DRMF: a Distributed Resource Management Framework for industry 4.0 environments

Asma Lahbib, Khalifa Toumi, Anis Laouiti, Steven Martin

DRMF: A Distributed Resource Management Framework for Industry 4.0 Environments

Asma LAHBIB*, Khalifa Toumi**, Anis Laouiti*, and Steven Martin***

*SAMOVAR, Télécom SudParis, CNRS, Université Paris-Saclay, 9 rue Charles Fourier 91011 Evry, France.
Email: {asma.lahbib, anis.laouiti}@telecom-sudparis.eu
**IRT SystemX, 8 Avenue de la Vauve, 91127 Palaiseau, France.
Email: {khalifa.toumi}@irt-systemx.fr
***LRI, Université Paris-Sud, 15 Rue Georges Clemenceau, 91400 Orsay, France.
Email: {steven.martin}@lri.fr

***Abstract*— While smart factories are becoming widely recognized as a fundamental concept of Industry 4.0, their implementation has posed several challenges insofar that they generate, process, and exchange vast amounts of security critical and privacy sensitive data, which makes them attractive targets of attacks and unauthorized access. Security requirements in such scenario include integrity, confidentiality, traceability and notarization of exchanged data in the one hand plus access control, privacy and trust in the other one. In this context, we design a distributed resource management framework using the emerging smart contracts technology for Industry 4.0 applications and more specifically for smart factories environments. This last, named DRMF, utilizes three Ethereum smart contracts specifically a Governance Contract (GC), an Access Contract (AC) and a Lookup Contract (LC) that are respectively responsible for the registration of new joining entities as well as those requesting consensus partaking permissions, second the dynamic access authorization and third the mapping between the required services and contracts ensuring their management. Using the blockchain technology, this framework is expected to achieve distributed, flexible, verifiable and trustworthy access control in addition to a transparent, traceable and notarized resource usage and sharing. Results from a real implementation plus performance evaluation prove the proposed concept and demonstrate its feasibility.**

***Index Terms*— Blockchain, smart contracts, notarization, access management, governance, Smart factories**

I. INTRODUCTION

With the introduction of the Internet of Things (IoT) and Cyber Physical System (CPS) concepts in addition to the current big data and data analytics environments within industrial application scenarios, industrial automation is undergoing a tremendous change introducing as a consequence thereof the fourth industrial revolution sometimes referred to as Industry 4.0 [1], [2], [3]. This concept has introduced what has been called the Smart Factory, the revolutionized version of traditional factory in which CPSs communicate over the IoT to monitor the physical processes of the factory and to make decentralized decisions.

This revolution includes the introduction of highly flexible and greatly efficient supply chains, manufacturing on demand, logistics operations and production processes, the provision of new services and the allowance of mass-customization and virtual production.

However, the deployment of such technologies in addition to the heterogeneous and constrained nature of IoT devices, is expected to intensify encountered security threats and issues. In such environments, security is one of the most important concerns, given that vulnerabilities introduced during manufacturing can be hard to detect and harder to react especially with open connectivity to the external world in addition to the increasing amount of data shared between devices deployed in heterogeneous, distributed and unknown sites while collaborating altogether along the production process. For example, attackers can manipulate and infiltrate industrial systems, malware injection can disturb their functioning and put them out of action, which could cause significant damage to the whole production area. Also, users with different motivations, roles and in different contexts can utilize the terminals to interact with data storage systems. They may query data or attempt to take control of some physical resources (i.e. terminals and the communication channel are the target). Or an attacker can seek to collect available data through harvesting and exfiltration of sensitive information from these terminals. Consequently, it is not feasible to design a solution that can comprehensively address and mitigate in once all encountered security threats.

In this work, our study mainly focuses on the following security properties:

- Integrity, confidentiality and notarization: Entities and resources related information are generally recorded in a local storage structure. Hence, there is a risk that these structures can be subject to unauthorized access and modification. Consequently, traceability and auditability of both the flow of data and their access history records will be challenging.
- Dynamic access management: Dynamic reconfiguration of access rules in response to time, events and more importantly to entities changing behavior and attitudes.
- Distributed system governance: Distributed management of the shared resources plus the authorization of new entities willing to partake in the system without the need for a third party that could be itself vulnerable to attacks.

Recently, blockchain technology has attracted a lot of interest in both the research and the industrial communities. The decentralized, fault tolerant computing, storage and sharing of blockchain technology [4] can lead to a whole wave of security innovations. This last consists of [4], [5] blocks chained together as a ledger. Any single transaction is verified and validated by entities in the peer to peer network, then packaged into block and chained within the blockchain network according to an established consensus mechanism. The security of this technology returns to the fact that it uses hash functions to chain blocks in order to ensure immutability in addition to the use of digital signatures to secure information. Availability is also ensured because of blockchain's distributed characteristics. Besides in smart factories environments, the transmission, the distribution and the storage of information require high confidentiality, integrity, validity, and authenticity. In this context and regarding the different benefits provided by the

978-1-7281-2522-0/19/$31.00 ©2019 IEEE

blockchain technology, we propose in this paper to use it within a resource management framework in order to take advantage of security features it provides regarding reliability, traceability, control, information integrity and notarization. The proposed framework utilizes blockchain to keep a living document trace about the flow of resources being distributed and shared among collaborating parties while using the OrBAC access control model to implement distributed, fine grained, flexible and secure resource access authorization. Moreover, and in order to better support the security requirement, this framework adds the notion of trust management to the access control model. Here a trust framework is integrated to evaluate access requester entities' behavior guaranteeing thereof dynamicity of security policies insofar that they would be defined and validated function of the access requester entity's behavior. Finally our proposal is conceived to support distributed and dynamic governance of the system. Herein, all relevant parts can make a comprehensive decision of access requests, policies definition and even consensus management through the registration of new entities requesting mining permissions. To do so, it is essential to confirm before that they do not pose a threat to the system, otherwise, they would be eliminated and rejected.

To demonstrate the application of the framework, we provide a case study, in which we set an Ethereum private blockchain network and we implement three smart contracts respectively responsible for: the access control, the governance and the entities registration.

