HASH FUNCTIONS AND APPLICATIONS

QUESTION: YOU HOLD A LARGE FILE F1.ZIP
I ALSO HOLD A LARGE FILE F2.ZIP.

HOW CAN WE CHECK IF THE TWO FILES ARE IDENTICAL?

SOLUTION 1: YOU JUST SEND ME YOUR FILE AND I'LL CHECK.

DRWBACK: VERY INEFFICIENT IF THE FILES ARE BIG.

SOLUTION 2: YOU JUST SEND THE LAST 1KB OF YOUR FILE.

PROBLEM: TWO DIFFERENT FILES MIGHT END IN EXACTLY THE SAME WAY.

SOLUTION 3: SEND SOME ALGORITHM H APPLIED TO THE FILE.

WHAT PROPERTIES DO WE WANT FROM H?

1) THE OUTPUT OF H IS SMALL (SAY 256 BITS) ← COMPRESSING

2) $H(F1) = H(F2)$ IF $F1 = F2$.

SO H IS DETERMINISTIC.

3) $H(F1) \neq H(F2)$ IF $F1 \neq F2$.

CAN WE EVER HAVE (1) + (3) TOGETHER?

Q1: HOW MANY POSSIBLE 1000-BIT FILES ARE THERE? 2^{1000}

Q2: IF H-VALUES ARE 256-BITS LONG, THERE ARE 2^{256} POSSIBLE VALUES.

BUT $2^{1000} \gg 2^{256}$. SOME H-VALUES GET HIT MORE THAN ONCE.

LOWER THE STANDARD SLIGHTLY:

(3) IT IS PRACTICALLY IMPOSSIBLE TO COME UP WITH F1 AND F2 SUCH THAT

$$F1 \neq F2 \wedge H(F1) = H(F2).$$

QUESTION: WHY NOT CONSIDER THIS NOTION:

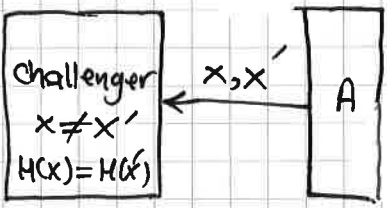
(3)' FOR RANDOM F1, F2 $H(F1) \neq H(F2)$?

ANSWER: FILES / DATA ARE NOT RANDOM. AND HAVE STRUCTURE.

LET'S FORMALIZE THIS.

DEFINITION: A **HASH FUNCTION** WITH ℓ-BIT OUTPUTS IS AN ALGORITHM H THAT TAKES AN ARBITRARY STRING $x \in \{0,1\}^*$ (SET OF ALL STRINGS) AND OUTPUTS AN ℓ-BIT STRING $h \in \{0,1\}^\ell$.

A HASH FUNCTION H IS CALLED COLLISION-RESISTANT IF NO EFFICIENT ADVERSARY A CAN WIN THE GAME:



THAT IS, FIND **DIFFERENT** x, x' **THAT HASH TO THE SAME VALUE**.

NOTE: SOMETIMES HASH FUNCTIONS ARE **KEYED**. THAT IS, H TAKES A KEY $K \in \{0,1\}^k$ AND $x \in \{0,1\}^*$ AND OUTPUTS $h \in \{0,1\}^\ell$. WE CAN CONVERT AN (UNKEYED) HASH FUNCTION TO A KEYED ONE BY PREPENDING THE KEY TO THE INPUT, i.e., COMPUTE $H(K \| x)$.

QUESTIONS: WHICH OF THE FOLLOWING ARE COLLISION RESISTANT?

1) $H(X) :=$ MOST-SIGNIFICANT-BIT(X)

2) $H(X) := X \pmod{47}$

3) $H(x) := x_1 \oplus x_2 \oplus \cdots \oplus x_n \quad X = x_1 \cdots x_n$

4) $H(X) := X^3 \bmod P \quad P$ PRIME

5) $H(X) := X$

6) $H(x) :=$ OUTPUT A RANDOM VALUE.

ANSWERS:

1) NO: TAKE $X = 10$, $X' = 11$ THEN $H(10) = H(11) = 1$.
2) NO: TAKE $X = 0$, $X' = 47$ THEN $H(X) = H(X') = 0$
3) NO: TAKE $X = 0 \cdots 0$, $X' = 1 \cdots 1$, n even.
THEN $H(X) = H(X') = 0$.
4) YES, BUT NOT COMPRESSING.
5) YES, BUT NOT COMPRESSING.
6) YES, BUT NOT DETERMINISTIC.

TWO WEAKER NOTIONS OF SECURITY

SOME APPLICATIONS DO NOT REQUIRE THE FULL POWER OF COLLISION RESISTANCE. INSTEAD THEY RELY ON WEAKER REQUIREMENTS.

