Secured proxy re-encryption with post-quantum cryptography for android and its performance bottlenecks

MSc Research Project
MSc Cyber Security

Waleed Mustafa
Student ID: x20251785

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Waleed Mustafa……………………………………………………………………………………
Student ID:	x20251785……………………………………………………………………………………..……
Programme:	MSc Cyber Security………………………………… **Year:** 2021/2022………….
Module:	MSc Research Project / Internship…………………………………………….………
Supervisor:	Michael Pantridge………………………………………………………………………….………
Submission Due Date:	15/08/2022……………………………………………………………………………….………
Project Title:	Secured proxy re-encryption with post quantum cryptography for android and its performance bottlenecks………………………………….………
Word Count:	6250…………………………… **Page Count** 18……………………………………….……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: waleed…………………………………………………………………………………………

Date: 04/08/2022……………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Secured proxy re-encryption with post quantum cryptography for android and its performance bottlenecks

Waleed Mustafa
x20251785
MSc cyber security
National College of Ireland

Abstract

The purpose of this report is to research the secure proxy re-encryption method by utilizing post-quantum cryptography. The disadvantages of this method are also evaluated by making an android application and testing this method on different models. The major bottleneck of this method is large encryption keys which take more time to encrypt data. There are three different parts of this methodology owner, user, and SRS (Setup and Re-encryption Server). Public cloud storage is also used in the implementation but that is only used to store data. The owner uploads their encrypted data into the cloud storage and controls the access by ACL (Access Control List). Users can download that data and SRS is responsible for generating encryption key pairs. In this whole process, the encryption key is never shared with CSP (Cloud Service Provider), and SRS has no access to owners' data. The encryption and decryption processes are handled by the end-user applications. Our method of sharing data is also secured against any type of post-quantum attacks.

Table of Contents

1 Introduction 3

2 Literature review 4

2.1 Introduction 4

2.2 Key Definitions 4

2.2.1 Proxy re-encryption 4

2.2.2 Post Quantum Cryptography (PQC) 4

2.3 Related Work 5

2.3.1 Critical analysis of the papers related to proxy re-encryption 5

2.3.2 Critical analysis of papers related to post-quantum cryptography 6

2.4 Gap Analysis 7

2.5 Conclusion 8

3 Model Explanation 8

3.1 Preliminaries 8

3.2 Model Detail 9

3.2.1 Android Devices 9

3.2.2 Cloud Storage 9

3.2.3 Proxy re-encryption 9

3.2.4 Setup and re-encryption server (SRS) 10

3.3 Technical Analysis 11

3.3.1 Java 11

3.3.2 Firebase Cloud Storage 11

3.3.3 NTRU 11

4 Evaluation 12

4.1 Experimental Setup 12

4.2 Results 13

4.2.1 Key Generation 13

4.2.2 Encryption and Decryption 14

4.2.3 Turnaround Time 15

5 Conclusions and Future Work 16

6 References 16

1 Introduction

In the space of IT, one of the most developing fields is cloud computing. In this field data is stored in the pools of storage known as "the cloud". In this scenario, CSP provides the hardware and storage over the Internet. Users have no access to the real hardware. The security of this hardware is also managed by the CSP. For the hosting of clients' applications and storing data organizations lease or purchase the storage from the CSP. The main issue arises when CSP stores data in the storage with symmetric key cryptography. Due to this process, CSP has access to our plain data. The security issue is that any of the internal employees have access to plain data.

A new technique was introduced to remove this security issue known as proxy re-encryption. In this technique, the encrypted data is uploaded to the cloud platform, but the decryption key will not be shared with CSP. Other methods are used to send the decryption key to 2nd user. The user requests the encrypted data from the CSP while at the same time the decryption key is also shared by the owner. There are many ways to share the private key with users sometimes with the help of email. The most reliable and efficient way to share the private is using a dedicated server that manages all key distribution between owner and user. One of the main benefits of using the dedicated server is the owner doesn't have to remain online for 24 hours to share the key.

Our technique is the most secure way of sharing data on the public cloud known as secured proxy re-encryption. This technique is powered by NTRU which is secured against quantum cryptography. The application is built for the android platform. Android devices are turning into a significant piece of our life. Seventy-five percent of cell phone market shares are covered by android. According to different surveys, there are more than 2.8 billion android clients. Most of the smart home appliances are also using android based operating systems.

All methods of conventional cryptography are now susceptible to quantum attacks. The most recent generation of computers that do so is known as quantum computers which substitute Qbits for conventional bits. Qbits are more powerful than regular bits because they can exist in a superposition state. Qbits may be both zero and one simultaneously. The true threat to all widely used security measures is posed by the quantum computer, which offers enormous processing capabilities.

We are going to use the proxy re-encryption approach with quantum-proof cryptography to prevent the threat posed by quantum computers. In the field of computers, each approach has benefits and disadvantages of its own. Because of the huge key sizes required by the quantum-proof encryption, this will defend us from quantum attacks. These huge keys need a highly expensive processing power that also has certain performance drawbacks. The setup and re-encryption server, public cloud storage, and end-user devices are the main three players in our approach. The data will be encrypted before uploading into the cloud storage. The keys will be managed by the SRS server.

The following are the main contributions of our work:

- Quantum secured proxy re-encryption for android users in which user can share their files in the cloud.
- NTRU encryption scheme is used in the implementation of this methodology which is secured against quantum attacks.
- A dedicated server is used for maintaining the encryption keys and ACL (Access Control List).
- Benchmarking of this methodology is done to access the bottlenecks of the proposed system.

2 Literature review

2.1 Introduction

The primary goal of this review of the literature is to examine earlier research on post-quantum cryptography, proxy re-encryption, and significant advancements in this area. The development of quantum computing has made all current, widely used classical encryptions vulnerable. Although several academics have contributed their discoveries in this field, their research studies all have significant flaws. Some people employ a vulnerable technique for using proxy re-encryption. In their research, the other researchers employed antiquated cryptographic methods. Therefore, the most significant studies on this topic as well as its shortcomings have been discussed in this literature review.

