Hash functions

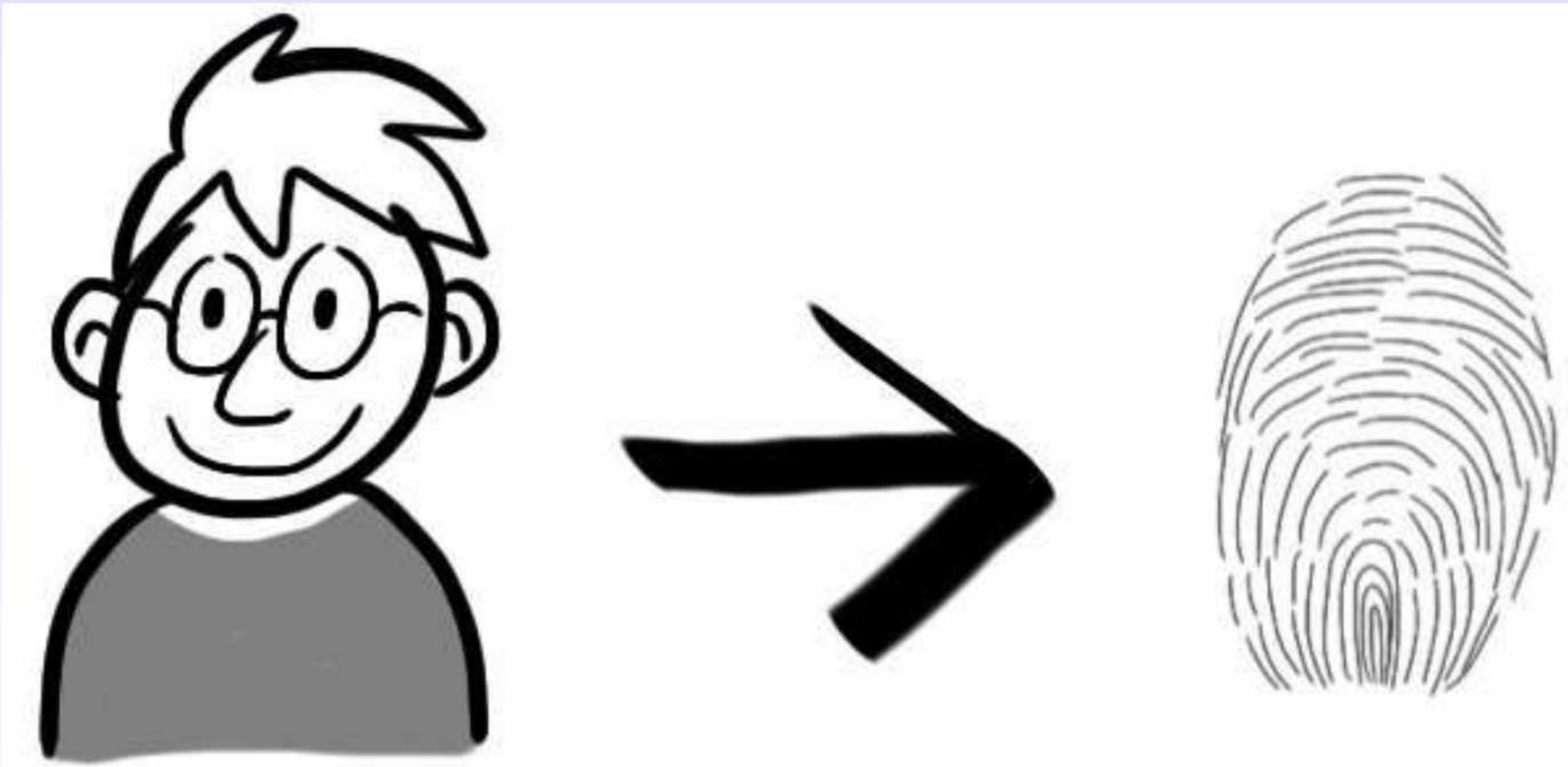
Chanathip Namprempre

Computer Science
Reed College

Agenda: Hash functions

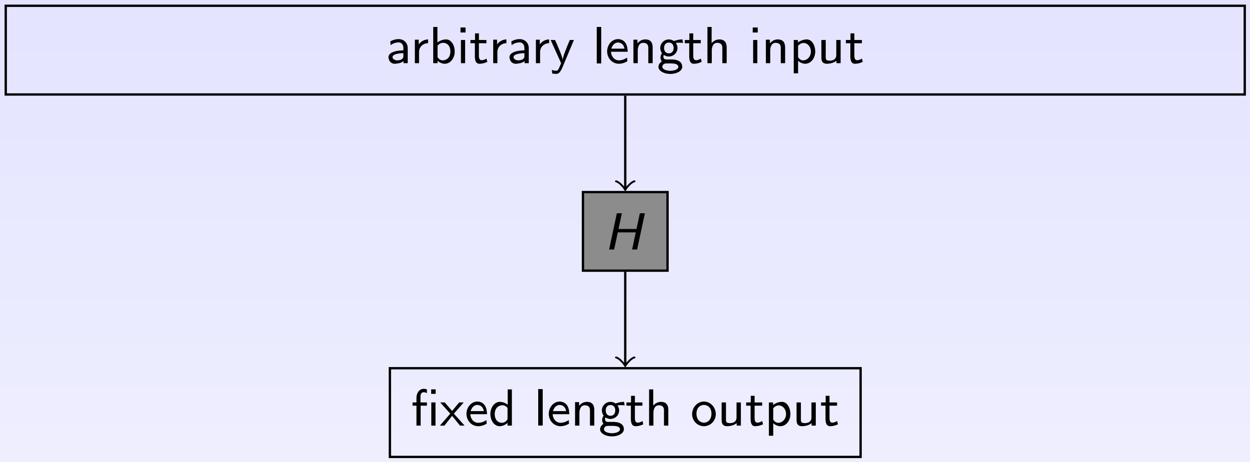
1. Concept
2. Hash function design
3. SHA1
4. Security definitions for hash functions: pre-image attacks
5. Security definitions for hash functions: second pre-image attacks
6. Security definitions for hash functions: collision attacks
7. Examples
8. Birthday paradox

A way to think about hash functions



A way to think about hash functions. The digest is supposed to capture the essense of the object being hashed.

Hash functions



The output is often called a message digest.

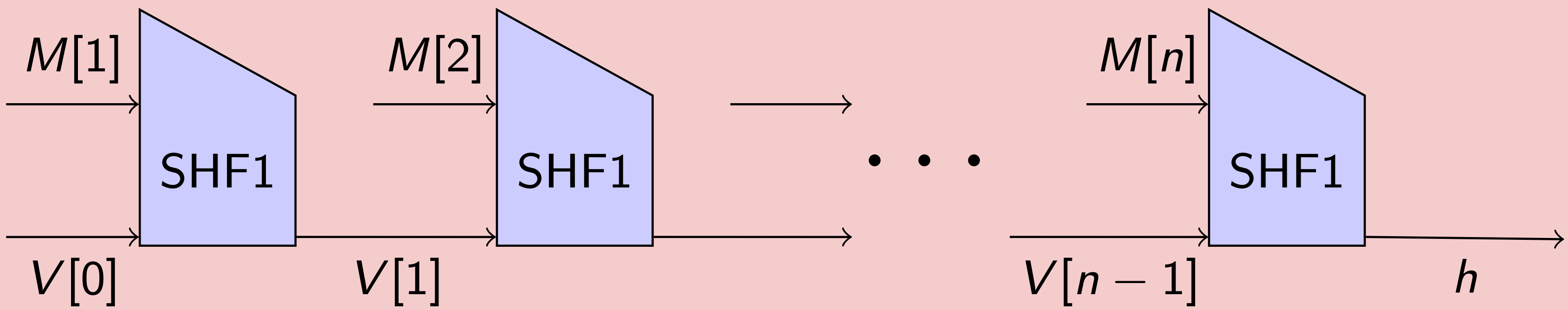
Example: SHA1 : $\{0,1\}^{<2^{64}} \rightarrow \{0,1\}^{160}$

Applications: how hash functions are used

- Password hashing: good idea, but be careful
- Message integrity: bad idea
- Fingerprint of large file: good idea, but make sure the fingerprint is the “real” one.
- Downline load security: same idea as above
- Digital signature efficiency: sign $H(M)$ instead of signing M

Hash function design: Merkle-Damgard

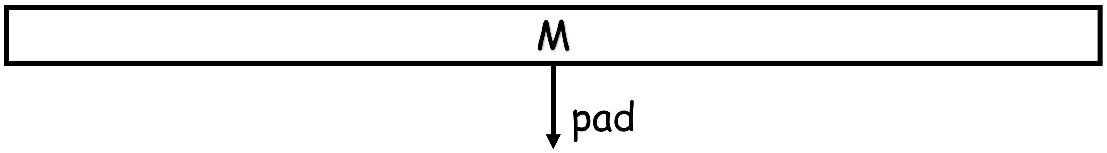
Merkle-Damgard: for example SHA1

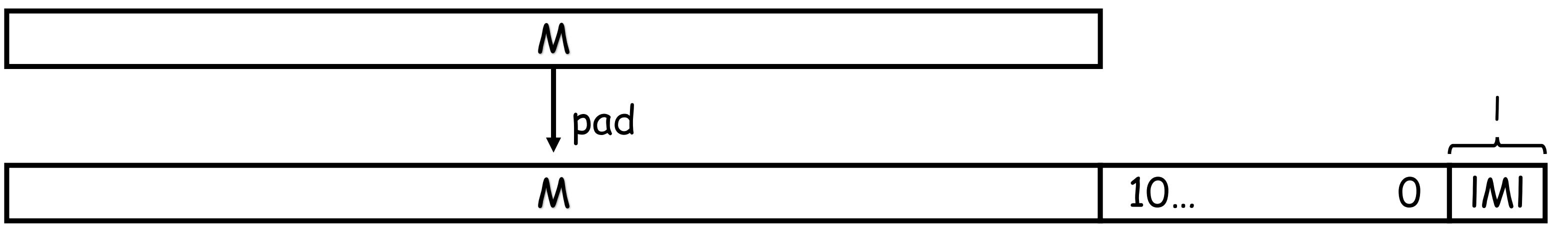


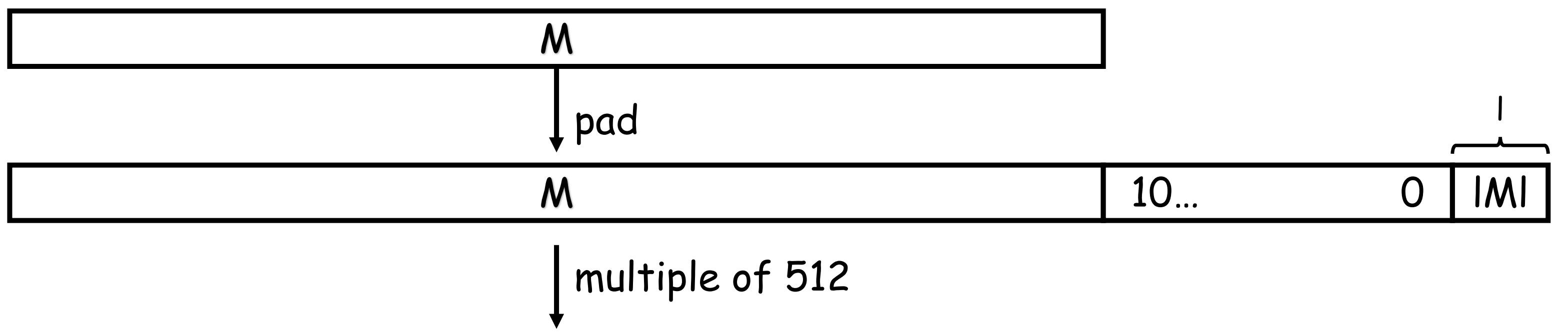
The Merkle-Damgard construction.

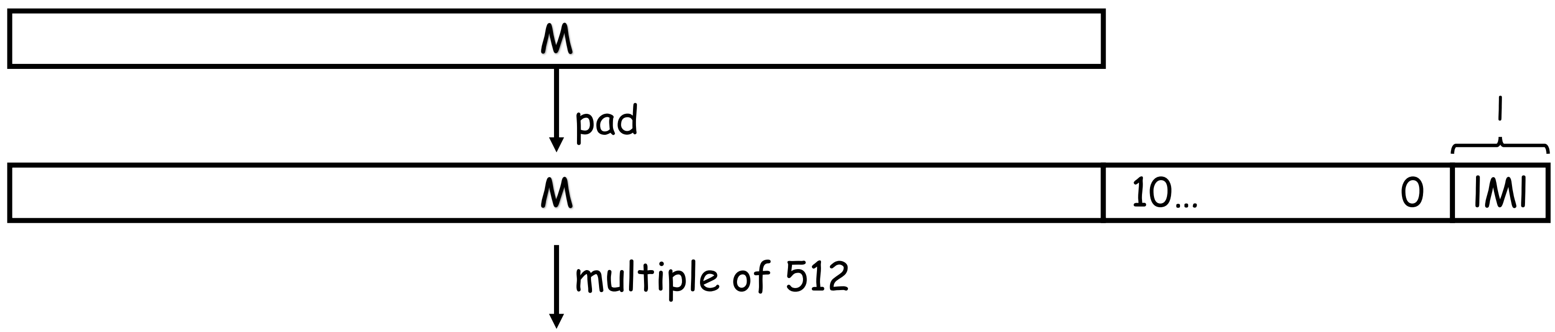
- The trapezoids represent a compression function.
- The initial vector $V[0]$ is a fixed, public value.
- The messages $M[1], \ldots, M[n]$ are fixed-length blocks of the padded input message.
- The output is the output h of the compression function on the last round.

