What's New in HomeKit

Session 210

Anush Nadathur HomeKit Engineer

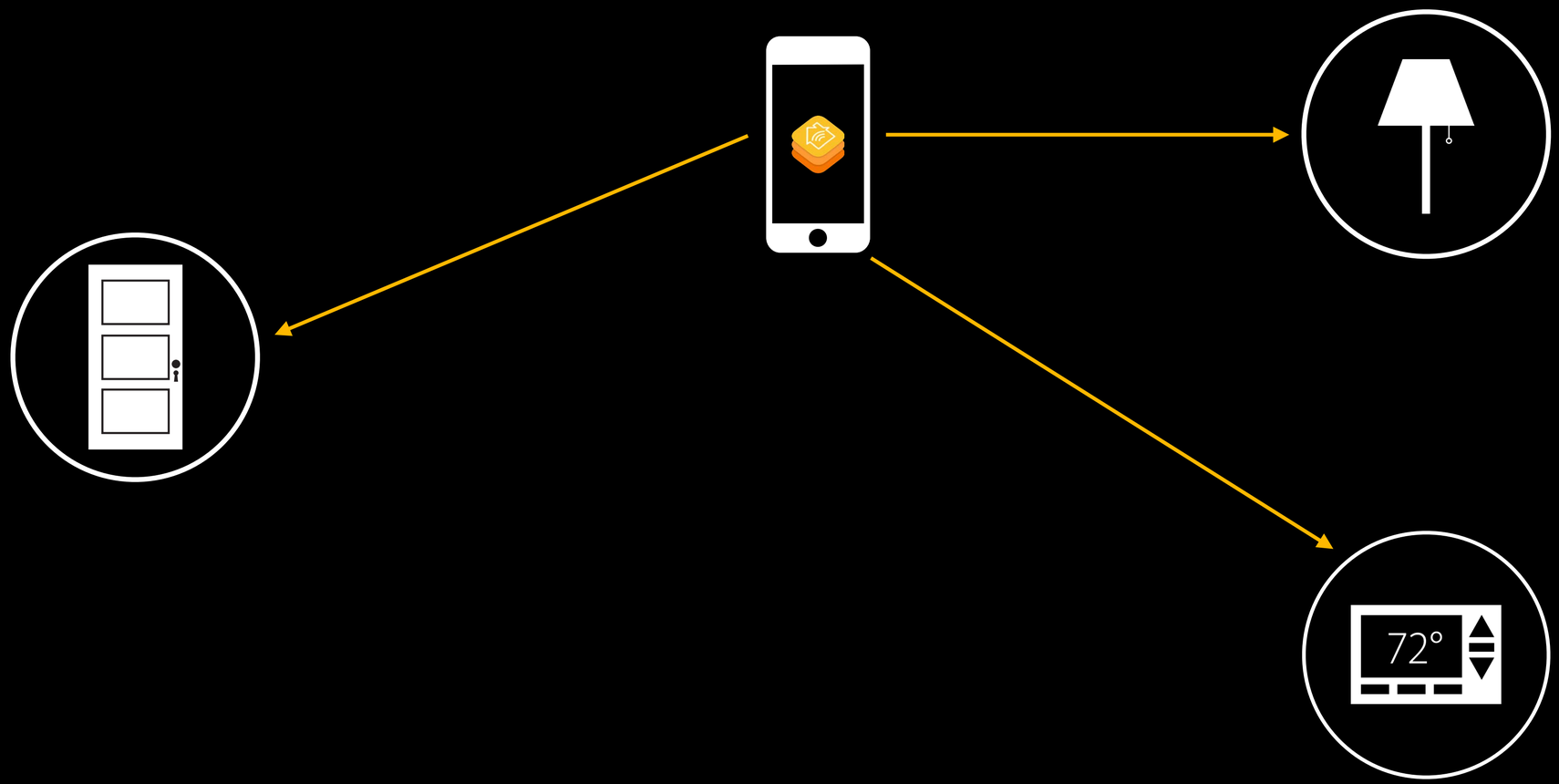
Naveen Kommareddi HomeKit Engineer

HomeKit

Simplifying home automation

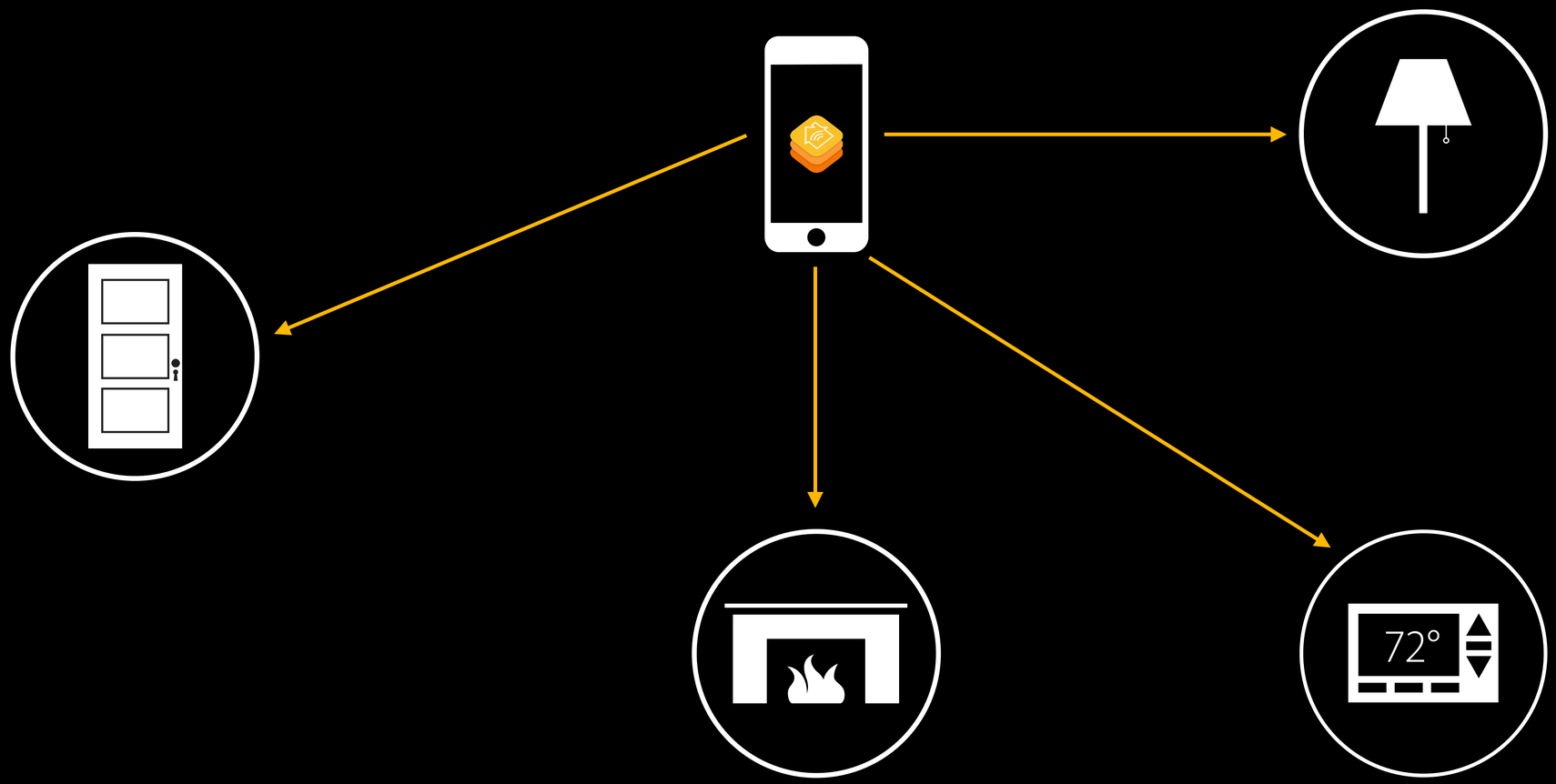
HomeKit

Simplifying home automation



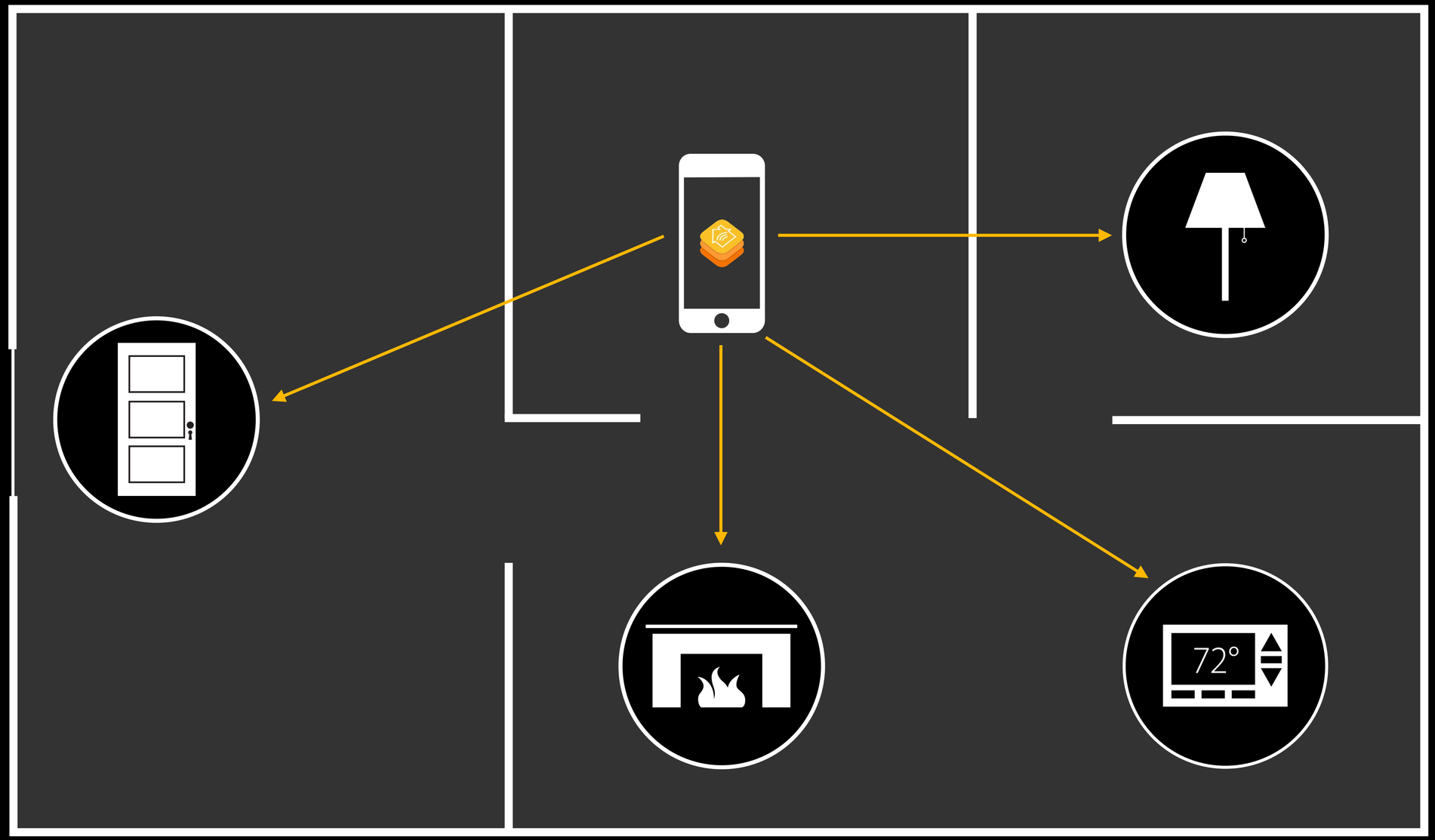
HomeKit

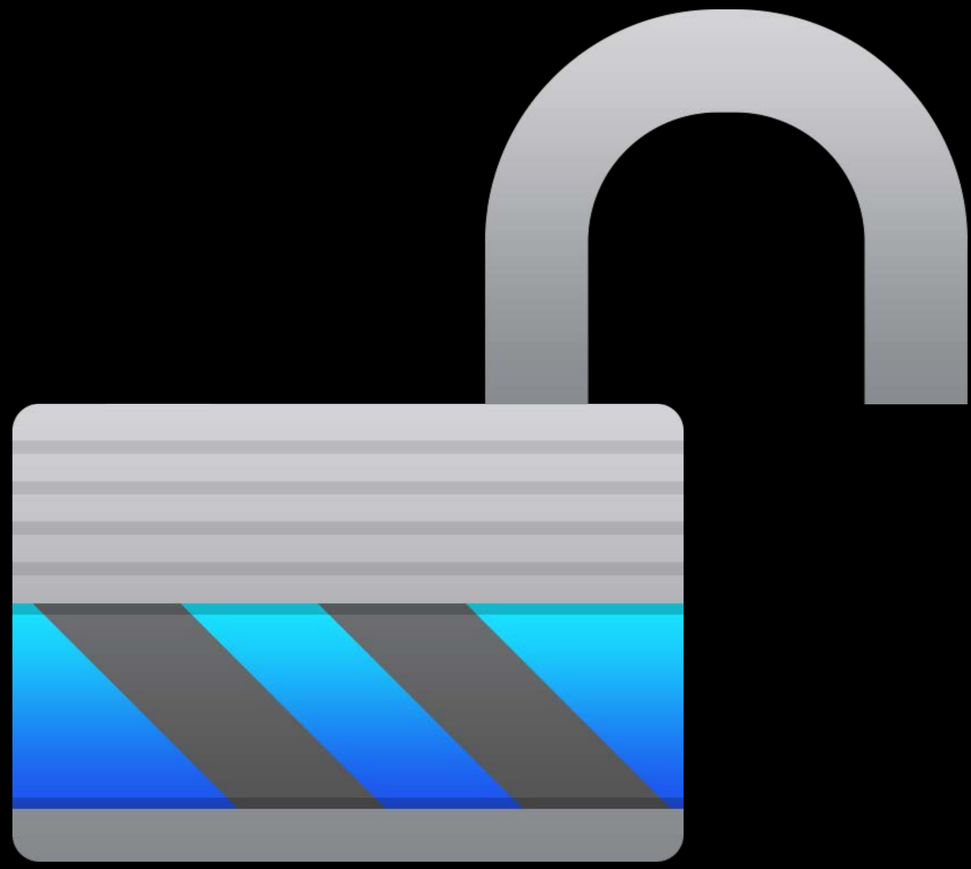
Simplifying home automation



HomeKit

Simplifying home automation





HomeKit in iOS 9

HomeKit in iOS 9

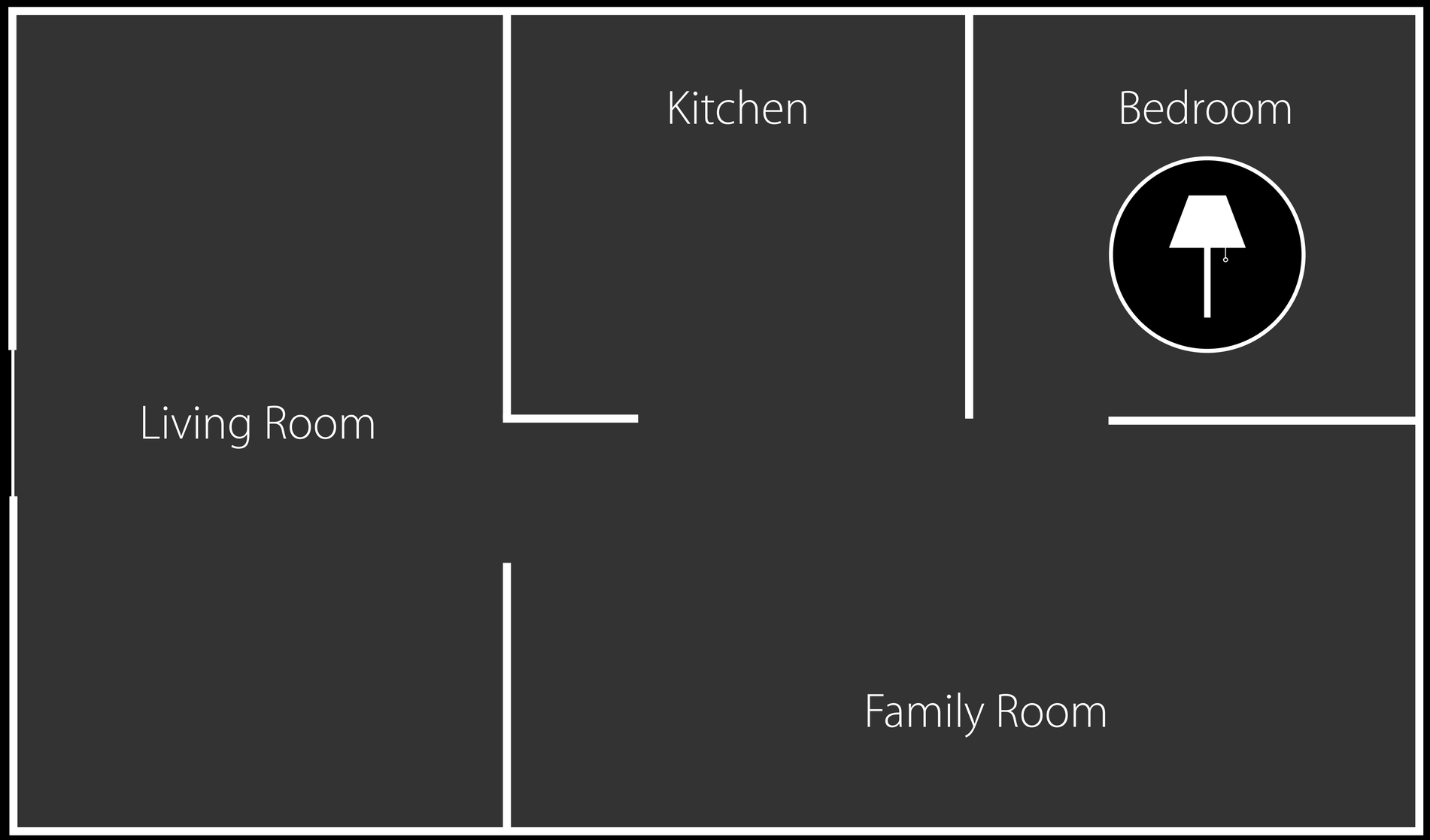
Accessory Updates

HomeKit in iOS 9

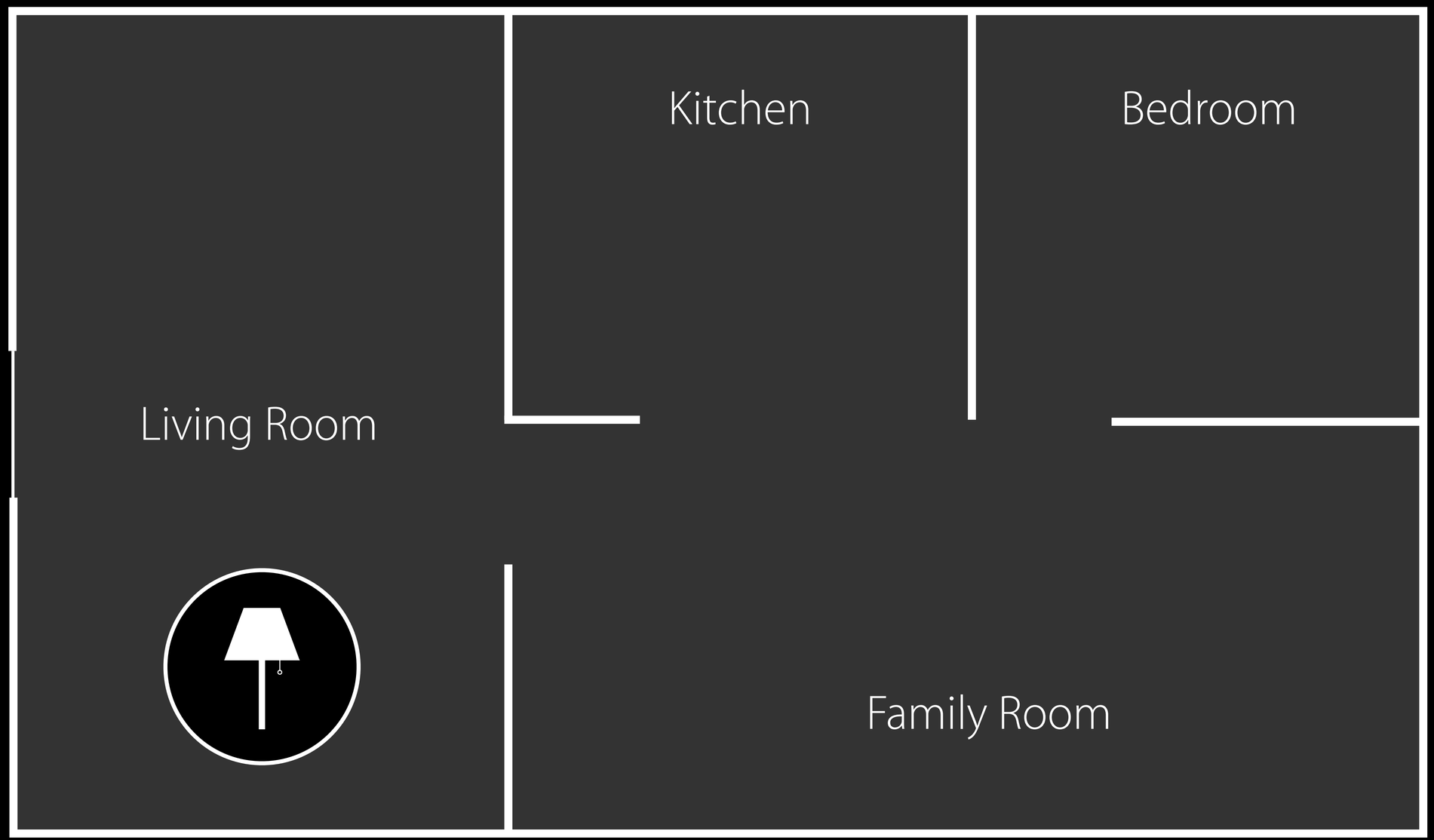
Accessory Updates

HomeKit in iOS 9

Maintain Existing Objects



Maintain Existing Objects



Maintain Existing Objects

Targeted change notifications

```
protocol HMHomeManagerDelegate : NSObjectProtocol {
    func homeManagerDidUpdateHomes(manager: HMHomeManager)
}
```

Maintain Existing Objects

Targeted change notifications

```
protocol HMHomeManagerDelegate : NSObjectProtocol {
    func homeManagerDidUpdateHomes(manager: HMHomeManager)
}
```

Maintain Existing Objects

Targeted change notifications

```
protocol HMHomeManagerDelegate : NSObjectProtocol {
    func homeManagerDidUpdateHomes(manager: HMHomeManager)
}
```

```
protocol HMHomeDelegate : NSObjectProtocol {
    func home(home: HMHome, didUpdateRoom: HMRoom, forAccessory: HMAccessory)
}
```

My Living Room

```
class HMRoom : NSObject {
    var name: String { get }
}
```

Persistent Identifiers

NEW

```
class HMRoom : NSObject {
    var name: String { get }
}
```

Persistent Identifiers

NEW

```
class HMRoom : NSObject {
    var name: String { get }
    var uniqueIdentifier: NSUUID { get }
}
```

Available on all relevant HomeKit classes

User Management

```
extension HMHome {
    func addUserWithCompletionHandler(completion: (HMUser?, NSError?)-> Void)

    func removeUser(user: HMUser, completionHandler: (NSError?)-> Void)

    var users: [HMUser] { get }
}
```

User Management

```
extension HMHome {
    func addUserWithCompletionHandler(completion: (HMUser?, NSError?)-> Void)

    func removeUser(user: HMUser, completionHandler: (NSError?)-> Void)

    var users: [HMUser] { get }
}
```

User Management

```
extension HMHome {
    func addUserWithCompletionHandler(completion: (HMUser?, NSError?)-> Void)

    func removeUser(user: HMUser, completionHandler: (NSError?)-> Void)

    var users: [HMUser] { get }
}
```

User Management

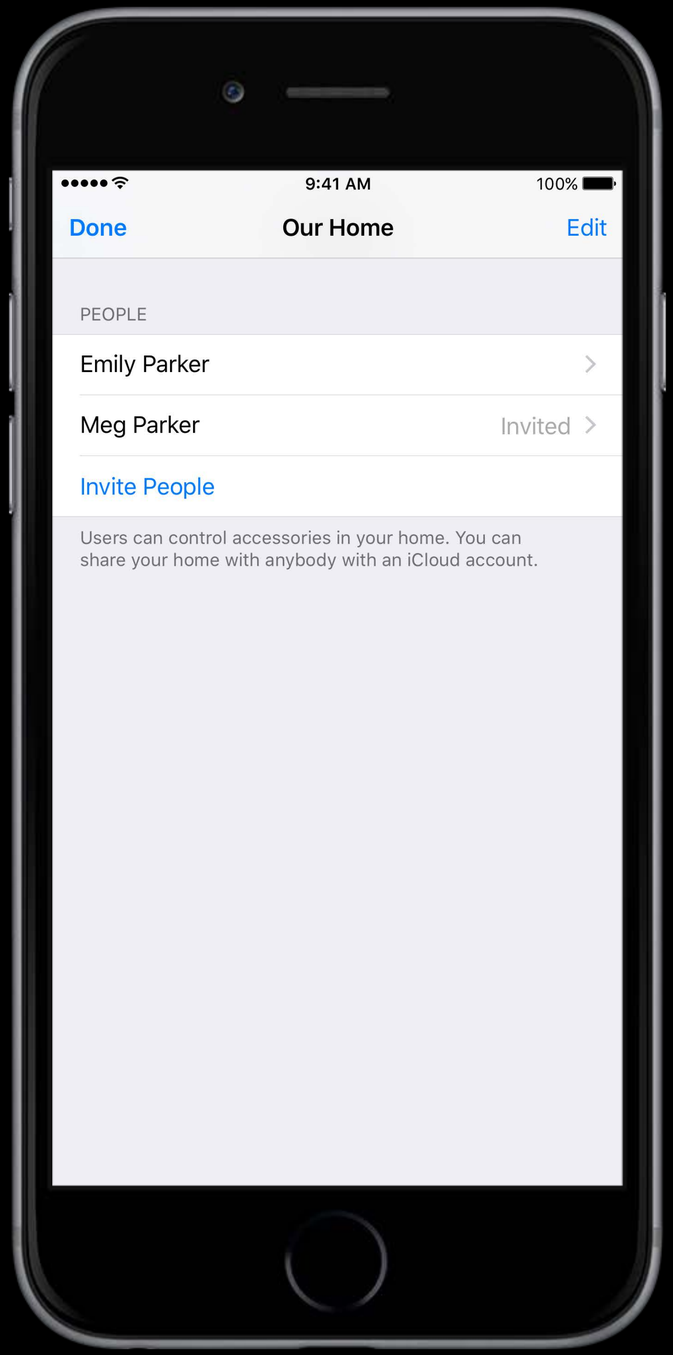
NEW

```
extension HMHome {
    func manageUsersWithCompletionHandler(completion: (NSError?) -> Void)
}
```

User Management

NEW

```
extension HMHome {
    func manageUsersWithCompletionHandler(completion: (NSError?) -> Void)
}
```


9:41 AM
100%
Done
Our Home
Edit
PEOPLE
Emily Parker
Meg Parker
Invited
Invite People
Users can control accessories in your home. You can share your home with anybody with an iCloud account.

User Capabilities

NEW

User Capabilities

NEW

Display relevant capabilities

User Capabilities

NEW

Display relevant capabilities

Determine privileges

User Capabilities

HMHomeAccessControl

```
extension HMHome {
    var currentUser: HMUser { get }
