crypto/hwrng linux 驱动开发指南

发布版本：V1.03

作者邮箱：

Troy Lin troy.lin@rock-chips.com

日期：2021.03

文件密级：公开资料

前言

概述

本文主要指导读者如何进行 kernel 4.4 上 crypto/hwrng 驱动开发

读者对象

本文档（本指南）主要适用于以下工程师：

linux kernel 开发者使用

修订记录

日期	版本	作者	修改说明
2019-12-10	V1.00	林金寒	初始版本
2020-08-06	V1.01	林金寒	更新硬件crypto启用配置
2020-08-06	V1.02	林金寒	增加user space调用硬件crypto说明
2021-03-04	V1.03	林金寒	1.RNG和CRYPTO兼容性修改说明 2.增加硬件CRYPTO性能说明

---

1. 概述

当前 RK 平台上 crypto IP 有两个版本，Crypto V1 和 Crypto V2，两个 IP 版本支持的算法不同，使用方式差异也较大。当前硬件随机数模块都是存在于硬件 Crypto IP 之中，而不是独立的硬件模块。

1.1 Crypto V1

算法	描述
DES/TDES	支持 ECB/CBC 两种模式，其中 TDES 支持 EEE 和 EDE 两种密钥模式
AES	支持 ECB/CBC/CTR 三种模式，支持 128/192/256 bit 三种密钥长度
HASH	支持 SHA1/SHA256/MD5。
RSA	支持 512/1024/2048 三种密钥长度。（RK3126、RK3128、RK3288 和 RK3368 不支持）
TRNG	支持 256bit 硬件随机数

1.2 Crypto V2

算法	描述
DES/TDES	支持 ECB/CBC/OFB/CFB 四种模式，其中 TDES 只支持 EDE 密钥模式。
AES	支持 ECB/CBC/OFB/CFB/CTR/CTS/XTS/CCM/GCM/CBC-MAC/CMAC。
HASH	支持 MD5/SHA1/SHA224/SHA256/SHA384/SHA512-224/SHA512-256 带硬件填充。
HMAC	支持 MD5/SHA1/SHA256/SHA512 带硬件填充。
RSA/ECC	支持最大 4096bit 的常用大数运算操作，通过软件封装该操作可实现 RSA/ECC 算法。
TRNG	支持 256bit 硬件随机数

1.3 各平台版本情况

各个芯片平台的 Crypto IP 版本如下：

采用 CRYPTO V1 的平台有：

- RK3399、RK3288、RK3368、RK3328/RK3228H、RK322x、RK3128、RK1108、RK3126

采用 CRYPTO V2 的平台有：

- RK3326/PX30、RK3308、RK1808、RV1126/RV1109、RK2206

2. 驱动代码说明

当前只维护 kernel 4.4 下的 crypto/hwrng 驱动。目前硬件 TRNG 模块都是属于 crypto 模块中，跟 crypto 模块共用 clock，且大部分芯片中 crypto IP 只有一个，因此 hwrng 驱动和 crypto 驱动不能同时打开，否则会出现 clock 冲突的情况，导致驱动无法正常使用。

2.1 hwrng

由于 hwrng 驱动比较简单，因此 Crypto V1 和 Crypto V2 两种平台都集中到同一个.c 文件中。

驱动中不区分具体的芯片型号，只按照"rockchip,cryptov1-rng"和"rockchip,cryptov2-rng"两种 compatible 进行划分。

驱动代码：drivers/char/hw_random/rockchip-rng.c

2.2 crypto

当前驱动实现的算法如下：

Crypto V1:

- **AES:** ECB/CBC
- **DES/TDES:** ECB/CBC
- **HASH:** SHA1/SHA256/MD5

Crypto V2:（驱动已经实现的算法列表，有些算法在某些平台上支持，请对照算法支持表）

- **AES:** ECB/CBC/CFB/CTR/XTS
- **DES/TDES:** ECB/CBC/CFB/OFB
- **SM4:** ECB/CBC/CFB/OFB/CTR/XTS
- **HASH:** SHA1/SHA256/SHA512/MD5/SM3
- **HMAC:** HMAC_SHA1/HMAC_SHA256/HMAC_SHA512/HMAC_MD5/HMAC_SM3
- **RSA:** 最大4096bit