The rest of this paper is organized as follows. Section 2 presents the proposed case study. Section 3 recalls the basic concepts of blockchain, Ethereum plus access control systems and presents related proposals carried out in the area of blockchain based access control. Thereafter, an overview of the proposed scheme as well as its detailed design is given in Section 4. Section 5 delves into the implementation of the proposed scheme, a set of experimental results validating our approach are shown, and finally in Section 6, the paper ends up with some conclusions and an outlook of our future work in this area.

II. Motivation and use case study

Throughout this paper, we will project and illustrate our approach with a scenario example inspired from a real world study of existing applications related to Industry 4.0.

As a case study, let us take the example of three automaker factories SF1, SF2 and SF3 respectively responsible for: (i) the manufacturing of mechanical and electrical components, (ii) the assembly operations and (iii) the test plus the performance and quality control. These factories over time interact all together along the production processes and alongside with automotive suppliers, as an example a raw material supplier SP1 and a component supplier SP2 proposing, buying and shipping goods and products through transportation partners. These parties while looking to ensure sophisticated shipping and logistics operations, agreed to invest in shipping and logistic equipments as well as technologies and personnel to manage their day-to-day trucking operations instead of relying on third parties to ensure them. Obviously, the fact of using their own resources for shipping operations will increase the efficiency of processes, the immediate availability of vehicles, the reduction of cost and the increase of profits insofar that they could rent these resources to other companies and help them load, deliver and unload their items and products. Another important advantage within such decision is the fact of preserving their privacy as well as those related to their resources especially when the third part is not trustful enough so that they could rely on it to ensure such service which could lead to several issues as they put a non trustful third part in control of one of the business functions with the most impact on the smooth running of production processes and the greatest effect on their customers satisfaction.

That's why having their own shipping and logistic resources is really important to overcome such limitations. We notice here that to share such resources among several parties collaborating and working all together and especially that could not always have a strong confidence established in advance, different challenges should be firstly resolved as follows:

- Fully distributed management framework: where we don't need to pass through a third part or to involve several services to manage shared resources and to perform common processes.
- High security level: that guarantees integrity, confidentiality, traceability and auditability of both established transactions and access records and procedures.
- Dynamic access control: insofar that security rules can change dynamically in response to time, events and more importantly to involved parties changing behavior and attitudes.
- Distribute governance: where collaborating parties could join, leave the system and partake in the consensus mechanism any moment they want without worrying neither about the well conduct nor about the security of common processes and shared resources obviously after a consortium is established between the collaborating parties.

III. Background and problem statement

In this section, we will introduce first the main preliminaries used in our proposal, we will review then related works and discuss the main features and benefits of distributed peer to peer networks.

A. *Blockchain technology*

Originally designed for keeping a financial ledger and meeting the purpose of cryptocurrency applications, the blockchain paradigm can be extended to provide a generalized framework for managing any movements of data related to goods, devices, information records, etc. This last could be defined as a distributed ledger of transactions whereby records of all established interactions are registered providing thereof a proof of existence, of ownership and modification of this data during interaction [4], [6]. The established transactions are held within blocks chained together and containing within their headers the hash of the previous block in order to ensure immutability since blocks once chained, data contained within will be available and couldn't be easily changed or altered. To ensure that all entities have the same copy of the ledger, a consensus is required to maintain the blockchain architecture and to ensure its functioning. This last makes sure that an agreement is reached between a set of predefined entities to support a decision making. After reaching consensus, valid blocks are added to the blockchain. Moreover, each node in this distributed peer-to-peer network holds the same copy of transaction records, which provides robustness against single point of failure attacks.

B. *Ethereum and smart contracts*

Ethereum is a global, open source blockchain based distributed computing platform for decentralized applications [7]. The novelty of such protocol is considering the blockchain technology not only to track the flow of value's exchange but also to carry out executable code through transactions created and sent by Externally Owned Accounts (EOA). [8] Those pieces of code deployed and residing at a specific address on the Ethereum blockchain, are called smart contracts (SC). These last include a set of data which are the state variables and code corresponding to the executable functions. SC functions are executed when transactions are made and broadcast to the network. These transactions include input parameters, as a result an eventual return value is shown to the outside.

SC are written in a low level bytecode language interpreted by the Ethereum Virtual Machine (EVM) which corresponds to the run time environment in Ethereum. High level languages whose programs can be compiled in EVM bytecode have also been developed. The most widespread language is Solidity [9] which is a JavaScript style contract-oriented, statically-typed, high-level programming language designed for implementing smart contracts.

C. Access control systems and blockchain

In the current literature several models were proposed to define access rights and to control access requests [10], [11], among them: Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role Based Access Control (RBAC), Attribute-based Access Control (ABAC), and Organization Based Access Control (OrBAC). Many derivatives have been deduced as well from these models in order to resolve a specific need.

In the context of smart factories and considering the need for dynamic security rules definition and parameterization, context awareness, security rules abstraction, scalability of resources, actions, subjects and situations, plus expressivity and fine-granularity, we can consider OrBAC as a good candidate for providing an adequate access control model for such environments.

Reminding that within a multi factories based environment, entities belonging to different factories interact with each other in order to realize a common goal where the concept of multi organizational environment characterized by large scale and independent structures with decentralized systems, where each factory defines its own model, assigns its own roles and specifies its own access control policies. The question to be asked here how to verify the role attribution procedure and how can we provide flexibility to entities to fully control their roles as well as access requests related to their resources? Another issue is the integrity and the confidentiality where entities access requests and operations history are stored within a local database or a cloud infrastructure. Hence, there is a risk that these databases can be subject to unauthorized access and modification. Consequently, traceability, notarization and auditability of access records will be challenging. A last but not least issue is the difficulty of managing security policies according to the context dynamicity especially with the colossal number of entities supposed to be managed and that could change their behavior over time.

That's why providing an adequate solution to distributively govern the system and to dynamically, securely and contextually manage the access while keeping a living document trace about the flow of resources data being shared become crucial and no more important than ever.

In this direction and given the noted features of blockchain technology, this last applied to such systems provides promising possibilities and solutions to issues they encounter as it was previously discussed. Few proposals of blockchain related access control systems have been presented in the current literature.

