Course: Cryptography and Network Security
Code: CS-34310
Branch: M.C.A - 4th Semester

Lecture 11 Cryptographic Hash Function

Faculty & Coordinator : Dr. J Sathish Kumar (JSK)

Department of Computer Science and Engineering

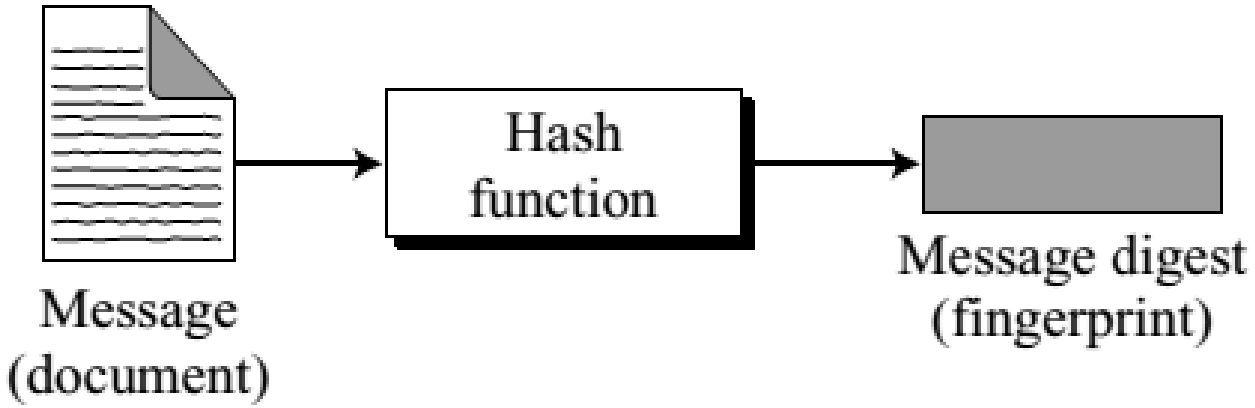
Motilal Nehru National Institute of Technology Allahabad,
Prayagraj-211004

MESSAGE INTEGRITY

- Alice may write a will to distribute her estate upon her death.
- The will does not need to be encrypted. After her death, anyone can examine the will.
- The integrity of the will, however, needs to be preserved.
- Alice does not want the contents of the will to be changed.
- Document and Fingerprint
  - If Alice needs to be sure that the contents of her document will not be changed, she can put her fingerprint at the bottom of the document.
- Eve cannot modify the contents of this document or create a false document because she cannot forge Alice's fingerprint.

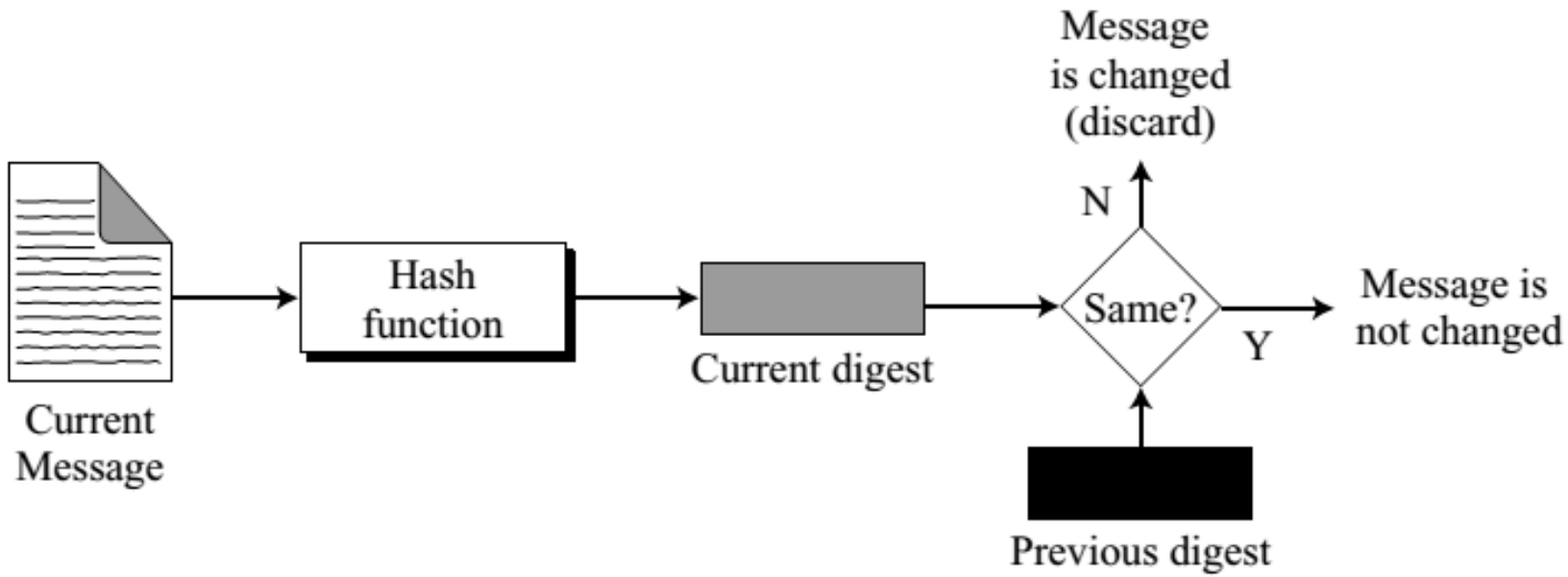
Message and Message Digest

- The electronic equivalent of the document and fingerprint pair is the message and digest pair.
- To preserve the integrity of a message, the message is passed through an algorithm called a cryptographic hash function.
- The function creates a compressed image of the message that can be used like a fingerprint.



