IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2021, No. 4, pp. 36–73. DOI:10.46586/tosc.v2021.i4.36-73

Perfect Trees: Designing Energy-Optimal Symmetric Encryption Primitives

Andrea Caforio[1], Subhadeep Banik[1], Yosuke Todo[2], Willi Meier[3], Takanori Isobe[4,5], Fukang Liu[4] and Bin Zhang[6,7,8,9]

[1] Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland,
{andrea.caforio,subhadeep.banik}@epfl.ch
[2] NTT Social Informatics Laboratories, Tokyo, Japan, yosuke.todo.xt@hco.ntt.co.jp
[3] University of Applied Sciences and Arts Northwestern Switzerland (FHNW), Windisch, Switzerland willimeier48@gmail.com
[4] University of Hyogo, Kobe, Japan takanori.isobe@ai.u-hyogo.ac.jp, liufukangs@163.com
[5] National Institute of Information and Communications Technology (NICT), Tokyo, Japan
[6] TCA Laboratory, SKLCS, Institute of Software, Chinese Academy of Sciences, Beijing, China
martin_zhangbin@hotmail.com
[7] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing, 100878, China
[8] University of Chinese Academy of Sciences, Beijing, 100049, China
[9] Guizhou Shujubao Network Technology Co. Ltd, Guizhou, China

Abstract. Energy efficiency is critical in battery-driven devices, and designing energy-optimal symmetric-key ciphers is one of the goals for the use of ciphers in such environments. In the paper by Banik et al. (IACR ToSC 2018), stream ciphers were identified as ideal candidates for low-energy solutions. One of the main conclusions of this paper was that Trivium, when implemented in an unrolled fashion, was by far the most energy-efficient way of encrypting larger quantity of data. In fact, it was shown that as soon as the number of databits to be encrypted exceeded 320 bits, Trivium consumed the least amount of energy on STM 90 nm ASIC circuits and outperformed the Midori family of block ciphers even in the least energy hungry ECB mode (Midori was designed specifically for energy efficiency).
In this work, we devise the first heuristic energy model in the realm of stream ciphers that links the underlying algebraic topology of the state update function to the consumptive behaviour. The model is then used to derive a metric that exhibits a heavy negative correlation with the energy consumption of a broad range of stream cipher architectures, i.e., the families of Trivium-like, Grain-like and Subterranean-like constructions. We demonstrate that this correlation is especially pronounced for Trivium-like ciphers which leads us to establish a link between the energy consumption and the security guarantees that makes it possible to find several alternative *energy-optimal* versions of Trivium that meet the requirements but consume less energy. We present two such designs Trivium-LE(F) and Trivium-LE(S) that consume around 15% and 25% less energy respectively making them the to date most energy-efficient encryption primitives. They inherit the same security level as Trivium, i.e., 80-bit security. We further present Triad-LE as an energy-efficient variant satisfying a higher security level. The simplicity and wide applicability of our model has direct consequences for the conception of future hardware-targeted stream ciphers as for the first time it is possible to optimize for energy during the design phase. Moreover, we extend the reach of our model beyond plain encryption primitives and propose a novel energy-efficient message authentication code Trivium-LE-MAC.

Keywords: Lightweight Cryptography · Stream Cipher · Hardware · Low Energy Encryption · Trivium · Grain · Subterranean

Received: 2021-09-01 Accepted: 2021-11-01 Published: 2021-12-03

1 Introduction

Energy efficiency has become an eminent research discipline particularly in the context of lightweight cryptography [BBI+15, BBR16, BMA+18, BDE+13, KDH+12]. Low-energy consuming encryption solutions are critical, for example, in battery-driven devices that run on tight budgets like portable devices, medical implants, sensor nodes or active RFID tags. Power and energy are correlated parameters, as energy is essentially the time integral of power, and power is simply the rate of energy consumption. In a nutshell, energy is a measure of the total electrical work done by the battery source during the execution of any operation, i.e.,

$$E = \int P\,dt.$$

Hence, a less energy-hungry operation drains the battery less and is important for applications that run on tight energy budgets.

Power/energy consumed in semiconductor circuits come from two principal sources: dynamic and static. Static power is accounted for by the leakage current and other current drawn continuously from the power supply. This type of power is generally not dependent on the frequency of the clock driving the circuit. Dynamic power, on the other hand, is due to the charging and discharging of load capacitances in CMOS circuits. Each $0 \to 1 \,/\, 1 \to 0$ transition contributes to the dynamic dissipation, and hence this component varies directly as the clock frequency. Since energy consumed in an operation is roughly equal to the product of the average power and the time taken for it, this implies that the leakage energy increases with any increase in the physical time required to do a task (which can occur if we lower the clock frequency). Dynamic energy, on the other hand, would by a similar logic be independent of the frequency of the signal clocking the circuit. In this framework, there have been numerous previous works that have investigated the energy efficiency of block ciphers. In [BDE+13, KDH+12], an evaluation of several lightweight block ciphers with respect to various hardware performance metrics, with a particular focus on the energy cost, was done. In [BBR16], the authors looked at design strategies like serialization and round unrolling and the effect it has on the energy consumption required to encrypt a single block of data. They concluded that in a low-leakage environment, at high enough frequencies, the energy consumed for encrypting one block of plaintext was actually independent of the clock frequency of the circuit, (the authors of [KDH+12] also had independently come to the same conclusion). This is because if the leakage power is low, then the lion's share of the energy consumption is due to the dynamic component, which is basically given by the sum total of all the glitches produced in the circuit which is generally independent of the clock frequency. The readers will note that the frequency has to be high enough for the above observation to hold. Otherwise, at lower clock frequencies, the physical time taken to encrypt becomes larger and even small leakage power results in significant enough energy consumption of the order of the dynamic energy. Then the total energy increases monotonously as the frequency decreases. In [BBR16], it was also proved that encrypting one block of plaintext for any r-round unrolled implementation (given the above conditions and low leakage environment) had a quasi-quadratic form

$$E(r) = (Ar^2 + Br + C) \cdot \left(1 + \left\lceil \frac{R}{r} \right\rceil\right).$$

Here, A, B, C are constants and R is the number of iterations of the round function prescribed for the design. $Ar^2 + Br + C$ denotes the energy consumed per cycle and $\left(1 + \left\lceil \frac{R}{r} \right\rceil\right)$ is the total clock cycles required to encrypt. This expression was arrived at due to the following arguments: Since an r-round unrolled structure has r copies of the round function circuit connected serially one after the other, the glitches (which are really due to transients at beginning of the clock cycle) produced due to signal delays in the i-th round function, are compounded in the $(i+1)$-st round function and are compounded further in the $(i+2)$-nd round function (see [BBI+15, Figs. 1,2,3,4]). It was then shown that the power consumed in each round function formed a simple arithmetic sequence. Since the total power consumed is a sum of these r terms of the sequence, it results in a quadratic function in r. Multiplying this with the total time taken to encrypt, i.e., $\left(1+\lceil\frac{R}{r}\rceil\right)$ gives us the required expression. We can see that although an r-round unrolled cipher consumes *more energy per cycle* for increasing values of r, it takes *fewer cycles to complete the encryption operation* itself.

In the realm of stream ciphers, no energy model of the sort is currently known. However, in another work by Banik et al. [BMA+18], some broader conclusions about the effects of unrolling stream cipher circuits were made. They show that an unrolled stream cipher circuit that produces multiple keystream bits in one clock cycle is more energy-efficient in an asymptotic sense, i.e., when the encryption of multiple data blocks is considered instead of a single block. In fact, it was shown that for over 320 bits of data, Trivium consumed the least amount of energy on STM 90 nm ASIC circuits and outperformed the Midori block cipher family. For asymptotically large amount of data, the regular Trivium circuit reached its point of optimality relatively late at $r = 160$, and at this degree of unrolling it was around 9 times more energy-efficient than Midori-64. These findings are reflected in Figure 1, indicating that the baseline Trivium design is a fitting starting point from which new low-energy constructions can be derived.

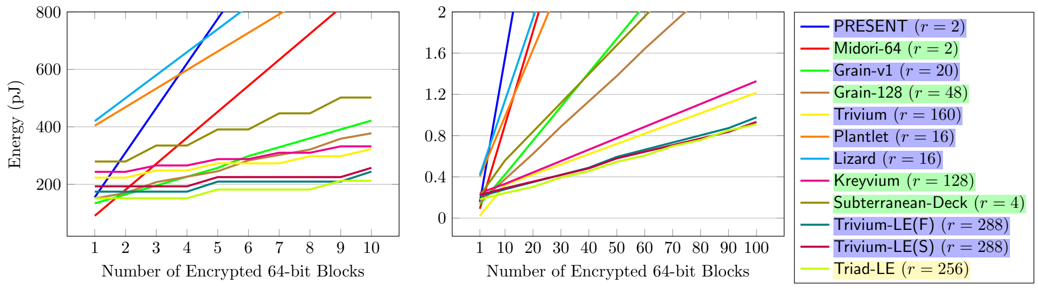


Figure 1: Energy consumption (pJ) chart from Banik et al. [BMA+18] using the STM 90 nm cell library process at a clock frequency of 10 MHz. Added to the plot are figures for the energy consumptions of Subterranean-Deck, and the designs Trivium-LE(F), Trivium-LE(S), Triad-LE that we propose in this paper, for the same standard cell library and operating frequency. Figures are reported for short messages (1 to 10 blocks of 64-bits) and longer messages (1-100 blocks). Legend entries highlighted in blue and green have a security level of 80 and 128 bits respectively, whereas Triad-LE offers 112-bit security.

The reasons why a heuristic energy model for stream ciphers appears to be harder to conceive are manifold. For one, stream ciphers circuits are often not more than a single large register bank whose outputs are fed into a thin combinatorial layer, e.g., in Trivium the state update function only consists of 12 two-input logic gates. This means that for small r the energy consumption of the algorithm is almost entirely determined by the storage elements, i.e., the contribution of the round function circuit is insignificant. Further note that when r is small the switching activity of the state update function heavily depends on the underlying cell library process and can thus vary widely. Only for large r the energy consumption of the round function layer renders itself decisive, however it becomes increasingly complex to reason about the circuit as the algebraic complexity of the underlying equations grows unmanageable, thus preventing any deeper analysis of the involved switching activity. This stands in contrast to block ciphers where the unrolling factor r is usually small and thus the complexity of the round function circuits remains bounded.

Analogously, the reasons why some hardware stream ciphers outperform block ciphers in energy efficiency are also many. Most hardware stream ciphers (like Trivium and the Grain family) are designed with a few register locations at the beginning being untapped, i.e., not used in register update. This allows for efficient hardware unrolling, so that, unlike block ciphers, each individual round in these stream ciphers can be implemented in parallel and hence does not increase the circuit depth. As such, the glitches produced in the circuit of round i do not increase the glitches in round $i+1$, at least when the circuit is unrolled for small values of r. Perhaps the most important reason is that stream ciphers perform the key-IV setup only once and then are able to encrypt multiple bits of data without having to do it again. For example, an implementation of Trivium that is unrolled $r = 128$ times, would only need $\frac{1152}{128} = 9$ clock cycles to complete key-IV setup and takes 100 more cycles to encrypt up to 12800 bits of data. The most energy-efficient implementation of Midori64 (at degree of unrolling $r = 2$), needs 8 cycles to encrypt every 64-bit block of data, and hence would need $\frac{8*12800}{64} = 1600$ cycles to encrypt the same length of data which is around 15 times more. Consequently, lightweight stream ciphers are preferable when factors like energy and throughput are concerned.

1.1 Contributions

In this paper, we investigate unrolled stream cipher constructions and make some fundamental discoveries about their energy consumption behaviour. More specifically, our contributions can be summarized as follows:

1. **Perfect Tree Energy Model.** Our first contribution in this paper is to re-implement r-round unrolled stream cipher circuits in a generic more energy-efficient manner. We shall define shortly the concept of a *circuit strand*, which basically comprises of the logic functions involved in one register update. We demonstrate that rather than following the approach in [BMA+18], if we adopt a technique in which each strand is implemented separately as a unit and the circuit synthesizer is prevented from performing any *inter-strand* optimization, then the power consumption increases in a slower manner with the respect to the degree of unrolling r. Trivium is especially suited for this restricted mode of compilation and reaches its point of optimality in the fully unrolled setting at $r = 288$. This optimal energy is significantly lower than the 160-round circuit reported in [BMA+18] under the same operating environment.

   This tessellation enables us to partition the entire circuit into smaller units which are obviously the strands. Since these are interconnected, it gives rise to a natural tree structure among them in the following way: a strand j is a child node of strand i, if the output of j is one of the inputs of i. Hereafter, by observing the variation of the power consumption in these strands, it is possible to deduce a strong correlation between the power consumed by each strand its position in the above tree, which leads to the definition of a tree-based metric that correlates the energy consumption to a wide range of stream ciphers, namely:

   (a) Trivium-like constructions [De 06] with register output tap locations chosen randomly.

   (b) Trivium-like constructions proposed in the literature that have some structural differences in comparison to the original Trivium design. These include the modified Trivium proposed in [MB07], TriviA [CCHN18], Kreyvium [CCF+18] and Triad-SC [BIM+19].

   (c) Algebraically more complex ciphers with large state update functions such as Grain-128 [HJMM06].

(d) Subterranean-like constructions [DMMR20], which do not exhibit rotating state registers.

Thus this leads to the proposal of the fist formal energy model for stream cipher constructions akin to that for block ciphers in [BBR16].

2. **New Energy-Optimal Stream Ciphers.** By leveraging the obtained energy model, we are able to show that register tap positions significantly affect the energy efficiency. Hence, our next attempt is to design new energy-optimal ciphers, where our approach is to change the register tap positions of the original Trivium cipher. However, the change of the register tap positions also affects the security, and we carefully chose these positions without decreasing the claimed security level, i.e., the 80-bit security of Trivium. We present two candidates, which we call Trivium-LE(F) and Trivium-LE(S), that consume around 10-15% and 25% less energy than Trivium, respectively. Note that Trivium-LE(F) is conservative with enough security margin, and Trivium-LE(S) is challenging with a thin security margin. As shown in Figure 1, both constructions stand as the currently most energy-efficient encryption primitives in the literature when at least 24 bytes are encrypted. The energy efficiency of Trivium-LE(F) outperforms known ciphers, and the structure is also useful to design an energy-efficient message authentication code. We present Trivium-LE-MAC whose update function inherits to Trivium-LE(F) but the message is absorbed instead of key-stream generation.

   It is important to note that our model makes it, for the first time, possible to design stream ciphers for hardware environments that are specifically optimized in terms of energy consumption as the metric is both simple and widely applicable. We also applied the same strategy to Triad-SC, which supports 112-bit security, because it seems to be the most promising for the energy efficiency due to the shorter state size of 256 bits. By altering tap locations, we present one candidate, which we call Triad-LE, that lower the energy consumption than the original Triad-SC.

1.2 Comparison with Other Works

Note that previous major works in the field of energy efficiency [BBI+15, BBR16, BMA+18] were limited in their approach in the sense that their findings were restricted to a 90 nm standard cell library and energy was computed at 10 MHz throughout. This was feasible as 90 nm standard cells have very low leakage and at a frequency of 10 MHz or higher the contribution of the leakage energy to the total energy consumption was minimal. Since the dynamic component of the energy is constant with respect to frequency, as a result, at all frequencies upwards of 1 MHz the energy consumption was more or less constant (see [BBR16, Fig. 1]). However, we present our findings for 4 different standard cell libraries in which the underlying transistors have sizes 90 nm (TSMC), 65 nm (UMC), 45 nm and 15 nm (NanGate) respectively and therefore we do not ignore leakage energy. Although for presentability, we report results at certain fixed frequencies for each library, primarily to bring out the dynamic part of it, the energy trends that we present hold across libraries and a wide range of clock frequencies, and we argue that convincingly in the paper. When this is not possible, for space constraints, the results are reported at clock frequency 10 MHz for the TSMC 90 nm and UMC 65 nm libraries and at 1 GHz for the NanGate libraries. This is done so that the dynamic energy component is the dominant contributor of the total energy consumption (for better comparison with [BBI+15, BBR16, BMA+18, KDH+12]). All energy figures are reported for encryption of 1.28 Mbits of data and are generated after a timing simulation of around 10000 test vectors on the corresponding netlist post-synthesis.

Note that in real-world, on-chip implementations of these circuits, typically there are more sources that cost energy like (a) energy consumed in the clock-tree or (b) energy consumed when the device is idling. In this work, we do not focus on these issues primarily because they are common to all circuits. Instead our focus will be on the energy consumed by the circuit itself.

1.3 Outline

In Section 2, we present the effects that different compiler directives used to synthesize stream cipher circuits have on the energy consumption. Section 3 details the obtained heuristic energy model. In Section 4, we propose energy-optimal Trivium variants and an energy-efficient message authentication code. Subsequently, in Section 5, we study recent Trivium-like, Grain-like and Subterranean-like constructions proposed in the literature and show that our derived energy model works for these designs too. The paper is then concluded in Section 6.

2 Restricted Circuits

Combinatorially heavy circuits, such as the increasingly complex algebraic state update equations in r-round unrolled stream ciphers, induce synthesis tools to produce optimized architectures in terms of circuit area. They also introduce a gap when it comes to reasoning about the overall energy consumption, which is significantly hindered as the synthesized circuits have mutated into opaque, garbled constructions.

