MarsCat

A Privacy-First Web3 Social
Running on Peer-to-Peer Networks

Version 1.0 — 2026

MarsCat Global
contact@marscat.io

Table of Contents

Abstract

MarsCat is a privacy-first social platform designed to operate natively on peer-to-peer networks, built upon the RelayX decentralized protocol [1]. Unlike conventional messaging platforms that retrofit privacy features onto centralized architectures, MarsCat embeds privacy as a foundational design constraint across three independent dimensions: **identity privacy** (zero real-world information requirement), **data privacy** (Blowfish encryption with infinite-solution properties for text and a novel Shredder Protocol for attachments), and **social relationship privacy** (unique cryptographic identities generated per friendship, preventing social graph reconstruction). The platform further introduces MarsApp, a decentralized application deployment model enabling developers to distribute applications through the P2P network with inherent DDoS resistance and service privacy protection. This whitepaper formalizes the three-dimensional privacy model, details the cryptographic protocols for messaging and attachment transmission, and describes the MarsApp ecosystem architecture.

---

1. Introduction

The global communications landscape is dominated by centralized platforms that operate under a fundamental contradiction: they promise private communication while architecturally requiring complete access to user data, social graphs, and metadata [2]. Even platforms that implement end-to-end encryption—such as Signal [3] and WhatsApp—retain access to extensive metadata including contact lists, message timing, frequency patterns, and device fingerprints. This metadata alone is sufficient to reconstruct intimate details of users' social lives, political affiliations, and daily routines [4].

The emergence of blockchain technology and decentralized protocols has created new possibilities for communication systems that do not require centralized trust. However, most decentralized messaging projects address only one dimension of privacy—typically content encryption—while neglecting identity privacy, relationship privacy, and the broader application ecosystem. A truly private social platform must protect not only *what* users say, but also *who* they are, *who* they communicate with, and *how* their social relationships are structured.

MarsCat addresses this comprehensive privacy challenge through a three-dimensional privacy model that simultaneously protects user identity, message content, and social relationship structure. Built upon the RelayX peer-to-peer protocol [1], MarsCat operates without any centralized infrastructure—there are no servers to subpoena, no databases to breach, and no corporate entities with the ability to access user communications.

Beyond its core social functionality, MarsCat introduces MarsApp, a decentralized application deployment and distribution model that enables developers to publish and operate applications entirely within the P2P network. MarsApp provides inherent DDoS resistance, allows deployment on any device in any location, and protects backend services from direct exposure through P2P message relay.

This whitepaper is organized as follows. Section 2 presents the problem statement. Section 3 describes the system architecture. Section 4 formalizes the three-dimensional privacy model. Section 5 details core communication protocols. Section 6 describes the MarsApp ecosystem. Section 7 presents security analysis. Section 8 compares with existing platforms. Sections 9 through 11 discuss use cases, future work, and conclusions.

Design Principles

MarsCat is guided by five core design principles that inform every architectural decision: **Privacy by Design**—privacy is not a feature added to an existing system, but the foundational constraint around which the entire architecture is built; **Zero Trust**—no node, operator, or component in the system is presumed trustworthy; security guarantees emerge from cryptographic mechanisms rather than operational trust; **Defense in Depth**—multiple independent layers of protection ensure that failure of any single mechanism does not compromise overall privacy; **Minimal Metadata**—the system is designed to generate, transmit, and store the absolute minimum metadata necessary for operation; **User Sovereignty**—users maintain unconditional control

over their identity, data, and social relationships, with no dependency on any external authority.

2. Problem Statement and Motivation

Contemporary social platforms suffer from interconnected privacy failures that MarsCat is designed to address comprehensively:

Identity Exposure. Every major platform requires real-world identity anchors—phone numbers, email addresses, or government identification—as prerequisites for account creation. This creates permanent links between digital activity and physical identity, enabling mass surveillance by both state and corporate actors [5]. For journalists, activists, and dissidents in authoritarian regimes, this linkage can be life-threatening.

Content Vulnerability. While end-to-end encryption has become more common, many implementations are compromised by key escrow mechanisms, client-side scanning proposals, or reliance on centralized key distribution servers [6]. Furthermore, standard encryption schemes produce ciphertext that is distinguishable from random data, allowing adversaries to identify encrypted communications and target them for cryptanalysis.

Social Graph Exposure. Even with encrypted content, centralized platforms retain complete knowledge of users' social graphs—who communicates with whom, how frequently, and at what times. Research has demonstrated that social graph metadata alone can reveal political affiliations, romantic relationships, medical conditions, and criminal associations with high accuracy [4]. No existing mainstream platform protects against social graph reconstruction.

Attachment Vulnerability. File transfers in conventional platforms traverse centralized servers where they can be intercepted, scanned, and stored. Even encrypted file transfers typically use a single encrypted blob that, if the key is compromised, reveals the complete file contents. There is no defense-in-depth for attachment privacy.

Centralized Application Dependencies. Social platforms increasingly serve as application ecosystems (mini-programs, bots, integrations), but these applications invariably depend on centralized hosting infrastructure that is vulnerable to censorship, DDoS attacks, and service provider takedowns.

These five failures are not independent—they compound each other. A platform that requires phone number registration and operates through centralized servers simultaneously exposes identity, content (through server-side access), social graphs (through contact lists and message routing metadata), and application availability (through centralized hosting). Addressing only one failure while leaving others intact provides a false sense of security that may be worse than no encryption at all, as it encourages sensitive communication on a platform with known vulnerabilities [6].

MarsCat's approach is to address all five failures simultaneously through a unified architectural design, ensuring that privacy protection is comprehensive rather than piecemeal. The three-dimensional privacy model—covering identity, data, and relationships—combined with the decentralized MarsApp ecosystem provides a coherent response to the full spectrum of privacy threats facing modern digital communication.

3. System Architecture Overview

MarsCat is organized as a three-layer architecture (Figure 1): the Network Layer providing P2P connectivity through the RelayX protocol [1], the Privacy Layer implementing three-dimensional privacy protection, and the Application Layer delivering social messaging and the MarsApp ecosystem.

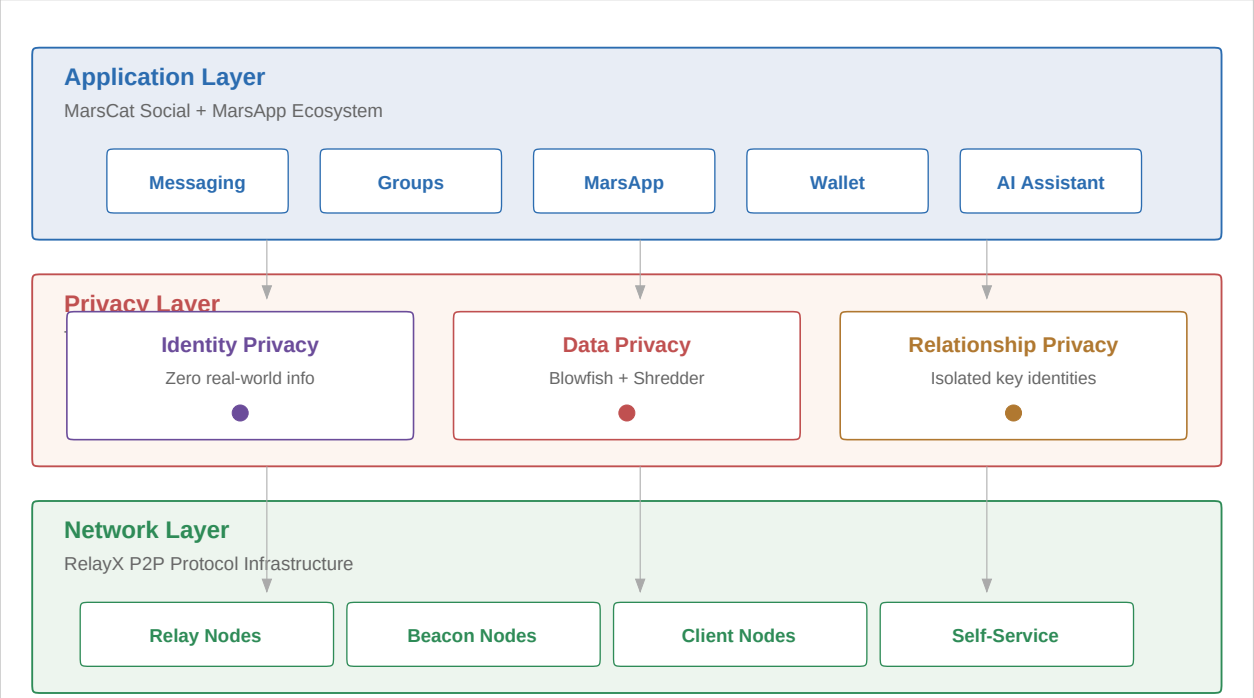

Fig. 1. MarsCat Three-Layer Architecture

Figure 1. MarsCat Three-Layer Architecture. The Network Layer provides P2P infrastructure via RelayX. The Privacy Layer implements identity, data, and relationship privacy. The Application Layer delivers social features and the MarsApp ecosystem.

