FEASIBILITY AND PERFORMANCE OF QUANTUM RESISTANT CRYPTOGRAPHY IN EMBEDDED DEVICES

Brian Hession

Quantum Computing

- Developing at an increasing rate
- Import to consider the security implications
- Theorized to break RSA, ECDSA, Diffie-Helman, etc. by 2025
  - *Most secure web and network protocols currently rely on these algorithms*

Post-Quantum Initiative

- NIST has released an initiative to evaluate new quantum-resistant public key cryptographic standards
  - *Round 1 submissions ended November 30, 2017*
  - *Round 2 to be published January 2019*
- Most algorithms target only x86 and require system calls provided by operating systems
  - *IoT device popularity is rising at an alarming rate (estimate 75.44 billion devices by 2025)*
  - *Embedded systems will be left be behind*

Security Levels

Level	Security Description
I	At least as hard to break as AES-128 using exhaustive key search
II	At least as hard to break as SHA-256 using collision search
III	At least as hard to break as AES-192 using exhaustive key search
IV	At least as hard to break as SHA-384 using collision search
V	At least as hard to break as AES-256 using exhaustive key search

Embedded Device Struggles
EK-tm4c123gxl

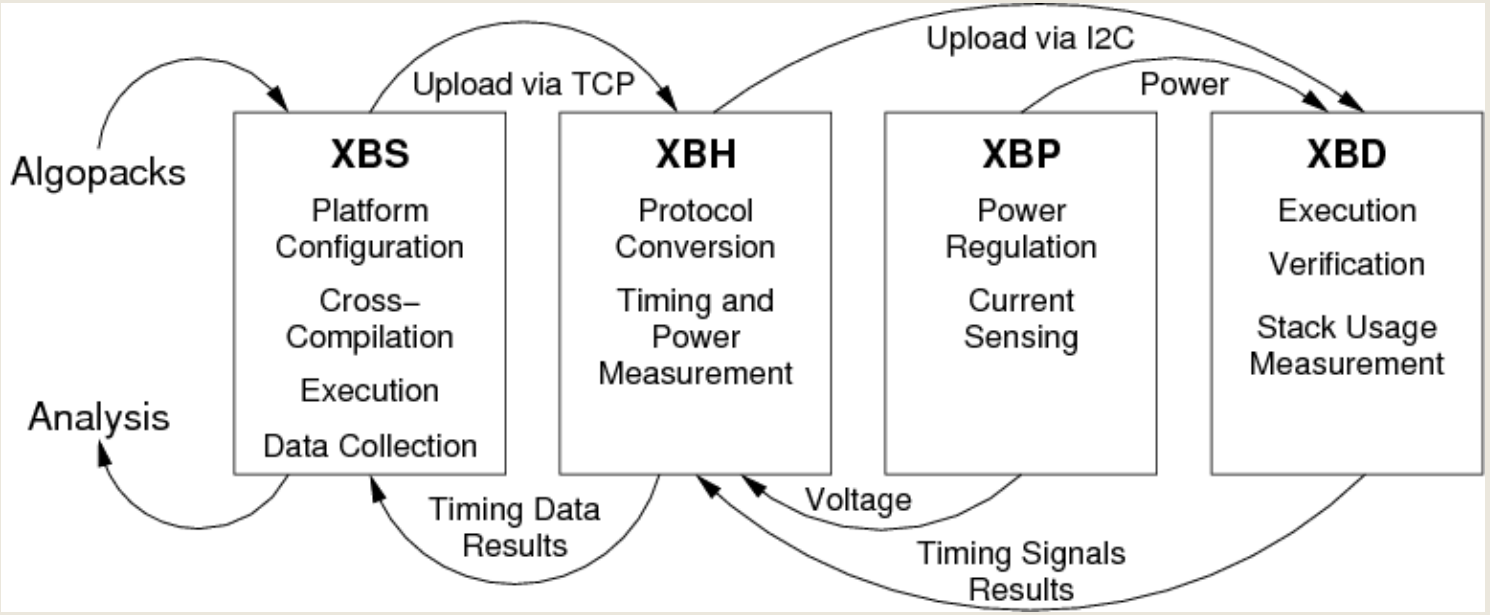
Manufacturer	TI
CPU	ARM Cortex M4F
ISA	ARMv7E-M
Bus	32-bit
Frequency	80 MHz
ROM	1024 kB
RAM	256 kB

- Strict constraints
  - *ROM*
  - *RAM*
  - *Power consumption*
- Tool to measure feasibility and performance of quantum-resistant cryptography in embedded environments?

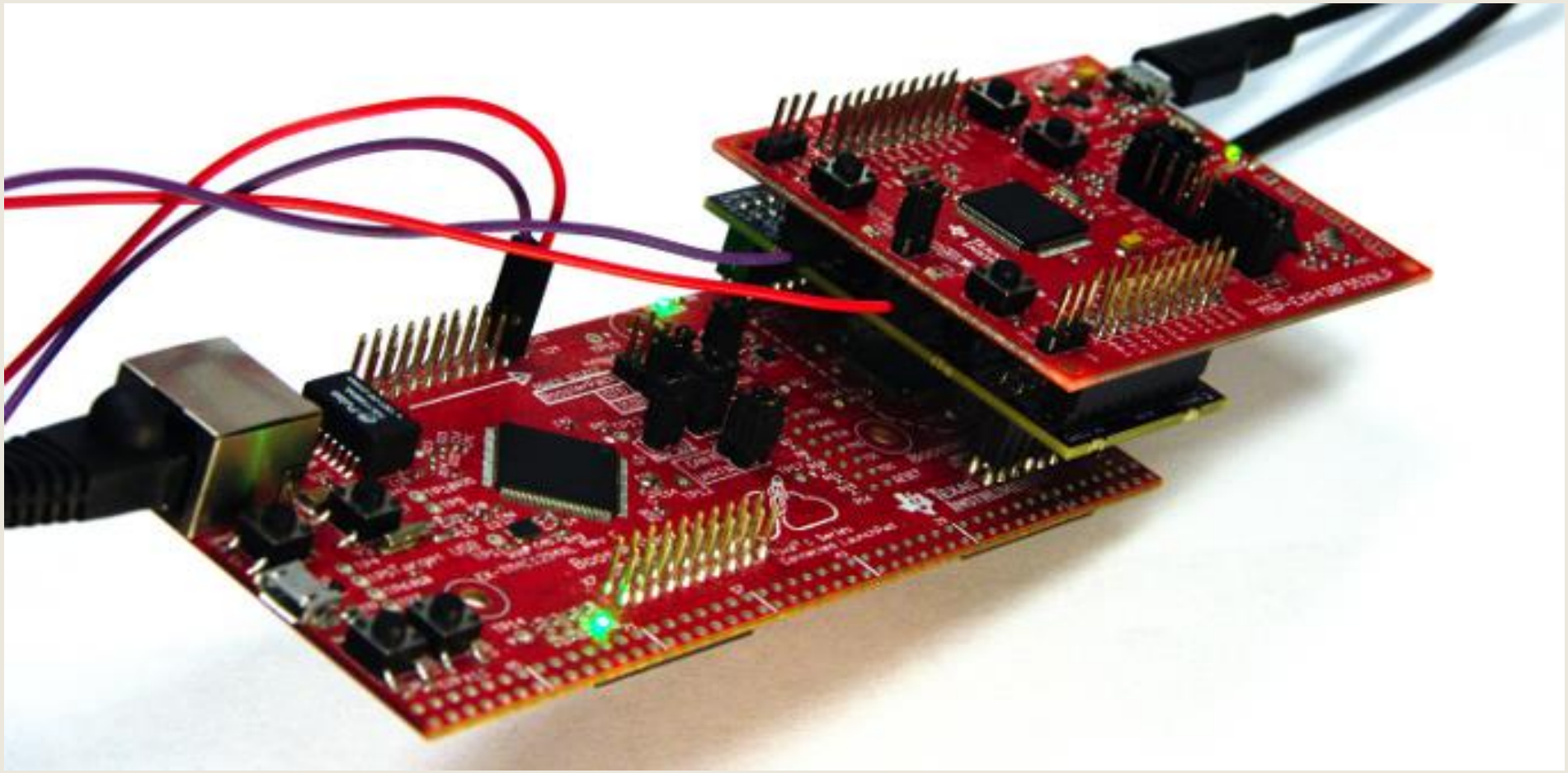
eXtended eXternal Benchmarking eXtension (XXBX)

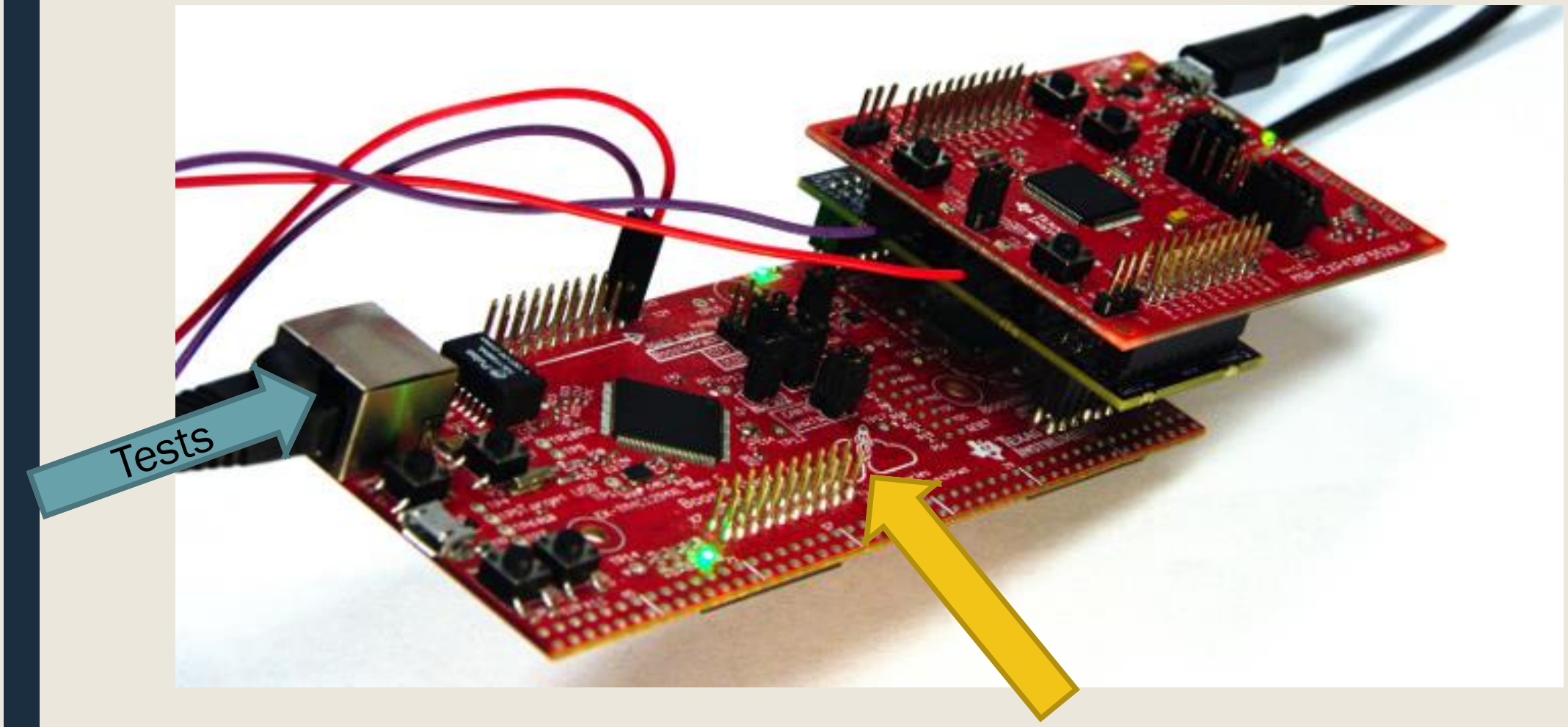
- Originally developed by John Pham and Dr. Kaps of GMU (CERG)
- Tool used for benchmarking the performance, memory usage, and power/energy consumption of cryptographic software on microcontrollers
- XXBX is an extension of XBX which originally benchmarked many different hashing functions on several different microcontrollers
  - *Developed by Christian Wenzel-Benner and Jens Gräf*
  - *XBX is an extension of SUPERCOP developed at VAMPIRE Labs*

