In radare2 /**c** means Cryptography

Sylvain Pelissier

September 4, 2020

R2CON

2020

Introduction

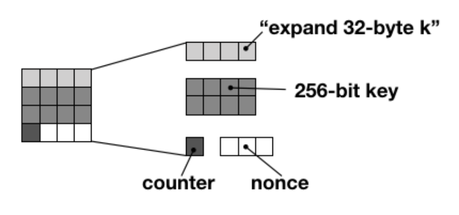
- Reverse or exploitation often imply Cryptography.
- Radare2 has some helper commands included.
- Practical use case where these commands are useful.
- Inner working of these commands.

Identify

- During a reverse, on crucial step may be to identify if a Cryptographic algorithms is used and which one.
- Often Crypto algorithms have constants, sbox or nonce which are public and may help to identify them.

Example

For example, Chacha is a stream cipher. It uses a 16-byte constant **”expand 32-byte k”**.
Meaning that finding this string in a binary would mean that it uses Chacha.



yara

- Yara is an open-source software developed by Virustotal to detect malware.
- It is based on flexible rules allowing the detection of malwares.
- Many rules are available at https://github.com/Yara-Rules/rules including rules to detect many cryptography algorithms.
- Available in **radare2-extras** and the yara commands are then accessible directly in r2.

r2-yara

```
$ r2pm -i yara yara-r2
```

yara rule

```
rule Chacha_256_constant{
  strings:
    $c0 = "expand 32-byte k"
  conditions:
    $c0
}
```

r2-yara

DEMO

Warning

Cryptographers tend to reuse primitives:

- Blake2 is a hash function based on Chacha.
- Argon2 is a password hash function using Blake2.
- Yara rule may trigger for Chacha constant but binary uses Argon2.

CRC and hash

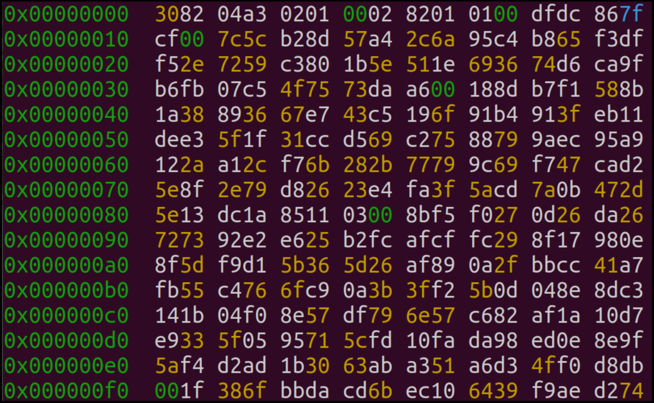
- **ph** prints the hash or the CRC of binary data.
- **ph?**: sha1, sha256, sha512, md4, xor, xorpair, parity, entropy, crc16, crc32, ...
- **rahash2** binary offers these features externally.
- **/h** finds if the hash of a block in the binary match the given hash:

```
/h md5 348a9791dc41b89796ec3808b5b5262f 512
```

Search RSA and ECC keys

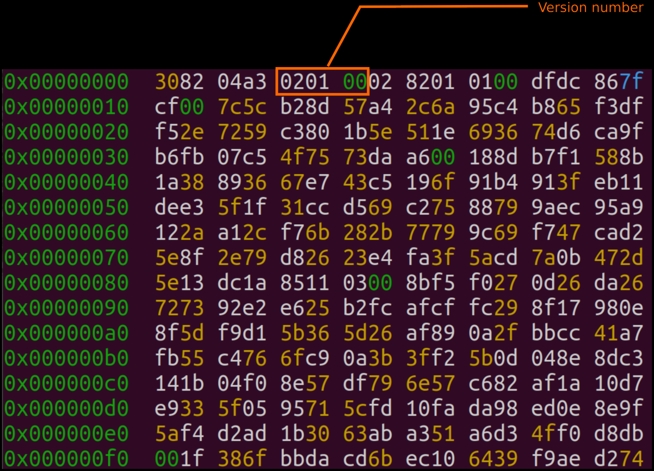
- RSA and Elliptic curve private keys are usually manipulated in ASN.1 format.
- This pattern structure can be parsed to find a key in memory.
- **/cr** command implements the search of private keys in r2.
- **/cd** command implements a similar feature to search certificates.

