

# IoTivity Security

Habib Virji,  
Samsung Open Source Group, UK.

# Agenda

- Introduction
- OCF Security Functionality
- IoTivity Implementation
- Sample Apps

# Introduction

- IoTivity Security intends to provide:
  - Ensure only authenticated user has access.
  - Data is secured and encrypted.
  - Authorization to access the resource.
- IoTivity security steps:
  - Onboarding a new device to the user network.
  - Provisioning a new device into the user network.
  - Secure connection establishment (DTLS).
  - Control access based on ACL (Access Control List).

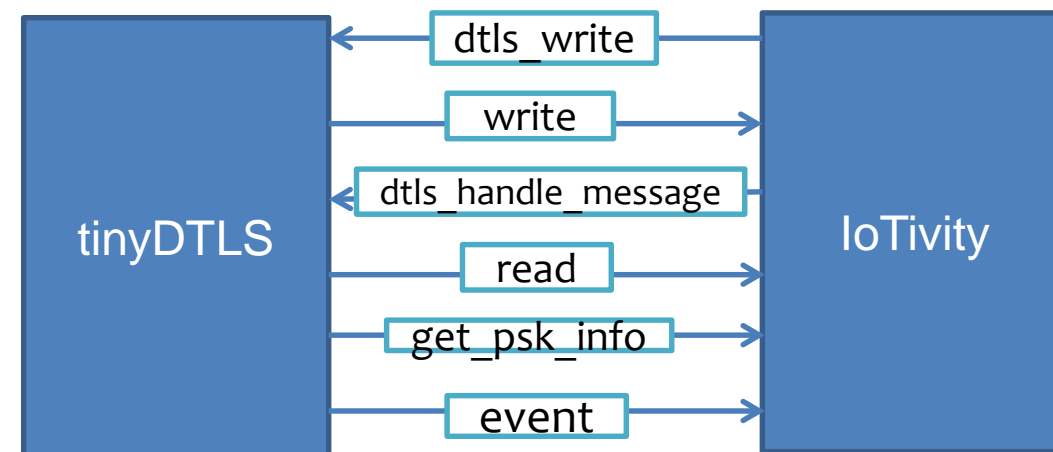
# OCF Security

# Onboarding

- Three possible ways supported:
  - Just Work
    - This mode is specifically for device without display.
  - Random PIN
    - Both ends need to enter same PIN.
  - Asymmetric Key
    - Self-signed or Manufacturer certificate.
- Key is generated by provisioning tool and is transferred securely.
- DTLS connection established uses the key generated.
- It also generates a device id (UUID) that identifies the device.

# Network Connection

- Network security relies on DTLS.
- DTLS connection uses private key generated via onboarding.
- DTLS provides packet by packet encryption.
- DTLS steps involved are:
  - Client verifies server using Device ID.
  - Client if it matches send server message.
  - Server verifies message exchange.



# Access Control List

- Control access to the which device has access to what resources.
- Any packet coming from CA layer is first handled by secure resource manager.
- Secure manager check resource and the device id.
- Each resource has a permission which allows read or write operation.
- ACL can be changed/updated via the provisioning tool.
- ACL is handled at the server end.
- AMS can be used to manage ACL remotely.

# Security Resources

- Different type of secure resources exist:
  - Doxm resource specifies properties needed to establish a device ownership.
  - Pstat resource specifies device provisioning status.
  - Cred resource specifies credentials a device may use to establish secure communication.
  - ACL resource specifies the local access control list.
  - AMACL resource specifies the host resources with access permission that is managed by an AMS.
  - SVC resource specifies the services device recognizes.
  - CRL resource specifies certificate revocation lists as X.509 objects.