}
```

User Capabilities

HMHomeAccessControl

```
extension HMHome {
    var currentUser: HMUser { get }
}
```

User Capabilities

HMHomeAccessControl

```
extension HMHome {
    var currentUser: HMUser { get }
    func homeAccessControlForUser(user: HMUser) -> HMHomeAccessControl
}
```

User Capabilities

HMHomeAccessControl

```
extension HMHome {
    var currentUser: HMUser { get }
    func homeAccessControlForUser(user: HMUser) -> HMHomeAccessControl
}
```

User Capabilities

HMHomeAccessControl

```
extension HMHome {
    var currentUser: HMUser { get }
    func homeAccessControlForUser(user: HMUser) -> HMHomeAccessControl
}
```

```
class HMHomeAccessControl : NSObject {
    var administrator: Bool { get }
}
```

User Capabilities

HMHomeAccessControl

```
extension HMHome {
    var currentUser: HMUser { get }
    func homeAccessControlForUser(user: HMUser) -> HMHomeAccessControl
}
```

```
class HMHomeAccessControl : NSObject {
    var administrator: Bool { get }
}
```

Light off

Lock doors

Temp down

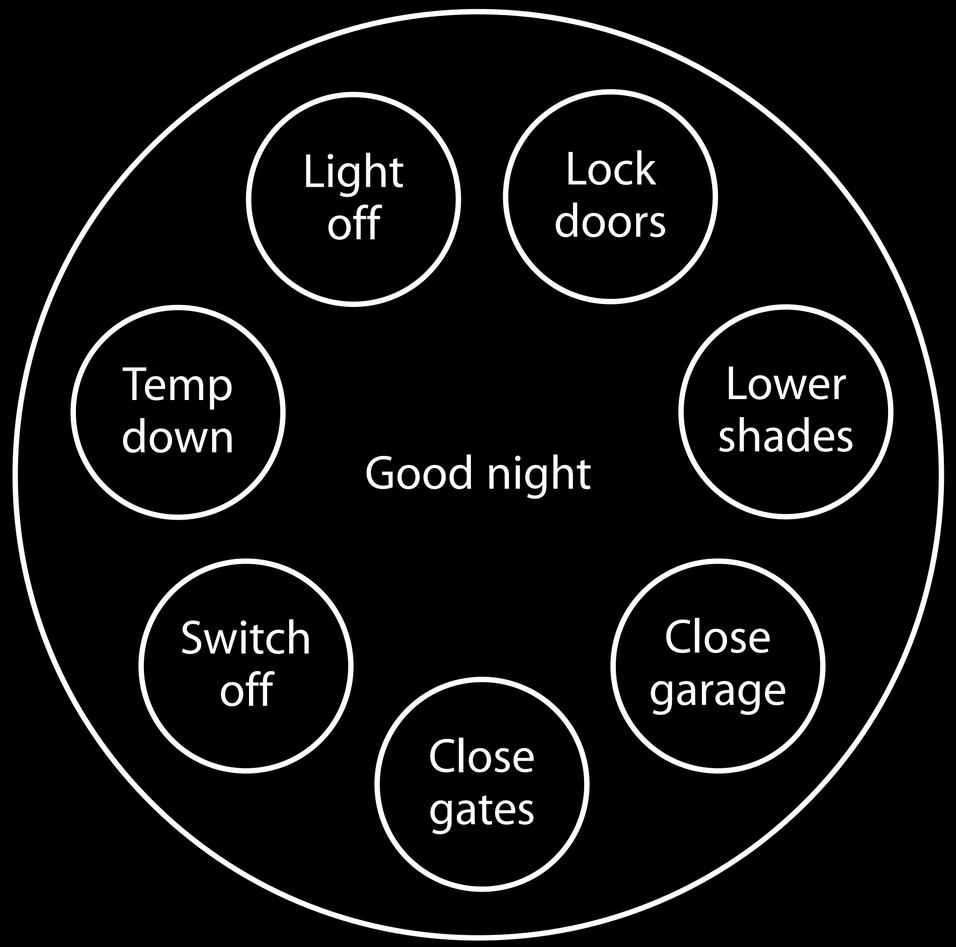
Good night

Lower shades

Switch off

Close garage

Close gates


Light off
Lock doors
Temp down
Lower shades
Good night
Switch off
Close garage
Close gates

Predefined Scenes

NEW

Predefined Scenes

NEW

Four common events

Predefined Scenes

NEW

Four common events

- Get up

Predefined Scenes

NEW

Four common events

- Get up
- Leave

Predefined Scenes

NEW

Four common events

- Get up
- Leave
- Return

Predefined Scenes

NEW

Four common events

- Get up
- Leave
- Return
- Go to bed

Predefined Scenes

NEW

Four common events

- Get up
- Leave
- Return
- Go to bed

Suggest actions

Predefined Scenes

NEW

Four common events

- Get up
- Leave
- Return
- Go to bed

Suggest actions

Cannot be deleted

Predefined Scenes

HMActionSet

```
class HMActionSet : NSObject {
    var actionSetType: String { get }
}
```

Predefined Scenes

HMActionSet

```
class HMActionSet : NSObject {
    var actionSetType: String { get }
}
```

Predefined Scenes

HMActionSet

```
class HMActionSet : NSObject {
    var actionSetType: String { get }
}

let HMActionSetTypeWakeUp: String
let HMActionSetTypeSleep: String
let HMActionSetTypeHomeDeparture: String
let HMActionSetTypeHomeArrival: String
```

Predefined Scenes

HMActionSet

```
class HMActionSet : NSObject {
    var actionSetType: String { get }
}
```

```
let HMActionSetTypeWakeUp: String
let HMActionSetTypeSleep: String
let HMActionSetTypeHomeDeparture: String
let HMActionSetTypeHomeArrival: String
```

Predefined Scenes

HMActionSet

```
class HMActionSet : NSObject {
    var actionSetType: String { get }
}

let HMActionSetTypeWakeUp: String
let HMActionSetTypeSleep: String
let HMActionSetTypeHomeDeparture: String
let HMActionSetTypeHomeArrival: String
let HMActionSetTypeUserDefined: String
```

Predefined Scenes

HMActionSet

```
class HMActionSet : NSObject {
    var actionSetType: String { get }
}

let HMActionSetTypeWakeUp: String
let HMActionSetTypeSleep: String
let HMActionSetTypeHomeDeparture: String
let HMActionSetTypeHomeArrival: String
let HMActionSetTypeUserDefined: String
```

