Non-Profiled Deep Learning-Based Attacks on Schwaemm Notes

Cassi Chen & Michel Liao

June 2022

Contents

1 Background

1.1 Key Schedule

1.2 Feistel Cipher

A Feistel Cipher isn't a cipher, but rather a framework for building encryption algorithms. [Computerphile,]

1.3 Advanced Encryption Standard

Much of the literature on deep learning side channel attacks are based on the Advanced Encryption Standard (AES). Understanding AES serves as a foundation to understanding AES

2 Side-channel Attacks (SCA)

2.1 Test Vector Leakage Assessment (TVLA)

2.2 Correlation Power Analysis (CPA)

Correlation Power Analysis is a SCA that identifies the highest correlation between hypothetical traces using a hypothesis key guesses to find a subkey [Brier et al., 2004].

1. Collect traces.
2. Compute hypothetical intermediate values $(V_{i,k})_{1 \leq i \leq N}$ such that $V_{i,k} = F(d_i, k)$ for each key hypothesis $\mathbf{k}^* \in \mathcal{K}$, where F is a target function, N is the number of side-channel traces, d is known random values, $\mathbf{k}^*$ is the secret key, and $\mathcal{K}$ is the vector of all guessed keys.
    - E.g. in the case of AES, $F(d_i, \mathbf{k}^*) = Sbox(d_i \oplus \mathbf{k}^*)$.
3. Apply a leakage model to $(V_{i,k})_{1 \leq i \leq N}$ to get a series of hypothetical power consumption values $(H_{i,k})_{1 \leq i \leq N}$.
    - E.g. Hamming Weight or Hamming Distance
4. Compute the correlation between $(H_{i,k})_{1 \leq i \leq N}$ and the side-channel traces.
    - E.g. Pearson's correlation coefficient (PCC)
5. The correct subkey will be the $\mathbf{k}^*$ that gave the highest PCC.

3 Schwaemm

Schwaemm is an encryption scheme submitted as one of the 10 NIST finalists [Beierle et al., 2020].

3.1 Sparkle Permutation

The Sparkle Permutation is an SPN cryptosystem used in Schwaemm [Beierle et al., 2020].

3.1.1 ARX-box

Instead of an S-box, the Sparkle Permutation uses an ARX-box named Alzette to give the SPN cryptosystem its non-linearity [Beierle et al., 2020].

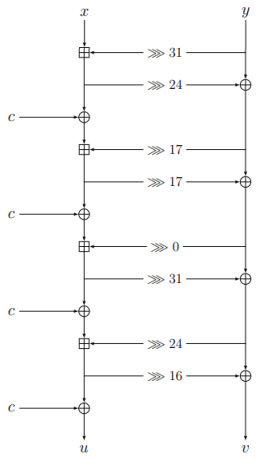

Figure 1: The Alzette instance A_c [Beierle et al., 2020].

Alzette is a 64-bit block cipher. We can understand Alzette as a four-round iterated block cipher, where each iterated performs a bitwise rotation to the left and right words and XORs the key, c, afterwards with the left word.

3.1.2 Linear Layer

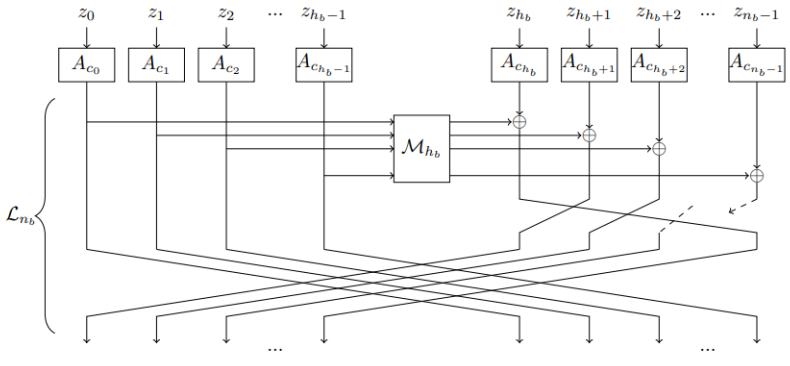

Figure 2: The overall structure of a step of Sparkle. z_i denotes the 64-bit input (x_i, y_i) [Beierle et al., 2020].

