32-bit RISC-V 프로세서 상에서의 경량 블록 암호 SPECK 카운터 운용 모드 최적 구현

한성대학교

심민주, 이민우, 송민호, 서화정

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

관련 연구

구현 기법

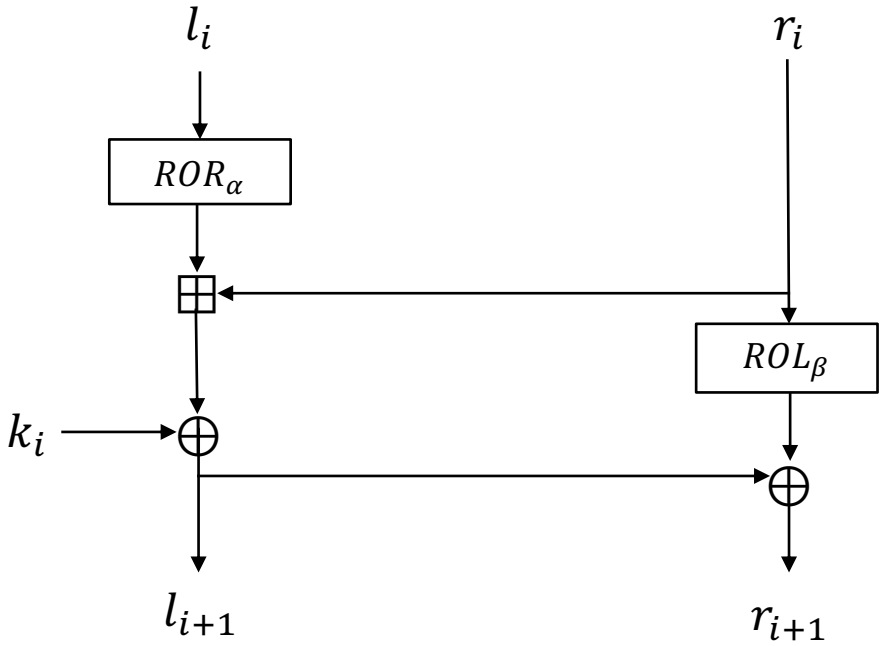
성능 평가

결론

경량 블록 암호 SPECK

- 2013년 미국 국가안보국에서 개발한 Feistel 구조의 경량블록 암호
- ARX(Addition, Rotation, XOR) 구조

Block Size	Key Size	Rounds	Word Size	Rotation α	Rotation β
32	64	22	16	7	2
48	72	22	24	8	3
	96	23			
64	96	26	32		
	128	27			
96	96	28	48		
	144	29			
128	128	32	64		
	192	33			
	256	34			



RISC-V Processor

- 오픈 소스로 제공되는 명령어 셋을 기반으로 한 프로세서

- 32-bit 프로세서인 RV32I에서는 32개의 32-bit 레지스터 제공

X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
ZERO	RA	SP	GP	TP	T0	T1	T2	S0	S1	A0	A1	A2	A3	A4	A5
X16	X17	X18	X19	X20	X21	X22	X23	X24	X25	X26	X27	X28	X29	X30	X31
A6	A7	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	T3	T4	T5	T6

RA : Return Address Register

SP : Stack Pointer Register

GP : Global Pointer Register

TP : Tread Pointer Register

T0 ~ T6 : temporal registers

S0 ~ S11 : saved registers(callee saved)

