

CRYPTO - HASH

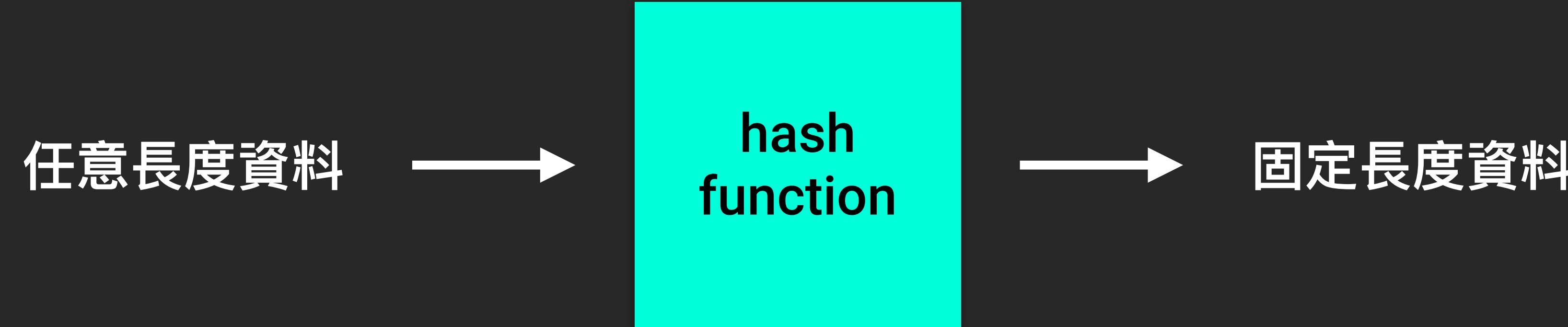
oalieno



目錄

- 1. Introduction to Hash**
- 2. Merkle–Damgård Construction**
- 3. Rainbow Table**
- 4. MD5 Collision**
- 5. SHA1 Collision**
- 6. Length Extension Attack**