1) 2ND PRE-IMAGE RESISTANCE (A.K.A. TARGET-COLLISION RESISTANCE):

GIVE A RANDOM x, FIND AN $x' \neq x$ THAT COLLIDES, i.e., $H(x) = H(x')$.

2) PRE-IMAGE RESISTANCE:

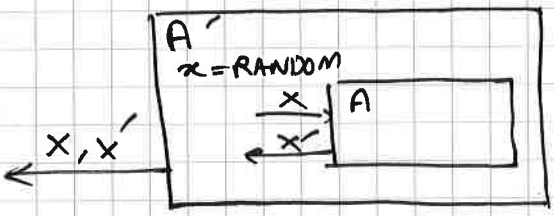
GIVEN A RANDOM y, FIND AN x SUCH THAT $H(x) = y$.

QUESTIONS

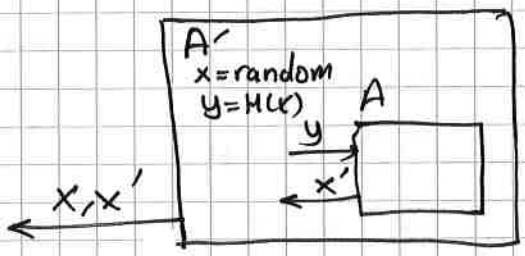
IS EVERY COLLISION-RESISTANT (CR) HASH FUNCTION ALSO { 2ND PRE-IMAGE RESISTANT? / PRE-IMAGE RESISTANT? }

ANSWERS: YES & YES.

IF GIVEN A RANDOM x A CAN FIND AN $x' \neq x$ S.T. $H(x) = H(x')$, CONSIDER A' AS FOLLOWS

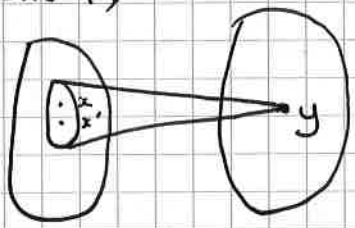


FOR PRE-IMAGE RESISTANCE



IF A IS SUCCESSFUL $H(x') = y = H(x)$.
BUT IS $x \neq x'$?

IF H IS COMPRESSING;



A HAS NO IDEA "WHICH x" GAVE RISE TO y.
SO IT WILL OUTPUT AN x' WHICH IS DIFFERENT THAN x (CHOSEN BY A') WITH GOOD CHANCE.

(THINK OF IT LIKE THIS: LET $H(x) =$ LAST-2-BITS(x)

I CHOOSE A RANDOM x, SAY $x = 5678$

I GIVE A, $H(x) = 78$, A HAS NO IDEA IF $x = 5678$, or 1278 or 1378. ALL IT SEES IS 78.).

BUILDING HASH FUNCTIONS

WE WANT TO BUILD A HASH FUNCTION THAT CAN TAKE AS INPUT ARBITRARY-LENGTH INPUTS, THAT IS:

$$H: \{0,1\}^* \longrightarrow \{0,1\}^\ell$$

THIS IS OFTEN DONE BY FIRST DESGING A HASH (OR COMPRESSION) FONCTION

$$f: \{0,1\}^{2\ell} \longrightarrow \{0,1\}^\ell$$

AND THEN EXTENDING THE DOMAIN FROM $\{0,1\}^{2\ell}$ TO $\{0,1\}^*$. SO

1) HOW DO WE EXTEND THE DOMAIN?
2) HOW DO WE DESGIN f?

1) DOMAIN EXTENSION

A POPULAR APPROACH IS USING THE MERKLE-DAMGÅRD TRANFORM.

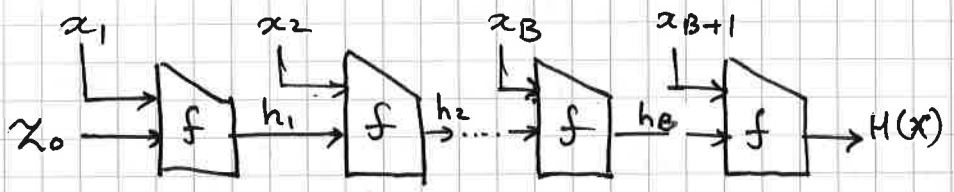
SUPPOSE WE WANT TO HASH A LONG MESSAGE $x \in \{0,1\}^*$. FIRST WRITE x AS

$$x = x_1 x_2 \cdots x_B \leftarrow \#\text{ OF BLOCKS}$$

WHERE $|x_i| = \ell$ AND $B = \lceil \frac{L}{n} \rceil$

PAD x_B WITH ZEROS.

