An Introduction to Hash Functions

Christophe Clavier - Florent Bruguier

University of Limoges - University of Montpellier

Definition and properties

$$
\begin{array}{rcl}
\mathcal{H} : \{0,1\}^* & \longrightarrow & \{0,1\}^n \\
m & \longmapsto & \mathcal{H}(m)
\end{array}
$$

Definition and properties

$$\begin{array}{rcl}\mathcal{H}: \{0,1\}^* & \longrightarrow & \{0,1\}^n \\ m & \longmapsto & \mathcal{H}(m)\end{array}$$

- takes a message m of arbitrary length as an input

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Definition and properties

Definition and properties

$$\begin{array}{rcl}\mathcal{H}: \{0,1\}^* & \longrightarrow & \{0,1\}^n \\ m & \longmapsto & \mathcal{H}(m)\end{array}$$

- takes a message m of arbitrary length as an input
- output a fixed length *message digest* or *hash value* of 128 to 512 bits

C. Clavier - F. Bruguier (Unilim - UM) An Introduction to Hash Functions Hanoi, April 2020 4 / 34

Definition and properties

$$\begin{aligned} \mathcal{H} : \{0,1\}^* &\longrightarrow \{0,1\}^n \\ m &\longmapsto \mathcal{H}(m) \end{aligned}$$

- takes a message m of arbitrary length as an input
- output a fixed length *message digest* or *hash value* of 128 to 512 bits

A hash function is ...

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Definition and properties

Definition and properties

$$\begin{aligned} \mathcal{H} : \{0,1\}^* &\longrightarrow \{0,1\}^n \\ m &\longmapsto \mathcal{H}(m) \end{aligned}$$

- takes a message m of arbitrary length as an input
- output a fixed length *message digest* or *hash value* of 128 to 512 bits

A hash function is ...

- public: this is not a secret

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 4 / 34

Definition and properties

$$\begin{aligned} \mathcal{H} : \{0,1\}^* &\longrightarrow \{0,1\}^n \\ m &\longmapsto \mathcal{H}(m) \end{aligned}$$

- takes a message m of arbitrary length as an input
- output a fixed length *message digest* or *hash value* of 128 to 512 bits

A hash function is ...

- public: this is not a secret
- deterministic: anybody can compute $\mathcal{H}(m)$ unambiguously

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Definition and properties

Definition and properties

$$\begin{aligned} \mathcal{H} : \{0,1\}^* &\longrightarrow \{0,1\}^n \\ m &\longmapsto \mathcal{H}(m) \end{aligned}$$

- takes a message m of arbitrary length as an input
- output a fixed length *message digest* or *hash value* of 128 to 512 bits

A hash function is ...

- public: this is not a secret
- deterministic: anybody can compute $\mathcal{H}(m)$ unambiguously
- keyless: while used as an important cryptographic primitive

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 4 / 34

Definition and properties

- Change one bit in $m \Longrightarrow$ about half the bits change in $\mathcal{H}(m)$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Definition and properties

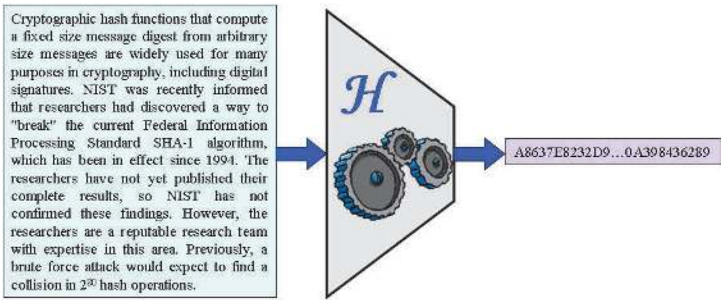
Definition and properties

- Change one bit in $m \Longrightarrow$ about half the bits change in $\mathcal{H}(m)$
  - Expected random behavior

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 5 / 34

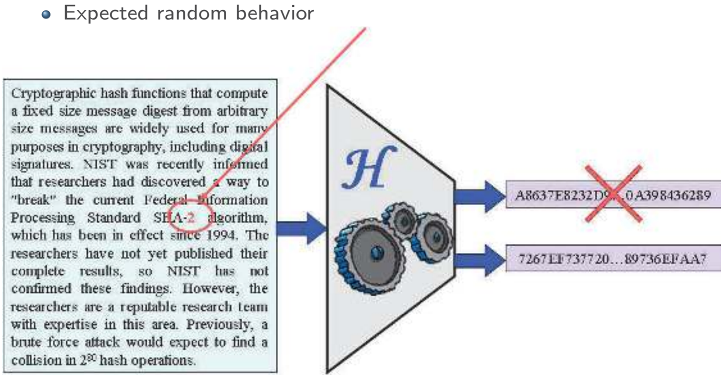
Definition and properties

- Change one bit in $m \Longrightarrow$ about half the bits change in $\mathcal{H}(m)$
  - Expected random behavior



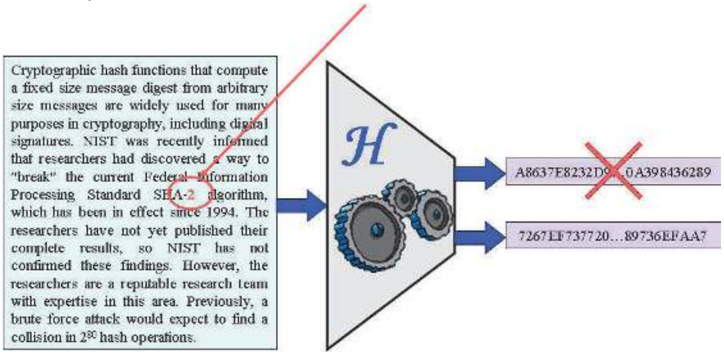
Definition and properties

- Change one bit in $m \Longrightarrow$ about half the bits change in $\mathcal{H}(m)$
  - Expected random behavior



Definition and properties

- Change one bit in $m \Longrightarrow$ about half the bits change in $\mathcal{H}(m)$
  - Expected random behavior



The hash value can be regarded as a fingerprint of the message

Examples of hash functions

Example (some hash functions)

Examples of hash functions

Example (some hash functions)

- MD4, MD5, RIPEMD have 128-bit hash values

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Examples of hash functions

Examples of hash functions

Example (some hash functions)

- MD4, MD5, RIPEMD have 128-bit hash values
- SHA-1 and RIPEMD-160 have 160-bit hash values

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 6 / 34

Examples of hash functions

Example (some hash functions)

- MD4, MD5, RIPEMD have 128-bit hash values
- SHA-1 and RIPEMD-160 have 160-bit hash values
- SHA-224 has a 224-bit hash value

Examples of hash functions

Example (some hash functions)

- MD4, MD5, RIPEMD have 128-bit hash values
- SHA-1 and RIPEMD-160 have 160-bit hash values
- SHA-224 has a 224-bit hash value
- SHA-256 has a 256-bit hash value

Examples of hash functions

Example (some hash functions)

- MD4, MD5, RIPEMD have 128-bit hash values
- SHA-1 and RIPEMD-160 have 160-bit hash values
- SHA-224 has a 224-bit hash value
- SHA-256 has a 256-bit hash value
- SHA-384 has a 384-bit hash value

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Examples of hash functions

Examples of hash functions

Example (some hash functions)

- MD4, MD5, RIPEMD have 128-bit hash values
- SHA-1 and RIPEMD-160 have 160-bit hash values
- SHA-224 has a 224-bit hash value
- SHA-256 has a 256-bit hash value
- SHA-384 has a 384-bit hash value
- SHA-512 has a 512-bit hash value

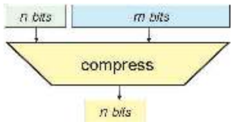
The Merkle-Damgård construction

How to tackle with arbitrarily long inputs ?

