Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
2326-9865

Pay as You Decrypt Using FEPOD Scheme and Blockchain

Fatimunisa 1, Qudsiya Fatima 2, Rubeena 3,

Mrs Imreena Ali 4

1, BE Student – Dept. Of CSE, ISL Engineering College, TS, India

2, BE Student – Dept. Of CSE, ISL Engineering College, TS, India

3, BE Student – Dept. Of CSE, ISL Engineering College, TS, India

4, Assistant Professor – Dept. Of CSE, ISL Engineering College, TS, India

Article Info
Page Number: 1462-1468
Publication Issue:
Vol. 72 No. 1 (2023)

***ABSTRACT* –** The concept of functional encryption (FE) has been introduced to address the shortcomings of public-key encryption (PKE) in many emerging applications which require both data storage and data sharing (e.g., cloud storage service). One of the major issues existing in most FE schemes is the efficiency, as they are built from bilinear pairings of which the computation is very expensive. A widely accepted solution to this problem is outsourcing the heavy workloads to a powerful third party and leaving the user with the light computation. Nevertheless, it is impractical to assume that the third party (e.g., the cloud) will provide free services. To our knowledge, no attention has been paid to the payment procedure between the user and the third party in an FE with outsourced decryption (FEOD) scheme under the assumption that neither of them should be trusted. Leveraging the transactions on cryptocurrencies supported by the blockchain technology, in this paper, we aim to design FE with payable outsourced decryption (FEPOD) schemes. The payment in an FEPOD scheme is achieved through a blockchain-based cryptocurrency, which enables the user to pay a third party when it correctly completes the outsourced decryption. We define the adversarial model for FEPOD schemes, and then present a generic construction of FEPOD schemes. Also, we evaluate the performance of the proposed generic construction by implementing a concrete FEPOD scheme over a blockchain platform.

***Keywords*:** FEPOD, Public-key encryption, Blockchain, Cryptocurrencies.

Article History
Article Received: 15 October 2022
Revised: 24 November 2022
Accepted: 18 December 2022

1. INTRODUCTION

Take the cloud storage scenario as an example, where all data items are encrypted via an encryption mechanism such as functional encryption1 (FE) [7], [23] and stored in encrypted forms to protect the data security and privacy. Suppose that Alice, a privilege user of a cloud storage application, is using a device with the constrained resource. Alice intends to access the encrypted data stored on the cloud, but she is unable to perform the heavy computation (e.g., the pairing operation) of decryption. A straightforward solution to this problem is an FE with outsourced decryption (FEOD) scheme such as an identity-based encryption (IBE) with outsourced decryption or an attribute-based encryption (ABE) with outsourced decryption scheme, which supports the user (i.e., Alice) to outsource the majority of the computation workloads in decryption to a powerful third party without leaking any sensitive information about the original data. Considering that in practice, the third parties would not like to provide a free service, and they expect to be paid for what they have done for others. Alice may send

the computation task to a nearby device possessed by Bob, which is capable of conducting the computation in the network, and promise that “I will pay $1 to Bob once he provides me the correct result to this computation task”. Bob receives the message, executes the computation, and sends Alice the result. To this end, two issues remain to be addressed: firstly, a mechanism that enables Alice to verify the correctness of Bob’s answer before she pays Bob; secondly, a mechanism to address the worry of Bob that Alice may deny the correctness of his answer to escape the payment.

2. LITERATURE REVIEW

We describe Charm, an extensible framework for rapidly prototyping cryptographic systems[2]. Charm provides a number of features that explicitly support the development of new protocols, including: support for modular composition of cryptographic building blocks, infrastructure for developing interactive protocols, and an extensive library of reusable code. Our framework also provides a series of specialized tools that enable different cryptosystems to interoperate. We implemented over forty cryptographic schemes using Charm [10], including some new ones that to our knowledge have never been built in practice. This paper describes our modular architecture, which includes a built-in benchmarking module to compare the performance of Charm primitives to existing C implementations. We show that in many cases our techniques result in an order of magnitude decrease in code size, while inducing an acceptable performance impact. Lastly, the Charm framework is freely available to the research community and to date, we have developed a large, active user base.

