Matrix Cryptographic Key Infrastructure

Sumner Evans

21 September 2024

Beeper (Automattic)

Why Cryptography?

Matrix uses cryptography for two main purposes:

1. **Message Security** — only the people who are part of the conversation should be allowed to view messages of the conversation.
2. **Identity** — verifying that a user or device is who they say they are.

Why Cryptography?

Matrix uses cryptography for two main purposes:

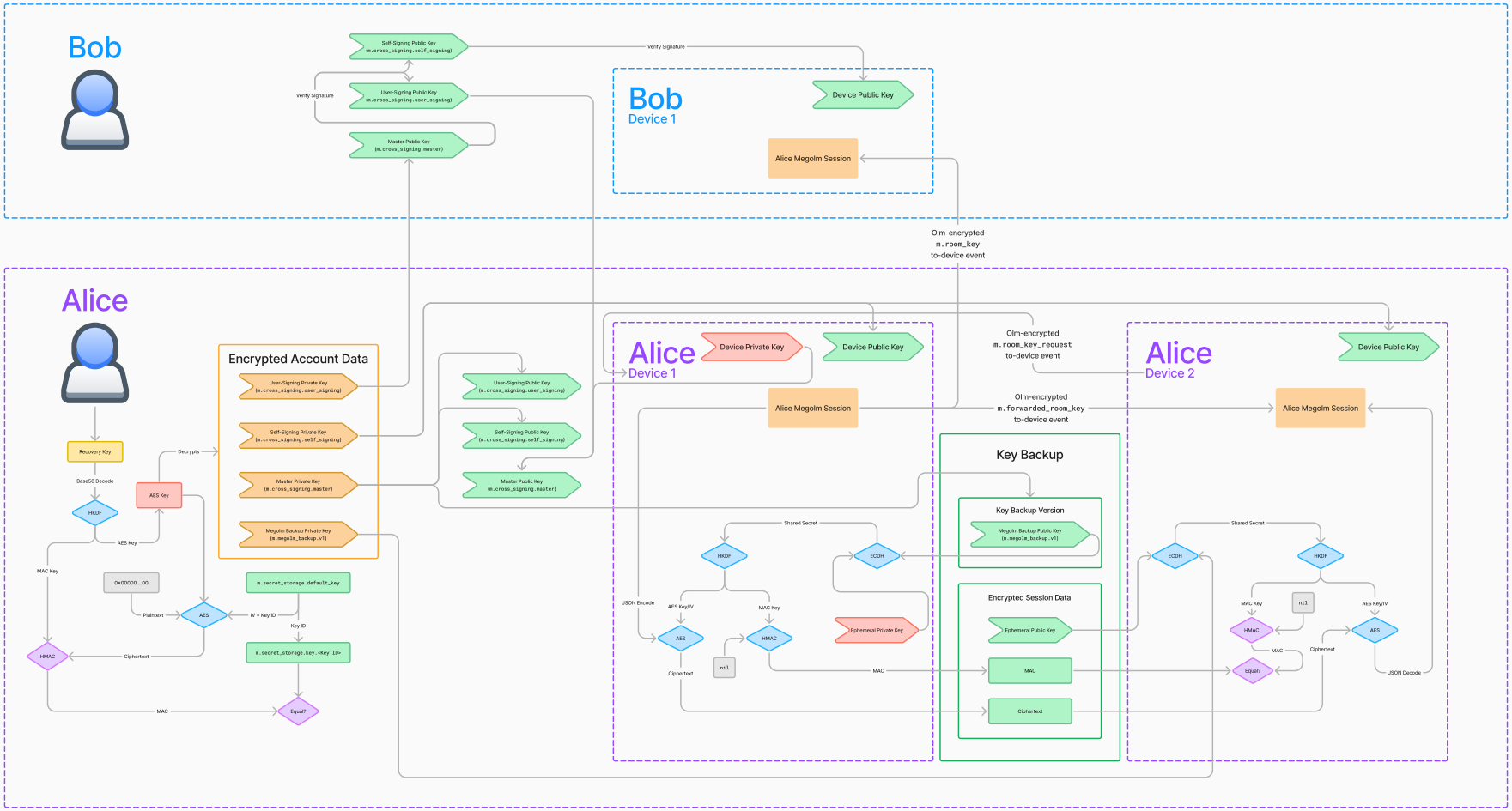
1. **Message Security** — only the people who are part of the conversation should be allowed to view messages of the conversation.
2. **Identity** — verifying that a user or device is who they say they are.

Why Cryptography?

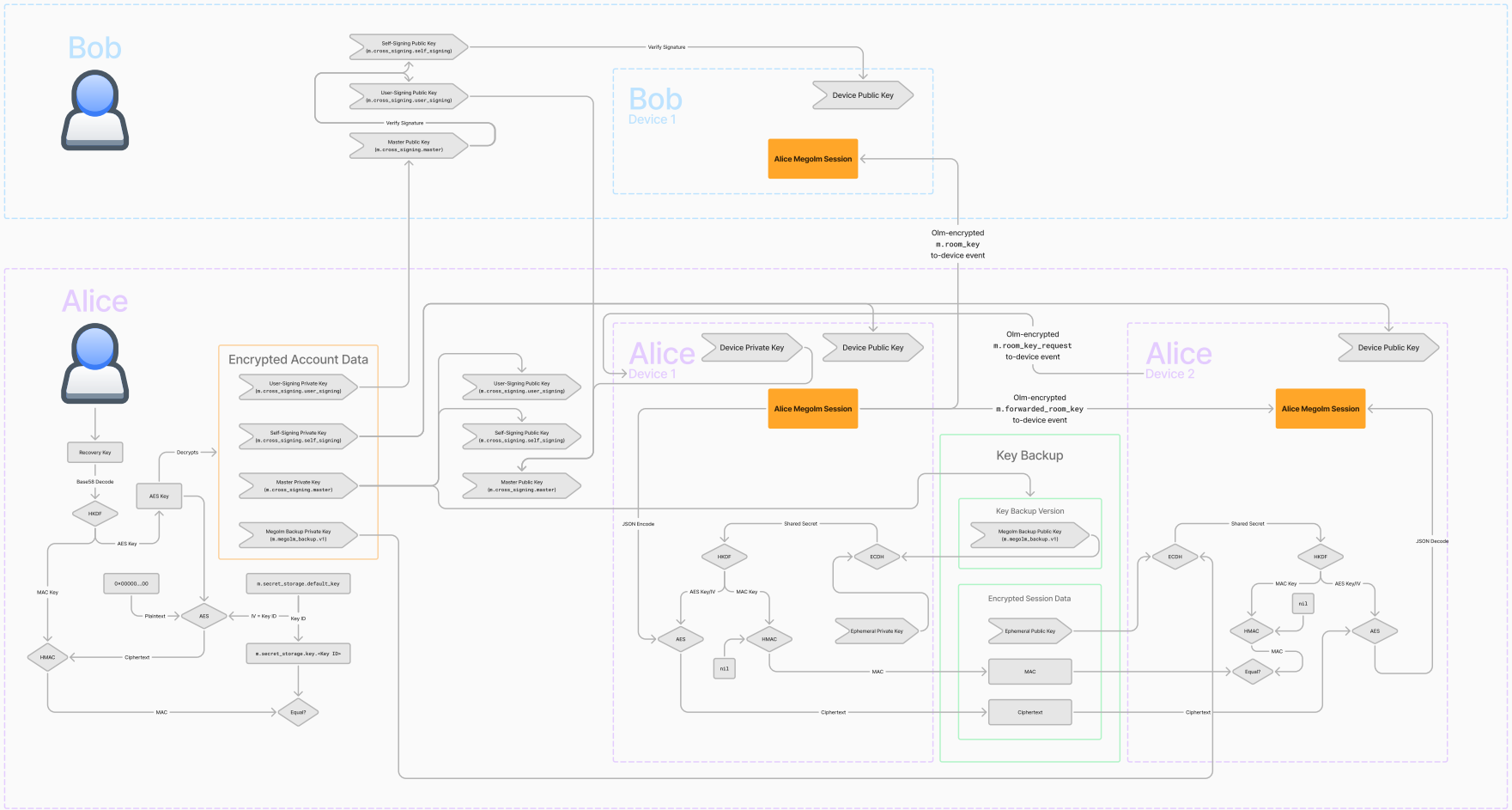
Matrix uses cryptography for two main purposes:

1. **Message Security** — only the people who are part of the conversation should be allowed to view messages of the conversation.
2. **Identity** — verifying that a user or device is who they say they are.

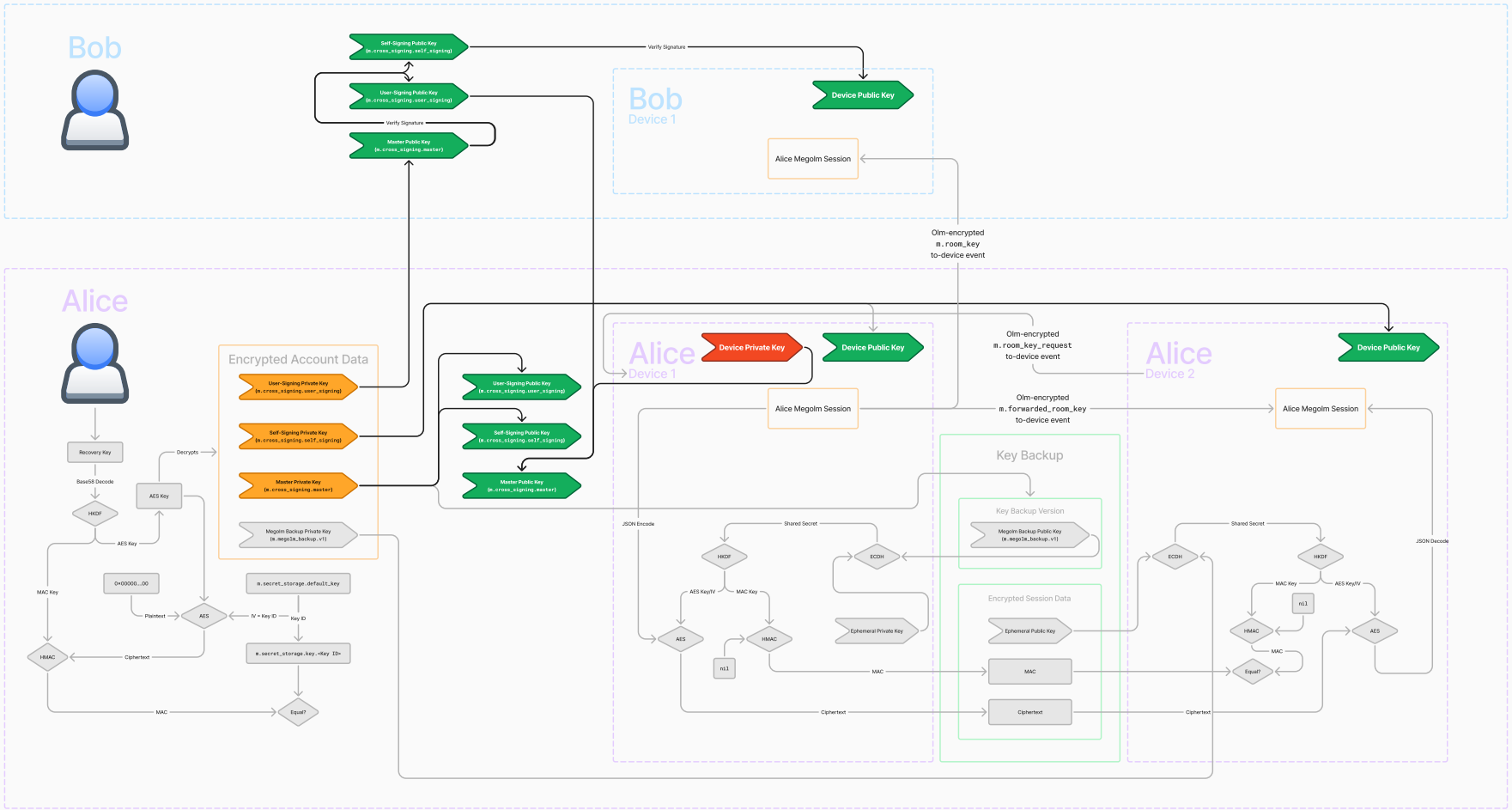
Big Picture


Bob
Self-Signing Public Key
(m.cross_signing.self_signing)
Verify Signature
User-Signing Public Key
(m.cross_signing.user_signing)
Verify Signature
Master Public Key
(m.cross_signing.master)
Bob
Device 1
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key
to-device event
Alice
Encrypted Account Data
User-Signing Private Key
(m.cross_signing.user_signing)
Self-Signing Private Key
(m.cross_signing.self_signing)
Master Private Key
(m.cross_signing.master)
Megolm Backup Private Key
(m.megolm_backup.v1)
Recovery Key
Decrypts
Base58 Decode
HKDF
AES Key
AES Key
MAC Key
0x00000...00
Plaintext
AES
IV = Key ID
Key ID
m.secret_storage.default_key
m.secret_storage.key.<Key ID>
HMAC
Ciphertext
MAC
Equal?
User-Signing Public Key
(m.cross_signing.user_signing)
Self-Signing Public Key
(m.cross_signing.self_signing)
Master Public Key
(m.cross_signing.master)
Alice
Device 1
Device Private Key
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key_request
to-device event
Olm-encrypted
m.forwarded_room_key
to-device event
Alice
Device 2
Device Public Key
Alice Megolm Session
Key Backup
Key Backup Version
Megolm Backup Public Key
(m.megolm_backup.v1)
Encrypted Session Data
Ephemeral Public Key
MAC
Ciphertext
Shared Secret
HKDF
ECDH
JSON Encode
AES Key/IV
MAC Key
AES
HMAC
nil
Ciphertext
MAC
Ephemeral Private Key
Shared Secret
ECDH
HKDF
MAC Key
nil
AES Key/IV
HMAC
AES
MAC
Ciphertext
Equal?
JSON Decode