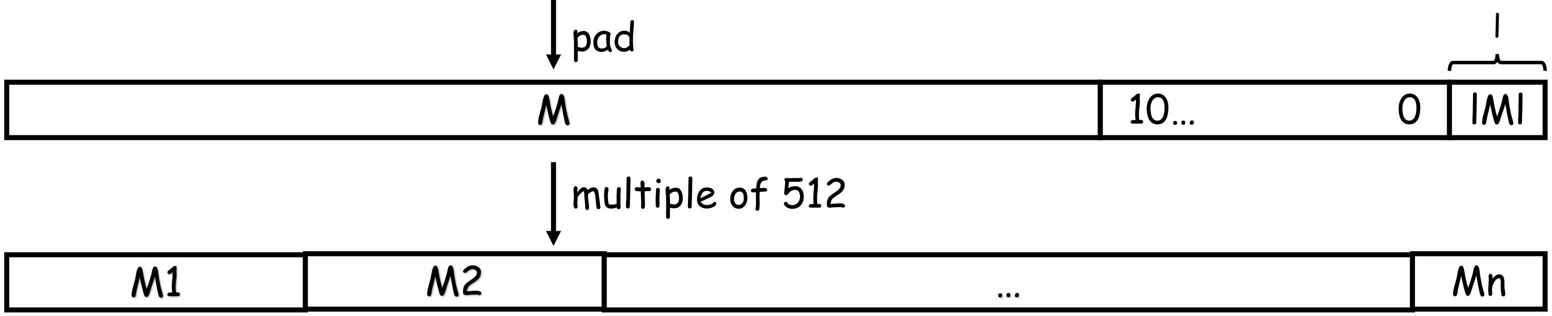
M

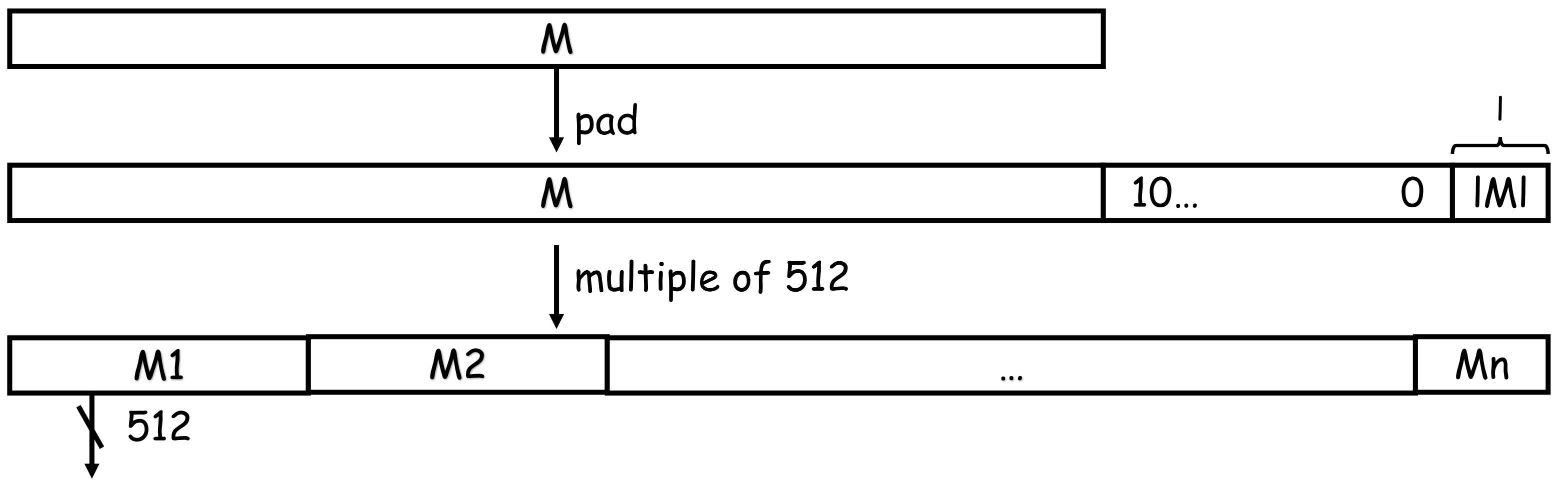

M
pad

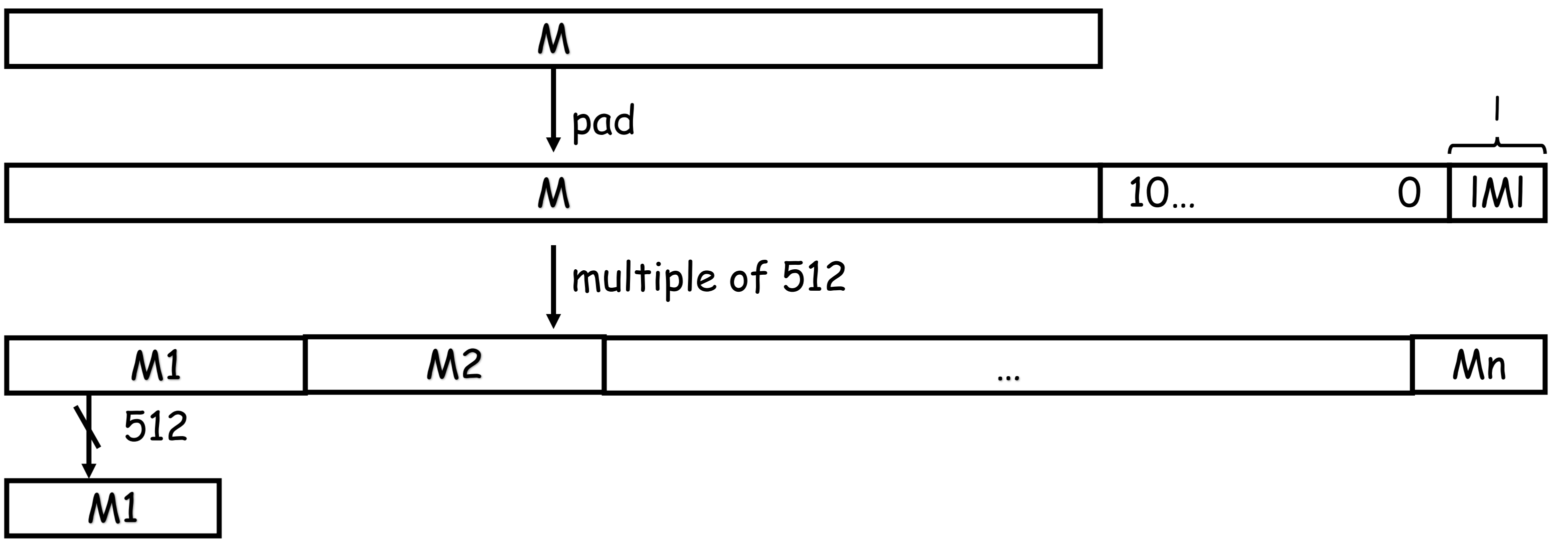

M
pad
l
M
10…
0
|M|

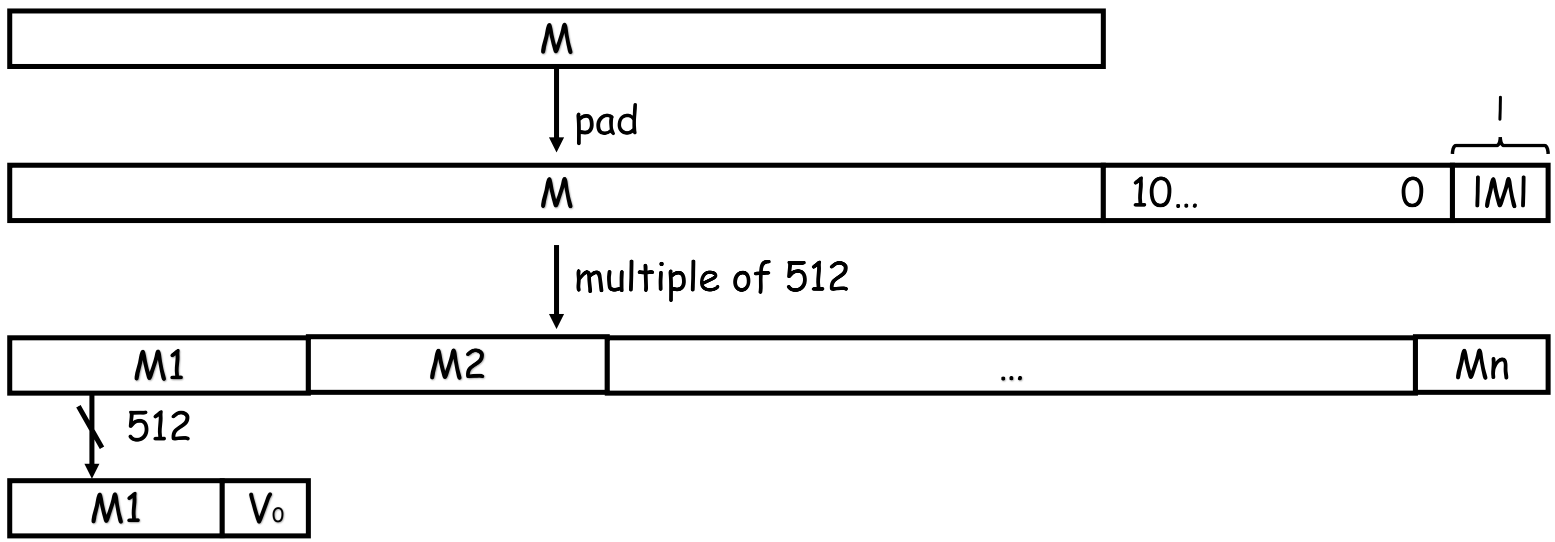

M
pad
l
M
10...
0
|M|
multiple of 512

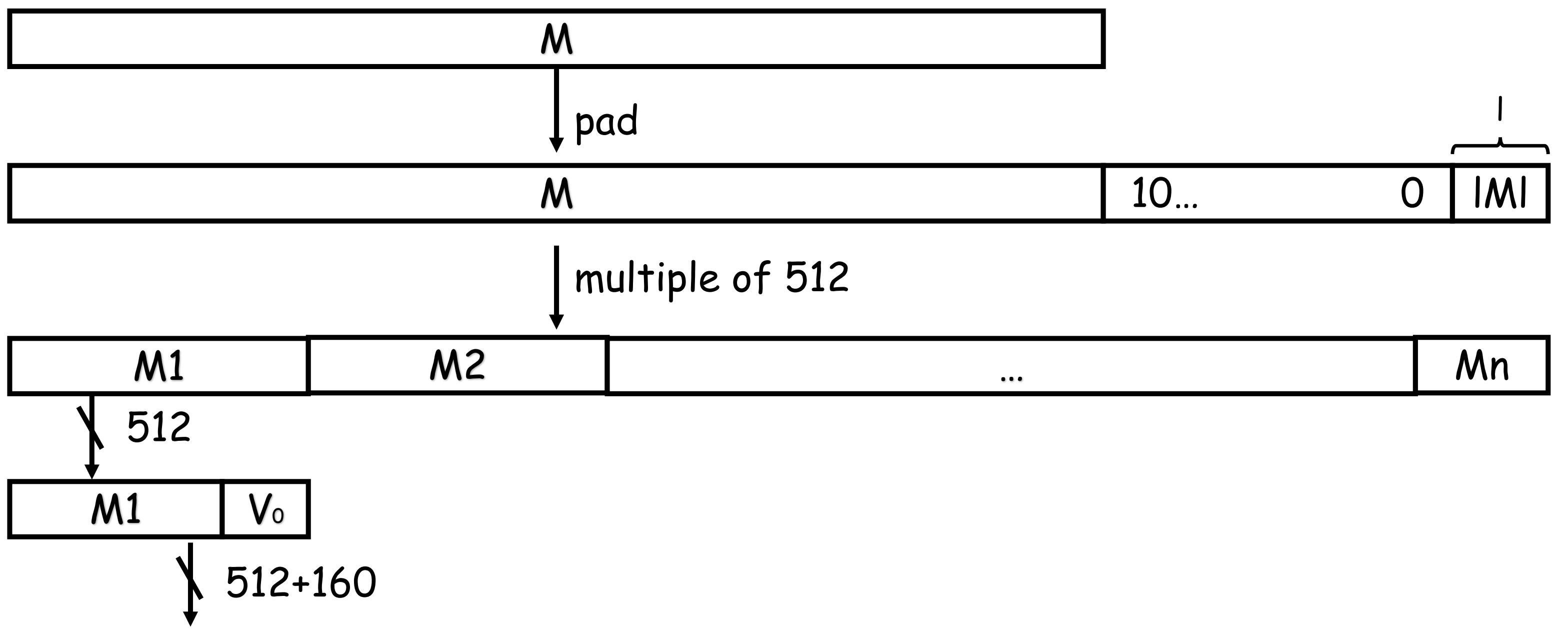

M
pad
l
M
10...
0
|M|
multiple of 512

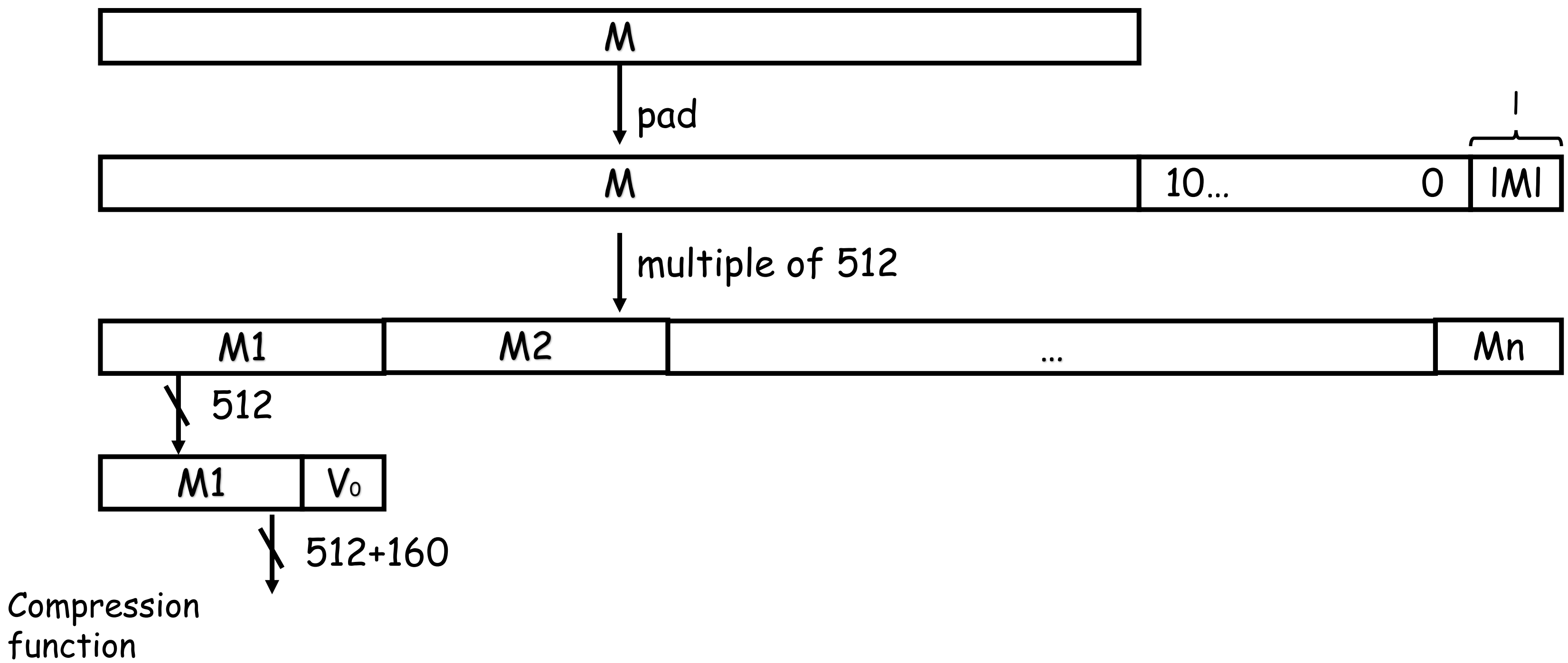

M
pad
l
M
10…
0
|M|
multiple of 512
M1
