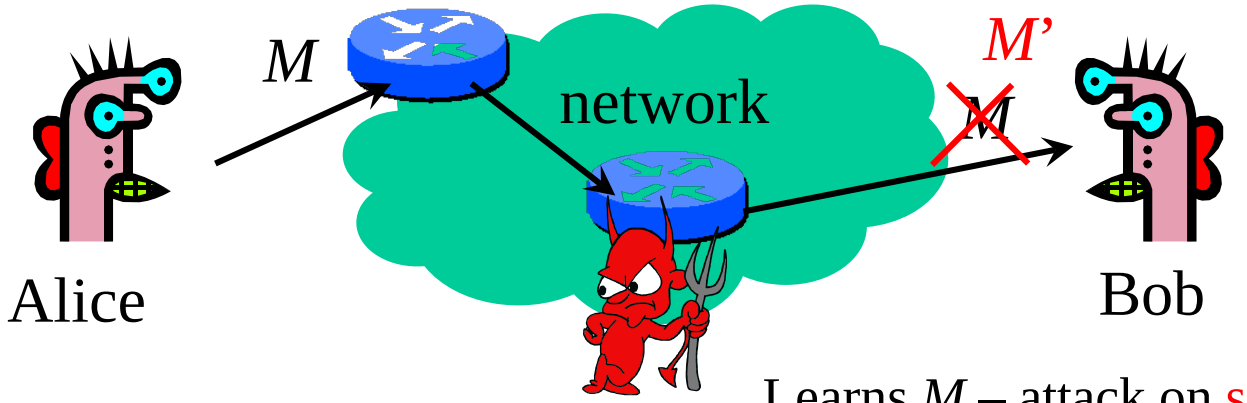
Hashes and Message Digests

Haipeng Dai

haipengdai@nju.edu.cn
313 CS Building
Department of Computer Science and Technology
Nanjing University

Integrity vs. Secrecy

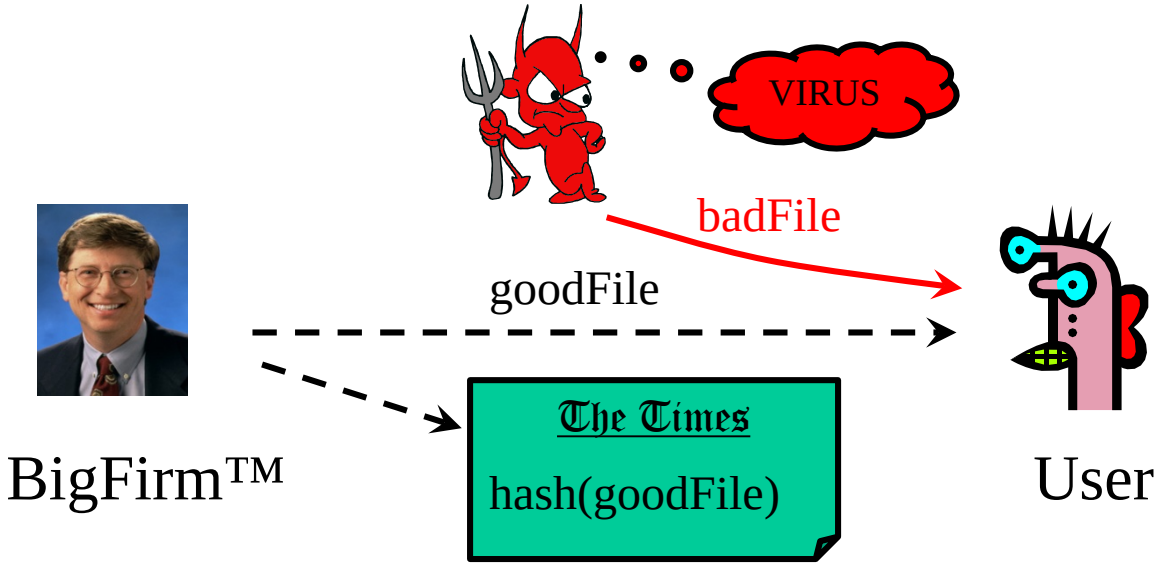
- Integrity: attacker cannot change message without being detected
- Encryption: attacker cannot read message correctly without key
  - Note: encryption does not guarantee integrity.
  - Reason: attacker can change ciphertext arbitrarily and any ciphertext can be decrypted to get the corresponding plaintext (although possibly garbage).



