

Package: mx.crypto (via r-universe)

June 13, 2026

Type Package

Title Matrix End-to-End Encryption Primitives

Version 0.2.0

Date 2026-05-13

Description 'Olm' and 'Megolm' encryption ratchet primitives for the 'Matrix' messaging protocol <<https://matrix.org/>>, wrapping the 'vodozamac' Rust crate. Provides device-key generation, one-time-key management, 1:1 'Olm' sessions, and 'Megolm' group sessions. Pairs with the 'mx.api' package, which handles 'Matrix' HTTP transport.

License MIT + file LICENSE | Apache License 2.0

URL <https://github.com/cornball-ai/mx.crypto>

BugReports <https://github.com/cornball-ai/mx.crypto/issues>

Depends R (>= 4.3)

SystemRequirements Cargo (Rust's package manager), rustc (>= 1.85)

Suggests tinytest, mx.api (>= 0.2.0), simplermarkdown

VignetteBuilder simplermarkdown

Encoding UTF-8

Config/pak/sysreqs libclang-dev

Repository <https://cornball-ai.r-universe.dev>

Date/Publication 2026-05-13 21:12:17 UTC

RemoteUrl <https://github.com/cornball-ai/mx.crypto>

RemoteRef HEAD

RemoteSha 50620be65a3057a197e5ca0afaf5093d0465c0ba

Contents

mx.crypto-package	2
mxc_account_fallback_key	3

mx_account_generate_one_time_keys	3
mx_account_identity_keys	4
mx_account_mark_published	4
mx_account_new	5
mx_account_one_time_keys	5
mx_account_pickle	6
mx_account_sign	6
mx_account_unpickle	7
mx_ed25519_verify	7
mx_megolm_decrypt	8
mx_megolm_encrypt	8
mx_megolm_inbound_new	9
mx_megolm_inbound_pickle	9
mx_megolm_inbound_unpickle	10
mx_megolm_outbound_info	10
mx_megolm_outbound_new	11
mx_megolm_outbound_pickle	11
mx_megolm_outbound_unpickle	12
mx_olm_create_inbound	12
mx_olm_create_outbound	13
mx_olm_decrypt	13
mx_olm_encrypt	14
mx_olm_session_pickle	14
mx_olm_session_unpickle	15
mx_verify_device_keys	15
mx_verify_one_time_key	16
Index	18

mx.crypto-package	<i>mx.crypto: Matrix End-to-End Encryption Primitives</i>
-------------------	---

Description

Olm and Megolm ratchet primitives for the Matrix protocol, wrapping the ‘vodozamac’ Rust crate. Pairs with ‘mx.api’, which handles HTTP transport. mx.crypto is crypto only: no network, no canonical-JSON, no ‘m.room_key_request’, no cross-signing or SAS in 0.1.0.

Author(s)

Maintainer: Troy Hernandez <troy@cornball.ai>

`mxc_account_fallback_key`*Generate and return a fallback key*

Description

Generates a fallback curve25519 key, which the homeserver hands out when an OTK pool is exhausted. Returns the freshly generated key. Calling it again rotates the previous fallback.

Usage

```
mxc_account_fallback_key(account)
```

Arguments

`account` An Account.

Value

Named list with 'key_id' and 'curve25519' (both base64).

`mxc_account_generate_one_time_keys`*Generate one-time keys*

Description

Adds 'n' curve25519 one-time keys to the Account's pool. Call [mxc_account_one_time_keys] to read them out for upload, then [mxc_account_mark_published] once the homeserver has accepted them.

Usage

```
mxc_account_generate_one_time_keys(account, n)
```

Arguments

`account` An Account.
`n` Number of OTKs to generate.

Value

Invisible NULL; mutates 'account' in place.

Examples

```
acct <- mxc_account_new()  
mxc_account_generate_one_time_keys(acct, 5L)
```

mxc_account_identity_keys

Public identity keys for an Account

Description

Returns the device's two public keys: curve25519 (Olm sender / identity key, used to start sessions) and ed25519 (signing / fingerprint key). Both are unpadded base64 strings, ready for '/keys/upload'.

Usage

```
mxc_account_identity_keys(account)
```

Arguments

account An Account from [mxc_account_new] or [mxc_account_unpickle].

Value

A named list with 'curve25519' and 'ed25519' strings.

