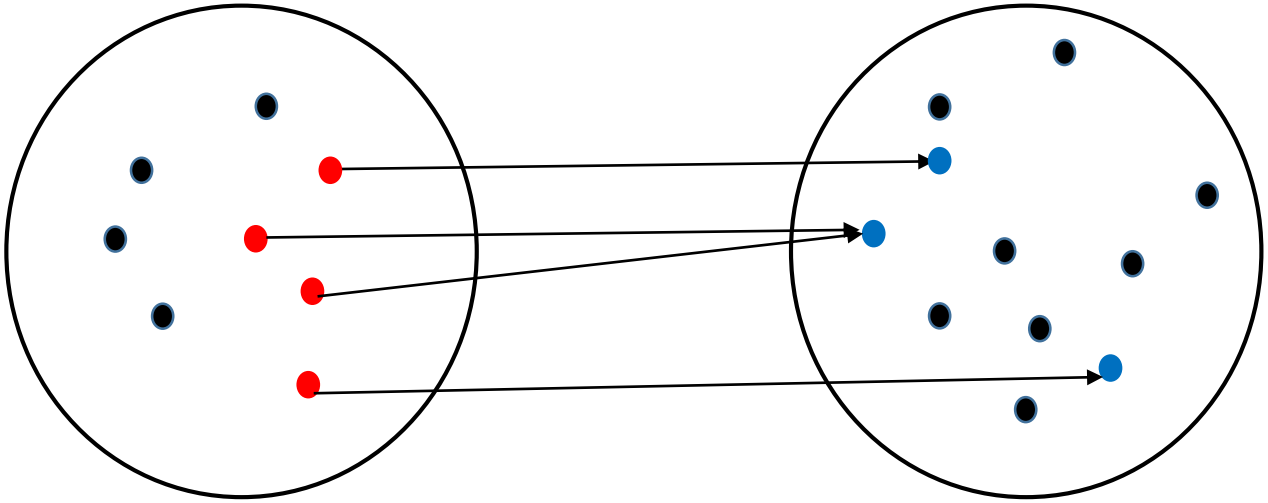
COMP 250

Lecture 27

hashing

Nov. 10, 2017

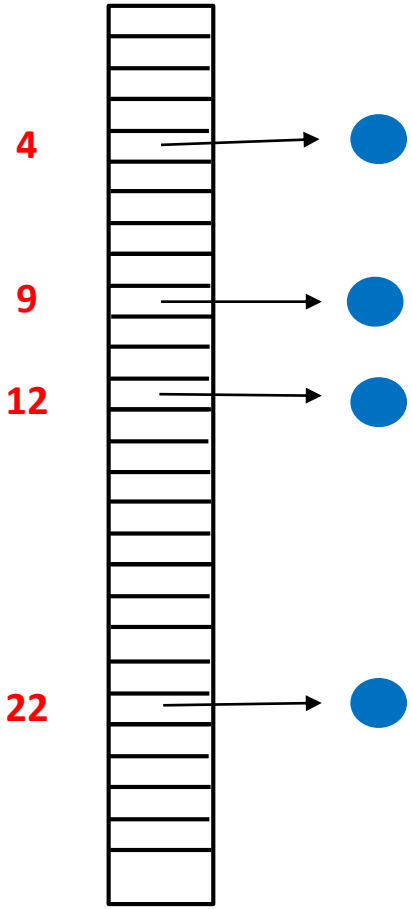
RECALL: Map



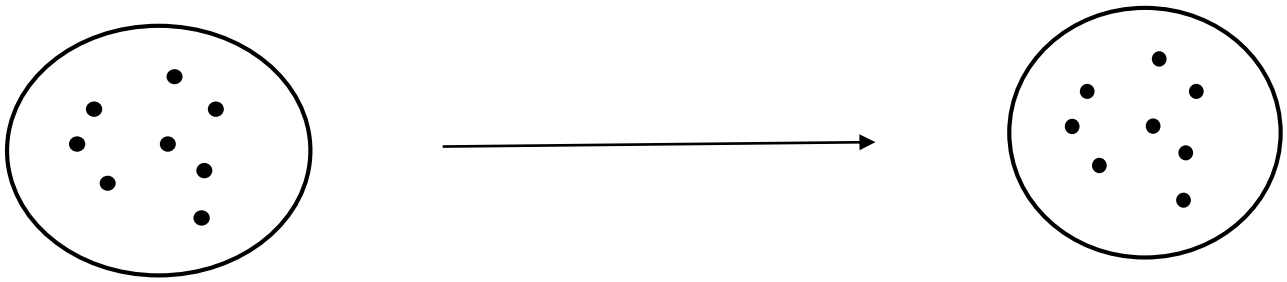
keys (type K) values (type V)

Each (key, value) pairs is an “entry”.
For each key, there is at most one value.

