**Development of a Compressed Sparse Matrix Generation tool as a Module for Cryptanalysis of Public Key Crypto-systems**

**A Project Report**

**Submitted by**

**Vishwajeet Singh**

***in partial fulfilment for the award of the degree of***

**B.TECH COMPUTER SCIENCE AND ENGINEERING**

***Under the supervision of***

**Ms. Shruti Rawal**



**Scientific Analysis Group**

**Defence Research And Development Organisation**

**New Delhi**

# **CERTIFICATE**

This is to certify that Vishwajeet Singh has successfully completed his project from 1st June 2023 to 31 August 2023 for the partial fulfilment of B.tech (Computer Science). The project titled “Sparse Matrix” was done at SAG, DRDO India, under the guidance of Ms. Shruti Rawal.

We congratulate you on the successful completion of the project and wish you success in all future endeavors!

# **ACKNOWLEDGEMENT**

I would want to express my gratitude to a number of persons who have encouraged and assisted me in the preparation of this Sparse Matrix Project, both directly and indirectly. It allows me to look back and reflect on the support I've had during this process.

I would like to express my gratitude to Ms. Shruti Rawal, my mentor, as well as the Scientist D at SAG, DRDO, for their invaluable recommendations, innovative criticisms, and support throughout the writing process. I would like to offer my heartfelt gratitude to the entire Scientific Analysis Group (SAG).

My heartfelt gratitude goes out to my family and friends, as well as to Defence Research And Development Organisation, New Delhi for providing me with the chance and infrastructure to complete this project. Finally, I want to express my gratitude to everyone who assisted me in gathering data during the creation of the project, without whom it would not have been possible.

# **Index**

|  |  |  |
| --- | --- | --- |
| S.no | Topic | Page No |
| 1. | Cryptology | 5 |
| 2. | Types of Cryptology | 6-11 |
| 3. | RSA Algorithm | 12-13 |
| 4. | Integer Factorization | 14 |
| 5. | Number Field Sieve | 15-17 |
| 6. | Development of a Compressed Sparse Matrix Generation tool as a Module for Cryptanalysis of Public Key Crypto-systems | 18-19 |
| 7. | Project Code | 20-21 |
| 8. | Screenshots of output | 22-23 |
| 9. | Conclusion | 24-25 |

# CRYPTOLOGY

**cryptology**, [science](https://www.britannica.com/science/science) concerned with data [communication](https://www.britannica.com/topic/communication) and storage in secure and usually secret form. It [encompasses](https://www.merriam-webster.com/dictionary/encompasses) both [cryptography](https://www.britannica.com/topic/cryptography) and cryptanalysis.

The term cryptology is derived from the Greek kryptós (“hidden”) and lógos (“word”). Security obtains from [legitimate](https://www.merriam-webster.com/dictionary/legitimate) users being able to transform [information](https://www.britannica.com/technology/information-processing) by virtue of a secret key or keys—i.e., information known only to them. The resulting [cipher](https://www.britannica.com/topic/cipher), although generally inscrutable and not forgeable without the secret key, can be decrypted by anyone knowing the key either to recover the hidden information or to authenticate the source. Secrecy, though still an important function in cryptology, is often no longer the main purpose of using a transformation, and the resulting transformation may be only loosely considered a cipher.

In order For data to be secured for storage or transmission, it must be transformed in such a manner that it would be difficult for an unauthorized individual to be able to discover its true meaning. To do this, security systems and software use certain mathematical equations that are very difficult to solve unless strict criteria are met.

## TYPES OF CRYTOLOGY

Cryptography

Cryptography is the study of securing communications from outside observers. [Encryption algorithms](https://www.encryptionconsulting.com/what-is-an-encryption-algorithm/)take the original message, or [plaintext](https://www.encryptionconsulting.com/what-are-plaintext-and-ciphertext/), and converts it into ciphertext, which is not understandable. The key allows the user to [decrypt](https://www.encryptionconsulting.com/what-is-decryption/)the message, thus ensuring on they can read the message. The strength of the randomness of an [encryption](https://www.encryptionconsulting.com/what-is-encryption/)is also studied, which makes it harder for anyone to guess the key or input of the algorithm. Cryptography is how we can achieve more secure and robust connections to elevate our privacy. Advancements in cryptography makes it harder to break encryptions so that encrypted files, folders, or network connections are only accessible to authorized users.

Cryptography focuses on four different objectives:

1. **Confidentiality**

Confidentiality ensures that only the intended recipient can decrypt the message and read its contents.