We find that imposing a regular structure which is exclusively composed of simple combinatorial logic gates in which the state update function is replicated unaltered across different r in an unrolled setting yields equivalent if not better power figures for basic as well as more feature-rich cell libraries when compared to the highly optimized circuits of the Synopsys Design Compiler synthesis tool. We define one such structure as follows:

Definition 1 (Strand)**.** Recall the Trivium update function that consists of three independent logic blocks of the form whose inputs are tapped from the 288-bit state register $x_1, x_2, \ldots, x_{288}$ such that

$$\begin{aligned}
t_1 &\leftarrow x_{66} + (x_{91} \cdot x_{92}) + x_{93} + x_{171} & (x'_1, \ldots, x'_{93}) &\leftarrow (t_3, x_1, \ldots, x_{92}) \\
t_2 &\leftarrow x_{162} + (x_{175} \cdot x_{176}) + x_{177} + x_{264} & (x'_{94}, \ldots, x'_{177}) &\leftarrow (t_1, x_{94}, \ldots, x_{176}) \\
t_3 &\leftarrow x_{243} + (x_{286} \cdot x_{287}) + x_{288} + x_{69} & (x'_{178}, \ldots, x'_{288}) &\leftarrow (t_2, x_{178}, \ldots, x_{287}).
\end{aligned}$$

We define each individual logic block as a *strand* of the following form:

$$a + b + (c \cdot d) + e.$$

A feature-rich library with 3-pin linear cells can implement one strand with 3 gates (1 NAND2, 1 XNOR2, 1 XNOR3), hence the entire Trivium combinatorial layer then consists of 10 gates in total (9 for the 3 strands and one 3-input XOR gate for the output function). A simpler library that only consists of 2-pin linear logic elements such as the NanGate cell library family requires 14 gates for the combinatorial layer. A full description of Trivium is given in Appendix A.

In this respect, we investigate several circuit and compilation directives supported by the Synopsys Design Compiler.

- *Regular*. The entire circuit is compiled with the regular compile command which moderately attempts to optimize the synthesis result. In this setup, the synthesizer is free to choose the mapping and the corresponding optimization. The compiler may choose to not respect the boundaries between two strands and make any optimization it deems fit. This is actually equivalent to the implementation strategy of [BMA+18], i.e., in which the compiler has the freedom to optimize given the logical representation of the update function.

- *Restricted.* Same compilation directive as in the regular configuration, i.e., compile, however the synthesis of the state update function is restricted to the logical mapping, where the state update circuit for $r = 1$ is simply replicated for higher degrees of unrolling. Under this directive, the compiler puts together each strand separately and is forced to respect the boundaries between 2 strands. Thus when used as such, the compiled circuit consists of exactly $3r$ strands for an r-round unrolled construction.
- *Ultra.* The circuit is synthesized using compile_ultra directive which is a high-effort routine that optimizes beyond the entity boundaries and often yields the most area- and latency-efficient constructions. Here too, the compiler may choose not to respect strand boundaries.

One of our empirical findings is that for Trivium circuits compiled under the *Restricted* directive, the increase in the power consumption (for encrypting a given number of data blocks) is much slower (with respect to the degree of unrolling r) than circuits compiled under the *Regular* or *Ultra* directives.[1] Note that a more fundamental answer to the question whether the energy figures increase or decrease when a cipher is further unrolled is directly linked to its latency and power consumption.

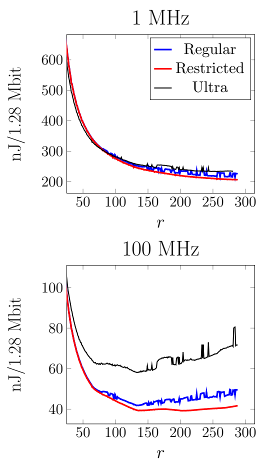
Let $L(r)$ be the total number of clock cycles required to encrypt a fixed-size plaintext block in the r-round unrolled setting and denote by $P(r)$ and $E(r)$ the power and energy values respectively. It is crucial to note that $L(r)$ will decrease and consequently $P(r)$ will increase as r increases and thus the value of r which minimizes $E(r) = P(r) \cdot L(r)$ (this is true for block ciphers too) was exactly the problem studied for block ciphers in [BBR16] and for stream ciphers in [BMA+18].

In Figure 2 and Figure 3, we detail the energy and area simulation results for four standard cell libraries (TSMC 90 nm, UMC 65 nm and NanGate 45 and 15 nm) over a wide range of frequencies. The choice of frequencies was indeed library specific: so that the critical path of the circuit was well below the clock period even when the circuit was fully unrolled. This obviates the need for the compiler to use higher drive strength based cells just to get a positive slack (i.e., ensure clock period larger than critical path), which alters the basic character of the circuit for different values of r and prevents a fair evaluation. Hence for the faster NanGate library based circuits we used the frequency range 1 MHz to 1 GHz, and for the other libraries we used the range 0.2 MHz to 100 MHz. We find that the circuits compiled in the restricted mode are by far the most energy-efficient of the three. Its energy consumption more or less decreases monotonously for $r \geq 150$, which suggests that if r is allowed to vary up to 288, then the fully unrolled cipher, i.e., $r = 288$, is the best setup for energy constrained environments (though not always). This empirical observation naturally allows us to segue into the next round of results in Section 3 where we look more closely at the circuits compiled under the restricted mode.

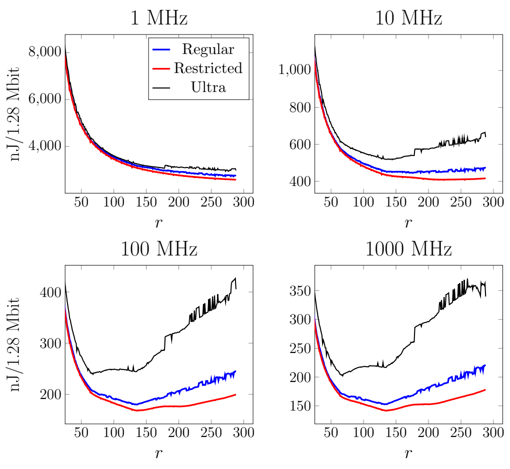
3 Perfect Tree Energy Model

For the remaining experiments, we look to investigate unrolled Trivium circuits with $r = 288$ since they achieve maximum throughput and deliver close to the best energy efficiency for all libraries across a wide range of frequencies. Though it was theoretically possible

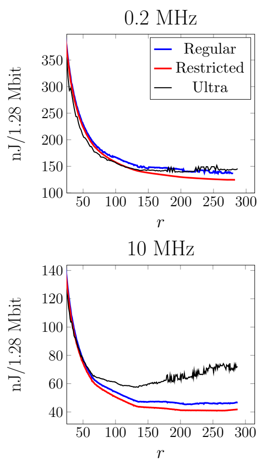
[1]One reason for this is that with the other compiler directives, the main optimization effort goes behind reducing area of the circuit and meeting timing slack requirements. And the result it blurs the boundaries between individual strands and is thus not necessarily power-optimal.



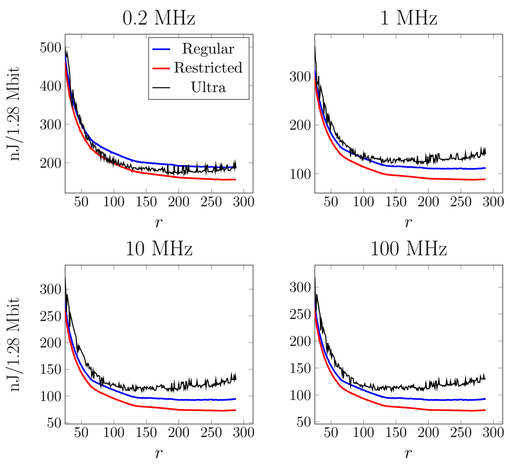
(a) NanGate 15 nm



(b) NanGate 45 nm



(c) UMC 65 nm



(d) TSMC 90 nm

Figure 2: Trivium energy measurements for the three synthesis settings for different frequencies and libraries. Note that energy graphs are noisier for the regular/ultra modes which indicates that the synthesizer chooses different mapping strategies for varying r.

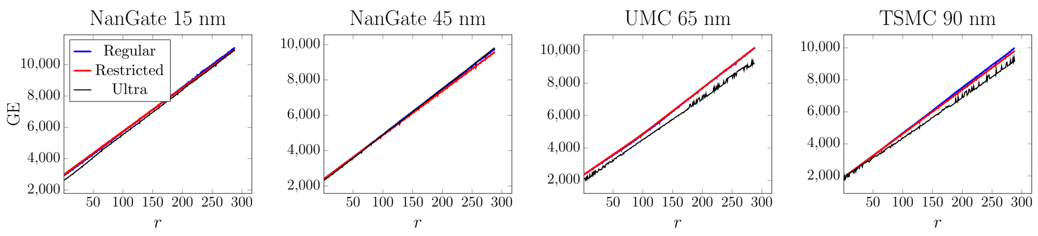


Figure 3: Trivium area measurements (Gate Equivalent) for the three synthesis settings for all unrolling factors r and cell libraries. Note that the number of clock cycles that are required in order to encrypt x bits of data is given by $\lceil \frac{1152}{r} \rceil + \lceil \frac{x}{r} \rceil$, hence the encryption of 1.28 MBit of data for $r = 288$ has a latency of 4449 cycles.

to unroll more, it would require more silicon area and improve energy efficiency only fractionally more. Since the circuits are compiled in restricted mode, it is possible to see how much power each strand consumes. We commence by introducing some notations and definitions that will help us formalize the write-up better. We commence by introducing some notations and definitions that will help us formalize the write-up better.

As mentioned in Section 2, each state update function of Trivium consists of three strands t_1, t_2, t_3, i.e.,

$$\begin{aligned} t_1 &= x_{66} + x_{93} + (x_{91} \cdot x_{92}) + x_{171} \\ t_2 &= x_{162} + x_{177} + (x_{175} \cdot x_{176}) + x_{264} \\ t_3 &= x_{243} + x_{288} + (x_{286} \cdot x_{287}) + x_{69}. \end{aligned}$$

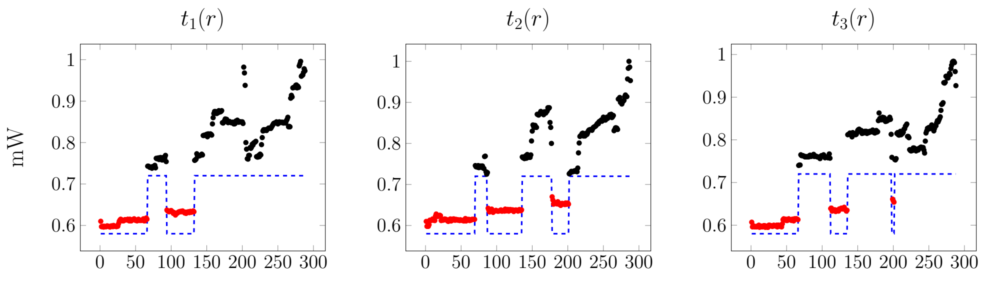
Definition 2 (i-th Strand)**.** Denote by $t_i(r)$ the strand for equation t_i in the r-th unrolled round with $i \in \{1, 2, 3\}$ and $r \in \{1, \ldots, 288\}$ such that each successive $t_i(r)$ can be recursively defined as:

$$\begin{aligned} t_1(r) &= t_3(r-66) + t_3(r-93) + [t_3(r-91) \cdot t_3(r-92)] + t_1(r-78) \\ t_2(r) &= t_1(r-69) + t_1(r-84) + [t_1(r-82) \cdot t_1(r-83)] + t_2(r-87) \\ t_3(r) &= t_2(r-66) + t_2(r-111) + [t_2(r-109) \cdot t_2(r-110)] + t_3(r-69), \end{aligned}$$

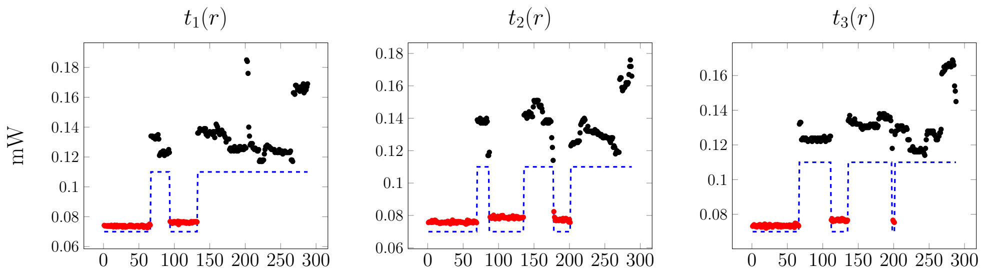
where $t_1(r) = x_{94-r}$, $t_2(r) = x_{178-r}$ and $t_3(r) = x_{1-r}$ whenever $r \leq 0$.

Figure 4 shows the power consumed in each of the strands $t_i(r)$ for increasing values of r for 2 of the libraries we experiment with in this paper. We had expected the power in the strands to increase monotonously with r as in block ciphers, but the figure clearly suggests that the increase is far from monotonous. The red marks represent the strands whose power consumption experiences a sudden dip. This observation seemed at first to be counter-intuitive, and so we set about trying to understand this curious phenomenon. We first observed that all $t_1(r)$'s (for $1 \leq r \leq 66$) consume the same power until $t_1(67)$ whose power consumption is considerably larger (note the red to black jump in Figure 4 around $r = 66$ for $t_1(r)$ for all the libraries). All inputs to $t_1(r)$ (for $1 \leq r \leq 66$) come directly from the register. Thus in some sense their input nodes are all at a distance 0 from the register. However, one of the inputs of $t_1(67)$ comes from the output of $t_3(1)$ and thus not all its inputs are at distance 0 from the register. This delay imbalance in the input wires gives rise to more glitches in the internal circuitry of $t_1(67)$ and this hints at one of the reasons why it consumes more. Further consider the boundary around $r = 93$. At $r = 94$, the power consumption of $t_1(94)$ drops. It is easy to see that all the inputs of $t_1(94)$ are at distance 2 from the register, whereas the inputs of $t_1(93)$ are still unbalanced with respect to the delay from the register. This led us to believe that delay imbalance plays a major role in determining how much power the strands consume.

Through the Looking Glass. In order to verify the above phenomenon, we looked at the internal timing diagrams of both the strand pairs (a) $t_1(66)$ and $t_1(67)$, and (b) $t_1(93)$ and $t_1(94)$, presented in Figure 5 (the circuit was synthesized using NanGate 45 nm cell library and clocked at 1 GHz). Let us examine $t_1(66)$. The first 2 input pins $x_1,\ x_{28}$, according to the circuit synthesizer, have an average delay of 0.09 ns from the clock edge at which the new inputs are written on to the registers. As a result, the output of the first XOR gate in the strand i.e., $x_1 \oplus x_{28}$ is only moderately glitchy. Over 4450 clock cycles this net switches logic only 2271 times, as found by a post-synthesis timing simulation on the netlist. On the other hand, in $t_1(67)$, x_{27} is at a delay 0.09 ns whereas the other input $t_3(1)$ is at an average delay 0.25 ns. The output of the corresponding XOR gate $x_{27} \oplus t_3(1)$ is glitchier as compared to $x_1 \oplus x_{28}$, it switches 4512 times in the same interval. This clearly indicates that $t_1(67)$ consumes more power. Conversely, consider $t_1(93)$. The first 2 input pins $x_1,\ t_3(27)$ have delays 0.09 ns and 0.25 ns from the clock edge. Hence the net



(a) NanGate 45 nm



(b) TSMC 90 nm

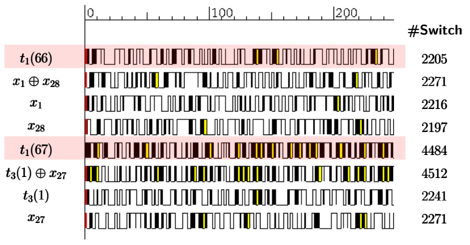
Figure 4: Power measurements for all the perfect unrolled strand trees for two cell library processes. The red data points indicate unrolled strand equations which correspond to perfect trees. The dashed blue line signifies the transition boundaries between perfect and imperfect trees, i.e., low points represent perfect unrolled strand trees while high points correspond to imperfect trees.

$x_1 \oplus t_3(27)$ switches around 4665 times in the same interval. However, in $t_1(94)$, the pins $t_3(1)$, $t_3(28)$ have delays 0.24 ns and 0.25 ns. Hence, many of the glitches produced by them cancel out and the XOR net $t_3(1) \oplus t_3(28)$ switches 2551 times in this interval. This indicates that depth-balanced strands consume less power than unbalanced ones.

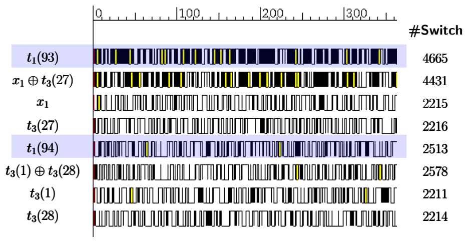
3.1 Circuit to Tree

In order to formalize the above phenomenon, we found that the circuit strands are connected naturally in a well-defined graphical topology. Each unrolled strand can be translated into a 5-ary tree with the root node as the output bit whose subtrees are other unrolled strand trees or leaf nodes.

Definition 3 (Unrolled Strand Tree)**.** Let $T_i(r)$ be the 5-ary unrolled strand tree corre-



(a)



(b)

Figure 5: Timing diagrams for internals in (a) $t_1(66)$, $t_1(67)$, and (b) $t_1(93)$, $t_1(94)$.