Predefined Scenes

HMHome

```
extension HMHome {
    var actionSets: [HMActionSet] { get }
}
```

Predefined Scenes

HMHome

```
extension HMHome {
    var actionSets: [HMActionSet] { get }
}
```

Predefined Scenes

HMHome

```
extension HMHome {
    var actionSets: [HMActionSet] { get }

    func builtinActionSetOfType(actionSetType: String) -> HMActionSet?
}
```

Predefined Scenes

HMHome

```
extension HMHome {
    var actionSets: [HMActionSet] { get }

    func builtinActionSetOfType(actionSetType: String) -> HMActionSet?
}
```

Scenes and Siri

Scenes and Siri

Siri understands the names of scenes

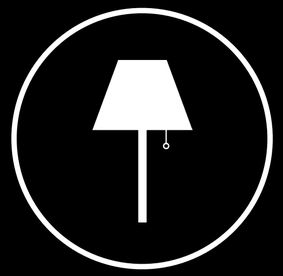
Scenes and Siri

Siri understands the names of scenes

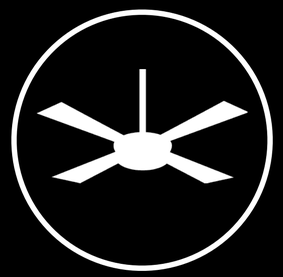
Speaking the name of scene executes it

Acme Ultra
A984CCD90714

Acme Deluxe
4C6B8F191C84



Acme Ultra
A984CCD90714



Acme Deluxe
4C6B8F191C84

Accessory Category

NEW

Accessory Category

NEW

Available during setup

Accessory Category

NEW

Available during setup

Specifies primary category of the accessory

Accessory Category

HMAccessoryCategory

```
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
```

Accessory Category

HMAccessoryCategory

```
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
```

Accessory Category

HMAccessoryCategory

```
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
let HMAccessoryCategoryTypeLightbulb: String
```

Accessory Category

HMAccessoryCategory

```
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
let HMAccessoryCategoryTypeLightbulb: String
```

Accessory Category

HMAccessoryCategory

```
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
let HMAccessoryCategoryTypeLightbulb: String
let HMAccessoryCategoryTypeFan: String
```

Accessory Category

HMAccessoryCategory

```
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
let HMAccessoryCategoryTypeLightbulb: String
let HMAccessoryCategoryTypeFan: String
```

Accessory Category

HMAccessoryCategory

```
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
let HMAccessoryCategoryTypeLightbulb: String
let HMAccessoryCategoryTypeFan: String
```

```
class HMAccessory : NSObject {
    var category: HMAccessoryCategory { get }
}
```

Accessory Category

HMAccessoryCategory

```
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
let HMAccessoryCategoryTypeLightbulb: String
let HMAccessoryCategoryTypeFan: String
```

```
class HMAccessory : NSObject {
    var category: HMAccessoryCategory { get }
}
```

HomeKit and Apple Watch

NEW

HomeKit and Apple Watch

NEW

HomeKit available on watchOS

HomeKit and Apple Watch

NEW

HomeKit available on watchOS

Home data is mirrored on Apple Watch

HomeKit and Apple Watch

NEW

HomeKit available on watchOS

Home data is mirrored on Apple Watch

Capabilities

- View homes

HomeKit and Apple Watch

NEW

HomeKit available on watchOS

Home data is mirrored on Apple Watch

Capabilities

- View homes
- Control accessories

HomeKit and Apple Watch

NEW

HomeKit available on watchOS

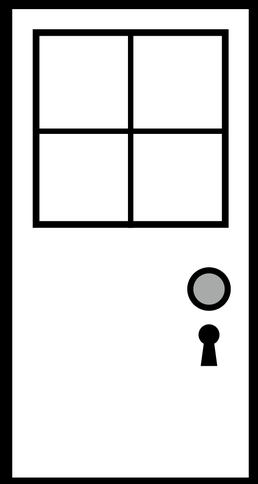
Home data is mirrored on Apple Watch

Capabilities

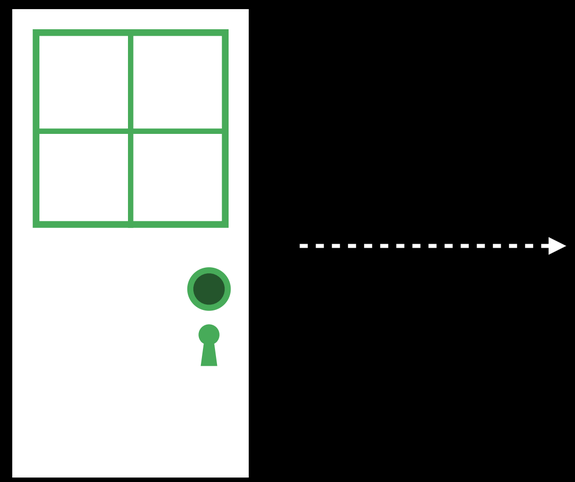
- View homes
- Control accessories
- Execute scenes

Triggers

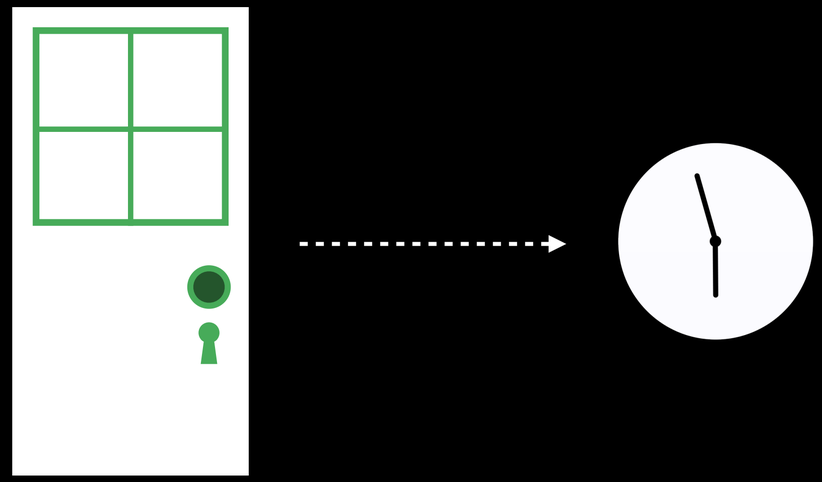
Triggers



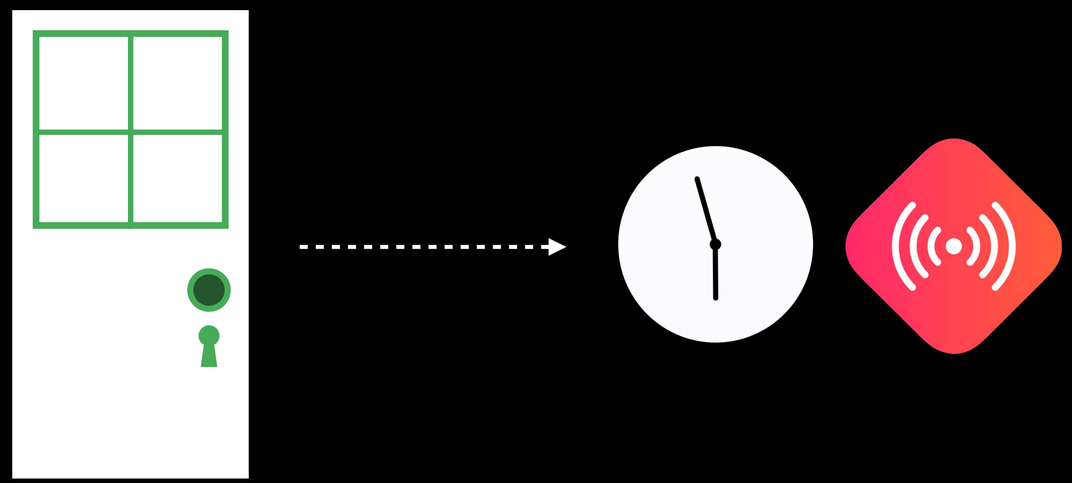
Triggers



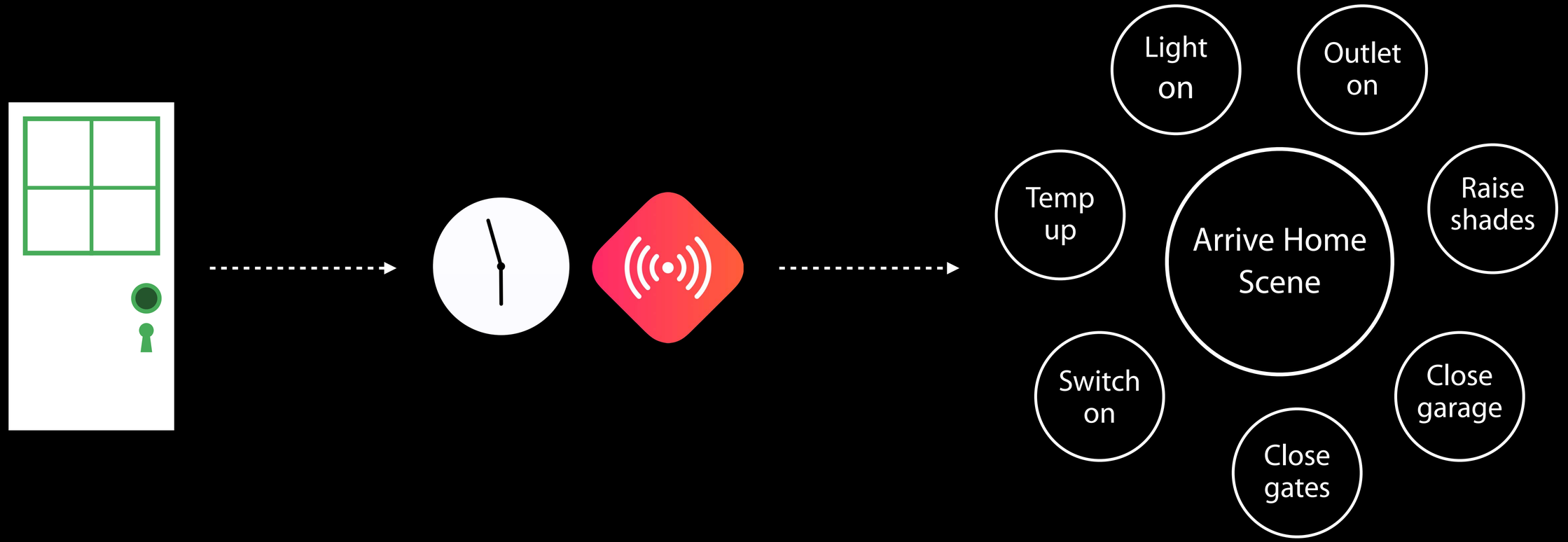
Triggers



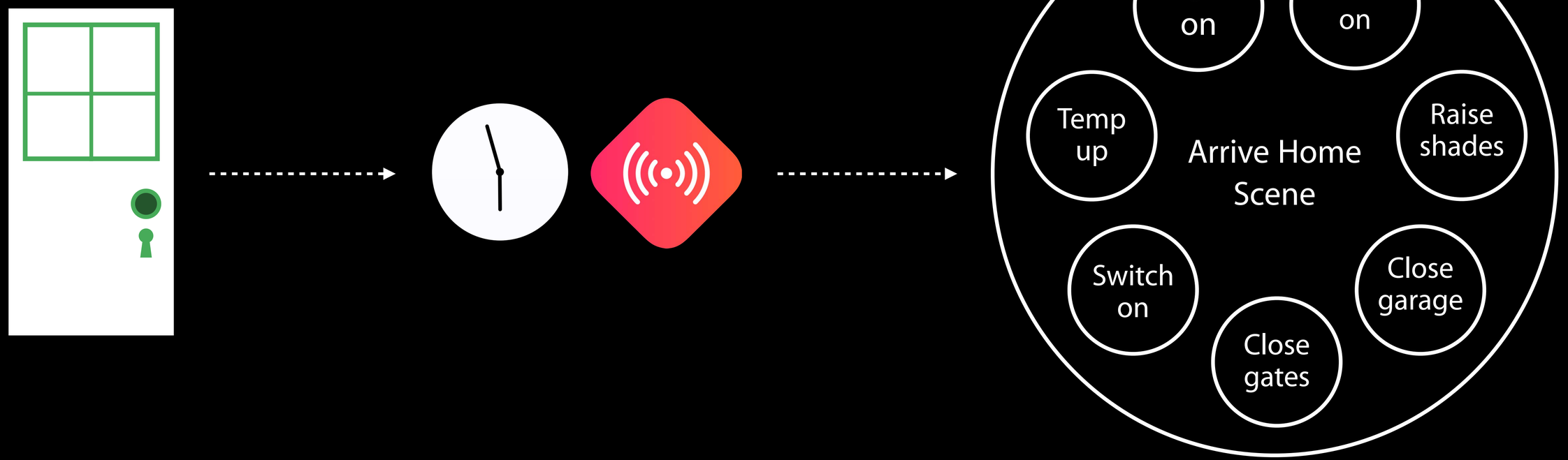
Triggers



Triggers


Light on
Outlet on
Temp up
Arrive Home Scene
Raise shades
Switch on
Close garage
Close gates

Triggers


Light on
Outlet on
Temp up
Raise shades
Arrive Home Scene
Switch on
Close garage
Close gates

Event Triggers

NEW

Event Triggers

Events

NEW

Events activate a trigger

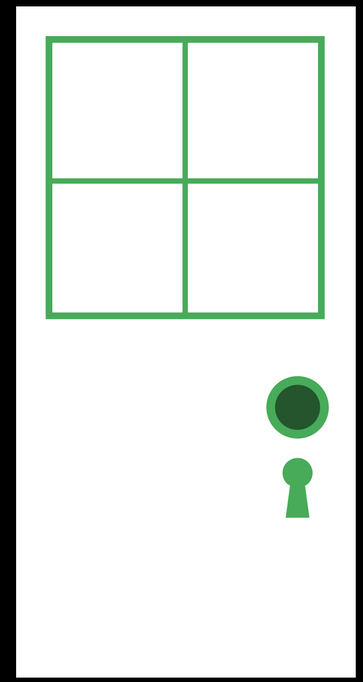
Event Triggers

Events

NEW

Events activate a trigger

- State of an accessory



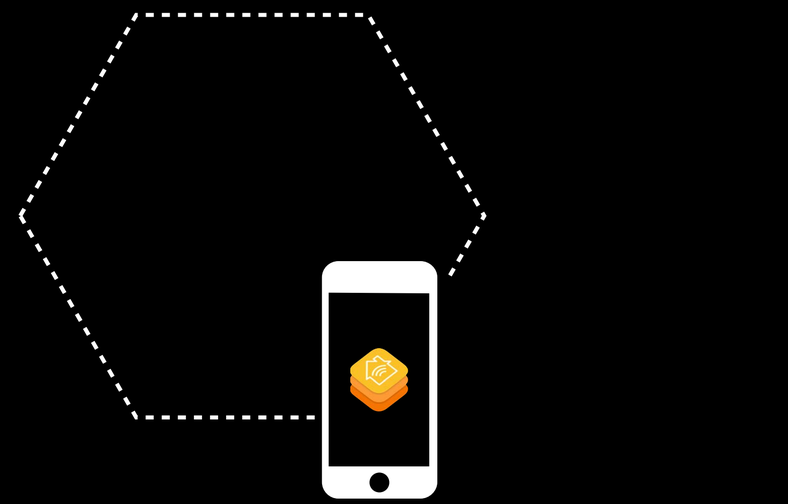
Event Triggers

Events

NEW

Events activate a trigger

- State of an accessory
- Geofence



Event Triggers

HMEvent

```
class HMCharacteristicEvent : HMEvent {
    init(characteristic: HMCharacteristic, triggerValue: NSCopying)
}
```

Event Triggers

HMEvent

```
class HMCharacteristicEvent : HMEvent {
    init(characteristic: HMCharacteristic, triggerValue: NSCopying)
}
```

Event Triggers

HMEvent

```
class HMCharacteristicEvent : HMEvent {
    init(characteristic: HMCharacteristic, triggerValue: NSCopying)
}

class HMLocationEvent : HMEvent {
    init(region: CLRegion)
}
```

Event Triggers

HMEvent

```
class HMCharacteristicEvent : HMEvent {
    init(characteristic: HMCharacteristic, triggerValue: NSCopying)
}
```

```
class HMLocationEvent : HMEvent {
    init(region: CLRegion)
}
```

Event Triggers

HMCharacteristicEvent

```
let frontDoorUnlockedEvent = HMCharacteristicEvent(characteristic:
            frontDoorCurrentStateCharacteristic,
            triggerValue:HMCharacteristicValue.DoorStateOpen)
```

Event Triggers

HMCharacteristicEvent

```
let frontDoorUnlockedEvent = HMCharacteristicEvent(characteristic:
                frontDoorCurrentStateCharacteristic,
                triggerValue:HMCharacteristicValue.DoorStateOpen)
```

