(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2016-0105617
(43) 공개일자 2016년09월07일

(51) 국제특허분류(Int. Cl.)
H04L 9/06 (2006.01)
(52) CPC특허분류
H04L 9/0631 (2013.01)
H04L 2209/805 (2013.01)
(21) 출원번호 10-2015-0028240
(22) 출원일자 2015년02월27일
심사청구일자 2015년02월27일

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
홍석희
서울특별시 노원구 덕릉로71길 30, 102동 1701호 (중계동, 양지대림아파트)
김성곤
서울특별시 서초구 잠원로 150, 104동 103호 (잠원동, 잠원한신아파트)
김현민
서울특별시 성북구 길음로 118 414동 1801호 (길음동,대림e편한세상아파트)
(74) 대리인
특허법인충현

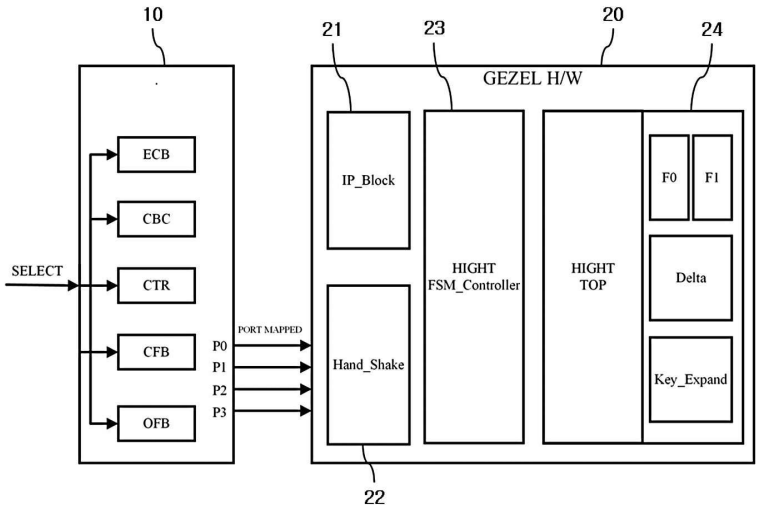
전체 청구항 수 : 총 5 항

(54) 발명의 명칭 **ＨＩＧＨＴ를 이용한 경량암호 장치**

(57) 요 약

본 발명은 HIGHT를 이용한 경량암호 장치에 관한 것으로, 블록 암호 운용 방식에 따르는 하드웨어 처리기 및 하드웨어 처리기와 연결되어 통신하는 GEZEL 코-프로세서(co-processor)를 포함하며, GEZEL 코-프로세서는, GEZEL IP(intellectual property) 블록 및 경량암호 HIGHT 기법을 데이터패스를 이용한 유한상태기계(Finite state machine with datapath, FSMD) 방식으로 동작시키는 하드웨어 IP를 포함한다.

대 표 도 - 도6



이 발명을 지원한 국가연구개발사업
과제고유번호 NIPA-2014-H0301-14-1004
부처명 미래창조과학부
연구관리전문기관 정보통신산업진흥원
연구사업명 ITRC
연구과제명 제어시스템 보안 연구
기 여 율 1/1
주관기관 고려대학교 산학협력단
연구기간 2014.01.01 ~ 2014.12.31

명 세 서

청구범위

청구항 1

블록 암호 운용 방식에 따르는 하드웨어 처리기; 및

상기 하드웨어 처리기와 연결되어 통신하는 GEZEL 코-프로세서(co-processor);를 포함하며,

상기 GEZEL 코-프로세서는,

GEZEL IP(intellectual property) 블록; 및

경량암호 HIGHT 기법을 데이터패스를 이용한 유한상태기계(Finite state machine with datapath, FSMD) 방식으로 동작시키는 하드웨어 IP;를 포함하는 것을 특징으로 하는 경량암호 장치.

청구항 2

제 1 항에 있어서,

상기 하드웨어 IP는,

상기 경량암호 HIGHT 기법의 각 데이터패스를 유한상태기계로 정의하여 제어하는 제어기;

상기 경량암호 HIGHT 기법의 암호 알고리즘을 수행하는 'HIGHT_round_dp' 데이터패스;

키 스케줄(key schedule)을 수행하는 'KEY expand' 데이터패스;

룩-업 테이블(look-up table) 형태로 내부 함수의 출력 값을 갖는 F_0 및 F_1 데이터패스; 및

라운드의 서브키(subkey) 조합에 사용되는 델타(delta)값을 저장하는 'delta' 데이터패스;를 포함하는 것을 특징으로 하는 경량암호 장치.

청구항 3

제 2 항에 있어서,

상기 제어기는,

서브키를 16개 사용하는 4라운드 수행 후 키스케줄을 수행하되,

4라운드 수행시 레지스터의 위치를 변경하는 리타이밍(retiming)을 적용하여 크리티컬 패스(critical path)의 지연(delay)을 감소시키는 것을 특징으로 하는 경량암호 장치.

