효율적인 키이분배 및 암호 시스템의 제안 : 제 II 부 [1]

임 채 훈 . 이 필 중
포항공과대학 전자전기공학과

Proposal of efficient key distribution and encryption systems : Part II [1]

Chae Hoon Lim . Pil Joong Lee
Dept. E.E., Pohang Institute of Science and Technology

요약 본 논문의 제 II 부에서는 I 부에서 제안된 대칭형의 키이분배 시스템을 변형하여 일방향 통신만으로 키이분배를 가능하게 하는 일방향 키이분배 시스템을 제시한다. 또한 키이분배시에 직접적인 상호인증 기능을 제공할 수 있도록 제안된 시스템을 간단히 확장시킬 수 있으며 뿐만아니라 이를 응용하면 안전한 암호시스템을 구축할 수 있음도 보인다. 따라서 제안된 시스템은 키이분배와 동시에 암호화 기능까지 제공할 수 있으므로 특정한 암호시스템을 갖추지 않은 경우에도 매우 효율적으로 사용될 수 있을 것이다.

1. 서론

본 논문의 전반부인 제 I 부에서는 두 사용자의 전송정보 및 세션키이 계산이 완전 대칭인 대칭형의 키이분배 시스템으로 한번 혹은 두번의 모듈라 멱승 연산만으로 안전하게 세션키이를 공유할 수 있는 새로운 시스템들을 제안하였다. 제 II 부에서는 I 부에서 제안된 시스템을 변형하거나 이를 응용하여 여러가지 부가기능을 제공할 수 있게 하는 방안들을 제시하고자 한다.

[1] 본 논문의 일부는 한국 전자 통신 연구소에서 수탁한 연구의 결과임.

응용에 따라서는 양방향의 전송보다는 일방향의 전송만으로 세션키이를 공유할 수 있는 키이분배 시스템이 유용한 경우도 있을 것이므로 우선 2 장에서는 제 I 부에서 제안된 시스템을 변형하여 일방향 키이분배 시스템을 설계해 보기로 한다. 다음으로 3 장에서는 키이분배시에 통신하고 있는 당사자가 자신이 원하는 상대방인지를 직접 확인할 수 있도록 해 주는 상호인증 기능을 쉽게 제안 시스템에 결합시킬 수 있음을 보인다. 또한 4 장에서는 제안된 키이분배 시스템을 응용하여 키이분배와 동시에 암호화 기능까지 제공할 수 있는 안전한 암호화 알고리즘을 제시한다. 이로써 본 논문에서 제안된 시스템들을 통합 구현한다면 이용 가능한 관용 암호시스템을 갖추지 않은 경우에도 완전한 하나의 암호시스템을 구축할 수 있으므로 일석이조의 효과를 거둘 수 있을 것이다.

2. 일방향 키이분배 시스템

키이분배를 위해 일방향의 전송만을 이용하는 경우는 timestamp 나 동기화된 sequence number 등 해당 전송정보의 신구를 판별할 수 있는 메카니즘을 갖추지 않는 한 과거의 전송정보를 재전송함으로써 과거의 세션키이를 합법적인 사용자로 하여금 재계산하게 하는 replay attack 을 막는 것은 불가능하다. 따라서 여기서도 timestamp 를 결합한 키이분배 시스템을 고려하기로 한다 [1] [2].