2.2 Key Definitions

2.2.1 Proxy re-encryption

The data owner uses this technology to encrypt his data and put it in public cloud storage. So, the cloud is where the encrypted data is kept. The decoding key is not in the possession of the CPP. No one possesses the key to decode the data in the event of an accident, theft, or leak. When a legitimate user requests data from cloud storage, the user receives the plain text after the data has been decrypted. There are several methods for doing this; in some, the owner will send the key to the user through email upon request, while in others, a dedicated server is used. The administration and distribution of keys that this server must do. The encrypted data is obtained from the cloud when a legitimate user wants it, and the proxy key needed to decode it is obtained from another proxy server. (Ali et al., 2014; Ateniese et al., 2006; Chen et al., 2011).

2.2.2 Post Quantum Cryptography (PQC)

The latest generation of computers, known as quantum computers, uses quantum computation to address issues. To function, they have interference, entanglement, and other quantum state features (Ladd et al., 2010). They provided a user with a significant amount of computing power that could somehow break all of the established cryptographic methods now in use. Assume an attacker possesses a powerful quantum computer that can defeat all current types of cryptography. We require a quantum-proof cryptography method for this situation (Bernstein and Lange, 2017). The National Institute of Standards and Technology (NIST) is holding a competition to establish standards for quantum-proof cryptography. The majority of the remaining competitors use lattice-based encryption. Fully homomorphic lattice-based encryption has been proven to be quantum-proof up until this point.

2.3 Related Work

This section is a presentation of earlier work that is relevant to our study. In this part, the main flaws and holes in each relevant study are explored.

The data is simply uploaded to the public by the researchers (Kandukuri and Paturi, 2009; Pecarina et al., 2012) using a regular device. They used a conventional, public key-based encryption method to upload their important data to the somewhat trustworthy cloud. Their medical histories are among the data. Using the key provided by the CPP, the user encrypts the data. The data is then decrypted by CPP using their private key. Then, re-encrypt the data using the symmetric encryption method, and put it in the storage with the location index. This is the problem—this method does not guarantee data confidentiality. The major security concern is that cloud service provider employees may quickly decode data. Therefore, the proxy re-encryption method we have suggested is far more secure than this method since it guarantees the privacy of every user's data.

2.3.1 Critical analysis of the papers related to proxy re-encryption

(Li et al.,2013) presented a novel method called attribute-based encryption to address this problem. With this method, the author uses an ephemeral key pair (SKE) to dynamically regulate who may access the information stored in the cloud. The automatic revocation of the user is the most crucial component of this method. To prevent key management, they only have a limited relationship with the users. There is a flaw in this plan as well. To allow users to access data, the data owner must always be online. If the owner has not been online, the user cannot access the data using this method. Our method does not have any partial relationship problems; therefore, the owner does not need to be online continuously for the user to have access to the data.

To increase the security of data in the cloud, other researchers (Jafari et al., 2011) suggested a Digital Right Management (DRM) strategy. They utilized Content Key Encryption (CKE), which leverages the license to provide the user access to the data. The most secure method, proxy re-encryption, was then presented by (Liang et al., 2009). The safest method for sharing data in the cloud as of right now is this one, however, it does make use of ciphertext characteristics and ciphertext size. With repeated usage, the ciphertext grows in length. It was exceedingly challenging to handle the data as it grew to be too huge. The method used by (Xhafa et al., 2015) also makes use of the ciphertext policy ABE (CP-ABE), which has the same issue with ciphertext length when handling excessive data. When processing a lot of data, this approach has a very serious problem. In our suggested method, this is avoided, hence the owner should not be concerned about the length of the ciphertext.

Authors (Ali et al., 2021) developed the Setup and Re-encryption Server (SRS) as a novel proxy re-encryption technique to address this problem. They utilized a separate server that was just responsible for managing keys. Data is exported to any public cloud after being initially encrypted by the user using their key. Transmit parameters to the Setup and Re-encryption Server concurrently. when a legitimate user requests information from the cloud. The third proxy server provides the user with the decryption key. They are encrypting the data using El-Gamal Encryption, which is not a quantum-proof cryptographic method and is only suggested for computers, not for android devices. Additionally, this technique is limited to Personal Health Records exclusively (PHR).

2.3.2 Critical analysis of papers related to post-quantum cryptography

El-Gamal encryption's semi-homomorphic lattice-based cryptography security can be compromised if you can compute discrete logarithms accurately. Finding the value of x in the equation $\alpha x = \beta \pmod{p}$ allows us to transform a public key into a private one with ease. Utilizing Shor's algorithm makes this simple to do (Lanyon et al., 2007; Tsiounis and Yung, 1998). However, completely homomorphic Lattice-based cryptography methods such as Ring-LWE key exchange, FrodoKEM, NTRU, and LWE key exchange are quantum-proof (Bernstein and Lange, 2017; Septien-Hernandez et al., 2022). In our suggested method, proxy re-encryption as presented by (Ali et al., 2021) is made more secure by using quantum-proof encryption. To protect against quantum attacks, the new proxy re-encryption strategy proposes using a high key size, However, this will also have some bottlenecks on low power android devices.

Researchers (Ali et al., 2014) also suggested an incremental proxy re-encryption strategy, although they do it using a methodology that is more suitable for editing existing files than for uploading new ones securely. As a result, the same authors (Ali et al., 2021) developed a new proxy re-encryption scheme, however, it still has the same drawbacks of security difficulties brought on by quantum computers.

