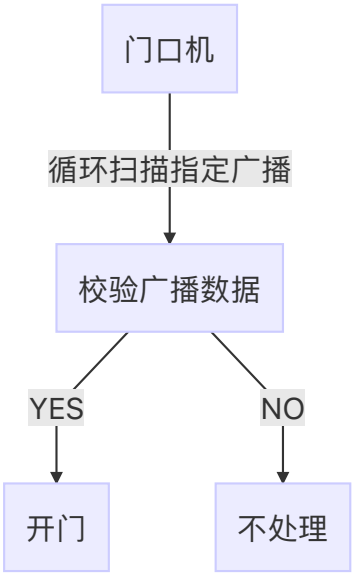
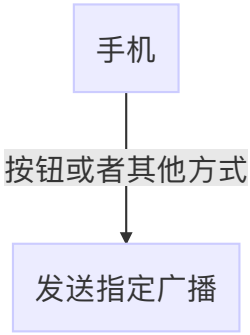
BLE广播方式实现

- 目录：
  - BLE广播方式实现
    - 门禁端：
    - 移动端：
    - 相关说明：
      - 门口机/围墙机作为中心设备，扫描指定UUID服务。
      - 广播数据：（注意，以下做数据校验的时候均为大写）
        - Android
        - iOS
      - UUID校验（20171207改）
    - 接收端过滤验证步骤：
      - ios 例
      - android 例
    - 广播数据类型定义参考
    - 临时密码规则(20171211改)
    - 20171027增加时间广播发送规则
      - Android
      - ios
      - 校验
      - 例子

门禁端：



移动端：



相关说明：

门口机/围墙机作为中心设备，扫描指定UUID服务。

16bit的UUID格式： 0000XXXX-0000-1000-8000-00805f9b34fb ，其中XXXX可变。

- 门口机：如楼栋号0001

扫描指定UUID:00000001-0000-1000-8000-00805f9b34fb

- 围墙机：000a

扫描指定UUID:0000000a-0000-1000-8000-00805f9b34fb

注：如初始化楼栋列表>1也为围墙机，过滤规则使用收到广播中的楼栋号是否在该门禁初始化楼栋列表中

广播数据：（注意，以下做数据校验的时候均为大写）

Android

- 指定ServerUUID用于门口机过滤：门口机UUID
- 指定ServerUUID用于围墙机过滤：围墙机UUID
- 厂商自定义数据：
  - 厂商id：0x5A57(即ZW)
  - 数据：门口机+围墙机临时密码
- Server data :
  - UUID 使用：0000aaaa-0000-1000-8000-00805f9b34fb
  - 数据：门口机+围墙机临时密码

注：厂商数据与Serverdata数据相同，目的为方便解析与加强校验减少数据结构改动

iOS

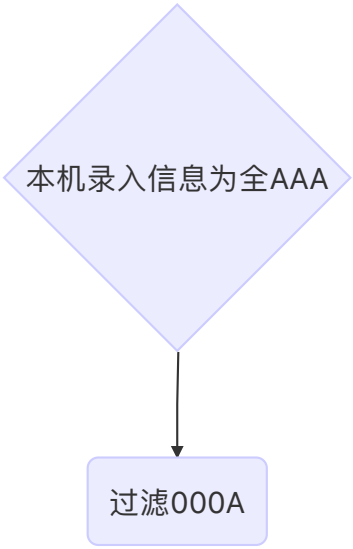
ios通过ibeacon发送数据。

- UUID，格式如：000A0001-0000-1000-8000-00805F9B34FB
  - 前八位中的前四位为围墙机位000A，后四位为门口机楼栋号（如0001），其后面由于新增手机号原因，所以不定，只需要校验前8位楼栋号即可。
- 名称Name为：门口机+围墙机临时密码
- major+minor:为门口机+围墙机临时密码

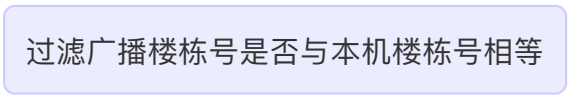
注：Name与major+minor数据相同，目的为方便解析与加强校验及减少数据结构改动