Event Triggers

Conditions

Event Triggers

Conditions

Gate execution of scenes

Event Triggers

Conditions

Gate execution of scenes

- Time-based

Event Triggers

Conditions

Gate execution of scenes

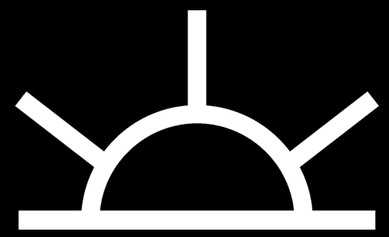
- Time-based
- State of an accessory

Event Triggers

Conditions

Gate execution of scenes

- Time-based
- State of an accessory
- Significant events
  - Sunrise
  - Sunset



Event Triggers

Conditions–time

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate

}
```

Event Triggers

Conditions–time

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate

}
```

Event Triggers

Conditions–time

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate

}
```

Event Triggers

Conditions–time

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate
    class func predicateForEvaluatingTriggerOccurringAfterDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate
}
```

Event Triggers

Conditions–time

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringOnDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate
}
```

Event Triggers

Conditions–time

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringOnDateWithComponents
                (dateComponents: NSDateComponents) -> NSPredicate
}
```

Event Triggers

Conditions–time

Event Triggers

Conditions–time

```
let before6PM  = NSDateComponents()
before6PM.hour = 18
```

Event Triggers

Conditions–time

```
let before6PM  = NSDateComponents()
before6PM.hour = 18

let before6PMPredicate = HMEventTrigger.
predicateForEvaluatingTriggerOccurringBeforeDateWithComponents(before6PM)
```

Event Triggers

Conditions–time

```
let before6PM  = NSDateComponents()
before6PM.hour = 18
```

```
let before6PMPredicate = HMEventTrigger.
predicateForEvaluatingTriggerOccurringBeforeDateWithComponents(before6PM)
```

Event Triggers

Conditions–accessory state

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerWithCharacteristic
                (characteristic: HMCharacteristic,
                 matchingValue: AnyObject) -> NSPredicate
}
```

Event Triggers

Conditions–accessory state

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerWithCharacteristic
                (characteristic: HMCharacteristic,
                 matchingValue: AnyObject) -> NSPredicate
}
```

Event Triggers

Conditions–accessory state

```
let motionDetectedPredicate = HMEventTrigger.
predicateForEvaluatingTriggerWithCharacteristic(characteristic:
            frontDoorMotionSensorCharacteristic,
            matchingValue:true)
```

Event Triggers

Conditions–accessory state

```
let motionDetectedPredicate = HMEventTrigger.
predicateForEvaluatingTriggerWithCharacteristic(characteristic:
            frontDoorMotionSensorCharacteristic,
            matchingValue:true)
```

Event Triggers

Conditions–significant events in a day

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeSignificantEvent
                (significantEvent: String,
                 applyingOffset: NSDateComponents?) -> NSPredicate

}
```

Event Triggers

Conditions–significant events in a day

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeSignificantEvent
                (significantEvent: String,
                 applyingOffset: NSDateComponents?) -> NSPredicate
}
```

Event Triggers

Conditions–significant events in a day

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeSignificantEvent
                (significantEvent: String,
                 applyingOffset: NSDateComponents?) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterSignificantEvent
                (significantEvent: String,
                 applyingOffset: NSDateComponents?) -> NSPredicate
}
```

Event Triggers

Conditions–significant events in a day

```
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeSignificantEvent
                (significantEvent: String,
                 applyingOffset: NSDateComponents?) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterSignificantEvent
                (significantEvent: String,
                 applyingOffset: NSDateComponents?) -> NSPredicate
}
```

Event Triggers

Conditions–significant events in a day

```
let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String
```

Event Triggers

Conditions–significant events in a day

```
let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String
```

Event Triggers

Conditions–significant events in a day

```
let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String

let offset    = NSDateComponents()
offset.minute = -30
```

Event Triggers

Conditions–significant events in a day

```
let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String
```

```
let offset    = NSDateComponents()
offset.minute = -30
```

Event Triggers

Conditions–significant events in a day

```
let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String

let offset    = NSDateComponents()
offset.minute = -30

let thirtyMinutesBeforeSunsetPredicate =
HMEventTrigger.predicateForEvaluatingTriggerOccurringBeforeSignificantEvent
(HMSignificantEventSunset, applyingOffset:offset)
```

Event Triggers

Conditions–significant events in a day

```
let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String

let offset    = NSDateComponents()
offset.minute = -30
```

```
let thirtyMinutesBeforeSunsetPredicate =
HMEventTrigger.predicateForEvaluatingTriggerOccurringBeforeSignificantEvent
(HMSignificantEventSunset, applyingOffset:offset)
```

Event Triggers

HMEventTrigger

```
class HMEventTrigger : HMTrigger {
    init(name: String,
         events:[HMEvent],
         predicate: NSPredicate?)
}
```

Event Triggers

HMEventTrigger

```
class HMEventTrigger : HMTrigger {
    init(name: String,
         events:[HMEvent],
         predicate: NSPredicate?)
}
```

Event Triggers

HMEventTrigger

```
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates
                ([before6PMPredicate, motionDetectedPredicate])
```

Event Triggers

HMEventTrigger

```
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates
                ([before6PMPredicate, motionDetectedPredicate])
```

Event Triggers

HMEventTrigger

```
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates
                ([before6PMPredicate, motionDetectedPredicate])

let eventTrigger = HMEventTrigger("Arrived Home",
                   events:[frontDoorUnlockedEvent], predicate:predicate)
```

Event Triggers

HMEventTrigger

```
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates
                ([before6PMPredicate, motionDetectedPredicate])

let eventTrigger = HMEventTrigger("Arrived Home",
                   events:[frontDoorUnlockedEvent], predicate:predicate)
```

Event Triggers

HMEventTrigger

```
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates
                ([before6PMPredicate, motionDetectedPredicate])

let eventTrigger = HMEventTrigger("Arrived Home",
                   events:[frontDoorUnlockedEvent], predicate:predicate)
```

```
let arrivedHome = home.builtinActionSetOfType(HMActionSetTypeHomeArrival)
eventTrigger.addActionSet(arrivedHome,
                  completionHandler:(NSError?) -> Void { error in /* */ })
```

Event Triggers

HMEventTrigger

```
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates
                ([before6PMPredicate, motionDetectedPredicate])

let eventTrigger = HMEventTrigger("Arrived Home",
                   events:[frontDoorUnlockedEvent], predicate:predicate)
```