RECALL Special Case: keys are unique positive integers in small range



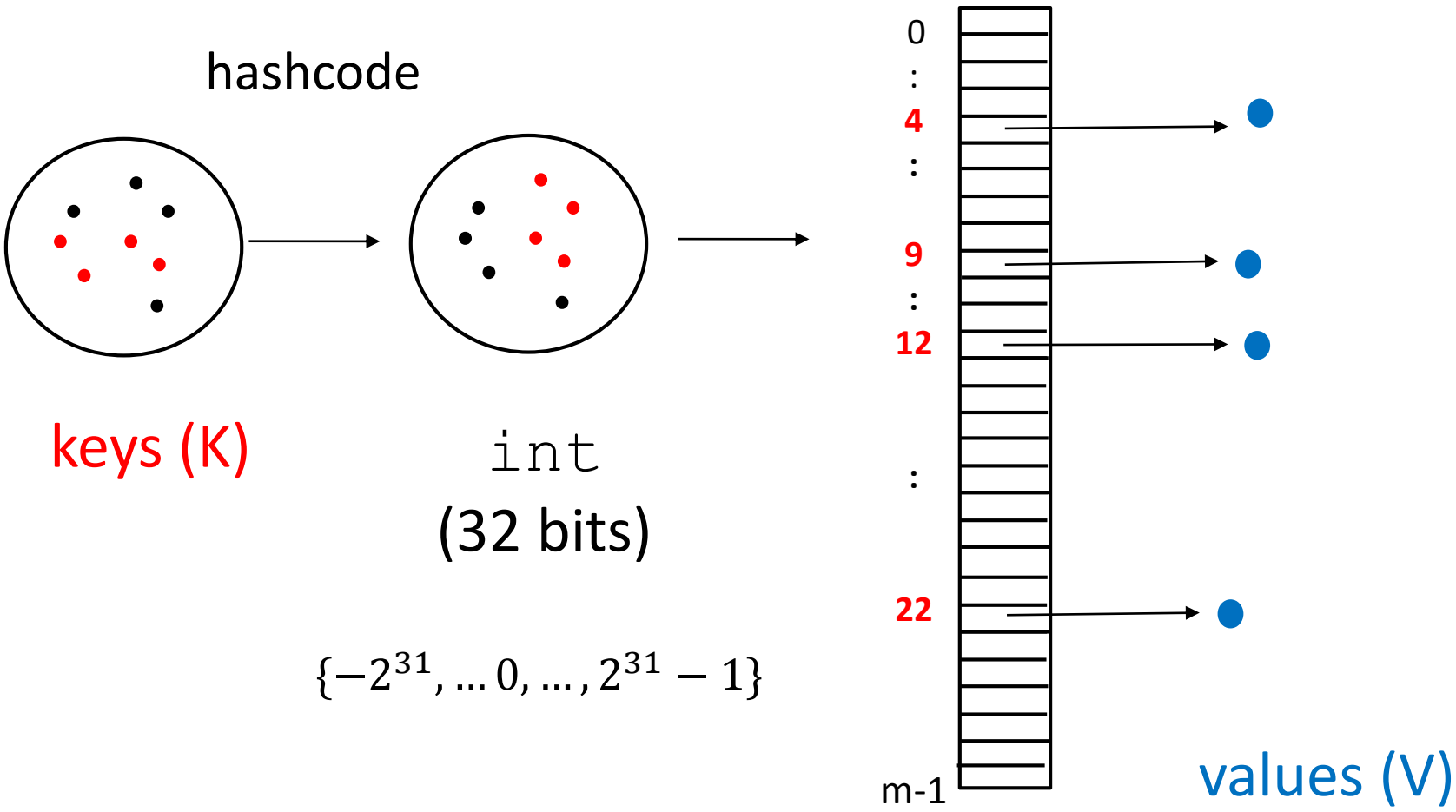
Java hashcode()

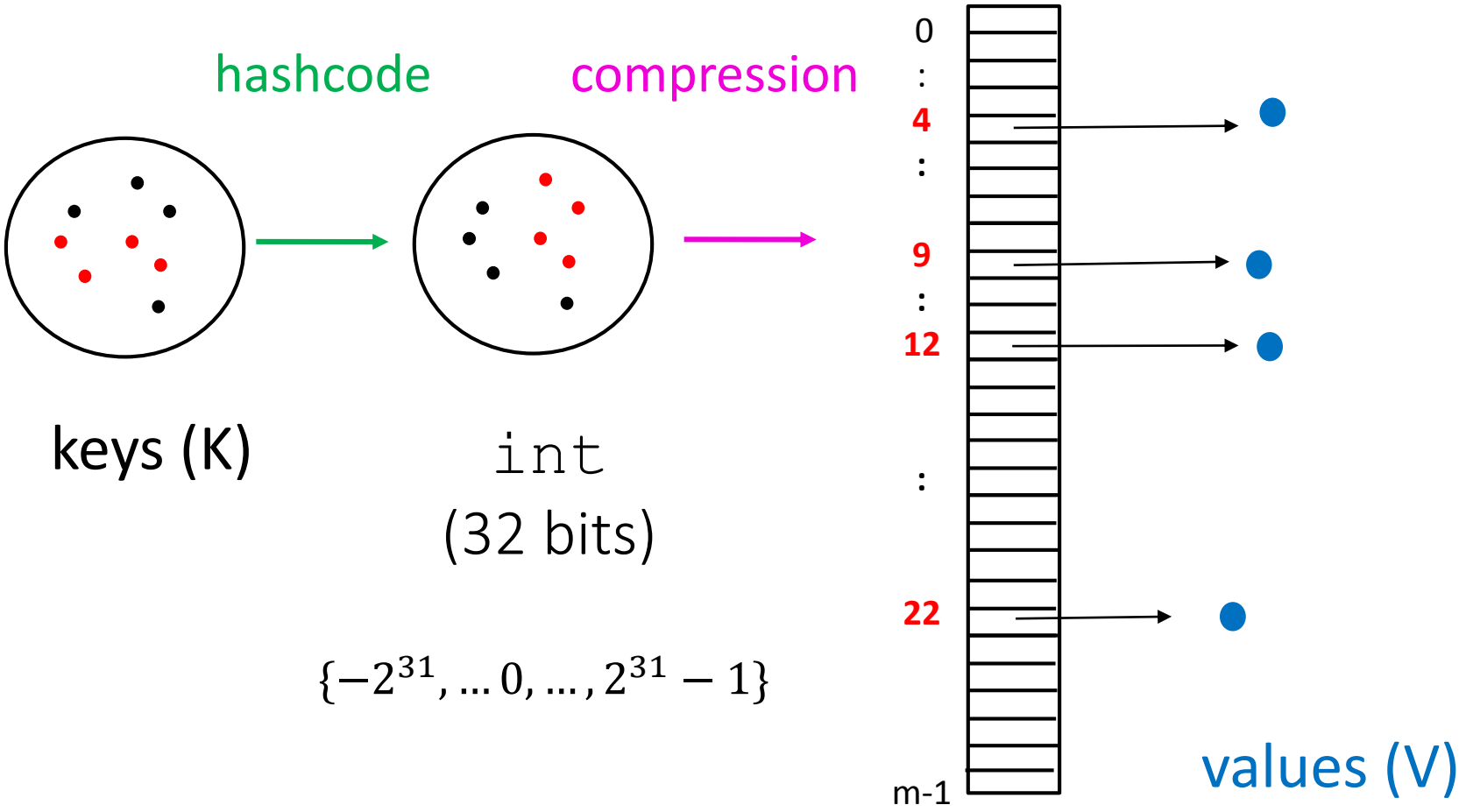


keys K

`int`
(32 bits)

