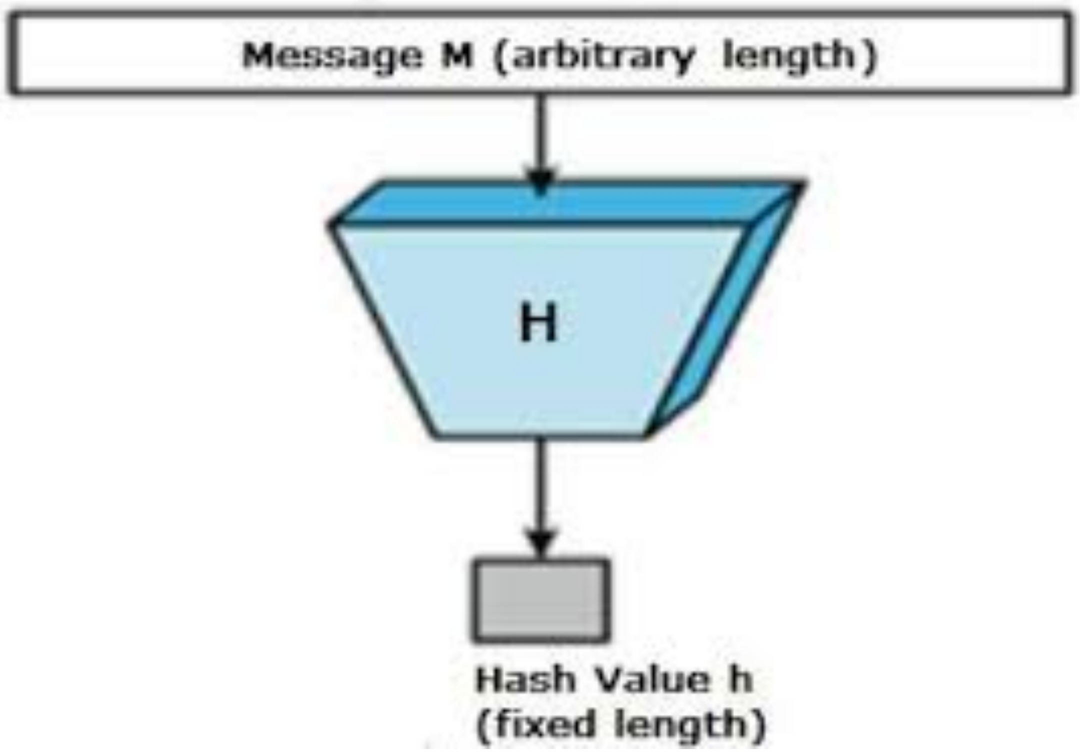
F5-1: (cryptographic) hash function

tkkwon@snu.ac.kr

some slides from Dan Boneh@Stanford

What is a hash function?

- any function that can be used to map data of arbitrary size to data of fixed size
  - Output: hash, digest, tag, fingerprint, …
- Hash functions accelerate table or database lookup by detecting duplicated records in a large file



$$H: \{0,1\}^* \to \{0,1\}^n$$

Source: tutorialspoint.com

Classifications of hash functions

- Cryptographic
- Non-cryptographic
  - try to avoid collisions for non malicious inputs
    - E.g. data corruption check in communications
  - Weaker guarantee on collision avoidance
    - faster computation

Another classification

- Unkeyed hash fn.