3.1 Three-Layer Architecture

The **Network Layer** is provided by the RelayX Protocol [1], a purpose-designed P2P infrastructure comprising four node types: Relay Nodes (core routing hubs), Beacon Nodes (P2P backbone), Client Nodes (edge participants on user devices), and Self-Service Nodes (P2P-to-HTTP bridges). This layer handles all message routing, peer discovery, and network topology maintenance. MarsCat inherits RelayX's asymmetric send/receive channel architecture: outbound messages traverse the P2P gossip network for sender anonymity, while inbound messages are pushed via persistent Socket connections for real-time delivery.

The **Privacy Layer** implements MarsCat's three-dimensional privacy model, operating independently of the network transport. Identity privacy is enforced through key-only identification with no registration requirements. Data privacy is provided through Blowfish encryption for text content and the Shredder Protocol for attachments. Social relationship privacy is achieved through per-friendship key generation that prevents social graph reconstruction.

The **Application Layer** provides the user-facing social platform functionality—messaging, group conversations, file sharing, voice messages—and the MarsApp decentralized application ecosystem. Applications at this layer interact with the Privacy and Network layers through standardized APIs, ensuring that all communication inherits the full privacy protection stack regardless of the specific feature being used.

3.2 RelayX Protocol Foundation

The RelayX Protocol [1] serves as the foundational network infrastructure for MarsCat. Key protocol characteristics include: a fixed-length message header (34-byte BTC-address Application ID, 44-byte sender/receiver identities, 2-byte version, 4-byte payload length) supporting messages up to 10 MiB; a health-score-based peer management system providing organic Sybil resistance [7]; a 2-2-1 routing fan-out strategy (2 officially certified + 2 self-signed + 1 uncertified node) balancing reliability with decentralization; and support for both IPv4 and IPv6 dual-stack connectivity.

MarsCat extends the RelayX message format by utilizing the Payload field to carry its application-specific cryptographic envelopes, including Blowfish-encrypted text messages, Shredder Protocol fragment metadata, and per-friendship key exchange handshakes. This layered approach ensures that the network transport layer and the application privacy layer can evolve independently.

3.3 Network Topology and Node Roles

Understanding the network topology is essential for appreciating how MarsCat achieves both performance and privacy. The four node types operate in complementary roles within a hybrid topology that combines the reliability of structured infrastructure with the censorship resistance of unstructured P2P networks.

Relay Nodes serve as the network's core routing infrastructure. Each Relay Node contains two sub-modules: a Beacon Node module that participates in the P2P mesh for message reception and forwarding, and a Relay Hub module that provides HTTPS/WebSocket endpoints for client connectivity. This dual-module architecture enables the asymmetric channel design: outbound messages from clients enter the P2P network through the Beacon module (preserving sender anonymity), while inbound messages are pushed to clients through the Relay Hub (ensuring real-time delivery). Relay Nodes are deployed on enterprise-grade servers with high bandwidth and low latency, and project operators may deploy dedicated Relay Nodes for specific application communities.

Beacon Nodes form the P2P backbone, maintaining network topology through randomized TCP connections with peer nodes. Each Beacon Node validates incoming messages (checking ECDSA signatures and timestamps), forwards valid messages to its peer set, and maintains a local peer list sorted by certificate tier and health score. Beacon Nodes provide the randomized routing paths essential for sender anonymity—the more Beacon Nodes a message traverses, the more difficult it becomes to trace the message back to its origin.

Client Nodes are embedded in the MarsCat application on user devices. Each Client Node operates with a dual identity (detailed in Section 5): as a P2P network participant relaying messages for others and contributing to network resilience, and as a secure runtime container executing MarsApps and managing the user's cryptographic key material. Client Nodes with public IP addresses contribute directly to the global P2P mesh, while nodes behind NAT use Relay Nodes as connectivity bridges through the RelayX protocol's NAT traversal mechanisms.

Self-Service Nodes bridge the P2P network and traditional backend applications through bidirectional protocol translation (HTTP $\leftrightarrow$ P2P). Deployed by MarsApp developers, these nodes enable backend services to operate within the P2P network without public internet exposure. The Self-Service Node receives P2P messages addressed to its application, converts them to HTTP requests for the local backend, and returns HTTP responses as P2P messages through the network. This architecture eliminates DDoS attack surfaces and prevents IP address leakage in both directions.

3.4 Data Flow Architecture

Data within the MarsCat ecosystem follows a well-defined flow through the three architectural layers. When a user sends a message, the data traverses the following path:

Step 1 — Application Layer Processing: The MarsCat client receives user input (text, file, or command) and applies application-level logic: text messages are passed to the Blowfish encryption pipeline; attachments are processed through the Shredder Protocol; friend requests trigger the key generation and exchange handshake.

Step 2 — Privacy Layer Transformation: The appropriate privacy mechanism transforms the data: Blowfish encryption with ECDH-derived session keys for text; fragmentation, per-fragment encryption, and route randomization for attachments; per-friendship key selection for all message types.

Step 3 — Network Layer Transport: The encrypted payload is encapsulated in a RelayX message with the appropriate friendship-specific sender/receiver identities, submitted to the P2P network through the Client Node's connection to the Beacon mesh, and routed through multiple hops to reach the destination Relay Node.

Step 4 — Delivery: The Relay Node caches the message and pushes it to the receiver's Client Node via the persistent Socket connection. The receiver's client reverses the privacy layer transformations (signature verification, decryption, fragment reassembly) and presents the original content to the user.

This four-step flow ensures that privacy protection is applied consistently regardless of the specific feature being used, and that no single layer has access to both plaintext content and network routing information.

4. Three-Dimensional Privacy Model

MarsCat's privacy architecture is built on the principle that comprehensive privacy requires simultaneous protection across three orthogonal dimensions: identity, data, and social relationships. Weakness in any single dimension can compromise the others—for example, strong content encryption is meaningless if the social graph reveals who is communicating with whom. This section formalizes each dimension.

4.1 Identity Privacy: Zero Real-World Information

MarsCat enforces a strict zero-knowledge identity model in which no real-world information is ever collected, stored, or transmitted. The system requires no email address, phone number, government ID, biometric data, or any other personally identifiable information (PII) at any point in the user lifecycle—not during registration, not during use, and not during account recovery.

User identity is defined exclusively by cryptographic keypairs generated locally on the user's device. A master keypair is generated during initial setup using a cryptographically secure random number generator (CSPRNG) seeded from device entropy sources [8]. The master public key serves as the user's root identifier, but is never directly exposed to other users or the network at large.

Instead of exposing the master key, MarsCat derives friendship-specific keys from the master keypair (detailed in Section 4.4), ensuring that no single key is associated with multiple relationships. The master key's sole function is to serve as the deterministic root from which all derived keys can be regenerated—enabling account recovery on a new device without requiring any centralized backup service.

This architecture provides several critical properties: (i) no registration database exists that could be breached or subpoenaed; (ii) no centralized authority can revoke user access; (iii) users cannot be de-anonymized through account metadata; (iv) the system is inherently compliant with data minimization principles under frameworks such as GDPR [9], as no personal data is collected in the first place.

Comparison with Alternative Identity Models

To contextualize MarsCat's identity approach, we compare it with three prevalent models. **Phone-number identity** (Signal, WhatsApp, Telegram) permanently links digital activity to a government-registered physical identity. Even if communication content is encrypted, the phone number itself is a persistent identifier that enables cross-platform tracking, law enforcement identification, and social engineering attacks. **Email-based identity** (traditional messaging, many Web3 platforms) is marginally better but still requires a persistent identifier that is often linked to real-world identity through the email provider's registration requirements.

Single-key identity (Session, Briar, early blockchain wallets) eliminates the real-world anchor but creates a different problem: a single public key becomes a persistent pseudonym that, over time, accumulates contextual information sufficient for de-anonymization. Research has shown that behavioral patterns, writing style, and communication timing associated with a persistent pseudonym can often be linked to a real-world identity with high probability [4]. MarsCat's **per-friendship key identity** model avoids this accumulation by ensuring that no single key is associated with more than one social relationship, fundamentally limiting the contextual information available to any observer.

Account Recovery Without Centralized Backup

A common objection to key-based identity systems is the risk of key loss. MarsCat addresses this through a mnemonic-based recovery mechanism following BIP-39 conventions [13]. During initial setup, the master key is

encoded as a 24-word mnemonic phrase that the user records and stores securely. On a new device, the user enters the mnemonic to regenerate the master key, from which all friendship keys can be deterministically re-derived using the stored salt values (which are included in the encrypted local database backup). No centralized key escrow, cloud backup service, or recovery server is involved at any point. The recovery process is entirely local and cryptographic, preserving the zero-trust architecture.

4.2 Data Privacy: Blowfish Encryption and the Infinite-Solution Property

MarsCat employs Blowfish [10] as its primary symmetric cipher for text message encryption, chosen specifically for a property we term the **infinite-solution characteristic**: when an attacker attempts brute-force decryption of a Blowfish-encrypted ciphertext, every possible key produces a valid-appearing decryption output. Unlike AES, where incorrect keys typically produce identifiable random noise, Blowfish's block structure ensures that any key applied to any ciphertext yields output that is indistinguishable from meaningful plaintext without additional context.

