Domain Extensions

Morris Dworkin

SHA-3 2014 Workshop

August 22, 2014

Hint in Draft FIPS 202

"All of the SHA-3 functions are designed to allow for extensions to new, separate domains that NIST may develop in the future."

Appendix A.1.2

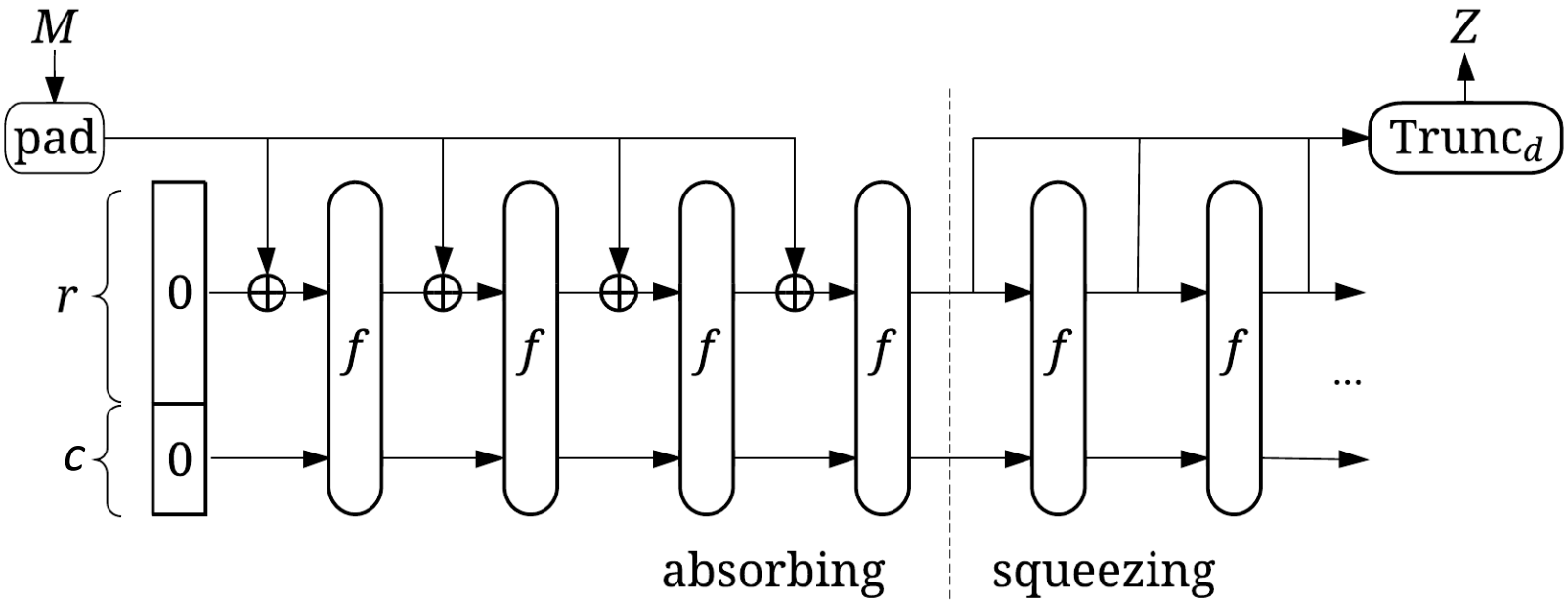
Goal

Any two distinct uses/variants/extensions of KECCAK sponge functions should produce distinct outputs, e.g.

