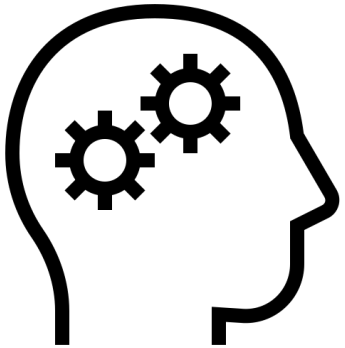
# 머신러닝과 암호 설계 아이디어

https://youtu.be/EqFRb6LXUjk

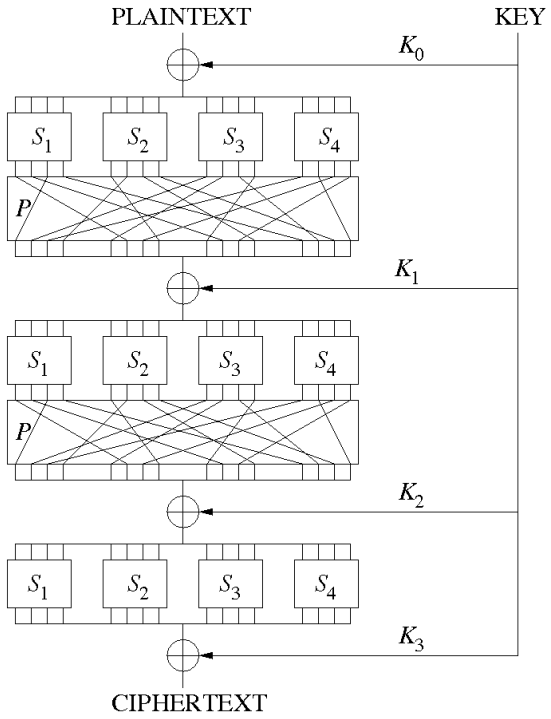
# 암호 설계 AI

- 신경 아키텍처 검색 (Searching for Neural Architectures, NAS):
  인공 신경망 (ANN) 의 설계를 자동화
  사람이 설계한 아키텍처와 동등하거나 성능이 우수한 네트워크를 설계



  - 강화학습
  - 진화알고리즘

- **암호 설계에 이러한 기법 적용**




# 관련 연구

- Modeling Power Efficiency of S-boxes Using Machine Learning(2019)
  - S-box가 전력 효율이 좋은지 아닌지를 예측하는 최초의 기계 학습 기반 접근 방식

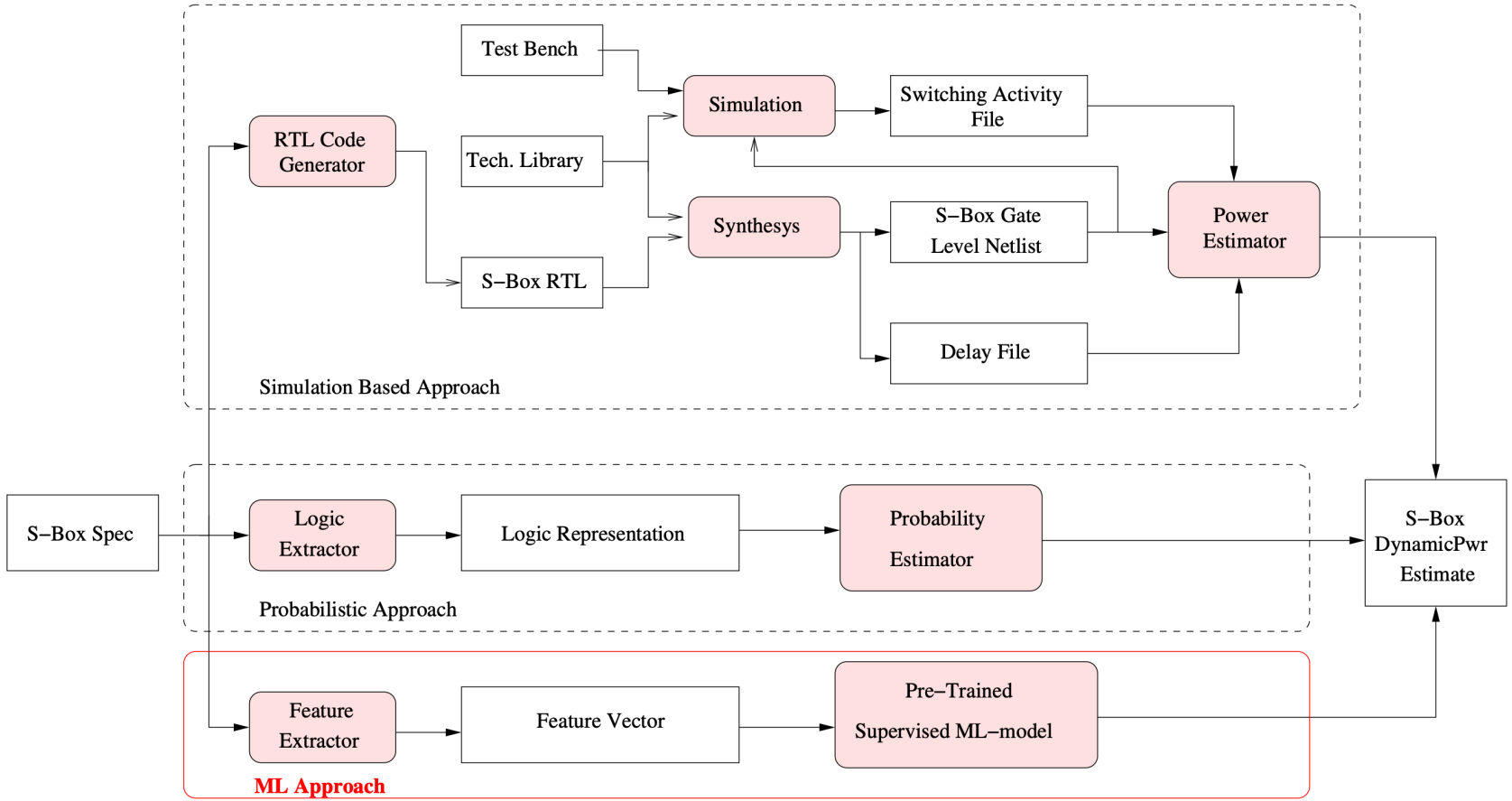
- Generating Cryptographic S-Boxes Using the Reinforcement Learning(2021)
  - 강화 학습을 통해 부채널 공격에 대한 대응책인 마스킹 기법을 효율적으로 적용할 수 있는 S-box를 생성

