논문 2006-43IE-2-4

통신시스템을 위한 암호 알고리즘에 관한 연구

(A study on the cipher algorithm for the communication system)

안 인 수*

(In-Soo Ahn)

요 약

본 논문은 음성 신호의 암·복호화를 위해 국내 표준으로 지향하는 SEED 암호 알고리즘의 입력 데이터를 192비트 키 입력을 256비트로 확장하고, 16 라운드 함수 수행으로 암호화 강도를 향상시킨 개선된 알고리즘을 제안하였다. 또한, 제안한 알고리즘을 적용한 암·복호화 칩을 VHDL로 설계하고 C 컴파일러와 Foundation Express Tool을 이용한 시뮬레이션으로 입력 데이터의 암·복호화 결과를 생성하였으며, 하드웨어 자원 활용도과 속도 효율이 개선됨을 확인하였다.

Abstract

In this paper we proposed of the SEED cipher algorithm which improved cipher intensity. The proposed algorithm has input data of 192bit and key input data of 256bit and it performs 16 Rounds for improvement of cipher intensity. We simulated the algorithm employing C compiler and the Foundation Express Tool so that verified performance of it.

Keywords : 블록암호알고리즘, SEED 알고리즘, VHDL, 통신

Ⅰ. 서 론

통신의 급속한 발전과 통신망의 공유로 정보가 국가 경제 발전을 좌우하는 중요한 요소로 작용하는 가운데 컴퓨터와 통신 시스템을 기반으로 한 정보처리 과정에서 보안의 필요성을 인식하게 되었다. 1970년대 미국에서는 민간 분야에서 사용될 암호시스템의 표준으로 DES(Data Encryption Standard)[1~2]를 개발하여 사용하여 왔고, 국내에서는 1999년에 한국정보보호진흥원이 128비트 블록암호알고리즘 SEED를 개발하여 국가표준(KICS)으로 제정 추진하였다[3]. 본 연구는 통신시스템에 적용할 수 있는 알고리즘을 제안하고 시뮬레이션과 모듈 설계를 통해 그 성능을 확인하였다.

Ⅱ. 192비트 입력 데이터를 갖는 확장된 알고리즘 제안

본 논문은 국내 표준 블록암호알고리즘인 SEED 알고리즘을 데이터와 키 입력을 확장하여 192비트 입력 데이터와 256비트 키 입력 데이터로 16 라운드를 수행하는 개선된 알고리즘을 제안하였으며, Foundation Express Tool을 이용한 시뮬레이션[4~5]을 통해 데이터의 암·복호화 결과를 확인하였다.

키 입력 256비트를 128비트씩 각각 분리하여 키 생성 과정을 수행한 후, 각 단계에서 생성된 값을 서로 XOR 하여 16 라운드의 최종 킷값을 생성한다. 생성된 키는 192비트 입력 데이터와 함께 라운드부의 입력값으로 전달되어 단일 라운드 방식의 라운드 함수부를 수행한다.

그림 1은 제안한 알고리즘의 전체 구조를 나타낸 것으로 192비트의 입력 데이터를 64비트씩 3 부분의 L_0 (191 downto 128), M_0 (127 downto 64), R_0 (63 downto 0)로 분리하여 수행한다.

* 정회원, 경인여자대학 컴퓨터정보학부
(School of Computer Information, Kyungin College)
※ 본 연구는 2004년도 경인여자대학 교내연구지원 연구비에 의해 수행되었음.
접수일자: 2006년2월3일, 수정완료일: 2006년6월10일

