RIFT Authenticator explained

Table of Contents

1. Communication with TRION's servers

All communication uses the HTTPS protocol.

1.1. TRION server host names

We're communicating with the following TRION servers:

- The TRION Authentication server: https://auth.trionworlds.com
  The request method is HTTP-POST.
- The TRION API server: https://rift.trionworlds.com
  The request method is HTTP-GET.

1.2. SSL certificate verification

The SSL certificates for the server must always have a CN (common name) that ends with one of the following host names:

- .trionworlds.com
- .triongames.com
- .trionworld.priv
- .triongames.priv

1.3. User-Agent

The user agent used by the RIFT Authenticator for Android is somehting like this:

RIFT Mobile Authenticator {0}; Android {1} ({2}); {3}, {4}, {5};

Placeholder	Description	Example
{0}	Authenticator application version.	1.0.4
{1}	Android version.	2.3.3
{2}	Android SDK version.	10
{3}	Android product.	GINGERBREAD*
{4}	Cell phone model.	HTC Desire*
{5}	Cell phone brand.	O2*

* I'm not sure if this is correct.

2. Secret key creation

2.1. Create a device id

The device id can be any combination of the characters A-Z and digits 0-9. I don't know what the maximum length is, but you can at least use 32 characters and digits.

Hint: You should create a device id that can always be calculated for a mobile device (simple) or a computer (not so simple).

When it's not possible to choose/create such a device id, then you should notify the user about the importance of the device id.

2.2. HTTP POST request

2.2.1. URL

https://rift.trionworlds.com/external/create-device-key

2.2.2. POST variables

Name	Description
deviceId	The device id TRION has to generate the secret key for.

2.2.3. Result

The request returns a XML file with one or more of the following fields (in XPath notion):

XPath	Description
/DeviceKey/DeviceId	The same device id as created by the client.
/DeviceKey/SerialKey	The authenticator serial number requested by Trion when you enable using the Authenticator in your Rift profile.
/DeviceKey/SecretKey	This is the secret key itself. It’s not encrypted.
/DeviceKey/ErrorCode	This is an error id. This field is missing when everything is OK.

3. Time sync

3.1. HTTP GET request

3.1.1. URL

https://auth.trionworlds.com/time

3.1.2. Result

This returns the time (just the number, no fancy XML stuff). The time offset is calculated as (server_time – client_time). Both numbers are milliseconds starting from 01. Jan. 1970.

4. Login token creation

For the login token creation, we need both the secret key (see Secret key creation) and the time offset (see Time sync).

Every login token is valid for 30 seconds. This time span is denoted by an interval number. This interval number will be used to calculate the login token.

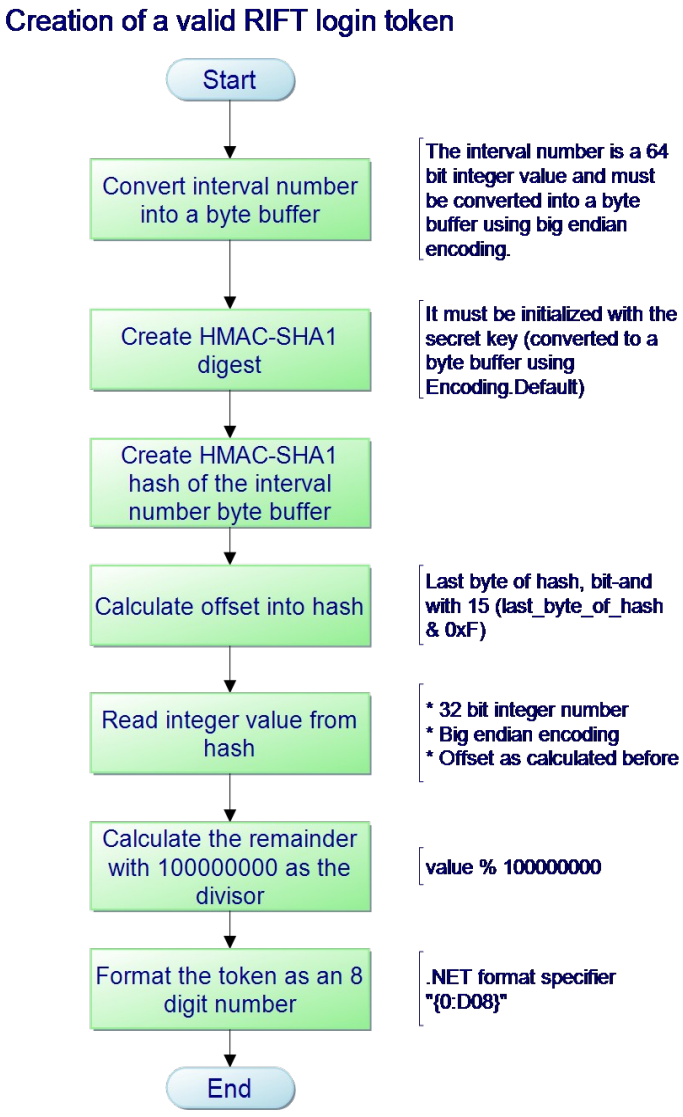
4.1. Interval number

It's calculated using the following formula:

```
(current_time_millis + time_offset) / 30000
```

This also implies, that every interval starts at HH:MM:00 or HH:MM:30 with HH=Hours, MM=Minutes.

