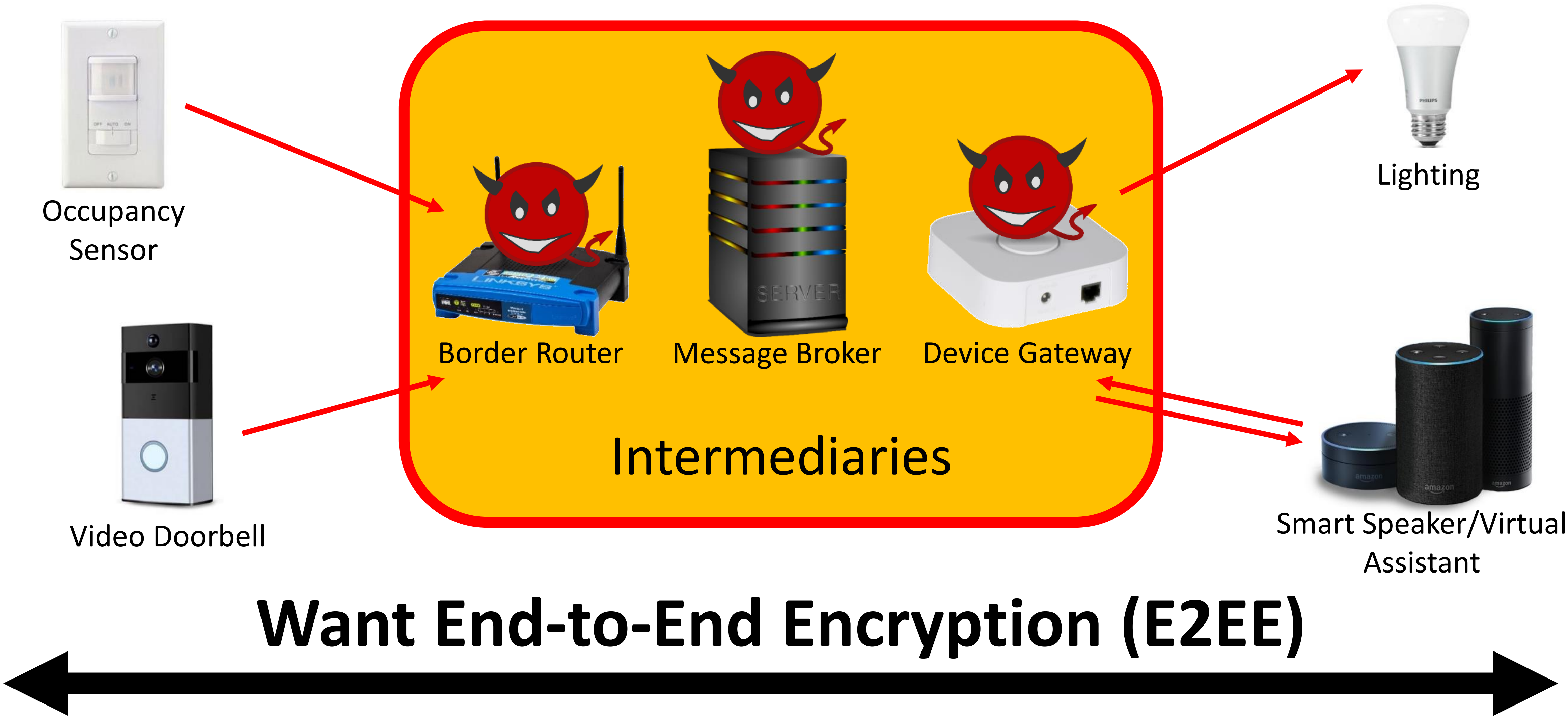
JEDI: Many-to-Many End-to-End Encryption and Key Delegation for IoT

Sam Kumar, Yuncong Hu, Michael P Andersen, Raluca Ada Popa, David E. Culler

University of California, Berkeley

IoT Devices Collect Privacy-Sensitive Data

IoT Devices Collect Privacy-Sensitive Data

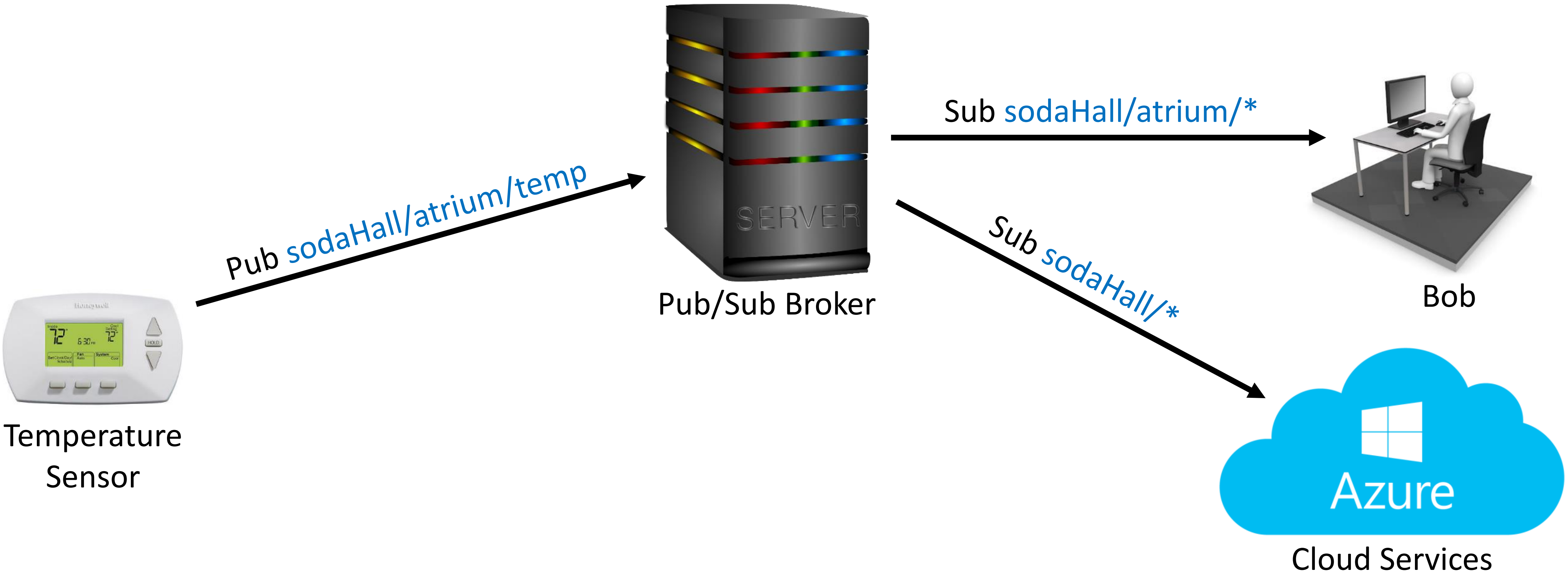


Want End-to-End Encryption (E2EE)

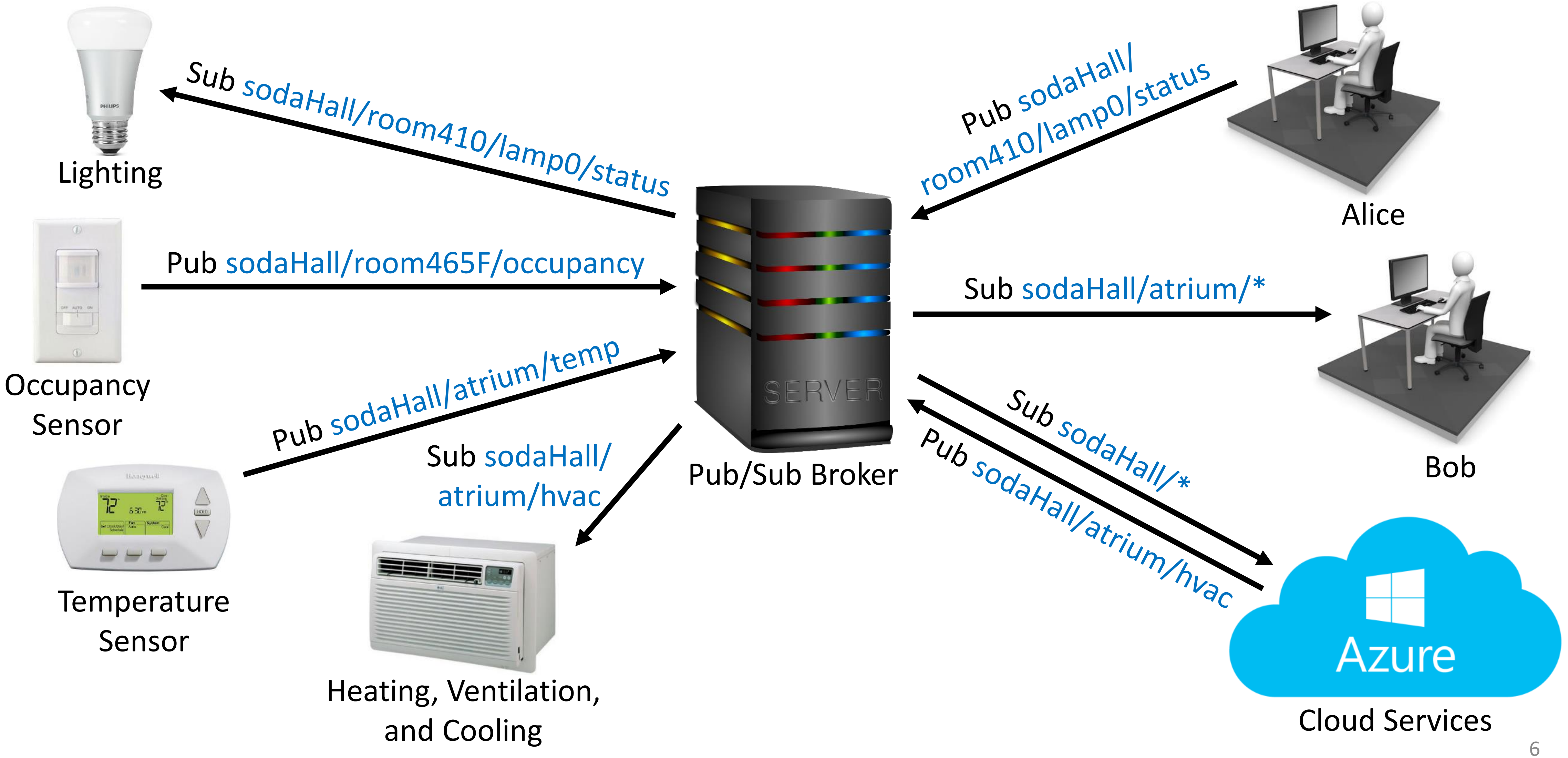
Existing E2EE is a Poor Fit for Large-Scale IoT

