

密级状态：绝密() 秘密() 内部() 公开(√)

RK3368/RK3368H_ANDROID8.1_SDK_V1.00 发布说明

(技术部，第二系统产品部)

文件状态： <input type="checkbox"/> 正在修改 <input checked="" type="checkbox"/> 正式发布	当前版本：	V1.1
	作 者：	吴良清，刘益星
	完成日期：	2018-01-31
	审 核：	陈海燕，黄祖芳
	完成日期：	2018-01-31

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

(版本所有,翻版必究)

版本历史

版本号	作者	修改日期	修改说明	备注
V1.00	吴良清, 刘益星	2018.01.23	正式发布初版	
V1.1	刘益星	2018.01.31	添加 RK3368 支持说明	

目 录

1	概述.....	1
2	RK3368/RK3368H 差异.....	1
3	主要支持功能.....	1
4	SDK 获取说明.....	1
4.1	获取 SDK.....	1
4.2	补充说明.....	2
5	SDK 编译说明.....	3
5.1	JDK 安装.....	3
5.2	编译模式.....	3
5.3	代码编译.....	3
5.3.1	uboot 编译步骤.....	3
5.3.2	kernel 编译步骤.....	3
5.3.3	Android 编译及固件生成步骤.....	4
5.4	刷机说明.....	5
附录 A	编译开发环境搭建.....	6
附录 B	SSH 公钥操作说明.....	10
附录 B-1	SSH 公钥生成.....	10
附录 B-2	使用 key-chain 管理密钥.....	10
附录 B-3	多台机器使用相同 ssh 公钥.....	11
附录 B-4	一台机器切换不同 ssh 公钥.....	12
附录 B-5	密钥权限管理.....	13
附录 B-6	Git 权限申请说明.....	13

1 概述

本 SDK 是基于谷歌 Android8.1 64bit 系统，适配瑞芯微 RK3368/RK3368H 芯片的软件包，适用于 RK3368/RK3368H TABLET 平台以及基于其上所有开发产品。

注：以下文档描述以 RK3368H 为例，RK3368 除部分芯片功能不同外（芯片差异说明如下），其他部分完全相同，不再重复说明。

2 RK3368/RK3368H 差异

RK3368 与 RK3368H 芯片差异如下：

	RK3368	RK3368H
Video Decoder	4K	1080P
GMAC	Embedded	N/A
TS	Embedded	N/A
Smart Card	Embedded	N/A

3 主要支持功能

参数	模块名
数据通信	Wi-Fi、USB 以太网卡、USB、SDCARD
应用程序	Launcher3、APK 安装器、浏览器、计算器、日历、相机、闹钟、下载、电子邮件、资源管理器、GMS 应用、音乐、录音、设置、视频播放器

4 SDK 获取说明

4.1 获取 SDK

SDK 通过瑞芯微代码服务器对外发布。其编译开发环境，参考[附录 A 编译开发环境搭建](#)。

客户向瑞芯微技术窗口申请 SDK，需同步提供 SSH 公钥进行服务器认证授权，获得授权后即可同步代码。关于瑞芯微代码服务器 SSH 公钥授权，请参考[附录 B SSH 公钥操作说明](#)。

RK3368H_ANDROID8.1_SDK 下载地址如下：

```
repo init --repo-url=ssh://git@www.rockchip.com.cn:2222/repo-release/tools/repo.git  
-u ssh://git@www.rockchip.com.cn:2222/rk3368-orc/manifests.git -m  
rk3368_orc_release.xml
```

注，repo 是 google 用 Python 脚本写的调用 git 的一个脚本，主要是用来下载、管理 Android 项目的软件仓库，其下载地址如下：

```
git clone ssh://git@www.rockchip.com.cn:2222/repo-release/tools/repo
```

