Group Digital Signature on a Mobile Cloud with Signcryption and EdDSA

Siwanon Trairattana, Jirachaya Na Songkhla, Chitipat Yusong, Somchart Fugkeaw

School of ICT, Sirindhorn International Institute of Technology, Thammasat University, Thailand

6522780468@g.siit.tu.ac.th, 6522780418@g.siit.tu.ac.th, 6522781689@g.siit.tu.ac.th, somchart@siit.tu.ac.th

***Abstract*— Mobile device is one of the technologies that has become essential factor of human life. Mobile devices are widely used for various tasks, including communication, computational processing, data storage, and data transmission. However, due to their limited resources and computational power, performing multiple tasks can lead to performance issues. Mobile Cloud Computing (MCC) addresses this problem by offloading certain tasks from the mobile device to the cloud. To implement group digital signatures, we use an encryption scheme called signcryption, which provides confidentiality, integrity, authentication, and non-repudiation. However, managing cryptographic keys becomes a significant challenge due to the involvement of multiple parties. In this paper, we introduce the Edwards-curve Digital Signature Algorithm (EdDSA) as a signing method that effectively addresses the key management problem.**

***Keywords*— Mobile cloud computing, encryption, digital signature, confidentially, integrity, non-repudiation, authentication, EdDSA**

I. INTRODUCTION

Mobile devices have become an essential part of everyday communication, evolving through multiple generations with increasing computing power, processing capabilities, and portability. Despite being fast and convenient for data sharing and communication, they still face limitations such as restricted battery life, processing power, and storage capacity. To overcome these limitations, Mobile Cloud Computing (MCC) has emerged as a solution that allows mobile devices to offload data storage and computation-intensive tasks to the cloud. While this provides significant benefits, it also introduces new challenges particularly in terms of privacy and security.

One major concern is that storing sensitive data in the cloud can expose it to unauthorized access or data breaches if proper protection mechanisms are not in place. Furthermore, mobile devices are vulnerable to loss or theft, making it easier for attackers to steal data or misuse cloud services. Weak authentication and poor access control can further increase these risks. Another issue is the potential visibility of user data to cloud service providers if it is not adequately encrypted. To address this, data should be encrypted before uploading to the cloud. However, because mobile devices have limited resources, cryptographic operations must be lightweight to avoid draining performance, battery, and storage.

In this paper, we use the mobile cloud as the environment of the system model. Mobile Cloud Computing (MCC) [8],[14] refers to an infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile cloud computing works with mobile devices which have resource constraints to support offloading, scalability, and flexibility. The cryptographic scheme that we use with mobile cloud computing is signcryption. Signcryption [6] is one of the cryptographic schemes where both function of signing and encryption is combined in one step. This scheme supports confidentiality, integrity, authentication and non-repudiation. Compared with the encryption scheme, signcryption is suitable for environments where confidentiality and authenticity are required with minimal computational cost. Most systems which use signcryption are mobile and Internet of Things (IoT). In our paper, we separate tasks of signcryption between mobile devices and mobile cloud computing depending on each task. Tasks assigned to the mobile device are lightweight tasks and user information to support battery life, processing power, storage capacity, and network bandwidth of the mobile device. While intensive tasks are offloaded to the cloud, consisting of storing data and computing some function or hashing function. However, key management is one of the concerns of using signcryption with mobile cloud computing. It’s complex and critical.

In our approach, we introduced an Edwards-curve Digital Signature Algorithm of EdDSA to support lightweight and secure against side-channel attack without relying on additional trusted entities between mobile and mobile cloud computing. Additionally, our system proposes Public Key Infrastructure (PKI) to provide lightweight encryption on mobile devices and safely secure the message before uploading it to the cloud. The main contributions of this work are

1. We proposed an outsourced signing and encryption model based on EdDSA to provide resistance against timing attacks and other side-channel attacks.

2. Our proposed model also uses public key infrastructure. Which makes the encryption and decryption process more efficient and lightweight

3. We conducted experiments compare with traditional system to determine the performance of the system proposed in this paper.

