Pollard's ρ Method

Cryptanalysis

2016.11

Contents

Section 5.4

COLLISION ALGORITHMS AND MEET-IN-THE-MIDDLE ATTACKS

The Birthday Paradox

- A simple, yet surprisingly powerful, search method is based on the observation that it is usually much easier to find matching objects than it is to find a particular object
- Methods of this sort go by many names, including meet-in-the-middle attacks and collision algorithms

The Birthday Paradox

- The fundamental idea behind collision algorithms is strikingly illustrated by the famous birthday paradox
- In a random group of 40 people, consider the following two questions:
  1. What is the probability that someone has the same birthday as you?
  2. What is the probability that at least two people share the same birthday?

The Birthday Paradox

- The answer for (1) is obtained by computing the probability that none of the people share your birthday and then subtracting that value from 1

$$\begin{aligned}
\Pr\begin{pmatrix}\text{someone has}\\ \text{your birthday}\end{pmatrix} &= 1 - \Pr\begin{pmatrix}\text{none of the 40 people}\\ \text{has your birthday}\end{pmatrix}\\
&= 1 - \prod_{i=1}^{40} \Pr\begin{pmatrix}i\text{th person does not}\\ \text{have your birthday}\end{pmatrix}\\
&= 1 - \left(\frac{364}{365}\right)^{40}\\
&\approx 10.4\,\%.
\end{aligned}$$

The Birthday Paradox

- Now consider the second question, in which you win if any two of the people in the group have the same birthday
- Again it is easier to compute the probability that all 40 people have different birthdays
- We now require that the i-th person have a birthday that is different from all of the previous $i-1$ people's birthdays

The Birthday Paradox

$$\begin{aligned}
\Pr\begin{pmatrix}\text{two people have}\\ \text{the same birthday}\end{pmatrix} &= 1 - \Pr\begin{pmatrix}\text{all 40 people have}\\ \text{different birthdays}\end{pmatrix}\\
&= 1 - \prod_{i=1}^{40} \Pr\begin{pmatrix}i\text{th person does not have}\\ \text{the same birthday as any}\\ \text{of the previous } i-1 \text{ people}\end{pmatrix}\\
&= 1 - \prod_{i=1}^{40} \frac{365-(i-1)}{365}\\
&= 1 - \frac{365}{365}\cdot\frac{364}{365}\cdot\frac{363}{365}\cdots\frac{326}{365}\\
&\approx 89.1\,\%.
\end{aligned}$$

The Birthday Paradox

- Most people tend to assume that questions (1) and (2) have essentially the same answer
- The fact that they do not is called the birthday paradox
- In fact, it requires only 23 people to have a better than 50% chance of a matched birthday, while it takes 253 people to have better than a 50% chance of finding someone who has your birthday

A Collision Theorem

- **Theorem** (Collision Theorem). An urn contains N balls, of which n are red and $N-n$ are blue
- Bob randomly selects a ball from the urn, replaces it in the urn, randomly selects a second ball, replaces it, and so on
- He does this until he has looked at a total of m balls

A Collision Theorem

a) The probability that Bob selects at least one red ball is

$$\Pr(\textit{at least one red}) = 1 - \left(1 - \frac{n}{N}\right)^m$$

b) A lower bound for the probability is

$$\Pr(\textit{at least one red}) \geq 1 - e^{-mn/N}$$

If N is large and if m and n are not too much larger than $\sqrt{N}$ (e.g. $m, n < 10\sqrt{N}$), then it is almost an equality

A Collision Theorem

- *Proof.* For (a), directly compute the probability

$$\begin{aligned}
\Pr\begin{pmatrix}\text{at least one red} \\ \text{ball in } m \text{ attempts}\end{pmatrix} &= 1 - \Pr(\text{all } m \text{ choices are blue}) \\
&= 1 - \prod_{i=1}^{m} \Pr(i\text{th choice is blue}) \\
&= 1 - \prod_{i=1}^{m} \left(\frac{N-n}{N}\right) \\
&= 1 - \left(1 - \frac{n}{N}\right)^m.
\end{aligned}$$

A Collision Theorem

- For (b), we use the inequality
$$e^{-x} \geq 1 - x$$
for all $x \in \mathbb{R}$
- Setting $x = n/N$ and raising both sides of the inequality to the m-th power shows that
$$1 - \left(1 - \frac{n}{N}\right)^m \geq 1 - (e^{-n/N})^m$$
$$= 1 - e^{-mn/N}$$

