

Agents

- A* - Alice
- M* - Webauthn authenticator (passkey manager)
- B* - Browser with SubtleCrypto API and libsodium library
- S* - Quick Crypt server
- R* - <https://www.random.org/cgi-bin/randbyte?nbytes=48>
- D* - Insecure persistent storage system

Browser and Libsodium Functions

- E_a* - Symmetric cipher using AEAD algorithm *a*. One of:
 1. AES-256 in Galois/Counter mode from SubtleCrypto
 2. XChaCha20 with Poly1305 MAC from libsodium
 3. AEGIS 256 from libsodium
- D_H* - HKDF key derivation using SHA-512 FIPS-180-4 from SubtleCrypto
- D_P* - PBKDF2 key derivation using SHA-512 FIPS-180-4 from SubtleCrypto
- H* - BLAKE2b keyed hash (MAC) generator from libsodium
- V* - BLAKE2b keyed hash (MAC) validator from libsodium
- G* - Cryptographic pseudorandom generator from SubtleCrypto

Cipher Variables

- N* - Block number
- m* - Clear text message
- m₀* - Clear text block 0
- m_N* - Clear text block N
- m_E* - Block of encrypted message
- p* - Password text
- h* - Password hint text
- u_c* - 256 bit user credential
- a* - Symmetric AEAD cipher and mode: [1, 2, 3]
- i* - PBKDF2 iteration count, minimum 400,000
- k_M* - 256 bit message cipher key
- k_H* - 256 bit hint cipher key
- k_S* - 256 bit MAC key
- kp_S* - Key purpose text: “cipherdata signing key”
- kp_H* - Key purpose text: “hint encryption key”
- r* - 384 bits of either true or pseudo random data
- n_{IV}* - True or pseudo random initialization vector
- n_{IV}l* - *n_{IV}* bit length: [96, 192, 256]
- n_S* - 128 bit true or pseudo random salt
- lp* - Loop count (0-15)
- le* - Loop end (0-15)
- ad* - Additional data
- v* - Cipher data version
- h_E* - Encrypted hint
- h_El* - Encrypted hint length
- t* - 256 bit MAC tag
- l* - Payload length
- b* - Valid or invalid MAC tag
- cd* - Cipher data
- cd₀* - Cipher data block 0
- cd_N* - Cipher data block N
- err* - Error message and exit

Message Encryption by A

$A \xleftrightarrow{\text{webauthn}} B, M \xleftrightarrow{\text{webauthn}} S$
 $B \leftarrow S : u_c$
 $A \rightarrow B : m, i, le$
 $v = 4$
 $lp = 0$
LOOP : B compute
 $A \rightarrow B : p, h, a$
 $r = G(384) \vee B \xleftarrow{\text{https}} R : r$
 $n_S = r[0 : 128]$
 $n_{IV} = r[128 : 128 + n_{IV}l]$
 $k_M = D_P(p \parallel u_c, n_S, i)$
 $k_S = D_H(u_c, n_S, kp_S)$
 $cd = cd_0 \parallel cd_N \parallel \dots$
 $m = cd$
 $lp = lp + 1$
 goto *LOOP* if $lp < le$
 $A \leftarrow B : cd$

Block 0 Encryption by B

$k_H = D_H(u_c, n_S, kp_H)$
 $h_E = E_a(h, n_{IV}, k_H)$
 $h_El = len(h_E)$
 $ad = a \parallel n_{IV} \parallel n_S \parallel i \parallel le \parallel lp \parallel h_El \parallel h_E$
 $m_E = E_a(m_0, n_{IV}, ad, k_M)$
 $l = len(ad \parallel m_E)$
 $t = H(v, l, ad, m_E, k_S)$
 $cd_0 = t \parallel v \parallel l \parallel ad \parallel m_E$