The Merkle-Damgård construction

How to tackle with arbitrarily long inputs ?

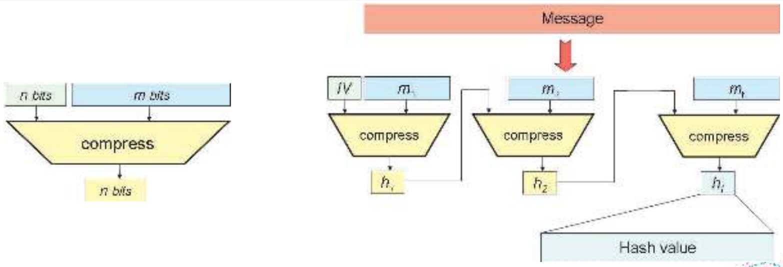
- A compression function maintains an n-bit internal state while processing m-bit message blocks



The Merkle-Damgård construction

How to tackle with arbitrarily long inputs ?

- A compression function maintains an n-bit internal state while processing m-bit message blocks
- A chaining construction builds the hash function upon the compression function



Security requirements

Preimage resistance (one-wayness)

Given $y \in \{0,1\}^n$, it should be impossible to find x s.t. $\mathcal{H}(x) = y$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Security requirements

Security requirements

Preimage resistance (one-wayness)

Given $y \in \{0,1\}^n$, it should be impossible to find x s.t. $\mathcal{H}(x) = y$

Idea: the function is not invertible

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 9 / 34

Security requirements

Preimage resistance (one-wayness)

Given $y \in \{0,1\}^n$, it should be impossible to find x s.t. $\mathcal{H}(x) = y$

Idea: the function is not invertible

Second preimage resistance

Given x, it should be impossible to find $x' \neq x$ s.t. $\mathcal{H}(x) = \mathcal{H}(x')$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Security requirements

Security requirements

Preimage resistance (one-wayness)

Given $y \in \{0,1\}^n$, it should be impossible to find x s.t. $\mathcal{H}(x) = y$

Idea: the function is not invertible

Second preimage resistance

Given x, it should be impossible to find $x' \neq x$ s.t. $\mathcal{H}(x) = \mathcal{H}(x')$

Idea: the knowledge of a preimage does not help to find a second one

Security requirements

Preimage resistance (one-wayness)

Given $y \in \{0,1\}^n$, it should be impossible to find x s.t. $\mathcal{H}(x) = y$

Idea: the function is not invertible

Second preimage resistance

Given x, it should be impossible to find $x' \neq x$ s.t. $\mathcal{H}(x) = \mathcal{H}(x')$

Idea: the knowledge of a preimage does not help to find a second one

Collision resistance

It should be impossible to find x and x' s.t. $\mathcal{H}(x) = \mathcal{H}(x')$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Security requirements

Security requirements

Preimage resistance (one-wayness)

Given $y \in \{0,1\}^n$, it should be impossible to find x s.t. $\mathcal{H}(x) = y$

Idea: the function is not invertible

Second preimage resistance

Given x, it should be impossible to find $x' \neq x$ s.t. $\mathcal{H}(x) = \mathcal{H}(x')$

Idea: the knowledge of a preimage does not help to find a second one

Collision resistance

It should be impossible to find x and x' s.t. $\mathcal{H}(x) = \mathcal{H}(x')$

Idea: if someone provides $y = \mathcal{H}(x)$, and next reveals x, he can not cheat

Security requirements

Preimage resistance (one-wayness)

Given $y \in \{0,1\}^n$, it should be impossible to find x s.t. $\mathcal{H}(x) = y$

Idea: the function is not invertible

Second preimage resistance

Given x, it should be impossible to find $x' \neq x$ s.t. $\mathcal{H}(x) = \mathcal{H}(x')$

Idea: the knowledge of a preimage does not help to find a second one

Collision resistance

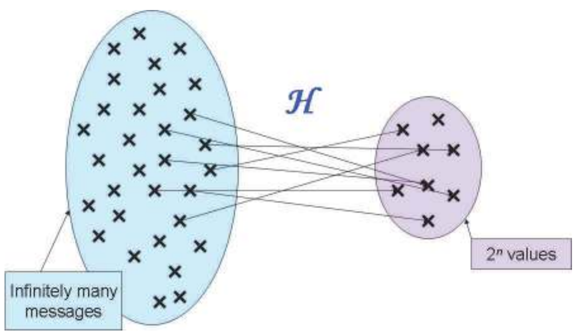
It should be impossible to find x and x' s.t. $\mathcal{H}(x) = \mathcal{H}(x')$

Idea: if someone provides $y = \mathcal{H}(x)$, and next reveals x, he can not cheat

Question

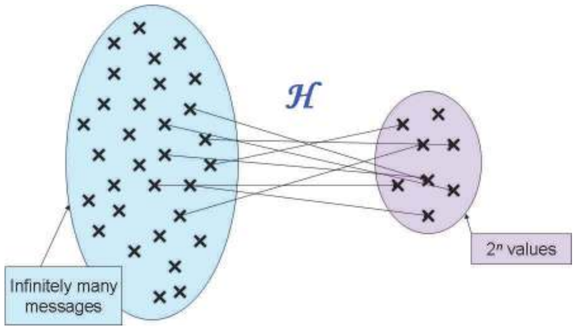
What does impossible means ?