```
let arrivedHome = home.builtinActionSetOfType(HMActionSetTypeHomeArrival)
eventTrigger.addActionSet(arrivedHome,
                  completionHandler:(NSError?) -> Void { error in /* */ })
```

Demo

EventTriggers using HomeKitCatalog

Accessories

Naveen Kommareddi HomeKit Engineer

Remote Access

Remote Access

Bluetooth Low Energy

Remote Access

Bluetooth Low Energy

Accessory Categories

Remote Access

Bluetooth Low Energy

Accessory Categories

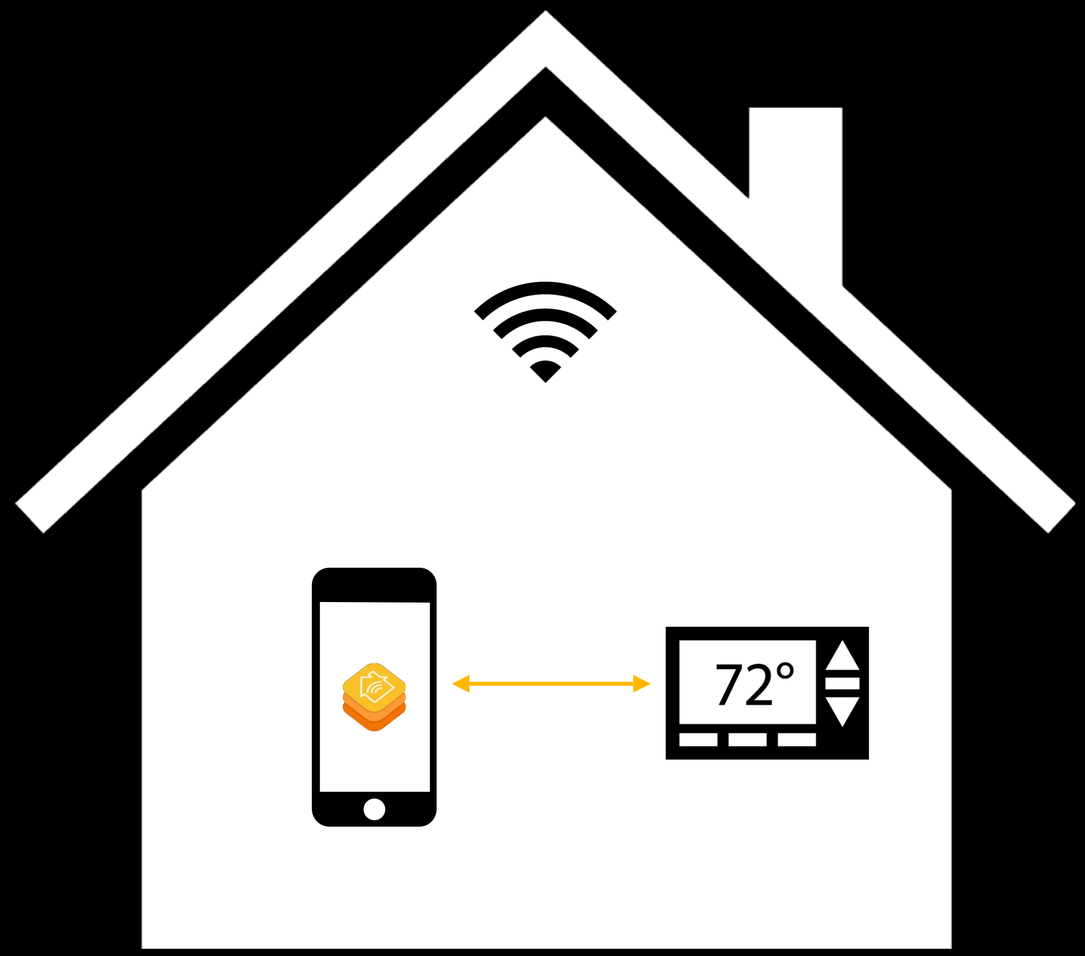
Remote Access

Remote Access

Control when away from home

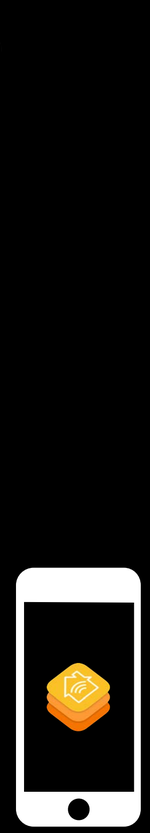
Remote Access

Control when away from home



Remote Access

Control when away from home

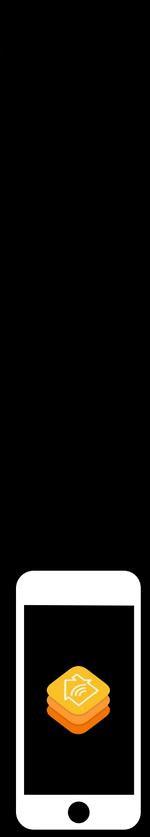


Remote Access

Control when away from home

Apple TV

- 3rd generation

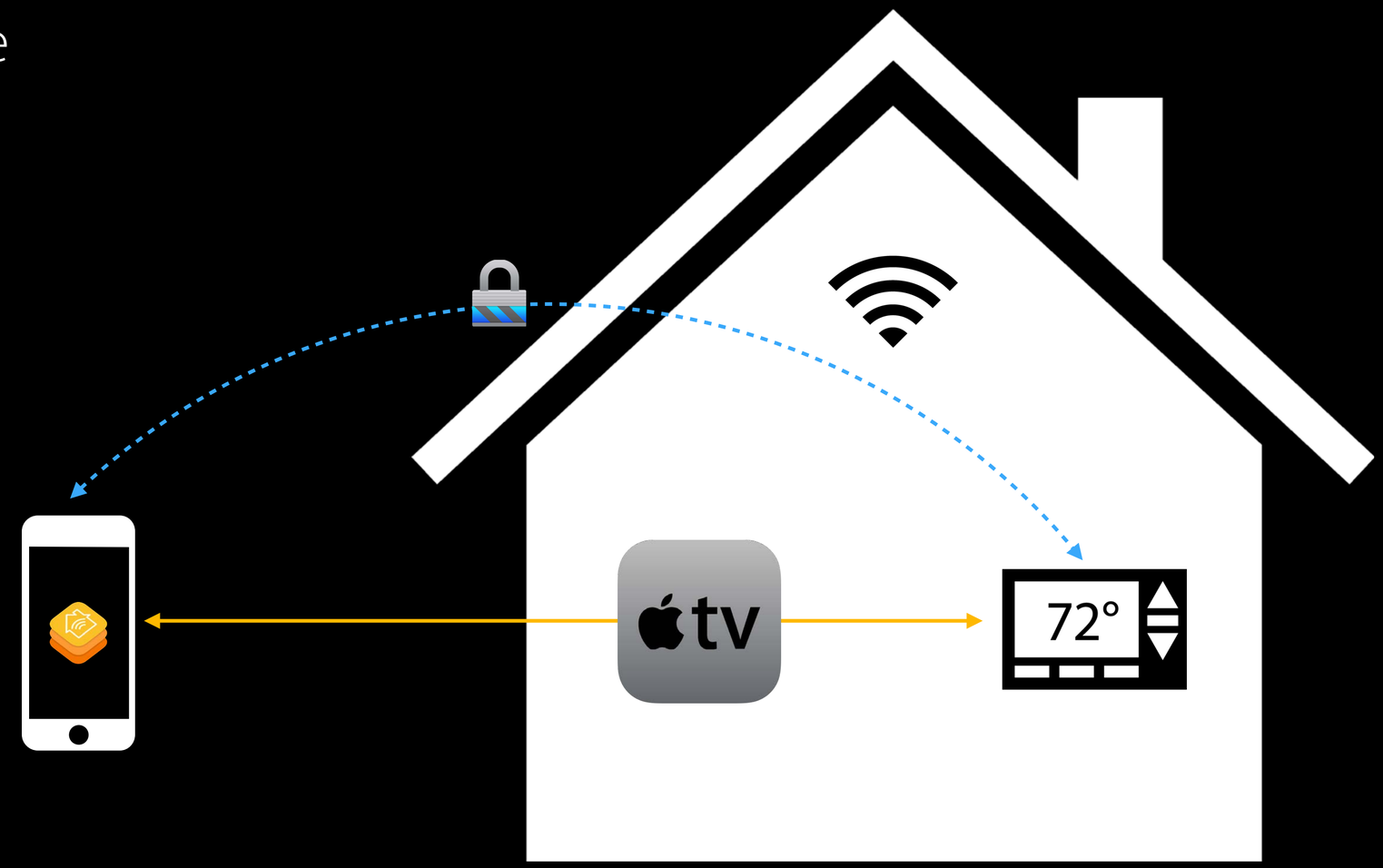


Remote Access

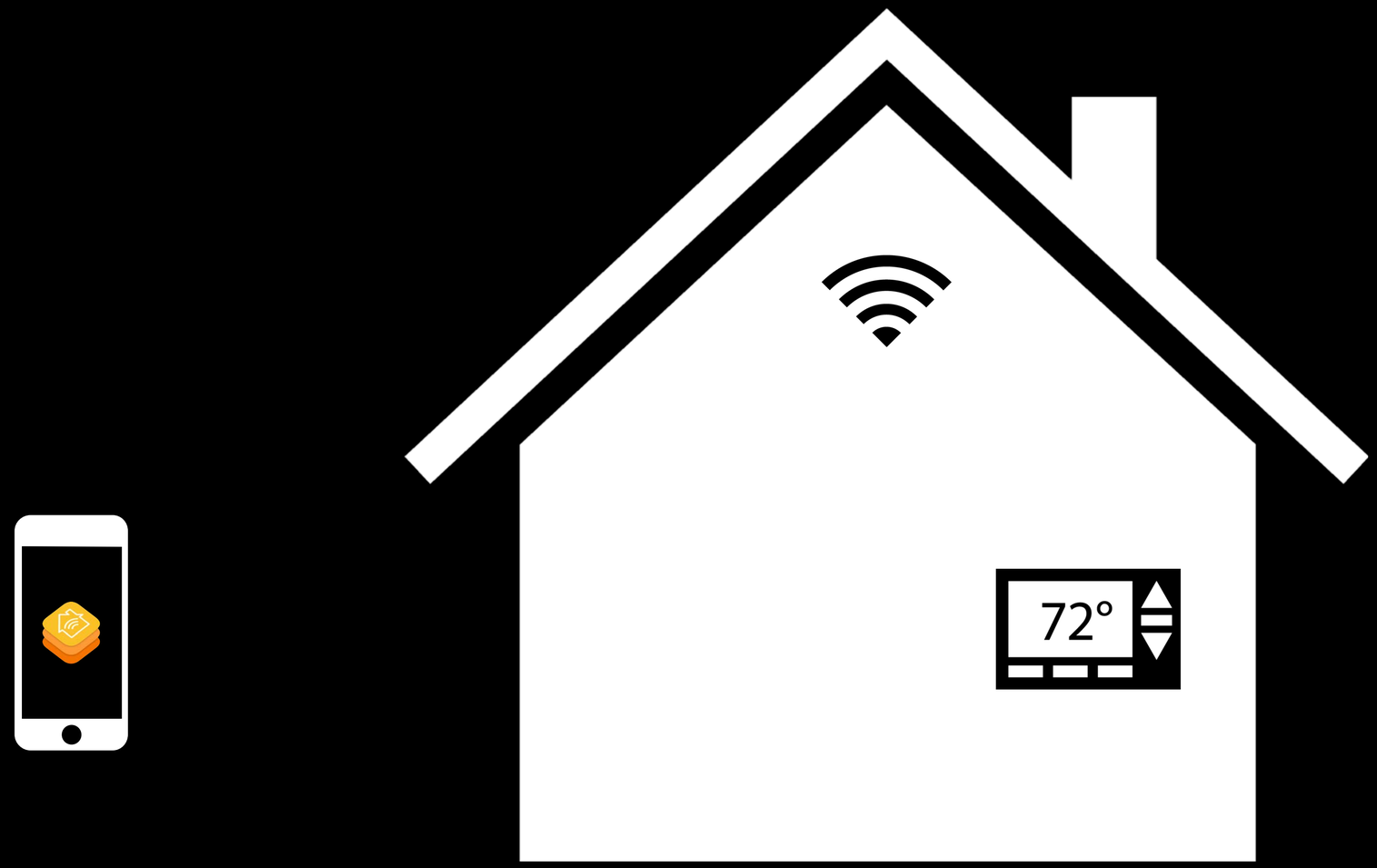
Control when away from home

Apple TV

- 3rd generation
- Sign in with Apple ID



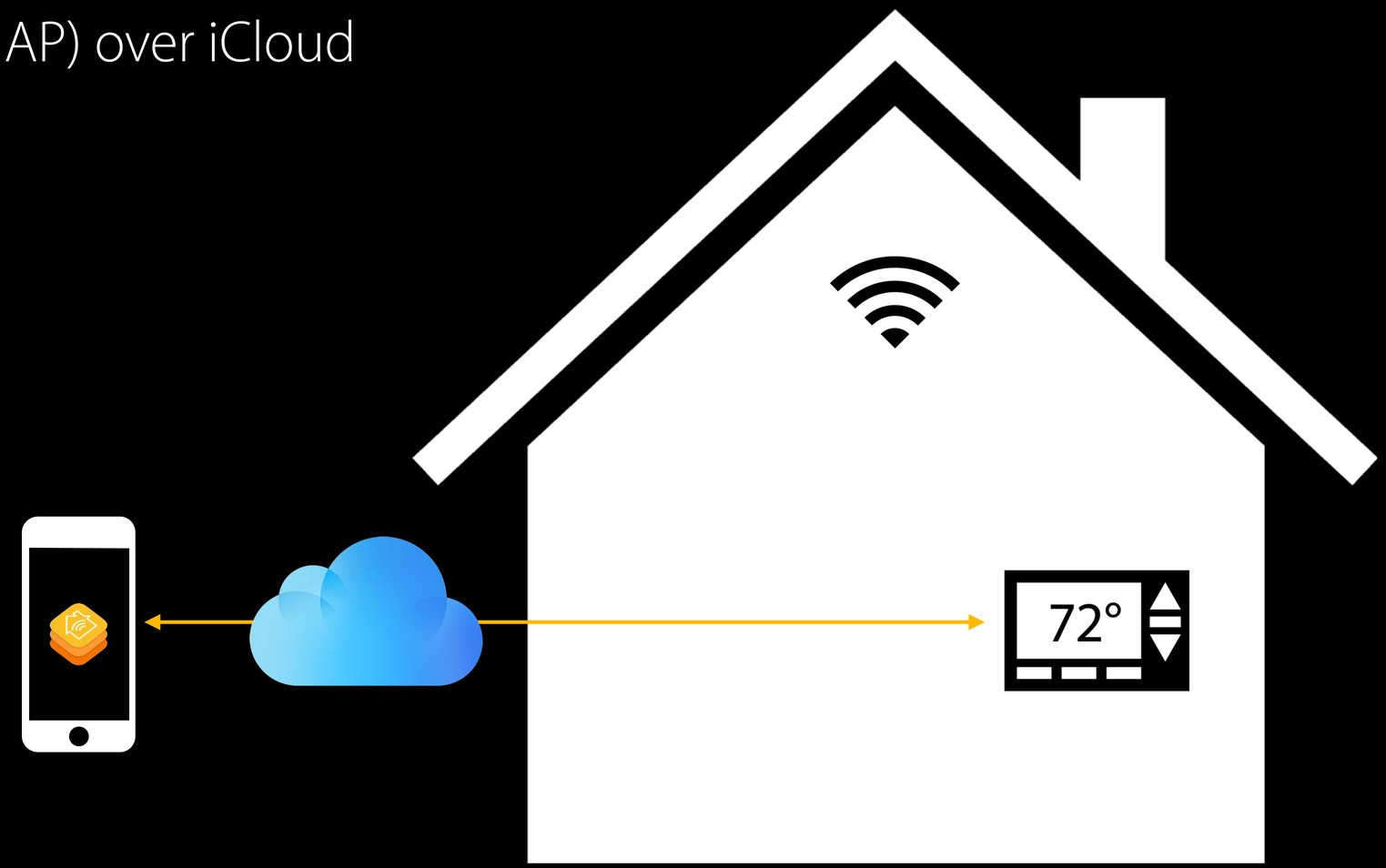
Remote Access



Remote Access

NEW

HomeKit Accessory Protocol (HAP) over iCloud

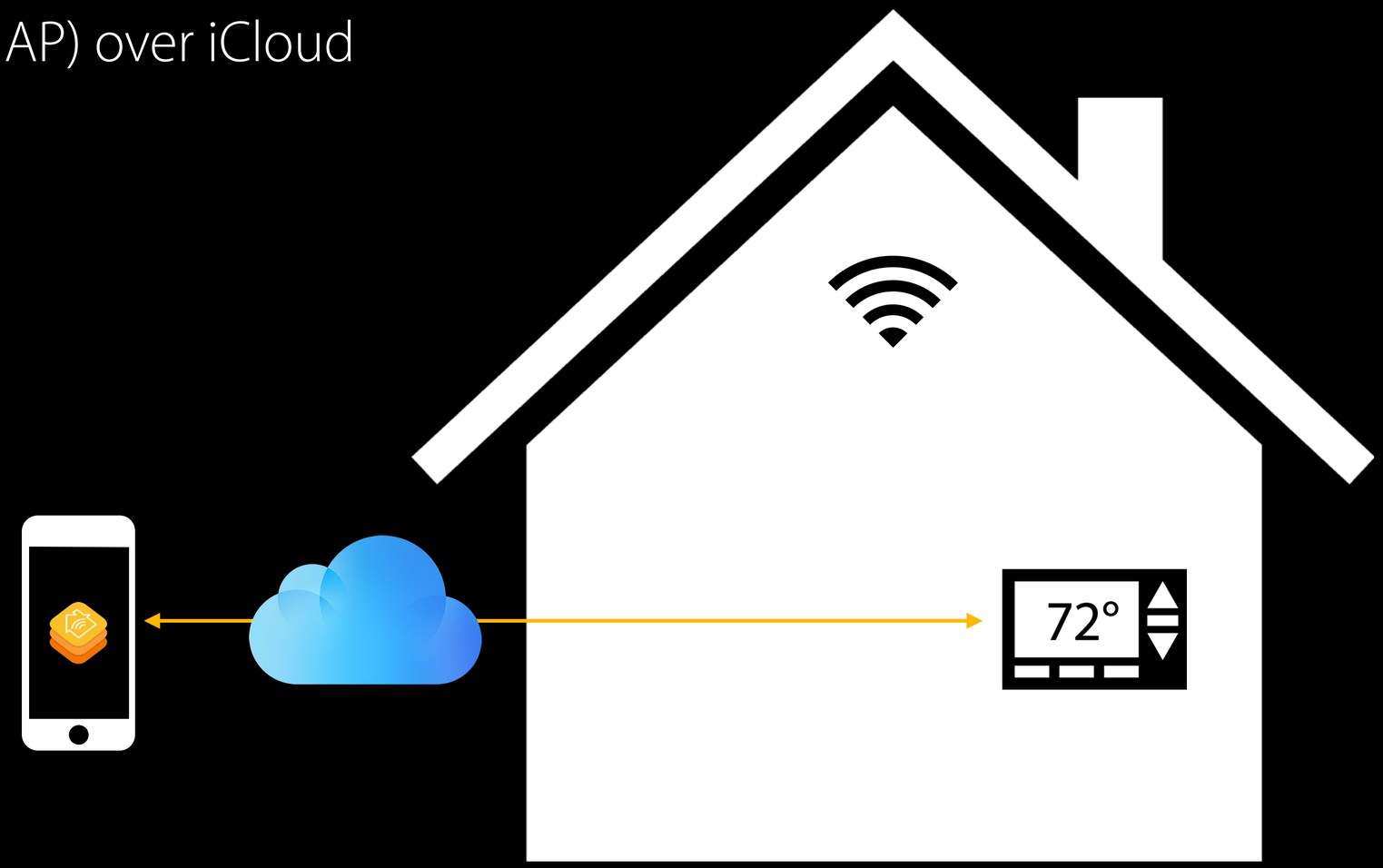


Remote Access

NEW

HomeKit Accessory Protocol (HAP) over iCloud

- Control and notifications