Block N Encryption by B

$r = G(384) \vee B \xleftarrow{\text{https}} R : r$
 $n_{IV} = r[0 : n_{IV}l]$
 $ad = a \parallel n_{IV}$
 $m_E = E_a(m_N, n_{IV}, ad, k_M)$
 $l = len(ad \parallel m_E)$
 $t = H(v, l, ad, m_E, k_S)$
 $cd_N = t \parallel v \parallel l \parallel ad \parallel m_E$

Message Storage by A

$A \rightarrow D : cd$

Message Retrieval by A

$A \leftarrow D : cd$

Message Decryption by A

$A \stackrel{\text{webauthn}}{\leftrightarrow} B, M \stackrel{\text{webauthn}}{\leftrightarrow} S$
 $B \leftarrow S : u_c$
 $A \rightarrow B : cd$
 $lp = 0$
LOOP : B compute
 $t, v, l, ad, m_E = cd_0$
 $a, n_{IV}, n_S, i, le, lp, h_{El}, h_E = ad$
 $k_S = D_H(u_c, n_S, kp_S)$
 $b = V(v, l, ad, m_E, k_S, t)$
 if ! b :
 $A \leftarrow B : err$
 $k_H = D_H(u_c, n_S, kp_H)$
 $h = E_a^{-1}(h_E, n_{IV}, k_H)$
 $B \rightarrow A : h$
 $B \leftarrow A : p$
 $k_M = D_P(p \parallel u_c, n_S, i)$
 $m = m_0 \parallel m_N \parallel \dots$
 $cd = m$
 $lp = lp + 1$
 goto *LOOP* if $lp < le$
 $A \leftarrow B : m$

Block 0 Decryption by B

$m_o = E_a^{-1}(m_E, n_{IV}, ad, k_M)$

Block N Decryption by B

$t, v, l, ad, m_E = cd_N$
 $a, n_{IV} = ad$
 $b = V(v, l, ad, m_E, k_S, t)$
if ! b :
 $A \leftarrow B : err$
 $m_N = E_a^{-1}(m_E, n_{IV}, ad, k_M)$

Webauthn Variables

u_n - A's chosen user name
 u_i - 128 bit user id guaranteed to be unique
 u_c - 256 bit user credential
 o - Quick Crypt origin "https://quickcrypt.org"
 ch - 256 bit challenge value
 ro - Registration options, including o, ch, u_i
 rr - Registration response, including signed ch
 ao - Authentication options, including o, ch
 ar - Authentication response, including signed ch
 cw - Alice's webauthn authenticator credentials

Registration by A

$A \rightarrow B : u_n$
 $B \rightarrow S : u_n, o$
 S create and store :
 $u_i = G(128)$
 $u_c = G(256)$
 $ch = G(256)$
 $B \leftarrow S : ro$
 $B \rightarrow M : ro$
 $A \rightarrow M : cw$
 M create and store passkey, sign ch
 $B \leftarrow M : rr$
 $B \rightarrow S : rr, u_i, ch$
 S verify signature, store rr , remove ch
 $B \leftarrow S : u_i, u_n, u_c$
 $A \leftarrow B : u_i, u_c$

Authentication by A

$B \rightarrow S : o[, u_i]$
 S create and store :
 $ch = G(256)$
 $B \leftarrow S : ao$
 $B \rightarrow M : ao$
 $A \rightarrow M : cw$
 M sign ch
 $B \leftarrow M : ar$
 $B \rightarrow S : ar, ch$
 S verify signature, remove ch
 $B \leftarrow S : u_i, u_n, u_c$

Recovery from Lost Passkey by A

$A \rightarrow B : u_i, u_c$
 $B \rightarrow S : u_i, u_c, o$
 S delete existing rr , create and store :
 $ch = G(256)$
 $B \leftarrow S : ro$
 $B \rightarrow M : ro$
 $A \rightarrow M : cw$
 M create and store passkey, sign ch
 $B \leftarrow M : rr$
 $B \rightarrow S : rr, u_i, ch$
 S verify signature, store rr , remove ch
 $B \leftarrow S : u_i, u_n, u_c$