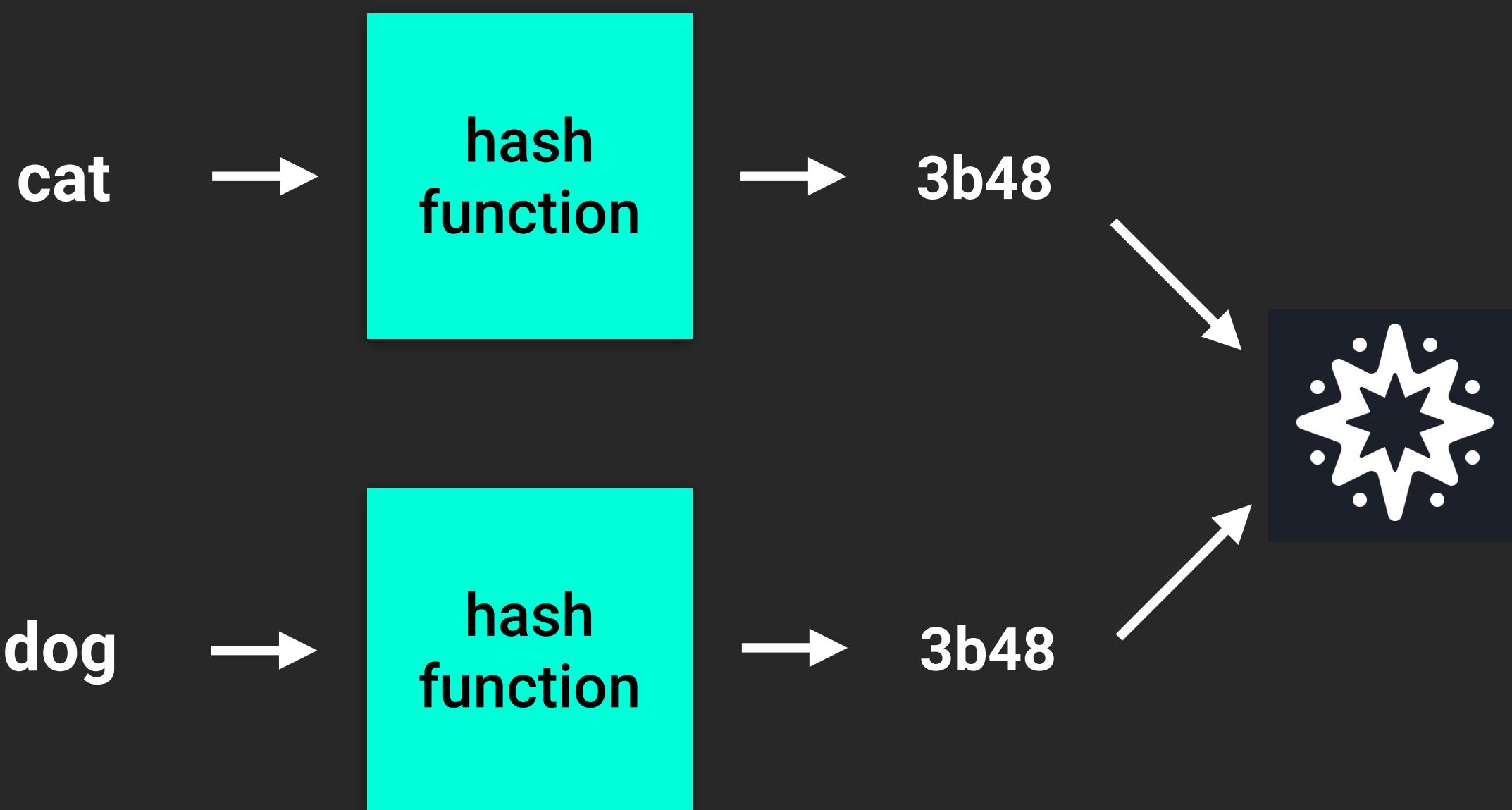
Introduction to Hash



特性：單向函式，不可逆

碰撞

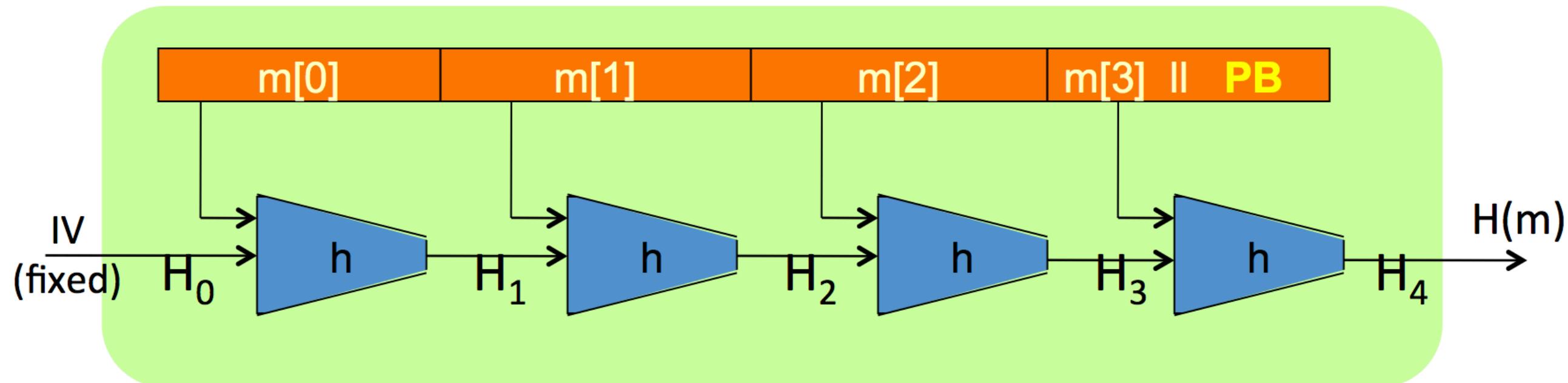
- 碰撞 (Collision)：
hash function 是將無限大小的集合映射到有限大小的集合
根據鴿籠原理，產生夠多 hash value 後一定會有重複的
也就是碰撞 (Collision)



Merkle–Damgård Construction

<https://www.youtube.com/watch?v=sawkPgsQPwg>

The Merkle-Damgard iterated construction



Given $h: T \times X \rightarrow T$ (compression function)

we obtain $H: X^{\leq L} \rightarrow T$. H_i - chaining variables

PB: padding block

1000...0 || msg len

64 bits

If no space for PB
add another block

Dan Boneh

許多的 hash function 是用這種構造方式 (ex: md5, sha1, sha256, ...)

Rainbow Table - 彩虹表的前身

- 字典攻擊是預先儲存一個很大的資料庫
- 裡面包含許多字串的 hash value
- 透過查表的方式找回 hash key
- **用空間換取時間**

aaa	47bce5c7
aab	e62595ee
aac	a9ced3da
aad	c2f7ab46
...	...