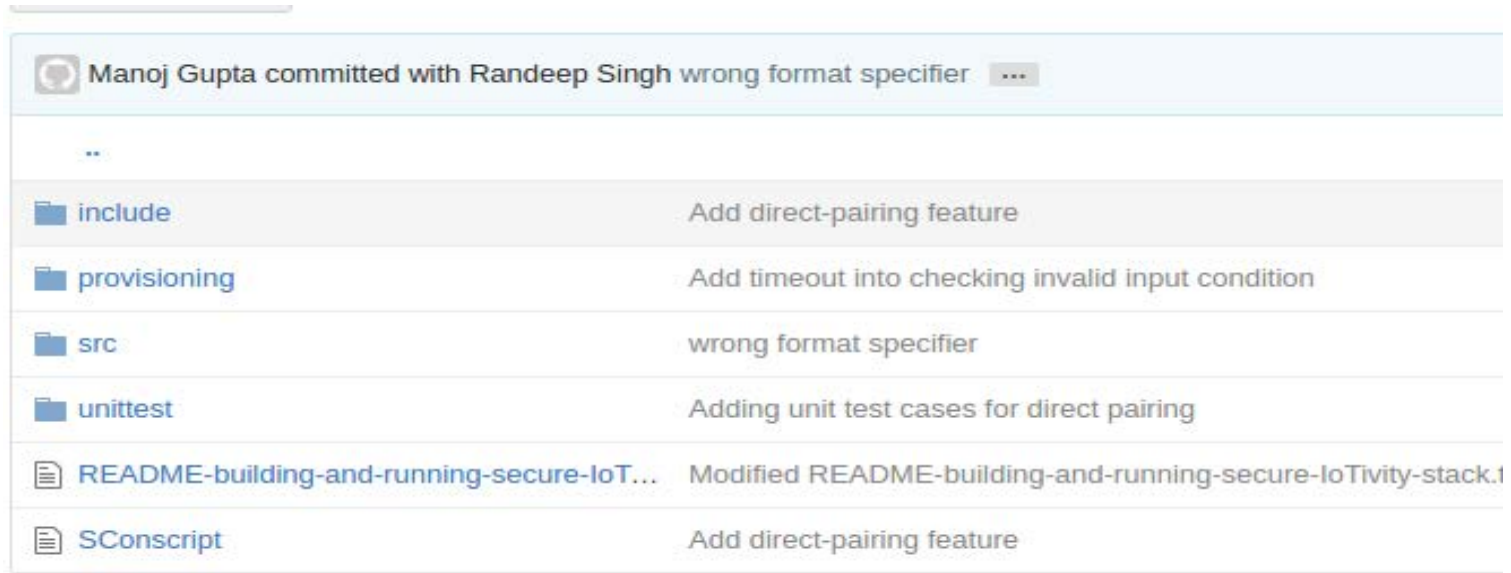


# IoTivity Implementation

# Building Security

- Default build does not include security.
- Building options:
  - `scons SECURED=1`
  - `scons SECURED=1 resource`
  - `./auto_build.sh linux_secured`
  - There are other option to build secured with RD, Remote too.
- Location of the security codebase: [resource/csdk/security/](https://github.com/Open-Connectivity-Foundation/ocf-csdk/tree/master/security)

