저사양 프로세서 상에서의 경량 블록암호 SIMECK 최적 구현 동향

한성대학교

심민주, 이민우, 김동현, 윤세영, 서화정

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

SIMECK

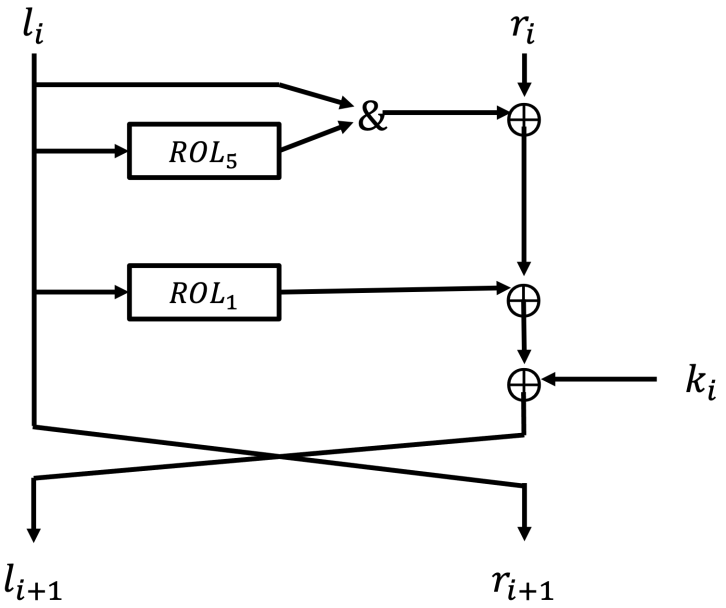
저사양 프로세서 상에서의 SIMECK 최적 구현 동향

결론

SIMECK

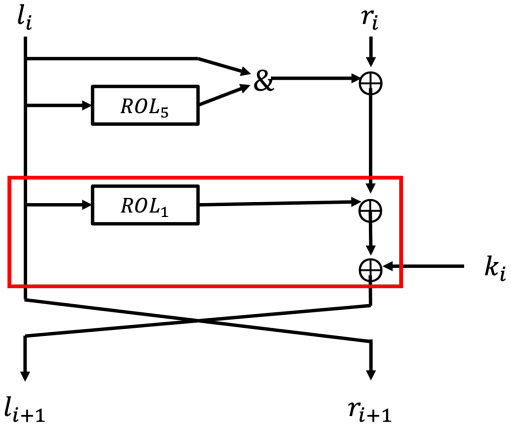
- CHES'15에서 발표된 Feistel 구조의 경량 블록암호
- NSA에서 개발한 경량 블록 암호인 SIMON과 SPECK의 장점을 결합하여 개발
  - SIMON의 라운드 함수를 일부 변형
  - SPECK의 키 스케줄링과 유사한 구조

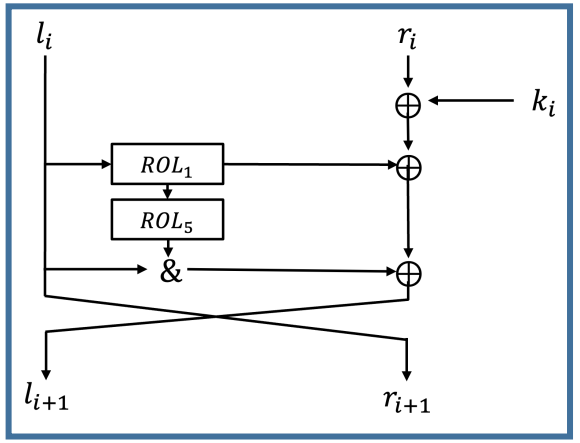
Cipher	n	k	r	w
SIMECK-32/64	32	64	32	16
SIMECK-48/96	48	96	36	24
SIMECK-64/128	64	128	44	32