A Collision Theorem

- Proof of the inequality and the last statement of (b) is left in Exercise 5.38
- In order to connect Theorem with the problem of finding a match in two lists of numbers, we view the list of numbers as an urn containing N numbered blue balls
- After making our first list of n different numbered balls, we repaint those n balls with red paint and return them to the box

A Collision Theorem

- The second list is constructed by drawing m balls out of the urn one at a time, noting their number and color, and then replacing them
- The probability of selecting at least one red ball is the same as the probability of a matched number on the two lists

A Collision Theorem

- Example. A deck of cards is shuffled and eight cards are dealt face up
- Bob then takes a second deck of cards and chooses eight cards at random, replacing each chosen card before making the next choice
- What is Bob's probability of matching one of the cards from the first deck?

$$\Pr(\text{a match}) = 1 - \left(1 - \frac{8}{52}\right)^8 \approx 73.7\,\%$$

A Collision Theorem

- Example. A box contains 10 billion labeled objects
- Bob randomly selects 100,000 distinct objects from the box, makes a list of which objects he's chosen, and returns them to the box
- If he next randomly selects another 100,000 objects (with replacement) and makes a second list, what is the probability that the two lists contain a match?

A Collision Theorem

- The formula in the theorem says that

$$\Pr(\text{a match}) = 1 - \left(1 - \frac{100{,}000}{10^{10}}\right)^{100{,}000} \approx 0.632122$$

- The approximate lower bound given by the formula in the theorem is 0.632121, which is quite accurate

A Collision Theorem

- It is interesting to observe that if Bob doubles the number of objects in his lists to 200,000, then his probability of getting a match increases quite substantially to 98.2%
- And if he triples the number of elements in each list to 300,000, then the probability of a match is 99.988 %
- This rapid increase reflects that fact that the exponential function decreases very rapidly as soon as mn becomes larger than N

A Collision Theorem

- Example. A set contains N objects. Bob randomly chooses n of them, makes a list of his choices, replaces them, and then chooses another n of them
- How large should he choose n to give himself a 50% chance of getting a match?
- How about if he wants a 99.99% chance of getting a match?

A Collision Theorem

- Bob uses the reasonably accurate lower bound

$$\Pr(\text{match}) \approx 1 - e^{-n^2/N} = \frac{1}{2}$$

- It is easy to solve this for n

$$n = \sqrt{N \cdot \ln 2} \approx 0.83\sqrt{N}$$

A Collision Theorem

- The second question is similar, but now Bob solves

$$\Pr(\text{match}) \approx 1 - e^{-n^2/N} = 0.9999 = 1 - 10^{-4}$$

- The solution is

$$n = \sqrt{N \cdot \ln 10^4} \approx 3.035 \cdot \sqrt{N}$$

A Discrete Logarithm Collision Algorithm

- There are many applications of collision algorithms to cryptography
- These may involve searching a space of keys or plaintexts or ciphertexts, or for public key cryptosystems, they may be aimed at solving the underlying hard mathematical problem

A Discrete Logarithm Collision Algorithm

- In this section we illustrate the general theory by formulating an abstract randomized collision algorithm to solve the discrete logarithm problem (DLP)
- For the finite field $\mathbb{F}_p$, it solves the DLP in approximately $\sqrt{p}$ steps

A Discrete Logarithm Collision Algorithm

- Although the index calculus solves the DLP in $\mathbb{F}_p$ much more rapidly, there are other groups, such as elliptic curve groups, for which collision algorithms are the fastest known way to solve the DLP
- This collision algorithm may be viewed as a warm-up for Pollard's ρ algorithm, which uses only $\mathcal{O}(1)$ storage

A Discrete Logarithm Collision Algorithm

- **Proposition**. Let G be a group, and let $g \in G$ be an element of order N
- Then, assuming that the DLP

$$g^x = h$$

has a solution, a solution can be found in $\mathcal{O}(\sqrt{N})$ steps, where each step is an exponentiation in the group G

A Discrete Logarithm Collision Algorithm

- *Proof*. The idea is to write x as $x = y - z$ and look for a solution to

$$g^y = h \cdot g^z$$

- Do this by making a list of g^y values and a list of $h \cdot g^z$ values and looking for a match between the two lists