가장 간단한 일방향 키이분배 시스템으로 관용 암호시스템을 이용하여 두 사용자간의 공유키이 M 으로 세션키이를 암호화하여 상대방에게 전송하는 방법을 생각할 수 있다. 즉 사용자 i는 두 랜덤수 R_1, R_2 를 발생시켜 시간/날자 등으로 구성된 timestamp T 와 함께 M을 키이로 하여 암호화된 $E_M(R_1, R_2, T)$ 를 사용자 j 에게 전송하면 사용자 j는 이를 복호화하여 얻은 timestamp T 를 현재의 시간과 비교하여 정해진 한계 (예를들면 1 분) 를 넘는지의 여부에 의해 재전송 공격 여부를 검출해 낼 수 있을 것이다. 이때 두 사용자의 세션키이는 $K = R_1 \oplus R_2$ 가 된다. 이와같이 두 랜덤수의 EXOR 로 세션키이를 계산하는 것은 세션키이가 노출된다 하더라도 R_1 이나 R_2 에 대해 아무런 정보도 누출되지 않으므로 전송에서 사용된 암호문으로부터 비밀정보 M 을 얻어려 하는 known-plaintext attack 의 가능성을 줄일 수 있다는 장점을 지니게 된다. 그러나 이와같이 관용 암호시스템을 이용하는 방법은 키이분배 시스템의 안전성이 사용되는 관용 암호시스템에 의존하게 되며, 더우기 두 사용자간의 고정된 비밀정보인 M이 시스템의 유일한 비밀정보임

을 상기하면 이는 결코 바람직한 방법은 아닐 것이다.

보다 일반적인 방법으로 다음의 일방향 키이분배 시스템을 살펴보자. 즉 사용자 i는 비밀 랜덤수 R_i, 통신하고자 하는 상대방 j와의 공유키이 $M \equiv_m g^{S_iS_j}$, 그리고 시간/날자 등으로 구성된 timestamp T를 이용하여 전송정보 $Z_i \equiv_m (M.T) \oplus P_i^{R_i}$를 계산하여 T와 함께 j에게 전송한다. 이를 받은 사용자 j는 평문으로 받은 T, 사용자 i와의 공유키이 M, 그리고 자신의 비밀키이 S_j를 이용하여 사용자 i와의 세션키이로 $K \equiv_m [(M.T) \oplus Z_i]^{S_j} \equiv_m g^{S_iS_jR_i}$를 계산한다. 이 시스템은 가능한 어떤 공격에 의해서도 세션키이를 불법 계산하는 것이 불가능하므로 안전한 것으로 볼 수 있으나 세션키이 공유를 위해 사용자 i는 3번, 사용자 j는 2번의 모듈라 멱승을 계산해야 하므로 효율성에 있어서는 기존의 방식들에 비해 별다른 잇점이 없다.

보다 효율적인 시스템으로 본 논문의 I부에서 제안된 양방향의 키이분배 시스템을 다음과 같이 일방향 키이분배 시스템으로 변형시켜 보자. 일방향 키이분배 시스템에서는 전송정보의 일부로 timestamp T를 전송해야 하므로 랜덤수 자체를 전송하는 방법은 적합치 않다. 따라서 I부에서 기술한 제안방식 1의 변형에서와 같이 전송정보로 $Z_i = M \oplus R_i$나 $Z_i \equiv_m M.R_i$를 이용하여 일방향 키이분배 시스템을 설계하기로 한다.

[제안방식 3]

❶ 사용자 i는 자신이 선택한 랜덤수 $R_i \in [0,m)$, 시간/날자 등으로 구성된 timestamp T, 그리고 사용자 j와의 공유키이 M을 이용하여 $Z_i \equiv_m (M.T) \oplus R_i$를 계산하고 전송정보로 이 Z_i와 T를 사용자 j에게 전송한다.

❷ 사용자 j는 T가 현재의 시간에 비해 정해진 한계를 넘는지의 여부를 조사하여 조건이 만족되면 이 T와 M을 이용하여 랜덤수 $R_i \equiv_m (M.T) \oplus Z_i$를 얻고 세션키이로 $K \equiv_m R^2$ 혹은 g^R, $R \equiv_m M.R_i$를 계산한다.

(물론 I부에서 기술한 제안방식 1의 각종 변형이나 제안방식 2를 이용하더라도 위와 유사하게 일방향 키이분배 시스템을 구성할 수 있을 것이다.)