RSA key anatomy



```
0x00000000  3082 04a3 0201 0002 8201 0100 dfdc 867f
0x00000010  cf00 7c5c b28d 57a4 2c6a 95c4 b865 f3df
0x00000020  f52e 7259 c380 1b5e 511e 6936 74d6 ca9f
0x00000030  b6fb 07c5 4f75 73da a600 188d b7f1 588b
0x00000040  1a38 8936 67e7 43c5 196f 91b4 913f eb11
0x00000050  dee3 5f1f 31cc d569 c275 8879 9aec 95a9
0x00000060  122a a12c f76b 282b 7779 9c69 f747 cad2
0x00000070  5e8f 2e79 d826 23e4 fa3f 5acd 7a0b 472d
0x00000080  5e13 dc1a 8511 0300 8bf5 f027 0d26 da26
0x00000090  7273 92e2 e625 b2fc afcf fc29 8f17 980e
0x000000a0  8f5d f9d1 5b36 5d26 af89 0a2f bbcc 41a7
0x000000b0  fb55 c476 6fc9 0a3b 3ff2 5b0d 048e 8dc3
0x000000c0  141b 04f0 8e57 df79 6e57 c682 af1a 10d7
0x000000d0  e933 5f05 9571 5cfd 10fa da98 ed0e 8e9f
0x000000e0  5af4 d2ad 1b30 63ab a351 a6d3 4ff0 d8db
0x000000f0  001f 386f bbda cd6b ec10 6439 f9ae d274
```

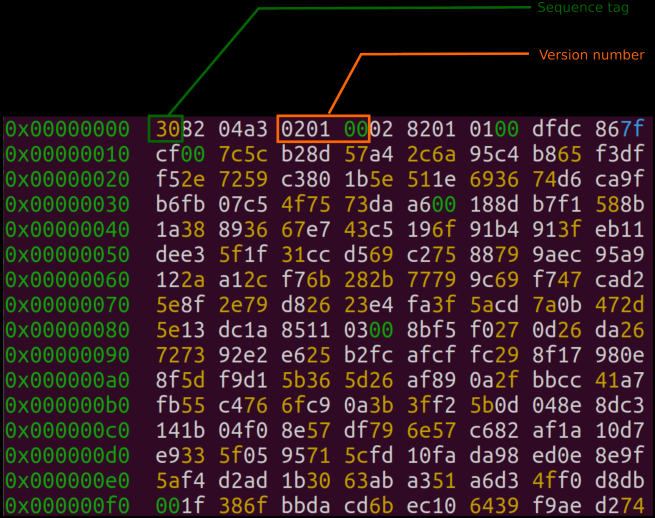
RSA key anatomy



Version number

```
0x00000000   3082 04a3 0201 0002 8201 0100 dfdc 867f
0x00000010   cf00 7c5c b28d 57a4 2c6a 95c4 b865 f3df
0x00000020   f52e 7259 c380 1b5e 511e 6936 74d6 ca9f
0x00000030   b6fb 07c5 4f75 73da a600 188d b7f1 588b
0x00000040   1a38 8936 67e7 43c5 196f 91b4 913f eb11
0x00000050   dee3 5f1f 31cc d569 c275 8879 9aec 95a9
0x00000060   122a a12c f76b 282b 7779 9c69 f747 cad2
0x00000070   5e8f 2e79 d826 23e4 fa3f 5acd 7a0b 472d
0x00000080   5e13 dc1a 8511 0300 8bf5 f027 0d26 da26
0x00000090   7273 92e2 e625 b2fc afcf fc29 8f17 980e
0x000000a0   8f5d f9d1 5b36 5d26 af89 0a2f bbcc 41a7
0x000000b0   fb55 c476 6fc9 0a3b 3ff2 5b0d 048e 8dc3
0x000000c0   141b 04f0 8e57 df79 6e57 c682 af1a 10d7
0x000000d0   e933 5f05 9571 5cfd 10fa da98 ed0e 8e9f
0x000000e0   5af4 d2ad 1b30 63ab a351 a6d3 4ff0 d8db
0x000000f0   001f 386f bbda cd6b ec10 6439 f9ae d274
```

