Full key recovery side-channel attack against ephemeral SIKE on the Cortex-M4

Aymeric Genêt[1,2], Natacha Linard de Guertechin[3], and Novak Kaluđerović[1]

[1] École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
aymeric.genet@epfl.ch,novak.kaluderovic@epfl.ch
[2] Kudelski Group, Cheseaux-sur-Lausanne, Switzerland
[3] CYSEC SA, Lausanne, Switzerland
natacha.linard@cysec.com

Abstract. This paper describes the first practical single-trace side-channel power analysis of SIKE. While SIKE is a post-quantum key exchange, the scheme still relies on a secret elliptic curve scalar multiplication which involves a loop of a double-and-add procedure, of which each iteration depends on a single bit of the private key. The attack therefore exploits the nature of elliptic curve point addition formulas which require the same function to be executed multiple times. We show how a single trace of a loop iteration can be segmented into several power traces on which 32-bit words can be hypothesised based on the value of a single private key bit. This segmentation enables a classical correlation power analysis in an extend-and-prune approach. Further error-correction techniques based on depth-search are suggested. The attack is explicitly geared towards and experimentally verified on an STM32F3 featuring a Cortex-M4 microcontroller which runs the SIKEp434 implementation adapted to 32-bit ARM that is part of the official implementations of SIKE. We obtained a resounding 100% success rate recovering the full private key in each experiment. We argue that our attack defeats many countermeasures which were suggested in a previous power analysis of SIKE, and finally show that the well-known countermeasure of projective coordinate randomisation stops the attack with a negligible overhead.

Keywords: sike · side-channel analysis · correlation power analysis · single-trace attack · post-quantum key exchange · isogeny-based cryptography

1 Introduction

The advancement of theoretical quantum computing in the last three decades has brought algorithms which pose a threat to modern day cryptography. In particular, almost all public-key protocols that are currently used can be completely broken with Shor's algorithm [45]. Practical quantum computers seem to be lagging behind their theoretical counterparts and there are still doubts about their feasibility [29]. In order to prepare for the potential threat of quantum computers, the National Institute of Standards and Technology (NIST) published a call

for proposals for setting new standards in quantum-resistant cryptography. The proposed protocols are classical algorithms for classical computers and, as such, are prone to standard side-channel attacks, such as timing attacks, power analyses, or electromagnetic attacks. At the time of writing, the NIST standardisation process has reached the third round in which one of the alternative candidates is SIKE – "Supersingular Isogeny Key Encapsulation"; the main topic of this paper.

The development of isogeny-based cryptography protocols started in 1997 by Couveignes [16], only to be independently rediscovered in 2006 by Rostovsev and Stolbunov [43]. Their algorithm was resistant to known classical attacks, but a subexponential quantum attack was found by Childs et al. [11]. This attack was mitigated by Jao and De Feo [18] (and Plût [17]) where they proposed to use supersingular instead of ordinary elliptic curves in an algorithm called SIDH which later became SIKE. Due to the nature of supersingular elliptic curves, i.e., non-commutativity of the endomorphism ring, the previously mentioned attack is prevented. In addition to the lack of subexponential attacks in both classical and quantum settings, the new SIKE algorithm stood out in its simple structure reminiscent of the classical Diffie–Hellman protocol, but also, and more importantly, SIKE was more efficient and had lower key sizes. Over the years, SIKE was improved [3,15,49,6,7] and the current implementation stands competitive with respect to other NIST candidates in the third round. One of the main downsides of SIKE is its high run-time which currently qualifies the scheme as the slowest surviving candidate. However, this downside is compensated with the lowest key sizes among all quantum-resistant candidates. The trade-off between the cost-effectiveness of the key-size and the computational cost was studied in [34,33,40,47].

Our work follows the NIST recommendation to study side-channel attacks on post-quantum cryptographic schemes [38,1] and consists of the side-channel power analysis of the SIKE implementation adapted to the 32-bit ARM Cortex-M4 chip architecture. The Cortex-M4 implementation by Azarderakhsh et al. [44] is included in the official third round NIST submission of SIKE [27]. Both implementations are constant-time. The main differences lie in the low-level functions, such as multi-precision additions, multiplications, and modular reductions that have been rewritten in assembly in order to take full advantage of the Cortex-M4 capabilities. This allowed the authors to obtain a performance improvement of about 20× when compared to the official implementation in C. Furthermore, this improvement comes at no security cost, at least from the point of view of our attack, as the power analysis can be easily adapted to the C implementation (when run on the same microcontroller).

The Cortex-M4 [23] is a low-power and low-cost embedded microcontroller from the ARM Cortex-M family, which is recommended by NIST for post-quantum cryptography evaluation [39,31]. As such, the Cortex-M4 should be used with care in cryptographic settings. In particular, Le Corre et al. [14] have assessed the leakage on a chip from the Cortex family and have shown how the

power consumption is correlated with the operands and results from the pipeline registers. These properties will show to be useful in our own analysis.

1.1 Contributions

The main contribution of our paper is a full private key extraction using a side-channel power analysis of the Cortex-M4 implementation of SIKE with only a single trace, which therefore breaks confidentiality in a passive setting. In particular, we target the three point ladder with a straightforward vertical attack (i.e., with multiple traces and a fixed secret) and show how to extend it to the case of a horizontal attack (i.e., with a single trace and a secret which can therefore be ephemeral). Because the three point ladder is similar to an elliptic point scalar multiplication, our attack is completely analogous to a power analysis of the pre-quantum elliptic curve cryptography. This attack can be applied at any stage of the protocol: key generation, key encapsulation, and key decapsulation. Finally, we argue how our horizontal power analysis defeats many countermeasures that were mentioned in the power analysis of SIKE as presented in [50]; namely, starting with a random isomorphic curve, masking the scalar, splitting the key randomly, and using a window-based scalar multiplication. We recommend the well-known projective point coordinate randomisation, which stops our attack with a negligible performance overhead.

1.2 Related work

Side-channel analysis of supersingular isogeny protocols was initially conducted in [32] in which the authors address concerns about power analysis without carrying out a practical experiment. The first paper to practically evaluate the side-channel vulnerabilities of SIKE is due to Zhang et al. [50]. In their study, the authors fully describe a practical vertical differential power analysis on the three point ladder of the key decapsulation procedure, and discuss potential countermeasures. However, since the authors rely on the fact that the private key is fixed across the measurements, the attack is applicable only to the semi-static settings of the SIKE protocol. We extend these results and target SIKE in ephemeral settings.

In the past, many papers have already mounted horizontal attacks against the classical Montgomery ladder in the case of elliptic curve cryptography, such as [12], [41], and [4]. We apply similar techniques, but on the variant of the ladder with three points used in SIKE.

For the sake of completeness, let us also mention template attacks; a different kind of single-trace attacks in which an adversary profiles the power consumption. Such attacks have also been explored against the elliptic curve scalar multiplication, for instance in [36], [51], and [19]. As opposed to horizontal correlation power analyses, template attacks require control over the input of the targeted procedure and sometimes even further interactions with the targeted device. Online template attacks [5] against classical elliptic curve cryptography

require only one power trace of the target device, but additional power measurements on the same or a similar template device are needed. Our horizontal correlation power analysis does not rely on such a hypothesis and is executed purely offline. In comparison, our attack is based on an entirely different setup where, instead of correlating power traces with each others, we correlate Hamming weights of processed values. Our results show much stronger correlations due to the reliance on a specific leakage model, unlike the online template attack which is leakage-agnostic.

Other post-quantum algorithms have been targeted by power analyses. In a similar fashion, Aysu et al. [2] have attacked the lattice-based key exchanges of Frodo and NewHope with a horizontal correlation power analysis. Bos et al. have addressed this attack in [8] and proposed a profiled extend-and-prune approach. Recently, Sim et al. [46] have shown a single-trace ephemeral-key recovery against various lattice-based key exchanges. Finally, let us mention the work of Primas et al. [42] in which the first single-trace attack on lattice-based encryption was described using belief propagation. This work was recently extended by Kannwischer et al. [30] to a single-trace power analysis of the Keccak hash function, used in various applications, including the hash-based signature scheme SPHINCS+.

2 Background

We recall some of the definitions which will be used in this paper. For a more formal discussion the reader is advised to see [17] and [27].

