1. 팀 정보

과제명	Secure and Energy-Efficient Resource Optimization Framework for Real-time Task Execution in Industrial IoT
팀 정보 (팀번호/팀이름)	10 / 2ys
팀 구성원	도연수(2276103), 박연수(2276131)

2. Project-summary(과제 요약)

1. 문제 정의

IIoT는 센서, 전송, 실시간 분석, 자동 제어 순서로 동작하고, 성능·비용을 위해 작업을 public edge server (multi-tenant) 로 오프로딩 하는 흐름이 일반화되었다. 하지만 이때, 보안 및 실시간 데드라인 만족율 등의 성능과 소비 전력, 생산 속도에서 발생하는 비용의 trade-off가 발생할 수 있으므로 전압·메모리·네트워크/오프로딩의 요소를 복합적으로 고려해야 한다.

다음과 같이 4가지 양상에 대해 문제 정의를 할 수 있다.

첫째, 보안 문제다. multi-tenant public edge server를 사용할 경우 호스트 OS나 관리자가 root 권한으로 데이터에 접근할 수 있다. 따라서 기존 연구가 집중하고 있는 application level TEE 는 IIoT 환경에 적용하기 어렵다는 문제가 발생한다.

둘째, 상호의존적인 요소들을 통합적으로 고려하는 프레임워크가 부족하다. CPU, 메모리, 네트워크, 오프로딩 등의 자원은 서로 연동되는데 다수의 선행 연구는 이를 단일 자원 혹은 단일 목표로 분리하여 최적화하고 있다. 일부 상호작용을 일부 고려한 시도도 있으나 보안 요소가 빠져 multi-tenant 환경에는 부적합하다.

셋째, 에너지 관련 문제가 있다. 테스크의 처리 속도는 CPU 전압에 비례하고, 소비 전력은 CPU 전압의 제곱에 비례한다. 따라서 전압 증가는 전력 급증을 유발하고, 무분별한 전압 감소는 처리 지연으로 데드라인을 충족하지 못한다. 이러한 시간, 소비 전력 trade-off는 메모리 선택과 오프로딩 전략에서도 발생한다.

마지막으로, 실시간성 문제가 있다. IIoT에서는 작업 처리뿐만 아니라 자원 배분 알고리즘 또한 ms 단위로 이루어져야 한다. 전통 스케줄러는 빠르지만 복잡한 자원 상호작용과

보안상의 제약을 반영하기 어렵고 딥러닝을 활용한 방식은 데이터 부족과 학습·추론 지연으로 실시간 의사결정에 부적합하다.

2. 기존연구와의 비교

기존 자원 최적화 연구는 대체로 실시간 스케줄링, 에너지 효율, 보안 실행 가운데 일부만을 다뤘고, 이들을 통합적으로 고려한 사례는 드물다. 본 연구는 IIoT 환경에서 보안 보장, 데드라인 준수, 에너지 효율을 하나의 최적화 문제로 엮어 신뢰 가능한 오프로딩을 가능케 한다는 점에서 이 간극을 메운다.

2.2.1. TEE 환경 적용 여부

- **Ki, S., Byun, G., Cho, K., & Bahn, H. (2023). "Co-Optimizing CPU Voltage, Memory Placement, and Task Offloading for Energy-Efficient Mobile Systems". IEEE Internet of Things Journal, 10(10), 9177–9192. https://doi.org/10.1109/JIOT.2022.3233830**

*Ki et al. (IEEE IoTJ, 2023)*은 모바일 실시간 시스템을 대상으로 DVFS·하이브리드 메모리·오프로딩을 공동 최적화(Co-TOMS) 하고, 테스크 수준(real-time) 데드라인과 유전 알고리즘 기반의 빠른 수렴을 제시해 자원 간 상호의존성을 잘 드러냈다. 그러나 해당 연구는 신뢰 모델이 없는(=엣지/클라우드를 신뢰) 전제를 따르며, 보안을 다뤄도 주로 애플리케이션 레벨 관점에 머물러 멀티테넌트 엣지에서의 인프라 수준 TEE 요구와 호스트 OS 위협을 모델링하지 않는다. 또한 TEE 진입/검증/컨텍스트 전환에 따른 지연·에너지 오버헤드가 오프로딩·메모리·전압 선택에 미치는 교차 영향도 최적화 변수로 포함되지 않는다.