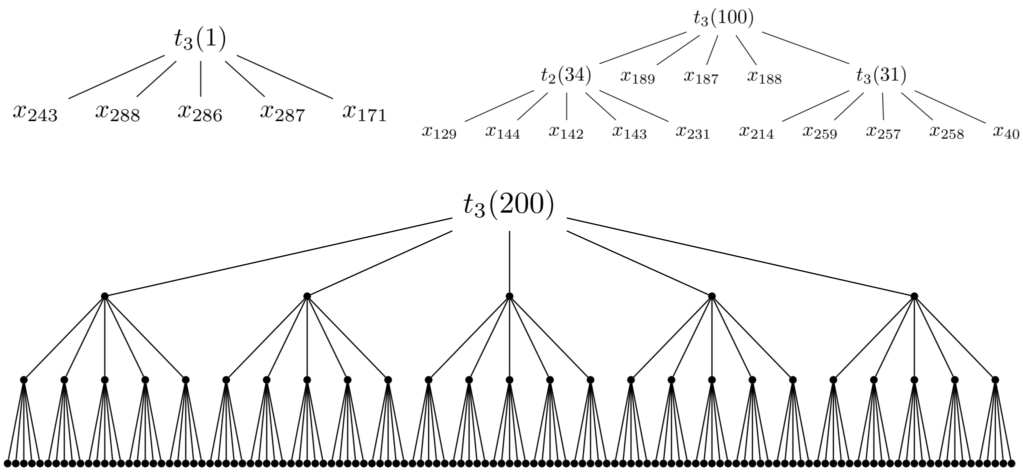

$t_3(1)$
x_{243} x_{288} x_{286} x_{287} x_{171}
$t_3(100)$
$t_2(34)$ x_{189} x_{187} x_{188} $t_3(31)$
x_{129} x_{144} x_{142} x_{143} x_{231} x_{214} x_{259} x_{257} x_{258} x_{40}
$t_3(200)$

Figure 6: The strand trees $T_3(1)$, $T_3(100)$ and $T_3(200)$. $T_3(1)$, $T_3(200)$ are perfect.

sponding to the unrolled strand equation $t_i(r)$. The child nodes of the strand $T_i(r)$ are therefore all the 5 nodes $T_j(u)$ for which the corresponding terms $t_j(u)$ are present in its recursive definition as per Definition 2.

Example 1. To make the link between unrolled strand equations, and their respective trees clearer, we give 3 examples of varying complexity. The unrolled strand trees $T_3(1)$, $T_3(100)$ alongside $T_3(200)$ are displayed in Figure 6. Note that terms that appear several times in an unrolled strand equation result in duplicate nodes in the corresponding unrolled strand tree. This is to ensure that the equations are a one-to-one representation of the actual circuit.

We can further classify our unrolled strand trees as either perfect or imperfect according to the following definitions.

Definition 4 (Perfect m-ary Tree)**.** A perfect m-ary tree is a tree in which all non-leaf nodes have m children and all leaf nodes are at the same depth.

Clearly, the unrolled strand trees in Trivium are 5-ary. Further, remark that in Figure 6, $T_3(1)$ and $T_3(200)$ are perfect unrolled strand trees while $T_3(100)$ is imperfect due to having leaf nodes at different depths. In the example in the previous subsection clearly, $T_1(66)$, $T_1(94)$ were perfect trees whereas $T_1(67)$, $T_1(93)$ were not. This gives us a very good understanding of the power consumption of strands vis-à-vis the position of the corresponding nodes in the circuit tree graph. A strand evidently consumes less power if the node it occupies in the circuit graph houses a perfect tree.

Let us try to argue this inductively. A tree is 5-ary perfect if and only if all of its 5 child nodes are also perfect. Thus it is easy to see that in a perfect tree all its input nodes are at approximately the same average delay from the register. This being so all perfect trees tend to consume less power. On the other hand a tree is imperfect if and only if one of its child nodes is also imperfect, due to which the gate output corresponding to this imperfect child node is considerably more glitchy. This excess glitch from the child node would naturally be carried forward in the parent strand making its output glitchier and thus causing it to consume more dynamic power. This observation naturally leads us to the next question: is it possible to have a general Trivium-like stream cipher (with tap locations perhaps different from the original Trivium specifications) that is more energy-efficient and also secure at the same time? The translation of circuit to an equivalent algebraic topology may have given us a quick way to check this. Since perfect trees consume less dynamic power, a variant of Trivium (with different tap locations) is likely to consume less energy if its circuit tree graph has a larger total number of perfect trees.

Let us provide more arguments as to why the above makes sense. Consider two configurations of Trivium: Trivium-A and Trivium-B with different tap locations (both synthesized in restricted mode). At a degree of unrolling equal to 288, the circuits of both these variants consist of exactly the same amount of gates and flip-flops. Since the leakage power in a circuit depends directly on the total silicon area, both these circuits are likely to consume the same leakage power. Furthermore, the circuit graphs of both these variants have exactly the same amount of nodes. If for example Trivium-A has more perfect trees in the graph than Trivium-B, then it automatically implies that Trivium-A has fewer imperfect trees than Trivium-B, which more or less implies that Trivium-A is likely to be the variant that consumes less dynamic power. Since the leakage power is the same, this means that the Trivium-A consumes less total power and hence less total energy. This of course should hold irrespective of the standard cell library used to synthesize the circuit or the frequency of signal used to clock the circuit.

We can estimate the total number of perfect trees in a generic Trivium configuration. To ease notation we will denote the total number of perfect trees among all strands $t_i(r)$ as $S(T_i)$ such that the total number of perfect trees in the circuit is $S(T) = \sum_i S(T_i)$. More formally, let f be a function from the set of all trees to $\{0, 1\}$ such that $f(T_i(r)) = 1$ if and only if $T_i(r)$ is a perfect tree, and is 0 otherwise: then $S(T_i) = \sum_r f(T_i(r))$. Below, we report the distribution of perfect unrolled strand trees in the original Trivium. In Trivium, we have $S(T_1) = 105$, $S(T_2) = 144$, $S(T_3) = 93$, and hence $S(T) = 339$. Note that there are no perfect unrolled strand trees of depth 4 or larger.

3.2 Enumerating Perfect Trees

In the following, let us consider a generic Trivium layout in order to determine configurations that yield a high number of perfect trees and consequently lower the power consumption.

Definition 5. Denote by Trivium(X, n) a generic Trivium configuration composed of n chained registers $(X_1, \ldots, X_n)$ such that X_j^ℓ is the j^{th} register's leftmost forward tap, X_j^f is the feedback tap and X_j^{op} is the output tap. See Figure 7 for a schematic depiction. Note that X_j^{op} is essentially the final tap location of the j^{th} register (this is required to ensure the one-to-one nature of the Trivium update). The figure does not explicitly show the taps for the AND gates, as we will show that if both the AND taps are between X_j^ℓ and X_j^{op} then it does not affect the total number of perfect trees in the circuit graph.

Note that this notation corresponds to n update function strands hence the unrolled strand tree of $t_j(r)$ is $T_j(r)$.

Example 2. The original Trivium specification composed of three update function strands is congruent to Trivium$(X, 3)$ where $X_1^\ell = 66$, $X_1^f = 69$, $X_1^{op} = 93$, $X_2^\ell = 69$, $X_2^f = 78$, $X_2^{op} = 84$, $X_3^\ell = 66$, $X_3^f = 87$, $X_3^{op} = 111$ with an additional non-linear gate between the leftmost and output tap in each register.

Finding configurations that lead to an increased number of perfect trees seems non-trivial as the search space is enormous. Additionally, a closed-form solution that evaluates the exact number of perfect trees for a given circuit Trivium(X, n) appears equally hard. A brute-force solution consists of individually creating the unrolled strand tree for each equation and checking that all leaf nodes are at the same distance from the root. However, this approach is expensive and hard to optimize apart from ordinary parallelizations. Nevertheless, transcribing the problem into a recurrence relation offers some remedy to this issue.

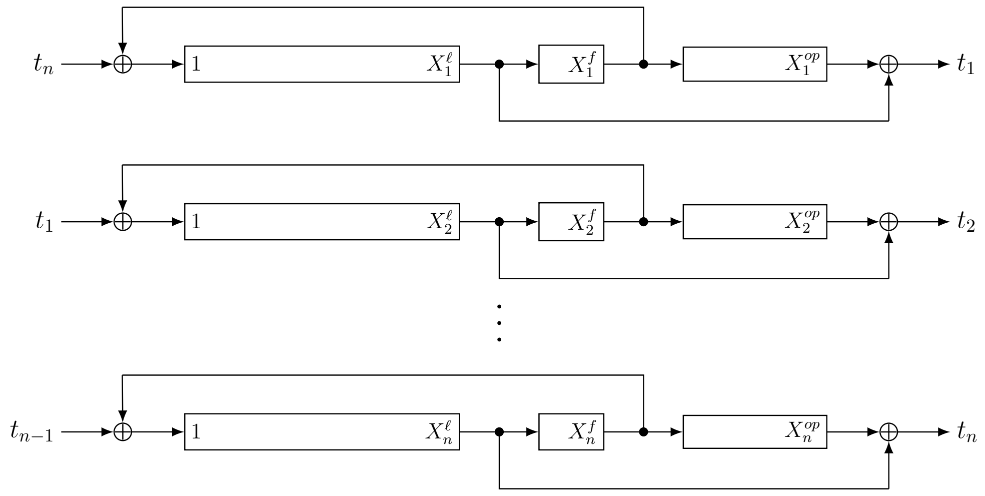


Figure 7: Generic $\mathsf{Trivium}(X, n)$ configuration of n chained registers.

Lemma 1. *Given an arbitrary, generic* $\mathsf{Trivium}(X, n)$ *circuit composed of* n *registers, the total number of perfect unrolled strand trees* $S(T)$ *in the fully unrolled setting is given by* $S(T) = \sum_{j=1}^{n} S(T_j) = \sum_{j=1}^{n} \sum_{l=1}^{n} (g_l(X_j) - f_l(X_j))^+$ *, where* $y^+ = \max\{y, 0\}$ *and* $f_l(X_j)$*,* $g_l(X_j)$ *are recursively defined functions for* $1 \le l \le n$ *of the form*

$$f_l(X_j) = \max\left\{f_{l-1}(X_{j-1}) + X_j^{op},\ f_{l-1}(X_j) + X_{j+1}^f\right\}$$

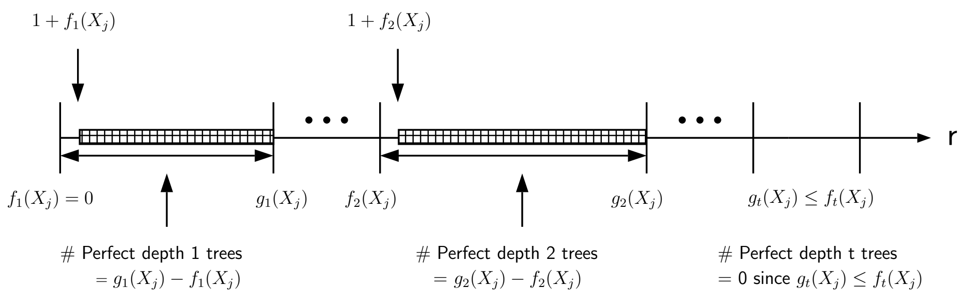
$$g_l(X_j) = \min\left\{f_{l-1}(X_{j-1}) + X_j^{op} + \left[(g_{l-1}(X_{j-1}) - f_{l-1}(X_{j-1})) - (X_j^{op} - X_j^\ell)\right]^+,\right.$$
$$\left. g_{l-1}(X_j) + X_{j+1}^f\right\},$$

such that $f_1(X_j) = 0$ *and* $g_1(X_j) = \min\left\{X_j^\ell,\ X_{j+1}^f\right\}$*. The number of perfect trees of depth* t *for the* j*-th strand is* $S(T_j)|_{depth=t} = (g_t(X_j) - f_t(X_j))^+$*. Hence the total number of trees of all depths is* $S(T_j) = \sum_{l=1}^{n} (g_l(X_j) - f_l(X_j))^+$ *and thus the lemma follows.*

We remark that since there are n registers indexed 1 to n the value of $j + 1$ (resp. $j - 1$) refers to addition (resp. subtraction) mod$n$ in the set $\{1, 2, \cdots, n\}$. Further note that a tree is perfect if and only if all its subtrees are perfect.

Proof. (Intuition) From Figure 8, we can see that there are certain values of r for which the circuit for $t_j(r)$ produces a perfect depth 1, depth 2 tree etc. We define two families of functions f_t, g_t such that $f_t(X_j) + 1$ is the minimum value of r for which $t_j(r)$ corresponds to a perfect depth t tree, and similarly $g_t(X_j)$ is the maximum such value of r. It stands to reason that the total number of depth t trees produced in this range of r is $g_t(X_j) - f_t(X_j)$. Note that, obviously if $g_t(X_j) \le f_t(X_j)$ for some t then there do not exist any depth t trees. It remains to show that f_t and g_t can be recursively defined. The full proof is considerably involved and is given in Appendix B.

To conclude let's argue why the number of perfect trees is independent of the AND gate taps as long as they are to the right of the leftmost tap X_j^ℓ. It is intuitively not difficult to reason why and let us argue with the help of our previous example: $t_1(66)$ corresponds to a perfect tree but $t_1(67)$ is imperfect. This is because in the process of unrolling $X_1^\ell + 1$ is the first value of r at which $t_1(r)$ no longer takes inputs directly from the register. Thereafter, it does not matter where exactly the AND taps are as long as they are to the right of X_1^ℓ: all subsequent values of r until X_1^{op} continue to produce imperfect trees. □



$1 + f_1(X_j)$
$1 + f_2(X_j)$
r
$f_1(X_j) = 0$
$g_1(X_j)$
$f_2(X_j)$
$g_2(X_j)$
$g_t(X_j) \leq f_t(X_j)$
# Perfect depth 1 trees $= g_1(X_j) - f_1(X_j)$
# Perfect depth 2 trees $= g_2(X_j) - f_2(X_j)$
# Perfect depth t trees = 0 since $g_t(X_j) \leq f_t(X_j)$

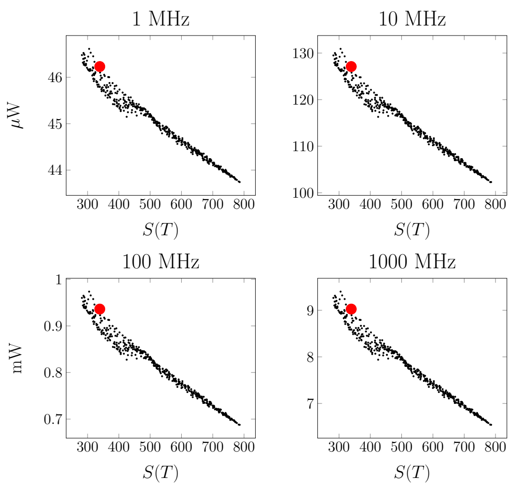
Figure 8: Illustration of finite depth trees in Trivium circuit.

Verification: In order to verify our hypothesis (at least empirically) that **(a)** the number of perfect trees is actually a good indicator of the energy consumption of a generalized Trivium circuit, and **(b)** that the above holds irrespective of the cell library used to construct the circuit or frequency of the signal used to clock it, we performed an extensive simulation experiment. We generated a large number of Trivium circuits with random taps and calculated the number of perfect trees with the help of the recursion formula given above. We synthesized each circuit in restricted mode using the 4 cell libraries used in all of our experiments and computed the total power consumed at a wide range of frequencies. The results are plotted in Figure 9. Not only is there a strong negative correlation between the power consumed (and hence energy) and the number of perfect trees, the results hold across libraries and clock frequencies as claimed in Section 3.1. For each cell library the same trend is visible across all frequencies. Similarly, since the leakage power of each random Trivium instance is the same and frequency independent (say it is equal to P_l), and since decreasing the frequency (alt. increasing the clock period by ΔT) only increases the physical time required for encrypting a fixed size plaintext block by an amount proportional to ΔT, hence it follows that the leakage energy of each Trivium instance increases by an amount proportional to $P_l \cdot \Delta T$ when the frequency is decreased. Since the dynamic energy is frequency independent, hence all other things remaining the same, when only the frequency is varied, it is equivalent to translating each energy scatter plot by a constant amount along the Y(energy)-axis.

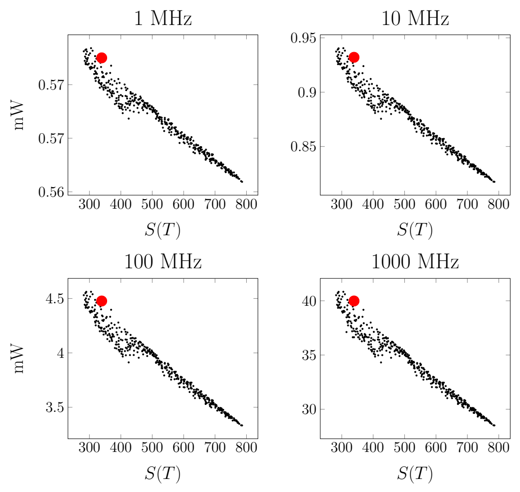
Note that even for configurations with same number of perfect trees, there may be a slight variation in energy consumption, but this variation is negligible as the number of perfect trees increase. This really depends on how badly the imperfect trees are configured in the graph, i.e., configurations with large number of trees with wide variation of delays at their input nodes tend to consume more energy. To model such situations when the number of perfect trees is small, one can think of secondary metrics like the distribution $D(x)$ of number of trees where the absolute difference of the maximum and minimum depths of leaves in the tree is equal to x (note $D(0)$ is the number of perfect trees). It is easy to see that configurations for which $D(x)$ is lower for higher values of x (i.e. lesser number of highly imbalanced trees) are better for energy. Also note that the graph tells us that to get any significant decrease in energy consumption over the original specifications of Trivium (around 10-20%) one needs at least 500 perfect trees.

