Cryptographic Hash Functions

CSIE 7190 Cryptography and Network Security, Spring 2019

https://ceiba.ntu.edu.tw/1072csie_cns

cns@csie.ntu.edu.tw

Hsu-Chun Hsiao

National Taiwan University

Housekeeping

3/12 (or later): HW1 out

3/12 2pm: Reading critique #3 due

4/9: 繳交期末報告分組和題目

- 4-5人一組
- 可以先列大致的主題跟方向與我們討論

Reading critique #3

Write a critique on one of the following:

1. A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," in RSA CryptoBytes, 2005.
2. Laurent Eschenauer and Virgil D. Gligor. "A key-management scheme for distributed sensor networks," in ACM CCS, 2002.

Text only, one page

2019/02/19, 國際

中國臉部辨識公司遭爆資料外洩，新疆逾250萬居民動態全都露

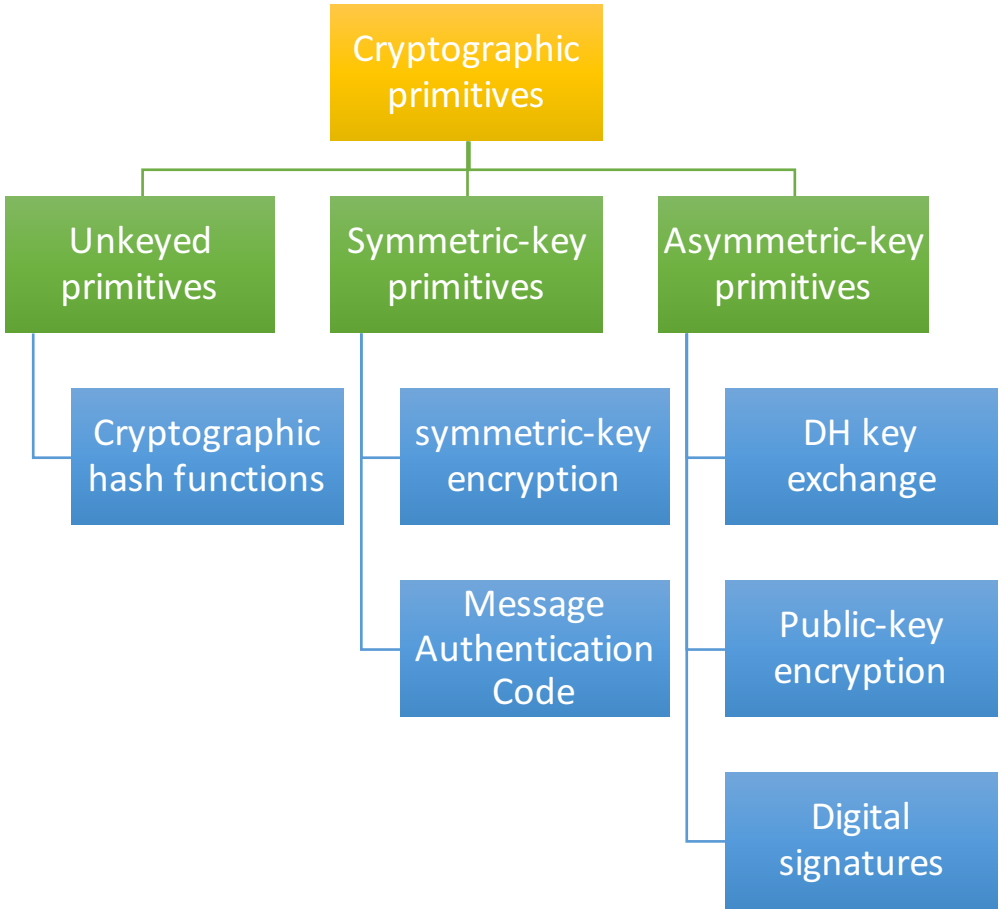
https://www.thenewslens.com/article/113972

Abelson, Harold, et al. "Keys under doormats: mandating insecurity by requiring government access to all data and communications." Journal of Cybersecurity 1.1 (2015): 69-79.

Agenda

Cryptographic hash functions

- Requirements
- Applications
- Attacks



* what we will cover in this course; not a complete list

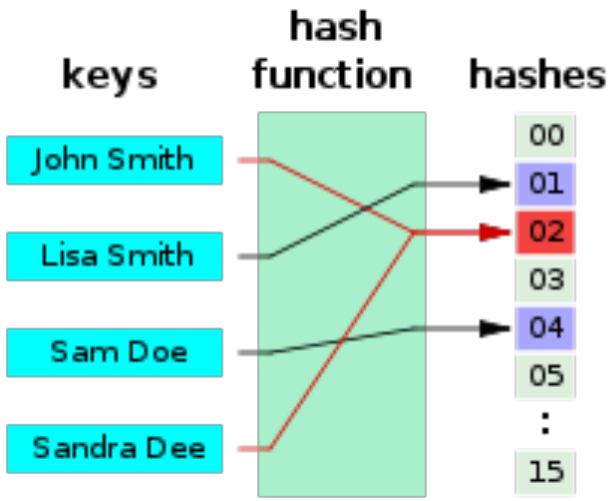
Hash functions

An efficiently computable, public function: $H\colon \mathcal{M} \to \mathcal{T}$

Maps **arbitrary**-length input to **finite** length output

- $y = H(x)$, y is the *hash* of x, and x is a *preimage* of y
- If $H(x') = H(x)$ and $x' \neq x$, then this is a *collision*

(Non-cryptographic) hash function vs. cryptographic hash function



(Non-cryptographic) hash functions

E.g., the hash functions used in most of the hash table implementations

Collisions are common and easy to find in hash tables

- Small table size
- table resized when load factor > threshold
- Collision resolution: chaining, open addressing

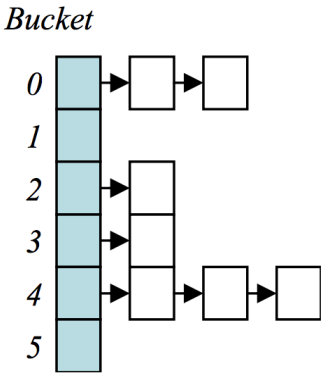


Figure 1: Normal operation of a hash table.

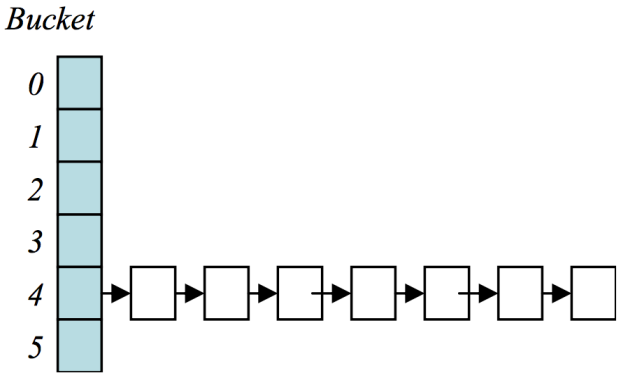


Figure 2: Worst-case hash table collisions.

oCERT Advisories

#2011-003 multiple implementations denial-of-service via hash algorithm collision

Description:

A variety of programming languages suffer from a denial-of-service (DoS) condition against storage functions of key/value pairs in hash data structures, the condition can be leveraged by exploiting predictable collisions in the underlying hashing algorithms.

The issue finds particular exposure in web server applications and/or frameworks. In particular, the lack of sufficient limits for the number of parameters in POST requests in conjunction with the predictable collision properties in the hashing functions of the underlying languages can render web applications vulnerable to the DoS condition. The attacker, using specially crafted HTTP requests, can lead to a 100% of CPU usage which can last up to several hours depending on the targeted application and server performance, the amplification effect is considerable and requires little bandwidth and time on the attacker side.

The condition for predictable collisions in the hashing functions has been reported for the following language implementations: Java, JRuby, PHP, Python, Rubinius, Ruby. In the case of the Ruby language, the 1.9.x branch is not affected by the predictable collision condition since this version includes a randomization of the hashing function.

The vulnerability outlined in this advisory is practically identical to the one reported in 2003 and described in the paper Denial of Service via Algorithmic Complexity Attacks which affected the Perl language.

The reporters own advisory can be found at http://www.nruns.com/_downloads/advisory28122011.pdf

S. A. Crosby and D. S. Wallach, "Denial of Service via Algorithmic Complexity Attacks," in *Proceedings of USENIX Security*, 2003.
http://www.ocert.org/advisories/ocert-2011-003.html

Severe collision may lead to denial of service

Hash table collision in Python

```
n = 20000
d=2**64-1
h={}
for i in xrange(n):
  h[i * d] = i
```

Similar attacks work in PHP, ASP, Python, Ruby, C++, Java...

我們不希望一直遇到碰撞，碰撞是不好的。

Cryptographic Hash Functions

One-wayness

Weak collision resistance

Strong collision resistance

Cryptographic hash functions

An efficiently computable, public function: $H: \mathcal{M} \rightarrow \mathcal{T}$

Maps **arbitrary**-length input to **finite** length output

- $y = H(x)$, y is the *hash* of x, and x is a *preimage* of y
- If $H(x') = H(x)$ and $x' \neq x$, then this is a *collision*

Properties of a secure hash function, H

1. One-wayness (preimage resistance): 很難找到他的 preimage
   Given y, hard to find x s.t. $y = H(x)$ hard 指的是，computational infeasible
2. Weak collision resistance (second preimage resistance):
   Given x, hard to find $x' \neq x$ s.t. $H(x') = H(x)$
3. Strong collision resistance (collision resistance):
   Hard to find x and x' s.t. $x' \neq x$ and $H(x') = H(x)$

2 跟 3 差別，3 沒有前提。
3 最難達成，e.g. 生日攻擊：一個班上，兩個人同生日的機率很高。

*Formally, we should say “computationally infeasible” instead of “hard”.

Collisions are unavoidable

無限多種可能的input -> 固定長度output

碰撞一定存在

只是很難找到（若hash function設計得好的話）

Cryptographic hash functions

Well-known algorithms: MD5, SHA1, SHA256, SHA3, …

- Some of them are already broken
- No proof of security for the rest either

Note that hash functions are public and deterministic

SHA256("ntu") =
`2de2a32d6fe8f588139f503673d6f951204993186cf2f9af5701408452a2f46c`

SHA256("ntU") =
`68c11a8758633d665d6d3d4534e0ec42fe16997 4862f9336b11091e5e54f219b`

SHA256("csie department, national taiwan university") =
`e3bdfe4d39eb6bf51a0bca7de03d88fcb7ab557d51869c8989ddcd57588260ba`

http://www.hashgenerator.de/

One-wayness

Given y, hard to find x s.t. $y = H(x)$

Example

SHA1(x) = `eb25ae311c1e3b88e6eb967749e9aca5632ace26`

What is x?

How hard to find x?

Attack complexity: One-wayness

Assume a secure hash function with n-bit output

Assume $\Pr[H(x) = y] = 2^{-n}$ for all x, y

Assume the best attack is *exhaustive search* 亂猜

Given output y, how many hash operations does it take to find an x, such that $H(x) = y$?

- For each trial x, probability that it outputs y is 2^{-n}
- Pr[find x after m trials] = $1 - (1 - 2^{-n})^m$
- Rule of thumb: find x after 2^{n-1} trials on average

Weak collision resistance

Given x, hard to find $x' \neq x$ s.t. $H(x') = H(x)$

Example

x = ntu-csie

SHA1(x) = `6d20c2af339fc21e399edfa07f0fadb02f8a2b70`

Can you find $x' \neq x$ such that SHA1(x) = SHA1(x')?

How hard to find x'?

Attack complexity: Weak collision resistance

Assume a secure hash function with n-bit output

Assume $\Pr[H(x) = y] = 2^{-n}$ for all x, y

Assume the best attack is *exhaustive search*

Given input x, how many operations does it take to find another $x' \neq x$ s.t. $H(x') = H(x)$?

- For each trial x', probability that output is equal is 2^{-n}
- Pr[find x after m trials] = $1 - (1 - 2^{-n})^m$
- Rule of thumb: find x after 2^{n-1} trials on average

Strong collision resistance

Hard to find x and x' s.t. $x' \neq x$ and $H(x') = H(x)$

How hard to find one pair of x and x'?

Attack complexity: Strong collision resistance

Assume a secure hash function with n-bit output

Assume $\Pr[H(x) = y] = 2^{-n}$ for all x, y

Assume the best attack is *exhaustive search*

How many operations does it take to find x and x' s.t. $x' \neq x$ and $H(x') = H(x)$?