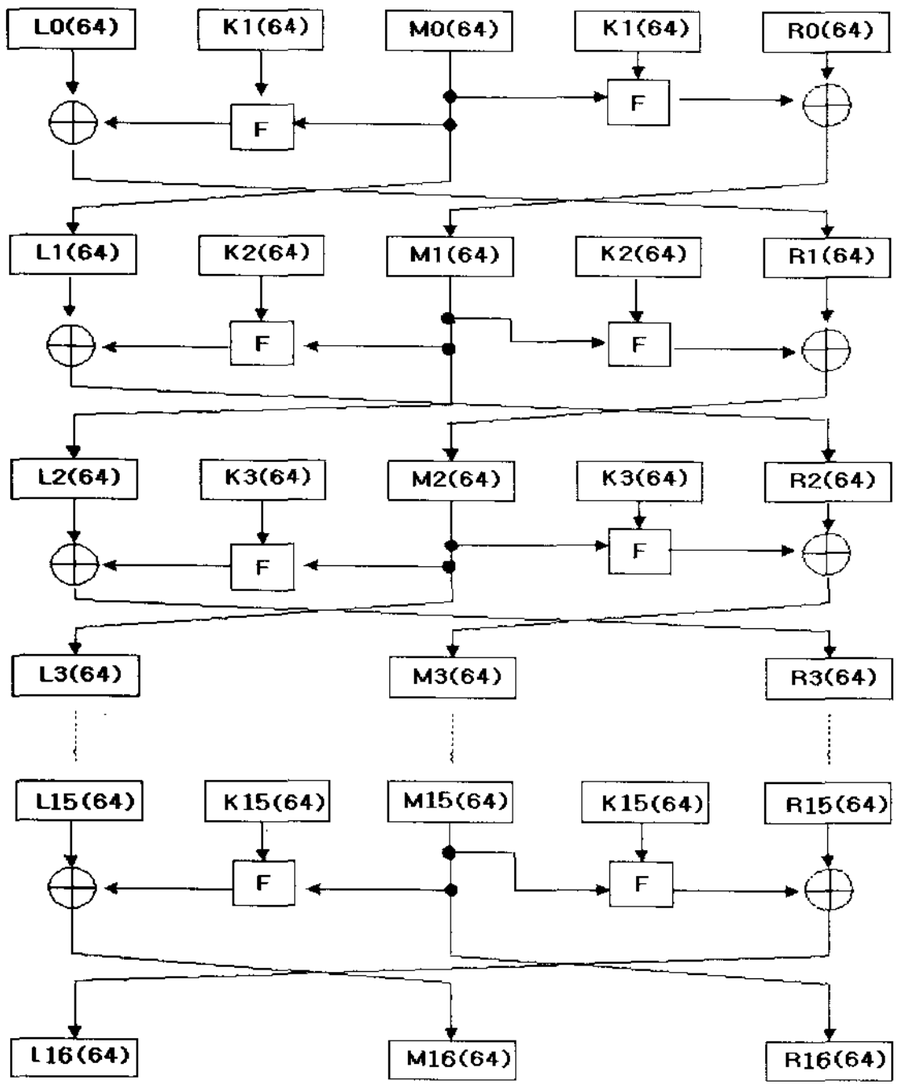


그림 1. 제안한 알고리즘의 전체 구조
Fig. 1. The global architecture of the proposed algorithm.

그림 1의 과정을 정리하면 식 (1)과 같다. 이 식에서 L_0, M_0, R_0는 192비트 입력 데이터를 각 64비트씩 세 부분으로 나눈 것이며, 각 라운드에서의 연산과 16 라운드의 함수 수행을 거쳐 최종 암호화된 데이터를 출력한다. 복호화는 암호화된 데이터를 입력으로 하여 역으로 수행한다.

$$
\begin{aligned}
&L_0,\quad R_0,\quad M_0 \quad L_1 = M_0,\quad M_1 = R_0 \oplus F(M_0, K_1),\quad R_1 = L_0 \oplus F(M_0, K_1)\\
&L_2 = M_1,\quad M_2 = R_1 \oplus F(M_1, K_2),\quad R_2 = L_1 \oplus F(M_1, K_2)\\
&L_3 = M_2,\quad M_3 = R_2 \oplus F(M_2, K_3),\quad R_3 = L_2 \oplus F(M_2, K_3)\\
&L_4 = M_3,\quad M_4 = R_3 \oplus F(M_3, K_4),\quad R_4 = L_3 \oplus F(M_3, K_4)\\
&L_5 = M_4,\quad M_5 = R_4 \oplus F(M_4, K_5),\quad R_5 = L_4 \oplus F(M_4, K_5)\\
&L_6 = M_5,\quad M_6 = R_5 \oplus F(M_5, K_6),\quad R_6 = L_5 \oplus F(M_5, K_6)\\
&L_7 = M_6,\quad M_7 = R_6 \oplus F(M_6, K_7),\quad R_7 = L_6 \oplus F(M_6, K_7)\\
&\qquad\vdots\\
&L_{14} = M_{13},\ M_{14} = R_{13} \oplus F(M_{13}, K_{14}),\ R_{14} = L_{13} \oplus F(M_{13}, K_{14})\\
&L_{15} = M_{14},\ M_{15} = R_{14} \oplus F(M_{14}, K_{15}),\ R_{15} = L_{14} \oplus F(M_{14}, K_{15})\\
&L_{16} = R_{15} \oplus F(M_{15}, K_{16}),\ M_{16} = L_{15} \oplus F(M_{15}, K_{16}),\ R_{16} = M_{15}
\end{aligned}
\tag{1}
$$

식 (1)에서 라운드 수를 i라 하면, 15 라운드까지는 식 (2)가 적용되며, 최종 16 라운드에서는 15 라운드의 값들을 교차하여 식 (3)과 같이 나타낼 수 있다.

$$
\begin{aligned}
&L_i = M_{i-1},\quad M_i = R_{i-1} \bigoplus F(M_{i-1}, K_i),\\
&R_i = L_{i-1} \bigoplus F(M_{i-1}, K_i) \qquad (1 \leq i \leq 15)
\end{aligned}
\tag{2}
$$

$$
\begin{aligned}
&L_{16} = R_{15} \bigoplus F(M_{15}, K_{16}),\\
&M_{16} = L_{15} \bigoplus F(M_{15}, K_{16}),\ R_{16} = M_{15} \quad (i = 16)
\end{aligned}
\tag{3}
$$

암호 키 입력 데이터는 256비트로 이것을 각각 64비트씩 네 부분으로 나누어 교대로 8비트씩 좌우로 회전 이동한 후, 결과의 4워드(word)들에 대한 산술 연산과 G 함수를 적용하여 회전키를 생성한다.

라운드 키는 256비트 키를 입력으로 하여 키 생성 알고리즘에 의해 상위 128비트인 L 블록에서 생성된 16개의 암호 킷값과 하위 128비트의 R 블록에서 생성된 16개의 암호 킷값을 서로 XOR 하여 생성된다.

2.1 키 생성부

256(255 downto 0)비트 암호키를 32(31 downto 0)비트씩 L 블록 A, B, C, D와 R 블록 E, F, G, H으로 키 생성부의 입력으로 전달한다. L 블록과 R 블록은 같은 연산 과정을 수행하게 되는데 각 단계마다 회전 상수 KC_i를 적용하여 G 함수의 입력값으로 전달하고, G 함수의 결과로 생성된 L 블록과 R 블록의 출력값은 서로 XOR 되어 각 라운드의 키를 생성한다. 생성된 라운드 키는 키 저장부에 저장되어 데이터패스부의 입력되기를 기다리게 된다.

2.2 데이터패스부 설계

데이터패스부는 암호와 복호를 담당하는 블록으로 F 함수부와 레지스터를 포함하고, F 함수부는 G 함수부를 포함하여 연산을 수행한다. 그림 2는 키 생성부와 데이터패스부의 관계를 나타낸 것이다.