UUID校验（20171207改）

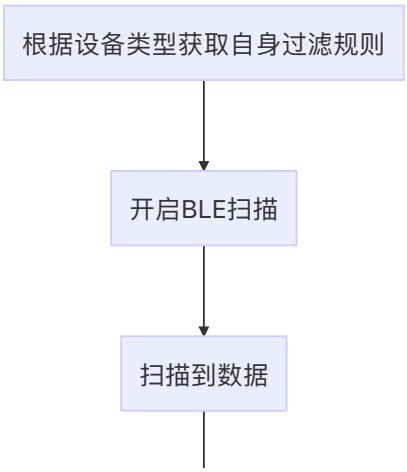
- 蓝牙门禁-围墙机（全部楼栋）

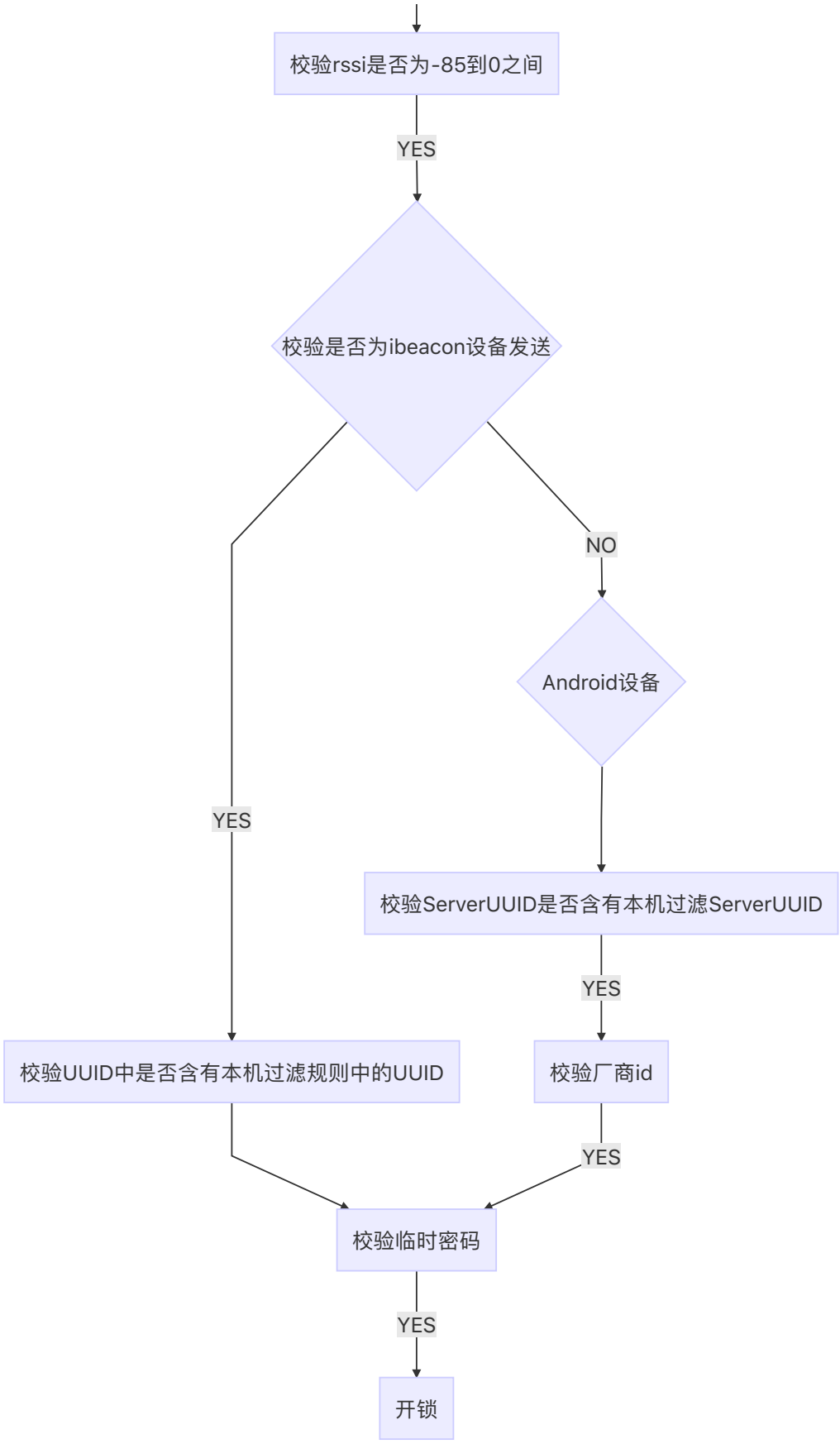


- 蓝牙门禁-门口机（只有1个楼栋）



接收端过滤验证步骤：




校验rssi是否为-85到0之间
YES
校验是否为ibeacon设备发送
NO
Android设备
YES
校验ServerUUID是否含有本机过滤ServerUUID
YES
校验UUID中是否含有本机过滤规则中的UUID
校验厂商id
YES
校验临时密码
YES
开锁

ios 例

接收端接收到数据例子如下：

```
//接收数据：

02011A1AFF4C000215000A000100001000800000805F9B34FB014D029AC50709363
6363636360000000000000000000000000000000000000000000000000000000

/*解析说明：*/
02 //后面2字节
01 // flags 类型标识
1A // flags 类型数据

1A // 后面26字节
FF //厂商设备类型标识
4C00 //组织标识，0x4c00苹果公司标识,https://www.bluetooth.org/en-
us/specification/assigned-numbers/company-identifiers
02 //ibeacon标识位，用于判断ibeacon广播
15 //22个字节标识长度，uuid，major，minor总和的长度。用于与ibeacon标识位共同
一起判断为ibeacon广播

000A000100001000800000805F9B34FB //UUID
014D //Major
029A //Minor

C5 //iBeacon的信号发出强度值 TxPower。这里是补码，转化为原码就是-59，，用来作
为和RSSI一起测距的基准

07 //后面7个字节
09 //完整设备名称标识

363636363636 //设备名称 Name

/*数据说明：*/
   Name: 666666 //门口机临时密码
   Mac: 7D:B6:18:6F:7C:2F
   UUID: 000A0001-0000-1000-8000-00805F9B34FB
   Major: 333
   Minor: 666
