# SYMMETRIC AND PUBLIC KEY ENCRYPTION LESSON

Mark Emry, McNeil High School, Round Rock ISD, Austin, TX

This material is based upon work supported by
the **National Science Foundation** under Grant No.1548315.

Additional materials may be found at www.ncyte.net

# SYMMETRIC AND PUBLIC KEY CRYPTOGRAPHY

Prior to World War II, cryptography's focus was on creating and breaking ciphers. The proliferation of high-speed computers following World War II, resulted in the creation of ciphers (e.g., DES) which were fast, reliable, and essentially unbreakable. The explosive growth of the Internet created a need for efficient and reliable encryption between counterparties who were essentially anonymous. As such, the focus of Modern Cryptography shifted from deciphering encrypted messages to finding ways to reliably and efficiently pass encryption/decryption keys between anonymous counterparties. This Cybersecurity Concept Lesson (CCL) advances student understanding of modern symmetric and public key exchange methodologies through a) an investigation of the historical context into which Modern Cryptography fits b) hands-on, in-class activities, c) a teacher-led discussion, and d) a multi-day programming assignment. The programming assignment will both a) advance students' understanding of modern key exchange methodologies, and b) advance students' programming ability through engagement with a problem of substance.

The complete mathematics of modern key exchange methodologies likely surpass the capabilities of many high school students. As such, many high school curricula introduce these methodologies as concepts and then gloss over much of the detail about how these methodologies work. This CCL takes a different approach. By going deeper than many high school curricula, but at the same time avoiding the more esoteric mathematical aspects of these methodologies, this CCL navigates a middle course. Here, students actually program a slightly simplified version of the Symmetric Key Exchange Method, and, by so doing, gain a deeper and more connected understanding of how these key exchange methodologies function, how they are different, and how both methodologies fit into the historical context of modern Cryptography. A not unappreciated benefit of this approach is that students use programming to gain a deeper understanding of important concepts and, by so doing, become more practiced programmers.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

## OVERVIEW

**Prerequisite Knowledge:** This CCL is optimally sequenced after student work on programming is complete. Specifically, students taking this CCL should be familiar with the following programming fundamentals: conditional statements, iteration, arrays, array processing, strings, modulus mathematics, and string processing. For students who have not completed their study of programming a pseudo-code version of the programming exercise has been included. This CCL works best as a continuation of the two-day Caesar Cipher CCL.

**Length of Completion**: The CCL is designed to take approximately 100-150 minutes. A programming assignment is required to be completed outside of class.

**Homework**: Students read the *Wikipedia* article on Public Key Cryptography prior to starting Day 1. Additionally, students watch the Kahn Academy video on the historical context of modern Cryptography, the Discrete Logarithm Problem and the Khan Academy video on the Diffie-Hellman key exchange

Students watch the following short *Kahn Academy* videos on Public Key Encryption prior to the start of Day 2:

- RSA 1
- RSA 2
- RSA 3
- Euler's function, and
- RSA 4

Students are not expected to understand all the mathematics reviewed in these videos. Rather, students should watch with an eye towards appreciating the complexity, beauty, and rigor of the underlying mathematics.

Students should prepare for a multiple-choice assessment which is scheduled for the end of Day 2. Students submit a programming assignment (described below) on the day immediately after the end of this CCL.

**Learning Setting:** The CCL assumes that students will be able to move around and work in small groups.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

**Lab Environment:** While the CCL does not specify a particular programming language, it assumes that students have in-class access to the Internet and an IDE they are familiar with.

**Activity/Lab Tasks:** The CCL includes four separate activities: two in-class, unplugged activities, one summative assessment, and one multi-day programming assignment. A programming activity will span both days of the CCL.

