LowMC 구현

장경배

https://youtu.be/WJlcFbeMSj8

LowMC's Motivation

- **암호학의 발전으로 인한 차세대 암호 기술의 등장**
  - Zero-Knowledge(ZK), fully homomorphic encryption(FHE), Multi Parity Computation (MPC)

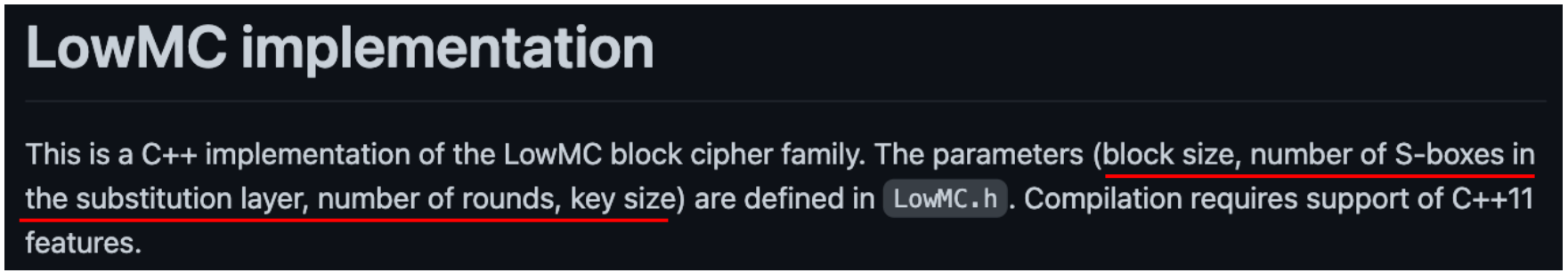
- **암호 알고리즘을 설계하는 것은 선형, 비선형 연산의 섬세한 균형으로 이루어짐**
  - 선형, 비선형 연산을 하드웨어, 소프트웨어로 구현하는 비용은 거의 비슷함

- **그러나 ZK, FHE, MPC의 경우에는 그렇지 않음**
  - **선형 연산은 local computation**만을 발생시키기 때문에 **거의 비용이 들지 않음**
    - Noise가 많이 없음
  - 대칭 암호화와 당사자 간의 통신에서 발생하는 **비선형 연산이 대부분의 비용을 차지**
    - 상당한 Noise

LowMC

- **LowMC는 ZK, FHE, MPC 체계에서의 암호 구현을 타겟으로 함**
  - **비선형 연산을 최소화**하고, **대부분의 암호화 작업을 선형 연산으로 설계**
  - **AND 게이트 최소화**

- **파라미터화 가능한 LowMC**
  - 블록 크기, 라운드에서 사용 할 Sbox의 개수, 라운드 수



  - Data 복잡도, 시간 복잡도에 대한 보안성을 구체적으로 인스턴스화 할 수 있음
    - LowMC의 인스턴스를 생성하는 과정에서 **Linear layer가 랜덤하게 생성됨**

LowMC Round Function

- **LowMC 라운드 함수**
  - SboxLayer → LinearLayer → ConstantAddition → KeyAddition

$$\textsc{LowMCRound}(i) = \textsc{KeyAddition}(i) \circ \textsc{ConstantAddition}(i) \circ \textsc{LinearLayer}(i) \circ \textsc{SboxLayer}$$

In the following we give a detailed description of the individual steps.

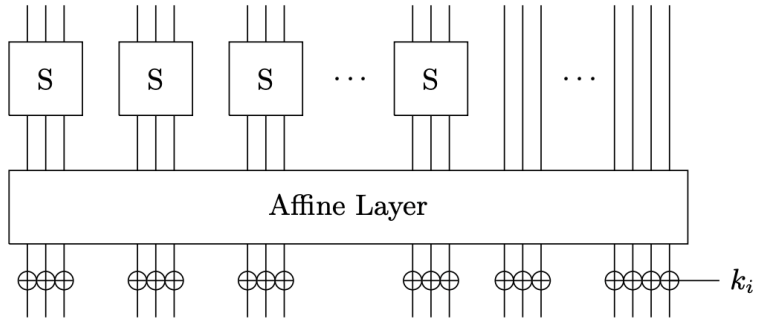
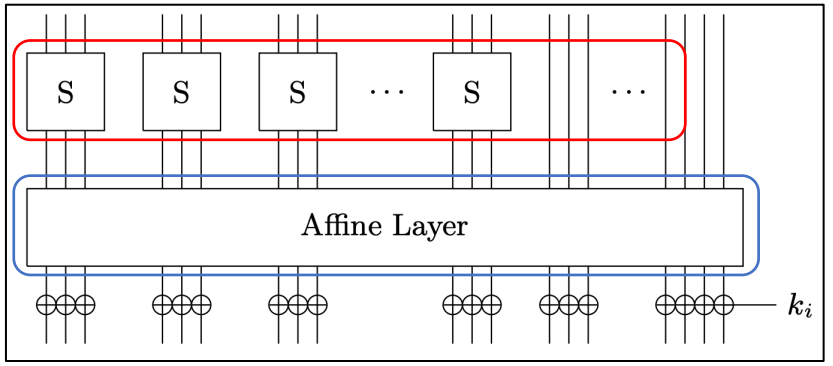


Fig. 1. Depiction of one round of encryption with LowMC.