   TxPower: -59  //iBeacon的信号发出强度值 TxPower
   rssi: -73  // (接收信号强度) Received Signal Strength Indicator,低功
```

```
耗蓝牙分四级
//-60 ~ 0   4
//-70 ~ -60 3
//-80 ~ -70  2
//<-80 1
   distance: 4.756481003459447
```

- 匹配UUID中前八位，解析出前四位和后四位，中含有该门口机过滤0001，则判定为yes
- Name为门口机临时密码： 666666
- Major+Minor为围墙机临时密码:333666

android 例

```
//android端接收数据例子:
address:51:AD:E0:D9:A5:25,
name:null,
rssi :-60, //（接收信号强度）Received Signal Strength Indicator,低功耗蓝牙分四级
//-60 ~ 0   4
//-70 ~ -60 3
//-80 ~ -70  2
//<-80 1
scanRecord:02010209FF777A303634333231050301000A000916AAAA3938373635340000000000000000000000000000000000000000000000000000000000000

//解析说明
02 //后面2个字节
01 //flags 类型标识
02 //flags 类型数据

09 //后面9个字节
FF //厂商设备类型标识
777A //厂商id，注意这是倒叙
303634333231 //厂商数据

05 //后面5个字节
03 //服务uuid类型标识
0100 //服务uuid ，注意这里是倒叙，即实际是0001
0A00 //服务uuid ，注意这里是倒叙，即实际是000A

09 //后面9个字节
```

```
16 //服务数据 server data 类型标识
AAAA //服务数据uuid
393837363534 //服务数据

/*数据说明：*/
厂商id:777A -> ZW
厂商数据:303634333231 -> 064321//围墙机临时密码

ServerUUid:01000A00 -> 0001000A，分割出来即为0001,000A

ServerUUIDData：AAAA393837363534 -> serveruuid AAAA ,数据 987654//门口机临时密码
```

