Securing the Integrity of Workflows in IoT

Prabhakaran Kasinathan
Siemens AG, CT IT Security, Munich, Germany
University of Passau, Germany
prabhakaran.kasinathan@siemens.com

Jorge Cuellar
Siemens AG , CT IT Security, Munich, Germany
jorge.cuellar@siemens.com

Abstract

In a multi-tenant, self-configuring IoT system, entities – devices, services or "things" – might know each other or might be complete strangers. The different owners will probably have different security goals and will want to impose their own security policy rules on their own entities. Thus, there is a need to negotiate a compromise and to interoperate the security policies of the different components. How will other devices react if a particular event arises?

In this paper we propose a framework to specify and manage workflows to be performed in a cyber-physical system, without assuming the availability of a centralized management system. Further, the method provides a formal background to guarantee the integrity of such processes and to enforce a least privilege principle for the authorizations required to execute the tasks in the workflow. More precisely, the proposed method a) supports the declaration of workflows to be executed in a given context, b) allow parties to propose and accept (or reject) "contracts" that describe the workflows in which they will participate, c) constrain an IoT application to obey a prescribed workflow, and d) restrict the access rights of subjects to secured objects for the execution of their tasks in the workflow, but not more.

We propose to use Petri Nets and Smart Contracts to specify and enforce workflows. This concept can also be applied to other application areas not restricted to IoT.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: General

Keywords

Internet of Things, Workflow, Integrity, Security

1 Introduction

The EU Research Cluster on IoT (IERC) [20] defines the Internet of Things (IoT) as an "infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual *things* have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network". An IoT system thus interacts with external entities, human users or services connected via the Internet. Today, IoT services are offered both by consumer appliances and by Industrial devices.

Internet of Things (IoT) applications are being used in various areas including smart manufacturing, industrial control, intelligent logistics, transportation, medical and healthcare applications, smart grid, intelligent traffic, environmental monitoring, smart home, assisted living, agriculture, and many more. In those applications, the main security concern relates to the processes themselves. Except for privacy of personal data, which is not our main focus here, confidentiality is not as important as the availability and integrity of the cyber-security processes, which are mission-critical. Due to the nature of the devices used, which are often constrained in processing power and in communication capabilities, IoT opens new vulnerabilities, in particular because keys are often not well protected in the devices or the cryptography on the channels is not too strong. Unfortunately, those vulnerabilities in constrained devices or networks can become the perfect entry point to escalate attacks to otherwise well-secured elements, in particular for DoS attacks.

Securing the IoT must imply a particular form of resilience: even in the case of breaches in single devices or networks, the system as a whole continues functioning correctly, due – partly – to redundancy of the data, but in particular because an attacker has no means to use the information or privileges obtained to gain elevated access to further resources.

Two relevant aspects of IoT applications are to be mentioned: self-configuration and multi-tenancy. The need for self-configuration appears because new devices or things may enter the application environment and should then – without too much human involvement – collaborate with the rest of the system. Also, some devices may fail, disappear or lose their connectivity. In both cases the system must readapt to the changes. Multi-tenancy refers to the fact that devices or services belong to different owners with different or competing goals. Those parties prefer to cooperate by exchanging information or collaborating on an activity than to work on its own, because they will profit from information or activities performed by another entity in exchange for other

International Conference on Embedded Wireless
Systems and Networks (EWSN) 2018
14–16 February, Madrid, Spain

ISBN: 978-0-9949886-2-1

information or activities.

A workflow can be defined as a pattern of activities or tasks to be completed in a particular partial order by the involved entities, following predefined rules, in order to accomplish a specific goal or sub-goal. During the execution of the workflow, the participants pass to each other documents, information or further tasks, see [23]. One of the main objectives of our framework is to support dynamic workflows, which respond to error conditions or based on resource allocation or just-in-time (JIT) considerations, as well as multi-tenant systems, where services, devices or things have different owners with different interests.