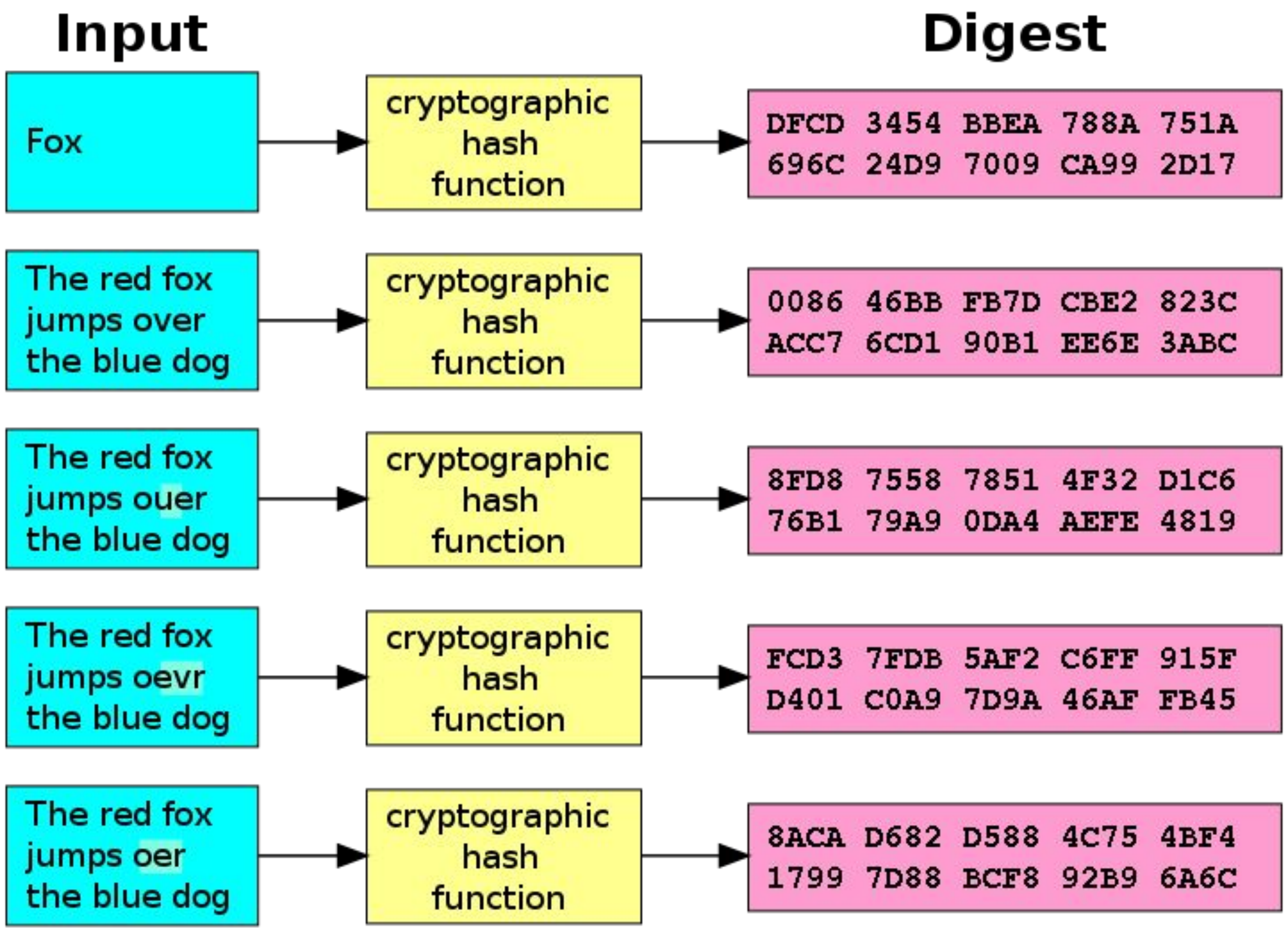
- Keyed hash fn.
    - Two inputs: message and key

Cryptographic hash function (CHF)

- A cryptographic hash function is a special class of hash functions that has certain properties which make it suitable for use in cryptography.

- What properties?
  - naive statement: a hash should be sufficiently long
  - What else?

An Example of CHF



Source: wikipedia

"avalanche effect"

Desirable properties of CHF

- One way:
  - Given y, it is infeasible to find x such that $h(x) = y$
- Collision-resistance:
  - it is infeasible to find x and y, with $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$
- Second pre-image resistance:
  - Given x, it is infeasible to find $x' \neq x$ such that $h(x') = h(x)$
  - Target collision resistance

- Pseudo-randomness
- Non-malleability
  - Given $h(x_1)$, It is infeasible to derive $h(x_2)$ where x_1 and x_2 are related,
  - e.g. $x_2 = x_1 + 1$

Random oracle (RO) model

- a random oracle is an oracle (a theoretical black box) that responds to every unique query (or input) with a (truly) random response chosen uniformly from its output domain.
- If a query is repeated, it responds with the same result
- RO is an ideal cryptographic hash function

- How the RO behaves:

1. At the beginning of the game, the oracle's table is empty.
2. Whenever some party asks the oracle to hash a message (i.e., input), the oracle first checks to see if that input value is already stored in the table.
3. If not, it generates a *random output string*, stores the input message and the new output string in its table, and returns the output string to the caller.
4. If the oracle *does* find the input value in the table, it returns output value that's already stored there.

