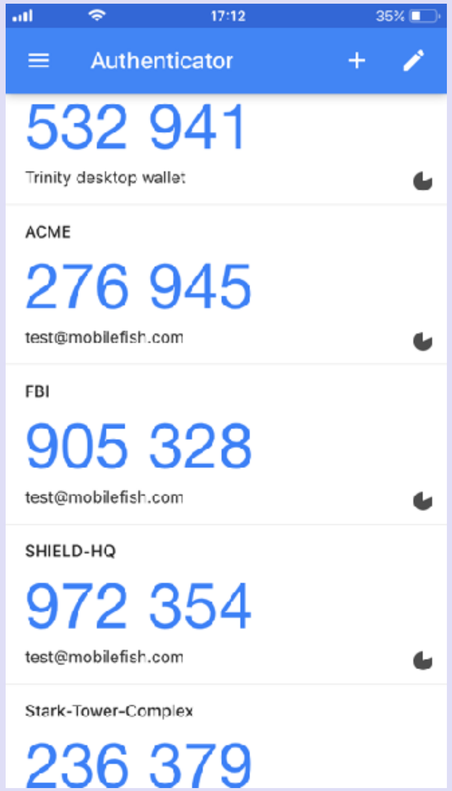
IOTA TUTORIAL 34

Time-based One-Time Password (TOTP)



INTRO

- In this tutorial I will explain in detail how the Time-based One-time Password algorithm works.

- This tutorial is not specific IOTA related. It is intended for developers who wants to understand how the Time-based One-time Password algorithm works.

1-2-3 FACTOR AUTHENTICATION

- An application can implement several methods for the user to authenticate itself.
  - Method A: What the user know - Using username/password.
  - Method B: What the user has - A device to generate a one time password (token).
  - Method C: What the user is - Using fingerprints or face recognition.
- In an 1 Factor Authentication (1FA), an application often implements method A.
- In a 2 Factor Authentication (2FA), an application often implements method A and a second method B or C.

1-2-3 FACTOR AUTHENTICATION

- In a 3 Factor Authentication (3FA), an application often implements method A, B and C.

TRINITY WALLET & 2 FACTOR AUTHENTICATION

- The Trinity desktop and mobile wallet allows you to set 2 Factor Authentication (2FA). This will add an additional security layer to prevent unauthorised access to the wallet.

- There are many 2FA apps which you can install on a mobile device such as Google Authenticator and Authy.

- Google Authenticator and Authy both implements the same Time-based One-time Password (TOTP) algorithm.

TOTP AND HOTP

- The Time-based One-time Password algorithm generates single use passwords, also known as tokens, which are only valid for a certain time period.
  Often this time period is 60 seconds.

- These generated tokens are based on a shared secret key.

- The Time-Based One-Time Password algorithm was published as RFC 6238 by the Internet Engineering Task Force (IETF).
  See: https://tools.ietf.org/html/rfc6238

- In RFC 6238 a Java reference Time-based One-time Password algorithm can be found.
  See also: https://www.mobilefish.com/download/java/TOTP.java