The other main motivation of our work is to provide an access control that restricts the entities to do only what is allowed in the workflow, but not much more. For this, we have to support a workflow-driven access control, in contrast to the commonly used mandatory (MAC), discretionary (DAC), or role-based access control (RBAC), which have been well-studied in the literature, see [18].

The rest of the paper is structured as the follows: section 2 presents related work, section 3 describes two use cases, section 4 describes our framework and the method used to enforce secure workflows, section 5 presents how our method can solve those two use cases and at the end, section 6 presents the conclusion and future work.

2 Related work

There has been extensive work on the specification and enforcement of workflows; in particular, [4] studied how to model and enforce workflow authorization constraints such as separation-of-duties in workflows, but using a centralized workflow management system. Workflow-driven access control is also well-known (see [12]), but mostly this requires predefined workflows (not created on the fly as response to unexpected conditions).

Petri Nets (see [16]) provide a graphical modeling tool used to describe processes performing an orchestrated activity, or in other words, a workflow (see [22]). Petri Nets have the advantage that many properties such as liveness (deadlock-freeness), reachability are easy to verify. (see [15, 17, 9]). Atluri et al., [1, 2] studied how to model workflows using Petri Nets, but did not describe the implementation details. Huang et al., [10] presented a web-enabled workflow management system and Compagna et al., [7] presented an automatic enforcement of security policies based on workflow-driven web application, but both works presented a centralized architecture.

The IETF working group ACE [11] is developing secure and efficient protocols for the IoT scenario with three actors: an authorization server, a client and a server (usually constrained). The authorization server provides a token (representing a permission) to the client to access a resource in the server. We have similar requirements in our method, for example, an authority of an entity in a given context can pass tokens to another; but, for the purposes of workflows, we need further types of tokens (not just permissions), as it will be explained later.

Smart Contracts, introduced in [21], have become popular with the advancements in blockchain technology. Smart contracts are often written to ensure fairness between participating entities even when one entity may attempt to cheat the other entity in arbitrary ways (see [8]). In [6] and [3] an example of an IoT application using Smart Contracts and Blockchains is presented. Bitcoins has a simple stack language to express the rules and conditions for a successful transaction and how new coins are produced and consumed. Ethereum, which has popularized the use of smart contracts, uses a Turing complete language to specify them. In [14], the authors have studied the security of running smart contracts based on Ethereum, and presented some problems in Ethereum's smart contract language solidity; they also show some ways to enhance the operational semantics of Ethereum to make smart contracts less vulnerable.

3 Use Cases

In this section we describe two Use Cases of IoT applications and their requirements. In Section 5 we will sketch how our proposal can be used to implement the Use Cases in a natural way.

3.1 Smart Manufacturing

Typical manufacturing plants produce few types of products in mass numbers. Nowadays, customers demand a high variety of products, customized to particular needs, with a high quality, but in smaller amounts. Manufacturers should adapt quickly to supply-chain disruptions, errors in manufacturing process and customer customization demands, i.e., manufacturing plants should be flexible. Smart Manufacturing improves the production agility, quality, and efficiency in manufacturing process, see [13]. Zhekun et al., [24] describes how RFID technology enabled smart-parts in manufacturing industries to implement unique-identification, communication between parts and manufacturing equipment; and to improve flexible and concurrent product manufacturing, and quality of the products.

3.1.1 Smart Manufacturing Use-Case Problems

- The manufacturing company should be able to monitor the production progress of an individual product, and to track the location and use of the single parts in real time.
- The manufacturing plant should be able to get information from their suppliers and signalize the production process accordingly i.e., to quickly adapt to supply-chain changes, and to recover from errors during production.
- The manufactured product should have production and testing data available continuously for inspection and be able to react to test results on real-time.
- The manufacturing plant should be flexible enough to produce products with customer customizable options, without compromising the quality.