- 匹配UUID列表，中含有该门口机过滤0001，则判定为yes
- 厂商数据:为围墙机临时密码
- ServerUUIDData:为门口机临时密码

注意：临时密码组合后（因为转换为int类型的原因），如果低于6位，需要在前面补0到6位后比较（是否需要转换后比较取决与比较方式，如果都是int则不需要，如果转换为string则需要补0），ios的major和minor都为3位，不足需要在前补0到3位后再拼接成6位。

广播数据类型定义参考

此门禁中主要用到:

- Android
  - FF（厂商设备类型标识）
  - 03（服务uuid类型标识）
  - 16（server data 类型标识）
  - 以下给出一些类型标识参考。
- ios
  - FF
  - 4C00（组织标识）
  - 02（ibeacon标识位）
  - 15（22个字节标识长度）
  - 09（完整设备名称标识）

以下是部分表示类型参考（仅供参考，ibeacon的是苹果自己定过的，与andorid的蓝牙联盟使用的协议不尽相同，所以类型标识不尽相同）：

```
public static final int BLE_GAP_AD_TYPE_FLAGS = 0x01;//< Flags for
discoverAbility.
    public static final int
BLE_GAP_AD_TYPE_16BIT_SERVICE_UUID_MORE_AVAILABLE = 0x02;//<
Partial list of 16 bit service UUIDs.
    public static final int
BLE_GAP_AD_TYPE_16BIT_SERVICE_UUID_COMPLETE = 0x03;//< Complete
list of 16 bit service UUIDs.
    public static final int
BLE_GAP_AD_TYPE_32BIT_SERVICE_UUID_MORE_AVAILABLE = 0x04;//<
Partial list of 32 bit service UUIDs.
    public static final int
BLE_GAP_AD_TYPE_32BIT_SERVICE_UUID_COMPLETE = 0x05;//< Complete
list of 32 bit service UUIDs.
    public static final int
BLE_GAP_AD_TYPE_128BIT_SERVICE_UUID_MORE_AVAILABLE = 0x06;//<
Partial list of 128 bit service UUIDs.
    public static final int
BLE_GAP_AD_TYPE_128BIT_SERVICE_UUID_COMPLETE = 0x07;//< Complete
list of 128 bit service UUIDs.
    public static final int BLE_GAP_AD_TYPE_SHORT_LOCAL_NAME =
0x08;//< Short local device name.
    public static final int BLE_GAP_AD_TYPE_COMPLETE_LOCAL_NAME =
0x09;//< Complete local device name.
    public static final int BLE_GAP_AD_TYPE_TX_POWER_LEVEL =
0x0A;//< Transmit power level.
    public static final int BLE_GAP_AD_TYPE_CLASS_OF_DEVICE =
0x0D;//< Class of device.
    public static final int BLE_GAP_AD_TYPE_SIMPLE_PAIRING_HASH_C =
0x0E;//< Simple Pairing Hash C.
    public static final int
BLE_GAP_AD_TYPE_SIMPLE_PAIRING_RANDOMIZER_R = 0x0F;//< Simple
Pairing Randomizer R.
    public static final int
BLE_GAP_AD_TYPE_SECURITY_MANAGER_TK_VALUE = 0x10;//< Security
Manager TK Value.
    public static final int
BLE_GAP_AD_TYPE_SECURITY_MANAGER_OOB_FLAGS = 0x11;//< Security
Manager Out Of Band Flags.
    public static final int
BLE_GAP_AD_TYPE_SLAVE_CONNECTION_INTERVAL_RANGE = 0x12;//< Slave
Connection Interval Range.
    public static final int
BLE_GAP_AD_TYPE_SOLICITED_SERVICE_UUIDS_16BIT = 0x14;//< List of
```

```
16-bit Service Solicitation UUIDs.
    public static final int
BLE_GAP_AD_TYPE_SOLICITED_SERVICE_UUIDS_128BIT = 0x15;//< List of
128-bit Service Solicitation UUIDs.
    public static final int BLE_GAP_AD_TYPE_SERVICE_DATA = 0x16;//<
Service Data - 16-bit UUID.
    public static final int BLE_GAP_AD_TYPE_PUBLIC_TARGET_ADDRESS =
0x17;//< Public Target Address.
    public static final int BLE_GAP_AD_TYPE_RANDOM_TARGET_ADDRESS =
0x18;//< Random Target Address.
    public static final int BLE_GAP_AD_TYPE_APPEARANCE = 0x19;//<
Appearance.
    public static final int BLE_GAP_AD_TYPE_ADVERTISING_INTERVAL =
0x1A;//< Advertising Interval.
    public static final int
BLE_GAP_AD_TYPE_LE_BLUETOOTH_DEVICE_ADDRESS = 0x1B;//< LE Bluetooth
Device Address.
    public static final int BLE_GAP_AD_TYPE_LE_ROLE = 0x1C;//< LE
Role.
    public static final int
BLE_GAP_AD_TYPE_SIMPLE_PAIRING_HASH_C256 = 0x1D;//< Simple Pairing
Hash C-256.
    public static final int
BLE_GAP_AD_TYPE_SIMPLE_PAIRING_RANDOMIZER_R256 = 0x1E;//< Simple
Pairing Randomizer R-256.
    public static final int BLE_GAP_AD_TYPE_SERVICE_DATA_32BIT_UUID
= 0x20;//< Service Data - 32-bit UUID.
    public static final int
BLE_GAP_AD_TYPE_SERVICE_DATA_128BIT_UUID = 0x21;//< Service Data -
128-bit UUID.
    public static final int BLE_GAP_AD_TYPE_3D_INFORMATION_DATA =
0x3D;//< 3D Information Data.
    public static final int
BLE_GAP_AD_TYPE_MANUFACTURER_SPECIFIC_DATA = 0xFF;//< Manufacturer
Specific Data.
```

