Advancing the AFF4 to the challenges of Volatile Memory & Single Hashes

Dr. Bradley Schatz

Director, Schatz Forensic

v1.0 – DFRWS - 2017

Agenda

- AFF4 background
- What can be improved with hashing in AFF4?
- Our refined single hashing proposal: the *Linear Chunk Hash*

Traditional forensic imaging introduces unnecessary delays

- Physical acquisition delays analysis by hours
  - Doesn't scale to multi-core/RAID/SSD/NVMe
  - Doesn't allow non-linear writes (concurrent imaging & analysis)

- Possible solution: logical imaging/triage
  - loses context and unallocated

Traditional forensic imaging doesn't scale

- ZIP/Deflate compression is really CPU expensive
  - (~ 40MB/s/core for high entropy)
- Hashing is CPU expensive
  - (~ 600MB/s/core for 2016 i7)
- ***Linear bitstream hashing isn't parallelizable***
  - ***Ceiling of around 600MB/s on current generation i7 CPU's***

The AFF4 removes the bottleneck from forensic imaging

- Snappy compression is CPU inexpensive
  - (> 1000 MB/s/core)
- Hashing is still CPU expensive
  - (~ 600MB/s/core for 2016 i7)
- AFF4 block hashing is parallelizable
  - Seeing > 1000 MB/s on current generation CPU's

The AFF4 removes enables acquisition to occur at the same time as analysis

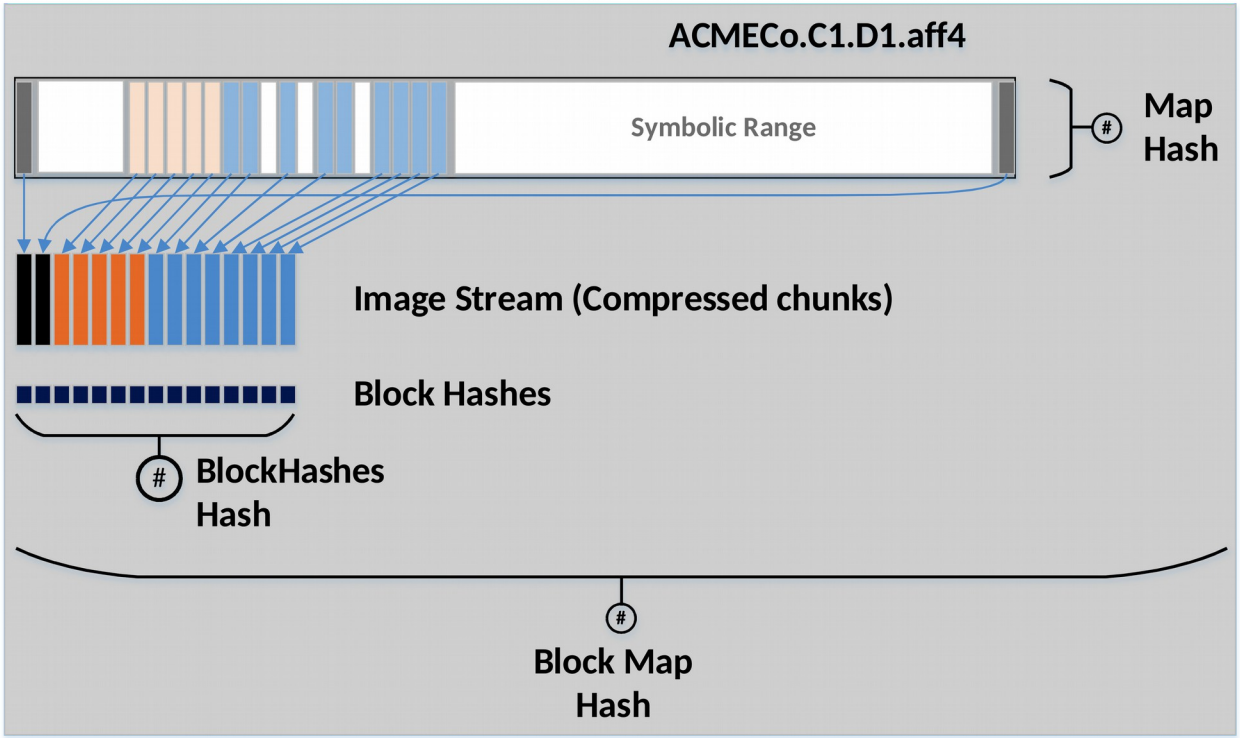
- Non-linear, potentially partial compressed block stream
- Hashing mechanism that enables incremental hash calculation

AFF4 Standard v1.0 Hashing

- Defines a hash construction called the **Block Map Hash**
  - Single hash for image

