The **title page** should include (either 1, 2 or 3 should ALL have this)

* the title of the paper or project
* group member name(s) and email address(es)
* course name, number, and instructor's name
* a maximum 300-word abstract
* three to five keywords.

1. For **implementation** oriented project, the main content should include the following:

* A summary of the system, algorithm, or protocol you want to implement.
* Justification about why you want to implement it.
* The architecture of your implementation.
* Your implementation plan, including milestones (or check points) and deliverables.
* Your plan for demonstrating your implementation. You may have to write additional programs to demonstrate implementation. In this case, you should explicitly identify what these additional programs are.
* A reference list with at least 5 references.
* 10 minute presentation to the class

Knudsen, J., & EBSCO Publishing (Firm). (1998). *Java Cryptography* (1st;1; ed.). Sebastopol, Calif: O'Reilly.

Maia, L. A. , & Correia, M. E. (2012). Java JCA/JCE programming in android with SD smart cards . Paper presented at the 1-6.

Buchmann, J., May, A., & Vollmer, U. (2006). *Perspectives for cryptographic long-term security.* New York: ACM. doi: 10.1145/1151030.1151055

Kruh, L. (1999). Java cryptography. *Cryptologia,* 23(1), 44.

Koved, L., Nadalin, A., Nagaratnam, N., Pistoia, M., & Shrader, T. (2001). Security challenges for enterprise java in an e-business environment. *IBM Systems Journal,* 40(1), 130-152. doi:10.1147lsj.401.0130

Mishev, A., Bothe, K., Budimac, Z., Jurca, I., & lvanovic, M. (2013). Java across different curricula, courses and countries using a common pool of teaching material. *Informatics in Education* - *an International Journal,* 12(Vol12\_2), 153-179.

Rabah, K. (2006). Implementing secure RSA cryptosystems using your own cryptographic JCE provider. *Journal of Applied Sciences,* 6(3), 482-510. doi:10.3923ljas.2006.482.510

http://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html

https://docs.oracle.com/javase/tutorial/security/

<https://docs.oracle.com/javase/8/docs/technotes/guides/security/overview/jsoverview.html>

<http://www.javaworld.com/article/2076989/core-java/java-s-security-architecture.html>

<http://www.infoworld.com/article/2608786/application-security/application-security-how-to-solve-java-s-security-problem.html>

<http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html>

<http://www.cert.org/secure-coding/publications/books/java-coding-guidelines.cfm>

<https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices>

https://www.sans.org/event/network-security-2013/course/secure-coding-java-jee-developing-defensible-applications

<http://ptgmedia.pearsoncmg.com/images/9780321933157/samplepages/9780321933157_Long_WebSample.pdf>

<http://www.sei.cmu.edu/news/article.cfm?assetid=77817>

<http://www.ibm.com/support/knowledgecenter/ssw_ibm_i_71/rzaj4/rzaj45bejavasecurity.htm>

<http://www.javaworld.com/article/2076292/core-java/secure-a-web-application--java-style.html>

<https://www.sans.org/course/secure-coding-java-jee-developing-defensible-applications>

<https://www.javacodegeeks.com/2012/12/test-using-cryptography-in-java-applications.html>

<http://www.cubrid.org/blog/dev-platform/understanding-encryption-security-through-java-cryptography-architecture/>

<http://www.ibm.com/developerworks/java/tutorials/j-sec2/j-sec2.html>

<http://www.ibm.com/developerworks/java/tutorials/j-sec1/j-sec1.html>

<http://www.ibm.com/developerworks/java/tutorials/j-sec1/j-sec1.html>

<http://www.ibm.com/developerworks/java/tutorials/j-sec1/j-sec1-pdf.pdf>

<http://people.ucalgary.ca/~salindne/418/Files/JCA.pdf>

<https://www.cs.umd.edu/~jkatz/security/f09/lectures/JCA.pdf>

<http://cs.lmu.edu/~ray/notes/javanetexamples/>

<http://www.javaworld.com/article/2076087/java-security/java-security-evolution-and-concepts--part-1--security-nuts-and-bolts.html?page=2>

Add cryptography to a server and client program that accesses a password file.

I would like to use: The Java Cryptographic Extension (available since JDK 1.4) API to implement encryption and Java Cryptography Architecture Reference Guide (JCA Guide) - <http://docs.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html#Introduction>

# Summary

The first part of the project will consist of an overview of what has been discovered in the area of probabilistic cryptography and will conclude with my impression of what potential the area holds. The first goal of the project is to do a study of what different types of probabilistic cryptography are out there. For each type I will see how much research has been put into them and how secure they seem to be. The paper will also need to check how efficient the encryption and decryption process is. All of this should show if the different encryption schemes have potential to be used in applications and whether probabilistic encryption as a whole has more potential to be studied. I have decided to study 3 different schemes: Goldwasser-Micali, Blum-Goldwasser, and Paillier’s scheme. There are others out there but these seem to be the main ones that are mentioned. The others are based on altering these schemes.

