INTERMODULAR DESCRIPTION SHEET: UMAP Unit 808

TITLE: Cards, Codes, and Kangaroos

AUTHOR: Lindsey R. Bosko
Dept. of Mathematics
North Carolina State University
Raleigh, NC 27606
lrbosko@ncsu.edu

MATHEMATICAL FIELD: Numerical analysis

APPLICATION FIELD: Games

TARGET AUDIENCE: Undergraduate or early graduate student with one semester each of abstract algebra and linear algebra

ABSTRACT: The Kruskal Count is a card trick invented by mathematician (not magician) Martin Kruskal. The mathematics of the trick illustrate Pollard's kangaroo method, which was designed to solve the discrete log problem. Given a finite cyclic group, $G = \langle g \rangle$, and $x \in G$ the discrete log problem involves finding $n \in \mathbb{Z}$ such that $g^n = x$. In this Module, I demonstrate the card trick and in revealing its secret, we uncover connections to the discrete log problem, cryptography, and Markov chains.

PREREQUISITES: Cyclic groups, generators, modular arithmetic, matrix inverses, modular arithmetic and factoring functions for a computer algebra system (e.g., for Maple, the functions **mod** and **ifactor**, if-then statements and for-loops in programming in such a system, expected value, and standard results about Markov chains (transient and absorbing states, canonical form of the transition matrix, and the fundamental matrix).

The UMAP Journal 32 (3): 199–236.

COMAP, Inc., Suite 3B, 175 Middlesex Tpke., Bedford, MA 01730
(800) 77-COMAP = (800) 772-6627, or (781) 862-7878; http://www.comap.com

Cards, Codes, and Kangaroos

Lindsey R. Bosko

Dept. of Mathematics
North Carolina State University
Raleigh, NC 27606
lrbosko@ncsu.edu

Table of Contents

1. Introduction

This Module begins with a card trick explained in **Section 2**, followed by an example of the trick succeeding. Before describing the secret behind the trick's success, we describes several cryptosystems that rely on what is known as the discrete logarithm problem (DLP) (**Section 3**). The link between these topics is Pollard's kangaroo method, which is an attack on the DLP as well as the basis for the trick. In **Section 4**, we reveal the secret behind the trick, together with many of the links among cryptography, discrete logarithms, cards, and kangaroos. Before further analysis on the card trick, we set out results about Markov chains **Section 5**. We then model the card trick as a Markov process **Section 6**. The results of the Markov chain analysis confirm that the card trick will "probably" work. "Probably" was the precise word used by Martin Gardner [1978], who first published the trick; his intent was to convey its difference from a typical magician's trick. We also perform another probabilistic analysis, on a modified version of the card trick, in **Section 7**, which provides an upper bound on the probability of the trick failing.

2. A Card Trick

A group of mathematicians gathers at a party. To entertain her peers, one brings a standard deck of 52 cards (i.e., a poker/bridge deck) and throughout the evening performs the following trick:

> As dealer, she invites a person to be the player and instructs the player to choose secretly a number between 1 and 10. She informs all watching that each card has a value: Aces 1, face cards 5, and all other cards the number on the card. She deals the cards, one at a time face up. When the number of cards dealt equals the player's secretly chosen number, the player is to note (silently) the value of the corresponding card, which we refer to as the player's first *key card*. With the first key card's value in mind, the player silently counts until the number of cards dealt from his key card on equals its value; the corresponding card is the player's second key card.
>
> For instance, suppose that the player initially chose 4 as the secret number and the first four cards dealt are $A, 10, 2, 8$; then the player's first key card is the 8. The player next counts to the eighth card dealt after the 8 to arrive at the player's second key card.
>
> Play continues in this manner, from one key card to the next, until all 52 cards have been dealt. The dealer then announces what she believes to be the player's last secret card. To the player's astonishment, the dealer is correct (well, maybe—as we will see).

An example of a deck, with the player's key cards in red/gray) appears below.

Example 1.

$$A, 10, 2, 8, 7, 4, 7, J, 6, 5, K, 8, 2, 3, 9, 8, 5, Q, J, 5, 6, 8, K, 10, K, 3,$$

$$9, 5, 2, K, 4, Q, 9, Q, 3, 7, 6, A, J, 10, J, A, 6, 4, 9, 4, 7, A, 2, 3, \mathbf{Q}, 10$$

The initial secret number was 4 and the last key card is a Queen. (The trick does not distinguish suits or colors.)

The dealer does not always correctly guess the player's last card, but most of the time she is successful. As she performs the trick with other players, her fellow mathematicians begin to inquire how the trick works. They note that the deck is random, neither shuffled nor stacked in any particular order. The dealer, not being a true magician, is happy to discuss the secret, but first she must give some background on a seemingly unrelated topic—cryptography.

For Websites where you can play the card trick, see Haga and Robins [1997] or Montenegro [2009].

Exercises

1. **a)** Perform the card trick from **Example 1** as if you were the player, using a different initial secret number. What is the last card that you land on (your last key card)?
   b) Is there an initial secret number between 1 and 10 that will lead to something other than the Queen being the last key card? If so, what number(s) will do so?
2. What are the possible last key cards if the card trick is played on the ordered deck

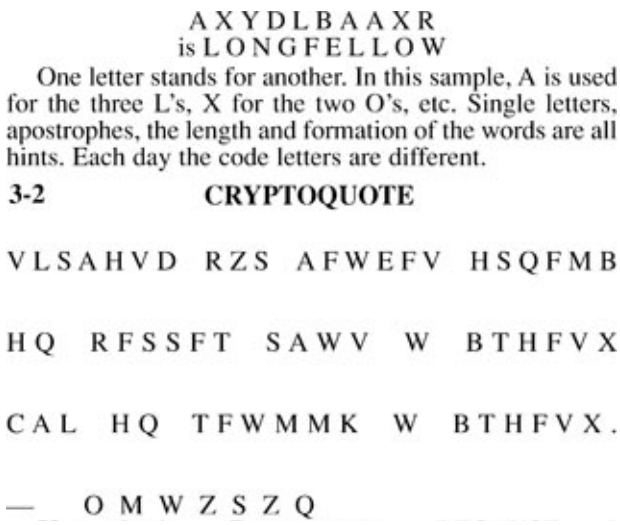
$$A, A, A, A, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7,$$
$$8, 8, 8, 8, 9, 9, 9, 9, 10, 10, 10, 10, J, J, J, J, Q, Q, Q, Q, K, K, K, K?$$

3. Cryptography

This section gives realistic scenarios in which the mathematics behind the card trick's secret can be used. We will divulge the secret of the trick in **Section 4**. For now, we say only that the trick is related to an attack on the Discrete Logarithm Problem (DLP), which also has ties to cryptography and will be discussed later in this section.

3.1 Ciphertext

Cryptography' goal is to send messages across an insecure channel to the intended recipient without eavesdroppers reading them. To achieve this goal, an encryption algorithm turns the original plaintext message into ciphertext, which in turn is sent to the intended recipient. The recipient uses a decryption algorithm to convert the message back into its original plaintext. Completing the Sunday paper's Cryptoquote is an example of deciphering ciphertext (**Figure 1**).



A X Y D L B A A X R
is L O N G F E L L O W

One letter stands for another. In this sample, A is used for the three L's, X for the two O's, etc. Single letters, apostrophes, the length and formation of the words are all hints. Each day the code letters are different.

3-2 **CRYPTOQUOTE**

V L S A H V D R Z S A F W E F V H S Q F M B

H Q R F S S F T S A W V W B T H F V X

C A L H Q T F W M M K W B T H F V X.

— O M W Z S Z Q

Figure 1. Cryptoquote. The correspondence in the sample (LONGFELLOW etc.) is not the same as the correspondence in the Cryptoquote. **Exercise 3** asks you to solve the Cryptoquote. (Courtesy of the *Arkansas Democrat Gazette*, 2 March 2011.)

The Cryptoquote puzzle uses a very naïve encryption algorithm, in which a bijective function f maps each letter to another letter. The decryption algorithm applies the inverse of f to the ciphertext and uncovers the original plaintext.

A stronger and more useful encryption algorithm would be a function that receives plaintext and a secret key as inputs and outputs ciphertext. The decryption algorithm would input ciphertext and a secret key and output plaintext. The key is not known to eavesdroppers. Thus, the security would rely less on the algorithm and more on the security of the keys.

3.2 Diffie-Hellman Key Exchange Protocol

How do two people agree on a key while keeping it secret? Certainly, two people can meet in person and decide on a key, but such a meeting is not always feasible. Instead, we consider a method by which two parties can agree on a secret key in communication across a possibly insecure channel, the Diffie-Hellman key exchange protocol, devised by Whitfield Diffie and

Martin Hellman [1976]. An earlier version was discovered independently in 1974 by Malcolm J. Williamson of British Government Communications Headquarters but kept classified. Singh [1999, Ch. 6, 243–292] tells the stories of the discoveries.

The Diffie-Hellman key exchange protocol uses the group of nonzero integers module p, denoted $\mathbb{Z}_p^*$. So we remind the reader of a relevant definition:

> An element g is a *generator* of a group G if for all $x \in G$, there exists an $n \in \mathbb{N}$ such that $g^n = x$. If so, we write $\langle g \rangle = G$.

Suppose that Alice and Bob wish to exchange private information. They can arrive at a shared key through the Diffie-Hellman protocol as detailed in **Figure 2**.

---

1. Alice and Bob agree on a prime p and an integer g such that $1 < g < p$ and $\langle g \rangle = Z_p^*$.
2. Now, Alice and Bob secretly choose integers x and y, respectively, with $1 < x, y < p$.
3. Alice sends $X = g^x \mod p$ to Bob and Bob sends $Y = g^y \pmod p$ to Alice.
4. Alice computes $Y^x = g^{xy} \mod p$ while Bob computes $X^y = g^{yx} \pmod p$. They arrive at the same number, which is the *key*.