# Modeling Power Efficiency of S-boxes Using Machine Learning

- 암호학적으로 우수하고 전력 효율적인 4x4 S-box를 설계
- 상용 CAD 도구를 사용하여 전력 소비를 결정하는 기존의 접근 방식은 시간이 많이 소요
- Sbox의 부울 함수 표현에서 전력 효율성을 신속하게 특성화해야 하는 자동화의 개발
  - 기존 접근 방식보다 14배

# S-box의 전력 효율성 확인

- 시뮬레이션 기반, 확률론적 접근



# 학습

- S-box의 AOI 구현(LUT 기반)을 위한 동적 전력 소비

- 기계 학습 모델 지도 학습 접근
  - S-box의 정확한 동적 전력을 보고 X
  - 전력 효율인지 아닌지를 보고
  - 회귀가 아닌 분류에 문제를 매핑

- 사전 정의된 임계값 전력 Pth가 주어지고 나쁨 0, 양호 1 로 분류

# 학습

- 유전자 알고리듬을 사용하여
  4의 미분 균일성과 비선형성을 갖는
  암호화적으로 강력한 S-박스 10000개의 목록으로 시작

- 130μW 전력 값을 임계값으로 선택
- 데이터 세트 S-box
  동적 전력 값이 130μW 이상(좋은 S-box) 5000개
  동적 전력 값이 130μW 미만 (나쁜 S-box) 5000개

# Generating Cryptographic S-Boxes Using the Reinforcement Learning

- 부채널 공격(전자파 등으로 키값 탐지) 대응을 잘 하면서도 안정적인 S-Box(암호학에서 필수적인 구성요소)를 만드는 것이 중요함.

- S-Box는 Algebric Normal Form으로 표현가능해 AND, XOR 등을 중첩해 생성 가능한데, 명확한 리워드도 잇고 MDP를 정의하기 용이해 강화학습으로 접근

- 기존에 존재하지 않던 구조의 S-Box를 생성했고 몇 가지 S-Box는 기존 휴리스틱으로 만든 S-Box보다 더 좋은 성능을 냄

# 강화 학습

- S-box 생성 방법은 현재 S-box의 비트 슬라이스 구현에 비트 연산을 스택하는 작업을 반복하여 다음 S-box의 비트 슬라이스 구현을 유도
  - S-box와 비트 슬라이스 구현이 동시에 생성
- RL 문제는 상태, 동작, 상태 전환 및 보상의 네 가지 핵심 요소
- AND-XOR 또는 XOR 연산을 IS에 누적하여 S-box를 업데이트하고 생성합니다. RL 에이전트가 AND-XOR 연산을 사용하는 경우 환경은 S-box의 암호화 속성을 확인하고 환경의 보상 설정에 따라 RL 에이전트에게 보상을 제공