- 02.SymmetricAndPublicKeyEncryption_Presentation.pptx
- 03.SymmetricAndPublicKeyEncryption_PublicKeyEncryptionReview.docx
- 05.SymmetricAndPublicKeyEncryption_Encryption_Berkley.pdf
- 06.SymmetricAndPublicKeyEncryption_MultiDay_ProgrammingExercise.docx
- 07.SymmetricAndPublicKeyEncryption_InClass_UnpluggedPublicKeyEncryptionExercise.docx

06.Programming Folder

06.Processing Folder
  - SymmetricAndPublicKeyEncryption_CipherFullProcessingCode.docx

06.Pseudocode Folder
  - SymmetricAndPublicKeyEncryption_PseudoCode_FullAnswer.docx
  - SymmetricAndPublicKeyEncryption_PseudoCodeReference.pdf

06.Python Folder
  - SymmetricAndPublicKeyEncryption_CipherFullPythonCode.docx

## LEARNING OBJECTIVES AND AP CSP ALIGNMENT

### Lesson Learning Objectives

Students will:

1) be able to explain Modern Cryptography's historical context
2) be able to explain the difference between Symmetric and Public Key Encryption methodologies
3) be able to use more advanced mathematical operators (e.g., modulus, exponents) to strengthen their programming capabilities.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

## ASSOCIATED AP CSP SUB LEARNING OBJECTIVES

### AP COMPUTER SCIENCE PRINCIPLES COURSE, BIG IDEA 1: CREATIVE DEVELOPMENT

- LO CRD-1.A Explain how computing innovations are improved through collaboration.
    - CRD-1.A.3 Effective collaboration produces a computing innovation that reflects the diversity of talents and perspectives of those who designed it.
- LO CRD-1.C Demonstrate effective interpersonal skills during collaboration.
    - CRD-1.C.1 Effective collaborative teams practice interpersonal skills, including but not limited to: communication, consensus building, conflict resolution, and negotiation.
- LO CRD-2.B Explain how a program or code segment functions.
    - CRD-2.B.1 A program is a collection of program statements that performs a specific task when run by a computer. A program is often referred to as software.
- LO CRD-2.I For errors in an algorithm or program:
        - Identify the error.
        - Correct the error.
    - CRD-2.I.1 A logic error is a mistake in the algorithm or program that causes it to behave incorrectly or unexpectedly.
    - CRD-2.I.2 A syntax error is a mistake in the program where the rules of the programming language are not followed.
    - CRD-2.I.3 A run-time error is a mistake in the program that occurs during the execution of a program. Programming languages define their own run-time errors.

### AP COMPUTER SCIENCE PRINCIPLES COURSE, BIG IDEA 3: ALGORITHMS AND PROGRAMMING

- LO AAP-1.A Represent a value with a variable.
    - AAP-1.A.1 A variable is an abstraction inside a program that can hold a value. Each variable has associated data storage that represents one value at a time, but that value can be a list or other collection that in turn contains multiple values.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

- AAP-1.A.2 Using meaningful variable names helps with the readability of program code and understanding of what values are represented by the variables.
  - AAP-1.A.3 Some programming languages provide types to represent data, which are referenced using variables. These types include numbers, Booleans, lists, and strings.
  - AAP-1.A.4 Some values are better suited to representation using one type of data rather than another.
- LO AAP-1.C Represent a list or string using a variable.
  AAP-1.C.1 A list is an ordered sequence of elements. For example, [value1, value2, value3, …] describes a list where value1 is the first element, value2 is the second element, value3 is the third element, and so on.
  - AAP-1.C.3 An index is a common method for referencing the elements in a list or string using natural numbers.
  - AAP-1.C.4 A string is an ordered sequence of characters.
- LO AAP-2.C Evaluate expressions that use arithmetic operators.
  - AAP-2.C.1 Arithmetic operators are part of most programming languages and include addition, subtraction, multiplication, division, and modulus operators.
  - AAP-2.C.2 The exam reference sheet provides a MOD b, which evaluates to the remainder when a is divided by b. Assume that a is an integer greater than or equal to 0 and b is an integer greater than 0. For example, 17 MOD 5 evaluates to 2.
- LO AAP-2.D Evaluate expressions that manipulate strings.
  - AAP-2.D.1 String concatenation joins together two or more strings end-to-end to make a new string.
- LO AAP-2.G Express an algorithm that uses selection without using a programming language.
  - AAP-2.G.1 Selection determines which parts of an algorithm are executed based on a condition being true or false
- LO AAP-2.H For selection:
    - Write conditional statements.
    - Determine the result of conditional statements.
  - AAP-2.H.1 Conditional statements or "if-statements" affect the sequential flow of control by executing different statements based on the value of a Boolean expression.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the National Science Foundation under Grant No.1548315.

