# Preliminary Hardware Benchmarking of Eight Round 2 NIST Lightweight AEAD Candidates

Mustafa Khairallah, Thomas Peyrin and Anupam Chattopadhyay

Nanyang Technological University, Singapore

**Abstract.** In this report, we analyze the hardware implementations if 8 candidates for Round 2 of the NIST lightweight cryptography standardization process. These candidates are DryGascon, Elephant, PHOTON-Beetle, Pyjamask, Romulus, Subterranean, TinyJambu and Xoodyak. Specifically, we study the implementations of these algorithms when synthesized using the TSMC 65nm technology and Synopsys Design Compiler, targeting different trade-offs and different use-cases. We show how different candidates stack-up against such trade-offs.

**Keywords:** ASIC · authenticated encryption · hash functions · AEAD · lightweight cryptography · NIST · beanchmarking

## Contents



## 1 Introduction

Lightweight Symmetric Key Cryptography has been a growing research area in the past 10 years or more, with applications varying from block cipher design to authenticated encryption or hash functions and much more. This has lead the National Institute of Standards and Technology (NIST) to release a call for proposals for a new lightweight cryptography standard [NIS18], to be used in applications such as Internet-of-Things (IoT) and Sensor Networks. Authenticated Encryption with Associated Data (AEAD) is one of the most important requirements of symmetric key cryptography (SKE) in these environments, since it is usually cheaper than having independent solutions for the authentication and encryption requirements of the system. The NIST received 57 submissions. 56 submissions were accepted into round 1 of the process, and after an period of analysis 32 designs were selected for round 2. Round 2 is expected to last till December, 2020. Hence, we have decided to study the ASIC performance of different Round 2 candidates.

**Why ASIC?** We have chosen to study ASIC implementations for two reasons:

1. ASIC is an important technology in practice, as many real world products rely on ASIC accelerators for improving the performance of cryptographic algorithm. This is evident by the wide adoption of ASIC accelerators of the Advanced Encryption Standard (AES) and standard hash functions such as SHA-2 and SHA-3. However, due to either expensive tools, lack of expertise, or simplicity of other technologies, *e.g.*, FPGA or micro-controllers, ASIC benchmarking and estimations are sometimes overlooked. During the CAESAR competition [CAE20], ASIC benchmarking was not thoroughly studied except at the late stages of the competition [KHYKC17]. Having early ASIC benchmarks will help the designers improve the performance of their algorithms and give a better picture about the performance of the candidates.

2. Several benchmarking projects have been launched targeting micro-controllers [SR20], general-purpose processors [BL20] or FPGA [MHN+20].

**The LWC Hardware API** After a period of public discussions, Kaps *et.al.* proposed the Hardware API for Lightweight Cryptography, commonly known as the LWC Hardware API [KDT+19], as specification of the compliance criteria, bus interface and communication protocol expected from the implementations submitted for hardware benchmarking of lightweight cryptography. The purpose of the API is to ensure uniformity of the implementations submitted, in terms of communications and a certain level of functionality. Only implementations complaint with this API will be considered in our benchmarking efforts.

**Considered Candidates** On May 7, 2020, we announced our intention to perform a study on ASIC benchmarking of the NIST lightweight candidates on the NIST lightweight cryptography forum. Since then, we have received 22 implementations of 8 candidates from 8 different designers. All the 22 implementations are compliant with the LWC hardware API. The candidates considered are DryGascon, Elephant, PHOTON-Beetle, Pyjamask, Romulus, Subterranean, TinyJambu and Xoodyak. Four submissions (12 implementations) are submitted by the design teams of these candidates, namely DryGascon, Romulus, Subterranean and Xoodyak (Silvia Mella). Five submissions are submitted by Kris Gaj from the GMU CERG team, namely Elephant, PHOTON-Beetle, Pyjamask, TinyJambu and Xoodyak (Richard Haeussler). A summary of these implementations is given in Table 1. All the implementations target only the primary AEAD variant of each candidate.

**Table 1:** Candidates and implementations considered in this report.

| Candidate | Architecture | Identifier | Designer | Language |
|---|---|---|---|---|
| DryGascon | Basic iterative: 1-round | drygascon-eh | Ekawat Homsirikamol | VHDL |
| Elephant | Basic iterative: 1-round<br>Basic iterative: 5-round | elephant-rh-1<br>elephant-rh-5 | Richard Haeussler | VHDL |
| PHOTON-Beetle | Basic iterative: 1-round | beetle-vl | Vivian Ledynh | VHDL |
| Pyjamask | Folded<br>Pipelined | pyjamask-rn-f<br>pyjamask-rn-p | Rishub Nagpal | VHDL |
| Romulus | Basic iterative: 1-round<br>Basic iterative: 2-round<br>Basic iterative: 4-round<br>Basic iterative: 8-round | romulus-mk-1<br>romulus-mk-2<br>romulus-mk-4<br>romulus-mk-8 | Mustafa Khairallah | Verilog-VHDL |
| Subterranean | Basic iterative | subterranean-pm | Pedro Maat Costa Massolino | Verilog |
| TinyJambu | Serial: 32-bit<br>Serial: 16-bit<br>Serial: 1-bit | tinyjambu-sl-32<br>tinyjambu-sl-16<br>tinyjambu-sl-1 | Sammy Lin | VHDL |
| Xoodyak | Basic Iterative: 1-round<br>Basic Iterative: 2-round<br>Basic Iterative: 3-round<br>Basic Iterative: 4-round<br>Basic Iterative: 6-round<br>Basic Iterative: 12-round | xoodyak-sm-1<br>xoodyak-sm-2<br>xoodyak-sm-3<br>xoodyak-sm-4<br>xoodyak-sm-6<br>xoodyak-sm-12 | Silvia Mella | VHDL |
| | Basic Iterative: 1-round<br>Serial: 128-bit | xoodyak-rh-1<br>xoodyak-rh-s | Richard Haeussler | |

**Use Cases** In this report, we consider two use-cases for our initial study:

1. Performance Efficiency: We try to optimize the designs towards the best throughput/area ratio, by varying architectures, area and speed synthesis constraints. Once the optimization is done, extract different measurements for each architecture: area, power, throughput and energy.
2. Lightweight Protocols: We optimize the designs towards practical lightweight protocols. We choose two representative target protocols: Bluetooth and Bluetooth Low Energy (BLE). The application data rate of these protocols ranges between 0.27 and 2.1 Mbps, with an air data rate between 125 Kbps and 3 Mbps. Hence, we synthesize the designs for a target throughput of 3 Mbps for very long messages, and measure the corresponding area, power and energy.

In [BCL18], Burg *et.al.* provided a survey of the security needs of different wireless communication standard. They show that most relevant wireless communication protocols, with the exceptions of 802.11 variants, have data rates below 20 Mbps. The SigFox standard has a data rate of 100 bps and most of the standards have data rates in the Kbps range. However, our study, as shall shows that the power consumption and area of the circuit does not change significantly when the throughput is below the Mbps range. Hence, in order to simplify reading our reports, we consider the Bluetooth/BLE case with a target throughput of 3 Mbps, assuming the area and power consumption is almost constant below such rate and the energy varies linearly with the data rate. This is due to the fact that at such frequencies, the power consumption is dominated by the static power that does not depend on the frequency. The same is not true for high data rates as they can affect the area and power consumption significantly and non-linearly, as the switching power depends on the target frequency, while the synthesizer may require larger, more power consuming standard cells to achieve such high data rates. For 802.11 and other applications that require high data rates, we introduce the first use case.

**Process and Flow** For this study, we used and area-oriented and throughput-oriented synthesis flow, given in Figure 1. We used the Synopsys VCS K-2015.09SP2-10 and Xilinx ISIM 14.7 simulators, and the Synopsys Design Compiler Q-2019-12-SP5. We used the general-purpose industry grade TSMC TSBN 65nm 9-track standard cell library as a target. We used Python for generating and analyzing the results. The simulation is used to generate the data useful to the analyze the synthesis outputs, namely throughput and energy.

**Data Size** In accordance with the FPGA benchmarking project by the CERG team, we consider 9 different data sizes:

1. 16 bytes of associated data.
2. 64 bytes of associated data.
3. 1536 bytes of associated data.
4. 16 bytes of plaintext.
5. 64 bytes of plaintext.
6. 1536 bytes of plaintext.
7. 16 bytes of both associated data and plaintext.
8. 64 bytes of both associated data and plaintext.

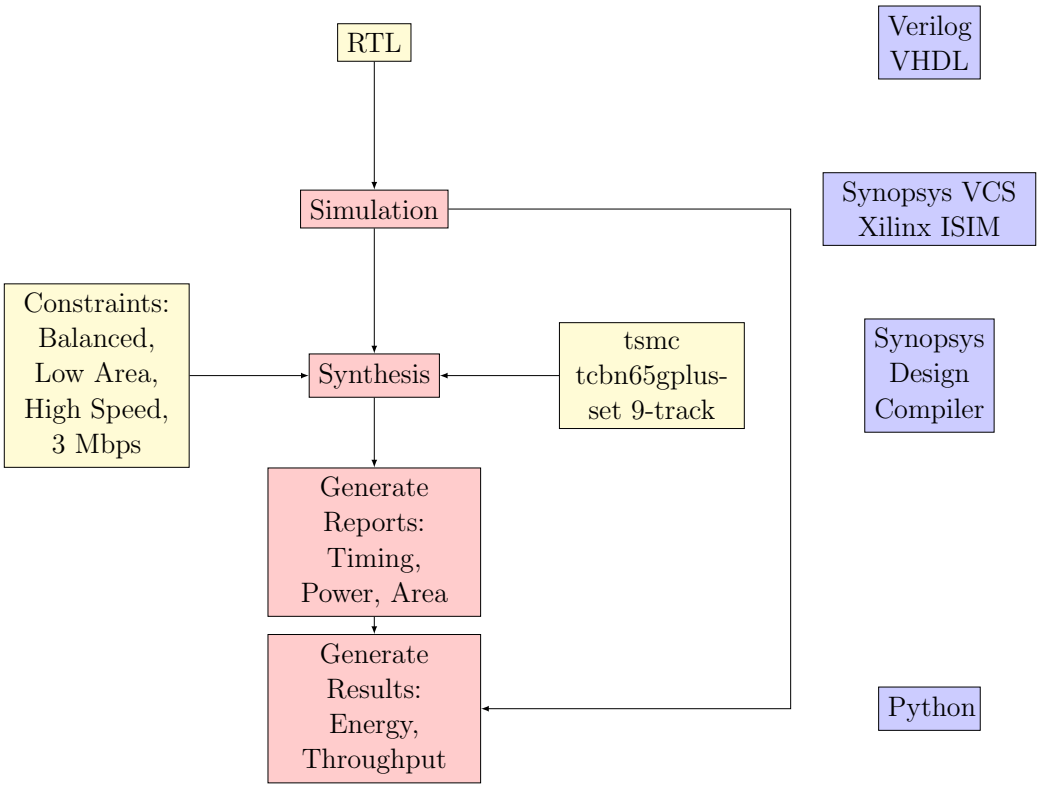


**Figure 1:** The synthesis flow and tools used for our Study.

9. 1536 bytes of both associated data and plaintext.

Given these data sizes, we cover both short messages and relatively long messages. We can also assess the cost of authentication vs. encryption.

## 2 Raw Synthesis Results

In this section, we report the raw synthesis results for the implementations considered. Each implementation is synthesized against four different corner cases: balanced (BC), low-area (LA), high-speed (HS) and low frequency (LF) targetting ∼3 Mbps throughput. The results reported are area in both $\mu m^2$ and gate equivalents (GE), clock period in ns and power in mW. The results are shown in Tables 2 and 3. The xoodyak-rh-s implementation of Xoodyak presents a anomaly in the results as it is both slow and large in terms of area. We beleive this is due to the use of large register files that maybe cheap in FPGAs but not suitable for ASIC. In the spirit of fairness, we include its results in this section, but we do not include it in subsequent analytic sections.

**Table 2:** Raw synthesis results using the TSMC 65nm standard cell library.

| Implementation | Corner | Area ($\mu m^2$) | Area (GE) | Clock Period (ns) | Power (mW) |
|---|---|---|---|---|---|
| drygascon-eh | BC | 24172.6 | 17267 | 0.51 | 1.6033 |
| drygascon-eh | LA | 23934.6 | 17097 | 0.48 | 1.5934 |
| drygascon-eh | HS | 23934.6 | 17097 | 0.48 | 1.5934 |
| drygascon-eh | LF | 24161.0 | 17258 | 2031.75 | 0.5113 |
| elephant-rh-1 | BC | 17437.3 | 12456 | 0.67 | 1.1372 |
| elephant-rh-1 | LA | 17437.3 | 12456 | 0.67 | 1.1372 |
| elephant-rh-1 | HS | 18075.6 | 12912 | 0.50 | 23.3220 |
| elephant-rh-1 | LF | 17436.6 | 12455 | 606.06 | 0.3503 |
| elephant-rh-5 | BC | 23294.2 | 16639 | 0.72 | 1.2216 |
| elephant-rh-5 | LA | 23294.2 | 16639 | 0.72 | 1.2216 |
| elephant-rh-5 | HS | 30395.5 | 21712 | 0.57 | 24.6182 |
| elephant-rh-5 | LF | 23282.6 | 16631 | 2222.22 | 0.4450 |
| beetle-vl | BC | 20175.1 | 14411 | 0.60 | 1.1561 |
| beetle-vl | LA | 20175.1 | 14411 | 0.60 | 1.1561 |
| beetle-vl | HS | 26795.2 | 19140 | 0.50 | 22.2438 |
| beetle-vl | LF | 20171.2 | 14408 | 1292.93 | 0.3974 |
| pyjamask-rn-f | BC | 74189.9 | 52993 | 0.68 | 6.4000 |
| pyjamask-rn-f | LA | 74189.9 | 52993 | 0.68 | 6.4000 |
| pyjamask-rn-f | HS | 73295.6 | 52354 | 0.50 | 136.4332 |
| pyjamask-rn-f | LF | 74083.7 | 52917 | 162.85 | 1.9280 |
| pyjamask-rn-p | BC | 74507.4 | 53220 | 0.62 | 5.9839 |
| pyjamask-rn-p | LA | 74507.4 | 53220 | 0.62 | 5.9839 |
| pyjamask-rn-p | HS | 81637.2 | 58313 | 0.50 | 139.7180 |
| pyjamask-rn-p | LF | 73825.2 | 52733 | 418.30 | 1.6614 |
| romulus-mk-1 | BC | 8672.8 | 6195 | 0.86 | 0.3675 |
| romulus-mk-1 | LA | 8672.8 | 6195 | 0.86 | 0.3675 |
| romulus-mk-1 | HS | 9079.6 | 6486 | 0.50 | 4.1685 |
| romulus-mk-1 | LF | 8664.8 | 6190 | 711.11 | 0.1800 |
| romulus-mk-2 | BC | 10070.3 | 7194 | 0.97 | 0.4032 |
| romulus-mk-2 | LA | 10070.3 | 7194 | 0.97 | 0.4032 |
| romulus-mk-2 | HS | 11665.1 | 8333 | 0.50 | 4.3832 |
| romulus-mk-2 | LF | 10060.2 | 7186 | 1333.33 | 0.2110 |
| romulus-mk-4 | BC | 13243.7 | 9460 | 0.80 | 0.4856 |
| romulus-mk-4 | LA | 13243.7 | 9460 | 0.80 | 0.4856 |
| romulus-mk-4 | HS | 20629.1 | 14736 | 0.78 | 5.2348 |
| romulus-mk-4 | LF | 13236.5 | 9455 | 2370.37 | 0.2874 |
| romulus-mk-8 | BC | 19906.2 | 14219 | 0.88 | 0.6612 |
| romulus-mk-8 | LA | 19905.1 | 14218 | 0.88 | 0.6612 |
| romulus-mk-8 | HS | 19905.1 | 14218 | 0.88 | 0.6612 |
| romulus-mk-8 | LF | 19858.3 | 14185 | 3878.78 | 0.4494 |

**Table 3:** Raw synthesis results using the TSMC 65nm standard cell library (Continued).

| Implementation | Corner | Area ($\mu m^2$) | Area (GE) | Clock Period (ns) | Power (mW) |
|---|---|---|---|---|---|
| subterranean-pm | BC | 9869.8 | 7050 | 0.47 | 0.5522 |
| subterranean-pm | LA | 9869.8 | 7050 | 0.47 | 0.5522 |
| subterranean-pm | HS | 9869.8 | 7050 | 0.47 | 0.5522 |
| subterranean-pm | LF | 9869.8 | 7050 | 10666.67 | 0.2120 |
| tinyjambu-sl-32 | BC | 6201.4 | 4430 | 0.60 | 0.4327 |
| tinyjambu-sl-32 | LA | 6201.4 | 4430 | 0.60 | 0.4327 |
| tinyjambu-sl-32 | HS | 6399.7 | 4572 | 0.50 | 10.3589 |
| tinyjambu-sl-32 | LF | 6203.5 | 4432 | 313.7254902 | 0.1362 |
| tinyjambu-sl-16 | BC | 6396.8 | 4570 | 0.60 | 0.4351 |
| tinyjambu-sl-16 | LA | 6396.8 | 4570 | 0.60 | 0.4351 |
| tinyjambu-sl-16 | HS | 6567.5 | 4692 | 0.50 | 10.2917 |
| tinyjambu-sl-16 | LF | 6398.6 | 4571 | 161.6161616 | 0.1558 |
| tinyjambu-sl-1 | BC | 6201.4 | 4430 | 0.58 | 0.4326 |
| tinyjambu-sl-1 | LA | 6201.4 | 4430 | 0.58 | 0.4326 |
| tinyjambu-sl-1 | HS | 6399.7 | 4572 | 0.50 | 10.3589 |
| tinyjambu-sl-1 | LF | 6201.4 | 4430 | 41.34366925 | 0.2434 |
| xoodyak-sm-1 | BC | 13469.0 | 9621 | 0.91 | 0.8403 |
| xoodyak-sm-1 | LA | 13472.3 | 9624 | 0.91 | 0.8403 |
| xoodyak-sm-1 | HS | 13719.6 | 9800 | 0.50 | 17.6858 |
| xoodyak-sm-1 | LF | 13462.6 | 9617 | 3047.62 | 0.2432 |
| xoodyak-sm-2 | BC | 19721.9 | 14088 | 0.91 | 0.9897 |
| xoodyak-sm-2 | LA | 19723.3 | 14089 | 0.92 | 0.9902 |
| xoodyak-sm-2 | HS | 25056.7 | 17898 | 0.50 | 18.5889 |
| xoodyak-sm-2 | LF | 19716.8 | 14084 | 4266.67 | 0.3907 |
| xoodyak-sm-3 | BC | 26852.0 | 19180 | 0.93 | 1.1667 |
| xoodyak-sm-3 | LA | 26851.0 | 19180 | 0.93 | 1.1667 |
| xoodyak-sm-3 | HS | 46927.1 | 33520 | 0.62 | 20.2193 |
| xoodyak-sm-3 | LF | 26837.6 | 19170 | 4923.08 | 0.5651 |
| xoodyak-sm-4 | BC | 31643.6 | 22603 | 0.93 | 1.2811 |
| xoodyak-sm-4 | LA | 31643.6 | 22603 | 0.93 | 1.2811 |
| xoodyak-sm-4 | HS | 57271.3 | 40909 | 0.80 | 15.7559 |
| xoodyak-sm-4 | LF | 31631.4 | 22594 | 5333.33 | 0.6767 |
| xoodyak-sm-6 | BC | 41436.7 | 29598 | 0.93 | 1.5168 |
| xoodyak-sm-6 | LA | 41436.7 | 29598 | 0.93 | 1.5168 |
| xoodyak-sm-6 | HS | 41436.7 | 29598 | 0.93 | 1.5168 |
| xoodyak-sm-6 | LF | 41430.6 | 29594 | 5818.19 | 0.9122 |
| xoodyak-sm-12 | BC | 70597.1 | 50427 | 0.93 | 2.1340 |
| xoodyak-sm-12 | LA | 70597.1 | 50427 | 0.93 | 2.1340 |
| xoodyak-sm-12 | HS | 70597.1 | 50427 | 0.93 | 2.1340 |
| xoodyak-sm-12 | LF | 70610.8 | 50437 | 6400.00 | 1.5960 |
| xoodyak-rh-1 | BC | 14304.2 | 10218 | 0.78 | 0.6633 |
| xoodyak-rh-1 | LA | 14304.2 | 10218 | 0.78 | 0.6633 |
| xoodyak-rh-1 | HS | 15361.6 | 10973 | 0.50 | 13.9210 |
| xoodyak-rh-1 | LF | 14317.2 | 10227 | 3368.42 | 0.2710 |
| xoodyak-rh-s | BC | 34322.8 | 24517 | 1.24 | 2.6524 |
| xoodyak-rh-s | LA | 34322.8 | 24517 | 1.24 | 2.6524 |
| xoodyak-rh-s | HS | 34322.8 | 24517 | 1.24 | 2.6524 |
| xoodyak-rh-s | LF | 34307.6 | 24506 | 245.21 | 0.7560 |

# 3 Maximum Throughput

Given the different corner cases with the aid of simulation outputs, we get the throughput for different message sizes. Tables 4, 5 and 6 include the maximum achievable throughput for 16 bytes, 64 bytes and 1536 bytes, respectively. |A| and |M| refers to the size of associated data and plaintext in bytes, respectively. AD efficiency refers to the gain in throughput when encrypting $X$ bytes of plaintext and $X$ bytes of associated data vs. only $X$ bytes of plaintext, with 1.00 being the highest.

**Table 4:** Maximum throughput for 16-byte messages.

| Implementation | Throughput (Mbps) | | | AD Efficiency |
|---|---|---|---|---|
| | \|A\| = 16 bytes | \|M\| = 16 bytes | \|A\| = \|M\| = 16 bytes | |
| drygascon-eh | 3707.7 | 3707.7 | 5734.8 | 0.55 |
| elephant-rh-1 | 583.1 | 729.3 | 1166.3 | 0.60 |
| elephant-rh-5 | 1887.1 | 2363.8 | 3774.1 | 0.60 |
| beetle-vl | 3324.6 | 3200.0 | 4654.5 | 0.45 |
| pyjamask-rn-f | 204.0 | 204.6 | 337.3 | 0.65 |
| pyjamask-rn-p | 436.1 | 439.1 | 742.0 | 0.69 |
| romulus-mk-1 | 1828.6 | 1828.8 | 3657.1 | 1.00 |
| romulus-mk-2 | 3047.6 | 3047.6 | 6095.2 | 1.00 |
| romulus-mk-4 | 2930.4 | 2930.4 | 5860.8 | 1.00 |
| romulus-mk-8 | 3463.2 | 3463.2 | 6926.4 | 1.00 |
| subterranean-pm | 5340.0 | 5237.3 | 9903.3 | 0.89 |
| tinyjambu-sl-32 | 1361.7 | 951.7 | 1580.2 | 0.66 |
| tinyjambu-sl-16 | 735.6 | 502.9 | 836.6 | 0.66 |
| tinyjambu-sl-1 | 49.7 | 33.2 | 55.3 | 0.67 |
| xoodyak-sm-1 | 3200.0 | 3160.5 | 6095.2 | 0.93 |
| xoodyak-rh-1 | 3506.7 | 3506.7 | 6736.8 | 0.92 |
| xoodyak-sm-2 | 4571.4 | 4491.2 | 8533.3 | 0.87 |
| xoodyak-sm-3 | 4301.0 | 4213.3 | 7940.4 | 0.88 |
| xoodyak-sm-4 | 3636.4 | 3555.5 | 6666.7 | 0.88 |
| xoodyak-sm-6 | 3440.9 | 3357.0 | 6256.1 | 0.86 |
| xoodyak-sm-12 | 3823.2 | 3719.8 | 6881.7 | 0.85 |

**Table 5:** Maximum throughput for 64-byte messages.

| Implementation | Throughput (Mbps) | | | AD Efficiency |
|---|---|---|---|---|
| | \|A\| = 64 bytes | \|M\| = 64 bytes | \|A\| = \|M\| = 64 bytes | |
| drygascon-eh | 7901.2 | 7901.2 | 9168.2 | 0.16 |
| elephant-rh-1 | 1665.0 | 1185.2 | 1815.7 | 0.53 |
| elephant-rh-5 | 5378.7 | 4010.0 | 6069.2 | 0.52 |
| beetle-vl | 6360.2 | 5720.7 | 6989.7 | 0.22 |
| pyjamask-rn-f | 504.6 | 503.7 | 665.4 | 0.32 |
| pyjamask-rn-p | 1162.3 | 1151.9 | 1587.6 | 0.34 |
| romulus-mk-1 | 3820.9 | 3200.0 | 4571.4 | 0.43 |
| romulus-mk-2 | 6564.1 | 5688.9 | 8127.0 | 0.43 |
| romulus-mk-4 | 6564.1 | 5967.4 | 8524.8 | 0.43 |
| romulus-mk-8 | 8080.8 | 7757.6 | 11082.3 | 0.43 |
| subterranean-pm | 17291.5 | 17021.2 | 27578.8 | 0.62 |
| tinyjambu-sl-32 | 2876.4 | 1512.5 | 2275.6 | 0.50 |
| tinyjambu-sl-16 | 1551.5 | 787.1 | 1193.5 | 0.52 |
| tinyjambu-sl-1 | 104.7 | 51.1 | 78.2 | 0.53 |
| xoodyak-sm-1 | 9570.0 | 8325.2 | 13385.6 | 0.61 |
| xoodyak-rh-1 | 10449.0 | 9225.2 | 14733.8 | 0.60 |
| xoodyak-sm-2 | 13298.7 | 11770.1 | 18450.5 | 0.57 |
| xoodyak-sm-3 | 12325.5 | 11010.8 | 17026.9 | 0.55 |
| xoodyak-sm-4 | 10322.6 | 9275.4 | 14222.2 | 0.53 |
| xoodyak-sm-6 | 9658.6 | 8738.7 | 13266.0 | 0.52 |
| xoodyak-sm-12 | 10587.3 | 9658.9 | 14487.8 | 0.50 |

**Table 6:** Maximum throughput for 1536-byte messages.

| Implementation | Throughput (Mbps) | | | AD Efficiency |
|---|---|---|---|---|
| | \|A\| = 1536 bytes | \|M\| = 1536 bytes | \|A\| = \|M\| = 1536 bytes | |
| drygascon-eh | 12385.1 | 12385.1 | 12539.8 | 0.01 |
| elephant-rh-1 | 3448.3 | 1841.3 | 2442.6 | 0.32 |
| elephant-rh-5 | 11095.2 | 6410.3 | 8274.0 | 0.29 |
| beetle-vl | 8979.2 | 7644.2 | 8323.8 | 0.09 |
| pyjamask-rn-f | 953.9 | 940.1 | 965.3 | 0.03 |
| pyjamask-rn-p | 2483.1 | 2392.2 | 2496.3 | 0.04 |
| romulus-mk-1 | 7651.3 | 4208.2 | 5515.2 | 0.31 |
| romulus-mk-2 | 13563.0 | 7866.8 | 10130.3 | 0.29 |
| romulus-mk-4 | 14167.1 | 8920.6 | 11165.0 | 0.25 |
| romulus-mk-8 | 18325.0 | 12846.0 | 15455.0 | 0.20 |
| subterranean-pm | 60660.5 | 60520.1 | 64158.7 | 0.06 |
| tinyjambu-sl-32 | 4461.9 | 1863.4 | 2647.7 | 0.42 |
| tinyjambu-sl-16 | 2402.8 | 960.4 | 1381.8 | 0.44 |
| tinyjambu-sl-1 | 161.9 | 61.8 | 90.1 | 0.46 |
| xoodyak-sm-1 | 25336.1 | 17479.4 | 21379.8 | 0.22 |
| xoodyak-rh-1 | 27459.2 | 19320.8 | 23439.2 | 0.21 |
| xoodyak-sm-2 | 33121.3 | 24478.1 | 29032.5 | 0.19 |
| xoodyak-sm-3 | 29758.8 | 22780.9 | 26585.3 | 0.17 |
| xoodyak-sm-4 | 24458.6 | 19128.3 | 22100.7 | 0.16 |
| xoodyak-sm-6 | 22394.8 | 17952.3 | 20501.0 | 0.14 |
| xoodyak-sm-12 | 23936.4 | 19750.2 | 22243.9 | 0.13 |

# 4 Minimum Energy

Tables 7, 8 and 7 give the estimated energy consumption needed in order to process messages of 16 bytes, 64 bytes and 1536 bytes, respectively, using the outputs of simulations and synthesis . |A| and |M| refers to the size of associated data and plaintext in bytes, respectively. AD cost refers to the energy overhead when encrypting $X$ bytes of plaintext and $X$ bytes of associated data vs. only $X$ bytes of plaintext, with 0.00 being the lowest.