Security considerations



- There are infinitely many messages

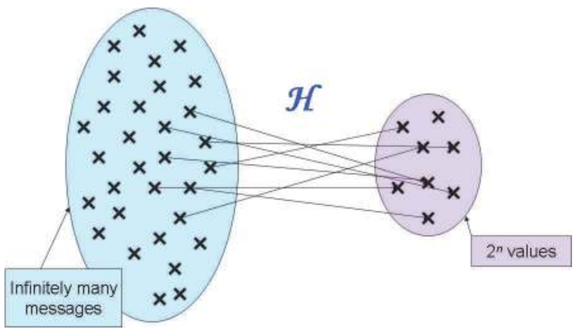
Security considerations



- There are infinitely many messages
  - Preimages and second preimages always exist

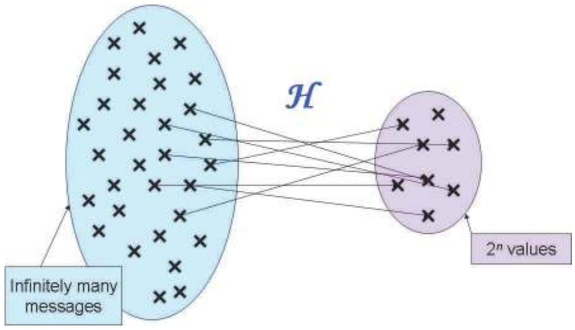
Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Security considerations

Security considerations



- There are infinitely many messages
  - Preimages and second preimages always exist
  - Collisions are unavoidable

Security considerations



- There are infinitely many messages
  - Preimages and second preimages always exist
  - Collisions are unavoidable
- Impossibility (absolute) so reduces to computational unfeasibility (relative)

Generic Attacks

- A generic attack is one whose complexity depends only on the size of the hash result, not on the details of the algorithm

Generic Attacks

- A generic attack is one whose complexity depends only on the size of the hash result, not on the details of the algorithm
- Some generic attacks apply to all hash functions

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Security considerations

Generic Attacks

- A generic attack is one whose complexity depends only on the size of the hash result, not on the details of the algorithm
- Some generic attacks apply to all hash functions
    - Finding preimages or second preimages is always possible by trivial exhaustive search within 2^n computations

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 11 / 34

Generic Attacks

- A generic attack is one whose complexity depends only on the size of the hash result, not on the details of the algorithm
- Some generic attacks apply to all hash functions
  - Finding preimages or second preimages is always possible by trivial exhaustive search within 2^n computations
  - Finding collisions is always possible within $2^{n/2}$ computations (birthday paradox)

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Security considerations

Generic Attacks

- A generic attack is one whose complexity depends only on the size of the hash result, not on the details of the algorithm
- Some generic attacks apply to all hash functions
  - Finding preimages or second preimages is always possible by trivial exhaustive search within 2^n computations
  - Finding collisions is always possible within $2^{n/2}$ computations (birthday paradox)

A secure hash function must not be vulnerable to better attacks

Complexity figures

- A machine able to perform 10^9 computations per second will perform:

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Complexity figures

Complexity figures

- A machine able to perform 10^9 computations per second will perform:
  - 2^{46} computations per day

Complexity figures

- A machine able to perform 10^9 computations per second will perform:
  - 2^{46} computations per day
  - 2^{55} computations per year

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
Complexity figures

Complexity figures

- A machine able to perform 10^9 computations per second will perform:
  - 2^{46} computations per day
  - 2^{55} computations per year
  - 2^{90} computations in 15 billions years

C. Clavier - F. Bruguier (Unilim - UM) An Introduction to Hash Functions Hanoi, April 2020 12 / 34

Complexity figures

- A machine able to perform 10^9 computations per second will perform:
  - 2^{46} computations per day
  - 2^{55} computations per year
  - 2^{90} computations in 15 billions years

One must choose the hash output size n so that an attacker can not reach $2^{n/2}$ computations

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Complexity figures

Complexity figures

- A machine able to perform 10^9 computations per second will perform:
  - 2^{46} computations per day
  - 2^{55} computations per year
  - 2^{90} computations in 15 billions years

One must choose the hash output size n so that an attacker can not reach $2^{n/2}$ computations

- $n = 80$ $\longrightarrow$ $2^{n/2} = 2^{40}$ (feasible)

Complexity figures

- A machine able to perform 10^9 computations per second will perform:
  - 2^{46} computations per day
  - 2^{55} computations per year
  - 2^{90} computations in 15 billions years

One must choose the hash output size n so that an attacker can not reach $2^{n/2}$ computations

- $n = 80$ $\longrightarrow$ $2^{n/2} = 2^{40}$ (feasible)
- $n = 128$ (MD5) $\longrightarrow$ $2^{n/2} = 2^{64}$ (becomes difficult)

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
Complexity figures

Complexity figures

- A machine able to perform 10^9 computations per second will perform:
  - 2^{46} computations per day
  - 2^{55} computations per year
  - 2^{90} computations in 15 billions years

One must choose the hash output size n so that an attacker can not reach $2^{n/2}$ computations

- $n = 80$ $\longrightarrow$ $2^{n/2} = 2^{40}$ (feasible)
- $n = 128$ (MD5) $\longrightarrow$ $2^{n/2} = 2^{64}$ (becomes difficult)
- $n = 160$ (SHA-1) $\longrightarrow$ $2^{n/2} = 2^{80}$ (believed secure for the moment)

C. Clavier - F. Bruguier (Unilim - UM) An Introduction to Hash Functions Hanoi, April 2020 12 / 34

Complexity figures

- A machine able to perform 10^9 computations per second will perform:
  - 2^{46} computations per day
  - 2^{55} computations per year
  - 2^{90} computations in 15 billions years

One must choose the hash output size n so that an attacker can not reach $2^{n/2}$ computations

- $n = 80$ $\longrightarrow$ $2^{n/2} = 2^{40}$ (feasible)
- $n = 128$ (MD5) $\longrightarrow$ $2^{n/2} = 2^{64}$ (becomes difficult)
- $n = 160$ (SHA-1) $\longrightarrow$ $2^{n/2} = 2^{80}$ (believed secure for the moment)
- $n = 256$ (SHA-256) $\longrightarrow$ $2^{n/2} = 2^{128}$ (highly secure)

Secured password storage

- To be granted access to her account, Alice must present a password which is to be compared with a previously stored value

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Secured password storage

Secured password storage

- To be granted access to her account, Alice must present a password which is to be compared with a previously stored value
- Clear text password storage may be jeopardized (reading, modification) by unauthorized file access

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 14 / 34

Secured password storage

- To be granted access to her account, Alice must present a password which is to be compared with a previously stored value
- Clear text password storage may be jeopardized (reading, modification) by unauthorized file access

Solution

Store a list of $\{user, \mathcal{H}(user\text{'s password})\}$
When Alice identifies herself by presenting password p, check that $\mathcal{H}(p) = \mathcal{H}(\text{Alice's password})$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Secured password storage

Secured password storage

- To be granted access to her account, Alice must present a password which is to be compared with a previously stored value
- Clear text password storage may be jeopardized (reading, modification) by unauthorized file access

Solution

Store a list of $\{user, \mathcal{H}(user\text{'s password})\}$
When Alice identifies herself by presenting password p, check that $\mathcal{H}(p) = \mathcal{H}(\text{Alice's password})$

- This solution does not prevent from dictionary attacks

Secured password storage

- To be granted access to her account, Alice must present a password which is to be compared with a previously stored value
- Clear text password storage may be jeopardized (reading, modification) by unauthorized file access

Solution

Store a list of $\{user, \mathcal{H}(user\text{'s password})\}$
When Alice identifies herself by presenting password p, check that $\mathcal{H}(p) = \mathcal{H}(\text{Alice's password})$

- This solution does not prevent from dictionary attacks
  - But usage of *salt* technique may circumvent the problem

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
Secured password storage

Secured password storage

- To be granted access to her account, Alice must present a password which is to be compared with a previously stored value
- Clear text password storage may be jeopardized (reading, modification) by unauthorized file access