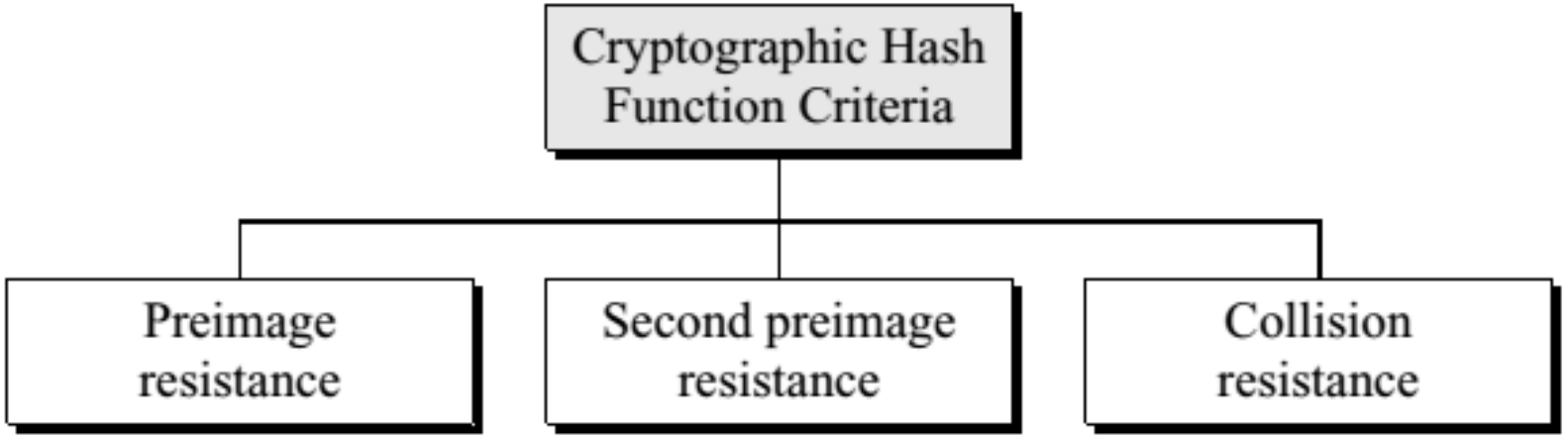
Checking Integrity

- To check the integrity of a message, or document, we run the cryptographic hash function again and compare the new message digest with the previous one.
- If both are the same, we are sure that the original message has not been changed