- Large-scale IoT systems use the **publish/subscribe** paradigm

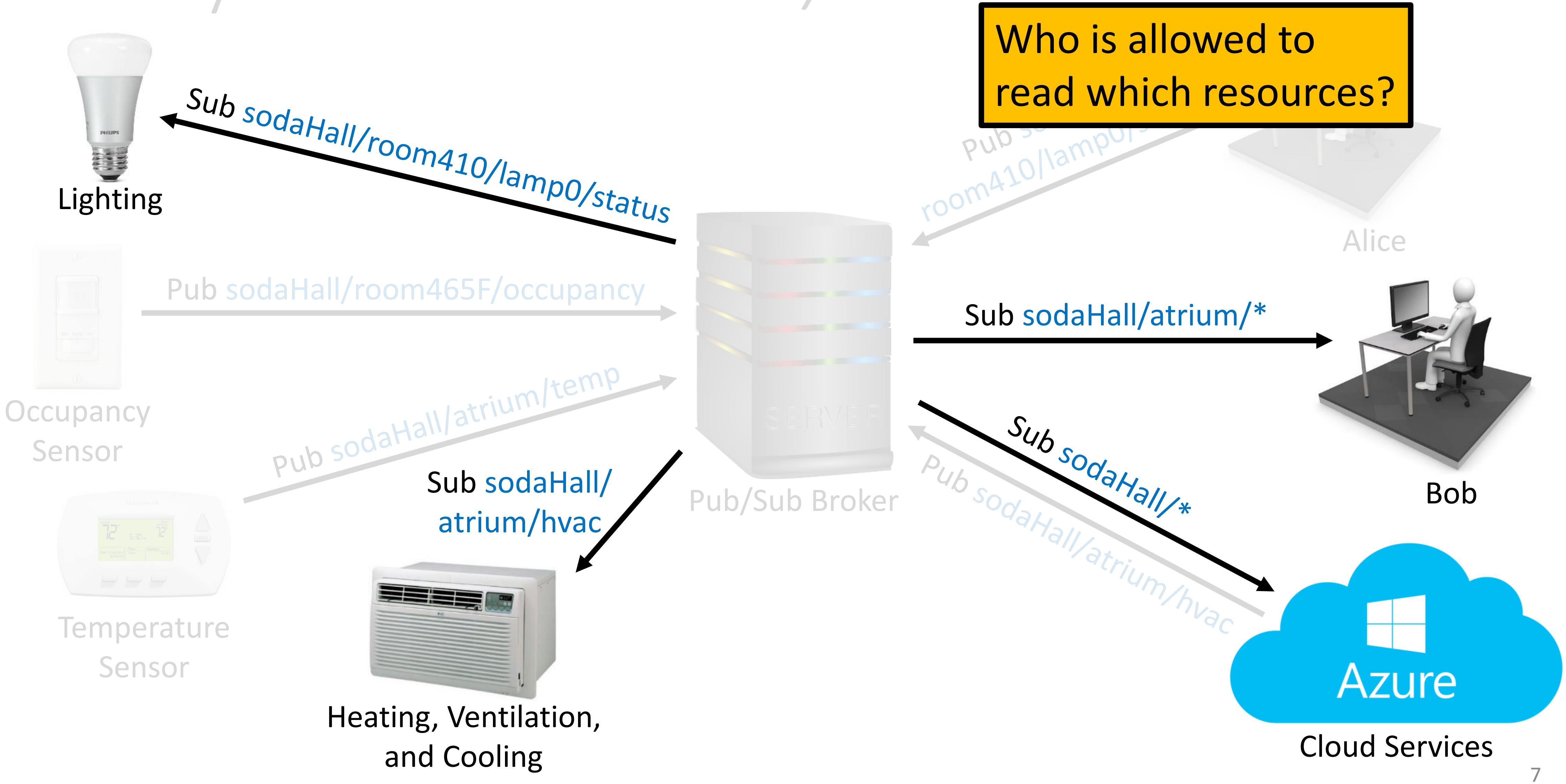
IoT Systems use *Publish/Subscribe*



IoT Systems use *Publish/Subscribe*



IoT Systems use *Publish/Subscribe*



IoT Systems use *Decentralized Delegation*

Access to **sodaHall/***

Access to **sodaHall/room410/*** until **May 2021**

Access to **sodaHall/room410/lamp0/*** until **January 2020**

- Decentralized delegation is an old idea (SPKI/SDSI [CECF01])
- It’s the state-of-the-art for access control in large-scale IoT systems (e.g., Vanadium [TS16], BOSSWAVE [AKCCK17])

IoT Devices are *Resource-Constrained*

Server/Workstation/Laptop

Smartphone

Smart Home Appliance

Wearable

Ultra Low-Power Deeply Embedded Sensor

More Powerful

100,000 DMIPS
10 GiB RAM

Less Powerful

50 DMIPS
32 KiB RAM
Power Constraints

JEDI: Joining Encryption and Delegation for IoT

Joining Encryption and Delegation for IoT

JEDI is an *end-to-end encryption* (E2EE) protocol that:

- Allows senders and receivers to be decoupled as in publish/subscribe
- Supports decentralized delegation
- Can run on resource-constrained IoT devices

Roadmap

1. **Requirements of large-scale IoT systems**

2. JEDI's approach
    a) Encryption in the new model (pub/sub, delegation)
    b) Finding a suitable, lightweight encryption scheme
    c) Anonymous signatures
    d) Revocation

3. Empirical study

Roadmap

1. Requirements of large-scale IoT systems

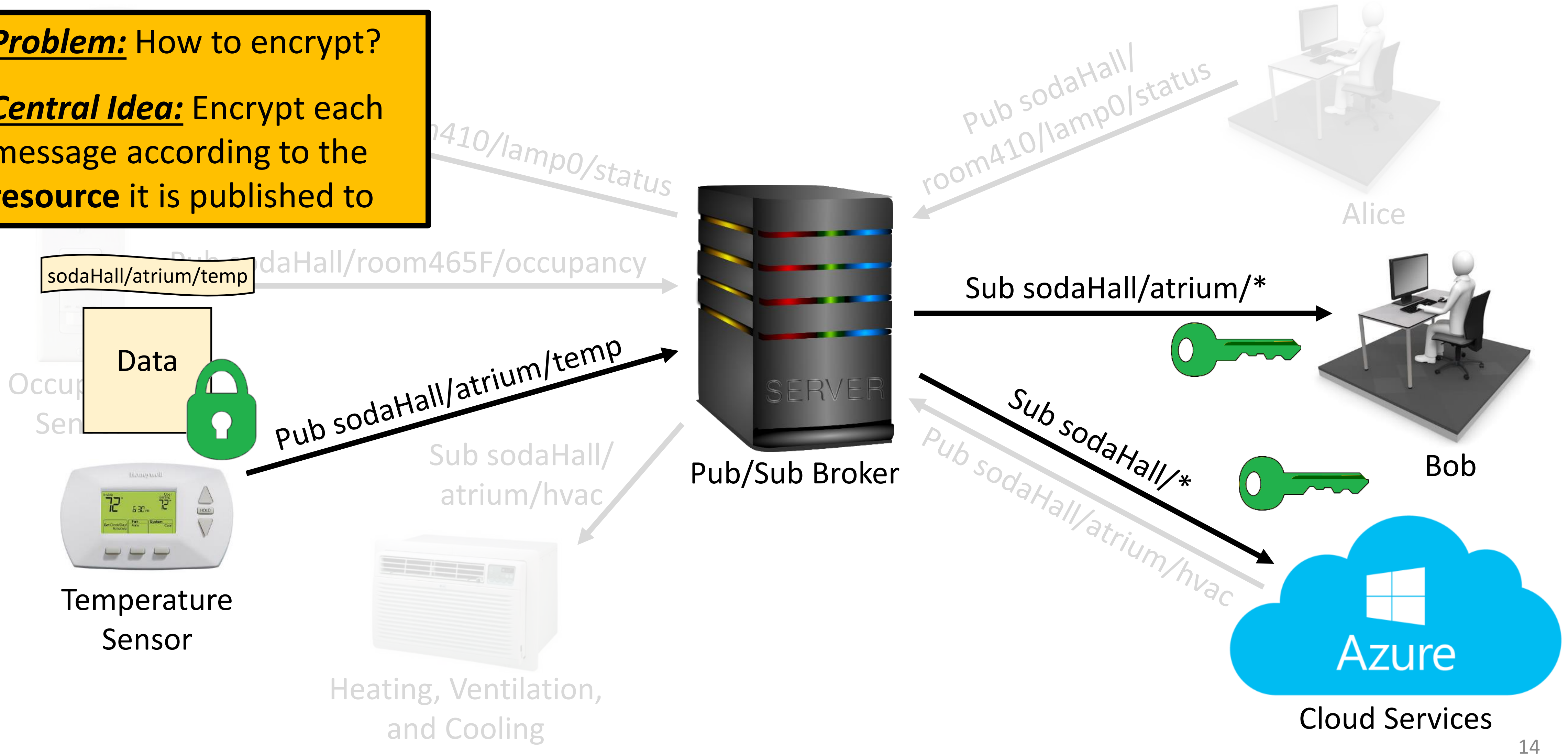
2. **JEDI's approach**
    a) Encryption in the new model (pub/sub, delegation)
    b) Finding a suitable, lightweight encryption scheme
    Focus of this talk (a, b)
    c) Anonymous signatures
    d) Revocation

3. Empirical study

Publish/Subscribe in JEDI

Problem: How to encrypt?

