Constructing a Model Combining Zalo and End-to-End Encryption for Application in Digital Transformation

Cao Thanh Vinh, Pham Van Huong

***Abstract*— This paper proposes and implements a model combining Zalo and end-to-end encryption to apply in digital transformation. The main idea is to promote the strengths of the Zalo social network to exchange information and manage work but ensure information confidentiality for both the server and man-in-the-middle attacks based on the end-to-end encryption model. End-to-end encryption is right at the client, so only the encrypted data is stored on the server and therefore the server cannot decrypt it. This solution can use international standard cryptographic algorithms or those of the Vietnam Government Information Security Commission. The solution model has been developed, deployed, and tested at the Academy of Cryptography Techniques. Our proposed system serves as an end-to-end encryption and adapter used to encrypt text data and files in various social networks. Within the scope of this paper, we deploy a test solution with Zalo.**

Tóm tắt **— Bài báo này đề xuất và triển khai mô hình kết hợp Zalo và mã hóa đầu cuối để ứng dụng trong chuyển đổi số. Ý tưởng chính là phát huy điểm mạnh của mạng xã hội Zalo để trao đổi thông tin, điều hành công việc nhưng đảm bảo bí mật thông tin đối với cả máy chủ và tấn công xen giữa dựa trên mô hình mã hóa đầu cuối. Mã hóa đầu cuối ngay tại máy trạm nên chỉ có dữ liệu mã lưu trữ trên máy chủ và do đó máy chủ cũng không thể giải mã được. Giải pháp này có thể dùng các thuật toán mật mã chuẩn quốc tế hoặc dùng mật mã riêng của Ban Cơ yếu Chính phủ. Mô hình giải pháp đã được phát triển, cài đặt, thử nghiệm tại Học viện Kỹ thuật mật mã. Hệ thống đề xuất thực hiện như một bộ mã hóa và chuyển mạch đầu cuối dùng để bảo mật dữ liệu văn bản và file trong các mạng xã hội khác nhau. Trong phạm vi bài báo, nhóm tác giả triển khai giải pháp thử nghiệm với Zalo.**

Keyword— Zalo API; end-to-end encrption; digital transformation.

Từ khóa— Zalo API; mã hóa đầu cuối; chuyển đổi số.

I. INTRODUCTION

Nowadays, with the explosive development of the 4th technological revolution, especially the rapid development of social networking platforms, it has helped connect people with each other anytime, anywhere. Through social networks, people can share information, images, sounds, etc. find friends, connect with others, etc. Everyone can easily access any social network via mobile, computer, etc. Social networks also often provide functions to help users chat and exchange information through their platforms. This is also a channel that helps businesses send notices to customers conveniently, quickly, and at low cost. Understanding this need, most social networks provide APIs to help agencies and businesses integrate into their applications. However, the APIs that social networks provide often have no guarantees about data security. Therefore, notifications that agencies and businesses send to users through APIs of social networks are at risk of being leaked and completely controlled by the above social networks. APIs of social networks are usually only provided to businesses.

In Vietnam, along with the international development trend, social networks are also quite commonly used. According to social networks, Facebook and Zalo have the largest number of users [1]. Both of these social networks provide APIs for business accounts to

This manuscript is received on December 1, 2023. It is commented on December 16, 2023 and is accepted on December 22, 2023 by the first reviewer. It is commented on December 27, 2023 and is accepted on December 29, 2023 by the second reviewer.

send messages to their customers. However, Zalo better supports state agencies and organizations. Specifically, implementing the government's digital transformation policy, Zalo launched the Zalo Digital Transformation program to provide OA accounts (Official Accounts) for agencies for propaganda purposes. As of the end of December 2021, across the country there were more than 6,000 Zalo pages of ministries, provinces/cities, localities, police, health, education, electricity, water, etc. established, an increase of 160% compared to 2020 [2]. To serve the national digital transformation goal, Zalo provides full APIs and chatbots to help agencies and organizations propagate and receive feedback from citizens. Organizations can fully use this channel to send notifications to lower levels quickly, effectively, and at a low cost. However, the APIs and chatbots provided by Zalo do not have security solutions, posing a risk of information insecurity, especially with important information.

Currently, Zalo also provides end-to-end encryption solutions for users. However, this is still a beta version, so the end-to-end encryption feature can only be set up manually for each person. In addition, the use of end-to-end encryption completely depends on Zalo's policies and we cannot control the security solutions provided by Zalo. Compared to Zalo's solution, our proposed solution has the following advantages and disadvantages:

Advantages: proactive about cryptography (cryptographic algorithms, key agreement, random number generation, key derivation, authentication algorithms), solution used for large groups of people (end-to-end encryption solution of Zalo is only used for two people),

