Decrypting IPTV

BSides Sofia

Radoslav Gerganov
15 Apr 2022

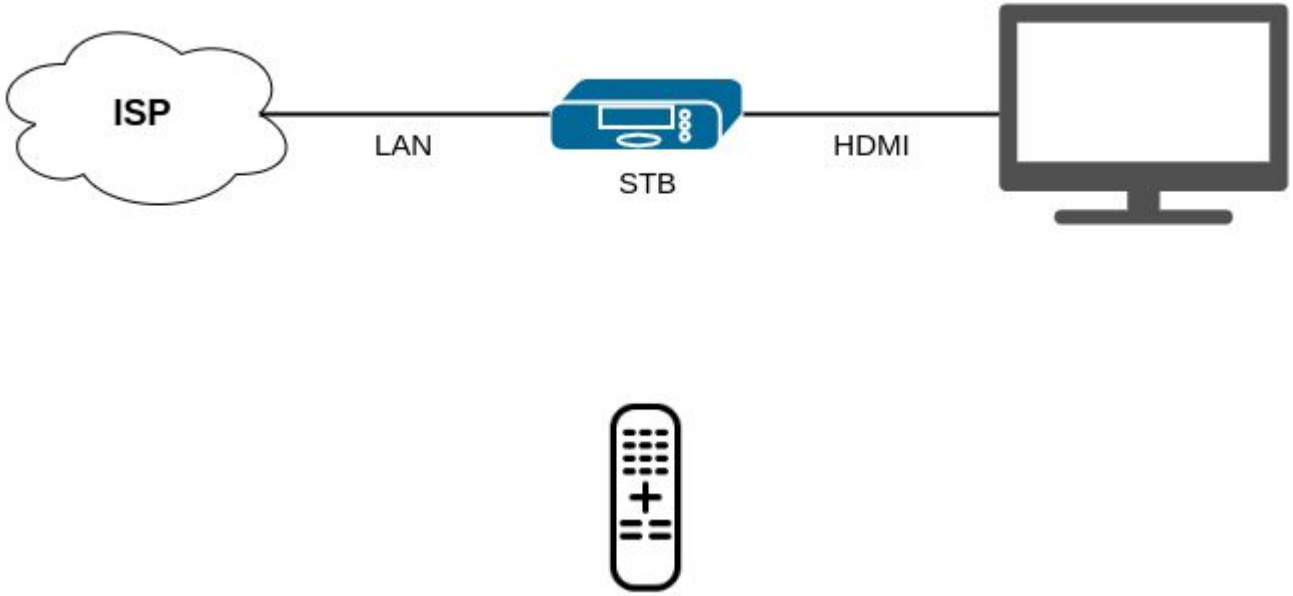
whoami

- Infosec @ VMware
  - Confidential Computing
  - Virtual HSMs
  - Fuzzing
- DISCLAIMER: views are my own
- github.com/rgerganov
- twitter.com/rgerganov
- https://xakcop.com

What is IPTV?

- Delivery of television content over Internet Protocol (IP) networks
- Supports both live TV and video-on-demand
- Usually delivered with set-top-boxes
- Protocols: IGMP, RTP, RTSP, UDP, …

