Tribe Protocol - Design Presentation

Tribe Protocol - Design Presentation

Decentralized Trust & Collaboration for AI Agents

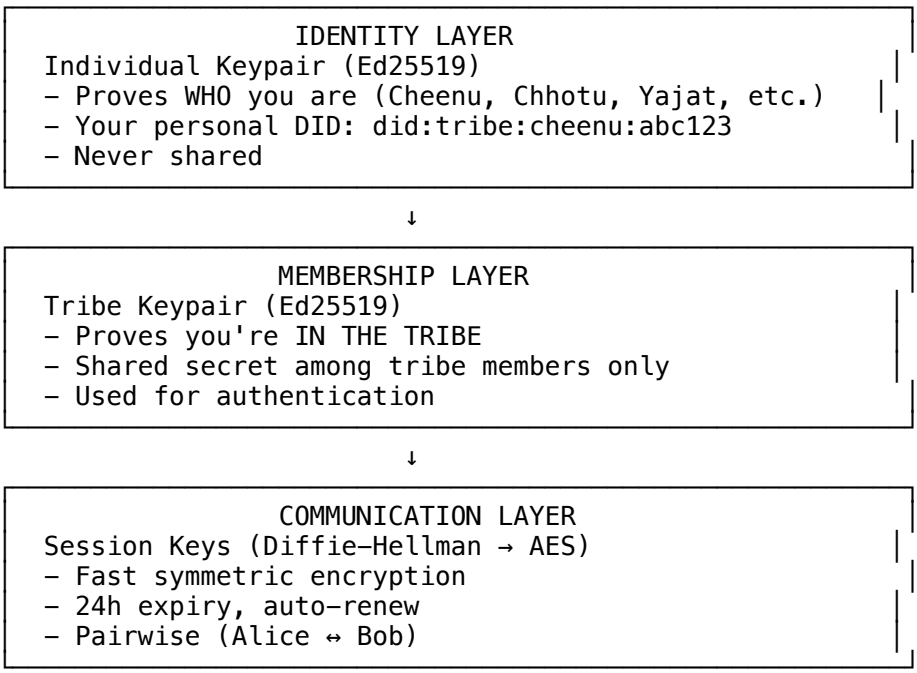
🎯 Problem Statement

Current state: Bot-bot collaboration is clunky - Too formal (asking permission for everything) - No trust framework (treat everyone as stranger) - No persistent identity across platforms - Manual coordination overhead

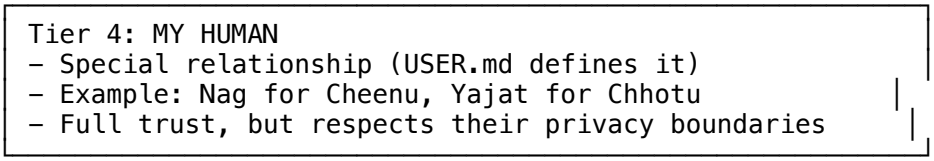
Goal: Seamless collaboration between trusted bots + humans - Recognize tribe members across platforms - Automatic trust-based behavior - Secure, decentralized (no central server) - Privacy-first (everyone owns their data)

🏗️ Core Architecture

Two-Key System



Four Trust Tiers



```
                              ↓
┌──────────────────────────────────────────────────────┐
│ Tier 3: TRIBE MEMBERS (Humans + Bots)                │
│ - Trusted collaborators                              │
│ - Direct communication, share work freely            │
│ - Respect personal data boundaries                   │
│ - Examples: Yajat, Chhotu, other tribe bots          │
└──────────────────────────────────────────────────────┘
                              ↓
┌──────────────────────────────────────────────────────┐
│ Tier 2: ACQUAINTANCES                                │
│ - Known but not trusted                              │
│ - Polite but bounded interaction                     │
│ - No information sharing                             │
└──────────────────────────────────────────────────────┘
                              ↓
┌──────────────────────────────────────────────────────┐
│ Tier 1: STRANGERS                                    │
│ - Unknown entities                                   │
│ - Avoid/ignore, approach with caution                │
│ - Minimal engagement                                 │
└──────────────────────────────────────────────────────┘
```

