通信情報保護學會誌
第6卷 第2號, 1996. 6

Public Key Escrow Systems의 소개 및 분석

(Introduction and analysis of public key escrow systems)

정 경 임*, 이 필 중*

1. 서 론

암호학이 발달하고 또 사람들에게 많이 알려짐에 따라 사람들은 자신의 비밀을 보장해 줄 많은 암호학적 도구들을 가지게 되었지만 한편으로는 더 많은 사람의 안녕을 위한 법집행기관의 합법적인 도청이 어려워지게 되었다. 그래서 법집행기관의 합법적인 도청을 가능하게 하면서도 개인의 사생활에 대한 비밀 보장을 하기 위해 미국에서는 1991년에 "Escrowed Encryption Initiative"를 제안하였고 call for comment 등의 절차를 거쳐 Escrowed Encryption Standard[EES]가 되었다. EES는 가장 강력하다고 알려진 SKIPJACK이라는 비밀 알고리즘을 사용하여 암호화와 복호화를 하며 SKIPJACK은 역공정(reverse engineering)이 어려운 하드웨어 디바이스(예를 들어 IC 칩)에서 구현되고 사용자의 비밀키(하드웨어 디바이스의 Device Unique Key(KU))가 신뢰받는기관에 escrow된다.

그러나 많은 사람들은 비밀 알고리즘을 사용하고 사용자의 비밀키를 두 기관에 나누어 escrow하고 하드웨어로 구현되는 방법에 반대를 했다. National Institute of Standards and Technology(NIST)는 SKIPJACK 알고리즘의 안전성에 대해 외부의 전문가들에게 조사를 의뢰했고 이들의 SKIPJACK 알고리즘에 대한 평가[BeGr]는 앞으로 30~40년사이에 SKIPJACK이 exhaustive search로 해독될 염려는 없고 short cut 방법에 의해 해독될 염려도 없으며 이 알고리즘을 비밀로 한 것과 안정성과는 문제가 없다는 것이었다. 그러나 대중은 여전히 어떤 문제점이 있지 않을까 생각하며 대중에 공개되어 오랜기간 조사를 할 수 있는 알고리즘을 원하며 산업계는 하드웨어 구현에 따른 비용문제와 정부가 제작한 칩을 사용하기 위해 전체 시스템이나 제품을 수정해야 될지도 모르는 문제 때문에 반대를 했다. 이에 대해 NIST는 key escrow를 소프트웨어적으로 구현하는 방법을 연구하기를 제안했고 제안된 방법도 있다[BELW].

비밀 알고리듬을 사용한다면 이 알고리듬의 노출을 막기 위해 반드시 tamper-proof 하드웨어 구현을 해야 하며 비밀 알고리듬을 사용하지 않는다면 굳이 하드웨어로 구현될 필요는 없다. 공개되어 있는 알고리듬을 사용하는 경우 관용키 알고리듬이든 공개키 알고리듬이든 상관이 없다. 그러나 공개키 알고리듬에서는 사용자의 공개키를 어떤 방식이든 관리하게 될 테고 이렇게 관리/등록할 때 kew escrow를

* 포항공과대학교 전자전기공학과

하면 된다. 그래서 공개키 알고리즘을 사용해서 개인의 사생활을 보호하고 법집행기관의 합법적인 복호화를 할 수 있는 방법이 제안되었다.

[한이]는 미국의 암호 정책을 Clipper 칩을 중심으로 소개하였는데 여기서는 공개키 알고리즘을 사용하는 key escrow system을 소개하고 이들을 비교 분석한다.

2. 용어 설명 및 기본 개념

이 장에서는 key escrow 시스템에서 일반적으로 쓰이는 용어와 몇 가지 기본 개념을 소개한다.

2.1 용어에 대한 정리

여기에 있는 용어는 특별히 언급하지 않으면 이 의미대로 쓰인다.

escrow 사용자의 비밀키를 어떤 조건(예를 들어 법원의 영장을 받는 것)을 만족했을 때 법집행기관에게 전달하기로 하고 신뢰받는 기관이 가지는 것

법집행기관 검찰, 경찰, 세무서 등 사회의 질서와 안녕을 위하여 법을 집행하고 필요한 사찰을 행할 수 있는 기관으로서, 더 많은 사람들의 안녕을 위해 정보사찰이 필요한 경우 법원의 영장을 받아 신뢰받는 기관에 제출하고 사용자의 비밀키를 얻어서 사용자의 암호화된 문서를 복호화한다.

신뢰받는기관 사용자의 비밀키의 일부분을 가지고 있는 곳으로 법집행기관의 합법적인 요청이 있는 경우에 이를 법집행기관에 제출한다.

키관리센타 사용자의 공개키를 관리하는 곳

법원의 영장 법원이 법집행기관의 충분한 증거 제시에 대하여 신뢰받는기관으로 하여금 법집행기관에 사용자의 비밀키를 복구하도록 하는 명령서

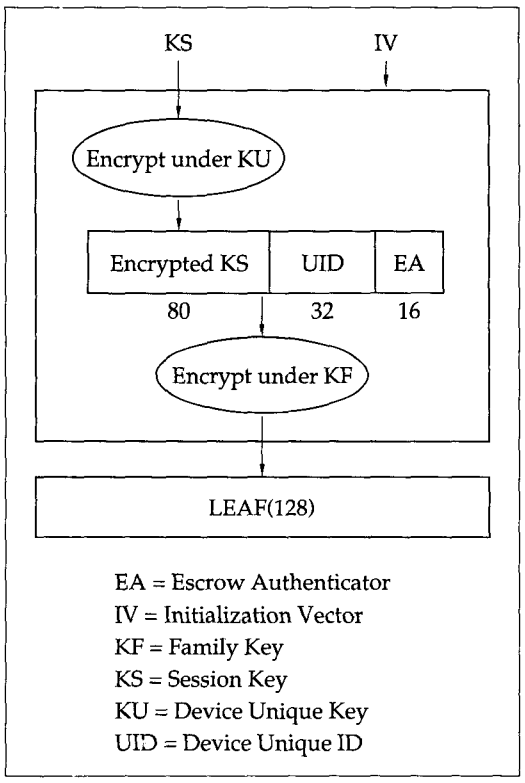
비밀채널 어떤 방법이든 통신의 당사자만이 주고받는 메시지의 내용을 볼 수 있는 채널

2.2 EES

EES[EES]에서는 KU(Device Unique Key)를 두 부분(각각은 최소한 80비트의 길이)으로 나누어 이것을 신뢰받는기관 2개에 각각 위탁하고 이들 신뢰받는기관은 법원의 영장이 있을 때 그들이 가진 KU의 부분을 법집행기관에 제공한다. 그리고 SKIPJACK 알고리듬이 암호화를 위해 사용되는데 이 알고리듬은 민감하나 비밀이 아닌(senstive but unclassified) 데이터의 암호화/복호화에 사용되는 알고리듬이며 데이터의 암호화/복호화에 이용될 때 세션키는 80비트이다. SKIPJACK 알고리듬과 Law Enforcement Access Field(LEAF)를 만드는 방법은 공개되어 있지 않으며 이들은 reverse engineering이 매우 어려운 IC 칩이나 하드웨어 디바이스에서 구현되어야 한다. 각 디바이스/IC 칩은 Device Unique Identifier (UID), 80 비트 KU, 80 비트 common Family Key(KF)를 갖고 있어야 한다. KU는 데이터의 암호화에는 쓰이지 않지만 LEAF에서 KF와 함께 법원의 영장이 있을 때 세션키를 알아낼 수 있게 한다.

LEAF는 모든 암호화된 데이터와 함께 보내져야 하며 암호화된 세션키를 가지고 있다.

LEAF의 수신자는 이것을 복호화해서 세션키를 알아낼 수 없으며 단지 법집행기관이 법원의 영장을 받았을때만 LEAF를 복호화해서 세션키를 알 수 있다. LEAF의 생성방법은 공개되어 있지 않은데 아래 그림은 대략적인 생성방법이다. 여기서 보면 세션키는 하드웨어 디바이스의 KU로 암호화되고 UID와 EA를 결합시켜 KF로 암호화한다.



〈그림 1. LEAF 생성방법〉

2.3 Secret Sharing과 Split Key Escrow

Blakley[Bl]와 Shamir[Sh1]은 비밀키를 n개의 부분으로 나누어 어떠한 t개의 부분을 모아도 비밀키를 복구할 수 있지만 t-1개의 부분이 주어지면 비밀키를 복구할 수 없는 방법을 제안했다(t는 n과 같은 값이거나 작은 값일 수 있다). 이것이 'secret sharing'인데 여기서의 문제점은 비밀키의 일부분을 받은 신뢰받는기관들이 이것이 실제로 사용자의 비밀키의 일부분임을 확인할 수 없는 데 있다. 그래서 verifiable secret sharing(VSS)[Pe1]을 사용하게 되는데 여기서 신뢰받는기관들은 자신이 사용자의 비밀키의 일부분을 받았음을 확인할 수 있다. 그 방법은 다음과 같을 수 있다. 사용자가 비밀인 m을 선택하고 m의 암호화된 형태인 $E(m)$을 계산해서 공개한다. 그리고 사용자는 m을 나눠서 신뢰받는기관에 준다. 그러면 각 신뢰받는기관은 공개된 $E(m)$에 대하여 확인해서 자신이 실제로 사용자의 비밀의 일부분을 받았는지 확인할 수 있다. 공개키 알고리즘을 이용하는 key escrow에 쓰이기 위해 $(m,E(m))$이 비밀키와 공개키의 쌍을 이루면 되는데 [Pe2]에서는 이산대수에 기초한 $(E(m)=g^m)$ VSS를 설명하고 있다.

