STAMPED ON BITCOIN · UNPRUNABLE · SERVERLESS

Stampchat

A serverless, ephemeral, end-to-end encrypted messaging application stamped permanently on Bitcoin.

A permanent tool for impermanent talk: the application can never be deleted, and the conversation can never be recovered.

A B S T R A C T

Stampchat is a peer-to-peer messaging application distributed as a single, self-contained HTML file stamped permanently onto the Bitcoin blockchain as an unprunable Bitcoin Stamp. It establishes direct, end-to-end encrypted connections between two participants using WebRTC, with no signaling server, no application backend, no user accounts, and no message storage of any kind. Connection setup is performed through the manual exchange of a short connection descriptor, removing the last conventional server dependency. The system embodies a deliberate inversion: the tool itself is maximally permanent — immutable and impossible to delete — while every conversation it carries is maximally transient, existing only in the volatile memory of two browsers and vanishing without trace when the session ends. This paper describes the motivation, architecture, ephemerality model, and security properties of the system, and is explicit about the boundaries of what an infrastructure-free application can and cannot guarantee.

1 Introduction

Contemporary “private” messaging rests on a paradox: privacy is administered by the party best positioned to violate it.

Messages travel through, and usually rest on, servers operated by a provider. Identity is an account the provider issues and can revoke. History accumulates in databases that can be breached, subpoenaed, sold, or mined. Even rigorously end-to-end encrypted systems retain metadata and depend on infrastructure that must be trusted to remain available and honest. In practice, the user’s privacy is a promise — contingent on a company’s policies, solvency, and goodwill.

Stampchat explores the opposite premise: that the strongest privacy guarantee is not a better-administered server but the absence of one. If there is no server, there are no server logs. If there are no accounts, there is no identity to compromise. If there is no database, there is no history to leak. Properties that other systems must promise to enforce, Stampchat obtains structurally — by having nothing to enforce them over. The result is closer to a private conversation in a room than to a messaging service: two people arrive, speak, and leave, and the room retains nothing because it was never built to.

This design is made possible, and given its distinctive character, by the medium of distribution: the application is not hosted but stamped — written permanently into the Bitcoin blockchain as an unprunable artifact that no party owns, hosts, or can take down.

2 Background: Bitcoin Stamps and Unprunability

A **BITCOIN STAMP** is a method of embedding arbitrary data — here, the complete application — directly into the Bitcoin blockchain. The bytes of the file are not referenced by a transaction; they are the transaction’s payload, replicated across every participating node in the network.

The defining property is **UNPRUNABILITY**. Some on-chain data is stored in regions of the blockchain that a node is permitted to discard after validation to reclaim space. Stamp data is embedded such that every full node must retain it indefinitely as a condition of continued operation. The data therefore cannot be selectively pruned, garbage-collected, or quietly forgotten. It persists for as long as the chain itself.

For an application, this confers a set of properties no conventionally hosted software can match:

- **No host.** There is no web server, no domain, and no storage bucket. There is nothing to expire, nothing to bill, and nothing whose failure removes the application from existence.
- **Censorship resistance.** There is no operator to compel and no single location to block. Removing the application would require altering the Bitcoin blockchain.
- **Immutability.** The artifact cannot be edited in place. What was stamped is what runs, byte for byte, indefinitely. Immutability is also the central constraint. A stamped application can never be patched; a corrected version is an entirely new stamp. Every line must be correct at the moment of stamping, and — critically — the application must depend on nothing that can change or disappear after it. This constraint, more than any aesthetic choice, dictates the architecture that follows.

3 Design Philosophy

3.1 Permanence of the tool, impermanence of the conversation

Stampchat is organised around a single inversion. The application is the most permanent software artifact available: unprunable, immutable, and ownerless. The conversations are the most transient

communication possible: they exist only in the volatile memory of the two participating browsers, are never written to any disk, and cease to exist the moment a tab is closed or refreshed. The instrument is eternal; what it produces is written on water.

3.2 The immutability constraint forces self-containment

Because a stamped artifact can never be amended, it must not depend on anything that can rot. A conventional web application is an illusion of self-containment: a single visible page typically loads fonts, libraries, frameworks, and images from numerous external servers, any of which may one day disappear and break it. An unprunable application that loads even one external asset is merely a permanent shell around a perishable core.

Stampchat therefore loads nothing. It contains no external scripts, no web fonts, no remote images or icons, no frameworks, and no analytics. Opened on a machine with no network connection, it loads and runs in full. Its self-containment is not a feature added for convenience; it is a precondition of being permanent at all.

