Side Channel Attack for Neural Network

https://youtu.be/vX9EFq5E51U

Side Channel Attack on Neural Network

Chip Whisperer

Side Channel Attack on Neural Network

- **부채널 공격**

  전력 소비 등의 부채널 정보로 비밀 정보를 알아냄

  Timing Attack, Power Analysis, Template Attack 등이 있음

- **암호 알고리즘에서의 부채널 공격**

  평문 (알려진 값)과 특정 연산 수행 시의 파형을 아는 상태에서 키를 찾음

- **신경망에 대한 부채널 공격**

  입력데이터 (알려진 값)과 특정 연산 수행 시의 파형을 아는 상태에서

  신경망의 구조 (뉴런의 수, 레이어의 수 및 종류, 활성화 함수 등)나 비밀 가중치를 찾음

Side Channel Attack on Neural Network

- Embedding layer / activation 등에 대한 부채널 분석 수행하려고 했으나

  embedding은 8-bit로 양자화 하지 않으며 (일반적인 레이어가 아니라 단어를 벡터로 바꿔주는 것),

  TPU 같은 경우는 지원하지 않는 레이어이기도 함

  activation은 주로 타이밍 공격을 수행

- 엣지 디바이스에서 효율적인 BNN에서 중요하기도 하고, 과적합 방지를 위해 여러 네트워크에서 자주 쓰이는 **Batch normalization에 대해 공격 수행**하고자 함

- 8-bit 자료형을 사용하는 이유

  엣지 디바이스에서는 추론을 위해 양자화시킨 후, int8을 사용하며

  부채널 공격을 위해 float는 사용할 수 없음

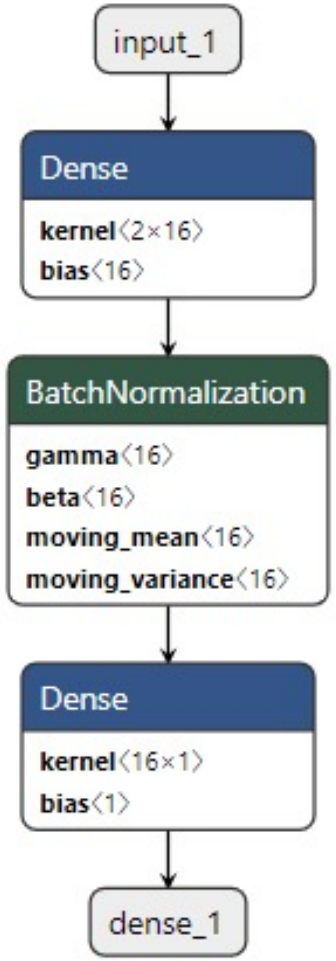
Side Channel Attack on Neural Network

- **Batch Normalization 파악** (전체 네트워크에서 해당 레이어의 유무와 구조 파악)
- **Plaintext**

  해당 레이어에 입력되는 데이터 (이전 레이어의 출력)
- **Key**

  Batch Normalization에서 사용되는 비밀 값 (mean, gamma, … 등의 옵션 있음)
- Fully connected – Batch Normalization – Fully connected 로 구성
- **파형 수집 대상 구간** : Batch Normalization
- 이미 **훈련된 후의 모델을 대상**으로 함

  → 따라서, 가중치 등의 내부 파라미터가 **고정**된 상태



Side Channel Attack on Neural Network

- **Data type**

  32-bit 부동 소수점 아닌 **8-bit unsigned int** 로 변환

  (0~255로 scaling)

- **Input** (이전 layer의 output)

  16진수 형태의 평문처럼 표현 가능

- **Tensor는 구조체로 표현**

  input, output tensor의 값, kernel의 array 등이 담김

```
struct k2c_tensor
{
    float * array;

    size_t ndim;

    size_t numel;

    size_t shape[K2C_MAX_NDIM];
};
```

```
uint8_t float2int(float num) {

    uint8_t num_bin = 0x00;

    float OldMin = -1;
    float OldMax = 1;
    float NewMin = 0;
    float NewMax = 255;
    int OldRange = (OldMax - OldMin);
    int NewRange = (NewMax - NewMin);

    num_bin = (int)((((num - OldMin) * NewRange) / OldRange) + NewMin);

    return num_bin;
}
```

```
Result of first layer
5D
96
B0
C8
4B
6E
76
E8
D1
E2
EA
CB
3B
F1
C6
F2
```

