black hat ASIA 2021
May 6-7, 2021
BRIEFINGS

# How I Can Unlock Your Smart Door: Security Pitfalls in Cross-Vendor IoT Access Control

Bin Yuan & Yan Jia

black hat ASIA 2021

29TH USENIX SECURITY SYMPOSIUM

# Shattered Chain of Trust: Understanding Security Risks in Cross-Cloud IoT Access Delegation

Bin Yuan, Yan Jia, Luyi Xing,

Dongfang Zhao, Xiaofeng Wang, Deqing Zou, Hai Jin, Yuqing Zhang

華中科技大學 Huazhong University of Science and Technology

西安電子科技大學 XIDIAN UNIVERSITY

INDIANA UNIVERSITY BLOOMINGTON

中国科学院大学 University of Chinese Academy of Sciences

# Smart home vendors

## Smart home device vendors

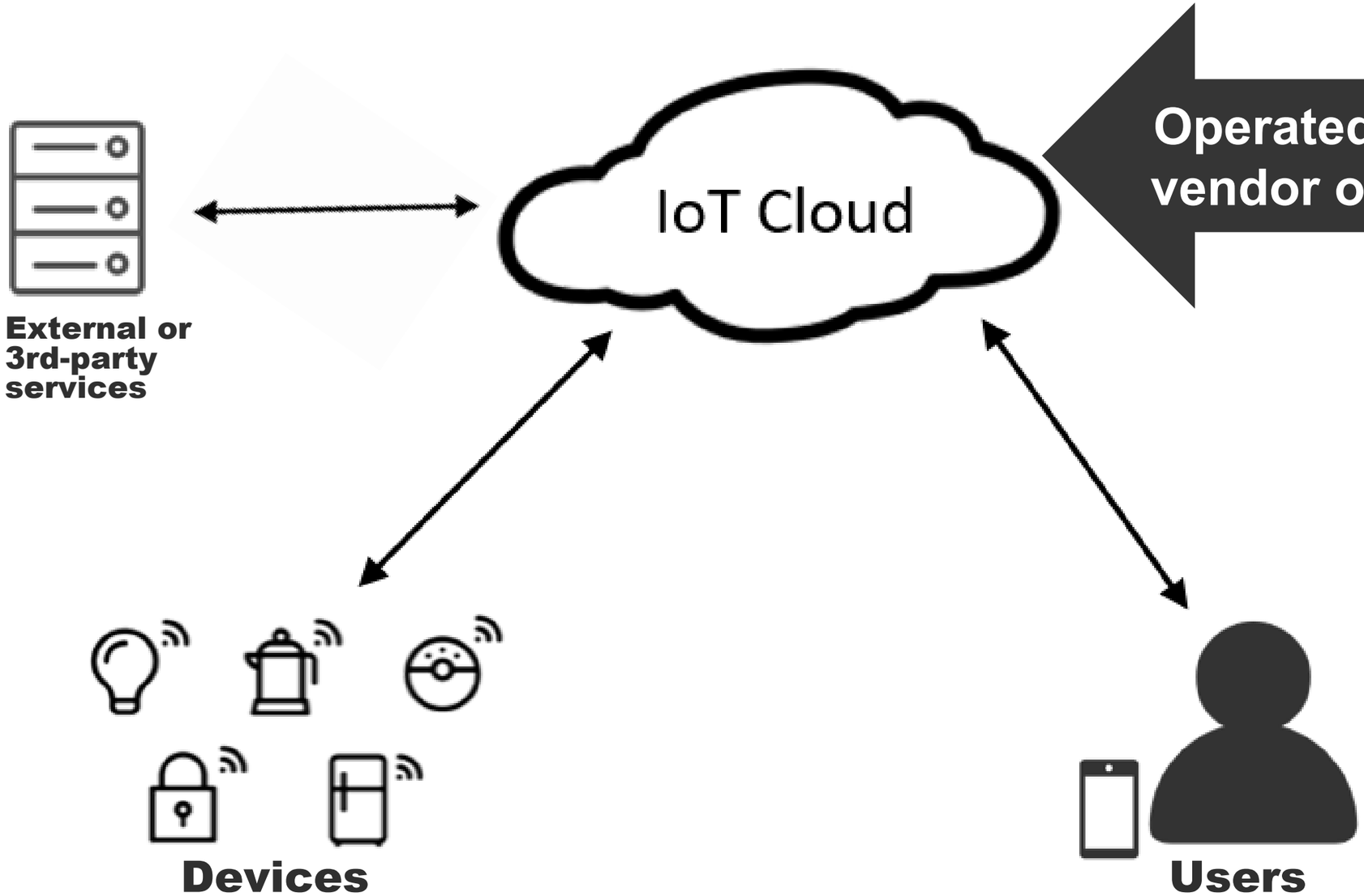





## Smart home cloud service providers



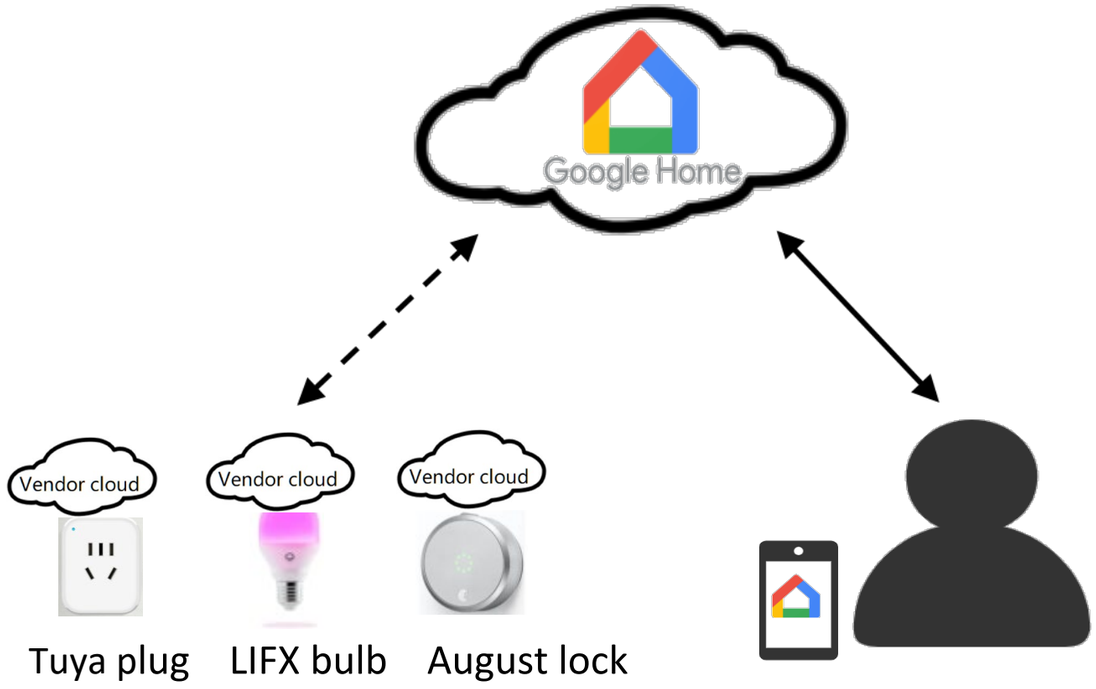

# The cloud-based IoT device management




Operated by either an IoT device vendor or cloud service provider

# Complicated functionalities supported by IoT clouds

- Cross-vendor/cross-cloud device control
  - Manage different vendor devices through the same console
- Sharing of device access
  - Share the access to the lock to an Airbnb guest (temporarily)
- Device control automation
  - Turn on the light when motion detected

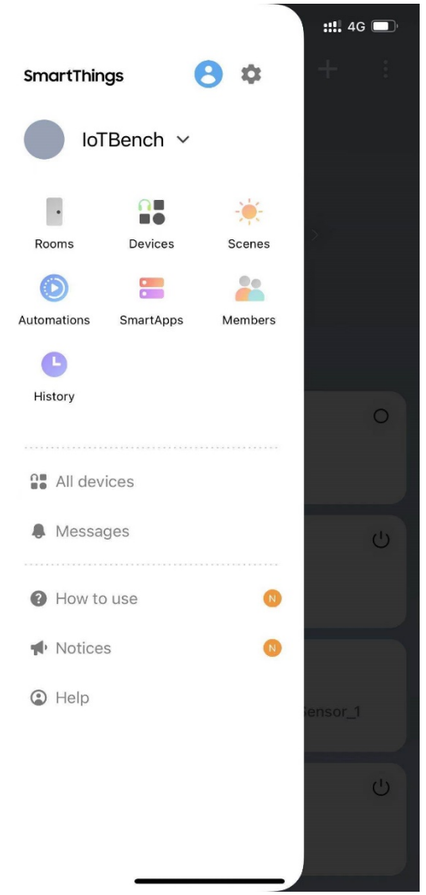



A user uses Google Home to control all her devices from different vendors (e.g., Tuya plug, LIFX bulb, August lock)