II. RELATE WORK

There are many studied introduced and explored Mobile Cloud Computing (MCC). S. Qureshi et al. [8] introduced mobile cloud computing as the future for mobile application. While M. Bahrami [13] applied mobile cloud computing with mobile application to improve emerging mobile cloud application.

J. Mishra et al. [14] proposed a Mobile Cloud Computing Architecture with three tier frameworks of mobile base on cloud computing which integrates mobile capabilities with cloud computing services and separate tasks between mobile and cloud to give more secure and service to users and also provides flexibility, availability and Platform support.

In 1997, Y. Zheng [6] proposed digital signcryption technique which combines the function of digital signature and public key encryption in a single operation and compared with the traditional method called "Signature followed by encryption" by using computation cost and communication cost as the factor. As a result, signcryption has a smaller cost than traditional methods. Many papers using signcryption to improve the security of their application or protocol. In [15], Z. Chuanrong et al. introduced a secure mobile agent protocol by applying signcryption schemes.

Signcryption is the scheme which contain signing and encryption in the one of the steps. The signing algorithm which we want to introduced is Edwards-Curve Digital Signature (EdDSA). Edwards-Curve Digital Signature (EdDSA) is a modern, fast and secure signing algorithm used for signing message. This algorithm is similar to Elliptic-Curve Digital Signature (ECDSA) but its faster and more secure.

Feng et al. [7] introduced two-party EdDSA signature generation with key protection to solve the problem of generating EdDSA signature in a two-party setting by spilt the private key among multiple parties which allow two party to jointly generate a valid EdDSA signature and each party holds a share of the private key instance of traditional EdDSA signature where use a single private key.

Mike et al. [5] introduced multi-signature authentication using an EdDSA-based Algorithm. The scheme constructs a multi signature by combining the sender's private key using AND gate operation. Senders can sign a message using a multi-signature private key, and the receiver can verify the digital signature using a multi-signature public key. The multi-signature can be created however, this method of key generation is not usual, which leads to security issues. And in case the signature is misused. It can't be traced to find whoever is the culprit that misused it.

Großschädl et al. [3] introduced and compared the two implementation strategies for Ed25519 signature verification between speed-optimized variant and memory-optimized variant. The result is memory-optimized variant 24 % slower than speed-optimized variant but this method achieves a 40% reduction in RAM usage which support lightweight.

S. Josefsson et. al [10] introduced high-speed digital signatures suitable for constrained devices. Work such as Mike Yuliana (2024) and Feng et al. (2024) explored multi-party and threshold EdDSA to enhance resilience. These eliminate central authority dependency and improve security under key compromise.

EdDSA is the one of signing algorithm that takes several adventages including fast performance, low cost, strong security and supporting lightweight. [5],[7] explore that EdDSA support strong security by using share signing by allow two party to generate signature together and each party can be holds a private key, meanwhile [3] proposed that EdDSA support the lightweight by tested on 16-bits MSP430 microcontroller, the result is it can reduce the RAM footprint up to 40%.

Chin et al. [4] introduced Group digital signatures are employed to provide confidentiality and authentication. This method entails generating a public key, private key, identity code (ID), and secret key for each member within the group to guarantee user authentication. The application of a challenge-response protocol incorporating overlapping-shifting-EXOR operations is implemented to ensure the confidentiality of individual secret keys.

III. OUR PROPOSED SCHEME

This section is containing the information of each entity and explain how each phase work in the system model

A. *System Model*

1. **Sender** refers to a party that wish to send a message to receiver.
Sender uses their mobile device to sign and encrypt data before the ciphertext is uploaded to the cloud
2. **Receiver** refers to a party that receive the message sent by the sender.
Receiver use their mobile device to decrypt the cipher text and verify the digital signature
3. **Mobile** refers to device that sender and receiver used to send and receive messages.
4. **Android Keystore System** refers to a secure enclave that prevents keys from being extracted. It is used to store the private key and public key of the user. Keystore-backed keys can be accessed using Android's KeyStore API.
5. **Mobile Cloud Computing (MCC)** refers to paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand. It will be used to store cipher text storage and user public key storage. It also computes hash for digital signature process.
6. **Cipher Text Storage** refers to cloud storage that stores cipher text uploaded by sender. The data is encrypted before upload to cloud storage.
7. **User Public Key Storage** refers to cloud storage that stores public key of every user for encryption, decryption and signature verifying process.