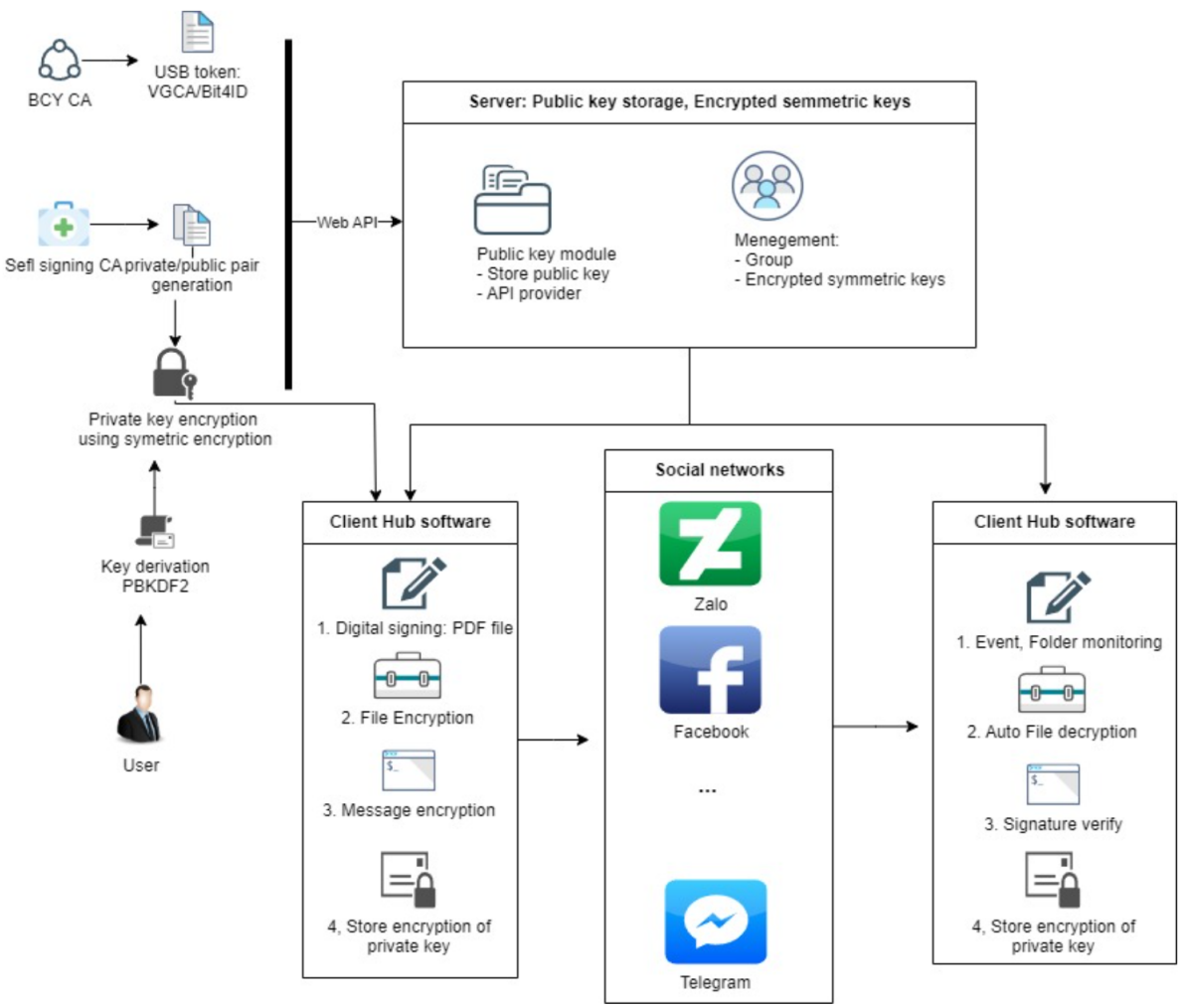

Figure 1. The overall solution model

integrated with the PDF file signing software of the Vietnam Government Information Security Commission, can be used with other social networks, and can be used to secure cloud storage data. In the case of integration with digital signing software on PDF files, first, the PDF file is signed by the user's USB token device. Then, this file is encrypted with symmetric cryptography. Generate the HMAC authentication hash on the file's cipher. HMAC is attached to the file header. At the receiving site, the received file is split into the cipher data and HMAC. Generate a new HMAC on the cipher data; compare two HMACs; If equal, authentication is successful. After successful authentication, decrypt the file's ciphertext to get the original PDF file. Use digital signing software on PDF files to verify signatures using the user's USB token.

Disadvantages: encryption/decryption is not "transparent" to the user, and some additional operations are required. The reason is that the Zalo developer only provides APIs to a certain extent, so it cannot intervene further.

In this paper, the authors have proposed a solution to build a communication system using Zalo combined with end-to-end encryption. The model proposed by the team will help take advantage of Zalo's infrastructure while still ensuring the requirements for controling information security solutions.

II. PROPOSING A MODEL COMBINATION OF ZALO AND END-TO-END ENCRYPTION

The purpose of our solution is to take advantages of social networks to perform some tasks in digital transformation while still ensuring confidentiality, data integrity, and server independence.

The system serves as an end-to-end encryption and adapter used to encrypt text data and files in various social networks. Within the scope of this paper, we deploy a test solution with Zalo.

The model of the information transmission system combining Zalo and end-to-end encryption in digital transformation is shown in Figure 1.

A. Cryptographic scheme

Key management:

There are two ways to distribute keys: using USB tokens (VGCA/Bit4ID of the Vietnam Government Information Security Commission) or self-generated private/public key pairs. With method 1, the USB token device for computer (PKI sim on the phone) is issued by the Vietnam Government Information Security Commission, used when there are high safety and security requirements because the private key and decryption algorithm, digital signature are stored and executed in the device. The private key is stored and protected in the device and cannot be read by other software. The public key is stored on the key management server. With method 2, when there is no USB token device (for computers) or PKI Sim (for phones), we use a key pair. Method 2 is more flexible and can be used on any device, but is less secure because the private key file must be stored on the device. Therefore, in method 2, the private key is also encrypted using a symmetric algorithm (e.g. AES) with the encryption key derived from the user's password.

With a USB token device: We get the user's public key and save it to the server. The private key is protected and stored on the USB token device.

With key file: The public key is also gotten and saved on the server. The private key is encrypted by a symmetric algorithm with key K_{0_i}, and saved on the computer.

Derive the key K_{0_i} from the password

$$K_{0_i} = PBKDF2(PRF,\ Pwd_i, Salt, c, dkLen) \quad (1)$$

where,

- PRF is a pseudorandom function.
- Pwd_i is the password of the user *i*.
- Salt is a sequence of bits.
- *c* is the number of iterations desired.
- dkLen is the desired bit-length of the derived key.