반면 본 연구는 VM-level TEE를 전제로 한 보안 제약을 최적화에 직접 포함하고, DVFS, 메모리 배치, 오프로딩의 상호의존성을 TEE 오버헤드와 멀티테넌트 위협 모델까지 아우르는 통합 목적 함수로 다룬다. 그 결과, 기존처럼 자원을 개별 최적화하지 않고 보안 보장·데드라인 준수·에너지 절감·플랫폼 호환성(공용 엣지 활용)을 동시에 달성하는 거래비용 인지형(offload/secure/cost-aware) 스케줄링을 제안한다는 점이 핵심 차별점이다.

2.2.2. 복합 요소 통합 여부

- **K. Choi, W. Lee, R. Soma, and M. Pedram, “Dynamic Voltage and Frequency Scaling under a Precise Energy Model Considering Variable and Fixed Components of the System Power Dissipation,” Proc. IEEE Int’l Conf. on Computer Aided Design (ICCAD), pp. 29–34, 2005.**

- **G. Wang, Y. Guan, Y. Wang, and Z. Shao, “Energy-aware Assignment and Scheduling for Hybrid Main Memory in Embedded Systems,” Computing, vol. 98, pp. 279–301, 2016.**
- **Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, “Mobile-edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling,” IEEE Transactions on Communications, vol. 64, no. 10, pp. 4268–4282, 2016.**

선행연구는 각기 단일 자원 층에 초점을 맞춘다. 예를 들어 *Choi et al.(2005)*는 CPU 전압/주파수 동적 조절(DVS)로 에너지를 줄이지만, 메모리·네트워크·보안은 거의 고려하지 않는다. *Wang et al.(2016)*은 임베디드 환경에서 DRAM+저전력 메모리(LPM)의 하이브리드 메모리 배치만 정밀 최적화하고, 오프로딩·보안·CPU 제어와의 연계는 부족하다. 또 다른 *Wang et al.(2016, IEEE TCOMM)*은 모바일 엣지에서 부분 오프로딩과 DVS를 결합하지만, 네트워크·플랫폼 보안(특히 VM-level 보호)까지 포괄하지 못한다.

우리 논문은 이러한 단편적 접근을 넘어, CPU 전압 스케일링(DVS), 메모리 배치, 오프로딩을 하나의 정책 공간에서 동시에 최적화하는 통합 프레임워크를 제안한다. 구체적으로, 각 자원 층의 조정이 서로의 최적화에 미치는 상호작용을 수학적으로 모델링하고, 실시간 데드라인 제약 하에서 목적함수(에너지·성능·비용)를 공동 최적화한다. 더 나아가 기존 연구들이 주로 application-level TEE에 머문 것과 달리, 우리는 VM-level TEE를 전제로 오프로딩의 보안 실행·적합성을 보장하고, TEE 진입/검증·컨텍스트 전환 오버헤드까지 최적화 변수에 포함해 멀티테넌트 공용 엣지에서의 신뢰 가능한(offload-secure) 스케줄링을 가능케 한다. 결과적으로, 단일 자원별 최적화에서 발생하는 지역해와 정책 충돌을 피하고, 보안·데드라인·에너지·비용을 동시에 만족하는 실용적 해법을 제공한다.

3. 제안 내용

IIoT 환경의 핵심 과제와 대응을 간략히 정리하면 다음과 같다.

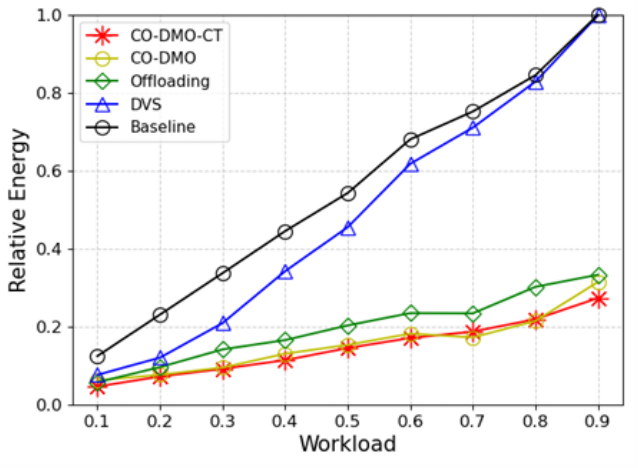
첫째, 호스트 OS에서의 root 권한을 통한 기밀 노출 위험이 있어 VM-level TEE (AWS Nitro Enclaves)를 도입해 민감한 데이터를 격리하고 호스트 접근을 원천 차단한다.

