iSHAKE: Incremental Hashing With SHAKE128 and SHAKE256 for the Zettabyte Era

Danilo Gligoroski and Simona Samardjiska
Department of Telematics,
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and TechnologyTechnology - NTNU, NORWAY

NTNU
Innovation and Creativity

Incremental cryptography

- Introduced 20 years ago in Crypto '94 paper by Bellare, Goldreich and Goldwasser, *"Incremental Cryptography: The Case of Hashing and Signing"*
- Bellare, Goldreich and Goldwasser, STOC '95, *"Incremental Cryptography and Applications to Virus Protection"*
- Bellare and Micciancio, *"A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost"*, Eurocrypt '97

NTNU
Innovation and Creativity

Incremental cryptography

- Introduced 20 years ago in Crypto '94 paper by Bellare, Goldreich and Goldwasser, *"Incremental Cryptography: The Case of Hashing and Signing"*
- Bellare, Goldreich and Goldwasser, STOC '95, *"Incremental Cryptography and Applications to Virus Protection"*
- Bellare and Micciancio, *"A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost"*, Eurocrypt '97

WARNING:

NTNU
Innovation and Creativity

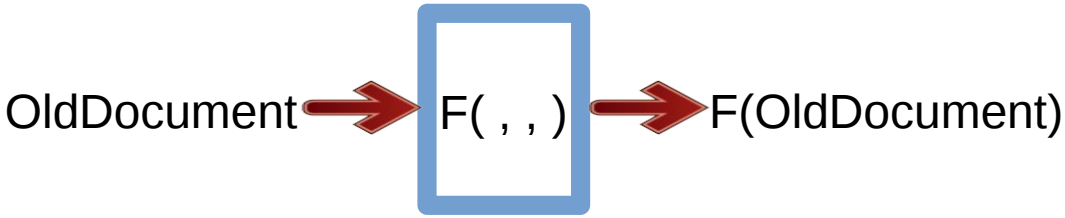
Basic idea of Incrementality

- If we have already computed the function on some document, and this document is modified, then we update the function value based on the old value, and introduced changes rather than re-computing it from scratch.

NTNU
Innovation and Creativity

Basic idea of Incrementality

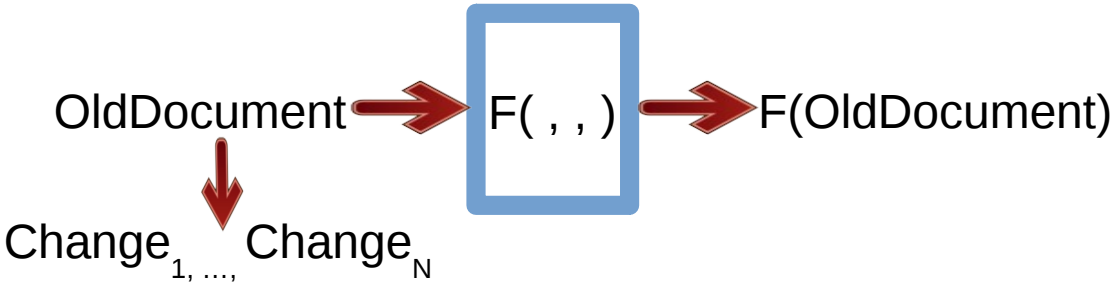
- If we have already computed the function on some document, and this document is modified, then we update the function value based on the old value, and introduced changes rather than re-computing it from scratch.



NTNU
Innovation and Creativity

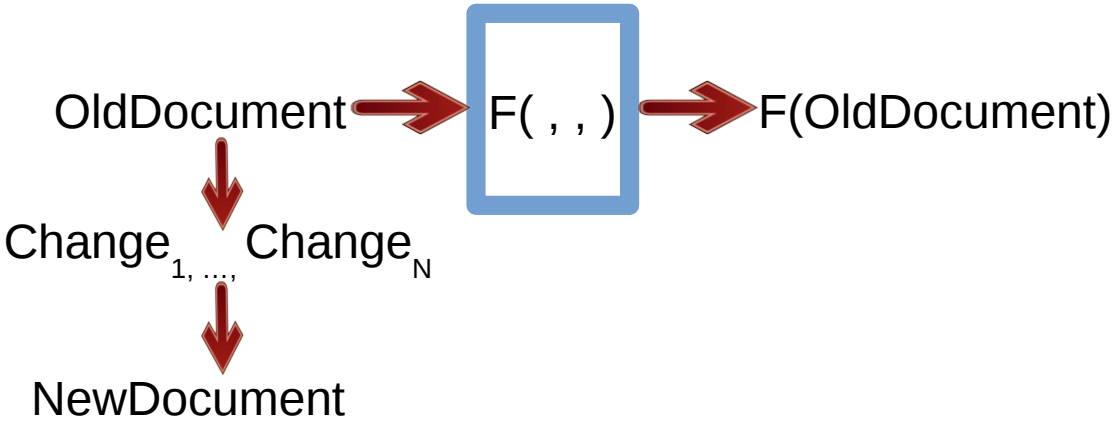
Basic idea of Incrementality

- If we have already computed the function on some document, and this document is modified, then we update the function value based on the old value, and introduced changes rather than re-computing it from scratch.



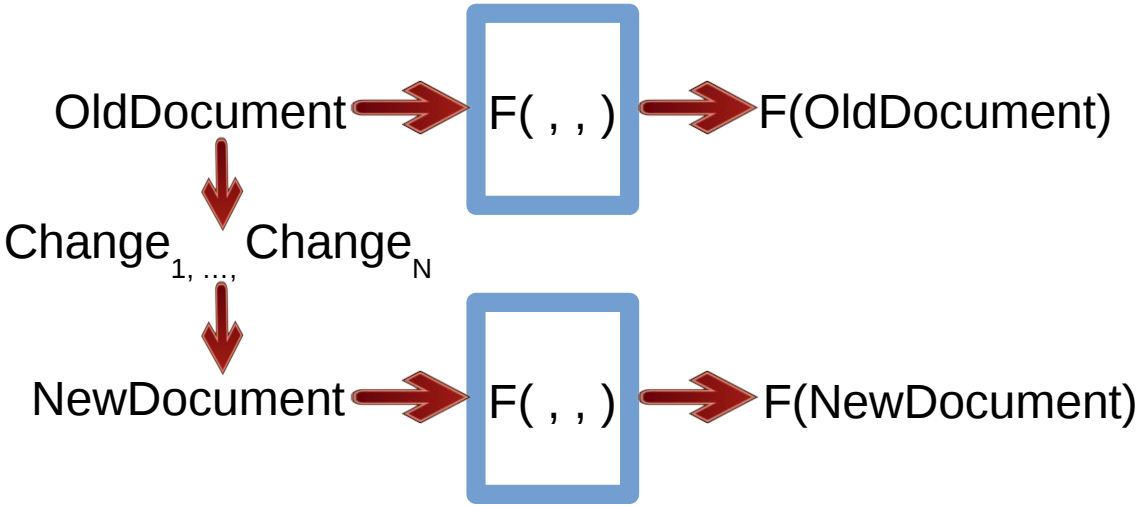
Basic idea of Incrementality

- If we have already computed the function on some document, and this document is modified, then we update the function value based on the old value, and introduced changes rather than re-computing it from scratch.



