DIT(Digital Investment Trust) 시스템의 보안 설계

정은희*, 이병관**
*관동대학교 전자계산공학과
**관동대학교 컴퓨터공학과
e-mail:jeongnala@hanmail.net
bklee@mail.kwandong.ac.kr

A Security of DIT(Digital Investment Trust) system

Eun-Hee Jeong*, Byung-Kwan Lee**
*Dept of Computer Science, Kwandong University
**Dept of Computer Engineering, Kwandong University

요약

본 논문에서 제안하는 DIT(Digital Investment Trust) 시스템의 보안설계는 계좌 생성 및 계좌 이체, 자산관리에 관련된 정보를 제 3자로부터 보호하기 위해 ADES(Advanced DES)를 이용하여 고객의 정보를 암호화시켰으며, 전자서명을 위해 ECC, S_SHA 알고리즘을 사용하였다. 특히 전자상거래 결재 상에서 사용하는 전자 화폐는 복사본 생성이 용이하기 때문에 악의의 사용자가 불법 복제하여 전자화폐를 반복적으로 사용할 수 있으므로, 이중사용을 방지하기 위한 사전 검출 방법인 Schnorr 알고리즘을 사용하였다.

1. 서론

전자상거래를 통한 상품 구매시, 대금 결제는 신용카드를 기반으로 하는 방식이 보편화되고 있으나, 인터넷을 통한 신용카드 결제시 고객의 신용카드 정보가 보호되지 못한다는 문제점을 갖고 있다. 이러한 문제점을 해결하기 위한 방법으로 인터넷상의 거래에서 가장 관심을 끄는 지불 방법이 전자화폐이다. 즉, 전자 화폐는 현재 인터넷상의 거래에 가장 많이 이용되고 있는 신용카드 거래의 문제점을 해결하고 구매자를 보호하기 위해 제시된 지불 방법이다[1][2][3].

인터넷이라는 가상 공간에서 전자상거래를 수행할 때, 상대방의 신원을 확인하는 방법과 거래 내역에 대한 쌍방이 부인 방지 대책 등은 중요한 문제이며, 이러한 문제점은 암호 및 인증 기술을 이용함으로써 해결할 수 있다[4].

본 논문에서 제안하는 DIT 시스템 보안 설계는 디지털 투자 신탁 시스템으로 사용자가 전자 화폐를 이용해, 소액지불뿐만 아니라, 계좌 생성 및 계좌 이체를 할 수 있으며, 투자 개념을 도입한 인터넷 기반 은행업무 프로젝트로서 사용자의 정보를 암호화 알고리즘인 ADES, ECC, S_SHA를 이용하여 보호할 수 있는 프로토콜을 설계하였다. 또한 전자상거래 결재 상에서 사용하는 전자 화폐는 복사본 생성이 용이하기 때문에 악의의 사용자가 불법 복제하여 전자화폐를 반복적으로 사용할 수 있으므로, 이중사용을 방지하기 위한 사전 검출 방법인 Schnorr 알고리즘을 사용하였다.

본 논문의 구성은 제 2장에서 DIT 시스템을 설명하였고 제 3장에서 DIT 시스템의 보안 설계 대해서 설명하였고, 제 4장에선 DIT 시스템 보안 설계의 시뮬레이션 결과에 대해 설명하였고, 마지막으로 제 5장에선 결론 및 향후 연구 방향을 제시하였다.

2. DIT(Digital Investment Trust) 시스템

DIT 시스템은 크게 사용자 측면의 정보를 관리하는 클라이언트 모듈, DIT 시스템의 운영을 관리하는 서버 모듈, 사용자의 정보를 관리하는 데이터베이스 모듈로 나눌 수 있다.

2.1 클라이언트 모듈

클라이언트 모듈은 DIT 시스템에 접근하려는 일반적인 웹 브라우저와 관련된 작업을 하며, 고객번호를 생성한 후 저장하며, 고객번호는 서버 모듈에 전송된다.

```
procedure Client()
begin
  generate private_key(x);
  y = g^x mod p;  y1 = SHA_1(y);
  send_server(y,y1,C);
  if (y==s1) then
   begin
     save(y, C, s);
   end;
end;
```

(그림 2-1) 클라이언트 모듈 알고리즘

2.2 서버 모듈

서버 모듈은 DIT 시스템 운영을 제어하기 위한 웹 서버 작업을 수행하며, 익명 계좌에 대해 예금 혹은 인출을 관리하며, DIT 계좌간의 계좌 이체를 관리한다. 또한 사용자가 자산 관리를 관리하며, 전자 화폐의 이중 사용을 감시한다.

