Western Norway University of Applied Sciences

Digital Signature and Hashing

DAT159 – Basic Cryptography Module

Tosin Daniel Oyetoyan

Digital Signature

- A digital signature is a mathematical scheme for presenting the authenticity of digital messages or documents
- A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication)
- that the sender cannot deny having sent the message (non-repudiation)
- and that the message was not altered in transit (integrity)

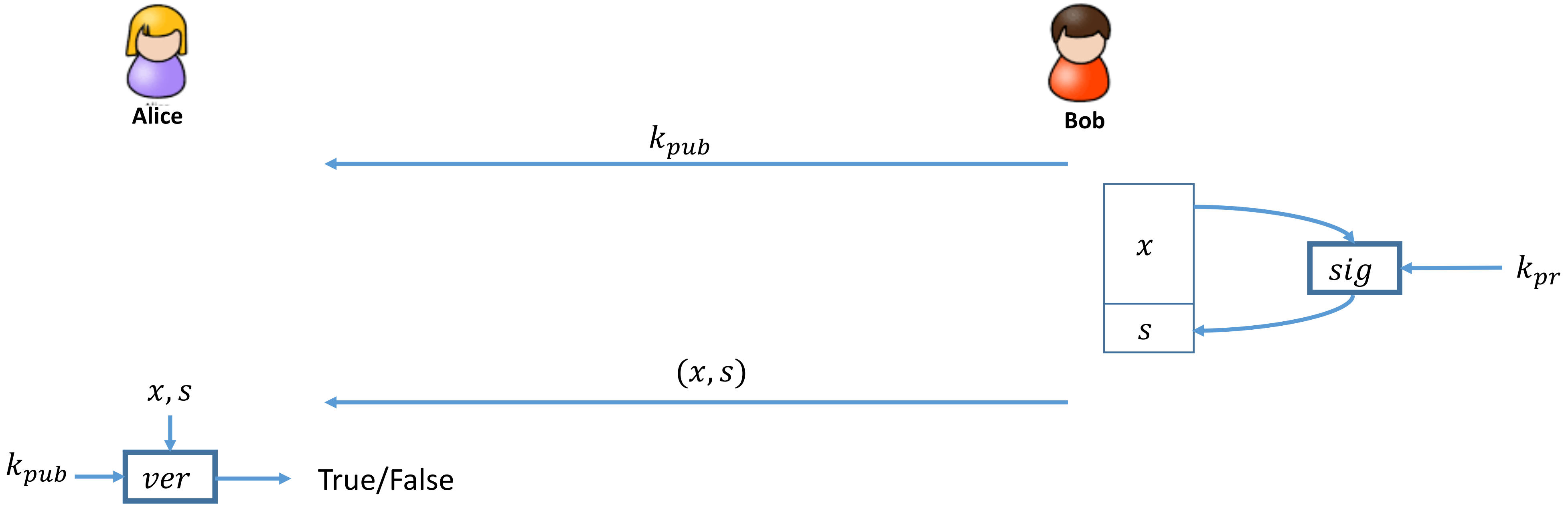
https://en.wikipedia.org/wiki/Digital_signature

Non-Repudiation,Authentication and Integrity

- How do we provide Non-repudiation, authentication and integrity of data
  - Non-repudiation (Digital Signature)
  - Authentication (Digital Signature)
  - Integrity (Digital Signature/Hashing)

Principle of Digital Signature

- Analogous to handwritten signature in the analog world
- Prove that the sender is the actual sender
- Only the person who creates a message should be able to sign it



Digital Signature Protocol

Alice

Bob

Generate k_{prB}, k_{pubB}

$\xleftarrow{k_{pubB}}$

Publish public key (k_{pubB})

Sign message:
$s = sig_{k_{prB}}(x)$

$\xleftarrow{(x, s)}$

Send message + signature

Verify $signature$:
$ver_{k_{pubB}}(x, s) = true/false$

- A signed message can be unambiguosly traced back to its originator
- Note that the above setup does not provide confidentiality

Asymmetric Encryption vs. Digital Signature

- Digital Signature is based on asymmetric cryptography
- In asymmetric encryption, the sender (Bob) uses:
    - The published public key of the intended recipient (Alice) to encrypt message
- The recipient (Alice) uses her private key to decrypt the message

- In digital signature, the sender (Bob) uses:
    - His private key to sign (‘encrypt’) the message intended for the recipient (Alice)
- The recipient (Alice) uses the sender’s (Bob) public key to verify the correctness of the signature.

Examples of Signature Schemes

- RSA Digital Signature
- Elgamal Digital Signature
- Digital Signature Algorithm (DSA) – NIST
- Schnorr Signature Scheme
- Elliptic Curve Digital Signature Algorithm (ECDSA)

RSA Digital Signature

- RSA Keys
  - Bob’s private key: $k_{pr} = (n, d)$
  - Bob’s public key: $k_{pub} = (n, e)$

Alice

Bob

Generate: $k_{pr} = d, k_{pub} = (n, e)$

$$\xleftarrow{\quad (n, e) \quad}$$

Compute signature:

$s = sig_{k_{pr}}(x) \equiv x^d \bmod n$

$$\xleftarrow{\quad (x, s) \quad}$$

Send message + signature

Verify $signature$: $ver_{k_{pubB}}(x, s)$

$x' \equiv s^e \bmod n$

$$x' = \begin{cases} \equiv x \bmod n, & \Longrightarrow \quad valid\ signature \\ \not\equiv x \bmod n, & \Longrightarrow invalid\ signature \end{cases}$$

Proof! (RSA Signature)

$s = x^d mod\ n$

$x' \equiv s^e\ mod\ n$

$x' \equiv (x^d)^e\ mod\ n$

$x' \equiv x^{d.e}\ mod\ n$

$d.e\ \equiv 1\ mod\ n$ (because ***d*** is the inverse of ***e***)

$x' \equiv x\ mod\ n$ (Valid signature).

Example

- Bob wants to send a signed message to Alice.
  - Bob computes his RSA key parameters (Public/Private keys).
  - Then Bob sends Alice his public key.
  - Bob signs the message ***x*** with his private key and sends the message ***x*** and signature ***s*** to Alice.
  - Alice verifies the signature ***s*** using Bob's public key.