Crypto V2硬件完整版（以下删除线部份模式驱动尚未实现）：

- **AES(128/192/256):** ECB/CBC/OFB/CFB/CTR/XTS/~~CTS~~/~~CCM~~/~~GCM~~/~~CBC-MAC~~/~~CMAC~~
- **SM4:** ECB/CBC/OFB/CFB/CTR/XTS/~~CTS~~/~~CCM~~/~~GCM~~/~~CBC-MAC~~/~~CMAC~~
- **DES/TDES:** ECB/CBC/OFB/CFB
- **HASH:** MD5/SHA-1/SHA256/SHA512/SM3/~~SHA224~~/~~SHA384~~/~~SHA512_224~~/~~SHA512_384~~
- **HMAC:** SHA-1/SHA-256/SHA-512/MD5 /SM3
- **RSA:** 4096bit PKA大数运算支持

Crypto V2硬件差异表

芯片平台	AES	DES/TDES	SM3/SM4	HASH	HMAC	RSA	基地址
3326/px30	√	√	×	√	√	√	0xFF0B0000
3308	√	√	×	√	√	√	0xFF2F0000
1808	AES-128 对驱动来说可以认为不支持	×	×	SHA-1/SHA-224/SHA-256/MD5	√	√	0xFF630000
rv1126/rv1109	AES-128/AES-256 由于不支持AES-192，因此AES-192部分只能通过软算法实现，但是软算法不能支持硬算法的所有模式。因此建议不要去改动代码里已配置好的算法列表。	√	√	√	√	√	0xFF500000
RK2206	√	√	×	√	√	√	0x43020000

驱动相关文件如下：

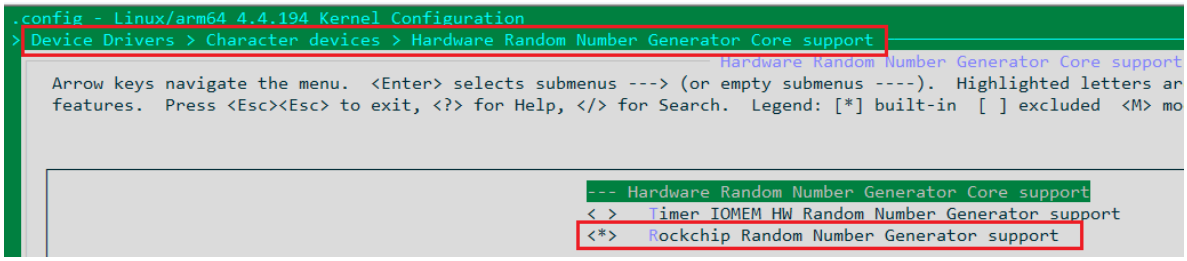
```
drivers/crypto/rockchip
|-- rk_crypto_bignum.c          // crypto大数操作接口
|-- rk_crypto_bignum.h          // crypto大数操作头文件
|-- rk_crypto_core.c            // linux crypto 驱动框架及公用接口，注册硬件crypto算
法到内核
|-- rk_crypto_core.h            // linux crypto公用头文件
|-- rk_crypto_v1.h              // crypto v1结构体定义及接口声明
|-- rk_crypto_v1_ablkcipher.c   // crypto v1硬件加解密算法实现
|-- rk_crypto_v1_ahash.c        // crypto v1硬件HASH算法实现
|-- rk_crypto_v1_reg.h          // crypto v1硬件寄存器定义
|-- rk_crypto_v2.h              // crypto v2结构体定义及接口声明
|-- rk_crypto_v2_ablkcipher.c   // crypto v2硬件加解密算法实现
|-- rk_crypto_v2_ahash.c        // crypto v2硬件HASH算法实现
|-- rk_crypto_v2_akcipher.c     // crypto v2硬件RSA算法实现
|-- rk_crypto_v2_pka.c          // crypto v2硬件pka大数运算实现
`-- rk_crypto_v2_reg.h          // crypto v2硬件寄存器定义
```

