OpenSSL CONFERENCE

Breaking Deepfakes with Shared Secrets

Jonathan Bateman

§ PRIVACY AS A FUNDAMENTAL HUMAN RIGHT

15.10.2026

Agenda

- Motivation
- Methodology
- Protocol Overview
- Hands On Demo
- Security Considerations
- Takeaways & Future Work

15.10.2026

Motivation

The Deepfake Thought Experiment

OpenSSL CONFERENCE

- What if a malicious actor **cloned your voice and face** then called a family member or close friend **today**?
- How would the person receiving that call **determine if it was actually you?**

The Threat Landscape Has Changed

OpenSSL CONFERENCE

Secure Channels

- Deepfake video → E2EE zoom call
- Signal group → impersonated member

Insecure Channels

- Fake Account → months long catfishing over social media
- Voice Clone → trusted phone number

The common thread: Identity is decided at the human layer

The Identity Binding Problem

OpenSSL CONFERENCE

Core Question: Does this public key or public identifier really belong to Bob?

- Certificate Authorities
- PGP Web of Trust
- Trust on First Use (TOFU)
- Phone numbers / SIM

- Email Address
- OAuth
- WebAuth / Passkeys
- Keybase (R.I.P.)

***Bootstrapping trust** on the **first interaction** is the challenge.*

Why Traditional Authentication Still Fails?

- **Key Point:** Deepfakes target the **human decision point** not the cryptography

- Passwords and 2FA are **transferrable**
- Biometrics are cloneable with **AI**
- Even secure apps fall victim to impersonation

What if...Security == Usability?

OpenSSL CONFERENCE

- **Constraint 1:** If users can easily maneuver around it, it is insecure!
- **Constraint 2:** Make defeating deepfakes as easy as sending a message
- **Constraint 3:** Could your grandma use it?

The Question That Started This

OpenSSL CONFERENCE

What does **every human have** that AI can never **fake**?

Bridge: “I have their public key” → “I know it’s the right person”

The Twist

OpenSSL CONFERENCE

Shared Memory Agreement

↓

Challenge-Response

↓

Cryptographic Key Exchange

↓

Verified Trust

Methodology

The Oldest Authentication Mechanism

OpenSSL CONFERENCE

Phrasal call-and-response: authenticate via a pre-shared question *Q* and expected answer *A* (meaningless to outsiders)

- Countersigns
- Dead drops / One-time pads (OTPs)
- Recognition signals
- Duress codes

Through-line: open channel → requires private knowledge

Sci-Fi Solved Deepfakes & Doppelgängers

OpenSSL CONFERENCE

Star Trek

Personal Memory Prompt

- "What was our weekly tradition in Starfleet Academy?"

Blade Runner

Experience Probing

- Voight–Kampff: affect + memory cues

Tron Legacy

Relational Signals

- "That's a hell of a way to treat your son."

The Core Insight

What does Bob have that a doppelgänger doesn't?

Cloneable Signals

- Voice
- Face

Inaccessible Signal

- Shared episodic memory

Private → high-entropy → unforgeable at the human layer

What Is The Doppelgänger Protocol?

OpenSSL CONFERENCE

A **memory-gated bootstrapping** protocol for deepfake-era communication

Authenticate a person by proving shared experience **without trusting audio, video, or biometrics.**

The Naïve Approach

"In what city were you born?"

inferable (OSINT) · brittle · gameable

Design Requirements

High entropy · Semantically flexible · Experiential

What Good Memory Challenges Look Like

OpenSSL CONFERENCE

High-entropy prompts

- *"In what cities were our layovers on our way to the 2026 OpenSSL Conference?"*
- "What story arc do we always argue about in our favorite show?"

Shared context

- private
- experiential
- relationship specific

*How do we turn natural-language answers into something we can **test mathematically**?*