청구항 4

제 1 항에 있어서,

상기 유한상태기계는,

통신 모듈과 소프트웨어의 통신이 연결되었는지를 확인하고 상기 통신 모듈로부터 평문과 키를 입력받는 'Check_start' 상태;

입력된 상기 평문과 상기 키를 HIGHT 내부 레지스터에 저장하고, HIGHT 암호화를 수행하는 동안 사용될 레지스터를 초기화하는 'Init' 상태;

라운드 시작 전, 입력된 상기 평문에 대해서 화이트닝키를 삽입하는 'Pre_round' 상태;

HIGHT의 라운드키로부터 2라운드에 사용될 8개의 서브키를 생성하되, delta 데이터패스의 출력값과 현재 키를 조합하여 서브키를 생성하는 'Sub_key' 상태;

생성된 상기 8개의 서브키를 이용하여 2라운드를 수행하고 출력된 값을 레지스터에 저장하고, 마지막 라운드는 32라운드 알고리즘을 수행하고 최종 화이트닝키를 삽입한 값을 출력하되, 키스케줄을 시작하기 전 4라운드 중 처음 2라운드만 수행한 경우 다시 서브키를 생성하는 상기 'sub_key' 상태로 회귀되고, 4라운드가 모두 수행되었으면 키스케줄을 수행하는 'Round' 상태;

라운드의 서브키 조합에 사용되는 델타 데이터패스의 입력값인 delta_constant를 갱신한 후 레지스터에 저장하는 'Delta_change' 상태;

키를 갱신하는 키스케줄을 수행하되, 현재 키의 상위 64비트와 하위 64비트 각각 8비트씩 좌측 순환 이동(left shift rotation)시킨 값을 키 레지스터에 저장하는 'Key_schedule' 상태; 및

최종 라운드까지 수행된 경우 암호문을 외부 통신 모듈로 전송하고, HIGHT의 시작 상태를 알려주는 레지스터를 다시 초기화시키는 'Finish' 상태를 포함하는 것을 특징으로 하는 경량암호 장치.

청구항 5

제 1 항에 있어서,

상기 GEZEL 코-프로세서는,

하드웨어 및 소프트웨어 간의 메시지 송수신시, HIGHT 하드웨어 코어의 암호화 과정과 데이터 송수신 과정을 각각 분리하여 처리하고, 서로 통신이 확립되면 처리 완료를 알리는 완료 시그널(done signal)이 도착 시 대기시간 없이 즉시 데이터를 송수신하는 것을 특징으로 하는 경량암호 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 경량암호 HIGHT의 하드웨어 설계 방법에 관한 것으로, 보다 상세하게는 HIGHT를 이용한 경량암호의 암호화를 효율적으로 동작하게 하기 위하여 하드웨어로 성능을 최적화시키기 위한 경량암호 장치에 관한 것이다.

배 경 기 술

[0002] 암호 시스템은 크게 공개키 암호 시스템과 암/복호화에 동일한 키를 사용하는 대칭키 암호 시스템으로 분류된다. RFID나 USN 같은 저전력/경량화 환경에서는 안전성을 보장하면서도 효율적인 암호 알고리즘을 필요로 한다. 이러한 환경에서 평문을 블록단위로 암호화를 수행하는 대칭키 경량 암호가 많이 사용된다. 그리고, 경량암호는 단독으로 사용되기보다는 다양한 길이를 가진 메시지를 암호화하기 위해 다양한 운영 모드와 함께 운용된다.

[0003] 대칭키 암호 HIGHT는 RFI, USN 등과 같이 저전력, 경량화를 요구하는 컴퓨팅 환경에서 기밀성을 제공하기 위해 개발된 암호이다. HIGHT는 초경량 암호 알고리즘으로 128비트 마스터키, 64비트 평문으로부터 64비트 암호문을 출력하는 32라운드 8-branch type generalized feistel 구조를 가진다. 또한, 제한적 자원을 갖는 환경에서 구현될 수 있도록 8비트 단위의 기본적인 산술 연산들인 법덧셈, XOR, 좌측순환이동만으로 설계되었다.

[0004] 기존의 경량암호 HIGHT의 최적 설계는 주로 소프트웨어적으로 이루어졌고, 관련 연구가 대부분 소프트웨어 기반으로 이루어졌다. 적은 수의 하드웨어 설계 최적화에 대한 연구가 진행이 되었지만, 그러한 하드웨어 최적 설계는 우선적으로 경량암호 HIGHT에 대한 하드웨어 단독 설계에 대한 최적화에 그 초점이 맞춰져 있었다. 이하에서 제시되는 선행기술문헌은 하드웨어와 소프트웨어를 함께 활용한 설계 방법론을 제안하였다.

선행기술문헌

비특허문헌

[0005] (비특허문헌 0001) Patrick R. Schaumont, "A Practical Introduction to Hardware/Software Codesign," Springer, Dec. 2012.

발명의 내용

해결하려는 과제

[0006] 본 발명이 해결하고자 하는 기술적 과제는 종래의 대칭키 암호화 알고리즘인 HIGHT에서 하드웨어 및 소프트웨어를 통합적으로 설계한 최적화 기법이 존재하지 않았으며, 그로 인해 통합 설계 과정에서 나타날 수 있는 데드락(deadlock), 레이턴시(latency) 및 경쟁 효과(race condition) 발생의 문제를 해결하고자 한다.

과제의 해결 수단

[0007] 상기 기술적 과제를 해결하기 위하여, 본 발명의 일 실시예에 따른 경량암호 장치는, 블록 암호 운용 방식에 따르는 하드웨어 처리기; 및 상기 하드웨어 처리기와 연결되어 통신하는 GEZEL 코-프로세서(co-processor);를 포함하며, 상기 GEZEL 코-프로세서는, GEZEL IP(intellectual property) 블록; 및 경량암호 HIGHT 기법을 데이터패스를 이용한 유한상태기계(Finite state machine with datapath, FSMD) 방식으로 동작시키는 하드웨어 IP;를 포함한다.