RSA key anatomy

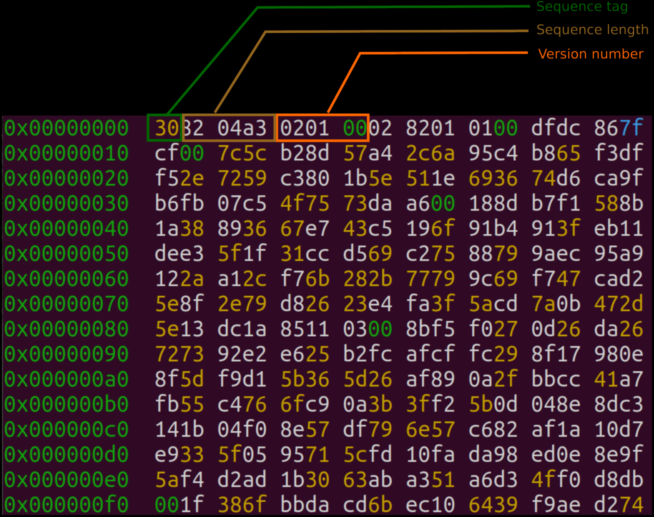


Sequence tag

Version number

```
0x00000000  3082 04a3 0201 0002 8201 0100 dfdc 867f
0x00000010  cf00 7c5c b28d 57a4 2c6a 95c4 b865 f3df
0x00000020  f52e 7259 c380 1b5e 511e 6936 74d6 ca9f
0x00000030  b6fb 07c5 4f75 73da a600 188d b7f1 588b
0x00000040  1a38 8936 67e7 43c5 196f 91b4 913f eb11
0x00000050  dee3 5f1f 31cc d569 c275 8879 9aec 95a9
0x00000060  122a a12c f76b 282b 7779 9c69 f747 cad2
0x00000070  5e8f 2e79 d826 23e4 fa3f 5acd 7a0b 472d
0x00000080  5e13 dc1a 8511 0300 8bf5 f027 0d26 da26
0x00000090  7273 92e2 e625 b2fc afcf fc29 8f17 980e
0x000000a0  8f5d f9d1 5b36 5d26 af89 0a2f bbcc 41a7
0x000000b0  fb55 c476 6fc9 0a3b 3ff2 5b0d 048e 8dc3
0x000000c0  141b 04f0 8e57 df79 6e57 c682 af1a 10d7
0x000000d0  e933 5f05 9571 5cfd 10fa da98 ed0e 8e9f
0x000000e0  5af4 d2ad 1b30 63ab a351 a6d3 4ff0 d8db
0x000000f0  001f 386f bbda cd6b ec10 6439 f9ae d274
```

