최상위 인증기관을 위한 비밀 공유 방식

윤호선*, 윤이중**, 염흥열*
순천향대학교 전기전자공학부*
한국전자통신연구원**

A Secret Sharing Scheme for Root CA

Ho-Sun Yoon*, Ee-Joong Yoon**, Heung-Youl Youm*
Dept. of Electrical and Electronic Eng. Soonhunhyang Univ.*
Electronics & Telecommunication Research Institute**

요 약

전자 상거래를 비롯한 많은 정보보호 응용분야가 공개키 암호 시스템을 기반으로 설계되고 있으며, 이러한 현상 때문에 PKI의 중요성이 날로 증가하고 있다. 또한 PKI에서 믿음의 근간이 되는 것은 루트 CA이며, 루트 CA의 안전성을 증가시키기 위한 연구가 지속되고 있다. 루트 CA의 안전성을 향상시키는 가장 좋은 접근 방법은 비밀 공유 방식을 이용하는 것이다. 본 논문에서는 기존에 발표된 RSA 키 생성 프로토콜과 서명문 생성 프로토콜을 분석하고 새로운 서명문 생성 프로토콜을 제안한다. 또한 실제 루트 CA에 이러한 결과들을 적용하는 방안을 제시한다.

제1장 서론

현재까지 비밀 공유 방식에 대한 많은 연구 결과들이 발표되었다. 특히 DSS의 서명문 공유 방식에 대한 연구 결과는 많다[5,6,7]. 또한 KCDSA에 비밀 공유 방식을 적용한 연구 결과도 발표되었다[8]. 하지만 DSS나 KCDSA에 적용되는 여러 비밀 공유 기법들을 그대로 루트 CA의 부분 서명문을 생성하는데 적용할 수는 없다. 물론 DSS나 KCDSA 비밀 공유 방식을 이용해서 루트 CA의 부분 서명

문을 생성할 수 있다. 하지만 일반적으로 공개키 암호 시스템에서 전자 서명은 RSA 서명 방식을 이용하므로, RSA의 키 및 서명문을 부분적으로 계산하는 프로토콜이 필요하다. 몇몇 논문에서 RSA의 비밀 공유 방식을 언급한 것들이 있지만, 아직까지는 많은 보완이 필요하며 더욱 많은 연구가 필요하다[9,10].

본 논문에서는 기존에 발표된 RSA 키 생성 프로토콜과 서명문 생성 프로토콜을 분석하고, 새로운 서명문 생성 프로토콜을 제안한다. 이렇게 제안된 RSA 키 생성 방식과 서명문 생성 방식은 실제 루트 CA에 적용할 수 있다.

본 논문은 2장에서 DSS, KCDSA뿐만 아니라 RSA의 비밀 공유 방식에도 적용할 수 있는 여러 가지 비밀 공유 기법들을 간단히 서술하고, 3장에서 RSA 키를 분배하는 방법과 서명문을 분배하는 방법을 기술한다. 4장에서는 분산적으로 RSA 부분 서명문을 생성하고 그중 일정한 수의 정족수가 모이면 서명문을 생성하는 방법을 제안하며, 제안된 방법은 실제 루트 CA에서 개인키를 분배하고 분배된 개인키를 이용해서 부분 서명문을 생성할 수 있다. 마지막으로 5장에서 결론을 맺고 추후에 진행될 연구 방향을 기술한다.

제2장 비밀 공유 방식

비밀 공유 방식의 개념은 Shamir가 처음으로 제안했다[1]. 그 이후에 검증 가능한 방식과 딜러가 필요없는 방식들이 제안되었으며, 계산적인 측면에서 좀더 효율적인 방식들이 제안되고 있다. 이 장에서는 RSA 키 생성 및 서명문 생성을 위해서 필요한 비밀 공유 기법들에 대해서 언급한다.

2.1 Shamir의 비밀 공유 방식

Shamir는 임계치 기법의 구성을 위해 유한체의 다항식을 사용하였다. (k, n) 임계치 비밀 공유 기법은 n 참여자들 각각에게 비밀에 대한 몫을 딜러가 분배하고 n 참여자들 중에서 k 참여자들 이상이 모이면 비밀 값을 계산할 수 있는 프로토콜이다. 이러한 비밀 공유 기법은 k보다 작은 참여자들의 집합이 비밀에 관한 어떠한 정보도 얻을 수 없음을 의미한다.

