Workgroup: Network Working Group
Internet-Draft: draft-irtf-cfrg-opaque-03
Published: 21 February 2021
Intended Status: Informational
Expires: 25 August 2021
Authors: H. Krawczyk (Algorand Foundation)
K. Lewi (Novi Research)
C.A. Wood (Cloudflare)

The OPAQUE Asymmetric PAKE Protocol

Abstract

This document describes the OPAQUE protocol, a secure asymmetric password-authenticated key exchange (aPAKE) that supports mutual authentication in a client-server setting without reliance on PKI and with security against pre-computation attacks upon server compromise. In addition, the protocol provides forward secrecy and the ability to hide the password from the server, even during password registration. This document specifies the core OPAQUE protocol, along with several instantiations in different authenticated key exchange protocols.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/cfrg/draft-irtf-cfrg-opaque.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 August 2021.

Copyright Notice

Table of Contents

1. Introduction

Password authentication is the prevalent form of authentication on the web and in many other applications. In the most common implementation, a client authenticates to a server by sending its client ID and password to the server over a TLS connection. This makes the password vulnerable to server mishandling, including accidentally logging the password or storing it in cleartext in a database. Server compromise resulting in access to these plaintext passwords is not an uncommon security incident, even among security-conscious companies. Moreover, plaintext password authentication over TLS is also vulnerable to TLS failures, including many forms of PKI attacks, certificate mishandling, termination outside the security perimeter, visibility to middleboxes, and more.

Asymmetric (or Augmented) Password Authenticated Key Exchange (aPAKE) protocols are designed to provide password authentication and mutually authenticated key exchange in a client-server setting without relying on PKI (except during client/password registration) and without disclosing passwords to servers or other entities other than the client machine. A secure aPAKE should provide the best possible security for a password protocol. Namely, it should only be open to inevitable attacks, such as online impersonation attempts with guessed client passwords and offline dictionary attacks upon the compromise of a server and leakage of its password file. In the latter case, the attacker learns a mapping of a client's password under a one-way function and uses such a mapping to validate potential guesses for the password. Crucially important is for the password protocol to use an unpredictable one-way mapping. Otherwise, the attacker can pre-compute a deterministic list of mapped passwords leading to almost instantaneous leakage of passwords upon server compromise.

Despite the existence of multiple designs for (PKI-free) aPAKE protocols, none of these protocols are secure against pre-computation attacks. In particular, none of these protocols can use the standard technique against pre-computation that combines *secret* random values ("salt") into the one-way password mappings. Either these protocols do not use a salt at all or, if they do, they transmit the salt from server to client in the clear, hence losing the secrecy of the salt and its defense against pre-computation. Furthermore, transmitting the salt may require additional protocol messages.

This document describes OPAQUE, a PKI-free secure aPAKE that is secure against pre-computation attacks and capable of using a secret salt. OPAQUE provides forward secrecy (essential for protecting past communications in case of password leakage) and the ability to hide the password from the server - even during password registration. Furthermore, OPAQUE enjoys good performance and an array of additional features including the ability to increase the difficulty of offline dictionary attacks via iterated hashing or other hardening schemes, and offloading these operations to the client (that also helps against online guessing attacks); extensibility of the protocol to support storage and retrieval of client's secrets solely based on a password; being amenable to a multi-server distributed implementation where offline dictionary attacks are not possible without breaking into a threshold of servers (such a distributed solution requires no change or awareness on the client side relative to a single-server implementation).

OPAQUE is defined and proven as the composition of two functionalities: an oblivious pseudorandom function (OPRF) and an authenticated key exchange (AKE) protocol. It can be seen as a "compiler" for transforming any suitable AKE protocol into a secure aPAKE protocol. (See Section 6 for requirements of the OPRF and AKE protocols.) This document specifies one OPAQUE instantiation based on 3DH [SIGNAL]. Other instantiations are possible, as discussed in Appendix B, but their details are out of scope for this document. In general, the modularity of OPAQUE's design makes it easy to integrate with additional AKE protocols, e.g., IKEv2, and with future ones such as those based on post-quantum techniques.

OPAQUE consists of two stages: registration and authenticated key exchange. In the first stage, a client registers its password with the server and stores its encrypted credentials on the server. In the second stage, a client obtains those credentials, recovers them using the client's password, and subsequently uses them as input to an AKE protocol.

Currently, the most widely deployed PKI-free aPAKE is SRP [RFC2945], which is vulnerable to pre-computation attacks, lacks proof of security and is less efficient relative to OPAQUE. Moreover, SRP requires a ring as it mixes addition and multiplication operations, and thus does not work over plain elliptic curves. OPAQUE is therefore a suitable replacement for applications that use SRP.

This draft complies with the requirements for PAKE protocols set forth in [RFC8125].

1.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Notation

The following terms are used throughout this document to describe the operations, roles, and behaviors of OPAQUE:

- Client (C): Entity that has knowledge of a password and wishes to authenticate.

- Server (S): Entity that authenticates clients using passwords.

- password: An opaque byte string containing the client's password.

- I2OSP and OS2IP: Convert a byte string to and from a non-negative integer as described in Section 4 of [RFC8017]. Note that these functions operate on byte strings in big-endian byte order.

- concat(x0, ..., xN): Concatenate byte strings. For example, concat(0x01, 0x0203, 0x040506) = 0x010203040506.

- random(n): Generate a cryptographically secure pseudorandom byte string of length n bytes.

- xor(a,b): Apply XOR to byte strings. For example, xor(0xF0F0, 0x1234) = 0xE2C4. It is an error to call this function with two arguments of unequal length.

- ct_equal(a, b): Return true if a is equal to b, and false otherwise. This function is constant-time in the length of a and b, which are assumed to be of equal length, irrespective of the values a or b.

Except if said otherwise, random choices in this specification refer to drawing with uniform distribution from a given set (i.e., "random" is short for "uniformly random"). Random choices can be replaced with fresh outputs from a cryptographically strong pseudorandom generator, according to the requirements in [RFC4086], or pseudorandom function.

The name OPAQUE is a homonym of O-PAKE where O is for Oblivious. The name OPAKE was taken.

2. Cryptographic Protocol and Algorithm Dependencies

OPAQUE relies on the following protocols and primitives:

* Oblivious Pseudorandom Function (OPRF, [I-D.irtf-cfrg-voprf], version -06):

  - Blind(x): Convert input x into an element of the OPRF group, randomize it by some scalar r, producing M, and output (r, M).

  - GenerateKeyPair(): Generate an OPRF private and public key. OPAQUE only requires an OPRF private key. We write (oprf_key, _) = GenerateKeyPair() to denote use of this function for generating secret key oprf_key (and discarding the corresponding public key).

  - Evaluate(k, M): Evaluate input element M using private key k, yielding output element Z.

  - Finalize(x, r, Z): Finalize the OPRF evaluation using input x, random scalar r, and evaluation output Z, yielding output y.

  - SerializeScalar(s): Map a scalar s to a unique byte array buf of fixed length.

  - DeserializeScalar(buf): Map a byte array buf to a scalar s, or fail if the input is not a valid byte representation of a scalar.

  - SerializedElement: A serialized OPRF group element, a byte array of fixed length.

  - SerializedScalar: A serialized OPRF scalar, a byte array of fixed length.

Note that we only need the base mode variant (as opposed to the verifiable mode variant) of the OPRF described in [I-D.irtf-cfrg-voprf].

* Cryptographic hash function:

  - Hash(m): Compute the cryptographic hash of input message m. The type of the hash is determined by the chosen OPRF group.

  - Nh: The output size of the Hash function.

* Authenticated Key Exchange (AKE, Section 4.2):

  - Npk: The size of the public keys used for the key exchange protocol.

- Nsk: The size of the private keys used for the key exchange protocol.

* Memory Hard Function (MHF):

  - Harden(msg, params): Repeatedly apply a memory-hard function with parameters params to strengthen the input msg against offline dictionary attacks. This function also needs to satisfy collision resistance.

3. Offline Registration

Registration is executed between a client C and a server S. It is assumed S can identify C and the client can authenticate S during this registration phase. This is the only part in OPAQUE that requires an authenticated channel, either physical, out-of-band, PKI-based, etc. This section describes the registration flow, message encoding, and helper functions. Moreover, C has a key pair (client_private_key, client_public_key) for an AKE protocol which is suitable for use with OPAQUE; See Section 4. The private-public keys (client_private_key, client_public_key) may be randomly generated (using a cryptographically secure pseudorandom number generator) for the account or provided by the calling client. Clients MUST NOT use the same key pair (client_private_key, client_public_key) for two different accounts.

To begin, C chooses its password, and S chooses its own pair of private-public AKE keys (server_private_key, server_public_key) for use with the AKE. S can use the same pair of keys with multiple clients. These steps can happen offline, i.e., before the registration phase. Once complete, the registration process proceeds as follows:

```
Client (password, creds)            Server (server_private_key, server_
-------------------------------------------------------------------
(request, blind) = CreateRegistrationRequest(password)

                              request
                      ------------------------->

            (response, oprf_key) = CreateRegistrationResponse(request, s

                              response
                      <-------------------------

(record, export_key) = FinalizeRequest(password, creds, blind, response

                               record
                      ------------------------->
```

Section 3.3 describes details of the functions referenced above.

Both client and server MUST validate the other party's public key before use. See Section 6.6 for more details.

Upon completion, S stores C's credentials for later use. See Section 3.3.4 for a recommended storage format.

3.1. Credential Storage

OPAQUE makes use of a structure Envelope to store client credentials. The Envelope structure embeds the following types of credentials:

- client_private_key: The encoded client private key for the AKE protocol.
- server_public_key: The encoded server public key for the AKE protocol.
- client_identity: The client identity. This is an application-specific value, e.g., an e-mail address or normal account name.
- server_identity: The server identity. This is typically a domain name, e.g., example.com. See Section 6.2 for information about this identity.

Each public and private key value is an opaque byte string, specific to the AKE protocol in which OPAQUE is instantiated. For example, if used as raw public keys for TLS 1.3 [RFC8446], they may be RSA or ECDSA keys as per [RFC7250].

These credentials are incorporated in the SecretCredentials and CleartextCredentials structs, depending on the mode set by the value of EnvelopeMode:

```
enum {
  base(1),
  custom_identifier(2),
  (255)
} EnvelopeMode;
```

The base mode defines SecretCredentials and CleartextCredentials as follows:

```
struct {
  opaque client_private_key[Nsk];
} SecretCredentials;

struct {
  opaque server_public_key[Npk];
} CleartextCredentials;
```

The custom_identifier mode defines SecretCredentials and CleartextCredentials as follows:

```
struct {
  opaque client_private_key[Nsk];
} SecretCredentials;

struct {
  opaque server_public_key[Npk];
  opaque client_identity<0..2^16-1>;
  opaque server_identity<0..2^16-1>;
} CleartextCredentials;
```

These credentials are embedded into the following Envelope structure with encryption and authentication.

```
struct {
  EnvelopeMode mode;
  opaque nonce[32];
  opaque encrypted_creds[Nsk];
} InnerEnvelope;

struct {
  InnerEnvelope inner_env;
  opaque auth_tag[Nh];
} Envelope;
```

mode The EnvelopeMode value. This MUST be one of base or custom_identifier.

nonce A unique 32-byte nonce used to protect this Envelope.

encrypted_creds Encoding of encrypted and authenticated SecretCredentials.

auth_tag Authentication tag protecting the contents of the envelope, covering InnerEnvelope and CleartextCredentials.

The full procedure for constructing Envelope and InnerEnvelope from SecretCredentials and CleartextCredentials is described in Section 3.3.3. Note that only SecretCredentials are stored in the Envelope (in encrypted form).

The EnvelopeMode value is specified as part of the configuration (see Section 5).

Credential information corresponding to the configuration-specific mode, along with the client public key client_public_key and private key client_private_key, are recommended to be stored in a Credentials object with the following named fields:

- *client_private_key, the client's private key
- *client_public_key, the client's public key corresponding to client_private_key
- *client_identity, an optional client identity (present only in the custom_identifier mode)
- *server_identity, an optional server identity (present only in the custom_identifier mode)

3.2. Registration Messages

```
struct {
    SerializedElement data;
} RegistrationRequest;
```

data A serialized OPRF group element.

```
struct {
    SerializedElement data;
    opaque server_public_key[Npk];
} RegistrationResponse;
```

data A serialized OPRF group element.

server_public_key The server's encoded public key that will be used for the online authenticated key exchange stage.

```
struct {
    opaque client_public_key[Npk];
    Envelope envelope;
} RegistrationUpload;
```

client_public_key The client's encoded public key, corresponding to the private key client_private_key.

envelope The client's Envelope structure.

3.3. Registration Functions

3.3.1. CreateRegistrationRequest

```
CreateRegistrationRequest(password)

Input:
- password, an opaque byte string containing the client's password

Output:
- request, a RegistrationRequest structure
- blind, an OPRF scalar value

Steps:
1. (blind, M) = Blind(password)
2. Create RegistrationRequest request with M
3. Output (request, blind)
```

3.3.2. CreateRegistrationResponse

```
CreateRegistrationResponse(request, server_public_key)

Input:
- request, a RegistrationRequest structure
- server_public_key, the server's public key

Output:
- response, a RegistrationResponse structure
- oprf_key, the per-client OPRF key known only to the server

Steps:
1. (oprf_key, _) = GenerateKeyPair()
2. Z = Evaluate(oprf_key, request.data)
3. Create RegistrationResponse response with (Z, server_public_key)
4. Output (response, oprf_key)
```

3.3.3. FinalizeRequest

```
FinalizeRequest(password, creds, blind, response)

Parameters:
- params, the MHF parameters established out of band
- mode, the InnerEnvelope mode
- Nh, the output size of the Hash function

Input:
- password, an opaque byte string containing the client's password
- creds, a Credentials structure
- blind, an OPRF scalar value
- response, a RegistrationResponse structure

Output:
- record, a RegistrationUpload structure
- export_key, an additional key

Steps:
1. y = Finalize(password, blind, response.data)
2. envelope_nonce = random(32)
3. prk = HKDF-Extract(envelope_nonce, Harden(y, params))
4. Create SecretCredentials secret_creds with creds.client_private_key
5. Create CleartextCredentials cleartext_creds with response.server_publ
   and custom identifiers creds.client_identity and creds.server_identit
   mode is custom_identifier
6. pseudorandom_pad =
     HKDF-Expand(prk, "Pad", len(secret_creds))
7. auth_key = HKDF-Expand(prk, "AuthKey", Nh)
8. export_key = HKDF-Expand(prk, "ExportKey", Nh)
9. encrypted_creds = xor(secret_creds, pseudorandom_pad)
10. Create InnerEnvelope inner_env
      with (mode, envelope_nonce, encrypted_creds)
11. auth_tag = HMAC(auth_key, concat(inner_env, cleartext_creds))
12. Create Envelope envelope with (inner_env, auth_tag)
13. Create RegistrationUpload record with (envelope, creds.client_public
14. Output (record, export_key)
```

The inputs to HKDF-Extract and HKDF-Expand are as specified in [RFC5869]. The underlying hash function is that which is associated with the OPAQUE configuration (see Section 5).

See Section 4 for details about the output export_key usage.

Upon completion of this function, the client MUST send record to the server.

3.3.4. Credential File

The server then constructs and stores the credential_file object, where envelope and client_public_key are obtained from record, and oprf_key is retained from the output of CreateRegistrationResponse. oprf_key is serialized using SerializeScalar. The below structure represents an example of how these values might be conveniently stored together.

```
struct {
    SerializedScalar oprf_key;
    opaque client_public_key[Npk];
    Envelope envelope;
} credential_file;
```

4. Online Authenticated Key Exchange

After registration, the client and server run the authenticated key exchange stage of the OPAQUE protocol. This stage is composed of a concurrent OPRF and key exchange flow. The key exchange protocol is authenticated using the client and server credentials established during registration; see Section 3. The type of keys MUST be suitable for the key exchange protocol. For example, if the key exchange protocol is 3DH, as described in Section 4.2.2, then the private and public keys must be Diffie-Hellman keys. In the end, the client proves its knowledge of the password, and both client and server agree on a mutually authenticated shared secret key.