키 생성기에서 생성된 키는 키 저장부에 저장되어 1

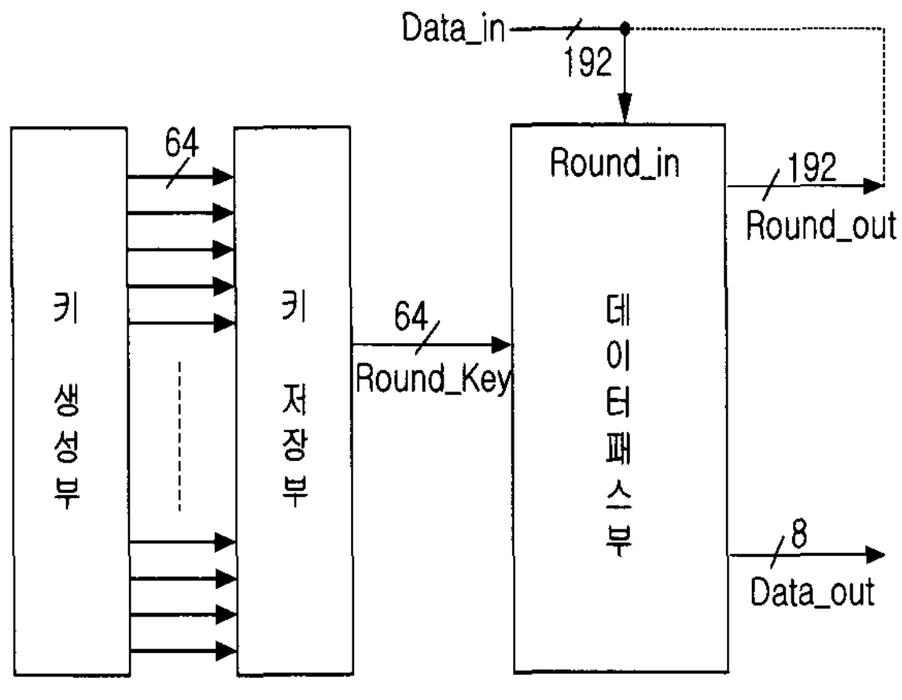


그림 2. 키 생성부와 데이터패스부
Fig. 2. Key generation and datapath block.

개씩 순차적으로 라운드 키를 데이터패스부에 전달한다. 192비트 입력 데이터는 64비트 단위의 세 부분으로 분리하여 처리하며, 처음 수행시 1 라운드의 입력 Round_in은 첫 번째 라운드 키와 함께 라운드 함수를 수행하여 1 라운드 출력값을 생성한다. 생성된 라운드 출력값은 다음 2 라운드의 입력값으로 전달되고, 여기에 두 번째 라운드 키를 적용하여 2 라운드 출력값을 생성한다. 이렇게 생성된 출력값은 다시 3 라운드의 라운드 입력값으로 전달되고, 여기에 세 번째 라운드 키를 적용하여 3 라운드 출력값을 생성하게 된다. 이러한 동작을 16라운드까지 반복하며 최종 16 라운드 출력값은 상위 64비트와 중간 64비트, 하위 64비트가 서로 교차하여 192비트의 암호화된 출력값을 생성한다.

라운드 함수부 내에는 F 함수부가 동작하고, F 함수부는 G 함수 3개와 가산기(Adder) 연산 3개로 이루어지며, F 함수는 64비트의 데이터와 64비트 라운드 키를 입력값으로 갖는다.

라운드 함수안에서 F 함수는 설계시 componet로 선언하였고, 각각 192비트의 라운드 입력과 레지스터 입력은 세 부분의 L 블록(191 downto 128), M 블록(127 downto 64), R 블록(63 downto 0)로 나누어 전달된다. 라운드 입력의 M 블록은 라운드 키와 함께 F 함수의 입력값으로 전달되고, 생성된 F 함수 출력값은 라운드 입력의 R 블록과 XOR 되어 192비트의 레지스터 입력 중 M 블록으로 전달되고, 라운드 입력의 M 블록은 레지스터 입력의 L 블록으로, 그리고 라운드 입력의 나머지 64비트 L 블록은 F 함수 출력값과 XOR 되어 레지스터 입력의 R블록으로 배치된다. 이렇게 구성된 192비트 $L(64)\|M(64)\|R(64)$을 레지스터를 거쳐 라운드 함수의 출력값으로 전달되도록 설계하였다.

그림 3은 G 함수와 S 함수의 관계를 나타낸 것으로 G 함수는 32비트 데이터를 4개의 블록으로 나누어 각각을 Sbox1, Sbox2에 분산되어 연산을 하며, Sbox1, Sbox2, SSbox를 component로 선언하여 설계하였다.

연산 결과를 $m1$, $m2$, $m3$, $m4$와 비트와 AND 연산후 XOR 연산을 통해 결과 값을 출력한다. Sbox1, Sbox2를 각각 2개씩 사용한다. Sbox의 구현 방식에 따라 회로의 크기와 지연 시간이 결정된다.