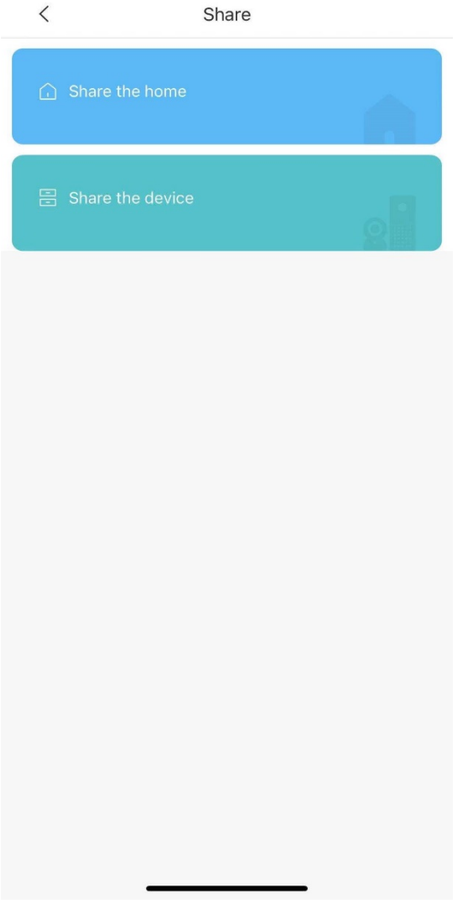
# Complicated functionalities supported by IoT clouds

- Cross-vendor/cross-cloud device control
  - Manage different vendor devices through the same console
- Sharing of device access
  - Share the access to the lock to an Airbnb guest (temporarily)
- Device control automation
  - Turn on the light when motion detected

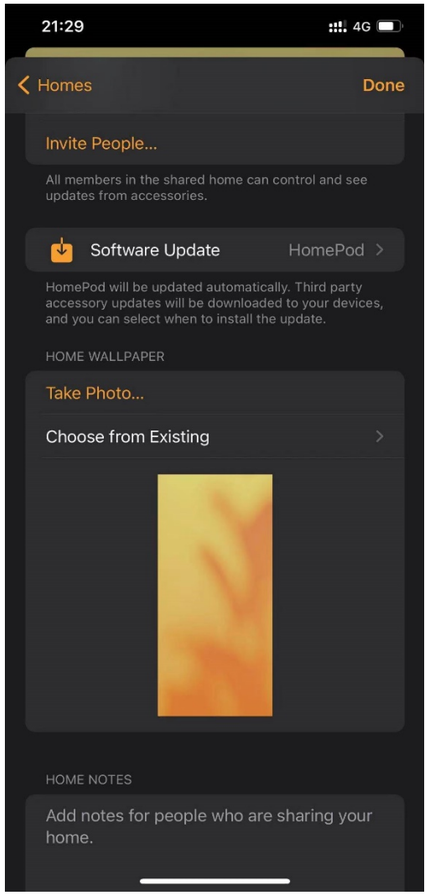
Member management in SmartThings




Share a home or device in Mi Home

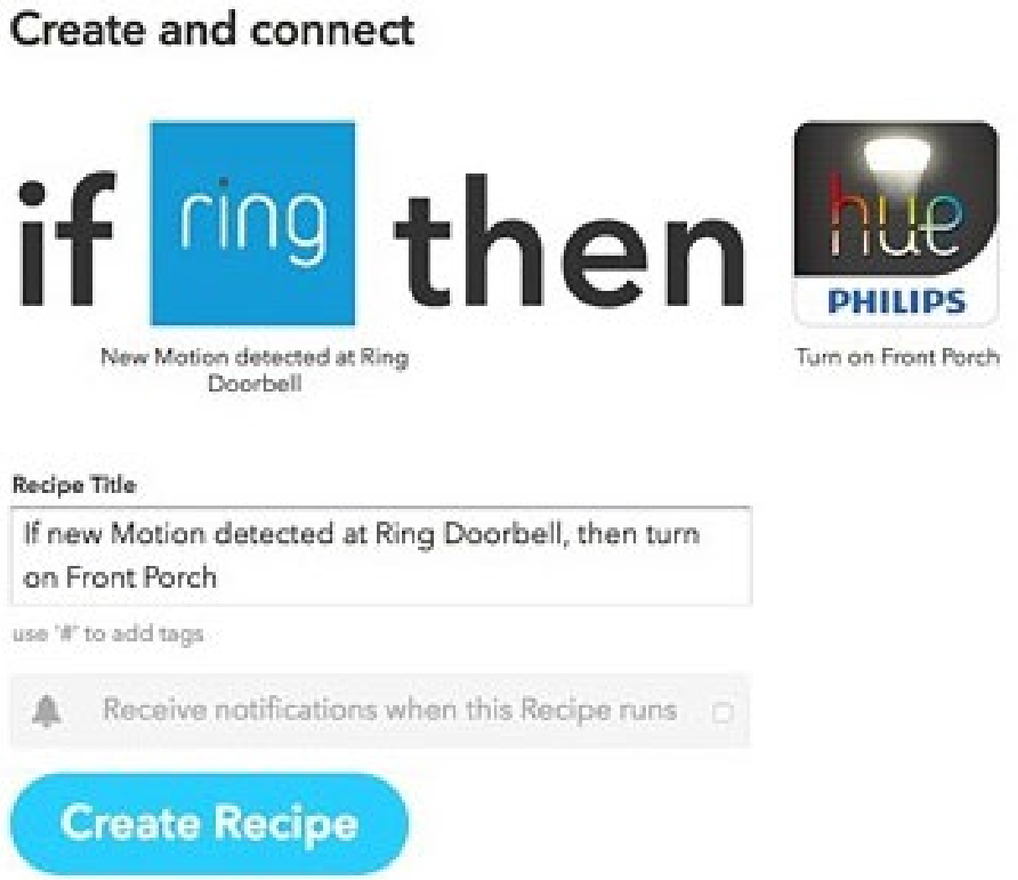



Invite people in HomeKit



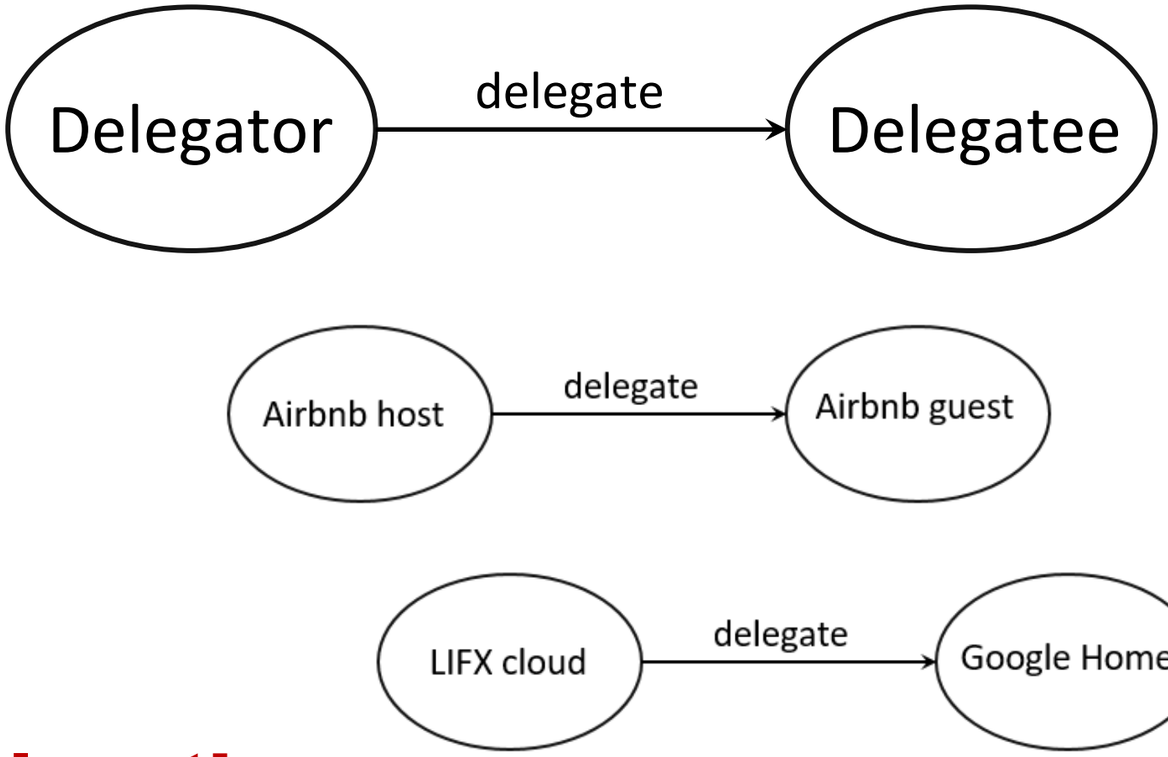
# Complicated functionalities supported by IoT clouds