This property has profound implications for brute-force resistance. In conventional encryption, an attacker who exhaustively tries all possible keys can identify the correct key by checking whether the decrypted output "looks like" valid plaintext (e.g., valid UTF-8 text, recognizable language patterns). With Blowfish's infinite-solution property, this distinguishing criterion is eliminated: the attacker obtains an effectively infinite number of plausible plaintexts with no cryptographic means to determine which is genuine [10].

To illustrate: suppose an attacker intercepts a Blowfish-encrypted message and attempts to decrypt it with every possible key. For a 128-bit key space, this yields 2^{128} candidate plaintexts. Unlike AES where the vast majority of candidates would appear as random bytes, Blowfish's structure produces outputs that include valid-looking text strings, alternative plausible messages, and even syntactically correct sentences in the target language. The attacker faces an information-theoretic barrier: without independent knowledge of the plaintext, there is no algorithmic criterion to select the correct decryption from the set of plausible candidates.

This provides a qualitatively different security guarantee from conventional ciphers. Standard encryption relies on *computational* security: the assumption that the attacker lacks sufficient computing power to try all keys. Blowfish's infinite-solution property adds an additional layer of *information-theoretic* resistance: even with unlimited computing power, the attacker cannot identify the correct plaintext without the key. This dual protection—computational and information-theoretic—motivates MarsCat's selection of Blowfish as its text cipher [10].

The encryption workflow operates as follows. For each message, a fresh session key is derived using ECDH (Elliptic Curve Diffie-Hellman) key agreement between the sender's ephemeral key and the receiver's friendship-specific public key. This session key is used as the Blowfish encryption key for the message content. The sender then computes an ECDSA signature over the ciphertext using their friendship-specific private key, providing authenticity verification. The complete package—ephemeral public key, ciphertext, and signature—is transmitted through the RelayX network.

Upon receipt, the receiver: (1) verifies the ECDSA signature to confirm message integrity and authenticity; (2) performs ECDH key agreement using the ephemeral public key and their own friendship-specific private key to derive the session key; (3) decrypts the ciphertext using Blowfish with the derived session key. If signature verification fails, the message is silently discarded, providing protection against tampering and injection attacks.

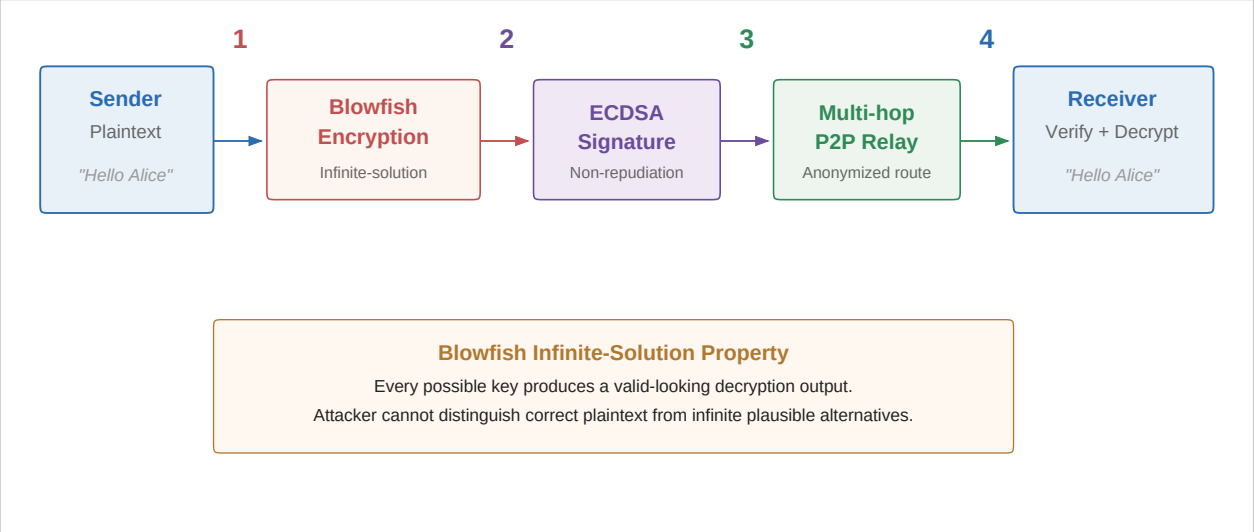

Figure 2. Text Message Encryption Flow. Plaintext is encrypted with Blowfish (step 1), signed with ECDSA (step 2), relayed through multi-hop P2P (step 3), and verified/decrypted by the receiver (step 4). The Blowfish infinite-solution property ensures brute-force attacks produce infinite plausible but incorrect plaintexts.

4.3 Data Privacy: The Shredder Protocol for Attachments

For attachment data—images, videos, voice messages, and files—MarsCat implements the **Shredder Protocol**, a novel defense-in-depth mechanism that provides privacy protection even if the encryption of individual fragments is compromised. The protocol draws conceptual inspiration from secret sharing schemes [11] and information-dispersal algorithms [12], adapted for real-time P2P communication.

The Shredder Protocol operates in five phases: (1) **Fragmentation:** The original file is split into N fragments of configurable size, with each fragment receiving a unique fragment identifier and sequence number. (2) **Per-Fragment Encryption:** Each fragment is independently encrypted using a unique key derived from the session key and the fragment identifier, ensuring that compromise of one fragment's key does not reveal others. (3) **Independent Routing:** Each encrypted fragment is transmitted through the RelayX network as an independent P2P message, taking a different multi-hop route through the network. (4) **Reassembly Metadata:** The sender transmits a separate encrypted metadata message containing the fragment identifiers, sequence order, and decryption keys for all fragments. (5) **Receiver Reassembly:** The receiver collects all fragments, verifies their integrity, decrypts each individually, and reassembles the original file.

The security implications of this design are significant. An attacker intercepting network traffic at any single point—or even at multiple points—cannot collect all fragments of a given attachment, as each fragment traverses a different network path. Even if an attacker obtains a subset of fragments, the individually encrypted fragments are meaningless without the complete set and the reassembly metadata. This provides a level of attachment privacy that is fundamentally stronger than encrypting the file as a single blob [12].

Security Properties of the Shredder Protocol

The Shredder Protocol achieves several formal security properties: **Fragment confidentiality:** Each fragment is encrypted with a unique key derived from the session key and fragment ID. Compromise of any individual fragment key reveals only that fragment's content, which is meaningless without the complete file context. **Fragment unlinkability:** An observer monitoring the P2P network cannot determine that two fragments belong to the same file, as they carry different source identifiers, traverse different routes, and are temporally separated by random jitter.

Threshold resistance: The protocol provides an all-or-nothing property—an adversary must obtain *all* N fragments plus the reassembly metadata to reconstruct the file. Possession of N-1 fragments yields zero information about the original file content. This is analogous to an (N,N)-threshold secret sharing scheme [11], but

with the additional advantage that each share (fragment) is independently encrypted, providing defense-in-depth even if the threshold property is somehow circumvented.

Deniability: Because fragments are individually encrypted and routed through different network paths, there is no single network observation point that reveals the complete file transfer. The sender can plausibly deny having transmitted any specific file, as the fragments they transmitted are indistinguishable from other encrypted P2P traffic. This property is particularly valuable in jurisdictions where the mere act of transmitting certain file types may carry legal consequences.

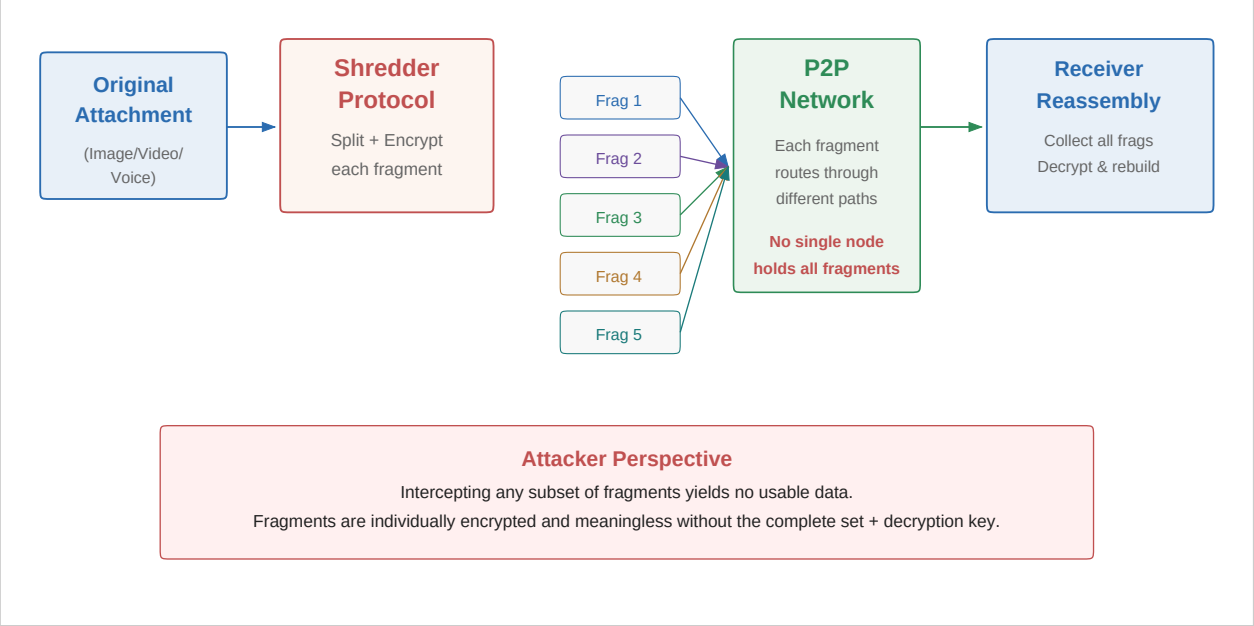

Figure 3. Shredder Protocol for Attachment Privacy. Original files are split into independently encrypted fragments, each routed through different P2P paths. No single network node holds sufficient fragments to reconstruct the file. The receiver collects all fragments and reassembles the original.