An Independence theorem

Suppose $h: X \to Y$ satisfies the random oracle model, and $X_0 \subset X$

Suppose the values of $h(x)$ have been determined for $x \in X_0$, then $P[h(z) = y] = 1/|Y|$ for all $y \in Y$ and for all $z \in X \setminus X_0$

- This is a conditional prob.
- The knowledge of previously computed values of $h(\bullet)$ does not give any advantages for future computation of $h(\bullet)$
- This assumption in the RO model will be used in the following proofs

Applications of CHF

- Password storage
- File modification detector
- Digital signature
- Commitment
  - E.g. auction

Brute force Attacks on Hash Functions

- Attacking one-wayness
  - Goal: given $h: X \rightarrow Y$ & $y \in Y$, find x such that $h(x)=y$
  - Algorithm:
    - pick a random value x in X, check if $h(x)=y$, and if so, returns x; otherwise iterate
    - after failing q iterations, return fail
  - The average-case success probability is
    - $$\varepsilon = 1 - \left(1 - {}^{1}/_{|Y|}\right)^{q} \approx \frac{q}{|Y|}$$
  - Let $|Y|=2^m$, to get ε to be close to 0.5, $q \approx 2^{m-1}$

The Birthday Problem (attack on collision resistance)

What is the probability that at least two of *k* randomly selected people have the same birthday? (Same month and day, but not necessarily the same year. Assume 365 days in a year)

The Birthday Problem

How large must *k* be so that the probability is greater than 50 percent?

The answer is 23

It is also called the birthday paradox in the sense that a mathematical truth contradicts common intuition.

Calculating the Probability (birthday example)

- Assumptions
  - Nobody was born on February 29 (365 days only)
  - People's birthdays are equally distributed over 365 days

Calculating the Probability-2

- In a room of *k* people

  q: the prob. all people have different birthdays

$$q = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdots \frac{365-(k-1)}{365}$$