- Cross-vendor/cross-cloud device control
  - Manage different vendor devices through the same console
- Sharing of device access
  - Share the access to the lock to an Airbnb guest (temporarily)
- Device control automation
  - Turn on the light when motion detected



# Complicated functionalities supported by IoT clouds

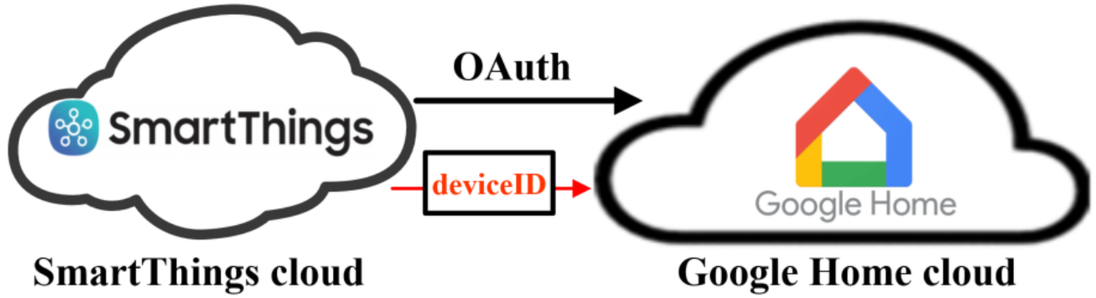
- Cross-vendor/cross-cloud device control
  - Manage different vendor devices through the same console
- Sharing of device access
  - Share the access to the lock to an Airbnb guest (temporarily)
- Device control automation
  - Turn on the light when motion detected



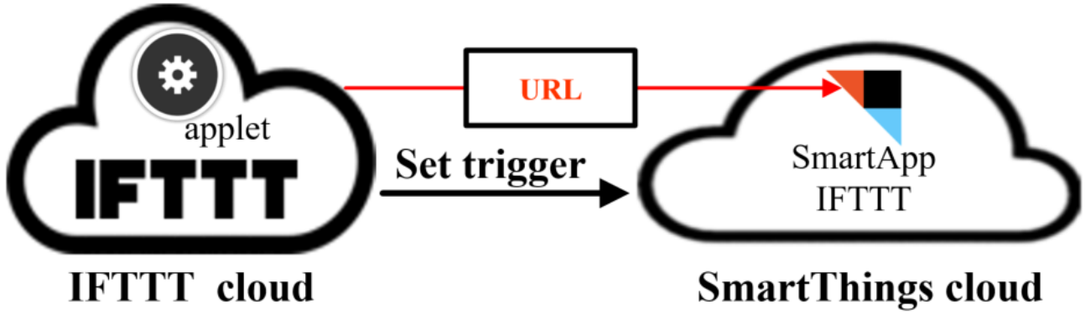

**It's all about device access delegation**

# The two most common cross-cloud delegation mechanisms

- OAuth and its customization
  - Actions on Google: OAuth + asking for additional information (e.g., deviceID, device name) from the delegator cloud

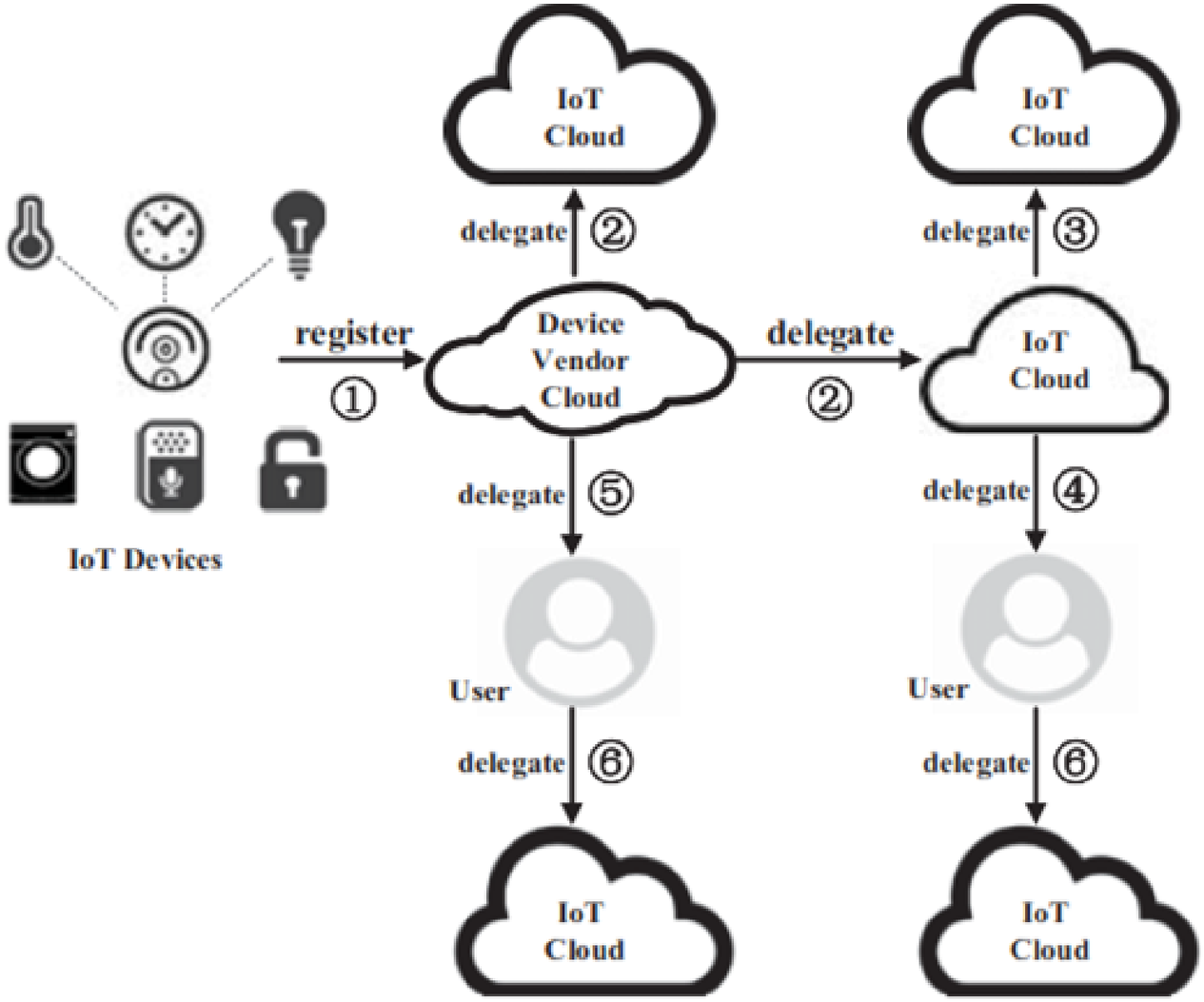



- Home-grown authorization
  - Set trigger event in IFTTT: issuing a URL which receives trigger events from the delegatee cloud



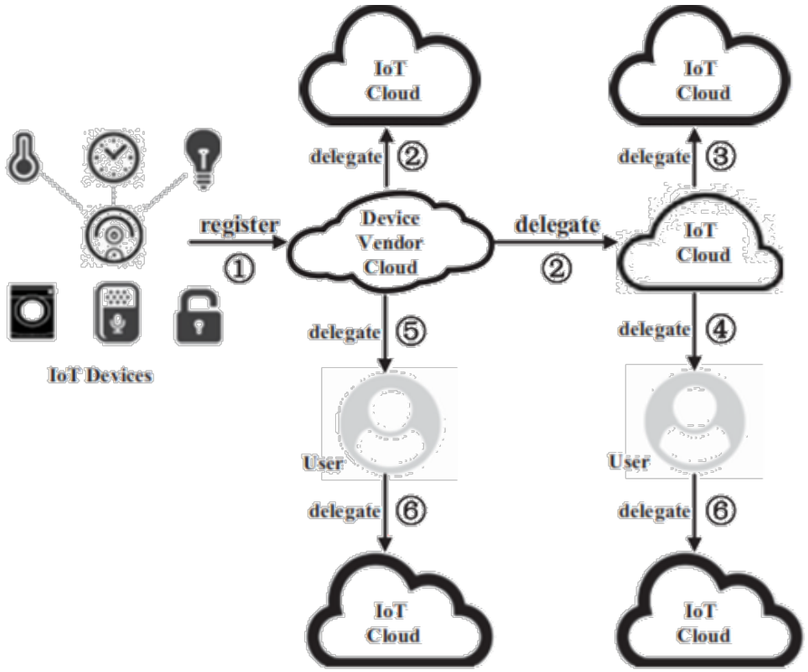
# Convoluted delegation chain in IoT

- IoT device access right delegation
  - first given to its device vendor cloud (①)
  - delegated to a delegatee cloud (②)
  - further handed over to another (delegatee) cloud (③)
  - granted (by the device administrator) to other (delegatee) users (④ ⑤)
  - the (delegatee) user may further give her access to another (delegatee) cloud (⑥)