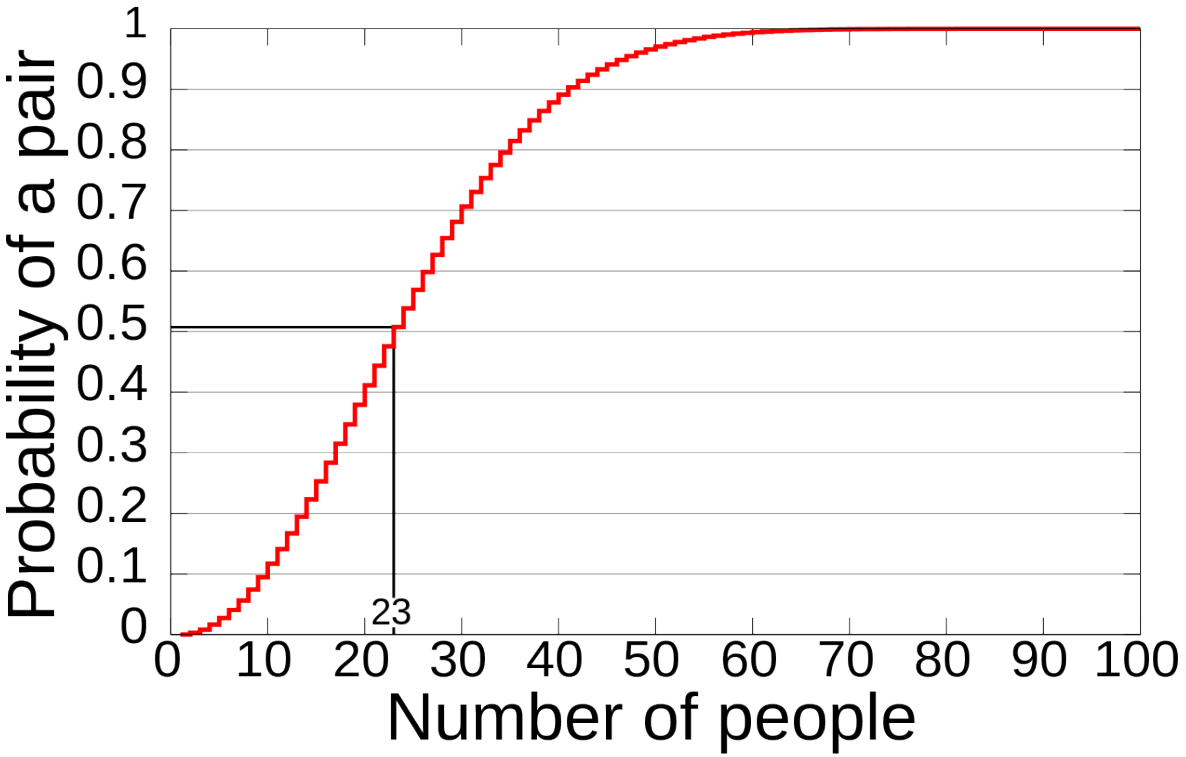
- Suppose we pick a random x, checks whether $H(x)$ matches any output value previously seen
- P[find collision after m trials]= $1 - (1 - 1/2^n)(1 - 2/2^n)(1 - 3/2^n) \ldots (1 - (m-1)/2^n)$
- Rule of thumb: find a collision after $2^{n/2}$ trials on average
  - ($1.17 * 2^{n/2}$ to be more precise)
  - Useful trick: for x << 1, $e^x \approx 1 + x$

試了 2^{th} 沒找到
= 3rd =

Birthday paradox

教室裡要有多少人才能使得至少有兩個人生日相同的機率 > 50%?

Answer: approximately $1.17*365^{1/2} \approx 22.4$



http://en.wikipedia.org/wiki/Birthday_problem

md5 collision example

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70
```

md5 collision example

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b6708280d1ec69821bcb6a8839396f965ab6ff72a70
```

Both have a md5 hash `79054025255fb1a26e4bc422aef54eb4`

Applications of Hash Functions

Applications

Integrity check

Message digest for signing

Commitment

Password hashing

挖礦 Proof of Work

Application: integrity check – file download

Windows 8 Enterprise 2012/11/30

Windows 8 Enterprise
無須序號 請參考認證說明網頁

(建議使用分流區下載)
MD5驗證碼
32bit 6578F2342974D3945845316E5C6DA62B
64bit 371F8DA6BB6A4B3701BE48143CCA1E96

Windows 8 32 bit	Filesize: 2.5G	download 下載
Windows 8 64 bit	Filesize: 3.4G	download 下載
Windows	Filesize: 2.4G	download 下載

這個應用需要hash function的哪些性質？

One-wayness: Given y = H(x), hard to find x' s.t. H(x') = y

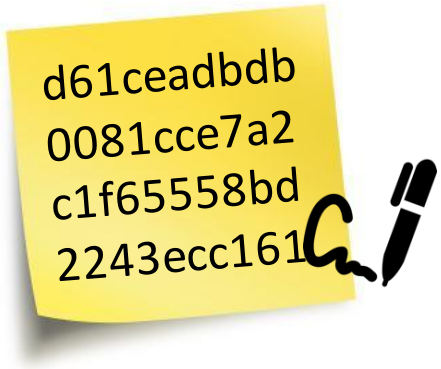
✓ Weak collision resistance: Given x, hard to find x' ≠ x s.t. H(x) = H(x')

Strong collision resistance: Hard to find x, x' s.t. x' ≠ x and H(x) = H(x')

若有另一個 file x'，H(x') = H(x)，如此無法透過 H(x) 得知檔案是否正確。

Application: Message digest for signing

Signature is usually signed over the hash of the message instead of the message itself: $\text{Sign}_{sk}(H(m))$



這個應用需要hash function的哪些性質？

One-wayness: Given y = H(x), hard to find x' s.t. H(x') = y

✓ Weak collision resistance: Given x, hard to find x' ≠ x s.t. H(x) = H(x')

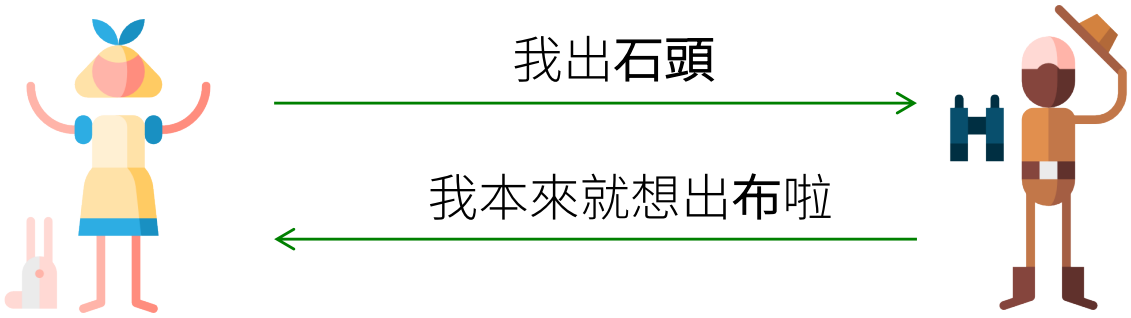
sign錯份

✓ Strong collision resistance: Hard to find x, x' s.t. x' ≠ x and H(x) = H(x')

取決於數位簽章要應用在什麼情境下。

Application: Commitment

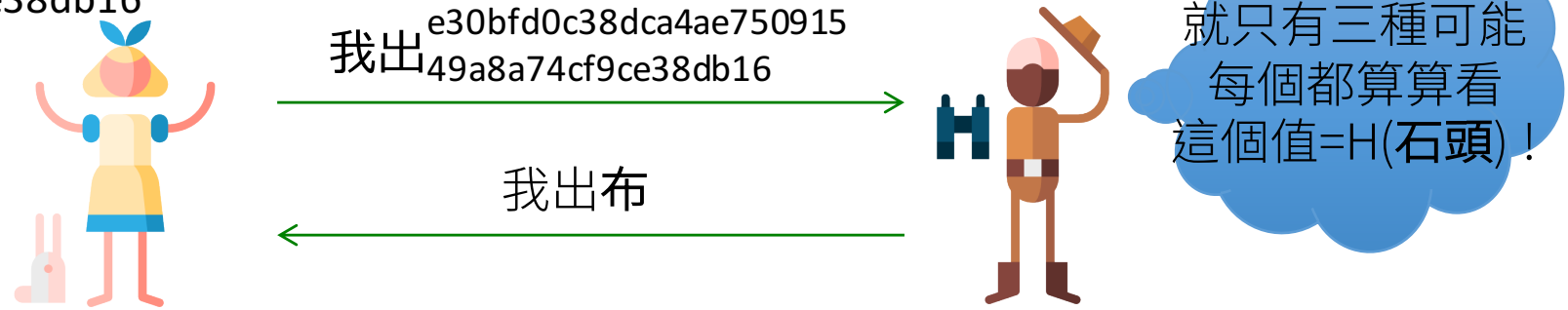
剪刀石頭布 online
假設通訊是安全的，但Alice和Bob彼此不信任對方。
在不仰賴公正第三方的前提之下，要如何維持遊戲的公平性？



剪刀石頭布online v2

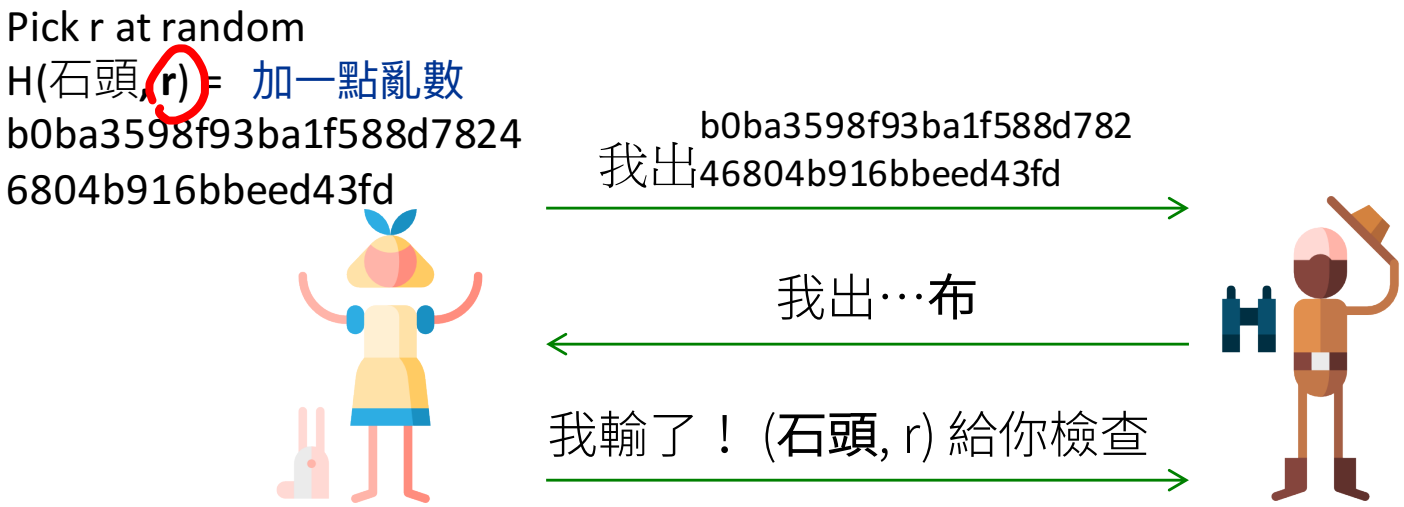
假設通訊是安全的，但Alice和Bob彼此不信任對方。
在不仰賴公正第三方的前提之下，要如何維持遊戲的公平性？

H(石頭) =
e30bfd0c38dca4ae750915
49a8a74cf9ce38db16



剪刀石頭布online v3

假設通訊是安全的，但Alice和Bob彼此不信任對方。
在不仰賴公正第三方的前提之下，要如何維持遊戲的公平性？



若 Alice 可以找到一 collision：(剪刀, r') = (石頭, r) 就可騙過 Bob

Bob 猜不出 Alice 出了什麼
Alice無法根據Bob的回答再改動自己的答案

剪刀石頭布online v3

這個應用需要hash function的哪些性質？

- ✓ One-wayness: Given y = H(x), hard to find x' s.t. H(x') = y
- ✓ Weak collision resistance: Given x, hard to find x' ≠ x s.t. H(x) = H(x')
- ✓ Strong collision resistance: Hard to find x, x' s.t. x' ≠ x and H(x) = H(x')

One-wayness：Bob 不能看到 HASH 就猜出是石頭。
Weak collision resistance：若 Alice 可以找到一 collision：(剪刀, r') = (石頭, r) 就可騙過 Bob。
Strong collision resistance：若存在一組 (剪刀, r') = (石頭, r) 就可魚目混珠。

Application: Password hashing

Why hashing passwords? 即便外洩，也是外洩 HASH。

- Threat model: leaked password database
- If passwords are stored in their hash forms (rather than in plaintext), then it's harder for the the attacker to recover the passwords in case of data leak.

