Examining the Practical Side Channel Resilience of ARX-boxes (Extended)

Yan Yan
No affiliation
yanyansmajesty@outlook.com

Elisabeth Oswald
University of Birmingham
m.e.oswald@bham.ac.uk

ABSTRACT

Implementations of ARX ciphers are hoped to have some intrinsic side channel resilience owing to the specific choice of cipher components: modular addition (A), rotation (R) and exclusive-or (X). Previous work has contributed to this understanding by developing theory regarding the side channel resilience of components (pioneered by the early works of Prouff) as well as some more recent practical investigations by Biryukov et al. that focused on lightweight cipher constructions. We add to this work by specifically studying ARX-boxes both mathematically as well as practically. Our results show that previous works' reliance on the simplistic assumption that intermediates independently leak (their Hamming weight) has led to the incorrect conclusion that the modular addition is necessarily the best target and that ARX constructions are therefore harder to attack in practice: we show that on an ARM M0, the best practical target is the exclusive or and attacks succeed with only tens of traces. In addition, we also provide results suggesting that the modular addition may also be a vulnerable target when partition based distinguishers are applied in the side channel attacks.

KEYWORDS

ARX, side channel, correlation attack

1 INTRODUCTION

Lightweight cipher constructions have drawn a lot of interest lately, due to the boom of IoT applications and the NIST effort with regards to lightweight ciphers. Among different approaches, the (long existing) ARX (modular addition, rotation, exclusive-or) paradigm is particularly interesting due to recent results that indicate a "potential intrinsic resilience" against an important class of side channel attacks:

> *The software implementations of the three ARX designs we considered are characterized by a certain level of "intrinsic" resilience against CPA. ... These features make ARX constructions excellent candidates for the implementation of lightweight block ciphers for the IoT. ([3], Conclusion)*

One can argue that because the non-linear component in an ARX design is given by the addition modulo 2^n, it does not need to be encoded as a table lookup. Therefore it is arguable that the ARX instructions should take constant time on most platforms. When considering cache timing attacks, the absence of tables is a distinctive advantage, as stated in [13] and [9].

However, could there be indeed, such as alluded to in [3], an intrinsic resilience against the power/EM attacks. Such intrinsic resilience is greatly appealing to IoT designers as many resource constrained devices cannot afford the overheads (additional randomness, memory) required by countermeasures against side channel attacks.

We examine and challenge the idea of "intrinsic resilience" by investigating mathematical properties of ARX-boxes as well as conducting further real world side channel experiments on an implementation on a typical target platform. Our results show that the mathematical properties of the modular addition coupled with the use of its within typical ARX designs can create additional obstacles for an adversary because of the existence of indistinguishable keys as well as the issue of enumeration size. However we do provide a generic strategy of configuring a divide-and-conquer attack that overcomes these problems.

Furthermore we find that our implementation of an attack directly on the RX part of an ARX-box is very effective in practice; in fact it is considerably more effective than any attack on the presumed "better" modular addition target (we use the word "better" according to [3]). This finding may be surprising at first but it can be rationalised by the leakage characteristics of the particular device that we are using: it "amplifies" data dependent leakages if the same or highly correlated data is manipulated in consecutive clock cycles. This result might be particularly concerning as it contradicts with those appear to hold generally (such as the conclusion in [3] that the modular addition is a "better" target than the rotation or exclusive-or). Such vulnerability could be easily overlooked in practice.

Our work is motivated by a representative IoT scenario where the application utilises an ARX cipher for a low cost solution, such as standardised by the recent ISO/IEC DIS 19823-22 that regulates the usage of SPECK[1] in RFID applications. We focus our experiments on the popular ARM Cortex-M architecture, specifically a NXP LPC1114F which is based on an ARM Cortex-M0 core. This architecture features frequently in the IoT market, such as in the NXP PN7462 family, Toshiba TMPM066FWUG and ST STM32F031F4, etc.

This paper is structured as follows. We introduce notation and background in Sect.2. Then we mathematically examine the existence of indistinguishable keys of modular addition in the context of ARX designs, reflect on single-bit DPA and formulate an efficient divide-and-conquer strategy for DPA in Sect.3. We conduct practical experiments using correlation as a distinguisher (in accordance with [3]) on the modular addition as well as the rotation/XOR components of an ARX design (in Sect.5). In Sect.7 we explain the use of partition based distinguishers in attacking the modular addition and finally conclude in Sect.8.

1.1 Our Contributions

We consider this work as an extension to the existing literature in terms of a thorough study against the perceived "intrinsic side channel resilience" of ARX ciphers in a practical manner. In Section 3 we present the impact to side channel analysis induced by the mathematical properties of modular addition in ARX-Boxes. We also propose a "bit-overlapping" strategy to overcome the particular issue of explosive enumeration space in DPA-style attacks targeting the modular addition. In Section 4 we report a negative result of a standard correlation attack targeting the modular addition that supports the conclusion drawn by [3], that is, the modular addition, despite being (theoretically) the most effective target, possesses certain level of intrinsic resilience against DPA style attacks. In contrast, Section 5 reports a successful attack on the XOR and rotation part of an ARX-box which are generally considered ineffective targets in DPA attacks. We rationalised this result as the consequence of a signal amplification effect induced by the naive reference implementation. In Section 7 we demonstrate the effectiveness of partition based distinguishers in attacking the modular addition and the cause behind with successful result on real devices. We consider this work as a standout example that shows the mismatch between theory and practice. It serves a precautionary note to cipher designers as well as to the practitioners to question if theoretical assumptions are upheld by a concrete device in practice.