为方便客户快速获取 SDK 源码，瑞芯微技术窗口通常会提供对应版本的 SDK 初始压缩包。以 RK3368H_ANDROID8.1_SDK_20180123.tar.gz 为例，拷贝到该初始化包后，通过如下命令可检出源码：

```
mkdir RK3368H  
tar zxvf RK3368H_ANDROID8.1_SDK_20180123.tar.gz -C RK3368H  
cd RK3368H  
.repo/repo/repo sync -l  
.repo/repo/repo sync
```

4.2 补充说明

Android8.1 SDK 已不再支持 UMS 功能，平台设备皆使用合并分区；

Android8.1 SDK 已支持全盘加密功能；

Android8.1 SDK 已支持 Verified boot 的功能。

5 SDK 编译说明

5.1 JDK 安装

Android8.1 系统编译依赖于 JAVA 8，编译之前需安装 OpenJDK。

安装命令如下：

```
sudo apt-get install openjdk-8-jdk
```

配置 JAVA 环境变量，例如，安装路径为 `/usr/lib/jvm/java-8-openjdk-amd64`，可在终端执行如下命令配置环境变量：

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

5.2 编译模式

SDK 默认以 `userdebug` 模式编译。

使用 `adb` 时，若要执行 `push` 等操作，需要先执行 `adb root`，`adb disable-verity`，`adb reboot` 重启机器后，再做 `adb root`，`adb remount`，如此之后才可执行 `adb push` 等操作。

5.3 代码编译

5.3.1 uboot 编译步骤

```
make rk3368h_defconfig
make ARCH=aarch64 -j12
```

编译完，会生成 `trust.img`、`rk3368h_loader_v2.02.260.bin`、`uboot.img` 三个文件。

该 `rk3368h_loader_v2.02.260.bin` 可兼容各 DDR 类型及容量，默认运行频率为 600MHz。

5.3.2 kernel 编译步骤

RK3368H 样机配置与编译如下：

```
make ARCH=arm64 rockchip_defconfig
make ARCH=arm64 rk368-xikp.img -j12
```

编译完成后，`kernel` 根目录，生成 `kernel.img`，`resource.img` 两个镜像文件。

5.3.3 Android 编译及固件生成步骤

客户按实际编译环境配置好 JDK 环境变量后，按照以下步骤配置完后，执行 `make` 即可。

```
$ source build/envsetup.sh
$ lunch
```

You're building on Linux

Lunch menu... pick a combo:

1. aosp_arm-eng
2. aosp_arm64-eng
3. aosp_mips-eng
4. aosp_mips64-eng
5. aosp_x86-eng
6. aosp_x86_64-eng
7. rk3368-userdebug
8. rk3368-user

选择 **rk3368-userdebug**，输入对应序号 **7**。

\$ make -j4

完成编译后，执行 SDK 根目录下的 **mkimage.sh** 脚本生成固件，所有烧写所需的镜像将都会拷贝于 **rockdev/Image-rk3368** 目录。

```
rockdev/Image-rk3368
├─ boot.img
├─ kernel.img
├─ MiniLoaderAll.bin
├─ misc.img
├─ oem.img
├─ parameter.txt
├─ pcba_small_misc.img
├─ pcba_whole_misc.img
├─ recovery.img
├─ resource.img
└─ system.img
```

└─ trust.img
└─ uboot.img
└─ vendor.img

得到所有镜像文件后，为了方便烧写及量产，通常可手动将这些单独的镜像通过脚本打包成为 update.img。

5.4 刷机说明

SDK 提供烧写工具，如下图所示。编译生成相应的固件后，进入 loader 模式，即可进行刷机。对于已烧过其它固件的机器，请选择低格设备，擦除 idb，然后进行刷机。



附录 A 编译开发环境搭建

1. Initializing a Build Environment

This section describes how to set up your local work environment to build the Android source files. You will need to use Linux or Mac OS. Building under Windows is not currently supported.

Note: The source download is approximately 35GB in size. You will need over 45GB free to complete a single build, and up to 100GB (or more) for a full set of builds.

For an overview of the entire code-review and code-update process, see [Life of a Patch](#).