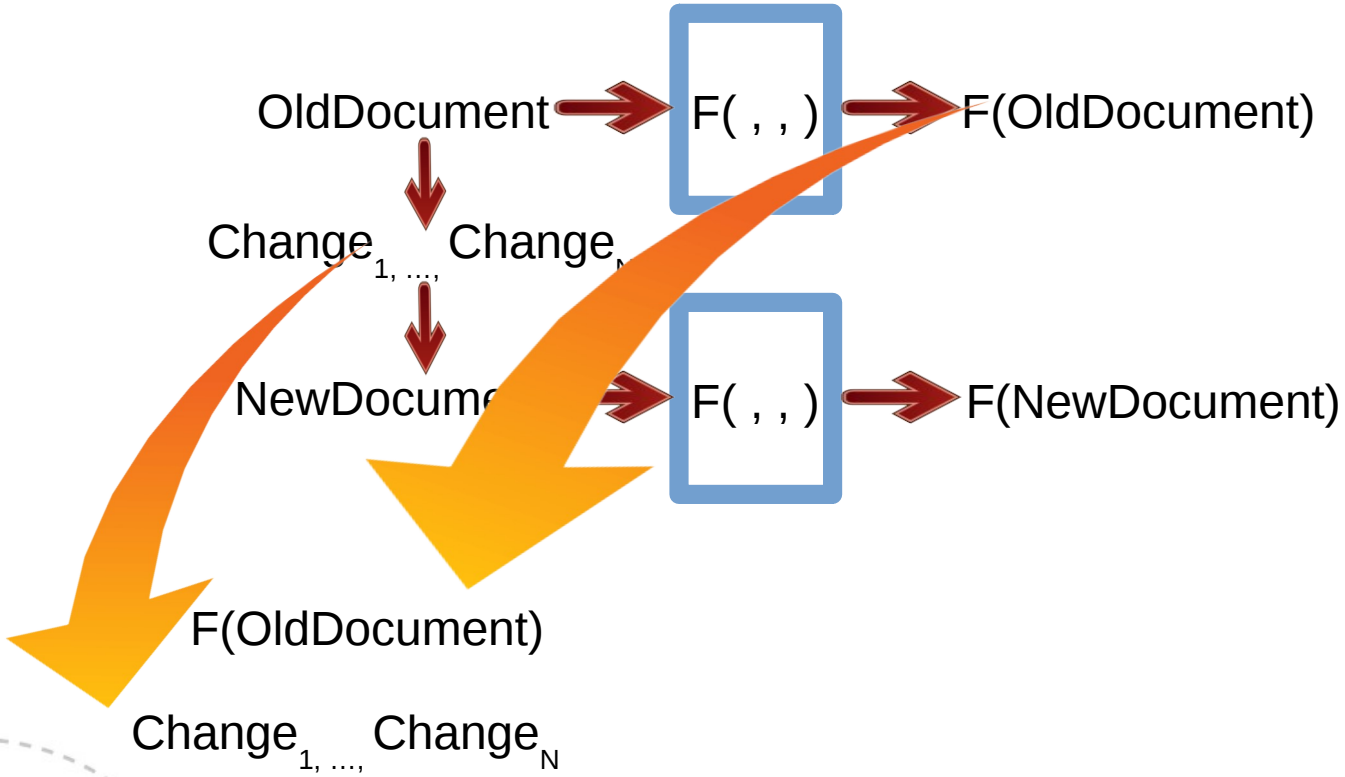
Basic idea of Incrementality

- If we have already computed the function on some document, and this document is modified, then we update the function value based on the old value, and introduced changes rather than re-computing it from scratch.



Basic idea of Incrementality

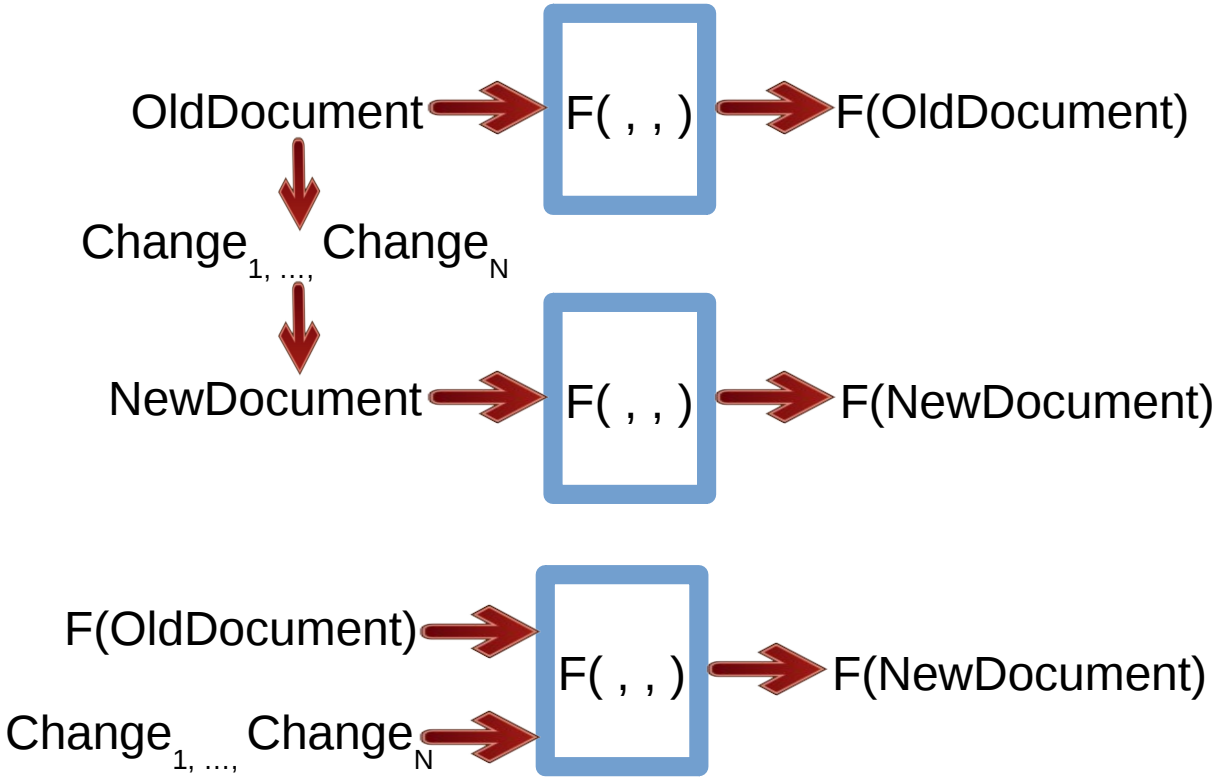
- If we have already computed the function on some document, and this document is modified, then we update the function value based on the old value, and introduced changes rather than re-computing it from scratch.



NTNU
Innovation and Creativity

Basic idea of Incrementality

- If we have already computed the function on some document, and this document is modified, then we update the function value based on the old value, and introduced changes rather than re-computing it from scratch.