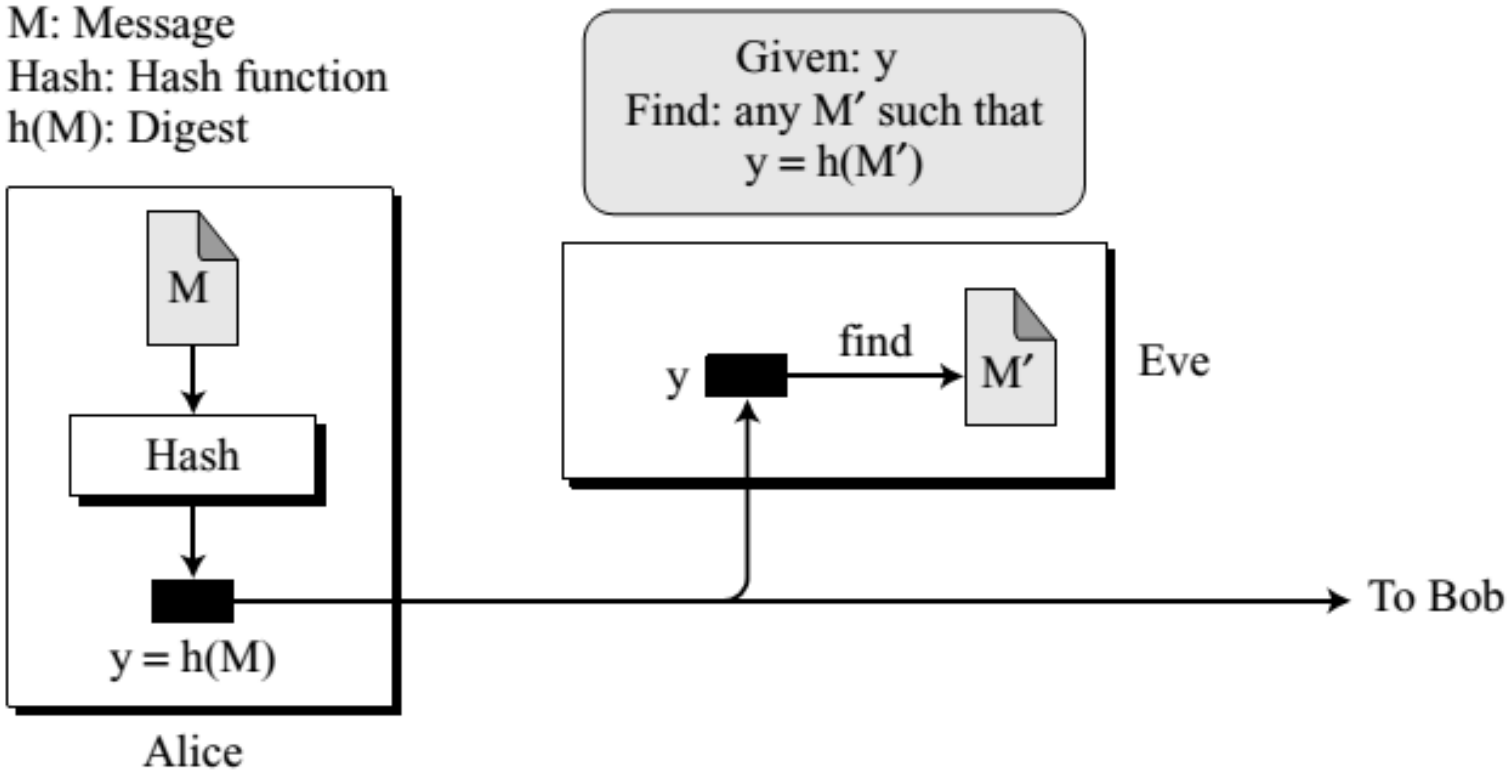
Cryptographic Hash Function Criteria

- A cryptographic hash function must satisfy three criteria:
  - preimage resistance,
  - second preimage resistance, and
  - collision resistance,



Preimage Resistance

- A cryptographic hash function must be preimage resistant.
- Given a hash function h and y = h(M), it must be extremely difficult for Eve to find any message, M′, such that y = h(M′).