We study the problem of verifiable computation (VC) in which a computationally weak client wishes to delegate the computation of a function f on an input x to a computationally strong but untrusted server. We present new general approaches for constructing VC protocols, as well as solving the related problems of program checking and self-correcting. The new approaches reduce the task of verifiable computation to suitable variants of secure multiparty computation (MPC) protocols. In particular, we show how to efficiently convert the secrecy property of MPC protocols into soundness of a VC protocol via the use of a message authentication code (MAC). The new connections allow us to apply results from the area of MPC towards simplifying, unifying, and improving over previous results on VC and related problems. In particular, we obtain the following concrete applications: (1) The first VC protocols for arithmetic computations which only make a blackbox use of the underlying field or ring; (2) a noninteractive VC protocol for boolean circuits in the preprocessing model, conceptually simplifying and improving the online complexity of a recent protocol of Gennaro et al. (Cryptology ePrint Archive: Report 2009/547); (3) NC0 self-correctors for complete languages in the complexity class NC1 and various log-space classes, strengthening previous AC0 correctors of Goldwasser et al. (STOC 2008).

One of the most *promising* innovations offered by the cryptographic currencies [4] (like Bitcoin) are the so-called \emph{smart contracts}, which can be viewed as financial agreements between mutually distrusting participants. Their execution is enforced by the mechanics of the currency, and typically has monetary consequences for the parties. The rules of these contracts are written in the form of so-called ``scripts", which are pieces of code in some ``scripting

language". Although smart contracts are believed to have a huge potential, for the moment they are not widely used in practice. In particular, most of Bitcoin miners allow only to post standard transactions (i.e.: those without the non-trivial scripts) on the blockchain. As a result, it is currently very hard to create nontrivial smart contracts in Bitcoin.

We propose a fully functional identity-based encryption scheme (IBE). The scheme has chosen ciphertext security in the random oracle model assuming a variant of the computational Diffie Hellman problem. Our system is based on bilinear maps between groups. The Weil pairing on elliptic curves is an example of such a map. We give precise definitions for secure identity-based encryption schemes and give several applications for such systems.

This paper addresses the problem of designing practical protocols for proving properties about encrypted data. To this end, it presents a variant of the new public key encryption of Cramer and Shoup based on Paillier's decision composite residuosity assumption, along with efficient protocols for verifiable encryption and decryption of discrete logarithms (and more generally, of representations with respect to multiple bases). This is the first verifiable encryption system that provides chosen ciphertext security and avoids inefficient cut-and choose proofs. The presented protocols have numerous applications, including key escrow, optimistic fair exchange, publicly verifiable secret and signature sharing, universally composable commitments, group signatures, and confirmer signatures.

3. SYSTEM ARCHITECTURE

This project will be built in Python. We are working on creating an encryption and decryption system using block chain and FEPOD scheme. If you want to encrypt a document or a file you can do it through this model, When anyone wants to decrypt it they need to pay the required charges to decrypt the file in order to access the data in it.

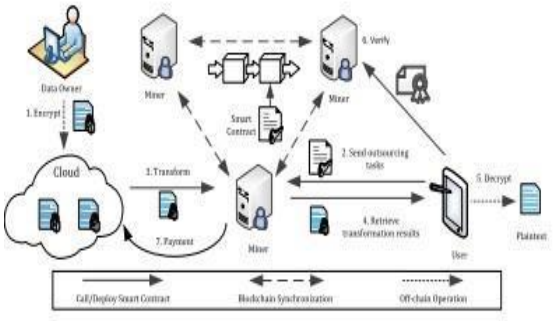


Fig 1: SYSTEM ARCHITECTURE

4. IMPLEMENTATION

Now-a-days all users data are managing from cloud and this cloud cannot be trusted as they may misuse data and to provide security to data Encryption techniques was introduced. This encryption techniques may secure data but require heavy computation for both encryption and decryption task. Users' tiny devices may not perform such heavy computation so third party services was introduced which decrypt outsource data on behalf of users. To provide these services third party may misuse decryption data and may charge more. So, in this paper we are

employing normal nearby or random users to participate in decryption process and they will get paid for decryption. So, all interested users may lend their resources for decryption process. All payments will be managed by Blockchain as crypto currency. As Blockchain has inbuilt support for data integrity and funds can be managed securely without any alteration or tamper.

To manage outsource payment we have used

Blockchain Ethereum with Truffle environment and to store or retrieve data in Blockchain we need to design SOLIDITY (Smart Contract) code which will contains function to store and retrieve data. To implement this project, we have designed below Solidity code.

