

> Project Summary and Outcomes

In this project, students will explore the class of data encryption techniques known as substitution ciphers. They will research the history of these ciphers, from their use by Julius Caesar to the Germans' WWII Enigma machine, and they will also explore some cryptanalysis techniques for breaking these ciphers.

Students will implement one substitution cipher in specific, the Vigenère cipher, using it to encode and decode text files.

> Student Assessment (see rubric)

- Research the history of substitution ciphers
- Read from a file
- Write to a file
- Encrypt using the Vigenère cipher
- Decrypt using the Vigenère cipher

> Prerequisite Knowledge and Skills

Java:

- File IO
- Text and non-text files
- Unicode and ASCII character sets
- String methods
- Arrays
- Modulo arithmetic

> This Project Targets the Following:

Pre-Java	Java Programming
<input type="checkbox"/> Hardware Basics	<input type="checkbox"/> Applet Programming
<input checked="" type="checkbox"/> Software Basics	<input checked="" type="checkbox"/> Application Programming
<input type="checkbox"/> Networks and Servers	<input type="checkbox"/> Class Programming
<input type="checkbox"/> HTML	<input checked="" type="checkbox"/> Method Programming
<input type="checkbox"/> Action Scripting	
<input type="checkbox"/> Java Scripting	

> Project - Framing Questions

Essential Questions

- What is encryption?
- How is encryption used in society?
- What is the role of encryption in providing security and privacy?

Activity Questions

- Why is data encryption important?
- What are cryptography and cryptanalysis?
- What are substitution ciphers and frequency analysis?

Sample Content Questions

- When are characters represented in ASCII and when are they represented in Unicode?
- How do you find the encoding or decoding of a character based on a character of the key?
- How do you find the correct character of the key to use?
- How will you perform file IO?
- How will you handle IO exceptions?

> Targeted Content Standards, Benchmarks, or State Frameworks

Massachusetts Curriculum Frameworks:

Patterns, Relations, and Algebra:

- 12.P.8 – Solve a variety of equations and inequalities using algebraic, graphical, and numerical methods, including the quadratic formula; use technology where appropriate. Include polynomial, exponential, logarithmic, and trigonometric functions; expressions involving absolute values; trigonometric relations; and simple rational expressions.
- 12.P.1 – Describe, complete, extend, analyze, generalize, and create a wide variety of patterns.

School-to-Career Competencies:

- Communication and literacy
 - Speaking, listening, reading, and writing
- Organizing and analyzing information
 - Collecting and organizing information
 - Research and analysis
 - Quantitative analysis and mathematics
- Problem solving
 - Identifying and solving problems
- Using technology
 - Using work tools and office equipment
 - Computer operation
- Completing entire activities
 - Initiating and completing projects
 - Time management

> Materials and Resources Required for Project

Equipment	<ul style="list-style-type: none"> • A computer with Java
Consumable Supplies	<ul style="list-style-type: none"> • Disks or CDs • Graph paper
Textbooks/Lesson Guides	<ul style="list-style-type: none"> • <i>Java Software Solutions: Foundations of Program Design</i> by John Lewis and William Loftus. Addison-Wesley, 2003, or any Java classroom text. This project requires only basic Java. Students should be comfortable with modulo arithmetic.
Internet Resources	<ul style="list-style-type: none"> • http://www.codesandciphers.org.uk/enigma/ • http://starbase.trincoll.edu/~crypto/ • http://en.wikipedia.org/wiki/Enigma_machine • http://www.enigma-replica.com/ • http://www.iwm.org.uk/online/enigma/index.htm • http://en.wikipedia.org/wiki/Vigen%E8re_cipher • http://en.wikipedia.org/wiki/Frequency_analysis • http://www.geometer.org/mathcircles/

> Activity/Lesson Plan Outline

Overview: This project will introduce data encryption techniques and will explore a class of encryption algorithms known as substitution ciphers and to gain some familiarity with using these algorithms to encrypt and decrypt text files.