NEXT CHAIN HASH VALUES LIKE THIS:



WHAT'S z_0? IT'S SET TO A CONSTANT VALUE, E.G. 0^n. (OR TO THE KEY IF HASH FUNCTION IS KEYED)

WHAT ABOUT x_{B+1}? $x_{B+1} = L$ WHE L IS THE LENGTH OF THE MESSAGE. (ASSUME $L < 2^l$).

x_B & L ARE PADDED WITH ZEORS TO BE EXACTLY m-BITS LONG.

WHY IS THIS TRANSFORMATION GOOD?
WE NEED TO SHOW IF f IS 'COLLISION-RESISTANT' THEN H IS ALSO COLLISION RESISTANT.

SUPPOSE H IS NOT COLLISION RESISTANT.

THIS MEANS THERE IS SOME ALGORITHM A THAT OUTPUTS (x, x') SUCH THAT $x \neq x'$ BUT $H(x) = H(x')$. WRITE $\begin{cases} x = x_1, \ldots, x_{B+1} \\ x' = x'_1, \ldots x'_{B+1} \end{cases}$

CASE 1: x, x' HAVE DIFFERENT LENGTHS SAY L, L'. (I.E., $L \neq L'$)

BUT IF $L \neq L'$ THEN $x_{B+1} \neq x'_{B+1}$

BUT THEN $f(h_B, x_{B+1}) = H(x)$
$f(h'_B, x'_{B+1}) = H(x')$

RECALL: (x, x') IS A COLLISION $\Rightarrow H(x) = H(x')$.

SO $f(h_B, x_{B+1}) = f(h'_B, x'_{B+1})$

NOW $(h_B, x_{B+1}) \neq (h'_B, x'_{B+1})$ SINCE $x_{B+1} \neq x'_{B+1}$

SO THIS IS A COLLISION FOR f.

THIS CONTRADICTS THE FACT THAT COLLISIONS ARE HARD TO FIND FOR f.

CASE 2: $x \neq x'$, $H(x) = H(x')$, $L = L'$ (SAME LENGTH)

THEN WE CAN WORK BACKWARDS AS FOLLOWS.

2.1) $(x_{B+1} = x'_{B+1})$, $h_B \neq h'_B \Rightarrow$ COLLISION FOR f FOUND

. WHY? $f(h_B, x_{B+1}) = H(x)$ $\wedge$ $(h_B, x_{B+1}) \neq (h'_B, x'_{B+1})$

$f(h'_B, x'_{B+1}) = H(x')$

2.2) IF $h_B = h'_B$ THEN:

$\begin{cases} x_B \neq x'_B \Rightarrow \text{COLLISION FOR } f \\ h_{B-1} \neq h'_{B-1} \Rightarrow \text{COLLISION FOR } f. \end{cases}$ $f(x_B, h_{B-1}) = f(x'_B, h_{B-1}) = h_B$

2.3) IF $h_{B-1} = h'_{B-1}$

$\begin{cases} x_{B-1} \neq x_{B-1} \Rightarrow \text{COLLISION} \\ h_{B-2} \neq h_{B-2} \Rightarrow \text{COLLISION} \end{cases}$

⋮

BUT SINCE $x \neq x'$ AT SOME POINT WE MUST HAVE THAT $x_i \neq x'_i$, IN EXACTLY THAT POINT WE GET A COLLISION.

QUESTION: WHAT HAPPENS IF WE DONT INCLUDE L?

2) THE COMPRESSION FUNCTION f.

ONE IDEA TO DESIGN f WOULD BE TO START WITH A BLOCKCIPHER (E, D) AND TRY TO CONVERT IT TO A COMPRESSION FUNCTION.

THE MOST SIMPLE IDEA IS:

$$f(x_1 | x_2) := E(x_1, x_2)$$

(KEY, MESSAGE)

THIS DOES NOT WORK.

WHY? TO FIND A COLLISION TAKE ANY VALUE, SAY 0^n; CHOOSE ANY TWO KEYS K, K' AND COMPUTE

$$x = D(K, 0^n)$$
$$x' = D(K', 0^n)$$

NOW (K, x) AND (K', x') is a COLLISION FOR f.

WHY?

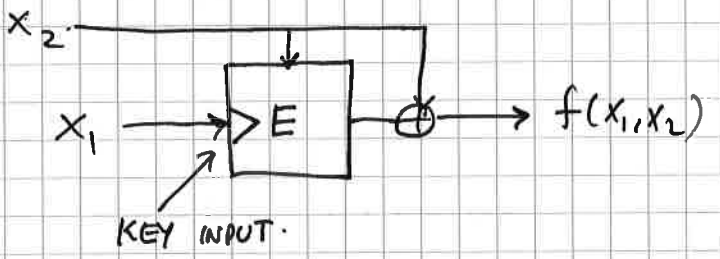
$$f(K, x) = E(K, D(K, 0)) = 0$$
$$f(K', x') = E(K', D(K', 0)) = 0$$

(RECALL: $E(k, D(k, x)) = D(k, E(k, x)) = x$ FOR A BLOCKCIPHER

SLIGHTLY CHANGING THIS IDEA WORKS.