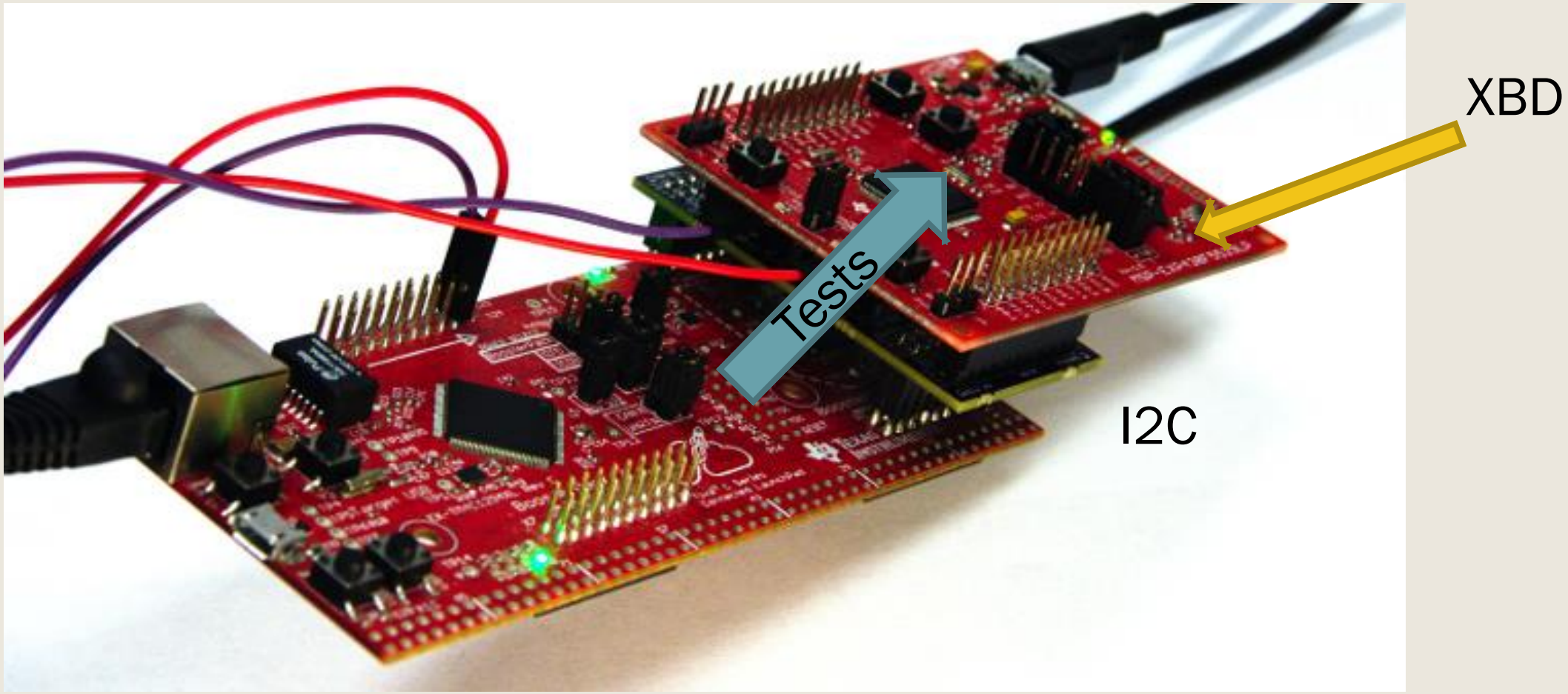
XXBX Flow

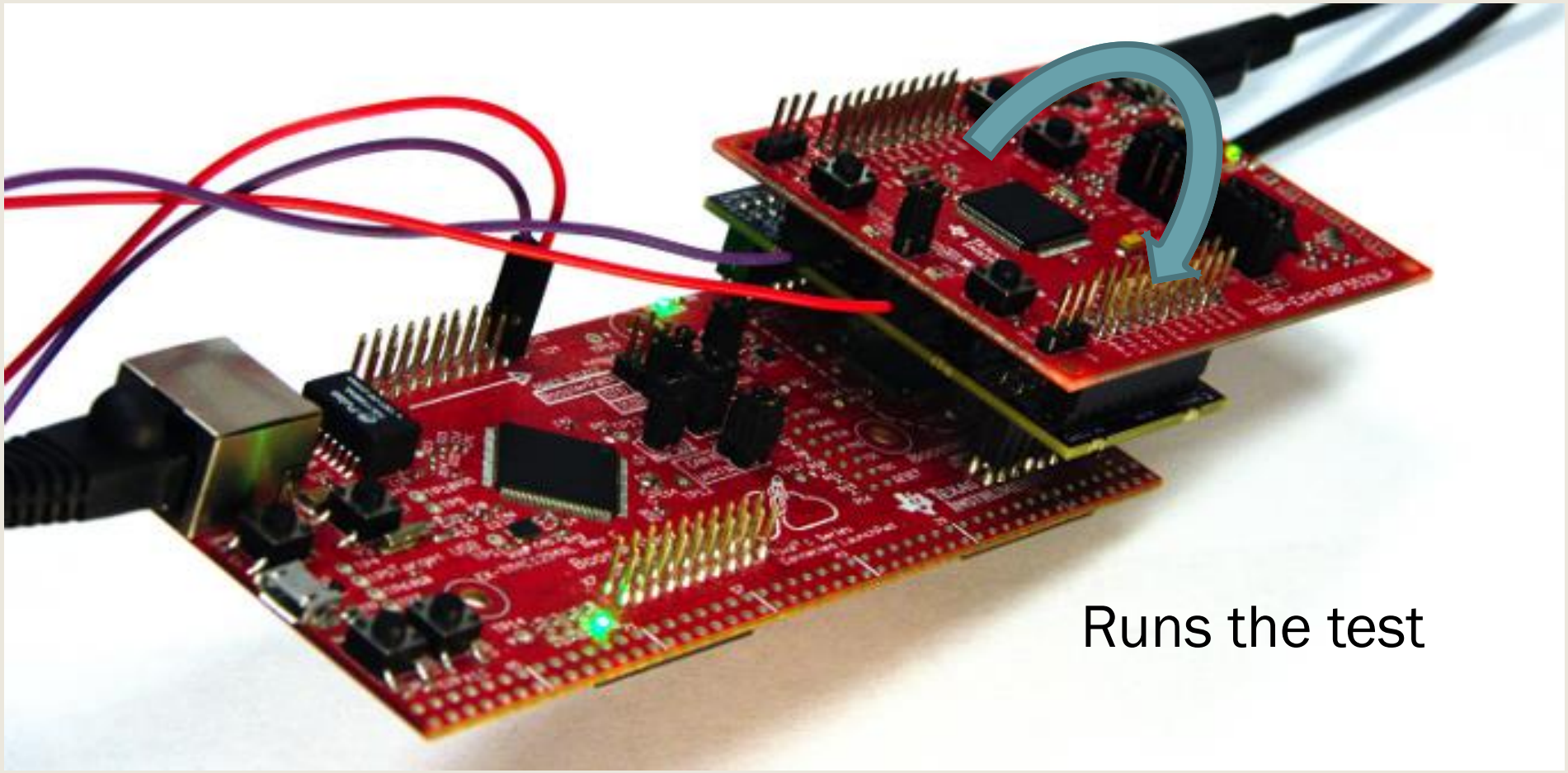


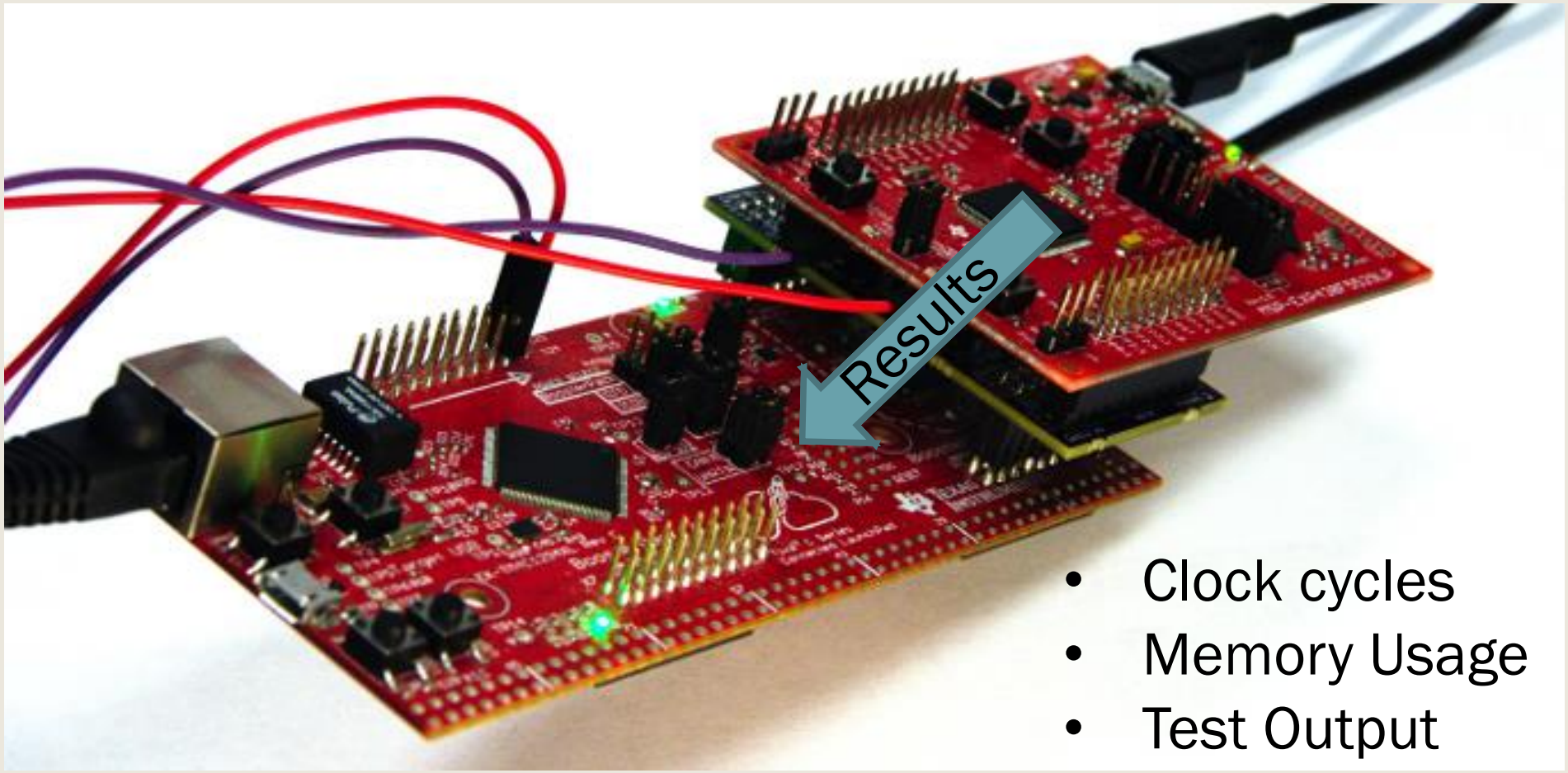
- XBS (eXternal Benchmarking Software) – Python scripts to interact with XXBX
- XBH (eXternal Benchmarking Harness) – Acts as the control center and interface
- XBP (eXternal Benchmarking Power) – Regulates power and current
- XBD (eXternal Benchmarking Device) – The device the tests will run on

