# Poster Abstract: Flexible, Fine Grained Access Control for Internet of Things

Qian Zhou
Stony Brook University
Department of Electrical and Computer Engineering
qian.zhou@stonybrook.edu

Mohammed Elbadry
Stony Brook University
Department of Computer Science
mohammed.salah@stonybrook.edu

Fan Ye
Stony Brook University
Department of Electrical and Computer Engineering
fan.ye@stonybrook.edu

Yuanyuan Yang
Stony Brook University
Department of Electrical and Computer Engineering
yuanyuan.yang@stonybrook.edu

## ABSTRACT

Existing access control strategies for Internet of Things cannot provide good flexibility, robustness or quick response. We propose an access control approach of finer granularity and techniques supporting bulk operation to make interaction in IoT more flexible. Also, a decentralized strategy and data centric network techniques are used to enhance the system robustness and swiftness. We implement our design and conduct evaluation on security and performance.

## CCS CONCEPTS

•**Networks** →**Security protocols;**

## KEYWORDS

Access Control, Security, Internet of Things

**ACM Reference format:**
Qian Zhou, Mohammed Elbadry, Fan Ye, and Yuanyuan Yang. 2017. Poster Abstract: Flexible, Fine Grained Access Control for Internet of Things. In *Proceedings of ACM/IEEE IOTDI, Pittsburgh, PA, USA, April 18-21, 2017 (IoTDI '17),* 2 pages.
DOI: http://dx.doi.org/10.1145/3054977.3057308

## 1 INTRODUCTION

The Internet of Things aims at bringing us a fantastic world in which many more things than traditional computing nodes such as computers and smartphones are connected in the network and operated by humans. However, a solution is currently lacking which allows subjects (i.e., the devices interacting directly with humans, e.g., smartphones) to access to objects (i.e., things in IoT such as door locks and lights) in a flexible, robust and swift way.

A typical existing solution gives inflexible access control by granting a person either full or none authority to an object [2], while in a realistic scenario control of finer granularity is necessary. For example, a cleaner must have authority to open an office door to do his job, but that access should be limited to before, say, 9 AM to stagger his working hours with those of clerks such that the latter will not be disturbed. Also, bulk operation using a single command to operate multiple objects is wanted but not yet supported.

Also, many existing solutions use a centralized strategy [1, 3] to achieve easy security control at the expense of weaker robustness and longer latency. To control an object, a subject sends a command to the cloud first, and the cloud will validate that command and have the object execute it only if the command is legitimate. This approach requires the cloud to be always available to subjects, which is not practical. Additionally, the command's long detour of travelling up to the cloud and then down to the object can bring in latency which is too long to be negligible.

Furthermore, the heterogeneity of off-the-shelves makes it hard to add them into IoT as objects easily. Off-the-shelves like lights and temperature sensors can be heterogeneous in aspects of both communication interfaces and computing performances, thus they may be unable to interact with subjects directly or respond quickly to humans.

We design and implement an approach with good enough security but better flexibility, robustness and speed. We claim our four main contributions as follows: i) An access control approach of finer granularity and techniques supporting bulk operation are proposed to make interaction with objects in IoT more flexible; ii) A decentralized strategy is offered to bring better robustness and faster response speed than existing centralized solutions; iii) Data centric network techniques are used to decouple data from their producers to further enhance the robustness; iv) We have implemented our design and conducted real experiments, and evaluation on both security and performance is performed.

## 2 DESIGN

### 2.1 Bootstrapping

To join in a network, no matter a person or an object first needs to get registered at the administrator of that network. Upon registration is done, a person will have his ID, private key, public key certificate (signed by the administrator) and the administrator's public key stored in his devices (e.g., smartphone). For an object, its profile (also signed by the administrator) will additionally be given and stored which describes the object by telling what it is (category, model version, etc.) and what functions it offers.


*IoTDI '17, Pittsburgh, PA, USA*

DOI: http://dx.doi.org/10.1145/3054977.3057308

## 2.2 Administrator

The administrator is a server on behalf of a human administrator. It stores information of all registered persons and objects, and access control policies according to person attributes and object attributes. A subject, on behalf of its owner, requests his authorities to certain objects, and the administrator responds with a "ticket". Note that the administrator is only needed for updating policies and issuing tickets, and subjects then use tickets to operate objects, with no more participation from the administrator needed.

The authority of a person to an object in our design is finer grained than other works. Instead of just telling if a person can access to an object, we add constraints to describe what the allowed parameter range and operation time range are, and how many times a certain function is allowed to be used.

