성 확인 37
[그림 15] RABIN 방식 39
[그림 16] DAIVES-MEYER 방식 39
[그림 17] 메시지 부가형 디지털 서명의 생성과 확인 42
[그림 18] 메시지 복구형 디지털 서명의 생성과 확인 42
[그림 19] 메시지의 기밀성이 보장되는 RSA 디지털 서명 46
[그림 20] 익명의 트랜잭션 시스템 54
[그림 21] 제안한 익명의 트랜잭션 시스템 57

ABSTRACT

Group Digital Signature centered on Personal Identification based on SDH

Kim Young-Gie

Advisor: Prof. Bae Sang-Hyun

Department of Computer Science and Statistics

Graduate School of Chosun University

In this paper, Group Digital Signature a kind of Digital Signature is possible to check the truth, the signer in a member, using Signature and, way guarantee the anonymity that precise signer's identification not breathe a word. We can offer that more flexible than Established study change over from Assumption SDH(Strong Diffie-Hellman) to Transformation Fiat-Shamir. In this time, offered study takes about 400bite under when Established study takes 1238 bite.

And offered Digital Signature method don't stop reducing Signature's size and is centered on personal anonymity and identification more than Established study which add the tracing.

Established study is guaranteed the anonymity into group, but reality evil intention signer's identification make check on for every signature isn't guaranteed the anonymity. In other side offered study only retune true or false value of evil intention signer's identification so it isn't necessary for every signature to make check on..

I. 서론

정보보호 시스템과 정보통신 시스템의 발달과 확산은 정보화 사회를 앞당기고 있으며, 정보화 사회 구현에 필요한 자원의 효율적인 이용과 편리함 그리고 유익성 등 순기능 요소들과 더불어 허가 받지 않은 불법 사용자에 의한 정보의 도청 및 변조, 바이러스, 해킹, 개인신상의 비밀 누설 및 유출, 산업스파이에 의한 기업의 연구 기술 정보의 도난, 국가기관의 비밀 정보의 유출 등은 정보를 취급하면서 나타나는 역기능이다.[1]

정보보호에 대한 위협은 인터넷의 증가와 개방형 시스템의 네트워크의 발달과 더불어 증가하였고 이로 인한 보안의 필요성이 증가하게 되었다. 정보보호시스템 위협에 대한 정보보호의 대책은 법적, 제도적, 관리적, 기술적인 정보보호 대책이 상호 보완적이고 체계적으로 이루어져야 한다. 특히 사용자 정보보호에서는 접근통제, 인증 및 식별, 침입탐지 기술, 암호 및 키 관리, 바이러스 및 해킹방지에 역점을 두어야 한다. 이런 정보보호 서비스와 정보보호 기술의 관계는 하나의 정보보호 서비스를 위해 여러 가지의 정보보호 기술이 제공되고, 하나의 정보보호 기술로 여러 가지의 정보보호 서비스가 제공된다. 즉 정보보호 시스템을 구현하기 위한 기술은 서비스와 호환되어 사용된다.

분산 데이터 처리 기술은 분산 처리 환경이 단일 처리 시스템의 동작 정지가 전체 시스템에 미치는 영향을 줄여 신뢰성을 높일 수 있다. 또 여러 시스템을 동시에 사용할 수 있어 한 개의 대형 시스템을 순서대로 사용하는 것보다 저렴한 비용으로 나은 성능을 얻을 수 있다. 중앙 집중 데이터 처리 기술과 모든 자료를 파일시스템으로 관리하였다. 중앙 집중 데이터 처리 기술은 중앙에 모든 데이터의 처리가 집중되기 때문에 병목현상이 발생하고 시스템

구축에 큰 비용이 들었다. 또 파일 시스템은 시스템이나 응용프로그램에 종속적이어서 다른 데서 재사용하는 것이 어렵게 된 구조였다. 따라서 이런 비효율성을 없애고, 자료의 유동성을 높이려는 시도에 따라 최근에는 분산 데이터 처리 기술에 좀 더 적극적인 데이터베이스화를 생각하게 되었고, 이러한 데이터베이스 시스템은 응용프로그램이나 하드웨어와는 독립적으로 운영된다는 큰 장점이 있어서 기업들은 많은 비용을 투자하여 분산 데이터베이스 시스템을 구축하게 되었다.

분산처리기술은 분산처리환경을 구성하는 단일 시스템의 동작 정지가 전체 시스템에 미치는 영향을 적게 함으로써 신뢰도를 높일 수 있고, 여러 시스템을 동시에 활용할 수 있으므로 한 개의 대형 시스템을 활용하는 것보다 저렴한 비용으로 보다 나은 성능을 얻을 수 있다. 또한 시스템 확장이 필요한 경우 단순히 새로운 시스템을 분산 처리 환경에 추가하고 분산 처리 환경을 간단히 재구성함으로써 쉽게 목적을 달성할 수 있다. 1990년대 후반에 이르러서 떠오르는 분산 객체 처리 기술은 분산 처리 기술에 객체지향개념을 도입하여 분산된 자원들을 객체화 시켜 객체들 사이의 상호 운용성을 제공 하는 것으로 분산 처리의 장점을 상속, 다양화, 캡슐화, 재사용 등의 객체 지향 기술의 장점을 동시에 지닌 새로운 기술로 주목 받고 있다.

그러나 분산 환경은 데이터의 이동이 빈번하기 때문에 보안상의 문제점을 가지고 있다. 즉 허가되지 않은 정보의 유출, 합법적인 사용자로의 위장, 우회되는 보안 통제, 통신망 상의 자료의 불법적인 도청, 객체들 사이에 전송되는 자료의 변조 등이 있다. 응용 서비스들은 위의 문제점들로부터 보호되어야만 신뢰성 있고, 서비스들이 사용가능한지를 제고할 수 있다. 특히 금융 서비스 등과 같은 보안에 민감한 서비스의 경우에는 더욱 그러하다.

이와 같은 환경에서는 저장된 데이터의 보호를 위한 암호와 전달되는

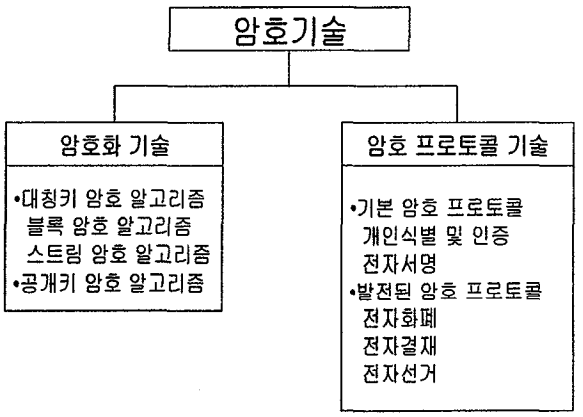
메시지의 인증과 무결성이 정보보호에 있어 필수적으로 요구된다.

암호란 1970년 중반, 미국의 민간분야에서 사용할 암호의 표준을 마련하였다. DES(Data Encryption Standard)로 불리는 표준 암호의 제정은 대학이나 연구소 그리고 정부를 중심으로 암호기술 연구에 박차를 가하게 되었고 현대 암호의 시발점이 되었다.

현대암호의 가장 큰 발전은 1976년에 발표한 공개키 암호(Public Key cryptography) 이다. 고도의 수학을 암호에 접목함으로써 탄생된 공개키 암호는 정보시스템 분야와 네트워크분야 인증분야 등 다양한 보안관련 서비스를 구현하는데 기초가 되었고, 곧바로 RSA공개키 암호, ElGamal 공개키 암호 등이 발표되었다. 공개키 암호의 표준이 발표된 이후 가장 두드러진 특징은 디지털 서명(Digital Signature)이다. 종이에 펜으로 직접 서명하는 대신 디지털 서명은 무결성, 개인식별 및 키 분배의 편리성 등으로 비약적으로 발전하여 우리의 일상생활에 적용된 기술이라고 할 수 있다.

1991년 RSA공개키 암호에 기반을 둔 디지털서명이 국제표준으로 제정되었고, 1994년 미국정부 주도하에 ElGamal 공개키 암호를 이용한 DSS(Digital Signature Standard)가 표준으로 제정되면서 우리의 생활에는 정부기관에서 발급하는 모든 민원서류와 금융거래, 증권거래 등을 공인인증서만 있으면 가능하게 되었다. 과거의 개인 인증은 인감증명서에 의해서만 가능하였다. 그러나 이제는 인감증명서도 공인인증서(전자서명)로 가능하게 되었다.

넓은 의미의 암호학(Cryptology)은 평문(Plain text)을 보호하기 위한 암호학(Cryptography)과 평문을 해독하기 위해 암호화 과정, 암호문을 연구하는 암호해독학(Cryptanalysis)으로 구분된다.



[그림 1] 암호기술의 분류

암호기술은 [그림 1]과 같이 암호화 기술과 암호 프로토콜 기술로 구분된다. 암호화 기술은 평문을 해독 불가능한 형태로 변형하거나 또는 생성된 암호문으로부터 해독 가능한 형태로 변환하는 원리, 수단, 방법 등을 제공하여 불법도청을 방지하기 위하여 사용한다. 암호화 기술은 사용하는 키의 종류에 따라 암호화키와 복호화키가 같은 대칭키 알고리즘과 암호화키와 복호화키가 다른 공개키 암호 알고리즘으로 구분되며, 대칭키 알고리즘은 변화하는 방법에 따라 블록 암호 알고리즘과 스트림 암호 알고리즘으로 구분된다.

암호 프로토콜 기술은 고도의 정보화 사회에서 요구되는 인증(Authorization) 및 무결성 문제를 해결하는 유일한 절차이다. 대표적인 암호 프로토콜에는 네트워크상에서 상대방의 신분을 확인하는 개인식별 및 인증기술, 현재의 도장이나 서명을 정보화 시대에 적합하게 진화된 전자서명기술 등이 있으며, 현재 많은 관심으로 개발되어 활용되고 있는 전자결재 및 전자화폐도 암호 프로토콜 기술의 응용 분야이다.

디지털 컨텐츠의 보호를 위한 일반적인 방법은 그 내용을 암호화하여 저장이나 전송한 후, 권한을 가진 자만이 해독하여 활용할 수 있도록 하는

것이다. 암호화 방법은 크게 대칭키 방식과 비대칭키 방식으로 나눌 수 있으며, 대칭키 방식으로는 비밀키를 사용하는 DES 알고리즘을 들 수 있으며 비대칭키 방식은 공개키와 개인키를 사용하는 RSA 알고리즘을 대표적으로 들 수 있다. DES는 암 · 복호화 속도가 빠른 반면 키의 관리가 어려운 점이 있고, RSA는 암 · 복호화 속도가 느린 반면 키 관리가 편리한 점이 있다.[3][4][9]

중요 정보의 무결성 확인과 디지털 서명(Digital Signature)의 효율성 증대가 중시되는데 디지털 서명은 해쉬 함수(Hash Function)로부터 만들 수 있다.[18] 해쉬 함수는 임의의 길이의 비트스트링을 입력으로 받아 고정된 짧은 길이(주로 120, 160비트)의 비트스트링으로 출력하는 함수이다. 해쉬 함수는 크게 비밀키의 유무에 따라 메시지 인증 코드(Message Authentication Code: MAC)와 조작 감지 코드(Manipulation Detection Code: MDC)로 나뉘며 MDC는 키를 이용하지 않는 해쉬 함수로써 일방향 해쉬 함수(One-way Hash Function)와 충돌 회피 해쉬 함수(Collision Resistant Hash Function)로 나누어진다.

디지털 서명은 우리가 주변에서 어떤 서비스를 이용하기 위해 서비스 제공자에게 인증 받는 절차에 필요하다. 서비스 제공자에게서 서비스를 받는 자를 고객이라 하자. 고객이 손쉽게 인증을 받기 위해서 인증 받는 절차를 자동화 시킬 수 있는 카드를 발급했다고 하자. 그 카드의 종류는 크게 두 가지로 나눌 수 있다.[2]

a. 고객의 신원이 포함된 인증 카드

b. 고객의 신원이 포함되지 않는 인증 카드

첫 번째의 고객의 신원이 포함된 인증 카드는 인증 요구 시 서비스 제공자가 고객의 신원을 파악할 수 있다. 그러나 서비스 요구마다 자신의 신원이 파악된다면 어떻게 생각하는가? 이는 사생활 보호의 관점에서 그리 좋다고 생각할 수 없다.

두 번째의 고객의 신원이 포함되지 않는 인증 카드는 첫 번째 경우의 보완 카드이다. 하지만 어떤 상황에서는 고객의 신원이 필요한 경우가 있다. 예를 들어 범죄가 연루된 경우는 사생활 보호보다는 신원 파악이 우선시 될 수 있다.

기존의 인증 방법은 고객의 신원이 완전히 공개되던가 전혀 공개되지 않는 양 극단 중의 하나만 선택함으로써 다양한 경우에 유연히 대처하지 못했다. 이때 사용하는 방법이 그룹 디지털 서명이다.

그룹 디지털 서명은 서명을 통해 서명자가 그룹의 회원인 사실만 확인 할 수 있고, 서명자의 정확한 신원은 알 수 없는 디지털 서명 방식이다. 법적인 분쟁이 일어날 경우에는 그룹관리자(Group Manager)가 해당 서명을 열어봐서(Open), 서명자의 신원을 확인할 수 있다. 이 특징을 보면 위의 예처럼 고객의 신원 보호보다 신원 확인이 우선시 되는 경우에 유연하게 대처가 가능하다. 이렇게 기본적인 개념이 참신함에도 불구하고 처음 제안할 당시의 그룹 서명 방법들은 서명의 크기가 그룹 멤버의 수에 비례해서 증가하여 매우 비효율적이었다.

따라서 실제적이고 효율적인 그룹 디지털 서명 기법에 대한 연구가 활발히 진행되었으며, 2000년 Atenienese, Camenisch, Joye, Tsudik이 최초로 확장성있고 (Scalable), 안정성이 증명된 방법을 소개하였다.[35] 최근에 Boneh 과 Boyen에 의해서 제안된 random oracles 없는 짧은 서명인 방법은 기존의 것보다 유연한 인증 방법을 제공할 수 있게 SDH(Strong Diffie-Hellman)가설을 사용했다. [19] 이것은 Mitsunari등에 의해 사용되어진 가설과 밀접한 관련이 있다.[36] SDH가설은 Strong-RSA가설과 매우 흡사한 성질을 가지고 있다. [19]에서 제안한 결과를 보면 SDH에 기반을 둔 시스템이 Strong-RSA 대응물이 보다 간략하고 짧다는 것을 알 수 있다. 이 제안한 방법은 SDH문제로 알려진 새로운 영지식 증명을 (new Zero-Knowledge Proof of Knowledge (ZKPK)) 기반으로 하고 있다. 본 고에서

는 이 SDH가설을 기반으로 개인 식별에 중심을 둔 그룹 디지털 서명에 대해 소개하겠다.