Hack of Cupid Media dating website exposes 42 million *plaintext* passwords

Sony hacked yet again, plaintext passwords, e-mails, DOB posted

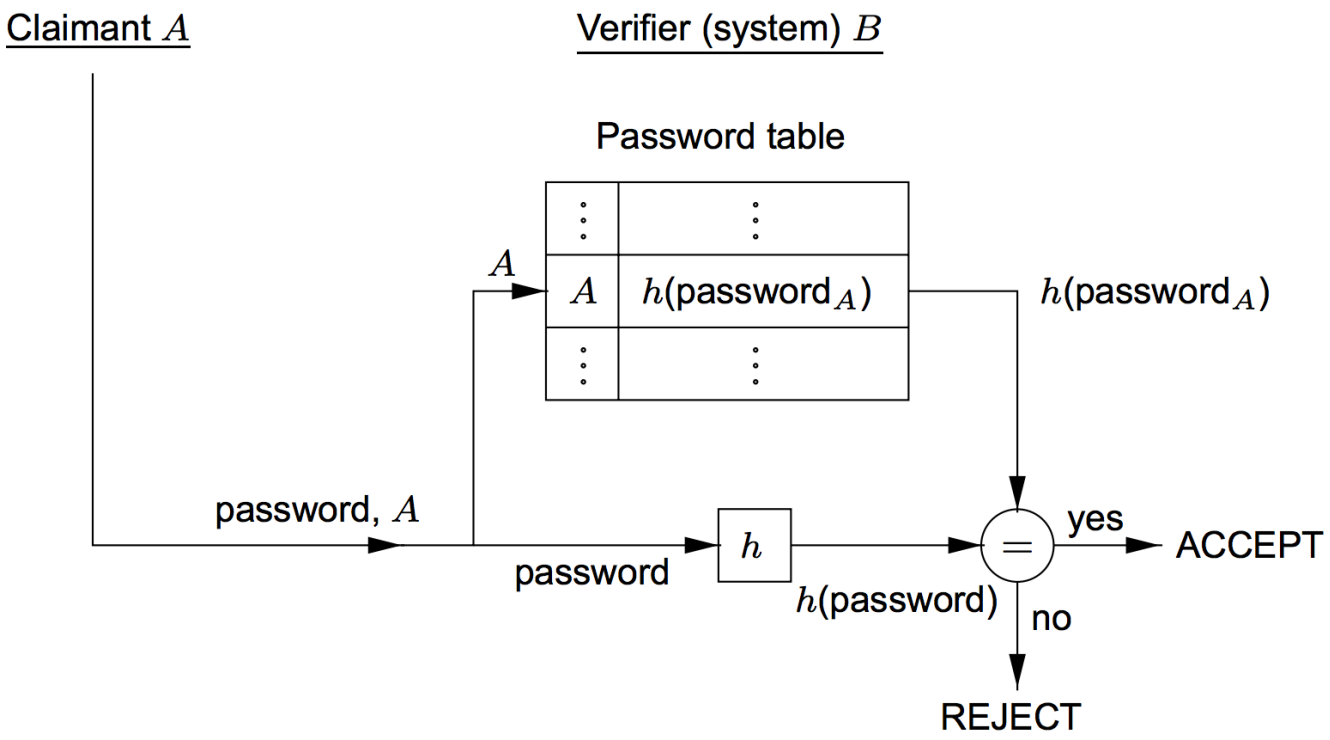
More passwords: https://github.com/danielmiessler/SecLists/tree/master/Passwords

A. Juels and R. L. Rivest, "Honeywords: Making Password-Cracking Detectable," in *Proceedings of ACM CCS*, 2013.

Password hashing

Store $H(P)$ instead of plaintext password P

Is this enough to protect leaked passwords?



http://cacr.uwaterloo.ca/hac/about/chap10.pdf

Exhaustive Search vs. Dictionary Attack

Exhaustive Search

Trying aaaa : failed
Trying aaab : failed
Trying aaac : failed
...
Trying acdb : failed
Trying acdc : success!

Dictionary Attack

Trying apple : failed
Trying blueberry : failed
Trying justinbeiber : failed
...
Trying letmein : failed
Trying s3cr3t : success!

https://crackstation.net/hashing-security.htm

Salting and stretching

Store $\{salt, H^t(salt, P)\}$ instead of $H(P)$

- H^t means the hash function is applied t times
- t is at the order of 1000 倍
- Standard: PBKDF2 (Password-Based Key Derivation Function 2)
- Why do these help?

就算 2 人 password 一樣，salt 不同，hacker 就無法馬上查覺。

Salting: adding random values such that two accounts with the same password generate different hashes

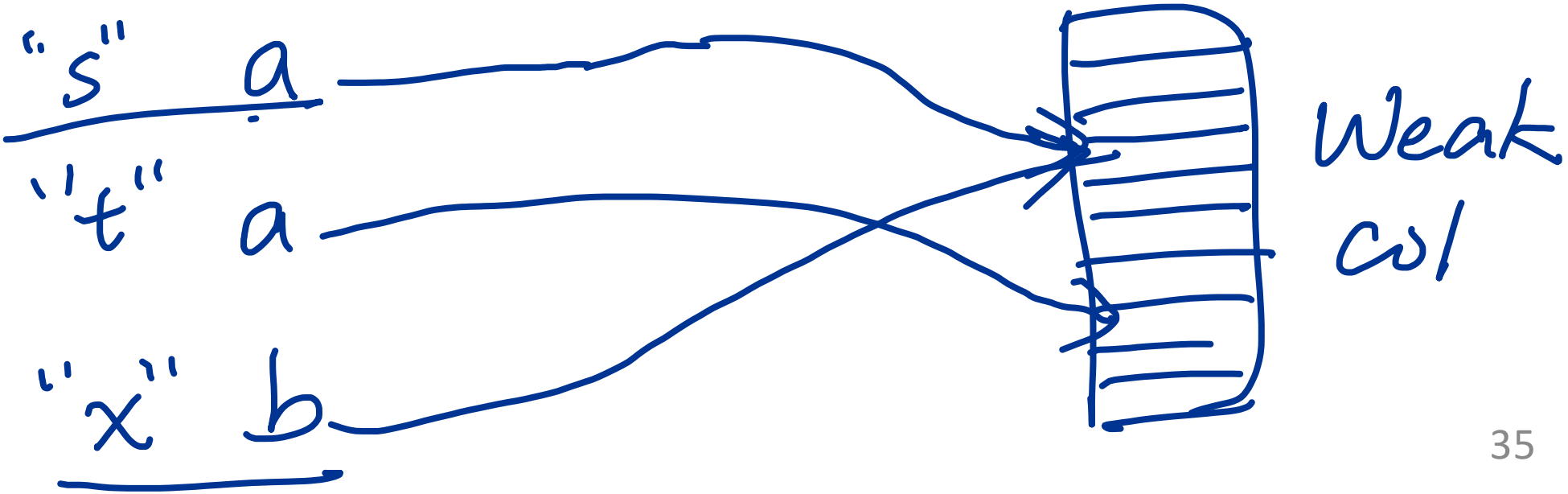
- Prevent the use of pre-computed tables (rainbow tables)

Stretching: deliberately slow down offline password cracking

Password hashing

這個應用需要hash function的哪些性質？

- ✓ One-wayness: Given y = H(x), hard to find x' s.t. H(x') = y
- ✗ Weak collision resistance: Given x, hard to find x' ≠ x s.t. H(x) = H(x')
- ✗ Strong collision resistance: Hard to find x, x' s.t. x' ≠ x and H(x) = H(x')



Password Hashing Competition

The advance of special hardware (e.g., GPU) significantly speeds up password cracking

The competition called for a key stretching standard that can resist such GPU-based cracking

Argon2 won the competition in 2015

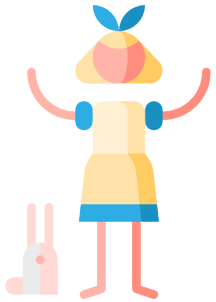
- "Argon2: the memory-hard function for password hashing and other applications"

https://password-hashing.net/

Application: Proof of Work (POW)

Alice (prover) proves to Bob (verifier) that she has performed a certain amount of computational work in a specified interval of time.

Can be used to slow down denial of service and achieve some sort of fairness in bitcoin mining



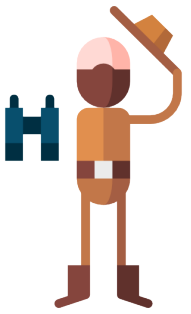
1. request

2. Pick random x, compute $y = [H(x)]_r$

3. H, y, r

4. Find x s.t. $[H(x)]_r = y$

5. H, y, r, x



Proof of Work (POW)

這個應用需要hash function的哪些性質？

✓ One-wayness: Given y = H(x), hard to find x' s.t. H(x') = y
如果馬上就找到 x' s.t. H(x') = y，那就沒有 Proof 到他是否有做一定時間的 work 了。
Weak collision resistance: Given x, hard to find x' ≠ x s.t. H(x) = H(x')

Strong collision resistance: Hard to find x, x' s.t. x' ≠ x and H(x) = H(x')

Hash Algorithms and Attacks

Merkle-Damgård construction (Boneh and Shoup, Ch. 8.4)

Length extension attack

MD5 and SHA-1 collision attack

Exercise: design your own hash function (and show that it's NOT secure)

X is a bit string

Let $H(x) = [x]_{0\ldots127} \oplus [x]_{128\ldots255} \oplus \ldots$

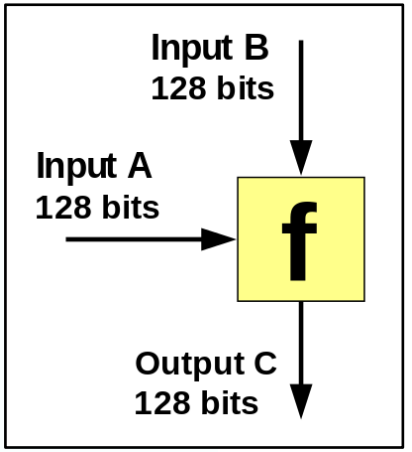
Can you break it w.r.t. if len(x) < 128 :

- One-wayness: Given y = H(x), hard to find x' s.t. H(x') = y
- Weak collision resistance: Given x, hard to find x' ≠ x s.t. H(x) = H(x')
- Strong collision resistance: Hard to find x, x' s.t. x' ≠ x and H(x) = H(x')

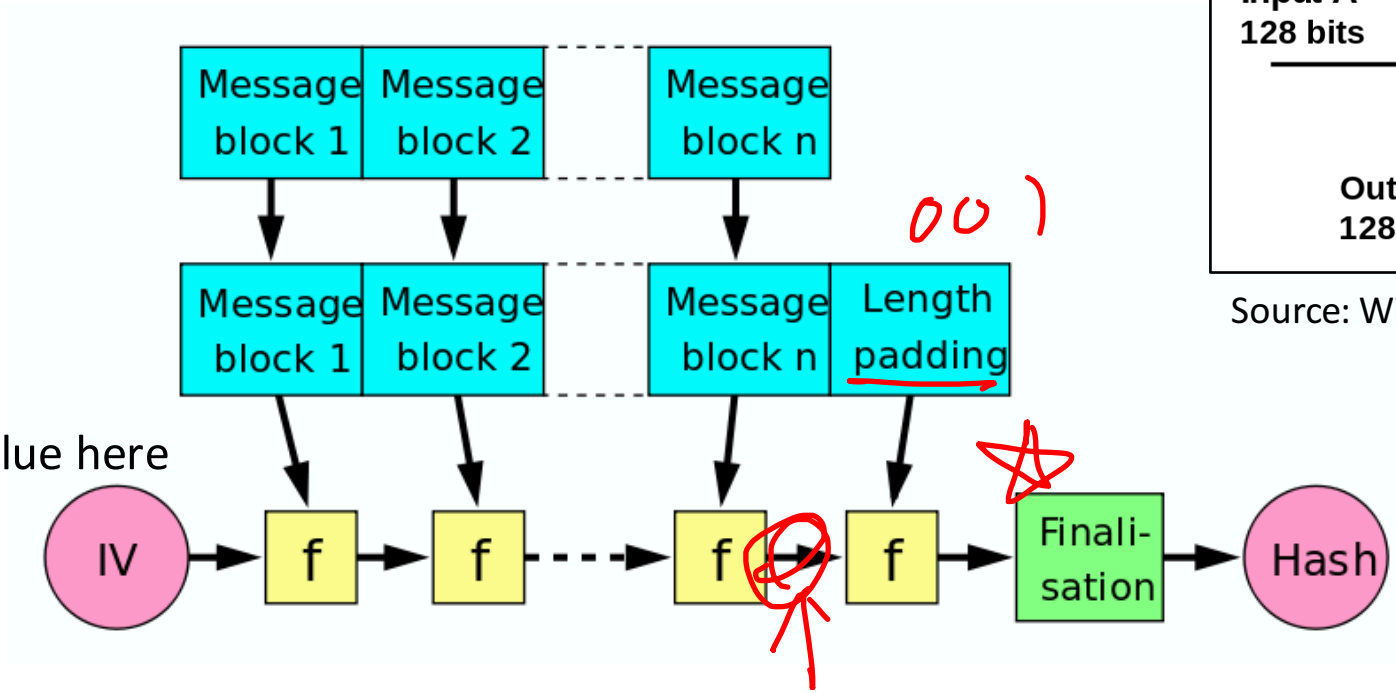
Merkle-Damgård construction

A common method to construct a hash function

- Assumes a one-way compression function, *f*, with **fixed-length input**
- Extends *f* to take an arbitrary-length input
- Used in MD5, SHA-1, SHA-2
- SHA-3 uses a different construction called sponge