Learns *M* – attack on secrecy

Changes *M* – attack on integrity

Integrity Protection



Software manufacturer wants to ensure that the executable file
is received by users without modification…

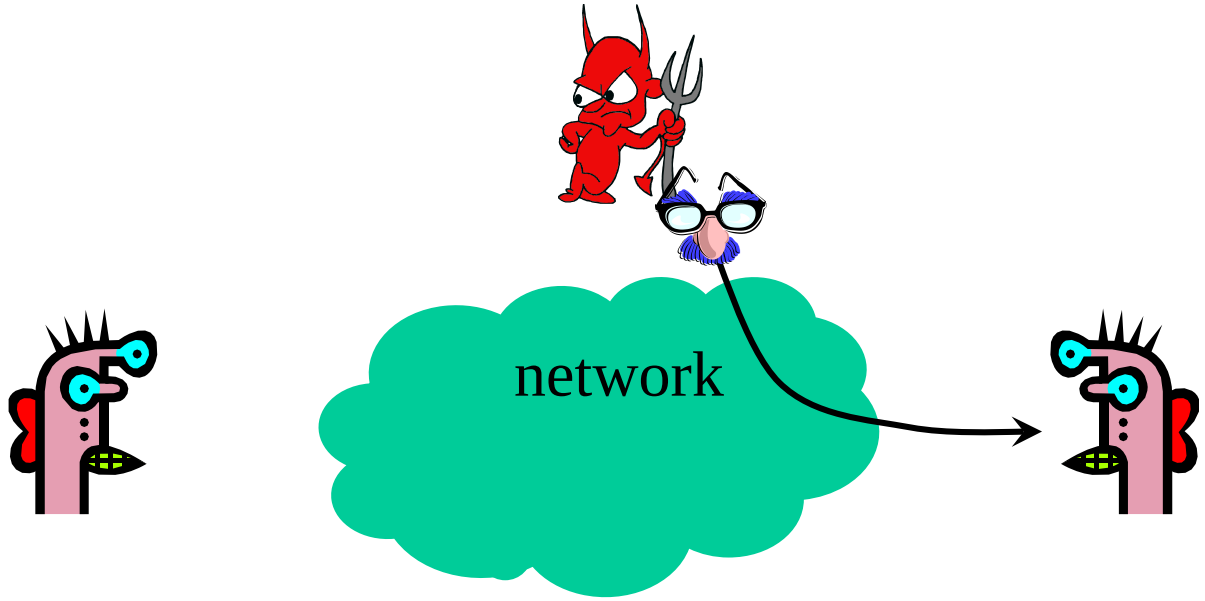
Sends out the file to users and publishes its hash in NY Times

The goal is integrity, not secrecy

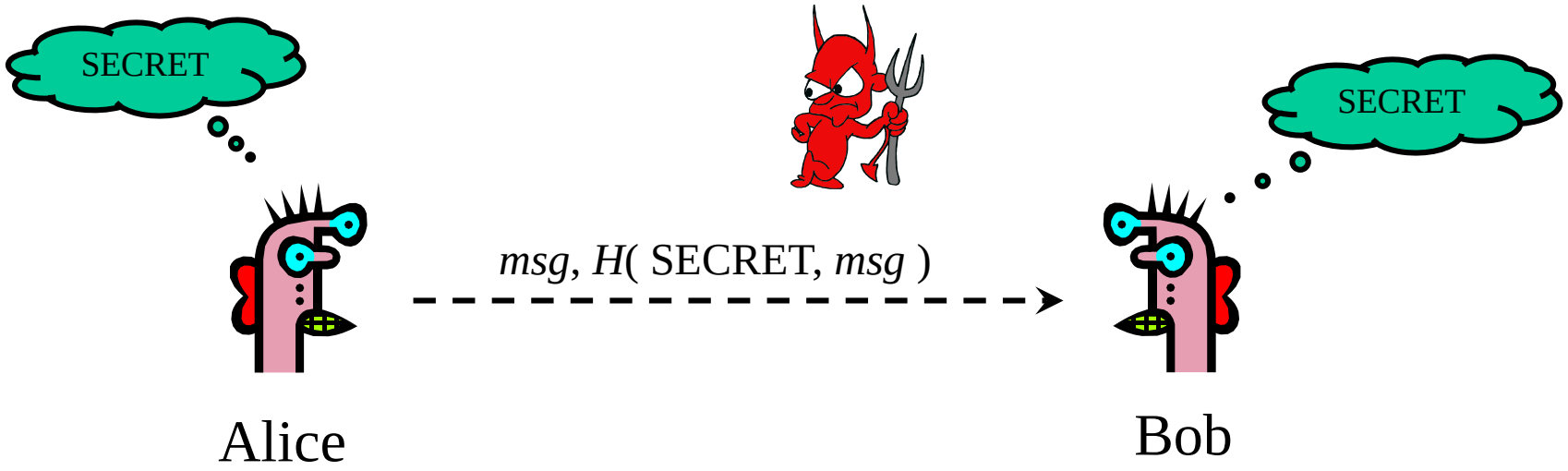
Idea: given goodFile and hash(goodFile),
very hard to find badFile such that hash(goodFile)=hash(badFile)

Authentication

- Authenticity is identification and assurance of origin of information
  - We'll see many specific examples in different scenarios



