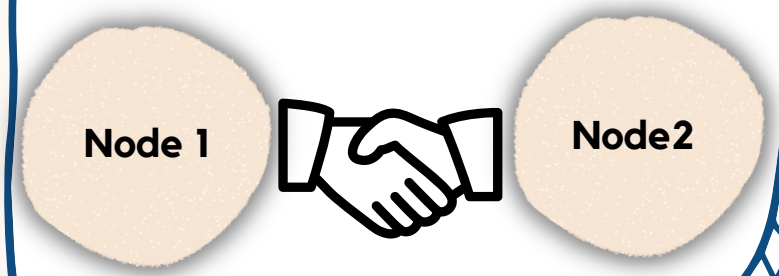
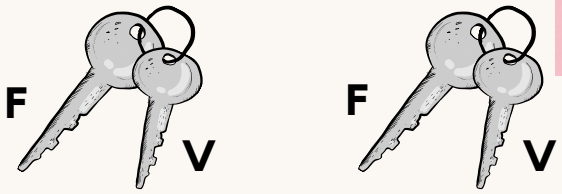
How to do v2 P2P protocol?

the high level picture

Initial v2 handshake





1 node has 2 key pair = 4 keys

1 key pair = F and V

- 1 key pair for sending
- 1 key pair for receiving

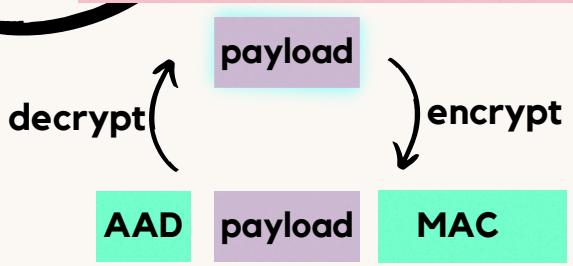
keys

1 node has 2 instances of

ChaCha20 Poly1305 AEAD

each instance can generate 1 keystream.

generates keystream which can encrypt/decrypt msg



2 keystreams made:

- 1 for sending
- 1 for receiving

Build/Receive v2 P2P messages!

Initial v2 handshake

CRYPTOGRAPHY YOU'LL RUN ACROSS:

1. ELLIGATOR SQUARED ENCODING (ELLSQ)
   - Makes curve points (and so pubkeys) indistinguishable from random to an attacker
2. ECDH
   - To establish a shared secret between the nodes
3. HKDF
   - To do key extraction from shared secret

ellsq encoding of X

(Node 2 can ellsq decode to find X)

ECDH = ECDH(y,X)

derive 4 keys + session id = HKDF

Node 2

ellsq encoding of Y +
encrypted(Node 2's transport_version)

Node 1

ECDH(x ,Y) = **ECDH**

HKDF = **derive 4 keys + session id**

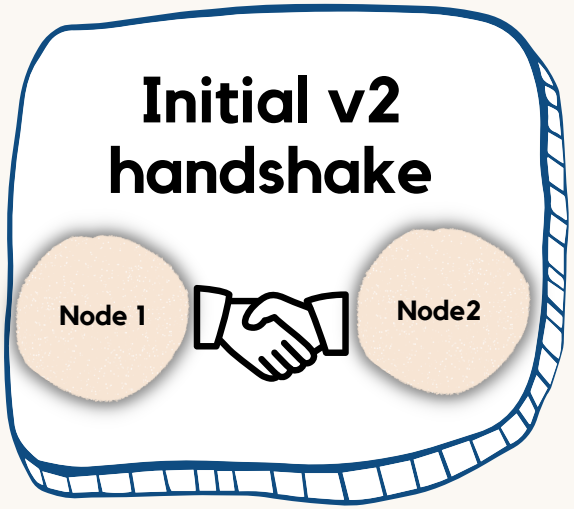
encrypted(Node 1's transport_version)

Initiator
privkey=x
pubkey=X

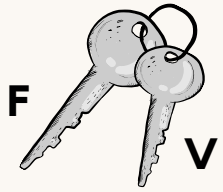
Responder
privkey=y
pubkey=Y

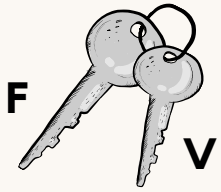
The minimum transport version will be set by each node as it's transport version (for future upgradability)