II. 정보 보안 방법

A. 접근통제 기술

접근권한이란 사용자들이 디렉토리나 파일과 같은 특정 객체에 영향을 줄 수 있음을 의미한다. 예를 들어 우리가 일상적으로 로그인 옵션에서 사용자를 선택하거나 FTP같은 공유 프로그램에서 권한을 설정했던 것을 생각하면 된다. 이러한 접근을 컴퓨터의 운영체계에 알리기 위해 설정해 놓은 목록을 접근제어목록(ACL(Access Control List))라 한다. 모든 객체는 ACL을 식별할 수 있는 보안 속성을 가지며 그 목록은 접근권한을 가진 사용자들을 위한 속성을 가진다. 가장 일반적인 권한은 1개의 파일이나 1개의 디렉토리 안에 있는 모든 파일들을 읽고(Read), 쓰고(Write), 수정하고(Modify), 삭제(Delete), 실행(Execute)하는 권한 등을 포함한다. 이런 권한에 의해 사용자의 접근이 제한되는 것을 접근 통제라 한다.

접근통제정책은 접근권한이 자원의 소유자에 의해서 주어지느냐 또는 시스템 관리자에 의해서 주어지느냐에 따라 임의적 접근 통제와 강제적 접근 통제 정책으로 나누어진다.

임의적 접근통제는 Subject나 그들이 소속되어 있는 그룹들의 ID에 근거하여 Object에 대한 접근을 제한하는 방식이다. 즉, 접근 통제는 Object의 소유자에 의하여 이루어진다. 따라서 어떠한 접근 허가를 가지고 있는 Subject는 임의의 다른 Subject에게 자신의 허가를 넘겨줄 수 있다. 강제적 접근 통제 정책은 Object에 포함된 정보의 비밀성과 이러한 비밀 정보에 대하여 주체가 갖는 정형화된 허가권에 근거하여 Object에 대한 접근을 제한하는 방법이다.

접근 통제 기술은 정보보호 시스템의 중요한 기능적 요구사항 중의 하나로 외부 사용자가 내부 네트워크로 또는 내부 사용자가 인터넷 등과 같은 외부 네트워크를 통해 정보보호 시스템에 접근할 때, 정보보호 시스템에서

접근권한을 부여하는지, 통신 대상이 되는 목적지 시스템에 대한 접근 권한이 있는지 검사하여 허용여부를 결정한다. 따라서 정보보호 시스템의 특정 자원에 대한 접근 자격이 있는 지를 검사한 후 접근여부를 경정함으로써 불법적인 자원 접근 및 파괴를 방지 할 수 있다.

1. 접근통제의 기본원칙

접근통제는 접근 통제 규칙에 의해서 이루어지며, 또한 접근 통제 규칙은 정보보호 시스템의 보안정책에 의해 결정이 된다. 정보보호정책의 수립이 올바로 이루어져야만 궁극적으로 정보를 보호하고자 하는 자원에 대한 접근 통제가 바로 이루어질 수 있다. 정보보호정책이 수립이 되면 정책에 따라 접근 통제 규칙이 설정되어야 한다. 시스템이 요정된 접근을 허가할 것인지 통제할 것인지를 결정하는데 사용하는 판단기준은 다음과 같다.[표 1]

[표 1] 접근통제 판단 기준

판 단 기 준		내 용
계 정		시스템 접근은 신분확인 및 인증과정을 통하여 통제된다
서비스제약		어떤 응용 프로그램을 사용하는 동안에 발생되거나 자원의 소유자, 관리자에 의해 미리 정해진 제한 사항이다
접근양식	Subject	대상에 접근할 수 있는 것. 즉, 개인 사용자, 사용자 그룹, 호스트, 단말기, 응용프로그램 등을 포함한다.
	Object	접근이 통제되어야 하는 개체를 의미하며, 데이터베이스, 파일, 프로그램 등을 포함한다.

	접근권한	주체가 가지는 대상에 접근할 수 있는 R, W, D, E 등의 형태를 포함한다.
위 치		물리적, 논리적 위치에 근거하여 시스템의 특정 자원에 대한 접근이 결정된다.
시 간		어느 특정 시간에는 접근을 제한한다.

2. 접근통제 정책

접근통제를 위한 정보보호정책은 누가(Who), 언제(When), 어디서(Where), 무엇(What)에 대하여, 어떤(How) 작업을 하도록 허용할 것인지에 대한 원칙을 정의하는 것이다. 일반적인 접근통제의 정책은 최소권한정책(minimum privilege policy)과 최대권한정책(maximum privilege policy)으로 나눌 수 있다. 최소권한정책은 시스템 주체가 작업을 진행함에 있어 최소한의 필요정보만을 사용하는 것이며, 최대권한정책은 시스템과 사용자간의 특별한 정보보호가 필요치 않는 환경에서 상호간의 자료의 공유를 증대시키기 위한 방안이다.

a. 신분-기반정책

신분-기반정책은 개인기반정책과 그룹기반정책을 기반으로 시스템내의 객체 접근을 제한하는 방법으로, 사용자가 접근을 요청하였을 때 사용자의 식별을 기초로 객체에 대한 접근권한을 추가하거나 철회할 수 있다.

개인기반정책은 각 사용자에 대하여 시스템내의 객체에 대한 접근허가를 부여하는 방식이다. 예를 들어 "객체 A에 대해 사용자 x의 읽기 권한을 부여한다."라는 정의가 있을 때 사용자 a는 오직 읽기만 할 수 있음을 나타낸다.

그룹기반정책은 다수의 사용자를 하나의 그룹으로 묶어 어떤 객체에 대해 동일한 접근허가를 부여하는 방식이다. 만일 사용자가 y1과 y2가 그룹 Y로

구성되어 있다고 할 때, "객체 A에 대해 사용자 x에만 읽기 권한을 부여하고 그룹 Y에는 읽기, 쓰기권한을 부여한다."라는 정의한다. 그렇다면 사용자 x는 읽기만 할 수 있고, 그룹 Y는 읽기, 쓰기가 가능함을 나타낸다.

이런 방법으로 객체에 대한 사용자의 접근여부를 접근허가문장의 수정 없이 그룹으로 변경하여 수정할 수 있다. 따라서, 개별적인 사용자의 허가대신 그룹으로 묶어서 표현하는 방법이 더욱 효율적인 정책 표현을 할 수 있다.

b. 규칙-기반정책

규칙-기반정책은 객체 내에 포함된 정보의 정보보호등급과 객체에 접근하는 사용자의 접근허가를 근거하여 객체의 접근을 제한하는 방법이다. 객체의 정보보호등급은 객체가 포함하고 있는 정보의 허용등급과 객체 정보가 포괄하는 응용분야의 범주(Category)로 구성된다.

정보의 허용등급을 할당하여 운영되며 사용자는 접근허가를 부여 받고 객체에 대한 접근을 허용 받는 정책을 다단계 정보보호 정책이라 부른다. [표 2]

[표 2] 허용등급

레벨	등급
4	Top Secret
3	Secret
2	Confidential
1	Restricted
0	Unclassified

다단계 정보보호정책은 사용자와 객체 사이의 정보보호 수준 사이의 관계를 정의하며 이 명시규칙에서 read-only 접근의 규칙은 사용자의 접근허가의 수준이 같거나 또는 하위수준의 허용등급을 갖는 정보만을 읽을 수 있도록 허용하고, write-only 접근은 사용자의 접근수준이 같거나 상위수준의 허용등급을 갖는 객체에 정보를 기록 할 수 있도록 한다.

정보가 포괄하는 응용분야의 범주에 접근하기 위해서는 그 범주에 대한 접근허가를 보유해야 한다. 만일 구역 A, B 가 존재할 때 A구역의 접근허가가 B구역에 대한 동일한 수준의 접근허가를 의미하지는 않는다. 따라서 A구역의 접근허가가 있는 사용자가 B구역의 객체를 허가 없이 사용하기 위해서는 두 구역 내에 연결된 요청이 있어야 한다.

3. 접근통제 기법

접근통제는 실제의 어떤 시스템에 있는 물리적 실체, OSI계층의 실체, 논리적 실체 그리고 일반적인 형태의 실체들과 관련된다. 접근통제를 위한 모델에서 능동적인 실체의 집합을 주체(subject)라 하고, 수동의 자원의 집합을 객체(object)라고 부른다. 이러한 주체와 객체는 접근 통제 매트릭스(ACM(Access Control Matrix))를 통하여 결정된다.[표 3]

[표 3] 접근통제 매트릭스(ACL)

객체				
주 체	File 1	File 2	File 3	비 고
사용자 1	R, W	E	-	R : Read
사용자 2	R, W, E	-	R	E : Execute
프로그램2	-	R	R, W	W : Write
프로그램2	R, W, E	R, E	R, W, E	

접근통제 기술은 ACM의 열을 포함하는 ACL(Access Control List), ACM의 행을 표현하는 CL(Capability List), 제어 대상에 레이블을 붙이는 SL(Security Label)을 기본적으로 생각할 수 있다. 그리고 이러한 3가지 정보를 종합적으로 생각하는 통합정보 기술, 각 파일에 접근통제를 위한 비트들을 부가하여 제어하는 Protection Bit(PB), 파일의 접근권한을 검증하기 위한 비밀번호 등의 기법이 있다.

a. ACL(Access Control List)

ACL이란 객체의 관점에서 ACM을 해석하는 것이다. ACL은 사용자들이 시스템내의 객체에 대하여 어떠한 행위를 할 수 있는지를 나타내는 목록이다[표 4]. 기본적인 ACL의 동작방식은 특정한 사용자 목록에 대해서 접근통제조건을 추가하거나 수행하는 방법으로 이루어진다.

[표 4] 객체 A에 대한 ACL

사용자	접근권한
x	Read
y1	Read, Write
y2	Read, Execute
z	Read, Write, Execute

ACL은 구분될 필요가 있는 사용자가 소수이거나 사용자분포가 안정적일 때 적합하다. 만일 사용자가 너무 많거나 분포가 유동적이라면 관리에 어려움이 발생한다.

b. CL(Capability List)

CL은 주체의 관점에서 ACM을 해석한 것이다. CL은 시스템의 객체에 명시된 규정방법에 따라 접근권한을 받은 주체가 객체에 대하여 접근하는 방식이다[표 5]. 이때 접근권한은 한 사용자에서 다른 사용자로 전달될 수 있으나 변경될 수 없고 또한 권한 없이 복제될 수 없는 특징을 지니고 있다. CL은 사용자의 접근허가 목록에 따라 생성된다.

[표 5] 사용자 x에 대한 CL

객체	접근권한
A	Read
B	Read, Execute
C	Read, Write, Execute

CL은 비교적 객체의 수가 적고, 주체와 근접한 곳에서 접근통제결정이 일어날 때 효율적으로 사용할 수 있다.

c. SL(Security Label)

SL은 사용자, 객체, 접근요청 또는 전송중인 접근통제정보에 부여된다. 주체의 환경에서 주체의 정보보호수준을 식별하는 레이블이 전체 접근요청에 부여되고 신뢰된 프로세스에 의해 생성 및 부여되어야 한다. 또한 모든 객체는 자신에게 부여된 정보보호 수준을 나타내는 레이블을 갖는다.

d. 정보보호 비트(Information Protection Bits)

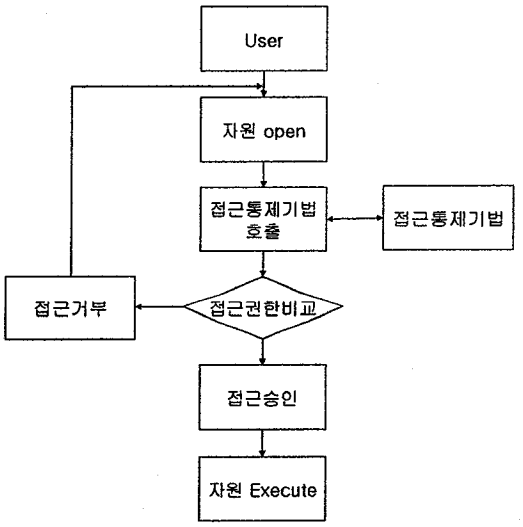
정보보호비트방식은 ACL방식을 수정한 형태로 각 객체에 대한 접근허가를 나타내는 비트들을 사용한다[표 6]. 따라서, ACL처럼 전체 사용자의 접근허용 리스트를 사용하지 않고 사용자의 종류에 따라 허가를 나타내는 비트들을 각 파일에 부가하는 방식이다.

[표 6] 정보보호비트(IPB)

OWNER					GROUP					WORLD				
R	W	E	D	L	R	W	E	D	L	R	W	E	D	L

e. 접근통제 설계

접근통제는 자원들에 대한 접근을 처리하는 과정의 한 부분으로 구현된다. 사용자 응용 프로그램은 자원을 접근할 수 있는 권한을 요청한다. 권한요청 과정에서 접근통제 기법이 호출되고, 접근통제 기법은 사용자가 요청한 접근 내용을 사용자가 가지고 있는 접근권한과 비교한다.



[그림 2] 접근통제

여기서 ACL 또는 다른 접근통제 방식을 사용하여 수행된다. 접근권한을 가지고 있으면 자원을 운용하고, 접근권한이 없으면 접근은 거절되고 에러 메시지를 만들어 보낸다[그림 2]

B. 인증 기술

사용자 인증이란 어떤 사람이 실제로 등록된 정당한 사용자임을 확인하는 과정이다. 개별 또는 인터넷을 포함한 통신 네트워크에서의 사용자 인증은 대개 로그인 시 암호의 사용으로 이루어진다. 암호를 알고 있는 사람은 일단 믿을 수 있는 사용자라고 생각한다. 그러나, 이것은 전자상거래 등이 수반되는 중요한 거래에서 인증의 단점은 암호가 도난 당하거나, 우연히 타인에게 알려지거나 또는 잊혀질 수 있다는 점을 간과하고 있다.

사용자 인증은 계정(ID, Password)을 암호화하고 복호화된 데이터베이스의 결과값을 비교하여 사용자를 확인하는 것을 전제로 한다. 이러한 인증은

일반적으로 두 가지 요소에 의해 보안을 유지한다. 키를 이용하는 방법과 ID를 이용한 방법이다. 사용자 인증은 대부분 접근제어나 사용자 의무를 규정하기 위한 기본적 사양으로서 보안의 중대한 요소이며, 방어를 위한 첫 번째 단계이다.