- Hybrid of:
  - Block (segment) hashing for concrete data chunks
  - Non concrete data chunks (eg. Sparse) protected by hash on map
  - Merkel tree like hash hierarchy

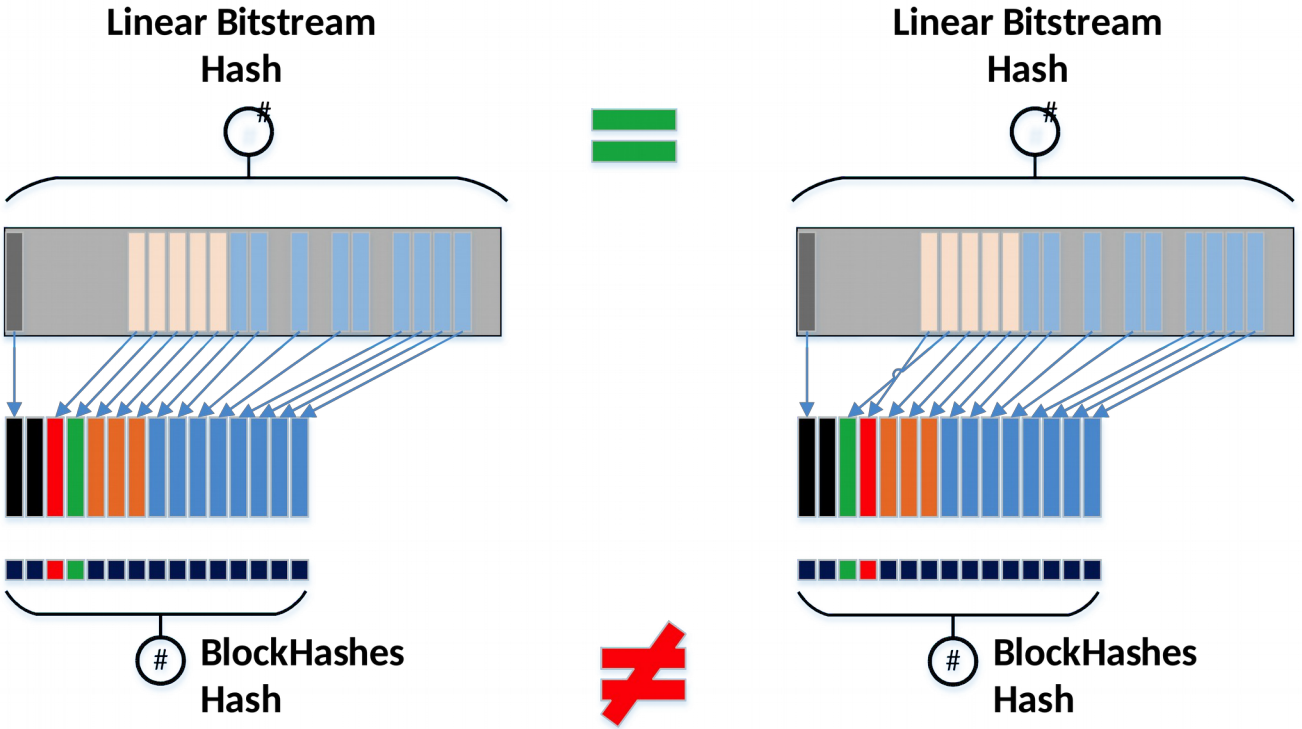
AFF4 Standard v1.0 Hashing



Desirable properties of hashing

	Linear Bitstream Hash	AFF4 Block Map Hash
Hash is concise	**Yes**	**Yes**
Hash of image matches hash calculated when imaging	**Yes**	**Yes**
Changed image results in failed hash match	**Yes**	**Yes**
Successive hashes of same source result in same hash	**Yes**	***Maybe***

Parallelism may lead to differing hashes



Can we generate a block based hash that is not subject to reordering ?

- Hypothesis 1: Reorder the lower Block Hashes based on their upper address
  - Works well for images of disks with no read errors.
  - Doesn't work so well for:
    - Disk images with read errors
    - *Memory images with chunk sizes > memory page size*