1. **Non-repudiation**

Non-repudiation means the sender of the message cannot backtrack in the future and deny their reasons for sending or creating the message.

1. **Integrity**

Integrity focuses on the ability to be certain that the information contained within the message cannot be modified while in storage or transit.

1. **Authenticity**

Authenticity ensures the sender and recipient can verify each other’s identities and the destination of the message.

Using cryptographic techniques, security pros can:

* Keep the contents of data confidential
* Authenticate the identity of a message's sender and receiver
* Ensure the integrity of the data, showing that it hasn't been altered
* Demonstrate that the supposed sender really sent this message, a principle known as non-repudiation

Types of cryptography –

# Symmetriccryptography –

With symmetric cryptography (or symmetric-key encryption), the same key is used for both encryption and decryption.

Symmetric key ciphers are valuable because:

* It is relatively inexpensive to produce a strong key for these ciphers.
* The keys tend to be much smaller for the level of protection they afford.
* The algorithms are relatively inexpensive to process.

1. **Asymmetric cryptography –**

In asymmetric cryptography, each participant has two keys. One is public and is sent to anyone the party wishes to communicate with. That's the key used to encrypt messages. But the other key is private, shared with nobody, and it's necessary to decrypt those messages. To use a metaphor: think of the public key as opening a slot on a mailbox just wide enough to drop a letter in. You give that key to anyone who you think might send you a letter so they can open the slot and deliver the envelope. The private key is what you use to open the mailbox so you can get the letters out.

Cryptanalysis

Cryptanalysis is the study and process of analysing and decrypting ciphers, codes, and encrypted text without using the real key. Alternately, we can say it’s the technique of accessing a communication’s plain text content when you don’t have access to the decryption key.

While the objective of cryptanalysis is to find weaknesses in or otherwise defeat [cryptographic algorithms](https://www.techtarget.com/whatis/definition/algorithm), cryptanalysts' research results are used by cryptographers to improve and strengthen or replace flawed algorithms. Both cryptanalysis, which focuses on deciphering encrypted data, and cryptography, which focuses on creating and improving encryption ciphers and other algorithms, are aspects of cryptology, the mathematical study of codes, ciphers, and related algorithms. Cryptanalysis is practiced by a broad range of organizations, including governments aiming to decipher other nations' confidential communications; companies developing security products that employ cryptanalysts to test their security features; and [hackers](https://www.techtarget.com/searchsecurity/definition/hacker), [crackers](https://www.techtarget.com/searchsecurity/definition/cracker), independent researchers and academicians who search for weaknesses in cryptographic protocols and algorithms. It is this constant battle between cryptographers trying to secure information and cryptanalysts trying to break cryptosystems that moves the entire body of cryptology knowledge forward.

# Cryptanalysis techniques and attacks –

* In a ciphertext-only attack, the attacker only has access to one or more encrypted messages but knows nothing about the plaintext data, the encryption algorithm being used or any data about the cryptographic key being used. This is the type of challenge that intelligence agencies often face when they have intercepted encrypted communications from an opponent.
* In a known plaintext attack, the analyst may have access to some or all the plaintext of the ciphertext; the analyst's goal in this case is to discover the key used to encrypt the message and decrypt the message. Once the key is discovered, an attacker can decrypt all messages that had been encrypted using that key. Linear cryptanalysis is a type of known plaintext attack that uses a linear approximation to describe how a [block cipher](https://www.techtarget.com/searchsecurity/definition/block-cipher) Known plaintext attacks depend on the attacker being able to discover or guess some or all of an encrypted message, or even the format of the original plaintext. For example, if the attacker is aware that a particular message is addressed to or about a particular person, that person's name may be a suitable known plaintext.
* In a chosen plaintext attack, the analyst either knows the encryption algorithm or has access to the device used to do the encryption. The analyst can encrypt the chosen plaintext with the targeted algorithm to derive information about the key.
* A differential cryptanalysis attack is a type of chosen plaintext attack on block ciphers that analyses pairs of plaintexts rather than single plaintexts, so the analyst can determine how the targeted algorithm works when it encounters different types of data.
* Integral cryptanalysis attacks are like differential cryptanalysis attacks, but instead of pairs of plaintexts, it uses sets of plaintexts in which part of the plaintext is kept constant but the rest of the plaintext is modified. This attack can be especially useful when applied to block ciphers that are based on substitution-permutation networks.
* A side-channel attack depends on information collected from the physical system being used to encrypt or decrypt. Successful side-channel attacks use data that is neither the ciphertext resulting from the encryption process nor the plaintext to be encrypted, but rather may be related to the amount of time it takes for a system to respond to specific queries, the amount of power consumed by the encrypting system, or electromagnetic radiation emitted by the encrypting system.
* A [dictionary attack](https://www.techtarget.com/searchsecurity/definition/dictionary-attack) is a technique typically used against password files and exploits the human tendency to use passwords based on natural words or easily guessed sequences of letters or numbers. The dictionary attack works by encrypting all the words in a dictionary and then checking whether the resulting hash matches an encrypted password stored in the SAM file format or other password file.
* [Man-in-the-middle attacks](https://internetofthingsagenda.techtarget.com/definition/man-in-the-middle-attack-MitM) occur when cryptanalysts find ways to insert themselves into the communication channel between two parties who wish to exchange their keys for secure communication via asymmetric or [public key infrastructure](https://www.techtarget.com/searchsecurity/definition/PKI) The attacker then performs a key exchange with each party, with the original parties believing they are exchanging keys with each other. The two parties then end up using keys that are known to the attacker.

