Smart Blockchain-Based Information Flow Control for SOA

Khaoula Braiki[1] and Olfa Dallel[2]
[1]*UFR SEGMI, Nanterre University, France*
[2]*ENISo, University of Sousse, Tunisia*

Keywords: Information Flow Control, Secure Service Composition, IoT, Blockchain, Interference.

Abstract: The Internet of Things (IoT) integrates smart devices that collect real time data from the environment. These data are leveraged to propose innovative services which transform the individual lives in a particular context such as smart homes. The Service Oriented Architecture (SOA) is adopted to support the composition of services. However, the service composition faces the problem of security, where data can illegitimately be shared with unauthorized services. This problem is called interference. The key challenge is to ensure end-to-end security which will guarantee the confidentiality and integrity of data. In this paper, we ensure the service binding in a blockchain-based SOA architecture and propose an approach based on the information flow control to protect data confidentiality. Service provider can express the service binding requirements by considering the service provider, the domain, the trust degree and the type of the operation to perform in order to secure the service composition. Moreover, we propose to apply a binding mode: a rule-based binding mode or smart binding based on a machine learning decision tree model. To avoid the interference issue, we integrate a non-interference verification mechanism by assigning a security level for each service. Our smart blockchain-based information flow control approach guarantees the confidentiality and integrity of information in IoT systems.

1 INTRODUCTION

The blockchain is based on the distributed ledger technology. It is a decentralized and shared a data ledger that records transactions and is which maintained by multiple nodes in a network (public, private, hybrid, consortium). This technology secures the exchange of information from cyberattacks by restricting the number of users sharing it. However, this security policy can be violated if there is an information leak from one application to another or by communicating secret data from a user to an unauthorized one. This problem is called interference.

For instance, in the context of the blockchain for the Internet of Things (IoT), we exemplify a smart home scenario. The smart home technology is based on the IoT, where devices are interconnected via the internet. The IoT allows the interconnected devices to send, receive and share data automatically in order to offer smart services to homeowners. Added to that, the blockchain tends to collect and store information and data. Let us consider a scenario of smart home application that is composed of units to control and manage the temperature of the house based on the presence of people. Each unit is represented by a service. Indeed, an *"Analyzer"* service plays the role of a server and handles the information received from a *"Temperature"* service and a *“Presence”* service. The application must adjust the temperature of the home at a certain threshold depending on the presence of the person. Consequently, the *“Analyzer”* service will make the decision to decrease or increase the temperature to optimize the energy usage. However, the data exchanged between services are private, and the *“Analyzer”* service is authorized to read information from the *"Temperature"* service and the *“Presence”* service to adjust the temperature value. Therefore, the *"Analyzer"* service is allowed to read the *“Presence”* service data. The *“Presence”* service cannot control how the *"Analyzer"* service distributes the information that has been read, which could disclose the presence status of a person in their home. Thus, in this scenario, there is a risk of interference that allows the disclosure of secret personal information of a person. In this case, we should secure and control the propagation of information.

Several techniques have been proposed to prevent information disclosure (Shari and Malip, 2022), such

Braiki, K. and Dallel, O.
Smart Blockchain-Based Information Flow Control for SOA.
DOI: 10.5220/0012767500003753
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 19th International Conference on Software Technologies (ICSOFT 2024)*, pages 531-538
ISBN: 978-989-758-706-1; ISSN: 2184-2833
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

as access control lists, firewalls, and cryptography. Nevertheless, all these techniques do not guarantee the propagation of information. The control of information propagation is supported by Information Flow Control (IFC) (Myers and Liskov, 1997). IFC allows the providers to track and control the propagation of data to a fine granularity throughout the system using the security labels that represent the data security levels. In fact, IFC aims to assign a certain category to a data in terms of:

- Confidentiality: The value of secret information should not be assigned to a public variable.
- Integrity: Only users who are authorized can modify the data content.

To control the flow of data, IFC tends to classify data into different security levels. Generally, this technique assigns two classes to classify data: one class for public information with a low security level L (Low), and another class for secret information with a high level of security H (High). If we consider that class H is more restrictive than class L, then the allowed flows are: from L to L, from L to H and from H to H. The flow from H to L is not allowed because it will disclose the secret information of class H.