LowMC Round Function

- **LowMC 3-bit Sbox**

$$S(a, b, c) = (a \oplus bc, a \oplus b \oplus ac, a \oplus b \oplus c \oplus ab)$$



- **LinearLayer(Affine Layer)**
  - 인스턴스 생성 시, 라운드 마다 사용할 행렬이 생성됨
  - **n-bit 블록과 (n x n)행렬의 곱**

- **ConstantAddition, KeyAddition**
  - 라운드 상수도 인스턴스 생성 시, 생성됨
  - 인스턴스 생성 시, 라운드 키 생성을 위한 행렬이 생성됨
  - **키 스케줄도 LinearLayer와 동일하게, n-bit key와 (n x n)행렬의 곱**

LowMC in EUROCRYPT

- **LowMC in PICNIC**

> We focus on the use of LowMC in the post-quantum digital signature scheme Picnic [CDG+17a, CDG+17b] which is based on zero-knowledge proofs of knowledge of pre-images of one-way functions.

- **EUROCRYPT 20에서는 PICNIC에서 사용되는 LowMC 인스턴스를 구현**

Implementing Grover oracles for quantum key search on AES and LowMC

Samuel Jaques[1*†], Michael Naehrig[2], Martin Roetteler[3], and Fernando Virdia[4†‡]

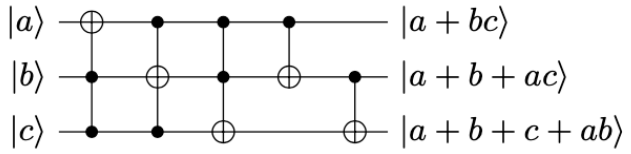
[1] Department of Materials, University of Oxford, UK
samuel.jaques@materials.ox.ac.uk
[2] Microsoft Research, Redmond, WA, USA
mnaehrig@microsoft.com
[3] Microsoft Quantum, Redmond, WA, USA
martinro@microsoft.com
[4] Information Security Group, Royal Holloway, University of London, UK
fernando.virdia.2016@rhul.ac.uk

- **LowMC L1, L3, L5**
  - **128-bit**, **192-bit**, **256-bit** Key & Block
  - **20 Round**, **30 Round**, **38 Round**
  - **10개의 3-bit Sbox 공통**

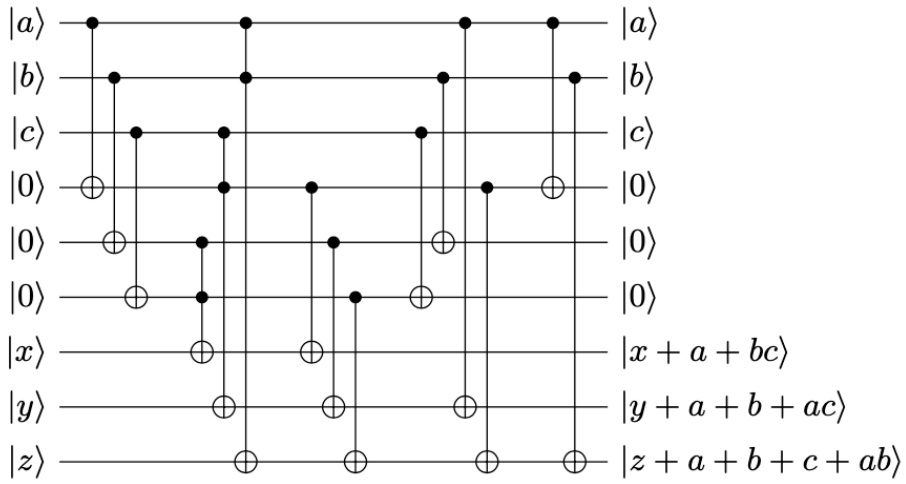
LowMC in Quantum

- LowMC는 비선형 연산보다는 **선형 연산의 비중이 더 큼**
  - 양자 회로 구현 시, Sbox의 비중이 적고 LinearLayer & Keyschedule이 대부분을 차지

- EUROCRPYT에서는 다음 두가지 Sbox 중 (b)를 사용
  - Qubit을 더 많이 사용하지만 **Toffoli depth가 1, (a)는 3**
  - 본 구현에서는 2가지 모두 사용 및 비교, (a)는 regular 버전, (b)는 shallow 버전



(a) LowMC in-place S-box.



(b) LowMC T-depth 1 S-box.