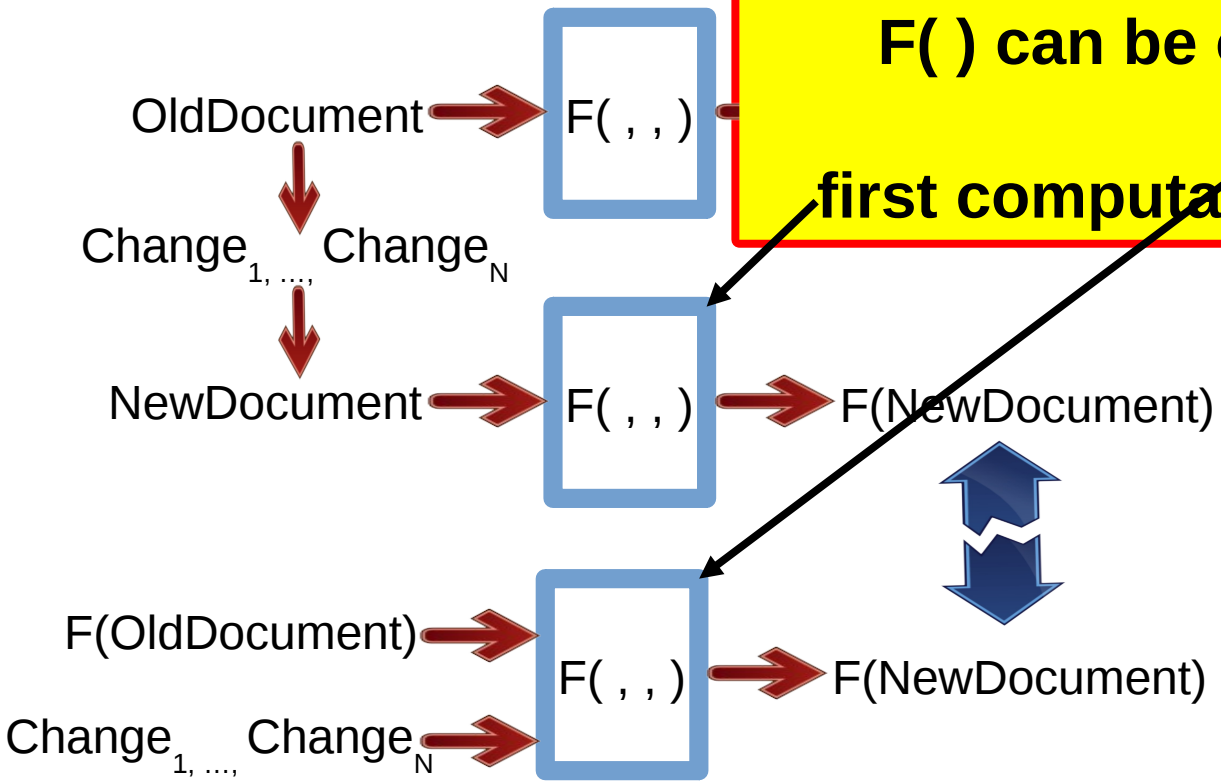
NTNU
Innovation and Creativity

Basic idea of Incrementality

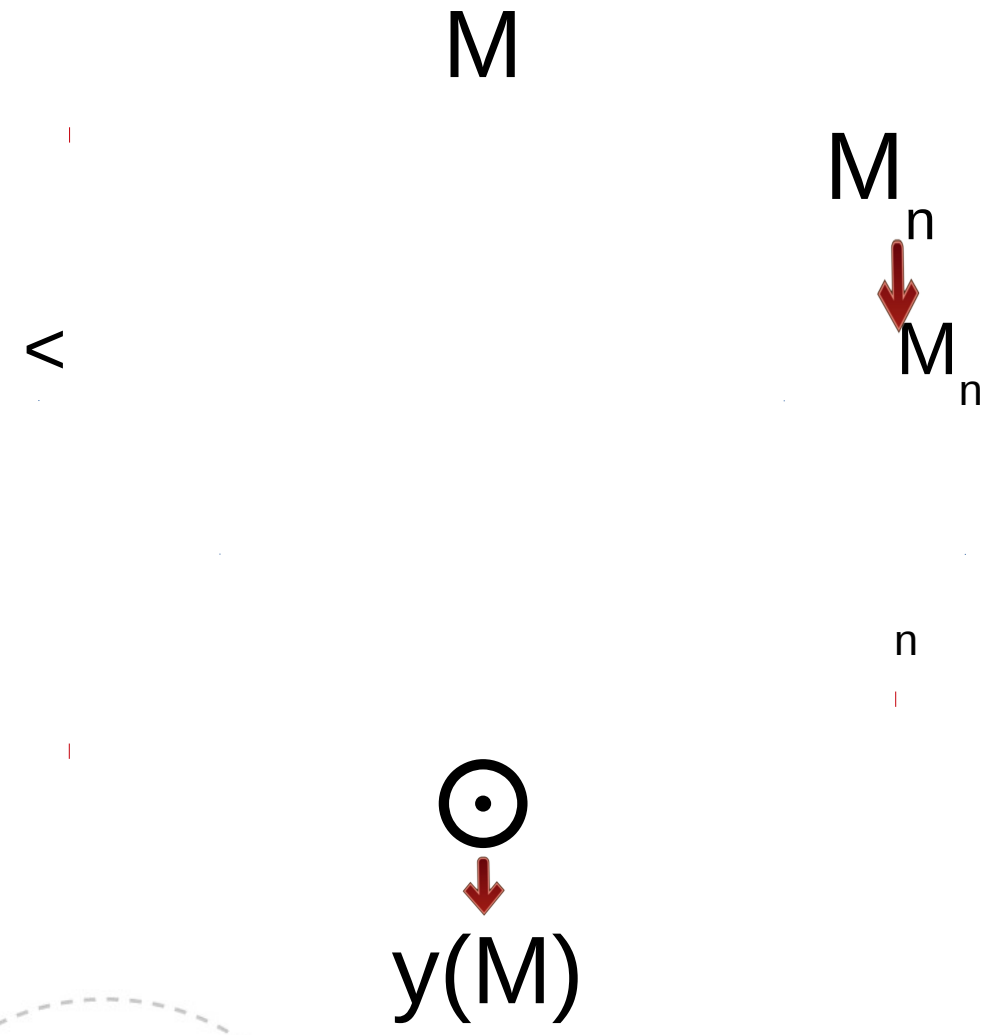
- If we have already computed the function on some document, and this document is modified, then we update the function value based on th
changes rather than re-compu

The main motivation is that the second computation of F( ) can be orders of

first computation of F( ).



Bellare-Miciancio concrete proposals for construction of Colision-free Hashing



- Group G with combining operation ⊙
- Randomizer or compression function h mapping fixed size strings to element of G
- Original message

  $M = M_1 M_2 \ldots M_n$

  that we transform into the message

  $M' = \langle 1 \rangle M_1 \langle 2 \rangle M_2 \ldots \langle n \rangle M_n$

  (prepend the index of each block to the block)
- Apply h to each block $M_i' = \langle i \rangle M_i$ to obtain $y_i = h(M_i')$
- Combine y_i in G by the operation ⊙:

  $y(M) = y_1 \odot y_2 \odot \ldots y_n$

NTNU
Innovation and Creativity

Operation ⊙, the size of the hash and properties of h

- Bellare and Micciancio proposed three variants for the operation ⊙:
  - MuHASH: Multiplication in a group
    - special cases:
      - multiplication in groups of prime order
      - integer multiplication modulo p
  - AdHASH: Addition modulo M
  - LtHASH: Vector addition
- They concluded that in order the incremental hash function y( ) to be collision free the following conditions must be satisfied:
  - In the group (G, ⊙) the balance problem should be hard,
  - The compression function h( ) should be collision free
  - The size k of the hash output should be around 1024 bits