둘째, 상호의존적인 CPU, 메모리, 오프로딩의 세 자원을 하나의 정책 공간에서 보안, 데드라인 만족, 에너지 절약을 동시 목표로 공동 최적화하는 통합 프레임워크로 해결한다.

셋째, 전압에 따른 시간, 소비 전력 trade-off 는 DVS, 하이브리드 메모리(DRAM, LPM), 적응형 오프로딩(로컬↔엣지)을 도입하여 전력 절감과 실시간성을 함께 달성하는 것을 목표로 한다. 마지막으로, 실시간 의사결정을 위해 경량 GA 알고리즘을 적용해 task당 10 ms 내에 전압, 메모리, 오프로딩 인덱스를 지정하고 실시간 제어를 확보한다.

4. 기대 효과 및 의의

본 과제는 보안 환경 속 multi-tenant public edge server를 적극 활용하면서도 실시간 데드라인을 준수하고 비용 효율성을 확보하는 자원 최적화 프레임워크를 제시한다. VM-level TEE를 통해 호스트 OS 침해로부터 오프로딩 작업을 격리·보호함으로써 보안성과 플랫폼 호환성을 동시에 보장하고, 정밀한 실행시간·에너지 모델을 바탕으로 프로세서 전압 스케일링(DVS), 메모리 배치, 오프로딩 결정을 데드라인 제약 하에 공동 최적화한다. 이는 각 자원을 따로 최적화하던 기존 방식과 달리, 한 층의 조정이 다른 층의 최적화에 영향을 미치는 자원 간 상호의존성을 명시적으로 반영하여 보안이 보장되는 실시간 실행을 위한 비용 효과적이고 호환 가능한 최적화 전략을 제공한다. 모의실험 결과, 제안 프레임워크는 baseline에 비해 에너지 소비를 70% 로 유의미하게 감소시키는 것으로 확인되었으며, 확장성과 보안을 동시에 요구하는 IIoT 환경에 높은 적용 가능성을 갖는다.



5. 주요 기능 리스트

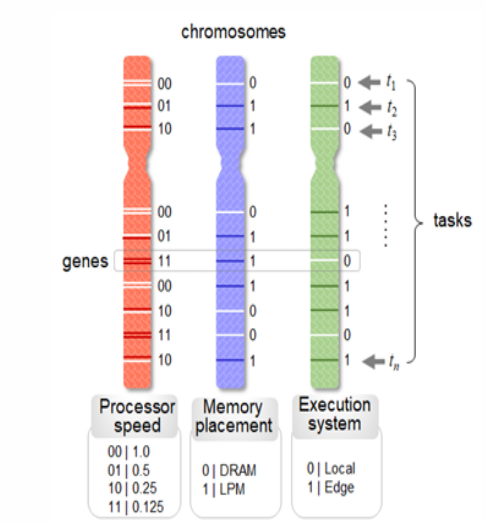
본 프레임워크는 각 테스크 t_i 에 대해 CPU 전압, 메모리 배치, 오프로딩 여부를 동시에 결정하며, 전체 태스크 집합 T의 에너지 소비를 최소화하는 동시에 실시간 제약 조건과 보안 요구사항을 만족하는 최적 해를 찾아낸다.

유전 알고리즘(GA)을 활용하여, 다양한 솔루션을 염색체로 표현하고 반복적인 선택·교차·변이 과정을 통해 에너지 최소화와 제약 조건 만족을 동시에 달성한다.

각 솔루션은 세 가지 염색체로 구성되어, 우리 프레임워크의 세 가지 주축인 자원 결정 및 인코딩, 보안 및 실시간성 보장, 그리고 에너지 최적화를 동시에 달성할 수 있도록 한다.

2.5.1. 자원 결정 및 인코딩(Resource Allocation & Encoding)

유전 알고리즘은 각 솔루션(해)을 세 개의 염색체로 인코딩하며, 이는 자원의 동시 최적화를 가능하게 하는 기본 구성 요소다.




```
# wcet_sacle power_active power_idle
*cpufreq
1     100      1
0.5   25    0.25
0.25  6.25  0.0625
0.125 1.5625 0.015625

# type max_capacity wcet_scale power_active power_idle
*mem
dram  1000 1    0.01   0.009
nvram 1000 0.8  0.01   0.0009

# offloading_ratio
*offloadingratio
0
1
```