A Discrete Logarithm Collision Algorithm

- Begin by choosing random exponents $y_1, y_2, \dots, y_n$ between 1 and N and computing the values

$$g^{y_1},\ g^{y_2},\ g^{y_3},\ \dots,\ g^{y_n} \quad \text{in } G \tag{5.31}$$

- Note that all of the values are in the set

$$S = \{1, g, g^2, g^3, \dots, g^{N-1}\}$$

- View S as an urn containing N balls and the list as a way of coloring n of those balls red

A Discrete Logarithm Collision Algorithm

- Choose additional random exponents $z_1, z_2, \ldots, z_n$ between 1 and N, and compute the quantities

$$h \cdot g^{z_1}, \; h \cdot g^{z_2}, \; h \cdot g^{z_3}, \; \ldots, \; h \cdot g^{z_n} \quad \text{in } G \tag{5.32}$$

- Since we are assuming that the DLP has a solution, i.e., h is equal to some power of g, it follows that each of the values $h \cdot g^{z_i}$ is also in the set S

A Discrete Logarithm Collision Algorithm

- This may be viewed as selecting n elements from the urn, and we would like to know the probability of selecting at least one red ball
- The collision theorem says that

$$\Pr\begin{pmatrix}\text{at least one match} \\ \text{between (5.31) and (5.32)}\end{pmatrix} \approx 1 - \left(1 - \frac{n}{N}\right)^n \approx 1 - e^{-n^2/N}$$

A Discrete Logarithm Collision Algorithm

- Thus if we choose $n \approx 3\sqrt{N}$, then our probability of getting a match is greater than 99.98 %, so we are almost guaranteed a match
- If that is not good enough, take $n \approx 5\sqrt{N}$ to get a probability of success greater than $1 - 10^{-10}$
- As soon as we find a match between the two lists, say $g^y = h \cdot g^z$, then we have solved the DLP by setting $x = y - z$

A Discrete Logarithm Collision Algorithm

- Each of the lists has n elements, so it takes approximately $2n$ steps to assemble each list
- And it takes approximately $2\log_2(i)$ group multiplications to compute g^i using the double-and-add algorithm
- Thus it takes approximately $4n\log_2(N)$ multiplications to assemble the two lists

A Discrete Logarithm Collision Algorithm

- In addition, it takes about $\log_2(n)$ steps to check whether an element of the second list is in the first list (e.g., sort the first list), so $n\log_2(n)$ comparisons altogether
- Hence the total computation time is approximately

$$4n\log_2(N) + n\log_2(n) = n\log_2(N^4 n) \text{ steps.}$$

- Taking $n \approx 3\sqrt{N}$,

$$\text{Computation Time} \approx 13.5 \cdot \sqrt{N} \cdot \log_2(1.3 \cdot N)$$

A Discrete Logarithm Collision Algorithm

- Example. We solve the discrete logarithm problem $2^x = 390$ in $\mathbb{F}_{659}$
- The number 2 has order 658 modulo 659, so it is a primitive root
- In this example $g = 2$ and $h = 390$
- We choose random exponents t and compute the values of g^t and $h \cdot g^t$ until we get a match

A Discrete Logarithm Collision Algorithm

t	g^t	$h \cdot g^t$
564	410	**422**
469	357	181
276	593	620
601	416	126
9	512	3
350	445	233

t	g^t	$h \cdot g^t$
53	10	605
332	651	175
178	121	401
477	450	206
503	116	428
198	426	72

t	g^t	$h \cdot g^t$
513	164	37
71	597	203
314	554	567
581	47	537
371	334	437
83	**422**	489

- We see that $2^{83} = 422 = 390 \cdot 2^{564}$ in $\mathbb{F}_{659}$
- The solution is $390 = 2^{83-564} = 2^{-481} = 2^{177}$

A Discrete Logarithm Collision Algorithm

- Note that in the example, we have solved a DLP using two lists of length 18
- We had a 39% chance of getting a match with lists of length 18, so we were a little bit lucky
- Remark. The algorithm solves the DLP in $\mathcal{O}(\sqrt{N})$ steps