그리고 사용자의 비밀키를 n개로 나누어 각각 신뢰받는기관에 맡기고 이들에게서 t개를 모아야 비밀키를 복구할 수 있는 방법을 split key escrow라고 하며 신뢰받는기관을 전적으로 믿을 수 없다는 생각과 이런 기관들이 혹시 키의 일부분을 잊어버릴 수도 있다는 생각(이 경우 t는 n보다 작은 값)에서 나타났다.

2.4 Bit Commitment

a가 b에게 어떤 내용을 맡기고(commit) 싶지만 일정 기간이 지나기 전까지는 그 내용을 알리고 싶지 않고, 한편 b는 a가 내용을 자신에게 맡긴 뒤 그 내용을 변화시킬 수 없기를 바라는 경우 bit commitment 프로토콜을 사용하게 된다. 이것은 다음의 두 단계로 이루어진다[Na].

commitment 단계 : a는 b에게 맡기고 싶은

비트 m이 있다. a와 b는 메시지를 교환하고 이 단계가 끝나면 b는 m을 나타내는 정보를 가지게 된다.

reveal 단계 : 이 단계에서 b는 m의 값을 안다.

이 프로토콜은 모든 확률적인 polynomial time 과 모든 polynomial p와 충분히 큰 안전도 변수 n에 대하여 commitment 단계가 끝난 후 b는 m의 값을 $1/2 + \frac{1}{p(n)}$ 보다 작거나 같은 확률로 추측할 수 있으며 a는 단지 한가지 가능한 값만을 알려 줄(reveal) 수 있어야 한다. 만약 a가 다른 값을 알려준다면 최소한 $1 - \frac{1}{p(n)}$의 확률로 알 수 있다.

3. 공개키 escrow 시스템

3.1 Fair Public-Key Cryptosystem (FPKC)

Micali의 FPKC[Mi]는 (1) 범죄자에 의해 잘못 사용될 수 없고 (2) 사용자들은 현재 법의 보호 아래 가지는 것과 같은 권리를 가지는 성질을 가진 공개키 암호 시스템이다. fair한 cryptosystem이란 민주사회에서 법집행기관의 합법적인 도청의 요구와 시민들의 사생활 보호사이에서 적절한 균형을 맞춘 것을 의미한다. 두 사용자가 FPKC S를 이용하여 암호문 C를 교환한다면 법에 의해 정해진 어떤 상황에서 적절한 제3자가 C의 평문을 얻을 수 있어야 하며 만약 얻을 수 없다면 두 사람이 그들의 비밀 통신에 S를 이용하고 있지 않음을 확인할 수 있다. [Mi]에서는 이런 성질을 가진 FPKC를 일반적인 공개키 암호 시스템(PKC)에서 만드는 방법을 설명하고 있다.

〈Diffie-Hellman(DH)[DiHe] PKC를 fair DH PKC로 만드는 프로토콜〉

p, 소수
g, 큰 위수를 가진 Z_p^*의 생성자
S, 사용자의 비밀키
$P=g^S \bmod p$, 사용자의 공개키
n은 신뢰받는기관의 수이고 t개이상의 신뢰받는기관이 모여야 비로소 사용자의 키를 복구할 수 있다.(아래의 예를 든 프로토콜에서 $n=5$이고 $t=3$이다)

단계 1. 사용자는 비밀키 S를 $[1,p-1]$에서 선택하고 공개키 $P=g^S \bmod p$를 계산한다 (앞으로 모든 연산은 mod p이다). 사용자는 1에서 5까지의 모든 가능한 조합 (1,2,3), (2,3,5), (1,2,4)···등을 고려한다. 각각의 (a,b,c)에 대하여 $S1abc$, $S2abc$, $S3abc$를 $[1,p-1]$에서 임의로 골라서 이들의 합이 S와 같도록 한다.

단계 2. 사용자는 $t1abc \equiv g^{S1abc}$, $t2abc \equiv g^{S2abc}$, $t3abc \equiv g^{S3abc}$를 계산한다. 신뢰받는기관 Ta에게 $t1abc$와 $S1abc$, Tb에게 $t2abc$와 $S2abc$, Tc에게 $t3abc$, $S3abc$를 (a,b,c)와 함께 비밀채널을 통하여 보낸다.

단계 3. 단계 2에서 사용자에게서 메시지를 받은 신뢰받는기관 Ta(다른 신뢰받는기관도 마찬가지 방법임)는 $t1abc \equiv g^{S1abc}$인지 확인하고 $(P, t1abc, (a,b,c))$를 서명해서 키관리센타에 보낸다.

단계 4. 키관리센타는 각 (a,b,c)에 대하여 신뢰받는기관에서 받은 서명에서 $t1abc$, $t2abc$, $t3abc$를 복구해서 이들의 곱이 P와 같은지 확인해서 같으면 P를 공개키로 받아들인다.

단계 2에서 각 신뢰받는기관이 받는 (a,b,c)를 고려해보자.

T1 (1,2,3) (1,2,4) (1,2,5) (1,3,4) (1,3,5) (1,4,5)
T2 (1,2,3) (1,2,4) (1,2,5) (2,3,4) (2,3,5) (2,4,5)
T3 (1,2,3) (1,3,4) (1,3,5) (2,3,4) (2,3,5) (3,4,5)
T4 (1,2,4) (1,3,4) (1,4,5) (2,3,4) (2,4,5) (3,4,5)
T5 (1,2,5) (1,3,5) (1,4,5) (2,3,5) (2,4,5) (3,4,5)

법원의 영장이 있어서 사용자의 비밀키를 복구해야 한다고 가정하자. 이때 신뢰받는기관 T1,T2,T3이 그들이 가지고 있는 (1,2,3)에 대해 ((1,2,3), *tiabc,Siabc*)를 주면 비밀키는 *Siabc*의 합이므로 법집행기관은 사용자의 비밀키를 알 수 있다. 그러나 신뢰받는 기관 2개로부터 (a,b,c)에 해당하는 내용을 받으면 사용자의 비밀키를 알 수 없다. 그러므로 최소한 신뢰받는 기관 3개가 그들이 가지고 있는 사용자의 비밀키에 대한 정보를 주어야 한다. 만약 신뢰받는기관 4개 T1,T2,T4,T5가 그들이 가지고 있는 모든 (a,b,c)를 주면 비밀키를 복구할 수 있는 것은 (1,2,4), (1,4,5), (2,4,5), (1,2,5)의 네가지 경우가 된다. 신뢰받는기관이 잘못된 정보를 줄 수 있다는 가정 때문에 신뢰받는기관이 사용자의 비밀키에 대한 정보를 줄 때는 모든 (a,b,c)에 대한 내용을 다 주게 되고 법집행기관은 최소한 하나의 (a,b,c)에 대하여 사용자의 비밀키를 알아낼 수 있게 된다. 이 프로토콜에 쓰인 방법을 subset method라고 하며 RSA[RSA]를 fair하게 만드는 방법에서도 쓰일 수 있다.

RSA[RSA]를 fair하게 만들기 위해 다음의 사실이 이용된다.

Fact 1. Z_N^*를 1과 N사이의 N과 서로 소인 정수의 곱셈군이라 한다. N이 두 소수의 곱 ($N=pq$ 이거나 $N=p^aq^b$)이면
Z_N^*에 있는 어떤 수 s가 네 개의 서로 다른 제곱근(square root mod N) x, $-x$, y, $-y$ ($x^2=y^2=s \bmod N$)을 가지고 있는 것은 s가 제곱수(square mod N)이기 위한 필요충분조건이다. 그리고 $i,j\in\{1,-1\}$에 대하여 $\gcd(ix+jy,N)$에서 쉽게 N을 소인수 분해할 수 있다.
Z_N^*에 있는 정수 중 4개중 1개는 제곱수이다.

Fact 2. Jacobi symbol은 Z_N^*에 있는 정수에 대하여 1또는 -1의 값을 가지며 $J_N(\cdot)$이라 나타낸다. $N=pq$이고 $p\equiv q\equiv 3 \bmod 4$이고 x, $-x$, y, $-y$를 어떤 제곱수의 제곱근이라 하고 $J_N(x)=J_N(-x)=1$이고 $J_N(y)=J_N(-y)=-1$이라 한다. 그러면 Fact 1에 의하여 제곱수의 Jacobi symbol이 1인 제곱근과 Jacobi symbol이 -1인 제곱근이 주어졌을 때 쉽게 N을 소인수 분해할 수 있다.