이 시스템은 전송정보가 두 비밀수의 mod-2 addition이므로 COP attack 하에서는 zero-knowledge이며 또한 세션키이가 $M.R_i \pmod m$의 제곱이나 이것을 지수로 하는 형태이므로 세션키이가 알려진다 하더라도 이로부터 M이나 R_i를 유도해 내는 것은 불가능하다. 또한 세션키이 계산에 이용할 랜덤수를 회복하는 과정 ($R_i \equiv_m (M.T) \oplus Z_i$)에서 두

사용자간의 공유키이 M 이 관련되므로 이 M 을 모르는 한 어떤 impersonation 에 의해서도 합법적인 사용자와 세션키이를 공유할 수 있는 방법은 없다는 것을 알 수 있으며 더우기 timestamp 의 사용으로 재전송 공격 역시 무력해진다. 따라서 이 시스템은 어떤 공격에 의해서도 이를 깰 수 있는 방법이 없다는 점에서 안전하다고 할 수 있다.

이와같은 일방향의 키이분배 시스템은 secure E-mail 과 같이 대화형이 아닌 일방향의 비밀통신에서 특히 유용하게 사용될 수 있을 것이다. 물론 E-mail 에서는 송신자가 보낸 메세지가 수신자의 수중에 들어가기까지 걸리는 시간은 일정치 않으므로 키이분배 시스템에서 사용된 timestamp 만으로는 재전송 공격 여부를 판별하기는 어렵다. 즉 E-mail 에서는 전송정보가 mail 의 envelope field 에 실려 전송될 것이므로 위의 제안방식 3 의 과정 ❷ 에서와 같이 과거의 전송정보를 재전송한 것인지의 여부를 판별하는 기준으로 단순히 T 를 현재의 시간과 비교하는 것은 store-and-forward 방식의 mail 특성상 적절한 기준으로 삼기 어렵다. 그러나 mail 의 송신자는 그 mail 을 전송한 날자/시간을 mail 의 body 에 포함시키고 또한 수신자는 일정한 기간동안의 수신된 mail 을 저장해 두는 것이 일반적이므로 timestamp T 가 암호화된 mail 의 body 에 포함된 날자/시간과 일치하는지의 여부와 저장된 mail 중에 수신된 mail 과 동일한 날자/시간을 가진 mail 이 있는지의 여부를 점검함으로써 재전송 공격을 쉽게 판별할 수 있을 것이다. 한편 트래픽이 중앙집중형인 경우나 간단한 질의 - 응답 형태의 통신에서도 일방향의 키이분배 시스템이 양방향의 통신을 이용하는 대칭형의 키이분배 시스템보다 트래픽을 줄일 수 있으므로 효율적으로 사용될 수 있을 것이다.

3. 상호 인증기능의 제공

대부분의 키이분배 시스템들은 제 3 자가 합법적인 사용자와 세션키이를 공유하는 것이 불가능하도록 설계되지만 그렇다고 자신이 원하는 상대방과 세션키이를 공유했다는 것을 직접적으로 확인할 수 있는 방법 (direct authentication) 을 제공하는 것은 아니며 주로 키이 공유 후의 해당 세션키이를 이용한 handshake 과정에 의해 상대방을 확인하는 방법을 취한다. 그러나 불필요한 전송이나 시간낭비를 줄일 수 있도록 키이분배와 동시에 상호 인증기능을 제공할 수 있도록 하는 것이 보다 바람직할 것이다.