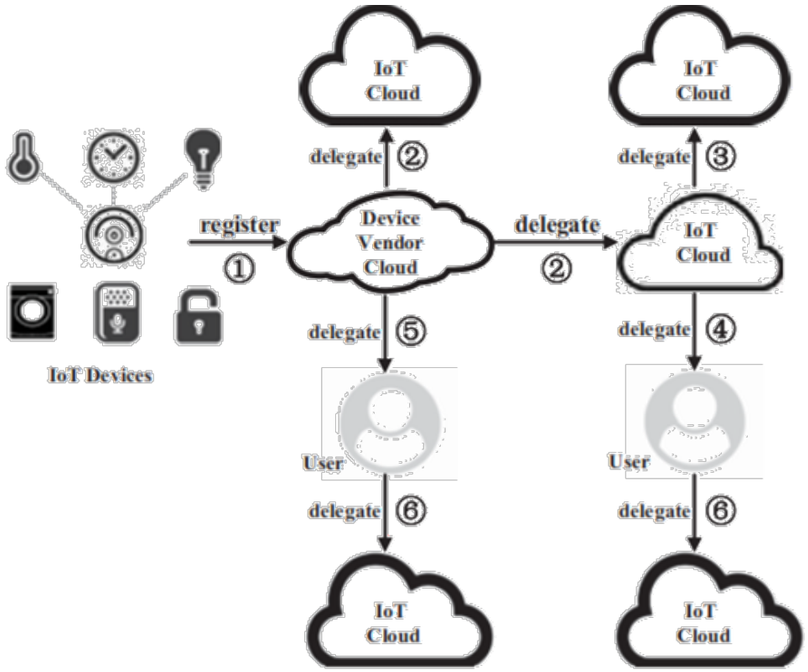
# Lack of understanding on the security of cross-cloud IoT delegation

- Risks in cross-cloud IoT delegation
  - Theoretic models analyzed before
    - all parties run the same delegation protocol and interact through unified interfaces
  - Delegation in today's real-world IoT clouds
    - individual, heterogeneous delegation protocols
    - incompatible with other clouds
    - not being properly verified



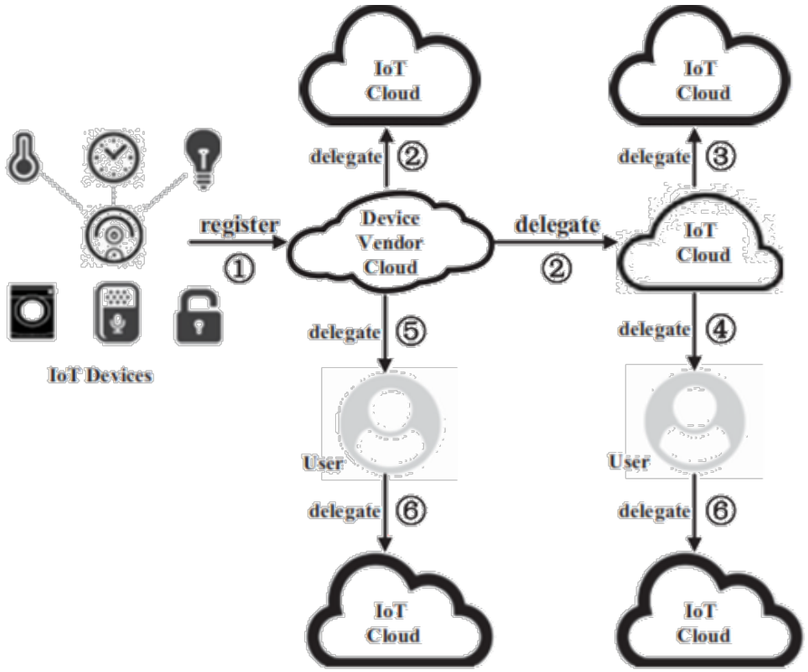
# Threat model

- Delegatee user can be malicious, while the administrator, cloud, and device are benign
- Goal of the adversary is to get unauthorized access to IoT devices
- The adversary would make full use of his power to acquire useful information, e.g., make API calls, extract information from system logs, official documentations and capture network traffic generated by/for his mobile app



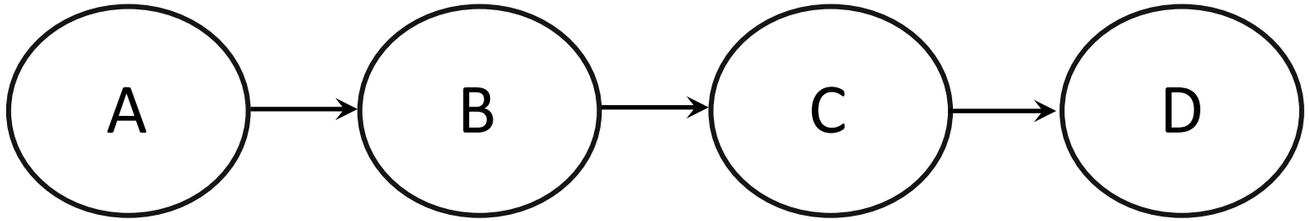
# Security requirements

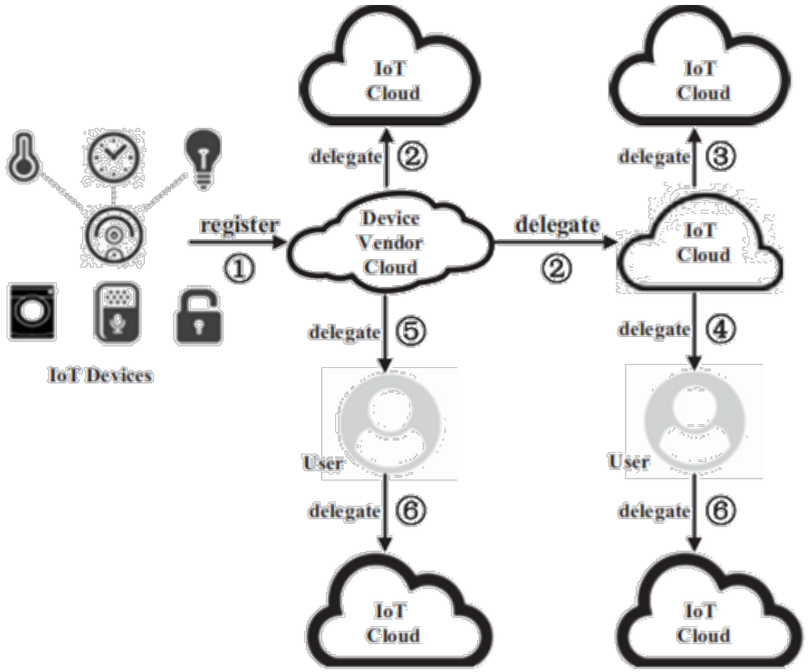
- Safe and consistent delegation policies



# Security requirements

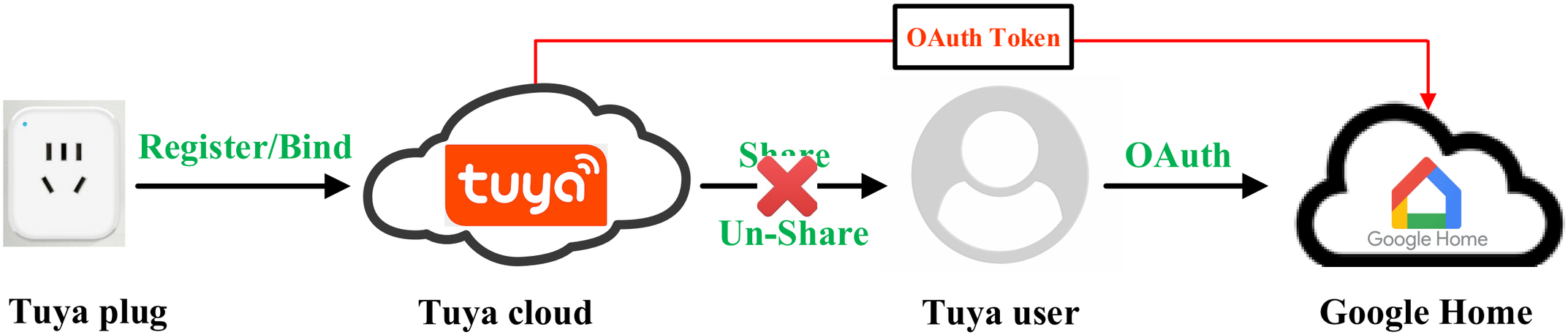
- Safe and consistent delegation policies
- Non-bypassable and transitive delegation control