Purpose: To learn about a revolutionary development in modern encryption, running algorithms and doing simple cryptanalysis in the process. To apply Java concepts, such as file IO, exceptions, looping, and modular arithmetic while working on an interesting problem. To write Java code that, given a key, can read a text file and write an encrypted file using a variation of the Vigenère cipher.

Assessment: Students are evaluated on their participation in this exercise as well as the degree to which they learn about algorithms and their role in modern cryptosystems. Students are assessed on the quality and success of their code and documentation as per the rubric.

Note: This activity plan outlines encryption and decryption techniques. For more information regarding these techniques and concepts, see the previous references.

Encryption algorithms take a plaintext input plus an encryption key and produce an encrypted ciphertext. The aim of an encryption algorithm is to prevent an adversary from reading the plaintext, while allowing selected receivers, who have the required decryption key, to read the file. The study of encryption techniques is called cryptography. The adversarial process of studying how to break encryption techniques is called cryptanalysis.

Used as a fundamental tool in modern, networked security systems, encryption allows provides the following:

Privacy – ensuring that only authorized users can access data

Integrity – ensuring that data received is identical to data sent

Authentication – ensuring that users are who they claim to be

Authorization – allowing users to access certain resources

Overview: In this activity, students will research the role of encryption in modern networked systems.

There are two main types of encryption: asymmetric (also called public-key encryption) and symmetric. When the encryption key is identical to the decryption key, the encryption algorithm is considered symmetric. Otherwise, the decryption key is considered asymmetric. Modern symmetric encryption algorithms include DES (Data Encryption Standard), 3DES (“triple DES”), and Blowfish. Asymmetric algorithms include RSA (named using the initials of its creators, Rivest, Shamir, and Adleman) and DSA (Digital Signature Algorithm). Symmetric algorithms are more widely used, but asymmetric algorithms have important applications, such as for network authentication and digital signatures.

Discuss why encryption is used on the Internet. What types of things are encrypted on the Internet (for example, passwords, credit card information, and corporate secrets)? What are the differences between passwords, encryption, and security? A good resource for this discussion is (<http://www.securityfocus.com/infocus/1181>).

Activity 2: Understanding the RSA Algorithm

In this activity, students will research the RSA algorithm, the first well-known and most commonly used asymmetric encryption algorithm. The mathematics of the RSA algorithm is difficult but accessible to advanced students. High school students can participate in what are commonly called Math circles or organizations of students interested in advanced mathematical topics (see: <http://www.geometer.org/mathcircles/>). The math circles umbrella organization has reviewed RSA. (For notes, see <http://www.geometer.org/mathcircles/RSA.pdf>.)

Create posters or put together presentations to explain how to do the following:

Execute RSA for small values

Use public keys to encrypt messages readable only by a recipient

Use private keys to prove a message came from a sender

To prepare for the programming project, focus on substitution ciphers, in which each character of plaintext is substituted with another character to generate the ciphertext. Substitution ciphers are symmetric. The decryption algorithm simply performs the substitutions backward.

The simplest substitution cipher is known as the Caesar cipher, attributed to Julius Caesar. Caesar used the cipher to send an encoded message to Marcus Cicero. The message was secure not because the cipher was difficult to break, but because most of Caesar's enemies were illiterate. While this cipher was named for Caesar, it was probably created by someone else.

The Caesar cipher takes each letter of the input text and shifts it by a fixed number of letters. A key for a particular Caesar cipher is described in the following table. For simplicity, use only lower case letters.

R	0	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	
a											0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
k																										
P	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
l																										

and the associated decryption algorithm is

$$p = D_k(c) = (c - k) \bmod 26$$

Cryptanalysis of text encoded with a Caesar cipher is straightforward. It involves trying all 25 keys.

Caesar ciphers are examples of monoalphabetic substitution ciphers, in that each plaintext character is substituted for by a single ciphertext character. A more secure monoalphabetic substitution cipher creates a random permutation of the alphabet to define the key. Here is an example.

