Security and Trust in Data Sharing Smart Cyber-Physical Systems

Ondřej Štumpf
Faculty of Mathematics and Physics
Charles University
Malostranské náměstí 2/27
Prague, Czech Republic
stumpf@d3s.mff.cuni.cz

Tomáš Bureš
Faculty of Mathematics and Physics
Charles University
Malostranské náměstí 2/27
Prague, Czech Republic
bures@d3s.mff.cuni.cz

Vladimír Matěna
Faculty of Mathematics and Physics
Charles University
Malostranské náměstí 2/27
Prague, Czech Republic
matena@d3s.mff.cuni.cz

ABSTRACT

Security and trust plays an important role in Smart Cyber-Physical Systems (sCPS), which are formed as open and large collections of autonomous context- and self-aware adaptive components that dynamically group themselves and cooperate (all in a rather decentralized manner). Such a high level of dynamicity, open-endedness and context-dependence however makes existing approaches to security and trust in distributed systems not fully suitable (typically being too static and not able to cope with decentralization). In this paper we introduce the concepts of context-dependent security and trust defined at the architecture level of sCPS. Contrary to traditional approaches, our solution allows components to adapt their security clearance according to their context (i.e. their state and the surrounding environment), while preserving high level security policies. We further define the interplay of security and trust in sCPS and show their interrelation as an important ingredient in achieving security in systems of adaptive autonomous components.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection – *access controls, information flow controls;* D.2.11 [**Software Engineering**]: Software Architectures – *Domain-specific architectures, Patterns.*

General Terms

Design, Security.

Keywords

Smart cyber-physical systems, architecture, security, trust.

1. INTRODUCTION

In recent years, research and development of widely distributed information systems has made a tremendous progress. Cyber-Physical Systems (CPS) of today may consist of vast number of entities with ability to communicate with each other, creating added value for their users. Such modern CPS are sometimes called Smart CPS (sCPS) as they take on a significant role in their own reaction and evolution to unforeseen situations. Similarly, the area of Internet of Things (IoT) revolutionized the very concept of networked systems by enabling embedded devices such as home appliances, cars or health monitors to share their data on-line.

The undisputed benefits of such systems are however burdened with security, privacy and trust issues. Each entity in the system may contain data of various levels of confidentiality, which it may (and should) want to protect by restricting access to them. Every entity should also be inherently suspicious about other entities in the system, particularly about using their data. This makes the two concepts of security (i.e. restricting access to one's data) and trust management (i.e. regulating how data of others are relied upon) inherently inseparable.

These concerns and solutions to them are nothing new to the scientific literature, where role-based access control is understood as a baseline and further approaches build on it, extending the ability to cope with a large number of entities in distributed settings (such as in IoT [10]).

An important point of these systems, however, is that they assume static (i.e. unchanging) context. This holds true even for state-of-the-art approaches for distributed role-based access control (dRBAC) [4], attribute-based access control (ABAC) [12] and distributed capability-based access control [10] even specifically adapted for operation in Internet of Things. This makes them not fully suitable for modern smart cyber-physical systems, which are formed as a large collection of autonomous components that dynamically group themselves and cooperate, while adapting their behavior to their specific context. Another important trait of sCPS is that they are decentralized and required to work with limited connectivity (e.g. relying on MANET/VANET based communication).

In this paper, we focus specifically on introducing context-dependent security and trust to sCPS. Contrary to traditional approaches to security and trust, we advocate that the security and trust must not be treated in isolation from the rest of the system design as this creates a semantic gap between the two. As such, we introduce the security and trust at the architectural level and connect them with the notion of autonomous components and their dynamically established cooperation groups, so called component ensembles (a concept which recently emerged in the scientific literature to capture intelligent self-organization of autonomous components [14]).

2. BACKGROUND AND RUNNING EXAMPLE

We demonstrate our approach on a running example of a sample police-assistance sCPS. In the example, we assume regular vehicles, state police vehicles and municipal police vehicles. We assume for the sake of illustration that the vehicles can communicate. The regular vehicles share information about their owner and are equipped with a GPS module to determine their position. The police vehicles moreover use a police speed radar to monitor regular vehicles nearby. If any regular vehicle moves too fast or is driven by a wanted criminal, the police vehicle starts a pursuit. While the state police has the jurisdiction to execute law everywhere, the municipal police can only access information about vehicles which are registered in the same city.