Side Channel Attack on Neural Network

- **Batch Normalization**

  stdev, gamma, beta, axis 등의 옵션이 있는 레이어

  input은 이전 레이어의 output

```
void k2c_batch_norm(k2c_tensor* outputs, const k2c_tensor* inputs, const k2c_tensor* mean, const k2c_tensor* stdev, const k2c_tensor* gamma, const k2c_tensor* beta, const size_t axis) {

    trigger_high();

    size_t offset = 1;
    for (size_t i = axis + 1; i < inputs->ndim; ++i) {
        offset *= inputs->shape[i];
    }
    const size_t step = inputs->shape[axis];

    for (size_t i = 0; i < inputs->numel; ++i) {
        size_t idx = (i / offset) % step;
        outputs->array[i] = (inputs->array[i] - mean->array[idx]) /
            stdev->array[idx] *
            gamma->array[idx] +
            beta->array[idx];
    }

    trigger_low();
}
```

Side Channel Attack on Neural Network

- **Parameters of Batch Normalization**

  stdev, gamma, beta, axis 등

  키나 평문이 최대 32바이트까지 가능해서 16바이트로 맞춤

  → float type이라서 4 * 16 byte이지만

  chip whisperer를 통한 부채널 공격 위해 uint8_t로 변경 후 사용

```
float batch_normalization_mean_array[16] = {
-2.88165426e+00,+2.51397896e+00,-3.09186161e-01,+5.86879969e-01,-3.28274846e+00,
+1.66503537e+00,-2.36293197e+00,+4.53979826e+00,-4.23647987e-04,+4.71465081e-01,
+4.57742023e+00,+3.52812815e+00,-3.97026467e+00,+2.64014339e+00,+5.89172468e-02,
+4.97595358e+00 };
```

```
float batch_normalization_gamma_array[16] = {
+1.08548467e+00,+1.06108236e+00,+1.06620824e+00,+1.05295527e+00,+1.05376351e+00,
+1.06392968e+00,+9.51860868e-01,+9.68106866e-01,+9.53865051e-01,+9.52293456e-01,
+9.61142600e-01,+1.06145430e+00,+1.03887475e+00,+1.05905712e+00,+1.01936054e+00,
+9.71215308e-01 };
```

```
float batch_normalization_stdev_array[16] = {
+1.25909841e+00,+1.10389912e+00,+8.82446647e-01,+1.06348407e+00,+1.44767845e+00,
+9.49389160e-01,+1.07727110e+00,+2.11294055e+00,+1.34478855e+00,+1.50487638e+00,
+2.15490961e+00,+1.60830998e+00,+1.75411034e+00,+1.79152536e+00,+1.11022174e+00,
+2.32292819e+00 };
```

```
float batch_normalization_beta_array[16] = {
-7.10431576e-01,+5.90539873e-01,-5.93534172e-01,-7.05252111e-01,-6.32478118e-01,
+5.66929340e-01,+6.14797413e-01,-6.71868563e-01,+6.14375889e-01,+6.05685532e-01,
-6.48504078e-01,+5.85249484e-01,-6.97062254e-01,+6.30346715e-01,-7.83671677e-01,
-6.81032062e-01, };
```

Chip Whisperer 파형 수집

- **CW를 target board로 사용**

  훈련된 모델을 Chip Whisperer에 업로드한 후 실행하면서 파형수집

- **순서**

  파형 수집 코드 작성 → 컴파일 → binary file (.hex) → CW → trace capture → trace analysis

- plaintext, key 및 trace capture 구간 설정위한 **코드 작성 필요**

  get_pt, get_key, trigger_high(), trigger_low()

Chip Whisperer

- get_key, get_pt 필요

  get_pt에서 추론 수행하고 trace capture

- get_key는 **고정된 가중치라서 상수값**이므로 냅둠

```
uint8_t get_key(uint8_t* k)
{
    return 0x00;
}
```