R a n k	0	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	
P l a i n	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
C i p h e r	h	f	p	u	i	n	b	v	e	r	m	g	y	s	t	a	l	o	j	x	k	z	c	w	q	d

Note: There are $26! - 1$ (approximately 4×10^{26}) possible keys and it becomes practically impossible to try each one. However, we can use frequency analysis, which takes into account the expected number of occurrences of each letter, digrams (two-letter combinations), trigrams (three-letter combinations), and small words to help break the code.

Activity 3: Code Breaking

Overview: In this activity, students will encrypt short poems by hand and then have a competition to see who can break each other's encryptions.

Divide into teams of three for a code-breaking competition. Take a short poem and encrypt it using a monoalphabetic substitution cipher of your own choosing. The competition is to see which team can decode the most poems within a given time frame.. The discussion found on <http://www.math.cudenver.edu/~wcherowi/courses/m5410/exsubcip.html> is a very helpful resource.

A more complex extension of monoalphabetic substitution ciphers are polyalphabetic substitution ciphers, through which a single plaintext character may be replaced by one of a number of ciphertext characters. Polyalphabetic substitution ciphers make frequency analysis more difficult.

Next, examine and implement a polyalphabetic substitution cipher known as the Vigenère cipher. Invented in 1553 by Giovan Batista Belaso, this technique was mistakenly attributed to Blaise de Vigenère, another cryptographer of the same era who further developed the technique.

This cipher uses a key that is a string of characters to select a Caesar cipher, which is then used to

encrypt (or decrypt) the plaintext characters. The first plaintext character is encrypted (decrypted) using the first character of the key. Each subsequent plaintext character is encrypted (decrypted) using subsequent characters of the key. When you get to the end of the key, you continue at the beginning of the key.

The following example of the Vigenère cipher uses the key “lincoln.”

Plain	f o u r s c o r e a n d s e v e n y e a r s
Key	l i n c o l n l i n c o l n l i n c o l n l
Cipher	q w g t g n b c m n p r d r g m a a s l e d

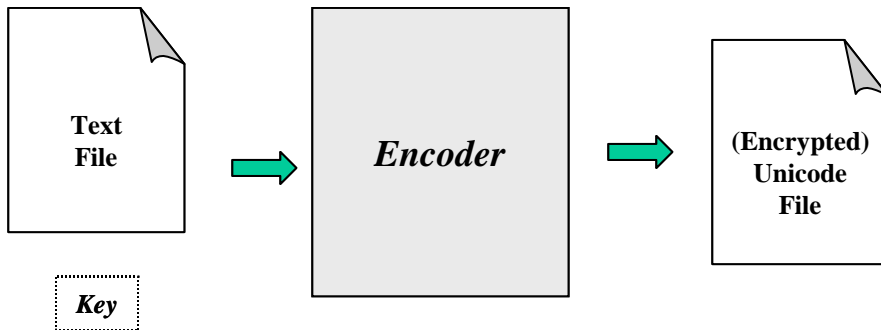
The Vigenère cipher can still be broken using frequency analysis, but it is harder to do than with a monoalphabetic substitution cipher, since each character maps to one of many characters.

Activity 4: Encrypting with the Vigenère Cipher

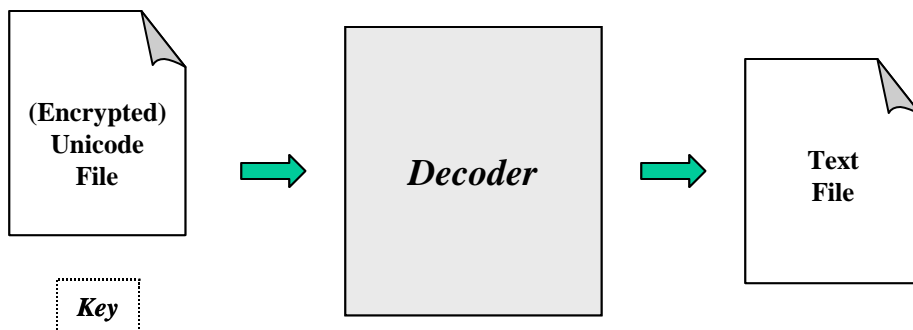
Overview: In this activity, students will implement a particular encryption algorithm, the Vigenère cipher.

