INTERNAL USE

MEDIATEK

# AndroidM Keyguard

ACS2/AP2
Rui He
2015/10/20

MEDIATEK CONFIDENTIAL
gineering USE ONLY

# FingerPrint

- **1. Keyguard何时获取FingerPrintService**

  KeyguardService在系统启动时创建, 创建时会new一个KeyguardUpdateMotinor对象, 在这个对象构造函数中去获得FingerPrintService.

```
private KeyguardUpdateMonitor(Context context) {
    mContext = context;
    mSubscriptionManager = SubscriptionManager.from(context);
    mPowerManager = context.getSystemService(PowerManager.class);
    mDeviceProvisioned = isDeviceProvisionedInSettingsDb();


    mFpm = (FingerprintManager) context.getSystemService(Context.FINGERPRINT_SERVICE);
    updateFingerprintListeningState();
} // end KeyguardUpdateMonitor
```

MEDIATEK CONFIDENTIAL
R dcc01@ gmail.com USE ONLY

# FingerPrint

- **2. 何时start OR stop 监听FingerPrint**

  KeyguardService在系统启动时创建, 创建会new一个KeyguardUpdateMotinor对象, 在这个对象构造函数中去获得FingerPrintService.

  由updateFingerprintListeningState()来start或stop监听FingerPrint.

```
private void updateFingerprintListeningState() {
    boolean shouldListenForFingerprint = shouldListenForFingerprint();
    if (mFingerprintDetectionRunning && !shouldListenForFingerprint) {
        stopListeningForFingerprint();
    } else if (!mFingerprintDetectionRunning && shouldListenForFinge  int) {
        startListeningForFingerprint();
```

start之后为true, stop后为false

Keyguard界面且非正在切换用户时(switchingUser)其值为true

MEDIATEK CONFIDENTIAL
R dcc01@ ginreen.com USE ONLY

# FingerPrint

- **2 何时start OR stop 监听FingerPrint**

(1) KeyguardService建立时

```
private KeyguardUpdateMonitor(Context context) {
    mContext = context;
    mSubscriptionManager = SubscriptionManager.from(context);
    mPowerManager = context.getSystemService(PowerManager.class);
    mDeviceProvisioned = isDeviceProvisionedInSettingsDb();


    mFpm = (FingerprintManager) context.getSystemService(Context.FINGERPRINT_SERVICE);
    updateFingerprintListeningState();
} ? end KeyguardUpdateMonitor ?
```

(2) 灭屏时

```
protected void handleFinishedGoingToSleep(int arg1) {
    clearFingerprintRecognized();
    final int count = mCallbacks.size();
    for (int i = 0; i < count; i++) {
        KeyguardUpdateMonitorCallback cb = mCallbacks.get(i).get();
        if (cb != null) {
            cb.onFinishedGoingToSleep(arg1);
        }
    }
    updateFingerprintListeningState();
}
```

MEDIATEK CONFIDENTIAL
R dc01@sinreen.com USE ONLY

# FingerPrint

- **2 何时start OR stop 监听FingerPrint**

(3) 亮屏时

```
protected void handleStartedWakingUp() {
    Log.d(TAG, "handleStartedWakingUp");
    updateFingerprintListeningState();
```

(4) 切换用户时

```
protected void handleUserSwitching(int userId, IRemoteCallback reply) {
    mSwitchingUser = true;
    updateFingerprintListeningState();
```

(5)切换用户完成时

```
protected void handleUserSwitchComplete(int userId) {
    mSwitchingUser = false;
    updateFingerprintListeningState();
```

MEDIATEK CONFIDENTIAL
R dcc01@sprd.com USE ONLY

# FingerPrint

- **2 何时start OR stop 监听FingerPrint**

  (6) 锁屏绘制完成时

```
private void handleKeyguardReset() {
    if (DEBUG) Log.d(TAG, "handleKeyguardReset");
    if (!isUnlockingWithFingerprintAllowed()) {
        updateFingerprintListeningState();
    }
}
```