From the perspective of the architecture, a vehicle is captured by an autonomous component, the opportunistic communication among them is captured by ensembles [14]. Strictly speaking, a vehicle may contain a number of devices, each of which can host a number of components. However for simplicity of explanation and without impacting the generality of the security model presented further in the paper, we associate a component with a device – in our case, a vehicle.

In component systems relying on ensembles of autonomous components, a component is typically defined by its state (i.e. the data it owns) and behavior. In DEECo [3] – a representative of ensemble-based component system which we will use to exemplify our approach – the data are termed knowledge and behavior is captured by periodic real-time processes. Referring to the running example (see Listing 1), a state of the vehicle includes the owner information and the position. The vehicle further comprises a process that regularly updates the position in the knowledge – e.g. based on reading of a GPS sensor. A police vehicle further includes knowledge of vehicles around and a process that checks for suspects among the vehicles nearby and potentially starts a pursuit. The municipal police vehicle extends this further by defining a knowledge field that captures the city in which the vehicle has jurisdiction.

Components are organized into ensembles (cooperation groups), which are dynamically created depending on the current components in the system and their state. An ensemble takes a role of a dynamic connector through which components communicate. In DEECo, an ensemble is defined by a predicate (membership condition based on components state) and the knowledge exchange. The knowledge exchange defines how to transfer data from one component to the other, if the membership predicate is satisfied. The ensemble in Listing 1 updates a police vehicle with information about all vehicles nearby.

```
1.  component Vehicle
2.      knowledge: id, ownerInfo, position, ...
3.      process updatePositionBasedOnGPS(out position) { ... }
4.  component PoliceVehicle extends Vehicle
5.      knowledge: vehiclesNearby, ...
6.      process checkForOffenders(in vehiclesNearby) { ... }
7.  component StatePoliceVehicle extends PoliceVehicle
8.  component MunicipalPoliceVehicle extends PoliceVehicle
9.      knowledge: cityOfJurisdiction
10. ensemble UpdateVehiclesNearby:
11.     coordinator: PoliceVehicle
12.     member: Vehicle
13.     membership: isCloseBy(coordinator.position, member.position)
14.     knowledgeExchange: coordinator.vehiclesNearby +=
        member.ownerInfo
```

Listing 1: Definition of components and ensembles of the running example

2.1 Security and Trust Issues

An obvious need in systems of autonomous components is a mechanism of limiting access to data. In the running example, a regular vehicle component may not want to share sensitive owner information with everyone – the access should only be granted to police vehicles. Moreover, even though they do have read permissions, police components should not be able to modify the owner data. Similarly, the municipal police should not be allowed to obtain information about vehicles outside of its jurisdiction.

The jurisdiction of the municipal police is determined by a knowledge field in the municipal police component. This brings an obvious advantage that the jurisdiction is dynamic and can be adaptively changed by appropriate authority based on the current context. For example in case of emergency or cross-city coordinated pursuit, municipal police vehicles may be temporarily allowed to exercise authority in another city. However, this also means that a chain of trust has to be established, making it possible to guarantee that the value of the city of jurisdiction field originates from someone who has the authority to set it.