Tools for cryptanalysis –

* CrypTool is an open-source project that produces e-learning programs and a web portal for learning about cryptanalysis and cryptographic algorithms.
* Cryptol is a [domain-specific language](https://www.techtarget.com/whatis/definition/domain-specific-language-DSL) originally designed to be used by the National Security Agency specifying cryptographic algorithms. Cryptol is published under an open source license and available for public use. Cryptol makes it possible for users to monitor how algorithms operate in software programs written to specify the algorithms or ciphers. Cryptol can be used to deal with cryptographic routines rather than with entire cryptographic suites.
* Crypto Bench is a program that can be used to do cryptanalysis of ciphertext generated with many common algorithms. It can encrypt or decrypt with 29 different symmetric encryption algorithms; encrypt, decrypt, sign and verify with six different public key algorithms; and generate 14 different kinds of cryptographic hashes as well as two different types of checksums.
* Ganzúa (meaning picklock or skeleton key in Spanish) is an open-source cryptanalysis tool used for classical polyalphabetic and monoalphabetic ciphers. Ganzúa lets users define nearly completely arbitrary cipher and plain alphabets, allowing for the proper cryptanalysis of cryptograms obtained from non-English text. A Java application, Ganzúa can run on Windows, Mac OS X or Linux.

## RSA ALGORITHM

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption. It was invented by Rivest, Shamir and Adleman in year 1978 and hence name **RSA** algorithm.

The **Public key** is used for encryption, and the **Private Key** is used for decryption.

RSA algorithm uses the following procedure to generate public and private keys:

* Select two large prime numbers, p and q.
* Multiply these numbers to find n = p x q, where n is called the modulus for encryption and decryption.
* Choose a number e less than n, such that n is relatively prime to (p - 1) x (q -1). It means that e and (p - 1) x (q - 1) have no common factor except 1. Choose "e" such that 1<e < φ (n), e is prime to φ (n),  
  gcd (e,d(n)) =1
* If n = p x q, then the public key is <e, n>. A plaintext message m is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.  
  C = me mod n  
  Here, m must be less than n. A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.
* To determine the private key, we use the following formula to calculate the d such that:  
  De mod {(p - 1) x (q - 1)} = 1  
  Or  
  De mod φ (n) = 1
* The private key is <d, n>. A ciphertext message c is decrypted using private key <d, n>. To calculate plain text m from the ciphertext c following formula is used to get plain text m.  
  m = cd mod n.

# INTEGER FACTORIZATION

The *integer factorization problem* (FACTORING) is the following: given a positive integer n, find its prime factorization; that is, write *n = pfpf ■ ■ ■pik* where the *Pi* are pairwise distinct primes and each e, > 1.

The problem of *deciding* whether an integer is composite or prime seems to be, in general, much easier than the factoring problem. Hence, before attempting to factor an integer, the integer should be tested to make sure that it is indeed composite.

A *non-trivial factorization* of *n* is a factorization of the form *n = ab* where 1 < *a < n* and 1 < *b < n a* and *b* are said to be *non-trivia! factors*of *n.* Here *a* and *b* are not necessarily prime. To solve the integer factorization problem, it suffices to study algorithms that *split n,* that is, find a non-trivial factorization *n* = *ab.* Once found, the factors *a* and *b* can be tested for primality. The algorithm for splitting integers can then be recursively applied to *a* and/or *b,* if either is found to be composite. In this manner, the prime factorization of *n* can be obtained.