2. Choosing a Branch

Some of the requirements for your build environment are determined by which version of the source code you plan to compile. See [Build Numbers](#) for a full listing of branches you may choose from. You may also choose to download and build the latest source code (called "master"), in which case you will simply omit the branch specification when you initialize the repository.

Once you have selected a branch, follow the appropriate instructions below to set up your build environment.

3. Setting up a Linux build environment

These instructions apply to all branches, including master.

The Android build is routinely tested in house on recent versions of Ubuntu LTS (10.04), but most distributions should have the required build tools available. Reports of successes or failures on other distributions are welcome.

For Gingerbread (2.3.x) and newer versions, including the master branch, a 64-bit environment is required. Older versions can be compiled on 32-bit systems.

Note: It is also possible to build Android in a virtual machine. If you are running Linux in a virtual machine, you will need at least 16GB of RAM/swap and 30GB or more of disk space in order to build the Android tree.

Detailed instructions for Ubuntu and MacOS follow. In general you will need:

- A. Python 2.6 -- 2.7, which you can download from python.org.
- B. GNU Make 3.81 -- 3.82, which you can download from gnu.org.
- C. JDK 6 if you wish to build Gingerbread or newer; JDK 5 for Froyo or older. You can download both from java.sun.com.
- D. Git 1.7 or newer. You can find it at git-scm.com.

4. Installing the JDK

The master branch of Android in the Android Open Source Project (AOSP) requires Java 8. On Ubuntu, use OpenJDK. Java 8: For the latest version of Android.

```
$ sudo apt-get update
```

```
$ sudo apt-get install openjdk-8-jdk
```

Optionally, update the default Java version by running:

```
$ sudo update-alternatives --config java
```

```
$ sudo update-alternatives --config javac
```

If you encounter version errors for Java, set its path as described in the Wrong Java Version section.

To develop older versions of Android, download and install the corresponding version of the Java JDK:

Java 8: for Gingerbread through Nougat

Java 7: for Gingerbread through Lollipop & Marshmallow

Java 6: for Gingerbread through KitKat

Java 5: for Cupcake through Froyo

5. Installing required packages (Ubuntu 12.04)

You will need a 64-bit version of Ubuntu. Ubuntu 12.04 is recommended. Building using

an older version of Ubuntu is not supported on master or recent releases.

```
$ sudo apt-get install git gnupg flex bison gperf build-essential \  
zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \  
libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \  
libgl1-mesa-dev g++-multilib mingw32 tofrodos \  
python-markdown libxml2-utils xsltproc zlib1g-dev:i386  
  
$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/libGL.so
```

6. Installing required packages (Ubuntu 10.04 -- 11.10)

Building on Ubuntu 10.04-11.10 is no longer supported, but may be useful for building older releases of AOSP.

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential \  
zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs \  
x11proto-core-dev libx11-dev lib32readline5-dev lib32z-dev \  
libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown \  
libxml2-utils xsltproc
```

On Ubuntu 10.10:

```
$ sudo ln -s /usr/lib32/mesa/libGL.so.1 /usr/lib32/mesa/libGL.so
```

On Ubuntu 11.10:

```
$ sudo apt-get install libx11-dev:i386
```

7. Configuring USB Access

Under GNU/Linux systems (and specifically under Ubuntu systems), regular users can't directly access USB devices by default. The system needs to be configured to allow such access.

The recommended approach is to create a file `/etc/udev/rules.d/51-android.rules` (as the root user) and to copy the following lines in it. `<username>` must be replaced by the actual username of the user who is authorized to access the phones over USB.