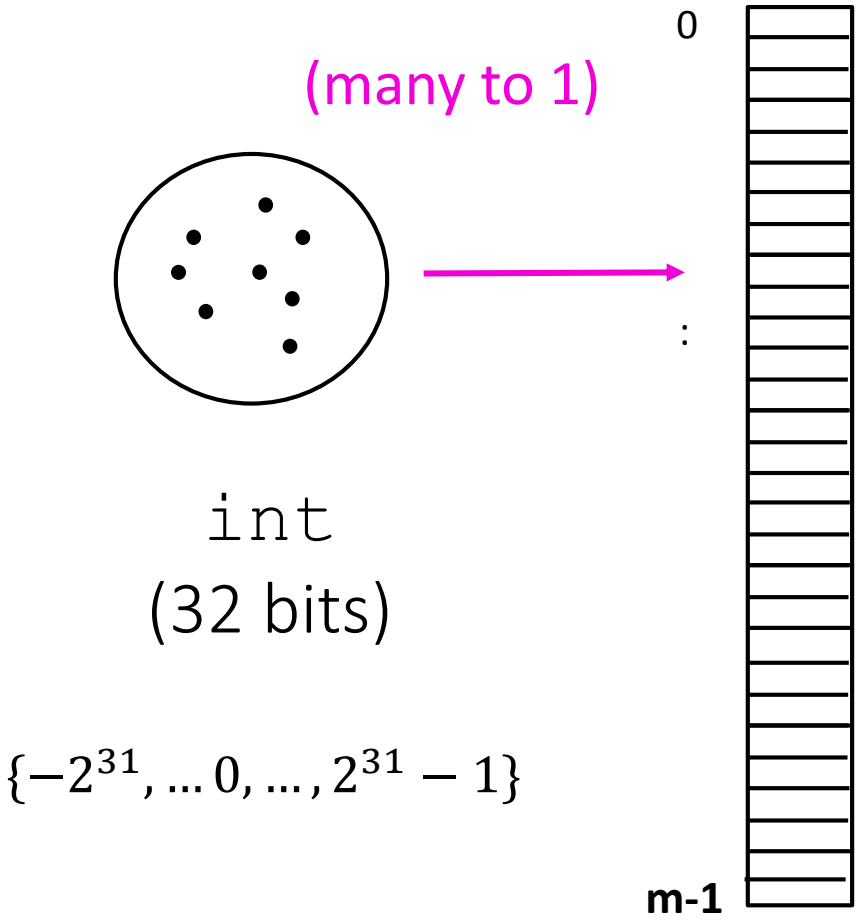
Today: Map Composition




hashcode
compression
keys (K)
int
(32 bits)
$\{-2^{31}, \ldots 0, \ldots, 2^{31} - 1\}$
0
:
4
:
9
:
12
:
22
m-1
values (V)

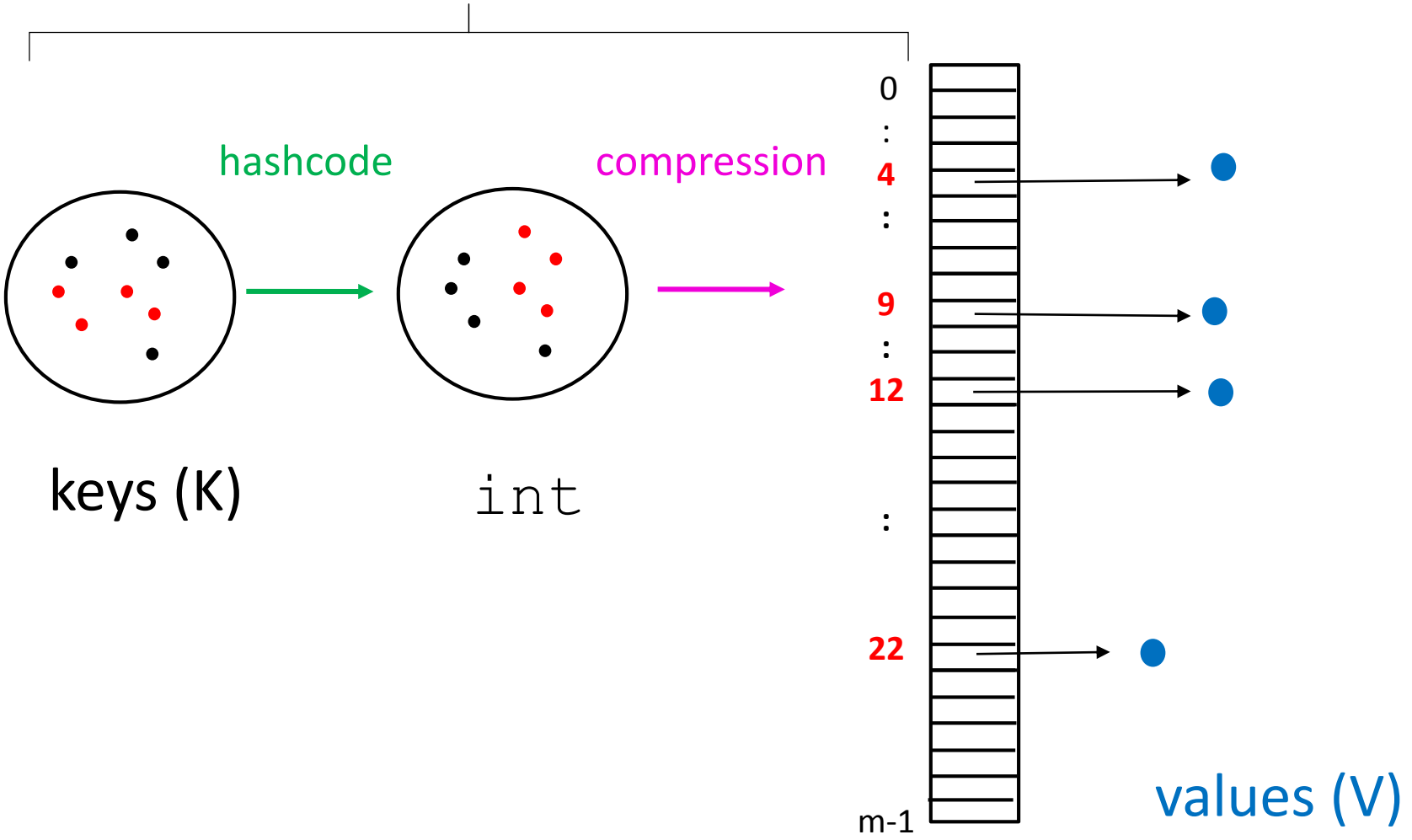
compression: $i \rightarrow |i| \ mod \ m,$