Figure 2. The Diffie-Hellman key exchange protocol.

Example 2. In practice, the value of p is often large (with length 1024 or 2048 bits) and the computations are done with a computer algebra system. We use a small p, $p = 23$ (with 5 bits, since $23 = 10111_2$), to exemplify the steps of the protocol. We first find a generator g for the group $G = \mathbb{Z}_{23}^* = \{1, 2, \ldots, 22\}$. Any such generator will have the property that

$$g^{p-1} = 1 \pmod p, \tag{1}$$

but no integer smaller than $p - 1 = 22$ satisfies **(1)**. Since

$$2^{11} = 3^{11} = 1 \bmod 23,$$

neither 2 nor 3 generates the group. Because 2 is not a generator, $4 = 2^2$ is not a generator either. After verification that $5^{22} = 1 \bmod 23$ and no other power less than 22 satisfies $5^n = 1 \bmod 23$, we know that 5 is a generator. (Verification can be done with Maple using the command **mod** or with Mathematica using its `Mod[m,n]`.)

Now we arbitrarily pick two numbers, say $x = 6$ and $y = 15$, and compute

$$X = 5^6 = (5^2)^3 = 2^3 = 8 \pmod{23}$$
$$Y = 5^{15} = (5^2)^7 5 = (2^7)5 = (13)5 = 19 \pmod{23}.$$

Thus in G,

$$Y^x = 19^6 = (19^2)^3 19 = 16^3 = (3)16 = 2,$$
$$X^y = 8^{15} = (8^3)^5 = 6^5 = (6^2)^2 6 = (13^2)6 = (8)6 = 2,$$

so the key is 2. This key is then used to encrypt messages to be sent between the two parties. We have used laws of exponents to complete these computations in detail, though the use of a computer algebra system can simplify the work.

There are many ways in which a key can encrypt a message. A type of substitution cipher known as the *Caesar cipher* is credited to Julius Caesar. Applied to modern-day English letters, it would convert the letters A... Z to the numbers $0 \ldots 25$, respectively, and use a key k to shift each number x in the message to

$$y = x + k \pmod{26}.$$

The message could be decrypted by computing $y - k \pmod{26}$. If an eavesdropper on the message knew the key and the algorithm, the message could be deciphered. For this reason, the security of the key needs high priority.

In the Diffie-Hellman key exchange, p, g, X, and Y are considered *public*, meaning that an eavesdropper may know the values of these variables—and can know them without compromising the security of transmissions. But if in addition an eavesdropper can determine x and y, then he or she can calculate the key k—and with k, any encrypted message can be deciphered. Thus, solving $X = g^x$ for x and then $Y = g^y$ for y is the eavesdropper's goal, since the key is $k = g^{xy}$.

If this computation were being done over the real numbers, one would use logarithms to determine exponents x and y. However, the Diffie-Hellman protocol uses a finite group, in which the analogue to the logarithm defined over the real numbers is not easily computed. Calculating such a is known as the *discrete logarithm problem (DLP)*. This problem is highly nontrivial and no efficient general solution is known. There are, however, methods that are faster than the brute-force guess-and-check method of taking higher powers of g until $g^x = X$ for some x.

We measure efficiency in terms of the number of computational steps needed to complete the algorithm. A *polynomial-time algorithm* is one that can be run in no more than cn^k steps where c and k are positive constants

and n represents the length of the input of the algorithm. The length of the input is the number of bits needed to represent the input, not the value itself of the input. Examples of polynomial-time algorithms include addition, subtraction, multiplication, and division. There is no known method to solve the discrete logarithm problem in steps bounded by a polynomial in the length of the input. In the next section, we describe a method that runs in time $\mathcal{O}(w^{1/2}) = \mathcal{O}(e^{(\log w)/2})$, where w is a bound on the length of the inputs and $\mathcal{O}$ is big-oh notation for the order of magnitude.

But first we introduce a cryptosystem similar to the Diffie-Hellman key exchange protocol. Our intent is to show that the inherent difficulty in solving the DLP is useful in cryptography. An attack on the DLP will connect these cryptography concepts to the card trick.

3.3 ElGamal Cryptosystem

Bob selects a prime p and generator g of $\mathbb{Z}_p^*$. He also picks $a \in \mathbb{Z}_p^*$ such that $a < p-1$. He then makes his key (p, g, g^a) public. Now, suppose Alice that wants to send Bob a message. She encrypts the message through the following steps:

1. Look up Bob's public key (p, g, g^a).
2. Convert the message into integers $m_1, m_2, m_3, \ldots, m_n$, with $m_i \in \mathbb{Z}_p$.
3. Choose a random number $b \in \mathbb{Z}_p^*$ such that $b < p-1$.
4. Compute $B = g^b \pmod{p}$ and $C_i = m_i(g^a)^b \pmod{p}$ for each i.
5. Send the ciphertext $(B, C_1, C_2, \ldots, C_n)$ to Bob.

To decode the ciphertext, Bob uses the following algorithm.

1. Use the private key a to compute $B^{p-a} \pmod{p}$, the inverse of g^{ab}.
2. Compute, for each i,

$$B^{-a}C_i = (g^b)^{-a}m_i(g^a)^b = g^{-ab}m_i g^{ab} = m_i \pmod{p}.$$

Exercise

3. Determine f and the plaintext message of the ciphertext from **Figure 1**.

In **Example 2**, we showed an example of the Diffie-Hellman key exchange protocol using 5 as a generator. In the following exercises, we wish to explore how to find other generators of this cyclic group.

4. Lagrange's Theorem states that any subgroup H of a finite group G has the property that $|H|$ divides $|G|$:

$$|H| \,\big|\, |G|.$$

Use this theorem to determine the possible orders of $\langle x \rangle$ where $x \in \mathbb{Z}_{23}^*$.

5. Determine the other generators for $\mathbb{Z}_{23}^*$ using the assertion above and **(1)**.

6. Use a computer algebra system, plus **(1)**, to determine the smallest generator for Z_{102877}^*. The **mod**(e, m) command in Maple calculates $e \pmod m$, while `Mod[e,m]` does the same in Mathematica.

7. Identify the public and private information in **Example 2**. How might you determine the private numbers from the public ones?

8. In the ElGamal cryptosystem, prove the assertion that B^{p-a} is the inverse of g^{ab}.

9. Suppose that $p = 1777$, $g = 6$, and $a = 1009$. Use the ElGamal cryptosystem and either Maple or Mathematica to encipher $m = 1341$, assuming that $b = 701$.

10. Use the same values for p, g, a, and b as given in **Exercise 9** to decipher $C = 1031$ with $B = 1664$, using either Maple or Mathematica.

11. Identify the public and private components of the ElGamal cryptosystem. How do they compare to the public and private information in the Diffie-Hellman key exchange?

4. Pollard's Kangaroo Method: An Attack on the DLP

We now know of cryptosystems whose security relies on the difficulty of the DLP. Even though there is no general efficient way to solve the DLP, we discuss one way to attack the DLP, which foreshadows the card trick.

4.1 Jumping Kangaroos

Let G be a finite multiplicative cyclic group generated by g, and let $X \in G$. We can think of G as the group of units in some $\mathbb{Z}_k$.

We wish to solve $g^x = X$; the x is called the *index* of X. We begin by constructing two sequences:

- one sequence has what we consider "tame" behavior, because we know its starting position; and

- the other sequence has what we consider "wild" behavior, because it starts at a position unknown to us.

We can visualize these sequences as kangaroos jumping through the cyclic group, landing on elements of G (**Figure 3**).

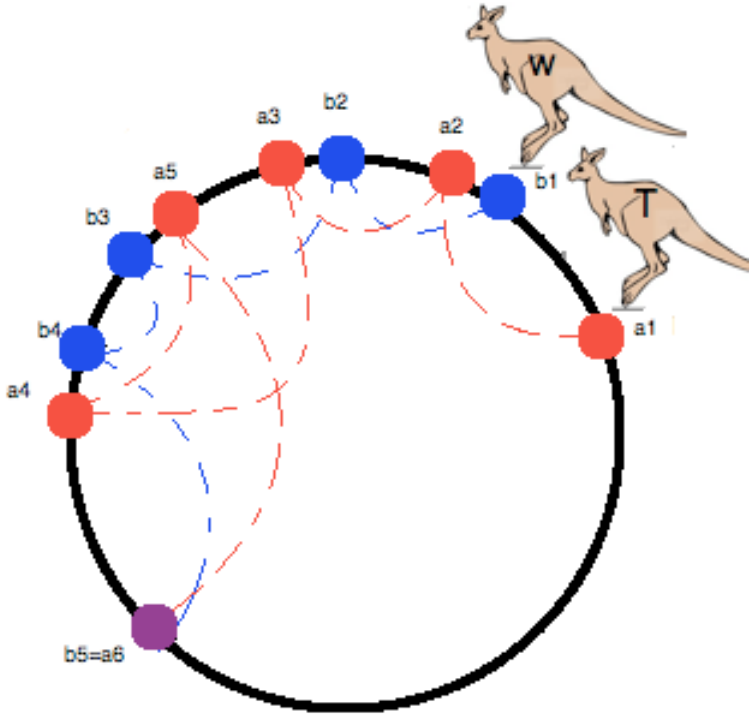


Figure 3. Conceptual diagram of the kangaroo method, with T and W denoting the tame and wild kangaroos.

To control the jumps of the kangaroos, we use a *hash function*

$$h : G \to J \subset \mathbb{Z}_k.$$

A hash function converts a large amount of data to a smaller amount; an example would be the function that converts your name to just your initials. We let our hash function h map the $|G|$ elements of G to the smaller set

$$J = \left\{1, 2, 3, 4, \ldots, \left\lfloor \sqrt{|G|} \right\rfloor\right\}$$

of size $\lfloor\sqrt{|G|}\rfloor$, where $\lfloor x \rfloor$ is the floor function that outputs the largest integer less than or equal to x. Further explanation of the choice of this hash function (and other possibilities) will follow after this section.