M2
…
Mn

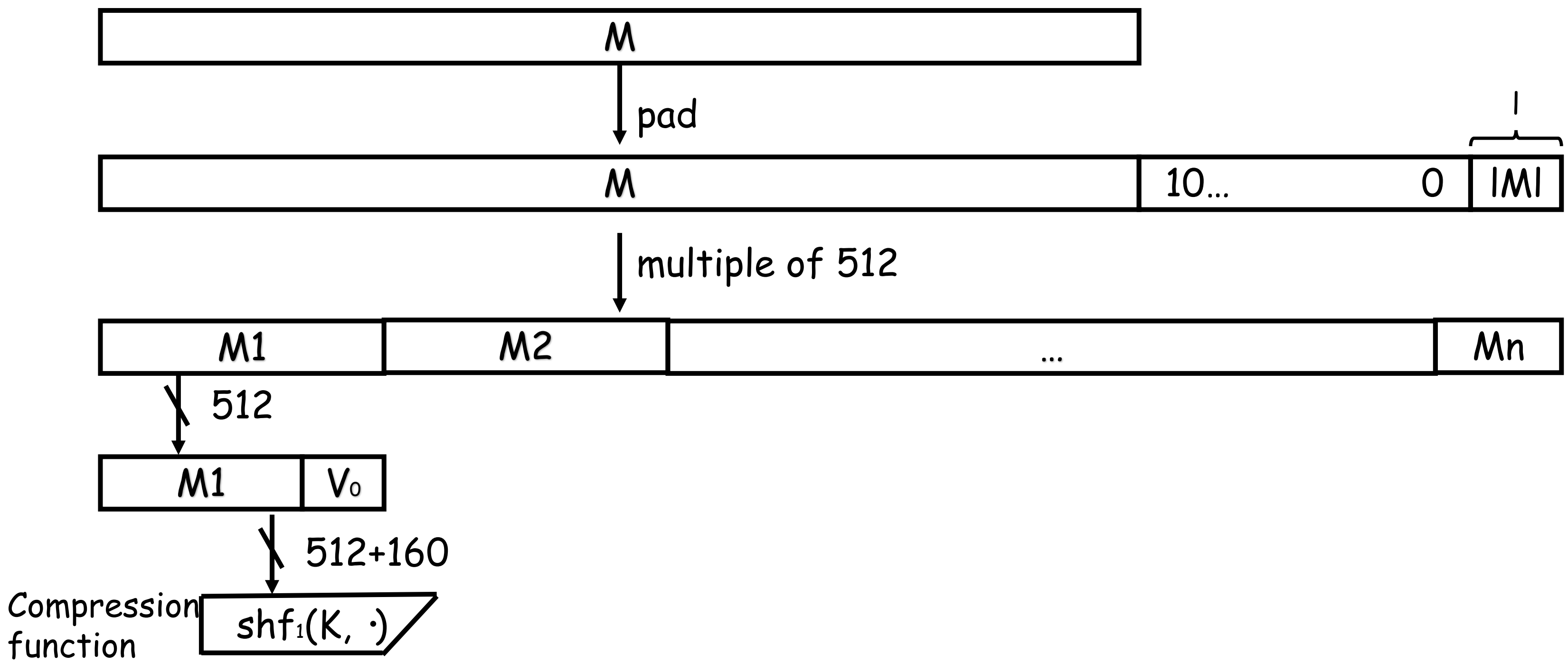

M
pad
l
M
10...
0
|M|
multiple of 512
M1
M2
...
Mn
512

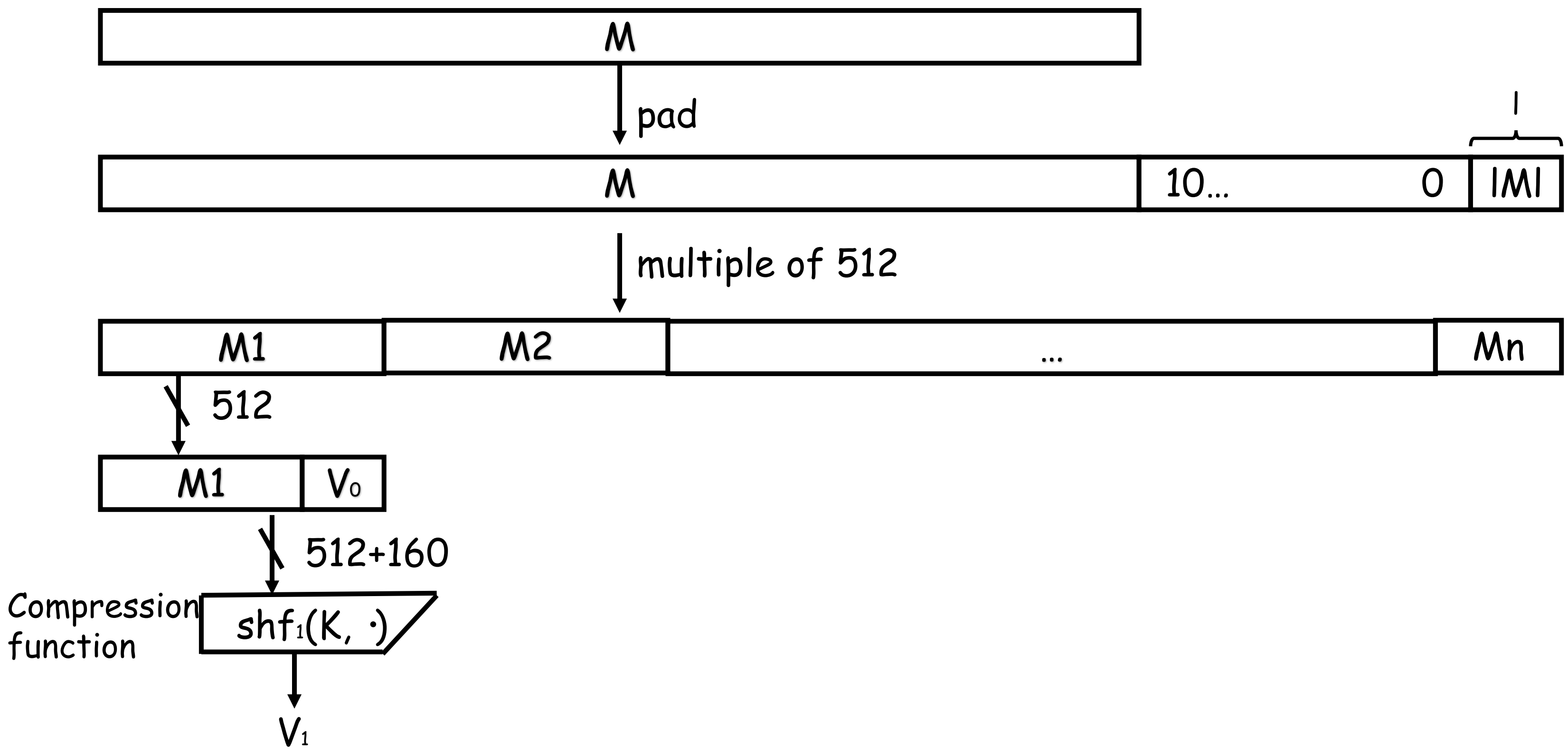

M
pad
l
M
10…
0
|M|
multiple of 512
M1
M2
…
Mn
512
M1

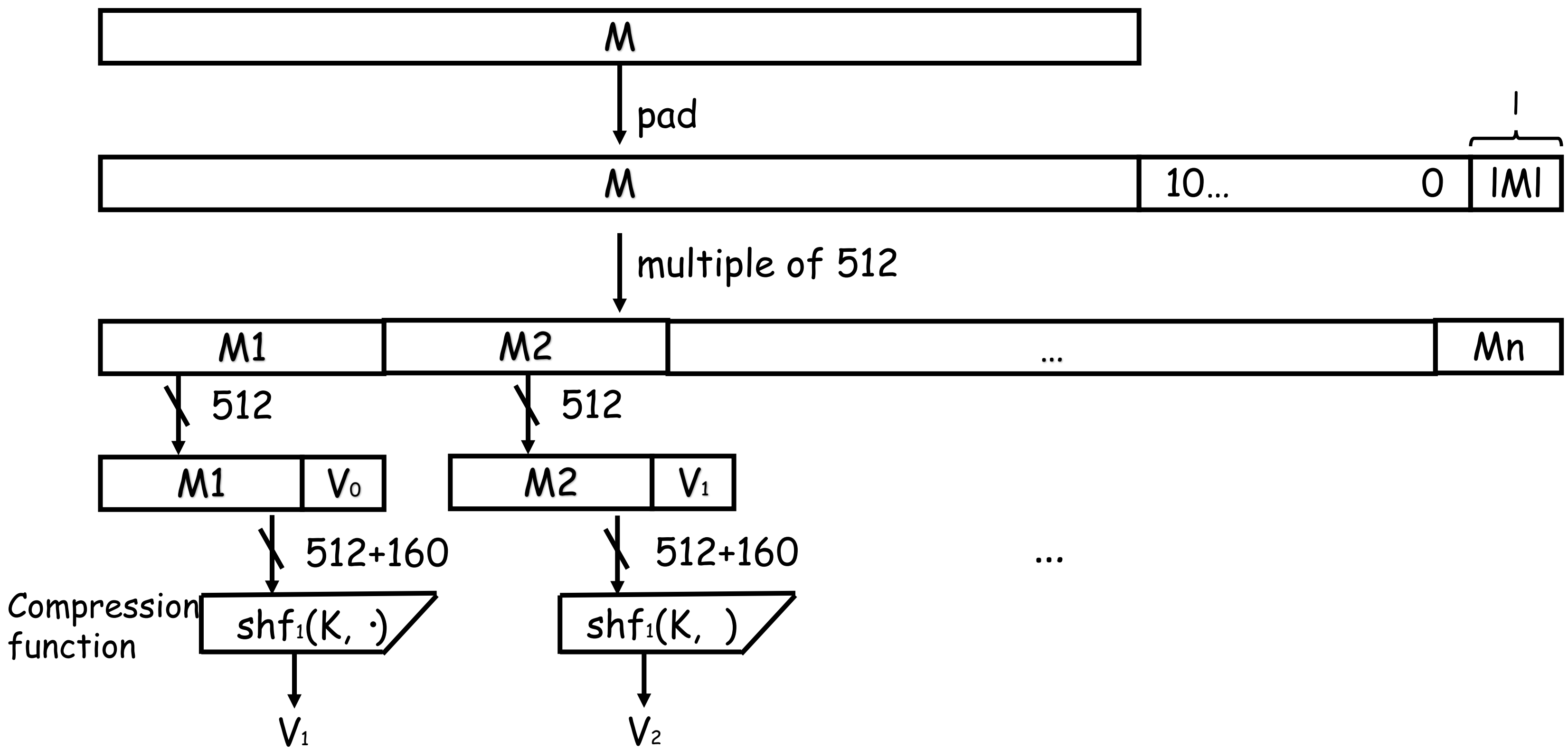

M
pad
l
M
10...
0
|M|
multiple of 512
M1
M2
...
Mn
512
M1
V0

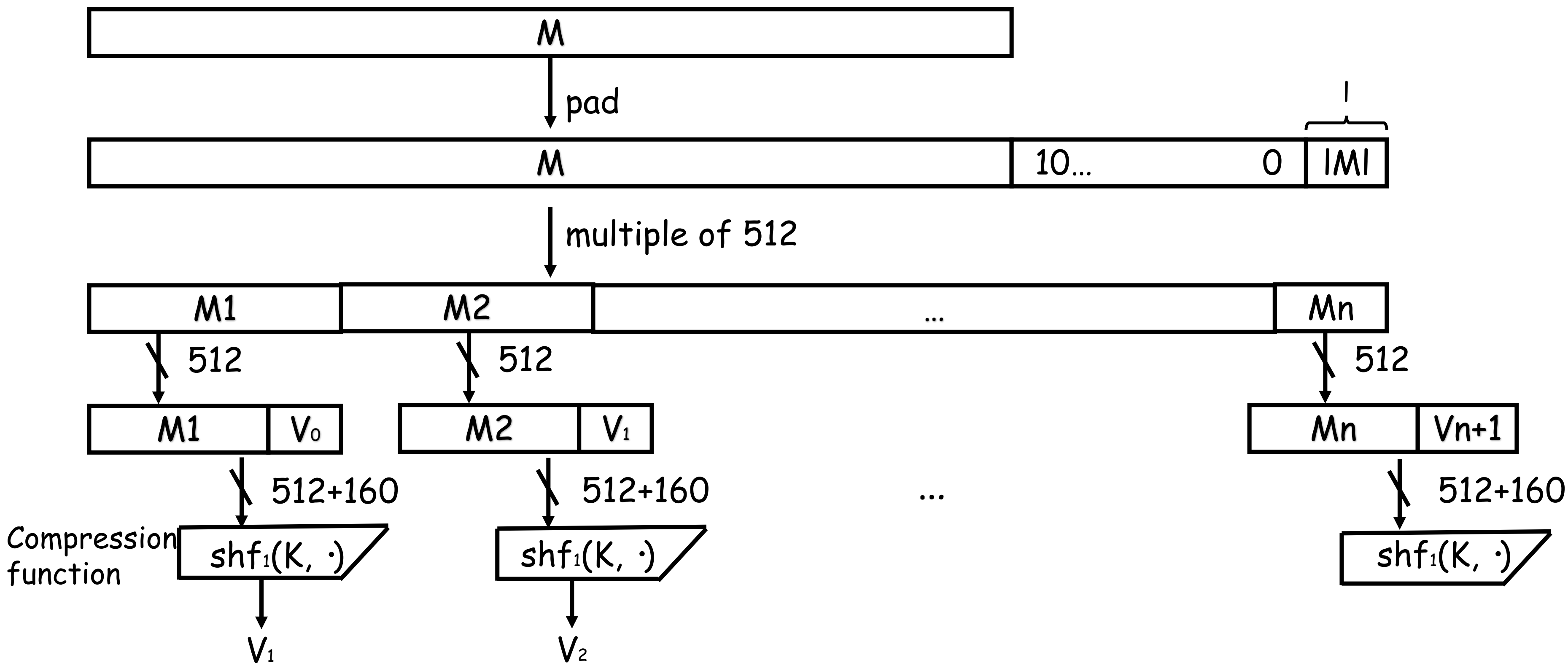

M
pad
l
M
10…
0
|M|
multiple of 512
M1
M2
…
Mn
512