The linear layer takes in inputs $z_i = (x_i, y_i)$ and applies Alzette to them. Then, the left branches go through the permutation function $\mathcal{M}_{h_b}$ and are XORed with the right branches. Finally, the branches on the right are rotated to the left by 1 and subsequently are swapped with the branches on the left.

3.2 Schwaemm

There are four instances of Schwaemm: Schwaemm128-128, Schwaemm256-128, Schwaemm192-192, and Schwaemm256-256. Each instance takes in a given key K and nonce N to process associated data A and messages M. They will output a ciphertext C, with $|C| = |M|$ and an authentication tag T.

The big version of SPARKLE is used for initialization, separation between the processing of associated data and secret message, and finalization. The slim version of SPARKLE is for updating the intermediate state [Beierle et al., 2020].

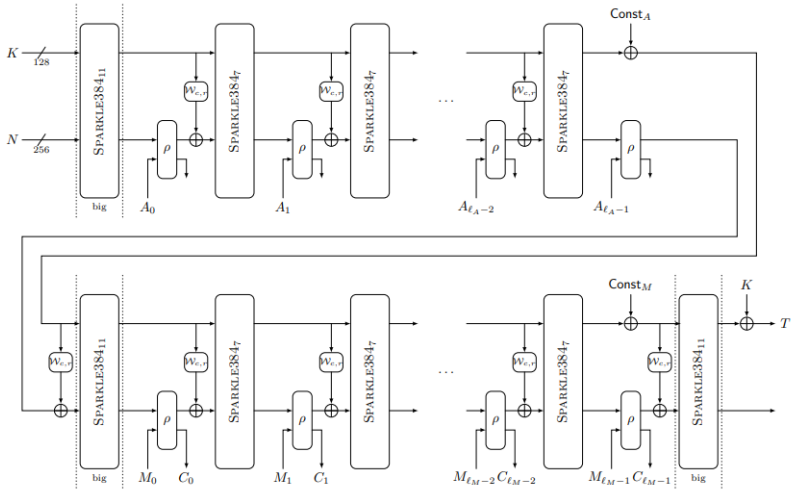

Figure 3: The AEAD Algorithm Schwaemm256-128 with $r = 256$ and $c = 128$. [Beierle et al., 2020].

We understand the feedback function to serve the purpose of an AEAD. The feedback function will add associated data to the outer state of SPARKLE in order to prevent malicious use of the cipher text from being copied and decrypted.

SPARKLE has an inner state, outer state, and intermediate state. Our understanding of rate whitening refers to XORing the nonce to function $\mathcal{M}_{c,r}$, where $\mathcal{M}$ takes in $\mathcal{S}_R$. r refers to the nonce and c refers to key. If $r = c$, $\mathcal{M}_{c,r}(\mathcal{S}_R)$ returns the XOR of the inner and outer part. We are unsure otherwise. This provides additional security because during the linear layer, half of the branches are unmodified, so attackers can use this information to learn about the permutation. However, with rate whitening, this situation

is prevented because now attackers need to have additional knowledge state [Beierle et al., 2020].

3.2.1 Algorithms

Algorithm 1 A_c
Input/Output: $(x, y) \in \mathbb{F}_2^{32} \times \mathbb{F}_2^{32}$

$x \leftarrow x + (y \ggg 31)$
$y \leftarrow y \oplus (x \ggg 24)$
$x \leftarrow x \oplus c$
$x \leftarrow x + (y \ggg 17)$
$u \leftarrow y + (x \ggg 17)$
$x \leftarrow x \oplus c$
$x \leftarrow x + (y \ggg 0)$
$y \leftarrow y \oplus (x \ggg 31)$
$x \leftarrow x \oplus c$
$x \leftarrow x + (y \ggg 24)$
$y \leftarrow y \oplus (x \ggg 16)$
$x \leftarrow x \oplus c$
return (x, y)

