# SCA Evaluation & Benchmarking of Finalists in the NIST Lightweight Cryptography Standardization Process


**Jens-Peter Kaps**
**& Kris Gaj**



GEORGE MASON UNIVERSITY

# GMU CERG Members & Affiliates Supporting LWC

## SCA Evaluations

**Bakry** Abdulgadir

PQSecure™

**Eddie** Ferrufino

GMU MS Student

## Benchmarking

**Kamyar** Mohajerani

**Luke** Beckwith

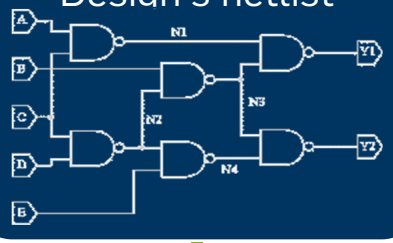
GMU PhD Students

# Hardware Implementations

# Development of Protected Hardware Implementations

Unprotected Implementation (behavioral verilog)

Design's netlist




**Settings**
- Level of protection (order)?
- Optimize for area or speed?

M: **Manual Design**

- Not straightforward
- Based on experience
- Algorithmic level
- Prone to errors/defeats

Side-Channel Protected Implementation (behavioral verilog)

A: **AGEMA**

- Identifies gates/modules to be secured
- Replaces with equivalent variants
- Adjusts the control logic

Side-Channel Protected Implementation (netlist)

- Free of heuristics
- Based on proofs
- Free of engineering's failures
- Open-source (GitHub)

Source: David Knichel, Amir Moradi, Nicolai Müller and Pascal Sasdrich, "Automated Generation of Masked Hardware (AGEMA)," CHES 2022, Sep. 2022

# Summary of Hardware Implementations

| Finalist | Unprotected | Order 1 | Order 2 | Order 3 |
|---|---|---|---|---|
| **Ascon** | Graz<br>VT<br>GMU (2) | M: Graz<br>A: Bochum (2) | M: Graz<br>A: Bochum (2) | A: Bochum (2) |
| **Elephant** | GMU | A: Bochum, M: GMU | A: Bochum | A: Bochum |
| **Grain-128AEAD** | GMU | | | |
| **GIFT-COFB** | VT<br>GMU | A: Bochum | A: Bochum | A: Bochum |
| **ISAP** | Graz | A: Bochum | A: Bochum | A: Bochum |
| **ISAP** | Graz (mode-level protection) | | | |
| **PHOTON-Beetle** | GMU | A: Bochum | A: Bochum | A: Bochum |
| **Romulus** | NTU | A: Bochum | A: Bochum | A: Bochum |
| **SPARKLE** | VT<br>GMU | A: Bochum | A: Bochum | A: Bochum |
| **TinyJAMBU** | GMU<br>TJ Team | A: Bochum, M: GMU | A: Bochum | A: Bochum |
| **Xoodyak** | XT Team<br>GMU (2) | A: Bochum, M: Tsinghua (2)<br>M: GMU | A: Bochum | A: Bochum |

# Mode-level Protection of ISAP
# **Differential Power Analysis (DPA)**

ISAP specifically designed to achieve inherent resistance to the DPA-based

- key recovery attacks
- plaintext recovery attacks
- tag recovery/message forgery attacks.

Unique session keys for encryption and authentication are derived from the long-term key and the nonce.

The two-pass construction prevents an attack involving querying the decryption with a constant nonce and varying ciphertexts.
The decryption starts only after the authenticity of the ciphertext and nonce was fully verified.

Message forgery attacks prevented by utilizing additional permutation calls before the tag verification.

# ISAP
# **Simple Power Analysis (SPA)**

ISAP does not provide inherent resistance against SPA.

SPA challenging for hardware implementations that calculate permutation concurrently on all bits of the state.

SPA challenging for software implementations on 32-bit and 64-bit processors.

# Side-Channel Security Evaluation Labs

# Side-Channel Security Evaluation Labs for Hardware


| Lab | Evaluated Implementations (Design Group) | Target FPGA Family | Target Boards | Leakage Assessment Methods | Attacks |
|---|---|---|---|---|---|
| IAIK, Graz University of Technology, Austria | Ascon, Elephant, GIFT-COFB, Romulus, Xoodyak (Bochum) | Artix-7 | NewAE CW305 | t-test | |
| Cryptology and Computer Security Laboratory, Shanghai Jiao Tong University, China | Ascon, GIFT-COFB, ISAP, Romulus (Bochum)<br>ISAP (Graz) | Kintex-7 | SAKURA-X | t-test, $\chi^2$<br>t-test, $\chi^2$<br>t-test, $\chi^2$<br>t-test, $\chi^2$<br>-<br>- | CPA<br>CPA<br>CPA<br>CPA, template<br>CPA<br>- |

# Side-Channel Security Evaluation Labs for Hardware


| Lab | Evaluated Implementations (Design Group) | Target FPGA Family | Target Boards | Leakage Assessment Methods | Attacks |
|---|---|---|---|---|---|
| Hardware Security and Cryptographic Processor Lab,Tsinghua University, China | Xoodyak (Bochum)<br>TinyJAMBU (GMU)<br>Ascon (Graz) | Kintex-7<br>Spartan-6<br>Spartan-6 | SAKURA-X<br>SAKURA-G<br>SAKURA-G | t-test<br>t-test<br>t-test | |
| Secure-IC, France | Xoodyak (GMU) | Artix-7 | SAKURA-G | t-test according to ISO/IEC 17825:2016 | |
| CERG, George Mason University, USA | Ascon, Elephant, PHOTON-Beetle, TinyJAMBU, Xoodyak (Bochum) | Artix-7 | NewAE CW305 | t-test | |

# Side-Channel Security Evaluation Labs for Software


| Lab | Evaluated Implementations (Design Group) | Target Processor | Target Device | Leakage Assessment Methods | Attacks |
|---|---|---|---|---|---|
| Cryptology and Computer Security Laboratory, Shanghai Jiao Tong University, China | Romulus (A.A.)<br>ISAP (Graz)<br>GIFT-COFB (A.A.)<br>Ascon (Graz) | ARM Cortex-M4F | STM32 F303RCT6 | t-test, $\chi^2$, DL-LA<br>-<br>t-test, $\chi^2$<br>t-test, $\chi^2$ | CPA, template<br>CPA<br>CPA<br>CPA |
| Hardware Security and Cryptographic Processor Lab, Tsinghua University, China | GIFT-COFB (A.A.)<br>Romulus (A.A.) | ARM Cortex-M4F | STM32 F303 | t-test | |
| CESCA Lab, Radboud University, Netherlands | Ascon (Graz)<br>Xoodyak (XKCP)<br>ISAP (Graz) | ARM Cortex-M4F | STM32 F407IGT6 | -<br>t-test<br>t-test | CPA<br>CPA |

# Evaluation Results

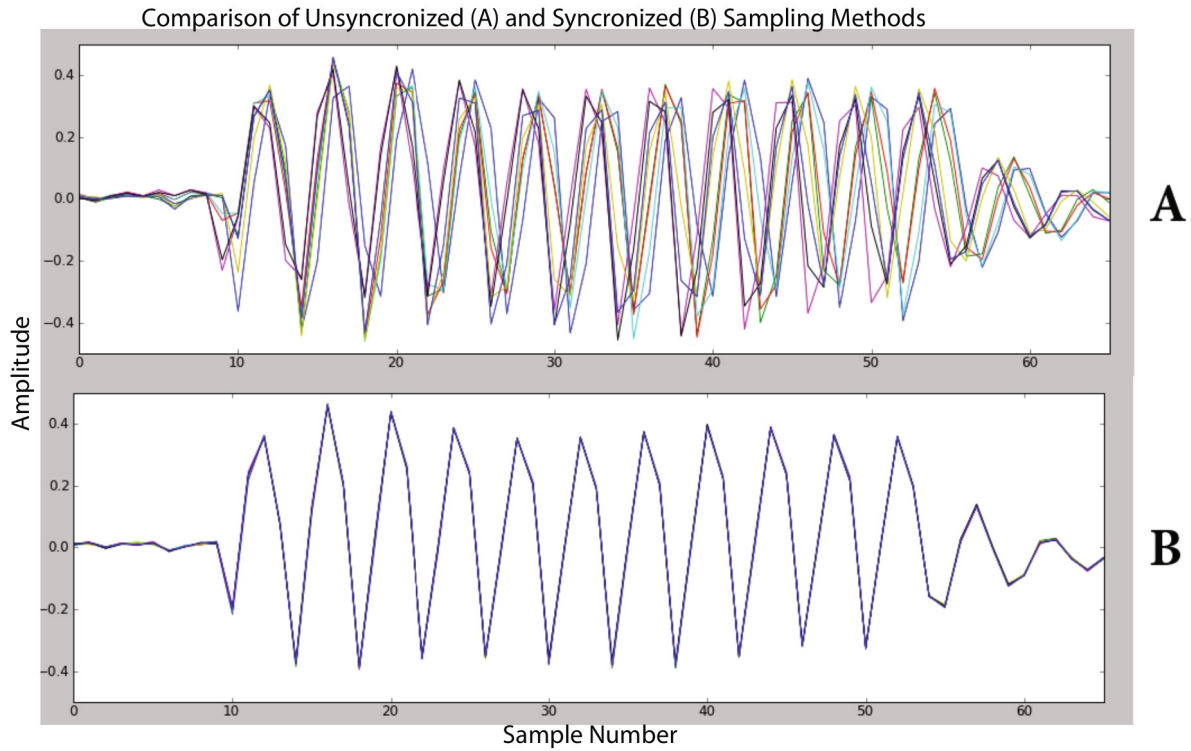
# T-Tests for Hardware Implementations

| Finalist | Protected Implementation of Order 1 |
|---|---|
| **Ascon** | Tsinghua: M: Graz: Passed for 10M traces<br>Graz: A: Bochum: Passed for 10M traces<br>Shanghai Jiao Tong: A: Bochum: Passed for 1M traces<br>GMU: A: Bochum: Failed for ~1.5M traces with clock synchronization |
| **Elephant** | Graz: A: Bochum: Passed for 10M traces<br>GMU: A: Bochum: Failed for ~21k traces with clock synchronization |
| **Grain-128AEAD** | Protected implementation not available |
| **GIFT-COFB** | Graz: A: Bochum: Passed for 10M traces<br>Shanghai Jiao Tong: A: Bochum: Passed for 1M traces |
| **ISAP** | T-test not applicable to the mode-protected implementation |
| **PHOTON-Beetle** | GMU: A: Bochum: Passed for 10M traces with clock synchronization |
| **Romulus** | Graz: A: Bochum: Passed for 10M traces<br>Shanghai Jiao Tong: A: Bochum: Failed for 1M traces |
| **SPARKLE** | Not tested by any lab |
| **TinyJAMBU** | Tsinghua: M: GMU: Passed for 10M traces<br>GMU: A: Bochum: Passed for 10M traces with clock synchronization |
| **Xoodyak** | Secure-IC: M: GMU: Passed for 100K traces<br>Graz: A: Bochum: Passed for 10M traces<br>Tsinghua: A: Bochum: Passed for 10M traces<br>GMU: A: Bochum: Failed for ~1.5M traces with clock synchronization |

# Synchronized Sampling

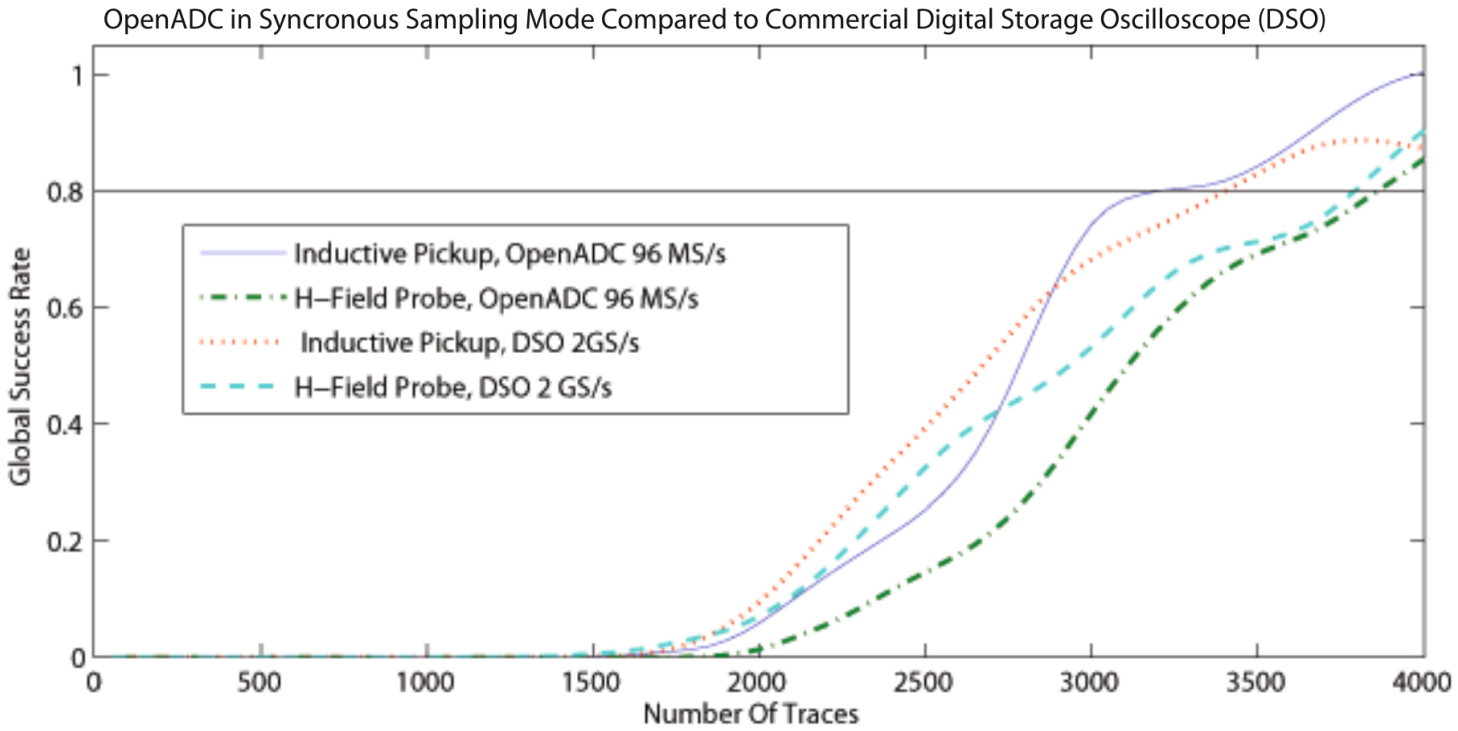
- By default, the internal sample clock of oscilloscope is not aligned with the clock of Design Under Test (DUT)
- Even though oscilloscope is triggered at DUT clock edge, there will be random jitter between traces
- In synchronized sampling, the sample clock is aligned with the clock of the DUT:
  - Reduces jitter
  - This is not possible for regular oscilloscopes.
    C. O'Flynn and C. Zhizhang. A case study of Side-Channel Analysis using Decoupling Capacitor Power Measurement with the OpenADC. In Proceedings of Workshop on Foundations and Practices of Security (FPS '13), volume 7743, pages 328–344. Springer, 2013.

# Unsynchronized vs. Synchronized




- 8 Power Traces
- A: 100 MHz sample clock, unsynchronized
- B: 96 MHz sample clock, synchronized

# Success Rates vs. Number of Traces




- OpenADC is synchronized, sampling at 96 MS/s
- Oscilloscope is unsynchronized, sampling at 2GS/s
- Both perform almost equally well

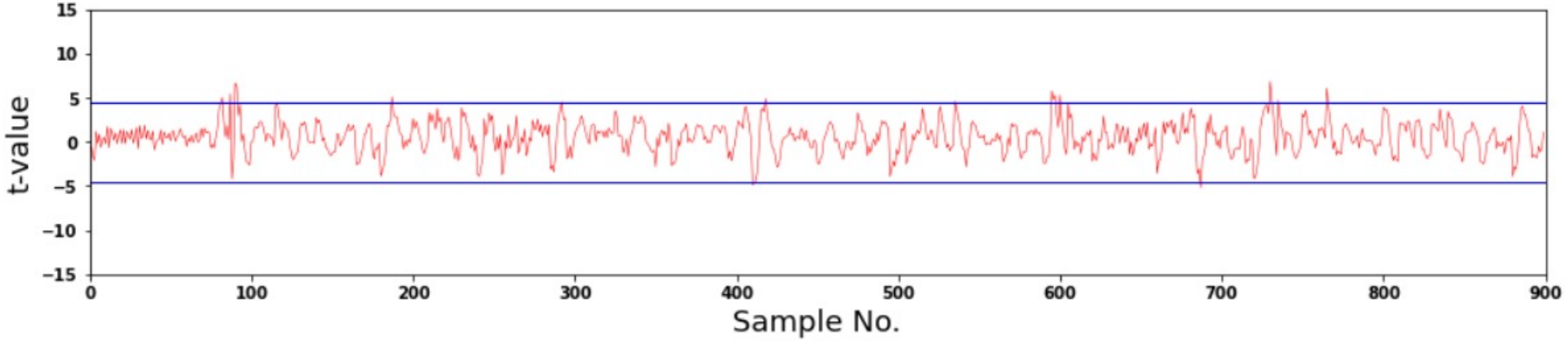
# Effect on Xoodyak Results

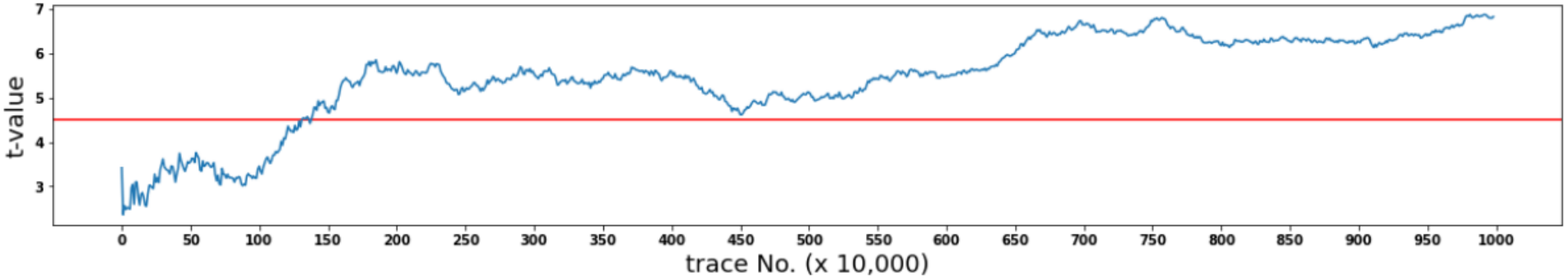
- Xoodyak Team's implementation of Xoodyak
- Protected using Bochum's AGEMA tool

| Lab | Target | Traces | Sample Rate | DUT Frequency | Resolu-tion | Result |
|---|---|---|---|---|---|---|
| Graz | Artix-7 | 10 Million | 22 MS/s | 1 MHz | 8 bit | Pass |
| Tsinghua | Kintex-7 | 10 Million | 1000 MS/s | 6 MHz | 8 bit | Pass |
| GMU | Artix-7 | 10 Million | 100 MS/s Synchron | 10 MHz | 10 bit | Fail |
| GMU | Artix-7 | 10 Million | 125 MS/s | 10 MHz | 8 bit | Pass |

# GMU Xoodyak Synchronized Results

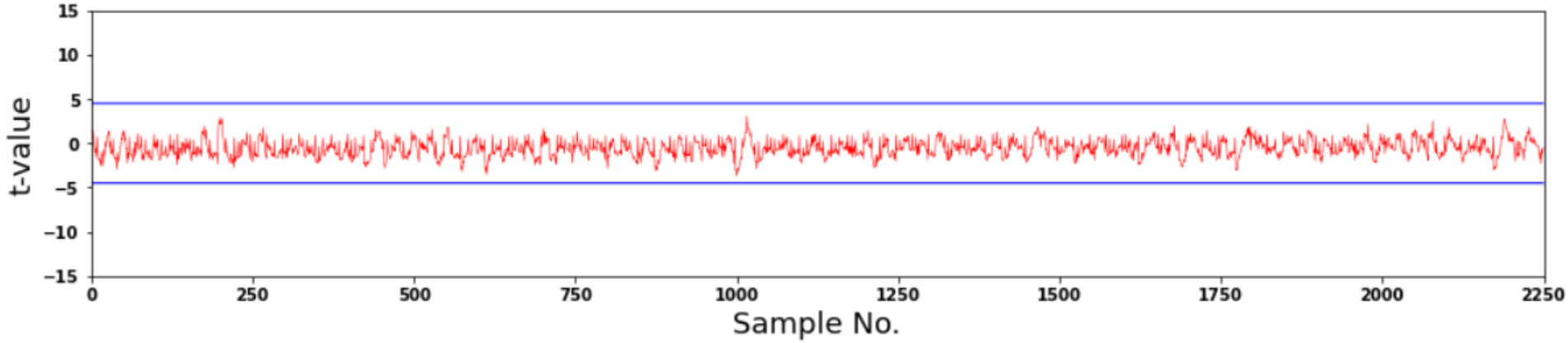
- TVLA Result: Fail

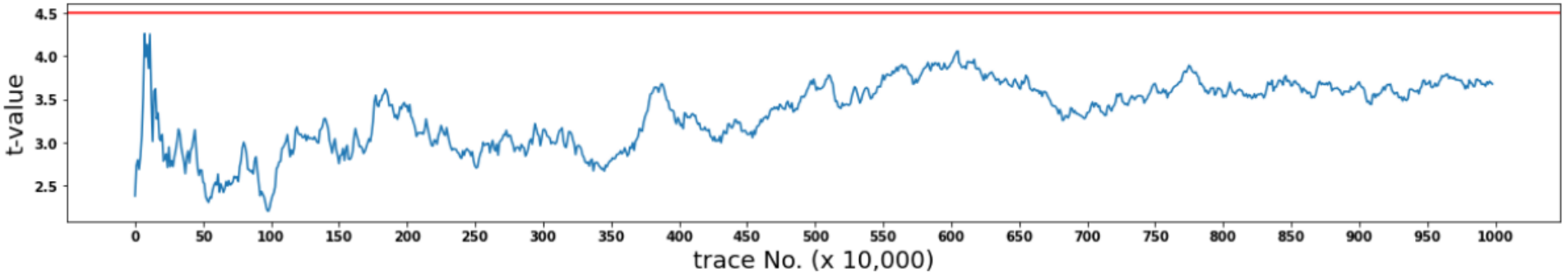


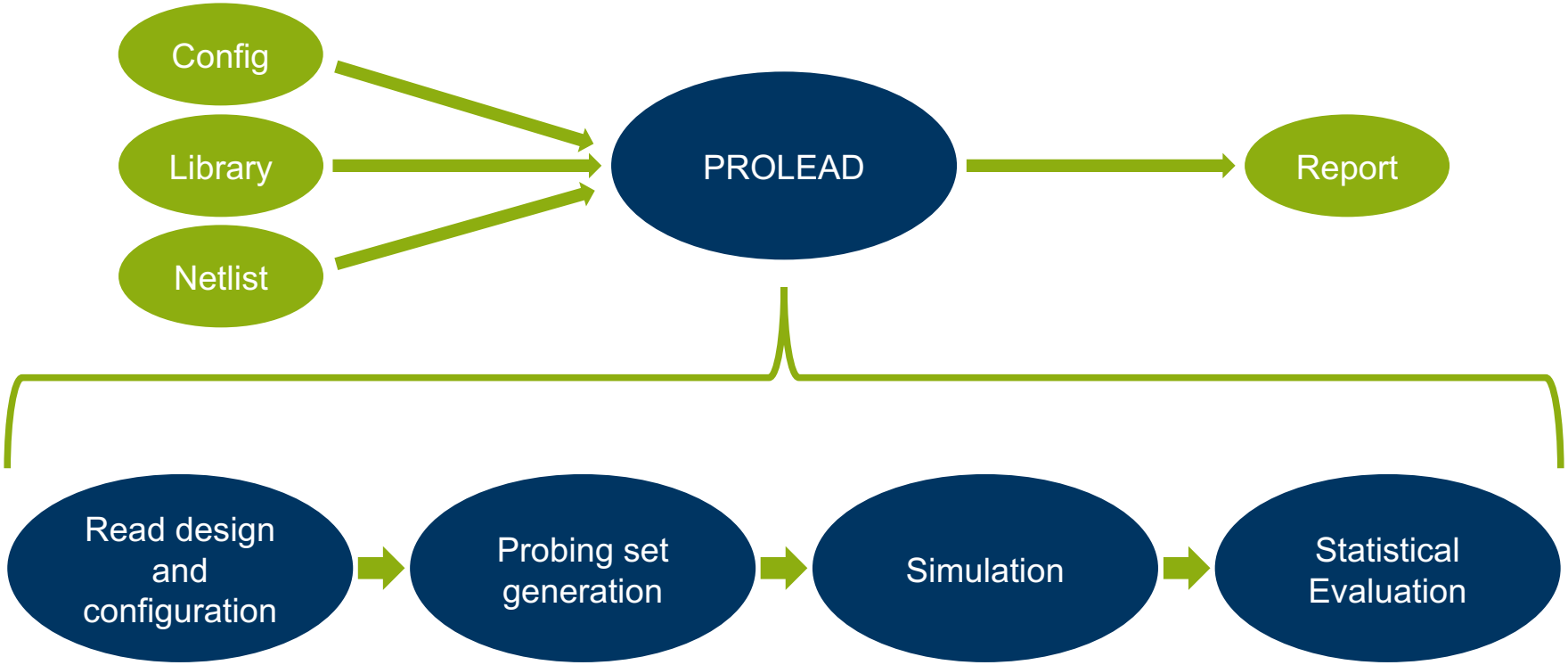
# GMU Xoodyak Unsynchronized Results

- TVLA Result: Pass



# PROLEAD
# A Probing-Based Hardware Leakage Detection Tool




Manually developed Ascon implementations of order 1 and 2 found to not leak outside of the tag checking procedure

Source: Nicolai Müller, Amir Moradi
"PROLEAD - A Probing-Based Hardware Leakage Detection Tool," CHES 2022, Sep. 2022

# Software Implementations

| Finalist | Order 1 | Order 2 |
|---|---|---|
| **Ascon** | M: Graz | M: Graz |
| **Elephant** | | |
| **Grain-128AEAD** | | |
| **GIFT-COFB** | M: Alexandre Adomnicai | |
| **ISAP** | M: Graz | |
| **PHOTON-Beetle** | | |
| **Romulus** | M: Alexandre Adomnicai | |
| **SPARKLE** | | |
| **TinyJAMBU** | | |
| **Xoodyak** | M: Tsinghua | |

# T-Tests & CPA for Software Implementations

| Finalist | Order 1 |
|---|---|
| **Ascon** | Shanghai Jiao Tong: M: Graz: Passed t-test for 60k traces<br>Radboud: M: Graz: Passed CPA for 15M traces |
| **Elephant** | |
| **Grain-128AEAD** | |
| **GIFT-COFB** | Tsinghua: M: Alexandre Adomnicai: Failed t-test for 100k traces<br>Shanghai Jiao Tong: M: Alexandre Adomnicai: Passed t-test for 20k traces |
| **ISAP** | T-test not applicable to the mode-protected implementation |
| **PHOTON-Beetle** | |
| **Romulus** | Tsinghua: M: Alexandre Adomnicai: Failed t-test for 100k traces<br>Shanghai Jiao Tong: M: Alexandre Adomnicai: Passed t-test for 100k traces |
| **SPARKLE** | |
| **TinyJAMBU** | |
| **Xoodyak** | |


Benchmarking
Platform
& Tools

# Benchmarking Approach

**FPGA Family:** Xilinx Artix-7

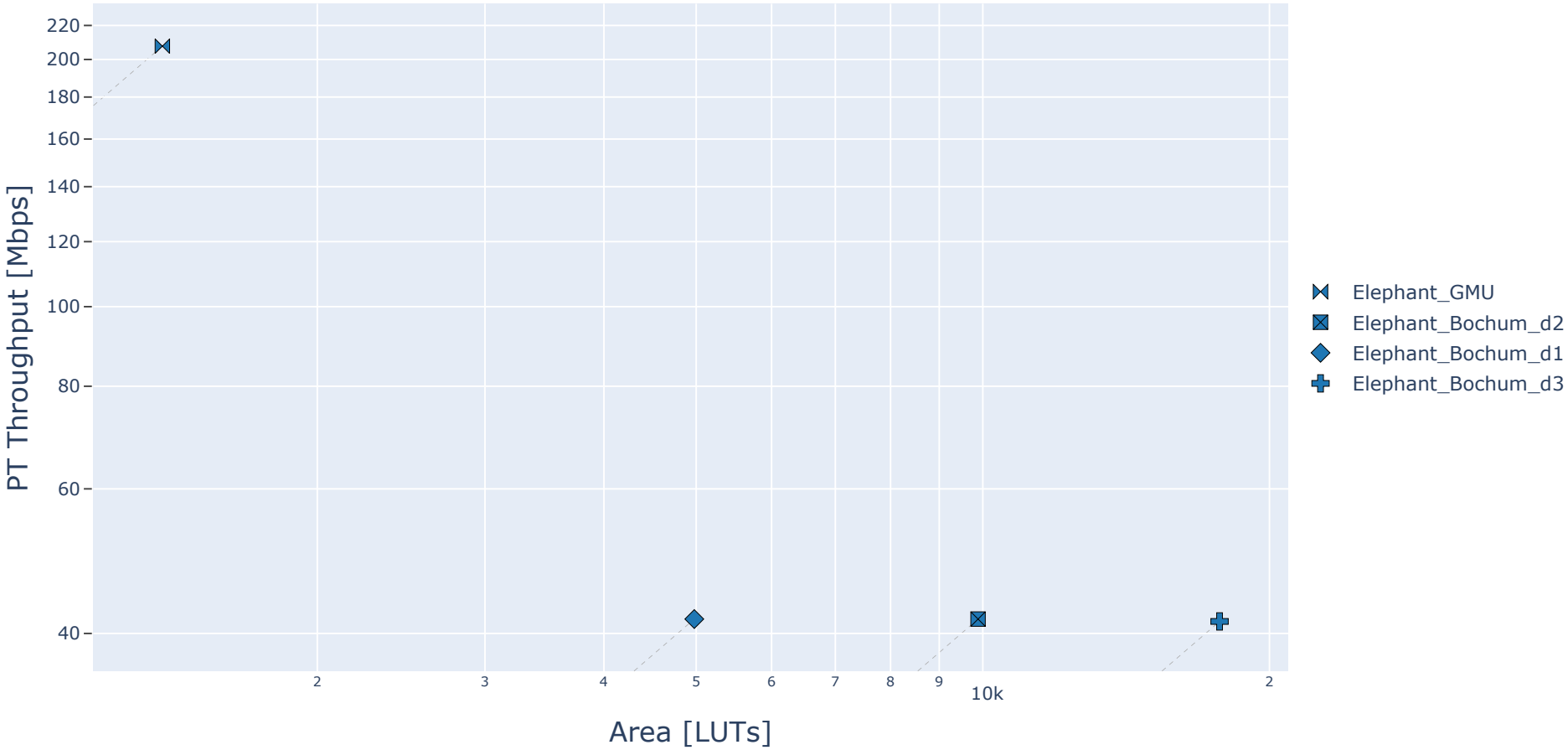
**FPGA Device:** xc7a100tftg256-2L
(from the CW305 board)

**FPGA Toolset:** Vivado ML 2022.1

**Automation & Optimization:**
Xeda
(developed at GMU)

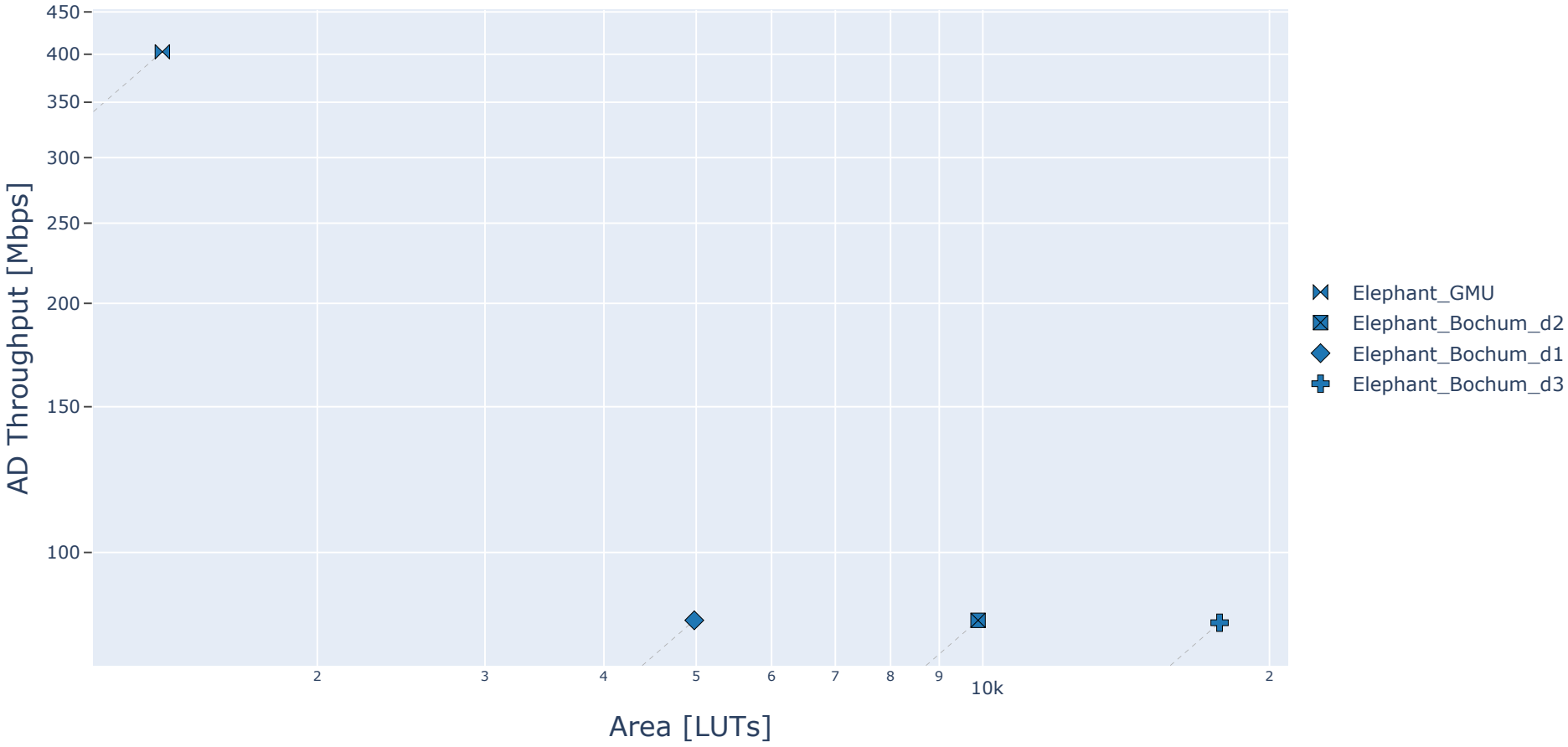
# Benchmarking Results

# Elephant: PT Throughput vs. Area



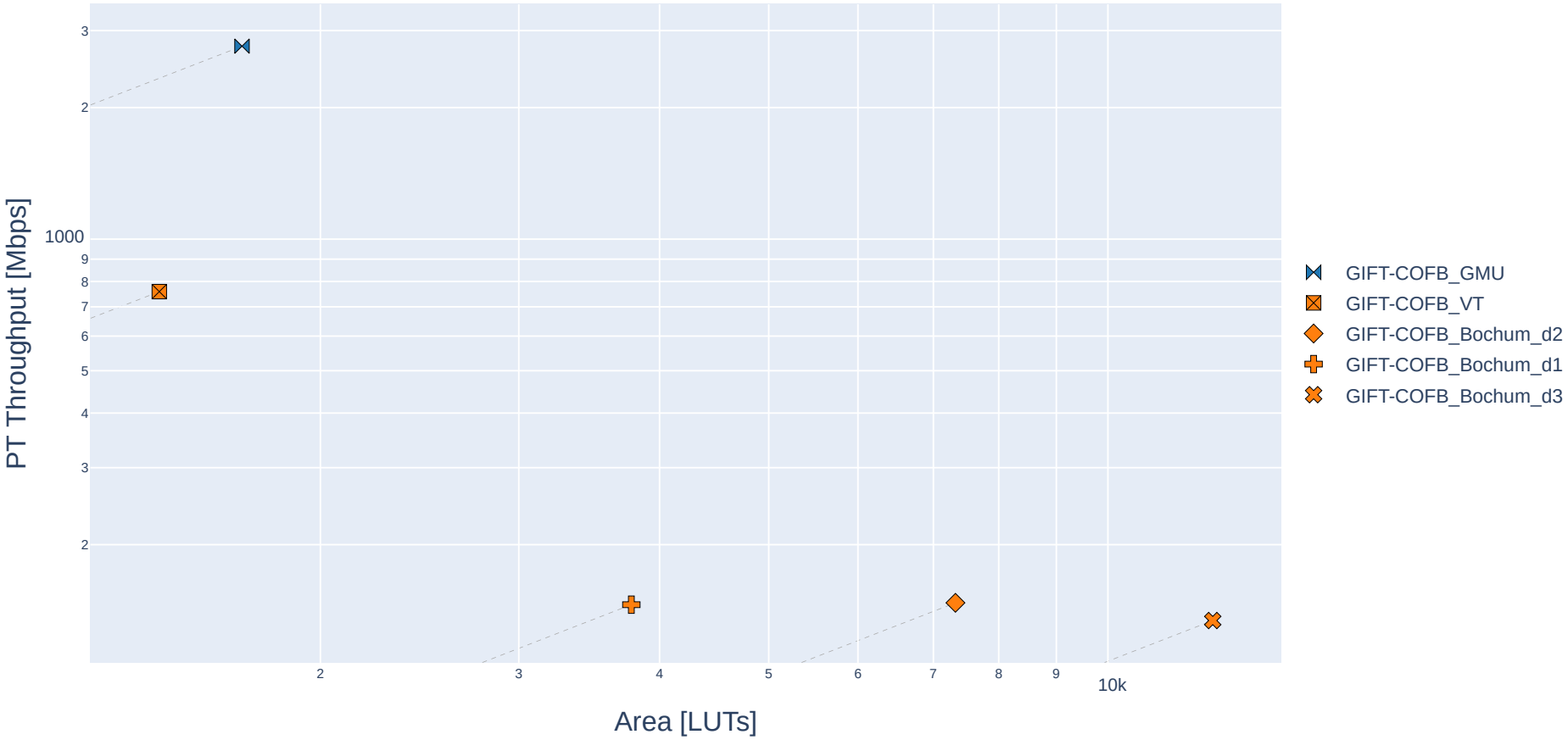

- Typical dependence
- A single unprotected implementation
- Protected implementations generated automatically using AGEMA
- Different-order AGEMA-generated protected implementations use the same number of clock cycles but differ in terms of maximum clock frequency