A Discrete Logarithm Collision Algorithm

- Victor Shoup has shown that there cannot exist a general algorithm to solve the DLP in an arbitrary finite group in fewer than $O(\sqrt{p})$ steps, where p is the largest prime dividing the order of the group
- This is the so-called black box DLP, in which you are given a box that instantaneously performs the group operations, but you're not allowed to see how it is doing

Section 5.5

POLLARD'S ρ METHOD

Abstract Formulation of Pollard's ρ Method

- Collision algorithms tend to require a considerable amount of storage
- A beautiful idea of Pollard often allows one to use almost no storage, at the cost of a small amount of extra computation
- Let S be a finite set and let $f: S \to S$ be a function that does a good job at mixing up the elements

Abstract Formulation of Pollard's ρ Method

- Start with some element $x \in S$ and repeatedly apply f to create a sequence of elements

$$x_0 = x, x_1 = f(x_0), x_2 = f(x_1), \ldots$$

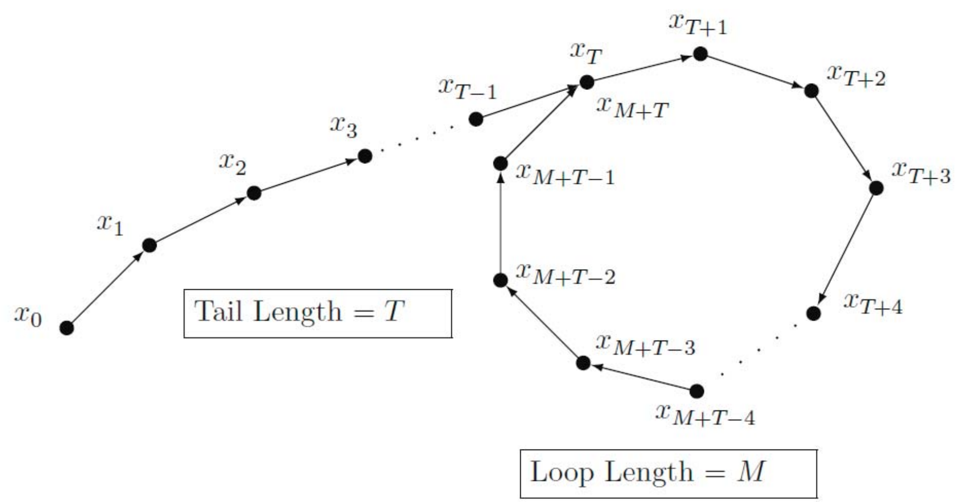
- In other words

$$x_i = \underbrace{(f \circ f \circ f \circ \cdots \circ f)}_{i \text{ iterations of } f}(x)$$

Abstract Formulation of Pollard's ρ Method

- The map f from S to itself is an example of a *discrete dynamical system*
- The sequence $x_0, x_1, x_2, \ldots$, is called the (*forward*) *orbit of* x by the map f and is denoted by
$$O_f^+(x)$$
- The set S is finite, so eventually there must be some element of S that appears twice in the orbit

Abstract Formulation of Pollard's ρ Method



Abstract Formulation of Pollard's ρ Method

- We let T be the number of elements in the "tail" before getting to the loop, and we let M be the number of elements in the loop
- Mathematically, T and M are defined by the conditions

$$T = \begin{pmatrix} \text{largest integer such that } x_{T-1} \\ \text{appears only once in } O_f^+(x) \end{pmatrix}$$

$$M = \begin{pmatrix} \text{smallest integer such} \\ \text{that } x_{T+M} = x_T \end{pmatrix}$$

Abstract Formulation of Pollard's ρ Method

- Suppose that S contains N elements
- We will later sketch a proof that the quantity $T + M$ is usually no more than a small multiple of $\sqrt{N}$
- Since $x_T = x_{T+M}$ by definition, this means that we obtain a collision in $O\left(\sqrt{N}\right)$ steps
- However, we don't know the values of T and M, it appears that we need to make a list of $x_0, x_1, \ldots$ to detect the collision

Abstract Formulation of Pollard's ρ Method

- Pollard's clever idea is that it is possible to detect a collision in $\mathcal{O}(\sqrt{N})$ steps without storing all of the values
- There are various ways to accomplish this
- We describe one such method which is easy to understand

