How to Register WinRAR for a License File

GreenYun

July 9, 2021

1 Introduction

WinRAR is a file archiver developed by Eugene Roshal. Trail version of WinRAR can be download and distributed freely. Sold and supported by win.rar GmbH, WinRAR is allowed to be used in a 40-day test period. A license should be purchased to continue using the software.

After a person/organization/company pay to win.rar GmbH, one will receive a file named "RARreg.key" for registering the software. The file may be sent on a CD or via email. Put the file and the installer within a same directory or copy it to the installation directory and complete the registration.

The proper way to register for a license file is pay to win.rar GmbH. Yet this article will discuss the generation process of "RARreg.key". (Valid from version 4.x till current.)

2 Prerequisites

To generate the license file, one must provide his name or the company name — referred to as "Name". Name containing characters other than ASCII characters is discouraged. (May lead to wrong key.)

Another information is provided by the receipt after the payment. "License Type" is a string that usually describes how many copies is authorized the buyer to keep. For instance, "Single PC usage license", "1000 PC usage license", etc.

3 A Variant of SHA-1 Algorithm

WinRAR adopted SHA-1 algorithm as message hash method. In general SHA-1 algorithm implementation, the digest S is combined with 5 state values as

$$S = S_0 S_1 S_2 S_3 S_4$$

where S_i ($i \in \{0, 1, 2, 3, 4\}$) are 32-bit unsigned integers.

The extra step before generate S is doing byte-wise reverse for each S_i. After that concatenate them as S, a 160-bit unsigned integer. Note that we will store S or other data discussed below in big-endian (or network order) — the most significant byte in the front (or "on the left").

3.1 Yet Another Variant

There is another SHA-1 function used by WinRAR with all initial state values set to zero. This function is found as the only use to generate a secret numbers with a zero-length string input, and the answer is

```
1050D90D0F27A54653461BD1B4E33C7C0FFD8D43
```

4 Digital Signature Algorithm

The digital signature algorithm (the DSA) used by WinRAR is a variant of the SM2 digital signature algorithm.

4.1 The Composite Field

WinRAR chose a composite field of $\mathbb{F}_{(2^{15})^{17}}$, described as follows.

4.1.1 The Base Field

The base field of $\mathbb{F}_{2^{15}}$ is generated by the primitive polynomial

$$B(x) = x^{15} + x + 1$$

where the coefficients are in $\mathbb{F}_2$.

$\forall a(x) \in \mathbb{F}_{2^{15}}$,

$$a(x) = a_{14}x^{14} + a_{13}x^{13} + \cdots + a_1 x + a_0$$

coefficients are combined as a 15-bit series, denoted as

$$a = a_{14}a_{13}\cdots a_1 a_0$$

4.1.2 The Extension Field

The extension field of $\mathbb{F}_{(2^{15})^{17}}$ is constructed by the primitive polynomial

$$E(x) = x^{17} + x^3 + 1$$

where the coefficients are in finite field $\mathbb{F}_{2^{15}}$.

$\forall b(x) \in \mathbb{F}_{(2^{15})^{17}}$,

$$b(x) = b_{16}x^{16} + b_{15}x^{15} + \cdots + b_1 x + b_0$$

and b is denoted b as

$$b = b_{16}b_{15}\cdots b_1 b_0$$

Note that, $\forall b_i \in \mathbb{F}_{2^{15}}$, $i \in \{0, 1, \cdots, 15, 16\}$, which means b_i can be translated into a 15-bit series, and the total length of b is $15 \times 17 = 255$ bits.

4.2 The Elliptic Curve

The selected curve C is

$$y^2 + xy = x^3 + \alpha x^2 + \beta$$

where $x, y, \alpha, \beta \in \mathbb{F}_{(2^{15})^{17}}$. WinRAR chose $\alpha = 0$ and $\beta = 161$, and a base point $G \in C$: (all numbers below are denoted as hexadecimal)

$$G = (x_G, y_G)$$
$$x_G = \texttt{56FDCBC6A27ACEE0CC2996E0096AE74FEB1ACF220A2341B898B549440297B8CC}$$
$$y_G = \texttt{20DA32E8AFC90B7CF0E76BDE44496B4D0794054E6EA60F388682463132F931A7}$$

And the order of G:

$$\mu = \texttt{1026DD85081B82314691CED9BBEC30547840E4BF72D8B5E0D258442BBCD31}$$

4.3 Key Generation

WinRAR use some string (ASCII) as seed to generate private–public key pair.

4.3.1 The Private Key

First, generate hash digest for the input message. Calculate the digest for the input message if the length of input message is not zero, use the method described in section 3. Assign the digest to g.

If zero-length string is input, directly assign

$$g = \texttt{CDE43B4C6847B9D5DC5EF4A350265329EB3EB781}$$

Now we treat g a 20-byte octet stream, (significant byte first,) and concatenate a counter c of a 32-bit unsigned integer after the last byte of g. We will use $\|$ to denote "concatenation", which means a message M should be

$$M = g \parallel c$$

which is a 25-byte stream.

The loop strats by setting the counter c to 1. Send M as message to the SHA-1 function described in section 3, and store the digest as S. Obviously, S is a 20-byte octet stream, denoted as

$$S = S_{19} S_{18} \cdots S_1 S_0$$

in network order, where S_i ($i \in \{0, 1, \cdots, 18, 19\}$) are the bytes.

Assume k is another octet stream, with zero-length before the loop starts. The most least two bytes of S will be taken to append to the left side of k:

$$k = S_1 \parallel S_0 \parallel k$$

Loop this process for 15 rounds, after each round increment c by 1 and update to M.

If the digest after the i-time loop is denoted as S^i, the final k after all 15 round will looks like:

$$k = S_1^{15} S_0^{15} S_1^{14} S_0^{14} \cdots S_1^2 S_0^2 S_1^1 S_0^1$$

and this is the private key we generated.