[0008] 일 실시예에 따른 상기 경량암호 장치에서, 상기 하드웨어 IP는, 상기 경량암호 HIGHT 기법의 각 데이터패스를 유한상태기계로 정의하여 제어하는 제어기; 상기 경량암호 HIGHT 기법의 암호 알고리즘을 수행하는 'HIGHT_round_dp' 데이터패스; 키 스케줄(key schedule)을 수행하는 'KEY expand' 데이터패스; 룩-업 테이블(look-up table) 형태로 내부 함수의 출력 값을 갖는 F_0 및 F_1 데이터패스; 및 라운드의 서브키(subkey) 조합에 사용되는 델타(delta)값을 저장하는 'delta' 데이터패스;를 포함한다.

[0009] 일 실시예에 따른 상기 경량암호 장치에서, 상기 제어기는, 서브키를 16개 사용하는 4라운드 수행 후 키스케줄을 수행하되, 4라운드 수행시 레지스터의 위치를 변경하는 리타이밍(retiming)을 적용하여 크리티컬 패스(critical path)의 지연(delay)을 감소시킬 수 있다.

[0010] 일 실시예에 따른 상기 경량암호 장치에서, 상기 유한상태기계는, 통신 모듈과 소프트웨어의 통신이 연결되었는지를 확인하고 상기 통신 모듈로부터 평문과 키를 입력받는 'Check_start' 상태; 입력된 상기 평문과 상기 키를 HIGHT 내부 레지스터에 저장하고, HIGHT 암호화를 수행하는 동안 사용될 레지스터를 초기화하는 'Init' 상태; 라운드 시작 전, 입력된 상기 평문에 대해서 화이트닝키를 삽입하는 'Pre_round' 상태; HIGHT의 라운드키로부터 2라운드에 사용될 8개의 서브키를 생성하되, delta 데이터패스의 출력값과 현재 키를 조합하여 서브키를 생성하는 'Sub_key' 상태; 생성된 상기 8개의 서브키를 이용하여 2라운드를 수행하고 출력된 값을 레지스터에 저장하고, 마지막 라운드는 32라운드 알고리즘을 수행하고 최종 화이트닝키를 삽입한 값을 출력하되, 키스케줄을 시작하기 전 4라운드 중 처음 2라운드만 수행한 경우 다시 서브키를 생성하는 상기 'sub_key' 상태로 회귀되고, 4라운드가 모두 수행되었으면 키스케줄을 수행하는 'Round' 상태; 라운드의 서브키 조합에 사용되는 델타 데이터패스의 입력값인 delta_constant를 갱신한 후 레지스터에 저장하는 'Delta_change' 상태; 키를 갱신하는 키스케줄을 수행하되, 현재 키의 상위 64비트와 하위 64비트 각각 8비트씩 좌측 순환 이동(left shift rotation)시킨 값을 키 레지스터에 저장하는 'Key_schedule' 상태; 및 최종 라운드까지 수행된 경우 암호문을 외부 통신 모듈로 전송하고, HIGHT의 시작 상태를 알려주는 레지스터를 다시 초기화시키는 'Finish' 상태를 포함한다.

[0011] 일 실시예에 따른 상기 경량암호 장치에서, 상기 GEZEL 코-프로세서는, 하드웨어 및 소프트웨어 간의 메시지 송수신시, HIGHT 하드웨어 코어의 암호화 과정과 데이터 송수신 과정을 각각 분리하여 처리하고, 서로 통신이 확립되면 처리 완료를 알리는 완료 시그널(done signal)이 도착 시 대기시간 없이 즉시 데이터를 송수신할 수 있다.

발명의 효과

[0012] 본 발명의 실시예들은, 경량암호 HIGHT를 하드웨어로 구현하고, 이를 8051 프로세서나 다른 소프트웨어 운용 플랫폼상에서 C 언어 등의 소프트웨어 개발언어로 개발된 운용모드들과 호환이 되도록 최적화하여 설계함으로써, 기존의 하드웨어만을 고려한 설계 기법에 비해 추가적인 통합 설계시의 동작 검증 없이 바로 적용이 가능한 HIGHT 하드웨어 IP를 효율적으로 구현할 수 있으며, 소프트웨어와 하드웨어 간의 데이터 송수신시 발생할 수 있는 지연시간을 최소화하기 위하여 하드웨어 상에서 개선된 운용방식을 적용함으로써 전체적인 하드웨어 성능을 향상시킬 수 있다.

도면의 간단한 설명

[0013] 도 1은 HIGHT 알고리즘의 암호화 과정을 설명하기 위한 도면이다.

도 2는 HIGHT 알고리즘의 1~31 라운드 함수를 설명하기 위한 도면이다.

도 3은 HIGHT 알고리즘의 마지막 32 라운드 함수를 설명하기 위한 도면이다.

도 4는 HIGHT 키 스케쥴 알고리즘을 예시한 도면이다.

도 5는 본 발명의 일 실시예에 따른 HIGHT 유한상태기계(FSM)의 전체 흐름을 도시한 도면이다.

도 6는 본 발명의 일 실시예에 따른 HIGHT의 HW/SW 통합 설계 기법을 적용한 통합 디자인의 전체 구조도를 도시한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0014] 이하에서는 도면을 참조하여 본 발명의 실시예들을 구체적으로 설명하도록 한다. 다만, 하기의 설명 및 첨부된 도면에서 본 발명의 요지를 흐릴 수 있는 공지 기능 또는 구성에 대한 상세한 설명은 생략한다.

