marc.beunardeau@nomadic-labs.com

Cryptographic hash functions and digital signature

Marc Beunardeau

September 2019

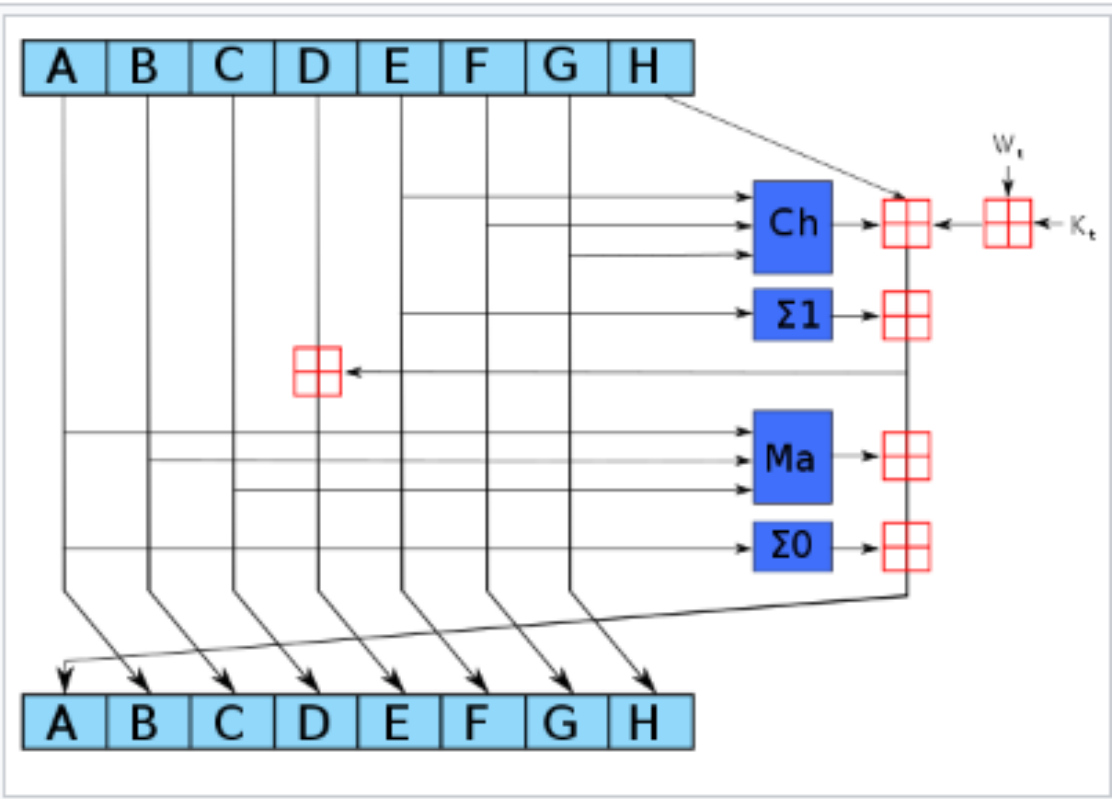
Cryptographic hash function, informal definition

A cryptographic hash function is an efficiently computable deterministic function from $\{0,1\}^* \rightarrow \{0,1\}^{256}$ such that there is no understandable link between the input and the output

Counter-example

$f : x \mapsto x + 1$ the link between x anf $f(x)$ is obvious
$g : 0 \mapsto 0001110011....$
$1 \mapsto 1010110111....$
$00 \mapsto 0110110101....$
$11 \mapsto 1101011010....$
...
g is not efficiently comutable
$h : x \mapsto Rand()$
h is not deterministic

