# Asymptotic Analysis of Plausible Tree Hash Modes for SHA-3

**Kevin Atighehchi**[1] and Alexis Bonnecaze[2]

[1]Université de Caen Normandie, ENSICAEN, CNRS, GREYC, France

[2]Aix Marseille Université, Centrale Marseille, CNRS, I2M, France

FSE 2018, Belgium
March 6, 2018



# Summary

# Objectives

Choosing a tree topology is motivated by several possible objectives:

- Minimizing the parallel time (whether or not the number of processors is bounded)
- Minimizing the number of processors for an optimal parallel time
- Optimizing the memory consumption

Objectives of this work:

- Give an overview
- Propose tree topologies with the aim of:
  - Minimizing the parallel time and the number of processors for a constrained memory usage
  - Minimizing the number of processors and the memory usage for an (asymptotically) optimal parallel time

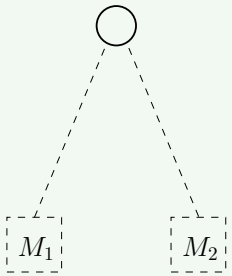
# Assumptions and notations

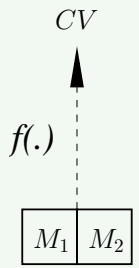
The *outer* hash function is constructed on top of an *inner* function $f$ according to a tree mode.

Assumptions :

- PRAM model with EREW strategy
- The operated *inner function* is a serial hash function
- Processing a $k$-ary node costs $\Theta(k)$ units of time
- For simplicity, all the leaves are at the same depth
- To give substance to the constructed outer hash function, we assume the use of SAKURA coding

## Representations of a node (coding bits ignored)



# Memory usage of a sequential execution

We refer to the *auxiliary space* required for executing the hash function, *i.e.* not counting the message.

Memory usage $=$ number of hash states in memory

- For a serial hash function, it is one hash state
- For a tree-based hash function, $O(h)$ hash states where $h$ is the tree height
  - Compute the *highest node first*
    - Strategy used in *Merkle tree traversal*
  - For nodes of high arity, use a threshold value $d$ to trigger the consumption of CVs

# Parallel time

Further notations:

- $t(m)$ denotes the sequential running time to process $m$ chaining values or message blocks
- $T(\mathrm{R})$ denotes the parallel time to process a node $\mathrm{R}$ and all its descendants

Let us consider a node, denoted $\mathrm{R}_*$, and its children, denoted $\mathrm{R}_i$ for $i = 1 \ldots k_{R_*}$, indexed in the order in which their corresponding chaining values are processed in $\mathrm{R}_*$. We have

$$T(\mathrm{R}_*) \approx \begin{cases} t(k_{R^*}) & \text{if } \mathrm{R}^* \text{ is a base level node,} \\ \max\limits_{i=1\ldots k_{R_*}} \{T(\mathrm{R}_i) + t(k_{R_*} - i + 1)\} & \text{otherwise.} \end{cases}$$

# An overview of tree hash modes

About the choice of a tree mode, two camps:

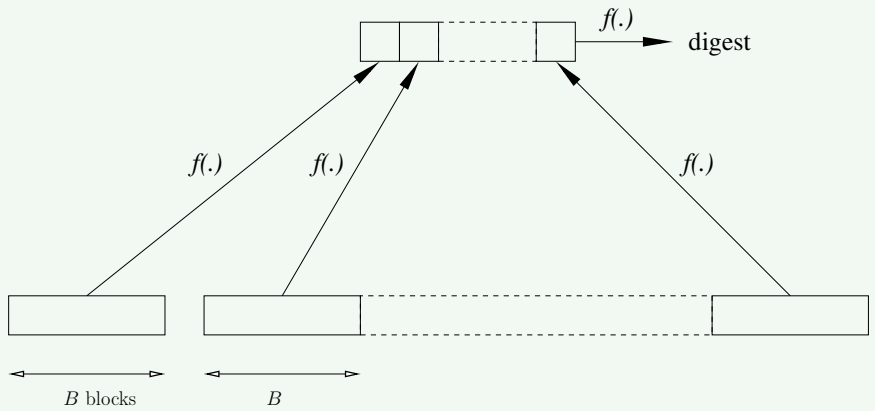
1. Think about general-purpose and low-end computers
2. Think about the potential speedup

One invariant in the discussions: a single tree mode is desired to fit all the computing platforms.