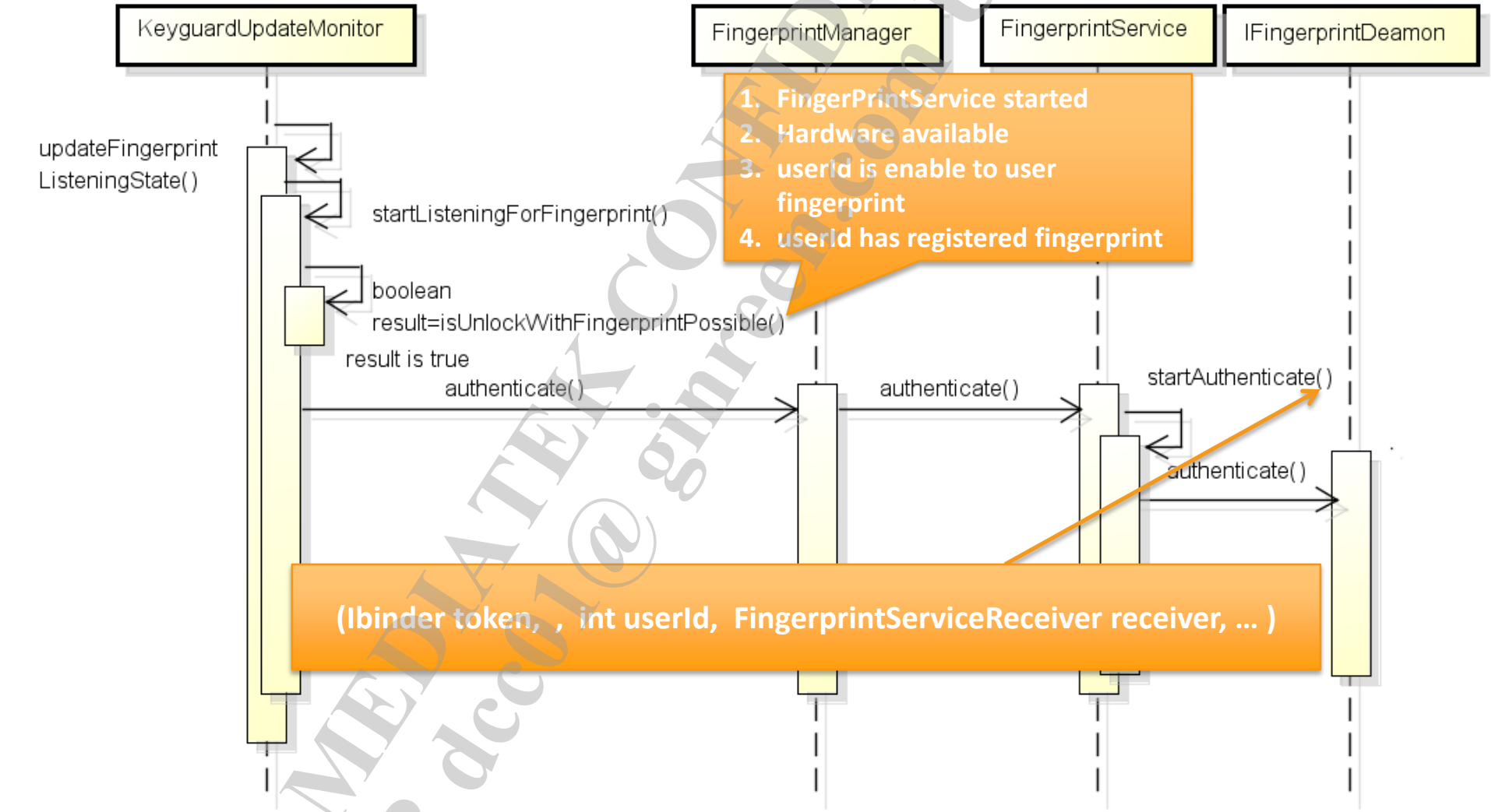
  (7) 锁屏visibility改变时

```
private void handleKeyguardVisibilityChanged(int showing) {
    if (DEBUG) Log.d(TAG, "handleKeyguardVisibilityChanged(" + showing + ")");
    boolean isShowing = (showing == 1);
    mKeyguardIsVisible = isShowing;
    for (int i = 0; i < mCallbacks.size(); i++) {
        KeyguardUpdateMonitorCallback cb = mCallbacks.get(i).get();
        if (cb != null) {
            cb.onKeyguardVisibilityChangedRaw(isShowing);
        }
    }
    updateFingerprintListeningState();
}
```