It has already been attempted by certain researchers to assess the efficiency of quantum-proof encryption. Different quantum cryptography algorithms are used by the authors (Khalid et al., 2019), who test their effectiveness on embedded computers. On a cheap FPGA, they used FrodoKEM-640 and FrodoKEM-976. We may conclude from the data that this is the least preferred option for embedded devices. They also addressed the lattice-based cryptography's performance problem in the section on challenges. These systems are battling post-quantum cryptography.

Researchers (Tamilmani et al., 2018) investigated post-quantum cryptography performance concerns on android mobile devices. They make use of a computer with a three GB primary memory and an octa-core CPU. Today's smartphones are powerful enough to manage post-quantum cryptography in some way. Additionally, the majority of the techniques they have utilized have previously been invalidated by the NIST because of security concerns (Computer Security Division, 2017). However, they only track the effectiveness of post-quantum cryptography when experimenting on a power-full device. On low-end devices like android devices, we'll track how well the safe proxy re-encryption mechanism performs.

Some researchers (Cheng et al., 2020) present a novel cryptographic approach for power-efficient devices to address these issues based on NTRU Prime cryptography. Great results were obtained when they tried this with an 8-bit AVR microcontroller. This was the response to the query we raised above, however, in the NIST competition's first round, NTRU Prime was disqualified because of security issues. Only four finalists for the public key encryption category survive in the final round of the NIST competition: Kyber, NTRU, SABER, and Classic Mceliece (Computer Security Division, 2020; O. Saarinen, 2020).

2.4 Gap Analysis

In gap analysis, we attempt to list all the drawbacks and benefits of earlier research in comparison to the suggested technique in this section.

Table 2.1 Gap Analysis

Related Work	Advantages	Disadvantages
• (Li et al., 2013)	• Quicker because of smaller keys	• To allow access, the owner must be reachable online for 24 hours. • Not secure
• (Liang et al., 2009) • (Xhafa et al., 2015)	• Quicker because of smaller keys	• The issue to handle large files due to large ciphertext length • Not secure
• (Ali et al., 2021)	• Most trustworthy than the previously suggested plans because of SRS	• Only works with PHR • Not available for low-power devices • Not secure
• Our approach	• It is more secure than any other suggested methods due to post-quantum cryptography • Not restricted to PHRs • Owners don't have to be online for 24 hours because of SRS	• Have performance issues on large data due to a huge number of files created after the encryption process • The key size is very large • Does not have a support of continuous data

2.5 Conclusion

Some doubts about the significance of using the proxy re-encryption process result from the research above. What if we just utilize a public encryption key that the CPP provides? There is a well-known incident when a US DVA employee obtained the personal health information of 26.5 million people without their consent. Since this occurrence, HIPAA has made it a requirement that no one interferes with the privacy of computerized personal health data (Ali et al., 2021). We already know that post-quantum encryption is a challenge for the Internet of Things. We go one step further and test this on a little bit more powerful device than IoT. We have implemented our suggested secure proxy re-encryption on android devices and measured their performance.

3 Model Explanation

Figure 3.1 is the model diagram of our implemented methodology.

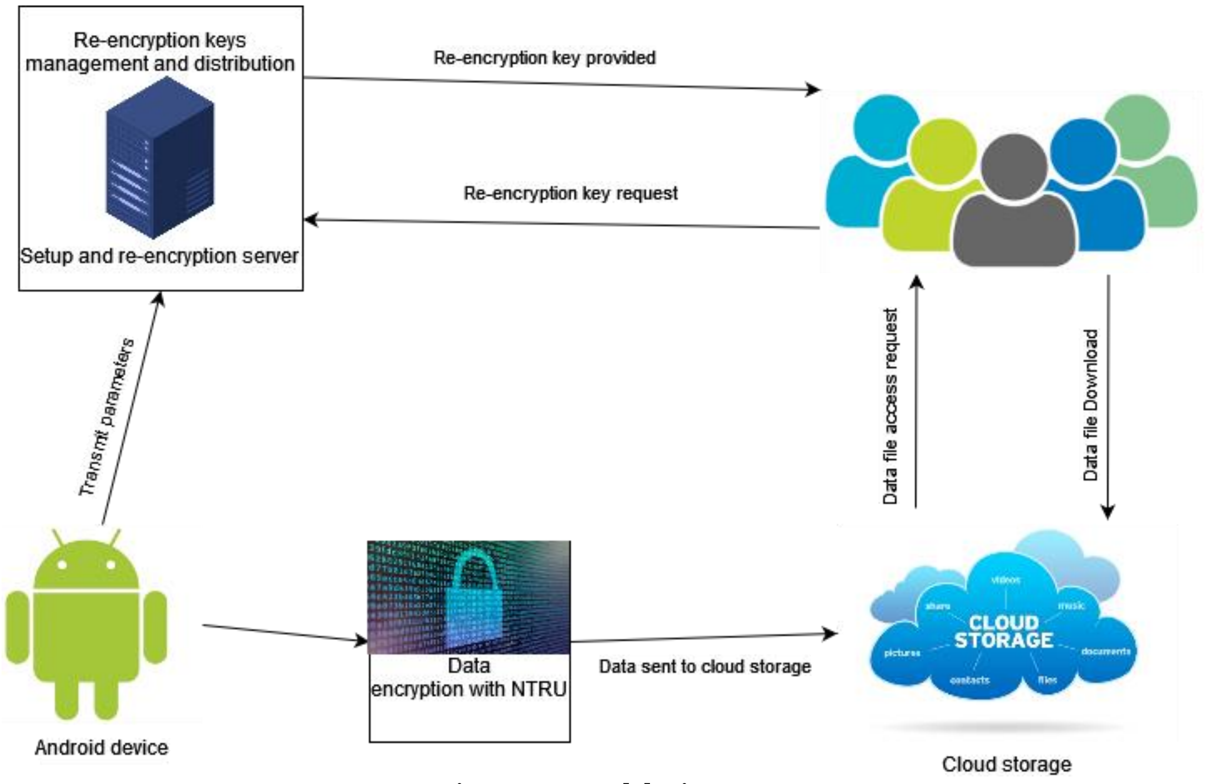
Re-encryption keys management and distribution
Setup and re-encryption server
Re-encryption key provided
Re-encryption key request
Transmit parameters
Data file access request
Data file Download
Android device
Data encryption with NTRU
Data sent to cloud storage
CLOUD STORAGE
videos
share
music
pictures
documents
contacts
files
Cloud storage

