phd 2 Positive Hack Days Fest

от ■ positive technologies

Rock eBOOT

или как зашифровать апельсинку

Фирсов Никита

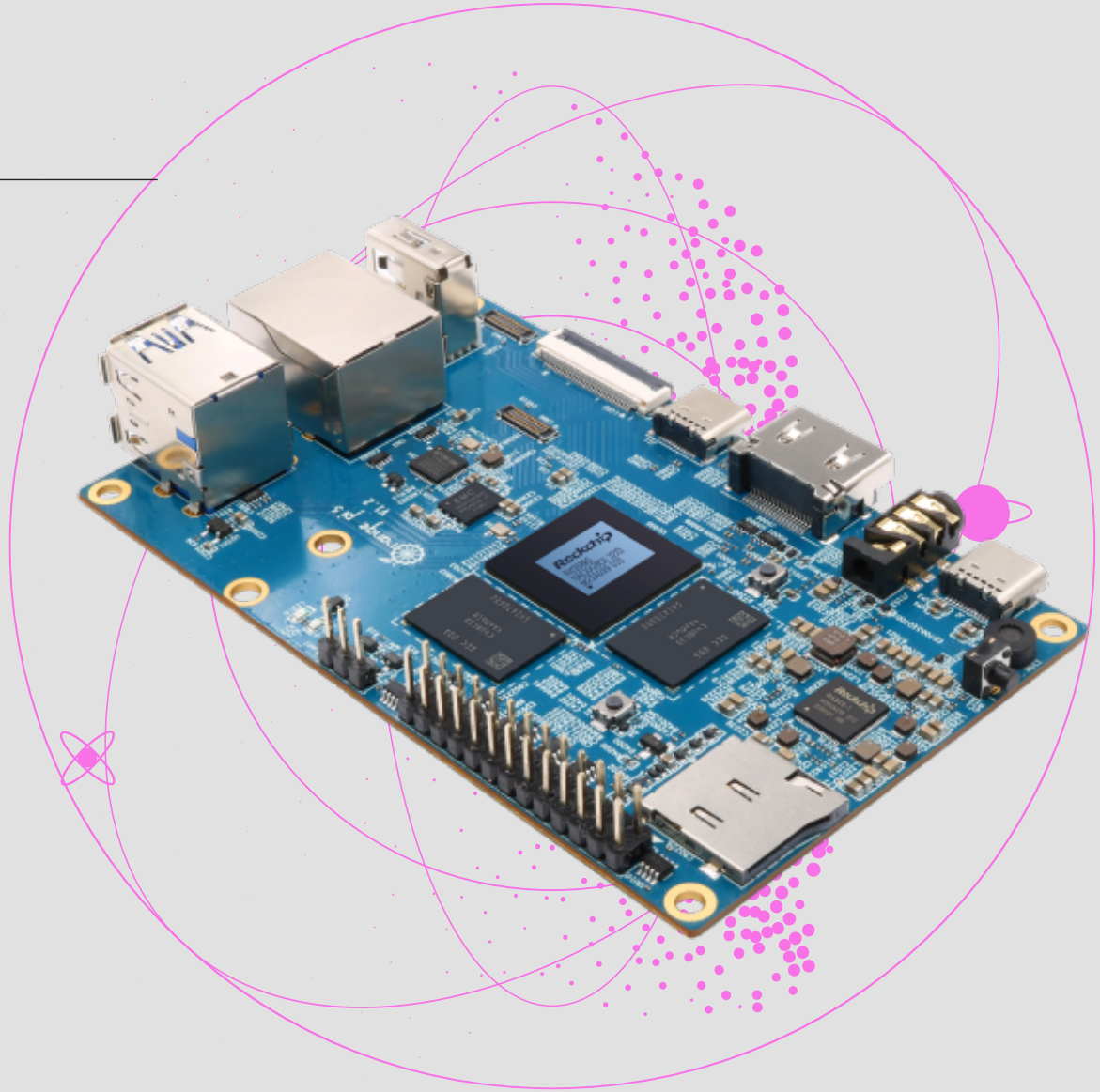
Специалист Positive Labs

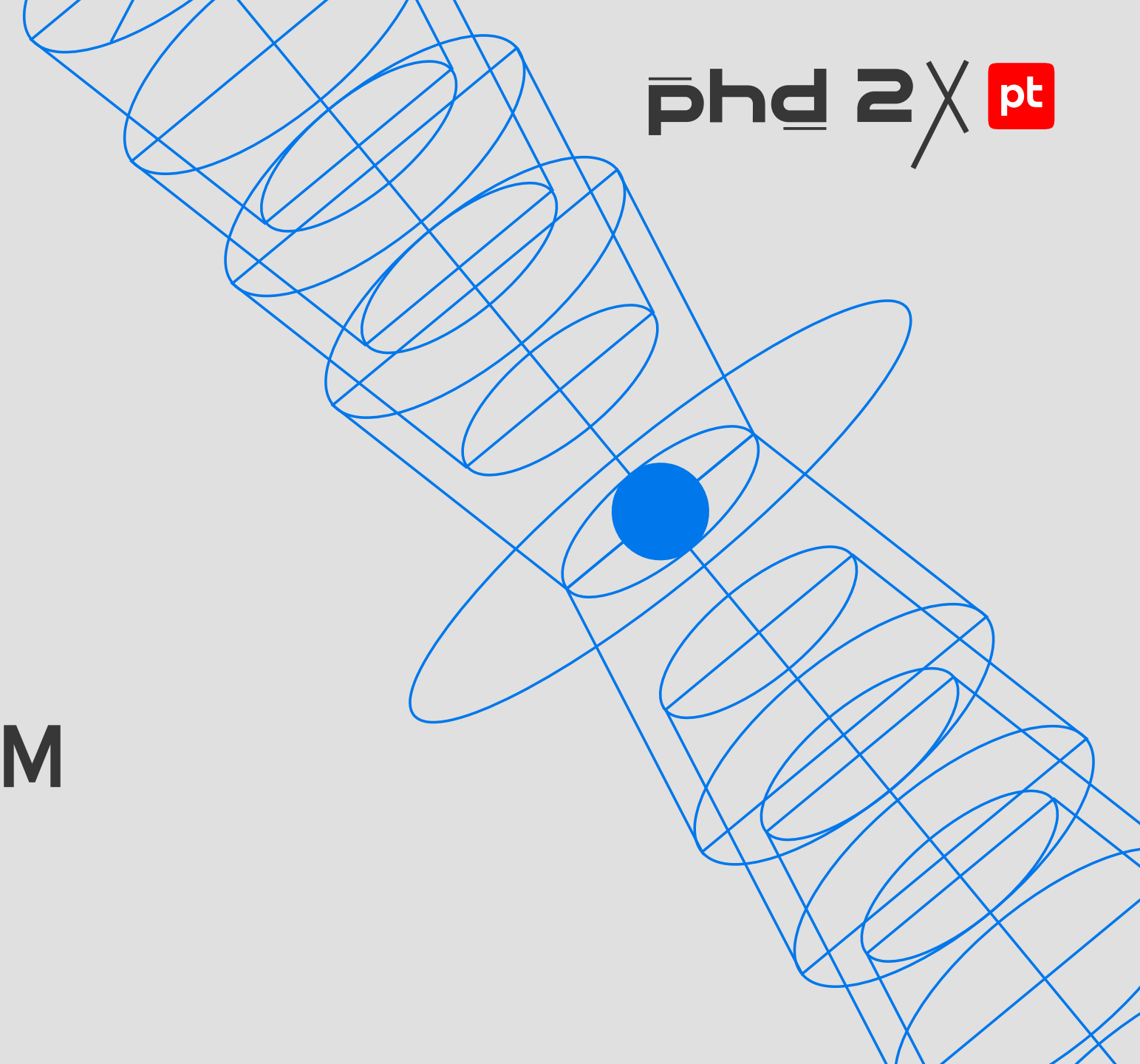
About me

Фирсов Никита

- Специалист POSITIVE LABS
- В основном занимаюсь исследованием “железок”, но всякое бывает.
- Tg: @Ahteesh0

Agenda





Intro

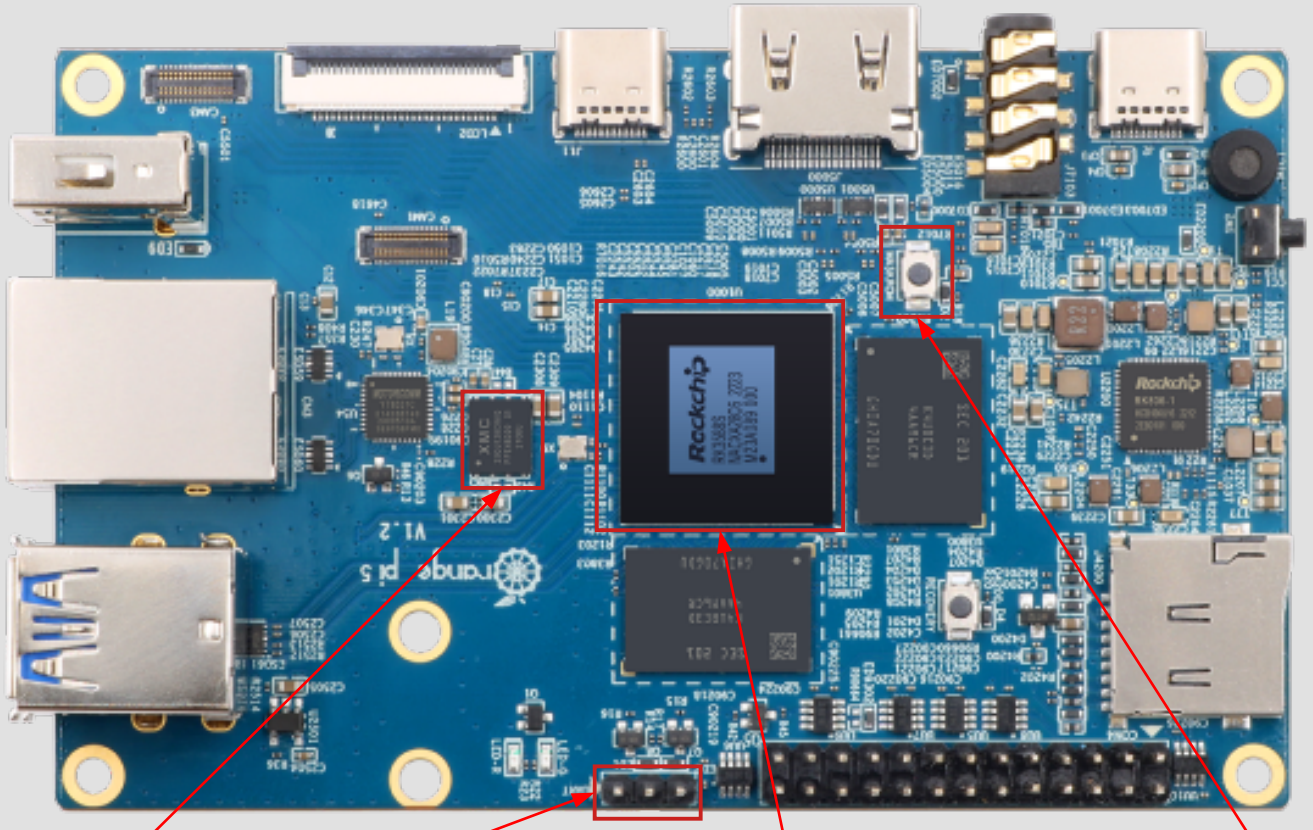
что, где, зачем

00

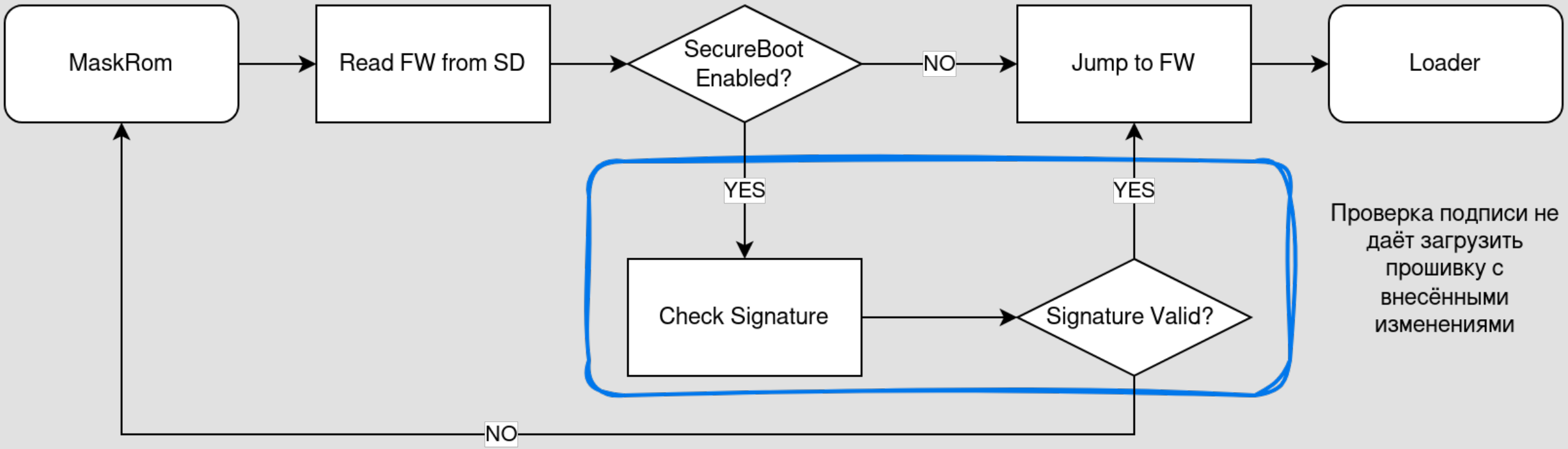
Технологии защиты
secure & encrypted boot

Я использовал Orange Pi 5, но результаты в той или иной мере применимы к любым устройствам на основе RK3588(S)