In the second part of my project I will create my own programs that will use one or more of the probabilistic encryption schemes. The plan is to test the timing of several versions of Paillier’s scheme versus RSA. It will be interesting to compare the schemes in speed and run some practical tests on them. By creating my own program of the schemes it should allow plenty of opportunities to test every aspect of it. If time permits, other probabilistic schemes could be implemented for more testing.

# Overview

Goldwasser and Micali were the first people to come up with a workable scheme for probabilistic cryptography. The main benefit of probabilistic cryptography is that you can encrypt the exact same plaintext into different ciphertexts but get the same plaintext back when decrypted. The Goldwasser-Micali probabilistic scheme [2] security is based on the quadratic residuosity problem and is considered very slow. The Blum-Goldwasser scheme [2] is much faster scheme, being comparable to an RSA scheme in speed. Its security is based on integer factorization but has been found to be susceptible to chosen- ciphertext attacks. It will be interesting to study how big of a threat these people think these attacks pose.

There is another scheme called Paillier’s probabilistic scheme that has been developed more recently. It is based on the composite residuosity class problem. This scheme seems to have a lot of potential as it seems secure and is computationally comparable to RSA [3]. There is a lot of different research on this type of scheme and there seems to be some practical applications for using it in electronic voting. This will be an interesting topic to look into with all the recent attention given to electronic voting.

The Paillier scheme actually has several variations and other people have changed it to create other versions. The following is the first Paillier scheme described in his paper [4]. It gives you the general idea of how encryption based on composite residuosity works. First you set n = pq with p and q being large primes as you would in RSA. Then you select a random base *g*. This can be found by checking if g works in *gcd*(*L*(*gλ) mod n*2*),n)*

*= 1*. The notation *λ* is equivalent to *λ*(*n*) and is called the Carmichael’s function. This can be calculated as *λ*(*n*) = *lcm*(*p* − 1*, q* − 1). We also define *L*(*U* ) = *U* −1 . The encryption

*n*

is done by the following:

*n* = modulus

*m* = the plaintext where *m < n*

*r* = a random number where *r < n*

*c* = ciphertext which is generated as follows *c* = *gm* ∗ *rn* mod *n*2.

The process of decryption uses the same mathematics described above and can be done as follows:

*c* = ciphertext where *c < n*2

λ 2

*m* = plaintext and can be found as follows *m* = *L*(*c modn* )

*L*(*g*λ*modn*2 )

mod *n*.

One of the main properties of Paillier’s scheme is that it is homomorphic. Homomor- phic encryption is just like other semantically secure public-key encryption, except it has has the following property:

*E*(*A*) ∗ *E*(*B*) = *E*(*A* ∗ *B*)

where \* can be addition or multiplication or any other binary operator (but not more than one)[8]. This means that after plaintext *A* and *B* have been encrypted, we can still perform operations on them. A 3rd party can add or multiply the encrypted values together without knowing the plaintext. The value can later be decrypted by another person to find the result. The 3rd party never needs to know the encrypted values or result to do the operation. This property has a lot of promise and has been applied in some e-voting schemes as well as other applications. The Paillier scheme is additively homomorphic.

# Contents of Project

The project will consist of two different parts. First, there will be an overview of prob- abilistic cryptography, explaining what has been done to study it and what research is currently out there. Secondly, some different types of probabilistic cryptography will be used to create my own software that will be used to study the schemes in practical circumstances. The project will be split so half the time is spent on both on these parts.

## Overview of the Theory of Probabilistic Cryptography

The goal in the first part will be to explain as completely as possible what probabilistic cryptography is. The explanation will include what the key differences are between this and other deterministic schemes. One key concept to understand is that probabilistic encryption is semantically secure. This implies that no information about the plaintext can be learned from the ciphertext. This can be of great importance in making sure a scheme is not insecure when sending the same message multiple times. Most schemes, like RSA, will send the same thing over and over. This can be a clue to someone spying on the encrypted message to tell what is being sent.

It will look into all of the underlying theory that allows these schemes to work and shows how robust the theory is. All of the probabilistic schemes seem to have some similarities in encrypting and decrypting. The paper will explain what these are and what the differences are. It will include an explanation on how efficient the algorithms are and about how many operations are necessary for a typical encryption and decryption. It will be related to the speed of more popular private key schemes in use today.