IN THE DAVIES-MEYER CONSTRUCTION WE SET:

$$f(x_1, x_2) = E(x_1, x_2) \oplus x_2$$



ONE CAN SHOW THAT IF THE OUTPUTS OF E ARE SUFFICIENTLY RANDOM, THE f IS COLLISION RESISTANT.

THERE ARE OTHER BLOCKCIPHER-BASED COMPRESSION FUNCTIONS. ONE CALLED MIYAGUCHI-PRENEEL IS:

$$f(x_1, x_2) = E(x_1, x_2) \oplus x_1 \oplus x_2$$

THIS ONE IS COLLISION-RESISTANT TOO. THERE ARE IN TOTAL 20 SUCH CONSTRUCTIONS, 12 OF WHICH HAS GOOD SECURITY.

QUESTION: WHAT ABOUT (ASSUME KEY-LENGTH = BLOCK-LENGTH)

$$f(x_1, x_2) = E(x_1, x_2) \oplus x_1 \; ?$$

ANSWER: NOT COLLISION RESISTANT:

1) CHOOSE (k,x) AND k' WITH $k' \neq k$.

2) LET $h := E(k,x) \oplus k$

3) LET $y := h \oplus k'$

4) LET $x' := D(k',y)$.

THEN $f(k',x') = E(k',x') \oplus k'$

$= E(k', D(k',y)) \oplus k'$

$= y \oplus k'$

$= (h \oplus k') \oplus k'$

$= h$

$= E(k,x) \oplus k$

$= f(k,x)$

SO $((k,x),(k',x'))$ IS A COLLISION FOR f.

SPECIFIC CONSTRUCTIONS

MD5:
- DESIGNED IN 1991.
- WAS BELIEVED TO BE COLLISION RESISTANT
- OVER SEVERAL YEARS WEAKNESSES BEGAN TO APPEAR.
- IN 2004, A GROUP OF CHINESE CRYPTANALYSTS FOUND A COLLISION!
- ONE CAN EVEN FIND "MEANINGFUL" COLLISIONS. (EG. TWO PDF FILES WITH SAME HASH).

SHA: SECURE HASH ALGORITHM

A SERIES OF HASH FUNCTIONS STANDARDIZED BY NIST (NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY)

SHA-1: INTRODUCED IN 1995.

160-BIT OUTPUT

WEAKNESSES DISCOVERED

IN FEB 2017 COLLISIONS FOUND! (6500 CPU YEARS, 2^{63} SHA-1 COMPUTATIONS).

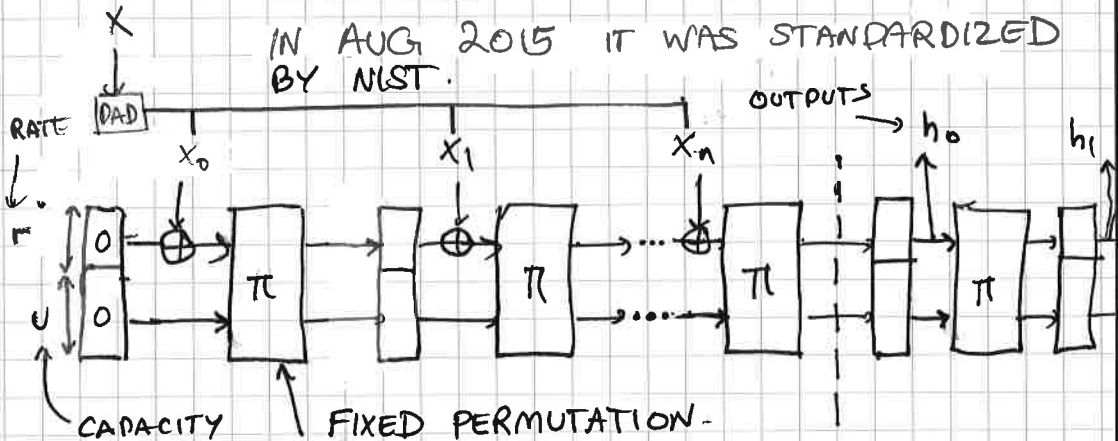
SHA-2: DOES NOT APPEAR TO HAVE THE SAME WEAKNESSES

BUT STILL SIMILAR.

SHA-3 : AFTER THE MD5 ATTACK AND WEAKNESSES (NOW INSECURITY!) OF SHA1 NIST IN 2007 ANNOUNCED A PUBLIC COMPETITION TO DESIGN A NEW HASH FUNCTION.