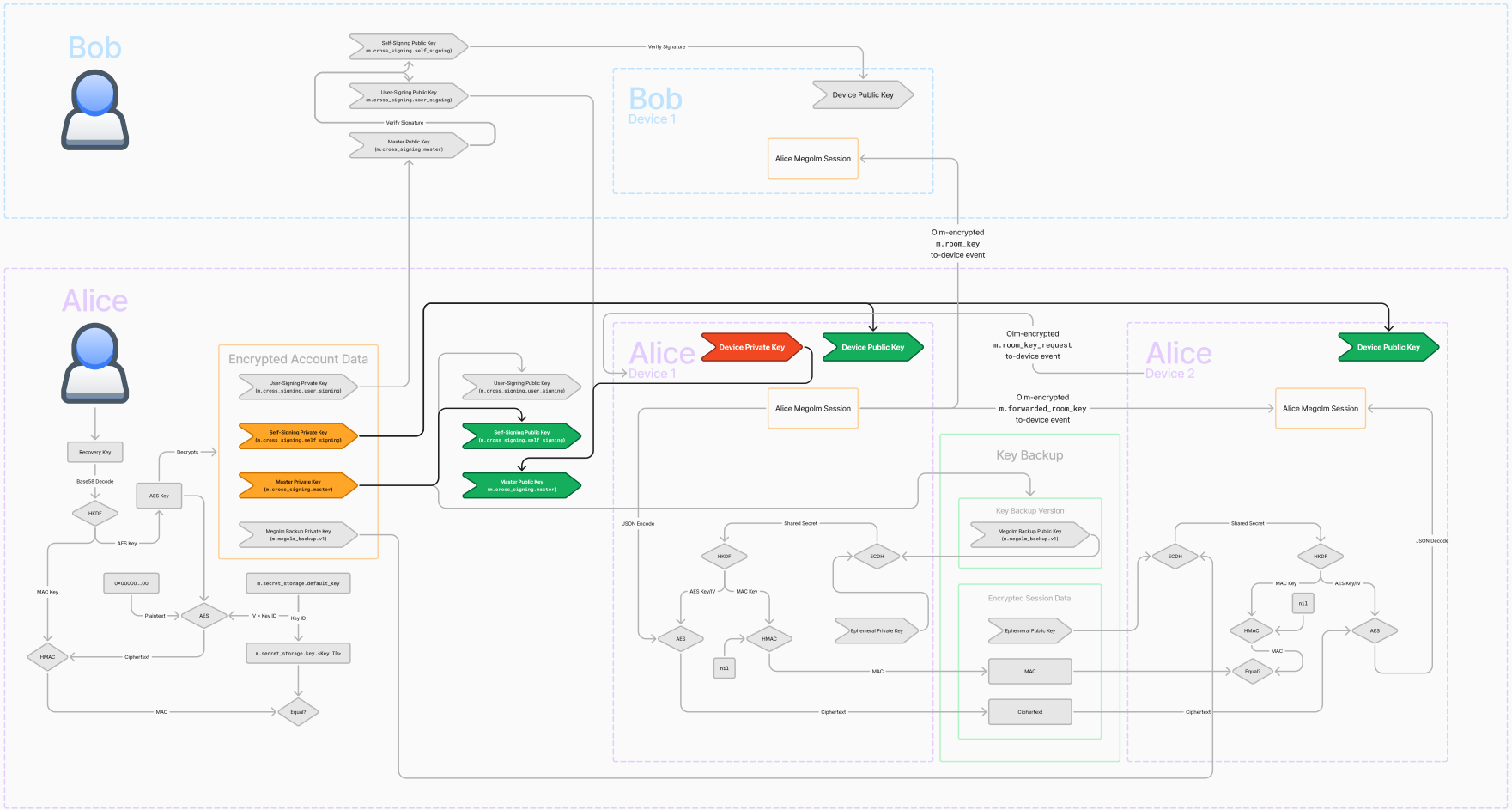
Big Picture: Message Security



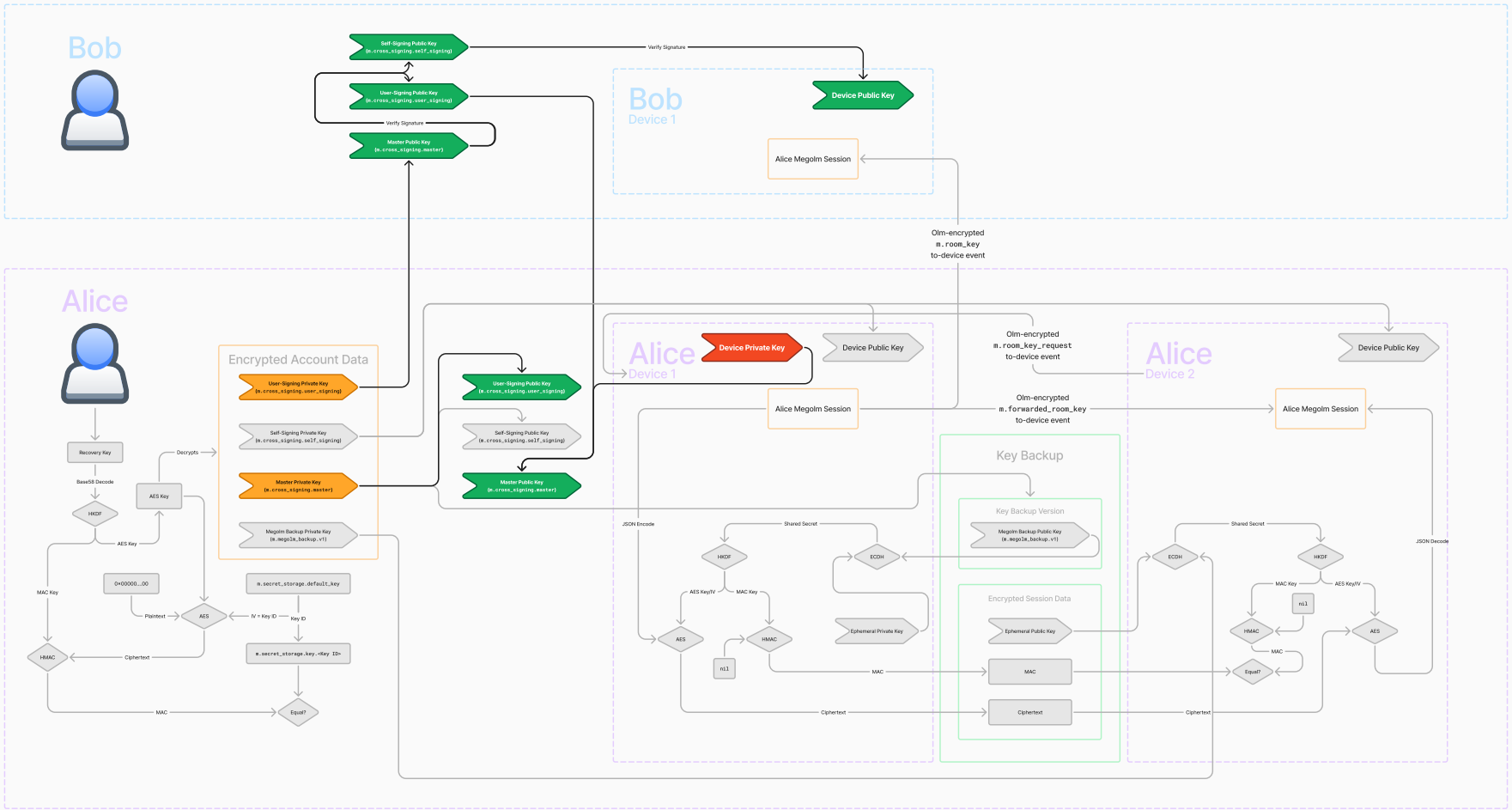
Big Picture: Identity



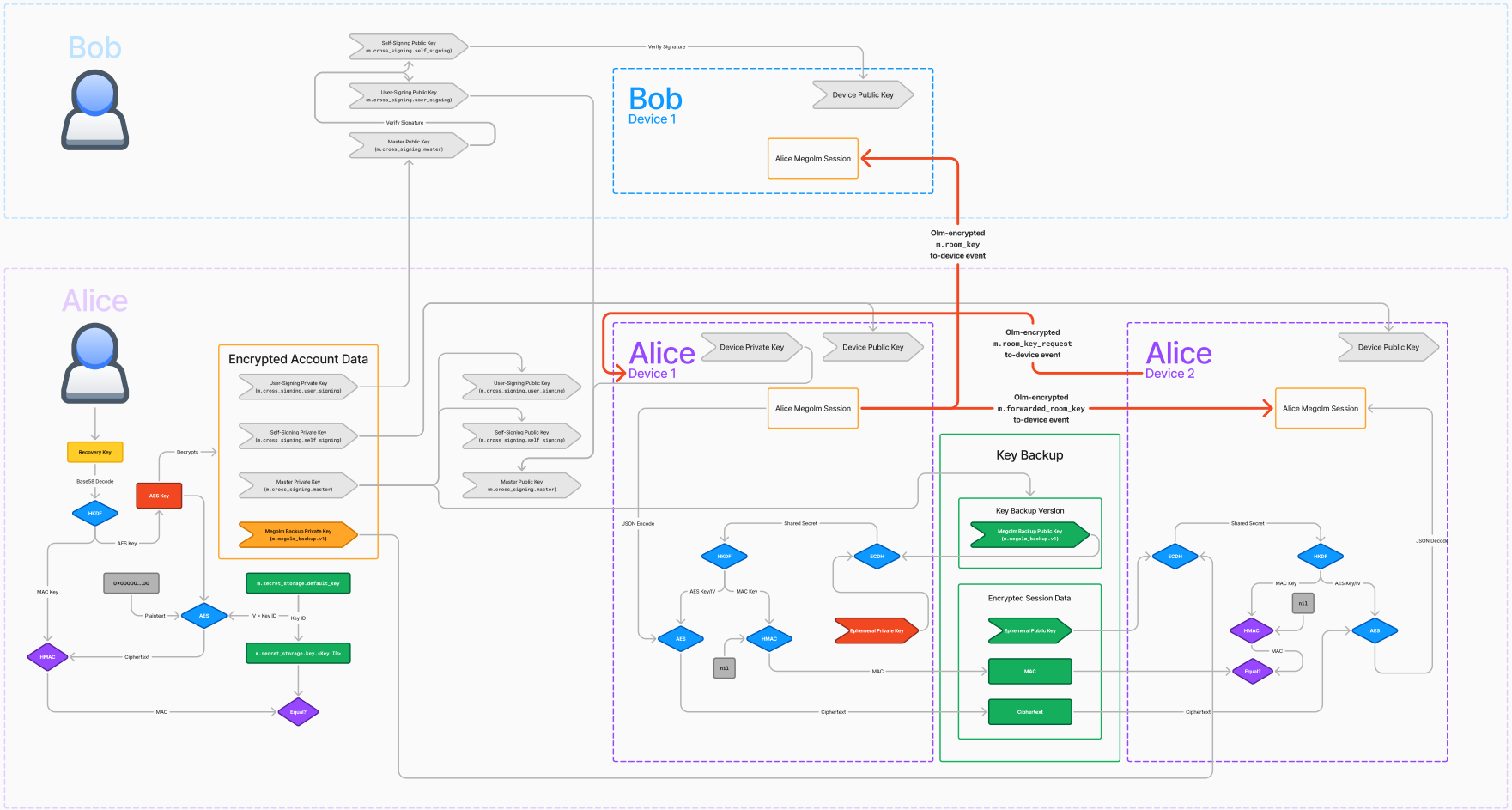
Big Picture: Identity: Device Verification



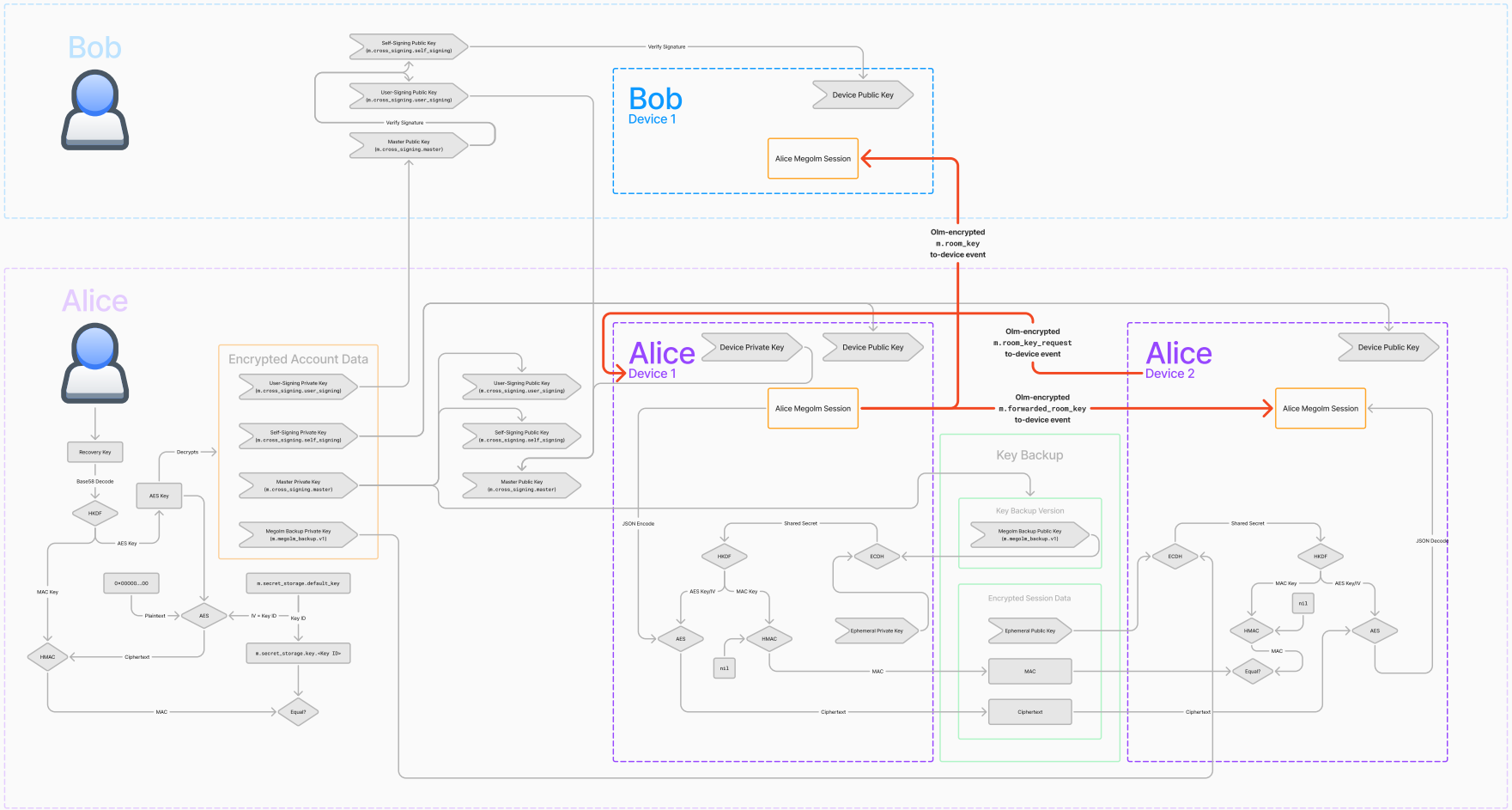
Big Picture: Identity: User Verification



