Alma Mater Studiorum – Università di Bologna

Relazione di Cybersecurity

Aegis: un framework per la condivisione sicura di immagini

Marzia De Maina – 0001194461
Chiara Sivieri – 0001198267

30/01/2026

Indice

1 Nascita dell'idea

La nascita del progetto **Aegis** deriva dall'esigenza crescente di affrontare una delle problematiche più critiche della cybersecurity moderna: la diffusione non autorizzata di contenuti digitali sensibili. In particolare, il progetto si colloca nel contesto del fenomeno noto come *revenge porn*, ovvero la pubblicazione o condivisione di immagini intime senza il consenso della persona coinvolta.

Dal punto di vista della sicurezza informatica, questo scenario evidenzia un limite strutturale delle tecniche tradizionali di protezione dei dati. Infatti, anche qualora un contenuto venga trasmesso in modo cifrato e sicuro, una volta che esso viene legittimamente ricevuto da un destinatario, diventa estremamente complesso impedirne la successiva distribuzione illecita.

Aegis nasce quindi con un obiettivo specifico: introdurre un meccanismo di **protezione forense** che consenta di supportare l'attribuzione del contenuto ad un soggetto responsabile, garantendo tracciabilità e non ripudiabilità.

Il nome stesso del progetto richiama questo concetto. **Aegis** deriva infatti dal termine greco *aegis*, lo scudo mitologico associato alla dea Atena, simbolo di protezione e difesa. In modo analogo, il sistema si propone come uno **scudo digitale** in grado di proteggere l'integrità e la sicurezza delle immagini personali mediante tecniche avanzate di watermarking.

1.1 Introduzione del progetto

Aegis è un'applicazione progettata per l'inserimento e la verifica di watermark invisibili all'interno di immagini digitali, con finalità investigative e di contrasto alla diffusione illecita.

È stato adottato un approccio di **forensic watermarking**, dove la firma nascosta non serve soltanto a marcare la proprietà del contenuto, ma diventa uno strumento di identificazione del percorso di distribuzione.

Il sistema è implementato secondo un'architettura client-server, composta da:

- un **backend forense** basato su Flask, responsabile dell'estrazione e verifica del watermark;
- un **frontend mobile** sviluppato in Flutter, orientato all'interazione utente;
- una **pagina amministrativa di verifica** per l'analisi manuale dell'applicazione effettiva del watermark.

1.2 Obiettivi di Aegis

Al cuore del progetto vi è lo sviluppo di un **sistema di watermarking invisibile** per immagini sensibili che sia non solo *efficace*, ma anche *robusto* contro tentativi di alterazione come la compressione o il ritaglio. Il modello proposto supporta la piena tracciabilità della diffusione dei dati e si configura come uno **strumento pratico** per la digital forensics. L'intento finale è quello di potenziare l'attuale panorama della cybersecurity con nuove soluzioni dedicate alla protezione dei dati personali.

Aegis rappresenta quindi un esempio concreto di applicazione della cybersecurity nel dominio della tutela digitale, ponendosi come strumento innovativo di prevenzione e attribuzione forense.

2 Cosa sono i watermark e utilizzo dei Watermark

Nel contesto della sicurezza informatica e della digital forensics, il concetto di **digital watermarking** rappresenta una delle tecniche più rilevanti per la protezione e la tracciabilità dei contenuti multimediali [3].

Un *watermark digitale* può essere definito come un'informazione aggiuntiva, incorporata all'interno di un contenuto (immagine, audio o video) o nei metadati, in modo tale da risultare:

- **impercettibile** all'occhio umano (invisibile);

- **persistente** anche dopo trasformazioni comuni (robustezza),

- **estraibile e verificabile** tramite la procedura implementata.

L'obiettivo fondamentale del watermarking non è la cifratura del contenuto, bensì la sua marcatura [5]. In ambito cybersecurity, questa proprietà è cruciale perché consente di affrontare scenari in cui un contenuto viene legittimamente ricevuto, ma successivamente diffuso senza autorizzazione.

In particolare, Aegis impiega un approccio di **forensic watermarking**, dove la firma nascosta è progettata per supportare:

- identificazione del mittente originario;

- attribuzione del destinatario iniziale;

- tracciabilità della catena di distribuzione.