Alice

Message x = 4

Bob

1. Choose p=3 and q=11
2. n=p.q=33
3. $\Phi(n)$=(3-1)(11-1)=20
4. Choose e=3
5. $d \equiv e^{-1} \equiv 7 \; mod \; 20$

k_{pub}=(33,3) and k_{pr}=(33,7)

(n, e)

$$\xleftarrow{\qquad k_{pub}=(33,3) \qquad}$$

Compute signature:

s = x^d mod n

= 4^7 mod 33 = 16 mod 33

$$\xleftarrow{\qquad (x,s) = (4,16) \qquad}$$

Verify:

x' = s^e mod n

= 16^3 mod 33 = 4

Elgamal Digital Signature

- **Key Generation** ($k_{pub} = (p, \alpha, \beta)$)
  - Choose a large prime p.
  - Choose a primitive element α of Z_p or a subgroup of Z_p
  - Choose a random integer $d \in \{2,3,\cdots,p-2\}$
  - Compute $\beta = \alpha^d \, mod \, p$

- **Signature Generation** ($sig_{k_{pr}}(x, k_E) = (r, s)$)
  - Choose a random ephemeral key $k_E \in \{0,1,2,\cdots,p-2\}$ such that $\gcd(k_E, p-1) = 1$
  - Compute the signature parameters:
    - $r \equiv \alpha^{k_E} \, mod \, p$
    - $s \equiv (x - d.r)k_E^{-1} mod \, p - 1$

- **Signature Verification** ($ver_{k_{pub}}(x, (r, s))$)
  - Compute the value:
    - $t \equiv \beta^r . r^s \, mod \, p$
  - The verification:
    - $t = \begin{cases} \equiv \alpha^x \, mod \, p, & \Longrightarrow \quad valid \, signature \\ \not\equiv \alpha^x \, mod \, p, & \Longrightarrow invalid \, signature \end{cases}$

Proof! (Elgamal Signature)

$\beta = \alpha^d \; mod \; p$

$r \equiv \alpha^{k_E} \; mod \; p$

$s \equiv (x - d.r)k_E^{-1} mod \; p - 1$

$\boldsymbol{t \equiv \beta^r . r^s \; mod \; p}$

$t = \alpha^{d.r} \; . \; \alpha^{k_E(x-d.r)k_E^{-1}}$

$t = \alpha^{d.r} \; . \; \alpha^{k_E \, .k_E^{-1}(x-d.r)}$

$t = \alpha^{d.r} \; . \; \alpha^{\,(x-d.r)} \quad (k_E \; . \, k_E^{-1} \equiv 1)$

$t \equiv \alpha^{d.r} . \alpha^{\,x} . \alpha^{-d.r}$

$t \equiv \alpha^{d.r-d.r} . \alpha^{\,x}$

$t \equiv \alpha^x \; mod \; p$ (Valid signature).

Challenge with RSA and Elgamal Signature

Digital Signature Algorithm

- Key Generation
  - Generate a prime p with $2^{1023} < p < 2^{1024}$
  - Find a prime divisor q of $p - 1$ with $2^{159} < q < 2^{160}$
  - Find an element α with $ord(\alpha) = q$, i.e. α generates the subgroup with $q\ elements$
  - Choose a random integer d with $0 < d < q$.
  - Compute $\beta = \alpha^{d}\ mod\ p$.
- The keys are:
  - $k_{pub} = (p, q, \alpha, \beta)$
  - $k_{pr} = (d)$

Digital Signature Algorithm

- **DSA Signature Generation** ($sig_{k_{pr}}(x, k_E) = (r, s)$)
  - Choose a random ephemeral key $k_E \in \{0,1,2,\cdots,q-1\}$ such that $\gcd(k_E, p-1) = 1$
  - Compute the signature parameters:
    - $r \equiv (\alpha^{k_E}\ mod\ p)\ mod\ q$
    - $s \equiv (SHA(x) - d.r)k_E^{-1} mod\ q.$

- **DSA Signature Verification** ($ver_{k_{pub}}(x, (r, s))$)
  - Compute auxiliary value $w \equiv s^{-1}\ mod\ q.$
  - Compute auxiliary value $u_1 \equiv w.SHA(x)\ mod\ q.$
  - Compute auxiliary value $u_2 \equiv w.r\ mod\ q.$
  - Compute:
    - $v \equiv (\alpha^{u_1}.\beta^{u_2}\ mod\ p)\ mod\ q$
  - The verification:
    - $v = \begin{cases} \equiv r\ mod\ q, & \Rightarrow\ valid\ signature \\ \not\equiv r\ mod\ q, & \Rightarrow invalid\ signature \end{cases}$

Proof! (Digital Signature Algorithm)

$s \equiv (SHA(x) + d.r).k_E^{-1} mod\ q$

$k_E \equiv s^{-1} SHA(x) + d.s^{-1}.r\ mod\ q$

$k_E \equiv u_1 + d.u_2\ mod\ q$

$\alpha^{k_E}\ mod\ p \equiv \alpha^{u_1 + d.u_2}\ mod\ p$

$\alpha^{k_E}\ mod\ p \equiv \alpha^{u_1}\ \beta^{u_2}\ mod\ p$

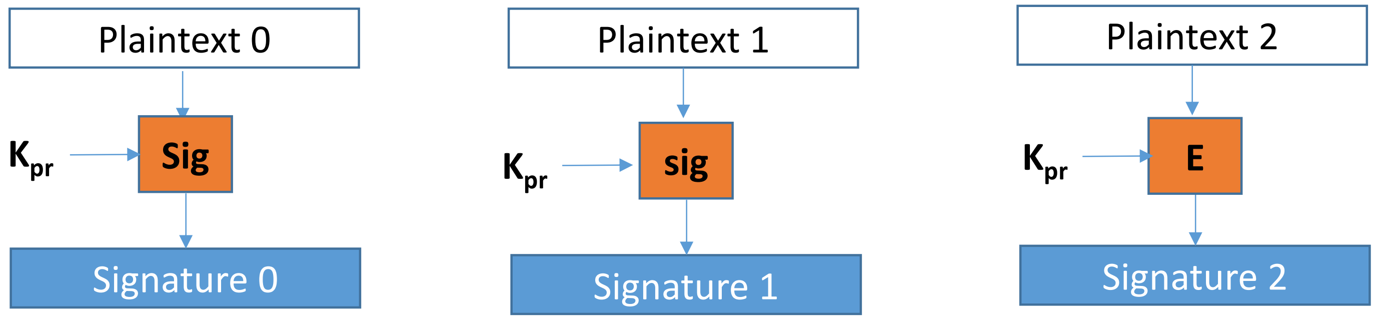
$(\alpha^{k_E}\ mod\ p)\ mod\ q \equiv (\alpha^{u_1}\ \beta^{u_2}\ mod\ p)\ mod\ q$

$r \equiv v\ mod\ q\ \ (Valid\ signature)$

$\beta = \alpha^d\ mod\ p$
$w \equiv s^{-1}\ mod\ q.$
$u_1 \equiv w.SHA(x)\ mod\ q.$
$u_2 \equiv w.r\ mod\ q.$
$v \equiv (\alpha^{u_1}.\beta^{u_2}\ mod\ p)\ mod\ q$
$r \equiv (\alpha^{k_E}\ mod\ p)\ mod\ q$
$s \equiv (SHA(x) - d.r)k_E^{-1} mod\ q.$

Signing Long messages

- Computational load
  - Based on computationally intensive asymmetric operations (e.g. modular exponentiation)
- Message overhead
  - Sends both the message and the signature (appr. Same length)
- Security Limitations
  - Messages and signatures can be tampered with (e.g. rearranging them)



Hash Functions

- Reduce arbitrary-length input (message) to fixed-length (128 or 160bit) output
  - Compression
  - Finger printing
  - Message Digest
- Use in conjuction with Digital signature for long messages
- Hash functions don’t have keys
- Hash functions don’t encrypt