Example 1/3 : SHA 2 Compression function



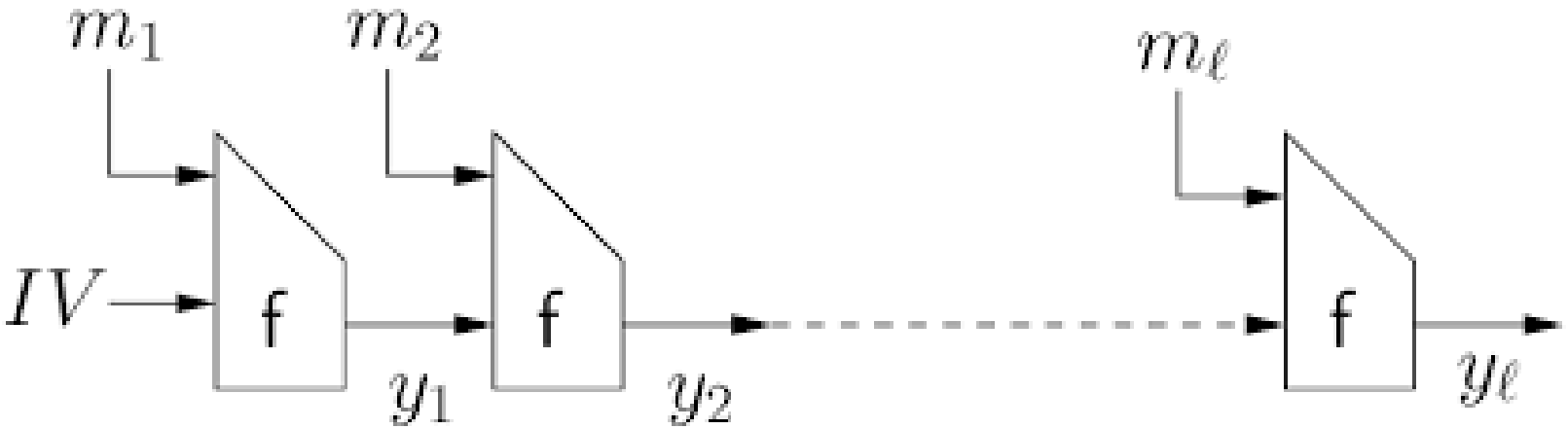
One iteration in a SHA-2 family compression function. The blue components perform the following operations:

$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

The bitwise rotation uses different constants for SHA-512. The given numbers are for SHA-256.

Example 2/3 Merkle-Damgard based hash function

$m = m_1 \cdots m_\ell$



Example 3/3 SHA 256

$SHA-256(0) = 5feceb66ffc86f38d952786c6d696c79$
$c2dbc239dd4e91b46729d73a27fb57e9$
$SHA-256(1) = 6b86b273ff34fce19d6b804eff5a3f57$
$47ada4eaa22f1d49c01e52ddb7875b4b$
$SHA-256(2) = d4735e3a265e16eee03f59718b9b5d03$
$019c07d8b6c51f90da3a666eec13ab35$
$SHA-256(3) = 4e07408562bedb8b60ce05c1decfe3ad$
$16b72230967de01f640b7e4729b49fce$

Pre-image resistance

For a hash function h it is impossible in a reasonable time ($\approx 2^{128}$) given $y \in \{0,1\}^{256}$ to find x such that $h(x) = y$

Collision resistance

For a hash function h it is impossible in a reasonable time ($\approx 2^{128}$) to find x, y such that $h(x) = h(y)$

Birthday paradox

for any h it is feasible to find a collision in 2^128 time and space
Indeed computing $h(x_1), \cdots, h(x_{2^{128}})$ we have a possible collision between $h(x_1)$ and $h(x_2), \cdots, h(x_{2^{128}})$ and $h(x_2)$ and $h(x_3), \cdots, h(x_{2^{128}})$ etc...

Second pre-image resistance

For a hash function h it is impossible in a reasonable time ($\approx 2^{128}$) given x to find y such that $h(x) = h(y)$

Relation between properties

second pre-image resistance implies pre-image resistance
Collision resistance is separated

