Applications of Hash Functions

Venkatramani Rajgopal

Department of Mathematics
University of Applied Sciences, Mittweida

25 April 2016

Outline

Introduction
Message Integrity

- The cryptography systems that we have seen so far, provide *secrecy* or *confidentiality* but not *integrity*. However in some instances we may need integrity.

Introduction
Message Integrity

- The cryptography systems that we have seen so far, provide *secrecy* or *confidentiality* but not *integrity*. However in some instances we may need integrity.

Example

Alice may write a will to distribute her estate upon her death. After her death, anyone can examine the will. Therefore the integrity of the will needs to be preserved. Alice does not want the contents of the will to change.

Introduction

Message Integrity

- The cryptography systems that we have seen so far, provide *secrecy* or *confidentiality* but not *integrity*. However in some instances we may need integrity.

Example

Alice may write a will to distribute her estate upon her death. After her death, anyone can examine the will. Therefore the integrity of the will needs to be preserved. Alice does not want the contents of the will to change.

One way to preserve the integrity of the document is through use of **fingerprints**.

Introduction

Message and Message Digest

Introduction

Message and Message Digest

- The electronic equivalent of the document and fingerprint is the message and digest pair. To preserve the *integrity* of the message it is passed through an algorithm called **Cryptographic Hash Function**.

Introduction

Message and Message Digest

- The electronic equivalent of the document and fingerprint is the message and digest pair. To preserve the *integrity* of the message it is passed through an algorithm called **Cryptographic Hash Function**.
  The function creates a compressed imagine of the message the can be used like a fingerprint. The hash creates a fingerprint of the data, often referred to as the **message digest**.

Introduction

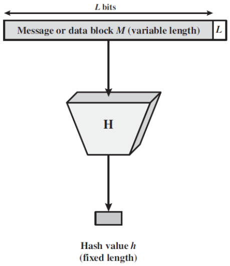
Message and Message Digest

- The electronic equivalent of the document and fingerprint is the message and digest pair. To preserve the *integrity* of the message it is passed through an algorithm called **Cryptographic Hash Function**.
  The function creates a compressed imagine of the message the can be used like a fingerprint. The hash creates a fingerprint of the data, often referred to as the **message digest**.
  A hash function H accepts a variable-length block of data M as input and produces a fixed size hash value $h = H(M)$

Introduction

Message and Message Digest

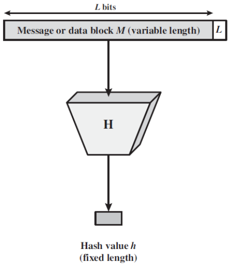
- The electronic equivalent of the document and fingerprint is the message and digest pair. To preserve the *integrity* of the message it is passed through an algorithm called **Cryptographic Hash Function**.
  The function creates a compressed imagine of the message the can be used like a fingerprint. The hash creates a fingerprint of the data, often referred to as the **message digest**.
  A hash function H accepts a variable-length block of data M as input and produces a fixed size hash value $h = H(M)$



Introduction

Message and Message Digest

- The electronic equivalent of the document and fingerprint is the message and digest pair. To preserve the *integrity* of the message it is passed through an algorithm called **Cryptographic Hash Function**.
  The function creates a compressed imagine of the message the can be used like a fingerprint. The hash creates a fingerprint of the data, often referred to as the **message digest**.
  A hash function H accepts a variable-length block of data M as input and produces a fixed size hash value $h = H(M)$



Introduction
Cryptographic Hash function Criteria

For its various applications, the hash function need to satisfy certain properties namely:

Introduction

Cryptographic Hash function Criteria

For its various applications, the hash function need to satisfy certain properties namely:

- 1) **Preimage Resistance:** It should be difficult to compute the preimage of the hashed value.
  A hash function is said to be preimage resistant, if the given value of $y = h(x)$, for some x it is difficult to compute the value of any x' such that $h(x') = y$.