4.4 Social Relationship Privacy: Isolated Key Identities

MarsCat's most distinctive privacy innovation is its **per-friendship isolated key identity** system. In conventional messaging platforms—including end-to-end encrypted ones like Signal—each user has a single identity (public key or phone number) that is shared with all contacts. This single-identity model enables social graph reconstruction: if an adversary learns that Alice communicates with both Bob and Carol, the adversary knows that Bob and Carol share a mutual contact.

MarsCat eliminates this vulnerability entirely. When Alice adds a new friend, the system generates a completely new, independent keypair specifically for that friendship. If Alice has 10 friends, she possesses 10 distinct cryptographic identities—one for each friendship. These identities share no mathematical relationship that an external observer could detect:

• Bob knows Alice only as public key $PK_{A \to B}$
• Carol knows Alice only as public key $PK_{A \to C}$
• Dave knows Alice only as public key $PK_{A \to D}$

There is no cryptographic or network-level mechanism by which Bob, Carol, or Dave can determine that they share a common contact. Each friendship exists in complete isolation.

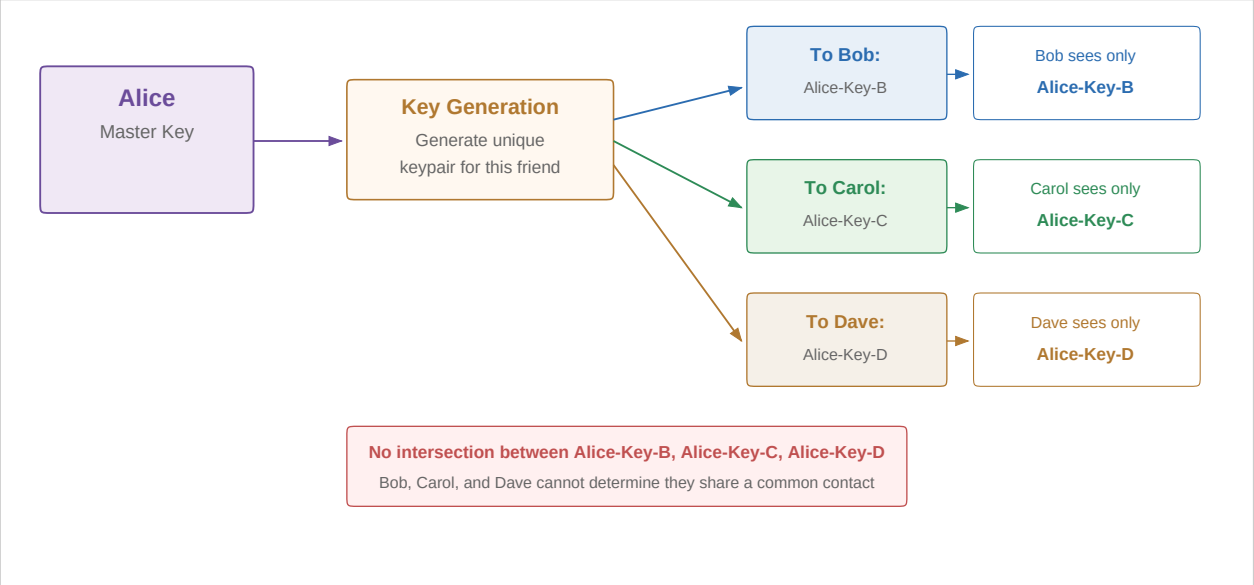

Figure 4. Per-Friendship Key Generation. Alice generates a unique keypair for each friend. Bob, Carol, and Dave each see a completely different Alice identity with no detectable intersection, preventing social graph reconstruction.

The key derivation process uses a hierarchical deterministic (HD) key generation scheme [13] rooted in the user's master key. Each friendship key is derived using a unique derivation path incorporating a random salt generated during the friend-adding handshake. This ensures that: (i) all friendship keys can be deterministically regenerated from the master key for account recovery; (ii) no friendship key reveals any information about the master key or other friendship keys; (iii) the compromise of any single friendship key does not affect other friendships.

Formal Properties of Key Isolation

Let MK denote Alice's master private key, and let $FK_i = \text{KDF}(MK, salt_i)$ denote the friendship key derived for the i-th friendship, where KDF is a key derivation function (e.g., HKDF-SHA256 [8]) and $salt_i$ is a unique random value. The isolation property requires that for any polynomial-time adversary A, given $FK_1, ..., FK_n$ (excluding FK_j), the probability that A can compute FK_j is negligible in the security parameter. This follows directly from the pseudorandomness of KDF under the random oracle model.

Furthermore, the scheme provides **unlinkability**: given any two friendship public keys $PK_{A \to B}$ and $PK_{A \to C}$, no efficient algorithm can determine whether they belong to the same master key holder. This property is essential for preventing social graph reconstruction—even if an adversary collects all public keys visible on the network, they cannot cluster them into user-level profiles.

Key Lifecycle Management

MarsCat implements comprehensive key lifecycle management: **Generation**—friendship keys are generated during the friend-adding handshake using OS-provided CSPRNG entropy; **Storage**—all keys are encrypted at rest using the device's secure enclave or a user-provided passphrase; **Rotation**—users can initiate key rotation for any friendship, generating a new keypair and re-keying the session through an authenticated protocol exchange; **Revocation**—when a friendship is terminated, the corresponding key is marked as revoked and all associated session keys are destroyed; **Recovery**—the HD derivation scheme enables complete key recovery from the master key and the stored salt values, which can be backed up through a mnemonic phrase following BIP-39 conventions [13].

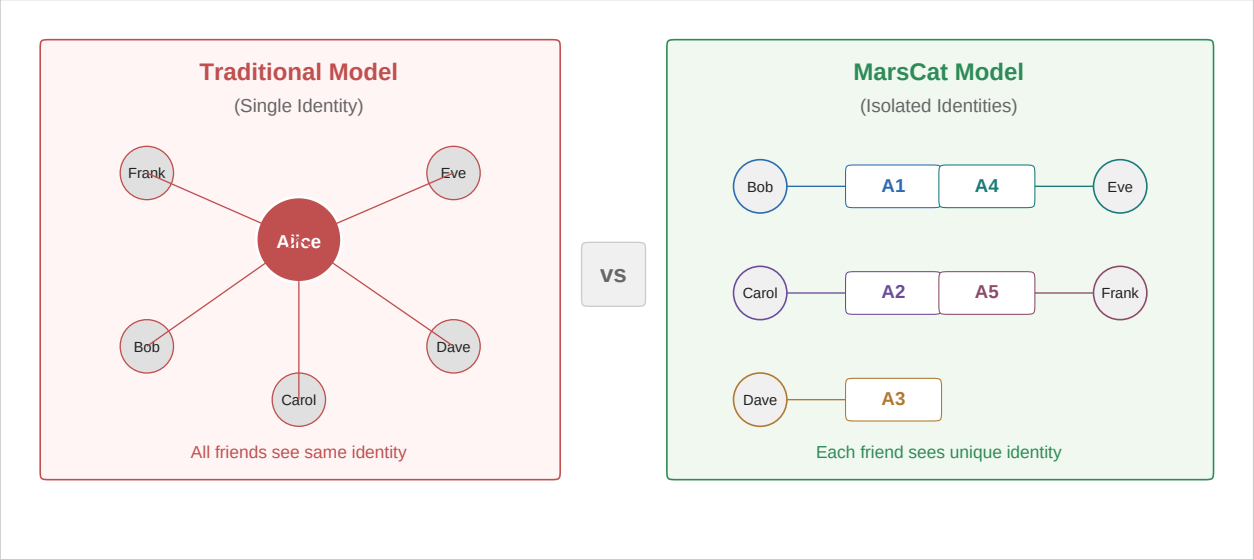

Figure 5. Social Relationship Privacy Comparison. Left: Traditional model where all friends see the same Alice identity, enabling social graph reconstruction. Right: MarsCat model where each friend sees a unique, isolated identity (A1–A5).

5. Core Communication Protocols

This section details the operational protocols for MarsCat's core social features, showing how the three-dimensional privacy model is realized in specific communication workflows.