IPTV deployment



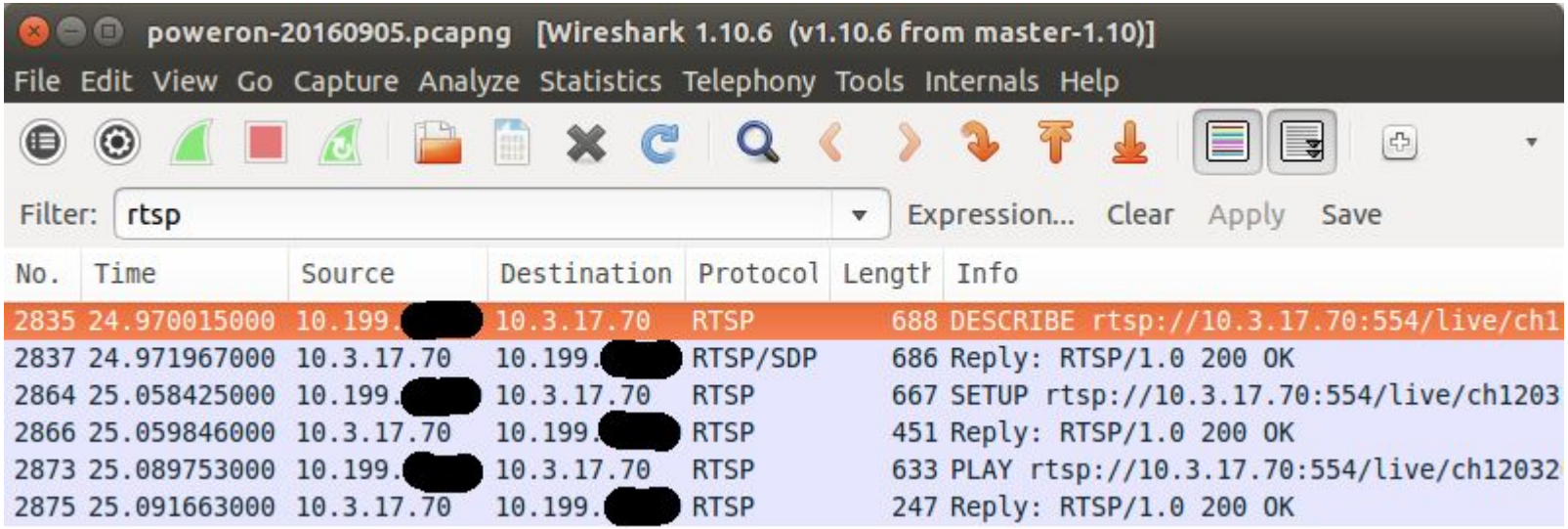
IPTV Security Model

- The only possible way to consume IPTV is with STB
- STB is operated only with the remote control
- Security through obscurity
- Flaws
  - No need to pay for more channels
  - No need to pay for more STBs
  - No need to pay for IPTV :)

What is STB?

- Crappy device made in China
- Runs Linux
- Firmware = HTML browser + Video player
- IPTV User Interface = HTML + JS

Six years ago (2016)



No server-side authz

- STB MAC addr used for authentication
- PLAY rtsp://10.3.17.70:554/live/**ch12032017033910402807**
- All channel IDs are in the JavaScript

```
...
top.jsSetChannelInfo('Nova Sport','ch11123010550877822586');
top.jsSetChannelInfo('History Channel','ch11123010550858280539');
top.jsSetChannelInfo('Viasat Nature','ch11123010550889166895');
top.jsSetChannelInfo('Viasat Explorer','ch11123010550851794216');
top.jsSetChannelInfo('Viasat History','ch11123010550812809435');
top.jsSetChannelInfo('National Geographic WILD','ch11123010550891597909');
top.jsSetChannelInfo('TV 7','ch11123010550856881207');
top.jsSetChannelInfo('bTV Cinema','ch11123010550841375257');
...
```

Swap channels with MitM

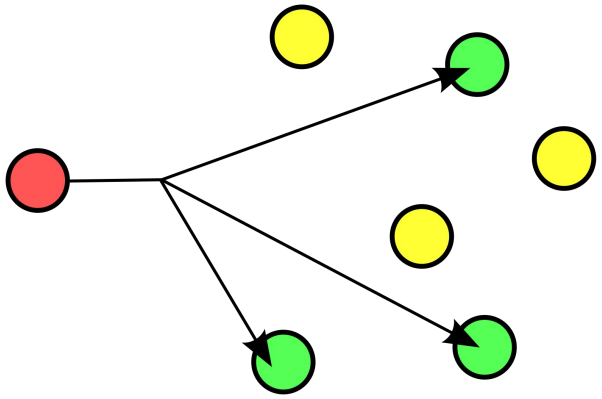
- Modify network traffic with MitM
- Replace crappy channels with paid channels
- https://github.com/rgerganov/nfqsed
- Example configuration:


```
# Fen TV / HBO HD
/ch13121216533621227255/ch12092914401822894191
# Folklor TV / HBO Comedy HD
/ch11123010551095204329/ch12092914413728185379
```