- LO AAP-2.K For iteration:
    - Write iteration statements.
    - Determine the result or side-effect of iteration statements.
  - AAP-2.K.1 Iteration statements change the sequential flow of control by repeating a set of statements zero or more times, until a stopping condition is met.
- LO AAP-2.M For algorithms:
    - Create algorithms.
    - Combine and modify existing algorithms.
  - AAP-2.M.1 Algorithms can be created from an idea, by combining existing algorithms, or by modifying existing algorithms.
- LO AAP-2.N For list operations:
    - Write expressions that use list indexing and list procedures.
    - Evaluate expressions that use list indexing and list procedures.
  - AAP-2.N.1 The exam reference sheet provides basic operations on lists, including: accessing an element by index, assigning a value of an element of a list to a variable, assigning a value to an element of a list, inserting elements at a given index, adding elements to the end of the list, removing elements, and determining the length of a list.
  - AAP-2.N.2 List procedures are implemented in accordance with the syntax rules of the programming language.
- LO AAP-2.O For algorithms involving elements of a list:
    - Write iteration statements to traverse a list.
    - Determine the result of an algorithm that includes list traversals.
  - AAP-2.O.1Traversing a list can be a complete traversal, where all elements in the list are accessed, or a partial traversal, where only a portion of elements are accessed.
  - AAP-2.O.2 Iteration statements can be used to traverse a list.
- LO AAP-3.A For procedure calls:
    - Write statements to call procedures.
    - Determine the result or effect of a procedure call.
  - AAP-3.A.1 A procedure is a named group of programming instructions that may have parameters and return values.
  - AAP-3.A.2 Procedures are referred to by different names, such as method or function, depending on the programming language.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

- AAP-3.A.3 Parameters are input variables of a procedure. Arguments specify the values of the parameters when a procedure is called.
- AAP-3.A.4 A procedure call interrupts the sequential execution of statements, causing the program to execute the statements within the procedure before continuing. Once the last statement in the procedure (or a return statement) has executed, flow of control is returned to the point immediately following where the procedure was called.

- LO AAP-3.B Explain how the use of procedural abstraction manages complexity in a program.
  - AAP-3.B.1 One common type of abstraction is procedural abstraction, which provides a name for a process and allows a procedure to be used only knowing what it does, not how it does it.
  - AAP-3.B.2 Procedural abstraction allows a solution to a large problem to be based on the solution of smaller subproblems. This is accomplished by creating procedures to solve each of the subproblems.
  - AAP-3.B.5 Using parameters allows procedures to be generalized, enabling the procedures to be reused with a range of input values or arguments.
- LO AAP-3.E For generating random values:
    - Write expressions to generate possible values.
    - Evaluate expressions to determine the possible results
  - AAP-3.E.1 The exam reference sheet provides RANDOM(a, b) which generates and returns a random integer from a to b, inclusive. Each result is equally likely to occur. For example, RANDOM (1, 3) could return 1, 2, or 3.
  - AAP-3.E.2 Using random number generation in a program means each execution may produce a different result.

## AP COMPUTER SCIENCE PRINCIPLES COURSE, BIG IDEA 5: IMPACT OF COMPUTING

- LO IOC-2.C Explain how unauthorized access to computing resources is gained.
  - IOC-2.C.1 Phishing is a technique that attempts to trick a user into providing personal information. That personal information can then be used to access sensitive online resources, such as bank accounts and emails.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

## LESSON DETAILS

**Overview of Lessons:** The CCL is broken down into two 60-minute lessons taught on consecutive days:

Day 1:

- Teacher-Led Discussion (20 minutes)
- In-Class Paint Activity (20 minutes)
- Start of Programming (20 minutes)

Day 2:

- Continue Programming (20 minutes)
- In-Class Unplugged Public Key Encryption Exercise (30 minutes)

| | DAY 1 | | | DAY 2 |
|---|---|---|---|---|
| Teacher-led Discussion on Symmetric and Public Key Encryption, and Programming | In-Class Unplugged Diffie-Hellman Key Exchange Exercise | Start Programming | Continue Programming | In-Class Unplugged Public Key Encryption Exercise |
| Programming | Assignment | Diffie-Hellman | Key | Exchange |

## DAY 1

**Teacher-Led Discussion (20 minutes)**
02.SymmetricAndPublicKeyEncryption_Presentation.pptx
The teacher should give the PowerPoint presentation and lead a classroom discussion.  To the extent possible, the teacher should reference the Khan Academy videos on Public Key Encryption that students watched as homework for the first day of the CCL.  By connecting the previous night's homework in this manner, students will feel better prepared for this new material and more comfortable raising issues and participating in this teacher-led discussion.

Students gain an understanding of how Symmetric and Public Key Exchange methodologies work, the historical context of these related methodologies, their respective advantages and disadvantages, and how Symmetric and Public Key Cryptography are the foundation for secure Internet communication.

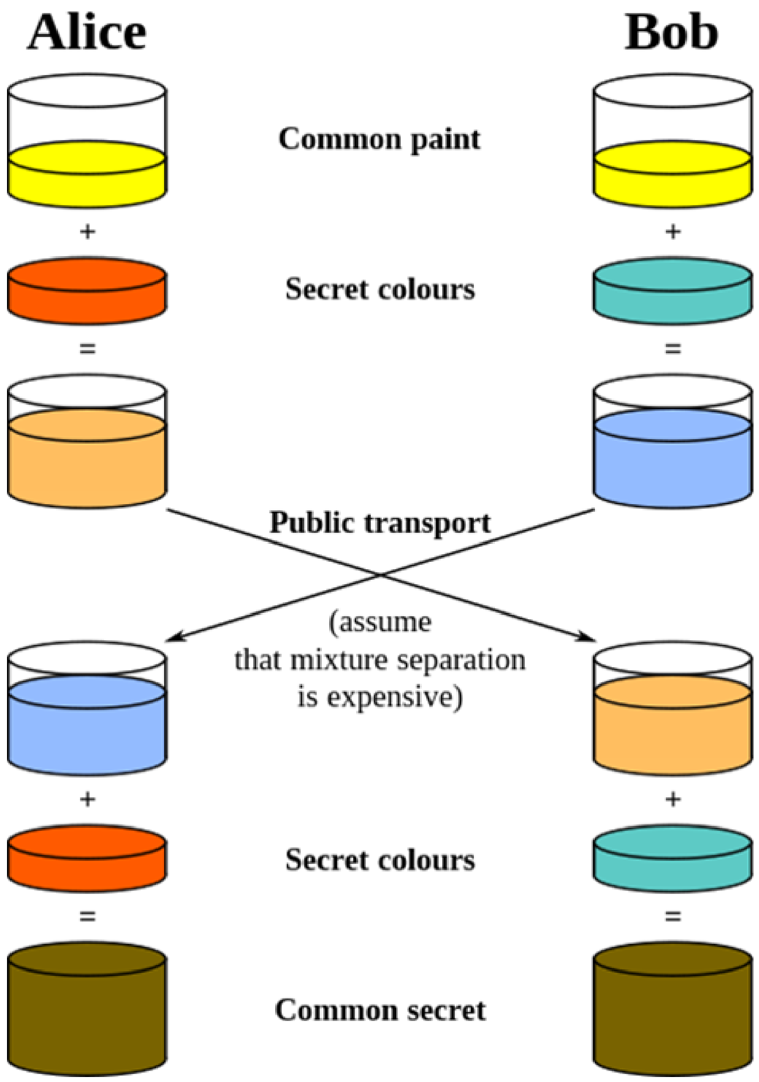
This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the National Science Foundation under Grant No.1548315.

**In-Class Unplugged Symmetric Key Exchange Exercise (20 minutes)**

This activity should be done in groups of two. Once the groups have been selected, the students within each group should designate themselves either Alice or Bob, such that each two-person group has one Alice and one Bob.