5.1 Friend Discovery and Key Exchange

Friend discovery in MarsCat does not rely on centralized contact servers or phone number matching. Instead, users share friendship invitation codes—compact representations of their master public key and network routing hints—through out-of-band channels (QR codes, NFC, or manual text exchange).

The friend-adding handshake proceeds as follows: (1) Alice generates a new friendship keypair $(SK_{A\to B}, PK_{A\to B})$ derived from her master key with a fresh random salt. (2) Alice encrypts her friendship public key and the salt using Bob's master public key (obtained from the invitation code) via ECIES. (3) The encrypted handshake message is transmitted through the RelayX network. (4) Bob receives the handshake, decrypts Alice's friendship public key, generates his own friendship keypair $(SK_{B\to A}, PK_{B\to A})$, and sends his friendship public key back to Alice. (5) Both parties now possess each other's friendship-specific public keys and can communicate with full privacy guarantees.

Invitation Code Structure

A MarsCat invitation code is a compact binary structure encoded as a Base58 string or QR code. It contains: the user's master public key (33 bytes, compressed secp256k1 point), a network routing hint (optional, 4–16 bytes indicating preferred Relay Nodes for initial contact), and a version byte enabling future protocol extensions. The total invitation code length is 38–50 bytes, represented as approximately 52–68 Base58 characters—compact enough for manual transcription or embedding in a QR code.

Critically, the invitation code itself does not contain any personally identifiable information. It is a mathematical object (a point on an elliptic curve) that enables encrypted communication but reveals nothing about the user's real-world identity. Users can generate and share invitation codes through any channel—printed on a business card, posted on a website, whispered in person, or transmitted through another secure channel—without compromising their privacy.

Mutual Authentication

The friend-adding handshake provides mutual authentication without a trusted third party. Both parties verify each other's identity through the cryptographic properties of the key exchange: Alice proves her identity to Bob by encrypting a challenge using the keypair she generated for this friendship, and Bob proves his identity by responding with a correctly encrypted acknowledgment. This exchange also establishes the shared session parameters (initial sequence numbers, message format version, supported features) for the new friendship channel.

5.2 Text Message Encryption and Delivery

Text message transmission follows the encryption pipeline described in Section 4.2 and illustrated in Figure 2. The sender encrypts the plaintext with Blowfish using a session key derived via ECDH, signs the ciphertext with ECDSA, and transmits the package through the RelayX network. The receiver verifies the signature and decrypts the content.

Message delivery leverages the asymmetric channel architecture of the RelayX protocol: the sender's message is broadcast through the P2P gossip network (uplink path), providing sender anonymity through multi-hop relay. The message is cached at a Relay Node and pushed to the receiver via a persistent Socket connection (downlink path), ensuring real-time delivery without P2P propagation delay.

Detailed Message Envelope Structure

Each text message is encapsulated in a multi-layer envelope that provides authentication, encryption, and routing metadata. The complete message structure is as follows:

Layer	Field	Size	Purpose
RelayX Header	Application ID	34 B	MarsCat namespace (BTC addr)
	Sender Identity	44 B	Friendship public key
	Receiver Identity	44 B	Receiver friendship public key
	Version + Length	6 B	Protocol version and payload size
Crypto Envelope	Ephemeral PK	33 B	For ECDH session key derivation
	ECDSA Signature	64–72 B	Sender authentication
	Blowfish Ciphertext	Variable	Encrypted message content
	Timestamp + Nonce	12 B	Replay attack prevention

Table 2. Complete Message Envelope Structure

Replay Attack Prevention

Each message includes a monotonically increasing timestamp and a random nonce. Receiving nodes maintain a sliding window of recently seen nonces (configurable, default 24 hours). Messages with duplicate nonces or timestamps outside the acceptable window are silently discarded. This prevents replay attacks while accommodating reasonable clock skew between nodes in the P2P network.

Message Ordering and Delivery Guarantees

MarsCat provides **eventual delivery** with **causal ordering** guarantees. Messages between any pair of users are delivered in the order they were sent, ensured through sequence numbers within each friendship session. If a message arrives out of order, the client buffers it until preceding messages are received. The system does not guarantee global ordering across different conversations, as this would require centralized coordination incompatible with the privacy model.

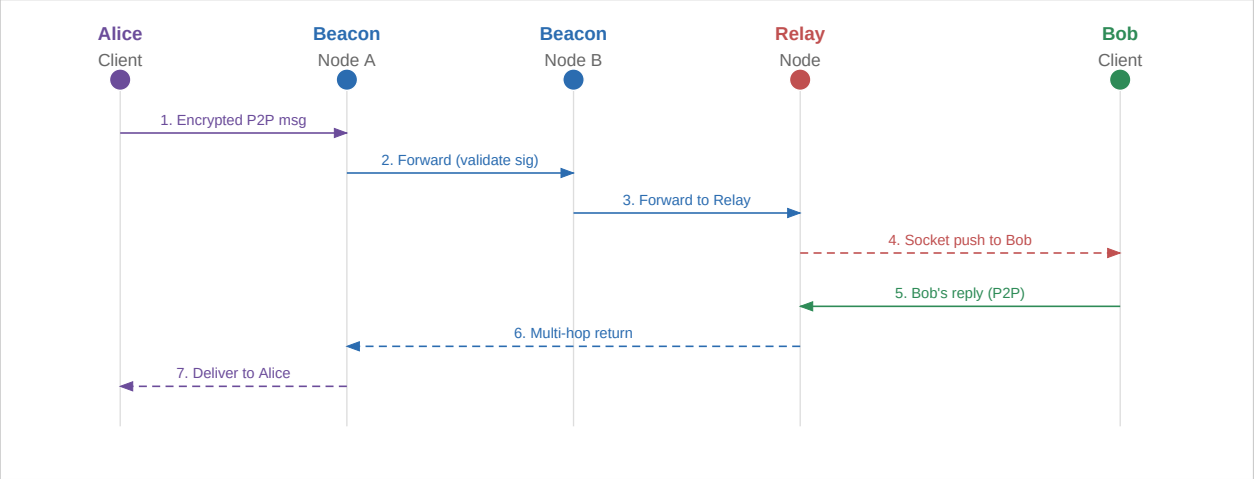

Figure 6. Message Delivery Sequence. Alice's encrypted message traverses Beacon Nodes via P2P gossip (steps 1–3), reaches the Relay Node which pushes to Bob via Socket (step 4). Bob's reply follows the reverse asymmetric path (steps 5–7).

5.3 Attachment Transmission via the Shredder Protocol

Attachment transmission follows the Shredder Protocol described in Section 4.3 and illustrated in Figure 3. The sender fragments the file, encrypts each fragment independently, transmits fragments through separate P2P routes, and sends encrypted reassembly metadata. The receiver collects all fragments, decrypts, and reassembles the original file.

For real-time media such as voice messages, the Shredder Protocol operates in streaming mode: audio data is continuously fragmented and transmitted in small chunks, with the receiver reassembling and playing fragments as they arrive. This introduces minimal additional latency while maintaining the full fragment-isolation privacy guarantees.

Fragment Size and Distribution Strategy

The optimal fragment size is determined by balancing two competing concerns: smaller fragments increase privacy (more routes, harder to collect all pieces) but increase overhead (more headers, more network transactions). MarsCat uses adaptive fragment sizing: for files under 1 MiB, fragments are 64 KiB; for files between 1–10 MiB, fragments are 256 KiB; for larger files, fragments are 1 MiB. Each fragment carries approximately 40 bytes of overhead for its unique identifier, sequence number, and per-fragment encryption header.

Fragment routing is intentionally randomized: the sender introduces random delays between fragment transmissions (50–200ms jitter) and each fragment is submitted to a different entry point in the P2P network. This temporal and spatial separation makes it extremely difficult for a network observer to correlate fragments belonging to the same attachment even through traffic analysis.

Integrity Verification and Error Recovery

Each fragment carries a SHA-256 hash for integrity verification. The reassembly metadata message contains the hash manifest—an ordered list of all fragment hashes—enabling the receiver to verify that each fragment was received intact and in the correct order. If a fragment fails integrity verification or is missing after a configurable timeout (default: 30 seconds per fragment), the receiver requests retransmission from the sender through a secure feedback channel.

5.4 Group Messaging

Group messaging in MarsCat extends the per-friendship key model to a multi-party setting while preserving the core privacy invariants. The design must balance four requirements: (i) only current members can read group messages; (ii) a departing member cannot read future messages (forward secrecy); (iii) a joining member cannot read past messages (backward secrecy); (iv) group membership should not leak information about members' bilateral friendships.

When a group is created, the group creator generates a shared group key and distributes it to all initial members using their individual friendship keys. Each member's identity within the group is represented by a group-specific derived key—not their friendship key with any individual member. This ensures that group participation does not leak information about bilateral friendships.

Group key rotation occurs when members join or leave, using a ratcheting mechanism inspired by the Signal Protocol's Double Ratchet Algorithm [3] but adapted for the multi-party setting. Forward secrecy is maintained: a departed member cannot decrypt messages sent after their departure, and a newly joined member cannot decrypt messages sent before their arrival.