Figure 3.1 Model Diagram

Major components of this technique

- Android device
- Cloud storage
- Proxy re-encryption
- Setup and re-encryption server (SRS)

3.1 Preliminaries

The proxy re-encryption that is being suggested is the most secure one ever implemented. We used NTRU which is post-quantum cryptography to increase the security of proxy re-encryption, which was proposed in (Ali et al., 2021). Everything has benefits and drawbacks. It has performance constraints on low-end devices due to very high key sizes, making it potentially inappropriate for low-end devices, it is developed for android smartphones. However, the focus of this research is not on network security. The sockets used for communication between the SRS and android devices are not encrypted.

3.2 Model Detail

3.2.1 Android Devices

The primary device in the implemented proxy re-encryption must perform the majority of the work. An individual must first register for the app. After that, he may upload any file. The file will be divided into parts of 65 bytes. Then, using the NTRU cryptography encryption key provided by the SRS, each of these parts is encrypted independently. Then every part is saved in a different file by using base 64 encoding and uploaded to the firebase public cloud. The user must enter the username to whom they want to see the file before uploading it. Following then, together with information about which user this file is permitted to decrypt, the parameters for the decryption key will be sent to the server for re-encryption.

On the other hand, the user will ask the SRS for the decryption key and the cloud for the data at the same time. The only data that is uploaded for that user will be decrypted. The data that the owner uploads for one user will be accessible to only that user. The SRS will not disclose the data decryption key to the other users. Android Studio is used to write the code of the android application.

3.2.2 Cloud Storage

We can utilize any publicly accessible cloud storage for our project. Other researchers have tested their applications on Google Cloud and Microsoft Azure and employ various types of public cloud storage (Xhafa et al., 2015). On AWS S3, the authors of (Ali et al., 2021) tested their application. Here, it is considered that the public cloud is an unreliable source, hence only encrypted data will be uploaded to its storage. The encryption key is not made available to CPP. In most cases, users encrypt data using the CPP's public key, and the CPP subsequently decodes it using their private key. The data is once again encrypted using symmetric encryption before being stored (Yandong and Yongsheng, 2012), yet the CPP has complete access to our raw data. The encryption and decryption are carried out on the end devices under the suggested technique, meaning that CPP does not have access to our plain data. As we have implemented our technique for the android device firebase has the best support for the android platform. So, we are using firebase cloud storage.

3.2.3 Proxy re-encryption

This method involves a third semi-trusted party across the entire model to guarantee that data security is never affected. In this method, plain data is available to both end parties. The semi-trusted server and CPP have no access to any plain data. Despite having encrypted data, the CPP lacks the decryption key. Although the decryption key is on the re-encryption server, the data is not accessible. As a result, we also refer to it as end-to-end encryption with cloud storage capability.

3.2.4 Setup and re-encryption server (SRS)

The server that will handle all keys used to encrypt and decrypt data is semi-confidential. Each time a user uploads data to a cloud storage system the SRS receives the parameters for the decryption key. The owner also provides the username for which data is uploaded. The owner never shares data with the SRS since it is a semi-trusted server. The SRS will get just create an encryption key pair whenever new user signup. The SRS will offer the key whenever the user uploads or wants any data from the cloud. In this instance, our simple data is not accessible to SRS or Cloud. On the end devices, encryption and decryption take place. The SRS oversees all keys. This server is also in charge of access control because of the ACL. Because the cloud is not a reliable party in our circumstance, this server is not set up on any public cloud platforms. Any entity, such as a group of people may manage the SRS. The communication between the SRS and end application is done by sockets however sockets are not encrypted because network security is not in the scope of this research. NetBeans IDE is used to write the code of SRS.

Figure 3.2 Shows all the internal activity of our implemented methodology.