The paper will look into the practical application of the Paillier scheme. Currently it seems to be the only scheme that is being studied for this purpose. The big advantage that Paillier scheme has is that it is homomorphic. This means that you can perform certain mathematical operations on the ciphertext and they will be preserved when it is decrypted. According to Paillier there are several practical areas that area being studied beause of this property, including: voting protocols, threshold cryptosystems, watermarking, and secret sharing schemes [4]. It will be good to look into these and any others uses that can be found. I would also like to see if there are any other applications that would use Paillier’s properities well. The applications could be tested in the implementation if something is found that looks interesting.

## Study of Experiments of Paillier Scheme

The second part of the project will consist of a program that will use several probabilistic encryption schemes. The program will use the basic Paillier scheme described in his paper. The program will also implement another version of the Paillier scheme that was developed by D˚amgard and Jurik. This scheme is like Paillier but is done using modulo *ns*+1 for any *s >*= 1 instead of *n*2 in Paillier. We will call this scheme D˚amgard-Jurik [3]. Once I have these schemes implemented it will be interested to vary the numbers used

in the schemes to see how they perform for both securty and speed. One thing that I will test is the ”base” *g* from the Paillier scheme. In both the Paillier and D˚amgard-Jurik papers they usually use the number *n* + 1 as the base *g*. It will be interesting to see if this works best in practicality. There will also be other numbers that can be altered and I may even try my own version of the schemes. The goal will be to find the best values that perform securly.

These seem to be the schemes that are being studied the most currently. There aren’t any recent papers on the other schemes, Goldwasser-Micali and Blum-Goldwasser, since they are all from the 1980’s. All the papers on Paillier’s scheme and other versions of it are within about six years, although some of the ideas seem to have been around longer. This seems to suggest that this scheme has much more promise and can be used in practical applications. The application of using Paillier for electronic voting is another bonus. I haven’t found any applications that have used the other schemes or any mention that it might happen.

The plan is to implement an RSA scheme as well to use for comparison. Since RSA is a well established public-key cryptosystem it will be a good standard to compare the less established Paillier. The main thing that will be tested is the speed of the two schemes to see how they compare. It will be interesting to compare the encryption and decryption speeds and see if Paillier can perform at a similar level.

# Design of Paillier Experiments

The goal of this program is to create software to test the Paillier schemes and see how well they perform. While all the theory and proofs are good to study, we must realize that the implementation must perform or the scheme will never be used. Using this software will give some practical proof of how well they can compete with what is currently being used. The design will be made so I can alter as many variables in the schemes as possible to see which ones outperform the others. The following is the current design of the software I will use to reach these goals.

## Overview of Design

* + - Create a peer to peer type of cryptographic application.
    - Front end application will be simple implemenation of cryptography classes.
    - Use RMI to comunicate messages between peers.
    - Can send messages as plaintext or encrypted by any scheme implemented.
    - Use the Java language along with Java Cryptographic Extension (JCE).
      * Cipher Class
      * CipherSpi Class
    - Create Paillier scheme as provider to implement in application.
    - Create D˚amgard-Jurik (another version of Paillier) scheme as provider to implement in application
    - Use a current RSA scheme already provided with RSA in application.
    - Create time tracking methods to compare timings.
    - Create simple message cracking functions to see if I can crack simple implemenations.
    - Test all schemes implemented for timing and security.

## Detail of Design

The plan is to create a peer to peer type of application that allows a user to set up a private-key encryption scheme and then pass an encrypted message from from one instance to another. The instances of the program can be set up on different machines and could possibly involve more than 2 instances at once. Either peer can send an encrypted message to the other peer. The program will be able to use either of the Paillier schemes or RSA to send the message and the user will not be able to see any real difference. The user interface and message sending will be very simple and most of the work will be done in

the specific methods in the back end. These cryptography methods will be created in a cryptographic service provider, which can be simply called a provider [7]. A provider is the term used in the Java Cryptography Extension (JCE) to describe someone that provides implementations for cryptographic functions.

I am planning on using the Java language to write the program. Java seems to be an obvious choice since most people are using it and the support it gives to specific applications with its API’s. This is especially helpful because of the Java Cryptogra- phy Extension (JCE) that is provided in the JDK. There are also specific providers that supports the RSA scheme. The classes that are given by the provider will be used to encrypt and decrypt with RSA. These should allow a realistic look at how a well estab- lished encryption scheme performs. If this is successful then it will allow more time to be spent on probabilistic schemes. If time permits I would also like to try implementing the Blum-Goldwasser scheme in my application as well.