# Elephant: AD Throughput vs. Area



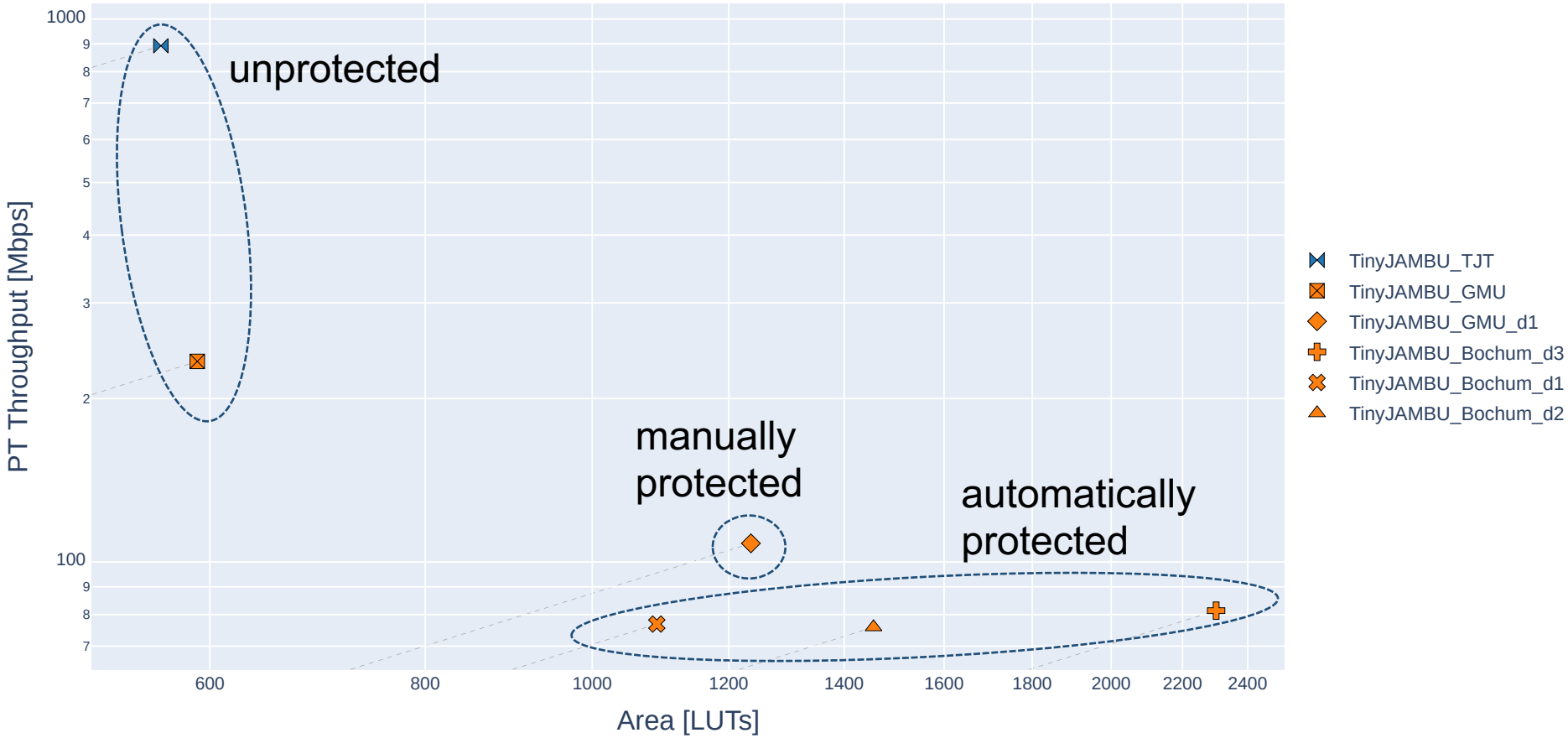

- Different throughputs for AD
- Identical areas (the same circuit used for processing AD and PT)
- Similar dependencies

# GIFT-COFB: PT Throughput vs. Area



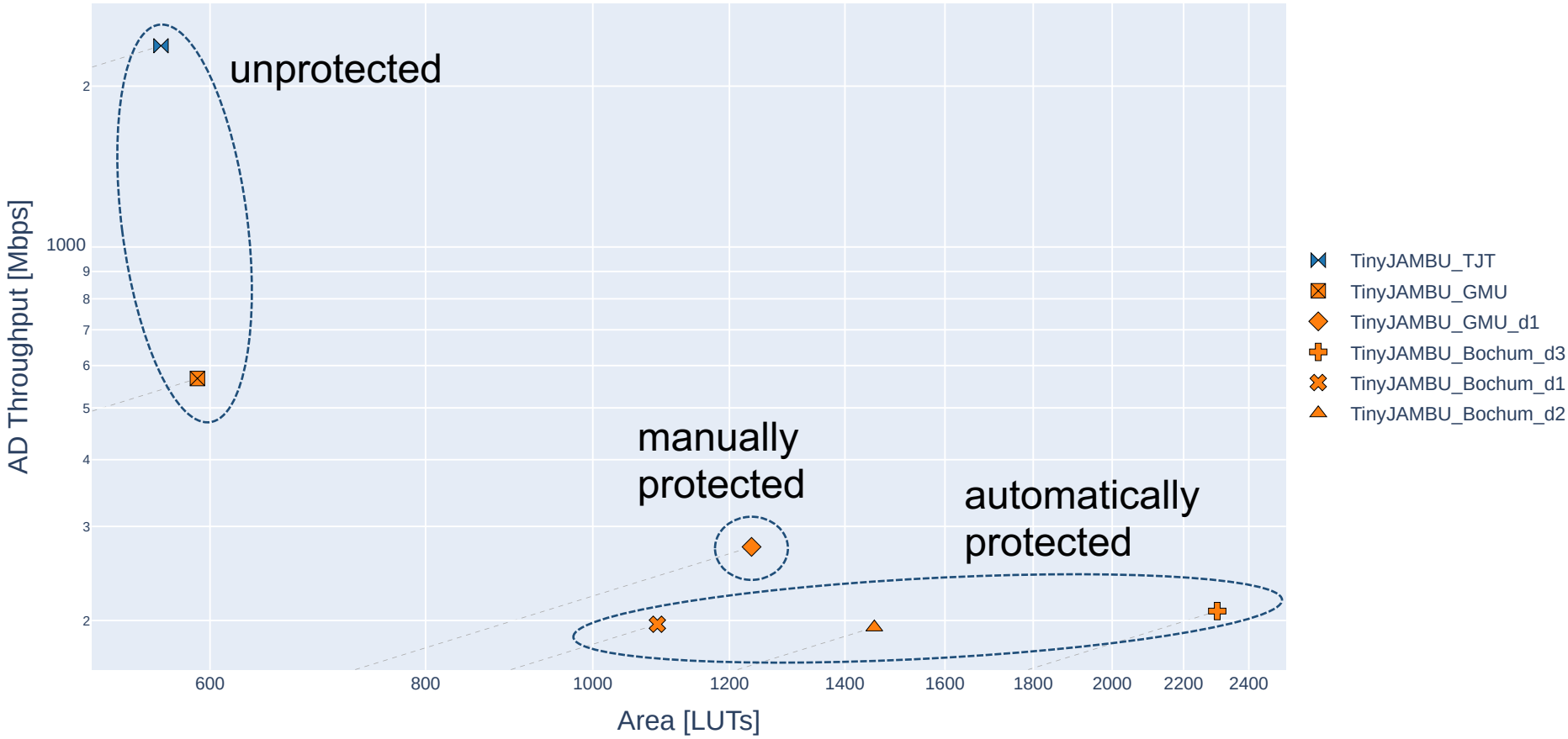

- Two unprotected designs
- Protected implementations generated automatically using AGEMA based on the less efficient design
- d2 design slightly faster than d1 design

# TinyJAMBU: PT Throughput vs. Area

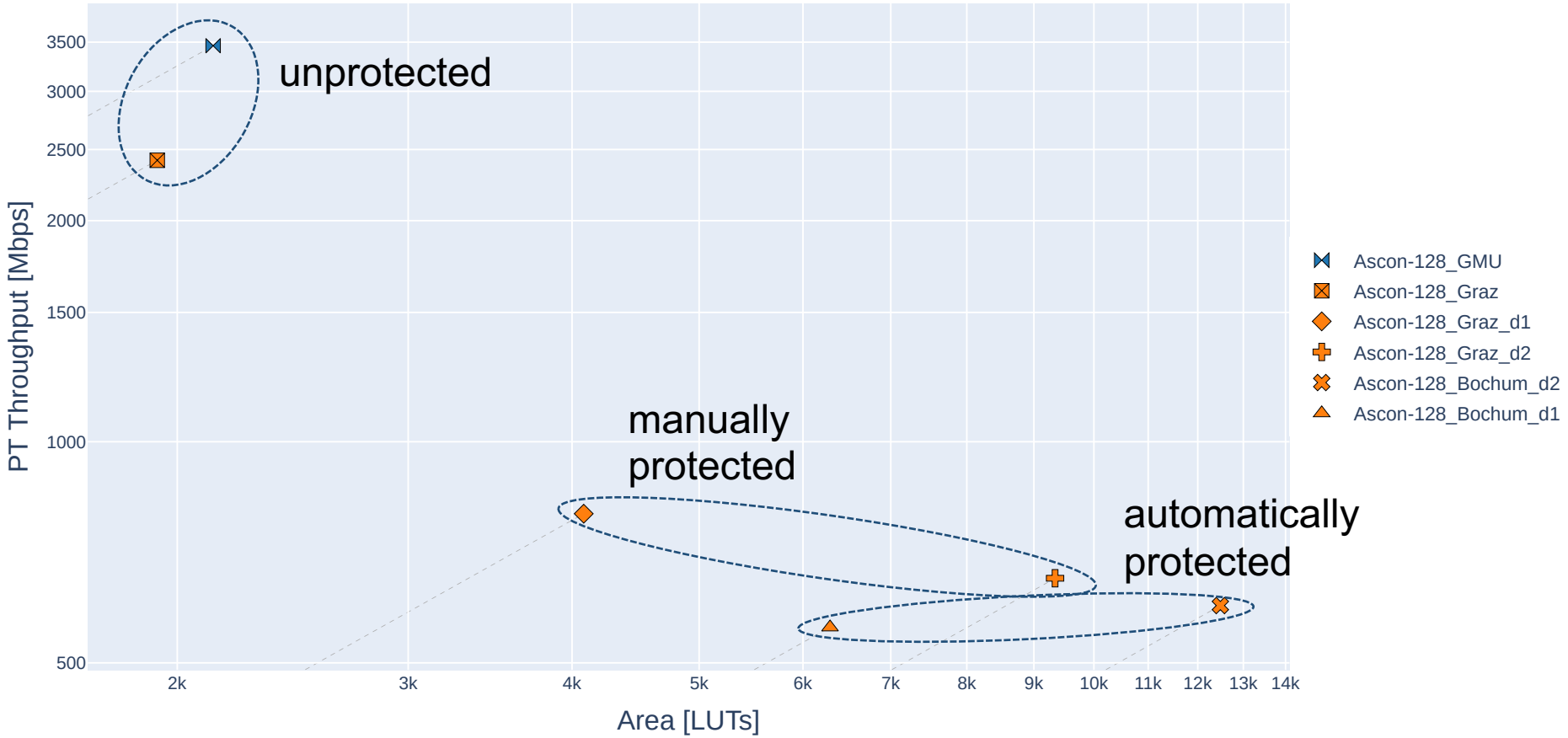



- Two unprotected designs
- All protected implementations generated based on the less efficient design
- One protected design (GMU_d1) generated manually; the remaining ones generated using AGEMA
- Anomaly: Order 1 manually protected design requires more area than the automatically generated one

# TinyJAMBU: AD Throughput vs. Area

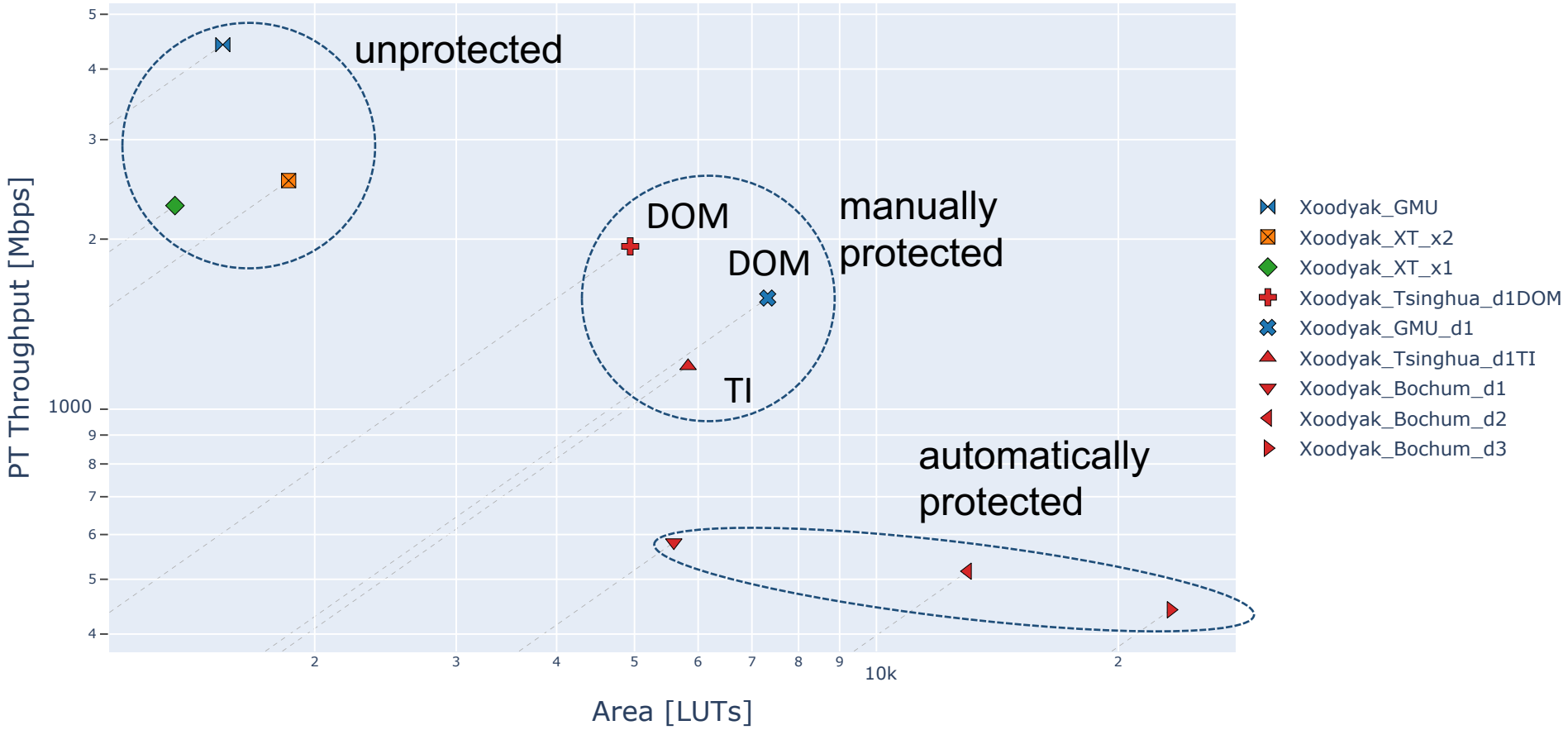


# Ascon-128: PT Throughput vs. Area



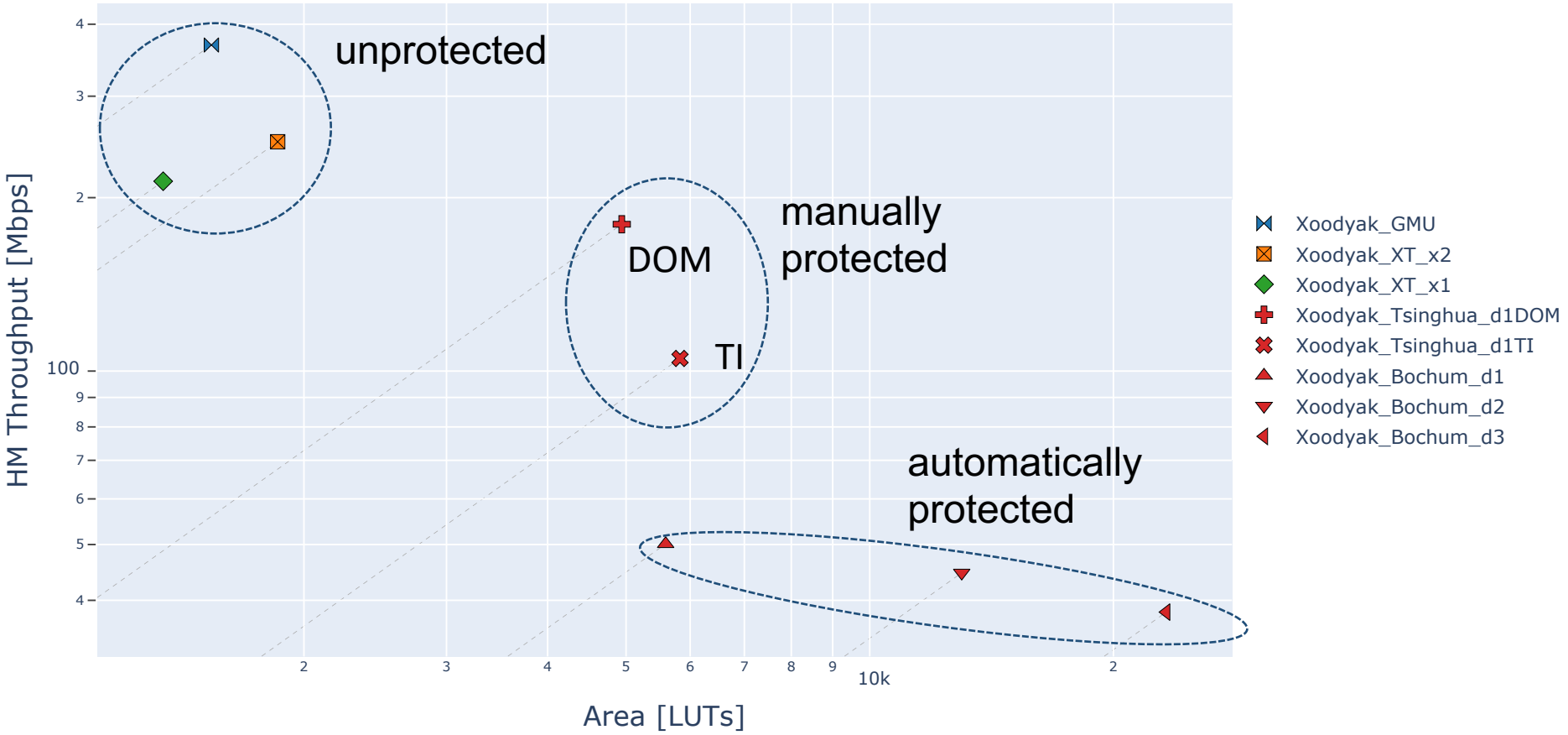

- Two unprotected designs
- All protected implementations generated based on the less efficient design
- Two protected design (Graz_d1 and Graz_d2) generated manually; the remaining two generated using AGEMA

# Xoodyak: PT Throughput vs. Area

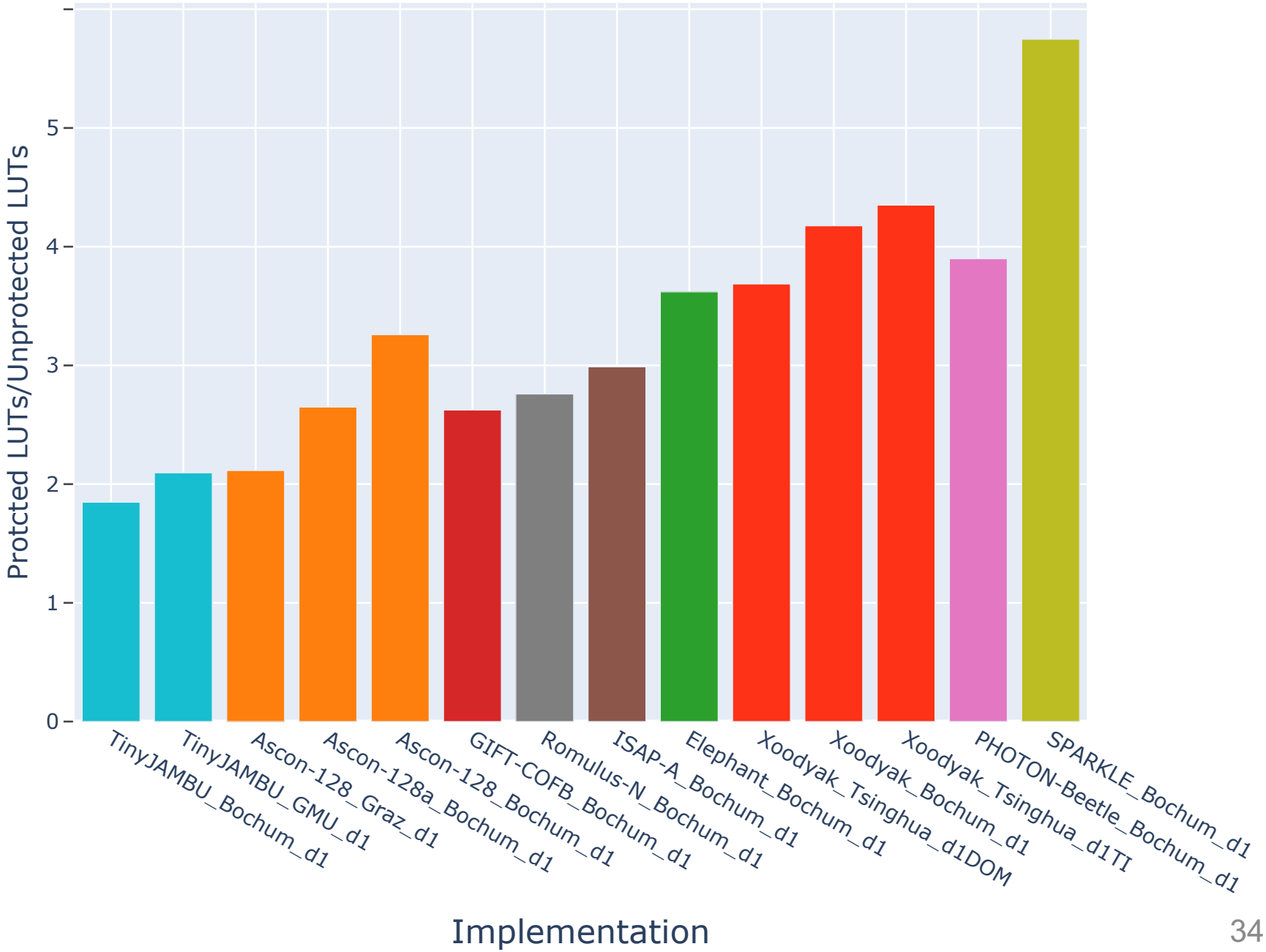



- Two manual protected designs by Tsinghua (based on Xoodyak_XT)
- One manual protected design by GMU (based on Xoodyak_GMU)
- Three protected designs from Bochum generated automatically based on Xoodyak_XT

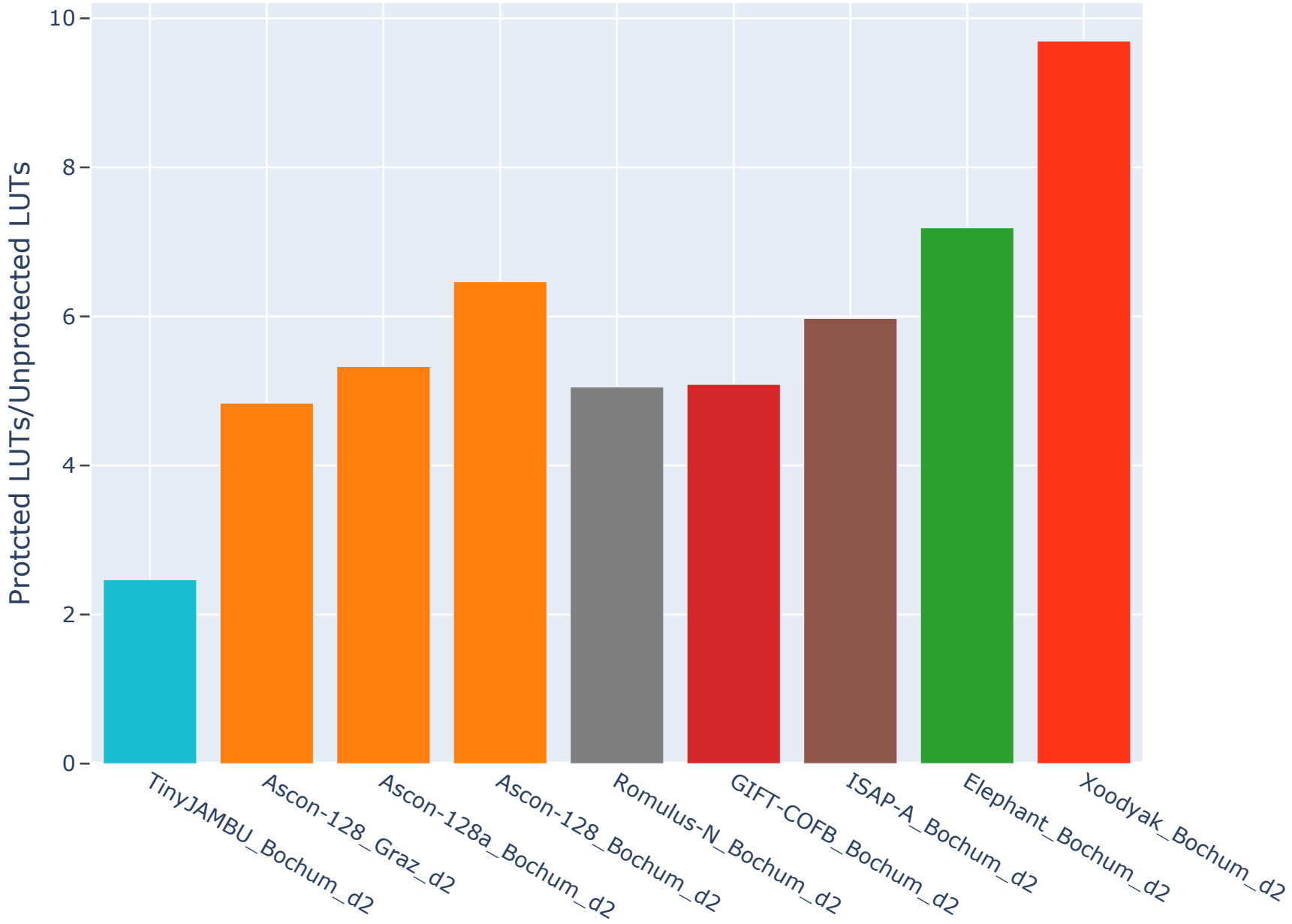
# Xoodyak: Hash Throughput vs. Area



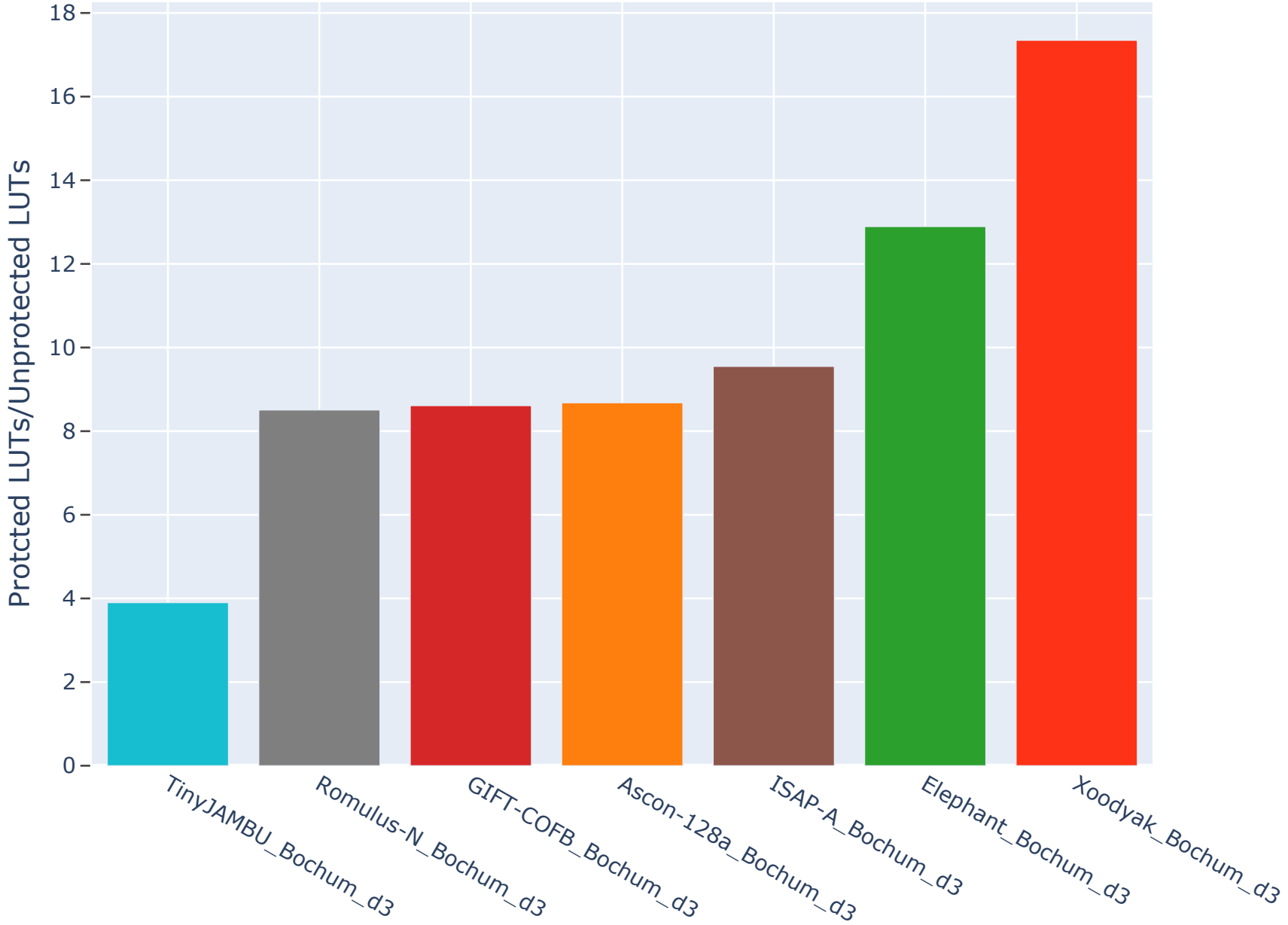
# Area Protected Order 1/Area Unprotected



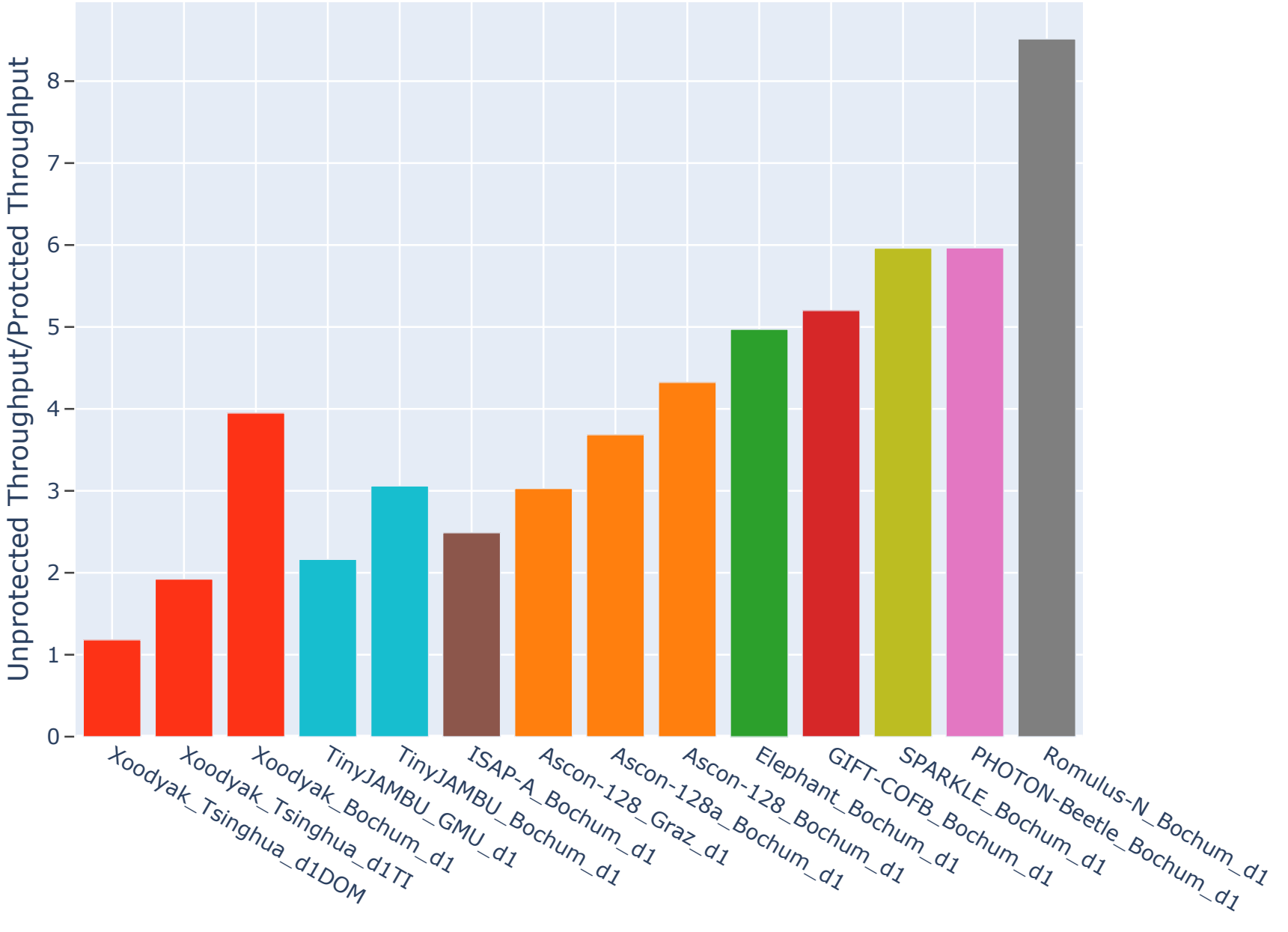
# Area Protected Order 2/Area Unprotected