```
# adb protocol on passion (Rockchip products)  
SUBSYSTEM=="usb", ATTR{idVendor}=="2207", ATTR{idProduct}=="0010", MODE="0600",
```

```
OWNER="<username>"
```

Those new rules take effect the next time a device is plugged in. It might therefore be necessary to unplug the device and plug it back into the computer.

This is known to work on both Ubuntu Hardy Heron (8.04.x LTS) and Lucid Lynx (10.04.x LTS). Other versions of Ubuntu or other variants of GNU/Linux might require different configurations.

References : <http://source.android.com/source/initializing.html>

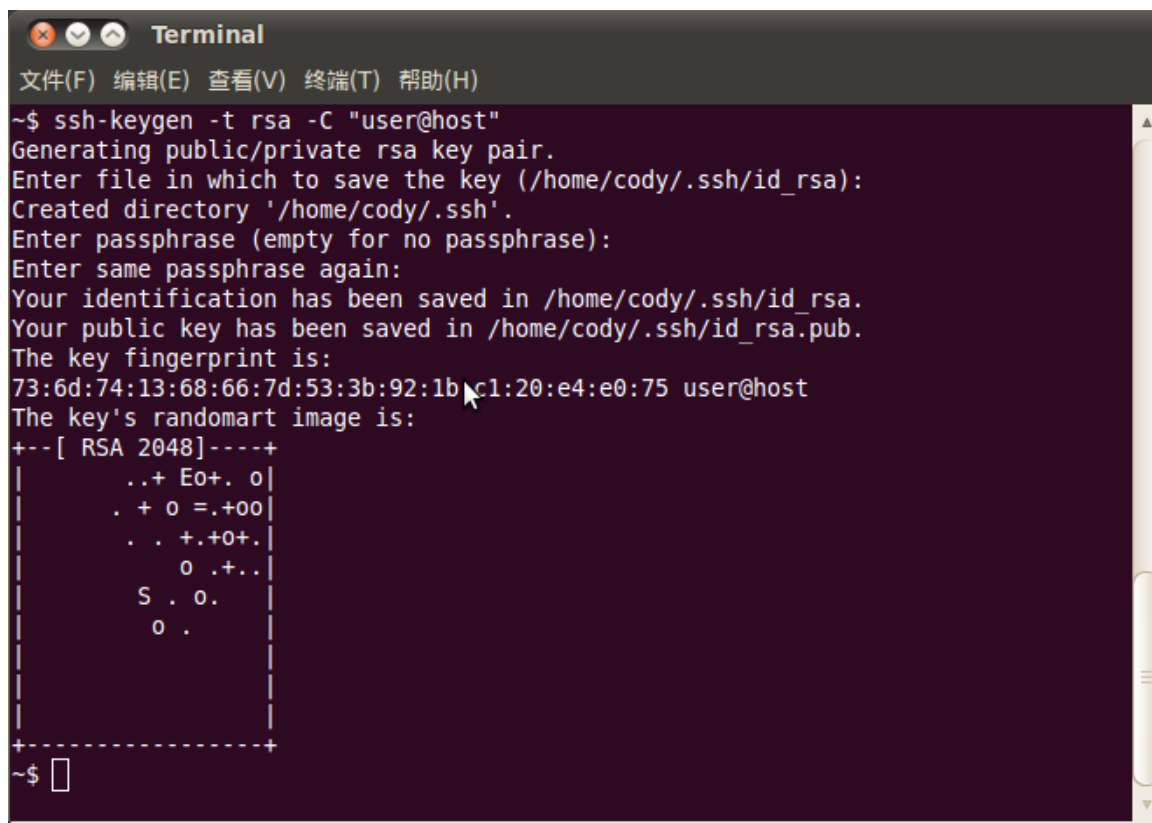
附录 B SSH 公钥操作说明

附录 B.1 SSH 公钥生成

使用如下命令生成:

```
ssh-keygen -t rsa -C "user@host"
```

请将 `user@host` 替换成您的邮箱地址。



```
Terminal
文件(F) 编辑(E) 查看(V) 终端(T) 帮助(H)
~$ ssh-keygen -t rsa -C "user@host"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cody/.ssh/id_rsa):
Created directory '/home/cody/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cody/.ssh/id_rsa.
Your public key has been saved in /home/cody/.ssh/id_rsa.pub.
The key fingerprint is:
73:6d:74:13:68:66:7d:53:3b:92:1b:c1:20:e4:e0:75 user@host
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      .+. Eo+. o |
|    . + o =.+oo |
|   . . +.+o+. |
|      o .+. . |
|     S . o. |
|      o . |
+-----+
~$
```

命令运行完成会在你的目录下生成 key 文件。