This technology was used as a storage structure for access control policies in [12]. Therefore, its computing feature was examined in [13] where it plays the role of a decentralized access control manager. In this work, FairAccess was proposed to offer a fully decentralized pseudonymous and privacy-preserving authorization management framework for IoT devices. The proposed framework used OrBAC access control model to enable users to own and control their data whose policies were stored in a private blockchain. However, it can handle only policy-based compatible systems and use cases, which cannot be applied to smart factories contexts.

Subsequently, the idea of using smart contracts for achieving access control has been adopted in [14], [15], [17] for different access control systems.

In [14], authors proposed RBAC-SC, a Role Based Access Control system using Smart Contracts. In this model, Ethereum's Smart Contract technology was used to realize a trans-organizational utilization of an organizations roles.

In [17], a blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0 was presented. The proposed framework leverages the underpinning characteristics of blockchain as well as several cryptographic materials to realize a decentralized, privacy preserving solution. However an implementation of the proposal is missing as well as the evaluation and the proof of its performance.

In [16], a Transaction based access control platform was proposed where the ABAC model was integrated within the blockchain system to manage subject registration, object escrowing and publication plus access request and grant.

D. Discussion

As seen, the blockchain technology has been used for several purposes and within various models of access control. However, just few studies have focused on integrating the blockchain technology within access control systems in the context of smart factories environments [17]. Our focus in this work is not only to ensure fine grained, flexible and secure resource access authorization in the context of smart factories environments but also to support distributed and dynamic governance and management of the overall system where all relevant parts can make a comprehensive decision of joining entities registration and consensus management for entities requesting mining permissions. Another focus of this framework is the integration of trust management with the access control model in order to determine whether subjects are trusted and well behaved enough so that they could access resource data.

IV. Proposed Approach

A. Overview

In this paper we present DRMF, a Distributed Resource Management Framework based on the blockchain technology for Industry 4.0 deployments. The main objectives of this work are as follow:

- Information notarization: The use of blockchain to keep a living document trace about the flow of data and resources being shared by collaborating entities guarantees an extra level of transparency, control and notarization during collaboration where a proof of existence, of ownership, of access and modification is essential for decision making process.
- Distributed system governance: For each entity willing either to partake in the consensus mechanism or to join the blockchain network and sharing common resources, a verification of its behavior by most participants in the system is made in order to ensure its legitimacy absolutely necessary to confirm its joining request. The registration of new entities as well as the management and the elimination of joined ones require an agreement to be reached between the pool of entities taking in charge the consensus mechanism.
- Dynamic access management: this framework achieves distributed, dynamic, contextual and trustworthy access authorization through the integration of the OrBAC access control

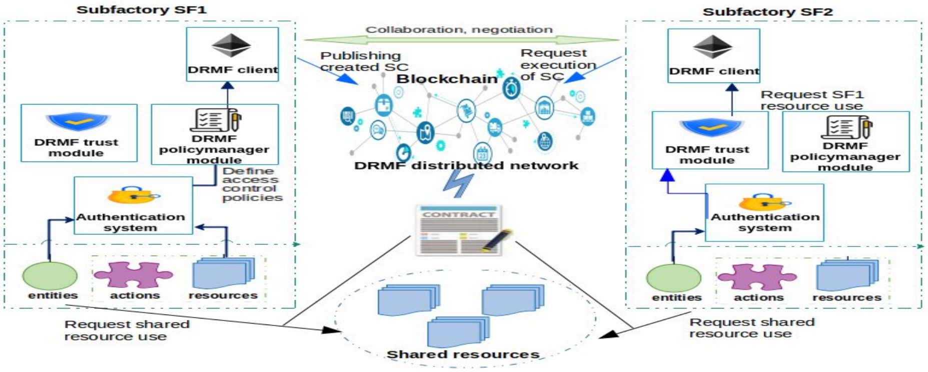


Fig. 1. System architecture

model within a distributed ledger where transactions serve as verifiable and traceable medium of policies definition and parametrization, access request procedures as well as related operations.

Fig. 1 shows the overall structure of the proposed system.
As illustrated in Fig. 1, each domain (e.g. SF1, SF2) holds its own entities with different roles, for example a human worker in the manufacturing subfactory SF1, a supervisor agent in the quality control subfactory SF3, these last could perform several actions on each shared resource such as using the shipping trucks, sharing their trust records, querying for available ones, etc. Let us assume that a human worker within SF1 wants to have access over the shared resource Truck 3 to perform a shipping operation. To do so, the process would work as follows: the agent needs to get authenticated first along with an authentication system, which is responsible for making authorization decisions, verifying devices identities and generating authorization tokens. Furthermore and instead of statically evaluating the access request, a DRMF trust-module is involved to assess the trustworthiness degree of the requesting entity and to judge its behavior taking into account different trust aspects and parameters (that will be discussed in Section. IV-B.3). The result of this evaluation will be incorporated within the context structure to be sent to the DRMF policymanager-module which is in charge of formulating the corresponding transaction to be broadcast via the DRMF client to the DRMF distributed network where the corresponding smart contract will be executed. Reminding that within a smart factory environment, IoT devices are used to collect and analyze data coming from smart products, other smart devices and related smart services, that's why in the case of IoT devices with tight resource constraints, DRMF modules are assumed to be deployed in more powerful network components that will be connected directly to each device, otherwise it is deployed within the device/entity itself.

B. System composition

In the following we will detail the different software modules composing our system, the specific role and the operation of each smart contract as well as the workflow of the overall architecture. As shown in Fig. 1, our framework consists of the following components:

1) DRMF distributed network: This component is the main core of our system. It defines: (1) a set of smart contacts ensuring the notarization, the distributed governance and the dynamic access management, (2) a distributed network composed by a set of peers responsible mainly for receiving the established transactions, verifying and validating its state and executing the functions contained within.

In this work, we utilize Geth technology to create a secure and a distributed structure for managing data related to shared resources which are the shipping trucks according to our use case. Besides, the proposed framework consists of three Ethereum smart contracts, namely: Access Contract (AC) designed to manage access authorization made over shared resources, Governance Contract (GC) used to ensure a distributed governance of the overall system, and Lookup Contract (LC) acting as a register to map between the required services and the contracts ensuring their management. More details about each smart contract will be provided in the following subsections.

