CSC 250: Foundations of Computer Science I

Fall 2023 - Lecture 2

Amitabha Dey

Department of Computer Science
University of North Carolina at Greensboro

Encryption

Encryption is the practice of scrambling information in a way that only someone with a corresponding key can unscramble and read it. Encryption is a two-way function. When you encrypt something, you're doing so with the intention of decrypting it later.

To encrypt data you use something called a cipher, which is an algorithm – a series of well-defined steps that can be followed procedurally – to encrypt and decrypt information. The algorithm can also be called the encryption key.

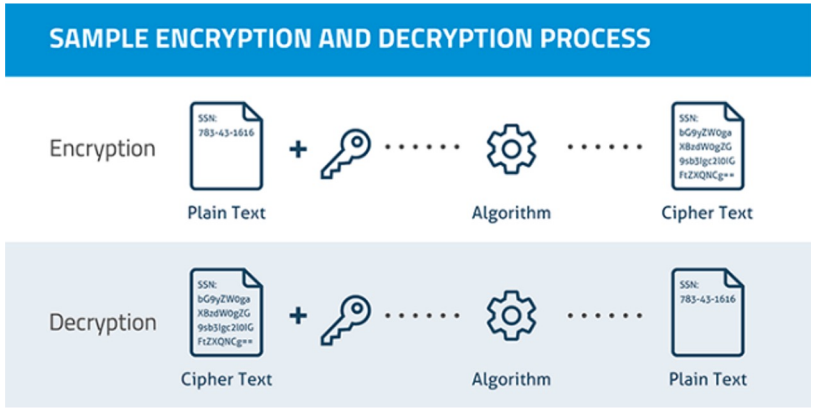


Image credit - https://mediawiki.middlebury.edu/

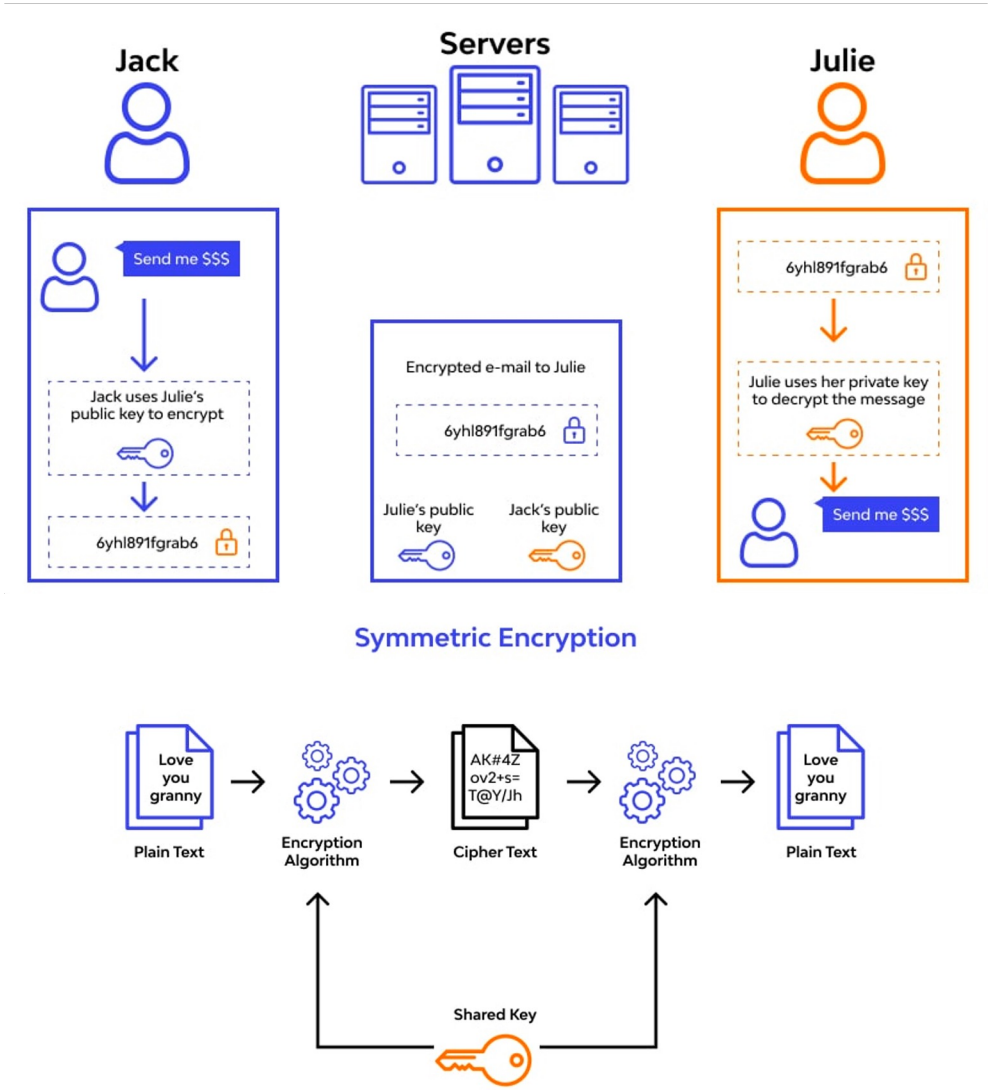


Image credit - https://www.wallarm.com/

Caesar Cipher

In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions. For instance, here is a Caesar cipher using a left rotation of three places, equivalent to a right shift of 23 (the shift parameter is used as the key):