이 장에서는 상대방을 확인할 수 있는 수단으로 두 사용자 (i, j) 간의 공유키이 $M_{ij} \equiv_m g^{S_i S_j}$을 알고 있는지의 여부에 의해 이를 성취할 수 있도록 제안된 키이분배 프로토콜을 확장시켜 보기로 한다. 아울러 공유키이 M 을 계산할때 공개키이 디렉토리로부터 상대방의 공개키이를 정확히 입수하는 것은 무엇보다도 중요하므로 사용자와 공개키이 디렉토리 사이의 통신에서도 이를 적용시켜 보기로 한다[3]. 이를 위해 공개키이 디렉토리도 다른 모든 사용자와 마찬가지로 자신의 비밀키이 S_c에 대응하는 공개키이 $P_c \equiv_m g^{S_c}$를 모든 사용자에게 공개하는 것으로 가정한다. 또한 공개키이 디렉토리는 RSA 서명을 이용하여 각 사용자가 요구한 공개키이를 디지탈 서명하여 제공함으로써 이를 받은 사용자는 자신이 요구한 상대방의 공개키이가 맞는지의 여부를 확인할 수 있게 한다. 따라서 공개키이 디렉토리는 RSA 서명용 공개키이 n, e 를 역시 모든 사용자에게 공개하고 해당 비밀키이 d 는 안전한 장소에 보관한다 (n > m). 공개키이 e 는 사용자들의 계산량을 줄일 수 있도록 2 나 3 정도의 작은 수가 적당할 것이다. 아래의 프로토콜은 이 모든 과정을 순서대로 정리한 것이다. 사용자 i 가 사용자 j 에게 비밀통신을 요청하여 서로 연결된 상태라고 가정한다.

❶ 사용자 i 는 자신의 ID 인 ID_i, timestamp T_i, 그리고 공개키이 디렉토리와의 공유키이 $M_{ic} \equiv_m P_c^{S_i} \equiv_m g^{S_i S_c}$ 등을 이용하여 $V_{ic} = h(T_i, ID_i, M_{ic})$를 계산한 다음 사용자 j 의 공개키이 분배를 요구하는 메세지 $\{ID_i, ID_j, T_i, V_{ic}\}$를 공개키이 디렉토리로 전송한다. 여기서 함수 h(.) 는 충돌회피성 (collison-free property) 을 갖는 M_{ic} 에 대한 일방함수이며 전송량을 줄일 수 있도록 가능하면 작은 길이로 압축하는 공개된 hash 함수가 적당할 것이다. 그러나 별도의 hash 함수를 사용하는 것이 용이하지 않다면 비록 메세지 길이는 약간 늘어나겠지만 $h(T_i, ID_i, M_{ic}) \equiv_m (M_{ic} \oplus T_i - ID_i)^E$ 과 같은 형태의 함수도 이 용도로는 충분할 것이다. 여기에 timestamp 를 사용한 것은 과거 메세지의 재전송 공격을 막기 위한 것이다.

❷ 공개키이 디렉토리는 사용자 i 와의 공유키이 $M_{ci} = M_{ic} \equiv_m g^{S_c S_i}$를 계산하고 이 값과 사용자 i 로부터 받은 메세지로부터 $h(T_i, ID_i, M_{ic}) = \underline{V}_{ic}$를 계산하여 $V_{ic} = \underline{V}_{ic}$ 인지를 점검한다. 만일 이 조건이 만족된다면 다음 단계로 넘어가고 그렇지않은 경우는 이 요구가 거짓 요구임이 분명하므로 연결을 끊는다. 여기서 M_{ci} 는 각 사용자 i 의 공개키이 등록시에 한번만 계산하여 공개키이 디렉토리의 비밀키이로 암호화한 결과를 보통의 메모리에 저장한다면 공개키이 디렉토리의 계산량을 훨씬 덜어 줄 수 있을 것이다.