2.1 SIDH – Supersingular isogeny Diffie-Hellman

Let p be a prime of form $p = l_A^{e_A} l_B^{e_B} f \pm 1$, with l_A, l_B different primes, e_A, e_B non-zero integers, and f a small cofactor. For ease of exposition we assume that $f = 1$. We define the starting curve E_0 to be a curve over $\mathbb{F}_{p^2}$ of cardinality $(p \mp 1)^2 = (l_A^{e_A} l_B^{e_B})^2$ and isomorphic to

$$E(\mathbb{F}_{p^2}) \cong E[l_A^{e_A}] \oplus E[l_B^{e_B}] \cong \langle P_A, Q_A \rangle \oplus \langle P_B, Q_B \rangle,$$

where (P_A, Q_A) and (P_B, Q_B) are bases of $E[l_A^{e_A}]$ and $E[l_B^{e_B}]$ respectively. The public parameters of the protocol are

$$(p, E_0, P_A, Q_A, P_B, Q_B).$$

The protocol itself, as the name suggests, is similar to the classical Diffie–Hellman protocol. Each of the two parties (Alice and Bob) go through two phases: the public key generation, and the shared secret key computation.

Public key generation Alice choses her private key $sk_A \in [0, l_A^{e_A})$, and computes the point $R_A = P_A + [sk_A]Q_A$. She then computes the isogeny $\phi_A : E_0 \to$

E_A of kernel $\langle R_A \rangle$. Finally she computes the images of points P_B, Q_B through ϕ_A, and sets her public key to be the triple

$$pk_A = (E_A, \phi_A(P_B), \phi_A(Q_B)).$$

Analogously, Bob sets $sk_B \in [0, l_B^{e_B})$ and computes $R_B = P_B + [sk_B]Q_B$, the isogeny $\phi_B : E_0 \to E_B$ of kernel $\langle R_B \rangle$, and $\phi_B(P_A)$ and $\phi_B(Q_A)$. His public key is

$$pk_B = (E_B, \phi_B(P_A), \phi_B(Q_A)).$$

Shared secret key computation In order to compute the shared secret, Alice computes $R'_A = \phi_B(P_A) + [sk_A]\phi_B(Q_A)$ and the isogeny $\phi'_A : E_B \to E_{BA}$ of kernel $\langle R'_A \rangle$. Bob computes $R'_B = \phi_A(P_B) + [sk_B]\phi_A(Q_B)$, and the isogeny $\phi'_B : E_A \to E_{AB}$ of kernel $\langle R'_B \rangle$. The final curves E_{BA} and E_{AB} are equal [35], and the j-invariant $j(E_{AB}) = j(E_{BA})$ constitutes the shared secret of Alice and Bob.

2.2 SIKE – Supersingular isogeny key encapsulation

The textbook SIDH protocol, as explained above, is insecure [22] in the static (i.e., the key pair of both parties is fixed) or semi-static settings (i.e., the key pair of one of the two parties is fixed). In order to overcome this weakness, the Fujisaki–Okamoto transform [21] is introduced which allows defence against known attacks at the cost of a performance overhead and losing the possibility of having fully static public keys.

The public parameters, as in SIDH, are

$$pp = (p, E_0, P_A, Q_A, P_B, Q_B).$$

For efficiency reasons, we set $l_A = 2$, $l_B = 3$, and the starting supersingular curve is selected to be

$$E_0 : y^2 = x^3 + 6x^2 + x.$$

The protocol is asymmetrical, so we will assume that Bob is the *server* and Alice is the *client*. There are three phases: the public key generation, the key encapsulation, and the key decapsulation.

Public key generation Bob starts by choosing a random string $s \in \{0,1\}^t$ ($t > 0$ public parameter) which will be used to create a random key K if he detects a cheating attempt from Alice. Then, as before, Bob chooses a random private key $sk_B \in [0, l_B^{e_B})$ and computes $R_B = P_B + [sk_B]Q_B$. After computing $\phi_B : E_0 \to E_B$ of kernel $\langle R_B \rangle$, and the images under ϕ_B of P_A and Q_A, Bob sets his public key to be

$$pk_B = (E_B, \phi_B(P_A), \phi_B(Q_A)).$$

Algorithm 1: PUBLIC KEY GENERATION

Procedure `Public key generation`(pp)

$sk_B \leftarrow [0, l_B^{e_B})$
$s \leftarrow \{0,1\}^t$
$R_B = P_B + [sk_B]Q_B$
Let $\phi_B : E_0 \rightarrow E_B$ be such that $\mathrm{Ker}(\phi_B) = \langle R_B \rangle$
Output: $pk_B = (E_B, \phi_B(P_A), \phi_B(Q_A))$

Key encapsulation Alice generates a random message $m \in \{0,1\}^t$ ($t > 0$ public parameter) which plays the role of the secret in the following. Then, Alice computes her private key by setting

$$sk_A = G(m \,||\, pk_B) \mod l_A^{e_A},$$

where G is a public cryptographic hash function (in practice, SHAKE256 is used). She proceeds by computing $R_A = P_A + [sk_A]Q_A$, the corresponding isogeny $\phi_A : E_0 \rightarrow E_A$, and the images under ϕ_A of P_B, Q_B. She sets

$$c_0 = pk_A = (E_A, \phi_A(P_B), \phi_A(Q_B)),$$

and proceeds by computing the common secret $j(E_{BA})$ and $c_1 = F(j(E_{BA})) \oplus m$, where F is also a cryptographic hash function that may or may not be different from G. Finally, Alice sends the concatenation of c_0 and c_1 as ciphertext (i.e., $ct = c_0 \,||\, c_1$) to Bob and computes the key K to be used as $K = H(m \,||\, ct)$ (where H is yet another cryptographic hash function which can be the same as G or F).

Algorithm 2: KEY ENCAPSULATION

Procedure `Key encapsulation`(pp, pk_b)

$m \leftarrow \{0,1\}^t$
$sk_A = G(m \,||\, pk_B) \mod l_A^{e_A}$
$R_A = P_A + [sk_A]Q_A$
Let $\phi_A : E_0 \rightarrow E_A$ be such that $\mathrm{Ker}(\phi_A) = \langle R_A \rangle$
$pk_A = (E_A, \phi_A(P_B), \phi_A(Q_B))$
$R'_A = \phi_B(P_A) + [sk_A]\phi_B(Q_A)$
Let $\phi'_A : E_B \rightarrow E_{BA}$ be such that $\mathrm{Ker}(\phi'_A) = \langle R'_A \rangle$
$c_0 = pk_A$
$c_1 = F(j(E_{BA})) \oplus m$
$K = H(m \,||\, ct)$
Output: $ct = (c_0 \,||\, c_1)$

Key decapsulation After receiving $ct = (c'_0 || c'_1)$, Bob sets $pk'_A := c'_0$, computes $j(E'_{AB})$, and extracts $m' = F(j(E'_{AB})) \oplus c'_1$ as shown in Algorithm 3. Bob then computes

$$sk'_A = G(m' || \, pk_B) \mod l_A^{e_A},$$

and proceeds by computing the corresponding public key pk''_A. Bob then checks that $pk''_A = pk'_A$, to confirm the truthfulness of Alice. In case the check passes, he sets $K = H(m' \,||\, ct)$, and $K = H(s \,||\, ct)$ otherwise.

Algorithm 3: KEY DECAPSULATION

Procedure `Key decapsulation`(ct)

1 $(E'_A, P'_B, Q'_B) = c'_0$
2 $R'_B = P'_B + [sk_B]Q'_B$
3 Let $\phi'_B : E'_A \to E'_{AB}$ be such that $\mathrm{Ker}(\phi'_B) = \langle R'_B \rangle$
4 $m' = F(j(E'_{AB})) \oplus c'_1$
5 $sk'_A = G(m' \,||\, pk_B) \mod l_A^{e_A}$
6 $R' = P_A + [sk'_A]Q_A$
7 Let $\phi' : E_0 \to E''_A$ be such that $\mathrm{Ker}(\phi') = \langle R' \rangle$
8 $pk''_A = (E''_A, \phi'(P_B), \phi'(Q_B))$
9 **if** $pk''_A = c_0$ **then**
  $K \leftarrow H(m' \,||\, ct)$
else
  $K \leftarrow H(s \,||\, ct)$
Output: K

2.3 Point of attack

The attack takes place at step 2 of key decapsulation which is coloured in red. This operation is computed using the "*three point ladder*". The input of the three point ladder is the public key of Alice and the execution depends on the private key of Bob. In the semi-static settings of the protocol, Bob executes the three point ladder with different inputs from different public keys of Alice (or other *client* parties) and with his own static private key. Our initial goal was to correlate the power traces from different executions of the three point ladder with the hamming weights of the corresponding public keys. This approach was successful, and we were actually able to obtain the full private key of Bob with only one power trace, i.e., from a measurement of only one communication with Alice. This allowed us to extend the attack to step 3 of Algorithm 1 and step 3 of Algorithm 2 coloured in blue, since these steps consist of the same three point ladder executed with, except for the secret keys, known inputs.