Abstract Formulation of Pollard's ρ Method

- The idea is to compute not only the sequence x_i, but also a second sequence y_i defined by

$$y_0 = x_0$$

and for $i = 0,1,2,3, \ldots$

$$y_{i+1} = f\left(f\left(y_i\right)\right)$$

- It is clear that

$$y_i = x_{2i}$$

Abstract Formulation of Pollard's ρ Method

- In general, for $j > i$ we have

$$x_j = x_i$$

if and only if $i \geq T$ and $j \equiv i \pmod{M}$

- Thus $x_{2i} = x_i$ if and only if $i \geq T$ and

$$2i \equiv i \pmod{M}$$

which is equivalent to $M|i$

- We get $x_{2i} = x_i$ exactly when i is equal to the first multiple of M that is larger than T

Abstract Formulation of Pollard's ρ Method

- Since one of the numbers $T, T+1, \ldots, T+M-1$ is divisible by M, this proves that

$$T \leq i < T + M$$

- We show in the next theorem that the average value of $T + M$ is approximately $1.25\sqrt{N}$, so we have a very good chance of getting a collision in a small multiple of $\sqrt{N}$ steps
- Notice that we need to store only the *current values* of the x_i sequence and the y_i sequence

Abstract Formulation of Pollard's ρ Method

- **Theorem** (Pollard's ρ Method: abstract version)
- Let S be a finite set containing N elements, let $f: S \to S$ be a map, and let $x \in S$ be an initial point

a) Suppose that the forward orbit
$$O_f^+(x) = \{x_0, x_1, x_2, \ldots\}$$
of x has a tail of length T and a loop of length M. Then
$$x_{2i} = x_i \quad \text{for some } 1 \le i < T + M$$

Abstract Formulation of Pollard's ρ Method

b) If the map f is sufficiently random, then the expected value of $T + M$ is
$$E(T+M) \approx 1.2533 \cdot \sqrt{N}$$
Hence if N is large, then we are likely to find a collision in $\mathcal{O}(\sqrt{N})$ steps, where a "step" is one evaluation of the function f

Abstract Formulation of Pollard's ρ Method

- *Proof.* (a) We proved this earlier in this section
- We only sketch the proof of (b) because it is an instructive blend of probability theory and analysis of algorithms
- Suppose that we compute the first k values
$$x_0, x_1, x_2, \ldots, x_{k-1}$$
- What is the probability that we do not get any matches?

Abstract Formulation of Pollard's ρ Method

- If we assume that the successive x_i's are randomly chosen from the set S, then we can compute this probability as

$$\Pr\begin{pmatrix} x_0, x_1, \ldots, x_{k-1} \\ \text{are all different} \end{pmatrix} = \prod_{i=1}^{k-1} \Pr\left(\begin{matrix} x_i \ne x_j \text{ for} \\ \text{all } 0 \le j < i \end{matrix} \,\middle|\, \begin{matrix} x_0, x_1, \ldots, x_{i-1} \\ \text{are all different} \end{matrix} \right)$$
$$= \prod_{i=1}^{k-1} \left(\frac{N-i}{N} \right)$$
$$= \prod_{i=1}^{k-1} \left(1 - \frac{i}{N} \right).$$

Abstract Formulation of Pollard's ρ Method

- We can approximate the product using the estimate
$$1 - t \approx e^{-t}$$
for small values of t
- In practice, k will be approximately $\sqrt{N}$ and N will be large, so $\frac{i}{N}$ will indeed be small for $1 \le i < k$

$$\Pr\begin{pmatrix} x_0, x_1, \ldots, x_{k-1} \\ \text{are all different} \end{pmatrix} \approx \prod_{i=1}^{k-1} e^{-i/N} = e^{-(1+2+\cdots+(k-1))/N}$$
$$\approx e^{-k^2/2N}$$

Abstract Formulation of Pollard's ρ Method

- Assuming that $x_0, x_1, x_2, \ldots, x_{k-1}$ are distinct, what is the probability that the next choice x_k gives a match?
- There are k elements for it to match among the N possible elements, so this conditional probability is

$$\Pr\bigl(x_k \text{ is a match} \bigm| x_0, \ldots, x_{k-1} \text{ are distinct}\bigr) = \frac{k}{N}$$