❸ 공개키이 디렉토리는 사용자 j 의 공개키이 P_j, timestamp T_c, 그리고 ID_j 를 이용하여

자신의 비밀키이로 서명한 공개키이 증명서 (public key certificate) $C_j \equiv_n (P_j \oplus T_c - ID_j)^d$ 를 계산한 다음 메세지 $\{ID_j, T_c, P_j, C_j\}$ 를 사용자 i 에게 전송한다. 여기서 timestamp 를 결합한 온-라인 서명 (on-line signature) 을 사용한 것은 사용자의 비밀키이 노출로 인해 새로운 비밀키이에 대한 공개키이를 재등록했을때 공격자가 과거의 노출된 비밀키이에 대응하는 공개키이 증명서를 재전송한다면 이후의 키이분배 프로토콜에서 공격자는 성공적으로 그 사용자를 가장할 수 있을 것이므로 이와같은 경우를 방지하기 위한 것이다. 비록 이산대수 문제를 풀 수없는 한 사용자의 공개키이로부터 그 비밀키이를 계산하는 것은 불가능하지만 사용자의 부주의 등으로 인한 비밀키이의 노출 가능성을 완전히 배제할 수는 없으므로 안전한 암호시스템의 운영을 위해서는 공개키이 디렉토리에서 이와같은 대응방안을 마련해야 하는 것은 당연한 귀결이라 할 것이다. 그러나 만일 사용자의 비밀키이 노출 가능성을 완전히 배제할 수 있다면 사용자의 등록시에 $C_j \equiv_n (P_j - ID_j)^d$ 의 형태로 공개키이 증명서를 생성하여 이를 공개키이 디렉토리에 등록하고 사용자의 요구시에는 이 C_j 만을 전송하는 것으로 충분할 것이다. 이는 Girault 등이 제안한 self-certified public key [4] 의 개념으로 C_j 자체에 사용자 j 의 ID 가 결합되어 이를 받은 사용자 i 는 자신이 원하는 상대방인 사용자 j 의 ID 로 키이분배에 사용될 P_j 를 계산할 것이므로 어떤 impersonation attack 도 불가능하게 될 것이기 때문이다.

❹ 사용자 i 는 공개키이 디렉토리의 공개키이 e 를 이용하여 $(C_j^e \pmod n) + ID_j) \oplus T_c = P_j$ 인지를 점검함으로써 이 P_j 가 유효한 사용자 j 의 공개키이인지를 확인할 수 있다. 이제 사용자 i 는 이 P_j 와 자신의 비밀키이를 이용하여 사용자 j 와의 공유키이 $M_{ij} \equiv_m g^{S_i S_j}$ 을 계산할 수 있다. 공개키이 증명서의 생성이나 인증과정에서 알 수 있듯이 공개키이 디렉토리의 서명용 공개키이 n 을 키이분배 시스템에서 사용되는 공통의 법 m 보다 큰 수로 선택하는 것이 바람직할 것이다.

사용자 j 역시 ❶ - ❹ 의 과정을 통하여 사용자 i 와의 공유키이 $M_{ji} = M_{ij}$ 를 계산한다.

❺ 사용자 i 는 랜덤수 R_i 를 선택하여 $V_i = h(ID_i, T_i, R_i, M_{ij})$ 를 계산한후 메세지 $\{ID_j, T_i, R_i, V_i\}$ 를 사용자 j 에게 전송한다. 여기서 V_i 는 사용자 j 에게 사용자 i 의 신원을 확인할 수 있도록 하는 인증표의 역활을 하는 것으로 hash 함수대신 $h(ID_i, T_i, R_i, M_{ij}) \equiv_m (M_{ij} \oplus R_i - ID_i \oplus T_i)^E$ 과 같은 간단한 연산을 이용할 수도 있을 것이다.

❻ 사용자 j 는 i 로부터 받은 메세지와 자신이 계산한 공유키이 $M_{ji} \equiv_m P_i^{S_j}$ 를 이용하여 $h(ID_i, T_i, R_i, M_{ji}) = V_i$ 를 얻어 V_i 와 비교함으로써 송신자가 진정한 사용자 i 인지의 여부를 판별할 수 있다. 이 확인 과정이 성공하면 사용자 j 역시 마찬가지로 메세지 $\{ID_i, T_j, R_j,$

V_j} 로 사용자 i 에게 응답하고 I 부에서 제안된 방식에 따라 사용자 i 와의 세션키이를 계산한다.