Properties of Cryptographic Hashing Functions

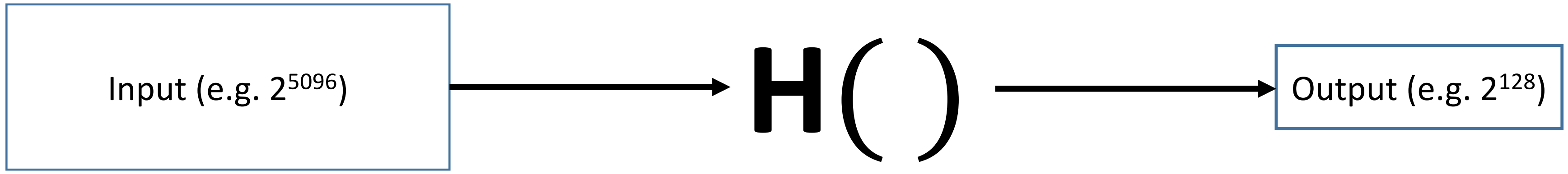
- Computationally efficient
  - To be useful in practice, it has to be fast
- Collision resistant
  - Must not produce the same hash outputs (Theoretically impossible)
- One-wayness
  - Infeasible to generate message from hash
- Infeasible to modify message without modifying the hash
- Infeasible to find two different message with same hash
- Randomness
  - Almost same message must produce two very different hashes
    - e.g. MD5(dat159) = **d782e7777341e93c2a040dfb71a6ba75**
    - **and MD5(Dat159) = f0cc49f6d282171e9f1900a76c83f07d**

Security Reqirements for Hash Functions

- Preimage resistance (one-wayness)
- Second preimage resistance (or weak collision resistance)
- Collision resistance (or strong resistance)

Properties of a Cryptographic Hash Function

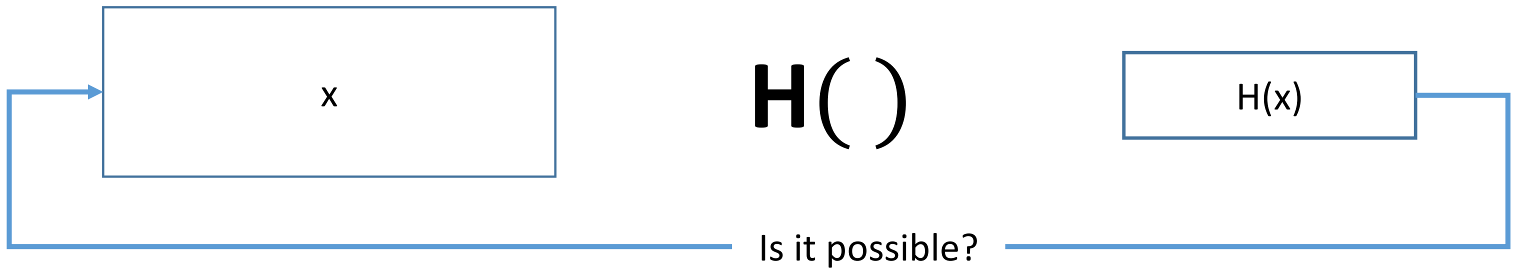
- Can't deduce input from output (Preimage resistance)
- Can't generate a given output (Weak collision resistance)
- Can't find two inputs which produce the same output (Strong collision resistance)
- A given input always produces the same output



Preimage Resistance

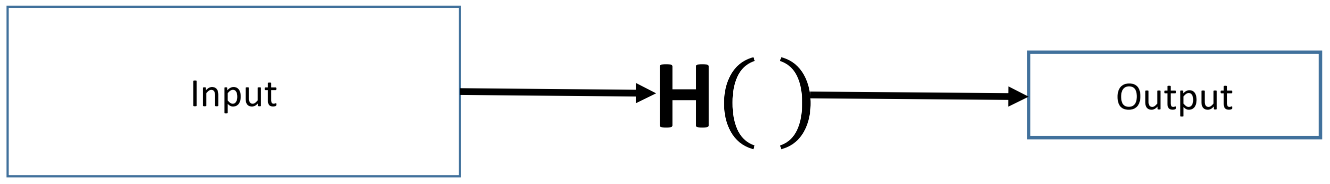
- One-wayness
  - Can we derive a message from its hash output?
- Given a hash output ***z*** it must be computationally infeasible to find an input message ***x*** such that
  - **z = h(x)**
  - Given a fingerprint, we cannot derive a matching message
  - i.e. $x \neq h^{-1}(x)$

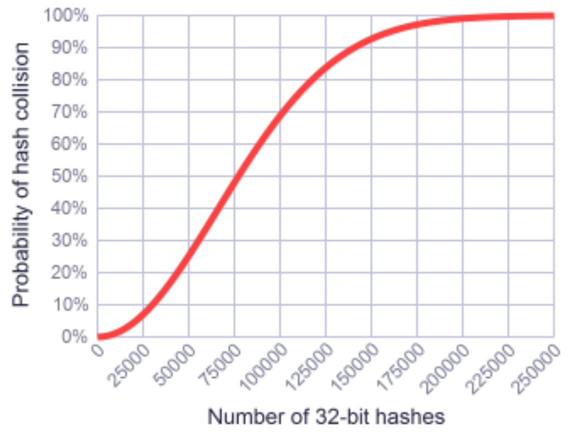
- Why?
  - Only verifies and does not parse



Second Preimage Resistance or Weak Collision Resistance

- A collision occurs when two different inputs hash to the same output
  - Two different messages must not hash to the same value
- $For\ x, y\ \in Input\ and\ H(x), H(y) \in Output \Rightarrow x\ != y\ \&\ H(x) = H(y)$
- If a Hash function is not injective (one-to-one), collisions are inevitable
- There is always a probability that collision happens in Hash function





Second Preimage Resistance or Weak Collision Resistance

- Susbstitution attack by Man-In-The-Middle
- i.e. given two different messages $x_1 \neq x_2$
- It should be computationally infeasible to create
  - $z_1 = h(x_1) = h(x_2) = z_2$

Think about password hash. If an arbitrary value can hash to the same password hash, then it is possible for an attacker to be authenticated

Strong Collision Resistance and Birthday Paradox

- It should be computationally infeasible to find two different messages $x_1 \neq x_2$ with $h(x_1) = h(x_2)$
- If an attacker can alter x_1 and x_2 in n locations to achieve $h(x_1) = h(x_2)$
  - 2^n different hash values
  - If n = 80bits, an attacker really needs 2^{40} computations due to the birthday paradox
- Birthday paradox
  - How many people are needed at a party such that there is a reasonable chance that at least two people have the same birthday?

$$P(no\ collision\ among\ k\ people) = (1 - \frac{1}{365}) \cdot (1 - \frac{2}{365}) \cdots (1 - \frac{k-1}{365})$$