Sub chunk size data introduces further complication

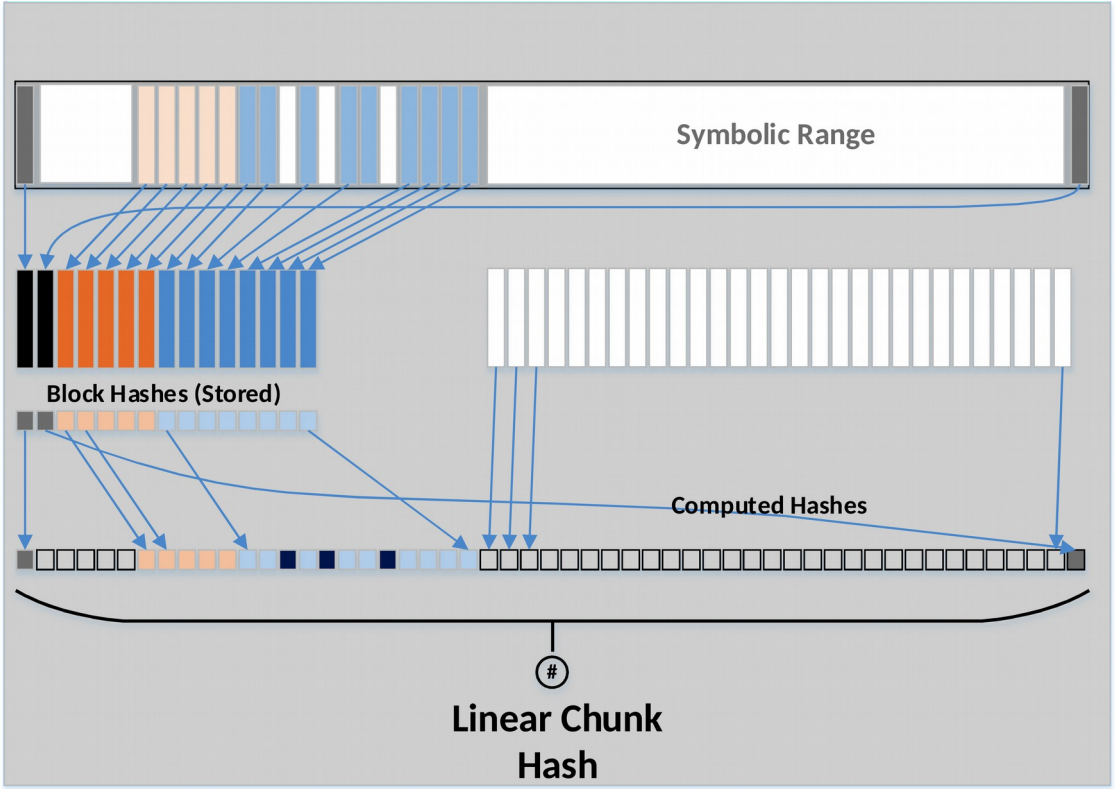
- High performance requires (chunk size > disk sector size)
  - IO Throughput - 1MB for disk IO
  - Memory consumption- Maps

- ddrescue bad sector imaging algorithm requires chunks of 512B or 4KB when the chunk size might be many multiples of this
- RAM physical address space holes lead to sub chunk sized data runs

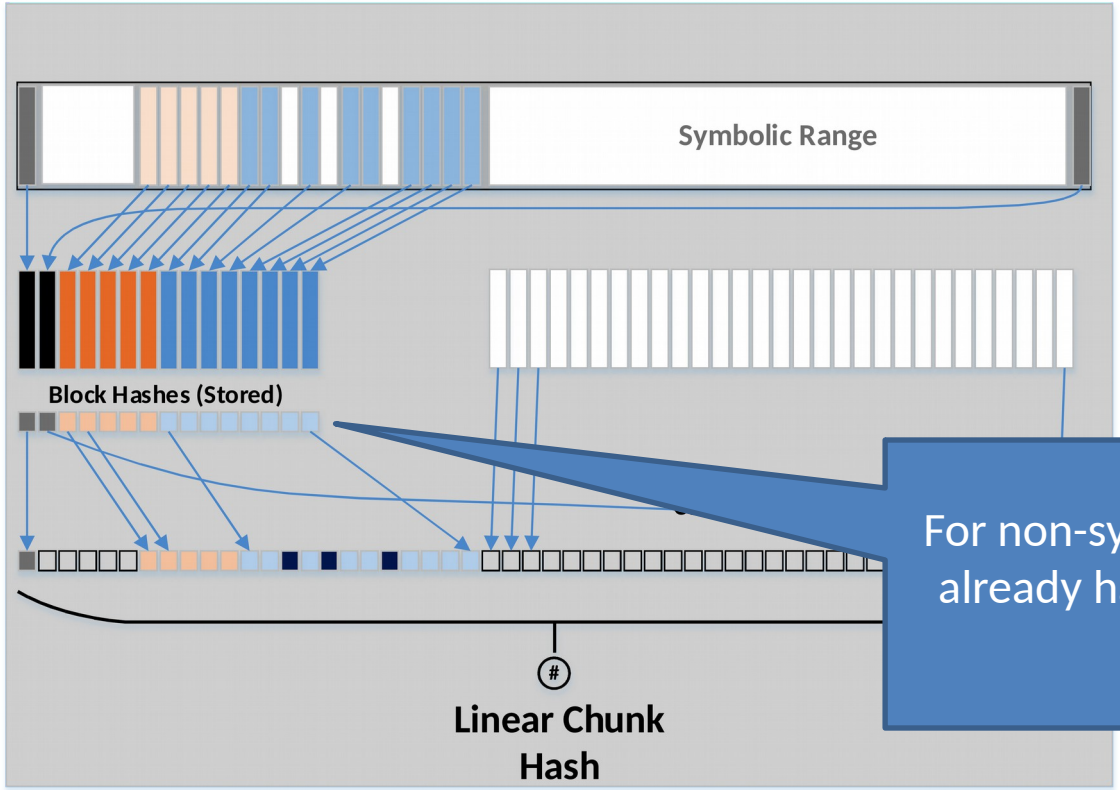
Hypothesis 1a: introduce sub chunk encoding rules