3.3 Post-Routing

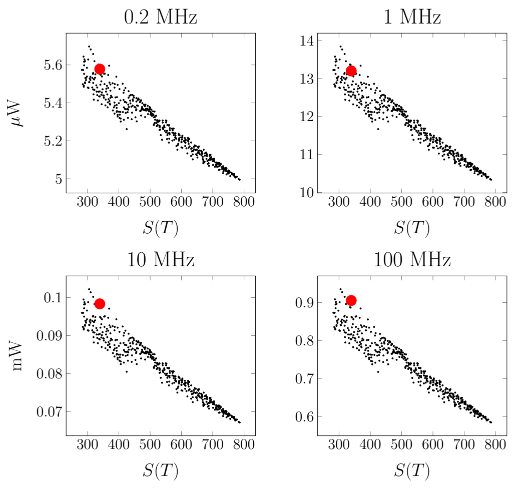
Power measurements of integrated systems are usually carried out at the gate level on the post-synthesis netlist and do not account for effects that normally arise after the circuit has been mapped into silicon which are mainly due to parasitics introduced by interconnects. Hence, in this post-routing setting, the obtained post-synthesis figures would need to be reevaluated as to obtain a more accurate picture.



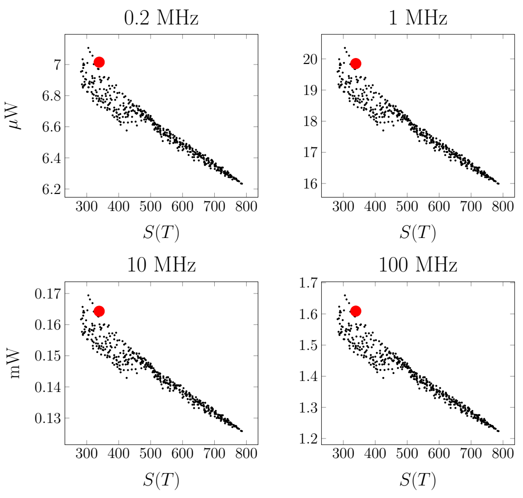
(a) NanGate 15 nm



(b) NanGate 45 nm



(c) UMC 65 nm



(d) TSMC 90 nm

Figure 9: Power measurements of several Trivium$(X, 3)$ circuits vs $S(T)$ for different libraries and frequencies. The red data points signify the power consumption of original Trivium.

In our case, we repeated the experiments from Figure 9 post-route using the Cadence Innovus place-and-route implementation system for the TSMC 90 nm cell library and report that the perfect tree model applies in almost the same magnitude as in the post-synthesis setting. On average, the added interconnect circuitry imposes an area penalty of roughly 4-5% and thus does not affect the overall results as shown in Figure 10.

4 Energy-Optimal Variants of Trivium

Before we start to look for more energy-efficient Trivium configurations with more perfect trees, let us once again look at the recursion relationship we have just stated. Note that most perfect trees are at depth 1. In order to increase the number of degree 1 perfect trees, it is obvious that we need to have higher values of $g_1(X_j) = \min\{X_j^\ell, X_{j+1}^f\}$, i.e., each tap

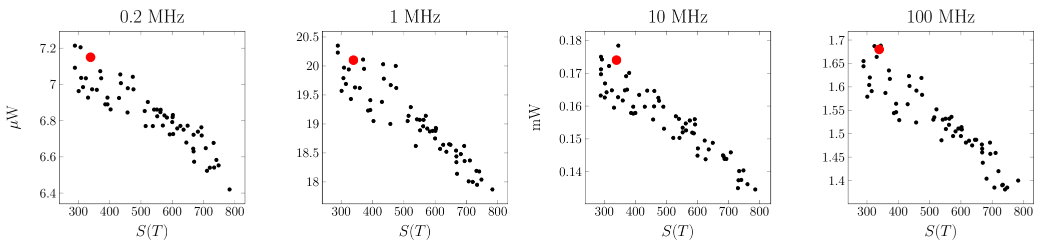


Figure 10: Post-routing power measurements of several Trivium$(X, 3)$ circuits as a function of $S(T)$ for using the TSMC 90 nm process. The red data points signify the power consumption of the original Trivium. Note that fewer data points are being plotted as the post-routing workflow is significantly more time consuming in comparison to post-synthesis analysis.

location should be chosen towards the end of the register. Naturally, it is not possible to choose each tap location only energy efficiency reasons as the new configuration must be as secure as the original Trivium. Since the search space is large, we decided to follow the following criteria, inherited from the original Trivium:

A: The linear tap locations X_i^ℓ, X_i^f and X_i^{op} for all i, are chosen from the multiple of 3. In other words, X_i^ℓ, X_i^f, and X_i^{op} are divisible by 3 for all i.

B: The locations of AND gates are fixed such that these two inputs are not divisible by 3. In Trivium, $X_i^{op} - 1,\ X_i^{op} - 2$ are chosen for all i. However, as discussed in the previous section, the impact on the energy consumption is negligible as long as the number of perfect trees is the same. Therefore, we change the AND location to $X_i^\ell + 1,\ X_i^\ell + 2$. Then, the number of perfect trees never changes, and the number of times that AND gates are applied increases according to the increase of the number of rounds. Thus, this choice is profitable for the security without increasing the energy consumption.

C: Each tap location for X_i^ℓ and X_i^f is larger than 64 such that a $64\times$ parallel implementation is possible in the software.

D: Under the condition where the output of each AND gate is approximated to 0, we denote by ϵ the maximum correlation in a linear combination of keystream bits. In Trivium, $\epsilon = 2^{-72}$, but it is quite robust against linear attacks because at least 2^{144} keystream bits are required. For a cipher targeting 80-bit security, $\epsilon \leq 2^{-40}$ is necessary.

In particular, **A** and **B** are two of the most important criteria in the design philosophy of Trivium. Thanks to them, we can expect that ϵ in criterion **D** is the highest correlation even when the condition where the output of each AND gate is approximated to 0 is removed. It is primarily because of the following reason. Under parameters following **A** and **B**, the whole cipher is divided into three sub-ciphers, and each sub-cipher is only connected non-linearly. In **D**, we first evaluate the correlation under the restriction, where the output of AND gate is always approximated to 0. In other words, only one sub-cipher is active, and the other two are inactive. Of course, this restriction is not exhaustive. However, intuitively, we are unlikely to find a better distinguisher beyond this restriction. Because, if at least one output of AND gate $x \cdot y$ is approximated to $x,\ y$, or $x + y$ instead of 0, it implies at least two sub ciphers are active. It intuitively increases the number of active AND gates and makes constructing linear distinguishers with high correlation much harder.

Table 1: List of configurations and associated security parameters. ϵ represents maximum linear bias. T is the complexity of guess-and-determine attack. c represents an additional cost required to do Gaussian elimination to solve a set of linear equations to recover the internal state. The first row represents the parameters for the original Trivium.

	Parameters									# Perfect Trees	Max Bias	G&D Complexity
#	X_1^ℓ	X_1^f	X_1^{op}	X_2^ℓ	X_2^f	X_2^{op}	X_3^ℓ	X_3^f	X_3^{op}		$-\log_2\epsilon$	$\log_2 T$
1	66	69	93	69	78	84	66	87	111	339	72	$\log_2 c + 83.5781$
2	87	78	93	66	90	99	75	87	96	495	72	$\log_2 c + 81.3796$
3	87	81	90	81	96	102	69	87	96	534	68	$\log_2 c + 80.0959$
4	75	84	87	84	81	105	81	90	96	570	64	$\log_2 c + 79.0486$
5	75	90	93	90	87	96	78	93	99	591	60	$\log_2 c + 78.8501$
6	81	90	93	90	84	96	78	93	99	624	56	$\log_2 c + 77.8853$
7	81	90	93	90	84	99	81	93	96	642	52	$\log_2 c + 77.2073$
8	75	90	93	96	87	99	87	93	96	666	48	$\log_2 c + 77.0503$
9	84	90	96	90	87	96	87	93	96	699	40	$\log_2 c + 75.8174$

As a result, three criteria **A**, **B**, and **C** allow us to reduce the number of candidates to $28534800 \approx 2^{24.8}$, and exhaustive search is possible. We exhaustively searched for the best candidates, i.e., the number of perfect trees is maximized, for correlation $\epsilon \in \{2^{-72}, 2^{-68}, 2^{-64}, 2^{-60}, 2^{-56}, 2^{-52}, 2^{-48}, 2^{-44}, 2^{-40}\}$ in **D**. In Table 1, we list the best candidates for each ϵ. In addition, we also applied Maximov and Biryukov's Guess-and-Determine attack [MB07] on each of the candidates and list the result. In this attack, the weakness of the multiple-of-3 choice is exploited, and this attack shows Trivium has 80-bit security but it does not have 128-bit security even if the key length is simply extended to 128 bits. Note that this attack has many parameters and scenarios. The complexity listed in Table 1 is the so-called scenario T1, i.e., the time complexity is minimized under the condition that solving only a linear system is enough to recover the key.

It is clear from the table that an increase in the number of perfect trees is generally accompanied by an increased maximum linear bias and decrease in the complexity of the guess-and-determine attack. Considering $c \approx 2^{16}$, all parameters would have 80-bit security, but the security margin is very marginal for parameters whose ϵ is close to 2^{-40}. The parameter in row 2 is the best one whose correlation is as low as the original Trivium, but the number of perfect trees is not over 500.

4.1 Trivium-LE(F) and Trivium-LE(S) [2]

Having established a set of potential configurations, we proceed to the proposal of two energy-optimal Trivium-like designs.

4.1.1 Trivium-LE(F)

As a good alternative which is almost equivalently secure as the original Trivium, the second row of Table 1 gives us the parameter set

$$(X_1^\ell, X_1^f, X_1^{op} \ ; \ X_2^\ell, X_2^f, X_2^{op} \ ; \ X_3^\ell, X_3^f, X_3^{op}) = (87, 78, 93 \ ; \ 66, 90, 99 \ ; \ 75, 87, 96).$$

A graphical depiction of those parameters is given in Figure 11.

This choice gives us a decrease in energy of around 15% over the original Trivium and still provides us with some headway over the margins of security. We therefore propose

[2]Note that the **(F)** and **(S)** stand for Fast and Slow respectively. This is because the **(S)** variant uses a larger number of initialization rounds.

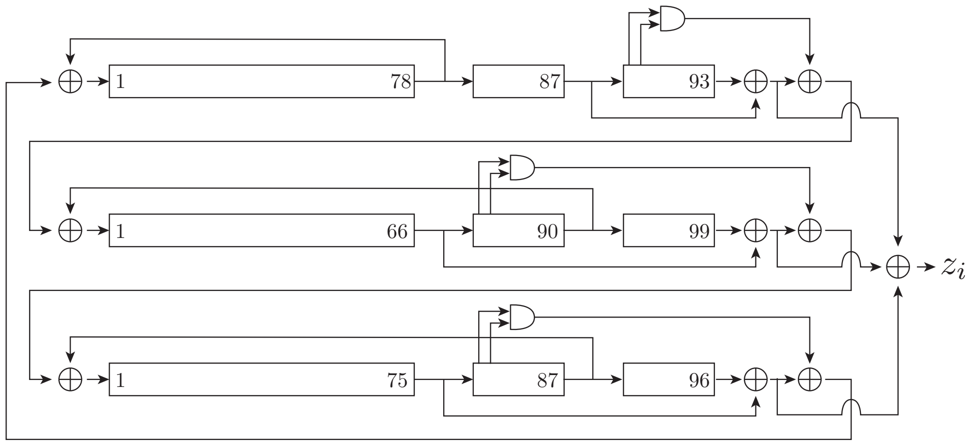


Figure 11: Update function of Trivium-LE(F).

this parameter set as a more energy-efficient variant of Trivium and call it Trivium-LE(F). We keep the key-IV setup and initialization routines for Trivium-LE(F) same as Trivium. For completeness, we round off this section with a preliminary security analysis. Since only the tap locations are modified in Trivium-LE(F), all types of attacks against Trivium can be applied against Trivium-LE(F). Three important attacks against Trivium are discussed below.

Linear Distinguishing Attack. In order to achieve 80-bit security, there should not be linear distinguishers whose correlation is higher than 2^{-40}. As we already discussed in the section before, the best correlation is 2^{-72} when outputs of AND gates are approximated to 0. While it is unlikely to find better distinguishers due to the multiple-of-3 property, we heuristically evaluated the case where these outputs are not approximated to 0. As we expected, we could not find better linear distinguishers with correlation higher than 2^{-72}.

Maximov and Biryukov's Guess-and-Determine Attack. This attack mainly exploits the multiple-of-3 property of Trivium, and it should be effective because Trivium-LE(F) also inherits the multiple-of-3 property. This attack first divides the internal state into three sub states and consists of two phases. In the first phase, we first guess one of three sub states at some time. In the second phase, assuming that the sub state is guessed correctly, we next recover the rest of the bits, i.e., $288 \times 2/3 = 192$ bits. Then, we guess outputs of any AND gates and collect keystream bits, which are linearly represented by the internal state. In the so-called scenario T0, no output of any AND gates is guessed. When we use T0 to attack Trivium-LE(F), the time complexity is $c \cdot 2^{74.0}$, which is the same as the attack against Trivium in the same scenario. However, only 96 linear equations are collected for the second phase and it is not enough to recover the remaining 192 bits. Thus, we need to solve a nonlinear system but an efficient algorithm is not known. In scenario T1, outputs of some AND gates are guessed to collect enough linear equations to recover the remaining 192 bits. When we use T1 to attack Trivium-LE(F), the time complexity is $c \cdot 2^{81.3796}$, where 48, 45, and 44 outputs of AND gates are guessed for each register. Then, we can collect 192 linear equations for the second phase, and an efficient algorithm such as the Gaussian elimination is available. Considering $c \approx 2^{16}$, Trivium-LE(F) is secure enough against this attack.

Cube Attack. Unlike the attacks above, the target of the cube attack is the initialization phase of the cipher. The cube attack was initially introduced in [DS09]. The original attack was experimental and its aim was to find linear or quadratic superpolies. However, after the division-property based cube attack was proposed [TIHM17, TIHM18], the theoretical security estimation is possible, and nowadays, the best cube attacks against Trivium are based on the division-property based method [WHT+18, HLM+20, HIJ+19].

Cube attacks exploit low algebraic degree in the initialization. The first keystream bit is regarded as the output of the Boolean function $f_k(iv)$. To execute cube attacks, the superpoly has to be recovered, and it becomes impossible several rounds after the degree of $f_k(iv)$ reaches 80. In practice, the best cube attack against Trivium is 842 rounds [HLM+20, HSWW20], and the degree reaches 80 in 840 rounds. Therefore, we first investigated the algebraic degree on $f_k(iv)$ by using the bit-based division property [Tod15, TM16] and the left plot in Figure 12 shows the increase in the upper bound of the algebraic degree. Thanks to changing the location of AND gates, the algebraic degree of Trivium-LE(F) increases faster than Trivium, and the degree reach 80 in 780 rounds. Moreover, to conservatively evaluate the degree of the superpoly to be high enough, we also investigated the upper bound of the algebraic degree on $f(k, iv)$ by using the bit-based division property. The right plot in Figure 12 shows the increase in the upper bound of the algebraic degree. About 900 rounds show the upper bound is full, i.e., 160, and it implies that the degree of the superpoly is unlikely to be lower even if we use the 80-dimensional cube. In both cases, $f_k(iv)$ and $f(k, iv)$, Trivium-LE(F) is more secure than Trivium against cube attacks. Thus, we conclude that Trivium-LE(F) has a large security margin against cube attacks.

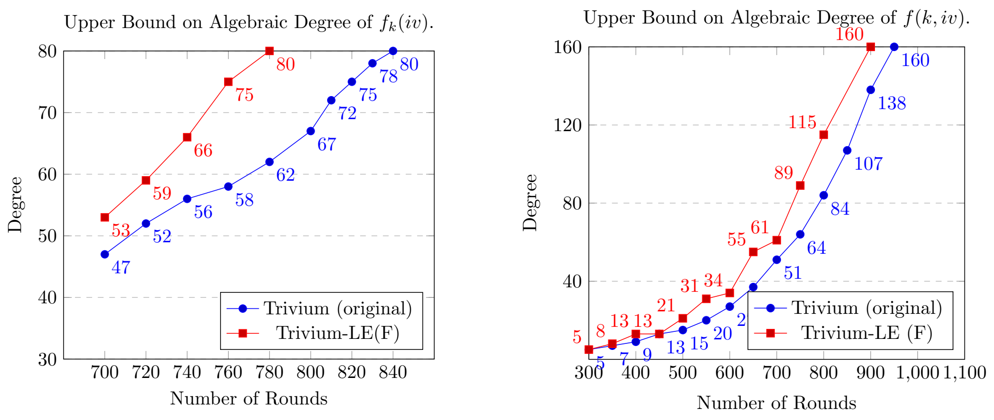


Figure 12: Increase in algebraic degree with respect to the #initialization rounds.

4.1.2 Trivium-LE(S)

We suggest another variant Trivium-LE(S) that results in around 25% lower energy when compared with Trivium. This variant is based on the 8th row of Table 1, and uses the parameter set

$$(X_1^\ell, X_1^f, X_1^{op} \; ; \; X_2^\ell, X_2^f, X_2^{op} \; ; \; X_3^\ell, X_3^f, X_3^{op}) = (96, 87, 99 \; ; \; 87, 93, 96 \; ; \; 75, 90, 93).$$

We have found that this cipher is algebraically weaker than Trivium, in as much as the algebraic degree of its output bit increases more slowly. It needs 1050 and 1200 rounds to reach the upper bounds of the degree of $f_k(iv)$ and $f(k, iv)$ be the full, respectively. Compared to the original Trivium, the increase of the degree is about 25% slower. Therefore we suggest that for a safe security margin, the number of initialization rounds used with this variant is $288 \times 5 = 1440$. Note that in terms of an 288 times unrolled circuit, this variant only takes 1 extra clock cycle to initialize, and so asymptotically speaking the energy consumption does not increase due to this extra cycle. For space constraints we defer the security analysis to Appendix C.