3.2 Building Automation

Modern buildings are equipped with embedded devices used within various automation systems, for instance, for the purposes of lighting, heating, ventilation, physical safety, etc. The devices contain sensors and actuators and collaborate autonomously. For example, the lighting system can adjust the light intensity and color of a room based on the ambient light available in the room; the security system can alert the nearest emergency responders or fire-stations in case of an emergency. In such a scenario, often it is required to perform software-updates, quality-control inspection, fix security patches and upgrade the firmware on the devices. Usually, the building owner delegates the installation or maintenance work to a contracting company. The RFC 7744 [19], provides a summary of authorization problems that emerge during the device life-cycle (commissioning, maintenance, re-commissioning, decommissioning). In addition to the authorization problems, the building owners may wish to ensure that only products with a certain provenance or quality are installed, and that the process complies to standard operating procedures. The building owner may also wish that the contractor obeys other conditions written on a contract.

3.2.1 Building Automation Use Case Problem

- The building owner wants to track the status of the entire process remotely, for instance, installation or maintenance process in real-time.
- The building owner wants to monitor, enforce automatically the agreed conditions with the contractor; for instance, if the contractor breaks any agreed condition, then a penalty can be enforced.
- The building owner wants to configure the installed devices with custom-rules, and the new installed devices should be interoperable with existing systems and devices.
- The building owner should be able to give/revoke fine-grained authorization permissions to the contractors enforcing the least privilege principle.

4 Specifying and Enforcing Workflows in IoT

In this section we present a framework and its components for specifying and enforcing workflows in IoT. We describe why we need such a framework by describing the requirements for an IoT system.

One of the problems in distributed multi-tenant IoT systems is the assumption that all of the humans, services, devices, and things will follow a coherent set of rules. Since we do not want to impose an external overarching authority to force them to act according to a consistent overall workflow, we propose a method that can be used by rather independent agents (owned by perhaps different autonomous parties) to collaborate in a workflow. Our framework allows different entities to agree to play a part in a workflow by promising to constrain its behavior in various ways. This point of view is similar to the one of Promise Theory, see [5], but the methods proposed for expressing the assumptions and commitments are quite different.

In discretionary access control, the permission to use a service is given by the owner of the service (or of the data). This is a reasonable assumption in IoT, since the owner of the device is the one that controls the keys or cryptographic material that the device uses for authentication or authorization purposes. We propose that the different tenants (or their devices, based on policies established by the owners) share the responsibility of creating and enforcing workflows in a decentralized and plug-and-play fashion. In the same way that each owner is able to give permissions to access his owned services, we also want that each different owner is the authority, who is able to decide which sequences of tasks should be performed on the devices or services.

A similar but somewhat different situation is well-known from workflows in hospitals or public authorities: a person may be asked to "take this document, bring it to office 205, pay 10 Euros, obtain a signature, go to office 405 to obtain a second signature (after some verification procedure) and then go to office 101 to obtain further instructions". We call this a *partial workflow*: the person agrees to follow the instructions in order to obtain a desired result or to progress in that direction. The common feature of this type of situation is that there are different authorities that decide which are the different partial workflows that should be followed for a successful cooperation.

In static access control systems, an entity has access rights to a resource at any time, thus the entities may abuse the permissions for purposes not foreseen. We want to have an access control system where the entity (*subject*) may only get the access rights (*permissions*) to access the resource (*object*) for a defined step in a workflow and for a limited time period. A *workflow aware access control* can be used to restrain an entity or a malicious attacker – even if he gains access to the system – from performing actions that are not specified in the workflow.

The Workflows can also be used by organizations to implement internal security policies and privacy procedures within the application processes. The participating entities are allowed to accept or reject conditions with respective consequences, this provides flexibility within the workflow.

A further complication is that the devices may have been provided by different manufacturers, have different specifications and implement "equivalent" tasks in a different manner. This implies that the workflows must be specified at a higher layer, abstracting away from implementation details One of the main motivations of this work is to secure the integrity of workflows in IoT applications.

First, we need a high level workflow specification language to express a process in a given context as a workflow. The workflow specification language should be amenable to lightweight formal methods, so that the different parties can reason locally about them.

Second, a workflow enforcement method is required to make sure that the involved entities obey the agreed workflow. We do not impose a central authority to enforce workflows; but the owner to enforce the workflow activities on their set of devices. For example, the owner of a constrained IoT device can create a workflow for a client entity that needs to access the IoT device; the client is enforced to execute the workflow to access the service from the constrained device.

