Bitcoin-Based Anti-collusion Fair Payments for Outsourcing Computations in Cloud Computing

Duo Zhang[1,2], Xiaodong Zhang[3], Sheng Gao[4], and Youliang Tian[1(✉)]

[1] State Key Laboratory of Public Big Data,
College of Computer Science and Technology, Guizhou University, GuiYang, China
youliangtian@163.com
[2] School of Mathematics and Statistics, Guizhou University, GuiYang, China
[3] College of Mathematics and Computer Science, Liupanshui Normal College, Liupanshui, China
[4] School of Information, Central University of Finance and Economics, Beijing, China

Abstract. As an attractive cloud computing model, outsourcing computation often has a fair payment issues. In cloud computing, resource-constrained users outsource tasks and pay for them. However, the traditional outsourcing computation model can hardly resist the threat of collusion between calculators. In this paper, we propose a new scheme of Bitcoin-based anti-collusion fair payments for outsourcing computations (BAPay), give the architecture of BAPay, security requirements, and describe the design details. Besides, the scheme guarantees that an honest calculator will be paid for doing the calculations, regardless of the outsourcer's behaviour. In addition, the security analysis shows that BAPay achieves completeness and fairness.

Keywords: Cloud computing · Outsourcing computation · Bitcoin · Anti-collusion · Fair payments

1 Introduction

As a promising computing paradigm, cloud computing has many attractive advantages. With the rapid development of cloud computing technology, more and more individuals and businesses to upload all kinds of data to a third party cloud platforms, to facilitate sharing or cost savings [1]. In cloud computing, as

This work is supported by the Key Projects of the Joint Fund of the National Natural Science Foundation of China (No. U1836205); the National Natural Science Foundation of China (No. 61662009, No. 61772008); the Guizhou Province Science and Technology Major Special Plan (No. 20183001); the Research on Key Technologies of Blockchain for Big Data Applications (No. [2019]1098); the Foundation of Postgraduate of Guizhou Province (No. YJSCXJH2019015).

S. Yu et al. (Eds.): SPDE 2020, CCIS 1268, pp. 309–321, 2020.
https://doi.org/10.1007/978-981-15-9129-7_22

end users are usually resource-constrained, they often need to outsource computing services, which brings great impetus to the development of cloud computing [2]. Although cloud computing provides users with flexible and efficient outsourcing services, the security, completeness and reliability of data remain the main concerns of users [3–5]. For example, in outsourcing computation, users are not willing to pay for services if the results returned by the outsourcing service provider are incorrect.

Recently, great efforts have been made to verifiable outsourcing computation [6]. Although these technologies can guarantee the security of data storage and the correctness of computing, they cannot solve all security threats in cloud computing. Because in an outsourcing computation, the calculators want to get the service reward before returning the calculation result to the outsourcer. However, the outsourcer expects to get and check the results, and then pays for the service if the calculations are correct. Therefore, if the outsourcer and the calculators do not trust each other, the payment issue can be extremely challenging for an outsourcing computation that considers fairness.

In fact, to address the issues of payment, the payment mechanism of most existing scheme still adopts the traditional, relies on the trusted third party [7]. However, in the cloud, the traditional payment need a bank generates payment token, which has some disadvantages. Recently, technologies such as Bitcoin and Blockchain have received a lot of attention, because they can operate without either party's control. Therefore, we divide the calculation task into smaller tasks and assign them to different calculators. After each calculator completes the corresponding calculation task, the calculation results are sent to the outsourcer. Establish a fair payment protocol based on Bitcoin between the outsourcer and the calculators so that cryptocurrency can be transferred between them without the need for a third party. To the best of our knowledge, Bitcoin technologys are rarely widely and fairly used in payments for the outsourcing computation.

1.1 Our Contribution

In this paper, we propose a Bitcoin-based anti-collusion fair payment framework (BAPay) for outsourcing computations of cloud computing, and eliminate the third-party, trusted or not, while ensuring the fairness of payment against malicious outsourcer (O) and calculators (C). In our proposed protocol, we use the idea of Bitcoin contract to guarantee that no matter how a malicious O behaves, C will be paid if they are honest. The contributions of this paper are three-fold:

- Firstly, the system architecture, specification and security requirements of BAPay system are proposed, and its design details are described. We demonstrate that BAPay enjoys completeness and fairness where means that the resistance to colluding attacks does not depend on any third party.
- Secondly, in the protocol, it is ensured that the calculators either earns the service fee and gets his guaranty back simultaneously or pays a penalty in the form of deposit to the outsourcer. The proposed protocol enables an automatic penalty to calculators if the outsourcer does not pay as he promised.

