

Tinyjambu CPA 공격 시도

정보컴퓨터공학과 권혁동

Tinyjambu

CPA 공격 시도

향후 과제

Tinyjambu

- NIST **경량 암호 공모전**에 출품된 경량 암호
 - **Finalist** 중 하나
 - AEAD 지원
- 총 세 가지의 규격을 제공
 - Tinyjambu-128
 - Tinyjambu-192
 - Tinyjambu-256
- 암호/복호화는 크게 네 단계로 구성됨
 - Initialization
 - Processing
 - Encryption/Decryption
 - Finalization

Tinyjambu

- Initialization

- 입력 받은 키와 논스를 사용하여 초기 상태 변수(state)를 생성하는 과정
- Tinyjambu-128: 128-bit 키, 96-bit 논스를 입력 받아 **P₁₀₂₄ state**를 생성
- Tinyjambu-192: 192-bit 키, 96-bit 논스를 입력 받아 **P₁₂₈₀ state**를 생성

```
StateUpdate(S, K, i):  
    feedback =  $s_0 \oplus s_{47} \oplus (\sim(s_{70} \& s_{85})) \oplus s_{91} \oplus k_{i \bmod klen}$   
    for j from 0 to 126:  $s_j = s_{j+1}$   
     $s_{127} = \text{feedback}$   
end
```

- Processing

- Associated data 생성 과정
- P384 state를 생성

```
for i from 0 to  $\lfloor adlen/32 \rfloor$ :  
     $s_{\{36 \dots 38\}} = s_{\{36 \dots 38\}} \oplus FrameBits_{\{0 \dots 2\}}$   
    Update the state using  $P_{384}$   
     $s_{\{96 \dots 127\}} = s_{\{96 \dots 127\}} \oplus ad_{\{32i \dots 32i+31\}}$   
end for
```

Tinyjambu

- Encryption

- 앞선 과정에서 생성한 상태 변수(state)를 사용하여 평문을 암호화
- Tinyjambu-128: P_{1024} state 사용
- Tinyjambu-192, 256: P_{1280} state 사용

```
for i from 0 to  $\lfloor mlen/32 \rfloor$ :  
     $s_{\{36...38\}} = s_{\{36...38\}} \oplus FrameBits_{\{0...2\}}$   
    Update the state using  $P_{1024}$   
     $s_{\{96...127\}} = s_{\{96...127\}} \oplus m_{\{32i...32i+31\}}$   
     $c_{\{32i...32i+31\}} = s_{\{64...95\}} \oplus m_{\{32i...32i+31\}}$   
end for
```

- Finalization

- 상태 변수를 사용하여 태그를 생성

```
 $s_{\{36...38\}} = s_{\{36...38\}} \oplus FrameBits_{\{0...2\}}$   
Update the state using  $P_{1024}$   
 $t_{\{0...31\}} = s_{\{64...95\}}$   
  
 $s_{\{36...38\}} = s_{\{36...38\}} \oplus FrameBits_{\{0...2\}}$   
Update the state using  $P_{384}$   
 $t_{\{32...63\}} = s_{\{64...95\}}$ 
```

Tinyjambu

- Tinyjambu의 구현 코드

- Initialization
- Processing
- Encryption/Decryption
- Finalization

```
for (i = 0; i < (adlen >> 2); i++)
{
    state[1] ^= FrameBitsAD;
    state_update(state, k, NROUND1);
    state[3] ^= ((unsigned int*)ad)[i]
}

// if adlen is not a multiple of 4, we pro
if ((adlen & 3) > 0)
{
    state[1] ^= FrameBitsAD;
    state_update(state, k, NROUND1);
    for (j = 0; j < (adlen & 3); j++)
        state[1] ^= adlen & 3;
}
```

```
//initialization stage
initialization(k, npub, state);
```

```
//process the associated data
process_ad(k, ad, adlen, state);
```

```
//process the plaintext
for (i = 0; i < (mlen >> 2); i++)
{
    state[1] ^= FrameBitsPC;
    state_update(state, k, NROUND2);
    state[3] ^= ((unsigned int*)m)[i];
    ((unsigned int*)c)[i] = state[2] ^
}

// if mlen is not a multiple of 4, we p
if ((mlen & 3) > 0)
{
    state[1] ^= FrameBitsPC;
    state_update(state, k, NROUND2);
    for (j = 0; j < (mlen & 3); j++)
    {
        ((unsigned char*)state)
        c[(i << 2) + j] = ((uns
    }
    state[1] ^= mlen & 3;
}
```

```
//finalization stage, we assume that th
state[1] ^= FrameBitsFinalization;
state_update(state, k, NROUND2);
((unsigned int*)mac)[0] = state[2];

state[1] ^= FrameBitsFinalization;
state_update(state, k, NROUND1);
((unsigned int*)mac)[1] = state[2];

*clen = mlen + 8;
for (j = 0; j < 8; j++) c[mlen+j] = mac
```

```
//initialize the state as 0
for (i = 0; i < 4; i++) state[i] = 0;

//update the state with the key
state_update(state, key, NROUND2);

//introduce IV into the state
for (i = 0; i < 3; i++)
{
    state[1] ^= FrameBitsIV;
    state_update(state, key, NROUND1);
    state[3] ^= ((unsigned int*)iv)[i];
}
```