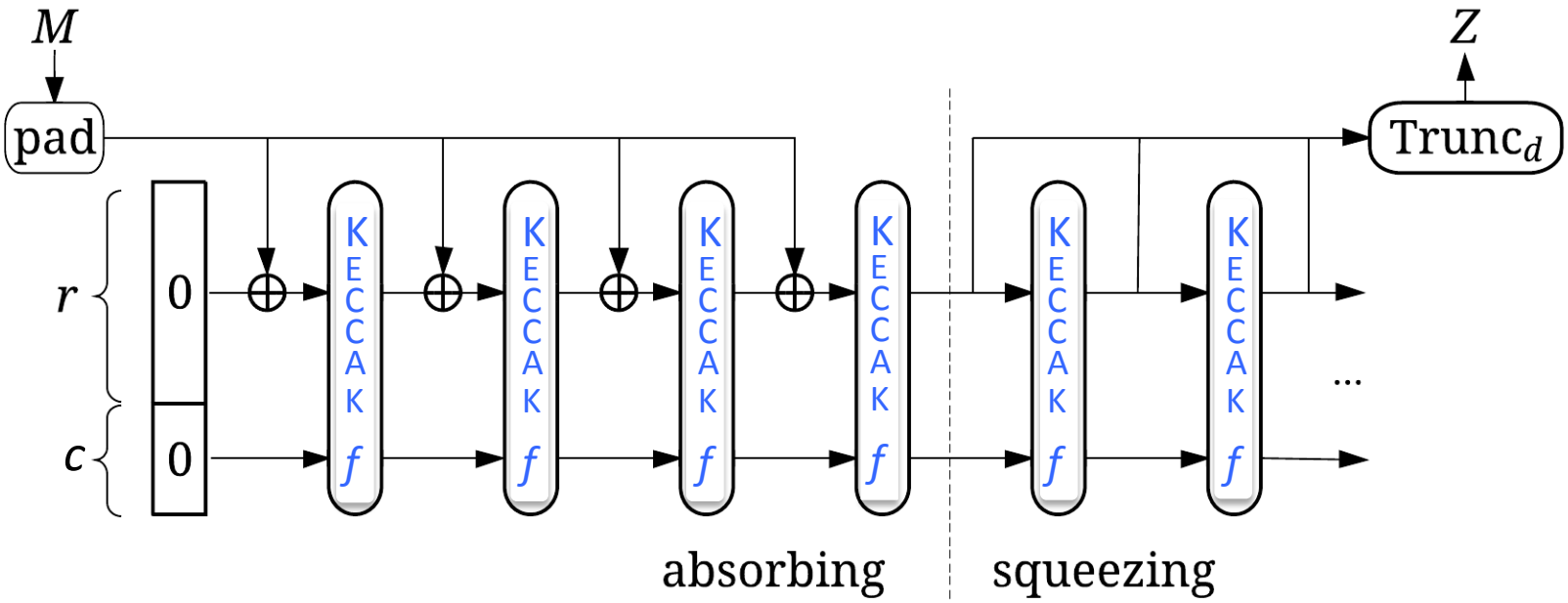
SHAKE128(*M*, *x*)
≠ MySHAKE128(*M*, *x*)
≠ YourSHAKE128(*M*, *x*)

Like a strongly typed programming language.