Introduction

Cryptographic Hash function Criteria

For its various applications, the hash function need to satisfy certain properties namely:

- 1) **Preimage Resistance:** It should be difficult to compute the preimage of the hashed value.
  A hash function is said to be preimage resistant, if the given value of $y = h(x)$, for some x it is difficult to compute the value of any x' such that $h(x') = y$.
- 2) **Second Preimage Resistance:** For any values x and y such that $y = h(x)$, it is computationally infeasible to find $x' \neq x$ such that $y = h(x')$

Introduction

Cryptographic Hash function Criteria

For its various applications, the hash function need to satisfy certain properties namely:

- 1) **Preimage Resistance:** It should be difficult to compute the preimage of the hashed value.
  A hash function is said to be preimage resistant, if the given value of $y = h(x)$, for some x it is difficult to compute the value of any x' such that $h(x') = y$.
- 2) **Second Preimage Resistance:** For any values x and y such that $y = h(x)$, it is computationally infeasible to find $x' \neq x$ such that $y = h(x')$
- 3) **Collision resistance:** A hash function H is collision resistant if it is hard to find two inputs that hash to the same output; that is, two inputs x and y such that $H(x) = H(y)$, and $x \neq y$.

Introduction

Hashes are "Digests", not "Encryption"

Introduction

Hashes are "Digests", not "Encryption"

- Encryption transforms data from a cleartext to ciphertext and back. **Encryption is a two-way operation**.

Introduction

Hashes are "Digests", not "Encryption"

- Encryption transforms data from a cleartext to ciphertext and back. **Encryption is a two-way operation**.
- Hashes, on the other hand, compile a stream of data into a small **digest** and it's strictly a **one way operation.**

Introduction

Hashes are "Digests", not "Encryption"

- Encryption transforms data from a cleartext to ciphertext and back. **Encryption is a two-way operation**.
- Hashes, on the other hand, compile a stream of data into a small **digest** and it's strictly a **one way operation.**

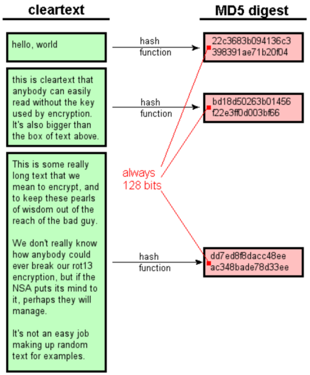

cleartext
MD5 digest
hello, world
hash function
22c3683b094136c3 398391ae71b20f04
this is cleartext that anybody can easily read without the key used by encryption. It's also bigger than the box of text above.
hash function
bd18d50263b01456 f22e3ff0d003bf66
This is some really long text that we mean to encrypt, and to keep these pearls of wisdom out of the reach of the bad guy. We don't really know how anybody could ever break our rot13 encryption, but if the NSA puts its mind to it, perhaps they will manage. It's not an easy job making up random text for examples.
hash function
dd7ed8f8dacc48ee ac348bade78d33ee
always 128 bits

Figure: Hashing - a one-way operation

Random Oracle

The Random Oracle model which was introduced in 1993 by *"Bellare and Rogaway"* is an ideal model of the hash function. A function based on this model behaves as follows:

Random Oracle

The Random Oracle model which was introduced in 1993 by *"Bellare and Rogaway"* is an ideal model of the hash function. A function based on this model behaves as follows:

- 1) When a new message of any length is given, the oracle creates and gives a fixed length message digest that is random string of 0's and 1's.

Random Oracle

The Random Oracle model which was introduced in 1993 by *"Bellare and Rogaway"* is an ideal model of the hash function. A function based on this model behaves as follows:

- 1) When a new message of any length is given, the oracle creates and gives a fixed length message digest that is random string of 0's and 1's.
- 2) When a message is given for which a digest exists, the oracle simply gives the digest in the record.

Random Oracle

The Random Oracle model which was introduced in 1993 by *"Bellare and Rogaway"* is an ideal model of the hash function. A function based on this model behaves as follows:

- 1) When a new message of any length is given, the oracle creates and gives a fixed length message digest that is random string of 0's and 1's.
- 2) When a message is given for which a digest exists, the oracle simply gives the digest in the record.
- 3) The digest for a new message needs to be chosen independently from all previous digests.

Random Oracle

The Random Oracle model which was introduced in 1993 by *"Bellare and Rogaway"* is an ideal model of the hash function. A function based on this model behaves as follows:

- 1) When a new message of any length is given, the oracle creates and gives a fixed length message digest that is random string of 0's and 1's.
- 2) When a message is given for which a digest exists, the oracle simply gives the digest in the record.
- 3) The digest for a new message needs to be chosen independently from all previous digests.