Sbox1과 Sbox2는 각각 8비트 데이터 256개에 해당하는 값을 갖는 멀티플렉스의 형태로 설계한다. G 함수와 S-box를 어떠한 방식으로 구현하느냐에 따라 칩의 면적과 속도를 향상시킬 수 있다. SEED 알고리즘의 특성상 가산기와 G 함수의 많은 사용으로 가산

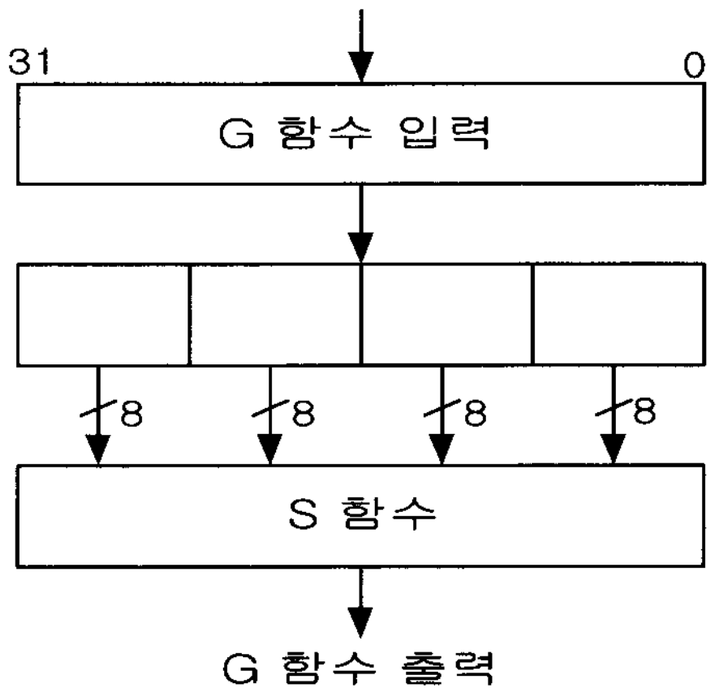


그림 3. G 함수와 S 함수 블록
Fig. 3. G function and S function block.

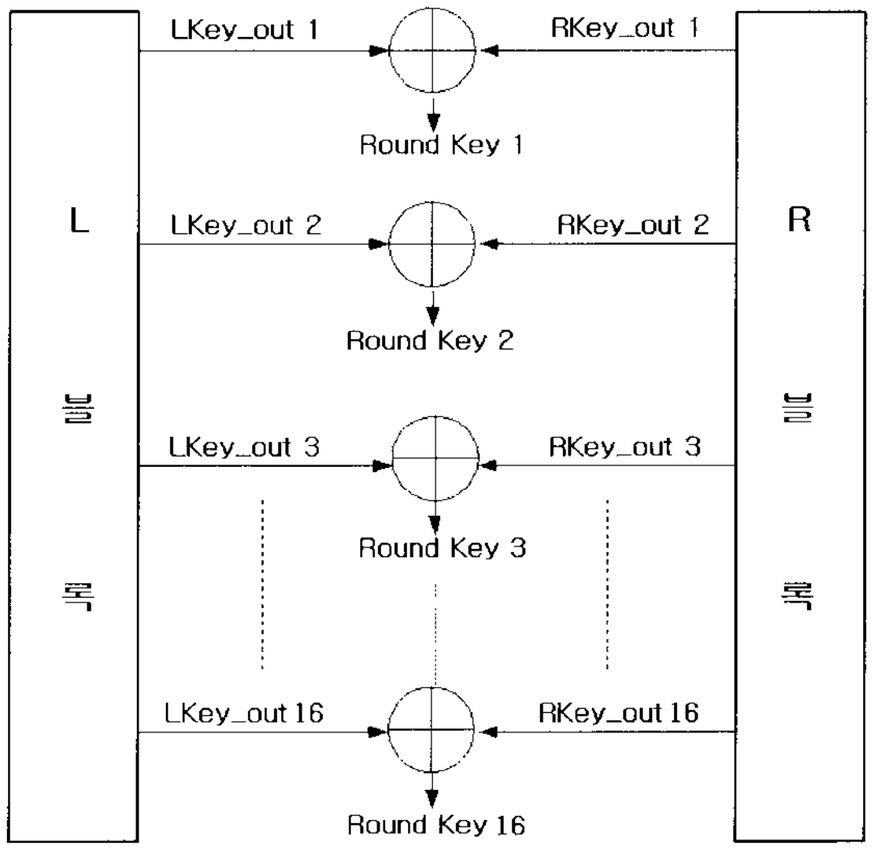


그림 4. 라운드 키 생성 블록도
Fig. 4. Block diagram for the round key generation.

기와 G 함수의 속도 향상이 전체 성능에 영향을 미친다.

그림 4는 키 생성 알고리즘에 의해 L 블록에서 생성된 LKey_out과 R 블록에서 생성된 RKey_out이 서로 XOR 되어 16개의 라운드 키 생성 과정을 나타낸 것이다. 키 생성 알고리즘에 의해 생성되어 암호화와 복호화를 위해 데이터패스부에 전달되는 키를 Round Key로 나타낸다. 키 생성 알고리즘에 의해 생성된 16개의 라운드 키는 데이터패스부에 전달되어 총 16회의 라운드 함수를 수행한다.

192비트의 입력 데이터는 16개의 라운드 키와 함께 암·복호화를 위한 데이터패스부의 입력으로 전달되어 라운드의 출력값을 생성한다.

그림 5는 키생성 알고리즘 블록도를 나타낸 것이다.