– Finally, taking advantage of the anonymity of Bitcoin, we solve the collusion problem of calculators in the outsourcing computation. Our system architecture can be used for general calculations in the outsourcing computation. Besides, the security analysis shows that BAPay achieves completeness and fairness.

1.2 Related Work

In recent years, Bitcoin has attracted widespread attention in recent years because of its anonymity. Bitcoin, an early and successful use of Blockchain technology, is issued under the pseudonym Satoshi Nakamoto [8]. To expand the potential uses of Bitcoin technologies, Ethereum, Smart contracts and related technologies have been proposed. Bonneau et al. [9] first systematically described Bitcoin and cryptocurrencies. Andrychowicz et al. [10,11] constructed a time limit commitment scheme, and also proposed a secure multi-party lottery protocol based on Bitcoin. Similarly, Bentov and Kumaresan [12] showed how to use Bitcoin systems to design fair and secure computing protocols. Note that the research work of these schemes mainly includes two aspects: outsourcing storage and outsourcing computation.

As for outsourcing computation, a number of solutions have been presented recently. There are different validation solutions for different computing tasks. In order to protect the rights and interests of the honest participants, Golle et al. [13,14] proposed a distributed computing outsourcing security model, which used repeated computation to verify the correctness of the calculated results. Szajda et al. [15], Sarmenta et al. [16], proposed a probabilistic verification mechanism for detecting cheaters, but the computational cost was high. In order to improve efficiency, Du et al. [17] proposed a promise-based scheme to prevent server spoofing. Monrose et al. [18] used computational proofs to ensure proper server behavior. Gennaro et al. [19] proposed a verifiable outsourcing computation scheme that protects the privacy of inputs and outputs. Carbunar et al. [20] first considered the payment problem in the outsourcing computation scenario and proposed a fair payment scheme. Chen et al. [21] further proposed a conditional electronic payment system based on restricted partial blind signature scheme.

Note that existing solutions in outsourcing computation payment schemes seem to require trusted third parties to achieve fair payment. For example, banks. However, if the transaction costs in the system are too high, banks are reluctant to implement them. To solve these problems, Dong et al. [22] proposed a protocol for checking the correctness of computing in cloud computing based on game theory and the Ethereum smart contract. Huang et al. [23] proposed a Blockchain-based outsourcing solution that still requires a trusted third party. Chen [24] proposed a Bitcoin-based fair payments for outsourcing computations of fog devices, but this scheme could not really realize decentralized outsourcing services based on Blockchain. While ensuring fairness of payment against malicious outsourcers and calculators, Zhang et al. [25] introduce an outsourcing service payment framework based on Blockchain in cloud computing. In this

paper, we propose an anti-collusion payment protocol based on Bitcoin that can solve the collusive threat of the calculators and ensure the correct execution of the service.

1.3 Organization

The rest of the paper is organized as follows: in Sect. 2, we present the system architecture, definition and security requirements. We propose a Bitcoin-based anti-collusion fair payment framework (BAPay) in Sect. 3. Security analysis is given in Sect. 4. Finally, we give a brief conclusion.

2 System Architecture, Definition and Security Requirements

In this section, we first present the system architecture and definition of BAPay. Then, the security requirements are described in detail.

2.1 System Architecture of BAPay

The main process of our protocol is depicted in Fig. 1, and it involves an outsourcer (i.e., resource-ronstraint user), calculators (i.e., cloud service providers) and a Bitcoin Network [26,27]. The outsourcer come to an agreement with calculators through Bitcoin Network system. Suppose O plans to subscribe an outsourcing service $\boldsymbol{sv}$ from C. The details are given as follows:

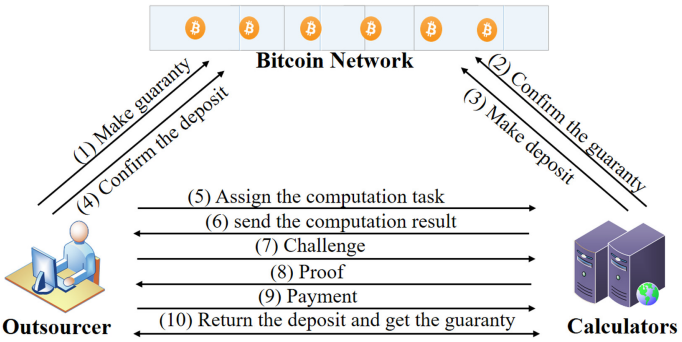

Fig. 1. An example of transaction

2.2 Definition of BAPay

BAPay consists of five phases: the system setup phase, the service implementation phase, the establish payment phase, the service verification phase, and the service redeem phase. The first four phases are compulsory and the service redeem phase is performed by O only if C fails to provide a valid service implementation proof.