The concept of a provider is essential in the use of the JCE. The providers have the job of implementing the complex cryptographic algorithms and making them available to other Java developers. The normal mainstream developer looking for cryptographic secu- rity will have very little understanding of the how the provider implements the algorithms. They should have a general knowledge of cryptography and understand how to bring a provider into their development and use the implementations. There are many providers around, available as open-source or for purchase. Sun includes a provider in the JCE that implements many basic cryptography functions. Many providers can be installed in Java that contain overlapping algorithms. The JCE has a group of rules based on the setup of the providers that chooses the algorithm. My provider will work in the same way to allow the implementation to be plugged into other applications.

The JCE is actually an extension of the Java Cryptography Architecture (JCA). They both work together to give you a complete set of cryptographic functions. The only reason they were split into separate libraries is because of United States export laws, which prevented the export of certain types of cryptography. These laws have been changed and the split is no longer necessary, but still exist in the current Java version. Each of the different categories of cryptography components are put into different classes, which are referred to as engines. Any of these engines can be invoked in the code to create an instance of the engine. The engines all use the factory pattern to return the instance. This pattern was found to be the most beneficial for allowing a developer independence to choose to the best algorithm from a provider and then later changing it if he finds a better one. The Service Provider Interface (SPI), which all providers must use, specifies that the engines be created using this pattern. To access a certain cryptographic function the developer just has to use the getInstatnce(”algorithm”) method on the engine where the algorithm is classified. The ”algorithm” parameter is the specific name of the algorithm the developer is looking for. The JCE will look for this algorithm in the providers that have been installed and return an instance (if one exists) according to the rules in the JCE. The developer can also specify a certain provider to ensure they receive the specific algorithm they want [6].

The two main classes that will be needed in the JCE are Cipher and CipherSpi

/citeWeiss. The CipherSpi class is an abstract class that allows you to create your own cipher. Implementing this class when you create your own cipher will force you to use all the same procedures that have been predefined. This allows the user to know how to use your implementation of the cipher. The Cipher class is actually an implementation of CipherSpi that gives you access to all the cryptography methods by different providers. Currently Cipher supports implementations of DES, AES, RSA, Blowfish /citeKnud along with some others that come with the JCE. You can get more schemes from other third party vendors. For my purposes I am going to create the methods for my Paillier schemes as a provider and load it into the JCE. This will allow me to access my schemes through the Cipher class and use it for encryption.

A main part of the project will be understanding the JCE and how to use it effectively to implement an encryption scheme. I want to understand how to make a front end application that uses encryption from a provider. This will allow the application to change the back end cryptography by just changing to other classes that implement the same interface. I want to be able to create a provider with the cryptography classes that implement the interface correctly. This will allow me to create cryptography schemes that can effectively be used in an application already set up to use the interface.

The design of the system will involve RMI. The plan is to allow communication between two separate instances of the program by using RMI to send the information across the network. RMI will allow 2 instances of the program to be set up on different machines and perform simple communication.

The actual capabilities of the front-end of the system will be simple but will accomplish the testing needed. The RMI will allow a message object to be sent across the network to another instance of the program. The message can be sent as not encrypted or encrypted by any implemented scheme. The message object will tell what encryption was used if any. The message will be decrypted when it is received by the accepting instance. The message will be able to be viewed either encrypted or decrypted by either the sender or receiver. There will also be time tracking functionality that will allow me to do some of my testing.

It would be good to create some functions that allow me to try to crack the message code. These functions would be based on very simplistic, brute force methods of finding the key. They would mainly be testing the encryption with an artificially small key. A normal size key would nearly impossible to break with my limited computing resources. The message cracking would be done to compare the robustness different schemes imple- mented.

# Deliverables

Final Report

1. Summary of Previous Research
2. Mathematical Background
3. Explanation of Algorithms
4. Results of Experiments
   1. Design of experimental system
   2. Use of system
   3. In depth explanation of all tests
5. Summary of All Work
6. Promising Areas for Reasearch Java Code
7. Provider Classes
8. Front-end Application Java Executable

# Proposed Timeline

|  |  |
| --- | --- |
| Task  Proposal Completed | Target Date Completed Date  5/28/05 |
| Proposal Approved | 6/1/05 |
| Background Report Completed | 6/15/05 |
| Design and User Tests Completed | 6/29/05 |
| Probabilistic Program Completed | 7/05/05 |
| Testing of Program Completed | 7/12/05 |
| Final Write Up and Summary Complete | 7/19/05 |
| Project Defense | ? |

From: Rochester Institute of Technology

Department of Computer Science

Master’s Project Proposal

A Study of Probabilistic Cryptography

Secure chat room notes:

This project aimed to develop a java-based internet chat program that will allow several parties to securely talk over the internet. This program will have several applicable uses such as allowing several trusted parties to exchange sensitive information over the internet. The program involves a server part and a client part. The following is some specific steps about how you can run these programs:

1. First you run the server program, when it starts, it generate a fresh triple-DES key and begin listening.

1. You start the client-side program. There are two versions, one is an applet that you run in your browser, another is a normal java application that you run in command line. You need to have Sun's Java Cryptography Extension (SunJCE) package installed on your computer and SunJCE installed as a security provider in your java. security file in order to run the program. This is because the program uses symmetric ciphers. The program first go through a secure password login process, with a mechanism as follows: it first ask you for your login name and password, and generates a fresh time-stamp and random number.Itthen creates a message digest of the four parameters (login, password, time-stamp and random number). It sends the server the login, time-stamp, random number and the message digest in the clear. The server then use the login to look through its database to find the password, and make another message digest using the received login, time-stamp, random number and the looked-up password. The server compare the two message digests, if they are the same, the client has successfully logged in. In this way the client doesn't send the password in clear form over the internet, which will prevent evansdropping. Also this login is session specific because of the time-stamp and random number, which can prevent a "replay" attack.
2. After successful login the server send the triple-DES key it generated, password encrypted using the client's password, to the client. Since every client will receive the same key, they will communicate using that key. More specifically, when a client types some chat message, it will be encrypted using the triple­ DES key and send to the server, the server then broadcast the message to every client current logged in without decrypting it, and every client receiving that message will decrypt it using the same key.

**Notes from:**

., K. R. (2006). Implementing secure RSA cryptosystems using your own cryptographic JCE provider. *Journal of Applied Sciences,* 6(3), 482-510. doi:10.3923ljas.2006.482.510

# Implementing Secure RSA Cryptosystems Using Your Own Cryptographic JCE Provider

Kefa Rabah

**Abstract:** Today and without a doubt, we live in a net-centric world in which new information technologies arrive at lightning speed, allowing us to share information locally to globally faster than ever before. The bad guys are equally out there to eavesdrop on our data transactions over the insecure commlllllcation channels. Cryptography can be used to help us secure our data commlllllcation. In this study we describe a Java implementation of secure RSA cryptosystems using your cryptographic provider, Java Cryptographic Extension (JCE). We then illustrate the use of this implementation in a working prototype. We then explain the nalve procedure for implementation of RSA cryptosystems and signature algorithm followed by the real time Java implementation using Open Source Bouncy Castle JCE provider.

Key words: Cryptography, RSA, DSA, DH, ECC, DLF, ElGama!, public and private keys, digital sigmture, confidentiality, authentication, integrity, Java, JCE, JCA, SSL

# ABSTRACT:

Java is currently one of the most widely used programming languages around the world. With the increase in cyber-attacks, developers will need to protect their code. Different methods will be needed to protect their code. Java has included several Security features in its platform. The security features are intended for developers, system administrators and users. Java Cryptography is part of Java’s built-in security designed to protect programs and data from malicious programs. Java now includes both the Java Cryptography Architecture (JCA) and the Java Cryptography Extension (JCE) packages. Cryptography is the most common and effective way to protect information by encoding it. Cryptography is based on a cipher algorithm which transforms plain text to cipher text. A cipher key(s) is used to encrypt and decrypt the text. This requires a key exchange between the sender and receiver of the data. Even though cryptography is the most common and effective way to protect data, Cryptography can be used incorrectly resulting in unsecure systems and data. Even with Java’s built-in cryptography features, programmers need to know how to use them correctly. In this study, we can see the assessment of Java encryption tactics for satisfied decision of each key and cryptographic scheme. Subsequently this comprehensive survey thrash outs the cutting-edge tendencies and study issues upon cryptographic factors to conclude the upcoming requirements concerning cryptographic key, algorithm constitution and stronger privateness especially in transferring the multimedia understanding.

* an understanding of cryptography, key management, and/or Public Key Infrastructure (PKI)
* Understands security concepts and tooling including but not limited to PKI, Cryptography, along with how those apply to web concepts (HTTP(S), SSL/TLS, certificates etc)
* A desire to learn and work with security components including cryptography, code signing, strong naming and authentication is an asset.
* Solid knowledge & development experiences with PKI, TLS, EC Cryptography
* PKI (Public Key Infrastructure)  
           Secure email  
           TCP/IP, LDAP, SMB  
           Kerberos, NTLM, SSO  
           Cryptography
* Knowledge of security concepts and tooling including but not limited to PKI, Cryptography etc. (preferred).
* Security protocols, cryptography, PKI, SSL/TLS experience

# KEYWORDS:

JAVA, Cryptography, JCA, JCE, Encryption, Key Exchange.