Solution

Store a list of $\{user, \mathcal{H}(user\text{'s password})\}$
When Alice identifies herself by presenting password p, check that $\mathcal{H}(p) = \mathcal{H}(\text{Alice's password})$

- This solution does not prevent from dictionary attacks
  - But usage of *salt* technique may circumvent the problem
- The hash function needs to be preimage resistant

Data integrity

- Alice downloads a text m from an internet server

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
Data integrity

Data integrity

- Alice downloads a text m from an internet server
- She wants to make sure the text hasn't been changed since it has been sent by the server

C. Clavier - F. Bruguier (Unilim - UM) An Introduction to Hash Functions Hanoi, April 2020 15 / 34

Data integrity

- Alice downloads a text m from an internet server
- She wants to make sure the text hasn't been changed since it has been sent by the server

Solution

Add a $\mathcal{H}(m)$ of the text m next to it, so that anybody can check whether the hash value matches that of the downloaded text

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Data integrity

Data integrity

- Alice downloads a text m from an internet server
- She wants to make sure the text hasn't been changed since it has been sent by the server

Solution

Add a $\mathcal{H}(m)$ of the text m next to it, so that anybody can check whether the hash value matches that of the downloaded text

- This solution does not prevent from an attacker who controls the web site

Data integrity

- Alice downloads a text m from an internet server
- She wants to make sure the text hasn't been changed since it has been sent by the server

Solution

Add a $\mathcal{H}(m)$ of the text m next to it, so that anybody can check whether the hash value matches that of the downloaded text

- This solution does not prevent from an attacker who controls the web site
- The hash function needs to be second preimage resistant

Entity authentication

- Juliet wants to identify Romeo on the phone

Entity authentication

- Juliet wants to identify Romeo on the phone
- Password based authentication is not appropriate
  (eavesdropper → one-time password!)

Entity authentication

- Juliet wants to identify Romeo on the phone
- Password based authentication is not appropriate
  (eavesdropper → one-time password!)

Solution

Challenge-response protocol + shared secret s

Entity authentication

- Juliet wants to identify Romeo on the phone
- Password based authentication is not appropriate
  (eavesdropper → one-time password!)

Solution

Challenge-response protocol + shared secret s

1. Juliet: sends a random challenge r

Entity authentication

- Juliet wants to identify Romeo on the phone
- Password based authentication is not appropriate
  (eavesdropper → one-time password!)

Solution

Challenge-response protocol + shared secret s

1. Juliet: sends a random challenge r
2. Romeo: sends challenge response as $C = \mathcal{H}(s||r)$

Entity authentication

- Juliet wants to identify Romeo on the phone
- Password based authentication is not appropriate
  (eavesdropper → one-time password!)

Solution

Challenge-response protocol + shared secret s

1. Juliet: sends a random challenge r
2. Romeo: sends challenge response as $C = \mathcal{H}(s||r)$
3. Juliet: computes $\mathcal{H}(s||r)$ and compares it with C

Message authentication

- John and Chris share a secret key K

Message authentication

- John and Chris share a secret key K
- John wants to send a message m to Chris without anybody being able to modify it

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Message authentication

Message authentication

- John and Chris share a secret key K
- John wants to send a message m to Chris without anybody being able to modify it
- Chris wants to make sure the sender is John and the message wasn't tampered with

C. Clavier - F. Bruguier (Unilim - UM) An Introduction to Hash Functions Hanoi, April 2020 17 / 34

Message authentication

- John and Chris share a secret key K
- John wants to send a message m to Chris without anybody being able to modify it
- Chris wants to make sure the sender is John and the message wasn't tampered with

Solution

Message authentication

- John and Chris share a secret key K
- John wants to send a message m to Chris without anybody being able to modify it
- Chris wants to make sure the sender is John and the message wasn't tampered with

Solution

- HMAC(m,K) computes a Message Authentication Code (MAC) on m using a keyed hash function

Message authentication

- John and Chris share a secret key K
- John wants to send a message m to Chris without anybody being able to modify it
- Chris wants to make sure the sender is John and the message wasn't tampered with

Solution

- HMAC(m,K) computes a Message Authentication Code (MAC) on m using a keyed hash function
- $\text{HMAC}(m, K) = \mathcal{H}(K \oplus \text{opad}||\mathcal{H}(K \oplus \text{ipad}||m))$

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)
- To ensure non-repudiation, use digital signatures

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Digital signature

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)
- To ensure non-repudiation, use digital signatures
    - Only John can sign the message using his private key

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 18 / 34

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)
- To ensure non-repudiation, use digital signatures
  - Only John can sign the message using his private key
  - Anybody can verify the signature using John's public key

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)
- To ensure non-repudiation, use digital signatures
  - Only John can sign the message using his private key
  - Anybody can verify the signature using John's public key
- But . . .

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)
- To ensure non-repudiation, use digital signatures
  - Only John can sign the message using his private key
  - Anybody can verify the signature using John's public key
- But ...
  - Signing several megabytes is too slow

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)
- To ensure non-repudiation, use digital signatures
  - Only John can sign the message using his private key
  - Anybody can verify the signature using John's public key
- But ...
  - Signing several megabytes is too slow
  - Signature algorithms may be malleable

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)
- To ensure non-repudiation, use digital signatures
  - Only John can sign the message using his private key
  - Anybody can verify the signature using John's public key
- But ...
  - Signing several megabytes is too slow
  - Signature algorithms may be malleable

Solution

Only sign the hash of the message: $\mathsf{Sign}(\mathcal{H}(m))$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Digital signature

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)
- To ensure non-repudiation, use digital signatures
  - Only John can sign the message using his private key
  - Anybody can verify the signature using John's public key
- But ...
  - Signing several megabytes is too slow
  - Signature algorithms may be malleable

Solution

Only sign the hash of the message: $\mathsf{Sign}(\mathcal{H}(m))$

- If the message is modified, the signature is not valid anymore ...

Digital signature

- The previous technique does not provide non-repudiation (John could claim Chris produced the MAC himself, using shared key K)
- To ensure non-repudiation, use digital signatures
  - Only John can sign the message using his private key
  - Anybody can verify the signature using John's public key
- But ...
  - Signing several megabytes is too slow
  - Signature algorithms may be malleable

Solution

Only sign the hash of the message: $\mathrm{Sign}(\mathcal{H}(m))$

- If the message is modified, the signature is not valid anymore ...
- ... provided the hash function is collision resistant

Birthday paradox

The classical problem

Question

How many persons are needed for having more than 50% chance that two of them share the same birthday ?

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
Birthday paradox

Birthday paradox

The classical problem

Question

How many persons are needed for having more than 50% chance that two of them share the same birthday ?

Let $p(365, m)$ the probability that, given m persons, at least two share the same birthday. Then ...

Birthday paradox
The classical problem

Question
How many persons are needed for having more than 50% chance that two of them share the same birthday ?

Let $p(365, m)$ the probability that, given m persons, at least two share the same birthday. Then ...