Abstract Formulation of Pollard's ρ Method

- Hence

$$\begin{aligned} &\Pr\bigl(x_k \text{ is the first match}\bigr) \\ &= \Pr\bigl(x_k \text{ is a match AND } x_0, \ldots, x_{k-1} \text{ are distinct}\bigr) \\ &= \Pr\bigl(x_k \text{ is a match} \bigm| x_0, \ldots, x_{k-1} \text{ are distinct}\bigr) \\ &\qquad\qquad \cdot \Pr\bigl(x_0, \ldots, x_{k-1} \text{ are distinct}\bigr) \\ &\approx \frac{k}{N} \cdot e^{-k^2/2N} \end{aligned}$$

Abstract Formulation of Pollard's ρ Method

- The expected number of steps before finding the first match is then given by the formula

$$E(\text{first match}) = \sum_{k \ge 1} k \cdot \Pr\bigl(x_k \text{ is the first match}\bigr)$$
$$\approx \sum_{k \ge 1} \frac{k^2}{N} \cdot e^{-k^2/2N}$$

- We need the following lemma to estimate this series as a function of N

Abstract Formulation of Pollard's ρ Method

- **Lemma**. Let $F(t)$ be a "nicely behaved" real valued function with the property that
$$\int_0^\infty F(t)\,dt$$
converges
- Then for large values of n we have
$$\sum_{k=1}^{\infty} F\left(\frac{k}{n}\right) \approx n \cdot \int_0^\infty F(t)\,dt$$

Abstract Formulation of Pollard's ρ Method

- *Proof of Lemma*. We start with the definite integral of $F(t)$ over an interval $0 \le t \le A$
- By definition, this integral is equal to a limit of Riemann sums,
$$\int_0^A F(t)\,dt = \lim_{n\to\infty} \sum_{k=1}^{An} F\left(\frac{k}{n}\right) \cdot \frac{1}{n}$$
where in the sum we have broken the interval $[0, A]$ into An pieces

Abstract Formulation of Pollard's ρ Method

- In particular, if n is large, then
$$n \cdot \int_0^A F(t)\,dt \approx \sum_{k=1}^{An} F\left(\frac{k}{n}\right)$$
- Now letting $A \to \infty$ completes the proof

Abstract Formulation of Pollard's ρ Method

- *Back to the proof of Theorem*. We use Lemma to estimate
$$\begin{aligned} E(\text{first match}) &\approx \sum_{k\ge 1} \frac{k^2}{N} \cdot e^{-k^2/2N} \\ &= \sum_{k\ge 1} F\left(\frac{k}{\sqrt{N}}\right) \\ &\approx \sqrt{N} \cdot \int_0^\infty t^2 e^{-t^2/2}\,dt \\ &\approx 1.2533 \cdot \sqrt{N} \end{aligned}$$

Abstract Formulation of Pollard's ρ Method

- Remark. It is instructive to check numerically the accuracy of the estimates used in the proof of Theorem.
- In that proof we claimed that for large values of N, the expected number of steps before finding a match is given by each of the following three formulas:

$E_1 = \sum_{k\geq 1} \frac{k^2}{N} \prod_{i=1}^{k-1}\left(1-\frac{i}{N}\right)$	$E_2 = \sum_{k\geq 1} \frac{k^2}{N} e^{-k^2/2N}$	$E_3 = \sqrt{N}\int_0^\infty t^2 e^{-t^2/2}\,dt$

Abstract Formulation of Pollard's ρ Method

- More precisely, E_1 is the exact formula, but hard to compute exactly if N is very large, while E_2 and E_3 are approximations
- We have computed the values of E_1, E_2, and E_3 for some moderate sized values of N and compiled the results in the following table

Abstract Formulation of Pollard's ρ Method

N	E_1	E_2	E_3	E_1/E_3
100	12.210	12.533	12.533	0.97421
500	27.696	28.025	28.025	0.98827
1000	39.303	39.633	39.633	0.99167
5000	88.291	88.623	88.623	0.99626
10000	124.999	125.331	125.331	0.99735
20000	176.913	177.245	177.245	0.99812
50000	279.917	280.250	280.250	0.99881

Abstract Formulation of Pollard's ρ Method

- E_2 and E_3 are quite close to one another, and once N gets reasonably large, they also provide a good approximation for E_1
- Hence for very large values of N, say $2^{80} < N < 2^{160}$, it is quite reasonable to estimate E_1 using E_3