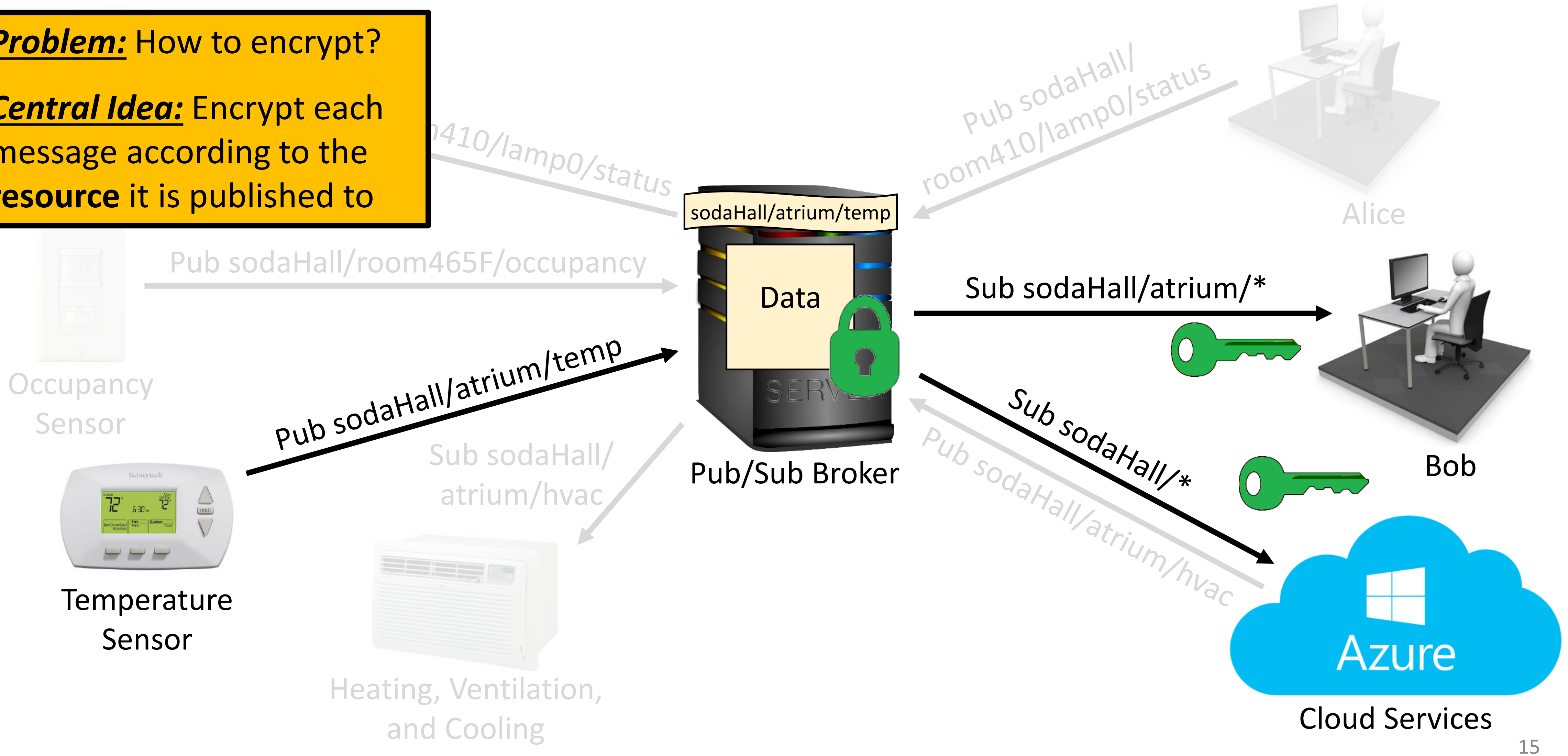
Central Idea: Encrypt each message according to the **resource** it is published to



Publish/Subscribe in JEDI

Problem: How to encrypt?

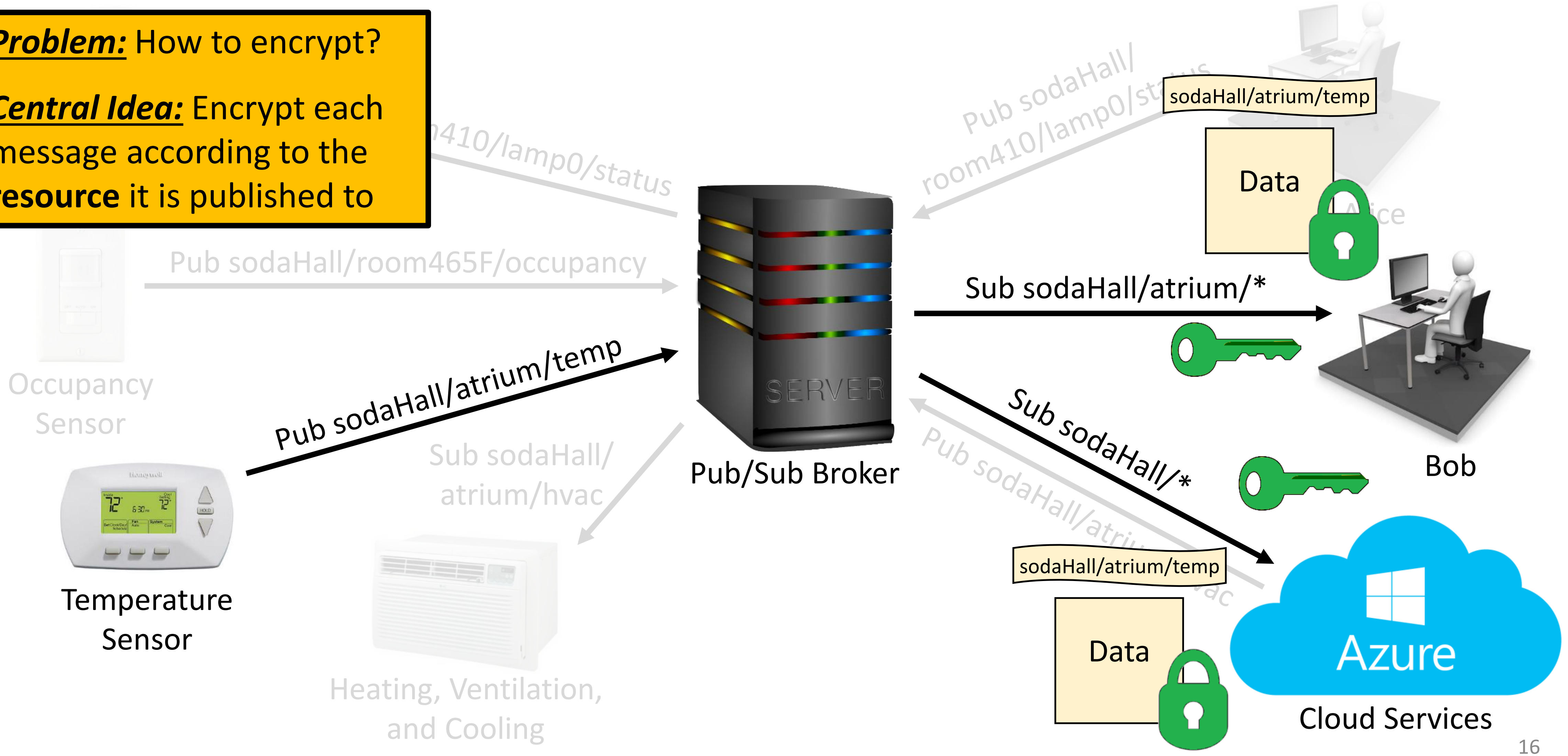
Central Idea: Encrypt each message according to the **resource** it is published to



Publish/Subscribe in JEDI

Problem: How to encrypt?

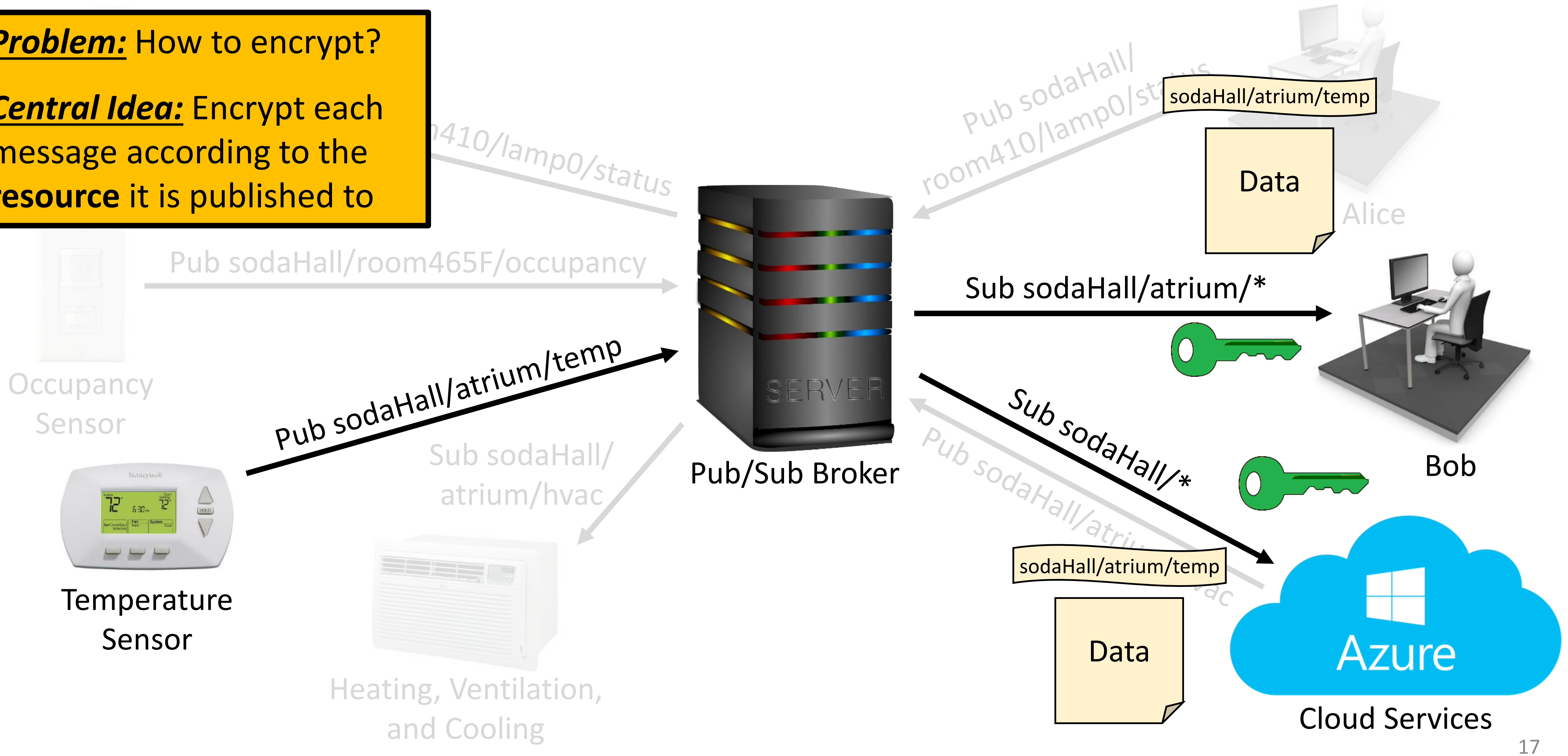
Central Idea: Encrypt each message according to the **resource** it is published to