Now, the tame kangaroo starts at a known value $a_0 = g$ and proceeds by jumping a distance $h(a_0)$, so that at the end of its first jump it is at $a_1 = a_0 g^{h(a_0)}$. In general, with subsequent jumps, $a_{m+1} = a_m g^{h(a_m)}$. With recursion and simplification, we have

$$a_{m+1} = g^{1+h(a_0)+h(a_1)+\ldots+h(a_m)}.$$

After $m+1$ jumps, the kangaroo has traveled a distance

$$d_{m+1} = h(a_0) + h(a_1) + \ldots + h(a_m).$$

More importantly, the path of this tame kangaroo is comprised of powers of g, the generator of G.

The wild kangaroo travels on a path with unknown starting value $b_0 = X = g^x$. It uses the same hash function to determine jump size, so that $b_{n+1} = b_n g^{h(b_n)}$. We have

$$b_{n+1} = X g^{h(b_0)+h(b_1)+\ldots+h(b_n)}.$$

Here, the distance traveled by the wild kangaroo after $n+1$ jumps is

$$d'_{n+1} = h(b_0) + h(b_1) + \ldots + h(b_n).$$

The path of the wild kangaroo is composed of X times powers of g.

Now, if at some point the wild kangaroo lands at a site visited also by the tame kangaroo, we have the equation

$$g^{1+h(a_0)+h(a_1)+\ldots+h(a_m)} = X g^{h(b_0)+h(b_1)+\ldots+h(b_n)}. \qquad \textbf{(4..1)}$$

Thus, $X = g^{1+h(a_0)+h(a_1)+\ldots+h(a_m)-[h(b_0)+h(b_1)+\ldots+h(b_n)]}$ and we now know the index of X. Explicitly, we have

$$x = 1 + h(a_0) + h(a_1) + \ldots + h(a_m) - [h(b_0) + h(b_1) + \ldots + h(b_n)].$$

This attack is known as *Pollard's kangaroo method*, a set of steps that can lead to a solution of the DLP (which has connection to the Diffie-Hellman key exchange and to the ElGamal cryptosystem from **Section 3**). We will use the kangaroo method to explain the success of card trick. For now, we remark that both Pollard's kangaroos and the card trick's player (and the dealer) perform jumps, the former pair through a cyclic group and the latter pair through a deck of cards. We give an example of the kangaroo method.

Example 3. Let $G = \mathbb{Z}_{29}^*$ and $g = 3$, $X = 2$. We wish to find x such that $3^x = 2 \pmod{29}$. We have j = $\lfloor\sqrt{|G|}\rfloor = 5$, so let us define the hash function $h : G \to J = \{1, 2, 3, 4, 5\}$ as follows, where h repeats with period $8 = 2s - 2$ for $s = \lfloor\sqrt{|G|}\rfloor = \lfloor\sqrt{29}\rfloor = 5$:

a	1	2	3	4	5	6	7	8	9	10	11	$\cdots$	28
$h(a)$	1	2	3	4	5	4	3	2	1	2	3	$\cdots$	4

Then, modulo 29 we have

$$
\begin{aligned}
a_0 &= 3,\\
a_1 &= 3 \cdot 3^{h(3)} = 3 \cdot 3^3 = 3^4 = 23,\\
a_2 &= 3^4 \cdot 3^{h(23)} = 3^4 \cdot 3^3 = 3^7 = 12,\\
a_3 &= 3^7 \cdot 3^{h(12)} = 3^7 \cdot 3^4 = 3^{11} = 15,\\
a_4 &= 3^{11} \cdot 3^{h(15)} = 3^{11} \cdot 3^3 = 3^{14} = 28,\\
a_5 &= 3^{14} \cdot 3^{h(28)} = 3^{14} \cdot 3^4 = 3^{18} = 6,\\
a_6 &= 3^{18} \cdot 3^{h(6)} = 3^{18} \cdot 3^4 = 3^{22} = 22,
\end{aligned}
$$

and

$$
\begin{aligned}
b_0 &= 2,\\
b_1 &= 2 \cdot 3^{h(2)} = 2 \cdot 3^2 = 18,\\
b_2 &= 2 \cdot 3^2 \cdot 3^{h(18)} = 2 \cdot 3^4 = 17,\\
b_3 &= 2 \cdot 3^4 \cdot 3^{h(17)} = 2 \cdot 3^5 = 22.
\end{aligned}
$$

Since $a_6 = b_3$, we have $3^{22} = 2 \cdot 3^5 \pmod{29}$. Solving for 2 results in $3^{17} = 2 \pmod{29}$. Thus, in only 9 calculations, we were able to determine the index of 3 in this group. You can verify the answer by using the function **mod** in Maple or the function `Mod` in Mathematica.

4.2 Analysis of Pollard's Kangaroo Method

To analyze the method, we assume that:

The landing positions of the kangaroos are independent random samples from a uniform distribution of the elements of G.

Now, suppose that the tame kangaroo jumps a total of $c\sqrt{|G|}$ times for some constant c. Then, at every jump made by the wild kangaroo, there is a chance

$$\frac{c\sqrt{|G|}}{|G|} = \frac{c}{\sqrt{|G|}}$$

that the wild kangaroo lands on the tame kangaroo's path.

Suppose instead that the wild kangaroo misses the tame kangaroo's path at each of $\sqrt{|G|}$ jumps. This occurs with probability:

$$\left(1 - \frac{c}{\sqrt{|G|}}\right)^{\sqrt{|G|}} = \left(1 + \frac{1}{\frac{-\sqrt{|G|}}{c}}\right)^{\frac{-\sqrt{|G|}}{c}(-c)} \approx e^{-c},$$

using the facts from calculus that

$$\left(1+\frac{1}{n}\right)^n \approx e, \qquad \left(1-\frac{1}{n}\right)^{-n} \approx e$$

for large n. We want c to be small compared to $\sqrt{|G|}$, since the larger $\sqrt{|G|}/c$, the closer the approximation to e^{-c}. In addition, we do not want to calculate an infeasible number of jumps. So, the probability of failure is small when c is sufficiently large, since e^{-c} will be small. For instance,

c	1	2	3	4	5
e^{-c}	.3679	.1353	.0498	.0183	.0067

We credit the above analysis to Lacey [2002]; Pollard himself computes a similar analysis [1978].

4.3 Hash Functions

As promised, we discuss the choice of the hash function h, which establishes the jump sizes. It may seem a bit arbitrary. Pollard [2000a,438] states

> . . . the methods work well when the jumps of the kangaroos are powers of two. . . or powers of another number. We are not claiming that these are the only good choices. Possibly most sufficiently large sets are good choices.

He cites Oorschot and Wiener [1999] along with his own work [1978]in making this statement.

In the exercises, you will explore a hash function using powers of 2. But first, we provide an example of a poorly chosen hash function that results in the method not succeeding.

Example 4. Given $G = \mathbb{Z}_7^*$, which is generated by 3, suppose that we want to know the value of x for which $3^x = 6$. If we (foolishly) take $h(y) = 6$ for all $y \in G$, then $a_0 = g = 3$ and

$$a_{i+1} = a_i 3^{h(a_i)} = a_i 3^6 = a_i.$$

Since $a_i = 3$ for all i, the tame kangaroo's sequence is $\{3, 3, 3, \ldots\}$. Now, $b_0 = 6$ and

$$b_{i+1} = b_i 3^{h(b_i)} = b_i 3^6 = b_i.$$

Thus, the wild kangaroo's sequence is $\{6, 6, 6, \ldots\}$, and a collision will never occur.

The DLP was not solved, and the blame lies solely with the hash function. We need a hash function that inserts some variety into each kangaroo's sequence; a kangaroo that jumps in place is not of much use.

A hash function that guarantees success is the constant function $h(y) = 1$ for all $y \in G = \langle g \rangle$. The tame kangaroo's sequence is $\{g, g^2, g^3, \ldots\}$, so this kangaroo eventually jumps to each element in the group and hence must collide with the wild kangaroo at some point. But this hash function is no better than a guess-and-check method.

4.4 The Secret

Returning to the card trick, we relate it to the kangaroo method and use a similar analysis. Recall that the player's sequence is initialized by choosing a number between 1 and 10, followed by counting until the number of cards dealt is precisely the chosen number. From there, the first key card, the player secretly notes the value of the card (aces worth 1, face cards worth 5, and all others worth face value), with the player continuing to count from key card to key card. While the player is constructing a sequence through the deck, the dealer constructs her own sequence in the same manner. The prediction that the dealer makes of the player's last secret card is actually the last card in the dealer's sequence. Simply put, this is the trick's secret.

Most of the time, the trick ends with the dealer correctly guessing the player's last secret card. This means that at some point, either at the last card or before, the two sequences met at the same card. Just as in the kangaroo method, we have two sequences running through a set. We can view the dealer's and player's sequences as tame and wild kangaroos, respectively. Visualizing the two paths becoming one resembles the lowercase Greek letter λ. For this reason, Pollard's kangaroo method is sometimes referred to as the λ-*method*. **Figure 4** shows an example, using the sequences from **Example 3**.

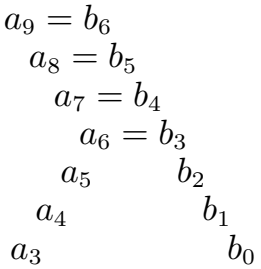


Figure 4. Graphical illustration of why the Pollard kangaroo method is sometimes called the λ-method.