Concrete Security

				attaques par collision				attaques de (seconde) pré-image			
hachage	année	bits	cpb	sans danger ?	comp	mem	ref	sans danger ?	comp	mem	ref
MD2	1989	128	638	non	2^{64}	2^{0}	[†]	oui	2^{72}	2^{72}	[2]
Snefru -2 [3]	1990	128	?	non	2^{13}	2^{0}	[4]	non	2^{25}	2^{0}	[4]
MD4	1990	128	4.0	non	2^{2}	2^{0}	[6]	oui	2^{95}	2^{38}	[5]
RIPEMD	1990	128	?	non	2^{18}	2^{0}	[36]	oui			
MD5	1992	128	5.1	non	2^{24}	2^{0}	[9]	oui	2^{123}	2^{48}	[8]
HAVAL-256-3 [25]	1992	256	?	non	2^{29}	2^{0}	[11]	oui	2^{225}	2^{68}	[10]
SHA-0	1993	160	?	non	2^{34}	2^{0}	[13]	oui	2^{189}	2^{8}	
GOST	1994	256	?	peut-être	2^{105}	2^{0}	[14]	oui	2^{192}	2^{70}	[14]
SHA-1	1995	160	18	non	2^{63}	2^{0}	[53]	oui			
RIPEMD-160 [30]	1996	160	17	peut-être	2^{80}	2^{0}	[§]	oui			
Tiger [31]	1996	192	6.2	oui				oui	2^{189}	2^{8}	[16]
Panama [33]	1998	512	2.5	non	2^{6}	2^{0}	[17]	oui			
Whirlpool [32]	2000	512	50	oui				oui			

Concrete Security

SHA-256 [37] [52]	2001	256	19	oui				oui			
RadioGatún [38]	2006	256	?	oui				oui			
Skein [39]	2008	256	8.7	oui				oui			
Blake [40]	2008	256	17	oui				oui			
Grøstl [41]	2008	256	24	oui				oui			
Keccak (SHA-3) [42]	2008	256	16	oui				oui			
JH [43]	2008	256	20	oui				oui			
BLAKE2 [44]	2012	256	5.7	oui				oui			

Blockchain example

The n-th block contains the hash of the $n-1$-th block Given a block chain of length n :

- It is easy to check that the chain is correct ($O(n)$)
- Changing the i-th block implies to change all the $j \geq i$ hashes (Second pre-image resistance)

Commitment scheme

A commitment scheme is composed of two algorithms $Commit(x, r) = c$ and $Reveal(c, x, r) = boolean$ satisfying two properties :

- Hiding : From c no reasonable algorithm can find x (nor any info about x)
- Binding : Given $c = Commit(x, \cdot)$ no reasonable algorithm can find x', r'such that $Reval(c, x', r') = T$