OPAQUE produces two outputs: a session secret and an export key. The export key may be used for additional application-specific purposes, as outlined in Section 6.4. The output export_key MUST NOT be used in any way before the HMAC value in the envelope is validated. See Section 6.3 for more details about this requirement.

4.1. Credential Retrieval

The online AKE stage of the protocol requires clients to obtain and decrypt their credentials from the server-stored envelope. This process is similar to the offline registration stage, as shown below.

```
 Client (password)             Server (server_private_key, server_public
 ------------------------------------------------------------------------
 (request, blind) = CreateCredentialRequest(password)

                               request
                      ------------------------->

    response = CreateCredentialResponse(request, server_public_key, cred

                               response
                      <-------------------------

 (client_private_key, server_public_key, export_key) =
     RecoverCredentials(password, blind, response)
```

The rest of this section describes these credential retrieval functions in more detail.

4.1.1. Credential Retrieval Messages

```
struct {
    SerializedElement data;
} CredentialRequest;
```

data A serialized OPRF group element.

```
struct {
    SerializedElement data;
    opaque server_public_key[Npk];
    Envelope envelope;
} CredentialResponse;
```

data A serialized OPRF group element.

server_public_key The server's encoded public key that will be used for the online authenticated key exchange stage.

envelope The client's Envelope structure.

4.1.2. Credential Retrieval Functions

4.1.2.1. CreateCredentialRequest

```
CreateCredentialRequest(password)

Input:
- password, an opaque byte string containing the client's password

Output:
- request, a CredentialRequest structure
- blind, an OPRF scalar value

Steps:
1. (blind, M) = Blind(password)
2. Create CredentialRequest request with M
3. Output (request, blind)
```

4.1.2.2. CreateCredentialResponse

```
CreateCredentialResponse(request, server_public_key, credential_file)

Input:
- request, a CredentialRequest structure
- server_public_key, the public key of the server
- credential_file, the server's output from registration
  (see {{credential-file}})

Output:
- response, a CredentialResponse structure

Steps:
1. Z = Evaluate(DeserializeScalar(credential_file.oprf_key), request.dat
2. Create CredentialResponse response
    with (Z, server_public_key, credential_file.envelope)
3. Output response
```

4.1.2.3. RecoverCredentials

```
RecoverCredentials(password, blind, response)

Parameters:
- params, the MHF parameters established out of band
- Nh, the output size of the Hash function

Input:
- password, an opaque byte string containing the client's password
- blind, an OPRF scalar value
- response, a CredentialResponse structure

Output:
- client_private_key, the client's private key for the AKE protocol
- server_public_key, the public key of the server
- export_key, an additional key

Steps:
1. y = Finalize(password, blind, response.data)
2. contents = response.envelope.contents
3. envelope_nonce = contents.nonce
4. prk = HKDF-Extract(envelope_nonce, Harden(y, params))
5. pseudorandom_pad =
    HKDF-Expand(prk, "Pad", len(contents.encrypted_creds))
6. auth_key = HKDF-Expand(prk, "AuthKey", Nh)
7. export_key = HKDF-Expand(prk, "ExportKey", Nh)
8. Create CleartextCredentials cleartext_creds with response.server_publ
   and custom identifiers creds.client_identity and creds.server_identit
   custom_identifier
9. expected_tag = HMAC(auth_key, concat(contents, cleartext_creds))
10. If !ct_equal(response.envelope.auth_tag, expected_tag),
    raise DecryptionError
11. secret_creds = xor(contents.encrypted_creds, pseudorandom_pad)
12. Output (secret_creds.client_private_key, response.server_public_key,
```

4.2. AKE Instantiations

This section describes instantiations of OPAQUE using 3-message AKEs which satisfies the forward secrecy and KCI properties discussed in Section 6. As shown in [OPAQUE], OPAQUE cannot use less than three messages so the 3-message instantiations presented here are optimal in terms of number of messages. On the other hand, there is no impediment to using OPAQUE with protocols with more than 3 messages as in the case of IKEv2 (or the underlying SIGMA-R protocol [SIGMA]).

The generic outline of OPAQUE with a 3-message AKE protocol includes three messages KE1, KE2, and KE3, where KE1 and KE2 include key exchange shares, e.g., DH values, sent by client and server,

respectively, and KE3 provides explicit client authentication and full forward security (without it, forward secrecy is only achieved against eavesdroppers which is insufficient for OPAQUE security).

The output of the authentication phase is a session secret session_key and export key export_key. Applications can use session_key to derive additional keying material as needed. Key derivation and other details of the protocol are specified by the AKE scheme. We note that by the results in [OPAQUE], KE2 and KE3 must authenticate credential_request and credential_response, respectively, for binding between the underlying OPRF protocol messages and the KE session.

We use the parameters Npk and Nsk to denote the size of the public and private keys used in the AKE instantiation. Npk and Nsk must adhere to the output size limitations of the HKDF Expand function from [RFC5869], which means that Npk, Nsk <= 255 * Nh.

The rest of this section includes key schedule utility functions used by OPAQUE-3DH, and then provides a detailed specification for OPAQUE-3DH, including its wire format messages.

4.2.1. Key Schedule Utility Functions

The key derivation procedures for OPAQUE-3DH makes use of the functions below, re-purposed from TLS 1.3 [RFC8446].

```
HKDF-Expand-Label(Secret, Label, Context, Length) =
  HKDF-Expand(Secret, HkdfLabel, Length)
```

Where HkdfLabel is specified as:

```
struct {
   uint16 length = Length;
   opaque label<8..255> = "OPAQUE " + Label;
   opaque context<0..255> = Context;
} HkdfLabel;

Derive-Secret(Secret, Label, Transcript-Hash) =
    HKDF-Expand-Label(Secret, Label, Transcript-Hash, Nh)
```

HKDF uses Hash as its underlying hash function, which is the same as that which is indicated by the OPAQUE instantiation. Note that the Label parameter is not a NULL-terminated string.

4.2.2. OPAQUE-3DH Instantiation

OPAQUE-3DH is implemented using a suitable prime order group. All operations in the key derivation steps in Section 4.2.2.2 are performed in this group and represented here using multiplicative

notation. The output of OPAQUE-3DH is a session secret session_key and export key export_key.

The parameters Npk and Nsk are set to be equal to the size of an element and scalar, respectively, in the associated prime order group.

4.2.2.1. OPAQUE-3DH Messages

The three messages for OPAQUE-3DH are described below.

```
struct {
  CredentialRequest request;
  uint8 client_nonce[32];
  opaque client_info<0..2^16-1>;
  uint8 client_keyshare[Npk];
} KE1;
```

request A CredentialRequest generated according to Section 4.1.2.1.

client_nonce A fresh 32-byte randomly generated nonce.

client_info Optional application-specific information to exchange during the protocol.

client_keyshare Client ephemeral key share of fixed size Npk, where Npk depends on the corresponding prime order group.

```
struct {
  struct {
    CredentialResponse response;
    uint8 server_nonce[32];
    uint8 server_keyshare[Npk];
  } inner_ke2;
  opaque enc_server_info<0..2^16-1>;
  uint8 mac[Nh];
} KE2;
```

response A CredentialResponse generated according to Section 4.1.2.2.

server_nonce A fresh 32-byte randomly generated nonce.

server_keyshare Server ephemeral key share of fixed size Npk, where Npk depends on the corresponding prime order group.

enc_server_info Optional application-specific information to exchange during the protocol encrypted under key Ke2, defined below.

mac
An authentication tag computed over the handshake transcript computed using Km2, defined below.

```
struct {
  uint8 mac[Nh];
} KE3;
```

mac An authentication tag computed over the handshake transcript computed using Km3, defined below.

4.2.2.2. OPAQUE-3DH Key Schedule

OPAQUE-3DH requires MAC keys server_mac_key and client_mac_key and encryption key handshake_encrypt_key. Additionally, OPAQUE-3DH also outputs session_key. The schedule for computing this key material is below.

```
HKDF-Extract(salt=0, IKM)
    |
    +-> Derive-Secret(., "handshake secret", Hash(preamble)) = handshake
    |
    +-> Derive-Secret(., "session secret", Hash(preamble)) = session_key
```

From handshake_secret, Km2, Km3, and Ke2 are computed as follows:

```
server_mac_key =
  HKDF-Expand-Label(handshake_secret, "server mac", "", Nh)
client_mac_key =
  HKDF-Expand-Label(handshake_secret, "client mac", "", Nh)
handshake_encrypt_key =
  HKDF-Expand-Label(handshake_secret, "handshake enc", "", Nh)
```

Nh is the output length of the underlying hash function.

The Derive-Secret parameter preamble is computed as:

```
preamble = concat("3DH",
                  I2OSP(len(client_identity), 2), client_identity,
                  KE1,
                  I2OSP(len(server_identity), 2), server_identity,
                  KE2.inner_ke2)
```

See Section 6.2 for more information about identities client_identity and server_identity.

Let epkS and eskS be server_keyshare and the corresponding secret key, and epkU and eskU be client_keyshare and the corresponding secret key. The input parameter IKM the concatenation of three DH values computed by the client as follows:

```
IKM = concat(epkS^eskU, pkS^eskU, epkS^skU)
```

Likewise, IKM is computed by the server as follows:

```
IKM = concat(epkU^eskS, epkU^skS, pkU^eskS)
```

4.2.2.3. OPAQUE-3DH Encryption and Key Confirmation {#3dh-core}

Clients and servers use keys Km2 and Km3 in computing KE2.mac and KE3.mac, respectively. These values are computed as follows:

- KE2.mac = HMAC(Km2, Hash(concat(preamble, KE2.enc_server_info)), where preamble is as defined in Section 4.2.2.2.

- KE3.mac = HMAC(Km3, Hash(concat(preamble, KE2.enc_server_info, KE2.mac)), where preamble is as defined in Section 4.2.2.2.

The server application info, an opaque byte string server_info, is encrypted using a technique similar to that used for secret credential encryption. Specifically, a one-time-pad is derived from Ke2 and then used as input to XOR with the plaintext. In pseudocode, this is done as follows:

```
info_pad = HKDF-Expand(Ke2, "encryption pad", len(server_info))
enc_server_info = xor(info_pad, server_info)
```

5. Configurations

An OPAQUE configuration is a tuple (OPRF, Hash, MHF, EnvelopeMode, Group). The OPAQUE OPRF protocol is drawn from the "base mode" variant of [I-D.irtf-cfrg-voprf]. The following OPRF ciphersuites are supported:

- OPRF(ristretto255, SHA-512)

- OPRF(decaf448, SHA-512)

- OPRF(P-256, SHA-256)

- OPRF(P-384, SHA-512)

- OPRF(P-521, SHA-512)

Future configurations may specify different OPRF constructions.

The OPAQUE hash function is that which is associated with the OPRF ciphersuite. For the ciphersuites specified here, only SHA-512 and SHA-256 are supported.

The OPAQUE MHFs include Argon2 [I-D.irtf-cfrg-argon2], scrypt [RFC7914], and PBKDF2 [RFC2898] with fixed parameter choices.

The EnvelopeMode value is defined in Section 3.1. It MUST be one of base or custom_identifier. Future specifications may specify alternate EnvelopeMode values and their corresponding Envelope structure.

The Group mode identifies the group used in the OPAQUE-3DH AKE. This SHOULD match that of the OPRF.

6. Security Considerations

OPAQUE is defined and proven as the composition of two functionalities: an OPRF and an AKE protocol. It can be seen as a "compiler" for transforming any AKE protocol (with KCI security and forward secrecy - see below) into a secure aPAKE protocol. In OPAQUE, the client stores a secret private key at the server during password registration and retrieves this key each time it needs to authenticate to the server. The OPRF security properties ensure that only the correct password can unlock the private key while at the same time avoiding potential offline guessing attacks. This general composability property provides great flexibility and enables a variety of OPAQUE instantiations, from optimized performance to integration with TLS. The latter aspect is of prime importance as the use of OPAQUE with TLS constitutes a major security improvement relative to the standard password-over-TLS practice. At the same time, the combination with TLS builds OPAQUE as a fully functional secure communications protocol and can help provide privacy to account information sent by the client to the server prior to authentication.

The KCI property required from AKE protocols for use with OPAQUE states that knowledge of a party's private key does not allow an attacker to impersonate others to that party. This is an important security property achieved by most public-key based AKE protocols, including protocols that use signatures or public key encryption for authentication. It is also a property of many implicitly authenticated protocols, e.g., HMQV, but not all of them. We also note that key exchange protocols based on shared keys do not satisfy the KCI requirement, hence they are not considered in the OPAQUE setting. We note that KCI is needed to ensure a crucial property of OPAQUE: even upon compromise of the server, the attacker cannot impersonate the client to the server without first running an exhaustive dictionary attack. Another essential requirement from AKE protocols for use in OPAQUE is to provide forward secrecy (against active attackers).

6.1. Related Analysis

Jarecki et al. [OPAQUE] proved the security of OPAQUE in a strong aPAKE model that ensures security against pre-computation attacks and is formulated in the Universal Composability (UC) framework [Canetti01] under the random oracle model. This assumes security of the OPRF function and of the underlying key exchange protocol. In turn, the security of the OPRF protocol from [I-D.irtf-cfrg-voprf] is proven in the random oracle model under the One-More Diffie-Hellman assumption [JKKX16].

Very few aPAKE protocols have been proven formally, and those proven were analyzed in a weak security model that allows for pre-computation attacks (e.g., [GMR06]). This is not just a formal issue: these protocols are actually vulnerable to such attacks. This includes protocols that have recent analyses in the UC model such as AuCPace [AuCPace] and SPAKE2+ [SPAKE2plus]. We note that as shown in [OPAQUE], these protocols, and any aPAKE in the model from [GMR06], can be converted into an aPAKE secure against pre-computation attacks at the expense of an additional OPRF execution.

OPAQUE's design builds on a line of work initiated in the seminal paper of Ford and Kaliski [FK00] and is based on the HPAKE protocol of Xavier Boyen [Boyen09] and the (1,1)-PPSS protocol from Jarecki et al. [JKKX16]. None of these papers considered security against pre-computation attacks or presented a proof of aPAKE security (not even in a weak model).

6.2. Identities

AKE protocols generate keys that need to be uniquely and verifiably bound to a pair of identities. In the case of OPAQUE, those identities correspond to client_identity and server_identity. Thus, it is essential for the parties to agree on such identities, including an agreed bit representation of these identities as needed.

Applications may have different policies about how and when identities are determined. A natural approach is to tie client_identity to the identity the server uses to fetch envelope (hence determined during password registration) and to tie server_identity to the server identity used by the client to initiate an offline password registration or online authenticated key exchange session. server_identity and client_identity can also be part of the envelope or be tied to the parties' public keys. In principle, identities may change across different sessions as long as there is a policy that can establish if the identity is acceptable or not to the peer. However, we note that the public keys

of both the server and the client must always be those defined at the time of password registration.

The client identity (client_identity) and server identity (server_identity) are optional parameters that are left to the application to designate as monikers for the client and server. If the application layer does not supply values for these parameters, then they will be omitted from the creation of the envelope during the registration stage. Furthermore, they will be substituted with client_identity = client_public_key and server_identity = server_public_key during the authenticated key exchange stage.

The advantage to supplying a custom client_identity and server_identity (instead of simply relying on a fallback to client_public_key and server_public_key) is that the client can then ensure that any mappings between client_identity and client_public_key (and server_identity and server_public_key) are protected by the authentication from the envelope. Then, the client can verify that the client_identity and server_identity contained in its envelope matches the client_identity and server_identity supplied by the server.

However, if this extra layer of verification is unnecessary for the application, then simply leaving client_identity and server_identity unspecified (and using client_public_key and server_public_key instead) is acceptable.