1.2 Experimental Setup

We use the reference C implementation of SPARX-64/128 [5] to instantiate the ARX cipher in our experiments. Our platform is an LPC1114FN28 (ARM Cortex-M0) configured to 12MHz hosted by a SCALE board [15]. All traces are collected on the full execution of 24 rounds of SPARX encryption on uniformly randomly generated plaintext. A 10ms interleave is inserted between each execution of encryption. The traces are measured by a Lecroy Wavepro 760Zi-A configured to 500 MS/s. The correlation analysis is performed on the raw traces without any preprocessing.

2 PRELIMINARIES

2.1 Notation

The notation $[x]$ refers to the binary representation of an integer x: $x = \sum_{i=0}^{n-1} 2^i [x]_i$, with $[x]_i$ denoting the i-th bit of $[x]$.

We are mostly concerned with modular addition, logical rotation (abbreviate as rotation) and exclusive-or (XOR) operations over $\mathbb{Z}_{2^n}$. We denote them as:

- $x \boxplus y$: $(x + y) \mod 2^n$
- $x >> y$: Right rotate x by y bits.
- $x << y$: Left rotate x by y bits which equates to $x >> (n-y)$.
- $x \oplus y$: Exclusive-or of x and y.

The flip function $\mathcal{F}_i(x)$ returns x with the i-th bit flipped:

$$\mathcal{F}_i(x) = x \oplus 2^i.$$

The one's complement of x (all bits are flipped) is denoted by $\widetilde{x}$:

$$\widetilde{x} = x \oplus (2^n - 1)$$

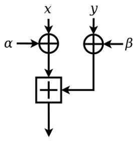


Figure 1: Generalised ARX-box $S_{\alpha,\beta}(x, y)$

2.2 ARX Constructions: a generalised ARX Sbox

The term "ARX cipher" refers to a family of ciphers that base their round function on the simple combination of modular addition($\boxplus$), rotation(>>), and XOR($\oplus$). The idea of combining these instructions as a round function has been suggested as early as 1987 in the block cipher FEAL[17]. The appeal of this construction is primarily that we can choose n equal to the word size of a processor. The fact that instructions for A, R, and X are typically available on small embedded devices further enables excellent performance both with respect to code size and energy consumption. This, plus the advantage of low memory consumption that naturally comes with the absence of look up tables, explains why the ARX paradigm has regained popularity in the context of resource constrained IoT scenarios. Recent examples for ARX constructions include Chacha20, a well studied stream cipher that has been included in RFC [14], the SKEIN hash function [6] which was a finalist of the SHA-3 competition, as well as the SPECK block cipher [1] proposed by NSA. Here we specifically address the SPARX block cipher [5] as it is the first instance that is provably secure against differential and linear cryptanalysis [5] (justifying the security of ARX ciphers is far more difficult due to its lack of S-Box and only uses modular addition for non-linear layer [4]).

The rotations, which are based on known constants, are necessary for the cryptanalytical resilience of ARX ciphers. For an adversary who uses side channels to gain additional information about intermediate variables, they can however be easily incorporated into the attack strategy and have no whatsoever impact. Hence in this paper we focus (without loss of generality) on a generalised ARX-box that omits the rotation. We thus are defining a generalised ARX-box as:

$$s = S_{\alpha,\beta}(x, y) = (x \oplus \alpha) \boxplus (y \oplus \beta). \quad (1)$$

In (1), the tuple (x, y) consists of the known inputs, and the tuple (α, β) consists of the secret keys. All variables are in the field $\mathbb{Z}_{2^n}$. Figure 1 presents the circuit that is equivalent to (1).

We argue 1 to be general because typical ARX-boxes can be reduced to this form (we provide two concrete examples in the following).

Example 2.1. In SPARX(Figure 2), the subkeys are first XOR-ed with two blocks of plaintext, then the left share is rotated, then the modular addition takes place. Denote the left and right half plaintext in Figure 2 as p_L and p_R and the subkeys as k_L and k_R, the modular sum s is:

$$\begin{aligned} s &= ((p_L \oplus k_L) >> 7) \boxplus (p_R \oplus k_R) \\ &= ((p_L >> 7) \oplus (k_L >> 7)) \boxplus (p_R \oplus k_R) \end{aligned} \quad (2)$$

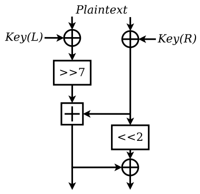


Figure 2: SPARX

To reduce from (2) to (1), we simply define:

$$\begin{cases} x = p_L >> 7 \\ \alpha = k_L >> 7 \\ y = p_R \\ \beta = k_R \end{cases} \tag{3}$$

Example 2.2. SKEIN combines the plaintext and key, denoted as p and k, by directly adding them together. The modular sum is therefore:

$$s = p \boxplus k \tag{4}$$

In this case, the generalise ARX-box is instantiated by setting:

$$\begin{cases} x = p \\ \alpha = 0 \\ y = 0 \\ \beta = k \end{cases}$$

2.3 Side Channel Attacks on ARX Ciphers

We briefly recall the assumptions and procedure of a non-profiled standard DPA style attack, followed by reflecting on implementation options and measures of success.