RSA key anatomy



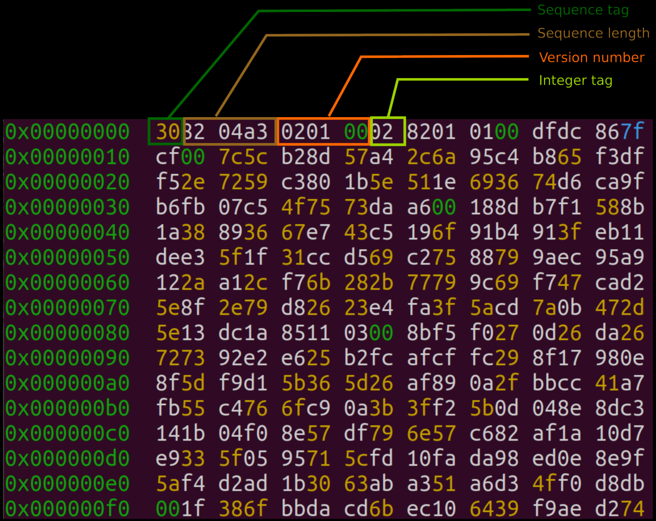
Sequence tag

Sequence length

Version number

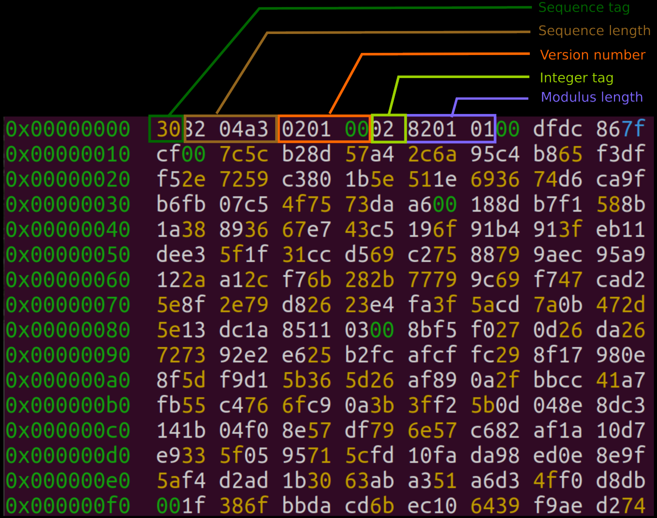
```
0x00000000  3082 04a3 0201 0002 8201 0100 dfdc 867f
0x00000010  cf00 7c5c b28d 57a4 2c6a 95c4 b865 f3df
0x00000020  f52e 7259 c380 1b5e 511e 6936 74d6 ca9f
0x00000030  b6fb 07c5 4f75 73da a600 188d b7f1 588b
0x00000040  1a38 8936 67e7 43c5 196f 91b4 913f eb11
0x00000050  dee3 5f1f 31cc d569 c275 8879 9aec 95a9
0x00000060  122a a12c f76b 282b 7779 9c69 f747 cad2
0x00000070  5e8f 2e79 d826 23e4 fa3f 5acd 7a0b 472d
0x00000080  5e13 dc1a 8511 0300 8bf5 f027 0d26 da26
0x00000090  7273 92e2 e625 b2fc afcf fc29 8f17 980e
0x000000a0  8f5d f9d1 5b36 5d26 af89 0a2f bbcc 41a7
0x000000b0  fb55 c476 6fc9 0a3b 3ff2 5b0d 048e 8dc3
0x000000c0  141b 04f0 8e57 df79 6e57 c682 af1a 10d7
0x000000d0  e933 5f05 9571 5cfd 10fa da98 ed0e 8e9f
0x000000e0  5af4 d2ad 1b30 63ab a351 a6d3 4ff0 d8db
0x000000f0  001f 386f bbda cd6b ec10 6439 f9ae d274
```

RSA key anatomy



```
0x00000000  3082 04a3 0201 0002 8201 0100 dfdc 867f
0x00000010  cf00 7c5c b28d 57a4 2c6a 95c4 b865 f3df
0x00000020  f52e 7259 c380 1b5e 511e 6936 74d6 ca9f
0x00000030  b6fb 07c5 4f75 73da a600 188d b7f1 588b
0x00000040  1a38 8936 67e7 43c5 196f 91b4 913f eb11
0x00000050  dee3 5f1f 31cc d569 c275 8879 9aec 95a9
0x00000060  122a a12c f76b 282b 7779 9c69 f747 cad2
0x00000070  5e8f 2e79 d826 23e4 fa3f 5acd 7a0b 472d
0x00000080  5e13 dc1a 8511 0300 8bf5 f027 0d26 da26
0x00000090  7273 92e2 e625 b2fc afcf fc29 8f17 980e
0x000000a0  8f5d f9d1 5b36 5d26 af89 0a2f bbcc 41a7
0x000000b0  fb55 c476 6fc9 0a3b 3ff2 5b0d 048e 8dc3
0x000000c0  141b 04f0 8e57 df79 6e57 c682 af1a 10d7
0x000000d0  e933 5f05 9571 5cfd 10fa da98 ed0e 8e9f
0x000000e0  5af4 d2ad 1b30 63ab a351 a6d3 4ff0 d8db
0x000000f0  001f 386f bbda cd6b ec10 6439 f9ae d274
```