❼ 사용자 i 역시 마찬가지 방법으로 j 로부터 받은 메세지와 M_{ij}를 이용하여 송신자가 자신이 원하는 상대방인 사용자 j 인지를 확인하고 세션키이를 계산한다.

이상에서 기술한 바와 같이 두 사용자간의 공유키이 M 의 소유(계산가능) 여부를 이용하면 간단히 상호인증 기능을 제안된 키이분배 시스템에 결합시킬 수 있다. 인증 기능의 추가로 각 사용자들의 계산량이 증가하는 것은 거의 없으며 (단지 hash 값의 계산이나 몇번의 모듈라 곱셈이 추가되었을 뿐이다) 전송량에 있어서도 timestamp T 와 인증표 V 가 추가되었을 뿐이다.

한편 공개키이 디렉토리의 경우는 다수의 사용자들에 의한 동시 다발적인 공개키이분배 요구에 대해 매번 온-라인 서명을 생성해야 하므로 공개키이 디렉토리로의 통신량 폭주 뿐만아니라 과중한 부하로 인한 시간지연 등의 많은 문제점들을 안고있다. 통신량 집중현상은 공개키이 디렉토리를 이용하는 한 피하기 힘든 본질적인 문제로 공개키이 디렉토리를 계층 구조화시켜 망전체에 분산시키는 등의 방법으로 어느 정도 완화시킬 수 있을 것이다.

또한 공개키이 디렉토리의 온-라인 서명에 의한 과부하를 덜어 줄 수 있는 방안으로 온-라인 서명을 사용하는 대신에 사용자와 공개키이 디렉토리 사이의 공유키이 M_{ci}를 사용하여 재전송 공격이나 impersonation attack 을 검출하도록 할 수 있다. 즉 공개키이 분배시에 $\{ID_j, T_c, P_j, V_{cj}\}$, $V_{cj} = h(ID_j, T_c, P_j, M_{ci})$를 사용자 i 에게 전송하고 이를 받은 사용자 i 는 과정 ❷ 에서와 마찬가지로 자신이 계산한 V_{cj}와 공개키이 디렉토리로부터 받은 값을 비교하여 재전송 공격 여부를 판별할 수 있을 것이다. V_{cj}는 M_{ci}를 알고있는 사용자 i 와 공개키이 디렉토리만이 계산할 수 있고 이를 계산하는데 timestamp 가 관련되어 있으므로 어떤 impersonation 이나 재전송 공격도 불가능할 것이기 때문이다. 과정 ❷ 에서도 언급했듯이 공개키이 디렉토리와 사용자 i 사이의 공유키이인 M_{ci}는 등록시에 한번만 계산하여 KDC 를 이용한 키이분배 시스템에서 KDC 가 터미날키이를 관리하는 방법 [5] [6] 과 마찬가지로 이를 공개키이 디렉토리의 비밀키이로 암호화한 결과만을 보통의 메모리에 저장하고 필요시 다시 복호화하여 사용하는 방법을 택할 수 있으므로 위와 같은 공개키이 인증방법은 공개키이 디렉토리가 시간이 많이 걸리는 모듈라 멱승 연산을 전혀 수행하지 않고도 안전하게 공개키이 증명서를 발급할 수 있을 것이다.

4. 암호시스템의 구현

제 I 부에서 제안된 키이분배 시스템에 의해 계산된 세션키이를 이용하여 다음과 같이 block cipher system 을 구성할 수 있다. 두 사용자 i 와 j 는 서로 교환된 랜덤수들을 이용하면 키이블럭 K_t 를 연속적으로 계산할 수 있으므로 이들을 키이스트림 (keystream) 으로 이용하여 Vernam cipher 를 구성한 것이다. 평문은 공통의 법으로 사용된 m 과 같은 비트길이의 r 개의 평문블럭 P_t, t=1, 2, ... , r, 로 구성되고 해당 암호문 블럭은 C_t 로 표기한다.

$C_t = P_t \oplus K_t$, $t = 1, 2, \ldots, r$.