3. 如何启用硬件 hwrng

3.1 Menuconfig 配置

要调用到 hwrng 驱动需要在 menuconfig 里面进行配置，目前在开发分支里面已经默认配置好，开启和关闭由板级 dts 文件来控制。

配置如下列图所示（红色标记表示配置路径和需要配置的选项）：



或在 config 文件（rockchip_defconfig 中已默认配置好）中添加如下语句：

```
CONFIG_HW_RANDOM=y
CONFIG_HW_RANDOM_ROCKCHIP=y
```

3.2 板级 dts 文件配置

当前大部分芯片 dtsi 都已配置好 hwrng 节点，只需在板级 dts 中将 rng 模块使能即可，如下所示：

```
&rng {
    status = "okay";
}
```

3.3 未支持芯片 dtsi 文件配置

当前大部分芯片平台均已配置好 rng 节点，如果 dtsi 未配置好 hwrng 节点，可以参考以下方式进行配置。

注意：

1. rng 基地址需要根据芯片 TRM 进行修改，rng 基地址即 CRYPTO 基地址
2. clocks 的宏不同平台可能略有不同，如果 dts 出现报错，可以去 include/dt-bindings/clock 目录下，grep -rn CRYPTO 查找对应的 clock 宏名称，如下所示：

```
troy@inno:~/kernel/include/dt-bindings/clock$ grep -rn CRYPTO
rk3328-cru.h:57:#define SCLK_CRYPTO             59
rk3328-cru.h:206:#define HCLK_CRYPTO_MST        336
rk3328-cru.h:207:#define HCLK_CRYPTO_SLV        337
rk3328-cru.h:284:#define SRST_CRYPTO            68
```

Crypto V1:

```
rng: rng@ff060000 {
    compatible = "rockchip,cryptov1-rng";
    reg = <0x0 0xff060000 0x0 0x4000>;
    clocks = <&cru SCLK_CRYPTO>, <&cru HCLK_CRYPTO_SLV>;
    clock-names = "clk_crypto", "hclk_crypto";
    assigned-clocks = <&cru SCLK_CRYPTO>, <&cru HCLK_CRYPTO_SLV>;
    assigned-clock-rates = <150000000>, <100000000>;
    status = "disabled";
};
```

Crypto V2:

实际 TRNG 不需要依赖全部的 clock，只需依赖 hclk_crypto 一个即可

```
rng: rng@ff500400 {
    compatible = "rockchip,cryptov2-rng";
    reg = <0xff500400 0x80>;
    clocks = <&cru HCLK_CRYPTO>;
    clock-names = "hclk_crypto";
    power-domains = <&power RV1126_PD_CRYPTO>;
    resets = <&cru SRST_CRYPTO_CORE>;
    reset-names = "reset";
    status = "disabled";
};
```