2.4 Correlation power analysis

A Correlation Power Analysis (CPA) [9] is a statistical known-text side-channel power analysis that aims to deduce a portion of a secret value across multiple power measurements. A CPA aims to use a correlation coefficient to quantify the link between power consumption and the values processed by a processing unit. In the scope of this paper, we consider two types of CPA:

- *Vertical* CPA, which targets a *fixed* secret value across different executions of the attacked algorithm by collecting *multiple* power traces that correspond to multiple executions of the *same* operation.
- *Horizontal* CPA, which targets an *ephemeral* secret value using a *single* power trace that correspond to *multiple* operations. These operations must be similar to allow the segmentation of the power trace into multiple ones to simulate a vertical CPA.

In a typical threat model for CPA, the adversary has the capability of measuring the power consumption of a target device which acts as a black-box key decapsulating device. The algorithm inputs are not required to be manipulated but are supposed to be accessible by the target device. As a result, a CPA attack is completely passive (i.e., non-intrusive) and can be mounted even during a trusted communication between two honest parties.

To assess correlation between the processed values and the power samples, the Pearson's Correlation Coefficient (PCC) is computed. Let $n > 0$ be the number of measurements, each of which consists of $S > 0$ power samples. Then, let $T(s) \in \mathbb{R}^n$ be a vector of power samples synchronised at a same instant $0 \leq s < S$, and $M \in \mathbb{N}^n$ a vector of the Hamming weight of the processed values.

$$\mathrm{PCC}(M, T(s)) = \frac{\mathrm{Cov}(M, T(s))}{\sqrt{\mathrm{Var}(M)\mathrm{Var}(T(s))}}.$$

The overall attack consists of the following steps:

1. Find an operation in the attacked procedure which involves:
   (a) A (small) portion of a secret value which is the same across all measurements.
   (b) A known input (resp. output).
   In the following, we refer to the result of this operation as the *intermediate value*.
2. Collect $n > 0$ power traces consisting of $S > 0$ power samples each, i.e., $T(s)$ for $0 \leq s < S$, that correspond to the computation of the intermediate value with different inputs (resp. outputs).
3. Take a guess for the portion of the secret value involved in the intermediate value computation.
4. Compute the vector of intermediate values from the known inputs (resp. outputs) and the secret value guess, and derive its corresponding vector of Hamming weight M.
5. For each vector of power samples at a same time, i.e., $T(s)$ for each $0 \leq s < S$, compute $\mathrm{PCC}(M, T(s))$.
   This results in a vector of PCC at each moment in time.

Using a large enough $n > 0$ given the signal-to-noise ratio of the power consumption, a strong PCC at any point in time indicates a valid guess, while a weak PCC at every point in time can rule out said guess.

Figure 1 gives a visual example of a CPA. In this example, the portion of the secret value is only one bit, resulting thus in two possible intermediate values. The PCC computation takes one of the two Hamming weight vectors M sketched on the left of the figure, and each vector of power samples at a same timing instant shown on the right, to produce each point in the corresponding PCC plot below. Since the PCC plot for the bit guess of one shows a spike, the corresponding bit for the secret value is successfully recovered.

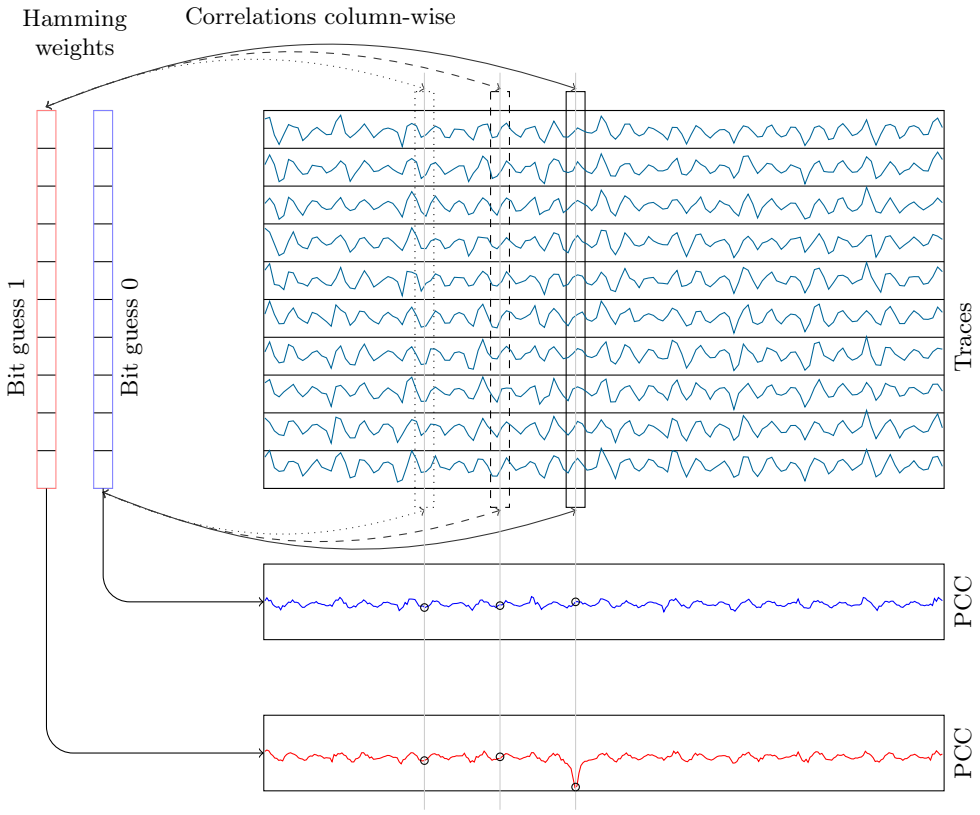

Fig. 1: Visual example of a CPA. Correlations between two arrays of Hamming weights and the power traces are plotted in the bottom. A strong correlation indicates that the bit value associated to these power traces is 1.

3 Side-channel analysis

In this chapter, we explain how to exploit the link between power consumption and processed data in order to recover private key bits.

3.1 The three point ladder

The main point of attack is the three point ladder. This is a function which takes as input an elliptic curve E and two points P and Q on that curve. These may be thought of as $pk_A = (E_A, \phi_A(P_B), \phi_A(Q_B))$ or (E_0, P_A, Q_A) etc. The three point ladder computes the point $R = P + [sk]Q$ where sk is the private key of the computing party.

Montgomery representation The curve E over the field $\mathbb{F}_{p^2}$ is represented in Montgomery representation [37] as

$$E : \beta Y^2 Z = X^3 + \alpha X^2 Z + X Z^2, \qquad \text{for some } \alpha, \beta \in \mathbb{F}_{p^2}.$$

In SIKE, we are only interested in curves where $\beta = 1$ so the curves which we work with depend on a single parameter α.

Montgomery curves allow for compact representation of points, up to sign, by using only the X and the Z coordinates. In particular, a point $S = [X_S : Y_S : Z_S] \neq [0 : 1 : 0]$ can be represented by a single field element $x_s = X_S/Z_S \in \mathbb{F}_{p^2}$. The value $[X_S : Z_S] = [x_s : 1]$ uniquely defines $\{\pm S\}$, and we write $S = [x_s : 1]$.

The triple (E, P, Q) containing a curve and two points is represented as three field elements (x_Q, x_P, x_{Q-P}), where $Q = [x_Q : 1]$, $P = [x_P : 1]$, $Q - P = [x_{Q-P} : 1]$; the coefficient α defining the curve E can be obtained from these values with a couple of modular multiplications, squarings and a single inversion.

The main ingredient of the three point ladder is a double-and-add function `xDBLADD`. It takes as input a triple of points $S, T, U \in E$ in Montgomery representation such that $U = S - T$, the curve defining coefficient α, and outputs $(2S, T + S, U)$. The ladder takes as input $Q, Q - P, P$ and computes $P + [sk]Q$ by going through the bits of sk starting from the least significant, as shown in Algorithm 4.