[0015] 본 발명의 실시예들에 따라 최적화된 HIGHT의 하드웨어 설계는 하드웨어 단독 사용 및 설계 시 최적화뿐만 아니라, HIGHT를 소프트웨어와 연동하여 사용시 데드락(deadlock)이 발생하여 레이턴시(latency) 측면에서 손해가 발생할 수 있는 부분까지 고려하여 하드웨어 최적 설계를 진행하였다. 또한, 본 발명의 실시예들은 GEZEL 하드웨어 설계 언어를 이용하여 설계하여 본 발명을 통해 제시되는 하드웨어 IP를 소프트웨어와 같이 연동하여 운용시 발생할 수 있는 경쟁 효과(race condition)를 최소화하였다.

[0016] 이를 위해, 사이클(cycle) 기반의 설계기법을 적용한 GEZEL을 이용하여 설계하여 운용환경에 따라 발생할 수 있는 상기와 같은 문제를 구현 단계에서 해결하였다. 또한, 경량암호의 특성상 최소한의 면적으로 구현될 수 있도록 설계하였으며, 경량암호가 저전력 장비에서 동작함을 고려하였다.

[0017] 일반적으로 블록 암호가 운용 환경에 따라 운영 모드를 다르게 설정함에도 불구하고, 종래에는 이러한 면을 고려하지않고 하드웨어 최적화에만 초점이 맞춰져 있는 문제점에 주목하였다. 따라서, 본 발명의 실시예들은 개발된 최적화된 하드웨어 IP 코어가 다양한 운용 모드와 연동시 성능의 저하 없이 운용될 수 있도록 이러한 운용 모드가 탑재된 경량암호를 포함한 블록암호들이 동작하는 하드웨어/소프트웨어 통합 운용에 적합하도록 최적화하였다.

[0018] HIGHT 암호에서 사용되는 평문 P와 암호문 C는 다음의 수학식 1, 2와 같이 8비트 단위로 나뉘어 표기된다.

수학식 1

[0019] $$P = P_7 || P_6 || P_5 || P_4 || P_3 || P_2 || P_1 || P_0$$

수학식 2

[0020] $$C = C_7 || C_6 || C_5 || C_4 || C_3 || C_2 || C_1 || C_0$$

[0021] 또한, 각 라운드 출력값 X_i는 다음과 같이 8비트 단위로 나뉘어 표기된다.

수학식 3

[0022] $$X_i = X_{i,7} \| X_{i,6} \| X_{i,5} \| X_{i,4} \| X_{i,3} \| X_{i,2} \| X_{i,1} \| X_{i,0}$$

[0023] $WK_{0 \le i \le 7}$는 화이트닝키, $SK_{0 \le i \le 127}$는 라운드키이고, 내부 함수 $F_0(x)$와 $F_1(x)$는 다음과 같다.

수학식 4

[0024] $$F_0(x) = (x \lll 1) \oplus (x \lll 2) \oplus (x \lll 7)$$

수학식 5

[0025] $$F_1(x) = (x \lll 3) \oplus (x \lll 4) \oplus (x \lll 6)$$

[0026] 도 1은 HIGHT 암호알고리즘의 전체 암호화 과정을 나타내었다.

[0027] HIGHT의 암호화 과정은 초기 화이트닝키 삽입, 라운드 함수, 최종 화이트닝키 삽입 과정으로 구성되며, 각 암호화 연산 과정은 다음과 같다.

[0028] **1) 초기/최종 화이트닝키 삽입**

[0029] 도 1에서 확인할 수 있듯이 초기 라운드 진입 전과 최종 라운드 이후에 화이트닝키를 삽입한다. 화이트닝키($WK_{0 \le i \le 7}$)는 비밀키(MK)를 이용하여 다음과 같은 방법으로 생성한다.

수학식 6

[0030] $$WK_i = MK_{i+12}, \ 0 \le i \le 3$$

수학식 7

[0031] $$WK_i = MK_{i-4}, \ 4 \le i \le 7$$

[0032] **2) 라운드 함수**

[0033] 도 2는 HIGHT의 라운드 1~31의 라운드 함수를 나타낸 도면이고, 도 3은 HIGHT의 라운드 32의 라운드 함수를 나타낸 도면이다. 마지막 라운드는 라운드 1~31과 다르게 좌우 블록의 교차가 없다.

[0034] HIGHT의 키 스케줄은 128비트 비밀키를 입력받아 8비트 화이트닝키($WK_{0 \le i \le 7}$) 8개와 8비트 라운드키

($SK_{0 \le i \le 127}$) 128개를 도 4에 예시된 키 스케줄 과정을 통해 생성된다. 또한, LFSR을 이용하여 델타값 ($\delta_{0 \le i \le 127}$)을 만들어 라운드키 생성에 사용한다.

[0035] 이하에서 제시되는 본 발명의 실시예들에서는 효율적인 경량암호 HIGHT를 CPU 플랫폼에서 C언어 등의 소프트웨어로 개발된 모듈과 효과적으로 같이 운용할 수 있도록 하드웨어/소프트웨어 통합설계 및 운용에 적합하도록 설계되었고, HIGHT의 하드웨어 최적화에는 데이터패스를 유한상태기계로 정의하여 컨트롤하는 유한상태기계(FSM)를 언폴딩(unfolding)과 리타이밍(retiming) 등의 최적화 이론을 적용하여 최적화하였다. 또한, 소프트웨어와 하드웨어 간의 통신 시 발생하는 지연시간을 최소화하기 위하여 소프트웨어와 하드웨어 간의 통신 프로토콜을 HIGHT 하드웨어 단에서 최적화하여 설계하였다.

[0036] 본 발명의 실시예들에서 제안한 HIGHT의 하드웨어 구현에 관한 구체적인 내용은 다음과 같다.

