Protocols

List of PAKEs to be considered:

Balanced

- ECJPAKE
- SPAKE2
- CPace

Augmented

- SRP
- SPAKE2+
- OPAQUE

Conventions

Input is marked "user" if it is coming from the user and "peer" if it is coming from the peer. Output is marked by "user" if the user needs that output and by "peer" if it is eventually consumed by the peer.

The output is called "session key" if it could in theory be used as a key directly and "key material" if it must only be used as an input to key derivation. It will be called "explicit" if it has been confirmed by explicit key confirmation and "implicit" otherwise.

CPace

Standard (draft): draft-haase-cpace-01
Link: https://tools.ietf.org/html/draft-haase-cpace-01
Primitives used: Prime order group, Hash, KDF
Options: Ciphersuite (Prime order group, serialisation and mappings, Hash, domain separation strings)
Rounds: 2

Inputs:

```
- sid: Bytes
- channel identifier: Bytes
- password releated string: Bytes
```

Output:

```
- session key: Bytes
```

Round 1

Inputs:

```
- sid: Bytes (user)
- channel identifier: Bytes (user)
```

```
- password releated string: Bytes (user)
```

Outputs:

```
- key share: Group element (peer)
- ephemeral key: Big integer
```

Round 2

Inputs:

```
- peer's key share: Group element (peer)
- ephemeral key: Big integer
- for transcript:
    - sid: Bytes
    - key share: Group element
```

Output:

```
- implicit session key: Bytes (user)
```

Key Confirmation

Not defined by the standard.

Remarks

- Serialisation is assumed to be defined, but the standard does not define it
- It is a balanced PAKE, but for transcript calculation it needs to know who sent their key share first, or have an agreement which key share to include first

SPAKE2

Standard (draft): draft-irtf-cfrg-spake2
Link: https://tools.ietf.org/html/draft-irtf-cfrg-spake2-18
Primitives used: KDF, MAC, Hash
Options: Prime Order Group, Per user M and N
Rounds: 2

Inputs:

```
- UserID: Bytes
- Peer's UserID: Bytes
- Additional Authenticated Data: Bytes
- PBKDF output: Bytes
```

Output:

```
- session key: Bytes
```

Round 1

Inputs:

```
- PBKDF output: Bytes (user)
```

Outputs:

```
- Key share: Group element (peer)
- Ephemeral private key: Big integer
```

Round 2

Inputs:

```
- UserID: Bytes (user)
- Peer's UserID: Bytes (user)
- Additional Authenticated Data: Bytes (user)
- Peer's key share: Group element (peer)
- PBKDF output: Bytes
- Key share: Group element
- Ephemeral private key: Big integer
```

Outputs:

```
- Key confirmation message: Bytes (peer)
- Key confirmation verification key: Bytes
- Implicit session key (user)
```

Key confirmation

Inputs:

```
- Peer's key confirmation message: Bytes (peer)
- Implicit session key
- Key confirmation verification key: Bytes
```

Outputs:

```
- Explicit session key (user)
```

Remarks

- Unlike some other protocols, this standard considers password hashing in scope, discusses the matter but it is not integral part of the protocol and any secure pre-agreed method will do. Therefore password hashing is not listed in the “primitives used” section.
- The KDF is used as part of the protocol, but it does not need to match the KDF used to derive keys from the shared secret. Because of this, the KDF is listed in the “primitives used section” In theory any pre-agreed secure KDF would do, but the “ciphersuites” proposed all use HKDF.

J-PAKE

Standard: RFC 8236, THREAD
Link: https://tools.ietf.org/html/rfc8236
Primitives used: Prime order group arithmetic, Schnorr Non-Interactive Zero Knowledge Proof (NIZKP)
Options: Group (can be over a finite field or an elliptic curve), Rounds (2 or 3), Hash algorithm
Rounds: 3

Inputs:

```
- Password: Bytes
- UserID: Bytes
```

Output:

```
- Key material: Group element
```

Round 1

Inputs:

```
- UserID: Bytes (user)
```

Outputs:

```
- Ephemeral public key 1: Group element (peer)
- ZKP for private key 1: Big Integer (peer)
- Ephemeral private key 2: Big Integer
- Ephemeral public key 2: Group element (peer)
- ZKP for private key 2: Big Integer (peer)
```

Round 2

Inputs:

```
- Peer's ephemeral public key 1: Group element (peer)
- Peer's ZKP for private key 1: Big Integer (peer)
- Peer's ephemeral public key 2: Group element (peer)
- Peer's ZKP for private key 2: Big Integer (peer)
- Password: Bytes (user)
- UserID: Bytes
- Ephemeral public key 1: Group element
- Ephemeral private key 2: Big Integer
```

Outputs:

```
- Round 2 public key: Group element (peer)
- ZKP for Round 2 private key: Big Integer (peer)
- Round 2 private key: Big integer
```

Round 3

Inputs:

```
- Peer's round 2 public key: Group element (peer)
- Peer's ZKP for Round 2 private key: Big Integer (peer)
- Round 2 private key: Big integer
- UserID: Bytes
- Password: Bytes
- Ephemeral private key 2: Big Integer
- Peer's ephemeral public key 2: Group element
```

Output:

```
- Implicit key material: Group element
```

Key confirmation

Examples provided, but nothing mandated.