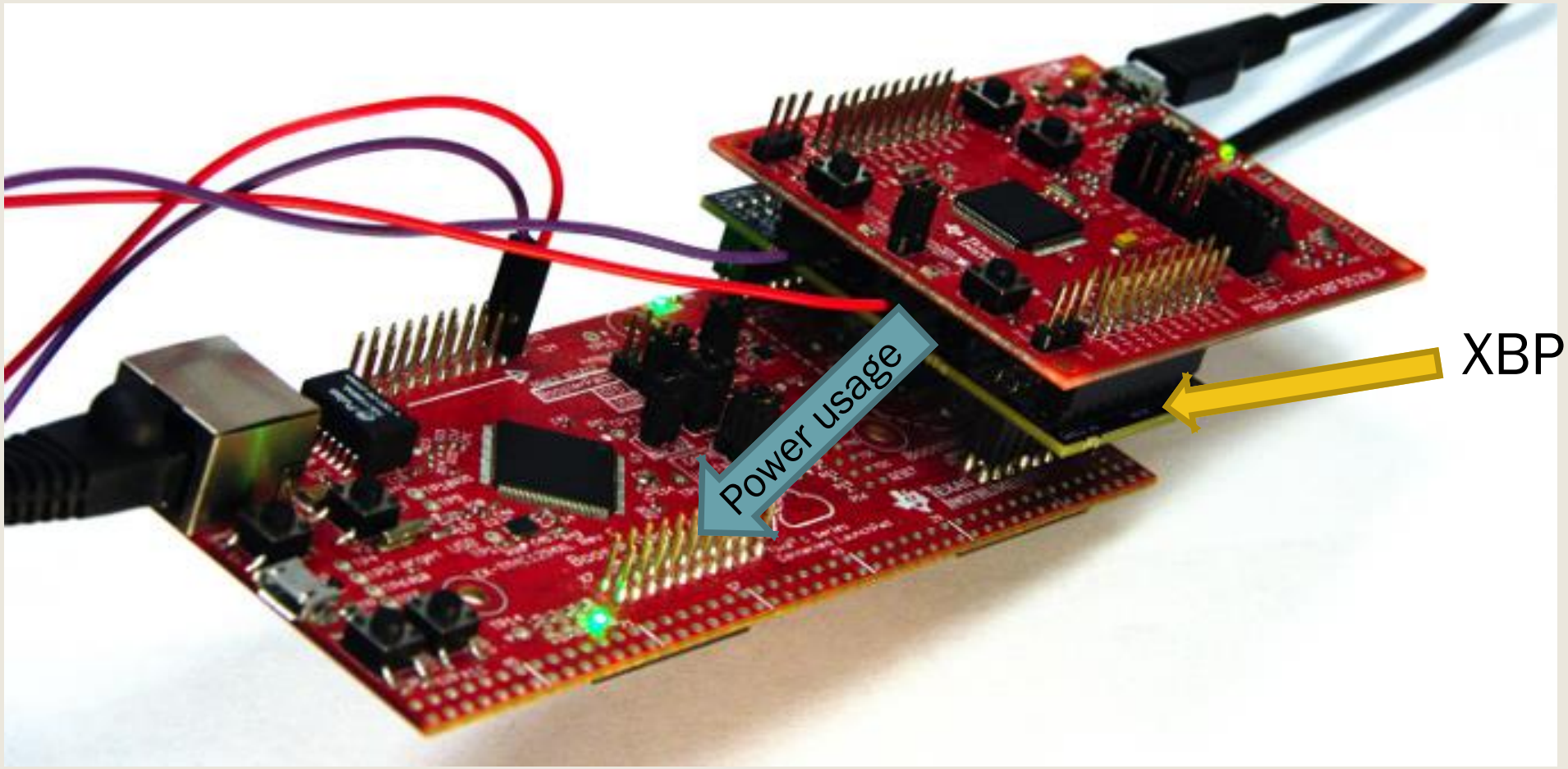


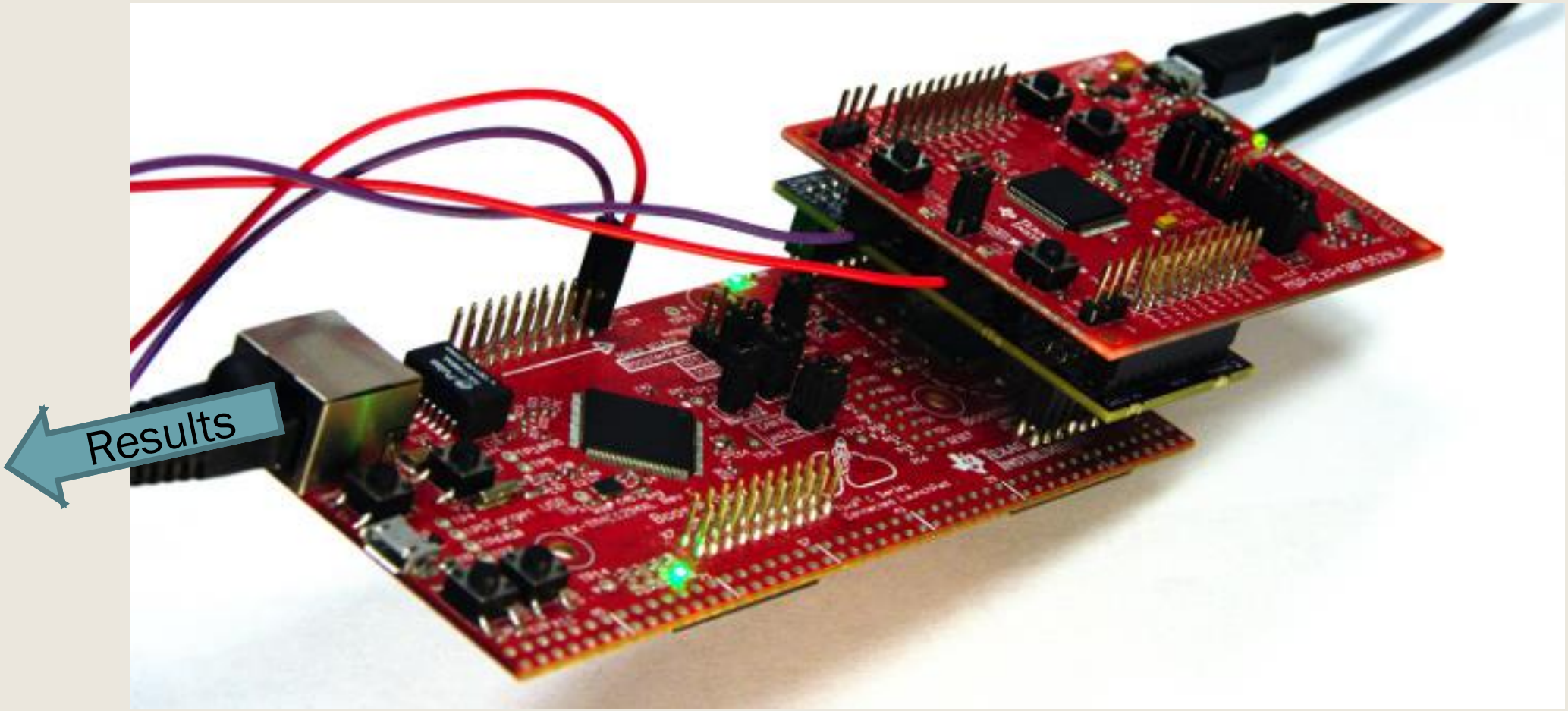

Tests
XBH


XBD
Tests
I2C


Runs the test


Results
• Clock cycles
• Memory Usage
• Test Output


Power usage
XBP


Results

XXBX Extended Functionality

- XXBX original functionality benchmarked hashing and authenticated ciphers
- Extended to support signature schemes as well as key encapsulation methods
  - *Key encapsulation is vital for communicating securely over the Internet*

XXBX Constraints

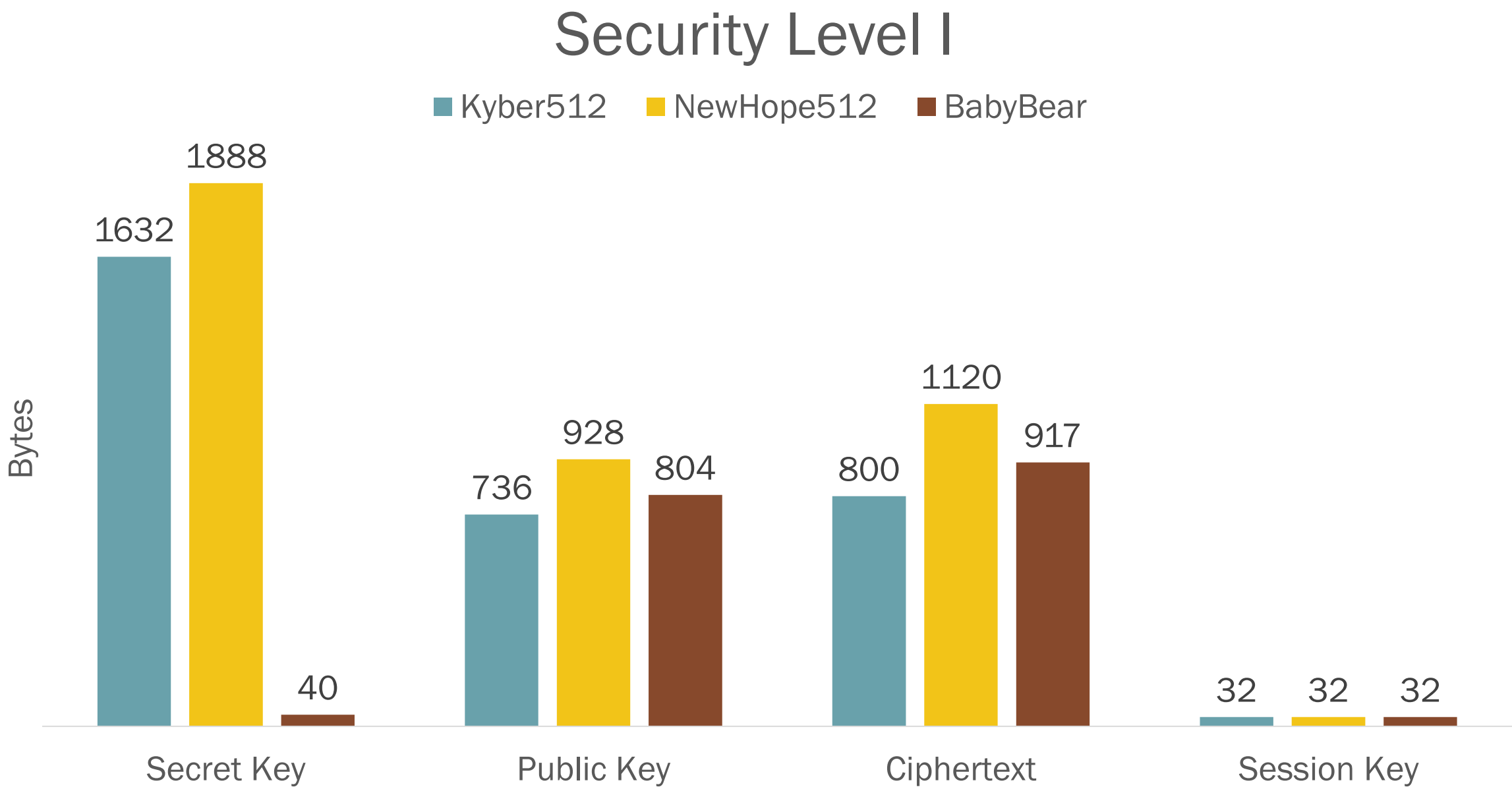
- XBH seems to only be able to handle receiving around 3000 bytes
  - *Communication halts and XBH crashed consistently*
  - *Implemented algorithms could not return any data greater than 3000 bytes*
- Because of time constraints, this issue is still left open