Algorithm 2 SPARKLE384$_{n_s}$

Require: $((x_0, y_0), ..., (x_5, y_5)), x_i, y_i \in \mathbb{F}_2^{32}$
$(c_0, c_1) \leftarrow (0xB7E15162, 0xBF715880)$
$(c_2, c_3) \leftarrow (0x38B4DA56, 0x324E7738)$
$(c_4, c_5) \leftarrow (0xBB1185EB, 0x4F7C7B57)$
$(c_6, c_7) \leftarrow (0xCFBFA1C8, 0xC2B3293D)$
for $s \in [0, n_s - 1]$ **do**
  $y_0 \leftarrow y_0 \oplus c_{(s \mod 8)}$
  $y_1 \leftarrow y_1 \oplus (s \mod 2^{32})$
  for $i \in [0, 5]$ **do**
    $(x_i, y_i) \leftarrow A_{c_i}(x_i, y_i)$
  end for
  $((x_0, y_0), ..., (x_5, y_5)) \leftarrow \mathcal{L}((x_0, y_0), ..., (x_5, y_5))$
end for
return $((x_0, y_0), ..., (x_5, y_5))$

SPARKLE384 will take in six tuples of bitstrings of length 32. Note that n_s stands for the number of **steps**, or the number of parallel applications of Alzette followed by the linear layer.

We assign the constants $c_0, c_1, \ldots, c_7$ with their respective constants in hexadecimal. For each integer in the interval $[0, n_s - 1]$, we assign y_0 as itself XORed with the round constant times the number of steps mod 8 and y_1 as itself XORed

with the number of steps mod 2^{32}. Then, we assign (x_i, y_i) to be the output of itself passed through A_c (Alzette with round constant c). We repeat this for all integers i in the interval $[0, 5]$.

Finally, we assign all of the tuples as their outputs after going through the linear layer.

Algorithm 3 SCHWAEMM256-128-ENC

Input: (K, N, A, M) where $K \in \mathbb{F}_2^{128}$ is a key, $N \in \mathbb{F}_2^{128}$ is a nonce and $A, M \in \mathbb{F}_2^*$

Output: (C, T), where $C \in \mathbb{F}_2^*$ is the ciphertext and $T \in \mathbb{F}_2^{128}$ is the authentication tag

▷ Padding the associated data and message

if $A \neq \epsilon$ **then**
  $A_0||A_1||\dots||A_{\ell_A-1} \leftarrow A$ with $\forall i \in \{0, \dots, \ell_A - 2\} : |A_i| = 256$ and $1 \leq |A_{\ell_A-1}| \leq 256$
  if $|A_{\ell_A-1}| < 256$ **then**
    $A_{\ell_A-1} \leftarrow \text{pad}_{256}(A_{\ell_A-1})$
    $\text{Const}_A \leftarrow 0 \oplus (1 \ll 2)$
  else
    $\text{Const}_A \leftarrow 1 \oplus (1 \ll 2)$
  end if
end if
if $M \neq \epsilon$ **then**
  $M_0||M_1||\dots||M_{\ell_M-1} \leftarrow M$ with $\forall i \in \{0, \dots, \ell_M - 2\} : |M_i| = 256$ and $1 \leq |M_{\ell_M-1}| \leq 256$
  $t \leftarrow |M_{\ell_M-1}$
  if $|M_{\ell_M-1}| < 256$ **then**
    $M_{\ell_M-1} \leftarrow \text{pad}_{256}(M_{\ell_M-1})$
    $\text{Const}_M \leftarrow 2 \oplus (1 \ll 2)$
  else
    $\text{Const}_M \leftarrow 3 \oplus (1 \ll 2)$
  end if
end if

▷ State initialization

$S_L||S_R \leftarrow \text{SPARKLE384}_{11}(N||K)$ with $|S_L| = 256$ and $|S_R| = 128$