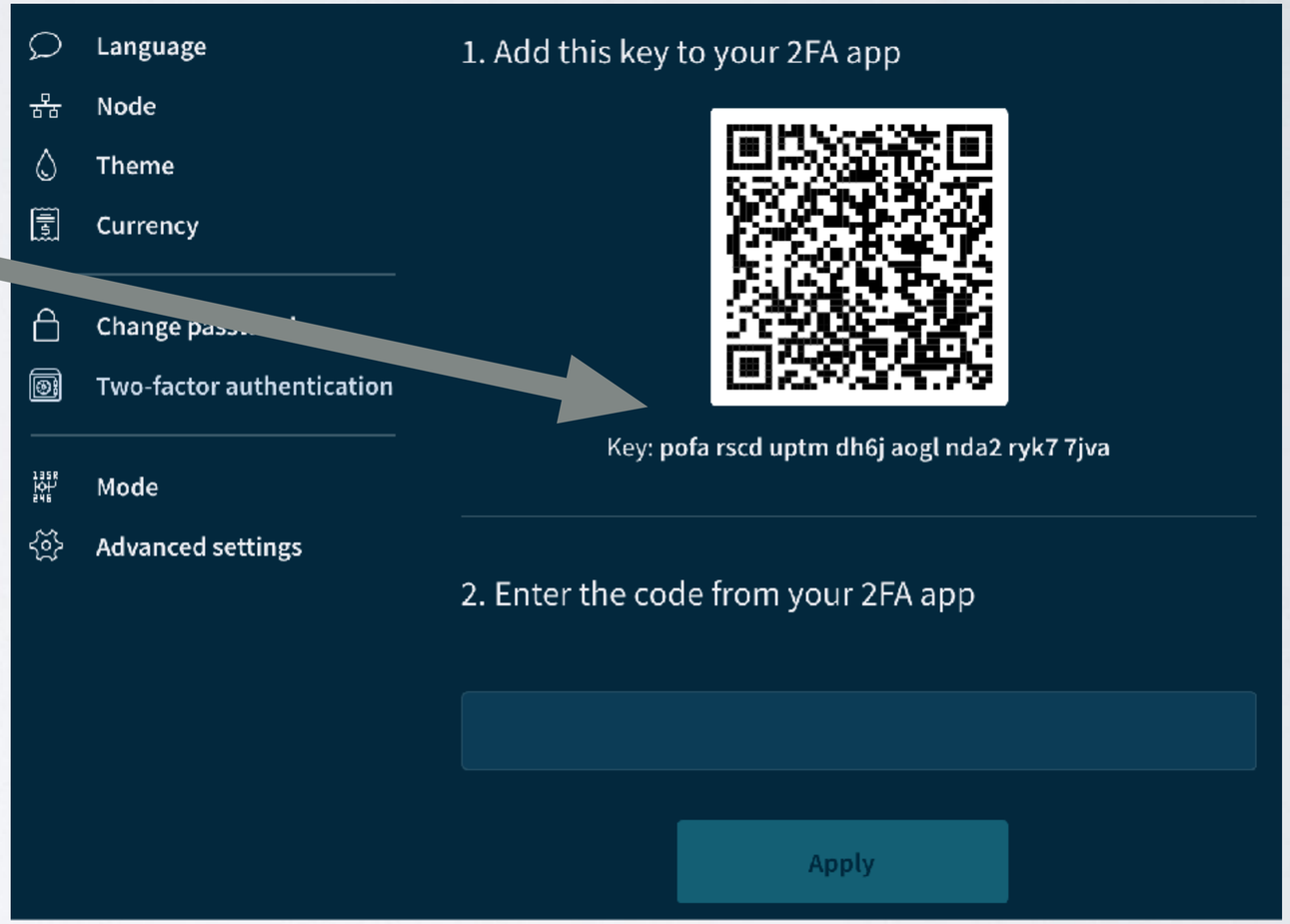
TOTP AND HOTP

- The Time-based One-time Password algorithm is an extension of the HMAC-Based One-Time Password (HOTP) algorithm, which was published as RFC 4226 by the IETF.

- The HMAC-Based One-Time Password defines an algorithm to create an one time password from a shared secret key and a counter.
  See: https://tools.ietf.org/html/rfc4226