(A) CPU 전압, 메모리 배치, 실행 위치 염색체 (B) 코드 설계

1. 프로세서 실행 전압 및 속도 **(Dynamic Voltage Scaling, DVS)**
   - 테스크 실행 속도를 결정하는 DVS는 프로세서 에너지 E_processor 최소화의 주요 수단이다.
   - 연속 전압 대신 4단계 이산 전압/주파수 레벨 사용
   - 전압이 낮을수록 테스크 최대 실행 시간(WCET_i) 증가, 에너지 E = C * V^2 * f 감소
   - 유전 알고리즘 염색체를 통해 최적 레벨 선택

2. 메모리 배치 **(Hybrid Memory Allocation)**
   - 테스크를 DRAM 또는 저전력 메모리(LPM)에 배치
   - LPM은 전력 소모가 낮지만 접근 시간이 길어 WCET에 영향
   - 동적/정적 메모리 에너지(E_memory_dynamic, E_memory_static)과 프로세서 활용률 고려
   - 유전 알고리즘으로 최적 배치 선택

3. 실행 위치 결정 **(Execution System / Offloading)**
   - 테스크를 로컬 CPU에서 실행할지, 에지 서버로 오프로딩할지 결정
   - 오프로딩 시 로컬 프로세서 부하 감소 → 낮은 전압/주파수 적용 가능
   - 네트워크 지연과 암복호화 오버헤드 고려

2.5.2. 보안 및 GA 알고리즘의 실시간성 보장 (Security & Real-time Guarantee)

프레임워크는 에너지 최적화 외에도 태스크 실행의 보안과 실시간성을 필수 제약 조건으로 다룹니다.

1. 보안 고려사항 (Security)

보안은 주로 오프로딩 과정에서 중요하게 다루어진다.

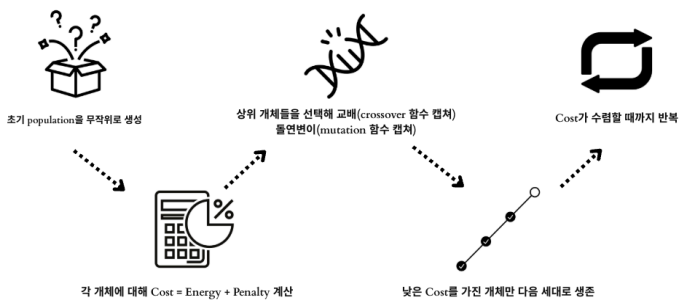
- 데이터 암호화/복호화 오버헤드: 에지 서버로 오프로드되는 모든 태스크($t_i \in EDGE$)는 데이터의 기밀성을 보장하기 위해 엔드 디바이스에서 전송 전 Input i를 암호화($T_{Enc}(Input_i)$)하고, 에지 서버에서 결과를 받은 후 $Output_i$를 복호화($TDec(Output_i)$)한다.
- 프로세서 에너지 반영: 이 암호화 및 복호화에 소요되는 시간은 로컬 프로세서의 컴퓨팅 에너지($E_{processor}$)에 추가적인 부하로 포함된다.
- **TEE** 환경 오버헤드: 에지 서버에서의 안전한 실행을 위한 Trusted Execution Environment (TEE) 사용은 $Remote(t_i)$ 시간 계산 시 $TEE_{overhead}(t_i)$ 항을 통해 반영되어, 실시간 제약 조건에 미치는 영향을 명시적으로 고려한다.

2. GA 알고리즘의 실시간성 보장 (Real-time Guarantee)

GA 알고리즘에서 도입하는 실시간 제약 조건은 비용 함수(Cost Function)의 페널티 항 $P(i)$을 통해 강제적으로 만족된다. 이는 최적화 과정에서 '에너지 최소화'만 추구하여 발생하는 비실행 가능 해(Infeasible Solutions)를 제거하는 역할을 한다.