Example

Here is an example which shows the Message and Message Digest listed in Hexadecimal.

Message	Message Digest
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB45CD1048765412AAAB6662BE	A38B

Example

The message "AB1234CD8765BDAD" is given for digest calculation. The oracle checks its table. This messgage is not in the table, so the oracle flips its coin 16 times.

Example

The message "AB1234CD8765BDAD" is given for digest calculation. The oracle checks its table. This messgage is not in the table, so the oracle flips its coin 16 times. Assume that the result is HHTHHHTTHTHHTTTH, in which the letter H represents "Heads" and T represents "Tails". The oracle interprets H as 1-bit and T as 0-bit and gives 1101110010110001 in binary or DCB1 in hexadecimal as a message and adds it to the table.

Example

The message "AB1234CD8765BDAD" is given for digest calculation. The oracle checks its table. This messgage is not in the table, so the oracle flips its coin 16 times. Assume that the result is HHTHHHTTHTHHTTTH, in which the letter H represents "Heads" and T represents "Tails". The oracle interprets H as 1-bit and T as 0-bit and gives 1101110010110001 in binary or DCB1 in hexadecimal as a message and adds it to the table.

Message	Message Digest
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB1234CD8765BDAD	DCB1
AB45CD1048765412AAAB6662BE	A38B

Iterated Hash Function

Iterated Hash Function

- All cryptographic hash functions need to create a *fixed-size digest* out of a variable size message. Creating such a function can be accomplished using **Iteration.**

Iterated Hash Function

- All cryptographic hash functions need to create a *fixed-size digest* out of a variable size message. Creating such a function can be accomplished using **Iteration.**
- Instead of using a variable size input, a function with fixed size input is created and is used number of times.

Iterated Hash Function

- All cryptographic hash functions need to create a *fixed-size digest* out of a variable size message. Creating such a function can be accomplished using **Iteration.**
- Instead of using a variable size input, a function with fixed size input is created and is used number of times.
- The fixed size input function is referred to as a **compression function**. It compresses an $n-$bit string to create an $m-$bit string where generally $n > m$.

Iterated Hash Function

- All cryptographic hash functions need to create a *fixed-size digest* out of a variable size message. Creating such a function can be accomplished using **Iteration.**
- Instead of using a variable size input, a function with fixed size input is created and is used number of times.
- The fixed size input function is referred to as a **compression function**. It compresses an $n-$bit string to create an $m-$bit string where generally $n > m$.
- We call this scheme as **iterated cryptographic hash function.**

Iterated Hash Function

Merkle-Damgard Scheme

Merkle-Damgard Scheme is an iterated hash function that is **collision resistant** if the compression function is collision resistant. The scheme uses the following steps.

- The message is augmented and divided into blocks of n bits, where n is the size of the block to be processed by the compression function.

Iterated Hash Function

Merkle-Damgard Scheme

Merkle-Damgard Scheme is an iterated hash function that is **collision resistant** if the compression function is collision resistant. The scheme uses the following steps.

- The message is augmented and divided into blocks of n bits, where n is the size of the block to be processed by the compression function.
- We call each block $M_1, M_2 .... M_t$. We call the digest created at t iterations $H_1, H_2 .... H_t$

Iterated Hash Function

Merkle-Damgard Scheme

Merkle-Damgard Scheme is an iterated hash function that is **collision resistant** if the compression function is collision resistant. The scheme uses the following steps.

- The message is augmented and divided into blocks of n bits, where n is the size of the block to be processed by the compression function.
- We call each block $M_1, M_2 .... M_t$. We call the digest created at t iterations $H_1, H_2 .... H_t$
- The *compression function* operates at each iteration on H_{i-1} and M_i to create new H_i. i.e we have $H_i = f(H_{i-1}, M_i)$, where f is the compression function.

Iterated Hash Function

Merkle-Damgard Scheme

Merkle-Damgard Scheme is an iterated hash function that is **collision resistant** if the compression function is collision resistant. The scheme uses the following steps.

- The message is augmented and divided into blocks of n bits, where n is the size of the block to be processed by the compression function.
- We call each block $M_1, M_2....M_t$. We call the digest created at t iterations $H_1, H_2....H_t$
- The *compression function* operates at each iteration on H_{i-1} and M_i to create new H_i. i.e we have $H_i = f(H_{i-1}, M_i)$, where f is the compression function.
- H_t is the cryptographic hash function of the original message, that is $h(M)$.