At the beginning of class, the teacher should establish a central color repository. There should be paint dabs (approximately two ounces each) of at least ten distinct paint colors made available to students in a central repository. The exercise will work best if the paint dabs are as distinct as possible. The teacher should also place a popsicle stick (or reasonable alternative) next to each color. These popsicle sticks allow students to transfer the three required colors to their respective, personal activity sheets.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.



The activity should be done in the following steps.

1. Alice should place her dab of paint into her Alice Secret Color Circle (Alice keeps this color a secret).
2. Bob should place his dab of paint into his Bob Secret Color Circle (Bob keeps this color a secret).
3. Alice and Bob should agree together on a Public Key Color Circle (the color should be placed into the public area visible to Eve).
4. Alice should put a combined Alice Secret Color/ Public Key Color into Alice's Combined Secret Color/Public Key Color Circle (this color is placed

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

into the public area visible to Eve once the two colors are thoroughly mixed).

5. Alice should take note of the two colors which she used to create her Combined Secret Color/Public Key Color.
6. Bob should put a combined Bob Secret Color/ Public Key Color into Bob's Combined Secret Color/Public Key Color Circle (this color is placed into the public area visible to Eve once the two colors are thoroughly mixed).
7. Bob should take note of the two colors which he used to create his Combined Secret Color/Public Key Color.
8. Alice should combine Bob's Combined Secret Color/Public Key Color with her Alice Secret Color and place the result into her Shared Secret Color Circle (this combination will be visible only to Alice).
9. Bob should combine Alice's Combined Secret Color/Public Key Color with his Bob Secret Color and place the result into his Shared Secret Color Circle (this combination will be visible only to Bob).
10. Bob and Alice should hide their respective Secret Color Circles and Shared Secret Color Circles.
11. Bob and Alice should then randomly select an Eve from one of the other groups.
12. The point of the game is for Eve to try to guess Bob's and Alice's respective Secret Colors given only the Public Key Color, Alice's Combined Secret Color, and Bob's Combined Secret Color.

Once done, the teacher should debrief the exercise drawing student attention to the similarities between this exercise and Symmetric Key Exchange Methodologies (e.g., Diffie-Hellman).

The students develop an intuitive sense of two separate issues. Firstly, students learn how Symmetric Key Exchange methodologies work from this extended analogy exercise. Secondly, students learn, first-hand, the ease with which a one-way function (mixing of paint) can create a public secret and the difficulty with which the process can be reversed (guessing the underlying private secret paint components which created the mix). Such one-way functions underlie both Symmetric Key Exchange and Public Key Exchange Methodologies.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

**Start of Programming (20 minutes)**
06.SymmetricAndPublicKeyEncryption_MultiDay_ProgrammingExercise.docx
Programming Folder
Students begin programming in class.  The teacher should look for common problems encountered by students and encourage collaborative problem-solving without plagiarism.

## DAY 2

**Continue Programming (20 more minutes)**
06.SymmetricAndPublicKeyEncryption_MultiDayProgrammingExercise.docx
Programming Folder

All of the above, with the addition that the students will have benefit of at-home work and reflection on programming problems encountered as part of their overnight programming efforts.

**In-Class Unplugged Public Key Encryption Exercise (30 minutes)**
07.SymmetricAndPublicKeyEncryption_InClass_UnpluggedPublicKeyEncryptionExercise.docx
This activity should be done in groups of three.  Once the groups have been selected, each student should designate themselves either as Alice or Bob, each three-person group has two Alices and one Bob.