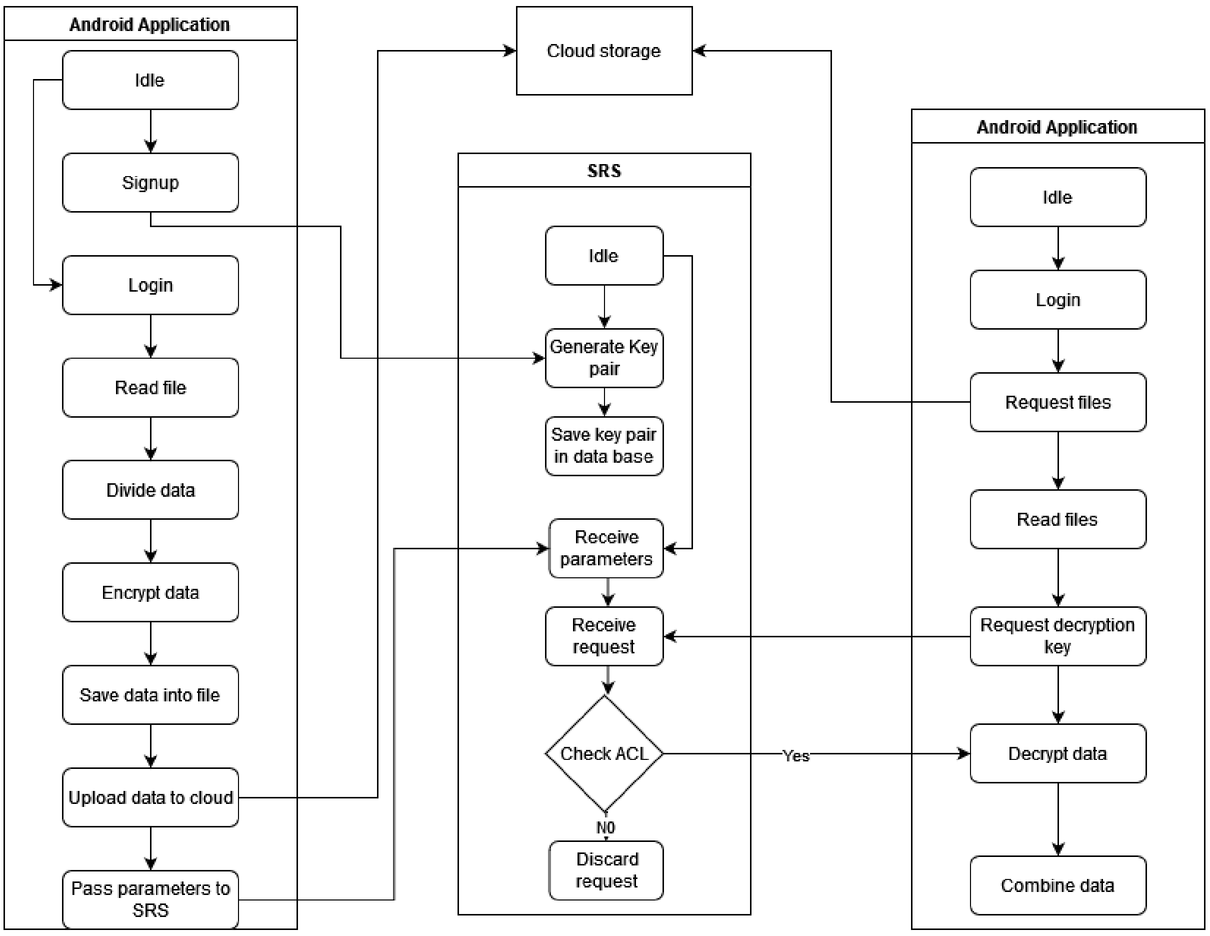

Figure 3.2 Activity Diagram

The user first registers for the application. A new encryption key pair is generated and stored in the database every time a new user registers with the application. The user program splits the data into many 65 bytes portions before encrypting each part with the user's public keys, which the SRS has supplied. After each component has been encrypted, it is stored into numerous base 64-encoded files and sent to the cloud storage. The program sends the parameters indicating which data is uploaded for which user in the last stage. The Android application will carry out the entire process. The second entity in this diagram is SRS. SRS is the second element in this diagram. For every new user, SRS produces the encryption key pairs. The parameters from the application will be sent to SRS. After that, it will wait for the decryption key request. SRS will verify the ACL to make sure that this user has access to decrypt data when it gets a request for the decryption key. It will transmit the decryption key to the user if they are listed in the ACL. The server will ignore the request if the user is not listed in the ACL.

The application on the other end of this activity diagram initially makes a cloud-based data request. The public cloud storage will be used to download the encrypted data. Now that it requires a decryption key, which the SRS has, this program will ask the SRS for the decryption key. The program will decrypt all the data files, combine them, and then provide the user with meaningful information.

3.3 Technical Analysis

The NTRU encryption scheme will be used as the quantum-proof cryptography in the implementation of this application, which will be implemented in the Java programming language. The public cloud storage will be a firebase cloud storage.

3.3.1 Java

The high-level programming language Java gives you access to an object-oriented methodology. Abstraction, polymorphism, inheritance, and encapsulation are the four foundations of OOP. The Java programming language is a core tenet of the Android JDK. Java is the language used to write any android application.

3.3.2 Firebase Cloud Storage

Firebase Cloud Storage service is used to store and download files created directly by clients. There's no requirement for server-side code. The files are kept in Google Cloud Storage buckets, providing access from both Google Cloud and Firebase.

3.3.3 NTRU

We have used NTRU encryption in this project. NTRU is based on the lattice-based public/private key cryptography method created by Joseph H. Silverman, Jill Pipher, and Jeffrey Hoffstein. NTRU completes the time-consuming private key task far more quickly than RSA does. The time required to accomplish actions involving private keys in RSA also rises as the square of key size does, whereas NTRU activity rises quadratically. The three main parameters that make up the mathematical notation for NTRU (N, p, q). The polynomial that has been reduced includes NTRU operations (Hoffstein et al., 1998; Lange, 2015).

$$\mathcal{B}(d) = \left\{ f(X) = \sum_{i=0}^{N-1} f_i X^i \in \mathcal{P} \mid f_i \in \{0,1\}, \sum_{i=0}^{N-1} f_i = d \right\}.$$

Key pair
f ∈ Lf random polynomial.
To produce, you must compute fp ≡ f-1 and fq ≡ f-1 (mod of q).
A Polynomial g ∈ Lg at random.
To produce, you must compute h ≡ g* fq (mod of q).
The parameters for the public key are (N, h) and p, q, Lf, Lg, Lr, and Lm.
The private key will continue to exist (f, fp).

Encryption
The message will be m ∈ Lm.
A random polynomial r ∈ Lr.
It will be encrypted with the public key according to the formula e ≡ p * r * h + m (mod of q).

Decryption
The user will calculate a ≡ f * e. (mod of q).
The user must convert a to a polynomial in the range [-q/2, q/2 [.
Determine m ≡ fp * a. (mod of p).

Most lattice-based encryption methods are resistant to quantum attacks, as we know from related research. In the NIST competition's last round, there aren't many remaining. The most promising and effective substitute for today's most widely used encryption methods is NTRU, which is quantum-proof (Computer Security Division, 2017; Guillen et al., 2017; Septien-Hernandez et al., 2022).