In this paper, our main contributions are as follows:

- We propose an approach based on IFC. In this approach, we present a new program level model based on a security policy which defines the formal specification of the security requirements for the interference problem.
- We propose an algorithm which checks whether a given non-interference policy is satisfied or not.
- We suggest that services are published in a service repository and the access is ensured through the Service Oriented Architecture (SOA) implemented as an application-specific blockchain (cosmos (Kwon and Buchman, 2019)).
- We extend the SOA architecture by adding a new module to verify the non-interference security requirements.

The rest of the paper is organized as follows. In section 2, we review the related literature. In section 3, we formally define a security policy model for the SOA . Our motivated scenario is presented in section 4. In section 5, we present our extended SOA architecture and explain how the non-interference security mechanism is applied. In section 6, we provide the simulation results. We conclude the paper with a summary and some future work in section 7.

2 RELATED WORK

The IoT faces the challenge of security, which constitutes research problems (Issa et al., 2023), (Siwakoti et al., 2023). We find a variety of solutions depending on the context and the IoT application problem (Sadeghi-Niaraki, 2023). In this paper, we present some work focusing especially on data sharing in IoT applications based on the blockchain (Mathur et al., 2023), (Shari and Malip, 2022). Based on the decentralization property of the blockchain, researchers have proposed a lot of data processing methods to preserve personal information.

Xie et al. (Xie et al., 2023) put forward a TEE-and-blockchain-supported data sharing system for the IoT. In fact, the consortium blockchain was used to secure user communications based on access control. Moreover, the TEE applied the Intel SGX to build SDSS for reducing the storage pressure and to guarantee the security of the off-chain data.
Si et al.(Si et al., 2019) suggested a lightweight information sharing security mechanism to protect the source data collection and information transactions. The data blockchain used a consensus mechanism to form data books to prevent human tampering or destruction of collected data. Besides, the chain utilized a distributed accounting system to achieve tamper resistance and traceability of bills.

Liu et al.(Liu et al., 2022) proposed a new solution for blockchain-enabled information sharing. This solution guaranteed anonymity, entity authentication, data privacy, data trustworthiness, participant stimulation and fairness in a zero-trust context. It could also detect and filter fabricated information through effective voting, smart contracts and consensus mechanisms. These mechanisms aimed to penalize and blacklist unauthenticated participants from sharing garbage information.

Luo et al.(Luo et al., 2019) put forward a new accountable data sharing scheme based on the blockchain and SGX. The proposed schema did not require a trusted third party as the records of users' data sharing acts were tamper-resistant. Furthermore, the confidentiality of data sharing was ensured by SGX.

Cha et al. (Cha et al., 2021) suggested an approach based on both the blockchain and the cloud for protecting and securing secret personal information. To secure the data of users from the cloud Service Provider (SP), the authors used a secret dispersion algorithm to recover the original data even if a particular cloud SP modulated or lost the data and to verify the integrity of data stored by the cloud SP.

Cecchetti et al.(Cecchetti et al., 2020) suggested an approach based on IFC to control security in blockchain smart contracts. The authors described how to reduce vulnerabilities while retaining the assurance of IFC by restricting designated entry points.

Although researchers have used the blockchain technology to solve the problem of IoT information sharing security, the interference problem is still a challenge, where the propagation of data is not ensured. In fact, we shall ensure the security of sharing data with respect to confidentiality and integrity. A Decentralized Label Model (DLM) (Myers and Liskov, 1997) is adopted for the interference problem, which provides a convenient way for expressing the security user requirements. It enables the tracking and control of data propagation at a fine granularity. In fact, we distinguish two categories of blockchains: Decentralized Applications (DApps) and application-specific blockchains. For DApps blockchains, such as Ethereum (Wood et al., 2014), the developer cannot adjust the execution of transactions, update the state or build complex applications by writing smart contracts. The application-specific blockchains, built using the Cosmos-SDK framework (Kwon and Buchman, 2019), allow the developer to build their application as part of the blockchain and customize the business logic of the application. In contrast to the solutions reported in (Cecchetti et al., 2020) and (Tolmach et al., 2021), which were proposed for smart contract blockchains, our solution is based on the application-specific blockchain.

3 SECURITY POLICY MODEL FOR SOA

In this paper, we adopt the SOA for the composition of services. To ensure a distributed environment for the SOA, we employ the blockchain technology where the blockchain ledger is leveraged as a service repository. In this latter, the service descriptions are published by the SP. The SP must specify and manage the security policies of the corresponding service. The defined policies will be applied in the service composition.