A0 ~ A7 : function arguments and return value

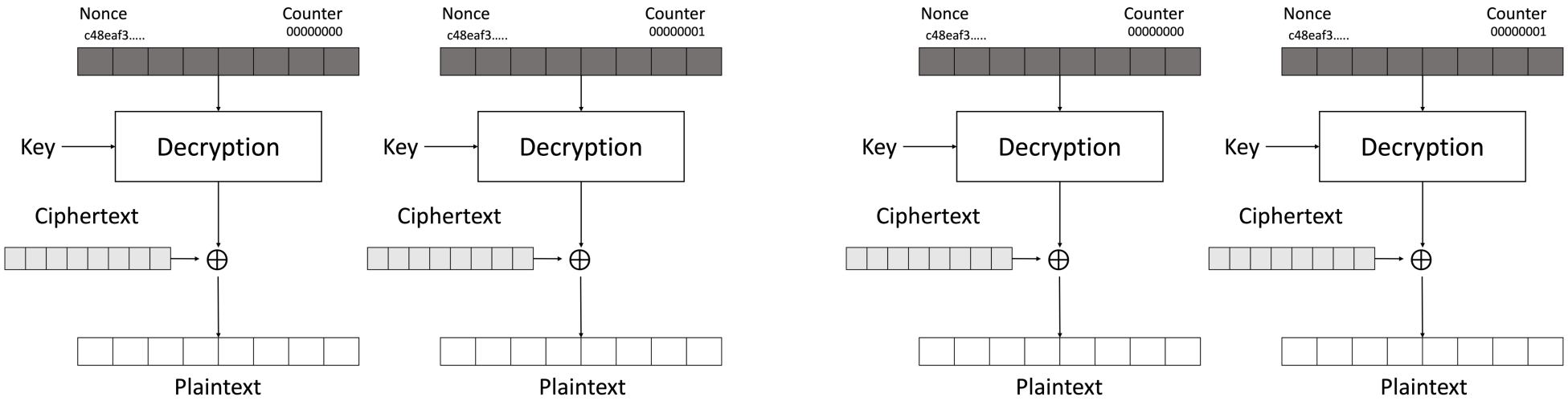
카운터 운용 모드

- 입력 : nonce 값 + Counter 값

  (nonce : 고정된 임의의 상수 / Counter :블록을 암호화 할 때마다 1씩 증가)

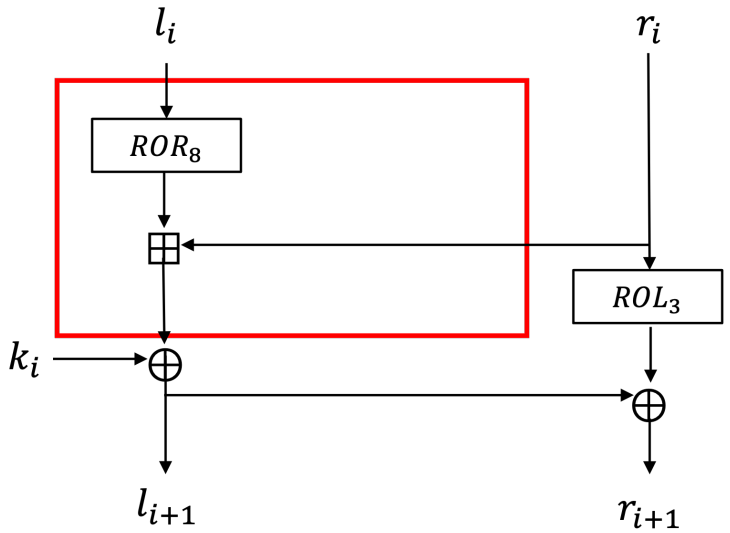
- 이렇게 결합된 입력 값을 암호화하여 키 스트림 생성

- 암호문 : 생성된 키 스트림 ⊕ 평문



구현 기법

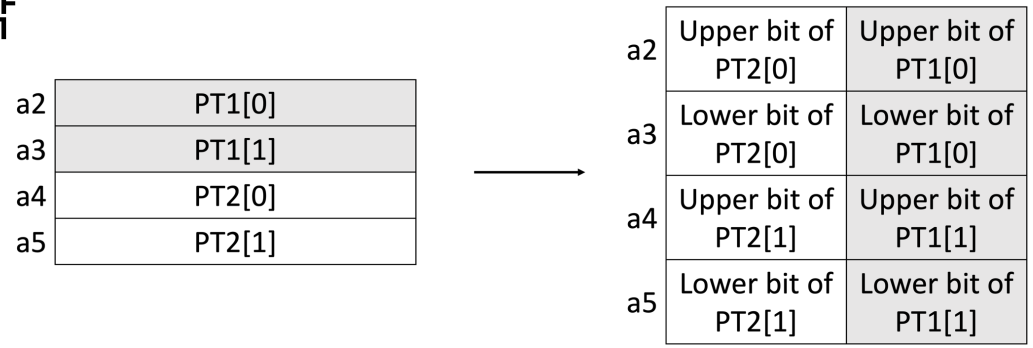
- 고정된 nonce 블록이 카운터 블록에 영향을 받기 전까지 해당되는 연산에 대해 사전 연산 가능
- 카운터 블록에 영향을 받기 전 **1라운드 일부만 사전 연산 가능**
- 왼쪽 블록: nonce, 오른쪽 블록 : 카운터
  - 일반적인 카운터 값은 32-bit
  - SPECK-64/128의 경우, 카운터 값으로 32-bit 사용



