The Wasmium X3DH Specification

The Wasmium X3DH algorithm is adopted from the Signal Protocol X3DH spec at X3DH page with some modifications. This document is mostly a replica of Signal's X3DH specsheet but modifies the spec or provides clarification on the spec where necessary.

- Curve used is `X25519`
- Hash is 256bit Blake3 hash
- Encoding Function `Encode(PK)` to a byte sequence is Borsh Encoding
- `info` is `WASMIUM_XXX3DH`
- `B3KDF` is a KDF derived from using `Blake3` in `derive_key()` function.

Crypographic Notation

- `X||Y` - the concatenation of a byte sequence `X` and `Y`
- `DH(PK1, PK2)` - represents a byte sequence which is the shared secret output from an X25519 ECDH function involving key pairs `PK1` and `PK2`.
- `Sig(PK, M)` - represents a byte sequence that is an Ed25519 signature on the byte sequence `M` signed with the key pair from the public key `PK`.
- `B3KDF(K,M)` represents 32 byte sequence with inputs where `K` is the secret key material and `M` is the `info` specified above as `WASMIUM_XXX3DH`.
- `WNCS` - Wasmium Network Core Service which is responsible for providing storage and retrieval of X25519 public keys and providing privacy of messages and contacts.

Parties

This X3DH spec involves 3 parties some of who might be part of a receiving message. The three parties are:

- Alice wants to send Bob some initial data and establish a shared secret for bi-directional communication.
- Bob wants to allow parties like Alice to establish as shared key with him and send encrypted data. Since Bob might be is offline when parties like Alice attempts to establish a shared secret with him, he first publishes his X25519 public keys to a derive account of his public key to the WNCS. This is a trustless approach to publishing X25519 public keys since only Bob hold his Ed25519 key pair.
- The Wasmium Network Core Service which handles transparency, censorship resistance and is responsible for verifying the smart contract publishing of the X25519 public keys to the derived account of the Ed25519 key pair.

Keys

NAME	DEFINITION
IK_A	Alice's Identity key
EK_A	Alice's ephemeral key
IK_B	Bob's Identity key
SPK_B	Bob's signed prekey
OPK_B	Bob's once-time prekey

- Each party has a long-term identity public key (IK_A for Alice, IK_B for Bob).
- Each X25519 public key as a corresponding private key, omitted in this description for simplicity.
- Bob also hash a signed prekey SPK_B which he will change periodically and a set of one-time prekeys OPK_B which are used in a single X3DH protocol run. "Prekeys" are so named because they are essentially protocol messages which Bob publishes to the server *prior* to Alice beginning the protocol run).
- During each protocol run, Alice generates a new ephemeral key pair with public key EK_A.
- After a successful protocol run Alice and Bob will share a 32-byte secret key SK. This key may be used within some post-X3DH secure communication protocol.

Protocol

The protocol has 3 phases:

1. Bob publishes his identity key and prekeys to the WNCS.
2. Alice queries the WNCS for a "prekey bundle" and uses it to send the initial message to Bob.
3. Bob receives and processes Alice's initial message.

Publishing the Keys

Bob publishes a set of X25519 keys to the WNCS containing

- Bob's identity key IK_B
- Bob's signed prekey SPK_B
- Bob's prekey signature $Sig(IK_B, Encode(SPK_B))$
- A set of Bob's one-time prekeys (OPK_B^1, OPK_B^2, OPK_B^3...)

Bob only needs to upload his identity key to the WNCS once but the set of one-time prekeys are updated regualry when WNCS informs Bob that the store of one-time prekeys is getting low.

Bob also publishes a new signed prekey and prekey signature after a period of one week which replace the previous signed prekey and prekey signature.

After publishing a new signed prekey, Bob may keep the private key corresponding to the previous signed prekey around for some period of time to handle messages that may have been delayed in transit. WNCS core may also inform Alice of a new prekey in order for her to encrypt afresh messages that might be delayed on her end due to network issues. Eventually, Bob will delete his private key for forward secrecy while the one-time prekey private keys will be deleted as Bob receives messages using them.

Sending the initial message

To perfom an X3DH key agreement with Bob, Alice contacts WNCS and fetches the prekey bundle containing

- Bob's identity key IK_B
- Bob's signed prekey SPK_B
- Bob's prekey signature $Sig(IK_B, Encode(SPK_B))$
- Bob's one-time prekey OPK_B

The WNCS should provide one of Bob's one-time prekeys if one exists and then deletes it. If all of Bob's one-time prekeys are used up, the WNCS requests Bob for new additional one-time prekeys. This process ensures forward secrecy.