Authentication with Shared Secrets

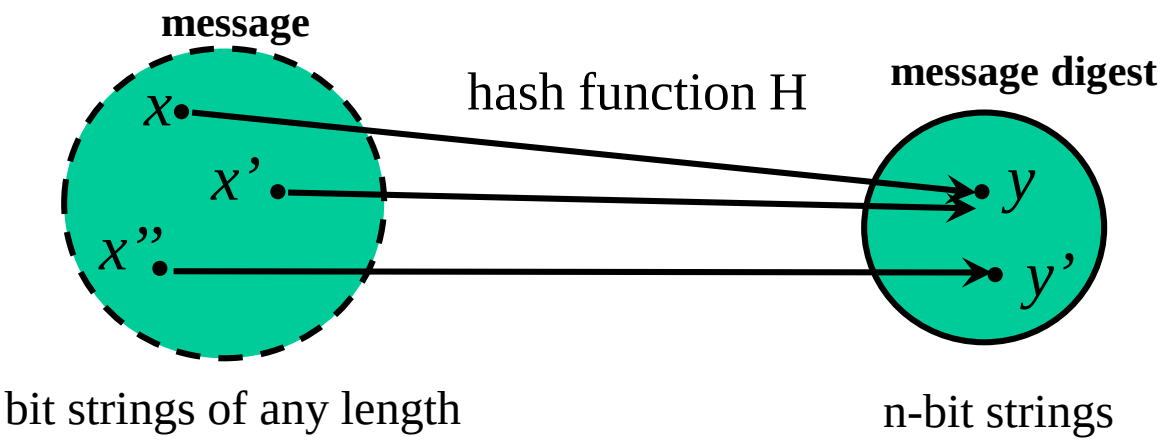


Alice wants to ensure that nobody modifies message in transit (both integrity and authentication)

Idea: given *msg*,
very hard to compute *H*(SECRET, *msg*) without SECRET;
easy with SECRET

Hash Functions: Main Idea

- Hash is also called message digest
- Arbitrary-length message to fixed-length digest



- H is a lossy compression function
  - Collisions: $h(x)=h(x')$ for some inputs x, x'
  - Result of hashing should "look random" (make this precise later)
    - Intuition: half of digest bits are "1"; any bit in digest is "1" half the time
- Cryptographic hash function needs a few properties…

Hashes and Message Digest

- **One-wayness**: given h(x), hard to find x
  - Also called “pre-image resistance”

- **Collision resistance**: hard to find x, x’ such that h(x)=h(x’)
  - Also called “strong collision resistance”
  - Brute-force collision search is only $O(2^{n/2})$, not 2^n
  - Birthday Paradox
  - n needs to be at least 128.

- **Weak collision resistance**: given x, hard to find x’ such that h(x)=h(x’)
  - Also called “second pre-image resistance”
  - Brute-force attack requires $O(2^n)$ time

One-Wayness

- Intuition: hash should be hard to invert
  - "Preimage resistance"
  - Let $h(x') = y \in \{0,1\}^n$ for a random x'
  - Given y, it should be hard to find x such that $h(x)=y$

- How hard?
  - Brute-force: try every possible x, see if $h(x)=y$
  - SHA-1 (common hash function) has 160-bit output
    - Assuming 2^{64} trials per second, can do 2^{89} trials per year
    - Will take 2^{71} years to invert SHA-1 on a random image

Collision Resistance (CR)

- Should be hard to find x, x' such that $h(x)=h(x')$
  - Brute-force collision search is only $O(2^{n/2})$, not $O(2^n)$
  - For SHA-1, this means $O(2^{80})$ vs. $O(2^{160})$

Birthday Paradox Problem

- **Birthday Problem:** What is the minimum number of persons in a room so that the odds are better than 50% that two of them will have the same birthday? **23 persons.**
- **General Problem:**

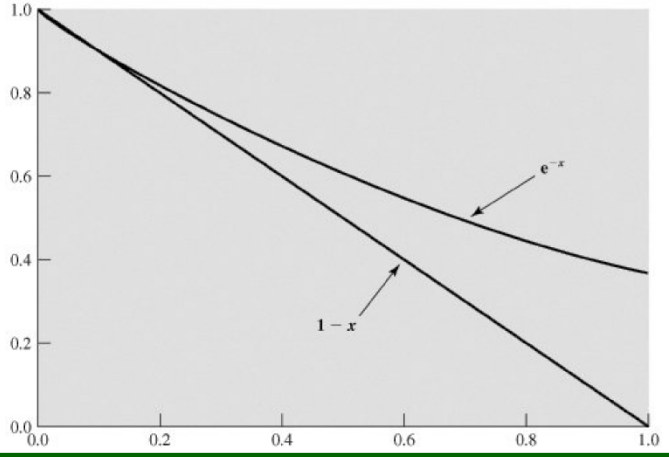
  Given a random variable that is an integer with uniform distribution between 1 and n and a selection of k instances ($k \le n$) of the random variable, what is the probability, $P(n, k)$, that there is at least one duplicate?
- How many possibilities?
  - n^k (samples with duplicate)
  - $n(n\text{-}1)\ldots(n\text{-}k\text{+}1)$ (samples without duplicate)
- Probability of no repetition? $\dfrac{n\times(n-1)\times\ldots\times(n-k+1)}{n^k}$
- Thus $P(n,k)=1-\dfrac{n\times(n-1)\times\ldots\times(n-k+1)}{n^k}=1-\left[\dfrac{n-1}{n}\times\dfrac{n-2}{n}\times\ldots\times\dfrac{n-k+1}{n}\right]=1-\left[\left(1-\dfrac{1}{n}\right)\times\left(1-\dfrac{2}{n}\right)\times\ldots\times\left(1-\dfrac{k-1}{n}\right)\right]$
- Because $1\text{-}x \le e^{-x}$ ($x \ge 0$), we have
- What value of k is required such that $P(n, k)>0.5$?