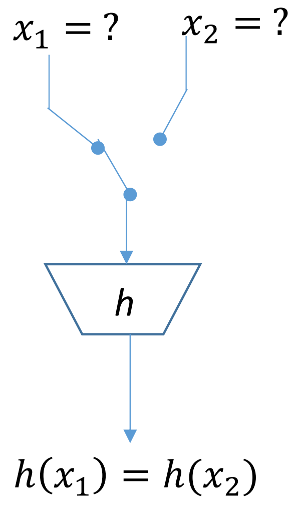
$$P(at\ least\ one\ collision) = 1 - P(no\ collision)$$

$$k = 23\ people$$

$$P(at\ least\ one\ collision) = 1 - \left(1 - \frac{1}{365}\right) \cdots \left(1 - \frac{23-1}{365}\right) = 0.507\ \approx 50\%$$

$$k = 40\ people$$

We get a probability of about 90%



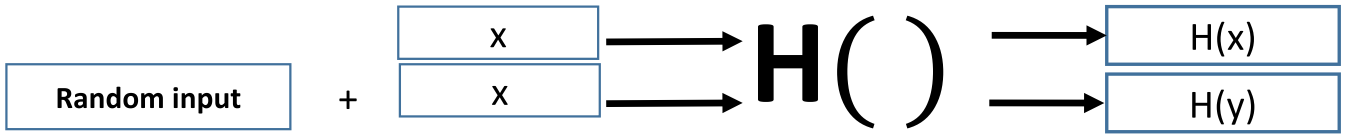
Hash values needed for collision in a birthday attack

- To thwart collision attacks based on birthday paradox
  - Output length of a hash function must be about twice as long as an output length that can protect against second preimage attack (brute-force attack)

	Hash output length				
λ	128 bit	160 bit	256 bit	384 bit	512 bit
0.5	2^{65}	2^{81}	2^{129}	2^{193}	2^{257}
0.9	2^{67}	2^{82}	2^{130}	2^{194}	2^{258}

Defense by salting Hash

- Since there is no way to produce a message from hash
- Brute-force attacks are mostly used
- e.g. attacker can precompute many hash outputs
  - Easy to use as lookup tables
- Solution is to add random input to the message before hashing
- Prevents brute-force and rainbow attacks



Common Hash Functions

- MD4 Family (Message Digest)
  - MD4 (128 bits)
  - MD5 (128 bits)
  - MD6 (up to 512 bits)
- SHA Family (Secure Hashing)
  - SHA-1 (160 bits)
  - SHA-2 (variants: SHA-256 and SHA-512 b)
  - SHA-3
- RIPEMD (RACE Integrity Primitives Evaluation Message Digest)
  - RIPEMD-160 (160 bits)

- Crypt, bcrypt, scrypt, PBKDF2

Collisions

Algorithm		Output [bit]	Input [bit]	No.of rounds	Collisions found
MD5		128	512	64	yes
SHA-1		160	512	80	Not yet
SHA-2	SHA-224	224	512	64	no
	SHA-256	256	512	64	no
	SHA-384	384	1024	80	no
	SHA-512	512	1024	80	no

- 2004 Wang et. al delivered an algorithm that could produce collisions in a few hours on an IBM p690 cluster
- Algorithm was improved by Lenstra et. al in 2005 to a few hours on a single laptop

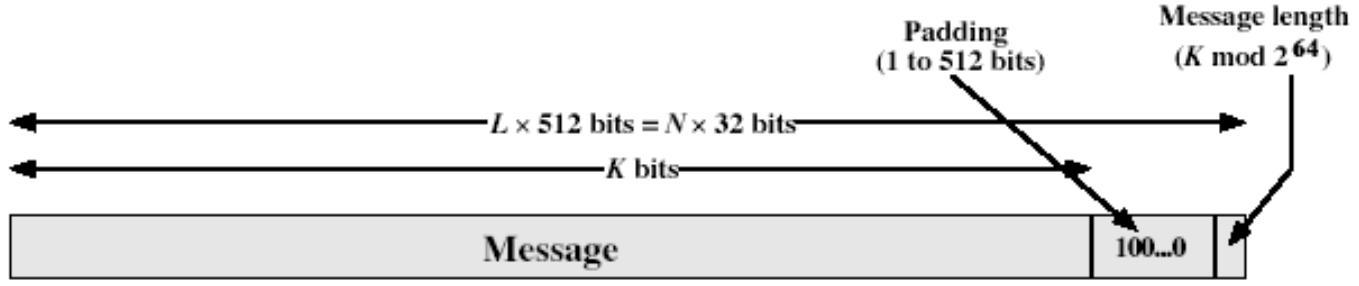
MD5 Algorithm

- designed by *Ronald Rivest* (the “*R*” in RSA)
- latest in a series of MD2, MD4
- produces a 128-bit hash value
- Used to be the most widely used hash algorithm
- specified as Internet standard RFC1321

Preprocessing – Message Padding

- Message
  - Must be padded to fit a multiple of 512 bit
- Append "1" followed by ***l*** zero bits and the binary 64-bit representation of ***k***

$l \equiv 448 - (k+1), where\ l = number\ of\ zeroes\ and\ k = length\ of\ message$



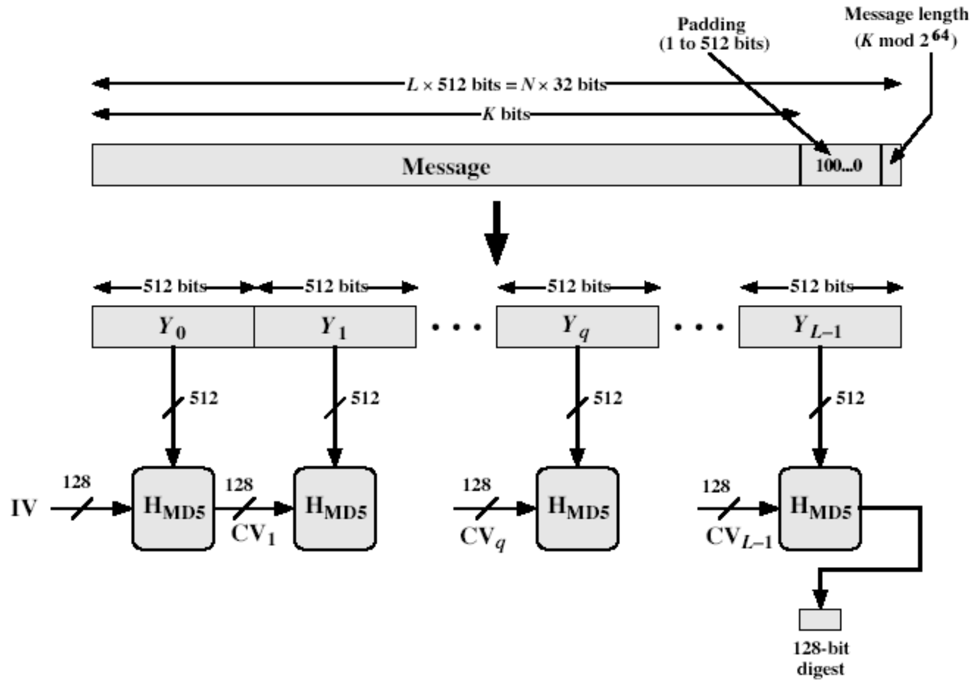
Message Padding - Example

- M = "abc"
- M = $01100001\ 01100010\ 01100011_2$
- len(M)= k = 8 * 3 = 24 bits
- Append a "1" followed by l = 423 zero bits ($l = 448 - (24 + 1) = 423\ mod\ 512$)
- Append 64-bit representation of k in binary
  - i.e. k = 24_{10} =110002 (we make this 64 bit by appending 0s to the left)

$$\underbrace{01100001}_{\text{a}} \quad \underbrace{01100010}_{\text{b}} \quad \underbrace{01100011}_{\text{c}} \quad 1 \quad \underbrace{00\ldots0}_{\text{423 zeros}} \quad \underbrace{00..011000}_{\text{64-bit for k=24}}$$