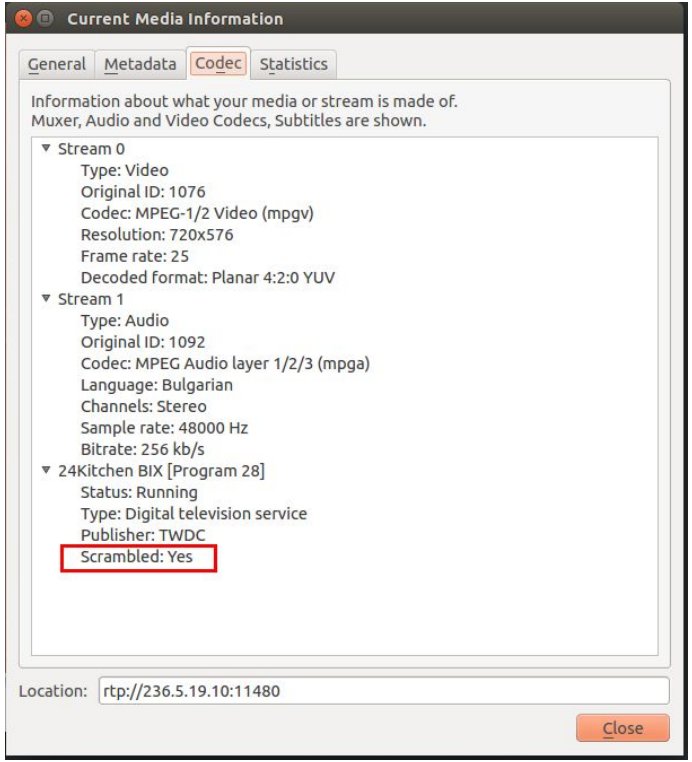
RaspberryPi in the middle

Fast forward to 2022

- Live streams delivered on multicast
- IGMP join/leave when switching TV channels
- More efficient for the IPTV provider
- IP range 224.0.0.0 - 239.255.255.255