To perform this activity successfully, review the difference between the way that most operating systems represent characters and the way that Java represents characters. Operating systems such as Windows typically represent characters using the ASCII standard. ASCII uses 8 bits to represent a character, which means there are at most 2^8 , or 256, possible ASCII characters. Java uses the Unicode standard to represent characters. Unicode uses 16 bits to represent a character, which means there are at most 2^{16} , or 65,536, possible Unicode characters. Having this capacity, Unicode can represent characters from many world languages.

The term textfile refers to a file of text characters organized according to the operating system's character representation method. We encrypt a textfile by reading one character at a time, encrypting the character using the Vigenère cipher, and writing the results a character at a time to a non-text file of Unicode characters. The following diagram shows the encryption process.



The following diagram shows the decryption process.



Use the following classes from the `java.io` package:

- `FileReader` – to read a textfile
- `FileWriter` – to write a textfile
- `FileInputStream` – to read a non-textfile
- `FileOutputStream` – to write a non-textfile

Unlike the previous examples, you will use a variation of the Vigenère cipher that encrypts all of the characters from the textfile, including spaces, newlines, and tabs. Therefore, you should not use the `BufferedReader` class, as that removes newline characters.

The encryption algorithm uses the Unicode representation of both the plaintext character and key character to find the cipher character. The encryption of Unicode plaintext character p using key character k is

$$c = (p+k) \bmod 65,536$$

or in Java

```
char c = (p+k)%(Character.MAX_VALUE+1);
```

Similarly, the decryption of Unicode cipher character c using key character k is

$$p = (c - k) \bmod 65,536$$

or in Java

```
char p = (c-k)%(Character.MAX_VALUE+1);
```

To test your implementation, encrypt a textfile, decrypt the resulting file, and test to see if the two files are identical. Also, it is a good idea to use a text editor to look at the encrypted file and try to explain what you find.

> Pacing/Timeline

- Activity 1: 5 hours
- Activity 2: 15 hours
- Activity 3: 5 hours
- Activity 4: 10 hours

> Teacher Reflection (For example, what worked well in this project? What would you change if you were to teach it again?)

- Did students seem engaged in the topic?
- Which activities stimulated their thinking about encryption?
- Did students make the cross-curricular connections to math and history?

> Encryption - Rubric

Attribute	Needs Improvement	Proficient	Advanced	Maximum Pts.
-----------	-------------------	------------	----------	--------------

Activity 2: Understand the RSA Algorithm

<i>Understands Algorithm</i>	Shows no understanding of the algorithm	Understands the basic structure of the algorithm, but not the details	Demonstrates a thorough understanding of the details of the algorithm	
Creates Poster	Creates a poster that is unreadable or that does not show RSA or an application	Creates a poster that demonstrates some aspect of the RSA algorithm, but not clearly	Creates a poster that a non-mathematician could understand that demonstrates some aspect of RSA	

Activity 3: Break the Code

<i>Encrypts a Poem</i>	Does not encrypt a poem properly	Encrypts properly	N/A	
<i>Decrypts Other Teams' Poems</i>	Does not succeed in decrypting others' poems	Successfully decrypts some poems	N/A	

Activity 4 Encrypt with the Vigenere Cipher

Code Style	Offers no comments. Uses poor formatting.	Offers some comments and uses adequate formatting	Comments in all appropriate locations.	
Reads From a File	Cannot write to a file or uses incorrect java.io class	Successfully reads from files	N/A	
<i>Writes to a File</i>	Cannot write to a file or uses incorrect java.io class	Successfully writes to files	N/A	
<i>Encrypts Using the Vigenère Cipher</i>	Does not encrypt or does not use the proper algorithm	Encrypts properly	Encrypts using the Vigenère cipher	

<i>Decrypts Using the Vigenère Cipher</i>	Does not decrypt or does not use the proper algorithm	Decrypts properly	Decrypts using the Vigenère cipher
---	---	-------------------	------------------------------------