We believe that the way to secure a clear and consistent set of rules in complex environments is using formal methods. For this, we propose to use Petri Nets and Smart Contracts, written in a simple declarative language. This also permits the entities to reason locally about their possible choices and their decisions, allowing plug-and-play configuration. Such a local reasoning facility will be necessary to support the dynamic configuration of security policies, but that is not the main focus of this paper.

4.1 Petri Nets for Workflow Specification

The advantages of using Petri Nets to specify workflows are:

- Workflows specified in Petri Nets enable us to create error free workflows because various properties of Petri Nets such as reachability, liveness (deadlock-free), and coverability [15, 22, 9] can be verified.
- Hierarchical Petri Nets can simplify the process of creating complex workflows by breaking them into smaller partial workflows.

In traditional Petri Nets there are places, tokens and transitions. If there is risk of confusing the Tokens (markings) in Petri Nets and the Tokens used for instance in ACE (say, oauth Tokens) which are passed from one entity to another, we call the later ones as oauth-tokens wherever necessary. A transition may have one or more input places, and a place may have one or several tokens. A transition fires if its input places have sufficient tokens and as a result it produces tokens in output places. Entities interacting with the Petri Net workflow change their state from one place to another via a transition firing. Extensions of Petri Nets such as colored Petri Nets have enabled Petri Nets to represent different types of tokens in one place.

In our Petri Net model for workflows, we introduce two additional concepts:

- A new type of place (called an *oracle*) that can receive tokens from an external source and it is represented as star shape in our Petri Nets. Tokens can represent information, endorsements or permissions, for instance implemented as oauth-tokens.
- Some transitions within a Petri Net workflow have additionally a *smart contract*, described in the section 4.2.

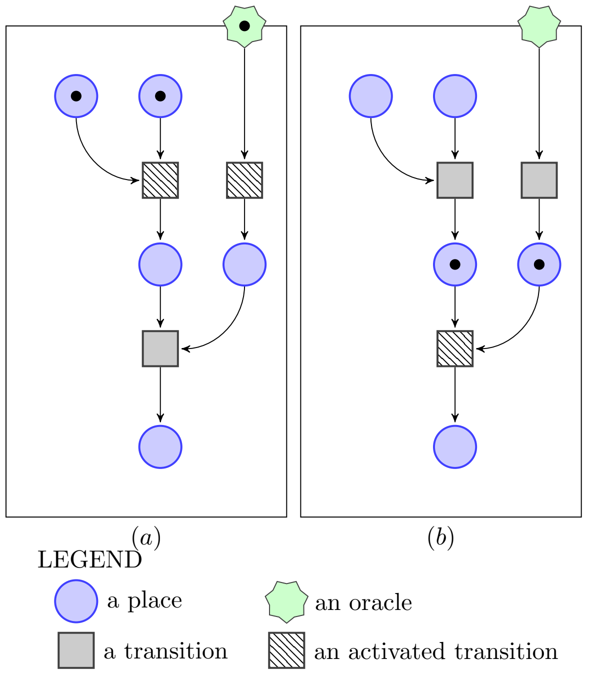

Figure 1. Petri Net Workflow Specification

The Figure 1 shows a simple workflow specified as a Petri Net. The oracle is represented as a star and all other places are represented as circles. A place that contains or holds a token is marked with a small black circle. The transitions waiting for tokens are presented as squares without patterns, and activated transitions ready for firing are shown as patterned squares. In Figure 1 (a) shows that the first two transitions are enabled (ready to fire) because the input places have tokens, and the Figure 1 (b) shows that those two transitions have fired and as a result produced tokens in the output places.

The subjects wanting to access protected resources interact with the Petri Net workflow. A subject can be a process, machine, or a human. The objects expose services which are usually the protected resources in the workflow. A place in the workflow can represent the state of both subjects and objects, which we collectively call entities. A token in this case can represent that the entity is available on the particular state in the workflow. In a more general setting, a token might mean that the subject is ready for an interaction with another subject.