4.2 Trivium-LE-MAC

In addition to the stream ciphers Trivium-LE(F) and Trivium-LE(S), we also propose a message authentication code (MAC) scheme called Trivium-LE-MAC, which is designed by slightly modifying the round function of Trivium-LE(F). To realize a MAC scheme whose energy consumption is competitive with the stream cipher Trivium-LE(F), it should absorb a 1-bit message into the internal state every round function. While the easiest method is simply XORing the 1-bit message with any 1 bit in the internal state, it is not secure enough against forgery attacks. To guarantee forgery security, we evaluated the lower bound in the number of active AND gates with an MILP-based method when two different messages are absorbed. After exhaustive experimentation we found that a 1-bit message has to be XORed to at least 3 positions of the internal to be secure against forgery attacks. For example, one possible choice is to XOR the 1-bit message with three output bits of state update function, i.e., t_1, t_2, t_3.

From an energy perspective, it is advisable that these injections take place as close as possible to the registers inputs, i.e., to locations $a_{u_1}, b_{u_2}, c_{u_3}$ for smaller values of u_1, u_2, u_3. If we model the message inputs as zero-depth nodes, then it makes each strand $t_i(r)$ correspond to 6-ary trees. It is, for example, easy to see that the first $1 \leq r \leq X_i^\ell - u_i$ strand trees for $t_i(r)$ are all depth 1 perfect 6-ary trees. Hence lower values of u_i intuitively make sense.

On the other hand, we chose the injected positions by respecting the multiple-of-3 property to efficiently evaluate the resistance against forgery attacks with an MILP-based method. Specifically, in the constructed model, the message difference is only allowed to be injected at clock cycles $3j_1 + j_0$ ($j_1 \geq 0$) when the non-zero difference is first introduced at the clock j_0. Moreover, the output difference of the active AND gate is always assumed to be 0. The goal is to minimize the number of active AND gates in a trail available for the forgery attack, i.e., the difference of the whole internal state becomes zero after a certain number of clocks. We evaluated all possible candidates of the three injected positions $(a_{1+3i_0}, b_{1+3i_1}, c_{1+3i_2})$ where $0 \leq i_0 \leq 30$, $0 \leq i_1 \leq 32$ and $0 \leq i_2 \leq 31$. When the total distance from the first bit of each register is smaller than 69, among all the candidates, the maximal number of active AND gates is 72. Thus, we choose the best candidate (a_1, b_7, c_1) which reaches 72 active AND gates while achieving the smallest total distance of 6.

Algorithm 1 shows the specifications. Note that Trivium-LE-MAC inherits the security level of Trivium-LE(F) against any key-recovery attack, i.e., 80-bit security. However, the tag length is at most 64 bits. In other words, the security level of the integrity is at most 64 bits.

4.3 Remark About Authenticated Encryption

Authenticated encryption schemes attract strong interest from both industrial and academic communities, and an authenticated encryption using Trivium-LE(F) would be beneficial to lower energy consumption. In one of the possible constructions, inspired by the duplex sponge construction, a 1-bit message is absorbed into the internal state, and simultaneously, a key stream is squeezed to encrypt the 1-bit message. However, since the round function of Trivium-LE(F) is very sparse, the guess-and-determine attack can recover the internal state with a practical complexity when attackers can control and observe the partial information in the internal state at the same time. Such an event would not happen when the implementation respects nonce and never releases unverified plaintexts. However, in case such implementation issues happens, attackers can recover the secret key with practical complexity.

We think the risk above should be avoided. Consequently, we suggest the so-called generic construction such as [NRS14], where the authenticated encryption with associated data can be constructed by an IV-based symmetric-key cipher and a message authentication

Algorithm 1 Trivium-LE-MAC

```
1: procedure R($\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, m$)
2:     $z \leftarrow a_{87} \oplus a_{93} \oplus b_{66} \oplus b_{99} \oplus c_{75} \oplus c_{96}$
3:     $t_1 \leftarrow a_{87} \oplus a_{93} \oplus a_{88} \cdot a_{89} \oplus b_{90}$
4:     $t_2 \leftarrow b_{66} \oplus b_{99} \oplus b_{67} \cdot b_{68} \oplus c_{87} \oplus m$
5:     $t_3 \leftarrow c_{75} \oplus c_{96} \oplus c_{76} \cdot c_{77} \oplus a_{78} \oplus m$
6:     $b_6 \leftarrow b_6 \oplus m$
7:     $(a_1, a_2, \ldots, a_{93}) \leftarrow (t_3, a_1, \ldots, a_{92})$
8:     $(b_1, b_2, \ldots, b_{99}) \leftarrow (t_1, b_1, \ldots, b_{98})$
9:     $(c_1, c_2, \ldots, c_{96}) \leftarrow (t_2, c_1, \ldots, c_{95})$
10:    return $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, z)$

11: procedure $\bar{\text{P}}$($\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$)
12:    $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, z) \leftarrow$ R$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, 1)$
13:    for $i = 2$ to 1152 do
14:        $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, z) \leftarrow$ R$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, 0)$

15:    return $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$
```

```
1: procedure Load($\boldsymbol{N}, \boldsymbol{K}$)
2:     $\boldsymbol{a} \leftarrow (K_1, \ldots, K_{80}, 0, 0, \ldots, 0)$
3:     $\boldsymbol{b} \leftarrow (N_1, \ldots, N_{80}, 0, 0, \ldots, 0)$
4:     $\boldsymbol{c} \leftarrow (0, \ldots, 0, 1, 1, 1)$
5:     return $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$

6: procedure TriMAC($\boldsymbol{N}, \boldsymbol{K}, \boldsymbol{M}$)
7:     $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \leftarrow \bar{\text{P}}$(Load$(N, K)$)
8:     $len \leftarrow$ the bit length of $M$.
9:     for $i = 1$ to $len$ do
10:        $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, z) \leftarrow$ R$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, M_i)$

11:    $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \leftarrow \bar{\text{P}}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$
12:    for $i = 1$ to 64 do
13:        $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, T_i) \leftarrow$ R$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, 0)$

14:    return $\boldsymbol{T}$
```

code (MAC). Due to the page limitation, we do not show a concrete combining methodology here. We suggest a reasonable choice based on the generic construction in Appendix D.

5 Generalization to Other Stream Ciphers

In this section, we study 4 Trivium-like ciphers. We will show that the circuit tree phenomenon translates seamlessly onto these more complex Trivium variations.

Trivium-MB. This construction was proposed by Maximov and Biryukov [MB07] and adds an additional noise term, i.e., a two-input AND gate, to each strand equation t_1, t_2, t_3 which is then backward connected to the register's input. The keystream output function remains unchanged. We denote it by Trivium-MB. Each register update function t_i is of the type

$$x_a + x_b + (x_c \cdot x_d) + (x_e \cdot x_f) + x_g,$$

implying that each $T_i(r)$ has seven child nodes instead of five.

TriviA. The stream cipher by Chakraborti et al. [CCHN18] was used in the authenticated encryption scheme of the same name. It also features a key size of 128-bit alongside a 96-bit initialization vector but exhibits an increased 384-bit internal state partitioned into three chunks of sizes 132, 105 and 147. Unlike Trivium and Maximov's construction, it adds a non-linear term to the keystream function in the form of a two-input AND gate. Each node in the circuit graph of TriviA also has 5 child nodes as in Trivium.

Kreyvium. Kreyvium is a stream cipher designed by Canteaut et al. [CCF+18] explicitly for the use in fully homomorphic encryption schemes. The cipher has the same structure and tap locations as Trivium, however a 128-bit security is achieved by additionally XORing bits from the key and IV to the update functions. It was shown in [BMA+18], that Kreyvium circuits are most energy-efficient at degrees of unrolling that are multiples of 128. This is because the circuit does not require additional shift registers to rotate the key/IV bits to produce the required bits in the update function. In our experiments, the cipher is unrolled 256 times.

Triad-SC. This construction was proposed by Banik et al. [BIM+19] as a low-energy alternative to Trivium. It has a much smaller state size (256 bits) and aims to provide

112-bit security. It counters the guess-and-determine attacks by using one additional AND gate over and above the original architecture of Trivium. The update functions in Triad-SC are asymmetric: t_1 is of the form $x_a + x_b + (x_c \cdot x_d) + (x_e \cdot x_f) + x_g$ whereas t_2, t_3 are of the form $x_a + x_b + (x_c \cdot x_d) + x_e$. Hence, $T_1(r)$'s are 7-ary trees and T_2, $T_3(r)$'s are 5-ary trees.

We performed a similar experiment for all the above ciphers as for the original Trivium: (a) We synthesized the circuit in restricted mode and record the power consumed in each strand. Results presented in Figure 14 show that strands associated with perfect trees consume much less power that the strands with imperfect trees. And (b) we generated numerous random instances of these ciphers with different tap locations. For every instance we plot power consumed vs number of perfect trees in the circuit tree graph. The results are plotted in Figure 13. The results are indeed on expected lines: there is strong negative correlation between energy consumed and number of perfect trees. For space constraints, in the figure we plot results only for the TSMC 90 nm cell library (with power measured at 10 MHz), however extrapolating the results from Section 3.2 we can make a similar argument that the results are neither library nor frequency specific.

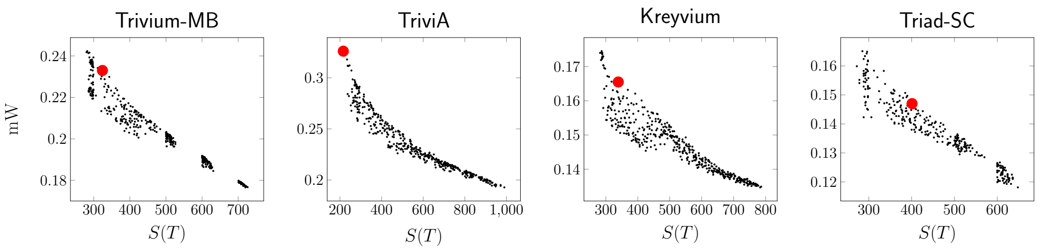


Figure 13: Power consumption figures as a function of the number of perfect trees for Trivium-like schemes. All data was obtained using the TSMC 90 nm process at a clock frequency of 10 MHz. Red data points mark the original schemes.

5.1 Applicability to Grain-128

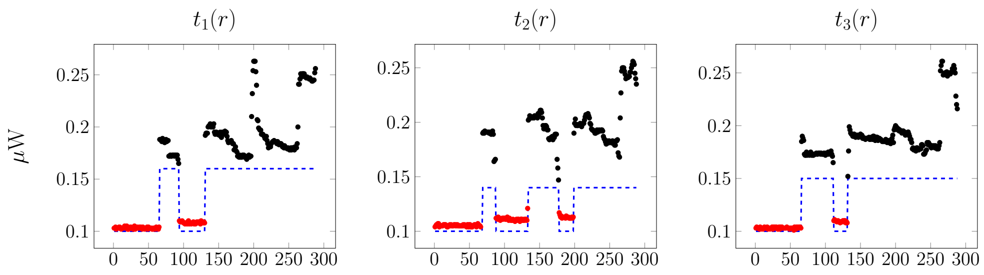
The concept of a circuit strand seamlessly translates over to Grain-128 [HJMM06] whose round function consists of two distinct strands that update two registers $b_1, b_2, \ldots, b_{128}$ and $s_1, s_2, \ldots, s_{128}$ such that

$$(x'_1, \ldots, x'_{128}) \leftarrow (f, x_1, \ldots, x_{127})$$
$$(y'_1, \ldots, y'_{128}) \leftarrow (g, y_1, \ldots, y_{127}),$$

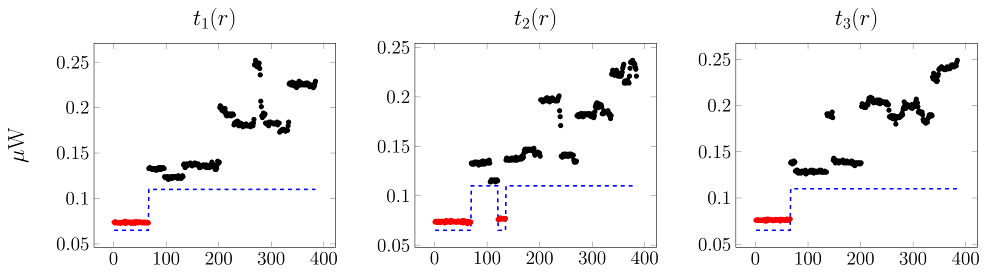
where f and g are linear and non-linear functions respectively defined as follows:

$$\begin{aligned} f &= x_{128} + x_{121} + x_{90} + x_{58} + x_{47} + x_{32}, \\ g &= x_{128} + y_{128} + y_{102} + y_{72} + y_{37} + y_{32} + y_{44}y_{60} + y_{61}y_{125} \\ &\quad + y_{63}y_{67} + y_{69}y_{101} + y_{80}y_{88} + y_{110}y_{111} + y_{115}y_{117} \end{aligned}$$

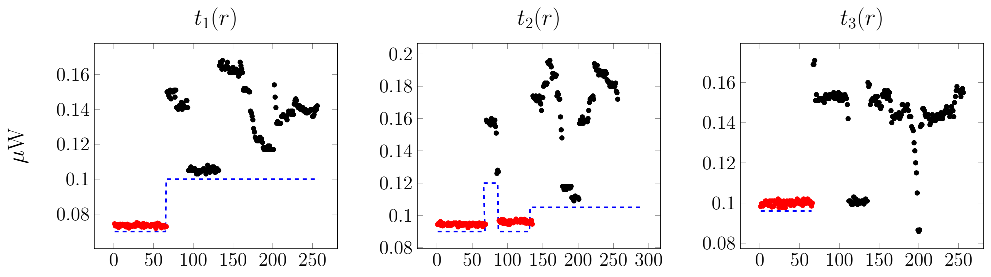
Evidently, the complexity of the update function in this family is higher than in Trivium and so finding a sensible restricted circuit configuration for these complex strands is a harder task. Even if we define the strand as a sub-circuit for the f, g functions, it is not immediately clear which configuration of gates is the best way to construct each strand. We can however delegate this responsibility to the circuit compiler, so that in the restricted mode it still respects the boundary between the strands, but chooses the internal structure of the strand independently. In Figure 15, we repeat the experiments from Section 2 by letting the synthesizer choose the circuit for each individual strand of f and g in *Restricted*



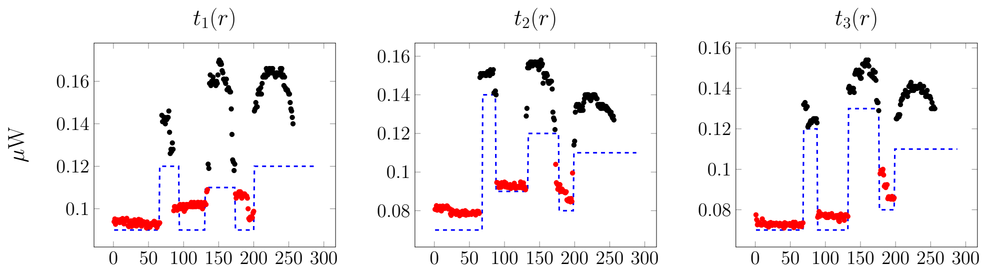
(a) Trivium-MB [MB07]



(b) TriviA [CCHN18]

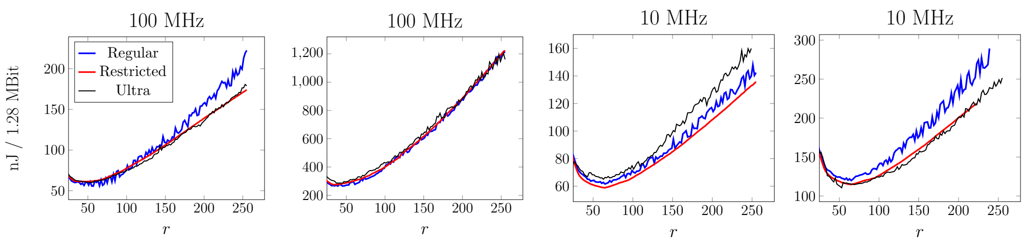


(c) Kreyvium [CCF+18]



(d) Triad-SC [BIM+19]

Figure 14: Power consumption measurements for all the unrolled strand trees for the 128-bit key size variation of Trivium proposed by Maximov and Biryukov [MB07], the state update function of the TriviA stream cipher [CCHN18], Kreyvium [CCF+18] and Triad-SC [BIM+19]. The Measurements were obtained using the TSMC 90 nm process at a frequency of 10 MHz.



(a) NanGate 15 nm **(b)** NanGate 45 nm **(c)** UMC 65 nm **(d)** TSMC 90 nm

Figure 15: Grain-128 energy measurements for three synthesis settings and cell libraries. The complete set of plots for different frequencies is given in Appendix F, Figure 19.

Table 2: Subterranean-Deck energy measurements for $r = 4$ for four cell libraries.

	nJ/1.28 MBit		
	Regular	Ultra	Restricted
NanGate 15 nm (100 MHz)	162.9	153.0	**150.5**
NanGate 45 nm (100 MHz)	341.2	389.1	**326.6**
UMC 65 nm (10 MHz)	129.9	156.1	**107.7**
TSMC 90 nm (10 MHz)	222.3	206.5	**192.1**

mode instead of imposing a predefined set of logic gates. The results in Figure 15 show that for Grain-128 suggests that restricted mode performs at least on par with other synthesis modes indicating that further optimizations for the restricted mode are possible by finding better circuit configurations. Additionally, by repeating the experiments from Section 3.2, we observe that increasing the number of perfect unrolled strand trees also correlates with the power consumption although in a weaker form, hence our results are also applicable to stream ciphers whose state update functions are significantly more complicated than in Trivium-like ciphers. Due to space constraints, we refer the reader to Figure 19 and Figure 20 in Appendix F.