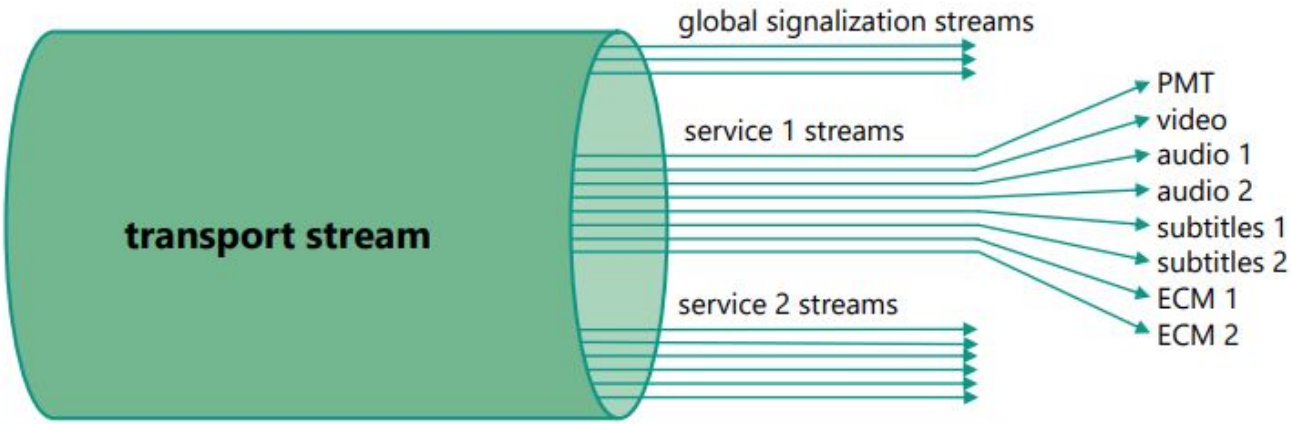


Let’s try with VLC



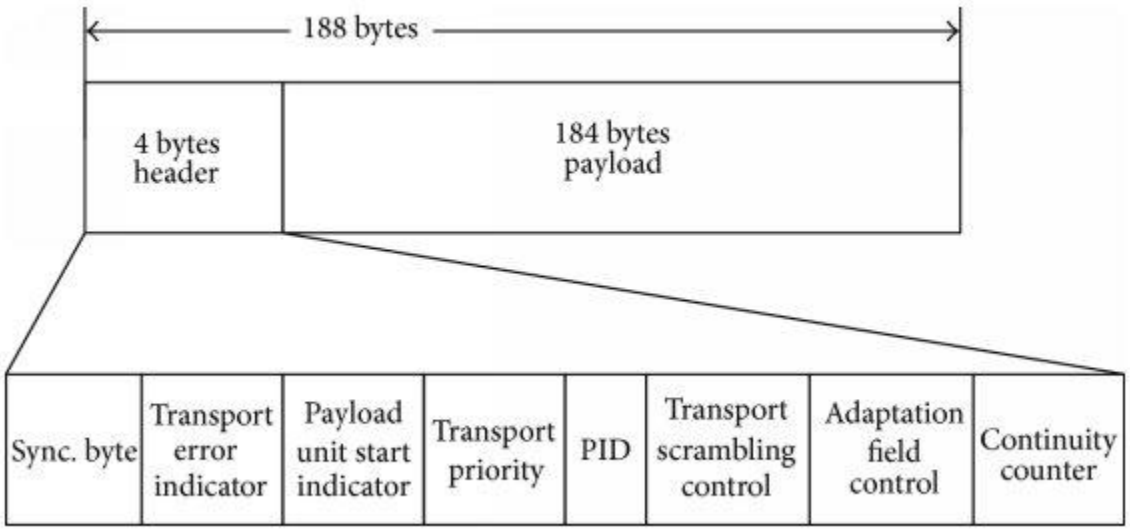
MPEG transport stream

- Multiplex of elementary streams
- Elementary streams are identified by PID
- Program map table (PMT)
- Entitlement control messages (ECM)



Source: tsduck.io

MPEG TS packet

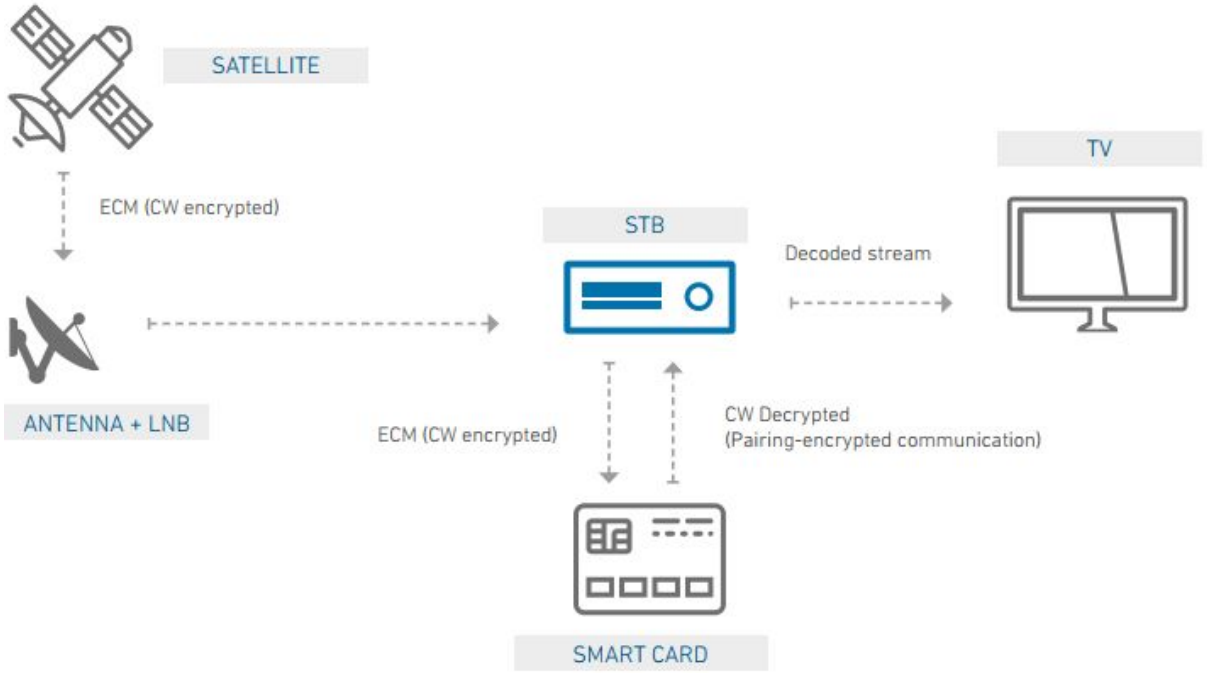


Conditional Access System (CAS)