Examples

```
k <- mxc_account_identity_keys(mxc_account_new())
k$curve25519
```

mxc_account_mark_published

Mark current one-time keys as published

Description

Call after the homeserver has accepted a '/keys/upload' containing the keys returned by [mxc_account_one_time_keys]. Future calls to that function will not re-include them.

Usage

```
mxc_account_mark_published(account)
```

Arguments

account An Account.

Value

Invisible NULL; mutates 'account'.

mxc_account_new	<i>Create a new Olm Account</i>
-----------------	---------------------------------

Description

Creates a fresh device identity. Holds long-lived curve25519 / ed25519 identity keys and a one-time-key pool. The returned object is an external pointer; persist it with [mxc_account_pickle].

Usage

```
mxc_account_new()
```

Value

An external pointer to a vodozamac Account.

Examples

```
acct <- mxc_account_new()
mxc_account_identity_keys(acct)
```

mxc_account_one_time_keys	<i>Read pending one-time keys</i>
---------------------------	-----------------------------------

Description

Returns OTKs that have been generated but not yet marked as published. Each value is a curve25519 public key; the caller should sign each with [mxc_account_sign] and upload as 'signed_curve25519:<key_id>'.

Usage

```
mxc_account_one_time_keys(account)
```

Arguments

account	An Account.
---------	-------------

Value

Named list mapping 'key_id' to 'curve25519_pub' (both unpadded base64 strings).

Examples

```
acct <- mxc_account_new()
mxc_account_generate_one_time_keys(acct, 5L)
otks <- mxc_account_one_time_keys(acct)
```

mxc_account_pickle *Pickle an Account to an encrypted blob*

Description

Serialises the Account's state and encrypts it under a 32-byte key. Restore with [mxc_account_unpickle]. Pickle after every state change that you want to survive a restart (OTK generation, fallback key rotation, mark-published).

Usage

```
mxc_account_pickle(account, key)
```

Arguments

account	An Account.
key	A 'raw' vector of length 32. The caller is responsible for key derivation; use a KDF on a passphrase, do not pass a passphrase.

Value

Base64 string containing the encrypted pickle.

mxc_account_sign *Sign canonical JSON with the Account's ed25519 key*

Description

The caller must canonicalise the JSON per the Matrix spec (UTF-8, sorted keys, no whitespace, no insignificant data) before passing it in. mx.crypto does not canonicalise; mx.api will.

Usage

```
mxc_account_sign(account, canonical_json)
```

Arguments

account	An Account.
canonical_json	A character string containing canonical JSON.

Value

Unpadded base64 ed25519 signature.

Examples

```
acct <- mxc_account_new()
sig <- mxc_account_sign(acct, '{"hello":"world"}')
```

mxc_account_unpickle *Restore an Account from an encrypted pickle*

Description

Restore an Account from an encrypted pickle

Usage

```
mxc_account_unpickle(blob, key)
```

Arguments

blob	A base64 string produced by [mxc_account_pickle].
key	The 32-byte key the pickle was encrypted under.

Value

An Account external pointer.

mxc_ed25519_verify *Verify an Ed25519 signature*

Description

Thin wrapper over vodozamac's strict Ed25519 verifier (`verify_strict`). Returns TRUE when the signature is valid and FALSE when it isn't; raises an error if any of `public_key`, `message`, or `signature` is malformed (bad base64, wrong length, etc.).

Usage

```
mxc_ed25519_verify(public_key, message, signature)
```

Arguments

public_key	Character. Unpadded base64 ed25519 public key (the same shape <code>mxc_account_identity_keys</code> returns).
message	A raw vector. The byte sequence the signer ran ed25519 over; typically the output of <code>mx.api::mx_canonical_json()</code> .
signature	Character. Unpadded base64 ed25519 signature (86 chars).

Value

Single logical: TRUE for a valid signature, FALSE otherwise.

Examples

```
acct <- mxc_account_new()
ids <- mxc_account_identity_keys(acct)
sig <- mxc_account_sign(acct, "{\"hello\":\"world\"}")
mxc_ed25519_verify(ids$ed25519, charToRaw("{\"hello\":\"world\"}"), sig)
```

mxc_megolm_decrypt *Decrypt a room message*

Description

Decrypt a room message

Usage

```
mxc_megolm_decrypt(igs, ciphertext_b64)
```

Arguments

`igs` An InboundGroupSession.
`ciphertext_b64` Base64 ciphertext (the ‘ciphertext’ field of the ‘m.room.encrypted’ event).