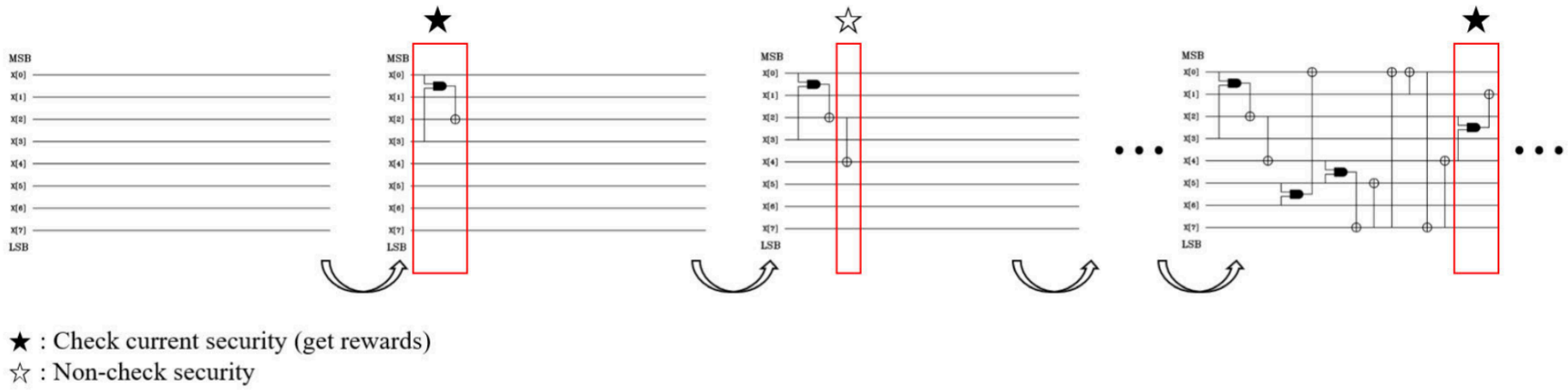



**FIGURE 2.** S-box generation process using RL.

# 강화 학습

- STATES

  S-box DDT(미분분포표) 및 LAT(선형근사표), S-box의 대상 암호 속성 및 현재 암호 속성과 같은 관찰

- THE SET OF ACTION
  - AND-XOR 또는 XOR을 선택하고 적용할 비트 위치를 선택
  - n비트에 AND-XOR을 적용하기 위해 3비트를 선택하는 경우의 수는 n×(n-1)×(n-2)
  - AND-XOR을 적용할 비트를 바꿀 수 있지만 결과는 동일 하므로 따라서 경우의 수를 2로 나눔
  - n비트에 XOR을 적용하기 위해 2비트를 선택하는 경우의 수는 n × (n - 1)
  - 이러한 이유로 RL 에이전트는 각 단계에서 (n × (n − 1) × (n − 2))/ 2 + n × (n − 1) 동작 중 하나를 선택

- STATE TRANSITION PROBABILITY
  - 현재 상태에 조치가 적용될 때 다음 상태로 업데이트될 확률
  - 결정론적으로 생성되기 때문에 현재 상태에서 다음 상태로 전환될 확률은 1

# 보상

- 암호화 속성을 만족하고 비선형 연산의 수를 최소화하기 위해 다양한 보상 형태를 실험
- AND-XOR의 수를 최소화

$$\textbf{reward}_1 = \begin{cases} \alpha & \text{When using AND-XOR and } \alpha \text{ is a positive number or zero} \\ \alpha - 1 & \text{When using AND-XOR and } \alpha \text{ is a negative number} \end{cases}$$

**termination condition$_1$**: $N$ steps (We set $N = 500$.)

$$\textbf{reward}_2 = \begin{cases} -0.001 & \text{When using XOR} \\ -1.001 & \text{When using AND-XOR} \\ -250 & \text{When reach maximum step} \end{cases}$$

**termination condition$_2$**: if satisfying some specific cryptographic properties, or $N$ steps (we set $N = 250$.)

$$\textbf{reward}_3 = \begin{cases} +1 & \text{When using AND-XOR improves security} \\ 0 & \text{When using AND-XOR maintains security} \\ -1.1 & \text{When using AND-XOR weakens security} \\ -100 & \text{When reach maximum step} \end{cases}$$

**termination condition$_3$**:$N$ steps or using $M$ AND-XORs (In our settings, $N = 100$ and $M \leq 4$.)

# 결과

- 최소 개수로 입증된 4개의 AND로 구현된 최적의 보안으로 4비트 S-box를 구현
- 12개의 AND로 구현된 차동 균일성 16 및 선형성 64로 8비트 S-박스를 달성
- 9개의 AND로 구현된 미분 균일성 16을 갖는 8비트 S-박스를 생성
- SKINNY와 동일한 선형성 128 및 동일한 수의 비선형 연산 8을 갖는 8비트 S-박스를 생성하면서 SKINNY보다 디퍼런셜 32가 더 우수