  $P(n, k)=1-e^{-k(k-1)/2n}>1/2$

  $\Rightarrow\ e^{-k(k-1)/2n}<1/2 \quad\Rightarrow\quad -k(k-1)/2n<-\ln 2$

  $\Rightarrow\ k(k-1)/2n>\ln 2$ Note: for large k, k(k-1) $\approx k^2$

  $\Rightarrow\ k^2/2n>\ln 2 \Rightarrow k>\sqrt{2(\ln 2)n}\approx 1.18\sqrt{n}$
- If n=365, we get $k>1.18\times\sqrt{365}=22.54$



Weak Collision Resistance (WCR)

- Given randomly chosen x, hard to find x' such that $h(x)=h(x')$
  - Attacker must find collision for a specific x. By contrast, to break collision resistance, it is enough to find any collision.
  - Brute-force attack requires $O(2^n)$ time

One-Wayness vs. Strong/Weak Collision Resistance

- **One-wayness does <u>not</u> imply collision resistance (neither strong nor weak).**
  - Suppose g is one-way
  - Define $h(x)$ as $g(x')$ where x' is x except the last bit

    i.e., $x'=x$ with last bit chopped off, define $h(x)$ as $g(x')$.
    - h is one-way (to invert h, must invert g)
    - Collisions for h are easy to find: for any x, $h(x0)=h(x1)$
    - Both strong and weak collisions are easy to find.
- **Collision resistance does not imply one-wayness.**
  - Suppose g is collision-resistant
  - Define $h(x) = \begin{cases} 0\|x & \text{if } x \text{ is } n\text{-bit long} \\ 1\|g(x) & \text{otherwise} \end{cases}$
    - Collisions for h are hard to find: if y starts with 0, then there are no collisions, if y starts with 1, then must find collisions in g
    - h is not one way: half of all y's (those whose first bit is 0) are easy to invert (how?). Random y is invertible with probability 1/2

CR vs. WCR (1/5)

- **Collision resistance implies weak collision resistance.**

Proof:

Given h; h is collision resistant.

Suppose h is not weak collision resistant.

Given x, we can find x' such that $h(x) = h(x')$.

So h is not collision resistant because we found a pair x and x' where $h(x) = h(x')$.

This is contradictory to our assumption.

CR vs. WCR (2/5)

- **Weak collision resistance does not imply collision resistance.**

Proof:

It is sufficient to construct a function, which is weak collision resistant and not collision resistant.

Let g be one way and collision resistant function.

Let $g^{(i)}(x)$ denote the result of applying g to x for i times.

For example $g^{(3)}(x) = g(g(g(x)))$

Define hash function h to work on a pair of values x, y.

$h(x, y) = (g^{(i)}(x), g^{(j)}(y), i + j)$

Here $i \geq 0$ is the least value to make $g^{(i)}(x)$ end with 4 zeroes.

If $x \in \{0, 1\}^n$ and x ends with 4 zeroes, then $i = 0$.

Similary $j \geq 0$ is the least value to make $g^{(j)}(y)$ end with 4 zeroes.

If $y \in \{0, 1\}^n$ and y ends with 4 zeroes, then $j = 0$.

We fix an upper bound, say 100, on these values, just to handle the case that this iteration would otherwise loop forever.

CR vs. WCR (3/5)

- ***h*(*x*, *y*) *is weak collision resistant.***

Given x_1, y_1, and $h(x_1, y_1) = (g^{(i1)}(x_1), g^{(j1)}(y_1), i_1 + j_1)$

Suppose we can find x_2, y_2 such that $(x_2, y_2) \neq (x_1, y_1)$, i.e. $x_2 \neq x_1$ or $y_2 \neq y_1$ but

$$h(x_2, y_2) = (g^{(i2)}(x_2), g^{(j2)}(y_2), i_2 + j_2) = h(x_1, y_1)$$
$$= (g^{(i1)}(x_1), g^{(j1)}(y_1), i_1 + j_1)$$
$$\Rightarrow \begin{cases} g^{(i1)}(x_1) = g^{(i2)}(x_2) \\ g^{(j1)}(y_1) = g^{(j2)}(y_2) \\ i_1 + j_1 = i_2 + j_2 \end{cases}$$

$g^{(i1)}(x_1) = g^{(i2)}(x_2)$ means that we can invert g **or** find a collision in g, i.e.,

$$\underbrace{g(g(\ldots.g(x_1)\ldots))}_{i_1} = \underbrace{g(\ g(\ldots.g(x_2)\ldots)\ )}_{i_2}$$

If $i_1 = 0$ but $i_2 \neq 0$, then $x_1 = g^{(i2)}(x_2)$, which means that we can invert g.