MD5 - Overview

1. pad message so its length is 448 mod 512
2. append a 64-bit length value to message
3. initialise 4-word (128-bit) MD register/buffer (A,B,C,D)
4. process message in 16-word (512-bit) blocks:
   - using 4 rounds of 16 bit operations on message block & buffer
   - add output to buffer input to form new buffer value
5. output hash value is the final buffer value



SHA

- SHA was designed by NIST & NSA in 1993, revised 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - ***note:*** the algorithm is SHA, the standard is SHS
- produces 160-bit hash values
- now the generally preferred hash algorithm
- based on design of MD4 with key differences

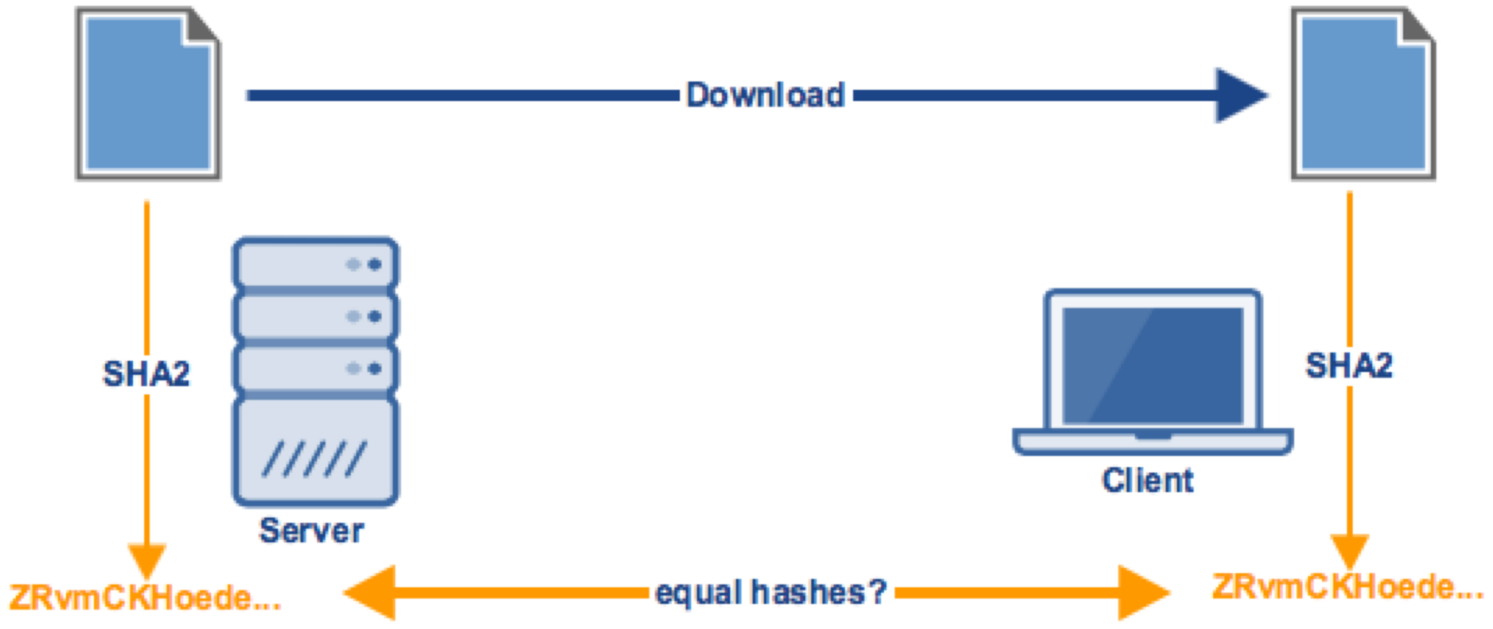
SHA Overview

1. pad message so its length is 448 mod 512
2. append a 64-bit length value to message
3. initialise 5-word (160-bit) buffer (A,B,C,D,E) to
   (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
4. process message in 16-word (512-bit) chunks:
   - expand 16 words into 80 words by mixing & shifting
   - use 4 rounds of 20 bit operations on message block & buffer
   - add output to input to form new buffer value
5. output hash value is the final buffer value

SHA-1 vs MD5

- brute force attack is harder (160 vs 128 bits for MD5)
- not vulnerable to any known attacks (compared to MD4/5)
- a little slower than MD5 (80 vs 64 steps)