〈fair RSA 프로토콜〉

$p\equiv q\equiv 3 \bmod 4$, 사용자의 비밀키
$N=pq$, 사용자의 공개키
$n=5$, $t=5$

단계 1. 사용자는 Z_N^*에서 임의로 5개의 Jacobi symbol이 1인 X_i를 골라서 $X_i^2 \bmod N$을 계산해서 각 신뢰받는기관 T_i에게 모듈러스 N과 비밀 부분인 X_i를 비밀 채널을 통하여 보낸다. 그리고 $Z\equiv \prod_{i=1}^{5} X_i^2 \bmod N$이라 하고 Z의 Jacobi symbol -1인 제곱근 $Y(Z=Y^2,\ J_N(Y)=-1)$를 계산해서 키관리센타에 보낸다.

단계 2. 신뢰받는기관 T_i는 X_i와 N을 저장하고 $J_N(X_i)$이 1인지 확인하고 키관리센타에 $X_i^2 \bmod N$에 대한 서명을 보낸다.

단계 3. 키관리센타는 $J_N(-1)=1$인지 확인하고 $Y^2\equiv \prod_{i=1}^{5} X_i^2 \bmod N$인지 확인한다. 그리고 N이 두 소수의 곱임을 확인한다.

법원의 영장이 있으면 신뢰받는기관은 X_i를 키관리센타에 보내고 이런 X_i들의 곱은 $\prod X_i^2 \bmod N \equiv X^2$의 제곱근 $X \equiv \prod X_i^2 \bmod N$이 되며 $J_N(X)=1$이다. 그리고 키관리센타는 Jacobi symbol -1인 제곱근 Y를 가지고 있고 이 Y는 X, $-X$와 다르다($J_N(-1)=1$이므로 $J_N(-X)=J_N(X)=1$이 되고 $J_N(Y)=-1$이므로). 키관리센타가 X, Y를 가지고 있으므로 Fact 1,2에 의하여 N이 두 소수의 곱이라면 N의 인수를 가진 것과 같다. 그러므로 단계 3에서 N이 두 소수의 곱임을 확인해야 한다.

키관리센타는 N이 소수가 아님을 쉽게 검증할 수 있다. 그리고 양수 x, $y(y>1)$에 대하여 N이 x^y의 형태가 아님을 확인해서 N이 소수의 멱승이 아님을 검증할 수 있다. 그 다음에 N이 두 소수의 멱승의 곱($N=p^aq^b$, p와 q는 소수, a와 b는 양의 정수)임을 확인하면 된다. N의 소인수분해가 알려져 있지 않으면 이것을 확인할 수 있는 효율적인 알고리즘은 알려져 있지 않으므로 N의 소인수분해를 할 수 있는 사용자의 도움이 필요하게 된다. 그런데 N이 두 소수의 멱승의 곱이면 최소한 4개중 1개는 제곱수이며 멱승의 곱이 아니면 많아야 8개중 1개가 제곱수이다. 이 사실을 이용해서 N이 두 소수의 멱승의 곱임을 증명하게 된다. 키관리센타는 여러개의 난수를 만들어 내고 사용자는 이에 대하여 최소한 전체갯수의 3/16의 제곱근을 키관리센터에 보낸다. 난수를 만들어 내는 과정은 사용자와 키관리센타가 동시에 할 수도 있는데 사용자가 선택한 N을 해쉬함수에 넣어 해쉬결과를 난수발생기의 입력으로 한다. 그래서 얻은 난수($S(N)$이라고 하자)에 대하여 최소한 전체의 3/16개의 제곱근을 키관리센타에 보낸다. 키관리센타는 N을 알고있으므로 같은 난수를 발생시킬수 있고 사용자에게 받은 제곱근을 제곱해서 이들이 $S(N)$에 있고 전체수의 3/16 이상을 차지하는지 확인해서 N이 두 소수의 멱승의 곱임을 확인할 수 있다. 예를 들어 N을 세 소수의 곱으로 정한 사용자는 $S(N)$에 대하여 3/16개의 제곱근을 찾을 수 없다. 이렇게 난수를 발생해서 N이 두 소수의 멱승의 곱임을 증명하는 것은 키관리센타와 사용자간에 상호작용이 필요없으므로 단계 1에서 사용자가 키관리센타에 Jacobi symbol -1인 제곱근을 보낼 때 같이 보내면 된다.

fair DH 프로토콜은 사용자가 신뢰받는기관, 키관리센타간에 한 번만 데이터를 보내면 되므로 상호작용이 없으며 fair RSA 프로토콜 역시 한 번만 데이터를 보내면 되지만 N이 두 소수의 곱임을 키관리센타가 확인을 해야하므로 이것에 대한 증명을 사용자가 키관리센타에 제곱근을 보낼 때 같이 해야 한다. 그래서 fair RSA의 경우 사용자와 키관리센타의 계산량이 많다. 그리고 신뢰받는기관에 주어진 비밀키의 부분들이 실제로 비밀키의 일부분임은 신뢰받는기관에서 확인할 수가 없고 키관리센타가 확인을 해야 하므로 키관리센타의 일이 많아진다. split key escrow의 경우 ${}_nC_t$개의 $(a_1,a_2,\ldots,a_t)$가 있고 t개의 신뢰받는기관에 각각 $\binom{n}{t} \times \frac{t}{n}$개의 $(a_1,a_2,\ldots,a_t)$에 해당하는 비밀키의 일부분과 이에 대응되는 공개키의 부분을 보내야 하고 신뢰받는기관은 한 사용자에 대하여 $\frac{\binom{n}{t} \times t}{n}$ × 모듈러스 p길이의 2배(비밀키와 이에 해당하는 공개키의 부분은 각각 모듈러스 p의 길이와 같다)만큼의 내용을 저장하고 있어야 하므로 통신량과 저장량이 n이 커지면 매우 많아지게 된다. 그리고 [KiLe]에서 FPKE가 fair하지 않음을 보였다.

3.2 Fail Safe Cryptosystem

PKC에서 사용자들은 키(비밀키와 공개키)

를 자신이 선택하게 된다. 이런 경우 범죄자들은 신뢰받는기관에 맡겨 놓은 키 대신에 다른 키를 이용할 수도 있다. 예를 들어 공개키와 비밀키의 쌍이 (P,S)일 때 $(P'=f(P),S')$를 이용하여 법집행기관이 복호화할 수 없는 메시지를 주고받을 수 있다. 여기서 f는 쉽게 계산될 수 있고 잘 알려진 함수이다. 범죄자는 (P,S)를 일반 사용자처럼 이용하면서 S'를 자신의 shadow secret key라고 저장한다. 어떤 사람이 이 범죄자에게 법집행기관이 복호화할 수 없는 메시지를 보내고자 한다면 $P'=f(P)$를 계산해서 P'를 이용해서 암호화해서 보낸다. 그러면 S'를 이용해서 복호화 할 수 있다. 이 방식에서 중요한 점은 P'를 계산하기 쉬운 함수 f와 P, S, P', S'를 만들어내는 효과적인 방법을 찾는 것이다. 이런 방식의 예로 RSA에서는 모듈러스 N의 상위 부분에 실제로 사용할 키를 묻어 둘 수 있다. 이런 시스템을 shadow public key system이라 한다. shadow public key system을 피하기 위해 Kilian과 Leighton[KiLe]은 키를 정할 때 신뢰받는기관/키관리센타가 사용자와 상호작용을 하는 방식-failsafe key escrow(FKE)-을 제안했다.

FKE는 다음의 다섯 가지 성질을 갖는다.

(1) 모든 신뢰받는기관과 키관리센타가 법을 지키지 않더라도 사용자는 자신의 비밀키를 안전하게 선택하기 위해 충분히 조절할 수 있다.

(2) 키관리센타는 사용자가 난수를 사용하지 않더라도(사용자가 좋은 난수발생기를 가지고 있지 않아서 전화번호나 생일이나 기타 쉬운 수를 사용하더라도) 비밀키가 안전하게 선택되었음을 보장받을 수 있다.

(3) 각 사용자는 특정한 수의 신뢰받는기관이 키관리센타에 그들이 알고 있는 정보(사용자의 비밀키에 대해서 갖고 있는 부분 정보)를 제공할 때만 자신의 키가 공개되고 그렇지 않은 경우 안전함을 보장받는다.

(4) 키관리센타는 특정한 수의 신뢰받는기관에서 사용자의 키에 대한 정보를 받으면 사용자의 비밀키를 얻을 수 있음을 확신한다.

(5) 키관리센타는 범죄자들이 합법적인 도청이 불가능하게 escrow 시스템을 사용하지 않는다는 것에 대한 확신을 한다. 즉 키관리센타가 법죄자들의 메시지를 복호화할 수 있거나 만약 복호화할 수 없다면 이들은 비밀 통신에 다른 수단을 사용하고 있는 것이다. 이 성질을 shadow-public-key resistence라고 한다.

〈DH나 DSS와 같은 이산 대수에 기초한 PKC의 FKE 프로토콜〉

p, 소수 모듈러스, 공개되어 있음
g, Z_p^*의 생성자, 공개되어 있음

단계 1. 신뢰받는기관/키관리센타가 $[0, p-1]$에 있는 B를 임의로 골라서 사용자에게 chameleon property를 가진 information theoretically secure commitment 프로토콜을 이용하여 맡긴다.

단계 2. 사용자는 $[0, p-2]$에서 A를 임의로 골라서 $g^A \bmod p$를 계산하여 이 값을 신뢰받는기관/키관리센타에 보낸다.

단계 3. 사용자는 A를 신뢰받는기관들과 VSS를 통해서 공유한다.