2.2 멱승 보간법

$(a_1, \ldots, a_n) \xleftarrow{(t,n)} a \bmod q$인 a_i가 각 참여자들에게 분배되고 각 참여자들은 $g^{a_i} \bmod p$로부터 $g^a \bmod p$를 계산하려고 한다면, 우리는 멱승 보간법을 이용할 수 있다. 멱승 보간법을 기술하기 위해서 Lagrange 보간 다항식을 먼저 서술한다 [5].

n 지점($y_1 = f(x_1)$, $y_2 = f(x_2), \ldots, y_n = f(x_n)$)을 지나는 n-1 차의 보간 다항식이 식 2-1과 같이 주어졌다고 하자.

$$P(x) = \sum_{j=1}^{n} P_j(x) \qquad (\text{여기서, } P_j(x) = \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k} y_j) \qquad (2\text{-}1)$$

식 2-1은 식 2-2로 표현된다.

$$P(x) = \sum_{j=1}^{n} y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k} \qquad (2\text{-}2)$$

식 2-2로부터 식 2-3을 구할 수 있다.

$$P(0) = \sum_{j=1}^{n} y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x_k}{x_k - x_j} \qquad (2\text{-}3)$$

다항식 $P(x)$에서 상수항이 비밀 값이 되므로 $P(0)$이 비밀 값이다. 다음은 멱승 보간법에 대한 내용이다.

i) t 이상의 참여자들은 g^{a_i}를 공개한다.
ii) 각 참여자들은 식 2-1을 계산한다.

$$\prod_{i=1}^{t} (g^{a_i})^{c_i} = g^a \qquad (\text{여기서, } c_i = \prod_{1 \leq j \leq t, j \neq i} \frac{x_j}{x_j - x_i}) \qquad (2\text{-}4)$$

식 2-4의 증명은 식 2-5와 같다.

$$\prod_{i=1}^{t} (g^{a_i})^{c_i} = (g^{a_1})^{c_1} \cdot (g^{a_2})^{c_2} \cdot \cdots \cdot (g^{a_t})^{c_t}$$

$$= (g^{a_1})^{\frac{x_2}{x_2-x_1}\cdot\frac{x_3}{x_3-x_1}\cdots\frac{x_t}{x_t-x_1}} \cdot \cdots \cdot (g^{a_t})^{\frac{x_1}{x_1-x_t}\cdots\frac{x_{t-1}}{x_{t-1}-x_t}} = g^a \qquad (2\text{-}5)$$

2.3 Feldman의 검증 가능한 비밀 공유 방식

비밀 공유 프로토콜에 참여하는 각 참여자들은 자신이 수신한 정보가 정당한 정보인지를 확인할 필요가 있다. Feldman은 이산대수의 어려움에 근거한 검증 가능한 프로토콜을 발표했다[2].

딜러는 각 참여자들에게 $(\sigma_1, \ldots, \sigma_n) \xleftarrow{(t,n)} \sigma \bmod q$인 공유 σ_i를 분배한다. 만약 $f(x) = \sum_j a_j x^j$이면 딜러는 $a_j = g^{a_j} \bmod p$를 공개한다. 이것은 행위자들이 식 2-6에서와 같이 $g^{\sigma_i} = \prod a_j^{i^j}$을 검사함으로써 딜러가 각 참여자들에게 정당한 몫 σ_i를 분배했는지 검사한다.

$$\begin{aligned} g^{\sigma_i} &= a_0 \cdot a_1^{i^1} \cdot a_2^{i^2} \cdot a_3^{i^3} \cdots a_t^{i^t} \\ &= g^{a_0} \cdot g^{a_1 i} \cdot g^{a_2 i^2} \cdots g^{a_t i^t} \\ &= g^{a_0} \cdot g^{a_1 i} \cdot g^{a_2 i^2} \cdots g^{a_t i^t} \\ &= g^{\sigma_i} (\because \sigma_i = a_0 + a_1 i + a_2 i^2 \cdots + a_t i^t) \end{aligned} \qquad (2\text{-}6)$$

이러한 프로토콜은 계산적으로 안전하다. 즉 이산대수 문제를 풀 수 있다면 비밀 값을 얻을 수 있다.