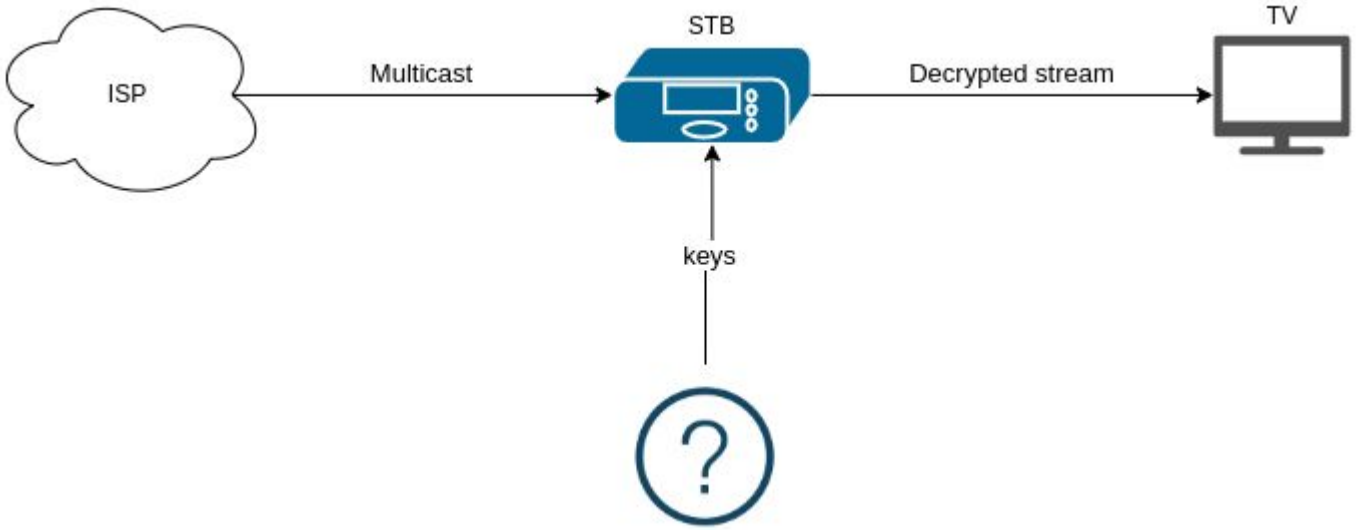
- Standard methods for obfuscating digital streams
- Access provided only to authorized users
- DVB Common Scrambling Algorithm (DVB-CSA)

CA ID	Name	Developed by	Introduced (year)
0x4AEB	Abel Quintic	Abel DRM Systems	2009
0x1700 – 0x1701, 0x1703 – 0x1721, 0x1723 – 0x1761, 0x1763 – 0x17ff, 0x5601 – 0x5604	VCAS DVB	Verimatrix Inc.	2010
0x2600	BISS	European Broadcasting Union	2002
0x2610	BISS2		2018
0x5581	Bulcrypt	Bulcrypt	2009
0x0606	Irdeto 1	Irdeto	1995