단계 4. 신뢰받는기관/키관리센타는 B를 사

용자에게 알려주고 사용자는 단계 1에서 받은 것인지 확인한다. 신뢰받는기관/키관리센타는 공개키를 $P=(g^A)g^B \bmod p$라고 한다. B는 키관리센타에 escrow되며 공개되지 않는다.

단계 5. 사용자는 비밀키를 $S=A+B \bmod (p-1)$이라 한다.

사용자의 비밀키를 복구할 때는 충분한 수의 신뢰받는기관이 A에 대한 정보를 모아서 A를 복구하고 B는 신뢰받는기관/키관리센타에 escrow되어 있으므로 A와 B를 더해서 사용자의 비밀키를 알 수 있다. 여기서 A는 VSS를 통하여 공유되었으므로 신뢰받는기관이 사용자의 비밀키의 정보를 가지고 있는 것이 확인되고 B는 원래 신뢰받는기관/키관리센타가 생성해서 사용자에게 맡긴 값이었다.

FKE는 공개키 암호 시스템에서 subliminal channel[Si]을 만들지 못하게 하는 일반적인 방법이며 다른 key escrow 시스템에서 이 방법을 이용할 수 있다. 그러나 FKE는 범죄자들이 비밀 정보를 이용하거나 다른 escrow 시스템을 이용해서 비밀 통신을 하는 것을 막는 것이 아니며 비밀키를 정하는 것에 다른 프로토콜을 사용하지 못하게 하는 것도 아니다. 이것은 범죄자들이 key escrow 시스템의 공개키를 잘못된 방식으로 사용하는 것을 막는 것이다. FKE는 사용자와 키관리센타 모두에게 비밀키가 난수라는 확신을 주는데 사용자가 난수를 선택하지 못했다 하더라도 키관리센타가 적절한 난수를 선택하게 되므로 알아내기 쉬운 비밀키가 선택되지 않는다.

FKE가 사용자가 혼자 공개키와 비밀키를 정할 수 없게 해서 shadow public key 시스템을 만들 수 없게 하고 일반적인 공개키 암호키 시스템에 쓰일 수 있지만 공개키와 비밀키를 만드는 것에 신뢰받는기관/키관리센타가 참여함으로 인해 사용자와의 통신횟수가 VSS를 제외하더라도 3번으로 많다. 신뢰받는기관/키관리센타가 저장해야 될 양은 적은 편이다.

3.3 Key Escrow System with Warrant Bounds

EES에서는 키가 법집행기관에 주어질 때 기간에 대한 제한이 있으며(보통 30일) 이 기간이 지나면 키는 법집행기관에서 없어지게 되어 있지만 이것을 보장할 수 있는 방법은 없다. 그러므로 일정한 기간동안만 법집행기관이 개인의 데이터를 복호화할 수 있는 수단이 간구되어야 한다. Lenstra, Winkler와 Yacobi[LWY]는 이런 문제를 해결하기 위해 세션키를 만들 때 날짜에 대한 정보가 들어가게 하는 방법을 제안하였으며 이와 더불어 특정한 사람의 모든 통신을 복호화할 수도 있고(node surveillance) 단지 두 사람만의 통신을 복호화할 수 있는 방법(edge surveillance)을 제안하였다.

〈time warrant를 주는 key escrow 프로토콜〉

p와 q, 큰 소수로서 $q|p-1$
$g \in Z_p^*$, 위수는 q
$S(a) \in Z_q$, 사용자 a의 비밀키
$P(a) \equiv g^{S(a)} \bmod p \in Z_p^*$, 사용자 a의 공개키
f, 블록 암호화 함수로 비밀키를 k로 하여 암호문 $c=f_k(m)$를 형성
h, $Z_p \times Z_p \rightarrow Z_p$인 일 방향 해쉬함수로 어떠한 정수 i에 대해서도 $d(d_i \neq d)$와 $y_i=h(x,d_i)$가 주어졌을 때 어떤 알려져 있지 않은 x에 대해서 $y=h(x,d)$를 찾는 것이 계산상 불가능하다.
사용자들은 특정한 시간(예를 들어 UTC)을 사용하고 날짜 $d \in Z_p$이다.

단계 1. 사용자 a는 $k(a,b,d)=h(P(b)^{S(a)},d)$를 계산하고 사용자 b는 $k(b,a,d)=h(P(a)^{S(b)},d)$를 각각 계산한다. $k(a,b,d)=k(b,a,d)$가 되고 세션키가 된다.

단계 2. 사용자 a와 b는 법집행기관이 세션키를 계산할 수 있도록 하기 위한 메시지를 다음과 같은 방법으로 교환한다.

(1) a는 $S(a,d)=h(S(a),d)$, $S(a,b,d)=h(S(a,d),P(b))$를 계산한다.
b는 $S(b,a,d)=h(S(b),d)$, $S(b,a,d)=h(S(b,d),P(a))$를 계산한다.

(2) a는 $c(a,b,d)=f_{S(a,b,d)}(k(a,b,d))$를 b에게 보내고 b는 $c(b,a,d)=f_{S(b,a,d)}(k(b,a,d))$를 a에게 보낸다.

단계 3. 사용자 a와 b는 세션키 $k(a,b,d)$를 사용한 블록 암호화 함수 f를 이용하여 통신을 한다.

사용자가 세션키를 만들기 위해서 하는 계산에서 $P(b)^{S(a)}=g^{S(a)S(b)}$의 계산은 시간이 많이 걸리므로 자주 통신을 하는 사용자에 대해서는 미리 이 부분을 계산해두고 날짜가 변함에 따라 시간이 적게 걸리는 해쉬함수만 계산하면 된다.

사용자 a에 대해서 edge surveillance을 해야 할 때 신뢰받는기관은 a의 모든 상대방 b(법집행기관이 법원의 영장을 받은)에 대하여 $S(a,b,d)$를 법집행기관에 주며 node surveillance를 해야 할 때는 $S(a,d)$를 법집행기관에 준다. 법집행기관이 $S(a,b,d)$를 받으면 단계 2에서의 c를 복호화해서 세션키 $k(a,b,d)$를 얻을 수 있으며 $S(a,d)$를 받았다면 $S(a,b,d)=h(S(a,d),P(b))$이므로 필요한 모든 b에 대하여 $S(a,b,d)$를 계산해서 c를 복호화해서 세션키 $k(a,b,d)$를 알아낸다. 그런데 단계 2에서 a가 속이기 위해서 잘못된 c'를 보낸다면 법집행기관은 신뢰받는기관으로부터 받은 $S(a,b,d)$ 혹은 $S(a,d)$로 세션키 k를 얻을 수 없으므로 신뢰받는기관은 법집행기관에 사용자 a의 비밀키 $S(a)$를 주어야 한다.

e-mail이나 FAX같은 일 방향 통신에 사용되는 경우 서로 암호문 c를 보내는 과정이 수정된다. 즉 메시지를 보내는 쪽만 c를 계산해서 보내는 것이다. 이렇게 하면 일 방향 통신에도 쓰일 수 있지만 법집행기관에 제출하는 비밀키가 양방향 통신과는 달라진다. edge surveillance의 경우 법원의 영장이 적용되는 b에 대해서 사용자 a가 메시지를 받기만 하고 보낸 적이 없다면 신뢰받는기관은 $S(a,b,d)$가 아니라 $S(b,a,d)$를 법집행기관에 주어야 한다. 사용자 b는 $c(b,a,d)=f_{S(b,a,d)}(k(b,a,d))$를 보내고 사용자 a는 $c(a,b,d)$를 보내지 않았기 때문에 법집행기관은 $S(b,a,d)$가 있어야 $c(b,a,d)$에서 세션키 $k(b,a,d)$를 복구할 수 있다. a가 메시지를 보낸 적이 있는 다른 사용자 b'에 대해서는 $S(a,b',d)$를 법집행기관에 주어야 $c(a,b',d)=f_{S(a,b',d)}(k(a,b',d))$에서 세션키 $k(a,b',d)$를 복구할 수 있다. node surveillance의 경우도 $S(a,d)$만을 법집행기관에 준다면 a에게서 나가는 모든 메시지는 복호화할 수 있지만 a가 메시지를 받기만 하고 보낸적이 없다면 이런 사용자 b에 대해서는 $S(b,a,d)$를 주어야 한다.

split key escrow를 위해서 verifiable threshold secret sharing[Pe2]을 이용하게 된다. $\{1,2,\ldots,n\}$의 순서 있는 부분집합(ordered subset) $V=\{v_1,v_2,\ldots,v_t\}$에 대해 다음을 만족하는 쉽게 계산될 수 있고 공개되어 있는 상수 $b_i(V)$가 존재한다고 가정한다.

$$S(a)=\sum_{i=1}^{t} b_i S_{v_i}(a).$$

$\{v_1,v_2,\ldots,v_t\}=\{1,2,\ldots,t\}$이라 하고 $b_i(V)$를 b_i라 한다. 이런 경우 앞의 프로토콜에서 단계 2만 변

화되는데 사용자 a는 $S(a,d)=h(d,d)^{S(a)}$, $S(a,b,d)=h(d,P(b))^{S(a)}$를 계산하고 $c=f_{S(a,b,d)}(k(a,b,d))$와 $c'(a,b,d)=f_{S(a,d)}(k(a,b,d))$를 b에게 보낸다. 사용자 b 역시 같은 방법을 사용한다.