DPA procedure. The adversary first collects a set of leakage traces $\mathbf{T} = \{t^1, t^2, \dots\}$ with known plaintexts $\mathbf{X} = \{x^1, x^2, \dots\}$ (the device uses a fixed known algorithm, and a fixed but unknown key that is comprised of subkeys k^* that the attack extracts). For simplicity of notation we drop the superscript that indexes traces, and we refrain from introducing notation to index into individual points in traces.

Traces are assumed to be noisy (Gaussian additive noise e) and have multiple points which are treated independently in a standard DPA style attack: $\mathbf{T} = \{t^i : t = M_D(F_{k^*}(x^i)) + e^i\}$. Thus the adversary will apply to following procedure to each trace point independently.

The adversary selects a target intermediate $v = F_k(x)$ corresponding to some step F that is executed during the cryptographic algorithm, and translates this to a leakage value using a predicted leakage model L (e.g. L is typically the Hamming weight or distance, or the value of a single bit of v). A divide-and-conquer strategy enables to extract information about a subkey using a Distinguisher D suitable for the selected leakage model (e.g. correlation for models with more than two classes as outcomes, or a t-test for purely binary models). This is done by the adversary by computing a distinguishing score for each subkey candidate (the function L is applied over all inputs in $\mathbf{X}$ and assuming a specific value of k, to produce a set of hypothetical leakage values $\mathbf{L}_k$): $D_k = D(\mathbf{T}, \mathbf{L}_k)$.

If all adversarial assumptions hold (in particular the leakage model L is sufficiently close to the true device leakage) the correct key can be identified as the one with the highest distinguishing score.

Instantiation options. There are several options in instantiating an attack, in regards to different target intermediate v (and hence target function F), different prediction model L and different distinguisher D. The most commonly instantiated attacks are single bit DPA [11] and Correlation Power Analysis (CPA)[11]. Single bit DPA targets a single bit intermediate v, e.g. a selected bit of an S-Box output, with identity function $L(v) = v$ as prediction. The commonly used distinguishers for single bit DPA include Difference of Means, Distance of Means and Generalised Maximum-Likelihood Testing[11]. In comparison, correlation attacks can use target intermediates v of multiple bits, e.g. the full output of an S-Box, and the Hamming weight (HW) prediction of leakage $L(v) = HW(v)$ which has practically been proven effective in many cases [11][22], together with the absolutes value of Pearson's correlation coefficients as the distinguisher D.

Notably, the selection of target intermediate v dominates the key enumeration space and hence the computational and space complexity in launching an attack. In case of our generalised ARX-box as in Figure 1, this implies that when selecting v as all bits of the n-bit modular sum, the adversary is required to enumerate over a space size of 2^{2n} for both α and β which could be costly as n increases. We provide more details of this subject later in Sect.3.3.

Effectiveness. The effectiveness of a DPA style attack is commonly evaluated by the number of traces required to recover the key. It is generally understood that completely linear targets such as the XOR and rotation operations are difficult to attack with DPA: attacks on such targets require many more traces than attacks on highly non-linear target functions, and even with very large numbers of leakages there remains some keys that cannot be distinguished from each other [16]. This statement is further supported by [3] where the authors reported the difficulties of independent correlation attacks using HW predictions against the XOR and rotation instructions. The study [3] quantified the difficulty of attacking different instructions utilised in ARX ciphers in terms of Pearson's correlation with HW prediction on an AVR processor and concluded that even the most effective target, which is the only non-linear operation, i.e., modular addition, does not seem effective enough to mount a practical attack. The idea that attacking a XOR is necessarily more difficult than attacking a modular addition has been picked up independently also here:

> *... In this case (ARX), side-channel analysis is still possible but the XOR or modular addition selection functions are less efficient than for the Sbox case. Moreover, it has been theoretically proven that the XOR selection function is less efficient that the modular addition operations. (Section 2.2, [2])*

Attacking ARX. To date, there have been minimal successful side channel attacks on ARX ciphers under a similar setup. The most significant result to our knowledge is [22], which demonstrated that it is possible to improve on straightforward DPA style attacks when targeting the modular addition in SKEIN [6] on a 32 bit ARM Cortex-M3 processor. In [22], the authors observed that the symmetrical structure of modular addition eventually results into a pair of correlation peaks; therefore the performance of an attack can be improved by testing pairs of correlations rather than a single correlation. However, attacks like [22] requires one of the adders to be known to the adversary. It also faces the practical problem that the number of key hypotheses tested via the target function increases exponentially with the operand size. For example, in the case of a 32-bit modern processor such as an ARM M0, performing a DPA style attack requires the adversary to enumerate both 32-bits adders which has a space complexity of 2^{64}. To solve this issue, [22] assumes a stronger adversary that is capable of choosing the plaintext to be encrypted, in comparison to a classic DPA setting where the DPA adversary only passively collect traces. It has also been shown possible that straight forward DPA could succeed on 8 bit and 16 bit microcontrollers targeting the writing of the output of key XOR in [7].

3 MATHEMATICAL EXPLORATION OF THE PROPERTIES OF MODULAR ADDITION AS A DPA TARGET

Considering the fact that modular addition is the only non-linear operation in an ARX-box, it is naturally the primary target for DPA attacks as explained by [3]. However, off-the shelf DPA strategies turn out to be problematic because of the specific nature of modular addition as we now explain in detail.