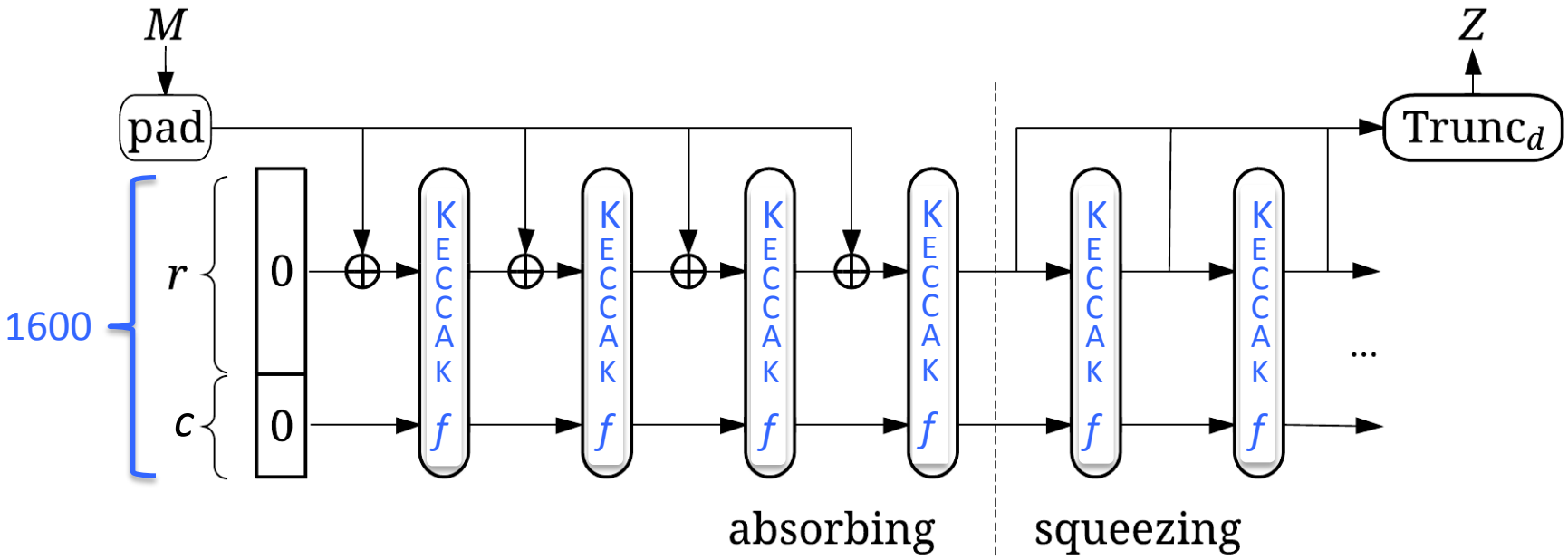
KECCAK[*c*]



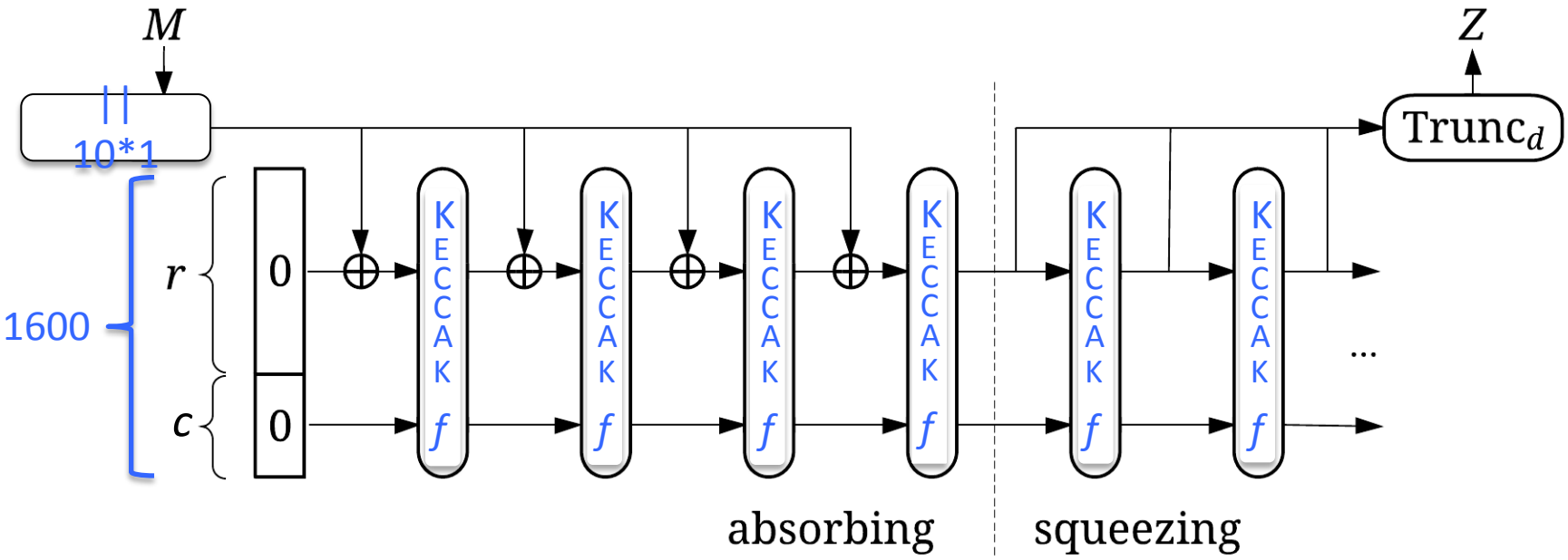
KECCAK[*c*]