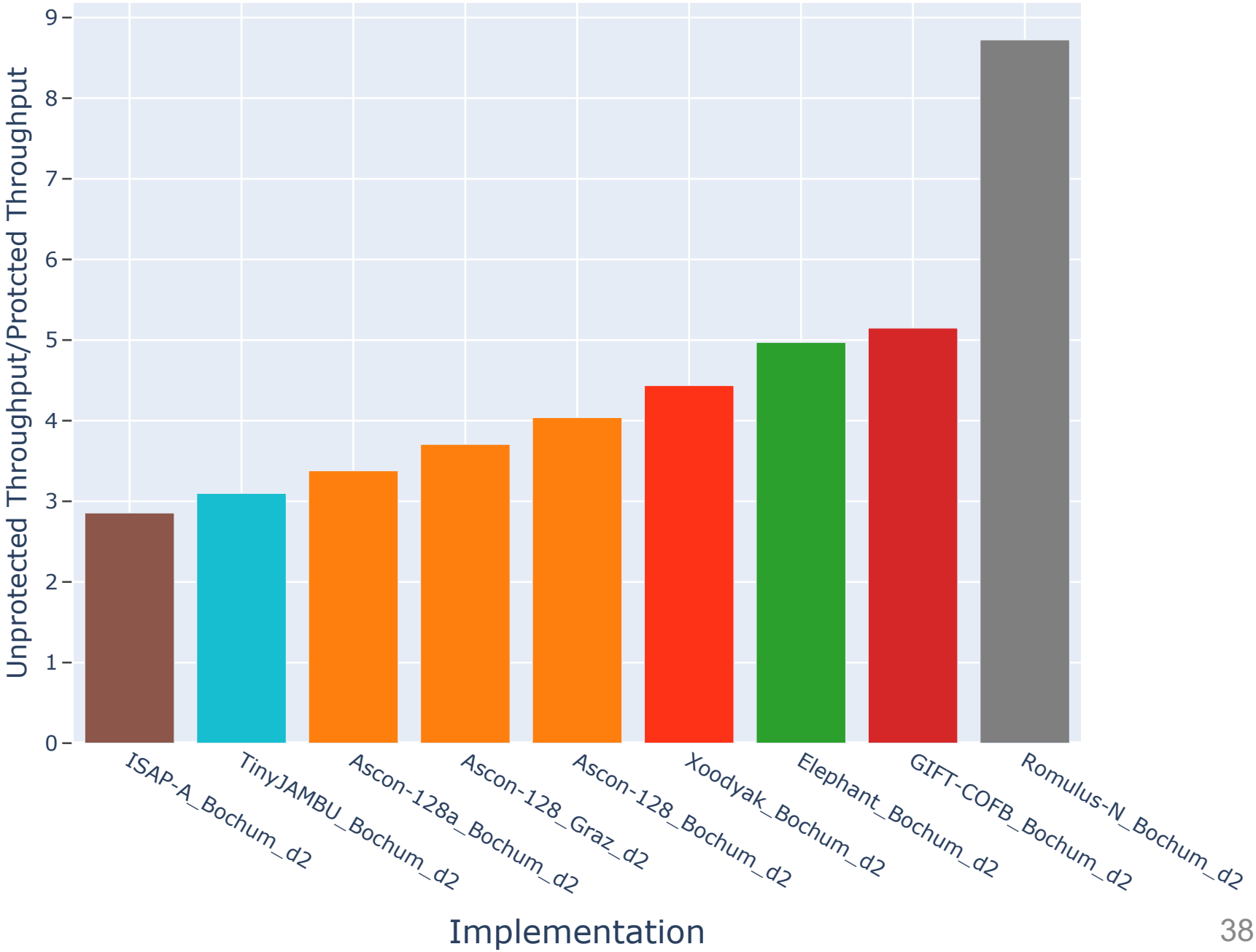
# Area Protected Order 3/Area Unprotected



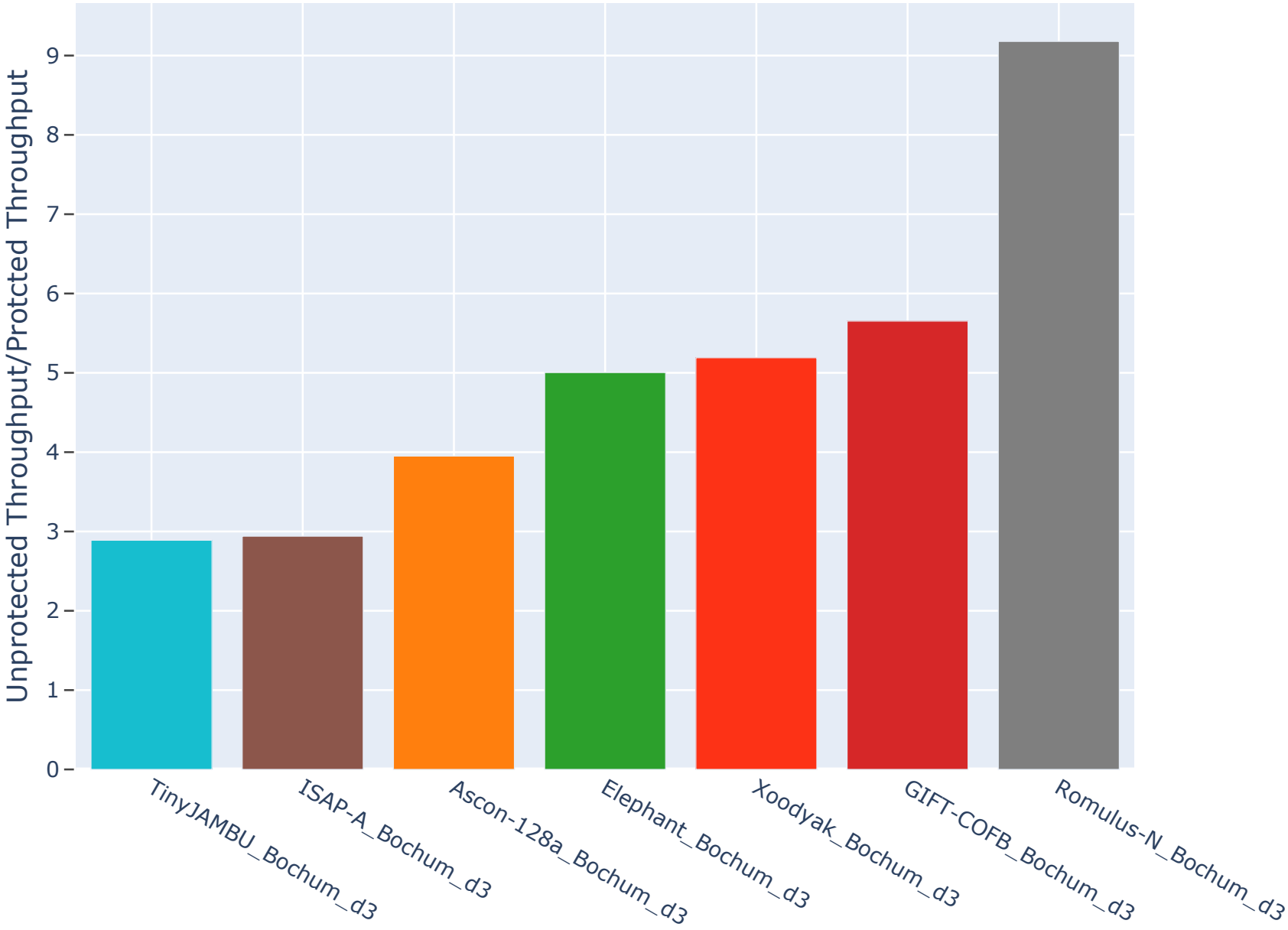
# Throughput Unprotected/Throughput Protected Order 1



# Throughput Unprotected/Throughput Protected Order 2

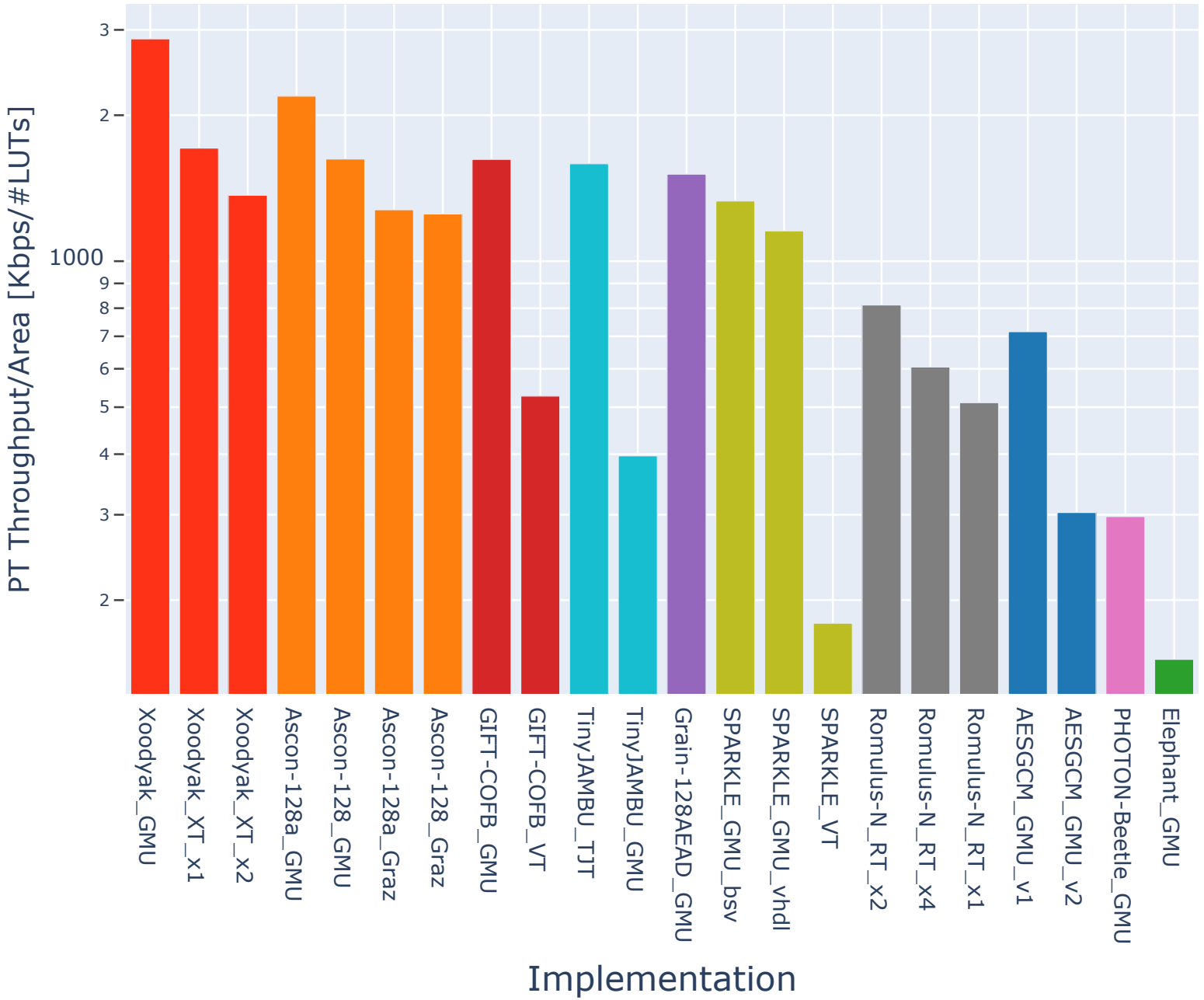


# Throughput Unprotected/Throughput Protected Order 3



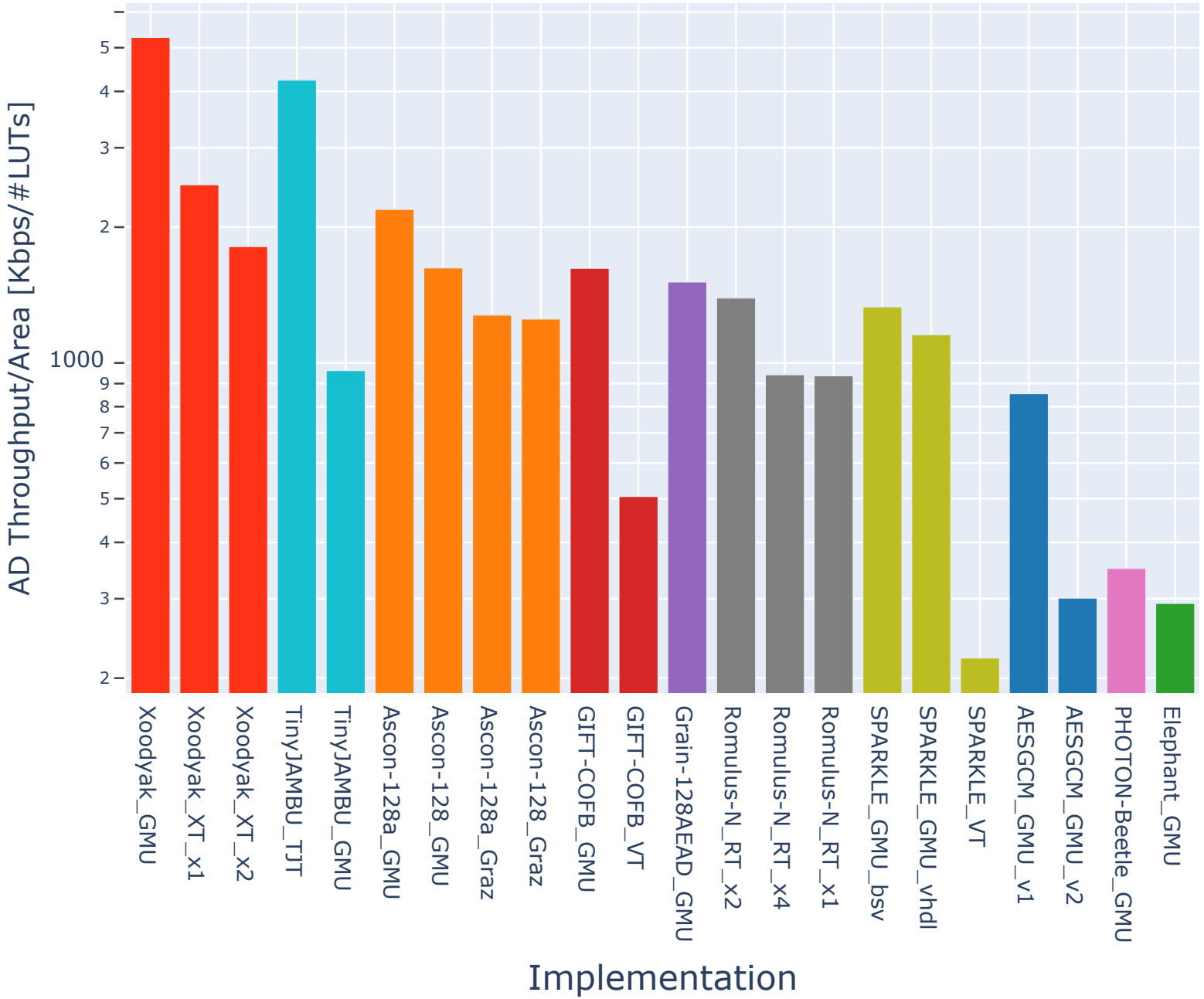
# PT Throughput vs. Area Unprotected

# PT Throughput/Area Unprotected

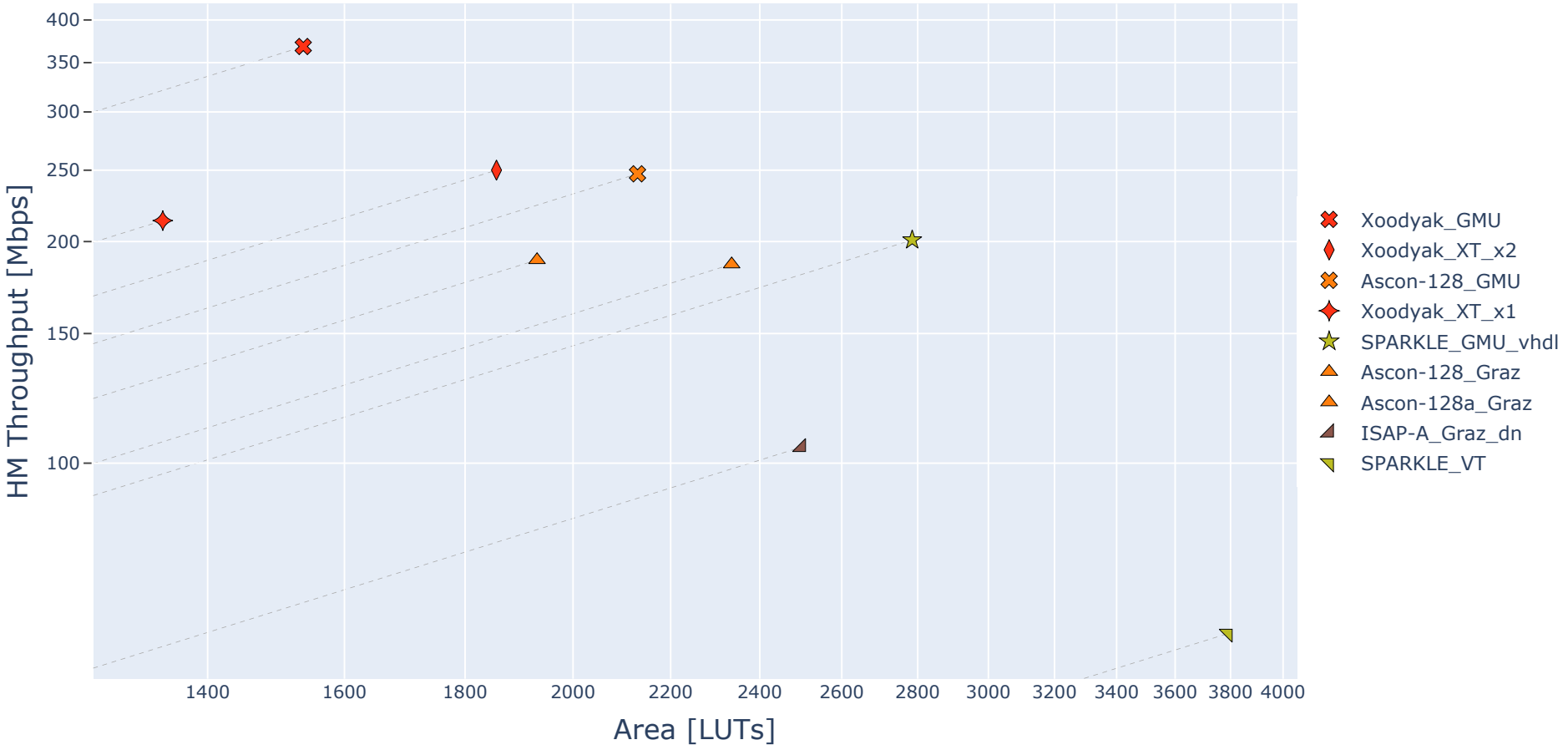


# AD Throughput vs. Area Unprotected

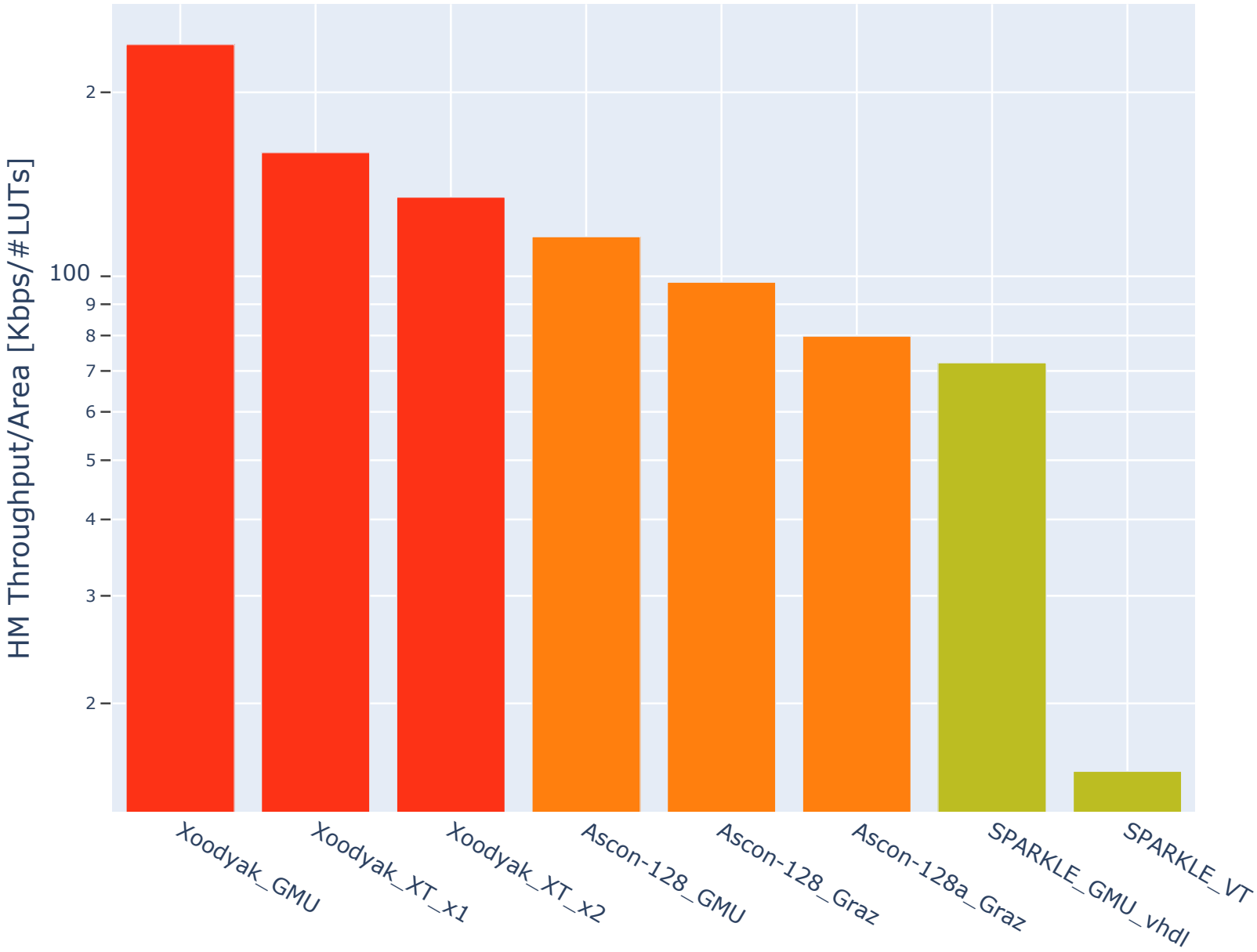
# AD Throughput/Area Unprotected



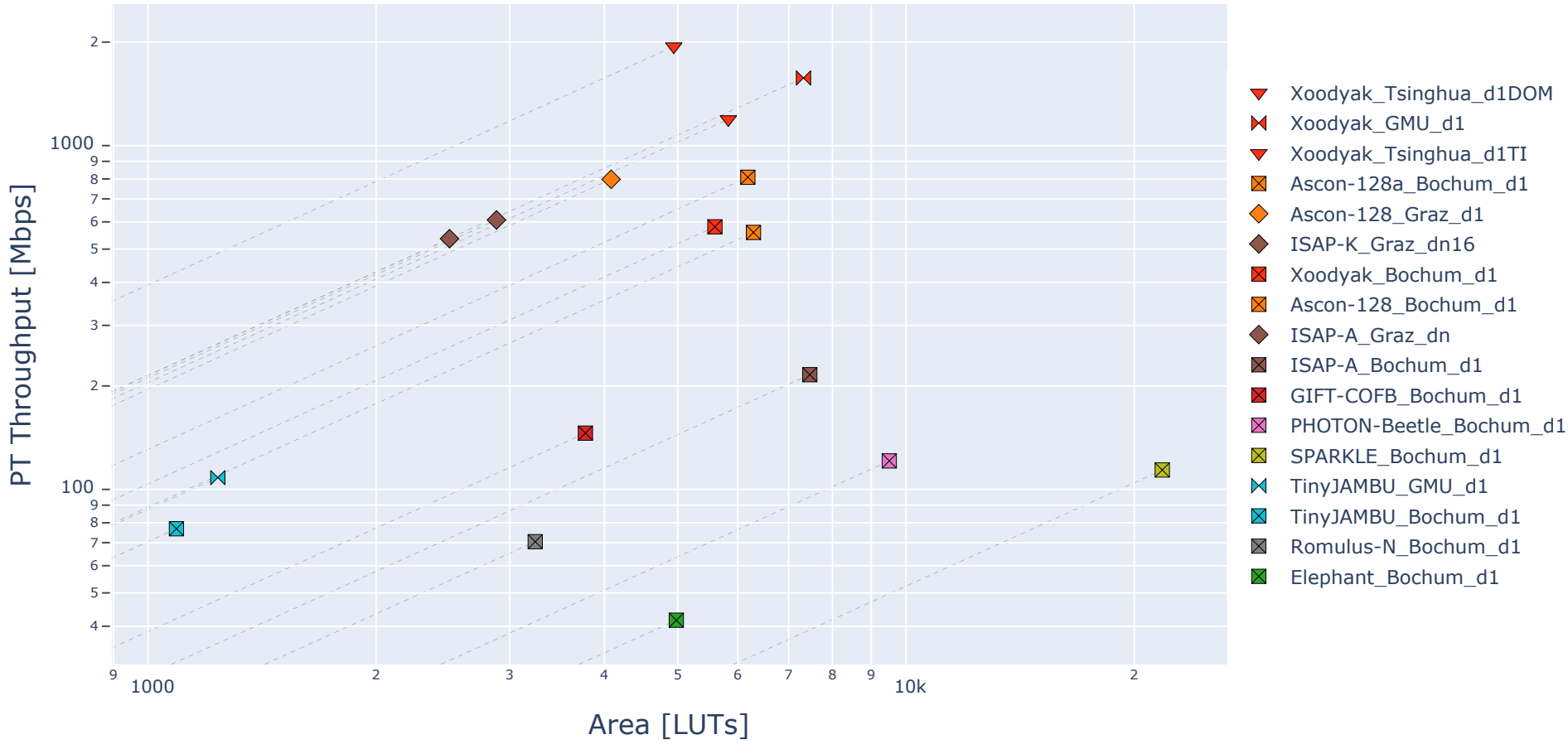
# Hashing Throughput vs. Area Unprotected



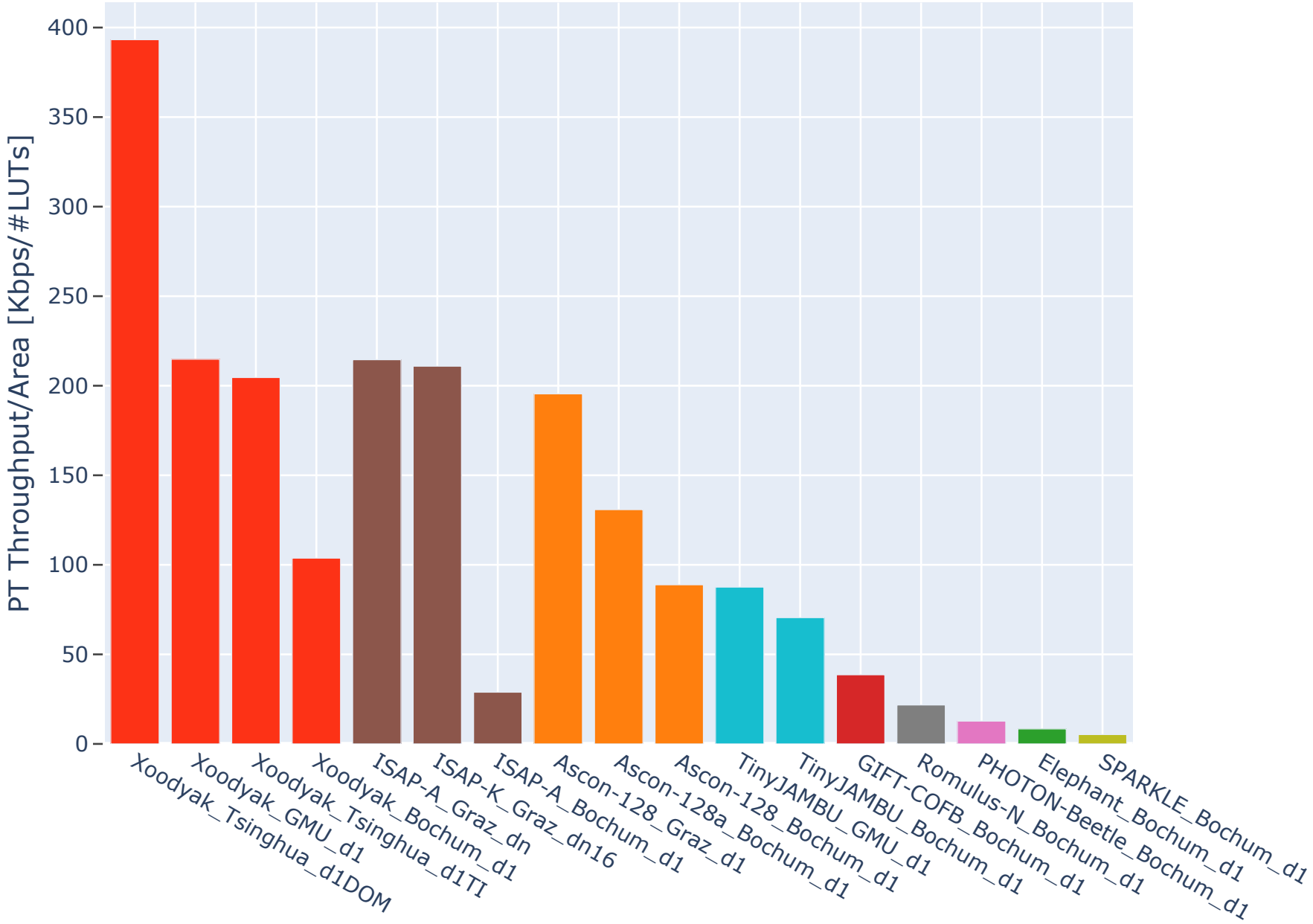
# Hashing Throughput/Area Unprotected



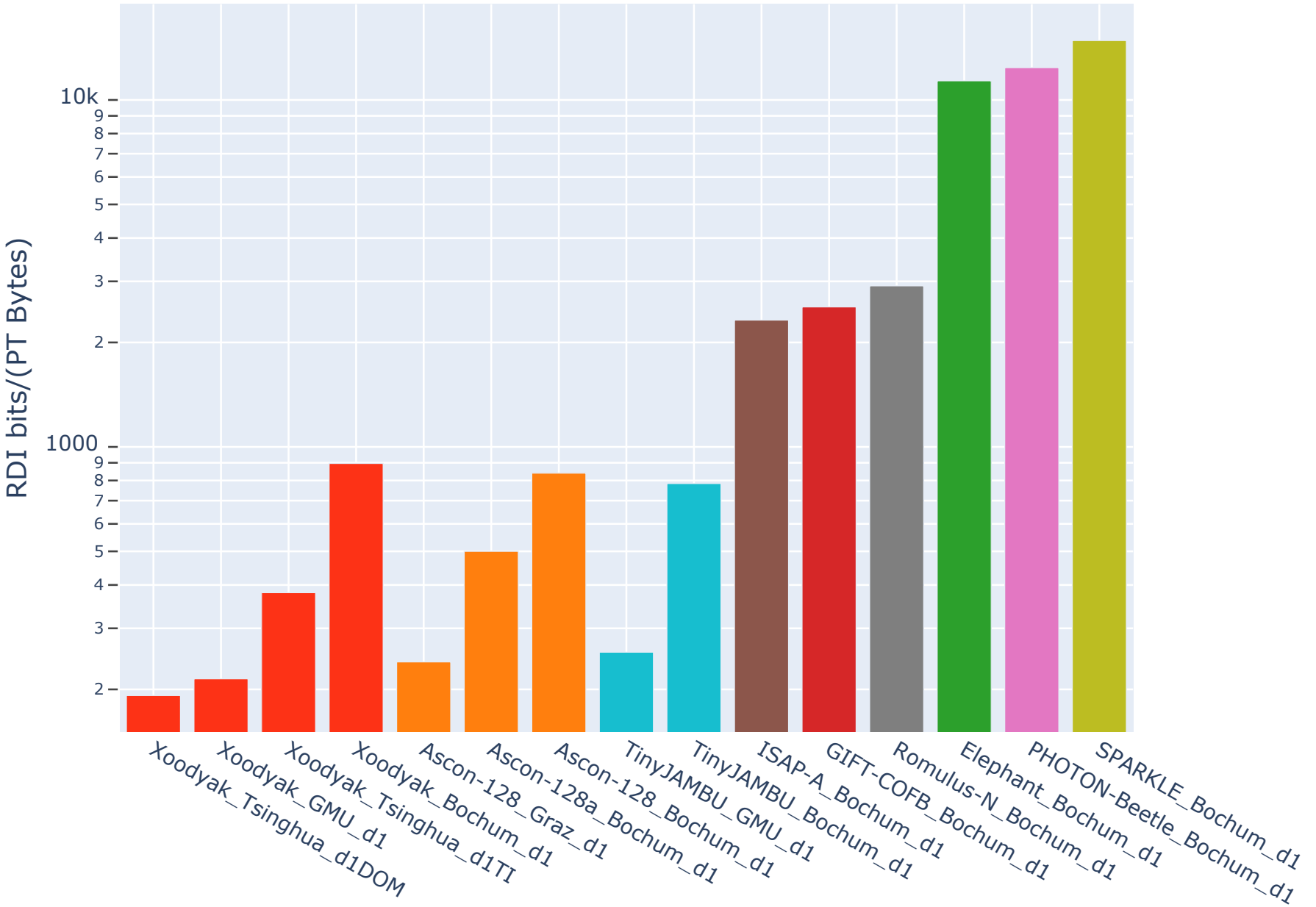
# Protected Order 1: PT Throughput vs. Area