- 51 FIRST-ROUND CANDIDATES
- 14 SECON-ROUND CANDIDATES 2008/12.
- 5 FINALISTS (2010)
- IN OCTOBER 2012, NIST ANNOUNCED **KECCAK** AS THE WINNER.

IN AUG 2015 IT WAS STANDARDIZED BY NIST.

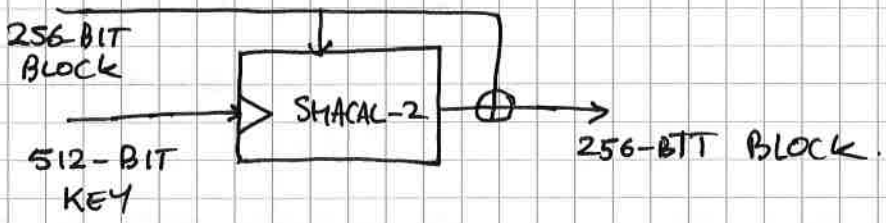


SECURITY: THE LARGER THE c THE BETTER.

EFFICIENCY: THE LARGER THE r THE BETTER

SHA-2 IS A FAMILY OF HASH FUNCTIONS WITH DIFFERENT HASH LENGTHS. WHICH INCLUDE SHA-256 AND SHA-512.

SHA-256 FOLLOWS DAVIES-MEYER WITH A BLOCKCIPHER CALLED SHACAL-2.



PROCESSES 512 BITS AT A TIME.

BIRTHDAY ATTACK FOR FINDING COLLISIONS

LET

$$H: \{0,1\}^* \longrightarrow \{0,1\}^\ell$$

BE A HASH FUNCTION.

SAY WE CHOOSE q DISTINCT INPUTS

$$x_1, x_2, \ldots, x_q$$

AND COMPUTE

$$y_i := H(x_i) \qquad i = 1, \ldots, q.$$

WHAT IS THE PROBABILITY THAT WE FIND A COLLISION? THAT IS, FOR SOME $i \neq j$ $y_i = y_j$?

LET US ASSUME THAT THE OUTPUTS OF H ARE RANDOM. THEN WE ARE LOOKING AT THE PROBABILITY THAT FOR SOME $i \neq j$ $1 \leq i, j \leq q$, $y_i = y_j$ WHERE y_i ARE RANDOMLY SAMPLED FROM $\{0,1\}^\ell$.

FOR EACH (i,j) $\Pr[y_i = y_j] = \frac{1}{2^\ell}$

THERE ARE $\frac{q(q-1)}{2!}$ SUCH PAIRS

SO THE PROBABILITY THAT FOR SOME i,j: $y_i = y_j$ IS $\approx \frac{1}{2^\ell} + \frac{1}{2^\ell} + \cdots + \frac{1}{2^\ell} \approx \frac{q(q-1)}{2 \cdot 2^\ell}$

THIS MEANS THAT AFTER ABOUT $q=\sqrt{2^\ell}=2^{\ell/2}$ QUERIES WE EXPECT A COLLISION SINCE THEN $\frac{q(q-1)}{2\cdot 2^\ell} \approx \frac{1}{2}$.

CONCLUSION: A HASH FUNCTION WITH ℓ-BIT OUTPUTS CAN HAVE COLLISION RESISTANCE UPTO $2^{\ell/2}$ HASH VALUES. THAT IS, IT HAS $\ell/2$-BIT COLLISION SECURITY.

WHAT ABOUT PRE-IMAGE RESISTANCE? RECALL THIS SAYS GIVEN y, FIND AN x SUCH THAT

$$H(x)=y$$

SUPPOSE WE DO q HASH COMPUTATIONS THE FOR EACH i $\Pr[H(x_i)=y] = \frac{1}{2^\ell}$ AND AFTER q TRIES WE GET PROBABLITY

$$\approx \underbrace{\frac{1}{2^\ell}+\cdots+\frac{1}{2^\ell}}_{q} = \frac{q}{2^\ell}.$$

SO WE GET SECURITY UPTO $q=2^\ell$ HASH COMPUTATIONS, MUCH HIGHER THAN COLLISION RESISTANCE

NOTE ~~THAT~~ WITH THE BIRTHDAY ATTACK WE CAN FIND MEANINGFUL COLLISIONS:

IT IS HARD/DIFFICULT/CHALLENGING/IMPOSSIBLE (4) TO IMAGINE/BELIEVE (2) THAT WE WILL FIND/LOCATE/HIRE (3) ANOTHER EMPLOYEE/PERSON (2) HAVING SIMILAR ABILITES/SKILLS/CHARACTER AS ALICE. SHE HAS DONE A GREAT/SUPERB (2) JOB

$4 \times 2 \times 3 \times 2 \times 3 \times 2 = 288$ DIFFERENT LETTERS.