저사양 프로세서 상에서의 SIMECK 최적 구현 동향

- 8-bit AVR 상에서의 최적 구현
  - 효율적인 로테이션 연산 구현
    - 데이터 블록을 나누어 8비트 레지스터에 위치
    - 쉬프트 오프셋과 방향 알고리즘을 통한 로테이션 최적 구현
      - 오프셋 값(o)을 4와 비교하여 4보다 큰 경우, 오프셋 값을 o-4로 설정 & 방향 변경
  - XOR 사전 연산을 통한 최적 구현
    - 라운드 함수의 일부 연산 순서 변경
  - 코드 최적화
    - Loop unrolling 기법을 통해 for문의 loop 비용 줄임





  - 성능 측정 결과
    - 최적화된 On-the-Fly 구현 : 평균 14.42배 빠르고, 1.53배 적은 메모리 사용
    - 속도와 메모리에 최적화된 구현 : 평균 12.98배 빠르고, 1.3배 적은 메모리 사용

저사양 프로세서 상에서의 SIMECK 최적 구현 동향

- 16-bit MSP430 상에서의 최적 구현
  - SIMECK-48/96을 구현하기 위해 16비트 + 8비트 조합으로 분리하여 구현
    - MSP430에는 8비트 크기의 데이터 연산에 대한 명령어도 지원
  - 코드 사이즈 최적화
    - loop-rolling & index 사용
    - 범용 레지스터에 데이터 블록을 저장, Register mode addressing을 사용하여 데이터 재사용 하는 기법 사용
      - 레퍼런스 코드는 데이터 블록을 로드할 때마다 index 값 연산
    - SIMECK-64/128의 경우, 8-word 코드 사이즈 줄임
  - 속도 최적화
    - loop-unrolling & register addressing mode 사용하여 Index addressing mode의 시간을 줄임.
    - 이 기법을 통해, 14 clock-cycle 절약
  - 성능 평가
      - 코드 사이즈 최적화 On-the-Fly는11.62% , 속도 최적화 On-the-Fly는 2.18배 의 속도 향상 보임

저사양 프로세서 상에서의 SIMECK 최적 구현 동향

- 32-bit ARM 상에서의 최적 구현
  - NEON 최적화
    - 데이터 블록을 효율적으로 로드 & 저장
    - 왼쪽 쉬프트와 오른쪽 쉬프트 활용하여 로테이션 연산 최적화
      - 오른족 쉬프트 연산 후, 내장 함수(vsriq_n_u16, vsriq_n_u32)를 사용하여 코드 사이즈 줄임.
  - 속도 최적화
    - NEON 파이프라인을 고려하여 NEON 레지스터 동작 순서 재설계
    - omp_set_num_threads(#) OpenMP 함수를 사용하여 thread 설정
      - # : thread 수
  - 성능 평가
    - SIMECK-32/64는 41.4%, SIMECK-64/128은 46.9% 성능 향상 보임

저사양 프로세서 상에서의 SIMECK 최적 구현 동향

- 32-bit RISC-V 상에서의 최적 구현
  - 단일 평문 최적 구현
    - 로테이션 연산 : 쉬프트 연산과 xor연산을 활용하여 구현
  - 2개의 평문 최적 병렬 구현
    - 암호화 시작 전, 하나의 레지스터에 서로 다른 평문 위치하도록 레지스터 정렬
    - 하나의 레지스터에 위치한 섞이지 않아야 하는 두개의 평문이 섞이지 않도록 로테이션 연산 구현
  - 라운드 함수의 특징을 활용한 블록의 이동 생략
  - 성능 평가
    - SIMECK-32/64 : 단일 평문의 경우 503% , 2개의 평문 병렬의 경우 1,038% 성능 향상
    - SIMECK-64/128 : 단일 평문의 경우 446% , 2개의 평문 병렬의 경우 350% 성능 향상

결론

- 저사양 프로세서 상에서의 경량 블록암호 SIMECK 최적 구현 동향에 대해 살펴봄.
- 다양한 환경에서 적합한 SIMECK 최적 구현 연구가 활발히 진행됨을 확인
- 향후 연구
  - 저사양 프로세서 상에서의 SIMECK-CTR 최적 구현 연구 제안

Q & A