Exercises

12. Use $G = \mathbb{Z}_{13}^*$ with $g = 6$ and $X = 3$ to determine $x \in \mathbb{Z}_{>0}$ such that $g^x = 6^x = X = 3 \pmod{13}$. Define $h : G \to J = \{1, 2, 3\}$ by the table below, where h repeats modulo $4 = 2s - 2$ for $s = \lfloor\sqrt{|G|}\rfloor = \lfloor\sqrt{13}\rfloor = 3$. It may be useful to note that $6^{12} = 1$ and $6^{-1} = 6^{11}$.

a	1	2	3	4	5	6	7	8	9	10	11	12
$h(a)$	1	2	3	2	1	2	3	2	1	2	3	2

13. Use $G = \mathbb{Z}_{102877}^*$ and the generator g found in **Exercise 6** to apply Pollard's kangaroo method to determine x in $g^x = 7 \pmod{102877}$. Use either the Maple or the Mathematica code in **Appendix A**, which take as inputs a prime p, a generator g, and an integer $X \in Z_p^*$. The output is the index of X in Z_p^* for generator g, arrived at by Pollard's kangaroo method. (The result can be confirmed using the command `MultiplicativeOrder[g,p,{X}]` in Mathematica.)

14. In the computer code in **Appendix A**, we have $s = \lfloor\sqrt{|G|}\rfloor$ and the hash function h:

Hash function in code of Appendix A

a	1	2	3	$\cdots$	s	$s+1$	$s+2$	$\cdots$	$2s-2$	$2s-1$	$2s$	$2s+1$	$\cdots$
$h(a)$	1	2	3	$\cdots$	s	$s-1$	$s-2$	$\cdots$	2	1	2	3	$\cdots$

Rewrite either the Maple or the Mathematica code using the following hash function g, where t is chosen such that $2^t < s$:

New hash function

a	1	2	3	$\cdots$	$t+1$	$t+2$	$t+3$	$\cdots$	$2t$	$2t+1$	$2t+2$	$2t+3$	$\cdots$
$g(a)$	2^0	2^1	2^2	$\cdots$	2^t	2^{t-1}	2^{t-2}	$\cdots$	2^1	2^0	2^1	2^2	$\cdots$

15. Run both the computer code from **Appendix A** and the revised code from **Exercise 14** using the numbers from **Exercise 13**. Does either hash function lead to a quicker collision of the sequences? What did you compare to come to this conclusion? Rerun the code using a few different inputs to determine if one hash function leads to quicker collisions.

16. Prove that if $a_n = b_m$ in Pollard's kangaroo method, then $a_{n+k} = b_{m+k}$ for all $k \geq 0$.

5. Markov Chains

Thus far, this Module has explored cryptography, the discrete log problem, and Pollard's kangaroo method, all of which relate to the card trick described in the Introduction. In the card trick, the "jump" to the player's next key card depends on only the current key card. For instance, suppose that the player's first six key cards are $8, 8, 5, K, K, 3$ (as in **Example 1**). The next key card would be three cards from the 3 card. The important point is that the next "jump" only depends on the current card, not on the previous cards in the sequence. This is similar to Pollard's kangaroo method, in which the next position depends only on the current position.

Both the card trick and the kangaroo method relate can be represented in terms of Markov chains, a useful tool to analyze long-term behavior of models using probability and matrix theory. Below we summarize results about Markov chains from Grinstead [1997], Kemeny and Snell [1960], and Roberts [1976], to which we refer the reader for further details and proofs of the results.

5.1 Results about Markov Chains

A *Markov process* (or *Markov chain*) is a system of states in which the probability of moving from one state to another depends only on the current state, not on states visited in the past.

We can display the probabilities involved in a *transition matrix*, with the previous states corresponding to rows and the next state corresponding to columns. A simple example for a system with three states is:

$$T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} .5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \\ 0.25 & 0.25 & 0.5 \end{pmatrix} \end{array}.$$

Theorem. *For a Markov process with transition matrix T and states $1, 2, \ldots, n$, the probability that the process will be in state j after k steps given that it started in state i is $T^k_{i,j}$, for $1 \le i, j \le n$.*

For a Markov chain with transition matrix, T, suppose that we are given the probabilities that the process begins in each of the n states as a vector $v = \{v_1, v_2, \ldots, v_n\}$. Then vT^k is the vector whose jth component gives the probability that the process is in state j after k steps.

A set of states is a *transient set* if there is a way in which to leave the set, and once the process leaves the set, there is zero probability that the process will return to any state in the set. A state is *transient* if it is in a transient set.

An *absorbing state* is a state from which one cannot leave. An *absorbing Markov chain* is a Markov chain that contains at least one absorbing state

and a path from each nonabsorbing state to an absorbing state.

An absorbing Markov chain can be simplified to a *canonical form*, as shown in **Figure 5**.

$$\begin{array}{c} \\ \textit{absorbing states} \\ \textit{nonabsorbing states} \end{array} \begin{array}{c} \begin{array}{cc} \textit{absorbing states} & \textit{nonabsorbing states} \end{array} \\ \begin{pmatrix} I & 0 \\ R & Q \end{pmatrix} \end{array}$$

Figure 5. Canonical form of a transition matrix for an absorbing Markov chain.

Let there be m absorbing states and $n - m$ nonabsorbing states. Then

- I is the $m \times m$ identity matrix composed of absorbing states,
- 0 is the $m \times (n - m)$ zero matrix,
- R is an $(n - m) \times m$ matrix, and
- Q is an $(n - m) \times (n - m)$ matrix composed of probabilities that the process moves from a transient state to a transient state.

The *fundamental matrix* for an absorbing Markov chain with transition matrix in canonical form is $N = (I - Q)^{-1}$.

The fundamental matrix N has useful properties:

- The entries of N give the expected value of the number of times the process is in each nonabsorbing state for each possible starting state.
- The sum of a row i of N is the expected value of the number of steps before the process is absorbed, assuming that it started in state i.
- NR is the matrix whose entries are the probabilities that the process ends in a particular absorbing state for each possible nonabsorbing starting state.

Exercises

17. Explain why each row of a transition matrix T must sum to 1.

18. An equivalence relation $\sim$ can be placed on a Markov chain: Two states, s_i and s_j, are considered equivalent and we write $s_i \sim s_j$ if either $i = j$ or there is a path from s_i to s_j and a path from s_j to s_i. Verify that $\sim$ is an equivalence relation. This equivalence relation partitions the states into two equivalence classes.

19. Explain why an absorbing chain cannot have a transition matrix with the property that $T_{ij}^n > 0$ for some n and all i, j.

20. For an absorbing Markov chain with transition matrix T, where we have $\lim_{n\to\infty} Q^n = 0$, confirm that $I - Q$ has an inverse by finding an explicit form for it.

21. Describe a scenario in which $NR = [1]$, the 1×1 identity matrix.

6. A Simplified Kruskal Count as a Markov Process

To view the card trick as a Markov process, we must first modify the deck to lead to more-feasible probability calculations. As with the original trick, we distinguish only the face value of the card, not the suit or color. We follow the treatment in Haga and Robins [1997]: To give every value equal probability, we toss out the face cards; and we assume an infinite random deck, with each value 1 through 10 having an equal and independent chance of being at any position in the deck.

We define the states of the Markov process as corresponding to the *distances*—the numbers of cards—between the dealer's current key card and the player's key card that is immediately ahead of it, or zero if they are the same card. With cards valued 1 though 10, this distance can be 0 through 9, so we have 10 states in the Markov chain. The state corresponding to the distance 0 is absorbing, since when the player's card and the dealer's card are the same, their paths have converged and the distance between each successive card remains 0. We define entry $M_{i,j}$ in the transition matrix to be the probability that the distance goes from i to j.

We illustrate with an example using a deck of cards whose values are only 1 and 2.

Example 5. Suppose that we have an infinite deck of cards with values 1 and 2, with each value equally likely at each position and independent of values at all other positions. For the card trick, the states of the Markov chain correspond to the number of cards between the dealer's current card and the closest player's card that is even with or ahead of the dealer's card, so the states are 0 and 1. The card trick is initialized by the player secretly choosing a number from $\{1, 2\}$ and then counting that number of cards to the next key card. We compute each entry of the transition matrix:

- $M_{0,0}$: the probability that the distance between the sequences starts at 0 and stays at 0 after one turn. If the distance is 0, then the player and dealer are on the same card and will move the same number of spaces to again be on the same card after one turn. Therefore, $M_{0,0} = 1$.
- $M_{0,1}$: the probability that the distance between the sequences

starts at 0 and then becomes 1 after a turn. By the explanation above, this is not possible. So, $M_{0,1} = 0$.

- $M_{1,1}$:the probability that the distance between the sequences starts at 1 and stays 1 after a turn. If the distance is 1, then the player is on the card directly after the dealer's card. The equally probable options for two successive cards are: 1 1, 1 2, 2 1, 2 2. The first three cases result in the player and dealer's sequences colliding on the next turn. When the dealer and player are on cards 2 2, they are 1 apart and will stay 1 apart after completing the turn. See the **Figure 6**, in which d_i and p_j represent the dealer's ith card and player's jth card, respectively. The cases in the figure confirm that $M_{1,1} = \frac{1}{4}$.