사용자 인증 절차는 사용자와 응용프로그램 또는 시스템 사이에 공유된 계정의 확인을 거친다. 분산 컴퓨팅 환경에서 사용자와 서버는 인증을 필요로 하는데, 사용자가 서버에 서비스를 요청할 때 서버로부터 인증을 받아야 하며, 경우에 따라서 서버도 사용자에게 인증을 받아야 한다. 이것을 Two-Party 인증이라고 하고 별도의 인증 서버를 두고 여기에서 모든 Password를 관리하며 인증하는 것을 Third-Party Authentication이라고 한다.[6][7]

C. 보안 기술

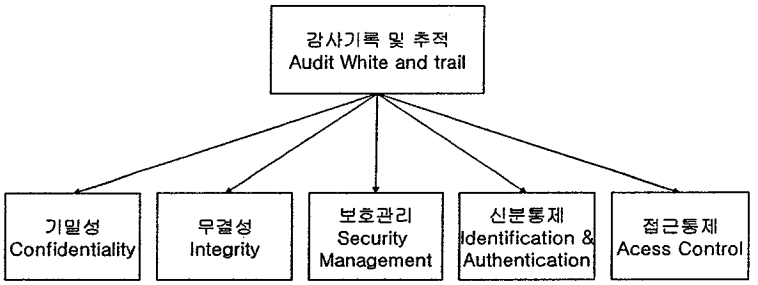
정보보안 시스템 구성요소 중 가장 취약한 요소가 바로 인간적인 요소이다. 분산 처리 시스템의 위협은 외부에서보다 내부사용자에 의한 내부적인 위협이 훨씬 더 크다. 내부 위협은 사용자의 실수 및 부주의에서 올 수 있으며, 불법적으로 장비나 정보를 외부로 유출하는 방법들이 있다. 그러므로 내부 사용자 관리가 시스템 보안에 큰 영향을 미친다는 것은 주지의 사실이다.

시스템의 설치, 운영, 백업 등의 정기적인 업무와 계획, 정책 수립 및 시스템의 안전 대책 등과 관련된 모든 일들이 인간에 의해 이루어지고 있다. 내부 사용자에 의한 정보시스템의 피해 사례가 날로 급증하고 피해 사건을 찾아내는 것 또한 매우 힘들다. 특히 정보시스템의 보안에 중요한 업무를 담당하고 있는 시스템 관리자, 운영자, 프로그래머에 의한 피해는 더욱 찾아내기가 힘들다. 이러한 정보보호 시스템에 대한 사용자의 보안문제는 당사자에게 보안 규정의 준수와 개인 시스템의 무결성 유지라는 권한과 책임을

부여함으로써 피해를 최소화 할 수 있다.

D. 감사 기술

감사기술이란 정보보호 시스템상의 정보보호관련 활동에 대해 기록하고, 기록된 자료를 분석하여 침입 예방 및 불법적인 행위를 추적하는 것이다. 감사기술의 기능들이 효율적으로 수행되기 위해서는 신분확인을 통한 주체의 식별자가 바르게 확인되어야 하고, 주체의 식별자에 따른 객체의 식별자, 접근통제, 무결성 및 기밀성 등의 정보보호관련 기술에서 감사대상이 되는 사건에 대한 정확한 감사정보를 기록 및 추적 기술에 전달되어야 한다. 또한 감사자료에 대한 접근통제도 함께 이루어져야 한다.



[그림 3] 감사 기술

1. 필요성

인터넷과 같은 외부망 또는 내부망으로부터 허가 받지 않은 사용자의 공격으로부터 사용자의 인증데이터가 고의로 또는 자신도 모르는 사이에 노출된 경우에는 아무리 정보보호 시스템의 신분확인 기능이 강력하고, 접근통제가 엄격히 이루어져도 보안상 허점을 갖는다.

이러한 경우 감사기록 및 추적 기술이 적용된 시스템이라면 신분확인 시

기록된 자료를 가지고 추적을 통해 불법 사용된 사용자의 식별자를 알아내서 적절한 조치를 취할 수 있다. 하지만, 감사추적을 통한 불법 사용자의 추적은 불법 사용된 사용자까지는 추적이 가능하지만 실제 사용자를 식별해내기는 매우 어렵다. 이런 방법으로 감사기술은 정보보호시스템의 정보보호기능을 하며, 추적을 통해 책임소재를 명확히 할 수 있다.

2. 감사기록 및 추적관리

감사기록 및 추적관리는 정보보호시스템과 연관된 데이터들에 대해 설정, 조회, 변경, 추가 및 삭제의 기능을 제공해 준다. 감사기록 및 추적관리는 감사기록 대상사건, 감사기록 파일, 침입감지 등 크게 세 가지로 나누어 볼 수 있다.

감사기록 대상사건은 모든 주체 및 객체와 관련되는 신분확인, 접근통제, 감사기록 및 추적, 정보보호관리의 정보보호관련 기술에 대한 모든 활동으로 감사기록 및 추적이 이루어지기 위해서는 먼저 감사기록대상에 대한 정의가 이루어지고, 개발자에 따라 감사기록대상사건은 기밀성 및 무결성 등과 같은 정보보호서비스를 추가적으로 정의한다.

감사기록 및 추적 요구사항에서는 감사기록 대상사건 발생시 생성된 감사 레코드에 대해 준 영구적인 보조기억장치에 저장한다. 관리자는 정보보호관리를 통해 보조기억장치에 저장된 감사기록 파일들에 대해 조회, 삭제 및 초기화를 할 수 있어야 한다. 또한 관리자는 감사기록 파일을 각 레코드 필드로 조건을 설정하여 선택적인 자동 요약 및 출력할 수 있어야 한다. 침입감지는 사용자의 현재활동과 정보보호시스템에 저장되어 있는 침입행위 리스트와 비교를 통해 침입을 예방하는 것이다.[7][8] 관리자는 과거에 알려진 단일사건의 침입행위와 일련의 사건들로 구성된 침입행위를 기반으로

정보보호관리시스템으로 침입행위 리스트를 설정, 변경, 추가 및 삭제한다.

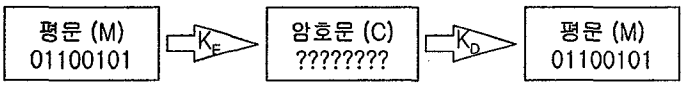
E. 암호 기술

1. 암호 시스템

암호화 되지 않은 상태의 평문을 암호문으로 만드는 과정(Encryption), 역으로 암호문을 평문으로 변환시키는 복호화 과정(Decryption), 암호화와 복호화 과정에서 사용되는 암호화 키, 키 관리 등 정보보호를 위한 일련의 과정을 암호 시스템이라고 한다.

평문을 M, 암호문을 C라 하면 암호화 과정은 함수 E로 간주하여 EK (M) = C라고 표시한다. 또한 복호화 과정은 함수 D로 간주하여 DK (C) = M으로 표시한다. 암호화 키는 일반적으로 K로 표시하며 키의 가능한 값의 범위를 키 공간 (Key Space)라고 한다.

키를 포함한 암호시스템은 [그림 4]와 같이 표현할 수 있다. 여기서 KE 는 암호화에 필요한 키이며, KD 는 복호화에 필요한 키이다. 키를 고려하는 암호화 및 복호화 과정은 각각 [그림 5]와 같이 함수로 표현할 수 있다



[그림 4] 암호시스템

[그림 5] 암호화 및 복호화 함수

a. 대칭키 암호 시스템

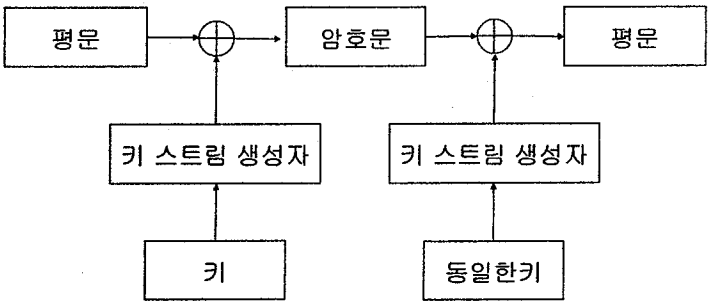
대칭키 암호시스템은 비밀키 암호 시스템이라고도 하며, 송·수신자가 동일한 키에 의하여 암 · 복호화 과정을 수행한다. 대칭키 암호 알고리즘은 변환하는 방법에 따라 블록 암호 알고리즘과 스트림 암호 알고리즘으로 구분한다.

대칭키 암호 시스템은 오랜 역사를 가지고 있고 이미 존재하는 많은 정보기술과 상호 운용이 쉽고 데이터 처리능력도 강력하다. 또한 시스템상에서 효율성을 좌우하는 암호화 키의 크기가 공개키 암호 시스템보다 작아서 효과적인 암호시스템을 구성할 수 있다. 그리고 알고리즘의 내부 구조가 간단한 치환과 순열의 조합으로 되어 있어서 시스템 환경에 맞는 적절한 암호 알고리즘을 쉽게 개발할 수 있다. 그러나 정보 교환 당사자 간에 동일한 키를 공유하여야 하므로 여러 사람과의 정보 교환 시 많은 키를 유지 관리하여야 하는 어려움이 존재한다.[3][4][9][10][11]

블록 암호 시스템은 고정된 크기의 입력 블록을 고정된 크기의 출력블록으로 변형하는 암호 알고리즘에 의해 암호화 및 복호화 과정을 수행한다. 대표적인 블록 암호 알고리즘은 DES, Triple-DES, Skipjack, IDEA(International Data Encryption Algorithm), SPEED(Secure Package for Encrypting Electronic Data), FEAL(Fast Data Encipherment Algorithm)이 있다.

스트림 암호 알고리즘은 이진화된 평문과 키 이진 수열을 배타적 논리합(XOR)으로 이진연산 결합하여 암호문을 생성하는 것을 말한다. 1970년대 유럽에서부터 발달한 스트림 암호는 [그림 6]와 같이 평문을 연속된 이진표현으로 하고 키 역시 이진 수열로 표현하여 두 이진수를 결합함으로써 암호문을 얻는다. 배타적 논리합 연산에서는 덧셈, 뺄셈이 같다는 특성으로 복호화 할 경우도 키의 결합을 풀기 위하여 동일한 키의 배타적 논리합으로 다시 평문을 얻는다.

따라서 스트림 암호 시스템은 어떠한 키 이진 수열을 발생하여 평문과 결합시키느냐가 암호 시스템의 안전도에 직접적인 영향을 미친다. 즉, 키 스트림이라고 불리는 키 이진 수열의 특성과 발생방법이 스트림 암호 시스템의 핵심이 된다. 키 스트림이 주기를 갖고 반복될 때 이를 주기적(Periodic)스트림 암호시스템이라고 하며, 키가 반복 없이 표현되는 일회용 패드(One-time pad)방식인 것을 비주기적 스트림 암호시스템이라고 한다.



[그림 6] 스트림 암호 시스템 개념도

따라서, 스트림 암호 시스템에서는 어떠한 키 이진 수열을 발생하여 평문과 결합시키느냐가 암호 시스템의 안전도에 영향을 미친다. 즉, 키 스트림이라고 하는 키 이진 수열의 특성과 발생방법이 스트림 암호 시스템의 핵심이 된다.

또한, 스트림 암호시스템은 평문과 키 스트림과의 관계에 따라 동기식 스트림 암호시스템과 자기 동기식 스트림 암호시스템으로 분류된다. 동기식 스트림 암호 시스템에 자기 동기식 스트림 암호시스템의 장점을 결합하여 사용할 수도 있다. 자기 동기식 스트림 암호 시스템은 DES와 같은 블록 암호 시스템에 비하여 오류 전파가 없다는 우수한 장점이 있는 반면에 블록 암호 시스템에서는 필요로 하지 않는 초기 값을 설정하여야 하는 단점이 있다.

그럼에도 불구하고 스트림 암호 시스템은 군사 및 외교용으로 널리 사용되고 있으며, 일부 사용으로도 활발히 사용되고 있다. 또한 스트림 암호 시스템은 이동 통신 환경에서 구현이 용이하고 안전성을 수학적으로 엄밀하게 분석할 수 있다는 장점 등으로 인하여 이동 통신 등의 무선 통신 데이터 보호에 적합하다.

b. 공개키 암호 시스템

공개키 암호 시스템은 암호화 키와 복호화 키가 서로 다르고, 하나를 알고 있어도 그에 대칭되는 키를 알기 어려운 암호 시스템을 말한다. 키 생성 알고리즘을 통해 두 개의 키를 생성하여 그 중 하나를 공개하고 나머지 하나를 개인키로 하여 자신이 보관하여 사용하는 것이다. 사용자 x가 사용자 y와 통신할 때 x는 y의 공개키를 이용하여 평문을 암호화해서 보내면 y는 수신한 암호문을 자신의 개인키로 풀어 평문을 확인 할 수 있다.[3][4][9][12]

따라서 공개키 암호 시스템의 사용자는 오직 자신의 개인키 만 보관하면 된다. 즉, 큰 네트워크 상에서 대칭키 암호 시스템보다는 훨씬 적은 수의 키로도 유지될 수 있다. 그러나, 암호화를 위한 키의 크기는 상대적으로 크기 때문에 데이터 처리량이 적고, 컴퓨터 부하가 많아진다. [표 7]는 대칭키 암호 시스템과 공개키 암호 시스템의 장점과 단점을 비교한 것이다.

[표 7] 대칭키 암호시스템과 공개키 암호 시스템의 비교

구분	대칭키 암호 시스템	공개키 암호 시스템
장점	•암 · 복호화 속도가 빠르다 •키의 길이가 짧다	•키의 분배가 용이하다 •사용자의 증가에 비해 관리하는 키의 수는 상대적으로 적다 •키 변화의 빈도가 적다 •많은 분야의 응용이 가능하다
단점	•키의 분배가 어렵다 •사용자의 증가에 따라 관리하는 키의 개수가 많다 •키 변화 빈도가 많다. •응용분야에 제한적이다.	•암 · 복호화 속도가 느리다 •키의 길이가 길다

2. RSA 알고리즘

RSA는 두 소수 p와 q의 곱은 계산하기 쉬우나 주어진 곱 $n=p\times q$로부터 p와 q를 추출하기 어렵다는 소인수 분해의 어려움에 근간을 두고 있다. RSA 암호 시스템은 모듈러 지수 연산이 주를 이룬다. 실제 응용에 있어서 작은 암호화 수를 사용함으로써 복호화보다는 암호화를 더 빠르게 수행할 수 있도록 설계되고 있으며, 서명보다는 검증을 더 빠르게 수행할 수 있도록 설계되고 있다. ,RSA 암호 시스템은 선택된 2개의 임의의 소수를 암호에 사용하므로 서로 다른 소수를 사용할 수 있는가가 주요한 문제이다.[13][14][15][16] RSA 암호 방식은 대칭키 암호 방식에 비하여 상대적으로 느리기 때문에 RSA 알고리즘을 대칭키 암호화에 사용된 세션키를 암호화하는데 사용하는 Hybird 암호방식이 주로 사용되고 있다.

RSA 공개키 암호에 사용될 공개키 $k_e = \{n, e\}$ 와 개인키 $k_d = \{d\}$ 를 생성하기 위하여 다음 과정을 수행한다.

- 두 개의 큰 소수 p, q를 선정한 후 법 $n = p \times q$와 $\varphi(n)$를 계산
- 공개키 e는 $\varphi(n) = (p-1)\cdot(q-1)$과 서로 소의 관계가 되게 임의 지정
- $e \cdot d \bmod \varphi(n) = 1$의 관계에 있는 개인키 d를 구함
- 공개키 $\{n, e\}$는 공개하고 개인키 $\{d\}$는 안전하게 보관

RSA 암호화를 수행하기 위해서 암호화할 메시지를 지니고 있는 송신자는 수신자의 공개키 $\{n, e\}$를 이용하여 암호화할 평문 m을 정수로 전환한 후 암호문 c로 암호화한다.