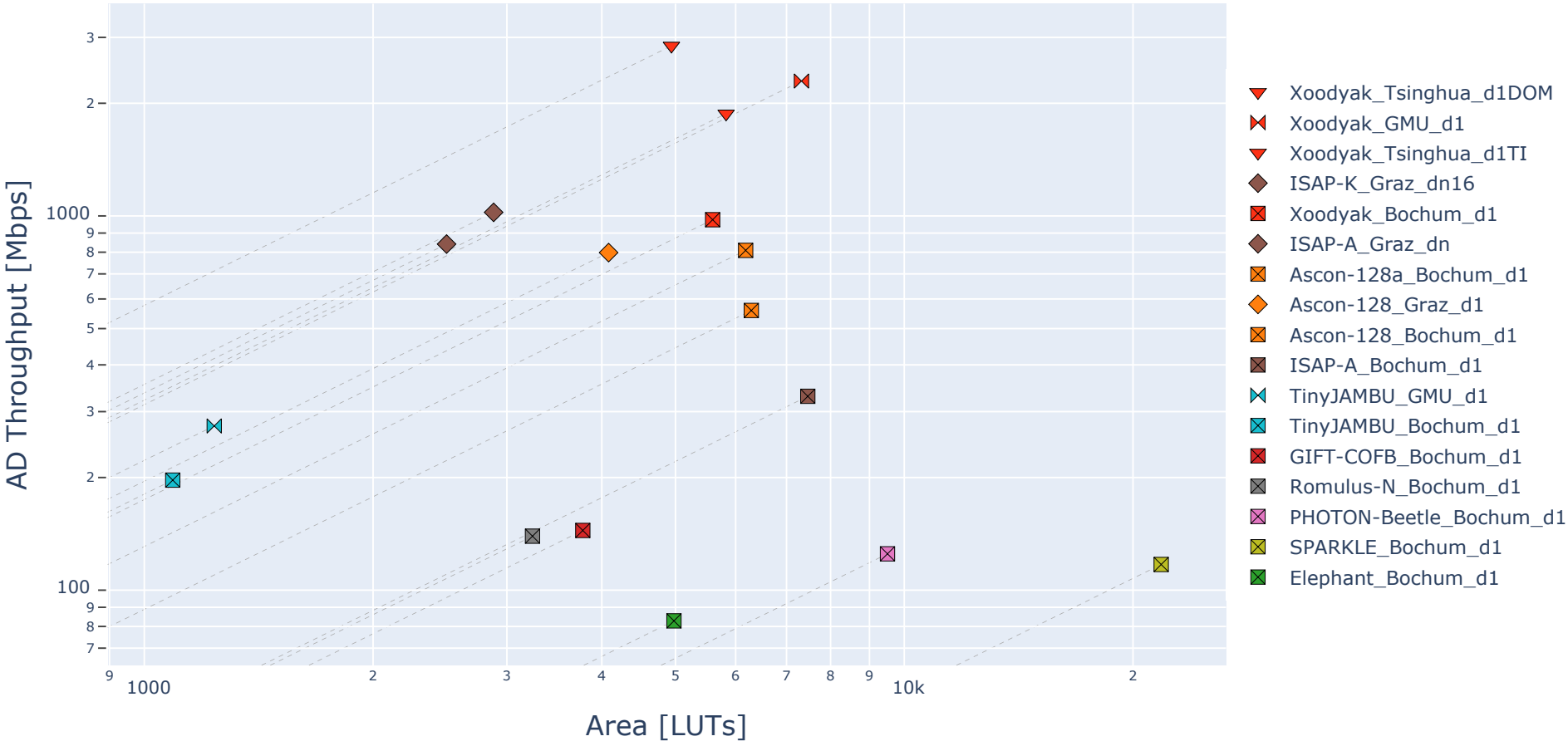
# Protected Order 1: PT Throughput/Area



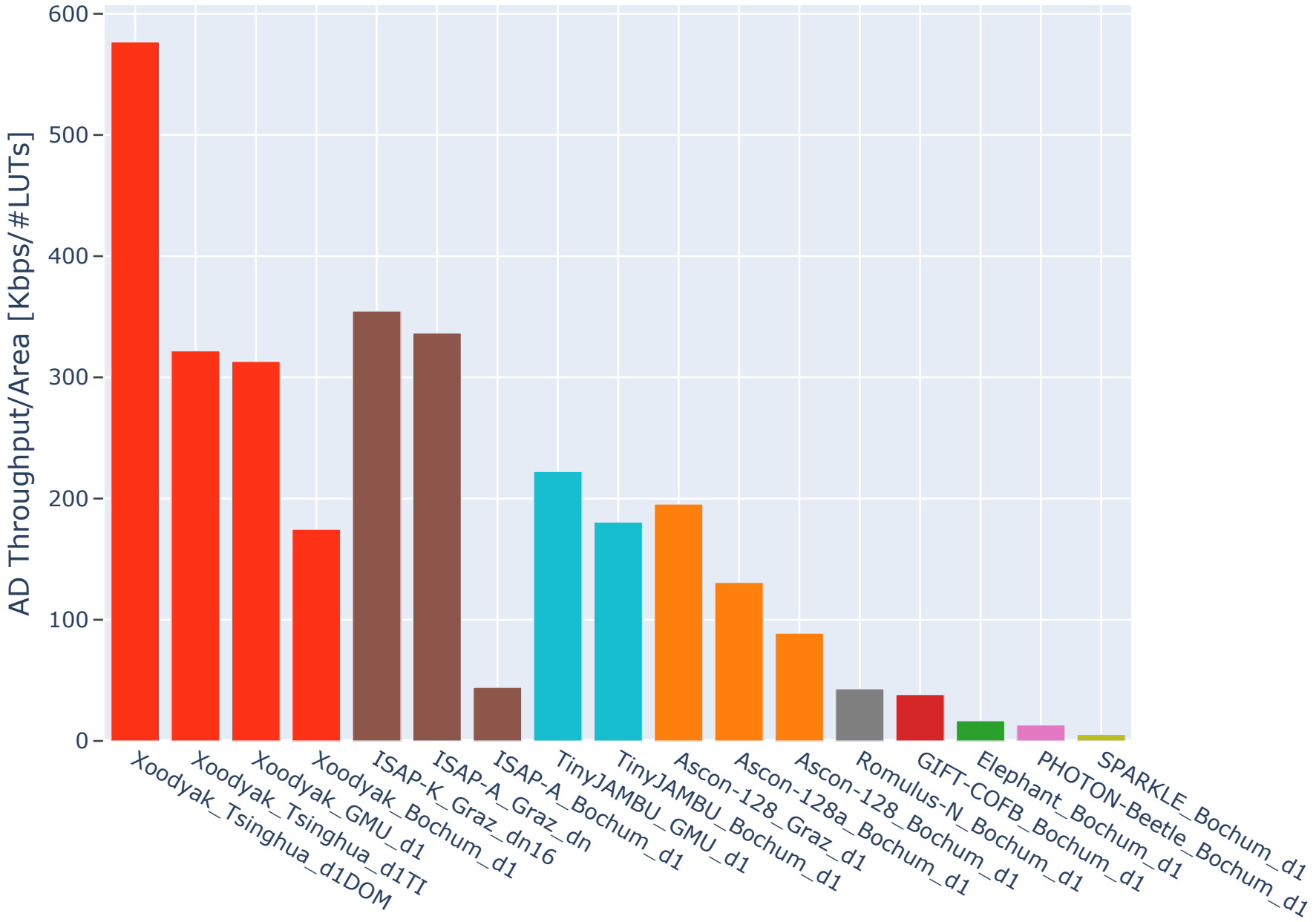
# Protected Order 1: Random bits/PT byte



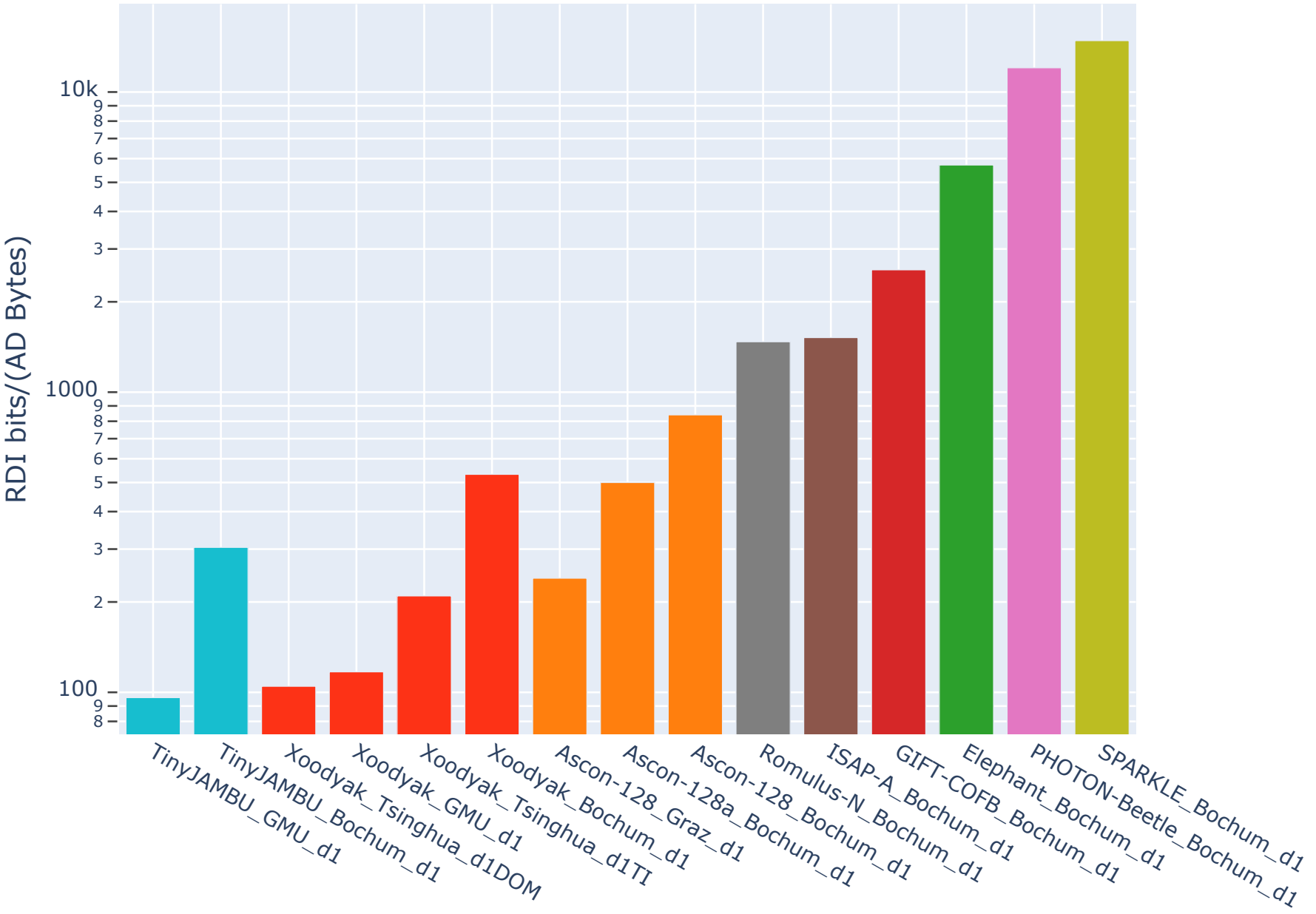
# Protected Order 1: AD Throughput vs. Area



# Protected Order 1: AD Throughput/Area



# Protected Order 1: Random bits/AD byte



# Masking for Hashing

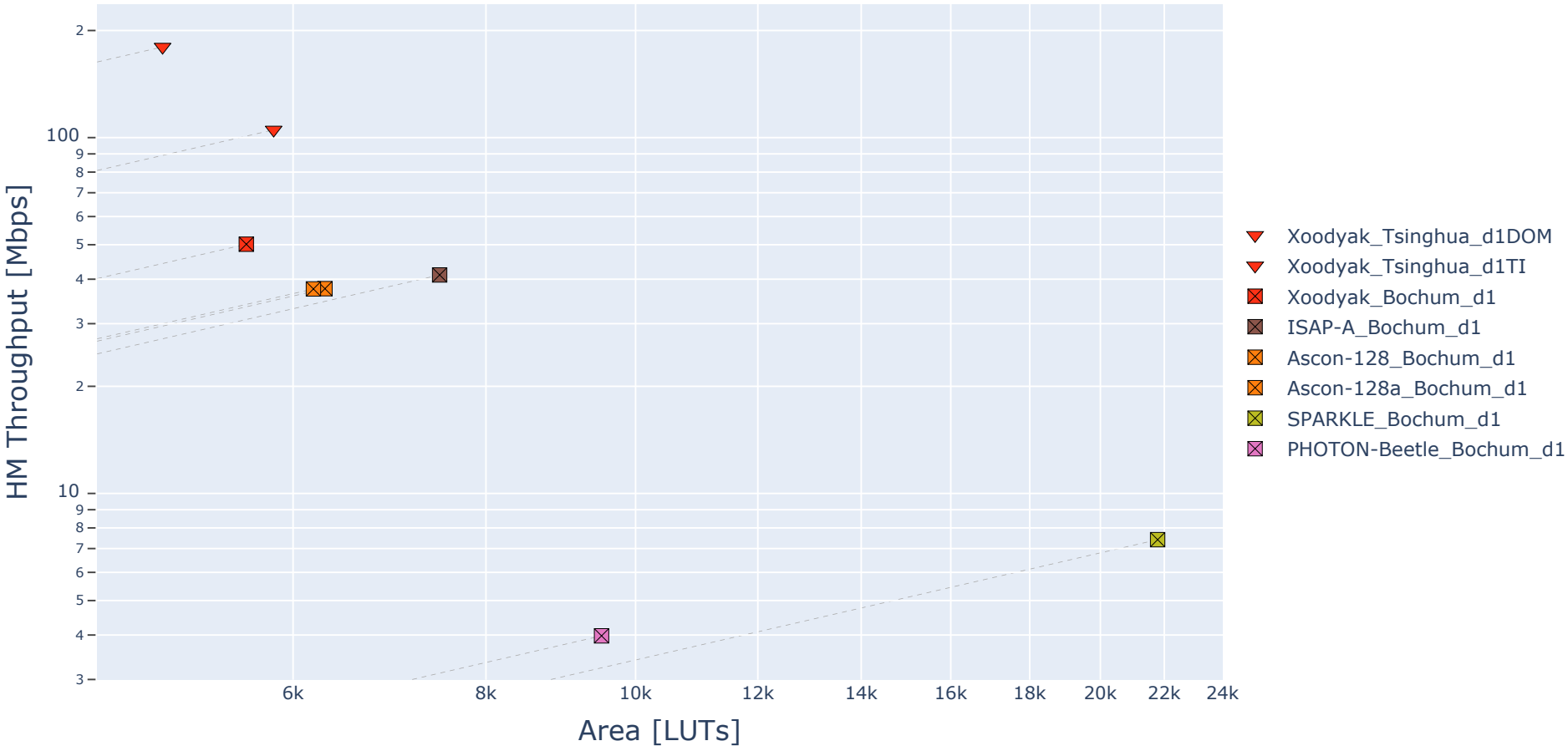
If input to hashing is public, no masking is required.

However, if implementations of authenticated encryption and hashing share resources, then it may be challenging to implement unmasked hashing on top of masked authenticated encryption.

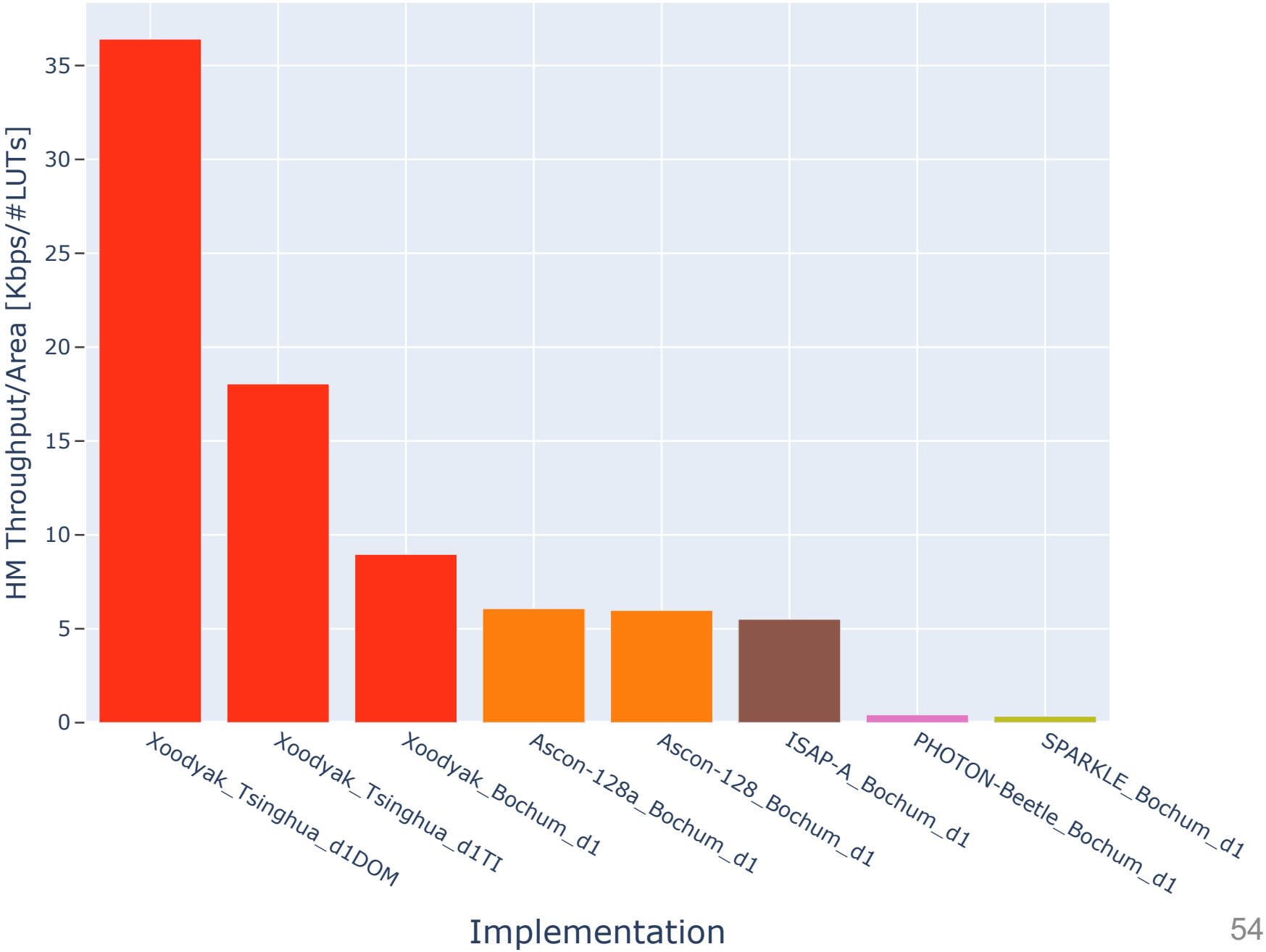
If input to hashing includes a secret key, as in HMAC, masked hashing may be advisable to prevent key recovery.

The hashing mode of ISAP does not offer protection against DPA-based message recovery attacks.
Thus, for HMAC-like applications, masking must be applied to hash functions sharing functionality with the authenticated encryption mode of ISAP.

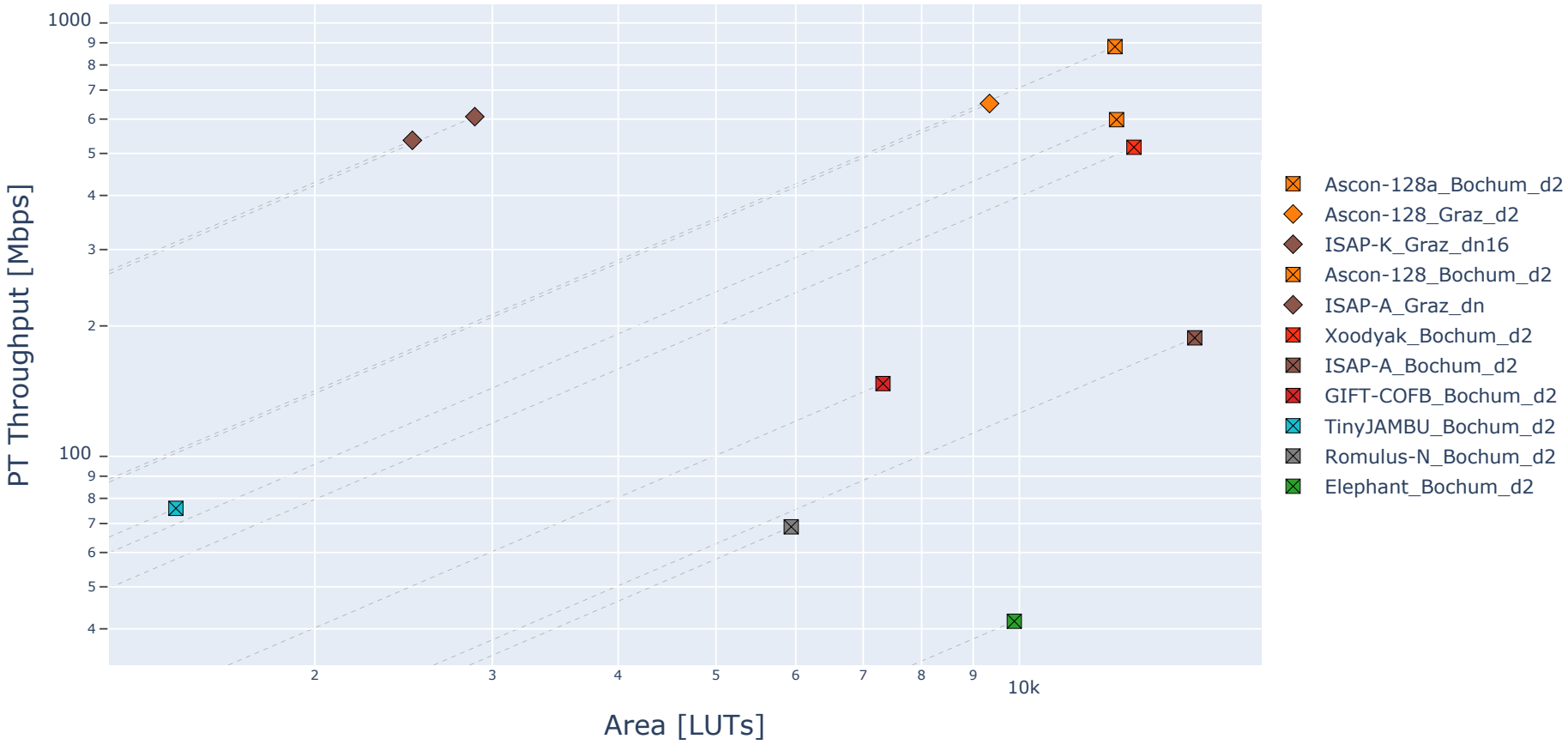
# Protected Order 1: Hashing Throughput vs. Area



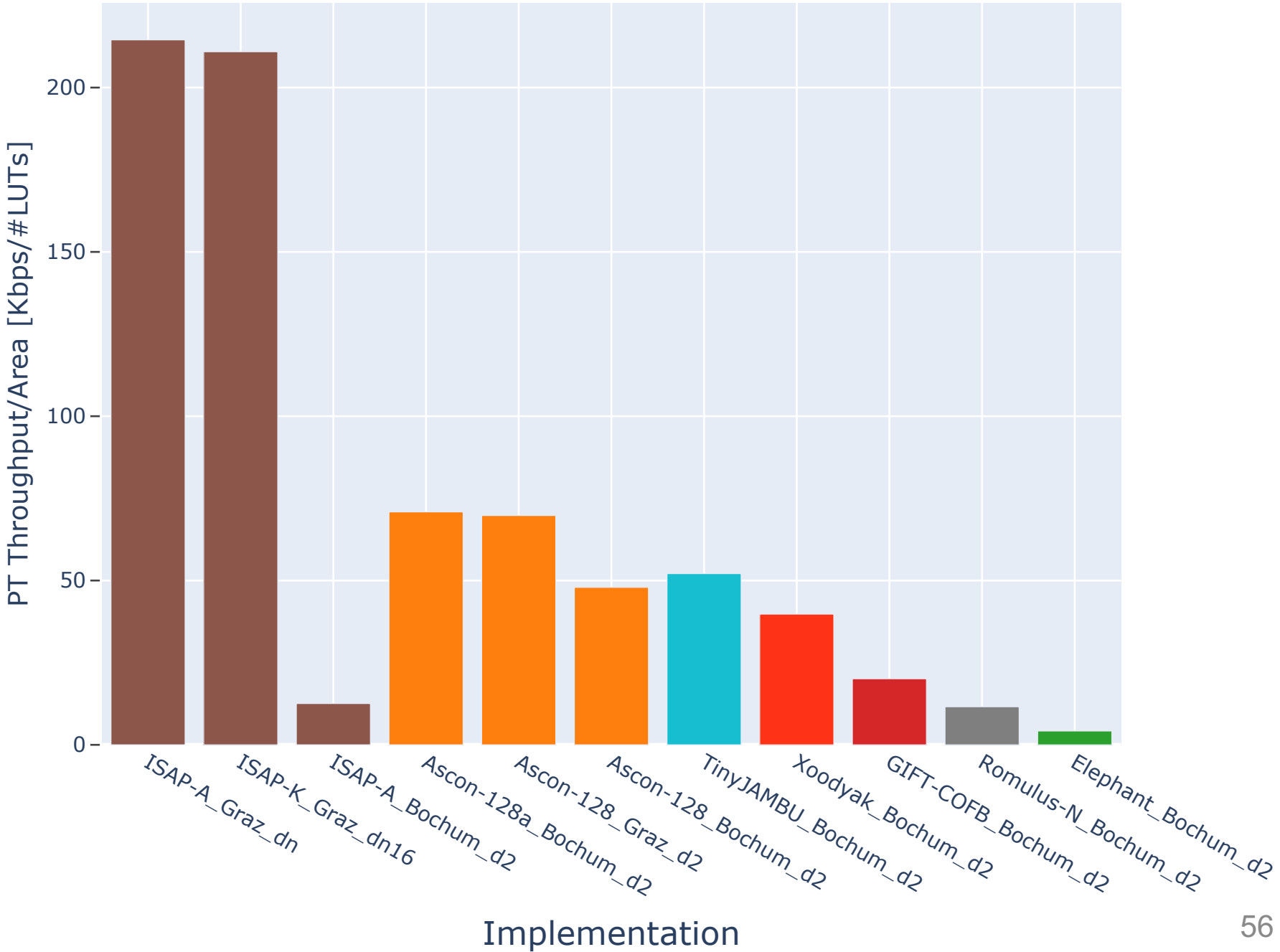
# Protected Order 1: Hashing Throughput/Area



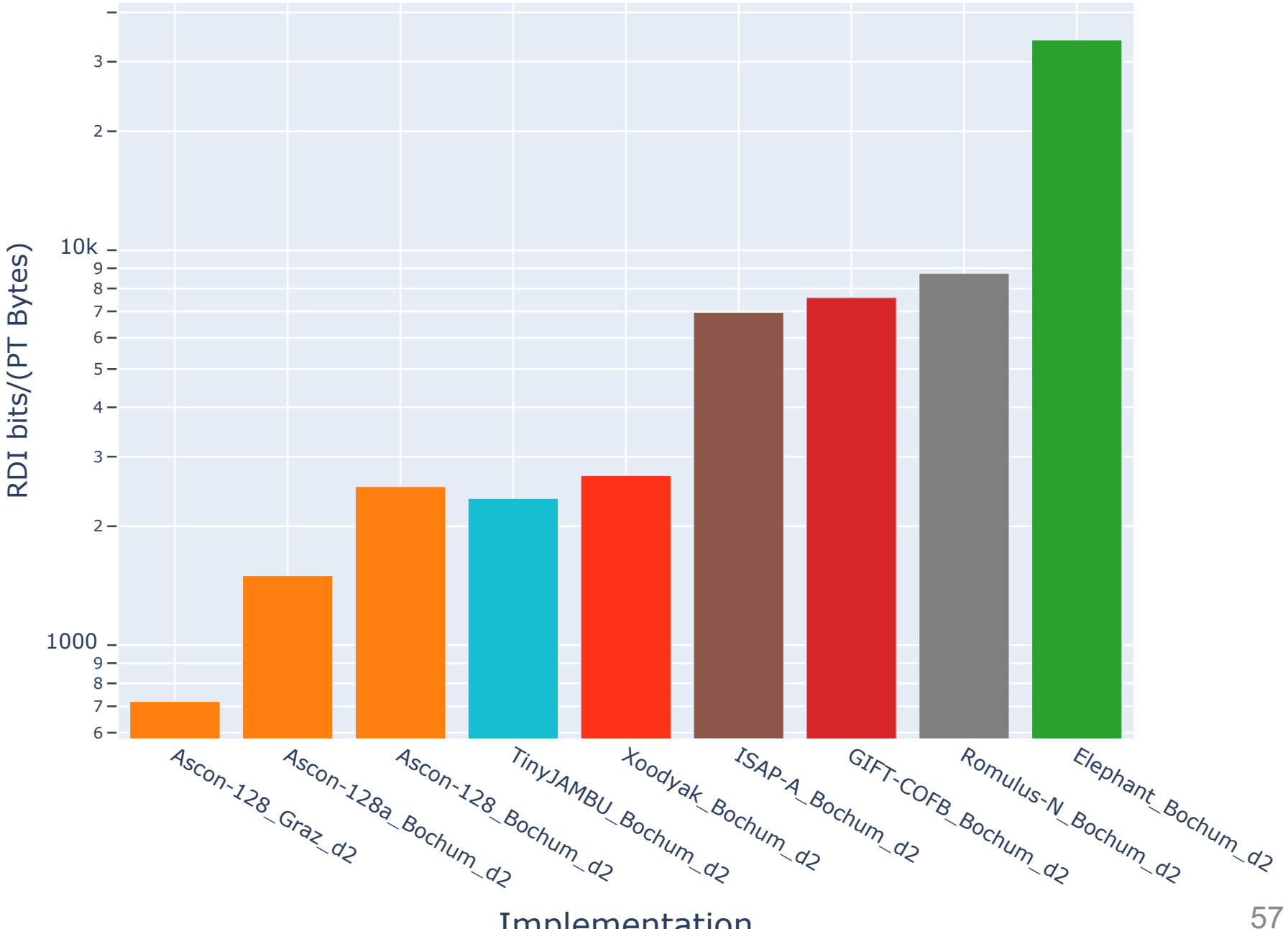
# Protected Order 2: PT Throughput vs. Area



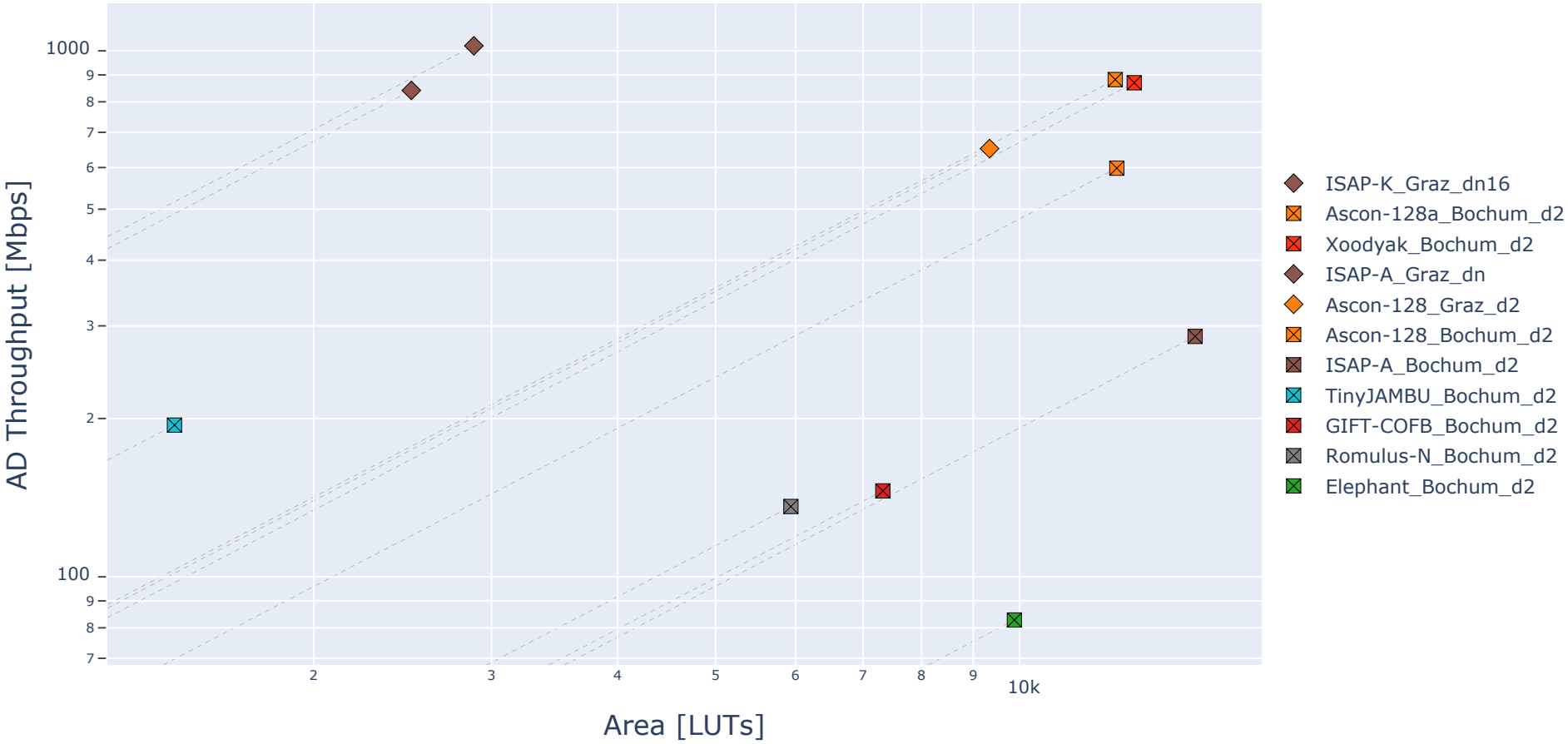
# Protected Order 2: PT Throughput/Area



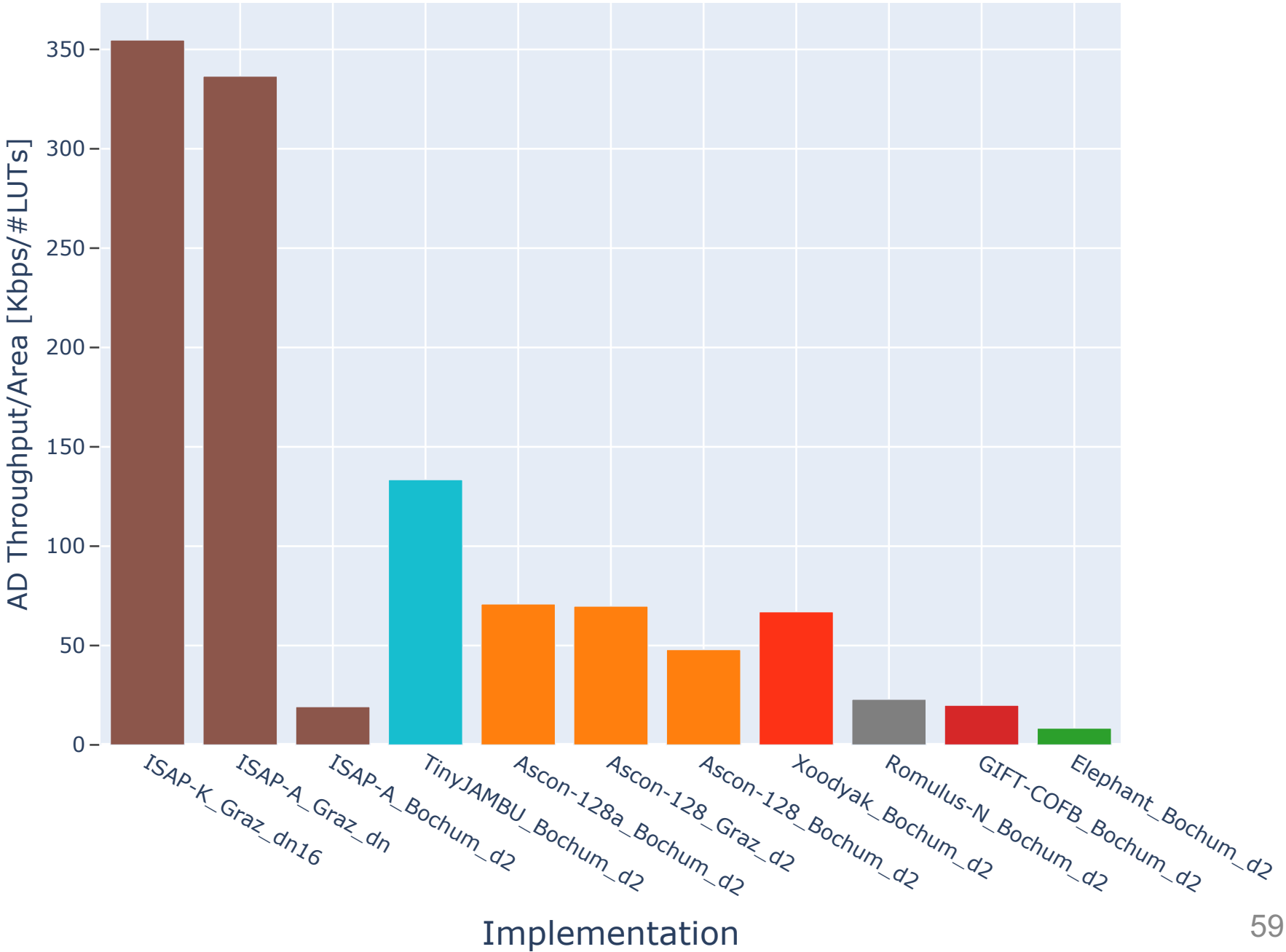
# Protected Order 2: Random bits/PT byte