2.4 Pedersen의 검증 가능한 비밀 공유 방식

Feldman의 VSS의 안전성은 이산 대수의 어려움에 근거하며, Pedersen의 VSS는 정보 이론적으로 안전한(Information-Theoretic Secure) VSS 이다[3,4]. 자세한 프로토콜은 다음과 같다.

1. 딜러는 s 위탁을 공개한다 : 임의적으로 선택된 t$\in Z_q$에 대해서 $E_0 = E(s,t)$ 이다. 여기서 $E(s,t) = g^s h^t$ 이다.

2. 딜러는 F(0)=s를 만족하는 최소한 k-1 차의 다항식 $F \in Z_q[x]$를 선택하며, $s_i = F(i)$ (for $i = 1, \ldots, n$)를계산한다. 여기서, $F(x) = s + F_1 x + \cdots + F_{k-1} x^{k-1}$ 이다. 또한 딜러는 임의적으로 $G_1, \cdots, G_{k-1} \in Z_q$를 선택하고 i=1, ⋯ , k-1 에 대해서 F_i에 위탁을 수행할 때 G_i를 사용한다. 즉, 딜러는 $E_i = E(F_i, G_i)$를 공개한다.
3. $G(x) = t + G_1 x + \cdots + G_{k-1} x^{k-1}$이라 하고 $t_i = G(i)$ $(i = 1, \ldots, n)$라 하자. 그리고 나서 딜러는 비밀스럽게 (s_i, t_i)를 P_i에게 전송한다. 여기서 i=1, 2, ... , n 이다.

여기서, g, h ∈ G_q 이다. G_q는 q차의 Z_q^*의 유일한 부분집합이고, g는 G_q의 생성자이다. 만약 요소 a∈ Z_q^*가 G_q에 있다면 $a \in G_q \Leftrightarrow a^g = 1$ 이다. Z_q는 필드이고, 딜러는 s∈ Z_q를 다음과 같이 분산한다.
P_i가 그의 공유 (s_i, t_i)를 수신하면, P_i는 식 2-7을 검증한다.

$$E(s_i, t_i) = \prod_{j=0}^{k-1} E_j^{\,i^j} \qquad (2\text{-}7)$$

이러한 비밀 공유 방식은 무조건적으로 비밀의 기밀성을 보호하지만, 공유의 정당성은 계산적인 가정에 의존한다.

2.5 연합 난수 비밀 공유 방식

지금까지 논의된 비밀 공유 방식은 신임받는 딜러가 존재해야만 한다. 하지만 이 절에서 논의하는 비밀 공유 방식은 참여자 각자가 딜러처럼 행동함으로써 딜러가 필요 없으며, 각 참여자는 임의의 난수를 공유하는 프로토콜이다. 즉, 자신의 비밀 공유 α_i를 선택하여, (t, n)-비밀 공유 기법을 이용하여 각 참여자에게 부분 비밀 α_{ij}을 분배한다. 또 각 참여자들은 수신된 부분 정보를 합하여 비밀 s에 대응되는 자신의 부분 비밀 공유 s_i를 계산한다. 즉, 각 참여자들은 $(s_1, \ldots, s_n) \xleftrightarrow{(t,n)} s \bmod q$ 가 되는 비밀 공유 s_i를 갖는다[5,6,7]. 연합 난수 비밀 공유 방식을 검증하기 위해서 Feldman의 비밀 공유 방식이나 Pedersen의 비밀

공유 방식을 적용할 수 있다.

연합 난수 비밀 공유 프로토콜과 유사한 방법을 이용함으로써 비밀 값이 영이 되도록 각 참여자들이 비밀 값을 공유할 수 있다. 연합 영 비밀 공유(Joint Zero Secret Sharing) 프로토콜은 비밀 값이 영인 총괄적인 공유를 생성한다[5,6]. 이 프로토콜은 연합 난수 비밀 공유 프로토콜과 비슷하지만 지역적인 비밀 대신 각 참여자들이 영 값의 공유를 분배한다. 영 값의 정당한 분배는 각 분배 다항식들의 상수항이 '0'이라는 것을 검사하는 것에 의해 수행된다. 앞에서 서술되었듯이 검증 가능한 비밀 공유 방식을 적용하면 검증 가능한 연합 영 비밀 공유 방식을 생성할 수 있다.