- 수신자의 공개키 $\{n, e\}$를 얻음
- 메시지 m을 $[0, n-1]$사이의 정수로 전환
- $c = E_{ke}(m) = m^2 \bmod n$를 수행하여 암호문 c를 얻음
- 암호문 c를 수신자에게 전송

암호문 c의 수신자는 자신의 개인키 $\{d\}$를 이용한 복호화작업을 통해서 원래의 평문 m을 복원하게 된다.

- 수신자는 개인키 $\{d\}$를 이용하여 암호문 c를 평문 m으로 복원한다.
- $D_{ke}(c) = c^d \bmod n = (m^e)^d \bmod n = m$

3. Deffie-Hellman 분배(Distribution) 알고리즘

Deffie-Hellman 공개키 분배(Public-key Distribution) 알고리즘은 두 개인간 또는 두 시스템간에 공통된 비밀키를 공유하기 위한 목적으로 사용된다. 이들의 공개키 분배 시스템은 이산 대수 문제의 어려움에 기반을 두고 있다.

- 각 사용자들의 공개키와 개인키를 도출하는데 매우 큰 소수 p와 위수가 $p-1$인 곱셈 상의 군 Z_p^*의 생성자 g를 요구
- Alice와 Bob은 임의의 난수 x_A와 x_B를 생성하여 자신의 개인키로 보관하고, 각각의 공개키 $y_A = g^{x_A} \bmod p$ 와 $y_B = g^{x_B} \bmod p$ 를 계산하여 상호 교환함
- Alice와 Bob의 공개키 y_B에 자신의 개인키 x_A를 적용하여 공통된 비밀키 $k = (y_B)^{x_A} \bmod p = g^{x_B x_A} \bmod p = g^{x_A x_B} \bmod p$를 구할 수 있음.

(cf) 공격자 Alice와 Bob간에 교환되는 공개키 y_A와 y_B를 도청하여 y_A와 y_B로부터 각각 x_A와 x_B를 도출해 낼 수 있다면 이러한 공개키 분배의 안정성은 깨지게 된다. → 그러나 공격자의 이산대수 문제를 풀어야 하기 때문에 불가능하다.

F. 해쉬 함수(Hash Function)

통신 네트워크에서 수신자는 자신이 받은 메시지가 전송되는 과정에서 제3자에 의해 의도적으로 변조되지 않았다는 확신을 가질 수 있게 하는 메시지 무결성(Integrity)과 실제 작성자를 확인하는 메시지 인증(Message Authentication)을 요구한다.[17][18]

원칙적으로 메시지 무결성과 인증은 분리해서 취급될 수 없는 개념으로 서로를 포함하는 관계에 있다. 수신자가 메시지를 공격자로부터 보호 받기 위해서는 메시지에 대한 암호화를 하거나 무결성을 유지해야 한다. 공격자의 수동적인 공격으로부터 메시지 내용을 은닉시키기 위한 방법은 암호화이다. 또 능동적인 공격으로부터 메시지를 보호하는 방법은 데이터 무결성이다. 여기서 능동적인 공격이란 메시지의 삽입(Insertion), 삭제(Deletion), 변조(Modification), 그리고 메시지 순서 변경(Reordering) 및 메시지 재생(Replay)등 이 있다.

메시지의 단순 암호화는 비밀키를 모르는 암호 공격자에 의한 불법적인 메시지의 삽입공격을 대부분 방지할 수 있지만, 그 이외의 능동적인 공격들은 암호공격자가 중간에 가로챈 이전의 메시지들을 이용하여 가할 수 있는 것들이기 때문에 단순히 메시지의 암호화만으로는 무결성의 보장이 완전하지 않기 때문이다. 무결성을 보장하기 위해서는 암호학적 해쉬 함수의 사용이 요구된다. 또 디지털 서명은 서로 신뢰할 수 있는 일반 상거래에서 계약서에 서명 날인하는 것과 유사한 개념이다.

1. 무결성과 해쉬 함수

암호화 기법을 사용할 경우에는 한글이나 영어와 같은 단순한 메시지의 경우에는 무결성과 인증을 보장할 수 있지만, 통계적 자료나 음성 및 영상 자료와 같은 2진 자료의 경우에는 그 무결성을 쉽게 파악할 수 없다. 또한 한글이나 영어로 된 메시지라 할지라도 그 메시지의 무결성을 수신측에서 자동적으로 인식할 수 있는 기능이 요구된다. 특별히 메시지에 대한 무결성이 요구되는 경우는 인증자(Authenticator), 또는 해쉬 값(Hash Value)으로 명명되는 특수한 형태의 데이터구조를 메시지로부터 압축, 생성하여 메시지에 첨가해야 한다. 또는 메시지 작성자에 대한 확인이 필요하다면 인증자를 생성하는 과정에 메시지 작성자와 수신자만이 알고 있는 비밀키(Secret key)가 포함되어야 한다. 이때 인증자를 생성하는 데 사용되는 함수를 암호학적 해쉬 함수(Cryptographic Hash Function)이라 한다.

해쉬 함수 h():의 길이가 유한한 임의의 메시지를 입력자료로 하여 고정된 길이의 비트 열(Bit Stream)을 출력하는 다대일(Many-to-One) 함수이기 때문에 필연적으로 서로 다른 입력자료에 대해서 동일한 해쉬 값이 출력되는 충돌현상(Collision)이 발생할 수 있다.

해쉬 함수는 전산학 분야에서 사용되는 용어로 임의의 길이의 문자나 숫자열을 고정된 길이의 값으로 전환하기 위해서 사용되는 함수로 파일의 레코드들을 가능한 균등하게 기억정치에 할당하기 위해서 사용하는 함수이다. 입력자료의 길이에 관계없이 고정된 길이의 값으로 전환시켜주는 해쉬 함수의 성질이 암호학 분야에서는 암호학적 해쉬 함수라 하며 인증 및 디지털 서명 분야에 응용이 된다.

a. 암호학적 해쉬 함수의 특징

암호학적 해쉬 함수 h()가 지녀야 하는 기본적인 제약조건은

- h()는 가변적인 길이의 임의의 입력자료 m을 고정길이의 해쉬 값 h(m)으로 압축하여 전환시킬 수 있어야 한다.
- h()와 m이 주어졌을 때, h(m)의 계산이 용이하여야 한다.
- 암호학적 해쉬 함수는 해쉬 값의 생성에 있어 비밀키의 사용여부에 따라 MAC와 MDC로 분류한다.

MAC 해쉬 함수는 입력자료 m과 비밀키 k가 해쉬 값 h(k, m)을 생성하는데 사용한다. 안전한 MAC 해쉬 함수가 되기 위한 조건은 0개 또는 1개 이상의 입력자료와 그에 해당하는 해쉬 값 $\{m_i, h(k, m_i)\}$들이 주어졌을 때 비밀키 k를 모르는 상황에서 임의의 입력자료 $m(\neq m')$에 대한 MAC 해쉬 값 h(k, m)을 구하는 것이 계산적으로 불가능해야 하고, 권고 길이는 64비트 이상이다.

MDC 해쉬 함수는 해쉬 값 h(m) 생성에 단지 입력 자료 m만 사용한다. MDC 해쉬 함수는 추가적으로 요구되는 조건에 따라 일방향 해쉬 함수(One-way Hash Function)[or 약한(week) 일방향 해쉬 함수]와 충돌 회피 해쉬 함수(Collision-free Hash Function)[or 강한(strong) 일방향 해쉬 함수]로 분류할 수 있다.

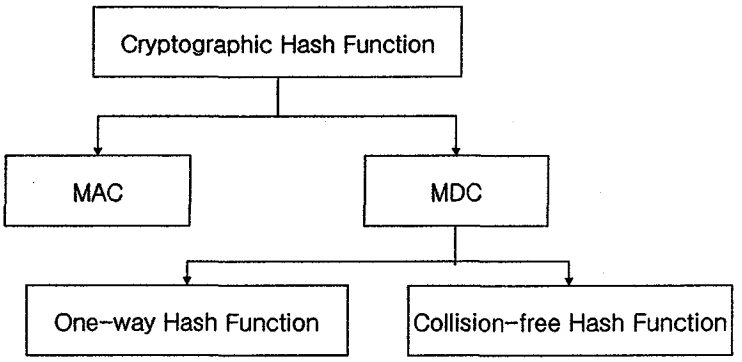
MDC 해쉬 함수는 메시지의 무결성 뿐만 아니라, 패스워드 확인, 세션 키 도출 및 디지털 서명 등의 여러 응용 분야에서도 폭 넓게 이용된다. 안전한 MDC 해쉬 함수가 되기 위한 추가 조건은

임의의 해쉬 값 h(m)이 주어졌을 때 그것에 해당하는 입력자료 m을 도출해 내는 것이 계산적으로 불가능해야 한다.

임의의 입력자료 m과 그에 해당하는 해쉬 값 h(m)이 주어졌을 때 h(m)=h(m')을 만족하는 다른 입력자료 m'(≠m)을 찾는 것은 역시 계산적으로 불가능해야 한다.

동일한 해쉬 값 h(m)=h(m')을 가지는 임의의 서로 다른 두 개의 입력자료 m과 m'을 찾는 것이 계산적으로 불가능해야 한다. 즉 충돌회피의 성질을 가지고 있어야 한다는 점이다.

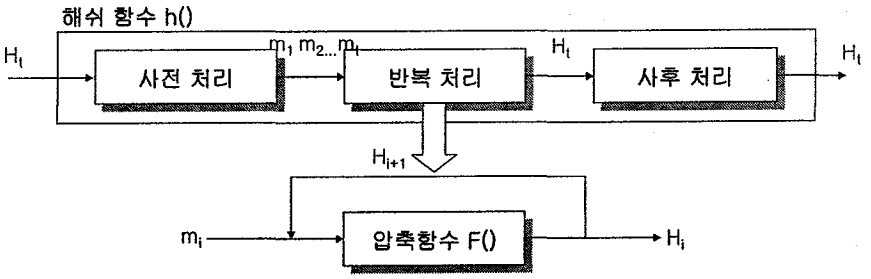
a와 b의 조건을 만족하는 MDC 해쉬 함수를 일방향 해쉬 함수라 하고, 권고 길이는 64-80비트 이상이다. 또, 위의 세가지 추가 조건을 모두 만족하는 MDC 해쉬 함수를 충돌회피 해쉬 함수라 하고, 권고 길이는 128-160비트 이상이다.



[그림 7] 암호학적 해쉬 함수의 분류

대부분의 일반적인 MDC 해쉬 함수들은 가변적인 길이의 메시지 m을

입력자료로 받아들여 고정된 길이의 t개의 블록들 m_1, m_2, ... m_t 로 나눈 다음 그 블록들을 연속으로 반복 처리(Iterated Processing)하여 해쉬 값 h(m)을 출력하도록 설계한다. 가변적인 길이의 메시지 m을 고정된 길이의 블록들로 나누는 사전 처리(Preprocessing)과정에 있어서 전체 메시지의 길이가 블록 길이의 정확한 배수가 되게 하기 위해서 채우기(Padding) 작업이 요구된다. 이때 실제 메시지의 길이와 채우기 작업을 통해서 첨가된 비트들의 길이를 구별하기 위해서 메시지의 실제 길이를 첨가하는 작업이 수반된다. 각각의 블록 m_i 는 반복처리 단계에서 내부 압축 함수(Compression Function) f()에 그 이전 블록 m_{i-1} 에 대해서 계산된 중간 결과 값 H_{I-1} 과 함께 입력되어 중간 결과 값 H_I 가 계산된다. 이러한 과정을 t번 반복하여 H_t 가 계산되고 사후처리(Post-processing)단계를 거쳐 최종적인 해쉬 값 h(m)이 생성된다.

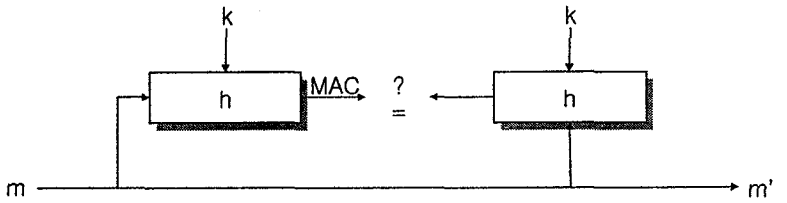


[그림 8] 해쉬 함수의 일반 모델

b. MAC을 이용한 무결성 보장

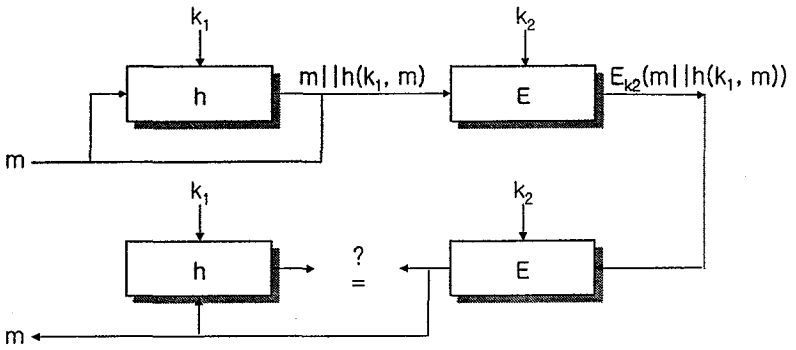
MAC은 1970년 대부터 출발한 현대 암호학보다도 더 오랜 역사를 가지고 있는 금융분야에서 이미 오래 전부터 사용되어 온 개념이다. 이 MAC를 이용한 무결성을 보장하는 방법은 첫째, MAC 해쉬 함수 h()는 메시지의 작성자와 수신자만이 알고 있는 비밀키 k를 이용하여 메시지 m에 대한 인증자 MAC=h(k,

m)을 계산한다. 둘째, 메시지와 함께 MAC이 수신자에게로 전달되면 수신자 측에서는 전달된 메시지를 이용하여 자기 자신이 계산한 MAC과 수신된 MAC을 비교함으로써 메시지에 대한 무결성을 확인한다. 마지막으로 송신자와 수신자만이 알고 있는 비밀키 k를 사용하고, 수신된 MAC과 계산된 MAC과 일치한다면 수신자는 그 메시지가 정당한 메시지 작성자로부터 온 것이며, 또한 전송 도중에 변조되지 않았다는 것을 확신한다.



[그림 9] MAC를 이용한 무결성 확인

메시지에 대한 무결성 만이 아니라, 기밀성도 동시에 요구되는 경우 메시지 m과 인증자 MAC = $h(k_1, m)$을 함께 암호화 할 수 있다.