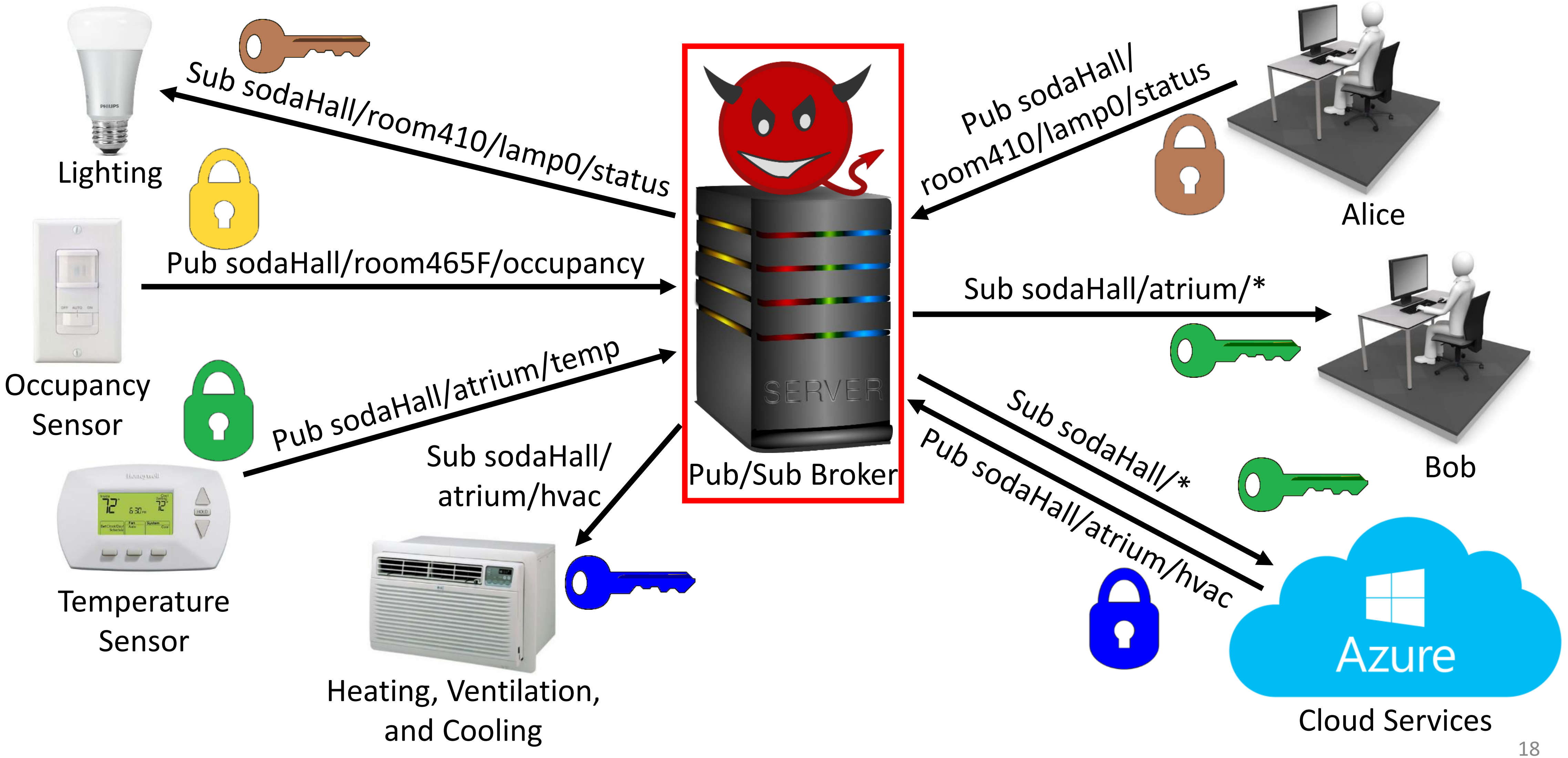
Publish/Subscribe in JEDI

Problem: How to encrypt?

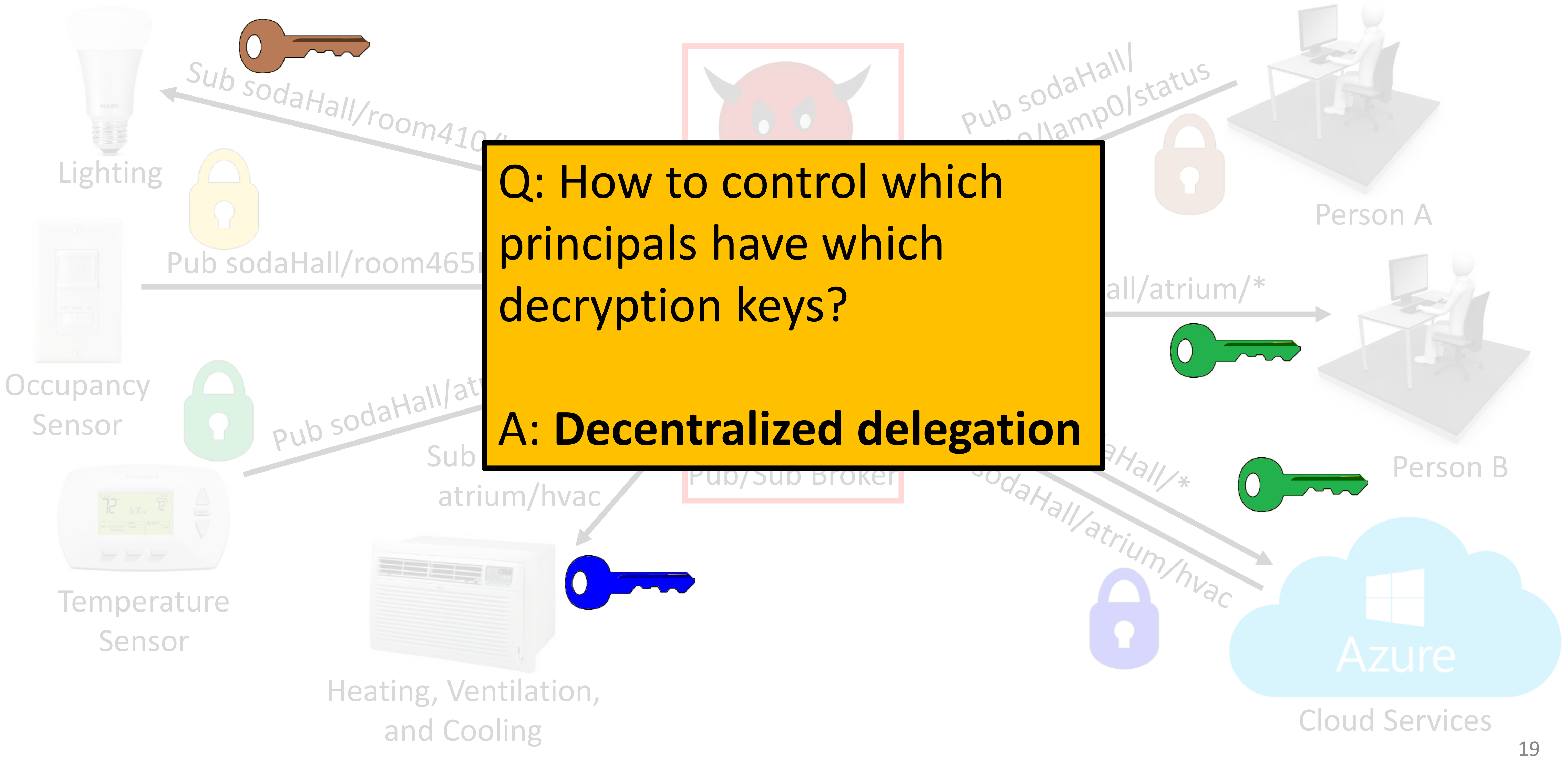
Central Idea: Encrypt each message according to the **resource** it is published to