Similarly for $i_2 = 0$ but $i_1 \neq 0$.

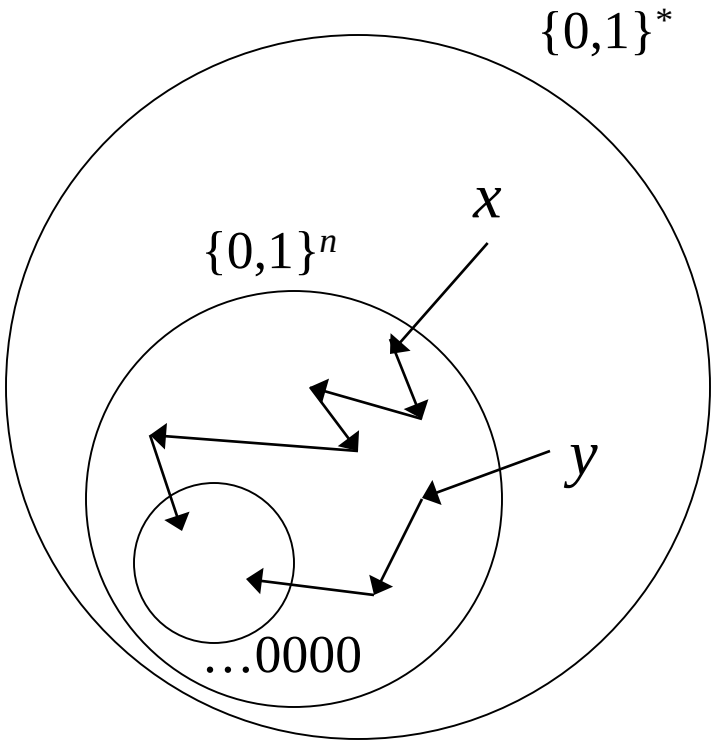
If both $i_1 = 0$ and $i_2 = 0$, then $x_2 = x_1$. Note that $x_2 \neq x_1$ or $y_2 \neq y_1$.

If $i_1 \neq 0$ and $i_2 \neq 0$, then $g(g(\ldots.g(x_1)\ldots)) = g(g(\ldots.g(x_2)\ldots))$, which means finding collision

CR vs. WCR (4/5)

- ***h*(*x*, *y*) *is not collision resistant.***

  Because $h(x, g(y))$ is **often** equal to $h(g(x), y)$.



$$h(x, g(y)) = (g^{(6)}(x), g^{(3)}(y), 6+2)$$

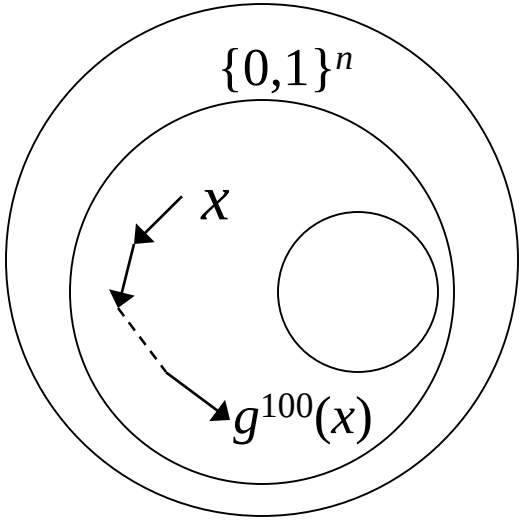
$$h(g(x), y) = (g^{(6)}(x), g^{(3)}(y), 5+3)$$

CR vs. WCR (5/5)

- Why **often and not always**?
  - Case 1: If $x \in \{0, 1\}^n$ and ends with 4 zeros and $g(x)$ ends with 4 zeros

    then, $h(x, g(y)) = (x, g^{(j)}(y), 0+j-1) \neq (g(x), g^{(j)}(y), 0+j) = h(g(x), y)$