Plain	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

When encrypting, a person looks up each letter of the message in the "plain" line and writes down the corresponding letter in the "cipher" line.

```
Plaintext:  THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
Ciphertext: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD
```

Deciphering is done in reverse, with a right shift of 3.

Image credit - https://en.wikipedia.org/

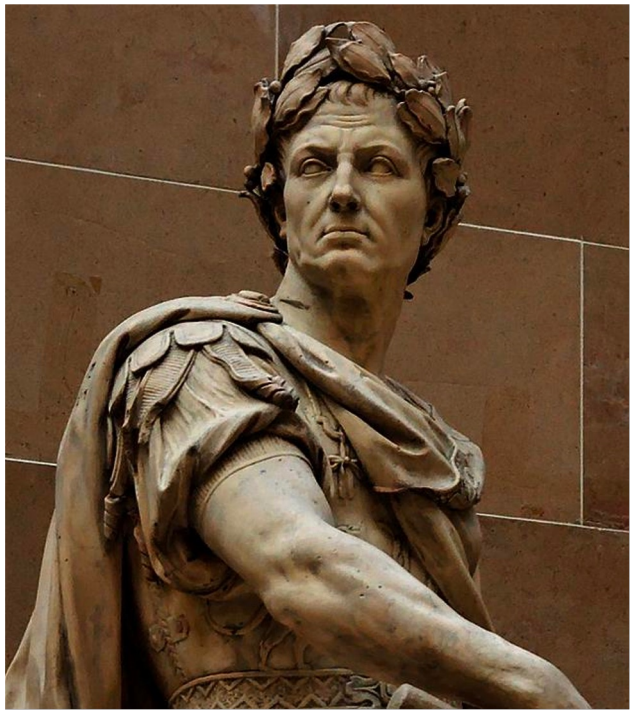


Image credit - https://historylearning.com/

Hill Cipher

Hill cipher is a polygraphic substitution cipher based on linear algebra.Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.
The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

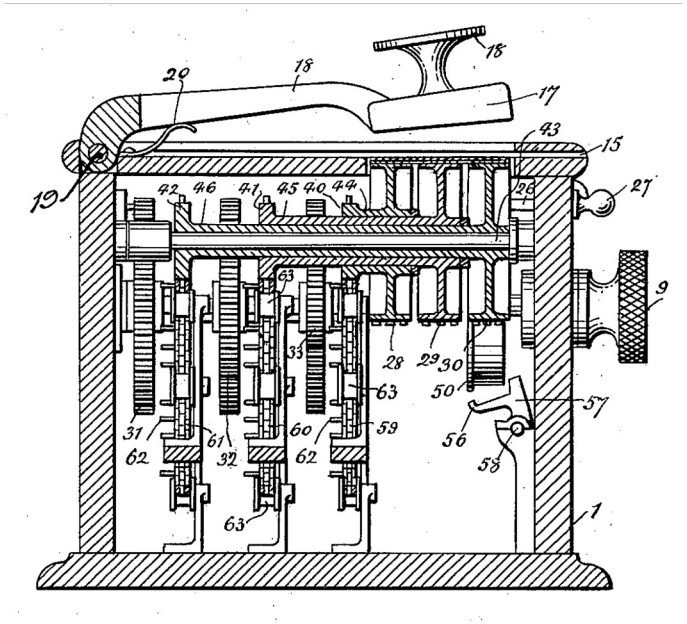


Image credit - https://en.wikipedia.org/

- **Encryption :**
  Cipher Text =(Plain Text x Key) Mod 26
- **Decryption:**
  Plain Text =(Cipher Text x Key^{-1}) Mod 26
- **Message:** ATTACK IS TONIGHT
- **Key** $= \begin{bmatrix} 3 & 10 & 20 \\ 20 & 9 & 17 \\ 9 & 4 & 17 \end{bmatrix}$