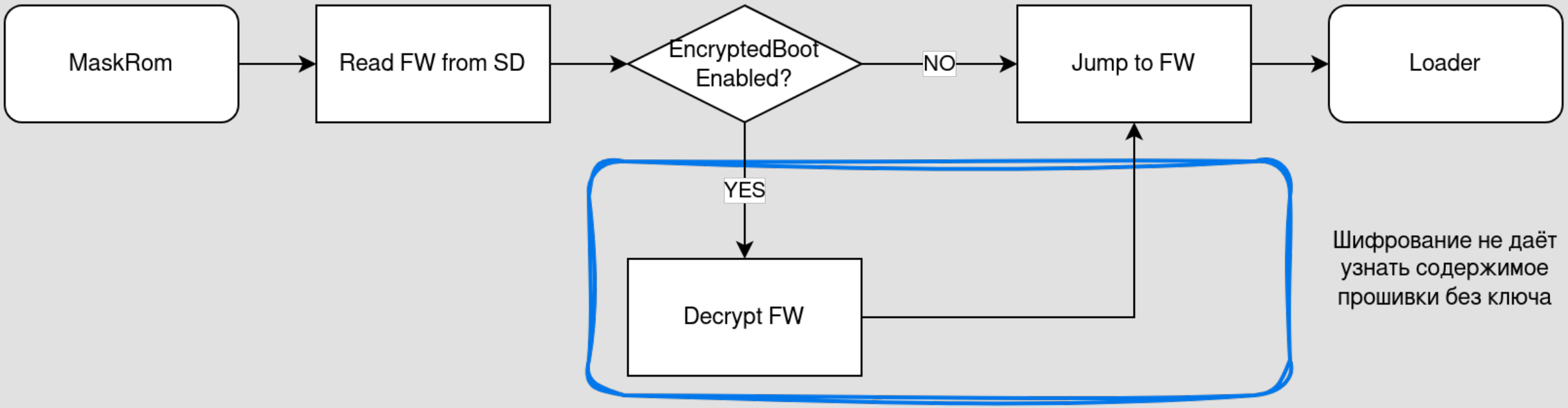
Orange Pi 5



SecureBoot
основа основ



EncryptedBoot
хочешь спрятать - зашифруй

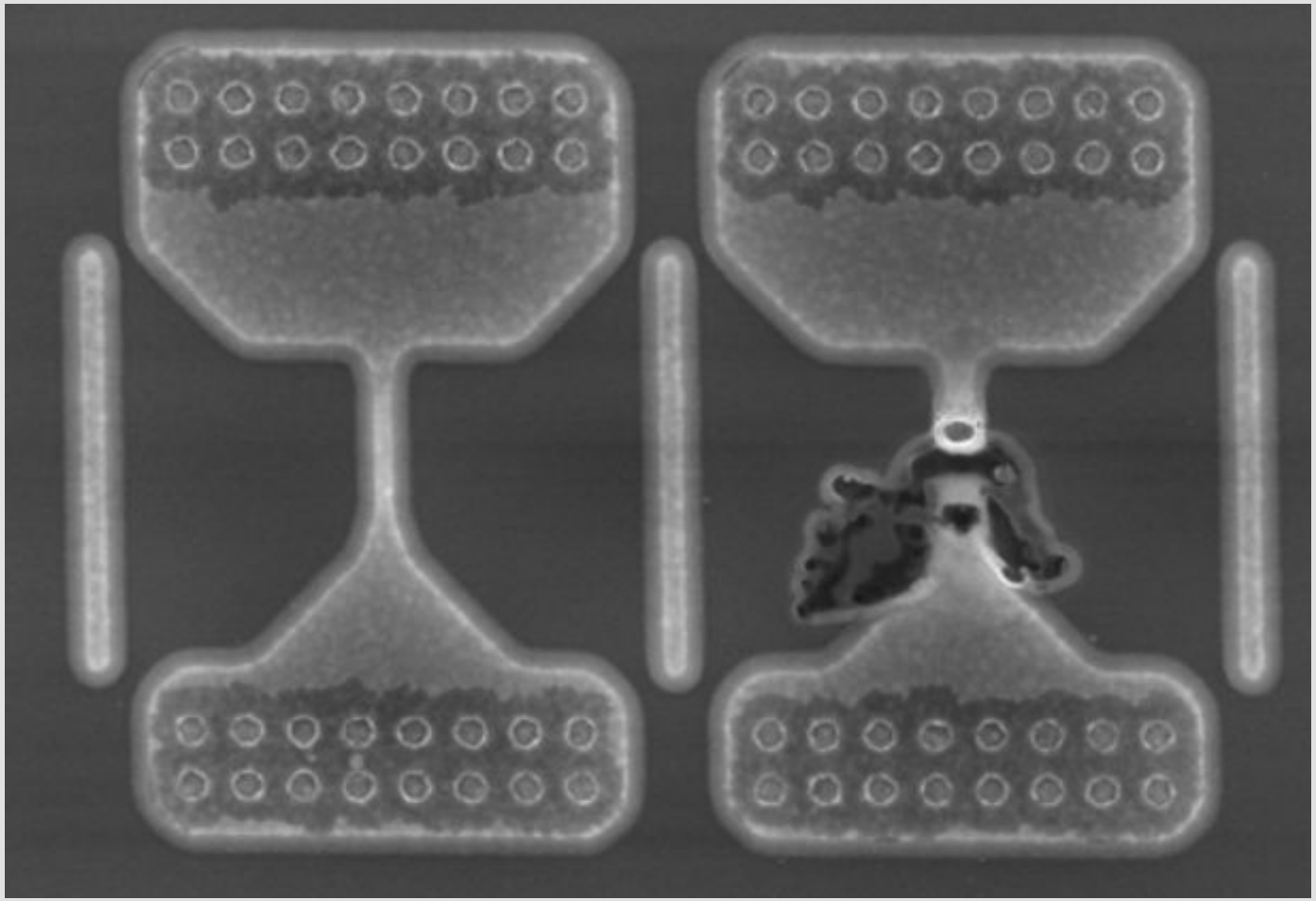


OTP
они же фьюзы

One-Time Programmable

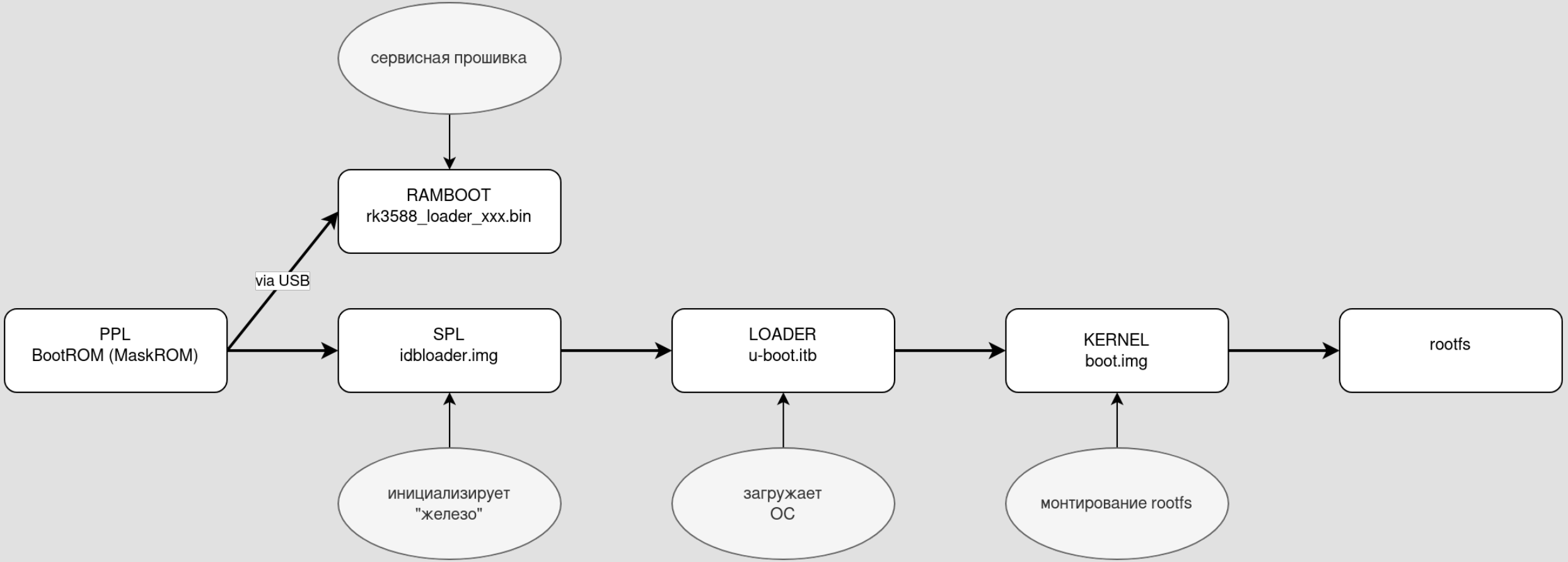
Можно записать только 1 раз

Обычно это правило применимо к отдельным битам, т.е. можно поменять любой 0 на 1, но нельзя поменять 1 на 0



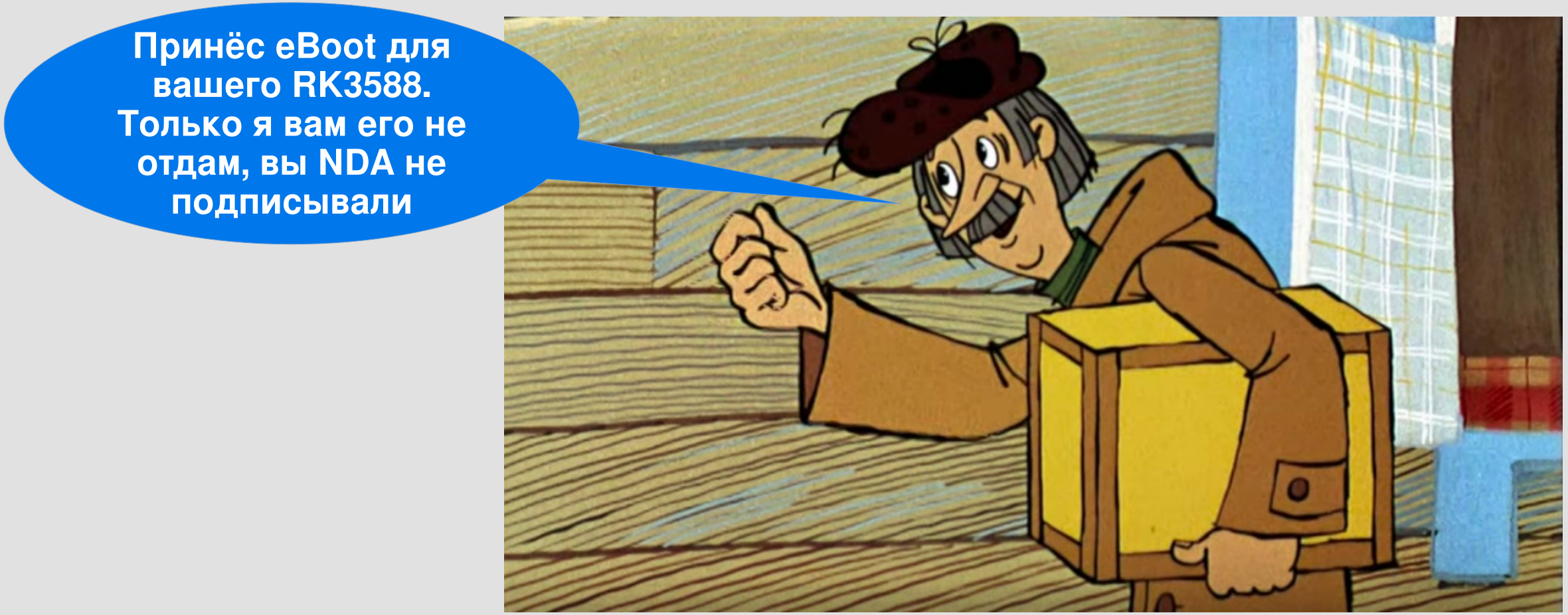
0 1

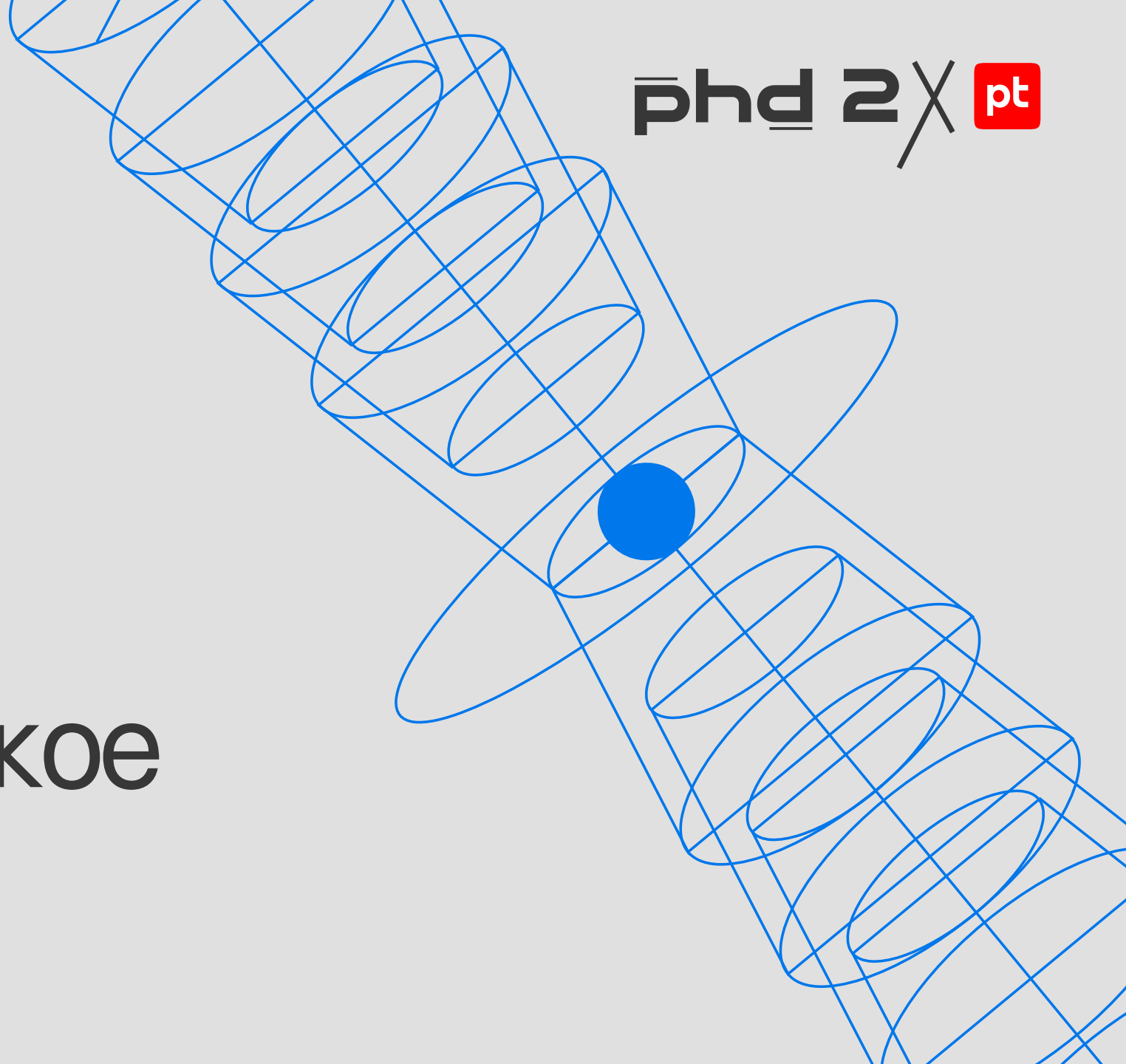
Boot sequence
загрузка шаг за шагом



NDA hell

технология у нас есть...