Iterated Hash Function

Merkle-Damgard Scheme

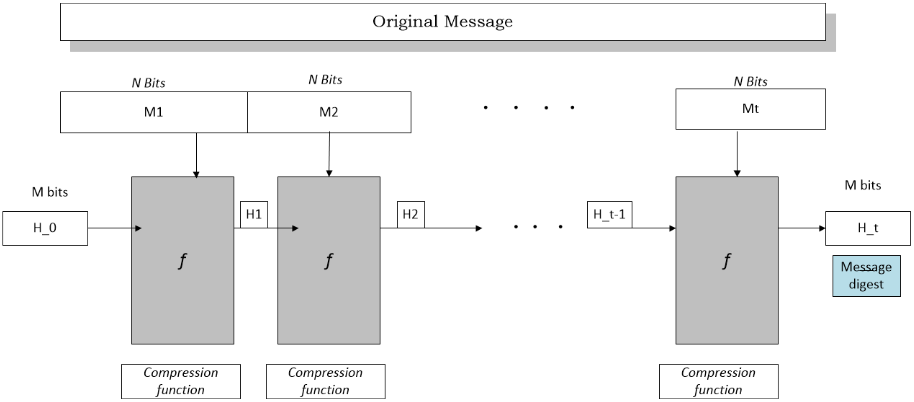


Figure: Merkle-Damgard scheme

Message Authentication

- A message digest guarantees the integrity of a message, guaranteeing that the message is not changed.

Message Authentication

- A message digest guarantees the integrity of a message, guaranteeing that the message is not changed.
- When Alice sends a message to Bob, Bob needs to know if the message is coming from Alice.

Message Authentication

- A message digest guarantees the integrity of a message, guaranteeing that the message is not changed.
- When Alice sends a message to Bob, Bob needs to know if the message is coming from Alice.
- Now to provide such a proof, Alice uses a hash function to create a **Message Authentication code (MAC)** from the concatenation of the key and the message, $h(K||M)$.

Message Authentication - Implementation Steps

- Alice uses a hash function to create a MAC from the key and message. The key is a shared secret key.

Message Authentication - Implementation Steps

- Alice uses a hash function to create a MAC from the key and message. The key is a shared secret key.
- She sends the message and the MAC to Bob over an insecure channel.

Message Authentication - Implementation Steps

- Alice uses a hash function to create a MAC from the key and message. The key is a shared secret key.
- She sends the message and the MAC to Bob over an insecure channel.
- Bob separates the message from the MAC. He then makes a new MAC from the concatenation of the message and the secret key and then compares the newly created MAC with the one received.

Message Authentication - Implementation Steps

- Alice uses a hash function to create a MAC from the key and message. The key is a shared secret key.
- She sends the message and the MAC to Bob over an insecure channel.
- Bob separates the message from the MAC. He then makes a new MAC from the concatenation of the message and the secret key and then compares the newly created MAC with the one received.
- If two MAC's match, then the message is authentic.

Message Authentication - Implementation

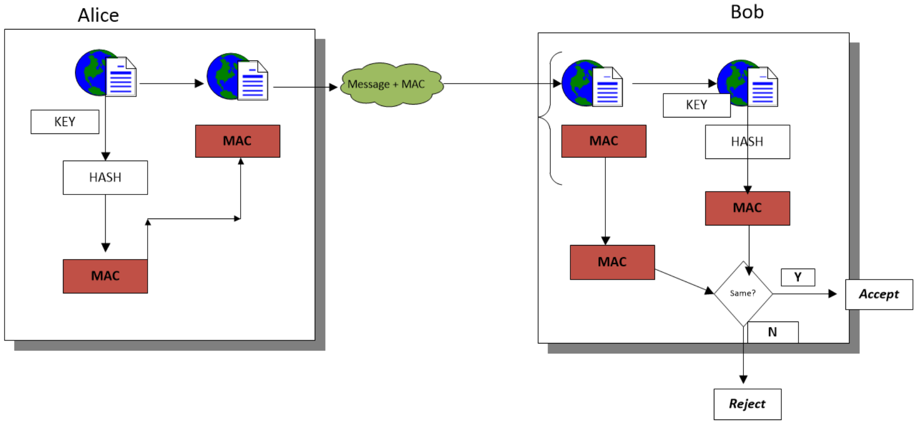


Figure: Message authentication code (MAC)

Digital Signatures

We can use hash functions to generate a signature that guarantees that the message came from a said source.