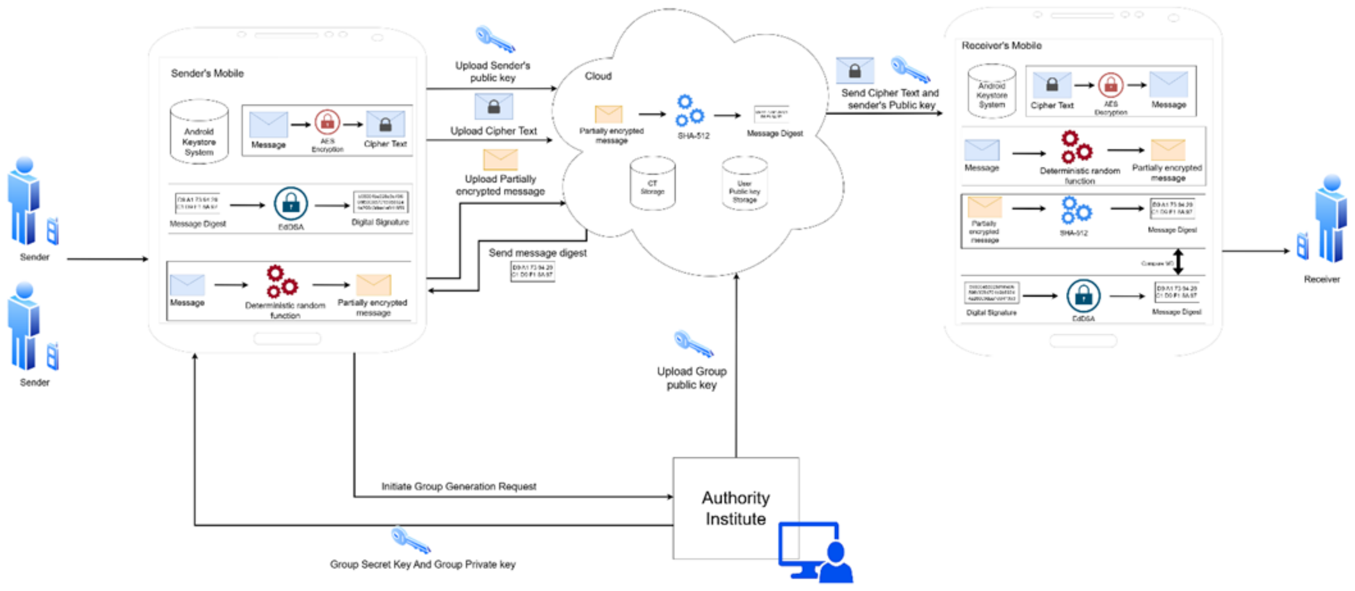

Fig 1: System model

As shown in Fig 1. Sender uploads the message to compute the hash on cloud. The message is randomized using random function to protect the content of the message before uploading to the cloud. The hash will be computed by using the randomized message, then will be sent back to sender. Then sender signs the message digest to obtain a digital signature using Ed25519 sender's private key. Sender combines message and digital signature then encrypt the message to obtain cipher text (CT) by using AES encryption with a key obtained from Diffie-Hellman shared secret between sender and receiver. Finally, sender uploads cipher text to CT storage. For the receiver side, receiver downloads CT from the cloud then decrypt the CT using AES key that obtained from using Diffie-Hellman shared secret between sender and receiver. After that, receiver obtain digital signature and message. Receiver then verifies the signature by using random function with the message and hash it to obtain a message digest, then compare it with another message digest obtained from using sender Ed25519 public key with digital signature. If the message digest matches, it means it was sent by sender, and the content is correct else, it means the message has been tampered with.