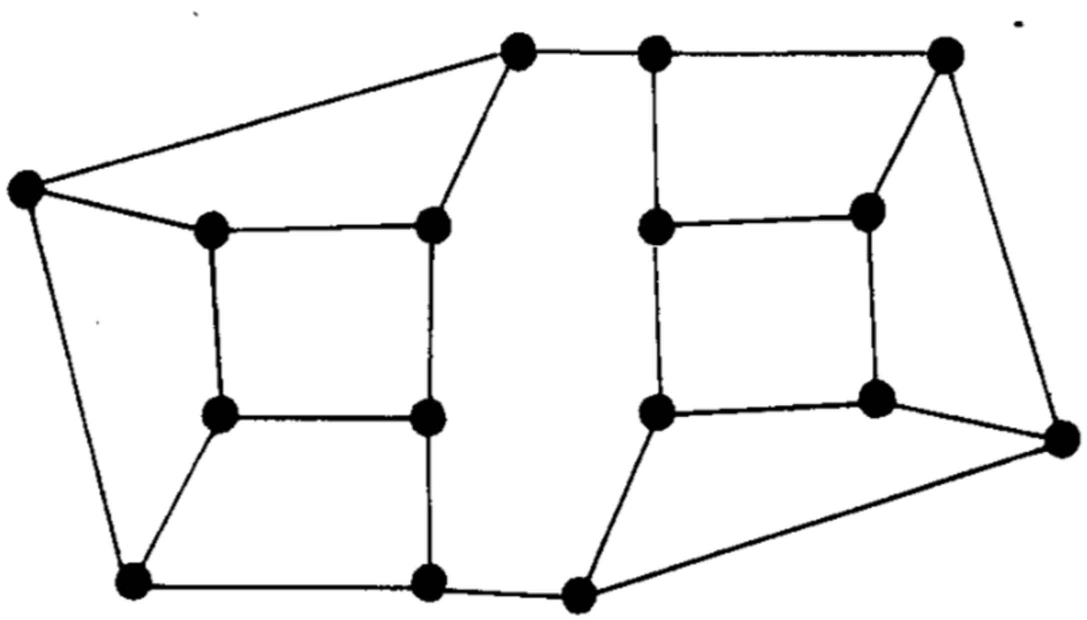
At the beginning of class, the teacher should hand out the Bob Public Key Map and the Bob Private Key Map.  In this exercise, the two Alices send an encrypted message to Bob consisting of a single number (preferably, less than 100).  The two Alices encrypt the message using Bob's Public Key Map.   Once done, Bob decrypts the message from the two Alices using Bob's Private Key Map.  Both Maps have been included in the resource section.

There are two Alices because there is a substantive (though not insurmountable) computational load. The two Alices need to create the encrypted message and double check that the associated computation is accurate.

To start the exercise, Bob creates his Bob's Private Key Map from the Blank Private Key Map (image immediately below).  Bob's Private Key Map consists of

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

16 specially connected dots. The dots are connected such that every dot is connected to exactly three other dots. This means that every dot in Bob's Private Key Map is part of a four-dot group. For purposes of this exercise, it is important that the students use this dot map and not a map of their own creation.



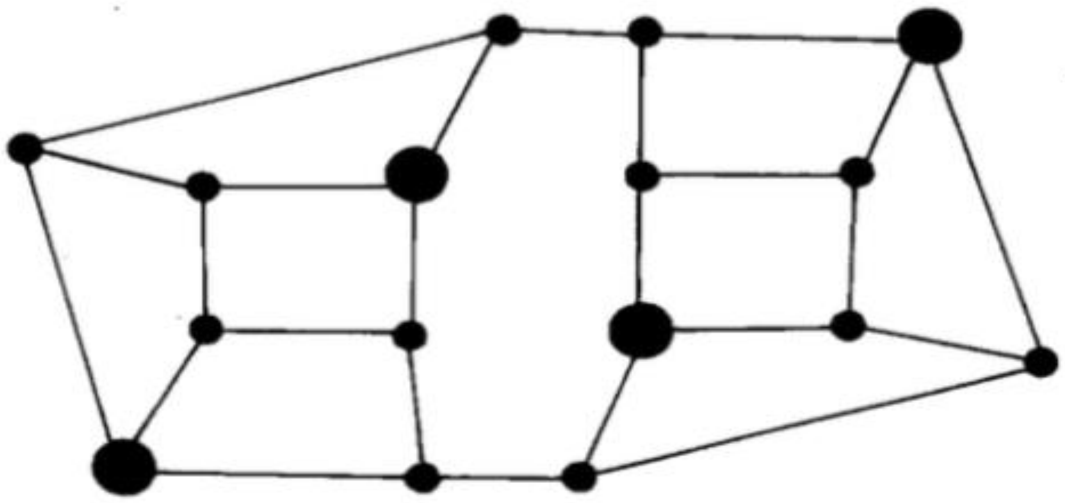
To create the Bob's Private Key Map from the Blank Private Key Map, Bob needs to select four unique dots such that all 16 dots are connected to one of those four selected dots. In this way, the three dots connected to each selected dot are a) inclusive of all 16 dots and b) subdivide the Bob's Private Key Map into four unique subsections. Students may find that it takes a few iterations to select four dots which meet these two criteria. Once done, these 16, dots with the four selected dots, constitute the Bob's Private Key Map. Bob should keep the Bob's Private Key Map, well, private, and not share it with either of the two Alices. Bob should identify these four special dots by making them bigger than the remaining 12 dots. The secret map below shows four correctly enlarged dots.