The DLM (Myers and Liskov, 1997) is used to support the computation in an entrusted environment by annotating data with labels. Each service can define and express security labels of its data using security policies. According to the DLM, security policies are expressed in terms of principals.

- **Principals**

Principals represent the main entities that own, write and read information. They can also release information to other principals and *act-for* them.
For example, let *Presence*, *Temperature* and *Analyzer* represent three *principals*. If *Analyzer* can *act-for Presence*, then *Analyzer* inherits all the privileges of *Presence*.

This *act-for* relation is represented by symbol $\succeq$ and is expressed formally as $Analyzer \succeq Presence$. That is to say, it enables principals to delegate their information ownership to other principals according to trust relationships. These relationships are defined by a principal hierarchy, which is used to model groups of principals and roles.

- **Labels**

In the SOA, the principals can be the SP or Service Consumer (SC). Both principals can create labels to annotate and protect their data. The SP can manage the service labels, while the SC imposes the required labels to ensure the security policy at a fine-granularity. In the rest of the paper, we consider the *"Analyzer"* as an SC and the *"Temperature"* and *"Presence"* are SPs. Label L is expressed by a set of *principals* to define a set of policies. A policy is associated by an *owner* and a set of *readers*. The *owner* represents the ownership of the data and the *readers* are the allowed principals to read these data. To simplify the notations, we are interested in confidentiality; it is well known that integrity policy is treated dually (Myers and Liskov, 1997).

For example, let *Presence*, *Temperature* and *Analyzer* represent three *principals*, where $L1 = \{Temperature \rightarrow Analyzer, Presence\}$ and $L2 = \{Presence \rightarrow Analyzer\}$ denote labels. The *owners* of labels $L1$ and $L2$ are *Temperature* and *Presence*, respectively. The reader set for the owner *Temperature* is *Analyzer* and *Presence* ($readers(L1, Temperature) = \{Analyzer, Presence\}$). For the owner *Presence*, the reader set is *Analyzer* ($readers(L2, Presence) = \{Analyzer\}$).

This label structure enables owners to specify an independent flow policy in order to trace the propagation of their data through the system during computation. In fact, labels are ordered using the *more restrictive than* relation, represented by symbol $\subseteq$. Given two labels L1 and L2, we say L1 is *more restrictive than* L2 iff L1 has more readers and fewer owners than L2. This means that the owners of L1 are included in L2 and the readers of L2 are included in those of L1, which can be expressed as follows:

$$owner(L1) \subseteq owner(L2)$$
$$\forall O \in owner(L1), readers(L1) \supseteq readers(L2)$$

For instance, if $L1 = \{Temperature \rightarrow Analyzer, Presence\}$ and $L2 = \{Presence \rightarrow$

$Analyzer\}$, we have $L1 \subseteq L2$.

- Derived labels

When the program computes two values respectively labeled with L1 and L2, the result should have the least restrictive label L which enforces the policies defined in L1 and L2. The least restrictive label of $L1, L2$ is $L1 \cup L2$. In fact, the owner set of $L1 \cup L2$ is the union of the owner sets of L1 and L2.

$$owners(L1 \cup L2) = owners(L1) \cup owners(L2)$$

In addition, the set of readers is the intersection of their corresponding reader sets.

$$readers(L1 \cup L2, o) = readers(L1, o) \cap readers(L2, o)$$

- Re-labeling

Re-labeling is the process of changing the security labels of data based on certain rules. In fact, the assignment of a value to a variable is re-labeling, during the system computation. Re-labeling must be safe. In fact, we distinguish two re-labeling rules: (1) restriction which defines the legality of assignment, and (2) declassification which allows an owner to modify their flow policy:

Rule1: Restriction. Re-labeling from L2 to L1 is legal if it is restriction: $L1 \subseteq L2$. Restriction aims to remove/add readers, remove/add owners and/or add policies.

However, this rule may increase restriction and make the data unreadable. Hence, the principals that own the data may need to relax their policies so that other principals can read them. This is done by a declassification rule.

Rule2: Declassification. This kind of re-labeling aims to add readers for some owners O or remove other owners. Declassification can be done only if the label ownership $act - for\ O$.