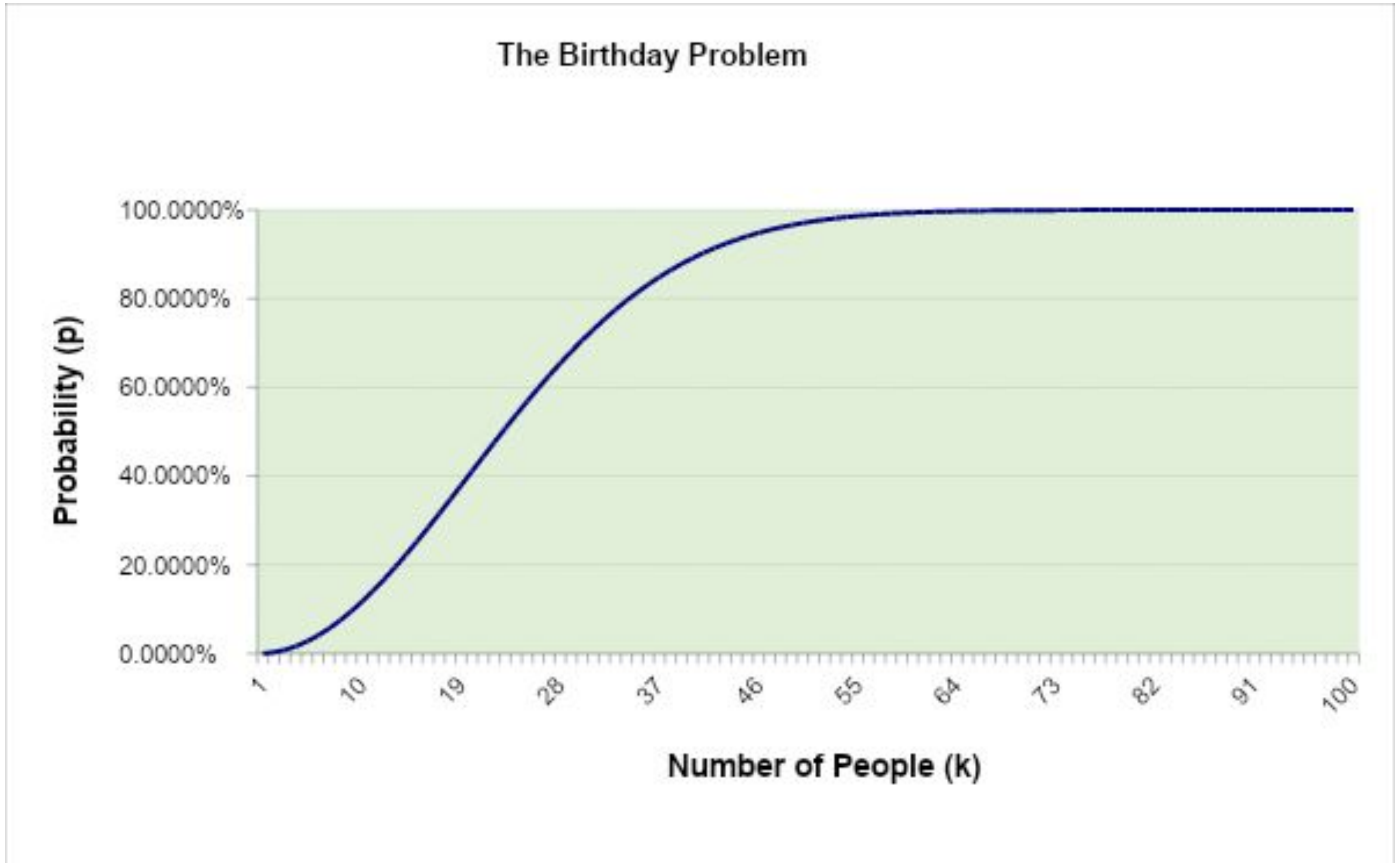
$$q = \frac{365!/(365-k)!}{365^k}$$

p: the prob. at least two of them have the same birthdays

$$p = 1 - q = 0.5 \Rightarrow k = 23$$

Calculating the Probability-3

- Shared Birthday Probabilities



Collision Search-1

For collision search, select distinct inputs x_i for i=1, 2, ... , k, where n is the number of possible hash values and check for a collision in the $h(x_i)$ values

The prob. that no collision is found after selecting k inputs is

$$p_{\text{no\_collision}} = \left(1 - \frac{1}{n}\right) \cdot \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{3}{n}\right) \cdot\cdot \left(1 - \frac{(k-1)}{n}\right)$$

(In the case of the birthday paradox, k is the number of people randomly selected, and the collision condition is the birthday of the people, and n=365.)

Collision Search-2

For large *n*

$$p_{\text{no collision}} = \left(1-\frac{1}{n}\right)\cdot\left(1-\frac{2}{n}\right)\cdots\left(1-\frac{(k-1)}{n}\right) \approx e^{-k^2/(2n)}$$

$1 - x \approx e^{-x}$ when x is small

$$\left(1-\frac{1}{n}\right) \approx e^{-1/n}$$

$$\begin{aligned} p_{\text{no collision}} &= e^{-1/n}\cdot e^{-2/n}\cdots e^{-(k-1)/n} \\ &= e^{-((1+2+3+\ldots+(k-1))/n)} \\ &= e^{-k\cdot(k-1)/2n} \end{aligned}$$

Collision Search-3

When *k* is large, the percentage difference between *k* and *k-1* is small, and we may approximate *k-1* ≈ *k*.

$$p_{\text{no collision}} = e^{-k\cdot(k-1)/2n} = e^{-k^2/2n}$$

$$p_{\text{at least one collision}} = 1 - e^{-k^2/2n}$$

Collision Search-4

$$p = 1 - e^{-k^2/2n}$$

$$e^{-k^2/2n} = 1 - p$$

$$e^{k^2/2n} = \frac{1}{1-p}$$

$$\frac{k^2}{2n} = \ln\left(\frac{1}{1-p}\right)$$

$$k = \sqrt{2n * \ln\left(\frac{1}{1-p}\right)}$$

- For the birthday case, the value of *k* that makes the probability closest to 1/2 is 23

$$k = \sqrt{2n * \ln 2}$$

$$= 1.1774\sqrt{n}$$

$$= 1.1774 * \sqrt{365} = 22.49$$

Attack Prevention

The important property is the length in bits of the message digest produced by the hash function.

If the number of *m* bit hash, the cardinality *n* of the hash function is

$$n = 2^m$$

The 0.5 probability of collision for m bit hash, expected number of operation *k* before finding a collision is very close to

$$k \approx \sqrt{n} = 2^{m/2}$$

m should be large enough so that it's not feasible to compute hash values!!!