RSA key anatomy



Sequence tag

Sequence length

Version number

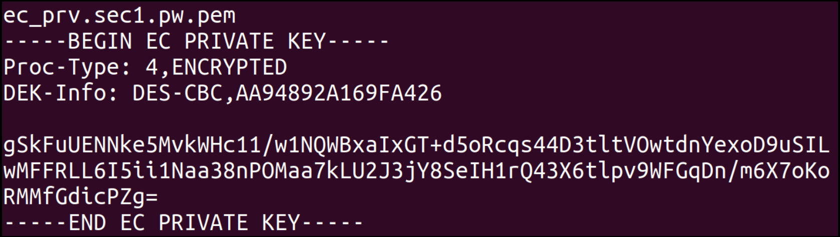
Integer tag

Modulus length

```
0x00000000  3082 04a3 0201 0002 8201 0100 dfdc 867f
0x00000010  cf00 7c5c b28d 57a4 2c6a 95c4 b865 f3df
0x00000020  f52e 7259 c380 1b5e 511e 6936 74d6 ca9f
0x00000030  b6fb 07c5 4f75 73da a600 188d b7f1 588b
0x00000040  1a38 8936 67e7 43c5 196f 91b4 913f eb11
0x00000050  dee3 5f1f 31cc d569 c275 8879 9aec 95a9
0x00000060  122a a12c f76b 282b 7779 9c69 f747 cad2
0x00000070  5e8f 2e79 d826 23e4 fa3f 5acd 7a0b 472d
0x00000080  5e13 dc1a 8511 0300 8bf5 f027 0d26 da26
0x00000090  7273 92e2 e625 b2fc afcf fc29 8f17 980e
0x000000a0  8f5d f9d1 5b36 5d26 af89 0a2f bbcc 41a7
0x000000b0  fb55 c476 6fc9 0a3b 3ff2 5b0d 048e 8dc3
0x000000c0  141b 04f0 8e57 df79 6e57 c682 af1a 10d7
0x000000d0  e933 5f05 9571 5cfd 10fa da98 ed0e 8e9f
0x000000e0  5af4 d2ad 1b30 63ab a351 a6d3 4ff0 d8db
0x000000f0  001f 386f bbda cd6b ec10 6439 f9ae d274
```

RSA key example

Private keys are usually stored encrypted in a file.



```
ec_prv.sec1.pw.pem
-----BEGIN EC PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,AA94892A169FA426

gSkFuUENNke5MvkWHc11/w1NQWBxaIxGT+d5oRcqs44D3tltVOwtdnYexoD9uSIL
wMFFRLL6I5ii1Naa38nPOMaa7kLU2J3jY8SeIH1rQ43X6tlpv9WFGqDn/m6X7oKo
RMMfGdicPZg=
-----END EC PRIVATE KEY-----
```

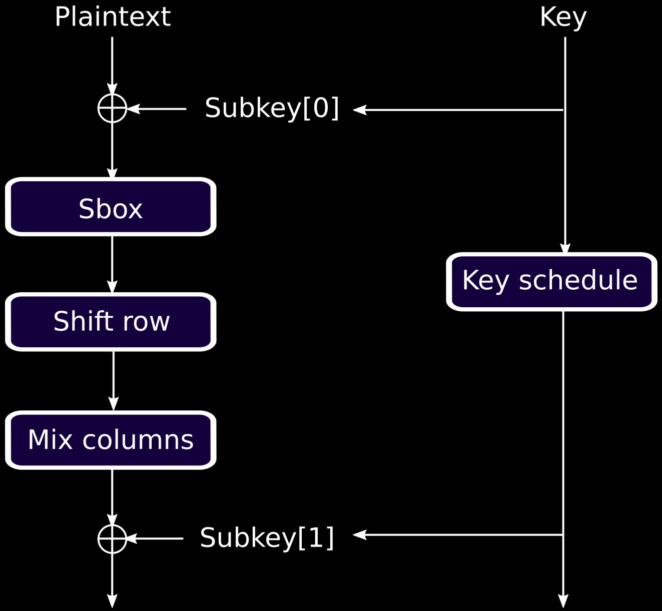
The private key is decrypted with a passphrase given by the user or the binary directly.

mbedTLS RSA key example

As soon as the key is decrypted it is in clear in ASN.1 format.

DEMO

AES



Plaintext

Key

Subkey[0]