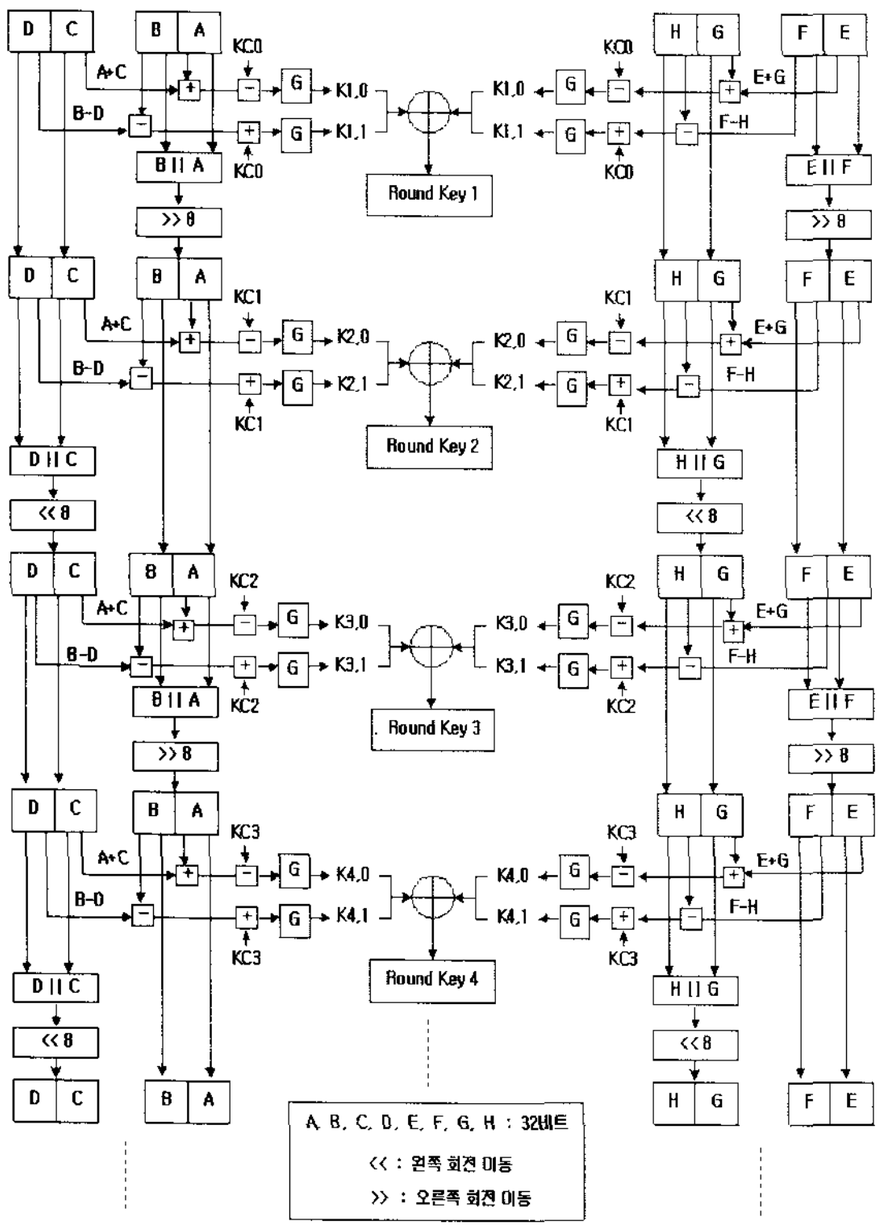


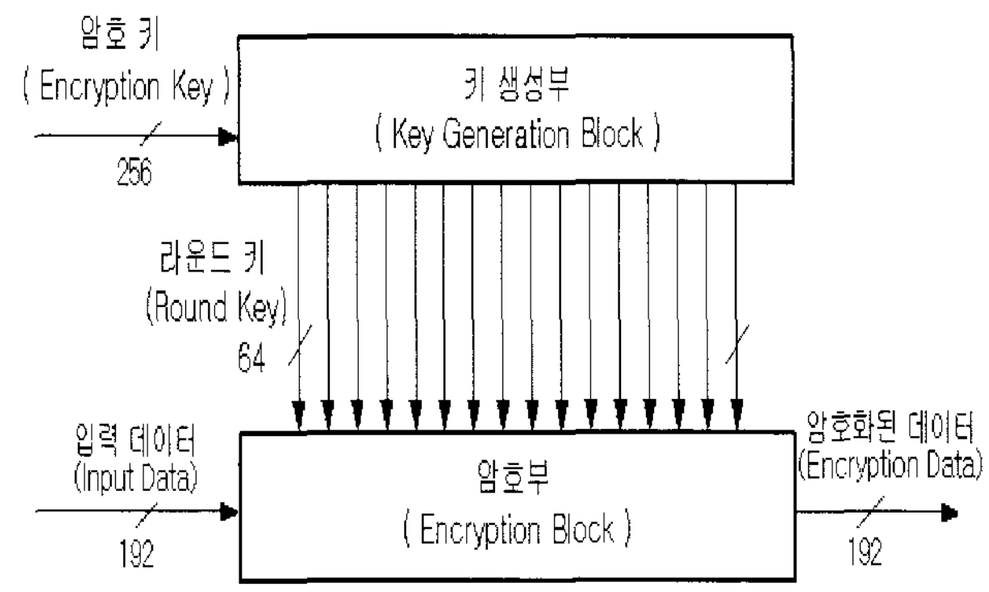
그림 5. 키 생성 알고리즘 블록도

Fig. 5. Block diagram of the key generation algorithm.