System Setup Phase. O and C initialize some parameters such as unredeemed transactions to be used in the subsequent phases.

Service Implementation Phase. ***sv*** is implemented based on three procedures: service subscription, service enforcement and preliminary service confirmation, which are sequentially performed as below.

- ***Service Subscription***: O subscribes ***sv*** from C by sending service-related data to C.
- ***Service Enforcement:*** In this procedure, ***sv*** is enforced by C. Upon receiving the subscription data from O, C enforces ***sv***.
- ***Preliminary Service Confirmation:*** After obtaining the signature from C, O thinks that ***sv*** has been preliminarily implemented, where ***preliminarily*** means that the ***sv*** implementation will be checked by O before the payment.

Establish Payment Phase. In this phase, O and C jointly initiate the service checking and specify the service requirements. A payment protocol is established between O and C to ensure the smooth operation of the outsourcing computation task. At the same time, in order to prevent collusion, O also sets the corresponding anti-collusion mechanism to ensure the real effectiveness of the calculation results.

Service Verification Phase. This phase is performed by C to earn the (partial) service fee from C by proving that the (partial) ***sv*** implementation meets the (partial) requirements. Certainly, O can ensure that the (partial) service fee is paid to C only if the (partial) ***sv*** implementation is what is expected.

Service Redeem Phase. Only if C fails to prove that the (partial) ***sv*** implementation meets the (partial) requirements of O before a specific time, BAPay comes to the service redeem phase. In this phase, O can claim enough deposits from C no matter how C behaves.

2.3 Security Requirements

In BAPay, both O and C are of mutual distrust and they can be malicious. Concretely, the malicious O aims to enjoy ***sv*** provided by C without paying the service fee, and the malicious C wants to get the service fee from O without implementing ***sv*** as specified in the requirements of O. The malicious C may collude with others to cheat O. Hence, for an efficient outsourcing system, it should be satisfy the properties of completeness, fairness and anti-collusion.

- ***Completeness:*** If both O and C are honest, then O can obtain the required service implementation and C can gain the corresponding service fee.

- ***Fairness:*** The fairness for C means that it is infeasible for the malicious O to enjoy valid ***sv*** provided by C without paying the service fee. In the case of the malicious C, the fairness for O means that it is infeasible for C to get the service fee paid by O without providing a valid ***sv*** implementation proof in terms of the requirements of O before a specific time. Particularly, if a malicious C fails to provide such a proof, O is able to get enough compensation or penalty from C.
- ***Anti-collusion:*** If C is honest, then C doesn't know what the real intentions of O. If C is honest but curious, then between two or more C's collusion gain the full result to the private data for O, and O's data security will face a huge challenge.

3 BAPay: Bitcoin-Based Anti-collusion Fair Payment Framework

3.1 Main Idea

When a resource-constrained outsourcer O wants to obtain certain results, the corresponding calculation task is sent to multiple calculators C, and after receiving the correct return results for all the tasks, it calculates the desired results through the returned results. For C, BAPay requires that there is no collusion between the calculators. If all the C receive the computing task collude and share the computing task of the O, they can recover the true intention of the outsourcer O. According to the security requirements, the main challenges to design BAPay include fairness and anti-collusion. This paper attempts to prevent collusion from the perspective of interests, and proposes a Bitcoin script based anti-collusion payment protocol, in which most C participating in collusion will suffer economic losses, so they will not choose to collude with other C.

3.2 Design Details of BAPay

As we know, the Bitcoin script is simple, stack-based and purposefully not Turing-complete. In order to achieve easy understanding and keep the exposition simple, we present a BAPay following the style of Bitcoin transactions.