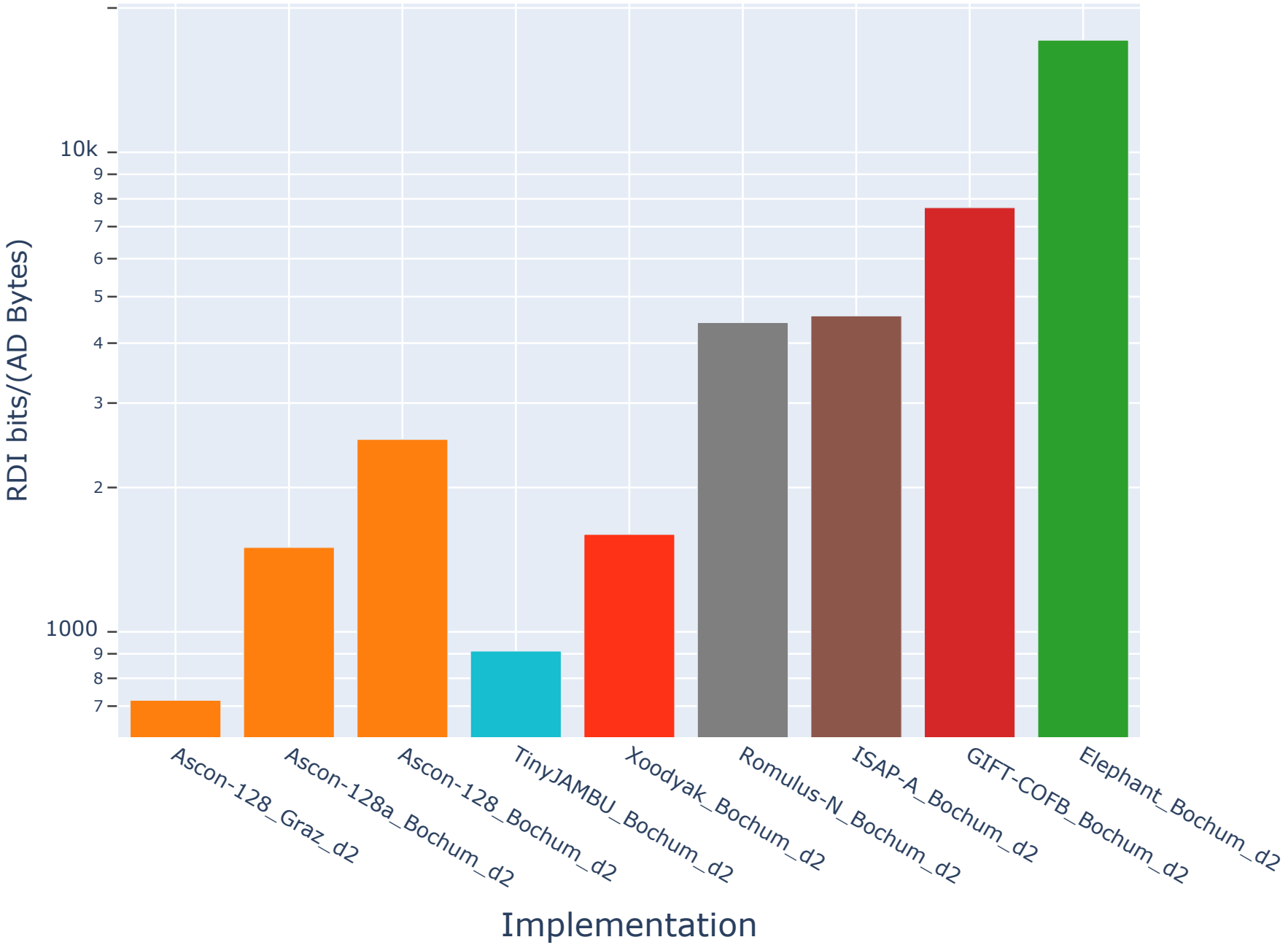
# Protected Order 2: AD Throughput vs. Area



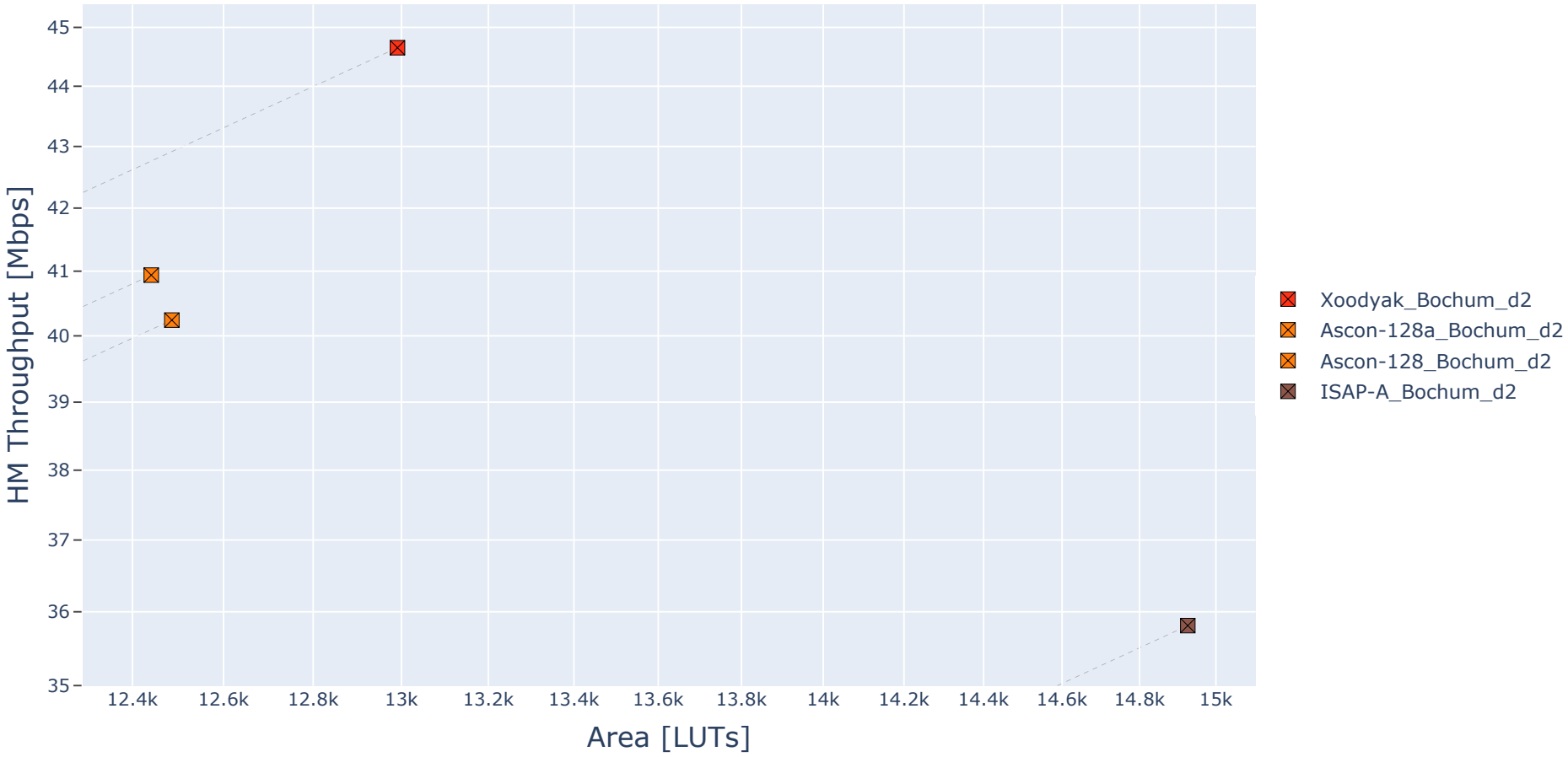
# Protected Order 2: AD Throughput/Area



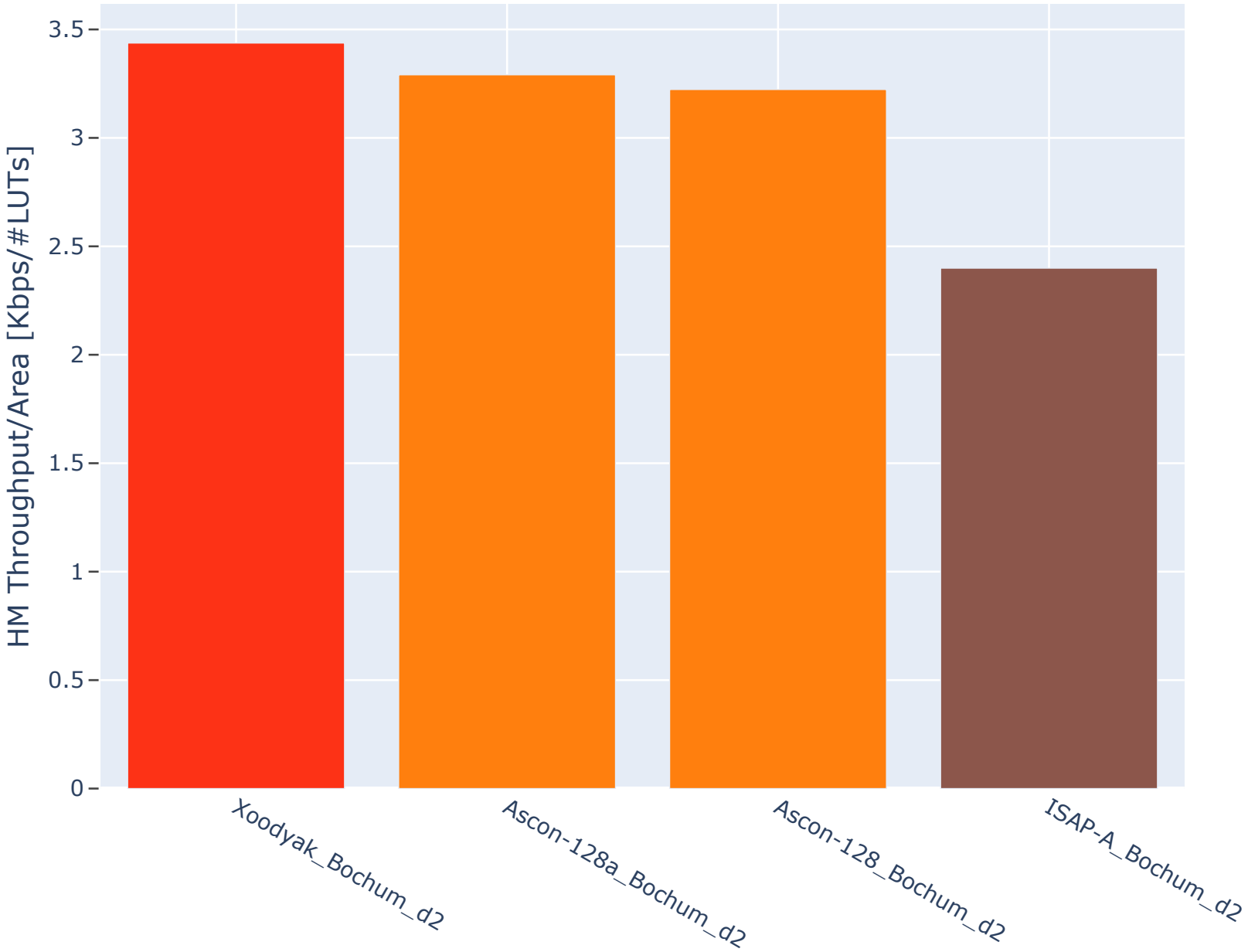
# Protected Order 2: Random bits/AD byte



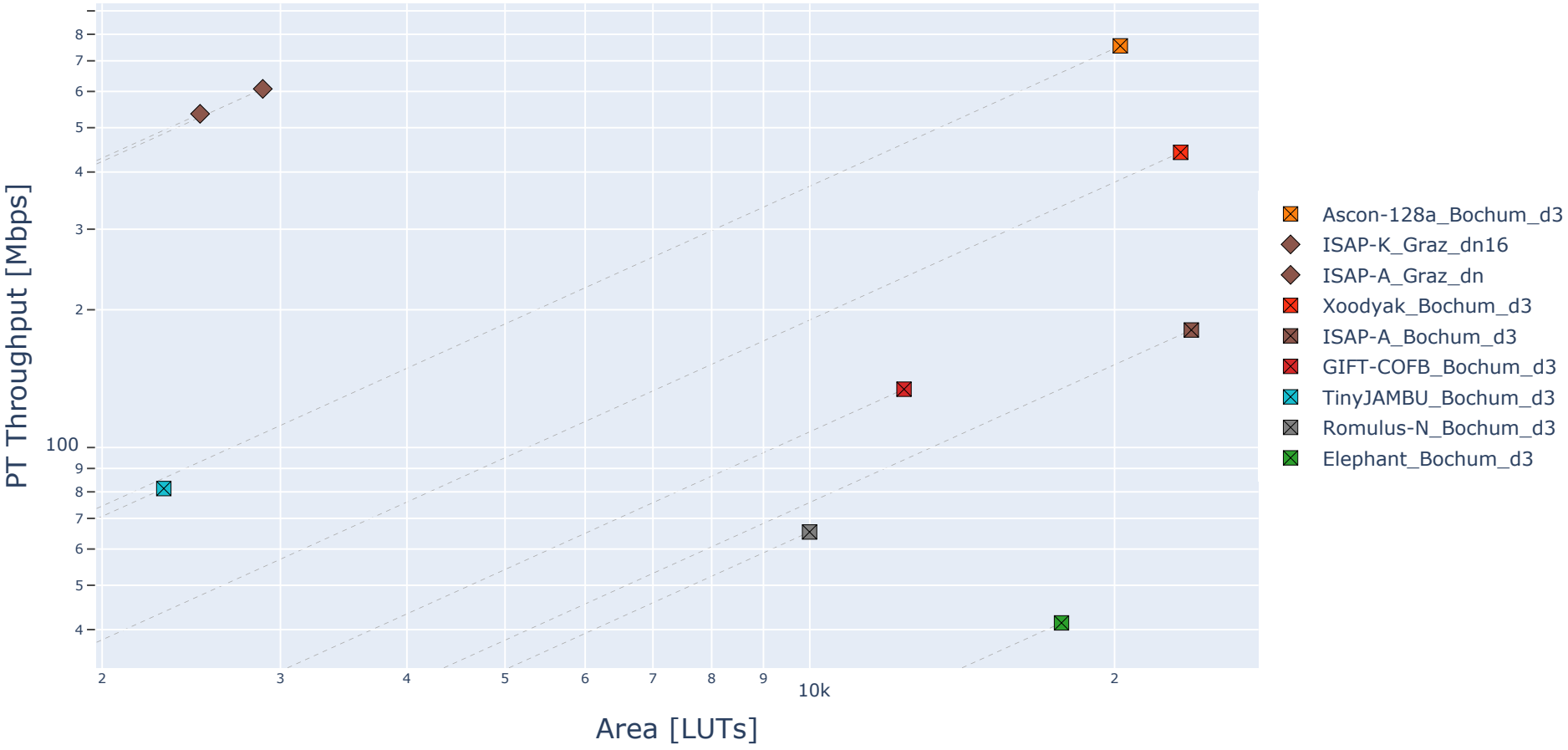
# Protected Order 2: Hashing Throughput vs. Area



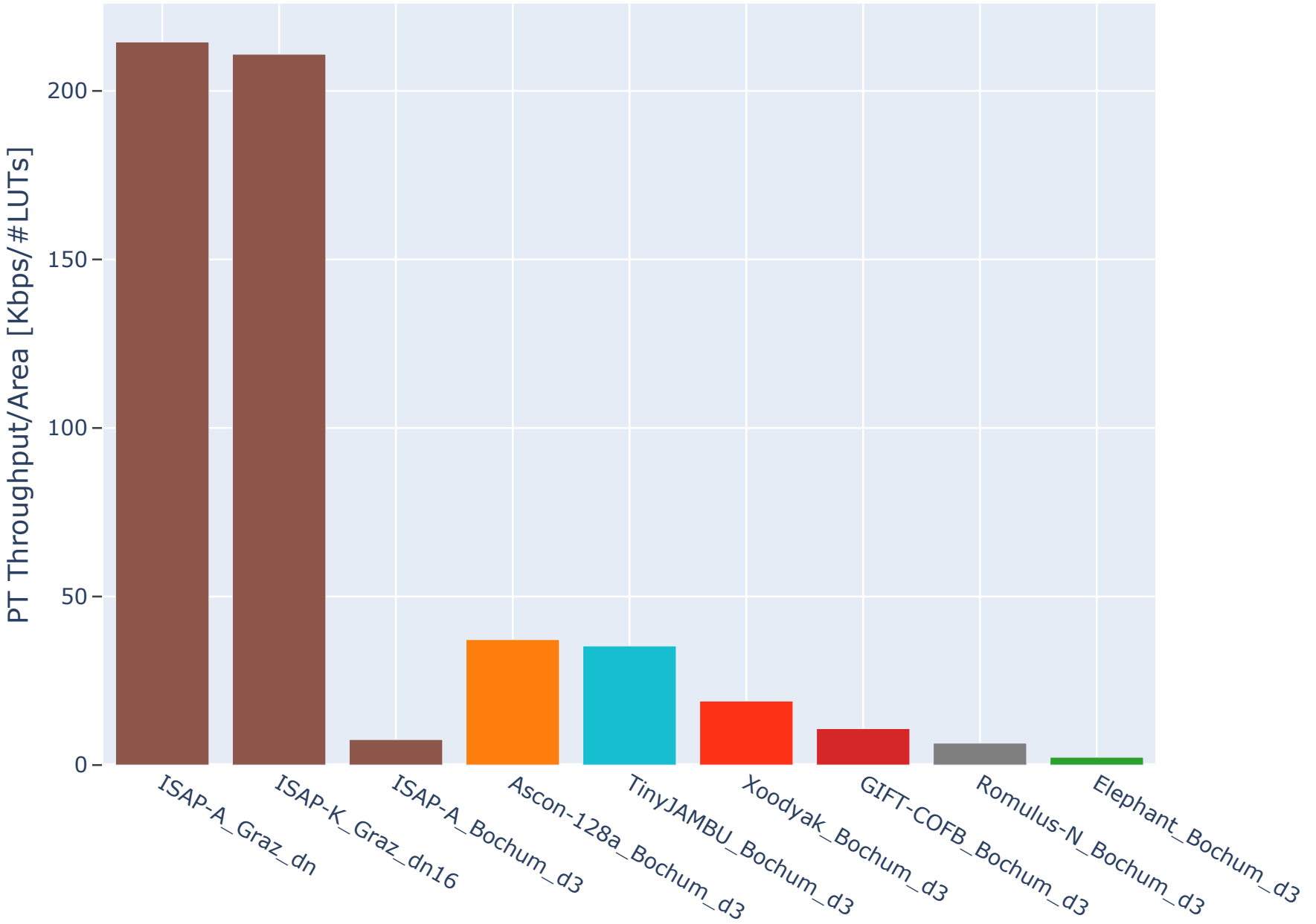
# Protected Order 2: Hashing Throughput/Area



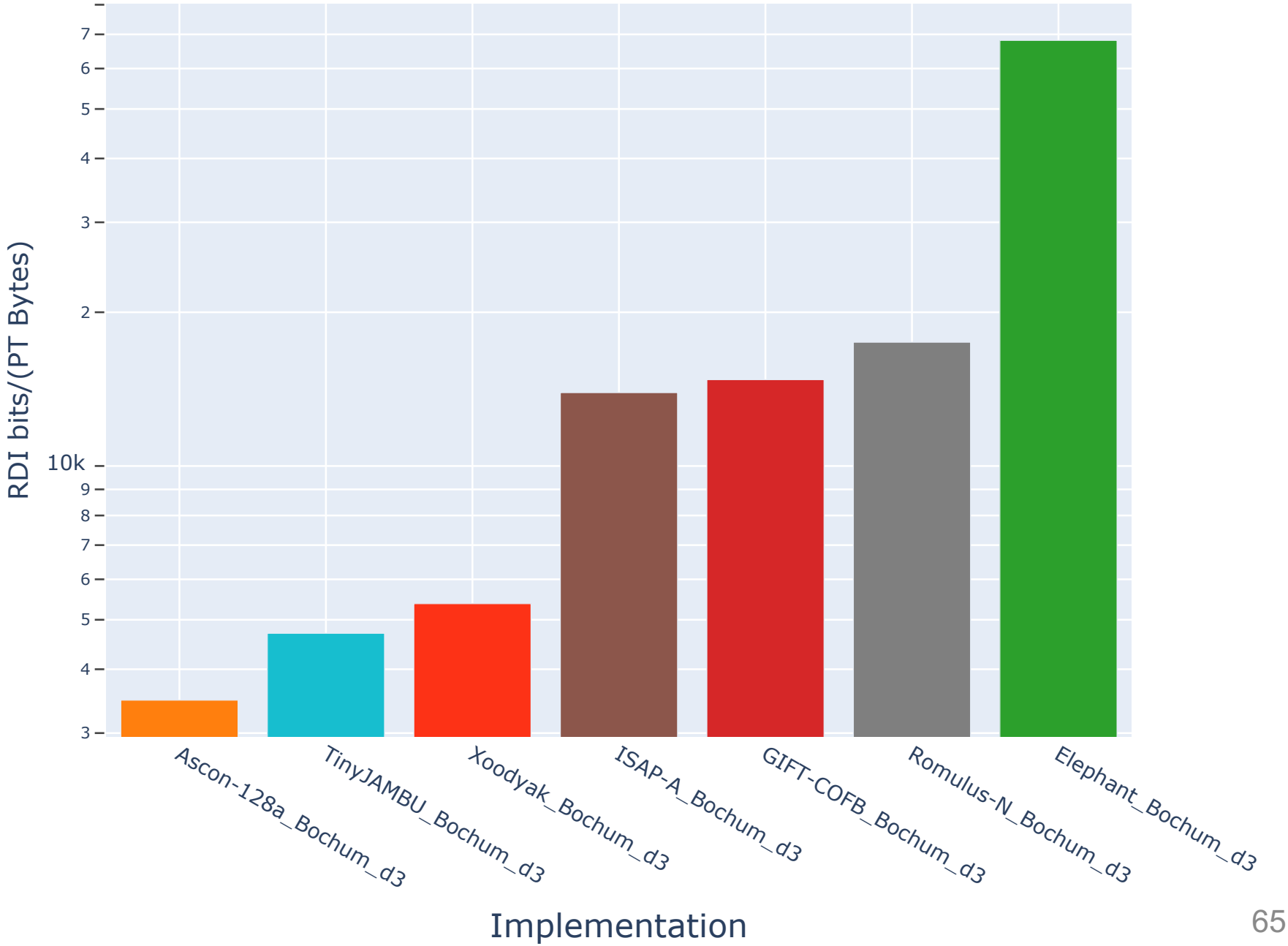
# Protected Order 3: PT Throughput vs. Area



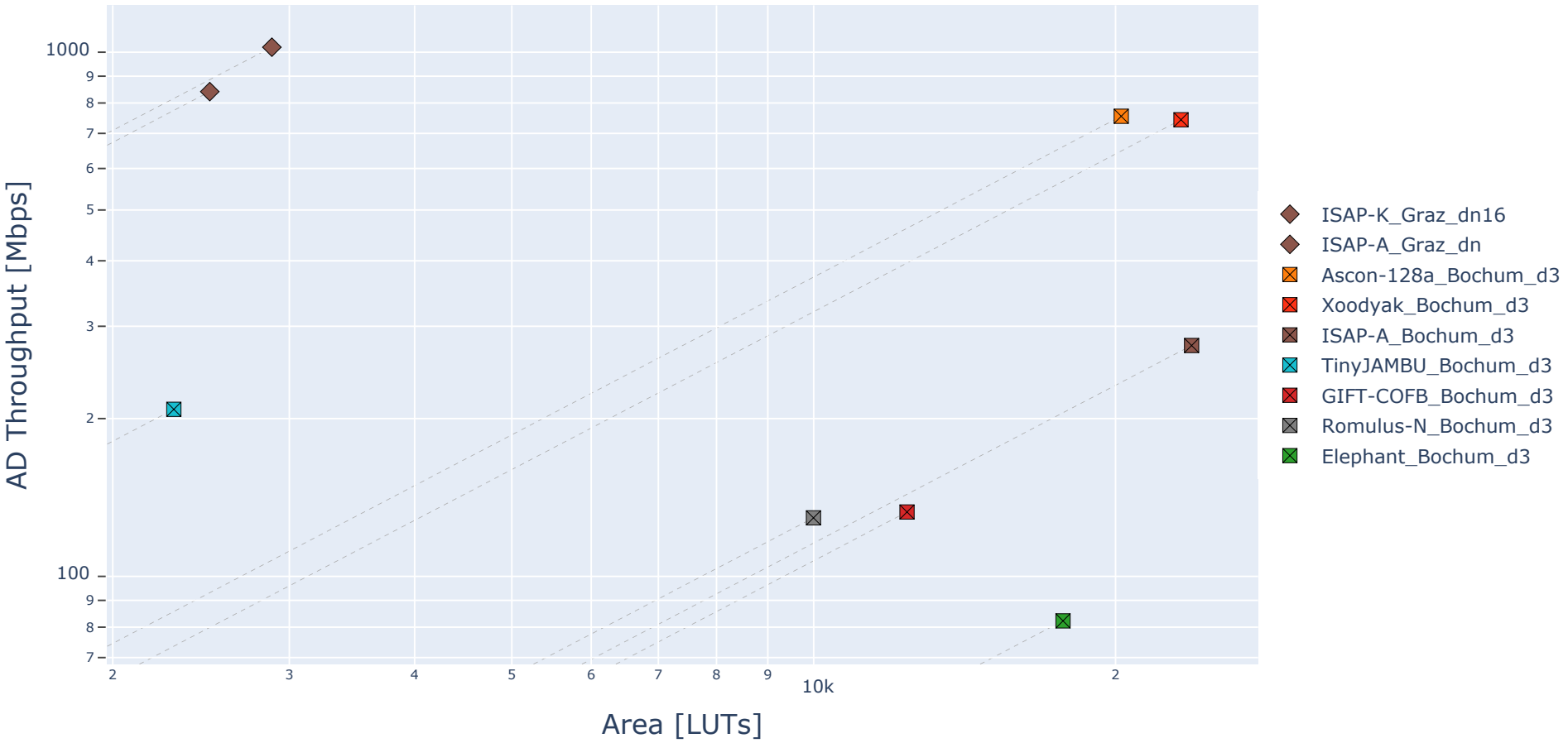
# Protected Order 3: PT Throughput/Area



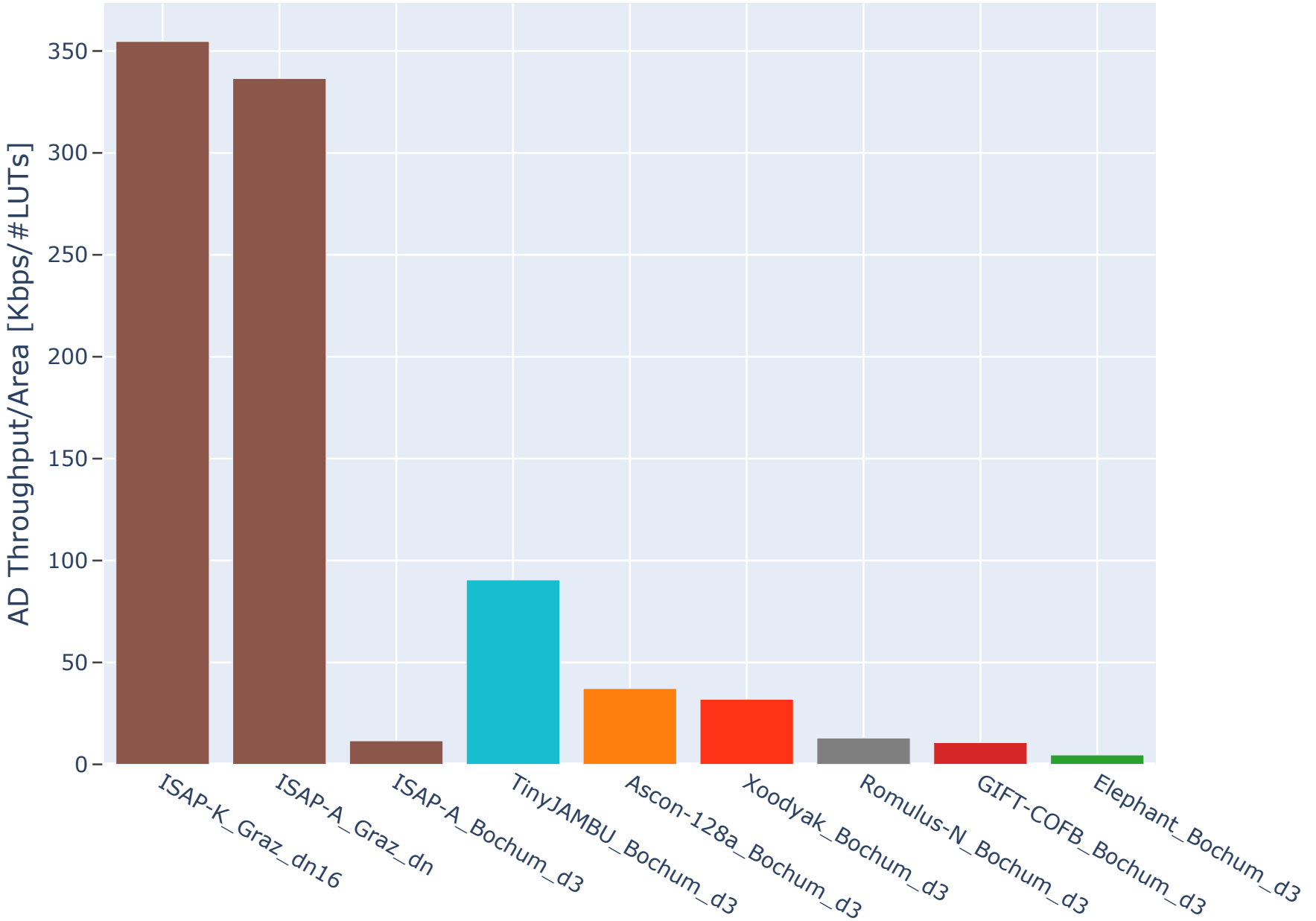
# Protected Order 3: Random bits/PT byte



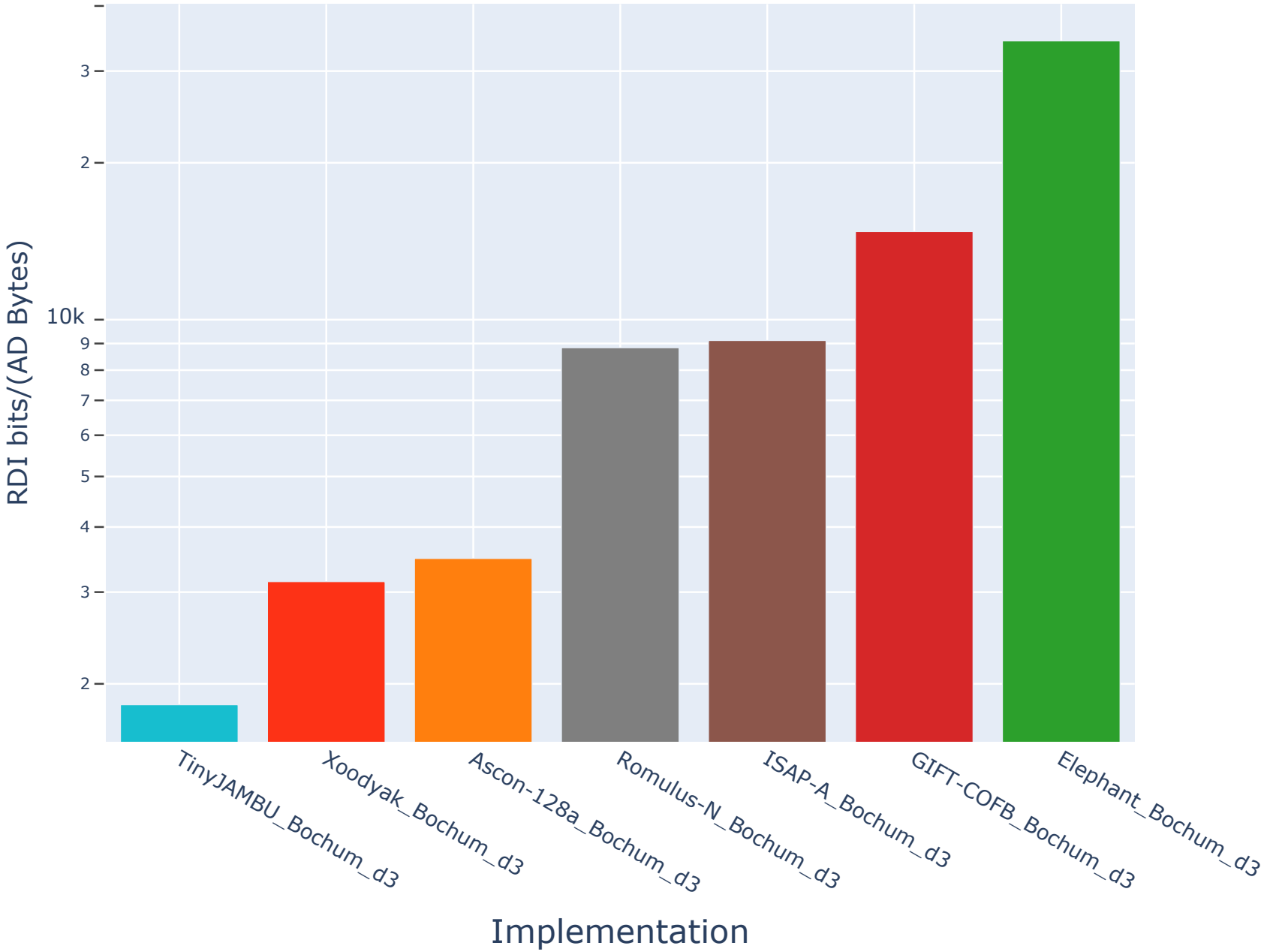
# Protected Order 3: AD Throughput vs. Area