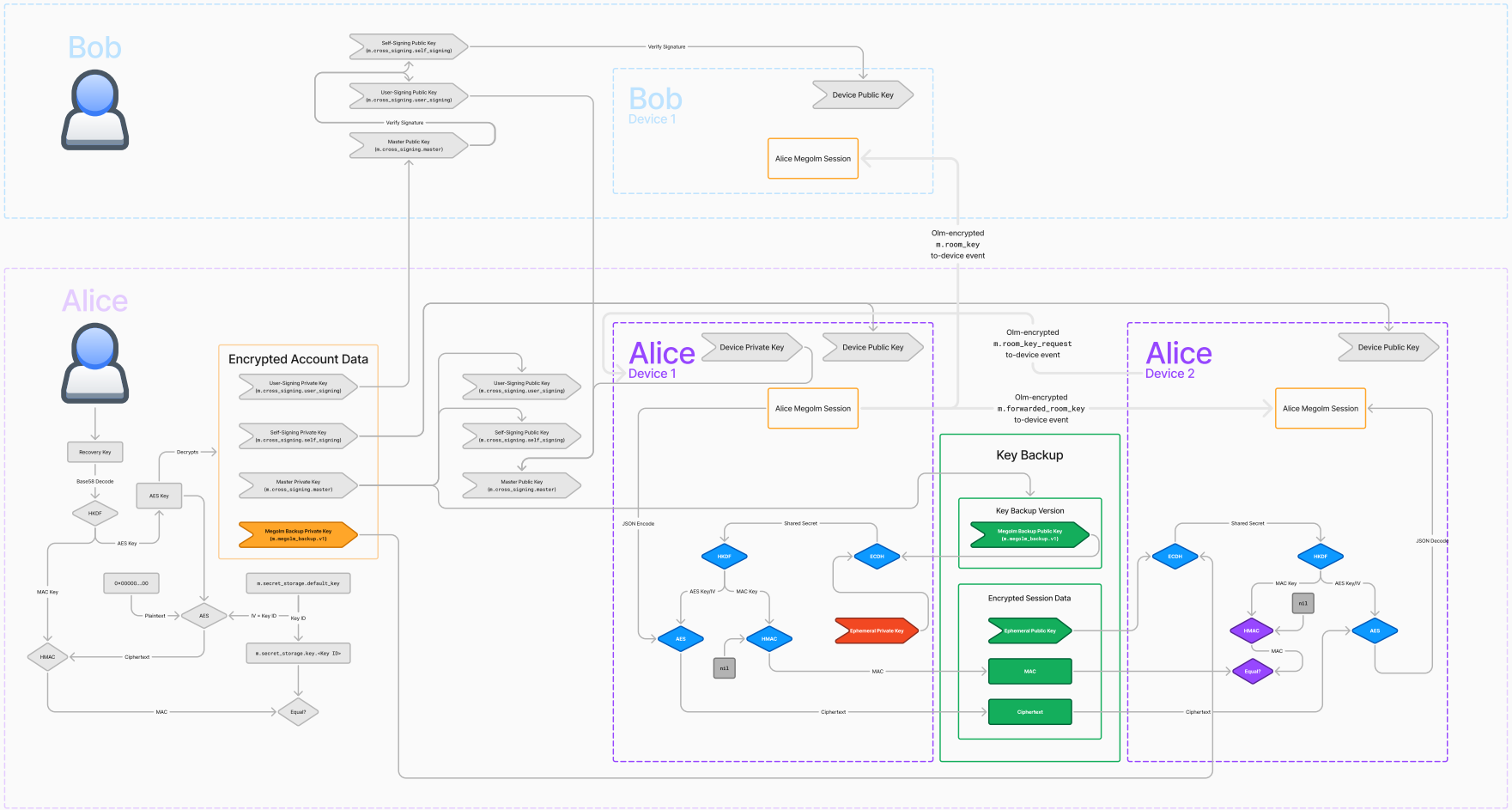
Big Picture: The Other Stuff



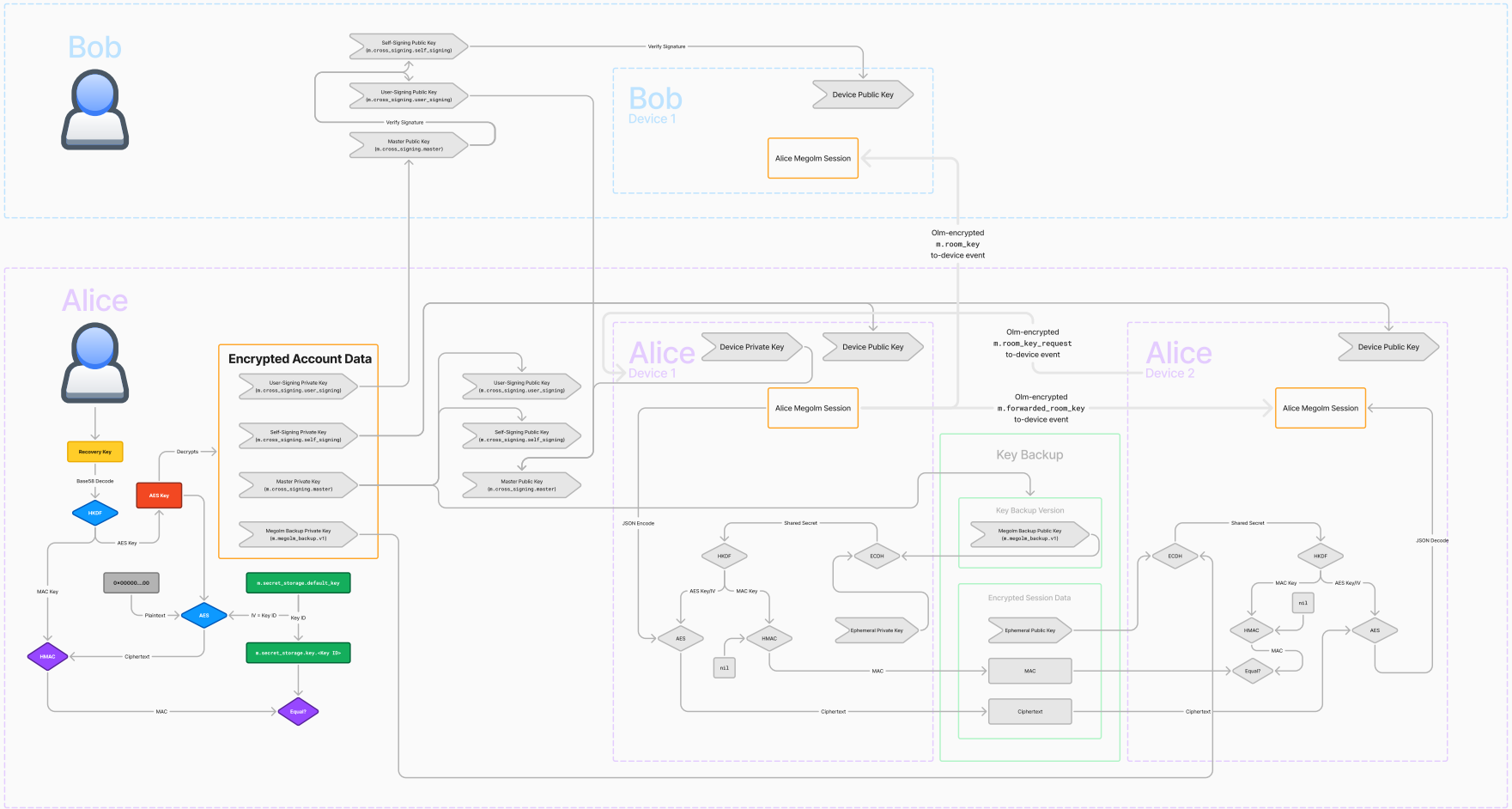
Big Picture: The Other Stuff: To-Device



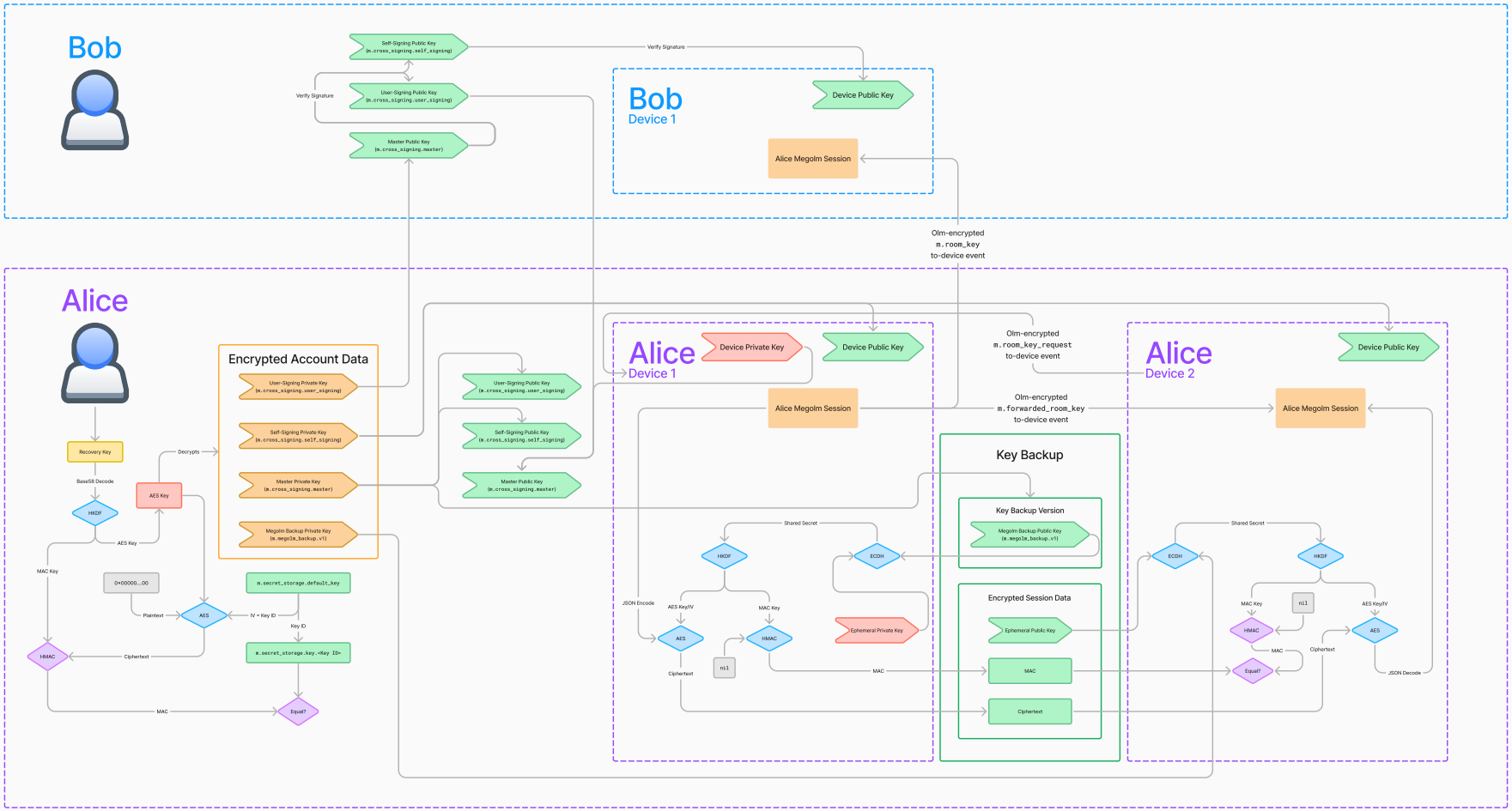
Big Picture: The Other Stuff: Key Backup



Big Picture: The Other Stuff: Secure Secret Storage and Sharing



Big Picture


Bob
Self-Signing Public Key
(m.cross_signing.self_signing)
Verify Signature
User-Signing Public Key
(m.cross_signing.user_signing)
Verify Signature
Master Public Key
(m.cross_signing.master)
Bob
Device 1
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key
to-device event
Alice
Encrypted Account Data
User-Signing Private Key
(m.cross_signing.user_signing)
Self-Signing Private Key
(m.cross_signing.self_signing)
Master Private Key
(m.cross_signing.master)
Megolm Backup Private Key
(m.megolm_backup.v1)
Recovery Key
Decrypts
Base58 Decode
AES Key
HKDF
AES Key
MAC Key
0x00000...00
Plaintext
AES
IV = Key ID
Key ID
m.secret_storage.default_key
m.secret_storage.key.<Key ID>
HMAC
Ciphertext
MAC
Equal?
User-Signing Public Key
(m.cross_signing.user_signing)
Self-Signing Public Key
(m.cross_signing.self_signing)
Master Public Key
(m.cross_signing.master)
Alice
Device 1
Device Private Key
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key_request
to-device event
Olm-encrypted
m.forwarded_room_key
to-device event
Alice
Device 2
Device Public Key
Alice Megolm Session
Key Backup
Key Backup Version
Megolm Backup Public Key
(m.megolm_backup.v1)
Encrypted Session Data
Ephemeral Public Key
MAC
Ciphertext
Shared Secret
HKDF
ECDH
JSON Encode
AES Key/IV
MAC Key
AES
HMAC
nil
Ciphertext
Ephemeral Private Key
MAC
Shared Secret
ECDH
HKDF
MAC Key
nil
AES Key/IV
HMAC
AES
MAC
Ciphertext
Equal?
JSON Decode

Cryptography Crash Course

Encryption: Symmetric vs Asymmetric

There are two main categories of encryption schemes:

- **Symmetric** — both **the encryptor and the decryptor share the same key** and that key is used in both the encryption and decryption of the message
- **Asymmetric** — **the encryptor needs the public key, and the decryptor needs the private key** and the encryptor encrypts the message with the public key, and the private key is required to decrypt the message

Encryption: Symmetric vs Asymmetric

There are two main categories of encryption schemes:

- **Symmetric** — both **the encryptor and the decryptor share the same key** and that key is used in both the encryption and decryption of the message
- **Asymmetric** — **the encryptor needs the public key, and the decryptor needs the private key** and the encryptor encrypts the message with the public key, and the private key is required to decrypt the message

Asymmetric Signatures

In addition to providing encryption, asymmetric encryption schemes also provide **signatures**.

Signing uses the *private* key, and anyone who possesses the *public* key can verify the signature.

Asymmetric Signatures

In addition to providing encryption, asymmetric encryption schemes also provide **signatures**.

Signing uses the *private* key, and anyone who possesses the *public* key can verify the signature.

Hashes and HMAC

A **cryptographic hash function** is a one-directional function which takes an arbitrarily large set of data and produces a unique fixed-size output (called the hash).

Given the same data, a hash function will always return the same output.

This allows us to verify that the data did not change in transit (for example, by a malicious actor).

Hashes are vulnerable to **metadata attacks**. To prevent these, we use HMAC which adds a secret key to the hash.

Hashes and HMAC

A **cryptographic hash function** is a one-directional function which takes an arbitrarily large set of data and produces a unique fixed-size output (called the hash).

Given the same data, a hash function will always return the same output.

This allows us to verify that the data did not change in transit (for example, by a malicious actor).

Hashes are vulnerable to **metadata attacks**. To prevent these, we use HMAC which adds a secret key to the hash.

Hashes and HMAC

A **cryptographic hash function** is a one-directional function which takes an arbitrarily large set of data and produces a unique fixed-size output (called the hash).

Given the same data, a hash function will always return the same output.

This allows us to verify that the data did not change in transit (for example, by a malicious actor).

Hashes are vulnerable to **metadata attacks**. To prevent these, we use HMAC which adds a secret key to the hash.

Hashes and HMAC

A **cryptographic hash function** is a one-directional function which takes an arbitrarily large set of data and produces a unique fixed-size output (called the hash).

Given the same data, a hash function will always return the same output.

This allows us to verify that the data did not change in transit (for example, by a malicious actor).

Hashes are vulnerable to **metadata attacks**. To prevent these, we use HMAC which adds a secret key to the hash.

Key-Derivation Functions (HKDF)

Sometimes, we want to turn a small key into a larger key (or set of larger keys).

Key-Derivation Functions (KDFs) are used to do this.

Matrix uses HKDF which uses HMAC for the key derivation process.

Key-Derivation Functions (HKDF)

Sometimes, we want to turn a small key into a larger key (or set of larger keys).

Key-Derivation Functions (KDFs) are used to do this.

Matrix uses HKDF which uses HMAC for the key derivation process.

Diffie-Hellman Key Exchanges

Often, we need a way to share keys with both the sending and receiving parties across an unsecured channel.