1. 로컬 스케줄 가능성 제약 **(Schedulability)**:
    - 제약: 로컬 태스크의 총 활용률 $U(i)$가 단일 코어에서 1을 초과해서는 안 된다.
    - 페널티: $\gamma \max\{0, U(i)-1\}$ 항을 통해 활용률이 1을 초과하는 해에 큰 페널티를 부과하여, 유전 알고리즘이 실행 가능한 해($U(i) \le 1$)로 수렴하도록 한다.
2. 오프로드 태스크 마감 기한 제약 **(Offloading Deadline)**:
    - 제약: 오프로드된 태스크의 전체 왕복 시간(T_{rnarnd_j})은 태스크의 주기($Period_j$)보다 짧아야 한다 ($\forall t_i \in EDGE, T_{rnarnd_i} < Period_i$).
    - 페널티: $\delta \sum_{t_j \in EDGE} \max\{0, T_{rnarnd_j} - Period_j\}$ 항을 통해 마감 기한을 위반하는 오프로드 태스크에 페널티를 부과한다. T_{rnarnd_j}는 암복호화, 업로드/다운로드 시간(Up_i, Dn_i), 에지 실행 시간($Remote(t_i)$)을 모두 포함하므로, 이 제약은 네트워크 및 서버 지연까지 고려한 실시간성을 보장한다.

GA 알고리즘의 최적화 과정을 간단한 그림으로 표현한 것이다.



2.5.3. GA 알고리즘의 에너지 최적화 (Energy Minimization)

GA 알고리즘에서 Cost function에 포함되는 에너지 소비(Energy(i))는 프로세서, 네트워크, 메모리 에너지의 총합으로 계산된다.

- **E_processor** 최소화: DVS를 통해 로컬 실행 태스크의 전압/주파수를 낮춘다.
- **E_network** 최소화: 오프로딩 결정 시 E_network (데이터 전송 시간)와 T_{rnarnd_i}를 동시에 고려하여, 네트워크 지연이 적은 환경에서만 오프로딩을 선택하게 한다.
- **E_memory** 최소화: 태스크를 LPM에 배치하여 동적/정적 메모리 에너지를 절감하는 동시에, 이로 인해 발생하는 WCET 증가가 로컬 활용률 제약을 위반하지 않도록 관리한다.

이러한 방식으로, 유전 알고리즘은 페널티가 0이 되는 (실시간 및 보안 제약 조건을 만족하는) 해 중에서 Energy(i)가 가장 낮은 해를 찾아나가며 전력 소비를 절감하는 기능을 제공한다.

```
*CO-DMO-CT
initial utilization: 1.467775
power: 5.650144 util: 0.681580
cpu power: 1.846591 memory power: 2.105383 network power: 1.698170
offloading ratio: 0.890000
cpu frequency:
1   0.5 0.25    0.125
26  19  27  28
period violation: 0
```

3. Project Design(과제 설계)

1. 요구사항 정의

3.1.1. 보안 관련 요구 사항

- 시스템은 VM 수준에서 보안 기능을 제공해야 한다.

- 시스템은 데이터 암호화와 복호화 기능을 수행해야 한다.
- 암호화/복호화 과정에서 발생하는 시간 지연은 Cost 계산에 반영되어야 한다.
- 암호화/복호화 과정에서 발생하는 전력 소비는 Cost 계산에 포함되어야 한다.
- 현실적인 보안 환경을 구현하여 성능과 전력 측면에서 최적화 가능해야 한다.

3.1.2. 통합 프레임워크 관련 요구 사항

- 시스템은 CPU, 메모리, 오프로딩 등 여러 자원을 동시에 고려하여 자원 배분을 수행해야 한다.
- 자원 배분의 기준이 되는 Cost 값은 Energy (소비 전력) + Penalty (데드라인 Miss)로 계산되어야 한다.
    - Energy는 각 자원에서 발생하는 소비 전력의 합이다.
    - Penalty는 max(0, Utilization - 1) + max(0, Turnaround_time - Period) 로 정의한다.

- 시스템은 각 자원의 상태를 인코딩하여 비용 계산에 활용해야 한다.
    - CPU는 전압을 4가지 종류로 설정한다.
    - 메모리는 DRAM과 LPM 2가지 종류로 설정한다.
    - 오프로딩은 true / false로 2가지 종류로 설정한다.
    - CPU 전압에 따라 task 실행 시간이 상대적으로 결정된다.
    - CPU 전압 1을 기준으로 전압이 N배될 때, task 실행 시간은 1/N배가 된다.
    - 모든 자원 종류마다 활성 전력 (Power Active)과 유휴 전력 (Power Idle)이 결정된다.
    - 결정된 전력은 누적 합을 통해 Cost에서 Energy 항목에 대입된다.