2.6 두 비밀들의 곱셈

이 절에서 다룰 프로토콜은, 참여자들 사이에 공유된 주어진 두 비밀 u와 v가 있을 때, 이 두 비밀을 밝히지 않고 두 비밀의 곱 uv를 계산하는 프로토콜이다.

각각 t 차의 다항식에 위해 공유된 주어진 u, v는 각 참여자들이 자신들의 u, v 공유를 지역적으로 곱한다. 결과는 2t 차 다항식의 uv 공유가 될 것이다. 따라서, 값 uv는 2t+1의 정당한 공유들의 집합으로부터 재구성된다. 이때 t차의 두 다항식을 곱하고 2t차의 다항식을 더한다. t차의 두 다항식은 연합 난수 비밀 공유 기법을 이용하고, 2t차의 다항식은 연합 영 비밀 공유 기법을 이용한다. 이렇게 함으로써 t차의 두 다항식을 곱한 2t차의 다항식을 랜덤하게 만들 수 있는데, 이것은 연합 영 비밀 공유 기법을 이용한 2t차의 다항식으로 더함으로써 가능해 진다. 연합 영 비밀 공유 방식의 비밀 값은 영이기 때문에, 비밀 값은 그대로 유지된다[5,6].

두 비밀 값의 곱에 대한 몫을 분배하는 또 다른 방법이 있다[7]. 이 프로토콜은 정방행렬을 이용함으로써 곱에 대한 몫이 t차의 다항식으로 분배되도록 하였다. 이러한 프로토콜의 장점은 연합 영 비밀 공유 방식을 이용하지 않고도 다항식을 랜덤하게 생성할 수 있다는 점이다.

제3장 RSA 키 및 서명문 공유 방식

RSA의 키를 분배하고 분배된 키로부터 부분 서명문을 생성하며 일정한 정족수

가 합의하면 전체 서명문을 생성하는 프로토콜에 대한 연구가 활발히 진행되고 있다[9,10]. 이 장에서는 작은 공개키 e에 대해서 RSA 키를 분배하고 분배된 키로부터 부분 서명문을 생성하는 방법을 기술한다.

3.1 RSA 키 생성

이 절에서는 RSA의 키를 공유하는 방법을 설명한다. RSA 키를 공유하기 위해서는 다음과 같은 네 단계의 과정을 수행한다[9].

- $p_1+\cdots+p_n$와 $q_1+\cdots+q_n$에 대해서 작은 소수로 나누어 떨어지는가를 확인하는 trial division을 수행한다.
- 분산적으로 $N=pq$을 계산한다.
- N이 두 소수들의 곱인지를 검사하기 위해서 분산적으로 소수성을 검사한다.
- 분산적으로 개인키를 계산한다.

이러한 네 단계를 수행하면 각 참여자들은 개인키에 대한 몫을 생성할 수 있다. 이렇게 생성된 부분 개인키는 부분 서명문을 생성하는데 이용될 수 있다. 설명의 간략화를 위해서 3-out-of-4에 대해서 기술한다.

3.1.1 Trial Division

Trial division은 각 참여자들이 선택한 $p_1+\cdots+p_n$와 $q_1+\cdots+q_n$의 값이 작은 소수로 나눔으로써, 뒤에 수행할 소수성 검사에서 소수일 확률을 높이는 역할을 한다. 즉, 한번 간략하게 소수가 아닌 수를 걸러내고, 그 후에 소수성 검사를 수행하는 것이다. 물론 Dan Boneh 등이 제안한 Trial Division 방식이 있지만 그러한 방법은 한 참여자가 Trial Division 한 결과를 공표해야만 하는 문제와 계산적인 부담이 있다[9]. 이러한 문제를 해결하기 위해서 본 논문에서는 3.1.2절에서 제안된 분산적으로 N을 계산하는 방법을 Trial Division에 적용했다.

1. 각 참여자들은 $q_1,\ldots,q_4$와 $r_1,\ldots,r_4$을 선택한다.