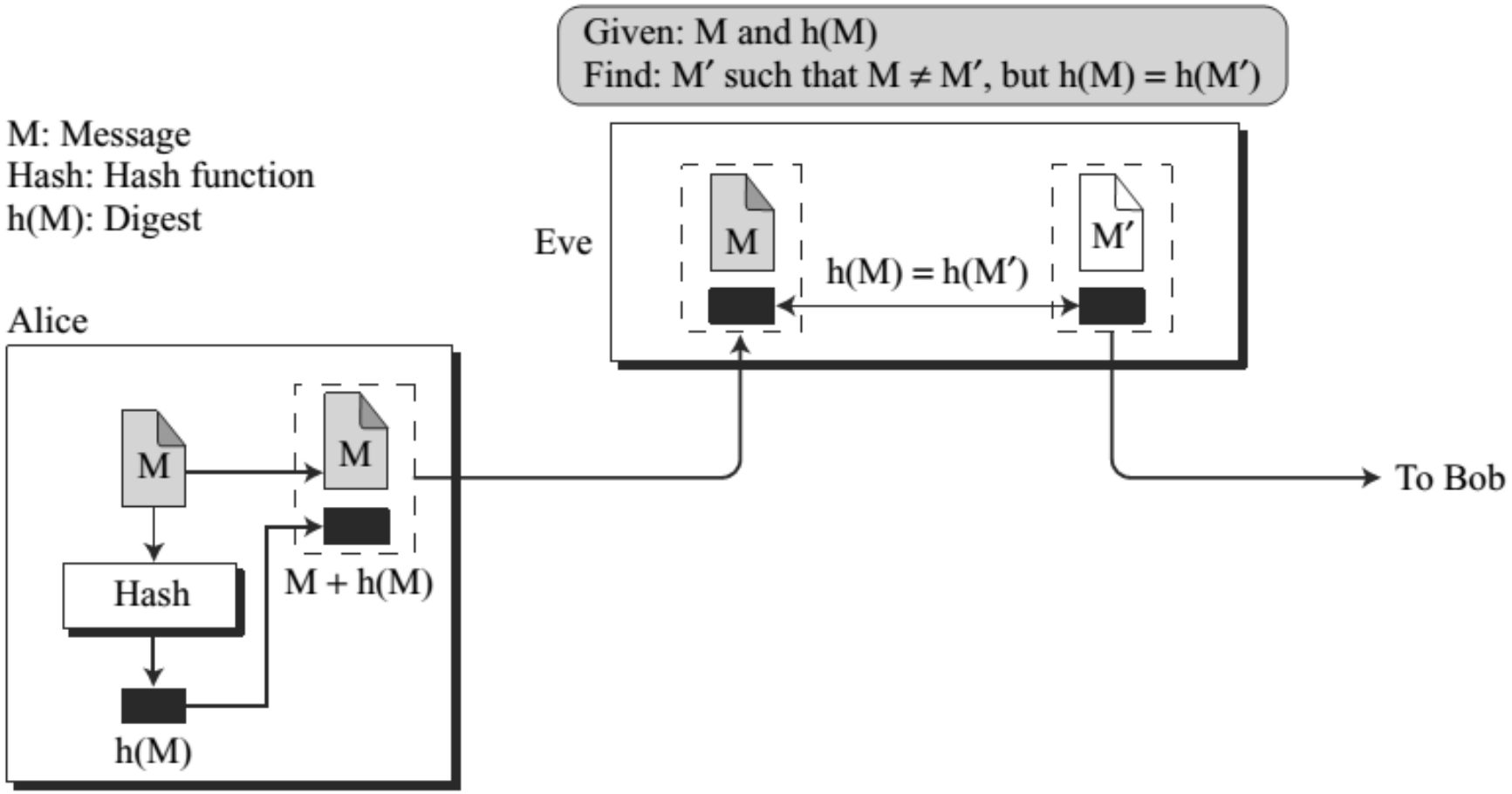
Preimage Resistance

- If the hash function is not preimage resistant, Eve can intercept the digest h(M) and create a message M’.
- Eve can then send M’ to Bob pretending it is M.
- Can we use a conventional lossless compression method such as StuffIt as a cryptographic hash function?
  - We cannot. A lossless compression method creates a compressed message that is reversible. You can uncompress the compressed message to get the original one.
- Can we use a checksum function as a cryptographic hash function?
  - We cannot. A checksum function is not preimage resistant, Eve may find several messages whose checksum matches the given one.

Second Preimage Resistance

- The second criterion, second preimage resistance, ensures that a message cannot easily be forged.
- If Alice creates a message and a digest and sends both to Bob, this criterion ensures that Eve cannot easily create another message that hashes to the exact same digest.
- In other words, given a specific message and its digest, it is impossible (or at least very difficult) to create another message with the same digest.
- Eve intercepts (has access to) a message M and its digest h(M). She creates another message M’≠ M, but h(M) = h(M’). Eve sends the M’ and h(M’) to Bob. Eve has forged the message.

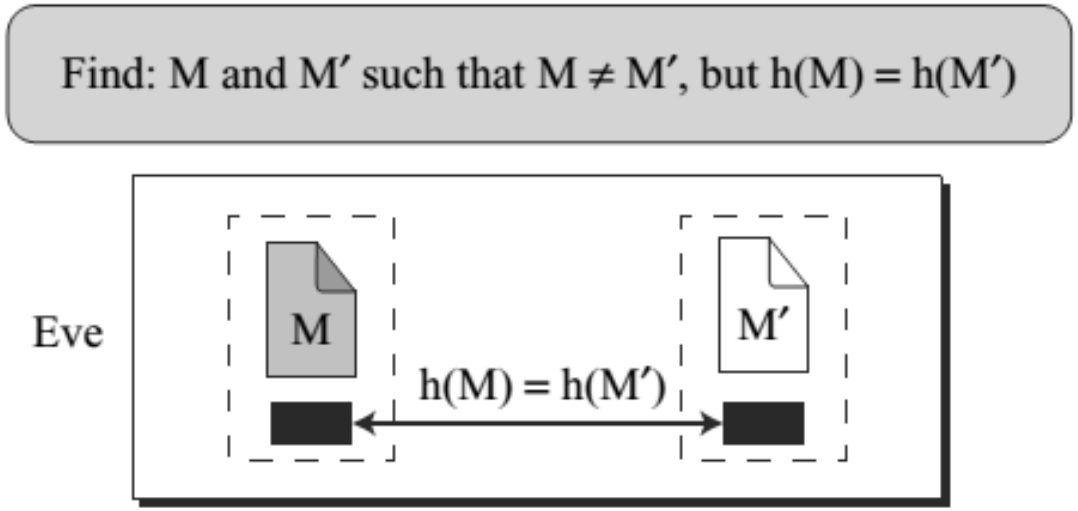
Second Preimage Resistance



Collision Resistance

- Collision resistance, ensures that Eve cannot find two messages that hash to the same digest.
- The adversary can create two messages (out of scratch) and hashed to the same digest