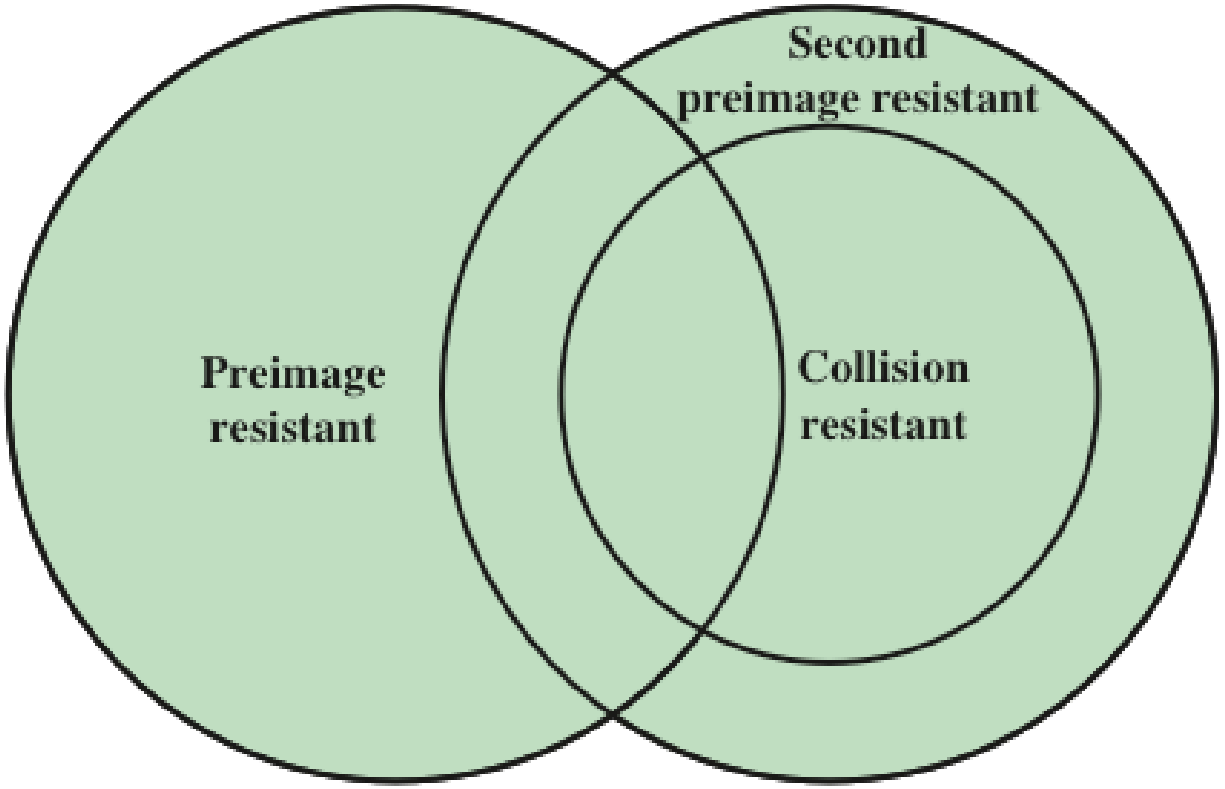
  - Case 2:for x, hashing 100 times still never get a value ending with 0000.



The condition of 4 zeroes is variable, you can choose 2 or 3 or 5 … zeroes.

Also the condition to apply function g a maximum of 100 times is again variable. You can select 10, 200,….

Relationship among Hash Function Properties



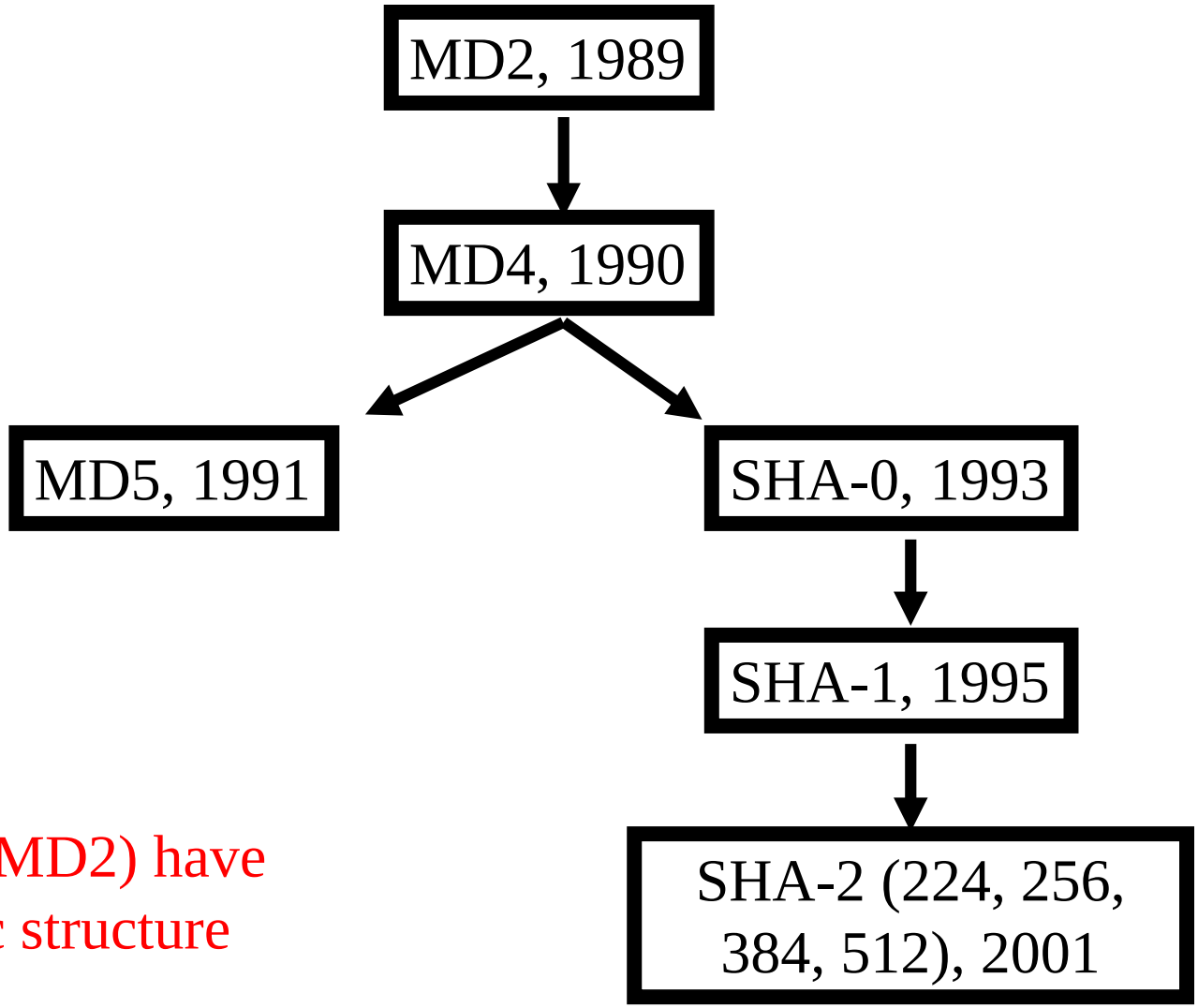
Which Property Do We Need?