NTNU
Innovation and Creativity

Operation ⊙, the size of the hash and properties of h

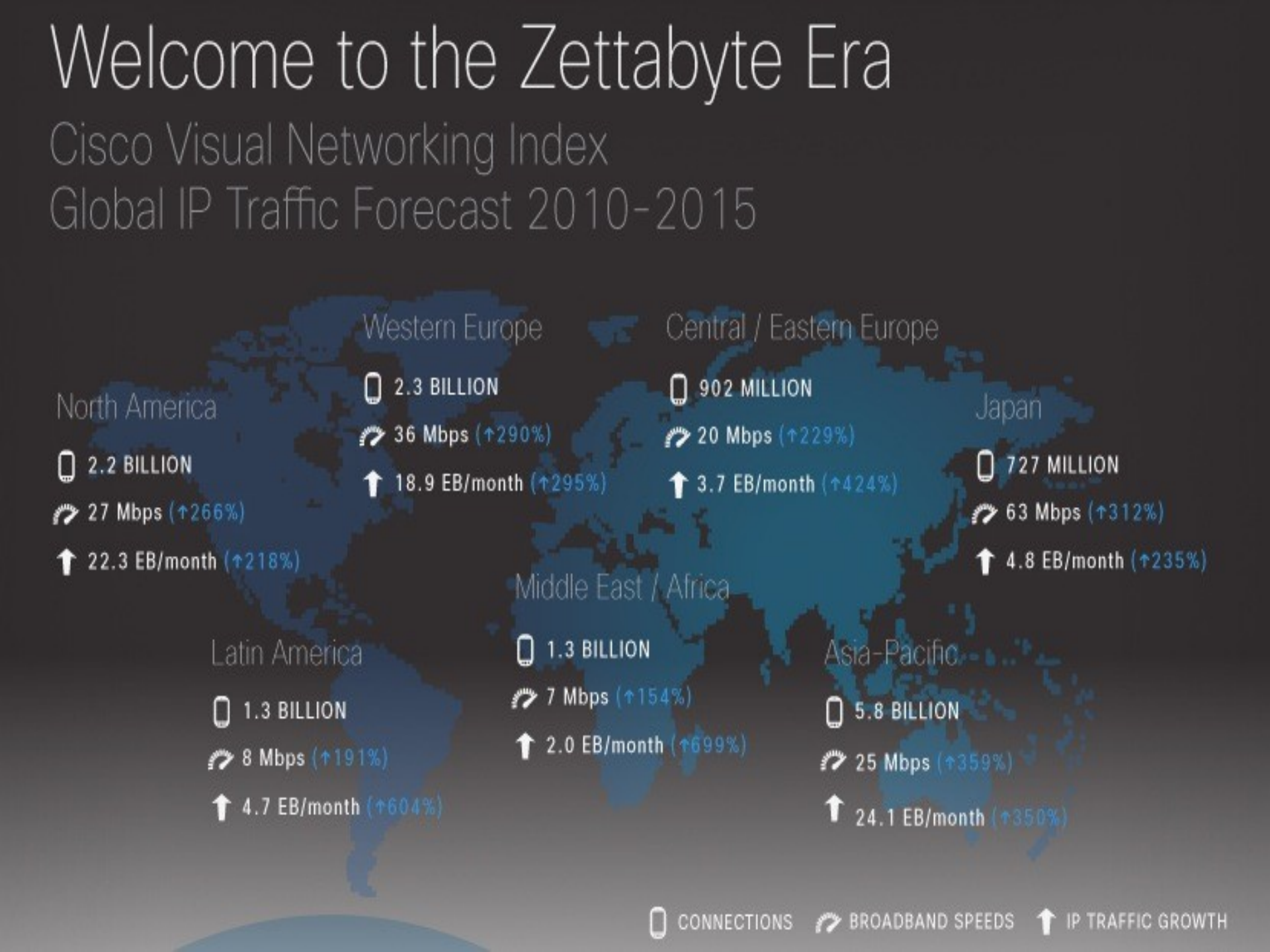
- However Wagner in *"A generalized birthday problem"* CRYPTO 2002, showed that the size k of the hash should be much bigger (for standard security levels, even up to tens of thousands of bits).
- Basically those findings killed the attractiveness of the concept of incremental hashing.
- Even more: None of the SHA-3 candidates explicitly explained can they offer some form of incremental hashing

NTNU
Innovation and Creativity

HOWEVER

Welcome to the Zettabyte Era

Cisco Visual Networking Index
Global IP Traffic Forecast 2010-2015



Welcome to the Zettabyte Era

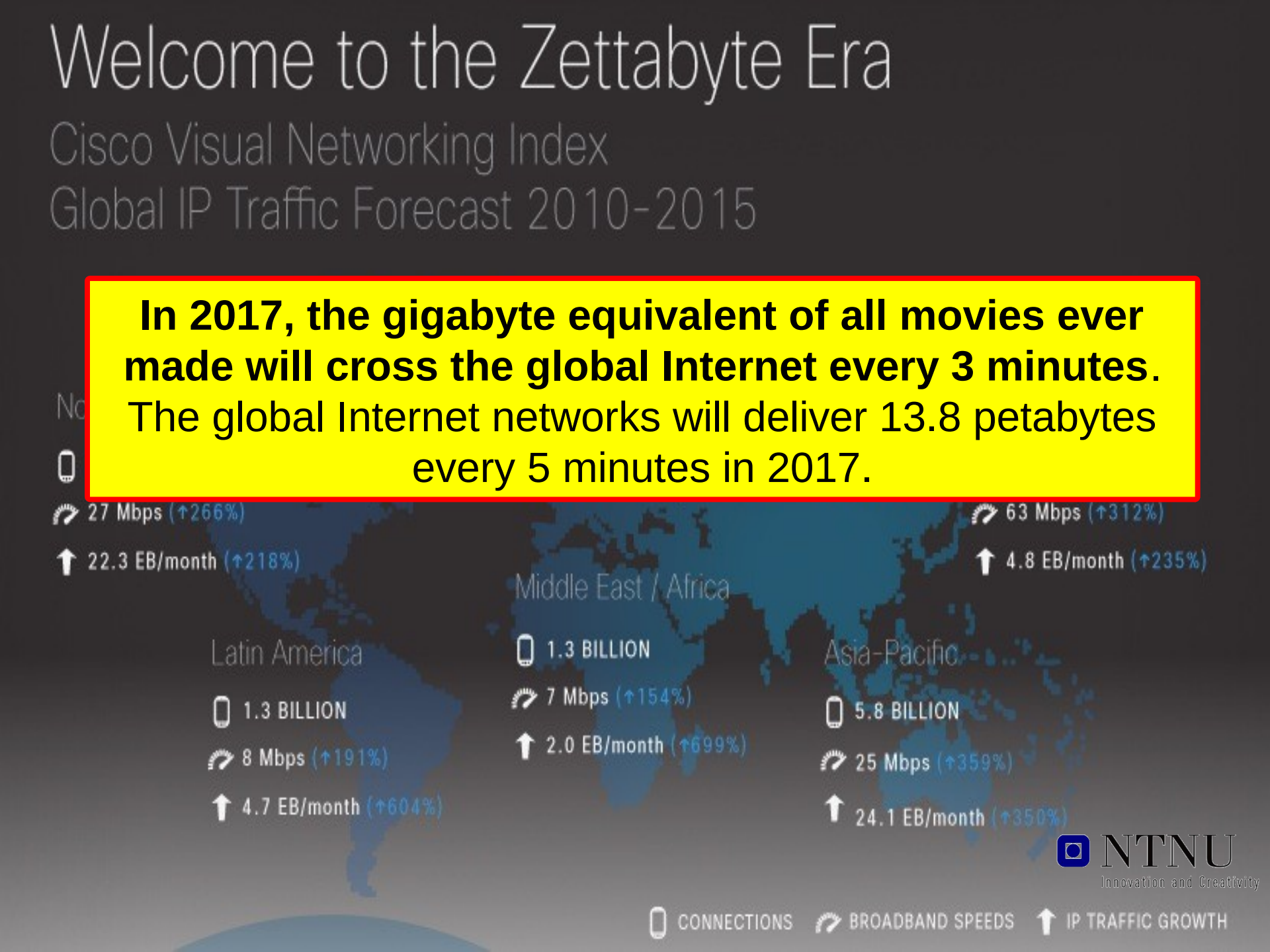
Cisco Visual Networking Index

Annual global IP traffic will pass the zettabyte threshold by the end of 2015, and will reach 1.4 zettabytes per year by 2017. In 2015, global IP traffic will reach 1.0 zettabytes per year or 83.8 exabytes per month, and by 2017, global IP traffic will reach 1.4 zettabytes per year or 120.6 exabytes per month.