3.1 Indistinguishable Keys

The first observation is that it is impossible to achieve a first order key recovery exploiting only the leakage of the modular sum, as explained by Proposition 3.1.

Proposition 3.1. *Let (α, β) be the correct key for an ARX-box. There is always another key $(\alpha', \beta') \neq (\alpha, \beta)$, such that:*

$$\forall(x, y) : S_{\alpha,\beta}(x, y) \equiv S_{\alpha',\beta'}(x, y) (\mod 2^n) \tag{5}$$

Proof. Let

$$\begin{cases} \alpha' = \mathcal{F}_{n-1}(\alpha) \\ \beta' = \mathcal{F}_{n-1}(\beta) \end{cases} \tag{6}$$

For arbitrary (x, y), it follows that

$$\begin{cases} S_{\alpha,\beta}(x, y) = (x \oplus \alpha) \boxplus (y \oplus \beta) \\ S_{\alpha',\beta'}(x, y) = (x \oplus \mathcal{F}_{n-1}(\alpha)) \boxplus (y \oplus \mathcal{F}_{n-1}(\beta)) \end{cases} \tag{7}$$

Note that $(x \oplus \alpha)$ and $(x \oplus \alpha')$ only differs at the MSB, and the same applies to $(y \oplus \beta)$ and $(y \oplus \beta')$. Hence:

$$\begin{cases} (x \oplus \alpha) = (x \oplus \alpha') \pm 2^{n-1} \\ (y \oplus \beta) = (y \oplus \beta') \pm 2^{n-1} \end{cases} \tag{8}$$

Therefore:

$$S_{\alpha,\beta}(x, y) - S_{\alpha',\beta'}(x, y) \equiv \pm 2^{n-1} \pm 2^{n-1} (\mod 2^n) \tag{9}$$

Note that:

$$-2^n \equiv 0 \equiv 2^n (\mod 2^n) \tag{10}$$

Therefore

$$S_{\alpha,\beta}(x, y) \equiv S_{\alpha',\beta'}(x, y) (\mod 2^n) \tag{11}$$

□ □

Proposition 3.1 implies that for a correct key (α, β), an incorrect key (α', β') always results in identical modular sums and consequently identical leakage distributions; therefore they cannot be distinguished from each other in any DPA style attack.

3.2 Ineffective Single Bit DPA

Our next observation is that single bit DPA is ineffective against modular addition. Observe that the i-bit of the modular sum $[s]_i$ can be represented as:

$$[s]_i = ([x]_i \oplus [\alpha]_i) \boxplus ([y]_i \oplus [\beta]_i) + c_i = [x]_i \oplus [\alpha]_i \oplus [y]_i \oplus [\beta]_i \oplus c_i \tag{12}$$

where c_i denotes the carry bit from adding the previous bits and specifically $c_0 = 0$. (12) implies that single bit key guesses $([\alpha]_i, [\beta]_i)$ and $(\widetilde{[\alpha]}_i, \widetilde{[\beta]}_i)$ are equivalent and thus cannot be distinguished from each other in a DPA attack. Consequently, applying a single bit DPA on each bit of s only recovers $\alpha \oplus \beta$ and thus only reduces the key space from 2^{2n} to 2^n, which might still be costly in practice.

3.3 The Enumeration Space and Divide-and-Conquer

When single bit DPA is not viable as we explained in Sect. 3.2, an adversary has to attack multiple bits simultaneously. Recall that determining s requires the adversary to enumerate both α and β simultaneously, which for n-bit operands quickly becomes very costly (e.g. consider SPARX [5] its 16 bit operands implies 2^{32} keys need to be enumerated).

Alternatively, a general solution to reduce the key enumeration space is a divide-and-conquer strategy that recovers (α, β) chunk-wise. Denote by $s_c, \alpha_c, \beta_c, x_c, y_c$ the l-bits chunks starting from the i-th bit of s, α, β, x, y respectively and their corresponding previous bits as $s_p, \alpha_p, \beta_p, x_p$ and y_p. Observe that:

$$s_c = (x_c \oplus \alpha_c) \boxplus (y_c \oplus \beta_c) \boxplus c_i \tag{13}$$

where c_i, the carry bit from adding the previous bits, can be expressed as:

$$c_i = \begin{cases} 0 \text{ if } (x_p \oplus \alpha_p) + (y_p \oplus \beta_p) < 2^i \\ 1 \text{ otherwise} \end{cases} \tag{14}$$

and specifically $c_0 = 0$.

(13) indeed suggests a naive approach of divide-and-conquer by recursively recovering the key bits from LSB to MSB. However, it can be easily shown that Proposition 3.1 also applies in this case. Therefore, in such a straightforward divide-and-conquer approach, all equivalent keys recovered in each chunk will have to be carried over into the next chunk, resulting in a key space that exponentially explodes with the number of chunks for each iteration.

Reviewing Proposition 3.1, we notice that the equivalent keys only differ at their MSBs; thus, attacking each chunk indeed recovers the unique lower $l - 1$ bits of (α_c, β_c). Exploiting this feature, by dropping the MSBs of (α_c, β_c) and overlapping them with the LSBs of the next chunk, one can avoid equivalent keys and uniquely recover the lower order of $l - i$ bits (α_c, β_c), except for the last chunk where there is no next chunk to be overlapped. Applying this overlapping method, we can reduce the resulting key space in a divide-and-conquer attack to 2.