- Intuitively, $p(365, m)$ increases with m

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
Birthday paradox

Birthday paradox
The classical problem

Question
How many persons are needed for having more than 50% chance that two of them share the same birthday ?

Let $p(365, m)$ the probability that, given m persons, at least two share the same birthday. Then ...

- Intuitively, $p(365, m)$ increases with m
- Obviously, $p(365, 1) = 0$

Birthday paradox

The classical problem

Question

How many persons are needed for having more than 50% chance that two of them share the same birthday ?

Let $p(365, m)$ the probability that, given m persons, at least two share the same birthday. Then ...

- Intuitively, $p(365, m)$ increases with m
- Obviously, $p(365, 1) = 0$
- Also, $p(365, 366) = 1$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Birthday paradox

Birthday paradox

The classical problem

Question

How many persons are needed for having more than 50% chance that two of them share the same birthday ?

Let $p(365, m)$ the probability that, given m persons, at least two share the same birthday. Then ...

- Intuitively, $p(365, m)$ increases with m
- Obviously, $p(365, 1) = 0$
- Also, $p(365, 366) = 1$
- For some m^*, $p(365, m^* - 1) \leqslant \frac{1}{2}$ and $p(365, m^*) > \frac{1}{2}$

Birthday paradox

The classical problem

Question

How many persons are needed for having more than 50% chance that two of them share the same birthday ?

Let $p(365, m)$ the probability that, given m persons, at least two share the same birthday. Then ...

- Intuitively, $p(365, m)$ increases with m
- Obviously, $p(365, 1) = 0$
- Also, $p(365, 366) = 1$
- For some m^*, $p(365, m^* - 1) \leqslant \frac{1}{2}$ and $p(365, m^*) > \frac{1}{2}$

Answer

$$\begin{array}{l} p(365, 22) = 0,476 \\ p(365, 23) = 0,507 \end{array} \longrightarrow m^* = 23$$

Birthday paradox

Generalization

- An urn contains t balls numbered 1 to t

Birthday paradox
Generalization

- An urn contains t balls numbered 1 to t
- m balls are drawn at random from the urn

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
Birthday paradox

Birthday paradox
Generalization

- An urn contains t balls numbered 1 to t
- m balls are drawn at random from the urn

What is the probability $p(t, m)$ of at least one coincidence (a ball drawn at least twice) ?

Birthday paradox

Generalization

- An urn contains t balls numbered 1 to t
- m balls are drawn at random from the urn

What is the probability $p(t,m)$ of at least one coincidence (a ball drawn at least twice) ?

If $m = \mathcal{O}(\sqrt{t})$ and $t \to \infty$ then:

$$p(t,m) \longrightarrow 1 - exp\left(-\frac{m(m-1)}{2t} + \mathcal{O}\left(\frac{1}{\sqrt{t}}\right)\right) \approx 1 - exp\left(-\frac{m^2}{2t}\right)$$

Birthday paradox

Generalization

Proof

Let $q(t,m) = 1 - p(t,m)$ the probability that each m balls differ:

$$q(t,m) = \prod_{k=0}^{m-1}\left(1 - \frac{k}{t}\right)$$

$$ln(q(t,m)) = \sum_{k=0}^{m-1} ln\left(1 - \frac{k}{t}\right)$$

$$= \sum_{k=0}^{m-1}\left[-\frac{k}{t} + o\left(\frac{k}{t}\right)\right]$$

$$= -\frac{m(m-1)}{2t} + \mathcal{O}\left(\frac{m}{t}\right) \qquad \text{if } m = \mathcal{O}(\sqrt{t})$$

$$p(t,m) \overset{t\to\infty}{\longrightarrow} 1 - exp\left(-\frac{m(m-1)}{2t} + \mathcal{O}\left(\frac{1}{\sqrt{t}}\right)\right) \approx 1 - exp\left(-\frac{m^2}{2t}\right)$$

Birthday paradox

Applications

$$p(t,m) \approx 1 - exp\left(-\frac{m^2}{2t}\right)$$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Birthday paradox

Birthday paradox

Applications

$$p(t,m) \approx 1 - exp\left(-\frac{m^2}{2t}\right)$$

$$p(t,m) = \frac{1}{2} \iff m = \sqrt{2.ln(2).t} \simeq 1,18\sqrt{t}$$

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 23 / 34

Birthday paradox

Applications

$$p(t,m) \approx 1 - exp\left(-\frac{m^2}{2t}\right)$$

$$p(t,m) = \frac{1}{2} \iff m = \sqrt{2.ln(2).t} \simeq 1,18\sqrt{t}$$

Birthday problem

$t = 365 \Rightarrow m = 22,54$
Among only 23 persons, you'd better bet on a birthday coincidence

Birthday paradox

Applications

$$p(t,m) \approx 1 - exp\left(-\frac{m^2}{2t}\right)$$

$$p(t,m) = \frac{1}{2} \iff m = \sqrt{2.ln(2).t} \simeq 1,18\sqrt{t}$$

Birthday problem

$t = 365 \Rightarrow m = 22,54$
Among only 23 persons, you'd better bet on a birthday coincidence

Hash collisions

For an n-bit hash function ($t = 2^n$), a collision may be expected after having computed about $2^{n/2}$ hash values

Naive method

A first collision search algorithm:

1. Choose m_1 at random and store $(m_1, \mathcal{H}(m_1))$

Naive method

A first collision search algorithm:

1. Choose m_1 at random and store $(m_1, \mathcal{H}(m_1))$
2. Choose m_2 at random, check whether $\mathcal{H}(m_2)$ has ever been computed, else store $(m_2, \mathcal{H}(m_2))$

Naive method

A first collision search algorithm:

1. Choose m_1 at random and store $(m_1, \mathcal{H}(m_1))$
2. Choose m_2 at random, check whether $\mathcal{H}(m_2)$ has ever been computed, else store $(m_2, \mathcal{H}(m_2))$
3. Choose m_3 at random, check whether $\mathcal{H}(m_3)$ has ever been computed, else store $(m_3, \mathcal{H}(m_3))$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Collision search

Naive method

A first collision search algorithm:

1. Choose m_1 at random and store $(m_1, \mathcal{H}(m_1))$
2. Choose m_2 at random, check whether $\mathcal{H}(m_2)$ has ever been computed, else store $(m_2, \mathcal{H}(m_2))$
3. Choose m_3 at random, check whether $\mathcal{H}(m_3)$ has ever been computed, else store $(m_3, \mathcal{H}(m_3))$
4. … and so on until the current hash is already present in the hash array

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 24 / 34

Naive method

A first collision search algorithm:

1. Choose m_1 at random and store $(m_1, \mathcal{H}(m_1))$
2. Choose m_2 at random, check whether $\mathcal{H}(m_2)$ has ever been computed, else store $(m_2, \mathcal{H}(m_2))$
3. Choose m_3 at random, check whether $\mathcal{H}(m_3)$ has ever been computed, else store $(m_3, \mathcal{H}(m_3))$
4. ... and so on until the current hash is already present in the hash array

