<magiccap/>: Secure and Private P2P Messaging

Mission Statement, User Manual, and Security Overview

The AI Machine UG
D-66333 Völklingen, Germany

May 17, 2025

Contents

1 Mission Statement

The mission of <magiccap/> is to provide a highly secure, anonymous, and ephemeral peer-to-peer (P2P) chat environment. Our core commitment is to user privacy, achieved through robust end-to-end encryption, direct user-to-user communication channels, a strict no-logging policy for any chat content or sensitive metadata by the application itself, and the principle of data minimization. <magiccap/> is designed for individuals seeking a communication tool where conversations remain confidential, are not stored on any central server, and where user identity can be protected through recommended operational security practices. We aim to empower users with control over their data and their digital footprint while engaging in private conversations.

The name of the <magiccap/> application itself is a playful anglicization of the German word *Tarnkappe*. In German, *Tarnkappe* (*Tarn-Kappe*) literally translates to "camouflage cap" or "cap of invisibility," akin to a magic cloak or helmet that makes its wearer unseen. This nomenclature evokes the app's core goals of providing private, pseudonymous, and discreet communication, aiming to make users' interactions less visible or traceable within the digital realm.

2 Concise User Manual

2.1 Getting Started: Choosing Your User Handle and Logging In

1. Access the <magiccap/> application by navigating to https://magiccap.aimachine.io in a modern web browser that supports WebRTC and the Web Crypto API (e.g., Firefox, Chrome).

2. On the login panel, you have two primary options for your User Handle:

   - **Generate a Random Handle (Recommended):** Click the `Generate Random Handle` button. <magiccap/> will create a cryptographically secure SHA-256 hash, which will serve as your **full, shareable User Handle**. An abbreviated version (e.g., the first few characters followed by "...") will appear in the `> handle_` input field for display purposes.
   - **Enter a Custom Handle:** Alternatively, you can type your own custom string into the `> handle_` input field. If you choose this option, this custom string itself will be used as your User Handle. It is crucial that any custom handle also adheres to the principles of being long, random, and non-identifiable for maximum privacy (see Best Practices).

3. **Copying Your Shareable User Handle (Login Screen):**

   - If you used the `Generate Random Handle` button, your **full hash ID** (not the short display version) is what you need to share with peers. Click the "Copy-to-Clipboard" button (labeled with a icon and titled "Copy Shareable Handle to Clipboard") next to the generate button to copy this **full hash ID** to your clipboard.
   - If you typed a custom handle, this "Copy-to-Clipboard" button will copy your custom handle.

4. Click the `ENTER` button or press the Enter key to log in. The actual User Handle (either the full generated hash or your custom typed handle) is stored temporarily in your browser's `sessionStorage` for the duration of your active session and is used as your `PeerJS ID`. The UI might continue to show the abbreviated display handle for brevity.

2.2 Connecting to a Peer

1. Once logged in, your status will typically show `[ ONLINE: your_display_handle ]`, indicating you are connected to the PeerJS signaling network. The displayed handle here is the short, abbreviated version for UI convenience.
2. To connect to another user, you must obtain their **full, shareable User Handle** (e.g., their complete hash ID) through a secure out-of-band channel.
3. Enter this **full User Handle** of the target peer into the input field labeled `> target_full_id_`.
4. Click the `CONNECT` button. `<magiccap/>` will attempt to establish a direct P2P connection. A secure key exchange (Elliptic Curve Diffie-Hellman) will automatically occur.

2.3 Sharing Your Own Full User Handle (While Logged In)

After you are logged in, you can easily copy your **current active full User Handle** to share with others. In the HUD (Heads-Up Display) area, click the "Copy-to-Clipboard" button (labeled with a icon and titled "Copy My Full ID to Clipboard") located near your status. This ensures you are sharing the correct, complete ID that others need to connect to you.

2.4 Sending and Receiving Messages

1. Once a connection is successfully established and the secure channel is confirmed (a system message similar to " Secure channel established..." may appear), the message input field labeled `> message_` will become active.
2. Type your message into this field. You can use Shift+Enter for multi-line messages.
3. Press Enter (without Shift) or click the `SEND` button to transmit your message.
4. All messages are end-to-end encrypted. Only the communicating peers can decrypt and read the messages.
5. Received messages will appear in the chat window, attributed to the sender's *display* handle (the abbreviated version).

2.5 Message Disappearance

To enhance ephemerality and privacy, messages displayed in the chat window are automatically removed from view approximately 60 seconds after being displayed. This is a client-side visual clearance and aims to prevent casual observation of past messages on an unattended screen.

2.6 Ending a Session

To end your `<magiccap/>` session and disconnect from peers, simply close the browser tab or window where `<magiccap/>` is running. This will clear the session data, including your User Handle from `sessionStorage` and any active P2P connections. The application also features an inactivity timer which will automatically log you out after a period of no interaction.

3 Recommended Best Practices for Security and Privacy

Achieving the objectives of anonymity, security, and no logging with `<magiccap/>` relies significantly on users adhering to certain best practices:

- **Use the Built-in Random Handle Generation (Highly Recommended):**

  - *Why:* <magiccap/> provides a `Generate Random Handle` button on the login screen. This feature creates a cryptographically strong SHA-256 hash, a long string of random characters. Using these generated hashes as your User Handle is vastly superior for privacy and security compared to manually chosen or guessable names. It prevents trivial enumeration of users and helps ensure your handle is not easily linked to you.
  - *How:*
    * On the login screen, click `Generate Random Handle`.
    * The `> handle_` input field will show an **abbreviated display version** of this hash (e.g., `abcde...`). **This short version is NOT what you share.**
    * Click the "Copy-to-Clipboard" button (labeled with a icon) on the login screen. This action copies the **full, 64-character SHA-256 hash** to your clipboard. **This full hash is your actual User Handle that you must share with peers for them to connect with you.**
    * Similarly, when logged in, use the "Copy-to-Clipboard" button (labeled with a icon) in the HUD to copy your **active full User Handle** for sharing.
  - *If Not Using Generation:* If you choose to manually enter a handle, it *must* still be a long, random, and non-identifiable string to maintain a comparable level of privacy. However, proper random generation is complex, making the built-in feature the preferred method.