Collision resistance: review

Let $H: M \to T$ be a hash function ($|M| >> |T|$)

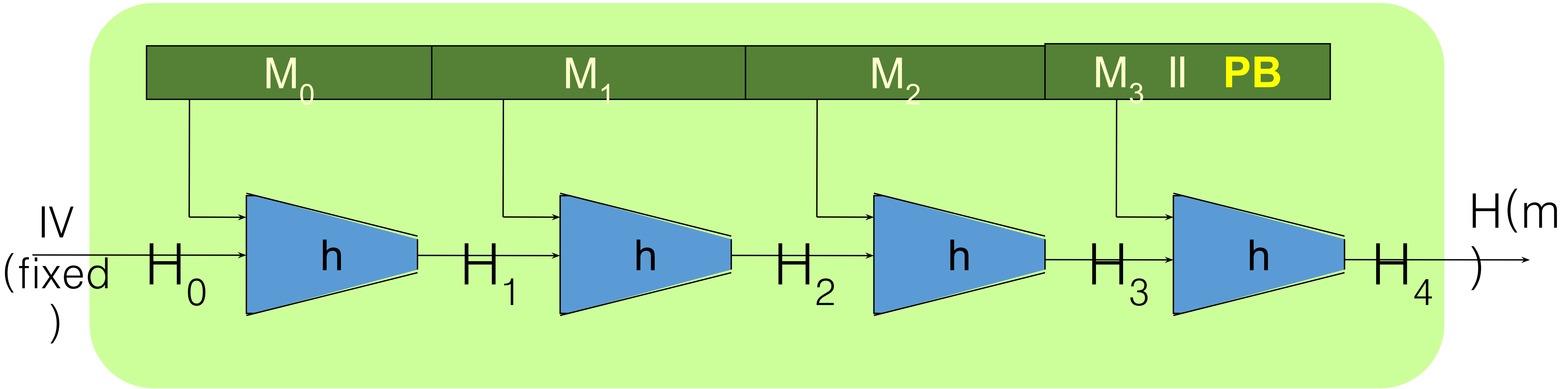
A **collision** for H is a pair m_0 , $m_1 \in M$ such that:

$$H(m_0) = H(m_1) \quad \text{and} \quad m_0 \neq m_1$$

Goal: collision resistant (C.R.) hash functions

Step 1: given C.R. hash function for **short** messages,
construct C.R. hash function for **long** messages

Merkle–Damgärd (MD) iterated construction



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

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

PB (padding block) 1000…0 ll msg len

64 bits

If no space for PB,
add another block

source: Dan Boneh@Stanford

MD collision resistance

Thm: if h is collision resistant, then so is H.

Proof: collision on H $\Rightarrow$ collision on h

Suppose H(M) = H(M'). We build a collision for h.

$\boxed{M \neq M'}$

$$IV = H_0, \quad H_1, \quad \cdots, \quad H_t, \quad H_{t+1} = H(M)$$

$$IV = H_0', \quad H_1', \quad \cdots, \quad H'_r, \quad H'_{r+1} = H(M')$$

$$h(H_t, M_t \parallel PB) = H_{t+1} = H'_{r+1} = h(H'_r, M'_r \parallel PB')$$

source: Dan Boneh@Stanford

If $H_t \neq H'_r$ or $M_t \neq M'_r$ or $PB \neq PB'$

We find the collision. Stop.

Suppose $H_t = H'_r$ and $M_t = M'_r$ and $PB = PB'$

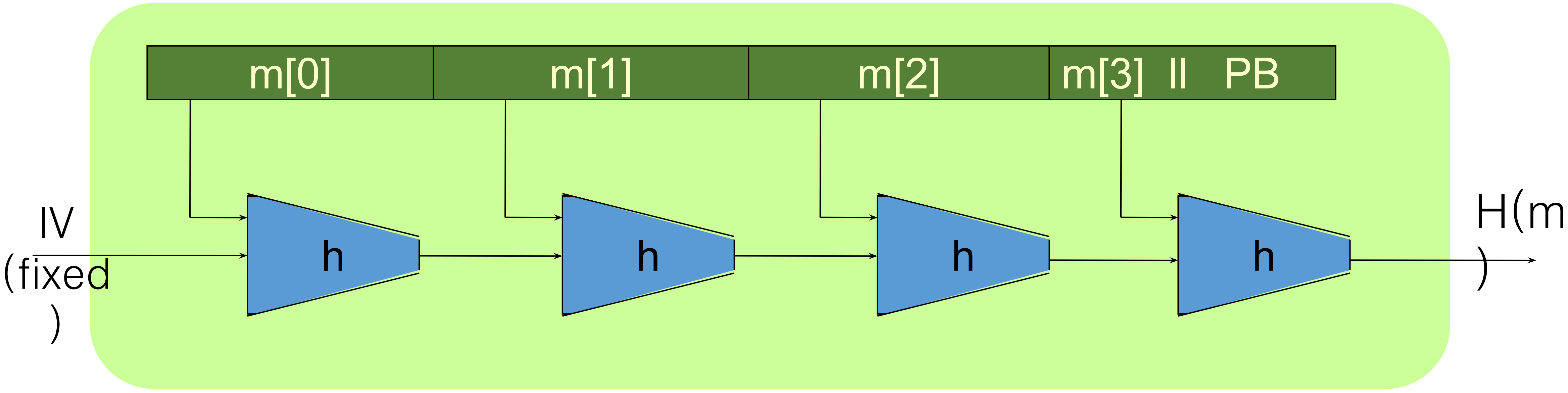
Then: $h(H_{t-1}, M_{t-1}) = H_t = H'_r = h(H'_{r-1}, M'_{r-1})$