The two Alices now need to do some work. It is their goal to pass a secret integer to Bob using his Bob's Public Key. First, the Alices need to pick an integer (preferably, less than 100). This number will be the secret message

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

which the two Alices are going to secretly pass to Bob. Next, the Alices need to fill in all 16 dots with a number, such that the sum of the 16 numbers equals the secret number which they want to pass to Bob. They can use negative numbers if they wish.

Now, for the computational bit; the two Alices need to visit all 16 dots of the map they just created. For each of the 16 dots, the two Alices need to sum that dot and the three dots connected to that dot. When finished, each of the 16 dots will have a number next to it which is the calculated sum of four dots. The last thing the two Alices need to do is to erase the original numbers which add to their secret integer message. If they left those numbers on the map, a snooping Eve would be able to add those 16 original numbers together and thereby decrypt the secret message from the two Alices. This grid of 16 dots with its 16 associated sums constitutes Bob's Public Key.



Once done, Bob decrypts the adjusted Bob's Public Key Map by adding together the sums (received from the Alices) listed at the four secret dots.  Magically, these four dots should sum to the encrypted integer message delivered from the two Alices. Once Bob has a number, Bob should reveal the secret integer message from the Alices.  There is no way that Eve can come up with this number without knowing Bob's the four secret enlarged dots contained in Bob's Private Key Map.

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.

There are several important learning elements in this exercise. Firstly, students benefit from a deeper dive into the mechanics of Public Key Exchange. Secondly, students gain an appreciation of the complexity and beauty of one-way functions as they are applied in Public Key Cryptography. Thirdly, students have a chance to ponder the limitations of these learning exercises relative to the actual complexity and interconnectedness of genuine Public Key Encryption methods. This exercise is a metaphor for the actual process. Understanding the limitations of this metaphor will help students round out their understanding of Public Key Cryptography.

## ACKNOWLEDGEMENTS

**Resources:**

Khan Academy on **Encryption and public keys**

Khan Academy on **Diffie-Hellman Key Exchange**

Khan Academy on Asymmetric Encryption

*The Code Book* by Simon Singh, Chapter 8: *Alice and Bob Go Public.* Pages 243-279.

*Nine Algorithms That Changed the Future: The Ingenious Ideas That Drive Today's Computers*, Chapter 4: *Public Key Cryptography: Sending Secrets on a* Postcard. Pages 38-51.

Tim Bell, Ian H. Witten, Mike Fellows: *Computer Science Unplugged – An enrichment and extension programme for primary-aged children*, 2005.

Activity 18 **Kid krypto—*Public-key encryption***[PDF]
*Blown to Bits: Your Life, Liberty, and Happiness After the Digital Explosion* by Hal Abelson, Ken Ledeen, and Harry Lewis. Chapter 5: *Secret Bits How Codes Became Unbreakable.* Pages 161-221

This document is licensed with a Creative Commons Attribution 4.0 International License ©2017 Catalyzing Computing and Cybersecurity in Community Colleges (C5). This material is based upon work supported by the **National Science Foundation** under Grant No.1548315.