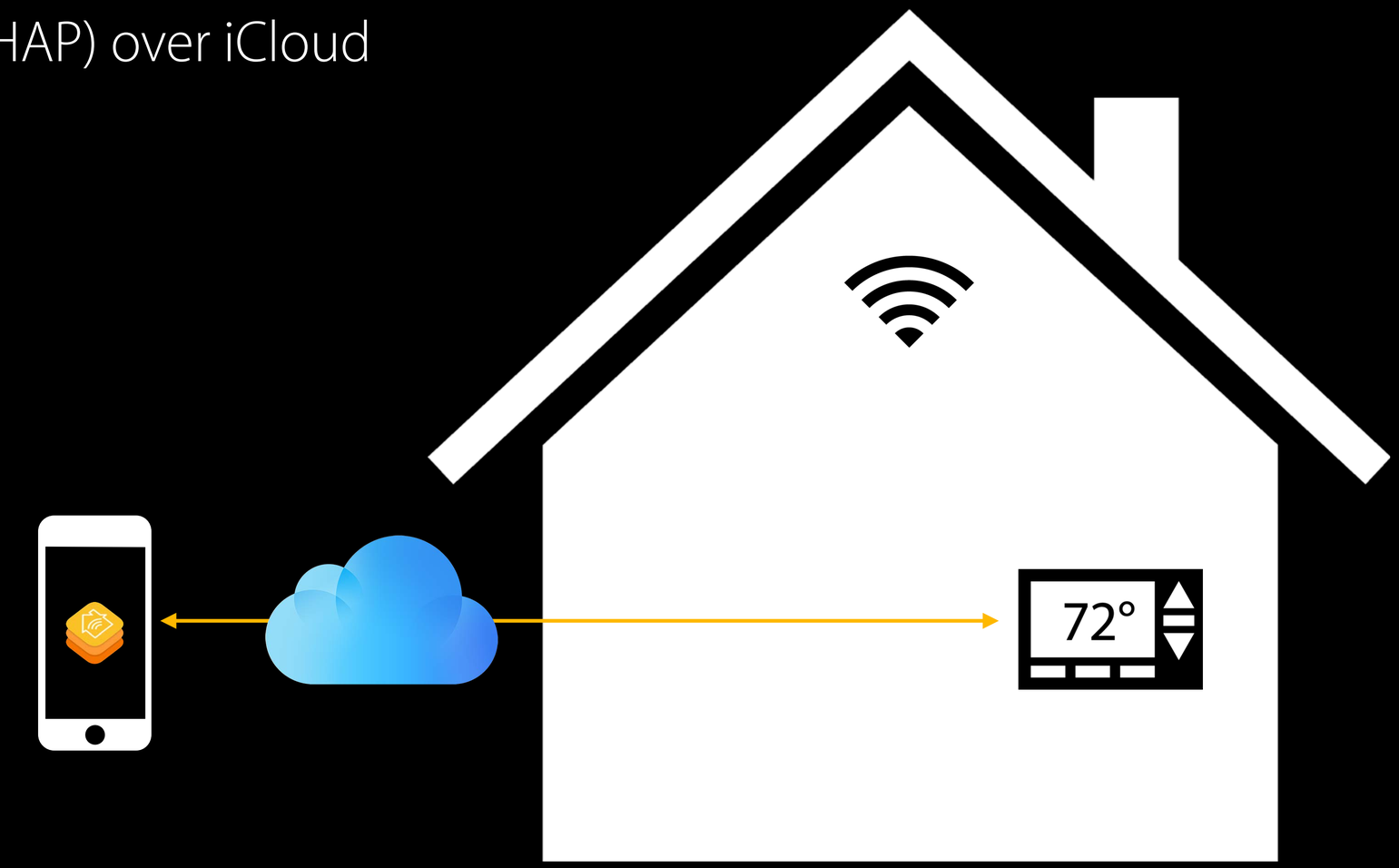


Remote Access

NEW

HomeKit Accessory Protocol (HAP) over iCloud

- Control and notifications
- Dedicated iCloud service

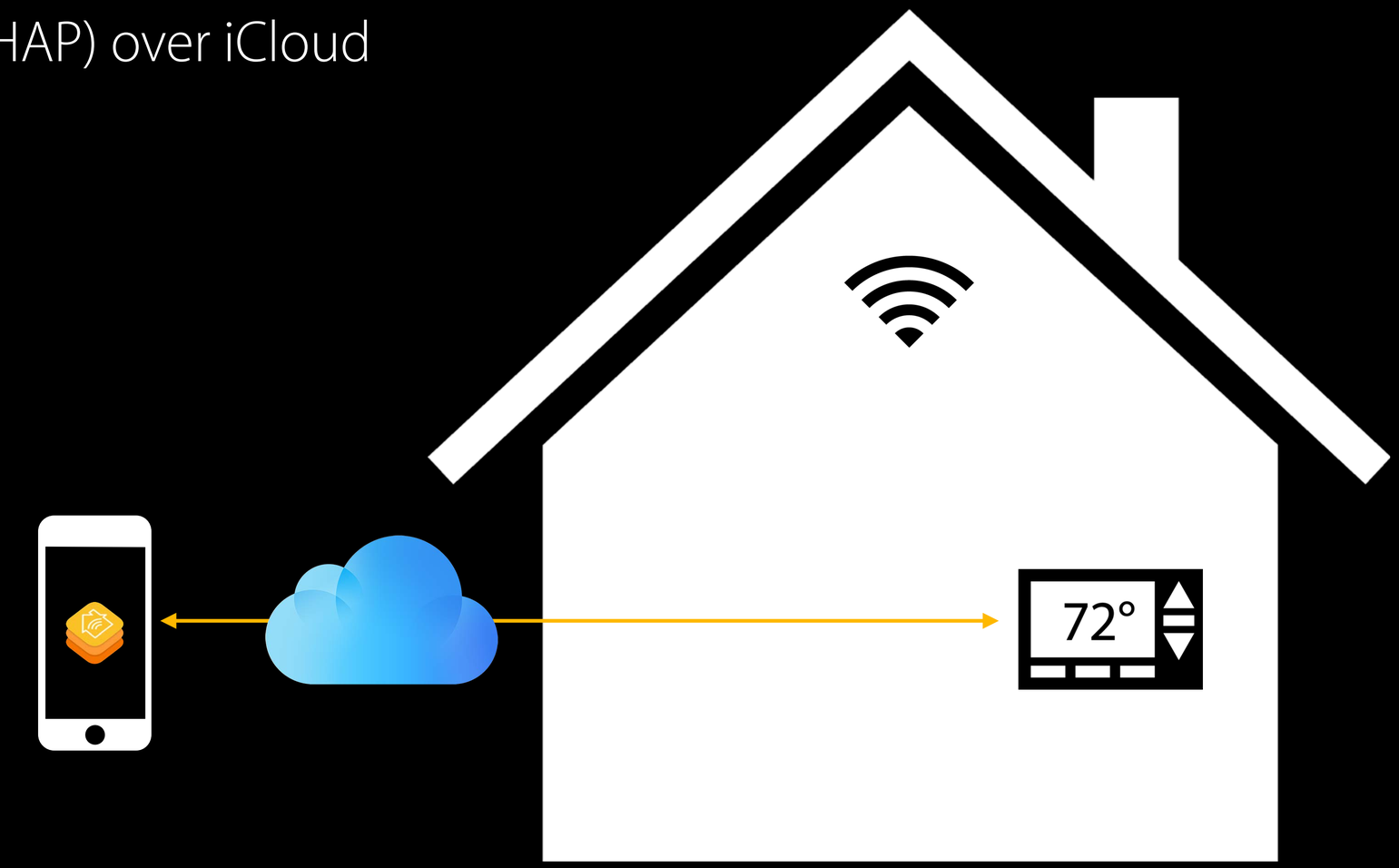


Remote Access

NEW

HomeKit Accessory Protocol (HAP) over iCloud

- Control and notifications
- Dedicated iCloud service
- Free

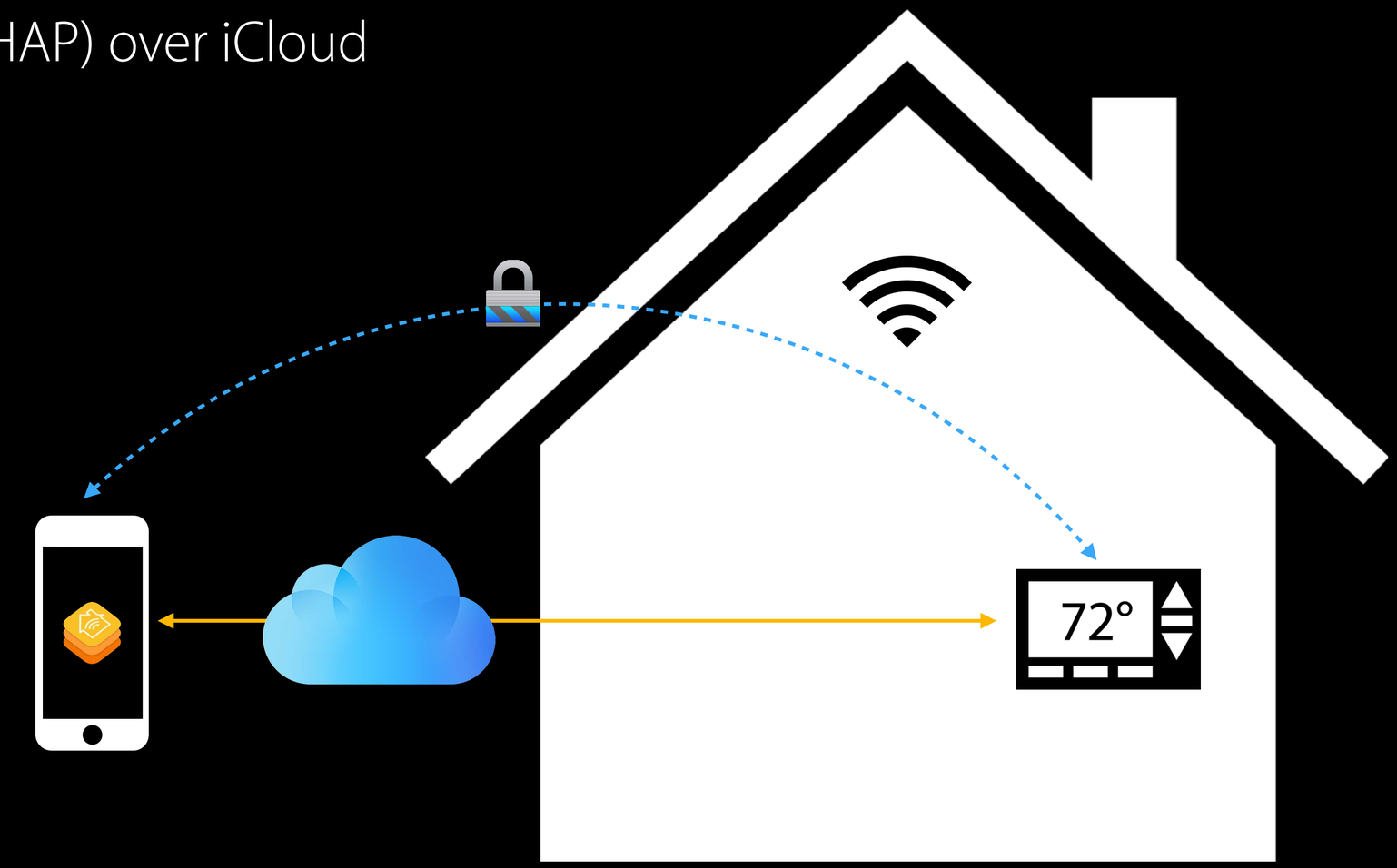


Remote Access

NEW

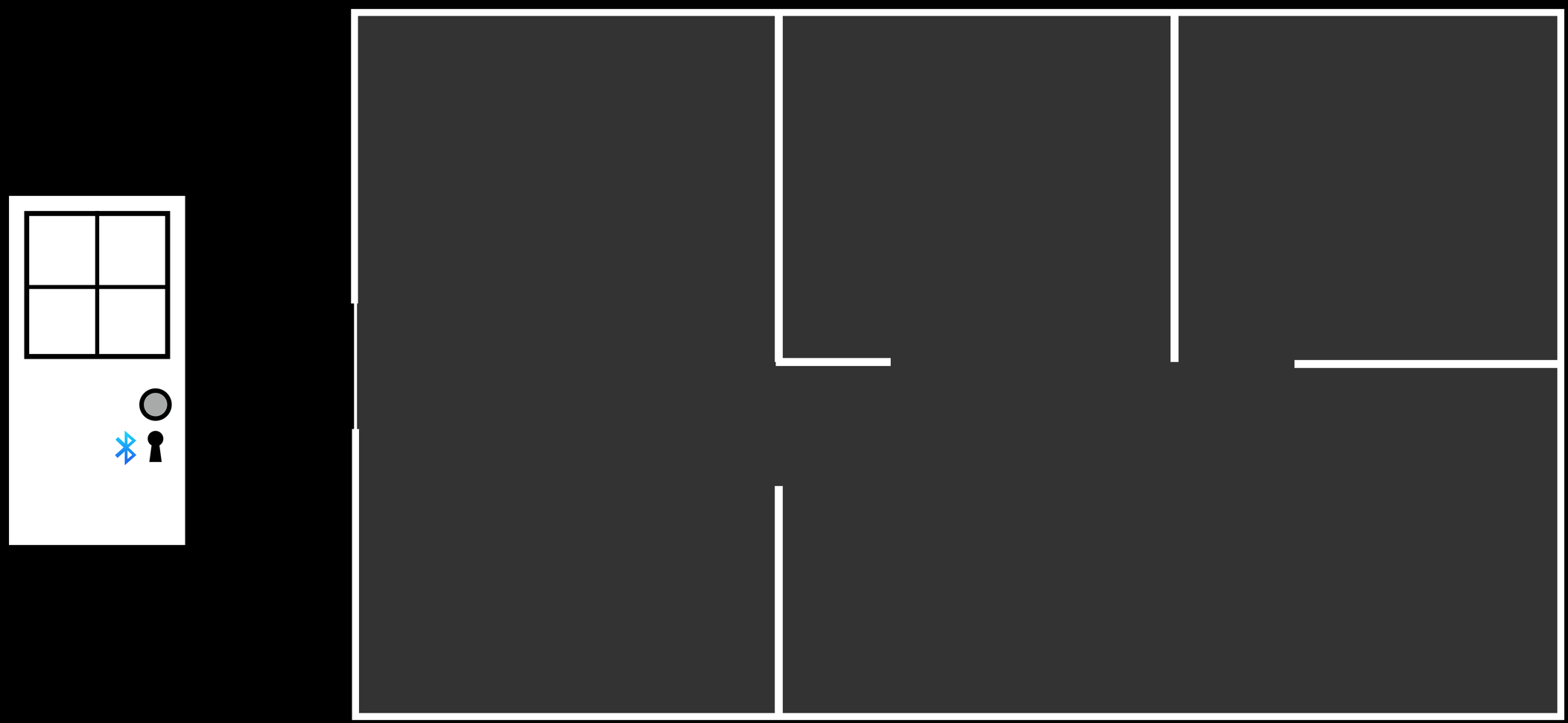
HomeKit Accessory Protocol (HAP) over iCloud

- Control and notifications
- Dedicated iCloud service
- Free
- Private and secure

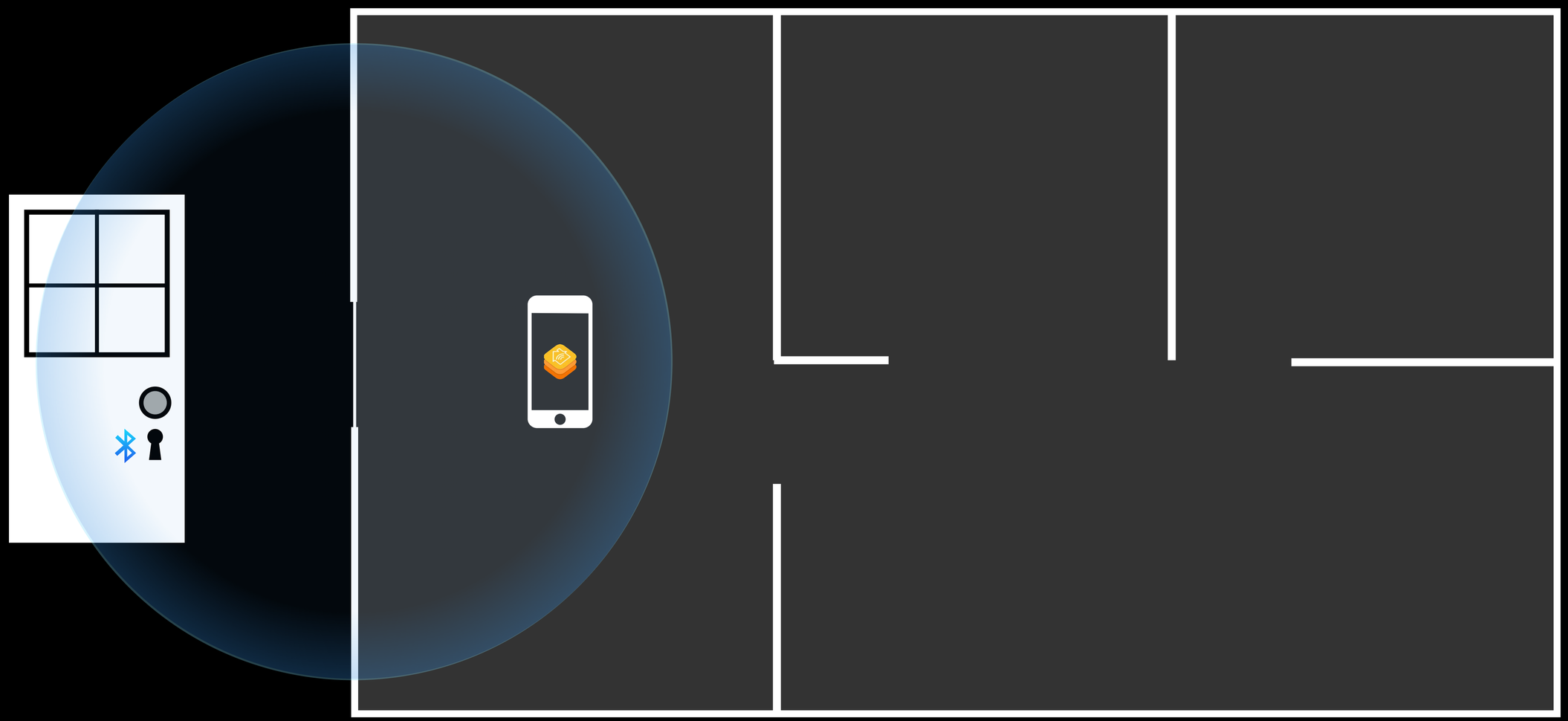


Bluetooth Low Energy

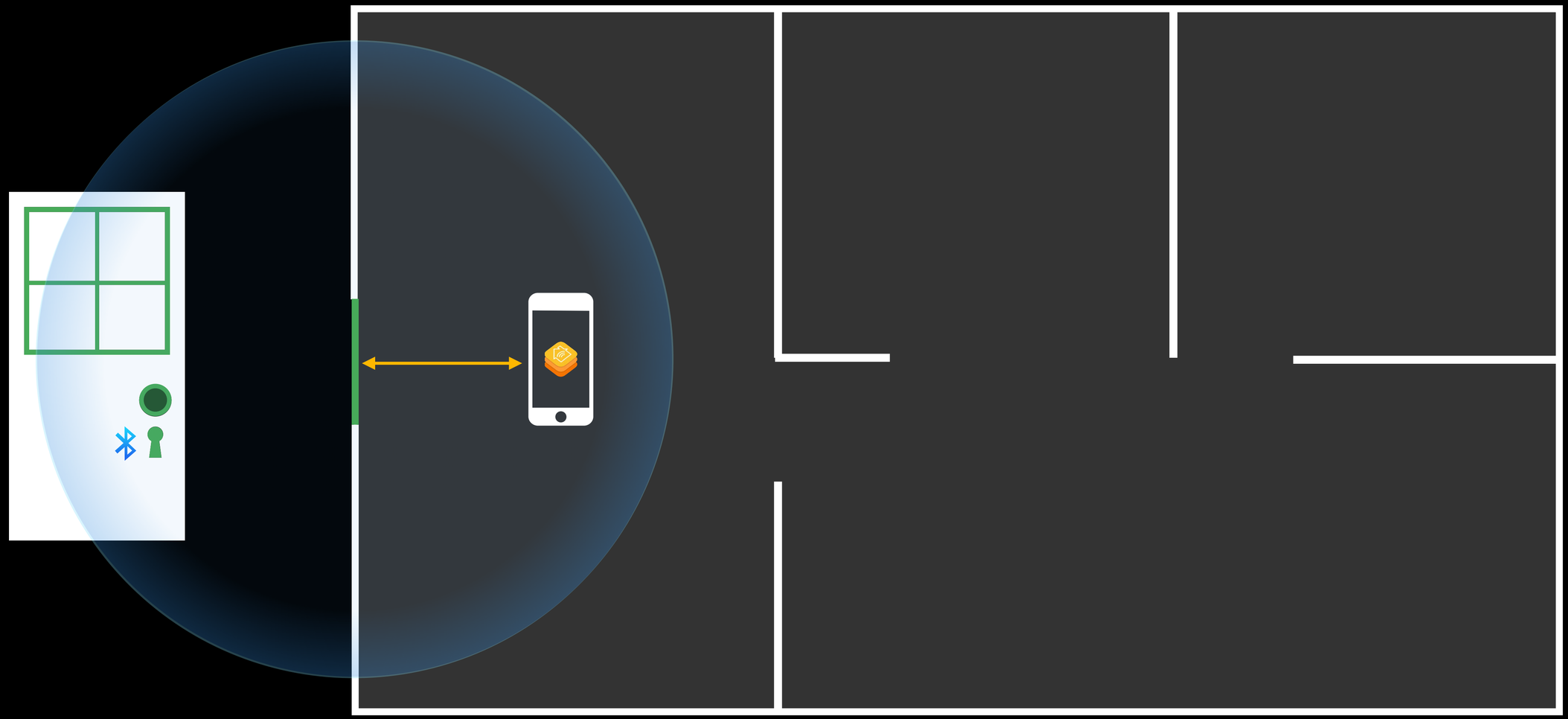
Bluetooth Low Energy



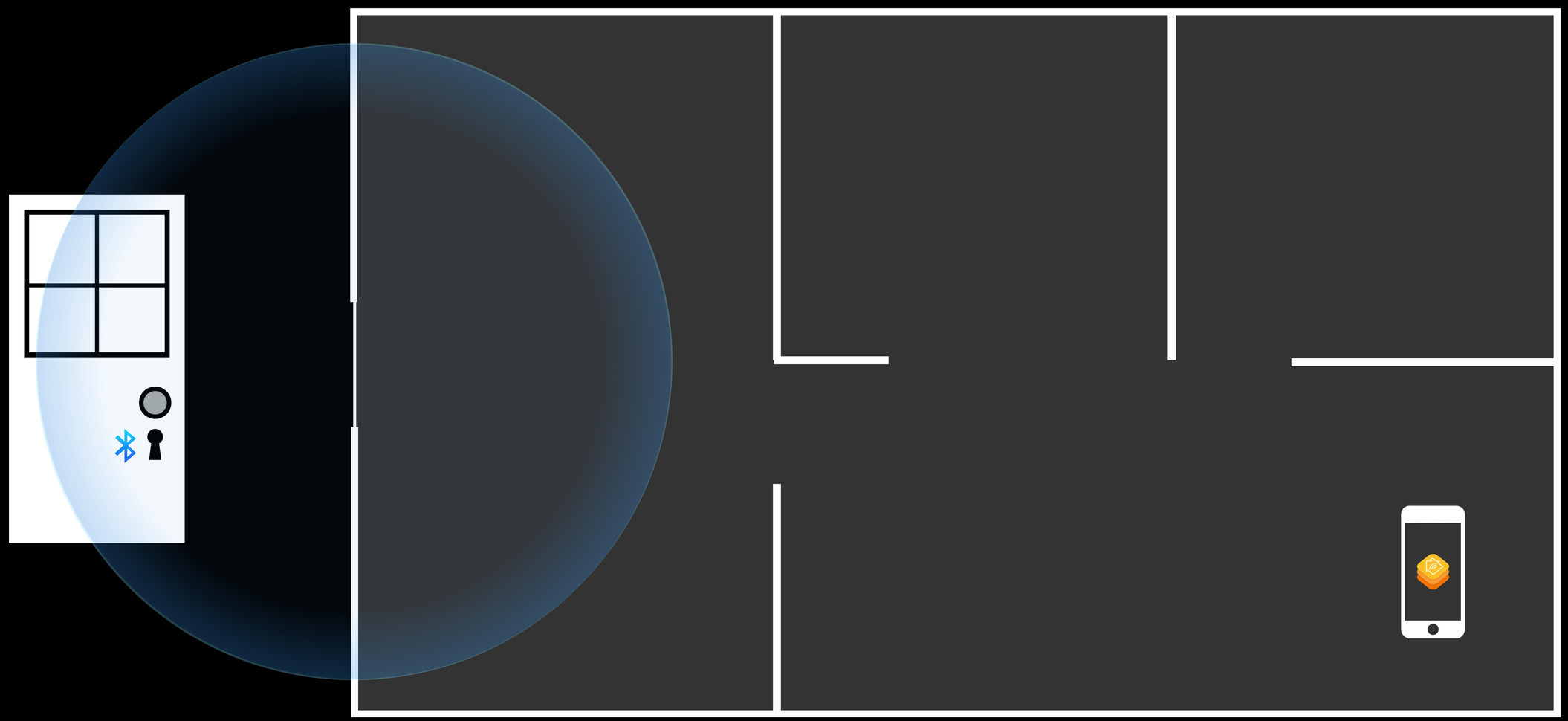
Bluetooth Low Energy



Bluetooth Low Energy

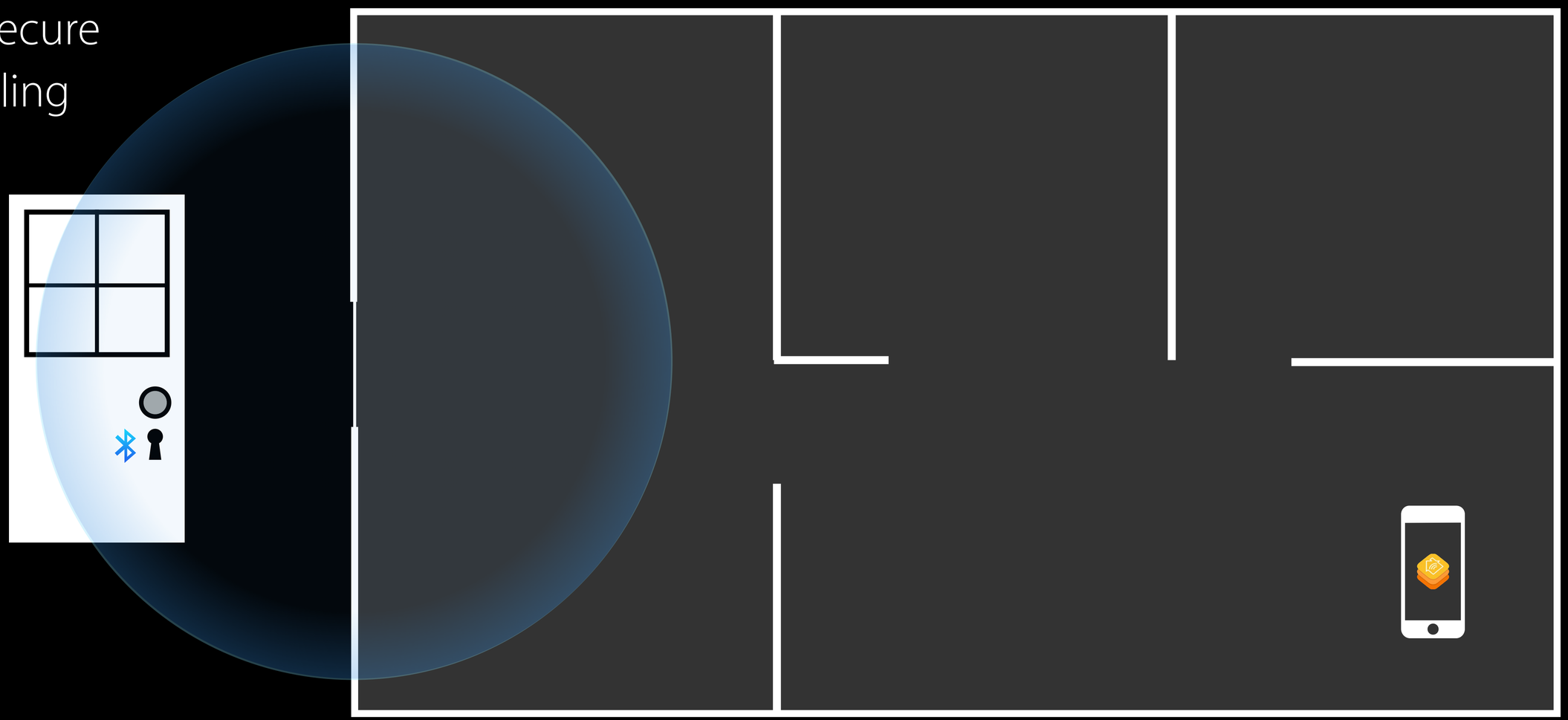


Bluetooth Low Energy



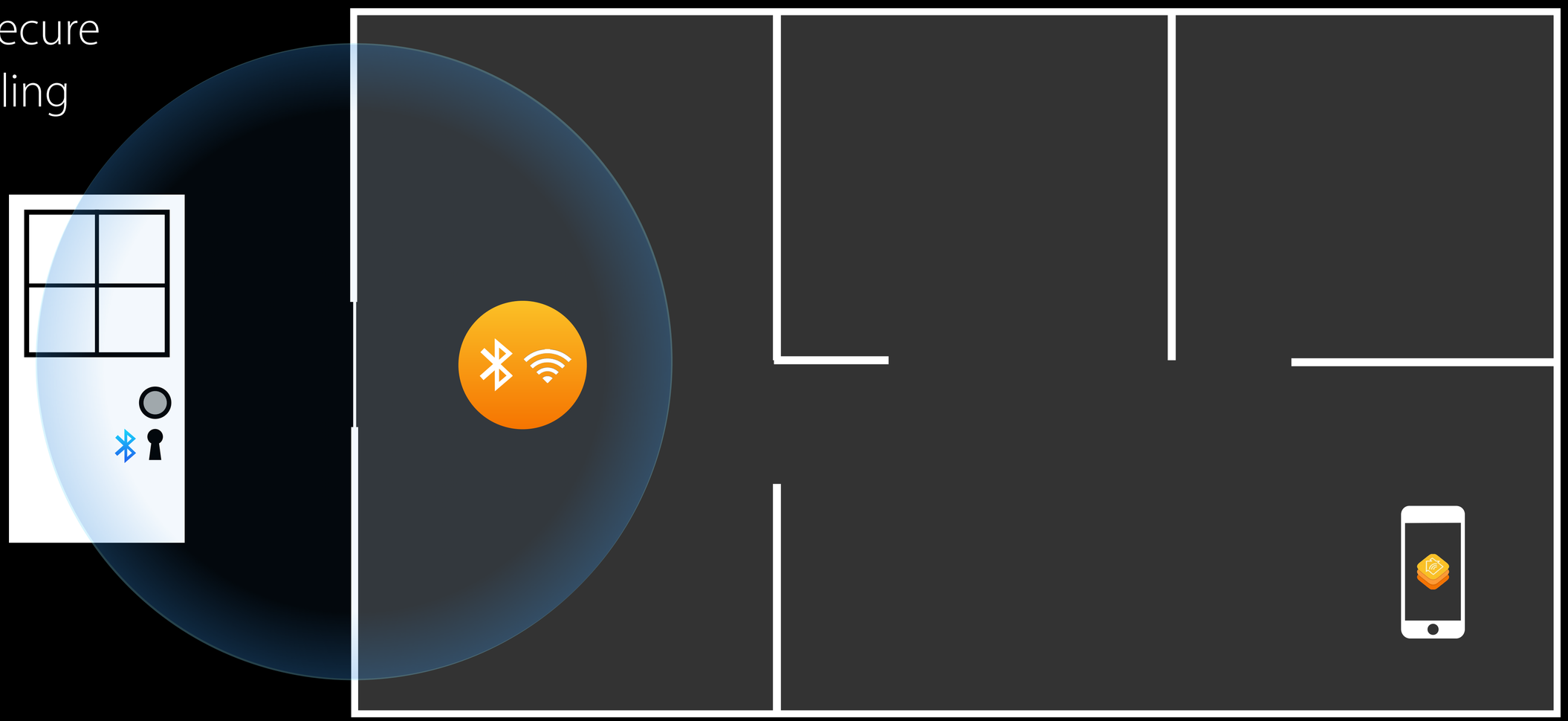
Bluetooth Low Energy

HAP secure tunneling



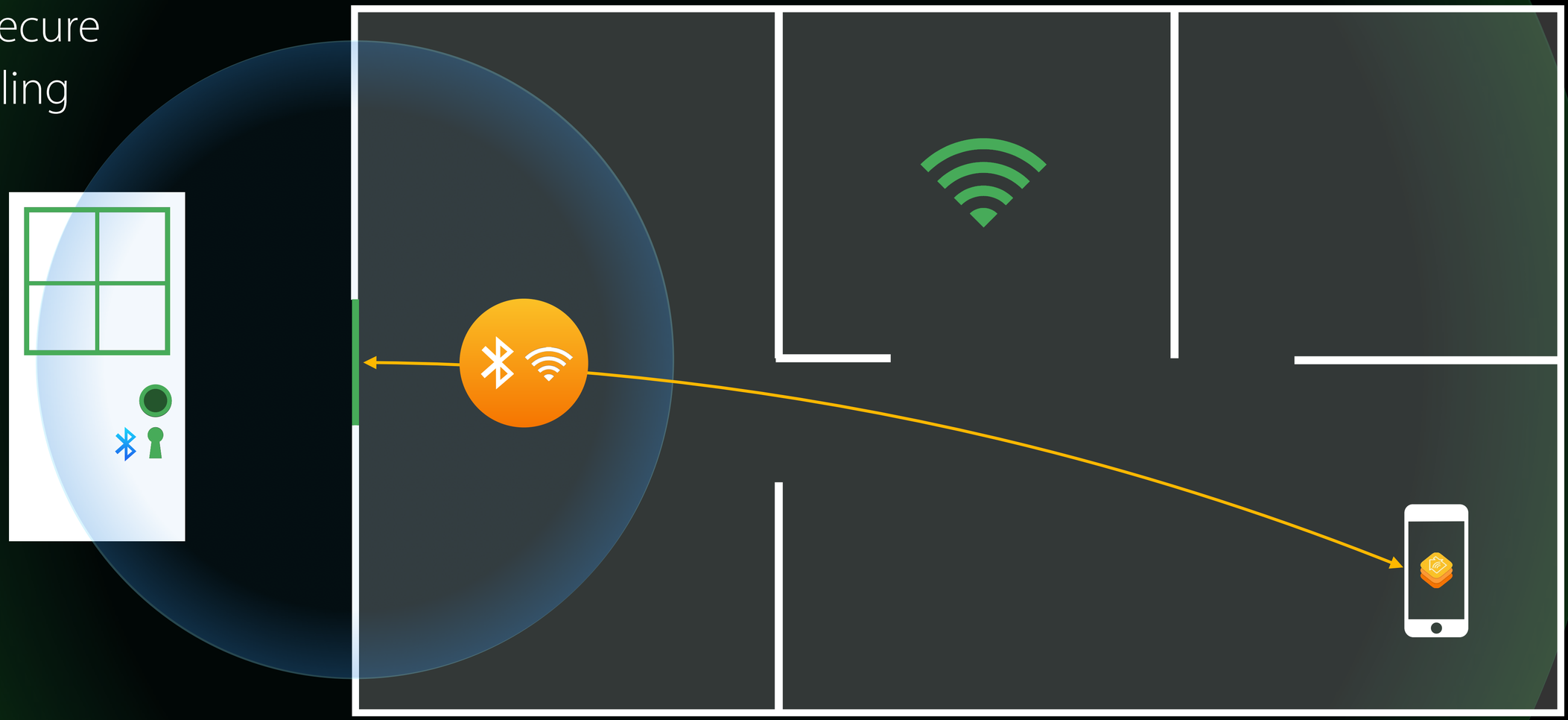
Bluetooth Low Energy

HAP secure
tunneling



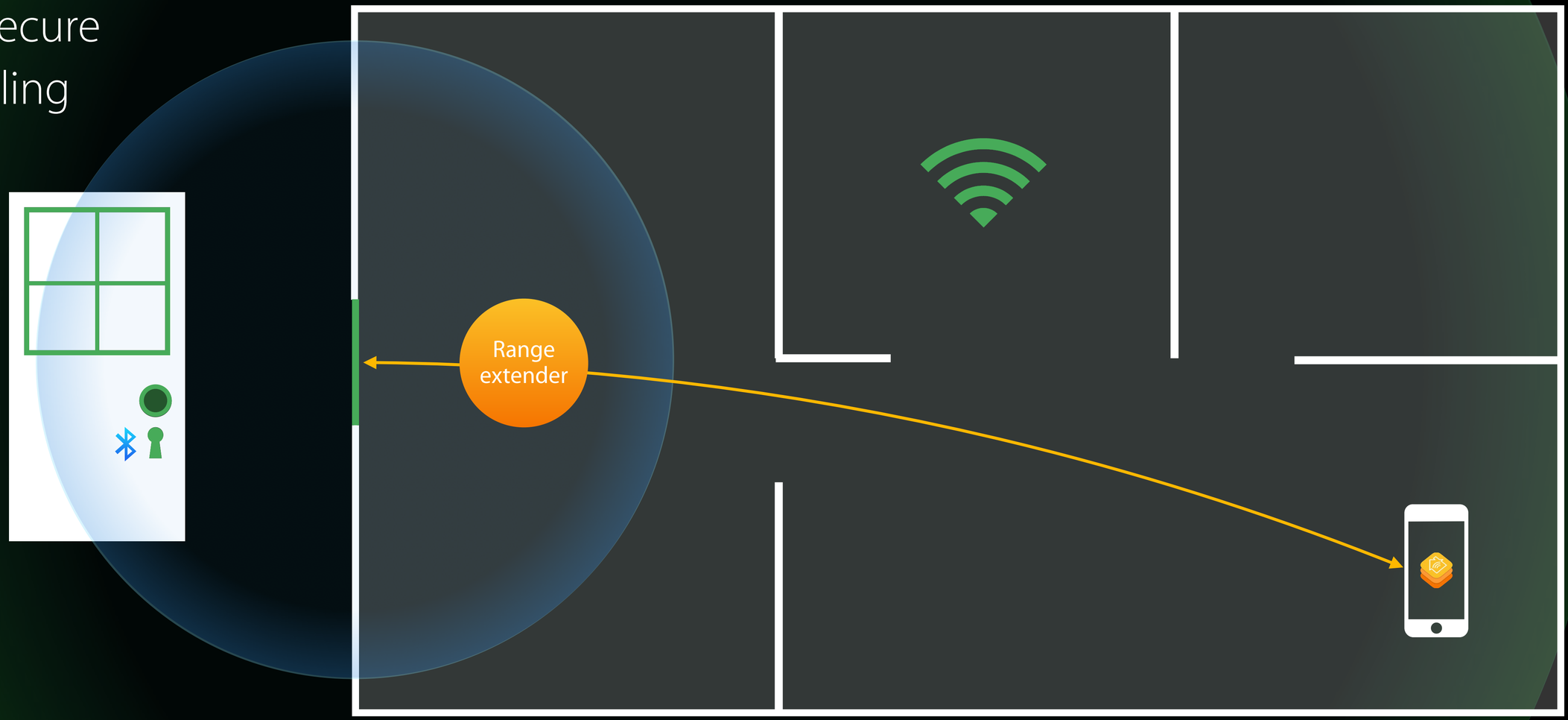
Bluetooth Low Energy

HAP secure tunneling



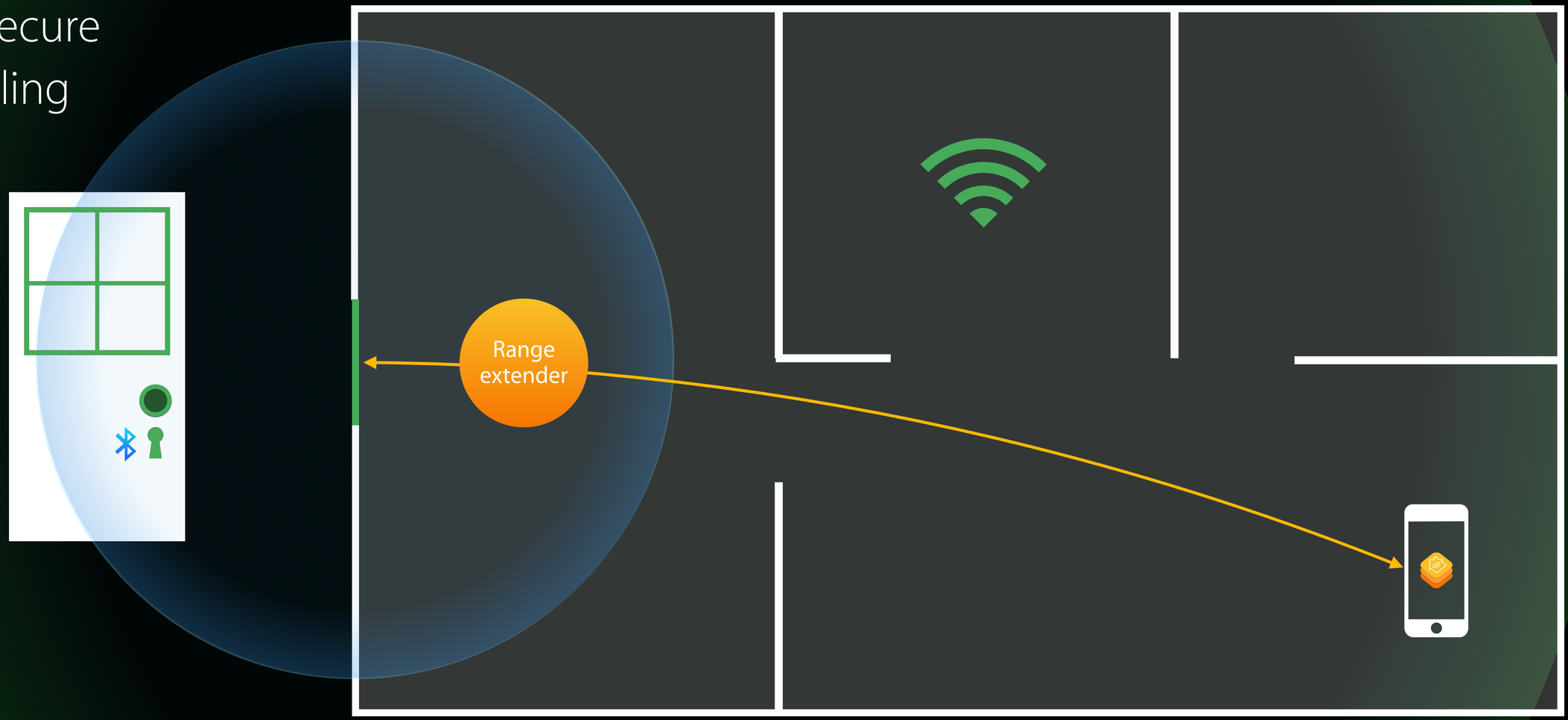
Bluetooth Low Energy

HAP secure tunneling



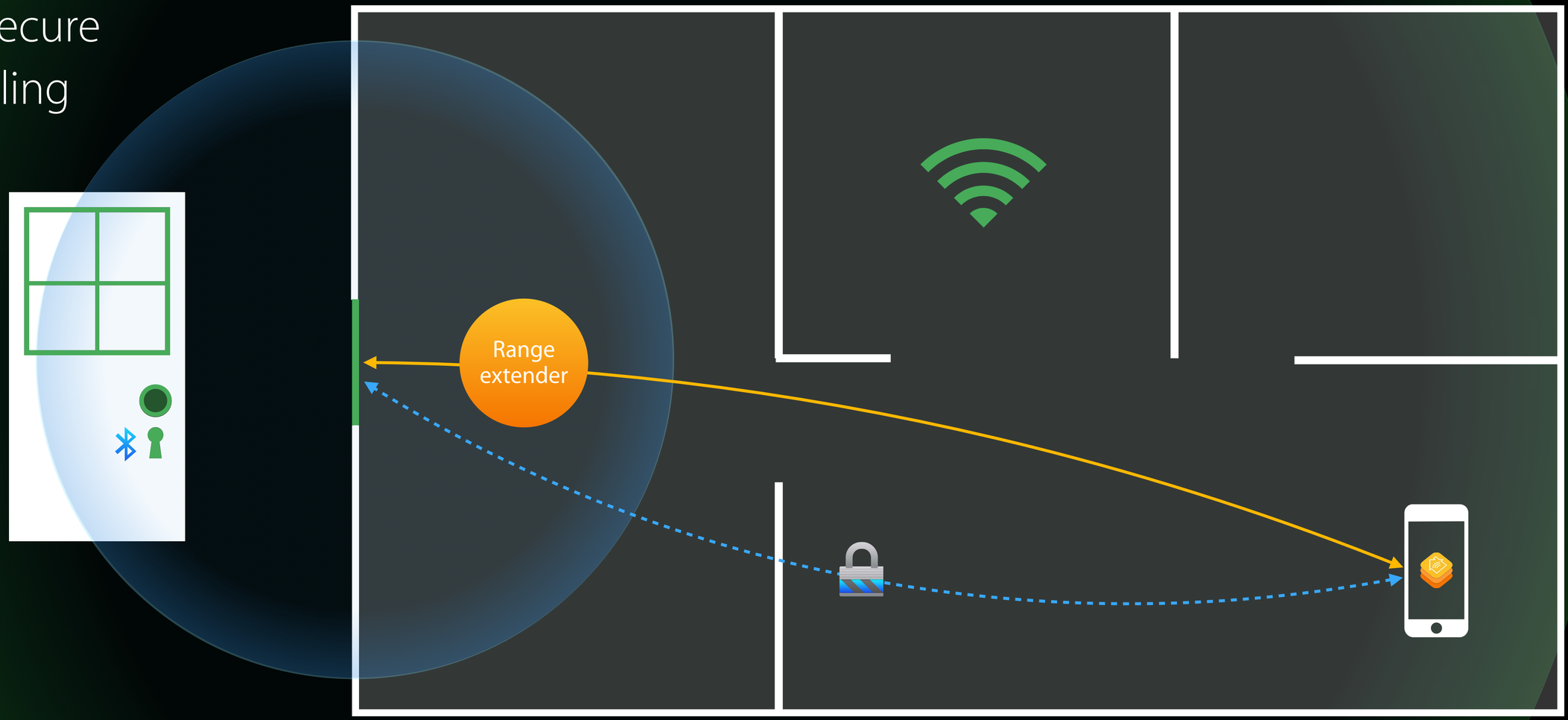
