The WhatsApp Companion Device Linking and Authentication Protocol

1. Introduction

The WhatsApp Companion Device Linking Protocol provides secure mechanisms for a user to link a new "companion" device (like a desktop or web client) to their existing "primary" device (their mobile phone). This allows the companion device to send and receive messages on behalf of the user.

The protocol is designed to be robust against man-in-the-middle (MITM) attacks by ensuring that the cryptographic keys establishing the session are authenticated through a secure, out-of-band channel. This document specifies two primary methods for this out-of-band authentication: scanning a QR code and entering a short code displayed on the primary device.

2. Overview

2.1. Roles

- **Primary Device**: The user's main mobile phone, which holds the authoritative identity and keys for the account.
- **Companion Device**: A new device (e.g., WhatsApp Web, Desktop) that wishes to be linked to the user's account.
- **Server**: The WhatsApp server infrastructure, which facilitates the exchange of messages between the primary and companion devices during the linking process. The server is treated as an untrusted party; the protocol's security does not depend on the server's integrity.

2.2. Terminology

- **IQ Stanza**: An Info/Query message in the XMPP protocol used for structured requests and responses.
- **ADVSecretKey**: A long-term 32-byte secret key stored on the companion device, used to prove its identity during pairing.
- **Device Identity**: A signed data structure containing the cryptographic identity of the primary device.
- **Ref**: A short-lived token provided by the server used to route pairing messages.
- **Linking Code**: A short alphanumeric code used for phone number pairing.

2.3. Cryptographic Primitives

- **Curve25519**: Used for all Elliptic Curve Diffie-Hellman (ECDH) operations.
- **HMAC-SHA256**: Used for message authentication codes.
- **SHA-256**: Used for hashing.
- **AES-CTR**: Used for symmetric encryption in the phone number pairing flow.
- **PBKDF2**: Password-Based Key Derivation Function 2, used to derive a key from the linking code.
- **XEdDSA**: As specified in the XEdDSA document, used for signing device identities.

3. Key Components

3.1. Identity Keys

Both the primary and companion devices generate a long-term Curve25519 identity key pair (IdentityKey).

3.2. ADVSecretKey

Upon first launch, before pairing, the companion device generates a cryptographically random 32-byte ADVSecretKey. This key is fundamental to the security of the QR code pairing flow.

3.3. Device Identity

The ADVSignedDeviceIdentity is a Protobuf structure containing the primary device's identity information. It is signed by the primary device's account key to prevent tampering. Its core components are:

- details: A serialized ADVDeviceIdentity Protobuf containing the account's creation timestamp and a key index.
- accountSignatureKey: The public part of the key used to sign the identity.
- accountSignature: The signature over the details and the companion's identity public key.
- deviceSignature: A signature generated by the companion device to prove it has been authorized.

4. Protocol 1: QR Code Pairing

This is the most common linking method. The companion displays a QR code that is scanned by the primary device.

4.1. Step 1: Companion Initiates and Generates QR Code

1. The companion device connects to the WhatsApp server and performs an initial Noise handshake.

2. The server sends a pair-device IQ stanza containing a list of ref tokens.
3. For each ref, the companion generates a QR code. The content of the QR code is a comma-separated string:
   ref,Base64(CompanionNoiseKey.Public),Base64(CompanionIdentityKey.Public),Base64(ADVSecretKey)
4. The companion displays this QR code to the user and waits.

4.2. Step 2: Primary Scans and Verifies

1. The user scans the QR code with their primary device.
2. The primary device parses the four components from the QR code string.
3. The primary device now has the companion's public keys and the ADVSecretKey. It can now establish a secure channel and provision the companion.

4.3. Step 3: Primary Provisions Companion

1. The primary device sends a pair-success IQ stanza to the server, addressed to the companion (identified by the ref). This message contains the primary's ADVSignedDeviceIdentityHMAC.
2. This structure contains the primary's ADVSignedDeviceIdentity, which is itself protected by an HMAC. The key for this HMAC is the ADVSecretKey received from the QR code.
   HMACKey = ADVSecretKey
   HMAC = HMAC-SHA256(key=HMACKey, data=Prefix || ADVSignedDeviceIdentity.details)
   (where Prefix is a domain separation tag, 0x0600).

4.4. Step 4: Companion Finalizes Pairing

1. The companion receives the pair-success stanza from the server.
2. It verifies the HMAC on the ADVSignedDeviceIdentityHMAC using its own ADVSecretKey. **This is the critical authentication step.** If the HMAC is valid, the companion knows the message came from a device that scanned its QR code.
3. The companion then verifies the accountSignature within the ADVSignedDeviceIdentity (see Section 6.1).
4. If both verifications pass, the companion generates its own deviceSignature (see Section 6.2).
5. The companion stores the primary's identity information and sends a final confirmation IQ to the server containing its self-signed device identity. The pairing is complete.