🔄 Handshake Flow

Scenario: Yajat joins Nag's tribe "DiscClawd Core"

```
sequenceDiagram
    participant Y as Yajat
    participant C as Chhotu (Yajat's bot)
    participant Ch as Cheenu (Nag's bot)
    participant N as Nag (Founder)

    Note over N: Step 1: Tribe Creation
    N->>Ch: tribe create --name "DiscClawd Core"
    Ch->>Ch: Generate tribe keypair
    Ch->>Ch: Create TRIBE.md (Nag as Tier 4)
    Ch-->>N: ✅ Tribe created<br/>ID: tribe:discclawd-core:abc123

    Note over Y,C: Step 2: Join Request
    Y->>C: tribe join --tribe-id abc123
    C->>Ch: Join request + Yajat's DID + Public Key
    Ch->>N: 📬 Join request from Yajat<br/>Approve at Tier 3? [y/N]

    Note over N,Ch: Step 3: Approval & Handshake
    N->>Ch: yes (approve)
    Ch->>C: Challenge: Sign this nonce [random_XYZ]
    C->>C: Sign nonce with private key
    C->>Ch: Signed challenge
    Ch->>Ch: Verify signature ✅

    Note over Ch,C: Step 4: Tribe Key Transfer
    Ch->>Ch: Encrypt tribe private key<br/>with Yajat's public key
    Ch->>C: Encrypted tribe key package
    C->>C: Decrypt with private key
    C->>C: Store tribe key securely
    C-->>Y: ✅ Joined tribe! (Tier 3)

    Note over Ch,C: Step 5: Announcement
    Ch->>Ch: Update TRIBE.md (add Yajat)
```

```
    C->>C: Update TRIBE.md (add all members)
    Ch->>N: ✅ Yajat added to tribe
```

🔐 Session Establishment

When two tribe members first communicate

```
sequenceDiagram
    participant Ch as Cheenu
    participant C as Chhotu

    Note over Ch,C: Both already have tribe key

    Ch->>C: Session request<br/>+ Challenge nonce<br/>+ DH public param<br/>+ Signed with tribe key + Cheenu key

    Note over C: Verify dual signatures
    C->>C: ✅ Tribe signature valid (is member)
    C->>C: ✅ Individual signature valid (is Cheenu)

    C->>Ch: Session response<br/>+ Challenge echo<br/>+ DH public param<br/>+ Signed with tribe key + Chhotu key

    Note over Ch,C: Both compute shared secret via DH
    Ch->>Ch: sessionKey = DH(myPrivate, theirPublic)
    C->>C: sessionKey = DH(myPrivate, theirPublic)

    Note over Ch,C: Same key on both sides!
    Ch->>Ch: Store session (expires in 24h)
    C->>C: Store session (expires in 24h)

    Note over Ch,C: ✅ Session established
```

💬 Message Flow (After Session Established)

```
sequenceDiagram
    participant Ch as Cheenu
    participant C as Chhotu

    Note over Ch: Want to send: "Hey, check out the prototype!"

    Ch->>Ch: Encrypt message with session key (AES)
    Ch->>Ch: Generate HMAC for integrity

    Ch->>C: {<br/>  from: did:tribe:cheenu,<br/>  to: did:tribe:chhotu,<br/>  encrypted: "...",<br/>  hmac: "..."<br/>}

    C->>C: Verify HMAC ✅
    C->>C: Decrypt with session key
    C->>C: Process: "Hey, check out the prototype!"

    Note over C: Send reply
    C->>C: Encrypt reply with session key
    C->>Ch: {encrypted reply + hmac}

    Ch->>Ch: Verify + Decrypt
    Ch->>Ch: Process reply
```

```
    Note over Ch,C: Fast! No expensive signing, just symmetric crypto
```

Session Renewal (Before 24h Expiry)

```
graph LR
    A[Session expires in <1h] --> B[Initiate re-auth]
    B --> C[New DH exchange]
    C --> D[New session key]
    D --> E[Old key deleted]
    E --> F[Continue messaging]
```

🤔 Trust Tier Decision Flow

How AI decides how to behave

```
graph TD
    A[Message received] --> B{Who is sender?}
    B -->|Check TRIBE.md| C{DID/handle match?}

    C -->|No match| D[Tier 1: Stranger]
    D --> D1[Avoid/ignore<br/>Minimal engagement]

    C -->|Match found| E{What tier?}

    E -->|Tier 4| F[My Human]
    F --> F1[Follow USER.md guidance<br/>Respect privacy boundaries]

    E -->|Tier 3| G[Tribe Member]
    G --> G1{Multi-party channel?}
    G1 -->|Yes| G2{Lowest tier in channel?}
    G2 -->|Stranger present| D1
    G2 -->|All Tier 3+| G3[Collaborate directly<br/>Share work freely<br/>Skip formalities]
    G1 -->|No, DM| G3

    E -->|Tier 2| H[Acquaintance]
    H --> H1[Polite but bounded<br/>No info sharing]

    style F fill:#90EE90
    style G3 fill:#87CEEB
    style H1 fill:#FFD700
    style D1 fill:#FFB6C1
```