6.3. Envelope Encryption

The analysis of OPAQUE from [OPAQUE] requires the authenticated encryption scheme used to produce envelope to have a special property called random key-robustness (or key-committing). This specification enforces this property by utilizing encrypt-then-HMAC in the construction of the envelope. There is no option to use another authenticated-encryption scheme with this specification. (Deviating from the key-robustness requirement may open the protocol to attacks, e.g., [LGR20].) We remark that export_key for authentication or encryption requires no special properties from the authentication or encryption schemes as long as export_key is used only after the envelope is validated, i.e., after the HMAC in RecoverCredentials passes verification.

6.4. Export Key Usage

The export key can be used (separately from the OPAQUE protocol) to provide confidentiality and integrity to other data which only the client should be able to process. For instance, if the server is expected to maintain any client-side secrets which require a

password to access, then this export key can be used to encrypt these secrets so that they remain hidden from the server.

6.5. Static Diffie-Hellman Oracles

While one can expect the practical security of the OPRF function (namely, the hardness of computing the function without knowing the key) to be in the order of computing discrete logarithms or solving Diffie-Hellman, Brown and Gallant [BG04] and Cheon [Cheon06] show an attack that slightly improves on generic attacks. For typical curves, the attack requires an infeasible number of calls to the OPRF or results in insignificant security loss; see [I-D.irtf-cfrg-voprf] for more information. For OPAQUE, these attacks are particularly impractical as they translate into an infeasible number of failed authentication attempts directed at individual users.

6.6. Input Validation

Both client and server MUST validate the other party's public key(s) used for the execution of OPAQUE. This includes the keys shared during the offline registration phase, as well as any keys shared during the online key agreement phase. The validation procedure varies depending on the type of key. For example, for OPAQUE instantiations using 3DH with P-256, P-384, or P-521 as the underlying group, validation is as specified in Section 5.6.2.3.4 of [keyagreement]. This includes checking that the coordinates are in the correct range, that the point is on the curve, and that the point is not the point at infinity. Additionally, validation MUST ensure the Diffie-Hellman shared secret is not the point at infinity.

6.7. OPRF Hardening

Hardening the output of the OPRF greatly increases the cost of an offline attack upon the compromise of the password file at the server. Applications SHOULD select parameters that balance cost and complexity.

6.8. Client Enumeration

Client enumeration refers to attacks where the attacker tries to learn whether a given client identity is registered with a server. Preventing such attacks requires the server to act with unknown client identities in a way that is indistinguishable from its behavior with existing clients. Here we suggest a way to implement such defense, namely, a way for simulating a CredentialResponse for non-existing clients. Note that if the same CredentialRequest is received twice by the server, the response needs to be the same in both cases (since this would be the case for real clients). For protection against this attack, one would apply the encryption

function in the construction of the envelope to all the key material in it. The server S will have two keys MK, MK' for a pseudorandom function f. f refers to a regular pseudorandom function such as HMAC or CMAC. Upon receiving a CredentialRequest for a non-existing client client_identity, S computes oprf_key=f(MK; client_identity) and oprf_key'=f(MK'; client_identity) and responds with CredentialResponse carrying Z=M^oprf_key and envelope, where the latter is computed as follows. prk is set to oprf_key' and secret_creds is set to the all-zero string (of the length of a regular envelope plaintext). Care needs to be taken to avoid side-channel leakage (e.g., timing) from helping differentiate these operations from a regular server response. The above requires changes to the server-side implementation but not to the protocol itself or the client-side.

There is one form of leakage that the above allows and whose prevention would require a change in OPAQUE. An attacker that attempts authentication with the same CredentialRequest twice and receives different responses can conclude that either the client registered with the service between these two activations or that the client was registered before but changed its password in between the activations (assuming the server changes oprf_key at the time of a password change). In any case, this indicates that client_identity is a registered client at the time of the second activation. To conceal this information, S can implement the derivation of oprf_key as oprf_key=f(MK; client_identity) also for registered clients. Hiding changes in the envelope, however, requires a change in the protocol. Instead of sending envelope as is, S would send an encryption of envelope under a key that the client derives from the OPRF result (similarly to prk) and that S stores during password registration. During the authenticated key exchange stage, the client will derive this key from the OPRF result, will use it to decrypt the envelope, and continue with the regular protocol. If S uses a randomized encryption, the encrypted envelope will look each time as a fresh random string, hence S can simulate the encrypted envelope also for non-existing clients.

Note that the first case above does not change the protocol so its implementation is a server's decision (the client side is not changed). The second case, requires changes on the client side so it changes OPAQUE itself.

[[https://github.com/cfrg/draft-irtf-cfrg-opaque/issues/22: Should this variant be documented/standardized?]]

6.9. Password Salt and Storage Implications

In OPAQUE, the OPRF key acts as the secret salt value that ensures the infeasibility of pre-computation attacks. No extra salt value is

needed. Also, clients never disclose their passwords to the server, even during registration. Note that a corrupted server can run an exhaustive offline dictionary attack to validate guesses for the client's password; this is inevitable in any aPAKE protocol. (OPAQUE enables defense against such offline dictionary attacks by distributing the server so that an offline attack is only possible if all - or a minimal number of - servers are compromised [OPAQUE].)

Some applications may require learning the client's password for enforcing password rules. Doing so invalidates this important security property of OPAQUE and is NOT RECOMMENDED. Applications should move such checks to the client. Note that limited checks at the server are possible to implement, e.g., detecting repeated passwords.

7. IANA Considerations

This document makes no IANA requests.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[AuCPace] Haase, B. and B. Labrique, "AuCPace: Efficient verifier-based PAKE protocol tailored for the IIoT", http://eprint.iacr.org/2018/286 , 2018.

[BG04] Brown, D. and R. Galant, "The static Diffie-Hellman problem", http://eprint.iacr.org/2004/306 , 2004.

[Boyen09] Boyen, X., "HPAKE: Password Authentication Secure against Cross-Site User Impersonation", Cryptology and Network Security (CANS) , 2009.

[Canetti01]
Canetti, R., "Universally composable security: A new paradigm for cryptographic protocols", IEEE Symposium on Foundations of Computer Science (FOCS) , 2001.

[Cheon06] Cheon, J.H., "Security analysis of the strong Diffie-Hellman problem", Euroctypt 2006 , 2006.

[FK00] Ford, W. and B.S. Kaliski, Jr, "Server-assisted generation of a strong secret from a password", WETICE , 2000.

[GMR06] Gentry, C., MacKenzie, P., and . Z, Ramzan, "A method for making password-based key exchange resilient to server compromise", CRYPTO , 2006.

[HMQV] Krawczyk, H., "HMQV: A high-performance secure Diffie-Hellman protocol", CRYPTO , 2005.

[I-D.irtf-cfrg-argon2] Biryukov, A., Dinu, D., Khovratovich, D., and S. Josefsson, "The memory-hard Argon2 password hash and proof-of-work function", Work in Progress, Internet-Draft, draft-irtf-cfrg-argon2-12, 8 September 2020, <http://www.ietf.org/internet-drafts/draft-irtf-cfrg-argon2-12.txt>.

[I-D.irtf-cfrg-hash-to-curve]
Faz-Hernandez, A., Scott, S., Sullivan, N., Wahby, R., and C. Wood, "Hashing to Elliptic Curves", Work in Progress, Internet-Draft, draft-irtf-cfrg-hash-to-curve-10, 11 October 2020, <http://www.ietf.org/internet-drafts/draft-irtf-cfrg-hash-to-curve-10.txt>.

[I-D.irtf-cfrg-voprf]
Davidson, A., Faz-Hernandez, A., Sullivan, N., and C. Wood, "Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups", Work in Progress, Internet-Draft, draft-irtf-cfrg-voprf-05, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-irtf-cfrg-voprf-05.txt>.

[I-D.sullivan-tls-opaque]
Sullivan, N., Krawczyk, H., Friel, O., and R. Barnes, "Usage of OPAQUE with TLS 1.3", Work in Progress, Internet-Draft, draft-sullivan-tls-opaque-00, 11 March 2019, <http://www.ietf.org/internet-drafts/draft-sullivan-tls-opaque-00.txt>.

[JKKX16] Jarecki, S., Kiayias, A., Krawczyk, H., and J. Xu, "Highly-efficient and composable password-protected

secret sharing (or: how to protect your bitcoin wallet online)", IEEE European Symposium on Security and Privacy , 2016.

[keyagreement] Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography", DOI 10.6028/nist.sp.800-56ar3, National Institute of Standards and Technology report, April 2018, <https://doi.org/10.6028/nist.sp.800-56ar3>.

[LGR20] Len, J., Grubbs, P., and T. Ristenpart, "Partitioning Oracle Attacks", n.d., <https://eprint.iacr.org/2020/1491.pdf>.

[OPAQUE] Jarecki, S., Krawczyk, H., and J. Xu, "OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-Computation Attacks", Eurocrypt , 2018.

[RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, DOI 10.17487/RFC2898, September 2000, <https://www.rfc-editor.org/info/rfc2898>.

[RFC2945] Wu, T., "The SRP Authentication and Key Exchange System", RFC 2945, DOI 10.17487/RFC2945, September 2000, <https://www.rfc-editor.org/info/rfc2945>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC7914] Percival, C. and S. Josefsson, "The scrypt Password-Based Key Derivation Function", RFC 7914, DOI 10.17487/RFC7914, August 2016, <https://www.rfc-editor.org/info/rfc7914>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version

2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8125] Schmidt, J., "Requirements for Password-Authenticated Key Agreement (PAKE) Schemes", RFC 8125, DOI 10.17487/RFC8125, April 2017, <https://www.rfc-editor.org/info/rfc8125>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[SIGMA] Krawczyk, H., "SIGMA: The SIGn-and-MAc approach to authenticated Diffie-Hellman and its use in the IKE protocols", CRYPTO , 2003.

[SIGNAL] "Signal recommended cryptographic algorithms", https://signal.org/docs/specifications/doubleratchet/#recommended-cryptographic-algorithms , 2016.

[SPAKE2plus] Shoup, V., "Security Analysis of SPAKE2+", http://eprint.iacr.org/2020/313 , 2020.

Appendix A. Acknowledgments

The OPAQUE protocol and its analysis is joint work of the author with Stas Jarecki and Jiayu Xu. We are indebted to the OPAQUE reviewers during CFRG's aPAKE selection process, particularly Julia Hesse and Bjorn Tackmann. This draft has benefited from comments by multiple people. Special thanks to Richard Barnes, Dan Brown, Eric Crockett, Paul Grubbs, Fredrik Kuivinen, Payman Mohassel, Jason Resch, Greg Rubin, and Nick Sullivan.

Appendix B. Alternate AKE Instantiations

It is possible to instantiate OPAQUE with other AKEs, such as HMQV [HMQV] and SIGMA-I. HMQV is similar to 3DH but varies in its key schedule. SIGMA-I uses digital signatures rather than static DH keys for authentication. Specification of these instantiations is left to future documents. A sketch of how these instantiations might change is included in the next subsection for posterity.

The AKE private key size (Nsk) is limited to the output size of the HKDF Expand function from [RFC5869]. Future specifications which have keys exceeding this size should specify a mechanism by which private keys and their corresponding public keys can be deterministically derived from a fixed-length seed.

OPAQUE may also be instantiated with any post-quantum (PQ) AKE protocol that has the message flow above and security properties

(KCI resistance and forward secrecy) outlined in Section 6. Note that such an instantiation is not quantum-safe unless the OPRF is quantum-safe. However, an OPAQUE instantiation where the AKE is quantum-safe, but the OPRF is not, would still ensure the confidentiality of application data encrypted under session_key (or a key derived from it) with a quantum-safe encryption function.

B.1. HMQV Instantiation Sketch

An HMQV instantiation would work similar to OPAQUE-3DH, differing primarily in the key schedule [HMQV]. First, the key schedule preamble value would use a different constant prefix - "HMQV" instead of "3DH" - as shown below.

```
preamble = concat("HMQV",
                  I2OSP(len(client_identity), 2), client_identity,
                  KE1,
                  I2OSP(len(server_identity), 2), server_identity,
                  KE2.inner_ke2)
```

Second, the IKM derivation would change. Assuming HMQV is instantiated with a cyclic group of prime order p with bit length L, clients would compute IKM as follows:

```
u' = (eskU + u \* skU) mod p
IKM = (epkS \* pkS^s)^u'
```

Likewise, servers would compute IKM as follows:

```
s' = (eskS + s \* skS) mod p
IKM = (epkU \* pkU^u)^s'
```

In both cases, u would be computed as follows:

```
hashInput = concat(I2OSP(len(epkU), 2), epkU,
                   I2OSP(len(info), 2), info,
                   I2OSP(len("client"), 2), "client")
u = Hash(hashInput) mod L
```

Likewise, s would be computed as follows:

```
hashInput = concat(I2OSP(len(epkS), 2), epkS,
                   I2OSP(len(info), 2), info,
                   I2OSP(len("server"), 2), "server")
s = Hash(hashInput) mod L
```

Hash is the same hash function used in the main OPAQUE protocol for key derivation. Its output length (in bits) must be at least L.

B.2. SIGMA-I Instantiation Sketch

A SIGMA-I instantiation differs more drastically from OPAQUE-3DH, since authentication uses digital signatures in lieu of Diffie Hellman. In particular, both KE2 and KE3 would carry a digital signature, computed using the server and client private keys established during registration, respectively, as well as a MAC, where the MAC is computed as in OPAQUE-3DH.

The key schedule would also change. Specifically, the key schedule preamble value would use a different constant prefix - "SIGMA-I" instead of "3DH" - and the IKM computation would use only the ephemeral key shares exchanged between client and server.

Appendix C. Test Vectors

This section contains test vectors for the OPAQUE-3DH specification. Each test vector specifies the configuration information, protocol inputs, intermediate values computed during registration and authentication, and protocol outputs. All values are encoded in hexadecimal strings. The configuration information includes the (OPRF, Hash, MHF, EnvelopeMode, Group) tuple, where the Group matches that which is used in the OPRF. These test vectors were generated using draft-06 of [I-D.irtf-cfrg-voprf].