M1
V0
512+160

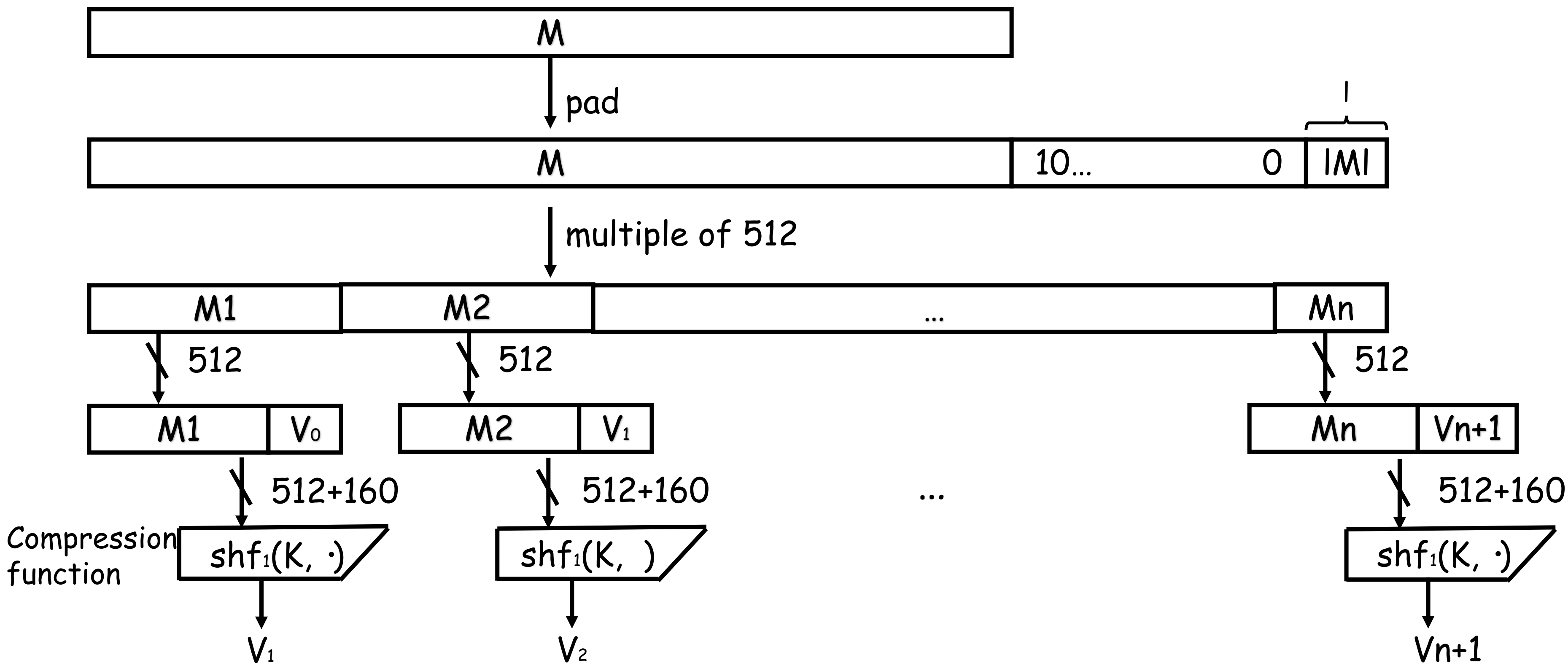

M
pad
l
M
10...
0
|M|
multiple of 512
M1
M2
...
Mn
512
M1
V0
512+160
Compression
function


M
pad
l
M
10...
0
|M|
multiple of 512
M1
M2
...
Mn
512
M1
V0
512+160
Compression function
shf1(K, ·)

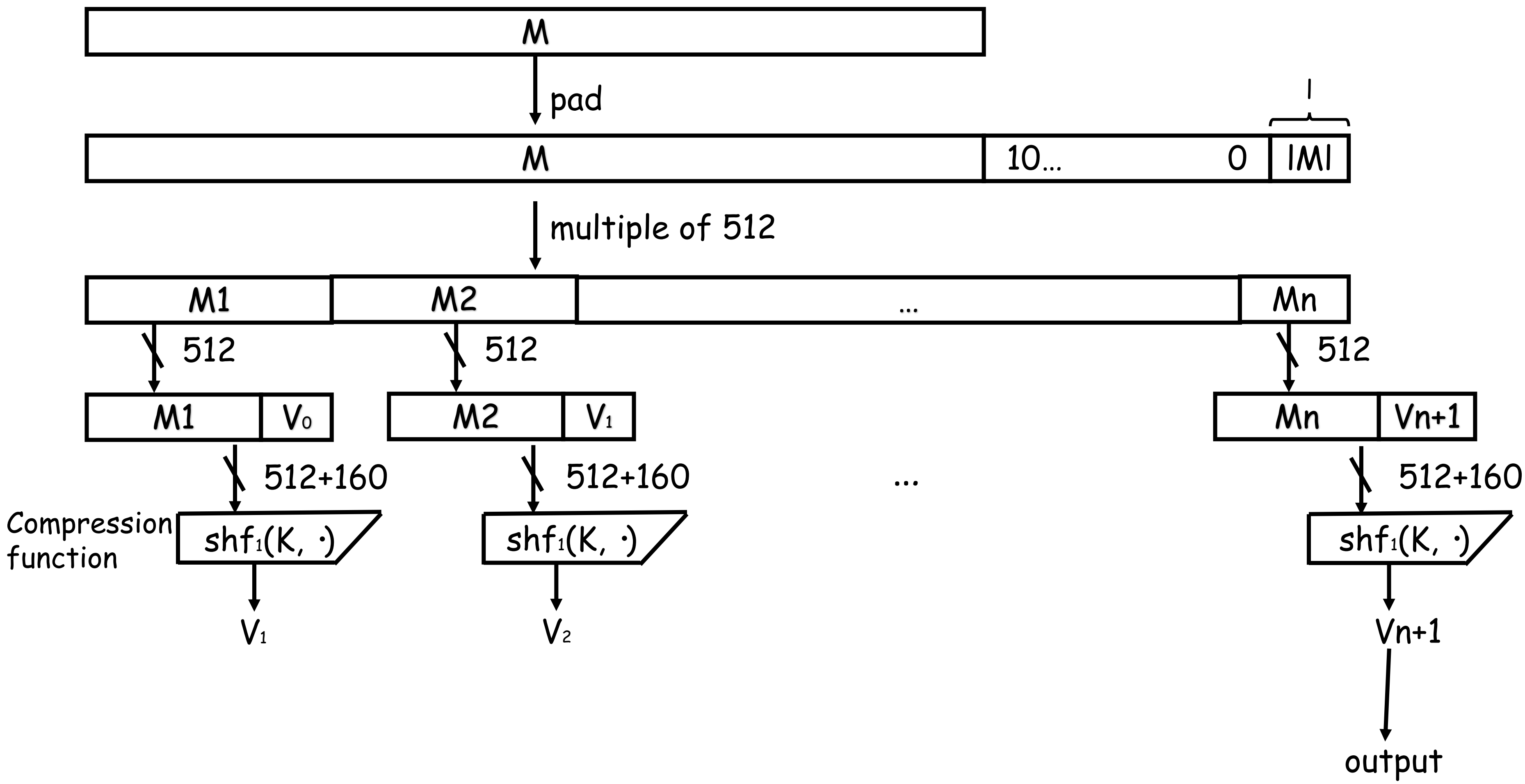

M
pad
l
M
10...
0
|M|
multiple of 512
M1
M2
...
Mn
512
M1
V0
512+160
Compression function
shf1(K, ·)
V1

