SAKE Protocol Pseudocode

Algorithm 1 SAKE: Inicializacia retazca klucov (Key Chain Initialization)

function SAKE_INIT_KEY_CHAIN($master\_key, is\_initiator$)
    Vstup: $master\_key$ (32B tajomstvo), $is\_initiator$ (boolean)
    Vystup: Inicializovana struktura key_chain
    $key\_chain \leftarrow$ nova struktura pre retazec klucov
    $key\_chain.master\_key \leftarrow master\_key$ ▷ Hlavny kluc
    $key\_chain.epoch \leftarrow 0$ ▷ Zaciatok od epochy 0
    $key\_chain.is\_initiator \leftarrow is\_initiator$
    $key\_chain.auth\_key\_curr \leftarrow$ derive_authentication_key($master\_key$)
    if $is\_initiator$ **then** ▷ Klient pripravuje kluce
        $temp\_master \leftarrow$ copy($master\_key$)
        $temp\_auth \leftarrow$ copy($key\_chain.auth\_key\_curr$)
        evolve_keys($temp\_master, temp\_auth, 1$) ▷ Pre epochu 1
        $key\_chain.auth\_key\_next \leftarrow temp\_auth$
        $key\_chain.auth\_key\_prev \leftarrow key\_chain.auth\_key\_curr$
        secure_wipe($temp\_master$) ▷ Bezpecne vycistenie
    else ▷ Server inicializuje vsetky kluce rovnako
        $key\_chain.auth\_key\_prev \leftarrow key\_chain.auth\_key\_curr$
        $key\_chain.auth\_key\_next \leftarrow key\_chain.auth\_key\_curr$
    end if
    return key_chain
end function
function DERIVE_AUTHENTICATION_KEY($master\_key$)
    Vstup: $master\_key$ (32B kluc)
    Vystup: $auth\_key$ (32B autentifikacny kluc)
    $auth\_key \leftarrow$ BLAKE2b($master\_key \parallel$ "SAKE_K_AUTH", 32) ▷ Hashovanie s oddelenim domeny
    return $auth\_key$
end function

Algorithm 2 SAKE: Proces vzajomnej autentifikacie (Mutual Authentication)

function SAKE_AUTHENTICATE(*key_chain*, *is_initiator*)
  Vstup: *key_chain* struktura, *is_initiator* flag
  Vystup: *session_key* alebo kod chyby
  if *is_initiator* **then** ▷ Klientska strana
    client_nonce ← generate_random_bytes(16)
    Odosli *client_nonce* serveru
    Prijmi *server_nonce* a *challenge* zo servera
    response ← compute_response(*key_chain.auth_key_curr*, *challenge*, *server_nonce*)
    Odosli *response* serveru
  else ▷ Serverova strana
    Prijmi *client_nonce* od klienta
    server_nonce ← generate_random_bytes(16)
    challenge ← generate_challenge(*key_chain.auth_key_curr*, *client_nonce*, *server_nonce*)
    Odosli *server_nonce* a *challenge* klientovi
    Prijmi *client_response* od klienta
    expected_response ← compute_response(*key_chain.auth_key_curr*, *challenge*, *server_nonce*)
    if *client_response* ≠ *expected_response* **then**
      return ERROR_AUTHENTICATION_FAILED ▷ Autentifikacia zlyhala
    end if
  end if
  session_key ← derive_session_key(*key_chain.master_key*, *client_nonce*, *server_nonce*)
  sake_update_key_chain(*key_chain*) ▷ Evolucne kluce
  return *session_key*
end function

Algorithm 3 SAKE: Pomocne funkcie pre proces autentifikacie

function GENERATE_CHALLENGE(*auth_key*, *client_nonce*, *server_nonce*)
  Vstup: Autentifikacny kluc, nonce klienta, nonce servera
  Vystup: Vyzva (challenge) vyzadujuca znalost *auth_key*
  challenge ← BLAKE2b(*auth_key* ‖ *client_nonce* ‖ *server_nonce* ‖ "SAKE_CHALLENGE", 32)
  return *challenge*
end function
function COMPUTE_RESPONSE(*auth_key*, *challenge*, *server_nonce*)
  Vstup: Autentifikacny kluc, vyzva, nonce servera
  Vystup: Odpoved dokazujuca vlastnictvo *auth_key*
  response ← BLAKE2b(*auth_key* ‖ *challenge* ‖ *server_nonce*, 32)
  return *response*
end function
function DERIVE_SESSION_KEY(*master_key*, *client_nonce*, *server_nonce*)
  Vstup: Hlavny kluc, nonce klienta, nonce servera
  Vystup: Relacny kluc pre zabezpecenu komunikaciu
  session_key ← BLAKE2b(*master_key* ‖ *client_nonce* ‖ *server_nonce* ‖ "SAKE_SESSION", 32)
  return *session_key*
end function

Algorithm 4 SAKE: Aktualizacia retazca klucov (Key Chain Update)

function SAKE_UPDATE_KEY_CHAIN(*key_chain*)
    Vstup: Struktura *key_chain* na aktualizaciu
    Vystup: Aktualizovana *key_chain* s vyvinutymi klucmi
    if *key_chain.is_initiator* **then** ▷ Iniciator (klient)
        key_chain.auth_key_prev ← *key_chain.auth_key_curr* ▷ Rotacia klucov
        key_chain.auth_key_curr ← *key_chain.auth_key_next*
        temp_master ← copy(*key_chain.master_key*)
        next_epoch ← *key_chain.epoch* + 1
        temp_master ← BLAKE2b(*key_chain.master_key* ‖ *next_epoch* ‖ "SAKE_K", 32)
        key_chain.auth_key_next ← derive_authentication_key(*temp_master*)
        key_chain.master_key ← BLAKE2b(*key_chain.master_key* ‖ *key_chain.epoch* ‖ "SAKE_K", 32)
        secure_wipe(*temp_master*)
    else ▷ Responder (server)
        key_chain.master_key ← BLAKE2b(*key_chain.master_key* ‖ *key_chain.epoch* ‖ "SAKE_K", 32)
        key_chain.auth_key_curr ← derive_authentication_key(*key_chain.master_key*)
        key_chain.auth_key_prev ← *key_chain.auth_key_curr*
        key_chain.auth_key_next ← *key_chain.auth_key_curr*
    end if
    key_chain.epoch ← *key_chain.epoch* + 1 ▷ Zvysenie epochy
end function