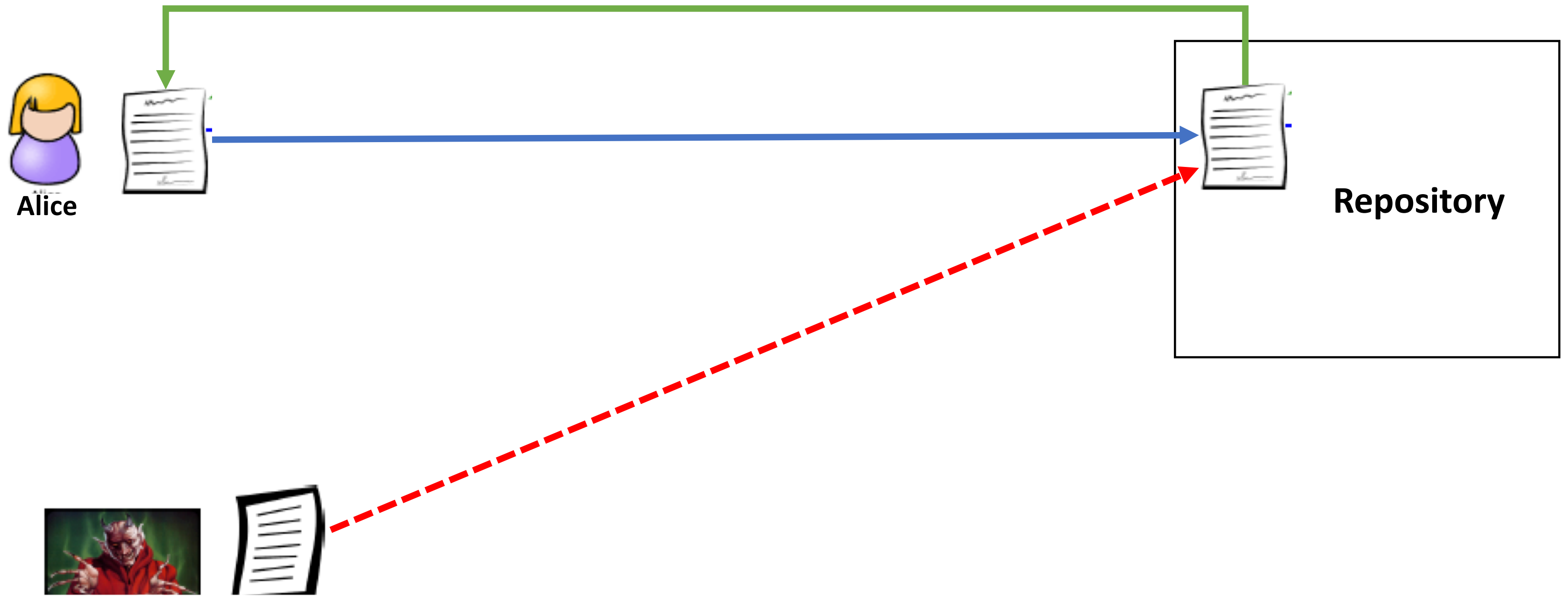
Applications



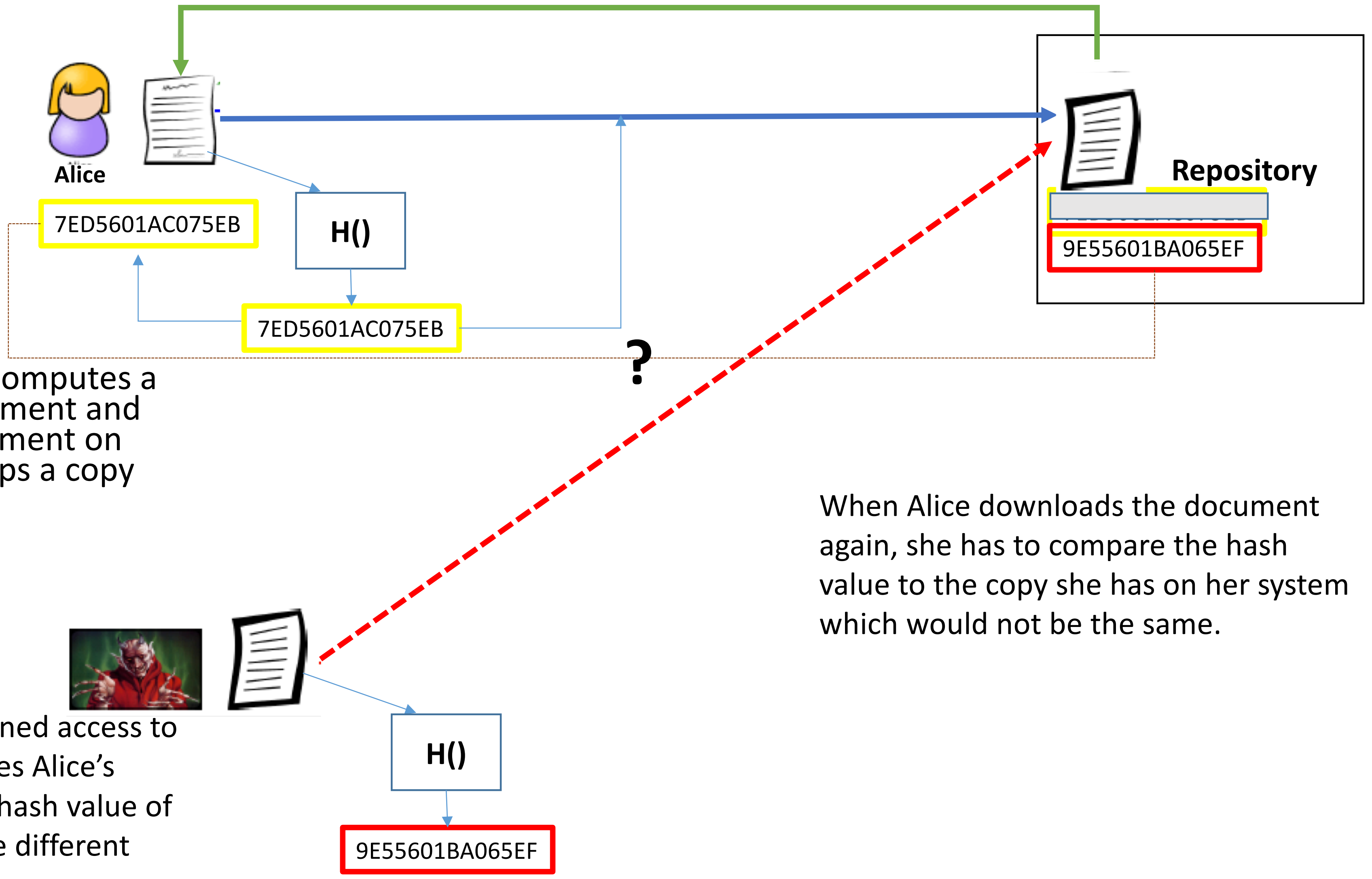
An example

Applications

- Alice wants to update a very large document in Dropbox repository.
- She wants to be sure that when she downloads the document it is exactly the same document.
- An adversary wants to substitute Alice's document for a forged one.