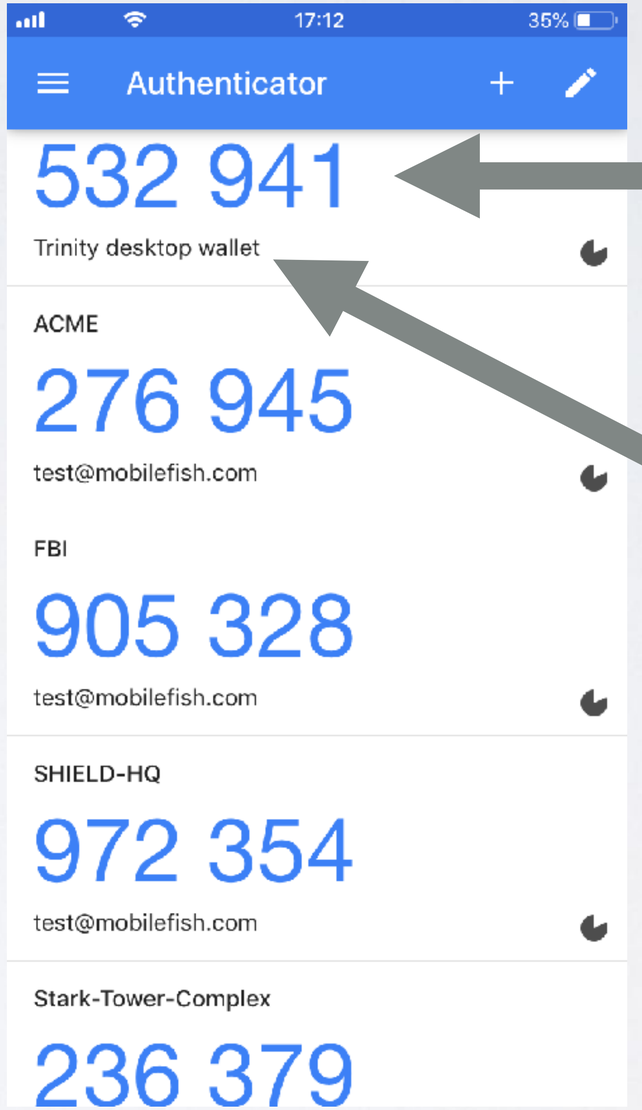
TRINITY WALLET ENABLE 2FA

- When 2FA is enabled on the Trinity wallet, it first generates a shared secret key.
- You must write down this shared secret key and safely store it.
- Usually this shared secret key with additional information is embedded in a QR code which you can scan by a 2FA app such as the Google Authenticator.



TRINITY WALLET ENABLE 2FA

- Lets assume you have installed the Google Authenticator on a mobile.
- After the QR code is scanned the Google Authenticator generates a token which is a unique code, based on the shared secret key and the current time.
- The question is, how is this token generated?



generated token

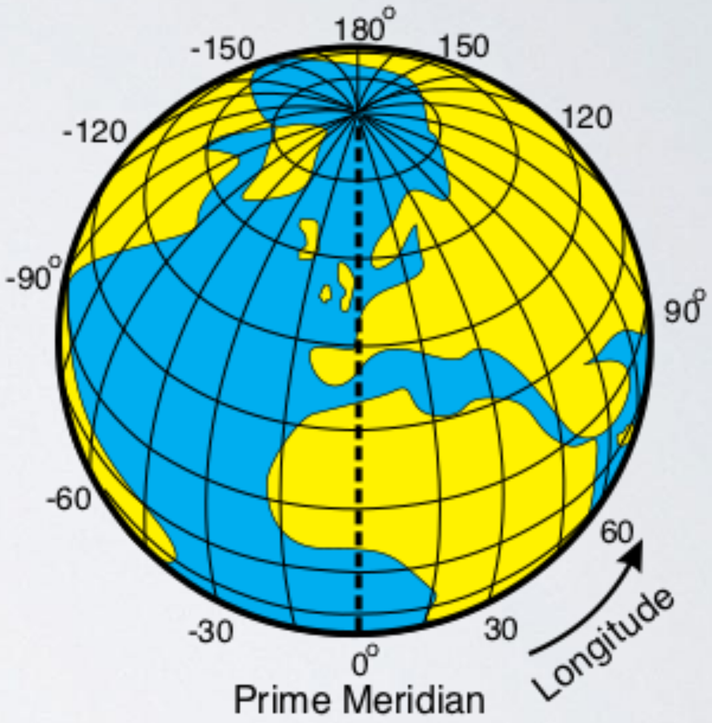
additional information

HOW TOTP WORKS

- I will now explain in detail how this token is generated on a mobile device.