SecureBoot

не пущать чужое

01

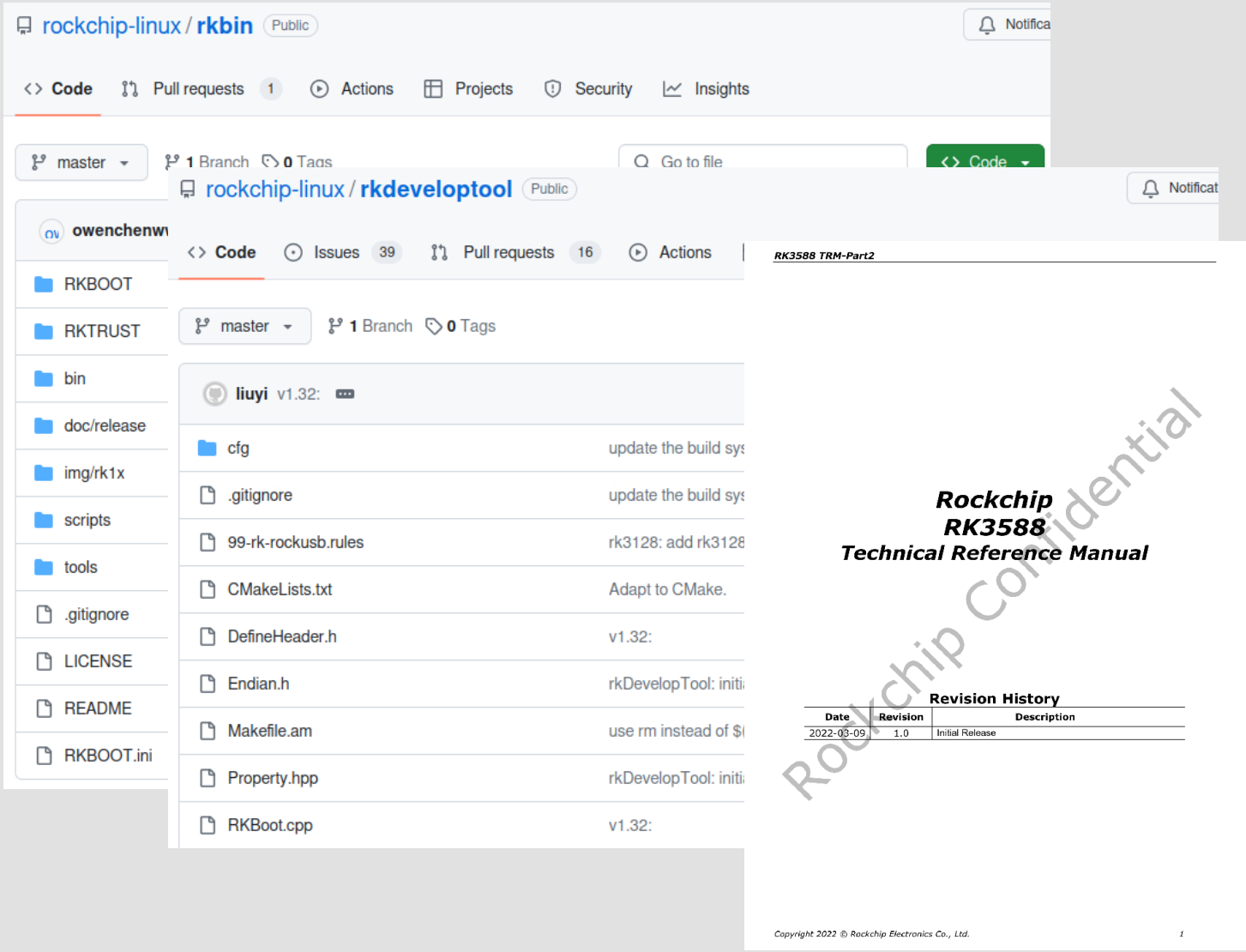
Tools & Docs
не всё так плохо

Tools:

- rkbin
- rkdeveloptool

Docs:

- Datasheet
- Technical Reference Manual



MaskROM
как его достать

DumpROM.s

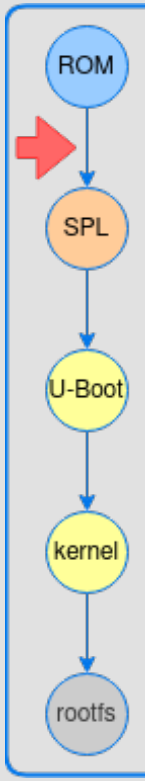
```
BASE=0x30076d8
hexdump=0x300A644
usleap=0x300439C

start:
MOV     X19, 0xffff0000
MOV     X20, 0xffff8000

step:
ADR     X0, .fmt
MOV     X1, X19
MOV     W2, 0x1
MOV     W3, 0x100
BL      (hexdump+start-BASE)

MOV     W0,0x10000
BL      (usleap+start-BASE)
ADD     X19, X19, 0x100
CMP     X19, X20
BNE     step

loop:
B loop

.fmt:
.ascii "dump: \x00"
```

The following table show the boot address when before remap and after remap

Table 1-2 Address Remapping

remap[1:0]=2'b00		remap[1:0]=2'b01		remap[1:0]=2'b10	
		not accessible	BootRom(32KB)	not accessible	BootRom(32KB)
0xFFFF0000	BootRom(32KB)	0xFFFF0000	PMU_SRAM(8KB)	0xFFFF0000	SYSTEM_SRAM (256KB)
0xFF100000	PMU_SRAM(8KB)	0xFF100000	PMU_SRAM(8KB)	0xFF100000	PMU_SRAM(8KB)
0xFF000000	SYSTEM_SRAM(1MB)	0xFF000000	SYSTEM_SRAM(1MB)	0xFF000000	SYSTEM_SRAM(1MB)

```
Boot1 Release Time: Feb 24 2022 10:23:56, version: 1.05 USB BOOT
ChipType = 0x32, 469
SecureMode = 0
atags_set_bootdev: ret:(0)
UsbBoot ...1050
powerOn 3334
dump:  00000000ffff0000 + 0x0:0xa0,0x00,0x38,0xd5,0x00,0x3c,0x40,0x92,0x1f,0x00,0x00,0xf1,0xa0,0x01,0x00,0x54,
dump:  00000000ffff0000 + 0x10:0x80,0x06,0x00,0x58,0x21,0x00,0x80,0x52,0x01,0x00,0x00,0xb9,0x5f,0x20,0x03,0xd5,
dump:  00000000ffff0000 + 0x20:0x82,0x06,0x00,0x18,0xe0,0x05,0x00,0x58,0x01,0x00,0x40,0xb9,0x3f,0x00,0x02,0x6b,
dump:  00000000ffff0000 + 0x30:0x61,0xff,0xff,0x54,0xa1,0x05,0x00,0x58,0x20,0x00,0x40,0xb9,0x00,0x00,0x1f,0xd6,
dump:  00000000ffff0000 + 0x40:0xe0,0x03,0x1f,0xaa,0xe1,0x03,0x1f,0xaa,0xe2,0x03,0x1f,0xaa,0xe3,0x03,0x1f,0xaa,
dump:  00000000ffff0000 + 0x50:0xe4,0x03,0x1f,0xaa,0xe5,0x03,0x1f,0xaa,0xe6,0x03,0x1f,0xaa,0xe7,0x03,0x1f,0xaa,
dump:  00000000ffff0000 + 0x60:0xe8,0x03,0x1f,0xaa,0xe9,0x03,0x1f,0xaa,0xea,0x03,0x1f,0xaa,0xeb,0x03,0x1f,0xaa,
dump:  00000000ffff0000 + 0x70:0xec,0x03,0x1f,0xaa,0xed,0x03,0x1f,0xaa,0xee,0x03,0x1f,0xaa,0xef,0x03,0x1f,0xaa,
dump:  00000000ffff0000 + 0x80:0xf0,0x03,0x1f,0xaa,0xf1,0x03,0x1f,0xaa,0xf2,0x03,0x1f,0xaa,0xf3,0x03,0x1f,0xaa,
dump:  00000000ffff0000 + 0x90:0xf4,0x03,0x1f,0xaa,0xf5,0x03,0x1f,0xaa,0xf6,0x03,0x1f,0xaa,0xf7,0x03,0x1f,0xaa,
dump:  00000000ffff0000 + 0xa0:0xf8,0x03,0x1f,0xaa,0xf9,0x03,0x1f,0xaa,0xfa,0x03,0x1f,0xaa,0xfb,0x03,0x1f,0xaa,
dump:  00000000ffff0000 + 0xb0:0xfc,0x03,0x1f,0xaa,0xfd,0x03,0x1f,0xaa,0xfe,0x03,0x1f,0xaa,0xa0,0x00,0x00,0x58,
dump:  00000000ffff0000 + 0xc0:0x1f,0x00,0x00,0x91,0xa0,0x00,0x00,0x58,0x00,0xc0,0x1e,0xd5,0x09,0x03,0x00,0x14,
dump:  00000000ffff0000 + 0xd0:0x00,0x10,0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x48,0xff,0xff,0x00,0x00,0x00,0x00,
dump:  00000000ffff0000 + 0xe0:0x04,0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x08,0x00,0x00,0xff,0x00,0x00,0x00,0x00,
dump:  00000000ffff0000 + 0xf0:0xaf,0xbe,0xad,0xde,0x00,0x00,0x00,0x00,0x03,0x00,0x01,0x2a,0x63,0x04,0x00,0x12,
```



rk_sign_tool
подписанный заголовок

unsigned

signed

магическая константа

нули

хэш

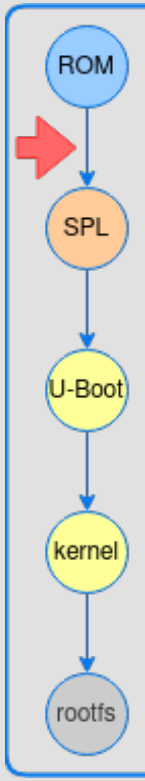
```
00000000  52 4b 4e 53 00 00 00 00  80 01 02 00 01 00 00 00  |RKNS............|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000070  00 00 00 00 00 00 00 00  04 00 80 00 ff ff ff ff  |................|
00000080  00 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  47 34 18 5d a5 53 34 24  14 ea 4f 1c a9 f7 5e 3d  |G4.].S4$..O...^=|
000000a0  1f 44 d8 07 07 90 c7 ef  80 6a e3 df 9f a5 fb 1f  |.D.......j......|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000000d0  84 00 b8 01 ff ff ff ff  00 00 00 00 02 00 00 00  |................|
000000e0  00 00 00 00 00 00 00 00  6d 89 7e 03 47 c9 f4 ba  |........m.~.G...|
000000f0  af e4 3a 2b 6c 56 25 49  17 a6 f8 b9 78 7d 77 63  |..:+lV%I....x}wc|
00000100  8c 75 a8 24 06 49 00 ad  00 00 00 00 00 00 00 00  |.u.$.I..........|
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000600  38 14 14 9b cb 32 82 bf  ff 2e 57 90 bc 95 4f b2  |8....2....W...O.|
00000610  ab 9f 39 cc 42 57 73 75  12 3b 00 14 cb ee 38 71  |..9.BWsu.;....8q|
00000620  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000800  01 00 00 14 78 3b 00 14  01 40 a0 d2 02 00 40 b9  |....x;...@....@.|
```

```
00000000  52 4b 53 53 00 00 00 00  80 01 02 00 11 00 00 00  |RKSS............|
00000200  a7 1c a5 e3 6b 3c d5 bc  d1 9c 54 fc 31 f3 8d 74  |....k<....T.1..t|
00000210  a2 56 9e 94 0e b6 94 c6  eb a7 5d 2e e2 0f 2b b0  |.V........]...+.|
00000220  e7 98 74 c4 5d 73 f5 51  54 0d 04 90 d6 fd ba ac  |..t.]s.QT.......|
00000230  89 62 47 6d 50 b3 3c 62  27 22 f7 15 93 62 12 13  |.bGmP.<b'"...b..|
00000240  4c 67 ba 5c cb 1d 27 b6  a8 44 21 44 ce d8 00 42  |Lg.\..'..D!D...B|
00000250  46 2a a4 70 87 fd 8e a7  b1 64 1f 11 64 fe e1 62  |F*.p.....d..d..b|
00000260  8c 8e 30 9f a3 f2 a8 23  51 0c 11 21 e7 58 ef e9  |..0....#Q..!.X..|
00000270  9f c7 7b cf 4e 3e 7d 44  e3 b2 25 15 71 7f 71 59  |..{.N>}D..%.q.qY|
00000280  ed b8 00 44 5b 63 67 10  fd ba 87 63 5b 4b 46 9b  |...D[cg....c[KF.|
00000290  35 04 c4 00 5c e1 d4 7d  6b e6 2f 6b 5b 3c 70 d6  |5...\..}k./k[<p.|
000002a0  3b ee 60 de 5b 2a b2 27  f0 d2 0e 7e 0c 44 d7 a5  |;.`.[*.'...~.D..|
000002b0  97 04 0d 5b e9 56 c8 e5  fc df 43 ca 71 a0 10 22  |...[.V....C.q.."|
000002c0  82 da 3d 9d 69 b9 05 dd  1e 82 d2 36 d1 b5 f7 45  |..=.i......6...E|
000002d0  2a c3 84 38 ce a7 16 1b  7a aa 0f e0 cf 1e ab ff  |*..8....z.......|
000002e0  c0 c2 4c 50 89 64 47 d9  f8 e6 da c2 b9 5c c8 88  |..LP.dG......\..|
000002f0  f4 40 ee 94 6c 3c 4a 32  11 a9 e8 70 82 70 a3 90  |.@..l<J2...p.p..|
00000300  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  01 00 01 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000410  61 5a 97 b1 93 56 bb f3  0b 8f 8c 6f cb cc a2 51  |aZ...V.....o...Q|
00000420  1c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000600  f0 48 9a 4e a5 b2 0d bf  8f 8f bc a6 cb e8 66 ee  |.H.N..........f.|
00000610  8b 04 d5 9b f9 0e 1b ce  e2 b9 7d 83 cd 41 4b cc  |..........}..AK.|
00000620  55 d8 05 32 04 65 36 ac  df 3a b0 fe cf 3d 52 0d  |U..2.e6..:...=R.|
00000630  d8 f0 d2 73 16 37 23 a5  1b e9 06 f1 33 a8 0f 46  |...s.7#.....3..F|
00000640  0a c5 7f 29 6e de fb 47  3e 26 03 70 49 3e 3e 38  |...)n..G>&.pI>>8|
00000650  f6 6c de 82 94 eb 84 51  c3 ad 9d 7c bc 34 00 72  |.l.....Q...|.4.r|
00000660  82 d6 86 97 50 a2 ba b1  47 88 f5 9c 0a 45 67 3d  |....P...G....Eg=|
00000670  b9 58 8b 23 26 1d e4 d2  6a ec 84 9d df 68 c7 4d  |.X.#&...j....h.M|
00000680  bf cf 7c 85 27 74 30 69  f4 e1 2b d1 f3 07 b2 02  |..|.'t0i..+.....|
00000690  1e a1 02 18 6a 0c ad 05  ea 89 ba 3c dc b1 e1 19  |....j......<....|
000006a0  8a b7 a8 41 24 76 f6 1b  7e e6 2a 42 17 db c0 16  |...A$v..~.*B....|
000006b0  b6 54 f2 14 a4 38 93 7c  ff 31 cc 01 ac ba b9 3f  |.T...8.|.1.....?|
000006c0  a5 f6 81 7c 15 75 d8 8f  79 74 78 ab 8f 47 98 dd  |...|.u..ytx..G..|
000006d0  55 ae 16 07 d3 ca 8c 14  9b 9b 6c f1 8f 85 0b 1a  |U.........l.....|
000006e0  5b f0 56 55 42 40 1b 66  d3 df 66 25 ea 90 7e 53  |[.VUB@.f..f%..~S|
000006f0  ba 74 91 9a 7f 61 1e ed  eb 35 08 4e d8 fa 44 87  |.t...a...5.N..D.|
00000700  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000800  01 00 00 14 78 3b 00 14  01 40 a0 d2 02 00 40 b9  |....x;...@....@.|
```