3. ADDRESSING SECURITY AND TRUST IN DYNAMIC CONTEXT

In our solution, we propose an architecture-level access model with security roles parameterized by components' knowledge. This provides mechanism for limiting the context in which component's security role is valid (as we have seen in the running example – the municipal police only has jurisdiction in the particular city). We call it Context-Dependent Role-Based Access Control model (cdRBAC). We further describe the technical realization on the communication level, allowing it to be applied for open-ended and decentralized systems that rely on knowledge sharing. The Domain Specific Language (DSL) in Listing 2 illustrates our approach. It shows the developers' perspective of the security specification, which is wholly defined on the component architecture level by extending it with security metadata. (Note that we have extended the example by adding the police station component that serves as an example of obtaining initial parameters for dynamically parameterized security roles.)

```
1.  role IdEntity(id [const])
2.  role MunicipalPolice(city [authoredBy PoliceStation, StatePolice])
3.  role StatePolice implies MunicipalPolice(*)
4.  role PoliceStation(city [const])
5.  component Vehicle hasRole IdEntity(id)
6.      knowledge [public]: position
7.      knowledge [public read, const write]: id
8.      knowledge [MunicipalPolice(positionToCity(position)) read]:
            ownerInfo
9.  component StatePoliceVehicle extends PoliceVehicle hasRole
    StatePolice
10.     ...
11. component MunicipalPoliceVehicle extends PoliceVehicle hasRole
    MunicipalPolice(cityOfJurisdiction)
12.     knowledge [public read, PoliceStation write]: cityOfJurisdiction
13.     ...
14. component PraguePoliceStation hasRole PoliceStation("Prague")
15.     knowledge [IdEntity(key) read, const write]:
            ownVehiclesMap = [728 -> "Prague", 961 -> "Prague", ...]
16.     ...
```

```
17. ensemble MunicipalPoliceCityAssignment:
18.     coordinator: PoliceStation
19.     member: MunicipalPoliceVehicle
20.     membership: coordinator.ownVehiclesMap.contains[member.id]
&& ...
21.     knowledgeExchange: member.city =
            coordinator.ownVehiclesMap[member.id]
```

Listing 2: Extension of the running example with security and trust specification

There are a few key concepts in our approach:

- Each component may have any number of security roles (lines 5, 10, 12, 15 of Listing 2) and security roles are inherited via **extends**-relationship between components. Security roles may form a hierarchy, where one role may imply others (line 3). The security role is guaranteed by providing a certificate for the component stating the role.
- A security role may be parameterized (lines 1-4) and the parameter qualified by stating which security role is allowed to provide the value of the parameter (line 2). Alternatively, the parameter may be required to be constant, which means that the value must be set at the time of deployment (lines 1, 4) – in this case, the certificate for the component states not only its role, but also permitted values of the parameter.
- Knowledge of a component may be qualified as to which security role is allowed to read it and which is allowed to provide values (lines 6-8, 13, 16).

Note that while the security roles used to restrict the **read** of knowledge correspond to traditional notion of security, the roles restricting the **write** and the origin of security role parameters (i.e. **authoredBy**) correspond to trust.

The concept of ensembles remains intact, however, there is a design pattern that prevents a rogue ensemble from assigning a trusted value (e.g. to the cityOfJurisdiction field) to a wrong component (e.g. to a police vehicle belonging to another city). The pattern lies in restricting the access to the trusted value only to the particular components that have the right to use the value in their role parameter. In our example this is done by assuming a globally unique component id and equipping every component with a security role that is parameterized by this constant id (line 1, 5, 7). The trusted value is in our example provided by the police station component (line 15) in the regular case and by the state police vehicle in the emergency case. The police station provides the trusted value through its field ownVehiclesMap, in which the read access to each entry is restricted only to the component having the same id as is the key of the entry.

4. TECHNICAL REALIZATION

Unlike more traditional and static systems, sCPS cannot rely on stable connectivity and on any kind of centralized authority that is always reachable. Rather, they employ heterogeneous fabric networks (including MANETs/VANETs, mobile networks and Wi-Fi). Similarly to communication means used by distributed real-time systems (e.g. the CAN bus), sCPS typically exploit some way of data sharing (internally realized by periodic messages) as the primary means of communication. This allows them to be resilient to temporary disconnections. As a result, the knowledge of a component is propagated through the system to all components that could potentially be interested in the data. From the architectural perspective, this propagation of the knowledge to target components is governed by ensembles definition.