Diffie-Hellman is a method for using public-key cryptography to facilitate keysharing.

$$\textbf{ECDH}(A_{private}, B_{public}) = \textbf{ECDH}(B_{private}, A_{public}) = K_{shared}.$$

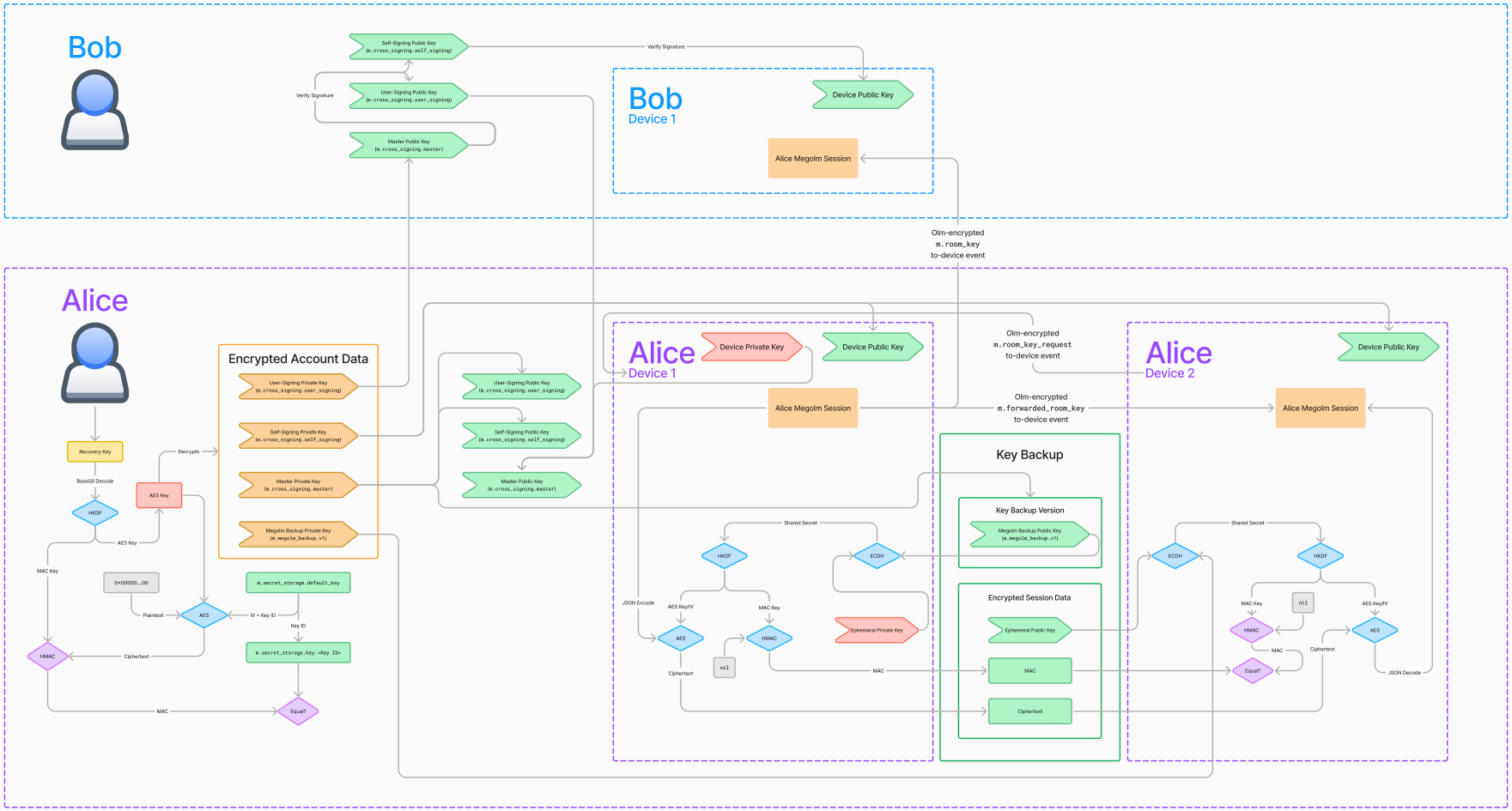
Diffie-Hellman Key Exchanges

Often, we need a way to share keys with both the sending and receiving parties across an unsecured channel.

Diffie-Hellman is a method for using public-key cryptography to facilitate keysharing.

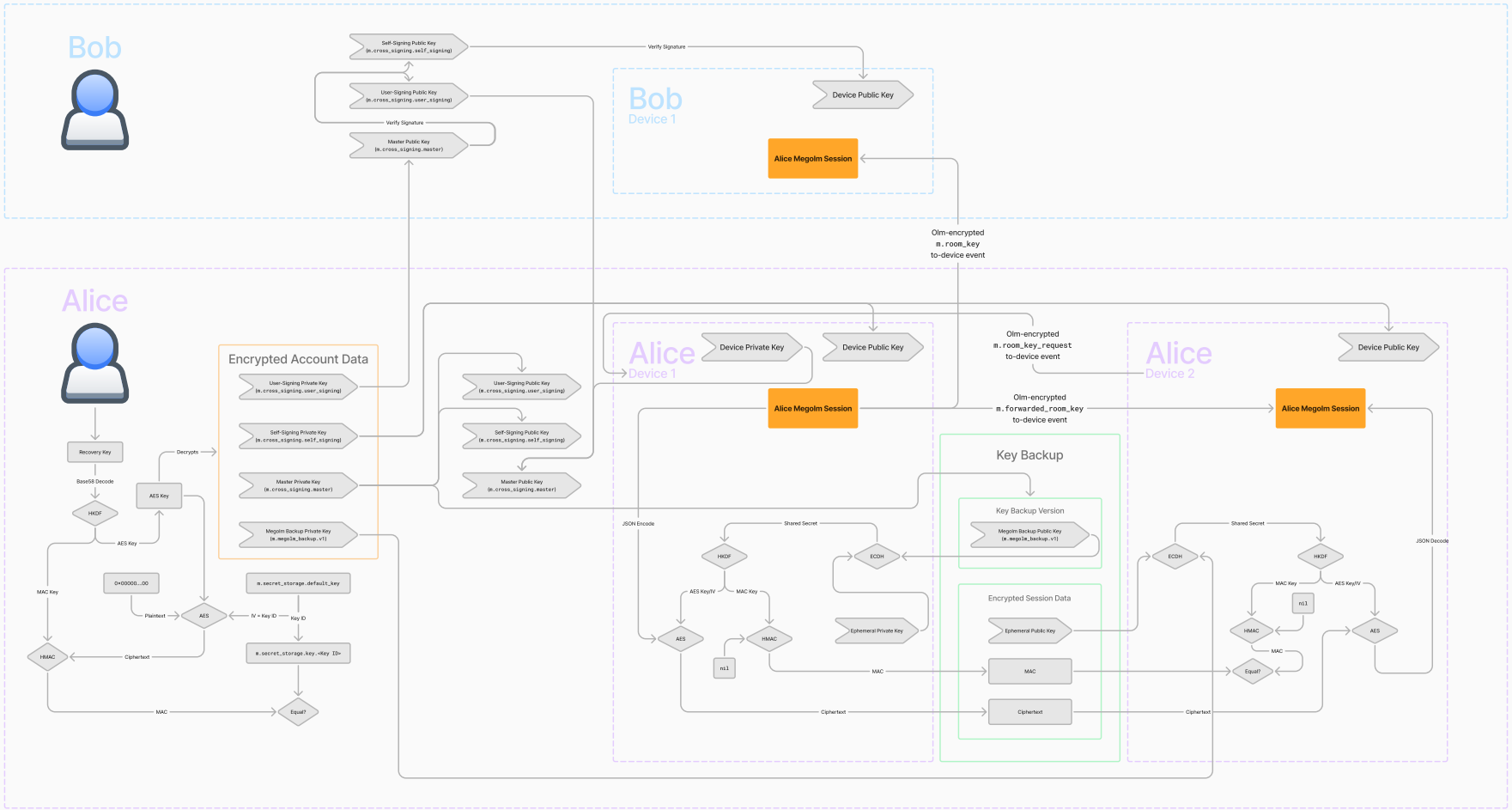
$$\textbf{ECDH}(A_{private}, B_{public}) = \textbf{ECDH}(B_{private}, A_{public}) = K_{shared}.$$

Big Picture

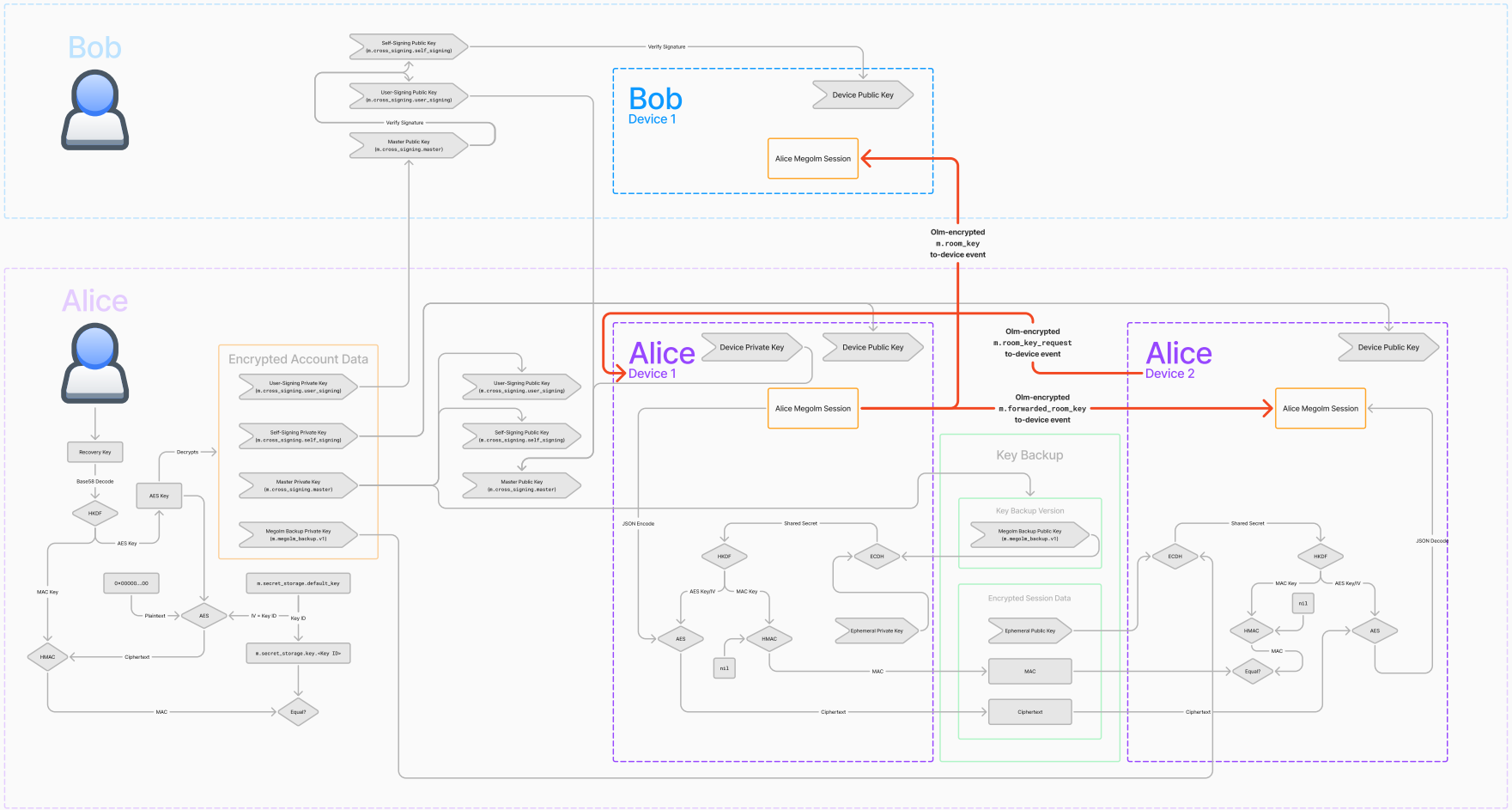

Bob
Self-Signing Public Key
(m.cross_signing.self_signing)
Verify Signature
User-Signing Public Key
(m.cross_signing.user_signing)
Verify Signature
Master Public Key
(m.cross_signing.master)
Bob
Device 1
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key
to-device event
Alice
Encrypted Account Data
User-Signing Private Key
(m.cross_signing.user_signing)
Self-Signing Private Key
(m.cross_signing.self_signing)
Master Private Key
(m.cross_signing.master)
Megolm Backup Private Key
(m.megolm_backup.v1)
Recovery Key
Decrypts
Base58 Decode
HKDF
AES Key
AES Key
MAC Key
0x00000...00
Plaintext
AES
IV = Key ID
Key ID
m.secret_storage.default_key
m.secret_storage.key.<Key ID>
HMAC
Ciphertext
MAC
Equal?
User-Signing Public Key
(m.cross_signing.user_signing)
Self-Signing Public Key
(m.cross_signing.self_signing)
Master Public Key
(m.cross_signing.master)
Alice
Device 1
Device Private Key
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key_request
to-device event
Olm-encrypted
m.forwarded_room_key
to-device event
Alice
Device 2
Device Public Key
Alice Megolm Session
Key Backup
Key Backup Version
Megolm Backup Public Key
(m.megolm_backup.v1)
Encrypted Session Data
Ephemeral Public Key
MAC
Ciphertext
Shared Secret
HKDF
ECDH
JSON Encode
AES Key/IV
MAC Key
AES
HMAC
nil
Ciphertext
Ephemeral Private Key
MAC
Shared Secret
ECDH
HKDF
MAC Key
nil
AES Key/IV
HMAC
AES
MAC
Ciphertext
Equal?
JSON Decode