магическая константа

ключ

подпись



MaskROM как грузится SPL

Non Secure Boot:

грузит любую прошивку
подходящего формата

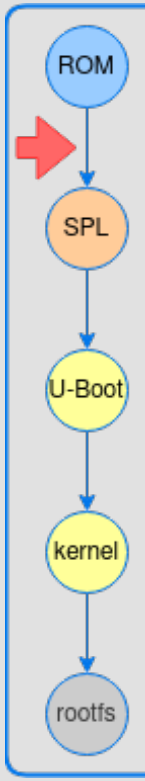
Secure Boot:

не обнаружен

IDA: MaskROM.bin

```
__int64 __fastcall sub_2E18(__int16 a1, _DWORD *loader, int a3)
{
  int v5; // w21
  __int64 result; // x0
  unsigned __int64 v7; // x0
  unsigned __int64 v8; // [xsp+8h] [xbp-18h]

  v5 = *(loader + a3 - ((a3 & 0xFFF) == 1) - 1) | (*(loader + a3 - ((a3 & 0xFFF) == 1) - 2) << 8);
  result = sub_560(loader);
  if ( v5 == result )
  {
    *(&unk_7FFF - 16744431) = 10;
    if ( a1 == 1137 )
    {
      if ( ((aRknsrkss[0] | (aRknsrkss[1] << 8)) | ((aRknsrkss[2] | (aRknsrkss[3] << 8)) << 16)) == *loader
        || ((aRknsrkss[4] | (aRknsrkss[5] << 8)) | ((aRknsrkss[6] | (aRknsrkss[7] << 8)) << 16)) == *loader )
      {
        return sub_1D0(0xFFFFFFFFFF010020i64, loader, 0x200ui64);
      }
      v7 = 0xFFFFFFFFFF011000ui64;
    }
    else
    {
      if ( a1 != 0x472 )
        return result;
      LODWORD(v7) = 0;
    }
    if ( v7 == 0xFF011000 || !v7 )
    {
      v8 = v7;
      sub_445C();
      result = (v8)(0xFFFFFFFFFF010078ui64, 8i64);
    }
    else
    {
      result = 0xFFFFFFFFi64;
    }
  }
  return result;
}
```

Проверка магических констант есть, но двух веток для Secure / NonSecure Boot не видно



MaskROM
а что там с OTP

Табличка из TRM

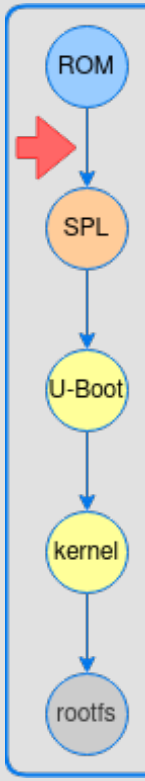
USB2PHY1_GRF	FD5D4000	16KB	CRYPTO_NS	FE370000	32KB
USB2PHY2_GRF	FD5D8000	16KB	TRNG_NS	FE378000	32KB
USB2PHY3_GRF	FD5DC000	16KB	KEYLADDER_S	FE380000	64KB
HDPTXPHY0_GRF	FD5E0000	16KB	CRYPTO_S(Slave)	FE390000	32KB
HDPTXPHY1_GRF	FD5E4000	16KB	TRNG_S	FE398000	32KB
MIPIDCPHY0_GRF	FD5E8000	16KB	OTP_S	FE3A0000	64KB
MIPIDCPHY1_GRF	FD5EC000	16KB	Reserved	FE3B0000	64KB
PMU1_IOC	FD5F0000	16KB	DCF	FE3C0000	64KB
PMU2_IOC	FD5F4000	16KB	TIMER_S_0(6CH)	FE3D0000	64KB
BUS_IOC	FD5F8000	4KB	WDT_S	FE3E0000	64KB
VCCIO1_4_IOC	FD5F9000	4KB	SEC_TRNG_CHK	FE3F0000	64KB

Сегмент sec_otp

```
SEC_OTP:FE3A0004 ?? ?? ?? ??            SEC_OTP_CTL      % 4
SEC_OTP:FE3A0008 ?? ?? ?? ??            SEC_OTP_ENABLE   % 4
SEC_OTP:FE3A000C ?? ?? ?? ??                             % 4
SEC_OTP:FE3A0010 ?? ?? ?? ??            SEC_OTP_INPUT    % 4
SEC_OTP:FE3A0014 ?? ?? ?? ?? ?? ?? ?? ??+                % 0xC
SEC_OTP:FE3A0020 ?? ?? ?? ??            SEC_OTP_OUTPUT   % 4
SEC_OTP:FE3A0024 ?? ?? ?? ?? ?? ?? ?? ??+                % 0x60
SEC_OTP:FE3A0084 ?? ?? ?? ??            SEC_OTP_STATUS   % 4
```

Функция чтения фьюзов

```
__int64 __fastcall SECotpRead(_DWORD *dst, int addr, unsigned int size)
{
  __int64 result; // x0
  int v6; // w8
  int v7; // w22
  __int64 v8; // x8
  __int32 *v9; // x9

  if ( size > 0x10 )
    return 0xFFFFFFFFi64;
  SEC_OTP_CTL = (size << 8) | (addr << 16);
  SEC_OTP_ENABLE = 1;
  v6 = *&SEC_OTP_STATUS;
  if ( (SEC_OTP_STATUS & 2) == 0 )
  {
    v7 = -10001;
    while ( ++v7 )
    {
      sleep(1);
      v6 = *&SEC_OTP_STATUS;
      if ( (SEC_OTP_STATUS & 2) != 0 )
        goto LABEL_7;
    }
    return 0xFFFFFFFFi64;
  }
LABEL_7:
  result = 0i64;
  *&SEC_OTP_STATUS = v6;
  if ( dst && size )
  {
    v8 = size;
    v9 = &SEC_OTP_OUTPUT;
    do
    {
      *dst++ = *(v9 & 0xFFFFFFFC);
      --v8;
      ++v9;
    }
    while ( v8 );
    result = 0i64;
  }
  return result;
}
```



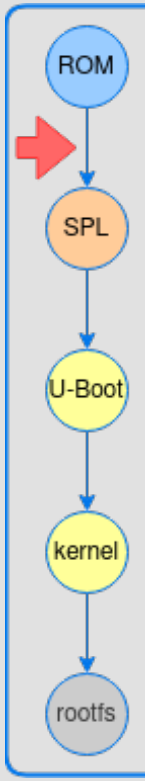
MaskROM
а что там с OTP

Чтение фьюзов в MaskROM

```
  if ( (SecureOTPRead(&v5, 1, 1u) & 0x80000000) != 0 || (SecureOTPRead(&v4, 0x17, 1u) & 0x80000000) != 0 )
    goto LABEL_18;
  if ( v5 != unk_FD586214 || v4 != unk_FD58621C )
  {
    v4 = 0;
    v5 = 0;
    goto LABEL_18;
  }
  if ( sub_4424(v5, 0x5A) > 2 )
  {
    v5 = 0;
    if ( (SecureOTPWriteByte(1, 0x5A) & 0x80000000) != 0 )
      goto LABEL_18;
  }
  else
  {
    v5 = 1;
  }
  if ( sub_4424(v4, 0xA5) > 2 )
  {
    v4 = 0;
    if ( (SecureOTPWriteByte(0x17, 0xA5) & 0x80000000) != 0 )
      goto LABEL_18;
    v0 = v4 == 0;
  }
  else
  {
    v0 = 0;
    v4 = 1;
  }
  if ( !v5 || v0 || (v3 = 0, v1 = SecureOTPRead(&v3, 8, 1u), (v1 & 0x80000000) != 0) || sub_4424(v3, 255) <= 4 )
LABEL_18:
    sub_10F4();
```

Заглушка с вечным циклом :(

Хочется увидеть либо 2 ветки загрузки (с проверкой подписи и без), либо чтение ключа или хэша (много байт подряд)



MaskROM их 2

Чтоб увидеть SecureBoot нужен секретный MaskROM?

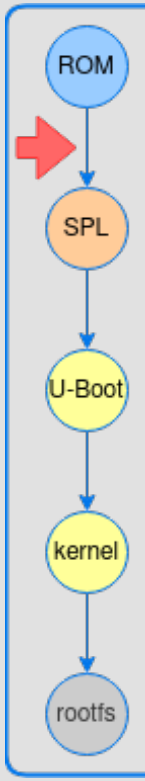
Чтоб увидеть секретный MaskROM нужен SecureBoot?

1.2 System Boot

RK3588 provides system boot from off-chip devices such as SDMMC card, eMMC memory, serial Nand or Nor flash. When boot code is not ready in these devices, also provide system code download into them by USB OTG interface. All of the boot code will be stored in internal BootRom. The following is the whole boot procedure for boot code, which will be stored in BootRom in advance.

The following features are supported.

- Support system boot from the following device:
  - Serial Nor Flash, 1bit or 4bits data width(device layout in FSPI IO)
  - Serial Nand Flash, 1bit data width(device layout in FSPI IO)
  - eMMC Interface, 8bits data width
  - SDMMC Card, 4bits data width
  - RK3588 has two bootwom:
    - Normal Bootrom in non-secure world, which CPU_S and CPU_NS both can access
    - Secure Bootrom in secure world, only CPU_S can access
- Support system code download by USB OTG



RAM Boot
SecureInit и SecureWriteOTP

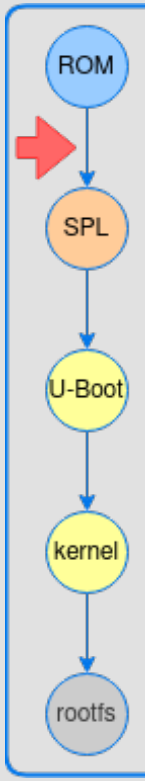
```
__int64 __fastcall SecureInit(unsigned __int8 *a1)
{
  printf("Head flags=0x%x\n", (a1[0xC] | (a1[0xD] << 8)) | ((a1[0xE] | (a1[(
  if ( !SecureCheckHeader(a1) )
  {
    if ( (a1[0xD] & 0x20) == 0 )
    {
      dword_3332DD0[0] = 1;
      unk_3122140 = 1;
LABEL_13:
      sub_3001250(0x3000800i64, a1, 0x368ui64);
      goto LABEL_14;
    }
    if ( SecureWriteOTP(a1) )
    {
      ret = 0xFFFFFFFA;
      goto LABEL_15;
    }
    dword_3332DD0[0] = 1;
LABEL_10:
    if ( sub_3006AF0(a1) )
    {
      ret = 0xFFFFFFFB;
      goto LABEL_15;
    }
    unk_3122140 = 1;
    goto LABEL_13;
  }
  ret = 0xFFFFFFFC;
LABEL_15:
  printf("SecureInit ret = %d, SecureMode = %d\n", ret, dword_3332DD0[0]);
```

Проверка корректности подписи

```
__int64 __fastcall SecureWriteOTP(__int64 data)
{
  printf("SecureWriteOTP\n");
  err = RkOTPRead(v5, 0x270u, 8u);
  if ( (err & 0x80000000) != 0 )
    goto ERROR;
  if ( v5[0] )
    sub_300A644("otp no empty!:", v5, 4u, 8u);
  err = sha256(data + 0x200, 0x230u, key_hash);
  if ( err )
    goto ERROR;
  err = RkOTPWrite(key_hash, 0x270u, 8u);
  if ( err )
    goto ERROR;
  err = RkOTPRead(v5, 0x270u, 8u);
  if ( err )
    goto ERROR;
  if ( sub_3001178(v5, key_hash, 0x20ui64) )
  {
    sub_300A644("key_hash:", key_hash, 4u, 8u);
    sub_300A644("otp_data:", v5, 4u, 8u);
    err = 0xFFFFFFFE;
ERROR:
    printf("otp write error: %d!!!\n", err);
    return err;
  }
  printf("enable secure flag:0x%x\n", v3);
  err = RkOTPWrite(&a1, 8u, 1u);
  if ( err )
    goto ERROR;
  v7 = 0;
  err = RkOTPRead(&v7, 8u, 1u);
  if ( err )
    goto ERROR;
  if ( v7 != a1 )
  {
    printf("otp flag:%x, error:\n", v7);
    err = 0xFFFFFFFF;
    goto ERROR;
  }
  printf("otp write key success!!!\n");
```

Запись хэша ключа

Запись флагов, включающих SecureBoot



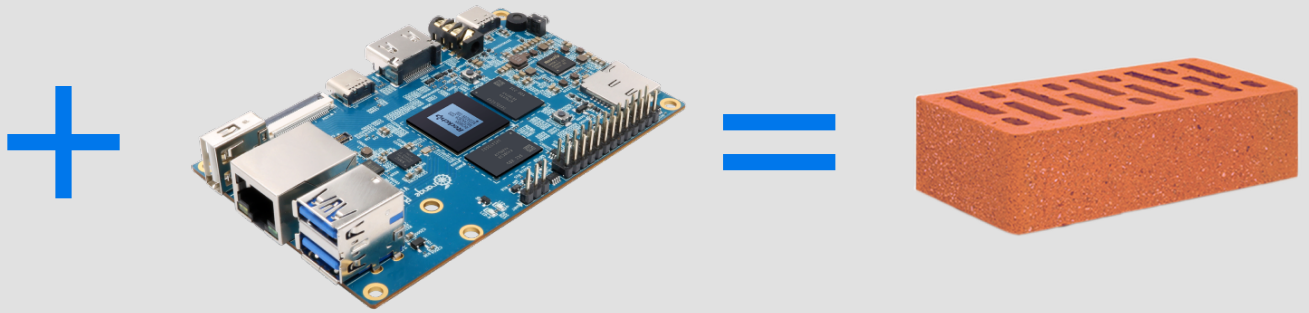
Write OTP
как я потерял первую апельсинку

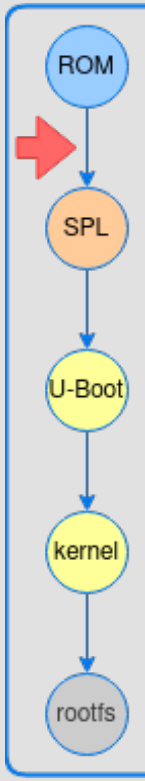
SecureInit.s

```
FILE=0x6ed8
BASE=0x30076d8
SecureWriteOTP=0x3006be4
SecureCheckHeader=0x3006934
printf=0x300A1F4
start:
LDR          X0,=0x3008bd0
BL          (SecureCheckHeader+start-BASE)
LDR          X0,=0x3008bd0
BL          (SecureWriteOTP+start-BASE)
LDR          X0,=0x3008bd0
BL          (printf+start-BASE)
loop:
B loop
```

```
out
Boot1 Release Time: Feb 24 2022 10:23:56, version: 1.05 USB BOOT
ChipType = 0x32, 466
SecureMode = 1
atags_set_bootdev: ret:(0)
UsbBoot ...1049
powerOn 3363
SecureCheckHeader -1
RKSS
```

Игнорирование ошибок
к добру не приводит.
Особенно с фьюзами.





