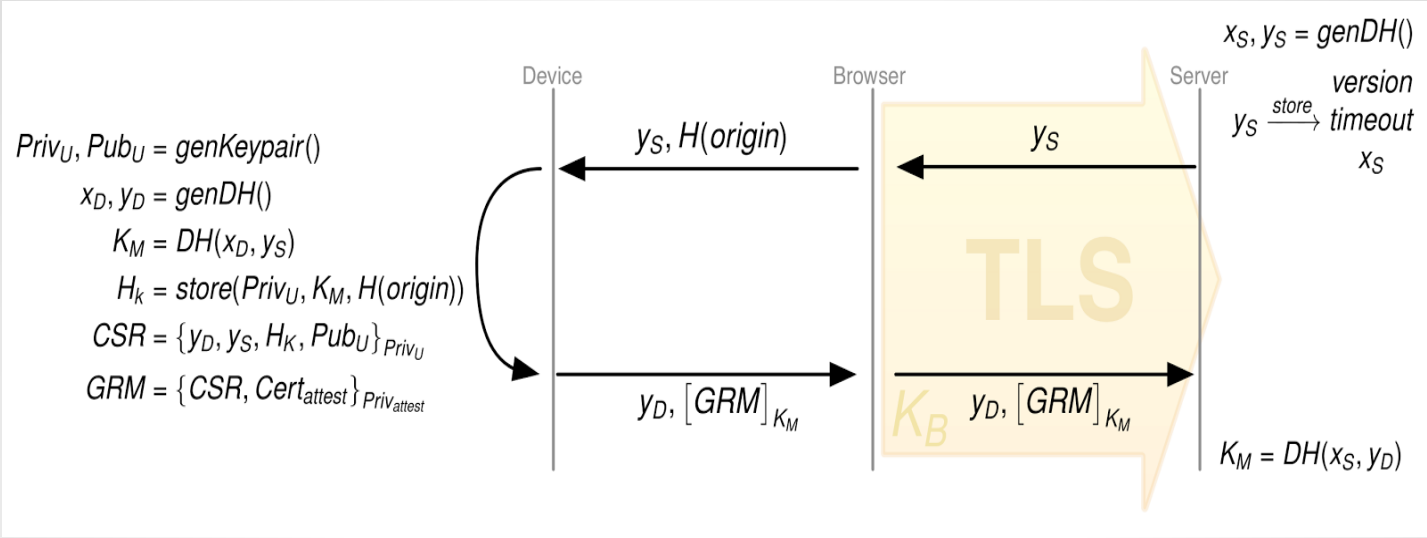
Enrollment Workflow

1. User goes to U2F registration page of target website
2. Taps in username and password
3. Website shows page with “Register” button
4. When user clicks, a U2F javascript “get public key” function is called
5. Browser implements javascript call - it discovers the attached U2F device and sends the “get public key” request to it
6. U2F generates key pair after touch, returns public key to browser, retains private key
7. Browser gives public key to website which associates it with the user.

Verification/Login Workflow

1. User goes to login page of target website and logs in with username and password
2. Website sends javascript call with user's public key (in fact key handle) and some random data to browser.
3. User sees an "Please present U2F" request
4. User 'attaches' U2F to computer and presses the activation button
5. Browser interprets javascript call:
   a. It looks up the origin server of the calling web page and (optionally) a SSL connection identifier (ChannelID) if available
   b. It concatenates this to the random data the server sent and sends is to the U2F along with the user's public key
   c. U2F signs the data with the private key and returns it.
6. Browser sends signed data to the server
7. Server verifies that the signature indeed verifies against the public key thus proving that the U2F had the corresponding private key.

Message Flow during Registration



$Priv_u$ Pub_u - U2F-device-generated private-public key pair

x, y - private and public key for Diffie-Hellman key exchange (point on NIST P256 elliptic curve)