# SUMMARY:

Offering privacy and authenticity remains a valuable purpose for cryptographic protocols, but the area has elevated to encompass many others, including e-voting, digital coins, and at ease auctions. Cryptographic systems tend to involve each algorithm and a secret worth. The key value is referred to as the key. The reason for having a key furthermore to an algorithm is that it is intricate to hold devising new algorithms in an effort to allow reversible scrambling of understanding.

Cryptography additionally depends on encryption strategies equivalent to symmetric and asymmetric to encode the exact text message like plain textual content with the usage of secret code referred to as key. The process of encoding or encrypting the undeniable textual content is referred as enciphering or encryption and the vise versed system is referred to as decoding or decryption. Symmetric encryption requires a single shared secret code referred to as confidential key and uneven encryption is headquartered on two keys known as confidential key and public key. Personal key stays secret and public secret's publically available. In asymmetric encryption, public key's used to encrypt the message and confidential key's used to decrypt the same message. Safety is the essential controlled factor now days ordinarily issues with big understanding alternate procedure like internet. More commonly customers demand at ease verbal exchange above all in case organizational linkage, Governmental conversation and banking transactions. Cryptographic algorithms are dependable phenomena in this situation.

The first part of the project will consist of an overview of Java cryptography .

Cryptography additionally depends on encryption strategies equivalent to symmetric and asymmetric to encode the exact text message like plain textual content with the usage of secret code referred to as key. The process of encoding or encrypting the undeniable textual content is referred as enciphering or encryption and the vise versed system is referred to as decoding or decryption. Symmetric encryption requires a single shared secret code referred to as confidential key and uneven encryption is headquartered on two keys known as confidential key and public key. Personal key stays secret and public secret's publically available. In asymmetric encryption, public key's used to encrypt the message and confidential key's used to decrypt the same message. Safety is the essential controlled factor now days ordinarily issues with big understanding alternate procedure like internet. More commonly customers demand at ease verbal exchange above all in case organizational linkage, Governmental conversation and banking transactions. Cryptographic algorithms are dependable phenomena in this situation.

*The first part of the project will consist of an overview of what has been discovered in the area of probabilistic cryptography and will conclude with my impression of what potential the area holds. The first goal of the project is to do a study of what different types of probabilistic cryptography are out there. For each type I will see how much research has been put into them and how secure they seem to be. The paper will also need to check how efficient the encryption and decryption process is. All of this should show if the different encryption schemes have potential to be used in applications and whether probabilistic encryption as a whole has more potential to be studied. I have decided to study 3 different schemes: Goldwasser-Micali, Blum-Goldwasser, and Paillier’s scheme. There are others out there but these seem to be the main ones that are mentioned. The others are based on altering these schemes.*

*In the second part of my project I will create my own programs that will use one or more of the probabilistic encryption schemes. The plan is to test the timing of several versions of Paillier’s scheme versus RSA. It will be interesting to compare the schemes in speed and run some practical tests on them. By creating my own program of the schemes it should allow plenty of opportunities to test every aspect of it. If time permits, other probabilistic schemes could be implemented for more testing.*

*“For purposes of digital signing and documents, verification of digital signatures, and handling digital certificates in the Java platform, the JCA is used. JCA is a specification that gives programmers a standard way to access cryptographic services, digital signatures and digital certificates.” Mobile.developer.com/java/…*

# MOTIVATION OF THE PROPOSED PROBLEM:

Cryptography plays a valuable role in networks and information confidentiality. Security protocols that aid to achieve distinctive security offerings make use of cryptography. Cryptography is a technology that may play fundamental roles in addressing precise varieties of expertise vulnerability. Cryptography enables individuals to preserve confidence within the digital world the place humans can do their trade on electric channel without stressful of deceit and deception.

In recent times, Confidentiality is seen because the central hindrance within the area of expertise security and cozy verbal exchange is the easy use of cryptography. The development of public-key cryptography creates a large-scale network of men and women who can verbal exchange securely with one another despite the fact that that they had never communicated earlier than. Typically, without cryptography money machines would now not be viable, because the machines would now not be ready to reliably keep in touch with the bank computers. Without cryptography, even the thought of electronic voting would not be viable. Cryptographic ideas may also be used in making message easier to decode. In an group, knowledge is the principal asset after human assets and cryptography plays a principal function in securing the expertise. Thus cryptography is principal in day-to-day life.

# MAJOR ISSUES:

In this part, we are able to see the several motives and challenges faced via cryptography. Principal hassle is the changing environment and chance items wherein cryptology will likely be deployed. When you consider that we're evolving closer to ambient intelligence, pervasive networking or ubiquitous computing, which have fully new traits, this would be a major problem. The next hindrance will also be the gradual erosion of the computational diﬃculty of the mathematical problems on which cryptology is based. This erosion is created partially by means of developments in and partially through progress in cryptanalytic algorithms. Additionally, the standards of recent applications and cryptographic implementations together with the dearth of physical safety in devices will also be acknowledged as a main quandary faced by cryptography.