M: Message
Hash: Hash function
h(M): Digest

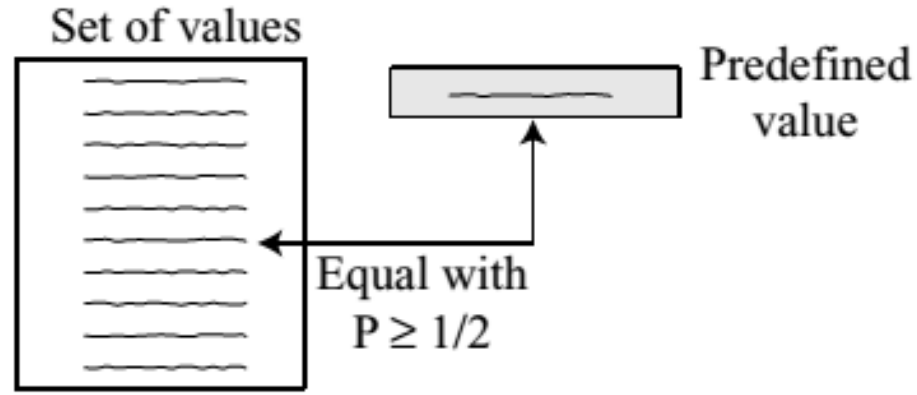


Birthday Problems

- Four different birthday problems are usually encountered in the probability courses.
- The third problem, sometimes referred to as birthday paradox, is the most common one in the literature.
- Description of Problems
  - Below the birthday problems are described in terms that can be applied to the security of hash functions.
  - Note that the term likely in all cases means with the probability $P \geq 1/2$.

Problem 1

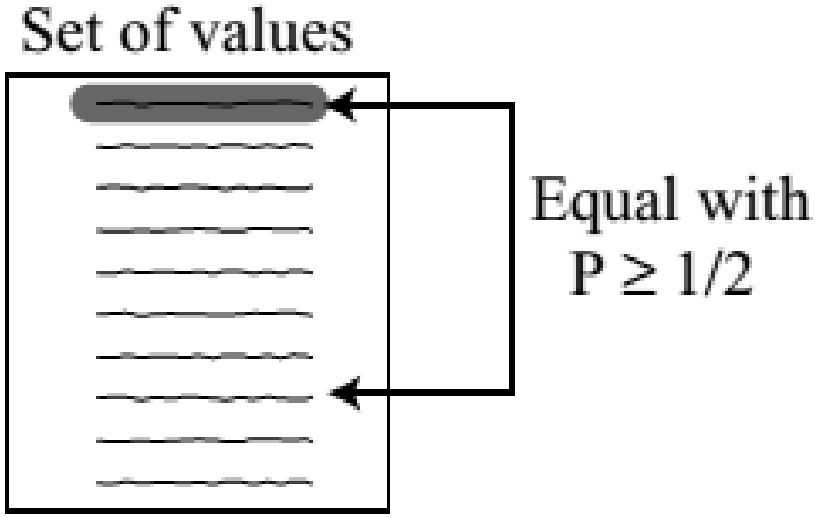
- What is the minimum number, k, of students in a classroom such that it is likely that at least one student has a predefined birthday?
- This problem can be generalized as follows.
  - We have a uniformly distributed random variable with N possible values (between 0 and N − 1).
  - What is the minimum number of instances, k, such that it is likely that at least one instance is equal to a predefined value?



a. First problem

Problem 2

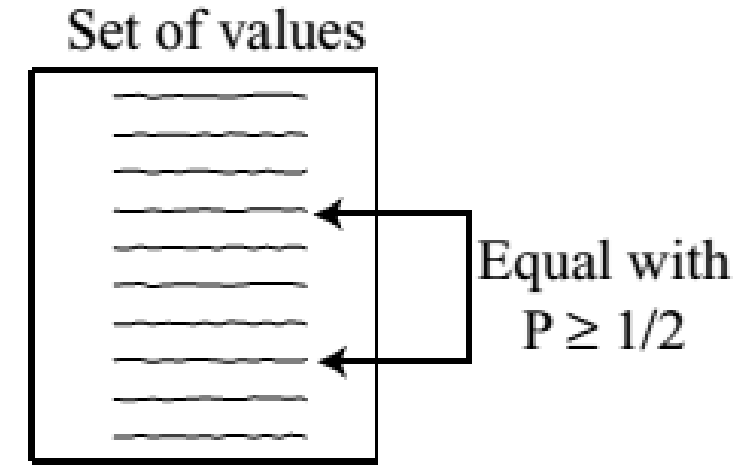
- What is the minimum number, k, of students in a classroom such that it is likely that at least one student has the same birthday as the student selected by the professor?
- This problem can be generalized as follows.
  - We have a uniformly distributed random variable with N possible values (between 0 and N − 1).
  - What is the minimum number of instances, k, such that it is likely that at least one instance is equal to the selected one?



b. Second problem

Problem 3

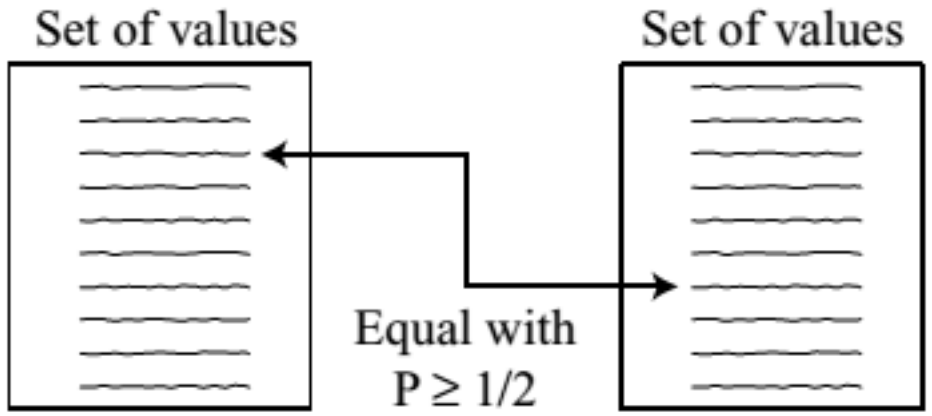
- What is the minimum number, k, of students in a classroom such that it is likely that at least two students have the same birthday?
- This problem can be generalized as follows.
  - We have a uniformly distributed random variable with N possible values (between 0 and N − 1).
  - What is the minimum number of instances, k, such that it is likely that at least two instances are equal?