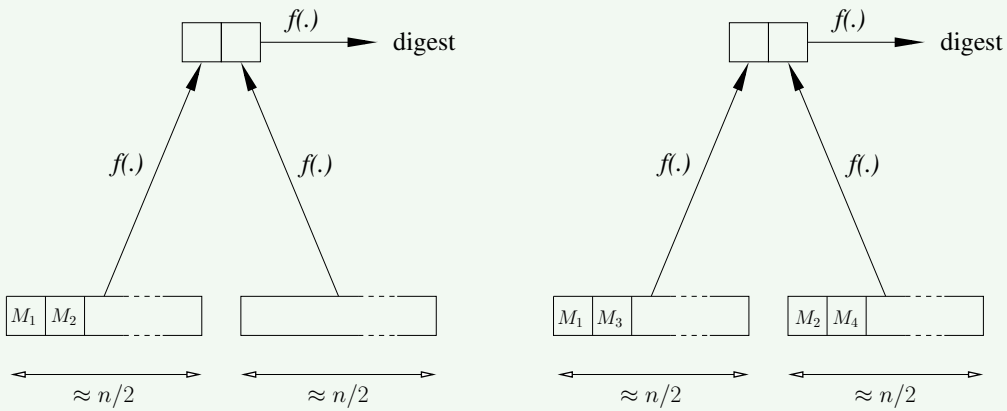
# 2-level trees

Choice between scalability and reduced root node arity:

- Scalability (*e.g.* SP 800-185 ParallelHash; KangarooTwelve)

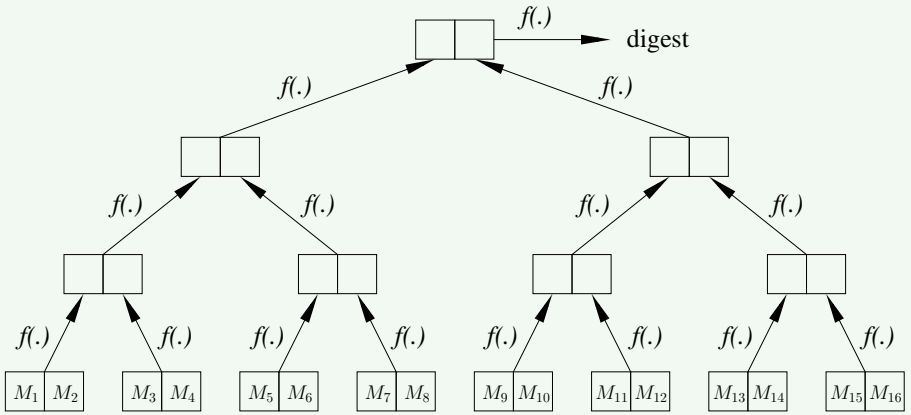



- Something dedicated to a fixed number $q$ of processors, *e.g.* $q = 2$

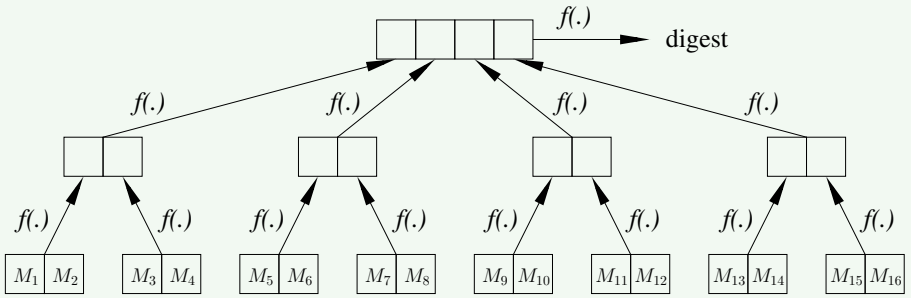


# Classic balanced $k$-ary hash tree

- w/o height restriction ($k = 2$)




- with height restriction ($k = 2$, $h \leq 3$)



# Asymptotic efficiency (in Big-O)

Asymptotic efficiency using Big-O notation of existing tree hash modes:

| Mode | Live streaming | Memory (sequential) | Number of processors | Parallel running time (ideal case) | Comments |
|---|---|---|---|---|---|
| 1 | ✓ | $1$ | $n$ | $n$ | root of unlimited arity |
| 2S | - | $1$ | $q$ | $n/q$ | not scalable (but reduced amount of work) |
| 2L | ✓ | $q$ | | | |
| 3 | ✓ | $\log n$ | $n$ | $\log n$ | tree of unlimited height |

where $n$ is the number of blocks of the message and $q$ is a fixed number of processors. ($q$ not hidden in the Big-O)

Criterias:

- Memory consumption of a sequential execution
- Number of processors to reach the lowest/ideal parallel running time
- The parallel time referring to the lowest running time achievable with the aforementioned number of proc.