Applications: Hash

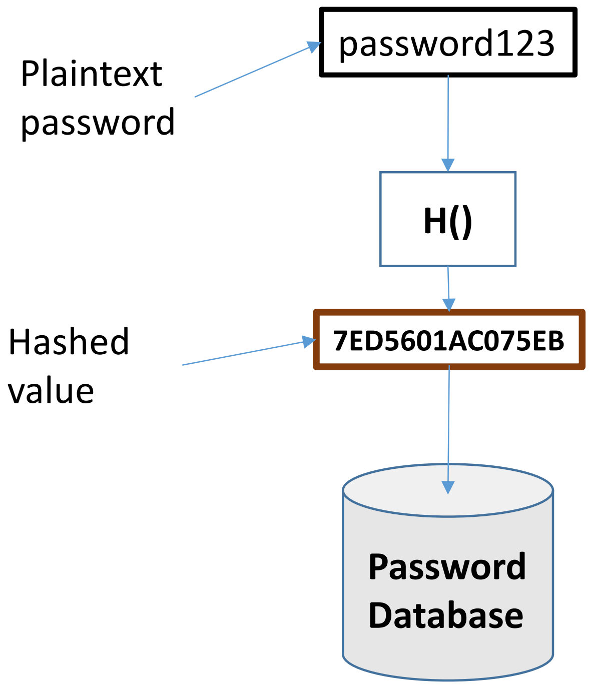


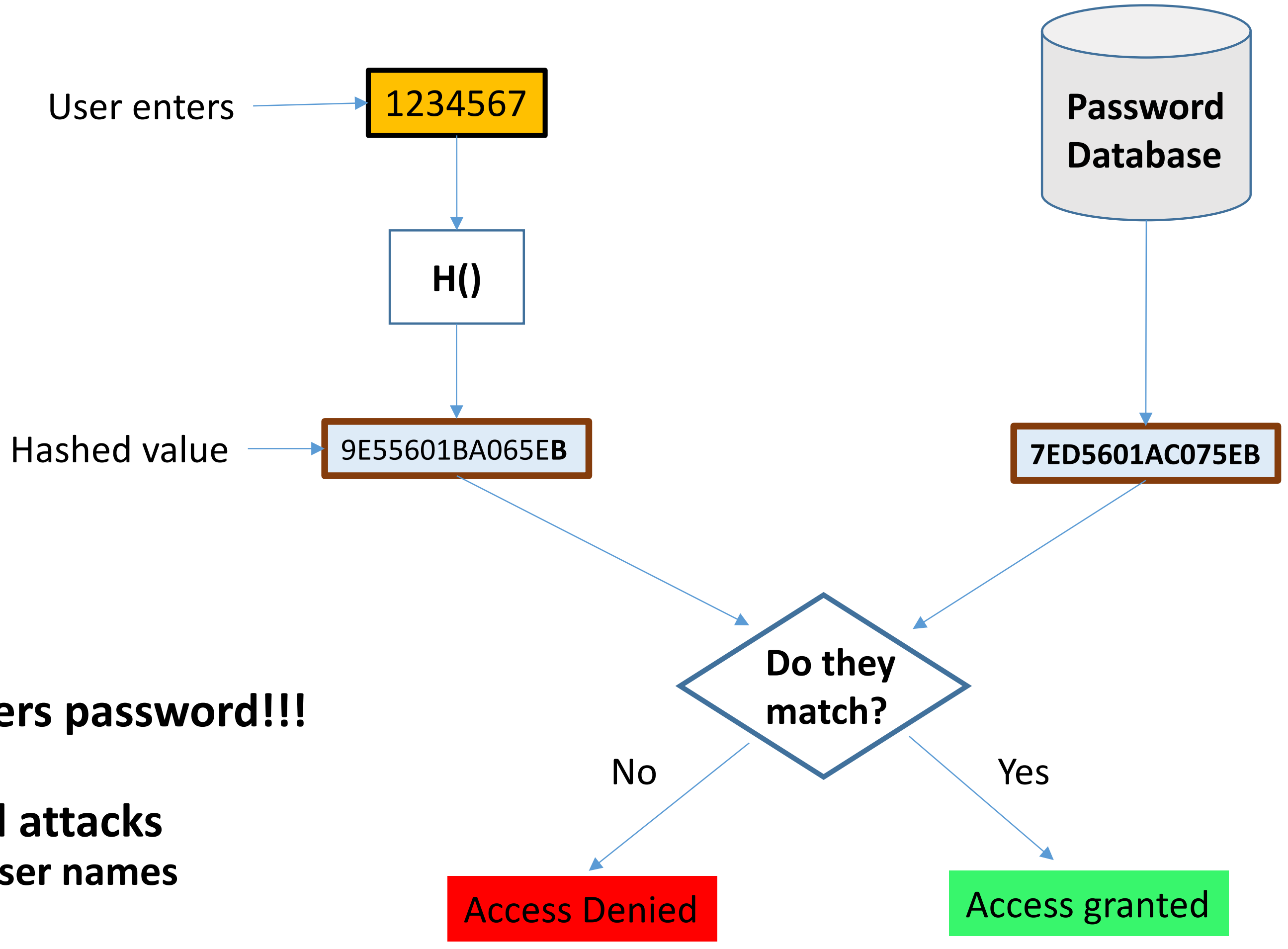
As a protection, Alice computes a hash value of the document and appends it to the document on the repository and keeps a copy on her local system

When Alice downloads the document again, she has to compare the hash value to the copy she has on her system which would not be the same.

Assume the adversary gained access to the repository and replaces Alice's document. However, the hash value of the new document will be different

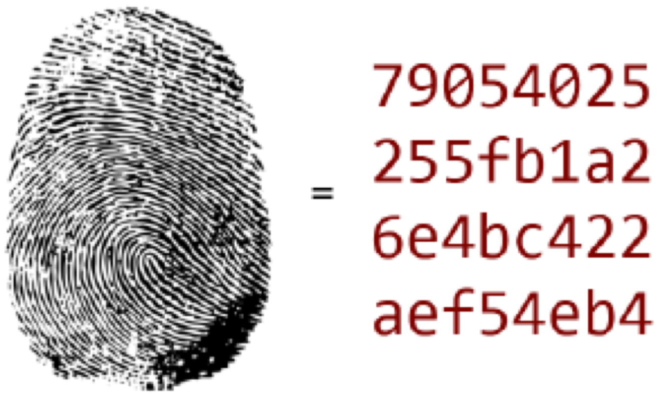
Applications: Password storage and verification

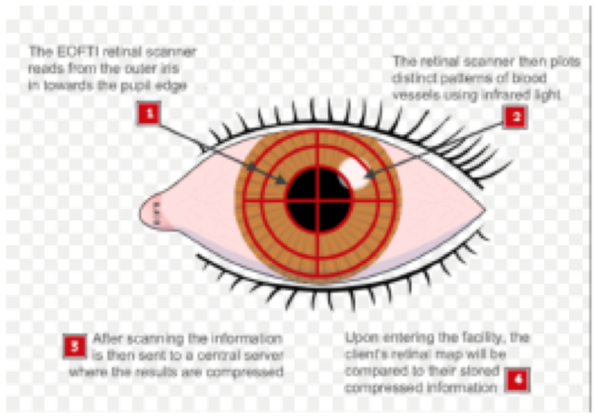




- **Even your org must not know the users password!!!**
  - **Internal and external threats**
- **Reduce security breach for password attacks**
  - **By storing the hash digests with the user names**

Applications: Biometrics and Fingerprints




The EOFTI retinal scanner reads from the outer iris in towards the pupil edge
1
The retinal scanner then plots distinct patterns of blood vessels using infrared light
2
3
After scanning the information is then sent to a central server where the results are compressed
Upon entering the facility, the client's retinal map will be compared to their stored compressed information
4

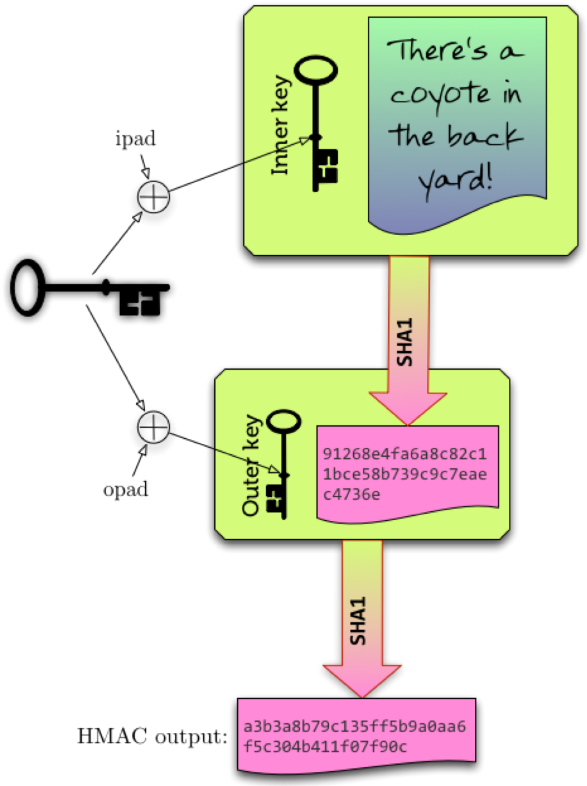
Cryptography Hash functions to compute cryptographic values from personal biometrics data (e.g. fingerprint, retinal-scan, etc.)