SPECK-CTR 구현 기법

- 단일 평문 구현의 사전 연산을 통한 생략
  - 2번의 쉬프트 연산, 1번의 ADD 연산 생략

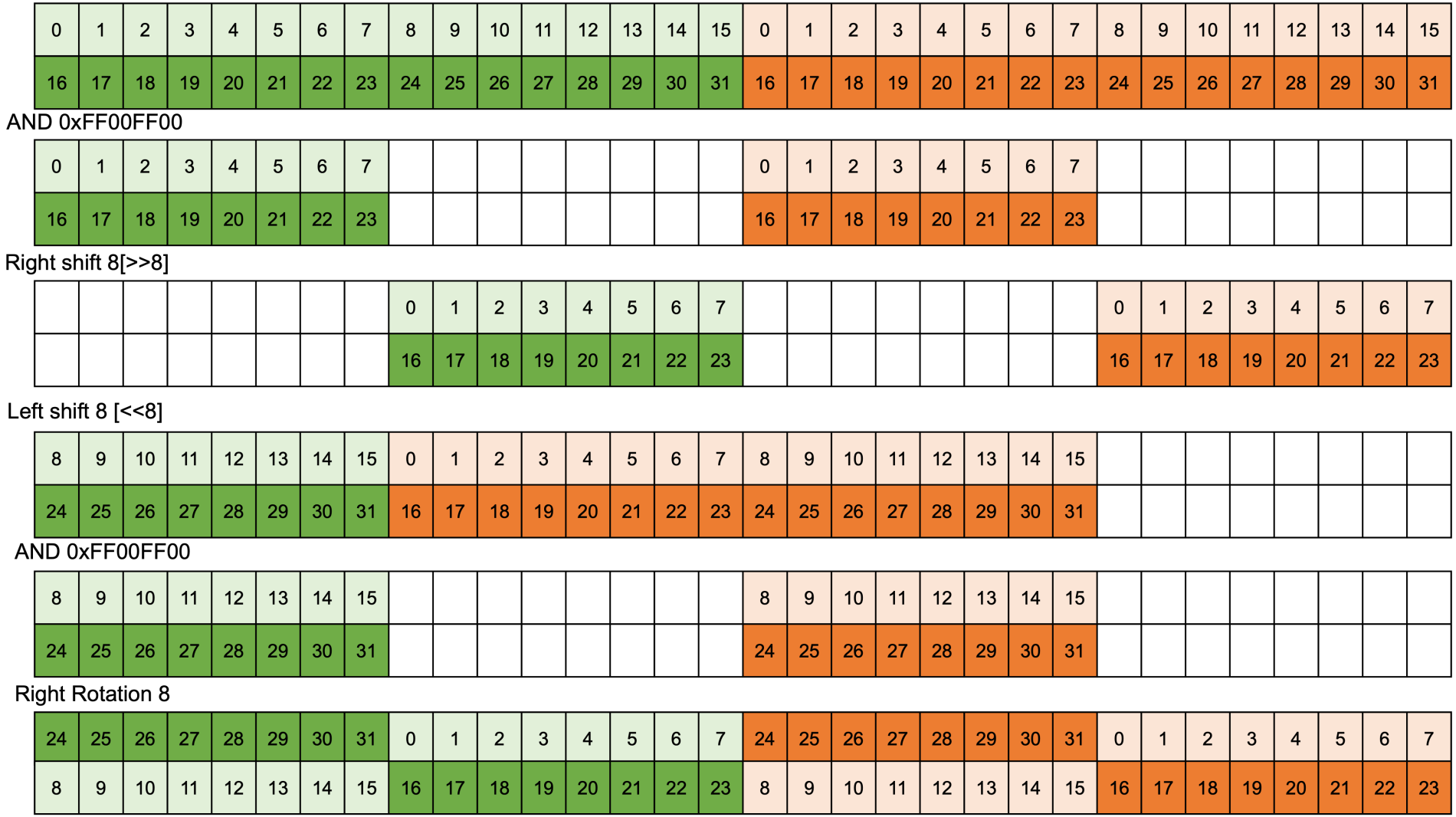
- 2개의 평문 병렬 구현의 사전 연산을 통한 생략
  - 암호화 시작 전, 레지스터 내부 정렬 진행
    - 서로 다른 두 개의 평문을 통해 연속 정렬