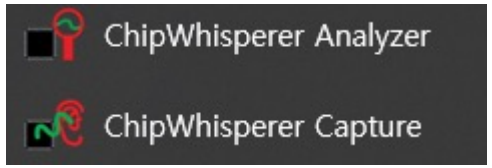
- **get_pt에서 앞서 본 레이어들을 수행하며 파형 수집**

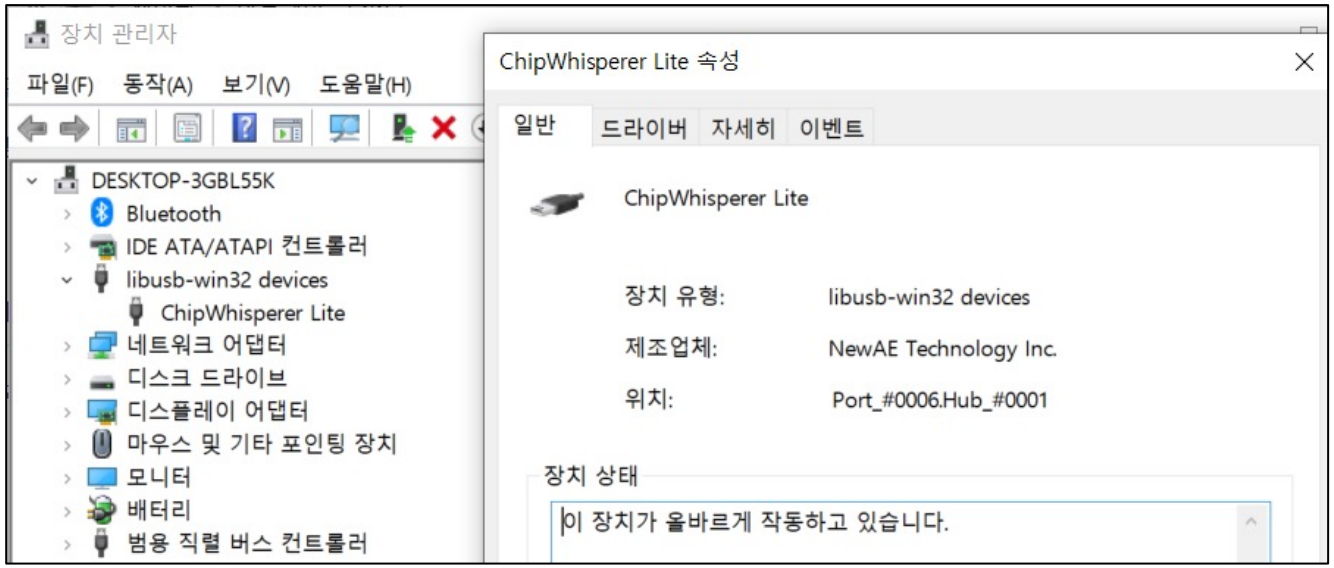
```
    trigger_high();

    size_t offset = 1;
    for (size_t i = axis + 1; i < inputs->ndim; ++i) {
        offset *= inputs->shape[i];
    }
    const size_t step = inputs->shape[axis];

    for (size_t i = 0; i < inputs->numel; ++i) {
        size_t idx = (i / offset) % step;
        outputs->array[i] = (inputs->array[i] - mean->array[idx]) /
            stdev->array[idx] *
            gamma->array[idx] +
            beta->array[idx];
    }

    trigger_low();
```

```
int main() {

    platform_init();
    init_uart();
    trigger_setup();

    /* Uncomment this to get a HELLO message for debug */
    putch('h');
    putch('e');
    putch('l');
    putch('l');
    putch('o');
    putch('\n');

    simpleserial_init();
    simpleserial_addcmd('k', 16, get_key);
    simpleserial_addcmd('p', 16, get_pt);

    while (1)
        simpleserial_get();

    return 0;
}
```

