

MH4311 Cryptography

Lecture 10

Birthday Attack

Wu Hongjun


Lecture Outline

- **Classical ciphers**
- **Symmetric key encryption**
- **Hash function and Message Authentication Code**
 - **[Birthday attack](#)**
 - **Hash function**
 - **Message Authentication Code**
- **Public key encryption**
- **Digital signature**
- **Key establishment and management**
- **Introduction to other cryptographic topics**

Recommended Reading

- CTP Section 4.2.2
- HAC Section 9.7.1, 3.2.2
- Wikipedia
 - Birthday problem
http://en.wikipedia.org/wiki/Birthday_problem
 - Floyd's cycle finding algorithm
(tortoise and hare algorithm)
http://en.wikipedia.org/wiki/Floyd%27s_cycle-finding_algorithm#Tortoise_and_hare

Birthday Problem (Birthday Paradox)

- In a set of randomly chosen people, what is the probability that at least two of them have the same birthday? (assume 365 days/year)
 - 100% if there are 366 people
 - Pigeonhole principle
 - 99% if there are 57 people
 - 50% if there are 23 people
- 
- Why?

Birthday Problem (Birthday Paradox)

- Example: 23 randomly chosen people
 - There are $\binom{23}{2} = \frac{23 \times 22}{2} = 253$ possible pairs
 - For each pair, the probability that their birthdays are the same is $1/365$ (assume 365 days)
 - With 253 pairs, the chance of getting identical birthdays is somehow high
 - Note: The probability that we get identical birthdays is **roughly** $253/365$ (note that it is not exactly $253/365$, since those 253 pairs are not independent. For example, if $A = B$, $B \neq C$, then $A \neq C$ with probability 1)

Birthday Problem

- To compute the probability p that two people (among n randomly chosen person) have the same birthday:
 - We first compute the probability that any two people do not have the same birthday (denoted as p')
 - Then $p = 1 - p'$

Birthday Problem

- To compute the probability p' that any two people do not have the same birthday
 - Randomly select two people,
 - the birthday of the second people should be different from the first one:

$$p' = (365-1)/365$$

- Randomly choose three people,
 - the birthday of the second should be different from the first
 - the birthday of the third should be different from the previous two:

$$p' = (365-1)/365 \times (365-2)/365$$

- Randomly choose n people,

$$p' = (365-1)/365 \times (365-2)/365 \times \dots \times (365-n+1)/365$$

Birthday Problem

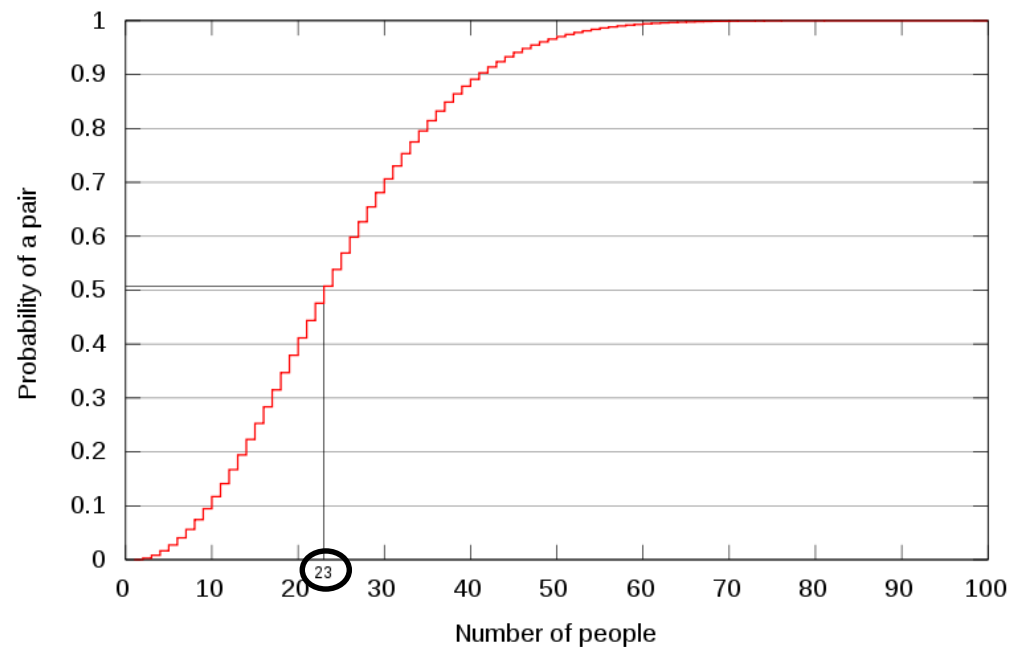
- General form for calculating $p = 1 - p'$
 - There are M possible values, we uniformly and randomly generate Q values, compute the probability that at least two values are identical.