Rainbow Table - 建立彩虹表

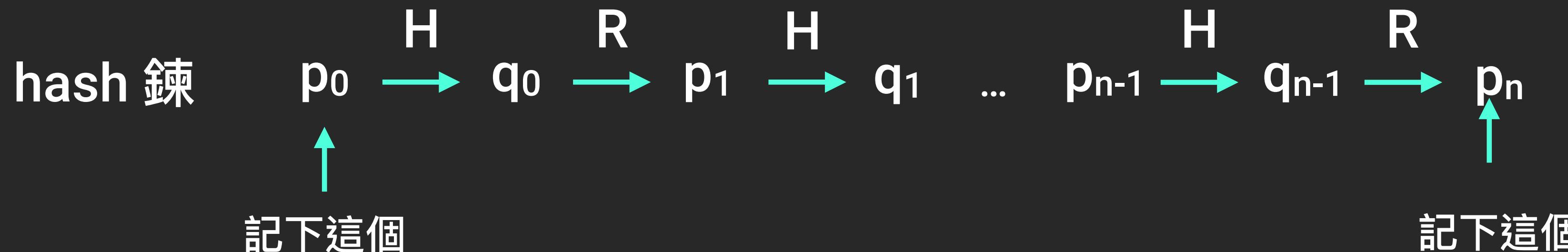
彩虹表和字典攻擊一樣是用空間換取時間，但彩虹表多犧牲了一點時間來換去更小的空間

我們有一個 hash function H ，輸入任意長度的資料，輸出 n bits 的資料

定義另一個 function R ，輸入 n bits 的資料，輸出 m bits 的資料

挑選許多初始值 p_0

計算 $\forall 1 \leq i \leq n : q_{i-1} = H(p_{i-1}), p_i = R(q_{i-1})$ 後將 (p_0, p_n) 存入資料庫



Rainbow Table - 查找彩虹表

給 Q 我們要找 P 使得 $H(P) = Q$

那我們就找資料庫裡面有沒有 $p_n = R(Q)$ ，有的話 $P = p_{n-1}$ ，因為 $H(p_{n-1}) = Q$

沒有的話就繼續找資料庫裡面有沒有 $p_n = R(H(R(Q)))$ ，有的話 $P = p_{n-2}$ ，因為 $H(p_{n-2}) = Q$

然後就這樣掃過 hash chain

工具

- [ophcrack](#)
- [rainbowcrack](#)
- [rtgen](#)

MD5 Collision - 碰撞實測

<https://www.mathstat.dal.ca/~selinger/md5collision/>

將兩串不同的 data 寫入檔案

```
$ echo d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f8955ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbdf280373c5bd8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70 | xxd -r -p > A  
$ echo d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f8955ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5bd8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70 | xxd -r -p > B
```

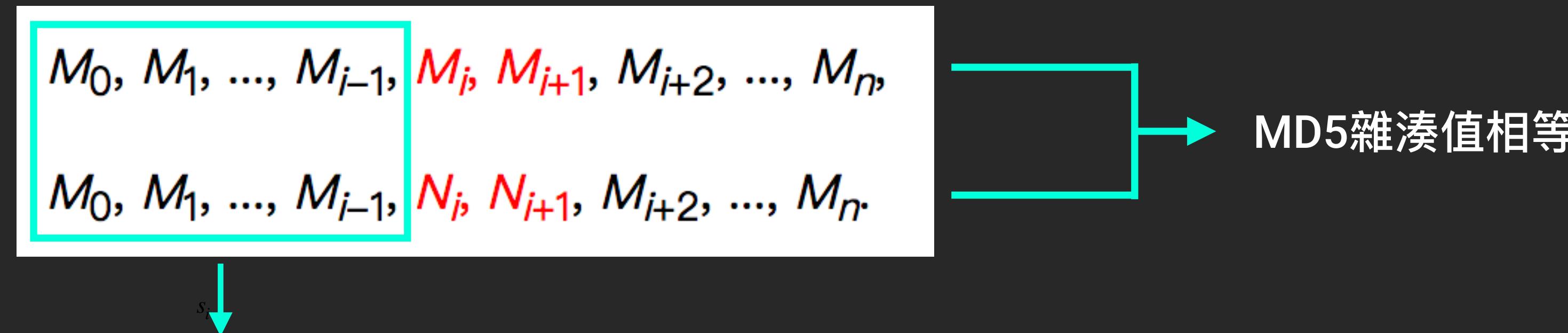
用 md5sum 看看他的 hash value

```
~/toolbox/md5-collision # md5sum A B  
79054025255fb1a26e4bc422aef54eb4 A  
79054025255fb1a26e4bc422aef54eb4 B
```