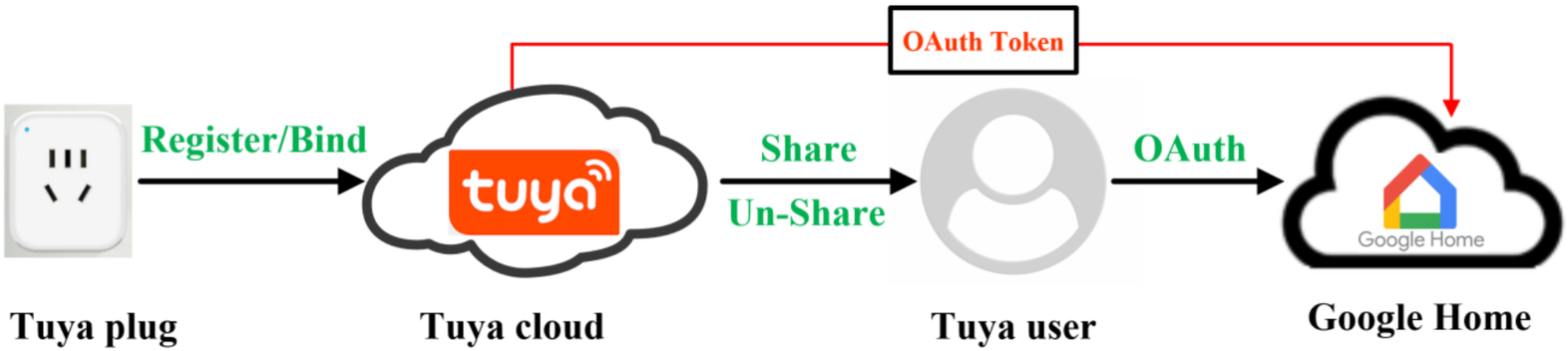
# Vulnerable Cross-Cloud IoT Delegation: a motivating example

- Violation of “transitive delegation control” in Tuya



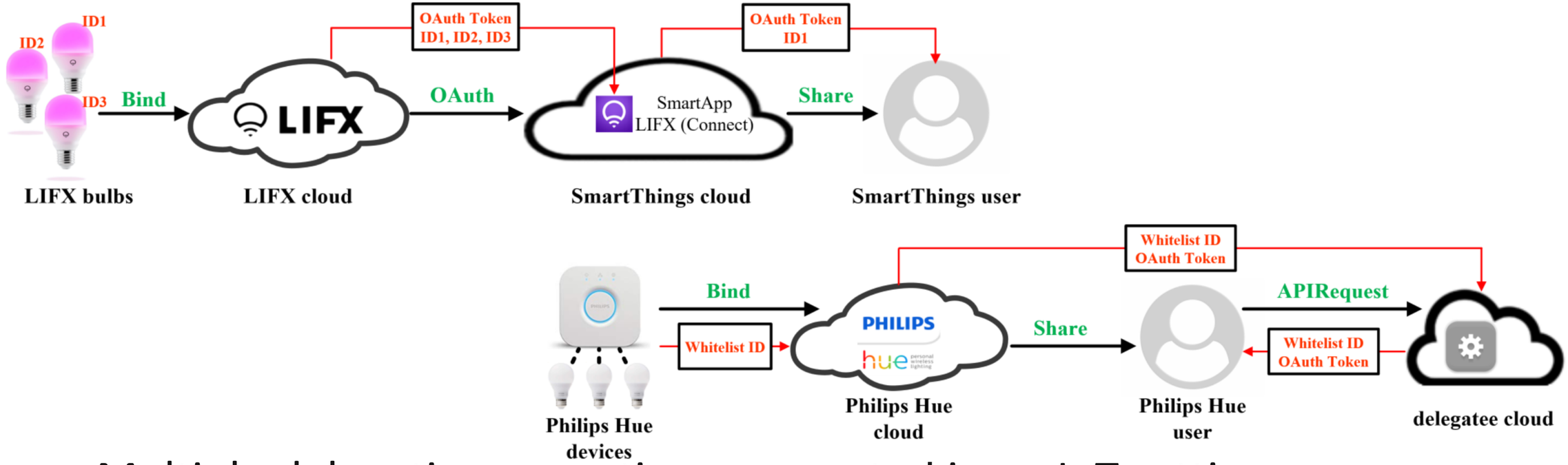

  - Google Home still holds a valid OAuth token ( issued by the Tuya cloud during the OAuth operation), allowing Google Home to access the Tuya plug even after the un-share operation
  - Attack: leveraging a third-party (e.g., Google Home) to make temporary access right permanent

# Observations from the Tuya Case



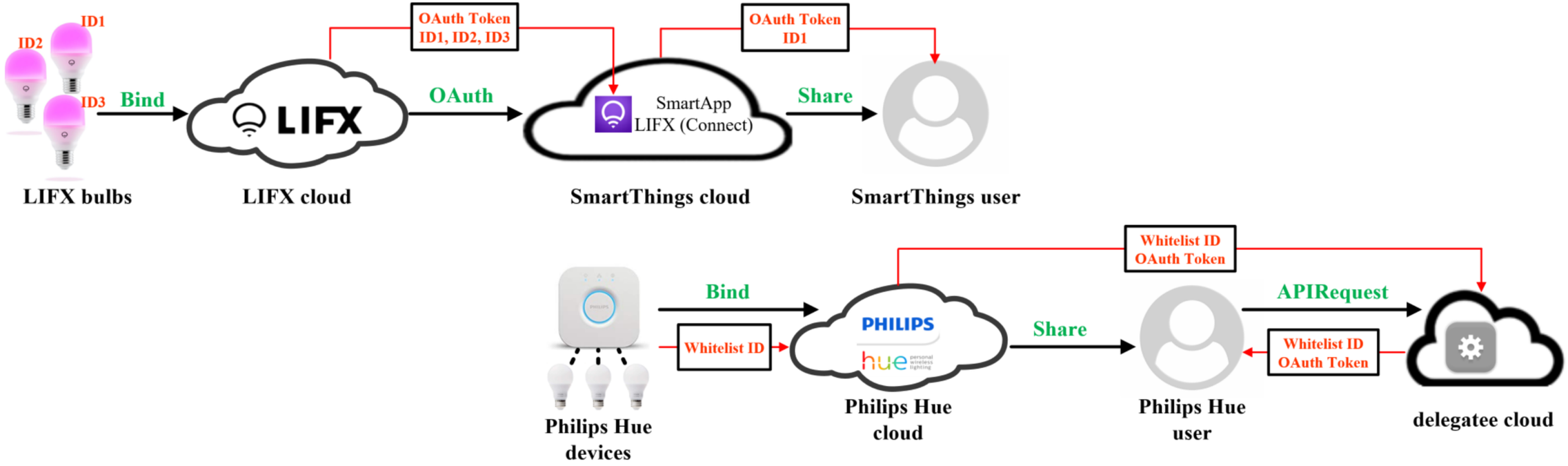

- Multiple delegation operations supported in an IoT setting
- Data flow (e.g., token issuing and distributing) along with operations
- Multi-step access path (with a valid token) to a device

# Same observations/patterns in other IoT settings




- Multiple delegation operations supported in an IoT setting
- Data flow (e.g., token issuing and distributing) along with operations
- Multi-step access path (with a valid token) to a device

# Towards automatic vulnerability discovery




Common delegation pattern identified in different settings

**Formal verification based IoT cross-cloud delegation vulnerability discovery**

# Formal verification based IoT cross-cloud delegation vulnerability discovery

- Security property
  - unauthorized delegatee user should not have a path to the IoT devices which he is not entitled to access

# Formal verification based IoT cross-cloud delegation vulnerability discovery

- Security property
  - unauthorized delegatee user should not have a path to the IoT devices which he is not entitled to access