System Setup Phase. Let H be a cryptographic hash function, such as SHA-256. A secure symmetric encryption algorithm should be chosen for specified services if necessary. O and C choose their own ECDSA public-secret key pairs, denoted by (pk_o, sk_o) and (pk_c, sk_c), respectively. All the parties have come to an agreement on a public key used in receiving Bitcoin.

Service Implementation Phase. The outsourcer O prepares the outsourcing instance F_i. Through a special transaction in Bitcoin system, O creates a Bitcoin contract to form agreements with C. The outsourcing service $\boldsymbol{sv}$ is implemented based on the following three procedures.

- ***Service Subscription:*** O preprocesses service-related local $(q_1, ..., q_n)$ and sends the result $(m_1, ..., m_n)$ to C for sub-scribing $\boldsymbol{sv}$. Note that the preprocessing is specified by concrete outsourcing services. Meanwhile, O also generates a random number R and sends it to each C.
- ***Service Enforcement:*** In this procedure, upon receiving the subscription $(m_1, ..., m_n)$ from O, C is first based on $(m_1, ..., m_n)$ to compute $(r_1, ..., r_n)$ then return it to the O and finish $\boldsymbol{sv}$. Finally, C generates a digital signature according to the enforcement of $\boldsymbol{sv}$, because the transaction Tx can only be redeemed through the signature of C private key sk_c.
- ***Preliminary Service Confirmation:*** After obtaining $(r_1, ..., r_n)$, O considers that $\boldsymbol{sv}$ has been preliminarily implemented, where preliminarily means that $\boldsymbol{sv}$ implementation will be checked by O before the payment.

Establish Payment Phase. This phase is performed by C to earn the service fee from O by proving that the $\boldsymbol{sv}$ implementation meets the requirements. Specifically speaking, O builds a Bitcoin transaction to pay for outsourcing services $\boldsymbol{sv}$. For each C, O creates a transaction Pay_i of value $d\mathbf{B}$, the structure of the transaction Pay_i is shown in Fig. 2.

$\mathbf{T}xPay_i$ (**in**: $\mathbf{T}x$)
in-script: sig_v ($\mathbf{T}xPay_i$)
out-script ($body, \sigma_1, \cdots, \sigma_n, \sigma_U, \sigma_B, \sigma_{S_i}$):
$(\mathrm{ver}_{m_1}(body, \sigma_1) \wedge \cdots \wedge \mathrm{ver}_{m_n}(body, \sigma_n)$
$\wedge \mathrm{ver}_B(body, \sigma_B)) \vee$
$(\mathrm{ver}_{s_i}(body, \sigma_{S_i}) \wedge \mathrm{ver}_U(body, \sigma_U))$
val: d ฿

Fig. 2. The structure of the transaction Pay_i

There are two ways to cash in Pay_i. Under normal circumstances, the transaction will be cashed by $Charge_i$ after time t, which corresponds to the way it is cashed after the symbol $\vee$ in the output script. The structure of the transaction $Charge_i$ is shown in Fig. 3. Note that the transaction body $Charge_i$ is created by O, which then generates a signature about the transaction and sends it to the corresponding C.

Service Verification Phase. C generates a signature about the transaction after checking the transaction $Charge_i$ sent by O. Note that the transaction $Charge_i$ can only be redeemed through the signature of C private key sk_c. Based on $(m_1, ..., m_n)$, C computes $(r_1, ..., r_n)$ and returns it to O. Afterwards, C receives service fees by publishing and cashing in transactions $Charge_i$ after time t.

Of course, the transaction Pay_i can also be cashed in through another the transaction $Collude_i$, which requires collusion between the calculators C because a signature of a set of private keys $(sk_{m_1}, ..., sk_{m_n}, sk_B)$ (i.e. (pk_B, sk_B) is a shared key pair) is required. The structure of the transaction $Collude$ is shown in Fig. 4.

Because the keys are generated by messages m_i, the calculators C who know the messages m_i can compute the corresponding private keys. If the calculators C know exactly what O wants, the best route to collude with each other C and must share the messages m_i freely. Therefore, if a C knows all the messages m_i, as well as the private keys sk_B, then he can cash in the transaction Pay_i. Furthermore, O restores the final result based on the result returned by the calculators C.