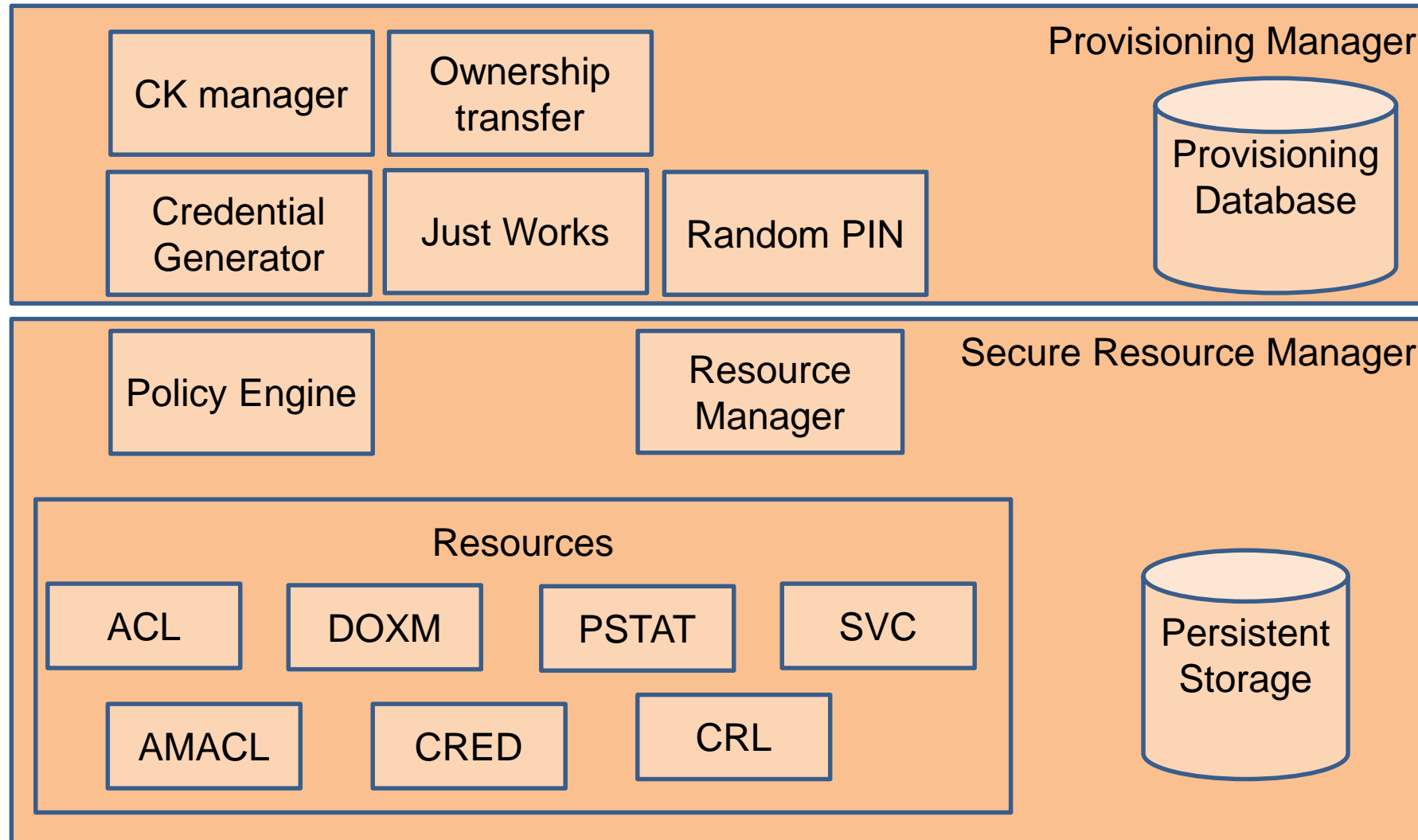
# Code Structure



A screenshot of a commit message and a list of files. The commit message at the top reads: "Manoj Gupta committed with Randeep Singh wrong format specifier". Below the message is a list of files and folders with their corresponding commit descriptions:

File/Folder	Description
..	
include	Add direct-pairing feature
provisioning	Add timeout into checking invalid input condition
src	wrong format specifier
unittest	Adding unit test cases for direct pairing
README-building-and-running-secure-IoT...	Modified README-building-and-running-secure-IoTivity-stack.1
SConscript	Add direct-pairing feature

# Security Building Blocks



# Sample Apps

# Onboarding - Symmetric

```
cd iotivity
```

```
cd out/linux/x86_64/debug/resource/csdk/security/provisioning/sample/
```

- Just Works

```
./sampleserver_justworks
```

- Random PIN

```
./sampleserver_randompin
```

- Provisioning Client

```
./provisioningclient
```

# Onboarding - Asymmetric

```
cd iotivity
```

```
cd out/linux/x86_64/debug/resource/csdk/security/provisioning/ck_manager/sample/
```

(Open each in different terminals)

```
./Light_server
```

```
./Door_server
```

```
./provisioningclient
```

# Thank You