Group Key Distribution Protocol

The group key distribution follows a star topology: the group administrator generates a new group key GK_{epoch} for each membership change epoch, encrypts it individually for each current member using their group-specific public keys, and broadcasts the encrypted key bundle to the group. Each member decrypts only their portion of the bundle to obtain the new group key. The use of per-member encryption ensures that no group message reveals the full membership list to any individual member—each member knows only their own group-specific key and the shared group key, not the keys of other members.

For large groups (>50 members), MarsCat employs a tree-based key distribution scheme to reduce the computational overhead of per-member encryption. Members are organized into a balanced binary tree, and key updates propagate through the tree structure, reducing the re-keying cost from O(n) to O(log n) per membership change [3].

6. MarsApp: Decentralized Application Ecosystem

MarsCat extends beyond social messaging to provide a comprehensive decentralized application platform through MarsApp. Built on the RelayX protocol's Self-Service mechanism [1], MarsApp enables developers to deploy and distribute applications entirely within the P2P network.

6.1 Application Deployment Model

MarsApp employs a fundamentally different deployment model from traditional web or mobile applications. Rather than hosting application code on centralized servers, developers package their applications and publish them to the MarsCat P2P network, indexed by a BTC address. The MarsCat client discovers, downloads, verifies, and executes MarsApps in a sandboxed runtime environment.

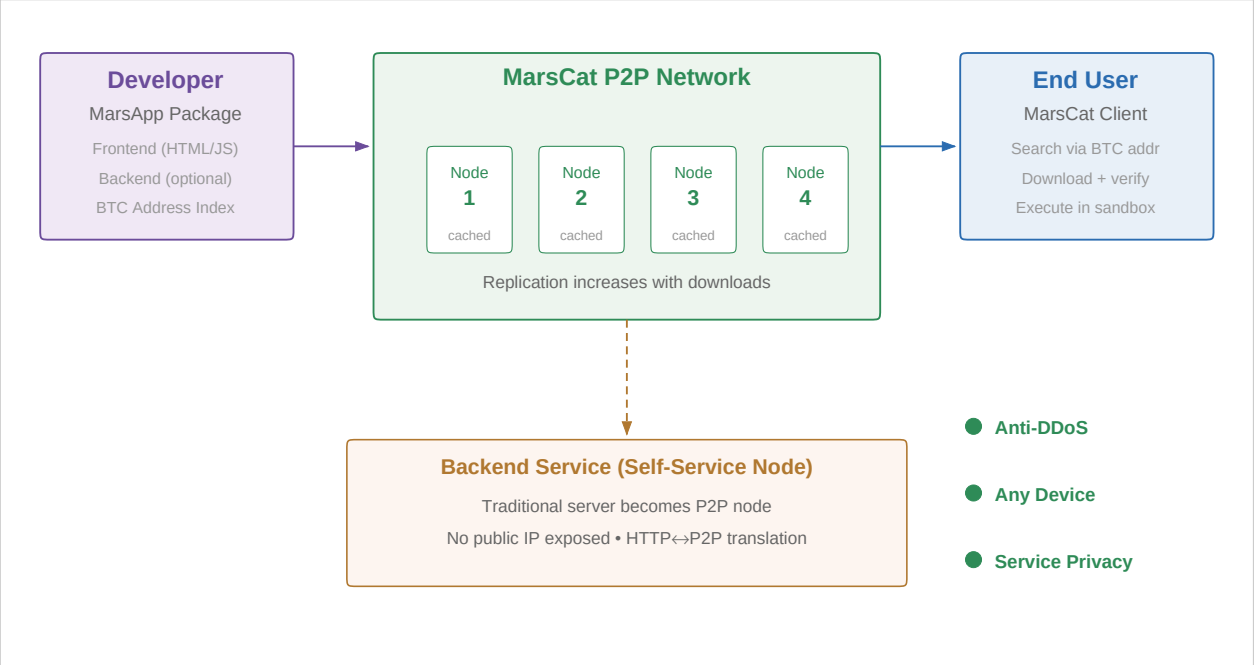

Figure 7. MarsApp Deployment Architecture. Developers publish applications to the P2P network indexed by BTC addresses. Frontend code is replicated across nodes. Backend services operate as Self-Service Nodes with no public IP exposure.

6.2 Frontend Distribution and Replication

MarsApp frontend code (HTML, JavaScript, CSS, assets) is distributed through the MarsCat P2P network using a progressive replication model. When a user downloads a MarsApp, their node caches a complete copy of the application package. Subsequent users who request the same application may download from any node that holds a cached copy, distributing bandwidth load across the network.

As the number of users downloading a MarsApp increases, the number of cached copies grows proportionally, creating a natural content distribution network (CDN) effect without any centralized infrastructure. Popular applications become highly available and fast to download, while rarely-used applications gradually age out of node caches based on configurable retention policies.

Application integrity is ensured through a multi-step verification process: (1) the MarsApp package includes a manifest file listing the SHA-256 hashes of all constituent files; (2) the manifest is signed by the developer's BTC address private key; (3) the MarsCat client verifies the manifest signature against the developer's known public key before executing the application; (4) individual file hashes are verified against the manifest during installation. This chain of verification ensures that no node in the P2P network can tamper with application code during distribution.

Version management follows a content-addressed model: each version of a MarsApp is identified by the hash of its package, and the developer publishes update notifications through the P2P network. Users can choose to auto-update or pin specific versions, providing control over their application environment while enabling efficient distribution of security patches.

6.3 Backend Service Integration via Self-Service Nodes

For MarsApps that require backend logic (database queries, computation, third-party API integration), the developer deploys a Self-Service Node [1] that bridges the P2P network and their backend application server. The Self-Service Node translates P2P messages into standard HTTP requests and vice versa, enabling the backend to operate as a standard web service while being accessible only through the MarsCat network.

This architecture provides three critical advantages: (1) **Anti-DDoS:** The backend server has no public IP address, making it unreachable by conventional DDoS attacks; (2) **Location Independence:** The backend can operate on any device in any network location—a home computer, a mobile phone, a cloud instance—without requiring static IP addresses or domain names; (3) **Service Privacy:** The end user and the backend server never establish a direct connection; all communication is mediated through the P2P network, preventing the server from learning the user's IP address and preventing network observers from linking users to specific services.

6.4 BTC Address-Based Application Indexing

MarsApp applications are indexed using BTC addresses rather than traditional domain names. Each application is registered to a unique Bitcoin address controlled by the developer. Users discover applications by searching for or scanning the BTC address, and the MarsCat client resolves the application location through the P2P network without DNS queries [14].

This indexing model provides inherent censorship resistance (no DNS server to seize), DDoS immunity (no centralized resolution service to overwhelm), cost efficiency (no domain registration or renewal), and cryptographic developer authentication (the BTC address serves as verifiable proof of developer identity). Application integrity is verified through signatures anchored to the developer's BTC address keypair.

MarsApp Advantages Summary

The MarsApp deployment model provides four structural advantages over traditional application hosting: **Natural DDoS Resistance**—with no centralized server or DNS entry, there is no target for volumetric attacks; application availability increases with user count as more cached copies proliferate across the network. **Universal Deployment**—backend services can operate on any device in any network, from a home Raspberry Pi to a mobile phone to a cloud instance, without requiring static IP addresses, domain names, or port forwarding. **Service Privacy**—the end user and backend server never establish direct connections; all communication is mediated through the P2P network, preventing IP address leakage in both directions. **Zero Infrastructure Cost**—the P2P network provides content distribution, service routing, and DDoS protection without any dedicated infrastructure investment.

Property	Traditional Web	IPFS	MarsApp
Frontend hosting	CDN/Cloud	P2P (static)	P2P (replicated)
Backend services	Cloud servers	N/A	Self-Service Nodes
DDoS resistance	Requires WAF	Moderate	Inherent
Service IP hidden	No	N/A	Yes
Domain required	Yes	Optional (DNSLink)	No (BTC addr)
Censorship resistant	No	Partial	Yes
Infrastructure cost	High	Low	Near zero

Table 3. MarsApp vs Traditional Deployment Models

6.5. Technical Specifications and Performance Characteristics

This section summarizes the key technical parameters of the MarsCat platform. Table 4 provides a comprehensive specification reference.

Category	Parameter	Specification
Encryption	Text cipher	Blowfish (variable key, 64-bit block)
	Key agreement	ECDH (secp256k1)
	Authentication	ECDSA (secp256k1)
	Fragment cipher	AES-256-GCM per fragment
	Key derivation	HKDF-SHA256
Identity	Master key	secp256k1 keypair
	Friendship keys	HD derived (BIP-32)
	Key storage	Device secure enclave
Network	Protocol base	RelayX v1.0 [1]
	Message latency	1–3 seconds E2E
	Max message size	10 MiB
	Max attachment	Unlimited (fragmented)
	Protocol support	IPv4 + IPv6 dual-stack
Shredder	Default fragment size	64 KiB – 1 MiB (adaptive)
	Inter-fragment jitter	50–200 ms random
	Integrity check	SHA-256 per fragment
MarsApp	App indexing	BTC address (34 chars)
	SDKs	JavaScript, Go, Rust
	Node deployment	One-click Docker

Table 4. MarsCat Complete Technical Specifications

7. Security Analysis

7.1 Threat Model

We consider a powerful adversary with the following capabilities: (i) ability to operate an arbitrary number of network nodes (Sybil capability); (ii) ability to observe network traffic on any subset of links (passive monitoring); (iii) ability to perform active attacks including message injection, modification, and replay; (iv) ability to compromise a limited number of user devices; (v) legal authority to compel disclosure from any centralized entity (subpoena capability).