Write OTP some_flag

```
__int64 __fastcall SecureCheckHeader(unsigned __int8 *data)
{
  __int64 v1; // x1
  unsigned __int8 *key; // x2
  unsigned int ret_code; // w20

  if ( ((*data | (data[1] << 8)) | ((data[2] | (data[3] << 8)) << 0x10)) != 'SSKR' )// RKSS
    goto RET_m1;
  key = data + 0x200;
  if ( !some_flag )
  {
LABEL_6:
    ret_code = sub_3006B90(data, v1, key, (data + 0x600));
    goto RET;
  }
  if ( some_buff[0] != 0x4B504B52 )             // RKPK
  {
RET_m1:
    ret_code = -1u;
    goto RET;
  }
  ret_code = sub_3006B90(some_buff, v1, key, &some_buff[0x180]);
  if ( !ret_code )
  {
    key = &some_buff[0x10];
    goto LABEL_6;
  }
RET:
  printf("SecureCheckHeader %d\n", ret_code);
  return ret_code;
}
```

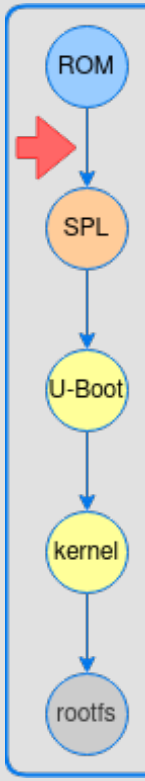
Опускаем some_flag

Проверяем код возврата ;)

```
UsbBoot ...1050
powerOn 3301
SecureCheckHeader 0
Succes!
```

SecureInit_Fixed.s

```
FILE=0x6ed8
BASE=0x30076d8
SecureWriteOTP=0x3006be4
SecureCheckHeader=0x3006934
printf=0x300A1F4
start:
LDR         X0,=0x3120030
MOV         W8, 0
STR         W8, [X0,4]
MOV         W8, 0x100
STR         W8, [X0]
LDR         X0,=0x3008bd0
BL          (SecureCheckHeader+start-BASE)
CBNZ        W0, fail
LDR         X0,=0x3008bd0
BL          (SecureWriteOTP+start-BASE)
CBNZ        W0, fail
ADR         X0,.ok_msg
BL          (printf+start-BASE)
B loop
fail:
ADR         X0,.fail_msg
BL          (printf+start-BASE)
loop:
B loop

.ok_msg: .ascii "Succes!\n\x00"
.fail_msg: .ascii "Fail :(\n\x00"
```



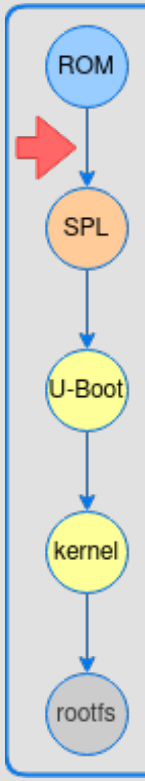
Signed SPL
rk_sign_tool - работает

```
DDR Version V1.08 20220617
LPDDR4X, 2112MHz
channel[0] BW=16 Col=10 Bk=8 CS0 Row=16 CS=1 Die BW=16 Size=1024MB
channel[1] BW=16 Col=10 Bk=8 CS0 Row=16 CS=1 Die BW=16 Size=1024MB
channel[2] BW=16 Col=10 Bk=8 CS0 Row=16 CS=1 Die BW=16 Size=1024MB
channel[3] BW=16 Col=10 Bk=8 CS0 Row=16 CS=1 Die BW=16 Size=1024MB
Manufacturer ID:0x1 Samsung
CH0 RX Vref:34.7%, TX Vref:18.8%,0.0%
CH1 RX Vref:34.7%, TX Vref:18.8%,0.0%
CH2 RX Vref:30.7%, TX Vref:21.8%,0.0%
CH3 RX Vref:31.7%, TX Vref:20.8%,0.0%
change to F1: 528MHz
change to F2: 1068MHz
change to F3: 1560MHz
change to F0: 2112MHz
out
U-Boot SPL board init
U-Boot SPL 2017.09-orangepi (Jun 01 2023 - 21:22:25)
unknown raw ID 0 0 0
unrecognized JEDEC id bytes: 00, 00, 00
Trying to boot from MMC1
Trying fit image at 0x4000 sector
## Verified-boot: 1
Verified-boot requires CONFIG_SPL_FIT_SIGNATURE enabled
### ERROR ### Please RESET the board ###
# Reset the board to bootrom #
```

```
/* if board sigs verify required, check self */
if (fit_board_verify_required_sigs() &&
    !IS_ENABLED(CONFIG_SPL_FIT_SIGNATURE)) {
        printf("Verified-boot requires CONFIG_SPL_FIT_SIGNATURE enabled\n");
        hang();
}
```

Эту строчку выводит SPL а не MaskROM,
значит SPL подписан правильно

SPL хочет проверить
подпись, но не умеет.



Signed U-boot config и сборка

Очередная ошибка, на этот раз SPL не видит ключ

```
Trying fit image at 0x4000 sector
## Verified-boot: 1
No RSA key found
fit verify configure failed, ret=-22
Trying fit image at 0x5000 sector
Not fit magic
SPL: failed to boot from all boot devices
### ERROR ### Please RESET the board ###
# Reset the board to bootrom #
```

Кладём ключ вручную

```
IndigoLemming::nik ~/rk/rkbin $ export SRC=u-boot-opl-patch && ssh user@10.4.1.110 "cd /home/user/$SRC && make clean && make rk358
8_defconfig && CROSS_COMPILE=aarch64-linux-gnu- make -j32 BL31=../rkbin/bin/rk35/rk3588_bl31_v1.40.elf spl/u-boot-spl.bin u-boot.dt
b u-boot.itb && ./tools/mkimage -f u-boot.its -K spl/u-boot-spl.dtb -k keys/ -E -r u-boot.itb && fdtput -tbx spl/u-boot-spl.dtb /s
ignature/key-dev rsa,exponent-BN 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 01 00 01  && fdtput -tbx spl/u-boot-spl.dtb /signature/key-dev rsa,np 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

U-Boot подписан

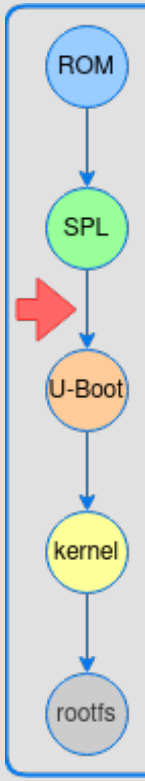
```
Trying fit image at 0x4000 sector
## Verified-boot: 1
sha256,rsa2048:dev## Verified-boot: 1
+
## Checking atf-1 0x00040000 ... sha256(b40ce543bb...) + OK
## Checking uboot 0x00200000 ... sha256(a330667f11...) + OK
## Checking fdt 0x0033c430 ... sha256(1b9011cdcb...) + OK
## Checking atf-2 0xff100000 ... sha256(af08976afd...) + OK
## Checking atf-3 0x000f0000 ... sha256(f202de33d9...) + OK
Jumping to U-Boot(0x00200000) via ARM Trusted Firmware(0x00040000)
Total: 212.109 ms
```

А ключ есть, посто формат не тот

```
    prop->public_exponent = fdt_getprop(blob, node, "rsa,exponent", &length);
    if (!prop->public_exponent || length < sizeof(uint64_t))
        prop->public_exponent = NULL;

    prop->exp_len = sizeof(uint64_t);
    prop->modulus = fdt_getprop(blob, node, "rsa,modulus", NULL);
    prop->public_exponent_BN = fdt_getprop(blob, node, "rsa,exponent-BN", NULL);
    prop->rr = fdt_getprop(blob, node, "rsa,r-squared", NULL);
#ifdef CONFIG_ROCKCHIP_CRYPTO_V1
    hash_node = fdt_subnode_offset(blob, node, "hash@c");
#else
    hash_node = fdt_subnode_offset(blob, node, "hash@np");
#endif
```

Но теперь не может найти ядро

```
## Booting FIT Image FIT: No fit blob
FIT: No FIT image
No CLI available
### ERROR ### Please RESET the board ###
```



Signed Kernel FIT как его собрать

Flattened Image Tree — формат загрузочного образа похожий на flattened device tree

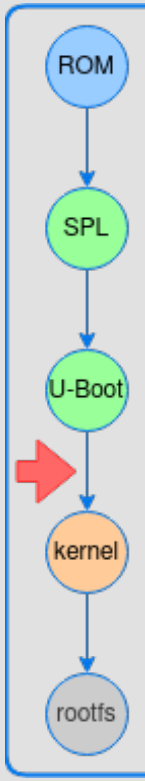
Параметры образа: содержимое, подпись

Параметры бинарей: путь к файлу, адрес загрузки, хэш

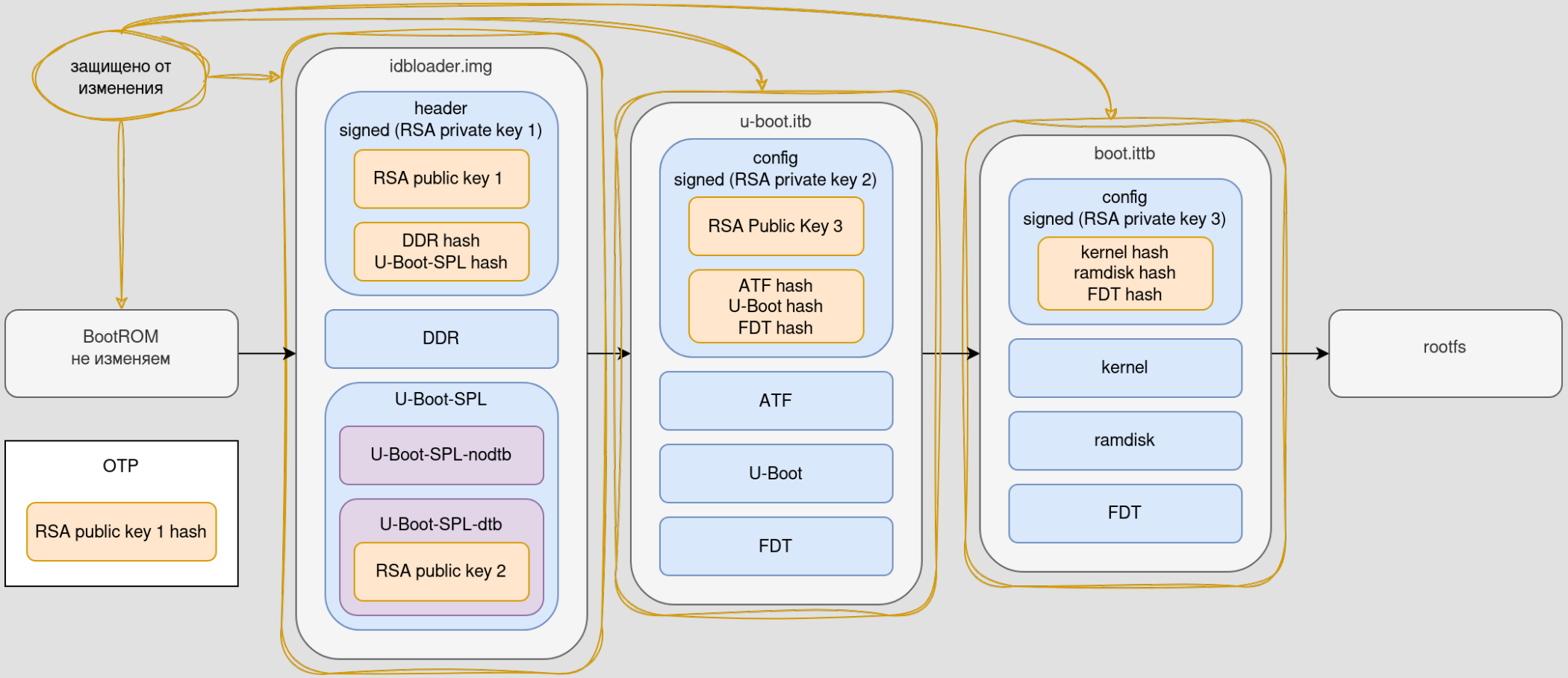
Конфиг для сборки

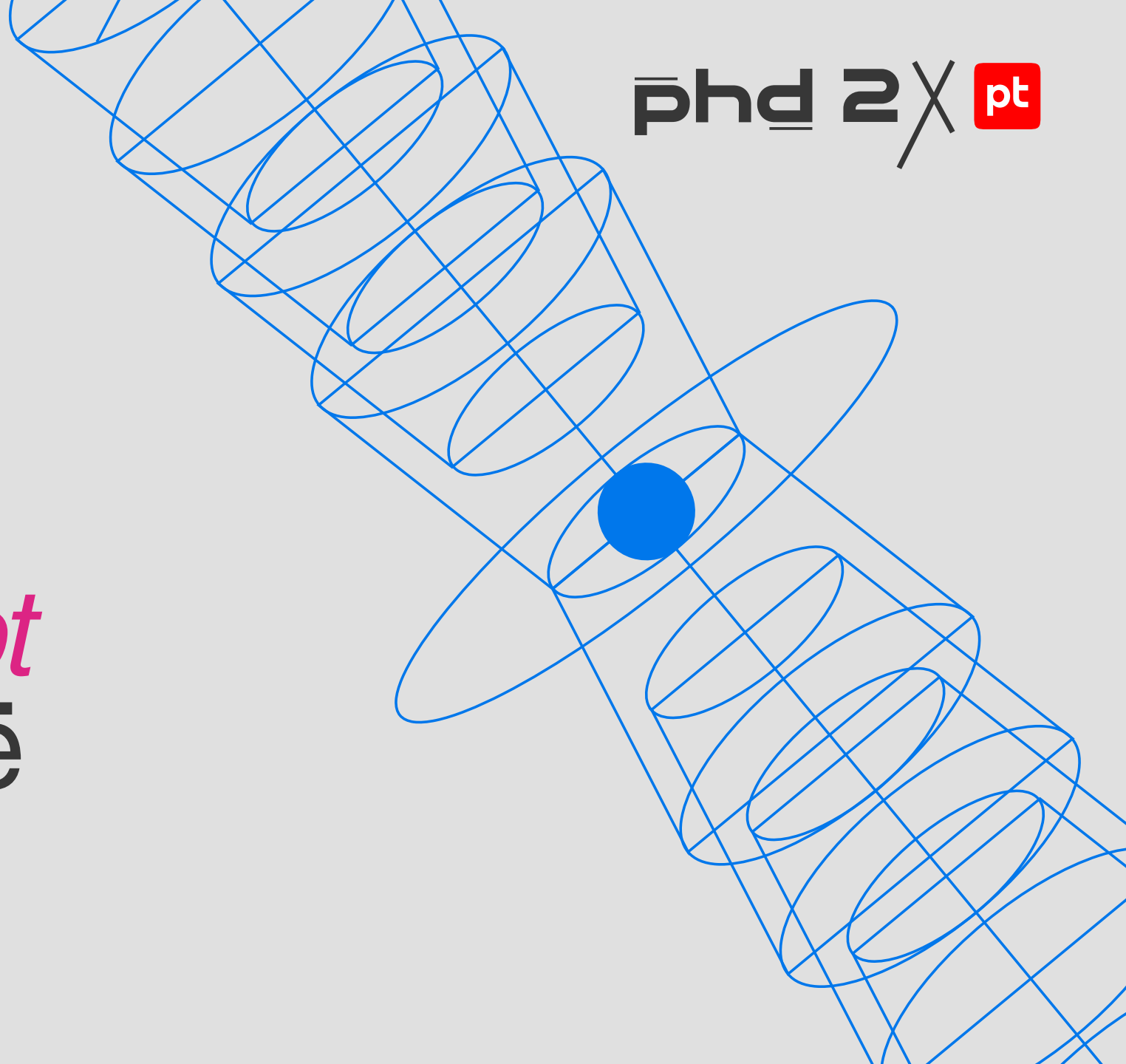
```
configurations {
    default = "conf";