$$p' = \left(1 - \frac{1}{M}\right) \left(1 - \frac{2}{M}\right) \cdots \left(1 - \frac{Q-1}{M}\right) = \prod_{i=1}^{Q-1} \left(1 - \frac{i}{M}\right)$$

Birthday Problem

- The values of $p = 1 - p'$

n	$p(n)$
10	11.7%
20	41.1%
<u>23</u>	<u>50.7%</u>
30	70.6%
50	97.0%
57	99.0%
100	99.99997%
200	99.9999999999999999999999999998%
300	$(100 - (6 \times 10^{-80}))\%$
350	$(100 - (3 \times 10^{-129}))\%$
366	100%*



Birthday Problem

- It is not easy to compute p for large Q and M
 - In cryptography, we will encounter $M = 2^{128}$ or 2^{256} or 2^{512}
 - Approximation is needed

- Approximation for $Q \ll M$:

- For small x , $1 - x \approx e^{-x}$
 - Reason: $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} \dots$

- Thus

$$\begin{aligned}\prod_{i=1}^{Q-1} \left(1 - \frac{i}{M}\right) &\approx \prod_{i=1}^{Q-1} e^{-\frac{i}{M}} \\ &= e^{-\sum_{i=1}^{Q-1} \frac{i}{M}} \\ &= e^{-\frac{Q(Q-1)}{2M}}.\end{aligned}$$

- $p \approx 1 - e^{-\frac{Q(Q-1)}{2M}}$

Birthday Problem

- Approximation (cont.)
 - For a given value of p , Q can be estimated as

$$Q \approx \sqrt{2M \ln \frac{1}{1-p}}$$

- For $p = 0.5$, $Q \approx 1.17\sqrt{M}$

Birthday Attack

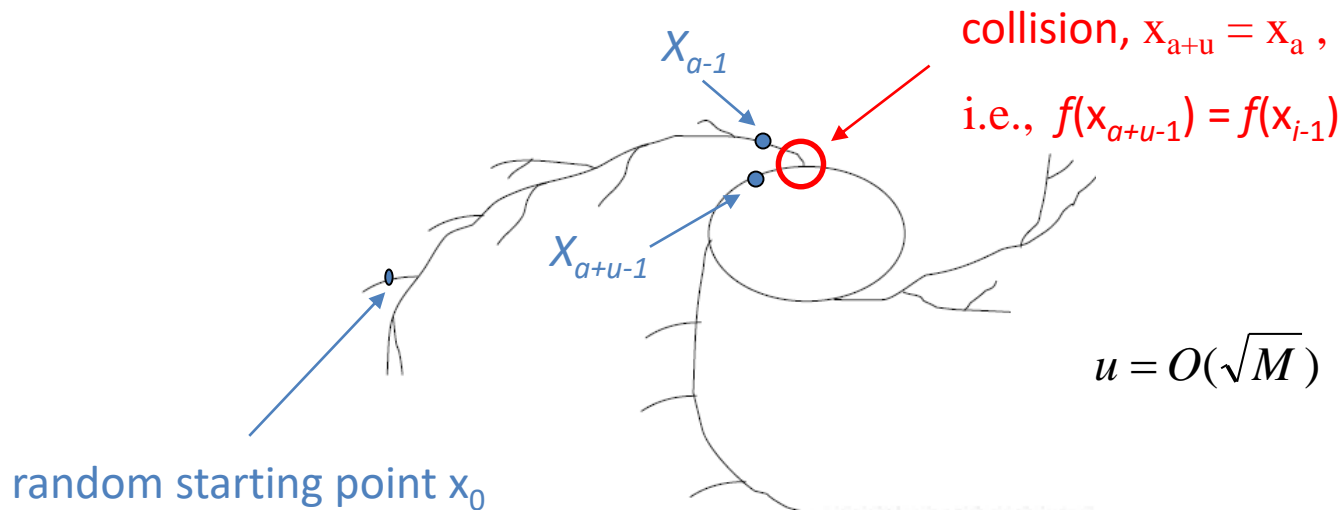
- Based on birthday problem
- For a given function f
 - function f is easy to compute,
 - but function f is difficult to invert
 - function f is non-injective
 - we try to find two different inputs s_1 and s_2 so that $f(s_1) = f(s_2)$, i.e., to find a collision of function f