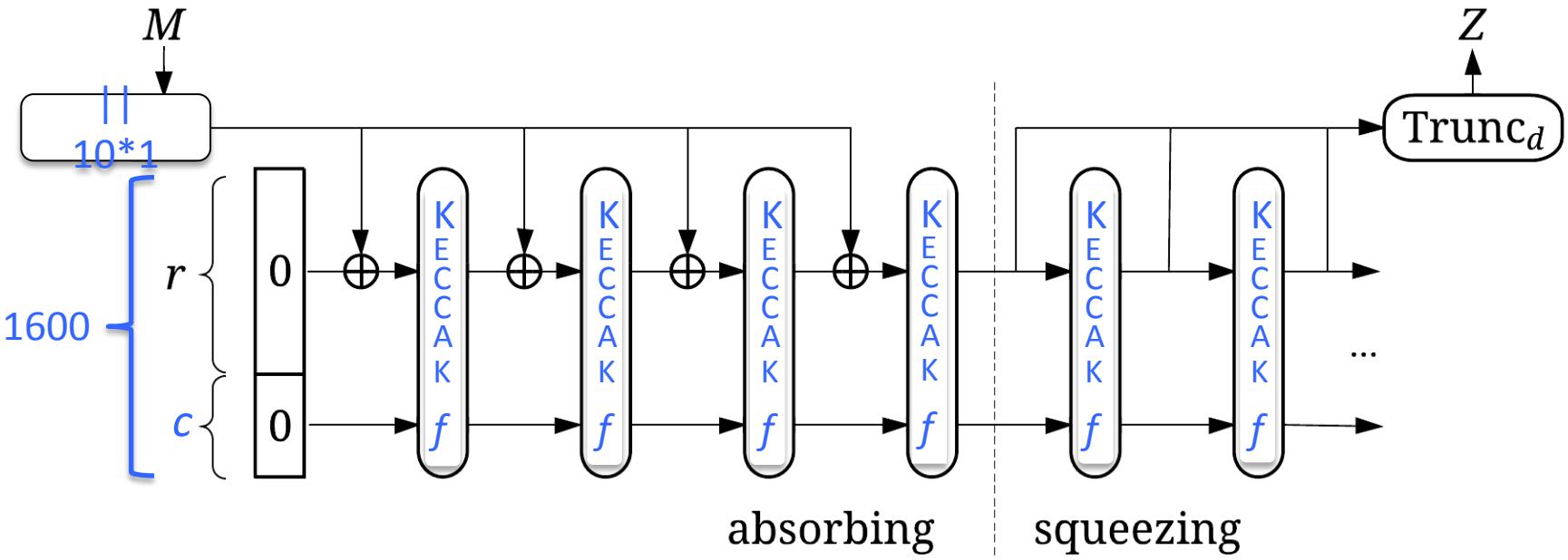
KECCAK[*c*]


M
pad
Z
Trunc$_d$
1600
r
c
0
0
KECCAK f
KECCAK f
KECCAK f
KECCAK f
KECCAK f
KECCAK f
...
absorbing
squeezing

KECCAK[*c*]


M
||
10*1
1600
r
c
0
0
KECCAK f
KECCAK f
KECCAK f
KECCAK f
KECCAK f
KECCAK f
...
absorbing
squeezing
Z
Trunc$_d$

KECCAK[c]


M
||
10*1
1600
r
c
0
0
KECCAK f
KECCAK f
KECCAK f
KECCAK f
KECCAK f
KECCAK f
...
Z
Trunc$_d$
absorbing
squeezing

Domain Separation in KECCAK[*c*]