MD5 Collision - 碰撞應用

<https://www.mathstat.dal.ca/~selinger/md5collision/>

給任意的初始值 s_i 用 method of Wang and Yu 可以找到 M_i, M_{i+1} 和 N_i, N_{i+1} 使得

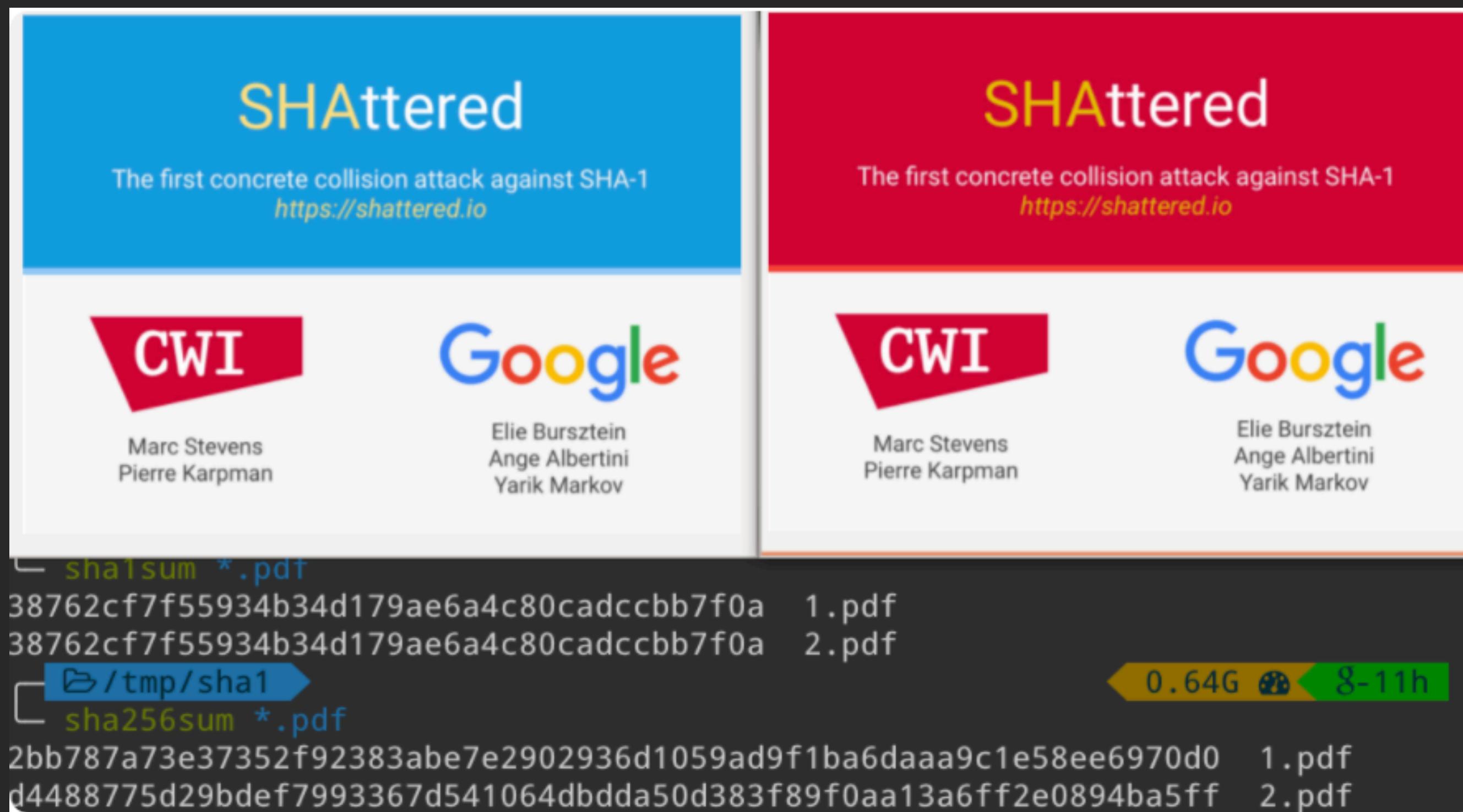


用這種方法可以構造出下面這樣的程式 (他們的 MD5 相等)

```
Program 1: if (data1 == data1) then { good_program } else { evil_program }
Program 2: if (data2 == data1) then { good_program } else { evil_program }
```