2) DRMF client: This component implements the full functionality required to join and to participate in the DRMF distributed network. This handles a broad set of tasks, such as connecting to the peer-to-peer network, encoding and sending transactions, keeping and exploring blocks copies and deploying and interacting with smart contracts.

3) DRMF-Trust module: this component is in charge of assessing the trustworthiness degree of shared resources as well as requesting entities in order to capture their ability and willingness to behave as expected. The realized assessment will be linked with access control decisions through the context structure defined within security rules that will no longer be statically defined and parametirized, but instead will depend on entities behavior in addition to the environment dynamicity and context change. We remind here that our proposal is based on the OrBAC access control model where the integration of trust management to enhance the security level of the corresponding system has been studied by the research area during the last years. There are multiple ways to define trust and to evaluate it. This last could be based on a set of paramaters where the most relevant ones defined in the literature are experience, reputation and knowledge [18], the experience parameter corresponds to the interpretation of the previous interactions established with immediate neighbors. These evaluations will be propagated as trust recommendations to other network nodes to constitute the reputation parameter, that once kept, will be considered after as the knowledge part of trust. How to evaluate trust is not the topic of this paper, however we will provide an example of metrics that matching with the proposed use case, could be considered as effective ones within the trust evaluation process.

We remind here that according to our use case, the evaluation of trust will focus on both access requesting entities and requested resources (which are the shipping trucks) as target nodes addressed by the trust model. The trustworthiness assessment of the shipping resources could be based on their speed, position, direction of motion, engine coolant temperature, battery voltage, real-time diagnostics and performance evaluation results of vehicle status, the trustworthiness degree of their drivers, etc, [19].

4) DRMF-PolicyManager module: This component serves in the one side as the defining part of access rules to be encapsulated into transactions and reloaded to the blockchain after validation, on the other side this module serves as the acquisition source of attribute values required for policy evaluation that once received and intercepted will be sent to the corresponding smart contract.

5) Authentication system: this component is mainly responsible for verifying the validity of entities' identities as well as the legitimacy of demands and requests sent to both DRMF-Trust and DRMF-PolicyManager modules. Participating entities are authenticated based on the provided credentials. In our framework, we rely on the openID Connect [20] (OIDC) which is an identity layer on top of the OAuth 2.0 protocol [21]. We have chosen OIDC since it is free, open and decentralized (no central authority approves or registers relying parties or service providers). Its integration does not require complicated update in the deployed application. Indeed, it follows a restful approach which makes it easy to use and to interoperate.

Remembering here that at the beginning of the authentication process, participating entities are authenticated based on the provided credentials. These last can be represented by using different mechanisms, such as login/password, digital certificates, authentication keys, etc. In case of a successful authentication process, this module generates an access token which is delivered in order to avoid subsequent authentication procedures.

6) Smart contracts: : In our work, we proposed three smart contracts: LC, AC and GC.

Lookup contract (LC): The main role of this smart contract is to map between the required services and contracts ensuring their management. To do so, it maintains a lookup structure that registers the required information to find and execute the methods in question. This structure contains the name of the smart contract in which the method is developed, its address, the address of its creator and the name of the requested method. The operation of this smart contract will be mainly based on the following methods:

- lookup(): This method receives in input the name of the requested method to return the address of the corresponding contract (i.e., the access contract AC, the governance contract GC).
- addFunction(): This method receives in input the information details of a new function to add to the lookup structure, obviously only the creator of the corresponding smart contract can add new methods.
- deleteFunction(): This method receives the name of an existing function to delete from the lookup structure.

Access contract (AC): This smart contract is mainly designed to achieve distributed, interoperable, contextual, trustworthy and secure access control for multi organization systems where participating parties interacting and collaborating all together share common resources for which access rules should be maintained and parameterized by the collaborating parties. This smart contract is based on the integration of the OrBAC access control model within a distributed ledger to express access control policies. We opt here for such integration firstly to guarantee the fact that access policies are available at any time and evaluated in distributed environments where there is no central authority to define roles and to generate security rules what would overcome the problem of having a single point of failure. Secondly to ensure verifiable and transparent role assignments where any entity can verify if another one own really the role it pretends to have and that this last is managed and issued by its factory or belonging organization. Third to enable the dynamic reconfiguration of access rules in response to time, events and more importantly to entities changing behavior and attitudes.

View	**Activity**	**Role**	**Context**	**Access type**
Truck related file	update	Supervisor	Sup.Trust-score $>$ T-Th1 AND Current-time IS IN Working-hours	Permission
Genesis-block-addr	mine	BC-node	BC-node.Trust-score $>$ T-Th2 AND Node-registered	Permission
Truck 3	use	human worker	Worker.Trust-score $>$ T-Th3 AND Current-time IS IN Working-hours AND Department IS IN Shipping	Permission

TABLE I

SECURITY RULES LIST EXAMPLE

To ensure such features, this smart contract maintains a set of security rules. Table. I illustrates a simple example of rules where each row corresponds to the policy defined on a certain tuple. Basic fields of each row are:

- View: it represents the resource for which the access is requested. According to our use case scenario, this last could be a record related to production data, manufacturing entities, trust records, etc.
- Activity: it represents the action to be performed on the resource such as check, update, use, etc.
- Role: that represents the entity requesting the access for a certain resource.
- Context: it is used to express different types of extra conditions or constraints that control activation of rules expressed in the access control policy.
- Access type: it defines the access type defined on the view, according to the OrBAC model, this last could be permission, prohibition, obligation and recommendation.

Returning to the use case presented in Section. IV-A, to perform the shipping operation using the Truck 3, an evaluation of the agent's access request is needed first to allow or not the demanded access. To do so a verification of the well conduct and functioning of both the requesting entity and the requested resource is essential, hence an authorization over the access control smart contract is required to decide whether the access is permitted or prohibited. The access authorization and according to the OrBAC access control model depends on a set of contextual conditions whose activation will activate the corresponding rule. In this example, to have the requested access accepted, the human agent should have a trust score above the defined threshold for the corresponding role, he should belong to the shipping department and request to use the Truck during his working hours. We can notice here that the fact of using trust management within the access control would enable the dynamic reconfiguration of security rules that will change in response to involved parties' changing behavior and attitudes. To do so, we have added a novel type of context which is related to trust management. The role of this latest is to check if the trust levels of both the access requesting and the requested entity respect well the threshold defined.