Publish/Subscribe in JEDI



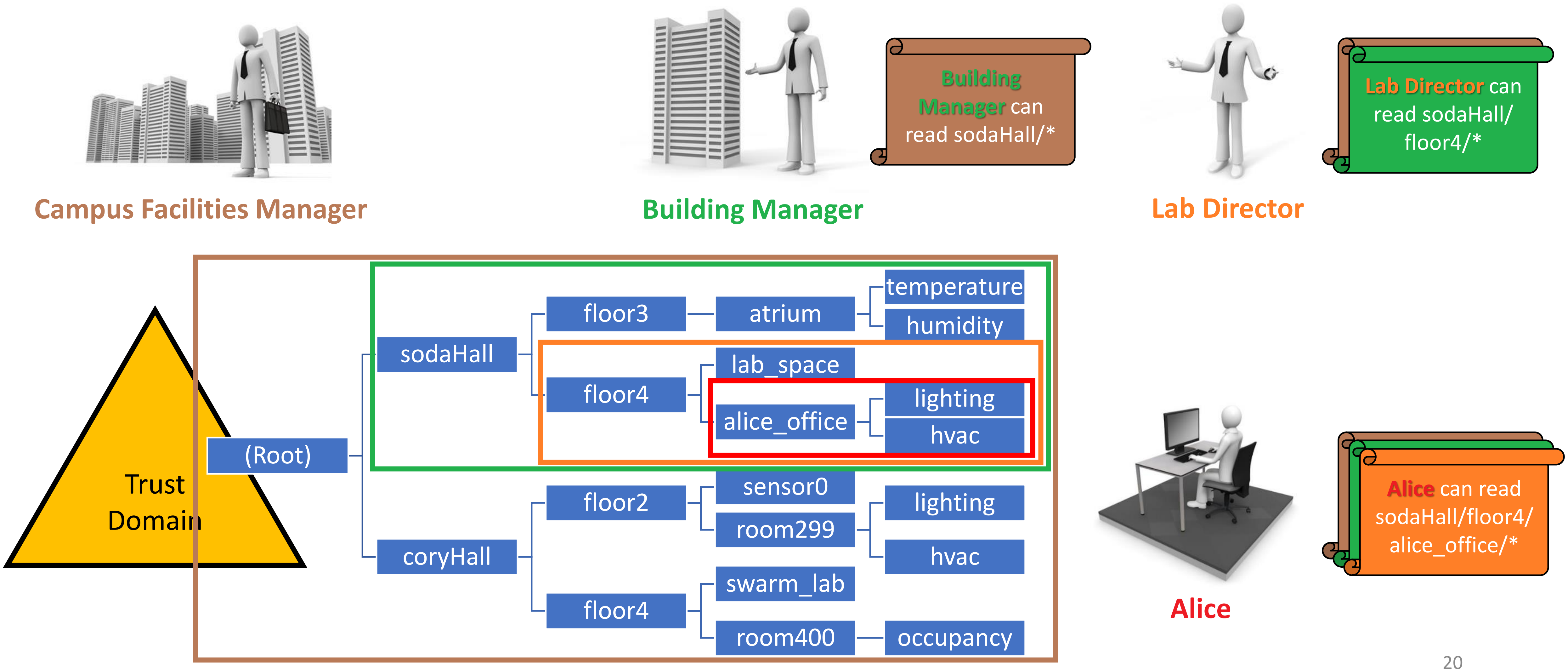
Publish/Subscribe in JEDI



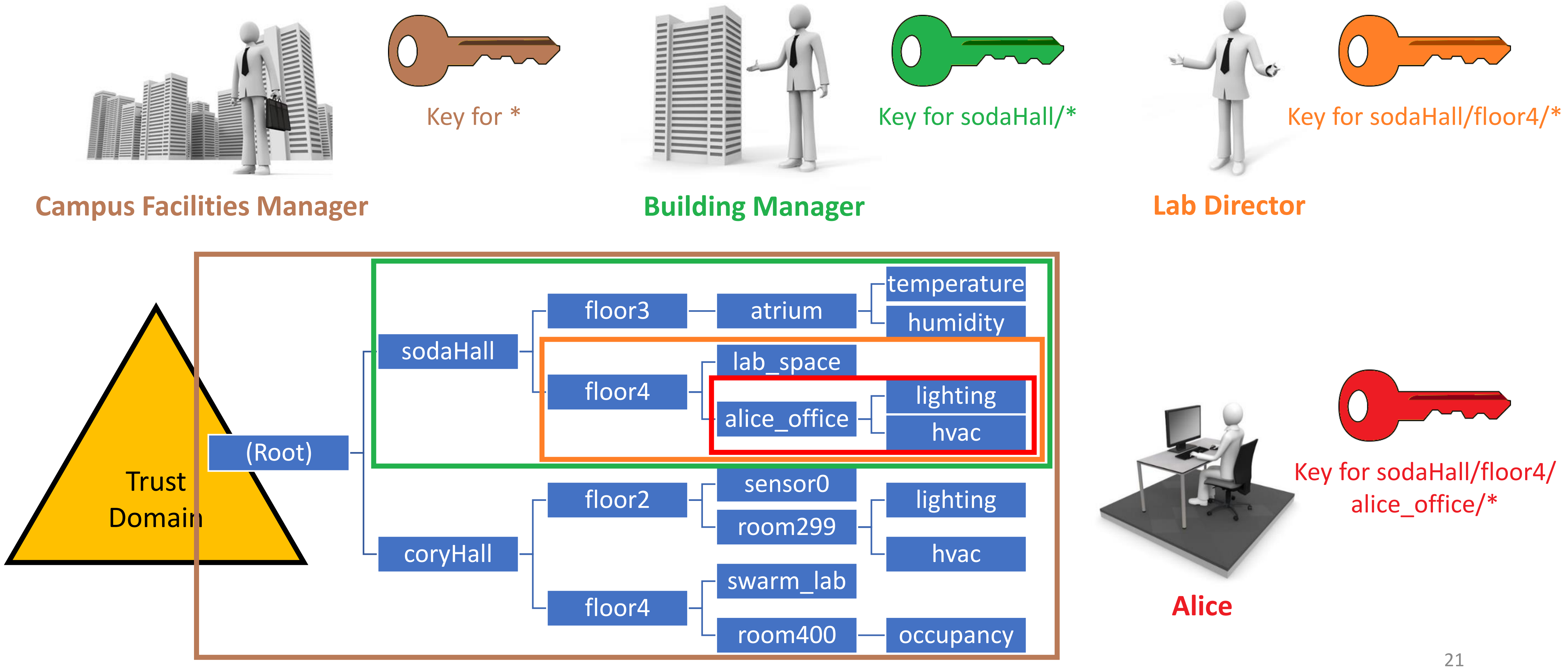
Q: How to control which principals have which decryption keys?

A: **Decentralized delegation**

Decentralized Delegation [CECF01, AKCCK17]



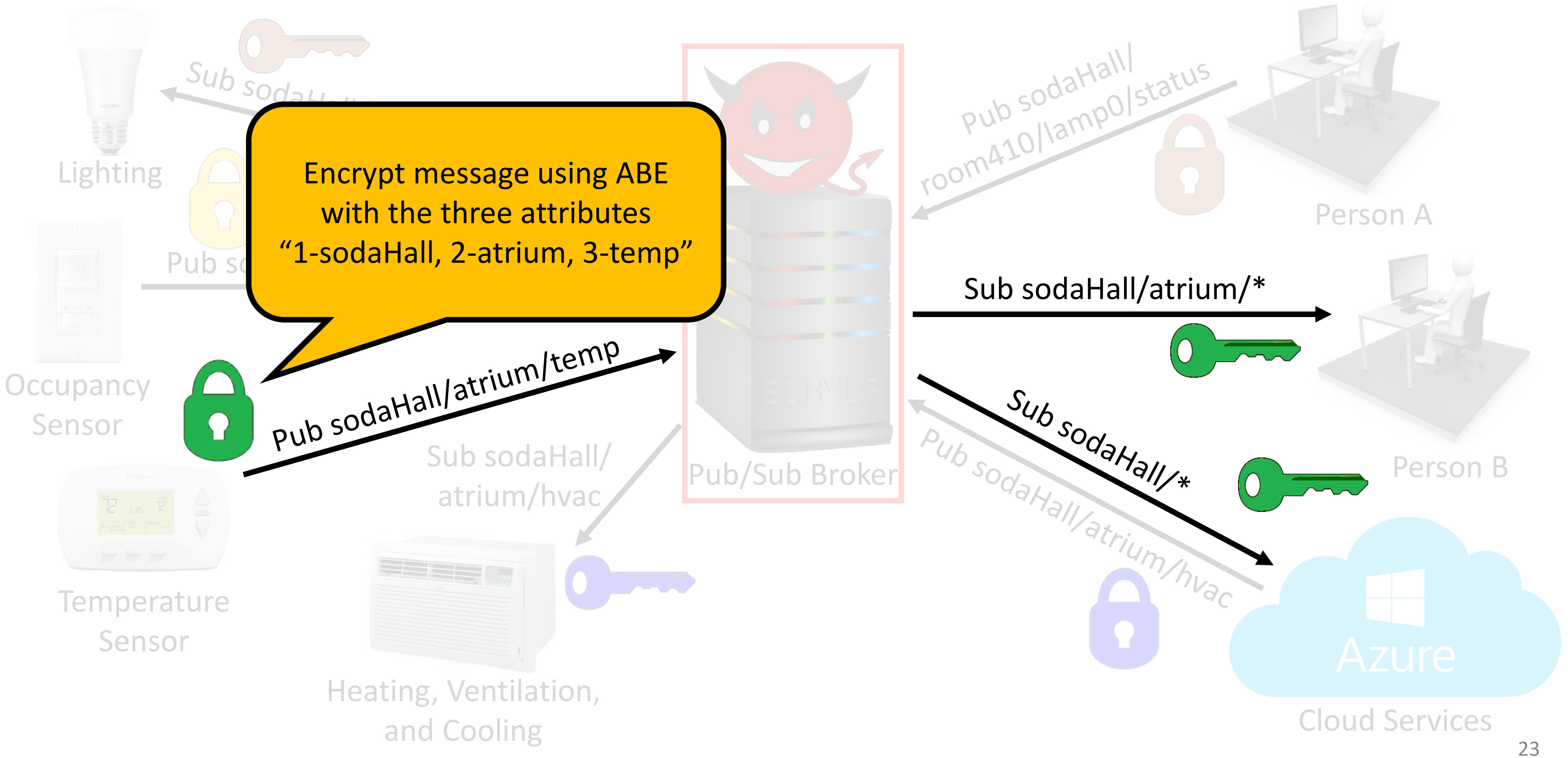
Decentralized Delegation in JEDI



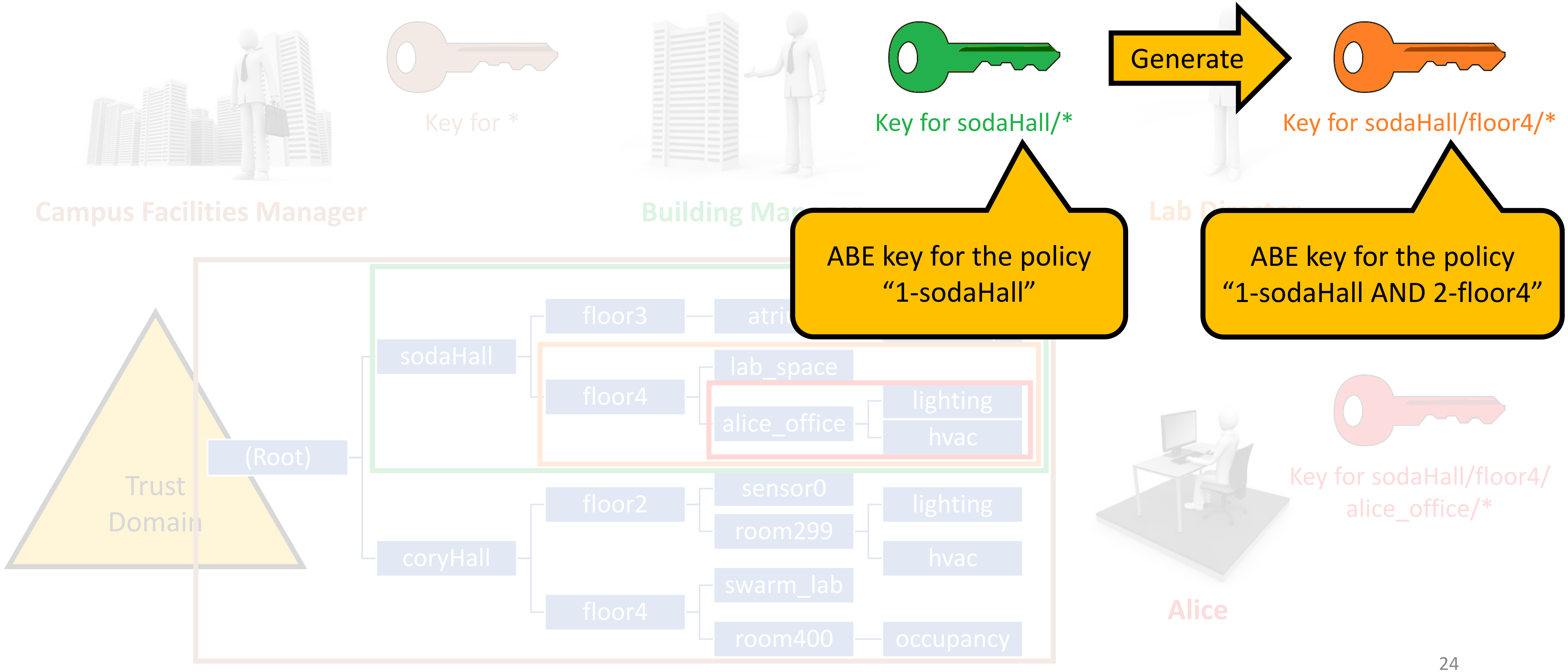
Instantiating JEDI Using Attribute-Based Encryption (**ABE** [GPSW06])