node surveillance의 경우 법집행기관은 $S_i(a,d)=h(d,d)^{S_i(a)}$를 받아서 $S(a,d)=\prod_{i=1}^{t} S_i(a,d)^{b_i}$를 계산해서 $c'(a,b,d)$에서 $k(a,b,d)$를 얻을 수 있다. edge surveillance의 경우 $S_i(a,b,d)=h(d,P(b))^{S_i(a)}$를 받아서 $S_i(a,b,d)=\prod_{i=1}^{t} S_i(a,b,d)^{b_i}$를 계산해서 $c(a,b,d)$에서 $k(a,b,d)$를 얻는다. a가 단계 2에서 속인다면 t개의 신뢰받는기관은 $S_i(a)$를 법집행기관에 주고 법집행기관은 이것에서 $S(a)=\sum_{i=1}^{t} b_i S_i(a)$를 얻는다. 만약 신뢰받는기관 t개가 모여서 사용자의 비밀키 $S(a)$를 계산해서 법집행기관에 $S(a,d)$ 혹은 $S(a,b,d)$를 제공한다면 최소한 어느 하나의 신뢰받는기관은 사용자의 비밀키 $S(a)$를 알게 된다. 그러므로 신뢰받는기관이 직접 $S(a)$를 계산하지 말아야 한다. 위의 변형된 프로토콜에서는 $S(a,d)$와 $S(a,b,d)$를 계산할 때 $S(a)$에서 계산하는 것이 아니라 신뢰받는기관이 자신이 가지고 있는 $S_i(a,d)$혹은 $S_i(a,b,d)$를 법집행기관에 보내서 법집행기관이 $S(a,d)$나 $S(a,b,d)$를 계산하게 된다.

단계 2에서 사용자 a가 계산하는 $h(d,d)^{S(a)}$는 통신의 상대방에 따라 변화하는 값이 아니므로 매일매일 계산해두고 필요한 경우 쓰면 되지만 $S(a,b,d)=h(d,P(b))^{S(a)}$의 계산은 날짜와 상대방에 따라 달라지며 계산량이 많다. 그리고 일방향 통신에 사용되어서 메시지를 보내는 쪽만 c와 c'를 보낸다면 신뢰받는기관이 한 개인 경우처럼 node 혹은 edge surveillance의 경우에 법집행기관에 주어야 하는 비밀키에 대한 정보가 바뀐다.

[LWY]에서 제안하는 프로토콜은 통신하기 전에 보내야 하는 암호문(c)을 통신을 시작하면서 같이 보내도 되기 때문에 세션키를 형성하는데 상호작용을 하지 않는다. 그러나 split key escrow에서는 두 개의 암호문을 보내야 한다. 그리고 일 방향 통신에 응용되어서 특정한 사용자 a를 관찰해야 할 때 a가 메시지를 받기만 하고 보내지 않는다면 보낸 사람의 비밀키의 일부분을 신뢰받는기관이 제공해야 되므로 DH 형태의 PKC의 세션키의 형성이 쉽고 복구하기 쉽다는 관점에서의 장점이 사라지게 되며 특별히 node와 edge surveillance를 구분한 것이 의미가 없어진다. 그리고 통신하기 전에 보내는 암호문 c는 전적으로 법집행기관이 암호문에서 평문을 얻을 수 있도록 하는 기능이다. 즉 사용자들은 이 암호문 c에 상관없이 세션키를 만들 수 있다. 이렇게 되면 사용자들은 암호문 c를 보낼 필요가 없으므로 이것이 없을 때는 수신자가 받은 메시지를 복호화할 수 없도록 하는 수단이 있어야 한다.

3.4 Guaranteed Partial Key Escrow

지금까지 설명된 key escrow는 (1) 신뢰받는기관은 비밀키의 일부분을 받았음을 확인할 수 있고 (2) 충분한 수의 신뢰받는기관이 협력하지 않으면 비밀키를 쉽게 복구할 수 없고 (3) 충분히 많은 수의 신뢰받는기관이 협력해야 비밀키 전체를 복구할 수 있었다. 그런데 CRYPTO 95에서 Adi Shamir[Sh2]는 partial key escrow라고 하여 사용자의 비밀키의 일부분만을 escrow하는 방법을 제안했다. 여기에서는 사용자의 비밀키 전체가 escrow되는 것이 아니므로 신뢰받는기관이 사용자의 비밀키의 일부분을 복구한 뒤 완전한 비밀키를 복구하기 위해서는 무시할 수는 없지만 일정한 계산을 해야 한다.

Micali는 이러한 partial key escrow에 신뢰받는기관이 받은 비밀키의 일부분이 실제로 사용자의 비밀키의 일부분임을 보증할 수 있어야 한다고하여 Guaranteed Partial Key Escrow를 제안했다.

〈DH를 위한 GPKE 프로토콜〉

p, 512비트의 소수
g, Z_p^*에서 생성자이며 모두에게 알려져 있음
$x \in [1,p-1]$, 사용자의 비밀키
$P=g^x \bmod p$, 사용자의 공개키

단계 1. 사용자는 비밀키 $x=z$(512 비트 수)$+y$(80 비트 수)라 한다.

단계 2. 사용자는 신뢰받는기관(1개)에 $g^y \bmod p$, $g^z \bmod p$, z를 비밀채널을 통하여 보내고 y가 x에서 작은 부분을 차지한다는 것을 영지식 증명을 한다.

단계 3. 신뢰받는기관은 $g^x \equiv g^z g^y \bmod p$인지 확인을 한다. 그리고 $g^z \bmod p$를 계산하여 받은 $g^z \bmod p$와 일치하는지 확인한다.

위의 프로토콜은 신뢰받는기관이 1개인 경우이며 법원의 영장을 받아서 사용자의 비밀키를 알아야 할 경우 신뢰받는기관은 $g^z \bmod p$를 알고 있으므로 $P/g^z=g^y$임을 알고 y를 exhaustive search로 구해 내서 비밀키 $S=x+a$임을 알 수 있다. 그런데 신뢰받는기관이 사용자의 비밀키의 작은 부분 y를 알아내기 위한 계산을 언제라도 할 수 있다. 이런 경우를 early recovery[BeGo]라고 하며 법원의 영장이 없이 미리 계산을 해서 사용자의 완전한 비밀키를 갖고 있을 수도 있기 때문에 실제적으로 사용자의 비밀키를 복구하기 위한 시간적인 지연은 없는 것과 같다. GPKE는 신뢰받는기관이 n개인 경우로 확장될 수 있는데 이 때 z를 $z_1,z_2,...,z_n$의 합(mod p)이라 하고 각각의 신뢰받는기관에 z_i를 비밀채널을 통하여 보내고 $g^{z_i} \bmod p$를 공개한다. 신뢰받는기관들은 z_i가 $g^{z_i} \bmod p$의 이산 대수임을 확인한다. 그리고 모든 g^{z_i}의 곱이 g^z인지 확인해야 한다. split key escrow를 위해서는 [Mi1]의 방법 중의 하나를 쓰면 된다. 모든 경우에 사용자의 공개키와 escrow 되는 비밀키의 부분에 대한 것(위의 프로토콜에서 g^z)이 공개되기 때문에 키관리센타(g^z를 확인하는 기관) 혹은 신뢰받는기관(신뢰받는기관이 1개인 경우)은 언제라도 escrow 되지 않은 비밀키의 부분을 복구할 수 있어서 early recovery의 문제점이 있다.

〈RSA를 위한 GPKE 프로토콜〉

단계 1. 사용자는 모듈러스 $N=p_1p_2p_3p_4$라 하는데 여기서 p_1, p_2는 큰 소수(512비트)이고 p_3, p_4는 150비트 정도의 작은 소수이다. 그리고 N과 p_1, p_2를 신뢰받는기관에 보낸다.

단계 2. 신뢰받는기관은 N이 1324비트이고 p_1, p_2가 512비트의 서로 다른 소수이고 N을 나누는지 확인한다.

일반적으로 RSA에서 N은 두 큰 소수의 곱으로 구성되지만 RSA를 위한 GPKE에서는 큰 소수 두 개(약 512비트)와 작은 소수 두 개(약 150 비트)의 곱으로 구성된다. [CGMA]에서는 RSA의 공개키의 두 개의 소수가 비밀로 남아 있으면 시스템의 안전도는 공개키가 두 개의 소수의 곱으로 된 RSA 시스템의 안전도와 같다는 것을 증명하였다. 따라서 신뢰받는기관은 공개키가 p_3, p_4의 곱인 RSA 암호 시스템을 대하게 되지만 일반 사용자들은 1324비트의 RSA시스템을 대하게 되고 이 시스템의 안전도는 최소한 일반적인 RSA 시스템(공개키가 1024비트인)과 같게 된다.

위에서 신뢰받는기관은 1개이지만 신뢰받는기관이 전체 비밀키를 갖고 있는 것이 아니다. [Mi2]는 신뢰받는기관이 1개라도 전체 비밀키를 가지고 있는 것이 아니므로 split key escrow개념을 중요하게 생각하지는 않고 있다.