Digital Signatures

We can use hash functions to generate a signature that guarantees that the message came from a said source.
In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.

Digital Signatures

We can use hash functions to generate a signature that guarantees that the message came from a said source.
In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.

In this case an attacker who wishes to alter the message would need to know the user's private key.

Digital Signatures

We can use hash functions to generate a signature that guarantees that the message came from a said source.
In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.

In this case an attacker who wishes to alter the message would need to know the user's private key.

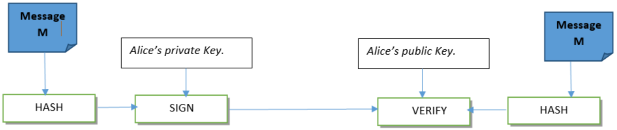


Figure: Signing the message digest

Digital Signatures

A digital signature is a ”stamp” Bob places on the data which is unique to Bob, and is very difficult to forge. In addition, the signature assures that any changes made to the data that has been signed can not go undetected.

Digital Signatures

A digital signature is a "stamp" Bob places on the data which is unique to Bob, and is very difficult to forge. In addition, the signature assures that any changes made to the data that has been signed can not go undetected.

It is not possible to change a message digest back into the original data from which it was created.

Digital Signatures

A digital signature is a "stamp" Bob places on the data which is unique to Bob, and is very difficult to forge. In addition, the signature assures that any changes made to the data that has been signed can not go undetected.

It is not possible to change a message digest back into the original data from which it was created.

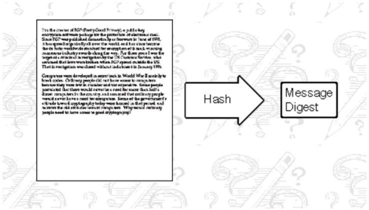
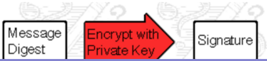


Figure: Converting message to message digest

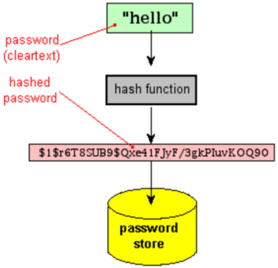


Hashing passwords

- Hash functions are commonly used to create a **one-way password** file. Hash of a password is stored by an operating system rather than the password itself.
- Since these hashes are **not reversible**, there is no way to find out for sure "what password produced this hash?"

Hashing passwords

- Hash functions are commonly used to create a **one-way password** file. Hash of a password is stored by an operating system rather than the password itself.
- Since these hashes are **not reversible**, there is no way to find out for sure "what password produced this hash?"
- In simple terms, when a user enters a password, the hash of that password is compared to the stored hash value for verification. This approach to **password protection** is used by most operating systems.



Hashing passwords

How will we know that some future user at a login prompt gives us the same password?

Hashing passwords

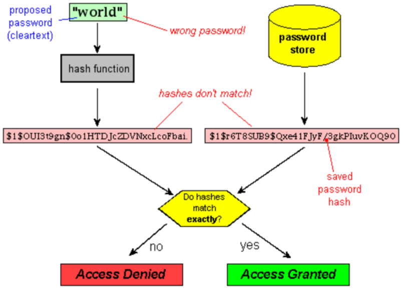
How will we know that some future user at a login prompt gives us the same password?

The answer is simple: Take the proposed password in cleartext, run it through the same hash function, and see whether this result matches the hash saved in the password store. If they match, the user must have known the proper password, so access is granted, but if the hashes are not identical, access is denied.

Hashing passwords

How will we know that some future user at a login prompt gives us the same password?

The answer is simple: Take the proposed password in cleartext, run it through the same hash function, and see whether this result matches the hash saved in the password store. If they match, the user must have known the proper password, so access is granted, but if the hashes are not identical, access is denied.



References I

Behrouz A Forouzan, Debdeep Mukhopadhyay
Cryptography and Network Security.
McGraw Hill Edition.

William Stallings.
Cryptography and Network Security
Principles and Practice.
5th Edition.

Thank you for your attention.