Value

Named list: ‘plaintext’ (raw) and ‘message_index’ (integer). Use ‘message_index’ to dedupe replays.

mxc_megolm_encrypt *Encrypt a room message*

Description

Encrypt a room message

Usage

```
mxc_megolm_encrypt(gs, plaintext)
```

Arguments

`gs` An outbound GroupSession.
`plaintext` A ‘raw’ vector. The caller is responsible for producing the canonical JSON event body.

Value

Base64 ciphertext (the ‘ciphertext’ field of ‘m.room.encrypted’).

`mxc_megolm_inbound_new`*Build an inbound Megolm session from a shared session_key*

Description

Constructs the receiver-side ratchet using the 'session_key' that was delivered (over Olm) in an 'm.room_key' event. The receiver stores the resulting object indexed by '(sender_curve25519, session_id)' and re-uses it to decrypt incoming messages on that session.

Usage

```
mxc_megolm_inbound_new(session_key)
```

Arguments

session_key Base64 session_key from 'm.room_key'.

Value

An InboundGroupSession external pointer.

`mxc_megolm_inbound_pickle`*Pickle an inbound group session*

Description

Pickle after every decrypt so the ratchet state survives restart.

Usage

```
mxc_megolm_inbound_pickle(igs, key)
```

Arguments

igs An InboundGroupSession.
key 32-byte raw vector.

Value

Base64 string.

mxc_megolm_inbound_unpickle

Restore an inbound group session from a pickle

Description

Restore an inbound group session from a pickle

Usage

```
mxc_megolm_inbound_unpickle(blob, key)
```

Arguments

blob	Base64 string.
key	32-byte raw vector.

Value

An InboundGroupSession external pointer.

mxc_megolm_outbound_info

Inspect an outbound group session

Description

Returns the 'session_id', current 'session_key' (the value to ship via 'm.room_key' to recipient devices), and 'message_index'.

Usage

```
mxc_megolm_outbound_info(gs)
```

Arguments

gs	An outbound GroupSession.
----	---------------------------

Value

Named list: 'session_id' (str), 'session_key' (str), 'message_index' (int).

`mxc_megolm_outbound_new`*Create an outbound Megolm group session*

Description

The sender side of a Megolm ratchet. Use [mxc_megolm_outbound_info] to read out the 'session_key' that must be shared (over Olm) with each recipient device, and [mxc_megolm_encrypt] to encrypt room messages. Rotate (create a new one) on membership changes or after your chosen number-of-messages / time threshold.

Usage`mxc_megolm_outbound_new()`**Value**

An outbound GroupSession external pointer.

`mxc_megolm_outbound_pickle`*Pickle an outbound group session*

Description

Pickle an outbound group session

Usage`mxc_megolm_outbound_pickle(gs, key)`**Arguments**

<code>gs</code>	An outbound GroupSession.
<code>key</code>	32-byte raw vector.

Value

Base64 string.

mxc_megolm_outbound_unpickle

Restore an outbound group session from a pickle

Description

Restore an outbound group session from a pickle

Usage

```
mxc_megolm_outbound_unpickle(blob, key)
```

Arguments

blob	Base64 string.
key	32-byte raw vector.

Value

An outbound GroupSession external pointer.

mxc_olm_create_inbound

Build an inbound Olm session from a pre-key message

Description

Consumes the matching one-time key from the local Account. The result has the new Session and the decrypted plaintext of the pre-key message; subsequent messages on this session use [mxc_olm_decrypt].

Usage

```
mxc_olm_create_inbound(account, peer_curve25519, prekey_b64)
```

Arguments

account	The local Account (will be mutated: an OTK is consumed).
peer_curve25519	Sender's curve25519 identity key (base64).
prekey_b64	Body of the pre-key message (base64).

Value

Named list: 'session' (external pointer) and 'plaintext' (raw).

mxc_olm_create_outbound
Start an outbound Olm session

Description

Use a peer's published curve25519 identity key + one of their signed one-time keys to bootstrap a 1:1 ratchet. The first ciphertext from this session is a pre-key message ('type = 0').

Usage

```
mxc_olm_create_outbound(account, peer_curve25519, peer_otk)
```

Arguments

account	The local Account.
peer_curve25519	Peer's curve25519 identity key (base64).
peer_otk	Peer's one-time key (base64).

Value

An Olm Session external pointer.

mxc_olm_decrypt *Decrypt a message on an Olm session*

Description

Decrypt a message on an Olm session

Usage

```
mxc_olm_decrypt(session, type, body)
```

Arguments

session	An Olm Session.
type	Message type: '0L' for pre-key, '1L' for normal.
body	Ciphertext (base64).