- Lets assume you are currently in Beijing (China).
  The local Beijing date and time is 4 December 2018, 20:24:20 (UTC+8).
  The date and time at that moment at 0° longitude meridian is 4 December 2018, 12:24:20.

- UTC stands for Coordinated Universal Time and is the time at the 0° longitude meridian (Prime Meridian).



HOW TOTP WORKS

- Convert this date and time (4 December 2018, 12:24:20) to Unix Epoch Time. Instead of Unix Epoch Time we can also say Unix Time (T_{unix}).

- Unix Epoch Time is the number of seconds that have elapsed since, 1 January 1970 00:00:00 UTC, not counting leap seconds.

- If the date and time at 0° longitude meridian is 4 December 2018, 12:24:20 than $T_{unix} = 1543926260$ sec

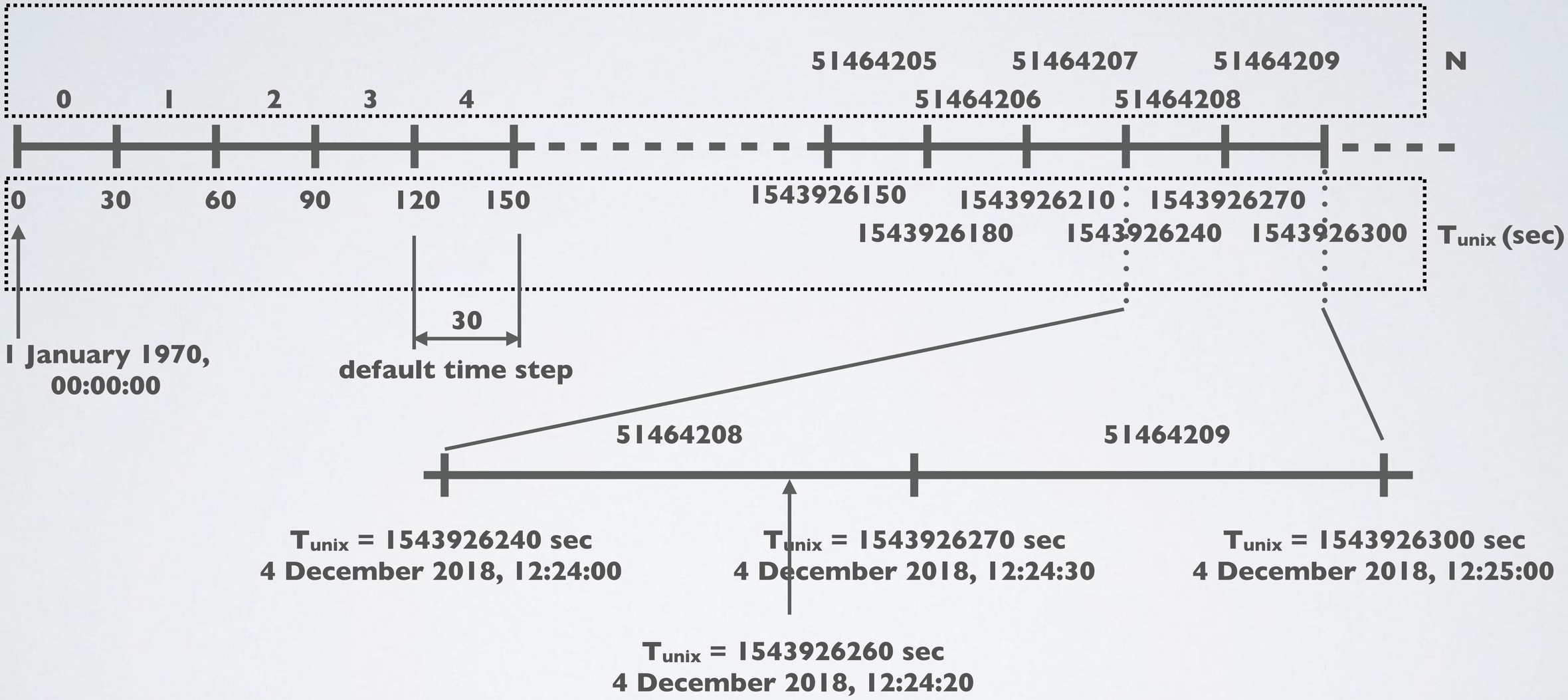
HOW TOTP WORKS

- Equation: **$N = floor(T_{unix} / ts)$**