4 CORRELATION ATTACKS AGAINST MODULAR ADDITION

With the observations of Sect. 3 in mind, we implemented a classic correlation attack using Pearson's correlation with HW predictions against the modular addition in accordance to [3]. We note that the optimisation proposed in [22] is not an option in our generalised ARX-box as it requires the knowledge of one of the adders, while both are unknown in our case. Our implementation targets the lowest 4 bits for a practical key enumeration space (2^8) and 2 pairs of equivalent key chunks are expected at the end of this attack as explained by Proposition 3.1. This 4-bit attack can be extended to all 16 bits of (α, β) as we explained in Section 3.3.

Simulation. We first applied the attack on traces simulated as the HW of s with additive Gaussian noise at an SNR setting of 2^1. The attack successfully recovered the key using 500 traces as we show in Figure 3a. The indistinguishable key was also observed as expected by Proposition 3.1. In addition, the simulated traces are showing multiple significant correlations on several incorrect key candidates. This can be explained by the weakly non-linear nature of modular addition. In general, consider a simplified modular addition

$$x \boxplus k = z$$

Flipping the higher bits of k leaves the lower order bits of the sum z unchanged. For instance, in an extreme case, flipping the MSB of k (or x) is a linear operation, because:

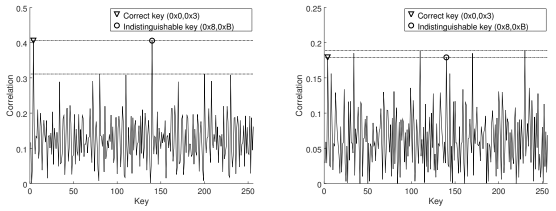
$$k' = k \oplus 2^{n-1} \implies (x \boxplus k) \oplus (x \boxplus k') = 2^{n-1}$$

that is, only the MSB of z is flipped in response.

In terms of DPA attacks, this implies that, given a set of plaintexts and a number of key guesses including k' as just described, the intermediate values related to the sum z are identical in most bits. Consequently, for typical linear leakage models, their resulting hypothetical leakages will be very close, lowering the difference in correlation between the correct key guess and "similar" incorrect key guesses.

The observation is then supported by our further simulations with SNR decreased to 2^{-3}. The attack failed with the same amount of traces, as showed in Figure 3b. Even though the correct key(s) still showed a significant correlation, an incorrect key candidate(s) showed an even higher correlation. Comparing Figure 3a to Figure 3b, we can see the difference in correlation between the correct key and the best among the others decreased significantly as we anticipated.

Real attack. We carried the same experiments out using real traces collected on the ARM Cortex-M0 executing SPARX-64/128 with the reference C implementation [18]. The SNR of this device, estimated by the method proposed in [10], is 2.823 for all 16 bits of s and 0.043 for the 4 targeted bits. Using sample sets of 2000 traces, certain points on the traces were found significantly correlating to our prediction of leakage under the correct key guess. Yet, the attacks failed due to incorrect keys are yielding at even higher correlations, reflecting the results in the above simulations.



(a) SNR=2^1 **(b) SNR=2^{-3}**

Figure 3: Correlations for SPARX modular addition using 500 simulated traces. Difference in correlation between the correct key guess and the best among the others showed in dash line.

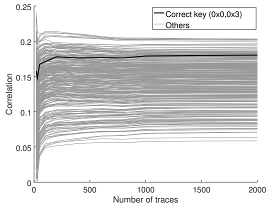


Figure 4: Evolution of correlations targeting addition

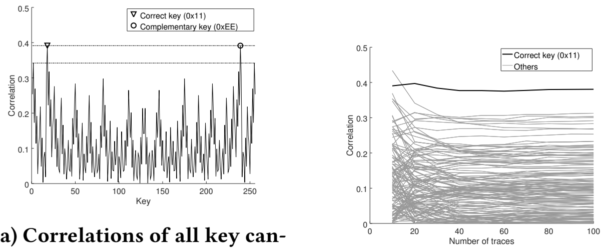
To further investigate the impact of the number of traces to the attack, we extracted the time points with the highest correlation to the correct key. We plotted the evolution of correlations (taking the average of 20 repeated experiments) of all key candidates in Figure 4. It suggests that the correlations indeed stabilise within a few hundred of traces; and hence more traces will not make the attack successful.

From a practical aspect, the mismatch between the device leakage model and the used HW model as the adversarial power model could be another potential cause for the failure of the attack. However, in a non-profiling scenario, the adversary is unable to acquire an accurate prediction model for the correlation attack and thus has to rely on a classic model, typically the HW, which we have shown to be ineffective in our experiments. Therefore we consider our results so far as evidence of the claimed "intrinsic resilience" of ARX ciphers stated in [3].

5 CORRELATION ATTACKS AGAINST ROTATION AND XOR

We further extended our experiments to the assumed to be more "difficult" targets in ARX-boxes described in [3], which are the linear operations (XOR and rotation). For a fair comparison with the experiments in Sect.4 in terms of key space, we selected the target intermediate to be the 8 lower-order bits of $(x \oplus \alpha)$ of SPARX. Note that rotation has no effect in changing the HW of the operand; hence XOR and rotation share the same predicted leakage in the HW model.

The correlation attack is again implemented with HW predictions and applied to the same real traces collected during the experiments of Sect.4. To our initial surprise, the result contradicts



(a) Correlations of all key candidates. Difference in correlation between the correct key guess and the best among the others showed in dash line.