Chip Whisperer 사용

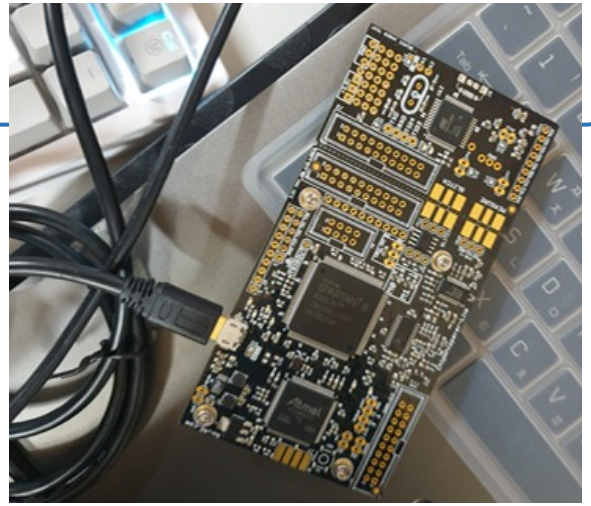
- CW와 드라이버 설치 및 컴파일





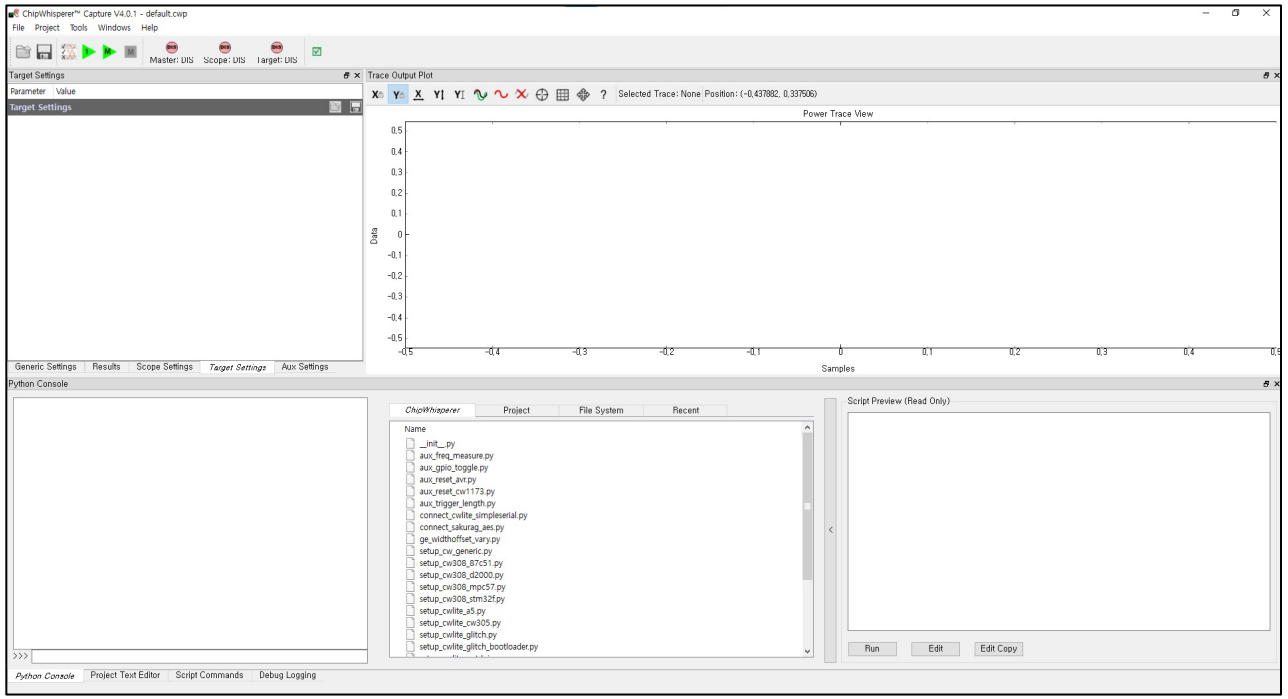

```
C:₩Program Files (x86)₩ChipWhisperer₩chipwhisperer₩hardware₩victims₩firmware₩simpleserial-LAB-SPECIFIC-FOLDER₩Embeddin make PLATFORM=CW303
rm -f -- simpleserial-aes-CW303.hex
rm -f -- simpleserial-aes-CW303.eep
rm -f -- simpleserial-aes-CW303.cof
rm -f -- simpleserial-aes-CW303.elf
rm -f -- simpleserial-aes-CW303.map
rm -f -- simpleserial-aes-CW303.sym
rm -f -- simpleserial-aes-CW303.lss
rm -f -- objdir/*.o
make: [clean_objs] Error -1073741502 (ignored)
rm -f -- objdir/*.lst
make: [clean_objs] Error -1073741502 (ignored)
rm -f -- simpleserial-aes.s simpleserial.s XMEGA_AES_driver.s uart.s usart_driver.s xmega_hal.s
rm -f -- simpleserial-aes.d simpleserial.d XMEGA_AES_driver.d uart.d usart_driver.d xmega_hal.d
rm -f -- simpleserial-aes.i simpleserial.i XMEGA_AES_driver.i uart.i usart_driver.i xmega_hal.i

-------- begin --------
```