Source: Wikipedia

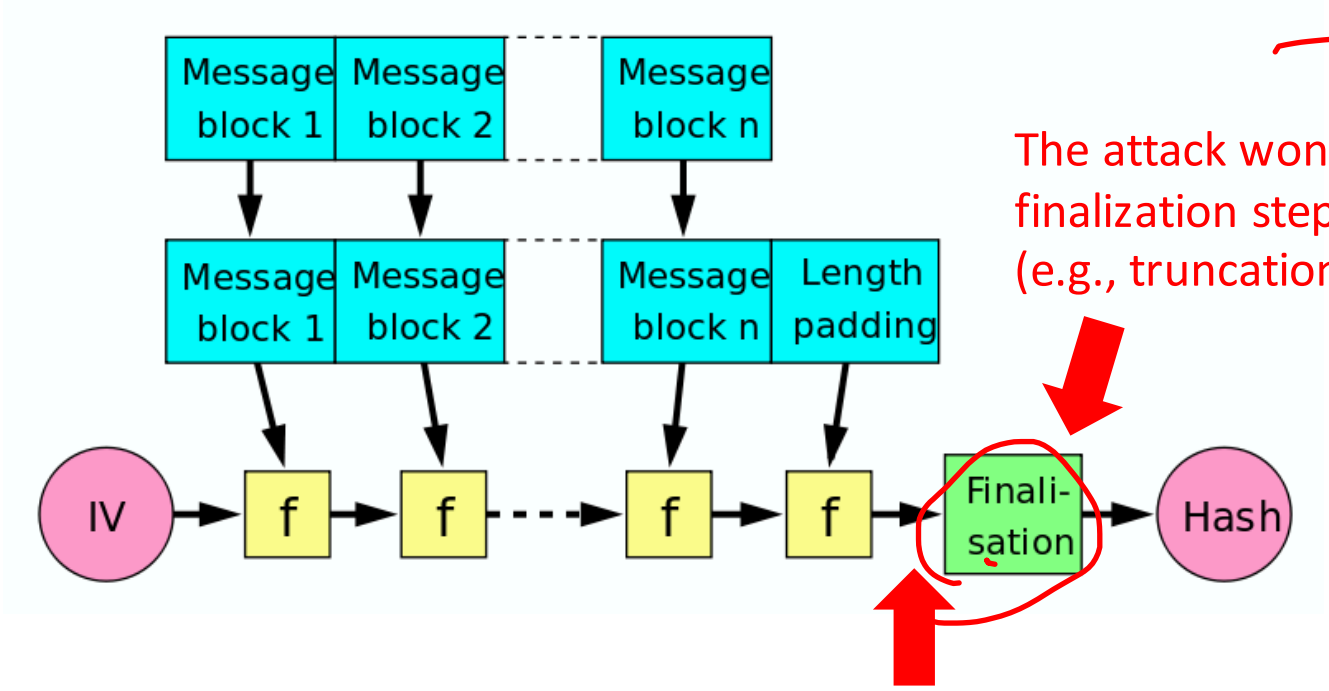


IV is a fixed value here

Attacks against Merkle-Damgård construction: Length extension attack

Given H(**secret**), can you compute H(**secret**||**ext**) for some **ext** without knowing **secret**?

- Possible for MD5, SHA-1, SHA-256/512
- SHA-224/384, SHA3 are not vulnerable



The attack won't work if the finalization step is irreversible (e.g., truncation)

All the attacker needs to know is this state!

Attacks against Merkle-Damgård construction: Length extension attack

Given H(**`secret`**), can you compute H(**`secret`**||**`ext`**) for some **`ext`** without knowing **`secret`**?

- Possible for MD5, SHA-1, SHA-256/512
- SHA-224/384, SHA3 are not vulnerable

Strictly speaking, this is not an attack against hash functions, because hash functions are designed to provide one-wayness and collision resistance, but not resistance to length extension.

- Be careful if you want to use hash function to construct other primitives, such as Message Authentication Codes.

Popular choices of hash algorithms

~~MD5~~

~~SHA-1~~

SHA-2 family (SHA-224, SHA-256, …)

SHA-3

MD5 and SHA-1 are considered broken

- There exist a better collision attack than the birthday attack
- No known practical pre-image attacks against MD5 or SHA-1 yet

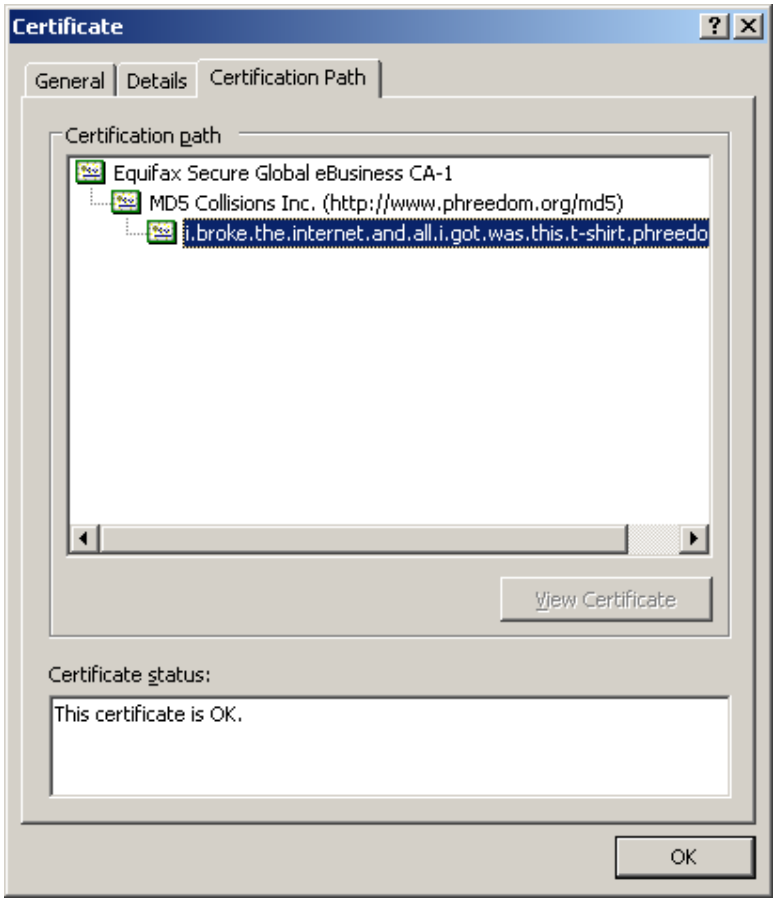
https://en.wikipedia.org/wiki/Hash_function_security_summary

MD5's collision-resistance is broken

MD5 collisions can be found easily (complexity ~2^{18})

Example of real-world exploits: create rogue certificates

- Digital signatures sign the message digest, not the message itself
- Register a certificate for www.something.com and use this certificate for www.bank.com if H(Cert of something.com) = H(Cert of bank.com)



MD5 considered harmful today. http://www.win.tue.nl/hashclash/rogue-ca/
http://www.md5.net/md5-cracker/

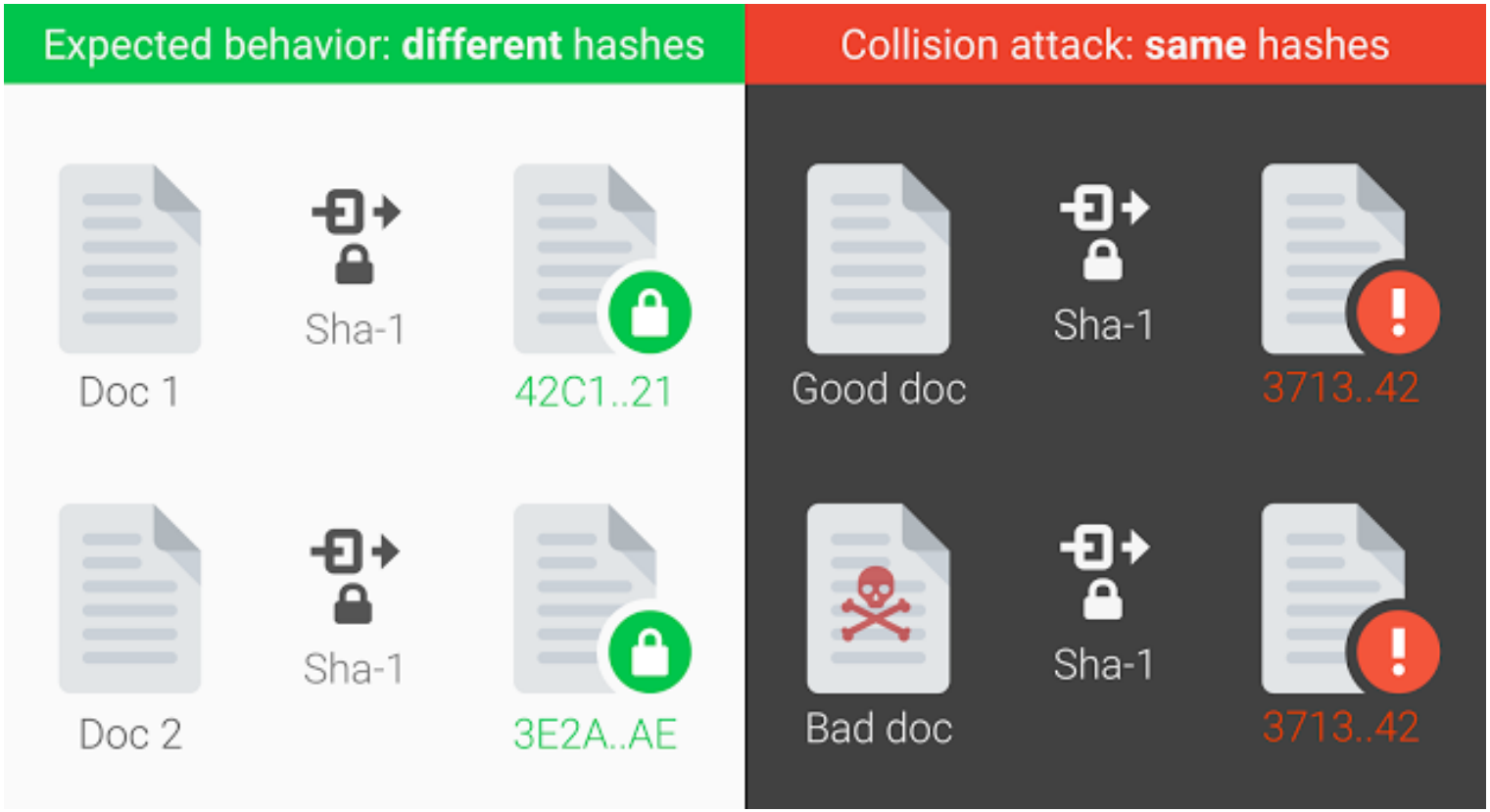
MD5's collision-resistance is broken

No known practical pre-image attacks against MD5 yet

- But it's recommended not to use MD5 anymore
- Large MD5 hash databases make it very easily to find MD5 pre-image of common phrases
  - Find x for MD5(x) = `e0d00b9f337d357c6faa2f8ceae4a60d`
  - Find x for MD5(x) = `d8578edf8458ce06fbc5bb76a58c5ca4`

MD5 considered harmful today. http://www.win.tue.nl/hashclash/rogue-ca/
https://crackstation.net/

SHA1's collision-resistance is broken



https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

Impact of hash collision

Digital certificates

Email PGP/GPG signatures

Software vendor signatures

Software updates

ISO checksums

Backup systems

Deduplication systems

GIT

Generate md5 collisions: https://marc-stevens.nl/p/hashclash/

Merkle Hash Tree

如何確保雲端資料的完整性？

Assumption: secure communication

Attacker model: cloud may return wrong or tampered data

How can Alice ensure the integrity of her data?