- Message: ATTACK IS TONIGHT
- **Cipher Text = (Plain Text x Key) Mod 26**

$$= \begin{bmatrix} 0 & 19 & 19 \\ 0 & 2 & 10 \\ 8 & 18 & 19 \\ 14 & 13 & 8 \\ 6 & 7 & 19 \end{bmatrix} \times \begin{bmatrix} 3 & 10 & 20 \\ 20 & 9 & 17 \\ 9 & 4 & 17 \end{bmatrix} \text{ Mod } 26$$

Image credit - https://www.cybrary.it/

- $= \begin{bmatrix} 0 & 19 & 19 \\ 0 & 2 & 10 \\ 8 & 18 & 19 \\ 14 & 13 & 8 \\ 6 & 7 & 19 \end{bmatrix} \times \begin{bmatrix} 3 & 10 & 20 \\ 20 & 9 & 17 \\ 9 & 4 & 17 \end{bmatrix}$ Mod 26

C_{11}= 0*3 + 19*20 + 19*9 = 551 Mod 26 = 05
C_{12}= 0*10 + 19*9 + 19*4 = 247 Mod 26 = 13
C_{13}= 0*20 + 19*17 + 19*17 = 646 Mod 26 = 22

ATT → FNW *Similarly you can calculate other...*

SQL Injection

SQL is “Structured Query Language”. It is a standardized language for accessing databases. Every programming language implements SQL functionality in its own way.

Examples of SQL statements -

- select name from employee where ssn=‘123456789’
- select name, ssn, dob from employee where ssn=‘123456789’ and id=‘31042’
- select code,namefrom products where code =‘536’ union select code,name from sales where code > ‘500’

Assume that the select statement implemented is:

- res = select CBalance from Balances where Acct=‘$acct’

$acct is the variable containing the account number input by the user (PHP style naming) This is a typical usage of a select statement to look up a value

Enter your account number	3215
Your balance	

Image credit - CSCI 476, Montana State University

Results in:

- res = select CBalance from Balances where Acct=‘3215’

But what if the user enters something like this

Enter your account number	9999’%20or%20’1’=‘1
Your balance	

%20 is the placeholder for a space

res = select CBalance from Balances where Acct=‘9999’ or ‘1’=‘1’

Since ‘1’=‘1’ is always true, the select statement will return all records

res will contain, depending on the language

- ❑ every record
- ❑ the first record
- ❑ the last record

The attacker will have valuable information for further attacks, such as issuing a transaction against the account number discovered

Hashing

Hashing is the practice of using an algorithm to map data of any size to a fixed length. This is called a hash value. Whereas encryption is a two-way function, hashing is a one-way function. While it's technically possible to reverse-hash something, the computing power required makes it unfeasible. Hashing is one-way.