Message Authentication Code (MAC)

- Cryptographic checksum or keyed hash function
- Widely used in practice
- Shares similar properties with digital signatures
    - Authentication and Integrity
    - i.e. Who is the sender? and Has the message been tampered with?
- Uses symmetric keys unlike digital signatures
- Do not provide non-repudiation

Message Authentication Code (MAC)

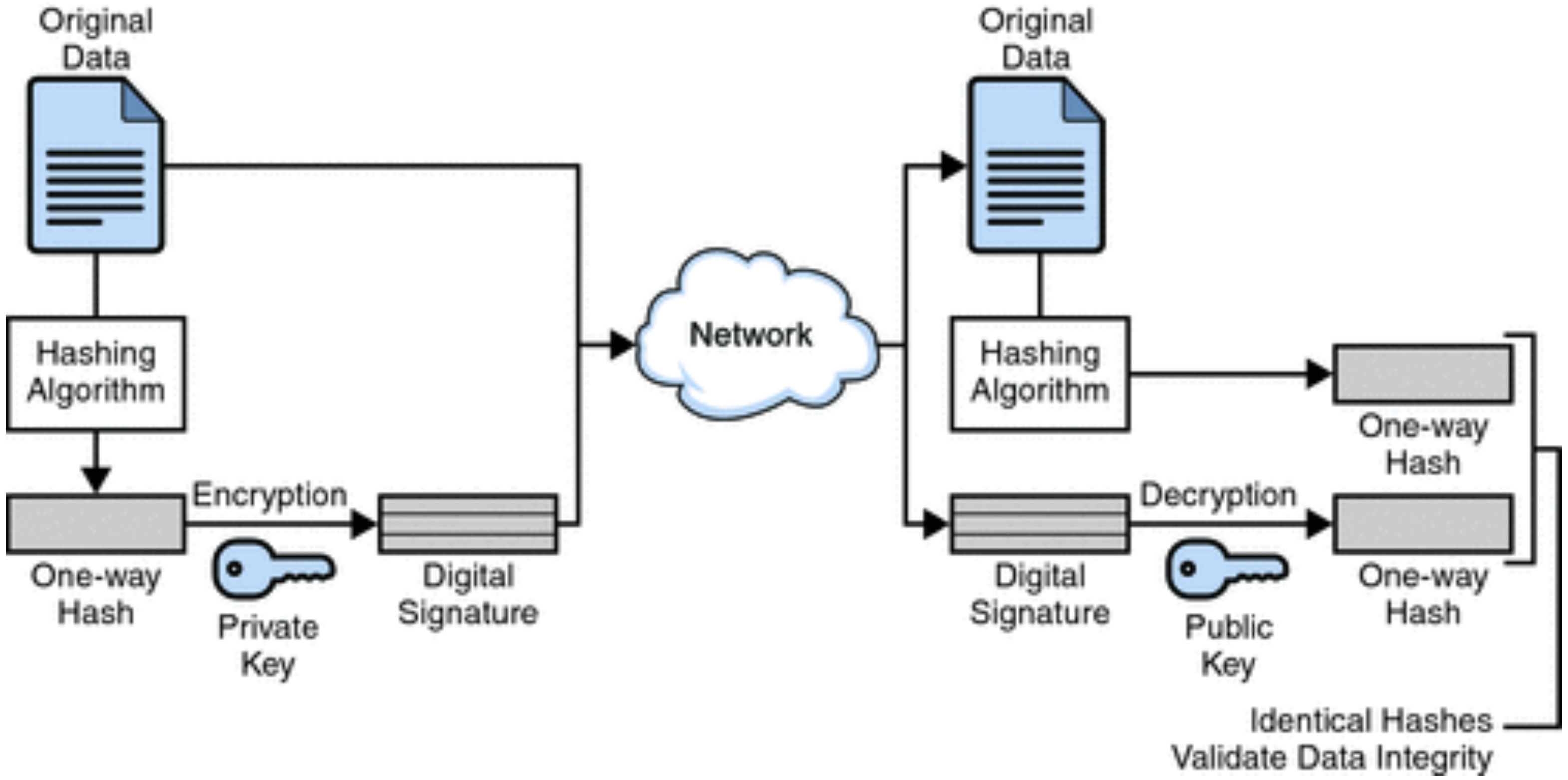
- A MAC authenticates a message
  - A signature based on a secret key
- Hash algorithm + key to make hash value dependent on the key
- Most common form is HMAC (hash MAC)
  - hash(key, hash(key, data))
- Naming: hash + key = HMAC-hash
  - MD5 1 HMAC-MD5
  - SHA-1 1 HMAC-SHA (recommended)


ipad
Inner key
There's a coyote in the back yard!
SHA1
opad
Outer key
91268e4fa6a8c82c1 1bce58b739c9c7eae c4736e
SHA1
HMAC output:
a3b3a8b79c135ff5b9a0aa6 f5c304b411f07f90c

Hashing and Digital Signature

- Combines a hash with a digital signature algorithm
- To sign
  - hash the data
  - encrypt the hash with the sender's private key
  - send data, signer’s name and signature.
- To verify
  - hash the data
  - find the sender’s public key
  - decrypt the signature with the sender's public key
  - the result of which should match the hash

Hashing + Digital Signature



Quiz – 5mins

1. In digital signature:
   a) The public key is used to sign the message and the private key used to verify the signature
   b) The public key is used to encrypt the message and the private key is used to decrypt it
   c) The private key is used to sign the message and the public key to verify
2. Hash functions should be collision resistant because:
   a) Authentication can only be achieved if there is no collision
   b) Integrity can only be preserved if there is no collision
   c) Confidentiality can only be achieved if there is no collision
3. MAC
   a) Uses asymmetric keys and provide authentication and integrity
   b) Uses symmetric key and provide authentication and integrity
   c) Uses symmetric key and provide authentication, integrity and non-repudiation
4. One-wayness of a hash function means
   a) Two different messages cannot hash to the same hash values
   b) One message cannot produce two different hash values
   c) It is not possible to derive the input from the hash output
5. Which hash function has collision been found?
   a) SHA-256
   b) MD5
   c) RIPEMD-160
6. What is the major difference between MAC and Hashing
   a) MAC provides integrity
   b) MAC provides authentication
   c) MAC provides confidentiality
7. What is the major difference between MAC and digital signature
   a) MAC hashes the message and encrypt with the private key
   b) MAC hashes the message and the symmetric key
   c) MAC hashes the message
8. Why do we need to hash in digital signature?
   a) Because we need irreversible function
   b) Because of speed and size
   c) To avoid message collisions

Resources

- https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/what-is-hashing
- https://preshing.com/20110504/hash-collision-probabilities/
- Cryptography and Network Security (Various Hash Algorithms) – Lecture slides by Laurie Brown
- https://blog.agilebits.com/2013/01/18/authenticated-encryption-and-how-not-to-get-caught-chasing-a-coyote/
- https://www.jscape.com/blog/what-is-hmac-and-how-does-it-secure-file-transfers
- Crypto-Hashing and applications: Curs 2017
- Paar & Pelzi. Understanding Cryptography: A Textbook for Students and Practitioners