(b) Evolution of correlations over number of traces

Figure 5: Correlation attack against XOR and rotation on real traces

those reported by [3]. The key was successfully recovered with a low number of traces (Figure 5). The symmetry in Figure 5a is due to the fact that:

$$HW(\tilde{x} \oplus \alpha) = HW(\widetilde{x \oplus \alpha}) = n - HW(x \oplus \alpha) \qquad (15)$$

where $n = 8$ is the size of the target intermediate in bits. (15) implies that complemented keys α and $\tilde{\alpha}$ result in HW leakage predictions complemented modulo n and thus the same absolute value of correlation. We also present the evolution of correlations (taken over 20 repeat experiments) at point of highest correlation over different number of traces in Figure 5b. The result suggests that the correlation of the correct key reaches a clear distinction against others for only tens of traces on this device.

What is then the reason that this practical result seemingly contradicts the existing theory? We argue that an explanation can be found when challenging an (unexamined) assumption in previous work. This assumption is omnipresent both in theoretical papers like [16] as well in practical results [3]: it is the assumption that devices leak the Hamming weight of just one operand irrespective of the previous (or next) data to be processed. The device which we are using, a completely standard ARM M0, by large fits this leakage model. A very detailed exploration of its leakage characteristics was recently published as a (side result) in [12]. Their work derived not only instruction-level leakage models, but it also hinted towards a "signal amplification" effect. This effect occurs when consecutive instructions operate on the data: latter instructions exhibit a considerably stronger signal than earlier instructions (on the same data). Because of this effect, the attack on the exclusive-or benefits from a significantly better signal-to-noise ratio, and therefore does perform much better in practice than the attack on the modular addition.

To check if our suspicion was correct, we examined the code that implements the ARX-box in our experiments, see Figure 6a (this is the SPARX reference implementation). Since rotation is not supported in standard C, it is naturally implemented as a combination of arithmetic shift and OR operations as in the macro ROTL of Figure 6a. When compiled with ARM toolchain arm-none-eabi 6.3.1, it is translated into the following assembly[1] (Figure 6b). We specifically marked the operands that are potentially leaking our intended leakage $HW(x \oplus \alpha)$ in Figure 6b. Clearly consecutive instructions are indeed invoking operands containing the targeted leakage; therefore the signal amplification occurs when this code is executed, making the instructions very vulnerable against a DPA attack. Note that the native rotation instructions cannot be directly invoked to implement the ARX-box when the word sizes are not matched, for example in this case where M0 has only the 32bits ROR instruction whilst SPARX requires 16bits rotations. To handle this, software implementations inevitably need to pad the register before rotation. The padding again results into consecutive instructions over the same operands; thus produces an amplified leakage consequently.

```
//Rotate left for 16 bit registers.
#define ROTL(x,n) (((x)<<n)|((x)>>(16-(n))))

static void A(uint16_t * l, uint16_t * r)
{
        (*l) = ROTL((*l), 9);
        (*l) += (*r);
        (*r) = ROTL((*r), 2);
        (*r) ^= (*l);
}

static void sparx_encrypt(uint16_t * x,
 uint16_t k[][2 * ROUNDS_PER_STEPS])
{
        //Unrelated code omitted.
        //Key XOR.
        x[2 * b] ^= k[N_BRANCHES * s + b][2 * r];
        x[2 * b + 1] ^= k[N_BRANCHES * s + b][2 * r + 1];
        //Rotation and Addition.
        A(x + 2 * b, x + 2 * b + 1);
}
```

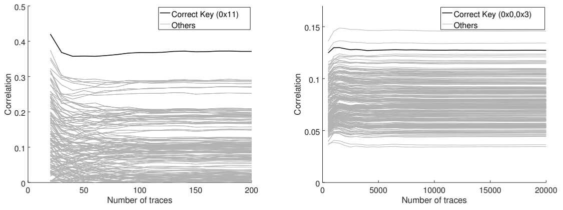
(a) Reference C implementation of SPARX [18]

```
@Key XOR
eors    r1, r2

@Rotation
lsrs    r2, r1, #7
lsls    r1, r1, #9
orrs    r1, r2

@Modular addition
adds    r1, r0, r2
```

(b) Assembly of SPARX ARX-box. Operands with potential leakage $HW(x \oplus \alpha)$ marked in bold underline

Figure 6: Code fragments of reference SPARX implementation

[1] The functions and macros are inlined by the compiler.



(a) XOR **(b) ADD**

Figure 7: Evolution of correlations with different target function on M3

6 COMPARING EXPERIMENTS ON M3

For comparison, we further conducted the same experiments on a different ARM Cortex-M3 platform instantiated by NXP LPC1313F, using the same reference C implementation shown in Figure 6a. Similar results were observed as we show in Figure 7. The correlation attack targetting the RX part succeeded within tens of traces, whereas for addition it failed with up to 20000 traces due to false positives on other key candidates, as shown in Figure 7a and Figure 7b respectively.

The relevant assemblies (Figure 8) changed as we switched the platform from M0 to M3[2]. Similar to Figure 6b, the 16bits rotation on a 32bits platform was again implemented as consecutive instructions manipulating the same operands with potentially the same leakage $HW(x \oplus \alpha)$. Further more, due to key addition being done by the outer function, an additional load instruction (ldrh) was performed beforehand which we expect to further amplifies the leakage, as memory instructions are reportedly more vulnerable against correlation attacks according to [3] and [12]. On the other hand, even though the modular sum has been additionally written back into the memory by a store instruction (strh), the leakage was not yet enough to recover the correct key in a straightforward correlation attack.