**Table 7:** Minimum energy for 16-byte messages.

| Implementation | Energy (pJ) | | | AD Cost |
|---|---|---|---|---|
| | \|A\| = 16 bytes | \|M\| = 16 bytes | \|A\| = \|M\| = 16 bytes | |
| drygascon-eh | 55.1 | 55.1 | 71.3 | 0.29 |
| elephant-rh-1 | 334.5 | 267.4 | 334.5 | 0.25 |
| elephant-rh-5 | 104.6 | 83.6 | 104.6 | 0.25 |
| beetle-vl | 53.4 | 55.5 | 76.3 | 0.38 |
| pyjamask-rn-f | 5461.8 | 5444.4 | 6606.3 | 0.21 |
| pyjamask-rn-p | 2177.8 | 2162.9 | 2559.9 | 0.18 |
| romulus-mk-1 | 44.2 | 44.2 | 44.2 | 0.00 |
| romulus-mk-2 | 32.9 | 32.9 | 32.9 | 0.00 |
| romulus-mk-4 | 21.8 | 21.8 | 21.8 | 0.00 |
| romulus-mk-8 | 24.4 | 24.4 | 24.4 | 0.00 |
| subterranean-pm | 13.2 | 13.5 | 14.3 | 0.06 |
| tinyjambu-sl-32 | 48.8 | 69.8 | 84.1 | 0.20 |
| tinyjambu-sl-16 | 90.8 | 132.9 | 159.8 | 0.20 |
| tinyjambu-sl-1 | 1291.7 | 1934.2 | 2321.4 | 0.20 |
| xoodyak-sm-1 | 61.2 | 61.9 | 64.2 | 0.04 |
| xoodyak-rh-1 | 37.8 | 37.8 | 39.3 | 0.04 |
| xoodyak-sm-2 | 50.4 | 51.3 | 54.0 | 0.05 |
| xoodyak-sm-3 | 52.1 | 53.2 | 56.4 | 0.06 |
| xoodyak-sm-4 | 52.4 | 53.6 | 57.2 | 0.07 |
| xoodyak-sm-6 | 56.4 | 57.8 | 62.1 | 0.07 |
| xoodyak-sm-12 | 71.4 | 73.4 | 79.4 | 0.08 |

**Table 8:** Minimum energy for 64-byte messages.

| Implementation | Energy (pJ) | | | AD Cost |
|---|---|---|---|---|
| | \|A\| = 64 bytes | \|M\| = 64 bytes | \|A\| = \|M\| = 64 bytes | |
| drygascon-eh | 103.3 | 103.3 | 167.5 | 0.62 |
| elephant-rh-1 | 468.6 | 658.3 | 859.5 | 0.31 |
| elephant-rh-5 | 146.9 | 197.0 | 260.3 | 0.32 |
| beetle-vl | 111.7 | 124.1 | 203.2 | 0.64 |
| pyjamask-rn-f | 8830.2 | 8865.0 | 13395.5 | 0.51 |
| pyjamask-rn-p | 3268.5 | 3298.2 | 4785.9 | 0.45 |
| romulus-mk-1 | 84.7 | 101.1 | 141.6 | 0.40 |
| romulus-mk-2 | 61.0 | 70.4 | 98.6 | 0.40 |
| romulus-mk-4 | 38.8 | 42.7 | 59.8 | 0.40 |
| romulus-mk-8 | 41.9 | 43.6 | 61.1 | 0.40 |
| subterranean-pm | 16.4 | 16.6 | 20.5 | 0.23 |
| tinyjambu-sl-32 | 92.4 | 175.8 | 233.7 | 0.32 |
| tinyjambu-sl-16 | 172.2 | 339.6 | 448.0 | 0.31 |
| tinyjambu-sl-1 | 2453.9 | 5023.4 | 6572.8 | 0.30 |
| xoodyak-sm-1 | 81.8 | 94.1 | 117.0 | 0.24 |
| xoodyak-rh-1 | 50.7 | 57.4 | 71.9 | 0.25 |
| xoodyak-sm-2 | 59.3 | 78.3 | 100.0 | 0.28 |
| xoodyak-sm-3 | 72.7 | 81.4 | 105.2 | 0.29 |
| xoodyak-sm-4 | 73.9 | 82.2 | 107.2 | 0.30 |
| xoodyak-sm-6 | 80.4 | 88.9 | 117.1 | 0.32 |
| xoodyak-sm-12 | 103.2 | 113.1 | 150.8 | 0.33 |

**Table 9:** Minimum energy for 1536-byte messages.

| Implementation | Energy (pJ) | | | AD Cost |
|---|---|---|---|---|
| | \|A\| = 1536 bytes | \|M\| = 1536 bytes | \|A\| = \|M\| = 1536 bytes | |
| drygascon-eh | 1580.9 | 1580.9 | 3122.8 | 0.98 |
| elephant-rh-1 | 5430.2 | 10169.4 | 15332.2 | 0.51 |
| elephant-rh-5 | 1709.0 | 2957.9 | 4583.3 | 0.55 |
| beetle-vl | 1898.5 | 2230.1 | 4096.1 | 0.84 |
| pyjamask-rn-f | 112129.3 | 113765.6 | 221595.1 | 0.95 |
| pyjamask-rn-p | 36718.0 | 38113.0 | 73050.3 | 0.92 |
| romulus-mk-1 | 1015.1 | 1845.7 | 2816.6 | 0.53 |
| romulus-mk-2 | 708.7 | 1221.8 | 1897.6 | 0.55 |
| romulus-mk-4 | 432.0 | 686.1 | 1096.3 | 0.60 |
| romulus-mk-8 | 443.4 | 632.5 | 1051.4 | 0.66 |
| subterranean-pm | 111.9 | 112.1 | 211.5 | 0.89 |
| tinyjambu-sl-32 | 1430.0 | 3424.1 | 4819.9 | 0.41 |
| tinyjambu-sl-16 | 2670.1 | 6680.2 | 9286.4 | 0.39 |
| tinyjambu-sl-1 | 38094.9 | 99758.3 | 136948.6 | 0.37 |
| xoodyak-sm-1 | 741.7 | 1075.1 | 1758.0 | 0.64 |
| xoodyak-rh-1 | 463.0 | 658.1 | 1084.9 | 0.65 |
| xoodyak-sm-2 | 668.3 | 904.2 | 1524.8 | 0.67 |
| xoodyak-sm-3 | 722.6 | 944.0 | 1617.8 | 0.71 |
| xoodyak-sm-4 | 748.2 | 956.7 | 1656.1 | 0.73 |
| xoodyak-sm-6 | 832.3 | 1038.2 | 1818.3 | 0.75 |
| xoodyak-sm-12 | 1095.5 | 1327.7 | 2357.7 | 0.78 |