B. Process

Phase 1: Key Generation

First phase is key generation which consist of key generation algorithms for generate each type of key for using in system model including

1. Ed25519 Key Pair Generation

Ed25519 Key Pair Generation is key generation algorithm which generate key used for sign and verify digital signature by generate two type of keys consists of Ed25519 public key ($PubK_{Ed}$) used for verifies signature and Ed25519 private key ($PrivK_{Ed}$) used for sign a message.

Sender runs the algorithm by using Ed25519 key generation algorithm to generate a 256-bits private key and public key which will be used for digital signature signing. After key generation, public key is uploaded to store in the cloud storage for digital signature verification. Private key is kept in the mobile device.

$$KeyPairGen_{Ed}(KeyGen_{Ed}, Key_{alisa}) \rightarrow (PubK_{Ed}, PrivK_{Ed})$$

2. X25519 Key Pair Generation

X25519 Key Pair Generation is a key generation algorithm which plays a role in key exchange of Diffie-Hellman by generate two types of keys including X25519 Public Key ($PubK_X$) used for shared key with parties and X25519 Private Key ($PrivK_X$) used for Diffie-Hellman key agreement.

Both sender and receive run this algorithm by using X22519 key algorithm generation algorithm to generate a 256-bits private key and public key. After key generation, the public key of both sender and receiver is uploaded to store in cloud storage for constructing a Diffie-Hellman key exchange, while private key is kept in mobile device.

$$KeyPairGen_X(KeyGen_X, Key_{alisa}) \rightarrow (PubK_X, PrivK_X)$$

3. Diffie-Hellman Shared Key Generation

Diffie-Hellman Shared Key Generation is a shared key generation algorithm which generate a derived symmetric key from Diffie-Hellman (*AES_Key*) used to encrypt symmetric key that generate by symmetric key generation (*AES_Key*) as a shared secret key that shared between each party by using X25519 public and private key as an input.

Sender runs this algorithm by using the sender's private key and receiver's public key to derive an *AES_key* from the Diffie-Hellman key exchange to encrypt M. Receiver runs this algorithm by using receiver private key and sender's public key to derive an *AES_key to decrypt* CT

$$Diffie\text{-}Hellman(PrivK_X, PubK_X) \rightarrow AES\_key$$

4. Group Digital Signature Key Generation

Group Digital Signature Key Generation is a key generation algorithm for generates key used for sign and verify digital signature. For group digital signature, we generate two types of keys consist of Group Secret Key (*GSK*) and Group Public Key (*GSK*).

Group Secret Key (GSK)

Group secret key is used for signing

$$SecretKeyGen_{Group}(SecretKey_{Member}, K_a) \rightarrow GSK$$

Group Public Key (GPK)

Group public key used for verifying the digital signature which uses group secret key to sign generate by using the secret key.

$$PublicKeyGen_{Group}(SecretKey_{Group}, K_a) \rightarrow GPK$$

Phase 2: Signing

This phase consists of two steps: Compute hash on cloud and sign the message in sender's mobile device.

1. Compute hash on cloud

In order to compute hash on cloud securely, sender compute intermediate cipher text (*Int_CT*) by using deterministic random generator (*F*) with a seed that will be the same for both sender and receiver producing *Int_CT* to protected the confidentiality of the plaintext (*PT*).

$$F(PT, seed) \rightarrow Int\_CT$$

Then, the *Int_CT* will be hashed with SHA512 to produce a message digest (*MD*).

$$HASH(Int\ \_\ CT) \rightarrow MD$$

2. Sign the message

The message digest (*MD*) is sent back to the sender to sign the message but in this case using Group Secret Key (*GSK*) to the message in case of group digital signature (*DS*).

$$ENC_{Ed}(GSK, MD) \equiv DS$$

Phase 3: Encryption

Before the message sends to receiver, the message needs to encrypt to ensure the security.