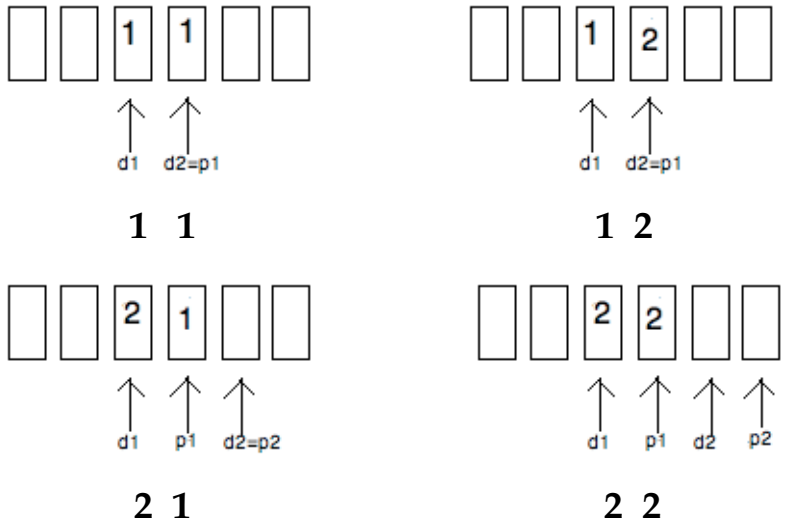


Figure 6. Cases for **Example 5** relating to $M(1,1)$.

- $M_{1,0}$: the probability that the distance between the sequences starts at one and then becomes 0 after a turn. By the explanation above, $M_{1,0} = \frac{3}{4}$ and

$$M = \begin{bmatrix} 1 & 0 \\ 3/4 & 1/4 \end{bmatrix}.$$

This matrix is already in canonical form with $N = (I - Q)^{-1} = [4/3]$. According to the results about Markov chains, the expected number of turns until the process is absorbed is $4/3$.

Now we explore what happens for a deck of cards with 10 independently, identically distributed values. The transition matrix M is a 10×10 matrix indexed from 0 to 9. From Haga and Robins [1997], we have the following two theorems.

Theorem. *The transition matrix M satisfies*

$$(a)\ M_{0,0} = 1 \ \text{ and } \ M_{0,j} = 0, \qquad \text{for } \ 0 \le j \le 9;$$

$$(b)\ M_{9,9} = \frac{1}{100} \ \text{ and } \ M_{9,j} = \frac{1}{10}\left(1 + \frac{1}{10}\right), \qquad \text{for } 0 \le j \le 8;$$

$$(c)\ M_{i,j} = \left(1 + \frac{1}{10}\right) M_{i+1,j}, \qquad \text{for } 0 < i \ne j < 9;$$

$$(d)\ M_{i,i} = \left(1 + \frac{1}{10}\right) M_{i+1,i} - \frac{1}{10}, \qquad \text{for } 0 < i < 9.$$

Proof: (a) Since a distance of 0 is the absorbing state, once the distance between the cards is 0, it will remain 0. Thus, $M_{0,0} = 1$ and $M_{0,j} = 0$ for $0 \le j \le 9$.

(b) The only way in which the distance between the cards is 9 and stays 9 is if both the dealer and the player have cards of value 10. This occurs with probability $\left(\frac{1}{10}\right)\left(\frac{1}{10}\right)$. Thus, $M_{9,9} = \frac{1}{100}$.

The remainder of the proof can be found in **Appendix B**. □

By induction, Haga and Robins [1997] get a closed form for M:

Theorem.

$$M_{i,i} = \frac{1}{10}\left[\left(1 + \frac{1}{10}\right)^{10-i} - 1\right], \qquad \text{for } 1 \le i \le 8;$$

$$M_{i,j} = \frac{1}{10}\left(1 + \frac{1}{10}\right)^{10-i}, \qquad \text{for } 0 < j < i < 9;$$

$$M_{i,j} = \frac{1}{10}\left(1 + \frac{1}{10}\right)^{j-i}\left[\left(1 + \frac{1}{10}\right)^{10-j} - 1\right], \quad \text{for } 0 < i < j \le 9.$$

Example 6. We verify the entry for $M_{8,9}$ by examining all possible paths that might be taken for the distance between the dealer and player to go from 8 to 9.

First, we suppose that the player's key card is 8 cards ahead of the dealer. The distance will become 9 if either of the following occurs:

- The dealer's key card is a 9 and the player's key card is a 10. Since all card values have equal and independent probability $1/10$, this occurs with probability $1/10^2$.
- The dealer's key card is a 10, the player's current key card is a 1, and the player's next key card is a 10. This case occurs with probability $1/10^3$.

The sum of these two probabilities is

$$M_{8,9} = \frac{1}{10^2} + \frac{1}{10^3} = \frac{1}{10}\left(1 + \frac{1}{10}\right)\frac{1}{10},$$

as stated in the theorem.

A more detailed construction of this matrix is done in Haga and Robins [1997], where there is also a general form for the transition matrix M for a deck with values 1 through m.

The closed form of the matrix M is already in canonical form, so and can calculate the matrices Q, R, and N. Since N provides the most important output, we explicitly calculate it:

$$\frac{1}{11}\begin{bmatrix} 20 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 10 & 19 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 10 & 9 & 18 & 6 & 5 & 4 & 3 & 2 & 1 \\ 10 & 9 & 8 & 17 & 5 & 4 & 3 & 2 & 1 \\ 10 & 9 & 8 & 7 & 16 & 4 & 3 & 2 & 1 \\ 10 & 9 & 8 & 7 & 6 & 15 & 3 & 2 & 1 \\ 10 & 9 & 8 & 7 & 6 & 5 & 14 & 2 & 1 \\ 10 & 9 & 8 & 7 & 6 & 5 & 4 & 13 & 1 \\ 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 12 \end{bmatrix}$$

Thus, summing the rows of N yields a 9×1 matrix:

$$\frac{1}{11}\begin{bmatrix} 56 & 57 & 58 & 59 & 60 & 61 & 62 & 63 & 64 \end{bmatrix}^T.$$

Recall that the summed rows of matrix N yield the expected values of the number of steps before the process is absorbed for each nonabsorbing starting state. For instance, if we start in state 1, meaning that the distance between the dealer's card is 1 away from the player's card, we expect $56/11 = 5.\overline{09}$ turns before the distance between the cards becomes 0. Over the nine nonabsorbing states, the largest such value is $64/11 = 5.\overline{81}$ turns, for a distance of 9. Though the deck is potentially infinite, we expect success after only a few turns! While this is not a concrete argument for why the card trick will work a majority of the time, it provides evidence for that conclusion. For a more mathematically convincing rationale that the trick works approximately $5/6$ of the time, see Lagarias et al. [2009] and Pollard [2000a].

Exercise

22. Verify that the entry for $M_{7,9} = \frac{1}{10}\left(1 + \frac{1}{10}\right)^2\left(\frac{1}{10}\right)$ by determining all possible paths that might be taken for the distance between the dealer and player to go from 7 to 9.

7. The Kruskal Count

The trick that we have been discussing is credited to mathematician Martin D. Kruskal and is sometimes called the *Kruskal Count*.

7.1 How to Increase the Chance of the Trick's Success

Further analysis has shown that the dealer can increase the chance of correctly guessing the player's last card by

- **using two decks of cards instead of one:** Using more cards lengthens the dealer's and player's paths, thus increasing the likelihood that the two paths will eventually meet.
- **decreasing the value of the face cards:** Decreasing the value of the face cards increases the average number of cards on the dealer's and player's paths, making the probability of collision greater. This would be the result of a version of the trick where each face card has value equal to the number of letters in its name: A jack is worth 4, queen worth 5, and king worth 4.
- **starting the dealer's sequence with the first card rather than randomly picking a digit 1 to 10:** By starting on the first card (i.e., choosing a secret number to be 1), the dealer potentially lengthens the dealer's sequence of cards, which increases the probability that the player will land on one of them.

In each of these modifications, we increase the number of jumps by the kangaroos, thereby making the chance of landing on the same card more probable. For a rigorous analysis of alterations that improve the probability of success, see Lagarias et al. [2009].

7.2 Estimating the Chance of Success

The Markov chain analysis in **Section 6** provides an expected value until success, but not a probability of success. So we develop a second Markov chain for the card trick, one that allows us to bound the probability of failure of the trick. Lagarias et al. [2009] present a "reduced" version of the chain that we develop here.

Define the Markov chain as follows. We deal from two independent decks of cards with independent and equally distributed values from the set $S = 1, 2, \ldots, L$. The dealer plays the card trick on one deck and the player uses the other deck. Both choose a number from S; and starting from the top of their decks, they simultaneously count down the cards until their number is reached. The card on which each lands is the first key card for each.

We consider one card to be *behind* another if it is closer to the top of its deck.

- Whichever card is behind the other moves again until it is ahead of the other deck's key card. [This is similar to the order of turns in the game of golf.] As before, the value of the key card is the number of cards counted until reaching the next key card.
- If both cards are an equal distance from the top of the deck, both the player and the dealer move onto the next key card in their deck.

We define the *distance* as the difference between the player's current key card and the top of the deck less the difference between the dealer's current key card and the top of the deck. The distance can range from $-(L-1)$ to $L-1$.

The states of the Markov chain are the $2L-1$ distances

$$-(L-1), -(L-2), \ldots, -1, 0, 1, \ldots, L-2, L-1.$$

The time until the distance becomes 0 is called the *coupling time*, which we denote by τ. We perform the actions described above until the Nth card in one of the decks is passed. We calculate the probability of failure of the decks to couple by the Nth card, i.e., $P(\tau > N)$.