To achieve security in this setting, it is necessary to ensure that a rogue component cannot access protected knowledge of other components and cannot fake trusted knowledge. We realize our solution by the following principles:

- We assume a trusted certification authority used offline to sign a component and certify that it has the claimed security roles.
- The certification authority assigns offline a public/private key pair to each security role. Parameterized security roles are considered without actual parameters. Components featuring a role possess the respective public/private key pair. Components producing knowledge that is readable by a certain role possess the respective public key.
- Each value of a parameterized role is associated with a public/private key pair. Such pairs are issued offline by the certification authority for roles with constant parameters and at runtime for non-constant parameters by components that have a role which is used in **authoredBy** (e.g. the PoliceStation).
- Whenever knowledge is sent, it is encrypted based on the public key of the role that is allowed to read the knowledge. If more roles can read the knowledge, the message is encrypted and sent multiple times, each time encrypted with the key respective to the given role. (Technically, to improve performance, hybrid cryptography is used.)
- When read access to knowledge is secured by a parameterized role, the knowledge is first encrypted based on the public key of the security role, and then this encrypted knowledge is further encrypted based on the public key of the role parameter value. If the role is parameterized by more parameters, the encryption is chained and performed for each parameter.
- Decryption is by using the keys in reverse order.
- Such knowledge that should be trusted is signed by the private key of the role from which the knowledge originates.

Note that the whole mechanism typically does not have to do any key exchange as the private and public keys are in major cases known in advance. The only exception is when read access to knowledge is restricted by a dynamically parameterized role. In that case, the public key of the parameter value has to be advertised by the recipient component (for instance MunicipalPoliceVehicle advertises the public key for its value of cityOfJurisdiction); they key is signed by the authority from which the trusted value has been received. This enables the regular vehicles in our running example to know how to encrypt their ownerInfo knowledge field.

Note also that to efficiently deal with the universal parameter value "*" (see the definition of the StatePolice role), it is necessary to employ hierarchical key assignment schemes [15, 16]. These schemes allow generation of a child's public key (i.e. the key of the particular parameter value) based on parent's public key (i.e. the key associated with "*" for a particular **authoredBy** role); and similarly for private keys. This allows the role owning the private key for "*" to decrypt knowledge regardless of a particular parameter value as long as public keys of parameter values are derived from the public key of "*".

5. INDICATIVE EVALUATION

To provide indication on performance impact brought by our cdRBAC model, we have implemented the basic security mechanisms in our Java implementation of DEECo and employed it with an implementation of the running example. The impact of security on performance is twofold – increased number of messages (each for different security role and values of

parameters, as described in the previous section) and extra time spent with encryption and signature generation. During a 10 minutes long simulation of the running example, the average number of messages sent with access control in place was 20% higher. The average extra time was 51%. The vast majority of the extra time is spent in the library methods for encrypting the messages. (The experiment methodology included 10 independent runs each after computer reboot.)

6. RELATED WORK

Similar to our approach (cdRBAC) to access control has been taken by Freudenthal et al. in their design of dRBAC (Distributed Role-based Access Control Management) [4]. Unlike in cdRBAC, the assignment of security roles is not signed by a globally trusted certification authority, but rather by any other subject in the system. This concept allows subjects not only to be granted certain roles, but also to be granted a right to delegate certain roles to others. While this is more dynamic and provides better support for delegation, to verify subject's access rights, it is necessary to backtrace the delegation chain and perform verification on each element, which can be a performance issue. The dRBAC systems targets this problem by introducing the proof monitors (an entity can be notified about changes in the delegation chain) and caching of delegations.

The RT framework [5] addresses the problem of decentralized access control by introducing a combination of RBAC, trust management and declarative description of policies. The RT_1 extension also contains a very similar concept of parametrized roles. Since the DATALOG language is used to specify access rights, it is also possible to enforce a separation of duty (SoD) principle through logic programming, unlike cdRBAC, where this is a responsibility of the certification authority.

When designing a security solution for the Internet of Things (IoT), several additional issues need to be addressed. Most importantly, devices in IoT may have very limited resources – memory, power, storage etc. In such environment, even a HTTP protocol is too complex and new protocols such as CoAP [9] are therefore being designed to access and control the devices.