$K_t \equiv_m [(M \oplus R_i^t) \cdot (M \oplus R_j^t)]^E$ 혹은 $[(M \cdot R_i^t) \oplus (M \cdot R_j^t)]^E$

혹은 $(R_i^{t+1} \oplus R_j^{t+1} \oplus M)^E$ <-- 제안방식 1

$K_t \equiv_m (I_{t-11} \oplus I_{t-12})^E$, $I_{t1} \equiv_m (M \oplus I_{t-11})^E$, $I_{t2} \equiv_m (M \oplus I_{t-12})^E$,

$I_{01} \equiv_m (R_o + M_e)^E$, $I_{02} \equiv_m (R_e + M_o)^E$ <-- 제안방식 2

이 암호시스템에서 각 키이블럭들은 매 세션마다 교환되는 랜덤수들에 따라 거의 랜덤하게 변할 것이므로 ciphertext-only attack 하에서의 안전성은 Vernam cipher 와 마찬가지로 완전하게 보장된다고 할 수 있다. 그러나 known (chosen) plaintext attack 하에서는 모든 키이블럭들이 노출될 것이므로 그 안전성은 알려진 (같은 랜덤수들에 의해 발생된 충분히 많은) 키이블럭들로부터 공유키이 M 에 대한 정보를 얼마나 유출해 낼 수 있는지의 여부에 의존하게 된다. 비록 체계적인 증명은 어려우나 같은 랜덤수들의 멱승을 이용해 연속적으로 계산된 키이블럭의 수가 늘어난다 하더라도 mod-2 addition 과 modular multiplication 의 적절한 결합은 이들간의 상관관계를 파괴시켜 거의 무관하게 만들 것이므로 이들로부터 M 에 대한 정보를 추가로 얻는 것은 불가능함을 알 수 있다.

한편 암호문 (ciphertext) 에 대응되는 평문 (Plaintext) 이 노출된다 하더라도 이로부터 사용된 키이블럭들을 계산하는 것조차도 불가능하도록 암호화 과정을 다음과 같이 변형시킬 수 있다.

$C_t \equiv_m (P_t \cdot K_{t-1}) \oplus K_t$, $t = 1, 2, \ldots, r$.

$K_t \equiv_m [(M \oplus R_i^{t+1}) \cdot (M \oplus R_j^{t+1})]^E$, 혹은 $[(M \cdot R_i^{t+1}) \oplus (M \cdot R_j^{t+1})]^E$,

혹은 $(R_i^{t+2} \oplus R_j^{t+2} \oplus M)^E$ <-- 제안방식 1

$K_t \equiv_m (I_{t-11} \oplus I_{t-12})^E$, $I_{t1} \equiv_m (M \oplus I_{t-11})^E$, $I_{t2} \equiv_m (M \oplus I_{t-12})^E$,

$I_{01} \equiv_m (R_o + M_e)^E$, $I_{02} \equiv_m (R_e + M_o)^E$ <-- 제안방식 2

이와같이 구성하면 비록 평문이 노출된다고 하더라도 암호문은 두 비밀 랜덤수의 mod-2 addition 에 의해 계산되었으므로 이로부터 어느 키이블럭도 계산하는 것은 불가능하게 된다 (r 개의 암호문 블럭에 대해 r+1 개의 키이블럭들이 사용되었다). 따라서 known (chosen) plaintext attack 하에서도 안전성을 보장할 수 있는 안전한 암호시스템을 구축할 수가 있다. 그러나 위와같이 암호시스템을 구성한 경우, 암호화 과정에서는 앞의 경우보다 단지 한번의 modular multiplication 이 추가될 뿐이나 복호화 과정에서는 각 K_t 에 대한 역원을 계산해야 하므로 훨씬 더 많은 계산량이 요구된다는 결함이 있다.

복호화 과정에서 매 블럭마다 역원을 계산할 필요가 없도록 다음과 같이 평문과 곱해지는 키이블럭을 고정시킨 경우를 생각해 보자.