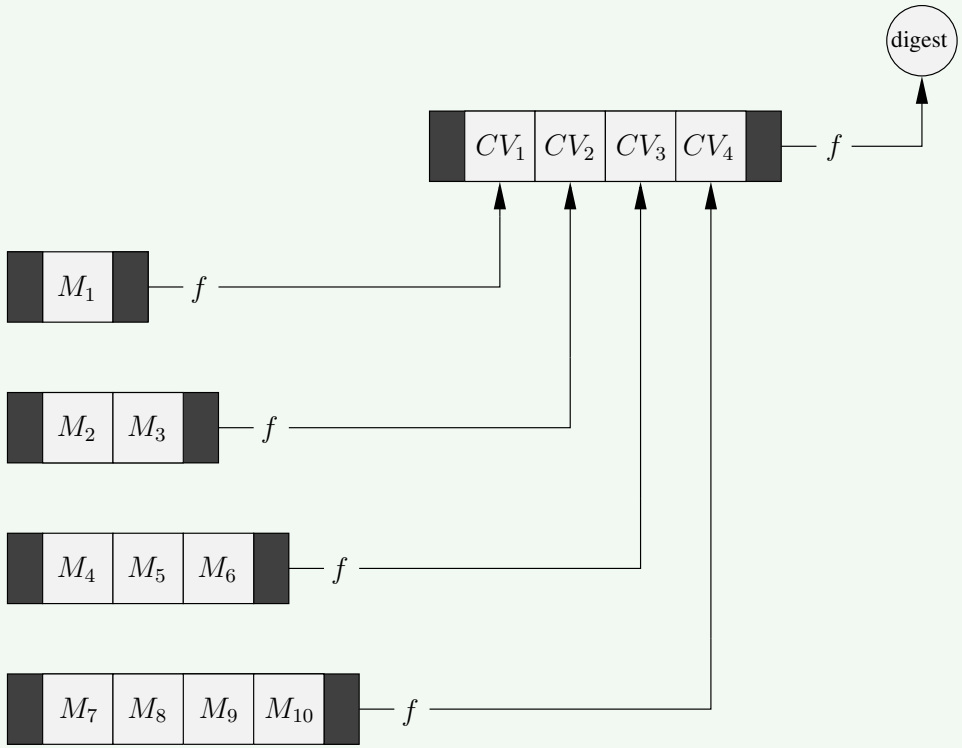
# New topologies for new complexity results

New plausible modes for streaming stored content (4S, 5S and 6S) and their variants for streaming live content (4L, 5L, 6L). Their efficiencies are expressed in Big-O.

| Tree mode | Live streaming | Memory usage (sequential) | Number of processors | Parallel running time (ideal case) | Priority |
|---|---|---|---|---|---|
| 1 | ✓ | $1$ | $n$ | $n$ | memory consumption |
| 2S | - | $1$ | $q$ | $n/q$ | memory and total work (for a fixed $q$) |
| 2L | ✓ | $q$ | | | |
| 3 | ✓ | $\log n$ | $n$ | $\log n$ | parallel time |
| 4S | - | $1$ | $n^{1-\epsilon}$ | $n^{\epsilon}$ | memory consumption |
| 4L | ✓ | | | | |
| 5S | - | - | - | - | none (any trade-off) |
| 5L | ✓ | | | | |
| 6S | - | $\log n - \log\log n$ | $\frac{n}{\log n}$ | $\log n$ | parallel time |
| 6L | ✓ | | | | |

# A tree mode for live streaming (Mode 4L)

## Processing of 10 blocks with a 2-level tree




Observations:

- Increasing node arities at level 1
- Number $k$ of CVs is $\arg\min_k \sum_{i=1}^{k} i \geq n$, *i.e.* $k = O(n^{1/2})$
- Memory consumption of a serial implementation: $O(1)$