- UNIX passwords stored as hash(password)
  - One-wayness: hard to recover password
- Integrity of software distribution
  - Weak collision resistance
- Intrusion Detection
  - For each file *F*, store *h*(*F*), for example on a CD-R. Keep the CD-R secure. Later, one can see if the file has been modified by recomputing *h*(*F*) for each file.
  - Needs weak collision resistance
- Auction bidding
  - Alice wants to bid *B*, sends *H*(*B*), later reveals *B*
  - One-wayness: rival bidders should not recover *B*
  - Collision resistance: Alice should not be able to change her mind to bid *B'* such that *H*(*B*)=*H*(*B'*)

Hash Function Genealogy (1/2)

- Ron Rivest ***Message Digest*** MD-family (***MD2, MD4*** and ***MD5***): 128-bit
- NIST ***Secure Hash Algorithm SHA***.
  - SHA designed by NIST & NSA in 1993 and revised in 1995 as SHA-1
    - SHA-1 is based on design of MD4 and produces 160-bit hash values
  - Regarding its use in future applications, NIST added 3 additional versions of SHA in 2002
    - SHA-256, SHA-384, SHA-512
    - Structure & detail is similar to SHA-1
    - Security levels are **rather higher**
- They take an *arbitrary-length* string and map it to a *fixed-length* quantity that appears to be randomly chosen.
  For example, two inputs that differ by only one bit should have outputs that look like completely independently chosen random numbers.

Hash Function Genealogy (2/2)



All (except MD2) have same basic structure

Common Hash Functions

- MD5
  - 128-bit output
  - Designed by Ron Rivest, used very widely
  - Specified as Internet standard RFC1321
  - Collision-resistance broken (summer of 2004)

- SHA-1
  - 160-bit output
  - Input length: at most 2^{64} bits
  - Developed by NIST, specified in the Secure Hash Standard (SHS, FIPS Pub 180), 1993
  - SHA is specified as the hash algorithm in the Digital Signature Standard (DSS) by NIST

SHA-1 vs. MD5

- Brute force attack is harder for SHA-1(160 vs 128 bits for MD5)
- SHA-1 is not vulnerable to any known cryptanalytic attacks (compared to MD5)
- SHA-1 is a little slower than MD5
- Both designed as simple and compact for implementation

Revised Secure Hash Standard

- NIST have issued a revision FIPS 180-2
- Added 3 additional hash algorithms
  - SHA-256, SHA-384, SHA-512
- Designed for compatibility with increased security provided by the AES cipher
- Structure & detail are similar to SHA-1

How Strong is SHA-1?

- Every bit of output depends on every bit of input
  - Very important property for collision-resistance