SHA1 Collision

Google 在 2017 年 2 月 23 號
發表了兩份有一模一樣的 SHA1 checksum 的 PDF



<https://shattered.io/>

SHA1 Collision

Boston Key Party CTF 2017 在 2017 年 2 月 25 號

馬上出了一題 SHA1 collision 的題目

Boston Key Party CTF 2017 - prudentialv2

要讓 name != password 但是 sha1(name) == sha1(password)

直接把 google 找到的那兩個 PDF 傳過去就對了 XD

```
if ($_GET['name'] == $_GET['password'])
    print 'Your password can not be your name.';
else if (sha1($_GET['name']) === sha1($_GET['password']))
    die('Flag: '.$flag);
```

SHA1 Collision

Seccon CTF 2017 - SHA-1 is dead 也是 SHA1 collision

上傳兩個檔案滿足：

1. file1 != file2
2. SHA1(file1) == SHA1(file2)
3. 2017 KiB < sizeof(file1) < 2018 KiB
4. 2017 KiB < sizeof(file2) < 2018 KiB

KiB = 1024 bytes
KB = 1000 bytes

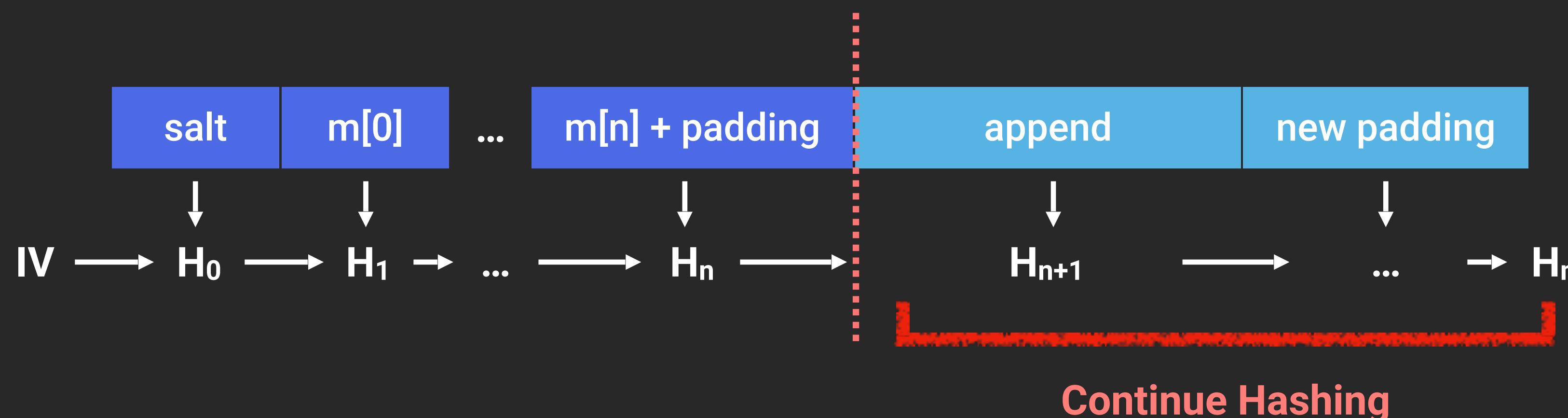
```
$ python3 -c "print('A' * (2017 * 1024 - 422435 + 1), end = "")" >> shattered-1.pdf
$ python3 -c "print('A' * (2017 * 1024 - 422435 + 1), end = "")" >> shattered-2.pdf
```

Length Extension Attack

- 當我們的 Message Authentication 是 $H(\text{salt} \parallel \text{message})$ 的形式
- 而我們的 hash function H 是 Merkle–Damgård Construction
- 我們就可以繞過 MAC 的驗證機制，對明文串接部分可控資料

Length Extension Attack

- 在改動明文後重新計算 hash
- salt 已經在前面被計算過了，完全不用理他
- 新的明文包含 message || padding || append



HashPump

<https://github.com/bwall/HashPump>

不用自己指定要用哪個 Hash Function
預設會自動根據 $H(salt \parallel message)$ 這個 hash 的長度判斷

CTF

- TUM CTF Teaser - bad_apple
- RuCTF Quals 2014 - MD5 lext
- Teaser CONFidence CTF 2015 - Mac hacking
- BAMBOOFOX CTF 2018 - baby-lea