  N = number of time steps which have been elapsed since Unix Epoch Time.
  floor = function which rounds a number downward to its nearest integer.
  T_{unix} = number of seconds that have elapsed since, 1 January 1970 00:00:00 UTC, not counting leap seconds.
  ts = time step. By default the time step is 30 sec.

- T_{unix} = 1543926260 sec
  N = floor(1543926260 / 30)
  N = 51464208

HOW TOTP WORKS



HOW TOTP WORKS

- Convert the number of time steps (N) into a hexadecimal value.
  The hexadecimal value must have 16 hexadecimal characters (=8 bytes).
  If not, prepend with 0's.
  N_{dec} = 51464208
  N_{hex} = 0x00000000031148l0

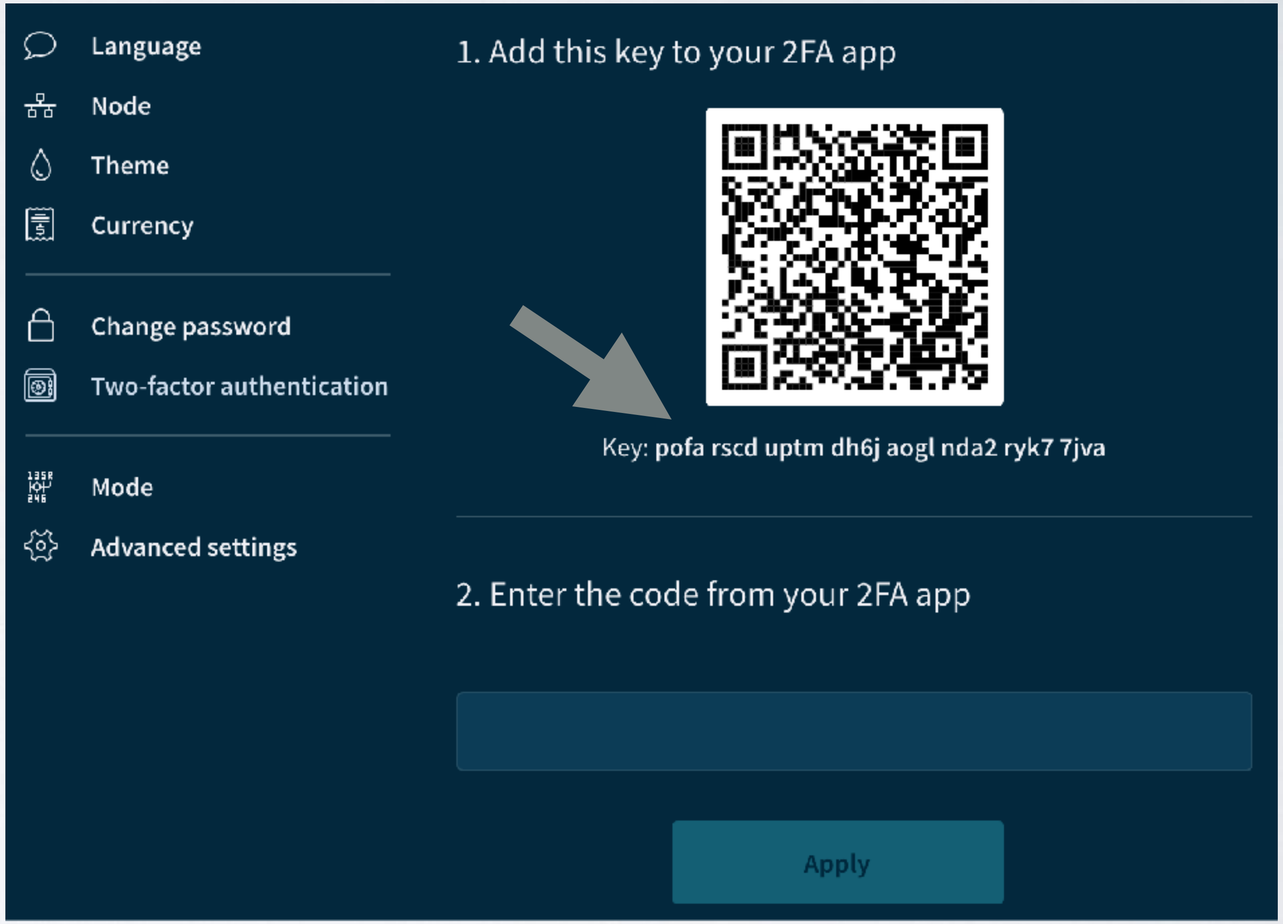
- Convert the hexadecimal value (= 0x00000000031148l0) into a 8 bytes array and assign this value to variable **m** (=message).