c. Third problem

Problem 4

- We have two classes, each with k students. What is the minimum value of k so that it is likely that at least one student from the first classroom has the same birthday as a student from the second classroom?
- This problem can be generalized as follows.
  - We have a uniformly distributed random variable with N possible values (between 0 and N − 1).
  - We generate two sets of random values each with k instances.
  - What is the minimum number of, k, such that it is likely that at least one instance from the first set is equal to one instance in the second set?



d. Fourth problem

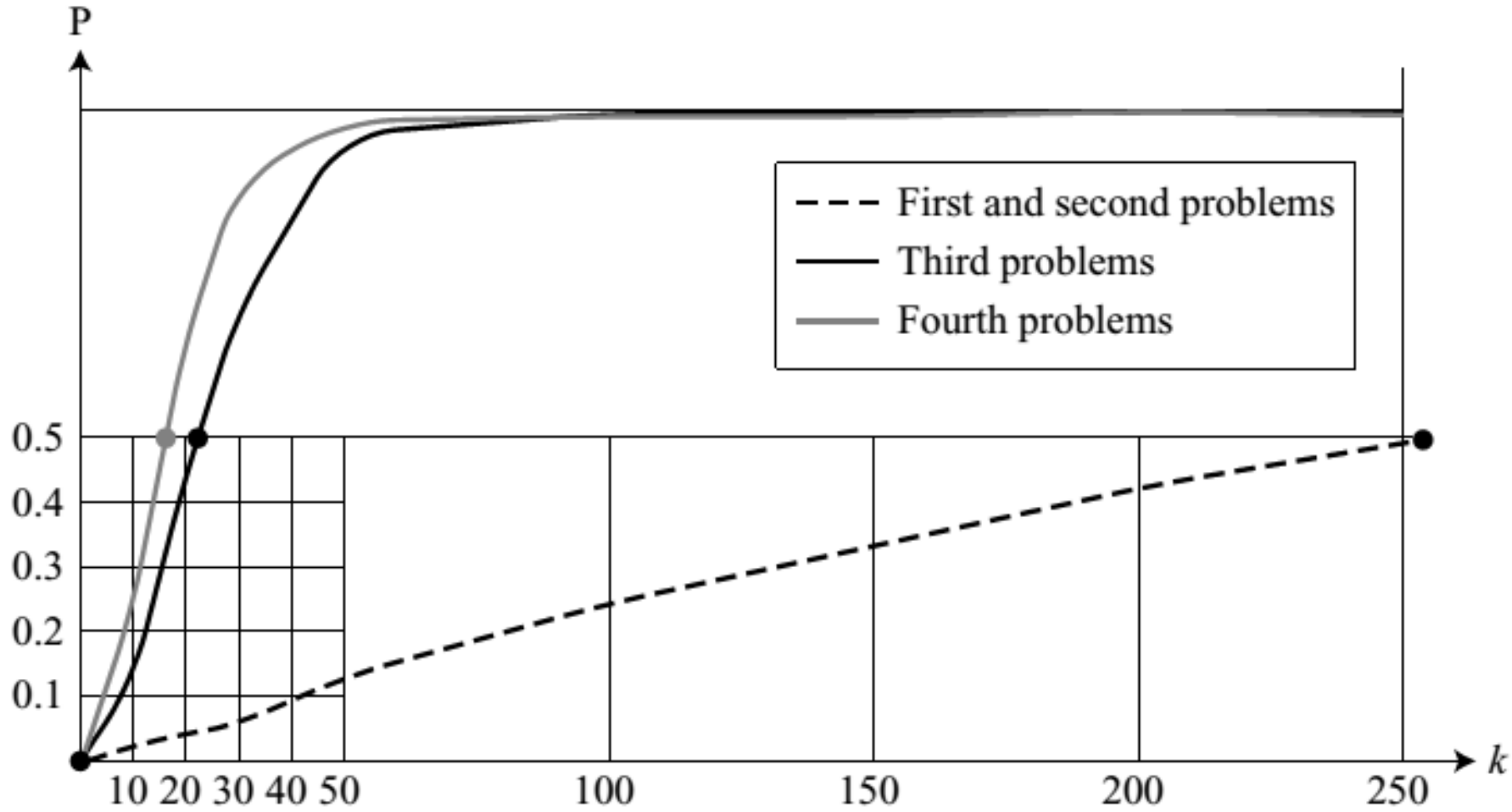
Summary of Solutions

Problem	*Probability*	*General value for k*	*Value of k with P = 1/2*	*Number of students (N = 365)*
1	$P \approx 1 - e^{-k/N}$	$k \approx \ln[1/(1-P)] \times N$	$k \approx 0.69 \times N$	253
2	$P \approx 1 - e^{-(k-1)/N}$	$k \approx \ln[1/(1-P)] \times N + 1$	$k \approx 0.69 \times N + 1$	254
3	$P \approx 1 - e^{-k(k-1)/2N}$	$k \approx \{2 \ln [1/(1-P)]\}^{1/2} \times N^{1/2}$	$k \approx 1.18 \times N^{1/2}$	23
4	$P \approx 1 - e^{-k^2/2N}$	$k \approx \{\ln [1/(1-P)]\}^{1/2} \times N^{1/2}$	$k \approx 0.83 \times N^{1/2}$	16

Comparison

- The value of k in problems 1 or 2 is proportional to N; the value of k in problems 3 or 4 is proportional to $N^{1/2}$.
- The first two problems are related to preimage and second preimage attacks; the third and the fourth problems are related to the collision attack.
- The comparison shows it is much more difficult to launch a preimage or second preimage attack than to launch a collision attack.
- For the first and second problem only one graph is shown (probabilities are very close). The graphs for the second and the third problems are more distinct.

Comparison


P
0.5
0.4
0.3
0.2
0.1
First and second problems
Third problems
Fourth problems
10 20 30 40 50
100
150
200
250
k

RANDOM ORACLE MODEL

- 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 a random string of 0s and 1s. The oracle records the message and the message digest.

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. This implies that the oracle cannot use a formula or an algorithm to calculate the digest.

Example

- Assume an oracle with a table and a fair coin. The table has two columns.
- The left column shows the messages whose digests have been issued by the oracle.
- The second column lists the digests created for those messages.
- We assume that the digest is always 16 bits regardless of the size of the message.

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

Example

- The message and the message digest are listed in hexadecimal. The oracle has already created three digests.
- Assume two events occur
- The message AB1234CD8765BDAD is given for digest calculation.
- The oracle checks its table. This message is not in the table, so the oracle flips its coin 16 times.
- Assume that result is HHTHHHTTHTHHTTTH, in which the letter H represents heads and the letter T represents tails.
- The oracle interprets H as a 1-bit and T as a 0-bit and gives 1101110010110001 in binary, or DCB1 in hexadecimal, as the message digest for this message and adds the note of the message and the digest in the table.