where m is the length of the array.



"hash values"

hash function : keys → {0, …, m-1}



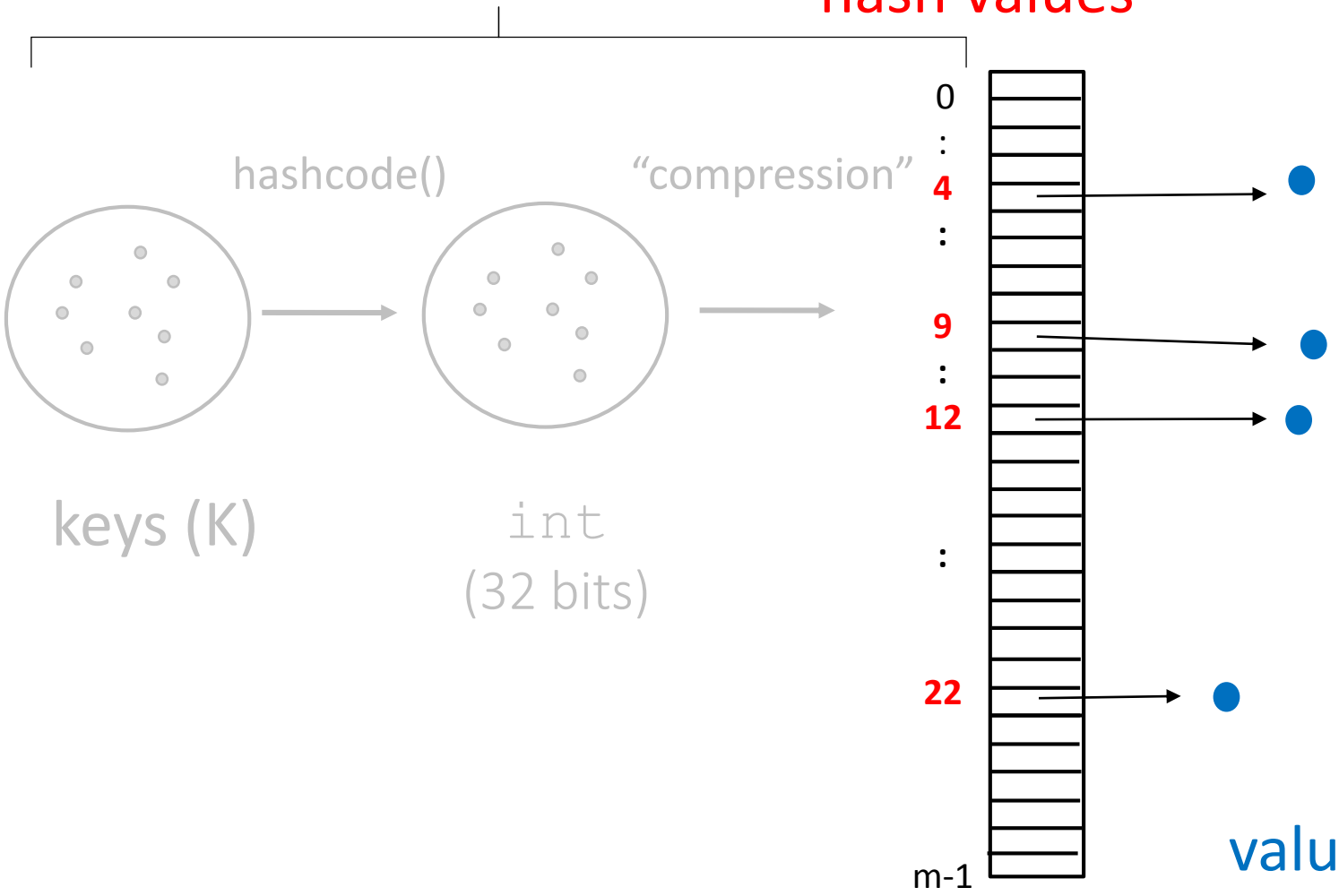
“hash function” ≡ compression ° hashCode

hash code	hash value (hash code % 7)
41	6
16	2
25	4
21	0
36	1
35	0
53	4

Heads up! "Values" is used in two ways.