Sbox

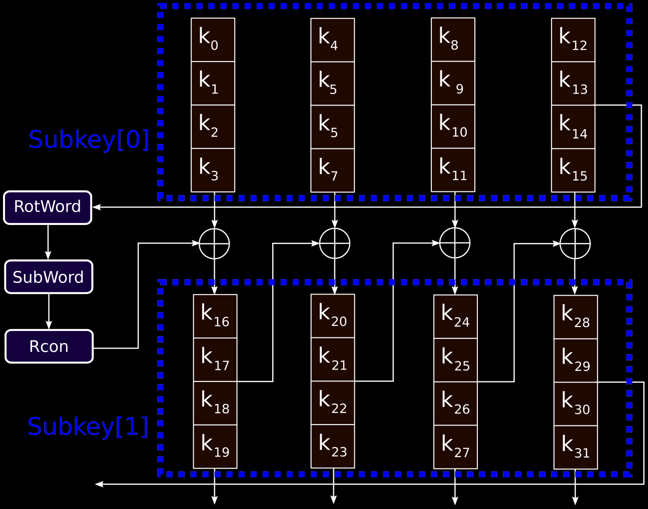
Shift row

Mix columns

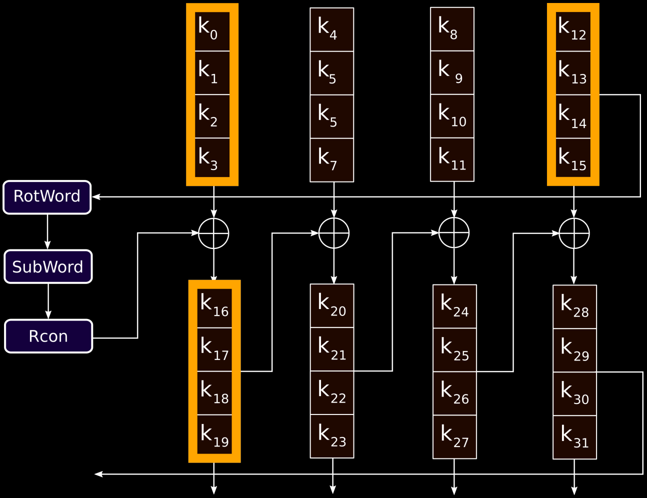
Key schedule

Subkey[1]