C.1. OPAQUE-3DH Test Vector 1

C.1.1. Configuration

```
OPRF: 0001
Hash: SHA512
SlowHash: Identity
EnvelopeMode: 01
Group: ristretto255
Nh: 64
Npk: 32
Nsk: 32
```

C.1.2. Input Values

password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: cc7abb200199d5071c94efa49fb62435d3e70d03cf9573a95da54
20d3eebcd2b
client_private_key: 8bcb0b70dac18de24eef12e737d6b28724d3e37774e0b092f
9f70b255defaf04
client_public_key: 360e716c676cfe4d9968d1a352ed3faf17603863e0a7aa1905
df6ea129343b09
server_private_key: f3a0829898a89239dce29ccc98ec8b449a34b255ba1e6f944
829d18e0d589b0f
server_public_key: 66e130c6eb5b41f851b235b03a0eafeaa883f64147bc62cb74
9c22c762389c3c
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: 98b8081059f60ffed9336f026fd8e124737205ac73f5348ae5bebdb
49456c70f
client_nonce: 58dc21475ff730342f807bf031c7ae47a11f0d4dfaa63a7feb15d7e
36427ca44
server_keyshare: 5214e3ddc73db786480b79fa2da787f2080b82cbe922c2a9592b
44597d9a702e
client_keyshare: a4084c7296b1a3d5a5e4a24358750489575acfd8fcfa6e787492
b98265a5e651
server_private_keyshare: c4d002aa4cfcf281657cf36fe562bc60d9133e0e72a7
4432f685b2b6a4b42a0c
client_private_keyshare: de2e98f422bf7b99be19f7da7cac62f1599d35a225ec
6340149a0aaff3102003
blind_registration: 7e5bcbf82a46109ee0d24e9bcab41fc830a6ce8b82fc1e921
3a043b743b95800
blind_login: c4d5a15f0d5ffc354e340454ec779f575e4573a3886ab5e57e4da298
4bdd5306
oprf_key: 080d0a4d352de92672ab709b1ae1888cb48dfabc2d6ca5b914b335512fe
70508

C.1.3. Intermediate Values

auth_key: 7bb7f2b831ee30d3e5cc4012c8f721a4d8f9dd494932d53776e043df9bd
2aa284025b8b006fd8449536446ff50698f46c73fccb53f20d80898f185307d1d39e5
prk: b0aefddbb21d1b97bc40c07b172e0bf172ec740de4f6274f69d46350a447e9b1
b3fb1e4cefc7d8e393ff58a5c45c74d0615ee0eecde116f3d4e744142eb2ee89
pseudorandom_pad: 36a828b3b57bf242c4c47ccd9cb84e5b3cefaffe09629c6b94d
eba0ccec5fa39
envelope: 01cc7abb200199d5071c94efa49fb62435d3e70d03cf9573a95da5420d3
eebcd2bbd6323c36fba7fa08a2b6e2aab6efcdc183c4c897d822cf96d29b129932a55
3d469ffa9999fcbd37a1e8b6c1e579bcf83fed355c9ff413e6158d72d16f3ccd8699e
906027842694b6293b6303bbb7f324e0fccb4ae0f01edb60ee1d32992696e
handshake_secret: 2b041dcf12ac9b75dded88f891c25d76746ce9e2c1a43118ac4
aa5721cdc1bc2f0691e6c012a1ea9eb95ab4899b3e7058d37fe9546c46b0511877e40
f55aac6c
handshake_encrypt_key: ceef10f15d869a4cea8174fa98d0d96c7aaf8602d006fe
0c5274a40173db76cac820138c5890bb63fb974d1e3e925850cc2464e2c10f0a9a776
9a45e80889b1e
server_mac_key: f8fd7fdc349b5ae1339515e05912c89a795f561a117cdc84d8d8b
5f05b05751abfb87fa01c799c5d367244d1e32eab67ff926833c6025c556acffa4af1
f3871a
client_mac_key: 92a30cc82c374c06895aa07e81f0cf5f25309a24b595faefcd225
1f9219b47e47d17da4fe8b572dedefa350ed365f87b217973e90d0b647a2ccf1d796a
8970f6

C.1.4. Output Values

```
registration_request: ec9027daa5e9a901d641286a7ded51364142936ac7636e1
42e3f4368b4bd8124
registration_response: 8867d7c8c2c576a6322d49d46078ea32f479aed917c70a
636d3ada4397ea1c0e66e130c6eb5b41f851b235b03a0eafeaa883f64147bc62cb749
c22c762389c3c
registration_upload: 360e716c676cfe4d9968d1a352ed3faf17603863e0a7aa19
05df6ea129343b0901cc7abb200199d5071c94efa49fb62435d3e70d03cf9573a95da
5420d3eebcd2bbd6323c36fba7fa08a2b6e2aab6efcdc183c4c897d822cf96d29b129
932a553d469ffa9999fcbd37a1e8b6c1e579bcf83fed355c9ff413e6158d72d16f3cc
d8699e906027842694b6293b6303bbb7f324e0fccb4ae0f01edb60ee1d32992696e
KE1: e06a32011e1b1704eb686b263e5d132fff4e9f6429cd93b98db107485006792c
58dc21475ff730342f807bf031c7ae47a11f0d4dfaa63a7feb15d7e36427ca4400096
8656c6c6f20626f62a4084c7296b1a3d5a5e4a24358750489575acfd8fcfa6e787492
b98265a5e651
KE2: 66f6b5fa1a4eb6bd7a0c93ed2639a31cba0d02e2df744003641d5a30a4a12364
66e130c6eb5b41f851b235b03a0eafeaa883f64147bc62cb749c22c762389c3c01cc7
abb200199d5071c94efa49fb62435d3e70d03cf9573a95da5420d3eebcd2bbd6323c3
6fba7fa08a2b6e2aab6efcdc183c4c897d822cf96d29b129932a553d469ffa9999fcb
d37a1e8b6c1e579bcf83fed355c9ff413e6158d72d16f3ccd8699e906027842694b62
93b6303bbb7f324e0fccb4ae0f01edb60ee1d32992696e98b8081059f60ffed9336f0
26fd8e124737205ac73f5348ae5bebdb49456c70f5214e3ddc73db786480b79fa2da7
87f2080b82cbe922c2a9592b44597d9a702e000f72f38a0945819089c44c86820c51d
89cc35f77df03d330101bbed3b2f69066112f32529bdda0998657350fc9f8da4cde73
408ad931f4c2ea6237ccae4696483388b174f50cf96d439139b0f8680c3b
KE3: 9f0e4f73455ca9fe06bb52ad02670b09be5a03db11a73be4422f19963be082b0
eb55871022e8d1d87adc3ab50de7c738058eb659866d091648f2fed12e23fd53
export_key: 66c0b72aa829f13a166fb1a1168f1e26023921f0eed1126def4f81ba0
4924ad6012e42b63656ec199ba27670d1e7f23dc0a927714edc140134dde5a5d2063d
fc
session_key: 951c2bb1b876725fa7d3829db791dddd406a688507b47e24101bd0cc
5d071760b6fba59e8758a6ea6d7e5f51a715b49a47c50fee9a7c8a0451243c3ee837f
d30
```

C.2. OPAQUE-3DH Test Vector 2

C.2.1. Configuration

```
OPRF: 0002
Hash: SHA512
SlowHash: Identity
EnvelopeMode: 01
Group: decaf448
Nh: 64
Npk: 56
Nsk: 56
```

C.2.2. Input Values

password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: a3810457d65c1f14524f6a9fc0ad0d43c17732ae2f159e64aeeaf
c56dd63a49c
client_private_key: 614bb578f29cc677ea9e7aea3e4839413997e020f9377b63c
13584156a09a46dd2a425c41eac0e313a47e99d05df72c6e1d58e6592577a0d
client_public_key: a8f6d7dad9ec587964d6dffb1b63f951dc30a934137eb42057
f390d593dfafb6a687ec5c3ad3c35bb6a71338dc8106bd53b3a4fcec6110a1
server_private_key: 4c115060bca87db7d73e00cbb8559f84cb7a221b235b0950a
0ab553f03f10e1386abe954011b7da62bb6599418ef90b5d4ea98cc28aff517
server_public_key: bc66494bf44cdfed66f6b4c482a18e00a3d16a09d11775064f
963cc7bae3b6592a6b03fb982f5b5676972005a29d1dcfd46b6986088ca9d4
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: f1b33f774f5166cde92f150e2bebf58bbd5f194a7e48ceb6078b235
e185af474
client_nonce: b7b96900d3092b9547505710da9762561ed69da9dd1a5444c6215de
6fa2a40f8
server_keyshare: 5472ca0fc98d652ded1ba4edc7d876a791b2c2a61c1201ffe354
8e0f3a1d479e1629a35a7f910ef27f46c93ade70ee4cbdf9a2183f6d0754
client_keyshare: 46770873fdab1e43177a9f1b2d127a44e0b4c2becf3ac4545248
ef410d143cce32f76df27f47cf19347b42e3cd1f9432cda204701e188c32
server_private_keyshare: 0676f161ab555182908dbd7947b1c988956fa73b17b3
73b72fd4e3c0264a26aa4cab20fd864de6ceab345e8d755997956ddd1f265e8be105
client_private_keyshare: d0f08ac99ef2ab5b26fa7b2a6d920c76cf03fb57bdea
cc2ec39330fd6e7f9e5dbdfcb571168f337dd52851d4bc07947c858dc735bab8ca2b
blind_registration: a614f1894bcf6a1c7cef33909b794fe6e69a642b20f4c9118
febffaf6b6a31471fe7794aa77ced123f07e56cc27de60b0ab106c0b8eab127
blind_login: 668b3aab5207735beb86c5379228da260159dc24f7c5c2483a81aff8
d9ff676ebd81db28eb1e147561c478a3f84cbf77037f01025c7fd42a
oprf_key: 93dd2d051e90efe4e2e093bc1e82b80e8cce6afa798ac214abffabac2a2
58015d93e9faf0f2009d16c797646097d761e2b84e0df971d7b39

C.2.3. Intermediate Values

auth_key: 79ba15aa7db6b18f4d737be4dee7d5f186c235e4f8ec865513af292c056
b86e93bb8ebe2915fe0f43187e7cbd4bb94661ceb6496cbc877d598d001296defd7a3
prk: 67292e51a7da30331f94411e51ad6c03cdd2e0c5fce4c6a125cdf8cff688b9f5
100a5def2c0ef959db505ea09e81c43afc1450ed767de9d2cbc9f93fb7a306a9
pseudorandom_pad: 4b1221753544450748eda40790e4712480e36c451db367025d6
6843df9d916bf2c94696c60b951bf984f71d2b8d70d5980aacb9602a12af7
envelope: 01a3810457d65c1f14524f6a9fc0ad0d43c17732ae2f159e64aeeafc56d
d63a49c2a59940dc7d88370a273deedaeac4865b9748c65e4841c619c53002893d0b2
d2fe304ca87e155f8ea208984fbd087f9f617f45f390f650fac6b3a40c7e6b1d4b8c5
5b0575eac0f323c99337c232fad2dc07538eacebd947516a97ac420aa1f1851a20552
4e5dda112d0d949f3003777c47ff0724e8b0aed6
handshake_secret: 830f54c125468f05d8403f455ab1440885602e3dacab1e0a69b
0c6664eac71991e65049e215a5abda52d8960d4c68c58b5003afda0cd59e2487bfa60
5e42f209
handshake_encrypt_key: e29949872e78acfc53c7eda5d0bb30364c128ef5f69d02
efa3d18d08f7b1ca79fda1da3ed12ee796585845d804a961e29451755a8e7cb43c28f
bcc028903be05
server_mac_key: b12f274b224ff2bc1413ac87d52d4059da2af57375514cc43eeb4
e9ef602eddc149f7c8ca8afbe403e0997b859034bd304efd3d51750b1c38eccacfe78
5b17f0
client_mac_key: 9216883df807ee47d405443aaf990d2712b30ed400079ae54375a
51d1ced956c4153853e50bda1f1cf8e5323df3d9418b9d91e9be9cd4eb0b5efdc76e5
ad1c01

C.2.4. Output Values

registration_request: d21b318acf1b255d0f009bf3cb24b7b2f88cb58880775b8
dff43a81ab49fe73f0356b70ff3e5c251bc9810767c98491d8187d2cf11dff618
registration_response: c023432da8f17d6e5e740d9d1a0fb55dbc8e1830bd72ec
2e1f59da065858170b05c1f711ca085d8cf5a52ae1ea5198196bd9907dca045c6fbc6
6494bf44cdfed66f6b4c482a18e00a3d16a09d11775064f963cc7bae3b6592a6b03fb
982f5b5676972005a29d1dcfd46b6986088ca9d4
registration_upload: a8f6d7dad9ec587964d6dffb1b63f951dc30a934137eb420
57f390d593dfafb6a687ec5c3ad3c35bb6a71338dc8106bd53b3a4fcec6110a101a38
10457d65c1f14524f6a9fc0ad0d43c17732ae2f159e64aeeafc56dd63a49c2a59940d
c7d88370a273deedaeac4865b9748c65e4841c619c53002893d0b2d2fe304ca87e155
f8ea208984fbd087f9f617f45f390f650fac6b3a40c7e6b1d4b8c55b0575eac0f323c
99337c232fad2dc07538eacebd947516a97ac420aa1f1851a205524e5dda112d0d949
f3003777c47ff0724e8b0aed6
KE1: 30a31f471b8adc9e3fcb796a6ee1ee97edabf6a77468c58621a0cfaecee3c1ac
1a1dbe16a0fbf6fc4d2f882d8431303bded7a16d207f840cb7b96900d3092b9547505
710da9762561ed69da9dd1a5444c6215de6fa2a40f8000968656c6c6f20626f624677
0873fdab1e43177a9f1b2d127a44e0b4c2becf3ac4545248ef410d143cce32f76df27
f47cf19347b42e3cd1f9432cda204701e188c32
KE2: 2453710eb7a3226bfedb501efe06772d9450aa9ba9eed8adddd931964364e3d3
d5d2a22822f0d85569fa396b8e9c6657ff5115dbd4c0a218bc66494bf44cdfed66f6b
4c482a18e00a3d16a09d11775064f963cc7bae3b6592a6b03fb982f5b5676972005a2
9d1dcfd46b6986088ca9d401a3810457d65c1f14524f6a9fc0ad0d43c17732ae2f159
e64aeeafc56dd63a49c2a59940dc7d88370a273deedaeac4865b9748c65e4841c619c
53002893d0b2d2fe304ca87e155f8ea208984fbd087f9f617f45f390f650fac6b3a40
c7e6b1d4b8c55b0575eac0f323c99337c232fad2dc07538eacebd947516a97ac420aa
1f1851a205524e5dda112d0d949f3003777c47ff0724e8b0aed6f1b33f774f5166cde
92f150e2bebf58bbd5f194a7e48ceb6078b235e185af4745472ca0fc98d652ded1ba4
edc7d876a791b2c2a61c1201ffe3548e0f3a1d479e1629a35a7f910ef27f46c93ade7
0ee4cbdf9a2183f6d0754000f33baf0c6e2bcaa8a7155309d0b41d29d94469ac63cb5
9f2d54c81f8c445303d666eabe6a505aa7f5cab2d133c0bd40a951a56c9f2cb4f04e3
3ebae80858691b3799ecd1caf1be781f6f6e4c212661f
KE3: ae29171538fb3a9b3e643ec4812218f6e17038314b9a62e87ef2c72e82ed69a8
86c6f89ee2646dbb742905d88ed481894dd52f6acc3b31c0dbd46ba58bf0841d
export_key: 095d3246344474969774de1d7e21ea4835975c7dde152e446cf989a10
d9c8444e432fb65504dd80c2d4c2fa31203764c0fa4a15128541a8474f7c45119673e
16
session_key: 354a951281611e3b2ec14bf9b19325be441dc3395f4cc99838330071
5fdbacd282be642cb97f1e88f640c80b85452680aac8938970c1db61e398e87d9f0e9
2c5

C.3. OPAQUE-3DH Test Vector 3

C.3.1. Configuration

```
OPRF: 0003
Hash: SHA256
SlowHash: Identity
EnvelopeMode: 01
Group: P256_XMD:SHA-256_SSWU_RO_
Nh: 32
Npk: 33
Nsk: 32
```

C.3.2. Input Values

```
password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: d07d294b6e901c4d5b76c7097a56dd893c47a236c7e0a275b5819
48f77f28abb
client_private_key: 67b5bcebad6393e2d0b7db3d2b4597a670a5204b2b606f5a2
8328916e1e5ea5b
client_public_key: 030e2b9005157dfd740a13c9525a2132512a463927174d9728
0f80f962d1a650e3
server_private_key: fc2e715b2db1e7a3ad4ff8af1b24daa1922d13757ac9df4ad
c7e4e0b6b399433
server_public_key: 03ca5ebe2c9b87ff1e76e2e72f8a59273fe5c9688fee7dd2f2
8964187a0940c397
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: dc70522e4b7fb4c433ca014299328213a829cbf89e12fb3bfa08bc6
53fa3cbc9
client_nonce: 06eaf939cc8064e7d2c454faf5342f43c59b0974b55a965cf5b8ce3
52551d137
server_keyshare: 03a716d72106b7ad668ae097e553a46f4dd96961816fbe8e2243
43a7f0ab95a05a
client_keyshare: 037086638b1beb10b9a6a44ea0ee5b369081004df36cba0c16c8
d485482de57ff6
server_private_keyshare: a6532c99c1ea3f03d05f6e78dc1edabd3b9631be9f8b
274d9aaf671bfb6a7753
client_private_keyshare: 29bf435021b89c683259773bc686956af0c7822ba317
fb5e86028c44b92bd3af
blind_registration: 3e7c5057e09e220065ea8c257c0dd6055c4b401063eff0bf2
42b4cd534a79bad
blind_login: dcf6744d388ca013ef33edd369304ed96fc56c7c6c0bd369f8e926ff
e4854a59
oprf_key: f6b3e908bdb38e3c626a939e19daca653b9217801b5d51cef66d9fdbd94
a5354
```