$\mathbf{Tx}Charge_i$ (**in**: $\mathbf{Tx}Pay_i$)
in-script: Ø,⋯,Ø, sig_U (T$x$$Charge_i$), Ø, sig_{s_i} (T$x$$Charge_i$)
out-script ($body$,σ_1): ver_{s_i}($body$,σ_1)
val: d ₿
tlock: t

Fig. 3. The structure of the transaction $Charge_i$

$\mathbf{Tx}Collude$ (**in**: $\mathbf{Tx}Pay_i$)
in-script: sig_{m_1} (T$x$$Collude$),⋯, sig_{m_n} (T$x$$Collude$),Ø,$\text{sig}_B$ (T$x$$Collude$),Ø
out-script ($body$,σ_1): ver_{s_i}($body$,σ_1)
val: d ₿

Fig. 4. The structure of the transaction $Collude$

Service Redeem Phase. Although the payment protocol above somewhat inhibits collusion, there is still a issue. The co-conspirators may avoid loss of interest by sharing their results r_i rather than their messages m_i. If they have all the results, they can still get the outsourcer's true intentions. Therefore, the payment protocol must be modified to solve the collusion issue.

Assuming that the calculation returned by the calculators C are $(r_1, ..., r_n)$, according to the previous method of generating sk_{m_i}, the O generates a set of private keys $(sk_{r_1}, ..., sk_{r_n})$ using r_i and R. By swapping the private key of the cashing condition in the previous transaction Pay_i from the private key generated by the $(m_1, ..., m_n)$ for the private key generated by the calculation result $(r_1, ..., r_n)$, then the calculation result can be prevented from being shared by the calculators C. Therefore, we need to convert the transaction Pay_i into a new transaction Pay_i'.

To implement this transformation, the O needs to generate a new transaction Pay_i' instead of the previous transaction Pay_i. The output scripts for Pay_i' and Pay_i is similar, and the transaction structure of Pay_i' is shown in Fig. 5.

To implement Pay_i, the O needs the signature of the private key, so the O sends the transaction Pay_i to either calculators C for the signature. The case where the calculators C does not sign this transaction Pay_i will be discussed in the security analysis. Similarly, there are two ways to implement the transaction Pay_i'. The first way is the same as $Charge$, and the second way is similar to $Collude$, replacing sk_{r_i} with the private key sk_{m_i}. The transaction structure of $Collude_i'$ is shown in Fig. 6. In this case, if a C has enough r_i, he can implement the transaction Pay_i'.

Note that this protocol applies to payment agreements that can tolerate collusion by $n-1$ calculators C, that is, if not all calculators collude, they will not be able to recover the outsourcer O true results. In addition, it is also possible to set tolerable $l-1$ calculators collusion based on the idea of threshold, which is a case for further discussion.

$\mathbf{T}xPay'_i$ (**in**: $\mathbf{T}x$)
in-script: sig_{m_1} ($\mathbf{T}xPay'_i$), $\cdots$, sig_{m_n} ($\mathbf{T}xPay'_i$), sig_B ($\mathbf{T}xPay'_i$)
out-script ($body, \sigma_1, \cdots, \sigma_n, \sigma_U, \sigma_B, \sigma_{S_{l'}}$): ($\mathrm{ver}_{r_1}(body, \sigma_1) \wedge \cdots \wedge \mathrm{ver}_{r_n}(body, \sigma_n)$ $\wedge \mathrm{ver}_B(body, \sigma_B)) \vee$ ($\mathrm{ver}_{s_l}(body, \sigma_{S_l}) \wedge \mathrm{ver}_U(body, \sigma_U)$)
val: d ฿

Fig. 5. The structure of the transaction Pay_i'

$\mathbf{T}xCollude'$ (**in**: $\mathbf{T}xPay'_i$)
in-script: sig_{r_1} ($\mathrm{T}xCollude'$),$\cdots$, sig_{r_n} ($\mathrm{T}xCollude'$),Ø,sig_B ($\mathrm{T}xCollude'$),Ø
out-script ($body, \sigma_1$): $\mathrm{ver}_{s_l}(body, \sigma_1)$
val: d ฿

Fig. 6. The structure of the transaction $Collude_i'$

In this paper, the BAPay protocol based on Bitcoin mainly takes advantage of the anonymity of Bitcoin, which makes it impossible for calculators of the collusion to distinguish which a calculator has cashed in the transaction and taken away all the service fees. In other words, if the calculators collude, only one C will cash in on the transaction and take the fees, and all other calculators will lose the corresponding fees. From the perspective of interests, these calculators will not collude.