A workflow is defined to express and fulfill a particular purpose. For example, a workflow can be defined such that a subject may have access to append to a file (object) only to log status of the completed activity, not for anything else. A transition first evaluates whether the subject is allowed to access the object; second it evaluates whether the authorized subject is allowed to access the object for the particular action defined in the workflow or not. If both conditions satisfy, then an access token is granted to the subject for a limited time period to complete the task specified in the workflow; once the task is completed, the access is revoked. Workflows are written such that the different entities work together to complete a purpose.

4.2 Transition Contracts

In traditional Petri Nets, a transition fires when the input places of the transition has sufficient tokens. To implement a workflow-driven access control system in Petri Nets the transitions should be able to verify conditions and evaluate information encoded in the tokens.

We use the combination of Petri Nets and transition contracts to specify and enforce sequences of atomic transitions (transactions) and properties that must be satisfied during the single transitions. The properties (or rules) for each transition may be seen as small smart contracts that restrict the choices of the participants of the workflow for this step, or they impose additional conditions on the required or created during the transaction, and the Petri Net the restrictions in the order that they perform the steps. The conditions on a single transition will be simply called a *transition contract* (or *transaction contract*). This combination allows us to create *multi-step smart contracts*: say, in the first step a token is created based on some conditions (which may verify authentication or authorization status of participants), and then this token can only be used in a subsequent transition in a particular way, determined by the Petri Net and the next transition contracts.

A transition contract adds conditions to the firing of a transition: the transition procedure takes inputs from the entities which engage in the transition and from external sources, like an authenticated data feed, processes them (in particular, verifying the validity of the tokens), evaluates the conditions described (as guarded commands) in the transition contract, and produces the outputs as expected and as specified in those conditions. An output produced by the transition contract can be a token representing a particular information or can even be a dynamically created partial workflow for one or more entities.

The Figure 2 shows a simple Petri Net where two transitions (*T1 and T2*) have a pointer to the transition contracts (*TC (a) and TC (b)*) respectively. We propose to use a simple guarded command (a conditionally executed statement) language to write conditions on a transition contract, because of the security bugs (see [14]) and problems that exist in Turing complete smart contract languages such as in Ethereum. Note: smart contracts do not always have to run on blockchain, they can also be implemented between two or more parties without blockchain technology.

In our method we show that it is possible to bind many transition contracts grouped together in a sequence to enforce a workflow specified in Petri Nets.

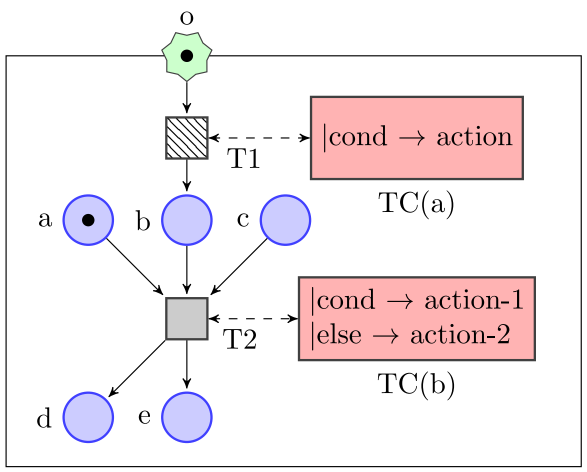

Figure 2. Smart Contract: Petri Net with Transition Contracts

4.3 Local Reasoning

IoT devices are manufactured by different manufacturers possibly with various standards, therefore an IoT system should tackle interoperability issues, authentication and authorization problems. IETF standardization groups such as ACE [11] are working towards addressing the authentication and authorization problems.

To enable dynamic configuration of security policies, an IoT system should support local reasoners. A local reasoner is a software module that takes decisions by analyzing available facts, for instance, it uses on-device information and contextual information from nearby devices to take decisions. Let us consider a home automation system that has access to your database (that contains your friends list), and a security policy stating that your friends can have wifi access in your home; if your friend is able to prove to the home automation system that he is one of your friend, then the friend's device can get wifi access in your home. This will allow devices to have plug-and-play functionality and they can mutually authenticate between themselves.