4 Evaluation

We examined the secured proxy re-encryption technique's performance from a variety of perspectives, including key generation times, encryption and decryption times, and turnaround times. The next sections give the specifics of the experimental design and findings.

4.1 Experimental Setup

Android client application development was used to assess the effectiveness of the secure proxy re-encryption approach. Cloud storage, SRS, and users are some of the entities included in the proposed secure proxy re-encryption. We stored data in the cloud using firebase cloud storage. The SRS is implemented as a third-party server that generates the public/private key pairs and the re-encryption keys. NTRU encryption is used with APR2011_439_FAST parameters which offer 128 bits of security level. SRS is implemented on the computer having Intel Core i7-10800H @ 2.60GHz (12 CPUs) with 32 GB ddr4 3200MHz memory.

On three separate Android emulators with Qualcomm SM7250 Snapdragon 765 processors, Android applications are installed. The first emulator has a 2GB memory and 2 cores limit, the second has a 4GB memory and 4 cores limit, and the third has an 8GB memory and 6 cores limit. The application is developed on 26_2 API.

4.2 Results

Secure proxy re-encryption performance was assessed in terms of creation, encryption, decryption, and turnaround time. Below is a discussion of the findings for each of the evaluation criteria mentioned above.

4.2.1 Key Generation

The SRS is responsible for creating the private/public key pairs for the authorized users, as was already mentioned. However, the duration of the key generation process for systems with a high user density may have an impact on the system's overall performance. As a result, we evaluated the secure proxy re-encryption efficiency in terms of how long the key generation phase took for various user counts.
Fig 5.1 shows how long it takes to generate keys for 10, 100, 500, 1000, 5000, and 10,000 users.

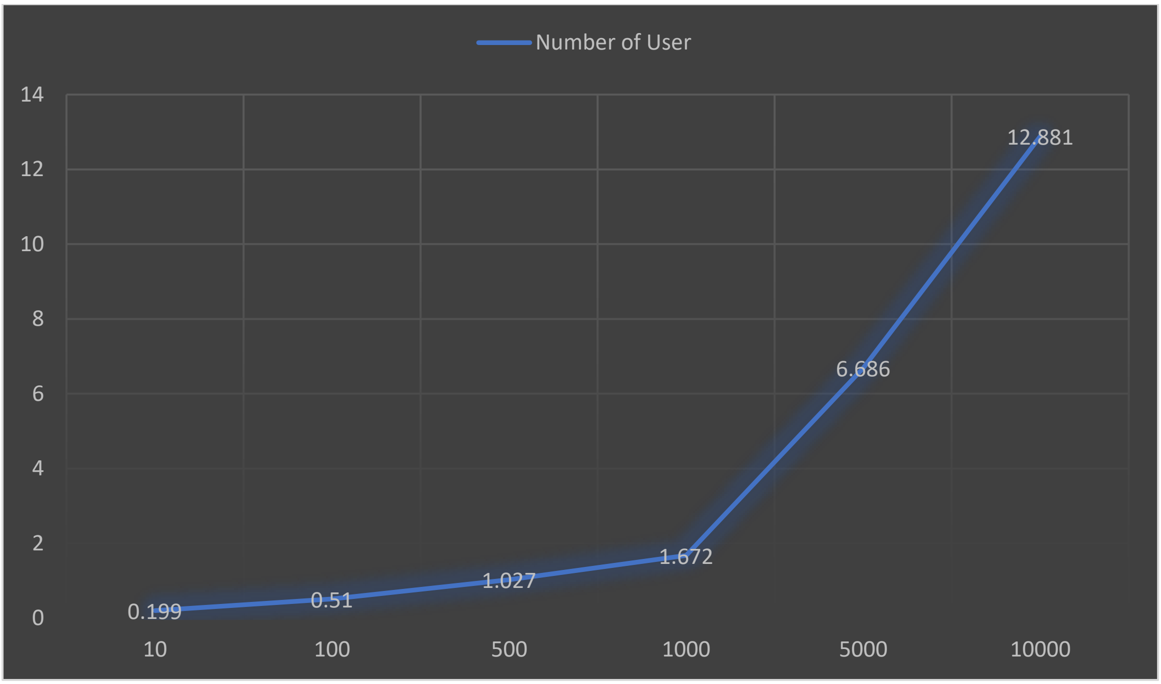

Fig 4.1 Time consumption for users

Figure 5.1 shows that it takes 0.51 seconds to produce keys for 100 users, whereas it takes 1.672 seconds to generate keys for 1000 users. Similarly, the time required to generate keys for 10,000 users is 12.881 seconds, which is also acceptable given a large number of users. Since these members-only sometimes join, the key creation time for them is likewise quite short, therefore creating keys for a single user is an efficient procedure.

4.2.2 Encryption and Decryption

To evaluate the effectiveness of the Android application, we tested it on three distinct devices. We have used a variety of metrics to evaluate system performance. Time spent during encryption, decryption, and turnaround. The time needed to finish the entire procedure is included in the turnaround time. Time is required to read a file, split it into separate 65-byte portions, encrypt it, and save it into the file using base 64 encoding. Similarly reading encrypted data from files using base64 decoding. After decrypting the data, all 65 bytes are combined to form a single meaningful piece of information. Remember that the time it takes to upload and download encrypted data is not included in the turnaround time because it heavily depends on your network connection.