The operation of the Access contract will be based on the following methods:

- addpolicy(factory id, role id, string activity, view id, struct context, string permName): this function launched by resource owners aiming to define and to add a new access control policy for a newly shared/stored resource. It takes as input the owner's belonging factory public key, the role id of the subject, the view id of the object, the context in which the access is demanded, the permission to be attributed. As a result of this function, a new policy item will be added to the policies list.
- Updatepolicy(factory id, role id, string activity, view id, struct context, string newPerm): this function launched by resource owners in order to update an already added access control policy.
- deletepolicy(factory id, role id, string activity, view id, struct context): this function receives the main identification information of a policy to be deleted from the policy list.
- accesscontrol(factory id, entity id, string activity, view id, struct context): this function executed by a subject requesting entity in order to authorize its access request upon a certain resource identified with its view id within a certain context. To do so a verification of the request is made to check the validity of the subject role, the existence of the policy within the defined ones and the behavior of the requesting entity to detect a potential doubtful/suspect access demand. As a result an access result will be returned to both the requesting and the requested entities and the access process will be executed.

Governance contract (GC): This contract is mainly designed to govern consensus and to manage who can partake in the consensus mechanism within the blockchain network. More specifically it is responsible for determining the consensus algorithm to be executed for the mining procedures, as well as for registering and managing miner entities. To do so, this contract is supposed to store blockchain addresses related to entities having either transaction validation, mining or voting permissions. For registering miner entities, the consensus contract is used to validate entities requesting mining permissions in order to be allowed either to validate, create and add new blocks to the ledger. Here we note that once the system is deployed, this contract would contain initial validators representing the collaborating parties. For overwriting miner entities, a request to delete the entity in question is submitted by the one who has noticed its malicious behavior or its breakdown. Thus once the rest of entities have reached a majority, the miner will be removed from the consensus contract.

We remind here that in order to be registered, or to send overwriting instruction, the requesting entity needs first to get authenticated and authorized over the access control smart contract, then the request will be transmitted to the pool of entities taking in charge the consensus mechanism, once confirmed that it does not pose a threat to the system, its address will be added to the governance contract.

C. Prototype workflow

The proposed framework involves a succession of operations wherein: (i) DRMF distributed network is created and a consortium is defined in order to reach the desired agreement among collaborating parties that once identified get registered and involved in the consensus procedure. (ii) given the existence of resources to be shared, these last are identified in order to be accessed and used in a notarized manner. (iii) to do so, first security rules should be defined in order to manage the access among the collaborating parties, (iv) entities requesting to join the system or to partake in the consensus mechanism, are registered obviously after evaluation of their behavior, corresponding roles are identified and attributed as well, (v) when willing to perform an action over an existing resource, the requesting entity sends an access request that further to which a decision is made.

These five scenarios our proposed framework is made up of are detailed in the next paragraphs.

1) DRMF distributed network creation and entities registration: Once the DRMF distributed network is created and becomes operational, the process of adding new entities with different roles and classifications can be launched. Here, it should be assumed that new entites have been already identified within the authentication system and possess a public identifier that is unique to their organization. It should also be assumed that they have received an Ethereum address required to partake in the Ethereum network. Therefore, the process of adding a new entity begins by having the pool of consensus entities validate that the public identifier corresponds to the pretended role and suits the requested classification.

2) Security rules definition: When collaborating sub factories agree on adding an access control policy for a newly deployed resource to be shared among them during a certain time, e.g., collaborating sub factories SF1, SF2, and SF3 agree on adding a new truck identified with a new attributed address R_addr1. To manage the access over the shared resource, an access policy needs to be added to the AC smart contract and shared within the DRMF distributed network. The proposed framework works as illustrated in the UML sequence diagram in Fig. 2. The resource holder needs to get authenticated first along with the authenticator, in case it is already authenticated, the attributed token is used to have access to the system. Therefore, the DRMF-Trust module evaluates the resource trust value, defines the trust threshold to be satisfied by the access requesting entity and generates the context structure to be sent to the DRMF-PolicyManager module. This last defines the possible roles, the activity to be performed and encapsulates the defined access control policy in form of a scripting language in order to be added to the policies list within the AC and sends it to the DRMF client in order to be broadcast to the DRMF network. The peer to peer nodes verify the transaction, and record it within the distributed ledger in case of success validation. At this stage, a new AC is created and deployed on the blockchain, obviously a new entry is added to the LC in order to register the required information of the newly created AC via the addFunction() method.

3) Access request transaction workflow: In case of access request, the proposed framework works as follows: A subject (e.g., a human worker within the supervision service of the performance and quality control subfactory SF3, identified with his ethereum address E_Addr) wants to perform an action (e.g., execute) on a protected resource (e.g., a controlled robotic arm responsible for auto body panels spot welding operations within the assembly subfactory SF2, identified as well with its ethereum address R_addr). The subject E_addr after being authenticated within the authentication system for a first authentication case or after validating the access token it uses, will submit its access request to perform the execute action on the resource R_addr. The DRMF-Trust module at this stage receives the access request, assesses