Sharing Keys

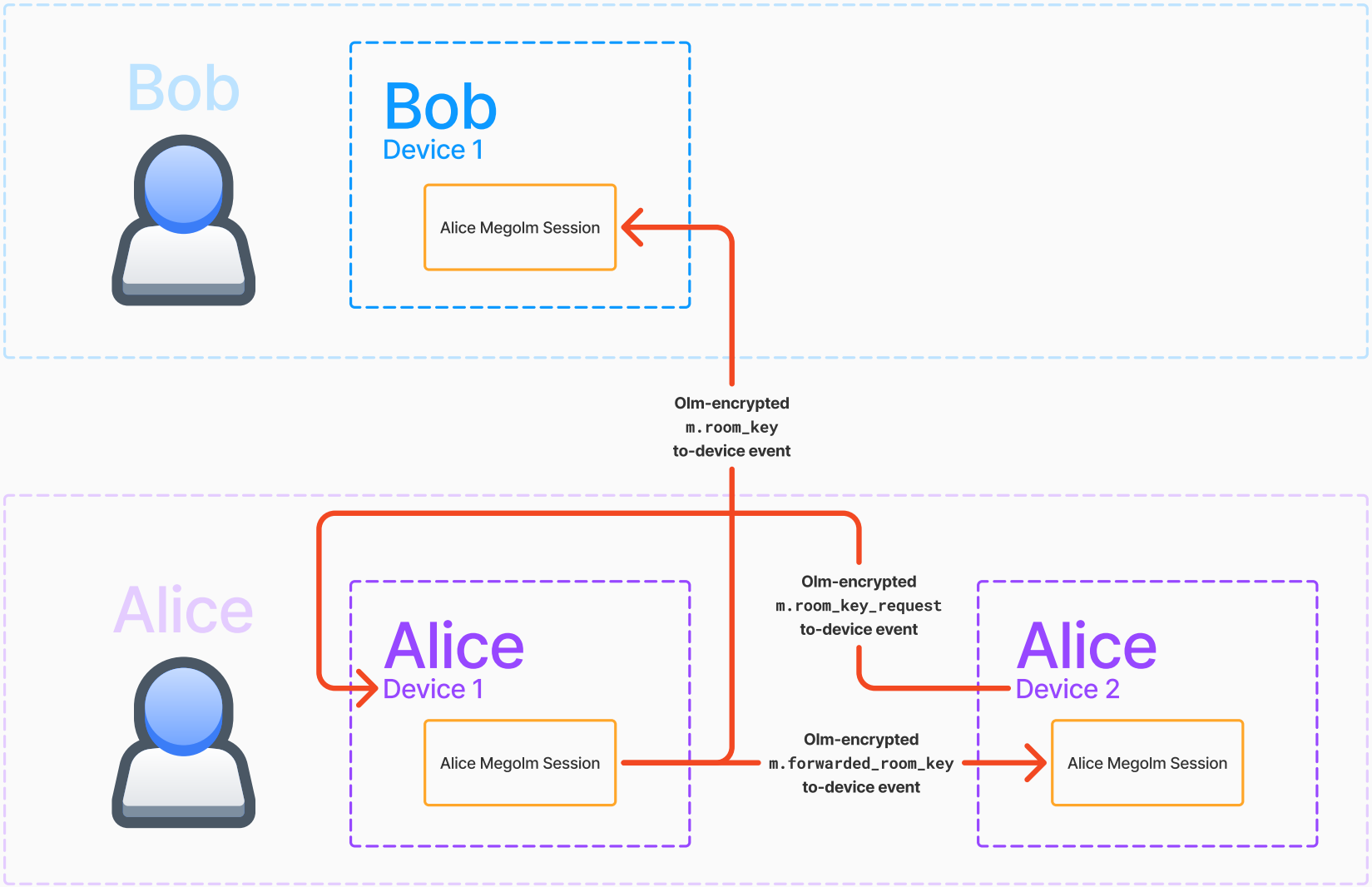
Big Picture: Message Security



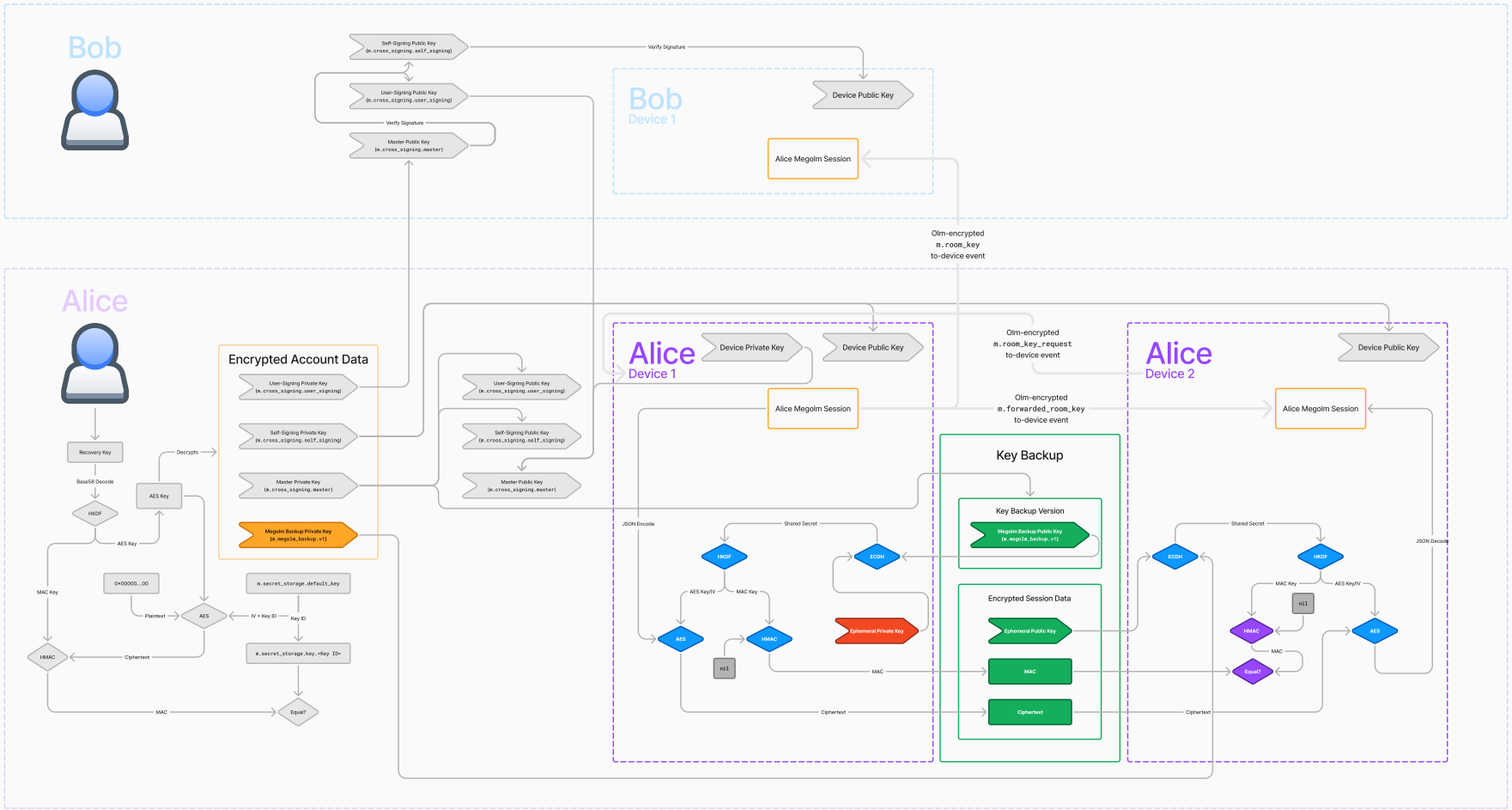
Encrypted Olm Events



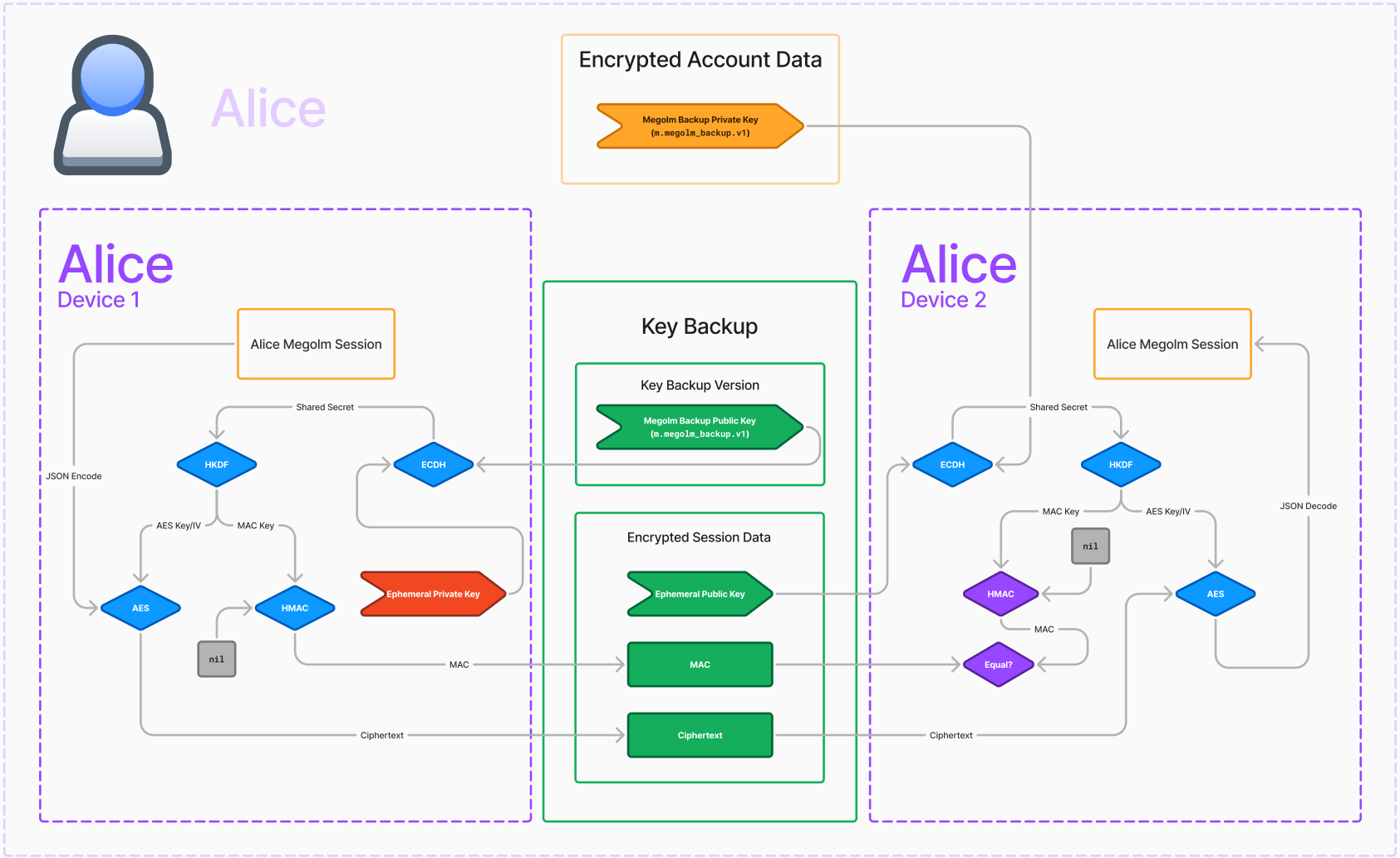
Encrypted Olm Events



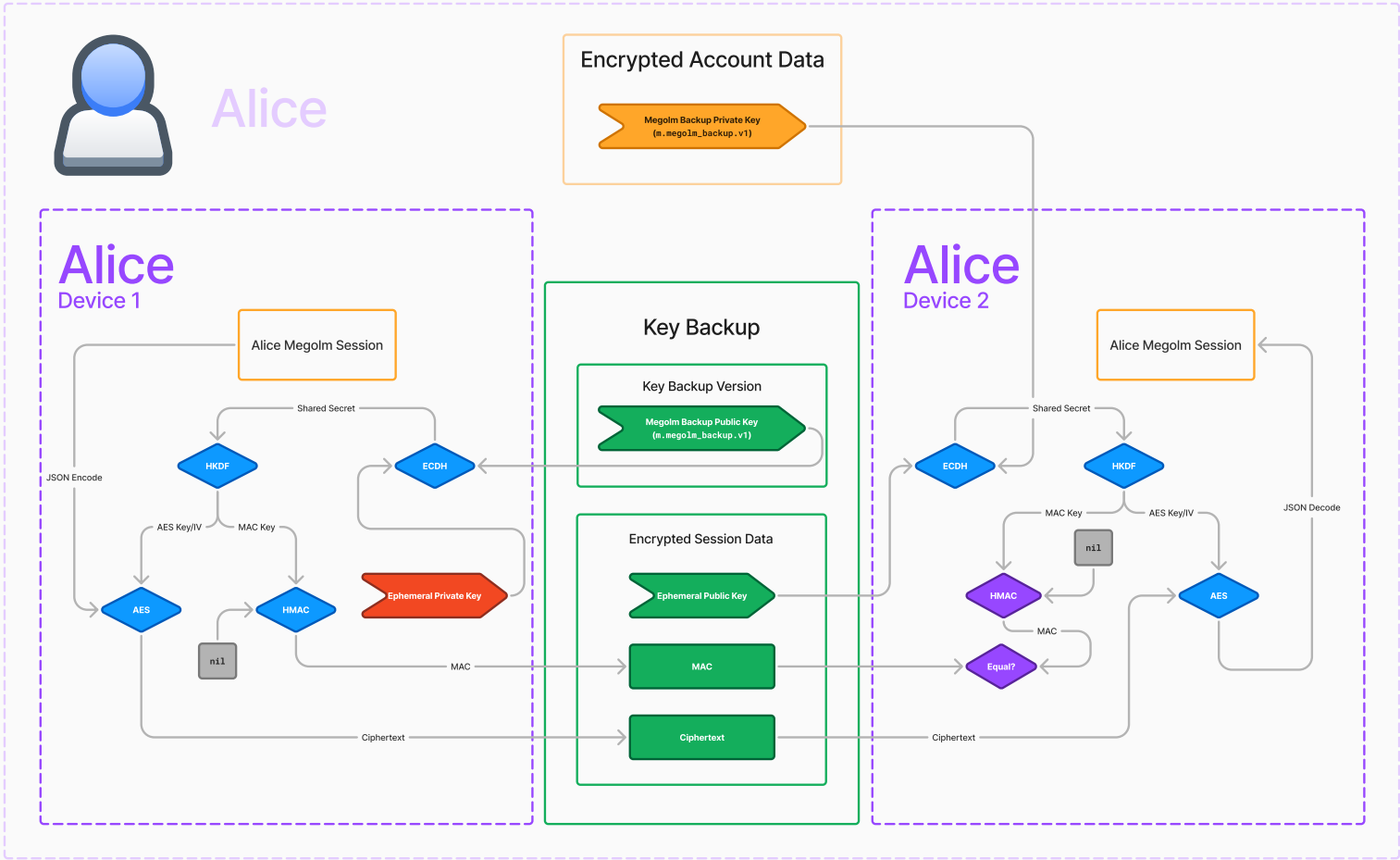
Key Backup


Bob
Self-Signing Public Key
Verify Signature
User-Signing Public Key
Verify Signature
Master Public Key
Bob
Device 1
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key
to-device event
Alice
Encrypted Account Data
Recovery Key
Decrypts
Base58 Decode
AES Key
HKDF
AES Key
MAC Key
0x00000...00
Plaintext
AES
IV + Key ID
Key ID
m.secret_storage.default_key
m.secret_storage.key.<Key ID>
HMAC
Ciphertext
MAC
Equal?
User-Signing Private Key
Self-Signing Private Key
Master Private Key
Megolm Backup Private Key
User-Signing Public Key
Self-Signing Public Key
Master Public Key
Alice
Device 1
Device Private Key
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key_request
to-device event
Olm-encrypted
m.forwarded_room_key
to-device event
Alice
Device 2
Device Public Key
Alice Megolm Session
Key Backup
Key Backup Version
Megolm Backup Public Key
Encrypted Session Data
Ephemeral Public Key
MAC
Ciphertext
JSON Encode
Shared Secret
HKDF
ECDH
AES Key/IV
MAC Key
AES
HMAC
Ephemeral Private Key
n11
MAC
Ciphertext
Shared Secret
ECDH
HKDF
JSON Decode
MAC Key
AES Key/IV
n11
HMAC
AES
MAC
Equal?
Ciphertext