- 4k sector gets written to first 4k of chunk, rest zero filled.
- -or-
- 4k sectors coalesced

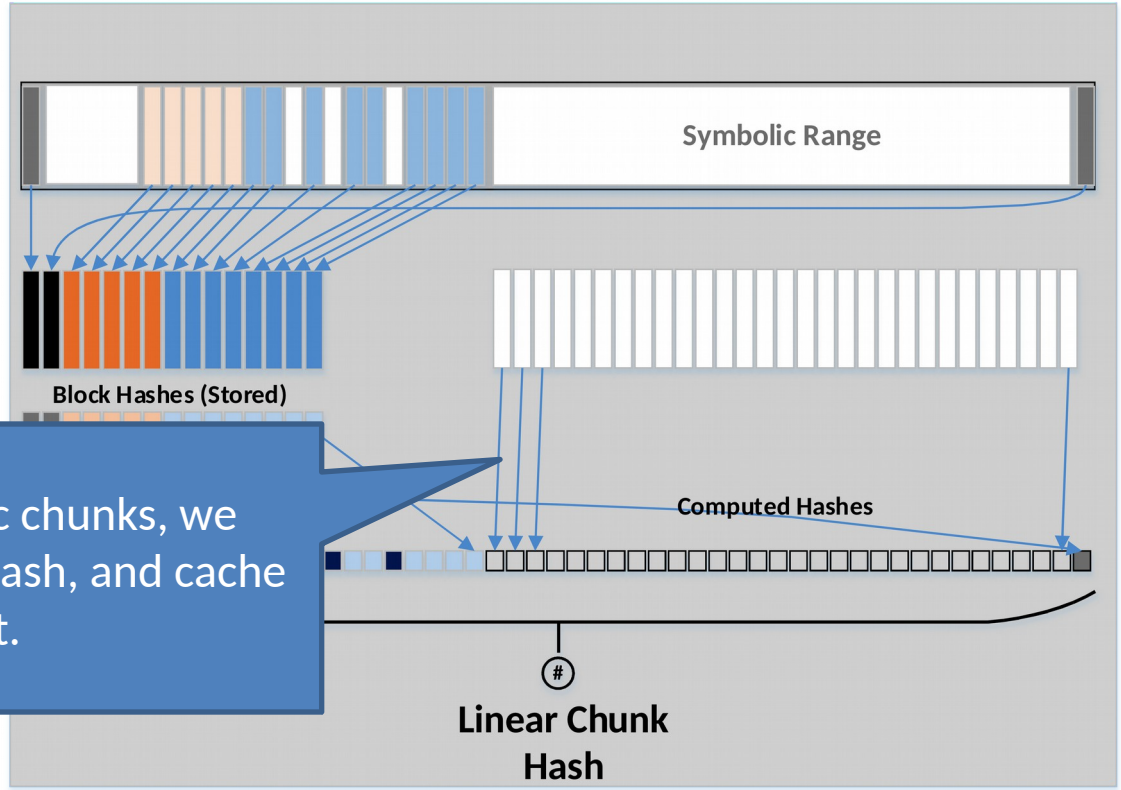
Hypothesis 2: hash all chunks of the source