# 5 Trade-offs

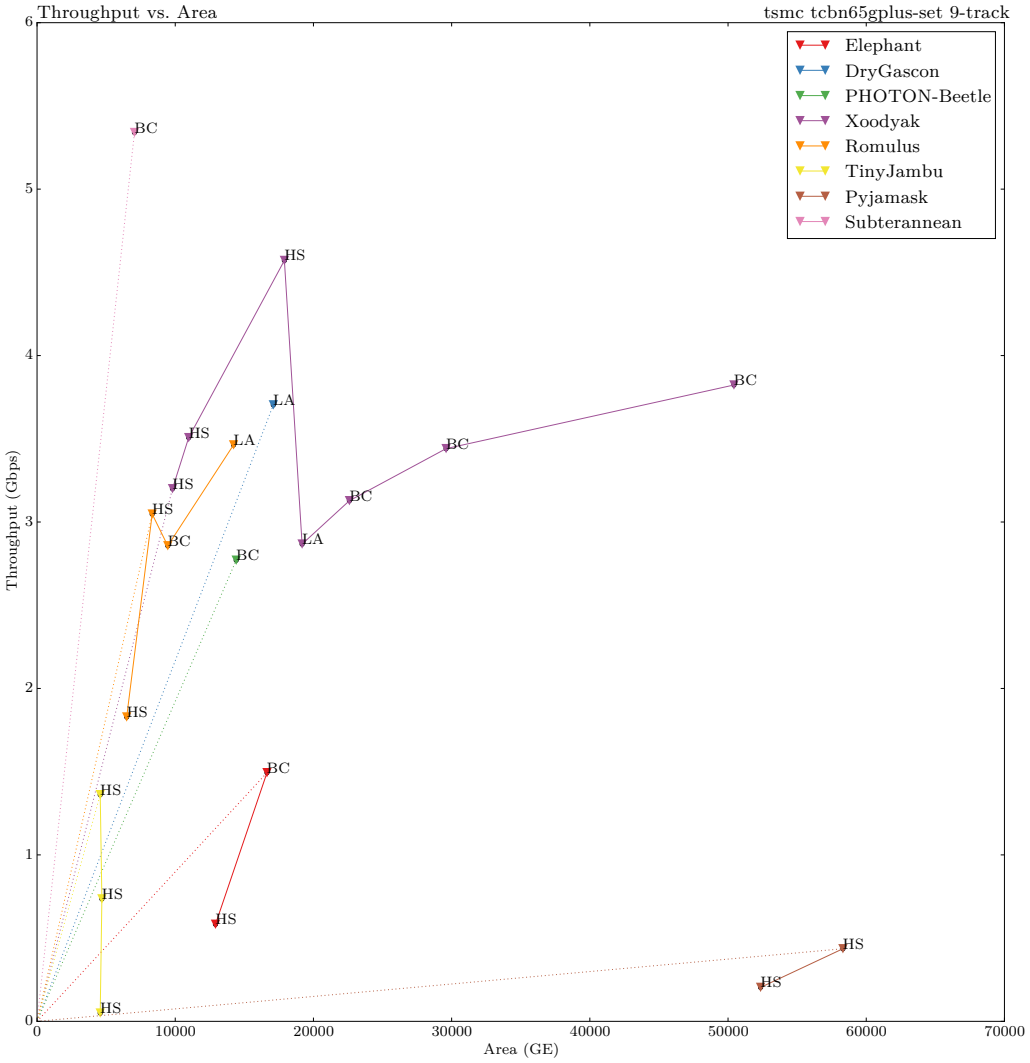


**Figure 2:** Throughput vs. Area for $|A| = 16$ bytes.

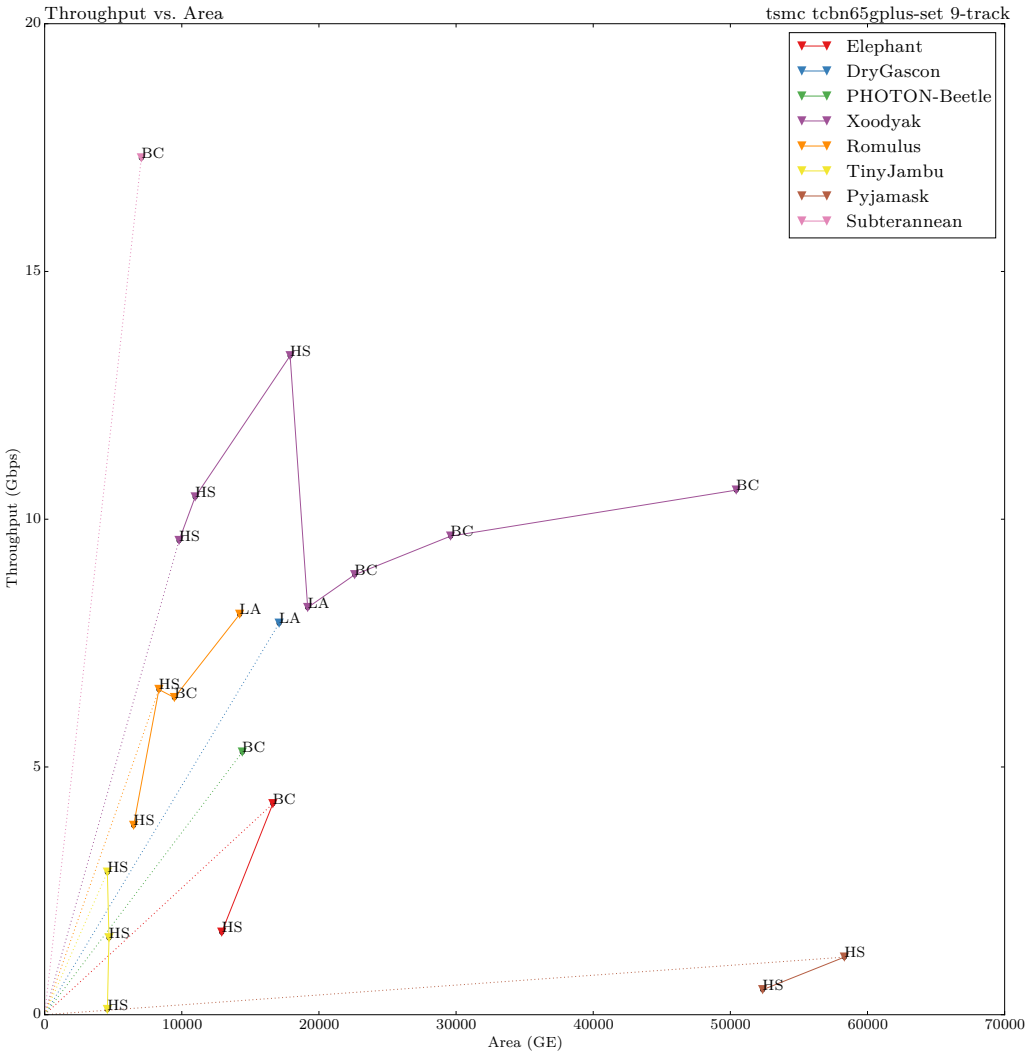


**Figure 3:** Throughput vs. Area for $|A| = 64$ bytes.

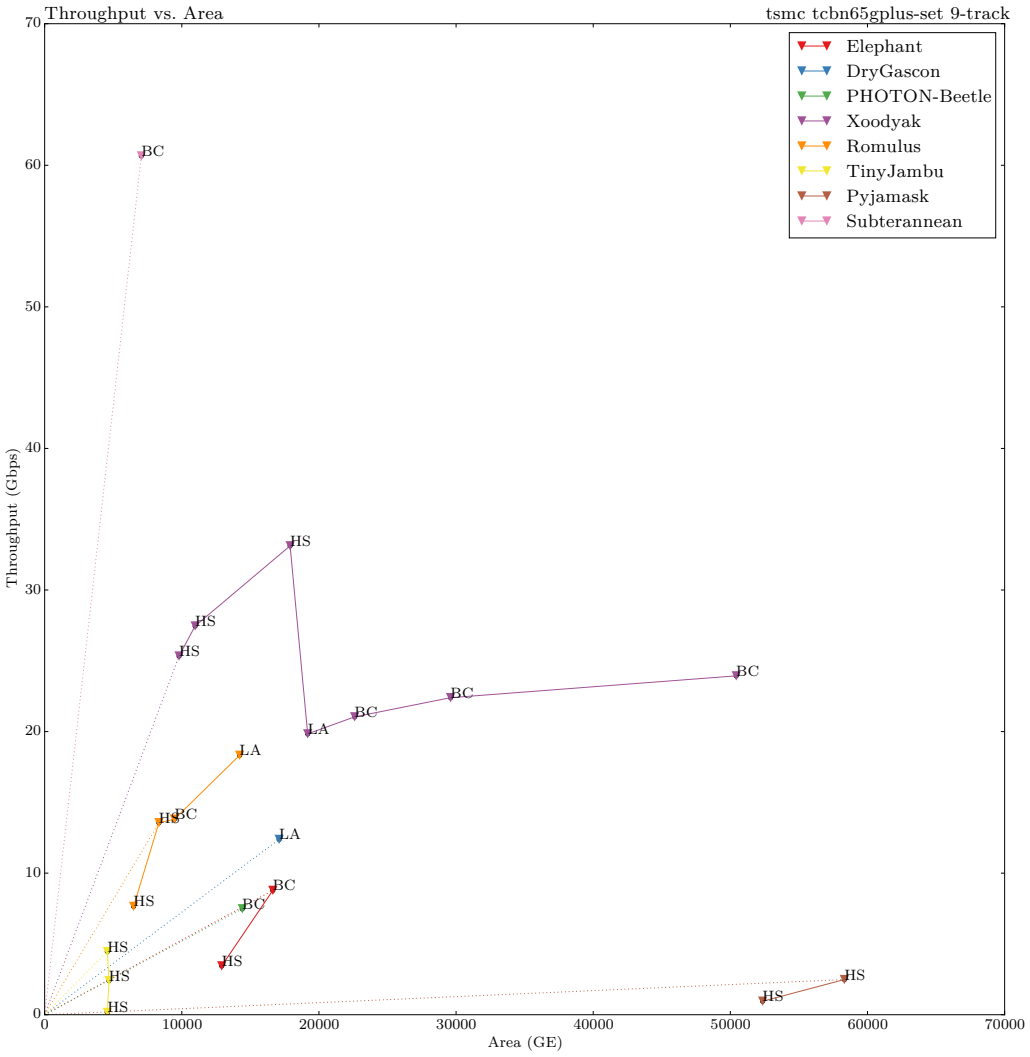


**Figure 4:** Throughput vs. Area for $|A| = 1536$ bytes.

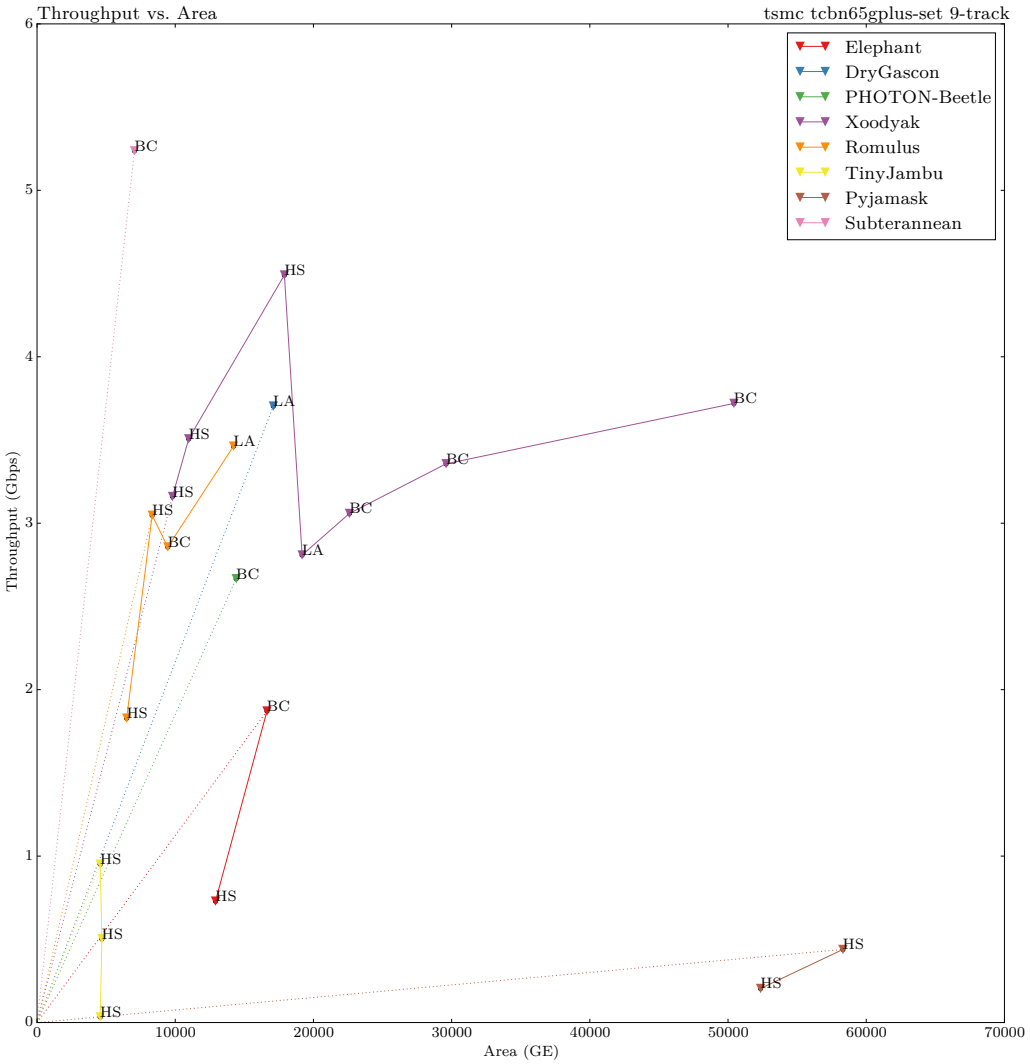


**Figure 5:** Throughput vs. Area for $|\mathrm{M}| = 16$ bytes.

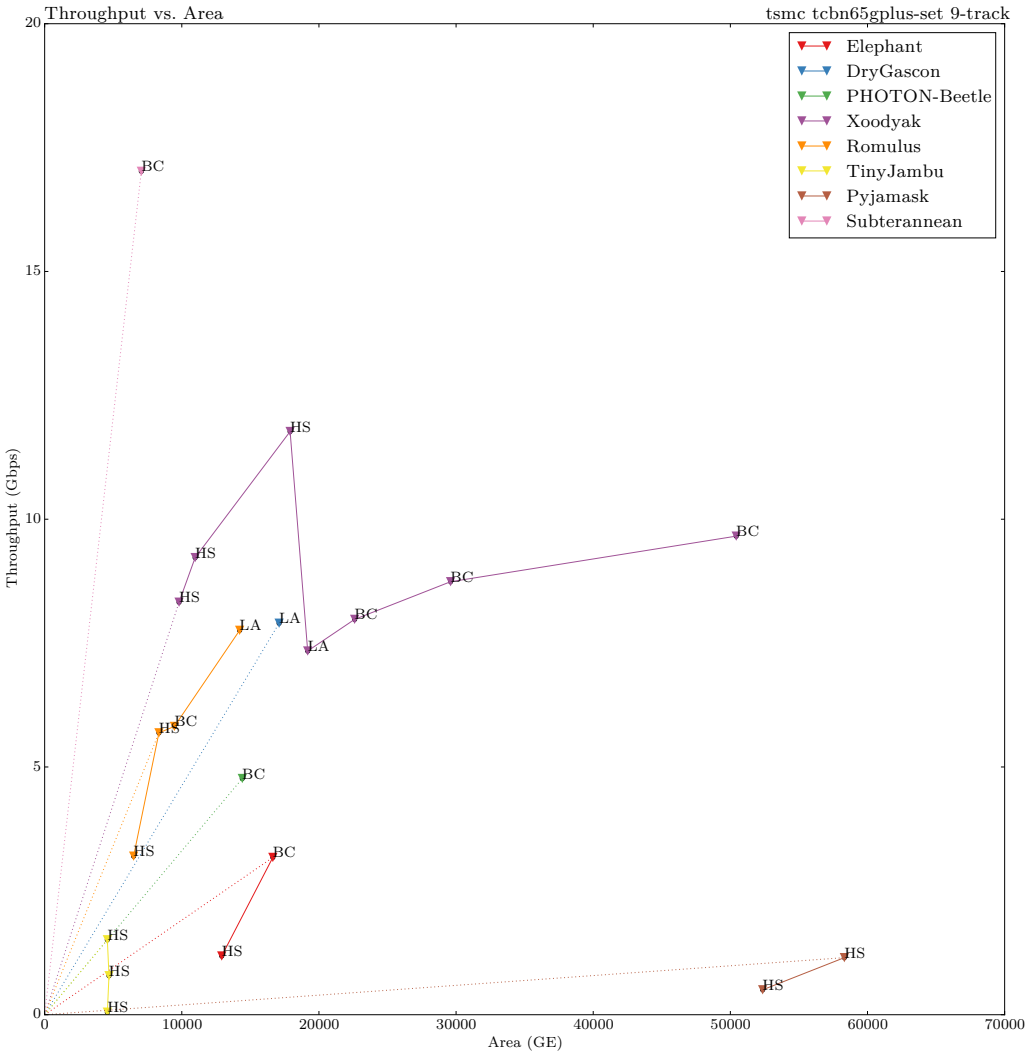


**Figure 6:** Throughput vs. Area for $|M| = 64$ bytes.

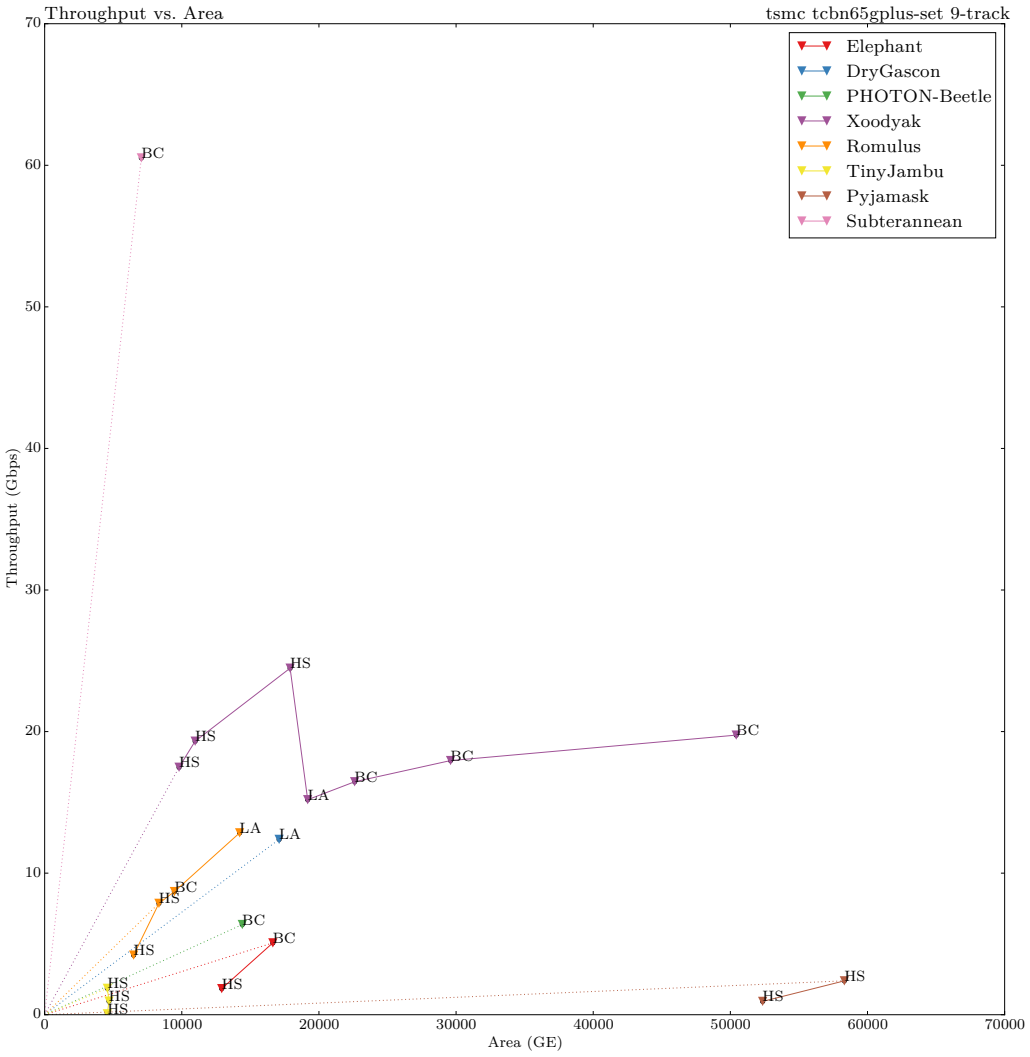


**Figure 7:** Throughput vs. Area for |M| = 1536 bytes.

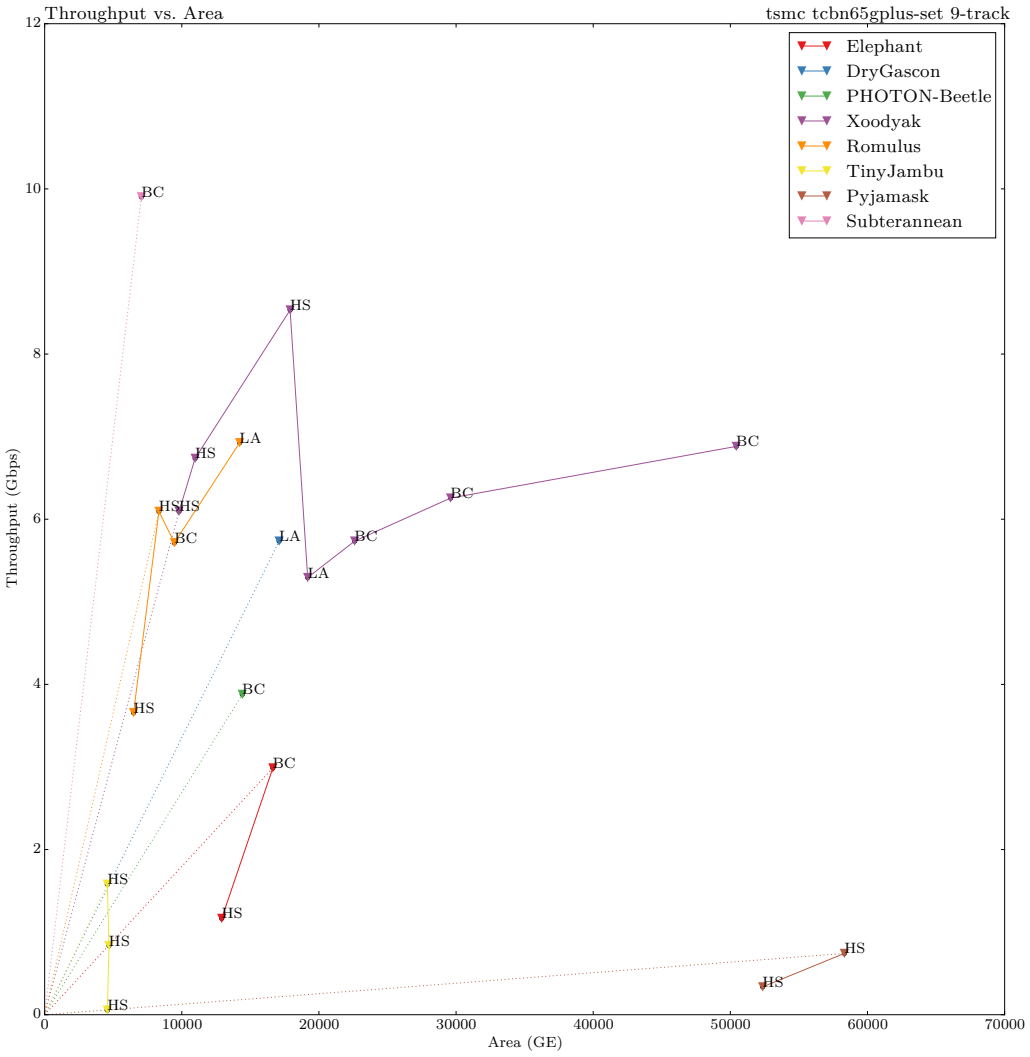


**Figure 8:** Throughput vs. Area for $|A| = |M| = 16$ bytes.

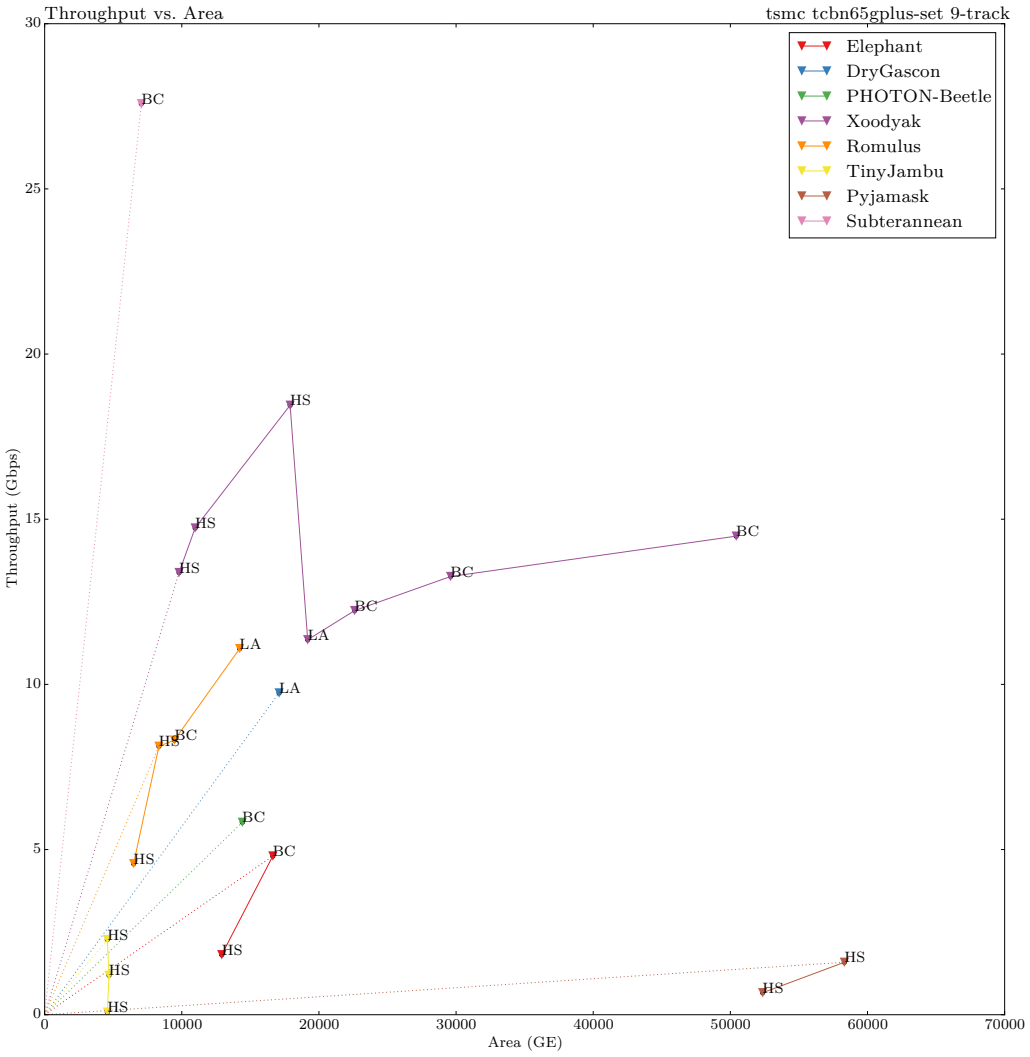


**Figure 9:** Throughput vs. Area for $|A| = |M| = 64$ bytes.

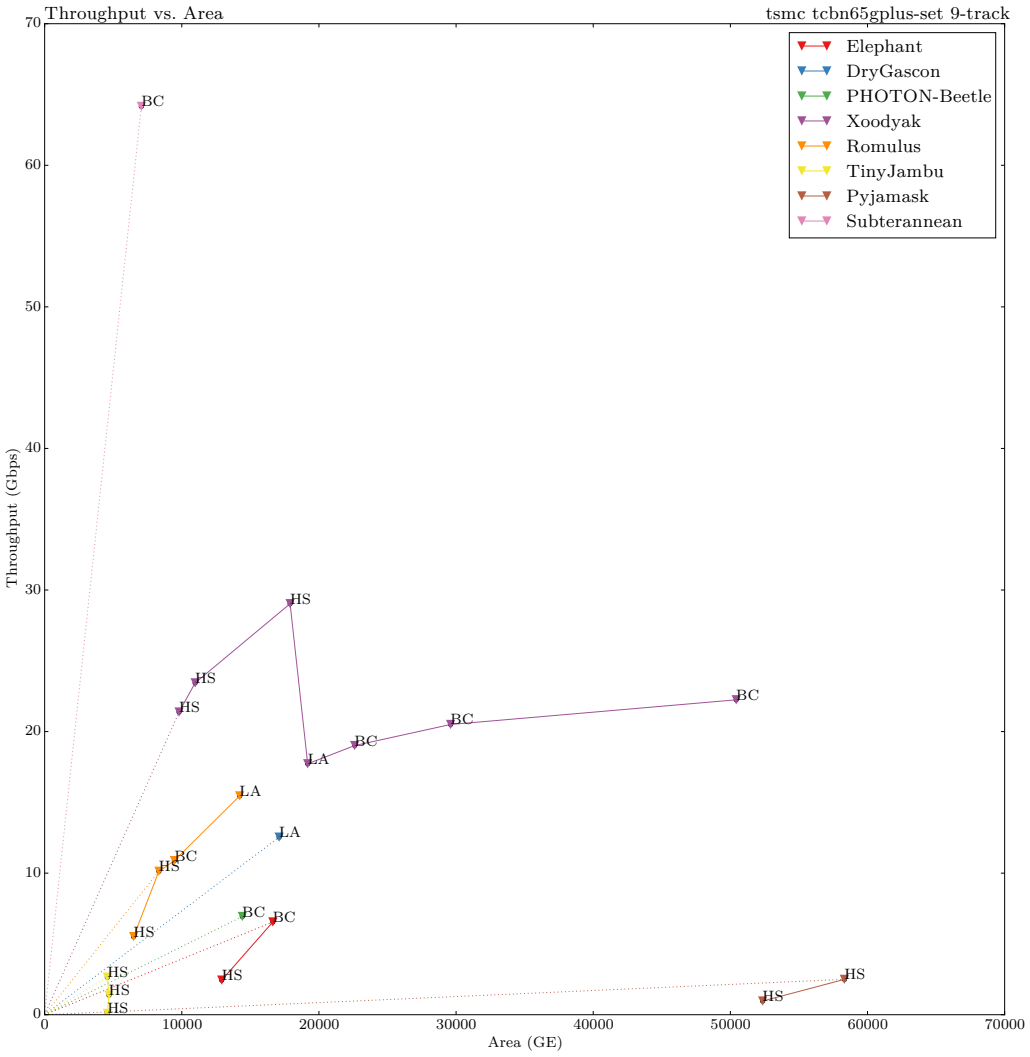


**Figure 10:** Throughput vs. Area for |A| = |M| = 1536 bytes.

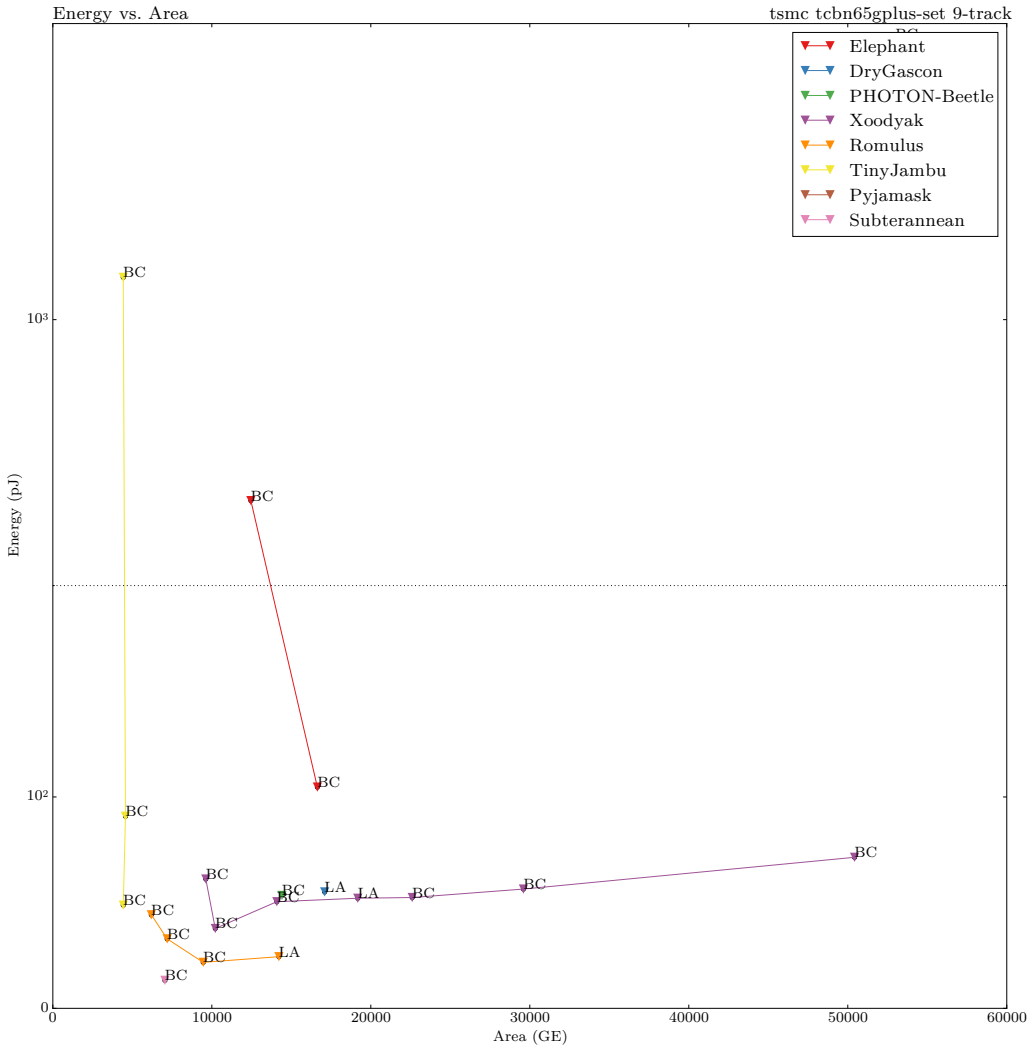


**Figure 11:** Energy vs. Area for $|A| = 16$ bytes. The energy axis follows a log scale above the dotted line.

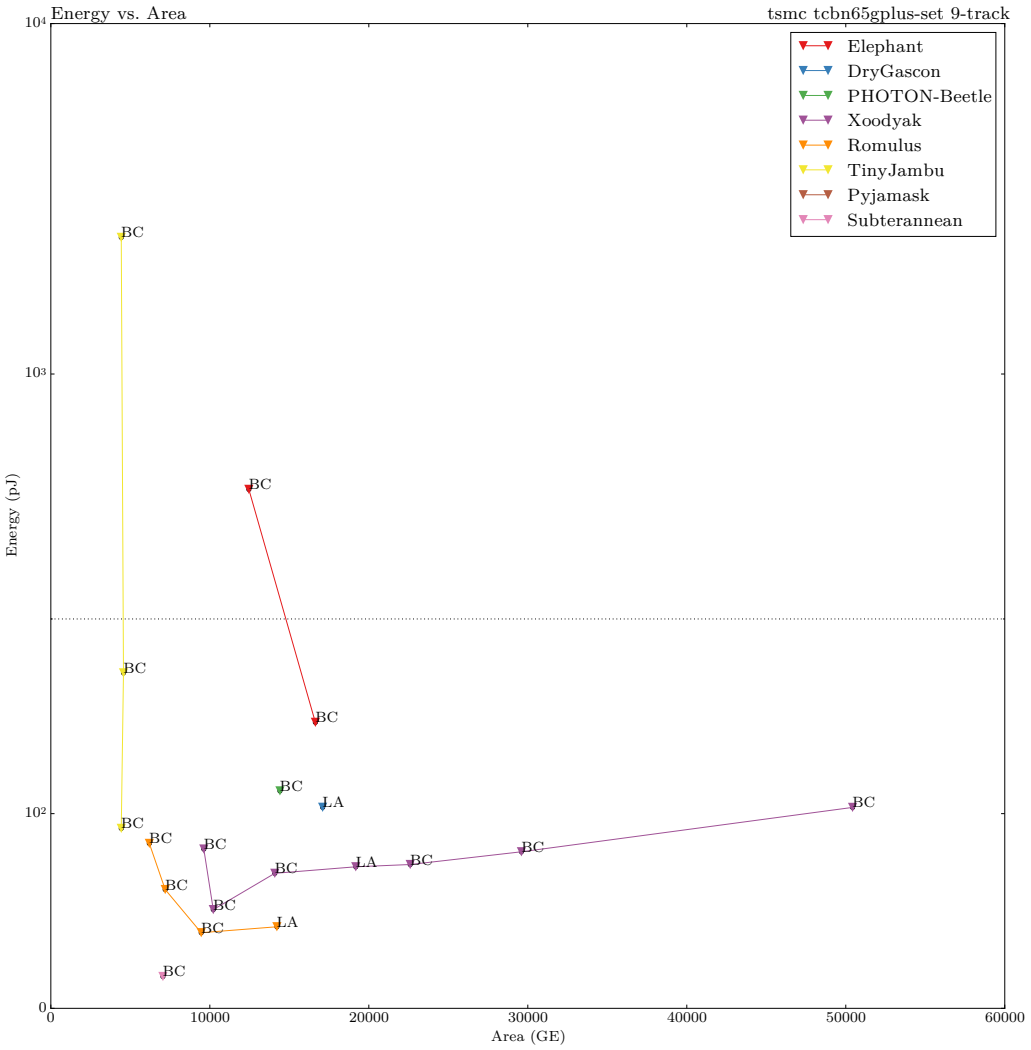


**Figure 12:** Energy vs. Area for $|A| = 64$ bytes. The energy axis follows a log scale above the dotted line.

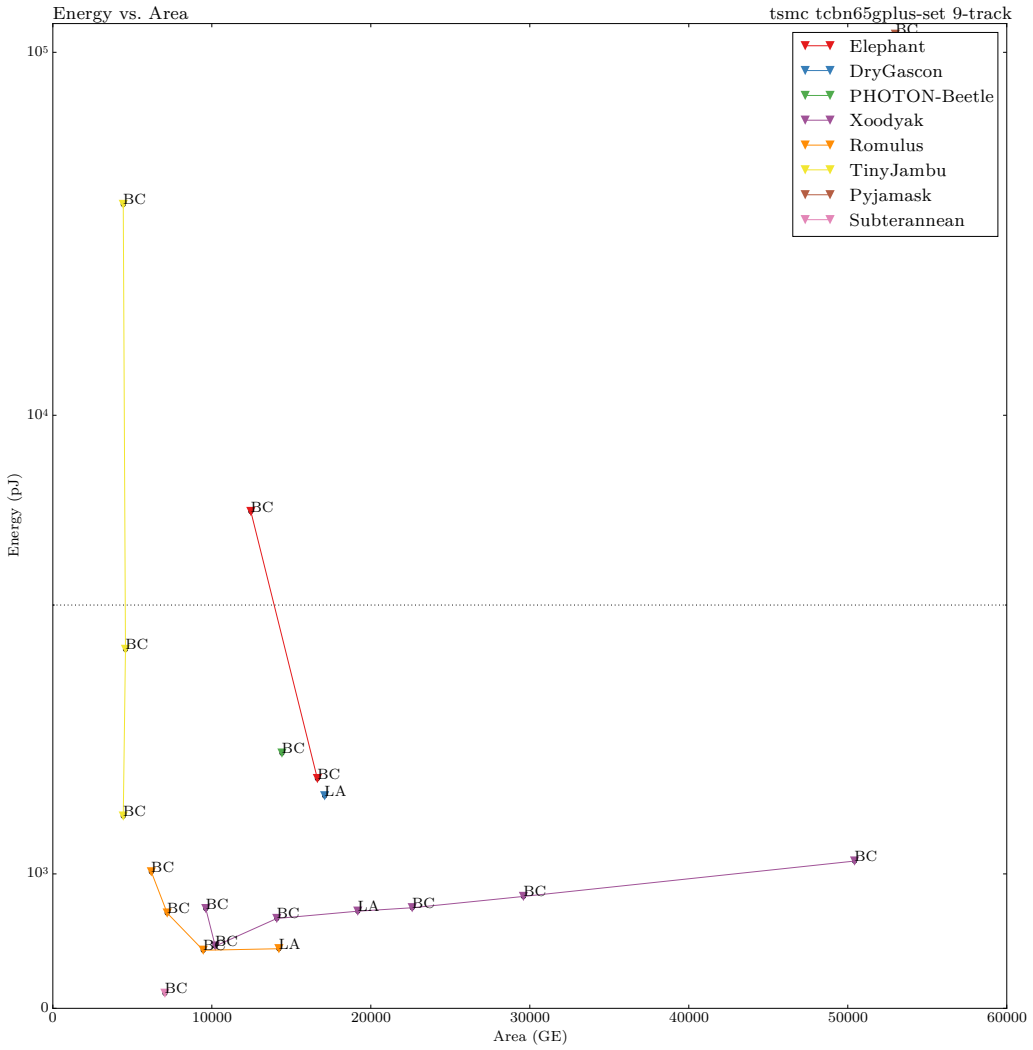


**Figure 13:** Energy vs. Area for $|A| = 1536$ bytes. The energy axis follows a log scale above the dotted line.

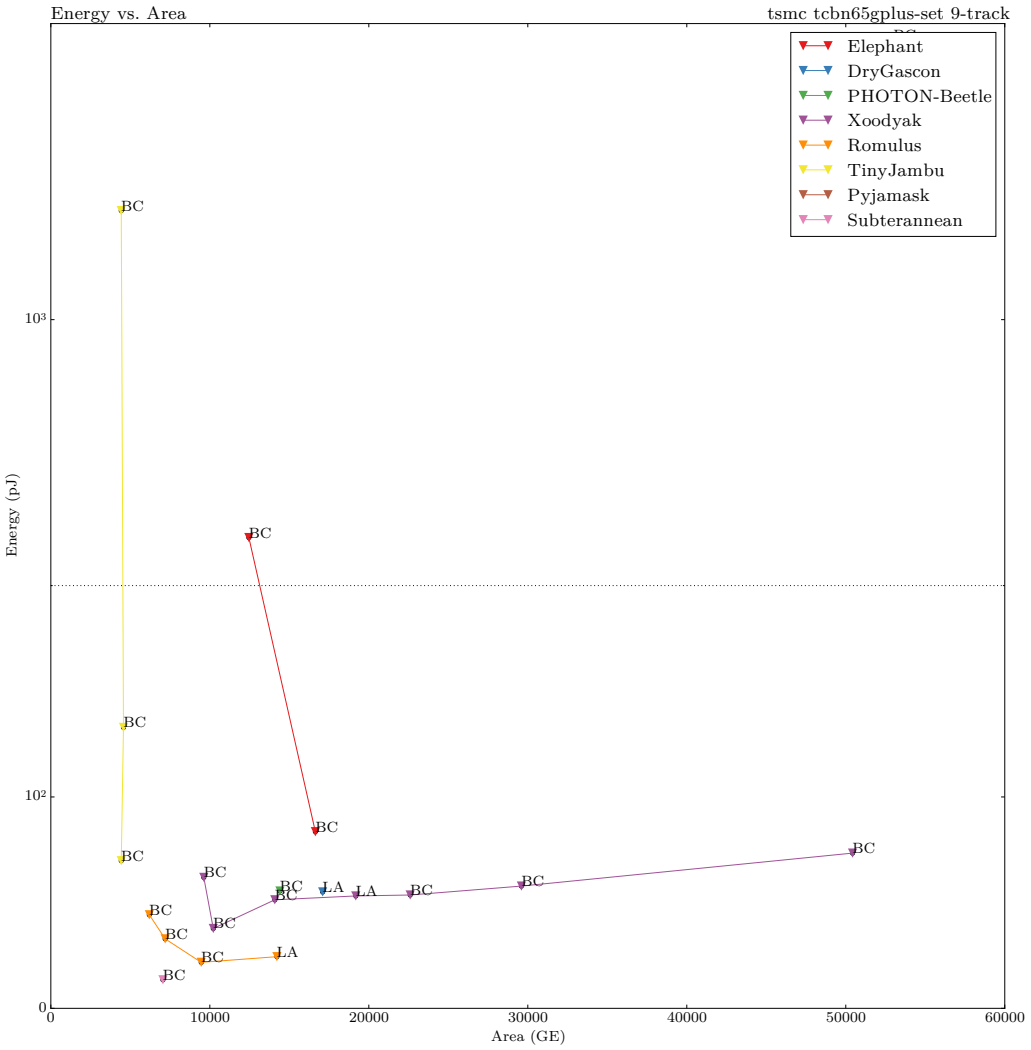


**Figure 14:** Energy vs. Area for |M| = 16 bytes. The energy axis follows a log scale above the dotted line.

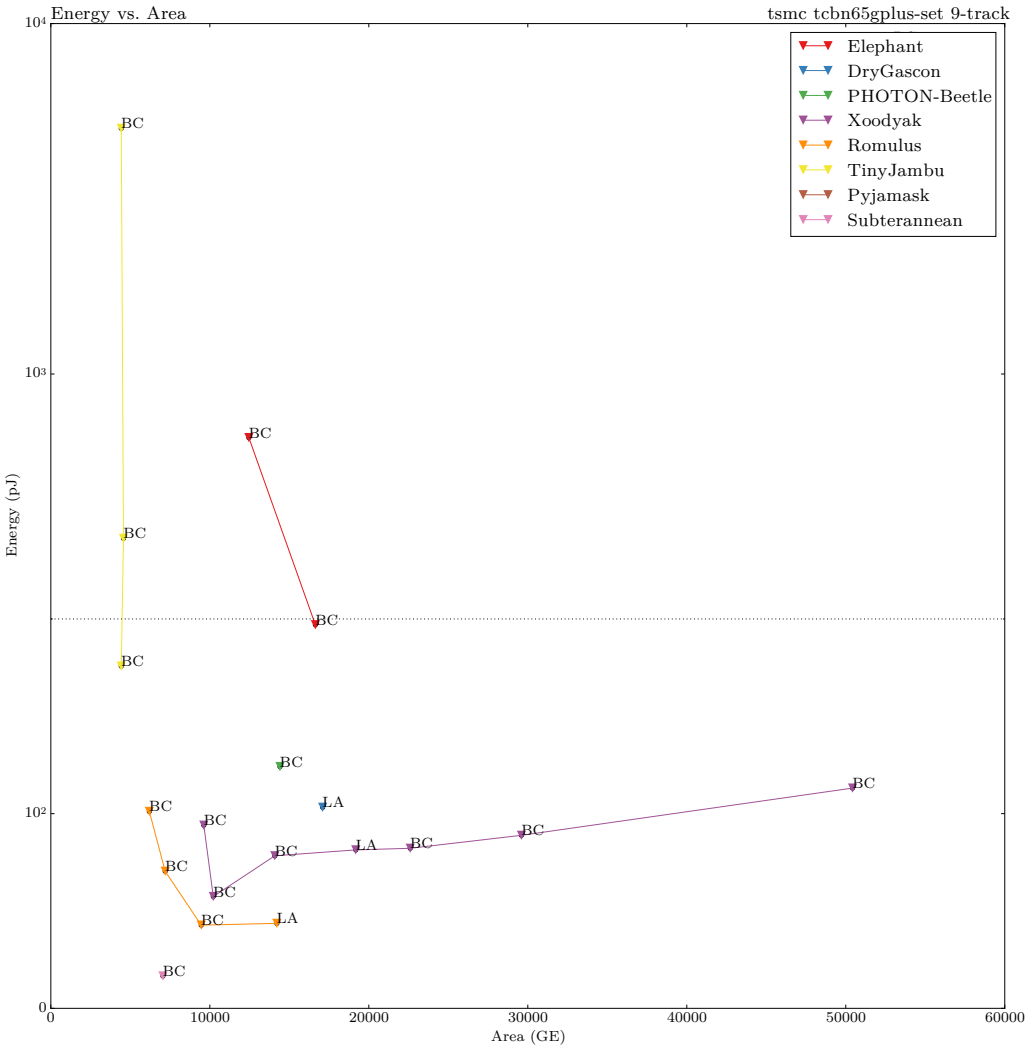


**Figure 15:** Energy vs. Area for $|M| = 64$ bytes. The energy axis follows a log scale above the dotted line.

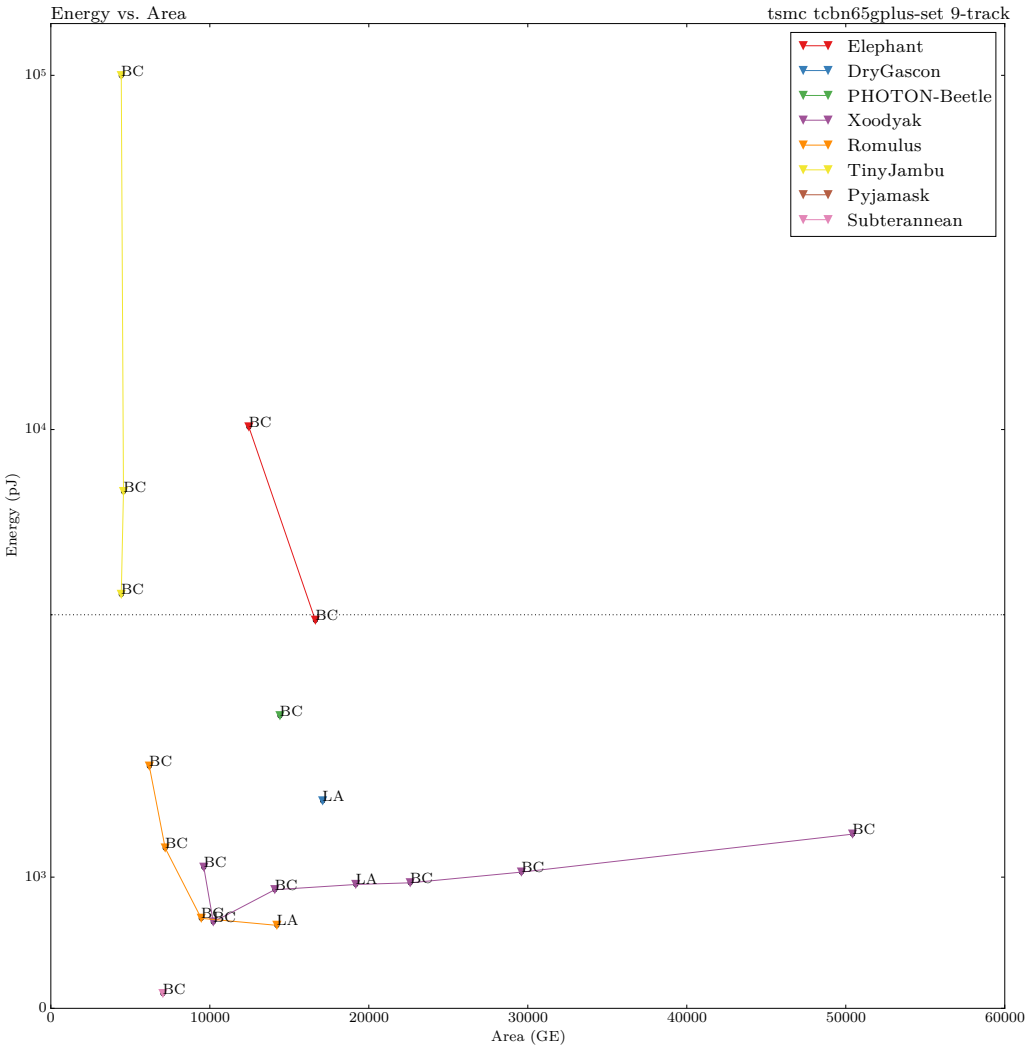


**Figure 16:** Energy vs. Area for $|M| = 1536$ bytes. The energy axis follows a log scale above the dotted line.

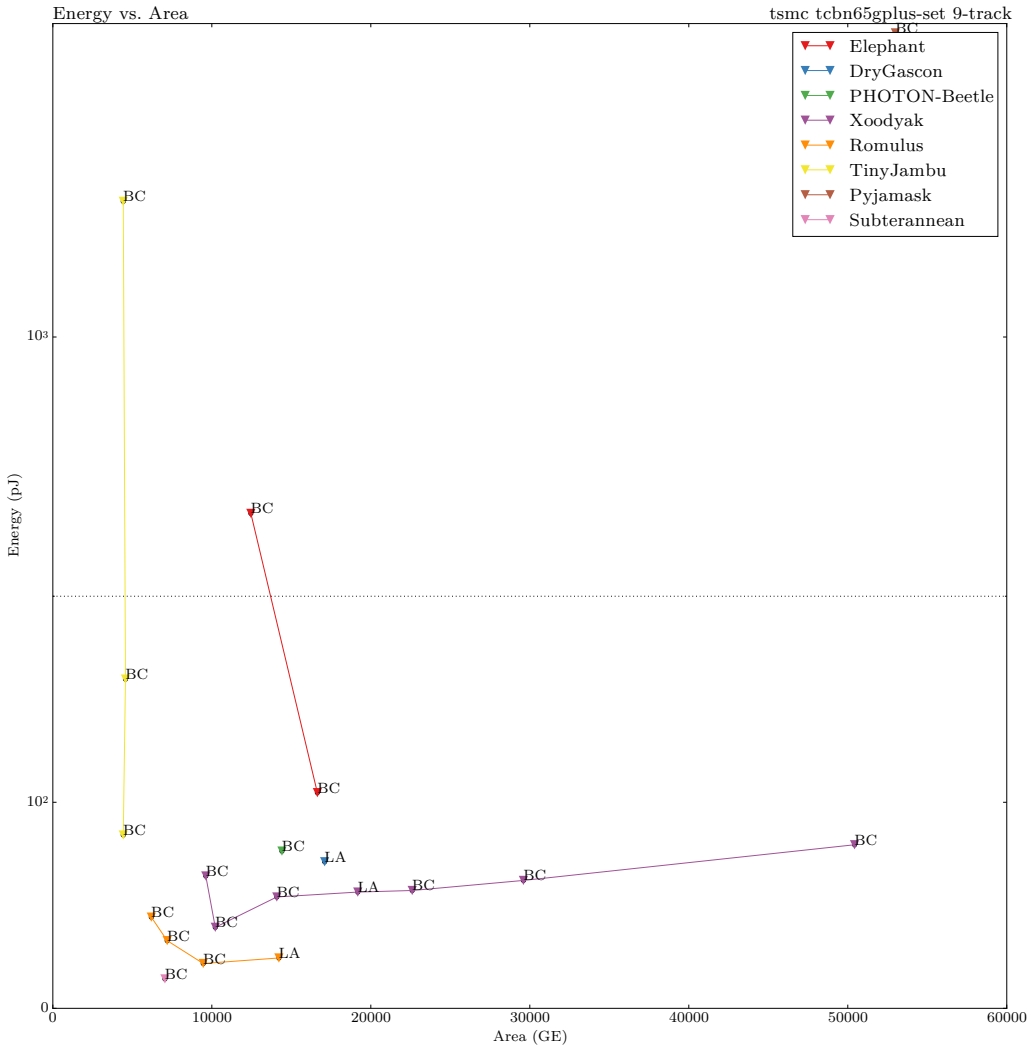