Encrypt the K_{pri_i} private key and save it at the client.

$$K_{pri_i}' = Encrypt_{K_{0_i}}^{AES-256,\ OFB/CTR}(K_{pri_i}) \quad (2)$$

Encrypt messages:

With M is the plain text:

Generate the symmetric key K_1 for each group according to the Equation (3), (4).

$$K_1 = Hdrbg(l1) \quad (3)$$

$$IV_1 = Hdrbg(l2) \quad (4)$$

where,

- Hdrbg is the Hash-based deterministic random bit generator function. This function is implemented by the Vietnam Government Information Security Commission and integrated into the OpenSSL library. Moreover, this function can use SHA 256/384/512 [10].
- $l_1 = 32$
- $l_2 = 16$.

For each user *i*, encrypt (K_1, IV_1) to $(K_1', IV_1')_i$ and save to the server.

$$(K_1', IV_1')_i = Encrypt_{Kpub_i}^{RSA-2048/3072/4096,\ OAEP}(K_1, IV_1) \quad (5)$$

Where $Kpub_i$ is the public key of user i.

Encrypt the message with K_1, IV_1 and send:

$$M' = Encrypt_{K_1,IV_1}^{AES-256,OFB/CTR}(M) \quad (6)$$

Decrypt the message:

Get the GroupID, UserID of the current user. Get the ciphertext of K_1 as K_1' from the server via WebAPI.

With USB token (VGCA/Bit4ID), decrypt K_1' from the device to be K_1. With key file:

Derive the K_{0_i} key from the user password:

$$K_{0_i} = PBKDF2(PRF, Pwd_i, Salt, c, dkLen) \quad (7)$$

Decrypt private key:

$$K_{pri_i} = Decrypt_{K_{0_i}}^{AES-256,OFB/CTR}(K_{pri_i}') \quad (8)$$

Decrypt the message key and *IV*:

$$(K_1, IV_1) = Decrypt_{Kpri_i}^{RSA-2048/3072/4096,\ OAEP}((K_1', IV_1')_i) \quad (9)$$

Decrypt messages with K_1

$$M = Decrypt_{(K_1,IV_1)}^{AES-256,OFB/CTR}(M') \quad (10)$$

Encrypt files:

Each file is encrypted and authenticated with its own, randomly generated encryption key, authentication key, and *IV*.

Randomly generate K_1, K_2, IV_1:

$$(K_1, K_2, IV_1) = (Hdrbg(l_1),\ Hdrbg(l_2),\ Hdrbg(l_3)) \quad (11)$$

where,

- $l_1 = l_2 = 32$ (because the length of K_1 is equal to the length of K_2, and equal to 32 bytes).
- $l_3 = 16$ (because the length of IV_1 is equal 16 bytes).

Get all public keys of users in the group and encrypt K_1, K_2, IV_1.

$$(K_1', K_2', IV_1')_i = Encrypt_{Kpub_i}^{RSA-2048/3072/4096,\ OAEP}(K_1, K_2, IV_1) \quad (12)$$

Encrypt data D of the file *D'*

$$D' = Encrypt_{K_1,IV_1}^{AES-256,OFB/CTR}(D) \quad (13)$$

Generate message authentication code H of the file:

$$H = HMAC_{K_2}^{SHA\ 256/512}(D') \quad (14)$$

Create encryption file according to the structure: Header contains $UserID_i$, $(K_1', K_2', IV_1')_i$ and *H*; The body part contains cipher of data D'. The structure of the encryption file has been carried out according to the regulations of the Cipher Committee, but due to confidentiality, we do not present it here.

Verify and decrypt files:

For each encryption file placed in the configured code folder to synchronize between client software and social networks.

Read the encryption file, separate the Header and code data *D*'. Analyze and retrieve the ciphertext $(K_1', K_2', IV_1')_i$ corresponding to the user's UserID i.

Decrypt K_1, K_2, IV_1:

With USB token (VGCA/Bit4ID), decrypt $(K_1', K_2', IV_1')_i$ from device.

With key file:

Derive the key K_{0_i} from the user password:

$$K_{0_i} = PBKDF2(PRF, Pwd_i, Salt, c, dkLen) \quad (15)$$

Decrypt private key:

$$K_{pri_i} = Decrypt_{K_{0_i}}^{AES-256,\ OFB/CTR}(K_{pri_i}') \quad (16)$$

Decrypt the message key:

$$(K_1, K_2, IV_1) = Decrypt_{K_{pri_i}}^{RSA-2048/3072/4096,\ OAEP}((K_1', K_2', IV_1')) \quad (17)$$

File validation:

Calculate $H' = HMAC_{K_2}^{SHA\ 256,512}(D')$

and compare H' with H to verify.

If authentication is successful ($H = H'$), decrypt the file data:

$$D = Decrypt_{K_1, IV_1}^{AES-256,\ OFB/CTR}(D') \quad (18)$$

B. The software structure

This secure communication system includes 2 softwares:

- Message Sending Module: this software uses organization's social network account to send notifications and official dispatches to individual accounts. It encrypts messages before sending them via social networks using APIs provided by social networks.

- Message Receiving Module: this software uses social network accounts of individuals to receive notifications and official dispatches from organizations. It receives informations from the social networking applications, decrypt and returns them to users.

- Server Module: stores the public key of all users and ciphers of keys and IV for symmetric encryption. This module provides Web API for the two above modules.

To ensure the safety of the system, authors built a security solution as follows: each user participating in the system will have a public key and a private key. When registering to receive notifications of the organization's secure communication system, users send the public key to the Message Sending Software. Message Sending Software stores the public key, and gives the user an identifier. When we need to send a message, Message Sending Software will randomly generate a session key "K1" and use it to encrypt the information. Then, use each user's public key to encrypt "K1", attach ciphertexts of "K1" according to the user's identification code to the beginning of the encrypted messages and send it by using Zalo's API. Message Receiving Software will read the message content of the Zalo application, using its secret key along with the provided identifier to decrypt the message.