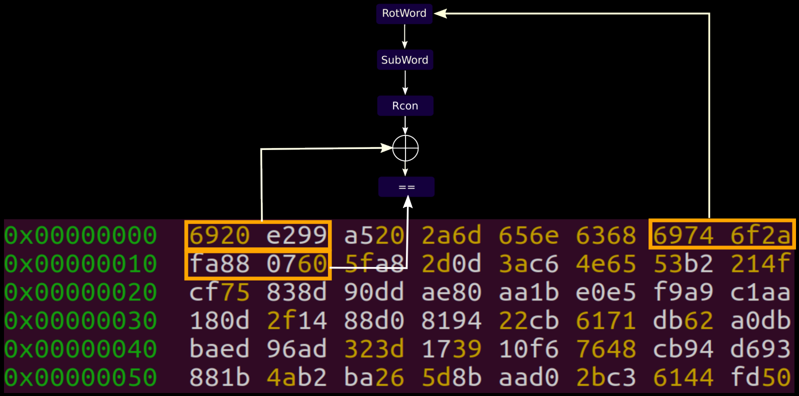
AES 128-bit key schedule



AES 128-bit key schedule

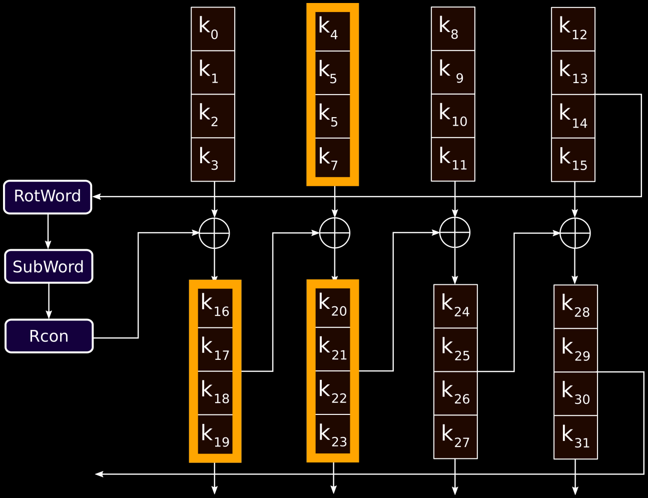


AES 128-bit key schedule

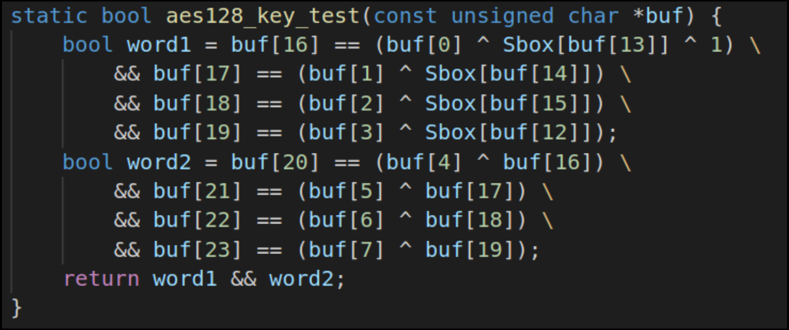


```
0x00000000  6920 e299 a520 2a6d 656e 6368 6974 6f2a
0x00000010  fa88 0760 5fa8 2d0d 3ac6 4e65 53b2 214f
0x00000020  cf75 838d 90dd ae80 aa1b e0e5 f9a9 c1aa
0x00000030  180d 2f14 88d0 8194 22cb 6171 db62 a0db
0x00000040  baed 96ad 323d 1739 10f6 7648 cb94 d693
0x00000050  881b 4ab2 ba26 5d8b aad0 2bc3 6144 fd50
```

AES 128-bit key schedule



AES key search



```
static bool aes128_key_test(const unsigned char *buf) {
    bool word1 = buf[16] == (buf[0] ^ Sbox[buf[13]] ^ 1) \
        && buf[17] == (buf[1] ^ Sbox[buf[14]]) \
        && buf[18] == (buf[2] ^ Sbox[buf[15]]) \
        && buf[19] == (buf[3] ^ Sbox[buf[12]]);
    bool word2 = buf[20] == (buf[4] ^ buf[16]) \
        && buf[21] == (buf[5] ^ buf[17]) \
        && buf[22] == (buf[6] ^ buf[18]) \
        && buf[23] == (buf[7] ^ buf[19]);
    return word1 && word2;
}
```

AES key search

- **radare2** now supports 128, 192 and 256-bit key search.
- The search can be applied on debug memory, process image, memory dump, ...
- **aeskeyfind** use this idea for 128 and 256-bit key but allows to have error in the key schedule.
- **Interrogate** allows key search for AES, Serpent, Twofish ciphers and RSA keys.

r2 AES search

DEMO

Decrypt

Encryption, decryption, hash and encoders are integrated in r2 under **woD** and **woE** commands.

The file has to be open in write mode and the result will be written directly in place.

woE? or **woD?** to have a full list of supported algorithms.

woE woD

DEMO

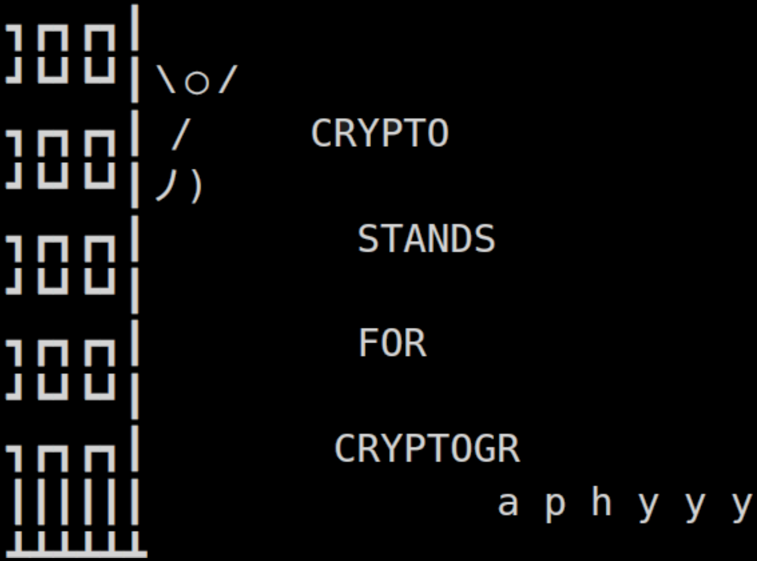
Conclusion

- Many features are already included in r2.
- Easy to extend yara rules or add new algorithms.
- Reverse and analysis happen entirely in r2, no need of external tools.

Possible contributions

- Add yara rules in `https://github.com/Yara-Rules/rules/tree/master/crypto`.
- Add new algorithm like SHA-3, Chacha, .. in **libr/crypto**.
- Add key search for other algorithms in **libr/search**.

Questions



CRYPTO

STANDS

FOR

CRYPTOGR

a p h y y y