Algorithm 4: THREE POINT LADDER

Procedure `Three point ladder`(x_Q, x_P, x_{Q-P})

$prev\_bit = 0$
$S = [x_Q : 1]$, $T = [x_{Q-P} : 1]$, $U = [x_P : 1]$
$\alpha =$ `curve_coefficient`(S, T, U)
for $i \leftarrow 0$ **to** *bitlength*$(sk) - 1$ **do**
  $current\_bit = sk[i]$
  if $(current\_bit \neq prev\_bit)$ **then**
    `swap`(T, U)
  $(S, T, U) =$ `xDBLADD`(S, T, U, α)
  $prev\_bit = current\_bit$
if $(prev\_bit)$ **then**
  `swap`(T, U)
Output: U

The goal of the attack is to measure the power consumption of the `xDBLADD` operation and to deduce if the function was executed with or without the `swap` at step 7. We may assume that we know the private key up to bit $i-1$, by induction. We also know the starting points $Q, P, Q-P$ since they are public. Therefore, we may obtain the two possible inputs for `xDBLADD`, and we know how they relate to the value of the i^{th} bit of the private key. The two inputs and their Hamming weights are computed and the power trace of certain instructions within `xDBLADD` is correlated with the Hamming weights. Thanks to CPA, this allows us to distinguish when the i^{th} bit is zero or one.

Double-and-add Despite the involvement of a (random) bit of the private key, `xDBLADD` is a deterministic function. The inputs and outputs of each subprocedure in `xDBLADD` depend only on the original inputs of the function. As a result, an educated guess on the original inputs allow us to infer the results of all the operations involved in `xDBLADD`.

The function consists of 7 multiplications and 4 squarings of $\mathbb{F}_{p^2}$ elements, and multiple field additions, subtractions, and modular reductions. Each $\mathbb{F}_{p^2}$ multiplication and each squaring contain two multi-precision additions of $\mathbb{F}_p$ elements, referred to as "`mp_addfast`". This multi-precision addition is the operation on which our attack is focused. In total, there are $11 \times 2 = 22$ `mp_addfast` functions, out of which only 10 have inputs which differ in case of a `swap` at step 6 of the three point ladder. The code of `xDBLADD` and the squaring and multiplication functions can be found in Figure 5.

Multi-precision addition In the Cortex-M4 implementation of SIKE, the `mp_addfast` is written in assembly. The function computes the addition of two $\mathbb{F}_p$ elements. Depending on the size of p, each field element is saved in an array of $n \in \{14, 16, 20, 24\}$ 32-bit words. Each `mp_addfast` executes $2n$ load instructions (`LDMIA`), n store instructions (`STMIA`), and n additions (`ADDS`, `ADCS`). These are executed in batches of four consecutive additions, due to the limited number of available registers on the Cortex-M4. The code of the `mp_addfast` function can be found in Figure 6.

3.2 Vertical attack

In a vertical attack against SIKE, we measure multiple executions of the three point ladder in which Bob's private key is fixed, but the client public key inputs are different. From these traces, we concentrate only on a single `mp_addfast` instruction per `xDBLADD`, i.e., per bit of the private key. Within the `mp_addfast`, we can decide to focus even further on the first addition instruction. We can thus compute the two possible outputs of the first `ADDS` depending on the (timing-constant) `swap`, for each public key, and then correlate the two vectors of Hamming weights of these outputs with the power traces using the CPA procedure from Section 2.4. This process can be repeated for each bit of Bob's private key, as the correctness of each guess depends on the correctness of previous ones, resulting thus in an *extend-and-prune* attack.

3.3 Horizontal attack

In the horizontal attack scenario, we can measure only one power trace for a single execution of the three point ladder. The same approach as in the vertical attack cannot be used because there would not be enough data to obtain strong correlations. We can work out this issue and re-obtain "*verticality*" by combining the power traces of all 10 `mp_addfast` functions within each `xDBLADD`. This way, we obtain 10 power traces with which we can correlate pairs of inputs – similarly as in a vertical attack with 10 power traces.

We can further improve this attack. A multi-precision addition takes two $\mathbb{F}_p$ elements as input and gives one as output. Each one of the $2n$ 32-bit input words is loaded once and then used in the addition instruction, and the n 32-bit output words are stored. In total, there are $3n$ words which pass through the pipeline registers and whose Hamming distance from the previous word in the pipeline are related to the power consumption.

For each of the $3n$ words, we compute the PCC between the 10 power traces and the 10 pairs of hamming weights of 32-bit words accounting for the two guesses of the current bit of the private key. For each word, a spike in the correlation is expected at a different position depending on the instruction which uses this particular word. The locations of spikes can be deduced from the shape of the power traces. Once the $3n$ pairs of correlations are computed, we can add them up such that the locations of the expected spikes are aligned. We expect to end up with two correlations for each guess of the private-key bit, with a clear spike in the correlation plot of the correctly guessed value.

In presence of noise in power measurement, the private key guesses may be erroneous. A single wrong guess of a bit of the private key leads to completely inconclusive results, because the following guesses depend on the correctness of the previous bits. Therefore, it is of particular importance that no erroneous guesses are made in the process of key extraction. We propose two measures to approach this problem.

Depth search When the guess of a single bit gives inconclusive results, we can proceed by making four guesses for the next two bits in hope of finding a correlation coefficient with a notable spike. In particular we can make a guess for k consecutive bits, obtaining in total 2^k different combinations. For each combination we compute a PCC for each of the k bits. In total there are $2(2^k-1)$ correlation coefficients, not counting repetitions. We then add up all the PCCs for each k-bit combination and we guess the current sk bit to be the trailing bit of the combination with the strongest correlation.

Increasing verticality We can increase verticality (i.e., the amount of power traces in the horizontal settings) by computing correlations for bits in windows of k. If, for one bit, 10 `mp_addfast` functions can be measured from a single `xDBLADD`, then, for k bits, there will be $k \times 10$ traces of `mp_addfast` functions from the k consecutive `xDBLADD` functions. In total, 2^k hypotheses need to be

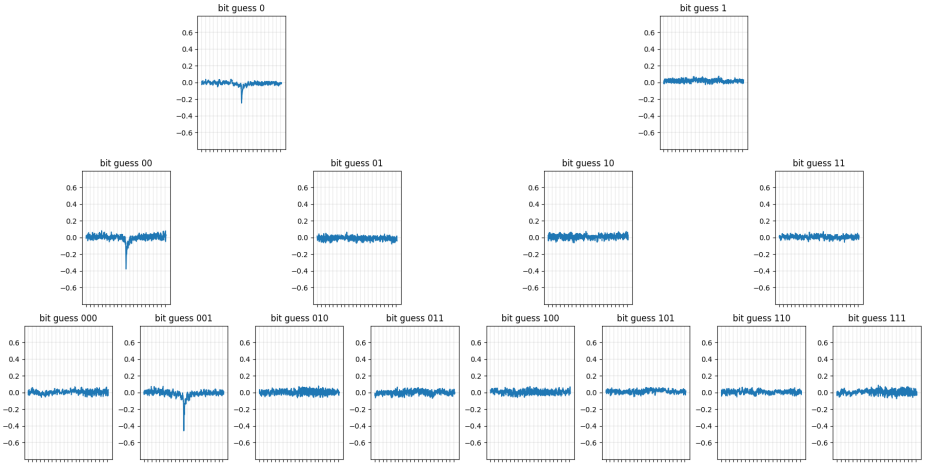

Fig. 2: Depth search.

made (one per bit), and 2^k correlation coefficients are computed for $10k$ power traces.

Finally, rather than performing the attack on contiguous windows of k bits, we select only one bit of Bob's private key to be the trailing bit of the k-bit combination with the strongest correlation. This way, we can re-run the process starting from the bit right afterwards as a way to correct errors due to the potential proximity of strong correlations. This process resembles the error-correction procedure introduced in [19].

Also, we mention that other operations, such as `fpmul_mont` and `fpsub`, can be measured and combined to increase verticality. While these are dissimilar operations and may leak information differently than `mp_addfast`, they may still add information to the overall selection of Bob's private bits.

4 Experimental results

In order to validate the horizontal attack described in Section 3, we reproduced the key recovery on a programmable board which runs an adapted version of reference implementation of SIKE [44].

4.1 Hardware setup

The experiment comprises the following equipment:

- The ChipWhisperer toolkit [25], that includes:
  - A (NAE-CW308T-)STM32F3 board which includes an ARM Cortex-M4 microcontroller (the victim).
  - A ChipWhisperer-Lite board which is solely used to communicate with the STM32F3 in serial through USB.

  - A ChipWhisperer (NAE-)CW308 UFO board which interfaces the signals between the ChipWhisperer-Lite and the STM32F3.
- A high-definition oscilloscope with the following specifications:
  - An analog bandwidth of at least 500 MHz.
  - A sampling rate of 250 samples per microsecond (i.e., 250 MS/s).
  - A resolution of 16 bits per sample.
  - A memory of 50,000 samples per acquisition.
- A general-purpose computer which runs an operating system compatible with the ChipWhisperer framework [26].