[0037] **(1) HIGHT 하드웨어 구현 최적화**

[0038] HIGHT 하드웨어 구현을 각 데이터패스를 유한상태기계로 정의하여 컨트롤하는 유한상태기계 컨트롤러를 설계하여 각 데이터패스들을 이용한 연산이 가능하게 하였다.

[0039] HIGHT는 암호 알고리즘 전체를 수행하는 HIGHT_round_dp 데이터패스를 최상위 개체로 설정하고 키 스케줄을 수행하는 KEY expand 데이터패스를 설계하였다. 또한, look-up table 형태로 내부 함수 와 의 출력 값을 갖고 있는 데이터패스, 데이터패스, 마지막으로 라운드의 서브키 조합에 사용되는 델타값을 저장해둔 delta 데이터패스의 총 5개의 데이터패스로 나누어 설계를 하였다.

[0040] HIGHT는 알고리즘 구성상 서브키를 16개 사용하고 난 이 후에 입력된 비밀키의 상위 64비트와 하위 64비트가 각각 8비트씩 순환 왼쪽 시프트를 하는 것에 착안하여 최적화를 수행하였다. 즉, 서브키를 16개 사용하는 4라운드 수행 후 키 스케줄을 수행할 수 있게 구현하고, 4라운드 수행 시 발행하는 크리티컬 패스(critical path)의 증가문제를 해결하기 위하여 리타이밍(retiming) 기법을 적용하여 크리티컬 패스의 지연을 감소시켜 주었다.

[0041] **(2) HIGHT의 각 유한상태기계(FSM)에서의 구체적인 최적화 방법**

[0042] 도 5는 본 발명의 효율적인 HIGHT 알고리즘 구현에 적용한 아래 FSM의 전체 흐름을 나타낸다.

[0043] **2-1) Check_start 상태 (510)**

[0044] 통신 모듈과 소프트웨어의 통신이 연결되었는지 확인하여, 연결이 되지 않았다면 암호화를 시작하지 않는다. 정상적으로 연결이 되고 통신 모듈로부터 평문과 키를 입력 받았다면 다음 단계인 Init 상태로 넘어간다.

[0045] **2-2) Init 상태 (520)**

[0046] 외부 통신 모듈로부터 평문과 키를 입력 받은 후 HIGHT 내부 레지스터에 저장한다. 그리고 HIGHT 암호화를 수행하는 동안 사용될 레지스터를 초기화하는 상태이다.

[0047] **2-3) Pre_round 상태 (530)**

[0048] 라운드 시작 전에 입력 받은 평문에 대해서 화이트닝키를 삽입하는 과정을 수행한다.

[0049] **2-4) Sub_key 상태 (540)**

[0050] HIGHT의 라운드키로부터 2라운드에 사용될 8개의 서브키를 생성한다. delta 데이터패스의 출력값과 현재 키를 조합하여 서브키를 생성한다.

[0051] **2-5) Round 수행 상태 (550)**

[0052] 앞서 생성된 8개의 서브키를 이용하여 2라운드를 수행하고 그 값을 레지스터에 저장하는 상태이다. 이때 HIGHT는 마지막 라운드가 기존 라운드와 다르기 때문에 마지막 라운드는 32라운드 알고리즘을 수행하고 최종 화이트닝키도 삽입한 값을 출력한다

[0053] 이 상태에서는 키스케줄을 시작하기 전에 4라운드중 처음 2라운드만 수행했을 때는 다시 서브키를 생성하는 sub_key 상태로 회귀되고, 4라운드가 모두 수행되었으면 키스케줄을 수행하는 상태로 이행된다.

[0054] **2-6) Delta_change 상태 (560)**

[0055] 라운드의 서브키 조합에 사용되는 델타 데이터패스의 입력값인 delta_constant를 갱신한 후 레지스터에 저장한다.

[0056] **2-7) Key_schedule 상태 (570)**

[0057] 키를 갱신하는 키스케줄을 수행하는 상태이다. 현재 키의 상위 64비트와 하위 64비트 각각 8비트씩 좌측 순환 이동(left shift rotation)시킨 값을 키 레지스터에 저장한다.

[0058] **2-8) Finish 상태 (580)**

[0059] 최종 라운드까지 수행하였다면, 암호문을 외부 통신 모듈로 전송한다. 그리고 HIGHT의 시작 상태를 알려주는 레지스터를 다시 초기화시킨다.

[0060] **(3) HIGHT의 HW/SW 통합설계를 고려한 모듈간 통신 최적화**

[0061] 블록암호의 운용 모드인 ECB(Electronic Code Block), CBC(Cipher Block Chaining), CTR(Counter), CFB(Cipher FeedBack), OFB(Output FeedBack) 등이 본 발명의 실시예들을 통해 개발된 HIGHT 하드웨어 IP와 효과적으로 상호 통신하며 통작하게 하기 위하여 하드웨어 IP 상에서의 동작 프로토콜을 개선하여 레이턴시를 향상시켰다.

[0062] 본 발명에서 사용한 방법은 암호화와 HW/SW 간의 통신 메시지를 주고 받는 부분을 따로 계산하여 완료시그널(done)이 도착하면 추가적인 대기시간 없이 바로 메시지를 송/수신할 수 있게 하였다.

[0063] 도 6은 HIGHT 알고리즘에 대한 HW/SW 통합 설계 시 HIGHT 하드웨어 IP의 적용에 대한 전체 구조도이다.

