

# EFPIX: A zero-trust encrypted flood protocol

Arin Upadhyay  
arinupadhyay.cs@gmail.com

July 1, 2025

## Abstract

We propose a flood-based relay communication protocol that achieves end-to-end encryption, plausible deniability for users, and untraceable messages. It is resistant to changes in topology and infrastructure failures. It is also designed to hide metadata, such as sender and receiver, from those not involved.

## 1 Introduction

In recent years, government and invasive intelligence agencies have transformed the Internet to erode many privacy rights using various surveillance programs, laws, and backdoors. [1] [2] They are not hesitant to restrict or ban encryption. [3] They have the power to subpoena any data, track network traffic, and force shutdown of services that counter these issues. This has made journalism, whistle-blowing, and activism extremely dangerous. Any internet user who desires some privacy rights is powerless to protect their data from intrusive authorities. It is difficult to securely communicate over the Internet even if one's own devices are not compromised.

An attractive solution to this problem is an encrypted flood protocol. Such protocols provide secure peer-to-peer communication while also being topology-independent and resilient to infrastructure failures. They can be used with radio communication to achieve wireless node-to-node fault-free networking. This has many possible applications, including space, research, and military fields, where the existence and maintenance of a central server is not feasible. They are also suitable for broadcast-type messages that reach all nodes in the network, such as emergency rescue calls, disaster warnings, news distribution, etc.

EFPIX (Encrypted Flood Protocol for Information eXchange) is a protocol that is designed with the above issues in mind. It is resilient to government surveillance, authoritarian takedown, infrastructure loss, and metadata leaks. It is ideal for peer-to-peer privacy-focused applications where a central network is unstable or infeasible. It uses public-key encryption and signatures [4] along with a hashing function [5]. The protocol is designed to prevent the identification of the source or destination of the message while also providing encryption. It is possible to build higher-layer protocols onto this protocol to serve various use cases. However, it does not protect against endpoint or full topological compromise.

## 2 Overview

The desired message is encrypted and encoded before being relayed to all available nodes. If the message is already seen by a relaying node, it is not processed or further relayed. Otherwise, it relays the message to all neighbors. Then it attempts decryption with its private key. A **known** connection exists when the sender and receiver know each other's public key and **alias**. **Alias** is a string that serves as a key/reference for a public key. The sender includes their alias in messages, allowing the receiver to retrieve the sender's public key for signature verification. If a message contains an unknown alias, it is treated as **anonymous**, and the signature is not verified. Public keys and aliases are exchanged securely out-of-band or via a trusted overlay.

### 3 Encoding

Before encryption, the desired message is concatenated with

**Timestamp:** A compactly encoded representation of the message’s creation date and time.

**Sender Alias:** It serves as the key/reference to look up the sender’s public key to the receiver in its memory. It is agreed upon before the start of the communication. It can be different for different receivers. It is retrieved from the local (other alias) to (public key, my alias) map.

**Internal Address:** It can be used in different ways; for instance, to identify sub-clients in the case of a server that acts as a proxy for multiple clients or in the case of multiple processes that can process the decoded message.

This concatenated data is used to generate a signature using the sender’s private key. The concatenated data is encrypted using the receiver’s public key which is also retrieved from the local (other alias) to (public key, my alias) map. Finally, the hash is computed on the encrypted blob, the signature, and a nonce.

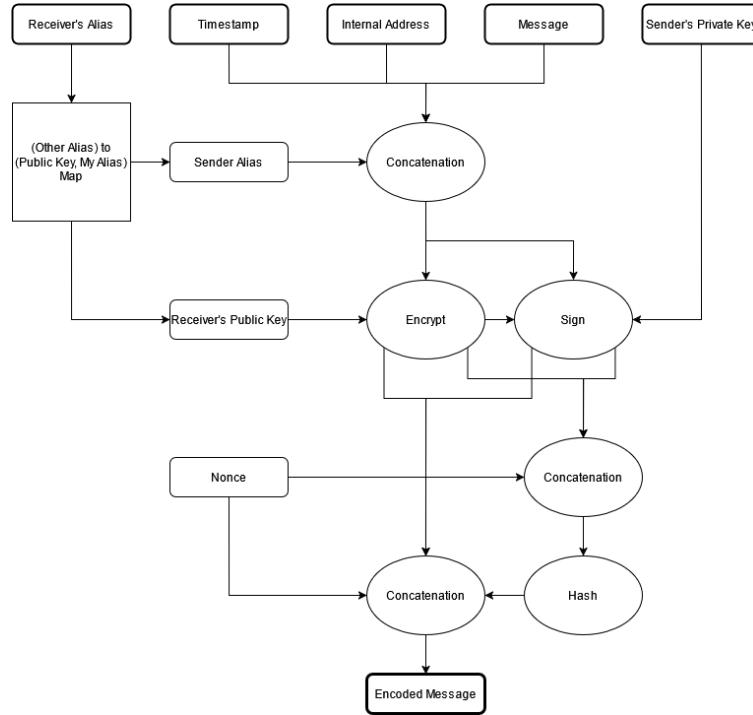


Figure 1: Message encoding process

**Hash:** It is used to determine whether the relaying nodes have already seen this message. It prevents infinite looping relays. A proof-of-work requirement must be met to minimize spam and tampering similar to Bitcoin [6].