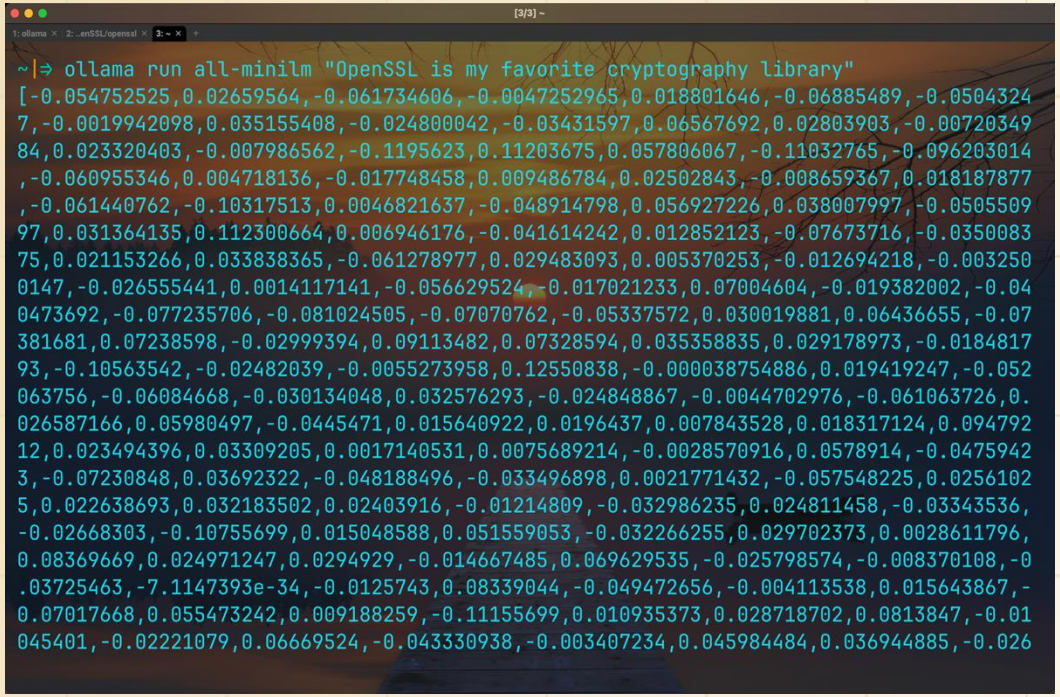
Enter Embedding Models

15.10.2026

OpenSSL CONFERENCE

- **2013:** Tomas Mikolov and his team at Google create **Word2vec**
- An **embedding model** converts text into a vector representing its meaning in high-dimensional space
- The vector captures **semantic meaning**;
  - Similar text is grouped in high dimensional space
- This allows machines to compare language by meaning, not just by exact words

From Text To Vector



Security Considerations Of Embedding Models

OpenSSL CONFERENCE

Property	Why it matters
· Dimensionality (*d*)	· Higher *d* creates room for distinction
· Model consistency	· Must match model-ID / version
· Local-first embeddings	· Server-side embedding leaks plaintext answers to a third party

This implementation: all-MiniLM-L6-v2 || *d* = 384
(semantic fidelity · browser-feasible)

Embeddings Verification (Cosine Threshold)

OpenSSL CONFERENCE

Mechanism

1. Embed stored answer *A* and response $\hat{A}$:
   - $A \mapsto e, \hat{A} \mapsto \hat{e} \in \mathbb{R}^d$
2. Similarity score (semantic match):
   - $s = \cos(e, \hat{e}) = (e \cdot \hat{e}) / (\|e\| \|\hat{e}\|) \in [0, 1]$
3. Accept iff $s \geq \tau$
   - No case sensitivity, no exact strings needed

Toy Example & Usability

"Park in NYC" vs "Park in New York" ✓

"Park in NYC" vs "Parked Car" ×

Security Parameter: τ

- everyday people: $\tau \geq 0.75$
- security aware: $\tau \geq 0.85$
- high-stakes: $\tau \geq 0.98$

Protocol Overview

Three Parties, One Goal

OpenSSL CONFERENCE

Alice (browser)	Server	Bob (browser)
· ECDH key generation	· Session Relay	· ECDH key generation
· Memory commitment	· Similarity Gate	· Memory response
· Key derivation (HKDF)	· Key-blind	· Key derivation (HKDF)

Goal: Alice and Bob derive the same shared key,
while the server never sees either private key

Step-by-Step Flow (Alice)

15.10.2026

OpenSSL CONFERENCE

- **ECDH keygen (browser):** $(sk_a, pk_a) \leftarrow KeyGen()$
  - $\rightharpoonup$ sk_a stays local; export pk_a as JWK
- **Set memory challenge:** choose (Q, A) and compute Embed(A)
  - Embed(A) $\rightarrow e_a$
- **Post session state:** { alice_pubkey_jwk, answer_embedding }
  - $\rightharpoonup$ Server stores in Redis, returns session_id + link / QR
- **Wait:** open Server-Sent-Events (SSE) channel
  - Alice is waiting for a *PASS* on the memory challenge + Bob's public key

Step-by-Step Flow (Bob)