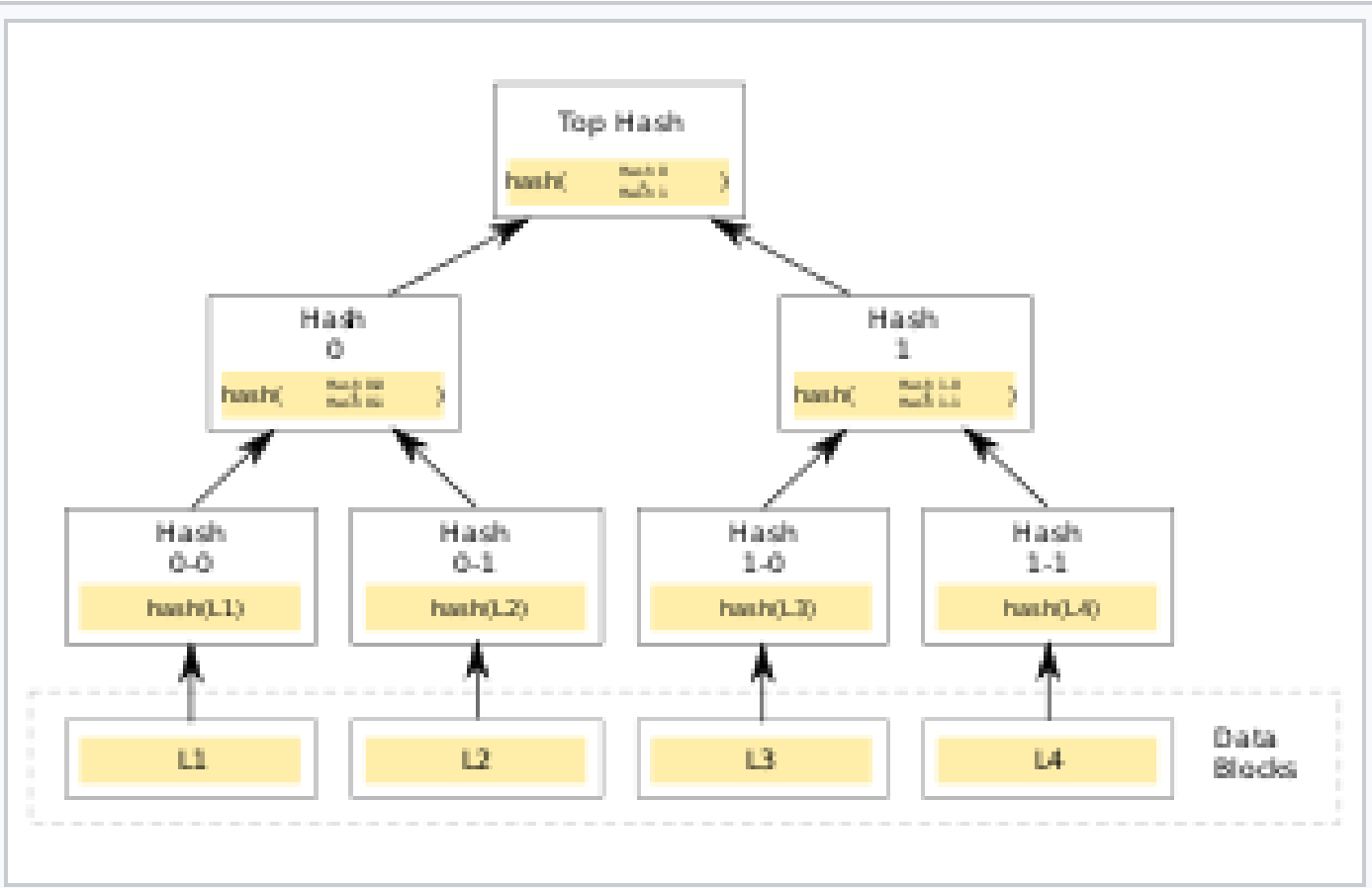
Commitment scheme from hash function

Define $Commit(x, r) = hash(x|r)$ with $r = Rand()$ and
$Reveal(c, x, r) = hash(x, r) == c$
This is hiding from pe-image resistance and binding from collision resistance
Not that we need r if x is in a small space.

Merkle Tree

Data Structure that allows to efficiently check that an element is in a set ($O(1)$ space and $\log(n)$ time)

Merkle Tree



An example of a binary hash tree. Hashes 0-0 and 0-1 are the hash values of data blocks L1 and L2, respectively, and hash 0 is the hash of the concatenation of hashes 0-0 and 0-1.

Digital Signature

Digital signature are used to prove the authenticity of a message (as real life signature). Indeed network can be open, and someone can claim for example his message came his neighbour address. It is a fundamental building block of public key cryptography.

Syntax

A digital signature schemes is composed of three algorithms :

- $KeyGen() = (pk, sk)$ randomised efficient
- $Sign(m, sk) = \sigma$ possibly randomised efficient
- $Verify(pk, m, \sigma) = boolean$ usually deterministic, efficient

The secret key sk is to be kept by the owner, the public key pk is to be given to whoever you want to be authenticated by.

Correctness

A digital signature scheme is said to be correct if :
for all $(pk, sk) = KeyGen()$, m and $\sigma = Sign(m, sk)$ then
$Verify(pk, m, \sigma) = T$

EUF-CMA security

A digital signature scheme is said to be Existentially UnForgeable agaisnt Chosen Message Adversary if given a key pair (pk, sk), no reasonable algotihm given pk and signatures on any messages of his choosing $m_1, \cdots, m_n$, can produce $m \neq m_i$, σ such that $Verify(pk, m, \sigma) = T$

Blockchain usage

An address is a public key (or something related). Sign the transaction with your private key. A transaction is correct only if the *Verify* algorithm says so.

Replay attacks

The preceding naive usage is exposed to a naive replay attack. A sends money to B, who records the message and play it later. to get the money again.
A simple solution is that A and B share a synchronised counter, which is incremented at each message and added at the start of the message.