**Book notes:**

Certificate Authority (CA)

Key

Generate Cipher code

S boxes

One time pad

Key space

Symmetric or private key

Asymmetric or public key

Stream buffers are finite state machines

Block ciphers are stateless machines

DES

3DES

AES

Issues with symmetric

Asymmetric

Digital signature

Nonrepudiation

Key distribution and PKI

Key Server

Public key example : RSA

Book Topic questions:

* Vigenere is a \_\_\_ cipher
* Although DES uses blocks of binary numbers transformed using a 64-bit key, there are only\_\_\_ \_\_\_\_ available to the actual key because 8 are used as parity bits and are dropped from the cipher algorithm
* How many Feistel cipher rounds does DES perform
* Euler’s totient function returns:
  + A public key half
  + Key space from which a key can be chosen
  + Positive integers less than n that are relatively prime to n
  + Large factors of prime numbers
* A key (or sets of keys acts too seed values for a cipeher (T/F)
* The DES cipher uses a single mode of operation (T/F)
* With DES, the electronic codebook(ECB) model is the simplest, but least secure (T/F)
* What is a hash function
* DES/ECB:
  + Uded 128 bits for a key to process each block of data
  + Uses 64 bits at a time and each block of data
  + Uses two keys: one to encrypt and the other to decrypt
  + Is the strongest of encryption modes for DES
* \_\_\_ and \_\_\_ are used in modern ciphers

Questions for further Study:

* Write a description of how Kerberos [erforms authentication, focusing on cryptography
* Explain how digital signatures are created and used
* Explain why AES is a preferred cryptography over DES or even 3DES
* Explain how PGP works, and explain why it is considered a hybrid public/private key cryptography

Key concepts and terms

* Asymmetric key (public key) cryptography uses two keys, one to encrypt and one to decrypt
* Certificate authorities are organizations that, for a fee, will attest to the authenticity of a third party by means of cryptography
* Digital signatures work by using what is known as a hash or message digest
* Keys need to be generated by a system so that they are random and significantly large
* Public keys are split in half, accompanied by a common prime. One is distributed as a public key. The other kept secret
* Symmetric ley (private key) cryptography uses one key shared between communicative partners
* Authentication is the process of validating a purported user
* Identification is the process of selecting or singling out an individual from a group
* Kerberos’s KDC has two functions: an authentication service (AS) and a ticket-granting service (TGS)

**Java notes:**

Message digest

Private key Crypto – cipher blocks : Algos: DES

Public Key - RSA

Digital signature

Digital certificate

Example project

#### **Encryption through Steganography, SOFTWARE DEVELOPMENT**

This project implements encrypting text data in an image file, which can be decrypted by another person using the same application. The user enters text he wants to send another person on GUI. The user also selects the image he wants this text to be encrypted in through GUI. The user can now select a phrase to encrypt the text he has enter through AES 128 bit encryption. On clicking encrypt, the text is encrypted in the image file. This image file created can be normally used to view the image. To view the text hidden in the image, a user need to input the encrypted image file, enter the phrase used for AES encryption and press the decrypt button on the GUI. The text would now be decrypted from the image and shown to the user.  
The application implemented Swings for the user interface. File input/output for reading and writing to the file. String functions to handle the image and text data, threads to do multiple steganography (encrypting multiple text messages to images, upto 4) at the same time. AES encryption algorithm for 128 bit AES encryption.

Technologies used - Java

## Types of Cryptography:

There are two main frameworks in cryptography known as:

* Secret key encryption
* Public key encryption

Secret key encryption makes use of a single key to both encrypt and decrypt messages. As such it need to be reward at both the supply and vacation spot of transmission to allow the message to be transmitted securely and recovered upon receipt on the correct vacation spot. The key ought to be kept secret by way of all events concerned within the communication. If the key falls into the fingers of an attacker, they would then be in a position to intercept and decrypt messages, therefore thwarting the try to obtain at ease communications by way of this process of encryption.

Public key encryption uses a pair of keys, every of which is able to decrypt the messages encrypted by using the opposite. Furnished this kind of keys corresponding to exclusive key's kept secret, any verbal exchange encrypted utilizing the corresponding public key may also be regarded comfortable as the one person able to decrypt it holds the corresponding confidential key. The algorithmic houses of the encryption and decryption methods make it infeasible to derive an exclusive key from a public key, an encrypted message, or a blend of each.