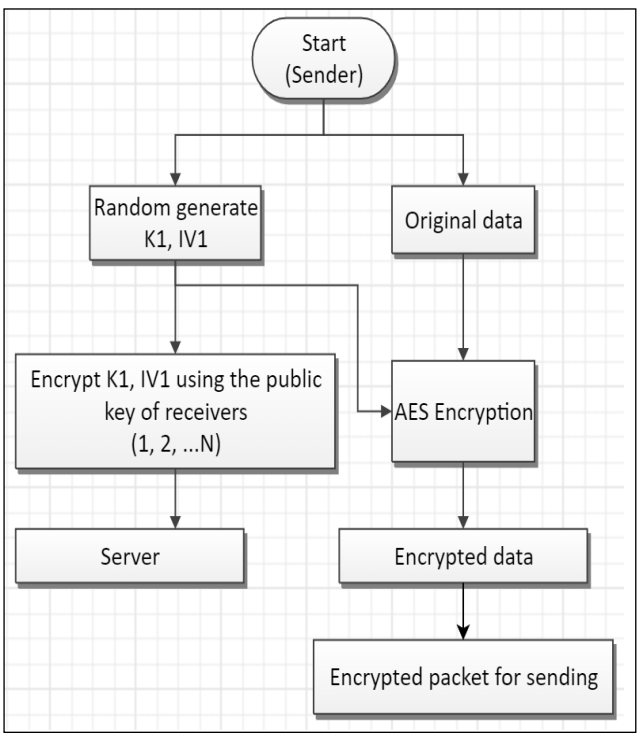

Figure 2. Model of creating encrypted packets to send

Messages transmitted through the system may be in the form of text files or text messages. The process of creating encrypted packets in Message Sending Software is carried out according to the model in Figure 2.

The process of data decryption in the Message Receiving Software is performed following to the model in Figure 3.

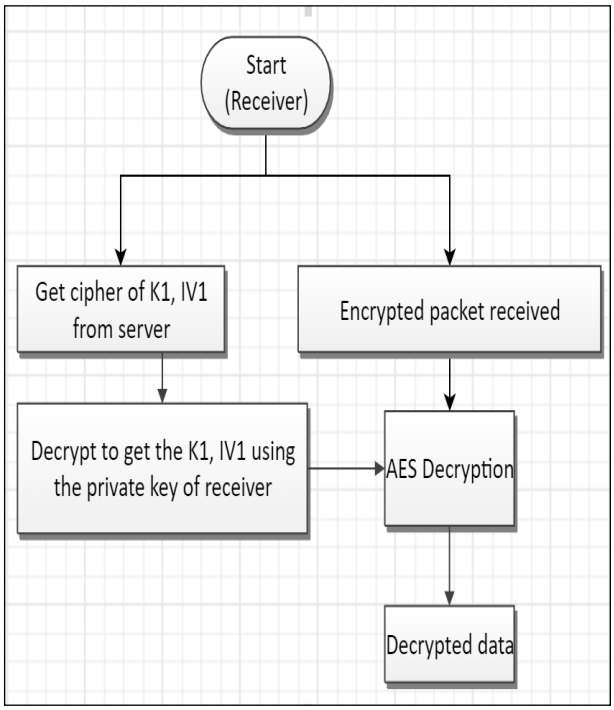

Figure 3. Model for data decryption from received encrypted packets

III. BUILDING A SECURITY COMMUNICATION SYSTEM COMBINING ZALO AND END-TO-END ENCRYPTION

A. Research Zalo's API system

Zalo's API system [3] (also known as Zalo OA OpenAPI) is a set of tools to support the operation of Zalo OA accounts for organizations and businesses (Zalo Official Account), through integrating Zalo OA with the internal system of them. OA OpenAPI helps organizations and businesses operate automatically, supports the operation of multiple OA accounts, and especially helps centralize information and data.

To manage Zalo OA using OA OpenAPI, organizations and businesses need to integrate Zalo OA with the system (server) by granting permissions to the Zalo App application. Organizations and businesses will call APIs corresponding to OA management rights and interact with users through the application. Permission groups of the OA OpenAPI system include: Messaging, Voice Calling, Posting, Management.

In Zalo API system, organizations and businesses can use the Messaging API group to send messages to users through their self-developed applications. Messages from the OA account will be sent to individual users through user ID (UID). Messages (UID messages) are categorized into three types: Advisory Messages, Transactional Messages, and Promotional Messages. Among these, Transactional and Promotional messages need to adhere to the specified format by Zalo; Advisory Messages allow sending messages with optional content (text advisory messages, Advisory Messages with images, or files). For text Advisory Messages, Zalo permits sending messages with a maximum length of 2000 characters. For Advisory messages with images or files, Zalo only supports attachments in standard formats (supported image formats: jpg and png; supported file formats: pdf, doc, and docx).

Therefore, with the content-control policies of the Zalo API system, Zalo does not allow sending files that are encrypted or files in inappropriate formats. Thus, to ensure the confidentiality of information sent through the Zalo API, authors proposed building a technical solution to send encrypted information via the Zalo API.

B. Secure Message Sending Solution using Zalo API

The Secure Message Sending Solution proposed by the authors is built upon Zalo's Advisory Message API with file attachments. At the sender's end, messages and files will be encrypted and then converted from binary data into text data using the base64 format. The base64-encoded text data is stored in a .docx file before being sent through the Zalo API. On the recipient's side, the software will read the text data from the received .docx file, convert it back to binary data, and then invoke the decryption function to obtain the original data.

