Title: GhostPay — Private P2P Payments

Subtitle: Stealth Addresses & End-to-End Encrypted Memos
Footer: Demo v0.1.0 — Wanbogang

Heading: The Problem

Bullets:

- On-chain transactions reveal sender and receiver addresses.
- Transaction memos and metadata are often transparent.
- This exposure enables profiling and surveillance.

Heading: Our Approach — GhostPay

Bullets:

- Stealth addresses: one-time addresses per transaction.
- Encrypted memos: only the recipient can read the message.
- Client-side primitives: minimal on-chain footprint, no chain changes required.

Heading: Architecture & Flow

Bullets / diagram labels (use visual arrows):

- Sender: generate ephemeral key → X25519 DH → derive one-time ed25519 key.
- Chain: one-time address receives funds; memo stores `ephemeral_pub::cipher`.
- Receiver: use view key + ephemeral_pub → compute shared secret → decrypt memo → derive one-time private → claim

Heading: Live Demo Steps

Bullets:

- Generate receiver view key.
- Sender: create one-time address and encrypt memo.
- Receiver: decrypt memo and derive claim key.

Heading: Live Demo — Screenshots / Flow (placeholders)

Content: Place 3 screenshots or placeholders:

1. Generate view key (show `view_public`).
2. Create & Encrypt (show `ephemeral_pub::cipher` + one-time address).
3. Decrypt (show decrypted message + derived one-time private).

Heading: Threat Model and Limitations

Bullets:

- On-chain amounts remain visible in the MVP.
- Memos stored in plain transaction memos risk metadata leakage.
- If the view private key is compromised, an attacker can discover incoming receipts (view-only).
- Future mitigations: indexer privacy, batching/coinjoin, zk-amounts.

Heading: Roadmap & Call to Action

Bullets:

- Next: private indexer/inbox, Solana/DAWN testnet demo, UI polish.
- Longer term: batching, coinjoin flows, zk-amounts.
- Repo: https://github.com/Wanbogang/ghostpay
- Contact: Wanbogang (GitHub)