Birthday Attack

- Direct method
 - Suppose that the output of function f is uniformly distributed, and the size of its output space is M
 - After generating $Q \approx 1.17\sqrt{M}$ outputs, two outputs would be equal with probability 0.5
- Complexity of the above attack
 - Computational complexity: $1.17\sqrt{M}$ computations of f
 - Memory complexity: store $1.17\sqrt{M}$ input-output pairs

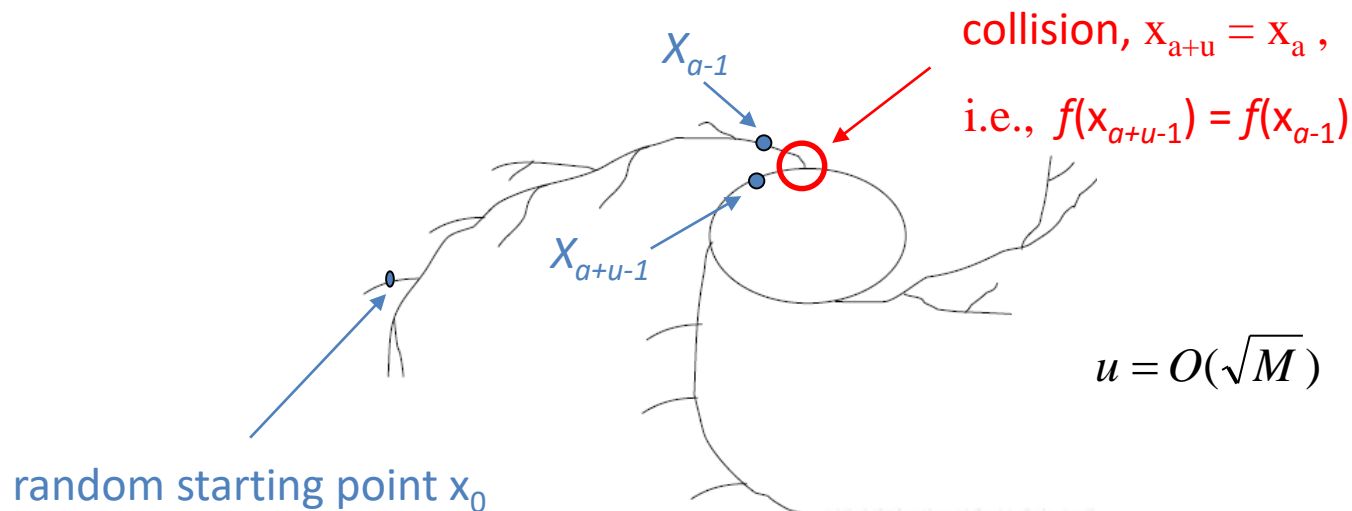
Improving Birthday Attack*

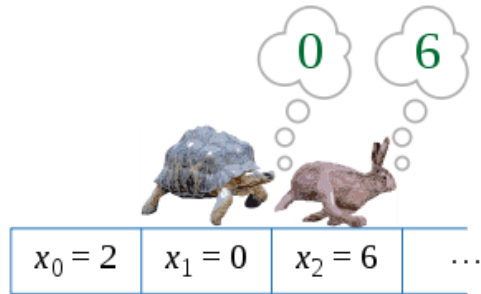
- How to reduce the memory in the attack?
 - Observe that for a function f , if we calculate
$$x_0, x_1 = f(x_0), x_2 = f(x_1), x_3 = f(x_2), x_4 = f(x_3) \dots$$
eventually we will get $x_{a+u} = x_a$
 - if f is non-injective, then:



Improving Birthday Attack*

- How to reduce the memory in the attack? (cont.)
 - How to find out a and u ?
 - Pollard's Rho method
 - Based on the “tortoise and hare” algorithm
 - » Used to detect cycle in a sequence