In addition, to distinguish between text messages and file messages, we add 04 characters to the beginning of the .docx file used to store the base64 text data of the coded message. As follows:

+ "text": text message.

+ "file": file message.

The operational flow diagram of the secret communication process is shown in Figure 4.

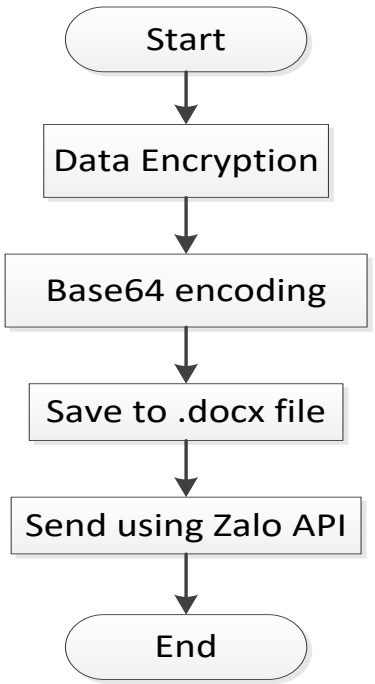

Figure 4. Flow chart of the secret message sending process

Specifically, the operation of the confidential information transmission process includes these following steps:

- *Step 1:* Firstly, before sending a message via Zalo API, the message will be encrypted, the data that needs to be encrypted can be a file, or a text message.

- *Step 2:* After encryption, the data will be converted into base64 before saving to the .docx file.

- *Step 3:* Save the base64 text of the encrypted data (with 4 characters to distinguish the type of message) into a .docx file.

- *Step 4:* Send the .docx file containing the base64 text of the encrypted data via Zalo API and finish.

The flow chart of the process of receiving confidential information is shown in Figure 5.

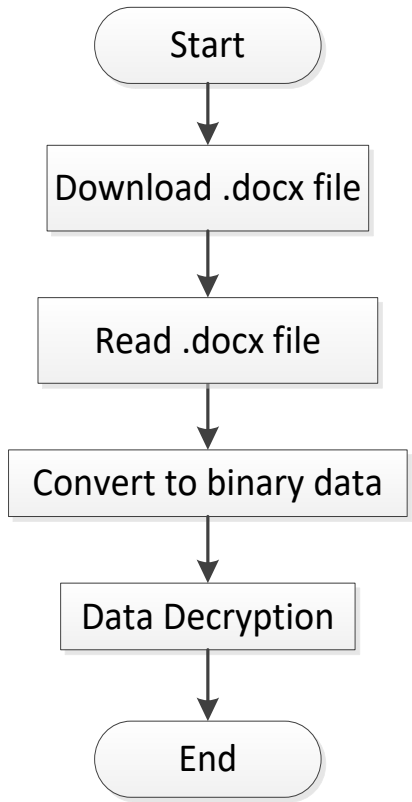

Figure 5. Flow chart of receiving confidential information process

The operation of receiving confidential information process will be opposite to the transmitting confidential information process, specifically as follows:

- *Step 1:* users open Zalo to download the .docx file (file containing base64 text of encrypted data).

- *Step 2:* users open Message Receiving Software, then, open the just downloaded .docx file to read the file content.

- *Step 3:* the content read from the .docx file after removing 4 characters used to distinguish the type of message will be converted into binary format.

- *Step 4:* decrypt data to get the original message.

C. Building a Cryptography Library

To facilitate the deployment of the applications and ensure the security of cryptographic operations, authors developed a

shared cryptography library for the entire system, also constructed this cryptography library using the C programming language to ensure robust security against reverse engineering. Main API functions within the cryptography library are as follows:

1. Encrypt File Function

This function is responsible for encrypting files before sending them to users in the group. The prototype of the function is built as follows:

```
int Zalo_EncryptFile
(const wchar_t *inputFileName,
const wchar_t *outputFileName,
unsigned int *listUserID,
const char *listUserPubKey,
unsigned int numUser);
```

Parameters:

`inputFileName`	Input File Name
`outputFileName`	Encrypted Output File Name
`listUserID`	List of User IDs
`listUserPubKey`	List of User Public Keys (in PEM format)
`numUser`	Number of Users

2. Decrypt File Function

This function is designed to decrypt a received file. The prototype of the function is built as follows:

```
int Zalo_DecryptFile
(const wchar_t *inputFileName,
const wchar_t *outputFileName,
unsigned int userID,
const char *userPrivKey,
unsigned int userPrivKeyLen);
```

Parameters:

`inputFileName`	Encrypted Input File Name
`outputFileName`	Decrypted Output File Name
`userID`	User ID
`userPrivKey`	Private Key (in PEM format)
`userPrivKeyLen`	Private Key Length

3. Encrypt Message function

This function encrypts messages before sending them to users in the group. The function prototype is built as follows:

```
int Zalo_EncryptMessage
(const unsigned char *inMes,
unsigned int inMesLen,
unsigned char *outMes,
unsigned int *outMesLen,
unsigned int outMesMaxLen,
unsigned int *listUserID,
const char *listUserPubKey,
unsigned int numUser);
```

Parameters:

`inMes`	Input Message
`inMesLen`	Input Message Length
`outMes`	Encrypted Output Message
`outMesLen`	Encrypted Message Length

outMesMaxLen	Max Encrypted Message Length
listUserID	List of User IDs
listUserPubKey	List of Public Keys (in PEM format)
numUser	Number of Users

4. Decrypt Message Function

This function used for decryption received messages. The function prototype is built as follows:

```
int Zalo_DecryptMessage
(const unsigned char *inMes,
unsigned int inMesLen,
unsigned char *outMes,
unsigned int *outMesLen,
unsigned int outMesMaxLen,
unsigned int userID,
const char *userPrivKey,
unsigned int userPrivKeyLen);
```