7 EFFECTIVE ATTACK AGAINST MODULAR ADDITION

Whilst the results in Section 4 suggests that correlation analysis (with Hamming weight predictions) did not succeed in recovering the key, our further observation suggests that the partition based distinguishers[20] could result into effective attacks against the modular addition. A few examples of this kind of distinguishers include Mutual Information(MI)[8], Kolmogorov-Smirnov(KS)[19] and Kruscal-Wallis(KW)[21].

Firstly, partition based distinguishers make less presumptions towards the predicted leakage model. This property greatly mitigates the factor of mismatched leakage model in the unsuccessful correlation analysis.

Secondly, a major prerequisite to apply the partition based distinguishers is that the target function must not be bijective. Since the n-bits modular addition is a mapping of:

$$\boxplus : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2^n$$

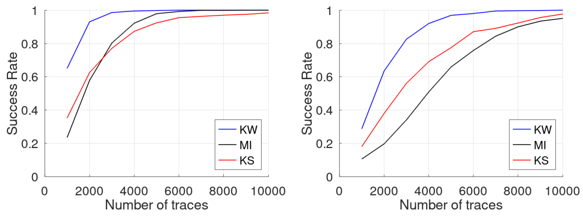
which naturally is compressive and thus non-bijective; therefore making the modular addition a nice fit without the need to preprocess the traces with techniques such as the "bit dropping trick" in [8].

We applied MI, KS and KW distinguishers on the same data set we used in correlation analysis on our targeted M0 and M3 in Figure 4 and Figure 7. The results are shown in Figure 9 where partition based distinguishers achieved a success rate of key recovery above 0.5 after 2000 and 4000 traces respectively, with the KW distinguisher being more trace efficient than the others.

The result supports our speculation that the partition based distinguishers are well suited in the attack against the ARX modular addition. It also indicates that the previous unsuccessful correlation

```
<A>:
ldrh    r2, [r0, #0]
@Rotation
lsrs    r3, r2, #7
orr.w   r3, r3, r2, lsl #9
uxth    r3, r3
strh    r3, [r0, #0]
ldrh    r2, [r1, #0]
@Modular Addition
add     r3, r2
strh    r3, [r0, #0]
...

<sparx_encrypt>:
@Key XOR
eors    r3, r1
...
strh    r3, [r6, #2]
mov     r1, r2
str     r2, [sp, #4]
@Invoke ARX-Box
bl      88 <A>
cmp.w   r8, #12
ldr     r2, [sp, #4]
...
```

Figure 8: Assembly of SPARX ARX-Box on M3. Operands with potential leakage $HW(x \oplus \alpha)$ marked in bold underline



(a) M0 **(b) M3**

Figure 9: Attack modular addition with partition based distinguishers on M0 and M3 with 1000 repetitions

[2] Instruction set also switched from Thumb to ARM.

analysis with Hamming weight leakage in Section 4 should not interpreted as an evidence of the “intrinsic resilience” of modular addition towards side channel attacks. Instead, our result clearly shows there is exploitable leakage which the Hamming weight prediction failed to capture in correlation analysis on our targeted devices.

8 SUMMARY

This work examines the intuition that ARX ciphers have intrinsic resilience against side channel attacks because of the absence of strong S-Boxes.

We first show and discuss that an adversary targeting the “stronger” modular addition must deal with the issues of indistinguishable keys and enumeration size. However we provide a mitigation by utilising a bit-overlapping trick, which we proposed. We furthermore ran attacks (simulated and real using a very popular ARM M0 architecture) on the modular addition which show that results from previous work apply also to the settings that we chose.

However we did not stop there and also investigated attacks on the XOR and rotation operations. Despite the fact that these operations are linear and thus are generally considered “bad” targets in side channel attacks, we found they could be easily broken in a straightforward correlation attack on real traces. In fact they were “better” targets than the modular addition. Investigating the source code suggested that the vulnerability is most likely caused by a signal amplification effect that is typical for our target platforms. This shows that previous work, which is based on the assumption that devices purely leak the Hamming weight of an intermediate value, is already too simplistic for platforms such as an M0 or an M3. Therefore seemingly device independent conclusions (such as that the modular addition is a better target than an exclusive or) have to be treated with a degree of scepticism. In addition, through the application partition based distinguishers even the modular addition can be broken within a reasonable amount of traces.

In conclusion, we argue that the “intrinsic resilience against side channel attacks” of ARX ciphers should not be taken for granted and implementing robust countermeasures is a must.

9 ACKNOWLEDGEMENTS AND DISCLAIMER

This work was in part supported by EPSRC via grant EP/N011635/1 (LADA). No research data was created for this paper.