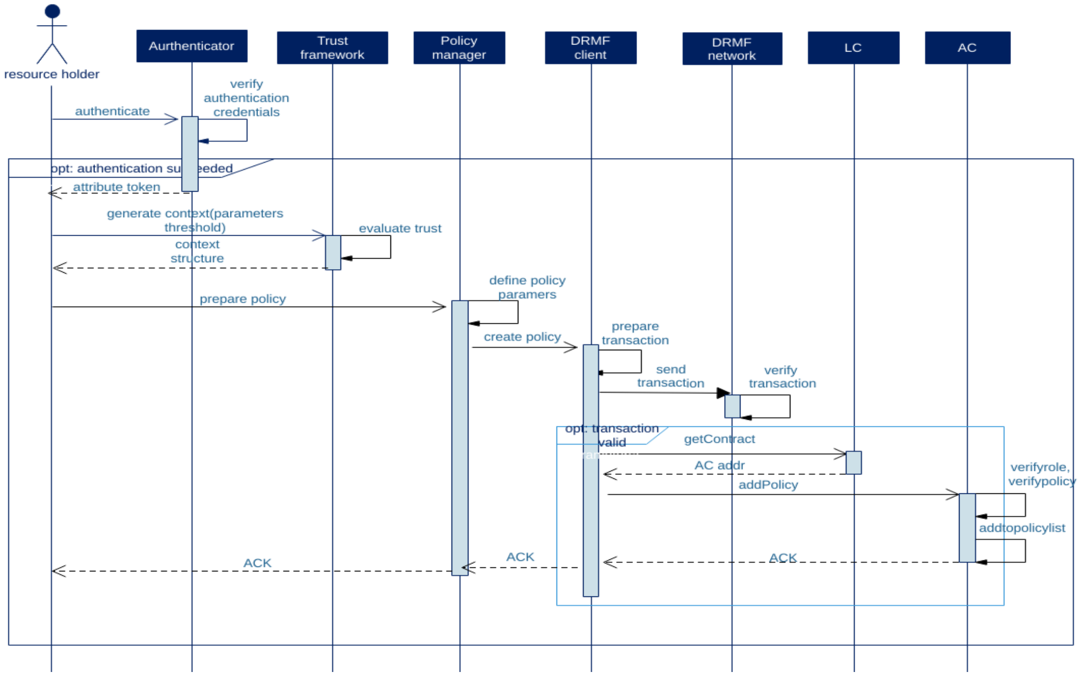


Fig. 2. Policy creation transaction workflow

the entity trustworthiness, derives its trust value, prepares context related attributes and generates the context structure to be sent to the DRMF-Policymanager module acting as a Policy Enforcement Point (PEP). This last formulates the access request to an access transaction and broadcasts it to the peer to peer network via the DRMF client in order to run the AC. We remind here that before sending the access demand transaction, the DRMF client needs to have the address of the AC, to do so it calls the getContract method of the LC to retrieve the AC address and concerned method. Once received, the access demand transaction is sent to the AC acting as a Policy Decision Point (PDP) that evaluates the access demand by verifying the validity of the subject role, the existence of the access policy within the defined policies list and especially checking the subject's behavior, as a result it determines whether the request should be permitted or denied. Finally, if it is permitted the transaction is valid and it will be recorded in the blockchain else the transaction will be rejected and a notification will be sent to the requester.
The described workflow is illustrated in Fig. 3.

V. Implementation and Evaluation

In this section, we will introduce first the different tools used for the implementation of our framework therefore we will show experiment results demonstrating its feasibility.

A. *Work environment*

For the implementation of our proposal, we have set up a testbed as illustrated in Fig.4 featuring several hardware and software components as listed below:

- a Dell Precision M6800 machine with 4th Generation Intel Core i7 processor and 8Gb of RAM in which we have configured a private Ethereum blockchain network consisting of three nodes having the functionalities of Ethereum miners.
- an Intel Core i5-3210M laptop with 6 Gb of RAM and 2.40 GHz of CPU frequency in which an Ethereum node was set up.

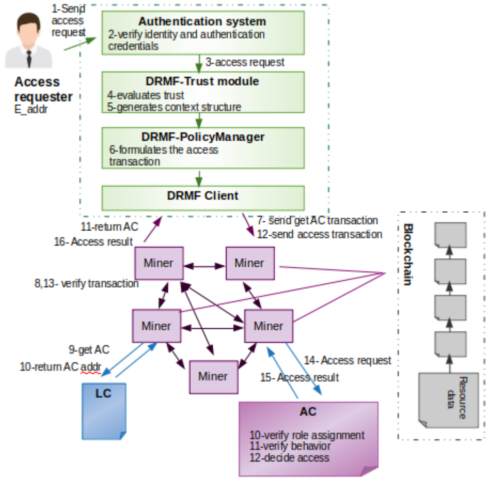


Fig. 3. Access request transaction workflow

- two Raspberry Pi 3 Model B configured to act as shipping resources shared among the collaborating factories.

According to the use case study we introduced in Sec. II, our scenario consists of 3 subfactories SF1, SF2 and SF3, where the administration service is represented by each Ethereum node within the Dell machine. These factories collaborate altogether alongside the production process and sharing as common resources two trucks responsible for the trucking and shipping operations. These last are represented by the two single boards (RPi3) that are connected to the blockchain network where related information according to their real-time diagnostics and performance evaluation in addition to their trustworthiness degree are collected, sent and stored within the distributed ledger. Moreover a human worker within the shipping service willing to perform a shipping operation using a truck

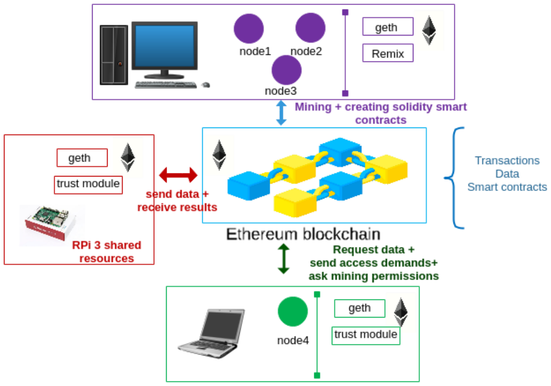
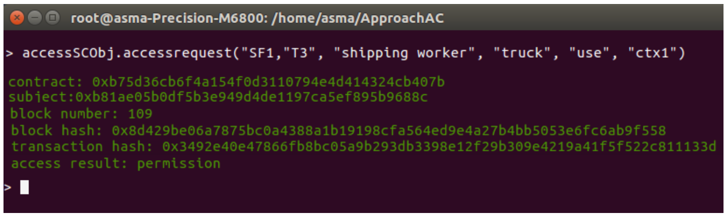


Fig. 4. Work environment

resource is represented by the laptop machine.
As illustrated in Fig.4, a geth client [22] is configured on each entity so that it could act as an Ethereum node. For each entity we have created an account and set it to form the DRMF network. Mining tasks are ensured by the Dell machine insofar that it has a relatively large computing software and storage capability where the Proof of Work (PoW) consensus mechanism is supported by each node. This entity took in charge also the creation and the deployment of solidity smart contracts. In what concerns the trust module integrated within either the laptop machine or the RPi share resources, we have used the model presented and implemented in our previous work [23].