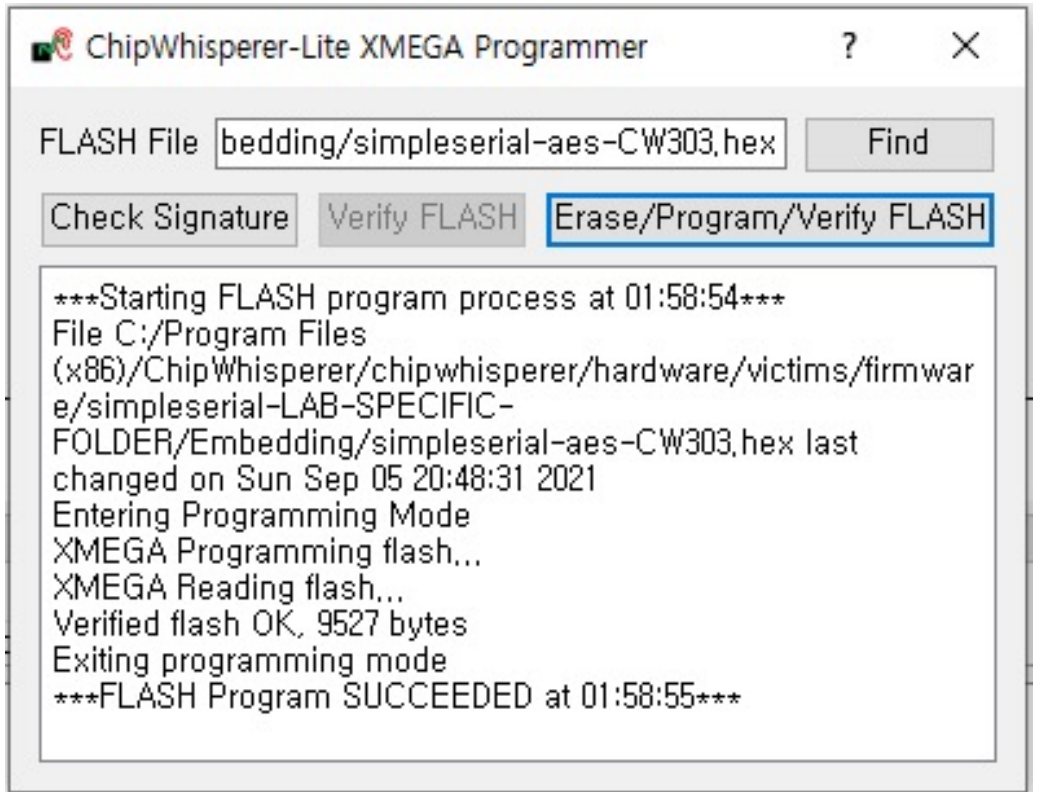
Chip Whisperer 사용



- CW capture

  CW와 노트북 연결 → connect → FLASH file 선택 → 파형 수집

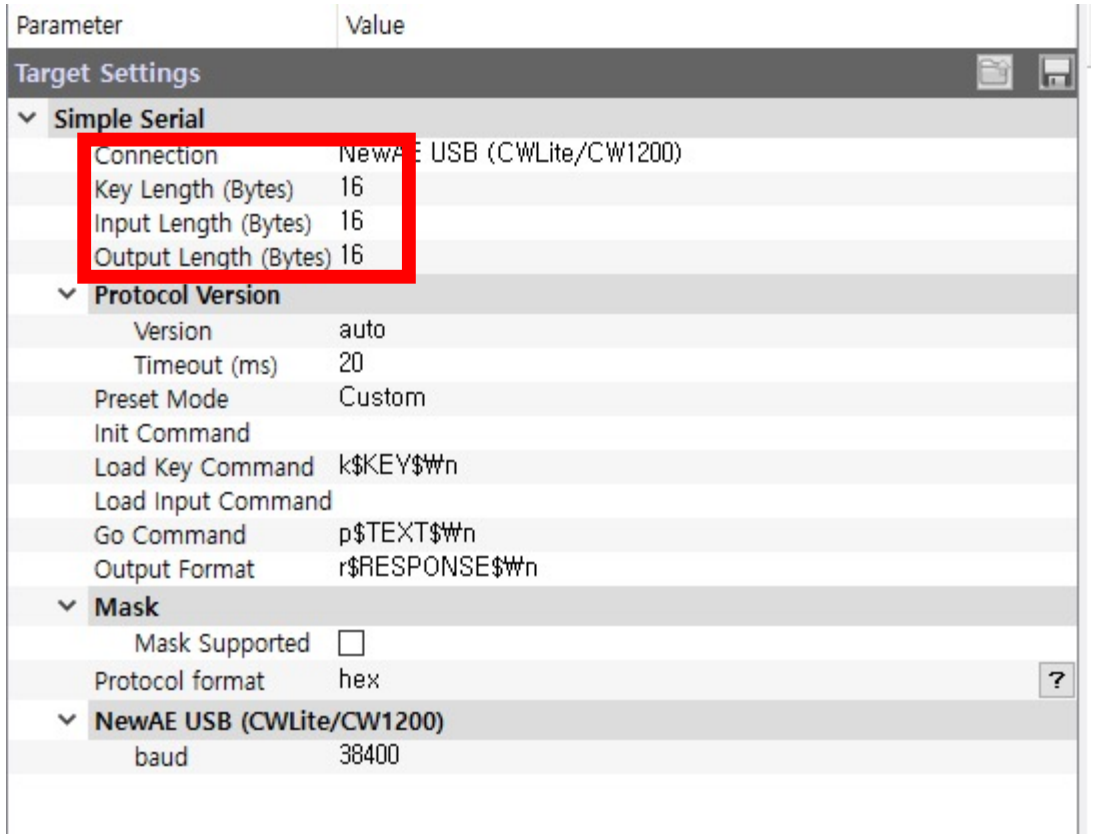


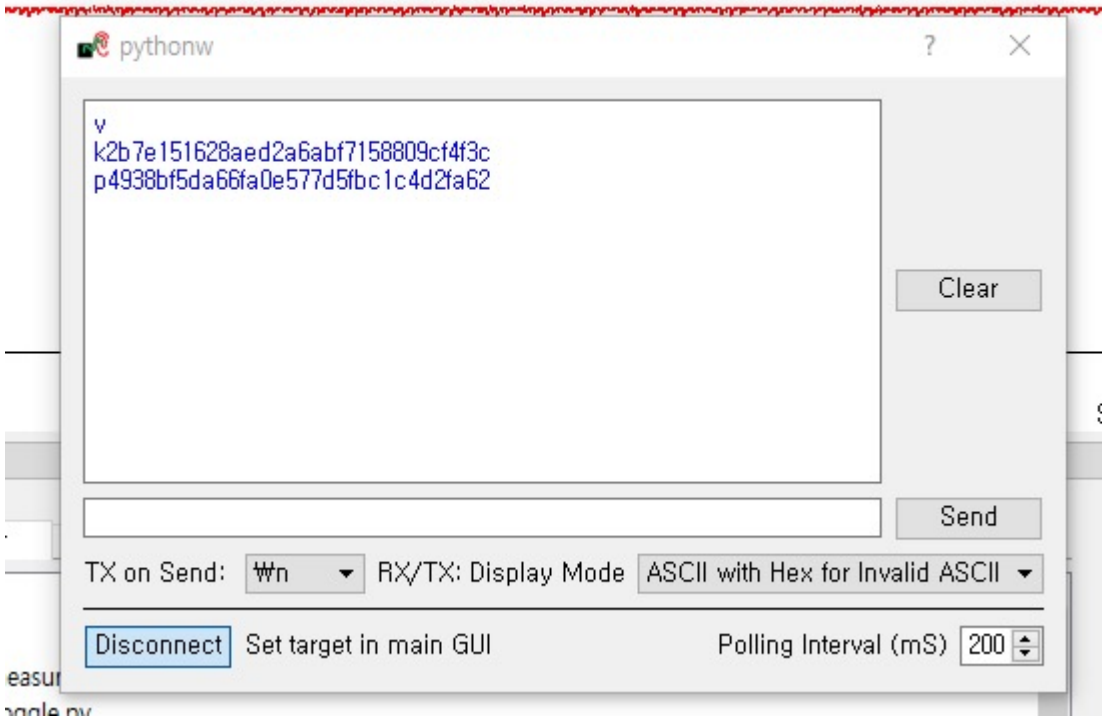


Chip Whisperer 사용

- CW capture

  평문, 키 byte 설정 → 입출력 제대로 되는 지 확인





문제점

- 파형 및 출력이 떠야함 (제대로 연결되었을 경우 hello가 먼저 떠야 함)

  1. CW가 연결이 안 되어서 다른 CW로 했더니 연결은 가능해짐
  2. 파형수집이 잘 되지 않음

  → 튜토리얼 코드와 아무 것도 안 넣은 코드 수행해도

  가장 먼저 실행되는 putch 자체가 실행되지 않음

- 세팅 문제인 것 같다고 하셔서 처음부터 다시 환경 구축해도 안 되어서 파형 수집하지 못했습니다..

감사합니다.