Klas Lindfors

One time passwords

#

About me

- Klas Lindfors
- Software Developer at Yubico (http://www.yubico.com)
  - Swedish startup with presence in Sweden, UK and US
  - Builds the YubiKey, showing itself as a keyboard
  - Big name customers: Google, Facebook...

#

Content

- What is there?
- How does it work?
- What is the future?

#

What is there?

- HOTP
- TOTP
- Yubico OTP
- ...

#

HOTP

> "HMAC-based One-time Password Algorithm"

- Standardized by Initiative for open authentication, later as RFC-4226
- 6 or 8 digits
- Event based
- Counter synchronization

#

HOTP Details

```
HOTP(K,C) = Truncate(HMAC(K,C)) & 0x7fffffff
```

- HMAC-SHA1 of 8-byte counter and key
- Truncation is by taking last 4 bits as index and getting 31 bits
- Modulo digits ** 10

#

HOTP Example

- Key "kaka" (0x6b616b61)
- Counter 1 (0x0000000000000001)
- HMAC-SHA1(K, C) = 0x5e3a6c99797fd **3ed5eb2** cff4ae21061aba748d9 **6**
- = 1408065202 % 1000000 = 065202

#

HOTP Validation

- Decide on a window of acceptable counters
    - A common window is 3
- Start with stored counter + 1 and generate up to counter + 1 + window codes
    - Has to be done for each token for the user
- Compare codes one by one until one match, store the value for that
- Requires for synchronized storage of current counter
- Requires a reset for when/if the counter is out of sync
    - Could become support intensive

#

TOTP

> "Time-based One-time Password Algorithm"

- Extension of the HMAC-based One Time Password algorithm HOTP to support a time based moving factor.
- RFC-6238
- Time based
- Time synchronization

#

TOTP Details

```
TOTP(K) = Truncate(HMAC(K, unixtime(now) / 30)) & 0x7fffffff
```

- Same algorithm as HOTP
- Time instead of counter

#

TOTP Example

- Key "kaka" (0x6b616b61)
- Time 2014-02-18 17:30:00 UTC (1392744600)
    - 1392744600 / 30 = 0x2C462F4
- HMAC-SHA1(K, C) = 0x8 **238609b** 904009d8a6f2ebcf6ce98d434ab0fec **0**
- = 37249179 % 1000000 = 249179

#

TOTP Validation

- Decide on a window of acceptable time-slide
    - If using hard tokens, might have to track time drift individually
- No need to track counters centrally

#

Yubico OTP

- Event based algorithm using AES encryption
- Not a standard
- 44 characters (12 + 32)
- id and OTP in one
- Session and use counter

#

Yubico OTP Details

- Private ID (6 bytes)
- Two counters (3 bytes)
- Timestamp since powerup (3 bytes)
- Random (2 bytes)
- CRC (2 bytes)
- Encrypted with AES-ECB
- Modhex is used

```
0123456789abcdef
cbdefghijklnrtuv
```

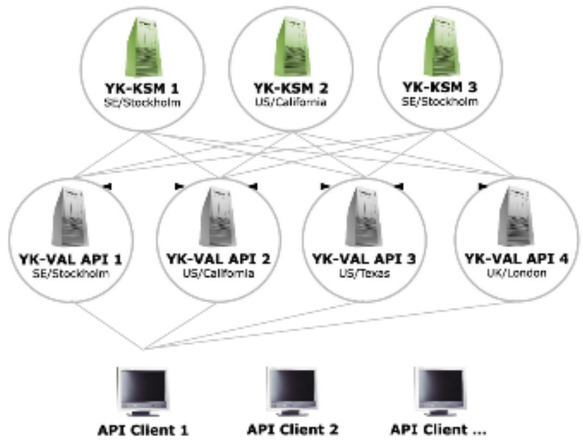
#

Yubico OTP Validation

- Use public-id to find the correct key
- Decrypt the 16 byte OTP with the key
- Verify CRC
- Compare private-id
- Compare session and use counter
    - Only looking at the use counter if the session is the same
- Optionally look at the timer, but it might wrap

Yubico OTP Validation (cont.)

- In the cloud with YubiCloud
    - Clients in most languages (PHP, Python, Ruby)
- Opensource packages to run yourself
- Implement the validation from scratch



#

OTP Comparisons

- OATH
  - + Standard
  - + Simple algorithm
  - - No identification of the token (unless OATH Token Identifier is used)
  - - Low entropy
  - - No way to know if software or hardware
  - - Bruteforce
- HOTP
  - + Simple hardware tokens exist
  - - Syncronized database of counters
- TOTP
  - + No database of counters
  - + Valid for a limited time
  - - More complex hardware with time

#

OTP Comparisons (cont.)

- Yubico OTP
  - + Comes with identification
  - + Longer OTP that is harder to bruteforce
  - + Might be guaranteed hardware
  - - Not a standard
  - - Syncronized database for the counters

#

Validation in the cloud

- For example YubiCloud or Symantec VIP
- One token can be used for several sites
- One phished token in one setting can be used for others
- With soft multi-credential authenticators the move seem to be to site-local validation
- Redundancy and uptime is critical
- Seed storage

#

What is the future?

- FIDO Alliance U2F
    - Universal Second Factor
- OATH OCRA
    - OATH Challenge-Response Algorithm

#

U2F

- Draft standards published
    - http://fidoalliance.org/specifications/download
- Assymetric cryptography (ECC NIST-P256)
- Unique Key Pair per service
- Contains attestation to ensure a hard authenticator
- Some protection for man-in-the-middle

fido alliance simpler stronger authentication

#

U2F Details

- Two primitives, Sign and Enroll
- Enroll creates a unique Key Pair, might let the service store it
- Sign takes challenge and site id, signs with ECDSA
- Browser centered, Javascript API
- Nothing secret in the server
- Presence

#

OCRA

> "OATH Challenge-Response Algorithm"

- RFC-6287
- Challenge-response based OATH
- Same algorithm as HOTP and TOTP
    - Though usually used with SHA-256 or SHA-512
- Starting to get some traction

#

Questions

#

Thank You

- Klas Lindfors <klas@yubico.com>