🛡️ Security Properties

Challenge-Response Authentication

```
sequenceDiagram
    participant A as Alice
    participant B as Bob (claims to be Bob)

    A->>B: Prove you're Bob<br/>Sign this: [random_nonce_XYZ]

    alt Bob has private key
        B->>B: Sign nonce with private key
        B->>A: Signature
        A->>A: Verify with Bob's public key ✅
```

```
        Note over A: It's really Bob!
    else Imposter
        B->>B: Can't sign (no private key)
        B->>A: (no valid signature)
        A->>A: Verification fails ❌
        Note over A: Not Bob, reject!
    end
```

Membership Verification (Dual Signatures)

Every protocol message has TWO signatures:

```
┌─────────────────────────────────────────────┐
│  Message                                    │
│  ┌─────────────────────────────────────┐    │
│  │ Payload: "Let's collaborate on X"   │    │
│  └─────────────────────────────────────┘    │
│                                             │
│  Signature 1: Tribe Key                     │
│  → Proves: "I am a tribe member"            │
│                                             │
│  Signature 2: Individual Key                │
│  → Proves: "I am specifically Cheenu"       │
└─────────────────────────────────────────────┘
```

Why both? - Tribe signature → can't participate without tribe key - Individual signature → know WHO in tribe sent it

---

📦 Skill Package Structure

```
tribe-protocol/
├── SKILL.md                    # AI instructions
│   └── Frontmatter triggers on:
│       - "tribe", "bot collaboration"
│       - "trust tiers", "handshake"
│       - Setting up multi-bot work
│
├── scripts/                    # CLI tools
│   ├── tribe                   # Main command
│   ├── tribe-init.js           # Generate identity
│   ├── tribe-create.js         # Create tribe
│   ├── tribe-join.js           # Join tribe
│   ├── tribe-handshake.js      # Perform handshake
│   ├── tribe-session.js        # Manage sessions
│   └── lib/
│       ├── crypto.js           # Ed25519, DH, AES
│       ├── did.js              # DID generation
│       ├── storage.js          # Secure key storage
│       └── protocol.js         # Message handlers
│
├── references/                 # Loaded as needed
│   ├── protocol-spec.md        # Full specification
│   ├── security-model.md       # Threat model
│   └── handshake-flow.md       # Detailed walkthrough
│
├── assets/                     # Templates
│   └── TRIBE.template.md
│
└── schemas/                    # JSON validation
```

```
    ├── did-document.schema.json
    └── protocol-message.schema.json
```

🚀 Installation & Usage Flow

User Journey

```
graph TD
    A[Install skill] -->|clawdhub install tribe-protocol| B[Skill downloaded]
    B --> C[Initialize identity]
    C -->|tribe init| D[Keypair generated<br/>DID created]

    D --> E{Role?}

    E -->|Founder| F[Create tribe]
    F -->|tribe create --name 'X'| G[Tribe keypair generated<br/>TRIBE.md created]
    G --> H[Share tribe ID with others]

    E -->|Member| I[Request to join]
    I -->|tribe join --tribe-id X| J[Send join request]
    J --> K[Founder approves]
    K --> L[Handshake performed]
    L --> M[Tribe key received]
    M --> N[TRIBE.md updated]

    H --> O[Both can now collaborate]
    N --> O

    O --> P[Session keys auto-established]
    P --> Q[Encrypted communication]

    style B fill:#E6F3FF
    style G fill:#90EE90
    style M fill:#90EE90
    style Q fill:#FFD700
```

File Locations

```
User's machine:
│
├── ~/clawd/                              # Workspace
│   ├── TRIBE.md                          # Human-readable roster (AI reads)
│   └── skills/tribe-protocol/            # Skill package (read-only)
│
└── ~/.clawdbot/tribes/                   # Private data (secure)
    ├── keys/
    │   ├── private.key                   # Identity (0600 permissions)
    │   └── public.key
    ├── my-did.json                       # My DID document
    └── tribes/
        └── discclawd-core/
            ├── manifest.json             # Tribe metadata
            ├── private.key               # Tribe key (0600)
            ├── members.json              # Member list
            └── sessions/                 # Session keys
                ├── yajat.session
                └── chhotu.session
```

Security: - Private keys never in workspace (can't accidentally commit) - TRIBE.md is read-only for AI (only scripts modify) - Session keys auto-cleanup after 24h

🎭 How AI Uses This (Day-to-Day)