4.2. Creation of a valid login token



4.3. Additional information

A useful tool for the big endian conversion to and from byte buffers is Mono.DataConvert.

5. Retrieve secret questions for a RIFT account

Retrieving the secret questions for a RIFT account is required to recover the secret key (Recover the secret key for the authenticator) using a device id.

5.1. HTTP POST request

https://rift.trionworlds.com/external/get-account-security-questions.action

5.2. POST variables

Name	Description
emailAddress	The Email address you use as login user name for Rift.
password	The password for the Rift login.

5.3. Result

The response is a XML file with the following XML elements (given in XPath notion):

XPath	Description
/SecurityQuestions/EmailAddress	The same Email address as given in the URL.
/SecurityQuestions/FirstQuestion	The first security question.
/SecurityQuestions/SecondQuestion	The second security question.
/SecurityQuestions/ErrorCode	An error ID if something failed.

6. Recover the secret key for the authenticator

When you know your device id and your login + answers to the secret questions, then you can recover the secret key. This feature was added to the RIFT Authenticator for Android Version 1.0.3.

6.1. HTTP POST request

https://rift.trionworlds.com/external/retrieve-device-key.action

6.2. POST variables

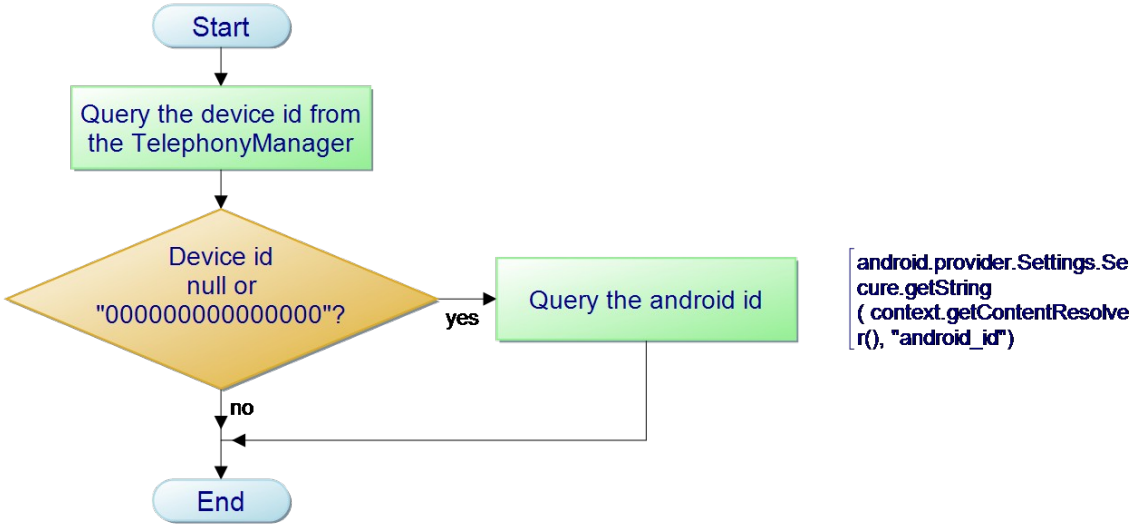
Name	Description
emailAddress	The Email address you use as login user name for Rift.
password	The password for the Rift login.
deviceId	The device id that you used to create the secret key.
securityAnswer	The first security answer (leave empty if none required).
secondSecurityAnswer	The second security answer(leave empty if none required).

6.3. Result

The result is a XML file in the same format as documented in the chapter „Secret key creation“.

7. Device id for Android phones

Determine device id for Android phones



8. Decrypting the secret key of the RIFT Authenticator for Android

8.1. Loading the configuration file

On my phone, the configuration file is stored in the following location:

`/data/data/com.trionworlds.mobile.auth/shared_prefs/system.xml`

8.1.1. „Access denied“ problem

The access to the shared preferences of a different application isn't possible by default. You have to use „su“ on a rooted android phone to access this file.

8.1.2. File format

The preferences file is a XML file with the following contents (in XPath notion):

XPath	Value element	Description
/map/string[@name='device_id']	text()	The Device id for Android phones.
/map/string[@name='secret_key']	text()	The encrypted secret key.
/map/string[@name='serial_key']	text()	The serial key used for registering an authenticator for a RIFT account.
/map/long[@name='time_offset']	@value	The time offset between client and server. Milliseconds since 1970-01-01.

8.2. Decryption of the secret key

- Create a SHA1 random number generator.
- Initialize it with the „TrionMasterKey_031611“ as seed (after converting it into a byte array).
- Create the key for the AES 128 cipher (16 bytes).
- HEX-decode the encrypted secret key.
- Decrypt the HEX-decoded encrypted secret key.

8.3. SHA1 random number generator

Android uses a bug-fixed SHA1 random number generator from the Apache Harmony project. I converted it to C# to have exactly the same behaviour as on my Android phone.

9. Error codes

Code	Description
device_id_missing	Device id variable missing in request.
account_not_available	Invalid user name or password.
account_missing	No RIFT account or device id doesn't match.
account_securityAnswers_incorrect	Security answer(s) incorrect.

10. Legal information