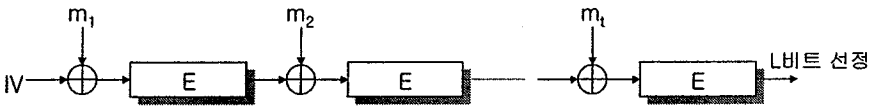
[그림 10] 메시지의 기밀성을 동반한 MAC의 무결성 확인

이때 m||h(k_1 , m)의 의미는 메시지 m과 MAC의 연결(Concatenation)을 의미한다. MAC의 생성에는 비밀키 k_1 을 적용하고, MAC을 포함한 메시지의 암호화에는 비밀키 k_2 를 사용한다. 물론 양쪽 모두 동일한 키를 사용할 수도 있다.

가장 대표적인 MAC은 CBC(Cipher Block Chaining)모드로 운영되는 DES에 기반을 두고 있다. 금융권에서 무결성의 목적으로 사용하기 위해 표준화된 MAC으로 구성한다. 메시지 m이 t개의 64비트들의 블록 m_1 m_2 ... m_t 로 구성된다면 MAC은 다음과 같이 계산된다.(이때 마지막 블록이 64비트가 안될 때는 부족한 만큼을 0으로 채운다.)

$$c_1 = E_k(IV \oplus m_1), c_2 = E_k(c_1 \oplus m_1), \ldots, c_t = E_k(c_{t-1} \oplus m_t)$$

여기서 초기화 벡터 IV는 '0'이고, 마지막 블록 c_t 중에서 $l(\leq 64)$ 비트를 MAC으로 선정한다.



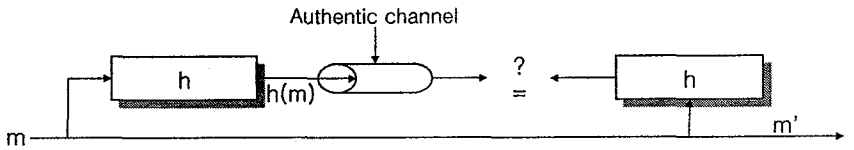
[그림 11] DES-CBC에 의한 MAC생성

c. MDC을 이용한 무결성 보장

메시지 작성자는 메시지 m에 MDC 해쉬 함수를 적용하여 해쉬 값 h(m)을 구한 후에 메시지 m은 기밀성 또는 무결성이 보장되지 않은 일반 채널(unsecured channel)을 통해서 수신자에게로 보내고 해쉬 값은 무결성이 보장되는 채널(9authentic channel)을 통해 보낸다.

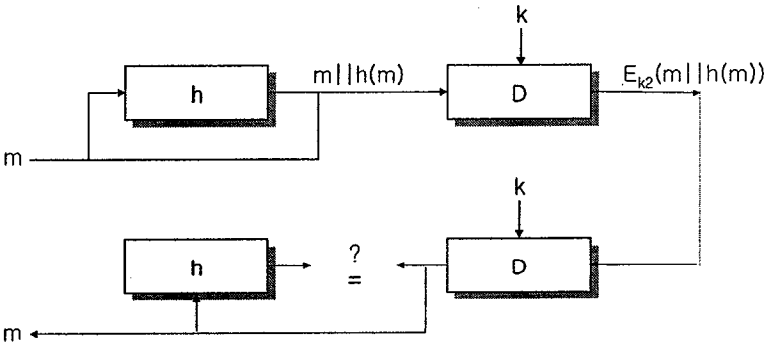
무결성이 보장되는 채널이란 첫째, 음성을 통한 상호간의 신빙성이 보장되는 전화, 등기우편과 같은 안전한 전달 매체 또는 출판물과 같이 위조가 불가능한 공공매체를 의미한다. 둘째, 디지털 서명 역시 무결성을 보장해 준다. 마지막으로 수신자는 수신된 메시지에 대해서 자신이 계산한 해쉬 값과 무결성이 보장되는 채널을 통해서 전달 받은 해쉬 값이 일치하는지를 비교해 봄으로서 메시지에 대한 무결성을 보장받는다.

바이러스로부터 소프트웨어를 보호하고 또는 무결성이 보장되지 않는 통신 네트워크를 통해서 소프트웨어나 공개키를 배포할 경우에는 무결성이 보장되는 채널과 함께 사용되는 MDC가 이용될 수 있다. 이때 컴퓨터 원시코드나 목적코드에 대한 바이러스 검사 시에는 MDC가 단순 암호화보다 선호된다.



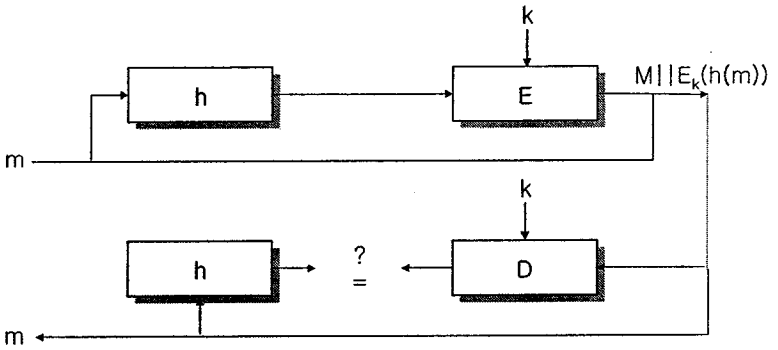
[그림 12] MDC 해쉬 함수를 이용한 무결성 확인

무결성과 더불어 메시지의 기밀성을 동시에 유지하기 위해 메시지 m과 MDC h(m)을 함께 암호화하는 방안이 있다.



[그림 13] 메시지의 기밀성을 동반한 무결성 확인

메시지의 기밀성이 요구되지 않을 경우에는 MDC만을 대칭형 암호로 암호화 할 수 있다.



[그림 14] MAC의 암호화를 통한 무결성 확인

참고로 각 메시지 블록마다 순번(sequence number)을 부착함으로써 메시지 블록들의 순서를 재배열(reordering)하는 암호 공격을 사후적으로 검출해 낼 수 있다. 그리고 송수신자간에 교환된 메시지를 관측하여 나중에 그 메시지를 다시 특정 수신자에게 보내는 메시지 재생(replay)공격은 단순 메시지 무결성

공격으로는 검출 될 수가 없고, 난수(random number)나 시각표(timestamp)를 메시지에 부착함으로써 검출이 가능하다.

d. 다양한 MDC 해쉬 함수

MDC 해쉬 함수는 블록암호에 기반을 두고 있는 해쉬 함수, 법 연산에 기반을 둔 해쉬 함수, 전용 소프트웨어 해쉬 함수로 분류할 수 있다.

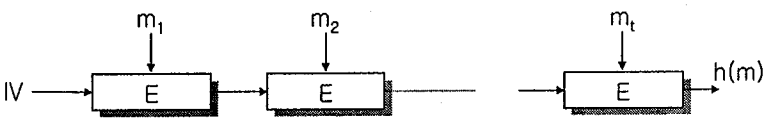
암호학적으로 안전한 해쉬 함수의 설계에 따른 노력을 최소화하기 위하여 대부분의 암호 설계자들은 DES와 같은 기존의 블록 암호에 기반을 둔 해쉬 함수의 특징은 메시지 또는 문서의 크기를 기반이 되는 블록암호의 입력 또는 키와 같은 크기의 블록으로 나눈 다음에 각각의 블록들에 대해 동일한 작업을 반복적으로 수행하여 해쉬 값을 구한다. 그리고 최초의 블록에는 초기화 벡터(Initialization vector)가 사용된다. 이 값은 공개하거나 비밀키로 유지할 수 있고, Rabin 방식과 Davies-Mejer 방식이 있다.

법(Modulus) 연산에 기반을 둔 해쉬 함수는 법 연산을 압축 함수(Compression Function)의 기본 구조로 이용한다. 이는 기존의 법 연산과 관련된 소프트웨어 또는 하드웨어를 그대로 이용하기 위해서 이다.

MD5, SHA(Secure Hash Algorithm)등의 전용 소프트웨어 해쉬 함수들은 블록 암호 또는 공개키 암호에서 이용된 구조물을 이용하지 않고, 또한 소프트웨어로 구현 시 고속으로 동작하게 설계된 해쉬 함수이다.

블록 암호에 기반을 둔 Rabin방식은 먼저 메시지를 블록 암호에 적용될 입력의 크기와 동일한 블록으로 나눈다. 기본 계산 알고리즘은

$$h_0 = IV$$
$$h_i = E_{mi}(h_{i-1}), \quad i = 1,2,3,\ldots,t$$
$$h(m) = h_t$$

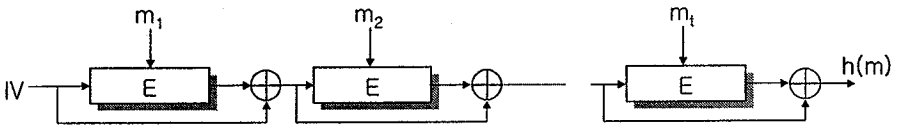


[그림 15] Rabin 방식

간단하고 효율적으로 보이지만, 해쉬 값이 $l = 64$ 비트의 길이이면 중간 충돌(Meet-in-the-Middle)공격에 노출되어 일 방향 해쉬 함수로 사용할 수 없다.

보다 중간 충돌 공격으로부터 안전한 해쉬 함수는 Davies-Meyer방식 이다. 기본 알고리즘은

$$h_0 = IV$$
$$h_i = E_{mi}(h_{i-1}) \oplus h_{i-1}, \quad i = 1,2,3,\ldots,t$$
$$h(m) = h_t$$



[그림 16] Daives-Meyer 방식

중간 충돌 공격을 적용할 수 없지만, 해쉬 값이 길이가 128비트 이상이 아닐 경우 여전히 중간 충돌 공격에 취약하고 결과적으로 충돌 회피 해쉬 함수의 조건을 충족하지 못한다. 64비트 블록 암호인 DES를 이용할 경우에는 디지털 서명과 함께 이용할 수 없다.

이 밖에도 N-HASH, MDC2, MDC4등의 블록 암호에 기반을 둔 해쉬 함수가 제안되고 있다. 해쉬 값의 길이가 128비트 이상이 되어야 한다는 것이 안전한 충돌 회피 암수가 되기 위한 필요조건이 된다.

결국 블록 암호를 이용한 해쉬 함수는 DES와 같이 64비트 블록의 적용을

지양하고 대신 128비트 블록 암호를 이용해야 한다.

Juneman에 의해서 제안된 법(Modulus) 연산을 이용한 방식의 구축은 법 지수 연산을 이용했다. 해쉬 함수가 디지털 서명과 같이 사용될 경우에 디지털 서명에는 RSA의 경우에서처럼 법 지수 연산이 요구되기 때문에 기존에 이 작업을 위해서 작성된 여러 하드웨어나 소프트웨어들을 그대로 이용할 수 있는 장점이 있다.

- 메시지 m이 t개의 블록 : m1, m2, ..., mt
- 초기화 벡터 : IV
- 소수(Prime) : C
- mi는 0과 2l-1사이의 값을 선정함

$$h_0 = IV$$
$$h_i = (m_i + h_{i-1})^2 \bmod C, \quad i = 1,2,\ldots,t$$
$$h(m) = h_t$$

이 방식은 중간 충돌 공격 및 생일 공격에 취약성을 지니고 있다.

e. 전용 소프트웨어 해쉬 함수

MD4, MD5, SHA등은 해쉬 함수 전용으로 제안된 알고리즘이다.

MD4은 Rivest에 의해 제안되었으며, 해쉬 값의 길이가 128비트이고 소프트웨어 지향적인 방법으로 32비트 컴퓨터에서 효율적으로 구현되기에 적합했다 MD5는 MD4에 대한 취약성 분석이 시도된 후에 Rivest는 역시 128비트의 해쉬 값을 가지는 보다 개선된 버전을 제안했었다. 마지막으로 SHA는 미국표준연구소(NIST)에서는 MD4에 기반을 둔 SHA(Secure Hash Algorithm)을 제안했다. 160비트의 해쉬 값을 가지고 MDC해쉬 함수이다.

이 밖에도 RIPEM-128(해쉬 값:128)과 RIPEM-160(해쉬 값:160)등이 있다.

G. 디지털 서명(Digital Signature)

1. 생성 및 확인작업

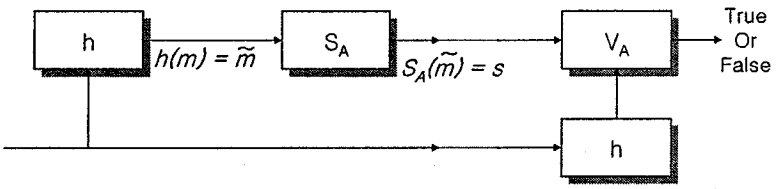
메시지를 서명이 된 메시지의 형태로 전환시키는 작업인 디지털 서명은 방식에 따라 2가지로 분류한다. 하나는 메시지 부가형 디지털 서명(Digital Signature Scheme with Appendix)이다. 나머지 하나는 메시지 복구형 디지털 서명(Digital Signature Scheme with Message Recovery)이다. 디지털 서명은 오직 서명자 자신에 의해서만 생성될 수 있어야 하기 때문에 서명자만이 알고 있는 비밀 정보가 서명 작업에 적용되어야 한다.

- 서명자가 사용할 수 있는 메시지 집합 : M
- 모든 사용 가능한 메시지 집합 : S
- 메시지에 대한 서명 생성(Signature Generation) 작업 : M에서 S로의 사상(Mapping)
- 메시지 m∈M 의 길이는 가변이므로 메시지 자체에 서명 작업을 하지 않고, 공개된 함수를 적용하여 서명에 적합한 형태의 메시지 $\tilde{m} \in \tilde{M}$ 으로 메시지를 재구성한다.
- 재구성된 메시지들의 집합 : $\tilde{M}$
- 서명 생성 작업 : $M \rightarrow \tilde{M}$, $\tilde{M} \rightarrow S$ 로의 연속적인 사상의 곱(Product)이다.