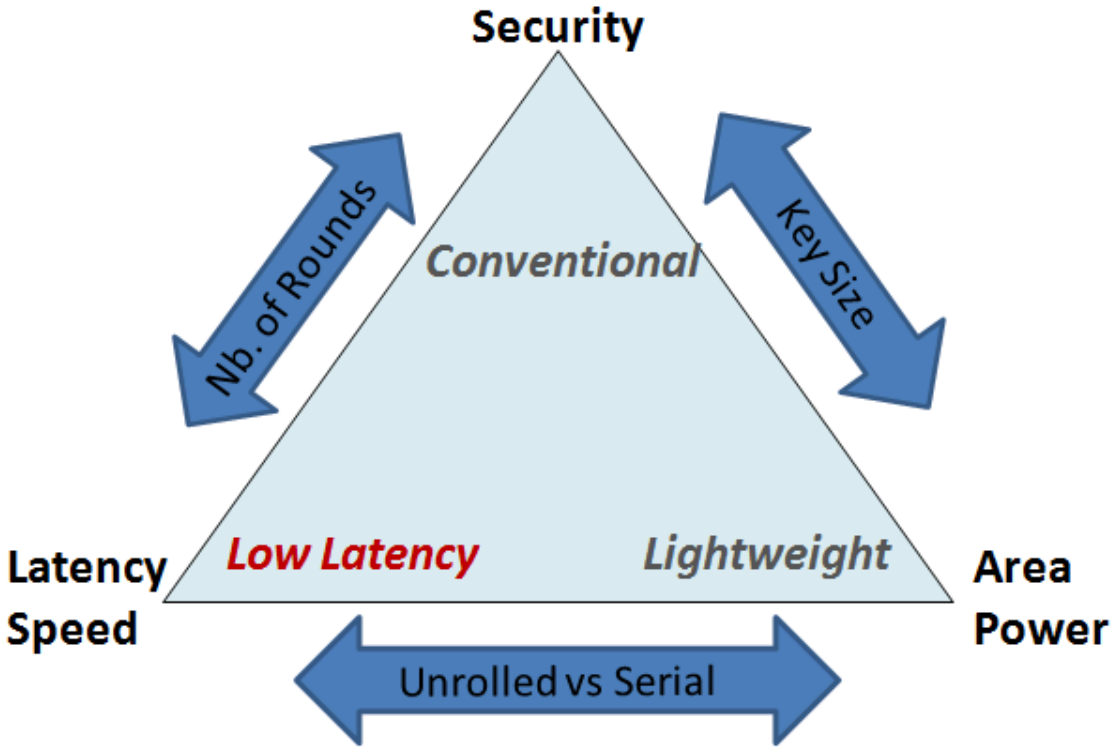
**TABLE 1. Comparison of 4- and 8-bit S-boxes.**

| S-box | S-box bit size | Differential uniformity | Linearity | #(Nonlinear operations) | Ref. |
|---|---|---|---|---|---|
| This paper | **4** | **4** | **8** | **4** | **Listing 1** |
| PRESENT | 4 | 4 | 8 | 4 | [54] |
| RECTANGLE | 4 | 4 | 8 | 4 | [6] |
| GIFT | 4 | 6 | 8 | 4 | [3] |
| Midori (Sb0) | 4 | 4 | 8 | 4* | [55] |
| Midori (Sb1) | 4 | 4 | 8 | 7* | [55] |
| This paper | **8** | **16** | **128** | **9** | **Listing 2** |
| This paper | **8** | **32** | **128** | **8** | **Listing 3** |
| SKINNY | 8 | 64 | 128 | 8 | [52] |
| Midori (SSb0∼SSb3) | 8 | 64 | 128 | 14* | [55] |
| This paper | **8** | **16** | **64** | **12** | **Listing 4** |
| Fantomas | 8 | 16 | 64 | 11 | [12] |
| Robin | 8 | 16 | 64 | 12 | [12] |
| LITTLUN | 8 | 16 | 64 | 12 | [5] |
| LILLIPUT | 8 | 8 | 64 | 12 | [56] |

*The number of nonlinear operations required to implement the S-boxes of Midori could be obtained using the tool Lighter [31].

# 결론

- Side channel
- Lightweight Block Cipher Design
- 혁신적인 디자인
- 필요에 맞춘 암호화
  - 소프트웨어 vs 하드웨어
  - Latency, lightweight, conventional



# Q & A