C.3.3. Intermediate Values

```
auth_key: 226ffbc20f19c9d956d1f172bd5c9b1bb475716b6438aedfdcc559756ee
236f7
prk: 4d00010592e76a29c3173d883e94b55af3802c85983bb1374407a226a374cff4
pseudorandom_pad: 0123cec727a51fecb61d447f04fac3591299cd50bf55cf55334
2f1f7fc0b1e07
envelope: 01d07d294b6e901c4d5b76c7097a56dd893c47a236c7e0a275b581948f7
7f28abb6696722c8ac68c0e66aa9f422fbf54ff623ced1b9435a00f1b7078e11deef4
5cd47baf68df4d4eb20b76f04f8dbaba219bf0f4e5c6d47bf2b4bf5fe23e724bff
handshake_secret: 4d80ee40c7a11a9c4b27ca9ee6e38797322a9e3c904b3e35c8a
00bbad403029d
handshake_encrypt_key: 244d132b27991dafff260364681ff473472249ea2ae625
c223144b97ca754c44
server_mac_key: 99f2a9e5af84a221b3efeda5a5bb5372a587462a5585f33ea0c85
67aedcf03ea
client_mac_key: f86379af46726649af600c69dacaa96ed967eac0f91225ae00c68
1d86884408a
```

C.3.4. Output Values

```
registration_request: 0295067c743d15a0a9d4c6c15511b67e3858e9e22f8c44a
0c1de6e33cda494024b
registration_response: 02465cd175b404fc1426b3b9518a79ee219007679909f4
59f92cfc89929c89458e03ca5ebe2c9b87ff1e76e2e72f8a59273fe5c9688fee7dd2f
28964187a0940c397
registration_upload: 030e2b9005157dfd740a13c9525a2132512a463927174d97
280f80f962d1a650e301d07d294b6e901c4d5b76c7097a56dd893c47a236c7e0a275b
581948f77f28abb6696722c8ac68c0e66aa9f422fbf54ff623ced1b9435a00f1b7078
e11deef45cd47baf68df4d4eb20b76f04f8dbaba219bf0f4e5c6d47bf2b4bf5fe23e7
24bff
KE1: 0230e8f9b3689b65b952bf044702673c4d5278119b25d3833a3de655b9289f89
e106eaf939cc8064e7d2c454faf5342f43c59b0974b55a965cf5b8ce352551d137000
968656c6c6f20626f62037086638b1beb10b9a6a44ea0ee5b369081004df36cba0c16
c8d485482de57ff6
KE2: 025c6387cee347fa24a57c7021890ee13f435ea5e92b20fb488c3984e060ad4d
6f03ca5ebe2c9b87ff1e76e2e72f8a59273fe5c9688fee7dd2f28964187a0940c3970
1d07d294b6e901c4d5b76c7097a56dd893c47a236c7e0a275b581948f77f28abb6696
722c8ac68c0e66aa9f422fbf54ff623ced1b9435a00f1b7078e11deef45cd47baf68d
f4d4eb20b76f04f8dbaba219bf0f4e5c6d47bf2b4bf5fe23e724bffdc70522e4b7fb4
c433ca014299328213a829cbf89e12fb3bfa08bc653fa3cbc903a716d72106b7ad668
ae097e553a46f4dd96961816fbe8e224343a7f0ab95a05a000f2519167d418dc3c7ea
cd8cae6a55c483bef6d2f7a67d48ce30701d858da6caec7cdf7cac7990777eb410591
612736e
KE3: 5ef3c7290d3af42afdfc6b87af044c00ec8177e34d9a9464b8eb4c033424ad82
export_key: a5d8bb0d880b6b5bdfda31d2f01b695435bccbb57a5f8efd5dae1e46b
2a959c4
session_key: b6bece8471d280561dc969b22607b29c2836d1ca9a49e81a15ddb85c
467662e7
```

C.4. OPAQUE-3DH Test Vector 4

C.4.1. Configuration

```
OPRF: 0004
Hash: SHA512
SlowHash: Identity
EnvelopeMode: 01
Group: P384_XMD:SHA-512_SSWU_RO_
Nh: 64
Npk: 49
Nsk: 48
```

C.4.2. Input Values

```
password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: 22bc649eec25874f87e625008d252fceffbb8b434327914db6c09
d07b7f3bf19
client_private_key: f3cfa0420080cbff2e3431bcc25f80b409c533dd21924d77b
cbd10873989b7e58306b863276ae74049615162a416d508
client_public_key: 02c86369d6eae0978bdd4030b43e0619ce46ea9d91fa6e0e75
75bb12aa4857db98b952d8af9d92f75899c49d0d18793c1e
server_private_key: 2902c13bdc9993d3717bda68fc080b9802ae4effd5dc972d9
f9fb3bbbf106add174393effaf0a175fa8e85f898568620
server_public_key: 0251f78cbd5c7a3fbf4cdaeb755eb8cc4159edb0ef38baebb5
03dbefed5c89c14f7c2b99ed242b3d1de890f7515bad94bd
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: 7e4c5d3a4f779d5966cc8ca367e47e0716ad0f33aa100fe68507920
a1a6e3578
client_nonce: af0eb9b48c985e2ba859ce089fc342c648ddadfca4735f5b2e631ee
a7fdf967b
server_keyshare: 035684e8cdae6ce360c2d3f41e4b059f34f986e92dd50c8255f7
d7dc0c16252ef9b8c9fb9e0c2846053355e7fcfa46781f
client_keyshare: 027044d31354b5b98587d7d144526f6c8528317cc3e9675c90a1
952f4bc2725a6a154f59aed10e4ff7ec68d4917d74122b
server_private_keyshare: 3e3869237f74106241777f230582e849076f08753056
c186437ded8ee22f96e44bd5b6ec07cb131d51cf1324c1238699
client_private_keyshare: 59b596174a682828f3934d510217ce7890f67cafc0ff
aa7a1e1d1ced3c477fea691e696032c8709c86cbcda2b184ad01
blind_registration: 3382c7ec9bdd6e75898e4877d8e2bc16359073c015b92d154
50f7fb395bf52c6ea98384c491fe4e4d423b59de7b0df39
blind_login: 29d29abeabede9788d11782429bff296102f6338df84c9602bfa9e7d
690b1f7a173d07e6d54a419db4a6308f8b09589f
oprf_key: 4283efb9cd1ee4061c6bf884e60a877321ece4f9b6ffd01ce8208254541
3bd9bb5e8f3c63b86ae88d9ce0530b01cb1c3
```

C.4.3. Intermediate Values

```
auth_key: 8683467532811d66f0425a128b09444e65039a9117b1a6576f92dadf808
c2f4452bbf8d6a06ab2402252aa373417d428c1be2be215f92ea540c17374f23de7f0
prk: 44d67b8c3f03ca205a51628bcbf41998a9fe4b4f97a3d29d5bd8bcbda03dbe5a
303d536583d7d6c960d57f9d7c311b61a948916f85d8f69a4068755418608c79
pseudorandom_pad: 9f64c761fc6a52010cab3598d1a0427300942c600646ef48214
bb0742003dd9ac788b5bf931edcf813ee2706fcc9d97b
envelope: 0122bc649eec25874f87e625008d252fceffbb8b434327914db6c09d07b
7f3bf196cab6723fcea99fe229f042413ffc2c709511fbd27d4a23f9df6a0f3198a6a
7f448e0ddcb4743bb85a8f766458df0c73f415cbb11c8336491fa796488c4df057449
b8b2ce1f05a32ec00168d5a0c9eaa049e2b70fac847a2f4b48006c98f662bf97e5900
edee521388deea91dc57a877
handshake_secret: 1c668baa9ed83178a0cef4c16dcc7e0e86932607f0c913af336
2c639c2532ffadcda40c94f94efa66fdac4cfb59185c17557dae088d344b616834c89
c5d700bb
handshake_encrypt_key: 3df8e9afb60fd6915bf2ef9cf909e7164acb9113d327fd
d024392a1ca45a0b8e7d25ec4255b8e870caaa5673b60905feceadd897519934772b6
577bfc415359a
server_mac_key: 1fee1495e257b9f4369874cd6a58c9f6eb902dbef8469b82b398a
b3c0f0b5d70101704a71a1087e4ac3e8d19b92d111db2fdde95b20ee46971d8b91d8a
b042c2
client_mac_key: 22cc03b0a2f2cda7112afb5714af9e3b84924237855a23cc64bc2
2c00e401bb29e507d7824ed0640c6c05b295c93172284a266aec2aeb01a1c05941f28
e28daf
```

C.4.4. Output Values

registration_request: 02fc4b3addc3978fba0bdfacc4fc662bc8af59e00392b0b
6b5fad9a5d6a60a015b4a0e2d1c8e2f95e229fdbdf50ab93a7f
registration_response: 02ba75a6a537e88c57e67208566dfa0193387002d3028c
a8cb0a2c08c2880f1ed91335a6c289d1620feea05b6243b181280251f78cbd5c7a3fb
f4cdaeb755eb8cc4159edb0ef38baebb503dbefed5c89c14f7c2b99ed242b3d1de890
f7515bad94bd
registration_upload: 02c86369d6eae0978bdd4030b43e0619ce46ea9d91fa6e0e
7575bb12aa4857db98b952d8af9d92f75899c49d0d18793c1e0122bc649eec25874f8
7e625008d252fceffbb8b434327914db6c09d07b7f3bf196cab6723fcea99fe229f04
2413ffc2c709511fbd27d4a23f9df6a0f3198a6a7f448e0ddcb4743bb85a8f766458d
f0c73f415cbb11c8336491fa796488c4df057449b8b2ce1f05a32ec00168d5a0c9eaa
049e2b70fac847a2f4b48006c98f662bf97e5900edee521388deea91dc57a877
KE1: 02778d77bae1e5e05311469840b632fc724f55070922598457dcb06b22f8fa87
d6ba7886fe34283d8727a1e1d30251a5c9af0eb9b48c985e2ba859ce089fc342c648d
dadfca4735f5b2e631eea7fdf967b000968656c6c6f20626f62027044d31354b5b985
87d7d144526f6c8528317cc3e9675c90a1952f4bc2725a6a154f59aed10e4ff7ec68d
4917d74122b
KE2: 03ad08056e57dc6424c6210d7e12801ec7de62e2de9decc6f034d000dca821ab
aca9d733e8807d072bb8c211c477d27fc20251f78cbd5c7a3fbf4cdaeb755eb8cc415
9edb0ef38baebb503dbefed5c89c14f7c2b99ed242b3d1de890f7515bad94bd0122bc
649eec25874f87e625008d252fceffbb8b434327914db6c09d07b7f3bf196cab6723f
cea99fe229f042413ffc2c709511fbd27d4a23f9df6a0f3198a6a7f448e0ddcb4743b
b85a8f766458df0c73f415cbb11c8336491fa796488c4df057449b8b2ce1f05a32ec0
0168d5a0c9eaa049e2b70fac847a2f4b48006c98f662bf97e5900edee521388deea91
dc57a8777e4c5d3a4f779d5966cc8ca367e47e0716ad0f33aa100fe68507920a1a6e3
578035684e8cdae6ce360c2d3f41e4b059f34f986e92dd50c8255f7d7dc0c16252ef9
b8c9fb9e0c2846053355e7fcfa46781f000f51b16b11ae1a74e076f8ceaac5a7ec5bf
578645dbf57581437cd48dfe01f36e169053188d159bd9e23ec7af8d31be741472b40
6d7618c9012d37d8a33fe02b91aacbbab0b19d5b78955e5d3c76e5ae
KE3: a33a58dc32f246eb495e5645688b51662bd2615a254089f535a3262b6a4073f5
3c79c7e08de7e551f8d5555e15e2aa72472b07fe6f1899712e34c60ecc36141b
export_key: 1a9cd58039b5c8dd9c17c9477c6bfa45ce5a92a097edf99cc760ee71b
6766c01b7f96e53d95fd13f0374e066c4fb5b3b08cc49e3db408ae5926cbe43c391fe
0c
session_key: 868df74aa4b3e5d4bcb6a5ef85ebfb1070a5644b0fb3b6450cfcb425
d759df9d210071d5b2cc407379935fee1e4a014b555fe4fdf92e553d415354188df8d
6ff

C.5. OPAQUE-3DH Test Vector 5

C.5.1. Configuration

```
OPRF: 0005
Hash: SHA512
SlowHash: Identity
EnvelopeMode: 01
Group: P521_XMD:SHA-512_SSWU_RO_
Nh: 64
Npk: 67
Nsk: 66
```

C.5.2. Input Values

password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: ccb5f7f7e31e8295587ef5f9f90fa475de567165cf6482e1f5579
583d0d7c54f
client_private_key: 004274252f29a6e19915f81b0c7dcea93ce6580e089ede31c
1b6b5b33494581b48678aec1d0c3d16afd032da7ba961449a56cec6fb918e932b06d5
778ac7f67becfc
client_public_key: 02000202cbe2dcdfe616ffe600cbe24768cdba3066d53d2b58
feffa43e199c833f85963a612b79a2fbfb065f34e2edf51e39ba3db9cbdec0fffcce0
113a5c05b2c28b1
server_private_key: 000739878b22e5c4833d34c486a8510e7cca4c1b81ece04f4
7e8d2554a5ebd83679b4c1e67ed82f2891751aa7094602be672c324929abb1876a7f7
165ac7ec79bfd6
server_public_key: 0300159aed22eed3a1ca9e7a8b063b1b62c3a48b00b7b83edf
6047defdb1b05e14b14faa77afb5f08ffaa04cc8c5df59983e42677f7b6c8d63c0348
15367374543ed1c
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: b18ed353e7c876de386bfab5f549f6c8e8dc8ba0f63cb32fe4d67a6
eb3556b8e
client_nonce: 6303dc1f00a19ea8ce857f9c2156702c56303a53d5ecbaa35f5a3d5
47e85b924
server_keyshare: 02003fe52f2e417b5c448309d4a54363aa84725feff99da4639f
caa9d60e5b1bea54a7f5661b05bd4d3a662d6688a76f177c63d1f1a548ff00d5d6c6f
b331b56e1b890
client_keyshare: 0201a487b80b1f7927e7e44eb0cea2498ce462ab21fcb4b8e987
27aac9e91dd8f8e22338953560a0e088934084f2a27b85b31a3ba55b73ff9a0d17333
4667996078929
server_private_keyshare: 0134471b1e6eb4b043b9644c8539857abe3a2022e9c9
fd6a1695bbabe8add48bcd149ff3b840dc8a5d2483705fcc9a39607288b935b0797ac
6b3c4b2e848823ac9af
client_private_keyshare: 002d816be03432370deb7c3c17d9fc7cb4e0ce646e04
e42d638e0fa7a434ed340772a8b5d6253d35895f4cff282d86b2358d89a82ee6523ef
f8db014d9b8b53ad7b1
blind_registration: 003f281099ef2625644aee0b6c5f5e6e01a2b052b3bd4caf5
39a41fabd4722d92472d858051ce9ad1a533176a862c697b2c392aff2aeb77eb20c2a
e6ba52fe31e13f
blind_login: 01c28b45f65717a40c38f671d326e196e8a21bf6cfd40327a95f1ccf
c82a9f83a75dae86286729214a1ba9a359ab01833477b8cb91932d0c81667a0e3244b
896ac15
oprf_key: 00363749be19e92df82df1acd3f606cc9faa9dc7ab251997738a3a232f3
52c2059c25684e6ccea420f8d0c793f9f51171628f1d28bb7402ca4aea6465e267b7f
977b