The STM32F3 is plugged into the CW308 UFO, which is itself connected to the ChipWhisperer-Lite with a 20-pin cable. The oscilloscope measures the power consumption in AC through a passive probe connected to the `SHUNTL`[4] pin on the CW308 UFO, and whose measurement is triggered by reacting to the active-high `GPIO4/TRIG`[4] pin also with a passive probe (both probes are grounded to the `GND` pins on the CW308 UFO). The computer is simply connected to the ChipWhisperer-Lite with a USB to micro-USB cable.

The reasoning behind such a setup was to overcome the limitations of memory of the ChipWhisperer-Lite by means of an oscilloscope with better specifications.

4.2 Target implementation

The attacked implementation is the official SIKE implementation adapted for (32-bit) ARM Cortex-M4 microcontrollers [44], which is part of the official submission package and is constant in timing.We attacked SIKE instantiated with a prime of 434 bits (i.e., `SIKEp434`); a choice that we elaborate in this section.

In our experiment, we wrote a small piece of software that interfaces the serial communication from the ChipWhisperer framework to the SIKE library. The code allows the computer to program the STM32F3 remotely through USB and simulate the key exchange while power consumption is measured.

Concretely, the software uses ChipWhisperer's SimpleSerial protocol [24] to program different commands to which the STM32F3 reacts. The computer uses these commands to communicate data to the STM32F3 by serially transmitting, first, the byte of the command in ASCII, then, the data of length specified for the command. When the procedure corresponding to the command ends, the STM32F3 responds with the letter `z` followed by a code returned by the procedure, which concludes the protocol exchange. Two custom commands of were introduced in the scope of this experiment – command `k` which sets Bob's private key used in the three point ladder, and the command `p` which sends Alice's public key and executes the three point ladder procedure of SIKE.

We made additional modifications in the SIKE implementation to ease the collection and the pre-processing of the traces. Note that these adjustments were made for efficiency purpose and are by no means necessary for our attack to work.

[4] We refer to the official NAE-CW308 UFO datasheet to find the mentioned pins: `http://media.newae.com/datasheets/NAE-CW308-datasheet.pdf`

In other words, we emphasise that the attack can be mounted on the original implementation of SIKE presented in [44] without any difficulty.

The list of adjustments are the following:

- A GPIO pin (`PA12`[5], a.k.a., the trigger) is toggled when the double-and-add operation of the three point ladder enters into an `mp_addfast` procedure that depends on the swap.
- An idle delay of about 1 millisecond was introduced in between each `mp_addfast` call, and of about 1 second after each loop iteration of the three point ladder.

Limitations of the software While the introduction of a trigger GPIO and multiple delays results in an unrealistic attack scenario, we emphasise on the fact that the attack is still possible on an unmodified SIKE implementation. The process of segmenting the power traces, as well as the correlation and Hamming weights computations can be done *offline*, after the power traces have been sampled. In a plain attack, as opposed to our experiment, the traces acquisition will be synchronised on serial communication. Then, the targeted operations need to be identified within the full resulting power trace (e.g., using cross-correlation techniques, as in [19]), so the sub-power traces corresponding to the attacked instructions can be manually segmented and carefully aligned to perform the CPA. This cumbersome process is not the main focus of our study and was therefore duly skipped.

Other SIKE instances To achieve various levels of security, the original SIKE submission [27] presents four different parameters sets; each of which with a prime of different size (i.e., a p with a bit-length of 434, 503, 610, and 751). While instantiating SIKE with a larger prime offers stronger security guarantees against theoretical cryptanalysis, larger instances present a wider attack surface in a single-trace power analysis. This property was also observed by Bos et al. [8], and is due to the increased number of instructions executed which, therefore, yield more power measurements. As a result, our attacked instance (`SIKEp434`) is expected to be the hardest to attack with a single trace.

Also, the compressed instances of SIKE are prone to the same horizontal attack, because the starting points of the three point ladder are deterministically obtained from the compressed public key.

4.3 Collection of traces

Our experiment simulated a portion of the SIKE key exchange between Alice (the computer) and Bob (the STM32F3); namely, the key decapsulation procedure. Our attack scenario can be summarised with the following steps:

1. On the computer, generate Bob's key pair at random, and send Bob's private key to the STM32F3 (with the command `k`).

[5] We refer to the official CW308T-STM32F3 datasheet to find the mentioned pins: `https://media.newae.com/datasheets/NAE-CW308T-STM32F_datasheet.pdf`

2. Given Bob's public key, generate Alice's key pair at random.
3. Send a public key to the STM32F3 (with the command `p`) during which the oscilloscope measures the power consumption of:
   - only the *second* `mp_addfast` call involved in steps 6 and 8,
   - and both `mp_addfast` call involved in steps 16, 17, 18, and 19,

   of the `xDBLADD` procedure (see Figure 5) as used in the three point ladder.

Once triggered, the oscilloscope was configured to sample the power consumption at a rate of 250 MS/s during a period of 20 μs. As a result, a power trace for a single execution of `mp_addfast` includes 5,000 power samples.

This attack scenario was repeated a total of 460 times to obtain at most 1 million traces. Each of these experiments includes the power traces of the 10 `mp_addfast` calls from the loop iterations for all the 217 bits of Bob's private key. Hence, $460 \times 10 \times 217 =$ 998,200 different power traces were acquired during that experiment.

For reference, Figure 3 (top) shows the average power consumption of an `mp_addfast` execution captured by our oscilloscope.

4.4 Traces polishing

Because our initial results turned out to be inconclusive due to a serious level of noise in the acquisition (see top of Figure 3), we processed the collected power traces with a denoising technique, in the hope that such a processing would increase the success rate of our CPA.

In our case, we applied a wavelet denoising compression, as initially explored in [48], to down-sample the power traces. This compression actually aims to decompose the signal into two sub-signals; approximation and details. Applied to a signal in one dimension, the approximation corresponds to the low frequencies of the signal, while the details contain the high frequencies. By keeping the approximation only, each application halves the number of samples (minus a few points due to a windowed convolution). Best results were experimentally obtained when Daubechies 3 wavelets ('db3') were used recursively three times to reduce the number of samples from 5,000 to 623. The average of the resulting traces is shown in Figure 3.

The denoised traces and public data are made accessible at `https://github.com/COSADE-anonymous-submission/SIKE-HPA-2021`.

4.5 Horizontal CPA procedure

Using the denoised power traces, we performed a horizontal CPA on each iteration of the loop in the three point ladder. Each time, a single bit of Bob's private key is attacked. This process can then be repeated across all the bits of the key.

Since a single bit is hypothesised at each step of the horizontal attack, there are only two hypotheses to consider:

- The points P and $Q-P$ were swapped (the bit is different from the previous bit).

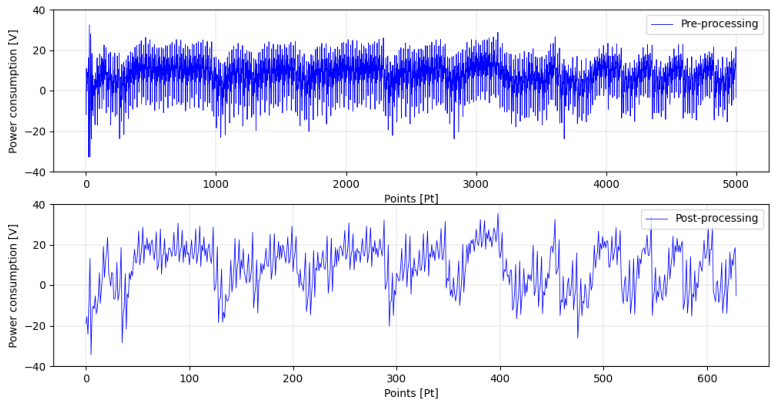

Fig. 3: Result of the discrete wavelet transform with Daubechies 3 wavelets ('db3').

– The points P and $Q - P$ were left un-swapped (the bit is the same as the previous bit).

A strong correlation between the power traces for one loop iteration and the values corresponding to one of the two hypotheses indicates the correctness of the hypothesised bit. As the attack moves forward, a successful recovery of the first bits allows the recovery of the next ones. Therefore, a full-key recovery can be incrementally mounted in an extend-and-prune manner.

Power traces segmentation Due to the ephemeral settings of the protocol, we have access to only a single trace per loop iteration involving a single bit of Bob's private key. Therefore, in order to apply a classical CPA, we need to obtain verticality, i.e., find a way to obtain a certain amount of multiple different power samples which are linked to a same portion of the private key. In our case study, we segmented the power trace that corresponds to an iteration of the three point ladder into 10 different power traces, each of which corresponding to an `mp_addfast` execution, for which, given either hypothesis, the full input and output (and thus, relevant Hamming distances information) are known. As a result, our horizontal CPA will amount to a vertical CPA with 10 power traces and 2 hypotheses.