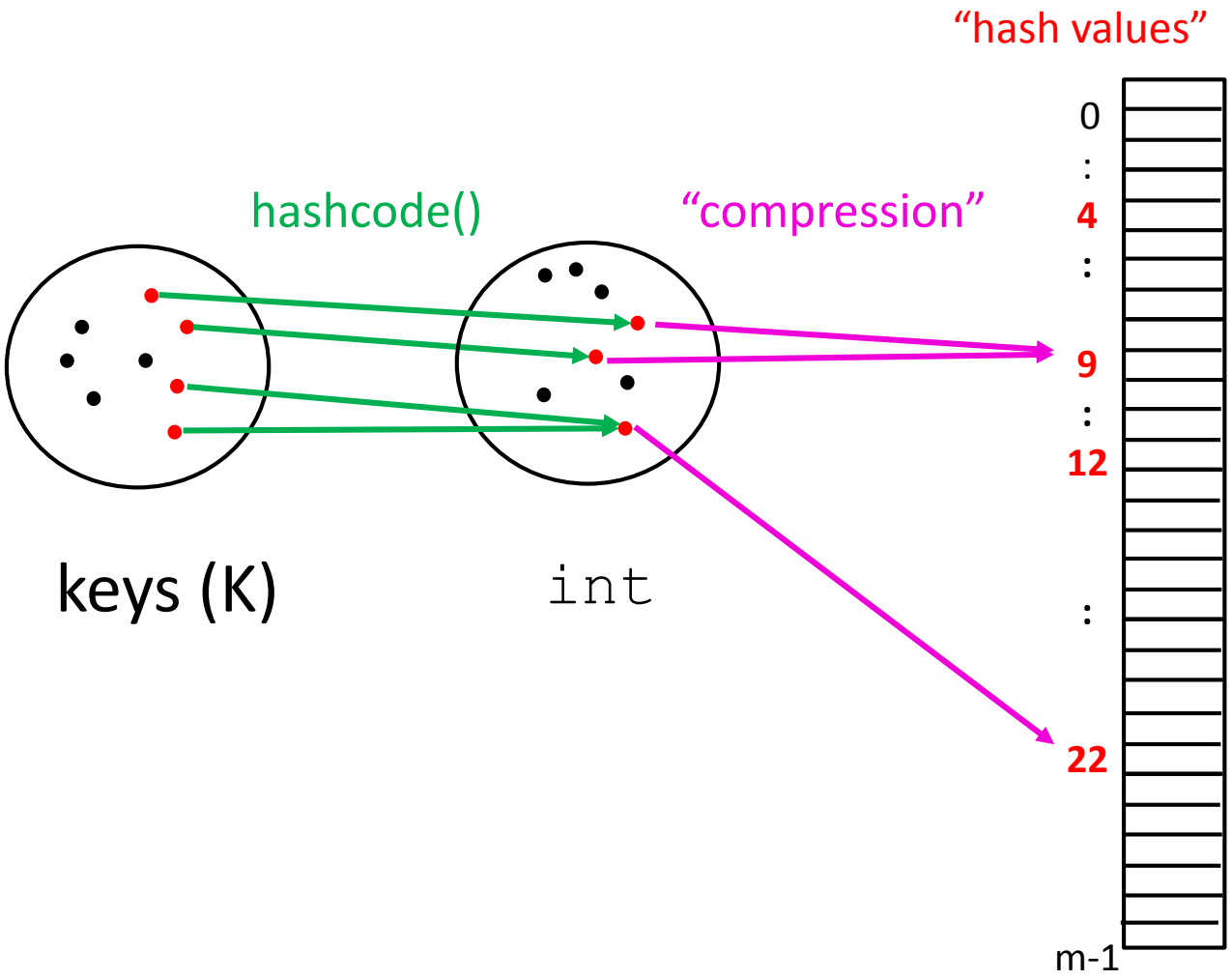
hash function : keys → {0, …, m-1}

"hash values"



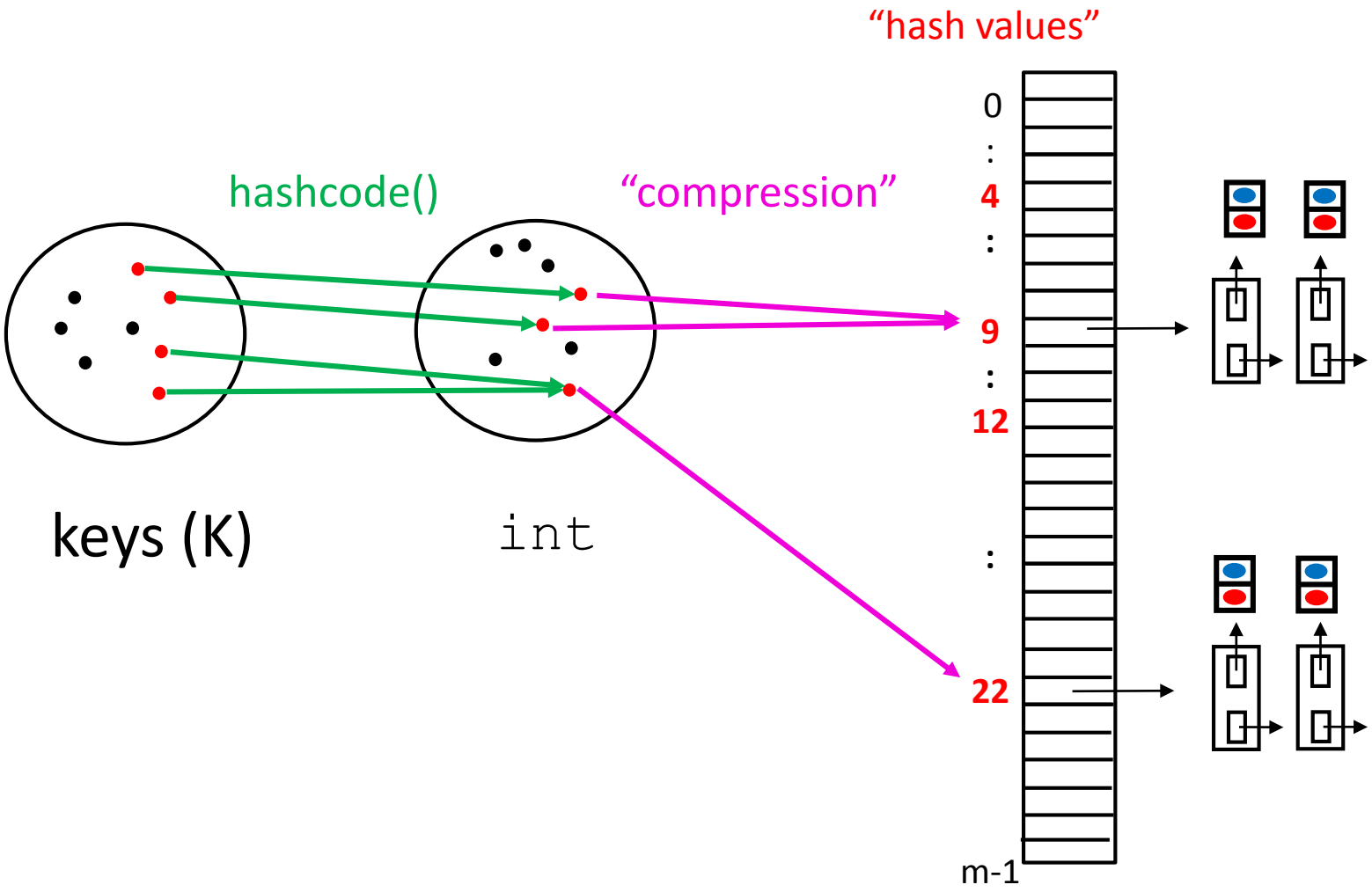
Collision:

when two or more keys k map to the same hash value.

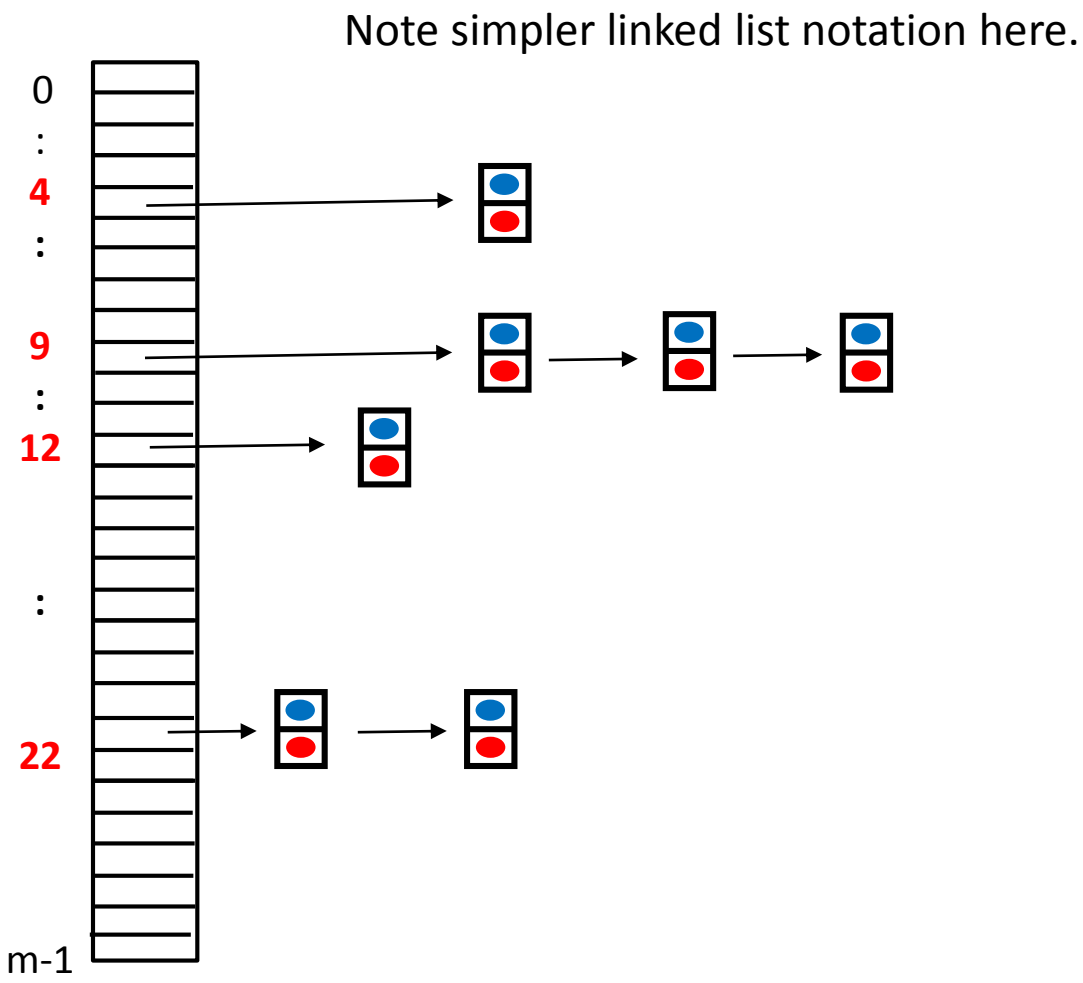


Solution: Hash Table (or Hash Map):

each array slot holds a singly linked list of entries



Each array slot + linked list is called a **bucket**.



Why is it necessary to store (key, value) pairs in the linked list?

Why not just the values?

Load factor of hash table

$$\equiv \frac{\text{number of (key, value) pairs in map}}{\text{number of buckets, } m}$$

One typically keeps the load factor below 1.

https://en.oxforddictionaries.com/definition/hash#hash_Noun_200

hash[1]

NOUN

1 A dish of cooked meat cut into small pieces and cooked again, usually with potatoes.

+ Example sentences

1.1 *North American* A finely chopped mixture.