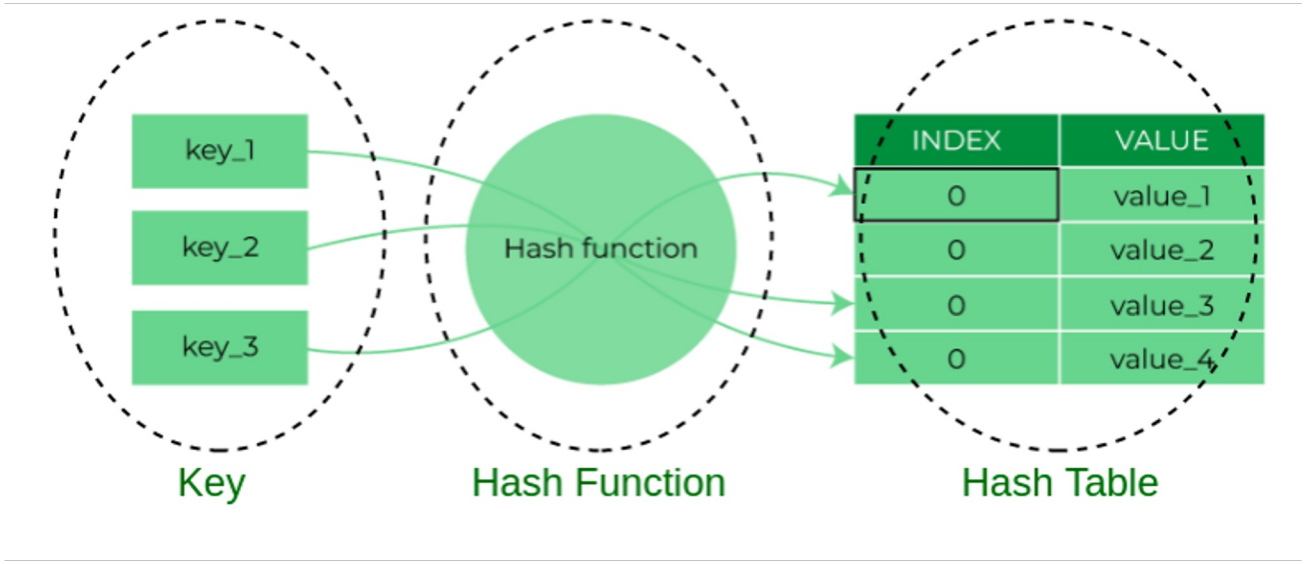


Image credit - https://www.geeksforgeeks.org/

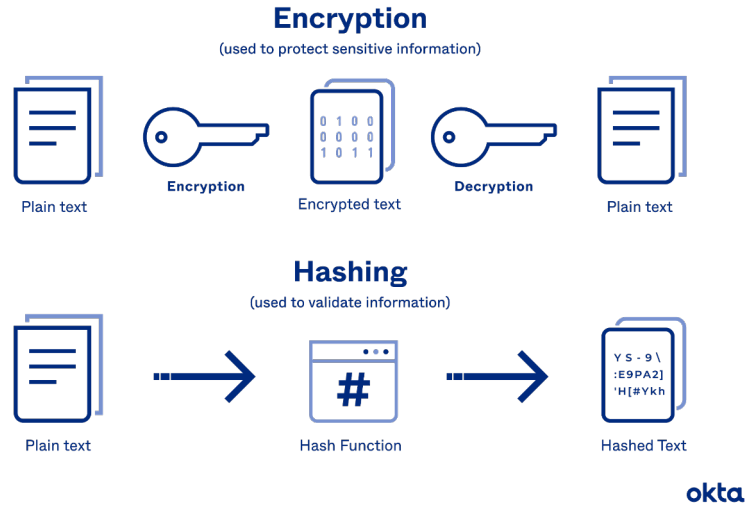


Image credit - https://www.okta.com/

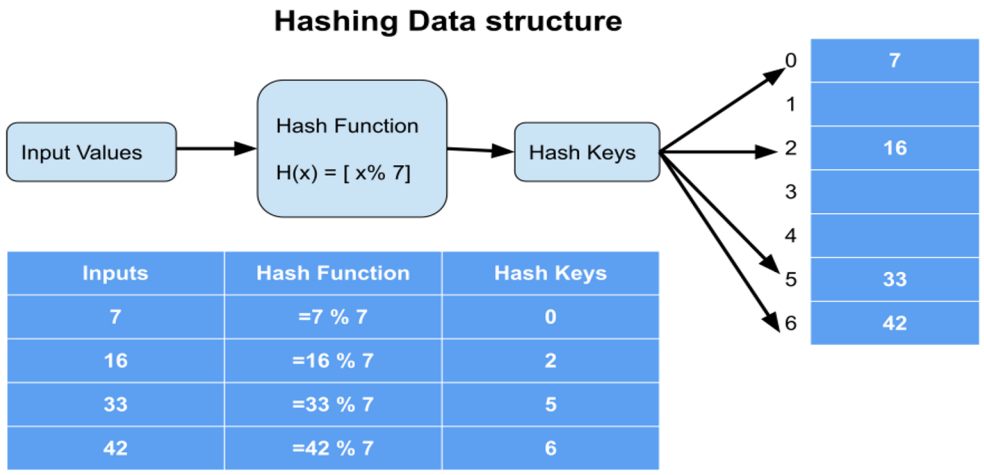
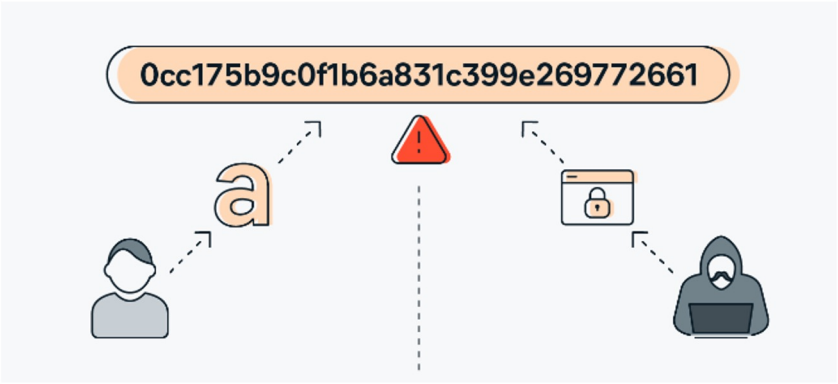


Image credit - https://technicalsand.com/

Popular Hashing Algorithms

- MD-5. The MD5 message-digest algorithm is a widely used hash function producing a 128-bit hash value. Very weak, not safe.
- SHA. The Secure Hash Algorithms are a family of cryptographic hash functions. More secure.

MD5 Hash Generator
https://www.md5hashgenerator.com/



An MD5 collision attack occurs when a hacker intentionally sends a malicious file with the same hash as a clean file.