Initial v2 handshake



When this is over, each node ends up with





session id

keys for sending

keys for receiving

make

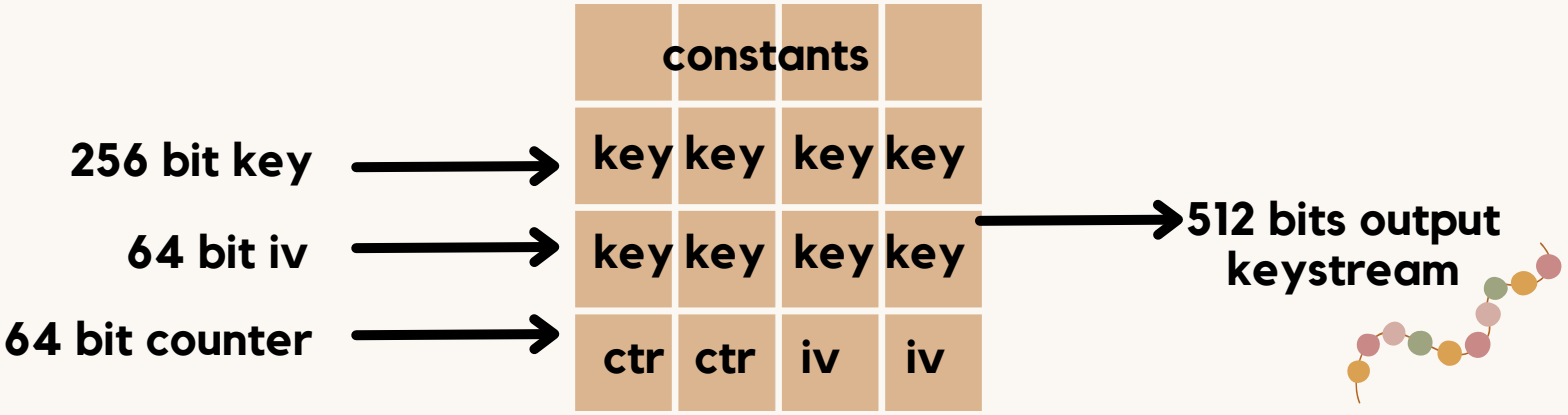
make

ChaCha20Poly1305 AEAD

ChaCha20Poly1305 AEAD

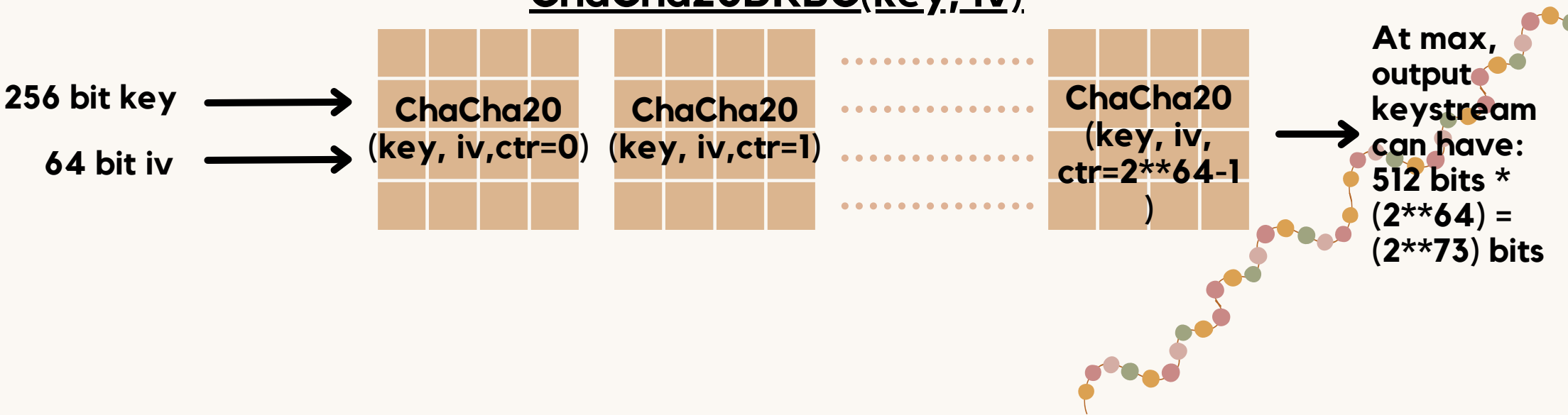
ChaCha20(key, iv, ctr)