Bluetooth Low Energy

HAP secure tunneling



Bluetooth Low Energy

HAP secure
tunneling



Bluetooth Low Energy

NEW

Bluetooth Low Energy

NEW

Notifications

Bluetooth Low Energy

NEW

Notifications

Metadata for custom characteristics

Bluetooth Low Energy

NEW

Notifications

Metadata for custom characteristics

Support for multiple transports

Accessory Categories

NEW

Accessory Categories

NEW

Accessory Categories

NEW

Window coverings

Awnings
Blinds
Shades

Accessory Categories

NEW

Window coverings

Awnings
Blinds
Shades

Doors and windows

Accessory Categories

NEW

Window coverings

Awnings
Blinds
Shades

Doors and windows

Alarm systems

Accessory Categories

NEW

Window coverings

Awnings
Blinds
Shades

Doors and windows

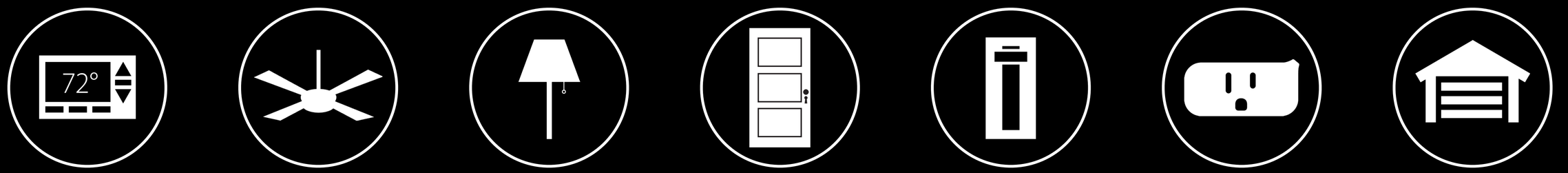
Alarm systems

Sensors

Motion
Air quality
Smoke

Accessory Categories

NEW



Window coverings

Awnings
Blinds
Shades

Doors and windows

Alarm systems

Sensors

Motion
Air quality
Smoke

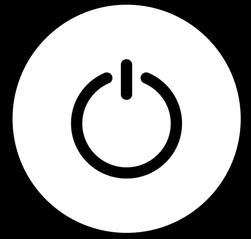
Programmable switches

Accessory Categories

Programmable switch

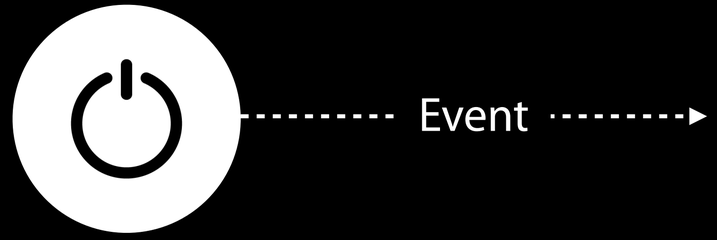
Accessory Categories

Programmable switch



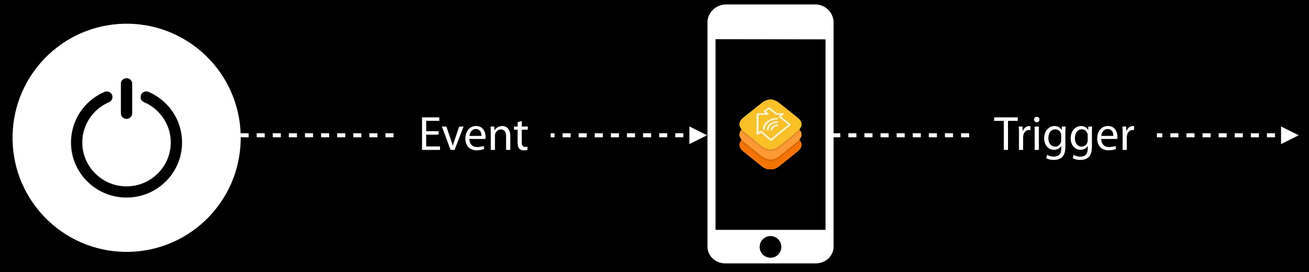
Accessory Categories

