EFPIX (Encrypted Flood Protocol for Information eXchange)

A zero-trust, flooding-based relay networking protocol for secure communication

Arin Upadhyay
arinupadhyay.cs@gmail.com
28 February, 2025

Abstract. We propose a cryptographic communication protocol designed to ensure anonymity and untraceability of messages while providing end-to-end encryption. It operates on a flooding-based relay system, making it resistant to infrastructure failures and also suitable for broadcast-type messages. It is also designed to hide non-content data, like sender and receiver, from those not involved. The sender only needs the receiver's public key to encode a message.

Introduction:

In recent years, the Internet has been transformed by government and invasive intelligence agencies to erode many privacy rights using various surveillance programs, laws, and backdoors.[1][2] They are not hesitant to restrict or outright ban encryption.[3] They have the power to subpoena any data, track network traffic, and force shut down services that counter these issues. This has made journalism, whistleblowing, and activism extremely dangerous. Any internet user that desires some privacy rights is powerless to protect their data from intrusive authorities. It is difficult to safely communicate over the internet even if your own devices are secure and uncompromised.
An attractive solution for this problem is an encrypted flooding-based relay protocol. They not only provide secure, server-less, censorship-resistant, and anonymous communication, but they are also resilient to infrastructure failures and network disruption and are topology-independent. They can be designed to even hide the source and the destination of messages. They can be used with radio communication to achieve wireless node-to-node fault-free networking. This has many possible applications, including space, scientific, and military fields, where the existence and maintenance of a central server is not feasible. They are also suitable to transfer broadcast-type messages that reach all nodes in the network, like emergency rescue calls, disaster warnings, news distribution, etc.
EFPIX is such a protocol that is designed with the above issues in mind. It uses RSA encryption and signatures along with a hashing function.[4] The protocol is designed to prevent identification of the source or the destination of the message while also providing encryption on top.

Overview:

A desired message is built into a specific structure before being relayed to any and all available nodes. Any relaying node checks if the message can be decrypted by its private key but relays it nonetheless. If the message is already seen by a node, it is not relayed and is not processed.
A "known" connection exists when both the sender and receiver know each other's public keys and agree upon "aliases.". Aliases are 16-byte strings that serve as a key/reference for a public key. The sender includes their alias in messages, allowing the recipient to retrieve the sender's public key for signature verification. If a message contains an unknown alias, it is treated as an anonymous message, and the signature is not verified. It is possible to build higher-layer protocols onto this protocol to serve various uses, including exchanging aliases and public keys to establish known connections.
In addition to encrypted communication, EFPIX supports unencrypted broadcast messages that every node receiving them can read and relay.

Encoding and Message structure:

(This explanatory version uses RSA 2048 with 11 bytes of OAEP padding along with SHA-512.)

Before encryption, any desired message is first formed into this structure:

timestamp (9 bytes)	sender alias (16 bytes)	internal address (4 bytes)	message (up to 216 bytes)

Timestamp: Contains the date and time of the message creation.

Sender alias: Provides the receiver with the “key” to look up the sender’s public key in its memory. It is the alias the receiver “knows” the sender as and can be different for different receivers.

Internal Address: Can be used in different ways; for instance, to address an internal client in case of a server that acts as a proxy for multiple clients or in case of multiple processes that can process the decoded message.

This construction is then encrypted using the receiver’s public key and also signed using the sender’s private key. These are used to generate a hash using the hashing function.

The final transmitted 576-byte structure is constructed as:

Hash (64 bytes)	Encrypted data (256 bytes)	Signature (256 bytes)

Decoding and Relaying:

Timestamp of reception: the data is recorded.

The first 64 bytes are extracted and verified as the valid hash for the rest of the data. Every node keeps a record of hashes of all previously relayed or received messages, and this record is cleared after a random time or after reaching a size limit. If it has seen the hash, the data is disregarded and discarded. If not, the hash is added to the record.

The node then attempts decryption on the encrypted part ([64:320]). If the decryption fails, the message is relayed unchanged. Else, the original components are extracted (timestamp of creation, sender alias, internal address, and the decoded messages). 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 “unknown” and “unverified.”. Else, the decryption product is used to verify the signature ([320:]) using the sender’s public key, which was retrieved using the sender’s alias.

The decoded message, received timestamp, sent timestamp, sender alias, internal address, etc., are the result of decoding.

The original received encoded data is relayed unchanged nonetheless.

Security Considerations:

• Replay Attacks: Prevented by including timestamps in the RSA signatures. Applications requiring stricter security can use one-time tokens.
• Traffic analysis: Creating and relaying dummy messages. Relaying messages only after a small duration of random time.
• Denial of Service and Spamming: Rate-limiting and proof-of-work in hashes can deter bot spamming.

Risks and Consequences:

• Message Size: The 216-byte limitation can be extended by using a higher-modulus RSA, at the cost of slower decoding times.
• Blackholing and tampering: A node can choose to only receive and not relay any data. A node can also tamper with the relayed data before relaying again. This is usually not a problem since the protocol verifies data using signatures and hashes and also relies on flooding. However, this can be critical if the blackholing or tampering node is a “chock-point” topologically.
• Circling: The source of a transmission can be identified if all the nodes that surround the source are compromised.

Conclusion:

EFPIX provides a robust framework for secure and censorship-resistant communication protected from hyper-invasive and potentially malicious surveillance. It also hides “non-content” data from prying eyes. Its decentralized, flooding-based design ensures delivery even in restricted or faulted environments.

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, 1977