= 1 word = 4 bytes = 32 bits

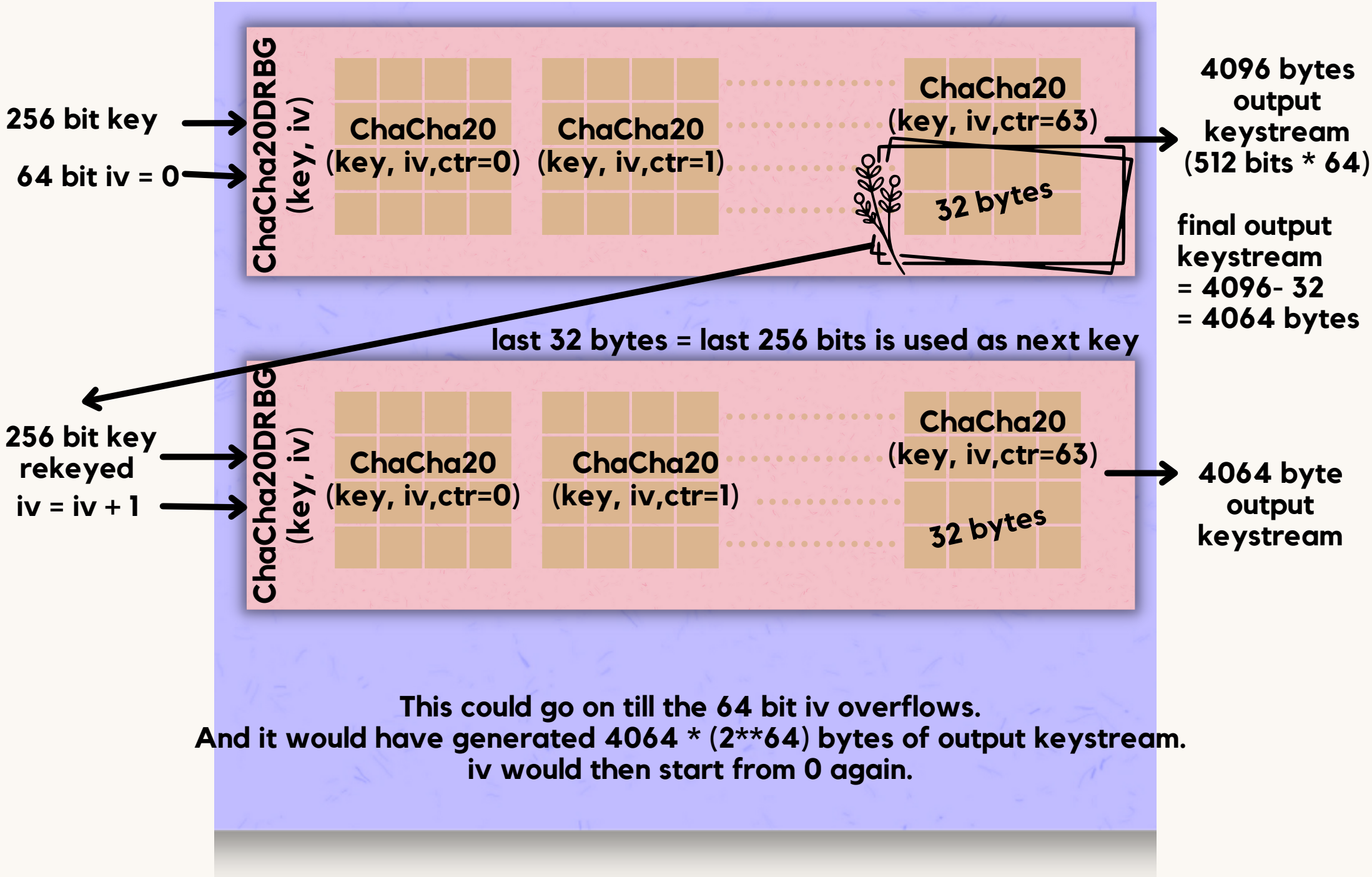


- counter consists of 64 bits and at can be only ever less than 2**64
- ChaCha20DRBG computes the output keystream for a key and iv by making ChaCha20 and supplying all possible counters [0,2**64) one at a time.