4 System Architecture

4.1 Single-file design

The entire application — structure, presentation, and logic — resides in one HTML document of roughly thirty kilobytes. All behaviour is implemented in vanilla JavaScript against native browser APIs; all styling is inline. There is no build artifact, no module graph, and no runtime dependency. This is both the unit of the stamp and the unit of execution.

4.2 Peer-to-peer transport

Communication uses **WEBRTC**, a native browser capability for establishing direct connections between two browsers. Once established, the connection is genuinely point-to-point: application data flows straight from one participant’s machine to the other’s, with no intermediary in the data path. Stampchat carries all messaging over a single reliable, ordered **RTCDataChannel**.

4.3 Manual signaling: the serverless bootstrap

WebRTC can carry a connection but cannot, by itself, initiate one: before a direct channel exists, the two browsers must exchange connection descriptors (an offer and an answer, each containing session parameters and candidate network paths). Conventionally this exchange is brokered by a signaling server — a live, always-on dependency, precisely the kind of rot risk a stamped artifact cannot tolerate.

Stampchat removes it by making the user the broker. The initiating participant generates an invite descriptor, which the application renders as a compact text token. The user transmits this token to their counterpart through any existing channel — a message, an email, a spoken dictation. The counterpart pastes it in, receives a reply token in return, and sends that back. With both descriptors exchanged, the direct channel opens. To make this a single round-trip rather than a continuous negotiation, candidate gathering is completed before a descriptor is emitted, so each descriptor is self-contained.

This manual step is deliberately retained as more than a workaround. It is a consent mechanism: a connection can only arise from two people who deliberately exchanged tokens. There is no directory, no discoverability, and no inbox — and therefore no unsolicited contact and no spam. A participant communicates only with counterparts to whom they personally handed a token.

4.4 Network address discovery and relay

To connect across distinct networks, each participant must learn the public address at which they can be reached. A device behind network address translation cannot observe this address directly. Stampchat resolves it using **STUN**, a lightweight protocol in which a public server reflects the requester’s observed address back to it. The **STUN** exchange is momentary, occurs only during setup, and never carries conversation content; the server is absent from the data path entirely. For resilience, the application is configured with independent public **STUN** servers and the field is user-editable, so no single provider is load-bearing and any can be replaced.

A minority of restrictive network configurations (symmetric or carrier-grade translation) admit no direct path even with address discovery. These require a **TURN** relay, which forwards traffic between participants. Because a relay is a persistent server, Stampchat never embeds one: the relay field ships empty, and a participant who requires one supplies it themselves for a single session. The permanent artifact thus references no relay; the dependency, when accepted, is the user’s temporary and informed choice.

4.5 Encryption

WebRTC mandates transport encryption. The data channel is secured with **DTLS**, with keys negotiated directly between the two browsers during connection setup. No third party participates in the key exchange. Consequently, every network element that carries the traffic — including, where used, a **STUN** or **TURN** server — observes only ciphertext.

Confidentiality of message content does not depend on trusting any server, because no server is positioned to read it.

4.6 Message model

All communication is expressed through a single abstraction: a typed envelope of the form { type, … }, serialised and dispatched on receipt to a handler selected by its type. Text, reactions, typing indicators, read receipts, presence announcements, and file-transfer control messages are each one envelope type. This uniformity means the message vocabulary can be extended — a new medium is one sender, one handler, and one renderer — without altering the transport.

4.7 Binary transfer

Images and arbitrary files are transmitted as binary frames over the same channel. A transfer is announced with a metadata envelope, then streamed as fixed-size chunks, each tagged with a transfer identifier and sequence number, and reassembled in order at the receiver. Transmission observes flow control: when the channel's outbound buffer grows beyond a threshold, sending pauses until it drains, preventing large transfers from overwhelming the connection. Images are rendered inline; other files are offered for download. A shared message identifier accompanies each transfer so that reactions and receipts address the same logical message on both sides.

4.8 Interaction layer

On top of plain text, the application provides reactions (one per participant per message, replaceable and removable), live typing indication, read receipts, and presence with user-chosen display names. Each is a control envelope of negligible size; none requires storage or a server. Together they provide the texture of a contemporary messenger while preserving the system's statelessness.

5 Ephemerality Model

Stampchat's privacy is obtained primarily through the absence of persistence. There is no server to log, no inbox to accumulate, and no database to retain history. All conversation state — messages, images, reactions — lives exclusively in browser memory for the duration of the session. Closing or refreshing the page destroys it completely; the destruction is not a deletion from storage but the simple disappearance of state that was never stored.