Welcome to the Zettabyte Era

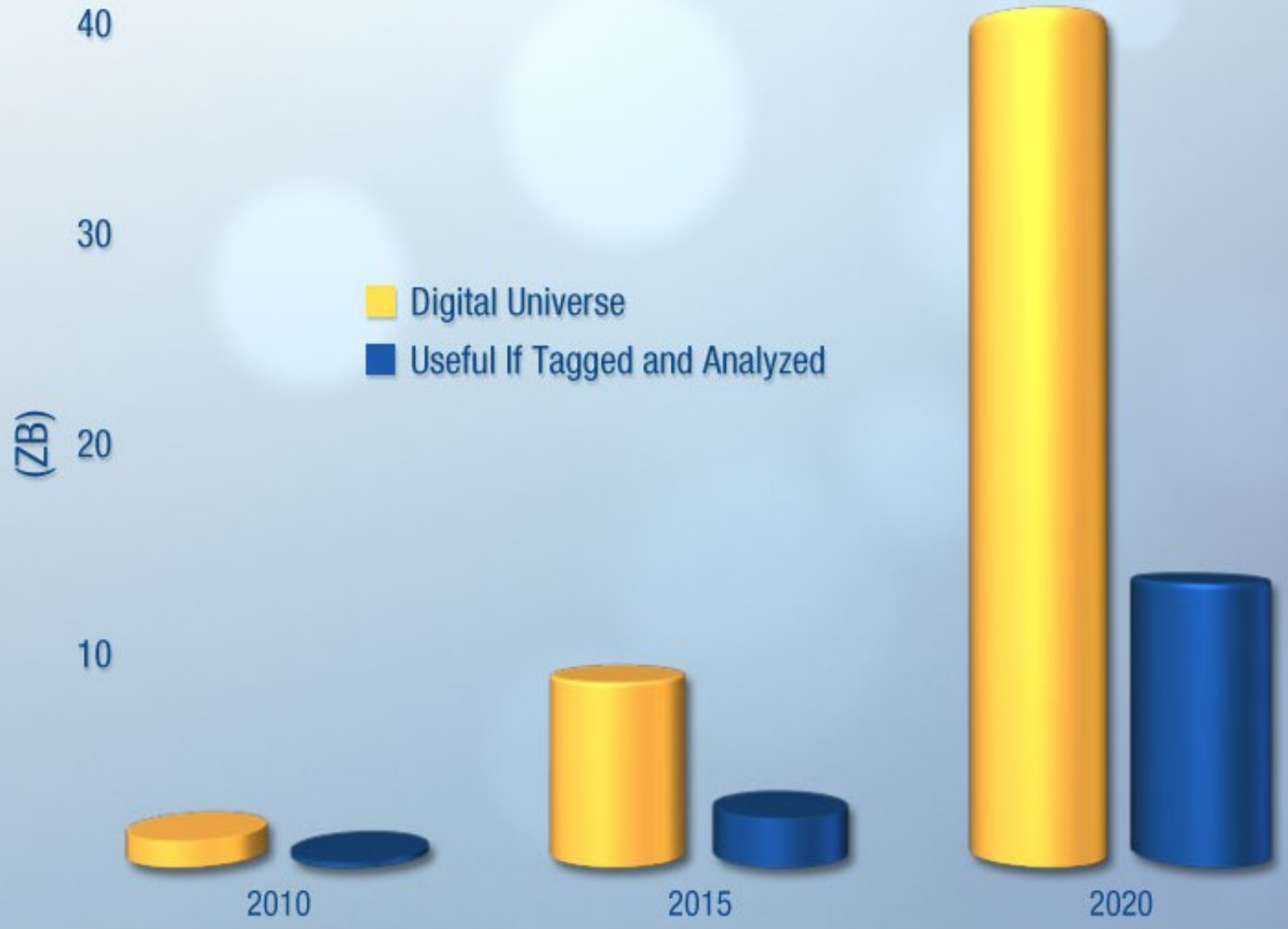
Cisco Visual Networking Index
Global IP Traffic Forecast 2010-2015

In 2017, the gigabyte equivalent of all movies ever made will cross the global Internet every 3 minutes. The global Internet networks will deliver 13.8 petabytes every 5 minutes in 2017.