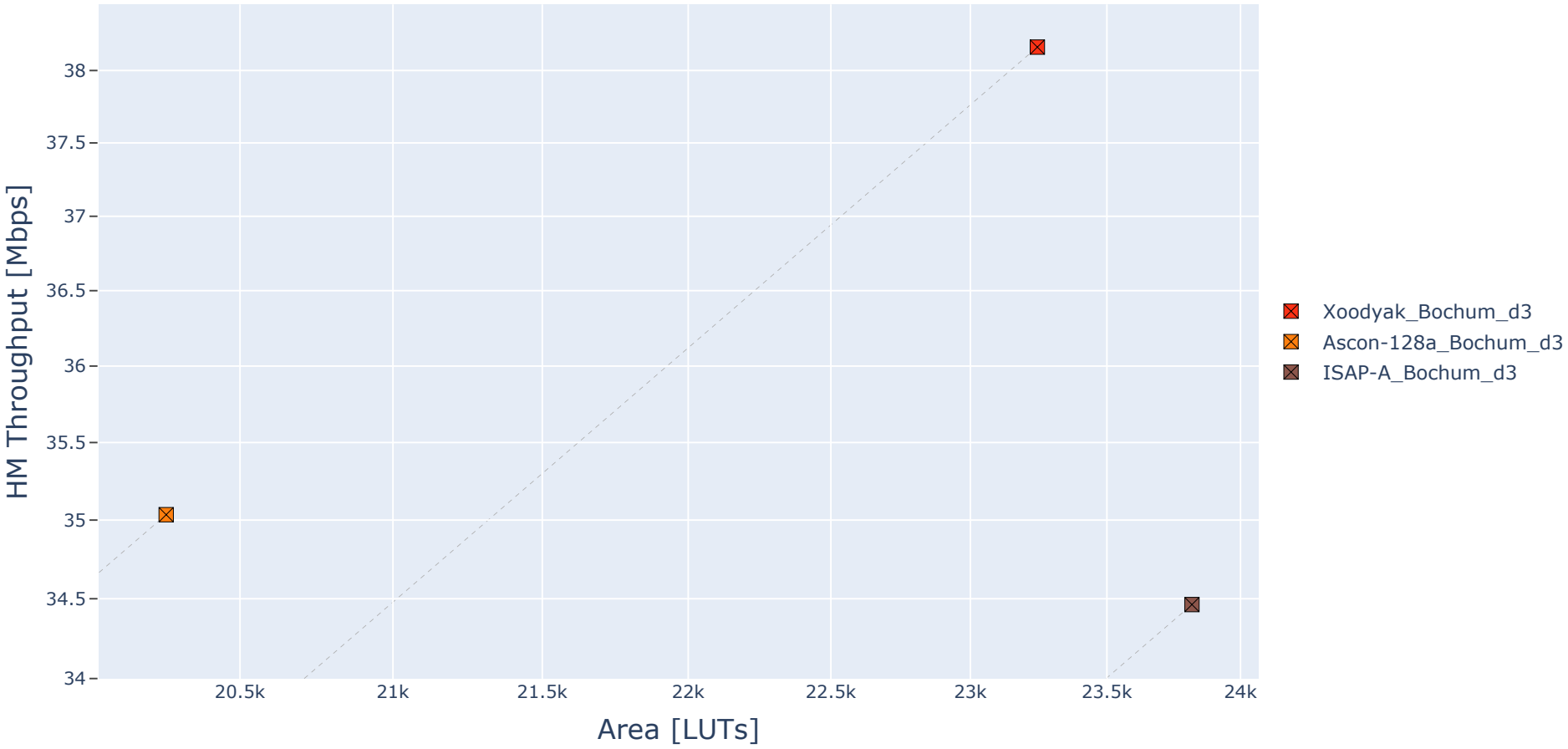
# Protected Order 3: AD Throughput/Area



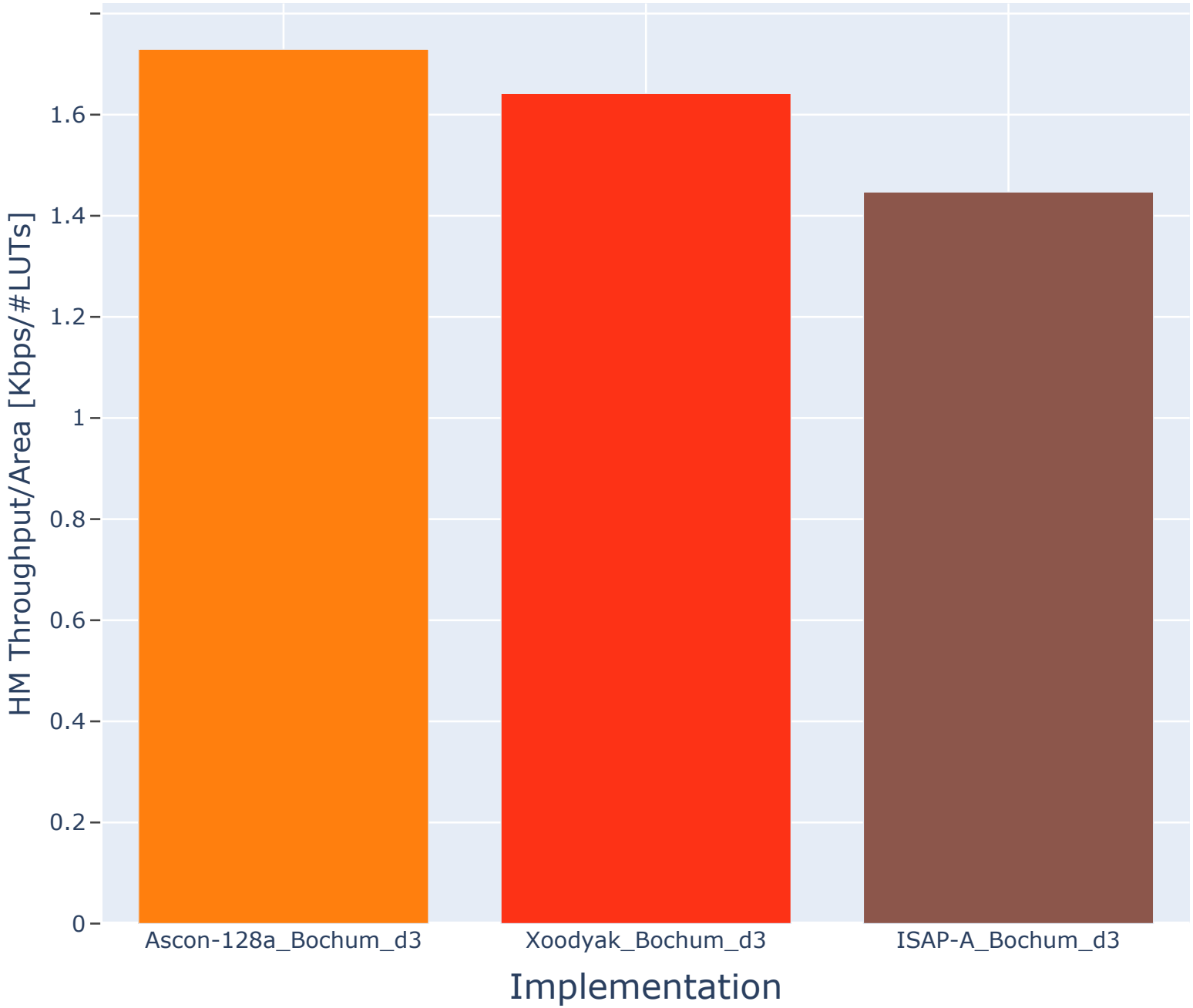
# Protected Order 3: Random bits/AD byte



# Protected Order 3: Hashing Throughput vs. Area



# Protected Order 3: Hashing Throughput/Area



# Conclusions

# Conclusions

- Protected hardware implementations of 9 out of 10 finalists
- Most of them generated automatically
- Most of them pass basic leakage assessment tests (any required corrections are not likely to affect results of benchmarking)

- Protected software implementations of 5 out of 10 finalists
- Two implementations fail a basic leakage assessment test
- One of the remaining ones uses a mode-level protection difficult to verify experimentally

# Conclusions

- Hardware benchmarking results demonstrate advantages of the following candidates:
- **Ascon and Xoodyak:**
  - High speed
  - High throughput/area ratio
  - Moderate randomness requirements
  - Support for hashing
- **TinyJAMBU:**
  - Low area
  - High throughput/area ratio
  - Moderate randomness requirements
- **ISAP:**
  - Mode-level protection against arbitrary-level DPA (no masking)
  - High throughput/area ratio among protected designs
  - Support for hashing

# Website

Lightweight Cryptography in Hardware and Embedded Systems

**https://cryptography.gmu.edu/athena/index.php?id=LWC**

## Evaluation of Finalists in the NIST LWC Process

- Summary of Results
- Assignments, Commitments, and Reports
- Side-Channel Security Evaluation Labs
- Protected Implementations
- Calls for Implementations & Labs
- Documentation
- Code (Development Package)
- Unprotected Implementations

# Thank you!

Questions?

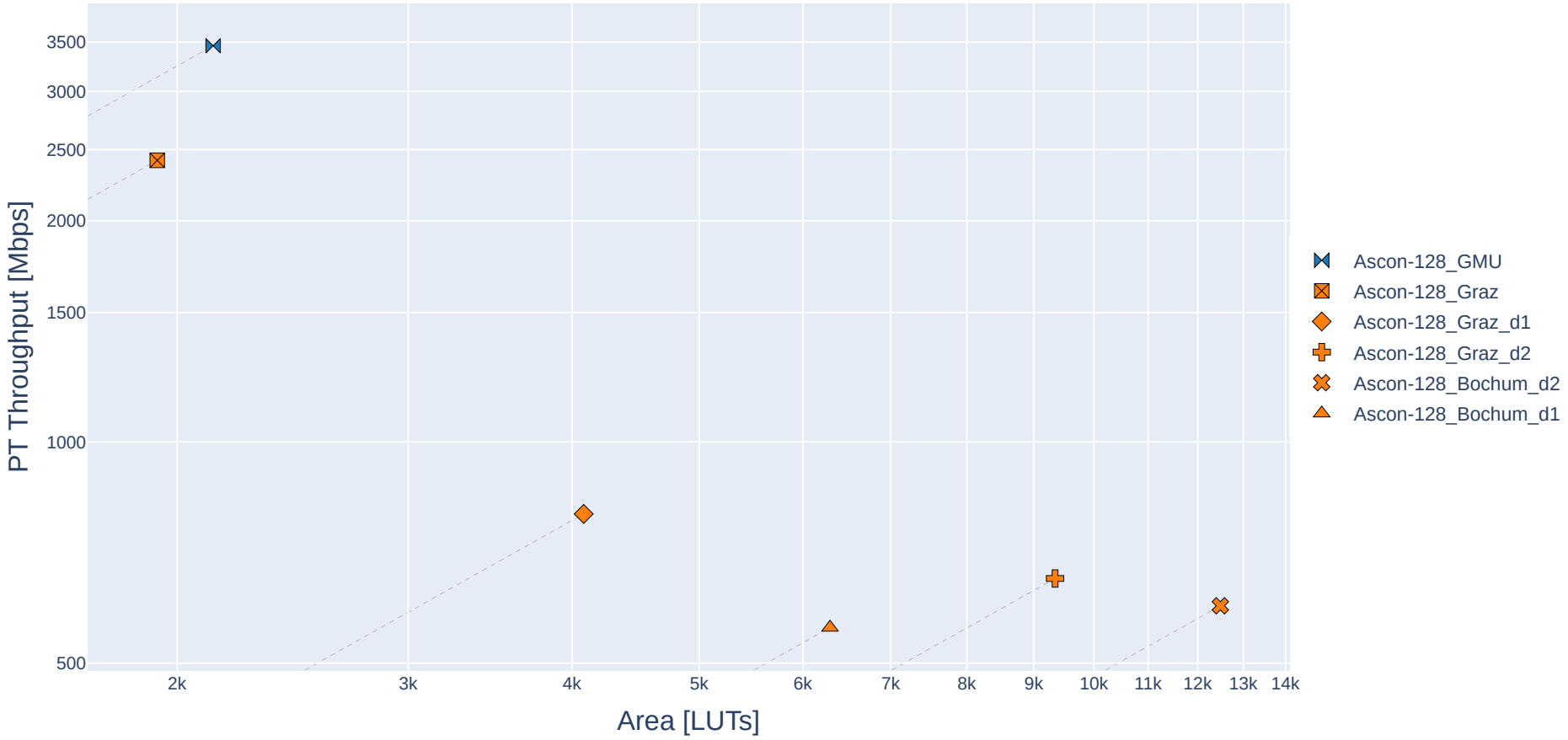
Comments?

Suggestions?

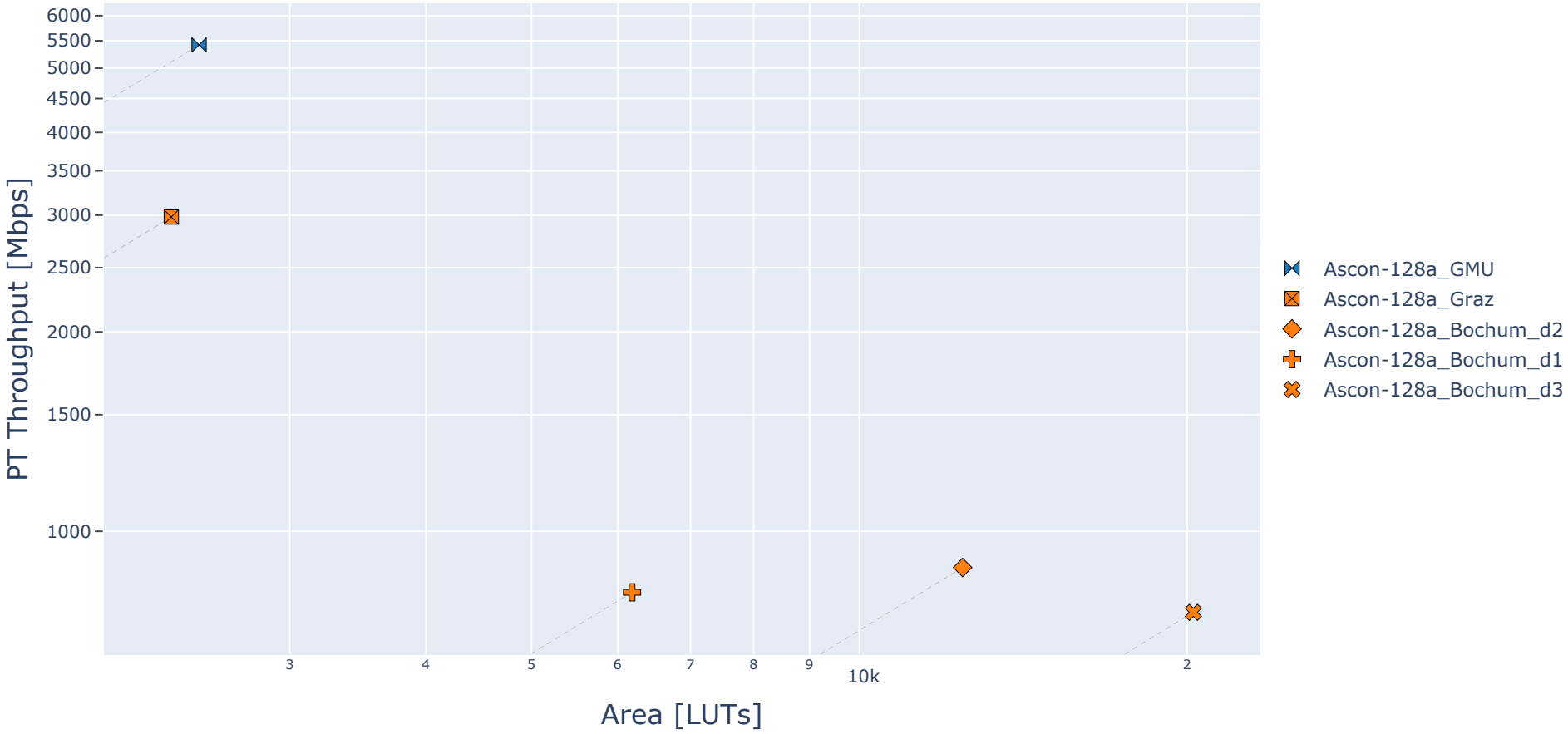
# Appendix A

# Remaining Graphs

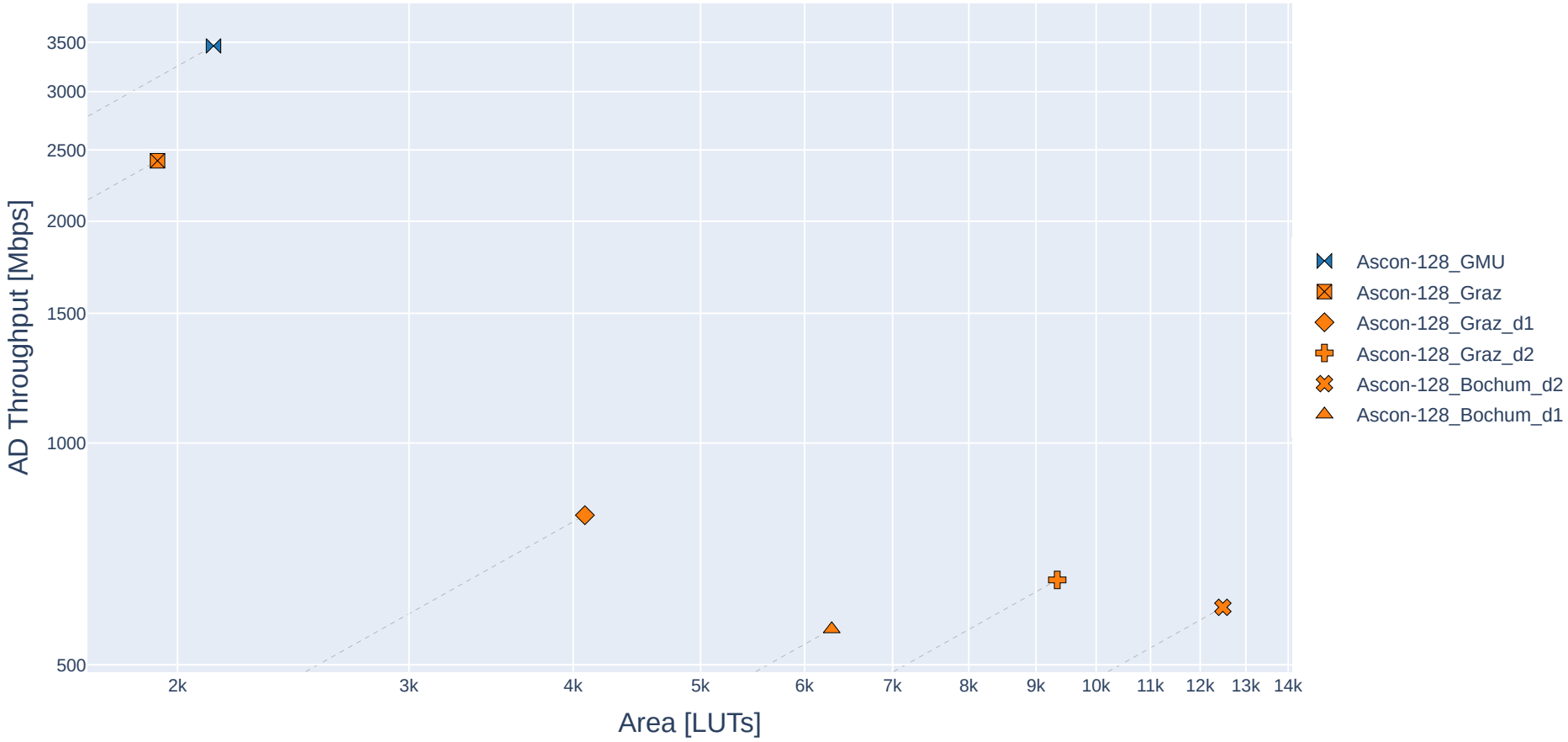
# Ascon-128: PT Throughput vs. Area



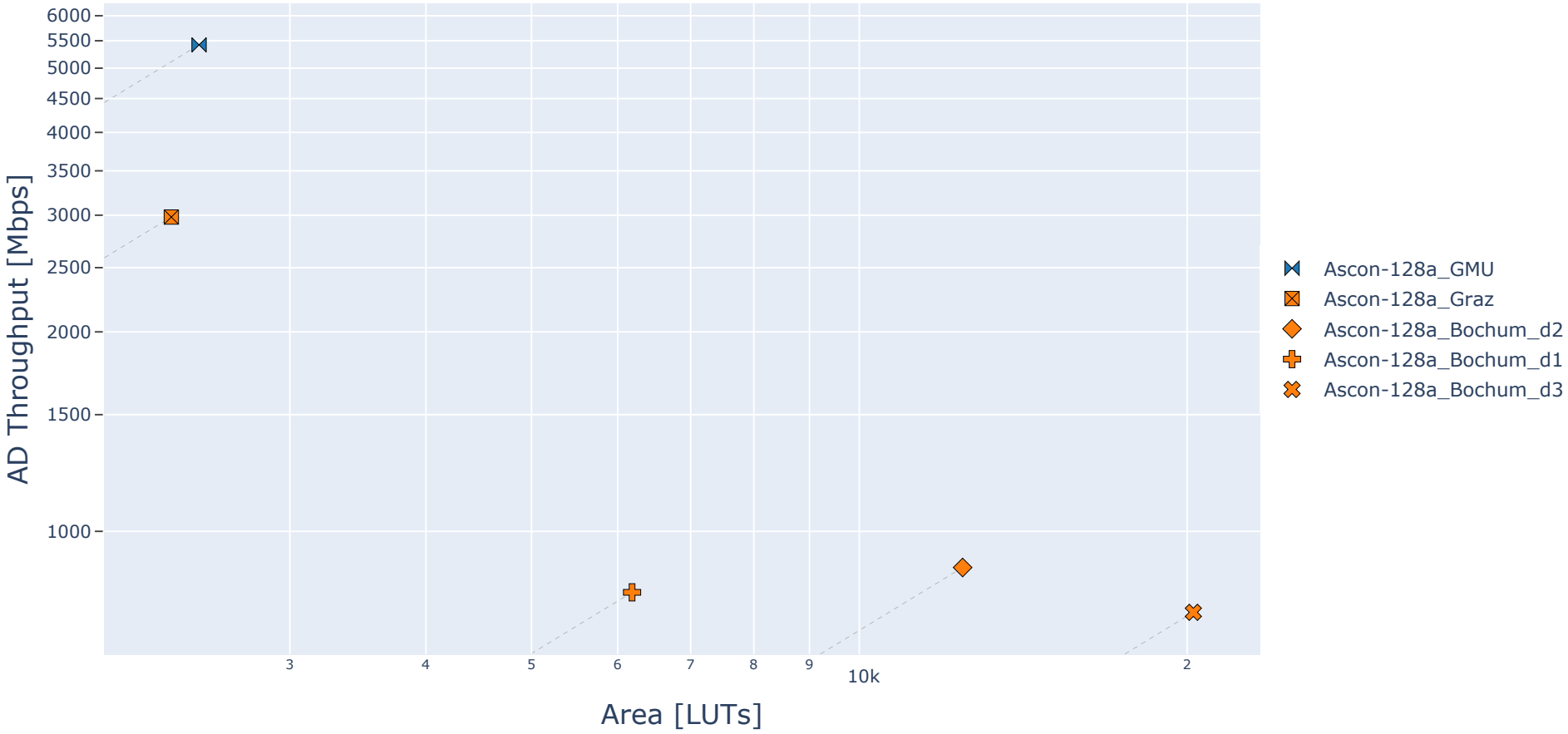
# Ascon-128a: PT Throughput vs. Area



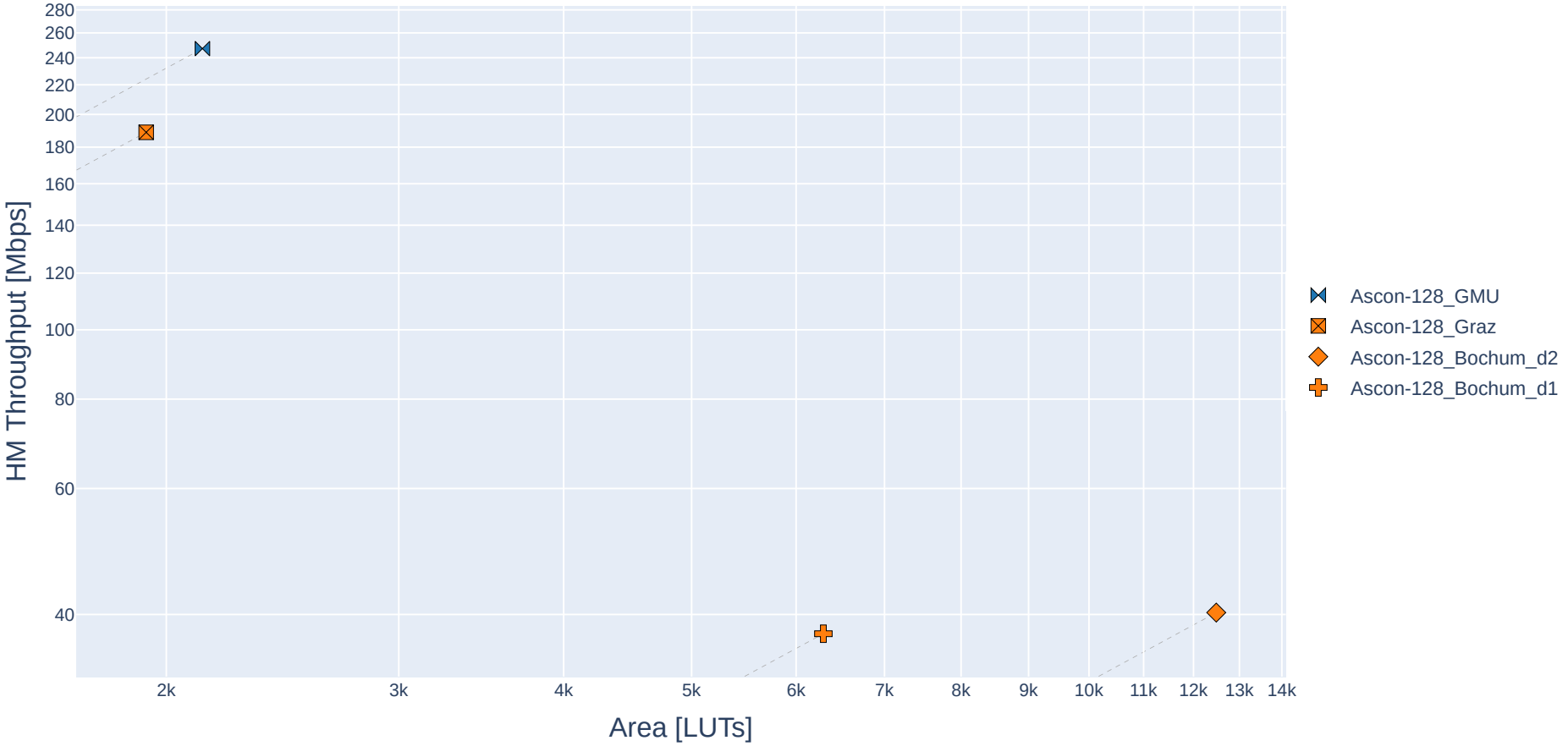
# Ascon-128: AD Throughput vs. Area



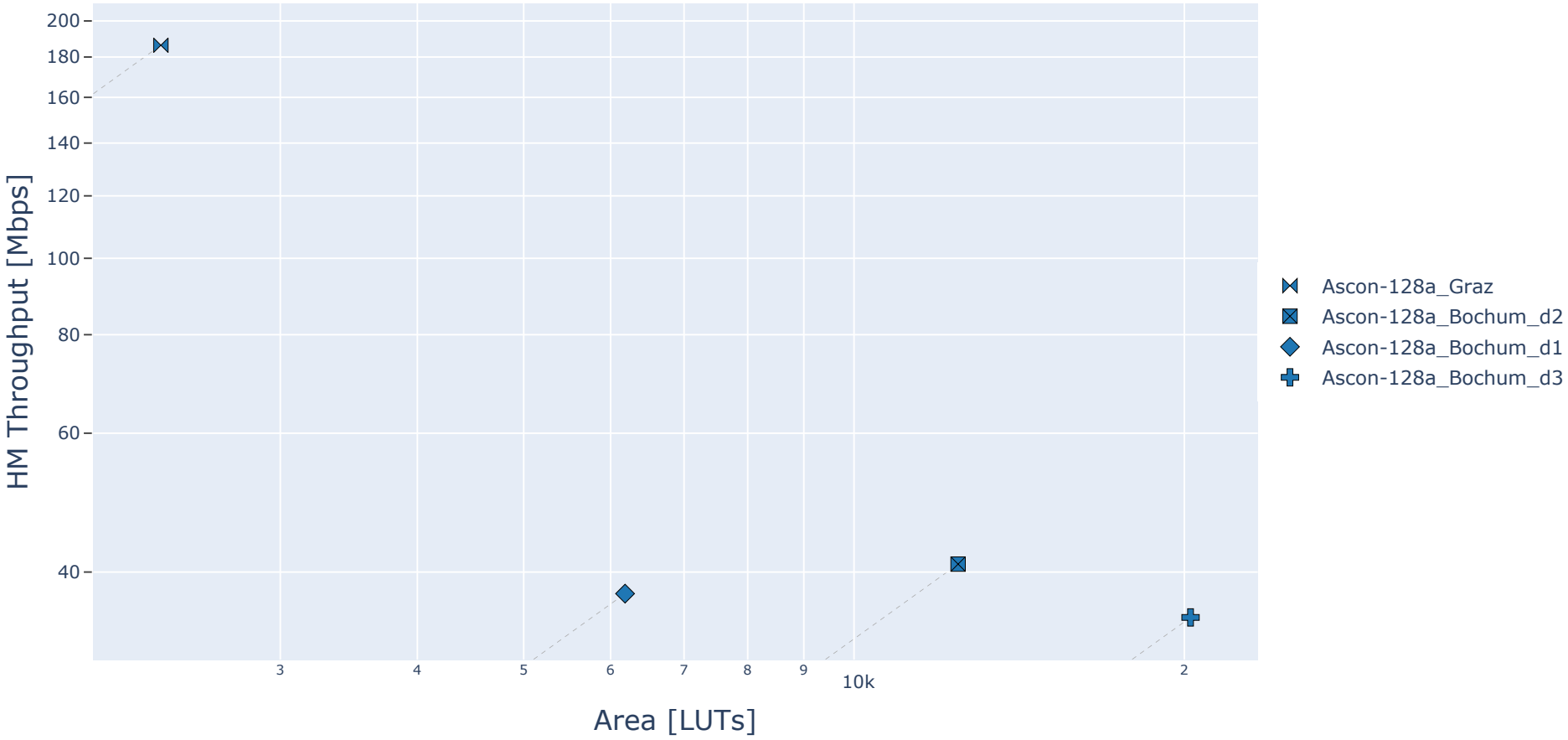
# Ascon-128a: AD Throughput vs. Area



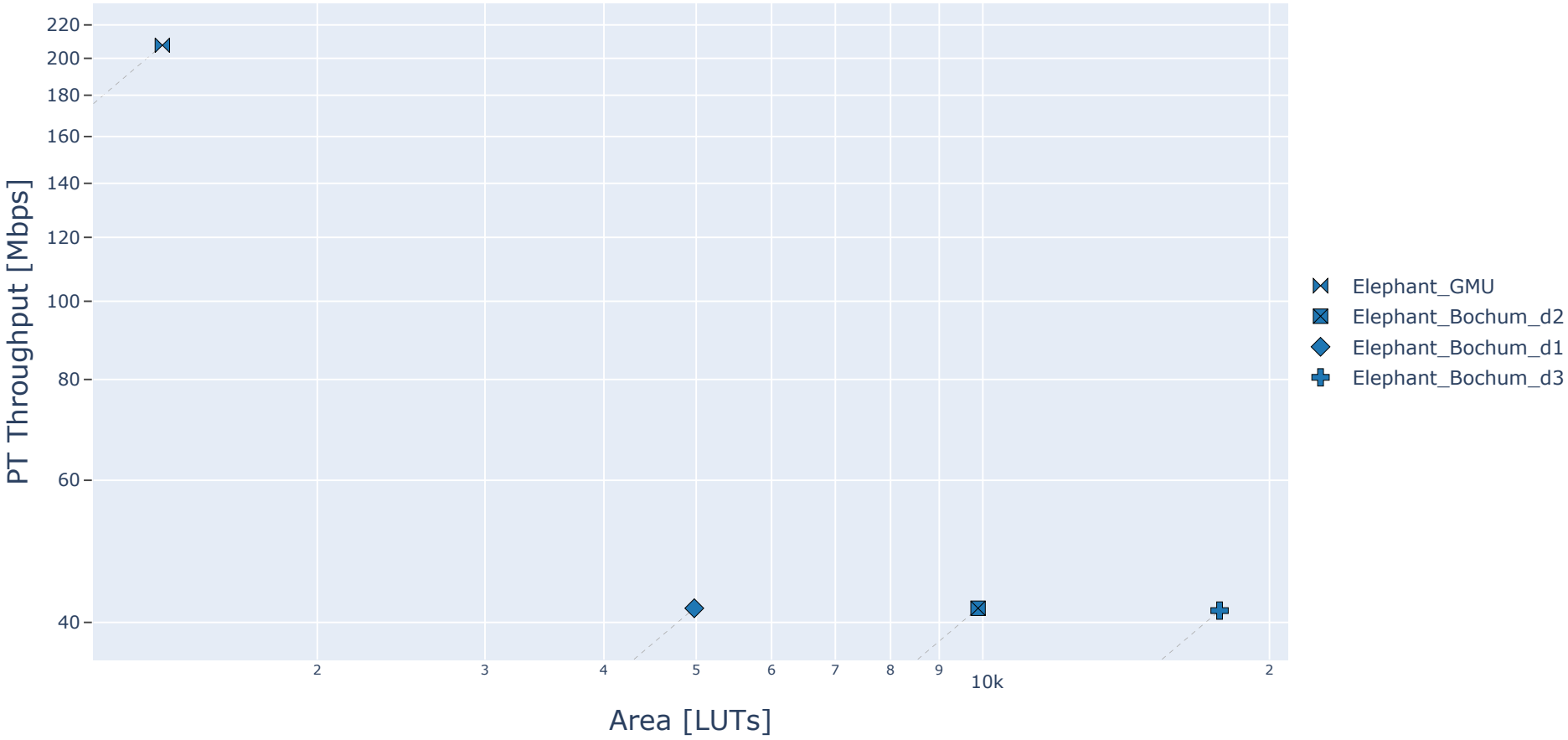
# Ascon-128: Hashing Throughput vs. Area



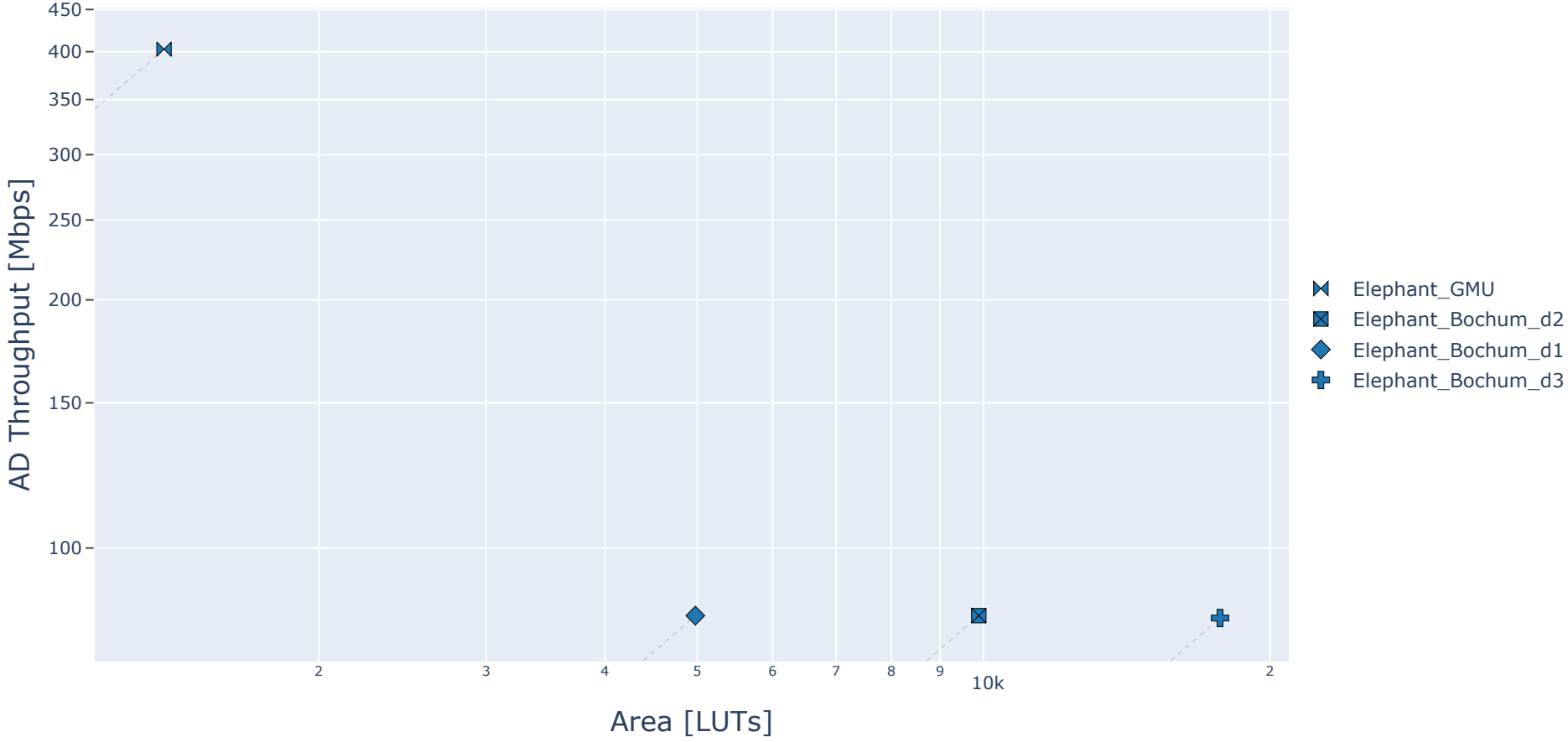
# Ascon-128a: Hashing Throughput vs. Area