This process goes on and on

source: Dan Boneh@Stanford

How to build a hash fn.

Merkle–Damgärd iterated construction



Thm: h collision resistant ⇒ H collision resistant

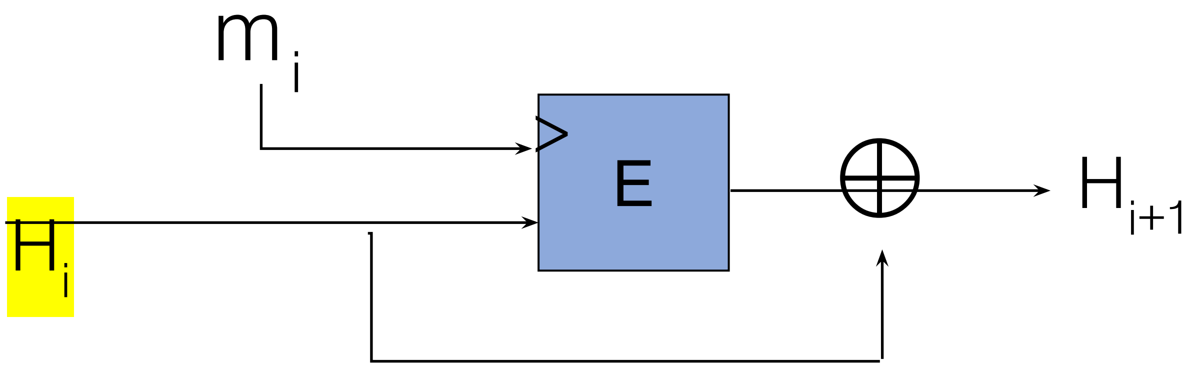
Goal: construct a compression function $h: T \times X \to T$

source: Dan Boneh@Stanford

Compression fn. from a block cipher

E: K× $\{0,1\}^n \to \{0,1\}^n$ a block cipher.

Davies-Meyer (D-M) compression function: $h(H_i, m) = E(m, H_i) \oplus H$



<u>**Thm**</u>: Suppose E is an ideal cipher ($|K| > 2^n!$)

Finding a collision **h(H,m)=h(H',m')** takes $O(2^{n/2})$ evaluations of (E,D).

Why is there XOR in D–M fn.?

For example, let's say you don't XOR, you only encrypt the IV with the block of text you are hashing. Let's call the result h.

h=E(IV1,text1)

We can now create a random block of the same size as the IV and decrypt.

text2 = D(IV2,h)

And now we know that: E(IV2,text2) = h

Then E(IV1,text1) = E(IV2,text2)

Thus we have created a collision.

Toy example of a compression function

- Break m into n-bit blocks, append zeros to get a multiple of *n*.
- There are l blocks, where $l = |m|/n$

- Fast! But not very secure.

$$m = \begin{bmatrix} m_1 \\ m_2 \\ \dots \\ m_l \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{l1} & m_{l2} & \cdots & m_{ln} \end{bmatrix}$$

$$\begin{matrix} \oplus & \oplus & \oplus & \oplus \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \end{matrix}$$

$$\begin{bmatrix} c_1 & c_2 & \dots & c_n \end{bmatrix} = h(m)$$

- Doing a left shift on the rows helps a little, but still not secure

$$\begin{bmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{22} & m_{23} & \dots & m_{21} \\ \vdots & \vdots & \vdots & \vdots \\ m_{ll} & m_{l,l+1} & \cdots & m_{l,l-1} \end{bmatrix}$$