    conf {
        rollback-index = <0x00>;
        fdt = "fdt";
        kernel = "kernel";
        ramdisk = "ramdisk";

        signature {
            algo = "sha256,rsa2048";
            padding = "pss";
            key-name-hint = "dev";
            sign-images = "fdt", "kernel","ramdisk";
        };
    };
};
```

```
images {
    kernel {
        data = /incbin/("./Image");
        type = "kernel";
        arch = "arm64";
        os = "linux";
        compression = "NONE";
        load = <0x00400000>;
        entry = <0x00400000>;

        hash {
            algo = "sha256";
        };
    };
    fdt {
        data = /incbin/("./rk3588s-orangepi-5.dtb");
        type = "flat_dt";
```



Secure BOOT
как выглядит цепочка





EncryptedBoot

спрятать своё

02

Uboot XOR
начнём с простого

arch/arm/mach-rockchip/fit_misc.c — post process

```
void board_fit_image_post_process(void *fit, int node, ulong *load_addr,
                                  ulong **src_addr, size_t *src_len, void *spec)
{
        fit_decrypt_image(fit, node, load_addr, src_addr, src_len, spec);
#if CONFIG_IS_ENABLED(MISC_DECOMPRESS) || CONFIG_IS_ENABLED(GZIP)
        fit_decomp_image(fit, node, load_addr, src_addr, src_len, spec);
#endif
#if CONFIG_IS_ENABLED(USING_KERNEL_DTB)
        /* Avoid overriding processed(overlay, hw-dtb, ...) kernel dtb */
        if (fit_image_check_type(fit, node, IH_TYPE_FLATDT)) {
                if ((gd->flags & GD_FLG_KDTB_READY) && !gd->fdt_blob_kern) {
                        *src_addr = (void *)gd->fdt_blob;
                        *src_len = (size_t)fdt_totalsize(gd->fdt_blob);
                } else {
                        printf("   Using fdt from load-in fdt\n");
                }
        }
#endif
}
```

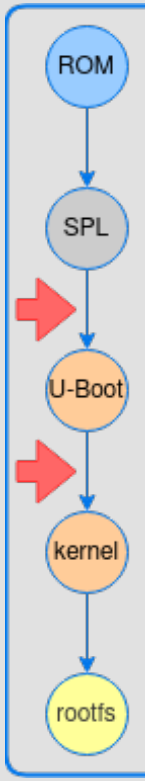
decompress — хорошая основа для decrypt

Поддержка разных методов шифрования
(как в decomp — разных алгоритмов сжатия)
пригодится когда будем добавлять AES

arch/arm/mach-rockchip/fit_misc.c — decrypt

```
#define FIT_ENC_KEY_SIZE 16
int fit_decrypt_image(void *fit, int node, ulong *load_addr,
                      ulong **src_addr, size_t *src_len, void *spec)
{
        uint8_t img_enc;
        u64 len = *src_len;
        fit_image_get_enc(fit, node, &img_enc);
        printf("Encryption type=%d\n",img_enc);
        printf("src_addr=%p, src_addr=%lx, len=%lld\n",*src_addr,**src_addr,len);
        if(img_enc==IH_ENC_NONE)
                return 0;
        if (img_enc!=IH_ENC_XOR ){
                printf("Image decryption faile: unknown encryption\n");
                return -1;
        }

        uint8_t key[FIT_ENC_KEY_SIZE] = { 0,1,2,3,4,5,6,7,8,9,0xA,0xB,0xC,0xD,0xE,0xF};
        uint8_t * xor_src=(uint8_t*) (*src_addr);
        for(u64 i=0; i<len;++i){
                xor_src[i]=xor_src[i]^key[i%FIT_ENC_KEY_SIZE];
        }
        *src_addr = (ulong *)*load_addr;
        *src_len = len;
        return 0;
}
```

XOR в 5 строчек. Ну где здесь можно ошибиться?



Uboot XOR OTP
спрячем ключ

Да-да, это сорцы, верим

drivers/misc/rk3588-secure-otp.S

```
.LVL27:
#APP
// 68 "drivers/misc/rk3588-secure-otp.c" 1
        dmb sy
// 0 "" 2
#NO_APP
        ldr   x1, [x20]
.LVL28:
```

Ограничения для слабаков

Коварный патч

```
.LVL50:
        ccmp        w0, 7, 0, ne
        bls   .L26
        .loc 1 102 0 is_stmt 0 discriminator 1
        sub  w0, w20, #2496
        cmp w0, 63
        bls   .L26
        .loc 1 102 0 discriminator 2
        sub  w0, w20, #416
        mov x19, 0
        cmp w0, 95
        bls   .L23
        bl    .L23
        .loc 1 106 0 is_stmt 1
        mov w21, -1
        .loc 1 105 0
        mov w1, w20
        adrp        x0, .LC3
        add  x0, x0, :lo12:.LC3
        bl    printf
```

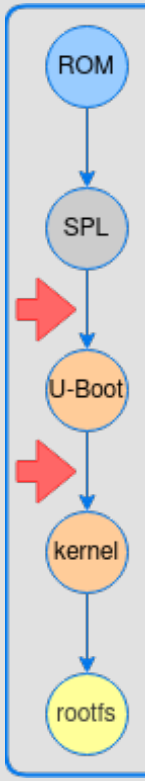
IDA: rk3588-secure-otp.o

```
                        ; rockchip_otp_platdata *
CCMP            W0, #7, #0, NE
B.LS            loc_2B0
SUB             W0, W20, #0x9C0
CMP             W0, #0x3F ; '?'
B.LS            loc_2B0
SUB             W0, W20, #0x1A0
MOV             X19, #0
CMP             W0, #0x5F ; '_'
B.LS            loc_2B4
MOV             W21, #0xFFFFFFFF
MOV             W1, W20
ADRL            X0, aPleaseInputCor ; "Please input correct addr,
BL              printf
```

IDA лучше “сорцов”

```
v7 = dev_get_platdata(dev);
if ( offset == 0x20 )
  v8 = 0;
else
  v8 = (offset - 0x150) > 7;
if ( v8 && (offset - 0x9C0) > 0x3F )
{
  v9 = 0LL;
  if ( (offset - 0x1A0) <= 0x5F )
    goto LABEL_13;
  v10 = 0xFFFFFFFF;
  printf("Please input correct addr, offset is %x\n", offset);
}
else
{
  v9 = 0LL;
LABEL_13:
  while ( size > v9 )
  {
    v12 = offset + v9;
    v13 = buf + v9++;
    v10 = rk3588_secure_otp_read_byte(v7, v13, v12);
    if ( v10 )
    {
      printf("Read secure otp fail.");
      return v10;
    }
  }
```

Особенно если жмакнуть F5



Uboot XOR FIT

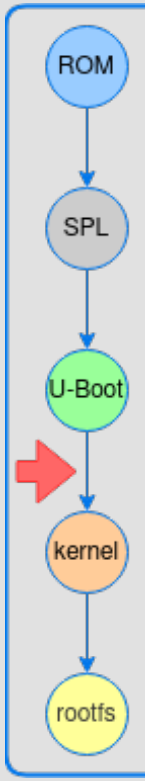
ну нет так нет

arch/arm/mach-rockchip/fit_misc.c : fit_decrypt_image

```
if(img_enc==IH_ENC_OTP){
        struct udevice* dev = misc_otp_get_device(OTP_S);
        misc_otp_read(dev, KEY_OTP_ADDR, key, FIT_ENC_KEY_SIZE);
}
if(img_enc==IH_ENC_FDT){
        const void *blob = gd_fdt_blob();
        int key_node=fdt_subnode_offset(blob, 0, "encryption");
        key  =(uint8_t *)  fdt_getprop(blob, key_node, "key", NULL);
}
```

Читаем ключ из OTP

Или берём из FDT



Uboot AES

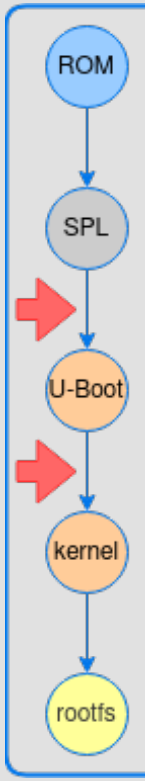
xor удобен для тестов, но небезопасен

```
int rk_crypto_ae(struct udevice *dev, u32 algo, u32 mode,
                 const u8 *key, u32 key_len, const u8 *nonce, u32 nonce_len,
                 const u8 *in, u32 len, const u8 *aad, u32 aad_len,
                 u8 *out, u8 *tag)
{
        u32 rk_mode = RK_GET_RK_MODE(mode);
        int ret;

        if (!IS_AE_MODE(rk_mode))
                return -EINVAL;

        if (algo != CRYPTO_AES && algo != CRYPTO_SM4)
                return -EINVAL;

        /* RV1126/RV1109 do not support aes-192 */
#if defined(CONFIG_ROCKCHIP_RV1126)
        if (algo == CRYPTO_AES && key_len == AES_KEYSIZE_192)
                return -EINVAL;
#endif

        ret = hw_cipher_init(g_key_chn, key, NULL, key_len, nonce, nonce_len,
                             algo, mode, false);
        if (ret)
                return ret;

        return hw_cipher_crypt(in, out, len, aad, aad_len,
                               tag, AES_BLOCK_SIZE, mode);
}
```

true = encrypt
false = decrypt

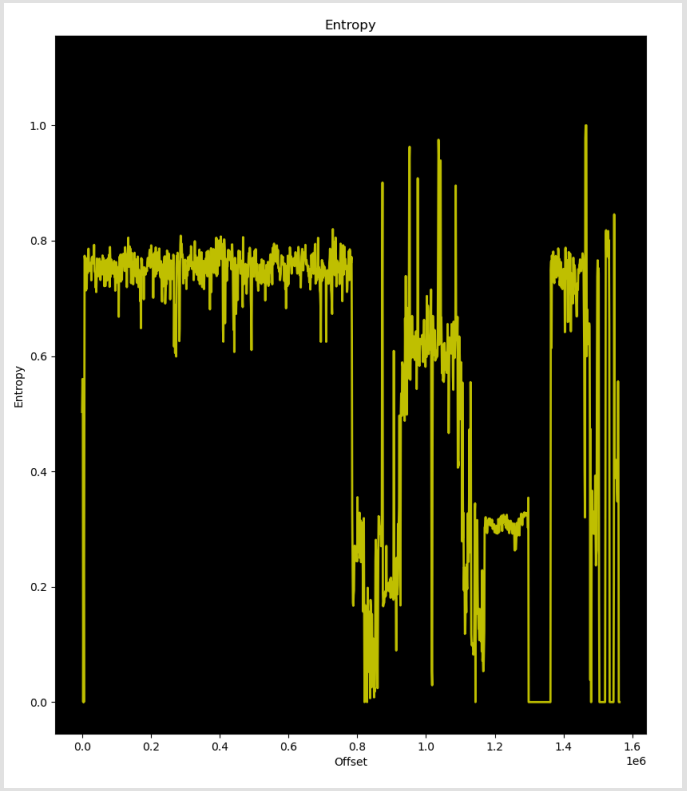
```
    memcpy(nonce,src,FIT_ENC_KEY_SIZE);
    memcpy(tag,src+FIT_ENC_KEY_SIZE,FIT_ENC_KEY_SIZE);

    dev = crypto_get_device(CRYPTO_AES);
    ctx.algo    = CRYPTO_AES;
    ctx.mode    = RK_MODE_GCM;
    ctx.key     = key;
    ctx.key_len = FIT_ENC_KEY_SIZE;
    ctx.iv      = &nonce[0];
    ctx.iv_len  = FIT_ENC_KEY_SIZE;
    len-=2*FIT_ENC_KEY_SIZE;