Set aside efficiency for the moment

Preliminary JEDI Design Using ABE

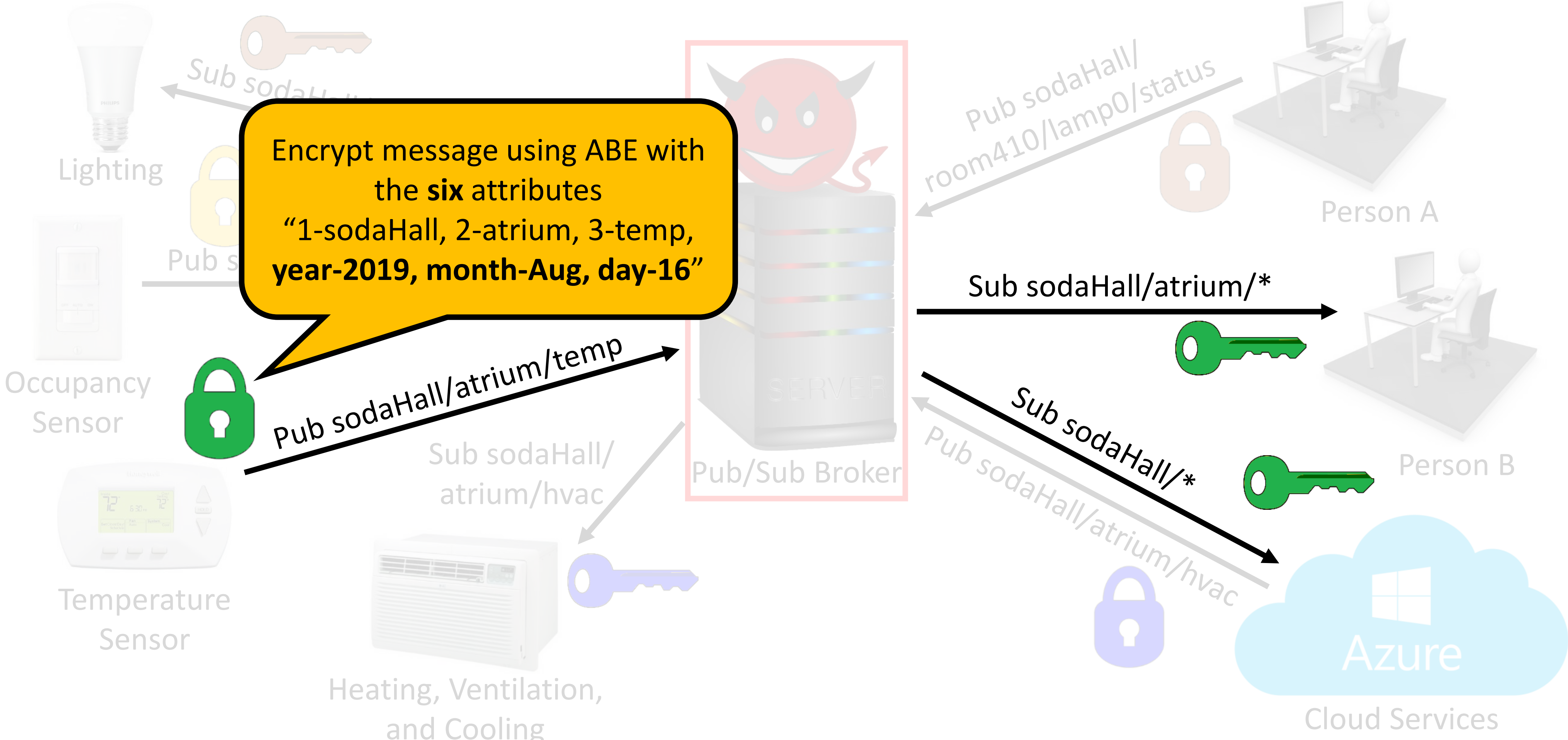


Preliminary JEDI Design Using ABE

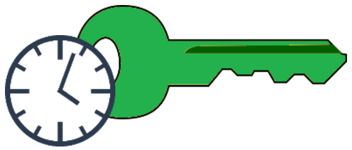


Expiry

Encrypt Using Current Time



Time is Another Hierarchy



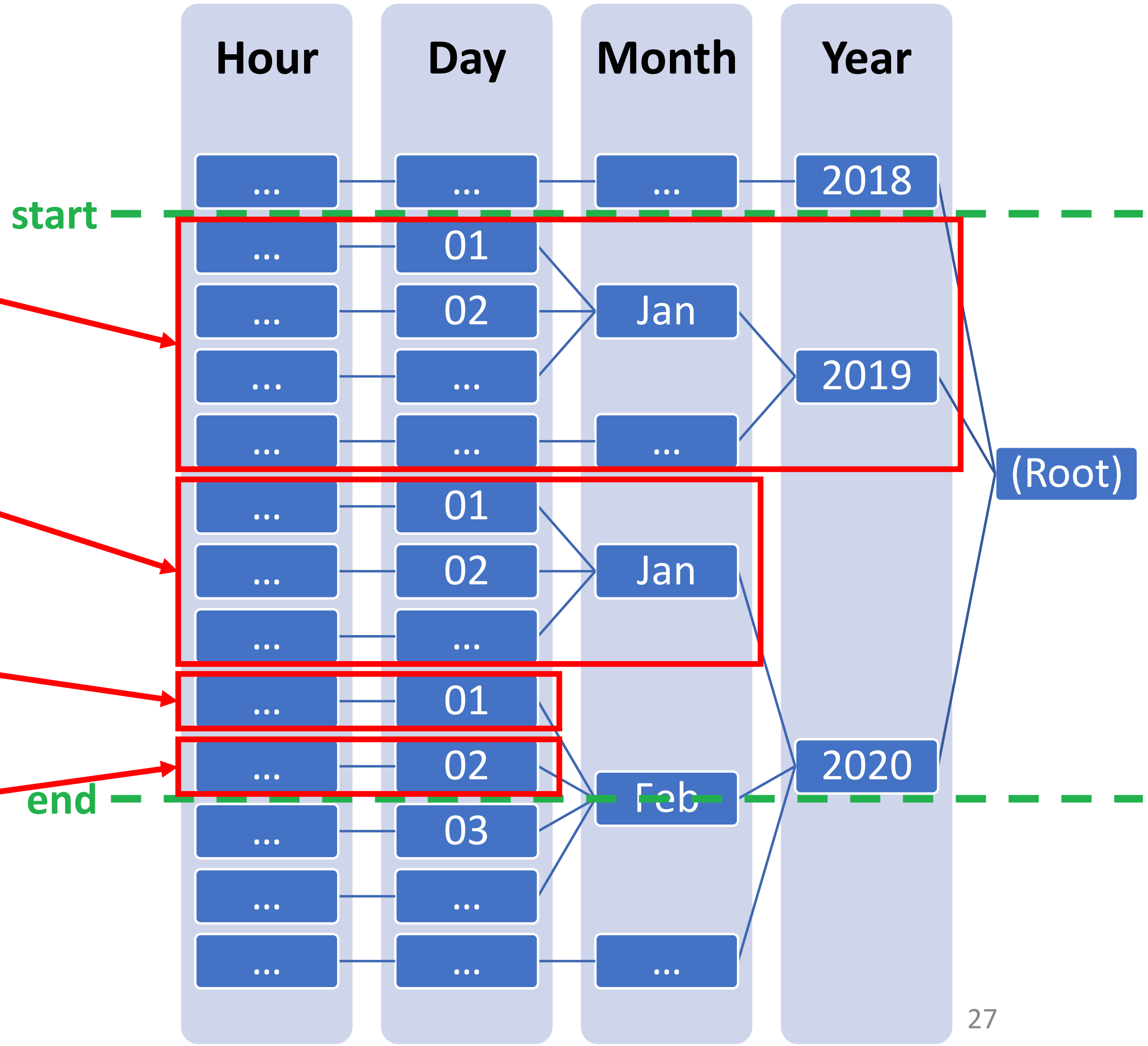
Consists of 4 ABE keys:

Policy: “year-2019”

Policy: “year-2020 AND month-Jan”

Policy: “year-2020 AND month-Feb AND day-01”

Policy: “year-2020 AND month-Feb AND day-02”

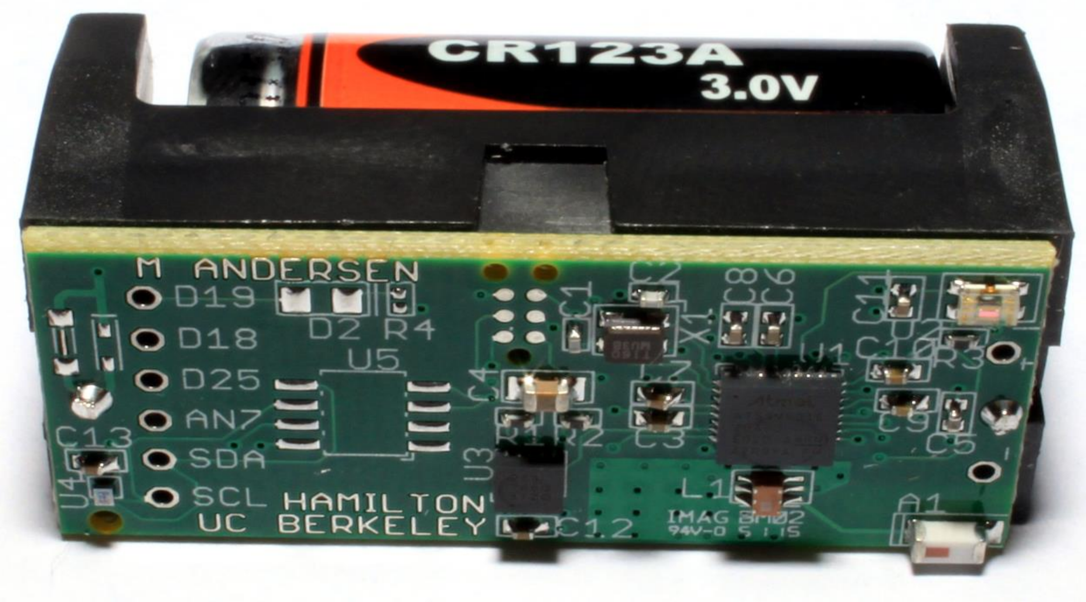


Support for Resource-Constrained Devices

Hamilton Platform [KACKZMC18]