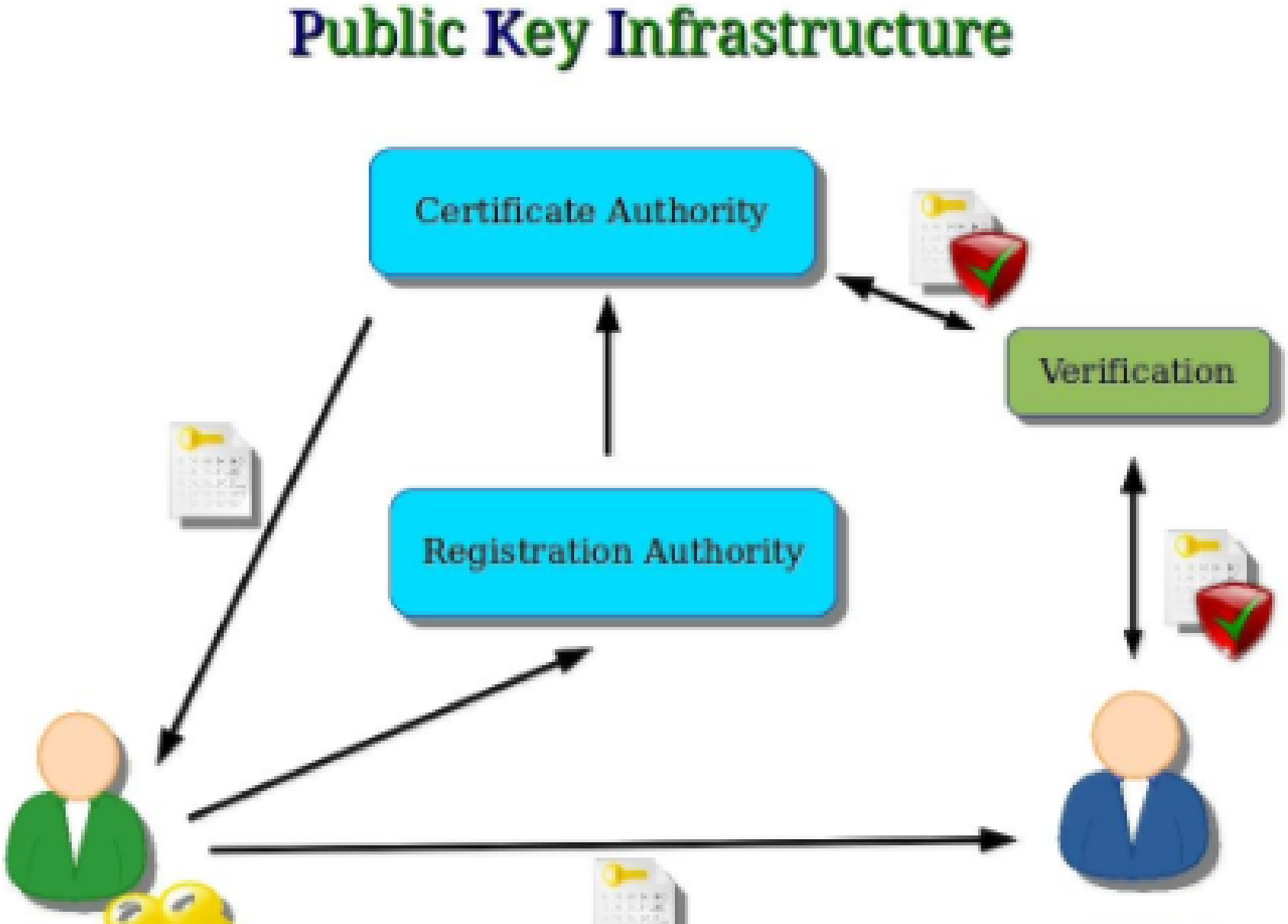
Cross chain attack

The preceding attack still works when a chain is forked. All transactions in one branch can be replicated in the other.
Solution : Sign the hash of the message concatenated with a unique string (eg. "This is bitcoin cash")

Public Key Infrastructure

Problem : I can only guarantee the authenticity of a cryptographic key (which can't have my name written), how do I know that this pk is indeed Alice's ?

Public Key Infrastructure



Random Oracle

Sometimes the aforementioned properties are not enough to prove security. We want to formalize the intuition given at the beginning. We replaces the hash function by a random function. An algortihm then has a chance to break a given property. If this chance is small we say that the property holds int the Radom Oracle Model.

Random Oracle

Collision resistance, pre-image resistance and second pre-image resistance hold in the random oracle model.