3.4 如何确定 hwrng 已启用

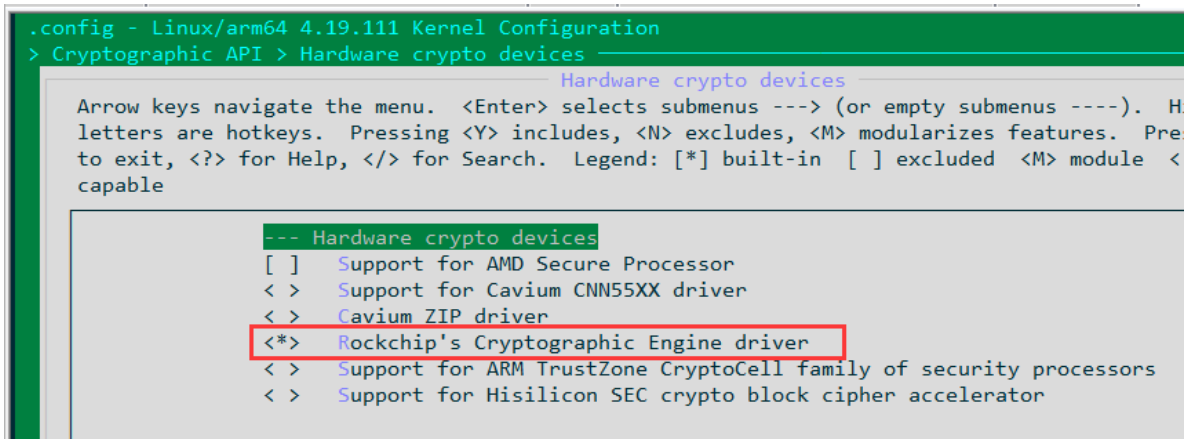
1. 执行 cat /sys/devices/virtual/misc/hw_random/rng_current 可以看到信息为 rockchip，确定当前调用的是硬件驱动
2. linux：执行 cat /dev/hwrng | od -x | head -n 1 可以获取到一行随机数，每次执行，随机数的内容都不相同
3. Android：执行 cat /dev/hw_random | od -x | head -n 1 可以获取到一行随机数，每次执行，随机数的内容都不相同

4. 如何启用硬件 Crypto

当前驱动代码当中只添加了 rk3288 和 px30 两种芯片平台，分别属于 crypto v1 和 crypto v2。

4.1 Menuconfig 配置

在menuconfig配置中使能Rockchip加解密驱动支持，在 dts 中会自动根据芯片平台 compatible id 进行自动适配v1或者v2。



或在 config 文件中添加如下语句：

> CONFIG_CRYPTO_DEV_ROCKCHIP=y

4.2 板级 dts 文件配置

注意：crypto和rng不能同时设置成okay，否则驱动无法正常工作。

对于 rk3288/rk3326/px30，直接在板级 dts 文件中开启 crypto 模块即可，如下所示：

```
&crypto {
    status = "okay";
};
```

4.3 新增芯片平台支持

1. 确定芯片 crypto IP 的版本是 v1 还是 v2
2. drivers/crypto/rockchip/rk_crypto_core.c 中添加对应的algs_name， soc_data，compatible 等信息。

```
/* 增加芯片支持的算法信息，px30属于crypto v2，支持的算法参见crypto_v2_algs */
/* 特别注意：crypto_v2_algs为crypto v2支持的所有算法。*/
/* 某些芯片在crypto v2上做了些裁剪，如rk1808不支持SHA512算法，因此需要对比TRM确认支持的算法 */
static char *px30_algs_name[] = {
    "ecb(aes)", "cbc(aes)", "xts(aes)",
    "ecb(des)", "cbc(des)",
    "ecb(des3_ede)", "cbc(des3_ede)",
    "sha1", "sha256", "sha512", "md5",
};

/* 绑定px30_algs_name到px30_soc_data */
static const struct rk_crypto_soc_data px30_soc_data =
    RK_CRYPTO_V2_SOC_DATA_INIT(px30_algs_name, false);

/* 绑定px30_soc_data到id_table */
static const struct of_device_id crypto_of_id_table[] = {
    /* crypto v2 in belows */
    {
        .compatible = "rockchip,px30-crypto",
        .data = (void *)&px30_soc_data,
    },
    {
        .compatible = "rockchip,rv1126-crypto",
        .data = (void *)&rv1126_soc_data,
    },
    /* crypto v1 in belows */
    {
        .compatible = "rockchip,rk3288-crypto",
        .data = (void *)&rk3288_soc_data,
    },
    { /* sentinel */ }
};
```