5.2 Applicability to Subterranean-Deck

Unlike Trivium and Grain, the Subterranean-Deck [DMMR20] stream cipher does not feature a rotating register but in each round each state bit $x_1, x_2, \ldots, x_{257}$ is replaced by the output of a single strand that is replicated 257 times such that

$$(x'_1, \ldots, x'_{257}) \leftarrow (T_{\pi(1)}, T_{\pi(2)}, \ldots, T_{\pi(257)})$$
$$T_i \leftarrow (x_i + (x_{i+1} + 1)x_{i+2}) + (x_{i+3} + (x_{i+4} + 1)x_{i+5}) + (x_{i+8} + (x_{i+9} + 1)x_{i+10}),$$

for some permutation π. This strand can be realized in 3 NOT, 3 NAND, 1 XNOR3 and 1 XOR4 gates for feature-rich cell libraries and with 3 NOT, 3 NAND and 4 XOR2 for the more basic libraries. Denote $t_i = (x_i + (x_{i+1} + 1)x_{i+2})$. Each T_i consists of 3 sub-strands t_i, t_{i+3}, t_{h+8} of exactly equal circuit complexity. In restricted mode, if each T_i is compiled separately, then the sub-strand t_i is replicated 3 times in the strands T_i, T_{i-3}, T_{i-8}. This increases the circuit size three times, and also adds to unnecessary power consumption. Instead, if we choose t_i (in place of T_i) as the minimal unit whose compilation is restricted, then this replication can be avoided, and this is precisely what we do here. This simplicity lends itself well to the restricted mode of synthesis as shown in Table 2 and this mode is decidedly better energy-wise.

A unique property of this structure is that all the sub-strands t_i that constitute the round-function at any level are perfect trees in the corresponding circuit graph. As a

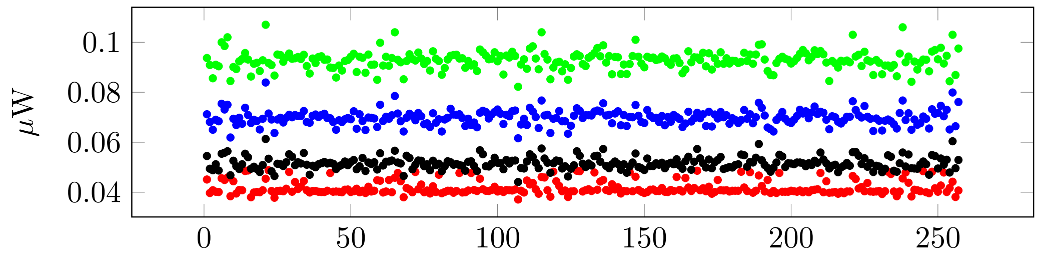


Figure 16: Power plot for 4-round unrolled Subterranean-Deck (TSMC 90 nm at 10 MHz). The points in red, black, blue and green represent the power consumed by the strands t_i in the first, second, third, and fourths levels of the round function.

result, it is expected that all the sub-strands at a given level consume similar energy. This is borne out by simulations performed on 4-round unrolled Subterranean-Deck as shown in Figure 16.

5.3 Triad-LE

Similarly to the case of the design of Trivium-LE(F) and Trivium-LE(S) based on Trivium, we present a low-energy variant of Triad-SC [BIM+19], which we call Triad-LE, that consumes around 5% less energy than Triad-SC. Triad-SC's update function has similar structure to Trivium, but the linear tap locations are chosen with respect to the multiple-of-2 property instead of the multiple-of-3 property. Each strand basically has the form $X_i^{op} = X_i^{\ell} \oplus (X_i^{op-1} \wedge X_i^{A_i}) \oplus X_{i+1}^{f}$, but to enhance the security level, the first strand is of the form $X_1^{op} = X_1^{\ell} \oplus (X_1^{op-1} \wedge X_1^{A_1}) \oplus X_2^{f} \oplus (X_2^{op-3} \wedge X_3^{op-3})$. By comprehensive cryptanalysis and evaluation using the number of perfect trees, the following parameters are adopted in Triad-LE.

$$(X_1^{\ell}, X_1^{f}, X_1^{op} \ ; \ X_2^{\ell}, X_2^{f}, X_2^{op} \ ; \ X_3^{\ell}, X_3^{f}, X_3^{op}) = (68, 76, 80 \ ; \ 64, 84, 88 \ ; \ 78, 74, 88).$$
$$(A_1, A_2, A_3) = (73, 67, 85).$$

Except for changing these tap positions, the specification is inherited from the original Triad-SC. Due to the page limit, we do not show the detail of the analysis here, and they are provided in Appendix E. As far as we apply several known attacks, the immunity of Triad-LE against the known attacks is almost equivalent with that of the original Triad-SC.

6 Conclusion

In this paper, we make some fundamental observations about the energy consumption of hardware-targeted stream ciphers and propose the first heuristic energy model that is based on the novel perfect tree metric. Our model is both simple and widely applicable to a wide range of stream ciphers and thus enables designers of future algorithms to specifically optimize for the energy consumption. The perfect tree energy model finds direct application in Trivium-LE(F) and Trivium-LE(S) that stand as the most energy-efficient encryption primitives known in the literature with a 10-15% (resp. around 25 %) margin to the next best cipher. A complete summary of all measurements is given in Table 3. Finally, we extend the reach of our model beyond stream ciphers and propose a novel, energy-efficient MAC Trivium-LE-MAC that can then be used to bootstrap an energy-efficient AEAD mode.

Table 3: Measurements summary for all investigated stream ciphers.

Scheme	Library	Area (GE)	Power (μW)					Energy (nJ/1.28 Mbit)				
			0.2 MHz	1 MHz	10 MHz	100 MHz	1 GHz	0.2 MHz	1 MHz	10 MHz	100 MHz	1 GHz
Trivium	NanGate 15 nm	10834	–	46.23	127.1	936.1	9026	–	205.7	56.39	41.64	40.15
($r = 288$)	NanGate 45 nm	9521	–	577.5	932.1	4477	39980	–	2569	414.6	199.2	177.2
	UMC 65 nm	9911	5.579	13.15	98.36	905.1	–	124.1	58.51	43.75	42.83	–
	TSMC 90 nm	9757	7.015	19.85	164.3	1609	–	155.8	88.29	73.08	71.57	–
Trivium-LE(F)	NanGate 15 nm	10834	–	45.35	118.3	848.4	8147	–	201.7	52.62	37.73	**36.24**
($r = 288$)	NanGate 45 nm	9521	–	574.5	901.0	4166	36863	–	2555	400.8	185.3	**163.9**
	UMC 65 nm	9911	5.391	12.21	88.97	856.5	–	119.3	54.34	39.57	**38.10**	–
	TSMC 90 nm	9757	6.659	18.12	147.0	1436	–	148.1	80.60	65.39	**63.89**	–
Trivium-LE(S)	NanGate 15 nm	10834	–	44.42	108.2	746.3	7127	–	197.6	48.13	32.68	**31.70**
($r = 288$)	NanGate 45 nm	9521	–	568.9	845.1	3608	31247	–	2531	376.1	160.5	**138.9**
	UMC 65 nm	9911	5.238	11.30	80.51	767.2	–	115.9	50.23	35.81	**34.13**	–
	TSMC 90 nm	9757	6.411	16.08	132.1	1311	–	142.58	71.53	58.76	**58.33**	–
Triad-SC	NanGate 15 nm	10834	–	44.88	121.6	889.1	8564	–	224.6	60.84	44.49	42.85
($r = 256$)	NanGate 45 nm	9199	–	561.8	918.8	4488	40075	–	2811	459.7	224.6	201.5
	UMC 65 nm	9487	5.035	11.79	87.79	847.5	–	126.0	59.01	43.39	42.42	–
	TSMC 90 nm	9350	6.422	17.90	147.1	1438	–	160.7	89.61	73.62	72.02	–
Triad-LE	NanGate 15 nm	10834	–	44.59	118.6	859.4	8266	–	223.1	59.39	43.00	41.36
($r = 256$)	NanGate 45 nm	9199	–	560.0	900.9	4309	38450	–	2802	450.8	215.6	192.4
	UMC 65 nm	9487	4.985	11.53	85.27	822.2	–	124.7	57.74	42.66	41.16	–
	TSMC 90 nm	9350	6.348	17.53	143.4	1400	–	158.8	87.77	71.17	70.01	–
Trivium-MB	NanGate 15 nm	13635	–	57.93	185.5	1460	14218	–	257.7	82.51	64.99	63.25
($r = 288$)	NanGate 45 nm	12132	–	768.9	1470	8487	75790	–	3402	654.1	377.6	335.9
	UMC 65 nm	12287	6.886	15.95	118.0	1138	–	153.2	70.99	52.49	50.66	–
	TSMC 90 nm	12783	9.798	28.02	233.0	2283	–	217.9	124.6	103.6	101.6	–
TriviA	NanGate 15 nm	15340	–	71.15	231.0	1829	17813	–	237.4	77.07	61.02	59.43
($r = 384$)	NanGate 45 nm	13440	–	839.1	1565	8835	59172	–	2799	522.5	294.7	196.1
	UMC 65 nm	13892	8.869	23.24	184.9	1809	–	147.9	77.55	61.69	60.12	–
	TSMC 90 nm	13867	11.94	37.59	326.1	3210	–	199.3	125.4	108.8	107.2	–
Kreyvium	NanGate 15 nm	11043	–	50.69	143.5	1078	10425	–	250.5	71.81	53.95	52.16
($r = 288$)	NanGate 45 nm	9703	–	594.8	1015	5220	47308	–	2976	508.0	261.2	236.7
	UMC 65 nm	10083	5.333	12.10	93.22	900.4	–	133.4	62.59	46.65	45.06	–
	TSMC 90 nm	9927	7.128	20.06	165.5	1620	–	178.4	100.4	82.83	81.07	–
Subterranean-Deck	NanGate 15 nm	12344	–	58.51	199.6	1505	15722	–	585.2	199.6	150.6	157.2
($r = 4$)	NanGate 45 nm	11170	–	706.6	940.1	3327	–	–	7067	940.2	332.2	–
	UMC 65 nm	11620	5.756	14.05	107.6	1003	–	575.7	140.1	107.7	103.1	–
	TSMC 90 nm	11125	8.741	24.05	192.2	1930	–	874.2	240.6	192.3	193.0	–
Grain-128	NanGate 15 nm	8565	–	31.89	51.12	457.1	4401	–	645.1	103.5	92.65	89.71
($r = 64$)	NanGate 45 nm	7592	–	456.3	748.6	3671	–	–	9238	1516	743.1	–
	UMC 65 nm	7544	3.636	8.356	61.45	592.4	–	294.3	169.7	124.4	119.9	–
	TSMC 90 nm	7512	5.411	15.30	126.6	1239	–	438.4	309.6	256.4	250.3	–

Acknowledgments

Subhadeep Banik is supported by the Swiss National Science Foundation (SNSF) through the Ambizione Grant PZ00P2_179921. Bin Zhang is supported by the National Key R&D Research Program Grant 2017YFB0802504, the National Natural Science Foundation of China Grant 61572482 and the National Cryptography Development Fund Grand MMJJ20170107.

References

[BBI$^+$15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 411–436. Springer, Heidelberg, November / December 2015.

[BBR16] Subhadeep Banik, Andrey Bogdanov, and Francesco Regazzoni. Exploring energy efficiency of lightweight block ciphers. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 178–194. Springer, Heidelberg, August 2016.

[BDE$^+$13] Lejla Batina, Amitabh Das, Baris Ege, Elif Bilge Kavun, Nele Mentens, Christof Paar, Ingrid Verbauwhede, and Tolga Yalçin. Dietary recommendations for lightweight block ciphers: Power, energy and area analysis of recently developed architectures. In Michael Hutter and Jörn-Marc Schmidt, editors, *Radio Frequency Identification - Security and Privacy Issues 9th International Workshop, RFIDsec 2013, Graz, Austria, July 9-11, 2013, Revised Selected Papers*, volume 8262 of *Lecture Notes in Computer Science*, pages 103–112. Springer, 2013.

[BIM$^+$19] Subhadeep Banik, Takanori Isobe, Willi Meier, Yosuke Todo, and Bin Zhang. Triad v1: A lightweight aead and hash function based on stream cipher. *NIST Lightweight Cryptography Project*, 2019.

[BMA$^+$18] Subhadeep Banik, Vasily Mikhalev, Frederik Armknecht, Takanori Isobe, Willi Meier, Andrey Bogdanov, Yuhei Watanabe, and Francesco Regazzoni. Towards low energy stream ciphers. *IACR Trans. Symm. Cryptol.*, 2018(2):1–19, 2018.

[BS00] Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 1–13. Springer, Heidelberg, December 2000.

[CCF$^+$18] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. *Journal of Cryptology*, 31(3):885–916, July 2018.

[CCHN18] Avik Chakraborti, Anupam Chattopadhyay, Muhammad Hassan, and Mridul Nandi. TriviA and uTriviA: two fast and secure authenticated encryption schemes. *Journal of Cryptographic Engineering*, 8(1):29–48, April 2018.

[De 06] Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC 2006*, volume 4176 of *LNCS*, pages 171–186. Springer, Heidelberg, August / September 2006.

[DMMR20] Joan Daemen, Pedro Maat Costa Massolino, Alireza Mehrdad, and Yann Rotella. The subterranean 2.0 cipher suite. *IACR Trans. Symm. Cryptol.*, 2020(S1):262–294, 2020.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 278–299. Springer, Heidelberg, April 2009.

[HIJ+19] Yonglin Hao, Takanori Isobe, Lin Jiao, Chaoyun Li, Willi Meier, Yosuke Todo, and Qingju Wang. Improved division property based cube attacks exploiting algebraic properties of superpoly. *IEEE Trans. Computers*, 68(10):1470–1486, 2019.

[HJMM06] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. In *Proceedings 2006 IEEE International Symposium on Information Theory, ISIT 2006, The Westin Seattle, Seattle, Washington, USA, July 9-14, 2006*, pages 1614–1618. IEEE, 2006.

[HLM+20] Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset - improved cube attacks against Trivium and Grain-128AEAD. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 466–495. Springer, Heidelberg, May 2020.

[HSWW20] Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 446–476. Springer, Heidelberg, December 2020.

[KDH+12] Stéphanie Kerckhof, François Durvaux, Cédric Hocquet, David Bol, and François-Xavier Standaert. Towards green cryptography: A comparison of lightweight ciphers from the energy viewpoint. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 390–407. Springer, Heidelberg, September 2012.

[MB07] Alexander Maximov and Alex Biryukov. Two trivial attacks on Trivium. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *SAC 2007*, volume 4876 of *LNCS*, pages 36–55. Springer, Heidelberg, August 2007.

[NRS14] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, Heidelberg, May 2014.

[TIHM17] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 250–279. Springer, Heidelberg, August 2017.

[TIHM18] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. *IEEE Trans. Computers*, 67(12):1720–1736, 2018.

[TM16] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 357–377. Springer, Heidelberg, March 2016.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 287–314. Springer, Heidelberg, April 2015.

[WHT+18] Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 275–305. Springer, Heidelberg, August 2018.

A Trivium

The stream cipher was proposed by De Cannière and Preneel [De 06] and is an eSTREAM portfolio member. The construction is specifically tailored to constrained hardware devices and can thus be efficiently implemented on a small circuit area budget. It features a state of 288 bits and key size of 80 bits alongside an initialization vector of the same length. At the beginning of the initialization phase, the 80-bit secret key $K = (k_1, \ldots, k_{80})$ and the publicly known 80-bit initial vector $IV = (iv_1, \ldots, iv_{80})$ are loaded into the internal state $(x_1, \ldots, x_{288})$. Subsequently, the state update function is run for $4 \cdot 288 = 1152$ iterations. During encryption, the keystream bits z_i are extracted from intermediate values of the state update equations. Both the initialization procedure and the keystream routine are shown in Algorithm 2.

Algorithm 2 Trivium

```
procedure Init($\boldsymbol{k}, \boldsymbol{iv}$)
    $(x_1, \ldots, x_{93}) \leftarrow (k_1, \ldots, k_{80}, 0, \ldots, 0)$
    $(x_{94}, \ldots, x_{177}) \leftarrow (iv_1, \ldots, iv_{80}, 0, \ldots, 0)$
    $(x_{178}, \ldots, x_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
    for $i = 1$ to $4 \cdot 288$ do
        $t_1 \leftarrow x_{66} + (x_{91} \cdot x_{92}) + x_{93} + x_{171}$
        $t_2 \leftarrow x_{162} + (x_{175} \cdot x_{176}) + x_{177} + x_{264}$
        $t_3 \leftarrow x_{243} + (x_{286} \cdot x_{287}) + x_{288} + x_{69}$
        $(x_1, \ldots, x_{93}) \leftarrow (t_3, x_1, \ldots, x_{92})$
        $(x_{94}, \ldots, x_{177}) \leftarrow (t_1, x_{94}, \ldots, x_{176})$
        $(x_{178}, \ldots, x_{288}) \leftarrow (t_2, x_{178}, .., x_{287})$
```

```
procedure KSG($\boldsymbol{x}$)
    for $i = 1$ to $N$ do
        $s_1 \leftarrow x_{66} + x_{93}$
        $s_2 \leftarrow x_{162} + x_{177}$
        $s_3 \leftarrow x_{243} + x_{288}$
        $z_i \leftarrow s_1 + s_2 + s_3$
        $t_1 \leftarrow s_1 + (x_{91} \cdot x_{92}) + x_{171}$
        $t_2 \leftarrow s_2 + (x_{175} \cdot x_{176}) + x_{264}$
        $t_3 \leftarrow s_3 + (x_{286} \cdot x_{287}) + x_{69}$
        $(x_1, \ldots, x_{93}) \leftarrow (t_3, x_1, \ldots, x_{92})$
        $(x_{94}, \ldots, x_{177}) \leftarrow (t_1, x_{94}, \ldots, x_{176})$
        $(x_{178}, \ldots, x_{288}) \leftarrow (t_2, x_{178}, .., x_{287})$
```

B Proof of Lemma 1

We prove Lemma by the means of induction.