```
~$ ls -l .ssh/
总用量 8
-rw----- 1 cody cody 1675 2012-10-15 11:38 id_rsa
-rw-r--r-- 1 cody cody 391 2012-10-15 11:38 id_rsa.pub
```

请妥善保存生成的私钥文件 **id_rsa** 和密码，并将 **id_rsa.pub** 发邮件给 SDK 发布服务器的管理人员。

附录 B.2 使用 key-chain 管理密钥

推荐您使用比较简易的工具 **keychain** 管理密钥。

具体使用方法如下：

1. 安装 **keychain** 软件包：

```
$sudo aptitude install keychain
```

2. 配置使用密钥：

```
$vim ~/.bashrc
```

增加下面这行：

```
eval `keychain --eval ~/.ssh/id_rsa`
```

其中，`id_rsa` 是私钥文件名称。

以上配置以后，重新登录控制台，会提示输入密码，只需输入生成密钥时使用的密码即可，若无密码可不输入。

另外，请尽量不要使用 `sudo` 或 `root` 用户，除非您知道如何处理，否则将导致权限以及密钥管理混乱。

附录 B.3 多台机器使用相同 `ssh` 公钥

在不同机器使用，可以将你的 `ssh` 私钥文件 `id_rsa` 拷贝到要使用的机器的“`~/.ssh/id_rsa`”即可。

在使用错误的私钥会出现如下提示，请注意替换成正确的私钥。

```
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
git@172.16.10.211's password: █
```

添加正确的私钥后，就可以使用 `git` 克隆代码，如下图：

```
~$ cd tmp/
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
remote: Counting objects: 237923, done.
remote: Compressing objects: 100% (168382/168382), done.
Receiving objects: 9% (21570/237923), 61.52 MiB | 11.14 MiB/s
```

添加 `ssh` 私钥可能出现如下提示错误：

```
Agent admitted failure to sign using the key
```

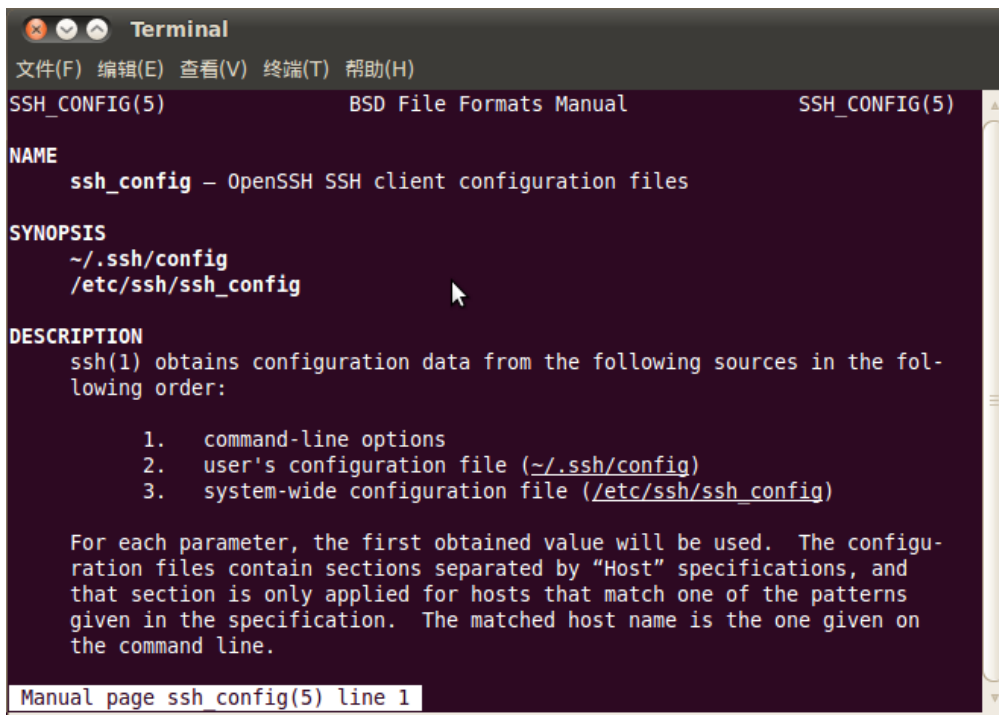
在 `console` 输入如下命令即可解决：