Parameters:

inMes	Encrypted Input Message
inMesLen	Encrypted Message Length
outMes	Decrypted Output Message
outMesLen	Decrypted Message Length
outMesMaxLen	Max Decrypted Message Length
userID	User ID
userPrivKey	Private Key (in PEM format)
userPrivKeyLen	Private Key Length

D. System Implementation and Deployment

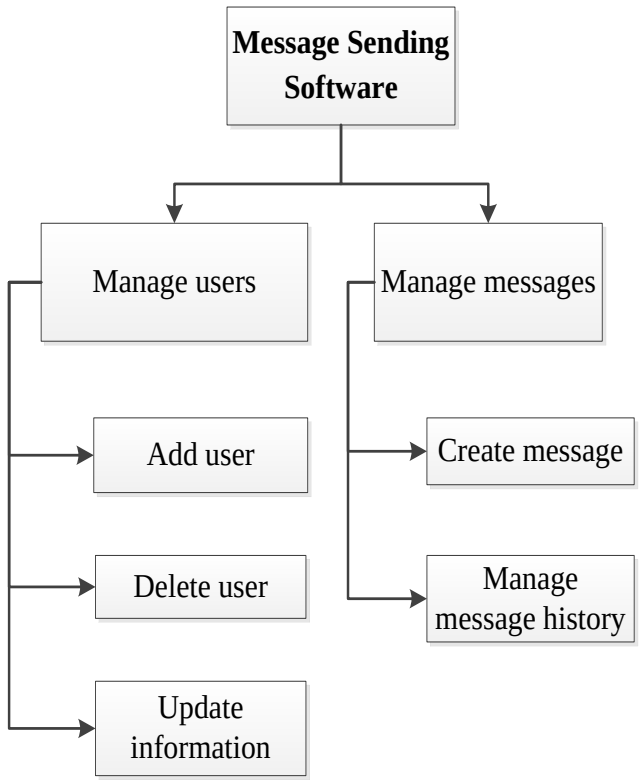

Figure 6. Functional decomposition model of Message Sending Software

On the general model basis of the system (Figure 1), we implemented a construction of a system consisting of two software components: the Message Sending Software and the Message Receiving Software.

1. Building the Message Sending Software

For the Message Sending Software, this will be managed by the personnel responsible for the operation and management of the official account (OA) of the organization. The main functionalities of this software are described in Figure 6.

Message Sending Software is designed to allow administrative personnel to compose announcements, official documents, and send them to end-to-end users within the system. To facilitate convenient operation, this software will

be developed on the Windows platform. To send messages via Zalo, it invokes APIs provided by Zalo, for cryptographic processing, it calls APIs from the cryptographic library.

2. Building the Message Receiving Software

For the Message Receiving Software, this will be used by end-to-end users of the system. The main functionalities of this software are illustrated in Figure 7.

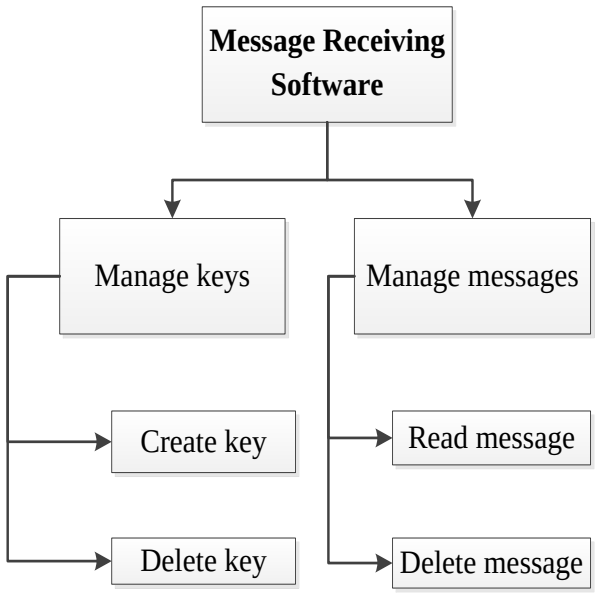

Figure 7. Functional Decomposition Model of the Message Receiving Software

The Message Receiving Software's primary function is to help end-to-end users receive messages from the organization's official account (OA) through their personal Zalo accounts. To facilitate convenient, timely, and user-friendly message reception, we developed the Message Receiving Software on a mobile environment. This software will be installed on the personal phones of users, and utilize their Zalo account information for system registration.

When registering, the software generates 2 keys, encrypts the private key, store it on the user's device, and sends the public key along with user information to the system. The cryptographic processing of the software will call APIs from the cryptographic library. As Zalo does not provide APIs for developing regular Zalo accounts, the message reception processing from the Zalo application will be user-initiated. Users actively download a .docx file containing base64-encoded text of encrypted data and upload it into the software.

IV. EXPERIMENT

After successfully building the system, we conducted testing at the Department of Information Technology, Academy of Cryptographic Technology. Cryptographic criteria in the experiment are summarized in Table 1.

TABLE 1. SUMMARY OF CRYPTOGRAPHIC CRITERIA OF THE SYSTEM

Component	Specification
Public Key Cryptography	RSA OAEP Key Length 2048 bit
Symmetric Key Cryptography	AES Mode OFB/CTR Key Length 256 bit
Hash Function	SHA 256

A.Experimental Execution

The message management function is used to create, encrypt, and push messages to Zalo through Web API. The group management function is used to generate session keys (K_1, IV_1), encrypt (K_1, IV_1), and push them to the server. The user management function is used to get the public key from USB token devices or generate a key pair and push the public key to the server as shown in Figure 11.

Then, in Zalo, users will receive a .docx file containing base64 text of the encrypted data (Figure 8).