Both the secret-key and public-key ways of cryptology have distinctive flaws. The challenge with public- key cryptology is that it's situated on the remarkable measurement of the numbers created with the aid of the combination of the key and the algorithm used to encode the message. These numbers can reach incredible proportions. The manager predicament with Secret Key algorithm is how the two customers agree on what secret key to use. The drawback with secret-key cryptology is that there's mostly a place for an unwanted third party to hear in and acquire know-how the customers do not need that character to have. This is identified in cryptology as the important thing distribution situation. It is among the quality challenges of cryptography.

In the recent decade, using high computing processors are creating an assorted situation for a symmetric encryption which are stylish on small length of key for the reason that dispensed computing can brake small key with no trouble at the same time it also faces issues as a result of at ease key alternate. Without secret and secure key exchange, the symmetric key trade turns into unconfident. The large problems which might be confronted are foundation authentication and crew founded comfy alternate. There is not any assurance that during the time of exchanging secret key, the acquired key is just not falsely modified by a hacker and sent from the professional sender.

The asymmetric scheme is hundred times slower then symmetric ones, on account that it offers with tremendous keys and involves third get together which might be risky for communication between two countries as a result of spy attacks and political reasons. Issuing and renewing of certification requires time and will have to be cost mighty. When big information is involved, it is not feasible because of laziness of encrypting system that requires lot of Random access memory (RAM) and electric energy.

## Digital Signature:

With the aid of digital signature, origin authentication can be archived which is quintessential for protection however the required minimum key size is 1024 bit for digital signature which is the greatest hurdle in processing velocity. The hash services are quick in processing but hash operate do not provide origin authentication. Message authentication codes are fast and are based on symmetric key. For that reason it requires to share and agree on single key as a prerequisite for encryption.

Reliable signature scheme for authentication and integration shouldn't be offered by way of quantum cryptography where it neighbors the likelihood of denial of provider and man in core assault. Signature verification is gradual in elliptical curve cryptography than symmetric encryption.

## Types of Ciphers:

Steganography itself is only a process of hiding know-how nevertheless it cannot provide required protection objectives and concealing method with image results the gigantic measurement of message. The primary difficulty with steganography is the implementation of Statistical and Radiofrequency methods like size and Signature intelligence (MASINT) used to fully grasp the inter–bit delays for cracking the expertise.

## Digital Certificate:

Excessive efficiency is required for efficient answer in purposes comparable to tough disk and bus encryption requiring encryption in terabit networks. If cryptography seems to be very high-priced without feasibility, it's going to now not be deployed. Taking reference to Gilders legislation which predicts that speed and storage will broaden with a factor of 10,000 for LAN devices and Moore’s law which predicts that the computing power for the identical rate will develop with a component of about a hundred, reflecting parallelism is significant in cryptographic operation and highlights the need for high efficiency design.

# JUSTIFICATION ON SOLVING THE PROBLEM:

By tackling the difficulty faced in cryptography, we are able to assure that the customers in each the tip can confidently trade knowledge and preclude any exploitation from 1/3 party. With the emphasis on refinement in the problems confronted in cryptography, an expand in information confidentiality, knowledge integrity, and authentication can also be accomplished. Consequently making it safer for the entities and individual who use the internet for transactions and conversation.

We will be able to be utilizing quite a lot of countermeasures so as to eliminate the maximum quantity of hazard. With probably the most original problems, we can be dealing with each of them in one other technique to curb the problems.

# EXPECTED RESULT:

With an in-depth understanding of Java cryptography, our workforce believes that the problems confronted may also be overcome with few corrective measures and strategically alterations to the overall body work accompanied with the attention measures from users.

# REFERENCES:

Workman, Michael; Phelps, Daniel; Gathegi, John. *Information Security for Managers.* Jones & Bartlett Learning, LLC. 2013.

Lokhande, P.S.; Meshram, B.B.. *E-Commerce Applications: Vulnerabilities, Attacks and Countermeasures.* February 2013[. http://ijarcet.org/wp-content/uploads/IJARCET-VOL-2-ISSUE-2-499-509.pdf](.%20http:/ijarcet.org/wp-content/uploads/IJARCET-VOL-2-ISSUE-2-499-509.pdf)

Java Security. <http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html>

*Java Cryptography Architecture (JCA) Reference Guide*. <http://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html>

Rubin, Brad. *Java security: Java security, Part 1: Crypto basics.* July 2002. [http://www.ibm.com/developerworks/java/tutorials/j-sec1/j-sec1.html](http://www.ibm.com/developerworks/java/tutorials/j-sec1/j-sec1.html%20)

Rubin, Brad. *Java security: Java security, Part 2: Authentication and Authorization.* July 2002

<http://www.ibm.com/developerworks/java/tutorials/j-sec2/j-sec2.html>