Signature Operations

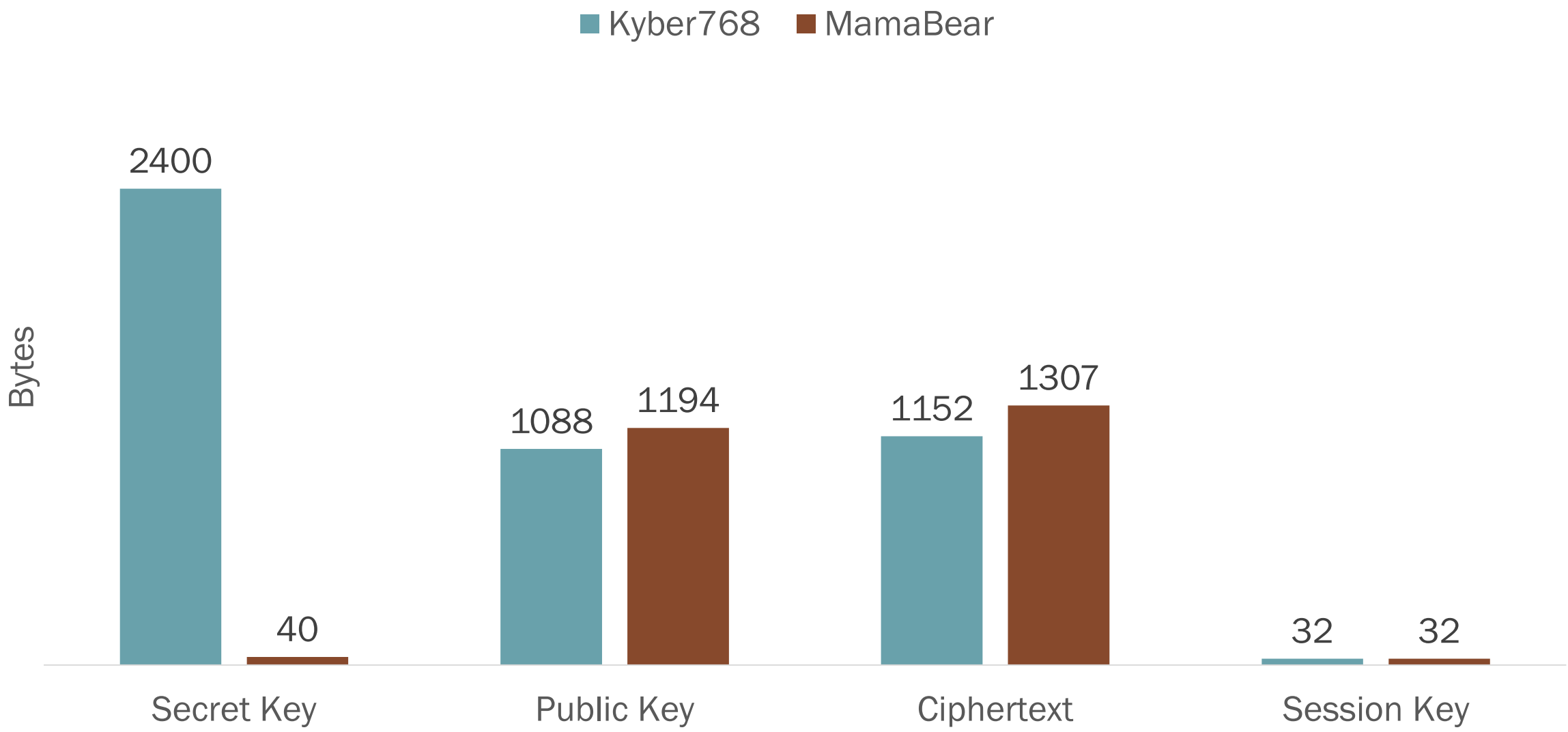
- Many issues were encountered implementing signing capabilities
  - *Harsh size constraints on message size*
    - Many signature sizes were enormous (10+ kB regardless of message length)
    - Limited feasibility of testing signatures in the XXBX environment
  - *Message length is not constant*
    - Many algorithms require dynamic allocation
- As a result, analysis of signature operations were dropped. However, the functionality still remains

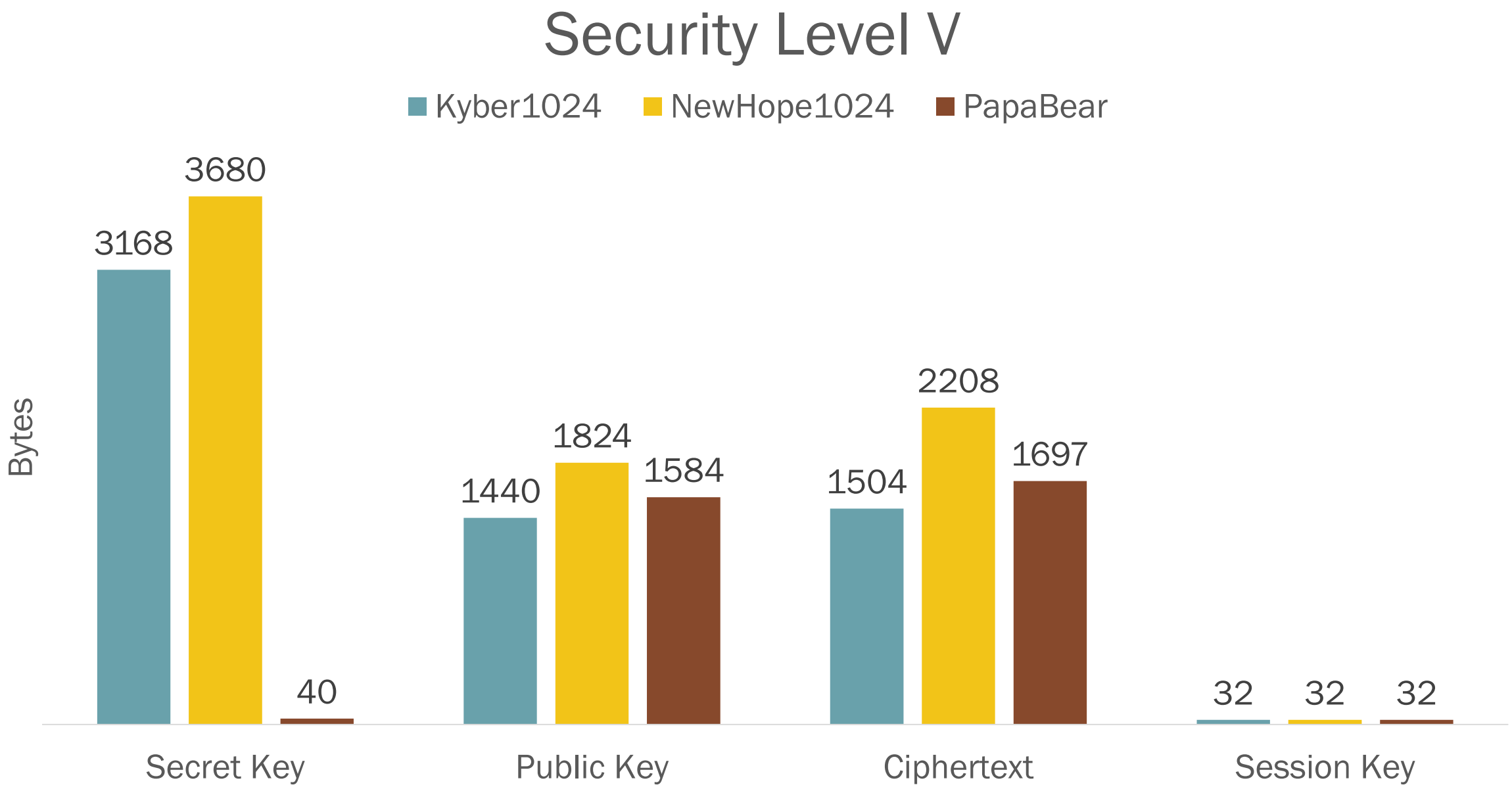
Post-Quantum Implementation (KEM)

- Originally started with the liboqs library.
  - *Developed by Open Quantum Safe at the University of Waterloo*
- Added more beyond the library


Security Level I
Kyber512
NewHope512
BabyBear
Bytes
1632
1888
40
736
928
804
800
1120
917
32
32
32
Secret Key
Public Key
Ciphertext
Session Key

Security Level III Algorithms




Security Level V
Kyber1024
NewHope1024
PapaBear
Bytes
3168
3680
40
1440
1824
1584
1504
2208
1697
32
32
32
Secret Key
Public Key
Ciphertext
Session Key