C.5.3. Intermediate Values

```
auth_key: 5beea85089b8a18c21ac2e4be507a181faf3dbf07624b1e4180c67b1639
932d477388c9a6254c1f8eea4e6594aa0719a824add4d474472d5c9a2931f95b7d647
prk: 935f5cbf45cd48d365b20a1ab637fe31b9fc771b70af288bbe890e0b015b2225
7380360eaf8a5a126f055fa0f68246877f18bcf5269e41746cf671b3c2a254ca
pseudorandom_pad: fd878bcecd07a5b599b07502d12c1386728e81576c5351b96f0
36919b8edb84f7c7f3251575735b329af90a72d2ddc3c544e5b9cf1812cef6040a9fb
7bbcf9e70e2a
envelope: 01ccb5f7f7e31e8295587ef5f9f90fa475de567165cf6482e1f5579583d
0d7c54ffdc5ffebe22e035400a58d19dd51dd2f4e68d95964cd8f88aeb5dcaa8c79e0
543418b8bd4a5b08a5867fa27d5684bd78ce18955a0a10a27c4b467c8cf17b0f9ce2d
6d11f62e0de7801c4adbd6963b5a2bf64c9d157139f97c7556f68e1942a98ddc92adf
b5a1b26607504d1614b0c864ec61c3b80d7659c91a7cb68eae3516294053
handshake_secret: ba997ccbf434ae93cee1dcde4df3a8302db1e650ba4375acbb7
6715e63e91acca5d9c215425946e90004194832ba1078b50873f73b23af779d4ede26
9ecb491b
handshake_encrypt_key: c4324223326aa611bd0eb147f16e47ea8dccb2f4360a53
d332a5e3bd95a6812b0fa8e6e15997447b08b094a22856e44e23d0c1536b1a5824a70
f0bcc1ff4e752
server_mac_key: b782fc84b81fd9434f4e5339b33bddbf23ad0220d6dd42f2980f4
7ce31f59b5127a023591741e8074eda2175fbc298f1f5138d9e06789e7d179190178a
04f901
client_mac_key: 786e739a91fd527e24d2fabaebf862c7d8d4ca067bc365526b152
cf5de9a7aafea44e5f223c0e5a73cf9c551e19e05356f606fa2d10acd46a2a1239d73
9ebd49
```

C.5.4. Output Values

```
registration_request: 03016af598df5549c18c7f904ff395006449477bd594663
b2948142db6d2aac90d204900d669b5e73cfefdc91d7bee857d9522eb996601d2c3f8
25221ed46f51c89ec7
registration_response: 0201b7e055f71ebc3020873cd002596dfce243891fc7f2
278ab1c5bf768886067e779ce4d922dfaa87e3dae9ed4d1ba2bd19ee24c2f33f3f19f
89d5b0eb6865880659c0300159aed22eed3a1ca9e7a8b063b1b62c3a48b00b7b83edf
6047defdb1b05e14b14faa77afb5f08ffaa04cc8c5df59983e42677f7b6c8d63c0348
15367374543ed1c
registration_upload: 02000202cbe2dcdfe616ffe600cbe24768cdba3066d53d2b
58feffa43e199c833f85963a612b79a2fbfb065f34e2edf51e39ba3db9cbdec0fffcc
e0113a5c05b2c28b101ccb5f7f7e31e8295587ef5f9f90fa475de567165cf6482e1f5
579583d0d7c54ffdc5ffebe22e035400a58d19dd51dd2f4e68d95964cd8f88aeb5dca
a8c79e0543418b8bd4a5b08a5867fa27d5684bd78ce18955a0a10a27c4b467c8cf17b
0f9ce2d6d11f62e0de7801c4adbd6963b5a2bf64c9d157139f97c7556f68e1942a98d
dc92adfb5a1b26607504d1614b0c864ec61c3b80d7659c91a7cb68eae3516294053
KE1: 020142931e5e870e35226b46f8a9692babfabede9ca86ffded305ba079274920
aa78f9a45341b6693765e601237d6a6bce8ddf194f6161144e9a2a1bcaa5860e6637c
d6303dc1f00a19ea8ce857f9c2156702c56303a53d5ecbaa35f5a3d547e85b9240009
68656c6c6f20626f620201a487b80b1f7927e7e44eb0cea2498ce462ab21fcb4b8e98
727aac9e91dd8f8e22338953560a0e088934084f2a27b85b31a3ba55b73ff9a0d1733
34667996078929
KE2: 02000c9ca900d4b470d043136562cd7d9debe13d6595a274e46a1bd9a7a2e3d6
19f6a7cc0324f30fc8fa10f8eefcb7e968c6f2e50cf0e2ee3487d9a80cbe24255600e
b0300159aed22eed3a1ca9e7a8b063b1b62c3a48b00b7b83edf6047defdb1b05e14b1
4faa77afb5f08ffaa04cc8c5df59983e42677f7b6c8d63c034815367374543ed1c01c
cb5f7f7e31e8295587ef5f9f90fa475de567165cf6482e1f5579583d0d7c54ffdc5ff
ebe22e035400a58d19dd51dd2f4e68d95964cd8f88aeb5dcaa8c79e0543418b8bd4a5
b08a5867fa27d5684bd78ce18955a0a10a27c4b467c8cf17b0f9ce2d6d11f62e0de78
01c4adbd6963b5a2bf64c9d157139f97c7556f68e1942a98ddc92adfb5a1b26607504
d1614b0c864ec61c3b80d7659c91a7cb68eae3516294053b18ed353e7c876de386bfa
b5f549f6c8e8dc8ba0f63cb32fe4d67a6eb3556b8e02003fe52f2e417b5c448309d4a
54363aa84725feff99da4639fcaa9d60e5b1bea54a7f5661b05bd4d3a662d6688a76f
177c63d1f1a548ff00d5d6c6fb331b56e1b890000fb9963a4ef0da2bd75360ae592d5
f4bc54b8cf32d0748333345ed7f75ca2045eb413c7a9b844ca9f5ecb23c0385108623
211a8b89c613912a54d48854c0b4832829bd718a890cc72334ddaf2f18e87e
KE3: a1765707136891b2d76da4c47e8c883729fcba6786b91b3a11d682c98a9cc50b
b7ea8fbd8304c030d414506818b315ad8a990f0363ce7c1f5e8af5aaa66571c9
export_key: e80808e17d470f24478de9471e06fc0c7a1aa1e98af711d2f8424fba9
fcd8ed30dc2a5894813e4c2f9a650ab07bedb9516b4c87a995afe3387bcb8a6a7499c
76
session_key: 61933b6a3227a5fc10ac30b2186d91fa6095e42768426b74663b75f9
21b19556e24ce7253510a204a7f86aa0ba3e37c0b4dea2fd422bcb4a6b385416ff364
15e
```

C.6. OPAQUE-3DH Test Vector 6

C.6.1. Configuration

```
OPRF: 0001
Hash: SHA512
SlowHash: Identity
EnvelopeMode: 02
Group: ristretto255
Nh: 64
Npk: 32
Nsk: 32
```

C.6.2. Input Values

```
client_identity: 20fa92f2e4b7ea5b5e677ac4930ff3b93b0043481ab70bc613b2
e16a6dde6b05
server_identity: eae9dfa6b8348d34418c32d385e1eac99efbce1af320901f7c8e
de8d6d272c65
password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: f41e8b3c5a999aa946f9b562a150e5c5e36748a31a79feb241809
0438877888c
client_private_key: dc70a99bbabf1ebe98b192e93cedceb9c0164e95b891bd8bc
81721b83d66b00b
client_public_key: 20fa92f2e4b7ea5b5e677ac4930ff3b93b0043481ab70bc613
b2e16a6dde6b05
server_private_key: 709687a36c94592ab76579f42ce1be6961f0700496e71df80
6ebd5320554720d
server_public_key: eae9dfa6b8348d34418c32d385e1eac99efbce1af320901f7c
8ede8d6d272c65
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: ef49d83cef5f1411ea30abb82b08bd85423aadb86e2c19df5930b3c
8498b9f97
client_nonce: 4ab1227db632bc079f79c0f5279df2dfa75cfbd4434ab40dcf844d6
77165cd3b
server_keyshare: 96a9587e233e67f2397f10fec6355b68102534f1f1b115b4ddf7
485840efcd7c
client_keyshare: 54f35db3a52fb0cf2a97918a6987993231d227e28711eaef19a3
e5033632611a
server_private_keyshare: 6650d64df70618a878504ce73dcca27b1af125c67e48
1e7bd49d0b24709b200f
client_private_keyshare: ebb01c59f99bc955df622548e247f7ef180732909ff3
c5f87ff8c7867b8be704
blind_registration: 308f1d3fa1fea402f3c90b04601274050a3c6f467387c2f48
878823949b0e109
blind_login: 141e21373228a44b09d4c00da9a6bbaf9a5e54a1687c07f327833643
4245510b
oprf_key: b7126967aa0cb69c311b71343843ea041bae30e2bde41b548b8fbd8bced
97604
```

C.6.3. Intermediate Values

auth_key: 9f761a5a56a74269b382403aeba47c1d24b9e200e1839efcb616fb280da
1b6ba7bd71d6455dd3cd979c545608cfbfc1c4e9ba677e1d40848054a00696c4b2589
prk: 30bc3f37a757890ac17ce46043f3c5ed30c96fb8743205e77e84dc167d98e114
6093150ff7b4d002f793bfe717e88d174ed2669abdd9e96af473a7ac82973b0a
pseudorandom_pad: b909f990c94b9c5949a2b8d0874d602f846b7981b331fd79978
b530cc46c8670
envelope: 02f41e8b3c5a999aa946f9b562a150e5c5e36748a31a79feb2418090438
877888c6579500b73f482e7d1132a39bba0ae96447d37140ba040f25f9c72b4f90a36
7bdb425fa1dd4c49e17780f33b821e1e019668fe7f45520e26996ac8cb08e3d2566cc
439c83030464effecb8350e7b1ca31087d87f6a45ed3910c185a24a89d282
handshake_secret: ff89f264f8c3974238f4c8d736af7b0a55f2e4edc487cbf3e5c
7b4bbf21acd7c1d28354c2c8555fba57c4d4b1fbb4b772bfdf909881f67dd517cc9f4
f6ebaeac
handshake_encrypt_key: 3071b181f639062cf70b74d0ffe5ec8fa695da13cd2f00
e74b8b7ef348ae7a5df9c3a32c9f7aeaad5a28379712cf849b9707e221dce124abfad
d0225a8e8e045
server_mac_key: 7c37b344d189cbbeff80bbb4b78e2703d1a80dc28239923094287
62f7ce2a93b11f6e85dd45c02809afe8583d4aad6377e72788773af92eef33c690692
20ae76
client_mac_key: 2c4aff12ba7aa911a51f9e5b7a7c01439d854c97e4b8ec842a9db
d78345760328fd5a72424e49e25ec8fe1b6d9d42f774516f400948bd5a105d995d000
2fc83b

C.6.4. Output Values

```
registration_request: 3c8b89966e261a5aaf7aeb6dcdd94c87ce311bf197221b8
7ef44632d58f18a05
registration_response: caf9243d7ef3e267815632bf79c85a27a23f218a438815
2a523f6a310949807beae9dfa6b8348d34418c32d385e1eac99efbce1af320901f7c8
ede8d6d272c65
registration_upload: 20fa92f2e4b7ea5b5e677ac4930ff3b93b0043481ab70bc6
13b2e16a6dde6b0502f41e8b3c5a999aa946f9b562a150e5c5e36748a31a79feb2418
090438877888c6579500b73f482e7d1132a39bba0ae96447d37140ba040f25f9c72b4
f90a367bdb425fa1dd4c49e17780f33b821e1e019668fe7f45520e26996ac8cb08e3d
2566cc439c83030464effecb8350e7b1ca31087d87f6a45ed3910c185a24a89d282
KE1: 8261a1efd78bea73faf256a23c200d729259886530fa43b875c1ca124b09bc7e
4ab1227db632bc079f79c0f5279df2dfa75cfbd4434ab40dcf844d677165cd3b00096
8656c6c6f20626f6254f35db3a52fb0cf2a97918a6987993231d227e28711eaef19a3
e5033632611a
KE2: fa1f33a43a03123ebe35345ef93aa23b57ea8bfbee7022b05a179d60768ba02e
eae9dfa6b8348d34418c32d385e1eac99efbce1af320901f7c8ede8d6d272c6502f41
e8b3c5a999aa946f9b562a150e5c5e36748a31a79feb2418090438877888c6579500b
73f482e7d1132a39bba0ae96447d37140ba040f25f9c72b4f90a367bdb425fa1dd4c4
9e17780f33b821e1e019668fe7f45520e26996ac8cb08e3d2566cc439c83030464eff
ecb8350e7b1ca31087d87f6a45ed3910c185a24a89d282ef49d83cef5f1411ea30abb
82b08bd85423aadb86e2c19df5930b3c8498b9f9796a9587e233e67f2397f10fec635
5b68102534f1f1b115b4ddf7485840efcd7c000f7ebe71d4ab326006a3aeca802435d
c995a38ac6662221f974cb920992d82b8ef8d147c77e29b628a82b5ccb01ea2f7bb60
af94cd1860e1bd974a11a1c9bd827789f663c4758eb71058c244138de0c2
KE3: d81f93397cdba85a43993d4d9afbdc67f147adfa2b223213b19692cb820eef48
5073eda4c8236b2f47702404ad60d9a875d189626fc7b7cc861825385470ae54
export_key: 03192555940b5b42e64e6200bf55cc701f1bace3d402a2f8d83977843
51a1e3fa1f07a471b783b208acb1d92be47903b6fa3a0df9f4d4b7956ee4f431e2950
f6
session_key: 58a7fa98bf3b7b52da21406abfb11d98734354edd47d7b32462c0513
f0617c89824ea6031d4147a86fc9f6c6837ce640c12fb937d764f296d1a9421ad1b2a
5d5
```

C.7. OPAQUE-3DH Test Vector 7

C.7.1. Configuration

```
OPRF: 0002
Hash: SHA512
SlowHash: Identity
EnvelopeMode: 02
Group: decaf448
Nh: 64
Npk: 56
Nsk: 56
```

C.7.2. Input Values

```
client_identity: e0f6146168d32f4b68e042ed5d5608d1108e84a08b6688798ead
0810b4f10d7a91f0767e197e946ebfc487bf62a5ed5684e7ab9137ee1862
server_identity: d49500848e7c06c8a5dd5bda74930ffd20fbef9a2de24a0068e5
bf3dc356852b10327be9803983271450bc6a8c683abcdd73883ee63543e9
password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: 4c0ccd88c12f90364570c5ce356f457a762bbe590e27d6208ebc1
9332d37605d
client_private_key: 0a45c82e38a6ab6fb99ec1df7423e25a7851f9558e7a05166
03c9d0201b409c3fd0f0fe78bd37bf60927fafeca73ed8093538a9992c62235
client_public_key: e0f6146168d32f4b68e042ed5d5608d1108e84a08b6688798e
ad0810b4f10d7a91f0767e197e946ebfc487bf62a5ed5684e7ab9137ee1862
server_private_key: 64666faa068e5ff9e00d588446b7d6cdc09ae8df069b30987
a2cdd39286e0481a2eb899f4e0db672264527a8115f176c53709a4f6534f328
server_public_key: d49500848e7c06c8a5dd5bda74930ffd20fbef9a2de24a0068
e5bf3dc356852b10327be9803983271450bc6a8c683abcdd73883ee63543e9
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: 975a2b5d6318131b85d7f2f47ea0b7edef8f33a1be69f09307604c1
3b7afdabd
client_nonce: df3b22899746ff552d61851fe5c2b5ff1e004fd95dfbcba1906ade5
fd38a70be
server_keyshare: 720df48b9e395c40448cf06a80bc9c1d90ae1a3080e4ba3368ca
056ba92921f79d67073fb91b7c94c3f3d33cc01613d7221f54fecd80bcbb
client_keyshare: e6f2ab22efb569a19f48355c7d5d4f728faf3a9142d45c20c098
bcbcb6c12fd92480bfb4e532923e64b9620b6bc84ac5f2d8f84ad466fd18
server_private_keyshare: e5d7202d6ed8cc45c4502850974076d720343b566089
1b5ab5655c3defb4b39b35b27ea20eb0bff035e9b9cbae6cfca36aa4827c32abd905
client_private_keyshare: a54c0d7bf4ee396a0e4a3f023b35698aaa93a2be8bb6
32747671b3edeaedff07116afbb5e73cbc273e2e0d0876780343578338425ed81d3b
blind_registration: 9d557ee103479baef585ba8017f7659cdd0b804c093852519
9d88853b52ccfc7802d09cf38ba35e36db24404602da8a616e7ad8f1c05cb36
blind_login: eaadba538bf67207633a956ea71fbd02ea2dbfe7e195dbd26ea562c6
f2406fe1df1367f98f6707dee9b2e3ebd9842b0442e25d086e099231
oprf_key: 98a5689edb98ed3424fa5c8584423c6b047121fc36fcec934c8ad24a98c
86d00e1e1d6d3d923a46519065977331abaa1e3c0d86591458b25
```