Discrete Logarithms via Pollard's ρ Method

- In this section we describe how to use Pollard's ρ method to solve the DLP
$$g^t = h$$
in $\mathbb{F}_p^*$, where g is a primitive root modulo p
- The idea is to find a collision between $g^i h^j$ and $g^k h^l$ for some known exponents i, j, k, l
- Then $g^{i-k} = h^{l-j}$, and taking roots in $\mathbb{F}_p$ to solve the DLP

Discrete Logarithms via Pollard's ρ Method

- The difficulty is finding a function $f\colon \mathbb{F}_p \to \mathbb{F}_p$ that is complicated enough to mix up the elements of $\mathbb{F}_p$, yet simple enough to keep track of its orbits
- Pollard suggests using the function

$$f(x) = \begin{cases} gx & \text{if } 0 \le x < p/3, \\ x^2 & \text{if } p/3 \le x < 2p/3, \\ hx & \text{if } 2p/3 \le x < p. \end{cases}$$

Discrete Logarithms via Pollard's ρ Method

- <u>Remark</u>. Note that x and $f(x)$ must be reduced modulo p in order to repeatedly apply the function f
- No one has proven that the function $f(x)$ is sufficiently random, but experimentally, the function f works fairly well
- However, Teske has shown that f is not sufficiently random to give optimal results, and she gives examples of more complicated functions that work better in practice

Discrete Logarithms via Pollard's ρ Method

- Starting with $x_0 = 1$
- At each step, we either multiply by g, multiply by h, or square the previous value
- After i steps we have
$$x_i = \underbrace{(f \circ f \circ f \circ \cdots \circ f)}_{i \text{ iterations of } f}(1) = g^{\alpha_i} \cdot h^{\beta_i}$$
- We cannot predict the values of α_i and β_i, but we can compute them at the same time that we are computing the x_i's

Discrete Logarithms via Pollard's ρ Method

- Clearly $\alpha_0 = \beta_0 = 0$
- Then subsequent values are given by

$$\alpha_{i+1} = \begin{cases} \alpha_i + 1 & \text{if } 0 \le x < p/3, \\ 2\alpha_i & \text{if } p/3 \le x < 2p/3, \\ \alpha_i & \text{if } 2p/3 \le x < p, \end{cases}$$

$$\beta_{i+1} = \begin{cases} \beta_i & \text{if } 0 \le x < p/3, \\ 2\beta_i & \text{if } p/3 \le x < 2p/3, \\ \beta_i + 1 & \text{if } 2p/3 \le x < p. \end{cases}$$

Discrete Logarithms via Pollard's ρ Method

- In computing α_i and β_i, it suffices to keep track of their values modulo $p-1$, since $g^{p-1} = 1$ and $h^{p-1} = 1$
- This is important, since otherwise the values of α_i and β_i would become prohibitively large
- In a similar fashion we compute the sequence given by

$$y_0 = 1 \text{ and } y_{i+1} = f(f(y_i))$$

Discrete Logarithms via Pollard's ρ Method

- Then

$$y_i = x_{2i} = g^{\gamma_i} \cdot h^{\delta_i},$$

where the exponents γ_i and δ_i can be computed by two repetitions of the recursions used for α_i and β_i

- Applying the above procedure, we eventually find a collision in the x and the y sequences, say

$$y_i = x_i$$

- This means that $g^{\alpha_i} \cdot h^{\beta_i} = g^{\gamma_i} \cdot h^{\delta_i}$

Discrete Logarithms via Pollard's ρ Method

- Let $u \equiv \alpha_i - \gamma_i$ and $v \equiv \delta_i - \beta_i \pmod{p-1}$
- Then $g^u = h^v$ in $\mathbb{F}_p$
- Equivalently,

$$v \cdot \log_g(h) \equiv u \pmod{p-1}$$

- If $\gcd(v, p-1) = 1$, then we can multiply both sides by $v^{-1} \pmod{p-1}$ to solve the DLP

Discrete Logarithms via Pollard's ρ Method

- If $d = \gcd(v, p-1) \geq 2$, we use the extended Euclidean algorithm to find an integer s such that
$$s \cdot v \equiv d \pmod{p-1}$$
- Multiplying both sides by s yields
$$d \cdot \log_g(h) \equiv w \pmod{p-1}$$
where $w = s \cdot u \pmod{p-1}$