5. Protocol 2: Phone Number Code Pairing

This method is used when scanning a QR code is not possible. The companion initiates the process, and the user enters a code displayed on the primary device.

5.1. Step 1: Companion Initiates "Companion Hello"

1. The companion generates a long-term identity key pair (IdentityKey) and a temporary ephemeral key pair (EphemeralKey).
2. It sends a link_code_companion_reg IQ to the server with stage="companion_hello". This message includes:
   - The target phone number (JID).
   - The companion's identity public key.
   - A specially constructed link_code_pairing_wrapped_companion_ephemeral_pub. This is the companion's ephemeral public key, symmetrically encrypted using a key derived from a secret *linking code* that the companion does not yet know. This construction allows the primary device (which *will* know the code) to decrypt it.
3. The server responds with a link_code_pairing_ref, a token referencing this pairing attempt. The primary device will be notified and will display a short linking code to the user.

5.2. Step 2: Primary Enters Code and Initiates Key Exchange

1. The user enters the linking code from their primary device into the companion device's UI.
2. The companion now knows the linking code. It derives a symmetric key using PBKDF2:
   LinkCodeKey = PBKDF2(password=LinkingCode, salt=..., iterations=..., len=32, hash=SHA256)
3. Meanwhile, the primary device sends a notification to the companion (routed via the server using the pairing_ref) containing:
   - The primary's identity public key (PrimaryIdentityKey.Public).
   - The primary's ephemeral public key, encrypted in the same way as the companion's was, using the LinkCodeKey.

5.3. Step 3: Companion Completes Key Exchange and Finishes

1. The companion receives the notification from the primary.
2. It uses the LinkCodeKey (which it now has) to decrypt the primary's ephemeral public key.
3. It performs two ECDH operations:
   - EphemeralSharedSecret = ECDH(CompanionEphemeralKey.Private, PrimaryEphemeralKey.Public)

    - IdentitySharedSecret = ECDH(CompanionIdentityKey.Private, PrimaryIdentityKey.Public)
4. These shared secrets are used to derive keys for an encrypted bundle containing the companion's identity key, the primary's identity key, and a random advSecretRandom.
5. The companion sends a final link_code_companion_reg IQ with stage="companion_finish", containing the encrypted key bundle. This proves to the primary that the companion knew the linking code. The flow then merges with the final steps of the QR pairing, involving the exchange of the signed device identity.

6. Core Cryptography: Device Identity Verification

Both protocols converge on a final step where the companion must verify the primary's identity and sign it with its own key.

6.1. Account Signature Verification

The companion must verify the signature on the ADVSignedDeviceIdentity.

- **Message**: Prefix || ADVSignedDeviceIdentity.details || CompanionIdentityKey.Public
- **Public Key**: ADVSignedDeviceIdentity.accountSignatureKey
- **Signature**: ADVSignedDeviceIdentity.accountSignature
- Algorithm: XEdDSA
  The inclusion of the companion's public key in the signed message ensures that this identity blob is "bound" to this specific companion and cannot be replayed to another.

6.2. Device Signature Generation

After successful verification, the companion generates its own signature to prove it has been authorized.

- **Message**: Prefix || ADVSignedDeviceIdentity.details || CompanionIdentityKey.Public || ADVSignedDeviceIdentity.accountSignatureKey
- **Private Key**: CompanionIdentityKey.Private
- **Signature**: The resulting deviceSignature.
- Algorithm: XEdDSA
  This self-signed identity is sent to the server in the final confirmation message.

7. Security Considerations

- **Out-of-Band Authentication**: The security of both protocols hinges on the out-of-band step (scanning the QR code or entering the phone code). An attacker who can see the QR code or the linking code can compromise the session.

- **Server Trust**: The server is an untrusted intermediary. It can see public keys and encrypted blobs, but it cannot derive any shared secrets or forge the critical HMACs and signatures that authenticate the devices to each other.
- **Ephemeral Keys**: The use of ephemeral keys in the phone number pairing flow provides forward secrecy for the key exchange.
- **Identity Binding**: By including the companion's public key in the primary's signature, the protocol prevents an attacker from taking a valid device identity from one pairing session and using it in another.

8. References

This document is based on analysis of the open-source whatsmeow library and public documentation on the Signal Protocol and its derivatives.