Declassification depends of the principal hierarchy with the *act-for* relation and requires a special privilege to be performed. In this rule, principals can delegate their information ownership to other principals with regard to a trust relation. According to the hierarchy of principals, information can be relabeled in a safe way. In fact, owner O can be replaced by owner O' iff O trusts O'.

4 MOTIVATION SCENARIO

To illustrate the main idea of IFC, we propose a use case scenario of a smart home, as depicted in Figure 1. For the sake of simplicity, we suppose that the

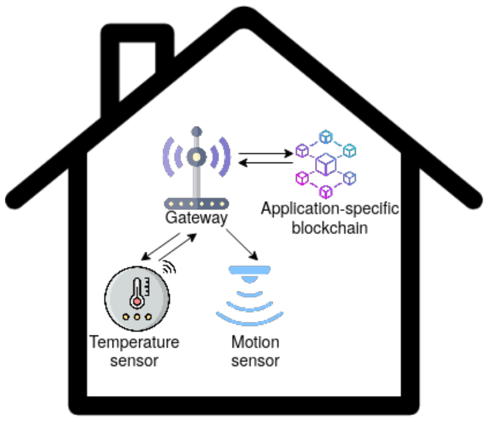

Figure 1: Use case scenario: Smart home.

smart home is equipped with a motion sensor to detect the presence of a person and a temperature sensor to adjust heat or air conditioning. These sensors are connected to a gateway which collects the data and sends it to the application service. The *"Temperature"* service and *"Presence"* service descriptions are published by the SP and stored in the blockchain ledger to make them available for the service consumers. To optimize the energy consumption, the *"Analyzer"* service, considered as the SC, periodically retrieves the presence data from the *"Presence"* service and the temperature value from the *"Temperature"* service. Accordingly, it adjusts the temperature value at home. For example, if no one is at home (i.e., presence is equal to 0) and the temperature value is greater than or equal to 23 degrees Celsius, the *"Analyzer"* service reduces the home temperature to 15 degrees Celsius. The *"Temperature"* service is critical, since according to the temperature value (i.e. public information), the presence of a person at home (i.e. confidential information) can be easily deduced. For instance, a temperature value equals to 15 degrees Celsius means that no one is at home, while a temperature value equals to 23 degrees Celsius is an indicator that an occupant exists at home. Deducing secret information from public information refers to the interference security problem. To avoid the interference issue, the *"Temperature"* service and the *"Presence"* service must have the same high security level to make the temperature value and presence information confidential. Therefore, we propose to integrate a non-interference validation process to IFC in order to check whether the security levels of the *"Presence"* service and *"Temperature"* service are compatible, so as to allow the *"Analyzer"* service to perform the required adjustments on the temperature value.

5 BLOCKCHAIN-BASED SOA FOR IoT APPLICATION

The SOA architecture is implemented as an application specific blockchain. It consists of two main modules: the repository module and the validation module, as illustrated in Figure 2. Table 1 presents the main transactions provided by the application-specific blockchain-based SOA architecture.

5.1 Repository Module

The repository module provides two services: service management and service discovery.

5.1.1 Service Management

The service management allows the SP to manage the service description. The service description involves the metadata of the service, such as the label, the provider, the domain, the level and the trust degree. The service level can be high (H) or low (L). A high-level service means that the service involves sensitive data that can be shared only with the services having the same security level. A low-level service refers to a service that can be shared with other services. The trust degree is a score computed, as proposed in (Adewuyi et al., 2021), based on the workflow, relevant partial trust scores of the SPs and the weights assigned by the service requester. The trustworthiness management of services is out of the scope of this paper.

To publish a service, the SP sends a *Tx_publish_service* transaction containing the service description. Once the transaction is successfully executed, the service description is stored in the service store. The SP can update or delete the service metadata by sending a *Tx_update_service* transaction or a *Tx_delete_service* transaction, respectively.

Additionally, the SP can optionally define a set of binding rules that represent the requirements needed to be fulfilled by the SC in order to allow service binding. The service binding requirements can be the provider, the domain, the trust degree and the type of the operation to be performed on the requested service, for instance the get() operation to retrieve the current temperature value and the set() operation to increase or decrease the temperature value. The SP adds, updates and deletes a binding rule by sending a *Tx_add_binding_rule* transaction, a *Tx_update_binding_rule* transaction, a *Tx_delete_binding_rule* transaction, respectively.