B. Proof of concept

As it was presented in Sec.IV-B.6, each smart contract is based on a set of methods developed under specific algorithms according to the defined use case scenario. Reminding that our DRMF framework is based on the following main functionalities: (1) Registering a new resource with a corresponding address. (2) Definition of security rules. (3) Access request. (4) Behavior evaluation. (5) System governance.
In order to show the feasibility of our proposal, we conducted some experiments related to the access control and the consensus governance procedures. For the validation of the access contract, we added a new role 'shipping worker', a new context with the trust threshold, the affiliation and the working hours, a corresponding policy then is added to the policies list that according to the OrBAC model, is specified as follow:
permission(org: SF1, shipping worker, use, truck3, shipping worker.trust-score > trust-th3
AND current-time IS IN working-hours
AND shipping worker.department IS IN shipping)
Therefore, we defined a malicious behavior where an on-off attack is launched to cause a low trust score that was calculated by the DRMF-trust module. As a penalty for the malicious behavior, access requests from the subject will be blocked for a certain period of time (until the trust score is above the trust threshold fixed to 0.4). Fig.5 and Fig.6 show access results displayed after a legitimate behavior and a malicious one respectively.
For the validation of the consensus contract, we need here to prove the well achievement of consensus while dynamically adding and removing nodes to and from the network. We will consider the case of a new node willing to partake in the consensus mechanism within the blockchain network. This last needs to get authenticated and authorized first over the access control smart contract, then the request will be transmitted to the pool of entities taking charge of the consensus mechanism, once confirmed that it does not pose a threat to the system, its address will be added to the consensus contract.

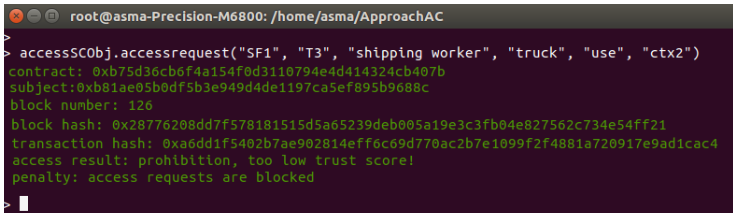

```
root@asma-Precision-M6800: /home/asma/ApproachAC

> accessSCObj.accessrequest("SF1,"T3", "shipping worker", "truck", "use", "ctx1")

contract: 0xb75d36cb6f4a154f0d3110794e4d414324cb407b
subject:0xb81ae05b0df5b3e949d4de1197ca5ef895b9688c
 block number: 109
 block hash: 0x8d429be06a7875bc0a4388a1b19198cfa564ed9e4a27b4bb5053e6fc6ab9f558
 transaction hash: 0x3492e40e47866fb8bc05a9b293db3398e12f29b309e4219a41f5f522c811133d
 access result: permission

> 
```

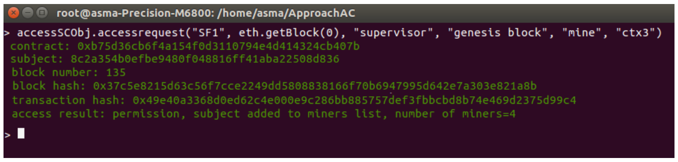
Fig. 5. permission of access


```
root@asma-Precision-M6800: /home/asma/ApproachAC
>
> accessSCObj.accessrequest("SF1", "T3", "shipping worker", "truck", "use", "ctx2")
contract: 0xb75d36cb6f4a154f0d3110794e4d414324cb407b
subject:0xb81ae05b0df5b3e949d4de1197ca5ef895b9688c
 block number: 126
 block hash: 0x28776208dd7f578181515d5a65239deb005a19e3c3fb04e827562c734e54ff21
 transaction hash: 0xa6dd1f5402b7ae902814eff6c69d770ac2b7e1099f2f4881a720917e9ad1cac4
 access result: prohibition, too low trust score!
 penalty: access requests are blocked

> 
```

Fig. 6. prohibition of access

We added a new role 'supervisor', a new context with the trust threshold, the affiliation and the working hours. We specify thereafter the policy to be added as follow:
Obligation(org: SF1, supervisor, mine, genesis-block-addr, registered AND supervisor.trust-score > trust-th2
AND current-time IS IN working-hours
AND supervisor.department IS IN administration)
Fig.7 shows corresponding access results.


```
root@asma-Precision-M6800: /home/asma/ApproachAC
> accessSCObj.accessrequest("SF1", eth.getBlock(0), "supervisor", "genesis block", "mine", "ctx3")
 contract: 0xb75d36cb6f4a154f0d3110794e4d414324cb407b
 subject: 8c2a354b0efbe9480f048816ff41aba22508d836
  block number: 135
  block hash: 0x37c5e8215d63c56f7cce2249dd5808838166f70b6947995d642e7a303e821a8b
  transaction hash: 0x49e40a3368d0ed62c4e000e9c286bb885757def3fbbcbd8b74e469d2375d99c4
  access result: permission, subject added to miners list, number of miners=4

> 
```

Fig. 7. registration of a new entity within the consensus mechanism

C. Performance evaluation

To evaluate the performance of our proposal, we have evaluated the gas it consumes to perform required task. Gas is the unit used for measuring the computational work of running transactions or smart contracts in the Ethereum network. This unit reflects the complexity of executed tasks as the more complex they become, the higher amount of gas they require to complete. Table. II enumerates the amount of gas required for the completion of specific transactions specifically the creation of a new access rule, the access request, the registration of a new entity within the consensus mechanism, the deployment of GC, LC and AC.