[0064] 하드웨어 처리기(10)는 블록 암호 운용 방식에 따르며, 8051 마이크로프로세서(microprocessor)로 구현될 수 있다. 블록 단위로 암호화 되는 대칭키 암호는 다양한 길이의 입력과 출력을 보장하기 위해 운영 모드를 이용할 수 있으며, 도 6에서 대칭키 암호의 운용모드는 8051 마이크로프로세서를 사용하여 C언어로 구현할 수 있다.

[0065] GEZEL 코-프로세서(co-processor)(20)는 상기 하드웨어 처리기(10)와 포트-맵(port-mapped) 또는 메모리-맵(memory-mapped) 방식으로 연결되어 통신하며, GEZEL IP(intellectual property) 블록(21) 및 경량암호 HIGHT 기법을 데이터패스를 이용한 유한상태기계(Finite state machine with datapath, FSMD) 방식으로 동작시키는 하드웨어 IP(23, 24)를 포함한다.

[0066] GEZEL은 ARM, 8051 마이크로프로세서와의 통합 설계를 지원하기 때문에 다른 추가적인 구현 및 컴파일 없이 GEZEL에서 제공하는 IP block만 코드에 삽입 후 합성하면 8051 마이크로프로세서와의 통신을 통해 평문과 암호문을 전송할 수 있다.

[0067] 본 발명의 실시예들에서는 GEZEL로 구현된 암호 알고리즘의 코어와 8051 프로세서에서 구현된 ECB(Electronic Code Block), CBC(Cipher Block Chaining), CTR(Counter), CFB(Cipher FeedBack), OFB(Output FeedBack)의 5개의 운영모드를 통해 하드웨어/소프트웨어의 통합설계를 구축하였다.

[0068] 8051 프로세서에서는 사용자의 입력을 받아서 ECB, CBC, CTR, CFB, OFB의 5가지 운영모드 중 하나를 선택할 수 있다. 즉, 사용 플랫폼과 상황에 맞추어 운영모드와 암호를 선택할 수 있는 유연성을 제공한다. 또한 마스터키가 하드웨어에 저장된 것이 아닌 소프트웨어로부터 입력받는 것이기 때문에 추후 키가 유출되었을 경우 쉽게 수정 및 업데이트가 가능하다는 안전성 측면에서의 유연성을 갖는다.

[0069] 설계 측면에서도 하드웨어와 소프트웨어의 통합설계 시 각 대칭키 암호의 알고리즘 중 키스케줄 부분을 소프트웨어에서 구현하여 미리 각 라운드키 값을 사전 계산하여 라운드키를 하드웨어로 보내는 방식으로도 구현할 수 있다. 그에 따라 하드웨어에서는 키스케줄 연산을 처리하지 않아도 되기 때문에 면적 감소와 속도 증대 효과를 기대할 수 있다.

[0070] GEZEL을 이용한 통합설계에서 8051 마이크로프로세서가 하드웨어 입출력 장치와 통신을 할 때, 사용하는 방식은 메모리 맵 입출력(memory mapped I/O)방식과 포트 맵 입출력 (port mapped I/O)방식이 있다. 메모리 맵 입출력 방식은 통신을 할 때 입출력과 메모리의 주소 공간을 분리하지 않는 방식이고, 포트 맵 입출력 방식은 입출력 회로와 메모리 주소 공간을 분리하는 방식이다. 입출력 회로가 메모리에 비해 속도가 느린 경우 이 두 공간을 분리하면 속도와 공간의 효율성이 있다. 본 발명의 실시예들에서는 IP block을 연결할 때 포트 맵 입출력 방식을 사용하여 속도와 공간의 효율성을 향상시켰다.

[0071] 하드웨어와 소프트웨어간의 통신 시 신뢰성을 높이기 위해 8051 마이크로프로세서와 하드웨어 코-프로세서 간에 개선된 핸드쉐이크(handshake) 프로토콜(22)을 사용하였다. 서로 통신이 확립되면 완료가 되었다는 완료 시그널(done signal)을 보내어 데이터를 주고 받을 수 있는 상태가 되었는지 확인한다. 도 6의 실시예에서는 암호화와 메시지를 주고 받는 부분이 다른 모듈로 설계되어 있기 때문에 동시에 병렬적으로 수행하도록 핸드쉐이크 프로토콜(22)을 개선하였다. 따라서 하드웨어 코-프로세서(20)에서 암호화 수행 중에도 8051 마이크로프로세서(10)와 메시지를 주고 받을 수 있게 하였다.

[0072] 상기된 본 발명의 실시예들에 따르면, 본 발명은 경량암호가 사용될 수 있는 IoT 제품이나 RFID/USN 및 스마트카드 등에 탑재되어 효율적인 암호화 연산에 사용될 수 있으며, 또한, 기존에 탑재되어 있는 HIGHT 경량암호 기반의 전자 제품에 대체하여 탑재할 수 있으며, 다양한 운용모드에 바로 적용할 수 있어 제품 및 플랫폼에 상관없이 적용할 수 있다는 장점을 갖는다.