    int ret = crypto_ae(dev,&ctx,src+2*FIT_ENC_KEY_SIZE,len, aad, 0x10, src, dig);
    uint64_t * a = (uint64_t *) tag;
    uint64_t * b = (uint64_t *) dig;
    if (ret || ((a[0]^b[0])|(a[1]^b[1]))){
        memset(src,0,len+2*FIT_ENC_KEY_SIZE);
        return -2;
    }
    *src_len=len;
    return 0;
```

Проверяем что тэг совпал

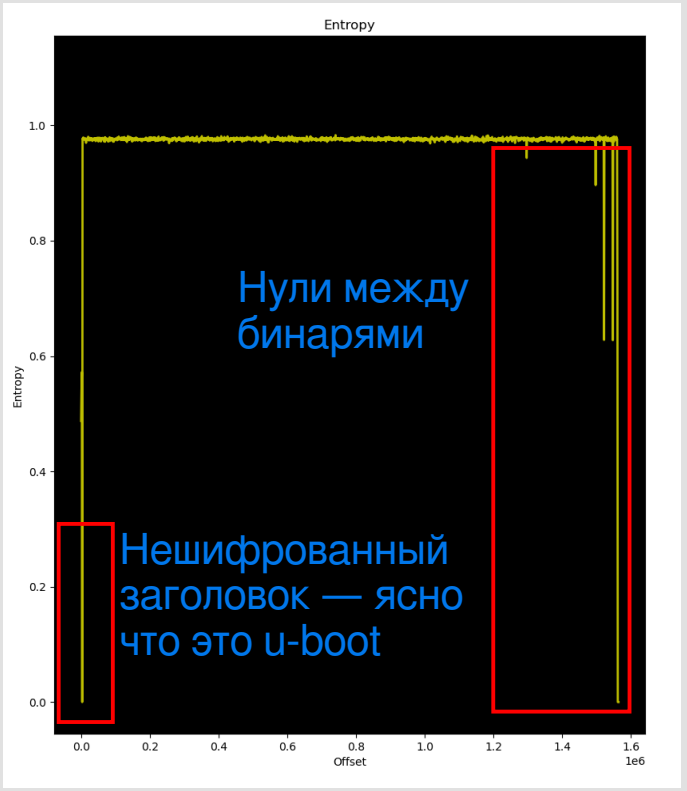


Encrypt more
шифруем всё

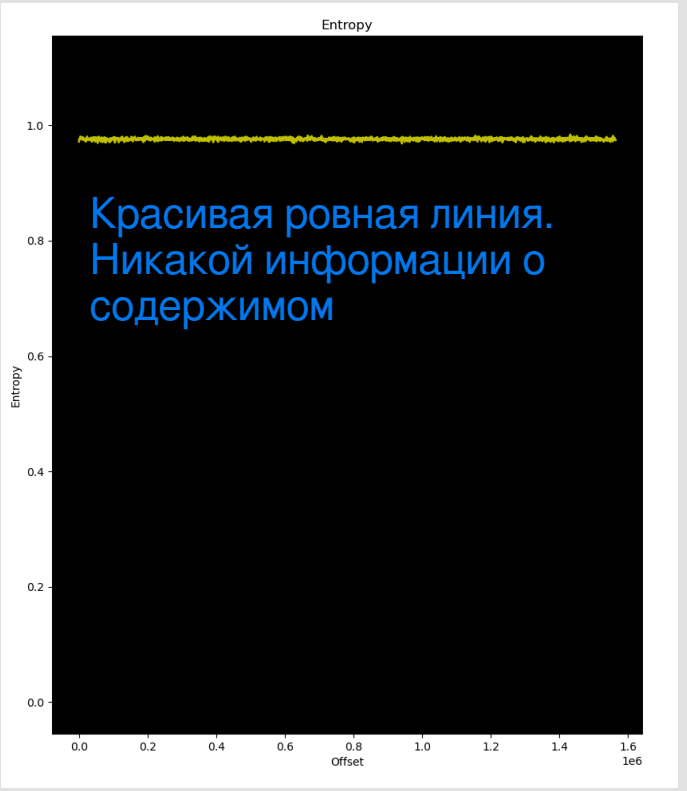
No encryption

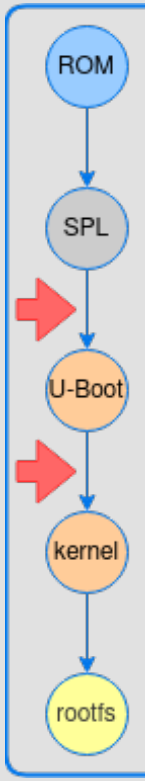


Encrypted binaries



Full image encrypton





Encrypt more
шифруем всё

arch/arm/mach-rockchip/fit.c

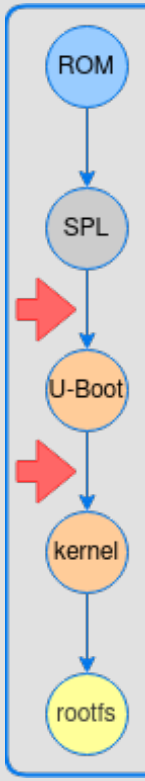
```
int decrypt(void* src, void* dst, size_t size, void* aad, void* tag_out){
	printf("decrypting %lx bytes...\n",size);
	uint8_t* key;
	const void *blob = gd_fdt_blob();
	int key_node=fdt_subnode_offset(blob, 0, "encryption");
	key  =(uint8_t *)  fdt_getprop(blob, key_node, "key", NULL);

	uint8_t nonce[ENC_KEY_SIZE] = {0};
	uint8_t tag[ENC_KEY_SIZE]={0};
	uint8_t dig[ENC_KEY_SIZE]={0};
	struct udevice *dev;
	cipher_context ctx;

	memcpy(nonce,src,ENC_KEY_SIZE);
	memcpy(tag,src+ENC_KEY_SIZE,ENC_KEY_SIZE);

	dev = crypto_get_device(CRYPTO_AES);
	ctx.algo    = CRYPTO_AES;
	ctx.mode    = RK_MODE_GCM;
	ctx.key     = key;
	ctx.key_len = ENC_KEY_SIZE;
	ctx.iv      = &nonce[0];
	ctx.iv_len  = ENC_KEY_SIZE;

	int ret = crypto_ae(dev,&ctx,src+2*ENC_KEY_SIZE,size, aad, ENC_KEY_SIZE, dst, dig);
	uint64_t * a = (uint64_t *) tag;
	uint64_t * b = (uint64_t *) dig;
	if (ret || ((a[0]^b[0])|(a[1]^b[1]))){
		memset(src,0,size+2*ENC_KEY_SIZE);
		memset(dst,0,size);
		return -2;
	}
	memset(dst+size,0,2*ENC_KEY_SIZE);
	if (tag_out)
		memcpy(tag_out,dig,ENC_KEY_SIZE);
	printf("decrypt OK\n");
	return 0;
}
```

arch/arm/mach-rockchip/fit.c : load bootables

```
if (blk_dread(dev_desc, part.start, blk_num, fit) != blk_num) {
	FIT_I("Failed to load bootable images\n");
	return NULL;
}
size_t head_size=sizeof(struct fdt_header); //0x28
uint8_t aad[ENC_KEY_SIZE]={0};
if (decrypt(fit,fit,head_size,aad,aad))
	return NULL;
size_t head2_size=fdt_totalsize(fit);	//0x800
if (decrypt(fit+head_size+2*ENC_KEY_SIZE,fit+head_size,
		head2_size-head_size,aad,aad))
	return NULL;
if (decrypt(fit+head2_size+4*ENC_KEY_SIZE,fit+head2_size,
		*size-head2_size,aad,aad))
	return NULL;
printf("\n%s OK\n",__func__);
return fit;
```

чтение внешней памяти

“Зацепление” частей — мелочь, но красиво



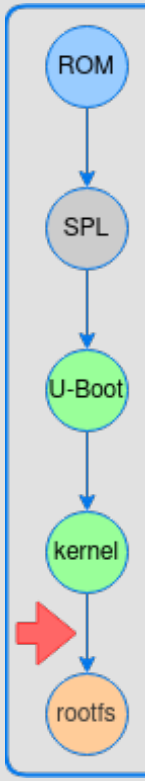
LUKS
шифруем диск без пароля

Распаковываем initrams
Добавляем root в crypttab
Добавляем ключ
Упаковываем initramfs

```
mkdir build/rd
cd build/rd
cpio -i -I ../kernel/rd
mkdir cryptroot
echo "root UUID=${1} /cryptroot/keyfiles/root.key initramfs,luks" > cryptroot/crypttab
mkdir cryptroot/keyfiles
cp ../../keys/luks/root.key cryptroot/keyfiles/
find . | cpio -o -H newc > ../kernel/ramdisk0
```

Создаём luks раздел
Записываем rootfs
Исправляем /boot

```
mount boot.img /mnt/p1
cryptsetup --key-file=keys/luks/root.key luksFormat /dev/mmcblk0p2
cryptsetup --key-file=keys/luks/root.key open /dev/mmcblk0p2 cryptosd
dd if=root.img of=/dev/mapper/cryptosd bs=64M status=progress
resize2fs /dev/mapper/cryptosd
mount /dev/mapper/cryptosd /mnt/p2
cp -r /mnt/p1/* /mnt/p2/boot/
vim /mnt/p2/etc/fstab # delete /boot entry
umount /mnt/p1
umount /mnt/p2
cryptsetup close cryptosd
```



SecureMaskROM
а вот и загрузчик

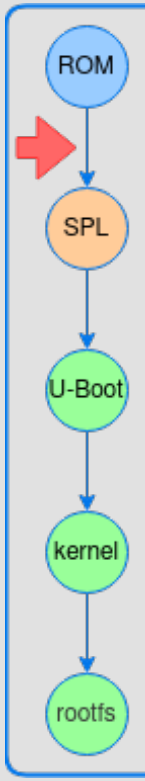
```
__int64 __fastcall load_binary__(header_struct *header, char *data, header_2_struct *fw_desc)
{
  unsigned int size; // w21
  unsigned int v6; // w0
  unsigned int v7; // w22
  int v9[4]; // [xsp+0h] [xbp-60h] BYREF
  char v10[32]; // [xsp+10h] [xbp-50h] BYREF

  size = (LOBYTE(fw_desc->data_size) | (HIBYTE(fw_desc->data_size) << 8)) << 9;
  if ( (header->boot_flag & 0x1000) != 0 )
  {
    v9[0] = (header->field_18[0] | (header->field_18[1] << 8)) | ((header->field_18[2] | (header->field_18[3] << 8)) << 0x10);
    v9[1] = (header->field_18[4] | (header->field_18[5] << 8)) | ((header->field_18[6] | (header->field_18[7] << 8)) << 0x10);
    v9[2] = 0;
    v9[3] = bswap32((fw_desc->field_C[0] | (fw_desc->field_C[1] << 8)) | ((fw_desc->field_C[2] | (fw_desc->field_C[3] << 8)) << 0x10));
    if ( !unk_FF00002C )
    {
      v7 = sub_FFFF0F98(0x20, 4u, 0);
      if ( v7 )
        return v7;
      unk_FF00002C = 1;
    }
    v6 = sub_FFFF5C5C(0i64, v9, data, size, v10);
  }
  else
  {
    v6 = sub_FFFF60B0(data, size, v10);
  }
  v7 = v6;
  if ( !v6 )
  {
    v7 = sub_FFFF628C(fw_desc->field_18, v10, 0x20);
    sub_FFFF37B0(*word_FF000270 + 1, v7);
  }
  return v7;
}
```

Флаг определяет в какую ветку мы попадём

Какая-то подготовка

Что-то с hw_crypto

Что-то с hw_crypto, но поменьше



SecureMaskROM

интересно, зачем тут AES?

boot_flag & 0x1000 == 0

```
__int64 __fastcall hash(char *data, unsigned int size,
{
  __int64 v4; // x11
  __int64 v5; // x12
  __int64 result; // x0
  __int64 v7; // [xsp+0h] [xbp-30h] BYREF
  __int64 v8; // [xsp+8h] [xbp-28h]
  __int64 v9; // [xsp+10h] [xbp-20h]
  __int64 v10; // [xsp+18h] [xbp-18h]

  v10 = 0i64;
  v7 = 0i64;
  v8 = 0i64;
  v4 = 0i64;
  v9 = 0x20i64;
  do
  {
    v5 = v4 + 0xFE390008i64;
    v4 += 4i64;
    *(v5 + 0x398) = 0;
  }
  while ( v4 != 0x40 );
  CRYPTO_HASH_CTL = 0xFFFF0024;
  CRYPTO_FIFO_CTL = 0x30003;
  SCRYPTO_DMA_INT_EN = 0;
  LODWORD(v8) = 0x1A1A1A1A;
  result = cryHash_sub_FFFF6148(&v7, data, size, 1);
  if ( !result )
    result = cryHash_sub_FFFF5EA4(&v7, dst, 0i64);
  return result;
}
```

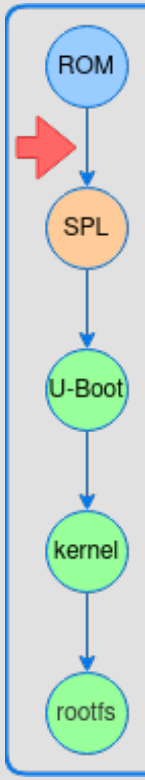
boot_flag & 0x1000 != 0

```
__int64 __fastcall decrypt_and_hash(__int64 key, __int64 iv, __int64 data,
{
  __int64 result; // x0
  struc_2 ctx; // [xsp+10h] [xbp-40h] BYREF

  result = aes_ctr_sha256_init(&ctx, key, 0x10, iv, 0x10u, 3, 0, 0, 5);
  if ( !result )
  {
    result = cry_sub_FFFF5CE4(&ctx, data, size, 1);
    if ( !result )
      result = cryHash_sub_FFFF5920(&ctx, hash, 0i64);
  }
  return result;
}
```

AES-CTR

SHA-256

```
v33 = encrypt ? 0xFFFF0030 : 0xFFFF0032;  // aes_ctr
*(xFE390008 + 0x2F8) = iv_len;            // CRYPTO_CHn_IV_LEN_0
*(xFE390008 + 0x3C) = v33;                // CRYPTO_BC_CTL
if ( a9_const5 == 5 )                     // true
{
  v34 = 0i64;
  ctx->field_B = 0x20;
  do
  {
    v35 = (v34 + xFE390008);
    v34 += 4i64;
    v35[0xE6] = 0;
  }
  while ( v34 != 0x40 );
  result = 0i64;
  if ( hash_from_tx_fifo )
    hash_ctl = 0xFFFF0026;                // SHA-256 | hw_pad_enable | hash_src_sel From TX-FIFO
  else                                    // SHA-256 | hw_pad_enable | hash_src_sel From RX-FIFO
    hash_ctl = 0xFFFF0024;
  *(xFE390008 + 0x40) = hash_ctl;
  ctx->magic = 0x5F5F5F5F;
  ctx->field_4 = a6_const3;
  ctx->field_8 = 0;
  ctx->encrypt = encrypt;
  ctx->hash_from_tx_fifo = hash_from_tx_fifo;
  ctx->field_C = 5;
  return result;
```



SecureMaskROM

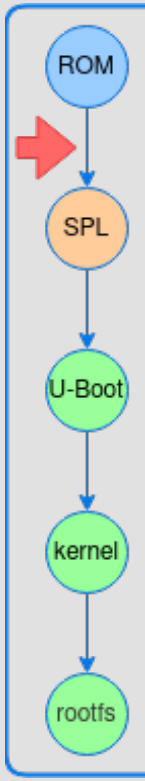
nonce