This algorithm requires $\mathcal{O}(\sqrt{2^n})$ time and $\mathcal{O}(\sqrt{2^n})$ memory

Floyd's collision finding algorithm

- Starting from an arbitrary x_0, consider the sequence of iterated hashes

$$x_{i+1} = \mathcal{H}(x_i)$$

Floyd's collision finding algorithm

- Starting from an arbitrary x_0, consider the sequence of iterated hashes

$$x_{i+1} = \mathcal{H}(x_i)$$

- After about $\sqrt{2^n}$ steps, two sequence elements x_α and x_β will be the same:

$$\begin{cases} x_\alpha = x_\beta \\ x_{\alpha-1} \neq x_{\beta-1} \end{cases} \tag{1}$$

Outline · What is a Hash Function ? · Security of Hash Functions · Applications · Generic Attacks · Some Dedicated Attacks
Collision search

Floyd's collision finding algorithm

- Starting from an arbitrary x_0, consider the sequence of iterated hashes

$$x_{i+1} = \mathcal{H}(x_i)$$

- After about $\sqrt{2^n}$ steps, two sequence elements x_α and x_β will be the same:

$$\begin{cases} x_\alpha = x_\beta \\ x_{\alpha-1} \neq x_{\beta-1} \end{cases} \tag{1}$$

- This gives a collision on the hash function since $\mathcal{H}(x_{\alpha-1}) = \mathcal{H}(x_{\beta-1})$

Floyd's collision finding algorithm

- Starting from an arbitrary x_0, consider the sequence of iterated hashes

$$x_{i+1} = \mathcal{H}(x_i)$$

- After about $\sqrt{2^n}$ steps, two sequence elements x_α and x_β will be the same:

$$\begin{cases} x_\alpha = x_\beta \\ x_{\alpha-1} \neq x_{\beta-1} \end{cases} \quad (1)$$

- This gives a collision on the hash function since $\mathcal{H}(x_{\alpha-1}) = \mathcal{H}(x_{\beta-1})$
- From then on, the sequence will endlessly cycle

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Collision search

Floyd's collision finding algorithm

- Starting from an arbitrary x_0, consider the sequence of iterated hashes

$$x_{i+1} = \mathcal{H}(x_i)$$

- After about $\sqrt{2^n}$ steps, two sequence elements x_α and x_β will be the same:

$$\begin{cases} x_\alpha = x_\beta \\ x_{\alpha-1} \neq x_{\beta-1} \end{cases} \quad (1)$$

- This gives a collision on the hash function since $\mathcal{H}(x_{\alpha-1}) = \mathcal{H}(x_{\beta-1})$
- From then on, the sequence will endlessly cycle
  - α is called the *tail length*

Floyd's collision finding algorithm

- Starting from an arbitrary x_0, consider the sequence of iterated hashes

$$x_{i+1} = \mathcal{H}(x_i)$$

- After about $\sqrt{2^n}$ steps, two sequence elements x_α and x_β will be the same:

$$\begin{cases} x_\alpha = x_\beta \\ x_{\alpha-1} \neq x_{\beta-1} \end{cases} \tag{1}$$

- This gives a collision on the hash function since $\mathcal{H}(x_{\alpha-1}) = \mathcal{H}(x_{\beta-1})$
- From then on, the sequence will endlessly cycle
  - α is called the *tail length*
  - $\delta = \beta - \alpha$ is called the *cycle length*

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Collision search

Floyd's collision finding algorithm

- Starting from an arbitrary x_0, consider the sequence of iterated hashes

$$x_{i+1} = \mathcal{H}(x_i)$$

- After about $\sqrt{2^n}$ steps, two sequence elements x_α and x_β will be the same:

$$\begin{cases} x_\alpha = x_\beta \\ x_{\alpha-1} \neq x_{\beta-1} \end{cases} \tag{1}$$

- This gives a collision on the hash function since $\mathcal{H}(x_{\alpha-1}) = \mathcal{H}(x_{\beta-1})$
- From then on, the sequence will endlessly cycle
  - α is called the *tail length*
  - $\delta = \beta - \alpha$ is called the *cycle length*

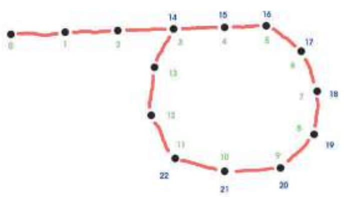
Problem ...

How to find two such equal sequence elements ?

Floyd's collision finding algorithm (example)

- A collision occurred at $\alpha = 3$ and $\beta = 14$

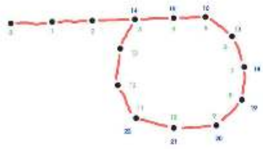
Floyd's collision finding algorithm (example)



- A collision occurred at $\alpha = 3$ and $\beta = 14$
- The cycle length is $\delta = 11$

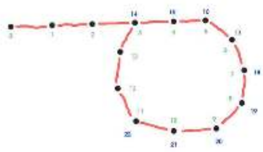
Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)



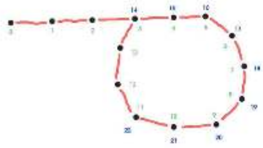
Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$



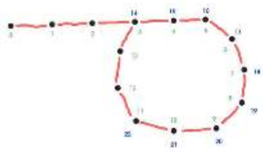
Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- $x_i = x_{2i}$ whenever $i \geqslant \alpha$ and $i = k.\delta$



Floyd's collision finding algorithm (description)

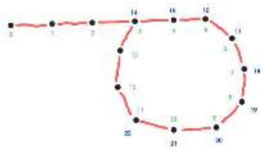
- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- $x_i = x_{2i}$ whenever $i \geqslant \alpha$ and $i = k.\delta$



Floyd's algorithm (step 1)

Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- $x_i = x_{2i}$ whenever $i \geqslant \alpha$ and $i = k.\delta$



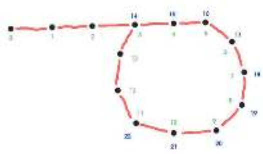
Floyd's algorithm (step 1)

1. Start with $(a_0, b_0) \leftarrow (x_0, x_0)$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Collision search

Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- $x_i = x_{2i}$ whenever $i \geqslant \alpha$ and $i = k.\delta$

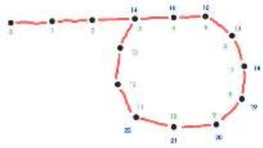


Floyd's algorithm (step 1)

1. Start with $(a_0, b_0) \leftarrow (x_0, x_0)$
2. Iteratively compute $(a_i, b_i) \leftarrow (\mathcal{H}(a_{i-1}), \mathcal{H}(\mathcal{H}(b_{i-1}))) = (x_i, x_{2i})$

Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- $x_i = x_{2i}$ whenever $i \geqslant \alpha$ and $i = k.\delta$



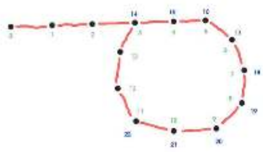
Floyd's algorithm (step 1)