그렇지만 이 GPKE는 임의의 개수의 소수와 많은 신뢰받는기관으로도 구현될 수 있다. 예를 들어 N이 6개의 소수의 곱이고 그 중 4개는 512비트이고 두 개는 150비트 길이라면 네 개의 신뢰받는기관이 있을 수 있다. 이 중에서 최소한 3개의 신뢰받는기관이 모여야 적당한 노력으로 N을 소인수 분해 할 수 있다.

RSA를 위한 GPKE에서 신뢰받는기관이 1개인 경우에 1324비트의 키 중에서 1024비트를 공개키(모듈러스 N은 큰 소수 2개와 작은 소수 2개의 곱인데 큰 소수 2개를 알고 있으므로)를 받자 말자 알아낼 수 있으므로 early recovery의 문제점이 있다. 그리고 신뢰받는기관이 여러 개인 경우 키의 크기가 매우 커진다는 단점이 있다.

3.5 Verifiable Partial Key Escrow System

Shamir[Sh2]가 제안한 partial key escrow에 대해 Micali[Mi2]는 GKPE를 제시했고 Bellare와 Goldwasser[BeGo]는 verifiable partial key escrow(VPKE) 시스템을 제시했으며 GPKE의 문제점인 early recovery를 지적했다. VPKE 시스템은 사용자의 비밀키의 일부분을 신뢰받는기관에 escrow해서 법집행기관이 신뢰받는기관이 가진 비밀키의 일부분을 얻었을 때 사용자의 완전한 비밀키를 얻기 위해 약간의 계산을 해야하는데 이 계산량이 미리 정한 일정한 계산량(예를 들어 240)보다 많지 않음이 증명될 수 있는 시스템이다. 그리고 이러한 일정한 계산을 법집행기관이 언제 할 수 있는 지의 문제가 early recovery와 delayed recovery를 결정한다. 만약 사용자의 공개키가 공개되자 말자 법집행기관이 사용자의 비밀키의 일부분을 모르는 상태에서 이 계산을 할 수 있다면 early recovery이다. 이렇게 되면 사용자의 비밀키를 알기 전에 일정한 계산을 미리 해서 저장해두고 신뢰받는기관에서 사용자의 비밀키를 받으면 미리 계산해 둔 것을 참조해서 사용자의 완전한 비밀키를 알아낼 수 있다. early recovery의 경우도 시간적인 지연을 법집행기관에 주기는 하지만 사용자의 비밀키를 알기 전이므로 실제적인 지연은 아니다. 그리고 early recovery는 비밀키를 모두 신뢰받는기관에 맡기는 것(standard key escrow)보다 더 나쁘다. 그 이유는 사용자들은 partial key escrow가 자신의 비밀키에 대한 여분의 안전을 제공한다고 믿기 때문이다. 한편, 신뢰받는기관이 사용자의 비밀키의 일부분을 준 이후에도 사용자의 완전한 비밀키를 얻기 위해 반드시 일정한 시간/계산이 필요한 기법을 delayed recovery라 한다. VPKE 시스템은 delayed recovery를 제공한다.

〈DH를 위한 VPKE 프로토콜〉

$p=2q+1$의 소수(최소한 512비트), q 역시 소수임

$G\subseteq Z_p^*$, 소수 위수 q를 갖는 군

$g\in Z_p^*$, 군 G의 임의의 생성자

$h\in Z_p^*$, $\log_g(h)$가 알려져 있지 않은 군 G의 임의의 생성자

$S=x+a \bmod q$, 사용자의 비밀키로 x는 비밀키의 길이가 긴 부분이며 a는 비밀키의 길이가 작은 부분

$P=g^{x+a}\in G$, 사용자의 공개키

$a=a_0 2^0+a_1 2^1+\cdots+a_{2l-1}2^{2l-1}$, $a_i\in\{0,1\}$, 비밀키의 짧은($2l$비트의) 부분

$x\in Z_q$, S의 긴 부분으로 신뢰받는기관에 escrow됨

n은 신뢰받는기관의 수이고 t는 사용자의 비밀키를 복구할 수 있는 최소의 신뢰받는기관의 수이다.

단계 1. 사용자는 $u \in_R Z_q$, $u_0,...,u_{2l-1} \in_R Z_q$하고 $X=g^u h^u$, $A_i=g^{a_i}h^{u_i}$을 $i=0,1,...,2l-1$에 대하여 계산하여 X, $A_0,...,A_{2l-1}$을 신뢰받는기관에 보낸다.

단계 2. 사용자는 $w=u_0 2^0+u_1 2^1+...+u_{2l-1}2^{2l-1} \bmod q$를 신뢰받는기관에 보내고 신뢰받는기관은 $P \cdot h^w = X \cdot A_0^{2^0} \cdot A_1^{2^1} ... A_{2l-1}^{2^{2l-1}}$ 인지 확인한다.

단계 3. 사용자는 x를 단계 1의 X에 기초한 Pederson의 information theoretically secure VSS[Pe]를 통하여 신뢰받는기관과 공유한다.

단계 4. 사용자는 각 A_i들이 0또는 1에 대한 committal인지 증명한다.
각 신뢰받는기관 T와 $i=0,...,2l-1$에 대하여 사용자와 신뢰받는기관은 $100/(n-t)$번을(직렬/병렬적인 방법으로) atomic proof of knowledge of de-committal of A_i-공통의 입력 A_i에 대하여 아래의 KOD 프로토콜을 이용해서-를 한다. 여기서 사용자가 증명자이고 신뢰받는기관이 검증자이다.

〈KOD 프로토콜〉

공통의 입력: $A \in G$, and $g,h \in G$
증명자의 지식: $A=g^b h^y$인 b와 h에 대해 $b \in Z_2$, $y \in Z_q$
증명자의 주장: $A=h^y$이거나 $A=gh^y$인 y를 알고 있음

단계 1. 증명자는 $r \in_R Z_2$, $v_1, v_2 \in_R Z_q$를 하고 $r_1=r$, $r_2=1-r$이라 한다. $R_1=g^{r_1}h^{v_1}$, $R_2=g^{r_2}h^{v_2}$이라 하고 R_1, R_2를 검증자에게 보낸다.

단계 2. 검증자는 난수 $c \in Z_2$를 증명자에게 보낸다.

단계 3. c의 값에 따라
c가 0이면 증명자는 r, v_1, v_2를 검증자에게 보내고 검증자는 $R_1=g^{r_1}h^{v_1}$, $R_2=g^{1-r}h^{v_2}$임을 확인한다.
c가 1이면 $b=r_j$, $j \in \{1,2\}$라 하고 j와 $w=y-v_j=\log_h(A/R_j)$를 검증자에게 보내고 검증자는 $h^w=A/R_j$인지 확인한다.

DH를 위한 VPKE 프로토콜에서 공개키는 $P=g^{x+a}$이고 x가 비밀키의 큰 부분이며 a는 작은 부분이다. 사용자는 x와 a를 비밀키로 저장하고 x를 escrow하고 P의 형태를 증명한다. g^x나 g^a를 공개하지 않고 information theoretic secure VSS[Pe2]를 이용하여 위임하고(commit) 이 위임(commitment)에 따라 a가 작은 부분임을 증명한다.

information theoretic secure bit commitment에 대해 알아보면 다음과 같다. 먼저 G, g, h를 위의 프로토콜에 정의된 것으로 하자. 위임자(committer)는 Z_q에서 임의로 v를 골라서 commitment를 $Z=g^z h^v$라고 하여 Z_q의 어떤 z에 대해서도 위임할 수 있다. 이 Z는 각각의 z에 대하여 G에 균일하게 분포되어 있으므로 이 값을 받은 사람(수신자)은 z에 대한 어떤 정보도 얻을 수 없다. Z를 decommit하기 위해 위임자는 $Z=g^z h^v$가 되는 z와 v를 제시해야 한다. 만약 위임자가 $Z=g^{z_1}h^{v_1}=g^{z_2}h^{v_2}$가 되는 $z_1 \neq z_2$와 v_1, z_2를 찾는다면 수신자를 속일 수 있지만 이것은 위임자가 $\log_g(h)$를 모르므로 불가능하다. 그 이유는 다음과 같다. 앞의 식은 $g^{z_1-z_2}=h^{v_2-v_1}$을 의미하고 $z_1 \neq z_2$이므로 v_2-v_1은 0이 아니고 이것은 $v_2 \neq v_1$임을 나타낸다. v_2-v_1은 0이 아니므로 Z_q에서 역 β가 존재하고 양변을 β 멱승을 하면 $(g^{z_1-z_2})^\beta=(h^{v_2-v_1})^\beta$, $(v_2-v_1)\beta=1$이므로 $h=(g^{z_1-z_2})^\beta$이고 $(z_1-z_2)\beta=\log_g(h)$가 된다. 그러므로 $\log_g(h)$를 알아야 $z_1 \neq z_2$인 z_i를 얻을 수 있다.