**Encryption:** The encryption allows only the intended receiver to view the message and also metadata such as sender alias, internal address, and timestamp.

**Signature:** The sender’s signature allows the receiver to verify whether the message is actually sent by the sender. Inclusion of the timestamp in the signature serves as a simple counter to general replay attacks.

### 4 Decoding and Relaying

**Timestamp of reception:** The timestamp of data reception is recorded.

**Deduplication:** The hash and nonce are extracted and the proof-of-work is then verified. It is then verified whether the hash correctly matches the rest of the message contents. Every node keeps a record of hashes of previously relayed messages. If it has seen the hash, the message is discarded. If not, the hash is added

to the record and the message is relayed to all neighbors.

**Decryption:** The node then attempts decryption on the encrypted part. If decryption fails, the message is discarded. Otherwise, the original components are extracted (timestamp of creation, sender alias, internal address, and decoded message).

**Verification:** The node then looks up the sender's public key using the sender alias. If the alias is not found, the message is treated as **anonymous**, and the signature is not verified. Otherwise, the decryption product is used to verify the signature using the sender's public key.

**Result:** The decoded message data, timestamp of reception, timestamp of creation, sender alias, and internal address are the result of decoding.

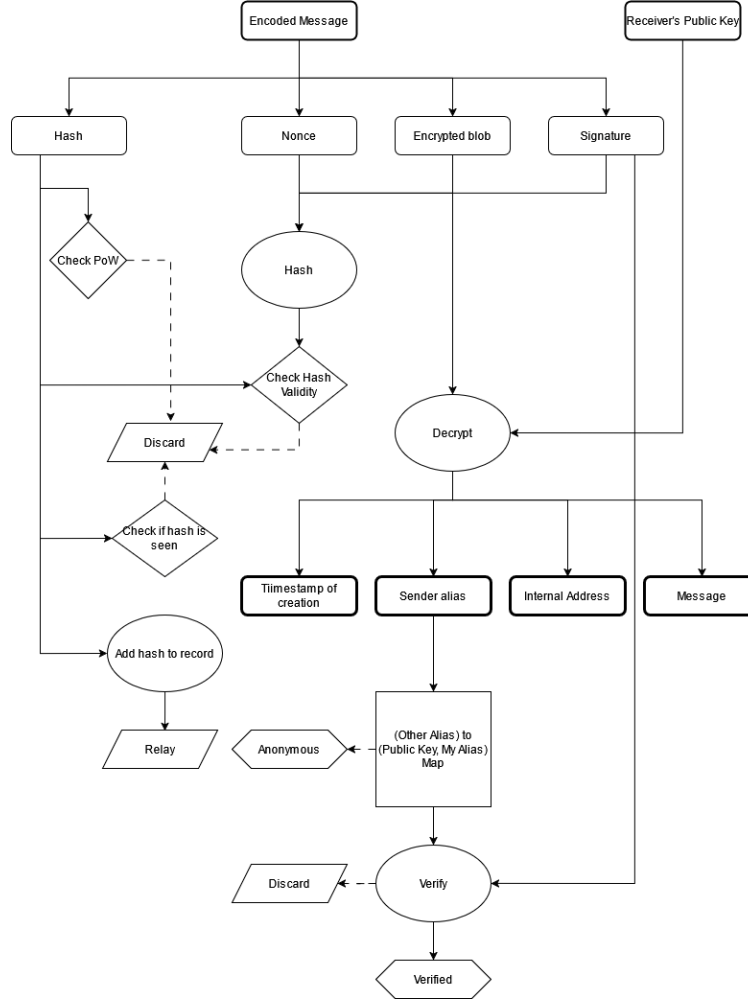


Figure 2: Message decoding and relay process

## 5 Optional Improvements

**Sector-based routing:** To improve efficiency in large networks, the message can be flooded in only specific regions called **sectors**, which is an arbitrary group of nodes. However, this may come at the cost of a greater risk of deanonymization of the receiver and sender. Moreover, the sender must know in which sector the receiver is in.

**Time To Live:** TTL is used in structured systems, but here topologies are unknown and dynamic. It may also make the deanonymization of the receiver and sender easier. It could be used for niche cases.

**Better Seen-Hash:** The record of hashes of previously relayed messages can grow uncontrollably. It may be cleared after a random time or after reaching a size limit. Bloom filters or other data structures may be used.