1. Start with $(a_0, b_0) \leftarrow (x_0, x_0)$
2. Iteratively compute $(a_i, b_i) \leftarrow (\mathcal{H}(a_{i-1}), \mathcal{H}(\mathcal{H}(b_{i-1}))) = (x_i, x_{2i})$
3. Stop whenever $a_{i_0} = b_{i_0}$ (note that $i_0 = k.\delta = \left\lceil \frac{\alpha}{\delta} \right\rceil .\delta$)

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Collision search

Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- $x_i = x_{2i}$ whenever $i \geqslant \alpha$ and $i = k.\delta$



Floyd's algorithm (step 1)

1. Start with $(a_0, b_0) \leftarrow (x_0, x_0)$
2. Iteratively compute $(a_i, b_i) \leftarrow (\mathcal{H}(a_{i-1}), \mathcal{H}(\mathcal{H}(b_{i-1}))) = (x_i, x_{2i})$
3. Stop whenever $a_{i_0} = b_{i_0}$ (note that $i_0 = k.\delta = \left\lceil \frac{\alpha}{\delta} \right\rceil .\delta$)

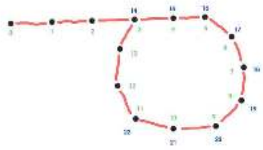
Cycle length found

A multiple $i_0 = k.\delta$ of the cycle length is obtained

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 27 / 34

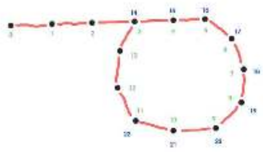
Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)



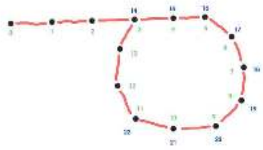
Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$



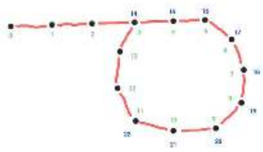
Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- x_{i_0} is known, where $i_0 = k.\delta$



Floyd's collision finding algorithm (description)

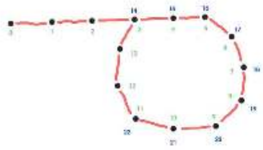
- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- x_{i_0} is known, where $i_0 = k.\delta$



Floyd's algorithm (step 2)

Floyd's collision finding algorithm (description)

- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- x_{i_0} is known, where $i_0 = k.\delta$



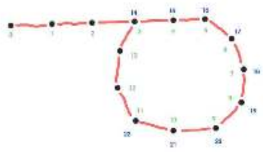
Floyd's algorithm (step 2)

4. Start with $(c_0, d_0) \leftarrow (x_0, x_{i_0}) = (x_0, x_{k.\delta})$

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Collision search

Floyd's collision finding algorithm (description)

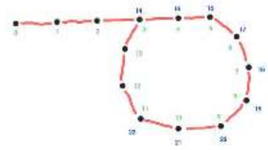
- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- x_{i_0} is known, where $i_0 = k.\delta$



Floyd's algorithm (step 2)

4. Start with $(c_0, d_0) \leftarrow (x_0, x_{i_0}) = (x_0, x_{k.\delta})$
5. Iteratively compute $(c_i, d_i) \leftarrow (\mathcal{H}(c_{i-1}), \mathcal{H}(d_{i-1})) = (x_i, x_{i+k.\delta})$

Floyd's collision finding algorithm (description)



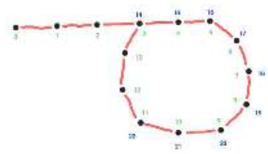
- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- x_{i_0} is known, where $i_0 = k.\delta$

Floyd's algorithm (step 2)

4. Start with $(c_0, d_0) \leftarrow (x_0, x_{i_0}) = (x_0, x_{k.\delta})$
5. Iteratively compute $(c_i, d_i) \leftarrow (\mathcal{H}(c_{i-1}), \mathcal{H}(d_{i-1})) = (x_i, x_{i+k.\delta})$
6. Stop whenever $c_{i_1} = d_{i_1}$ (note that $i_1 = \alpha$)

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Collision search

Floyd's collision finding algorithm (description)



- $\alpha = 3$ (tail) and $\delta = 11$ (cycle)
- $x_i = x_{i+k.\delta}$ for all $i \geqslant \alpha$ and $k \in \mathbb{N}$
- x_{i_0} is known, where $i_0 = k.\delta$

Floyd's algorithm (step 2)

4. Start with $(c_0, d_0) \leftarrow (x_0, x_{i_0}) = (x_0, x_{k.\delta})$
5. Iteratively compute $(c_i, d_i) \leftarrow (\mathcal{H}(c_{i-1}), \mathcal{H}(d_{i-1})) = (x_i, x_{i+k.\delta})$
6. Stop whenever $c_{i_1} = d_{i_1}$ (note that $i_1 = \alpha$)

Collision found !

The collision is given by $\mathcal{H}(x_{\alpha-1}) = \mathcal{H}(x_{\alpha-1+k\delta})$ with $x_{\alpha-1} \neq x_{\alpha-1+k\delta}$

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 28 / 34

Floyd's collision finding algorithm (complexity)

- This algorithm finds a collision in $3\left\lceil\frac{\alpha}{\delta}\right\rceil.\delta + 2\alpha$ hash evaluations

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
Collision search

Floyd's collision finding algorithm (complexity)

- This algorithm finds a collision in $3\left\lceil\frac{\alpha}{\delta}\right\rceil.\delta + 2\alpha$ hash evaluations
- As both α and δ are $\mathcal{O}(\sqrt{2^n})$, so is Floyd's algorithm time complexity

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 29 / 34

Floyd's collision finding algorithm (complexity)

- This algorithm finds a collision in $3\left\lceil\frac{\alpha}{\delta}\right\rceil.\delta + 2\alpha$ hash evaluations
- As both α and δ are $\mathcal{O}(\sqrt{2^n})$, so is Floyd's algorithm time complexity
- Memory requirement is negligible

What's happened ?

- Weaknesses were known on MD4, MD5, RIPEMD since the nineties...

What's happened ?

- Weaknesses were known on MD4, MD5, RIPEMD since the nineties...
  - ...but they did not lead to practical attacks

What's happened ?

- Weaknesses were known on MD4, MD5, RIPEMD since the nineties...
  - ...but they did not lead to practical attacks
- In summer 2004, a team of chinese researchers announced new collision attacks on MD4, MD5, RIPEMD and HAVAL-128

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
A burst of new attacks

What's happened ?

- Weaknesses were known on MD4, MD5, RIPEMD since the nineties...
  - ...but they did not lead to practical attacks
- In summer 2004, a team of chinese researchers announced new collision attacks on MD4, MD5, RIPEMD and HAVAL-128
- The crypto community was stunned ...

C. Clavier - F. Bruguier (Unilim - UM) An Introduction to Hash Functions Hanoi, April 2020 31 / 34

What's happened ?

- Weaknesses were known on MD4, MD5, RIPEMD since the nineties. . .
  - . . . but they did not lead to practical attacks
- In summer 2004, a team of chinese researchers announced new collision attacks on MD4, MD5, RIPEMD and HAVAL-128
- The crypto community was stunned . . .
- Extending some previously used collision search techniques, they were able to compute new collisions in a matter of minutes, even seconds

Results so far . . .