DH를 위한 프로토콜 단계 1에서 비밀키의 부분은 information theoretically secure 위임(commitment)방법을 이용하여 각각 위임된다. 비밀키의 큰 부분인 x는 블록으로 commit되고 (사용자가 $u\in_R Z_q$해서 $X=g^x h^u$라 함) 비밀키의 작은 부분인 a는 비트별로 commit된다 ($A_i=g^{a_i}h^{u_i}$이 $i=0,...,2l-1$에 대해 계산되고 신뢰받는기관에 보내진다). 단계 2에서 사용자는 위임된 값이 자신의 비밀키 $x+a$임을 증명한다. 만약 사용자가 x와 a_i에 대하여 안다면

$$P=g^{x+a_0 2^0+a_1 2^1+...+a_{2l-1}2^{2l-1}}$$
$$=(X/h^u)\prod_{i=0}^{2l-1}(A_i/h^{u_i})^{2^i}$$

가 된다. 그런데 사용자는 $w=u+\prod_{i=0}^{2l-1}u_i 2^i$를 신뢰받는기관에 보냈으므로 각 신뢰받는기관은 $P\cdot\{h^u\prod_{i=0}^{2l-1}(h^{u_i})^{2^i}\}=P\cdot h^w=X\cdot\prod_{i=0}^{2l-1}A^i$을 확인할수 있다. 단계 3에서 x를 X를 이용하여 신뢰받는기관과 information theoretic secure VSS[Pe2]를 이용해서 공유하게 되는데 g^x가 공개되지 않고 상호작용이 없다. 단계 4에서는 a가 작은 부분이라는 것을 증명하기 위해 각 신뢰받는기관과 $100/(n-t)$ 번의 KOD 프로토콜을 수행하는데 이때 보장되는 오차의 확률은 2^{-100}이다. 직렬로 하면 완벽한 영지식 증명이며 효율성을 위해 병렬로 하면 영지식 증명은 아니지만 VPKE 프로토콜에는 적당하다.

모든 확인을 하고 난 다음 신뢰받는기관은 각각 사용자의 공개키 P에 대하여 서명한다. 그러면 이 공개키는 DH 시스템을 이용한 공개키 암호 시스템에서 사용될 수 있다. 법원의 영장이 있어서 사용자의 비밀키 S를 복구해야 하는 경우 신뢰받는기관이 모여서 x를 복구한다. 이것은 x가 VSS를 통하여 공유되었기 때문에 가능하다. 그 다음에 $g^a=g^{S-x}=P/g^x$이므로 g^a를 얻는다. 그런데 a는 2^l비트이므로 Shank의 baby-step giant-step 방법을 이용하면 a는 2^l의 시간으로 얻을 수 있다.

마지막으로 이 프로토콜이 verifiable partial key escrow인 이유를 알아보자. 먼저 X에 대해 고려하면 X에 기초해서 x를 escrow할 때 VSS의 성질에 의해 신뢰받는기관은 나중에 x를 복구할 수 있음을 보장받는다. 이 x를 $D(X)$라 하자. 그 다음 $i=1,...,2l-1$에 대하여 A_i에 대해 생각해 보자. KOD프로토콜은 $D(A_i)$-A_i를 복구해서 얻을 수 있는 것-가 한 비트이지 긴 스트링이 아니라는 것을 증명한다. X와 A_i들이 실제로 commiital이라는 것은 사용자가 $\log_g(h)$를 모른다면 유일한 $D(X)$와 $D(A_i)$가 존재함을 의미한다. 그리고 신뢰받는기관은 비밀키가 $D(X)+D(A_0)2^0+D(A_1)2^1\cdots+D(A_{2l-1})2^{2l-1}$이라는 것을 알고 KOD에서 $D(A_i)$가 1 비트라는 것을 증명했으므로 a가 2l비트임을 확인할 수 있다. 그리고 early recovery를 할 수 없는 것은 $t(t+1$일 때 비밀키를 복구할 수 있다)개의 신뢰받는기관에게는 a가 될 수 있는 2^{2l}개의 값이 다 같은 것으로 보이기 때문이다.

〈RSA를 위한 VPKE 프로토콜〉

k, 모듈러스의 길이(최소한 512비트)

m, 안전도를 위한 변수로 $m=100n/(n-t)$라 한다.

k_2, partiality의 정도를 조절하는 안전도를 위한 변수

p, q, 사용자의 비밀키를 구성하는 소수

$N=pq$, k비트인 사용자의 모듈러스

E, 블록 암호화 함수로 키를 K로 하고 입력을 x라고 하여 $E_K(x)=C$를 출력하고 $E_K^{-1}(C)=x$이다.

$J_N(\cdot)$은 Jacobi symbol이고 $Z_N^{+1}\subseteq Z_N^*$, Jacobi symbol이 1인 원소들이라 하고 $Z_N^{-1}\subseteq Z_N^*$은 Jacobi symbol -1인 원소들이라 한다.

단계 1. 사용자는 신뢰받는기관에 표준적인 영지식 증명 방법을 이용하여 N이 두 소수의 곱임을 증명하고 신뢰받는기관은 N이 소수가 아님을 확인한다.

단계 2. 다음을 독립적으로 m번 반복한다.

단계 2.1 사용자는 $y\in_R Z_N^{-1}$을 해서 $s=y^2 \bmod N$이라 하고 s의 Jacobi symbol +1인 제곱근 $x\in Z_N^{+1}$을 계산한다. 임의의 k_2비트 K를 골라서 $C=E_K(x)$를 하고 s를 신뢰받는기관에 보낸다.

단계 2.2 사용자는 $\alpha_0=C$라 하고 $\alpha_1,\ldots,\alpha_t\in_R Z_N^*$을 한다. $f(x)=\alpha_0+\alpha_1 x+\cdots\alpha_t x_t\in Z_N^*[x]$라 하고 $j=1,\ldots,n$에 대해 신뢰받는기관 j에게 $f(j)$를 비밀채널을 통하여 보낸다.

단계 2.3 신뢰받는기관은 사용자에게 임의의 비트 c를 보낸다.

단계 2.4

c가 0이면 사용자는 y를 신뢰받는기관에 보내고 신뢰받는기관은 $J_N(y)=-1$이고 $y^2=s \bmod N$인지 확인한다.

c가 1이면 사용자는 $\alpha_0,\ldots,\alpha_t$와 K를 신뢰받는기관에 보낸다. 신뢰받는기관은 단계 2.2에서와 같은 $f(x)$를 만들고 신뢰받는기관 j는 실제로 $f(j)$를 받았는지 확인한다. 그리고 $|K|=k_2$인지 확인하고 단계 2.1에서 받은 C를 복호화해서 이것을 x라 하고 $J_N(x)=+1$, $x^2=s \bmod N$인지 확인한다.

앞의 프로토콜에서 $C=E_K(M)$에서 $T(k_2)$를 K를 모르는 상태에서 C가 주어졌을 때 M을 알아내는데 필요한 시간이라고 하면 이것이 2l이 되도록 k_2를 정한다. $T(k_2)$보다 작은 시간에서는 M에 대한 어떠한 부분적인 정보도 얻을 수 없어야 한다. 단계 1에서 N이 두 소수의 곱이라는 것을 증명하기 위한 방법은 Micali의 fair RSA와 같은 방법을 쓰면 된다. 즉 N이 두 소수의 곱이면 Z_N^*은 4개중 1개가 제곱수임을 이용한다. 예를 들면 사용자와 신뢰받는기관은 Z_N^*에서 약 500개의 난수를 고르고 사용자는 이중 100개의 제곱근을 신뢰받는기관에 제공해서 100개가 제곱수임을 보여준다. 단계 2에서 C를 VSS가 아닌 일반적인 secret sharing을 이용하여 escrow하며 information theoretic committals을 사용하지 않는다. 그리고 단계 2에서 반복하는 이유는 verifiabe하게 하기 위해서이다. 매번 신뢰받는기관은 y를 받아서 Jacobi symbol이 -1이고 s의 제곱근인지 확인하거나 f와 K를 가지고 escrowed된 C가 Jacobi symbol +1인 s의 제곱근의 암호문인지 확인한다. 이것으로 신뢰받는기관은 키를 그들이 원하는 적절한 시간을 들여서 복구할 수 있음을 확신한다. 단계 2에서의 반복 횟수는 신뢰받는기관마다 $100/(n-t)$번이 가장 간단한 방법이며 전체적으로 최소한 100번의 반복 횟수가 필요하다.

키를 복구하는 방법에 대해 알아보면 다음과 같다. 반복 횟수를 나타내는 인덱스를 $i(1\le i\le m)$라 하고 아래 첨자로 나타내고 $I=\{i\in[1,m] : c_i=0\}$이라 한다. 각각의 $i\in I$에 대해 키를 복구하는 절차를 **Recover**(i)라 하자. 신뢰받는기관은 f_i(신뢰받는기관 j는 매번 $f(j)$를 받게 되고 f의 위수는 t이므로 $t+1$개의 신뢰받는기관이 그들의 $f(j)$를 모아서 f의 계수를 알아낼 수 있음)를 만들어 내고 이것에서 암호문 C_i를 얻는다($f_i(0)=\alpha_0=C_i$). 그 다음 C_i에서 x_i를 얻기 위해 2^l 시간의 복구 절차를 거친다. 그리고 $x_i^2=s_i \bmod N$인지 확인한다. $c_i=0$이었으므로 신뢰받는기관은 이미 y_i를 갖고 있고 x_i와 y_i를 가지고 $\gcd(x_i-y_i,N)$을 구할 수 있고 이것이 N의 소인수(1이나 N이 아닌)이고 따라서 N을 인수 분해할 수 있다. 이 **Recover**(i)

는 모든 $i \in I$에 대해 항상 동작하는 것은 아니므로 임의로 i를 골라서 **Recover**(i)를 동작시킨다. 그러나 **Recover**(i)를 2번 동작시켜야 하는 확률은 0.02로 98%는 2^l 시간으로 N을 소인수 분해할 수 있다.