메시지 부가형 디지털 서명 기법은

- S_A : 서명자 Alice의 비밀 서명 함수
  - 메시지 m∈M에 대한 서명자 Alice의 디지털 서명 : $s = S_A(\tilde{M})$
- 메시지가 부가된 디지털 서명 : {m, s}
- 길이가 긴 메시지의 디지털 서명에는 ElGamal 서명, Schnorr 서명등과 같이 메시지가 첨가된 디지털 서명 기법 사용

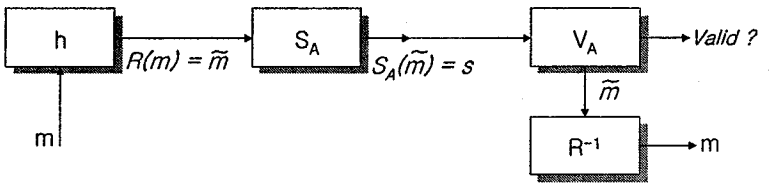
· MDC 해쉬 함수 : 재구성된 메시지 $\tilde{M}$ 를 도출하기 위해 사용한다. . $h(m) = \tilde{m} \in h(M) \subseteq \tilde{M}$, 따라서 $h(M) = \{h(m) : m \in M\}$



[그림 17] 메시지 부가형 디지털 서명의 생성과 확인

메시지 복구형 디지털 서명 기법은

- S_A : 서명자 Alice의 비밀 서명 함수
- 메시지 m∈M에 대한 서명자 Alice의 디지털 서명 : $s = S_A(\tilde{M})$ 으로부터 m이 직접 복구될 수 있기 때문에 s가 디지털 서명된 메시지가 된다.
- 길이가 짧은 메시지에는 RSA 서명이나 Rabin 서명과 같이 메시지 복구형 디지털 서명 기법 사용
- 재구성된 메시지 $\tilde{M}$ 를 도출하기 위해서는 역함수가 존재하는 중복함수(Redundancy Function), $R(m) = \tilde{m} \in R(M) \subset \tilde{M}$ 가 적용된다. $R(m) = \{h(m) \in M\}$



[그림 18] 메시지 복구형 디지털 서명의 생성과 확인

서명에 대한 확인 작업(Signature Verification)작업은 누구에 의해서도 행해질 수 있어야 하기 때문에 공개된 서명 확인 함수 V_A가 요구된다. 메시지 부가형 디지털 서명 기법에서 서명 확인 함수 V_A 는 $\tilde{M} \times S$ 로부터 {True, False}로의 사상(Mapping)으로 다음 2가지 조건을 만족해야 한다. 첫째, $s = S_A(h(m)) = S_A(\tilde{m})$ 이기만 하면 $V_A(\tilde{m}, s) = true$ 이어야 한다. 둘째, 서명자 Alice이외의 어느 누구도 $V_A(\tilde{m}, s) = true$ 를 만족하는 임의의 m∈M과 s∈S를 생성해 내는 것이 계산적으로 불가능해야 한다.

메시지 복구형 디지털 서명기법에서 서명 확인 함수 V_A 는 $S \rightarrow \tilde{M}$ 로의 사상이다. $s = S_A(R(m))$ 이기만 하면 $V_A(s) = \tilde{m} \in R(M) \subseteq \tilde{M}$ 을 만족한다. R의 역함수를 취함으로써 메시지 $m = R^{-1}(\tilde{m})$ 을 복구할 수 있다. 만약, $V_A \notin R(m)$ 면 유효하지 않는 서명으로 간주한다. 메시지 복구형 디지털 서명의 제약 조건은 구성 원소들이 특수한 유형의 구조를 취해야 하는 것을 요구하는데 결국 중복함수가 이러한 구조를 보장해 주는 역할을 담당한다. 이때 메시지들의 집합으로부터 재구성된 메시지들의 집합으로의 일대일 사상, 즉, $R: M \rightarrow \tilde{M}$ 인 중복함수의 적정한 선정은 메시지 복구형 디지털 서명의 안전성에 매우 중요한 영향을 미치게 된다.

즉 공개키 암호에 기반을 둔 디지털 서명의 가장 대표적인 것은 RSA디지털 서명과 ElGamal 디지털 서명이다.

a. 충돌 회피 해쉬 함수와 디지털 서명

메시지의 길이가 긴 경우에 메시지의 길이를 디지털 서명에 적합한 형태의 블록으로 나누어 각 블록 단위로 디지털 서명을 할 경우 즉, 메시지 m을 일정한 길이의 블록 m_1, m_2, ... m_t 로 구성되면, 메시지 m에 대한 Alice의 서명은 $S_A(m_1)$, $S_A(m_2)$, ... $S_A(m_t)$ 가 된다. 이 방식의 문제점은 각각의

메시지 블록에 대해서 독립적으로 디지털 서명이 생성되어 전달되므로 블록간의 어떤 연관도 존재하지 않는 것이다. 따라서 메시지 블록의 삭제, 반복의 검출이 불가능하다. 그리고, 일반적으로 디지털 서명을 생성할 때 비교적 많은 계산 량이 소요되기 때문에 각각의 블록 별로 디지털 서명을 생성하는 것은 비효율적이다. 이런 이유로 메시지 m에 MDC 해쉬 함수 h()를 적용한 결과 h(m)에 대하여 디지털 서명 $s = S_A(h(m))$ 을 생성한다. 이때 s만 가지고는 원래의 메시지를 복원할 수가 없으므로 원래의 메시지 m이 항상 디지털 서명 s와 함께 제시되어야 한다.

디지털 서명에서 요구되는 MDC 해쉬 함수의 다음의 조건을 만족한다면 '충돌 회피'성질을 가지고 있다고 할 수 있다.

- 함수 h()는 공개된 함수이고, 어떠한 비밀도 함수 계산이 사용되지 않았다.
- 메시지 m 의 길이에는 제한이 없고, h(m) 의 길이는 l비트로 고정된다. ($l \geq 128$)
- h()와 m이 주어졌을 때 h(m)의 계산은 용이해야 한다.
- m 과 h(m) 이 주어졌을 때 $h(m) = h(m')$ 을 만족하는 $m'(\neq m)$ 을 찾는 것은 계산적으로 불가능해야 한다.
- 동일한 해쉬 값을 가지는 임의의 두 개의 서로 다른 메시지를 찾는 것은 계산적으로 불가능하다

만약 충돌 회피 성질을 가지지 않는 해쉬 함수가 사용된다면 위조자(Forgery)에 의한 다음과 같은 공격이 가능해 진다.

- 위조자는 $h(m) = h(m')$ 을 만족하는 두 개의 메시지 m 과 m' 를 생성한다. 메시지 m 은 서명자에 의해 받아들여질 수 있는 내용이지만, 메시지 m'는 위조자에게 불법적인 이득을 초래하는 위조된 메시지이다.

• 서명자는 위조자에 의해서 제시된 메시지 m에 대하여 서명한다. • 서명자는 메시지 m' 에 대해서는 서명을 하지 않으려 하지만 위조자는 결국 서명자 Alice의 서명 $S_A(h(m'))=S_A(h(m))$ 을 얻게 된다.

2. 다양한 디지털 서명 기법

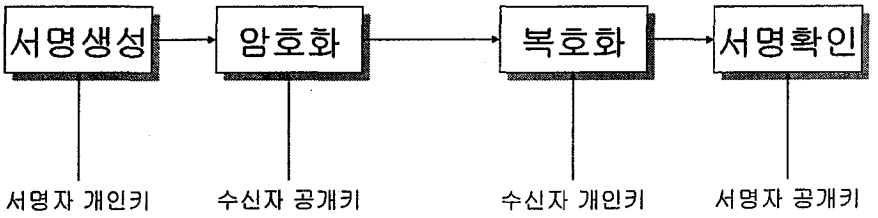
a. RSA 디지털 서명

가장 대표적인 디지털 서명으로 메시지 복구 기능이 있다. RSA 공개키 암호의 암호화 및 복호화를 서명의 생성과 확인에 이용한다. RSA 디지털 서명의 안정성은 소인수 분해의 어려움에 기반을 두고 있다. 방법은 평문 m에 공개키 $k_e=e$를 적용하여 암호화시키면 암호문 $c=E_{Ke}(m)=m^e \bmod n$이 생성되고, 개인키 $k_d=d$ 를 암호문 c에 적용하면 $m=D_{Kd}(c)=c^d \bmod n$ 이 복호화 된다. 여기서 법 $n=p\cdot q$는 매우 큰 두 소수 p와 q의 곱이다.[13]

디지털 서명 과정은 메시지 $m \notin M$ 에 중복 함수를 적용하여 얻은 $R(m)=\tilde{m} \notin \{0,1,2,\ldots,n-1\}$ 에 대해서 서명자의 RSA 개인키 $k_d=d$를 기반으로 하여 디지털 서명 $S_A(\tilde{m})=\tilde{m}^d \bmod n=s$ 를 생성한다. s가 메시지 m에 대한 디지털 서명이다.

디지털 확인 과정은 RSA 공개키 $k_e=e$를 이용하여 $\tilde{m}=s^e \bmod n$을 구한 후, $\tilde{m} \in R(M)$ 을 점검하게 되고, 만약 이를 만족한다면 정당한 서명으로 간주하여 $(V_A(s)=s^e \bmod n=\tilde{m} \in R(M), R^{-1}(\tilde{m})=m)$ 메시지 $m=R^{-1}(\tilde{m})$을 복구한다.

서명된 메시지에 대한 기밀성을 보장하기 위해서는 수신자의 RSA 공개키로 암호화를 추가로 수행해야 한다[그림 19]. 수신자의 암호화된 메시지를 먼저 자신의 RSA 개인키로 복화화를 한 후에, 서명자의 RSA 공개키로 서명에 대한 확인 작업을 수행하게 된다.



[그림 19] 메시지의 기밀성이 보장되는 RSA 디지털 서명

b. Rabin 디지털 서명

역시 메시지 복구 기능이 있는 디지털 서명으로 서명의 안전성은 Rabin 공개키 암호와 마찬가지로 법이 두 소수 p와 q의 합성수 $n = p \cdot q$일 경우 제곱 계산의 어려움에 기반을 두고 있다. 서명자의 공개키는 n, 서명자의 개인키를 p와 q로 한다.

- 메시지 $m \in M$에 중복 함수를 적용하여 얻은 $R(m) = \tilde{m} \in R(M)$의 법 n에 대한 제곱근 s를 구하면, 이것이 메시지 m에 대한 서명자의 디지털 서명이 된다. 여기서의 R(M)은 법 n에 대한 2차 잉여류의 집합이다.
- 서명에 대한 확인은 서명자의 공개키 n을 이용하여 $\tilde{m} = s^2 \bmod n$을 구한 후에 $\tilde{m} \in R(M)$ 임을 점검하게 되고 만약 이를 만족한다면 서명으로 간주하여 $m = R^{-1}(\tilde{m})$을 복구하게 된다.

c. ElGamal 디지털 서명

메시지 부가형 디지털 서명 기법에서 사용되고, ElGamal 디지털 서명의 안전성은 매우 큰 소수 p에 대해서 Z_p^* 상에서의 이산대수 문제의 어려움에 기반을 두고 있다. 현재 소수 p는 Pohlig-Hellman 알고리즘으로부터 안전하기 위해 768비트 이상이 권고된다. 서명자의 공개키는 $\{p, g, y = g^x \bmod p\}$이고 여기서 x는 난수, g는 곱셈상의 군 Z_p^*의 생성자 이다. 서명자의 개인키는 {x}이고,

서명자는 임의의 길이의 2진 메시지 m을 자신의 개인키로 이용하여 서명하고, 공개키를 통해서 그 서명에 해당하는 확인 작업을 할 수 있다.

메시지 m에 대한 서명 작업은

- $\gcd(\hat{x}, p-1)=1$의 관계에 있는 난수 $\hat{x} \in [1, p-2]$를 선정한다.
- 메시지 m에 대한 해쉬 함수 값 $c = h(m)$을 구한다.
- 메시지 m에 대한 디지털 서명 $\{r, \hat{y}\}$를 생성한다.

$$\hat{y} = g^{\hat{x}} \bmod p$$
$$r = \hat{x}^{-1}\{c - x \cdot y\} \pmod{p-1}$$

서명에 대한 확인 작업은

- 메시지 m에 대한 해쉬 함수 값 $c = h(m)$을 계산한다.
- $g^c = \hat{y}^r \cdot y^{\hat{y}}$이 성립하면 서명을 받아들인다.

d. DSA(Digital Signature Algorithm)

DSA는 ElGamal의 변형 형태로 SHA(Secure Hash Algorithm)가 사용되었다. ElGamal서명과 달리 위수가 q인 부분 군을 이용하기 때문에 비록 계산적인 속도나 보안수준이 더 향상되지는 않지만 서명의 크기가 작아지기 때문에 메모리 공간 및 통신 비용 측면에서 효율적이다.

디지털 서명과정은

- 먼저 난수 $\hat{x} \in [1, q-1]$을 선정하고 메시지 m에 대한 해쉬 값 $c = h(m)$을 구한다.
- 아래의 계산을 통해 메시지 m에 대한 서명 $\{r, \hat{y}\}$을 생성한다.

$$\hat{y} = (g^{\hat{x}} \bmod p) \bmod q$$
$$r = \hat{x}^{-1}\{c + x \cdot y\} \bmod q$$

여기서 p, q 는 두 개의 큰 소수이다.

서명에 대한 확인 작업은

- 먼저 메시지 m에 대한 해쉬 함수 값 $c=h(m)$을 계산한다.
- 아래 계산 과정을 통해서 얻은 결과를 기반으로 $u=\hat{y}$ 이면 서명을 받아들인다.

$$u_1 = c \cdot r^{-1} \bmod q, \quad u_2 = \hat{y} \cdot r^{-1} \bmod q$$
$$u = (g^{u_1} y^{u_2} \bmod p) \bmod q$$

e. Schnorr 디지털 서명

ElGamal 디지털 서명의 또 다른 변형 형태로서 서명자의 공개키와 개인키를 생성하는 방법은 DSA와 동일하다.

디지털 서명과정은

- 먼저 난수 $\hat{x} \in [1, q-1]$을 선정하고 메시지 m에 대한 해쉬 값 $c=h(m)$을 구한다.
- 아래의 계산을 통해 메시지 m에 대한 서명 $\{r, \hat{y}\}$을 생성한다.

$$\hat{y} = g^{\hat{x}} \bmod p$$
$$c = h(m \| \hat{y})$$
$$r = c \cdot x + \hat{x} \bmod q$$

서명에 대한 확인 작업은

- 먼저 메시지 m에 대한 해쉬 함수 값 $c=h(m \| \hat{y})$을 계산한다.
- $g^r \equiv y^c \hat{y} \bmod p$ 이면, 서명을 받아들인다.

f. 일회용 디지털 서명

오직 한 개의 메시지를 서명하기 위해서 사용될 수 있는 디지털 서명 기법이다. 매 메시지 마다 새로운 공개키와 개인키가 적용되어야 한다. Merkle에

의해서 제안된 일회용(one-time) 디지털 서명은 DES와 같은 대칭형 암호를 이용한 일방향 해쉬 함수에 기반을 두고 있기 때문에 서명의 생성과 확인 작업에 적은 계산 량이 소요된다. 낮은 계산 복잡도가 요구되는 스마트카드를 기반으로 하는 응용분야에 적합한 방식이다.

III. 개인 식별이 가능한 디지털 서명

A. 방법

그룹 디지털 서명에서 제공하는 익명성은 영지식 증명을 통해서 구현된다. 영지식(Zero-Knowledge)증명이란 사용자가 서비스 제공자-기관-를 신뢰할 수 없을 때 단지 자신의 신분만을 서비스 제공자에게 밝히기 원하는 경우에 사용하는 방법이다. 영지식 개인 식별 프로토콜은 서비스 제공자와 사용자와의 대화형 프로토콜로서 사용자의 비밀 정보를 서비스 제공자에게 직접적으로 제공함이 없이 사용자가 단지 그 비밀 정보를 알고 있다는 사실만을 서비스 제공자에게 확인시켜 주는 아주 정교한 개인 식별 프로토콜이다. 이때 자신의 신분을 밝히려는 사용자를 증명자(Prover), 사용자의 신분을 확인하려는 서비스 제공자를 검증자(Verifier)라 한다. 이는 암호학에서 매우 중요한 개념이므로 본고에서는 Discrete Logarithm에 대한 영지식 증명 방법에 대해 간략히 설명하겠다.