서버 모듈의 알고리즘은 다음과 같다.

```
procedure server_module()
begin
  if(S_SHA(y) == y1) then
  begin
      s1=sign_ECC(y1); send_user(y,s1);
  end;
  if(equal_ECC(c, s2))then
    begin
      serach_db(y);    replace_db(y, C-T);
      c1 = concatenate(y, C-T);
      c2 = S_SHA(c1);
      s3 = sign_ECC(c2); send_user(c2, s3);
      save(c2, y, C-T);
    end;
end;
```

(그림 2-2) 서버 모듈 알고리즘

2.3 데이터베이스 모듈

암호화 알고리즘인 ADES를 이용해 사용자 계좌 정보를 암호화하여 저장하며, ADES에 이용된 대칭 암호키 s를 타원곡선 암호화 기법인 ECC를 이용해 암호화하여 저장한다.

데이터베이스 모듈의 계좌 정보 관리 알고리즘은 다음과 같다.

```
procedure db_module()
begin
  s = generate_ADES_key();
  c = concatenate(C, y, nonce);
  v = ADES(c); s1 = ECC(s);
  save(v, s1);
end;
```

(그림 2-3) 계좌 정보 관리 알고리즘

데이터베이스 모듈의 고객정보테이블은 다음과 같다.

구분	index_no	Id	pw	name	register_no	phone	addr
값	int	text	text	text	text	text	text

(그림 2-4) 고객 정보 테이블

데이터베이스 모듈의 계좌 정보 저장 테이블은 다음과 같다.

구분	index_no	ADES_value	ECC_value
값	int	text	text

(그림 2-5) 계좌 정보 테이블

3. DIT 시스템 보안 설계

3.1 ADES(Advanced DES)

ADES는 기존의 암호화 알고리즘인 DES를 변형시킨 것으로, 암호화 하고자하는 메시지와 암호화시키는 키의 값을 각각 결합하는 과정에서 기존의 DES는 하나의 고정된 PE(Permutation) 박스를 기본 16라운드 순환할 때마다 같은 PE 박스를 사용하였는데, ADES는 각 라운드마다 PE 박스를 각각 다르게 설정함으로서 도청자에 의해 하나의 PE 박스를 도청 당하더라도 전체의 암호문 혹은 복호문을 알 수 없기 때문에 보다 강력한 암호화 알고리즘이다. 또한 기존의 DES와 본 논문에서 제안한 ADES의 실행 시간을 비교해보면, PE 박스를 각 라운드마다 다르게 설정하더라도 차이가 없다.

ADES 알고리즘 그림 3-1과 같다.

```
Procedure Ades()
begin
      permute message using ip_box;
      permute key using pc1_box;
      split c0 and d0 from key;
      for i=0 to 15 do
      begin
         left_rotation_shift(c, d, c', d', shift[i]);
         merge(c[i], d[i], k[i], pc2_box);
      end;
      for i=0 to 15 do
      begin
         create pe_box = random(1, 32, seed);
         expand_r e_r into r using pe_box;
         pre_s = e_r ⊕ k;
         reduce pre_s to post_s using s_box;
         permute f using p;
         L[i+1] = R[i];  R[i+1]= L[i] ⊕ f;
      end;
end;
```

(그림 3-1) ADES 알고리즘

3.2 S_SHA(Shift Secure Hash Algorithm)

S_SHA는 임의의 길이의 메시지를 고정된 길이의 출력값으로 압축시키는 함수로서 데이터의 무결성 검증, 메시지 인증에 사용한다.

S_SHA는 입력 메시지를 512 비트 블록으로 처리하고, 출력은 160 비트로 생성한다. S_SHA는 임의의 길이의 메시지가 입력되면, 이 메시지를 512 비트의 배수로 만든 후, 512 비트 블록으로 나눈다. 각각의 512 비트의 메시지 블록은 32 비트 워드 $W_0, W_1, \ldots, W_{15}$로 나뉘어진 후 각각의 워드를 왼쪽 순환 쉬프트를 한 후 80단계를 거쳐 메시지를 다이제스트한 후에 연쇄변수 H_0, H_1, H_2, H_3, H_4 비트열로 변환한 것이 해쉬값이 된다.