**Figure 17:** Energy vs. Area for $|A| = |M| = 16$ bytes. The energy axis follows a log scale above the dotted line.

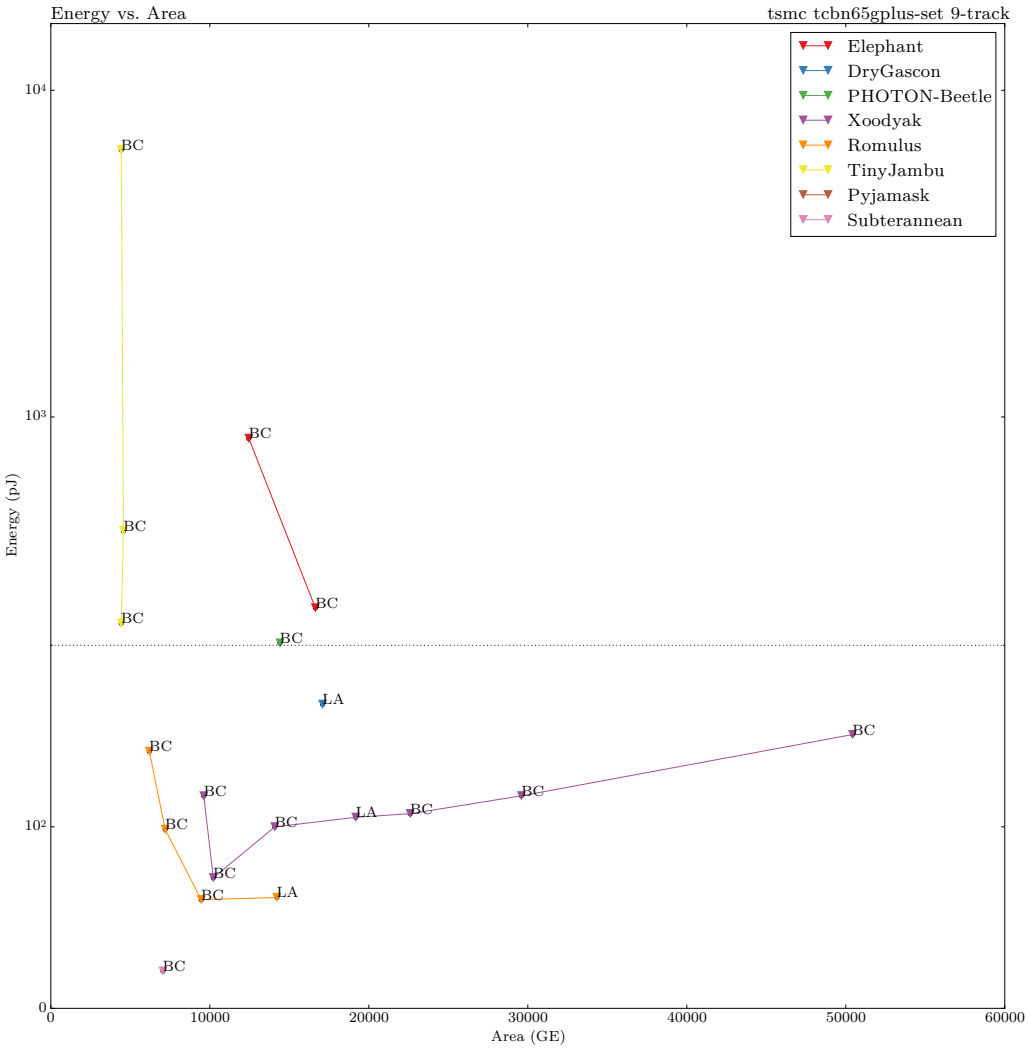


**Figure 18:** Energy vs. Area for $|A| = |M| = 64$ bytes. The energy axis follows a log scale above the dotted line.

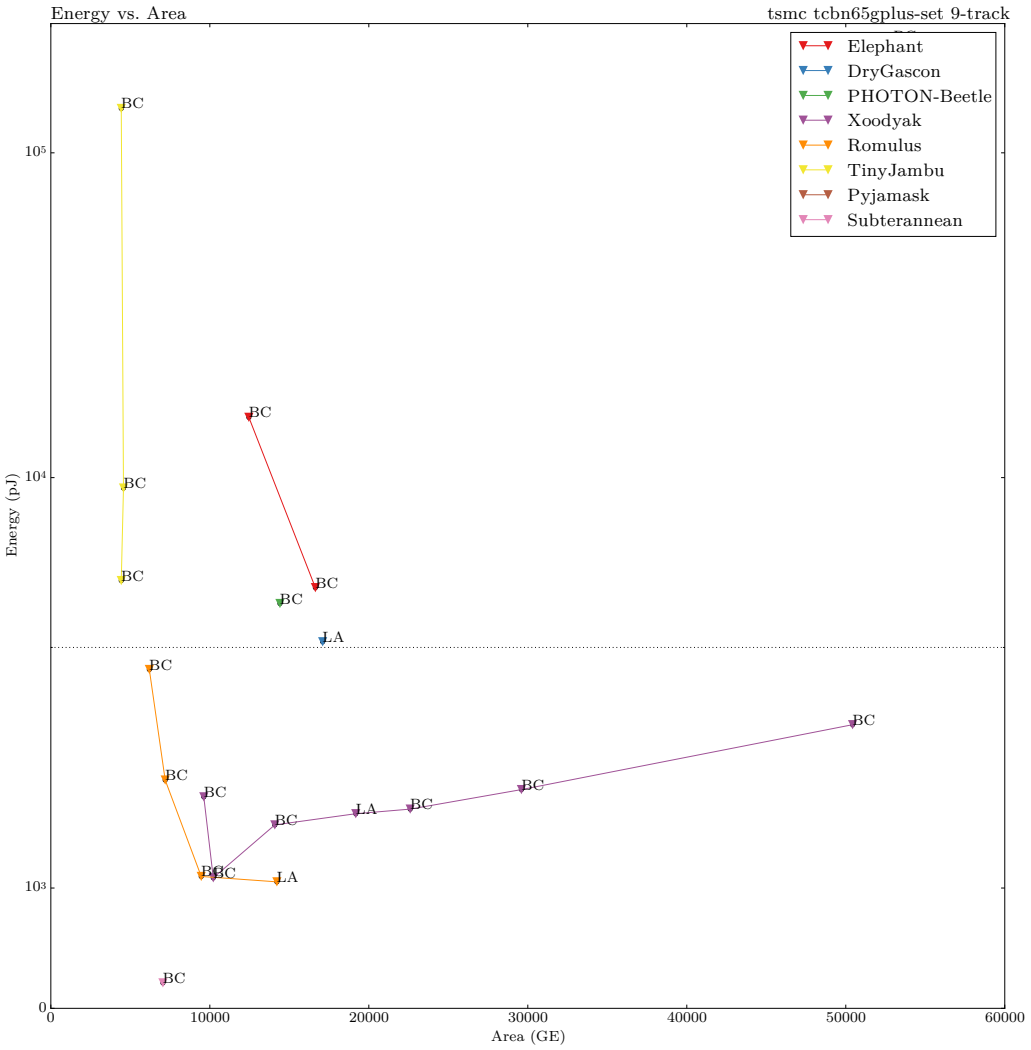


**Figure 19:** Energy vs. Area for $|A| = |M| = 1536$ bytes. The energy axis follows a log scale above the dotted line.

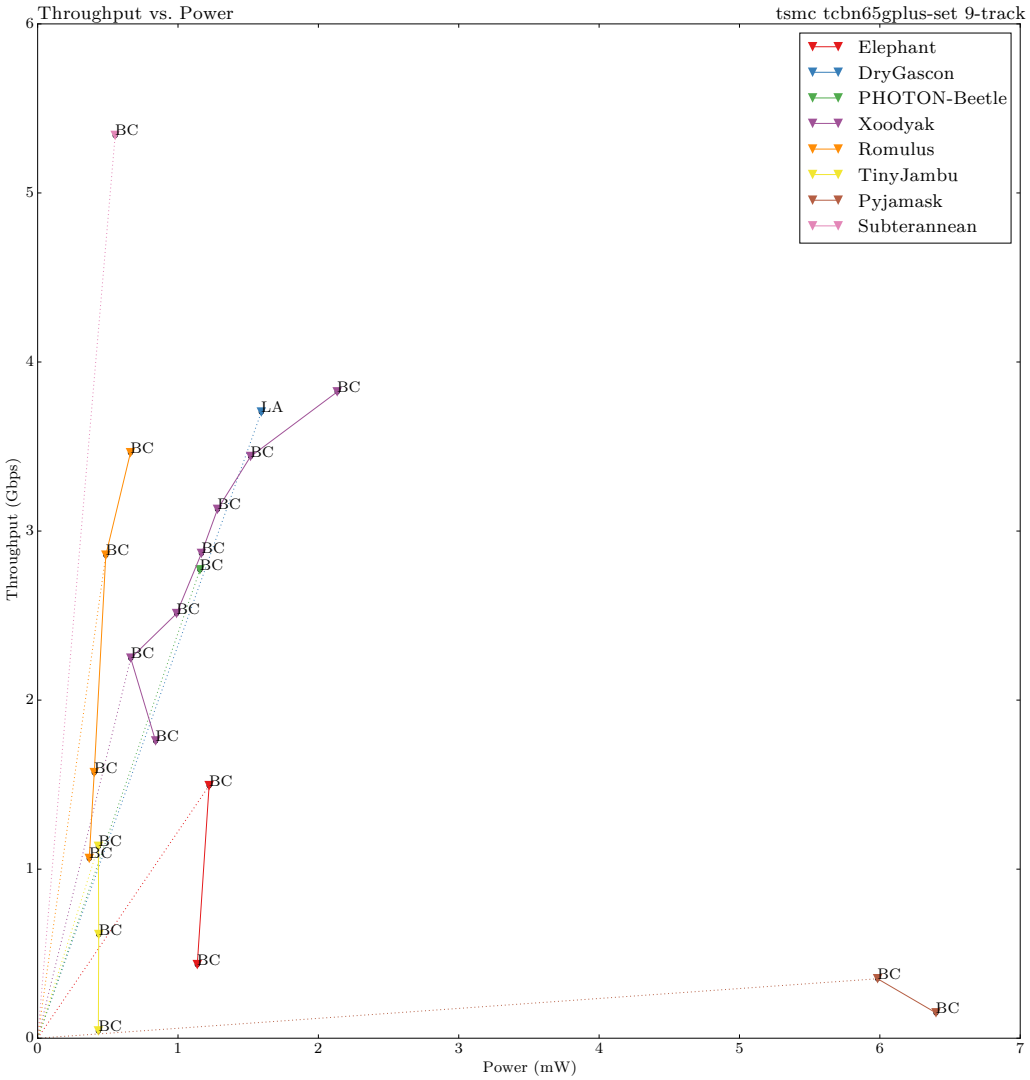


**Figure 20:** Throughput vs. Power for $|A| = 16$ bytes.

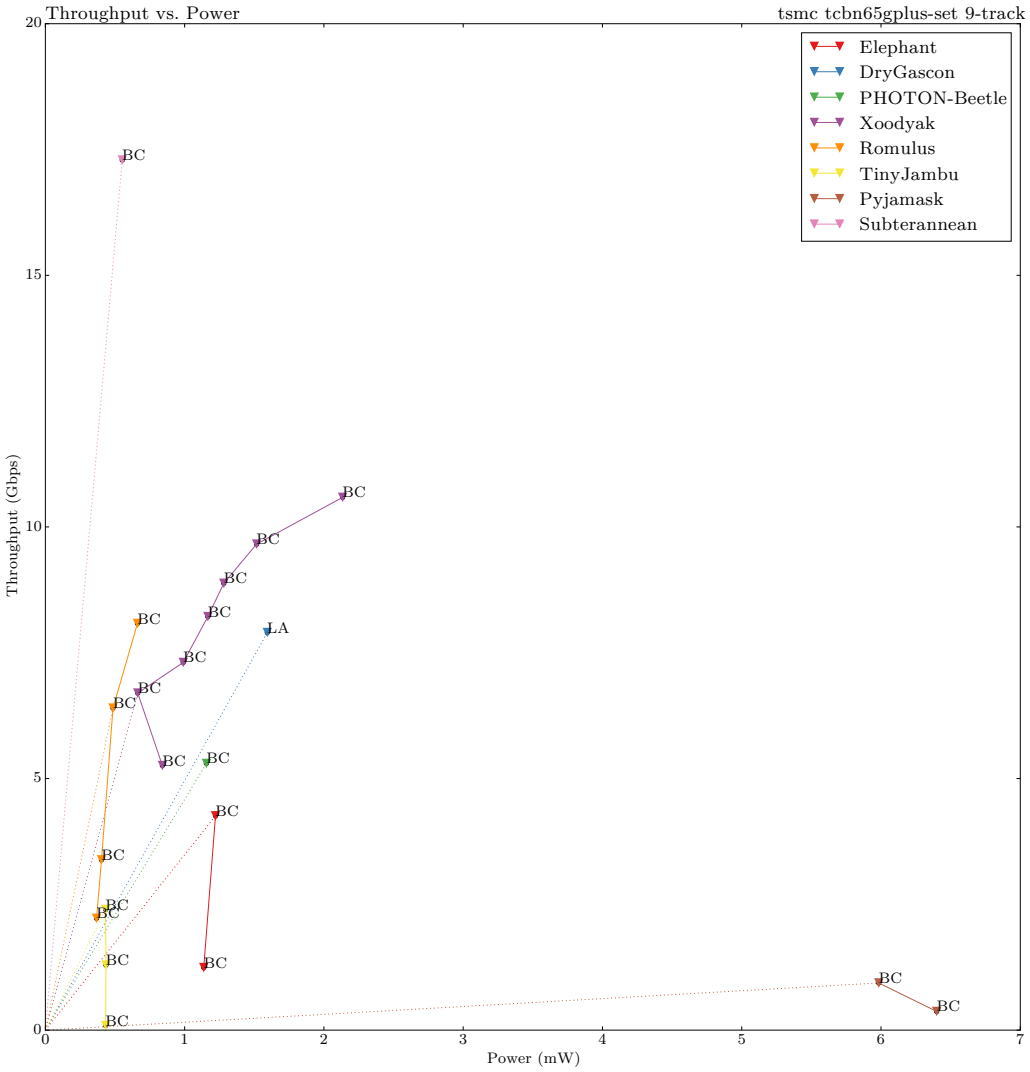


**Figure 21:** Throughput vs. Power for $|A| = 64$ bytes.

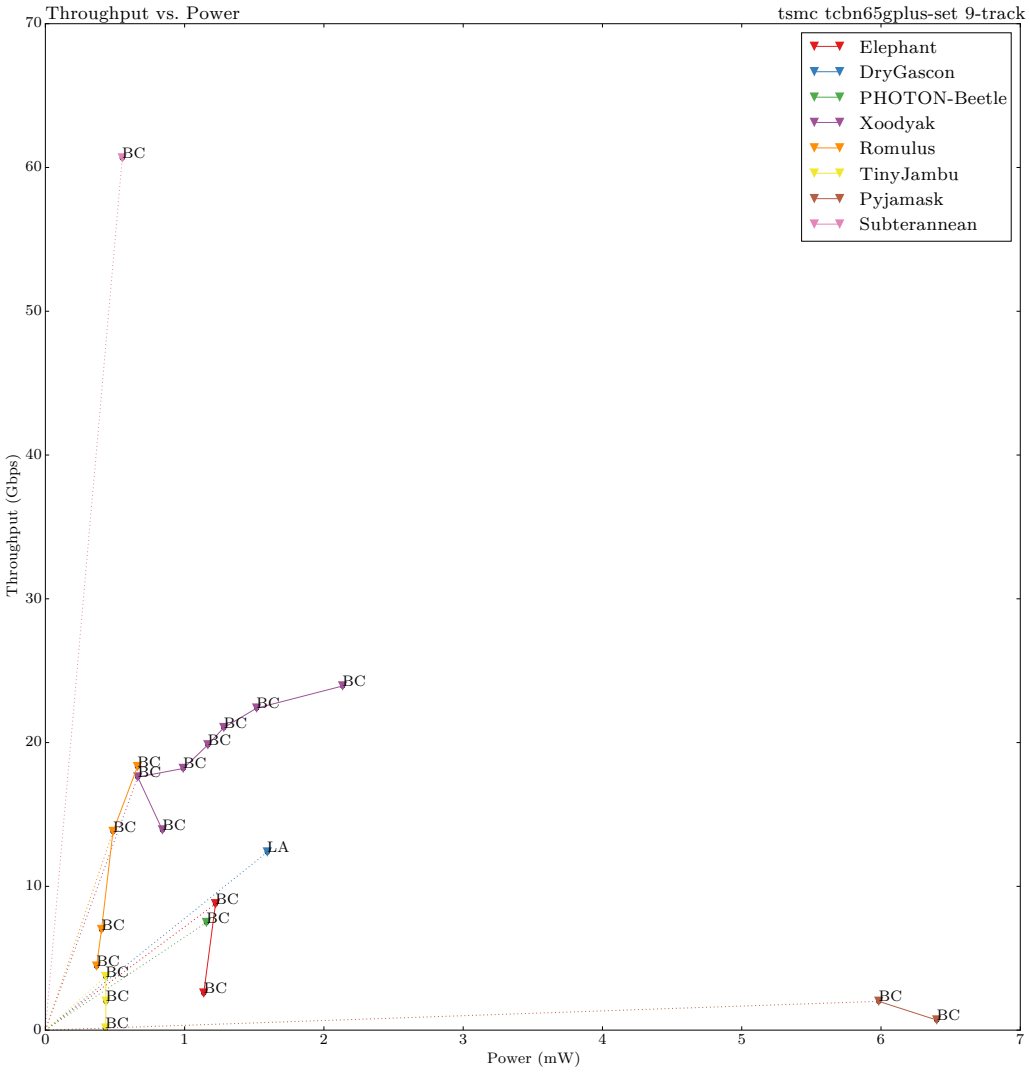


**Figure 22:** Throughput vs. Power for $|A| = 1536$ bytes.

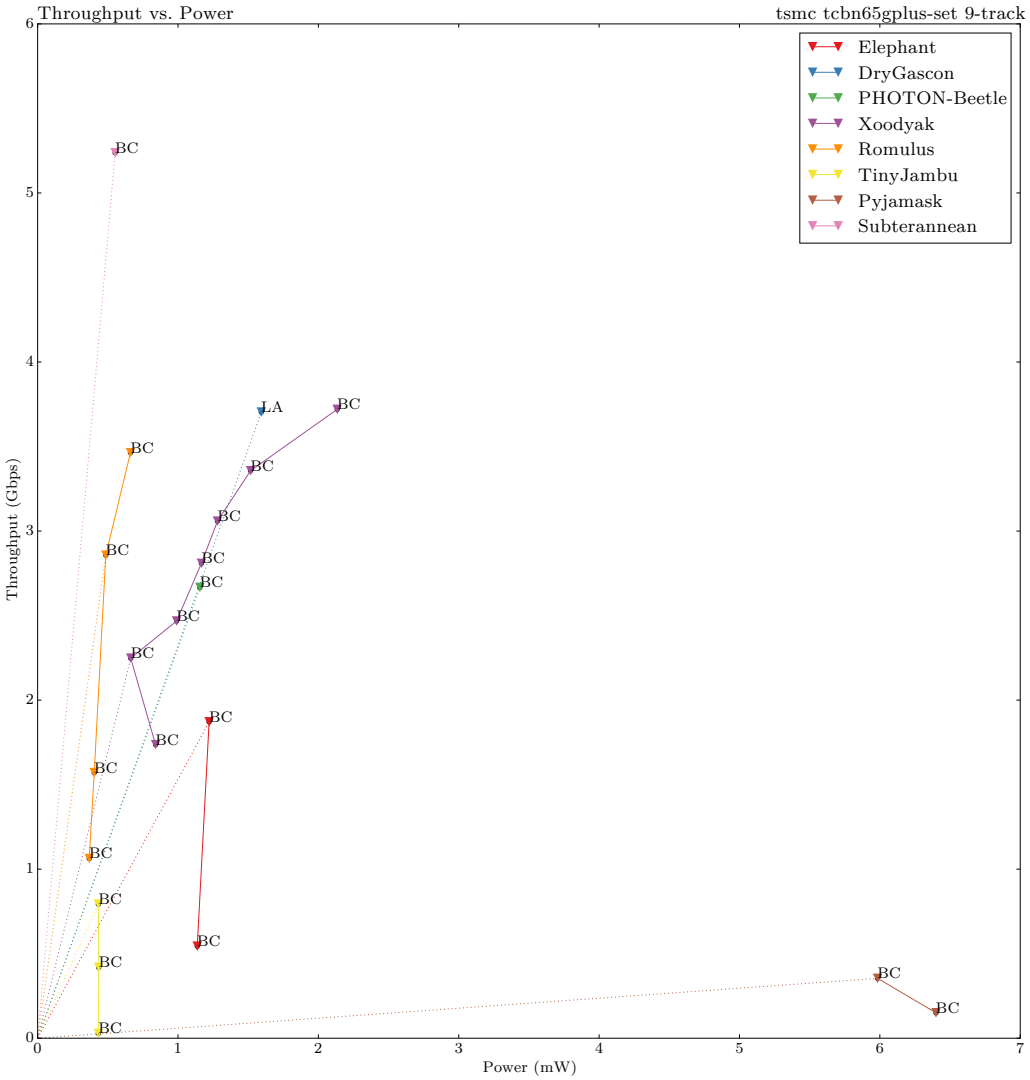


**Figure 23:** Throughput vs. Power for $|M| = 16$ bytes.

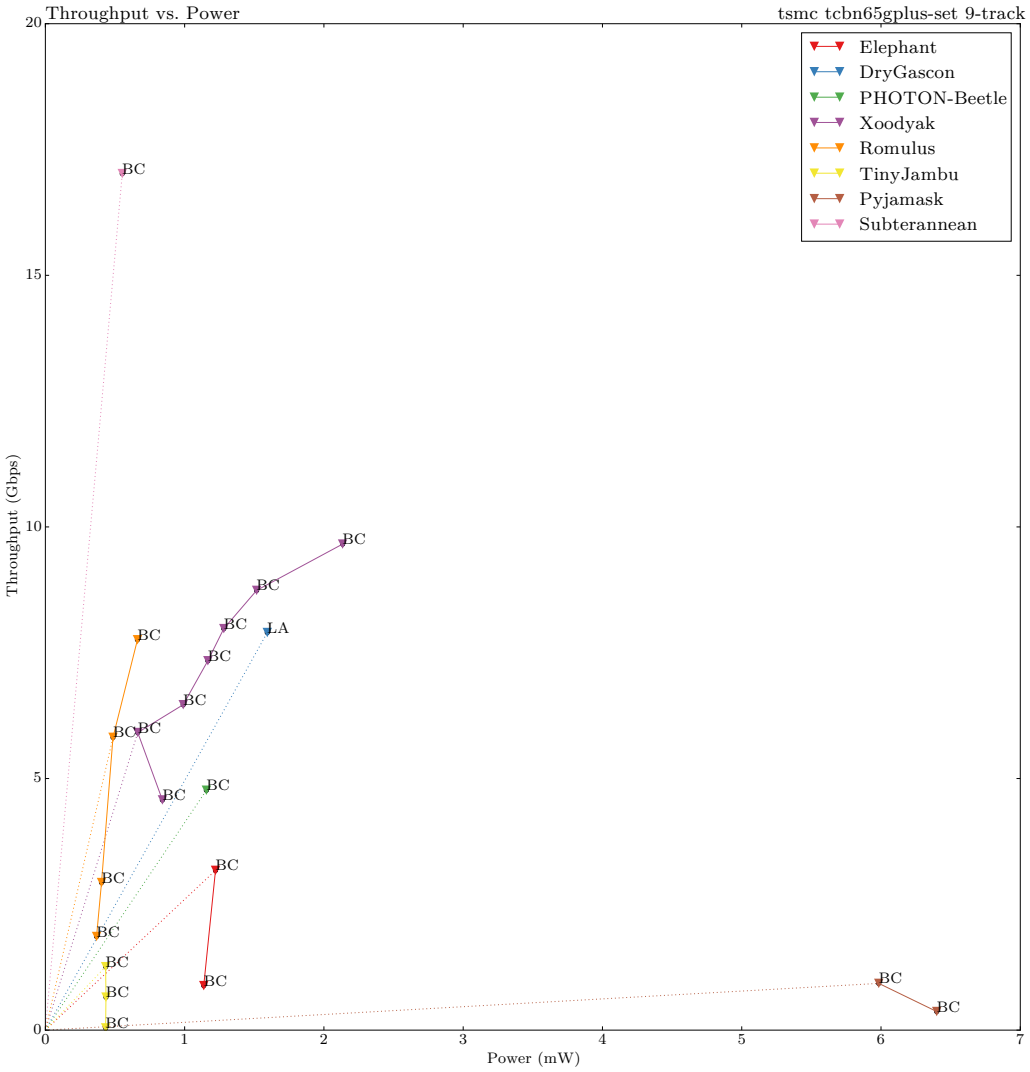


**Figure 24:** Throughput vs. Power for $|M| = 64$ bytes.

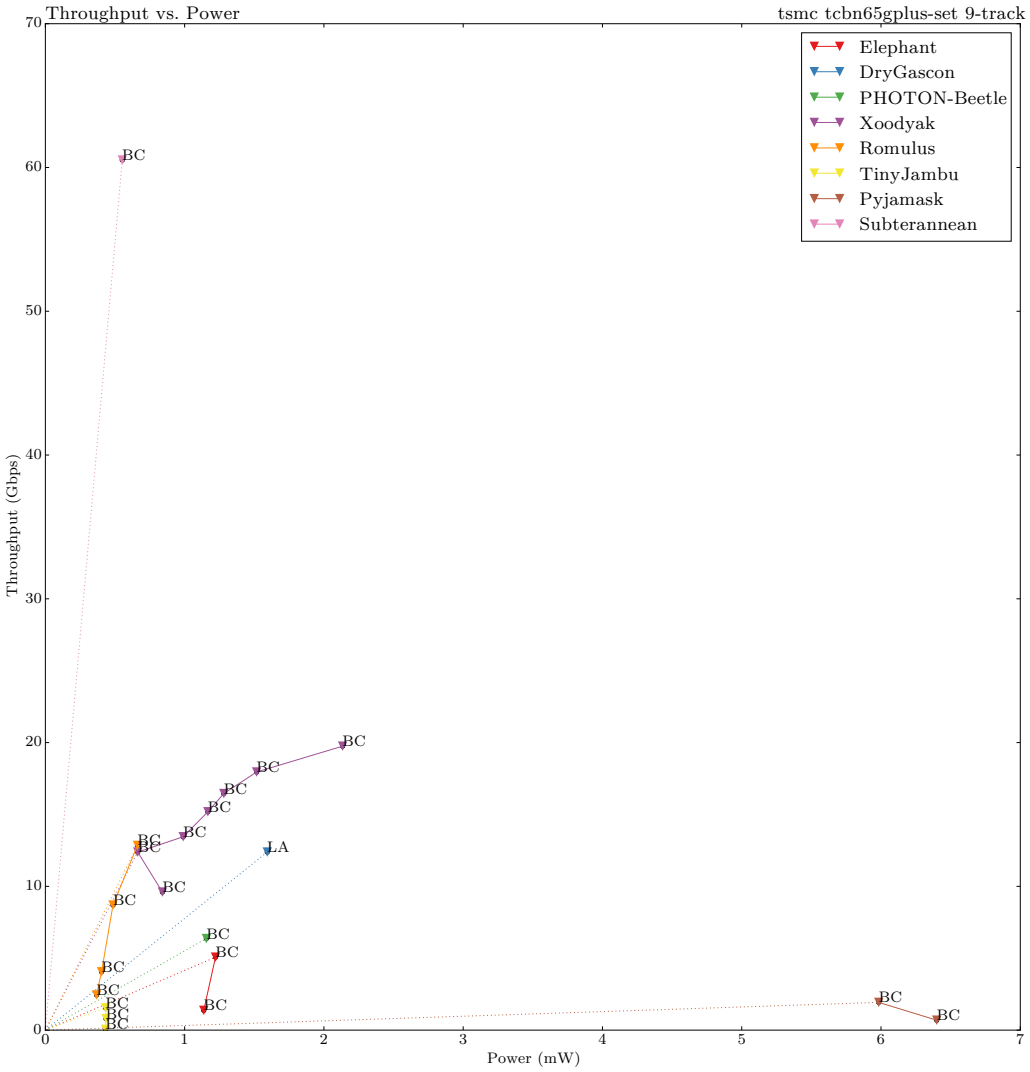


**Figure 25:** Throughput vs. Power for $|M| = 1536$ bytes.

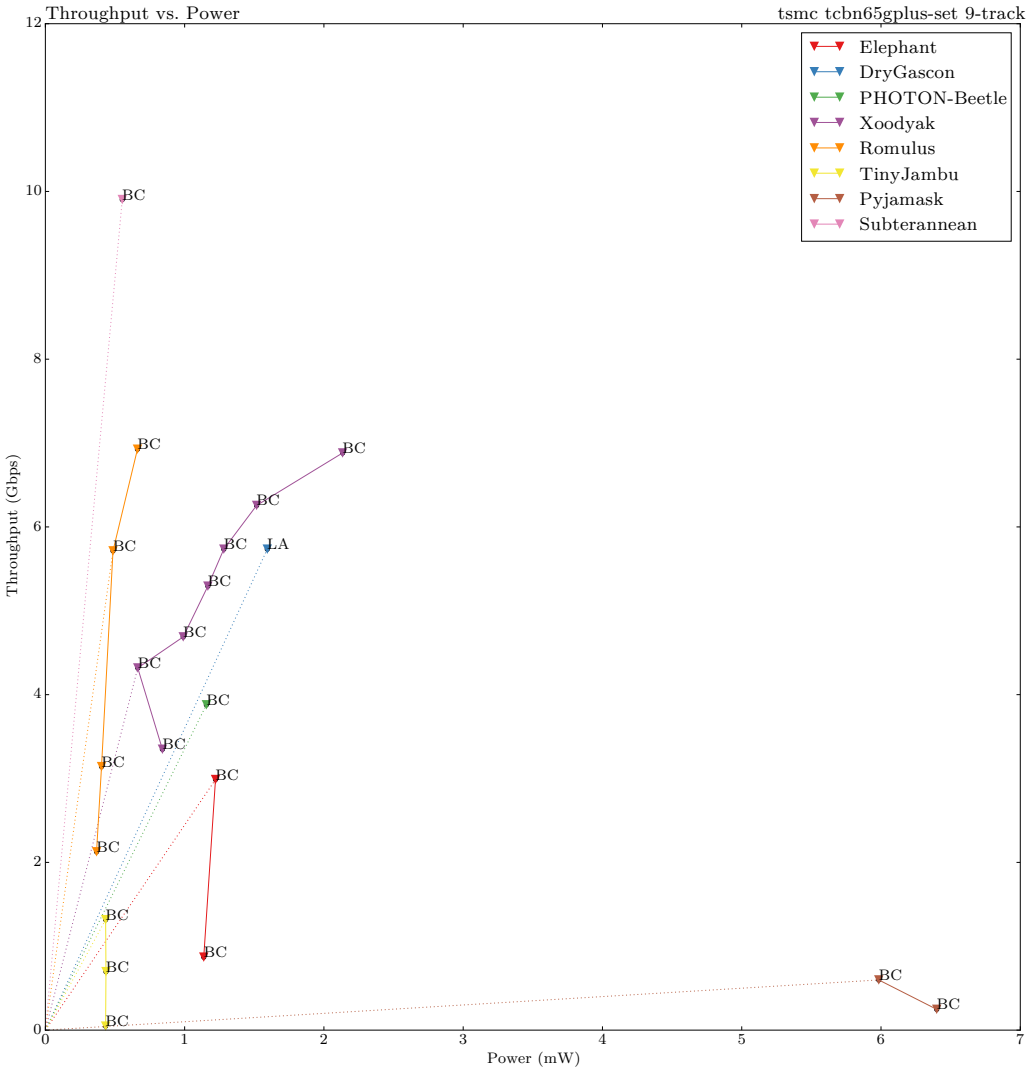


**Figure 26:** Throughput vs. Power for $|A| = |M| = 16$ bytes.

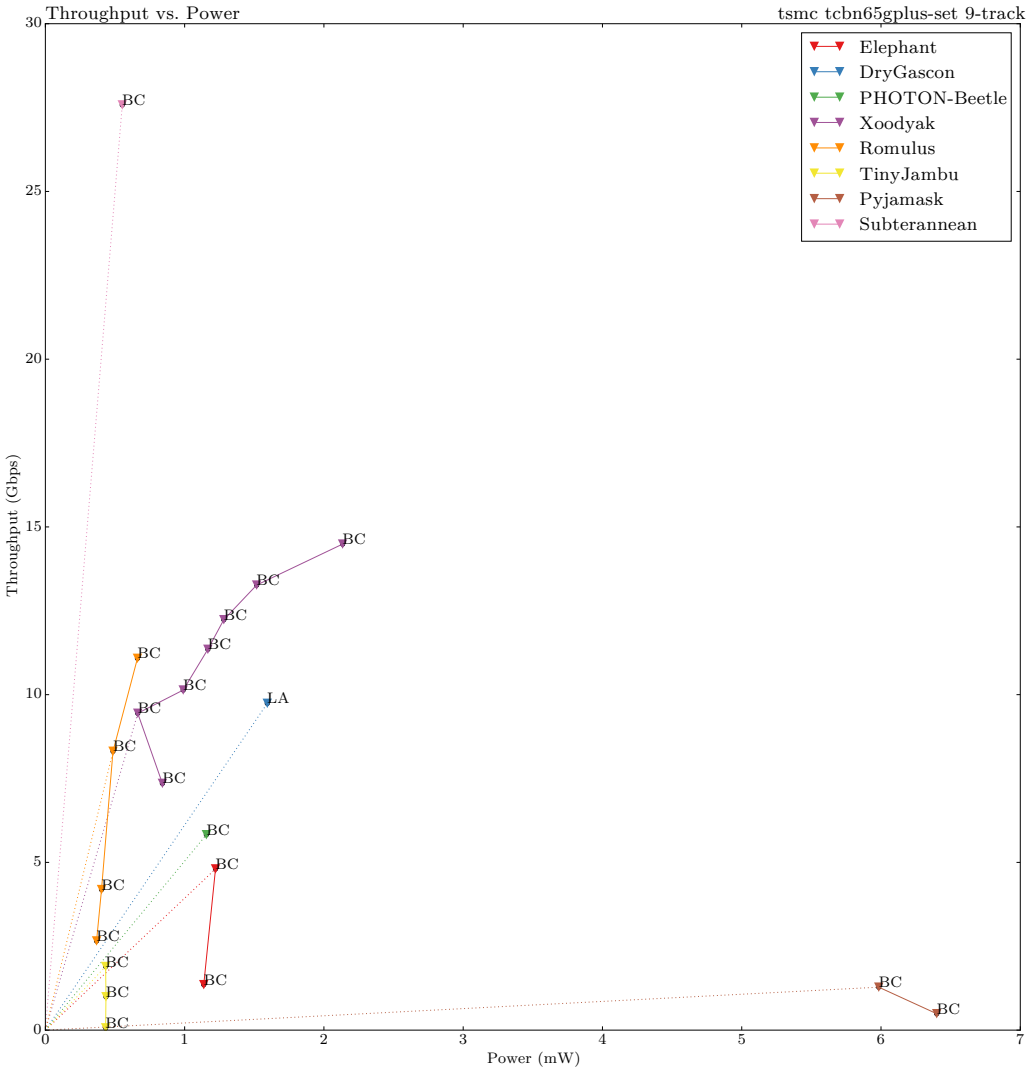


**Figure 27:** Throughput vs. Power for $|\mathrm{A}| = |\mathrm{M}| = 64$ bytes.

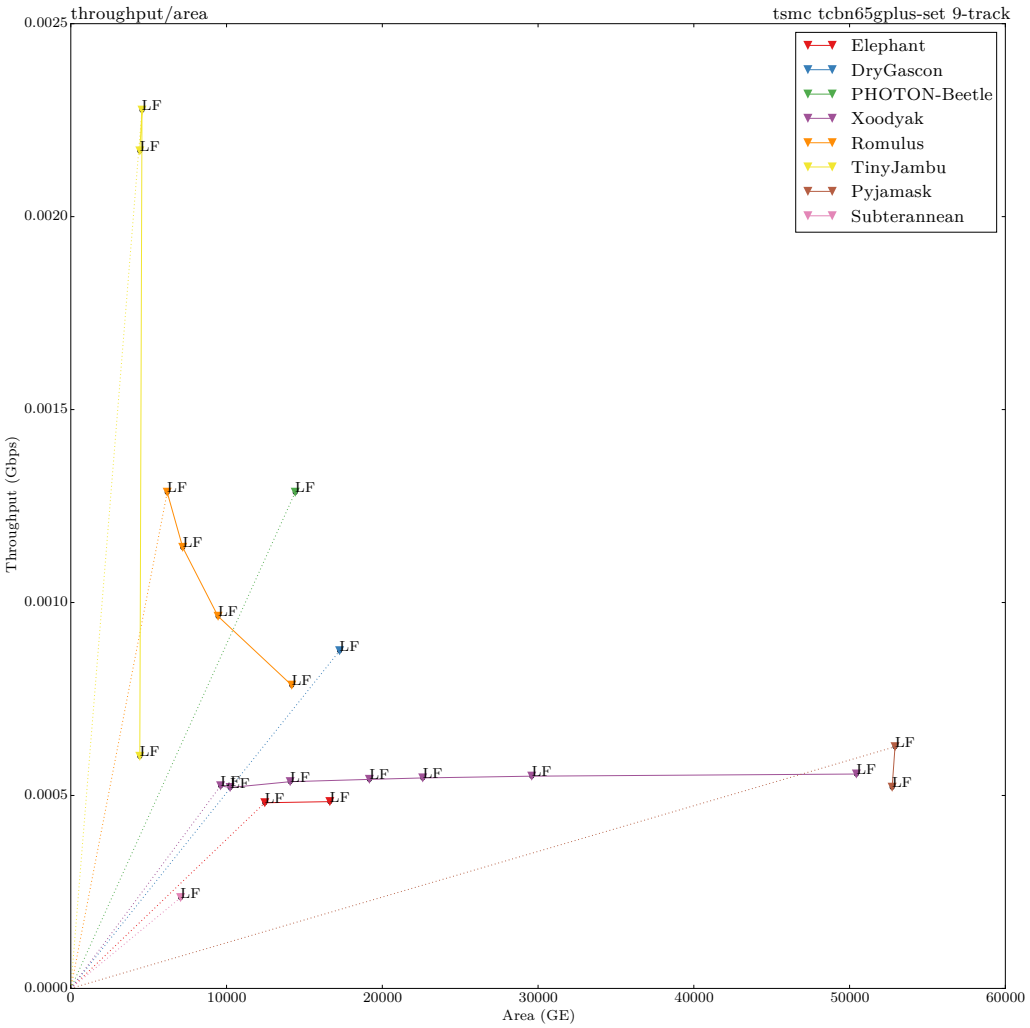


**Figure 28:** 3 Mbps: Throughput vs. Area for |A| = 16 bytes.

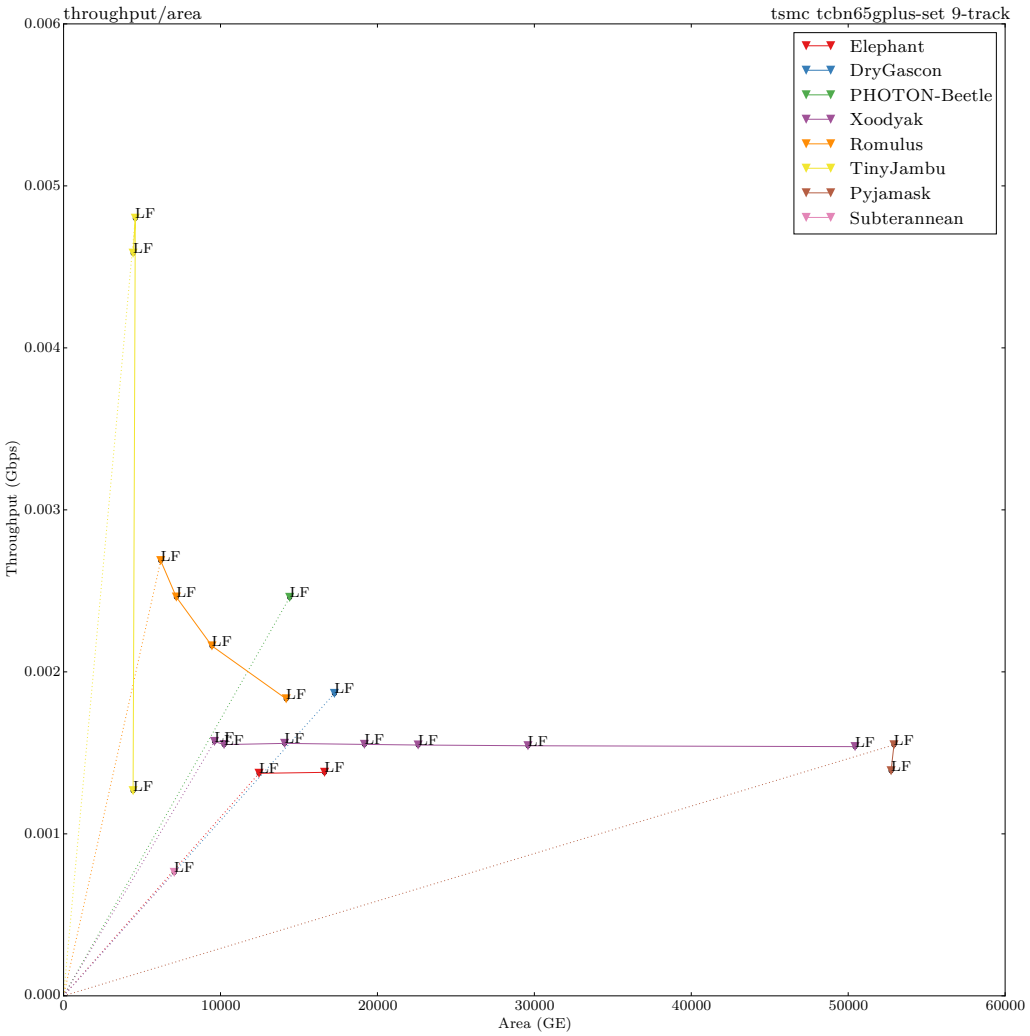


**Figure 29:** 3 Mbps: Throughput vs. Area for $|A| = 64$ bytes.

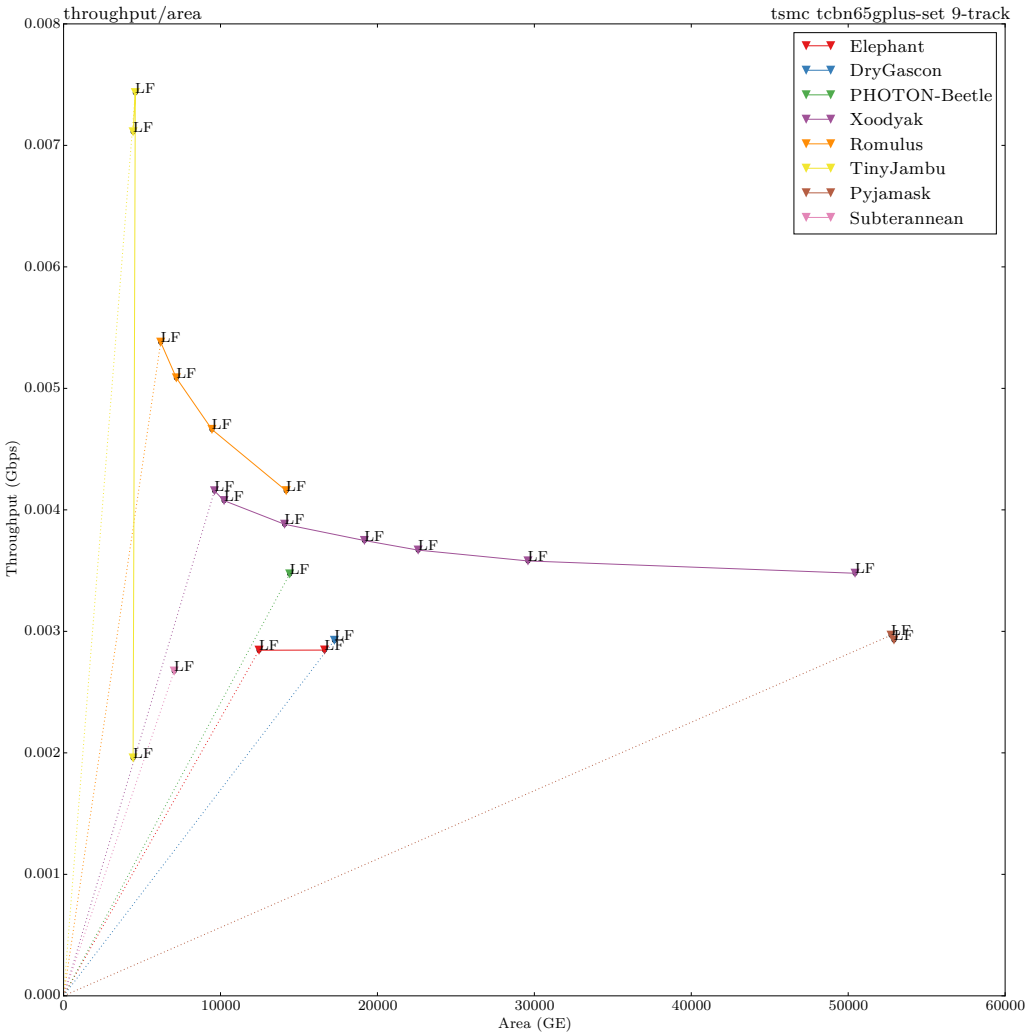


**Figure 30:** 3 Mbps: Throughput vs. Area for $|A| = 1536$ bytes.

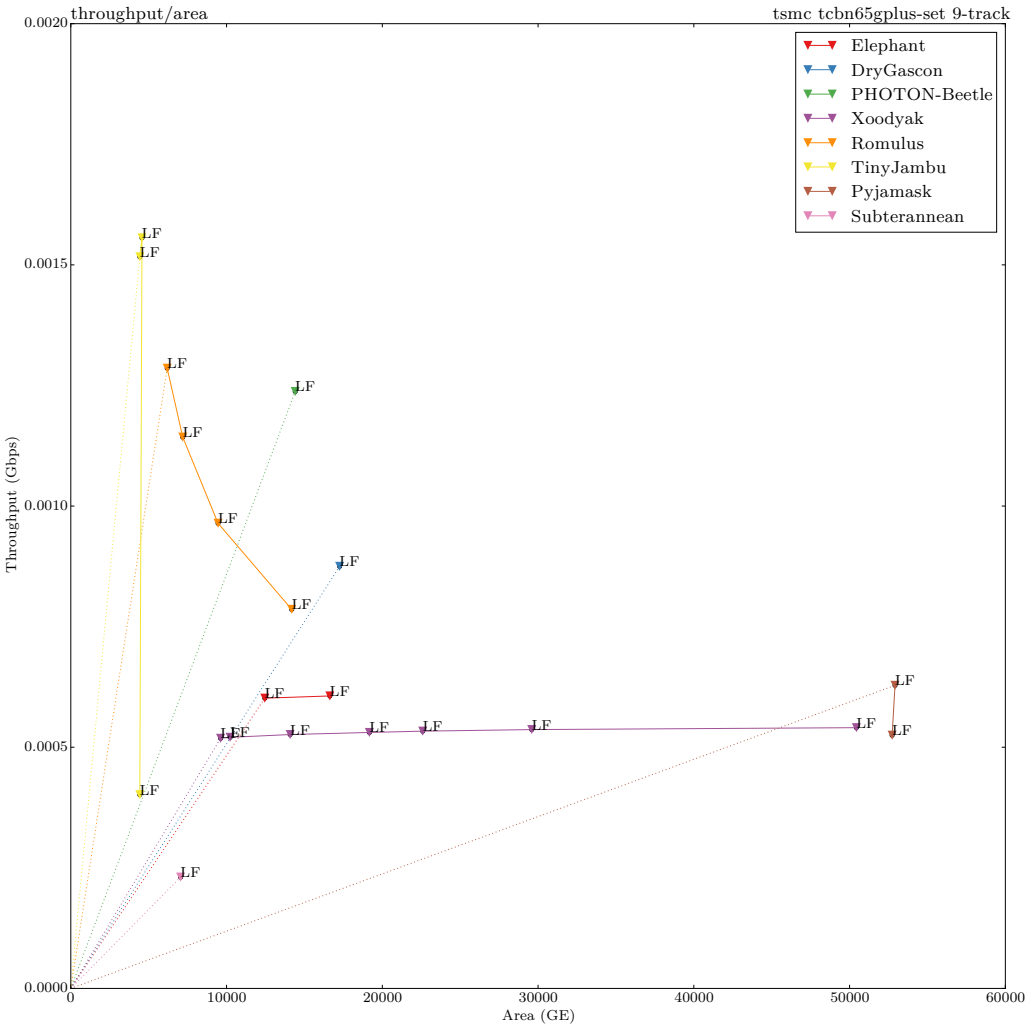


**Figure 31:** 3 Mbps: Throughput vs. Area for |M| = 16 bytes.

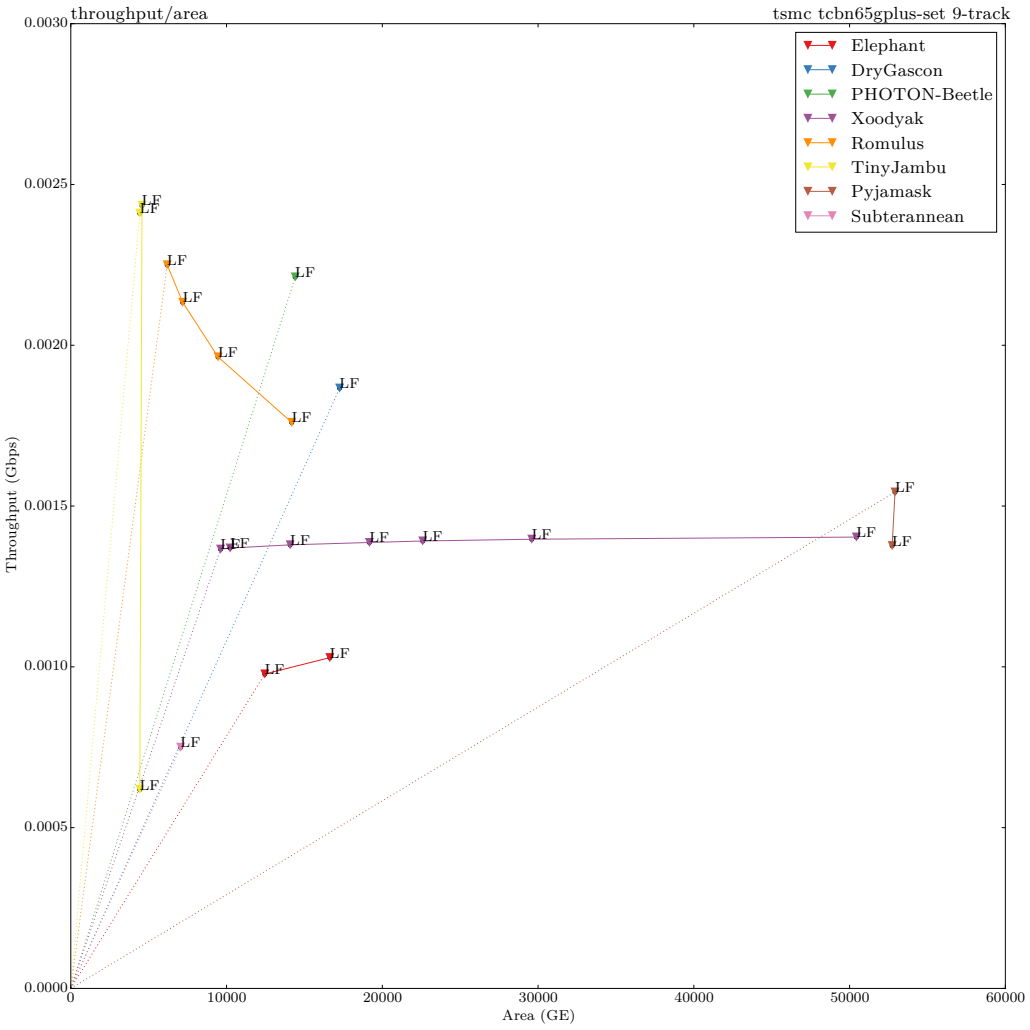


**Figure 32:** 3 Mbps: Throughput vs. Area for |M| = 64 bytes.

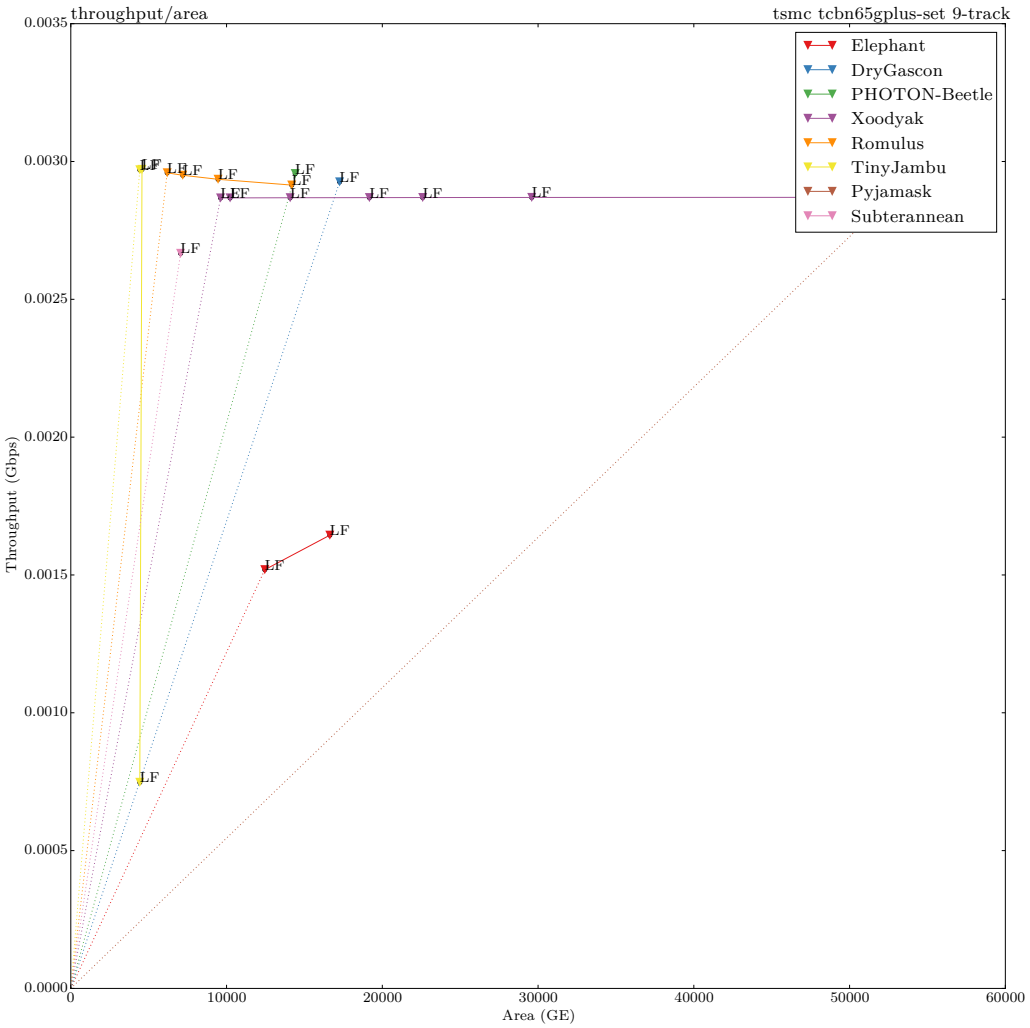


**Figure 33:** 3 Mbps: Throughput vs. Area for |M| = 1536 bytes.

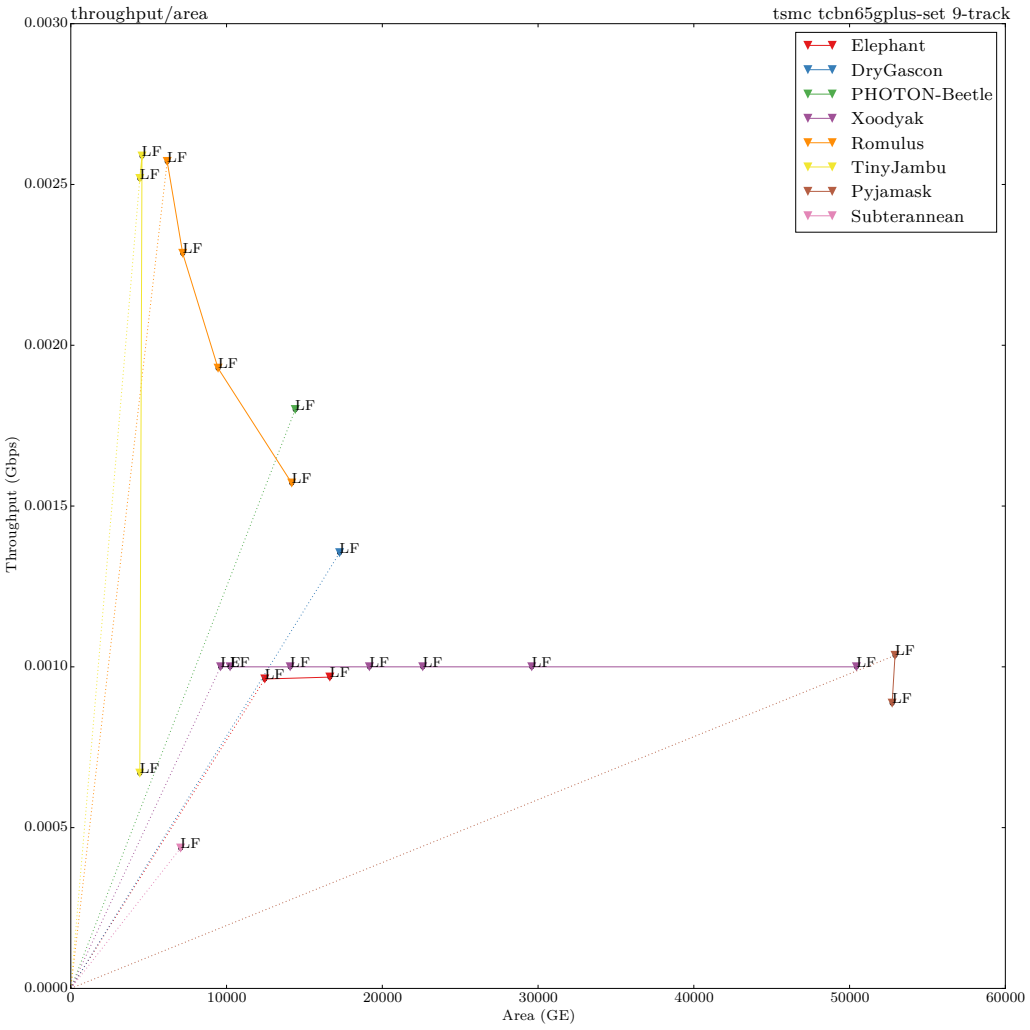