- For uses with same capacity
  - E.g., SHA3-256, SHAKE256 (SHAKE256 variants?)
  - Domain separation from specified suffixes

$$Use\_1\_Input \,||\, Suffix_1 \neq Use\_2\_Input \,||\, Suffix_2$$

- For uses with different capacities
  - E.g., SHA3-256, SHA3-224, SHAKE128
  - Domain separation from multi-rate padding

Layers of Padding in SHAKE256

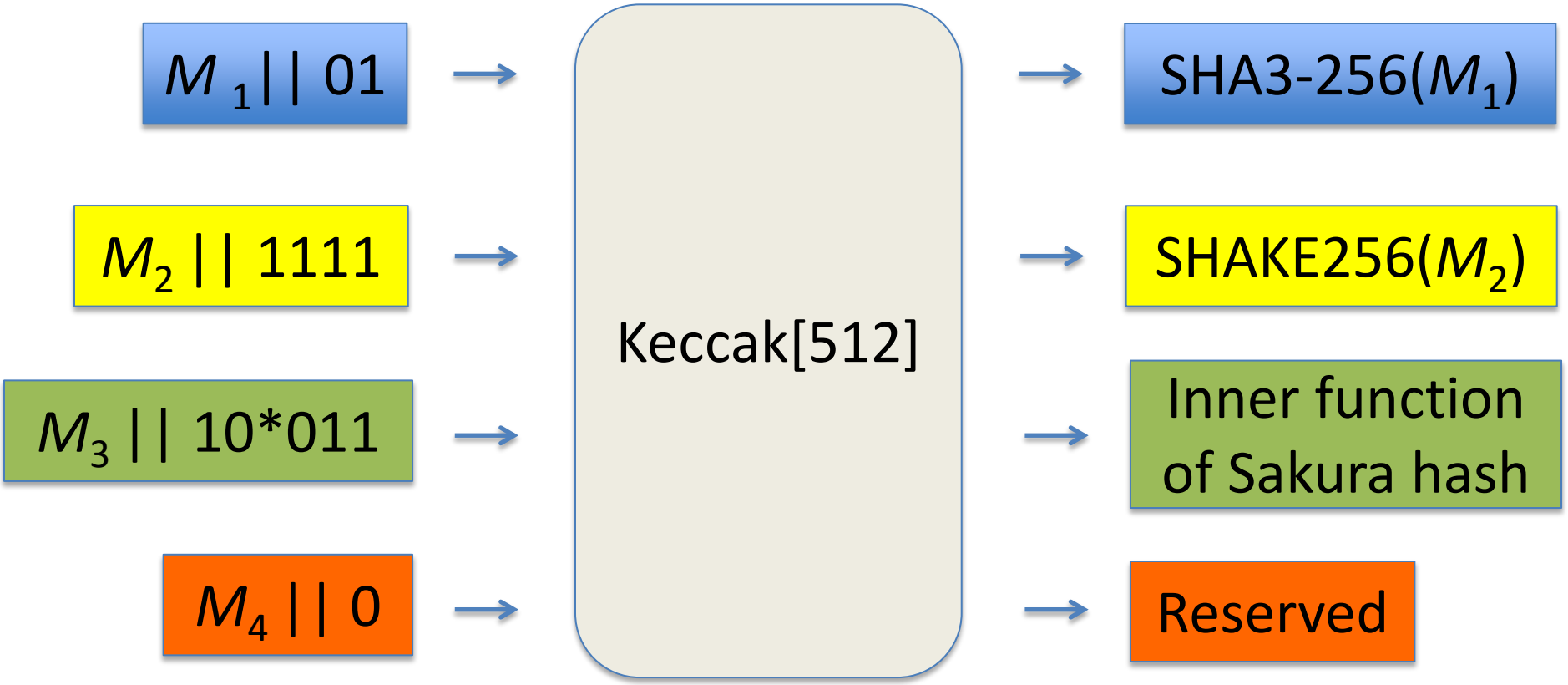
M input

M || 11 Sakura encoding (sequential)