$C_t \equiv_m (P_t \cdot K_0) \oplus K_t$, $t = 1, 2, \ldots, r$.

$K_0 \equiv_m [(R_o + M_o)^E \oplus (R_e + M_o)^E]^E$, <-- 제안방식 2

$K_t \equiv_m [(M \oplus R_i^t) \cdot (M \oplus R_j^t)]^E$, 혹은 $[(M \cdot R_i^t) \oplus (M \cdot R_j^t)]^E$,

혹은 $(R_i^{t+1} \oplus R_j^{t+1} \oplus M)^E$ <-- 제안방식 1

이 경우 역시 r 개의 암호문을 생성하는데 r+1 개의 키이블럭이 사용되었으며 K_0 가 고정되더라도 $P_t \cdot K_0 \pmod m$ 은 P_t 에 따라 랜덤하게 변하는 비밀수가 될 것이고 그 결과가 또다른 랜덤 키이블럭 K_t 와 mod-2 addition 으로 결합되어 암호문을 생성하므로 앞의 경우와 마찬가지로 키이블럭의 노출을 막을 수 있다. 더우기 수신자의 경우 초기에 K_0 의 역원을 한번만 계산하면 되므로 앞의 경우보다 훨씬 효율적이라 할 수 있다. 이 암호시스템은 E=2 혹은 3 인 경우 한 블럭당 4 - 6 번 정도의 modular multiplication 연산이면 암/복호화를 할 수 있게 된다. 암호문의 생성을 $C_t \equiv_m (P_t \oplus K_0) + K_t$ 와 같이 구성하여도 안전성에는 변함이 없으며 한번의 modular multiplication 을 줄일 수 있을 것이다.

5. 결론

본 논문의 제 2 부에서는 I 부에서 제안된 대칭형의 키이분배 시스템을 변형하여 일방

향의 통신만으로도 키이분배를 가능하게 하는 일방향 키이분배 프로토콜들을 제시하였고 또한 키이분배시에 상대방을 직접 확인할 수 있는 상호인증 기능을 간단히 제안 시스템에 결합시킬 수 있음을 보였다. 뿐만아니라 제안된 키이분배 프로토콜들에 의해 계산된 세션키이를 변형하여 연쇄적으로 키이블럭들을 생성함으로써 Vernam cipher 형태의 암호시스템을 구성할 수 있음도 보였다. 제안된 암호시스템은 약 5번 정도의 modular multiplication 연산으로 메세지의 한 블럭을 암호화할 수 있으며 known (chosen) plaintext attack 하에서도 그 안전성을 보장할 수가 있다. 따라서 본 논문에서 제안된 시스템들을 통합 구현한다면 특정한 암호시스템을 갖추지 않은 경우에도 키이분배와 동시에 암호화 기능까지 제공할 수 있으므로 매우 효율적인 암호시스템을 구축할 수 있을 것이다.

[참고문헌]

[1] R.M.Needham and M.D.Schroeder, "Using encryption for authentication in large networks of computers," Comm. ACM, Vol.21, No.12, 1978, pp.993-999.

[2] D.E.Denning and G.M.Sacco, "Timestamps in key distribution protocols," Comm. ACM, Vol.24, No.8, 1981, pp.553-536.

[3] D.E.Denning, "Protecting public keys and signature keys," IEEE Computer, 16, 2, 1983, pp.27-35.

[4] M.Girault, "Self-certified public keys," Proc. Eurocrypt,91, pp.236-241.

[5] W.F.Ehrsam, S.M.Matyas, C.H.Meyer, and W.L.Tuchman, "A cryptographic key management scheme for implementing the Data Encryption Standards," IBM Systems J., 17, No.2, 1978, pp.106-125.

[6] C.H.Meyer and S.M.Matyas, "Cryptography : A new dimension in computer data security," John Wiley and Son, N.Y., 1982.