D_0

D_1

D_2

Get file, id = 1

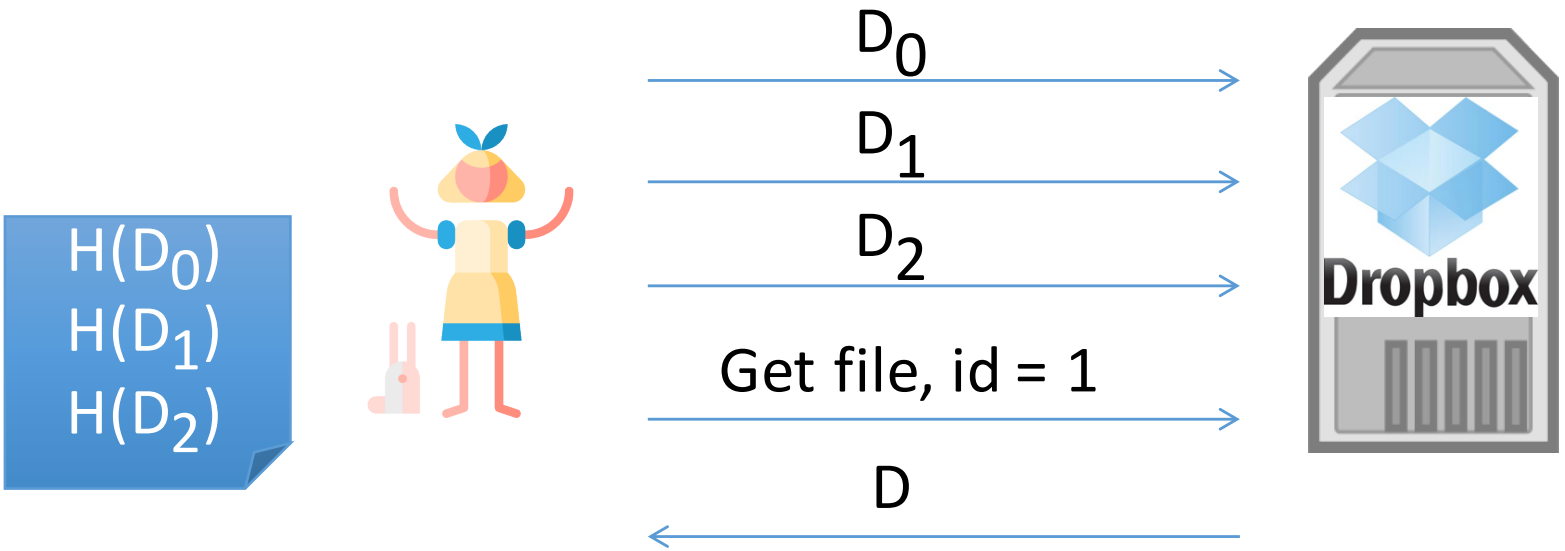
D

Is $D = D_1$?

如何確保雲端資料的完整性？

Approach #1: keep file hashes

Disadvantages: storage overhead at client, Alice needs to copy this large hash file when using another computer



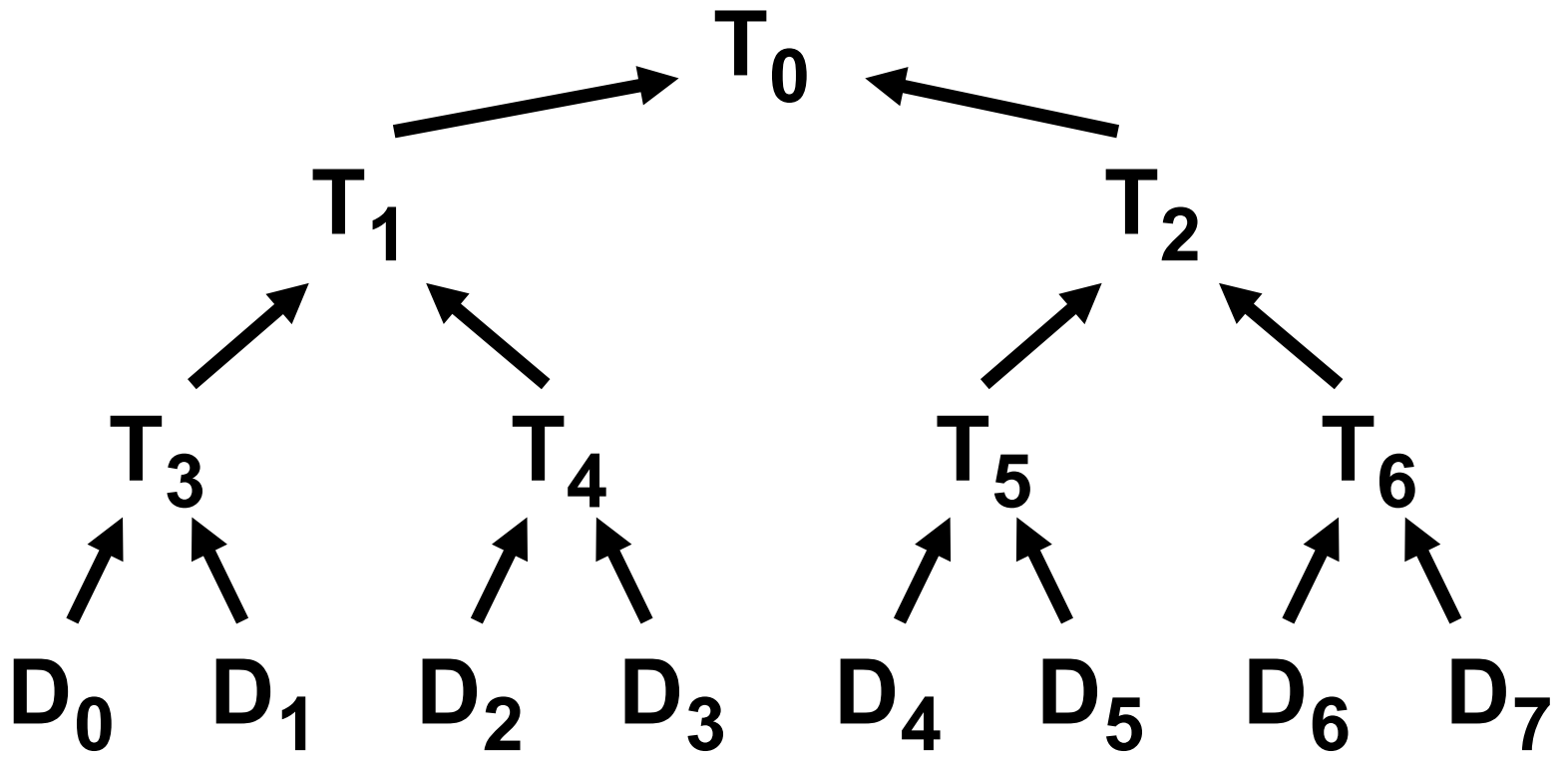
Verify if H(D) = H(D_1)

Merkle Hash Tree (Merkle 1979)

Construct binary tree over data values

Leaf nodes are data values

Parent = Hash(Left-child | | Right-child)



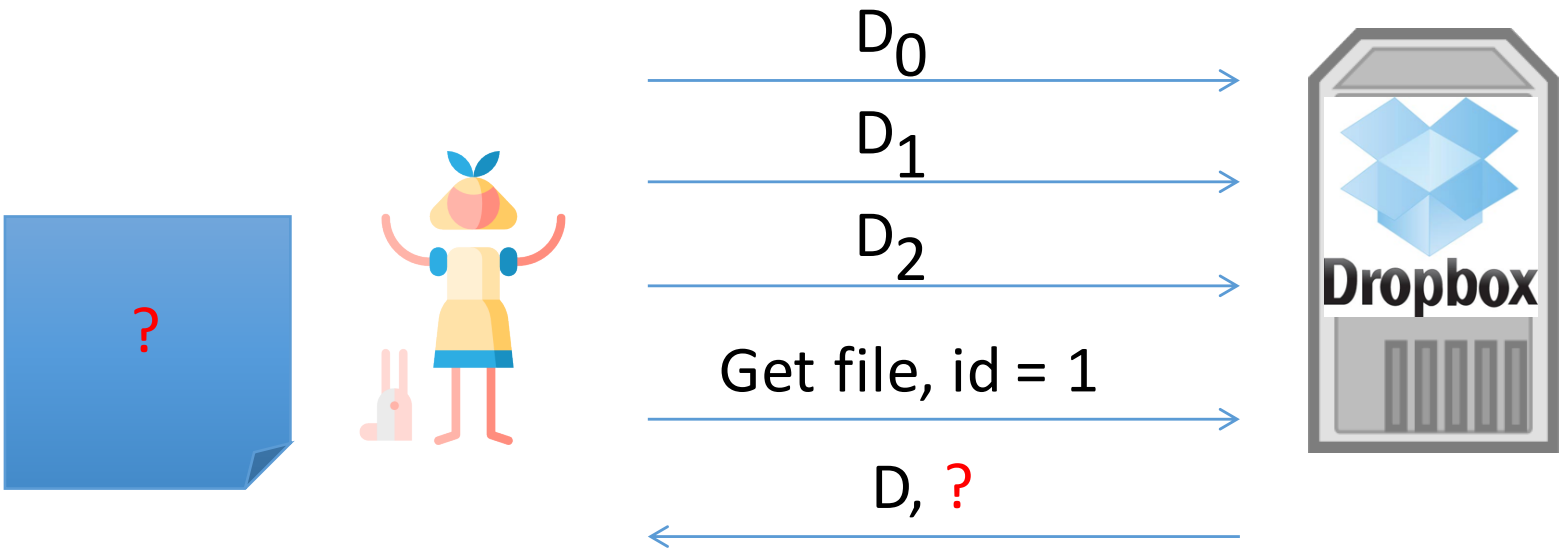
如何確保雲端資料的完整性？

Approach #2: using a Merkle hash tree

For simplicity, assume Alice never updates her files

What should Alice keep and verify?

What should the cloud return?



Verify if ?