MEDIATEK CONFIDENTIAL
R dcc01@gmeen.com USE ONLY

# FingerPrint

- 3 开启FingerPrint验证流程




MEDIATEK CONFIDENTIAL
R dcc01@ginreen.com USE ONLY

powered by Astah

# FingerPrint

- **4 FingerPrint验证结果回传**

底层回传给FingerprintService

```
private IFingerprintDaemonCallback mDaemonCallback = new IFingerprintDaemonCallback.Stub() {

    @Override
    public void onEnrollResult(long deviceId, int fingerId, int groupId, int remaining) {
        dispatchEnrollResult(deviceId, fingerId, groupId, remaining);
    }

    @Override
    public void onAcquired(long deviceId, int acquiredInfo) {
        dispatchAcquired(deviceId, acquiredInfo);
    }

    @Override
    public void onAuthenticated(long deviceId, int fingerId, int groupId) {
        dispatchAuthenticated(deviceId, fingerId, groupId);
    }

    @Override
    public void onError(long deviceId, int error) {
        dispatchError(deviceId, error);
    }

    @Override
    public void onRemoved(long deviceId, int fingerId, int groupId) {
        dispatchRemoved(deviceId, fingerId, groupId);
    }

    @Override
    public void onEnumerate(long deviceId, int[] fingerIds, int[] groupIds) {
        dispatchEnumerate(deviceId, fingerIds, groupIds);
    }

};
```