**Figure 34:** 3 Mbps: Throughput vs. Area for |A| = |M| = 16 bytes.

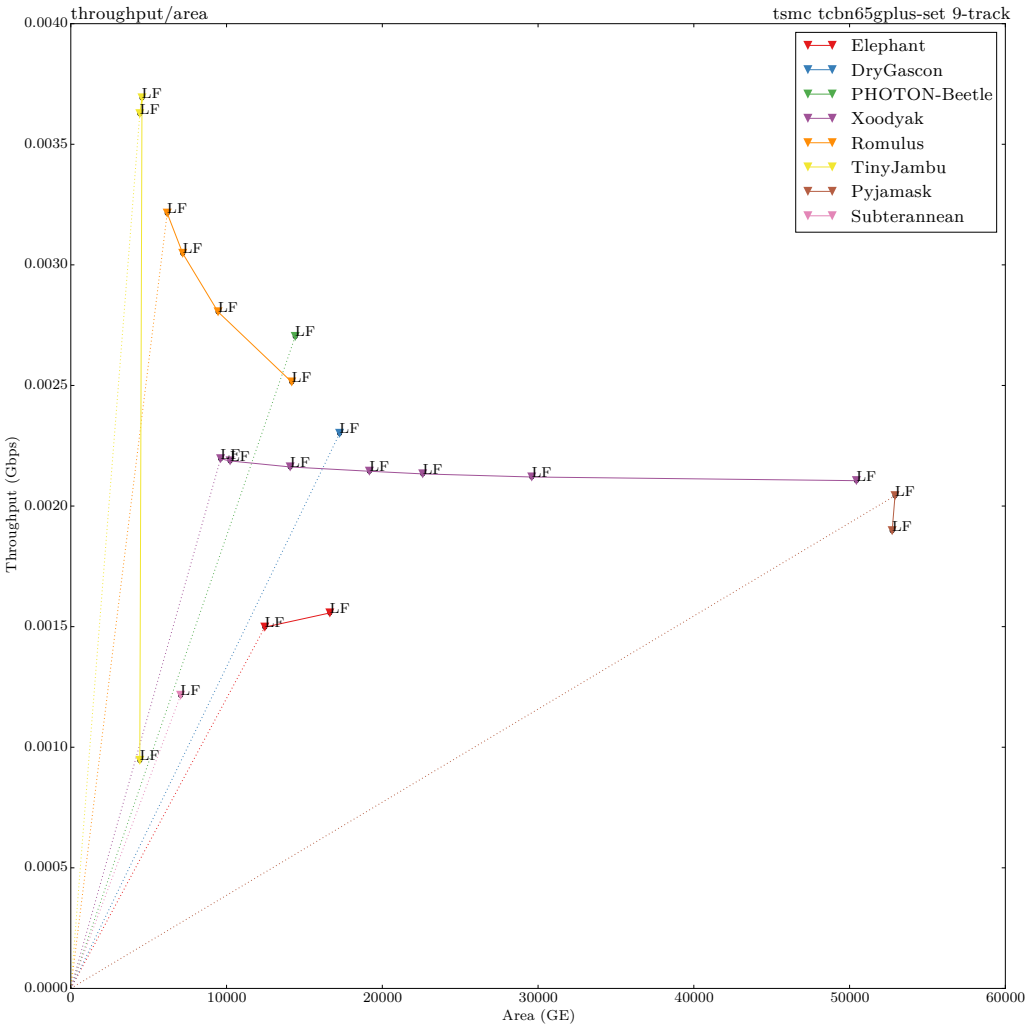


**Figure 35:** 3 Mbps: Throughput vs. Area for $|A| = |M| = 64$ bytes.

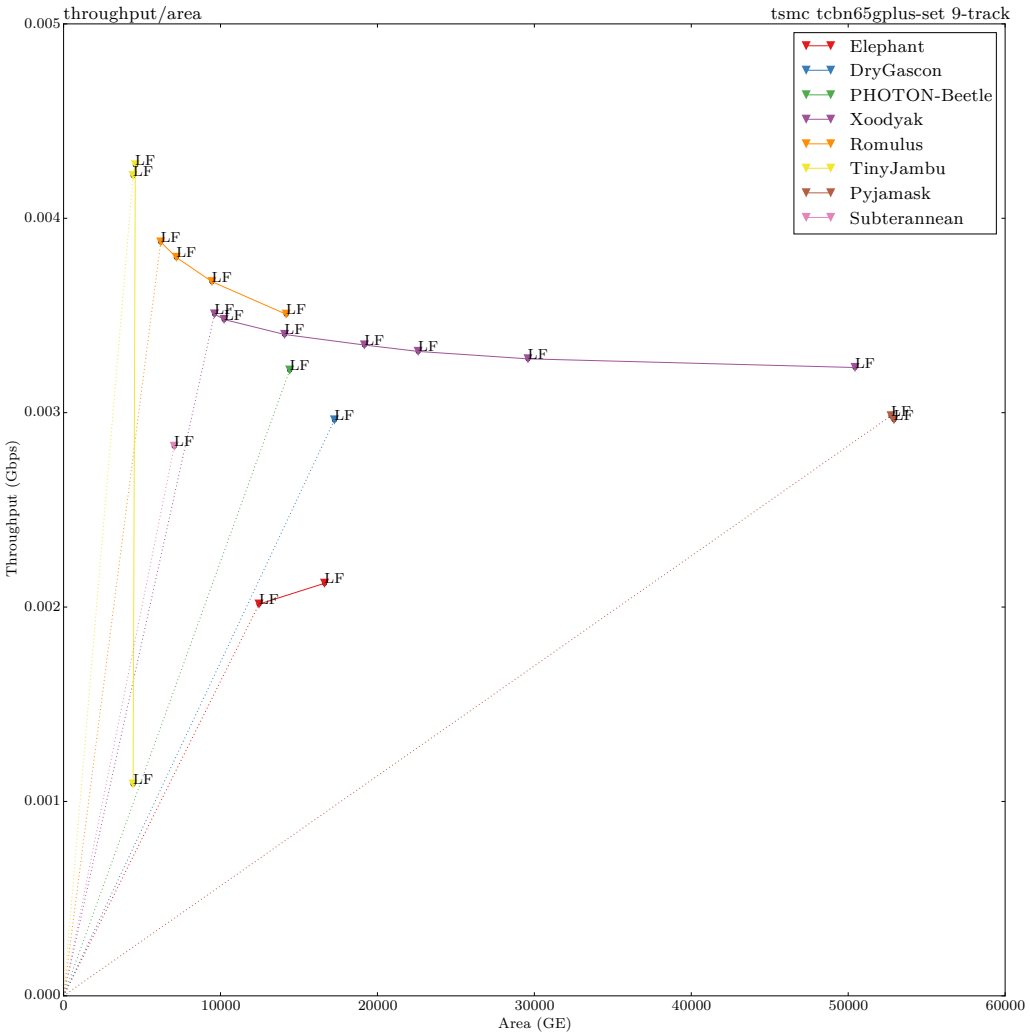


**Figure 36:** 3 Mbps: Throughput vs. Area for |A| = |M| = 1536 bytes.

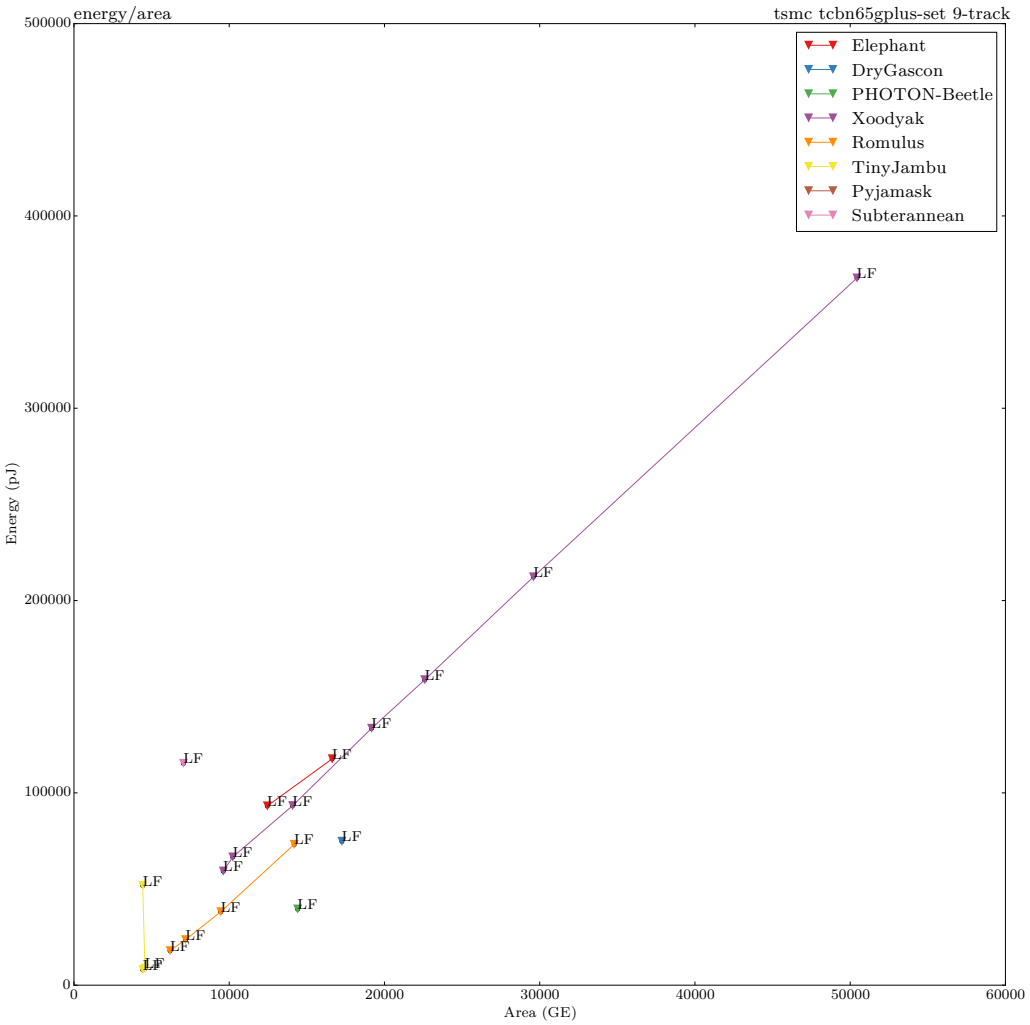


**Figure 37:** 3 Mbps: Energy vs. Area for $|A| = 16$ bytes.

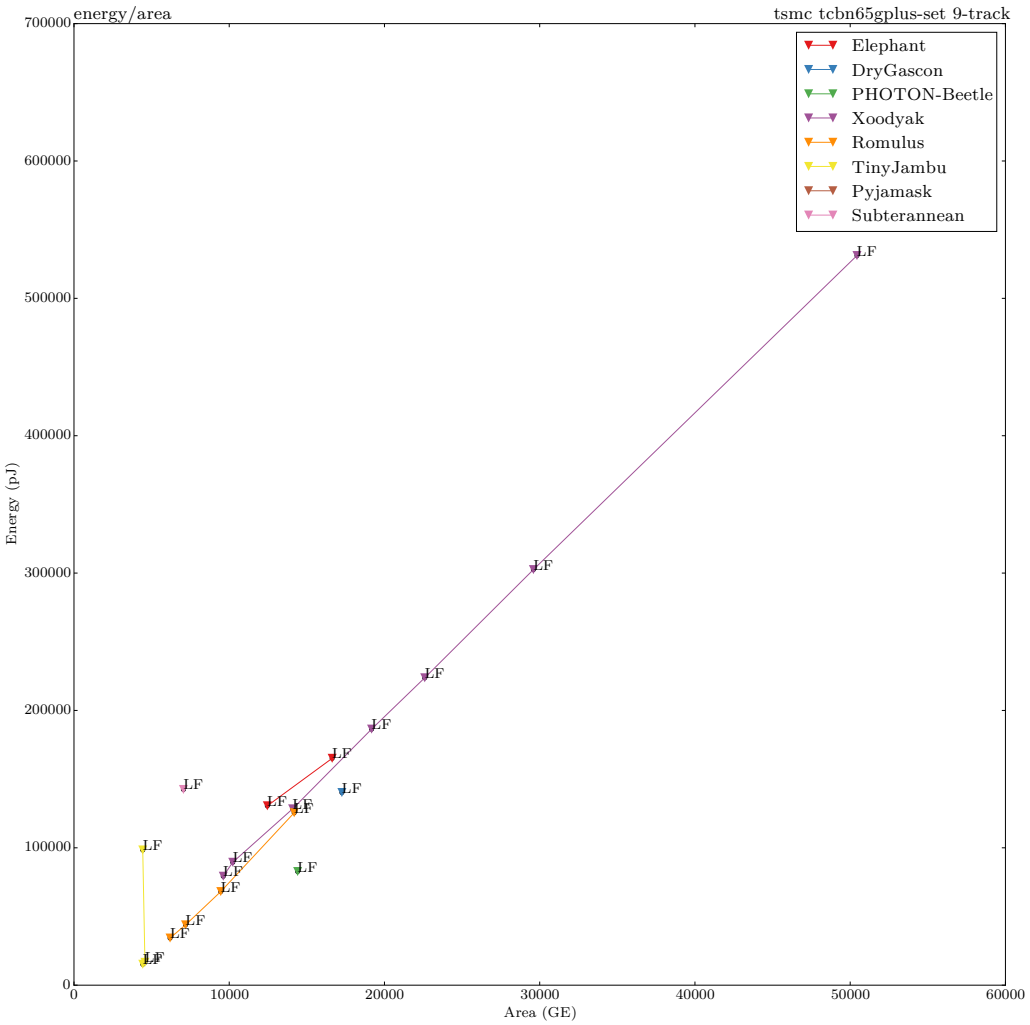


**Figure 38:** 3 Mbps: Energy vs. Area for |A| = 64 bytes.

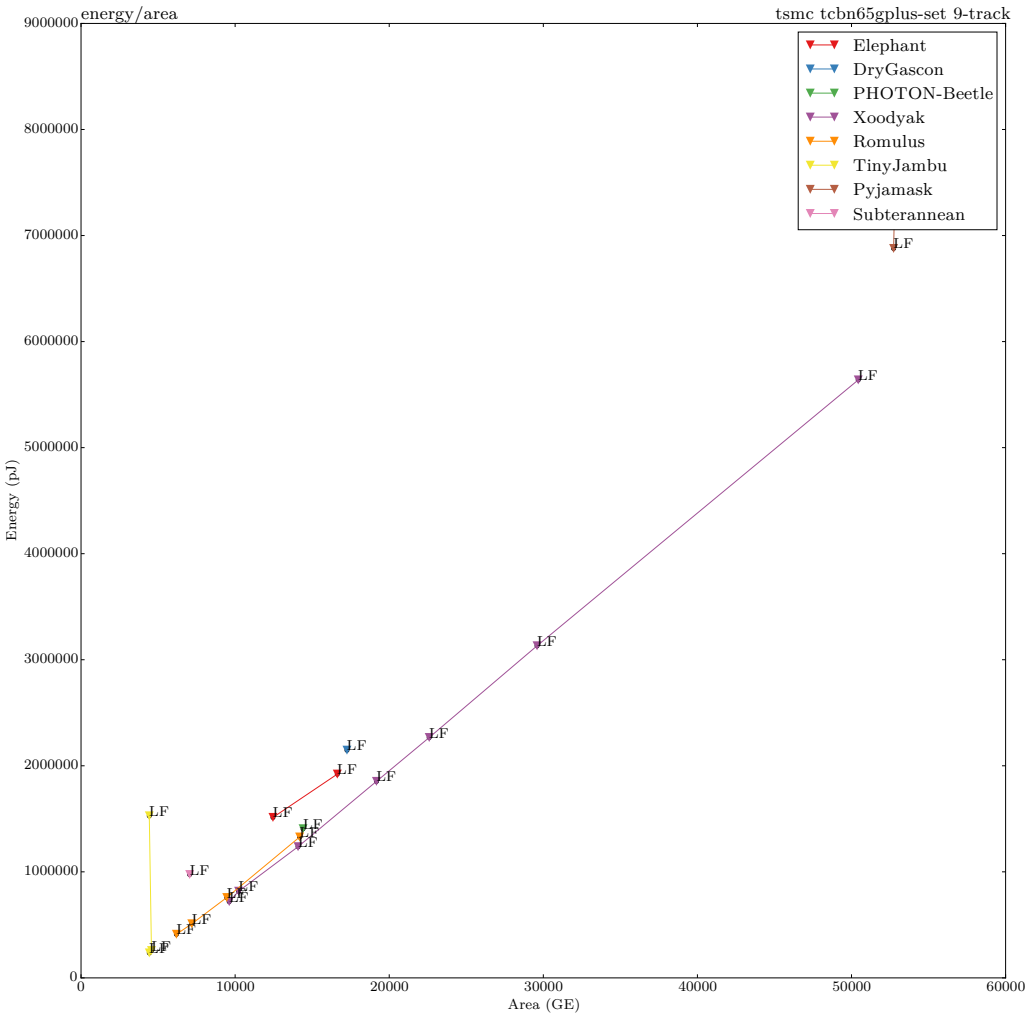


**Figure 39:** 3 Mbps: Energy vs. Area for |A| = 1536 bytes.

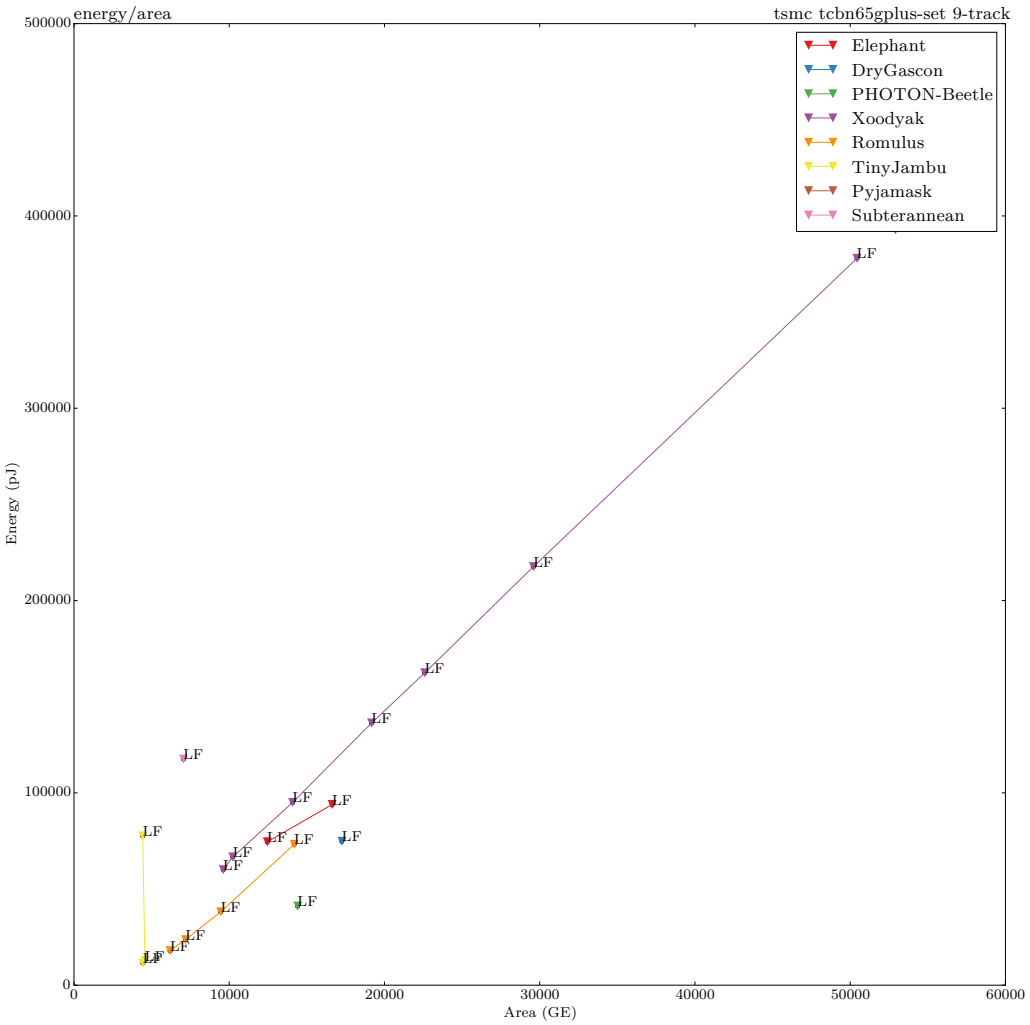


**Figure 40:** 3 Mbps: Energy vs. Area for $|M| = 16$ bytes.

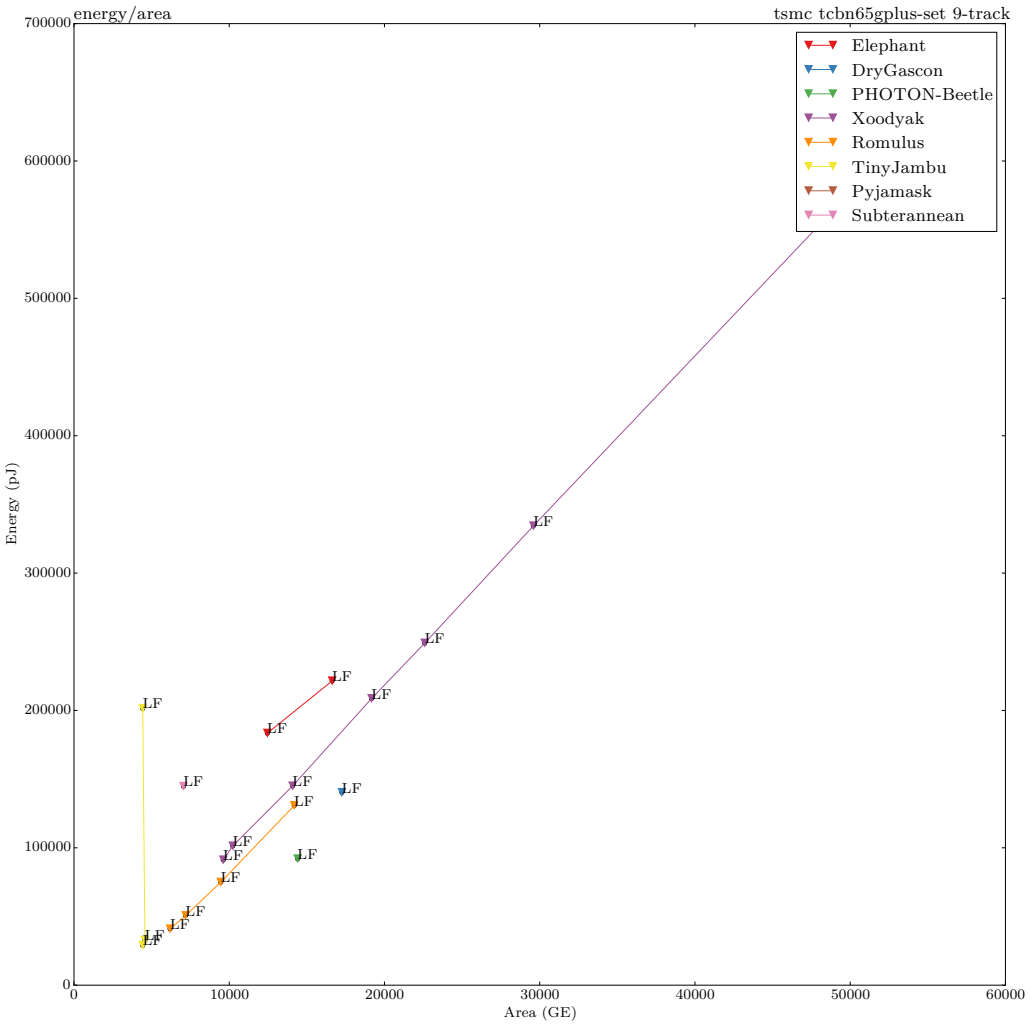


**Figure 41:** 3 Mbps: Energy vs. Area for $|M| = 64$ bytes.

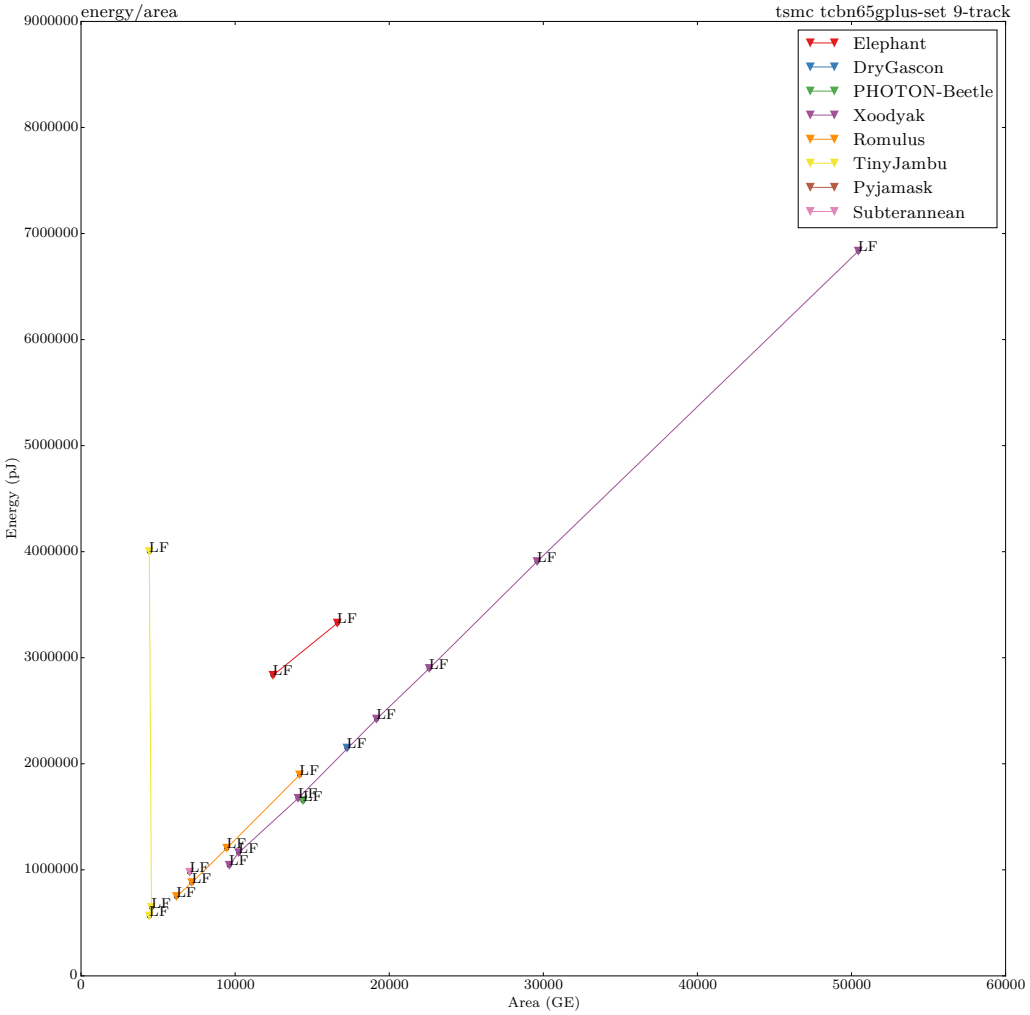


**Figure 42:** 3 Mbps: Energy vs. Area for |M| = 1536 bytes.

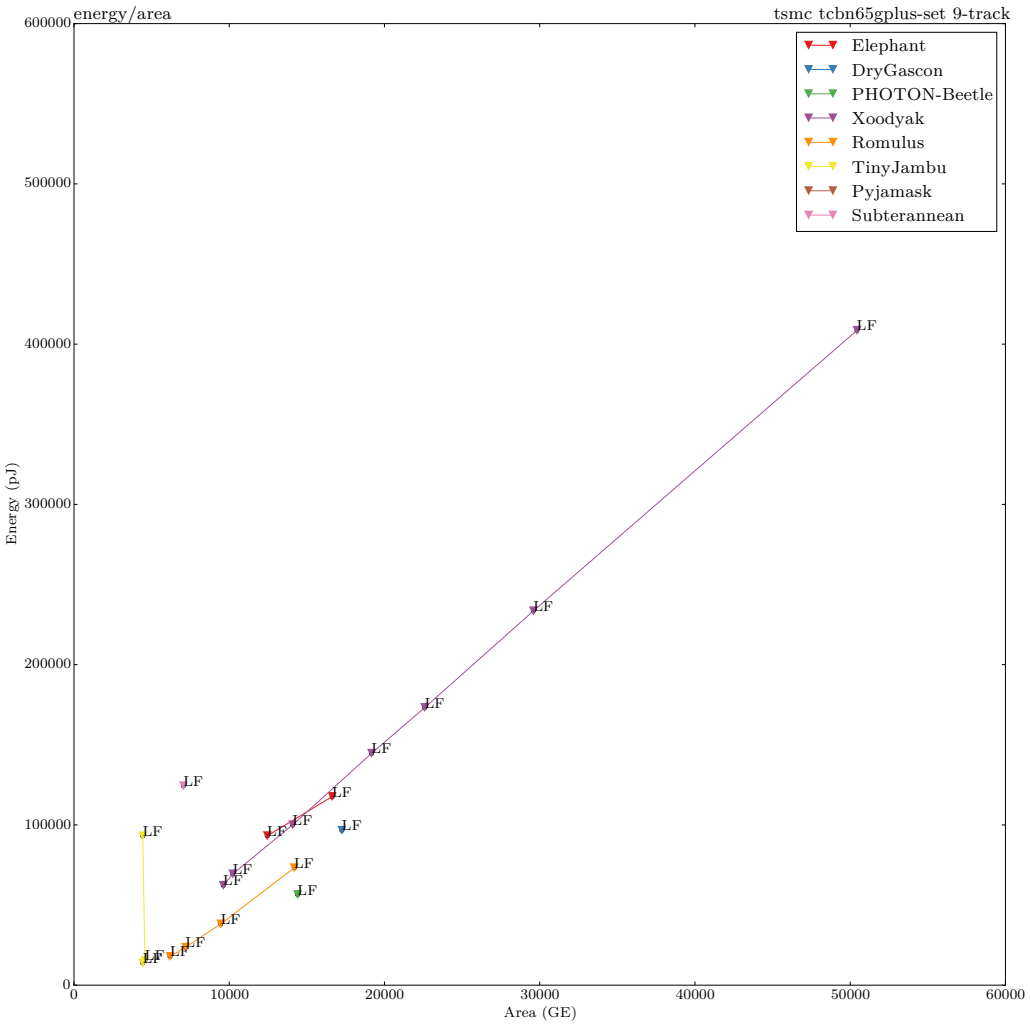


**Figure 43:** 3 Mbps: Energy vs. Area for $|A| = |M| = 16$ bytes.

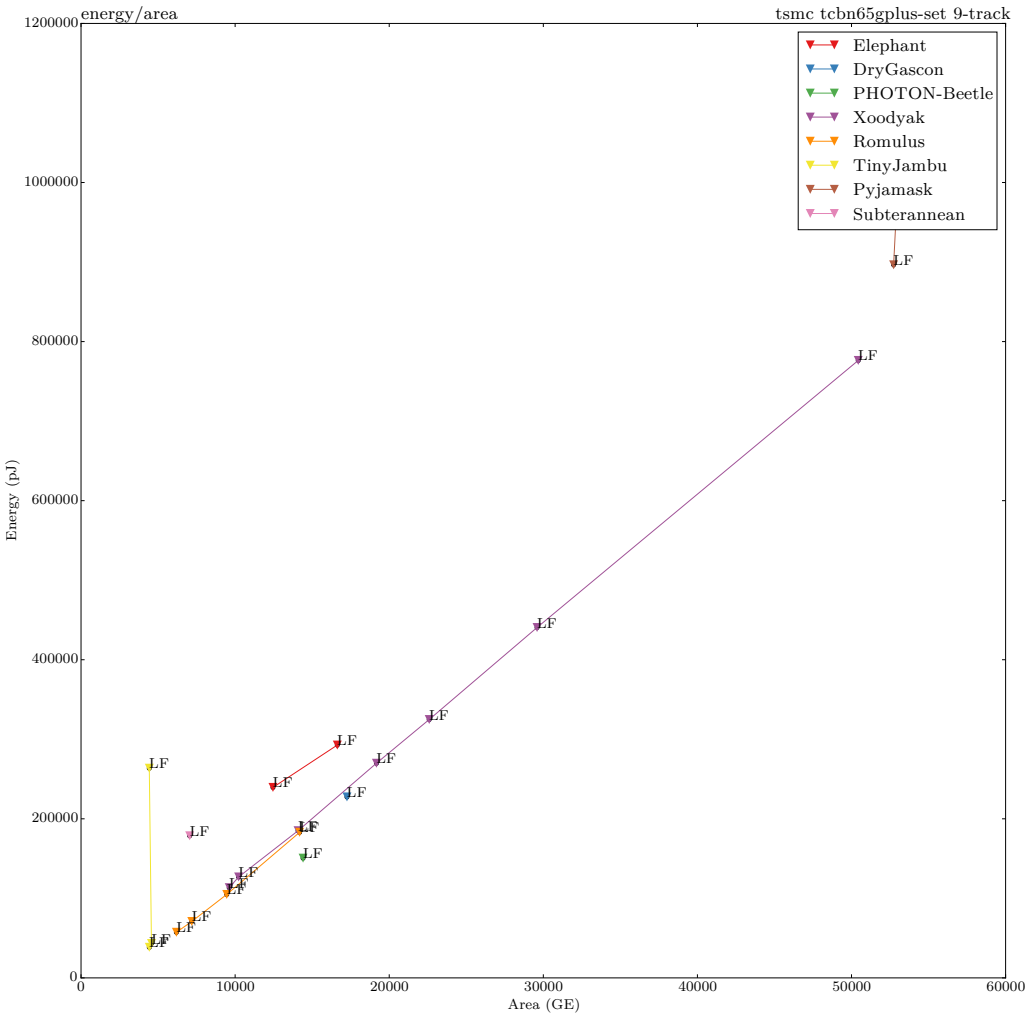


**Figure 44:** 3 Mbps: Energy vs. Area for $|A| = |M| = 64$ bytes.

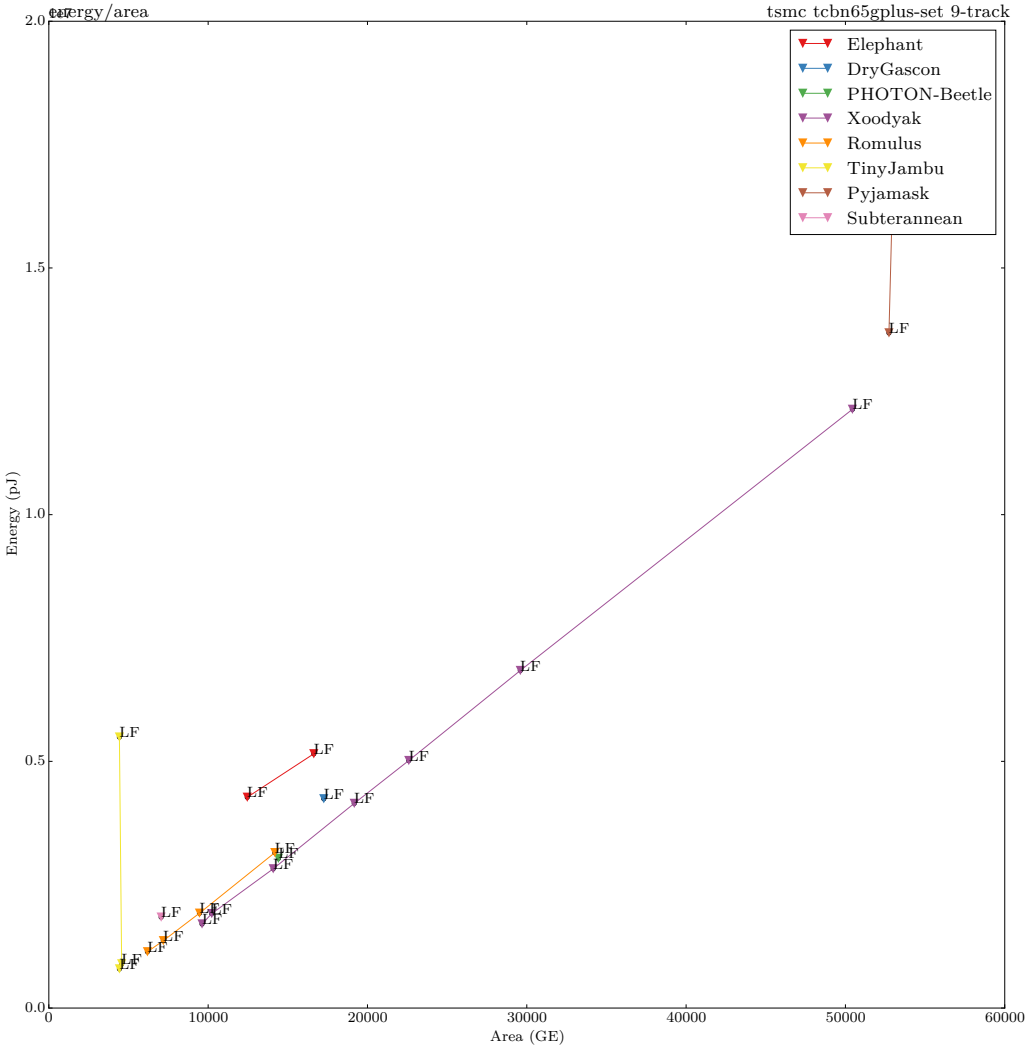


**Figure 45:** 3 Mbps: Energy vs. Area for $|A| = |M| = 1536$ bytes.

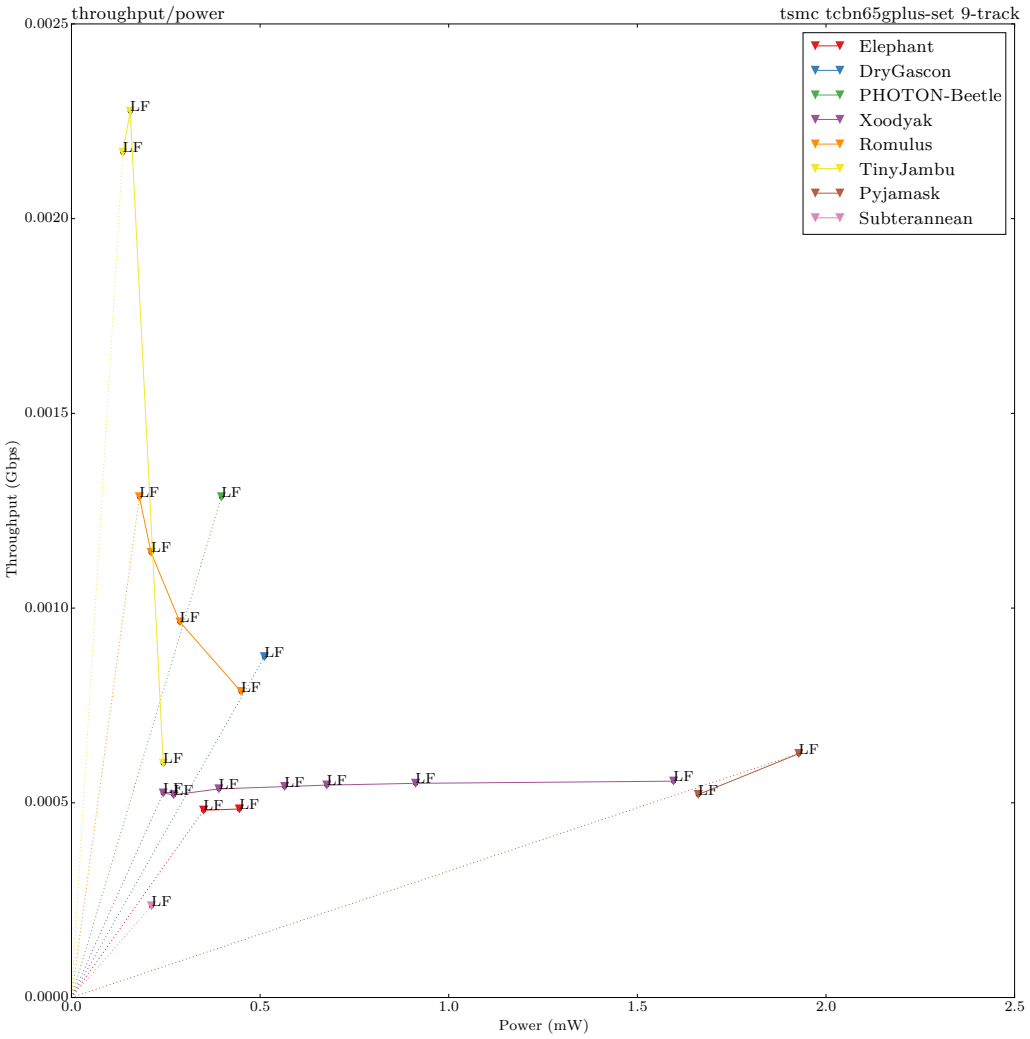


**Figure 46:** 3 Mbps: Throughput vs. Power for $|A| = 16$ bytes.

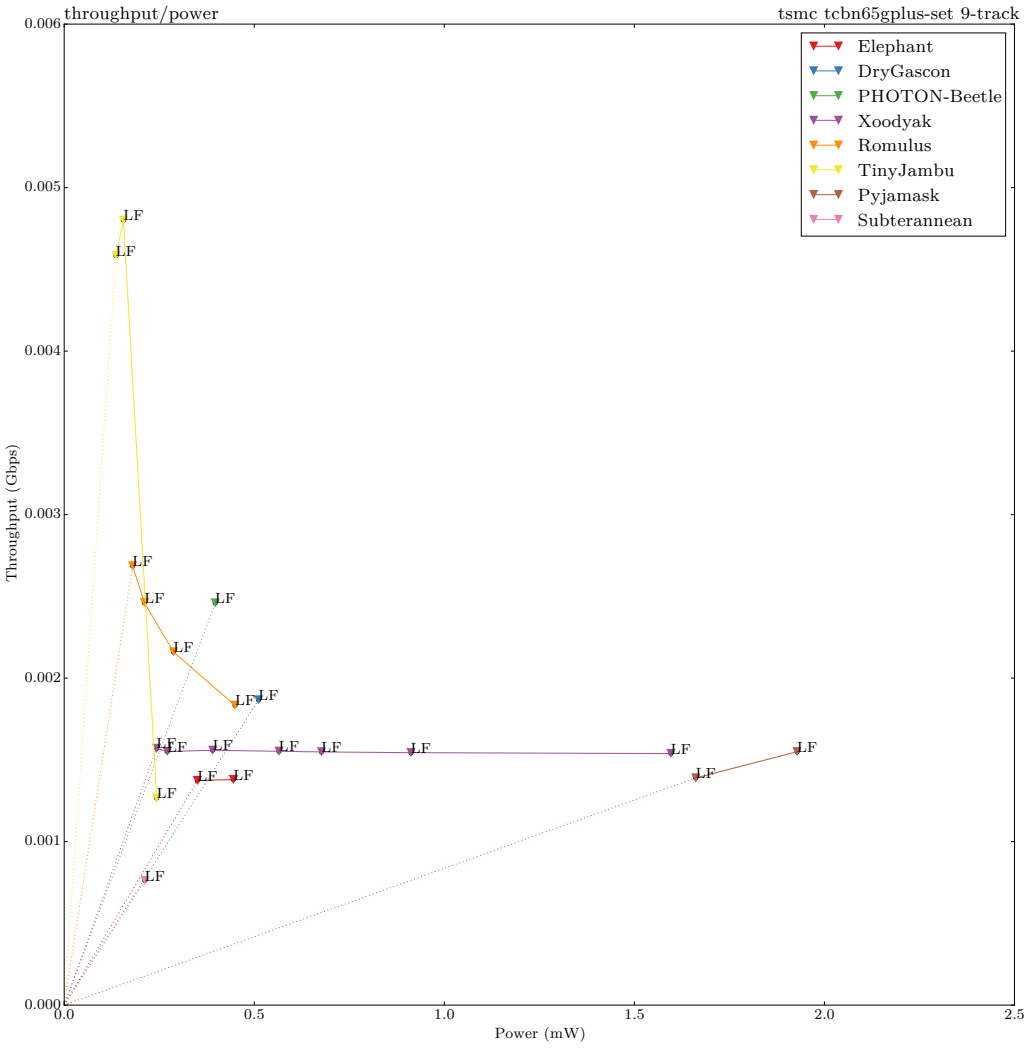


**Figure 47:** 3 Mbps: Throughput vs. Power for |A| = 64 bytes.

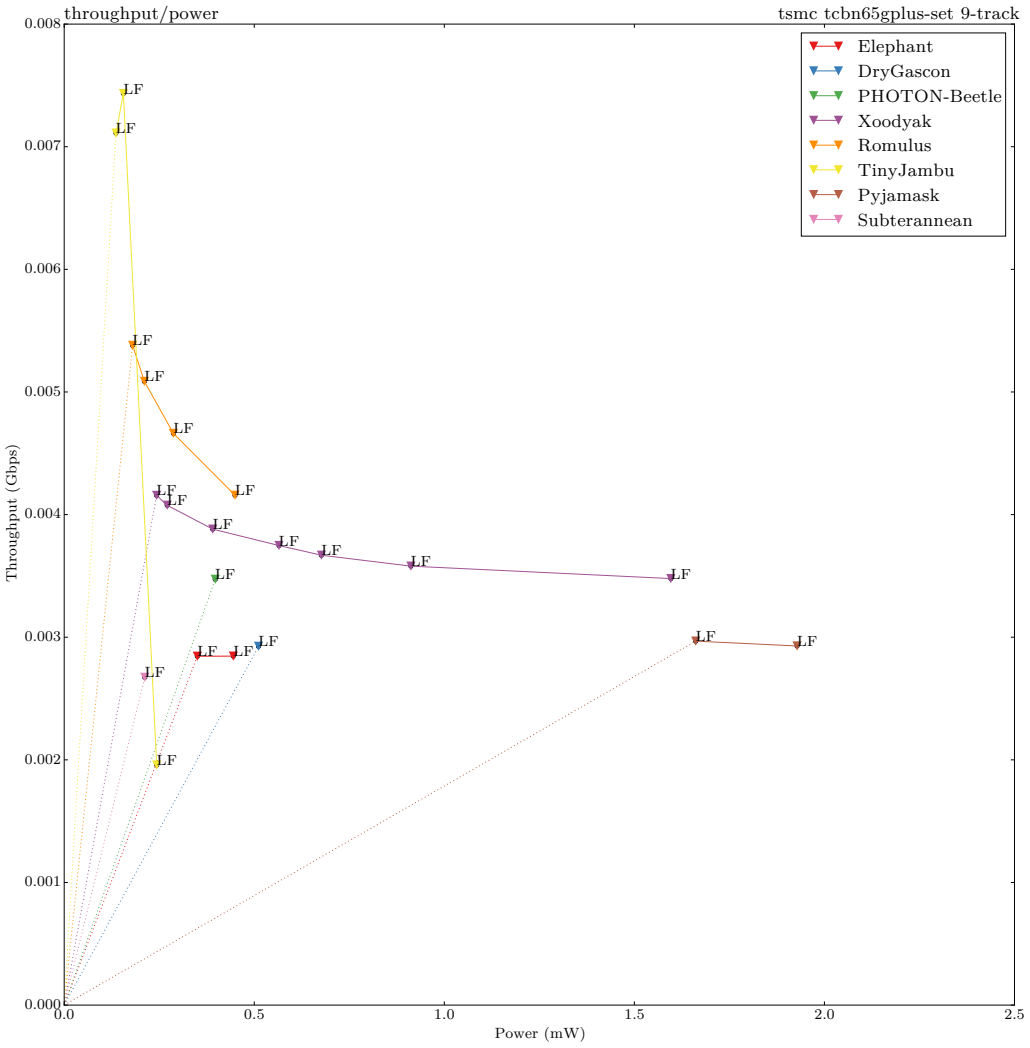


**Figure 48:** 3 Mbps: Throughput vs. Power for |A| = 1536 bytes.

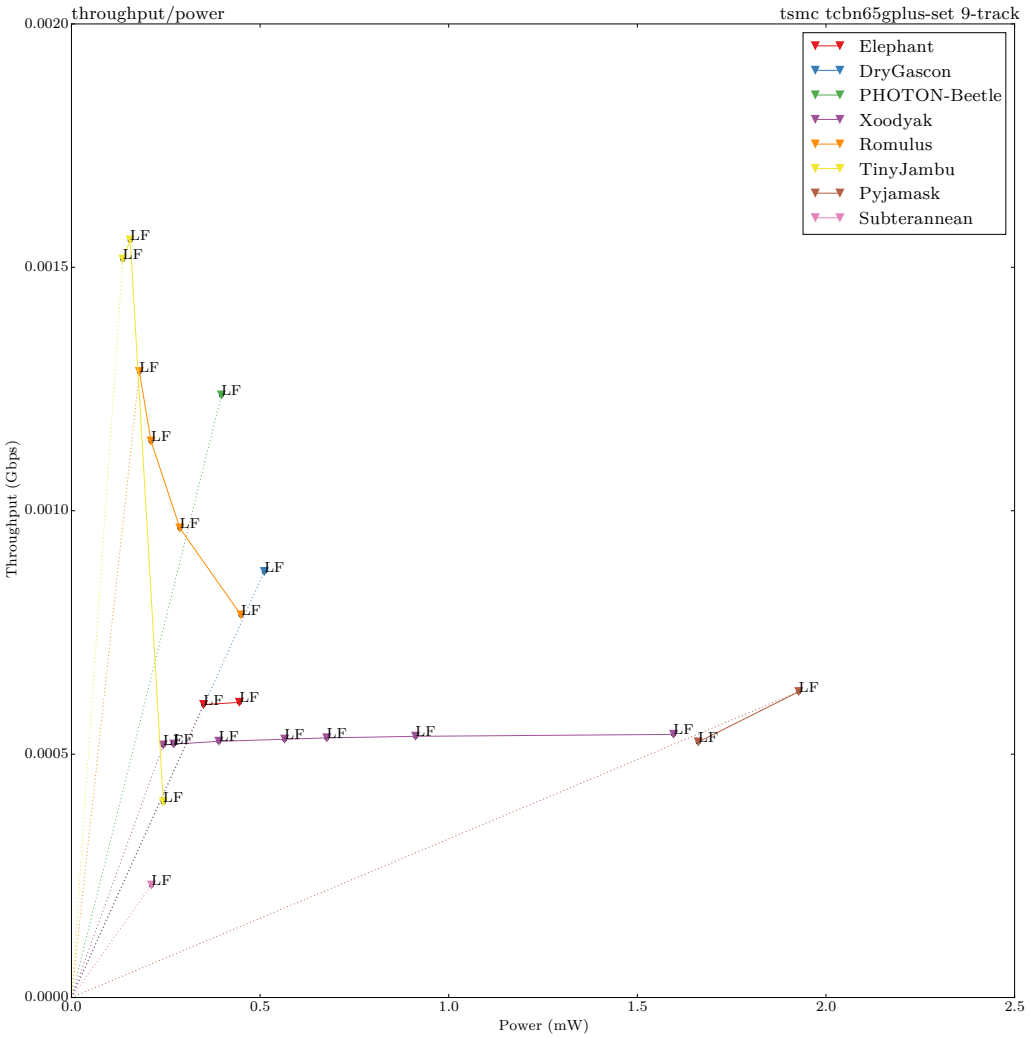


**Figure 49:** 3 Mbps: Throughput vs. Power for |M| = 16 bytes.

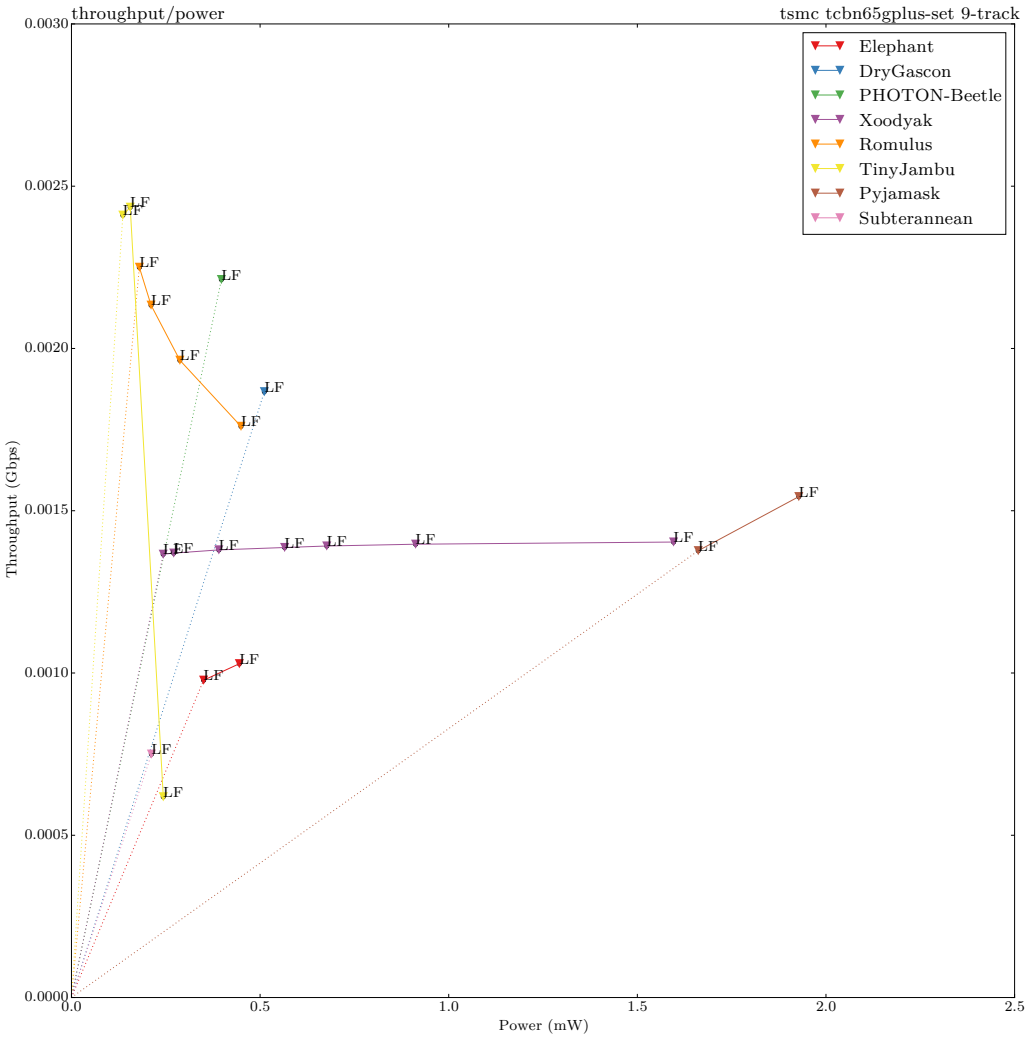


**Figure 50:** 3 Mbps: Throughput vs. Power for |M| = 64 bytes.

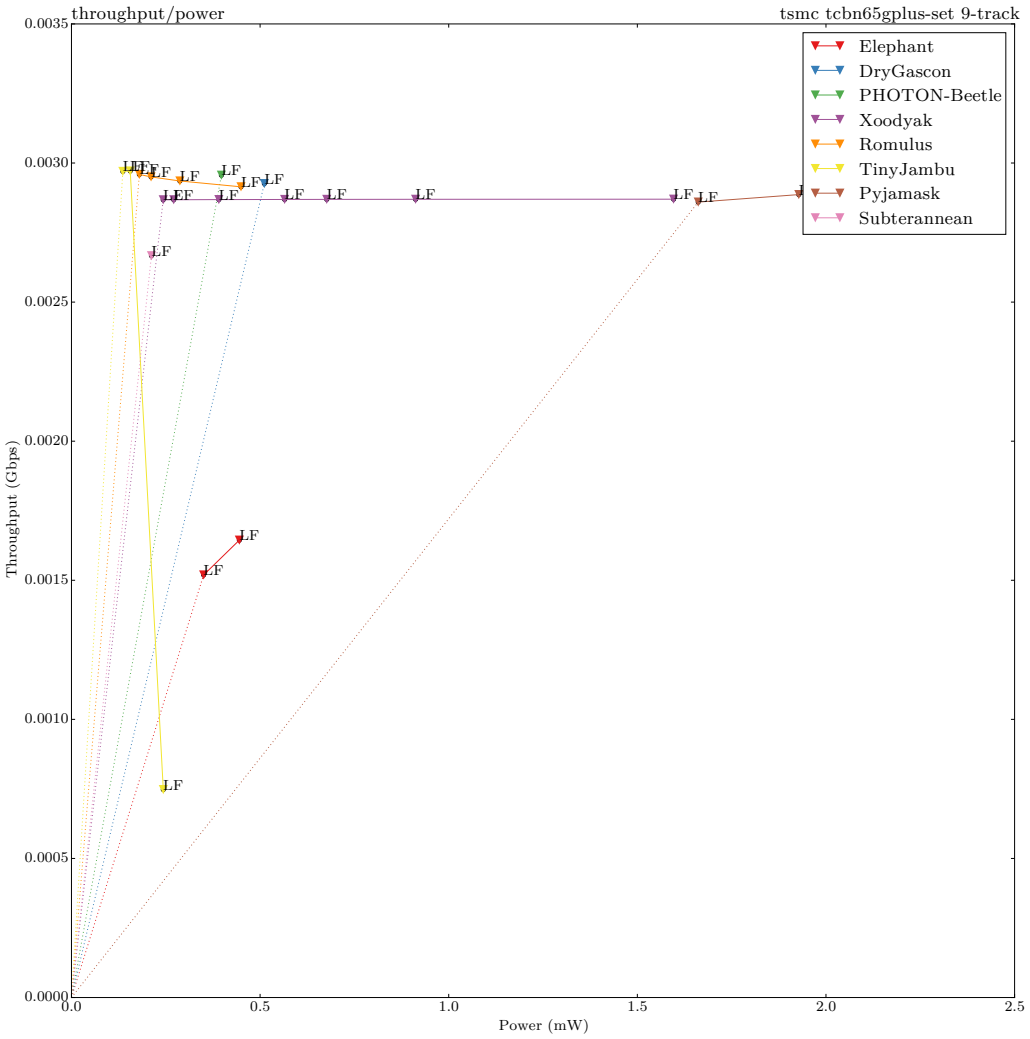


**Figure 51:** 3 Mbps: Throughput vs. Power for |M| = 1536 bytes.

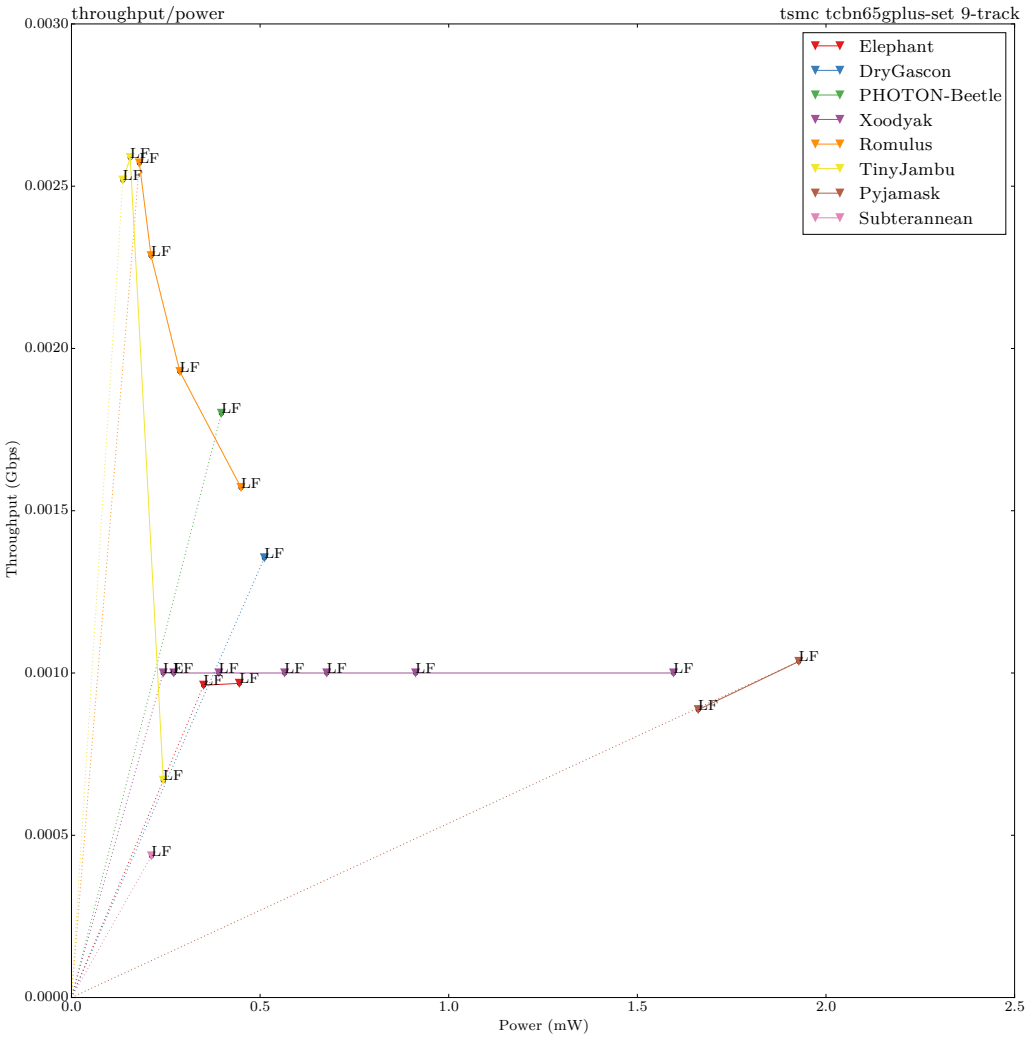