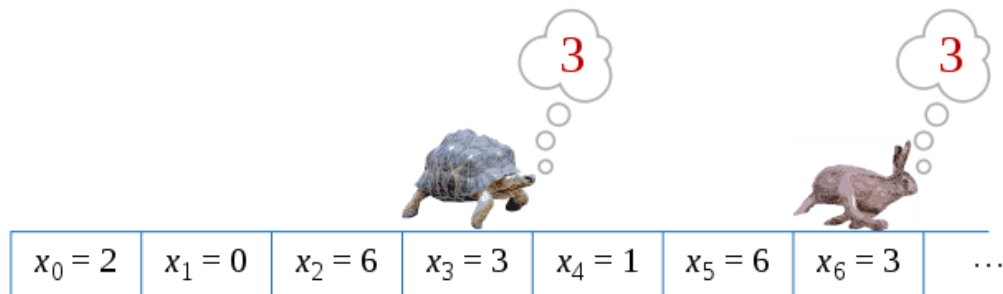
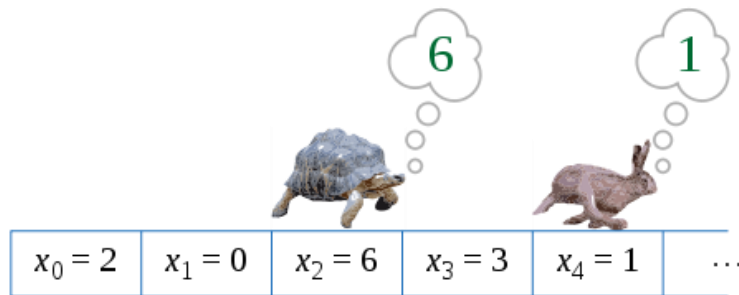




Tortoise & Hare

Sequence:

2, 0, 6, 3, 1, 6, 3, 1, 6, 3, 1 ...



[http://en.wikipedia.org/wiki/Floyd%27s_cycle-finding_algorithm#Tortoise and hare](http://en.wikipedia.org/wiki/Floyd%27s_cycle-finding_algorithm#Tortoise_and_hare)

Improving Birthday Attack*

- “tortoise and hare” algorithm
 - Invented by Floyd in the 1960s
 - For a sequence of numbers $x_0, x_1, x_2, x_3 \dots$, this algorithm is to find the cycle in the sequence (the starting point of sequence and period)
 - One pointer x_i (tortoise), each step index increases by 1
 - Another pointer x_{2i} (hare), each step index increases by 2
 - The distance between x_i and x_{2i} is i
 - the distance i keeps increasing
 - Observation:
 - If x_i enters the cycle, and i is the multiple of the period of the cycle, we will get $x_{2i} = x_i$

Improving Birthday Attack*

- “tortoise and hare” algorithm (cont.)
 - In the algorithm, we detect whether x_{2i} is the same as x_i , once we obtained $x_{2i}=x_i$, we know that $2i - i = i$ is a multiple of the period of the cycle
 - To find out the starting point of the cycle
 - Observation: When i is a multiple of the period, x_j is equal to x_{j+i} as long as x_j is on the cycle
 - Now we start with two pointers
 - one from x_0 , another from x_i
 - Each pointer increased by 1 at each step
 - Now for the first $x_j = x_{j+i}$, we know that x_j just enters the cycle, i.e., $a = j$
 - To find out the period of the cycle
 - Starting from x_j , we increase a pointer to find the first $x_{j+k} = x_j$, then the cycle period $u = k$

Improving Birthday Attack*

- Pollard's Rho method
 - reduces significantly the memory complexity of the birthday attack
 - but increases the computational complexity for a few times
- There was further improvement on birthday attack so that it
 - Requires little memory
 - Achieves parallel processing
 - P.C. van Oorschot, M.J. Wiener. Parallel collision search with cryptanalytic applications. Journal of Cryptology, vol.12 no.1 (Jan. 1999) pp.1-28.

Summary

- Birthday problem
 - The probability that at least two elements among Q random elements out of n elements are the same
- Birthday attack
 - Find a collision of a function f
 - Function f is non-injective
 - Methods:
 - Direct birthday attack
 - computational & memory complexity $1.17\sqrt{M}$
 - Rho method*
 - Reduce the memory complexity