[0073] 이상에서 본 발명에 대하여 그 다양한 실시예들을 중심으로 살펴보았다. 본 발명에 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

부호의 설명

[0074] 10 : 하드웨어 처리기

20 : GEZEL 코-프로세서

21 : GEZEL IP 블록

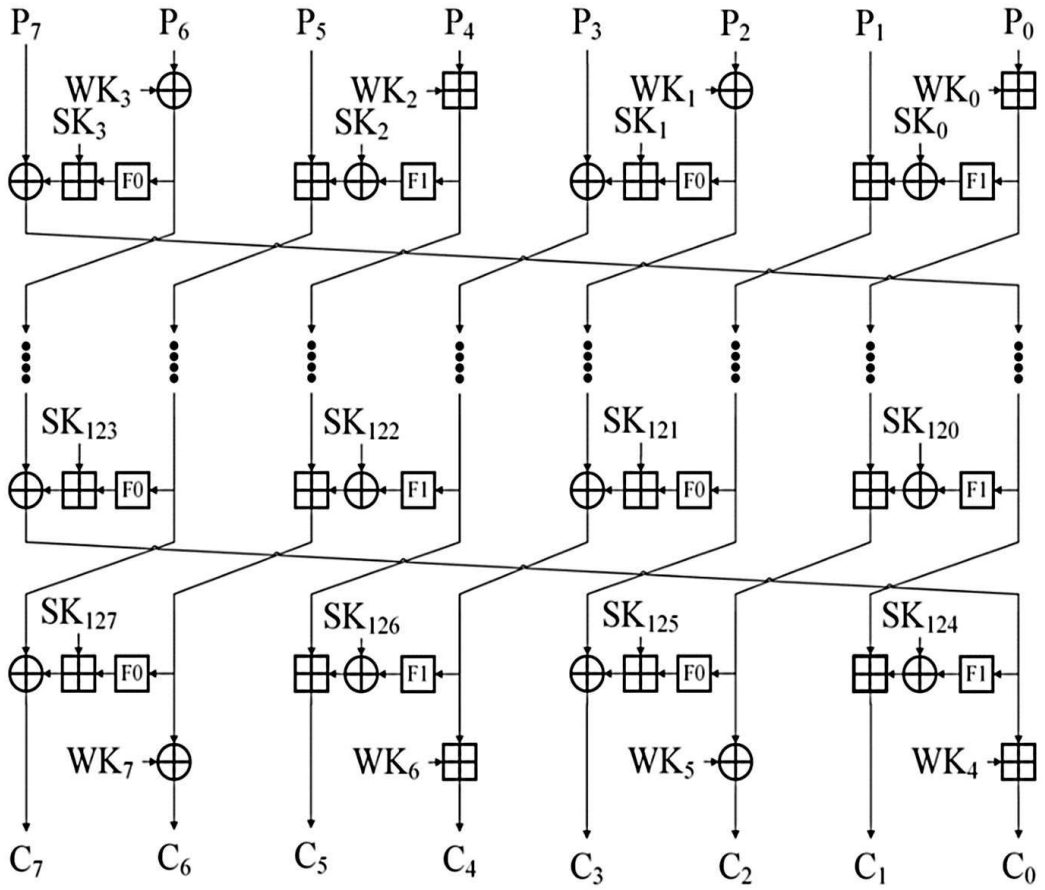
22 : 핸드쉐이크 프로토콜

23 : HIGHT 유한상태기계 제어기

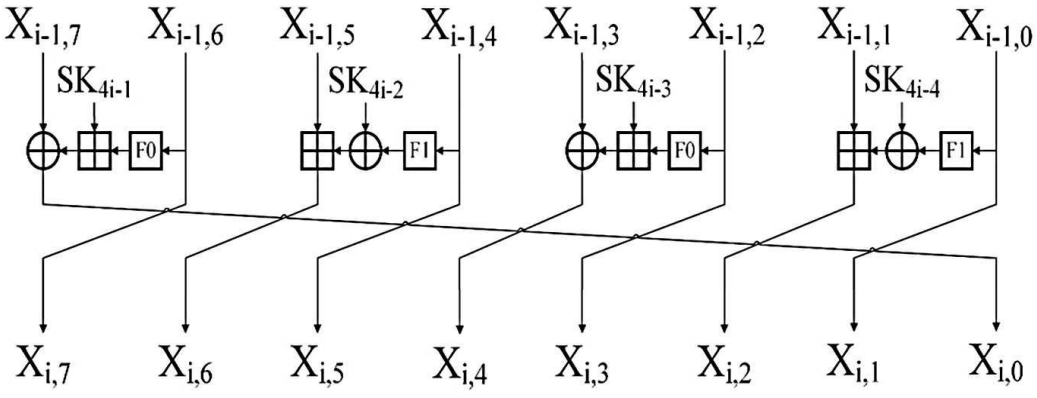
24 : HIGHT 데이터패스

도면

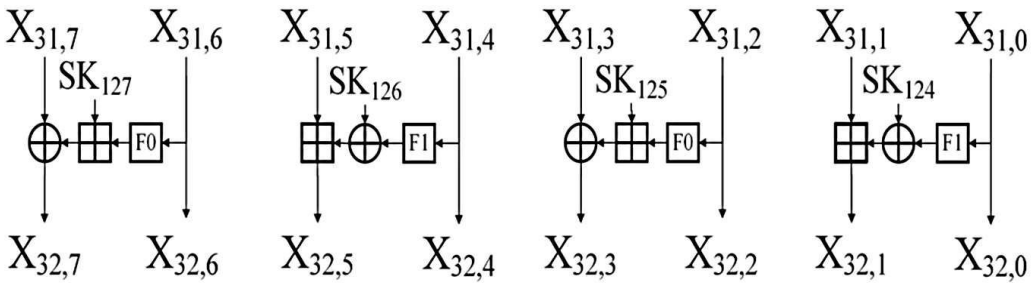
도면1


P7
P6
P5
P4
P3
P2
P1
P0
WK3
SK3
WK2
SK2
WK1
SK1
WK0
SK0
F0
F1
SK123
SK122
SK121
SK120
SK127
SK126
SK125
SK124
WK7
WK6
WK5
WK4
C7
C6
C5
C4
C3
C2
C1
C0