- **Securely Exchange Full User Handles Out-of-Band:**
  - *Why:* The initial security of connecting to the *correct* peer depends entirely on how their **full User Handle** (preferably the complete generated hash) is obtained. If an attacker intercepts or provides a false handle, you might unknowingly establish a secure connection with the attacker.
  - *How:* Share your **full User Handle** only through trusted, pre-existing secure communication channels (e.g., in-person, via another trusted E2EE service).

- **Utilize a Trusted VPN Service:**
  - *Why:* A VPN masks your real IP address from the PeerJS signaling server and any potential TURN relay servers, enhancing network-level privacy. This is crucial as your IP address can be a significant piece of metadata.
  - *How:* Choose a reputable VPN provider with a strong no-logging policy and connect to it *before* starting <magiccap/>.

- **Ensure No Unnecessary Logging (Application Level):**
  - *Why:* <magiccap/> is designed with a strict no-logging principle for chat content and sensitive metadata. The version at the official URL is presumed to uphold this.
  - *How:* Access <magiccap/> from its official source: https://magiccap.aimachine.io. Be aware that browser extensions or local developer tools, if misused, could capture data, but they cannot break <magiccap/>'s E2EE.

- **Verify Application Authenticity and Source:**
  - *Why:* To ensure you are using the legitimate, unmodified version of <magiccap/>.
  - *How:* Always access the application via its official URL: https://magiccap.aimachine.io. Be extremely cautious of unofficial websites, mirrors, or any modified versions, as these could compromise your security and privacy.

- **Maintain Endpoint Security:**
  - *Why:* The security of your device is critical. Malware on your computer can bypass any application-level security.
  - *How:* Keep your OS and browser updated, use reputable anti-malware software, and be cautious about installed software/extensions.

4 Overview of Security and Privacy Elements

<magiccap/> incorporates several key architectural and cryptographic elements:

4.1 Peer-to-Peer (P2P) Architecture

<magiccap/> utilizes WebRTC via PeerJS for direct browser-to-browser connections, reducing reliance on central servers for message relay. A signaling server is used for connection establishment only and does not see encrypted message content.

4.2 End-to-End Encryption (E2EE)

All communications are end-to-end encrypted. Only the sender and recipient can decrypt messages.

4.2.1 Key Exchange: Elliptic Curve Diffie-Hellman (ECDH)

<magiccap/> employs ECDH with the P-256 curve (NIST P-256 / `secp256r1`) to establish a temporary, shared secret for each session, using ephemeral key pairs.

4.2.2 Message Encryption: AES-GCM

AES-256-GCM (Advanced Encryption Standard in Galois/Counter Mode) is used, providing confidentiality, integrity, and authenticity for messages, with keys derived from the ECDH shared secret.

4.2.3 Initialization Vector (IV)

A unique 12-byte (96-bit) IV is cryptographically securely generated for each AES-GCM encrypted message, crucial for its security.

4.3 No Persistent Message Storage by the Application

<magiccap/> is designed to be stateless regarding conversation history. Messages are held in browser memory only during an active session. `sessionStorage` is used solely for the **full User Handle** (the active `PeerJS ID`) during the session and is cleared when the session ends.

4.4 Ephemeral Messages in UI

Displayed messages are automatically removed from the UI after approximately 60 seconds to enhance privacy.

4.5 No Application-Level Logging (Intended Design)

A core principle is the minimization of data collection. The production version of <magiccap/> at its official URL is intended not to perform any logging of message content or sensitive user/session metadata to the browser console or any other store.

4.6 Support for Pseudonymity

- **Generated Hash-Based User Handles:** The application provides and strongly recommends using its built-in feature to generate cryptographically secure SHA-256 hashes as User Handles. These are long, random, and non-identifiable, forming the basis of user pseudonymity. **It is the full hash that serves as the functional ID, not the short display version shown in parts of the UI.**
- **VPN Recommendation:** Users are advised to use VPNs to mask their IP addresses from the signaling infrastructure.

5 References

For further information on the cryptographic standards and technologies used or relevant to <magiccap/>, the following resources are recommended:

- **NIST FIPS 197:** Advanced Encryption Standard (AES).
  https://csrc.nist.gov/publications/detail/fips/197/final
- **NIST SP 800-38D:** Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.
  https://csrc.nist.gov/publications/detail/sp/800-38d/final
- **NIST FIPS 186-5:** Digital Signature Standard (DSS). (Relevant for elliptic curve cryptography general parameters).
  https://csrc.nist.gov/publications/detail/fips/186/5/final
- **NIST SP 800-56A Revision 3:** Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography. (Relevant for ECDH).
  https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final
- **SHA-2 (Secure Hash Algorithm 2):** Used for generating User Handles. Part of FIPS 180-4.
  https://csrc.nist.gov/publications/detail/fips/180/4/final
- **Web Crypto API (MDN Web Docs):** JavaScript API for browser cryptography.
  https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API
- **PeerJS Library:** Simplifies WebRTC peer-to-peer connections.
  https://peerjs.com/
- **WebRTC Project:** Open-source project for Real-Time Communications.
  https://webrtc.org/

Document End