T1 = reading file + encryption + base64 encoding + saving into files

T2 = reading from files + base 64 decoding + Decryption + combing data

The files which are used to measure the performance of the system are 50KB, 100KB, 200KB, 500KB, 1024KB, 1500KB, and 2048KB. Time consumption on the different specifications is given below. As we can see from Table 5.1, the time required for encryption of a 50KB file on a device with 2GB of RAM and 2 cores is 0.39 seconds, which is practically the same across all devices. A device with 4GB of RAM and 4 cores takes 0.41 seconds, and a device with 8GB of RAM and 6 cores takes 0.4 seconds. The 2048KB file continues to follow a similar trend. The smartphone with 2GB of RAM and 2 cores finished the test in roughly 8.27 seconds, followed by the device with 4GB of RAM and 4 cores in 8.733 seconds, and the device with the highest specifications, 8GB of RAM and 6 cores, in 9.112 seconds. All devices follow the same pattern during the decryption procedure.

Table 4.1 Encryption and decryption time among different devices.

File Size	**The device with 2GB ram and 2 cores**		**The device with 4GB ram and 4 cores**		**The device with 8GB ram and 6 cores**	
	Encryption	**Decryption**	**Encryption**	**Decryption**	**Encryption**	**Decryption**
50KB	0.39	0.41	0.41	0.37	0.405	0.392
100KB	0.462	0.402	0.547	0.492	0.467	0.495
200KB	1.036	0.846	0.995	0.843	1.011	0.892
500KB	2.309	2.055	2.494	2.07	2.323	2.014
1024KB	4.523	4.108	4.428	3.92	4.765	3.708
1500KB	6.187	5.986	6.524	5.780	6.404	5.973
2048KB	8.27	8.073	8.733	7.778	9.112	7.743

4.2.3 Turnaround Time

Table 5.1 only displays the encryption and decryption times for various file sizes across various devices, while table 5.2 show the values for turnaround time are much different and quite unsettling.

Table 4.2 Turnaround time among different devices.

File Size	**The device with 2GB ram and 2 cores**		**The device with 4GB ram and 4 cores**		**The device with 8GB ram and 6 cores**	
	T1	**T2**	**T1**	**T2**	**T1**	**T2**
50KB	1.201	0.997	1.244	1.076	1.446	1.053
100KB	1.846	2.058	2.228	1.942	2.203	2.069
200KB	4.056	5.03	3.7	5.067	4.211	5.219
500KB	11.216	18.142	10.628	20.021	11.619	18.504
1024KB	26.829	51.171	28.690	50.805	29.684	51.860
1500KB	41.289	90.067	42.745	90.023	41.762	90.253
2048KB	57.140	154.365	64.763	150.684	62.305	149.367

Here we can see that T2 is taking more time than T1. T2 involves the process of decryption and T1 involves the process of encryption but the results are the opposite. The T1 on the device with 8GB ram is 62.305 seconds for the 2048KB file and the T2 is 149.367 seconds which is very poor performance. The main reason behind this poor performance is handling and reading the huge number of files while decrypting. As we know the encryption process encrypts the data into parts of 65 bytes and saves them into a different file. While encrypting the 2048KB file it will create 32,264 new encrypted files and this will create the performance bottleneck for the decryption process. The device with 2GB ram and 2 cores becomes unresponsive for a few seconds in both encrypting and decrypting process when the file size increase from 1024KB.

Here, Table 5.2 clears that T2 is taking longer than T1. T1 involves encryption and T2 involves decryption, but the outcomes are dramatically opposed. T1 for a 2048KB file on a device with 8GB of RAM is 62.305 seconds, and T2 is 149.367 seconds, which is a very poor performance. The handling and reading of the enormous number of files during decrypting is the primary cause of this poor performance. As far as we are aware, the encryption mechanism divides the data into 65-byte segments and saves them to separate files. 32,264 additional encrypted files will be created while encrypting a 2048 KB file, which may slow down the decryption process' speed. When encrypting and decrypting files larger than 1024 KB, a device with 2GB of RAM and 2 CPUs becomes unresponsive for a short period.

5 Conclusions and Future Work

The purpose of this study is to identify potential weaknesses in the previously suggested proxy re-encryption approaches. There are a lot of conclusions from prior relevant work that can be found with the use of a literature review. The earlier suggested techniques are vulnerable to quantum attacks. Some of the schemes additionally demand that the owner stay online for the entire 24 hours to provide the user access. The recently adopted approach promises to use SRS and quantum-proof proxy re-encryption techniques so that the owner does not need to remain online constantly. This strategy is put into practice using the Android platform. Due to the creation of many encrypted files and high key sizes of NTRU cryptography. On the decryption end, this approach suffers from a severe bottleneck. In the future, we will try to increase the performance of this methodology by improving file management.

6 References

Ali, M., Abbas, A., Khan, M. usman S.K., Khan, S.U., 2021. SeSPHR: A Methodology for Secure Sharing of Personal Health Records in the Cloud. IEEE Trans. 9, 347–359.

Ali, M., Madani, S.A., Kiah, M.L.M., Khan, A.N., Shamshirband, S., 2014. Incremental proxy re-encryption scheme for mobile cloud computing environment. Springer 68, 624–651.

Ateniese, G., FU, kevin, Matthew, G., Susan, H., 2006. Improved proxy re-encryption schemes with applications to secure distributed storage. ACM 9, 1–30.

Bernstein, D.J., Lange, T., 2017. Post-quantum cryptography. Nature 549, 188–194. https://doi.org/10.1038/nature23461

Chen, Y., Tygar, J.D., Tzeng, W.-G., 2011. Secure group key management using uni-directional proxy re-encryption schemes. IEEE 1952–1960.

Cheng, H., Dinu, D., Großschädl, J., Rønne, P.B., Ryan, P.Y.A., 2020. A Lightweight Implementation of NTRU Prime for the Post-quantum Internet of Things, in: Laurent, M., Giannetsos, T. (Eds.), Information Security Theory and Practice. Springer International Publishing, Cham, pp. 103–119. https://doi.org/10.1007/978-3-030-41702-4_7

Computer Security Division, I.T.L., 2020. PQC Third Round Candidate Announcement | CSRC [WWW Document]. CSRC NIST. URL https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement (accessed 3.28.22).