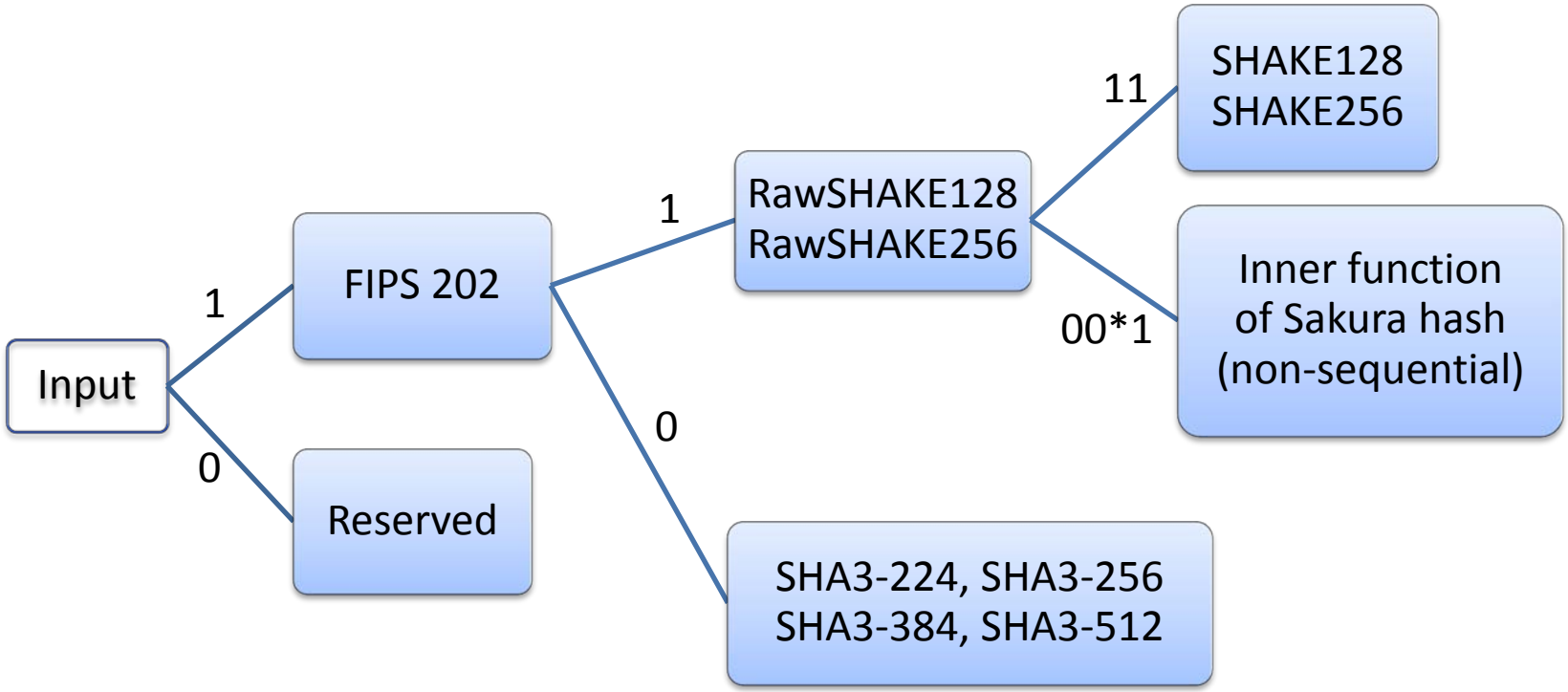
M || 11 || 11 domain suffix (RawSHAKE)