▷ Processing of associated data

if $A \neq \epsilon$ **then**
  for $j = 0, \dots, \ell_A - 2$ **do**
    $S_L||S_R \leftarrow \text{SPARKLE384}_7\left((\rho_1(S_L, A_j) \oplus \mathcal{W}_{128,256}(S_R))||S_R\right)$
  end for

▷ Finalization if message is empty

  $S_L||S_R \leftarrow \text{SPARKLE384}_{11}((\rho_1(S_L, A_{\ell_A-1}) \oplus \mathcal{W}_{128,256}(S_R \oplus \text{Const}_A))||(S_R \oplus \text{Const}_A))$
end if

▷ Encrypting

if $M \neq \epsilon$ **then**
  for $j = 0, \dots, \ell_M - 2$ **do**
    $C_j \leftarrow \rho_2(S_L, M_j)$
    $S_L||S_R \leftarrow \text{SPARKLE384}_7((\rho_1(S_L, M_j) \oplus \mathcal{W}_{128,256}(S_R))||S_R))$
  end for
  $C_{\ell_M-1} \leftarrow \text{trunc}_t(\rho_2(S_L, M_{\ell_M-1}))$

▷ Finalization

  $S_L||S_R \leftarrow \text{SPARKLE384}_{11}((\rho_1(S_L, M_{\ell_M-1}) \oplus \mathcal{W}_{128,256}(S_R \oplus \text{Const}_M))||(S_R \oplus \text{Const}_M))$
end if
  return $(C_0||C_1||\dots||C_{\ell_M-1}, S_R \overset{7}{\oplus} K)$

trunc_t returns the MSB of its input.

3.3 Schwaemm Encryption Written Out

if $M \neq \epsilon$ **then**
  for $j = 0, \ldots, \ell_M - 2$ **do**
    $C_j \leftarrow \rho_2(S_L, M_j)$
    $S_L||S_R \leftarrow \text{SPARKLE384}_7((\rho_1(S_L, M_j) \oplus \mathcal{W}_{128,256}(S_R))||S_R))$
  end for
  $C_{\ell_M - 1} \leftarrow \text{trunc}_t(\rho_2(S_L, M_{\ell_M - 1}))$
  ▷ Finalization
  $S_L||S_R \leftarrow \text{SPARKLE384}_{11}\left((\rho_1(S_L, M_{\ell_M - 1}) \oplus \mathcal{W}_{128,256}(S_R \oplus \text{Const}_M))||(S_R \oplus \text{Const}_M)\right)$
end if

If the message isn't empty, then:
(1) $j = 0$**.**
Assign $C_0 = \rho_2(S_L, M_0) \approx S_L \oplus M_0$
Assign $S_L||S_R = \text{SPARKLE384}_{11}((\rho_1(S_L, M_{\ell_M - 1}) \oplus \mathcal{W}_{128,256}(S_R \oplus \text{Const}_M))||(S_R \oplus \text{Const}_M))$

Algorithm 4 SPARKLE384_{n_s}

Require: $((x_0, y_0), ..., (x_5, y_5)), x_i, y_i \in \mathbb{F}_2^{32}$
$(c_0, c_1) \leftarrow (0xB7E15162, 0xBF715880)$
$(c_2, c_3) \leftarrow (0x38B4DA56, 0x324E7738)$
$(c_4, c_5) \leftarrow (0xBB1185EB, 0x4F7C7B57)$
$(c_6, c_7) \leftarrow (0xCFBFA1C8, 0xC2B3293D)$
for $s \in [0, n_s - 1]$ **do**
  $y_0 \leftarrow y_0 \oplus c_{(s \mod 8)}$
  $y_1 \leftarrow y_1 \oplus (s \mod 2^{32})$
  for $i \in [0, 5]$ **do**
    $(x_i, y_i) \leftarrow A_{c_i}(x_i, y_i)$
  end for
  $((x_0, y_0), ..., (x_5, y_5)) \leftarrow \mathcal{L}((x_0, y_0), ..., (x_5, y_5))$
end for
return $((x_0, y_0), ..., (x_5, y_5))$