If n > 2, it can be efficiently checked as follows whether or not *n* is a *perfect power,* i.e., *n = xk* for some integers *x >* 2, *к* > 2. For each prime *P* < lg *n,* an integer approximation *x* of *n1^p* is computed. This can be done by performing a binary search for *x* satisfying *n = xp* in the interval [2, 2L1s"/pJ+1]. The entire procedure takes 0((lg3 ?г) lg lg lg n) bit operations. For the remainder of this section, it will always be assumed that *n* is not a perfect power. It follows that if *n* is composite, then n has at least two distinct prime factors.

x · y = N

∑~ei =~0 mod 2

exp( √ 2logN loglogN)

# NUMBER FIELD SIEVE

The Number Field Sieve (NFS) is the best currently known general-purpose [integer factoring](https://doi.org/10.1007/978-1-4419-5906-5_455) algorithm. Since [RSA public-key encryption](https://doi.org/10.1007/978-1-4419-5906-5_153) can be broken by factoring the public-key modulus, the cost of factoring integers with the NFS is used to assess the security of RSA key sizes.

## Poly Selection

The number field sieve is a factorization algorithm used to factor large integers. It is particularly effective for factoring numbers with special algebraic properties, such as numbers of the form a^x - b^y. The polynomial selection phase of the number field sieve is a crucial step in the algorithm that involves choosing appropriate polynomials to use in the sieving process.

The poly selection phase involves finding polynomials that have certain desirable properties, such as having many small prime factors and being relatively smooth over the integer domain. These properties help in reducing the number of potential candidates for the factors of the large integer being factored.

The poly selection number field sieve algorithm is an enhanced version of the general number field.

## Sieving

The Sieving Number Field Sieve (SNFS) is a factorization algorithm, specifically designed to factor large composite numbers. It is based on the Number Field Sieve, but incorporates sieving techniques to significantly reduce the overall complexity of the algorithm.

The SNFS works by first choosing appropriate number fields and constructing suitable polynomials. This is followed by a large sieving process, where potential factors are identified by sieving a large search space of integers. The sieving step is the most time-consuming part of the SNFS algorithm.

Once the sieving process is complete, the remaining potential factors are then subjected to a linear.

## Filtering

The filtering step builds the matrix that is given to the linear

algebra step from the relations computed by the sieving step.

Common to NFS for factorization and NFS for discrete

logarithm.

Also common to other factoring algorithms and to other

discrete logarithm algorithms.

The set of relations is seen as a matrix where:

* 1. a row corresponds to a relation;
  2. a column corresponds to an ideal;
  3. the coefficient is the valuation of the corresponding ideal in the corresponding relation.

# Linear Algebra

The linear algebra number field sieve is a powerful algorithm used for integer factorization. It is particularly efficient for factoring large composite numbers. The algorithm combines techniques from both number theory and linear algebra to compute the factors of a given number.

In the linear algebra number field sieve, a number field is constructed using algebraic number theory techniques. This number field is then used to find polynomial equations, known as "Sieving Polynomials," to generate smooth numbers. These smooth numbers are then used to construct a matrix where each row represents a unique factorization of a smooth number.

The algorithm then uses linear algebra techniques, such as Gaussian elimination, to solve the matrix equation and find a non-trivial solution.

# Square Root

The square root number field sieve is a factoring algorithm that is used to find the prime factors of large numbers. It is an advancement of the number field sieve algorithm, which was developed in the 1980s.

The square root number field sieve algorithm works by taking the square root of the input number and then using number fields to find the prime factors. This algorithm is particularly efficient for factoring numbers that are of the form "p^2 - q^2", where p and q are primes.

One advantage of the square root number field sieve over other factoring algorithms is its theoretical efficiency. It has been shown to be significantly faster than other methods, such as the quadratic sieve algorithm.

# Development of a Compressed Sparse Matrix Generation tool as a Module for Cryptanalysis of Public Key Crypto-systems

The development of a compressed sparse matrix generation tool as a module for cryptanalysis of public key crypto-systems is a significant step towards enhancing the efficiency and accuracy of cryptanalysis algorithms. A compressed sparse matrix representation enables efficient storage and manipulation of large matrices, saving both memory and computational resources.

By incorporating this tool into the cryptanalysis process, researchers and practitioners can improve the scalability and speed of their analysis. The compressed sparse matrix generation tool can efficiently handle the large matrices involved in public key crypto-systems, reducing the computational burden and enabling quicker analysis.