Base Case: Consider $t_1(r)$ in the original Trivium, whose recursive description is given in Figure 6. We know that for $r = 1 \rightarrow 66 (= X_1^1)$, $t_1(r)$ can be written as functions of depth 0 nodes of the circuit i.e. the state variables $x_1, x_2, x_3, \ldots, x_{288}$, and it is easy to see that all $t_1(r),\ r \in [1, 66]$ are perfect depth 1 trees. For $r = 67$, $t_1(r)$ is expanded as $t_3(1) + x_{27} + x_{28} \cdot x_{29} + x_{105}$. Note that $t_3(1)$ is no longer a depth 0 node, and hence $t_1(67)$ is not a perfect tree. Also consider a sightly modified form of Trivium in which $X_2^f = 62$ (say). In this case the recursive definition of $t_1(r)$ is as follows:

$$t_1(r) = t_3(r - 66) + t_3(r - 93) + [t_3(r - 91) \cdot t_3(r - 92)] + t_1(r - 62)$$

Now it is easy to see that $t_1(r)$ is a perfect depth 1 tree only upto $r = 62$, as $t_1(63)$ will involve a $t_1(1)$ term which is no longer at depth 0. Thus the number of perfect depth 1 trees for $t_1(r)$ in a generalized Trivium circuit has to be the smaller of 66 and 62, i.e. $\min\left\{X_1^\ell, X_2^f\right\}$. Does this also depend on the tap position of the two AND gates and the final XOR term $t_3(r-93)$? The final XOR term must be tapped from the final location of each register to ensure that the state update function is one-to-one. So numerically, X_j^{op} has to be the length of the register X_j. Since X_j^{ℓ} is an intermediate location and X_j^{op} is the final location of register j, we always have $X_j^{\ell} < X_j^{op}$. If we select the tap locations for the AND gates in the range (X_j^{ℓ}, X_j^{op}), it is easy to see that the perfect depth 1 trees only occur till the smaller of X_1^{ℓ} and X_2^{f}. Even if the the tap location of one or both inputs to the AND gate is less than X_j^{1}, we can simply select the numerically smallest tap location of register X_j as X_j^{ℓ}, since in terms of the circuit graph it does not make a difference if X_j^{ℓ} is input to an XOR or an AND gate. However, here we have the AND taps strictly in between X_j^{ℓ} and X_j^{op} and so the the actual locations do of the AND taps not make a difference. Thus it is pretty easy to see base case for our recursive formula $f_1(X_j)=0$ and $g_1(X_j)=\min\left\{X_j^{\ell}, X_{j+1}^{f}\right\}$.

Inductive Step: Now let us assume the inductive hypothesis, i.e. $g_l,\ f_l$ are as defined in the Lemma statement for $t=1,2,3,\ldots,l-1$. Consider the equation for $t_1(r)$ at $r=r_0=f_{l-1}(X_3)+X_1^{op}$ and $r=r_0+1$. For conciseness, denote by the symbol α the value of $f_{l-1}(X_3)$ and $\Delta = X_1^{op}-X_1^{\ell}$. It holds (note a_1, a_2 are the AND gate taps with $X_1^{\ell} < a_1, a_2 < X_1^{op}$)

$$\begin{aligned}
t_1(r_0) &= t_3(r_0-X_1^{\ell}) + t_3(r_0-X_1^{op}) + [t_3(r_0-a_1)\cdot t_3(r_0-a_2)] + t_1(r_0-X_2^{f}) \\
&= t_3(\alpha+\Delta) + t_3(\alpha) + [t_3(\alpha+(X_1^{op}-a_1))\cdot t_3(\alpha+(X_1^{op}-a_2))] \\
&\quad + t_1(\alpha+(X_1^{op}-X_2^{f})) \\
t_1(r_0+1) &= t_3(\alpha+\Delta+1) + t_3(\alpha+1) \\
&\quad + [t_3(\alpha+1+(X_1^{op}-a_1))\cdot t_3(\alpha+1+(X_1^{op}-a_2))] + t_1(\alpha+1+(X_1^{op}-X_2^{f}))
\end{aligned}$$

Note that by the inductive hypothesis, $t_3(\alpha)$ corresponds to a depth $l-2$ tree, whereas $t_3(\alpha+1)$ corresponds to a depth $l-1$ tree. All other t_3 terms in the above expressions are depth $l-1$ trees or greater by the inductive hypothesis. If $t_1(\alpha+(X_1^{op}-X_2^{f}))$ also corresponds to a depth $l-1$ tree, it is easy to see that r_0+1 is the first value of r for which $t_1(r)$ produces a perfect depth l tree. However that is always not the case. It may so happen that $t_1(\alpha+(X_1^{op}-X_2^{f}))$ still corresponds to a depth $l-2$ tree for certain specific instances of the generic Trivium circuit. In such cases the value of r has to be equal to $u=f_{l-1}(X_1)+X_2^{f}+1$ to ensure that the t_1 term in the expression for $t_1(r)$ also produces a depth $l-1$ tree by the inductive hypothesis. This is true since $t_1(u-X_2^{f})=t_1(f_{l-1}(X_1)+1)$ which corresponds to a depth $l-1$ tree by the inductive hypothesis.

For $t_1(r)$ to definitely correspond to a depth l perfect tree both the depth conditions on the above t_3 and t_1 nodes must be satisfied. This leads us to the easy conclusion that the first value of r for which $t_1(r)$ is a perfect depth r tree is the maximum of $f_{l-1}(X_3)+X_1^{op}+1$ and $f_{l-1}(X_1)+X_2^{f}+1$. Generalizing over all configurations of n-stage registers, we have

$$f_l(X_j) = \max\left\{f_{l-1}(X_{j-1}) + X_j^{op},\ f_{l-1}(X_j) + X_{j+1}^{f}\right\}.$$

Now to prove the recursive expression for g_l, consider again $t_1(r)$ for the generic Trivium circuit for $r=r_1=g_{l-1}(X_3)+X_1^{\ell}$ and $r=r_1+1$. For conciseness, denote by the symbol

β the value of $g_{l-1}(X_3)$. It holds

$$\begin{aligned}
t_1(r_1) &= t_3(r_1 - X_1^{\ell}) + t_3(r_1 - X_1^{op}) + [t_3(r_1 - a_1) \cdot t_3(r_1 - a_2)] + t_1(r_1 - X_2^{f})\\
&= t_3(\beta) + t_3(\beta - \Delta) + [t_3(\beta - \Delta + (X_1^{op} - a_1)) \cdot t_3(\beta - \Delta + (X_1^{op} - a_1))]\\
&\quad + t_1(\beta - \Delta + (X_1^{op} - X_2^{f}))\\
t_1(r_1+1) &= t_3(\beta+1) + t_3(\beta + 1 - \Delta)\\
&\quad + [t_3(\beta + 1 - \Delta + (X_1^{op} - a_1)) \cdot t_3(\beta + 1 - \Delta + (X_1^{op} - a_2))]\\
&\quad + t_1(\beta + 1 - \Delta + (X_1^{op} - X_2^{f}))
\end{aligned}$$

By the inductive hypothesis $t_3(\beta+1)$, no longer corresponds to a perfect tree of depth $l-1$. Assuming that $t_3(\beta-\Delta), t_3(\beta - \Delta + (X_1^{op} - a_1)), t_3(\beta - \Delta + (X_1^{op} - a_1))$ and $t_1(\beta - \Delta + (X_1^{op} - X_2^{f}))$ do correspond to perfect depth $l-1$ trees, $r_1 = g_{l-1}(X_3) + X_1^{\ell}$ is of course the largest value of r that produces depth l trees.

There are 2 assumptions made in the above proof which may not always hold for all configurations of the generic Trivium circuit. The first is if $t_3(\beta - \Delta)$ does not correspond to a perfect depth $l-1$ tree (note if $t_3(\beta-\Delta)$ is not a perfect depth $l-1$ tree, neither of the AND terms will correspond to depth $l-1$ trees since their index values are larger than $\beta - \Delta$). The above happens when

$$\beta - \Delta \leq f_{l-1}(X_3) \Rightarrow g_{l-1}(X_3) - f_{l-1}(X_3) - (X_1^{op} - X_1^{\ell}) \leq 0$$

The above condition essentially means that the number of perfect depth $l-1$ trees for $t_3(r)$ is less than or equal to $X_1^{op} - X_1^{\ell}$. This implies that the terms $t_3(r - X_1^{\ell})$ and $t_3(r - X_1^{op})$ can never be both of depth $l-1$, which in turn implies that the expression for $t_1(r)$ can never produce a depth l tree. In this case we can simply set $g_l(X_1)$ to be some value less than or equal to $f_l(X_1)$ to indicate this impossibility. We can simply pick one of the expressions for $f_l(X_1)$, i.e. $f_{l-1}(X_3) + X_1^{op}$ for this purpose. Combining the two assumptions we can write the new expression as $f_{l-1}(X_3) + X_1^{op} + \left[(g_{l-1}(X_3) - f_{l-1}(X_3)) - (X_1^{op} - X_1^{\ell})\right]^{+}$

The second assumption was that t_1 term also produces a perfect depth $l-1$ tree. For a generic Trivium circuit, this assumption may be false. The term $t_1(r - X_2^{f})$ will produce a depth $l-1$ tree if $r - X_2^{f} \leq g_{l-1}(X_1) \Rightarrow r \leq X_2^{f} + g_{l-1}(X_1)$. Since we need both depth conditions to be satisfied, we take minimum of the above two. Generalizing for all n stage Trivium circuits we have

$$\begin{aligned}
g_l(X_j) = \min \Big\{ & f_{l-1}(X_{j-1}) + X_j^{op} + \left[(g_{l-1}(X_{j-1}) - f_{l-1}(X_{j-1})) - (X_j^{op} - X_j^{\ell})\right]^{+},\\
& g_{l-1}(X_j) + X_{j+1}^{f} \Big\},
\end{aligned}$$

This completes the proof.

C Trivium-LE(S) Security Analysis

Trivium-LE(S) is another low-energy variant of Trivium and results in around 25% lower energy when compared with Trivium.

$$(X_1^{\ell}, X_1^{f}, X_1^{op}\ ;\ X_2^{\ell}, X_2^{f}, X_2^{op}\ ;\ X_3^{\ell}, X_3^{f}, X_3^{op}) = (96, 87, 99\ ;\ 87, 93, 96\ ;\ 75, 90, 93).$$

The tap location for AND gates is $X_i^{\ell} + 1$ and $X_i^{\ell} + 2$.

Security Analysis. We inherit the security of Trivium-LE(F) against the TMD tradeoff attack because it does not exploit the tap location. Here, we discuss the security against

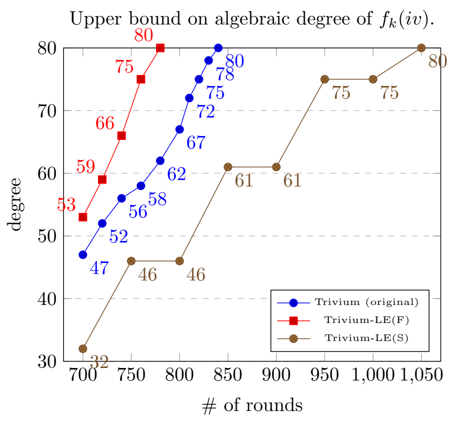


Figure 17: Increase in algebraic degree with respect to the number of initialization rounds.

the correlation attack, the Maximov/Biryukov's guess-and-determine attack, and the cube attack.

Linear Distinguishing Attack. On this parameter, the maximum linear correlation is 2^{-48}, and the required data is about 2^{96} to distinguish the keystream from ideal one. Compared to 2^{144} in Trivium or Trivium-LE(F), the security margin is very narrow. However, it is still enough to achieve the claimed security, i.e., 80 bits, which is the same as Trivium.

Maximov/Biryukov's Guess-and-Determine Attack. Similarly to the case for Trivium-LE(F), we evaluated the number of collectable linear equations after guessing some outputs of AND gates. In the scenario T1, the time complexity is $c \cdot 2^{77.0503}$, where 37, 41, and 44 outputs of AND gates are guessed for each register. Considering $c \approx 2^{16}$, this attack never threatens the claimed security, i.e., 80 bits.

Cube Attack. We also investigated the increase in algebraic degree by using the bit-based division property. Figure 17 shows the upper bound of the algebraic degree of $f_k(iv)$. Trivium-LE(S) is clearly more vulnerable than Trivium and Trivium-LE(F) against the cube attack. The degree of even 1000 rounds does not reach 80. In other words, we can attack 1000-round Trivium-LE(S) with the use of any 76-dimensional cube. The upper bound reaches the full, i.e., 80, in 1050 rounds although the original Trivium reach the same level in 840 rounds. In other words, the increase of the degree is about 25% slower because $1050/840 = 1.25$. This is the main reason why we increase the number of rounds in the initialization from $288 \times 4 = 1152$ to $288 \times 5 = 1440$.

D AEAD Using Trivium-LE(F) and Trivium-LE-MAC

In this section, we present an authenticated encryption with associated data (AEAD) by combining Trivium-LE(F) and Trivium-LE-MAC with the so-called generic construction [NRS14], where the authenticated encryption with associated data can be constructed by an IV-based symmetric-key cipher and a message authentication code (MAC). In our case, Trivium-LE(F) is an IV-based symmetric-key cipher and Trivium-LE-MAC is a vecMAC accepting two inputs, i.e., IV and message. We recommend the use of the N1 scheme due to the parallelizable computation of the encryption and the MAC. However, to apply the N1 scheme, we have two problems: the first is the use of multiple keys, and another is how to involve associated data.

The first problem is the use of different keys for the encryption and the MAC, which is necessary in the context of the provable security. We recommend the use of multiple keys if possible, but as far as we evaluated, there is no attacks even if they use the same key for the encryption and the MAC because the initialization of Trivium-LE(F) and Trivium-LE-MAC is different.

For another problem, we need a vecMAC accepting three inputs, but Trivium-LE-MAC accepts only two inputs. Namprempre et al also showed the generic method to construct such a vecMAC from a normal MAC, but it calls a normal MAC several times [NRS14]. Such construction is not efficient for our case because the cost by the initialization of Trivium-LE-MAC is relatively high. Instead, to realize such a vecMAC efficiently, we treat a message m and an associated data ad as a single message with appropriate domain separation. Specifically, we use the following map

$$encoder : (ad, m) \to (ad \| len(ad) \| m \| len(m)),$$

where $len(ad)$ and $len(m)$ are 7-byte values representing the byte length of ad and m, respectively. This implies that associated data and message accepts at most 2^{56} bytes. Note that the AEAD only accepts byte strings and never accepts bit strings whose length is not the multiple of 8. When *encoder* is injective, we never have two different (ad, m) which are encoded to the same single message. To prove it, we confirm the *decoder* corresponding to the *encoder* is uniquely determined. We assume a byte string x is the output of the *encoder*. Moreover, $len(x)$ denotes the byte length of x, and $x[i]$ denotes the ith byte of x. We first extract the last 7-byte of x and get the byte length of m, i.e.,

$$\begin{aligned} len(m) = (&x[len(x)-6] \| x[len(x)-5] \| x[len(x)-4] \| x[len(x)-3] \| x[len(x)-2] \\ &\| x[len(x)-1] \| x[len(x)]. \end{aligned}$$

Since the byte length of m is determined, we can extract a message as

$$m = (x[len(x)-6-len(m)], \ldots, x[len(x)-6-2], x[len(x)-6-1]).$$

We next extract the byte length of ad as

$$len(ad) = (x[len(x)-6-len(m)-6] \| \cdots \| x[len(x)-6-len(m)],$$

and the remaining $(x[0], \ldots, x[len(x)-6-len(m)-6-1])$ represents an associated data. Since the *decoder* corresponding to the *encoder* is uniquely determined, *encoder* is injective.

E Triad-LE

Triad-SC was proposed by Banik et al. [BIM+19] as a low energy alternative to Trivium. It has a much smaller state size (256 bits) and aims to provide 112-bit security. It counters guess and determine attacks by using one additional AND gate over and above the original architecture of Trivium. It uses a 128-bit key arranged bytewise as $(k[1], k[2], \ldots, k[16])$ and 96-bit IV $(iv[1], iv[2], \ldots, iv[12])$, where each $k[i], iv[j]$ are the i^{th} key byte and j^{th} IV byte respectively. Algorithm 3 lists the update function of Triad-SC. It uses 1024 rounds for initialization after the Key-IV setup.

To begin this section, we try to enumerate the number of perfect trees for a generic Triad-SC architecture.

Definition 6. Denote by Triad$(X, 3)$ a generic Triad-SC configuration composed of 3 chained registers $(X_1, \ldots, X_n)$ such that X_j^{ℓ} is the register's leftmost forward tap, X_j^f is a feedback, X_j^{op} is the output tap and Y, Z are two additional non-linear taps that feed into the output of register X_1.