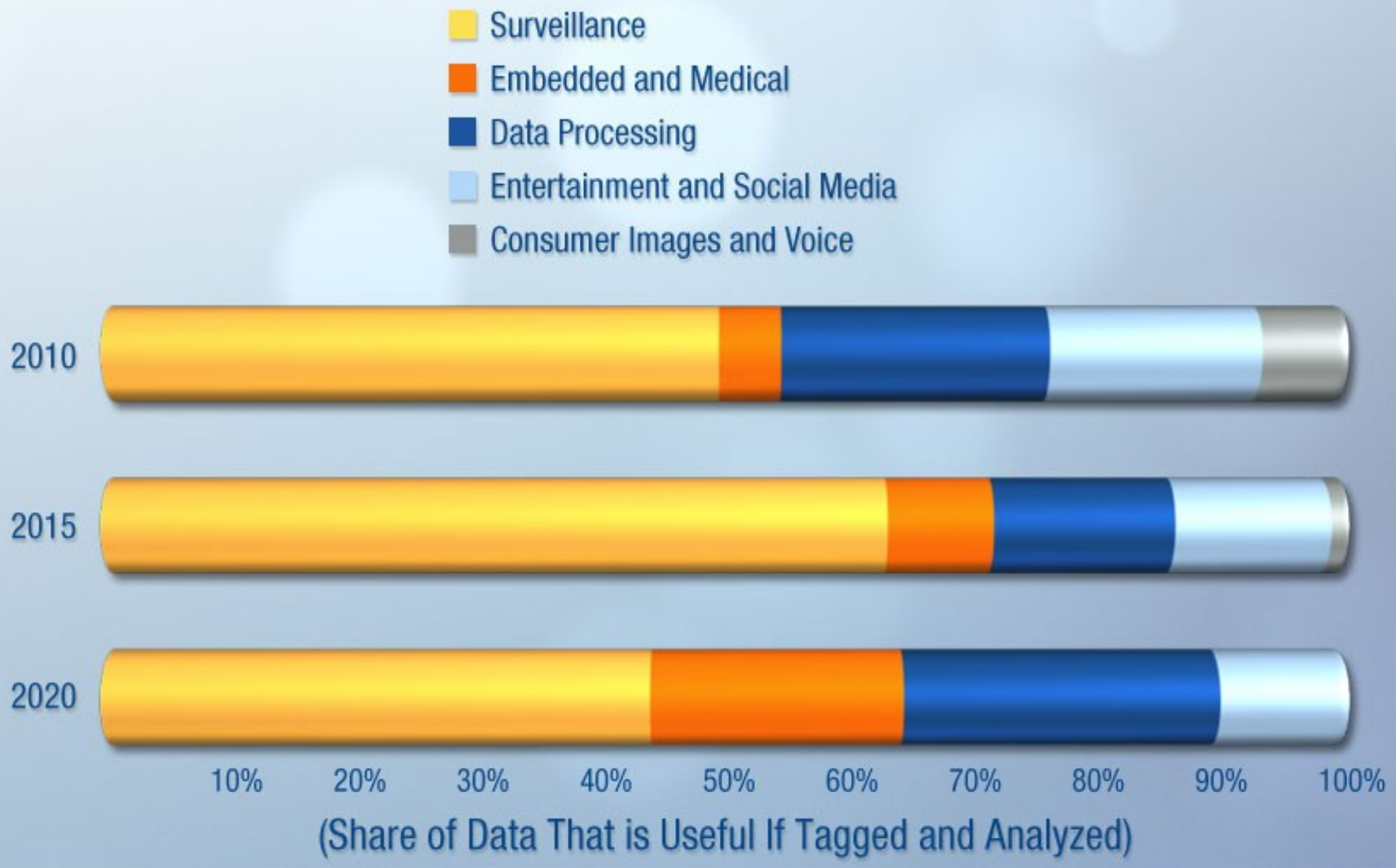
NTNU
Innovation and Creativity

Opportunity for Big Data



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

Candidates for Big Data



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

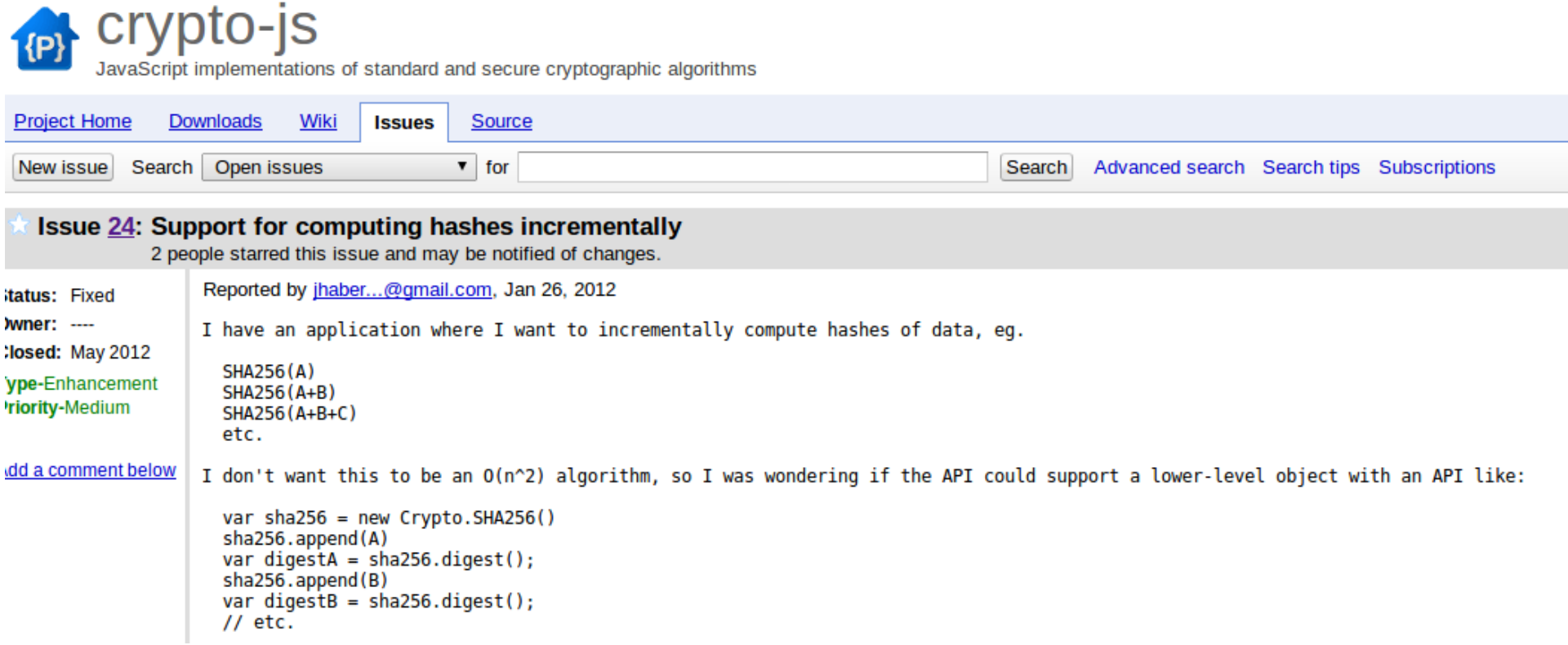
Are the practical needs for incremental hashing present today?

I would like to use MurmurHash3 to uniquely identify large pieces of data. This implementation:

http://code.google.com/p/smhasher/source/browse/trunk/MurmurHash3.h

Doesn't seem to provide a way to update the hash incrementally, though--it seems to compute one separate hash per block of data given. For example, if I were hashing 512MB of data from disk I might not want to load it all in memory at once, or if I were hashing an unknown amount of data from the network. Has anyone ever used MurmurHash3 in such a context before (hashing a large amount of

Are the practical needs for incremental hashing present today?



crypto-js

JavaScript implementations of standard and secure cryptographic algorithms

Project Home | Downloads | Wiki | **Issues** | Source

New issue | Search | Open issues | for | Search | Advanced search | Search tips | Subscriptions

Issue 24: Support for computing hashes incrementally

2 people starred this issue and may be notified of changes.

tatus: Fixed
wner: ----
losed: May 2012
ype-Enhancement
riority-Medium

dd a comment below

Reported by jhaber...@gmail.com, Jan 26, 2012

I have an application where I want to incrementally compute hashes of data, eg.

```
SHA256(A)
SHA256(A+B)
SHA256(A+B+C)
etc.
```

I don't want this to be an O(n^2) algorithm, so I was wondering if the API could support a lower-level object with an API like:

```
var sha256 = new Crypto.SHA256()
sha256.append(A)
var digestA = sha256.digest();
sha256.append(B)
var digestB = sha256.digest();
// etc.
```

NTNU
Innovation and Creativity

Are the practical needs for incremental hashing present today?

Duplicity (software)

From Wikipedia, the free encyclopedia

Duplicity is a software suite that provides encrypted, digitally signed, versioned, remote backup of files requiring little of the remote server.[1] Released under the terms of the GNU General Public License (GPL), Duplicity is free software.

Duplicity devises a scheme where the first archive is a complete (full) backup, and subsequent (incremental) backups only add differences from the latest full or incremental backup.[2] Chains consisting of a full backup and a series of incremental backups can be recovered to the point in time that any of the incremental steps were taken. If any of the incremental backups are missing then the incremental backups following it cannot be reconstructed. It does this using GnuPG, librsync, tar, and rdiff.[1] To transmit data to the backup repository it can use SSH/SCP/SFTP, local file access, rsync, FTP, Amazon S3,[3] Google Cloud Storage,[4] Rackspace Cloud Files,[5] and others. Refer to its man page for the constantly growing list of back-ends.

Are the practical needs for incremental hashing present today?

Duplicity - Bandwidth Efficient Encrypted Backup

Overview Code **Bugs** Blueprints Translations Answers

Very slow incremental backup of modified large files

Duplicity » Bugs » **Bug #1301892**

Reported by Andrei on 2014-04-03

This bug affects 1 person

Affects	Status	Importance	Assigned to	Milestone
Duplicity	New	Undecided	Unassigned	

Also affects project | Also affects distribution/package | Nominate for series

Bug Description

I'm using duplicity to backup to a LOCAL drive my virtual machine files.

The initial full backup goes swimmingly. Subsequent incremental backups also work very well as long as NONE of those large files (> 30 GB) have changes made to them. IF however those large >30 GB files have changes to them, it may take duplicity up to 10 hours to process the changes made to a single one of these large files.

Are the practical needs for incremental hashing present today?

Duplicity - Bandwidth Efficient Encrypted Backup

Overview Code **Bugs** Blueprints Translations Answers

Very slow incremental backup of modified large files

Duplicity » Bugs » **Bug #1301892**

Reported by Andrei

This bug affects 1 pe

Affects	Milestone
Duplicity	

In the latest version of Duplicity this issue is resolved (by using hash trees and storing intermediate hash values and signatures).

Also affects project | Also affects distribution/package | Nominate for series

Bug Description

I'm using duplicity to backup to a LOCAL drive my virtual machine files.

The initial full backup goes swimmingly. Subsequent incremental backups also work very well as long as NONE of those large files (> 30 GB) have changes made to them. IF however those large >30 GB files have changes to them, it may take duplicity up to 10 hours to process the changes made to a single one of these large files.

ity

Are the practical needs for incremental hashing present today?

Signature files are not required to restore a backup set, but without an up-to-date signature, duplicity cannot append an incremental backup to an existing archive.