Moreover, the use of compressed sparse matrices can enhance the data representation and analysis techniques used in cryptanalysis. The tool can facilitate the efficient handling of structuredpatterns and sparsity in the matrices, leading to more effective analysis.

This module can also benefit cryptanalysts by providing a more streamlined and automated approach to matrix generation. With the compressed sparse matrix generation tool, cryptanalysts can save significant time and effort in generating the required matrices for their analysis. They can focus more on developing and implementing advanced algorithms instead of getting bogged down by manual matrix generation.

Furthermore, the integration of this module into cryptanalysis tools can promote collaboration and knowledge sharing among researchers. The standardized and optimized matrix generation process can be easily shared and utilized by different researchers working on similar problems, fostering a more efficient and cohesive research community.

# Project Code

*#include* <iostream>

*#include* <math.h>

*#include*<time.h>

*#include* <stdlib.h>

*#define* n 10

*#define* exp 3

*#define* z pow(n, exp)

using namespace std;

int main()

{

    srand((unsigned)(0));

    int ran\_data;

*for*(float index = 0.5; index<=1;index++){

        ran\_data = (rand() % 5129) + 1;

    }

    FILE \*fp;

    fp = fopen("sample.txt", "w+");

    int \*ptr;

    int a;

    ptr = (int \*)calloc(a, sizeof(int));

    int countRow;

*for* (int i = 1; i <= z; i++)

    {

        int countRow = 0;

*for* (int j = i; j <= z; j++)

        {

*if* ((i\*j) % ran\_data == 1)

            {

                ptr[countRow] = 0;

                countRow++;

            }

*else*{

                ptr[countRow]=1;

            }

        }

        fprintf(fp, "%d\t", countRow);

*for* (int i = 0; i < countRow; i++)

        {

            fprintf(fp, "%d\t", ptr[i]);

        }

        fprintf(fp, "\n");

    }

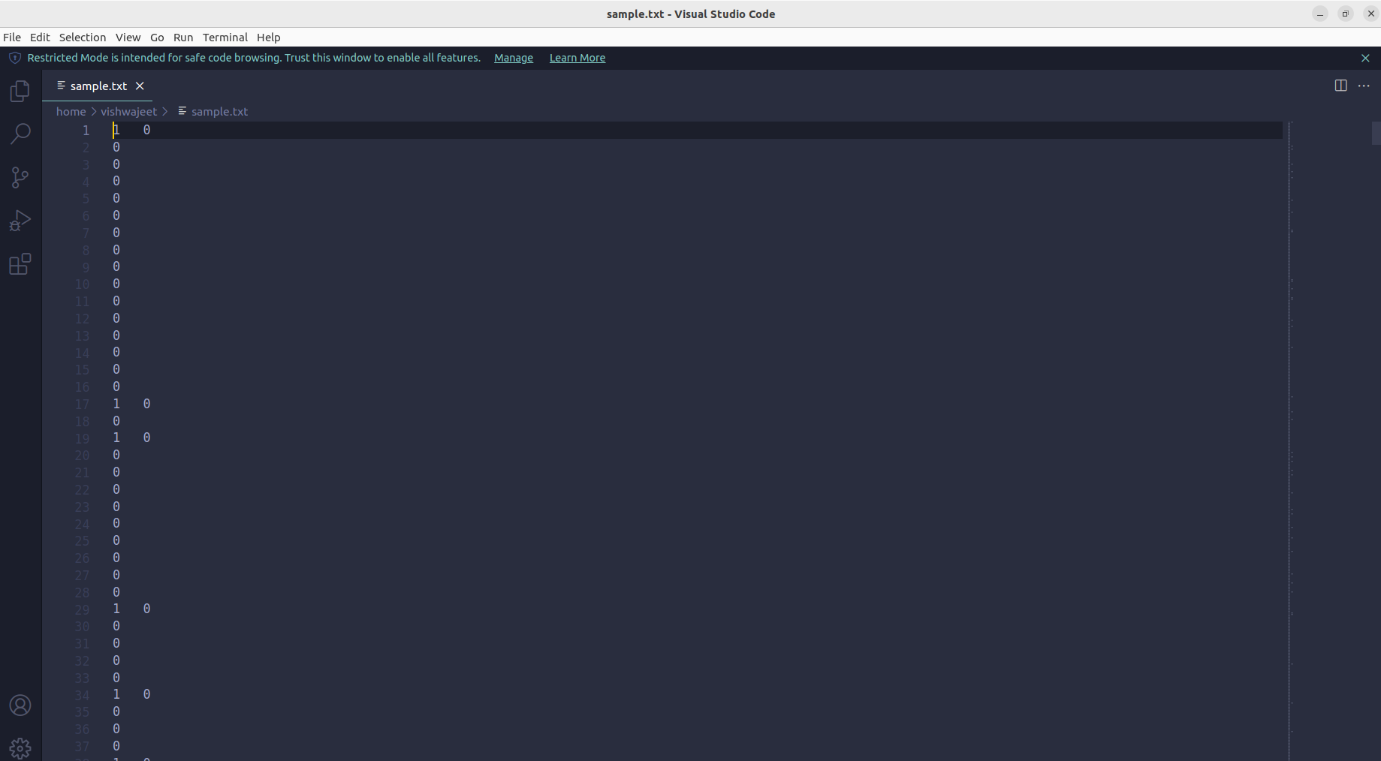
    free(ptr);