Table 1: Transactions provided by blockchain-based SOA.

Transaction	Module	Role	Actor
Tx_publish_service	Repository module	Add new service	Service provider
Tx_update_service	Repository module	Update service metadata	Service provider
Tx_delete_service	Repository module	Delete service	Service provider
Tx_add_binding_rule	Repository module	Add new service requirement rule	Service provider
Tx_update_binding_rule	Repository module	Update service requirement rule	Service provider
Tx_delete_binding_rule	Repository module	Delete service requirement rule	Service provider
Tx_train	Validation module	Generate classification model for service binding	Service provider
Tx_find_service	Repository module	Find service	Service consumer
Tx_check_non_interference	Validation module	Validate binding requirements and service security levels	Service composition

5.1.2 Service Discovery

The service discovery allows a service consumer to find a service by sending a *Tx_find_service* transaction. It is out of the scope of this paper.

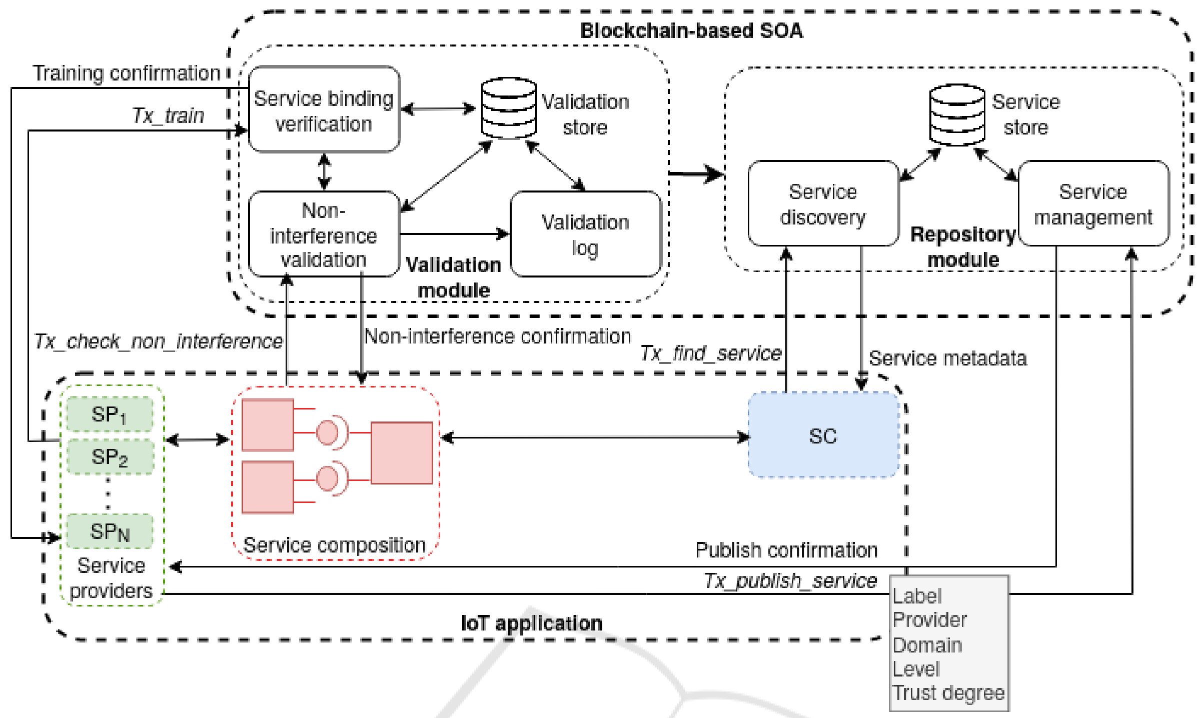

Figure 2: Blockchain-based SOA for an IoT application.

5.2 Validation Module

The validation module provides three services: validation log, service binding verification and non-interference validation.

5.2.1 Validation Log

The validation log service records all the non-interference validation transactions. Each log contains the metadata of the service requester as well as the metadata of the requested services, the non-interference decision and the validation operation timing. The validation logs are stored in the validation store to be used by the service binding verification.

5.2.2 Service Binding Verification

If the binding rules are not specified, the SP can send a *Tx_train* transaction to the service binding verification in order to generate a smart binding decision model for a specific service. The service binding verification retrieves the non-interference validation logs associated with the specified service from the validation store and trains a machine learning decision tree.