First, we define the random variable $Z_{L,N}$ to be the total number of key cards in both decks up to and including the Nth card. Thus, our probability space is $\Omega = \bigcup_{k=0}^{\infty}\{Z_{L,N} = k\}$. So,

$$\begin{aligned}
\{\omega \in \Omega \mid \tau(\omega) > N\} &= \{\omega \in \Omega \mid \tau(\omega) > N\} \cap \left(\bigcup_{k=0}^{\infty}\{Z_{L,N} = k\}\right) \\
\implies \{\tau > N\} &= \bigcup_{k=0}^{\infty}\left(\{\tau > N\} \cap \{Z_{L,N} = k\}\right) \\
\implies P(\tau > N) &= \sum_{k=0}^{\infty} P(\tau > N,\ Z_{L,N} = k) \\
&= P(\tau > N \mid Z_{L,N} = k)\, P(Z_{L,N} = k) \\
&= \sum_{k=0}^{\infty}\left(\frac{L-1}{L}\right)^{k} \cdot P(Z_{L,N} = k) \\
&= E\left[\left(1 - \frac{1}{L}\right)^{Z_{L,N}}\right].
\end{aligned}$$

Since $Z_{L,N} \geq 2N/L$, we have

$$\left(1 - \frac{1}{L}\right)^{Z_{L,N}} \leq \left(1 - \frac{1}{L}\right)^{2N/L}.$$

Thus,

$$E\left[\left(1-\frac{1}{L}\right)^{Z_{L,N}}\right] \le E\left[\left(1-\frac{1}{L}\right)^{2N/L}\right] = \left(1-\frac{1}{L}\right)^{2N/L}.$$

Therefore, $P(\tau > N) \le \left(1-\frac{1}{L}\right)^{2N/L}$.

Exercises

23. Suppose $L = 5$. This means that the two decks are composed of a uniform distribution of 5 cards: Ace, 2, 3, 4, and 5. Determine the transition matrix T of size 5×5 for this Markov chain. Why can't we use the analysis performed in **Section 6** on this Markov chain?

24. Assuming $L = 10$ (since a standard deck of cards has card values 1, 2, 3, ..., 10) and $N = 52$, compute the probability that the card trick will succeed.

8. Other Results and Open Problems

The DLP's difficulty gives rise to several algorithms in cryptography. Apart from the Diffie-Hellman key exchange and the ElGamal cryptosystem (whose security relies on the DLP), other cryptosystems using the DLP include the U.S. Government's Digital Signature Algorithm and its elliptic curve analogue [Teske 2001].

Montenegro and Tetali [2009] provide bounds on the probability of success that supplement work by Pollard [2000a].

As is common practice, in our analysis we took a uniformly-distributed deck of cards. However, a standard deck of 52 cards with the original assignment of card values does not have equally likely card values; so our calculations did not give an exact calculation of the probability trick's success. Pollard himself admits that "An exact calculation seems difficult" [Pollard 2000b]. He gives an approximate calculation to show that we expect the trick to succeed with probability 89.3%, together with a simulation that admits 85.4% success.

Grime [n.d.] uses a geometric distribution and gives an approximate probability of success to be 83.88%. Grime also includes several other interesting results, including probabilities on the placement of the final card in the sequence and how many possible different last cards can occur. In his simulation, 58.39% of the decks had the following property: Independent of the 10 possible starting positions, the final card reached was the same. Moreover, taking all 10 starting positions into account, 97.9% of decks will have no more than two different possible final cards. We end this discussion with proof of a result of Pollard that any deck of cards cannot have more than six different final cards.

The goal is to show that seven final cards or more is impossible. To do so, we examine the length of a path through the deck. We show that the total of the minimum possible lengths of seven distinct paths is longer than the total value of all the cards in the entire deck. For a standard 52-card deck and the usual card values, that total value is

$$4 \cdot (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 5 + 5 + 5) = 280.$$

We look at the 10 sequences formed by the 10 starting positions. Suppose that these result in seven different final cards. Now, take one additional turn with each of these seven cards; doing so will lead beyond the last of the 52 cards, but we just imagine the deck to extend a bit further. The total length of each card's extended path is minimized if the path ends right on the imaginary 53rd card (let it have value 0). The lengths of the extended paths that end at the 53rd card and start at cards 4 through 10 (which are the shortest seven such paths) are

$$\begin{aligned} 53 - 4 &= 49 \\ 53 - 5 &= 48 \\ 53 - 6 &= 47 \\ 53 - 7 &= 46 \\ 53 - 8 &= 45 \\ 53 - 9 &= 44 \\ 53 - 10 &= 43 \end{aligned}$$

for a total length of 322, which is greater than 280. This makes seven paths impossible.

Six paths have a minimum total length of $322 - 49 = 273$, so six is the upper bound on number of final cards possible at the end of the trick. However, no example of an ordered deck with six final cards appears in the literature. The minimum total length discussed would involve starting the six paths on cards 5 through 10 and finishing on cards with values $6, 5, 4, 3, 2, 1$ where the value 6 would appear on the 47th card in the deck, value 5 on the 48th, etc.

9. Conclusion

The purpose of this Module was to introduce Pollard's kangaroo method via an interesting hook (the card trick). The Diffie-Hellman key exchange and the ElGamal cryptosystem both rely on the difficulty of solving the DLP, for which the kangaroo method provides an attack. We used Markov chains to achieve results about the card trick, concluding that it is successful most of the time. We encourage the reader to attempt the role of dealer and perform the trick on an audience!

10. Answers to Exercises

1. Regardless of which secret number $(1, 2, \ldots, 10)$ is chosen, the last card will be the 51st card in the deck, a Queen.

2.

Secret Number	Last Card
$1, 2, 3, 4, 5, 7, 9$	fourth Queen
$6, 8, 10$	first King

3. Not all letters were encrypted, so we skip several in the definition of f in **Table S1**. Quote: "Nothing but heaven itself is better than a friend who is really a friend." —Plautus

x	A	B	D	E	F	G	H	I	L	N	O	P	R	S	T	U	V	W
$f(x)$	W	R	X	F	B	D	A	H	M	V	L	O	T	Q	S	Z	E	C

Table S1. Solution for Exercise 3.

4. $|\langle x \rangle| \in \{1, 2, 11, 22\}$.

5. We need to find x such that $x^1 \neq 1$, $x^2 \neq 1$, $x^{11} \neq 1$, but $x^{22} = 1$. The acceptable x are 5, 7, 10, 11, 14, 15, 17, 19, 20, and 21. The Mathematica command `Solve[x^11==22,{x},Modulus->23]` gives these values, plus 22, which does not satisfy $x^2 \neq 1$.

6. Since $|\mathbb{Z}^*_{102877}| = 102866 = 2^2 \cdot 3 \cdot 8573$, we need to find x such that $x \neq 1$, $x^2 \neq 1$, $x^3 \neq 1$, $x^4 \neq 1$, $x^6 \neq 1$, $x^{12} \neq 1$, $x^{8573} \neq 1$,, $x^{17146} \neq 1$, $x^{25719} \neq 1$,, $x^{34292} \neq 1$, and $x^{51428} \neq 1$, but $x^{102866} = 1$. Using Maple or Mathematica, it can be verified that $x = 2$ qualifies (and of course is the smallest such).

7. The public information is $X = 8, Y = 19, p = 23$, and $g = 5$. The private information is $Y^x = 2 = X^y$. This can be calculated if one can solve $19^x = 2$ for x or $8^y = 2$ for y with computations in Z^*_{23}.

8. $B^{p-a} g^{ab} = (g^b)^{p-a} g^{ab} = g^{bp-ab+ab} = (g^p)^b = 1^b = 1$.

9. $B = 1664$ and $C = 1103$. The ciphertext is $(1664, 1103)$.

10. $m = 108$.

11. The public information is p, g, g^a, $B = g^b$, and $C = mg^{ab}$. The private information is a, b, and m. Since we know g, g^b, and p, if we can solve the DLP, we can compute a and b, which will lead to m. This is the same as the Diffie-Hellman key exchange in which an eavesdropper tries to solve for x if Y and Y^x are known.

12.
$$\begin{aligned} a_0 &= 6, & b_0 &= 4, \\ a_1 &= 6 \cdot 6^2 = 6^3 = 8, & b_1 &= 4 \cdot 6^2 = 1, \\ a_2 &= 6^3 \cdot 6^2 = 6^5 = 2, & b_2 &= 4 \cdot 6^2 \cdot 6 = 4 \cdot 6^3 = 6. \end{aligned}$$
The collision occurs when $a_0 = 6 = b_2$. Thus, $6 = 4 \cdot 6^3$ and $6^{-2} = 4$. Since $6^{-1} = 6^{11}$, we have $4 = 6^{22} = 6^{12+10} = 6^{12}6^{10} = 6^{10}$.

13. 86843.

14. The new Maple code with changes is shown in **Figure S1** and the Mathematica code in **Figure S2**.

```
> Index2 := proc(p, g, x)
  local s, L1, L2, p1, p2, i, k, m, n, h, f, z, y, t;
  s := floor(sqrt(p)); L1 := [g]; p1 := [1]; t := floor( ln(floor(sqrt(p))) / ln(2) );
  print(t);
  L2 := [x]; p2 := [0]; i := 1; k := 0;
  while k = 0 do
      for z in L1 do
          for y in L2 do
              if (k = 0) and (z = y) then k := 1;
              print(z, y); member(z, L1,'u'); member(y, L2,'v');
              break; end if;
          end do
      end do;
      m := mod(op(i, L1) - 1, 2·t); n := mod(op(i, L2) - 1, 2· t);
      if m ≤ t then h := 2^m; else h := 2^(2·t - m); end if;
      L1 := [op(L1), mod(op(i, L1)·g^h, p)]; p1 := [op(p1), op(i, p1) + h];
      if n ≤ t then f := 2^n; else f := 2^(2·t - n); end if;
      L2 := [op(L2), mod(op(i, L2)·g^f, p)]; p2 := [op(p2), op(i, p2) + f];
      i := i + 1;
  end do;
  print(L1); print(L2); print(p1); print(p2);
  if op(u, p1) - op(v, p2) > 0 then print(op(u, p1) - op(v, p2));
  else print(p - 1 + (op(u, p1) - op(v, p2)));end if;
  end proc;
```

Figure S1. Revised Maple code for solution to **Exercise 14**, with changes in red (light gray).