1. Encrypt message M

In order to make the message much more secure, The sender constructs a shared secret key with Diffie-Hellman by using the sender's private key (*sPrivK*$_X$) and receiver's public key (*rPubK*$_X$) that generate by using X25519 key generation to derive an *AES_key*. Then, the sender used *AES_Key* to encrypt the message (*M*), which is digital signature (*DS*) combined with plaintext (*PT*), into cipher text *CT* by using Advanced Encryption Symmetric (ENC_{AES})

$$M = DS + PT$$
$$Diffie\text{-}Hellman(sPrivK_X, rPubK_X) \rightarrow AES\_key$$
$$ENC_{AES}(AES\_Key, M) \equiv CT$$

The *CT* is uploaded to the cloud after encryption.

Phase 4: Decryption

In order to receive the message, receiver need to decrypt the ciphertext and verify digital signature.

1. Decrypt CT

The receiver constructed a shared secret with Diffie-Hellman by using the receiver's private key (*rPrivK*$_X$) and the sender's public key (*sPubK*$_X$) that generate by using X25519 key generation to derive an *AES_key* then use to decrypt the *CT* based on the AES decryption algorithm.

$$Diffie\text{-}Hellman(rPrivK_X, sPubK_X) \rightarrow AES\_key$$
$$DEC_{AES}(AES\_Key, CT) \equiv M$$

2. Verify digital signature

The receiver split the message (*M*) to obtain digital signature (*DS*) then verify the signature by using group public key (*GPK*).

$$verify(DS,GPK) \equiv VerifyGroupDS$$

IV. IV. SECURITY ANALYSIS

In this section, we analyze the security aspects of our proposed group digital signature system, which combines EdDSA and signcryption within a mobile cloud environment. Our analysis focuses on how the design ensures confidentiality, integrity, authenticity, and resilience against common attacks, all while considering the limitations of mobile devices.

A. Message Confidentiality

To protect message content, the plaintext is combined with its digital signature and encrypted using a symmetric AES key (referred to as AES_Key). This key is securely derived from a Diffie-Hellman key exchange using the X25519 algorithm. This ensures that only the intended receiver, who shares the same Diffie-Hellman secret, can decrypt the message. Even if intercepted, the ciphertext remains secure without access to the shared AES key.

B. Integrity and Authentication

Our scheme uses Ed25519, a variant of EdDSA, to sign the hash of the message before encryption. This allows the receiver to verify whether the message before encryption. This allows the receiver to verify whether the message has been altered during transmission. If even a single bit of the message changes, the hash will no longer match and the signature verification will fail. Also, since the signature was created with the sender's private key, which is securely stored in the mobile device using the Android Keystore system, it proves that the message came from the expected sender.

C. Non-repudiation

Because each sender signs messages with their private key, and that key is never shared or uploaded, the sender cannot deny sending the message later. This property is crucial in group scenarios where tracking the message origin is important. While our current model allows group members to sign collectively, each signature is still verifiable using the corresponding group public key, supporting accountability.

D. Protection Against Side-Channel Attacks

One of the reasons we chose EdDSA is because of its built-in resistance to side-channel attacks. Ed25519 does not depend on random number generation during the signing process, which helps avoid timing-based attacks. Moreover, private keys are kept in the Android Keystore's secure hardware zone, making it very difficult for attackers to extract them using memory or power analysis techniques.

F. Group Signature Privacy and Accountability

When group signatures are used, the system create a group private key for each group member for signing. The group public key allows receivers to verify the signature as belonging to the group. While this offers sender anonymity within the group, our model assumes that all group members are trusted. Future improvements could include features like traceable signatures to identify misbehaving group members when needed.

G. Protection Against Replay and Forgery Attacks

Before signing, the message is processed using a deterministic random function to create an intermediate ciphertext. This means each message is hashed in a unique way before being signed, even if the plaintext is the same. This prevents attackers from reusing old messages (Replay attacks). Also, Ed25519 signatures are cryptographically secure and extremely difficult to forge without the sender's private key.