Other KEM Candidates

Not Embedded-Compatible

- Saber
- Kindi
- LAC
- Lake
- LedaKEM
- Locker
- SIKE

Too large for XXBX

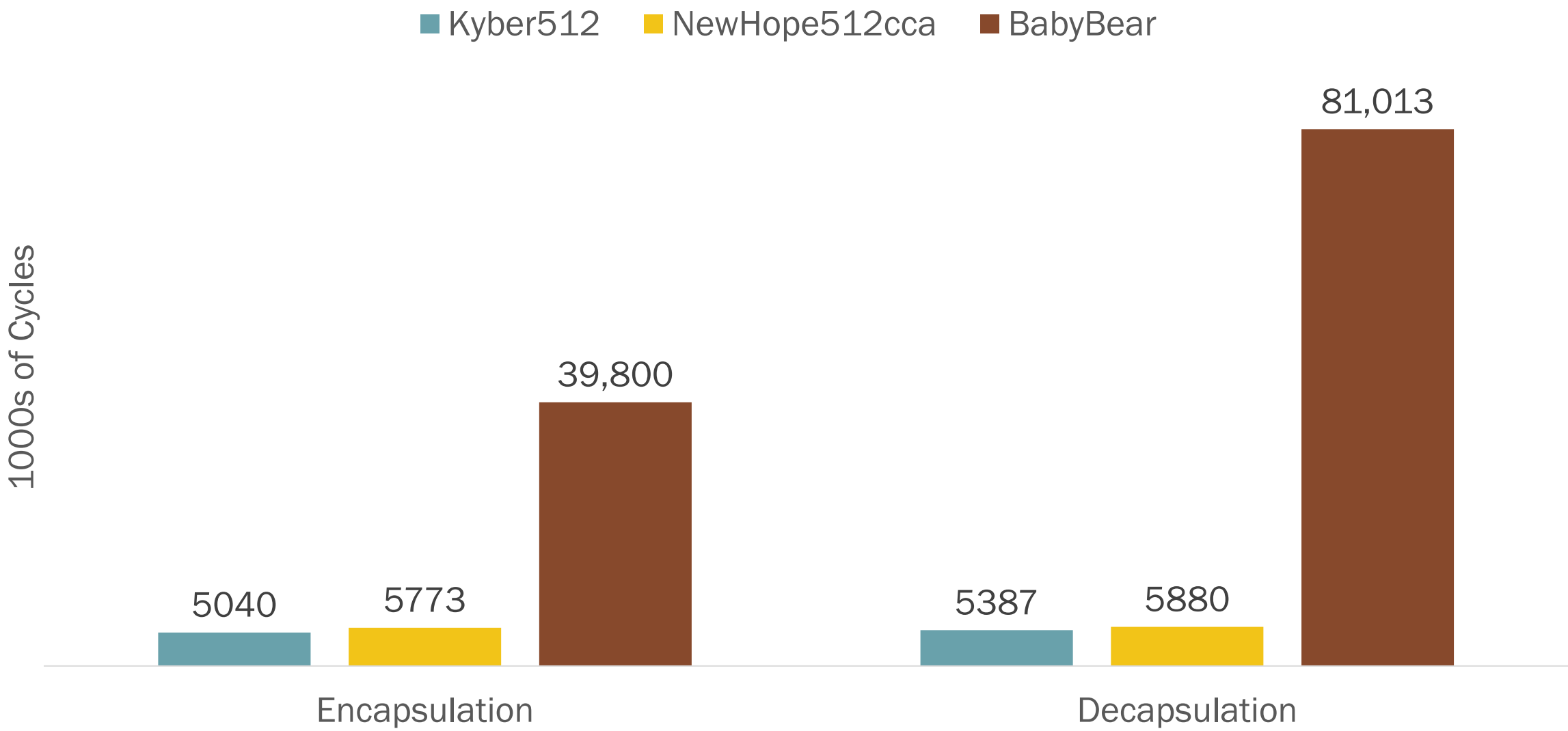
- BIG QUAKE (25+ kB public key)
- DAGS (2+ MB secret key)
- FrodoKEM (9+ kB ciphertext)
- Lima (4+ kB ciphertext)
- Lotus (65+ kB public key)
- McEliece (1+ MB public key)
- Titanium (3+ kB ciphertext)

I2C Constraint Strikes Again

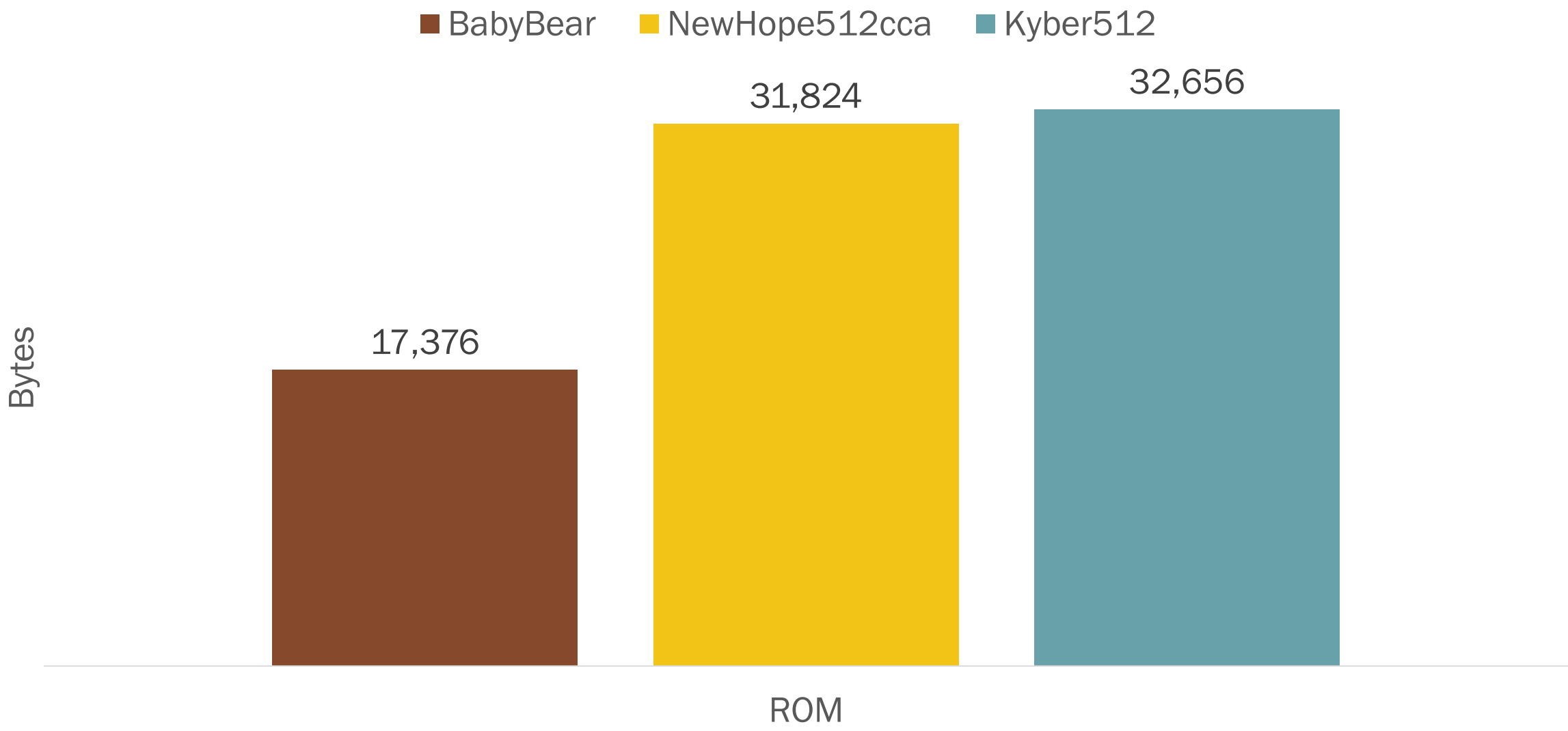
- Only Level I KEM algorithms could successfully send back the generated keypair
  - *Generated keypairs and uploaded them the test encapsulation and decapsulation*