Discrete Logarithms via Pollard's ρ Method

- The fact that $d | p-1$ will force $d | w$, so
$$\log_g(h) = \frac{w}{d}$$
is one solution, but there are others
- The full set of solutions is obtained by starting with w/d and adding multiples of $(p-1)/d$,
$$\log_g(h) \in \left\{ \frac{w}{d} + k \cdot \frac{p-1}{d} : k = 0, 1, 2, \ldots, d-1 \right\}$$

Discrete Logarithms via Pollard's ρ Method

- In practice, d will tend to be fairly small, so it suffices to check each of the d possibilities for $\log_g(h)$ until the correct value is found
- Example. We illustrate Pollard's ρ method by solving the DLP
$$19^t \equiv 24717 \pmod{48611}$$
- The first step is to compute the x and y sequences until we find a match $y_i = x_i$, while also computing the exponent sequences $\alpha, \beta, \gamma, \delta$

Discrete Logarithms via Pollard's ρ Method

i	x_i	$y_i = x_{2i}$	α_i	β_i	γ_i	δ_i
0	1	1	0	0	0	0
1	19	361	1	0	2	0
2	361	33099	2	0	4	0
3	6859	13523	3	0	4	2
4	33099	20703	4	0	6	2
5	33464	14974	4	1	13	4
6	13523	18931	4	2	14	5
7	13882	30726	5	2	56	20
8	20703	1000	6	2	113	40
9	11022	14714	12	4	228	80

$\vdots$

Discrete Logarithms via Pollard's ρ Method

i	x_i	$y_i = x_{2i}$	α_i	β_i	γ_i	δ_i
			$\vdots$			
542	21034	46993	13669	2519	27258	30257
543	20445	37138	27338	5038	27259	30258
544	40647	33210	6066	10076	5908	11908
545	28362	21034	6066	10077	5909	11909
546	36827	40647	12132	20154	23636	47636
547	11984	36827	12132	20155	47272	46664
548	33252	33252	12133	20155	47273	46665

Discrete Logarithms via Pollard's ρ Method

- From the table we see that $x_{1096} = x_{548} = 33252$ in $\mathbb{F}_{48611}$
- The associated exponent values are

$$\alpha_{548} = 12133, \qquad \beta_{548} = 20155,$$
$$\gamma_{548} = 47273, \qquad \delta_{548} = 46665,$$

- So we know that

$$19^{12133} \cdot 24717^{20155} = 19^{47273} \cdot 24717^{46665}$$

in $\mathbb{F}_{48611}$

Discrete Logarithms via Pollard's ρ Method

- Moving the powers gives

$$19^{13470} = 24717^{26510} \quad \text{in } \mathbb{F}_{48611}.$$

- We next observe that

$$\gcd(26510, 48610) = 10$$

and

$$970 \cdot 26510 \equiv 10 \pmod{48610}$$

Discrete Logarithms via Pollard's ρ Method

- Raising both sides to the 970-th power yields

$$\begin{aligned} 19^{13470 \cdot 970} &= 19^{13065900} \\ &= 19^{38420} \\ &= 24717^{10} \end{aligned}$$

in $\mathbb{F}_{48611}$

- Hence

$$10 \cdot \log_{19}(24717) \equiv 38420 \pmod{48610},$$

which means that

$$\log_{19}(24717) \equiv 3842 \pmod{4861}.$$

Discrete Logarithms via Pollard's ρ Method

- The possible values for the discrete logarithm are obtained by adding multiples of 4861 to 3842
- So $\log_{19}(24717)$ is one of the numbers in the set

$$\{3842, 8703, 13564, 18425, 23286,$$
$$28147, 33008, 37869, 42730, 47591\}$$

Discrete Logarithms via Pollard's ρ Method

- To complete the solution, we compute 19 raised to each of these 10 values until we find the one that is equal to 24717:

$$19^{3842} = 16580, \quad 19^{8703} = 29850,$$
$$19^{13564} = 23894, \quad 19^{18425} = 20794,$$
$$19^{23286} = 10170, \quad 19^{28147} = 32031,$$
$$19^{33008} = 18761, \quad 19^{37869} = \boxed{24717}.$$