Several access control models are built over such protocols. Hernández-Ramos et al. [10] propose their solution based on capabilities, which is designed to be as simple and as power-saving as possible. When a subject wants to perform certain operation on an object, it must first obtain corresponding Capability Token from an Issuer. The Token contains information about the object, the operation to be performed and additional conditions to be checked on the target device. To prevent tampering, the Token is signed. The Subject then issues the request to the object and attaches the corresponding Token – the device hosting the Object simply checks if the request complies with the Token and eventually performs the operation. This approach is primarily designed for situations where the target device knows all subjects that may access it. Such assumption is legitimate in certain use cases (e.g. smart home can be controlled only by family members), but in others can be problematic.

7. CONCLUSION

In this paper, we have described a context-dependent role-based security and trust model applicable to modern sCPS and IoT applications and demonstrated its inclusion in an ensemble-based system (DEECo). The main novelty of the proposed solution lies in the merger of architecture-level design, adaptability to dynamic context and synergy between security and trustworthiness. We have also provided indicative results on performance, by measurement performed on an implementation of the running example and a basic implementation of the security model for DEECo.

8. ACKNOWLEDGEMENT

The work on this paper has been partially supported by the Charles University Grant Agency project No. 391115 and Charles University institutional funding SVV-2015-260222.

9. REFERENCES

[1] Ferraiolo, D. F. and Kuhn, D. R. *Role-Based Access Controls.* In 15th National Computer Security Conference (NCSC, 1992), pp. 554-563.

[2] Clark, D. D. and Wilson, D. R. *A Comparison of Commercial and Military Computer Security Policies.* IEEE, 1987.

[3] Bures, T., Gerostathopoulos, I., Hnetynka, P., Keznikl, J., Kit, M., Plasil, F. *DEECo – an Ensemble-Based Component System.* In Proceedings of the 16th International ACM Sigsoft symposium on Component-based software engineering (CBSE '13). ACM, New York, NY, USA, 81-90.

[4] Freudenthal, E., Pesin, T., Port, L., Keenan, E., Karamcheti, V. *dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments.* Proceedings of ICDCS 2002.

[5] Ninghui, Li and Mitchell, J. C. *RT: A Role-based Trust-management Framework.* In DARPA Information Survivability Conference and Exposition, 2003, pp.201-212 vol.1, 22-24 April 2003.

[6] Venkatasubramanian, K. K. *Security Solutions for Cyber-Physical Systems.* Arizona State University, 2009.

[7] Yao, W. *Trust management for widely distributed systems.* University of Cambridge, 2008. ISSN 1476-2986.

[8] Sandhu, S. R., Coyne, E. J., Feinstein, H. L., Youman, C. E. *Role-Based Access Control Models.* In IEEE Computer, vol. 29, nr. 2, February 1996, pp. 38-47.

[9] Shelby, Z., Hartke, K., Bormann, C. *Constrained Application Protocol (CoAP).* IETF, 2013, ISSN: 2070-1721.

[10] Hernández-Ramos, J. L., et al. *Distributed Capability-based Access Control for the Internet of Things.* Journal of Internet Services and Information Security (JISIS), volume: 3, number: 3/4, pp. 1-16.

[11] Blaze, M., Feigenbaum J., Ioannidis, J., Keromytis A. D. *The Role of Trust Management in Distributed Systems Security.* In Secure Internet Programming, pp. 185-210, Springer-Verlag, 1999.

[12] Yuan, E., and Tong, J. *Attributed Based Access Control (ABAC) for Web Services.* In Proceedings of ICWS 2005.

[13] Steiner, J. G., Neuman, C. and Schiller, J. I. *Kerberos: An authentication service for open network systems.* USENIX Association, 1988.

[14] De Nicola, R., Loreti, M., Pugliese, R., Tiezzi, F. *A Formal Approach to Autonomic Systems Programming: The SCEL Language*. ACM 2014.

[15] Wuille, P. *BIP32: Hierarchical Deterministic Wallets.* February 2012, https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki.

[16] Castiglione, A., Santis, A. D., Masucci, B. *Key Indistinguishability vs. Strong Key Indistinguishability for Hierarchical Key Assignment Schemes*, IACR Cryptology ePrint Archive, Report 2014/752, 2014