Programmable switch



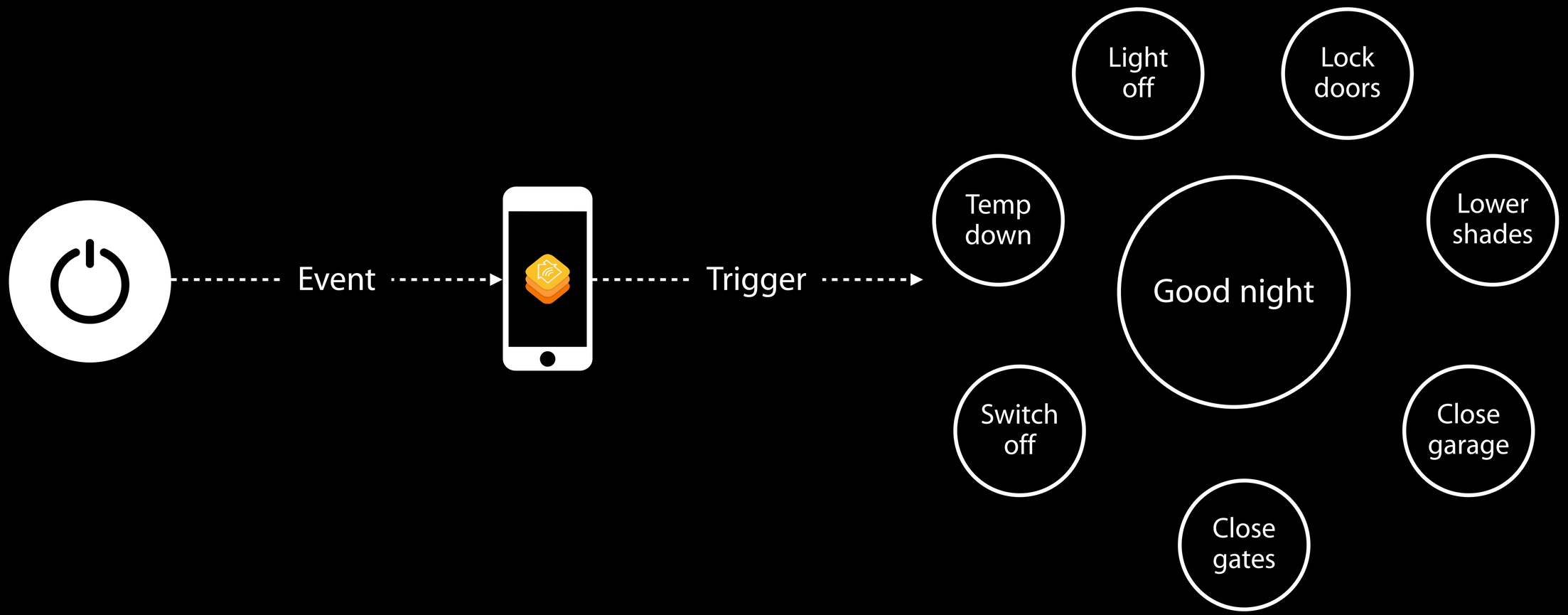
Accessory Categories

Programmable switch



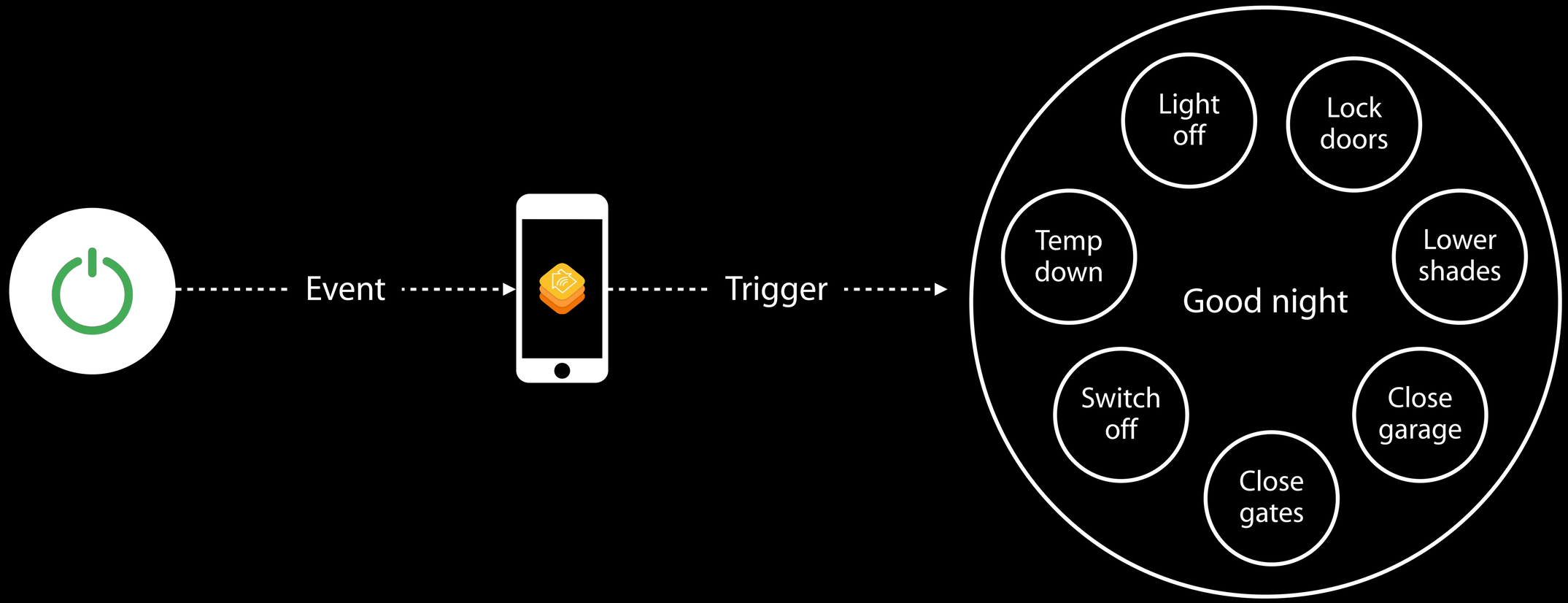
Accessory Categories

Programmable switch



Accessory Categories

Programmable switch



Resources

Resources

HomeKit Accessory Simulator (HAS)

Resources

HomeKit Accessory Simulator (HAS)

HomeKit Accessory Tester (HAT)

Resources

HomeKit Accessory Simulator (HAS)

HomeKit Accessory Tester (HAT)

Updated specifications and tools at MFi Portal

Resources

HomeKit Accessory Simulator (HAS)

HomeKit Accessory Tester (HAT)

Updated specifications and tools at MFi Portal

MFi Program

Resources

HomeKit Accessory Simulator (HAS)

HomeKit Accessory Tester (HAT)

Updated specifications and tools at MFi Portal

MFi Program

- developer.apple.com/programs/mfi/

Summary

Enhancements

Predefined scenes

HomeKit on Apple Watch

Event triggers

Remote access

New features for Bluetooth accessories

New accessory categories

More Information

Documentation

HomeKit

developer.apple.com/homekit

General Inquiries

homekit@apple.com

Technical Support

Apple Developer Forum

Developer Technical Support

developer.apple.com/forums

Labs

HomeKit Lab	App Frameworks Lab Wednesday 2:30PM

WWDC15