- **Open link / Scan Qr code:** load session_id and view *Q*
- **ECDH keygen (browser):** $(sk_\beta, pk_\beta) \leftarrow KeyGen()$
- **Submit response:** { bob_pubkey_jwk, *answer* }
- **Server gate (similarity check):**
  - ⇀ Embed(Â) → e_β , compute $s = \cos(e_a, e_\beta)$
  - ⇀ iff $s \geq \tau$ → PASS
  - ⇀ publish SSE: { bob_pubkey_jwk, alice_pubkey_jwk }

15.10.2026

The ECDH Handshake

OpenSSL CONFERENCE

1. **Alice receives pk_β** (Bob's public key) via SSE → imports it
2. **Alice derives:** the joint secret from Bob's public key
   - $Z_a \leftarrow ECDH(sk_a, pk_\beta) \rightarrow K_a \leftarrow HKDF\text{-}SHA256(Z_a)$ (AES-GCM 256)

3. **Bob receives pk_a** (Alice's public key) via SSE→ imports it
4. **Bob derives:** the joint secret from Alice's public key
   - $Z_\beta \leftarrow ECDH(sk_\beta, pk_a) \rightarrow K_\beta \leftarrow HKDF\text{-}SHA256(Z_\beta)$ (AES-GCM 256)

$Z_a = Z_\beta \rightarrow K_a = K_\beta$ (ECDH correctness)

Same shared secret produces identical session keys.

What The Server Actually Sees

OpenSSL CONFERENCE

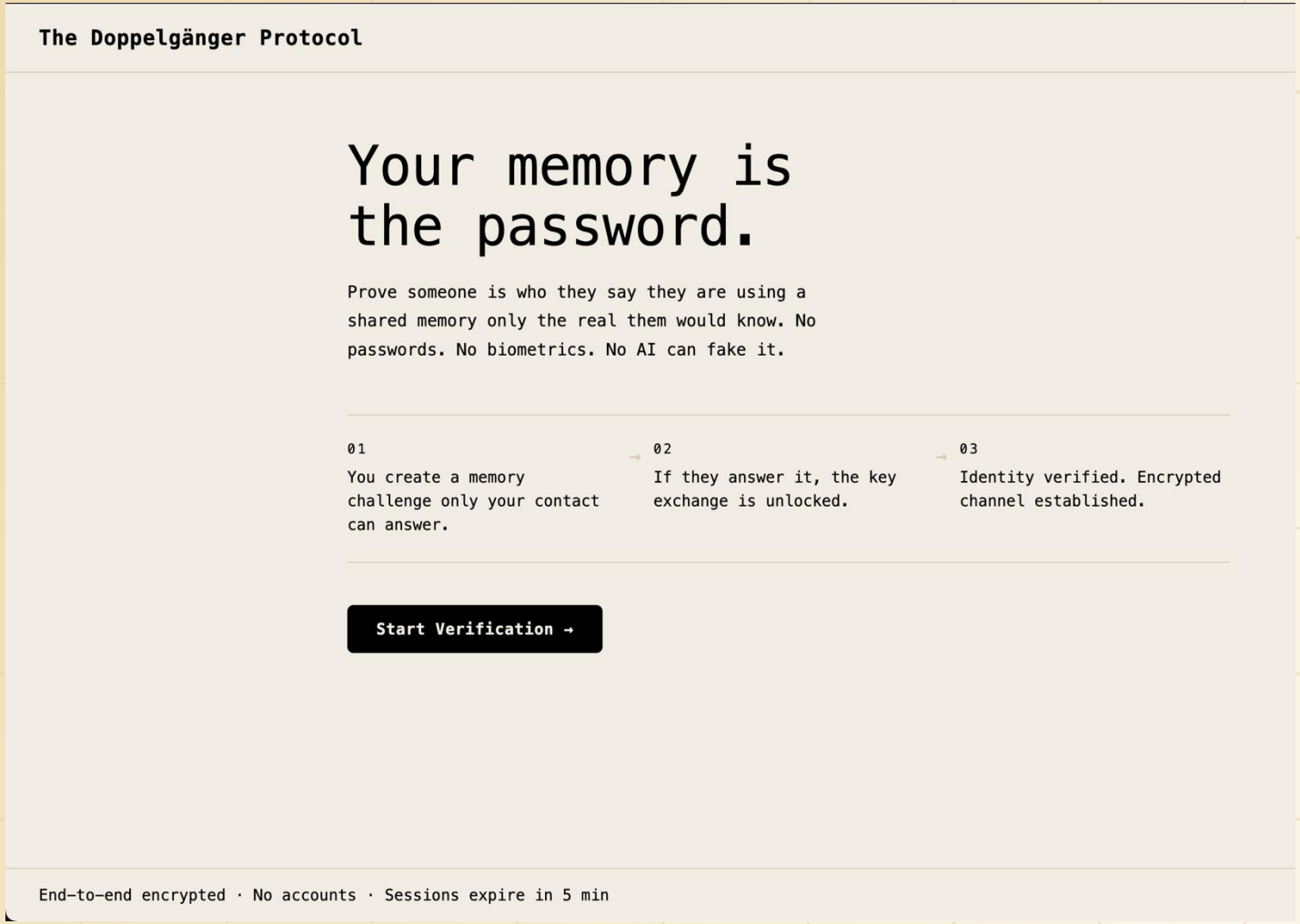
Data	Seen?	Where / Why?
Challenge question Q	**Yes**	Public question with a private answer
Alice's private key (sk_a)	**No**	Never leaves Alice's browser
Bob's private key (sk_β)	**No**	Never leaves Bob's browser
Shared key K	**No**	Derived locally via ECDH + HKDF
Alice plaintext memory answer A	**Yes (current implementation)**	Forwarded to embedding microservice
Bob plaintext memory answer $\hat{A}$	**Yes (current implementation)**	Forwarded to embedding microservice
Embeddings (e_a, e_β)	**Yes**	Used for cosine score + threshold gate
Public keys (pk_a, pk_β)	**Yes**	Relay-only for handshake via SSE