- Collisions can be found on MD4 with only 3 hash function computations!

Results so far . . .

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
A burst of new attacks

Results so far . . .

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)
  - Complexity 2^{36} for the first block

C. Clavier - F. Bruguier (Unilim - UM) An Introduction to Hash Functions Hanoi, April 2020 32 / 34

Results so far . . .

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)
  - Complexity 2^{36} for the first block
  - Complexity as low as 2^{27} for the second block

Outline · What is a Hash Function ? · Security of Hash Functions · Applications · Generic Attacks · Some Dedicated Attacks
A burst of new attacks

Results so far . . .

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)
  - Complexity 2^{36} for the first block
  - Complexity as low as 2^{27} for the second block
  - The second step takes 30 minutes on a regular PC

C. Clavier - F. Bruguier (Unilim - UM) An Introduction to Hash Functions Hanoi, April 2020 32 / 34

Results so far ...

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)
  - Complexity 2^{36} for the first block
  - Complexity as low as 2^{27} for the second block
  - The second step takes 30 minutes on a regular PC
- Collisions are now found on SHA-0 in 2^{39} computations

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
A burst of new attacks

Results so far ...

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)
  - Complexity 2^{36} for the first block
  - Complexity as low as 2^{27} for the second block
  - The second step takes 30 minutes on a regular PC
- Collisions are now found on SHA-0 in 2^{39} computations
- Collisions were estimated on SHA-1 in about 2^{69} computations

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 32 / 34

Results so far . . .

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)
  - Complexity 2^{36} for the first block
  - Complexity as low as 2^{27} for the second block
  - The second step takes 30 minutes on a regular PC
- Collisions are now found on SHA-0 in 2^{39} computations
- Collisions were estimated on SHA-1 in about 2^{69} computations
  - Crypto 2005 rump session: (theoretical) collisions in 2^{63} computations

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
A burst of new attacks

Results so far . . .

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)
  - Complexity 2^{36} for the first block
  - Complexity as low as 2^{27} for the second block
  - The second step takes 30 minutes on a regular PC
- Collisions are now found on SHA-0 in 2^{39} computations
- Collisions were estimated on SHA-1 in about 2^{69} computations
  - Crypto 2005 rump session: (theoretical) collisions in 2^{63} computations
  - Eurocrypt 2009 rump session: (theoretical) collisions in 2^{52} computations !

C. Clavier - F. Bruguier (Unilim - UM) | An Introduction to Hash Functions | Hanoi, April 2020 | 32 / 34

Results so far . . .

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)
  - Complexity 2^{36} for the first block
  - Complexity as low as 2^{27} for the second block
  - The second step takes 30 minutes on a regular PC
- Collisions are now found on SHA-0 in 2^{39} computations
- Collisions were estimated on SHA-1 in about 2^{69} computations
  - Crypto 2005 rump session: (theoretical) collisions in 2^{63} computations
  - Eurocrypt 2009 rump session: (theoretical) collisions in 2^{52} computations !
  - A collision on SHA-1 has been found by Google only on February 2017

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
A burst of new attacks

Results so far . . .

- Collisions can be found on MD4 with only 3 hash function computations!
- Collisions can be found on MD5 using 2-block messages (*i.e.* 1024 bits)
  - Complexity 2^{36} for the first block
  - Complexity as low as 2^{27} for the second block
  - The second step takes 30 minutes on a regular PC
- Collisions are now found on SHA-0 in 2^{39} computations
- Collisions were estimated on SHA-1 in about 2^{69} computations
  - Crypto 2005 rump session: (theoretical) collisions in 2^{63} computations
  - Eurocrypt 2009 rump session: (theoretical) collisions in 2^{52} computations !
  - A collision on SHA-1 has been found by Google only on February 2017
- Nothing announced (yet) on recent SHA-2 hash functions family

Toward a new standard. . .

An open competition has been launched by the NIST on November 2, 2007 to define a new hash function standard SHA-3 (instead of SHA-1 and SHA-2)

Outline What is a Hash Function ? Security of Hash Functions Applications Generic Attacks Some Dedicated Attacks
The SHA-3 competition

Toward a new standard. . .

An open competition has been launched by the NIST on November 2, 2007 to define a new hash function standard SHA-3 (instead of SHA-1 and SHA-2)

- Fall 2008, 51 submissions were accepted by NIST to Round 1 (of which 5 from France)

C. Clavier - F. Bruguier (Unilim - UM) An Introduction to Hash Functions Hanoi, April 2020 33 / 34

Toward a new standard. . .

An open competition has been launched by the NIST on November 2, 2007 to define a new hash function standard SHA-3 (instead of SHA-1 and SHA-2)

- Fall 2008, 51 submissions were accepted by NIST to Round 1
  (of which 5 from France)
- 14 candidates accepted to Round 2 on July 24, 2009:

Toward a new standard. . .

An open competition has been launched by the NIST on November 2, 2007 to define a new hash function standard SHA-3 (instead of SHA-1 and SHA-2)

- Fall 2008, 51 submissions were accepted by NIST to Round 1
  (of which 5 from France)
- 14 candidates accepted to Round 2 on July 24, 2009:
  - BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, Skein

Toward a new standard. . .

An open competition has been launched by the NIST on November 2, 2007 to define a new hash function standard SHA-3 (instead of SHA-1 and SHA-2)

- Fall 2008, 51 submissions were accepted by NIST to Round 1
  (of which 5 from France)
- 14 candidates accepted to Round 2 on July 24, 2009:
  - BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, Skein
- 5 finalist candidates accepted to Round 3 on December 9, 2010:

Outline | What is a Hash Function ? | Security of Hash Functions | Applications | Generic Attacks | Some Dedicated Attacks
The SHA-3 competition

Toward a new standard. . .

An open competition has been launched by the NIST on November 2, 2007 to define a new hash function standard SHA-3 (instead of SHA-1 and SHA-2)

- Fall 2008, 51 submissions were accepted by NIST to Round 1
  (of which 5 from France)
- 14 candidates accepted to Round 2 on July 24, 2009:
  - BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, Skein
- 5 finalist candidates accepted to Round 3 on December 9, 2010:
  - BLAKE, Grøstl, JH, Keccak, Skein

Toward a new standard. . .

An open competition has been launched by the NIST on November 2, 2007 to define a new hash function standard SHA-3 (instead of SHA-1 and SHA-2)

- Fall 2008, 51 submissions were accepted by NIST to Round 1
  (of which 5 from France)
- 14 candidates accepted to Round 2 on July 24, 2009:
  - BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, Skein
- 5 finalist candidates accepted to Round 3 on December 9, 2010:
  - BLAKE, Grøstl, JH, Keccak, Skein
- Proclamation of the winner on October 2, 2012:
  (standardized on August 5, 2015)

KECCAK

An Introduction to Hash Functions

Christophe Clavier - Florent Bruguier

University of Limoges - University of Montpellier