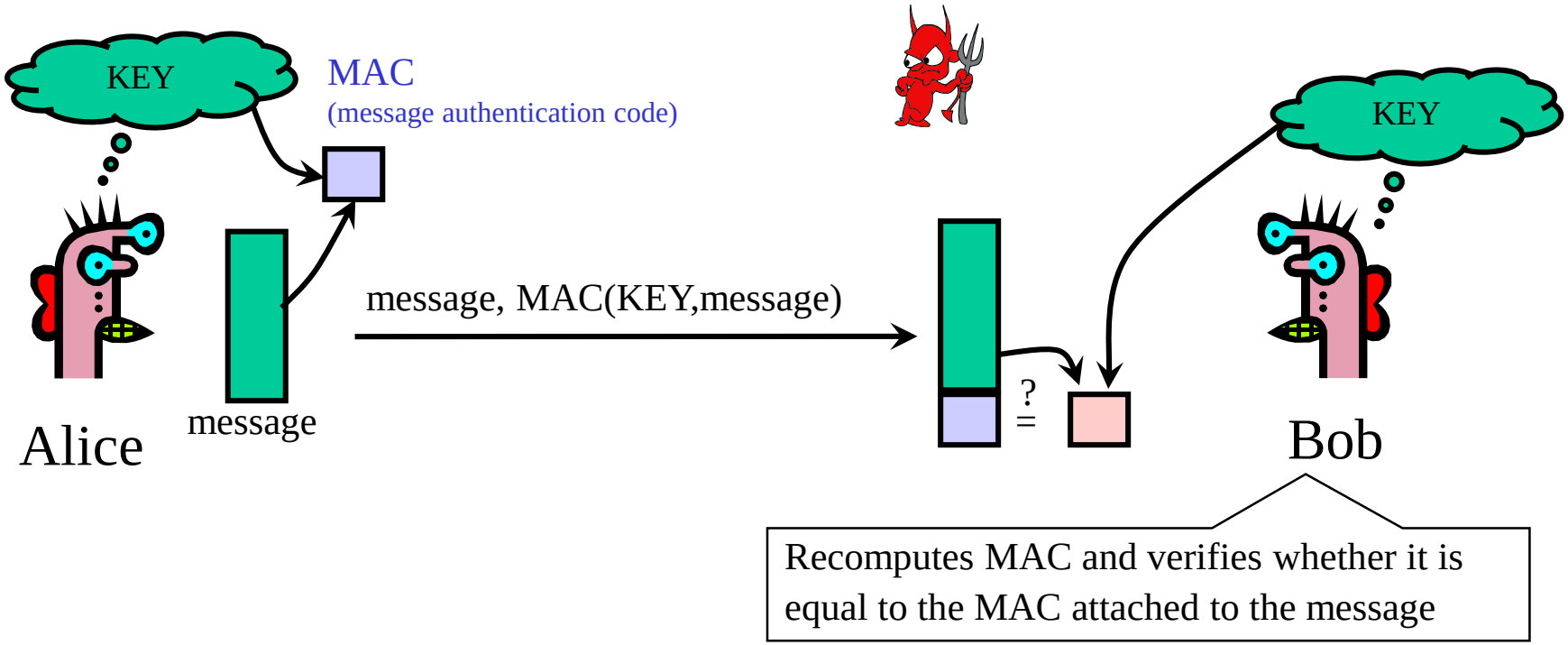
- Brute-force inversion requires 2^{160} ops, birthday attack on collision resistance requires 2^{80} ops

- Some very recent weaknesses (2005)
  - Collisions can be found in 2^{63} ops

How Many Bits for Hash?

- If the message digest is m bits, then $k=2^m$
- We need to examine $2^{m/2}$ to likely find two with the same hash
- If $m = 128$, takes 2^{64} messages to search
- Need at least 128 bits

Authentication Without Encryption



Integrity and authentication: only someone who knows KEY can compute MAC for a given message

Message Authentication Code with Hash

- Message Authentication Code (MAC)
  - It can be also used to protect the integrity of a message
- Can we just compute $h(m)$?
  - No Authentication
- Several choices to compute MAC with a key
  - Prefix: $\text{MAC}_k(m) = h(k|m)$
    - It is not secure
      - The hashes are usually computed by repeatedly hashing blocks and combining with previously computed value: $h(h(h(k|m_1)|m_2)\ |m_3)$
      - Attackers can append to the message without knowing key k
  - Suffix: $\text{MAC}_k(m) = h(m|k)$
  - Envelope: $\text{MAC}_k(m) = h(k_1|m|k_2)$
  - HMAC: $\text{MAC}_k(m) = h(k_1|h(m|k_2))$
    - provably secure; slower, popular in Internet standards.

Encryption with Hash

- One-time pad
  - compute bit streams using h, k, and IV
    - $b_1 = h(k|\text{IV})$, $b_i = h(k|b_{i-1})$, …
  - $\oplus$ with message blocks

- Or mixing in the plaintext
  - similar to cipher feedback mode (CFB)
    - $b_1 = h(k|\text{IV})$, $c_1 = p_1 \oplus b_1$
    - $b_2 = h(k|c_1)$, $c_2 = p_2 \oplus b_2$

Hash Algorithm Structure

- This structure is referred to as an iterated hash function
  - It takes an input message and partitions it into L b-bit blocks. The final block is padded to b bits.
  - It repeatedly uses a compression function, f, that takes two inputs: an n-bit input from the previous step and a b-bit block, and produces an n-bit output.

$$\begin{cases} CV_0 = IV = \text{initial } n\text{-bit value} \\ CV_i = f(CV_{i-1}, Y_{L-1}) \\ H(M) = CV_L \end{cases}$$

- Theorem: If f is collision resistant, then H is collision resistant.