CPA enhancements To further improve the success of our attack, we have inspected the targeted function for which the power traces were collected. Particularly, the power traces correspond to the `mp_addfast` function which adds two input $\mathbb{F}_p$ elements and returns a single $\mathbb{F}_p$ element (see Figure 6). Because, in our experiment, p is 434-bit long, each element is saved as an array of $\lceil 434/32 \rceil = 14$ words of 32 bits. This results in exactly 14 addition instructions, hence 14 leakage points, in a single `mp_addfast` power trace.

Moreover, we considered the leakage model from a Cortex-M4 microcontroller as explained in [14]. Because the power consumption leaks in the Hamming distance between the pipeline registers, we actually obtain *three* leakage points on a power trace per instruction:

(1) the Hamming distance between the first inputs of the current and the previous instruction,
(2) the Hamming distance between the second inputs of the current and the previous instruction, and
(3) the Hamming distance between the output of the current and the previous instruction.

This results in an additional segmentation of $3 \times 14 = 42$ points of leakage. For each point of leakage, a PCC is computed with the 10 `mp_addfast` power traces and the 10 Hamming distances.

We expect each of these PCCs to produce a spike at a different point in time in the correlation plot which we try to recover. The location of the spike corresponds to the position at which the associated 32-bit word is processed by a pipeline register. Each of these leakage points is constant throughout the `mp_addfast` executions and the three point ladder loop (assuming the power traces are properly aligned, which can be automated using basic peak alignment methods). These positions can even be identified by analysing the spike structure of the power trace (using, e.g., cross-correlation techniques).

Finally, the 42 PCCs at each point of leakage are added together to produce a larger spike. This consists of aligning all correlation plots on their leakage points and adding them together. We expect the difference of added correlation coefficients to be large enough to correctly validate the private bit.

4.6 Results

Among the 460 trials, our experimental results returned a resounding success rate of 100% in recovering the full key. None of the improvements described in Section 3.3 were even required. An example of the corresponding CPA is shown in Figure 4 where six bits are shown to be successfully recovered. This proof of concept shows that, even in ephemeral settings, the official ARM implementation of SIKE is vulnerable to classical power analysis techniques.

All the code used to derive our results is shared on `https://github.com/nKolja/SIKE-HPA-2021`.

5 Countermeasures

The attack arises as a consequence of the three point ladder being a deterministic function with predictable inputs. Each value going through the pipeline registers can be reduced to only two cases. These inputs depend on the public triple x_Q, x_P, x_{Q-P} (which define $Q = [x_Q : 1]$, $P = [x_P : 1]$, $Q - P = [x_{Q-P} : 1]$), the bits of Bob's private key up to the step at which the instruction in question is being executed (which we may assume to be known by induction), and the two possibilities for the current bit of the private key.

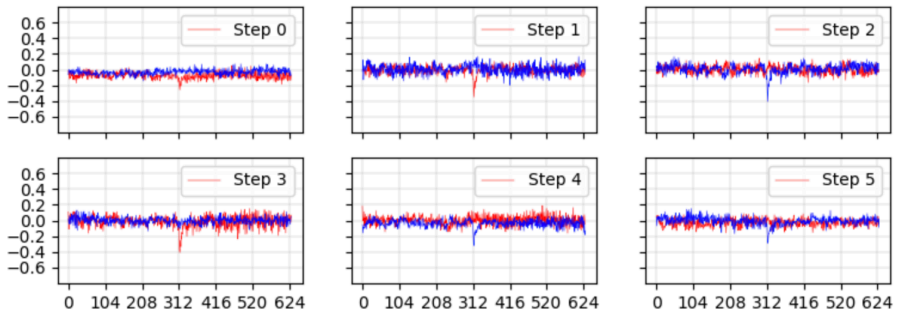

Fig. 4: Addition of shifted PCC results with 10 segments of a single power trace. Each step corresponds to a different bit. The blue curve corresponds to a bit hypothesis of zero, while the red curve corresponds to bit hypothesis of one.

5.1 Recommended countermeasure

A simple and low-cost countermeasure, which was also mentioned in [20,13,50] consists of randomising the coordinates that define the starting points, i.e., generate three random non-zero field elements r_Q, r_P, r_{Q-P} and set

$$Q = [x_Q r_Q : r_Q], \;\; P = [x_P r_P : r_P], \;\; Q - P = [x_{Q-P} r_{Q-P} : r_{Q-P}].$$

The increase in complexity comes from generating three random $\mathbb{F}_{p^2}^*$ elements and three field multiplications. This is negligible with respect to the overall cost of the three point ladder. The execution of the protocol is still correct because the points $Q, P, Q-P$ are not changed, but the input of `xDBLADD`, seen as three pairs of $\mathbb{F}_{p^2}$ elements is now randomised. Since the values r_Q, r_P, r_{Q-P} are secret, we cannot predict the loaded and stored values in the pipeline registers, and thus cannot apply the same attack anymore.

Point randomisation is in general still vulnerable to refined power analysis, as shown in [20,32]. Such power analysis constitutes in finding a point P such that one of its coordinates is 0, so that randomisation would not change this coordinate. Feeding P to the attacked device would lead to some of the coordinates being known in the computation of the ladder. However, the only points that have a zero in the X or Z coordinates are $[0 : 1 : 0]$ (i.e, the point at infinity) and $[0 : 0 : 1]$, a point of order 2. Neither of these points can be a part of a public key or an input of the three point ladder, so they can be avoided by a simple sanity check.

5.2 Other countermeasures

In addition to the randomised projective coordinates described above, the authors of [50] proposed a series of countermeasures (based on [20,28]) against CPA on SIKE that we aim to evaluate in the case of a horizontal attack. However these countermeasures are either too expensive, or do not offer additional protection against horizontal attacks. We also comment atomic elliptic curve algorithms.

1. **Masking the base point Q**
   The starting point Q is masked with a random point R in order to obtain $Q \leftarrow Q + R$. The final point $P + [sk](Q + R)$ of the three point ladder is then adjusted by subtracting $[sk]R$.
   Masking the base point prevents both a vertical and a horizontal attacks but cannot be done without leaving Montgomery representation. As a result, such a countermeasure requires at least a square root computation over the field $\mathbb{F}_{p^2}$, which is very expensive.
2. **Random isomorphic elliptic curve**
   The point Q is mapped to a random elliptic curve E' where the scalar multiplication is computed. The result is then mapped back to the original curve E in order to obtain $[sk]Q$ which is then added to P.
   Such a countermeasure is unfortunately limiting, since the number of curves of isomorphic to E is low, and finding a non-trivial isomorphism is not trivial. In particular, mapping Q to an isomorphic elliptic curve does not provide enough security against a horizontal attack due to the possibility of testing all isomorphic curves.
3. **Masking the scalar sk**
   The secret key sk is masked with a random value r by setting $sk \leftarrow sk + r \cdot \mathrm{ord}(Q)$.
   If the masking is different at each execution and big enough, the vertical attack can be conceivably prevented with this countermeasure. However, the horizontal attack is simply extended by $r \cdot \mathrm{ord}(Q)$ bits and recover a value congruent to the actual $sk \pmod{\mathrm{ord}(Q)}$. Besides, the execution of the three point ladder would be a factor of $\log(r)$ slower.
4. **Random key splitting**
   The private key sk is divided randomly as $sk = sk_1 + sk_2$. Then two three point ladders are computed in order to obtain $(P + [sk_1]Q) + [sk_2]Q$.
   While splitting sk differently across executions produces measurements of dissimilar operations in a vertical attack, this countermeasure is not effective against a horizontal attack, as both shares can be independently recovered.
5. **Window-based countermeasure**
   Instead of making a binary choice for swapping at each step of the three point ladder, a 3-bit window is used, and two additions and three doublings are computed per window.
   While a window-based method increases the complexity of a vertical attack, such a countermeasure is ineffective in the settings of a horizontal attack, as the number of guesses per CPA iteration simply increases from 2^1 to 2^3. Besides, similarly as with the base point masking, this countermeasure is not cost-efficient, as the new ladder will require to leave the Montgomery representation, requiring at least one computation of a square root over $\mathbb{F}_{p^2}$.
6. **Atomic three point ladder**
   The authors of [10] propose atomic algorithms for preventing simple side-channel analysis. An atomic algorithm is made out of a sequence of instructions which are indistinguishable from a side-channel point of view.