5.2.3 Non-Interference Validation

To bind the SC to the requested services, the service composition sends a transaction'*Tx_check_non_interference* to the non-interference validation service. This latter verifies the binding requirements and checks the compatibility of the security levels. Algorithm 1 represents the pseudocode of the validation process.

- Step 1: The validation process starts by extracting the metadata associated with the service requester and the requested services from the service store in the repository module.
- Step 2: The process verifies the binding requirements. We propose two binding verification modes: a rule-based binding mode and a smart verification binding mode. The rule-based mode leverages the binding rules specified by the SP, whereas the smart verification mode is based on the decision tree model generated from the verification logs. If the SP specifies a rule-based mode, the process extracts the set of binding rules, iterates over the rules, checks if there is at least one selection rule that satisfies the requirements, and returns the binding decision. In the second case, when the SP selects a smart verification binding mode, the process inquires the service binding verification for a binding decision. For the two binding modes, the binding decision can be ”To bind” if the binding requirements are satisfied, and ”Not to bind” otherwise. If the binding decision is ”Not to bind”, the validation process re-

turns ”Rejected” as a composition decision. Otherwise, the process checks the compatibility of the security levels.

- Step 3: The security level compatibility imposes that the service holding a high level (H) can bind any service whatever its security level, while the service holding a low level can bind only services holding a low level (L). If the security levels are compatible, then ”Accepted” is returned as a composition decision. Otherwise, the composition decision is ”Rejected”.
- Step 4: The validation operation is logged by the validation log service.

```
Data: $ServiceRequester$, $RequestedService1$,
      $RequestedService2$
Result: $Decision$
$metadata \leftarrow$
  ExtractMetadata($ServiceRequester$,
  $RequestedService1$, $RequestedService2$);
$v \leftarrow$ VerifyBinding($metadata$);
if $v == true$ then
    $Decision \leftarrow$
      CheckNonInterference($metadata$);
else
    $Decision \leftarrow$ ”Rejected”;
end
SaveValidationLog();
Return $Decision$;
```

Algorithm 1: Non-interference validation algorithm.

6 SIMULATION

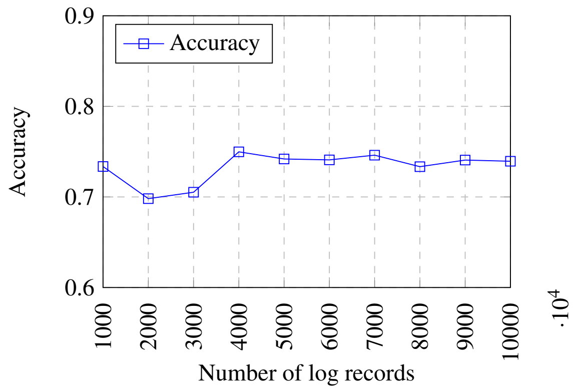

Figure 3: Accuracy of decision tree classification model.

We implement our proposed IFC for an SOA on an application-specific blockchain using the Cosmos-SDK framework. Then, we evaluate the computation time leveraged to process the service requests by varying the number of services. To verify the reliability and accuracy of the smart binding mode, we evaluate the accuracy of the decision-tree-based classification model in terms of number of non-interference validation log records. In the training phase, we use the log records generated from the non-interference validation operations performed using the rule-based mode. Figure 3 shows that the minimum accuracy value is equal to 0.7 and the maximum accuracy value is equal to 0.75. We remark that the overall accuracy is constant while increasing the number of log records. Therefore, we conclude that the smart binding mode based on a decision tree model provides acceptable decision results. Figure 4 depicts the execution time needed for the binding requirement verification and the non-interference validation. We conclude that increasing the number of services from 100 to 1000 results in a slight rise in the time processing from 0.03s to 3.10s for the smart binding and from 0.05s to 4.25s for the rule-based binding, respectively. Therefore, the smart binding mode reduces the execution time compared to the execution time of the rule-based mode. This comes down to the fact that the rule-based mode requires more time to check all the service binding rules specified by the SP.