Image credit - https://www.avast.com/

Algorithm	Word Size	Block Size	Output Size	Rounds	Collision F
MD-2	32	128	128	18	YES
MD-4	32	512	128	48	YES
MD-5	32	512	128	64	YES
SHA-0	32	512	160	80	YES
SHA-1	40	512	160	80	YES
SHA-2	56/64	512/1024	224/256 /384/512	64/80	THEORETI
SHA-3	64	1152/1088 832/576	224/256/384/512	24	NO

Image credit - Security of Password Hashing in Cloud - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Comparison-between-different-hash-algorithms-19_fig2_331380160 [accessed 16 Jan, 2023]

Properties of a Good Hash Function

Main characteristics of a good hash function:

1. Output is deterministic. The same input produces the same output.
2. Non-reversible - it should be very difficult to go backwards from the hash to all of potential inputs.
   One-way function y = h(x). Given y, it is very hard to find x.
1. The hash function generates very different hash values for similar strings.
2. The hash is collision free, meaning it is not feasible that two different messages would result in the same hash value
3. Easy to compute - Given message m, hash function h(m) is easy to compute

Using `password_hash` is the recommended way to store passwords. Don't separate them to DB and files.

Let's say we have the following input:

```
$password = $_POST['password'];
```

You first hash the password by doing this:

```
$hashed_password = password_hash($password, PASSWORD_DEFAULT);
```

Then see the output:

```
var_dump($hashed_password);
```

How to Not Store Passwords

- Storing user passwords in the database as it is in plain text.
- Using an encryption method
- Using a simple hash function