- Alice verifies the prekey signature and aborts if verification fails.
- Alice then generates an ephemeral key pair with public key EK_A.

Alice calculates

- $DH1 = DH(IK_A, SPK_B)$
- $DH2 = DH(EK_A, IK_B)$
- $DH3 = DH(EK_A, SPK_B)$
- $DH4 = DH(EK_A, OPK_B)$
- $SK = B3KDF(DH1||DH2||DH3||DH4)$

`DH1` and `DH2` provide `mutual authentication` while `DH3` and `DH4` provide `forward secrecy`.

After calculating `SK`, Alice deletes her ephemeral private key and the `DH` outputs.

Alice then calculates an `associated data` byte sequence `AD` that contains identity information for both parties:

$$\mathbf{AD = Encode(IKI_A) \,||\, Encode(IK_B)}$$

Alice may optionally append additional information to `AD`, such as Alice and Bob's username, certificates or other identifying information.

Alice then send Bob an initial message containing:

- Alice's identity key IK_A
- Alice's ephemeral key EK_A
- Identifiers stating which of Bob's prekeys Alice used
- An initial ciphertext encrypted with some `AEAD` encryption scheme using `AD` as associated data and using an encryption key which is either `SK` or the output from some cryptographic `PRF` keyed by `SK`.

The initial ciphertext is typically the first message in some post-X3DH communication protocol. In other words, this ciphertext typically has two roles, serving as the first message within some post-X3DH protocol, and as part of Alice's X3DH initial message.

After sending this, Alice may continue using *SK* or keys derived from *SK* within the post-X3DH protocol for communication with Bob.

Receiving the initial message

Upon receiving Alice's initial message, Bob retrieves Alice's identity key and ephemeral key from the message. Bob also loads his identity private key, and the private key(s) corresponding to whichever signed prekey and one-time prekey Alice used.

Using these keys, Bob repeats the `DH` and `KDF` calculations from the previous section to derive `SK`, and then deletes the `DH` values.

Bob then constructs the `AD` byte sequence using IK_A and IK_B, as described in the previous section. Finally, Bob attempts to decrypt the initial ciphertext using `SK` and `AD`. If the initial ciphertext fails to decrypt, then Bob aborts the protocol and deletes `SK`.

If the initial ciphertext decrypts successfully the protocol is complete for Bob. Bob deletes any one-time prekey private key that was used, for forward secrecy. Bob may then continue using `SK` or keys derived from `SK` within the post-X3DH protocol for communication with Alice.

Security Considerations

- Before the X3DH key agreement, the parties may authenticate each others X25519 identity public key by verifying the signature of the x25519 identity public key signed by each others wallet Ed25519 public keys. This can be done by retrieving their username from WNCS or from their associated account on the blockchain. If authentication is not performed, the parties receive no cryptographic quarantee as to who they are communicating with.
- By ensuring that one-time prekeys are enforced, protocol replay can be avoided. Protocol replay happens when Alice's initial message is replayed to Bob and Bob accepts the message. A post-X3Dh protocol may quickly negotiate a new encryption key for Alice based on a fresh random input from Bob. This is common behaviour of Diffie-Hellman ratcheting protocols.
- X3DH dosen't give either Alice or Bob a publishable cryptographic proof of the contents of their communication or the fact that they communicated. If a third party has compromised legitimate private keys from Alice and Bob they might provide a communication transcript that appears to be between Alice and Bob. Signing the hashes of messages using the wallet Ed25519 keypair might mitigate this unless the Ed25519 keypair of a party is compromised.
- A malicious server might provide Alice a prekey bundle with forged prekeys and later compromise Bob's IK_B to calculate `SK`. Signing the prekeys with Bob's wallet Ed25519 keypair would mitigate this issue. This would remove deniability properties of the key exchange protocol. If deniability is needed, the signature verification may be ommited.
- To avoid DoS attack where an adversary can try to make Alice or Bob keep generating one-time prekeys, each party can whitelist the addresses that can fetch prekeys and the Wasmium Network Core Service can apply various algorithms like IP whitelisting and rate-limiting or frequency of one-time prekey access. Maintaining the whitelist of contacts in the Wasmium Network Core Service also allows privacy to be maintained since the contacts of a party are not published on the open blockchain for whitelisting.
- Using X3DH, double-ratchet and stream encryption allows communication to be fast since other privacy enhancing protocols that rely on zero-knowledge proofs and fully homomorphic encryption require extra computation on data thus slowing down communication.