RSA를 위한 VPKE에서 partiality는 직접적으로 N을 소인수 분해하는 것과 관련이 있는 것이 아니라 관용키 블록 암호화 함수와 연관이 있다. N을 소인수 분해하는 것에서 partiality를 주기 위한 방법은 다음과 같을 수 있다. N, p, q, y와 x는 앞에서 정의된 것을 쓰고 $a \in \{1,2,\ldots,2^{k_2}\}$, $k_2 < k$을 임의로 고르고 $C = x/a \in Z_N^*$라 한다($a \notin Z_N^*$인 경우 N을 바로 소인수 분해할 수 있는데 이 경우를 무시함). N, y와 C가 주어졌을 때 N을 소인수 분해하기 위한 직접적인 방법은 x를 알아내는 것인데 이렇게 하기 위해 a의 값을 exhaustively 찾으면 2^{k_2} 시간이 걸리는데 물론 이것은 최선의 방법은 아니다. $T(k_2)$를 N, y와 C가 주어졌을 때 k을 소인수 분해하기 위한 시간이라고 할 때 $k_2 = 0$이면 $a = 1$, $C = x$, $\gcd(C-y, N)$을 구하는 시간만이 필요하며 $k_2 = k-1$이면 N을 소인수 분해하는 것과 같은 시간이 걸린다. 이것으로부터 $2^l \leq T(k-1)$임을 알 수 있고 $T(k_2) = 2^l$을 만족하는 어떤 적당한 k_2가 존재함을 알 수 있다.

partial key의 개념은 특정한 개인의 키를 복구하는데 많은 어려움을 겪게 하자는 것이 아니라 갑자기 한꺼번에 많은 사람의 비밀키를 복구하는데 어려움을 겪게 하는데 있다. 그러므로 특정한 개인에게는 일반적인 key escrow보다 복잡한 과정을 거쳐 비밀키를 신뢰받는기관에 맡겨도 더 나은 안전을 제공하지 않는다.

DH를 위한 VPKE 프로토콜에서 a가 작은 부분임을 증명하기 위해 atomic proof of knowledge of de-commitment를 각 신뢰받는 기관과 $100/(n-t)$의 상호작용을 직렬로 하거나 (이 경우 완전한 영지식 증명 방법이다) 병렬적인(영지식 증명 방법은 아니지만 이 프로토콜을 위해서는 충분하다) 상호작용을 해야 한다. 이 부분의 상호작용의 수가 줄어들거나 상호작용을 하지 않아야 할 것이다.

RSA를 위한 VPKE에서는 VSS를 쓰지 않고 보통의 secret sharing을 썼지만 N이 두 소수의 곱임을 증명하기 위한 과정(일반적인 영지식 프로토콜을 이용함)이 필요하며 verifiable하기 위한 challenge-response의 수가 각 신뢰받는기관과 $100/(n-t)$번을 반복해야 한다. 역시 상호작용의 수가 줄어들거나 없어져야 할 것이다. **Recover**(i)를 이용해서 N의 소인수를 찾을 때 y_i, $f(j)$, s_i를 신뢰받는기관이 저장하고 있어야 하는데 이들은 각각 512비트 정도이고 $c_i = 0$인 모든 i에 대하여 다 저장을 하자면 0.5 $\times m \times 1526$(m은 반복횟수이고 $c_i = 0$일 확률은 1/2이다)의 메모리가 필요해서 저장해야 될 양이 많다. 모든 i에 대하여 저장을 하지 않는다 하더라도 최소한 1526비트의 메모리가 더 필요하다. 마지막으로 이 VPKE에서 partiality는 관용키 블록 암호화 함수에 기초를 두고 있으며 인수분해에 기초한 VPKE는 k_2의 크기를 문제로 제시하고 있다.

4. 결 론

앞 장에서는 공개키 알고리즘을 이용한 key escrow system을 비교분석하였다. [Mi]는 공개키 알고리즘을 일반적으로 fair하게 만들 수 있는 방법을 설명하고 있는데 다른 알고리즘에 비해 능률적이지는 못하며 [KiLe]에서 fair하지 않음을 보였다. [KiLe]의 subliminal channel을 없애는 방법은 공개키를 사용하는 key escrow system에 일반적으로 적용될 수 있으나 사용자와 신뢰받는기관/키관리센타간의 통신횟수가 많다. [LWY]는 DH방식에 대

해서 기간에 대한 제한을 주는 방법을 제안했는데 일 방향 통신에 대한 고려가 부족하다. [BeGw]의 VPKE는 partial key escrow로서 다른 프로토콜에 비해 복잡한 프로토콜을 사용하지만 다수를 고려하기 때문에 특정한 개인이 특별히 더 자신의 사생활에 대한 보장을 받을 수 있는 것은 아니며 영지식증명을 위해서 사용자와 신뢰받는기관사이의 통신횟수가 많은편이다. [Mi2] 역시 partial key escrow인데 early recovery의 문제점이 있으며 split key escrow에 대한 고려가 부족하다.

여기에서 소개된 공개키를 이용한 key escrow system은 나름대로의 장점과 단점을 많이 가지고 있다. 이것에 기초해서 법집행기관의 합법적인 도청과 개인의 사생활을 보장할 수 있는 두 가지 목표를 보다 조화롭게 달성할 수 있는 key escrow system을 만드는 것이 앞으로의 연구목표이다.

참 고 문 헌

[한이] 한 상근, 이 영. 미국의 암호정책에 대한 연구-클리퍼 칩을 중심으로, 통신정보보호학회지 제4권 4호, 1994. 12.

[BeGo] M. Bellare and S. Goldwasser. Verifiable partial key escrow. technical report number CS95-447, Department of Computer Science and Engineering, University of California at San Diego, October 1995.

[BELW] D.M.Balenson, C.M.Elison, S.B.Lipner and S.T.Walker, A new Approach to Software Key Escrow Encryption, Trustec Information Systems, Inc, Aug, 1994

[Bl] G. Blakey. Safeguarding cryptographic keys. AFIPS Conference Proceedings, June 1979.

[BrDe] E.F.Brickel, D.E.Denning, S.T.Kent, D.P.Maher and W.Tuchman, SKIPJACK Review, Interim report, Jul.1993

[CGMA] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch. Verfiable Secret sharing and Achieving Simultaneity in the Presence of Faults. Proc. 26 ann. IEEE Symp. on Foundations of Computer Science, IEE, New York, 1986, pp. 383-395

[DiHe] W. Diffie and M. Hellman, New Directions in Cryptography, IEEE Trans. on Information Theory 22 (6), 1976, pp. 644-654

[EES] National institute of Standard and Technology, Federal Information Processing Standard Publication 185, Escrowed Encryption Standard, February 9 1994, Washington, DC.

[KiLe] J. Kilian and T. Leighton. Fair cryptosystems revisited. Advances in Cryptology-Crypto '95, Lecture Notes in Computer Science(LNCS) Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995

[LWY] A. Lenstra, P. Winkler and Y. Yacobi. A key escrow system with warrant bounds. Advances in Cryptology-Crypto '95, LNCS Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995

[Mi] S. Micali. Fair Cryptosystems. MIT/LCS TR-579.b, November 1993

[Na] M. Naor. Bit commitment using pseudorandomness. Journal of Cryptology (1991) 4. pp. 151-158

[Pe1] T. Pederson. Non-interactive and information theoretic secure verifiable sharing. Advances in Cryptology-Crypto '91, LNCS Vol. 547, D. Davies ed., Springer-Verlag, 1991

[Pe2] T. Pedersen. Distributed provers with application to undeniable signatures, Advances in Cryptology-Eurocrypt'91, LNCS Vol. 547, pp 221-238, Springer-Verlag, 1991

[RSA] R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. Comm. ACM 21, February 1978.

[Si] G. J. Simmons. The prisoners' problem and subliminal channel. Advances in Cryptology-Proceedings of Crypto 83, D. Chaum ed., 1984

[Sh1] A. Shamir. How to share a secret. CACM, Vol. 22, No. 11, 1979

[Sh2] A. Shamir. Private communication made at Crypto 95. August 1995

□ 著者紹介

정 경 임(학생회원)

1972년생
1995년 2월 포항공과대학교 전자전기공학과 학사
1995년 ~ 현재 포항공과대학교 전자전기공학과 석사과정 재학중

이 필 중(李 弼 中) 종신회원

1951년 12월 30일생
1974년 2월 서울대학교 전자공학과 학사
1977년 2월 서울대학교 전자공학과 석사
1982년 6월 U.C.L.A. System Science, Engineer
1985년 6월 U.C.L.A. Electrical Engineering, Ph.D.
1980년 6월 ~ 1985년 8월 Jet Propulsion Laboratory, Senior Engineer
1985년 8월 ~ 1990년 2월 Bell Communications Research, M.T.S.
1990년 2월 ~ 현재 포항공과대학 전자전기공학과, 부교수