2. $q_1+\cdots+q_4$, $r_1+\cdots+r_4$를 공유한다.
   - 참여자 1 : $q_{1,1}=a_1x_1+q_1$, $\cdots$, $q_{1,4}=a_1x_4+q_1$
     참여자 2 : $q_{2,1}=a_2x_1+q_2$, $\cdots$, $q_{2,4}=a_2x_4+q_2$
     참여자 3 : $q_{3,1}=a_3x_1+q_3$, $\cdots$, $q_{3,4}=a_3x_4+q_3$
     참여자 4 : $q_{4,1}=a_4x_1+q_4$, $\cdots$, $q_{4,4}=a_4x_4+q_4$
   - 참여자 i :

     $$s_{qi}(x)=\sum_{j=1}^{4}q_{j,i}=(a_1+\cdots+a_4)x_i+(q_1+\cdots+q_4)$$

   - 같은 방법으로 $r_1+\cdots+r_4$ 공유

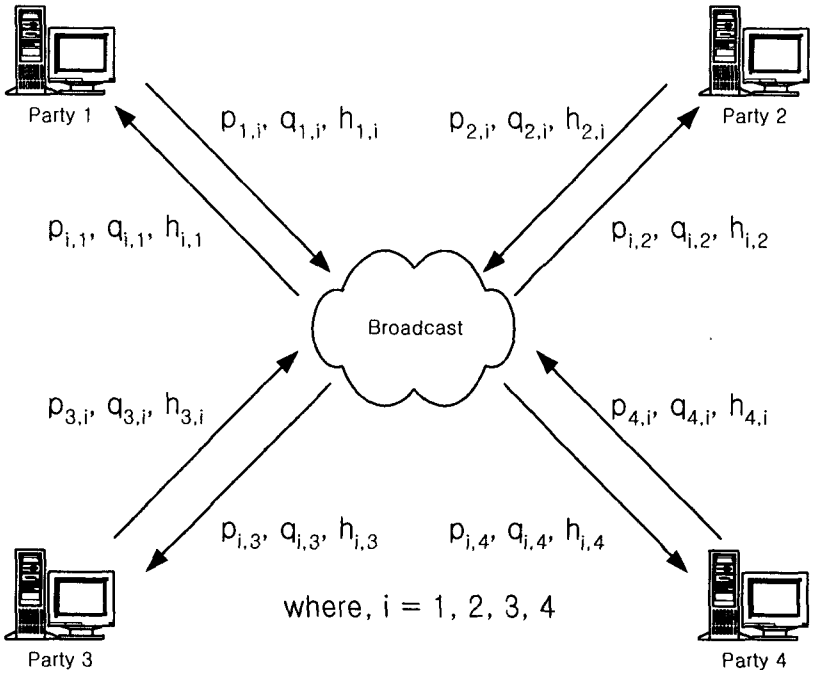
     $$s_{ri}(x)=\sum_{j=1}^{4}r_{j,i}=(b_1+\cdots+b_4)x_i+(r_1+\cdots+r_4)$$

3. 연합 영 비밀 공유 방식을 이용해서 2차의 임의의 다항식을 공유한다.
   - $h(x_i)=h_i=l_2\,x_i^2+l_1x_i+0$
4. $(q_1+\cdots+q_4)(r_1+\cdots+r_4)$ mod m_i를 공유한다.
   - $s_{qi}(x)\cdot s_{ri}(x)+h_i = k_2\,x_i^2+k_1x_i+(q_1+\cdots+q_4)(r_1+\cdots+r_4)$
5. 3 참여자 이상의 합의에 의해 $(q_1+\cdots+q_4)(r_1+\cdots+r_4)$ mod m_i 계산
6. 각 참여자들은 각각 소수성 검사
   - $qr\neq0$ 이면 소수 m_i로 나누어 떨어지지 않는다.

m_i는 작은 소수를 나타내며, 이러한 작은 소수들의 집합에 대해서 위의 과정을 반복적으로 수행한다. 이러한 프로토콜을 p에 대해서도 수행한다.

3.1.2 분산적인 N 계산

분산적으로 N을 계산하는 방법은 3.1.1절의 trial division에서 이용한 방법과 같다. 프로토콜의 마지막에 각 참여자들은 각자 N을 계산할 수 있다.



$$\alpha(x) = (\sum_j f_j(x))(\sum_j g_j(x)) + \sum_j h_j(x) \pmod P$$

$$\alpha(0) = N$$

그림 3.1 분산적인 N의 계산

각 참여자들은 3.1.1절과 같은 방법으로 다항식들을 선택하고, 참여자 i는 $p_{i,j}$와 $q_{i,j}$를 공개하고 참여자 j로부터 $p_{j,i}$와 $q_{j,i}$를 수신한다. 수신한 값들로부터 2차 다항식 $\alpha(x)$을 생성한다. 4명의 참여자들 중에 3명의 참여자들이 합의하면 N을 계산할 수 있다[10].