Computer Security Division, I.T.L., 2017. Round 3 Submissions - Post-Quantum Cryptography | CSRC | CSRC [WWW Document]. CSRC NIST. URL https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions (accessed 3.27.22).

Guillen, O.M., Pöppelmann, T., Bermudo Mera, J.M., Bongenaar, E.F., Sigl, G., Sepulveda, J., 2017. Towards post-quantum security for IoT endpoints with NTRU, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2017. Presented at the Design, Automation Test in Europe Conference Exhibition (DATE), 2017, pp. 698–703. https://doi.org/10.23919/DATE.2017.7927079

Hoffstein, J., Pipher, J., Silverman, J.H., 1998. NTRU: A ring-based public key cryptosystem, in: Buhler, J.P. (Ed.), Algorithmic Number Theory. Springer, Berlin, Heidelberg, pp. 267–288. https://doi.org/10.1007/BFb0054868

Jafari, M., Safavi-Naini, R., Sheppard, N.P., 2011. A rights management approach to protection of privacy in a cloud of electronic health records, in: Proceedings of the 11th Annual ACM Workshop on Digital Rights Management, DRM '11. Association for Computing Machinery, New York, NY, USA, pp. 23–30. https://doi.org/10.1145/2046631.2046637

Kandukuri, B., Paturi, R., 2009. Cloud Security Issues. IEEE 200.

Khalid, A., McCarthy, S., O'Neill, M., Liu, W., 2019. Lattice-based Cryptography for IoT in A Quantum World: Are We Ready?, in: 2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI). Presented at the 2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI), pp. 194–199. https://doi.org/10.1109/IWASI.2019.8791343

Ladd, T.D., Jelezko, F., Laflamme, R., Nakamura, Y., Monroe, C., O'Brien, J.L., 2010. Quantum computers. Nature 464, 45–53. https://doi.org/10.1038/nature08812

Lange, T., 2015. Initial recommendations of long-term secure post-quantum systems. PQCRYPTOEU Horiz. 2020 ICT-645622.

Lanyon, B.P., Weinhold, T.J., Langford, N.K., Barbieri, M., James, D.F.V., Gilchrist, A., White, A.G., 2007. Experimental Demonstration of a Compiled Version of Shor's Algorithm with Quantum Entanglement. Phys. Rev. Lett. 99, 250505. https://doi.org/10.1103/PhysRevLett.99.250505

Li, M., Yu, S., Zheng, Y., Ren, K., Lou, W., 2013. Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. IEEE Trans. Parallel Distrib. Syst. 24, 131–143. https://doi.org/10.1109/TPDS.2012.97

Liang, X., Cao, Z., Lin, H., Shao, J., 2009. Attribute based proxy re-encryption with delegating capabilities, in: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS '09. Association for Computing Machinery, New York, NY, USA, pp. 276–286. https://doi.org/10.1145/1533057.1533094

O. Saarinen, M.-J., 2020. Mobile Energy Requirements of the Upcoming NIST Post-Quantum Cryptography Standards, in: 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud). Presented at the 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 23–30. https://doi.org/10.1109/MobileCloud48802.2020.00012

Pecarina, J., Pu, S., Liu, J.-C., 2012. SAPPHIRE: Anonymity for enhanced control and private collaboration in healthcare clouds, in: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings. Presented at the 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, pp. 99–106. https://doi.org/10.1109/CloudCom.2012.6427488

Pirandola, S., Pirandola, S., Andersen, U.L., Banchi, L., Berta, M., Bunandar, D., Colbeck, R., Englund, D., Gehring, T., Lupo, C., Ottaviani, C., Pereira, J.L., Razavi, M., Shaari, J.S., Shaari, J.S., Tomamichel, M., Tomamichel, M., Usenko, V.C., Vallone, G., Villoresi, P., Wallden, P.,

2020. Advances in quantum cryptography. Adv. Opt. Photonics 12, 1012–1236. https://doi.org/10.1364/AOP.361502

Septien-Hernandez, J.-A., Arellano-Vazquez, M., Contreras-Cruz, M.A., Ramirez-Paredes, J.-P., 2022. A Comparative Study of Post-Quantum Cryptosystems for Internet-of-Things Applications. Sensors 22, 489. https://doi.org/10.3390/s22020489

Tamilmani, K., Rana, N.P., Dwivedi, Y.K., 2018. Mobile Application Adoption Predictors: Systematic Review of UTAUT2 Studies Using Weight Analysis, in: Al-Sharhan, S.A., Simintiras, A.C., Dwivedi, Y.K., Janssen, M., Mäntymäki, M., Tahat, L., Moughrabi, I., Ali, T.M., Rana, N.P. (Eds.), Challenges and Opportunities in the Digital Era. Springer International Publishing, Cham, pp. 1–12. https://doi.org/10.1007/978-3-030-02131-3_1

Tsiounis, Y., Yung, M., 1998. On the security of ElGamal based encryption, in: Imai, H., Zheng, Y. (Eds.), Public Key Cryptography. Springer, Berlin, Heidelberg, pp. 117–134. https://doi.org/10.1007/BFb0054019

What is Amazon S3? - Amazon Simple Storage Service [WWW Document], n.d. URL https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html (accessed 4.6.22).

Xhafa, F., Feng, J., Zhang, Y., Chen, X., Li, J., 2015. Privacy-aware attribute-based PHR sharing with user accountability in cloud computing. J. Supercomput. 71, 1607–1619. https://doi.org/10.1007/s11227-014-1253-3

Yandong, Z., Yongsheng, Z., 2012. Cloud computing and cloud security challenges, in: 2012 International Symposium on Information Technologies in Medicine and Education. Presented at the 2012 International Symposium on Information Technologies in Medicine and Education, pp. 1084–1088. https://doi.org/10.1109/ITiME.2012.6291488