Remarks

- Getting the Round 2 private key as an input instead of the password saves a bignum multiplication, but it probably will be stored in the state and will be an implementation detail
- Here password means a value derived from the password lower than the group order. The exact method is out of scope for the standard.

SRP

Standard: RFC2945 (IEEE 1363.2, ISO/IEC 11770-4, RFC5054)
Link: https://tools.ietf.org/html/rfc2945
Primitives used: Hash, Prime field arithmetic
Options: Hash, Prime (preferrably of the form 2p+1), Generator of multiplicative subgroup
Rounds: 2

Inputs:

```
- username: Bytes (client)
- password: Bytes (client)
- salt: Bytes
- verifier: Field element (server)
```

Outputs:

```
- session key: Bytes
```

Round 1

Client

Inputs: none

Outputs:

```
- client keyshare: Field element (peer)
- client ephemeral key: Big integer
```

Server

Inputs:

```
- verifier: Field element (user)
```

Outputs:

```
- server keyshare: Field element (peer)
- server ephemeral key: Big integer
```

Round 2

Client

Inputs:

```
- salt: Bytes (user)
- password: Bytes (user)
- username: Bytes (user)
- server keyshare: Field element (peer)
- client keyshare: Field element
- client ephemeral key: Big integer
```

Outputs:

```
- client key confirmation: Bytes (peer)
- implicit session key: Bytes
```

Server

Inputs:

```
- client keyshare: Field element (peer)
- server keyshare: Field element
- verifier: Field element
```

Outputs:

```
- implicit session key: Bytes
```

Key confirmation

Server

Inputs:

```
- client key confirmation: Bytes (peer)
- transcript:
    - username: Bytes
    - salt: Bytes
    - client keyshare: Field element
    - server keyshare: Field element
- implicit session key: Bytes
```

Outputs:

```
- explicit session key: Bytes
- server key confirmation: Bytes (peer)
```

Client

Inputs:

```
- server key confirmation: Bytes (peer)
- client key confirmation: Bytes
- client keyshare: Field element
- implicit session key: Bytes
```

Outputs:

```
- explicit session key: Bytes
```

Remarks

- The username and the salt are exchanged in plaintext as a setup, leaving it out from the flow for now
- The standard defines integer serialisation (RFC2945)
- Broader message format is out of scope for the standard (RFC2945), but they are defined for example for TLS (RFC5054)
- Both RFCs use an instantiation with SHA1, this is clearly outdated
- The RFC2945 standard uses plain hash for key confirmation (in a safe way), but mentions that HMAC could be used as well
- There are several ways to organise message flow. The above flow was presented by RFC2945 with the modification that key confirmation is separated, this way the above abstract API could potentially serve RFC5054 as well, if there is an option for early key extraction
- RFC2945 describes SRP-3, but SRP-6 and SRP-6a only differs in internals and does not affect the interface
- For the handshake transcript one or two running hashes could be used as well, or the components stored in a different way as depicted above,

but this should be an implementation detail and the API should allow for different approaches

OPAQUE

Standard (draft): draft-irtf-cfrg-opaque
Link: https://tools.ietf.org/html/draft-irtf-cfrg-opaque-03 (,uses https://tools.ietf.org/html/draft-irtf-cfrg-voprf-06 for Oblivious PRF)
Primitives used: Authenticated Key Exchange, HKDF, HMAC, Oblivious Pseudorandom Function, Memory Hard Function
Options: Oblivious PRF (prime order group, hash), Authenticated Key Exchange (prime order group - it is not a must, but should match the group in the Oblivious PRF), MHF (Argon2, scypt or PBKDF2), Envelope mode (authenticated credentials format)
Rounds: 2

Inputs:

```
- password: Bytes (client)
- client info: Bytes (client)
- server private key: Big integer (server)
- server public key: Group element (server)
- credential file (server):
    - oprf private key: Big integer
    - client public key: Group element
    - envelope (encrypted client private key): Bytes
```

Output:

```
- session secret: Bytes
- export key: Bytes (client)
```

Round 1

Client

Inputs:

```
- password: Bytes (user)
- client info: Bytes (user)
```

Outputs:

```
- blind: Big integer
- client ephemeral private key: Big integer
- key exchange message 1: (peer)
    - credential request: Group element
    - client keyshare: Group element
    - client nonce: Bytes
    - client info: Bytes
```