K_m - generated shared key used for encrypting GRM

H_k - hashed key handle

CSR - Certificate Signing Request to encode y_D, H_k and Pub_u

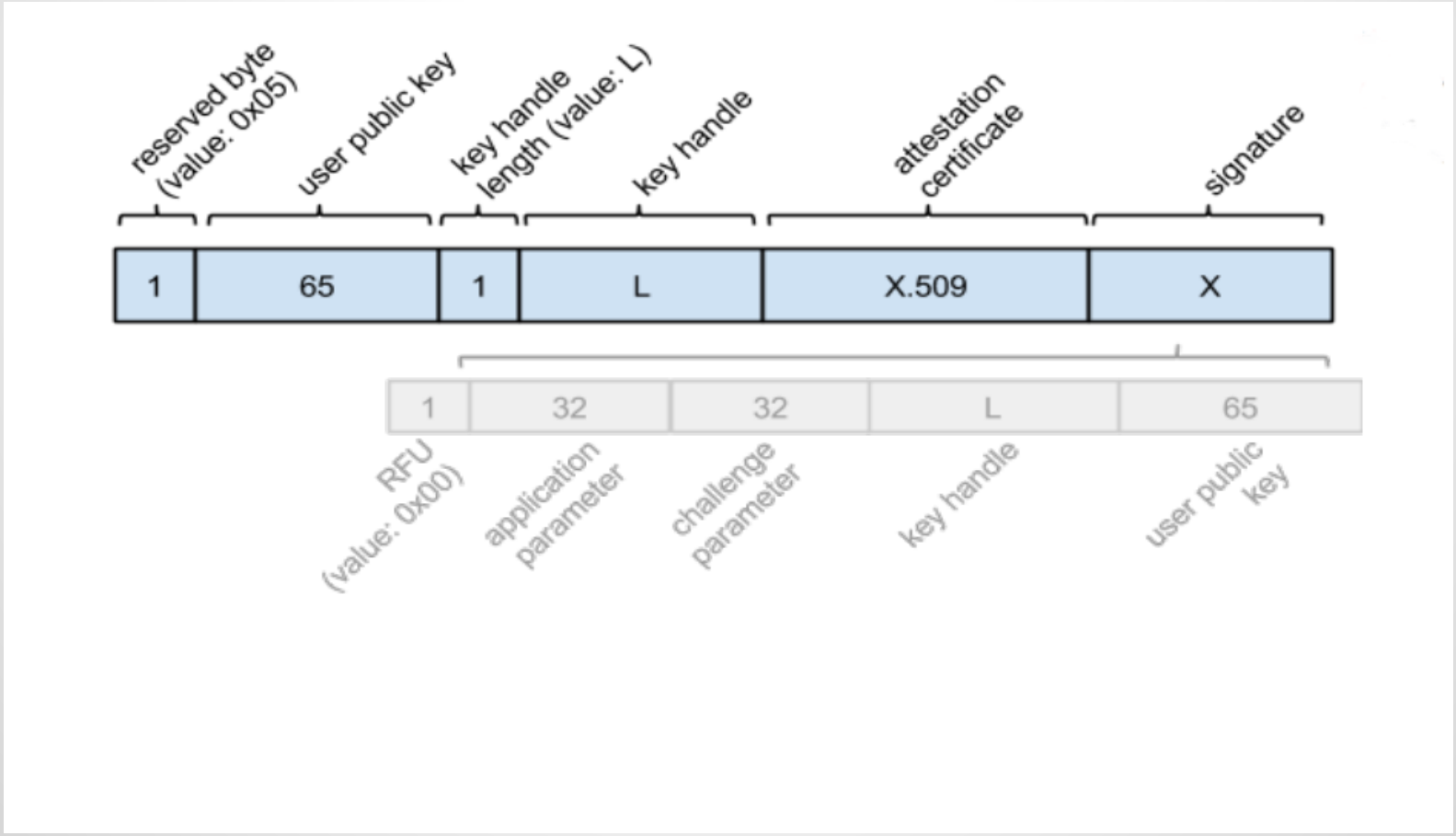
GRM - Device Registration Message, sent back to browser

$Cert_{attest}$ - guarantee that key in CSR was generated in a secure environment (verified outside of protocol)