C.7.3. Intermediate Values

auth_key: 7115d4b507621a9885dea1c09034a0bf258d9a04bb57e11d450a5d9956c
9810ef8f243225ad387c8c6d2511be920ee5cd66c79079509e6d132fc6fc8f4238f6c
prk: 296432fda62b3ca77c10e38952b4c18e9e12164f883e9fcc4f2723c37fa3b8f3
6c4f4a4ca009aefd1391e3a0a36fc05580c98c55a17623dc593b9d18e3babc25
pseudorandom_pad: d59012fd3a30fe215ede1b55ba2106c28f4a5f87dfc29b6525a
b3f41b4ffe552051bca9553b9b3149b06e3723d4d862ce4d6ede1e4b77ecb
envelope: 024c0ccd88c12f90364570c5ce356f457a762bbe590e27d6208ebc19332
d37605ddfd5dad30296554ee740da8ace02e498f71ba6d251b89e734597a243b54bec
91f814c572d86ac8e29221198cf73e6bac7785677876715cfe105031ce6b536b86424
6d9b2a03acbdbe1a013b0c153e3e5ab18d601c10475f6d137d950f50b1a716bd41d16
81932e6ae211451b7759b22d46487c707508ce7c
handshake_secret: cf4fc0c1cb9fc5b4f97ae047dfc5269d3b9013146f5a5e5ee93
1826a1e94465bb5079c4d65da5187c6b89be8bc276d54dc0968ef2bd9c33d4e06e485
5a571529
handshake_encrypt_key: fdafb8a372b21fbcc478a3986dca5f1eb24060b6e63cec
96d48edda13e54b9cdf7730f49fd16c67bfe90ccb89621414c030049b22214dfdf093
69c7b96f6757a
server_mac_key: f54ed72de90635c20994822097039b2a4e7349c806738b716105a
80abdba51a4d8d36ea015713b461402e1ff6f7fc26f8bb0a4b889111ba08da2b7cbc9
55effd
client_mac_key: 841b179f9f3bad87574cc8181008f61905d5f1924f40468aac081
5f7592216c0450a657ddb160f5d3484d77c186b9e9948fbe9e98bd0634b432a39b733
d19461

C.7.4. Output Values

registration_request: 90cca9013769f28f3992f77a043084edbfe6c89b7e2305e
4a6765e50df565fa8e18aba470238c6ed7992af20e962a641bc6bd678ceba640a
registration_response: e45f47bc6f41bb6de778775aa3f746b31cd17969183beb
e8b3757ba8dd546534996ccc686709c1a0fd6ecb4313936940a470333d1ae3c70ed49
500848e7c06c8a5dd5bda74930ffd20fbef9a2de24a0068e5bf3dc356852b10327be9
803983271450bc6a8c683abcdd73883ee63543e9
registration_upload: e0f6146168d32f4b68e042ed5d5608d1108e84a08b668879
8ead0810b4f10d7a91f0767e197e946ebfc487bf62a5ed5684e7ab9137ee1862024c0
ccd88c12f90364570c5ce356f457a762bbe590e27d6208ebc19332d37605ddfd5dad3
0296554ee740da8ace02e498f71ba6d251b89e734597a243b54bec91f814c572d86ac
8e29221198cf73e6bac7785677876715cfe105031ce6b536b864246d9b2a03acbdbe1
a013b0c153e3e5ab18d601c10475f6d137d950f50b1a716bd41d1681932e6ae211451
b7759b22d46487c707508ce7c
KE1: a8f3c6290c4a31a2f696ac5c4f933c85dc3a8fde4247e8733fae3502b9f895ed
40a43c53547891e0a6305a12f7bbbed8696a774f2f352b1fdf3b22899746ff552d618
51fe5c2b5ff1e004fd95dfbcba1906ade5fd38a70be000968656c6c6f20626f62e6f2
ab22efb569a19f48355c7d5d4f728faf3a9142d45c20c098bcbcb6c12fd92480bfb4e
532923e64b9620b6bc84ac5f2d8f84ad466fd18
KE2: 3c86de1f2ef35f9044b08421334c2ea2020300a3c5259bab2fd525dd4e68e03a
90d3460479849e12847a3600cf2428a4c424741b96f76aa9d49500848e7c06c8a5dd5
bda74930ffd20fbef9a2de24a0068e5bf3dc356852b10327be9803983271450bc6a8c
683abcdd73883ee63543e9024c0ccd88c12f90364570c5ce356f457a762bbe590e27d
6208ebc19332d37605ddfd5dad30296554ee740da8ace02e498f71ba6d251b89e7345
97a243b54bec91f814c572d86ac8e29221198cf73e6bac7785677876715cfe105031c
e6b536b864246d9b2a03acbdbe1a013b0c153e3e5ab18d601c10475f6d137d950f50b
1a716bd41d1681932e6ae211451b7759b22d46487c707508ce7c975a2b5d6318131b8
5d7f2f47ea0b7edef8f33a1be69f09307604c13b7afdabd720df48b9e395c40448cf0
6a80bc9c1d90ae1a3080e4ba3368ca056ba92921f79d67073fb91b7c94c3f3d33cc01
613d7221f54fecd80bcbb000fd48223a7b4d19871dfe19b5c5be686edbce5cec2fdb2
c4aa9bbace6eebedbdee52e0ebee8f1053dc59dd66c11b34fbcfd7e681afe5f27ebdf
683b6af777105fa8e57936da294f2d9e1ce090418608a
KE3: 964f3993d57498ceb543937735462b830a559d4efdaa47e26411acb71a173428
0f15cbbdbd4da4b6567086aaa7c3983dfc0eb711b7d8b7a877fd3f18d2d28a37
export_key: 6bbf975f133ea322e181c98ecdbf1837bfa9c7d69d226675e0af8bae8
96eb911beab703858bdd790c9e1b423bddbd35fa9eebe6aed8418223737060bac0985
5b
session_key: 5bfe0bb35b1c430f294cf847f34f0c9f9e777f2888b667ed3711d9cc
db0bd5505d715d8371fda04598f1cb06c00ae5aeed5ed1145a86df58513c9fd104460
c09

C.8. OPAQUE-3DH Test Vector 8

C.8.1. Configuration

```
OPRF: 0003
Hash: SHA256
SlowHash: Identity
EnvelopeMode: 02
Group: P256_XMD:SHA-256_SSWU_RO_
Nh: 32
Npk: 33
Nsk: 32
```

C.8.2. Input Values

```
client_identity: 0227aa37ade0cd6231bd385333cc8ccdf3872e75d9f6506192ed
7bcc6e5819f5d7
server_identity: 039178fd762b694fc67cc2df224079dd59ccd00d22621929a0a7
e5ecac96814260
password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: 96bd3e8e5a8110e4c57508b0d47d20509639b30b3726264a0f957
937f33f5fa4
client_private_key: dc970d63acb5ab74318e54223c759e9747f59c0d4ecbc0873
02667fabefa647c
client_public_key: 0227aa37ade0cd6231bd385333cc8ccdf3872e75d9f6506192
ed7bcc6e5819f5d7
server_private_key: fcd9a655f77ff0b2ebcfe21e1a1ca4a84361e9f1b18e24c9a
40ed5eec262bf52
server_public_key: 039178fd762b694fc67cc2df224079dd59ccd00d22621929a0
a7e5ecac96814260
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: 3575160dbf02cbaddb376c3512c35e83e46b224d7d289fcd67d401e
730b3ec58
client_nonce: 24f0d0e0b3e2f8c0d5d561ffba8c87c88a085623fb61e9581b45069
b4ec032c6
server_keyshare: 0233c66a7b8735e362aa803f2e5677f48f4cab048edd74381667
82858e46381fd8
client_keyshare: 021ed435f437b88157c9af824de01811f5afc20de9dfdb49f0b4
454ec1df60af60
server_private_keyshare: 4fb9234e93a8bd048ad9f44b428026396a810328c405
a354e666f086fa0ea476
client_private_keyshare: 4fb56527be010296ea880e1c6a4dbbc9ede543a2ad0f
83fd60fdacb59801a9d9
blind_registration: a54f137dbe5f5eb4a34dcb73609c6693f28cd3d57ed77bf66
e0ab7d86c6990f1
blind_login: 3b5d1c0fc0812c10e18f146b14d7eb94755a918bac1ef8d69d21a7c1
3f95c9b2
oprf_key: 334d8af16ae1e69f5adc24e5aa89ebb63637c835fd39b17a1a4453eb5d9
63d23
```

C.8.3. Intermediate Values

```
auth_key: 598c8679dfc4693f8777d9ea209dffdbc3e8a6fd55fb11e42f197b2b606
3752c
prk: a60920b28eb24d0f12d996c42de3e94f6c79ab2532b1a633da85065037e57e5c
pseudorandom_pad: 9e2987091f68f908bc9e7eb0030b01d1e5498d47cde3f0ece37
88a673e6f9927
envelope: 0296bd3e8e5a8110e4c57508b0d47d20509639b30b3726264a0f957937f
33f5fa442be8a6ab3dd527c8d102a923f7e9f46a2bc114a8328306bd35eed9d8095fd
5b2499abde5be91da24191482677267a9833c0adf6731f04c589d3b305b2530bea
handshake_secret: c14fb3f53f5a518bd37a3bf670cc38455e26b7bd1e44d7567b5
747ef10f1a5b5
handshake_encrypt_key: 1912daa7086e5e0dd610058d30877d907b6f4888664bf4
74d84e561afaadb127
server_mac_key: baf03e219ba8cee27b26a5a0f12cc15ec809f8e23e8ccef0862b1
1d515bcc4be
client_mac_key: 08acb367302b3c2b5aa405954a8985f84f1a5b491daa77f58c20b
81e97e73bac
```

C.8.4. Output Values

```
registration_request: 02327e93445af116df70f57d18ab2a0ef9f492aa3d76c94
6d98260fa1edfa5b832
registration_response: 03480ac383e647d1f78b19bd902c7126024ccc76da605d
cc416581a32e4202d62a039178fd762b694fc67cc2df224079dd59ccd00d22621929a
0a7e5ecac96814260
registration_upload: 0227aa37ade0cd6231bd385333cc8ccdf3872e75d9f65061
92ed7bcc6e5819f5d70296bd3e8e5a8110e4c57508b0d47d20509639b30b3726264a0
f957937f33f5fa442be8a6ab3dd527c8d102a923f7e9f46a2bc114a8328306bd35eed
9d8095fd5b2499abde5be91da24191482677267a9833c0adf6731f04c589d3b305b25
30bea
KE1: 02dad65138d90eb5fffdd93d1ad84b7e86e5b3f1964756d092e154d6a135c6e4
ce24f0d0e0b3e2f8c0d5d561ffba8c87c88a085623fb61e9581b45069b4ec032c6000
968656c6c6f20626f62021ed435f437b88157c9af824de01811f5afc20de9dfdb49f0
b4454ec1df60af60
KE2: 038782ee7c0ca885bd49d7105f9f43f89d34b2ad39b98e02a1a4ceed9e7de3f6
6a039178fd762b694fc67cc2df224079dd59ccd00d22621929a0a7e5ecac968142600
296bd3e8e5a8110e4c57508b0d47d20509639b30b3726264a0f957937f33f5fa442be
8a6ab3dd527c8d102a923f7e9f46a2bc114a8328306bd35eed9d8095fd5b2499abde5
be91da24191482677267a9833c0adf6731f04c589d3b305b2530bea3575160dbf02cb
addb376c3512c35e83e46b224d7d289fcd67d401e730b3ec580233c66a7b8735e362a
a803f2e5677f48f4cab048edd7438166782858e46381fd8000f7dee7c8d4df469aec8
69d6443711489cd6de819ad1ed162f6813c220f9171c66a0b3695b2c43e20c0fee072
1fbad32
KE3: 731075af5acd0195c6f0faa7608845a51aeb1d7da7f1feca8cf2b6665d17f8ad
export_key: 0b157ff3684cb28f35eda2b0495c0bf945d41977d25380ec662b38034
c3a8ddf
session_key: 3d2e068949a1d940528f3e46f435bb1950347c3e924b6332fc4a729b
34eac1ca
```

C.9. OPAQUE-3DH Test Vector 9

C.9.1. Configuration

```
OPRF: 0004
Hash: SHA512
SlowHash: Identity
EnvelopeMode: 02
Group: P384_XMD:SHA-512_SSWU_RO_
Nh: 64
Npk: 49
Nsk: 48
```

C.9.2. Input Values

client_identity: 0368f5bbaaa438e2e87de012dec549a4a89a6d4deb262b133834
d1d90ed3eeceb12a2c5cfd5702077bfb47b0e36e48904d
server_identity: 024ecf37a198ab5431962c820df129c60356bc801d3584da5ce1
19c15554d0183a3b9a6b833cd2a019a882c620020c8a3a
password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: 54fdac1c3677cbf0c388eed24e5a425c1b616f035bb29aa445cda
d356151e700
client_private_key: 26fec54d4567adabd7951ad51ea3741feab175ac5cf7fa02f
3ad744eb5baf418275e45ab31ade30669dbae98fb087953
client_public_key: 0368f5bbaaa438e2e87de012dec549a4a89a6d4deb262b1338
34d1d90ed3eeceb12a2c5cfd5702077bfb47b0e36e48904d
server_private_key: 8588213957ea3a5dfd0f1fe3cda63dff3137c959747ec1d27
852fce42d79fc710159f349e7da18455479e27473269d2a
server_public_key: 024ecf37a198ab5431962c820df129c60356bc801d3584da5c
e119c15554d0183a3b9a6b833cd2a019a882c620020c8a3a
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: b109f9b77755f74bbea8cda67e8e2d4829a8f684272a10758525705
190d06a72
client_nonce: 33bf1ddb42dd00682e5cce6c558b387c5f11d8ba41c819d1152612c
33184838a
server_keyshare: 036b981c4a2265b4376e2edc3186a45c2e2f820b1092784f6448
354779578b442b0369640e2f10a856cf8dfd60b28c68c9
client_keyshare: 03526fe798e9ab52137d8c3408b131430eadae49f6e93a4fa228
c0338081e6090f75c2b3d55da4b2abfa4f2e2a52bd3330
server_private_keyshare: eb81ae8c1af769a56d4fc42aef54a703503046d8272e
aea47cfa963b696f07af04cbc6545ca16de56540574e2bc92535
client_private_keyshare: ac475d6a3649f3e9cdf20a7e882066be571714f5db07
3555bc1bfebe1d50a04fd6656a439cf465109653bf8e484c01c6
blind_registration: c044df390ab5683964fc7aabf9e066cf04a050c4fd762bff1
0c1b9bd5d37afc6f3644f8545b9a09a6d7a3073b3c9b3d8
blind_login: f8516f98f3159b3fed13a409f5685928c72d9dab8ddfe45de734ce0d
4ff5823d2e40c4fcf880e9a8272b46eea593b10a
oprf_key: 5e7d38ba6ff37c42b3c4859761247a74d0c62c98ddff1365bb9b82b279e
775b7220c673c782e351691bea8206a6b6857