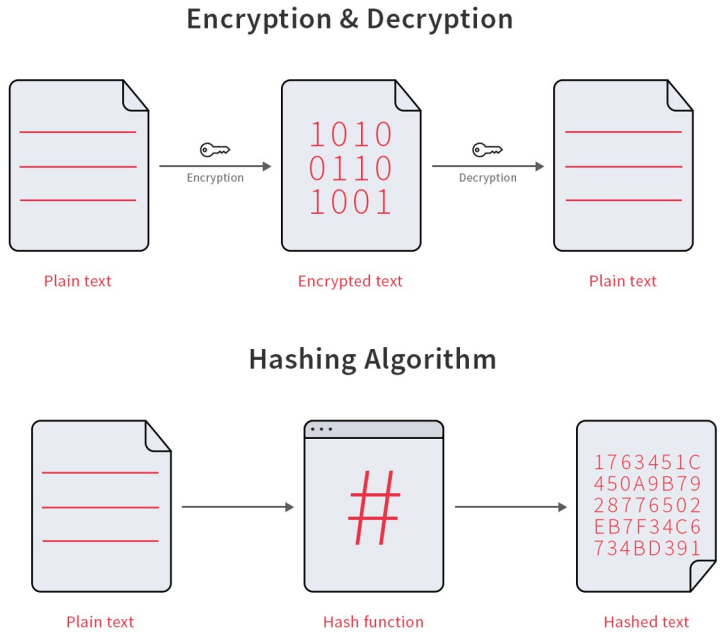


Image credit - https://blog.elcomsoft.com/

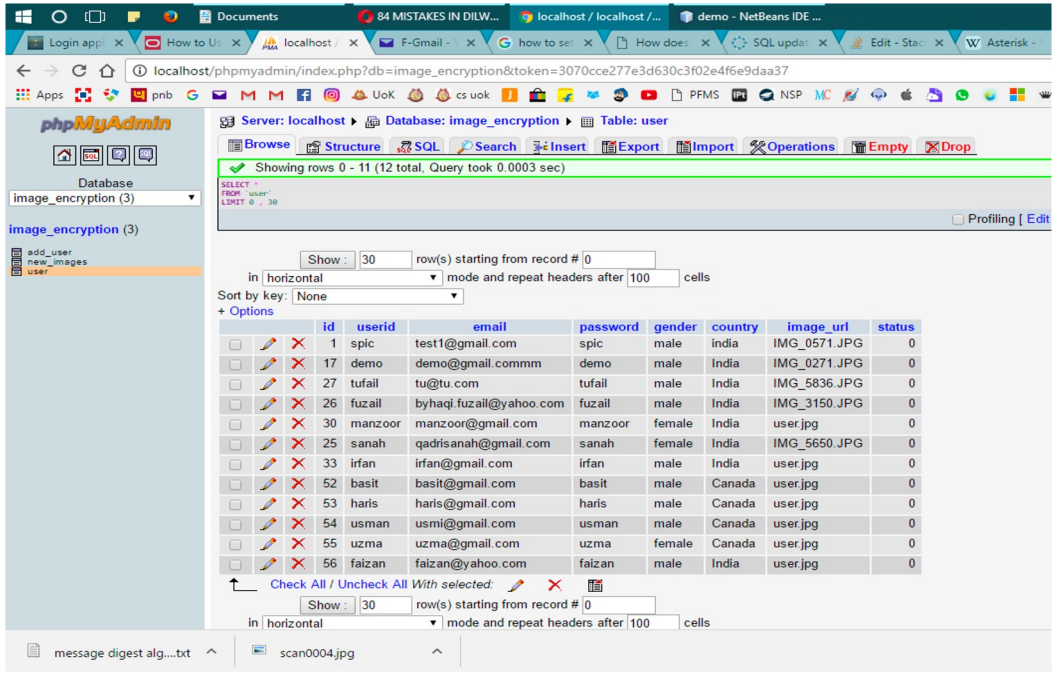


Image credit - https://stackoverflow.com/

A **Rainbow Table** is a set of precomputed passwords and their corresponding hash values that can be used to find out what plaintext password matches a particular hash

Salted Secure Hash Function

A salt is added to the hashing process to force their uniqueness, increase their complexity without increasing user requirements, and to mitigate password attacks like hash tables

Hashed passwords are not unique to themselves due to the deterministic nature of hash function: when given the same input, the same output is always produced. If Alice and Bob both choose `dontpwnme4` as a password, their hash would be the same:

username	hash
alice	4420d1918bbcf7686defdf9560bb5087d20076de5f77b7cb4c3b40bf46ec428b
jason	695ddccd984217fe8d79858dc485b67d66489145afa78e8b27c1451b27cc7a2b
mario	cd5cb49b8b62fb8dca38ff2503798eae71bfb87b0ce3210cf0acac43a3f2883c
teresa	73fb51a0c9be7d988355706b18374e775b18707a8a03f7a61198eefc64b409e8
bob	4420d1918bbcf7686defdf9560bb5087d20076de5f77b7cb4c3b40bf46ec428b
mike	77b177de23f81d37b5b4495046b227befa4546db63cfe6fe541fc4c3cd216eb9

As we can see, `alice` and `bob` have the same password as we can see that both share the same hash: `4420d1918bbcf7686defdf9560bb5087d20076de5f77b7cb4c3b40bf46ec428b`.

The attacker can better predict the password that legitimately maps to that hash. Once the password is known, the same password can be used to access all the accounts that use that hash.

Image credit - https://auth0.com/

Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	lvn49sa	z32i6t0

Image credit - https://auth0.com/

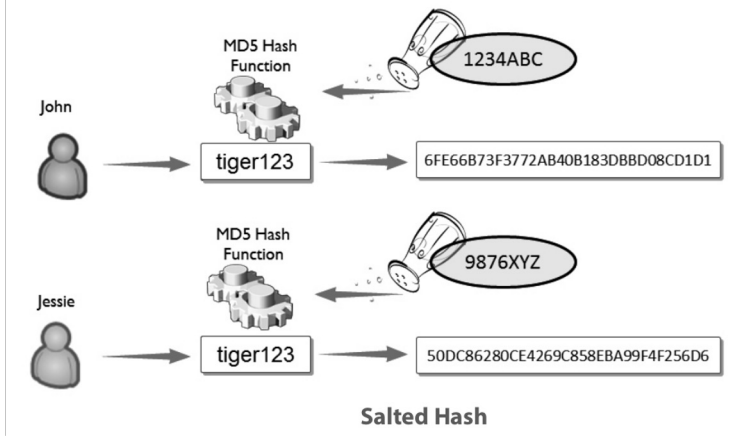


Image credit - https://forum.huawei.com/

Division Method

The division method is generally a reasonable strategy, unless the key happens to have some undesirable properties. For example, if the table size is 10 and all of the keys end in zero.

In this case, the choice of hash function and table size needs to be carefully considered.

The best table sizes are prime numbers.

One problem though is that keys are not always numeric. In fact, it's common for them to be strings.

One possible solution: add up the ASCII values of the characters in the string to get a numeric value and then perform the division method.

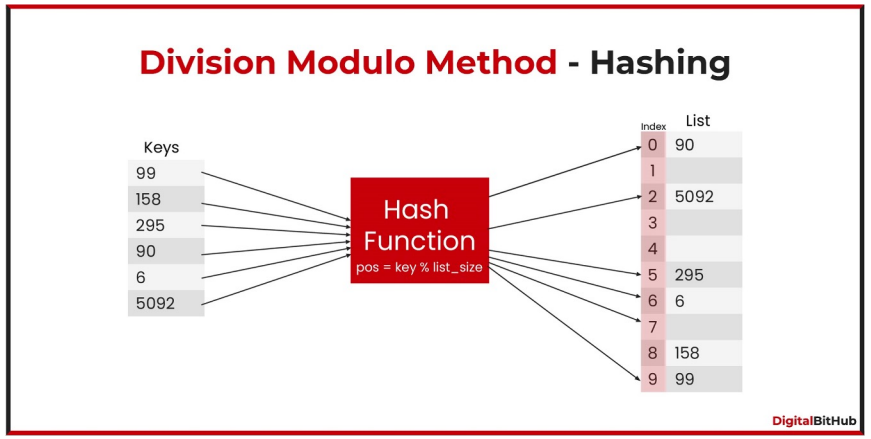


Image credit - https://www.digitalbithub.com/

Assume a table with 8 slots:

Hash key = key % table size

4 = 36 % 8

2 = 18 % 8

0 = 72 % 8

3 = 43 % 8

6 = 6 % 8

[0]	72
[1]	
[2]	18
[3]	43
[4]	36
[5]	
[6]	6
[7]	

Image credit - https://faculty.cs.niu.edu/

Pros:

1. This method is quite good for any value of M.
2. The division method is very fast since it requires only a single division operation.

Cons:

1. This method leads to poor performance since consecutive keys map to consecutive hash values in the hash table.
2. Sometimes extra care should be taken to choose the value of M.

Image credit - https://www.geeksforgeeks.org/

Multiplication Method

Hash Function:
Using Multiplication Method

InterviewBit

Key = 50
Assume c = 0.81, where 0 < c < 1
Assume Table_Size = 1000

hash(Key) = floor(Table_Size * fractional(k * c))

hash(50) = floor(1000 * fractional(50 * 0.81))

50 * 0.81 = 40.5 => Fractional Part = x - floor(x)
= 40.5 - floor(40.5)
= 40.5 - 40
= 0.5

hash(50) = floor(1000 * 0.5) = floor(500) = 500

Place record of key 50 at 500th position in hash table

Image credit - https://www.interviewbit.com/

Pros:

The advantage of the multiplication method is that it can work with any value between 0 and 1, although there are some values that tend to give better results than the rest.

Cons:

The multiplication method is generally suitable when the table size is the power of two, then the whole process of computing the index by the key using multiplication hashing is very fast.

h(K) = floor (M (kA mod 1))

Here,

***M** is the size of the hash table.*

***k** is the key value.*

***A** is a constant value.*

k = 12345
A = 0.357840
M = 100

*h(12345) = floor[100 (12345*0.357840 mod 1)]*
= floor[100 (4417.5348 mod 1)]
= floor[100 (0.5348)]
= floor[53.48]
= 53

Image credit - https://www.geeksforgeeks.org/

Mid Square Method

- Square the value of the key k i.e. k^2
- Extract the middle r digits as the hash value.

Suppose the size of the Hash Table (m) = 10 (0 - 9) maximum digits required for the index is 1
Element (x) = 12 ⇒ x2 = 144
Mid 1 digit of 144 is 4, so the element x=12 will be stored at the index=4 in the hash table with the size of 10 slots.