QUESTION: WHAT HAPPENS IF WE USE AES IN DAVIES-MEYER?

THE BLOCKLENGTH IS 128-BITS.

THIS MEANS COLLISION SECURITY IS ONLY $2^{64} = 2^{128/2}$ WHICH IS NOT SUFFICIENT.

A FEW MORE APPLICATIONS

- **DEDUPLICATION IN CLOUD STORAGE:**
  - USER FIRST UPLOADS A HASH OF THE FILE HE/SHE WANTS TO UPLOAD. IF A FILE WITH THIS HASH ALREADY EXISTS THE PROVIDER SIMPLY ADDS A POINTER TO IT.

- **P2P FILE SHARING**

  AGAIN WE CAN USE HASH FUNCTIONS TO IMPLEMENT A FILE LOOK-UP SERVICE.

- **PASSWORD HASHING**

  SUPPOSE YOU WANT TO LOGIN TO YOUR COMPUTER WITH YOUR PASSWORD.

1) ONE IDEA WOULD BE TO STORE THE PASSWORD ON YOUR LAPTOP AND WHEN YOU LOGIN LATER, COMPARE VALUES. BUT IF YOUR LAPTOP IS STOLEN SO IS YOUR PASSWORD.

2) INSTEAD WE CAN STORE HASH OF THE PASSWORD AND CHECK EQUALITY OF HASH VALUES.

$$H(pw) \stackrel{?}{=} hpw$$

↑ ENTERED. ↑ ON LAPTOP

ONE PROBLEM WITH PASSWORDS IS THAT THEY ARE CHOSEN FROM A RELATIVELY SMALL SPACE. $pw_1, pw_2, \ldots \in D \quad |D| = 2^{50}$

AN ATTACKER CAN DO A ONE-TIME CALCULATION OF ALL PASSWORDS AND USE THIS TO INVERT h_{pw}.

TWO DEFENSES AGAINST THIS:

1) MAKE HASH SLOWER:

COMPUTE $\underbrace{H(H(H(\cdots(pw))\cdots))}_{I \text{ TIMES}}$.

$I \approx 1000$ or 10^6 : HASHING TAKES JUST A BIT MORE TIME.

BUT FOR THE ATTACKER, IT HAS TO DO 10^6 TIMES MORE WORK (WEAKS $\rightsquigarrow$ YEARS).

2) STORE (s, h_{pw}) WHERE

$$h_{pw} = H(s, pw)$$

FOR A RANDOM VALUE s. ← ONE-TIME VALUE IN REGISTRATION.

THE ATTACKER HAS TO DO ONE PRE-COMP. FOR EACH S.

- KEY DERIVATION

SUPPOSE YOU HAVE AGREED ON A KEY k WITH A FRIEND. THIS KEY HAS A LOT OF RANDOMNESS, BUT IS NOT UNIFORM.

1) YOU CAN SMOOTHEN THE RANDOMNESS in K BY HASHING IT.

2) YOU CAN DERIVE MANY KEYS FROM k (WHICH CAN BE USED FOR DIFFERENT PURPOSES).

$H(K, 1)$
$H(k, 2)$
$\vdots$

NOTE FOR (1) & (2) COLLISION-RESISTANCE IS NOT ENOUGH. WE NEED THE HASH FUNCTION TO HAVE "RANDOM-LOOKING" OUTPUTS.

MESSAGE AUTHENTICATION

① SUPPOSE ALICE AND BOB SHARE A KEY K. SOMEONE APPEARS ONLINE AND AND CLAIMS TO BE BOB TO ALICE.

HOW CAN ALICE CHECK THIS?

1) ALICE ASKS BOB TO SEND HER K.

OK: BUT K CAN NO LONGER BE USED.

2) ALICE ASKS BOB TO SEND HER H(K).

OK: BUT AGAIN CANNOT AUTHENTICATE AGAIN: JUST RESEND H(K).

3). ALICE SENDS BOB A RANDOM VALUE R AND BOB RESPONDS WITH H(K|R)

THIS IS MUCH BETTER.

② SUPPOSE WE HAVE SOME SYSTEM FILES ON OUR COMPUTER. (EVERYBODY KNOWS THEM SO THEY DON'T NEED TO BE SECRET.)

BOT WE ARE WORRIED THAT SOME VIRUS MIGHT MODIFY THEM.

HOW CAN WE SOLVE THIS?