M || 11 || 11 || 10*1 KECCAK padding

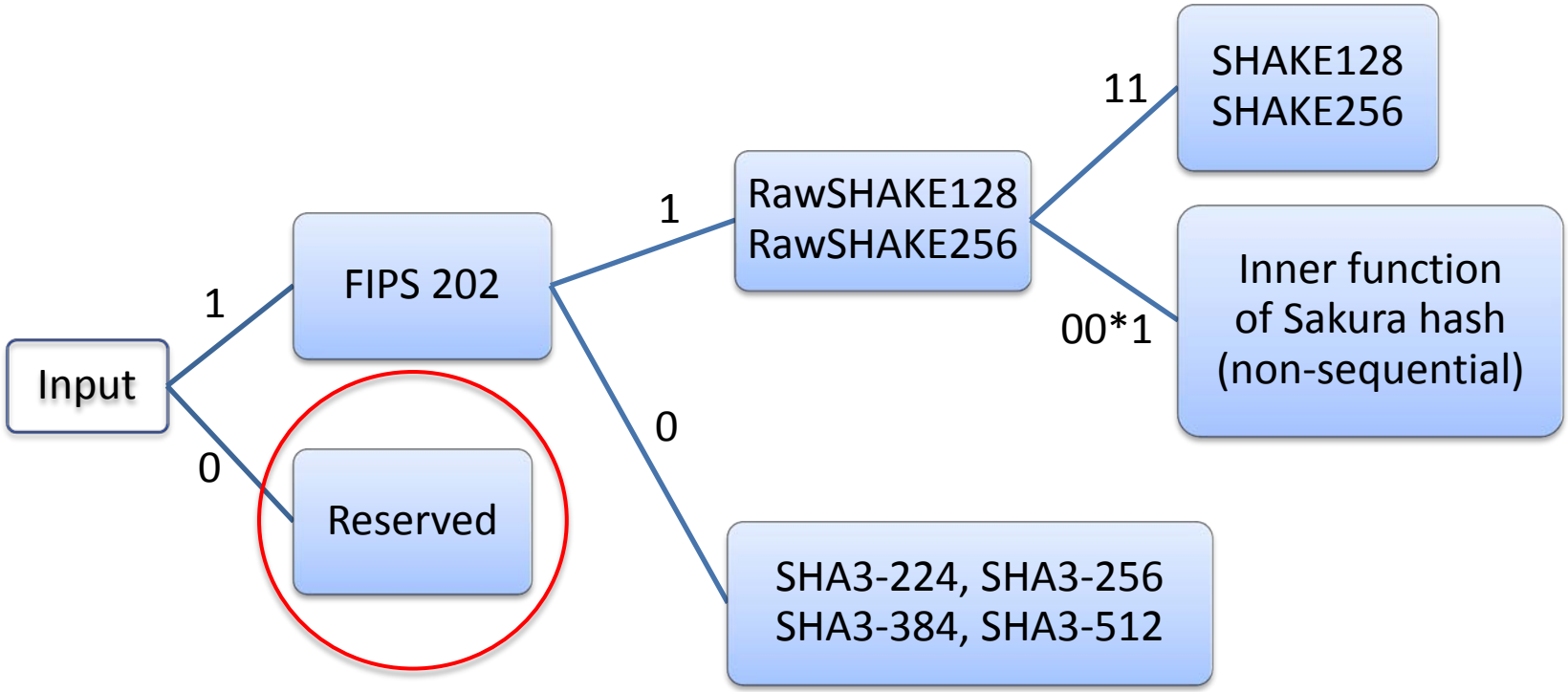
Partition of Keccak[512] Domain



Suffixes in Draft FIPS 202 (in reverse order)



Suffixes in Draft FIPS 202 (in reverse order)



What to do?

How to Define Reserved Suffixes?

- Can partition by any distinguishing feature(s):
  - Type of function
  - Type of application
  - User/Implementer
  - …
- Can associate a set of message encoding rules for each suffix
  - e.g. Sakura encoding for tree hashs