3.1.3 소수성 검사

분산적으로 생성된 N은 두 소수 p와 q의 곱으로 생성된 합성수이다. 각 참여자들은 N이 두 소수들의 곱으로 생성되었는지를 확인할 필요가 있다[9,10]. 실제적인 프로토콜은 다음과 같다.

1. 참여자들은 난수를 선택한다.

· $g \in Z_N^*$: 모든 참여자들에게 알려진 값

2. v_i 값 계산

· 참여자 1 : $v_1 = g^{N-p_1-q_1+1} \bmod N$

참여자 i : $v_i = g^{p_i+q_i} \bmod N$

3. 검증

· $v_1 = \prod_{i=2}^{4} v_i \pmod N$

만약 검사가 실패하면 N은 두 소수들의 곱으로 구성된 합성수가 아니다. 위의 프로토콜에 대한 정당성은 Dan Boneh 등이 발표한 논문에 제시되어있다[9].

3.1.4 개인키 생성

각 참여자가 분배된 비밀키를 생성하기 위해서 다음과 같은 프로토콜을 수행한다[10].

1. 분산적인 ϕ_i 계산

· $\phi_1 = N - p_1 - q_1 + 1$, $\phi_i = -p_i - q_i$ 여기서, $\phi(N) = \sum_{i=1} \phi_i$

2. $\phi^{-1} \bmod e$ 계산

· $r_{i,j}$를 각 참여자 j에게 전송. 여기서, $\phi_i = \sum_j r_{i,j} \bmod e$

· 참여자 i는 j로부터 $r_{j,i}$ 수신

· $\alpha_i = \sum_j r_{j,i} \bmod e$을 계산하고 모든 다른 서버들에게 전송

· 각 서버는 $l \bmod e$를 계산 : $l \bmod e = \sum \alpha_i$

· $\zeta = l^{-1} \bmod e = \phi^{-1} \bmod e$

3. 분산적인 개인키 생성

· $d_i = \lfloor \frac{-\zeta \cdot \phi_i}{e} \rfloor$

· 참여자 1은 $c^d \equiv c^r \cdot \prod c^{d_i} \bmod N$인 $r(0 \le r \le k)$을 선택해서 d_1에서 r을 뺀다. 3-out-of-4인 경우에 k는 4이다.

프로토콜의 마지막에 $\log_2 e + \log_2 k$ 비트들이 누설되지만, ϵ가 작기 때문에 공격자에게는 큰 도움이 되지 못한다.

3.1.5 부분 서명문 생성

3-out-of-4인 경우에 부분 서명문을 생성하기 위해서 3-out-of-3로 개인키를 분배하는 작업을 $\left[\begin{matrix}4\\3\end{matrix}\right](=4)$ 번을 수행해야만 한다. 표 3.1은 각 참여자들이 개인키를 분배한 결과를 나타낸다.

표 3.1 참여자들이 공유한 부분적인 개인키

참여자 / 개인키	참여자 1	참여자 2	참여자 3	참여자 4
d_1	$d_{1,1}$	$d_{2,1}$	$d_{3,1}$	
d_2	$d_{1,2}$	$d_{2,2}$		$d_{4,2}$
d_3	$d_{1,3}$		$d_{3,3}$	$d_{4,3}$
d_4		$d_{2,4}$	$d_{3,4}$	$d_{4,4}$

표 3.1에서 $d_1 = d_{1,1} + d_{2,1} + d_{3,1}$이며 d_2, d_3, d_4도 같다. 이러한 방법으로 개인키를 분배하면, 분배된 개인키를 이용해서 부분 서명문을 생성한다. 즉 참여자 1, 2, 3은 각각 부분 서명문 $m^{d_{1,1}}$, $m^{d_{2,1}}$, $m^{d_{3,1}}$을 생성해서 공개하고, 공개된 부분 서명문을 모두 곱하면 전체 서명문을 생성할 수 있다.

k-out-of-k로부터 t-out-of-k을 생성하는 방법이 제안되기도 했다[9].