15. Running **Index2** yields the same answer of 86843 that **Index** provided. The sequences a_i and b_j are longer for **Index2**. Running each program on the numbers from **Example 3** results in sequences of equal length.

```
Index[p_, g_, X_] := Module[
  {s, Collision, L1, L2, p1, p2, i, m, h, n, f, u, v},
  s = Floor[Sqrt[p]];
  Collision = False;
  L1 = {g};
  L2 = {X};
  p1 = {1};
  p2 = {0};
  u = 0;
  v = 0;
  i = 1;
  While[(Collision == False && i < p),
   (If[Intersection[L1, L2] ≠ {},
     (u = Flatten[Position[L1, Intersection[L1, L2][[1]]]];
      Collision = True), Collision = False];
    If[Intersection[L1, L2] ≠ {},
     v = Flatten[Position[L2, Intersection[L1, L2][[1]]]];
    m = Mod[L1[[i]], 2 * s - 2];
    If[m == 0, h = 2, If[m ≤ s, h = m, h = 2 * s - m]];
    L1 = Append[L1, Mod[(Last[L1] * g^h), p]];
    p1 = Append[p1, Last[p1] + h];
    n = Mod[L2[[i]], 2 * s - 2];
    If[n == 0, f = 2, If[n ≤ s, f = n, f = 2 * s - n]];
    L2 = Append[L2, Mod[(Last[L2] * g^f), p]];
    p2 = Append[p2, Last[p2] + f];
    i = i + 1)];
  Print["tame jump locations: ", L1];
  Print["wild jump locations: ", L2];
  Print["tame powers of p: ", p1];
  Print["wild powers of p: ", p2];
  Print["tame intersection location: ", u];
  Print["wild intersection location: ", v];
  If [i ≠ p,
   (If [
     (p1[[u]][[1]] - p2[[v]][[1]] > 0),
     (Print["index of X for generator g: ",
       p1[[u]][[1]] - p2[[v]][[1]]]),
     (Print["index of X for generator g: ",
       p - 1 + p1[[u]][[1]] - p2[[v]][[1]]])]),
    Print[
     "there is no such index"];]]
```

Figure S2. Revised Mathematica code for solution to **Exercise 14**, with changes in red (light gray).

As another example, it can be verified that 1987 is prime with generator 2. Suppose that we want to find the index of 1000. Both programs Index2 and **Index2A** yield the answer $2^{1356} = 1000$, but again **Index2** shows longer sequences for a_i and b_j.

16. Suppose that $a_n = b_m$; then $a_{n+1} = a_n g^{h(a_n)} = b_m g^{h(b_m)} = b_{m+1}$. Now, suppose that $a_{n+k} = b_{m+k}$ for some $k \geq 1$. Then

$$a_{n+k+1} = a_{n+k} g^{h(a_{n+k})} = b_{m+k} g^{h(b_{m+k})} = b_{m+k+1}.$$

17. The columns of a transition matrix T represent all possible states of the process. Given that the process is currently in row i, it either stays in state i or moves to one of the other states by the next move. As such, $\sum_j T_{i,j} = 1$.

18. Symmetric: If $s_i \sim s_j$ then $s_j \sim s_i$ by definition. Reflexive: Additionally, $s_i \sim s_i$ by definition. Transitive: For $s_i \sim s_j$ and $s_j \sim s_k$, there is a path from s_i to s_j to s_k and a path from s_k to s_j to s_i, so $s_i \sim s_k$.

19. In an absorbing Markov chain's transition matrix, there must be an entry of 1 somewhere and all other entries in that row must be 0. For any power of this matrix, the same argument will hold.

20. $(I - Q)^{-1} = \sum_{n=0}^{\infty} Q^n$. Since $\lim_{n\to\infty} Q^n = 0$ (because the sum converges),

$$(I - Q)(I + Q + Q^2 + Q^3 + \cdots) = I.$$

21. Since NR is a matrix whose entries represent the probabilities that the process ends in a particular absorbing state for each nonabsorbing state, $NR = [1]$ means that there is one nonabsorbing state in the process.

22. Begin by supposing that the player's card is 7 cards ahead of the dealer's card. For the distance to go from 7 to 9, one of the following must occur:

- The dealer's current card is an 8 and the player's current card is a 10. The probability of this occurring is $(1/10)^2$.
- The dealer's current card is a 9 and the player's current card is 1. Since distance is defined by a player's card being ahead of the dealer's card, this turn is not complete until the player moves again. For the distance to be 9, the player's card must be a 10. The probability is $(1/10)\,(1/10)^2$.
- The dealer's current card is a 10 and either
  - the player's next cards are 1, 1, and 10; or

– the player's next cards are 2 and 10.

This probability is $\frac{1}{10}\left[\left(\frac{1}{10}\right)^3 + \left(\frac{1}{10}\right)^2\right]$.

In total, $M_{7,9} = \frac{1}{100}\left(1 + \frac{1}{10}\right)^2$.

23. $T = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1/5 & 8/25 & 6/25 & 4/25 & 2/25 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 2/5 & 1/5 & 1/5 & 0 \\ 1/5 & 2/5 & 2/5 & 0 & 0 \\ 1/5 & 2/5 & 1/5 & 1/5 & 0 \end{pmatrix} \end{array}.$

This is not an absorbing Markov chain.

24. $P(\{\tau > N\}) \le \left(1 - \frac{1}{10}\right)^{104/10} \approx 0.334$. So, the probability of success is at least $1 - 0.334 = 0.666$.

11. Appendix A: Computer Code

This code in **Figures A1** and **A2** is for use with **Exercises 13–15**.

12. Appendix B: Continuation of Proof

We continue expositing the proof from Haga and Robins [1997] of the first of their Theorems in **Section 6**.

First, we introduce some notation. Define sequences (y_i) and (x_i) to be the dealer's and player's secret cards, respectively. Let d_k be the distance between the dealer's current card y_i and the player's next-closest card appearing after the dealer's card in the deck, x_j.

To prove part (b), we assume that $d_k = 9$ and $d_{k+1} = j < 9$. There are two ways in which this can happen:

- Either the dealer's card has face value $9 - j$, with probability $1/10$; or
- the dealer's card has face value 10 and the player's card has face value $j + 1$, with probability $1/100$.

Thus, $M_{9,j} = \frac{1}{10}\left(1 + \frac{1}{10}\right)$.

For $M_{i,j}$ we will compare the **Figures B1 and B2**:

Now, we claim that the difference between computing $M_{i+1,j}$ depicted in the **Figure B1** and $M_{i,j}$ depicted in **Figure B2** is the card just before y_{k+1}, which Haga and Robins call "star" and we denote by $*$.

If $*$ in **Figure B2** is vacant, then the player's cards can occupy any of the empty boxes between x_l and y_{k+1} in the same number of ways as **Figure B1**. Thus, when $*$ is vacant, $M_{i,j} = M_{i+1,j}$.

```
>Index:=proc(p,g,x) %Input prime p, generator, g, and element x
local s,L1,L2,p1,p2,i,k,m,n,h,f,z,y %identify local variables
s:= floor(sqrt (p)); %s = ⌊√|G|⌋ where |G| = p
L1:=[g]; L2:=[x]; %L1 and L2 will record the sequences (a_i) and (b_i)
p1:=[1]; p2:=[0]; %p1 and p2 will keep track of the power of g for each a_i and b_i
as a list
i=1; k:=0; %starting values for other variables to be used in loops
while k=0 do
for z in L1 do
for y in L2 do
if (k=0) and (z=y) then k:=1 %if z = y then there is a collision
print(z,y); member(z,L1,'u');member(y,L2,'v');
break; end if;
end do
end do;

m:=mod(op(i,L1),2s-2); %Determine g(a_i)
if m=0 then h:=2;
elif m ≤ s then h:=m;
else h:= 2s-m; end if;
L1:=[op(L1),mod(op(i,L1)· g^h,p)]; p1:=[op(p1),op(i,p1)+h];

n:=mod(op(i,L2),2s-2); %Determine g(b_i)
if n=0 then f:=2;
elif n ≤ s then f:=n;
else f:=2s-n; end if;
L2:=[op(L2), mod(op(i,L2)·g^f,p)]; p2:=[op(p2), op(i,p2)+f];
i:=i+1;
end do;
print(L1); print(L2); print(p1); print(p2);
if op(u,p1)-op(v,p2)>0 then print(op(u,p1)-op(v,p2));
else print(p-1+(op(u,p1)-op(v,p2))); end if;
end proc;
```

Figure A1. Maple code for Pollard's kangaroo method.

```
Index[p_, g_, X_] := Module[
  {s, Collision, L1, L2, p1, p2, i, m, h, n, f, u, v},
  s = Floor[Sqrt[p]];
  Collision = False;
  L1 = {g};
  L2 = {X};
  p1 = {1};
  p2 = {0};
  u = 0;
  v = 0;
  i = 1;
  While[(Collision == False && i < p),
   (If[Intersection[L1, L2] ≠ {},
     (u = Flatten[Position[L1, Intersection[L1, L2][[1]]]];
      Collision = True), Collision = False];
    If[Intersection[L1, L2] ≠ {},
     v = Flatten[Position[L2, Intersection[L1, L2][[1]]]];
    m = Mod[L1[[i]], 2 * s - 2];
    If[m == 0, h = 2, If[m ≤ s, h = m, h = 2 * s - m]];
    L1 = Append[L1, Mod[(Last[L1] * g^h), p]];
    p1 = Append[p1, Last[p1] + h];
    n = Mod[L2[[i]], 2 * s - 2];
    If[n == 0, f = 2, If[n ≤ s, f = n, f = 2 * s - n]];
    L2 = Append[L2, Mod[(Last[L2] * g^f), p]];
    p2 = Append[p2, Last[p2] + f];
    i = i + 1)];
  Print["tame jump locations: ", L1];
  Print["wild jump locations: ", L2];
  Print["tame powers of p: ", p1];
  Print["wild powers of p: ", p2];
  Print["tame intersection location: ", u];
  Print["wild intersection location: ", v];
  If [i ≠ p,
   (If [
     (p1[[u]][[1]] - p2[[v]][[1]] > 0),
     (Print["index of X for generator g: ",
       p1[[u]][[1]] - p2[[v]][[1]]]),
     (Print["index of X for generator g: ",
       p - 1 + p1[[u]][[1]] - p2[[v]][[1]]])]),
    Print[
     "there is no such index"];]]
```

Figure A2. Revised Mathematica code for solution to **Exercise 14**, with changes in red (light gray).