At the first look, the three point ladder might seem to be atomic, however the assumption in [10] that modular operations are side-channel equivalent fails in the Cortex-M4 environment. While we are not able to distinguish a single pair of modular additions with two different inputs, we are able to distinguish 10 tuples of modular additions with two different 10-tuples of inputs, which breaks indistinguishability.

6 Conclusion

The report describes a CPA on SIKE in ephemeral settings that recovers Bob's entire private key using a single power trace of the three point ladder in the key decapsulation procedure. The attack was experimentally verified on an STM32F3 which features a Cortex-M4 microcontroller in the context of the ChipWhisperer framework. A countermeasure based on point randomisation is finally suggested.

The impact of this attack on the security of SIKE is critical when the reference implementation is adapted in an unprotected manner to a Cortex-M4 microcontroller. This is especially important, because of the exceptionally leaky nature of such microcontrollers, thanks to the findings of [14]. Due to the simplicity of the CPA, countermeasures are required to be deployed when the reference implementation of SIKE is used in an embedded environment.

We emphasise on the fact that the three point ladder attacked in the key decapsulation is not the only point of attack of the SIKE protocol and that *each* use of the three point ladder (even in the key generation, and key encapsulation) requires to be protected when exposed to power analyses. Also, for future study, we propose to investigate the secret isogeny computation which is independent from the scalar multiplication.

References

1. Apon, D.: Passing the final checkpoint! NIST PQC 3rd round begins (2020), https://meetings.ams.org/math/fall2020se/meetingapp.cgi/Paper/1656, https://www.scribd.com/document/474476570/PQC-Overview-Aug-2020-NIST
2. Aysu, A., Tobah, Y., Tiwari, M., Gerstlauer, A., Orshansky, M.: Horizontal side-channel vulnerabilities of post-quantum key exchange protocols. In: 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). pp. 81–88 (2018). https://doi.org/10.1109/HST.2018.8383894
3. Azarderakhsh, R., Jao, D., Kalach, K., Koziel, B., Leonardi, C.: Key compression for isogeny-based cryptosystems. Cryptology ePrint Archive, Report 2016/229 (2016), https://eprint.iacr.org/2016/229
4. Azouaoui, M., Poussier, R., Standaert, F.: Fast side-channel security evaluation of ECC implementations - shortcut formulas for horizontal side-channel attacks against ECSM with the Montgomery ladder. In: Polian, I., Stöttinger, M. (eds.) Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11421, pp. 25–42. Springer (2019). https://doi.org/10.1007/978-3-030-16350-1_3, https://doi.org/10.1007/978-3-030-16350-1_3

5. Batina, L., Chmielewski, L., Papachristodoulou, L., Schwabe, P., Tunstall, M.: Online template attacks. J. Cryptogr. Eng. **9**(1), 21–36 (2019). https://doi.org/10.1007/s13389-017-0171-8, https://doi.org/10.1007/s13389-017-0171-8
6. Bos, J.W., Friedberger, S.J.: Arithmetic considerations for isogeny based cryptography. Cryptology ePrint Archive, Report 2018/376 (2018), https://eprint.iacr.org/2018/376
7. Bos, J.W., Friedberger, S.J.: Faster modular arithmetic for isogeny based crypto on embedded devices. Cryptology ePrint Archive, Report 2018/792 (2018), https://eprint.iacr.org/2018/792
8. Bos, J.W., Friedberger, S.J., Martinoli, M., Oswald, E., Stam, M.: Assessing the feasibility of single trace power analysis of Frodo. In: Cid, C., Jr., M.J.J. (eds.) Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11349, pp. 216–234. Springer (2018). https://doi.org/10.1007/978-3-030-10970-7_10, https://doi.org/10.1007/978-3-030-10970-7_10
9. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings. Lecture Notes in Computer Science, vol. 3156, pp. 16–29. Springer (2004). https://doi.org/10.1007/978-3-540-28632-5_2, https://doi.org/10.1007/978-3-540-28632-5_2
10. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Transactions on Computers **53**(6), 760–768 (2004). https://doi.org/10.1109/TC.2004.13
11. Childs, A., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. Journal of Mathematical Cryptology **8**(1), 1âĂŞ29 (Jan 2014). https://doi.org/10.1515/jmc-2012-0016, http://dx.doi.org/10.1515/jmc-2012-0016
12. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6476, pp. 46–61. Springer (2010). https://doi.org/10.1007/978-3-642-17650-0_5, https://doi.org/10.1007/978-3-642-17650-0_5
13. Coron, J.S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems. pp. 292–302. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
14. Corre, Y.L., Großschädl, J., Dinu, D.: Micro-architectural power simulator for leakage assessment of cryptographic software on ARM Cortex-M3 processors. Cryptology ePrint Archive, Report 2017/1253 (2017), https://eprint.iacr.org/2017/1253
15. Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for supersingular isogeny Diffie-Hellman. Cryptology ePrint Archive, Report 2016/413 (2016), https://eprint.iacr.org/2016/413
16. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
17. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology **8**(3),

209 – 247 (01 Sep 2014). https://doi.org/https://doi.org/10.1515/jmc-2012-0015, https://www.degruyter.com/view/journals/jmc/8/3/article-p209.xml
18. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Cryptology ePrint Archive, Report 2011/506 (2011), https://eprint.iacr.org/2011/506
19. Dugardin, M., Papachristodoulou, L., Najm, Z., Batina, L., Danger, J., Guilley, S.: Dismantling real-world ECC with horizontal and vertical template attacks. In: Standaert, F., Oswald, E. (eds.) Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9689, pp. 88–108. Springer (2016). https://doi.org/10.1007/978-3-319-43283-0_6, https://doi.org/10.1007/978-3-319-43283-0_6
20. Fan, J., Guo, X., De Mulder, E., Schaumont, P., Preneel, B., Verbauwhede, I.: State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures. In: 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). pp. 76–87 (2010). https://doi.org/10.1109/HST.2010.5513110
21. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology. p. 537âĂŞ554. CRYPTO '99, Springer-Verlag, Berlin, Heidelberg (1999)
22. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. pp. 63–91. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
23. Holdings, A.: Cortex-M4 specifications, https://developer.arm.com/ip-products/processors/cortex-m/cortex-m4
24. Inc., N.T.: SimpleSerial - ChipWhisperer Wiki (2017), https://wiki.newae.com/SimpleSerial
25. Inc., N.T.: CHIPWHISPERER | NewAE Technology (2021), https://www.newae.com/chipwhisperer
26. Inc., N.T.: GitHub - newaetech/chipwhisperer: ChipWhisperer - the complete open-source toolchain for side-channel power analysis and glitching attacks (2021), https://github.com/newaetech/chipwhisperer
27. Jao, D., Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Hutchinson, A., Jalali, A., Karabina, K., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Pereira, G., Renes, J., Soukharev, V., Urbanik, D.: Supersingular isogeny key encapsulation (2017), https://sike.org/
28. Joye, M., Tymen, C.: Protections against differential analysis for elliptic curve cryptography — an algebraic approach —. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems — CHES 2001. pp. 377–390. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
29. Kalai, G.: The argument against quantum computers (2019), https://arxiv.org/abs/1908.02499
30. Kannwischer, M.J., Pessl, P., Primas, R.: Single-trace attacks on Keccak. IACR Cryptol. ePrint Arch. **2020**, 371 (2020), https://eprint.iacr.org/2020/371
31. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4. Workshop Record of the Second PQC Standardization Conference (2019), https://cryptojedi.org/papers/#pqm4