CPA 공격 시도

- CPA는 상관 관계를 분석하는 공격 기법
- 키 값이 평문에 적용된 직전/직후를 비교
 - 피어슨 상관계수 사용
- **가장 높은 상관관계를 보이는 값이 예측 키 값**
- 특별한 사유가 없다면 **첫 라운드에 적용**하는 것이 분석에 용이함
 - 반대로 가장 마지막 라운드를 분석하는 방법도 존재

CPA 공격 시도

- 공격을 위한 코드 수정
 - 첫 라운드의 전력 값만 측정

```
//process the plaintext
for (i = 0; i < (mlen >> 2); i++)
{
    state[1] ^= FrameBitsPC;
    state_update(state, k, NROUND2);
    state[3] ^= ((unsigned int*)m)[i];
    ((unsigned int*)c)[i] = state[2] ^ ((unsigned int*)m)[i];
}

// if mlen is not a multiple of 4, we process the remaining bytes
if ((mlen & 3) > 0)
{
    state[1] ^= FrameBitsPC;
    state_update(state, k, NROUND2);
    for (j = 0; j < (mlen & 3); j++)
    {
        ((unsigned char*)state)[12 + j] ^= m[(i << 2) + j];
        c[(i << 2) + j] = ((unsigned char*)state)[8 + j];
    }
    state[1] ^= mlen & 3;
}

//finalization stage, we assume that the tag length is 8 bytes
state[1] ^= FrameBitsFinalization;
state_update(state, k, NROUND2);
((unsigned int*)mac)[0] = state[2];

state[1] ^= FrameBitsFinalization;
state_update(state, k, NROUND1);
((unsigned int*)mac)[1] = state[2];

*crlen = mlen + 8;
for (j = 0; j < 8; j++) c[mlen+j] = mac[j];
```

```
//process the plaintext
for (i = 0; i < (mlen >> 2); i++)
{
    state[1] ^= FrameBitsPC;
    unsigned int i_status;
    unsigned int t1, t2, t3, t4, feedback;

    for (i_status = 0; i_status < (NROUND2 >> 5); i_status++)
    {
        t1 = (state[1] >> 15) | (state[2] << 17); // 47 = 1*32+15
        t2 = (state[2] >> 6) | (state[3] << 26); // 47 + 23 = 70 = 2*32 + 6
        t3 = (state[2] >> 21) | (state[3] << 11); // 47 + 23 + 15 = 85 = 2*32 + 21
        t4 = (state[2] >> 27) | (state[3] << 5); // 47 + 23 + 15 + 6 = 91 = 2*32 + 27
        feedback = state[0] ^ t1 ^ ~(t2 & t3) ^ t4 ^ ((unsigned int*)k)[i_status];
        // shift 32 bit positions
        state[0] = state[1]; state[1] = state[2]; state[2] = state[3];

        if(i_status == 0)
        {
            trigger_high();
            state[3] = feedback;
            trigger_low();
            simpleserial_put('r', 16, pt);
        }
        else
            state[3] = feedback;
    }

    state[3] ^= ((unsigned int*)m)[i];
    ((unsigned int*)c)[i] = state[2] ^ ((unsigned int*)m)[i];
}
```


CPA 공격 시도

- 트레이스 캡처가 되지 않는 문제
 - Tinyjambu 전체 과정을 캡처할 경우 캡처에 성공
 - 따라서 작성한 코드 자체의 문제는 아닌 것으로 판단

Capturing traces: 100%  10/10 [00:22<00:00, 2.23s/it]

WARNING:ChipWhisperer Scope:Timeout in OpenADC capture(), no trigger seen! Trigger forced, data is invalid. Status: 0b
WARNING:ChipWhisperer Scope:Timeout in OpenADC capture(), no trigger seen! Trigger forced, data is invalid. Status: 0a

Target timed out!

향후 과제

- 캡처가 되지 않는 문제를 수정
- 공격 지점이 정확한 것인지 추가 분석
 - Tinyjambu에서 키가 영향을 주는 부분은 여러 군데가 존재
 - 현재 공격하고자 하는 지점이 정확히 상관 계수를 계산하는지 확인
- 공격이 성공한다면 대응 방안에 대해 연구
 - AVR 상에서 대응 및 최적 기법 구현

Q & A