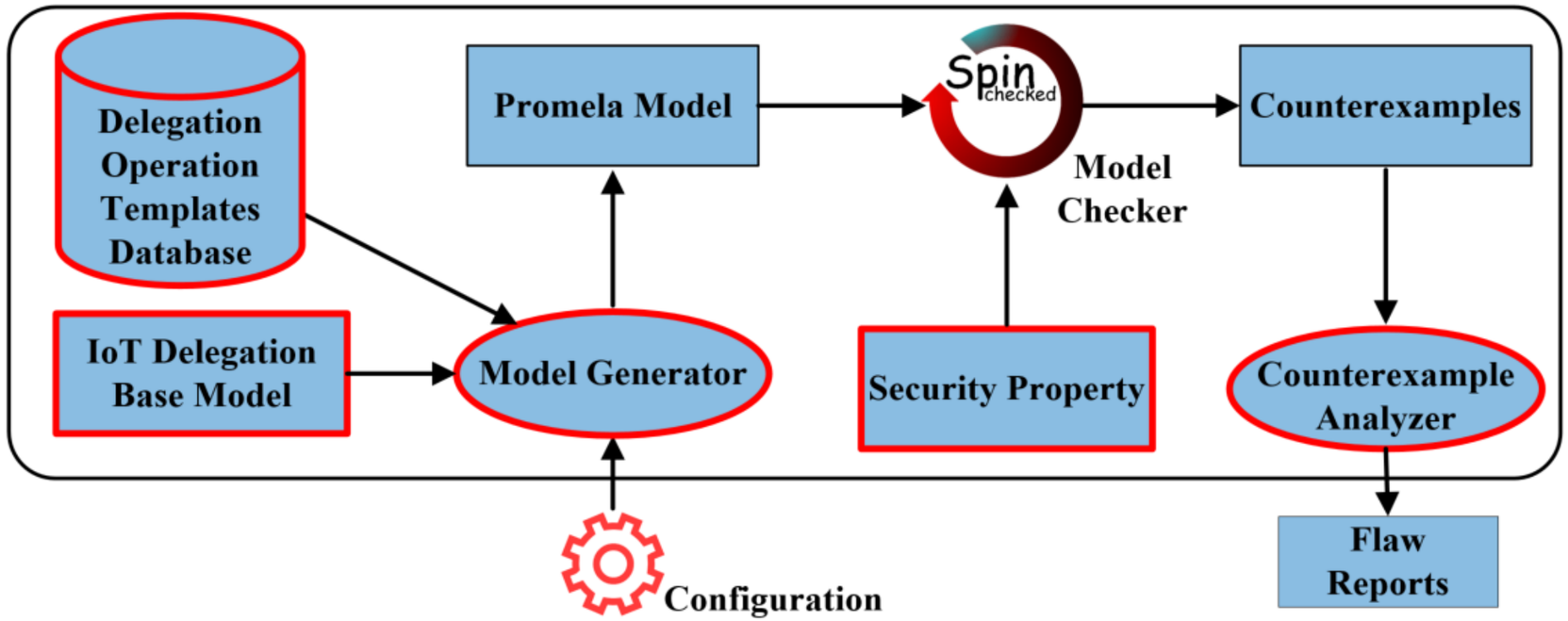
- IoT delegation modeled as a transition system $\mathcal{M} = (\mathcal{A}, \mathcal{S}, O, \mathcal{T}, s_0)$
  - $\mathcal{A}$ is the set of **actor**s (e.g., device, cloud, user)
  - $O$ is the set of **operation**s (e.g., OAuth, share, un-share, bind, unbind, APIRequest, etc.)
  - $\mathcal{S}$ is the set of **state**s, where $s_0$ is the **initial state** (where no delegation happens)
    - Tokens received and issued during delegation and the access control mapping between these tokens
  - $\mathcal{T}: \mathcal{S} \times O \rightarrow \mathcal{S}$ is a function that drives the **transition** from one state to the next

# Formal verification based IoT cross-cloud delegation vulnerability discovery

- Security property
  - unauthorized delegatee user should not have a path to the IoT devices which he is not entitled to access
- IoT delegation modeled as a transition system $\mathcal{M} = (\mathcal{A}, \mathcal{S}, O, \mathcal{T}, s_0)$
  - $\mathcal{A}$ is the set of **actor**s (e.g., device, cloud, user)
  - $O$ is the set of **operation**s (e.g., OAuth, share, un-share, bind, unbind, APIRequest, etc.)
  - $\mathcal{S}$ is the set of **state**s, where $s_0$ is the **initial state** (where no delegation happens)
    - Tokens received and issued during delegation and the access control mapping between these tokens
  - $\mathcal{T}: \mathcal{S} \times O \rightarrow \mathcal{S}$ is a function that drives the **transition** from one state to the next
- Detecting flaws
  - leveraging a model checker to verify whether pre-defined security properties hold in the model

# VerioT: the first (semi-automatic) verification tool for IoT cross-cloud delegation vulnerability discovery

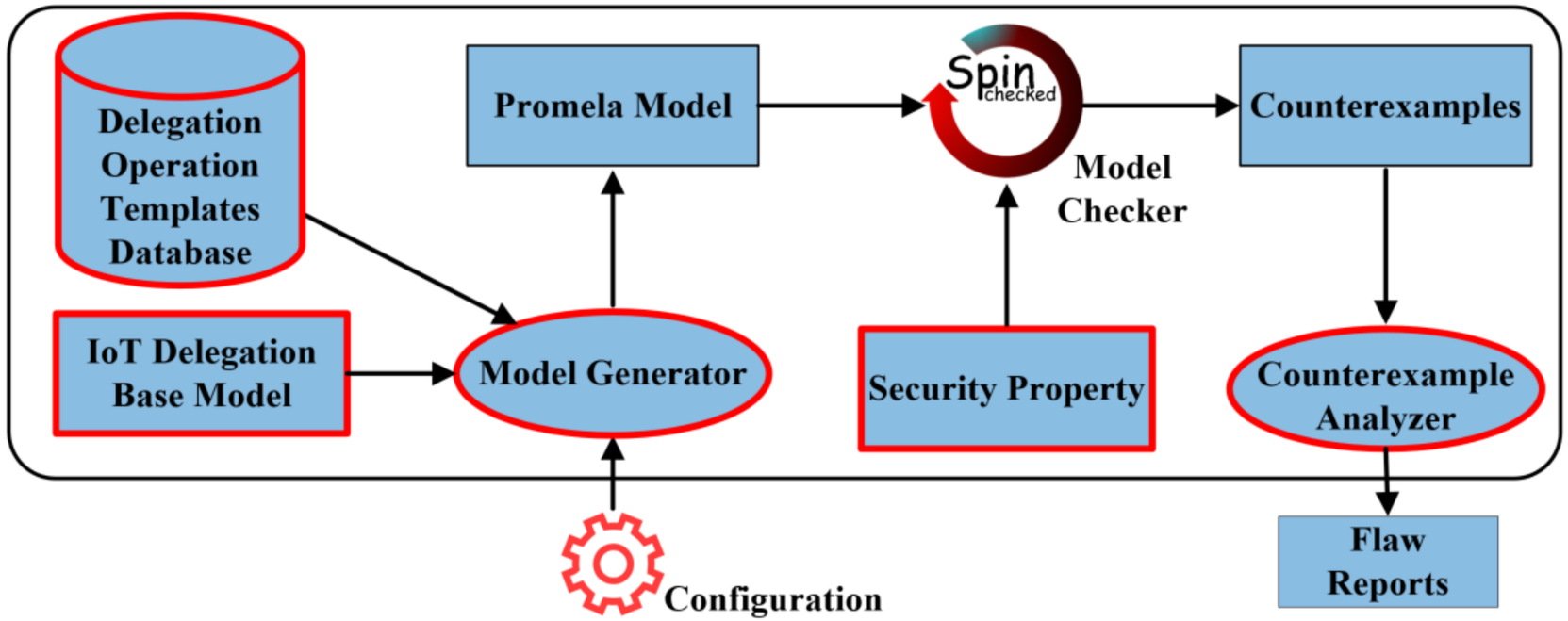
- Modeling different real-world IoT system
  - **Refinement**: operation template, base model, and configuration




The architecture of VerioT

# VerioT: the first (semi-automatic) verification tool for IoT cross-cloud delegation vulnerability discovery

- Modeling different real-world IoT system
  - **Refinement**: operation template, base model, and configuration

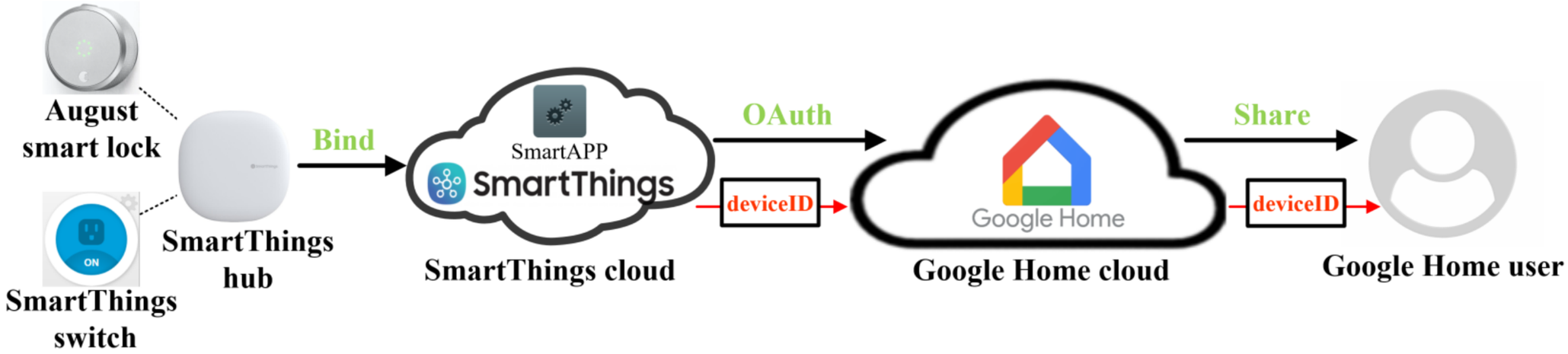



The architecture of VerioT

VerioT is made publicly available:
https://github.com/VerioT/VerioT

## Vulnerability Category 1: Conflicting Security Policies Across IoT Vendors/Clouds
## Vulnerability 1: Google Home leaked device ID of Samsung SmartThings cloud