Security Level I Results

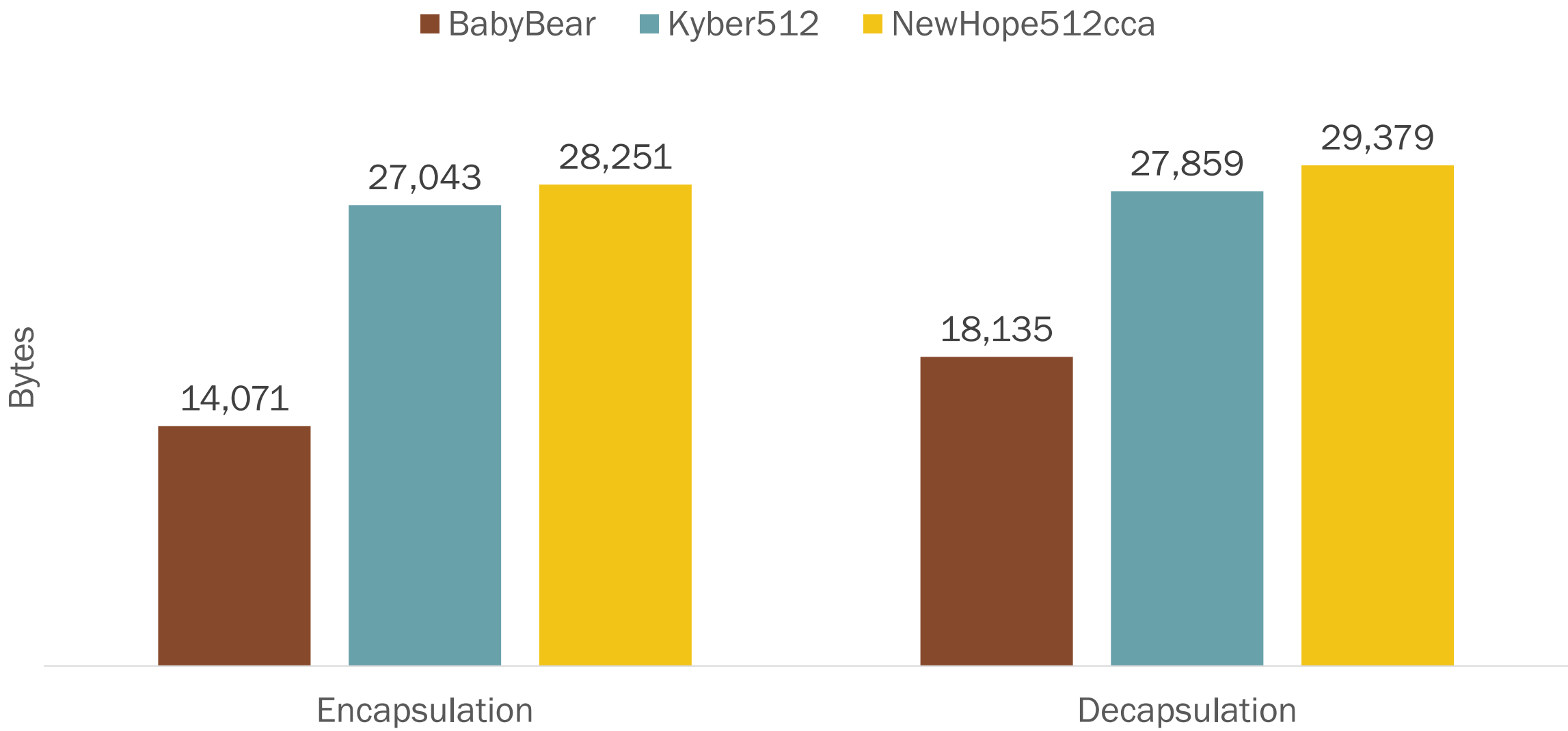
Security Level I – Speed



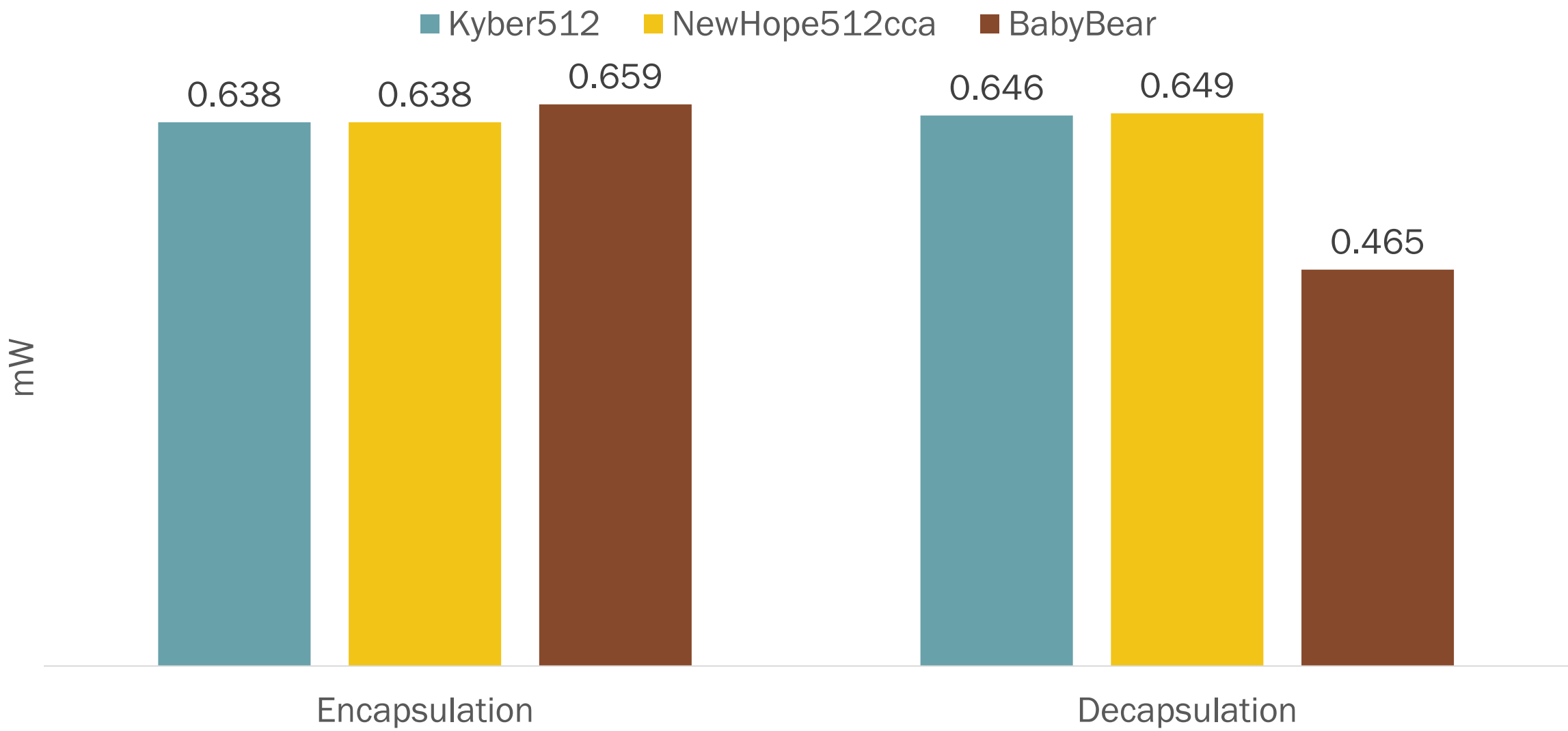
Security Level I – ROM Usage



Security Level I – RAM Usage



Security Level I – Max Power

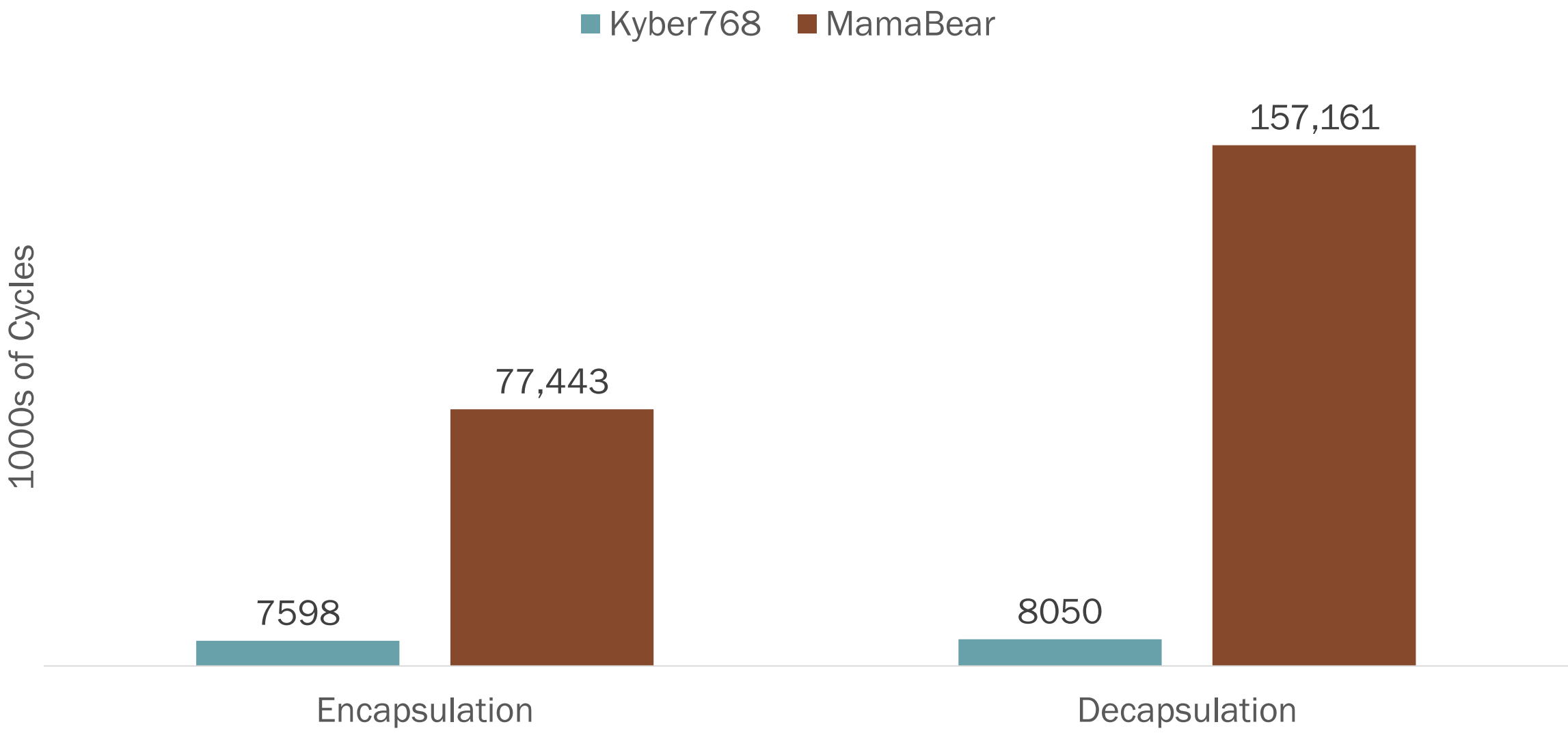


Security Level I Conclusions

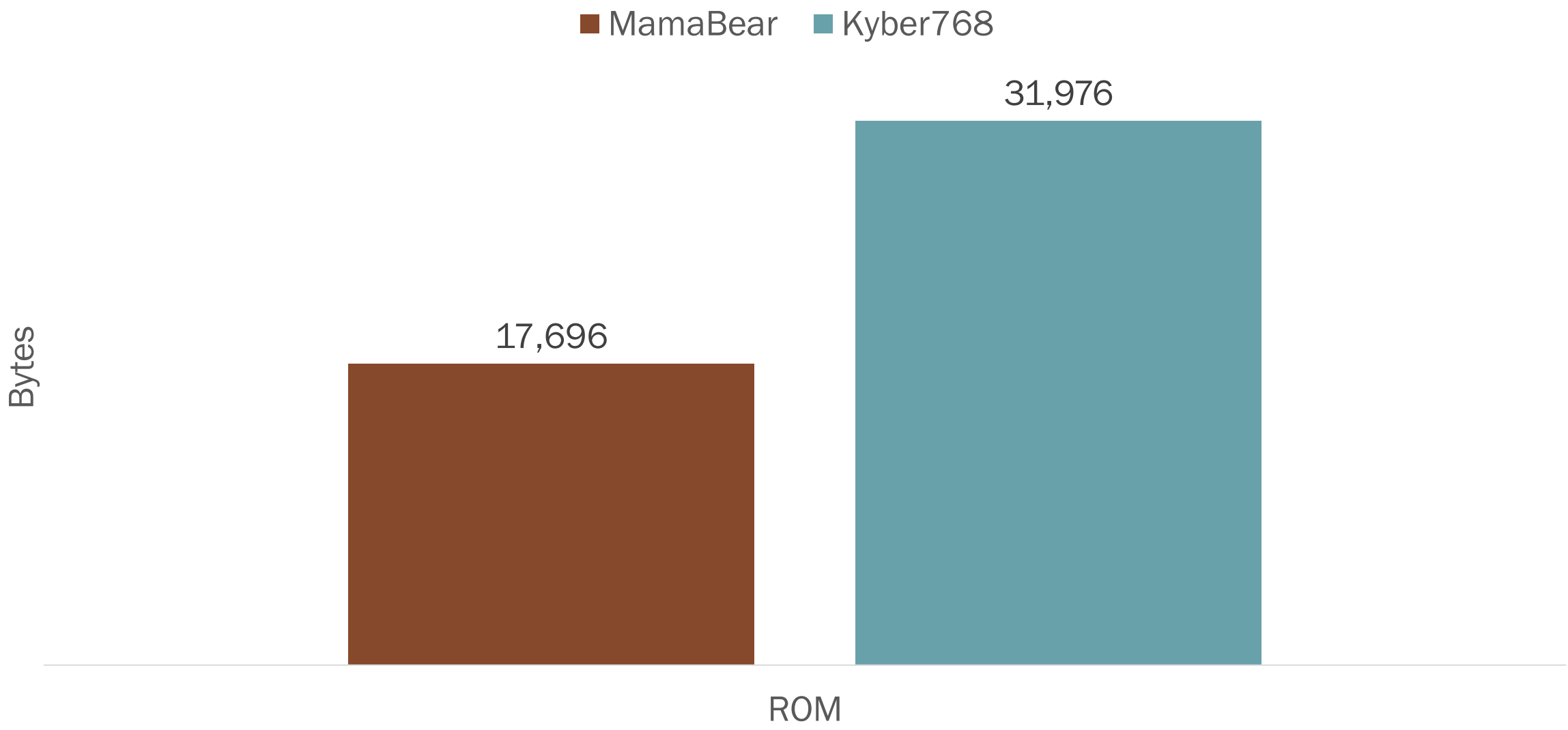
- If speed is the greatest concern, Kyber512 and NewHope512 are good candidates
- If memory usage or power consumption are the greatest concerns, BabyBear is the better choice