$7^2 \equiv 49 \pmod{71}$

$7^3 \equiv 59 \pmod{71}$

$7^4 \equiv 58 \pmod{71}$

$7^5 \equiv 51 \pmod{71}$

...

위의 식은 7의 멱 승에 71로 나눈 나머지 값을 나타낸 것이다. 위의 식을 기반으로 어떤 수에 대한 로그 값을 구할 수 있다. 예를 들어, $\log_7 51 \pmod{71}$ 은 5 라는 것을 알 수 있다. 이를 보통의 로그와 구별하기 위해 Discrete Logarithm이라고 한다.

일반적으로 $a^x = b \pmod{n}$ 에서 a, b, n 을 알고 있을 때, x 를 구하기 어렵다고 알려져 있다. 여기서 x 값을 알고 있다는 사실을 검증자에게 증명해야

한다고 생각해보자. 물론 x값을 알려주면 되지만 x값은 비밀로 간직해야 한다. 이 경우 영지식 증명 방법을 사용하게 되는 것이다. 그럼 이제 증명 방법에 대해 알아보자.

a.	Alice는 r값을 랜덤하게 선택해서 $B = a^r$를 Bob에게 보낸다.
b.	Bob는 c값을 랜덤하게 선택해서 Alice에게 보낸다.
c.	Alice는 $s = r + cx$를 계산해서 s값을 Bob에게 보낸다.
d.	$a^s = Bb^c$가 성립하면 Bob는 Alice가 x값을 알고 있다고 판단한다.

위의 영지식 증명을 제안하는 그룹 디지털 서명에서는 다수의 복잡한 식을 만족하는 Discrete Logarithm값에 대한 영지식 증명방법을 이용하여 증명한다. 이제 그룹 디지털 서명에 대해 알아보겠다. 본 고의 디지털 서명은 Bilinear Group에 기본을 두고 있다. Bilinear Group에 대해 알아보기 전에, Bilinear map에 대한 몇 가지 개념에 대해 살펴본다. 표기법은 [20]에 따랐다.

1.	G_1, G_2는 소수 p의 두 개(Multiplicative)의 환 그룹이다.
2.	g_1, g_2는 각각 G_1, G_2의 생성원이다.
3.	φ는 $\varphi(g_2) = g_1$을 만족하는 G_2, G_1에서의 계산 가능한 유질동상이다.
4.	e는 $e: G_1 \times G_2 \rightarrow G_T$에서 다음과 같은 성질을 갖는 계산 가능한 맵이다.
5.	Bilinearity : 모든 $u \in G_1$, $v \in G_2$, $a, b \in Z$에 대해서, $e(u^a, v^b) = e(u,v)^{ab}$이 성립한다.
6.	Non-degeneracy : $e(g1, g2) \neq 1$

위의 성질을 만족하는 맵 e는 타원곡선의 Pairing을 이용할 수 있다. 본 고에서는 G_1의 크기가 약 2^{170}으로 G_1의 원소를 171비트로 나타낼 수 있다는 사실만 이용한다. 그리고 [20]에 사용하고 있는 Q-Strong Diffie-Hellman Problem

(q-SDH)과, Dicision Linear Problem in G_1이 어렵다는 가정을 그대로 사용한다.

구성 요소는

- 멤버(Member): 그룹에 가입하여 발급자를 통해 서명키를 얻는다. 서명키를 얻으면 서명키를 이용하여 그룹 서명을 할 수 있다.
- 발급자(Issuer): 사용자가 그룹에 가입할 때 서명키를 획득할 수 있도록 해준다. 발급자는 서명키의 일부 정보를 알 수 있다. 이때 얻어낸 일부 정보는 등록 테이블에 저장된다.
- 개봉자(Opener): 분쟁이 발생했을 경우, 해당 서명을 열어봐서 서명자를 찾아낼 수 있다. 발급자와 개봉자의 개념을 합쳐서 그룹 관리자라고 통칭할 수 있다.
- 등록 테이블(Reg): 사용자가 가입할 때, 발급자는 등록 테이블에 그 서명자의 서명키의 일부 정보를 이 등록 테이블에 저장한다. 개봉자가 추후에 서명을 개봉하여 서명자를 찾아낼 때 이 테이블을 참고한다.

구성 키는

- 그룹 공개키(GPK): 이 키는 그룹 서명의 구성원들과 구성원 이외에 서명을 받은 사람 모두에게 공개된 키이다. 특히, 서명을 받은 멤버가 이 공개키를 이용해서 서명의 적법성을 판단하게 된다.

  $g_1,\ u_1,\ v_1,\ h_1,\ u_2,\ v_2,\ h_2 \in G_1\ \&\ u_1^{\varepsilon_{11}} = v_1^{\varepsilon_{12}} = h_1,\ u_2^{\varepsilon_{21}} = v_2^{\varepsilon_{22}} = h_2$

  $g_2,\ w_1,\ w_2\ \in G_2$

  $\varphi(g_2) = g_1$
- 발급자 비밀키(IK): 발급자는 발급자 비밀키를 이용해서 사용자가 가입했을 경우, 그 사용자가 서명키를 획득할 수 있게 도와준다.

  $w_1 = g_2^{r_1},\ w_2 = g_2^{r_2}$를 만족하는 $r_1,\ r_2$

- 개봉자 비밀키(OK): 개봉자는 개봉자 비밀키를 이용해서 문제가 발생한 그룹서명에 대해 그 서명의 주인이 누구인지 밝혀낼 수 있다.

  $OK = (\varepsilon_{11}, \varepsilon_{12})$

- 서명키($gsk[i]$): 멤버가 발급자를 통해 얻는 키이다. 그룹 멤버는 이 키를 이용하여 그룹 서명을 할 수 있다.

  $gsk[i] = (A, B, x, x')$

- 멤버 공개키, 비밀키($mpk[i]$, $msk[i]$): 각 멤버는 서명키로 공개키와 비밀키 쌍을 만든 다음 자신의 공개키를 공개한다.

  1. 조건 $A, B \in G_1$, $x, x' \in Z_p$, $A^{x+r_1} = g_1$, $B^{x+x'+r_2} = g_2$ 을 만족하면,
  2. $e(A, w_1 g_2^{x}) = e(g_1, g_2)$, $e(B, w_2 g^{x+x'})_2 = e(g_1, g_2)$ 으로 받은 비밀키의 적법성을 판단

연산 방법은

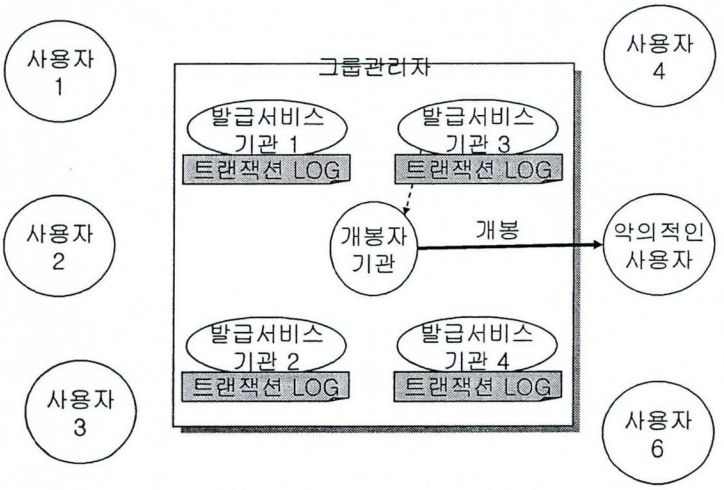
1. 키 생성($KeyGen(n)$): 신뢰할 수 있는 기관이 이 연산을 수행한다고 가정한다. 이 연산을 통해 그룹 공개키(GPK), 발급자 비밀키(IK), 개봉자 비밀키(OK), 추적자 비밀키(TSK)가 만들어 진다.
2. 멤버키 생성($MKeyGen$): 각 멤버는 이 연산을 수행하여 공개키와 비밀키 쌍($mpk[i]$, $msk[i]$)을 만들어 낼 수 있다.
3. 그룹 서명, 검증(GSign , GSVerify): 그룹 멤버(i)는 자신의 서명키($gsk[i]$)를 이용하여 이 연산을 수행함으로써 그룹 서명(GSign)을 만들어 내고, 그룹 서명을 받은 사람은 그룹 공개키(GPK)를 가지고 이 연산을 수행하여 받은 서명이 적법한지를 판단하게 된다.
4. 개봉(Open): 개봉자만이 이 연산을 수행 할 수 있다. 개봉자는 비밀키(OK)와 등록 테이블(Reg)을 이용하여 이 연산을 수행하며,

주어진 서명의 서명자를 찾아 낼 수 있다

- $OK = (\varepsilon_{11}, \varepsilon_{12})$ 값에서 서명의 $A = T_3 / T_1^{\varepsilon_{11}} T_2^{\varepsilon_{12}}$ 값을 구한다.
- 멤버들의 비밀키 중에 A 와 일치하는 멤버를 찾아내서 서명자의 신원을 파악한다.

5. 소유권 주장($Claim$): 서명자는 자신이 만든 서명에 대해 소유권을 주장할 때 이 연산을 수행한다. 이 연산은 해당 서명의 소유권을 가지고 있음을 주장하는 근거를 리턴 한다.

자신의 서명일 때, $h^{\alpha+\beta} = T_3 / A$, $rx = s_x - cx$ 을 정확히 구할 수 있다.



[그림 20] 익명의 트랜잭션 시스템

위에서 나타낸 구성요소들을 그림으로 표현하면 [그림 20]과 같이 된다. [그림 20]은 발급 서비스를 제공하는 기관이 여러 곳으로 나누어져 있고, 각 트랜잭션 내용은 해당 서비스 기관의 등록 테이블에 로그형태로 남는다. 그림과 같은 시스템에서 법적으로 문제가 있는 트랜잭션이 발생하여 악의적인 사용자를

찾는다고 가정하자. 이 경우 개봉자 기관에서 트랜잭션 안에 있는 사용자의 서명에 대해 개봉 연산을 수행하여 악의적인 사용자를 찾아낼 수 있을 것이다.

위의 경우 '발급 서비스 기관 3'에서 온 요청이 불법적인 트랜잭션 요구라는 것만 개봉자가 알았을 뿐 악의적인 사용자가 누구인지는 아직 알 수 없기 때문에 개봉자는 '발급 서비스 기관3'의 등록 테이블의 모든 트랜잭션 로그 파일과 등록 테이블의 사용자 조합으로 개봉 연산을 수행하여 악의적인 사용자를 밝혀낸다. 이는 그룹 서명의 근본적인 취지인 사생활 보호에 어긋나서 존재 의의를 잃게 만든다. 또 설사 존재 의의를 상실함에도 개봉 연산을 수행하여 악의적인 사용자를 밝혀낸다 하더라도 많은 시간이 소요된다. 개봉 연산은 등록 테이블 로그 파일을 순서대로 뒤져서 저장된 서명자를 찾아야 하기 때문에 그룹 멤버의 수가 n 이라면 $O(n)$ 의 시간이 걸린다. 또 서명자를 찾아낸 후 트랜잭션 로그 파일을 읽어 서명자와의 확인을 하기 때문에 트랜잭션의 개수가 m 이라면 총 소요시간은 $O(m \times n) = O(mn)$ 이다. 이는 기존의 방법에서 규모의 차이를 제외하면 별 차이가 없다.

제안하는 디지털 서명의 구성에 대해 알아보자. 기존의 디지털 서명에 몇 가지를 추가하였다.

구성 요소는

- 멤버(Member)
- 발급자(Issuer)
- 개봉자(Opener)
- 추적자(*Tracer*): 다수 일지도 모르는 추적자는 트랜잭션에 담긴 서명에 대해, 그 서명이 특정 멤버의 서명인지 확인하는 작업을 한다
- 등록 테이블(Reg)

구성 키는

- 그룹 공개키(GPK)
- 발급자 비밀키(IK)
- 개봉자 비밀키(OK)
- 추적자 비밀키(TSK): 추적자는 그룹 관리자가 개봉한 연산을 멤버가 만든 서명인지를 확인할 수 있다.

  $TSK = (\varepsilon_{21}, \varepsilon_{22})$
- 서명키($gsk[i]$)
- 멤버 공개키, 비밀키($mpk[i]$, $msk[i]$)

연산 방법은

1. 키 생성($KeyGen(n)$)
2. 멤버키 생성($MKeyGen$)
3. 그룹 서명, 검증(GSign , GSVerify)
   - SDH에 대한 영지식(Zero-knowlodg) 프로토콜을 확장
     1. Alice(검증자)는 $\alpha, \beta \in Z_p$ 를 랜덤하게 선택하여, $T_1 = u_1^{\alpha}$, $T_2 = v_1^{\beta}$, $T_3 = h_1^{\alpha+\beta}$, $T_4 = u_2^{\alpha}$, $T_5 = v_2^{\beta}$, $T_6 = h_2^{\alpha+\beta}$ 을 계산한다.
     2. $(\alpha, \beta, x, x', \delta_1 = x\alpha, \delta_2 = x\beta, \delta'_1 = x'\alpha, \delta'_2 = x'\beta)$ 에 대한 Proof of Knowledge를 수행해서 $T_1, ..., T_6$ 에 대한 $R_1, ..., R_{10}$ 값을 구한다.
   - Fiat-Shamir 변환을 이용하여 이 프로토콜을 디지털 서명으로 변환
     1. 위의 프로토콜대로 $T_1, ..., T_6$, $R_1, ..., R_{10}$ 을 구한다.
     2. challenge값 $c = H(m, T_1, ..., T_6, R_1, ..., R_{10})$ 를 구한다
     3. 디지털 서명 σ 는 $(T_1, \Lambda, T_6, c, s_\alpha, s_\beta \Lambda, s_{\delta'_1}, s_{\delta'_2})$ 이 된다. 이로서 각 요소는 171바이트를 차지하기 때문에 이 서명의 크기는 약 342바이트가

된다.

4. 노출(Re *veal*): 개봉자가 분쟁이 생긴 서명을 개봉하여 악의적인 멤버를 찾아낸 후, 이 연산을 수행한다. 이 연산은 등록 테이블(Reg)을 참고하여 해당 멤버의 서명키 일부를 리턴 한다. 이 정보는 다수의 추적자들에게 제공된다.