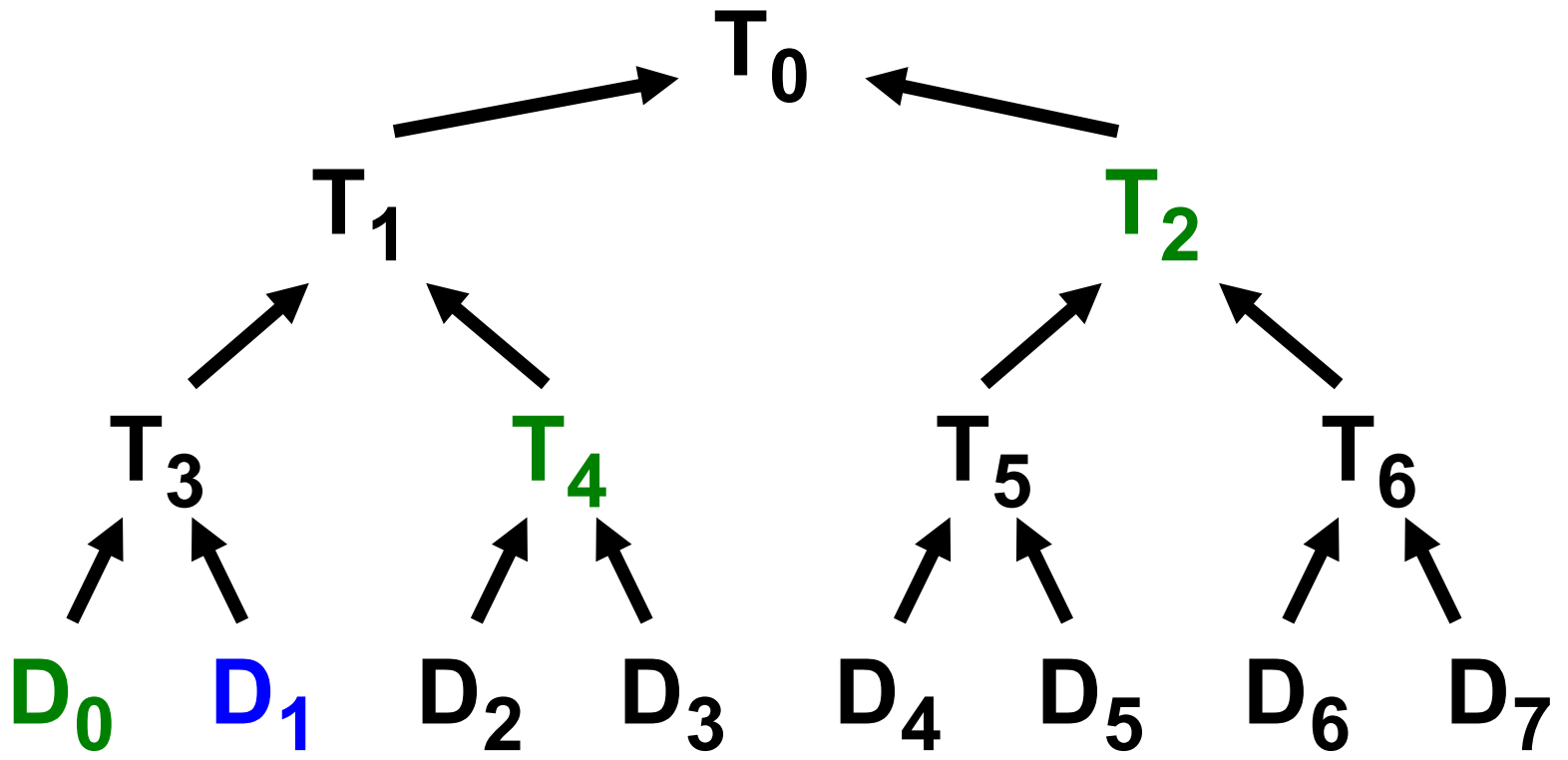
Integrity check using Hash Trees

Alice computes T_0 before uploading D_0 to D_7

Alice requests file id=1 along with a proof

Cloud returns D_1, $\{D_0, T_4, T_2\}$

Alice verifies $T_0 = H(H (H(D_0 \,||\, D_1) \,||\, T_4) \,||\, T_2)$



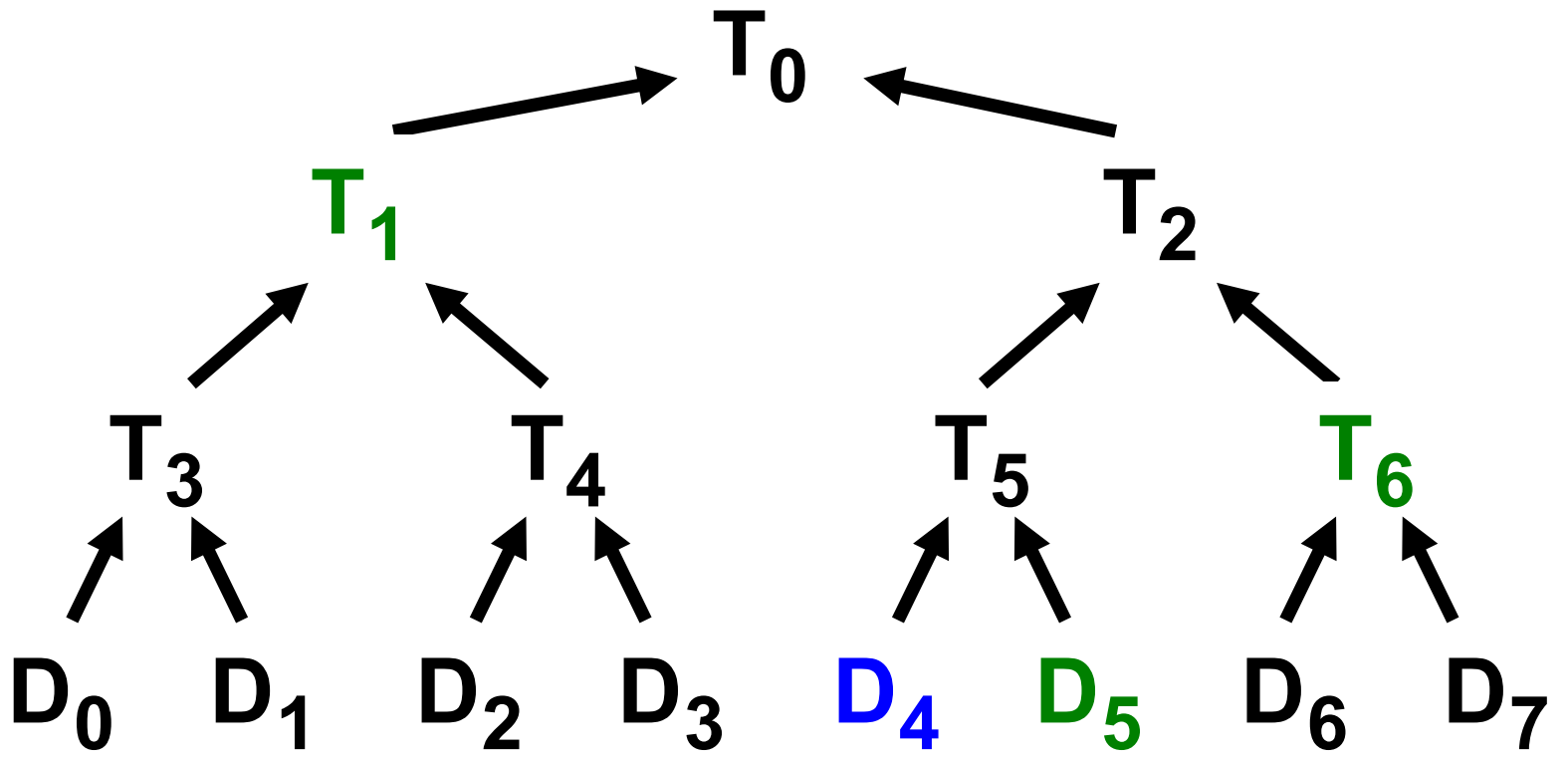
Integrity check using Hash Trees

Alice computes T_0 before uploading D_0 to D_7

Alice requests file id=4 along with a proof

Cloud returns D_4, {?}

Alice verifies T_0 = ?



One-Time Signatures

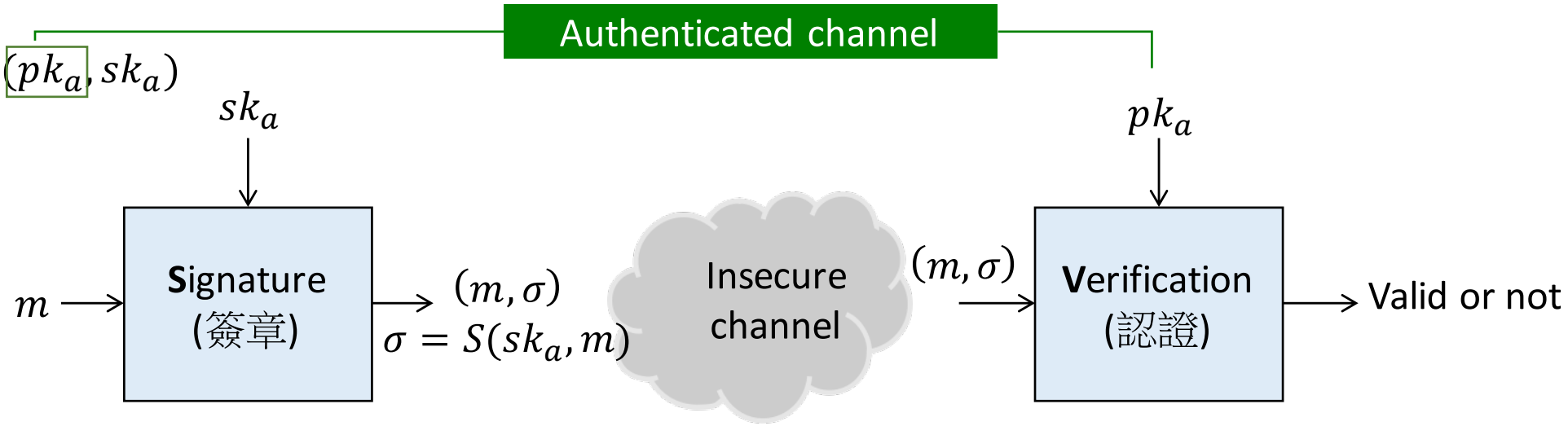
Digital Signatures

Alice has a public/private key pair (pk_a, sk_a)

Only Alice can create this signature (non-repudiation)

Eve can also verify this signature, but any modification will be detected

Can be combined with hash to efficiently sign longer messages



One-Time Signatures

RSA

Challenge: standard digital signatures are expensive for generation and verification

Goal: amortize single digital signature to sign multiple messages

One-time signatures

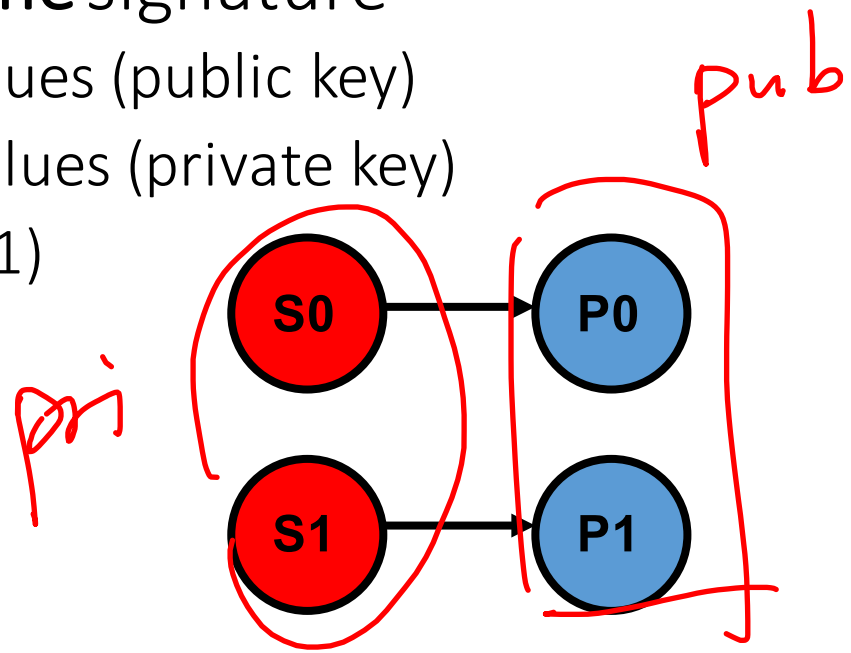
Efficient for signature generation and verification

Secure against quantum computing

Caveat: can only use one time, large keys

Example: **1-bit one-time** signature

- P0, P1 are public values (public key)
- S0, S1 are private values (private key)
- P0 = H(S0), P1 = H(S1)



One-time signatures

Alice wants to sign one bit in {0, 1}

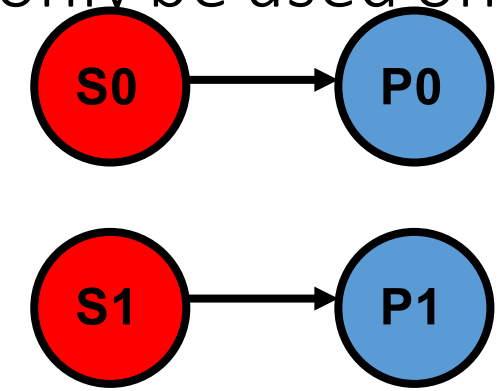
Alice picks S0, S1 at random and computes P0 and P1

Alice publishes P0 and P1

To sign m0, she releases S0 = Sig(m0)