Security Level III Results

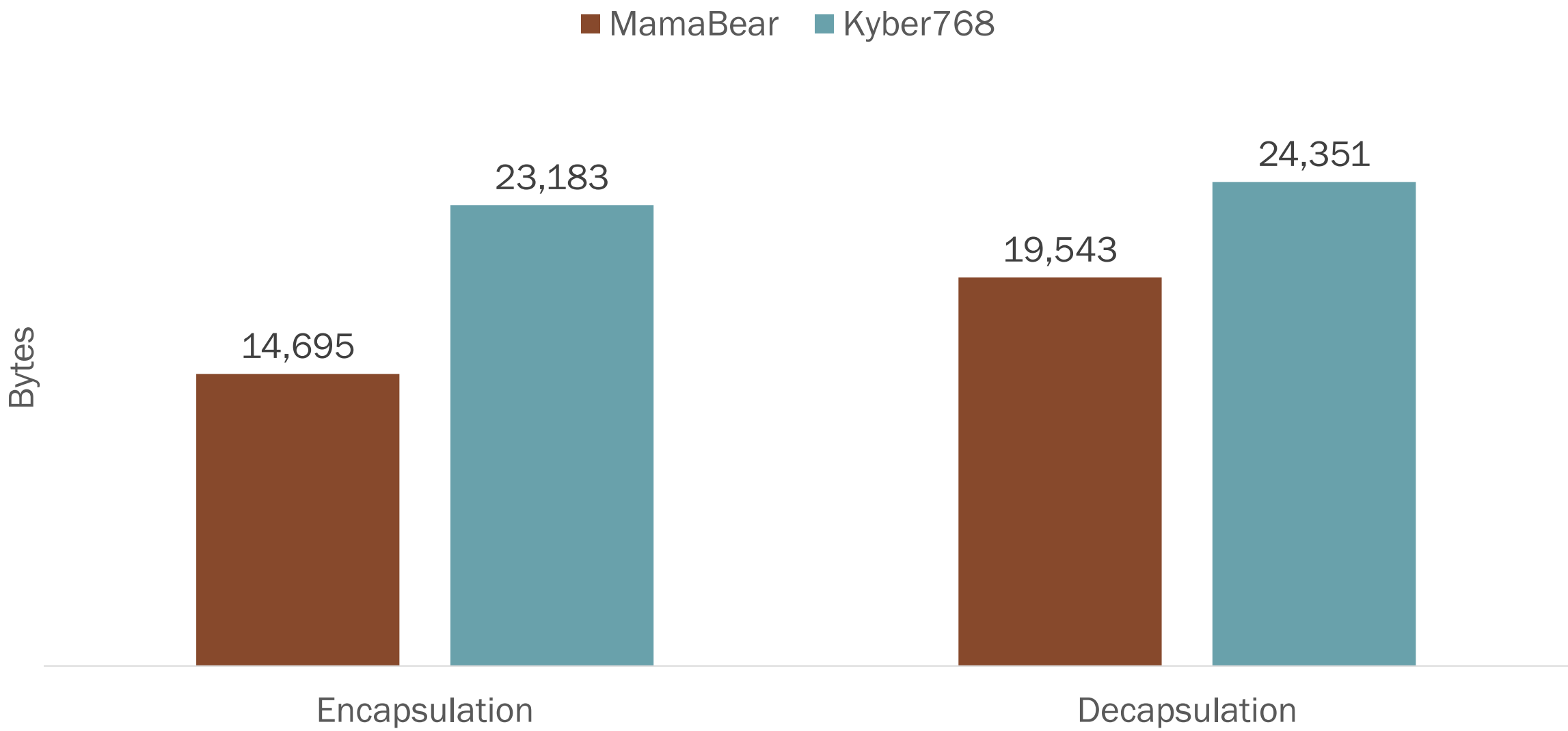
Security Level III – Speed



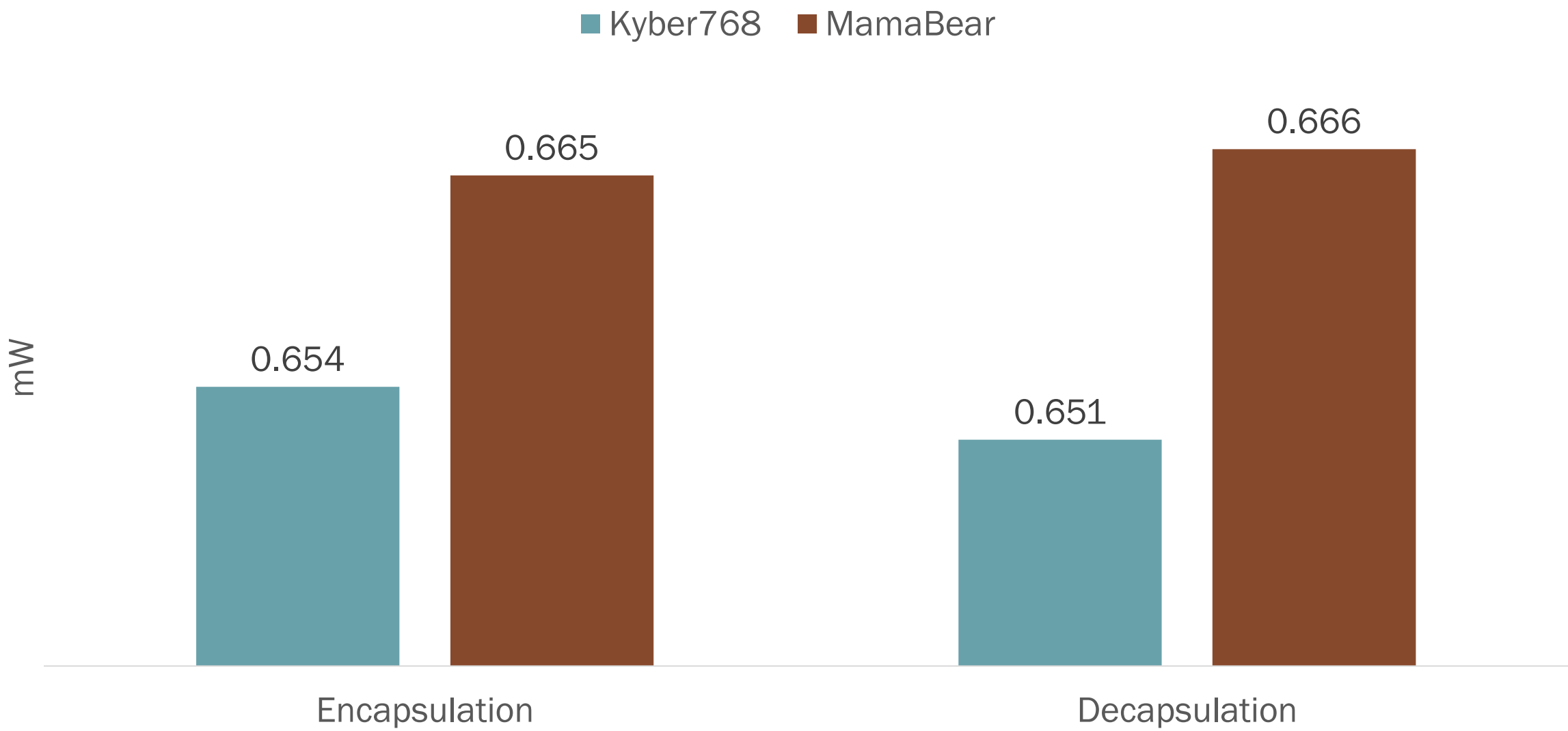
Security Level III – ROM Usage



Security Level III – RAM Usage



Security Level III – Max Power

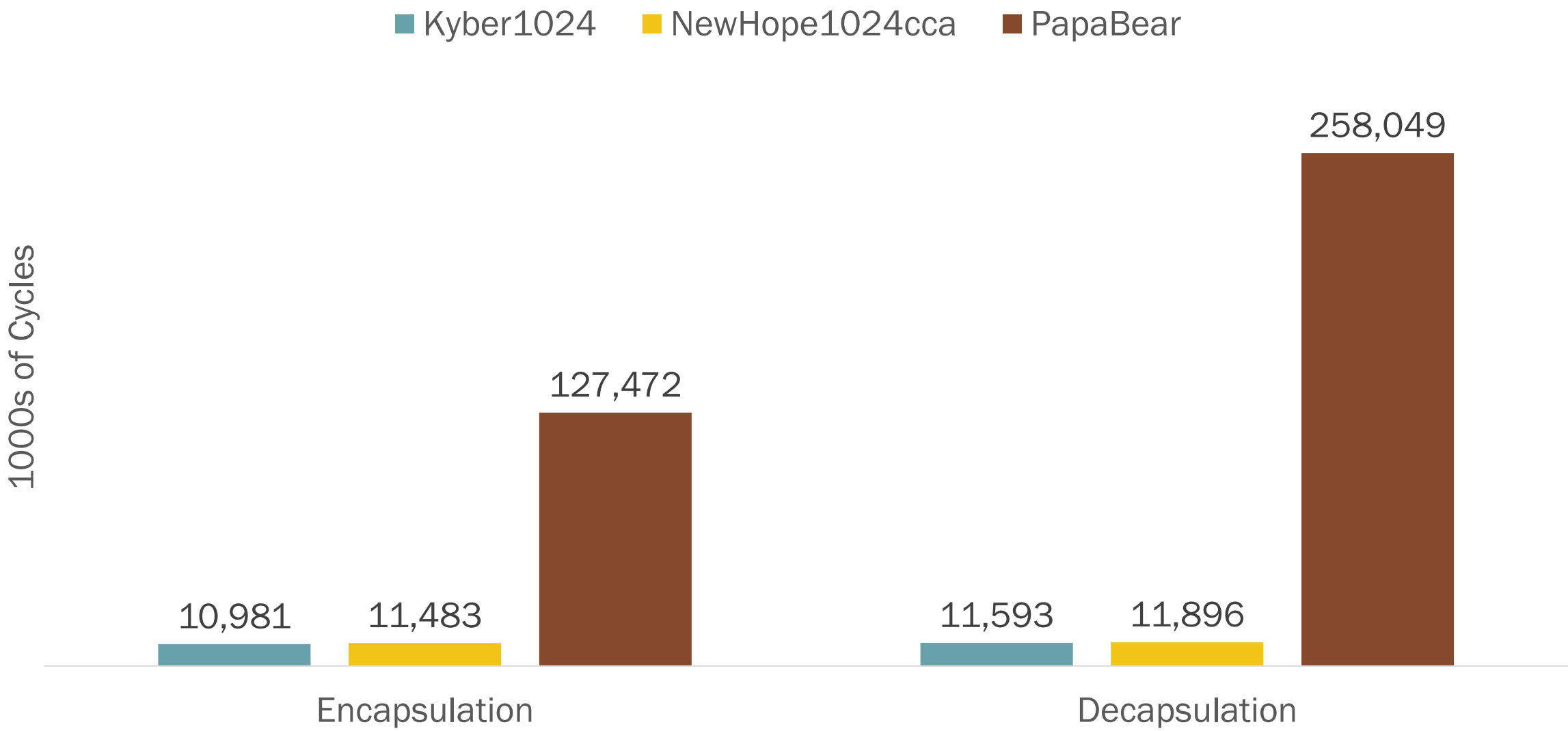


Security Level III Conclusions

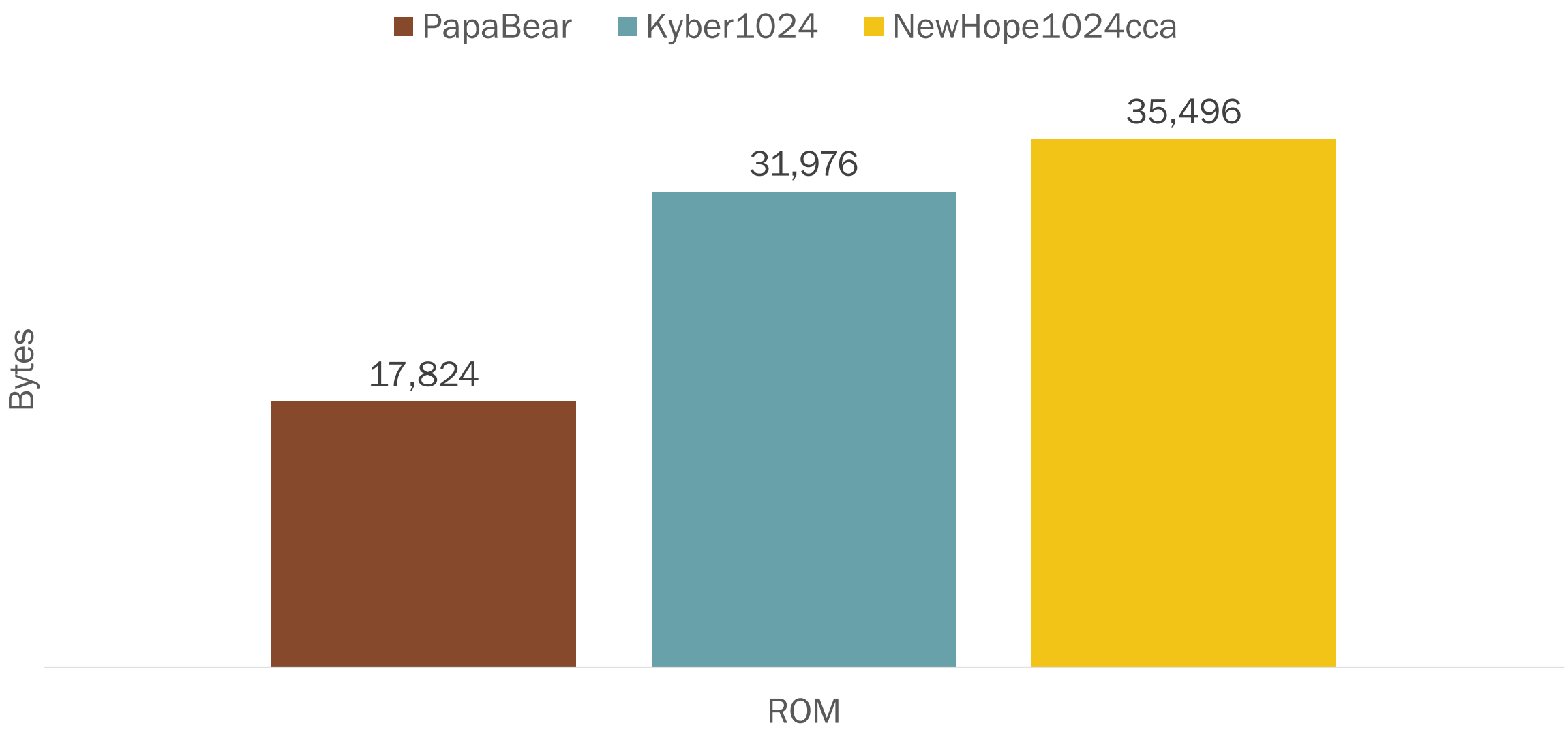
- If speed or power consumption are the greatest concerns, Kyber768 is the better choice
- If memory usage is the greatest concern, MamaBear is the better choice