Algorithm 3 Triad-SC update routine.

```
$(s_1, \ldots, s_{80}) \leftarrow (iv[1], k[5], 1^8, k[4], 1^8, k[3], 1^8, k[2], 1^7 0, k[1])$
$(s_{81}, \ldots, s_{168}) \leftarrow (iv[12], \ldots, iv[2])$
$(s_{169}, \ldots, s_{256}) \leftarrow (k[16], \ldots, k[6])$
for $i \leftarrow 1$ to $N$ do
    $u \leftarrow s_{68} + s_{80} + (s_{165} \cdot s_{253})$
    $v \leftarrow s_{144} + s_{168}$
    $w \leftarrow s_{236} + s_{256}$
    if $i > 1024$ then
        $z_i \leftarrow u + v + w$
    $t_1 \leftarrow u + (s_{73} \cdot s_{79}) + s_{146}$
    $t_2 \leftarrow v + (s_{145} \cdot s_{167}) + s_{252}$
    $t_3 \leftarrow w + (s_{245} \cdot s_{255}) + s_{74}$
    $(s_1, \ldots, s_{80}) \leftarrow (t_3, s_1, \ldots, s_{79})$
    $(s_{81}, .., s_{168}) \leftarrow (t_1, s_{81}, .., s_{167})$
    $(s_{169}, ., s_{256}) \leftarrow (t_2, s_{169}, .., s_{255})$
```

Corollary 1. *Given an arbitrary, generic* Triad$(X, 3)$ *circuit composed of* 3 *registers, the total number of perfect unrolled strand trees* $S(T)$ *in the fully unrolled setting is given by*

$$S(T) = \sum_{j=1}^{3} S(T_j) = \sum_{j=1}^{3} \sum_{l=1}^{3} (g_l(X_j) - f_l(X_j))^+ ,$$

where $y^+ = \max\{y, 0\}$ *and* $f_l(X_j)$*,* $g_l(X_j)$ *are recursively defined functions for* $1 \le l \le 3$ *of the form*

$$f_l(X_j) = \begin{cases} \max\Big\{f_{l-1}(X_3) + X_j^{op},\ f_{l-1}(X_1) + X_2^f,\ f_{l-1}(X_1) + Y, \\ \qquad f_{l-1}(X_2) + Z\Big\}, & \text{if } j = 1 \\ \max\Big\{f_{l-1}(X_{j-1}) + X_j^{op},\ f_{l-1}(X_j) + X_{j+1}^f\Big\}, & \text{otherwise.} \end{cases}$$

$$g_l(X_j) = \begin{cases} \min\Big\{f_{l-1}(X_3) + X_1^{op} + \big[(g_{l-1}(X_3) - f_{l-1}(X_3)) - (X_1^{op} - X_1^{\ell})\big]^+ , \\ \qquad g_{l-1}(X_1) + X_2^f,\ g_{l-1}(X_1) + Y,\ g_{l-1}(X_2) + Z\Big\}, & \text{if } j{=}1 \\ \min\Big\{f_{l-1}(X_{j-1}) + X_j^3 + \big[(g_{l-1}(X_{j-1}) - f_{l-1}(X_{j-1})) - (X_j^{op} - X_j^{\ell})\big]^+ , \\ \qquad g_{l-1}(X_j) + X_{j+1}^f\Big\}, & \text{otherwise.} \end{cases}$$

such that $f_1(X_j) = 0$ *and* $g_1(X_j) = \min\Big\{X_j^{\ell},\ X_{j+1}^f\Big\}$*. The number of perfect unrolled strand trees of depth* t *for the* j*-th strand is* $S(T_j)|_{depth=t} = (g_t(X_j) - f_t(X_j))^+$*.*

Proof. The addition of an AND gate that feeds into the output of X_1 can be regarded as two feedback taps. As a result, the same reasoning from Proof B applies. This means that two additional feedback terms are required for both $f_l(X_j)$ and $g_l(X_j)$, i.e., $f_{j-1}(X_1) + Y$ and $g_{j-1}(X_1) + Y$ for tap Y, similarly $f_{l-1}(X_2) + Z$ and $g_{j-1}(X_2) + Z$ for tap Z. □

E.1 Searching for More Energy-Efficient Parameters

We repeated the experiment that we did for generating energy-efficient candidates with the Trivium architecture. Again since the search space for tap locations is too large we focused on the following strategy.

A: We keep the multiple of 2 property of original Triad-SC (all linear taps are multiples of 2), for the same reason as in Trivium.

B: In Triad-SC, Y and Z were chosen to be equal to $X_2^{op} - 3$ and $X_3^{op} - 3$. Given the search space is large we decided to stick to this.

Table 4: List of configurations and associated security parameters. ϵ represent maximum linear bias. T is the complexity of guess and determine attack. c represents an additional cost required to do Gaussian elimination to solve a set of linear equations to recover the internal state. Y, Z were set to $X_2^{op} - 3, X_3^{op} - 3$ and one locations of the 3 and gates to $X_j^{op} - 1$. A_j represents the location of the other tap of the 3 AND gates. The first row represents the parameters for the original Triad-SC.

	Parameters												# Perfect Trees	Max Bias	G&D Complexity
#	X_1^ℓ	X_1^f	X_1^{op}	X_2^ℓ	X_2^f	X_2^{op}	X_3^ℓ	X_3^f	X_3^{op}	A_1	A_2	A_3		$-\log_2 \epsilon$	$\log_2 T$
1	68	74	80	64	66	88	68	84	88	73	65	77	401	72	$\log_2 c + 163.477$
2	68	76	80	64	84	88	78	74	88	73	67	85	469	72	$\log_2 c + 163.04$
3	82	68	84	66	86	90	70	78	82	77	65	73	473	72	$\log_2 c + 161.812$

C: For the taps for the other AND gates, one location was fixed at $X_j^{op} - 1$ (as in original Triad-SC) and the other location was searched for exhaustively.

D: Under the condition where the output of each AND gate is approximated to 0, we denote by ϵ the maximum correlation in a linear combination of keystream bits. Moreover, we also denote by $\#ACT_{and}$ the number of active AND gates. In Triad-SC, $\#ACT_{and} = 96$ and $\epsilon = 2^{-72}$. We inherited this condition.

We first computed the number of perfect trees using the recursion formula above. We thereafter chose candidates with minimum linear bias 2^{-72} and then check for security against guess and determine attacks. Table 4 details the 2 best candidates we found.

Although the candidate in row 3, has the maximum number of perfect trees, it would result in 3 registers of size 84, 90 and 82 bits each. The original Triad-SC had register lengths that are multiples of 8, so that it could be deployed efficiently on 8-bit processors. Since the candidate in row 2 satisfies this criterion, and also since the energy consumption resulting from 469 and 473 perfect trees are almost equal, we select this candidate for further analysis. We call this candidate Triad-LE. Note that we keep the key-IV setup and initialization same as in the original Triad-SC. To round off this section we perform a preliminary security analysis.

E.2 Security Analysis

Time-Memory-Data Trade-off Attack. In [BS00], the authors had mounted a table based generic TMD tradeoff attack against stream ciphers for which the tradeoff curve is $TM^2D^2 = N^2$ with $T \geq D^2$ and $P = N/D$ where T = Online time complexity, M = Memory complexity, D = Data complexity, N = Cardinality of the set of internal states and P= Offline Time complexity. From this can be easily seen that if K is the complexity of exhaustive search against the same cipher then $N \geq K^2$ ensures that there does not exist any point on the tradeoff curve for which both P and T are less than K. In other words if the size of the internal state of the cipher is atleast twice the desired security level in bits, then a TMD tradeoff attack can not be applied on it. We can see that Trivium, Trivium-LE, Triad-SC and Triad-LE easily satisfy this condition.

However we do have to take into account the sampling resistance of Triad-LE too. It can be verified that the sampling resistance of Triad-LE is $R = 2^{-64}$. In that case the new tradeoff curve will be $TM^2D^2 = N^2$ with $T \geq R^2D^2$ and $P = N/D$. It can be easily seen that there does not exist any point on the tradeoff curve that has all $P, T, D \leq 2^{112}$.

Linear Distinguishing Attack. As we already discussed in Sect. E.1, the correlation of the best linear distinguisher is 2^{-72} when outputs of AND gates is approximated to 0. While we unlikely to find better distinguisher due to the multiple-of-2 property, we heuristically evaluated the case where these outputs are not approximated to 0. As we expected, we cannot find better linear distinguishers.

Maximov and Biryukov's Guess-and-Determine Attack. Maximov/Biryukov's guess-and-determine attack mainly exploits the multiple-of-3 property in Trivium. Therefore, this attack cannot be applied to Triad-SC or Triad-LE directly. However, the same attack strategy can be extended to configuration with the multiple-of-2 property. We first divides the internal state into two sub states and consists of two phases. In the first phase, we first guess one of two sub states at some time. In the second phase, assuming that the sub state is guessed correctly, we next recover the rest of the bits, i.e., $256 \times 2 = 128$ bits. Similarly to the original attack, we guess outputs of any AND gates and collect keystream bits, which are linearly represented by the internal state. Unlike Trivium, Triad-LE has an additional AND gate and we need to guess these outputs. Moreover, the output of the additional AND gate is always involved in the keystream, and it implies we cannot collect any linear equation for free. As a result, by guessing 85, 85, 77, and 92 outputs of AND gates are guessed, we can recover the 256-bit state, but the required keystream is $2^{159.04}$ and the time complexity is $c \cdot 2^{163.04}$.

Cube Attack. We also investigated the increase in algebraic degree by using the bit-based division property. Note that it is not enough to achieve 112-bit security even if the degree of $f_k(iv)$ is the full, i.e., 96. When the corresponding superpoly is low degree, 1-bit secret key information can be efficiently recovered. Therefore, unlike Trivium-LE, we focus on $f(k, iv)$ and evaluated the increase in algebraic degree. Figure 18 shows the upper bound of the algebraic degree, and it reaches degree 224 in 680 rounds. Considering that the full rounds are 1024, Triad-LE has plenty of security margin.

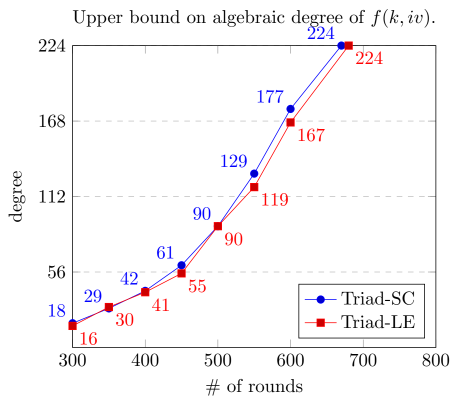
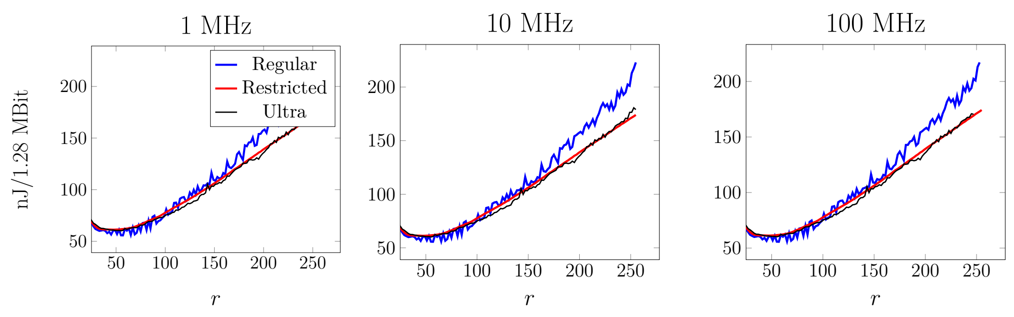
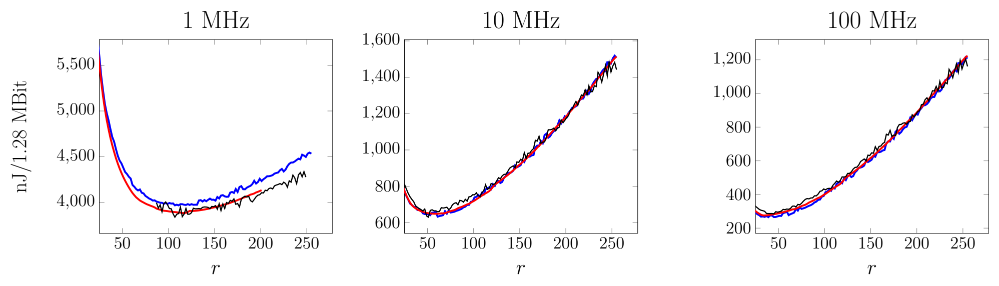


Figure 18: Increase in algebraic degree with respect to the number of initialization rounds.

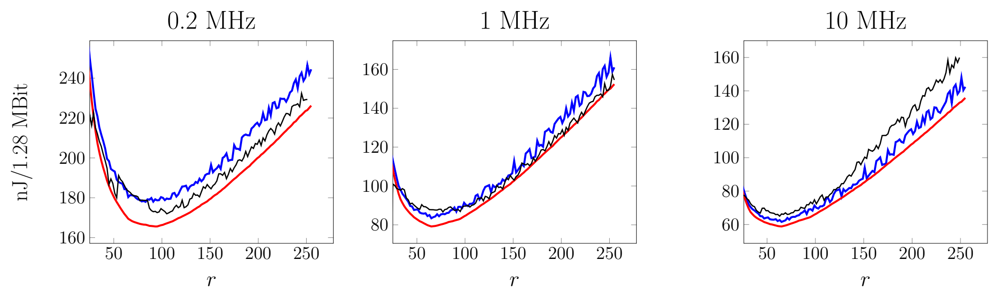
F Supplementary Plots



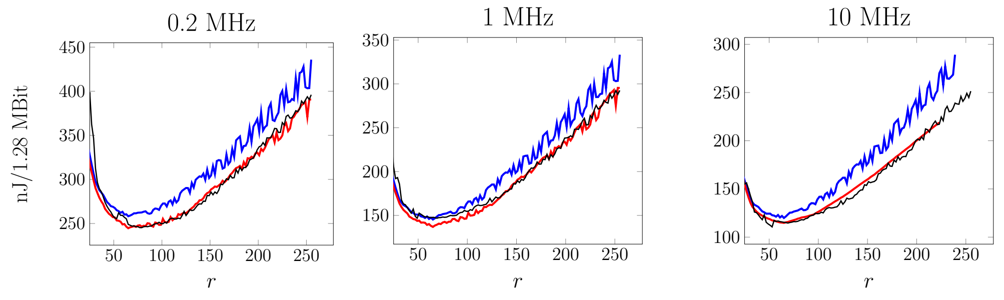
(a) NanGate 15 nm



(b) NanGate 45 nm

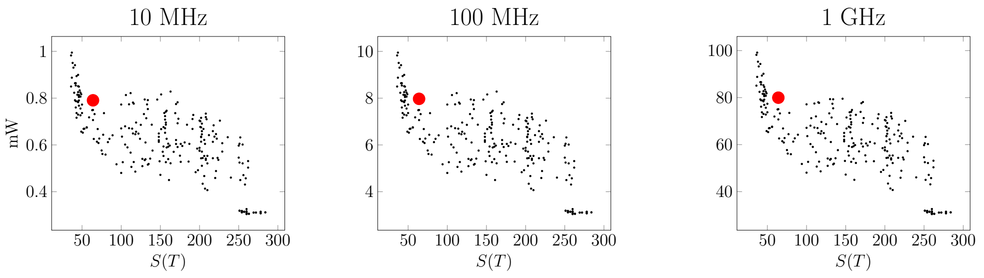


(c) UMC 65 nm

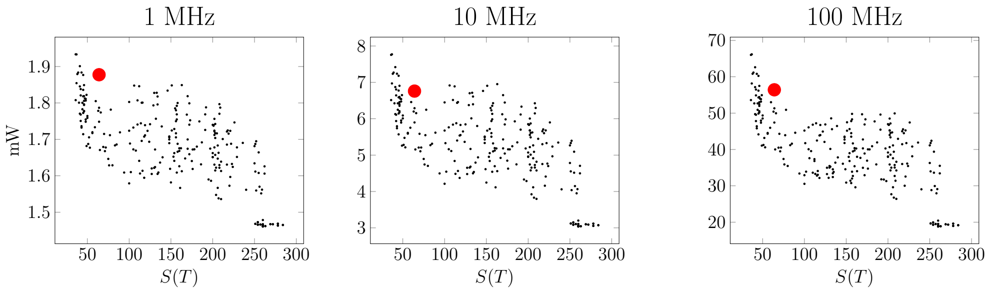


(d) TSMC 90 nm

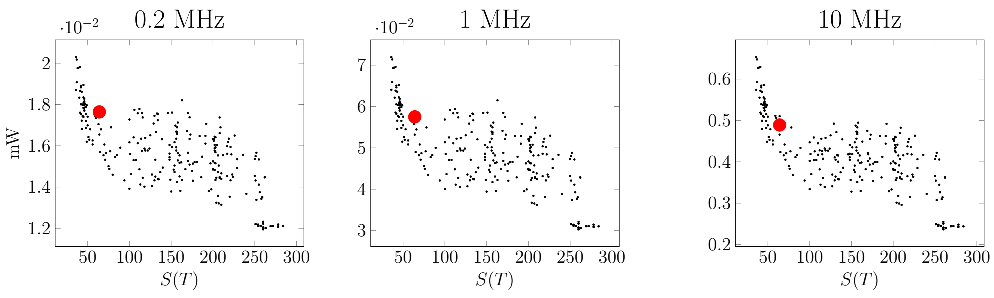
Figure 19: Energy consumption Grain-128.



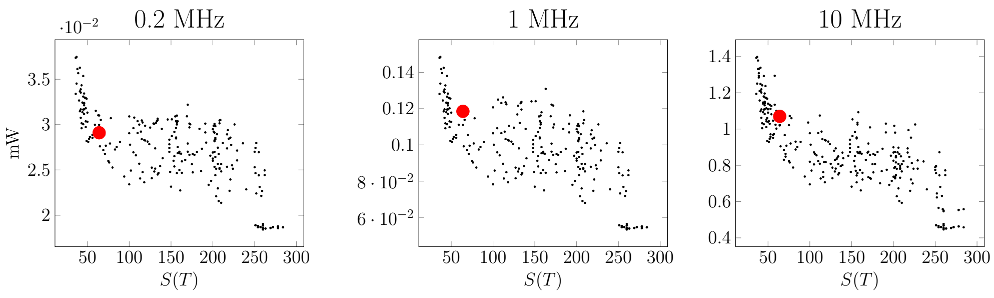
(a) NanGate 15 nm



(b) NanGate 45 nm



(c) UMC 65 nm



(d) TSMC 90 nm

Figure 20: Power trees Grain-128.