Ⅲ. 알고리즘을 적용한 모듈 설계

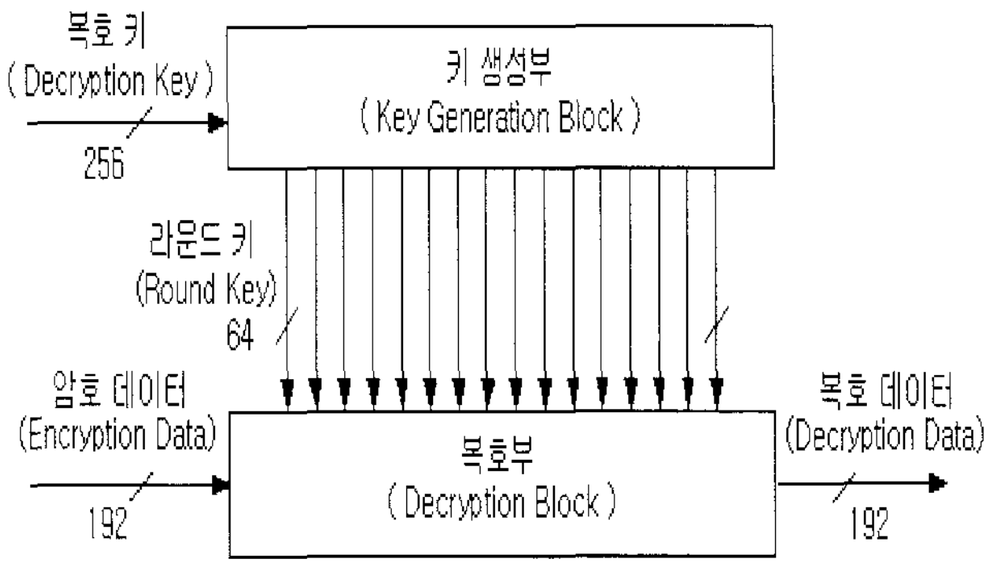
모듈 설계에 있어서 AD(Analog to Digital) 컨버터를 통해 변환된 형태의 음성 신호인 디지털 신호를 암·복호화 모듈의 입력값으로 전달하고, 전달된 데이터는 암·복호화 과정을 거친 후, DA(Digital to Analog) 컨버터를 거쳐 암·복호화된 아날로그 신호로 다시 변환되어 상대방에게 전달되는 것을 기반으로 하였다.

암호화 모듈 설계 시 블록암호알고리즘의 자체 특성 때문에 많은 하드웨어 자원을 필요로 하므로 SEED의 경우는 암호화 또는 복호화 과정에서 최적화된 하드웨어 설계가 요구되고, 모든 알고리즘이 하나의 모듈에서 부가적인 요소 없이 신호의 암호화가 이루어지도록 단일 라운드 방식을 사용하였다. 이것은 SEED의 16개 라운드를 1개의 단일 라운드 프로세서에 의해서 반복 수행하는 것으로 속도는 전라운드형에 비해 느리지만 많은 하드웨어 자원을 절약할 수 있다. 256비트의 암호키와 192비트 입력 데이터는 각각 버퍼에 저장되었다가 키 생성부와 암호부로 전달된다. 키 생성부에서 16개의 킷값이 출력되어 암호부의 라운드 키 입력값으로 전달되며, 이것은 192비트 데이터 입력값과 함께 암호화 과정을 거쳐 최종으로 192비트의 암호화된 출력값



(a) 키 생성부와 암호화부

(a) The key generation block and encryption block.



(b) 키 생성부와 복호화부

(b) The key generation block and decryption block.

그림 6. 키 생성부와 암·복호화부

Fig. 6. The key generation block and encryption & decryption block.

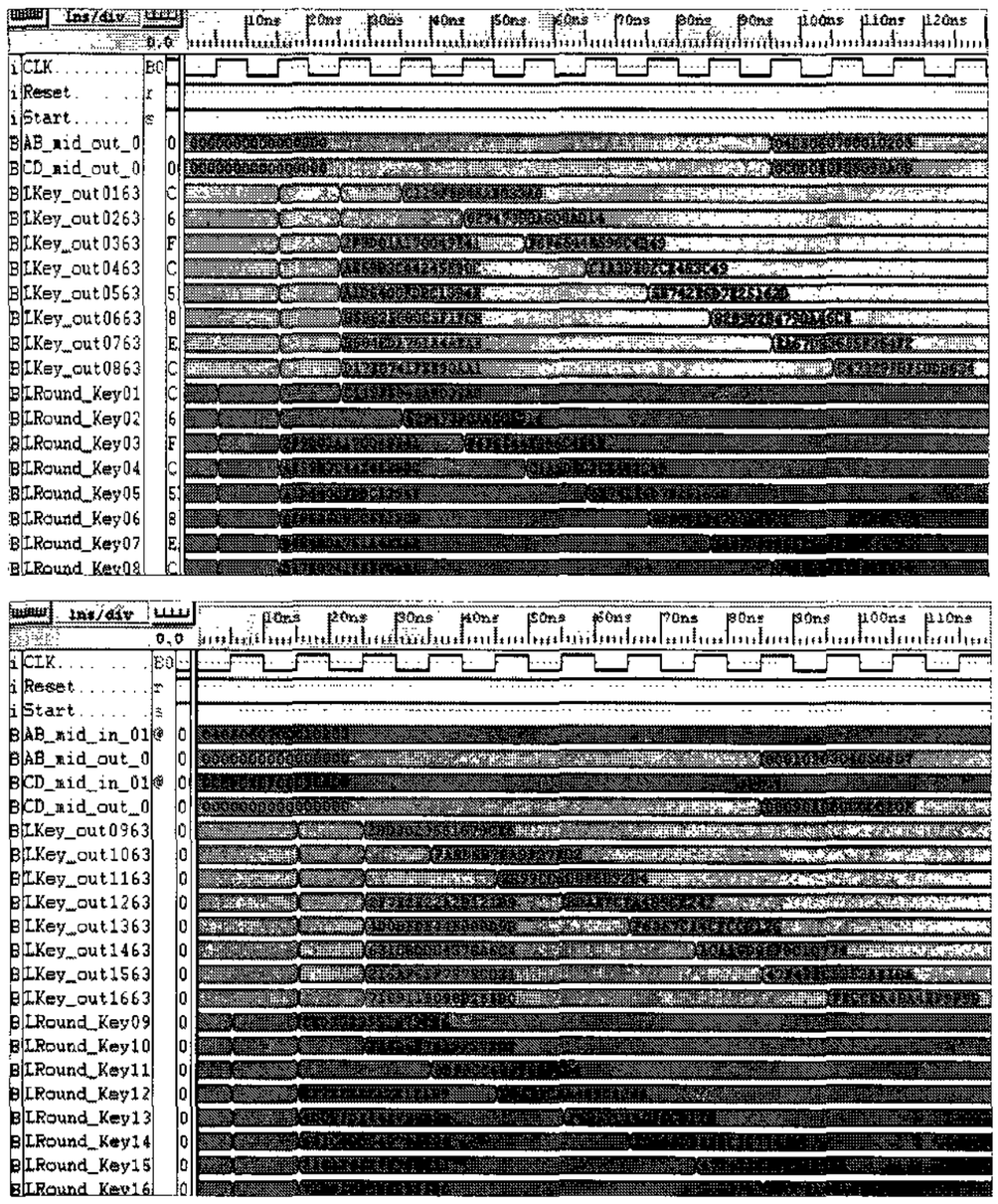
을 생성한다.