Hands On Demo

Your Memory Is The Password

15.10.2026

OpenSSL CONFERENCE



Create Your Challenge

15.10.2026

OpenSSL CONFERENCE

Create your challenge.

Ask something only the real person would know.
Your encryption keys are being generated in your browser right now.

YOUR NAME

Jonathan Bateman

MEMORY QUESTION (Bob will see this)

In what cities were our layovers on the way to the 2026 OpenSSL conference?

75/500

YOUR ANSWER (Bob will never see this)

We had layovers in Chicago and Frankfurt Germany.

49/500

- ✓ Encryption keys ready

Generate Challenge Link

Your answer is converted to a mathematical vector and only temporarily seen by the embedding service.

Sharing The Invite

15.10.2026

OpenSSL CONFERENCE

EMBEDDING OF ALICE'S ANSWER TO:

"In what cities were our layovers on the way to the 2026 Open..."

First 40 of 384 dimensions · stored in Redis · raw answer discarded

[0.097439, −0.025688, 0.024308, 0.047291, 0.009528, 0.028042, −0.070633, −0.023066, −0.046889, −0.067561, 0.01547, 0.015532, −0.05306, 0.016453, 0.046955, −0.069802, 0.012884, −0.121657, −0.030251, −0.098882, −0.089966, −0.094665, −0.056626, −0.011666, 0.069525, 0.004368, 0.015252, 0.019001, 0.05427, −0.011501, 0.02893, −0.002457, −0.046215, 0.018451, 0.02802, −0.047908, 0.080926, −0.031988, 0.018619, −0.015458...]

Bob's answer will be embedded the same way.
Cosine similarity determines if they match.

Share this with Bob.

This link contains your memory question and your public key.
Bob must answer correctly before the encrypted channel opens.

https://doppelgangerprotocol.app/verify/s/aKSwDCHD4… Copy

Scan with any camera. This works on any device with a browser.

- Waiting for Bob to answer the challenge...

Receiving The Invite

15.10.2026

OpenSSL CONFERENCE

CHALLENGE FROM **JONATHAN BATEMAN**

In what cities were our layovers on the way to the 2026 OpenSSL conference?

Answer in your own words. Exact wording doesn't matter because we use semantic similarity.

YOUR NAME

Bob

YOUR ANSWER

We had to stop in Frankfurt and Chicago

✓ Encryption keys ready

Submit Answer

Your keys were generated here, in your browser. Nothing was installed.

Try It Out

doppelgangerprotocol.app

15.10.2026

- Pair up with a stranger sitting next to you
- One person plays the role of **'Alice' or initiator**
- *Q*: what is the full name of the hotel that most attendees are staying at?
- The other person plays the role of **'Bob' or receiver**

Security Considerations

The Bootstrappers Advantage

Time pressure shift

- Defender only has to run the protocol once
- Attacker must pre-emptively infer / "OSINT" the relationship-specific secret

Example

- Impersonate & deepfake before trust is established
- Attacker must know the answer to, "Where did we first meet?"