3. 芯片 dtsi 增加 crypto 配置

注意：

1. 根据芯片 TRM 进行修改确定CRYPTO 基地址
2. clocks 的宏不同平台可能略有不同，如果 dts 出现报错，可以去 include/dt-bindings/clock 目录下，grep -rn CRYPTO 查找对应的 clock 宏名称，如下所示：

```
troy@inno:~/kernel/include/dt-bindings/clock$ grep -rn CRYPTO
rk3328-cru.h:57:#define SCLK_CRYPTO             59
rk3328-cru.h:206:#define HCLK_CRYPTO_MST        336
rk3328-cru.h:207:#define HCLK_CRYPTO_SLV        337
rk3328-cru.h:284:#define SRST_CRYPTO            68
```

Crypto V1:

```
crypto: cypto-controller@ff8a0000 {                 /* 根据实际配置crypto基地址*/
    compatible = "rockchip,rk3288-crypto";          /* 修改芯片平台，如"rk3399-crypto" */
    reg = <0x0 0xff8a0000 0x0 0x4000>;              /* 根据实际配置crypto基地址 */
    interrupts = <GIC_SPI 48 IRQ_TYPE_LEVEL_HIGH>;  /* 根据实际配置crypto中断号 */
    clocks = <&cru ACLK_CRYPTO>, <&cru HCLK_CRYPTO>,
             <&cru SCLK_CRYPTO>, <&cru ACLK_DMAC1>;
    clock-names = "aclk", "hclk", "sclk", "apb_pclk";
    resets = <&cru SRST_CRYPTO>;
    reset-names = "crypto-rst";
    status = "disabled";
};
```

Crypto V2:

对于大部分CRYPTO V2芯片，RNG的寄存器地址位于CRYPTO中间，因此配置reg时，需要将CRYPTO的地址空间拆分成两个部分，第一部分为CIPHER使用的寄存器，第二部分为RSA使用的寄存器

---------- reg map ----------

| cipher/hash | rng | pka |

```
crypto: crypto@ff500000 {                           /* 根据实际配置crypto基地址 */
    compatible = "rockchip,rv1126-crypto";          /* 修改芯片平台，如"rv1126-crypto" */
    reg = <0xff500000 0x400>, <0xff500480 0x3b80>;  /* 根据实际配置crypto基地址 */
    interrupts = <GIC_SPI 3 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&cru CLK_CRYPTO_CORE>, <&cru CLK_CRYPTO_PKA>,
    <&cru ACLK_CRYPTO>, <&cru HCLK_CRYPTO>;
    clock-names = "aclk", "hclk", "sclk", "apb_pclk";
    power-domains = <&power RV1126_PD_CRYPTO>;
    resets = <&cru SRST_CRYPTO_CORE>;
    reset-names = "crypto-rst";
    status = "disabled";
};
```

4. 板级 dts 配置 crypto 开启

```
&crypto {
    status = "okay";
};
```

5.user space调用硬件 Crypto

user space使用netlink导出的crypto内核接口，使用socket形式进行调用。为了方便使用，可以采用libkcapi库，屏蔽底层socket操作，简化调用方式。