Key Backup

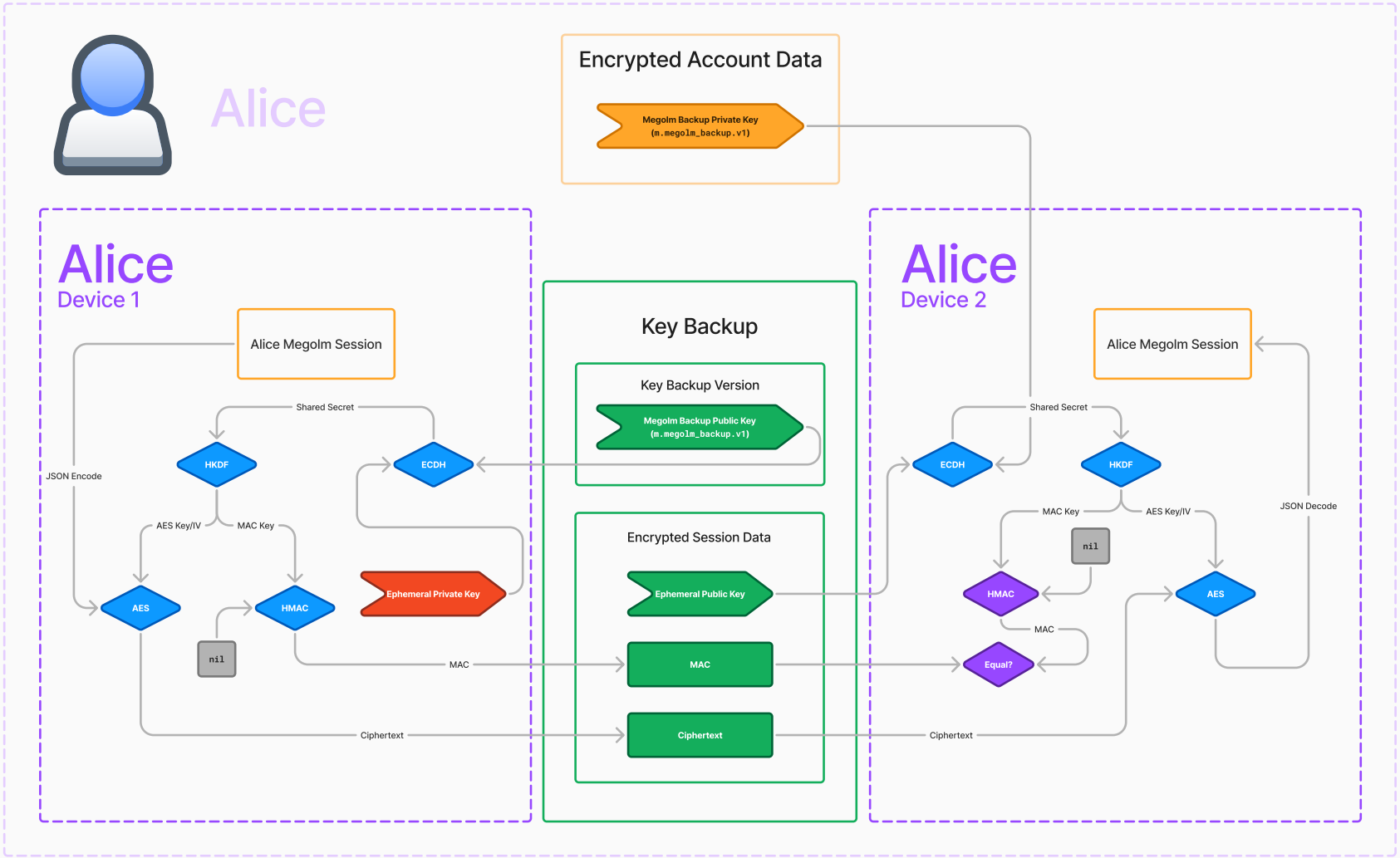

Alice
Encrypted Account Data
Megolm Backup Private Key
(m.megolm_backup.v1)
Alice
Device 1
Alice Megolm Session
Shared Secret
HKDF
ECDH
JSON Encode
AES Key/IV
MAC Key
AES
HMAC
nil
Ephemeral Private Key
MAC
Ciphertext
Key Backup
Key Backup Version
Megolm Backup Public Key
(m.megolm_backup.v1)
Encrypted Session Data
Ephemeral Public Key
MAC
Ciphertext
Alice
Device 2
Alice Megolm Session
Shared Secret
ECDH
HKDF
MAC Key
AES Key/IV
JSON Decode
nil
HMAC
AES
MAC
Equal?
Ciphertext

Key Backup



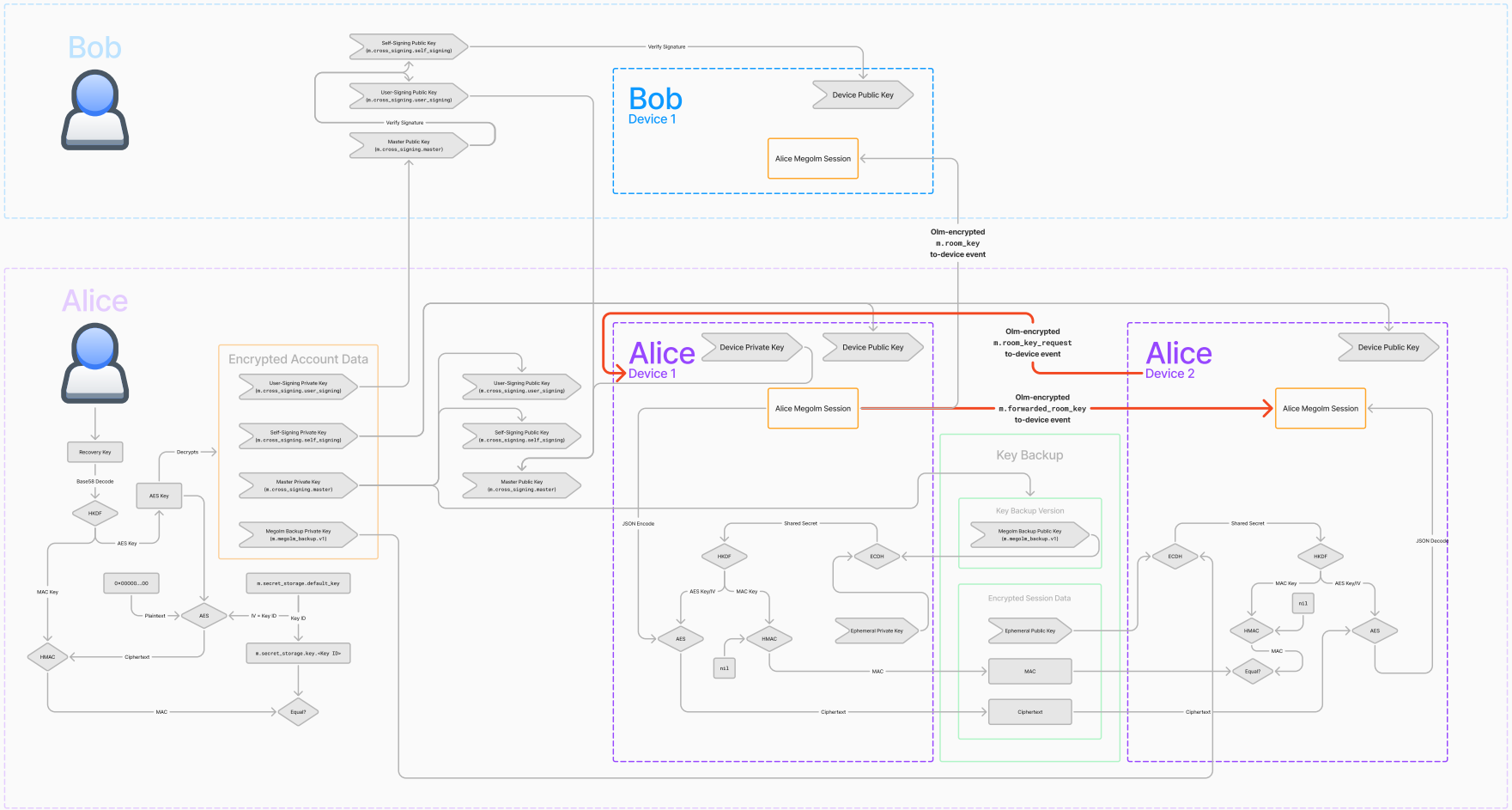
$$\mathbf{ECDH}(A_{private}, B_{public}) = \mathbf{ECDH}(B_{private}, A_{public}) = K_{shared}$$

Key Backup

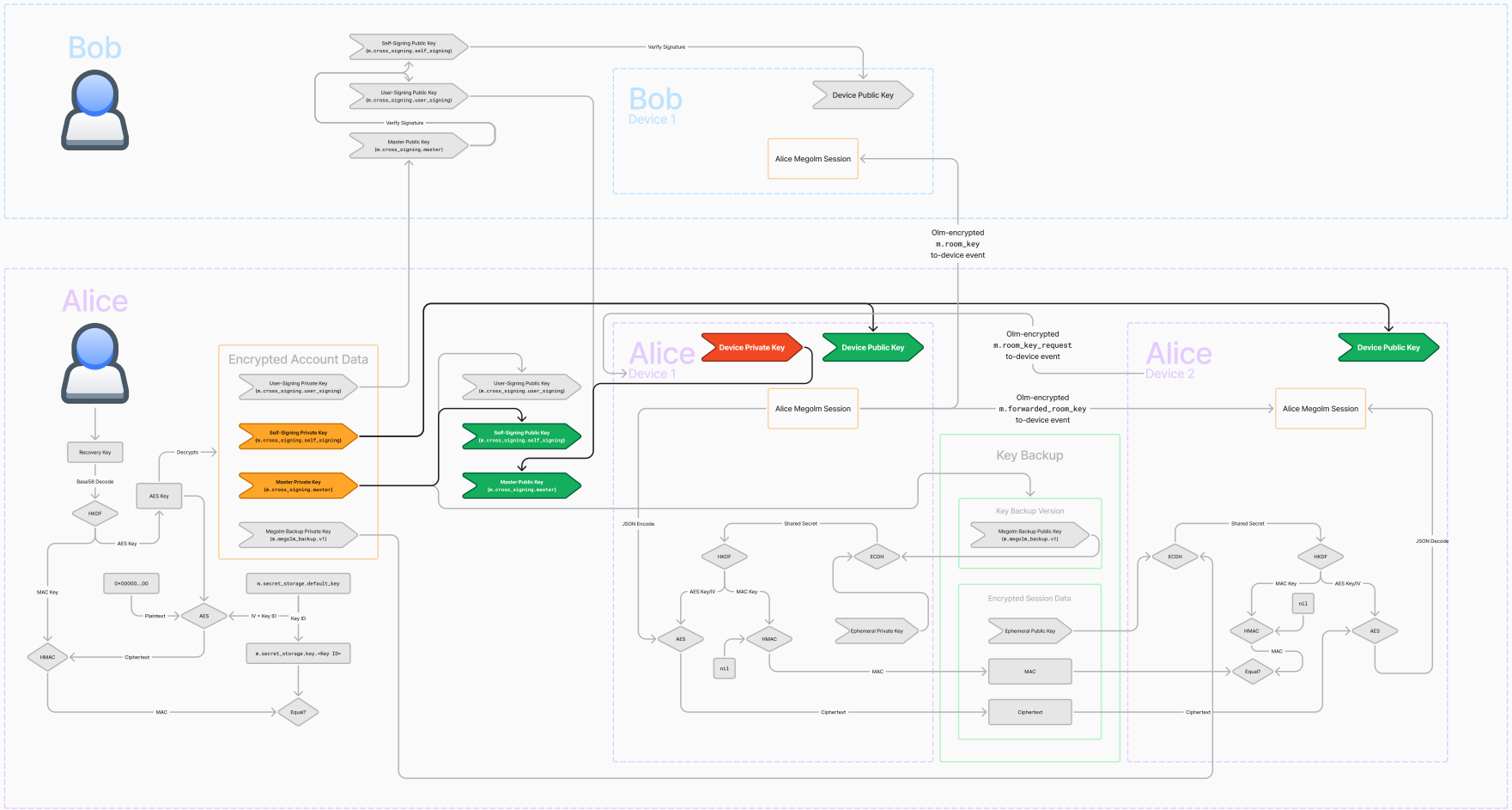

Alice
Encrypted Account Data
Megolm Backup Private Key
(m.megolm_backup.v1)
Alice
Device 1
Alice Megolm Session
Shared Secret
HKDF
ECDH
JSON Encode
AES Key/IV
MAC Key
AES
HMAC
nil
Ephemeral Private Key
MAC
Ciphertext
Key Backup
Key Backup Version
Megolm Backup Public Key
(m.megolm_backup.v1)
Encrypted Session Data
Ephemeral Public Key
MAC
Ciphertext
Alice
Device 2
Alice Megolm Session
Shared Secret
ECDH
HKDF
MAC Key
AES Key/IV
nil
HMAC
AES
MAC
Equal?
JSON Decode
Ciphertext

Device Verification

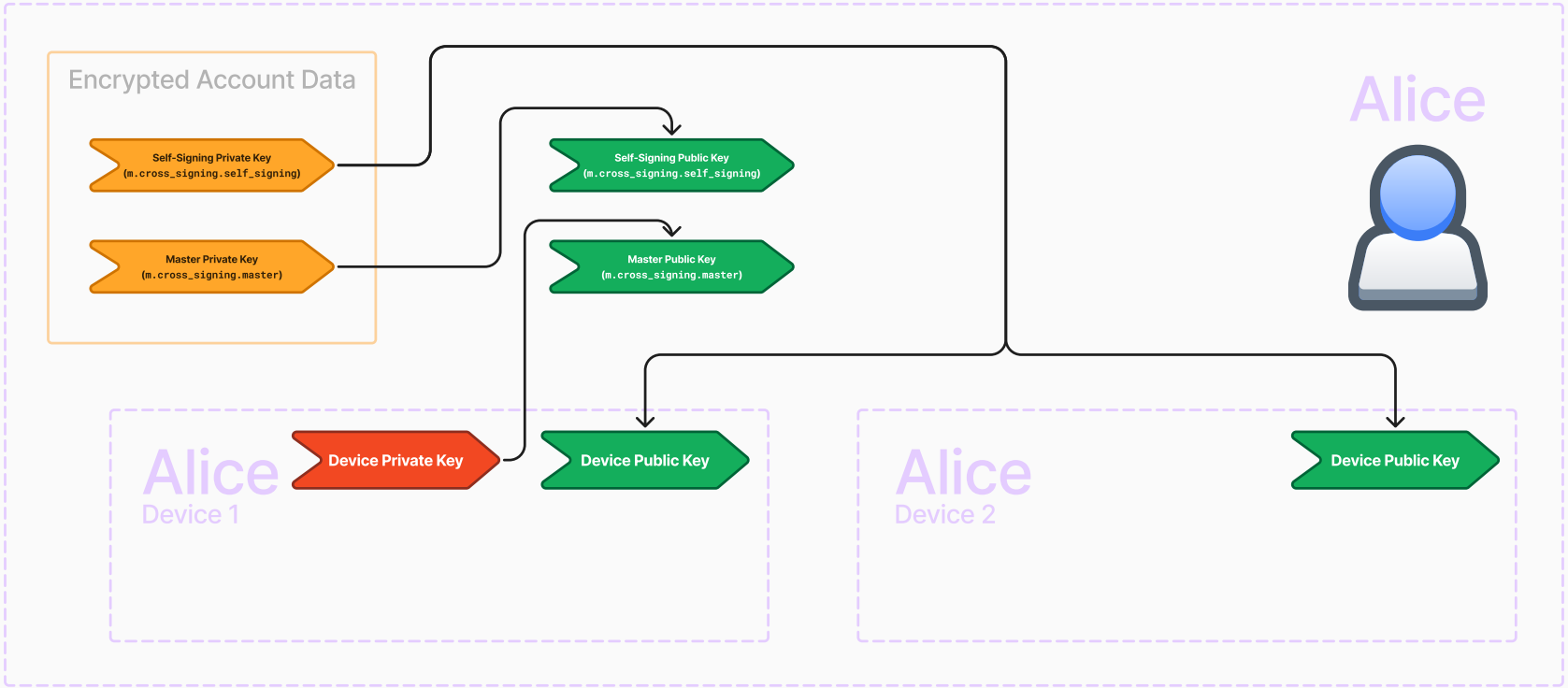
Who Can We Send Keys To?