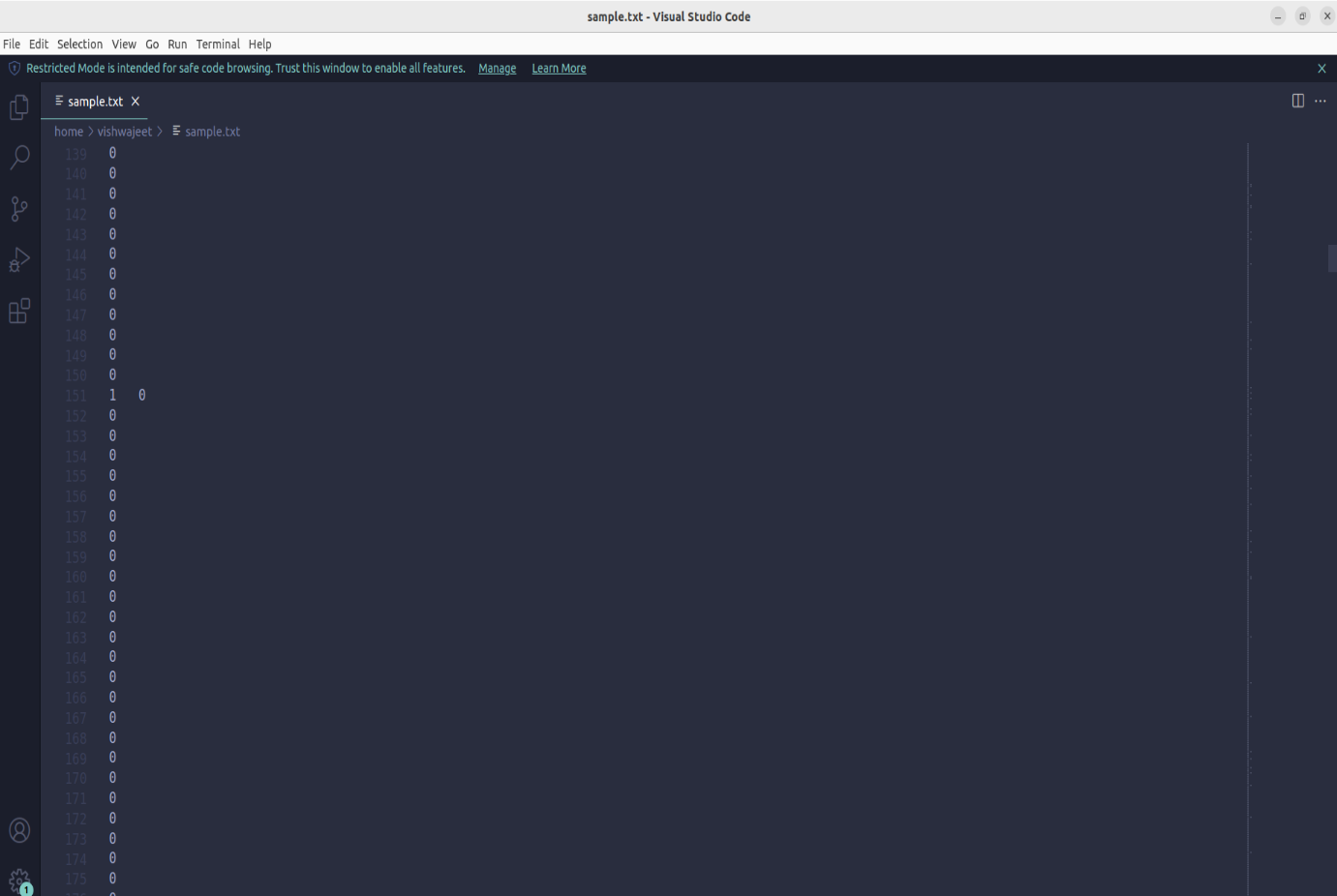
    fclose(fp);