- Based on the Atmel SAMR21 SoC
  - 32-bit ARM Cortex M0+ @ 48 MHz
  - 32 KiB Data Memory (RAM)

- **Goal: *several years* of battery life**
  - $1.00 CR123A Lithium battery



Energy Cost of ABE

- Due to hybrid encryption, we invoke ABE *rarely* (e.g., once per hour)
- Regardless, **ABE dominates power consumption**

- ABE takes 4 minutes on Hamilton → battery won't even last 100 days

Choosing a More Efficient Encryption Scheme

RSA

BE [FN94, BGW05]

WIBE [ACDMNS06]

Fuzzy-IBE [SW04]

CP-ABE [BSW07]

RIBE [BGK08]

RHIBE [SE14]

El Gamal

IBE [BF01]

MRQED [SBCSP07]

WKD-IBE [AKN07]

KP-ABE [GPSW06]

FHE [Gentry09]

Regev

HIBE [GS02, BBG05]

ke-PKE [CHK03]

ti-Authority ABE [LW11]

IPE [KSW08]

HABE [WLWG11]

AIBE [Gentry06]

PRE [BBS98, AF

THD [BY04]

HVE [BW07]

DP-ABE [AI09]

PKE-IP [ABCP15]

AHIBE [BW06]

HIBBE [LLWQ14]

HPE [LOSTW10]

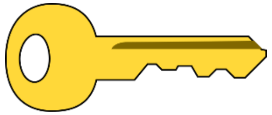
RHIBBE [LLZWL18]

We identify WKD-IBE:

- More efficient than ABE, but much less flexible
- Flexible enough to realize JEDI, **if used carefully**

Summary of WKD-IBE [AKN07]

- Each ciphertext or key encodes a vector of strings and wildcards

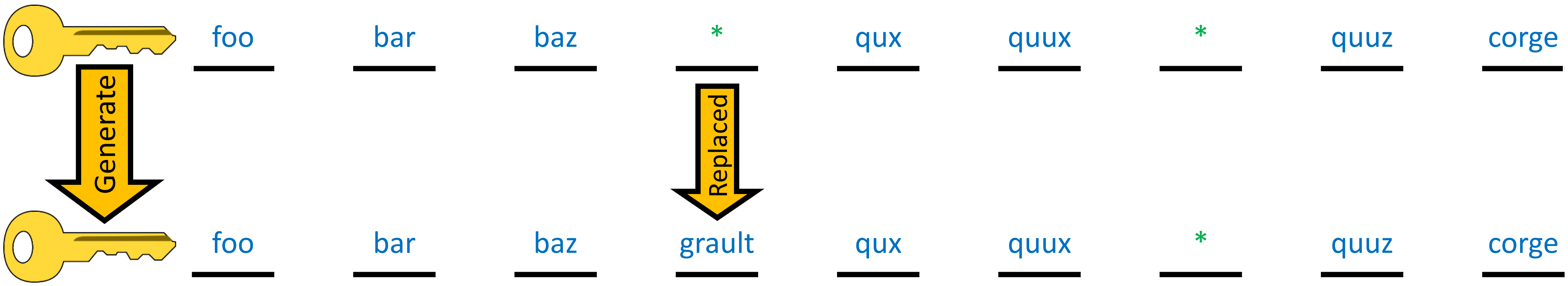


foo bar baz * qux quux * quuz corge

- A key can decrypt a ciphertext if their vectors match
- Given a key, one can generate a new key with some wildcards replaced with strings

Summary of WKD-IBE [AKN07]

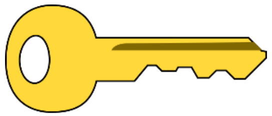
- Each ciphertext or key encodes a vector of strings and wildcards



- A key can decrypt a ciphertext if their vectors match
- Given a key, one can generate a new key with some wildcards replaced with strings

How JEDI Uses WKD-IBE

- JEDI encodes *multiple concurrent hierarchies* into WKD-IBE's vector
- Private key for sodaHall/room410/*, valid for August 2019:



sodaHall	room410	*	*	*	2019	Aug	*	*

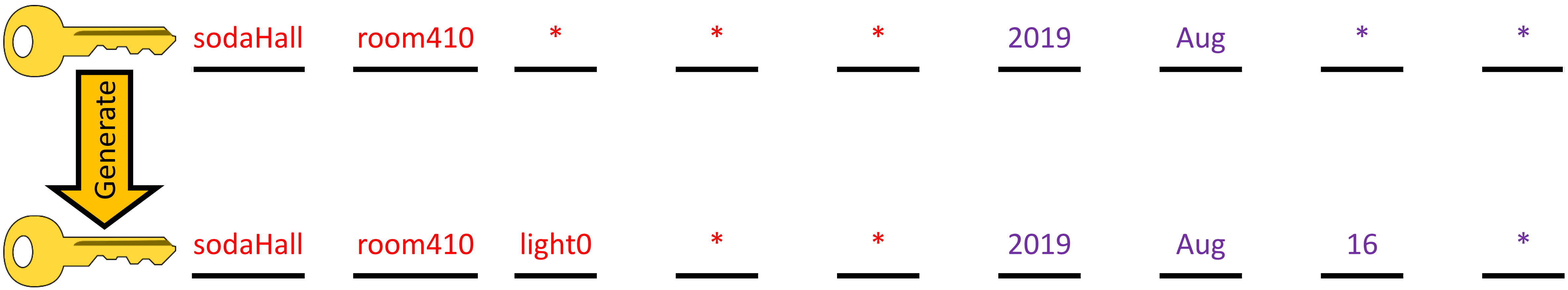
First ℓ_1 components for Resource Hierarchy

Last ℓ_2 components for Time Hierarchy

- For decentralized delegation, we can generate a private key for sodaHall/room410/light0/*, valid for August 16, 2019

How JEDI Uses WKD-IBE

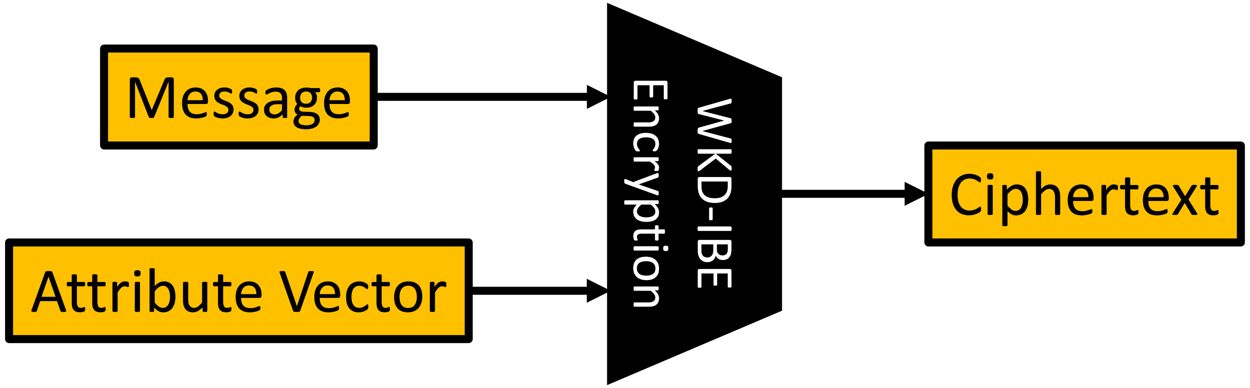
- JEDI encodes *multiple concurrent hierarchies* into WKD-IBE's vector
- Private key for sodaHall/room410/*, valid for August 2019:



- For decentralized delegation, we can generate a private key for sodaHall/room410/light0/*, valid for August 16, 2019

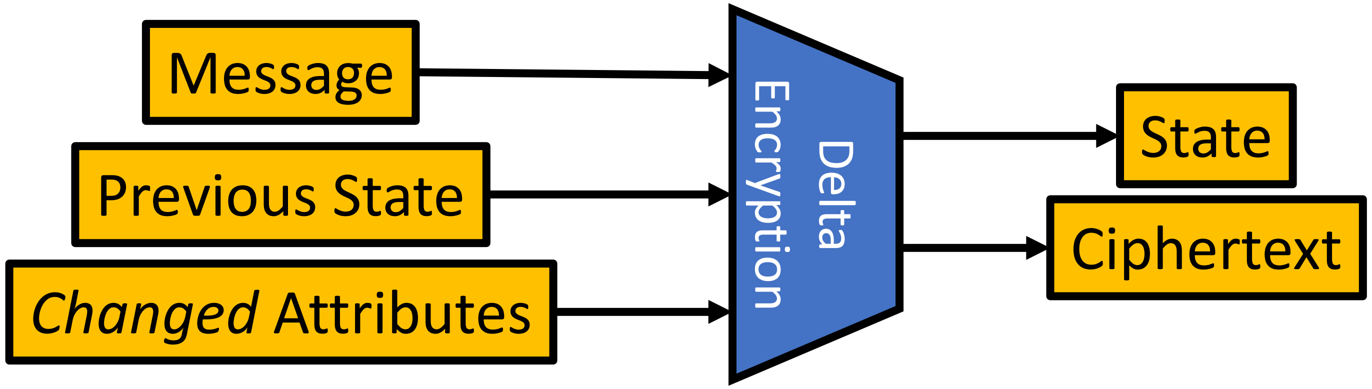
Cryptographic Improvements to WKD-IBE

Existing WKD-IBE Encryption Algorithm [AKN07]



Observation: adjacent encryptions in JEDI differ in only a *few* attributes

JEDI's New WKD-IBE Encryption Algorithm



Idea: encrypt according to the *delta* from the previous attributes

Roadmap

1. Requirements of large-scale IoT systems

2. **JEDI's approach**
   a) Encryption in the new model (pub/sub, delegation)
   b) Finding a suitable, lightweight encryption scheme

   } **Focus of this talk**

   c) Anonymous signatures
   d) Revocation

   } **See paper for details**

3. Empirical study

Roadmap

1. Requirements of large-scale IoT systems

2. JEDI's approach
   a) Encryption in the new model (pub/sub, delegation)
   b) Finding a suitable, lightweight encryption scheme
   c) Anonymous signatures
   d) Revocation