  - 6번의 XOR 연산, 6번의 쉬프트 연산, 15번의 AND 연산, 5번의 ADD 연산, 10번의 MV 연산, 1번의 STLU 연산 생략

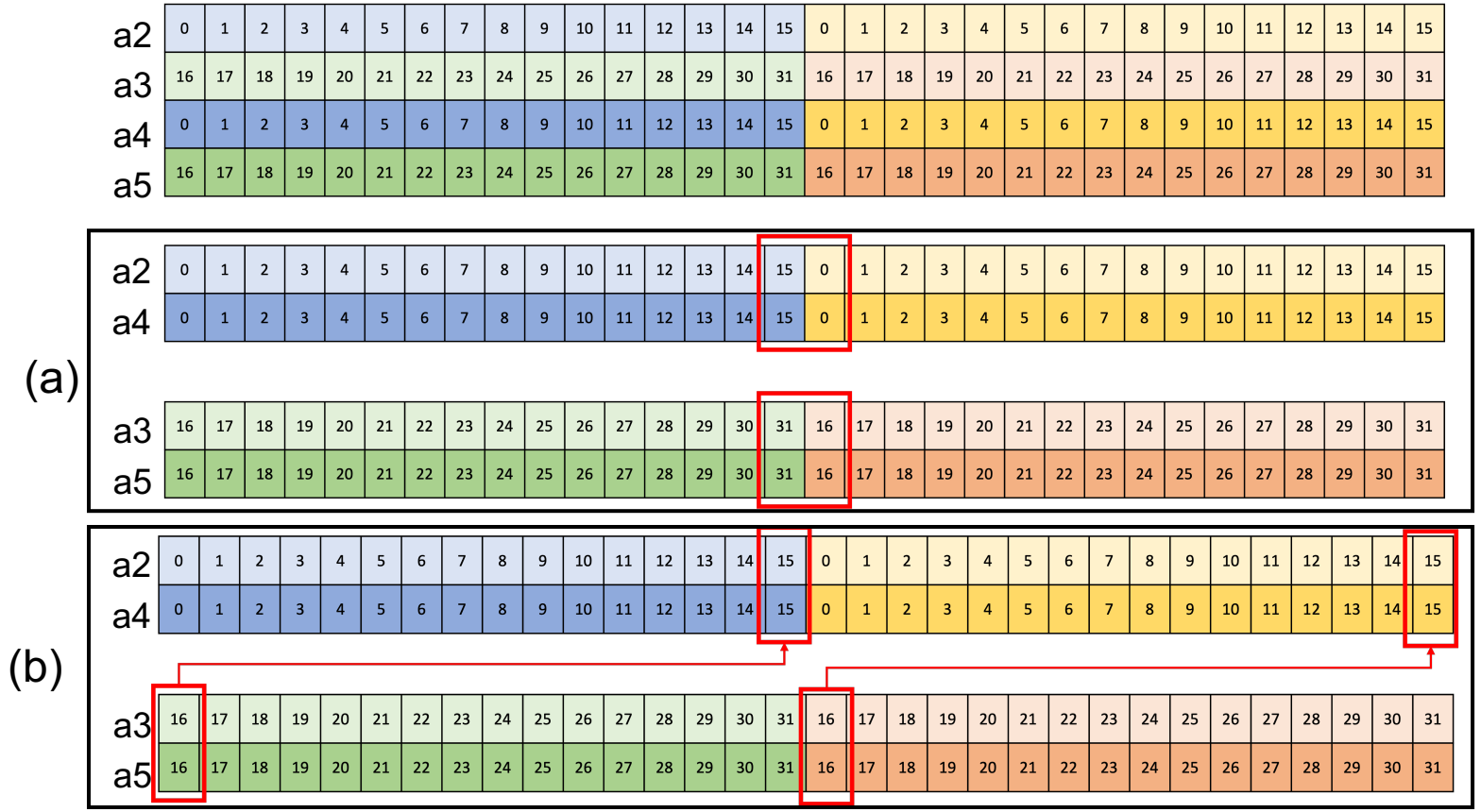
SPECK-CTR 구현 기법

- 로테이션 구현



SPECK-CTR 구현 기법

- Addition 구현



성능 평가

- 레퍼런스 대비
  - 단일 평문 최적 구현 (**Our work***) 의 경우, **5.76배** 성능 향상
  - 2개의 평문 최적 구현 (**Our work***) 의 경우, **2.24배** 성능 향상

- SPECK 최적화 구현(Our work)대비 사전 연산 기법 적용한 구현 (**Our work***)

1% 성능 향상

SPECK-64/128	Clock Cycles	
Ref-c	1,873	
1-pt	Our work	327
	Our work*	**325**
2-pt	Our work	1,768
	Our work*	**1,670**

결론

- RISC-V 상에서의 SPECK-CTR 구현
- 고정된 nonce 값을 활용한 사전 연산을 통한 연산 생략 구현
- SPECK-64/128에 대한 단일 평문과 2개의 평문 병렬 구현
  - 각각 알고리즘에 적합한 사전 연산을 통한 연산 생략
  - 2개의 SPECK-CTR에 대해 레퍼런스 대비 5.76배, 2.24배 성능 향상 확인
  - 2개의 SPECK-CTR에 대해 SPECK 최적 구현 대비 1% 성능 향상 확인

Q & A