**Figure 52:** 3 Mbps: Throughput vs. Power for |A| = |M| = 16 bytes.

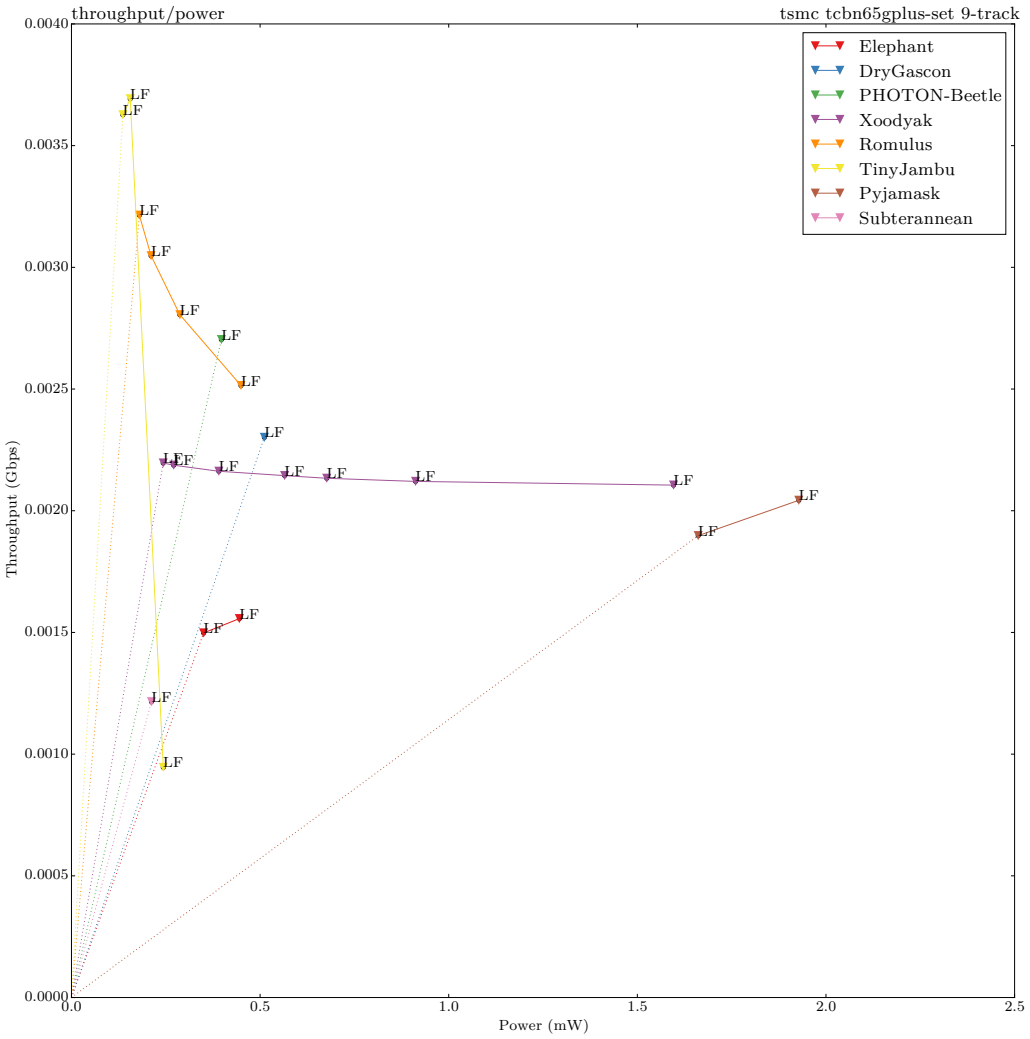


**Figure 53:** 3 Mbps: Throughput vs. Power for $|A| = |M| = 64$ bytes.

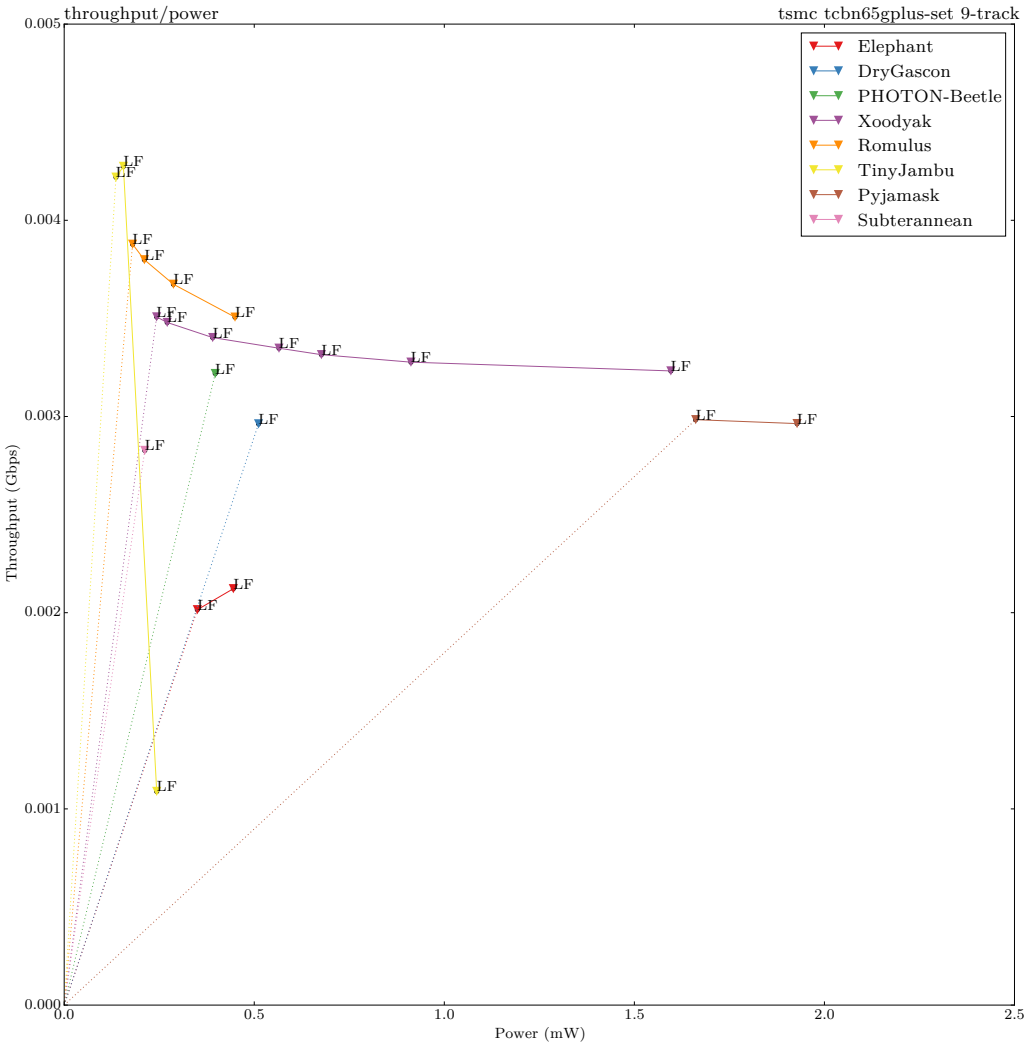


**Figure 54:** 3 Mbps: Throughput vs. Power for |A| = |M| = 1536 bytes.

# 6 Rankings

Given the previous results we rank the right candidates considered according to their best result in different metrics, with 1 being the best and 8 being the worst. This ranking is given in Table 10. The combined metrics are considered for the worst case scenario of 1536 bytes of both plaintext and associated data.

**Table 10:** Ranking of each algorithm against different criteria.

| Candidate | Throughput | | | Area | Power | Energy | | | Performance Efficiency | | | 3 Mbps | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 16 | 64 | 1536 | | | 16 | 64 | 1536 | Throughput/Area | Throughput/Power | Energy×Area | Throughput/Area | Throughput/Power | Energy×Area |
| DryGascon | 4 | 4 | 4 | 7 | 7 | 4 | 4 | 4 | 4 | 4 | 5 | 6 | 8 | 7 |
| Elephant | 6 | 6 | 6 | 5 | 5 | 6 | 6 | 6 | 7 | 6 | 7 | 7 | 7 | 6 |
| PHOTON-Beetle | 5 | 5 | 5 | 6 | 6 | 5 | 5 | 5 | 6 | 5 | 6 | 5 | 5 | 5 |
| Pyjamask | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 6 | 8 |
| Romulus | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 |
| Subterranean | 1 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 4 | 3 |
| TinyJambu | 7 | 7 | 7 | 1 | 1 | 7 | 7 | 7 | 5 | 7 | 4 | 1 | 1 | 1 |
| Xoodyak | 2 | 2 | 2 | 4 | 4 | 3 | 3 | 3 | 2 | 3 | 3 | 4 | 3 | 4 |

# 7 Conclusions

The results in this report give an idea about the ASIC performance of eight round 2 candidates for the NIST lightweight cryptography competition: DryGascon, Elephant, PHOTON-Beetle, Pyjamask, Romulus, Subterranean, TinyJambu and Xoodyak. We study different performance metrics and trade-offs, across two use cases: performance efficiency