Value

A 'raw' vector of plaintext bytes.

mxc_olm_encrypt *Encrypt a message on an Olm session*

Description

Encrypt a message on an Olm session

Usage

```
mxc_olm_encrypt(session, plaintext)
```

Arguments

session	An Olm Session.
plaintext	A 'raw' vector.

Value

Named list: 'type' ('0L' pre-key, '1L' normal) and 'body' (base64).

mxc_olm_session_pickle *Pickle an Olm session*

Description

Pickle after every encrypt/decrypt to keep the ratchet state on disk.

Usage

```
mxc_olm_session_pickle(session, key)
```

Arguments

session	An Olm Session.
key	32-byte raw vector.

Value

Base64 string.

`mxc_olm_session_unpickle`*Restore an Olm session from a pickle*

Description

Restore an Olm session from a pickle

Usage

```
mxc_olm_session_unpickle(blob, key)
```

Arguments

<code>blob</code>	Base64 string produced by [mxc_olm_session_pickle].
<code>key</code>	32-byte raw vector.

Value

An Olm Session external pointer.

`mxc_verify_device_keys`*Verify a Matrix device-keys object*

Description

Validates a single device_keys object as returned by `/_matrix/client/v3/keys/query`: checks that the structural fields (user_id, device_id, algorithms, keys, signatures) are present and match the expected identity, then verifies the ed25519 self-signature over the canonical-JSON byte sequence (with signatures and unsigned stripped). Any of the following raises an error rather than returning FALSE: missing field, wrong user_id or device_id, missing curve25519 / ed25519 key, missing signatures block, signature under the wrong user or device, invalid signature, malformed base64. This function deliberately accepts *any* ed25519 key the object claims for itself; it does not pin against a previously trusted key. Identity pinning is the caller's responsibility (typically TOFU + cross-signing).

Usage

```
mxc_verify_device_keys(device_keys, expected_user_id, expected_device_id,  
                        required_algorithms = NULL)
```


Arguments

algorithm_key_id	Character. The outer map key from the one_time_keys or fallback_keys response, e.g. "signed_curve25519:AAAAAAAAAAAA". The algorithm prefix must be signed_curve25519.
key_object	Named list. The signed key object (must contain key and signatures).
signing_ed25519	Character. The base64 ed25519 public key that should have signed this OTK (i.e. the ed25519 returned by mxc_verify_device_keys for the same device).
expected_user_id	Character. Matrix user id the OTK is supposed to come from.
expected_device_id	Character. Matrix device id.

Value

Character. The verified curve25519 OTK (base64), ready to feed into mxc_olm_create_outbound.

Examples

```
## Not run:
cl <- mx.api::mx_keys_claim(s, list(
  "@alice:server" = list("ALICEDEV" = "signed_curve25519")
))
entry <- cl$one_time_keys[["@alice:server"]][["ALICEDEV"]]
algo_kid <- names(entry)[[1]] # e.g. "signed_curve25519:AAAA"
otk <- mxc_verify_one_time_key(
  algo_kid, entry[[1]], alice_ed, "@alice:server", "ALICEDEV"
)

## End(Not run)
```

Index

`mx.crypto` (`mx.crypto-package`), 2
`mx.crypto-package`, 2
`mx_account_fallback_key`, 3
`mx_account_generate_one_time_keys`, 3
`mx_account_identity_keys`, 4
`mx_account_mark_published`, 4
`mx_account_new`, 5
`mx_account_one_time_keys`, 5
`mx_account_pickle`, 6
`mx_account_sign`, 6
`mx_account_unpickle`, 7
`mx_ed25519_verify`, 7
`mx_megolm_decrypt`, 8
`mx_megolm_encrypt`, 8
`mx_megolm_inbound_new`, 9
`mx_megolm_inbound_pickle`, 9
`mx_megolm_inbound_unpickle`, 10
`mx_megolm_outbound_info`, 10
`mx_megolm_outbound_new`, 11
`mx_megolm_outbound_pickle`, 11
`mx_megolm_outbound_unpickle`, 12
`mx_olm_create_inbound`, 12
`mx_olm_create_outbound`, 13
`mx_olm_decrypt`, 13
`mx_olm_encrypt`, 14
`mx_olm_session_pickle`, 14
`mx_olm_session_unpickle`, 15
`mx_verify_device_keys`, 15
`mx_verify_one_time_key`, 16