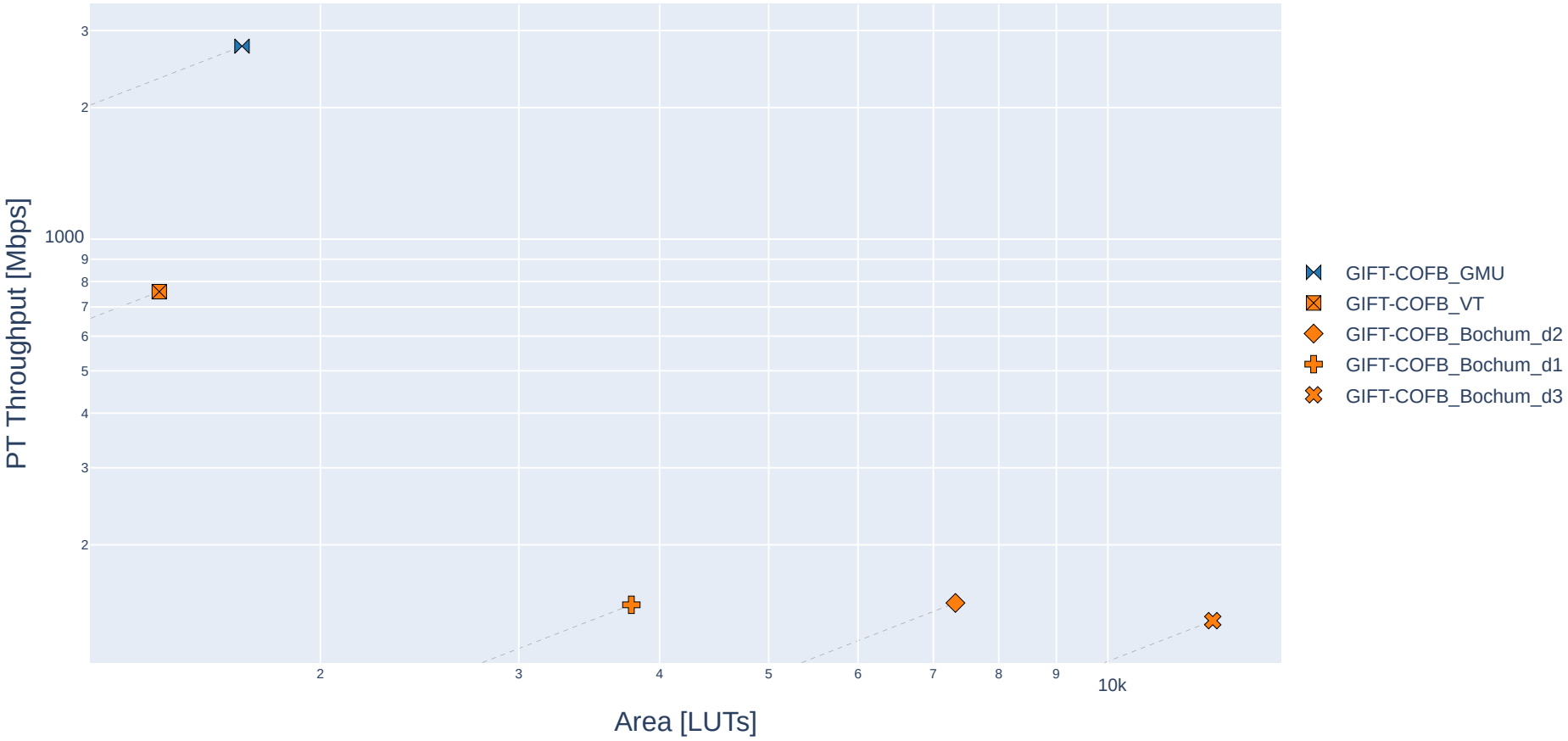
# Elephant: PT Throughput vs. Area



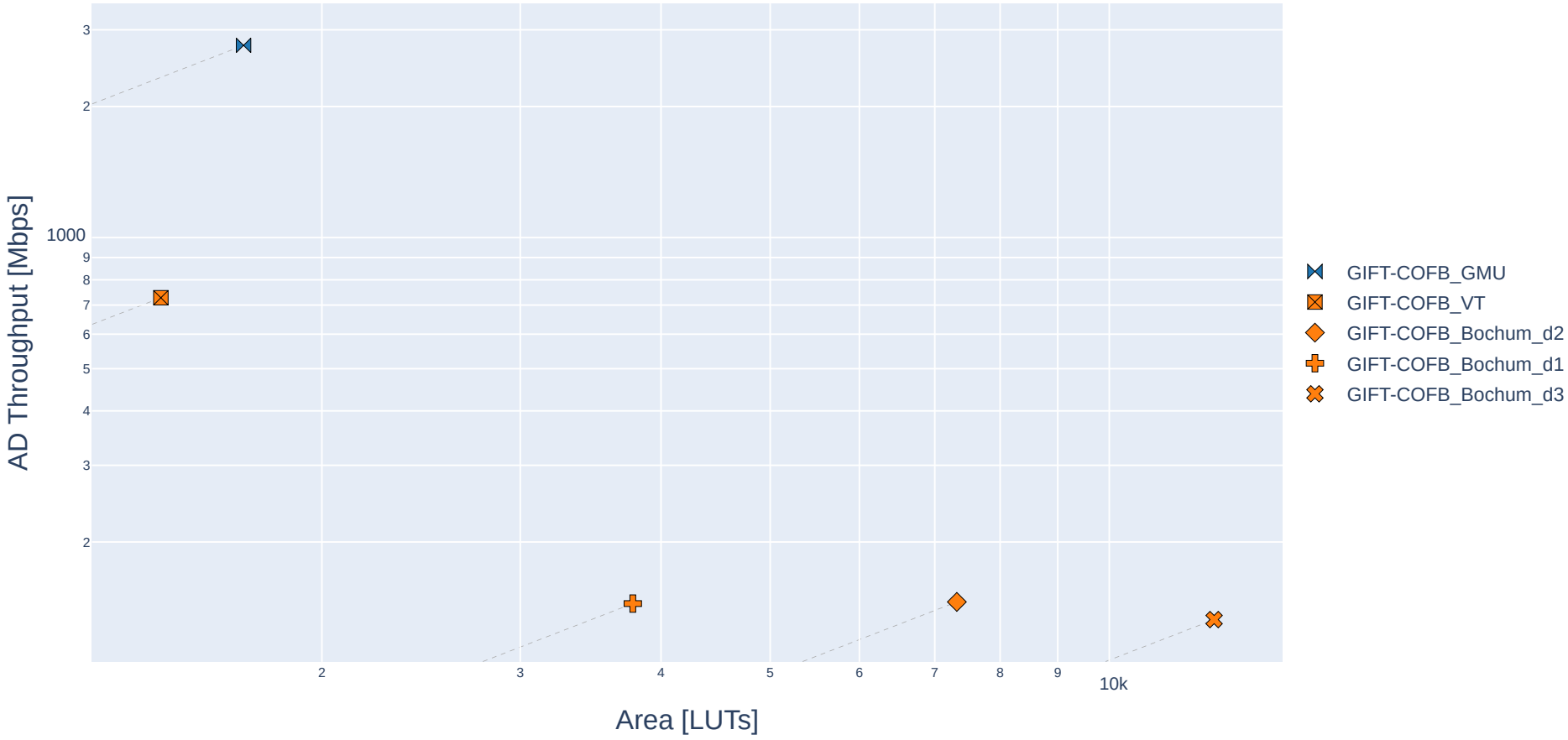
# Elephant: AD Throughput vs. Area



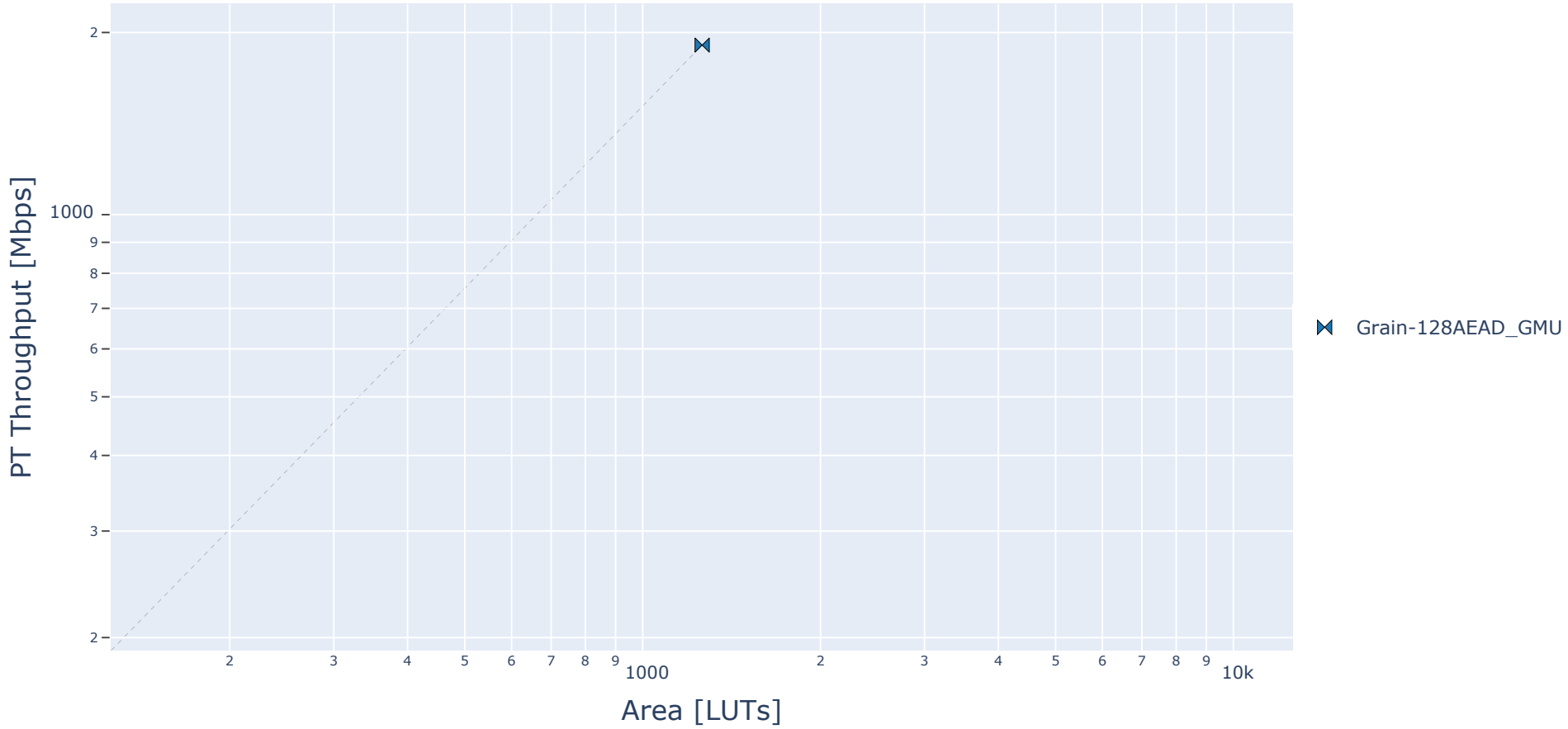
# GIFT-COFB: PT Throughput vs. Area



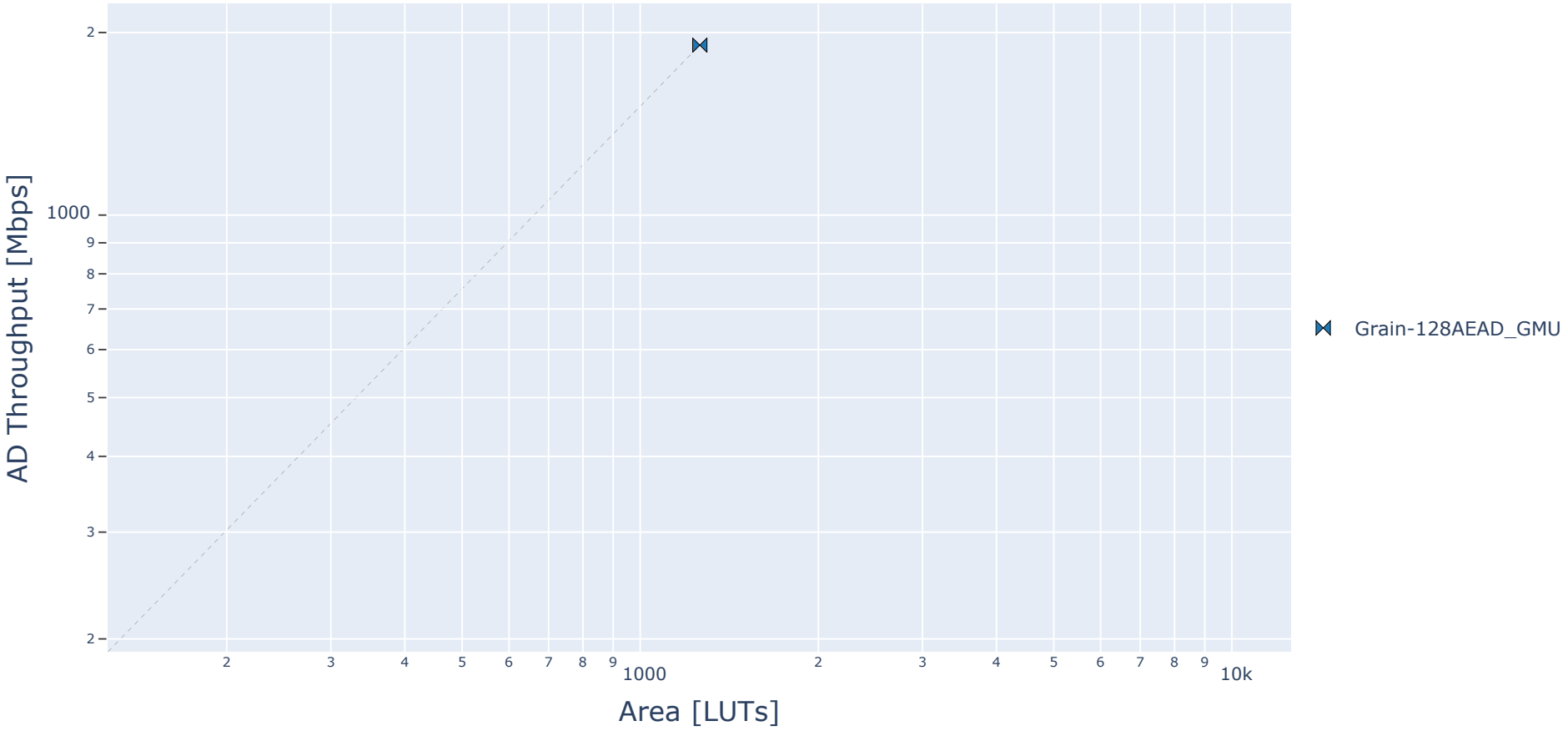
# GIFT-COFB: AD Throughput vs. Area



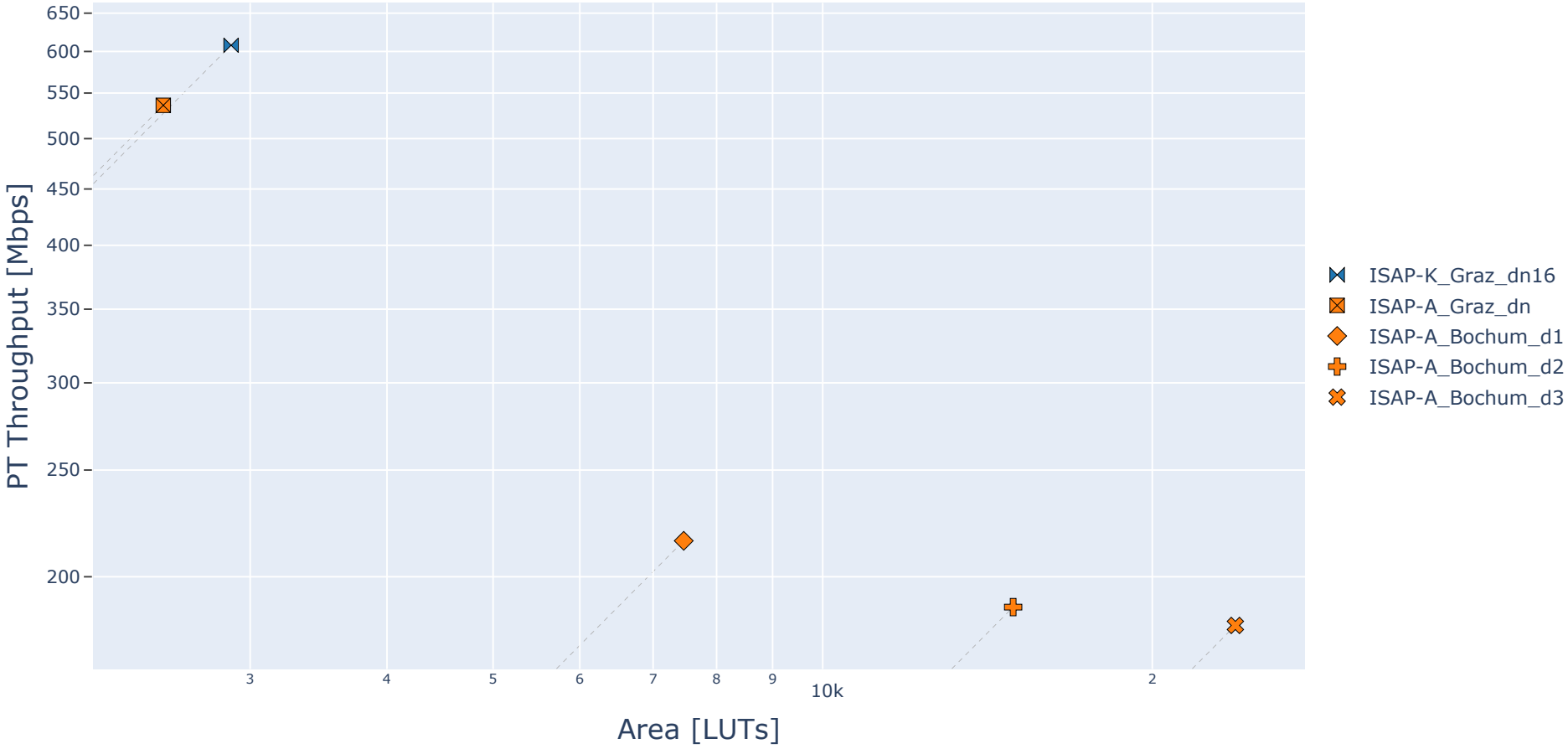
# Grain-128AEAD: PT Throughput vs. Area



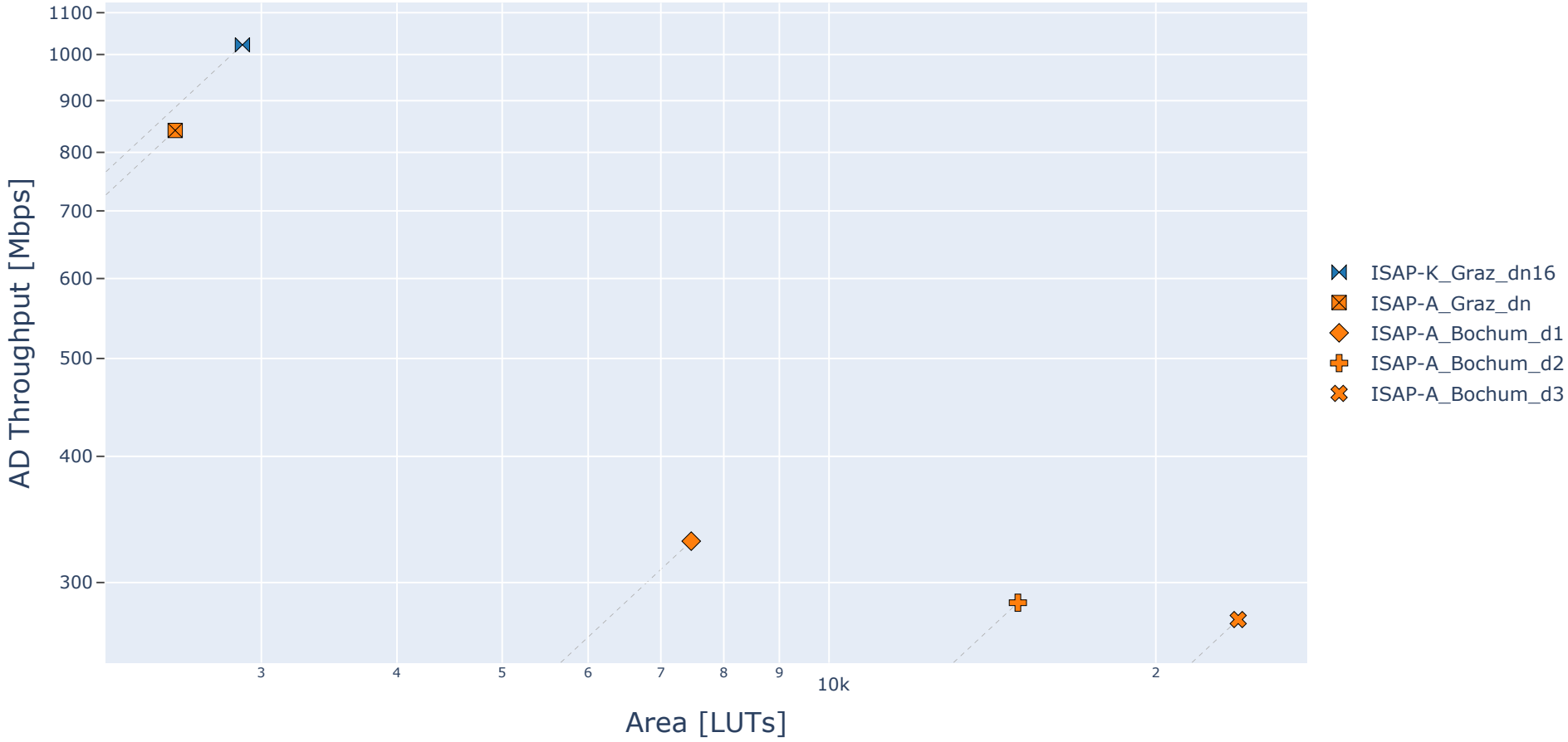
# Grain-128AEAD: AD Throughput vs. Area



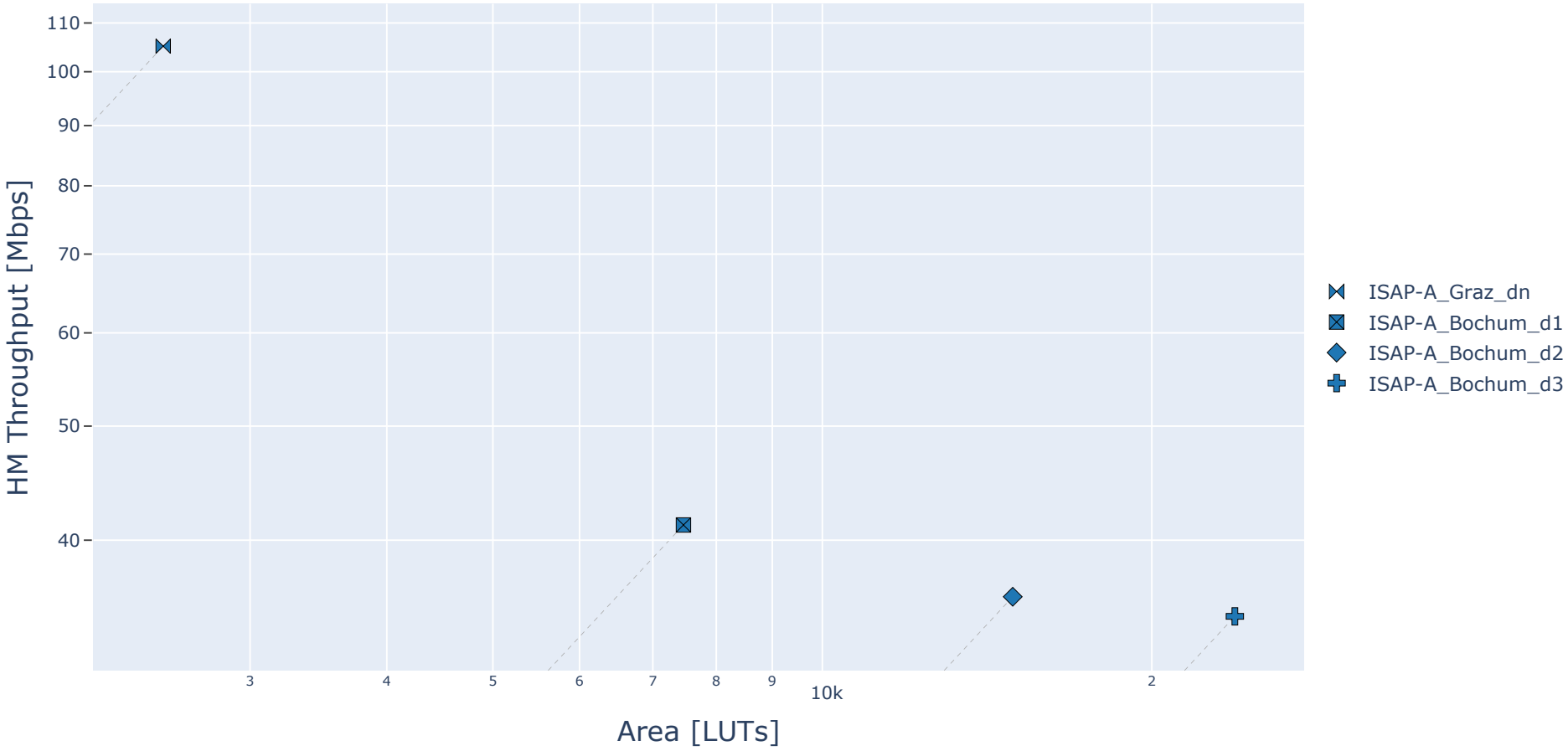
# ISAP: PT Throughput vs. Area



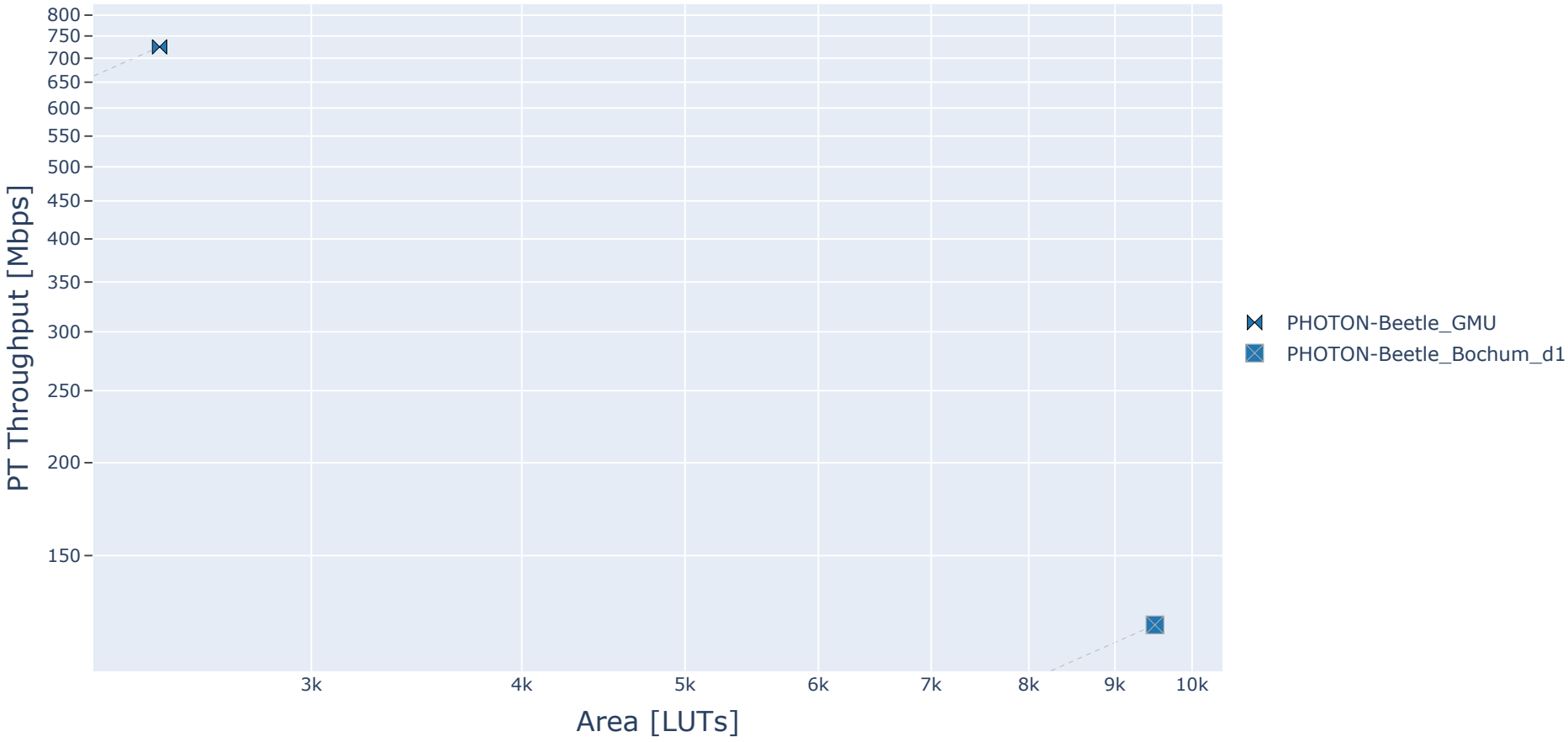
# ISAP: AD Throughput vs. Area



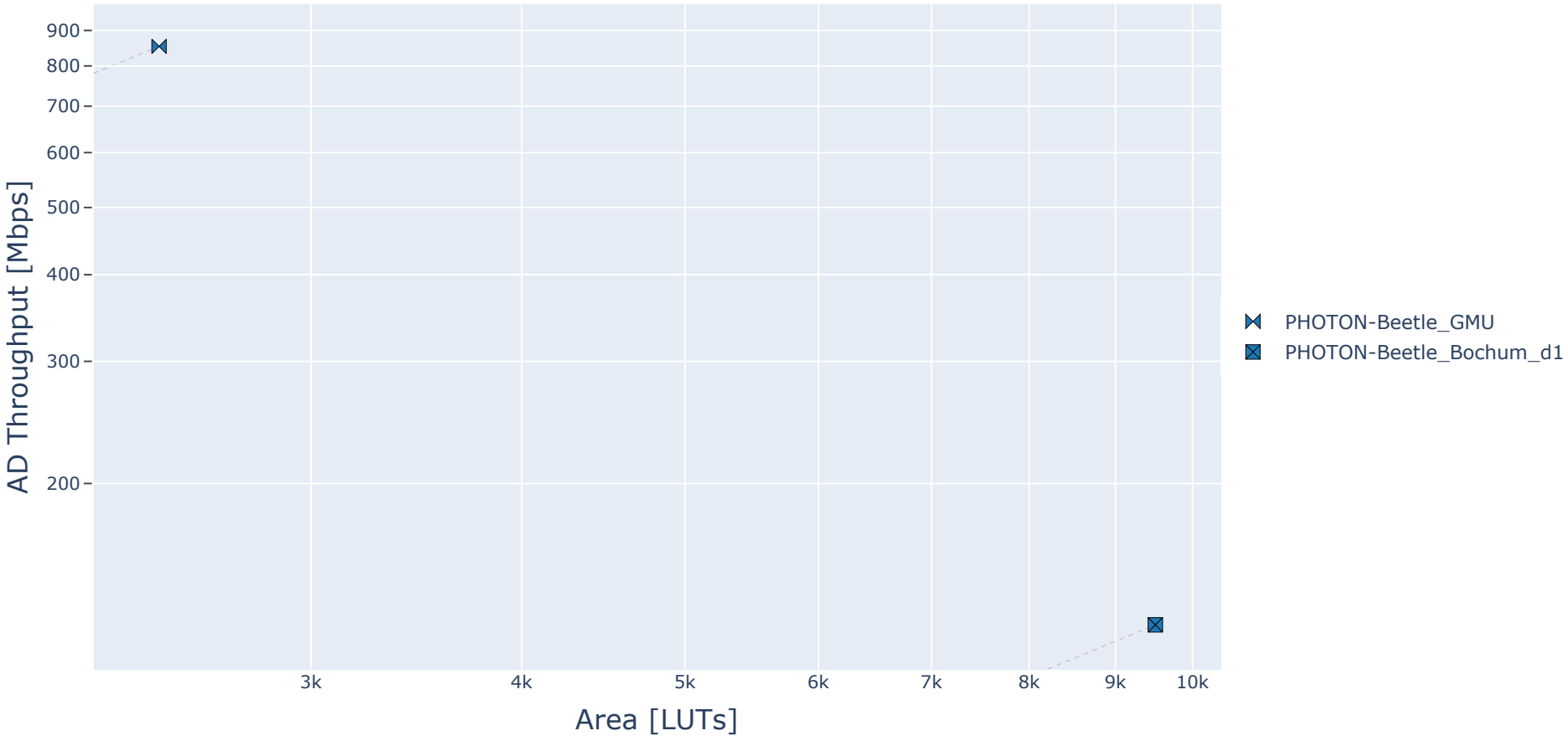
# ISAP: Hashing Throughput vs. Area



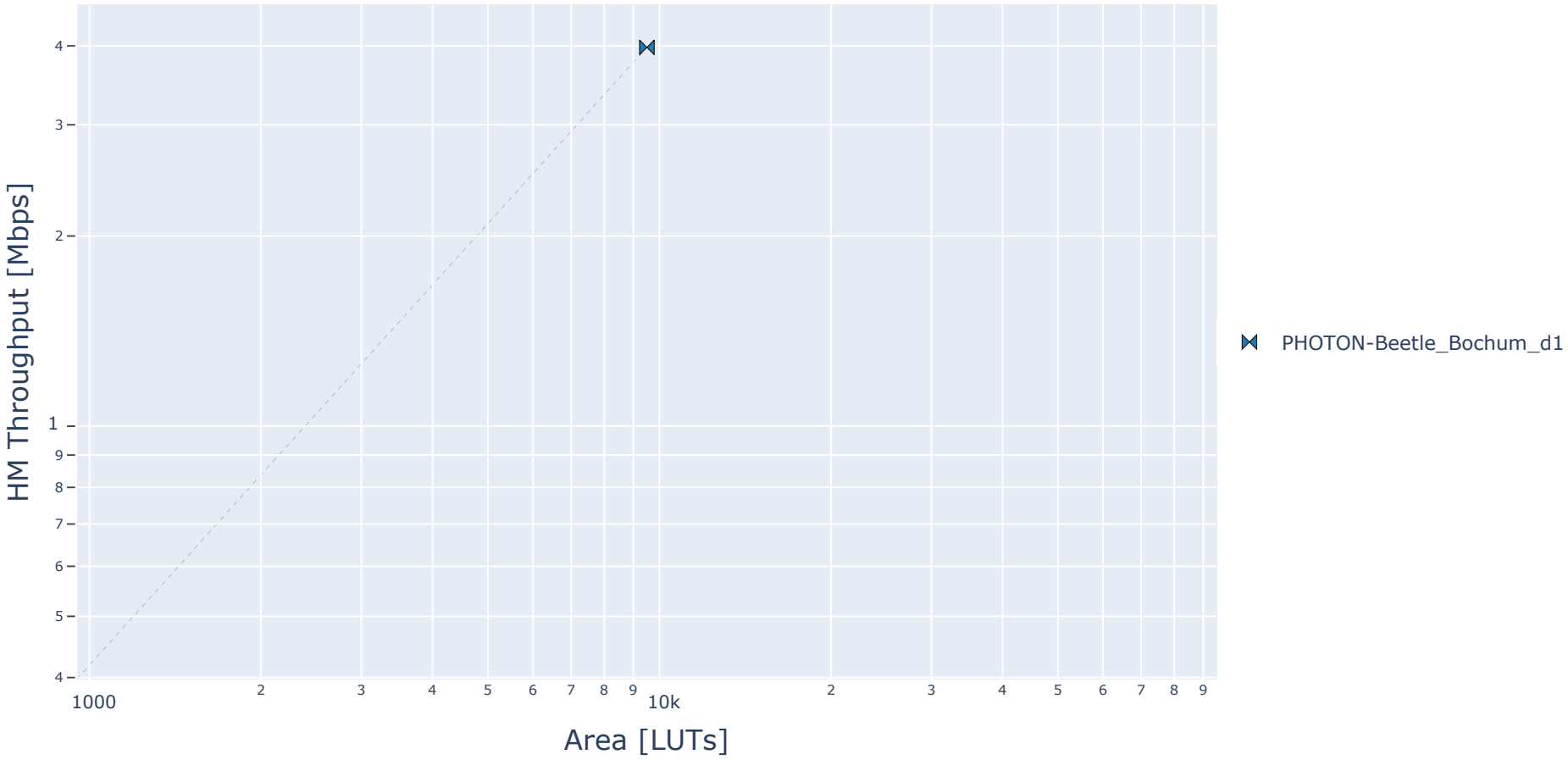
# PHOTON-Beetle: PT Throughput vs. Area



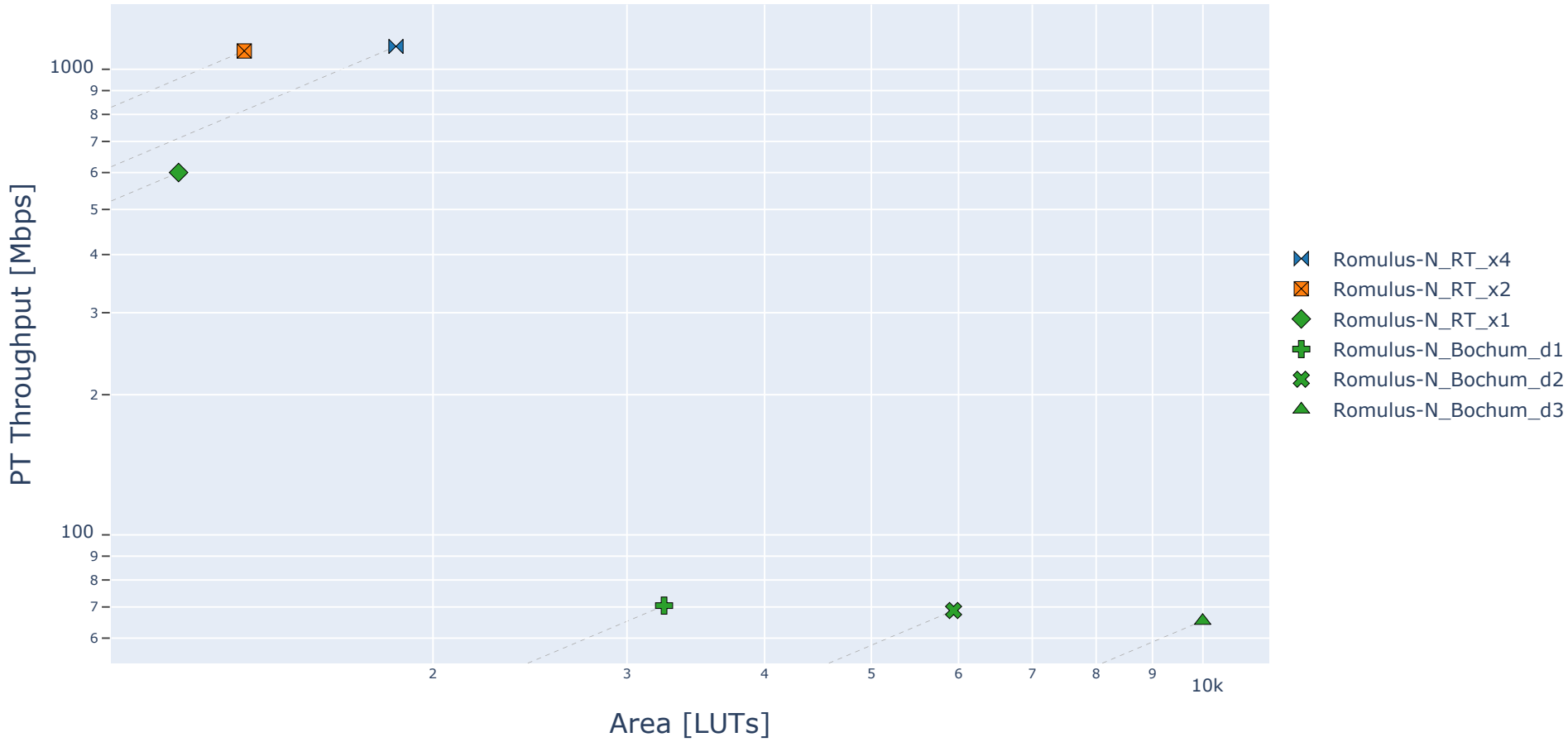