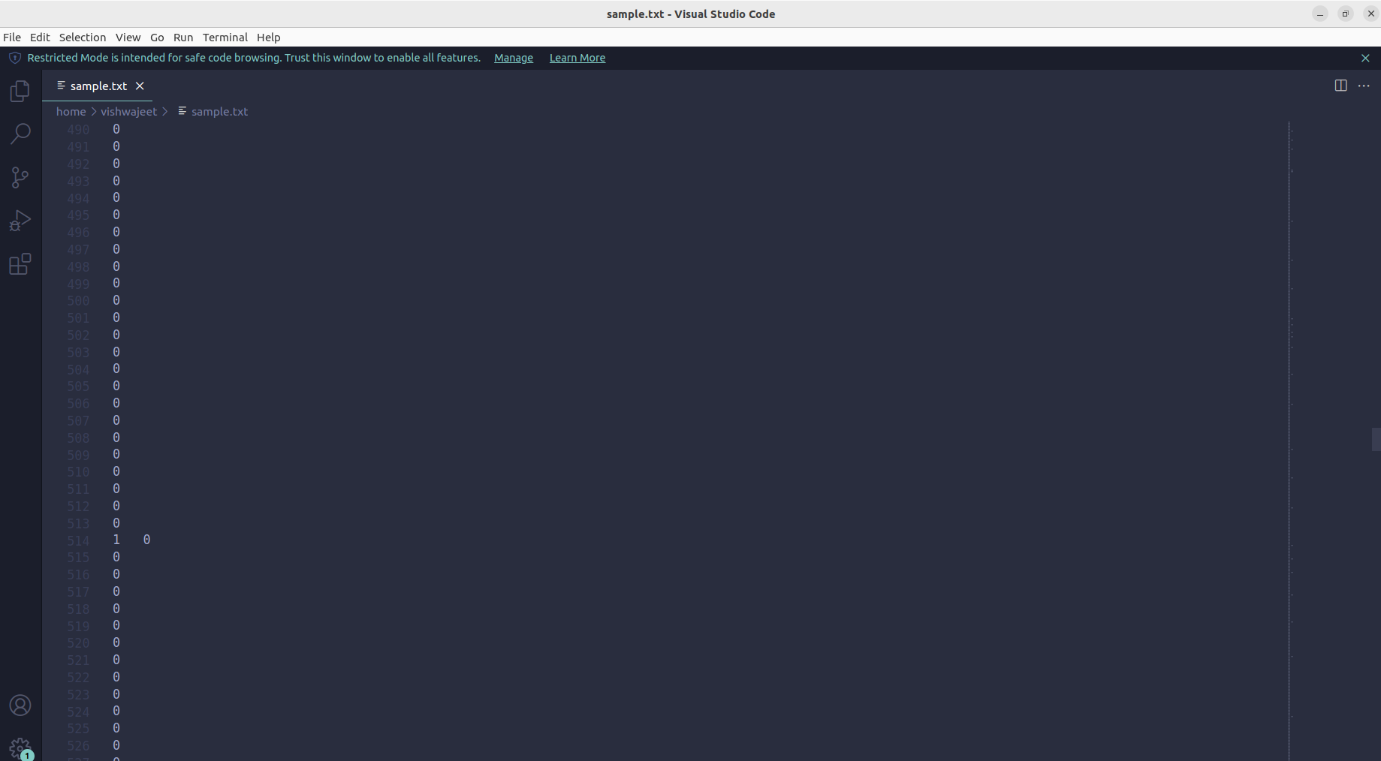
*return* 0;