LowMC in Quantum

- EUROCRYPT에서 LinearLayer(행렬 곱)는 **PLU 분해를 사용하여 in-place 구현**
  - 본 구현에서는 일반적인 행렬 곱 구현

일반적인 행렬 곱

PLU 분해를 사용한 구현

- 자원 비교

Method	#CNOT	#1qCliff	#qubits	Full depth
Linear layer L1 [JNRV20]	8,093	60	128	2,365
Linear layer L3 [JNRV20]	18,080	90	192	5,301
Linear layer L5 [JNRV20]	32,714	137	256	8,603
Linear layer L1 (this work)	8,205	0	256	225
Linear layer L3 (this work)	18,418	0	384	339
Linear layer L5 (this work)	32,793	0	512	455

LowMC in Quantum

- 키 스케줄은 같은 행렬 곱이지만 LinearLayer와 조금 다름
  - 입력 키와 계속 곱함

$$x_i = XM_i \cdot x_{i-1}$$

< LinearLayer >

$$rk_i = KM_i \cdot k$$

< Keyschedule >

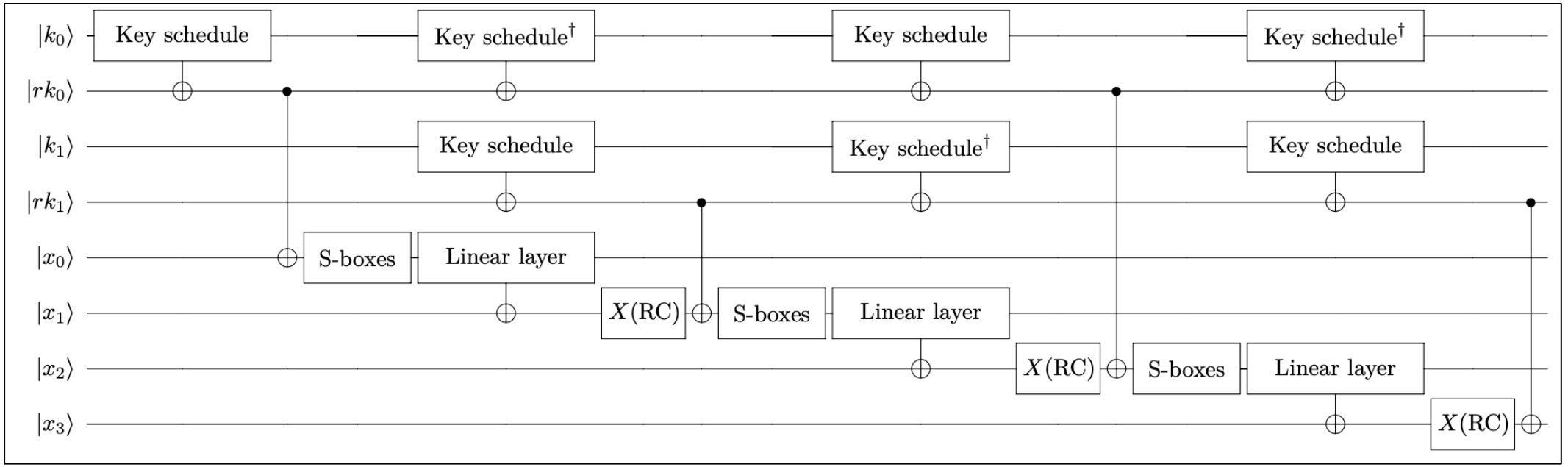
- EUROCRYPT 구현

$rk_i = KM_i \cdot KM_{i-1}^{-1} \cdot rk_{i-1}$ 를 PLU 분해

- 본 구현에서는 라운드 키 사용 후, reverse 연산으로 초기화
  - 키 스케줄은 LinearLayer와 달리 Qubit 절약 가능

LowMC in Quantum

- **Reverse 연산으로 인해 Depth가 증가하는 문제**
  - 입력 키를 복사(k_1)하여 2쌍을 교체하면서 사용 → 가능한 모든 연산들이 병렬로 동작



< LowMC 양자 회로 구조 >

LowMC in Quantum

* T-depth 차이는 Toffoli 게이트 분해 방식의 차이

/ 2

operation	#CNOT	#1qCliff	#T	#M	T-depth	full depth	width
LowMC L1	689944	4932	8400	0	40	98699	991
LowMC L3	2271870	9398	12600	0	60	319317	1483
LowMC L5	5070324	14274	15960	0	76	693471	1915

<EUROCRPYT>

Method	#CNOT	#1qCliff	#T	T depth	#qubits	Full depth
LowMC L1⌑	498,208	2,466	4,200	240	3,200	4,708
LowMC L1⋈	500,208	2,466	4,200	80	3,830	4,708
LowMC L3⌑	1,669,456	4,699	6,300	360	6,720	10,571
LowMC L3⋈	1,672,456	4,699	6,300	120	7,650	10,571
LowMC L5⌑	3,754,484	7,137	7,980	456	11,008	17,789
LowMC L5⋈	3,758,284	7,137	7,980	152	12,178	17,789

⌑: Regular version. ⋈: Shallow version.

<이번 구현>

감사합니다