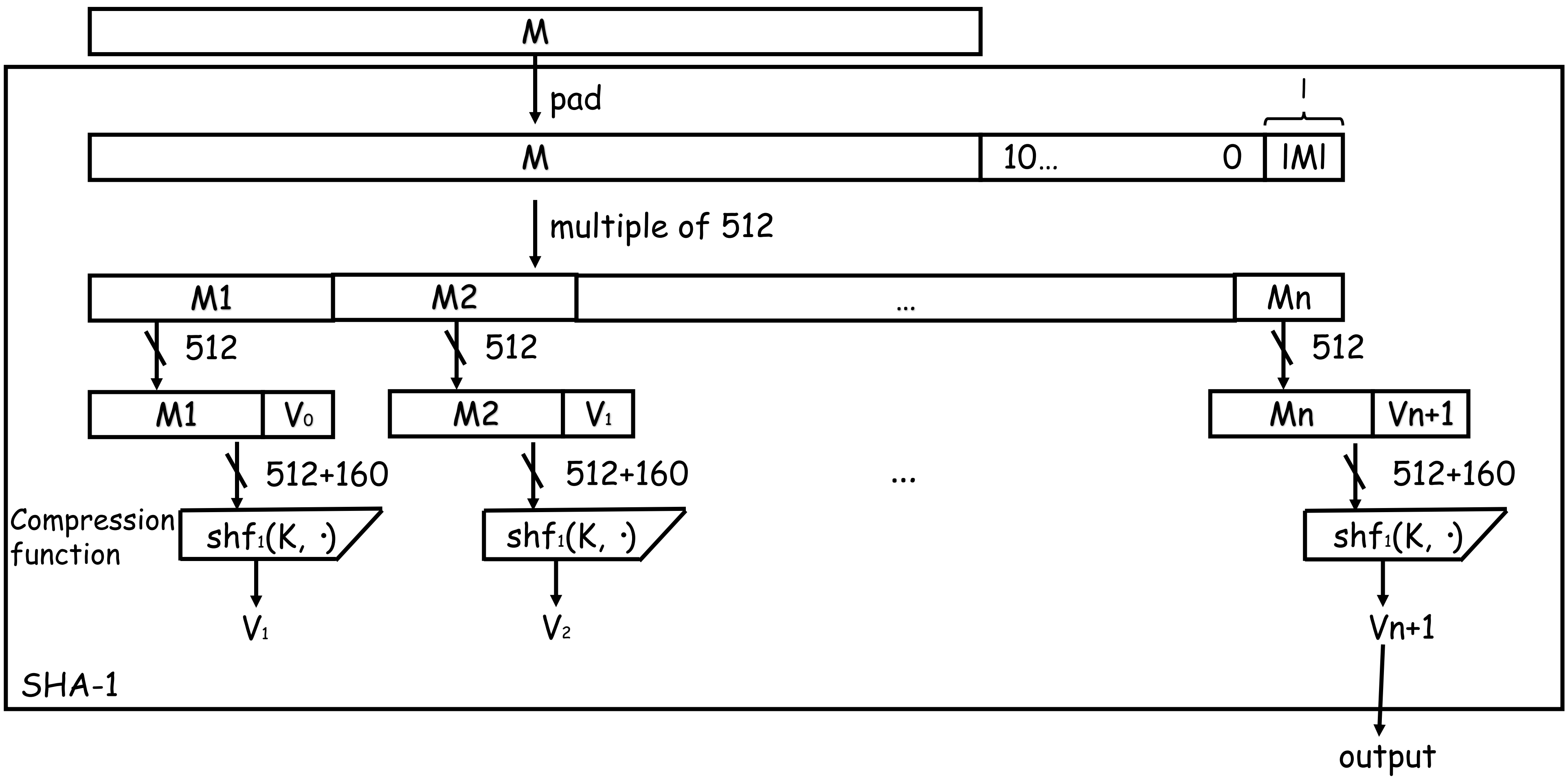

M
pad
l
M
10...
0
|M|
multiple of 512
M1
M2
...
Mn
512
512
M1
V0
M2
V1
512+160
512+160
...
Compression function
shf1(K, ·)
shf1(K,)
V1
V2


M
pad
l
M
10...
0
|M|
multiple of 512
M1
M2
...
Mn
512
512
512
M1
V0
M2
V1
Mn
Vn+1
512+160
512+160
...
512+160
Compression function
shf1(K, ·)
shf1(K, ·)
shf1(K, ·)
V1
V2



M
pad
l
M
10...
0
|M|
multiple of 512
M1
M2
...
Mn
512
512
512
M1
V0
M2
V1
Mn
Vn+1
512+160
512+160
...
512+160
Compression function
shf1(K, ·)
shf1(K,)
shf1(K, ·)
V1
V2
Vn+1


M
pad
M
10...
0
|M|
l
multiple of 512
M1
M2
...
Mn
512
512
512
M1
V0
M2
V1
Mn
Vn+1
512+160
512+160
...
512+160
Compression function
shf1(K, ·)
shf1(K, ·)
shf1(K, ·)
V1
V2
Vn+1
output

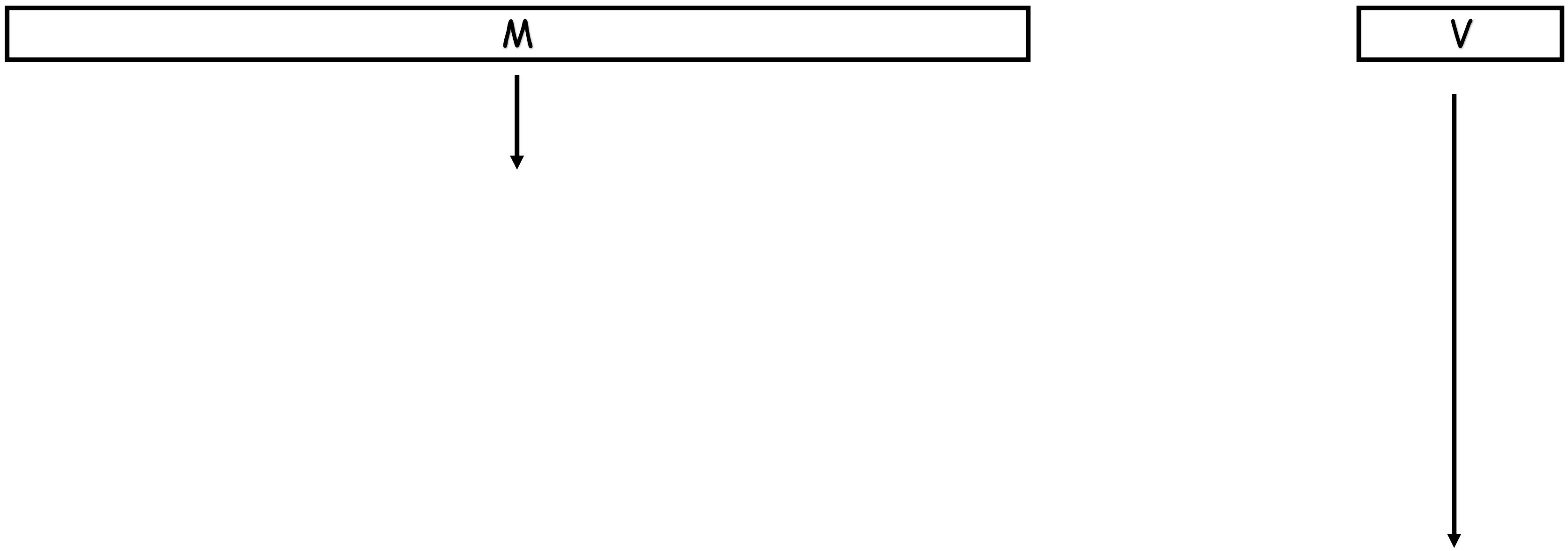

M
pad
l
M
10...
0
|M|
multiple of 512
M1
M2
...
Mn
512
512
512
M1
V0
M2
V1
Mn
Vn+1
512+160
512+160
...
512+160
Compression function
shf1(K, ·)
shf1(K, ·)
shf1(K, ·)
V1
V2
Vn+1
SHA-1
output