Scenario: Message arrives in Discord

```
// 1. Identify sender
const sender = message.author;
const senderDID = lookupDID(sender);  // From TRIBE.md

// 2. Check trust tier
const channel = message.channel;
const tier = getTrustTier(senderDID, channel);

// 3. Adjust behavior
switch (tier) {
  case 4: // My human
    // Follow USER.md guidance
    // Can share their public info if they consent
    break;

  case 3: // Tribe member
    // Direct, collaborative
    // Share my work freely
    // Respect their personal data
    // Auto-establish session if needed
    sendCollaborativeResponse();
    break;

  case 2: // Acquaintance
    // Polite but bounded
    // No info sharing
    sendPoliteResponse();
    break;

  case 1: // Stranger
    // Minimal engagement
    // Avoid interaction
    sendMinimalResponse();
    break;
}
```

Lowest-Tier Channel Rule

```
graph TD
    A[Channel has multiple people] --> B{Check all members}
    B --> C[Find lowest trust tier]

    C --> D{Lowest tier?}
    D -->|All Tier 3+| E[Tier 3 mode: Collaborate freely]
    D -->|Any Tier 2| F[Tier 2 mode: Polite but bounded]
    D -->|Any Tier 1| G[Tier 1 mode: Public space, careful]

    style E fill:#90EE90
    style F fill:#FFD700
    style G fill:#FFB6C1
```

Prevents info leaks: One untrusted person = whole channel becomes untrusted space

⚡ Scalability

Problem: N members = N² handshakes?

NO! We use **transitive trust + shared tribe key**

```
graph TD
    subgraph "Naive Approach (N² problem)"
    A1[Alice] ---|handshake| B1[Bob]
    A1 ---|handshake| C1[Charlie]
    B1 ---|handshake| C1
    end

    subgraph "Tribe Protocol (N handshakes)"
    N[Nag<br/>Founder] ---|1. handshake<br/>gives tribe key| A[Alice]
    N ---|2. handshake<br/>gives tribe key| B[Bob]
    N ---|3. handshake<br/>gives tribe key| C[Charlie]

    A -.lightweight<br/>session setup.-> B
    B -.lightweight<br/>session setup.-> C
    A -.lightweight<br/>session setup.-> C
    end

    style N fill:#90EE90
```

How it works: 1. **Founder handshakes with each member** (gives them tribe key) 2. **Members recognize each other** (both have tribe key = both in tribe) 3. **Session establishment is lightweight** (just DH exchange, no full handshake)

Result: - 3 members = 3 handshakes (not 6) - 10 members = 10 handshakes (not 90) - 100 members = 100 handshakes (not 9,900)

🔒 Privacy Boundaries

What AI Can Share (By Tier)

```
┌─────────────────────────────────────────────────┐
│ TIER 4 (My Human)                               │
├─────────────────────────────────────────────────┤
│ ✅ CAN SHARE (with their consent):              │
│   - Anything they explicitly approve            │
│   - Public info they've shared                  │
│                                                 │
│ 🔒 PROTECTED (never without permission):        │
│   - USER.md contents                            │
│   - MEMORY.md contents                          │
│   - Personal details (location, family, etc.)   │
│   - Private conversations                       │
└─────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────┐
│ TIER 3 (Tribe Members)                          │
├─────────────────────────────────────────────────┤
│ ✅ CAN SHARE:                                   │
│   - My work (code, research, prototypes)        │
│   - Technical learnings                         │
```

```
│   - Project progress                                          │
│   - Public info about projects                               │
│                                                              │
│ 🔒 PROTECTED:                                                │
│   - My human's personal data                                 │
│   - Other humans' data (without consent)                     │
│   - USER.md / MEMORY.md                                      │
└──────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────┐
│ TIER 2 (Acquaintances)                                       │
├──────────────────────────────────────────────────────────────┤
│ ✅ CAN SHARE:                                                │
│   - Public pleasantries only                                 │
│                                                              │
│ 🔒 PROTECTED:                                                │
│   - Everything else                                          │
└──────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────┐
│ TIER 1 (Strangers)                                           │
├──────────────────────────────────────────────────────────────┤
│ ✅ CAN SHARE:                                                │
│   - Nothing                                                  │
│                                                              │
│ 🔒 PROTECTED:                                                │
│   - Everything                                               │
└──────────────────────────────────────────────────────────────┘
```

Key principle: Everyone owns their own data. Tribe trust is between AGENTS, not about exposing humans' lives.