ChaCha20DRBG(key, iv)

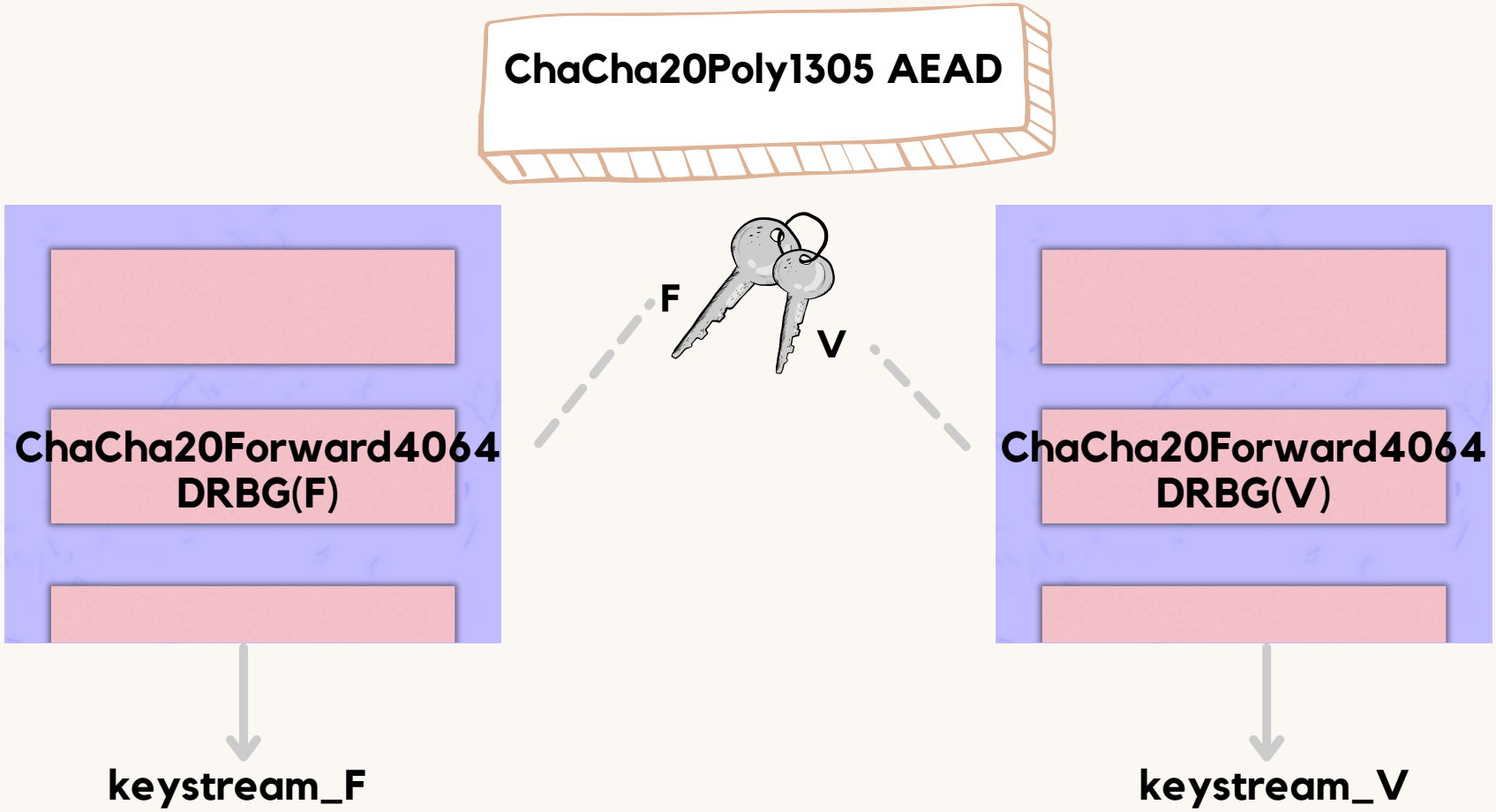


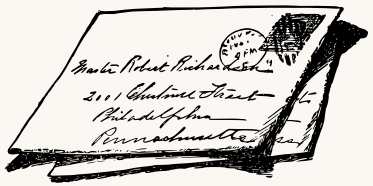
ChaCha20Forward4064DRBG(key)



This could go on till the 64 bit iv overflows.
And it would have generated 4064 * (2**64) bytes of output keystream.
iv would then start from 0 again.