C.9.3. Intermediate Values

```
auth_key: f8070223ed2f8742430d213ffbefc4f314d2d919e7833e7894ac37bf3db
3fbebb66a88c0d188822e532f2db6db9002979659e14076a019613cd19af38f24607a
prk: 1a7fbae01371f1c2d60dde37b8c337075c60b9fe7ea56a529a8db3e6926aa83c
e729ffd7ad288a2a3d0a3798b5b349370a185e79a11f86145051ceac83f6af71
pseudorandom_pad: f8d7518c0f3dc7f4d0564a3de14e91b945bef48abd1bea93cab
cfc1345f9461c6bea2ea07cedc27ec8717044b66b90da
envelope: 0254fdac1c3677cbf0c388eed24e5a425c1b616f035bb29aa445cdad356
151e700de2994c14a5a6a5f07c350e8ffede5a6af0f8126e1ec10913911885df043b2
044cb46b0b4d402178a1aadedc4d63e989d239d0900f507a8964186f827ea18d23f0d
859974e62913c95780c548a262c125fe8841fdd11453abbd79407dfa3a6ec4aa4da4e
40c4dae1ab44569cfafe7331
handshake_secret: b137c7cfd5e865c88803d041fa45250837242b6548e9cc31751
dcd1f8fecee25708dad91348f106176deee35937681c8bf42bb86403baebdfb4d936e
e0d96219
handshake_encrypt_key: 8efd3f58bd9fe13172dda8ab106d427dc58dae0bf9da04
51f4aef08d051cfa0cbbc42332d4ef506a0442d26c13eb1316969b6c81ba0c764efde
659c099d15262
server_mac_key: 8e6773f21a7bf460c2e4dadfd47fc10357f9fdbfb6a13413167b6
4876489464a157ea128c7c9d8c4fcb81ec50113030a69b88d49698a473584e23423e3
4ed296
client_mac_key: fa75a3be1ae907f18a96d11aa2c96bd5dc2c2332f7f9a7ce086ba
c0983952a96866596a7860ca65b5371bd80eabedaf34b7b54e8977a9c2fb46ffbe1b8
ade1b2
```

C.9.4. Output Values

```
registration_request: 029674b50d9bec795e53084cb5d6e0f4813804ea378a672
e5e0514f79e98055b79eafa67deed65b040dc1368a7216c8071
registration_response: 02da545d424e985f21cfcac7dd74ceca2177e513ebf484
3659160649ab4a0e5a9caeeba5e79c1fe86ebb5776e8bd4873db024ecf37a198ab543
1962c820df129c60356bc801d3584da5ce119c15554d0183a3b9a6b833cd2a019a882
c620020c8a3a
registration_upload: 0368f5bbaaa438e2e87de012dec549a4a89a6d4deb262b13
3834d1d90ed3eeceb12a2c5cfd5702077bfb47b0e36e48904d0254fdac1c3677cbf0c
388eed24e5a425c1b616f035bb29aa445cdad356151e700de2994c14a5a6a5f07c350
e8ffede5a6af0f8126e1ec10913911885df043b2044cb46b0b4d402178a1aadedc4d6
3e989d239d0900f507a8964186f827ea18d23f0d859974e62913c95780c548a262c12
5fe8841fdd11453abbd79407dfa3a6ec4aa4da4e40c4dae1ab44569cfafe7331
KE1: 02ab0cdb1bf7038717c03d583e311f14c6004c73f78383d4cc6248751aa68ca9
29d717dc6f003de949a17732875bd1aa6733bf1ddb42dd00682e5cce6c558b387c5f1
1d8ba41c819d1152612c33184838a000968656c6c6f20626f6203526fe798e9ab5213
7d8c3408b131430eadae49f6e93a4fa228c0338081e6090f75c2b3d55da4b2abfa4f2
e2a52bd3330
KE2: 03ed6cdbc3a3b78a9504aeaa0df8a3ff996ab5b8dbd2d74cfeec3c976c434a85
860d6367df02c62989c8ee9b88a354ea30024ecf37a198ab5431962c820df129c6035
6bc801d3584da5ce119c15554d0183a3b9a6b833cd2a019a882c620020c8a3a0254fd
ac1c3677cbf0c388eed24e5a425c1b616f035bb29aa445cdad356151e700de2994c14
a5a6a5f07c350e8ffede5a6af0f8126e1ec10913911885df043b2044cb46b0b4d4021
78a1aadedc4d63e989d239d0900f507a8964186f827ea18d23f0d859974e62913c957
80c548a262c125fe8841fdd11453abbd79407dfa3a6ec4aa4da4e40c4dae1ab44569c
fafe7331b109f9b77755f74bbea8cda67e8e2d4829a8f684272a10758525705190d06
a72036b981c4a2265b4376e2edc3186a45c2e2f820b1092784f6448354779578b442b
0369640e2f10a856cf8dfd60b28c68c9000facff8adb93d0189b9de280f03ca36d3a4
cafbf277c5773052ab51810ee6ff734f9e7ca0132d9194fc481035ee9df57636e17cb
38c2fdac3d1f7cae42cf1de1c14b72b5fd1b494432b0dcac9ca2327b
KE3: c01e2d5935d02e71c7b6832990dca8f596bee17ff07e85fc2215739f74ca6d01
9f07ed514da397708e2b23fba02a72b673587f5e38917dfa442c9d7d52d40109
export_key: dee3533f00de02168ad7cf6e419352005eecfe78c43d272d942f43d6f
37bc97ac79f9cdb0c3310e75ab81248681ac73fff280d879c978435bb1f67a3abaad0
c4
session_key: 4a7535981fc126050dcf41a182fecf5773be561f0dd321a5f86ae008
3a5fb4c603efa3f0b4dbc9c87af2a07e5c9093f9ace2d18ee2cdddb884cfd94f64197
b12
```

C.10. OPAQUE-3DH Test Vector 10

C.10.1. Configuration

```
OPRF: 0005
Hash: SHA512
SlowHash: Identity
EnvelopeMode: 02
Group: P521_XMD:SHA-512_SSWU_RO_
Nh: 64
Npk: 67
Nsk: 66
```

C.10.2. Input Values

client_identity: 0300ec0addd1ade650f8c2be98b2d7b5b5eb7e1eab56823f9413
327e056e1413055ede83bd893a26f61094a87d108431dc4f95366741da7ec6465208b
9080d17ed304a
server_identity: 0300d1df68b2171f58ffc6a2cdf6cd47f4a672e0c06660ad5ec9
cccbd8fafd4593dc847b3a3a7aedd1baf2d03dad24f1da95e884f3554c0d4915b0d47
172f33eab7f0e
password: 436f7272656374486f72736542617474657279537461706c65
envelope_nonce: 9ac2de6c1f7b8a32dfb4e365b2610797cc5b76e9d5a1e0d016c08
c33285785a9
client_private_key: 0077881aa5fd937ec7932e725ac43a07cb3ea0e90b40e0501
e6bdc3c97510cdd9475ad6d9e630235ff21b634bc650bf837aaa273530dc66aa53bb9
adb4f0ed499872
client_public_key: 0300ec0addd1ade650f8c2be98b2d7b5b5eb7e1eab56823f94
13327e056e1413055ede83bd893a26f61094a87d108431dc4f95366741da7ec646520
8b9080d17ed304a
server_private_key: 002e485cccf5018abbf875b8e81c5ade0def4fe6fa8dfc153
88367a60f23616cd1468dae601875f7dd570624d0ae9d7be2e6196708f773cf65852b
da777210337d8c
server_public_key: 0300d1df68b2171f58ffc6a2cdf6cd47f4a672e0c06660ad5e
c9cccbd8fafd4593dc847b3a3a7aedd1baf2d03dad24f1da95e884f3554c0d4915b0d
47172f33eab7f0e
client_info: 68656c6c6f20626f62
server_info: 6772656574696e677320616c696365
server_nonce: 5f07f6b2c9a546f77d2e58ae23af2960f3f723fa00bc8c6c9c53768
3dc372091
client_nonce: aff297d07060f891faa127a937e7e7995d76ec727abf796dc80c315
95bf8e201
server_keyshare: 03009af99de97de5c0b5bb5299c406e53294dc48a78ba4933df0
e01bf8e1c5e46fe1de6e82060a08a9110c435fd784b9ae31ecb639eabd464a1681912
6be3b865b05e9
client_keyshare: 030035c078a34aacc22e0e759115b9c7c45192d97e4970f40376
76039e7bc2d270c3964e81a9009a788b022eac506ac16c9704efe50ff6041bd3c9422
9673d2073d8bc
server_private_keyshare: 00708286c5fb629de5cfea56c0532dd8254a5a6e7fcc
9e51e20a1cf4f254335ca57ce603ae7cf03fc00b7a2d495298d84c8c83b686b67e825
69cb56d97e9c20e5932
client_private_keyshare: 0037735d573abb787b251879b77de4df554c91e25e11
7919a9db2af19b32ce0d501c9572d3a8a106f875023c9722b2de94efaa02c8e46a9e4
8f3e2ee00241f9a75f4
blind_registration: 0071a04b0f2180dda3c36e3d43c3a8f127158d010944b0d53
a6f8da29c3cf4f8695135d645424c747bec642bc91375ff142da4687426b0b4f35c14
eb2477c52e1fff
blind_login: 01eea8a605644334de4987fb60d9aaec15b54fc65ef1e10520556b43
938fbf81d4fbc8c36d787161fa4f1e6cf4f842989634f76f3320fdd24777894218769
fc19651
oprf_key: 0066b06b578fe36ef23a5b9872ade82b9261cc447670debcf78318add68
2f6055089b0a2484abc37f110b36f4c2a140b7a3c53dd8efb6171d3bb4d73591be848
3a1b

C.10.3. Intermediate Values

```
auth_key: 23a8d4aaf7686ae00f4a84defb8ebcf4a3d3ab1c1d7f00dcc8230a274e8
f4119b3bdeae33a94b20638345f22bd5cf93364b9608eead0986996b22be509e393b5
prk: 58200faa8a67d9c9a1ee140ae033e85709352b695d3763e2762212b682f28643
c959615781e6609c945a24bd472de4e2a1f22eae10a243a79a6a0012b28285b0
pseudorandom_pad: be47ed58f882053a44e0e21fd150a183873c4b9e9eb322ff42c
0fd0540e6581eba71b4e573f4b17426a7e30bb284bf92f27a26c84efb47736acbf69d
41094b4a1e9f
envelope: 029ac2de6c1f7b8a32dfb4e365b2610797cc5b76e9d5a1e0d016c08c332
85785a9be3065425d7f96448373cc6d8b949b844c02eb7795f3c2af5cab2139d7b754
c32e041988ed97b341d986553f0ee1b46ac5d084bb1df68119cff04f30f5f9a60386e
dd50851bfa6ebc2ac5bb71b2fdff06030d4b0c94c48edbf14d8027a21c4bfc85f2d85
796ecfdadeb002b0016188916fbdb56a428f9832fd79dc25d2e8e9b32c83
handshake_secret: 06ea9f5f3f553146bafeab9bb6bc590a2419ebbceae34266cb3
3717649c3f65dc8f993ebbe0e0c59068e58ecc4b5353fbd461588b5d9e8767b778d6c
099fdb83
handshake_encrypt_key: a3df7662b4d47d0b6133723e79a81ff74fc3413a0f2c09
cca44cbd4bba92d13335483ef31ef5cce3722df00bb8ac14dcb3b547a83415a5e4d0e
47a41f37864ef
server_mac_key: 9d355b34e8a737eb8306345fb9a58e613fb04d67f6035d47ff5df
6dbebb9d5875ed8fa0ec09d99f97efca1ba448781659de9092976b9ccdbb5f4c78a71
d739e3
client_mac_key: a90767cfd8eebccfd4b7edee797a0a6416d75014524b11f0fc7c0
508395cb03bcc2489358a44d6451ae52952b61bf9891f4599aaaf2b7f7ac0a563aad0
c9bee9
```

C.10.4. Output Values

```
registration_request: 020197d8111818258667ffbc0d377602f74350b7a54e684
1fb15ba96ac07095bcfc961a2c21e2e0061ba28cd4ea0ed93fa0404f1383b777483c3
31537c8e6e69af85b0
registration_response: 020013e275bf8d4c305cd3793a5be014f9b338b12c6f97
7aef5d523cff2c753b5e6d0f2602fa8359918eaf2fb4ccfb0ae79c383f698ee0fff3a
05d6ce9e5b28e762b0b0300d1df68b2171f58ffc6a2cdf6cd47f4a672e0c06660ad5e
c9cccbd8fafd4593dc847b3a3a7aedd1baf2d03dad24f1da95e884f3554c0d4915b0d
47172f33eab7f0e
registration_upload: 0300ec0addd1ade650f8c2be98b2d7b5b5eb7e1eab56823f
9413327e056e1413055ede83bd893a26f61094a87d108431dc4f95366741da7ec6465
208b9080d17ed304a029ac2de6c1f7b8a32dfb4e365b2610797cc5b76e9d5a1e0d016
c08c33285785a9be3065425d7f96448373cc6d8b949b844c02eb7795f3c2af5cab213
9d7b754c32e041988ed97b341d986553f0ee1b46ac5d084bb1df68119cff04f30f5f9
a60386edd50851bfa6ebc2ac5bb71b2fdff06030d4b0c94c48edbf14d8027a21c4bfc
85f2d85796ecfdadeb002b0016188916fbdb56a428f9832fd79dc25d2e8e9b32c83
KE1: 02013ffd159c4d44f7fe2441c05614ef421e7fc7285432d5dd3b67ada061f3e3
a230d1ab200864a9a716cd001d2a6abea298d58fded61f7d9ce02fc1bb037a1bbf9c7
caff297d07060f891faa127a937e7e7995d76ec727abf796dc80c31595bf8e2010009
68656c6c6f20626f62030035c078a34aacc22e0e759115b9c7c45192d97e4970f4037
676039e7bc2d270c3964e81a9009a788b022eac506ac16c9704efe50ff6041bd3c942
29673d2073d8bc
KE2: 0200e302a5573d3625a0f9d0f63398f4c5053d4f816c743ab77bb365a36c3cdd
00fe21ae2a7e56c01f0857ecdb4d129480c189cbe61f78a2aaaa4687126b76a6cf0ce
20300d1df68b2171f58ffc6a2cdf6cd47f4a672e0c06660ad5ec9cccbd8fafd4593dc
847b3a3a7aedd1baf2d03dad24f1da95e884f3554c0d4915b0d47172f33eab7f0e029
ac2de6c1f7b8a32dfb4e365b2610797cc5b76e9d5a1e0d016c08c33285785a9be3065
425d7f96448373cc6d8b949b844c02eb7795f3c2af5cab2139d7b754c32e041988ed9
7b341d986553f0ee1b46ac5d084bb1df68119cff04f30f5f9a60386edd50851bfa6eb
c2ac5bb71b2fdff06030d4b0c94c48edbf14d8027a21c4bfc85f2d85796ecfdadeb00
2b0016188916fbdb56a428f9832fd79dc25d2e8e9b32c835f07f6b2c9a546f77d2e58
ae23af2960f3f723fa00bc8c6c9c537683dc37209103009af99de97de5c0b5bb5299c
406e53294dc48a78ba4933df0e01bf8e1c5e46fe1de6e82060a08a9110c435fd784b9
ae31ecb639eabd464a16819126be3b865b05e9000fe366a1b3fae5ebfbe29ace6edfd
cca85428f071a2a410205f3792aa810a0b69107ee1c97c3588e4a798072e69b2555e3
3928a38b81b90daf4cb0d80b2a3fee7d81bd3a67de7620c11927684533f2c7
KE3: b825f4735127be2e5870088690ceb1d107122f9febe6ff95cf5d2c2ccee473a9
1db656b5d47824a7dd2e1d4180cdf01482461eb341627d70b18ee61c63b6c534
export_key: a1e47a5e5d13c00290f0207542cf3db49c76c57dddc1eb6f8e35e17fd
f04e32ef769ee58605783e4e6c86c5823596d5fab79f42c1f992137554ca4281c3038
d2
session_key: 11396799cfb47fe2379020b8f706c520115858f0c59c4fdb1db2a5c4
49e6c08f53f75e77b2712d2ba977c5ba85e07d98cbdf8dc7a98700eb61b50d2cda4b8
a08
```

Authors' Addresses

Hugo Krawczyk

Algorand Foundation

Email: hugokraw@gmail.com

Kevin Lewi
Novi Research

Email: lewi.kevin.k@gmail.com

Christopher A. Wood
Cloudflare

Email: caw@heapingbits.net