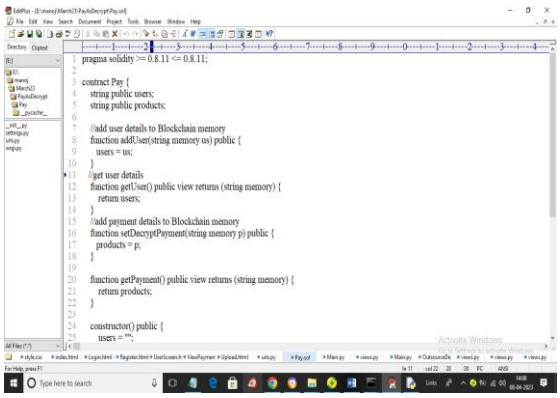


Fig 2: SOLIDITY CODE

In above Solidity code also known as smart contract we have designed functions to store and retrieve USERS and outsource payment details. Now we need to deploy above contract in Ethereum by using below instructions:

1) Go inside 'hello-eth/node modules/.bin' folder and then double click on 'runBlockchain.bat' file to get below screen.

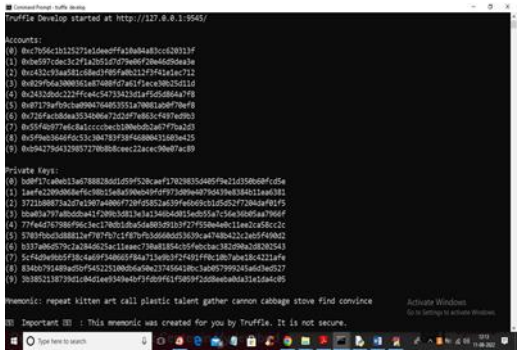


Fig 3

3) In above screen we can see Blockchain created some default accounts and private keys and in above screen just type command as 'truffle migrate' and press enter key to deploy contract and get below output.

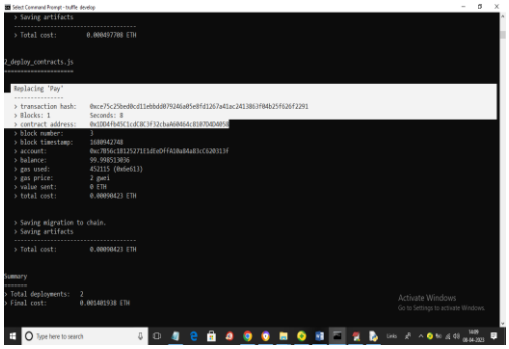
4) 

Fig 4

5) In above screen in white colour text, we can see 'Pay' contract deployed and we got contract address also and this address we need to specify in python program to store and retrieve Payment details. In below screen I am showing python code to access Blockchain contract by using above contract address.

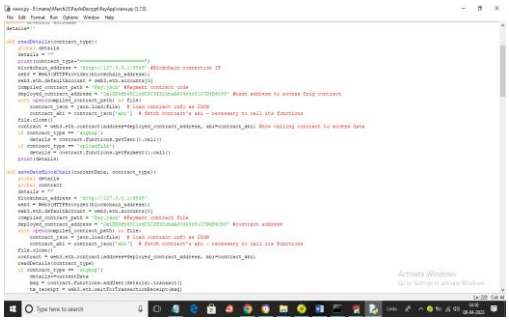
6) 

Fig 5

7) In above screen read red colour comments to know about calling Blockchain contract to store and retrieve data using python program.

Modules Information

To implement this project, we have designed 2 modules called User Login and User Signup.

User can sign up and login and then can upload encrypted files to cloud. Any user can view list of uploaded files with description and can request to download. While downloading file will be outsource to random user for decryption and after decryption file start download and payment will be credited to decryption user Blockchain account.

5. CONCLUSION

Due to the advantage over public-key encryption (PKE), functional encryption (FE) has been considered as an encryption mechanism to protect data security and privacy in many emerging applications such as the cloud computing services. Unfortunately, most of the existing FE schemes are not sufficiently efficient to be applied in the real world. The notion of FE with outsourced decryption (FEOD) has been suggested to reduce the heavy workloads of users by delegating the *majority* of the computation to a third party. However, one issue that has been missed attention in previous FEOD schemes is how to conduct the payment between the user

who sends out the outsourcing computation task and the third party who finishes the outsourcing computation task. With this observation in mind, in this paper, we presented a generic FE with payable outsourced decryption (FEPOD) scheme which is publicly verifiable to enable the third party to be paid for the service it provides via a blockchain-based cryptocurrency. After proving the security of the given generic FEPOD construction, we described the process of integrating FEPOD into a blockchain, and implemented an instantiation of FEPOD over a blockchain to evaluate its efficiency in practice.