4 Security Analysis

Theorem 1. *The proposed Bitcoin-based anti-collusion fair payments protocol (BAPay) satisfies the security requirement of completeness.*

Proof. Completeness refers to the completeness of the structure of the protocol, which can achieve the purpose of the protocol. In other words, the O can not only obtain the calculations, but also guarantees their privacy. At the same time, the calculators C can get the corresponding service fee.

In normal case, both O and C follow the protocol process. O sends the outsourcing tasks and corresponding parameters to the C, and publishes the transaction on the Bitcoin network. According to the requirements of the outsourcing task, C calculates and returns the calculations to O. After t time, C will exchange the transaction $Charge$ for the transaction Pay_i', and all C will obtain corresponding Bitcoin from O for his reward, and O will get back his deposit. At last, O gets the correct return result and restores the desired result.

Theorem 2. *Based on the anonymity of Bitcoin, Bitcoin-based anti-collusion fair payments protocol (BAPay) satisfies the the security requirement of fairness.*

Proof. We use Bitcoin script transactions to achieve fairness. If either O or C does not follow the normal process of the protocol, it will suffer the loss of interests or terminate the protocol to protect the interests of the participants.

Assuming that O does not collude with C, C will honestly return the correct results. But they are curious, and may collude with each other to obtain C private information. Both participants may not follow the normal procedures of the protocol. Two cases should be taken into account:

- *Case* 1. O does not follow the normal flow of the protocol.
  - If O does not build the payment protocol Pay for a payment, then C will not provide an outsourcing service and the protocol terminates $\perp$.
  - If O does not implement in the transaction Pay, it will be converted to the transaction Pay'. However, in order to prevent the disclosure of O privacy and prevent C from sharing the calculation results. Generally speaking, O chooses to convert the transaction Pay into Pay'. Otherwise, the protocol terminates $\perp$. At the same time, C can still get the reward, the privacy of O can no longer be guaranteed.
- *Case* 2. C does not follow the normal flow of the protocol. In the protocol, O sends the signature $Charge'$ to C, which signs the transaction and publishes it. If C does not sign the transaction, the agreement terminates $\perp$. Of course, C does not obtain the service reward.
  - If C chooses to collude and share the received message m_i, then one of C can redeem all transactions Pay paid to each C through the transaction $Charge$, thus the other C will lose their due service reward. From the perspective the interests of the individual, all of C do not choose collusion.
  - If C chooses to collude and share the received message r_i, similarity, one of C can redeem all transactions Pay' paid to each computing party through the transaction $Charge'$, thus the other C will lose their due service reward. As discussed earlier, when C converts the transaction Pay to the transaction Pay', the signature of the private key sk_c is required. If neither C signs the transaction as required by the protocol and shares the results, they still seem to know the results of O calculations and obtain the service reward. But says in practical terms, the C who knows all the returned results can still cash in the fees payable to other C. From the perspective the interests of the individual, C also does not choose collusion.

- If C negotiates in advance to avoid one C withdrawing the reward due to other C. Because of the anonymity of Bitcoin, they could not tell which of the co-conspirators cashed in. So it's hard to use a deposit or any other way to prevent one of the calculators from taking away the reward.

According to the above proof, the collusion C will lose its reward. Furthermore, from the perspective the interests of the individual, C does not collude. Therefore, it is reasonable to restrain collusion through interests in the protocol.

5 Conclusion

In this paper, we propose an anti-collusion fair payments framework based on Bitcoin. To be specific, we introduce the architecture of BAPay and describe the design details of BAPay. Under our protocol, the resource-constraint user pays the service fee through the Bitcoin, and the transaction can be exchanged under different conditions. In addition, by taking advantage of the excellent characteristics of Bitcoin and combining with the multi-signature scheme, we have solved the issue of the calculators collusion in the outsourcing computation. Our security analysis indicate that BAPay achieves completeness and fairness. In the future, it is interesting to address the issue of payment equity based on Bitcoin, Blockchain and Smart contract technologies in more complex cloud.