设置指纹时,注册指纹

获取到指纹image, 但还未处理

指纹验证结果返回

指纹验证时出错

删除指纹

提示信息

MEDIATEK CONFIDENTIAL
R tcc01@gireen.com USE ON

# FingerPrint

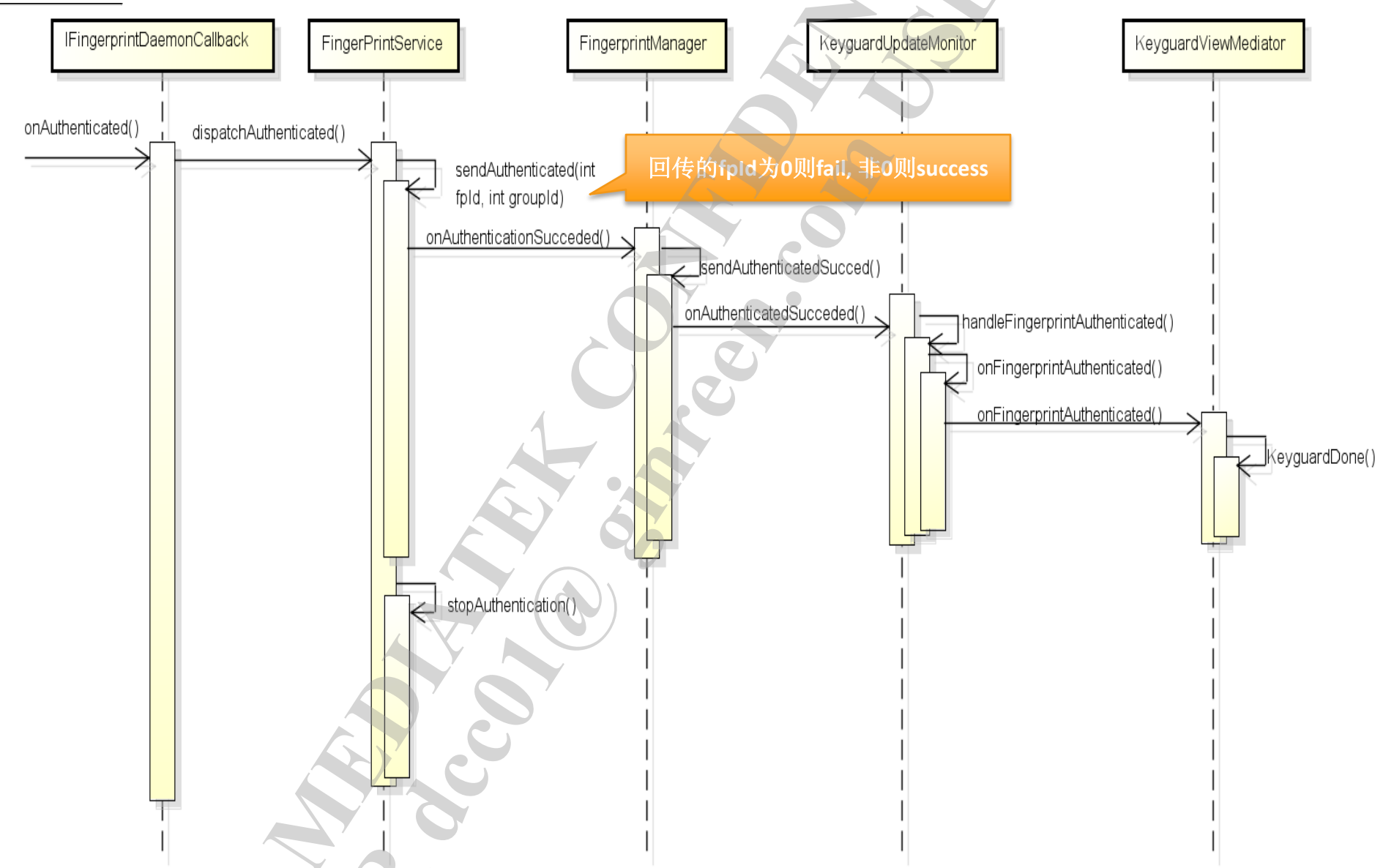
- **4 FingerPrint验证结果回传**

FingerprintService回传给FingerprintManager

```
private IFingerprintServiceReceiver mServiceReceiver = new IFingerprintServiceReceiver.Stub() {

    @Override // binder call
    public void onEnrollResult(long deviceId, int fingerId, int groupId, int remaining) {
        mHandler.obtainMessage(MSG_ENROLL_RESULT, remaining, 0,
                new Fingerprint(null, groupId, fingerId, deviceId)).sendToTarget();
    }

    @Override // binder call
    public void onAcquired(long deviceId, int acquireInfo) {
        mHandler.obtainMessage(MSG_ACQUIRED, acquireInfo, 0, deviceId).sendToTarget();
    }

    @Override // binder call
    public void onAuthenticationSucceeded(long deviceId, Fingerprint fp) {
        mHandler.obtainMessage(MSG_AUTHENTICATION_SUCCEEDED, fp).sendToTarget();
    }

    @Override // binder call
    public void onAuthenticationFailed(long deviceId) {
        mHandler.obtainMessage(MSG_AUTHENTICATION_FAILED).sendToTarget();;
    }

    @Override // binder call
    public void onError(long deviceId, int error) {
        mHandler.obtainMessage(MSG_ERROR, error, 0, deviceId).sendToTarget();
    }

    @Override // binder call
    public void onRemoved(long deviceId, int fingerId, int groupId) {
        mHandler.obtainMessage(MSG_REMOVED, fingerId, groupId, deviceId).sendToTarget();
    }
};
```

MEDIATEK CONFIDENTIAL
R_cc1@gtren.com USE ON

# FingerPrint




MEDIATEK CONFIDENTIAL
R dcc01@ qinreen.com USE ONLY

powered by Astah

# FingerPrint

- **4 FingerPrint验证错误回传**

```
@Override
public void onError(long deviceId, int error) {
    dispatchError(deviceId, error);
}
```

error类型:

(1) FINGER_PRINT_ERROR_UNABLE_TO_PROCESS
the sensor was unable to process the current image

(2) FINGER_PRINT_ERROR_HW_UNAVAILABLE
hardware unavailable

(3) FINGER_PRINT_ERROR_NO_SPACE
no enough storage to complete the operation

(4) FINGER_PRINT_ERROR_TIMEOUT
timeout(default 30seconds)

(5) FINGER_PRINT_ERROR_CANCELD
operation was canceld(the user switched or another pending operation disables it)

(6) FINGER_PRINT_ERROR_LOCKOUT
too many attempts

MEDIATEK CONFIDENTIAL
R dc01@ gfreen.com USE ONLY

# 重要文件及路径

KeyguardViewMediator.java(alps\frameworks\base\packages\SystemUI\src\com\android\systemui\keyguard)

KeyguardUpdateMonitor.java(alps\frameworks\base\packages\Keyguard\src\com\android\keyguard)

FingerprintManager.java(alps\frameworks\base\core\java\android\hardware\fingerprint)

FingerprintService.java(alps\frameworks\base\core\java\com\android\server\fingerprint)

MEDIATEK CONFIDENTIAL
R dcc01@ginreen.com USE ONLY


*everyday genius*

Copyright © MediaTek Inc. All rights reserved.

MEDIATEK CONFIDENTIAL
R dcc01@ gibreen.com USE ON