HOW TOTP WORKS

- Convert the shared secret key (= pofa rscd uptm dh6j aogl nda2 ryk7 7jva) into a 20 bytes array and assign this value to variable **K**.

- The shared secret key is a randomly generated 20 bytes number which is base-32 encoded. For readability this key is divided in groups of 4 characters and all in lower case.

- More information about base-32, see Blockchain tutorial 31: https://youtu.be/Va8FLD-iuTg



HOW TOTP WORKS

- Calculate the HMAC hash using the HMAC-SHA1 algorithm.

- More information about HMAC, see Blockchain tutorial 30: https://youtu.be/emBgrRIyyWQ

- This HMAC hash size is 160 bits (=20 bytes).
  Lets assume the HMAC value is:

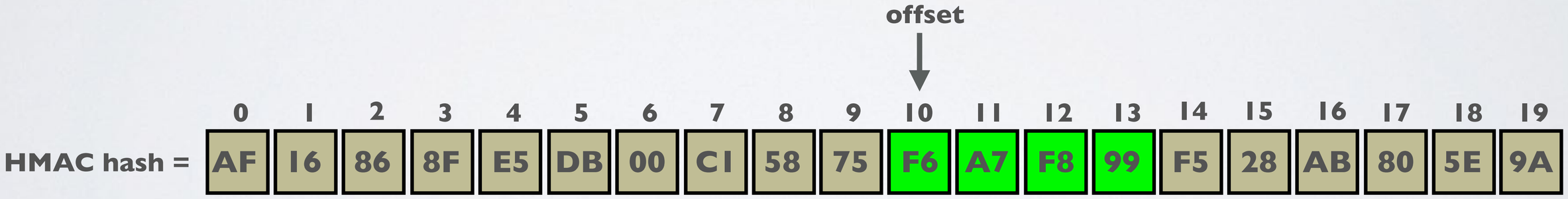
HMAC hash =	AF	16	86	8F	E5	DB	00	C1	58	75	F6	A7	F8	99	F5	28	AB	80	5E	9A

HOW TOTP WORKS

- Get the last 4 bits of this hash value and get its integer value.
  In this example, the last 4 bits is 0xA which represents integer 10.