K=	3205	7148	2345
K2=	10272025	51093904	5499025
H(K)=	72	93	99

Image credit - https://www.learnpick.in/

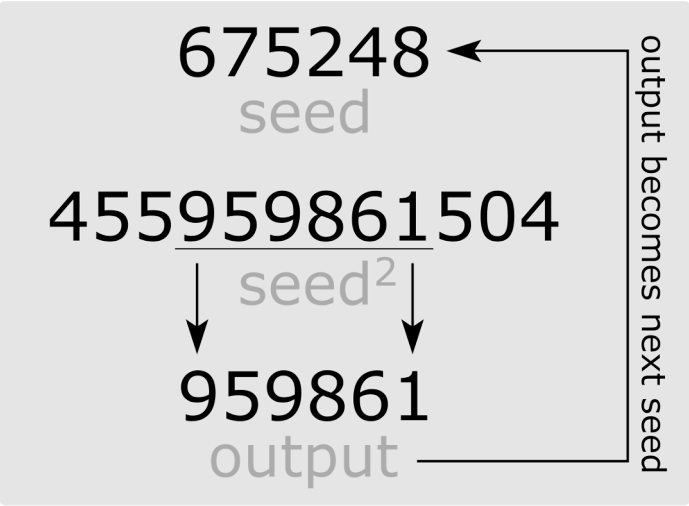


Image credit - https://en.wikipedia.org/

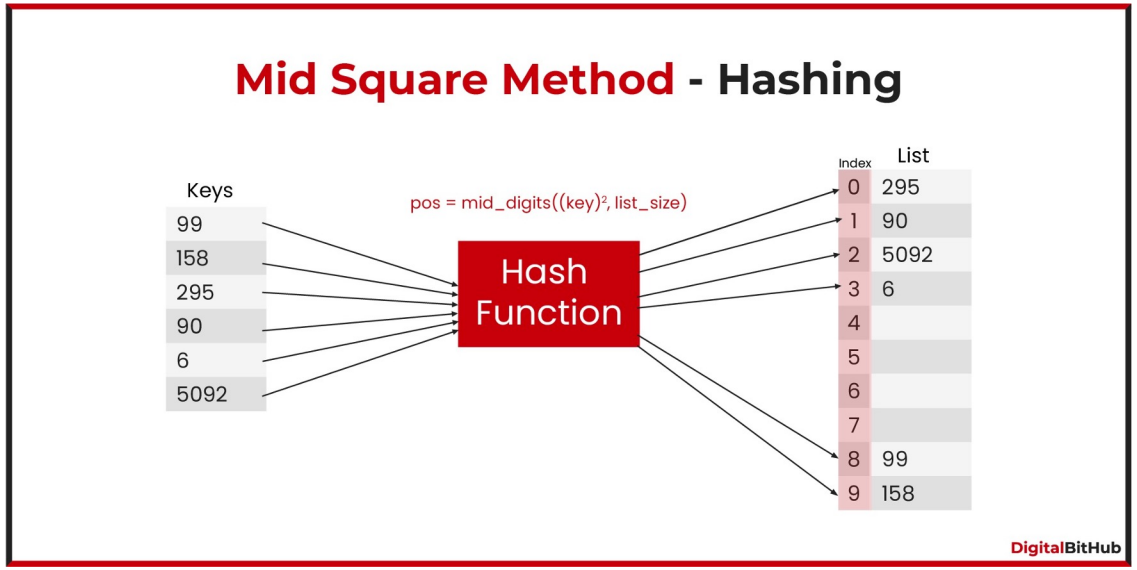


Image credit - https://www.digitalbithub.com/

Resolving Collision using Linear Probing

We use the following hash function to resolve the collision:

h(k, i) = [h(k) + i] mod m

Where,
k = key
m = size of the hash table
h(k) = hash function
i = the probe number that varies from 0 to m−1

Example

Insert the following sequence of keys in the hash table

{9, 7, 11, 13, 12, 8}

Use linear probing technique for collision resolution

h(k, i) = [h(k) + i] mod m

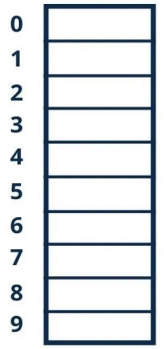
h(k) = 2k + 5

m=10

Step 01:

First Draw an empty hash table of Size 10.

The possible range of hash values will be [0, 9].



Step 02:

Insert the given keys one by one in the hash table.

First Key to be inserted in the hash table = 9.

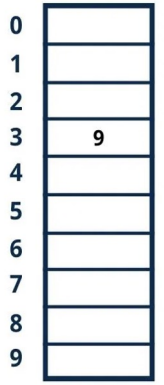
h(k) = 2k + 5

h(9) = 2*9 + 5 = 23

h(k, i) = [h(k) + i] mod m

h(9, 0) = [23 + 0] mod 10 = 3

So, key 9 will be inserted at index 3 of the hash table



Step 03:

Next Key to be inserted in the hash table = 7.

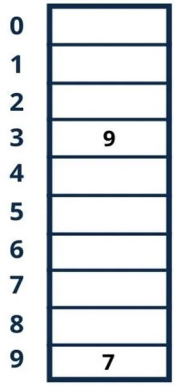
h(k) = 2k + 5

h(7) = 2*7 + 5 = 19

h(k, i) = [h(k) + i] mod m

h(7, 0) = [19 + 0] mod 10 = 9

So, key 7 will be inserted at index 9 of the hash table



Step 04:

Next Key to be inserted in the hash table = 11.

h(k) = 2k + 5

h(11) = 2*11 + 5 = 27

h(k, i) = [h(k) + i] mod m

h(11, 0) = [27 + 0] mod 10 = 7

So, key 11 will be inserted at index 7 of the hash table

0	
1	
2	
3	9
4	
5	
6	
7	11
8	
9	7

Step 05:

Next Key to be inserted in the hash table = 13.

h(k) = 2k + 5

h(13) = 2*13 + 5 = 31

h(k, i) = [h(k) + i] mod m

h(13, 0) = [31 + 0] mod 10 = 1

So, key 13 will be inserted at index 1 of the hash table

0	
1	13
2	
3	9
4	
5	
6	
7	11
8	
9	7

Image credit - https://quescol.com

Step 06:

Next key to be inserted in the hash table = 12.
h(k) = 2k + 5
h(12) = 2*12 + 5 = 29
h(k, i) = [h(k) + i] mod m
h(12, 0) = [29 + 0] mod 10 = 9

Here Collision has occurred because **index 9** is already filled.

Now we will increase **i by 1**.
h(12, 1) = [29 + 1] mod 10 = 0
So, key 12 will be inserted at index 0 of the hash table.

0	
1	13
2	
3	9
4	
5	
6	
7	11
8	12
9	7