1) STORE HASH VALUES OF THE FILES.
DOES NOT WORK AS THE VIRUS CAN ALSO MODIFY THE HASH VALUES

2) USE A SECRET KEY K TO
STORE HALUES $H(K \| File1), \ldots H(k \| File_n)$
DERIVE KEY K FROM USER PASSWORD
(SO K IS NOT STORED.) WE CAN CHECK VALIDITY USING PW.
NOW FOR VIRUS TO BREAK THIS SYSTEM:
1) IT SEES SOME TAGS $H(K | FILE_i)$
2) IT NEEDS TO COMPUTE $H(K | BAD\text{-}FILE)$
FOR A (NEW) BAD-FILE.
(THIS IS SIMILAR TO PREVIOUS EXAMPLE
WHERE EVE NEED TO COMPUTE $H(K|R)$
FOR A NEW VALUE R).

THE PRIMITIVE USED IN THESE SOLUTIONS IS CALLED A MESSAGE AUTHENTICATION CODE (MAC). IT IS SLIGHTY DIFFERENT FROM HASH FUNCTIONS

1) IT HAS A SECRET KEY K.
2) ITS VALUES ARE UNFORGEABLE WITHOUT K.

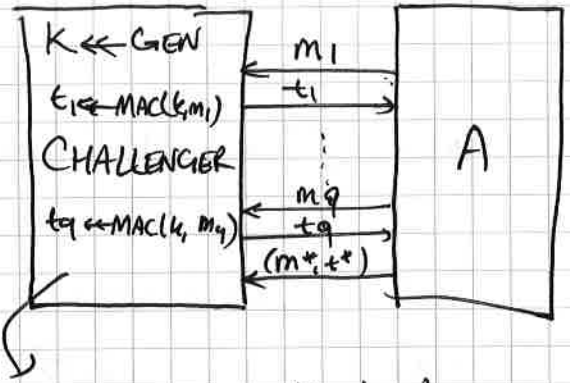
FORMALLY, A MAC IS A TRIPLE OF ALGORITHMS. (Gen, Mac, Verify) WHERE:

1) Gen IS A RANDOMIZED ALGORITHM THAT OUTPUTS A KEY K.
2) Mac (K, M) IS A POSSIBLY RANDOMIZED ALGORITHM THAT OUTPUTS A TAG VALUE t.
3) Verify (K, M, t) : IS DETERMINISTIC., TAKES THE KEY K, MESSAGE M, AND A tag VALUE t, AND OUTPUTS PASS/FAIL.

CORRECTNESS: IF $K \leftarrow$ Gen, $t \leftarrow$ MAC(K, M) FOR ANY MESSAGE M, THEN

VERIFY (K, M, t) = PASS.

SEURITY :



A WINS IF ① $m^* \notin \{m_1, \ldots, m_q\}$.

② Verify (K, m^*, t^*) = PASS

I.E A FORCES A MAC VALUE FOR m^*.

QUESTION : SUPPOSE A MAC SCHEME HAS 10-BIT TAG. VALUES. CAN IT BE SECURE?

HASH-AND-MAC :

SUPPOSE WE HAVE A MAC SMALL-MAC THAT CAN WORK ON SMALL MESSAGES ONLY. CAN WE CONVERT THIS TO A BIG-MAC? {THAT TAKES ARBITRARY LONG MESSAGES}.

CONSTRUCTION: Big-MAC

- DO NOT CHANGE Gen.
- BIG-MAC$(K, M) :=$ SMALL-MAC$(k, H(M))$ WHER $H: \{0,1\}^* \longrightarrow \{0,1\}^l$ is COLLISION RESISTANT.
- BIG-VERIFY(k, m, t).
  RETURN SMALL-VERIFY$(K, H(M), t)$.

WHY IS THIS SECURE?

SUPPOSE SOME A FINDS A VALID (m^*, t^*) (m^* IS LONG).

1) EITHER $H(m^*) = H(m_i)$ FOR SOME i. AND $m^* \neq m_i$ (m^* MUST BE NEW). IN THIS CASE WE HAVE A COLLISION FOR H.
2) $H(m^*) \neq H(m_i)$ FOR ALL i. BUT THEN $(H(m^*), t^*)$ IS A FORGERY (THE "NEW MESSAGE" IS $H(m^*)$).

HOW CAN WE BUILD SMALL-MAC?