**Periodic key rotation:** Compromising a private key compromises all past messages encrypted to that key. Ephemeral keys or periodic key rotation may be used.

**Dummy traffic:** Attempts of traffic analysis can be nullified by nodes creating and relaying dummy messages and/or relaying messages only after a short duration of random time.

**Replay Attacks:** Prevented by including timestamps in the signatures. The hash remains the same which will result in an immediate message drop by the attacker’s neighbors, unless it brute-forces a new nonce. Therefore, simple rules can be implemented on top, including session or one-time tokens, message counters, or having a maximum message “age” (time difference between the current time and timestamp of creation).

## 6 Threat Model and Security Analysis

### 6.1 Adversary Model

**Passive Eavesdropper:** Observes network traffic without modifying it.

**Active Adversary:** Can inject, modify, or replay messages. May act as a participating node.

**Compromised Node:** A legitimate node acting maliciously, leaking keys or dropping messages.

**Global Passive Observer (GPO):** Observes all network traffic, attempting to correlate timing or metadata.

**Circling Adversary:** Controls a majority of surrounding nodes to localize message origin.

**Sybil Attacker:** Launches many fake nodes to gain influence or disrupt hashing and routing.

**Replay Attacker:** Re-broadcasts valid messages to cause confusion or traffic analysis.

**Denial-of-Service Attacker:** Floods the network with spam or invalid messages.

### 6.2 Security Properties

**Passive Eavesdropping:** Encryption hides message content and metadata, as encrypted blob is opaque to observers.

**Active Modification:** Messages are protected by digital signatures and hash verification. Tampered messages are discarded.

**Compromised Nodes:** Flooding ensures alternate paths unless a compromised node becomes a topological choke point.

**Global Observation:** Absence of fixed routes. Dummy traffic and delay tactics can obfuscate timing.

**Circling Attack:** Probabilistically avoided with sufficient node density and entropy.

**Sybil Attack:** No built-in protection but mitigated by optional proof-of-work or staking overlays.

**Replay Attack:** Timestamps in signatures and PoW hash prevent simple replays; message counters, maximum message age and tokens recommended.

**Denial-of-Service:** PoW requirements and deduplication mitigate spam and overload.

### 6.3 Comparison with Related Protocols

Feature	EFPIX	Tor [7]	Bitmessage [8]	Matrix [9]
Topology-agnostic	✓	✗	✓	✗
End-to-End Encryption	✓	✓	✓	✓
Metadata Protection	✓	Partial	Partial	✗
Anonymity	Probabilistic	Strong	Moderate	✗
Sybil Resistance	Partial	✗	✗	✗
Offline Messaging	✓	✗	✓	✓
Broadcast Suitability	✓	✗	✓	✗
GPO Resistance	Partial	✗	✗	✗

Table 1: Comparison of EFPIX with selected secure communication protocols

## 7 Future Work

Future work could explore efficient partial flooding strategies, reputation models, integration with radio mesh hardware, and secure bootstrap methods for alias public key exchanges. Implementations can simulate or deploy on different networks to provide performance and other results to compare against existing protocols that aim to solve a similar problem.

## 8 Conclusion

EFPIX provides a robust framework for secure and censorship-resistant communication, protecting against invasive and potentially malicious surveillance. It also conceals “non-content” data from prying eyes. Its decentralized, flooding-based design ensures delivery even in restricted or faulty environments, making it an ideal option for disconnected, totalitarian, or disaster-struck areas.

## References

- [1] J. Bamford, *The NSA Is Building the Country’s Biggest Spy Center (Watch What You Say)*, <https://www.wired.com/2012/03/ff-nsadatacenter>, 2012.
- [2] E. Heilman, *A Brief History of NSA Backdoors*, <https://ethanheilman.tumblr.com/post/70646748808/a-brief-history-of-nsa-backdoors>, 2014.
- [3] *Apple pulls data protection tool after UK government security row*, <https://www.bbc.com/news/articles/cgj54eq4vejo>, 2025.
- [4] R. L. Rivest, A. Shamir, and L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, <https://people.csail.mit.edu/rivest/Rsapaper.pdf>, 1978.
- [5] National Institute of Standards and Technology (NIST), *FIPS PUB 180-4: Secure Hash Standard (SHS)*, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>, 2015.
- [6] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, <https://bitcoin.org/bitcoin.pdf>, 2008.
- [7] R. Dingledine, N. Mathewson, and P. Syverson, *Tor: The Second-Generation Onion Router*, <https://www.usenix.org/events/sec04/tech/dingledine.html>, 2004.
- [8] J. Warren, *Bitmessage: A Peer-to-Peer Message Authentication and Delivery System* <https://bitmessage.org/bitmessage.pdf>, 2012.
- [9] M. Hodgson and A. Le-Prevost, *Matrix: An Open Network for Secure, Decentralized Communication* <https://matrix.org/docs/spec>, 2014.