Server

Inputs:

```
- key exchange message 1: (peer)
    - credential request: Group element
    - client nonce: Bytes
    - client info: Bytes
    - client keyshare: Group element
- credential file: (user)
    - oprf private key: Big integer
    - client public key: Group element
    - envelope (encrypted client private key): Bytes
- server public key: Group element (user)
- server private key: Big integer (user)
- client identity: Bytes (user)
- server identity: Bytes (user)
```

Outputs:

```
- implicit session key
- client mac: Bytes
- key exchange message 2: (peer)
    - credential response:
        - Blinded PRF output: Group element
        - server public key: Group element
        - envelope (encrypted client private key): Bytes
    - server nonce: Bytes
    - server keyshare: Group element
    - encrypted server info: Bytes
    - server mac: Bytes
```

Round 2

Client

Inputs:

```
- key exchange message 2: (peer)
    - credential response:
        - Blinded PRF output: Group element
        - server public key: Group element
        - envelope (encrypted client private key): Bytes
    - server nonce: Bytes
    - server keyshare: Group element
    - encrypted server info: Bytes (optional)
    - mac: Bytes
- client identity: Bytes (user)
- server identity: Bytes (user)
```

```
- key exchange message 1: Bytes
```

Outputs:

```
- key exchange message 3 (client mac): Bytes (peer)
- explicit session key: Bytes (user)
- export key: Bytes (user)
- server info: Bytes (user)
```

Server

Inputs:

```
- key exchange message 3 (client mac): Bytes (peer)
- implicite session key: Bytes
- client mac: Bytes
```

Outputs:

```
- explicit session key: Bytes
```

Remarks

- Unlike some other PAKE schemes, here the MHF is integral part of the algorithm, not just some pre-processing
- In theory it can work with any AKE, but the standard only discusses Diffie-Hellman, after PQ schemes are standardised, it is likely to be extended to use PQ AKE algorithms
- Input or output marked "virtual" isn't necessary for the computations and some of these could be separated from the API if necessary and left to the user to handle them
- Input or output marked "hide" shouldn't be released to the user because it hasn't been authenticated yet, or it shouldn't be released to the user at all.
- In the calculation of MAC values a handshake transcript is involved. The draft standard does not define the handshake transcript, in the above I am assuming that it can be implemented with a running hash and hidden in the state. It might or might not be the same as the preamble
- The nonces are not used directly, their only purpose is to randomise the handshake transcript
- Client and server identities are only used in custom identifier mode and are communicated out of band, during the registration process

SPAKE2+

Standard (draft): draft-bar-cfrg-spake2plus
Link: https://tools.ietf.org/html/draft-bar-cfrg-spake2plus-02
Primitives used: KDF, MAC, Hash
Options: Group (can be over a finite field or an elliptic curve), key confirmation method
Rounds: 2

Inputs:

```
- UserID: Bytes
- Peer's UserID: Bytes
- Additional Authenticated Data: Bytes
- PBKDF output: Bytes (client)
- Verification value1: Big Integer (server)
- Verification value2: Group element (server)
```

Output:

```
- Shared secret: Bytes
```

Round 1

Client

Inputs:

```
- PBKDF output: Bytes (user)
```

Outputs:

```
- Key share: Group element (peer)
```

Server

Inputs:

```
- PBKDF output: Bytes (user)
```

Outputs:

```
- Key share: Group element (peer)
```

Round 2

Client

Inputs:

```
- UserID: Bytes (user)
- PBKDF output: Bytes (user)
- Peer's UserID: Bytes (user)
- Additional Authenticated Data: Bytes (user)
- Peer's key share: Group element (peer)
- Key share: Group element
```

Outputs:

```
- Key confirmation message: Bytes (peer)
- Key confirmation verification key: Bytes
- implicit session key: Bytes
```

Server

Inputs:

```
- UserID: Bytes (user)
- Verification value1: Big Integer (user)
- Verification value2: Group element (user)
- Peer's UserID: Bytes (user)
- Additional Authenticated Data: Bytes (user)
- Peer's key share: Group element (peer)
- Key share: Group element
```

Outputs:

```
- Key confirmation message: Bytes (peer)
- Key confirmation verification key: Bytes
- implicit session key: Bytes
```

Key confirmation

Inputs:

```
- Peer's key confirmation message: Bytes (peer)
- Key confirmation verification key: Bytes
- implicit session key: Bytes
```

Outputs:

```
- explicit session key: Bytes (user)
```

Remarks

- Unlike some other protocols, this standard considers password hashing in scope, discusses the matter but it is not integral part of the protocol and any secure pre-agreed method will do. Therefore password hashing is not listed in the "primitives used" section.
- The KDF is used as part of the protocol, but it does not need to match the KDF used to derive keys from the shared secret. Because of this, the KDF is listed in the "primitives used section" In theory any pre-agreed secure KDF would do, but the "ciphersuites" proposed all use HKDF.
- In Round 2 the implicit session key is deliberately not released to the user (key confirmation is a MUST in the standard)