REFERENCES

[1] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. 2015. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. ACM, 175:1–175:6. https://doi.org/10.1145/2744769.2747946

[2] Olivier Benoît and Thomas Peyrin. 2010. Side-Channel Analysis of Six SHA-3 Candidates. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings (Lecture Notes in Computer Science)*, Stefan Mangard and François-Xavier Standaert (Eds.), Vol. 6225. Springer, 140–157. https://doi.org/10.1007/978-3-642-15031-9_10

[3] Alex Biryukov, Daniel Dinu, and Johann Großschädl. 2016. Correlation Power Analysis of Lightweight Block Ciphers: From Theory to Practice. In *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings (Lecture Notes in Computer Science)*, Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider (Eds.), Vol. 9696. Springer, 537–557. https://doi.org/10.1007/978-3-319-39555-5_29

[4] Alex Biryukov and Léo Perrin. 2017. State of the Art in Lightweight Symmetric Cryptography. *IACR Cryptology ePrint Archive* 2017 (2017), 511. http://eprint.iacr.org/2017/511

[5] Daniel Dinu, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Johann Großschädl, and Alex Biryukov. 2016. Design Strategies for ARX with Provable Bounds: Sparx and LAX. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I (Lecture Notes in Computer Science)*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.), Vol. 10031. 484–513. https://doi.org/10.1007/978-3-662-53887-6_18

[6] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. 2010. The Skein hash function family. *Submission to NIST (round 3)* 7, 7.5 (2010), 3.

[7] Hasindu Gamaarachchi, Harsha Ganegoda, and Roshan Ragel. 2017. Breaking Speck cryptosystem using correlation power analysis attack. *Journal of the National Science Foundation of Sri Lanka* 45, 4 (2017).

[8] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. 2008. Mutual Information Analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings (Lecture Notes in Computer Science)*, Elisabeth Oswald and Pankaj Rohatgi (Eds.), Vol. 5154. Springer, 426–442. https://doi.org/10.1007/978-3-540-85053-3_27

[9] S. V. Dilip Kumar, Sikhar Patranabis, Jakub Breier, Debdeep Mukhopadhyay, Shivam Bhasin, Anupam Chattopadhyay, and Anubhab Baksi. 2017. A Practical Fault Attack on ARX-Like Ciphers with a Case Study on ChaCha20. In *2017 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2017, Taipei, Taiwan, September 25, 2017*. IEEE Computer Society, 33–40. https://doi.org/10.1109/FDTC.2017.14

[10] Stefan Mangard. 2004. Hardware Countermeasures against DPA ? A Statistical Analysis of Their Effectiveness. In *Topics in Cryptology - CT-RSA 2004, The Cryptographers’ Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings (Lecture Notes in Computer Science)*, Tatsuaki Okamoto (Ed.), Vol. 2964. Springer, 222–235. https://doi.org/10.1007/978-3-540-24660-2_18

[11] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2007. *Power analysis attacks - revealing the secrets of smart cards*. Springer.

[12] David McCann, Carolyn Whitnall, and Elisabeth Oswald. 2016. ELMO: Emulating Leaks for the ARM Cortex-M0 without Access to a Side Channel Lab. *IACR Cryptology ePrint Archive* 2016 (2016), 517. http://eprint.iacr.org/2016/517

[13] Nicky Mouha. [n. d.]. https://www.cosic.esat.kuleuven.be/ecrypt/courses/albena11/slides/nicky_mouha_arx-slides.pdf

[14] Y. Nir and A. Langley. 2015. ChaCha20 and Poly1305 for IETF Protocols. RFC 7539 (Informational). http://www.ietf.org/rfc/rfc7539.txt

[15] D. Page. [n. d.]. SCALE: Side-Channel Attack Lab. Exercises. http://www.github.com/danpage/scale

[16] Emmanuel Prouff. 2005. DPA Attacks and S-Boxes. In *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*. 424–441.

[17] Akihiro Shimizu and Shoji Miyaguchi. 1988. FEAL - Fast Data Encipherment Algorithm. *Systems and Computers in Japan* 19, 7 (1988), 20–34. https://doi.org/10.1002/scj.4690190703

[18] SPARX [n. d.]. https://www.cryptolux.org/index.php/SPARX.

[19] Carolyn Whitnall, Elisabeth Oswald, and Luke Mather. 2011. An Exploration of the Kolmogorov-Smirnov Test as a Competitor to Mutual Information Analysis. In *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers (Lecture Notes in Computer Science)*, Emmanuel Prouff (Ed.), Vol. 7079. Springer, 234–251. https://doi.org/10.1007/978-3-642-27257-8_15

[20] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. 2014. The Myth of Generic DPA…and the Magic of Learning. In *Topics in Cryptology – CT-RSA 2014*, Josh Benaloh (Ed.). Springer International Publishing, Cham, 183–205.

[21] Yan Yan, Elisabeth Oswald, and Arnab Roy. 2023. Not Optimal but Efficient: A Distinguisher Based on the Kruskal-Wallis Test. In *Information Security and Cryptology - ICISC 2023 - 26th International Conference on Information Security and Cryptology, ICISC 2023, Seoul, South Korea, November 29 - December 1, 2023, Revised Selected Papers, Part I (Lecture Notes in Computer Science)*, Hwajeong Seo and Suhri Kim (Eds.), Vol. 14561. Springer, 240–258. https://doi.org/10.1007/978-981-97-1235-9_13

[22] Michael Zohner, Michael Kasper, and Marc Stöttinger. 2012. Butterfly-Attack on Skein’s Modular Addition. In *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings (Lecture Notes in Computer Science)*, Werner Schindler and Sorin A. Huss (Eds.), Vol. 7275. Springer, 215–230. https://doi.org/10.1007/978-3-642-29912-4_16