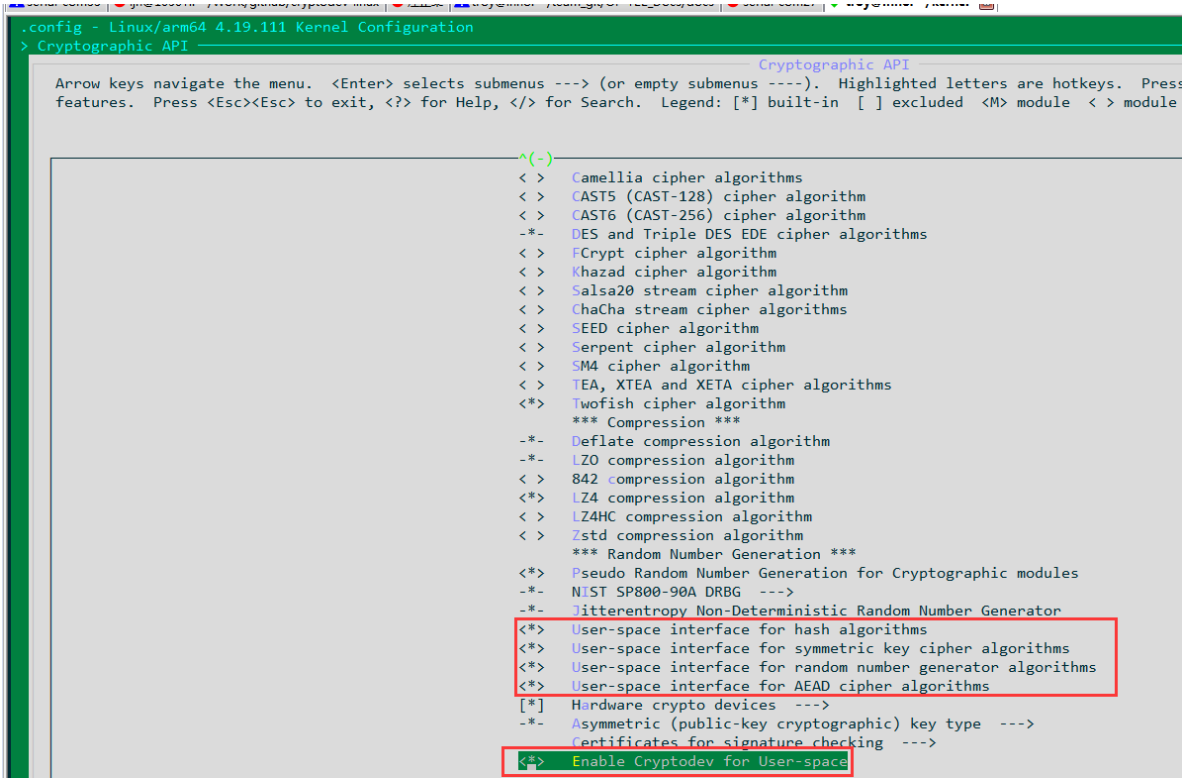
通过命令`cat /proc/crypto | grep rk`可以查看系统注册的RK硬件crypto算法。（以rv1126为例）

```
driver       : pkcs1pad(rsa-rk,sha256)
driver       : rsa-rk
driver       : hmac-sm3-rk
driver       : hmac-md5-rk
driver       : hmac-sha512-rk
driver       : hmac-sha256-rk
driver       : hmac-sha1-rk
driver       : sm3-rk
driver       : md5-rk
driver       : sha512-rk
driver       : sha256-rk
driver       : sha1-rk
driver       : ofb-des3_ede-rk
driver       : cfb-des3_ede-rk
driver       : cbc-des3_ede-rk
driver       : ecb-des3_ede-rk
driver       : ofb-des-rk
driver       : cfb-des-rk
driver       : cbc-des-rk
driver       : ecb-des-rk
driver       : xts-aes-rk
driver       : ctr-aes-rk
driver       : cfb-aes-rk
driver       : cbc-aes-rk
driver       : ecb-aes-rk
driver       : xts-sm4-rk
driver       : ctr-sm4-rk
driver       : ofb-sm4-rk
driver       : cfb-sm4-rk
driver       : cbc-sm4-rk
driver       : ecb-sm4-rk
```

5.1 Netlink 配置

kernel默认不导出netlink的crypto内核接口，需要自行打开。先按照章节《4. 如何启用硬件 Crypto》配置好硬件crypto，确保硬件crypto驱动加载正常。

在menuconfig配置



或在 config 文件中添加如下语句：

```
CONFIG_CRYPTO_USER=y
CONFIG_CRYPTO_USER_API_HASH=y
CONFIG_CRYPTO_USER_API_SKCIPHER=y
CONFIG_CRYPTO_USER_API_AEAD=y
CONFIG_CRYPTO_USER_API_RNG=y
CONFIG_CRYPTO_DEV_USER_API=y
```