5 Solving Use Cases

In this section we show how to solve the use cases presented in section 3 using our approach in a natural way.

5.1 Smart Manufacturing

A manufacturing plant should be able to produce customizable products on demand, adapt to supply chain disruptions, and recover from errors during the production process with some level of automation. To realize such a manufacturing plant, we need to introduce some intelligence to the parts and manufacturing equipment involved in manufacturing process. The parts and equipment are smart because they have embedded processors or at least an RFID to hold information. The equipment has computing and communication capabilities; therefore, they can read and write the workflow, verify conditions, and can communicate with other equipment and parts.

Let us consider that a customer places a customized order that requires some customizable parts and a core part to produce the final product. A part is able to listen to a request from an equipment, for instance, an assembly unit; and can reach a particular place in a given time. A partial workflow that describes what types of parts are required and the assembly instructions are uploaded to the core part. When the core part is ready for assembling, the equipment sends a request for the required parts; if the necessary parts and assembling machine are available, then they all agree and reach the assembly unit on a particular time. This enables JIT manufacturing.

Additionally, particular conditions can be checked and enforced via a workflow. For example, we can define that the assembled product should pass the stress test with a certain criteria to proceed to next step in a workflow, else a new workflow is uploaded to fix that error. In this way, the workflow can be used for enforcing production methods and standards. The equipment can reason locally about the production process errors, disruptions in supply chain, and can recover from them. Once the tasks described in a workflow are completed, then a new partial workflow is uploaded to the part by the equipment. The workflow status enables the manufacturing company to monitor the status of the production, parts and equipment involved in real time. This enables the manufacturing plant to gain production agility, quality and efficiency, and to realize the of JIT model. This use case shows how our method can fit within the existing manufacturing systems.

5.2 Building Automation

Assume that a contracting company agrees to maintain the existing building automation devices, install new devices and configure them with custom configurations, provided by the building owner, and to finish the work in time or incur some penalties.

A workflow is defined to express the above conditions by the building owner and as agreed by the contracting company. Let us consider that the workflow can be uploaded to a computing device (for example, a handheld or smartphone) which has a generic software application capable of executing any Petri Nets workflows and transition contracts specified within them. The participants involved in the workflow, for example, building owner, contracting company and their employees are able to interact with their handhelds.

Let us consider that as the first step the building owner or an authorized employee presses a button on his handheld and approves the contractors to begin the work; this event creates a signed-token in an oracle and enables a transition. The transition may point to a transition contract to verify whether the token is from an authorized employee or not. Assume that the transition contract is able to verify the signature of the token using pre-configured certificates, if the token is valid, then the transition contract can create an oauth-token for the contractor to access the devices for maintenance purposes. Similarly, tokens (can also be permissions or information) required by the contractor for other purposes defined in the workflow are created. After completing the authorization step, the next transition of the workflow may point to a transition contract that may point to the custom configuration file chosen by the building owner to be installed on the devices. The contractor uses the configuration file to configure the devices.

The contracting company might want to enforce specific conditions by creating partial workflows on their employees, and these workflows can be provided together with the main workflow provided by the building owner. Here we show that the owner of the task is able to create dynamic partial workflows for other entities to complete a task or resource that he owns. In this way by we have realized a distributed workflow management system. This use case shows how we can execute and enforce a workflow in a distributed setting.

Transition contracts obtain inputs from the devices describing their provenance, verify the results, and if the results are as expected, then the Smart Contract can pay the contractor with the amount that was agreed, else appropriate penalty is enforced.

6 Conclusion and Future Work

In this paper, we have described a new framework for securing the integrity of workflows in an IoT application by using Petri Nets and Smart Contracts. The method provides flexibility for entities interacting within a workflow; for instance creation of dynamic partial workflows. By integrating workflow driven access control, our method enforces the least privilege principle to authorize entities participating in the workflow. We have also studied two real world use cases and provided a solution to tackle those use-case problems.