그림 6은 키 생성부와 암·복호화부와의 흐름을 나타낸 것으로 키 생성부에서는 256비트 키 입력값을 받아서 64비트의 키 생성값 16개를 출력하여 암호부의 각 라운드의 키 입력값으로 전달한다.

복호화의 경우에는 암호화된 데이터를 입력하고, 키의 순서만 역으로 적용하여 복호화된 값을 얻는다.

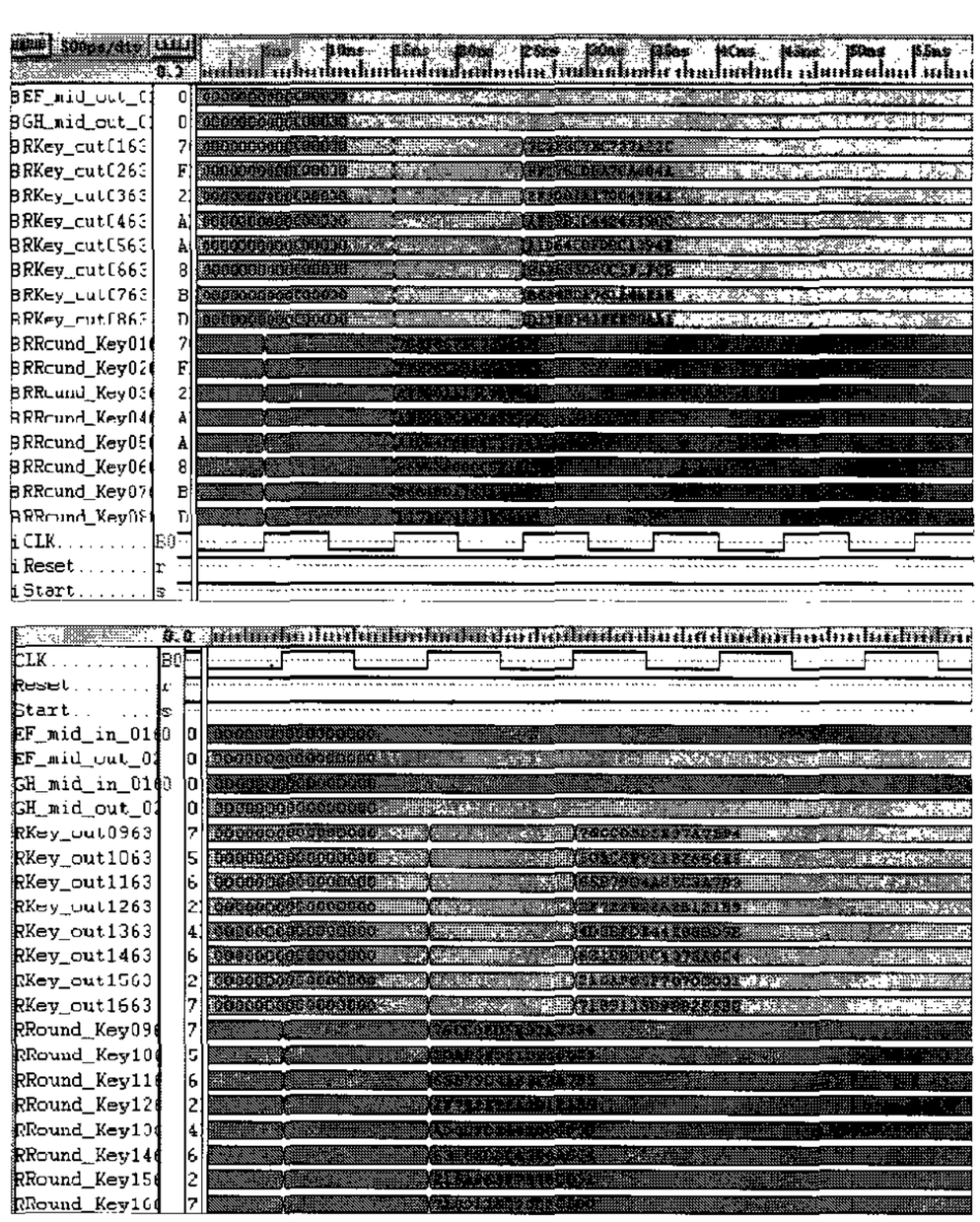
Ⅳ. 실험 및 고찰

본 실험에서는 음성 신호를 암호화하기 위해 음성 아날로그 신호를 디지털 신호로 변환하는 8비트 분해능의 AD/DA 컨버터(Converter)를 사용을 가정하여 설계하였다. 모듈은 VHDL 코드를 사용하여 탑 다운(Top down) 방식으로 설계하였으며, S-box 관련 요소, F 함수와 G 함수 그리고 라운드 함수는 F 함수와 G 함수가 포함될 수 있도록 구성하여 Foundation Express Tool로 시뮬레이션과 합성(synthesis)을 통해 칩의 성능을 고찰하였다.