IF WE HAVE A WELL-DESIGNED COMPRESSION FONCTION

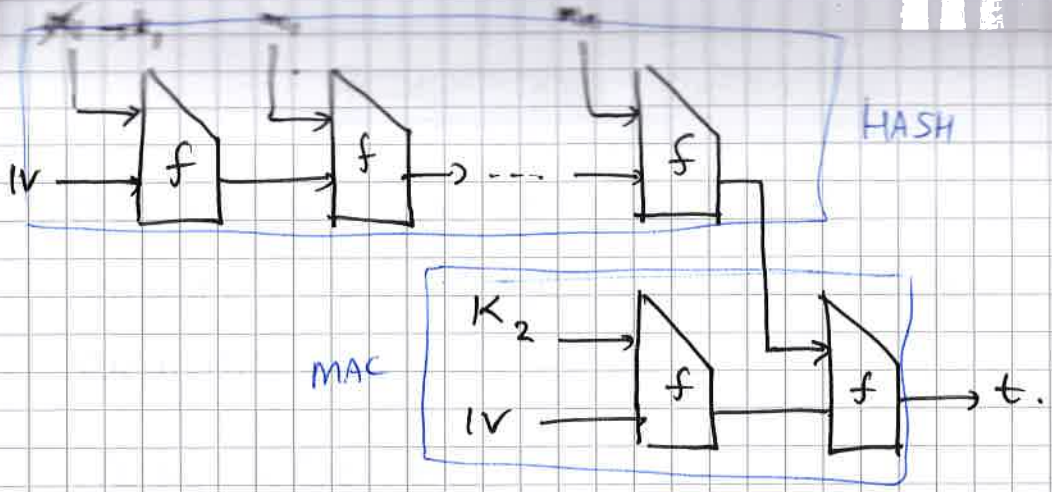
$$f : \{0,1\}^{2\ell} \longrightarrow \{0,1\}^{\ell}$$

WHICH BEHAVES RANDOMLY WE CAN SET

$$MAC(K,M) = f(k \| m).$$

TO GET A MAC. (BEHAVES RANDOMLY MEANS WITHOUT THE KEY NO ALOGRITHM CAN SAY IF SOME PROCEDURE IS RETURNING $f(K \| m)$ FOR MESSAGES m, OR SIMPLY RANDOM STRINGS)

WE CAN NOW UNWRAP HASH-AND-MAC WHERE HASH IS BUILD USING MERKLE-DAMGÅRD. WE GET:



THIS IS THE HASH-MAC (HMAC) CONSTRUCTION.

WE NEED K_2 FOR THE MAC COMPONENT. WE DON'T NEED K_1. BUT INCLUSION OF K_1 MAKES THE CONSTRUCTION STRONGER. INDEED WHEN MD5 WAS BROKEN, K_1 SAVED HMAC, AND GAVE TIME TO REPLACE WITH SHA.

K_1 AND K_2 ARE COMPUTED AS

$K_1 := K \oplus ipad$ (inner pad)

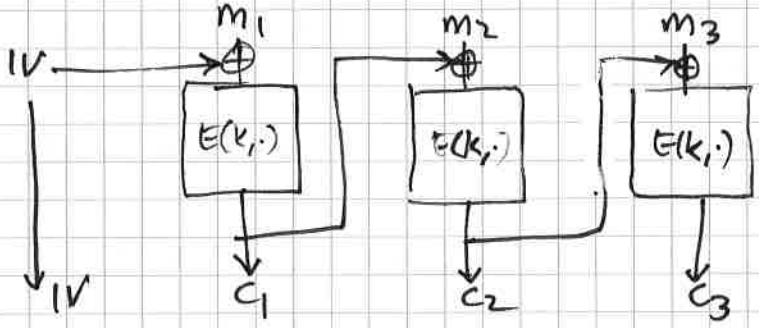
$K_2 := K \oplus opad$ (outer pad)

FOR EFFICIENCY.

AUTHENTICATED ENCRYPTION

WE CAN USE MAC TO PROTECT THE INTEGRITY OF CIPHERTEXTS IN A SYMMETRIC ENCRYPTION.

RECALL THE CBC MODE:



HERE IF WE FLIP SOME BITS IN IV THE RESULT IS THAT SOME BITS IN m_1 WILL GET FLIPPED.

HOW TO FIX THIS?

1) ENCRYPT-AND-MAC

$Enc(k_1, m) + Mac(k_2, m)$ $\quad K = (k_1, k_2)$

DOES NOT WORK

ADVERSARY CAN DETECT FROM $t = MAC(k_2, m)$ IF TWO CIPHERTEXTS ENCRYPT THE SAME MESSAGE. (OR MAC CAN EVEN REVEAL SOME BITS OF M).

2) ENCRYPT-THEN-MAC.

$C \leftarrow Enc(K_1, M)$

$t \leftarrow MAC(k_2, C)$

RETURN (C, t)

PASS/FAIL $\leftarrow$ Verify(K_2, C, t)

IF FAIL RETURN $\perp$.

ELSE $m \leftarrow DEC(K_1, C)$

RETURN m.

THIS WORKS.

AN ENCRPTION SCHEME WHICH IS

(1) IND-CPA SECURE.

(2) HAS UNFORGEABLE CIPHERTEXTS.

IS CALLED AN AUTHENTICATED ENCRYPTION.

AN AUTHENTICATED ENCRYPTION IS AUTOMATICALLY IND-CCA, SECURE SINCE DECRYPTION ORACLE WILL BE USELESS: IT WILL ALWAYS RETURN $\perp$.