Two explicit controls extend this default:

- **Burn.** A single action wipes the conversation on both participants' devices and tears down the connection. It is a mutual, on-demand erasure: when one party ends a session, the other's copy is wiped in the same moment.
- **Auto-expire.** A session may be configured at the outset to self-destruct after a chosen interval — from ten minutes to twenty-four hours — even if left open and unattended. The countdown is mirrored to both participants, and both are wiped together when it elapses. This bounds the lifetime of a conversation that the participants neglect to close, the one case that closing-on-exit does not cover. The default lifetime is the session itself. Nothing outlives the tab unless a participant deliberately chooses, and even then only within the bounds they set.

6 Security and Privacy Model

An honest system states precisely what it protects and what it does not.

6.1 What is protected

- **Content confidentiality.** Message content is end-to-end encrypted and readable only by the two participants. No server, including any relay in use, is positioned to read it.
- **Absence of stored history.** The application writes no conversation data anywhere. There is no artifact for an adversary, an operator, or a legal process to obtain, because none is produced.
- **Connection by consent.** A session exists only because two parties deliberately exchanged descriptors. There is no mechanism by which an uninvited party can initiate contact.

6.2 What is not protected

- **Metadata, when a relay is used.** A **TURN** relay, though unable to read content, necessarily observes that two endpoints communicated, when, and how much. Direct connections expose no such intermediary; relayed sessions extend this limited trust to the relay's operator, which is why relays are opt-in and user-chosen.
- **Endpoint capture.** The application cannot govern the devices on which it runs. A participant may screenshot a conversation, and an operating system may briefly cache a rendered image. This is true of all messaging software and lies outside what any application can prevent. The precise claim is therefore that the application retains nothing — not that no record can be made by a participant.
- **Traffic analysis at the network layer.** While content is encrypted, the existence and timing of a connection may be observable to a network position that can see the participants' traffic, as with any real-time communication.

The system's posture is to claim only what its architecture delivers. Content privacy is structural and strong; the limits above are stated plainly rather than obscured.

7 Limitations

- **Connectivity is not universal.** Direct peer-to-peer connection succeeds for most network pairings but is impossible for some restrictive configurations without a relay. This is a property of internet addressing, not a defect of the application; the relay field exists precisely for these cases.
- **One-to-one by design.** The manual-signaling model and the direct-connection topology are scoped to two participants. Group communication is not addressed in this version.
- **Manual setup friction.** Exchanging descriptors by hand is less convenient than automatic introduction. This friction is the cost of requiring no signaling server, and is partially mitigated by completing the exchange in a single round-trip.

8 Future Directions

Several extensions are consistent with the system's constraints. A self-bootstrapping group mode would retain manual signaling only for a participant's first connection; once joined, an existing channel could relay the descriptors needed to introduce further participants, reducing setup from one exchange per pair to one per joiner. The typed-envelope model admits new media — structured cards, location, voice notes — as additive message types. Where the hosting context permits camera access, code scanning could ease descriptor exchange between co-present devices, though it offers little to remote participants on desktop machines. Each of these sharpens the existing design without introducing a permanent external dependency.

9 Conclusion

Stampchat demonstrates that a complete, modern messaging application can be made permanent and infrastructure-free at once. By stamping a self-contained artifact on Bitcoin, the tool becomes unprunable, immutable, and ownerless. By connecting peers directly through user-brokered signaling and encrypting end-to-end, the conversations require no server and leave no trace. The privacy is not a policy to be trusted but a consequence of an architecture that has nothing to retain and no one to retain it. The application endures permanently; the conversations vanish completely.

That the two coexist in a single thirty-kilobyte file is the point.

A Technical Summary

Artifact	Single self-contained HTML file (~30 KB), zero external dependencies
Distribution	Stamped permanently on Bitcoin (unprunable)
Transport	WebRTC RTCDataChannel (reliable, ordered)
Topology	Direct peer-to-peer, one-to-one
Signaling	Manual descriptor exchange; single round-trip (non-trickle ICE)
NAT traversal	STUN (redundant, editable); optional user-supplied TURN

Encryption	DTLS, keys negotiated peer-to-peer
Storage	None; conversation state in memory only
Erasure	Close/refresh; mutual burn; mutual timed auto-expire
Accounts	None; user-chosen display name per session
Media	Text, images (inline), arbitrary files (chunked, flow-controlled)

Stampchat is offered as described. Peer-to-peer connectivity depends on network conditions and is not guaranteed for every pairing. Content confidentiality is provided by WebRTC's mandatory transport encryption; users should understand the metadata and endpoint considerations in Section 6 before relying on the system for sensitive communication.