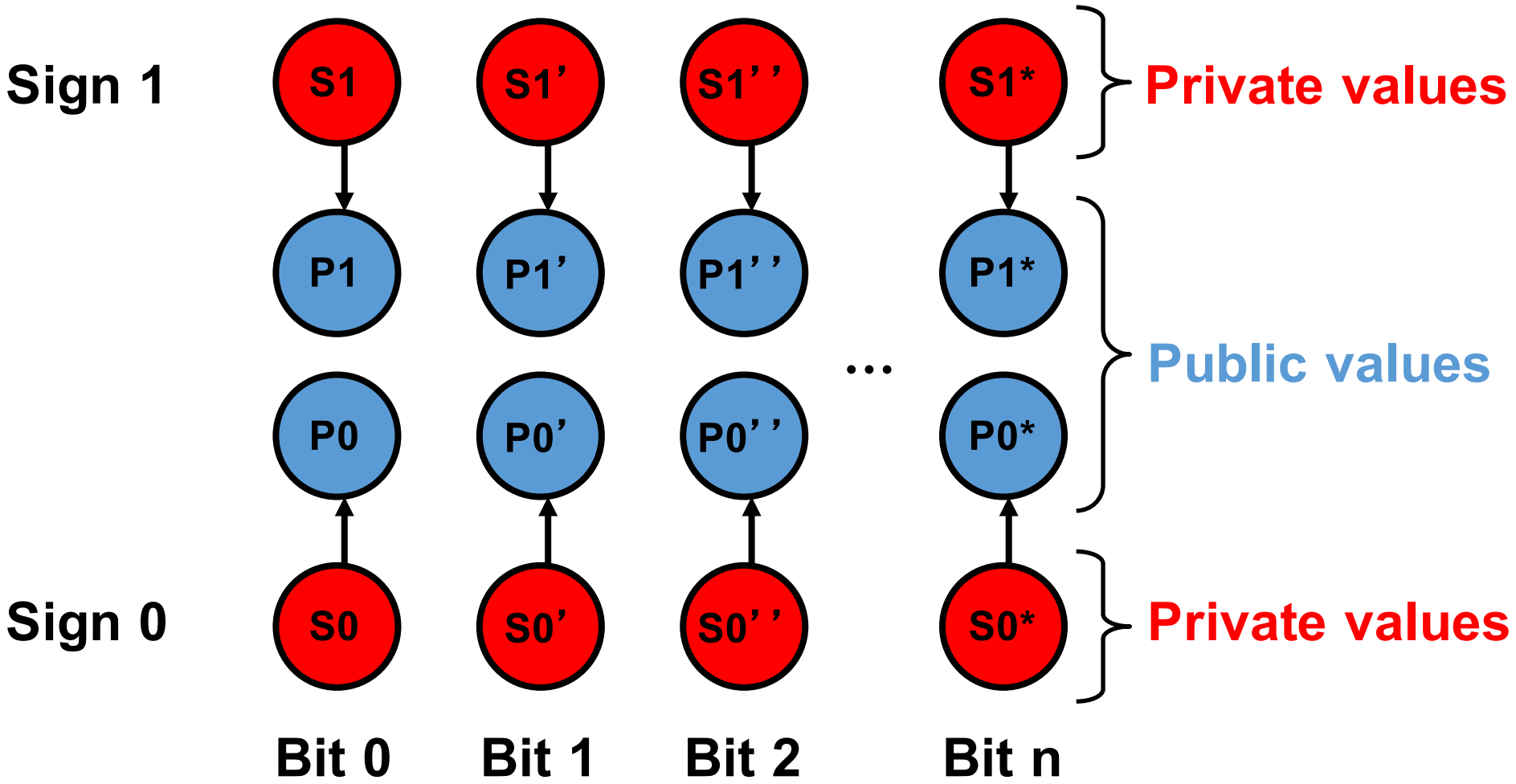
Otherwise, to sign m1, she releases S1 = Sig(m1)

Note that the signature can only be used once



Lamport's One-Time Signature

Uses 1-bit signature construction to sign multiple bits



假設我們只 sign 0000，這樣我們會公布對應的 S0, S0', S0'', S0''', 這時想說 sign 1 的 S1, S1', S1', S1''' 沒有用到，就留下來繼續等一下使用，這樣會有一個問題：等一下我們 sign 1111 時，attack 會取得某一個 bit 的 secret pair {(S0, S1), (S0', S1'), (S0'', S1''), (S0''', S1''')} 的 pair，這樣他日後就能偽造任何他喜歡的 signature

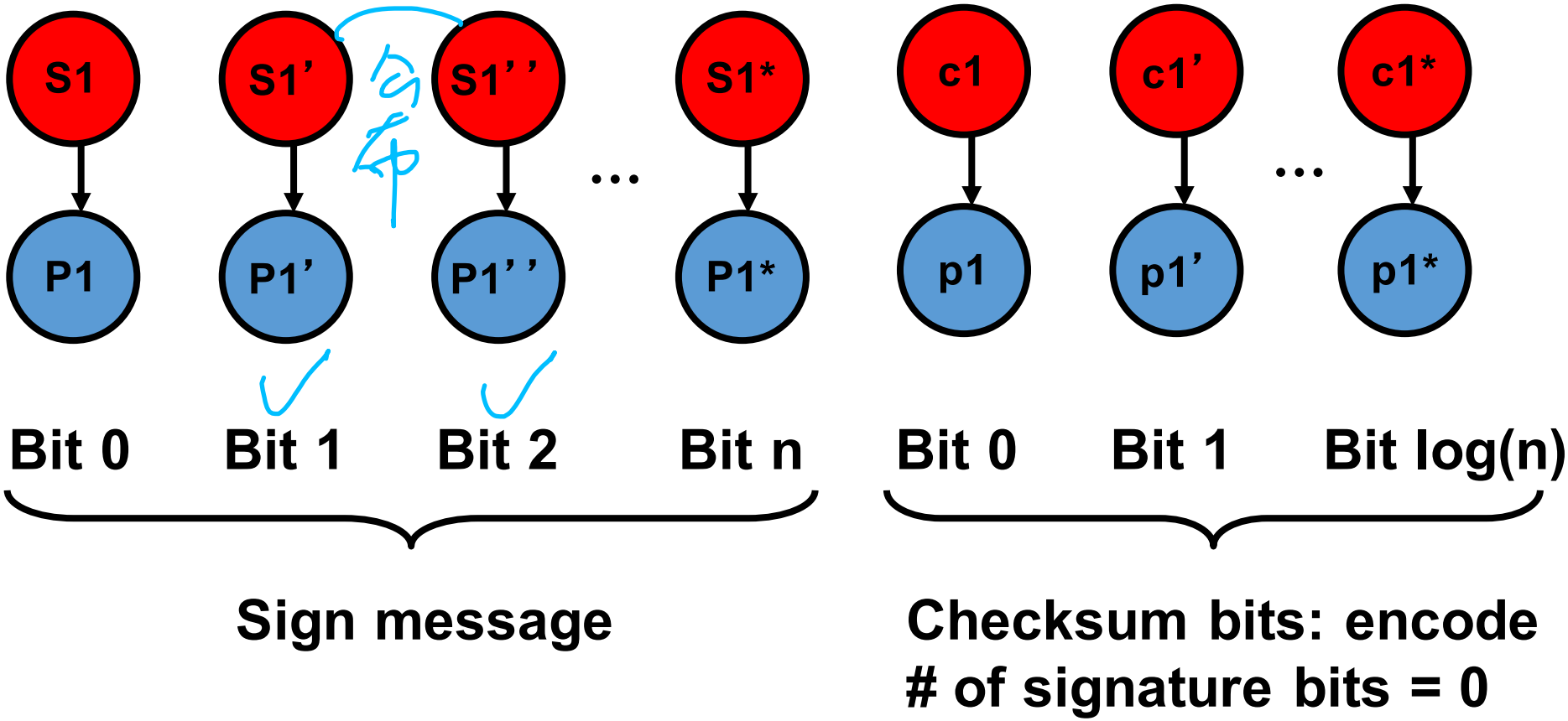
若我們要 sign 的 message 為 0110，我們就只公布 S1', S1''，這樣代表我們只 sign bit1 和 bit2，其它沒有公布的 bit0, bit3 就默認是 sign 0。
這樣會一個問題：如果 attacker 把 S1'' 丟掉，你會以為我們要 sign 的 message 為 0010。如果有 checksum bits 就能防止攻擊者把 pair 丟掉，我們會再多公布我們「總共 sign 了幾個 0」，以這裡例子來說是 2 個 0，所以我們就在多公布 "10" 代表我們 sign 了兩個 0，這樣大家如果只看到 sign 了 1 個 1，驗算後就會發現其實要有 "11" – "10" = "01"（0–index，所以 "01" 表 2）2 個 1。

Improved Construction I

注意：為什麼 checksum bits 要 encode 0 的個數而不是 encode 1 的個數？這樣 attacker 就能也把 checksum 丟掉某幾個 bit，去導致被丟掉幾個 bit 的 checksum bits' 記錄 1 的個數 = 被 attacker 丟掉幾個 1 後 sign 1 的個數。

Uses 1-bit signature construction to sign multiple bits

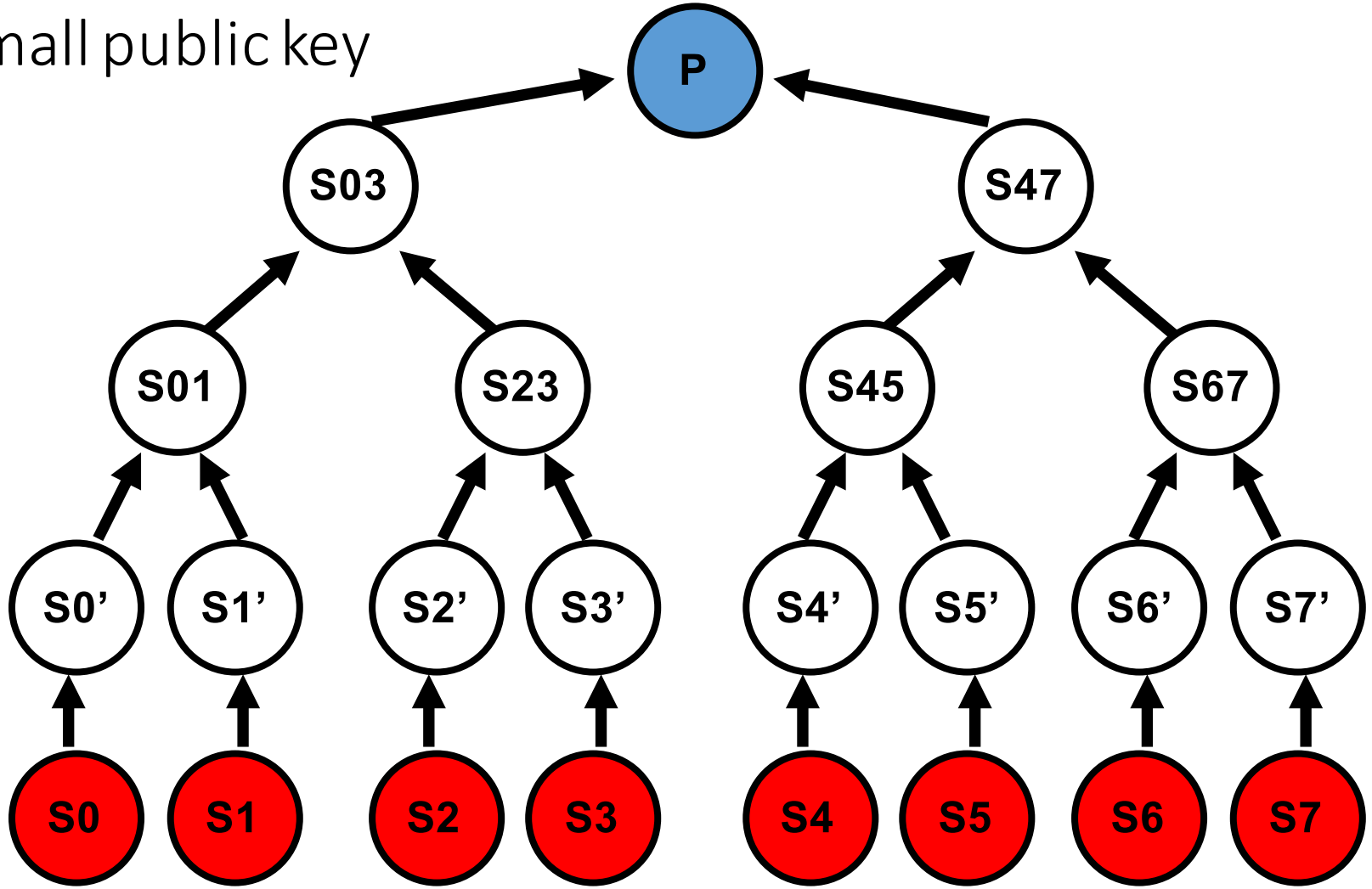
但如果 checksum bits 是 encode 0 的話，這樣就沒辦法，因為 attacker 若也想對 checksum bits 動手腳，也只能減少 0 的個數，減手 0 的個字數同時代表了增加 1 的個個字，沒有辦法 match 起來騙過去。