V. EVALUATION

In this section, we evaluate the performance of our proposed system by conducting a comparative analysis in terms of functionality of our scheme and the related schemes

A. Functional Analysis

Table I shows the functions of our scheme that use cloud off-loading, our model that doesn't use cloud off-loading and related works.

	Crypto Operation on mobile			Outsource Cloud off-loading	
	Diffie Hellman key exchange	AES	CP-ABE	pre-hashing	Signature verification
Our model without cloud off-loading	✓	✓	✗	✗	✗
Ours	✓	✓	✗	✓	✓
[2]	✗	✓	✓	✗	✗
[17]	✓	✓	✗	✗	✗

Table I: Functional Analysis Table

Our scheme uses Diffie Hellman to generate an AES key then use the generated AES key to encrypt the message. The difference between Our model that utilize cloud and our model that doesn't utilize cloud is operation that outsources to cloud. Pre-hashing is an operation in digital signature signing. Our model without outsourcing to cloud do hashing on device and have less communication cost than Model that outsource to cloud. Another operation that our model outsource to cloud is digital signature verifying operation. Verifying group digital signature consume more resource than single digital signature so outsourcing it can help reduce the workload on mobile device. [2] model doesn't have digital signature algorithm used and it use CP-ABE to encrypt AES key. It off-loads decryption operation to cloud which left only AES encryption on user side. [17] use Diffie Hellman key exchange to create an AES key with ECC and use ECDSA to sign the message.

B. Computational Analysis

Table II shows the computation comparison of our scheme and related schemes. The following notations are used.

Dif: Diffie-Hellman Shared Key Generation
AES: AES encryption/decryption operation.
Ha: Hashing operation
Ed: Sign/Verify a message with EdDSA
Par: Partial encrypt/decrypt message with random function
XOR: XOR operation in 256bits data.
CP-ABE; CP-ABE encryption/decryption
ECD: ECDSA Digital signature signing algorithm

	Encryption cost		Decryption Cost	
	Sender	Cloud	Receiver	Cloud
Ours	Dif + AES + Ed+ Par	Ha	Dif+ AES+Par	Ed
Ours without cloud	Dif + AES + Ha+ Ed+ Par	-	Dif+ AES+ Ed+Par	-
[2]	CP-ABE + AES + XOR	-	AES + XOR	CP-ABE
[17]	AES + Dif + ECD	-	AES+ Dif+ ECD	-

Table II: Computational Analysis Table

In our model, we use partial encryption to encrypt the message before off-loading to cloud to get message digest after that, we sign message digest to obtain digital signature. For encryption, we encrypt partial encrypted message with AES with a key from Diffie-Hellman Shared Key Generation. Since our model off-load hashing operation to the cloud, our model has less computation cost than not using cloud-offloading. [2] use AES encryption, CP-ABE, and XOR operation. [17] use AES encryption, Diffie Hellman key exchange and ECDSA to sign a message.

As for Decryption cost in our model, receiver can verify the signature on their device and decrypt the message with AES decryption and partial decryption. As for group signature verification, receiver send a partial encryption and group signature to verify cloud. The decryption and signature verification cost between using cloud off-load and not using cloud off-load is different at group signature verification operation. In [2], there is CP-ABE off-load to cloud to reduce load on receiver's device and in [17], it is similar to the encryption. That is, it uses ECDSA to verify a signature and Diffie Hellman key exchange to create AES key then decrypt a ciphertext. Our model use X25519 which specialize in Diffie Hellman key exchange can do key exchange operation 10 times faster than ECDSA according to our experiment.

C. Experimental Analysis

For performance analysis, we measure the time spent on encryption, signing the signature, decryption, and verifying the signature, then compare it with other schemes. We conducted a test with a Xiaomi Redmi Note 13 5G equipped with MediaTek Dimensity 6080 and 8GB of RAM running the Android 15 OS as a sender and receiver mobile phone. A Google Cloud is used to deploy application with instance class F2 that has 512 MiB of memory and 1.2 GHz vCPU to off-load hashing operation on the cloud. A Google Cloud SQL with 4 vCPUs and 16 GB of memory is used as a cloud storage to store the public key and ciphertext of sender and receiver.