and Bluetooth. The results show that some algorithms behave differently in different use cases, while others maintain a somewhat uniform profile across different metrics. For example, the results show that Pyjamask is not suitable for unprotected hardware implementations, ranking last in all but one of the metrics, and in most cases with a big margin. However, it must be noted that Pyjamask was originally designed to target protected implementations and its performance in that domain is yet to be seen. Subterranean ranks first in most metrics, while it drops in terms of power consumption and low data rate. Subterranean, Romulus and Xoodyak rank consistently the top 4 in every metric. TinyJambu ranks in the top when it comes to low area, low power and low data rates, while it is close to the bottom when it comes to performance efficiency and energy consumption.

**What does this mean?** The benchmarking of different candidates only measures the implementations submitted to it. Hence, it is not a definitive answer to the optimal performance and potential of every candidate as optimizations can be found. However, it measure the state-of-the-art of implementations. Designers are urged to find spots where their implementations are not optimal and enhance it accordingly.

**What is next?** For the remainder of round 2 of the competition, we plan to:

1. We expect more designers to submit their implementations of different candidates. Upon receiving such implementations, we will study them using the same flow and update our report accordingly.
2. If we receive any new implementations of the candidates already considered we will also include them.
3. We plan to add a power-oriented (rather than performance-oriented) optimization flow that may lead to other interesting insights.
4. We plan to publish a similar report round 2 hash function candidates.
5. Potentially, we will include more libraries to our results, besides the TSMC 65nm library.