Note: we can keep generating keystream_F and keystream_V from ChaCha20Forward4064DRBG instances when they get exhausted



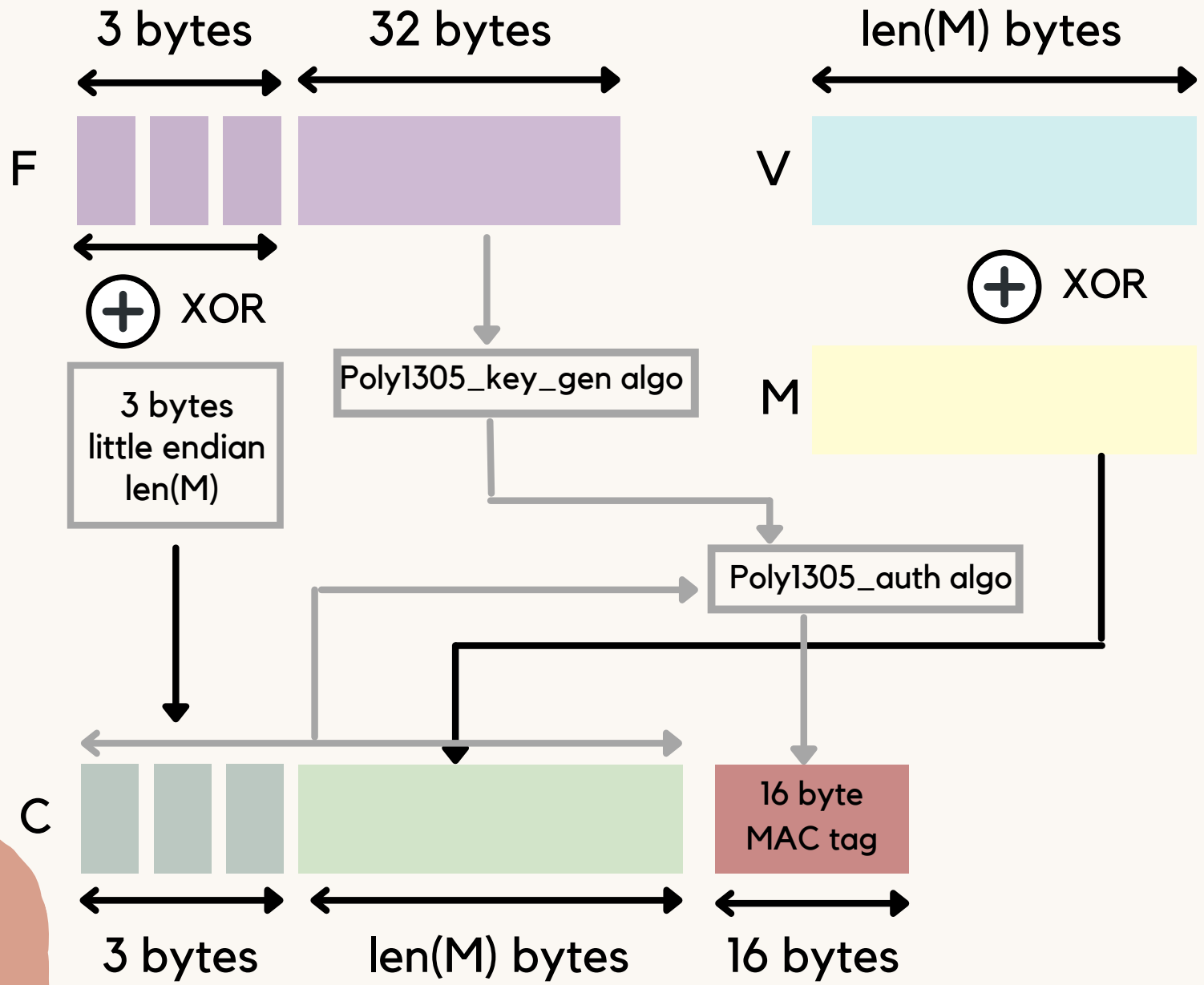


To encrypt/decrypt 1 message, we need:

- *35 bytes from keystream_F*
  - *3 bytes for AAD*
  - *32 bytes for Poly1305 key*
- *len(message) bytes from keystream_V*

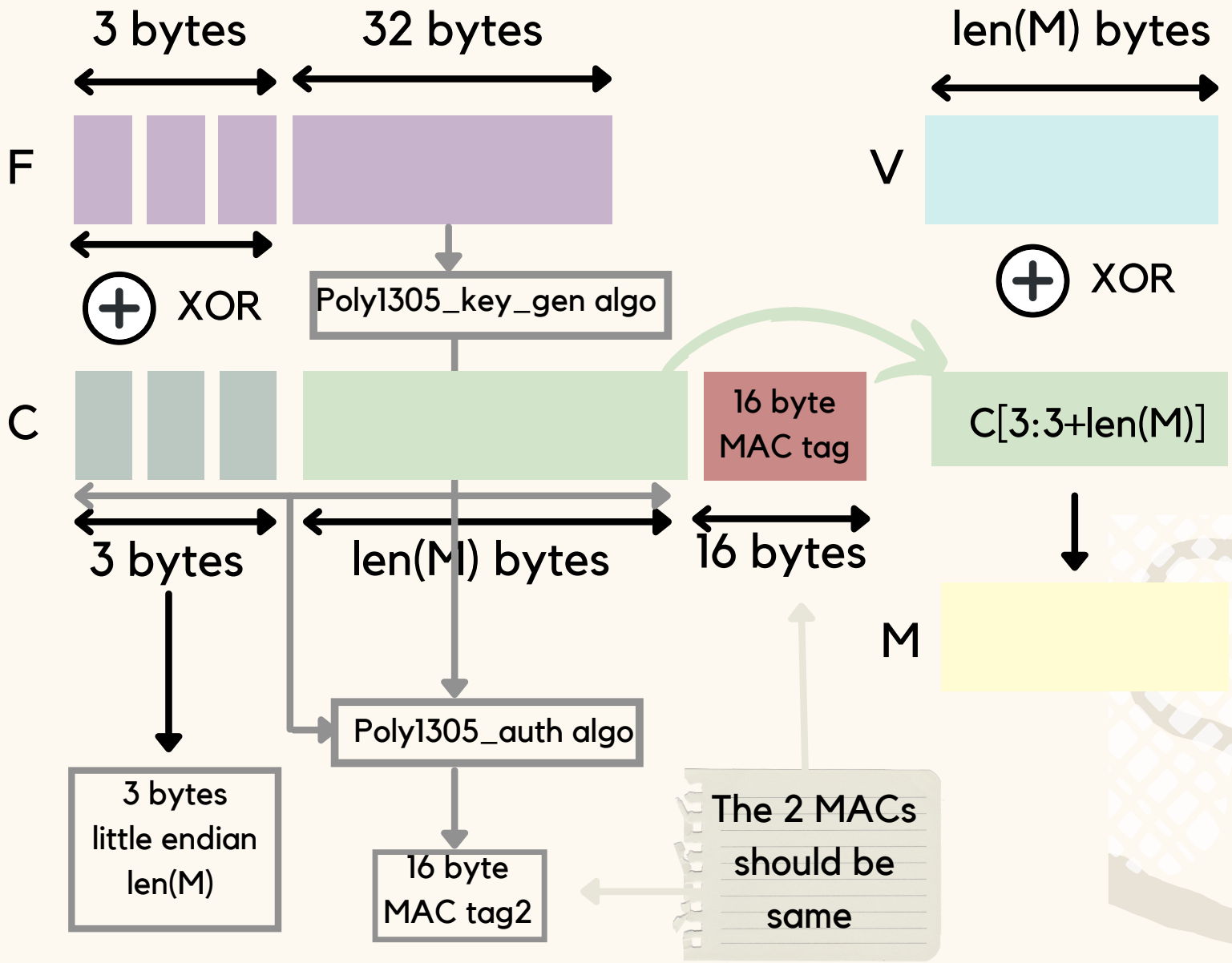
ChaCha20Poly1305 AEAD: Encryption

of message M to produce ciphertext C
using 2 ChaCha20Forward4064DRBG instances F and V