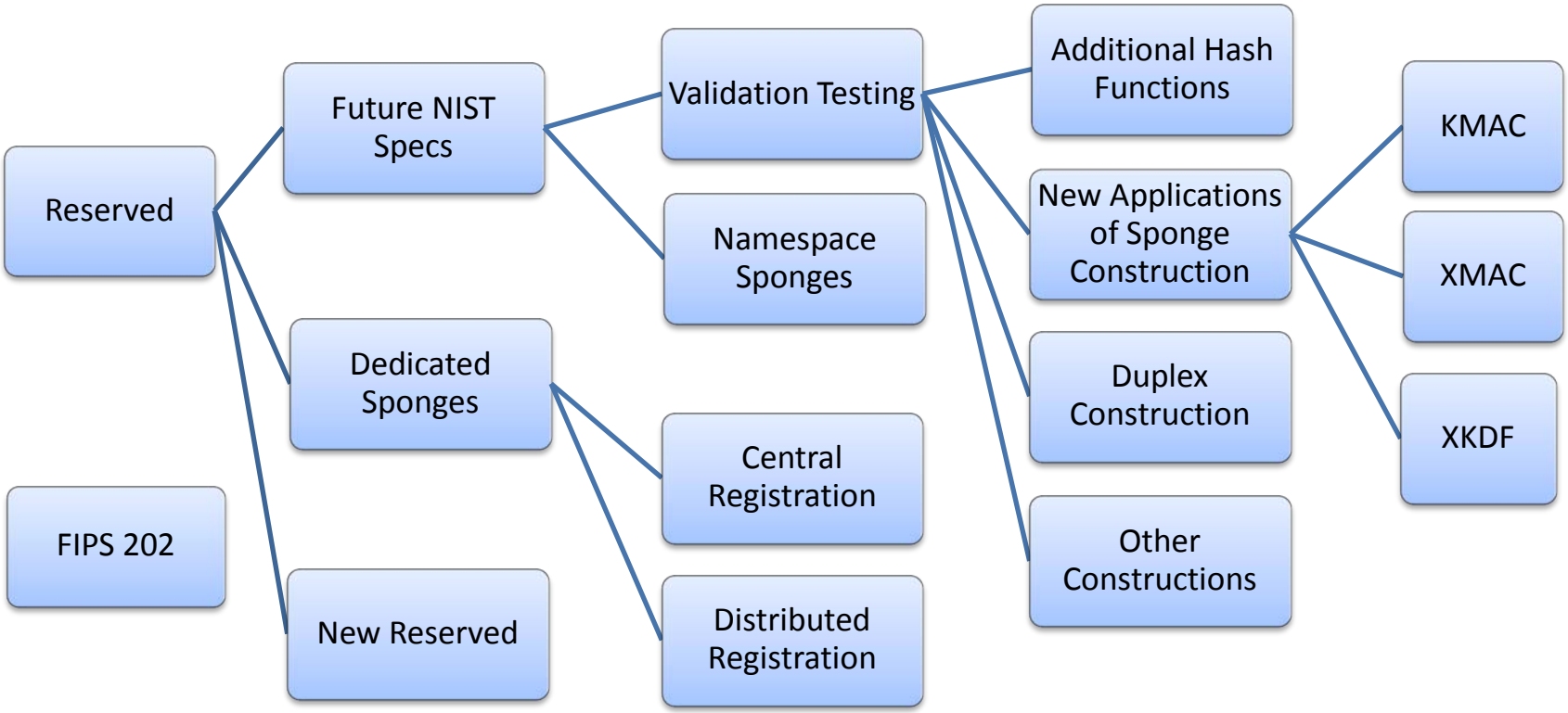
Domain Separation With Namespaces

- Suggested by KECCAK Team
  - Feb 6, 2013 Presentation at NIST
- Prepend input with namespace identifier (URI)
  - Payload syntax determined by namespace
  - Inspired from XML [http://www.w3.org/TR/REC-xml-names/]
- UTF8(URI) || 0^8 || *payload* || *domain separation suffix*

Dedicated Sponge Functions

- Even more general than namespaces
  - Replace URIs with user-defined structure
- Registration to avoid overlap
  - Central
    - NIST?
  - Distributed
    - like OIDs

Structure of Extensions (example, not a proposal)



Some Questions

- What categories of extensions?
  - Other NIST specs? Separate by applications too?
  - Namespaces?
  - Dedicated sponge functions?
    - Registration?
  - When to add Sakura encoding?
  - Which need to be NIST-approved? Tested?
  - Other suggestions?