제4장 루트 CA를 위한 비밀 공유 방식

이 장에서는 t-out-of-k로 RSA 부분 서명문을 생성하는 다른 방법을 제안한다. 이렇게 제안된 방식은 여러 루트 CA가 개인키를 분배하고 분배된 개인키를

이용해서 부분 서명문을 생성하며 일정한 정족수가 합의하면 전체 서명문을 생성할 수 있다. 이 장에서는 설명을 간략하게 하기 위해서 3-out-of-4에 대해서 기술한다. 3-out-of-4 방식은 t-out-of-k로 쉽게 확장할 수 있다.

4.1 RSA 서명문 생성(3-out-of-4)

이 절에서는 RSA 키를 t-out-of-k로 공유하고, 공유된 개인키로부터 부분 서명문을 생성하는 방식을 제안한다. 제안된 방식은 3장의 방식에 비해서 계산적인 측면에서 효율적이다.

4.1.1 RSA 키 생성(3-out-of-4)

3-out-of-4로 RSA 키를 생성하기 위해서는 먼저 4-out-of-4로 개인키를 분배하고, 분배된 개인키를 각각 3-out-of-4로 분배한다. 자세한 프로토콜은 다음과 같다.

1. 각 참여자들은 분산적인 ϕ_i 계산
   - $\phi_1 = N - p_1 - q_1 + 1,\ \phi_2 = -p_2 - q_2,\ \phi_3 = -p_3 - q_3,\ \phi_4 = -p_4 - q_4$
2. $\phi^{-1} \bmod e$ 계산
   - 참여자 i : $\phi_i = r_{i,1} + r_{i,2} + r_{i,3} + r_{i,4} \bmod e$ 인 $r_{i,j}$를 선택하고 각 참여자들 j에게 전송.
   - 참여자 i는 j로부터 $r_{j,i}$ 수신 : $r_{1,i}$, $r_{2,i}$, $r_{3,i}$, $r_{4,i}$
   - 참여자 i : $\alpha_i = r_{1,i} + r_{2,i} + r_{3,i} + r_{4,i} \bmod e$ 를 계산하고 각 참여자들 j에게 전송
   - 각 참여자들은 $l \bmod e$를 계산 : $l \bmod e = \sum \alpha_j$
   - $\zeta = l^{-1} \bmod e = \phi^{-1} \bmod e$
3. 분산적인 개인키 생성(4-out-of-4)
   - $d_i = \lfloor \frac{-\zeta \cdot \phi_i}{e} \rfloor$

참여자 1은 $c^d \equiv c^r \cdot \prod c^{d_i} \bmod N$인 $r(0 \le r \le 4)$을 선택해서 d_1에서 r을 뺀다.

4. 3-out-of-4 공유(연합 난수 비밀 공유 방식 이용)

참여자 1 : $d_{1,1} = a_1 x_1^2 + b_1 x_1 + d_1$, … , $d_{1,4} = a_1 x_4^2 + b_1 x_4 + d_1$

참여자 2 : $d_{2,1} = a_2 x_1^2 + b_2 x_1 + d_2$, … , $d_{2,4} = a_2 x_4^2 + b_2 x_4 + d_2$

참여자 3 : $d_{3,1} = a_3 x_1^2 + b_3 x_1 + d_3$, … , $d_{3,4} = a_3 x_4^2 + b_3 x_4 + d_3$

참여자 4 : $d_{4,1} = a_4 x_1^2 + b_4 x_1 + d_4$, … , $d_{4,4} = a_4 x_4^2 + b_4 x_4 + d_4$

참여자 1 : $s_1 = d_{1,1} + d_{2,1} + d_{3,1} + d_{4,1}$

$$= (a_1 + \cdots + a_4) x_1^2 + (b_1 + \cdots + b_4) x_1 + (d_1 + \cdots + d_4)$$

$$\vdots$$

참여자 4 : $s_4 = d_{1,4} + d_{2,4} + d_{3,4} + d_{4,4}$

$$= (a_1 + \cdots + a_4) x_4^2 + (b_1 + \cdots + b_4) x_4 + (d_1 + \cdots + d_4)$$

위의 프로토콜을 수행한 후에 각 참여자들은 개인키에 대한 몫 s_i를 유지한다. 세 참여자 이상이 합의하면 비밀키 $a\,(= d_1 + \cdots + d_4)$가 공개된다.