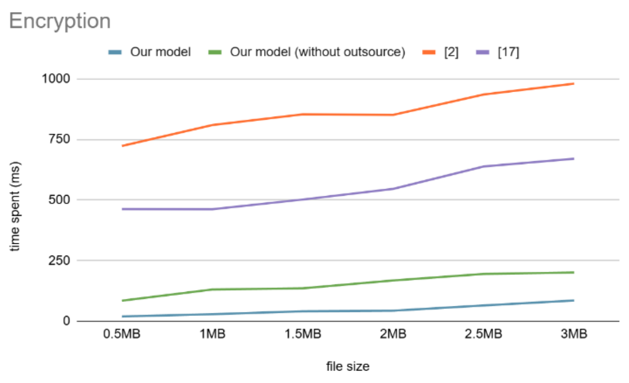

Fig 2: Encryption

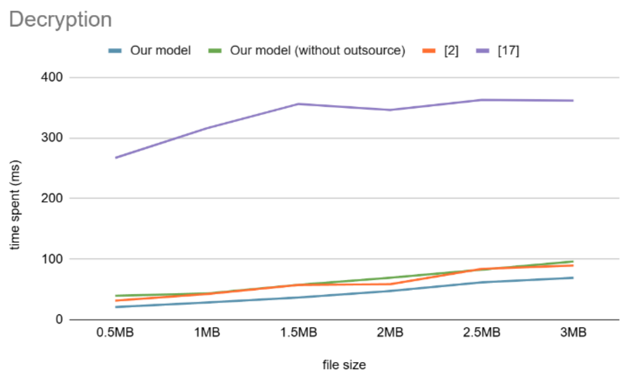

Fig 3: Decryption

The graph shows the result of experiment. Our model spent least time on encryption more than traditional model that doesn't use cloud and other model.[2] model uses AES encryption and RSA encryption and doesn't off-load to cloud. [17] model use ECC to do a Diffie Hellman and use AES encryption to encrypt the message which is similar to our model but our model use X25519 which specialize in Diffie Hellman key exchange. As for decryption, our model, [17], and [2] model uses AES Decryption to do decrypt the message. Our model also verifies the signature of the mesage using EdDSA. [2] model do some XOR operation in addition to its decryption time is a bit more than ours. And [17] has more time spent in Diffie Hellman key exchange than us so our model has least time spent in decryption.

VI. CONCLUSION AND FUTURE WORK

In this paper, we purposed a public key encryption model that use X25519 and EdDSA as encryption algorithm and signing algorithm. In our scheme, the hashing operation of encryption operation is off-loaded to the cloud to decrease the load on sender device. we conducted the comparison analysis to measure performance of our scheme and our scheme without cloud off-load. It can help reduce workload of mobile device but there is a communication cost as its downside. For future work, we will extend our model to off-load more encryption operation to the cloud and implement a way to allow multiple receivers to decrypt the file from the sender whose encrypt only single time using public key encryption.

REFERENCES

[1] S. Fugkeaw, "A Secure and Efficient Data Sharing Scheme with Outsourced Signcryption and Decryption in Mobile Cloud Computing," 2021 IEEE International Conference on Joint Cloud Computing (JCC), Oxford, United Kingdom, 2021, pp. 72-79, doi: 10.1109/JCC53141.2021.00024.

[2] S. Fugkeaw, "Implementing An Outsourced Dual-Proxy Signing and Decryption Scheme in Mobile Cloud Computing," 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), St.Petersburg, FL, USA, 2023, pp. 299-307, doi: 10.1109/IPDPSW59300.2023.00058.