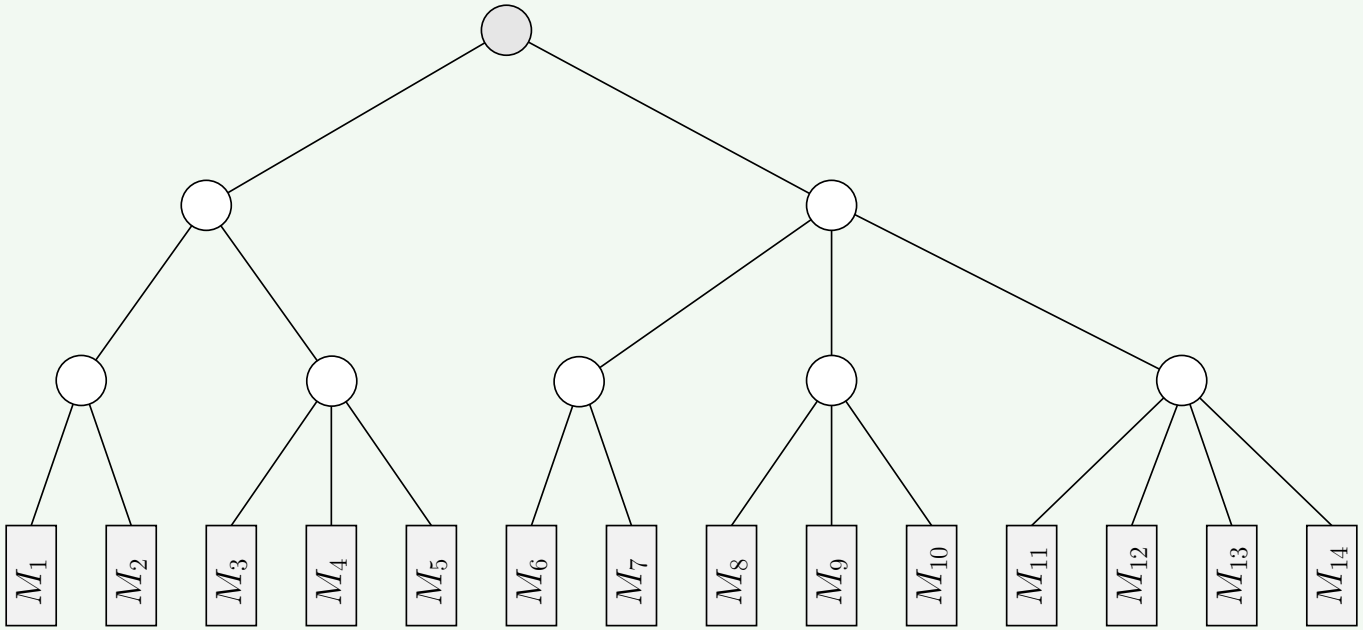
# Generalizing for a constant tree height

The underlying tree can be defined recursively until a given tree height $h$ is reached:

- Initially (before tree pruning), the root node is assumed to be of arity $\infty$
- Let $k$ the arity of a descendant node, its $i$-th child is of arity $i+1$, for $1 \leq i \leq k$
- The message blocks being at depth $k$, prune the unneeded branches

# Generalizing for a constant tree height

## Processing of 14 blocks with a 3-level tree




## Theorem

*Given $\epsilon = 1/h$ where $h$ is the height, Mode 4L has an ideal parallel running time in $O\left(n^{\epsilon}\right)$ by using only $O\left(n^{1-\epsilon}\right)$ processors, and requires $O\left(1/\epsilon\right) = O(1)$ hash states in memory to process the message with a single processor.*

# Generalizing for a constant tree height (Ctd)

Let $n$ the number of message blocks.

Steps to determine the parallel time:

- Once pruned, let $x$ denote the root node arity, $a_{br}$ the arity of the righmost node at the base level, and $h$ the height of the tree
- By construction of the tree, $a_{br} < x + h$
- Let $T$ be the parallel time to compute the hash. $T = O(a_{br} + h) = O(a_{br}) = O(x)$ since $h$ constant
- Let $n_{max}(x)$ be the maximum number of blocks covered by the tree when pruning only the edges at the root level. We show that

$$n_{max}(x) \sim \frac{x^h}{h!} \text{ and } n_{max}(x) \sim n.$$

As a result, $x = O(n^{1/h})$

# Generalizing for a constant tree height (Ctd)

Let $w$ be the number of involved processors.

Remaining steps to determine $w$:

- Let $w_{max}(x)$ be the number of nodes at level 2 when pruning only the edges at the root level. We have

$$w_{max}(x) \sim \frac{x^{h-1}}{(h-1)!}$$

- Because $w_{max}(c \cdot x) = O(w_{max}(x))$ and $x = O(n^{1/h})$, it follows that

$$w = O(n^{\frac{h-1}{h}})$$

# Concluding remarks

Pending issues:

- Asymptotic complexities in $O$?! Why not exact complexities?
  - They depend on the inner function parameters
  - Sometimes, only numerical results can be expected
  - Analytical or numerical result $\neq$ measurement in practice
- To what extent these topologies can be used? *E.g.*
  - Evaluation of software implementations
  - Hybrid Multithreaded/SIMD implementations
- The question of a parallelizable output extension?

Thanks for your attention...

# Questions ?