(a) 생성된 L 블록의 키

(a) The key of generated L block.



(b) 생성된 R 블록의 키

(b) The key of generated R block.

그림 7. 생성된 L 블록과 R 블록의 키

Fig. 7. The key of generated L & R block.

표 1. 최종 라운드 키

Table 1. Final Round Key.

라운드 키(Round Key)			
1	9FA9DBD967D7918C	9	7E0B85D1F01DEF72
2	F4B019DB31CAC55E	10	4E4B2AA210D518E7
3	5EF29D96B568D508	11	5C2A8EE2E1A33567
4	DC68C6FFA60DD545	12	2D8EA92D0D2D03FB
5	E62A5FFA26E42F73	13	1F081BB3E8443CBD
6	7BC75070FE555905	14	059BBC3B6EB9A1B0
7	591345CE3657FA5C	15	89A4D669E042218B
8	EBA0B150D2E4BC95	16	B183C8F45F5DCA2B

그림 7은 생성된 L 블록과 R 블록 킷값을 나타낸 것으로 모듈에서는 킷값의 각각 L 블록과 R 블록이 XOR 되어 최종 16개의 라운드 키를 순차적으로 생성하며, 표 1은 최종 라운드 키를 나타낸 것이다.

모듈 설계 시 하드웨어 자원 축소를 위해 단일 라운드 방식[6~8]을 사용하여 256비트 키 입력과 192비트 입력 데이터로 모듈을 수행한 결과 입력 데이터에 대한 암호화 데이터와 암호화된 입력에 대한 복호화 데이터를 얻을 수 있었다. SEED 알고리즘에서 하드웨어 비중을 많이 차지하는 S-box 연산을 간소화하여 구현하였으며, 기존보다 하드웨어 자원 활용과 속도 효율면에서 약 26.3% 정도 향상됨을 확인하였다.

V. 결 론

본 논문에서는 음성 신호의 암·복호화를 위해 국내 표준으로 지향하는 표준 SEED 암호 알고리즘을 192비트 입력 데와 256비트 키 입력으로 확장하고, 16 라운드 함수 수행으로 암호화 강도를 향상시킨 개선된 알고리즘을 제안하였다.

제안한 알고리즘의 키 생성 알고리즘에서의 성능 분석을 통해 킷값의 안전성을 확인하였으며, 모듈은 실제 음성 아날로그 신호를 AD/DA 컨버터를 이용하여 모듈의 암호화 또는 복호화를 수행하는 것을 가정하여 설계하였다. 이것은 C 컴파일러와 Foundation Express Tool로 시뮬레이션 하여 입력 데이터의 암·복호화 결과를 확인할 수 있었으며, 표준 SEED 암호 알고리즘으로 구현한 것보다 하드웨어 자원 활용과 속도 효율면에서 약 26.3% 정도 향상됨을 확인하였다.

SEED 암호 알고리즘의 상당한 비중과 크기를 차지하는 S-box의 효율적인 구성이 수반된다면 자원의 활용과 속도의 효율성이 더욱 더 개선될 것으로 사료된다. 그러나, 향후 실질적으로 다양한 통신 방식에 적용

하기 위해서는 통신 조건 및 방식, 주변 모듈과의 호환 등을 고려한 최적의 알고리즘 설계와 별도의 암·복호화 구현을 위한 연구가 필요할 것으로 사료된다.

참 고 문 헌

[1] T. Schaffer, A. Glaser, S. Rao, P. Franzon. "A ilp-chip implementation of the Data Encryption Standard(DES)," Multi-Chip Module Conference, MCMC '97., 1997 IEEE, pp.13-17, 1997.

[2] Seung-Jo Han, Heang-Soo Oh, Jongan Park, "The improved data encryption standard(DES) algorithm," Spread Spectrum Techniques and Applications Proceedings, IEEE 4th International Symposium on, vol. 3, pp.1310-1314, 1996.

[3] "128비트 블록 암호 알고리즘(SEED) 개발 및 분석 보고서," 한국정보보호센터, 1998. 12.

[4] Xilinx, The Programmable Logic Data Book, Xilinx, 1999년.

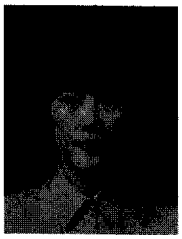
[5] David Van Bout 원저, 김만복 편역, FPGA 이론 및 실습, 홍릉과학출판사, 2000년.

[6] 박종국, "128비트 블록 암호 알고리즘(128-DES)의 설계 및 구현," 숭실대학교 정보과학대학원 정보통신학과, 석사학위논문, 1997년.

[7] 신혜진, "SEED 암호화 프로세서의 하드웨어 설계 및 구현," 한국항공대학교, 2000년.

[8] 서영호, "대한민국 표준 128비트 블록 암호 알고리즘의 하드웨어 구현에 관한 연구," 광운대학교, 2000년.

저 자 소 개



안 인 수(정회원)

1992년 국민대학교 전자공학과 학사 졸업.

1994년 국민대학교 전자공학과 석사 졸업.

2002년 국민대학교 전자공학과 박사 졸업.

2003년~현재 경인여자대학 컴퓨터정보학부 조교수

<주관심분야 : 통신, 암호, 신호처리>