Merkle Signatures

Can sign multiple messages

Small public key



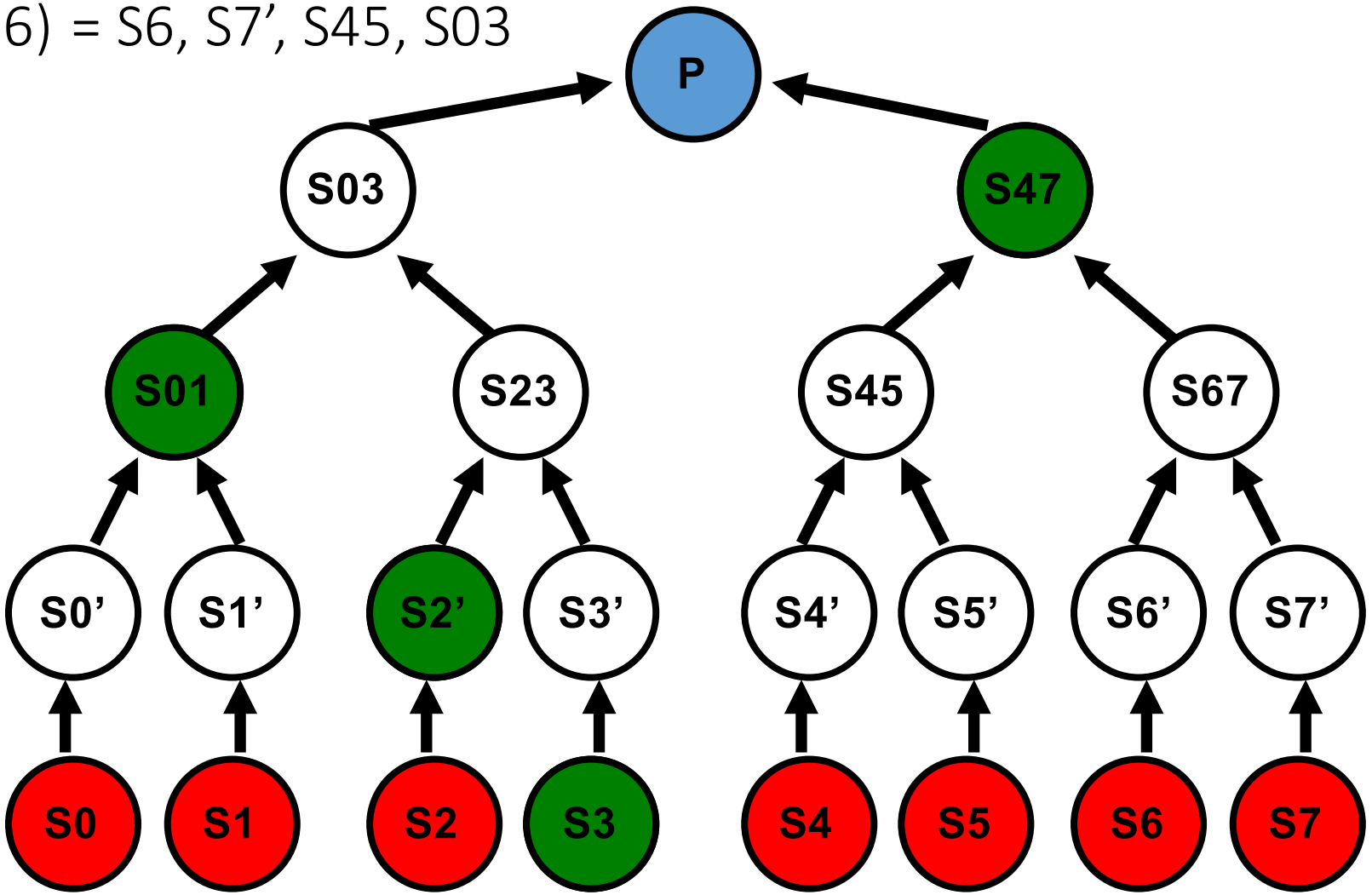
Alice wants to sign messages in {m0, m1, …, m7}

Alice picks S0…S7 at random

She computes and publishes P

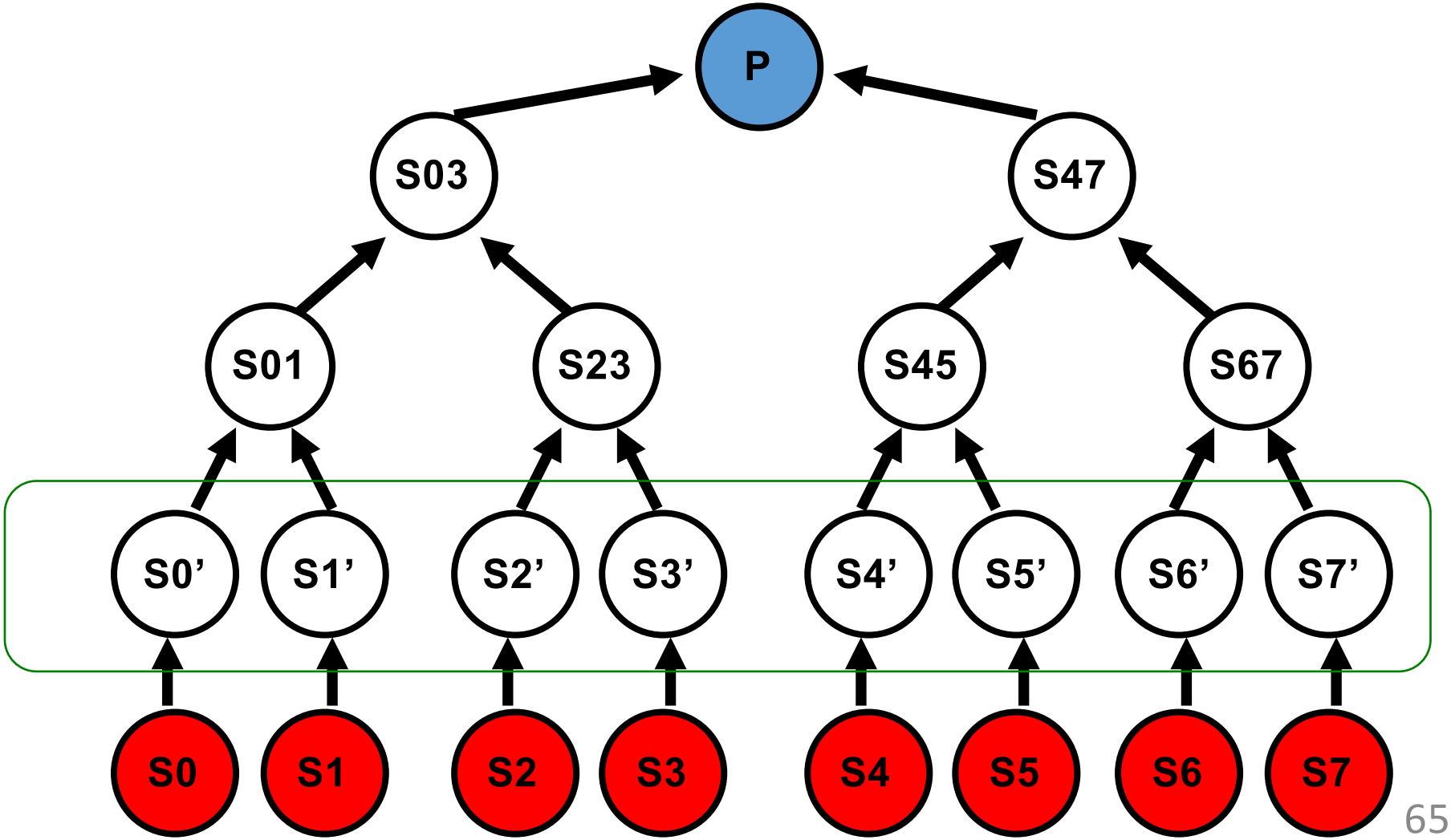
Sig(m3) = S3, S2', S01, S47

Sig(m6) = S6, S7', S45, S03



Merkle Signatures

Why extra layer of hashing?



同場加映:

One-Way Hash Chains

Versatile cryptographic primitive

Construction

- Pick random r_N and public one-way function F
- $r_i = F(r_{i+1})$
- Secret value: r_N, public value r_0

Properties

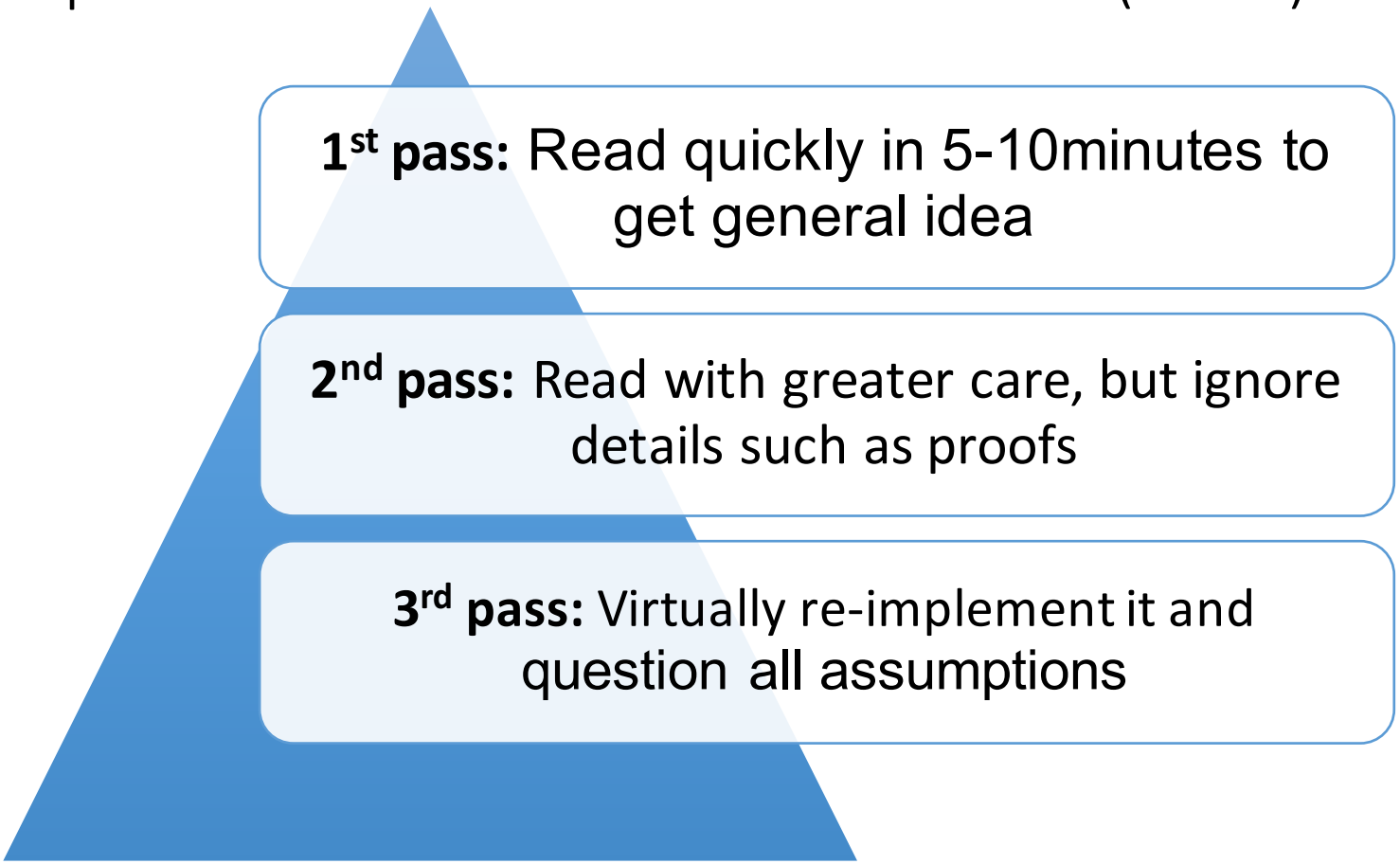
- Use in reverse order of construction: $r_1, r_2, r_3, \ldots, r_N$
- Infeasible to derive r_i from r_j $(j < i)$
- Efficiently authenticate r_i using r_j $(j < i)$: $r_j = F^{i-j}(r_i)$
- Robust to missing values

How to Read a Paper

同場加映!

A three-pass approach

Keshav, S. "How to read a paper." ACM SIGCOMM Computer Communication Review 37.3 (2007): 83-84.



1st pass: bird's eye view in 5-10 minutes

Quick scan of the paper

- Title, abstract, introduction
- Section and subsection headings
- Conclusions
- Glance over references

After this pass, you should be able to answer the five C's

- **Category**: What type of paper is it?
- **Context**: Where does it fit in?
- **Correctness**: Do assumptions make sense?
- **Contributions**: What are the main ones?
- **Clarity**: Is it well-written?

2nd pass: read in greater care

Spend ~1 hour to read in greater care, but ignore details

Write key points and make comments

Figures, diagrams, illustrations, graphs

Mark relevant unread references

After this pass, you should be able to summarize main story and identify main supporting evidence

3rd pass: virtually re-implement the paper

Can take one or more hours

Virtually re-implement the paper

Identify and challenge assumptions

Write down ideas for future work

After this pass, you should be able to

- Reconstruct entire structure of paper from memory
- Identify strong and weak points
- Pinpoint implicit assumptions, missing citations to related work, issues with experimental or analytical techniques

Questions?