Security Level V Results

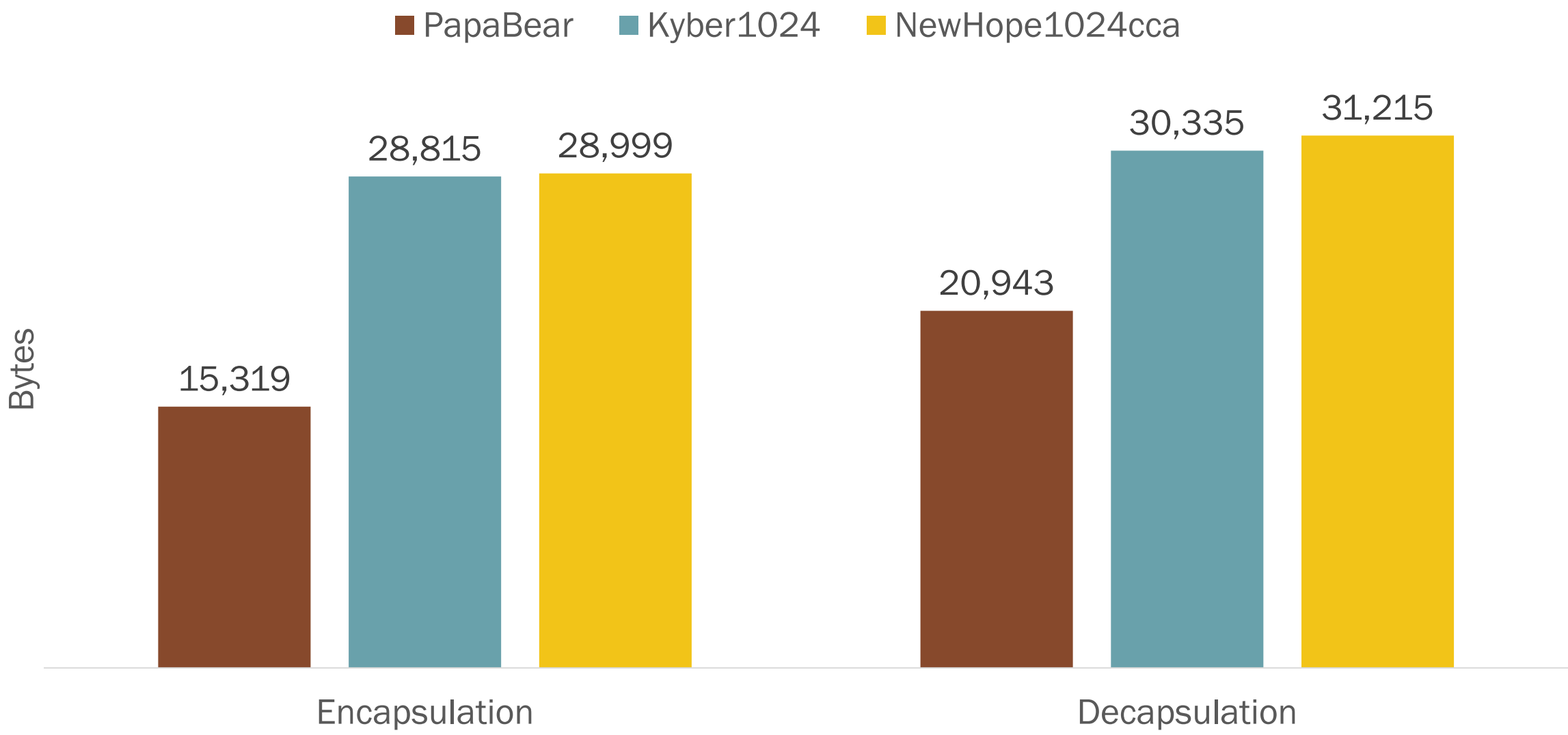
Security Level V – Speed



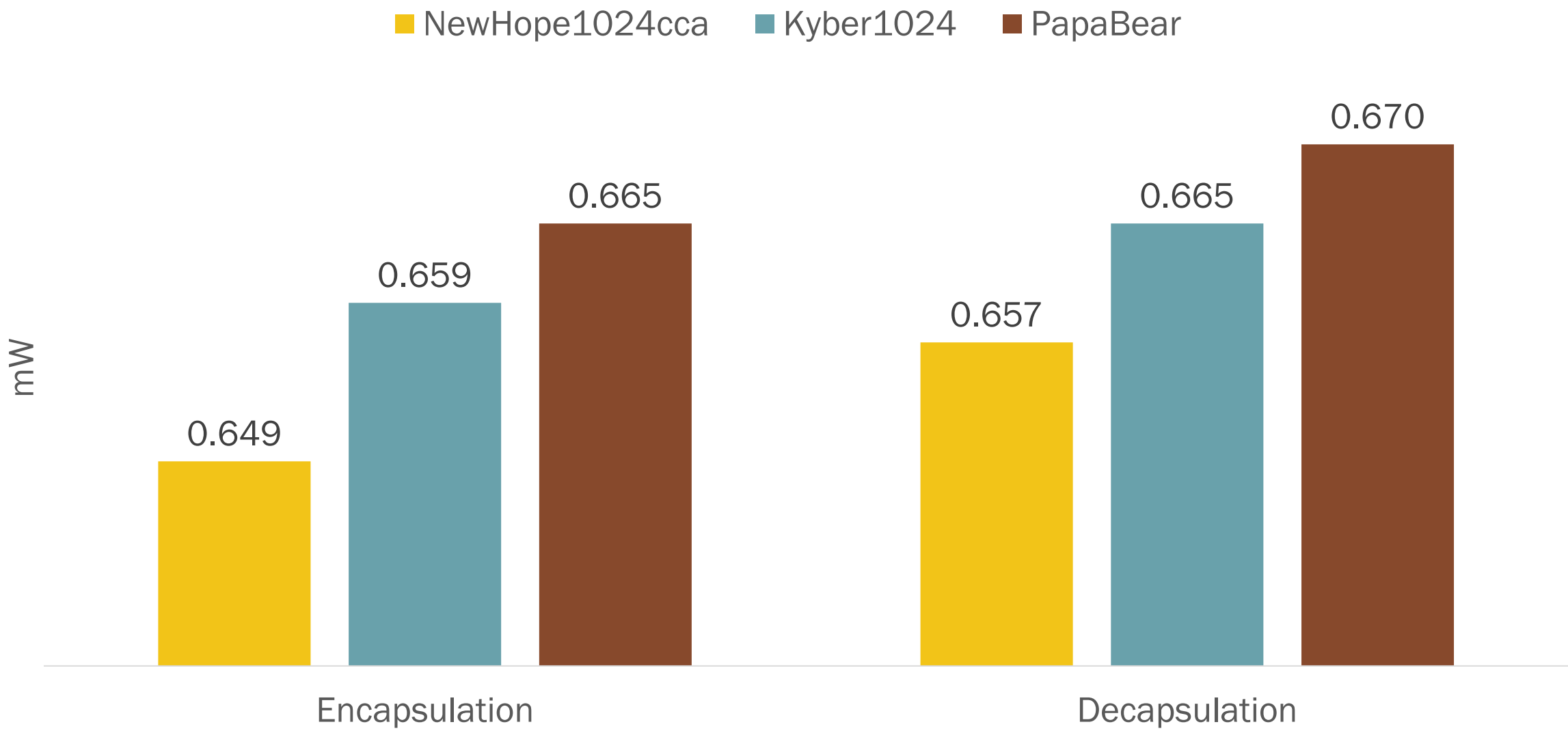
Security Level – ROM Usage



Security Level V – RAM Usage



Security Level V – Max Power



Security Level V Conclusions

- If speed is the greatest concern, Kyber1024 is the better choice
- If memory usage is the greatest concern, PapaBear is the better choice
- If power consumption is the greatest concern, NewHope1024 is the better choice

Questions?