Hypothesis 2: hash all chunks of the source

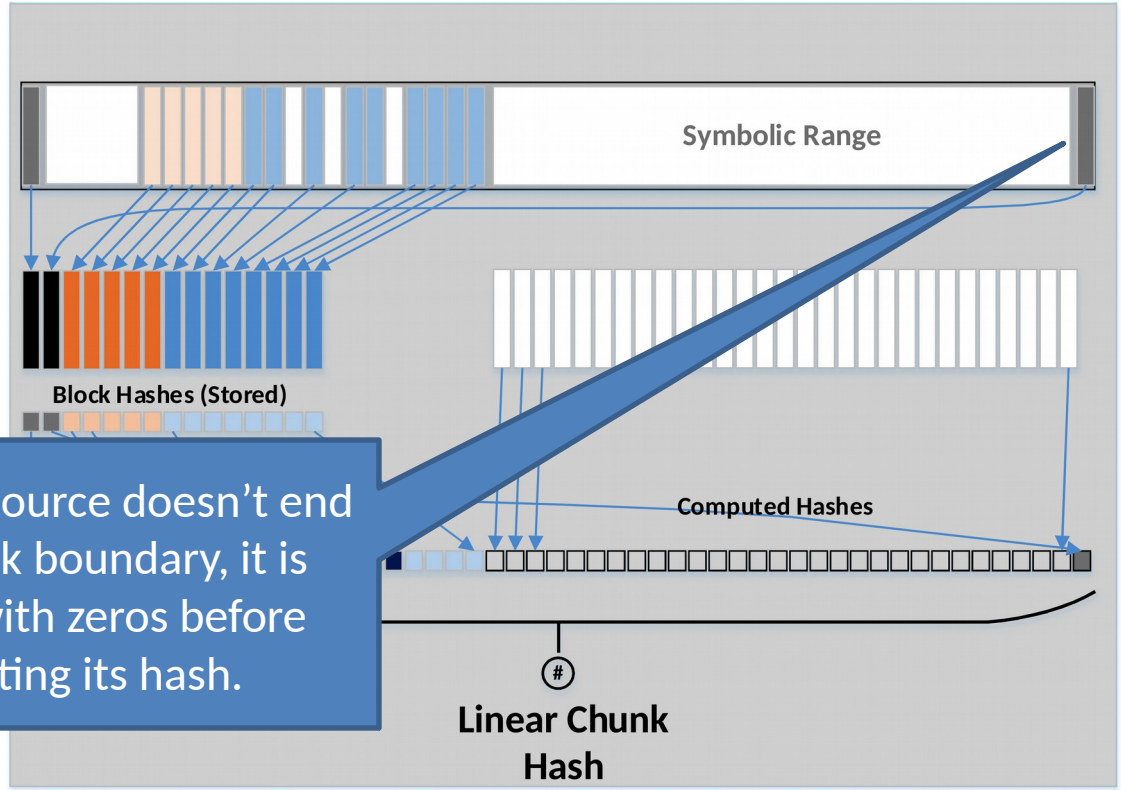


Hypothesis 2: hash all chunks of the source



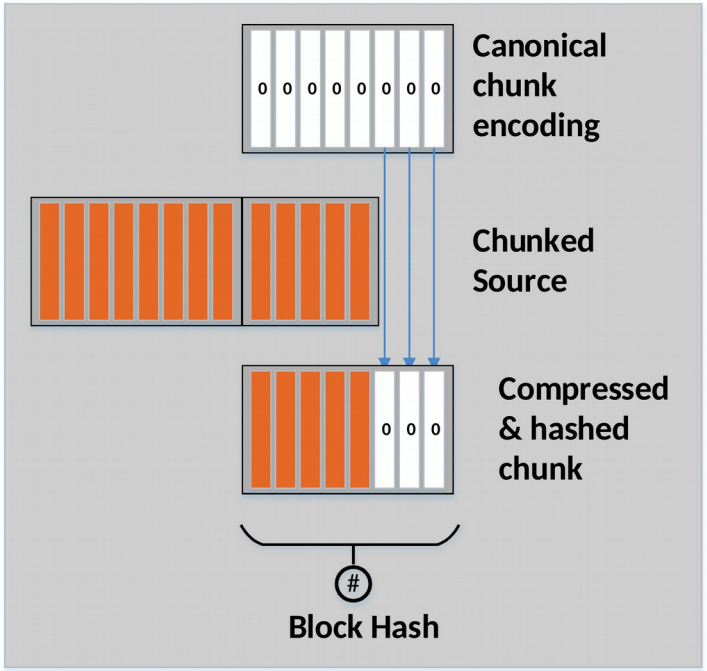
For symbolic chunks, we calculate the hash, and cache it.

