🚀 TrustCam-DTN: End-to-End Technical Summary

🔹 What the system does (1-line pitch)

TrustCam-DTN securely captures images at the device, encrypts them before transmission, attaches provenance proofs, and delivers them reliably using Delay-Tolerant Networking — even without internet — while maintaining tamper-evident logs.

🧩 System Flow (Step-by-Step)

1️⃣ Secure Capture at ESP32-CAM

- Camera captures an image into framebuffer
- Before storing or transmitting it, we encrypt it using **AES-GCM**
- This ensures **confidentiality + integrity (via auth-tag)**

🧠 Why AES-GCM?

It provides both encryption and authentication without needing a separate HMAC, and it's fast enough for embedded devices.

2️⃣ Provenance and Authenticity

After encryption, we generate:

- **SHA-256 hash** of encrypted file
- **ECDSA signature** over that hash
- Metadata: timestamp, device ID, firmware version

This proves:

Property	Mechanism
Who captured it	Signature w/ device private key
When it was captured	Timestamp in metadata
Whether modified	Hash + signature mismatch if altered

🧠 Expected viva question:
Q: Why sign after encryption instead of before?
✓ Because signing plaintext leaks info about raw data—signing ciphertext protects structure and maintains privacy.

3️⃣ Local Transparency Log

Each capture event is appended into a **hash-chained log** stored on device:

(prev_hash + event_hash) → SHA-256 → new entry

It behaves like a lightweight blockchain:

- No consensus
- No mining
- Just immutability

🧠 Viva Q:
Q: Why not use a full blockchain like Ethereum?
✓ Because blockchain is overkill for an IoT device. Hash chaining gives **tamper evidence with near-zero resource cost.**

4️⃣ Delay-Tolerant Networking (DTN)

Instead of assuming continuous internet connection, we use:

✓ **Store-carry-forward model (BPv7 Bundle Protocol)**
✓ Messages are queued if network unavailable
✓ Delivered automatically when a route appears
✓ Can integrate satellite (Iridium) later

🧠 Viva Q:
Q: How is this different from normal TCP/IP transmission?
✓ TCP requires continuous end-to-end connectivity.
✓ DTN can deliver data **hours or days later**, across disconnected nodes.

5️⃣ Proxy-Friendly, Zero-Trust Transport

Because encryption happens at the source:

- DTN nodes
- Servers
- Gateways

...cannot see or alter plaintext.

The server **stores encrypted data but cannot read it** unless authorized.

🧠 Viva Q:
Q: Why is this better than WhatsApp-style end-to-end encryption?
✓ WhatsApp encrypts messages *during transport*, but server still controls identity, routing, and metadata.
✓ TrustCam enforces:

- Provenance
- Tamper evidence
- Secure re-sharing (PRE)
- Offline delivery

...none of which WhatsApp guarantees.

6️⃣ Backend Verification + Decryption (Flask)

When data arrives:

Step	Operation
1	Compute new hash
2	Verify ECDSA signature
3	Verify transparency log chain
4	Decrypt AES-GCM
5	Store reconstructed image

If any step fails → backend rejects data.

🧠 Viva Q:
Q: Can the server modify logs or images?
✓ No — modification breaks signature and hash-chain integrity.

7️⃣ Testing and Validation

We tested:

- 🧪 Raw HTTP uploads

- 🧪 Curl-based multipart uploads
- 🧪 Retrieval via /images and /images/<file>
- 🧪 Log verification via /log/verify

Everything worked end-to-end.

🎓 Key Viva Talking Points

Category	Bullet Answer
Confidentiality	AES-GCM ensures encryption + authentication
Integrity	SHA-256 hash + signature prove no tampering
Authenticity	ECDSA keypair binds image to specific device
Non-Repudiation	Device cannot deny generating signed image
Tamper Detection	Hash-chained logs detect alterations
Offline Support	DTN ensures store-and-forward resilience
Zero-Trust Architecture	Server does not need to trust devices

---

💬 Expected Deep Questions & Strong Answers

🔥 Q: What happens if someone modifies a single byte in the encrypted payload?

AES-GCM authentication tag will fail, signature verification will fail, and log entry will mismatch — system rejects the bundle.

🔥 Q: How do you handle replay attacks?

Metadata includes timestamp + device ID. Backend rejects duplicates or stale entries.

🔥 Q: Why not just use HTTPS?

HTTPS protects transport *only while online*.
TrustCam protects:

- **before transmission**
- **during storage**
- **during routing**
- **without internet**

🔥 **Q: What's the biggest bottleneck?**

- Cryptographic operations on ESP32 if scaling many images.
- DTN routing latency when no direct route exists.

🏁 **Final One-Sentence Conclusion**

TrustCam-DTN is a zero-trust secure imaging platform combining encryption-at-source, cryptographic provenance, tamper-evident logging, and delay-tolerant delivery—ensuring images remain verifiable and confidential from capture to final storage, even in disconnected environments.