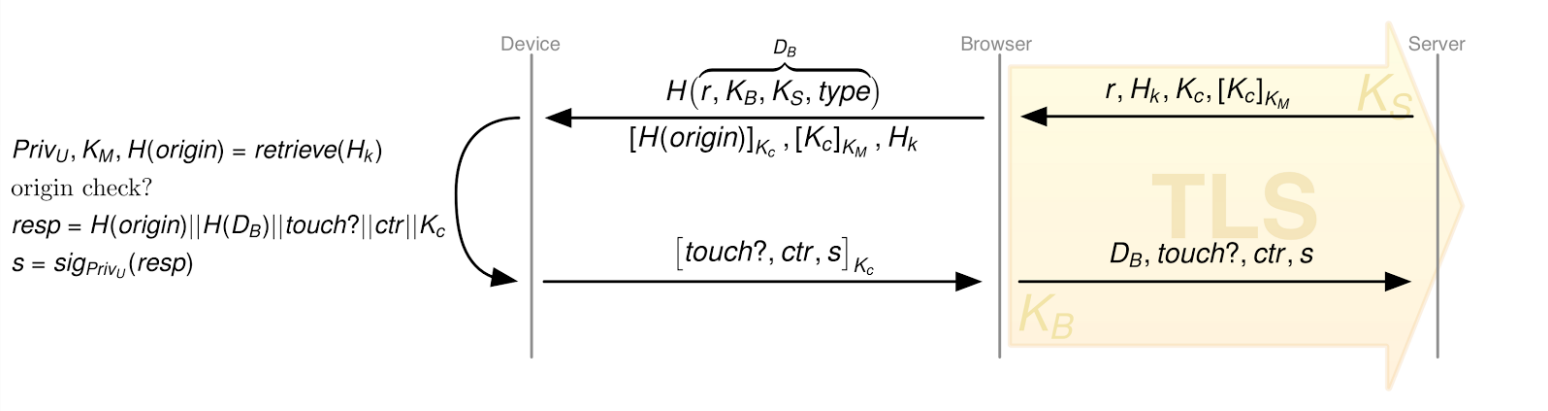
- the server can decrypt the GRM, save the H_k and Pub_u and verify origin, DH keys, $Cert_{attest,}$ etc.

Registration Response Message

- issued by U2F device



Message Flow during Login



H_k - key handle

K_C - channel protection key

r - random challenge

D_B - browser data

ctr - counter

- U2F device retrieves $H(D_B)$, $[K_C]_{Km}$, H_k and $[H(origin)]_{Kc}$
- decrypts H_k, decrypts K_C and H(Origin)
- performs an origin check
- if correct, it send resp, otherwise error code

- the server verifies the signature and that the origin matches its own origin

Private and public keys

- Multiple solutions are possible:
  - Gnubby can store one or many private keys for every site
    - hardware costs for memory of thousands of sites or origins
  - Gnubby can store exactly one private key
    - all origins have the same public key -> bad
  - Key export
    - sites and origins receives and store a public key and a blob
    - the blob is the private key which is encrypted
    - the key handle will also include the blob for the Gnubby

Prevention of security risks

- Man-In-The-Middle-Attack
  - Origin name must match with key handle
  - Browser-Data will be hashed
  - Will not protect an enrollment with a MITM
- Counters as a signal for detecting cloned U2F devices
  - Device and origin save a counter for each operation of key-pair
- An origin can discover that two accounts use the same U2F device
  - Usage of multiple devices
- Revoking a key from an origin
  - physical destruction of the secure element

Problems

- The chrome extension did not want to work with google’s example code
  - problem(s):
    - no knowledge about chrome extensions and javascript
    - the extension has a tld-check which does not allow localhost
    - solution: a simple check for localhost
- The demo for the server application only works with Google App Engine
  - for a separate Apache, Tomcat, JBoss, etc. module, more time is needed

Sources

Google Presentation: https://docs.google.com/presentation/d/16mB3Nptab1i4-IlFbn6vfkWYk-ozN6j3-fr7JL8XVyA/edit?pli=1#slide=id.g19c09a112_2_0

FIDO U2F Raw Message Formats:

http://fidoalliance.org/specs/fido-u2f-raw-message-formats-v1.0-rd-20140209.pdf

U2F Protocol and API Details:

https://docs.google.com/document/d/1Jm_xAJTZGulMOkYOQm-fIQhhkd2VDr9578oh0KOwcEw/edit#heading=h.q5kqrl82hzpj

U2F: Product Overview: Easy Strong Auth for the web

https://docs.google.com/document/d/1SjCwdrFbVPG1tYavO5RsSD1QpJwj8_im6sl-VWjJ6k0/edit#

U2F: Protocol Design + User Flows:

https://docs.google.com/document/d/12AdwNDIhs6blXGTCOReaUGviBqCtsVrGMtrxGeCCxPU/edit#