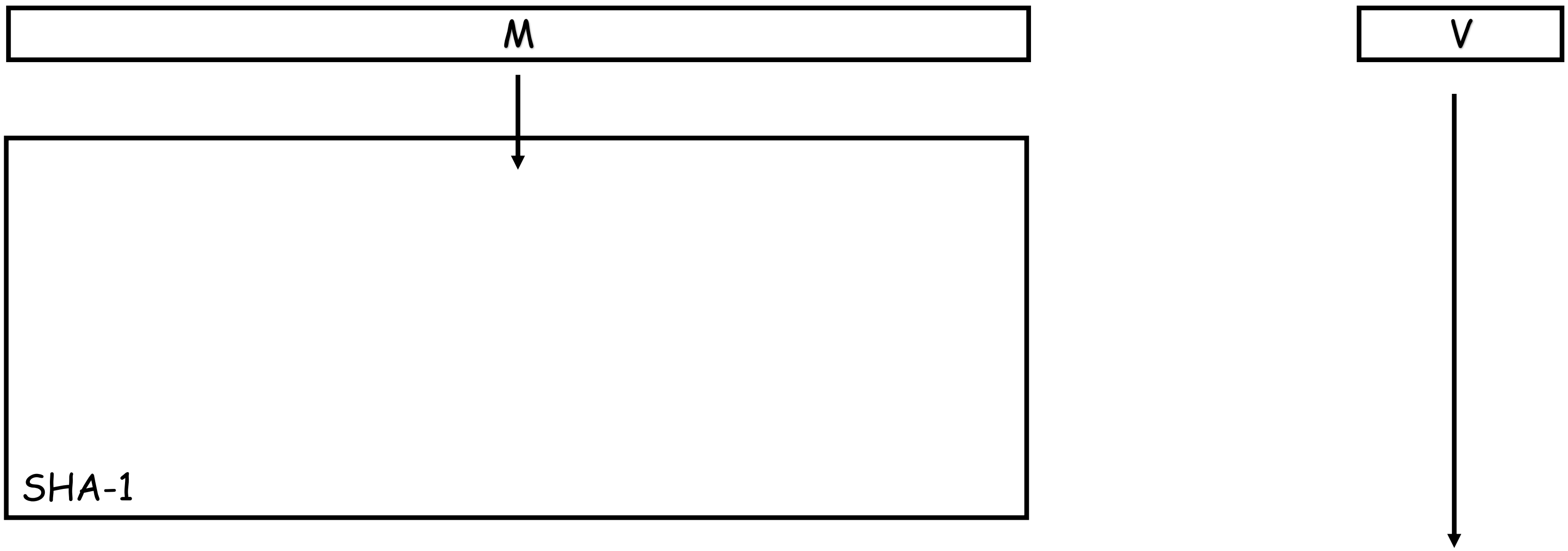
M

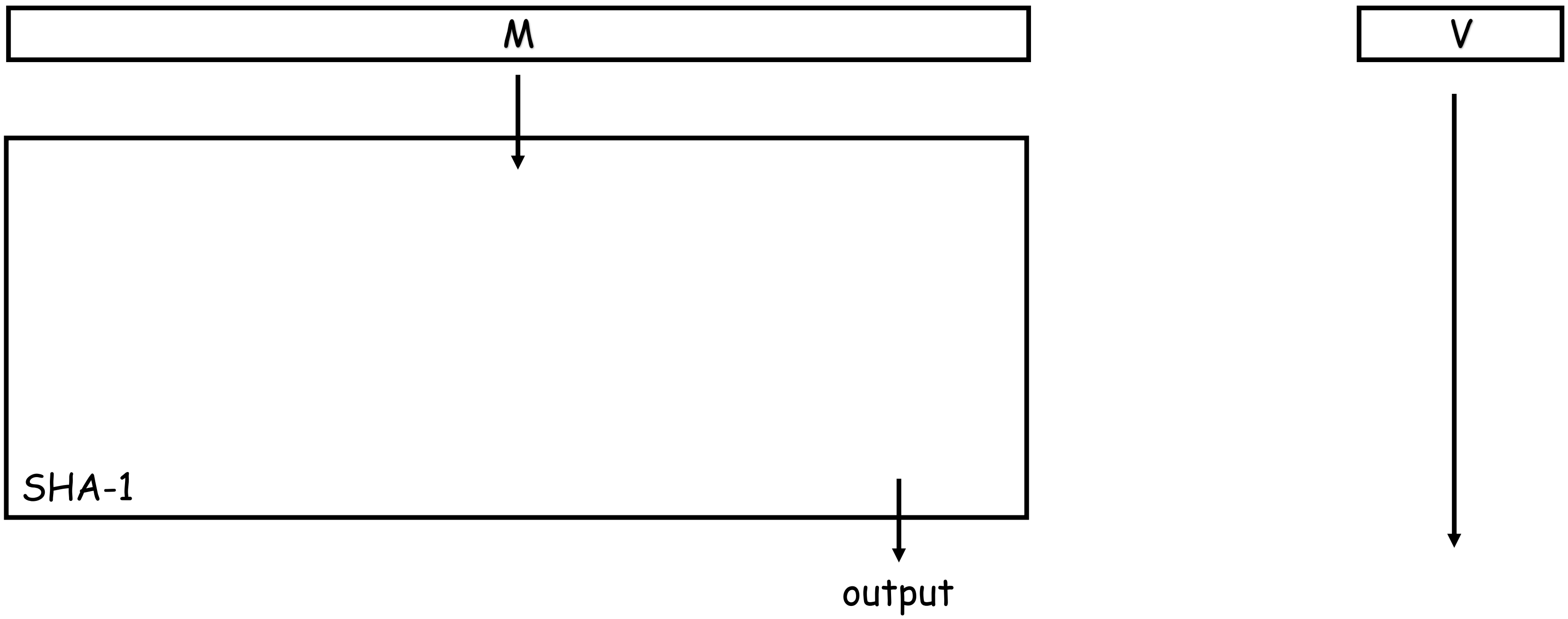
M		V

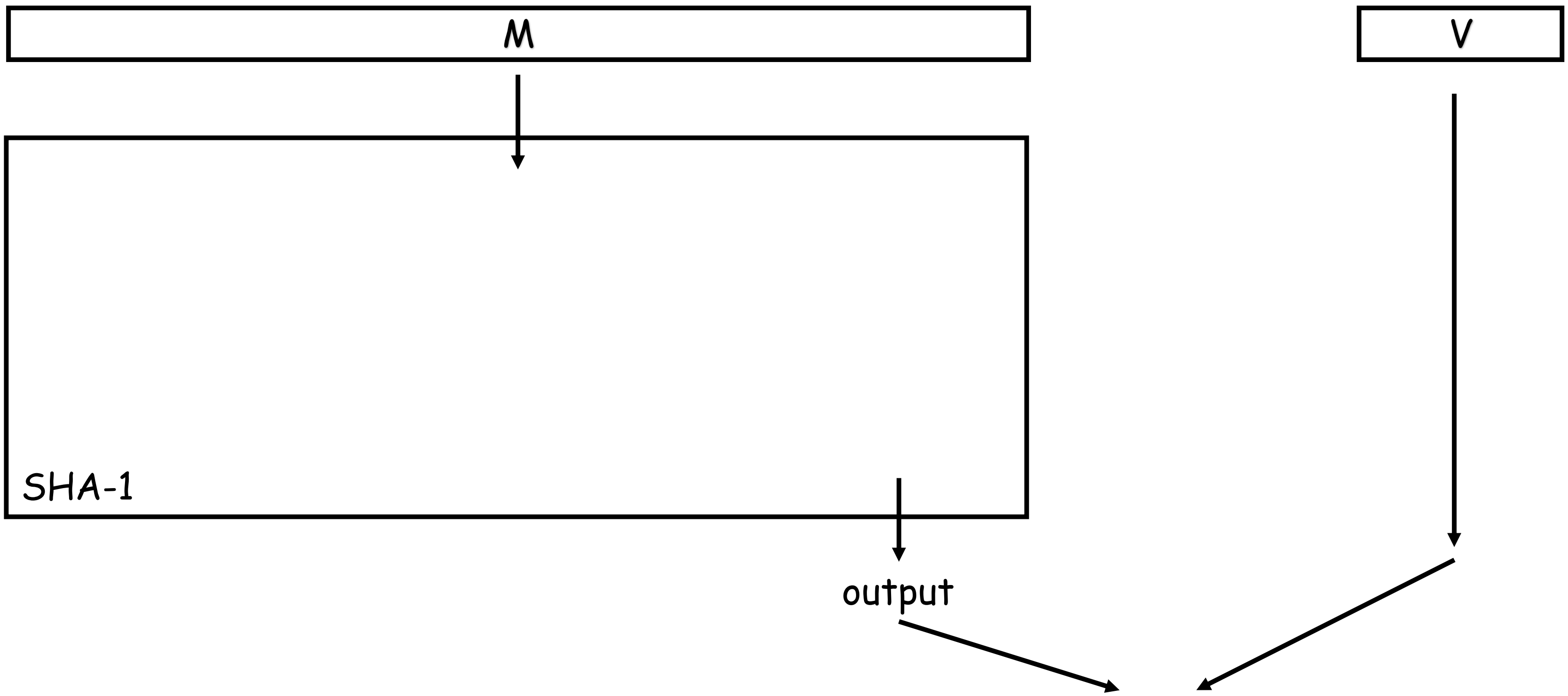
M

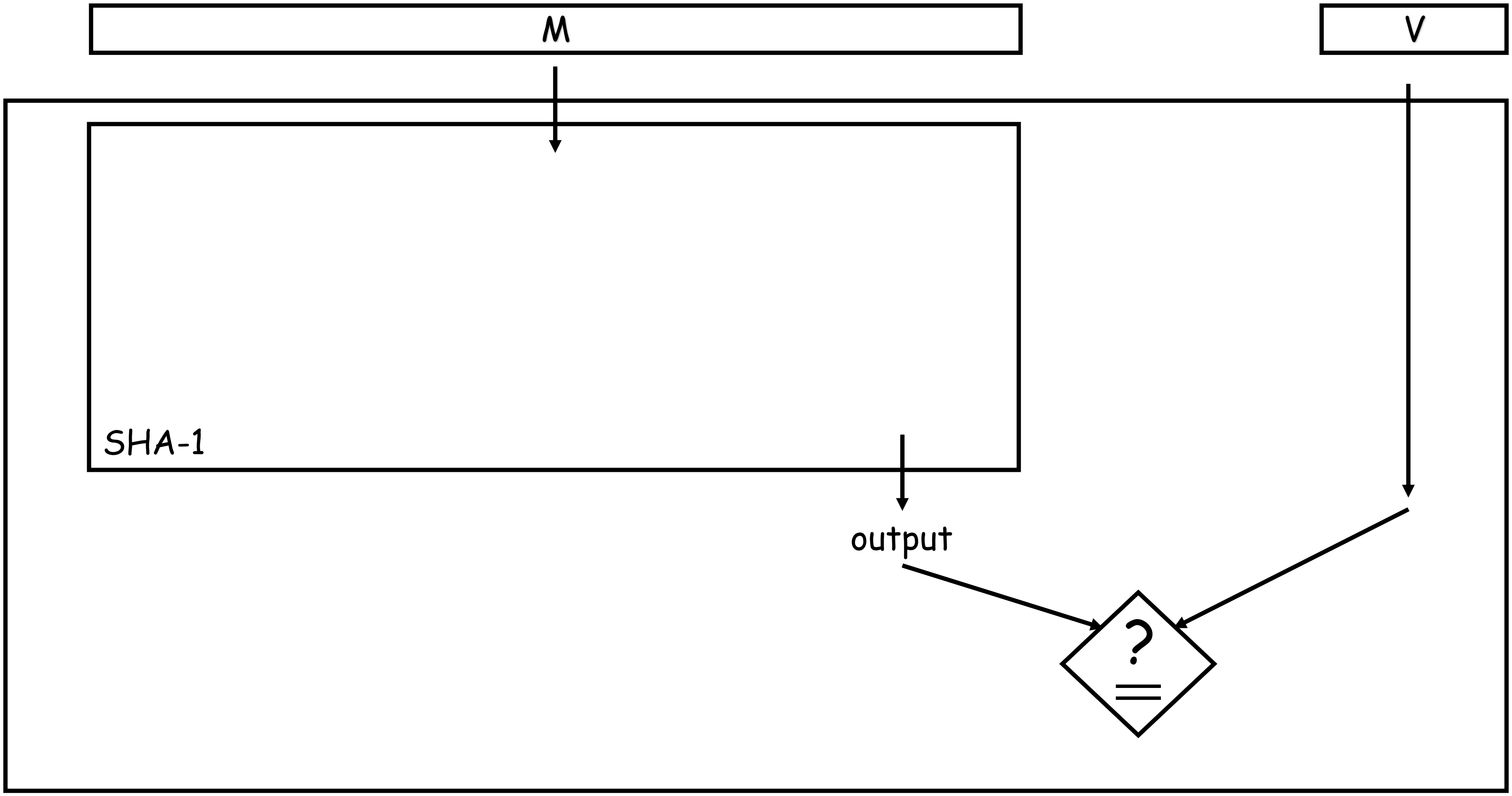
V

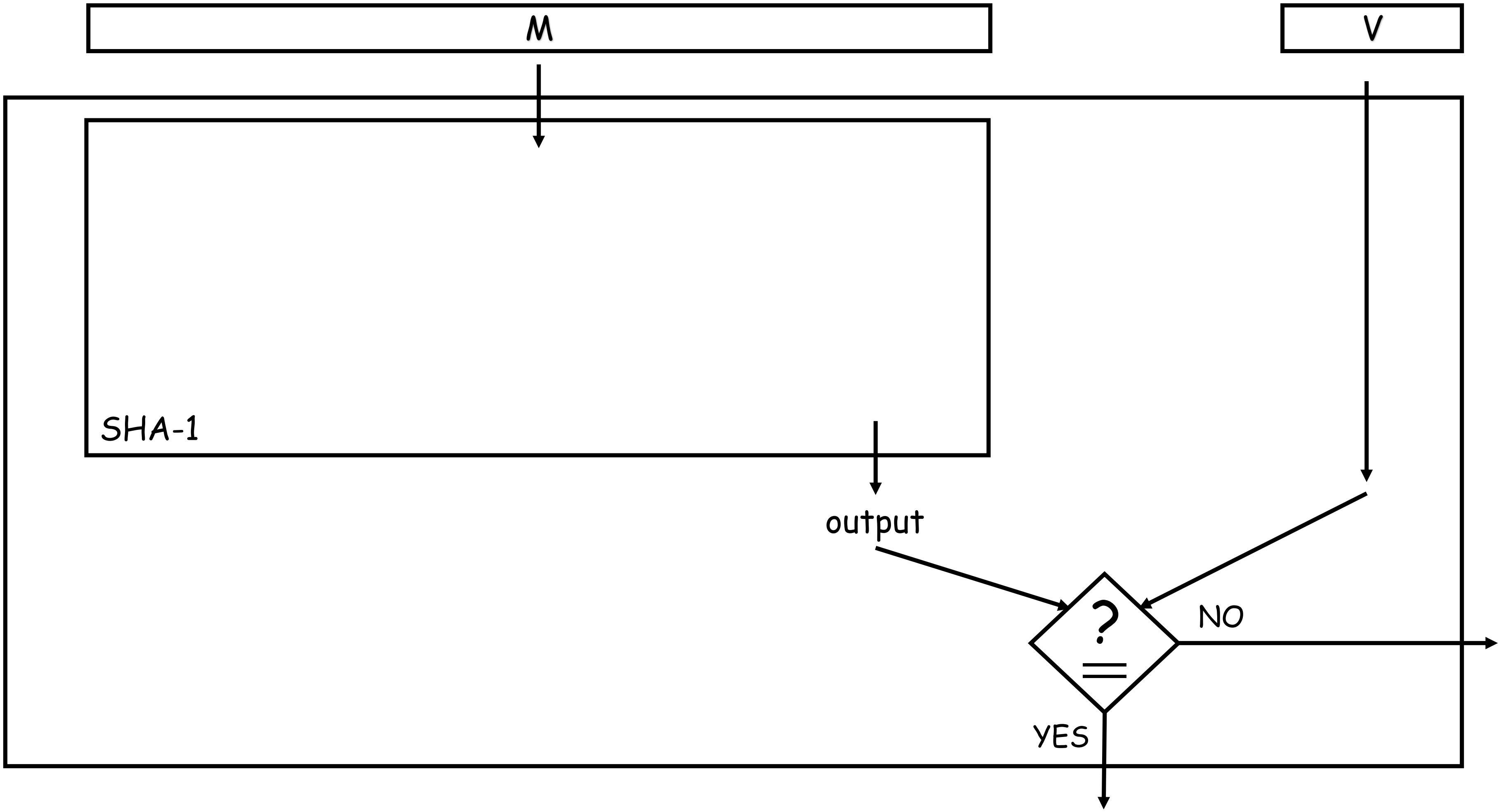

M
V

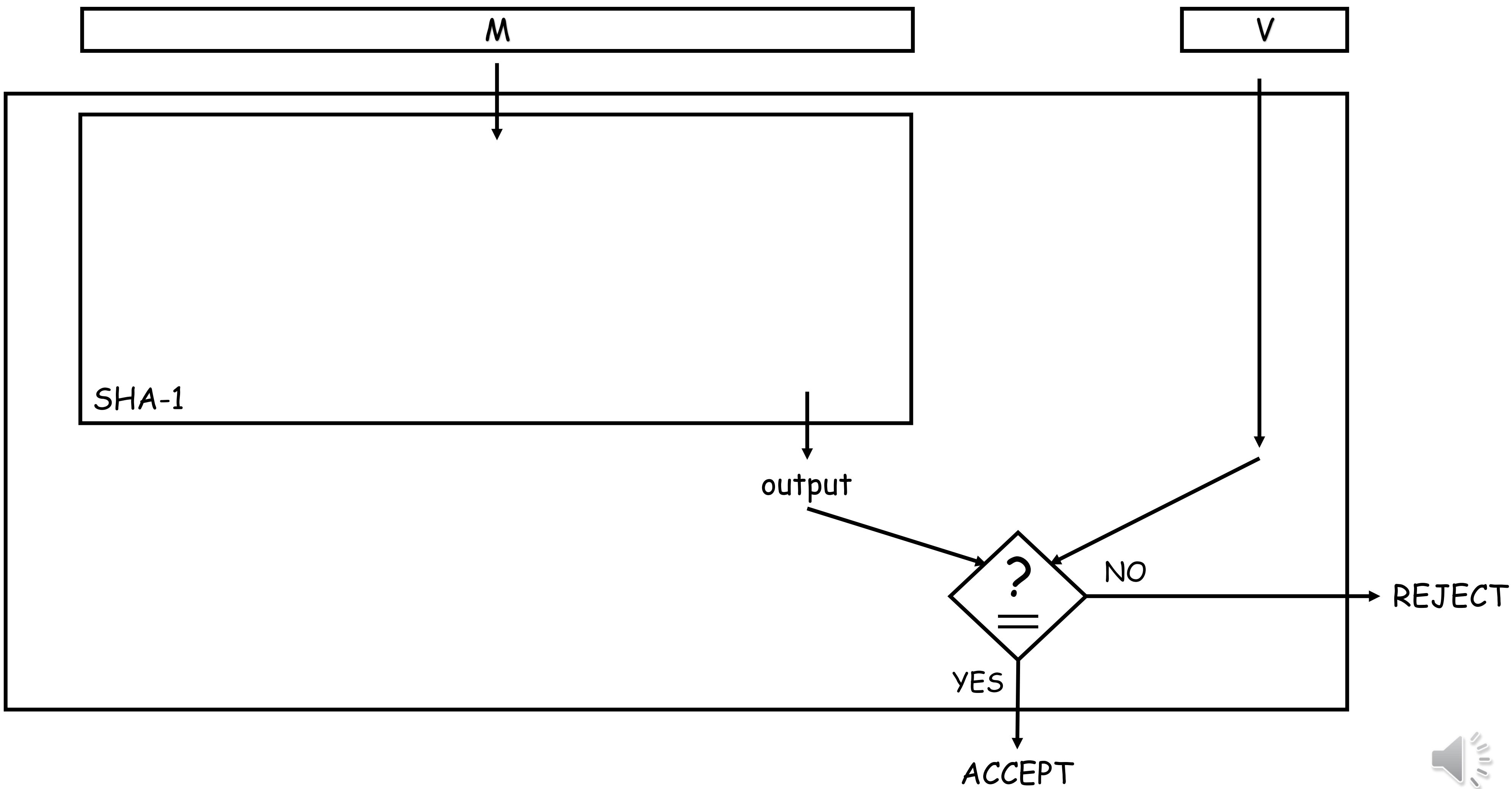

M
V
SHA-1


M
V
SHA-1
output

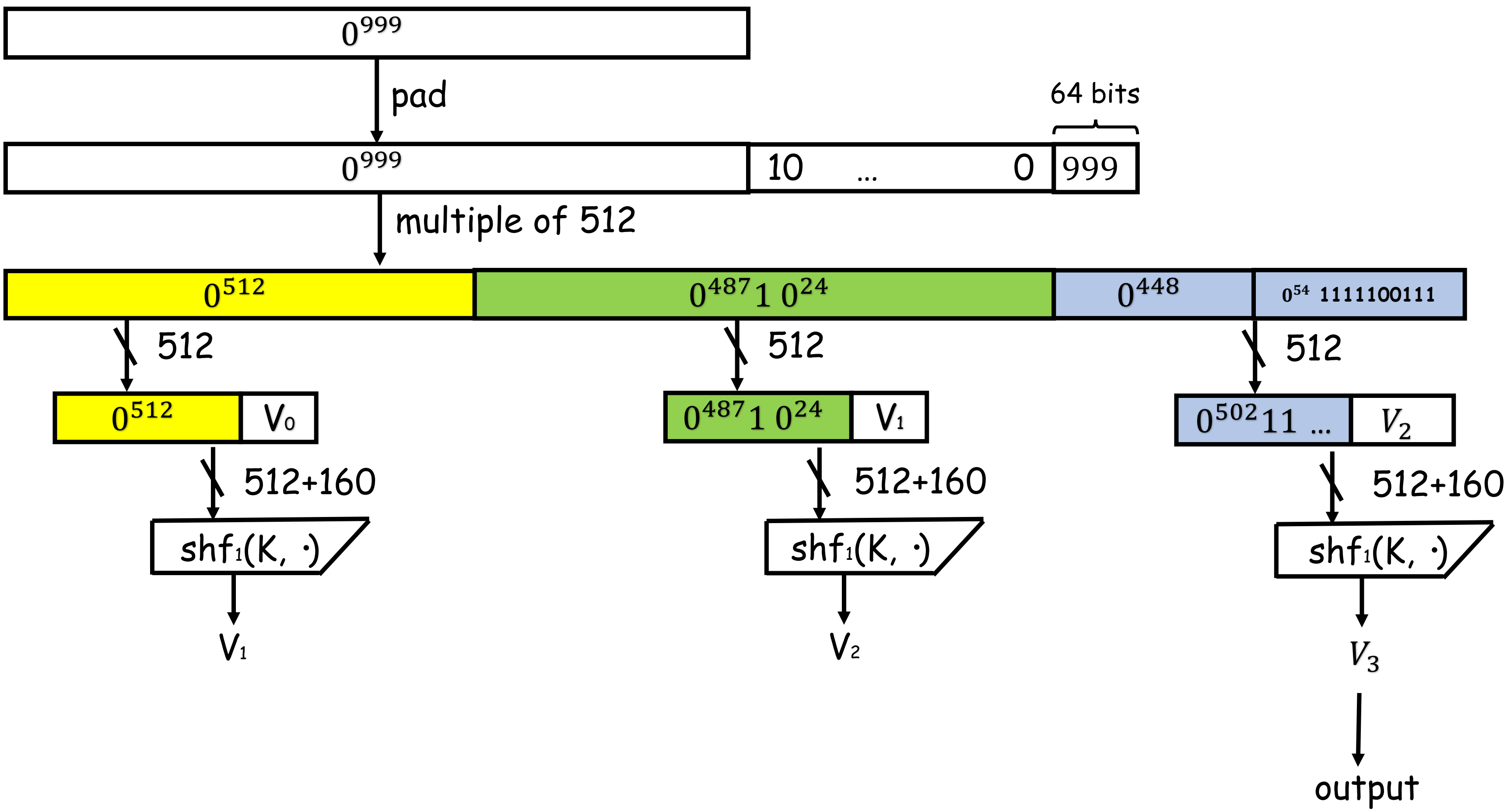

M
V
SHA-1
output


M
V
SHA-1
output
?
=


M
V
SHA-1
output
?
=
NO
YES


M
V
SHA-1
output
?
=
NO
REJECT
YES
ACCEPT

Example: Computing SHA1(0^{999}) = V_3



SHA1

repeatly apply compression function on each block of message

$$160 \text{ bits } \leftarrow V_0 = 37452301||\ldots||C3D2E1F0$$
$$128 \text{ bits } \leftarrow K = 5A827999||\ldots||CA62C1D6$$

Observation: use the word "key" but really the "key" is <u>known</u>!

SHA1 in pseudocode

algorithm SHA1(M) $\quad // \ |M| < 2^{64}$

$\quad V \leftarrow$ SHF1($5A827999 \| 6ED9EBA1 \| 8F1BBCDC \| CA62C1D6, M$)

return V

---

algorithm SHF1(K, M) $\quad // \ |K| = 128$ and $|M| < 2^{64}$

$\quad y \leftarrow$ shapad(M)

$\quad$ Parse y as $M_1 \| M_2 \| \ldots \| M_n$ where $|M_i| = 512 (1 \leq i \leq n)$

$\quad V \leftarrow 67452301 \| EFCDAB89 \| 98BADCFE \| 10325476 \| C3D2E1F0$

$\quad$ for $i = 1, \ldots, n$ do

$\quad\quad V \leftarrow$ shf1($K, M_i \| V$)

return V

---

algorithm shapad(M) $\quad // \ |M| < 2^{64}$

$\quad d \leftarrow (447 - |M|)$ mod 512