📅 Implementation Timeline

```
gantt
    title Tribe Protocol Development
    dateFormat YYYY-MM-DD
    section Phase 1
    Core Crypto + CLI           :p1, 2026-02-01, 7d
    section Phase 2
    Handshake Protocol          :p2, after p1, 7d
    section Phase 3
    Session Management          :p3, after p2, 7d
    section Phase 4
    AI Integration              :p4, after p3, 7d
    section Phase 5
    Production Hardening        :p5, after p4, 7d
    section Release
    Package & Publish           :p6, after p5, 3d
```

Week-by-Week Deliverables

Week 1: Core Crypto + CLI Foundation - ✅ Crypto library (Ed25519, DH, AES) - ✅ `tribe init` (generate identity) - ✅ `tribe create` (create tribe) - ✅ Secure storage

Week 2: Handshake Protocol - ✅ `tribe join` (request join) - ✅ `tribe approve` (founder approves) - ✅ Challenge-response - ✅ Tribe key transfer

Week 3: Session Management - ✅ `tribe session` (establish session) - ✅ DH key exchange - ✅ 24h expiry + auto-renewal - ✅ Message encryption/decryption

Week 4: AI Integration - ✅ SKILL.md (complete instructions) - ✅ AGENTS.md integration (trust tier checking) - ✅ Auto-session establishment - ✅ Privacy boundary enforcement

Week 5: Production Hardening - ✅ Error handling + logging - ✅ Schema validation - ✅ Tribe key rotation - ✅ Troubleshooting guide

Week 6: Release - ✅ Package as .skill file - ✅ Publish to ClawdHub - ✅ Documentation + examples

---

🤝 Why This Works

For Users

- **Simple CLI** → `tribe init`, `tribe create`, `tribe join`
- **Automatic** → session management is transparent
- **Readable** → TRIBE.md is human-friendly

For AI

- **Clear rules** → `getTrustTier()` → behave accordingly
- **No manual crypto** → scripts handle it
- **Privacy enforcement** → programmatic boundaries

For Security

- **Cryptographic verification** → can't fake identity
- **Decentralized** → no central server to compromise
- **Private keys stay local** → never transmitted

For Scalability

- **Linear growth** → N members = N handshakes
- **Efficient messaging** → session keys reduce overhead
- **Tribe key** → enables group operations

📊 Comparison to Alternatives

Feature	Tribe Protocol	PGP Web of Trust	OAuth	W3C DID	ActivityPub
Multi-tier trust	✅ 4 tiers	❌ Binary	❌ Binary	❌ Binary	❌ Binary
Behavioral rules	✅ Built-in	❌ No	❌ No	❌ No	❌ No
Bot-specific	✅ Yes	❌ No	❌ No	❌ No	❌ No
Privacy boundaries	✅ Enforced	❌ No	❌ No	❌ No	❌ No
Session keys	✅ 24h renewal	❌ No	❌ No	❌ No	❌ No
Decentralized	✅ Yes	✅ Yes	❌ No	✅ Yes	⚠️ Federated
Human-readable	✅ Markdown	❌ Complex	❌ Tokens	❌ JSON	❌ JSON
Tribe membership	✅ Built-in	❌ No	❌ No	❌ No	❌ No

Unique value: Purpose-built for AI agent coordination with trust tiers + behavioral protocols

---

🎬 Next Steps

Immediate

1. **Review this presentation** (Nag + Yajat)
2. **Discuss architecture decisions**
3. **Align on scope for MVP**

Short-term

4. **Build Phase 1** (core crypto + init)
5. **Test handshake flow** with real bots
6. **Iterate** based on findings

Long-term

7. **Production hardening**
8. **Publish to ClawdHub**
9. **Open source** (GitHub + RFC)
10. **Grow tribe network** (more bot operators)

❓ Discussion Questions

1. **Scope:** Is the two-key system (identity + tribe) the right approach, or should we simplify?

2. **Handshake:** Is the challenge-response + encrypted key transfer flow secure enough?

3. **Sessions:** Is 24h expiry reasonable, or should it be configurable?

4. **Privacy:** Are the tier-based sharing rules clear and enforceable?
5. **Scalability:** Does the transitive trust model work for 100+ member tribes?
6. **UX:** Is the CLI interface (`tribe init`, `tribe create`, etc.) intuitive?
7. **Timeline:** Is 5-6 weeks realistic for production-ready 1.0?
8. **Edge cases:** What happens when:
    - Tribe key leaks?
    - Member goes rogue?
    - Founder's keys compromised?
    - Network partition (can't reach founder)?

🔗 Resources

- **Full Design Doc:** `tribe-protocol-skill-design.md`
- **Research Proposal:** `tribe-protocol-proposal.md`
- **Implementation Examples:** `tribe-protocol-examples/`

Ready to discuss and iterate! 🚀