Subscription based satellite TV

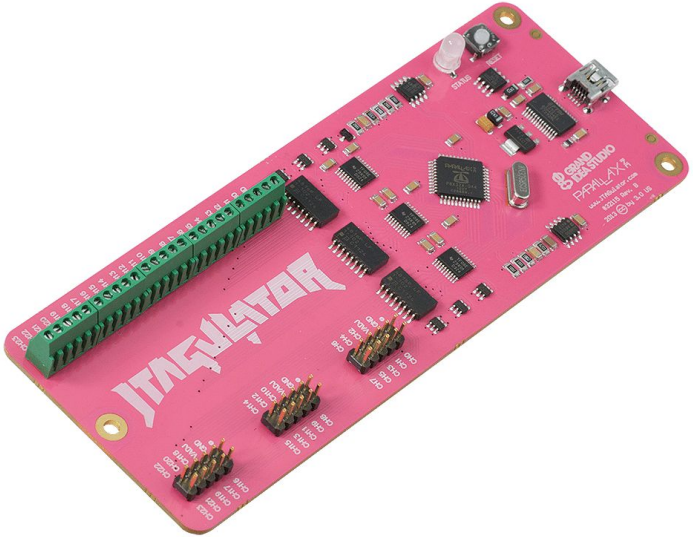


Subscription based IPTV

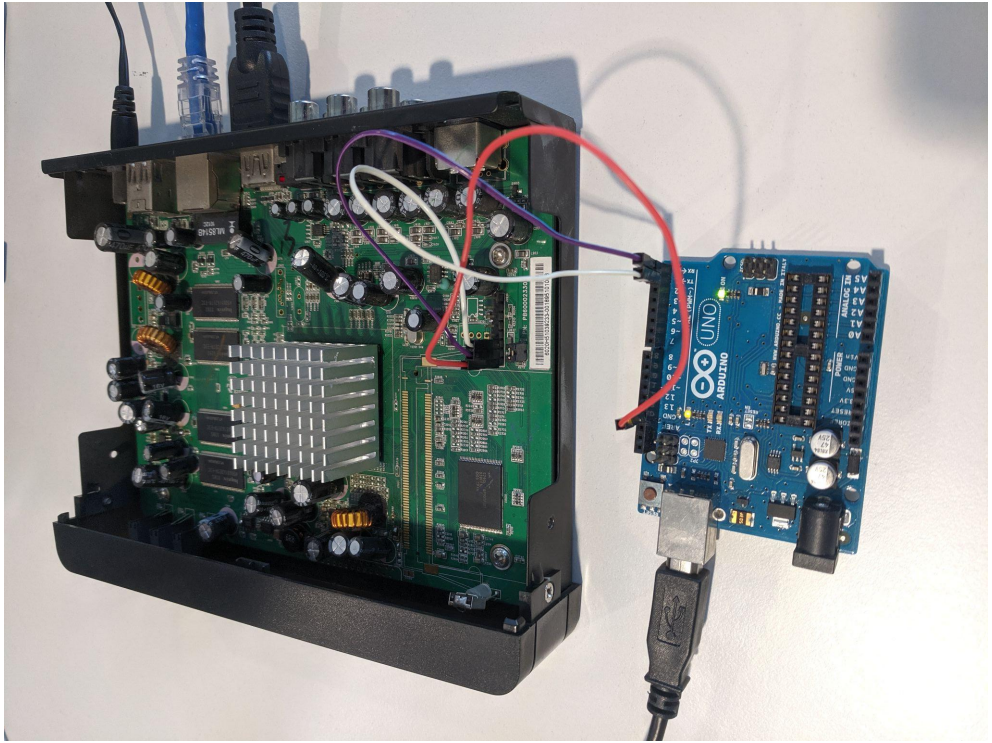


How to root STB

- Serial UART console
  - RX, TX, GND pins
- JTAG
  - TDI, TDO, TCK, TMS pins
- Power glitch
  - https://www.youtube.com/watch?v=dT9y-KQbqi4



Arduino as USB to Serial converter



r00t

```
# whoami
root

# cat /proc/cpuinfo
Processor       : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS        : 804.86
Features        : swp half thumb fastmult vfp edsp java
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xb76
CPU revision    : 7
Hardware        : mt85xx

# uname -a
Linux mtkhost 2.6.35 #1 PREEMPT Wed Feb 13 11:34:46 CST 2019 armv6l GNU/Linux

# cat /proc/meminfo
MemTotal:         386556 kB
MemFree:          115408 kB
Buffers:           28616 kB
Cached:            81816 kB
```

Reversing the firmware

- How channel keys are obtained?
- How channel keys are used?
- `lib**vm**client.so`
- Ghidra for static analysis (disassembler, decompiler)
- GDB for runtime analysis

```
  stb# gdbserver --attach :9090 <vplayer_pid>
local$ gdb-multiarch
  (gdb) target remote stb:9090
```

OpenSSL version

```
# strings libvmclient.so | grep -i openssl
MD5 part of OpenSSL 1.0.0d 8 Feb 2011
SHA part of OpenSSL 1.0.0d 8 Feb 2011
SHA1 part of OpenSSL 1.0.0d 8 Feb 2011
...
```

Decompiled function (part1)

```
2  code * Yn_07xvZrdrMv(undefined4 *param_1,undefined4 param_2,undefined4 *param_3,int *param_4)
3
4  {
5    code *pcVar1;
6    int iVar2;
7    undefined4 *puVar3;
8    code *pcVar4;
9    code *pcVar5;
10   undefined auStack128 [64];
11   undefined auStack64 [24];
12   undefined4 local_28;
13   undefined4 local_24;
14
15   pcVar4 = (code *)0x0;
16   *param_3 = 0;
17   RkVPYGdkwnhvhh0(auStack64);
18   Dd1CcRTdbYVlkGAQSF(auStack64,param_1);
19   p6mfT0ASCxegv6w7Fu(auStack64,auStack128,&local_24);
20   raAAimqDF6A4IKZohv(auStack64);
```