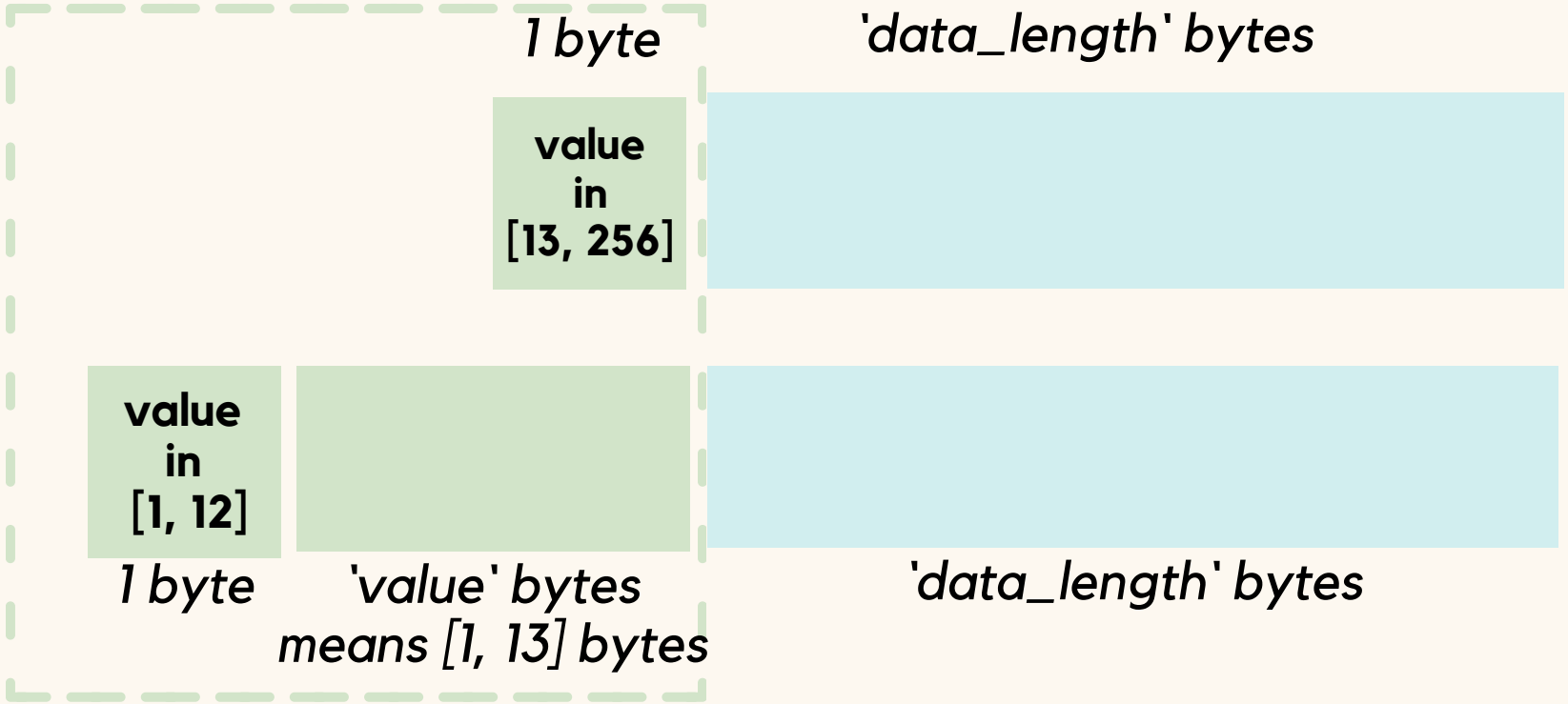
ChaCha20Poly1305 AEAD: Decryption

of ciphertext C and MAC to obtain message M
using 2 ChaCha20Forward4064DRBG instances F and V



Build/Receive v2 P2P messages!

Building the payload



payload = message_type + data

Building the v2 P2P message

Receiving v2 P2P message