S_SHA는 메시지를 다이제스트하기 전에 메시지를 1bit 씩 왼쪽으로 순환쉬프트 시킨 후에, 메시지를 다이제스트 함으로써 역변환 공격에 대해 좀 더 보안을 강화시켰다. 또한 라운드 t=16 to 79 일 때, 32 비트 워드를 생성하는 수식(1)을 수식(2)으로 변경시킴으로써 메시지 다이제스트 시간을 단축시켰다.

$$W_i = S^1(W_{i-3}\ XOR\ W_{i-8}\ XOR\ W_{i-14}\ XOR\ W_{i-16}) \quad \ldots\ldots (1)$$

$$W_i = W_{i-3}\ XOR\ W_{i-8}\ XOR\ W_{i-14}\ XOR\ W_{i-16} \quad \ldots\ldots (2)$$

```
procedure S_SHA()
begin
  for j=1 to Block_size do
     for i=0 to 79 do
     begin
       select W[i], K and F;
       temp = _lrotl((A), (5))+F+E+W+K;
       E=D; D=C; C=lrotr((B), (2)); B=A;  A=temp;
     end;
    H0=H0+A; H1=H1+B; H2=H2+C; H3=H3+D; H4=H4+E;
  end;
end;
```

(그림 3-2) S_SHA 알고리즘

3.3 ECC

본 절에서는 ECC 암호 알고리즘을 이용해 사용자 A, B가 메시지를 암호화하여 보내는 알고리즘을 설명한다. 메시지는 소수 t보다 작은 정수이고, 랜덤한 정수 즉, 비밀키 k로 메시지를 덧셈 연산하여 사용자 B에게 전송하고 사용자 B는 자신의 비밀키 r로 메시지를 복호화한다.

ECC 알고리즘은 다음과 같다.

```
procedure ECC()
begin
choose random number a,b and prime number t;
  double_root(a, b, t);
  search_point();
  choose random number p_s, s_s and initial_point x, y;
  choose message point m_x and m_y;
  secret_add_point(a, b, t, x, y, p_s, public_p);
  secret_add_point(a, b, t, x, y, s_s, secret_p);
  x1=public_p[0]; y1=public_p[1];
  secret_add_point(a, b, t, x1, y1, s_s, points)
  x2=points[0] ; y2=points[1];
  add_point(a, b, t, m_x, m_y, x2, y2, secret_p);
  x1=secret_p[0]; y1=secret_p[1];  y2=t+(-1)*y2;
  add_point(a, b, x1, y1, x2, y2, m_p);
end;
```

(그림 3-3) ECC 암·복호화 알고리즘

3.4 Double Spending 방지

사용자의 익명성을 허용하는 전자화폐는 그 자체가 가치(Value)를 가진 디지털 정보이다. 디지털 정보는 종이 문서와는 달리 복사본의 생성이 용이하기 때문에 악의(惡意)의 사용자가 전자화폐를 불법 복제하여 반복적으로 사용할 수 있다.

본 논문에서는 이중 사용 방지를 위해 Schnorr authentication Protocol을 사용한다. Schnorr authentication Protocol은 이산 대수에 기반을 둔 시스템으로 사용자의 비밀키를 공개하지 않으면서 사용자를 인증하는 프로토콜이다. Schnorr authentication Protocol의 알고리즘은 다음과 같다.

```
Procedure Schnorr_public_key()
begin
   select prime number p;
   select prime field q;
   g^q = 1 mod p;
   select prime number t;
   select private key x;
   h = g^x mod p;
end;
```

(그림 3-4) Schnorr 공개키 계산 알고리즘

```
Procedure Schnorr_authentication()
begin
   select random number w;
   a = g^w mod p;
   select challenge number c;
   y = w + x*c mod q;
   if (g^y == a*h^c mod p) then
      begin
         authentication.;
      end;
end;
```

(그림 3-5) Schnorr_authentication 알고리즘

4. 시뮬레이션

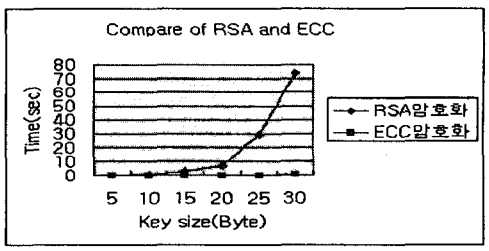
DIT 시스템 보안 설계에 대한 시뮬레이션은 O/S는 Winows XP, CPU는 PentiumIV 1.50GHz, RAM은 512MB의 환경에서 실행되었다.