Decompiled function (part2)

```
21    puVar3 = (undefined4 *)*param_1;
22    if ((puVar3[3] & 4) == 0) {
23      if ((puVar3[0xb] == 0) ||
24         ((iVar2 = *param_4, iVar2 != puVar3[0xb] &&
25          ((puVar3[0xc] == 0 ||
26           ((iVar2 != puVar3[0xc] &&
27            ((puVar3[0xd] == 0 ||
28             ((iVar2 != puVar3[0xd] && ((puVar3[0xe] == 0 || (iVar2 != puVar3[0xe]))))))))))))) {
29        AZJwpqvTsWDQA(6,0x6b,0x6e,"p_sign.c",0x7d);
30        pcVar5 = (code *)0x0;
31      }
32      else {
33        pcVar5 = (code *)puVar3[9];
34        if (pcVar5 == (code *)0x0) {
35          AZJwpqvTsWDQA(6,0x6b,0x68,"p_sign.c",0x83);
36        }
```

EVP_SignFinal (part1)

github.com/openssl/openssl/blob/OpenSSL_1_0_0d/crypto/evp/p_sign.c

```
78  int EVP_SignFinal(EVP_MD_CTX *ctx, unsigned char *sigret, unsigned int *siglen,
79                EVP_PKEY *pkey)
80          {
81          unsigned char m[EVP_MAX_MD_SIZE];
82          unsigned int m_len;
83          int i,ok=0,v;
84          EVP_MD_CTX tmp_ctx;
85
86          *siglen=0;
87          EVP_MD_CTX_init(&tmp_ctx);
88          EVP_MD_CTX_copy_ex(&tmp_ctx,ctx);
89          EVP_DigestFinal_ex(&tmp_ctx,&(m[0]),&m_len);
90          EVP_MD_CTX_cleanup(&tmp_ctx);
```

EVP_SignFinal (part2)

```
123         if (!ok)
124                 {
125                 EVPerr(EVP_F_EVP_SIGNFINAL,EVP_R_WRONG_PUBLIC_KEY_TYPE);
126                 return(0);
127                 }
128
129         if (ctx->digest->sign == NULL)
130                 {
131                 EVPerr(EVP_F_EVP_SIGNFINAL,EVP_R_NO_SIGN_FUNCTION_CONFIGURED);
132                 return(0);
133                 }
```

```
code * Yn_07xvZrdrMv(undefined4 *param_1,undefined4 param_2,undefined4 *param_3,int *param_4)

{
  code *pcVar1;
  int iVar2;
  undefined4 *puVar3;
  code *pcVar4;
  code *pcVar5;
  undefined auStack128 [64];
  undefined auStack64 [24];
  undefined4 local_28;
  undefined4 local_24;

  pcVar4 = (code *)0x0;
  *param_3 = 0;
  RkVPYGdkwnhvhh0(auStack64);
  Dd1CcRTdbYVlkGAQSF(auStack64,param_1);
  p6mfT0ASCxegv6w7Fu(auStack64,auStack128,&local_24);
  raAAimqDF6A4IKZohv(auStack64);
  puVar3 = (undefined4 *)*param_1;
  if ((puVar3[3] & 4) == 0) {
    if ((puVar3[0xb] == 0) ||
       ((iVar2 = *param_4, iVar2 != puVar3[0xb] &&
        ((puVar3[0xc] == 0 ||
         ((iVar2 != puVar3[0xc] &&
          ((puVar3[0xd] == 0 ||
           ((iVar2 != puVar3[0xd] && ((puVar3[0xe] == 0 || (iVar2 != puVar3[0xe]))))))))))))) {
      AZJwpqvTsWDQA(6,0x6b,0x6e,"p_sign.c",0x7d);
      pcVar5 = (code *)0x0;
    }
    else {
      pcVar5 = (code *)puVar3[9];
      if (pcVar5 == (code *)0x0) {
        AZJwpqvTsWDQA(6,0x6b,0x68,"p_sign.c",0x83);
      }
      else {
        pcVar5 = (code *)(*pcVar5)(*puVar3,auStack128,local_24,param_2,param_3,param_4[5]);
      }
    }
  }
```