```
ssh-add ~/.ssh/id_rsa
```

附录 B.4 一台机器切换不同 `ssh` 公钥

可以参考 `ssh_config` 文档配置 `ssh`。

```
~$ man ssh_config
```

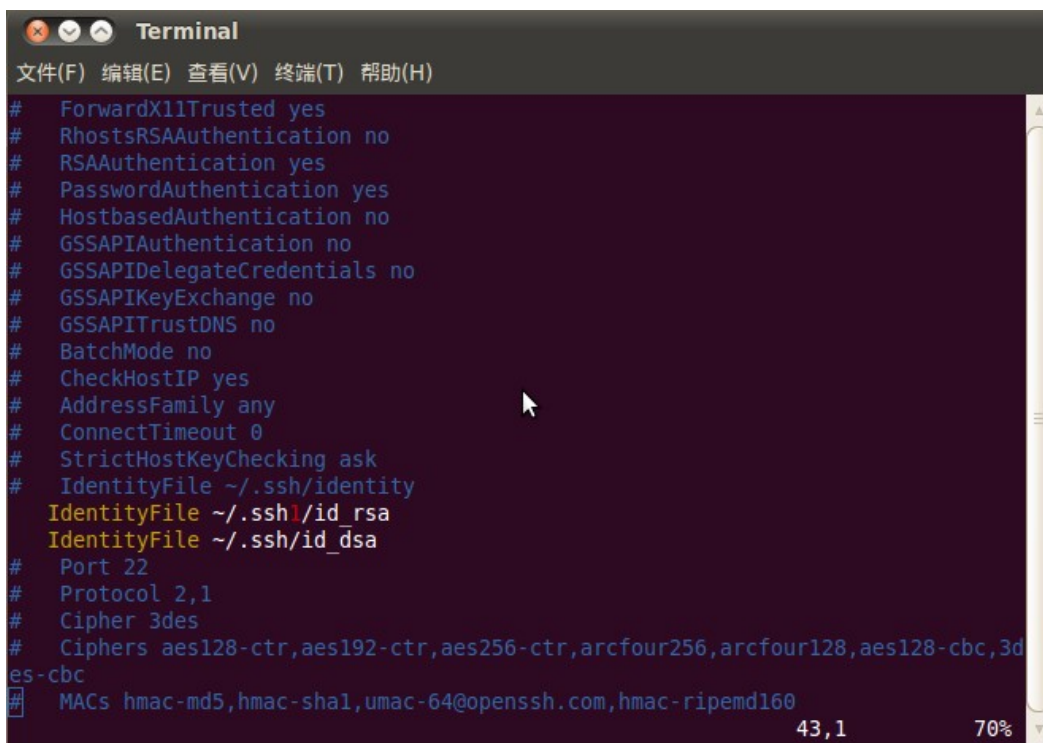


通过如下命令，配置当前用户的 ssh 配置。

```
~$ cp /etc/ssh/ssh_config ~/.ssh/config
```

```
~$ vi ~/.ssh/config
```

如图，将 ssh 使用另一个目录的文件“~/.ssh1/id_rsa”作为认证私钥。通过这种方法，可以切换不同的密钥。



附录 B.5 密钥权限管理

服务器可以实时监控某个 **key** 的下载次数、IP 等信息，如果发现异常将禁用相应的 **key** 的下载权限。

请妥善保管私钥文件。并不要二次授权与第三方使用。

附录 B.6 Git 权限申请说明

参考上述章节，生成公钥文件，发邮件至 fae@rock-chips.com，申请开通 SDK 代码下载权限。