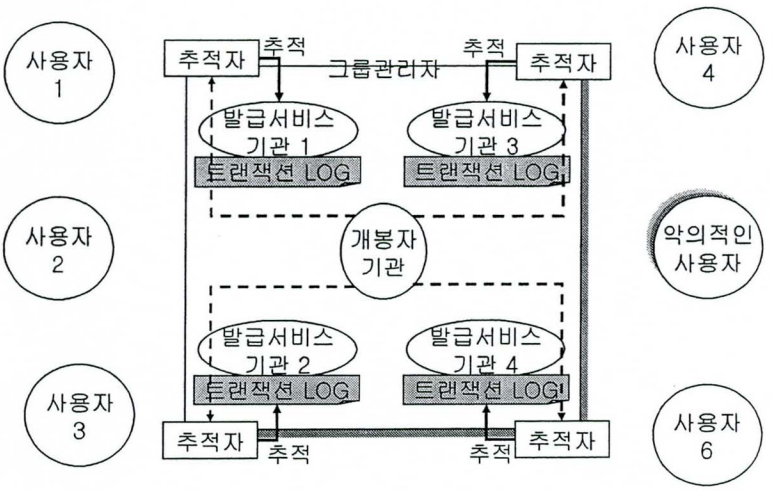
   회원 i 의 서명키 $gsk\ [i] = (A, B, x, x')$ 에서 B 값을 리턴한다.

5. 추적(*Trace*): 다수의 추적자들은 자신이 맡은 트랜잭션들에 대해 이 연산을 수행한다. 이 연산은 주어진 서명이 해당 멤버가 서명한 것인지 여부를 리턴 한다.

   · $TSK = (\varepsilon_{21}, \varepsilon_{22})$ 값에서 서명의 $B' = T_6 / T_1^{\varepsilon_{21}} T_2^{\varepsilon_{22}}$ 값을 구한다.

   · 인자로 주어진 B 와 구한 B' 값이 같으면 TRUE를 리턴하고 아니면 FALSE를 리턴한다. Open 연산에 비해 효율적이다.

6. 소유권 주장(*Claim*)



[그림 21] 제안한 익명의 트랜잭션 시스템

[그림 21]은 제안한 서명에 따른 연산을 그림으로 표현한 것이다. 여기에서 보면 [그림 20]과의 다른 점은 추적자의 유무이다. 앞에서 말한 익명성과 시간의 문제를 해결하기 위해 개봉자 기관은 악의적인 멤버의 신원을 [그림 20]에서와 동일한 방법으로 알아낸다. 그런 다음 개봉자 기관이 노출 연산을 이용하여 등록 테이블에서 서명키 일부를 알아낸다. 이 정보는 다수의 추적자들에게 전달되고 추적자들은 자신이 맡은 트랜잭션 로그에 대한 추적 연산을 수행하여 악의적인 멤버가 수행한 트랜잭션을 찾아낸다. 이 방법으로 익명성 문제가 해결 가능하다. 개봉 연산이 어떤 사용자에게 요구됐는지를 알려주는데 반해 추적 연산은 이 트랜잭션이 악의적인 사용자가 요구했는지에 대한 판단의 결과를 보여준다. 그 결과 개봉자는 모든 트랜잭션에 대한 개봉 연산이 아니라 추적자들이 추적 연산을 수행 후 악의적인 멤버의 트랜잭션이라고 확인된 로그파일에 대한 개봉 연산을 수행하기 때문에 익명성이 보장될 수 있다. 또 추적 연산은 등록 테이블을 참조하지 않고 단순히 로그 파일에 대한 계산만을 수행한다. 따라서 그룹 멤버의 수가 n 이라면 개봉 연산의 수행 시간은 $O(n)$ 인데 반해 추적 연산의 수행 시간은 $O(1)$ 이다. 그리고 다수의 추적자가 동시에 분담하여 일을 수행하기 때문에 트랜잭션의 개수가 m 이라면 제안한 그룹 디지털 서명의 수행 시간은 $O(1 \times \frac{m}{t}) = O(m/t)$ 가 되므로, 기존의 $O(mn)$ 에 비해 훨씬 효율적이다.

B. 예상 응용 분야

그룹 디지털 서명의 익명성은 여러 곳에서 응용이 가능할 것이다. 그 응용방법에 대해 보겠다.

그 예 중 첫 번째는 입찰이다. 입찰은 입찰에 참여하는 회사들이 하나의 그룹을 이룬다.[21][22] 각 회사는 입찰 가격을 그룹 서명으로 서명하여 제출한다. 낙찰을 결정하는 기관에서는 그룹 서명의 익명성 때문에 제시된 가격의 입찰 가격만을 알 수 있을 뿐 어떤 회사가 어떤 가격을 제출 했는지 알 수 없다. 따라서 입찰 가격을 가지고 공정하게 낙찰 여부를 판단 하게 된다. 또한 낙찰이 결정된 후에도 서명을 개봉하여 낙찰된 회사만을 알 수 있기 때문에 다른 회사가 어떤 가격을 제시했는지 알 수 없다.

두 번째는 자동차 안전 통신 시스템으로 자동차에 단거리 발신기를 장착하여 근거리에 있는 자동차끼리 서로 통신 할 수 있도록 하는 시스템을 생각해 볼 수 있다.[19] 자동차에 장착된 발신기는 자동차의 상태정보를 주위의 차들에게 발신한다. 예를 들어, 자동차가 급정거를 하는 경우, 이 사실이 발신기를 통해 주위의 모든 차에게 알려진다. 이런 시스템이 도입되면 자동차 사고를 줄이는데 많은 도움이 될 것이다.

문제는 자동차가 아닌 다른 장소에서 거짓 상태 정보를 발신함으로써 통신 시스템의 혼란을 야기하는 악의적인 사람이다. 이를 막기 위해서는 자동차 안에 복제할 수 없는 칩을 넣어서, 발신기의 상태 정보 메시지가 이 칩을 통해 디지털 서명이 되도록 하면 될 것이다. 하지만 단순히 디지털 서명을 하는 경우에는 자동차 번호가 노출될 수 있다. 발신기가 내보내는 메시지는 자동차의 상태나 위치 등의 내용이 담겨있기 때문에 결국 누가 어디에 있는지에 대한 정보가 노출되는 결과를 낳는다. 더군다나 모든 자동차에 대한 공개키를 저장할 공간도 없다. 그룹 디지털 서명은 이런 문제를 해결할 것이다.

그 외의 그룹 디지털 서명의 익명성의 매우 매력적인 특징은 여러 곳에서 이용될 수 있다는 점이다. 따라서 위의 예들 이외에도 많은 응용을 생각해 볼 수 있다.

투표 시스템에서 그룹 서명의 익명성을 이용하여 투표자의 신원을 숨길 수 있을 것이다. 전자 화폐 시스템에서 그룹 서명을 이용하여 화폐를 발급한 은행이나 전자 화폐를 소비하는 소비자의 신원을 숨길 수 있을 것이다.[23] 또한 신뢰 가능한 컴퓨팅에서도 그룹 서명을 이용할 수 있을 것이다. 신뢰 가을한 컴퓨팅은 Remote Attestation이라는 기능을 가지고 있는데, 이 기능은 데스크톱 PC가 원격의 상대방에게 어떤 소프트웨어를 수행하고 있는지 증명하는 기능이다. 이때 익명성을 보장하기 위해 그룹 서명을 사용할 수 있다.[24]

IV. 결론 및 향후 과제

본 고의 I장에서 정보보안의 중요성에 대해, II장에서는 구체적인 정보보안에 대해, 그리고 III장에서는 구체적인 정보보안 방법 중 제안하는 보안방법을 기술 하였다. 제안하는 보안방법은 디지털 서명의 확장 개념인 그룹 디지털 서명에 대해 중점을 두어 기술 하였다. 제안한 개인식별에 유리한 그룹 디지털 서명 방법은 기존의 그룹 디지털 서명 과정의 서명 연산에 SDH에 대한 영지식 프로토콜을 확장하여 Fiat-Shamir변환을 이용한다. 이렇게 할 경우 기존의 방법이 약 1238바이트를 차지한다고 할 때 약 400바이트 이하를 차지하기 때문에 서명의 크기를 대폭 줄였다.

아직 그룹 디지털 서명이라는 개념 자체가 익숙하지 않아 실질적으로 사용되는 예는 극히 드물다. 때문에 구체적인 언급이 미약할 수 밖에 없다. 또 제안한 연산방법 중 신뢰 가능하다고 예상되어지는 키 발급자의 가정이 확실치 않기 때문에 완전한 서명 방법이라고 할 수도 없다. 그러므로 우선 이 가정을 확실히 할 수 있는 안전한 Join 연산을 고안하는 것이 큰 과제라 할 수 있다.

이런 문제점들을 해결했을 때, 제안한 그룹 디지털 서명은 그룹 디지털 서명의 서명자가 그룹에 속한 사실만을 알려줄 뿐 그 서명자의 신원에 대한 익명성을 보장한다는 장점과 짧아진 서명으로 빠른 실행 속도, 추적연산의 도입으로 보다 익명성을 보장할 수 있기 때문에 그 응용분야가 많을 것으로 예상된다.

참고문헌

[1] 김창호, 분산컴퓨팅 네트워크 암호화 사용자 보안 시스템, 조선대학교 대학원 석사논문, 2004.2.

[2] 최승걸, 박근수, 익명성 메커니즘: 그룹 서명과 추적 가능한 서명, 정보과학회지 Vol.23, No.1, 2005.1.

[3] Schneier, Bruce, Applied Cryptography: Protocols, Algorithms and Source Code in C, 1995.

[4] Stinson and Douglas R. Cryptography: Theory and Practice, 1995.

[5] Hakin, Martha E. and Robert B. J. Warner, Smart Card Technology: New Method for Computer Access Control, Special Publication, Gaithersburg, 1988.9.

[6] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva and M. Yung, Systematic design of two-party authentication protocols, CRYPTO 1991, Springer.

[7] Teresa F. Lunt, A survey of intrusion detection techniques, Computers & Security, vol.4 no.12: 405-418, 1993.12.

[8] Sandeep Kumar, Eugene H. Spafford, A Software Architecture to support Misuse Intrusion Detection, Proceedings of the 18th National Information Security Conference, 1995.

[9] William Srallings, Cryptography Throry and Network Security: Principles and Practice, Pentice Hal, 1998.

[10] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, CRC, 1996.

[11] R. M. Davies, Data Encryption Standard in Perspective, Computer Security and Data Encryption Standard, National Bureau of Standard Special Publication, 1978.

[12] Eli Biham, Adi Shamir, Differential Cryptanalysis of DES-like Cryptosystems,

CRYPTO 1990 Vol.537: 2-21, 1991.

[13] RSA Security, Securing Data At Rest - Developing a Database Encryption.

[14] G. Davida, Chosen signature cryptanalysis of the RSA (MIT) public-key cryptosystem, Technical report TR-CS-82-2, Department of EECS, University of Wisconsin, 1982.

[15] El Gamal, T., A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory, Vol. 31: 469-472, IEEE, 1985.

[16] R.L. Rivest, A. Shamir, and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communication of the ACM. 21:12-126.

[17] I. Damgard, Collision Free Hash Functions and Public Key Signature Schemes, EUROCRYPT: 203-216, 1987.

[18] FIPS 180-1, Secure Hash Function, Federal Information Processing Standards Publiccation, U.S.Department of commerce/NIST, 1995

[19] D. Boneh, X. Boyen, and H. Shacham, Short Group Signature, CRYPTO 2004, Springer-Verlag.

[20] D. Boneh, B. Lynn and H. Shacham, Short Signatures from Weil pairing, ASIACRYPT 2001. vol.2248 of LNCS, Springer. (Full Paper: http://crypto.standard.edu/~dado/pups.html)

[21] A. Kiayias, Y. Tsiounis, and M. Yung, Traceable Signature, EUROCRYPT 2004 of LNCS, Springer, 2004.1.12.

[22] A. Kiayias and M. Yung, Group Signature: Efficient constructions and anonymity from trapdoor-holders, CRYPTOLOGY ePrint Archive, Report vol.076, 2004.

(http://eprint.iacr.org)

[23] Trusted Computer Group, Trusted Computing Platform Alliance(TCPA) Main Specification, 2003. (http://www.trustedcomputergroup.org)

[24] T. Garfinkel, B. Praff, J. Crow, M. Rosenblum and D. Boneh, Terra: A virtual machine-based platform for trusted computing, SOSP 2003.

[25] M. Abdalla, J. An, M. Bellare ang C. Namprepre, From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forword-security, EUROCRYPT 2002. vol.2332 of LNCS, Springer.

[26] M. Bellare, H. Shi, and C. Zhang, Foundation of group signatures: The case of dynamic groups, ePrint Archive, Report vol.077, 2004. (http://eprint.iacr.org)

[27] M. Bellare, D. Micciancio, and B. Wairnshci, Foundation of group signatures: Formial definitions, simplified requirements, and a construction based on general assumptions, EUROCRYPT 2003. vol.2656 of LNCS, Springer.

[28] D. Chaum, E. van Heyst, Group Signatures, EUROCRYPT 1991. vol.547 of LNCS, Springer.

[29] D. Chaum, Security without identification: Transactions systems to make big brother obsolete, Communications of ACM 28(10): 1030-1044, 1985.

[30] L. Chen, TP. Pedersen, New Group Signature schemes, EUROCRYPT 1994. of LNCS, Springer.

[31] A. Fiat, A. Shamir, How to prove yourself: Pracitical solutions to identification and signature problems, CRYPTO 1986. vol.263 of LNCS, Springer.

[32] A. Miyaji, M nakabkyashi and S. Takano, New explicit conditions of elliptic curve traces for FR-reduction. IEICE Trans, Fundamentals, E84-A(5): 1234-42, 2001.5.

[33] K. Rubin, A. Silverberg, Supersingular Abelian varirties in crytology, CRYPT

2002. of LNCS, Springer.

[34] G. Ateniese, B. de Medeirros, Efficient Group Signatures without Trapdoors, ASIACRYPTO 2003 of LNCS, Springer.

[35] G. Ateniese, J. Camenish, M. Joye and G. Tsudik, A practical and provably secure coalition-resistant group signature scheme, CRYPTO 2000 of LNCS, Springer.

[36] Mitsunari Shigeo, Sakai Ryuichi and Kasahara Masao, A New Traitor Tracing, IEICE TRANS., Vol.E82 No.1, 1999.1

[37] J. Camenisch and A. Lysyanskaya, Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation, EUROCRYPT 2001 of LNCS, Springer.

[38] J. Camenisch and M. Stadler, Efficient group signature schemes for large groups, CRYPT 1997 of LNCS, Springer.

[39] J. Camenisch, Efficient and Generalized group signatures schemes, EUROCRYPT 1997 of LNCS, Springer.

[40] A. Lysyanskaya and Z. Ramzan, Group blind digital signatures: A scalable solution to electronic cash, Financial Cryptography 1998.

[41] J. Kilian and E. Petrank, Identity escrow, CRYPTO 1998 of LNCS, Springer.

[42] L. Nguyen and R. Safavi-Naini, Efficient and Provably Secure Tradoor-free Group Signatures Schemes from Bilinear Pairing, ASIACRYPT 2004. (Available at: http://eprint.iacr.org/2004/104)

[43] D. Piontcheval and J. Stern, Security arguments for digital signatures and blind signatures, Journal of Cryptology 2000 Vol.3 No.3: 361-369.

[44] C. Shnorr, Efficient signature generation by smart cards, Journal of Cryptology 1991 Vol.4 No.3: 161-174.