- 시스템은 자원 외에도 오프로딩 시 네트워크 uplink bandwidth, downlink bandwidth와 파일 입출력(input_file size, output_file size) 등의 환경을 고려해야 한다.
    - 네트워크 bandwidth에 따라 추가 시간 오버헤드가 생긴다.
    - 오프로딩이 true일 경우, 네트워크와 보안 요소를 Cost에 반영해야 한다.
    - 오프로딩 시, Network Commander에서의 추가 소비 전력 오버헤드가 생긴다.
    - 네트워크 uplink bandwidth, downlink bandwidth 에 따라 turnaround time에 시간 오버헤드가 추가된다.
    - 보안 요소는 위에서 언급한 바와 같다.
    - 오프로딩이 false인 경우에도 local IoT device에서 Cost를 최소화하고, Deadline Miss를 거의 발생시키지 않아야 한다. (최소화하도록 설계)

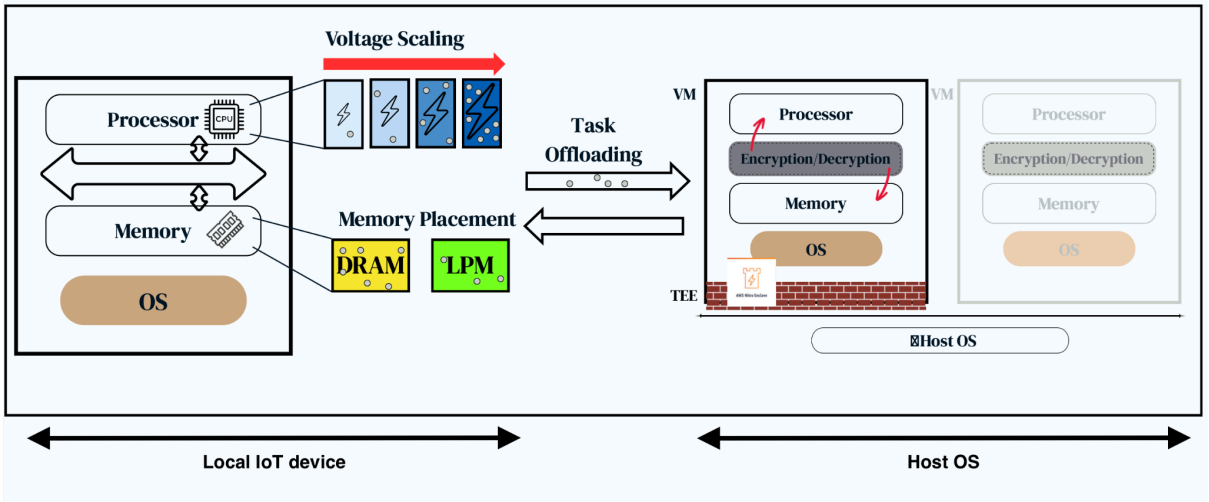
- 시스템은 GA(Genetic Algorithm)를 이용해 자원을 실시간으로 결정하고 배분할 수 있어야 한다. 즉, taskset당 100 ms 단위 내외로 수행되어야 한다.

3.1.2. 예상 결과

- Baseline보다 Energy 를 70% 감소시킨 결과
- 보안 요소를 고려하지 않은 기존 모델보다 Deadline Miss를 적게 유지하는 결과
- IIoT dataset 을 적용했을 때, Penalty가 Baseline Penalty보다 적은 결과
- 다양한 CPU 전압을 활성화하는 결과 (각 CPU 전압당 15~30 % 예상)
- Network가 불안정하여 Offloading 이 불가능한 환경에서도 Utilization을 1보다 작게 유지하는 결과

2. 전체 시스템 구성

전체 시스템 구성도는 다음과 같다.



Local IoT device에서 Host OS로 테스크를 오프로딩할 때, Host OS 내부에서는 Local IoT device의 오프로딩된 테스트를 VM 차원의 TEE로 보호해 안전한 환경을 조성한다.

로컬(=Local IoT device)과 엣지(=Host OS) 각각의 내부 구성은 다음과 같다.

3.2.1. Local IoT Device

- 4가지의 CPU 전압을 가지고 있다.
- 2가지의 Memory를 가지고 있다.
- 해당 프레임워크를 도입하여 GA 알고리즘 계산이 가능한 OS 위에서 작동한다.

3.2.2. Host OS

- AWS Nitro Enclave의 VM- level TEE를 도입하여 tenant마다의 VM을 보호하고 있다. 또한, Host OS에서는 각 VM Nitro Enclave 내부 에 접근할 수 없다.
- VM에서 data를 처리할 때 암호화, 복호화 과정이 일어난다. Host OS는 이 암호화 및 복호화를 할 수 없다.