Example

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

Example

- The message 4523AB1352CDEF45126 is given for digest calculation.
- The oracle checks its table and finds that there is a digest for this message in the table (first row).
- The oracle simply gives the corresponding digest (13AB).
- The oracle example cannot use a formula or algorithm to create the digest for a message.
- For example, imagine the oracle uses the formula h(M) = M mod n.
- Now suppose that the oracle has already given h(M1) and h(M2).
- If a new message is presented as M3 = M1 + M2, the oracle does not have to calculate the h(M3).
- The new digest is just [h(M1) + h(M2)] mod n since
  h(M3) = (M1 + M2) mod n = M1 mod n + M2 mod n = [h(M1) + h(M2)] mod n
- This violates the third requirement that each digest must be randomly chosen based on the message given to the oracle.

Attacks on Random Oracle Model

- Suppose that the hash function creates digests of n bits.
- Then the digest can be thought of as a random variable uniformly distributed between 0 and N − 1 in which $N = 2^n$.
- In other words, there are 2^n possible values for the digest; each time the oracle randomly selects one of these values for a message.
- Note that this does not mean that the selection is exhaustive; some values may never be selected, but some may be selected several times.
- We assume that the hash function algorithm is public and Eve knows the size of the digest, n.

Preimage Attack

- Eve has intercepted a digest D = h(M); she wants to find any message M′ such that D = h(M′).
- Eve can create a list of k messages and run Algorithm

```
Preimage_Attack (D)
{
   for (i = 1 to k)
   {
        create (M [i])
        T ← h(M [i])                    // T is a temporary digest
        if (T = D) return M [i]
   }
   return failure
}
```

Preimage Attack

- The algorithm can find a message for which D is the digest or it may fail.
- What is the probability of success of this algorithm? Obviously, it depends on the size of list, k, chosen by Eve.
- To find the probability, we use the first birthday problem.
- The digest created by the program defines the outcomes of a random variable.
- The probability of success is $P \approx 1 - e^{-k/N}$.
- If Eve needs to be at least 50 percent successful, what should be the size of k?: $k \approx 0.69 \times N$, or $k \approx 0.69 \times 2^n$.
- In other words, for Eve to be successful more than 50 percent of the time, she needs to create a list of digest that is proportional to 2^n

Preimage Attack

- A cryptographic hash function uses a digest of 64 bits. How many digests does Eve need to create to find the original message with the probability more than 0.5
- The number of digests to be created is $k \approx 0.69 \times 2^n \approx 0.69 \times 2^{64}$. This is a large number. Even if Eve can create 2^{30} (almost one billion) messages per second, it takes 0.69×2^{34} seconds or more than 500 years. This means that a message digest of size 64 bits is secure with respect to preimage attack.