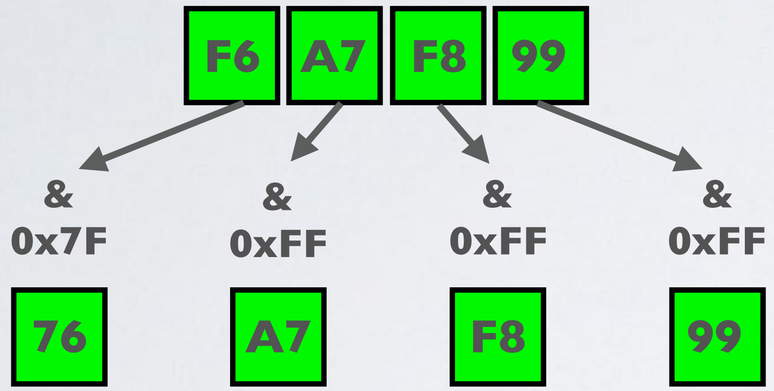
This integer is called the offset.

- Starting from the offset, get the first 4 bytes from the HMAC hash.



HOW TOTP WORKS

- Apply a binary operation for each byte.



- The new binary value is: **0x76A7F899**

- Convert this binary value to an integer: **1990719641**

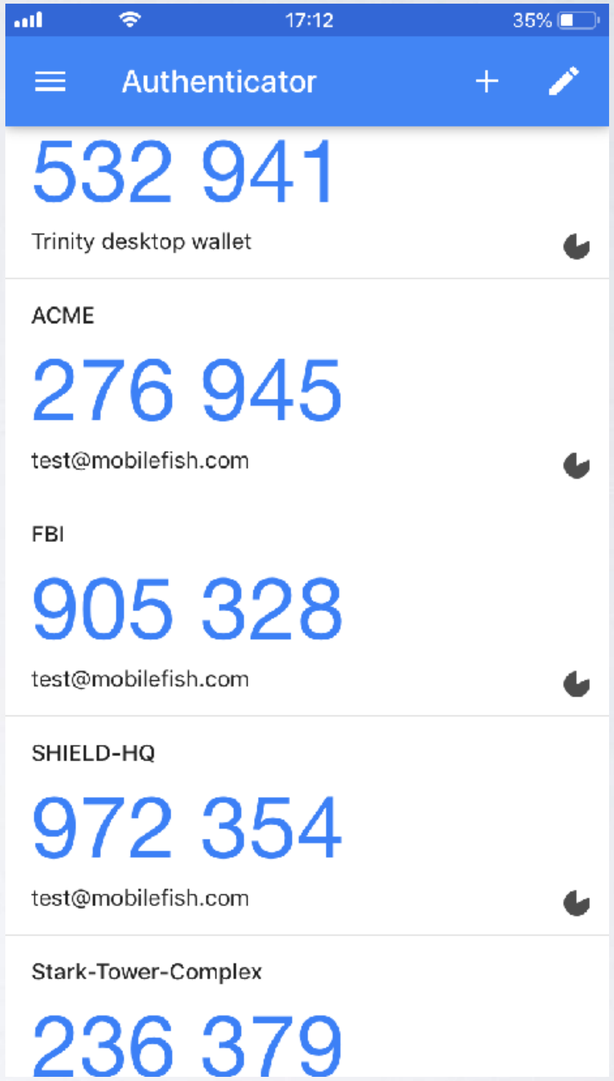
HOW TOTP WORKS

- Calculate the Token = **1990719641 % 10^n**

  where n is the token size.
  In this example n = 6

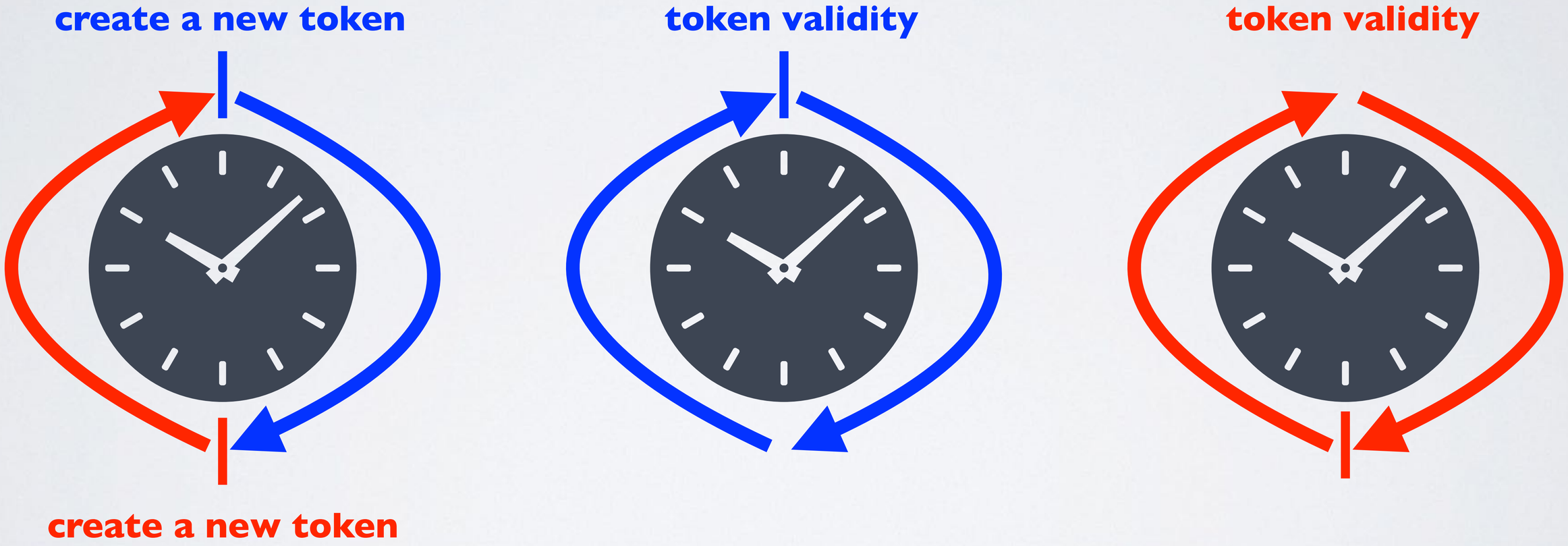
  Token = **1990719641 % 10^6 = 719641**

  If the token size < n, prefix with 0's.



HOW TOTP WORKS

- Every 30 seconds a new token is generated.
  But a token remains valid for 60 seconds.



LESSONS LEARNED

- If you have installed the Trinity Desktop wallet and have enabled 2FA, make sure your computer has the correct date and time set. The same applies to the mobile device where the 2FA app is installed, for example Google Authenticator.

- Instead of scanning the QR code you can manually enter the shared secret key in the 2FA app, but it is not recommended because you might make typing mistakes.

- An example of a shared secret key, generated by the Trinity wallet:
  pofa rscd uptm dh6j aogl nda2 ryk7 7jva
  If you manually enter the shared secret key in a 2FA app, you can enter upper or lower case letters with or without spaces. A 2FA app will convert all entered characters to upper case and will remove all spaces. The actual shared secret key only contains the letters A-Z and the digits 2-7, because it is base-32 encoded.

ONLINE TIME-BASED ONE-TIME PASSWORD GENERATOR

- An online Time-based One-time Password generator can be found at:
  https://www.mobilefish.com/services/cryptocurrency/totp.html
  It behaves the same as the Google Authenticator or Authy.

- This application also demonstrates how the token is validated.

- **WARNING: DO NOT USE THIS TOOL TO GENERATE YOUR TOKENS. IT IS ONLY INTENDED FOR EDUCATIONAL, TEST OR DEMONSTRATION PURPOSES.**