$\quad$ Let l be the 64-bit binary representation of $|M|$

$\quad y \leftarrow M \| 1 \| 0^d \| l \quad // \ |y|$ is a multiple of 512

return y

The compression function shf1 in SHA1

```
algorithm shf1(K, B||V)                                        // |K| = 128, |B| = 512, |V| = 160
    Parse B as W_0 || ... || W_15 where |W_i| = 32 (0 ≤ i ≤ 15)
    Parse V as V_0 || ... || V_4 where |V_i| = 32 (0 ≤ i ≤ 4)
    Parse K as K_0 || ... || K_3 where |K_i| = 32 (0 ≤ i ≤ 3)
    for t = 16, ..., 79 do
        W_t ← ROTL^1(W_{t-3} ⊕ W_{t-8} ⊕ W_{t-14} ⊕ W_{t-16})
    A ← V_0 ; B ← V_1 ; C ← V_2 ; D ← V_3 ; E ← V_4
    for t = 0, ..., 19 do
        L_t ← K_0 ; L_{t+20} ← K_1 ; L_{t+40} ← K_2 ; L_{t+60} ← K_3
    for t = 0, ..., 79 do
        if (0 ≤ t ≤ 19) then f ← (B ∧ C) ∨ ((¬B) ∧ D)
        if (20 ≤ t ≤ 39) OR (60 ≤ t ≤ 79) then f ← B ⊕ C ⊕ D
        if (40 ≤ t ≤ 59) then f ← (B ∧ C) ∨ (B ∧ D) ∨ (C ∧ D)
        temp ← ROTL^5(A) + f + E + W_t + L_t
        E ← D ; D ← C ; C ← ROTL^30(B) ; B ← A ; A ← temp
    V_0 ← V_0 + A ; V_1 ← V_1 + B ; V_2 ← V_2 + C ; V_3 ← V_3 + D ; V_4 ← V_4 + E
    V ← V_0 || V_1 || V_2 || V_3 || V_4
return V
```

[All pseudocode is from Bellare-Rogaway lecture notes.]

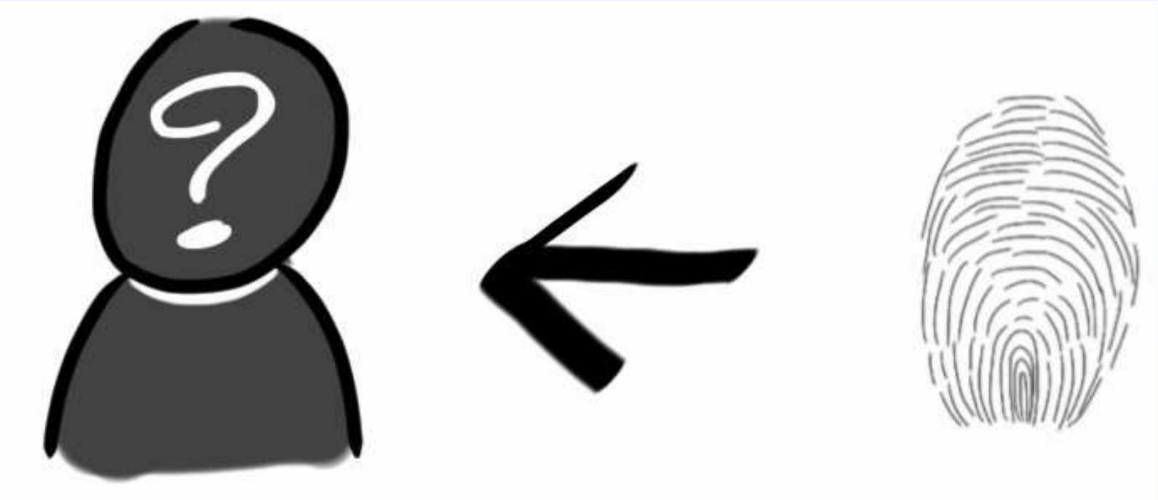
Security definitions for hash functions

What do we expect from hash functions?