4.1 RSA와 ECC

DIT 시스템의 보안 설계에 공개키 암호화 알고리즘으로 ECC를 사용하였으며, 현재 많이 사용되고 있는 RSA 비교하였다.

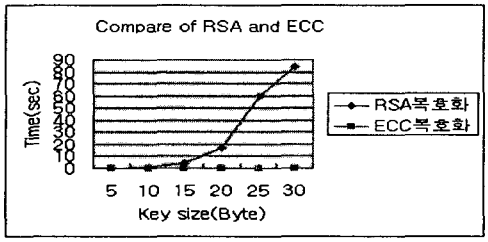
RSA 암호방식은 합성수의 소인수 분해 계산의 어려움을 이용하여 백 자리 이상의 두 개의 소수 p, q를 선택하여 공개키와 비밀키를 구성한 후 평문을 암, 복호화 하는 형식을 갖지만, ECC 암호화 방식은 이산대수에서 사용하는 유한체의 덧셈군을 타원 곡선군으로 대치한 암호시스템이다.

그림 4-1은 RSA와 ECC의 암호화 수행 시간 비교한 것으로 키 크기가 5byte일 때에는 6:1, 10일 때에는 18.3:1, 15일 때에는 44.1:1, 20일 때에는 23.9:1, 25일 때에는 76.6:1, 30일 때에는 136.9:1로 키 크기가 커질수록 ECC가 상당히 빠른 비율을 보였다.



(그림 4-1) RSA와 ECC의 암호화 시간 비교

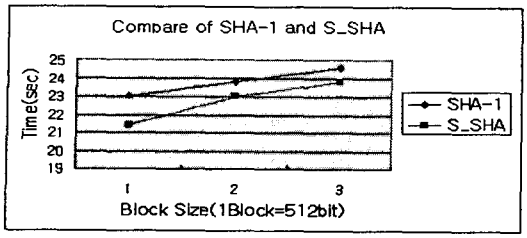
그림 4-2는 RSA와 ECC의 복호화 수행 시간을 비교한 것으로 키 크기가 5일 때에는 1.5:1, 10일 때에는 6.3:1, 15일 때에는 23.3:1, 20일 때에는 47.5:1, 25일 때에는 166.3:1, 30일 때에는 215.6:1의 비율을 나타냈다.



(그림 4-2) RSA와 ECC의 복호화 시간 비교

4.2 SHA-1와 S_SHA

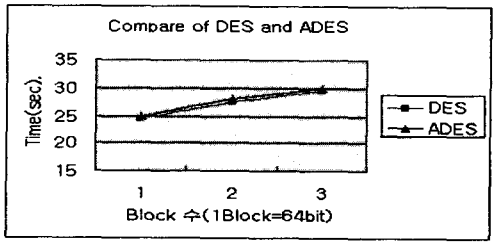
DIT 시스템의 보안 설계에서 메시지 다이제스트를 생성하는 S_SHA와 기존의 SHA_1의 메시지 다이제스트 시간을 비교하였다. 그림 4-3은 기존의 SHA-1과 S_SHA의 메시지 다이제스트 시간을 비교한 것으로 S_SHA의 메시지 다이제스트 수행 시간이 SHA-1보다 평균 1.06초가 빠르다.



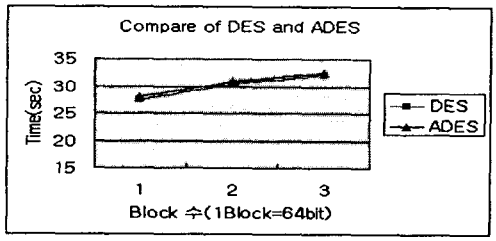
(그림 4-3) SHA-1과 S_SHA의 메시지 다이제스트 수행시간 비교

4.3 DES와 ADES

DIT 시스템의 보안 설계에서 메시지를 암호화시키는 ADES와 기존의 DES의 암·복호화 시간을 비교하였다.