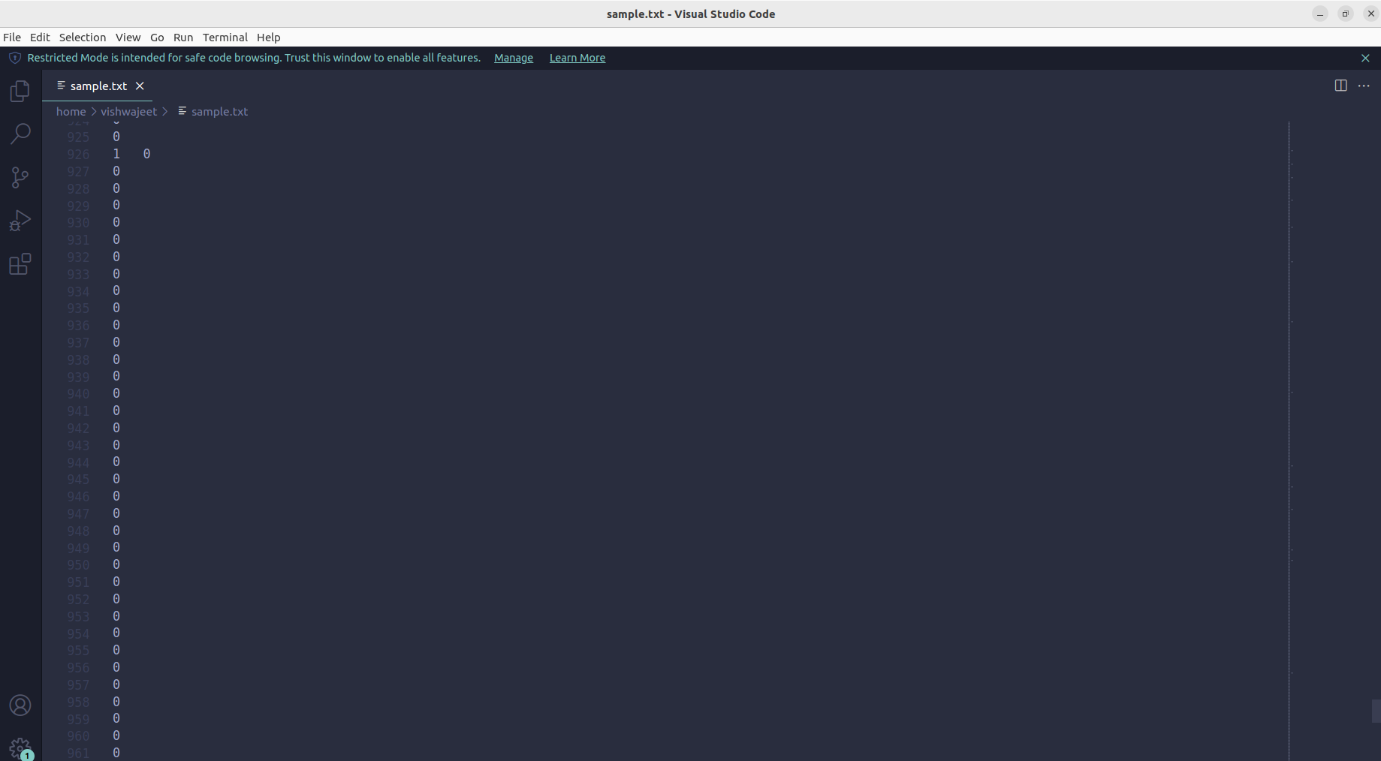
}

# Screenshots of Output









# Conclusion

The development of a compressed sparse matrix generation tool as a module for cryptanalysis of public key crypto-systems is a significant step in enhancing the efficiency and effectiveness of cryptanalysis algorithms.

By utilizing compressed sparse matrix representation, the module can efficiently store and manipulate the large matrices involved in cryptanalysis, resulting in reduced memory usage and computation time. This is particularly crucial in cryptanalysis, where the size and complexity of matrices associated with public key crypto-systems can be immense.

Moreover, the compressed sparse matrix generation tool enhances the scalability of cryptanalysis algorithms, allowing for more extensive and rapid analysis of different public key crypto-systems. This can significantly contribute to the advancement of cryptanalysis techniques and the potential discoveryof vulnerabilities or weaknesses in public key crypto-systems.

Furthermore, the development of this module can facilitate better collaboration and sharing of research in the field of cryptanalysis. Researchers working on different public key crypto-systems can utilize the compressed sparse matrix generation tool as a common framework, enabling them to exchange ideas, techniques, and data more efficiently. This collaborative approach can accelerate the overall progress in strengthening the security of public key crypto-systems.

In conclusion, the development of a compressed sparse matrix generation tool as a module for cryptanalysis of public key crypto-systems holds immense potential for improving the efficiency, scalability, and collaboration in the field. By providing a more efficient way to handle large matrices and enabling faster