```
__int64 __fastcall load_binary__(header_struct *header, char *data, header_2_struct *fw_desc)
{
  unsigned int size; // w21
  unsigned int v6; // w0
  unsigned int v7; // w22
  int iv[4]; // [xsp+0h] [xbp-60h] BYREF
  char hash[32]; // [xsp+10h] [xbp-50h] BYREF

  size = (LOBYTE(fw_desc->data_size) | (HIBYTE(fw_desc->data_size) << 8)) << 9;
  if ( (header->boot_flag & 0x1000) != 0 )
  {
    iv[0] = (header->nonce[0] | (header->nonce[1] << 8)) | ((header->nonce[2] | (header->nonce[3] << 8)) << 0x10);
    iv[1] = (header->nonce[4] | (header->nonce[5] << 8)) | ((header->nonce[6] | (header->nonce[7] << 8)) << 0x10);
    iv[2] = 0;
    iv[3] = bswap32((fw_desc->counter[0] | (fw_desc->counter[1] << 8)) | ((fw_desc->counter[2] | (fw_desc->counter[3] << 8)) << 0x10));
    if ( !unk_FF00002C )
    {
      v7 = sub_FFFF0F98(0x20, 4u, 0);
      if ( v7 )
        return v7;
      unk_FF00002C = 1;
    }
    v6 = decrypt_and_hash(0i64, iv, data, size, hash);
  }
  else
  {
    v6 = ::hash(data, size, hash);
  }
  v7 = v6;
  if ( !v6 )
  {
    v7 = sec_memcmp(fw_desc->hash, hash, 0x20);
    sub_FFFF37B0(*word_FF000270 + 1, v7);
  }
  return v7;
}
```

nonce

Counter - сумма размеров предыдущих частей

Что-то странное



SecureMaskROM

keyladder

Что-то странное:
функция загрузки ключа

```
if ( !is_key_loaded )
{
  v7 = load_key(0x20, 4u, 0);
  if ( v7 )
    return v7;
  is_key_loaded = 1;
}
```

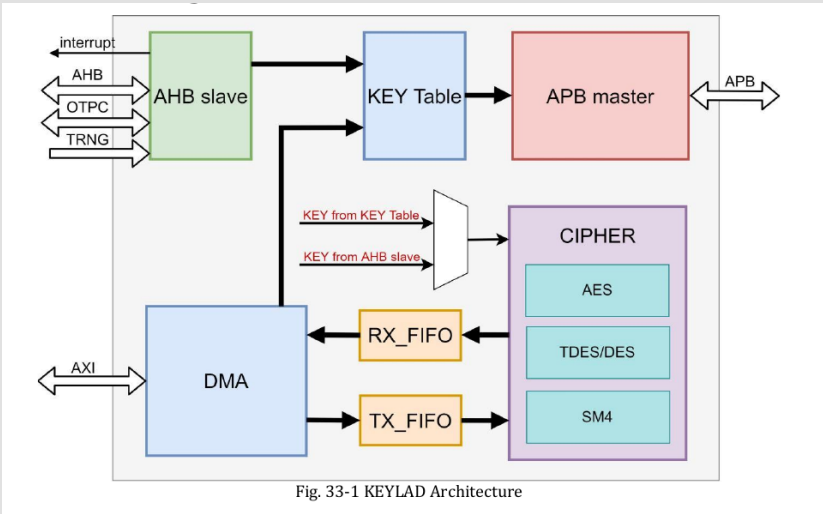
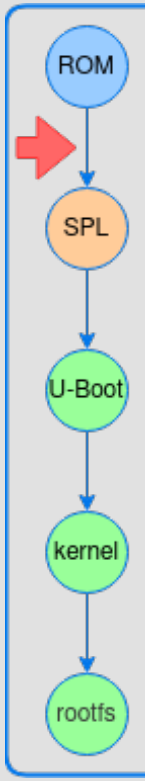


Fig. 33-1 KEYLAD Architecture

```
__int64 __fastcall load_key(int otp, unsigned int size, unsigned int a3)
{
  unsigned int v6; // w23
  unsigned int v7; // w22

  unk_FE380620 = 0;
  unk_FE380624 = 1;
  if ( size )
  {
    while ( 1 )
    {
      v6 = size >= 0x10 ? 0x10 : size;
      v7 = SECotpRead(0i64, otp, v6);
      if ( (v7 & 0x80000000) != 0 )
        break;
      size -= v6;
      otp += v6;
      if ( !size )
        goto LABEL_9;
    }
  }
  else
  {
    v7 = 0xFFFFFFFF;
LABEL_9:
    sub_FFFF4924(a3);
  }
  return v7;
}
```

Чтение фьюзов в nullptr
Если не знать про keyladder, выглядит немного странно



Rock eBOOT упаковщик

Шифруем бинари.

Формируем блок публичного ключа в Rockchip-овом формате

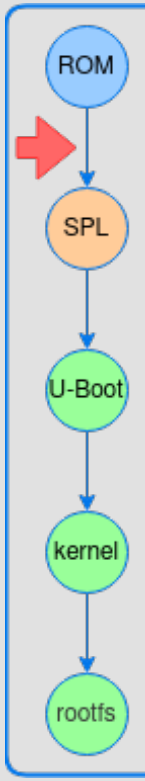
Формируем основной заголовок. В него входит nonce, флаги, включая флаг eBoot-a и указание того, что в прошивке 2 бинаря

Формируем заголовки бинарей. Добавляем хэши бинарей, а во второй — ещё и counter для расшифровки AESa

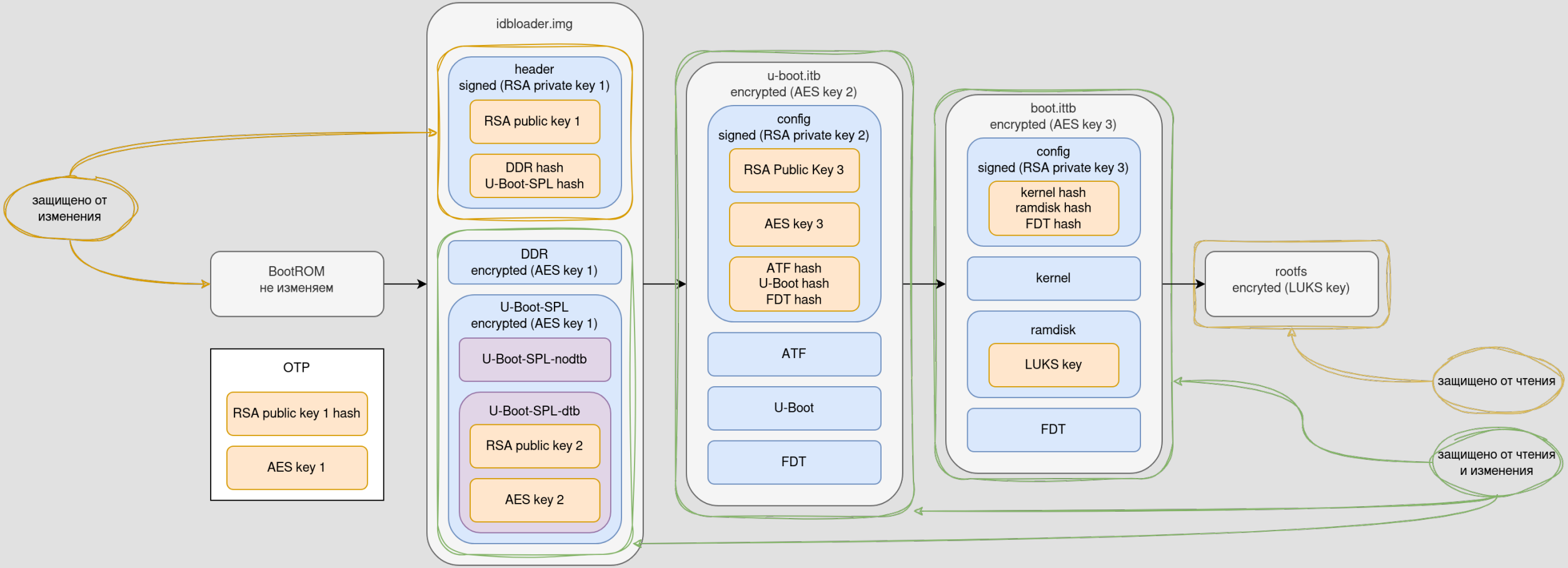
Подписываем хэш заголовка

packer.py

```
def pack(rsakey,aeskey,ddr,spl):
    nonce=get_random_bytes(8)
    aeskey=open(aeskey,'rb').read()

    ddr=open(ddr,'rb').read()
    if len(ddr)%0x200:
        ddr+=b'\x00'*(0x200-len(ddr)%0x200)
    a=AES.new(aeskey, AES.MODE_CTR,nonce=nonce)
    ddr=a.encrypt(ddr)

    fw=open(fw,'rb').read()
    if len(fw)%0x200:
        fw+=b'\x00'*(0x200-len(fw)%0x200)
    fw=a.encrypt(fw)

    key=RSA.importKey(open(keyfile,'rb').read())
    keyblock=(key.n).to_bytes(0x100)[::-1]
    keyblock+=b'\x00'*0x100
    keyblock=(key.e).to_bytes(0x10)[::-1]
    keyblock=(pow(2,2048+132)//key.n).to_bytes(0x100)[::-1]
    keyblock+=b'\x00'*0xf0

    flags=0x1011
    header=pad(struct.pack("<4sIHHIII8s",magic,0,size,2,flags,0,0,nonce),0x78)

    ddr_hash=SHA256.new(ddr).digest()
    header+=struct.pack("<HHIII",4,len(ddr)//0x200,0xffff_ffff,0,0)+b'\x00'*8+ddr_hash+b'\x00'*0x20

    fw_hash=SHA256.new(fw).digest()
    header+=struct.pack("<HHIII",4+len(ddr)//0x200,len(fw)//0x200,0xffff_ffff,0x02,len(ddr)//0x10)+b'\x00'*8+fw_hash+b'\x00'*0x20

    header=pad(header,0x200)
    header+=keyblock

    a=PKCS1_PSS.new(key)
    head_hash=SHA256.new(header)
    sign=a.sign(head_hash)

    return header+sign[::-1]+b'\x00'*0x100+ddr+fw
```



Encrypted BOOT
как выглядит цепочка



Summary
Было весело

Документация из опечаток, недосказанности и неоднозначности

- RK3588 has two bootwom:
  - Normal Bootrom in non-secure
  - Secure Bootrom in secure worl

15.3 Function Description

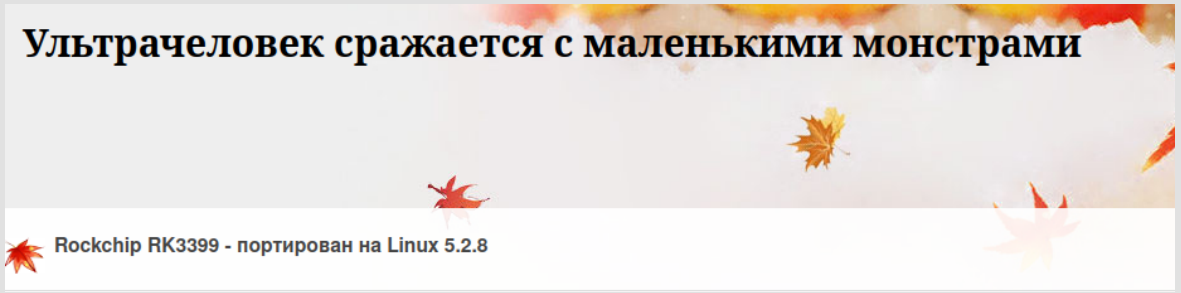
1.1
2.1
3.1
4.1
5.1

```
case TEST_BAD_BLOCK: /* Test Bad Block    : 3 */
case READ_SECTOR:          /* Read Pages        : 4 */
case READ_LBA:          /* Read Pages        : 4 */
case READ_SDRAM:          /* Write Pages       : 15 */
case READ_SPI_FLASH:
case READ_NEW_EFUSE:
    pCBW->ucCBWFlags = DIRECTION_IN;
    pCBW->ucCBWCBLength = 0x0a;
    break;
case WRITE_SECTOR:          /* Write Pages       : 5 */
case WRITE_LBA:          /* Write Pages       : 15 */
case WRITE_SDRAM:          /* Write Pages       : 15 */
case EXECUTE_SDRAM:
case ERASE_NORMAL:          /* Erase Blocks          : 6 */
case ERASE_FORCE:          /* Read Spare        : 11 */
case WRITE_EFUSE:
case WRITE_SPI_FLASH:
case WRITE_NEW_EFUSE:
case ERASE_LBA:
    pCBW->ucCBWFlags = DIRECTION_OUT;
    pCBW->ucCBWCBLength = 0x0a;
    break;
```

Не пишет. Но как?

```
                ADRP            X0, #aSStep7@PAGE ; "\n%s step 7\n"
                MOV             X1, X19
                ADD             X0, X0, #aSStep7@PAGEOFF ; "\n%s step 7\n"
                BL              printf
                CBZ             W24, loc_205564
                MOV             X1, X19
                ADRL            X0, aSStep8 ; "\n%s step 8\n"
                BL              printf
                MOV             X0, X20
                MOV             X1, #0
                BL              sub_22B1AC
                TBNZ            W0, #0x1F, loc_20569C
                ADRL            X20, aSStep10 ; "\n%s step 10\n"
                MOV             X1, X19
                ADRL            X0, aSStep9 ; "\n%s step 9\n"
                BL              printf

loc_2056F4                                  ; CODE XREF: sub_2054FC+204↓j
                MOV             X1, X19
                MOV             X0, X20
                BL              printf
                B               loc_2056F4
; End of function sub_2054FC
```

Шикарный Google-translate китайских блогов




```
fit_get_blob step 7

fit_get_blob step 8

fit_get_blob step 9
conf:
RKUSB: LUN 0, dev 0, hwpart 0, sector 0x0, count 0x7530000
```

Обязательно на github
Но потом.

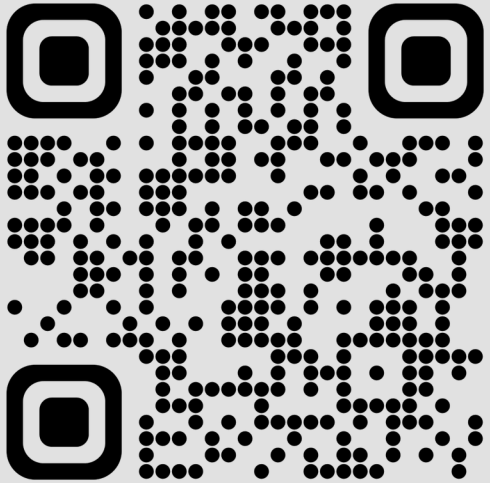
Сегодня:

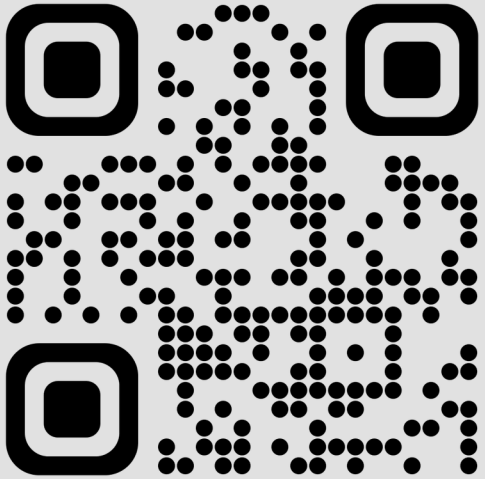
- Сборщик SPL с Eboot

Чуть позже:

- Набор утилит для прошивки ключей во фьюзы
- U-boot с поддержкой Secure Boot
- U-boot с Encrypted Boot
- Подробные инструкции

Наш телеграм канал





phd 2 Positive Hack Days Fest

от positive technologies

Спасибо!