- Different security assumptions on device ID
  - SmartThings uses device ID as an authentication token on the trigger-action management
  - Google Home discloses the device ID to any authorized user

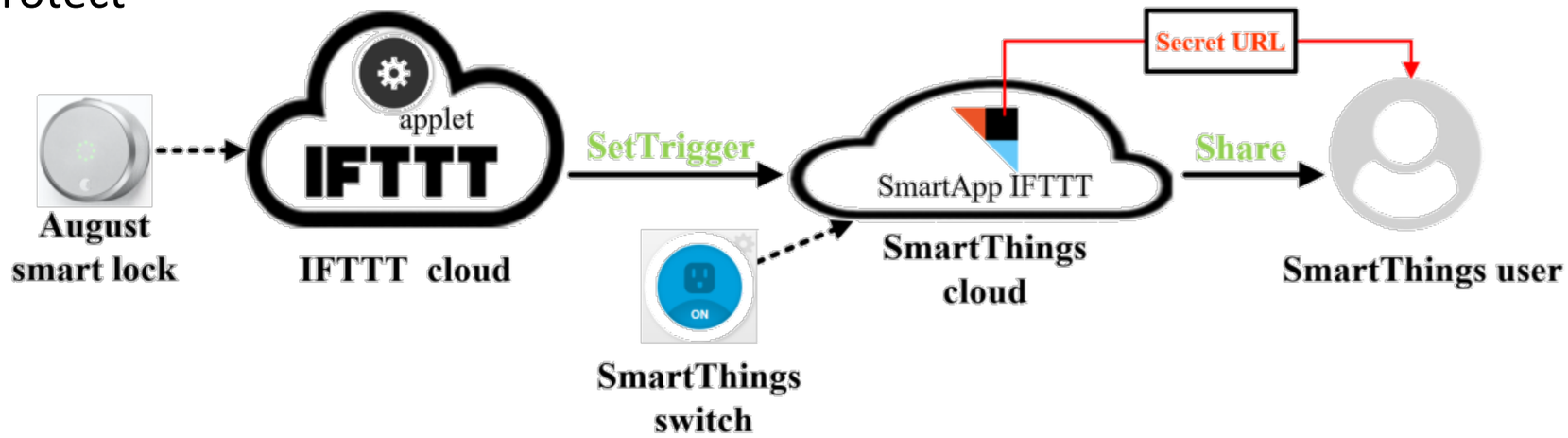



- Malicious delegatee user (e.g., an Airbnb guest) can use the device ID to spoof events to trigger SmartThings to open the lock even after he checks out

# Vulnerability Category 1: Conflicting Security Policies Across IoT Vendors/Clouds
# Vulnerability 1: Google Home leaked device ID of Samsung SmartThings cloud

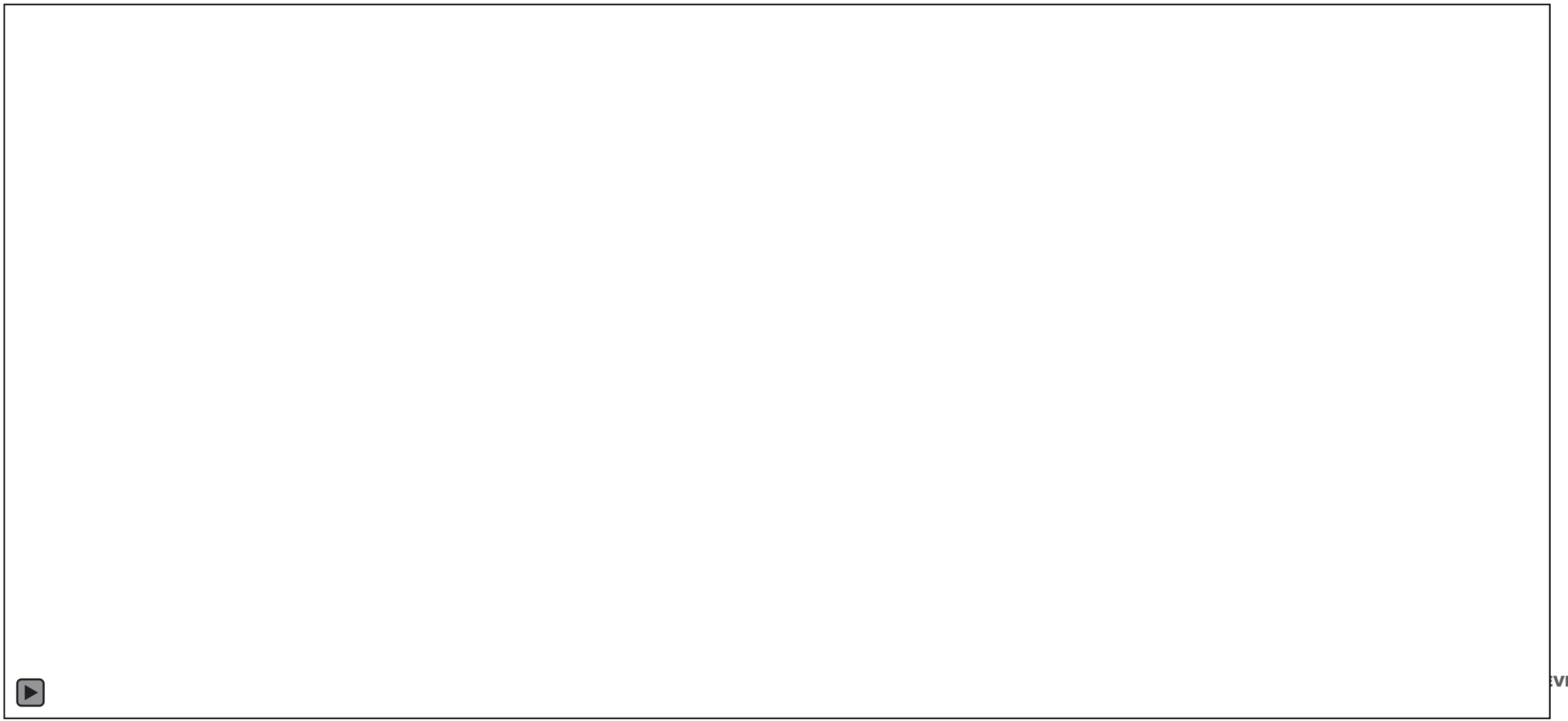
## Vulnerability Category 1: Conflicting Security Policies Across IoT Vendors/Clouds
## Vulnerability 2: IFTTT leaked the secret token of SmartThings cloud

- Mismatched security policy on secret URL management
  - IFTTT cloud (delegator) leaks the secret URL that SmartThings cloud (delegatee) wants to protect



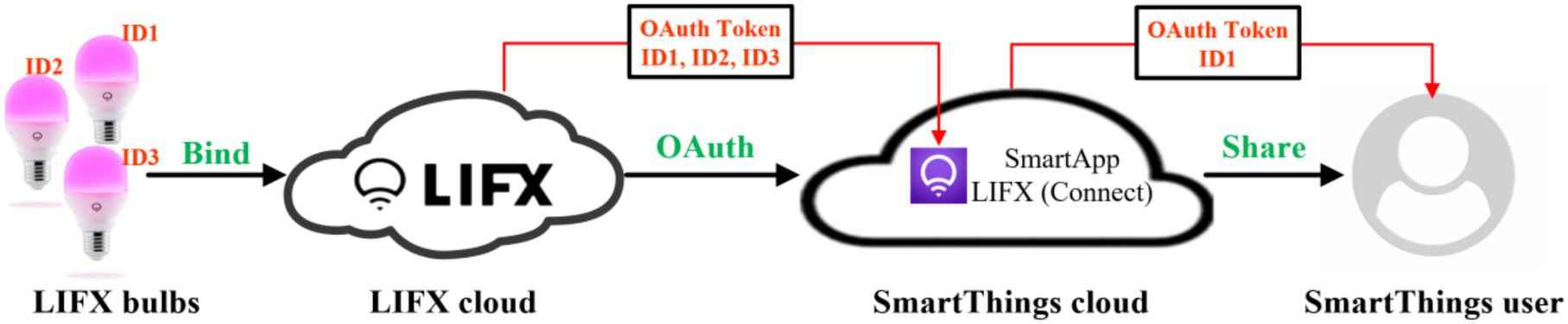

- Malicious delegatee user can post HTTP requests to the URL to trigger the action of the applet in the IFTTT cloud, thus to operate the devices behind the IFTTT cloud even after his access right is revoked by SmartThings cloud