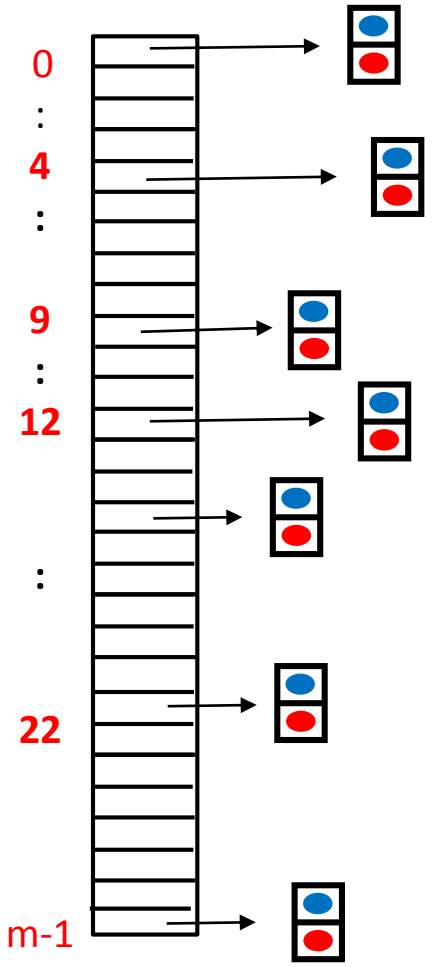
'a hash of raw tomatoes, chillies, and coriander'

+ More example sentences

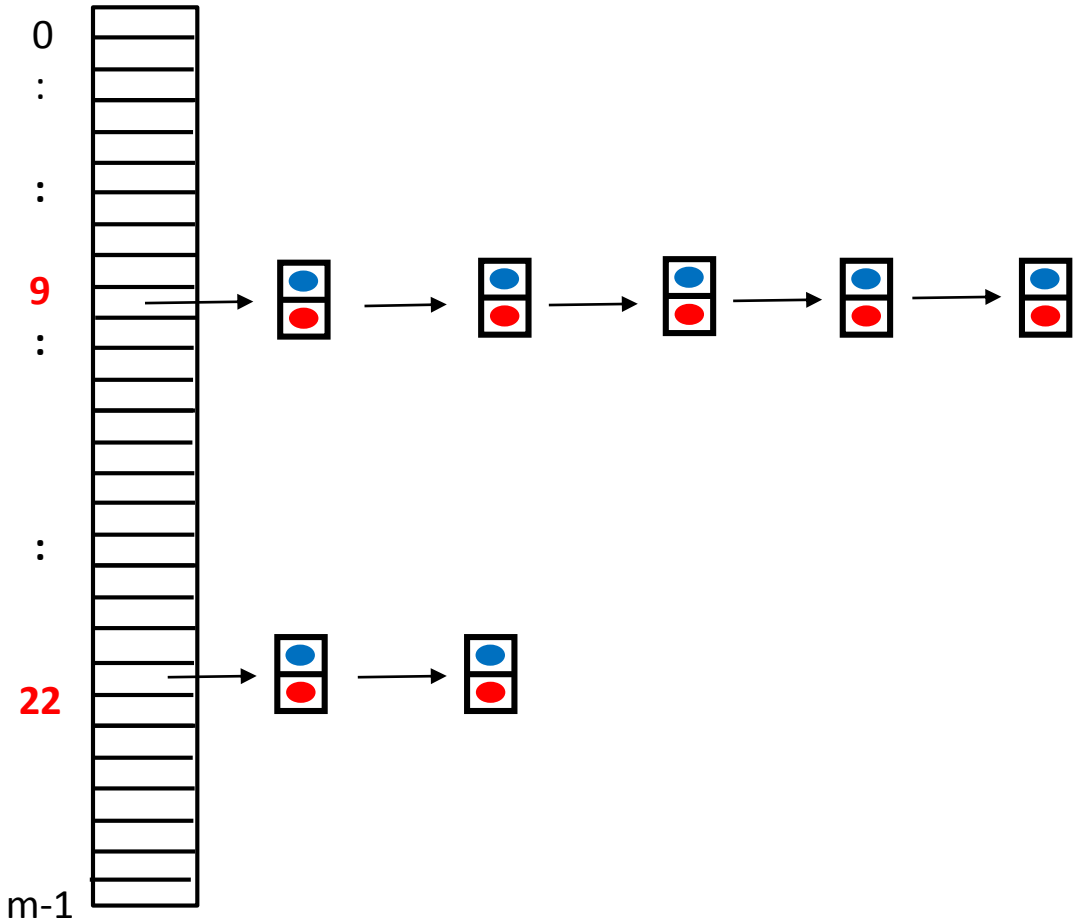
1.2 A mixture of jumbled incongruous things; a mess.

+ Example sentences + Synonyms

Good Hash



Bad Hash



Example

$h : K \rightarrow \{0, 1, \ldots, m-1\}$

Example: Suppose keys are McGill Student IDs, e.g. 260745918.

How many buckets to choose ?

Good hash function?

Bad hash function ?

Example

$h : K \rightarrow \{0, 1, \ldots, m-1\}$

Example: Suppose keys are McGill Student IDs,
e.g. 260745918.

How many buckets to choose ? (~number of entries)

Good hash function? (rightmost 5 digits)

Bad hash function ? (leftmost 5 digits)

Performance of Hash Maps

- put(key, value)
- get(key)
- remove(key)

If load factor is less than 1 and if hash function is good, then operations are O(1) “in practice”.

Note we can use a different hash function if performance is poor.

Performance of Hash Maps

- put(key, value)
- get(key)
- remove(key)
- contains(value)

Performance of Hash Maps

- put(key, value)
- get(key)
- remove(key)
- contains(value)

It will need to look at each bucket and check the list for that value. So we don't want too big an array.