7.2 Cryptographic Security

Text Message Security. Blowfish encryption with ECDH-derived session keys provides semantic security under chosen-plaintext attacks. The infinite-solution property provides additional resistance to brute-force attacks beyond the computational infeasibility of the key space. ECDSA signatures provide existential unforgeability under chosen-message attacks [14].

Attachment Security. The Shredder Protocol provides information-theoretic security against partial interception: an adversary holding fewer than all N fragments of an attachment gains zero information about the original file content, analogous to an (N,N)-threshold secret sharing scheme [11]. Each fragment is additionally encrypted, providing defense-in-depth.

Key Security. The HD key derivation scheme ensures that compromise of any friendship key does not reveal the master key or other friendship keys, providing key isolation. Forward secrecy for individual sessions is achieved through ephemeral ECDH key agreement.

7.3 Network-Level Security

The RelayX protocol's multi-hop gossip routing provides sender anonymity: an adversary monitoring a subset of network links cannot determine the originating node of a message [15]. The certificate-tiered routing priority system (2-2-1 strategy) combined with the health score mechanism provides structural Sybil resistance [7]. The asymmetric channel architecture separates uplink and downlink paths, complicating traffic correlation attacks.

7.4 Metadata Resistance

MarsCat provides comprehensive metadata resistance through the combination of: (i) identity privacy eliminating user-level metadata; (ii) per-friendship keys preventing contact list enumeration; (iii) multi-hop routing preventing IP-to-identity correlation; (iv) the Shredder Protocol preventing file-level traffic analysis. Against our threat model's subpoena capability, there is no centralized entity holding any user metadata to disclose.

Resistance to Specific Attack Vectors

Global Passive Adversary. An adversary monitoring all network links can observe encrypted message flows but cannot decrypt content (ECIES/Blowfish), identify senders (multi-hop gossip), link messages to friendships (per-friendship keys), or reconstruct attachments (Shredder Protocol). The cost of maintaining global surveillance of a P2P network grows linearly with network size, making this attack economically infeasible at scale [15].

Sybil Attack. An adversary operating many nodes can attempt to surround target users and intercept their traffic. MarsCat mitigates this through RelayX's certificate-tiered routing (2-2-1 strategy), health score mechanism, and the inherent redundancy of gossip propagation—messages propagate through multiple independent paths, so a Sybil attacker must control a substantial fraction of the network to achieve reliable interception [7].

Device Compromise. If an adversary gains access to a user's device, they can access the user's master key and all friendship keys. MarsCat mitigates this risk through device-level encryption (secure enclave storage), optional passphrase protection, and the per-friendship key design—if Alice's device is compromised, the adversary learns Alice's keys but gains no information about the keys held by Alice's contacts. The compromise is contained to one node and cannot propagate through the social graph.

Quantum Computing Threats. MarsCat's current cryptographic primitives (ECDSA, ECDH) are vulnerable to quantum attacks via Shor's algorithm. However, Blowfish as a symmetric cipher remains resistant to known quantum attacks (Grover's algorithm provides only a quadratic speedup, which is addressable by doubling key length). The roadmap includes migration to post-quantum cryptographic schemes (CRYSTALS-Kyber, CRYSTALS-Dilithium) as NIST standardization progresses [18].

Privacy Guarantees Summary

Privacy Property	Mechanism	Adversary Capability Defeated
Content confidentiality	Blowfish + ECIES	Passive interception, brute-force
Sender anonymity	Multi-hop P2P gossip	Traffic analysis, IP correlation
Identity privacy	Zero-PII, key-only identity	Database breach, subpoena
Relationship privacy	Per-friendship isolated keys	Social graph reconstruction
Attachment privacy	Shredder Protocol	Partial interception, traffic correlation
Forward secrecy	Ephemeral ECDH + ratchet	Post-compromise decryption
Group privacy	Per-group derived keys	Cross-group membership inference

Table 5. Privacy Guarantees and Corresponding Mechanisms

8. Comparison with Existing Platforms

Table 6 compares MarsCat's privacy capabilities with major existing platforms across the three privacy dimensions.

Feature	Signal	Telegram	Session	Briar	MarsCat
No phone/email required	✗	✗	✓	✓	✓
E2E encryption (default)	✓	✗	✓	✓	✓
Per-friendship keys	✗	✗	✗	✗	✓
Social graph protection	✗	✗	Partial	Partial	✓
Attachment shredding	✗	✗	✗	✗	✓
Infinite-solution cipher	✗	✗	✗	✗	✓
Decentralized app platform	✗	Partial	✗	✗	✓
No centralized servers	✗	✗	Partial	✓	✓

Table 6. Privacy Feature Comparison Across Platforms

As Table 6 demonstrates, MarsCat is the only platform that provides comprehensive protection across all three privacy dimensions simultaneously. While Session [16] and Briar [17] offer meaningful decentralization and identity privacy, neither implements per-friendship key isolation or attachment shredding. Signal [3] provides strong content encryption but requires phone number registration and operates through centralized servers, leaving identity and metadata fully exposed.

Detailed Platform Analysis

Signal [3] represents the current gold standard for encrypted messaging and introduced many of the cryptographic innovations (Double Ratchet, X3DH) that have been widely adopted. However, Signal's architecture fundamentally relies on centralized servers for message routing, key distribution, and contact discovery. Users must register with phone numbers, and Signal's servers necessarily have access to metadata including who communicates with whom, at what times, and how frequently. Signal's social graph protection is minimal.

Telegram provides convenience and reach but does not enable end-to-end encryption by default (only in "Secret Chats"), uses a non-standard MTProto encryption protocol that has received criticism from the cryptographic community, and operates through heavily centralized infrastructure. Telegram requires phone number registration and retains extensive metadata.

Session [16], developed by the Oxen Privacy Tech Foundation, makes meaningful progress toward decentralization by routing messages through a network of incentivized service nodes. Session does not require phone numbers and provides reasonable identity privacy. However, Session uses a single long-lived public key for each user, meaning all contacts see the same identity—enabling social graph reconstruction if the key is observed in multiple contexts.

Briar [17] is a fully peer-to-peer messenger that can operate over Tor, Wi-Fi, and Bluetooth, making it highly resilient to infrastructure censorship. Briar does not require phone numbers and provides strong decentralization. However, like Session, Briar uses a single identity per user, and it lacks any mechanism for attachment privacy beyond standard encryption. Briar also lacks an application ecosystem comparable to MarsApp.

MarsCat builds upon the strengths of these predecessors while addressing their remaining privacy gaps. The per-friendship key isolation system is, to our knowledge, unique among deployed messaging systems. The Shredder Protocol for attachment privacy and the infinite-solution property of Blowfish encryption provide defense-in-depth layers that go beyond what any single encryption algorithm can offer. The MarsApp ecosystem extends privacy protection to the application layer, providing a comprehensive platform rather than just a messaging tool.

9. Use Cases and Application Scenarios

MarsCat's three-dimensional privacy model enables a range of use cases that are impractical or impossible on conventional platforms. This section describes representative scenarios.

9.1 Journalism and Source Protection

Investigative journalists operating in hostile environments require communication channels that protect both the journalist and their sources. MarsCat's identity privacy ensures that neither party needs to reveal real-world identity. The per-friendship key system prevents an adversary who identifies the journalist from discovering their source network—each source communicates with a unique, isolated identity. The Shredder Protocol protects document transfers, and the MarsApp framework enables journalists to deploy secure document submission portals without centralized infrastructure that could be seized or blocked.

9.2 Medical and Health Communication

Patients communicating with healthcare providers require strong data privacy for medical records, test results, and treatment discussions. MarsCat's Blowfish encryption with infinite-solution properties ensures that intercepted messages cannot be decrypted through brute force. The Shredder Protocol protects medical imaging and lab results during transmission. The absence of metadata collection means that even the fact of a medical consultation remains private—no third party can determine that a user is communicating with a healthcare provider.

9.3 Corporate Confidential Communications

Organizations handling sensitive business communications—merger negotiations, intellectual property discussions, board deliberations—face risks from both external adversaries and insider threats. MarsCat's per-friendship key isolation ensures that corporate communication channels are compartmentalized: compromise of one channel does not expose others. MarsApp enables deployment of internal tools (document management, project coordination) without exposing backend services to the public internet, eliminating DDoS and unauthorized access vectors.