6.0 REFERENCES

1. A Python Interface for Interacting With the Ethereum Blockchain and Ecosystem. Accessed: Jul. 10, 2019. [Online]. Available: https://github.com/ethereum/web3.py
2. J. A. Akinyele et al., "Charm: A framework for rapidly prototyping cryptosystems," J. Cryptograph. Eng., vol. 3, no. 2, pp. 111–128, Jun. 2013.
3. B. Applebaum, Y. Ishai, and E. Kushilevitz, "From secrecy to soundness: Efficient verification via secure computation," in Proc. 37th Int. Colloq. Automat., Lang., Program., in Lecture Notes in Computer Science, vol. 6198. Bordeaux, France: Springer, Jul. 2010, pp. 152–163.
4. W. Banasik, S. Dziembowski, and D. Malinowski, "Efficient zeroknowledge contingent payments in cryptocurrencies without scripts," in Proc. 21st Eur. Symp. Res. Comput. Secur., in Lecture Notes in Computer Science, vol. 9879. Heraklion, Greece: Springer, Sep. 2016, pp. 261–280, doi: 10.1007/978-3-319-45741-3_14.
5. I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in Proc. 34th Annu. Cryptol. Conf., in Lecture Notes in Computer Science, vol. 8617. Santa Barbara, CA, USA: Springer, Aug. 2014, pp. 421–439.
6. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," SIAM J. Comput., vol. 32, no. 3, pp. 586–615, Jan. 2003.
7. D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in Proc. 8th Theory Cryptogr. Conf. (TCC), in Lecture Notes in Computer Science, vol. 6597. Providence, RI, USA: Springer, Mar. 2011, pp. 253–273.
8. V. Buterin. Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform. Accessed: Jul. 10, 2019. https://github.com/ethereum/wiki/wiki/White-Paper
9. J. Camenisch and V. Shoup, "Practical verifiable encryption and decryption of discrete logarithms," in Proc. 23rd Annu. Int. Cryptol. Conf., in Lecture Notes in Computer Science, vol. 2729. Santa Barbara, CA, USA: Springer, Aug. 2003, pp. 126–144.
10. Shukla, A., Juneja, V., Singh, S., Prajapati, U., Gupta, A., & Dhabliya, D. (2022). Role of hybrid optimization in improving performance of sentiment classification system. Paper presented at the PDGC 2022 - 2022 7th International Conference on Parallel, Distributed and Grid Computing, 541-546. doi:10.1109/PDGC56933.2022.10053333 Retrieved from www.scopus.com
11. Sindhwani, N., Anand, R., Vashisth, R., Chauhan, S., Talukdar, V., & Dhabliya, D. (2022). Thingspeak-based environmental monitoring system using IoT. Paper presented at the PDGC 2022 - 2022 7th International Conference on Parallel, Distributed and Grid

Computing, 675-680. doi:10.1109/PDGC56933.2022.10053167 Retrieved from www.scopus.com

12. Singh, H., Ahamad, S., Naidu, G. T., Arangi, V., Koujalagi, A., & Dhabliya, D. (2022). Application of machine learning in the classification of data over social media platform. Paper presented at the PDGC 2022 - 2022 7th International Conference on Parallel, Distributed and Grid Computing, 669-674. doi:10.1109/PDGC56933.2022.10053121 Retrieved from www.scopus.com
13. Talukdar, V., Dhabliya, D., Kumar, B., Talukdar, S. B., Ahamad, S., & Gupta, A. (2022). Suspicious activity detection and classification in IoT environment using machine learning approach. Paper presented at the PDGC 2022 - 2022 7th International Conference on Parallel, Distributed and Grid Computing, 531-535. doi:10.1109/PDGC56933.2022.10053312 Retrieved from www.scopus.com
14. M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, “Zeroknowledge contingent payments revisited: Attacks and payments for services,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS), Dallas, TX, USA, 2017, pp. 229–243, doi: 10.1145/3133956.3134060.
15. Baig, M. S., Bari, D. R. M. A., & Khan, P. A. (n.d.). Weapon detection using artificial intelligence and deep learning for security applications. Ijarst.In. Retrieved May 10, 2023.
16. Khan, 1. Pathan Ahmed, & Waheed Farooqi, 2. M. R. M. (n.d.). Functional outsourcing of linear programming in secured cloud computing. Pen2print.org. Retrieved May 10, 2023.