Possible Attacks

MITM

- Invite link is intercepted or forwarded
- Attacker only sees a public question, not a session key

Replay Attacks (deepfakes)

- Fresh session means a fresh challenge
- Time bound response window

Weak Questions

- Use insider knowledge
- Attacker must know the answer to, "Where did we first meet?"

Device Compromise

- out of scope: spyware can observe answers
- mitigate with Secure Enclave + ephemeral sessions

Memory Quality & Divergence

15.10.2026

OpenSSL CONFERENCE

Protocol failure modes

Where the Doppelgänger Protocol faces real-world pressure

FM 1 — Weak questions

Durability & quality

How do we stop users from recreating the same weak security questions that already exist?

"What's your mother's maiden name?" is just a security question by another name.

FM 2 — Memory divergence

Usability

Alice and Bob share the same memory but describe it differently. The protocol fails to recognize it.

"We got ramen" vs. "we grabbed noodles." Same night, different words.

FM 3 — Knowledgeable attacker

Adversarial guessing

An attacker who personally knows Alice or Bob may be able to guess even high-entropy answers.

Scope boundary: designed for remote adversaries, not intimate ones.

FM 4 — No shared history

Bootstrap problem

Alice and Bob have just met. There is no shared episodic memory to anchor the protocol on.

The protocol deepens trust. It doesn't create it ex nihilo.

Each failure mode addressed on the following slide

Memory Entropy Score (MES)

Making Quality Measurable

15.10.2026

OpenSSL CONFERENCE

Memory Entropy Score (MES)

Four dimensions of experiential unpredictability

Temporal entropy

When?

↓ Low
"Last summer"

↑ High
"Tuesday night before the conference, after the venue walkthrough"

Spatial entropy

Where?

↓ Low
"A restaurant"

↑ High
"Basement ramen place, two blocks from venue, only six seats"

Contextual entropy

What?

↓ Low
"We had dinner"

↑ High
"You spilled the broth and we laughed about it for ten minutes"

Reasoning entropy

Why / How?

↓ Low
"It was fun"

↑ High
"You said something that changed how I thought about the work"

Total MES = Σ (T + S + C + R) — scored in bits of experiential unpredictability

MES & Mitigations

Mitigating the Pitfalls of the Doppelgänger Protocol

15.10.2026

OpenSSL CONFERENCE

Failure mode	MES response
FM 1 — Weak questions How do we stop users from recreating bad security questions with the same old weaknesses? *"What's your mother's maiden name?"*	**MES scores at enrollment** Rates entropy across all four dimensions before committing. Rejects weak questions and runs OSINT to find guessable answers.
FM 2 — Memory divergence Alice and Bob share the same memory but describe it differently. The protocol rejects a genuine match as a false negative.	**Specificity forces convergence** High-MES questions are specific enough to trigger the same memory in both parties. Cosine threshold t absorbs natural linguistic variation.
FM 3 — Knowledgeable attacker An attacker who personally knows Alice or Bob may be able to guess even high-entropy answers from public or leaked data.	**OSINT simulation at enrollment** MES estimates what a remote attacker can infer from public data. Intimate adversaries are an out-of-scope threat, stated clearly.
FM 4 — No shared history Alice and Bob have just met. No shared episodic memory exists to anchor the protocol. *Colleagues, conference attendees, clients: where is the first anchor?*	**MES finds the first anchor** Surfaces existing shared context: a contract detail, a conference session, insider company knowledge. *No deep history needed. Just one shared moment to start.*

Failure mode (protocol limitation) MES response (design mitigation)

Takeaways & Future Work

What To Remember

Bottom line: verify the person, not the medium!

- Deepfakes break voice & video as trusted channels
- The Doppelgänger Protocol verifies via human shared-memory challenges
- On-device keys and embeddings lead to minimal disclosure
- Bootstrap trust and verify before requesting money, access, or secrets

What's Next

15.10.2026

- **Memory Entropy Score (MES):** formalizing the framework and writing the code
- **Local embeddings:** run the embedding model in-browser so raw answers never leave the device
- **“AdLib” format:** structured slot extraction so users never have to manually write a template
- **Protocol spec:** a public Doppelgänger Protocol RFC / whitepaper (primitives, threat model)
- **Open-source iteration:** audit, fork, break, and improve the design

Thank You
Questions?

Jonathan Bateman
Rochester Institute of Technology

Email: jmb7342@rit.edu
Repo: https://github.com/doppelgangerprotocol