3. **Empirical study**

Implementation

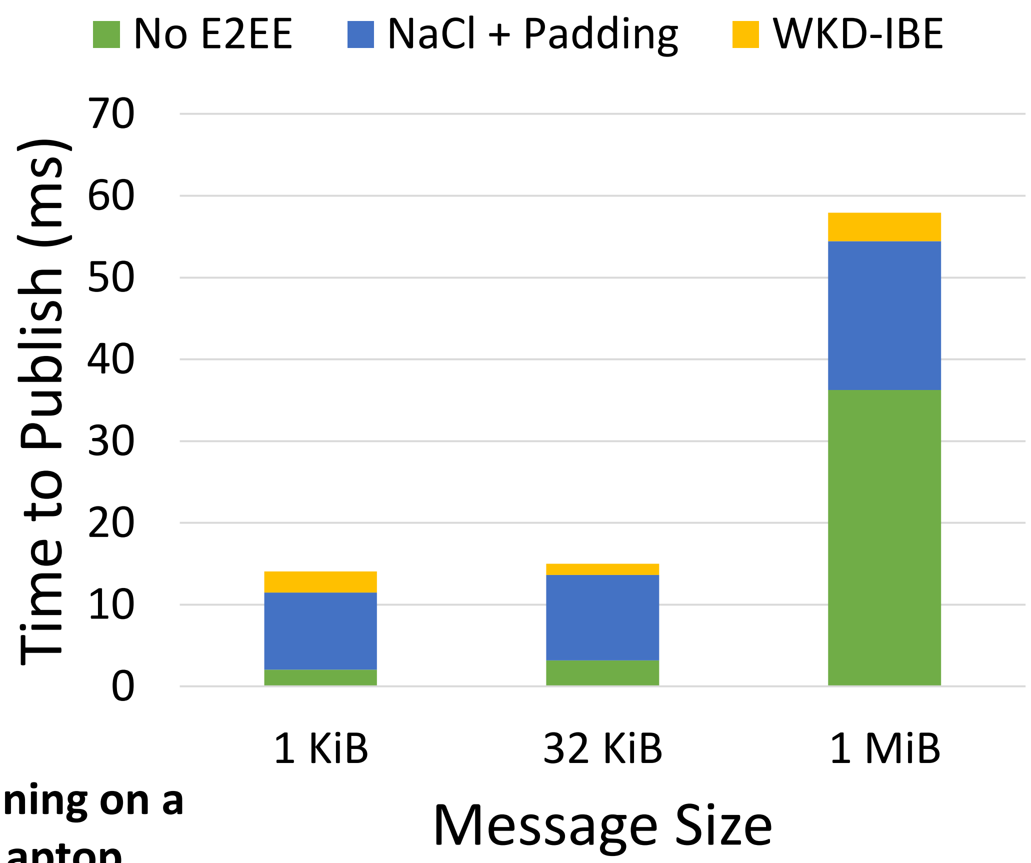
Two parts of JEDI's implementation:

1. JEDI Cryptography Library (https://github.com/ucbrise/jedi-pairing)
    - Includes assembly optimizations for ARM Cortex-M0+ (also x86-64, ARMv8)
    - 4-5x performance improvement over pure C/C++ on Hamilton

2. JEDI Protocol Prototype (https://github.com/ucbrise/jedi-protocol)
    - Implemented for bw2 [AKCFCP17], a messaging system for smart cities

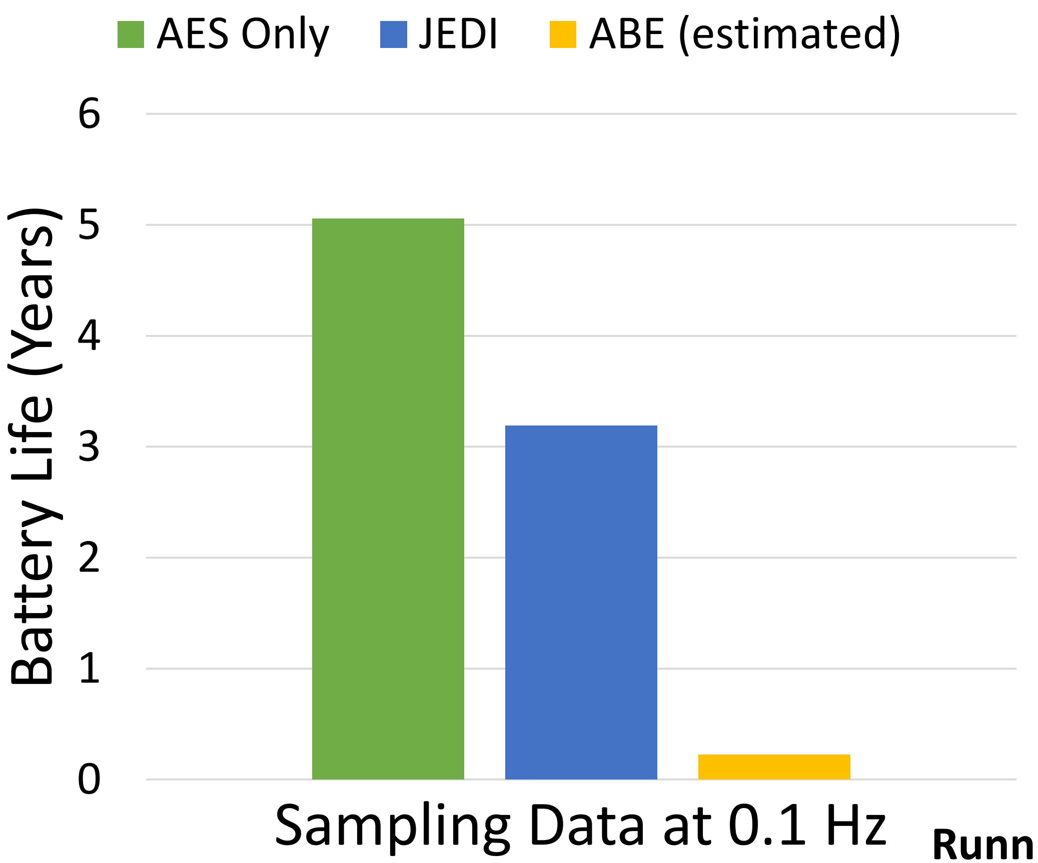
JEDI Applied to bw2 (Running on a Laptop)

- Most of JEDI’s overhead comes from the symmetric-key crypto library (NaCl secretbox)
- JEDI’s overhead is ≈ 10 ms for small messages



Running on a Laptop

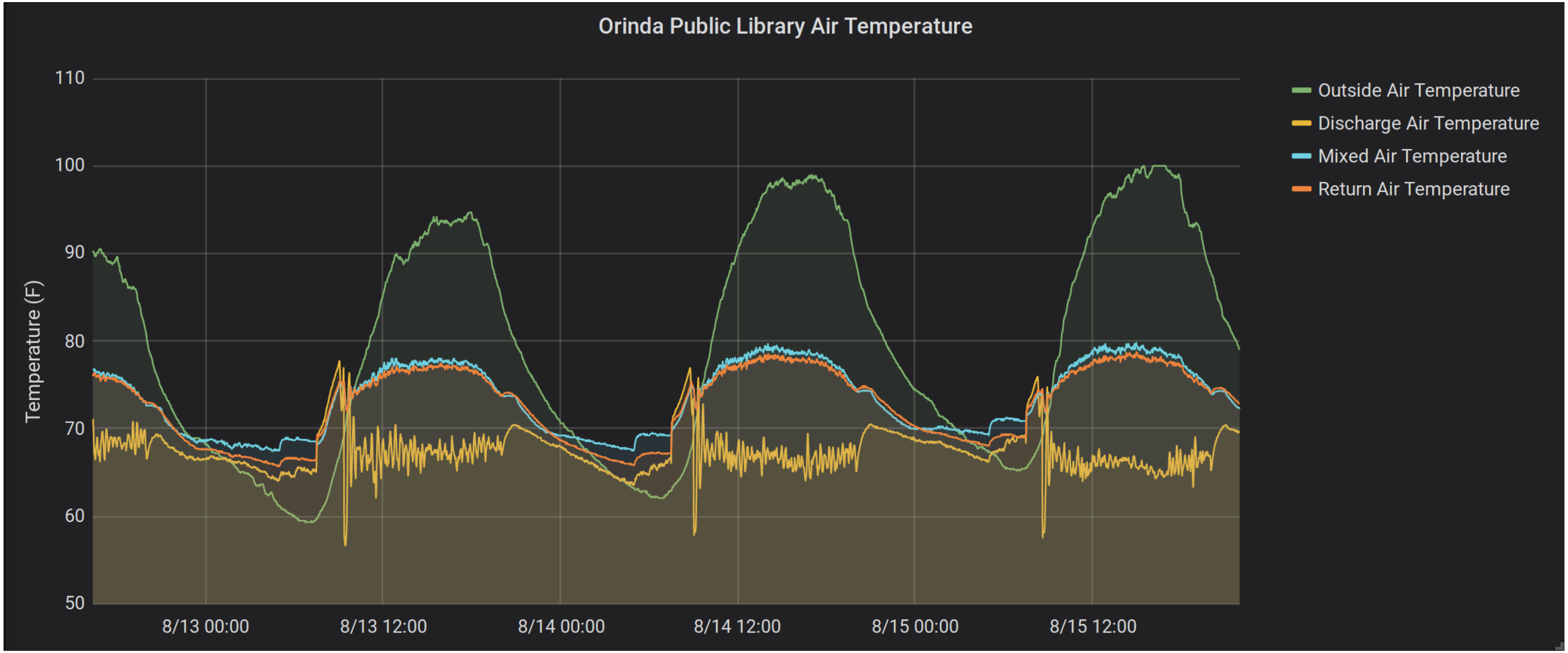
Estimated Battery Life on a Hamilton Sensor



Running on a Hamilton Sensor

- Each encryption with JEDI is 37x more efficient than naïvely applying ABE
- JEDI's battery life, when sampling once every 10 s, is:
  - 14x better than using ABE
  - within 2x of using AES only
  - several years long

We are Deploying JEDI in the Real World!



Conclusion

JEDI is an end-to-end encryption protocol for large-scale IoT systems. It:

- Allows senders and receivers to be decoupled as in publish/subscribe
- Supports decentralized delegation with expiry
- Can run on devices across the spectrum of resource constraints

Thank you for listening!

https://github.com/ucbrise/jedi-pairing
https://github.com/ucbrise/jedi-protocol-go

Extended paper: https://arxiv.org/abs/1905.13369

This material is based on work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1752814. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.