도면2


Xi-1,7
Xi-1,6
Xi-1,5
Xi-1,4
Xi-1,3
Xi-1,2
Xi-1,1
Xi-1,0
SK4i-1
SK4i-2
SK4i-3
SK4i-4
F0
F1
Xi,7
Xi,6
Xi,5
Xi,4
Xi,3
Xi,2
Xi,1
Xi,0

도면3

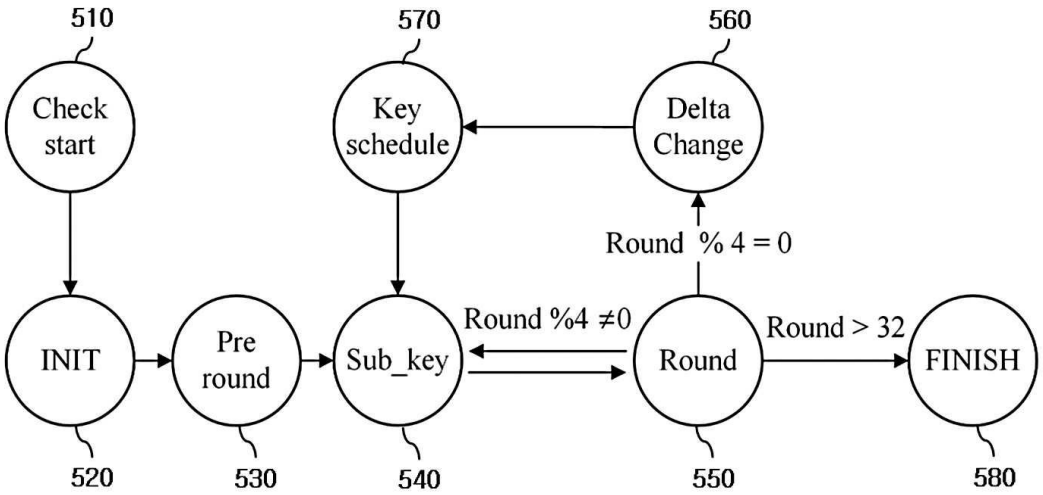


도면4

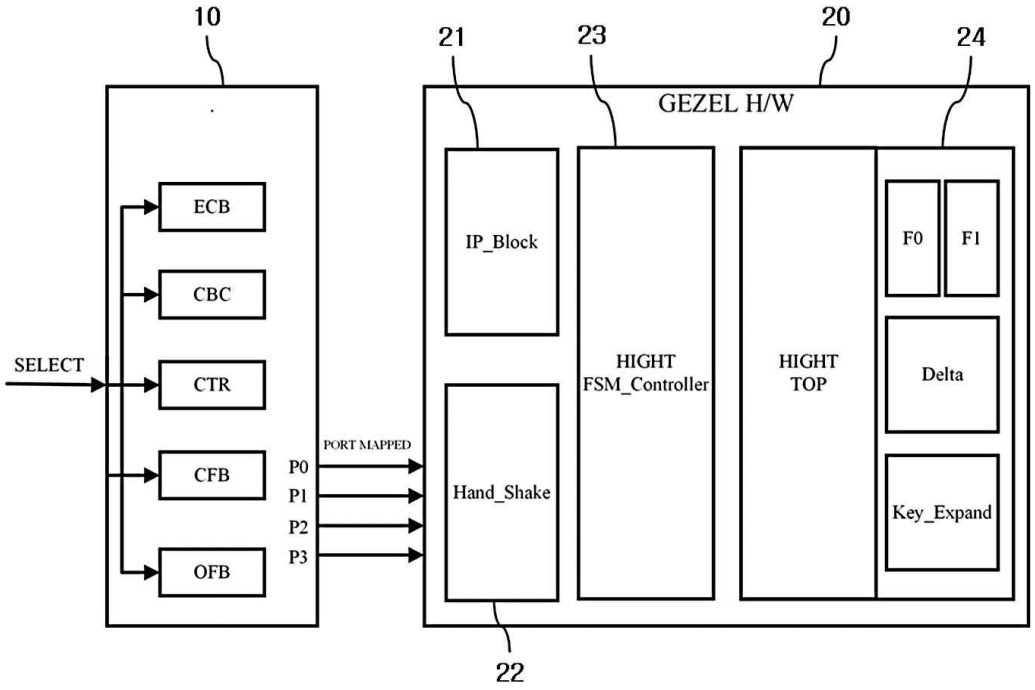
WhiteningKeyGeneration(K, WK)
for i = 0 to 7
  if $0 \le i \le 3$, then $WK_i = K_{i+12}$
  else, $WK_i = K_{i-4}$

SubkeyGeneration(K, SK)
for i = 0 to 7
  for j = 0 to 7
    $SK_{(16 \cdot i + j)} \leftarrow K_{(j - i \bmod 8)} \boxplus delta_{(16 \cdot i + j)}$
    $SK_{(16 \cdot i + j + 8)} \leftarrow K_{(j - i \bmod 8) + 8} \boxplus delta_{(16 \cdot i + j + 8)}$

도면5



도면6


10
21
23
20
24
GEZEL H/W
ECB
CBC
CTR
CFB
OFB
SELECT
PORT MAPPED
P0
P1
P2
P3
IP_Block
Hand_Shake
HIGHT
FSM_Controller
HIGHT
TOP
F0
F1
Delta
Key_Expand
22