Hypothesis 2: hash all chunks of the source

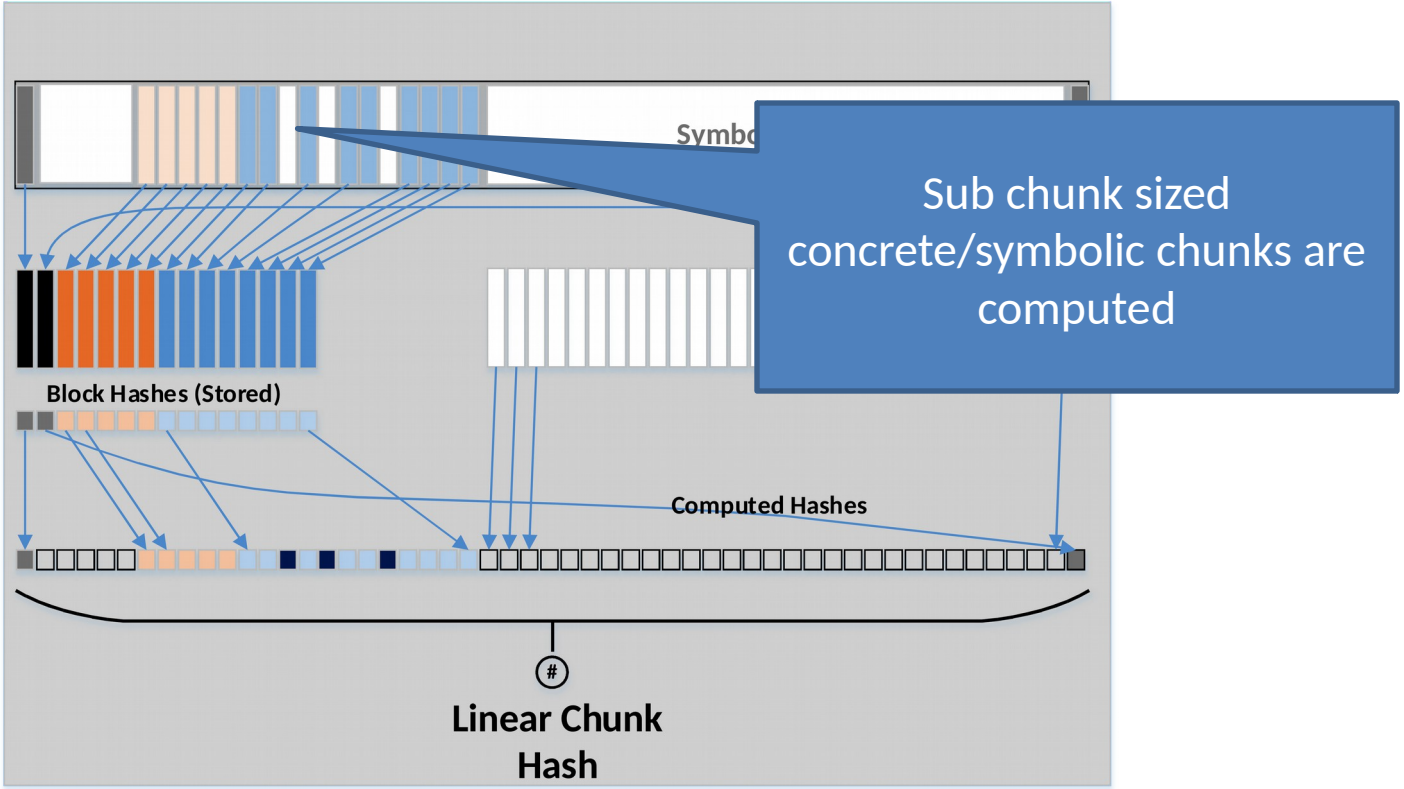


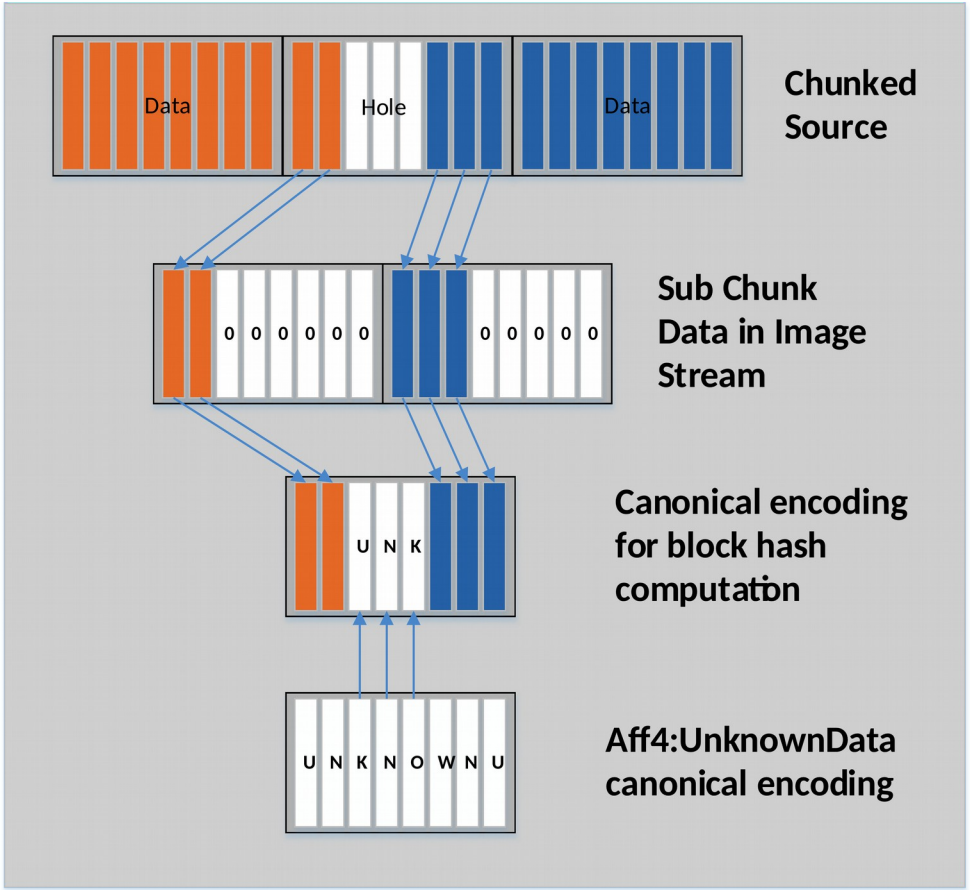
Where the source doesn't end on a chunk boundary, it is padded with zeros before generating its hash.