(1a) s = 0.
Assign $y_0^0 = y_0 \oplus c_{(s \mod 8)}$ and $y_1^0 = y_1 \oplus 0 \mod 2^{32}$
(1a1) i = 0.
Assign $(x_0^1, y_0^1) = A_{c_0}(x_0, y_0^0)$
(1a2) i = 1.
Assign $(x_1^1, y_1^1) = A_{c_1}(x_1, y_1^0)$
(1a3) i = 2.
Assign $(x_2^1, y_2^1) = A_{c_2}(x_2, y_2)$

(1a4) i = 3.
Assign $(x_3^1, y_3^1) = A_{c_3}(x_3, y_3)$
(1a5) i = 4.
Assign $(x_4^1, y_4^1) = A_{c_4}(x_4, y_4)$
(1a6) i = 5.
Assign $(x_5^1, y_5^1) = A_{c_5}(x_5, y_5)$
(1b) s = 1.
Assign $y_0^2 = y_0^1 \oplus c_{(s \mod 8)}$ and $y_1^2 = y_1^1 \oplus 1 \mod 2^{32}$
(1a1) i = 0.
Assign $(x_0^2, y_0^3) = A_{c_0}(x_0^1, y_0^2)$
(1a2) i = 1.
Assign $(x_1^2, y_1^3) = A_{c_1}(x_1^1, y_1^2)$
(1a3) i = 2.
Assign $(x_2^2, y_2^2) = A_{c_2}(x_2^1, y_2^1)$
(1a4) i = 3.
Assign $(x_3^2, y_3^2) = A_{c_3}(x_3^1, y_3^1)$
(1a5) i = 4.
Assign $(x_4^2, y_4^2) = A_{c_4}(x_4^1, y_4^1)$
(1a6) i = 5.
Assign $(x_5^2, y_5^2) = A_{c_5}(x_5^1, y_5^1)$
(1c-g) Repeat for $i \in [2, 7]$**.**
Then, apply the linear layer to $(x_i, y_i), i \in [0, 5]$.
(2) j = 1 Follows a similar pattern as above.

4 CPA Target Function on Schwaemm

5 Glossary

Traces - a graph of power consumption over time sample
Sample size - number of points plotted per trace
Real leakage - leakage that came from a real device
Simulated leakage - leakage from a simulation
Nonce - an arbitrary number used once

6 Questions

1. How does Schwaemm split apart K (128-bits) and N (256-bits) given that Alzette is a 64-bit block cipher? (Section 2.4, "Lightweight AEAD and Hashing using the Sparkle Permutation Family")
    - Combine the key and nonce (padding) and splits them into 64 bits.
        - A
2. What does steps mean in the Sparkle Permutation? ¡resolved¿

- How many times you loop through the ARX-box.

3. Does the entire A_0 come out of the feedback function ρ? (Section 2.4, "Lightweight AEAD and Hashing using the Sparkle Permutation Family")

   - The entire thing doesn't come out, XORs the A_0 with the previous output of the sparkle and XORs with the associated data which is then XORed with the rate whitening and then sent for the next Sparkle Permutation

4. In rate whitening, how does $\mathcal{M}_{c,r}(\mathcal{S}_R)$ work when r doesn't equal c? (Section 2.4 "Lightweight AEAD and Hashing using the Sparkle Permutation Family")

References

[Beierle et al., 2020] Beierle, C., Biryukov, A., dos Santos, L. C., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., and Wang, Q. (2020). Lightweight aead and hashing using the sparkle permutation family. *IACR Transactions on Symmetric Cryptology*, pages 208–261.

[Brier et al., 2004] Brier, E., Clavier, C., and Olivier, F. (2004). Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, pages 16–29. Springer.

[Computerphile,] Computerphile. Feistel cipher - computerphile.