# Vulnerability Category 1: Conflicting Security Policies Across IoT Vendors/Clouds
# Vulnerability 2: IFTTT leaked the secret token of SmartThings cloud



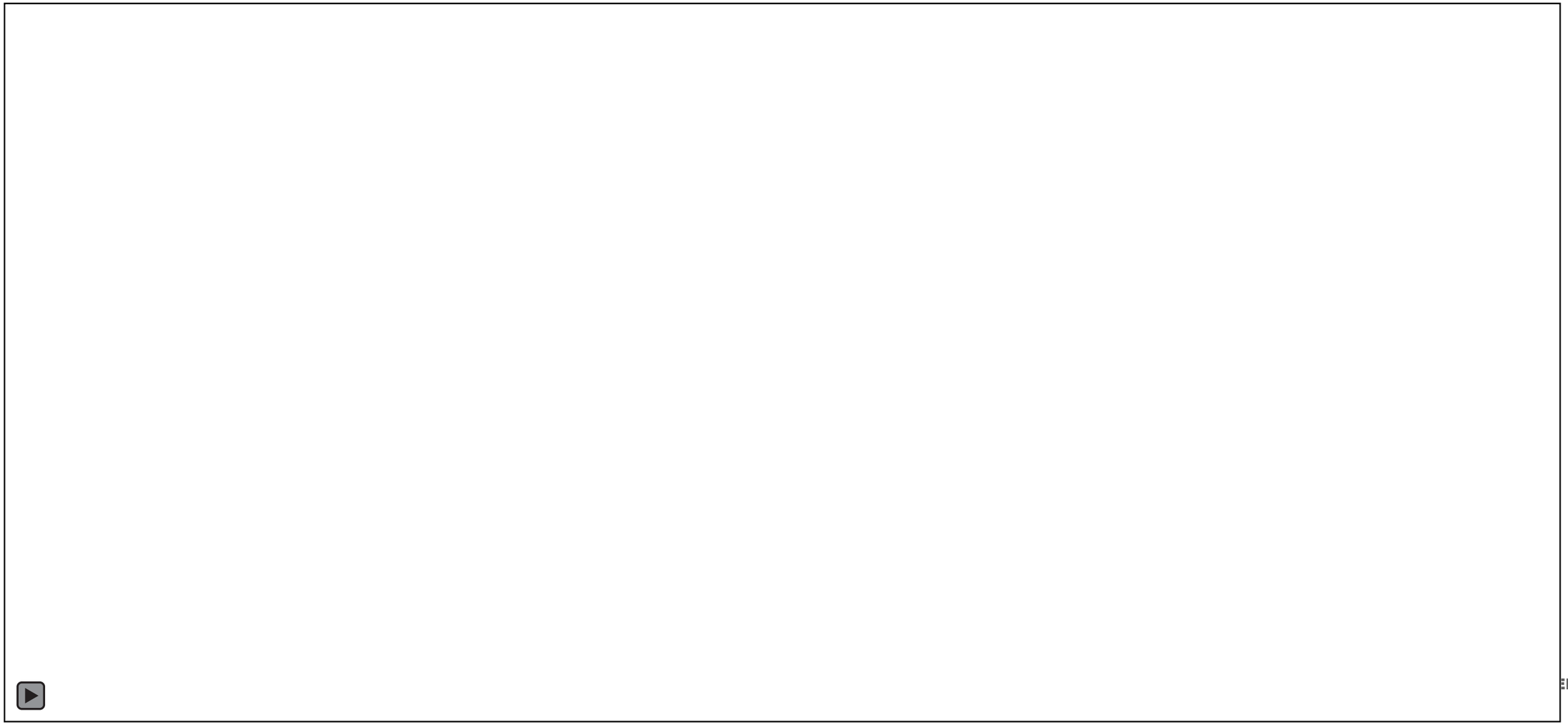
## Vulnerability Category 2: Pitfalls in Security Policy Enforcement Across IoT Vendors/Clouds
## Vulnerability 3: SmartThings cloud exposed hidden devices of LIFX cloud

- OAuth token is made accessible to users on the delegatee cloud
  - Shared users can read the OAuth token issued by LIFX cloud from the storage of SmartThings cloud, bypassing the control of device hiding.



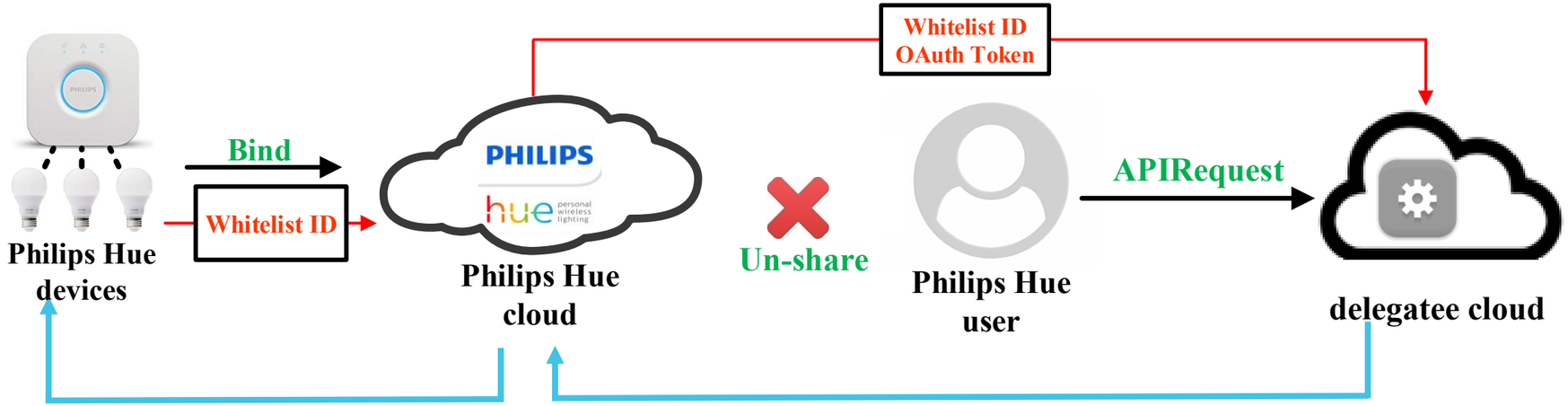

- Malicious delegatee user can use the the OAuth token to gain the ID of the devices hidden from him and control these hidden devices

# Vulnerability Category 2: Pitfalls in Complete Security Policy Enforcement Across IoT Vendors/Clouds
# Vulnerability 3: SmartThings cloud exposed hidden devices of LIFX cloud



## Vulnerability Category 2: Pitfalls in Security Policy Enforcement Across IoT Vendors/Clouds
## Vulnerability 4: Abusing cross-vendor delegation API of Philips Hue

- Incomplete revocation scheme in the delegator cloud
  - Philips Hue cloud only invalidates the token which is used for access check in the device, not the token that is used for authentication in the cloud.




- Malicious delegatee user can abuse the API to gain new OAuth token and Whitelist ID, allowing him to control the Philips Hue devices after the administrator revokes his access right.

# Measurement

- Prevalence of vulnerable IoT delegation
  - All the 10 mainstream IoT clouds being studied are affected
    - Device vendor clouds: Philips Hue, August, LIFX , MiHome, and iHome
    - Delegatee clouds: Google Home, IFTTT, SmartThings, Amazon Alexa, and Wink

- Scope of impact
  - Google Home disclosing device ID affects at least 3 IoT clouds
  - Leakage of IFTTT's Secret URL affects 34 IoT clouds
  - OAuth token disclosure problem exists in at least 18 SmartApps in SmartThings
  - Tuya's problematic management on OAuth token can affect up to 58 IoT manufacturers

# Conclusion

- Root cause
  - Heterogeneous and ad-hoc delegation process (because of the absence of a standardized cross-cloud delegation protocol)
- Lessons learnt
  - The caution one should take when applying a custom cross-cloud authorization scheme to today's already complicated IoT delegation
    - the delegator and the delegatee violate each other's security policies
    - problematic security policy enforcement due to lack of rigorous verification
- New design principles
  - Decoupling the delegatee and the delegator clouds
  - Communicating security assumptions and constraints
  - Verifying delegation design whenever possible

# Thanks!

Bin Yuan, Postdoc, Huazhong University of Science and Technology
Yan Jia, Postdoctoral Research Associate, Nankai University
*Work was done when Yan Jia was a PhD student at Xidian University.