Sub chunk sized end of stream is zero filled



Hypothesis 2: hash all chunks of the source

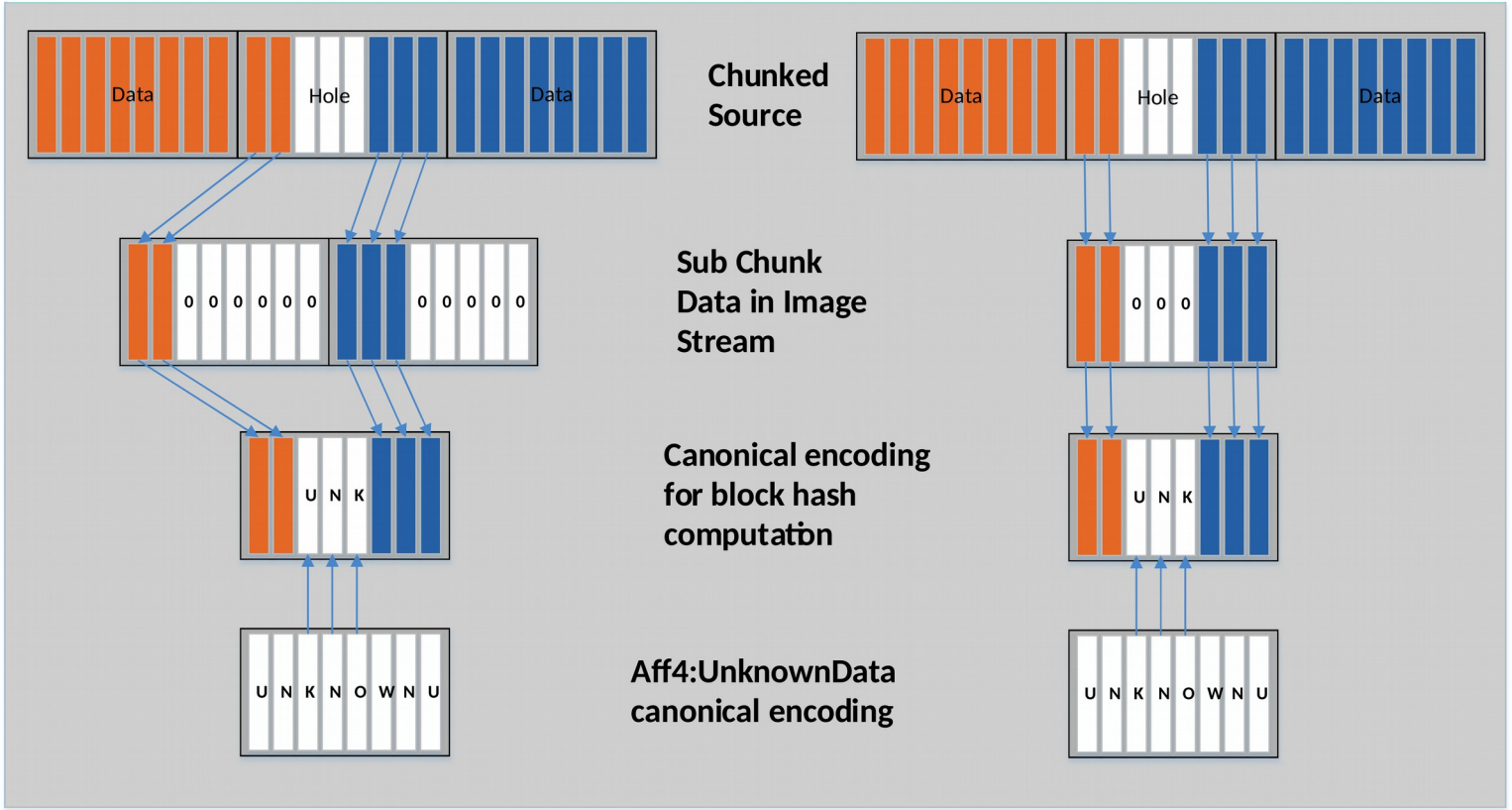




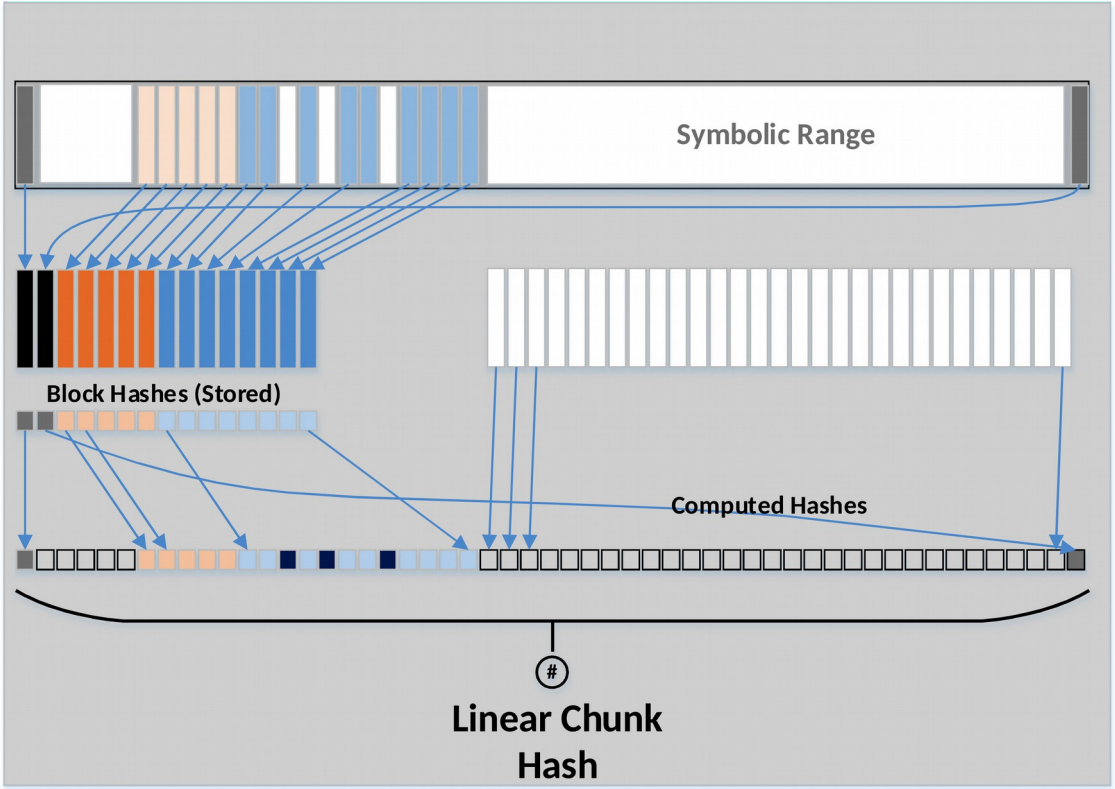
Sub chunk symbolic encoding rules

- Partial chunks stored padded
- Block hash computed from canonical block encoding

Alternate part chunk storage schemes result in the same hash



Hypothesis 2: hash all chunks of the source



Desirable properties of hashing

	Linear Bitstream Hash	AFF4 Block Map Hash	AFF4 Linear Chunk Hash
Hash is concise	Yes	Yes	Yes
Hash of image matches hash calculated when imaging	Yes	Yes	Yes
Changed image results in failed hash match	Yes	Yes	Yes
Successive hashes of same source result in same hash	Yes	Maybe	*Yes*

Contact

Dr Bradley Schatz
https://evimetry.com/
bradley@evimetry.com
@blschatz
@wirespeed4n6

'Hard Disk Drive X-Ray'
image by Jeff Kubina