Signatures

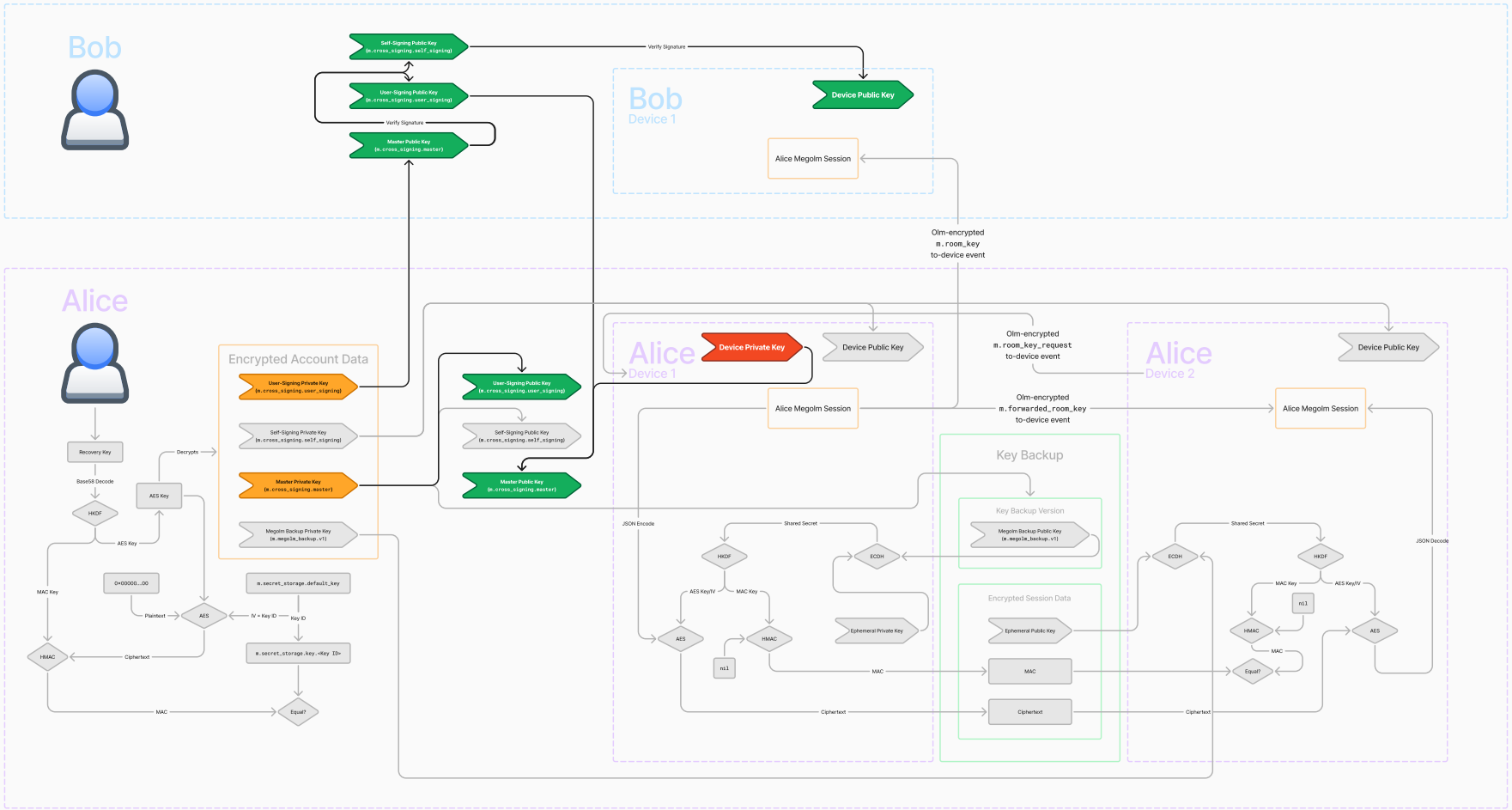

Bob
Self-Signing Public Key
(m.cross_signing.self_signing)
Verify Signature
User-Signing Public Key
(m.cross_signing.user_signing)
Verify Signature
Master Public Key
(m.cross_signing.master)
Bob
Device 1
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key
to-device event
Alice
Encrypted Account Data
User-Signing Private Key
(m.cross_signing.user_signing)
Self-Signing Private Key
(m.cross_signing.self_signing)
Master Private Key
(m.cross_signing.master)
Megolm Backup Private Key
(m.megolm_backup.v1)
Recovery Key
Decrypts
Base58 Decode
AES Key
HKDF
AES Key
MAC Key
0x00000...00
Plaintext
AES
IV + Key ID
Key ID
m.secret_storage.default_key
m.secret_storage.key.<Key ID>
HMAC
Ciphertext
MAC
Equal?
User-Signing Public Key
(m.cross_signing.user_signing)
Self-Signing Public Key
(m.cross_signing.self_signing)
Master Public Key
(m.cross_signing.master)
Alice
Device 1
Device Private Key
Device Public Key
Alice Megolm Session
JSON Encode
Shared Secret
HKDF
ECDH
AES Key
IV
MAC Key
AES
HMAC
nil
Ephemeral Private Key
MAC
Ciphertext
Olm-encrypted
m.room_key_request
to-device event
Olm-encrypted
m.forwarded_room_key
to-device event
Key Backup
Key Backup Version
Megolm Backup Public Key
(m.megolm_backup.v1)
Encrypted Session Data
Ephemeral Public Key
MAC
Ciphertext
Alice
Device 2
Device Public Key
Alice Megolm Session
Shared Secret
ECDH
HKDF
MAC Key
AES Key
IV
nil
HMAC
AES
MAC
Equal?
JSON Decode
Ciphertext

Signatures

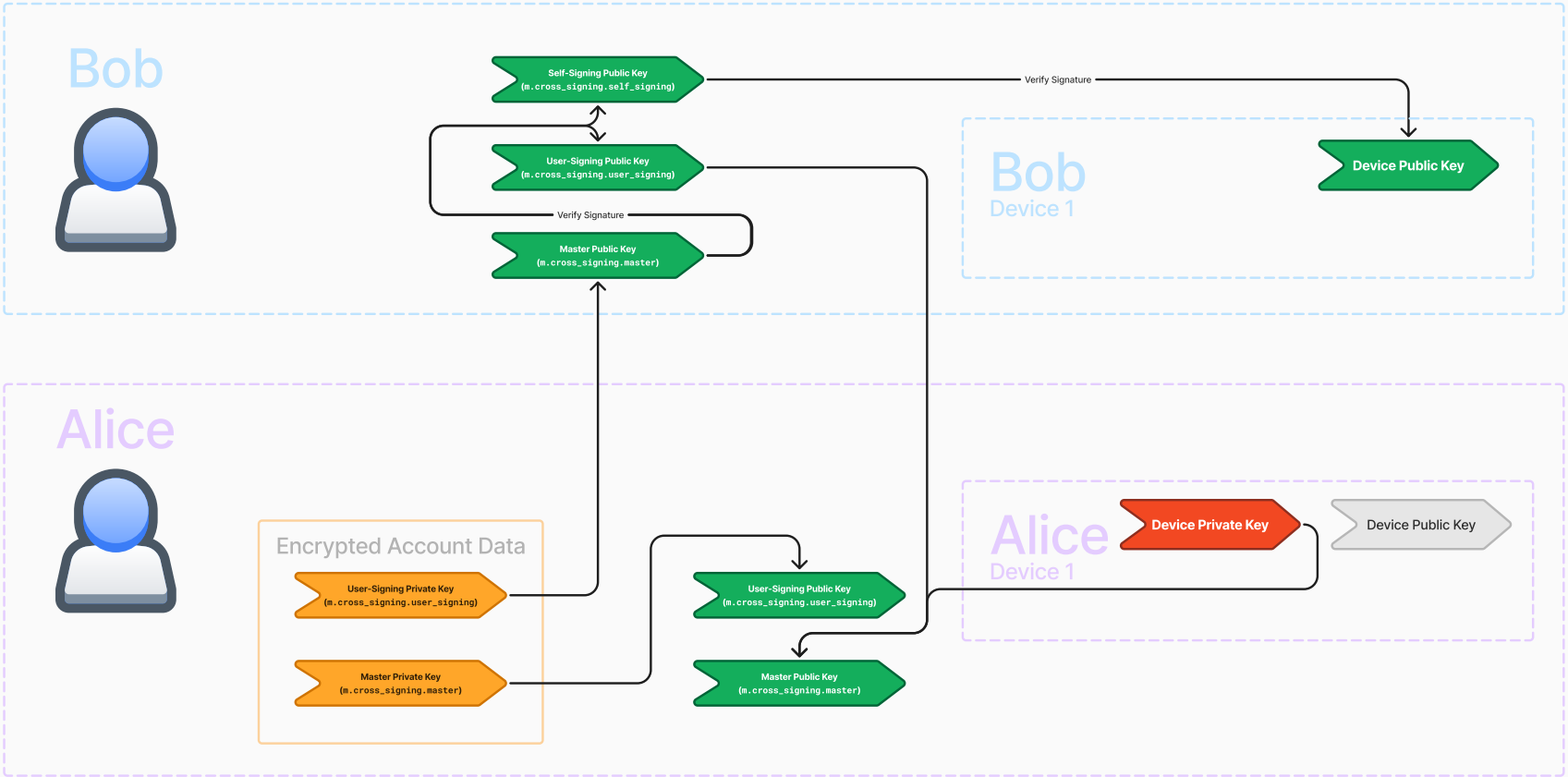

Encrypted Account Data
Self-Signing Private Key
(m.cross_signing.self_signing)
Master Private Key
(m.cross_signing.master)
Self-Signing Public Key
(m.cross_signing.self_signing)
Master Public Key
(m.cross_signing.master)
Alice
Alice
Device 1
Device Private Key
Device Public Key
Alice
Device 2
Device Public Key

User Verification

Additional Identity Verification

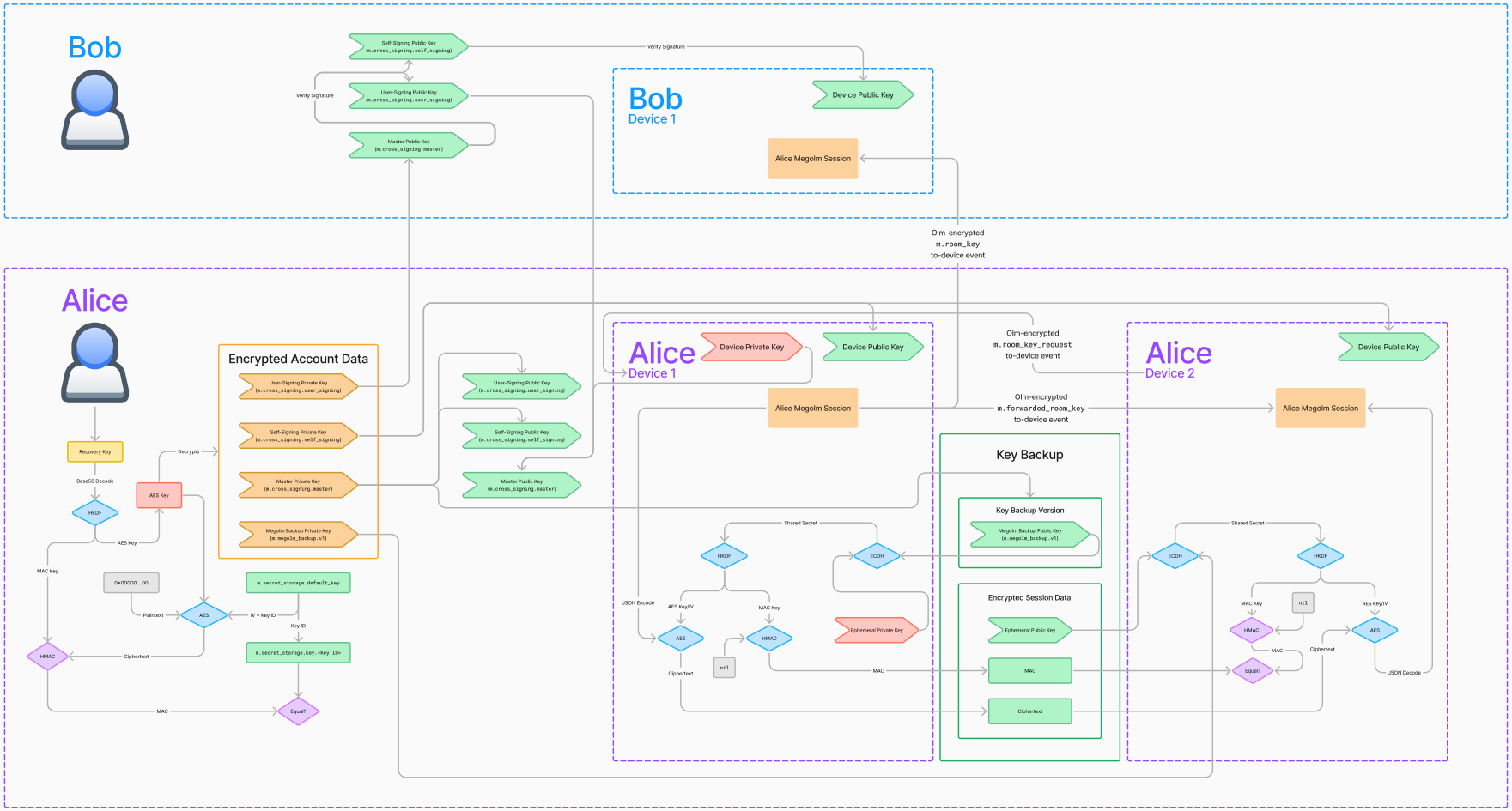


Additional Identity Verification

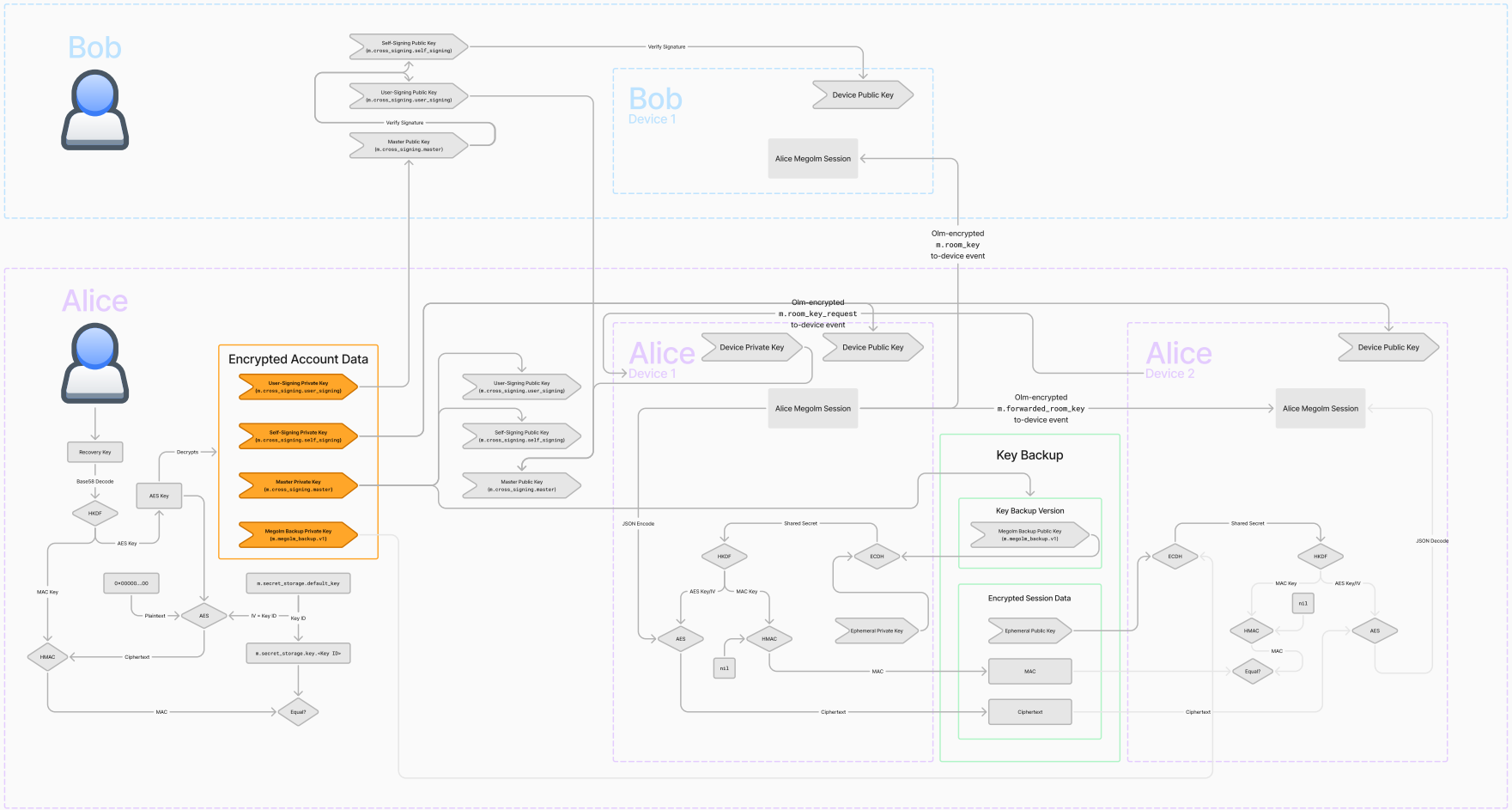


Secure Secret Storage and Sharing (SSSS)