To save bandwidth, duplicity generates full signature sets and incremental signature sets. A full signature set is generated for each full backup, and an incremental one for each incremental backup. These start with **duplicity-full-signatures** and **duplicity-new-signatures** respectively. These signatures will be stored both locally and remotely. The remote signatures will be encrypted if encryption is enabled. The local signatures will not be encrypted and stored in the archive dir (see **--archive-dir**).

NTNU
Innovation and Creativity

Are the practical needs for incremental hashing present today?

Signature files are not required to restore a backup set, but without an up-to-date signature, duplicity cannot append an incremental backup to an existing archive.

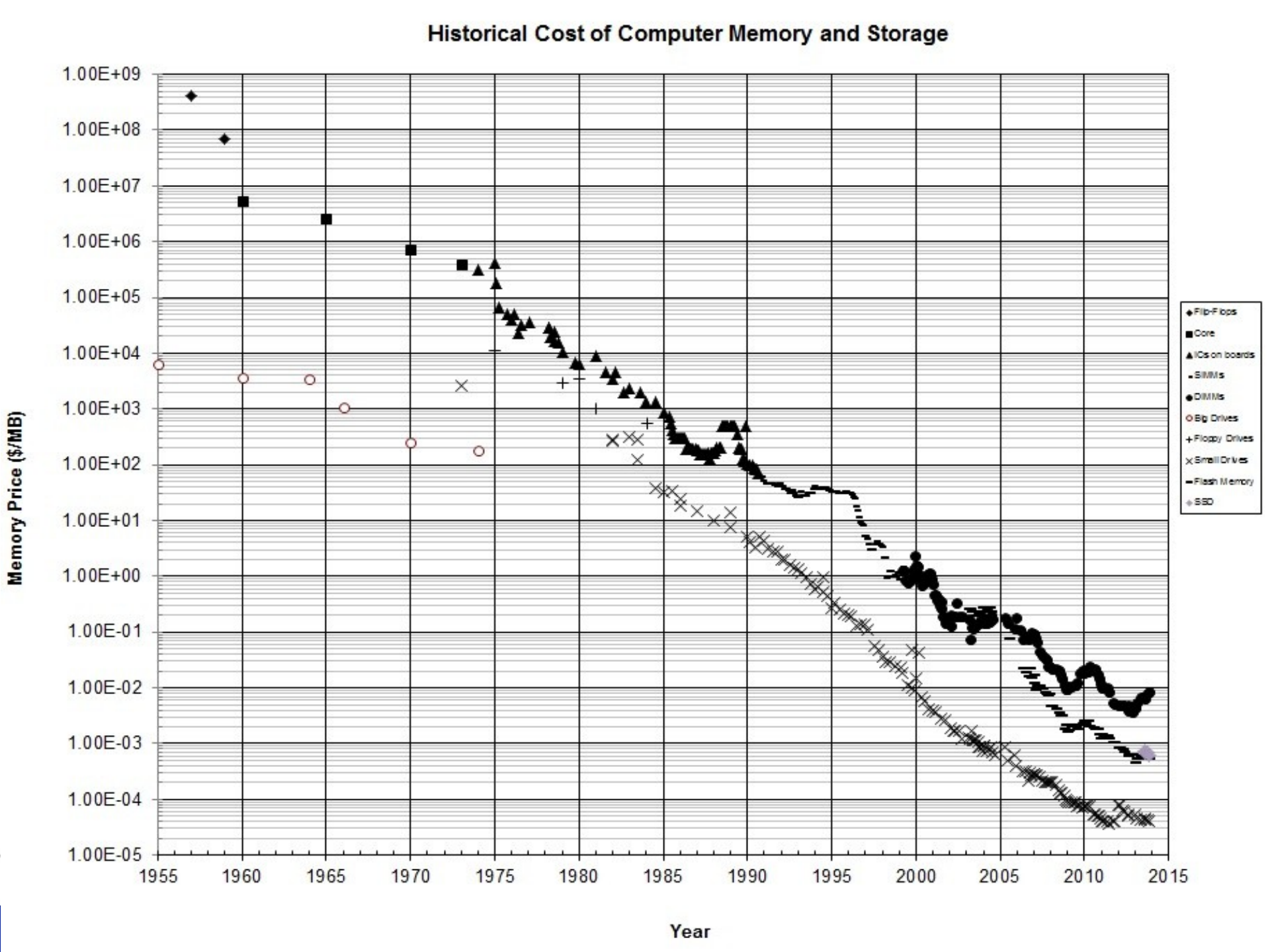
To save ba . A full signature set is gene p. These start with **dupl** tures will be stored both is enabled. The local signa

Having an incremental hash function would increase the speed of production of signed backups by order of magnitude, while decreasing the extra storage for intermediate signature sets.

NTNU
Innovation and Creativity

Observations

Observation 1. In the Zettabyte era the trend for reducing the cost of data storage will diminish the importance of the fact that incremental hashing needs longer digests than traditional hash functions.


Historical Cost of Computer Memory and Storage
Memory Price ($/MB)
1.00E+09
1.00E+08
1.00E+07
1.00E+06
1.00E+05
1.00E+04
1.00E+03
1.00E+02
1.00E+01
1.00E+00
1.00E-01
1.00E-02
1.00E-03
1.00E-04
1.00E-05
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
Year
Flip-Flops
Core
ICs on boards
SIMMs
DIMMs
Big Drives
Floppy Drives
Small Drives
Flash Memory
SSD

Observations

Observation 2. In the Zettabyte era there will be an increased need for an efficient and secure cryptographic primitive that will perform incremental collision-free hashing.

NTNU
Innovation and Creativity

Observations

Observation 3. *For the compression functions* $h : \{0,1\}^b \rightarrow \{0,1\}^k$ *where* k *is multiple of 64 bits i.e.* $k = 64 \cdot L$*, on modern 64-bit CPUs, instead of modular operations with* k *bit prime numbers in the group* $(G, \odot)$*, much more efficient operations would be word wise operations of addition in the group* $((\mathbb{Z}_{64})^L, \boxplus_{64})$.

Formalizing previous notations

Definition 1.

1. *Let* $h : \{0,1\}^b \to \{0,1\}^k$ *be a compression function that maps* b *bits into* k *bits.*
2. *Let the message* M *be represented as a concatenation of* n *blocks, where* $n < N$ *for some predefined number* N *which is larger than the number of blocks in any message we plan to hash, i.e.,* $M = M_1||M_2||\ldots||M_n$.
3. *The size of each block* M_i *is determined by the following relation:* $|M_i| = b - \lg(N)$.
4. *For each block* M_i, $i = 1, \ldots, n$, *prepend a* $\lg(N)$*-bit binary encoding* $\langle i \rangle$ *of the block index* i *to the block content* M_i *to get an augmented block* $\overline{M_i} = \langle i \rangle || M_i$.
5. *For each* $i = 1, \ldots, n$, *apply* h *to* $\overline{M_i}$ *to get a hash value* $y_i = h(\overline{M_i})$.
6. *Let* $(G, \odot)$ *be a commutative group with operation* $\odot$ *where* $G \subseteq \{0,1\}^k$.
7. *Combine* $y_1, \ldots, y_n$ *via a combining group operation* $\odot$ *to get the final hash value* $y = y_1 \odot y_2 \odot \ldots \odot y_n$.