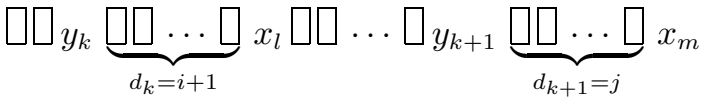


Figure B1.

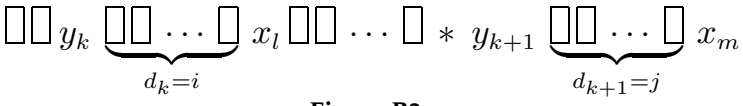


Figure B2.

When a player's card occupies $*$, the player's sequence of cards would have followed a path described when $d_k = i + 1$ and $d_{k+1} = j$, but with one additional turn, since some x_n lands on $*$.

When $*$ is occupied, $M_{i,j} = \frac{1}{10} M_{i+1,j}$.

In total,

$$M_{i,j} = \left(1 + \frac{1}{10}\right) M_{i+1,j}, \qquad \textit{for } i \neq j \textit{ and } 0, 9 \neq i.$$

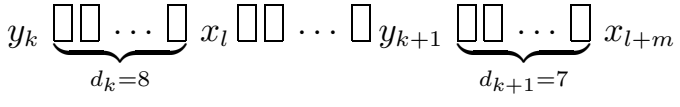
Last, we examine part (d) from the theorem. The paths in which $d_k = i + 1$ and $d_{k+1} = i$ include the cases:

- y_k has face value 1, which occurs with probability $1/10$.
- Any other path can be modified to work in the case where $d_k = d_{k+1} = i$ by forcing y_k to have face value one less. This accounts for all cases in which $y_k \neq 10$ and we have $M_{i,i} = M_{i+1,i} - \frac{1}{10}$.
- y_k has face value 10; we again examine a path traveled by the player's cards in which $d_k = i + 1$ and $d_{k+1} = i$. Say this path has $y_k = p$. Then we modify the path of the player's cards by inserting a card of face value $10 - p + 1$ as the first card in the player's path and all other cards having an unchanged face value (see example). So, this means when $y_k = 10$, $M_{i,i} = \frac{1}{10} M_{i+1,i}$. In total, $M_{i,i} = \left(1 + \frac{1}{10}\right) M_{i+1,i} - \frac{1}{10}$.

  Example for Proof: We want to show explicitly

$$M_{7,7} = \left(1 + \frac{1}{10}\right) M_{8,7} - \frac{1}{10}.$$

  First, we label the paths in which $d_k = 8$ and $d_{k+1} = 7$:



  Path 1: $y_k = 1$ and $x_l = x_{l+m}$
  Path 2: $y_k = 9$ and $x_l = 8$

Path 3: $y_k = 10$ and $x_l = 1,\ x_l + 1 = 8$
Path 4: $y_k = 10$ and $x_l = 9$

Now, we modify these paths in the two ways described in the proof above.

(a) Decrease y_k by 1 for $y_k \neq 1$. So, we have the paths given below.
Path 2a: $y_k = 8$ and $x_l = 8$
Path 3a: $y_k = 9$ and $x_l = 1,\ x_{l+1} = 8$
Path 4a: $y_k = 9$ and $x_l = 9$
Since the values of the cards are equally distributed throughout the deck, the probabilities of paths 2a, 3a, and 4a are equal to the probabilities of paths 2, 3, and 4, respectively.

(b) Change y_k to 10 and insert extra x_j. The paths are then,
Path 1b: $y_k = 10$ and $x_l = 10$
Path 2b: $y_k = 10$ and $x_l = 2,\ x_{l+1} = 8$
Path 3b: $y_k = 10$ and $x_l = 1,\ x_{l+1} = 1,\ x_{l+2} = 8$
Path 4b: $y_k = 10$ and $x_l = 1,\ x_{l+1} = 9$
Since we have inserted an additional card into the path, the probabilities of paths 1b, 2b, 3b, and 4b are each $\frac{1}{10}$ times the probability of paths 1, 2, 3, and 4, respectively.

In total, we have $M_{7,7} = \left(1 + \frac{1}{10}\right) M_{8,7} - \frac{1}{10}$.

References

Diffie, W., and M. Hellman. 1976. New directions in cryptography. *IEEE Transactions on Information Theory* IT-22 (6): 644–654.

Gardner, Martin. 1978. Mathematical Games: On checker jumping, the Amazon game, weird dice, card tricks and other playful pastimes. *Scientific American*, 238 (2) (February): 19–32. 1989. Reprinted under the title: Sicherman dice, the Kruskal count and other curiosities. Chapter 19 in *Penrose Tiles to Trapdoor Ciphers... and the Return of Dr. Matrix*. New York: W.H. Freeman. 1997. Rev. ed., with addendum, 265–280. Washington, DC: Mathematical Association of America. 2005. Reproduced in *Martin Gardner's Mathematical Games*. CD-ROM. Washington, DC: Mathematical Association of America.

Grime, J. n.d. Kruskal's count. `http://www.singingbanana.com/Kruskal.pdf` .

Grinstead, C.M. 1997. *Introduction to Probability*. Providence, RI: American Mathematical Society.

Haga, Wayne, and Sinai Robins. 1997. On Kruskal's principle. *Organic Mathematics; Proceedings of the Organic Mathematics Workshop*. 20: 407–412. `http://oldweb.cecm.sfu.ca/cgi-bin/organics/doc_vault/name=paper+access_path=vault/robins` . A demonstration of the

card trick—with face cards having value 1—is at `http://oldweb.cecm.sfu.ca/cgi-bin/organics/carddemo.pl`.

Kemeny, J., and J. Snell. 1960. *Finite Markov Chains*. Princeton, NJ: D. Van Nostrand Company, Inc.

_____, and G. Thompson. 1957. *Introduction to Finite Mathematics.* Englewood Cliffs, NJ: Prentice-Hall.

Klima, R. 1999. Applying the Diffie-Hellman key exchange to RSA. *The UMAP Journal* 20 (1): 21–27.

Lacey, Michael T. 2002. Cryptography, card tricks, and kangaroos. `http://people.math.gatech.edu/~lacey/talks/roos.pdf`.

Lagarias, Jeffrey C., Eric Rains, and Robert J. Vanderbei. 2001. The Kruskal count. `http://arxiv.org/abs/math/0110143`. 2009. In *The Mathematics of Preference, Choice and Order: Essays in Honor of Peter J. Fishburn*, edited by S. Brams, W.V. Gehrlein, and F.S. Roberts, 371–391. New York: Springer-Verlag.

Montenegro, Ravi. 2009. Kruskal count and kangaroo method. `http://faculty.uml.edu/rmontenegro/research/kruskal_count/index.html`. Includes links to one-deck and two-deck demonstrations of the card trick (with face cards having value 5) at `http://faculty.uml.edu/rmontenegro/research/kruskal_count/kruskal.html`, with a demonstration of variations on the kangaroo method to solve the DSP at `http://faculty.uml.edu/rmontenegro/research/kruskal_count/kangaroo.html`.

_____, and Prasad Tetali. 2009. How long does it take to catch a wild kangaroo? *Proceedings of 41st ACM Symposium on Theory of Computing (STOC 2009)*, 553–559. 2010. Rev. version (v2). `http://arxiv.org/abs/0812.0789`.

van Oorschot, P.C., and M.J. Wiener. 1999 Parallel collision search with cryptanalytic applications. *Journal of Cryptology* 12: 1–28. `http://homes.chass.utoronto.ca/~krybakov/links_files/papers/parallel%20colision%20search.pdf`.

Pollard, J.M. 1978. Monte Carlo methods for index computation $(\mathrm{mod}\ p)$. *Mathematics of Computation* 32 (No. 143, July 1978): 918–924. `http://www.ams.org/journals/mcom/1978-32-143/S0025-5718-1978-0491431-9/S0025-5718-1978-0491431-9.pdf`.

_____. 2000a. Kangaroos, Monopoly and discrete logarithms. *Journal of Cryptology* 13: 437–447. `http://www4.ncsu.edu/~singer/437/Kangaroos.pdf`.

_____. 2000b. 84.29 Kruskal's card trick. *Mathematical Gazette* 84 (No. 500, July 2000): 265–267. `http://www4.ncsu.edu/~singer/437/Kruskal_card.pdf`.

Roberts, F.S. 1976. *Discrete Mathematical Models*. Englewood Cliffs, NJ: Prentice-Hall.

Singh, Simon. 1999. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. New York: Doubleday.

Teske, E. 2001. Square-root algorithms for the discrete logarithm problem (a survey). In *Public-Key Cryptography and Computational Number Theory*, edited by Kazimierz Alster, Jerzy Urbanowicz, and Hugh C. Williams, 283–301. New York: Walter de Gruyter.

Acknowledgments

I would like to acknowledge the assistance of Dr. Michael Singer. His guidance and feedback were instrumental in completing this project. Thank you also to Dr. Ernie Stitzinger for his assistance in this project. A special thank-you to Dr. Min Kang, whose knowledge of probability and patience allowed me to take this project further. We would also like to thank the referee for the very helpful comments, references, and Mathematica codes.

The author was partially supported by NSF Grants CCR-0634123 and CCF-1017217.

About the Author

Lindsey Bosko...