Second Preimage Attack

- Eve has intercepted a digest D = h(M) and the corresponding message M; she wants to find another message M′ so that h(M′) = D.
- Eve can create a list of k − 1 messages and run Algorithm

```
Second_Preimage_Attack (D, M)
{
   for (i = 1 to k – 1)
   {
        create (M [i])
        T ← h (M [i])                    // T is a temporary digest
        if (T = D) return M [i]
   }
   return failure
}
```

Second Preimage Attack

- The algorithm can find a second message for which D is also the digest or it may fail.
- What is the probability of success of this algorithm? Obviously, it depends on the size of list, k, chosen by Eve.
- To find the probability, we use the second birthday problem.
- The digest created by the program defines the outcomes of a random variable.
- The probability of success is $P \approx 1 - e^{-(k-1)/N}$.
- If Eve needs to be at least 50 percent successful, what should be the size of k? :$k \approx 0.69 \times N + 1$ or $k \approx 0.69 \times 2^n + 1$.
- In other words, for Eve to be successful more than 50 percent of the time, she needs to create a list of digest that is proportional to 2^n

Collision Attack

- Eve needs to find two messages, M and M’; such that h(M) = h(M’). Eve can create a list of k messages and run Algorithm

```
Collision_Attack
{
   for (i = 1 to k )
   {
        create (M[i])
        D[i] ← h (M[i])                    // D [i] is a list of created digests
        for (j = 1 to i – 1)
        {
           if (D[i] = D[j]) return (M[i] and M[j])
        }
   }
   return failure
}
```

Collision Attack

- The algorithm can find two messages with the same digest.
- What is the probability of success of this algorithm? Obviously, it depends on the size of list, k, chosen by Eve.
- To find the probability, we use the third birthday problem.
- The digest created by program defines the outcomes of a random variable.
- The probability of success is $P \approx 1 - e^{-k(k-1)/2N}$.
- If Eve needs to be at least fifty percent successful, what should be the size of k? : $k \approx 1.18 \times N^{1/2}$, or $k \approx 1.18 \times 2^{n/2}$.
- In other words, for Eve to be successful more than 50 percent of the time, she needs to create a list of digests that is proportional to $2^{n/2}$.

Example

- A cryptographic hash function uses a digest of 64 bits. How many digests does Eve need to create to find two messages with the same digest with the probability more than 0.5?
- The number of digests to be created is $k \approx 1.18 \times 2^{n/2} \approx 1.18 \times 2^{32}$. If Eve can test 2^{20} (almost one million) messages per second, it takes 1.18×2^{12} seconds, or less than two hours. This means that a message digest of size 64 bits is not secure against the collision attack.

Alternate Collision Attack

- The previous collision attack may not be useful for Eve.
- The adversary needs to create two messages, one real and one bogus, that hash to the same value.
- Each message should be meaningful.
- The previous algorithm does not provide this type of collision.
- The solution is to create two meaningful messages, but add redundancies or modifications to the message to change the contents of the message without changing the meaning of each.
- For example, a number of messages can be made from the first message by adding spaces, or changing the words, or adding some redundant words, and so on.
- The second message can also create a number of messages.
- Let us call the original message M and the bogus message M’.

Alternate Collision Attack

- Eve creates k different variants of M (M1, M2, …, Mk) and k different variants of M′ (M′1, M′2, … , M′k).
- Eve then uses below Algorithm to launch the attack.

Alternate_Collision_Attack (M [*k*], M′[*k*])

```
{
   for (i = 1 to k )
   {
      D[i] ← h (M[i])
      D′[i] ← h (M′[i])
      if (D [i] = D′[j]) return (M[i], M′[j])
   }
   return failure
}
```

Alternate Collision Attack

- What is the probability of success of this algorithm? Obviously, it depends on the size of the list, k, chosen by Eve.
- To find the probability, we use the fourth birthday problem.
- The two digest lists created by program defines the two outcomes of a random variable.
- The probability of success is $P \approx 1 - e^{-k^2/N}$.
- If Eve needs to be at least 50 percent successful, what should be the size of k? : $k \approx 0.83 \times N^{1/2}$ or $k \approx 0.83 \times 2^{n/2}$.
- In other words, for Eve to be successful more than 50% of the time, she needs to create a list of digests that is proportional to $2^{n/2}$.

Summary of Attacks

Attack	*Value of k with P=1/2*	*Order*
Preimage	$k \approx 0.69 \times 2^n$	2^n
Second preimage	$k \approx 0.69 \times 2^n + 1$	2^n
Collision	$k \approx 1.18 \times 2^{n/2}$	$2^{n/2}$
Alternate collision	$k \approx 0.83 \times 2^{n/2}$	$2^{n/2}$