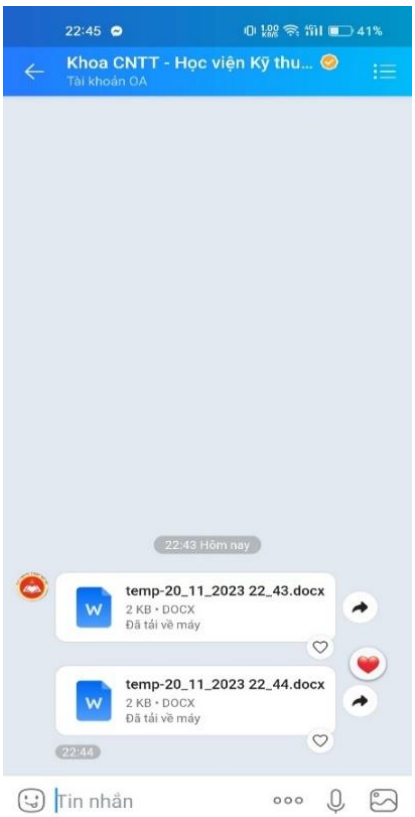

Figure 8. Zalo File Reception Interface

Next, users upload the received .docx file into the message receiving application for decryption (Figure 9). When decrypting successfully, the application will display a result notification to the user and show the message dialog on the screen (Figure 10).

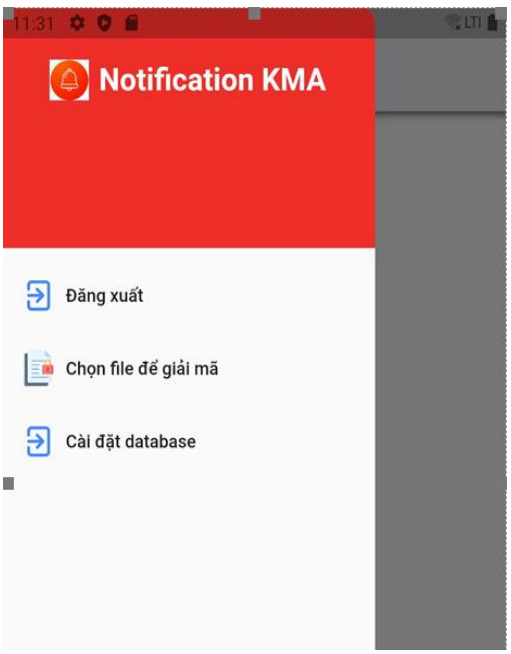

Figure 9. File Decryption Selection Interface of the Message Receiving Software

Figure 10. Message Interface of the Message Receiving Software

Authors have conducted multiple tests, and the results prove that the system operates stably, meeting the specified requirements.

B. Security evaluation

To evaluate the security of the solution, we have done the above experiments and analyze the results in Table 2.

TABLE 2. SECURITY ANALYSIS

Analysis items	**Result comparision**	
	Zalo	**Our solution**
Attacking data on the server	Messages and files sent and stored on the Zalo server are clear data, so they completely control user data.	Messages and files are encrypted right at the client. The server only stores the ciphertext. Because the server does not have the private key, it cannot decrypt to get the symmetric key to decrypt the messages and files.
Man-in-the-middle attack	Cannot prevent man-in-the-middle attacks unless Zalo incorporates a VPN solution.	Resists man-in-the-middle attacks.
Stealing messages on Zalo	Other users can completely peek at Zalo messages	Information sent via Zalo is encrypted so it cannot be viewed. To view, users must have a private key.
Attack private key		With a USB token/PKI SIM, the private key file is stored right in the device; Software, operating systems, and other software cannot read it. In the case of a self-generated key pair, the private key file is also encrypted and only stored ciphers on the server, so it is highly secure.