32. Koziel, B., Azarderakhsh, R., Jao, D.: Side-channel attacks on quantum-resistant supersingular isogeny diffie-hellman. In: SAC (2017)
33. Kwiatkowski, K.: Towards post-quantum cryptography in TLS (2019), https://blog.cloudflare.com/towards-post-quantum-cryptography-in-tls/
34. Langley, A.: Post-quantum confidentiality for TLS (2018), https://www.imperialviolet.org/2018/04/11/pqconftls.html
35. Leonardi, C.: A note on the ending elliptic curve in SIDH. Cryptology ePrint Archive, Report 2020/262 (2020), https://eprint.iacr.org/2020/262
36. Medwed, M., Oswald, E.: Template attacks on ECDSA. In: Chung, K., Sohn, K., Yung, M. (eds.) Information Security Applications, 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5379, pp. 14–27. Springer (2008). https://doi.org/10.1007/978-3-642-00306-6_2, https://doi.org/10.1007/978-3-642-00306-6_2
37. Montgomery, P.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation **48**, 243–264 (1987)
38. Moody, D.: Let's get ready to rumble - The NIST PQC "competition" (2018), https://csrc.nist.gov/presentations/2018/let-s-get-ready-to-rumble-the-nist-pqc-competiti
39. Moody, D.: Round 2 of the NIST PQC "competition" - What was NIST thinking? (2019), https://csrc.nist.gov/presentations/2019/round-2-of-the-nist-pqc-competition-what-was-nist
40. Paquin, C., Stebila, D., Tamvada, G.: Benchmarking post-quantum cryptography in TLS. Cryptology ePrint Archive, Report 2019/1447 (2019), https://eprint.iacr.org/2019/1447
41. Poussier, R., Zhou, Y., Standaert, F.: A systematic approach to the side-channel analysis of ECC implementations with worst-case horizontal attacks. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 534–554. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_26, https://doi.org/10.1007/978-3-319-66787-4_26
42. Primas, R., Pessl, P., Mangard, S.: Single-trace side-channel attacks on masked lattice-based encryption. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 513–533. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_25, https://doi.org/10.1007/978-3-319-66787-4_25
43. Rostovtsev, A., Stolbunov, A.: A public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145 (2006), https://eprint.iacr.org/2006/145
44. Seo, H., Anastasova, M., Jalali, A., Azarderakhsh, R.: Supersingular isogeny key encapsulation (SIKE) round 2 on ARM Cortex-M4. Cryptology ePrint Archive, Report 2020/410 (2020), https://eprint.iacr.org/2020/410
45. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing **26**(5), 1484âĂŞ1509 (Oct 1997). https://doi.org/10.1137/s0097539795293172, http://dx.doi.org/10.1137/S0097539795293172
46. Sim, B., Kwon, J., Lee, J., Kim, I., Lee, T., Han, J., Yoon, H.J., Cho, J., Han, D.: Single-trace attacks on message encoding in lattice-based KEMs. IEEE Access

8, 183175–183191 (2020). https://doi.org/10.1109/ACCESS.2020.3029521, https://doi.org/10.1109/ACCESS.2020.3029521
47. Weibel, A.: Round 2 hybrid post-quantum TLS benchmarks (2020), https://aws.amazon.com/blogs/security/round-2-hybrid-post-quantum-tls-benchmarks/
48. Xavier, C., Hervé, P.: Improving the DPA attack using wavelet transform (2005), https://www.researchgate.net/publication/228717434_Improving_the_DPA_attack_using_Wavelet_transform
49. Zanon, G.H.M., Simplicio, M.A., Pereira, G.C.C.F., Doliskani, J., Barreto, P.S.L.M.: Faster isogeny-based compressed key agreement. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 248–268. Springer International Publishing, Cham (2018)
50. Zhang, F., Yang, B., Dong, X., Guilley, S., Liu, Z., He, W., Zhang, F., Ren, K.: Side-channel analysis and countermeasure design on ARM-based quantum-resistant SIKE. IEEE Transactions on Computers **69**(11), 1681–1693 (2020). https://doi.org/10.1109/TC.2020.3020407
51. Zhang, Z., Wu, L., Mu, Z., Zhang, X.: A novel template attack on wnaf algorithm of ECC. In: Tenth International Conference on Computational Intelligence and Security, CIS 2014, Kunming, Yunnan, China, November 15-16, 2014. pp. 671–675. IEEE Computer Society (2014). https://doi.org/10.1109/CIS.2014.66, https://doi.org/10.1109/CIS.2014.66

A Appendix

We include the code of the `xDBLADD`, `fp2mul_mont`, `fp2sqr_mont` and `mp_addfast` functions from [44]. Minor changes, such as variable naming, have been made to the code in order to adapt it to the names used in this paper. The lines of code `3,6,7,8,9,10,11,15,16,17,18,19` and the `mp_addfast` (highlighted in red) correspond to the targeted instructions.

```
void xDBLADD(point_proj_t Q, point_proj_t P, point_proj_t QP, const
    f2elm_t A24)
{  // Simultaneous doubling and differential addition.
   // Input: projective Montgomery points Q=(Q->X:Q->Z), P=(P->X:P->Z),
     Q-P=(QP->X:QP->Z), and Montgomery curve constant A24=(A+2)/4.
   // Output: projective Montgomery points Q <- 2*Q, and P <- Q+P.

   f2elm_t t0, t1, t2;

1  fp2add(Q->X, Q->Z, t0);
2  fp2sub(Q->X, Q->Z, t1);
3  fp2sqr_mont(t0, Q->X);
4  fp2sub(P->X, P->Z, t2);
4.5 fp2correction(t2);
5  fp2add(P->X, P->Z, P->X);
6  fp2mul_mont(t0, t2, t0);
7  fp2sqr_mont(t1, Q->Z);
8  fp2mul_mont(t1, P->X, t1);
9  fp2sub(Q->X, Q->Z, t2);
10 fp2mul_mont(Q->X, Q->Z, Q->X);
11 fp2mul_mont(t2, A24, P->X);
12 fp2sub(t0, t1, P->Z);
13 fp2add(P->X, Q->Z, Q->Z);
14 fp2add(t0, t1, P->X);
15 fp2mul_mont(Q->Z, t2, Q->Z);
16 fp2sqr_mont(P->Z, P->Z);
17 fp2sqr_mont(P->X, P->X);
18 fp2mul_mont(P->Z, QP->X, P->Z);
19 fp2mul_mont(P->X, QP->Z, P->X); //In practice 19 is outside of xDBLADD
}
```

```
void fp2mul_mont(const f2elm_t a,
    const f2elm_t b, f2elm_t c)
{   // GF(p^2) multiplication.
    // Inputs: a = a0+a1*i and b =
    b0+b1*i.
    // Output: c = c0+c1*i.
    felm_t t1, t2;
    dfelm_t tt1, tt2, tt3;
    digit_t mask;
    unsigned int i;

    mp_addfast(a[0], a[1], t1);
    mp_addfast(b[0], b[1], t2);

    fpmul_mont(a[0], b[0], c[0]);
    fpmul_mont(a[1], b[1], tt2);
    fpmul_mont(t1, t2, c[1]);

    fpsub(c[1],c[0],c[1]);
    fpsub(c[1],tt2,c[1]);

    fpsub(c[0],tt2,c[0]);
}
```

```
void fp2sqr_mont(const f2elm_t a,
    f2elm_t c)
{ // GF(p^2) squaring.
  // Inputs: a = a0+a1*i.
  // Output: c = c0+c1*i.
    felm_t t1, t2, t3;

    mp_addfast(a[0], a[1], t1);
    fpsub(a[0], a[1], t2);
    mp_addfast(a[0], a[0], t3);
    fpmul_mont(t1, t2, c[0]);
    fpmul_mont(t3, a[1], c[1]);
}
```

Fig. 5: `xDBLADD`, `fp2mul_mont` and `fp2sqr_mont` from [44].

```
void __attribute__ ((noinline, naked))
    mp_addfast(const digit_t* a, const digit_t* b
    , digit_t* c)
{ // Multiprecision addition, c = a+b.
    asm(

         "push  {r4-r9,lr}             \n\t"
         "mov r14, r2                  \n\t"

         "ldmia r0!, {r2-r5}           \n\t"
         "ldmia r1!, {r6-r9}           \n\t"

         "adds r2, r2, r6              \n\t"
         "adcs r3, r3, r7              \n\t"
         "adcs r4, r4, r8              \n\t"
         "adcs r5, r5, r9              \n\t"

         "stmia r14!, {r2-r5}          \n\t"

         "ldmia r0!, {r2-r5}           \n\t"
         "ldmia r1!, {r6-r9}           \n\t"

         "adcs r2, r2, r6              \n\t"
         "adcs r3, r3, r7              \n\t"
         "adcs r4, r4, r8              \n\t"
         "adcs r5, r5, r9              \n\t"

         "stmia r14!, {r2-r5}          \n\t"

         "ldmia r0!, {r2-r5}           \n\t"
         "ldmia r1!, {r6-r9}           \n\t"

         "adcs r2, r2, r6              \n\t"
         "adcs r3, r3, r7              \n\t"
         "adcs r4, r4, r8              \n\t"
         "adcs r5, r5, r9              \n\t"

         "stmia r14!, {r2-r5}          \n\t"

         "ldmia r0!, {r2-r3}           \n\t"
         "ldmia r1!, {r6-r7}           \n\t"


         "adcs r2, r2, r6              \n\t"
         "adcs r3, r3, r7              \n\t"

         "stmia r14!, {r2-r3}          \n\t"

         "pop  {r4-r9,pc}              \n\t"
   :
   :
   :
   );

}
```

Fig. 6: `mp_addfast` from [44].