The format of a request is: $s, o, f, c, timestamp, signature_s$. The symbols respectively denote subject ID, object IDs or attributes, functions, constraints, the moment when the request was generated, and they are together signed by the subject. The format of a ticket is: $tid, s, o, f, c, lifetime, signature_{admin}$. $tid$ is ticket ID and $lifetime$ is the moment when the ticket will expire.

## 2.3 Managers and Objects

Facing the heterogeneity of objects' interfaces and performances, we propose an approach that each low-end object needs to find a high-end one nearby as its manager. A manager has high-end hardware and thus advantage in communication interfaces, computing performances and other resources such as storage and power. A manager takes over work related to asymmetric cryptography from its member objects.

The network made up of managers leverages data centric network techniques to do data discovery and command routing, and it is more robust than one using IP-based solutions.

## 2.4 Command

To operate an object, a subject needs to send a command, of which the format is: $ticket, ao, af, ap, count, signature_s$. In $ticket$ part, either the whole ticket is transmitted directly or only $tid$ is included as reference. *ao* is the actual object that the person wants to operate, $af$ and *ap* stand for actual function and actual parameter, namely the function to call and the parameter to feed in. *count* is the current value of a monotonic increasing counter and is included for anti-replay. All these are signed by the subject.

A manager that receives the command will check its legitimacy and route it to the destination if it is legitimate. Finally the command arrives at the manager of the target object. Besides the checks done by other managers along the routing path, this manager can additionally check the freshness of the command because it keeps a counter for each of its member object, and increments the counter value of an member object each time a command targeting that object is accepted.

If all checks pass, the manager will have its object execute the command by sending to it: $af, ap, count, HMAC$. *count* here is the value of another counter kept by the manager and its member object for anti-replay. And this time a symmetric signature (i.e., *HMAC*) is used to protect integrity because an asymmetric one is too heavy for a low-end object to generate or verify.

# 3 SECURITY ANALYSIS

We analyze the security of our design by presenting its resistance to different attacks. We classify a possible attack based on the malicious node's source, role and the security property it attempts to attack. About its source, a malicious node can be brought in IoT from external, or an internal one which gets compromised. About the role, a malicious node can choose to behave as a subject, a manager, or an object. And the possible security properties to harm include integrity, freshness and availability. Note that confidentiality is not covered in our this work because currently we aim at offering a solution for places like home, campus and company, where it is okay if others know what operations we do.

If an external node behaves as a subject and tries to send commands to control objects it is not authorized to, namely harming the integrity, because of lack of registration at the administrator and thus no validate public key certificate or private key, none of its commands will have a validate signature, and will thus be detected as invalid and discarded by the first benign manager along the routing path towards the destination. Second, if it tries to harm the freshness by replaying validate commands, because the target manager keeps a monotonic increasing counter and increments its value each time a validate command is accepted, the replayed one will be detected for its obsolete counter value and rejected. Third, it can also keep sending commands with wrong signatures to harm the availability, but benign managers will not help forward those commands after finding them wrong, thus only the one-hop neighbors will be kept busy.

If an external node behaves as a manager, it can first try to cajole objects into choosing it as their manager. But this is not going to work because it lacks a valid public key certificate and a private key, and once an object finds this out, it will abort its symmetric key establishment process with this malicious manager. Second, it can also send commands to objects around, but without knowing a correct symmetric key, a wrong HMAC will be generated, which makes objects ignore its commands. Third, it can replay commands, but similarly, an object also keeps a monotonic increasing counter and will discard any command containing an obsolete counter value. As for DoS attack, a malicious manager can keep sending messages to one-hop objects to make them busy.

An external node can also behave as an object. All it can do is keep sending messages to managers around and pretend to want to choose them as its managers, making them busy.

If a malicious node comes from compromising a benign one and is able to authenticate itself as a valid node, which has small probabilities to happen, then it can do more harm targeting integrity than an external one. First, if it behaves as a subject, the commands it gives will be accepted. Facing this, using tickets of smaller sizes and shorter lifetime can decrease the damage to some degree. Second, if a manager is compromised, then all of its member objects will be indirectly compromised and act as the attacker wants. Third, if an object is compromised, the harm will be limited to itself only.

# REFERENCES

[1] Amazon. 2017. AWS IoT. https://aws.amazon.com/iot/. (2017).
[2] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. 2013. A capability-based security approach to manage access control in the internet of things. *Mathematical and Computer Modelling* 58, 5 (2013), 1189–1205.
[3] Samsung. 2017. SmartThings. https://www.smartthings.com/. (2017).