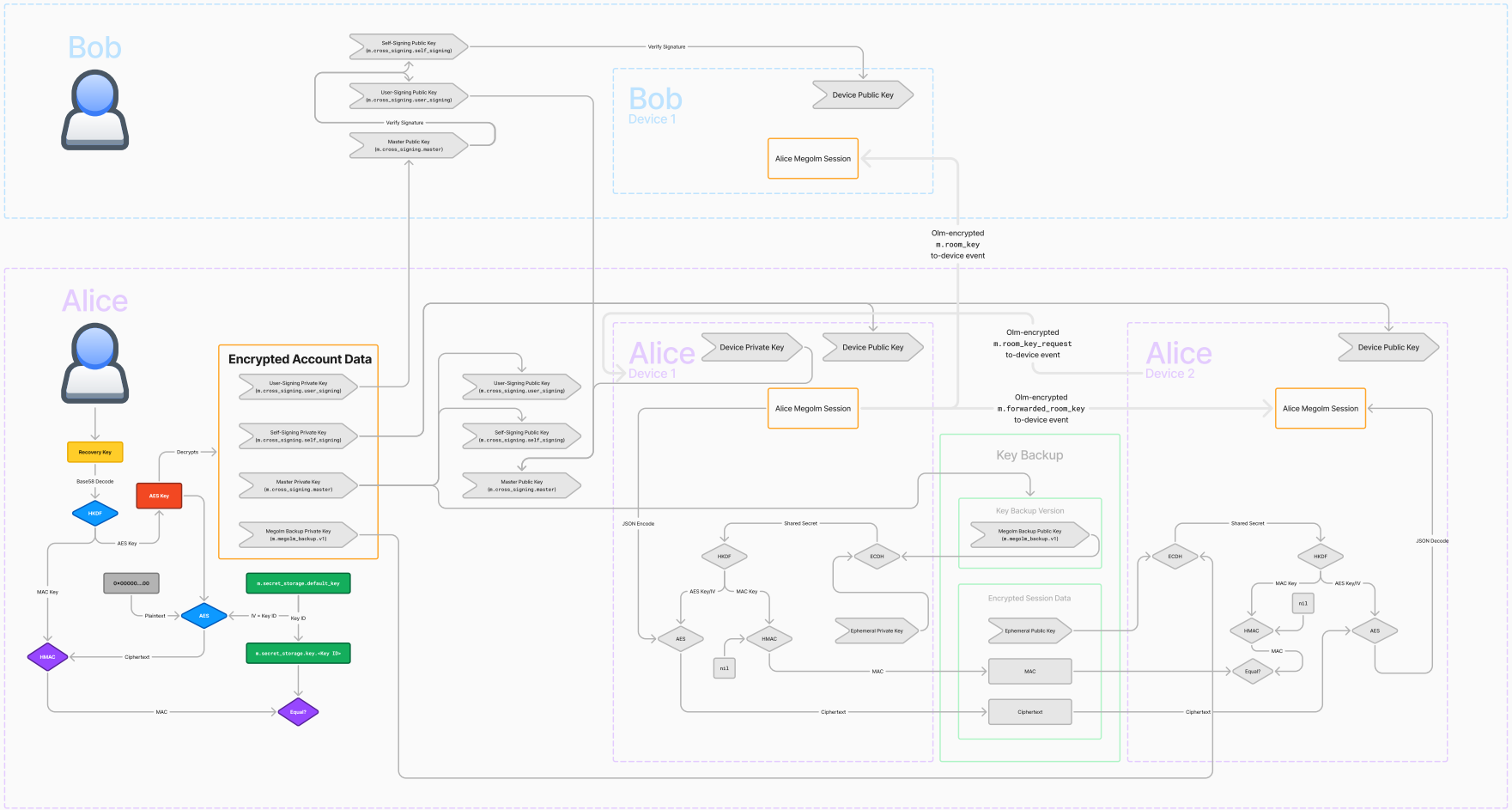
Don’t Forget Your Keys



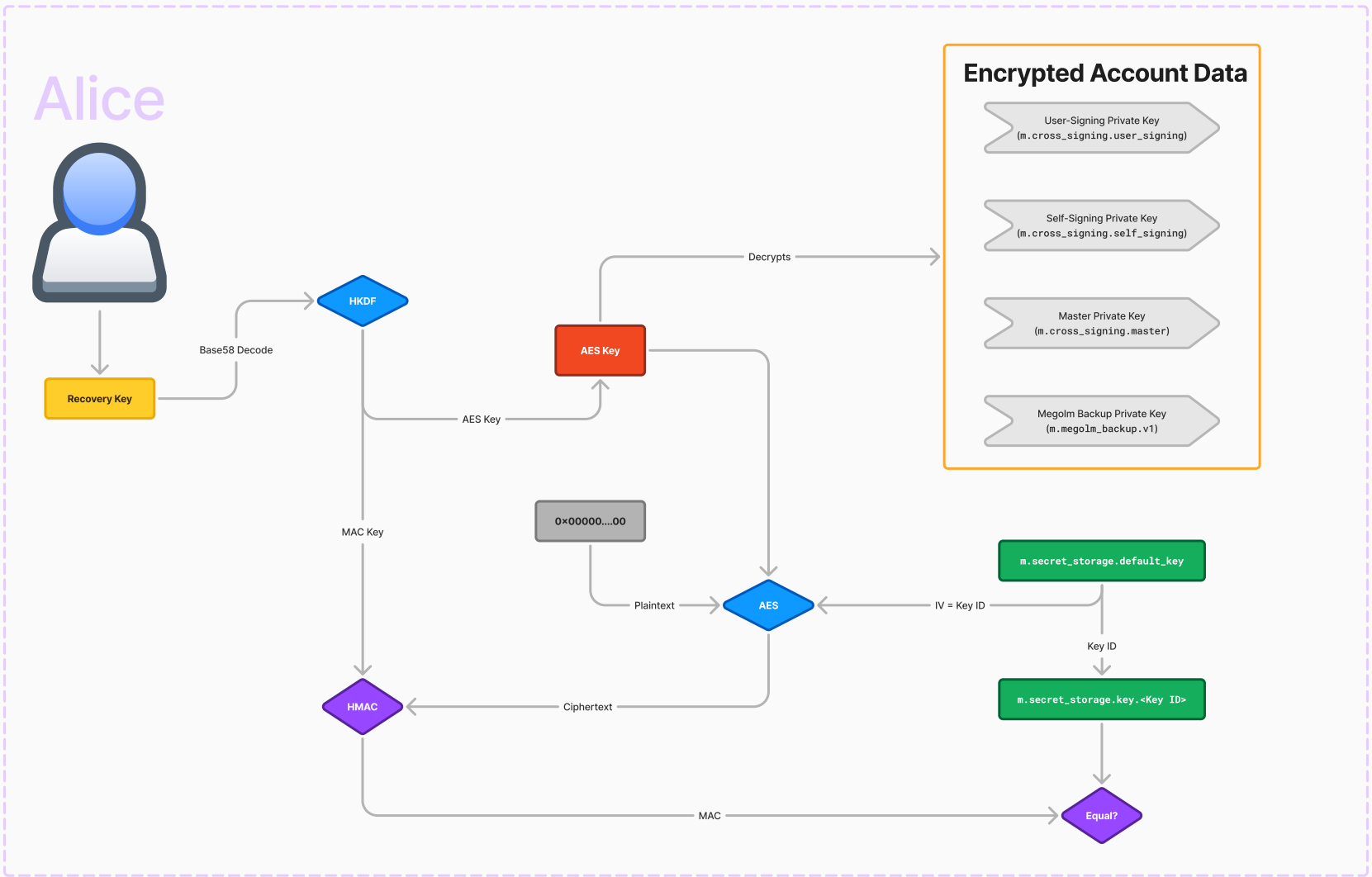
Don’t Forget Your Keys



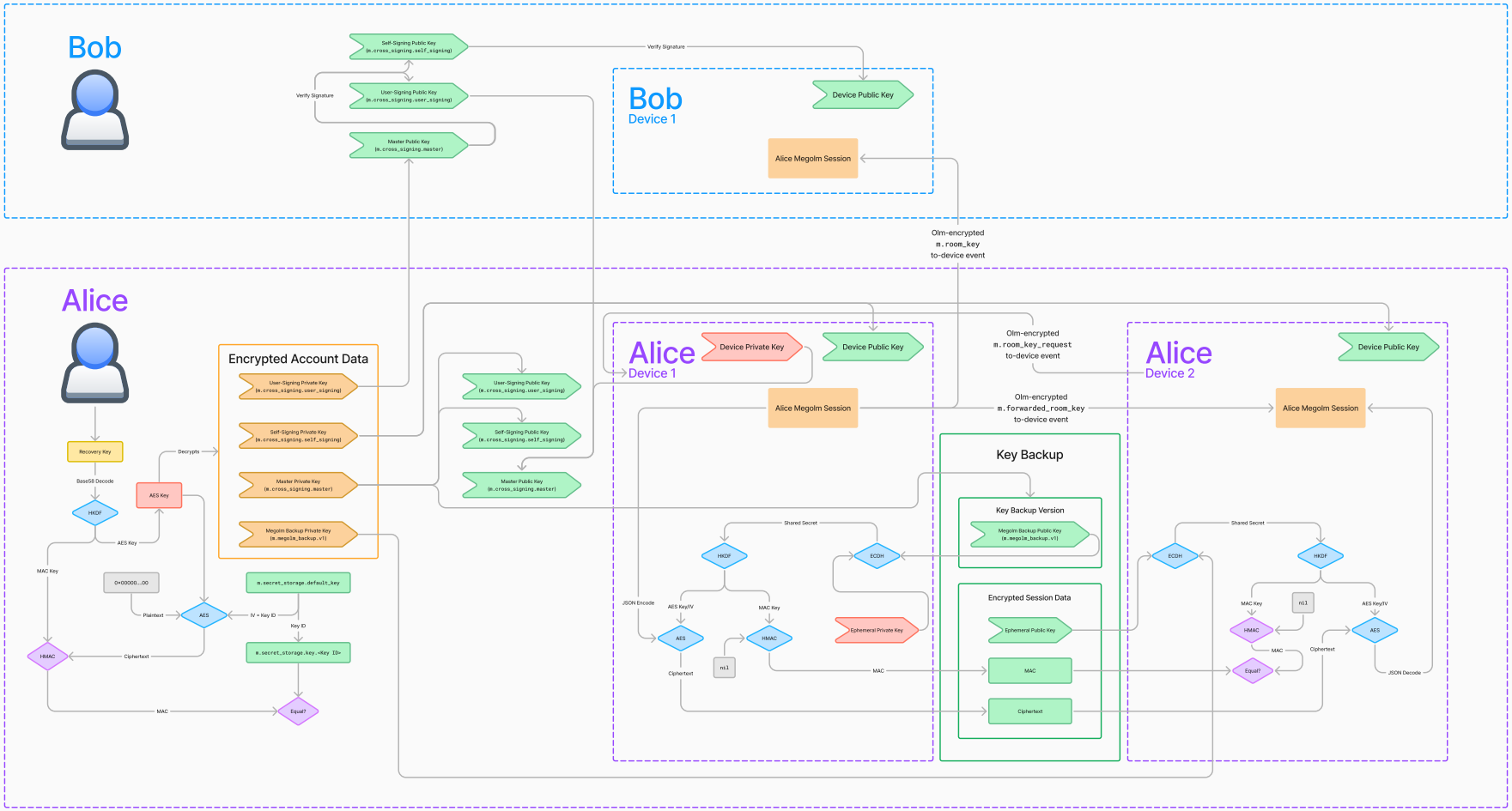
Don’t Forget Your Keys



Don’t Forget Your Keys



Big Picture


Bob
Self-Signing Public Key
(m.cross_signing.self_signing)
Verify Signature
User-Signing Public Key
(m.cross_signing.user_signing)
Verify Signature
Master Public Key
(m.cross_signing.master)
Bob
Device 1
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key
to-device event
Alice
Encrypted Account Data
User-Signing Private Key
(m.cross_signing.user_signing)
Self-Signing Private Key
(m.cross_signing.self_signing)
Master Private Key
(m.cross_signing.master)
Megolm Backup Private Key
(m.megolm_backup.v1)
Recovery Key
Decrypts
Base58 Decode
HKDF
AES Key
AES Key
MAC Key
0x00000...00
Plaintext
AES
IV = Key ID
Key ID
m.secret_storage.default_key
m.secret_storage.key.<Key ID>
HMAC
Ciphertext
MAC
Equal?
User-Signing Public Key
(m.cross_signing.user_signing)
Self-Signing Public Key
(m.cross_signing.self_signing)
Master Public Key
(m.cross_signing.master)
Alice
Device 1
Device Private Key
Device Public Key
Alice Megolm Session
Olm-encrypted
m.room_key_request
to-device event
Olm-encrypted
m.forwarded_room_key
to-device event
Alice
Device 2
Device Public Key
Alice Megolm Session
Key Backup
Key Backup Version
Megolm Backup Public Key
(m.megolm_backup.v1)
Encrypted Session Data
Ephemeral Public Key
MAC
Ciphertext
Shared Secret
HKDF
ECDH
JSON Encode
AES Key/IV
MAC Key
AES
HMAC
nil
Ciphertext
Ephemeral Private Key
MAC
Shared Secret
ECDH
HKDF
MAC Key
nil
AES Key/IV
HMAC
AES
MAC
Ciphertext
Equal?
JSON Decode

Thank You for Listening!

Questions?

sumnerevans.com/posts/matrix/cryptographic-key-infrastructure