D. Discussion

In this paragraph, we analyze the different features provided by DRMF that utilizes blockchain to keep a living document trace about the flow of resources being distributed and shared among collaborating parties while implementing distributed, dynamic and secure resource access authorization.

Transparency

The DRMF framewwork achieves the transparency property since

Executed transaction	Required gas
The creation of a new access rule	475200
The access request	420500
The registration of a new entity	350000
The deployment of GC	1513300
The deployment of LC	1320000
The deployment of AC	1681500

TABLE II
Security rules list example

all functions executed within smart contracts are reflected on the corresponding log of both the respective smart contract and the DRMF distributed network. By this way, an entity could not perform any transaction without other entities' knowing and validation, and also it cannot deny any transaction it has committed.

Verification
The DRMF framewwork effectively achieves verifiable access control through the verification of role assignments, access rules existence and non violation in addition to the legitimacy of access requests implemented within the AC.

Flexibility
Our proposal gives participating entities the flexibility to join or to leave the system or even to partake in the consensus mechanism easily. Functions implemented within the GC ensure the registration of new joining entities, the overwriting of existing ones and the management of those taking in charge the consensus mechanism.

Dynamicity
Dynamic reconfiguration of access rules in response to entities' changing behavior and attitudes is ensured within the AC that does not only verify and validate access authorization statically by checking access rules' defined conditions are met, but also dynamically by checking the behavior of the access requesting entity and judging its legitimacy or maliciousness according to the trust score derived by the DRMF-Trust module.

VI. Conclusion and future work

We have presented in this paper the design and the implementation of a distributed resource management framework based on the Blockchain technology in order to (1) notarize the flow of data and resources being shared by collaborating parties, (2) achieve a secure, trustworthy, fine-grained, and traceable access control and (3) dynamically and distributively manage new joining entities as well as those willing to partake in the consensus mechanism. Combining the resource management system with the blockchain technology, we enable a more reliable resource data confidentiality and integrity verification during sharing and we make a time-stamped log of both entities' access transactions and behavior. Our evaluation shows that our proposal is feasible, deployable and suited for smart factories environments given its features of being decentralized, ensuring security and requiring a low overhead in addition to low resources. For future work, we plan to implement other benchmarking blockchain consensus mechanisms in order to more ameliorate the overhead and the performance of our proposal.

References

[1] R. C. Schlaepfer and M. Koch. Industry 4.0 Challenges and solutions for the digital transformation and use of exponential technologies. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/ch/Documents /manufacturing/ch-en-manufacturing-industry-4-0-24102014.pdf

[2] K. Schwab. The Fourth Industrial Revolution. World Economic Forum, 2016.

[3] H. Kagermann and W. Wahlster, Industrie 4.0 - Smart Manufacturing for the Future. German Trade and Investment, July 2014. [Online]. Available: https://www.gtai.de/GTAI/Content/EN/Invest/_SharedDocs /Downloads/GTAI/Brochures/Industries/industrie4.0-smart-manufacturing-for-the-future-en.pdf

[4] Zheng, Zibin, et al. An overview of blockchain technology: Architecture, consensus, and future trends. 2017 IEEE International Congress on Big Data (BigData Congress). IEEE, 2017.

[5] Crosby, Michael, et al. Blockchain technology: Beyond bitcoin. Applied Innovation 2 (2016): 6-10.

[6] Abeyratne, Saveen A., and Radmehr P. Monfared. Blockchain ready manufacturing supply chain using distributed ledger. (2016).

[7] Ethereum. Blockchain App Platform. Available online: https://ethereum.org/

[8] Vujii, Dejan, Dijana Jagodi, and Sinia Rani. Blockchain technology, bitcoin, and Ethereum: A brief overview. 2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH). IEEE, 2018.

[9] Solidity. Programming language for smart contracts. Available online: https://solidity.readthedocs.io/en/v0.5.3/

[10] Ouaddah, Aafaf, et al. Access control in the Internet of Things: Big challenges and new opportunities. Computer Networks 112 (2017): 237-262.

[11] Toumi, Khalifa, Ana Cavalli, and Mazen EL Maarabani. "Role based interoperability security policies in collaborative systems." 2012 International Conference on Collaboration Technologies and Systems (CTS). IEEE, 2012.

[12] Maesa, Damiano Di Francesco, Paolo Mori, and Laura Ricci. Blockchain based access control. IFIP International Conference on Distributed Applications and Interoperable Systems. Springer, Cham, 2017.

[13] Ouaddah, Aafaf, Anas Abou Elkalam, and Abdellah Ait Ouahman. FairAccess: a new Blockchainbased access control framework for the Internet of Things. Security and Communication Networks 9.18 (2016): 5943-5964.

[14] Cruz, Jason Paul, Yuichi Kaji, and Naoto Yanai. RBAC-SC: Role-based access control using smart contract. IEEE Access 6 (2018): 12240-12251.

[15] Dagher, Gaby G., et al. Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. Sustainable cities and society 39 (2018): 283-297.

[16] Zhu, Yan, et al. TBAC: transaction-based access control on blockchain for resource sharing with cryptographically decentralized authorization. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). Vol. 1. IEEE, 2018.

[17] C. Lin, D. He, X. Huang, K.-K. R. Choo, A. V. Vasilakos, Bsein: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0, Elsevier Journal of Network and Computer Applications, vol. 116, pp. 42-52, 2018.

[18] Toumi, Khalifa, Csar Andrs, and Ana Cavalli. Trust-OrBAC: A trust access control model in multi-organization environments. International Conference on Information Systems Security. Springer, Berlin, Heidelberg, 2012.

[19] Zhang, Jie. A survey on trust management for vanets. 2011 IEEE International Conference on Advanced Information Networking and Applications. IEEE, 2011.

[20] N. S. et al., Openid connect core 1.0, OpenID Foundation, Tech. Rep., February 2014

[21] Hardt, Dick. The OAuth 2.0 authorization framework. (2012).

[22] Geth Client for Building Private Blockchain Networks. Available online at: https://github.com/ethereum/go-ethereum/wiki/geth

[23] Lahbib, Asma, et al. "Blockchain based trust management mechanism for IoT." 2019 IEEE Wireless Communications and Networking Conference (WCNC). 2019. (to appear)