As future work we intend to define a simple transition contract language, create a user-friendly tool to create, verify, and enforce workflows in an IoT application environment, implement a simple reasoner that can fit inside constrained IoT devices, and to evaluate our distributed workflow enforcement model.

7 References

[1] V. Atluri and W.-K. Huang. An Authorization Model for Workflows. *Computer Security - ESORICS 1996 - Proceedings of the 4th European Symposium on Research in Computer Security*, 1146:44–64, 1996.

[2] V. Atluri and W.-K. Huang. A Petri net based safety analysis of workflow authorization models. *Journal of Computer Security*, 8(2/3):209, 2000.

[3] A. Bahga and V. K. Madisetti. Blockchain Platform for Industrial Internet of Things. *Journal of Software Engineering and Applications*, 9:533–546, 2016.

[4] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security*, 2(1):65–104, 1999.

[5] M. Burgess and S. Fagernes. Promise theory - A model of autonomous objects for pervasive computing and swarms. In *International Conference on Networking and Services 2006, ICNS'06*, page 118. IEEE, 2006.

[6] K. Christidis and M. Devetsikiotis. Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, 4:2292–2303, 2016.

[7] L. Compagna, D. R. dos Santos, S. E. Ponta, and S. Ranise. Aegis: Automatic Enforcement of Security Policies in Workflow-drivenWeb Applications. *Proceedings of ACM on Conference on Data and Application Security and Privacy - CODASPY '17*, pages 321–328, 2017.

[8] K. Delmolino, M. Arnett, A. E. Kosba, A. Miller, and E. Shi. Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab. *IACR Cryptology ePrint Archive*, 2015:460, 2015.

[9] J. Esparza. Decidability and complexity of Petri net problems—an introduction. In *Lectures on Petri Nets I: Basic Models*, page 55. Springer, Berlin, Heidelberg, 1998.

[10] W.-K. Huang and V. Atluri. SecureFlow: A Secure Web-enabled Workflow Management System. *Proceedings of the fourth ACM workshop on Role-based access control - RBAC '99*, pages 83–94, 1999.

[11] IETF ACE Working Group. Authentication and Authorization for Constrained Environments (ACE). *The Internet Engineering Task Force (IETF)*, 2017.

[12] K. Knorr. Dynamic access control through Petri net workflows. *Proceedings - Annual Computer Security Applications Conference, ACSAC*, 2000-Janua:159–167, 2000.

[13] Y. Lu, K. Morris, and S. Frechette. Current Standards Landscape for Smart Manufacturing Systems. *National Institute of Standards and Technology, NISTIR*, 2016.

[14] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making Smart Contracts Smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*, pages 254–269, New York, New York, USA, 2016. ACM Press.

[15] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, apr 1989.

[16] C. A. Petri. Communication with automata. *http://edoc.sub.uni-hamburg.de/informatik/volltexte/2010/155/pdf/diss_petri_engl.pdf*, 1966.

[17] W. Reisig. *Petri Nets : an Introduction*. Springer Berlin Heidelberg, 1985.

[18] R. S. Sandhu and P. Samarati. Access Control: Principles and Practice. *IEEE Communications Magazine*, 32(9):40–48, sep 1994.

[19] G. Selander, M. Mani, and S. Kumar. Use Cases for Authentication and Authorization in Constrained Environments. *IETF RFC 7744*, 2016.

[20] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé. Vision and challenges for realising the Internet of Things. *Cluster of European Research Projects on the Internet of Things, European Commision*, 3(3):34–36, 2010.

[21] N. Szabo. Smart Contracts: Building Blocks for Digital Markets Copyright, 1996.

[22] W. M. P. van der Aalst. Verification of workflow nets. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1248, pages 407–426. Springer, Berlin, Heidelberg, 1997.

[23] WfMC. Workflow Management Coalition, 2009.

[24] L. Zhekun, R. Gadh, and B. S. Prabhu. Applications of RFID Technology and Smart Parts in Manufacturing. In *Volume 4: 24th Computers and Information in Engineering Conference*, volume 2004, pages 123–129. ASME, jan 2004.