4.1.2 RSA 부분 서명문 생성(3-out-of-4)

3.1.5절에서 기술된 방식을 이용하면 3-out-of-4인 경우에 개인키를 생성하는 과정을 4번 수행해야만 한다. 이러한 문제를 해결하기 위해 3-out-of-4로 분배된 개인키를 이용해서 각 참여자가 부분 서명문을 생성하는 과정은 다음과 같다. 물론 부분 서명문을 생성하면서, 각 참여자들은 상대방의 부분적인 개인키나 전체 개인키에 대한 정보를 얻을 수 없다.

- 멱승 보간법 적용
  - 셋 이상의 참여자들이 메시지 m에 대한 서명문을 생성하려고 할 때
  - 셋 이상의 참여자들은 m^{s_i}를 공개
  - 공개된 m^{s_i}로부터 $m^s (= m^a)$을 계산

위의 방식은 개인키를 공개하지 않고 세 참여자 이상이 합의하면 전체 서명문을 생성하는 프로토콜이다. 각 참여자들은 부분 서명문 m^{s_i}를 공개하고 이렇게 공개된 부분 서명문으로부터 멱승 보간법을 이용하면 전체 서명문 m^a $(=m^{d_1+\cdots+d_t})$을 생성할 수 있다.

위의 프로토콜에서는 부분 서명문에 대한 정당성은 입증할 수는 없지만, 전체 서명문에 대한 정당성은 검증이 가능하다. 즉, 전체 서명문을 공개키로 지수승을 하면 쉽게 전체 서명문이 정당한지 여부를 확인할 수 있다.

제5장 결론

본 논문에서는 기존에 발표된 비밀 공유 방식들을 분석하고, RSA 키를 분배하는 방법과 부분 서명문을 생성하는 방식을 분석하였다. 이러한 분석된 결과를 이용해서 t-out-of-k RSA 부분 서명문을 생성하는 방안을 제안하였다. 기존의 방식을 이용하면 비밀키를 생성하는 과정을 $\begin{bmatrix} k \\ t \end{bmatrix}$ 번 수행해야만 한다. 본 논문에서는 t-out-of-k를 만족하는 부분 서명문을 생성하기 위해, k-out-of-k로 개인키를 분산하고, 분배된 각 개인키를 다시 t-out-of-k로 분산했다. 이렇게 분배된 개인키를 이용해서 t 참여자 이상이 전체 서명문을 생성하려고 할 때, 각 참여자들은 서명하려고 하는 메시지를 자신의 개인키에 대한 몫으로 지수승을 해서 공개한 후에, 멱승 보간법을 이용해서 전체 서명문을 생성하였다.

향후에는, 큰 공개키 e에 대한 개인키 분배 및 서명문 분배에 관한 연구가 진행되어야 하며, 검증 가능한 프로토콜에 대한 연구가 지속되어야만 할 것이다. 물론 큰 공개키 e에 관한 연구 결과가 발표되었지만, 여러 문제가 있다[9]. 또한 새롭게 제안되는 프로토콜을 실제로 구현하는 연구도 진행될 것이다.

- 참고문헌 -

[1] A. Shamir, "How to Share a Secret", Communications of the ACM, 1979

[2] P. Feldman, "A practical scheme for non-interactive verifiable secret

sharing", In Proc of the 28th IEEE Symposium on the Foundations of Computer Science, pp 427-437, 1987

[3] T.Pedersen, "Distributed provers with applications to undeniable signatures", Proc. EUROCRYPTO 91, pp. 221-242, 1991

[4] T.Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing", Proc. CRYPTO 91, pp 129-140, 1991

[5] 백종현, "매직 잉크 서명 기법을 이용한 전자현금 프로토콜 설계", 순천향대학교 석사 논문, 1997

[6] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Robust Threshold DSS signature", Advances in Cryptology Proc of Eurocrypt '96, pp. 354-371, 1996.

[7] Rosario Gennaro, Michael Rabin, Tal Rabin, "Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold cryptography", http://theory.lcs.mit.edu/~rosario/research.html, preprint, 1998

[8] 류영규, 윤호선, 류종호, 염흥열, "검증 가능한 비밀 공유 기법을 이용한 매직 잉크 서명 방식", CISC'98, 1998.12

[9] Dan Boneh, Matthew Franklin, "Efficient Generation of Shared RSA keys", Proc CRYPTO 97, pp. 425-439, 1997

[10] M. Malkin, T. Wu, D. Boneh, "Experimenting with Shared Generation of RSA keys", http://theory.stanford.edu/~dabo/pubs.html, preprint