To verify the key generator, check if an empty message input generates $k = k_0$, and k_0 described as follows:

$k_0 =$ `59FE6ABCCA90BDB95F0105271FA85FB9F11F467450C1AE9044B7FD61D65E`

4.3.2 The Public Key

As the base point G on the elliptic curve is known (section 4.2), the public key is calculated by multiplying the private key k to the base point, according to elliptic curve arithmetics

$$P = k \cdot G$$

To verify the key generator, check if k_0 (as described before) generates $P = P_0$, and

$P_0 = (x_P, y_P)$

$x_P =$ `3861220ED9B36C9753DF09A159DFB148135D495DB3AF8373425EE9A28884BA1A`

$y_P =$ `12B64E62DB43A56114554B0CBD573379338CEA9124C8443C4F50E6C8B013EC20`

A public key compress method is used by the SM2 digital signature algorithm. But WinRAR followed only some simplified steps:

1. Let $P = (x_P, y_P)$ on the elliptic curve. if $x_P = 0$, $\tilde{y}_P = 0$; or else $\tilde{y}_P$ is the most least (right-most) bit of $e = y_P \cdot x_P^{-1}$ of the composite field $\mathbb{F}_{(2^{15})^{17}}$;
2. Concatenate x_P and $\tilde{y}_P$ and obtain the bit stream.

The compressed public key is denoted as

$$\tilde{P} = x_P \parallel \tilde{y}_P$$

4.4 The Signing Process

A message M is signed using a private key k, following these steps:

1. Pick a random number n, which satisfies $0 < n < \mu$;

2. Generate a digest h via the algorithm described in section 3, and extend h by pushing the least 10 bytes of the secret number (described in 3.1) to its left side, which has total 30 bytes and may looks like

$$h = \texttt{1BD1B4E33C7C0FFD8D43} \parallel \mathrm{Sha}_1\,(M)$$

3. For a point $P = (x_P, y_P)$ on the elliptic curve, let $X\,(P) = x_P$, then calculate

$$r \equiv X\,(n \cdot G) + h \mod \mu$$

4. Check if either $r = 0$ or $r + n = \mu$, go to step 1, else continue;

5. Calculate

$$s \equiv n - k \cdot r \mod \mu$$

6. Check if $s = 0$, go to step 1 or obtain signature (r, s).

5 The Generation of the License

Assume the input message "Name" and "License Type" are denoted as U and L, and follow the next steps to generate the license:

1. Follow the steps in section 4.3 to obtain private–public key pair with input U, and convert to the compressed public key form $\tilde{P}_U$;

2. Convert $\tilde{P}_U$ into hexadecimal string form, pad with `0` on the left side until the length of the string is 64;

3. Split the string form of $\tilde{P}_U$ into two parts — the first 48 characters (s^+) and the remainders (s^-);

4. Let D_3 be a string that is constructed as follows:

$$D_3 = \texttt{"60"} \parallel s^+$$

   where text between two double quotation mark is a string literal;

5. Follow the steps in section 4.3 to obtain private–public key pair with input D_3, and convert to the compressed public key form $\tilde{P}_3$;

6. Convert $\tilde{P}_3$ into hexadecimal string form, pad with `0` on the left side until the length of the string is 64, denoted as D_0;

7. Let I be a 20-character string that is constructed as follows:

$$I = s^- \parallel D_{0,0} D_{0,1} D_{0,2} D_{0,3}$$

   where $D_{0,i}$ is the ith character of the string D_0;
   (*Note:* In practice, WinRAR does not case the contents of I at all.)

8. Use the algorithm described in 4.4 with L as message input and k_0 as private key input, obtain signature (r_L, s_L);

9. Convert r_L and s_L into hexadecimal string form, s_L^+ and s_L^-, pad each with `0` on its left side until both length are 60; (Remain unchanged if the length is greater than 60)

10. Let D_1 be a string that is constructed as follows:

$$D_1 = \texttt{"60"} \parallel s_L^- \parallel s_L^+$$

11. Construct a message string M_1 as

$$M_1 = U \parallel D_0$$

and sign it with private key k_0, obtain (r_1, s_1);

12. Convert r_1 and s_1 into hexadecimal string form, s_1^+ and s_1^-, pad each with `0` on its left side until both length are 60; (Remain unchanged if the length is greater than 60)

13. Construct a string D_2 as

$$D_2 = \texttt{"60"} \parallel s_1^- \parallel s_1^+$$

14. A CRC32 checksum is calculated using the message string

$$L \parallel U \parallel D_0 \parallel D_1 \parallel D_2 \parallel D_3$$

15. Convert the checksum into **decimal** string form s_c, pad with `0` on the left side until the length is 10;

16. Let l_0, l_1, l_2 and l_3 are the length of D_0, D_1, D_2 and D_3, respectively, and convert l_i into decimal string forms $s_{l,i}$;

17. Let D be a string constructed as follows:

$$D = s_{l,0} \parallel s_{l,1} \parallel s_{l,2} \parallel s_{l,3} \parallel D_0 \parallel D_1 \parallel D_2 \parallel D_3 \parallel s_c$$

5.1 Output

The first line: a string literal: `"RAR registration data"`.
The second line: U.
The third line: L.
The fourth line: combined with a string literal `"UID="` and I.
The following lines: D separated into 7 lines, while the first 6 lines have 54 characters each.

Reference

1. https://en.wikipedia.org/wiki/WinRAR
2. https://github.com/bitcookies/winrar-keygen
3. https://github.com/obaby/winrar-keygen
4. "GB/T 32918—2016: Information security technology—Public key cryptographic algorithm SM2 based on elliptic curves"