临时密码规则(20171211改)

因MCU没有网络，扩大容错率，临时密码校验除了当前时间及以前时间的临时密码外，增加往后3分钟的临时密码。

- 临时密码方式生成参考：http://note.youdao.com/noteshare?id=69569dfb713175abd7604eb5863e9515

- 蓝牙声波门禁验证密码特别说明(有效密码个数以此为准)：
    - 蓝牙： 收到广播后，解析数据中的临时密码，用5个临时密码（前1分钟+当前分钟+后3分钟一共5个）进行比较，成功则开锁，失败则不处理。
    - 声波/按键输入：收到声波或按键输入后，解析数据中的临时密码，用63个临时密码(前59分钟+当前分钟+后3分钟)进行比较，成功则开锁，失败则不处理。

20171027增加时间广播发送规则

由于模块需要较时，故在开锁广播发送完成之后，添加两秒时间广播，用于MCU较时。

注意：此广播不一定会收到。此广播与前一个门锁广播无任何关联关系

Android

广播数据只包含厂商自定义数据，如下：

- 服务UUID
    - 0001000A 用于做过滤。
- 厂商自定义数据(两种可能性，可能是网络时间或者本地时间)：
    - 一 网络时间:
        - 厂商id：0x5449(即TI)
        - 数据：网络时间，格式：yyyyMMddHHmmss
    - 二 本地时间:
        - 厂商id：0x4C54(即LT)
        - 数据: 本地时间，格式：yyyyMMddHHmmss