5.2 Kcapi安装和使用

安装：

- 安装命令安装，如Debian系统执行`sudo apt-get install libkcapi-dev`
- 源码安装https://github.com/smuellerDD/libkcapi

后续会默认集成到SDK中，直接即可调用。

使用：

示例调用demo参见 https://github.com/smuellerDD/libkcapi/tree/master/test。编译时包含`kcapi.h`头文件，并链接`lkcapi`库。使用时，最好使用rk的drivername作为算法名称（如ecb-des-rk）可以保证调用到RK的硬件crypto算法。

5.3 kcapi调用性能

使用kcapi通过内核crypto框架调用硬件算法（以rv1126 crypto v2平台为例）。由于不同平台上硬件算法差距较小，而软件算法由于CPU主频，DDR频率的不同，不同芯片上差距较大，因此此处不列出软件算法调用性能。

本测试由libkcapi源码中的speed_test测试代码修改而来，测试分块大小64M。（RSA测试数据待补充）

测试环境（kernel kcapi rv1126）：

时钟：CRYPTO_CORE = 200M

算法	实测(MBps) 加密/解密
DES3-ECB	25.60 / 24.84
DES3-CBC	24.36 / 23.49
AES-256-ECB	142.2 / 131.31
AES-256-CBC	80.64 / 136.41
SM4-ECB	118.87 / 110.3
SM4-CBC	49.23 / 105.10
SM3	21.62
SHA-256	43.76
SHA512	41.41

6.uboot层硬件 Crypto性能测试

CRYPTO V1性能测试

测试环境（uboot rk3399）：

时钟：CRYPTO_CORE = 200M，不同芯片的最高频率略有不同

CIPHER/HASH算法性能测试：

算法	实测值(MBps)	理论值(MBps)
DES	待补充	<=94
TDES	待补充	<=31
AES-128	待补充	<=290
AES-192	待补充	<=246
AES-256	待补充	<213
MD5	125	<196
SHA1	125	<158
SHA256	125	-

RSA算法性能测试：

RSA算法长度(nbits)	公钥加密/私钥解密 (ms)
2048	8 / 632

CRYPTO V2性能测试

测试环境（uboot rv1126）：

时钟：CRYPTO_CORE = 200M，CRYPTO_PKA=300M, DDR=786M

Hash/HMAC：总共测试128M的数据，每次计算4M的数据

DES/3DES/AES/SM4：总共测试128M数据，每次计算4M的明文和4M的aad数据

算法	模式	实测值(MBps)	理论值(MBps)
HASH/HMAC	MD5	183	196
	SHA1	148	158
	SHA256/224	183	196
	SHA512/384/512_224/512_256	288	316
	SM3	183	-
DES	ECB	289	352
	CBC/CFB/OFB	79	88
3DES	ECB	107	116
	CBC/CFB/OFB	27	29
AES (128 | 192 | 256)	ECB/CTR/XTS	447 | 442 | 436	1066 | 914 | 800
	CBC/CFB/OFB/CTS	234 | 204 | 180	266 | 228 | 200
	CMAC/CBC_MAC	245 | 212 | 186	266 | 228 | 200
	CCM(data+aad)	180 | 162 | 146	-
	GCM(data+aad)	196 | 184 | 174	-
SM4	ECB/CTR/XTS	320	-
	CBC/CFB/OFB/CTS	87	-
	CMAC/CBC_MAC	89	-
	CCM(data+aad)	156	-
	GCM(data+aad)	114	-

RSA测试方法：生成rsa key，包含n, e, d，执行加密和解密测试

加密测试：密文 = d ^e % n

解密测试：明文 = d^d % n

算法	公钥加密/私钥解密	时间(ms)
RSA-1024	加密	小于1
	解密	12
RSA-2048	加密	1
	解密	93
RSA-3072	加密	1
	解密	304
RSA-4096	加密	2
	解密	710

References

附录

术语