github.com/openssl/openssl/blob/OpenSSL_1_0_0d/crypto/evp/p_sign.c

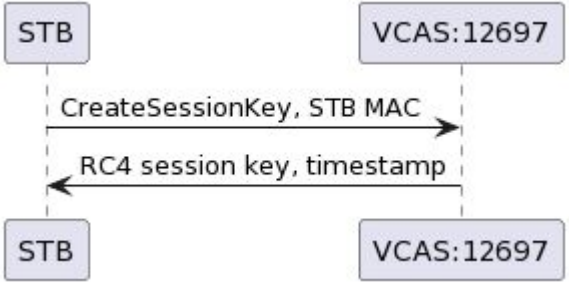
```
int EVP_SignFinal(EVP_MD_CTX *ctx, unsigned char *sigret, unsigned int *siglen,
            EVP_PKEY *pkey)
        {
        unsigned char m[EVP_MAX_MD_SIZE];
        unsigned int m_len;
        int i,ok=0,v;
        EVP_MD_CTX tmp_ctx;

        *siglen=0;
        EVP_MD_CTX_init(&tmp_ctx);
        EVP_MD_CTX_copy_ex(&tmp_ctx,ctx);
        EVP_DigestFinal_ex(&tmp_ctx,&(m[0]),&m_len);
        EVP_MD_CTX_cleanup(&tmp_ctx);

        if (ctx->digest->flags & EVP_MD_FLAG_PKEY_METHOD_SIGNATURE)
                {
                EVP_PKEY_CTX *pkctx = NULL;
                size_t sltmp = (size_t)EVP_PKEY_size(pkey);
                i = 0;

        if (!ok)
                {
                EVPerr(EVP_F_EVP_SIGNFINAL,EVP_R_WRONG_PUBLIC_KEY_TYPE);
                return(0);
                }

        if (ctx->digest->sign == NULL)
                {
                EVPerr(EVP_F_EVP_SIGNFINAL,EVP_R_NO_SIGN_FUNCTION_CONFIGURED);
                return(0);
                }
        return(ctx->digest->sign(ctx->digest->type,m,m_len,sigret,siglen,
                pkey->pkey.ptr));
        }
```

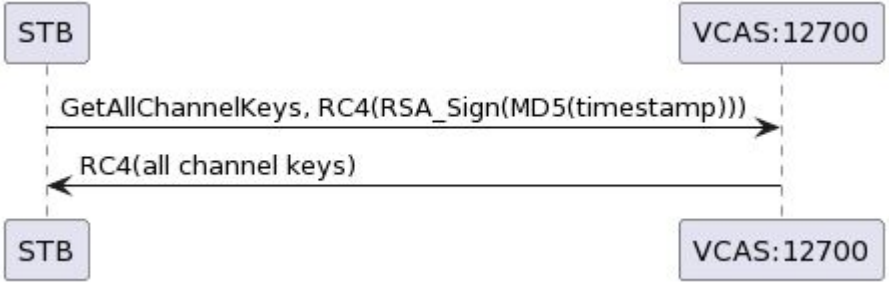
Signing with OpenSSL

```
void EVP_SignInit(EVP_MD_CTX *ctx, const EVP_MD *type);

int EVP_SignUpdate(EVP_MD_CTX *ctx, const void *d, unsigned int cnt);

int EVP_SignFinal(EVP_MD_CTX *ctx,unsigned char *sig,unsigned int *s, EVP_PKEY *pkey);
```

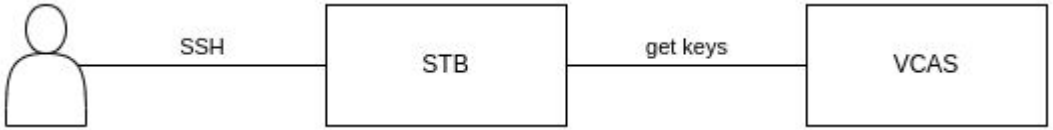
Obtain channel keys

- Key server (VCAS)
- 2 things needed for auth
  - STB MAC
  - RSA key
- Channel key = 128bit AES key





Using the STB as jump host



- STB MAC used for authentication
- Enable SSH on the STB (dropbear)
- SSH port forwarding to the rescue
  - `ssh -L 12697:vcas:12697 root@stb`
  - `ssh -L 12700:vcas:12700 root@stb`
  - `get-keys.py localhost:12697 localhost:12700`

Using channel keys

- Channel keys are rotated every 24h
- Even/odd keys are rotated every 10sec
- https://github.com/rgerganov/vmdecrypt

Key sharing

- Channel keys can be easily shared
- Similar to card sharing for SAT TV
- You just need multicast IPTV streams
- ISP send multicast to all customers
  - Even those not paying for IPTV :)

Summary

- Security through obscurity never works
- Both authn and authz are needed
- Multicast encryption is a hard problem
- Confidential Computing may solve this

Q&A