Denote the incremental hash function as:

$$y(M) = \mathrm{HASH}^h_{\langle G \rangle}(M_1||M_2||\ldots||M_n) = \bigodot_{i=1}^{n} h(\langle i \rangle || M_i) \tag{1}$$

NTNU
Innovation and Creativity

Formalizing previous notations

Proposition 1. *Let* $\mathrm{HASH}^{h}_{\langle G \rangle}$ *be an incremental hash function defined by Definition 1. For any* $Y \in \{0,1\}^k$ *the complexity of finding a preimage message* $M = M_1||M_2||\ldots||M_K$ *of length* $K \leqslant N$ *blocks such that* $Y = \mathrm{HASH}^{h}_{\langle G \rangle}(M)$ *is:*

$$\min_{K \leqslant N} O(K \cdot 2^{\frac{k}{1+\lg\lceil K \rceil}}) \tag{2}$$

If the length of the messages is not restricted, then the minimum in equation (2) is achieved for messages of $K = 2^{\sqrt{k}-1}$ *blocks.*

NTNU
Innovation and Creativity

Formalizing previous notations

Proposition 1. *Let* $\mathrm{HASH}^{h}_{\langle G \rangle}$ *be an incremental hash function defined by Definition 1. For any* $Y \in \{0,1\}^k$ *the complexity of finding a preimage message* $M = M_1||M_2||\ldots||M_K$ *of length* $K \leqslant N$ *blocks such that* $Y = \mathrm{HASH}^{h}_{\langle G \rangle}(M)$ *is:*

$$\min_{K \leqslant N} O(K \cdot 2^{\frac{k}{1+\lg\lceil K \rceil}}) \tag{2}$$

If the length of the messages is not restricted, then the minimum in equation (2) is achieved for messages of $K = 2^{\sqrt{k}-1}$ *blocks.*

Proof. Just adopt the notation from Wagner's Crypto 2002 paper [Sec. 2, Summary] to match the notation of variables in Definition 1.

NTNU
Innovation and Creativity

Extendable-Output Functions SHAKE128 and SHAKE256

- DRAFT FIPS 202 *"SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions"*

NTNU
Innovation and Creativity

Extendable-Output Functions SHAKE128 and SHAKE256

$$\text{SHAKE128}(M, d) = \text{RawSHAKE128}(M||11, d),$$

where

$$\text{RawSHAKE128}(M, d) = \text{KECCAK}[256](M||11, d),$$

and

$$\text{SHAKE256}(M, d) = \text{RawSHAKE256}(M||11, d),$$

where

$$\text{RawSHAKE256}(M, d) = \text{KECCAK}[512](M||11, d).$$

NTNU
Innovation and Creativity

iSHAKE128

Definition 2.

1. *Let* $h : \{0,1\}^{1344} \to \{0,1\}^{2688}$ *be defined as the function* $h(m) = \text{SHAKE128}(m, 2688)$ *where* $|m| = 1344$.
2. *Let the message* $M = M_1||M_2||\ldots||M_n$ *be represented as a concatenation of* n *blocks, where* $n < N$, *and* $N = 2^{25}$ *is the largest number of blocks in any message we plan to hash.*
3. *The size of each block* M_i *in bits is determined by the following relation:* $|M_i| = 1344 - 64 = 1280$.
4. *For each block* M_i, $i = 1, \ldots, n$, *prepend a 64-bit binary encoding* $\langle i \rangle$ *of the block index* i *to the block content* M_i *to get an augmented block* $\overline{M_i} = \langle i \rangle || M_i$.
5. *For each* $i = 1, \ldots, n$, *apply* h *to* $\overline{M_i}$ *to get a hash value* $y_i = h(\overline{M_i}) = \text{SHAKE128}(\overline{M_i}, 2688)$.
6. *Let* $((\mathbb{Z}_{64})^{42}, \boxplus_{64})$ *be a commutative group with the operation* $\boxplus_{64}$ *that represents a 64-bit word wise addition of 42 words.*
7. *Combine* $y_1, \ldots, y_n$ *via a combining group operation* $\boxplus_{64}$ *to get the final hash value*

$$y = y_1 \boxplus_{64} y_2 \boxplus_{64} \cdots \boxplus_{64} y_n.$$

Denote the incremental hash function iSHAKE128 *as:*

$$i\text{SHAKE128}(M) = \boxplus_{64,\, i=1}^{N} SHAKE128(\overline{M_i}, 2688). \tag{3}$$

NTNU
Innovation and Creativity

iSHAKE128

Corollary 1: Let $b = 1280$, $k = 2688$. and let the maximal allowed number of blocks be $N = 2^{25}$. Then

$$\min_{K \leqslant N} O(K \cdot 2^{\frac{k}{1+\lg\lceil K \rceil}}) = 2^{128.385} \tag{4}$$

NTNU
Innovation and Creativity

*i*SHAKE128

- For a 2^{128} level of security, the maximal size of the files that can be hashed with iSHAKE128 is 5 GB.
- For small file sizes such as 160 KB the complexity of finding collisions with Wagner's generalized birthday attack is 2^{254}

and

- For files long 1.25 TB the complexity of finding collisions drops down to 2^{112} .

NTNU
Innovation and Creativity

*i*SHAKE256

Definition 3.

1. *Let* $h : \{0,1\}^{1088} \rightarrow \{0,1\}^{6528}$ *be defined as the function* $h(m) = \text{SHAKE256}(m, 6528)$ *where* $|m| = 1088$.
2. *Let the message* $M = M_1||M_2||\ldots||M_n$ *be represented as a concatenation of* n *blocks, where* $n < N$, *and* $N = 2^{28}$ *is the largest number of blocks in any message we plan to hash.*
3. *The size of each block* M_i *in bits is determined by the following relation:* $|M_i| = 1088 - 64 = 1024$.
4. *For each block* M_i, $i = 1, \ldots, n$, *prepend a 64-bit binary encoding* $\langle i \rangle$ *of the block index* i *to the block content* M_i *to get an augmented block* $\overline{M_i} = \langle i \rangle||M_i$.
5. *For each* $i = 1, \ldots, n$, *apply* h *to* $\overline{M_i}$ *to get a hash value* $y_i = h(\overline{M_i}) = \text{SHAKE256}(\overline{M_i}, 6528)$.
6. *Let* $((\mathbb{Z}_{64})^{102}, \boxplus_{64})$ *be a commutative group with the operation* $\boxplus_{64}$ *that represents a 64-bit word wise addition of 102 words.*
7. *Combine* $y_1, \ldots, y_n$ *via a combining group operation* $\boxplus_{64}$ *to get the final hash value*

$$y = y_1 \boxplus_{64} y_2 \boxplus_{64} \cdots \boxplus_{64} y_n.$$

Denote the incremental hash function iSHAKE256 *as:*

$$i\text{SHAKE256}(M) = \boxplus_{64,\, i=1}^{N} SHAKE256(\overline{M_i}, 6528). \tag{5}$$

NTNU
Innovation and Creativity

iSHAKE256

Corollary 2: Let $b = 1024$, $k = 6528$. and let the maximal allowed number of blocks be $N = 2^{28}$. Then

$$\min_{K \leqslant N} O(K \cdot 2^{\frac{k}{1+\lg\lfloor K \rfloor}}) = 2^{253.103} \tag{6}$$

NTNU
Innovation and Creativity

*i*SHAKE256

- For a 2^{253} level of security, the maximal size of the files that can be hashed with iSHAKE256 is 32 GB.
- For small file sizes such of 1 MB the complexity of finding collisions with Wagner's generalized birthday attack is 2^{479}

and

- For files long 8 TB the complexity of finding collisions drops down to 2^{212}.

NTNU
Innovation and Creativity

Properties of iSHAKE parallel operations

- Trivial parallelization without a need for special scheduling
  - (except that every message bloc M_i has to be prepended with its index <i> in the message)
- The obtained hash is the same, regardless of the level of parallelism used in the computation
- No need to keep extra intermediate values in order to achieve incrementality
  - What is bigger: the amount of stored intermediate chain values in tree-hash modes or the size of hash values in iSHAKE (2688 / 6528 bits)?

NTNU
Innovation and Creativity

Thank you for your attention!

NTNU
Innovation and Creativity