# PHOTON-Beetle: AD Throughput vs. Area



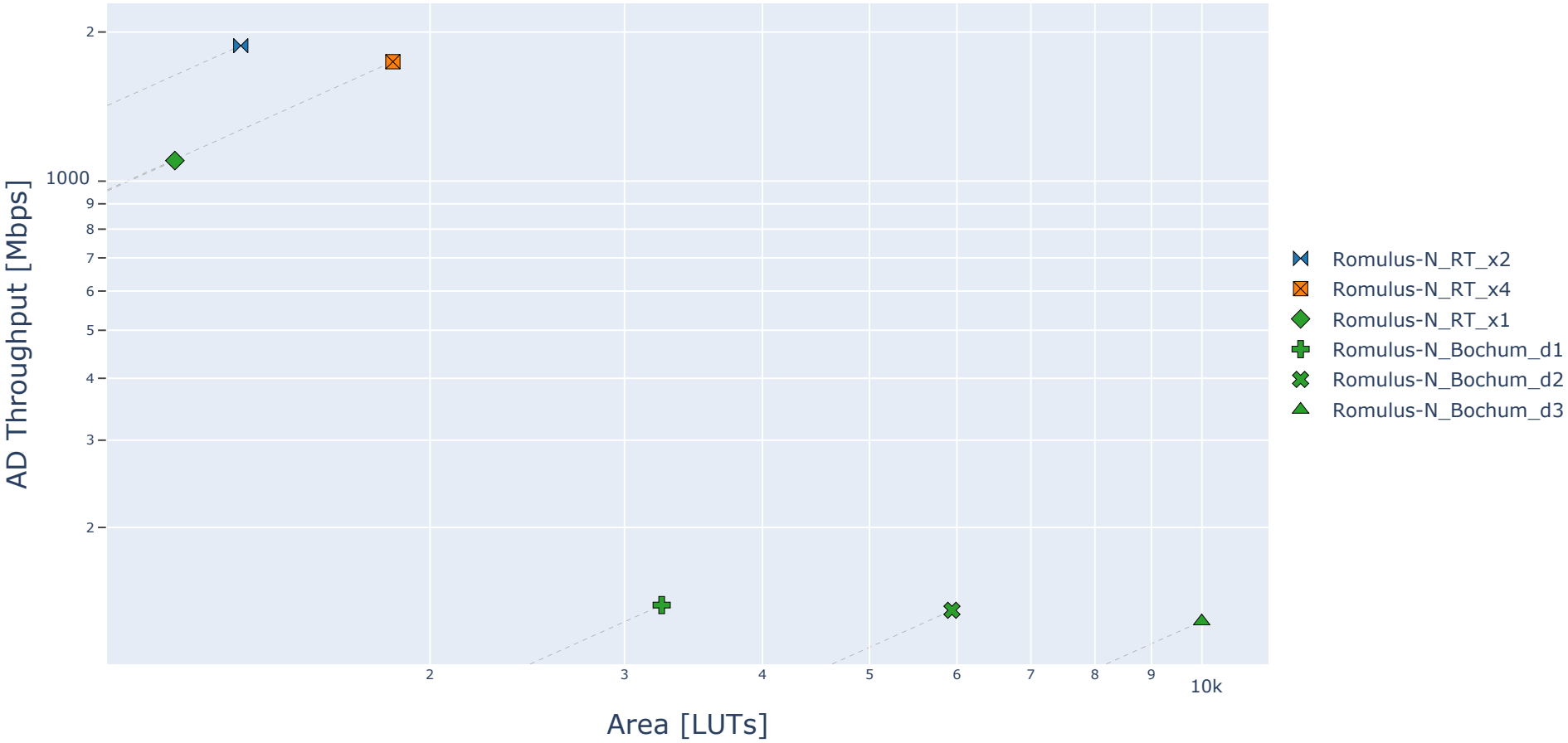
# PHOTON-Beetle: Hashing Throughput vs. Area



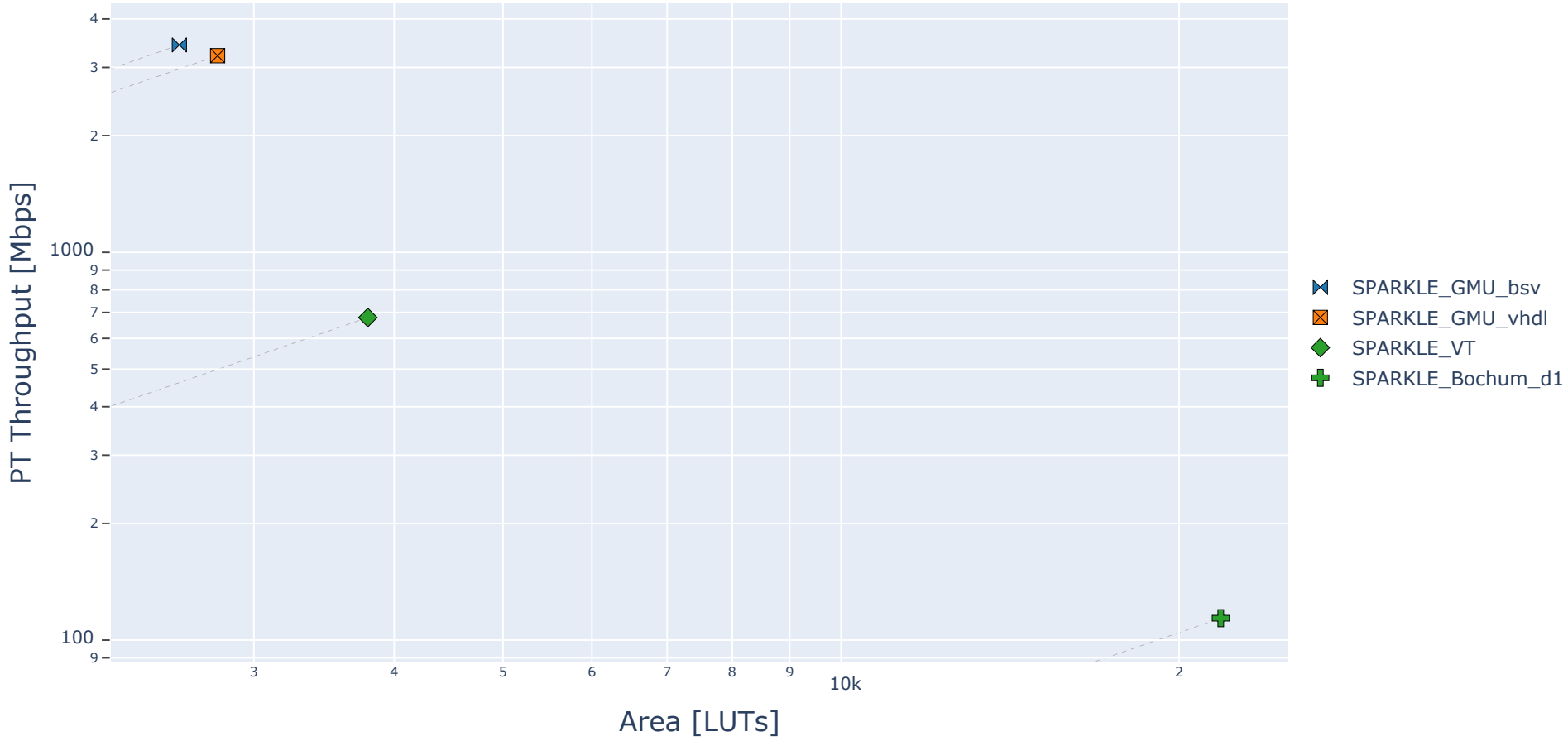
# Romulus: PT Throughput vs. Area



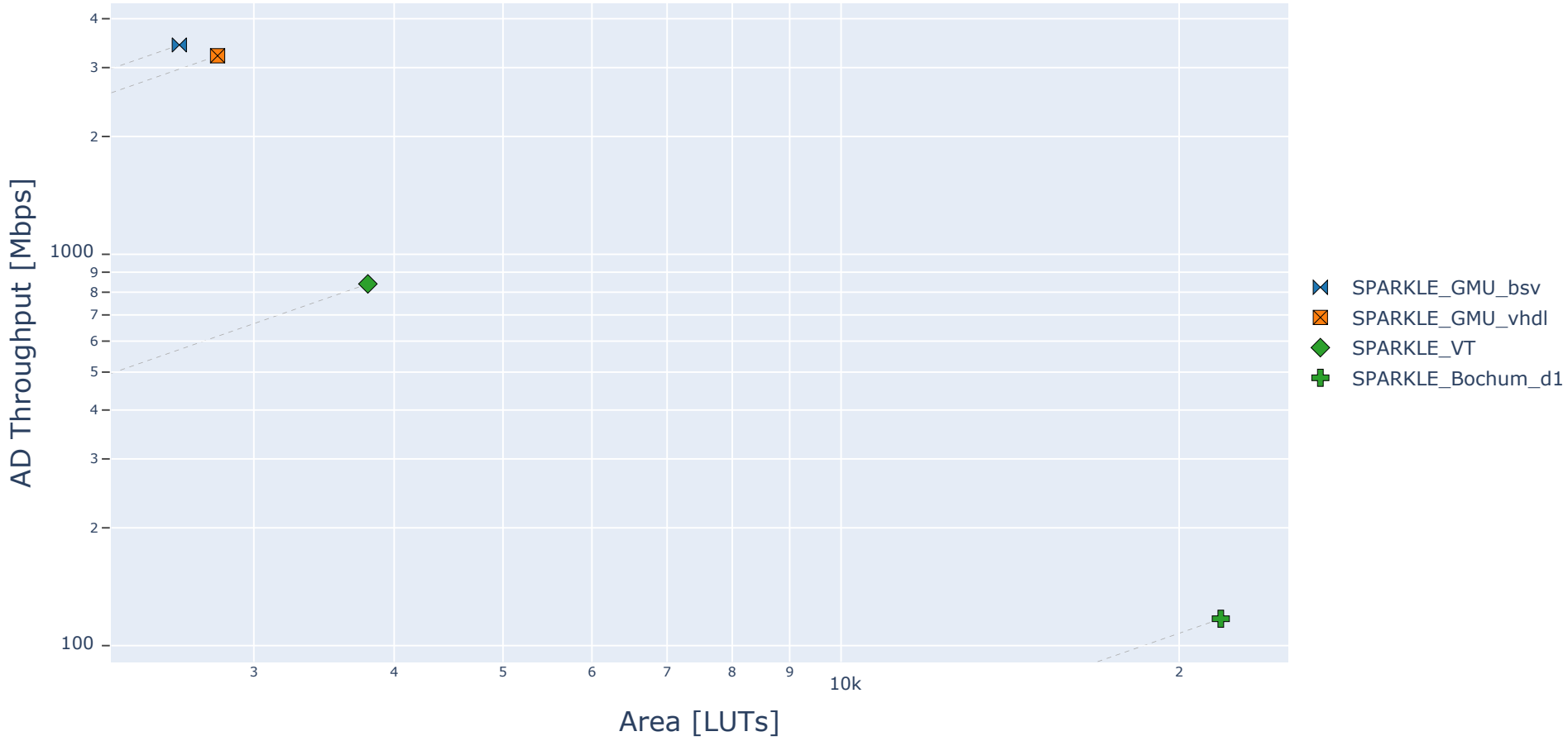
# Romulus: AD Throughput vs. Area



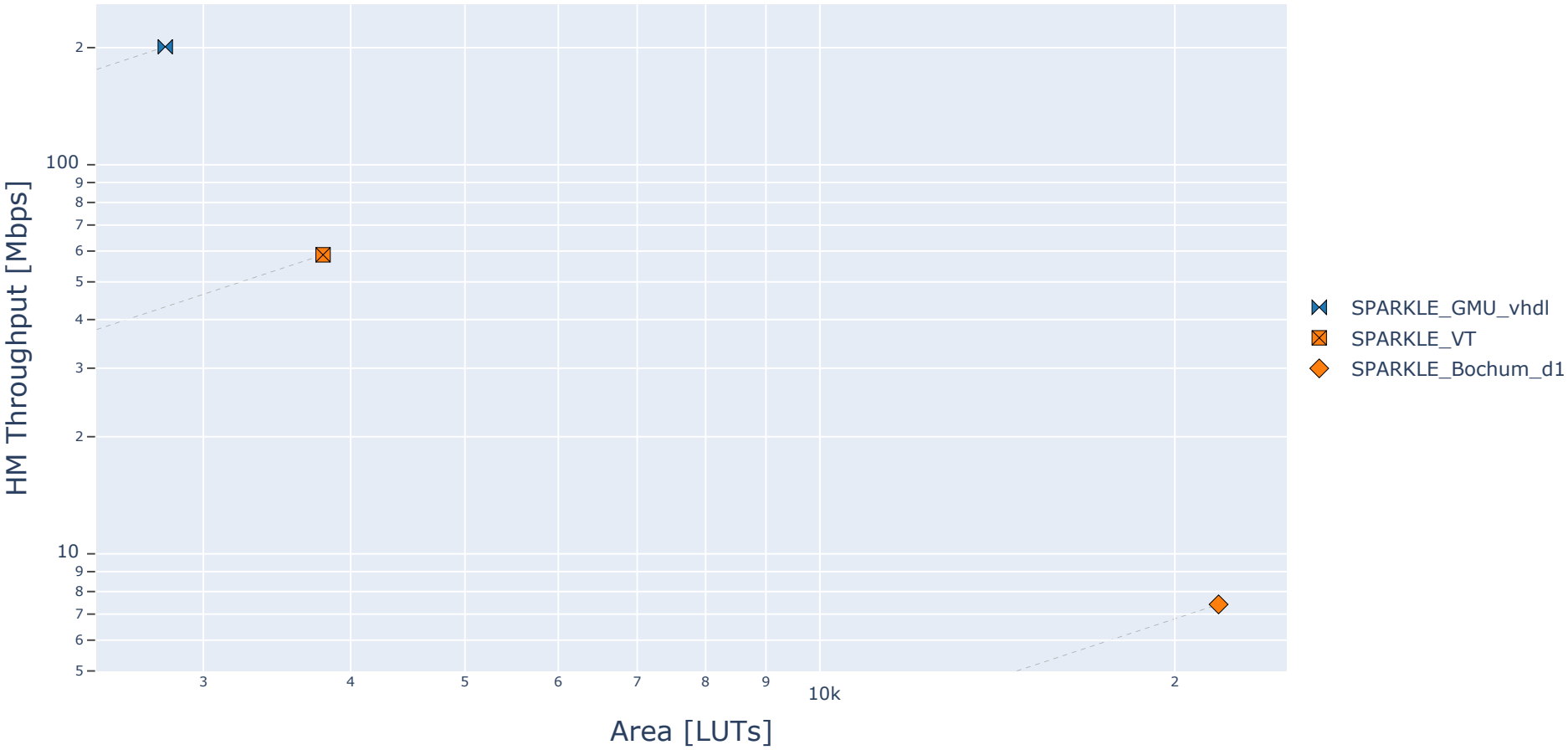
# SPARKLE: PT Throughput vs. Area



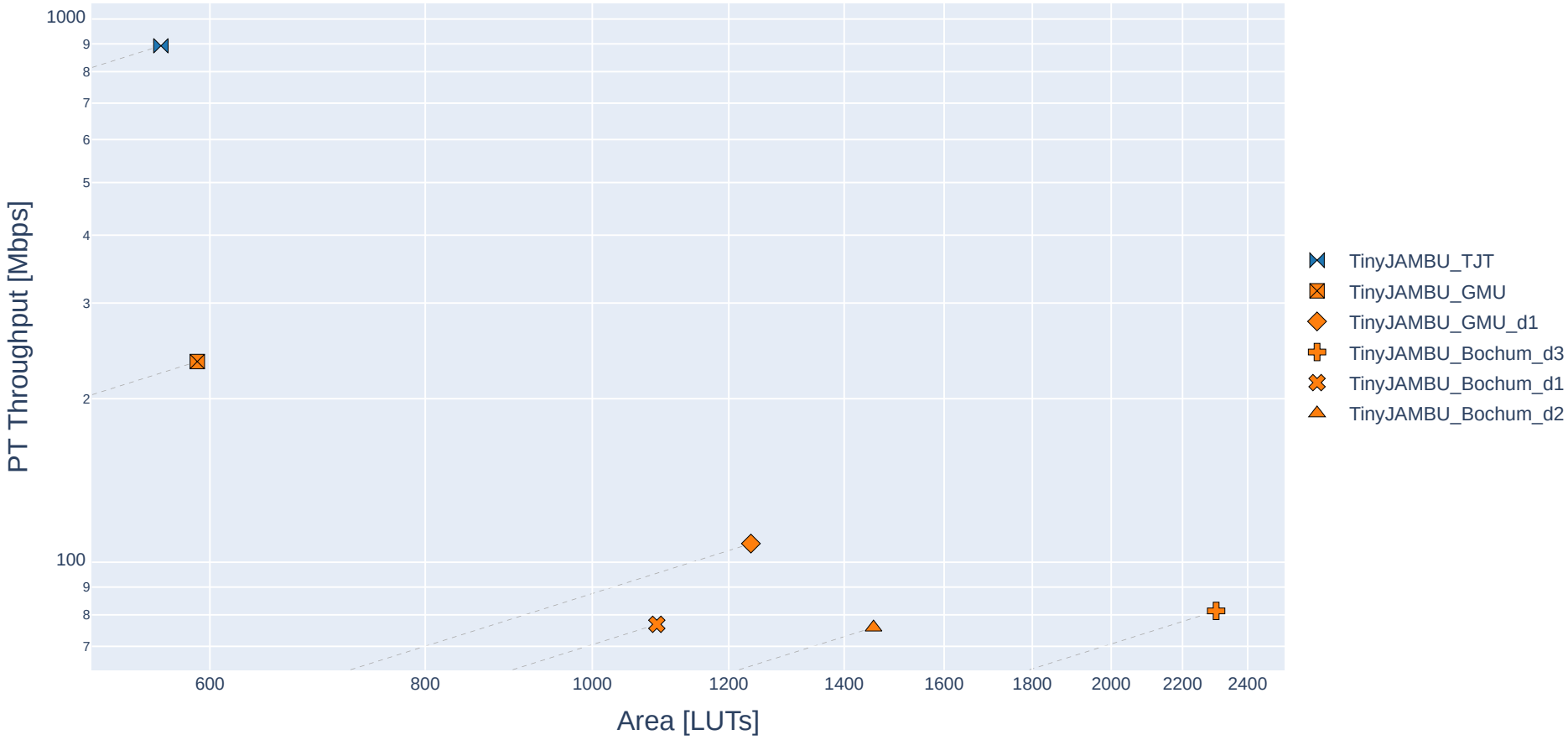
# SPARKLE: AD Throughput vs. Area



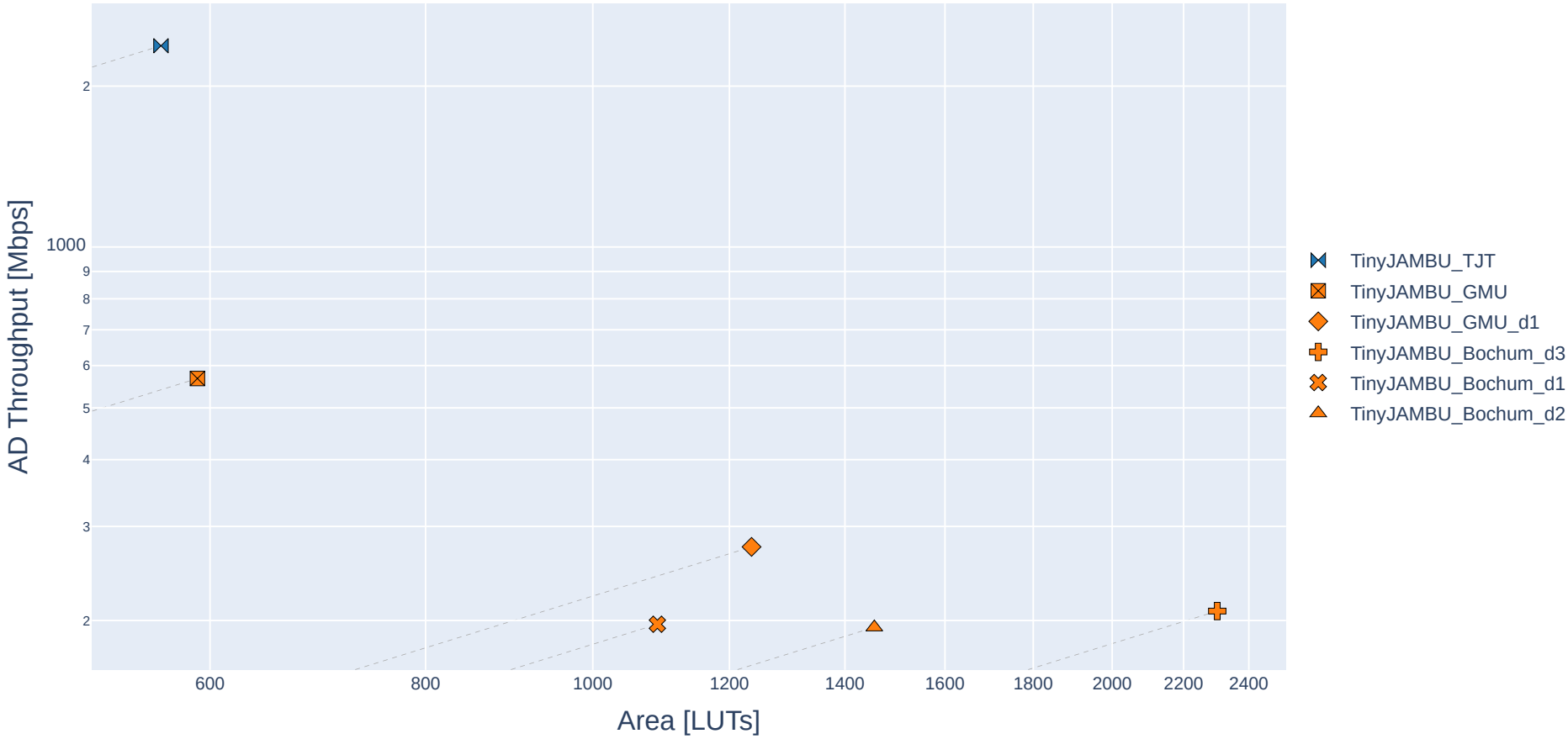
# SPARKLE: Hashing Throughput vs. Area



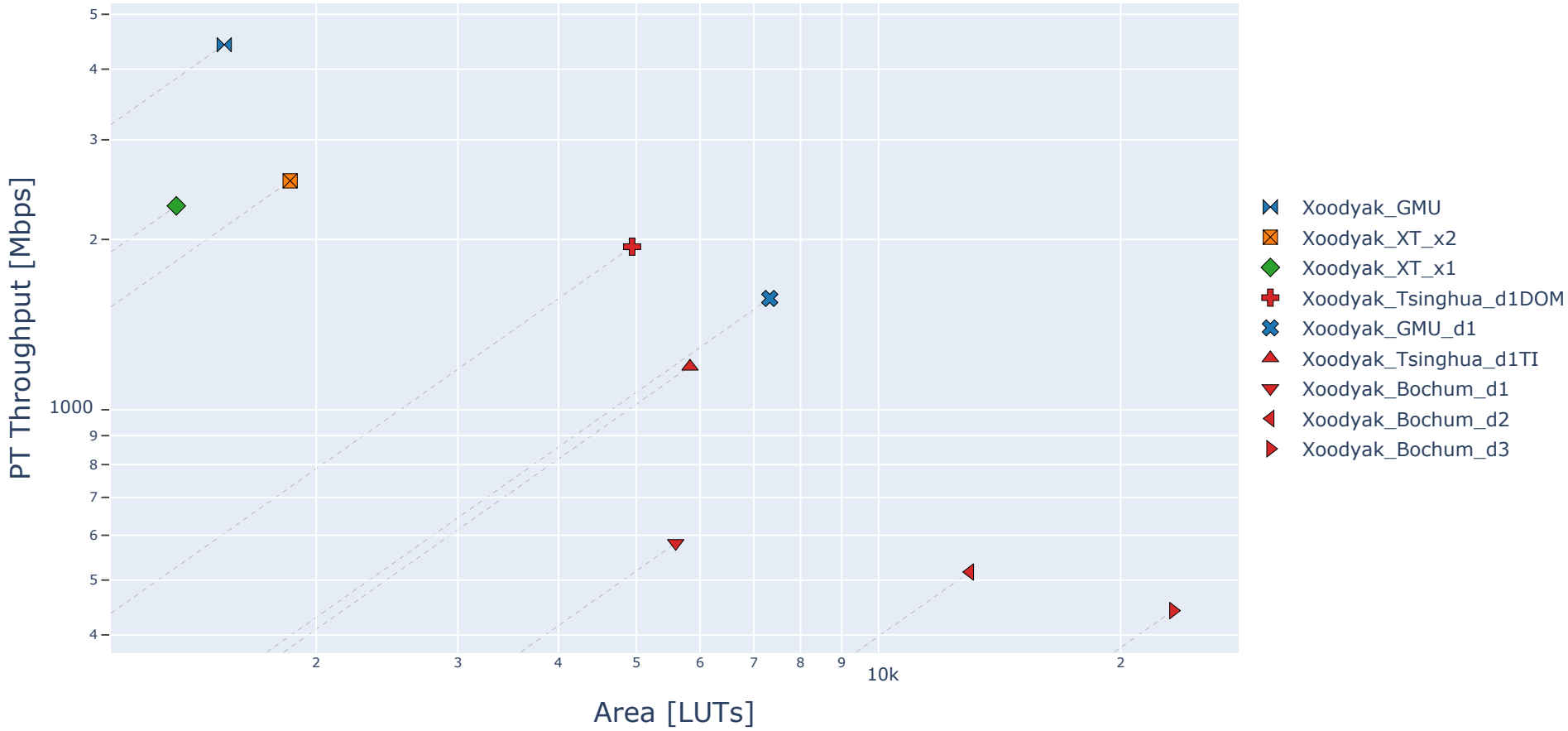
# TinyJAMBU: PT Throughput vs. Area



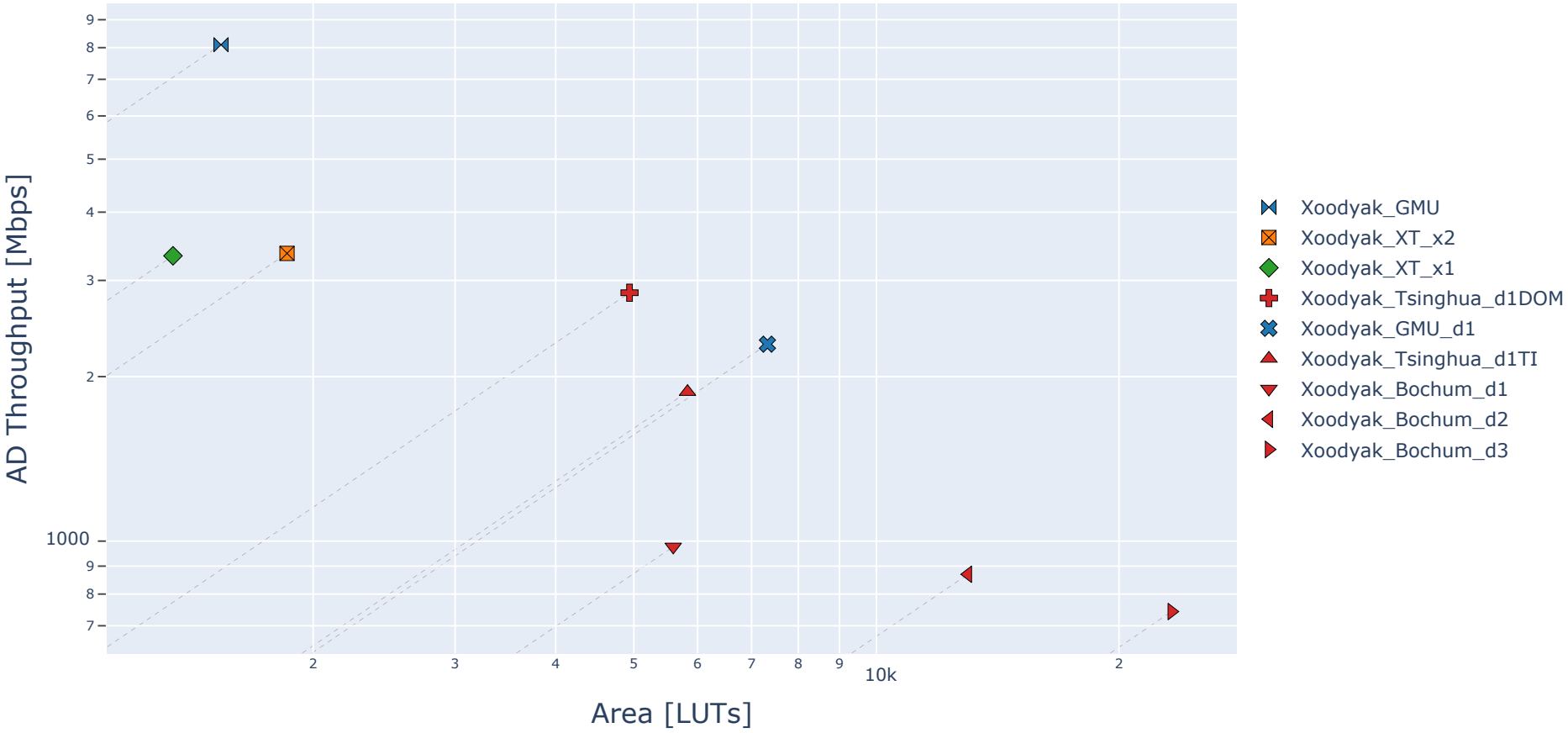
# TinyJAMBU: AD Throughput vs. Area



# Xoodyak: PT Throughput vs. Area



# Xoodyak: AD Throughput vs. Area



# Xoodyak: Hash Throughput vs. Area