ios

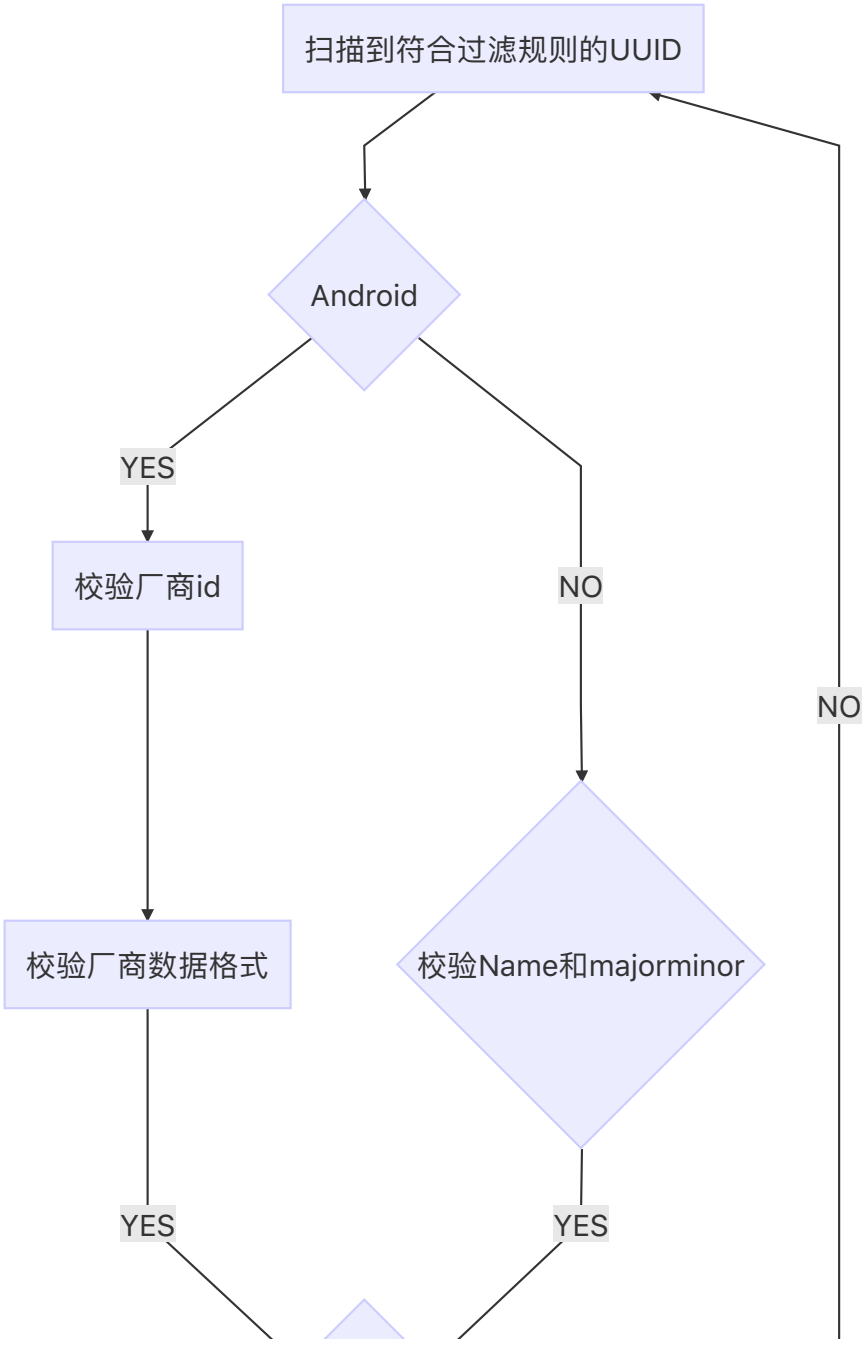
ios通过ibeacon发送数据。

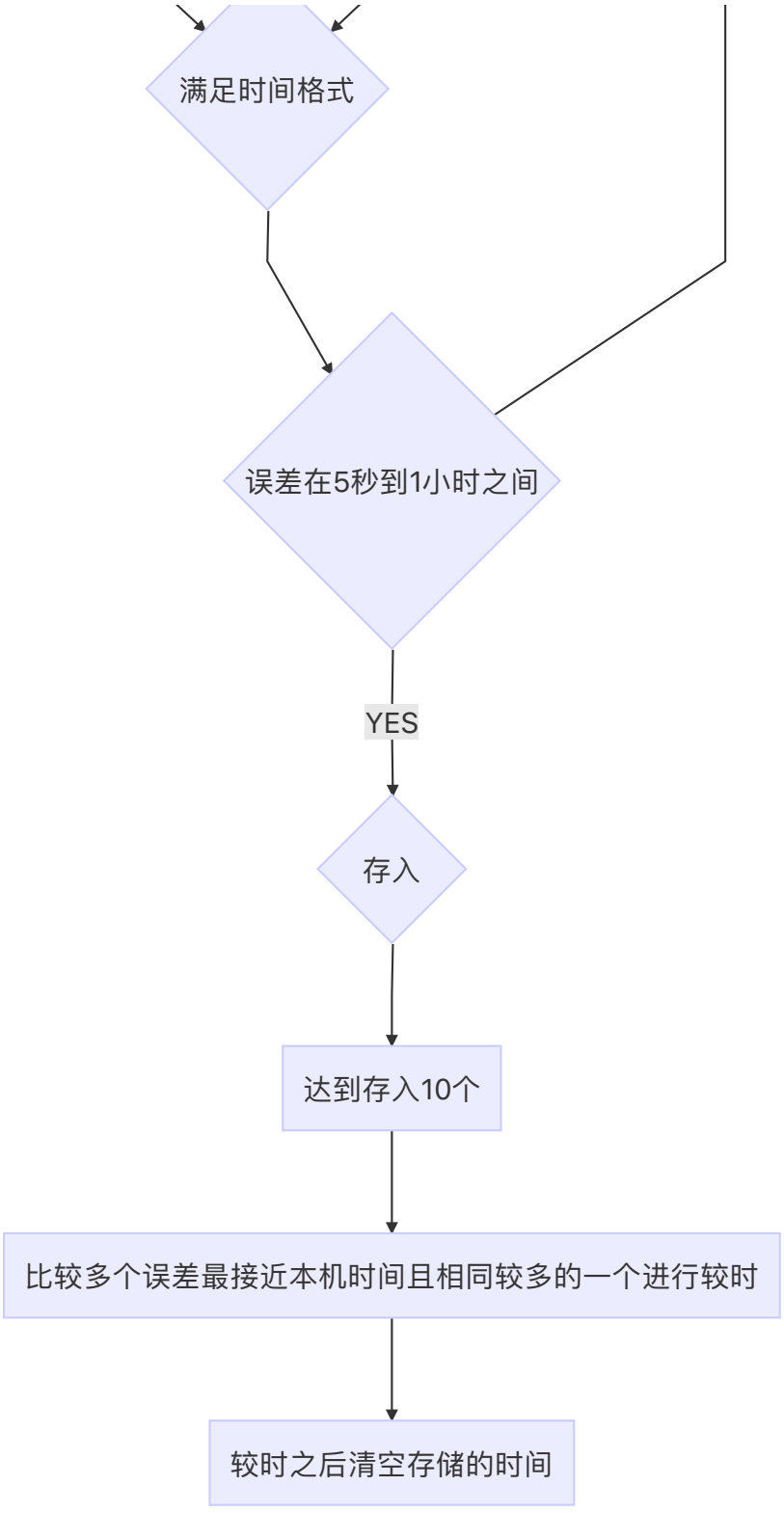
- UUID，格式如：000A0001-0000-1000-8000-00805F9B34FB
    - 前八位中的前四位为围墙机位000A，后四位为门口机楼栋号（如0001），用于过滤。
- Name(两种可能性，可能是网络时间或者本地时间)：

    - 一 网络时间:
        - TI+网络时间(格式同上)。如：TI20171027172122
    - 二 本地时间:
        - LT+本地时间(格式同上)。如：LT20171027172122

- major+minor: 000000 ， 即全0

校验




满足时间格式
误差在5秒到1小时之间
YES
存入
达到存入10个
比较多个误差最接近本机时间且相同较多的一个进行较时
较时之后清空存储的时间

例子

- Android

```
11FF4954323031373130323731373231323205030100 0A000000000000000000000
000000000000000000000000000000000000000000000000000000000

11 //11字节
FF ///厂商type
4954 //TI

3230313731303237313732313232  //20171027172122

05 //5字节
03 //服务uuid类型标识
0100   //0001
000A   //000A
```