Brushpass

Security review. Message format bp1. BOT Chain Mainnet, chain ID 677.

Review date	August 09, 2026
Contract	See deployments/botchain.json
Network	BOT Chain Mainnet, chain ID 677
Compiler	Solidity 0.8.24
Reviewer	@ayaanoncrypto, project author
Independence	None. This is a self review by the author, not a third party audit.
Scope	contracts/Brushpass.sol and the client encryption layer in frontend/src/utils/

1. Summary

Brushpass is end-to-end encrypted messaging between wallets, stored entirely on chain. The client uses X25519 key agreement with AES-256-GCM content encryption and per-recipient key wrapping. Public keys are published on chain through the message contract itself; private keys are derived from a wallet signature and never leave the browser.

This review covers the contract and the client encryption layer prior to public launch. No critical or high severity issues remain open. Two findings are open by design and one is an architectural limit that would require a redeployment to remove; all three are documented below and stated in the product interface rather than hidden.

Read this as an author self review. It is not a substitute for an independent audit, and readers should weight it accordingly.

2. Findings

Severity	Count	Status
Critical	0	None open
High	0	None open
Medium	1	Open, needs redeployment
Low	2	Open, inherent or by design
Informational	2	No action
Resolved in development	2	Fixed before launch

BP-M01	**MEDIUM**	**No pagination on getInbox and getSent**	Open
Detail	Both functions iterate the caller's full message id array twice and return every message in one response. Cost grows without bound. A wallet with a large inbox, or one deliberately spammed with messages by an attacker, will eventually exceed node eth_call gas and response limits, at which point that inbox can no longer be read through these functions.		

Resolution	Not fixable in deployed bytecode, since the contract has no admin and no proxy. A future deployment should add offset and limit parameters plus a count getter. The client mitigates partially by filtering key announcements after retrieval, which reduces render cost but not call cost.

BP-L01	LOW	**Metadata is fully public**	Open, inherent
Detail	Sender, recipient, timestamp, message id, and ciphertext length are stored unencrypted and are readable by anyone. The social graph of who messages whom, when, and how often is exposed even though content is not.		
Resolution	Inherent to storing messages on a public ledger. Not fixed and not fixable at this layer. The landing screen states this directly rather than implying full privacy.		

BP-L02	LOW	**Deletion hides, it does not erase**	Open, by design
Detail	deleteMessage sets a boolean flag that filters the message out of getInbox and getSent. The ciphertext remains in contract storage, remains retrievable through getMessage, and remains in the transaction history of every archive node permanently.		
Resolution	Accepted. The delete control is a mailbox filter, not erasure. Sender retraction is also unsupported: only the recipient can delete.		

BP-I01	INFO	**No admin, owner, or upgrade path**	No action
Detail	The contract has no privileged role, no owner, no pause, and no proxy. Nobody can seize, censor, or alter messages, including the author.		
Resolution	Positive finding. This also means the BP-M01 pagination limit cannot be patched in place.		

BP-I02	INFO	**Custom errors and 0.8.24 arithmetic**	No action
Detail	Custom errors replace revert strings, reducing deployment and runtime gas. Solidity 0.8.24 provides checked arithmetic, so no SafeMath is required. The only unbounded values are the message id counter and the per-address id arrays.		
Resolution	Positive finding.		

3. Resolved during development

Two issues were found and fixed before any public release. They are recorded here because the retired message prefixes are still recognised by the client, and because the first is a mistake common enough in on-chain messaging that it is worth naming.

BP-D01	**CRITICAL**	**Encryption key derivable from public data**	Fixed pre-launch
Detail	An early development scheme, message prefix enc:v2, derived the AES key as SHA-256 of the two wallet addresses. Both addresses are stored in the contract as plaintext struct fields and emitted in MessageSent as indexed topics. Every input to the key derivation was therefore public. Any observer could recompute the key for any message pair and decrypt offline, with no wallet access and no interaction. The scheme was obfuscation, not encryption.		
Resolution	Replaced with X25519 ECDH before launch. A random 256-bit content key encrypts the message body under AES-GCM, and that content key is wrapped twice, once to the recipient public key and once to the sender public key, both against a fresh ephemeral keypair. Only holders of the corresponding private keys can unwrap. Any enc:v2 payload encountered is refused and labelled LEGACY.		

BP-D02	**MEDIUM**	**Ciphertext replay across sender identities**	Fixed pre-launch
Detail	Contract storage binds a message to its sender, but the early ciphertext format carried no binding of its own. A third party could copy a ciphertext observed on chain and resend it from their own address, and the recipient interface would render it as a message authored by the replayer.		
Resolution	Both the wrapped keys and the message body now use GCM additional authenticated data set to the sender and recipient address pair. A ciphertext replayed from any other address fails authentication and is rejected rather than displayed.		

4. Encryption architecture

4.1 Identity

The user signs one fixed statement with their wallet. The client hashes that signature with SHA-256 and uses the digest as an X25519 private key. Standard EOA wallets produce deterministic ECDSA signatures, so the same wallet reproduces the same keypair on every device and every session with no stored secret. The signature is never transmitted and the statement authorizes no transfer. Smart contract wallets that produce non-deterministic signatures are not supported.

4.2 Public key distribution

Public keys are published through the message contract as a plaintext self-message with the prefix bpkey:v1: followed by 64 hex characters. Because the contract sets sender from msg.sender, an entry where sender equals recipient equals A can only have been written by A. No third party can publish or overwrite a key on another address's behalf. Lookup takes the highest message id, so rotation is just publishing a newer entry. No separate registry contract is required.

Consequence: an address must publish a key before it can receive mail. Sending to an unregistered address is blocked in the interface rather than silently downgraded.

4.3 Message format

```
bp1: followed by base64 of: ephemeral public key, 32 bytes | IV, 12 | recipient key wrap, 48 | IV, 12 | sender key wrap, 48 | IV, 12 | AES-256-GCM body.
```

Key wrapping keys are HKDF-SHA256 over the ECDH shared secret, salted with the ephemeral public key. Fixed overhead is 164 bytes before base64, which leaves 2889 bytes of plain text inside the contract's 4096 byte cap. The composer meters the encrypted size, not the typed size.

4.4 What Brushpass does not protect

• Metadata. See BP-L01.

• Compromised wallets. Anyone who can sign with the wallet can rederive the private key and read the full history.

• Forward secrecy. The identity key is long lived, so a leaked key exposes all past messages.

• Malicious frontends. Encryption runs in the browser, so a compromised deployment could exfiltrate plaintext. Users who need certainty should build from source.

• Recipient behaviour. A recipient can always screenshot or repost what they receive.

5. Verification

The security claims above are covered by an automated test suite at frontend/test/crypto.test.mjs, run with npm test. It asserts that a third party holding both addresses and the full ciphertext cannot recover plaintext, that replay from a different sender address fails, that tampering is rejected, and that retired payload formats are refused rather than trusted. Contract behaviour is covered by npx hardhat test.