(그림 4-4) DES와 ADES의 암호화



(그림 4-5) DES와 ADES의 복호화

그림 4-4와 그림 4-5는 DES와 ADES의 메시지 암·복호화 시간을 메시지 길이 따라 비교한 것이다. 기존의 DES와 본 논문에서 제안한 ADES의 암·복호화 시간은 ADES가 빠르다고는 할 수 없다. 하지만, PE 박스를 다르게 함으로써 키 값이 노출되더라도 메시지를 보호할 수 있다.

5. 결론

본 논문에서 제시한 DIT 시스템 익명 계좌 생성 및 계좌 이체와 투자 개념이 도입된 인터넷 기반 은행 업무 시스템이다. 그러므로 DIT 시스템의 사용자가 정보 누설 걱정 없이 자유롭게 사용할 수 있도록 DIT 시스템의 보안 설계가 필요하다.

DIT 시스템 보안 설계에선 공개키 보안을 강화한 ECC와 기존의 비밀키에 암호 기능을 강화시킨 ADES(Advanced DES)를 사용하였으며, 메시지 무결성과 부인 방지를 위해 S_SHA를 사용하였고, 전자화폐의 이중 사용을 방지하기 위해 Schnorr 알고리즘을 사용하였다. 이 경우에 기존의 공개키 암호화 알고리즘인 RSA와 대칭키 암호화 알고리즘인 DES, 메시지 무결성과 부인방지에 사용되는 SHA_1을 사용하였을 때보다 암호화 시간과 복호화 시간이 현저히 감소되는 알 수 있었다.

DIT 시스템 보안설계에선, 클라이언트 모듈과 서버 모듈 사이에서 데이터를 전송할 때, 공개키 암호화 방식을 사용하였으며, 특히, DIT 시스템 데이터베이스에 사용자 정보를 저장할 때 비밀키 암호화 방식을 보안한 ADES를 사용하였고, 대칭키를 다시 한번 ECC로 암호화시킴으로써 고객 정보 보안을 강화시킴으로서 제 3자로부터 고객 정보를 보호하였다. 또한, DIT 에이전트는 계좌 생성 및 계좌 이체를 익명으로 처리하므로, 고객의 정보와 자산을 보호할 수 있는 차세대 금융 관리 시스템이라고 볼 수 있다.

향후 과제로는 본 논문에서 제안한 DIT 에이전트에 자산 관리 기능을 추가시키는 것이다.

참 고 문 헌

[1] 김기병, 김수홍, "전자상거래를 위한 지불방법", 정보처리학회지, 제6권 제1호, pp35-14, 1999.
[2] 이은철, "뉴밀레니엄 시대의 전자화폐(상)", 지식재산21, 통권 제59호, 2000년 3월
[3] 이은철, "뉴밀레니엄 시대의 전자화폐(하)", 지식재산21, 통권 제60호, 2000년 5월
[4] 신홍식, "전자상거래 안전성과 인증서비스", 정보처리학회지 제7권 제2호, pp107-114, 2000.
[5] 이만영 외 5인 공저, "전자상거래 보안기술", 생능출판사
[6] 이병관, "전자상거래 보안", 남두도서
[7] 양승해, 이병관 "ECSET 설계를 위한 타원곡선 알고리즘" 정보처리학회 추계학술발표논문집, 제8권 2호, pp843-846, 2001.
[8] 정은희, 이병관 "DIT 시스템 설계를 위한 계좌관리" 정보처리학회 춘계학술발표논문집, 제9권 1호, pp951-954, 2002
[9] Dabid Pointchebal and Jacques Stern, "Security Proofs for Signature Schemes" Advances in Cryptology-Proceedings of EUROCRYPT'96, pp 387-398
[10] 배움닷컴, http://www.baeoom.com
[11] "PKCS#1 : RSA Encryption Standard"
[12] SSL 3.0 Specification, http://www.netscape.com/eng/ssl3/draft302.txt
[13] SSL 3.0 Implementation Assistance, http://www.netscape.com/eng/ssl3/traces/index.html
[14] http://polo.jmi.co.kr/html/00/000202_052_7.htm