Afterwards, we plan to study protected implementations in round 3 of the competition.

# Acknowledgment

We would like to thank Kris Gaj and Pedro Maat Costa Massolino for their inputs and insights on the ASIC benchmarking. We would like to also thank the designers of the LWC Hardware API for their efforts that made this benchmarking process possible.

# References

[BCL18] A. Burg, A. Chattopadhyay, and K. Lam. Wireless communication and security issues for cyber–physical systems and the internet-of-things. *Proceedings of the IEEE*, 106(1):38–60, 2018. https://ieeexplore.ieee.org/abstract/document/8232533.

[BL20] Daniel J. Bernstein and Tanja Lange. eBACS: ECRYPT Benchmarking of Cryptographic Systems. https://bench.cr.yp.to/supercop.html, 2020.

[CAE20] CAESAR Competition. CAESAR submissions. https://competitions.cr.yp.to/caesar-submissions.html, 2020.

[KDT+19] Jens-Peter Kaps, William Diehl, Michael Tempelmeier, Ekawat Homsirikamol, and Kris Gaj. Hardware API for Lightweight Cryptography. 2019.

[KHYKC17] Sachin Kumar, Jawad Haj-Yihia, Mustafa Khairallah, and Anupam Chattopadhyay. A Comprehensive Performance Analysis of Hardware Implementations of CAESAR Candidates. *IACR Cryptology ePrint Archive, Report 2017/1261*, 2017. https://eprint.iacr.org/2017/1261.pdf.

[MHN+20] Kamyar Mohajerani, Richard Haeussler, Rishub Nagpal, Farnoud Farahmand, Abubakr Abdulgadir, Jens-Peter Kaps, and Kris Gaj. Fpga benchmarking of round 2 candidates in the nist lightweight cryptography standardization process: Methodology, metrics, tools, and results. Cryptology ePrint Archive, Report 2020/1207, 2020. https://eprint.iacr.org/2020/1207.

[NIS18] NIST. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process, 2018. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf.

[SR20] Jürgen Mottok Sebastian Renner, Enrico Pozzobon. NIST LWC Software Performance Benchmarks on Microcontrollers. https://lwc.las3.de/, 2020.