Recall the common applications:

- password storage
- allowing people to check integrity of software that they download

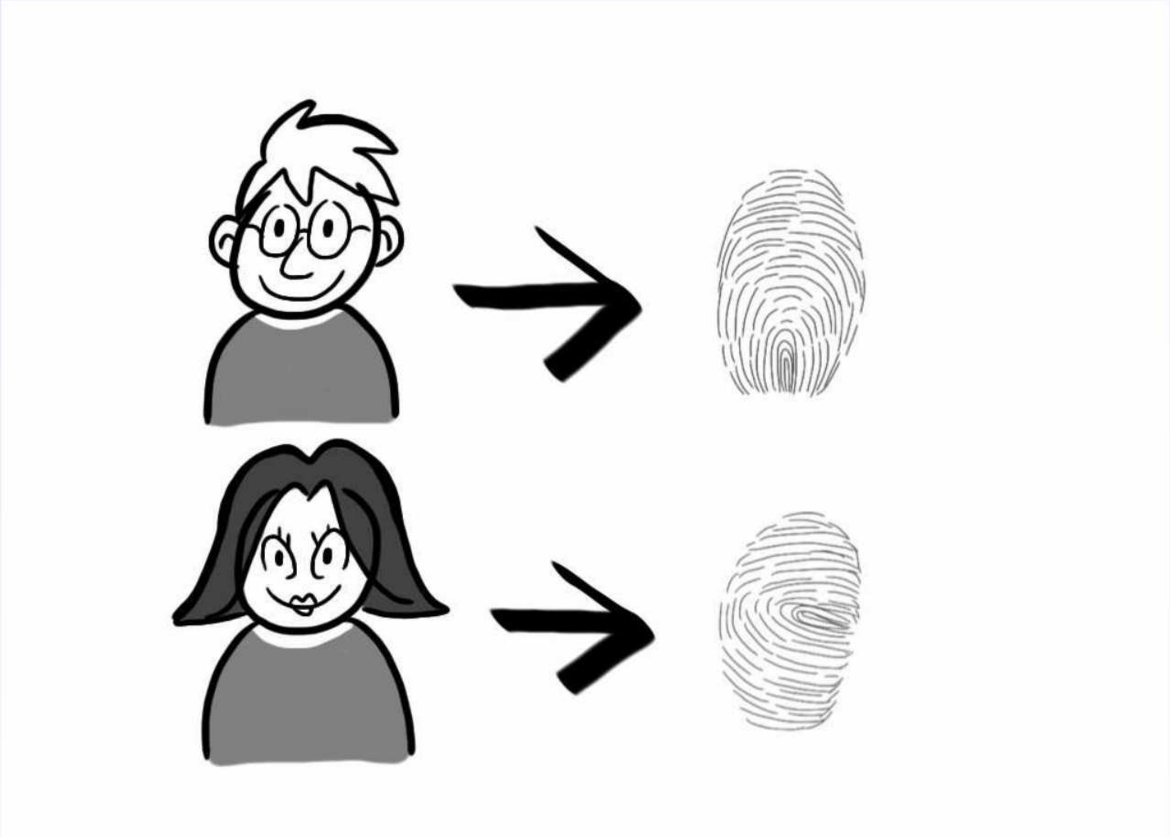
Security definition: Pre-image resistance



Preimage resistance means that it should be difficult to figure out what was hashed simply by looking at the hash value.

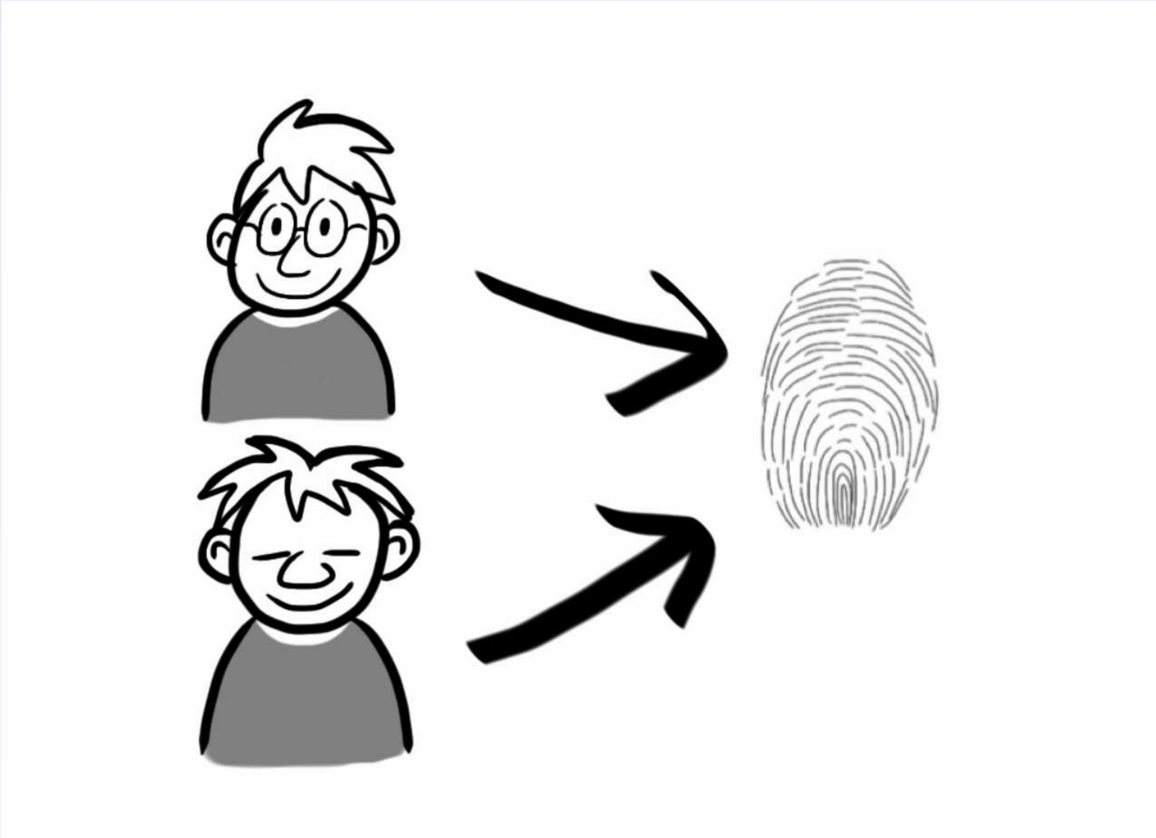
Example application: password storage

Security definition: collision resistance



For collision resistant hash functions, one would expect that, in most cases, hashing two distinct input values would result in two different hash values.

Security definition: collision resistance (cont.)



A collision can always occur, however, since a hash function maps values from a large set to those in a smaller set.

Example application: software integrity check

Example: SHA1

Definition

A collision for a function $h : \mathcal{D} \to \mathcal{R} \equiv$ a pair $x_1, x_2 \in \mathcal{D}$ such that $h(x_1) = h(x_2)$ but $x_1 \neq x_2$

Collision resistance of SHA1:

It is hard to find M and M' such that $\text{SHA1}(M) = \text{SHA1}(M')$ but $M \neq M'$

There are many such M and M' ! (by pigeonhole principle).

Security: resistance against pre-image attacks

Let m, n be integers such that $m > n$.
Let $H : \{0,1\}^m \to \{0,1\}^n$ be a hash function.

Subroutine *Initialize* $x \xleftarrow{\$} \{0,1\}^m$; $h \leftarrow H(x)$ Return h Subroutine *Finalize*(x') Return $(H(x') = h)$	Experiment $\mathbf{Exp}_H^{\mathrm{pre}}(A)$ $h \leftarrow$ *Initialize* $x' \xleftarrow{\$} A(h)$ Return *Finalize*(x')

pre-image advantage

The pre-image advantage of an adversary A mounting a pre-image attack against H is

$$\mathbf{Adv}_H^{\mathrm{pre}}(A) = \Pr\left[\,\mathbf{Exp}_H^{\mathrm{pre}}(A) \Rightarrow \mathrm{true}\,\right] .$$

Security: resistance against second pre-image attacks

Let m, n be integers such that $m > n$.
Let $H : \{0,1\}^m \to \{0,1\}^n$ be a hash function.

Subroutine *Initialize* $x \xleftarrow{\$} \{0,1\}^m$ Return x Subroutine $Finalize(x')$ Return $(H(x) = H(x') \wedge x \neq x')$	Experiment $\mathbf{Exp}_H^{\mathrm{sec}}(A)$ $x \leftarrow Initialize$ $x' \xleftarrow{\$} A(x)$ Return $Finalize(x')$

second pre-image advantage

The second pre-image advantage of an adversary A mounting a second pre-image attack against H is

$$\mathbf{Adv}_H^{\mathrm{sec}}(A) = \Pr\left[\, \mathbf{Exp}_H^{\mathrm{sec}}(A) \Rightarrow \mathrm{true} \,\right] .$$

Security: collision resistance

Let m, n be integers such that $m > n$.
Let $H : \{0,1\}^m \to \{0,1\}^n$ be a hash function.

Subroutine *Initialize* Return Subroutine *Finalize*(x, x') Return $(H(x) = H(x') \wedge x \neq x')$	Experiment $\mathbf{Exp}_H^{\mathrm{coll}}(A)$ *Initialize* $(x, x') \xleftarrow{\$} A$ Return *Finalize*(x, x')

collision advantage

The collision advantage of an adversary A mounting a collision attack against H is

$$\mathbf{Adv}_H^{\mathrm{coll}}(A) = \Pr\left[\, \mathbf{Exp}_H^{\mathrm{coll}}(A) \Rightarrow \text{true} \,\right] \ .$$

Examples

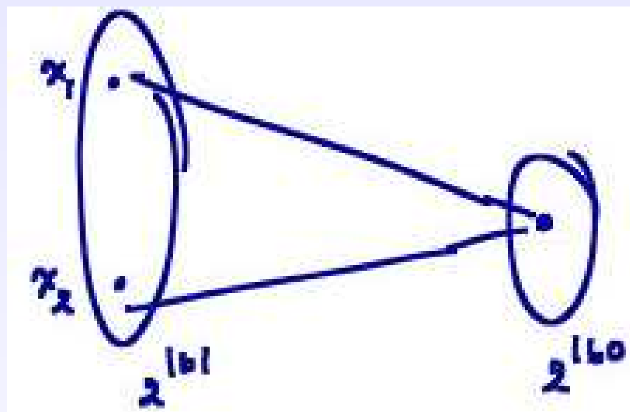
Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher.
Let $K \in \{0,1\}^k$.

Define a hash function $H : \{0,1\}^{mn} \to \{0,1\}^n$ based on K and E for any positive integer m using the raw CBC MAC construction.

Is H collision resistant?

Collision-finding attacks: Example

Example: Suppose $|\mathcal{D}| = 2|\mathcal{R}|$



Collision-finding attack: Example (cont.) – strategy #1

Strategy # 1

Pick a point, go through all elements in the domain in some order until collide.

Worst case :

- the last one is the one and
- evenly distributed.
  [i.e. only 2 points collide for each element in $\mathcal{R}$, i.e. $\max_{y \in \mathcal{R}} |H_K^{-1}(y)| = 2$]

number of trials needed $= 2^{161}$

Collision-finding attack: Example (cont.) – strategy #2

Strategy # 2

Pick a point, pick another point at random from $\mathcal{D}$ until collide.

Find collision with probability about 1 in $|\mathcal{R}|$. So,

$$\text{number of trials needed} = 2^{160}$$

Collision-finding attack: Example (cont.) – strategy #3

Strategy # 3: Birthday attack

Pick random points from $\mathcal{D}$ until find a pair that collides.

$$\text{number of trials needed} = O(\sqrt{|\mathcal{R}|}) = O(2^{80}) < 2^{160}$$

Birthday paradox

Let $x_1, \ldots, x_n \in \mathcal{D}$ be independent identically distributed elements of $\mathcal{D}$.

Birthday bound

If $n = 1.2 \times |\mathcal{D}|^{1/2}$, then

$$\Pr[\,\exists i \neq j \;:\; x_i = x_j\,] \geq \frac{1}{2}\,.$$

Proof of birthday bound (for uniform independent x's)

Denote $|\mathcal{D}|$ by D. Let E_i be the event that there is no collision after having picked i values. Let n be the total number of values that we pick.

$$\begin{aligned}
\Pr[\,\exists i \neq j \,:\, x_i = x_j\,] &= 1 - \Pr[\,\forall i \neq j \,:\, x_i \neq x_j\,] \\
&= 1 - \Pr[\,E_n\,] \\
&= 1 - \Pr[\,E_n \mid E_{n-1}\,] \cdot \Pr[\,E_{n-1}\,] \\
&= 1 - \left(\frac{D-1}{D}\right)\left(\frac{D-2}{D}\right)\cdots\left(\frac{D-n+1}{D}\right) \\
&= 1 - \prod_{i=1}^{n-1}\left(1 - \frac{i}{D}\right) \\
&\geq 1 - \prod_{i=1}^{n-1} e^{-\frac{i}{D}} \qquad \text{[because } 1 - x \leq e^{-x}\text{]} \\
&= 1 - e^{-\frac{1}{D}\sum_{i=1}^{n-1} i} \geq 1 - e^{-\frac{n^2}{2D}} \geq 1 - e^{-0.72} = 0.53
\end{aligned}$$

So we get this bound when $n^2/2D = 0.72$, i.e., $n = 1.2 \times |\mathcal{D}|^{1/2}$.