Java HashMap <K, V> class

- In constructor, you can specify initial number of buckets, and maximum load factor

- How is hash function specified ?

Java HashMap <K, V> class

- In constructor, you can specify initial number of buckets, and maximum load factor

- How is hash function specified ?

  Use key's hashCode(), take absolute value, and compress it by taking mod of the number of buckets.

Java HashSet<E> class

Similar to HashMap, but there are no values. Just use it to store a set of objects of some type.

- add(e)
- contains(e)
- remove(e)
- ……

If hash function is good, then these operations are O(1).

Cryptographic Hashing

e.g. h: key (String) → hash value (128 bits)

online tool for computing md5 hash of a string

Cryptographic Hashing

e.g. h: key (String) → hash value (128 bits)

online tool for computing md5 hash of a string

Displays 128 bit result in `hexadecimal`.

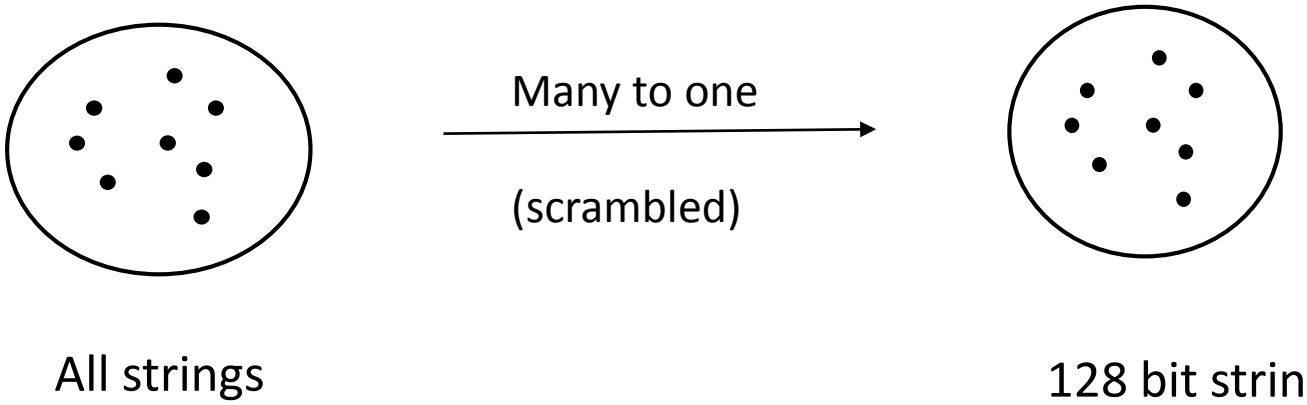
0101 0001 0011 1010 1111 1011 0010 ….

5 1 3 a f b 2

Cryptographic Hashing

We want a hash function h( ) such that if is given a hash value, then one can infer almost nothing about the key.

Small changes in the key give very different hash values.



Example Application (Sketch): Password Authentication

e.g. Web server needs to authenticate users.

Keys are usernames (String, number e.g. credit card)

Values are passwords (String)

{ (usernames, passwords) } defines a *map*.

Password Authentication (unsecure)

Suppose the {(username, password)} map is stored in a *plain text* file on the web server where user logs in.

What would the user do to log in?

What would the web server do?

What could a mischievous hacker do?

Password Authentication (unsecure)

Suppose the {(username, password)} map is stored in a *plain text* file on the web server where user logs in.

What would the user do to log in?

Enter username (key) and password (value).

What would the web server do?

Check if this entry matches what is stored in the map.

What could a mischievous hacker do?

Steal the password file, and login to user accounts.

Password Authentication (secure)

Suppose the {(username, h(password)) } map is stored in a file on the web server.

What would the user do?

What would the web server do ?

What could a mischievous hacker try to do?

Password Authentication (secure)

Suppose the {(user name, h(password)) } map is stored in a file on the web server.

What would the user do?

Enter a username and password.

What would the web server do ?

Hash the password and compare to entry in map.

What could a mischievous hacker try to do?

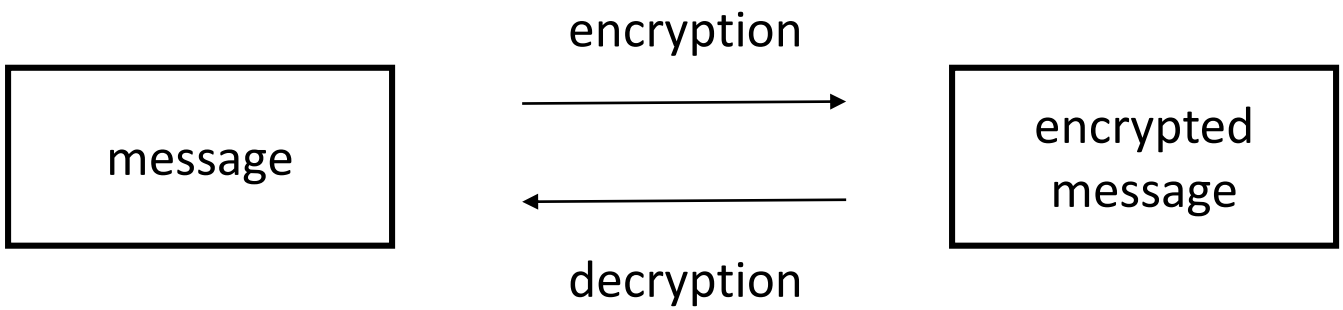
“Brute force” or “dictionary” attack.

Brute force & dictionary attacks

If hacker knows your user name, he can try logging in with *many* different passwords. (Brute force = try all, dictionary = try a chosen set e.g. “hello123”)

To reduce the probability of a hacker finding your password, user should choose long passwords with lots of special characters.

Note that hacker doesn’t need *your* password. He just needs a password such that h(your pass) = h(his pass).



You will learn about RSA encryption in MATH 240.