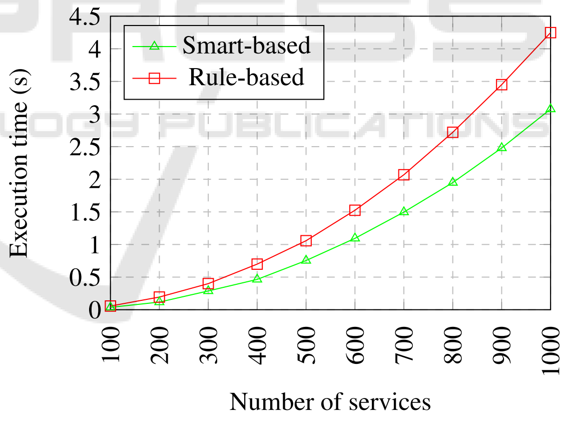

Figure 4: Execution time.

7 CONCLUSION

In this paper, we put forward an application-specific blockchain-based SOA architecture for IoT systems, such as the smart home. Our blockchain-based SOA allows the SP to manage the service descriptions. In addition, it enables the SP to express a set of binding rules or leverage a machine learning classification model for the service binding. To prevent the information interference, we propose to associate a security level with each service, and then check the security level compatibility between the requested services and the service requester. Our IFC mechanism ensures the confidentiality and integrity of the data in the IoT application. As future work, we will extend our IFC mechanism by considering more security objectives like the privacy of data.

REFERENCES

Adewuyi, A. A., Cheng, H., Shi, Q., Cao, J., Wang, X., and Zhou, B. (2021). Sc-trust: a dynamic model for trustworthy service composition in the internet of things. *IEEE Internet of Things Journal*, 9(5):3298–3312.

Cecchetti, E., Yao, S., Ni, H., and Myers, A. C. (2020). Securing smart contracts with information flow. In *International Symposium on Foundations and Applications of Blockchain*.

Cha, J., Singh, S. K., Kim, T. W., and Park, J. H. (2021). Blockchain-empowered cloud architecture based on secret sharing for smart city. *Journal of Information Security and Applications*, 57:102686.

Issa, W., Moustafa, N., Turnbull, B., Sohrabi, N., and Tari, Z. (2023). Blockchain-based federated learning for securing internet of things: A comprehensive survey. *ACM Computing Surveys*, 55(9):1–43.

Kwon, J. and Buchman, E. (2019). Cosmos whitepaper. *A Netw. Distrib. Ledgers*, 27.

Liu, Y., Hao, X., Ren, W., Xiong, R., Zhu, T., Choo, K.-K. R., and Min, G. (2022). A blockchain-based decentralized, fair and authenticated information sharing scheme in zero trust internet-of-things. *IEEE Transactions on Computers*, 72(2):501–512.

Luo, Y., Fan, J., Deng, C., Li, Y., Zheng, Y., and Ding, J. (2019). Accountable data sharing scheme based on blockchain and sgx. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 9–16. IEEE.

Mathur, S., Kalla, A., Gür, G., Bohra, M. K., and Liyanage, M. (2023). A survey on role of blockchain for iot: Applications and technical aspects. *Computer Networks*, 227:109726.

Myers, A. C. and Liskov, B. (1997). A decentralized model for information flow control. *ACM SIGOPS Operating Systems Review*, 31(5):129–142.

Sadeghi-Niaraki, A. (2023). Internet of thing (iot) review of review: Bibliometric overview since its foundation. *Future Generation Computer Systems*.

Shari, N. F. M. and Malip, A. (2022). State-of-the-art solutions of blockchain technology for data dissemination in smart cities: A comprehensive review. *Computer Communications*, 189:120–147.

Si, H., Sun, C., Li, Y., Qiao, H., and Shi, L. (2019). Iot information sharing security mechanism based on blockchain technology. *Future generation computer systems*, 101:1028–1040.

Siwakoti, Y. R., Bhurtel, M., Rawat, D. B., Oest, A., and Johnson, R. (2023). Advances in iot security: Vulnerabilities, enabled criminal services, attacks and countermeasures. *IEEE Internet of Things Journal*.

Tolmach, P., Li, Y., Lin, S.-W., Liu, Y., and Li, Z. (2021). A survey of smart contract formal specification and verification. *ACM Computing Surveys (CSUR)*, 54(7):1–38.

Wood, G. et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32.

Xie, H., Zheng, J., He, T., Wei, S., and Hu, C. (2023). Tebds: A trusted execution environment-and-blockchain-supported iot data sharing system. *Future Generation Computer Systems*, 140:321–330.