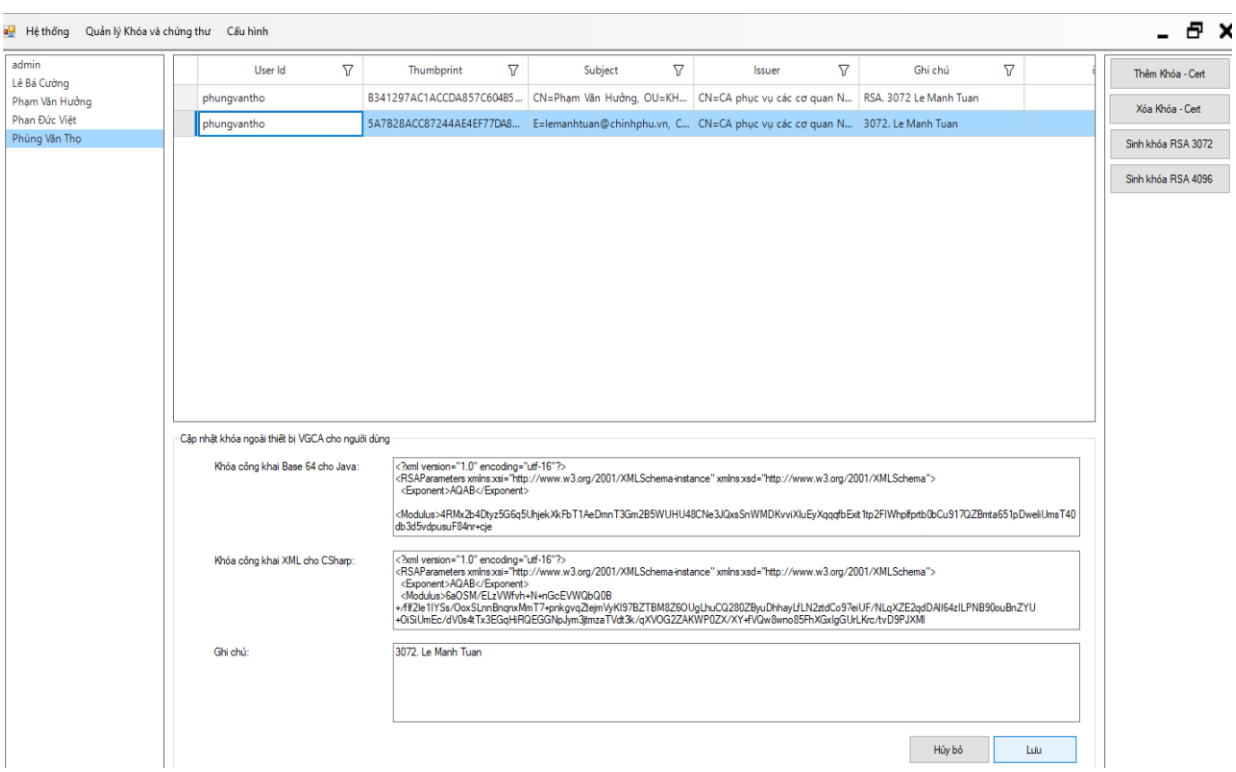

Figure 11. Get public key from VGCA/Bit4ID for users

Our proposed solution uses a combination of both symmetric key cryptography and public key cryptography according to the End-to-End encryption model, so it is highly secure. First, this solution protects against attacks from the server. Intermediate servers cannot decrypt it. Second, the solution prevents man-in-the-middle attacks because encryption and digital signing are done right at the client. Third, the solution has a mechanism to protect private keys and cryptographic parameters. The user's private key is stored in the USB token/PKI SIM device and cannot be read; Using a self-generated key pair, the private key file is encrypted before being stored at the client. Furthermore, the security of the proposed solution combines the security of public key cryptography (RSA) and the security of symmetric key cryptography (AES). According to [9], 2^{93} operations are equivalent to parsing RSA-1939 and 2^{109} operations are equivalent to parsing RSA-2919. Our experiment in the paper used RSA-2048/3072/4096, so it ensures a high level of safety. Also according to [9], to date, the most dangerous attack on AES-256 is biclique which leads to key recovery with a computational complexity of $2^{254.4}$.

V. Conclusion

Theoretically, the paper proposed a solution combining social networks and an end-to-end encryption model for applications in digital transformation. This is a new solution model to promote the strengths of social networks and ensure information safety and security based on the end-to-end encryption model.

Experimentally, we have constructed encryption and decryption software on the client and combined it with Zalo to exchange information and manage work. The system has been tested and operates stably in the IT department - Academy of Cryptography Techniques. Actual test results show that the system operates stably and meets the requirements. On this basis, we will continue to research and integrate digital signatures into the end-to-end encryption system to apply digital transformation more effectively. Moreover, if there is a combination of the Vietnam Government Information Security Commission (which provides several cryptographic libraries) and Zalo, the security features will be "truly" integrated into Zalo.

References

[1] Statista Research. (2023). Leading active social media apps among internet users in Vietnam as of 1st quarter of 2023. https://www.statista.com/statistics/941843/vietnam-leading-social-media-platforms.

[2] Zalo. (2023). Zalo digital transformation. URL: https://oa.zalo.me/chuyendoiso.

[3] Zalo. (2023). Zalo for Developers. https://developers.zalo.me.

[4] Van, T. M. (2005). “Information security”.

[5] Felipe, G. Spring Boot Messaging. Apress.

[6] Peter, L. (2013). Simple Steps to Data Encryption. https://doi.org/10.1016/C2012-0-06008-5.

[7] Khánh, T. V., & Vinh, N. T. (2020). Giải pháp bảo mật đầu cuối cho điện thoại di động. Journal of Science and Technology on Information Security, 9(01), 37-48. https://doi.org/10.54654/isj.v9i01.41.

[8] Phong, T. Q., Chi, D. D., Huy , T. D., & Diep, N. N. (2023). On some issues affecting the security of RSA and ECDSA digital signature schemes. Journal of Science and Technology on Information Security, 1(18), 38-46. https://doi.org/10.54654/isj.v1i18.884.

[9] Thuc, H.V & Tien, D.Q. (2023). A review of security levels for RSA key length. Journal of Science and Technology on Information Security.

[10] Elaine, B. & John, K. (2014). Recommendation for Random Number Generation Using Deterministic Random Bit Generators. NIST Special Publication 800-90A.

ABOUT THE AUTHORS

Cao Thanh Vinh
Workplace: Faculty of Information Technology, Vietnam Academy of Cryptography Techniques.
Email: ctvinh@actvn.edu.vn
Education: Received Master degree in 2016.
Recent research interests: Mobile app security; Mobile software development.

Cơ quan làm việc: Khoa Công nghệ thông tin, Học viện Kỹ thuật mật mã.
Email: ctvinh@actvn.edu.vn
Quá trình đào tạo: Nhận bằng Thạc sĩ vào năm 2016.
Hướng nghiên cứu hiện nay: Bảo mật ứng dụng di động; Phát triển phần mềm di động.

Pham Van Huong
Workplace: Faculty of Information Technology, Vietnam Academy of Cryptography Techniques.
Email: pvhuong@actvn.edu.vn
Education: Received Master degree in 2009 and PhD of Software Engineering in 2015.
Recent research interests: AI and applications; Security software development; IoT security; Cloud security; Software.

Cơ quan làm việc: Khoa Công nghệ thông tin, Học viện Kỹ thuật mật mã.
Email: pvhuong@actvn.edu.vn
Quá trình đào tạo: Nhận bằng Thạc sĩ năm 2019 và Tiến sĩ Kỹ thuật phần mềm năm 2015.
Hướng nghiên cứu hiện nay: AI và ứng dụng; Phát triển phần mềm bảo mật; Bảo mật IoT; Bảo mật đám mây; Phần mềm.