9.4 Human Rights and Civil Liberties

Activists, human rights workers, and civil society organizations in authoritarian contexts require communication tools that cannot be controlled by state authorities. MarsCat's fully decentralized architecture—no servers, no DNS, no corporate entity—makes censorship technically infeasible. The P2P network operates across jurisdictions without any single point of legal or technical control. MarsApps enable civil society organizations to operate secure information portals, anonymous reporting systems, and coordination tools entirely within the P2P network.

9.5 Decentralized Commerce

MarsApp enables merchants to deploy e-commerce applications with inherent privacy protection and DDoS resistance. Customer communications, order processing, and payment coordination can all occur through the P2P network without exposing the merchant's infrastructure to direct internet traffic. The BTC address indexing system provides natural integration with cryptocurrency payment systems, enabling end-to-end private commercial transactions.

9.6 Education and Research

Academic researchers studying sensitive topics—political dissent, substance abuse, sexual health, whistleblowing behavior—require secure communication channels with research participants. MarsCat's zero-PII identity model eliminates the need for complex data protection protocols typically required when handling participant identity data. The per-friendship key system ensures that even if one participant's communication is compromised, other

participants' identities and communications remain protected. MarsApp enables researchers to deploy custom data collection tools within the P2P network, ensuring that research data is never exposed to centralized servers that could be breached or subpoenaed.

9.7 Disaster and Emergency Communication

During natural disasters, conflicts, or infrastructure failures, centralized communication systems frequently become unavailable due to server overload, physical damage, or deliberate shutdown by authorities. MarsCat's fully decentralized P2P architecture continues to operate as long as any subset of nodes can reach each other, making it inherently resilient to infrastructure disruptions. Client Nodes on mobile devices can form local mesh networks, and the system's tolerance for intermittent connectivity (messages are cached at Relay Nodes for up to 24 hours) accommodates the unreliable network conditions common in emergency scenarios. MarsApps can provide emergency-specific functionality: location sharing, resource coordination, and status reporting—all operating within the privacy-preserving P2P network without dependence on any centralized infrastructure.

Use Case Summary

Use Case	Key Privacy Dimension	Critical Feature
Journalism	Identity + Relationship	Per-friendship keys prevent source identification
Healthcare	Data + Identity	Blowfish infinite-solution for medical records
Corporate	Data + Relationship	Compartmentalized channels via key isolation
Human Rights	All three dimensions	Full decentralization + censorship resistance
Commerce	Data + Infrastructure	MarsApp DDoS immunity + payment privacy
Research	Identity + Data	Zero-PII compliance + secure data collection
Emergency	Infrastructure resilience	Decentralized P2P + intermittent connectivity

Table 7. Use Case Summary and Key Privacy Features

10. Roadmap and Future Directions

MarsCat's development roadmap encompasses several key directions across cryptographic, network, and ecosystem dimensions:

10.1 Cryptographic Evolution

Post-Quantum Migration. As NIST finalizes post-quantum cryptographic standards, MarsCat will implement migration paths for CRYSTALS-Kyber (key encapsulation) and CRYSTALS-Dilithium (signatures) [18]. The migration strategy involves dual-signing during the transition period: messages will carry both classical ECDSA and post-quantum signatures, ensuring backward compatibility while building quantum resistance. Blowfish, as a symmetric cipher, requires only key-length doubling to resist Grover's algorithm.

Advanced Shredder Protocol. Future versions of the Shredder Protocol will incorporate erasure coding (Reed-Solomon) to enable file reconstruction from a threshold subset of fragments (e.g., any 7 of 10), improving reliability over lossy P2P connections while maintaining the privacy guarantees of fragment isolation.

10.2 Network and Ecosystem Development

Incentive Layer. Introduction of token-based incentive mechanisms to sustain long-term node participation and reward contributors who provide routing, caching, and relay services. The incentive design must balance rewarding participation without creating metadata that could compromise privacy—a challenge we plan to address through privacy-preserving reward protocols based on blind signatures.

Zero-Knowledge Proofs. Integration of ZK-proof based verification for group membership management and MarsApp authentication [19]. ZK proofs enable users to prove properties about themselves (e.g., group

membership, age threshold, geographic region) without revealing their identity or the specific property value.

Cross-Chain MarsApp Indexing. Expansion of the BTC address indexing system to support multi-chain application discovery, enabling developers to register MarsApps on Ethereum, Solana, or other blockchain networks while maintaining the same privacy and censorship-resistance properties.

10.3 Platform Expansion

Voice and Video Calls. Extension of the Shredder Protocol to support real-time voice and video communication through fragment-based streaming, providing the same attachment privacy guarantees for live media as for stored files.

Formal Verification. Rigorous formal analysis of the Shredder Protocol's security properties and the per-friendship key isolation guarantees using ProVerif and Tamarin computational models, producing machine-checkable proofs of the stated security properties.

Cross-Platform Availability. Native client development for iOS, Android, Windows, macOS, and Linux, with feature parity across all platforms and a shared core cryptographic library implemented in Rust for memory safety and performance.

11. Conclusion

This whitepaper has presented MarsCat, a privacy-first social platform that addresses the comprehensive privacy failures of contemporary communication systems through a novel three-dimensional privacy model. By simultaneously protecting user identity (zero real-world information), message content (Blowfish encryption with infinite-solution properties and the Shredder Protocol for attachments), and social relationships (per-friendship isolated key identities), MarsCat achieves a level of privacy protection that is fundamentally stronger than any existing platform.

The key technical contributions of MarsCat include: (1) the **per-friendship isolated key identity** system, which prevents social graph reconstruction by ensuring that each friendship operates through a unique, unlinkable cryptographic identity; (2) the **Blowfish infinite-solution encryption** model, which provides both computational and information-theoretic resistance to brute-force attacks; (3) the **Shredder Protocol**, which fragments and independently encrypts attachments across multiple P2P routes, providing defense-in-depth for file privacy; (4) the **MarsApp ecosystem**, which extends privacy protection to the application layer through decentralized deployment with inherent DDoS resistance and service privacy.

Built upon the RelayX peer-to-peer protocol [1], MarsCat operates without centralized infrastructure, eliminating the trust assumptions, single points of failure, and subpoena vulnerabilities inherent in server-based architectures. The combination of P2P networking, multi-dimensional privacy, and decentralized application deployment creates a comprehensive platform that addresses the full spectrum of privacy threats facing modern digital communication.

Looking ahead, MarsCat's roadmap includes post-quantum cryptographic migration, zero-knowledge proof integration, voice/video call support through Shredder Protocol streaming, and formal verification of security properties. These developments will further strengthen the platform's privacy guarantees while expanding its functional capabilities.

MarsCat represents a commitment to the principle that private communication is a fundamental human right—one that should be guaranteed by mathematics and protocol design, not by corporate policy or government restraint. In the MarsCat network, your identity is yours, your data is yours, your relationships are yours, and your freedom is yours.

References

[1] XProtocol Lab, “RelayX: A Decentralized Peer-to-Peer Application Deployment and Distribution Protocol for Privacy-Preserving Web3 Infrastructure,” 2025.

[2] S. Zuboff, *The Age of Surveillance Capitalism*. PublicAffairs, 2019.

[3] M. Marlinspike and T. Perrin, “The X3DH Key Agreement Protocol,” Signal Foundation, 2016.

[4] J. Mayer, P. Mutchler, and J. C. Mitchell, “Evaluating the Privacy Properties of Telephone Metadata,” in *Proc. National Academy of Sciences*, vol. 113, no. 20, pp. 5536–5541, 2016.

[5] B. Schneier, *Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World*. W.W. Norton, 2015.

[6] H. Abelson et al., “Keys Under Doormats: Mandating Insecurity by Requiring Government Access to All Data and Communications,” *J. Cybersecurity*, vol. 1, no. 1, pp. 69–79, 2015.

[7] J. R. Douceur, “The Sybil Attack,” in *Proc. 1st Int. Workshop Peer-to-Peer Syst. (IPTPS)*, 2002, pp. 251–260.

[8] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[9] European Parliament, “General Data Protection Regulation (GDPR),” Regulation (EU) 2016/679, 2016.

[10] B. Schneier, “Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish),” in *Proc. Fast Software Encryption (FSE)*, 1993, pp. 191–204.

[11] A. Shamir, “How to Share a Secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[12] M. O. Rabin, “Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance,” *J. ACM*, vol. 36, no. 2, pp. 335–348, 1989.

[13] P. Wuille, “BIP-32: Hierarchical Deterministic Wallets,” Bitcoin Improvement Proposal, 2012.

[14] D. Johnson, A. Menezes, and S. Vanstone, “The Elliptic Curve Digital Signature Algorithm (ECDSA),” *Int. J. Inf. Security*, vol. 1, no. 1, pp. 36–63, 2001.

[15] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The Second-Generation Onion Router,” in *Proc. 13th USENIX Security Symp.*, 2004, pp. 303–320.

[16] Oxen Privacy Tech Foundation, “Session: A Decentralized Messenger,” Tech. Rep., 2020.

[17] M. Briar, “Briar: Resilient P2P Messaging for Everyone,” Tech. Rep., 2017.

[18] R. Avanzi et al., “CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation,” NIST PQC Submission, 2020.

[19] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, “SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge,” in *Proc. CRYPTO*, 2013, pp. 90–108.