[3] Großschädl, J., Franck, C., Liu, Z. (2021). Lightweight EdDSA Signature Verification for the Ultra-Low-Power Internet of Things. In: Deng, R., et al. Information Security Practice and Experience. ISPEC 2021. Lecture Notes in Computer Science(), vol 13107. Springer, Cham. https://doi.org/10.1007/978-3-030-93206-0_16

[4] Chin-Ming Hsu, "A group digital signature technique for authentication," IEEE 37th Annual 2003 International Carnahan Conference onSecurity Technology, 2003. Proceedings., Taipei, Taiwan, 2003, pp. 253-256, doi: 10.1109/CCST.2003.1297568

[5] Mike Yuliana "Efficient Multi-signature and QR Code Integration for Document Authentication Using EdDSA-based Algorithm," International Journal of Intelligent Engineering and Systems, vol. 17, no. 2, pp. 390–401, Apr. 2024, doi: 10.22266/ijies2024.0430.32.

[6] Y. Zheng, "Digital Signcryption or How to Achieve Cost (Signature Encryption)? Cost (Signature)+ Cost (Encryption), " Proc. CRYPTO97, Springer Press, Aug. 1997, pp.165-179, doi: 10.1007/BF60052234.

[7] Q. Feng, D. He, M. Luo, Z. Li and K. -K. R. Choo, "Practical Secure Two-Party EdDSA Signature Generation with Key Protection and Applications in Cryptocurrency," *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Guangzhou, China, 2020, pp. 137-147, doi: 10.1109/TrustCom50675.2020.00031.

[8] S. Qureshi, T. Ahmad, K. Rafique and Shuja-ul-islam, "Mobile cloud computing as future for mobile applications - Implementation methods and challenging issues," *2011 IEEE International Conference on Cloud*

Computing and Intelligence Systems, Beijing, China, 2011, pp. 467-471, doi: 10.1109/CCIS.2011.6045111.

[9] G. Shankar, et al. “Improved Multisignature Scheme for Authenticity of Digital Document in Digital Forensics Using Ed-ward-Curve Digital Signature Algorithm,” Security and Communication Networks, vol.2023, Article ID 2093407, 18 pages,2023. Doi:10.1155/2023/2093407

[10] RFC 8032: J.L. Josefsson, S. F. Josefsson, “EdDSA and Ed25519/Ed448 for Signing,” IETF RFC 8032, Jan. 2017.

[11] Maxwell, G., Poelstra, A., Seurin, Y., & Wuille, P. (2019). Simple Schnorr multi-signatures with applications to Bitcoin. Designs, Codes and Cryptography, 87(9), 2139–2164. https://doi.org/10.1007/s10623-019-00608-x

[12] S. S. Qureshi, T. Ahmad, K. Rafique and Shuja-ul-islam, "Mobile cloud computing as future for mobile applications - Implementation methods and challenging issues," *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, Beijing, China, 2011, pp. 467-471, doi: 10.1109/CCIS.2011.6045111.

[13] M. Bahrami, "Cloud Computing for Emerging Mobile Cloud Apps," *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, San Francisco, CA, USA, 2015, pp. 4-5, doi: 10.1109/MobileCloud.2015.40.

[14] J. Mishra, S.K. Dash, S. Dash(2012). Mobile-Cloud: A Framework of Cloud Computing for Mobile Application. In: Meghanathan, N., Chaki, N., Nagamalai, D. (eds) Advances in Computer Science and Information Technology. Computer Science and Information Technology. CCSIT 2012. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 86. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27317-9_36

[15] Z. Chuanrong and Z. Yuqing, "Secure mobile agent protocol by using signcryption schemes," *2009 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Zhangjiajie, China, 2009, pp. 108-112, doi: 10.1109/CYBERC.2009.5342190.

[16] Przydatek, B., Wikström, D. (2010). Group Message Authentication. In: Garay, J.A., De Prisco, R. (eds) Security and Cryptography for Networks. SCN 2010. Lecture Notes in Computer Science, vol 6280. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15317-4_25

[17] Gayoso Martínez, V., et al. ELLIPTIC CURVE CRYPTOGRAPHY. JAVA PLATFORM IMPLEMENTATIONS. digital.csic.es/bitstream/10261/21232/3/Int%20J%20Inf%20Tech%20Sec-4%20-%20copia.pd