References

1. Zhang, Y.H., Deng, R., Liu, X.M., Zhang, D.: Blockchain based efficient and robust fair payment for outsourcing services in cloud computing. Inf. Sci. **462**, 262–277 (2018)
2. Armbrust, M., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
3. Huang, Z., Liu, S., Mao, X., Chen, K., Li, J.: Insight of the protection for data security under selective opening attacks. Inf. Sci. **412**, 223–241 (2017)
4. Li, J., Zhang, Y., Chen, X., Xiang, Y.: Secure attribute-based data sharing for resource-limited users in cloud computing. Comput. Secur. **72**, 1–12 (2018)
5. Zhang, Y., Li, J., Chen, X., Li, H.: Anonymous attribute-based proxy reencryption for access control in cloud computing. Secur. Commun. Netw. **9**(14), 2397–2411 (2016)
6. Ateniese, G., et al.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)2007, pp. 598–609. ACM (2007). https://doi.org/10.1007/978-1-59593-703-2/07/0011
7. Chen, X., Li, J., Huang, X., Ma, J., Lou, W.: New publicly verifiable databases with efficient updates. IEEE Trans. Dependable Secure Comput. **12**(5), 546–556 (2015)
8. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
9. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., Felten, E.W.: SoK: research perspectives and challenges for bitcoin and cryptocurrencies. In: IEEE Symposium on Security and Privacy 2015, San Jose, USA, pp. 104–121 (2015). https://doi.org/10.1109/SP.2015.14

10. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multiparty computations on bitcoin. In: IEEE Symposium on Security and Privacy(SP)2014, San Jose, USA, pp. 443–458 (2014). https://doi.org/10.1109/SP.2014.35
11. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, Ł.: Fair two-party computations via bitcoin deposits. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014. LNCS, vol. 8438, pp. 105–121. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44774-1_8
12. Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 421–439. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_24
13. Golle, P., Stubblebine, S.: Secure distributed computing in a commercial environment. In: Syverson, P. (ed.) FC 2001. LNCS, vol. 2339, pp. 289–304. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46088-8_23
14. Golle, P., Mironov, I.: Uncheatable distributed computations. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 425–440. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_31
15. Szajda, D., Lawson, B., Owen, J.: Hardening functions for large scale distributed computations. In: Symposium on Security and Privacy 2003, Berkeley, USA, pp. 216–224 (2003). https://doi.org/10.1109/SECPRI.2003.1199338
16. Sarmenta, L.F.G.: Sabotage-tolerance mechanisms for volunteer computing systems. Future Gener. Comput. Syst. **18**(4), 561–572 (2002)
17. Du, W., Jia, J., Mangal, M., Murugesan, M.: Uncheatable grid computing. In: Proceedings of the 24th International Conference on Distributed Computing, Systems, ICDCS 2004, pp. 4–11. IEEE Computer Society (2004). https://doi.org/10.1109/ICDCS.2004.1281562
18. Monrose, F., Wyckoff, P., Rubin, A. D.: Distributed execution with remote audit. In: Proceedings of the 1999 ISOC Network and Distributed System Security Symposium (NDSS) 1999, San Diego, pp. 103–113 (1999)
19. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_25
20. Carbunar, B., Tripunitara, M.V.: Payments for outsourced computations. IEEE Trans. Parallel Distrib. Syst. **23**(2), 313–320 (2012)
21. Chen, X., Li, J., Susilo, W.: Efficient fair conditional payments for outsourcing computations. IEEE Trans. Inf. Forensics Secur. **7**(6), 1687–1694 (2012)
22. Dong, C., Wang, Y., Aldweesh, A., McCorry, P., Moorsel, A.: Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS) 2017, pp. 211–227. ACM (2001). arXiv:1708.01171v4
23. Huang, H., Chen, X., Wu, Q., Huang, X., Shen, J.: Bitcoin-based fair payments for outsourcing computations of fog devices. Future Gener. Comput. Syst. **78**, 850–858 (2018)
24. Chen, X., Li, J., Ma, J., Lou, W., Wong, D.S.: New and efficient conditional e-payment systems with transferability. Future Gener. Comput. Syst. **37**, 252–258 (2014)
25. Zhang, Y., Deng, R., Liu, X., Zheng, D.: Outsourcing service fair payment based on blockchain and its applications in cloud computing. IEEE Trans. Serv. Comput. **8**(7), 1–14 (2018)

26. Ding, J., Yu, N., Lin, X., Zhang, W.: A private information retrieval and payment protocol based on bitcoin. Inf. Secur. **4**(06), 1–9 (2019)
27. Quick, D., Choo, K.R.: Digital droplets: microsoft skydrive forensic data remnants. Future Gener. Comput. Syst. **29**(6), 1378–1394 (2013)