Dal punto di vista tecnico, le tecniche di watermarking possono essere suddivise in due categorie principali:

1. watermarking nel **dominio spaziale**;
2. watermarking nel **dominio delle trasformate** (frequenze).

Aegis implementa e studia più strategie, tra cui DCT, LSB e Spread Spectrum.

2.1 LSB

Una delle tecniche più semplici e diffuse nel dominio spaziale è il watermarking tramite **Least Significant Bit (LSB)**.

Il principio consiste nel modificare il bit meno significativo dei pixel di un'immagine per codificare l'informazione del watermark [6]. Dato che tale bit contribuisce in modo minimo al valore complessivo del colore, l'alterazione risulta generalmente invisibile.

Ad esempio, un pixel con valore:

$$10110101$$

può essere modificato in:

$$10110100$$

senza che la differenza sia percettibile.

Il vantaggio principale dell'approccio LSB è la sua semplicità computazionale e l'elevata capacità informativa.

Tuttavia, dal punto di vista della sicurezza, esso presenta limiti significativi:

- vulnerabilità alla compressione JPEG;

- fragilità rispetto a rumore e filtri;

- facilità di attacco mediante steganalisi statistica.

Di conseguenza, LSB è spesso considerato un watermark **fragile**, adatto a scenari di verifica di integrità ma meno efficace contro avversari attivi.

2.2 DCT

La **Discrete Cosine Transform (DCT)** è una trasformata ampiamente utilizzata nella compressione delle immagini, in particolare negli standard JPEG [2]. Essa converte l'informazione da un dominio spaziale (pixel) ad un dominio di frequenza, rappresentando l'immagine come combinazione di coefficienti coseno [1].

Dal punto di vista matematico, la DCT bidimensionale per un blocco $N \times N$ (nel nostro caso 8×8) è definita dalla seguente formula:

$$F(u,v) = \frac{1}{4} C(u)C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right] \tag{1}$$

Dove:

- $f(x,y)$ è l'intensità del pixel alle coordinate (x,y);
- $F(u,v)$ è il coefficiente DCT risultante;
- $C(u), C(v) = \frac{1}{\sqrt{2}}$ per $u, v = 0$, altrimenti 1.

I coefficienti a bassa frequenza (in alto a sinistra nella matrice risultante) contengono la maggior parte dell'energia dell'immagine, ed è qui che Aegis opera per massimizzare la robustezza.

Nel contesto del watermarking, l'idea è quella di incorporare la firma non direttamente sui pixel, ma modificando selettivamente alcuni coefficienti DCT.

Questo approccio è particolarmente vantaggioso perché garantisce maggiore robustezza rispetto a trasformazioni tipiche come:

- compressione con perdita;
- filtraggio e smoothing;
- ridimensionamento moderato.

In cybersecurity, il watermark DCT viene considerato una tecnica di tipo **robusto**, poiché sopravvive a molte operazioni di post-processing che un attaccante potrebbe utilizzare per tentare la rimozione.

2.3 Spread Spectrum

Una tecnica avanzata spesso utilizzata in ambito forense è il watermarking basato su **Spread Spectrum**, ispirato ai principi delle telecomunicazioni [4].

L'idea fondamentale è distribuire il watermark su un ampio intervallo di frequenze o su molteplici componenti dell'immagine, rendendo estremamente difficile la sua rimozione completa [7].

In questo modello, la firma viene combinata con un segnale pseudo-casuale:

$$I_w = I + \alpha W_p$$

dove:

- I è l'immagine originale;
- W_p è il watermark modulato pseudo-random;

- α è un parametro di intensità.

I vantaggi cybersecurity di Spread Spectrum includono:

- elevata resilienza ad attacchi intenzionali;

- difficoltà di individuazione del watermark;

- capacità di sopravvivere a trasformazioni aggressive.

Questa tecnica è particolarmente rilevante nei sistemi di forensic watermarking, dove l'avversario non è passivo ma tenta attivamente di rimuovere ogni informazione di tracciabilità.

2.4 Considerazioni comparative

In sintesi, le tecniche adottate da Aegis si collocano lungo un continuum tra semplicità e robustezza:

- **LSB**: semplice e invisibile, ma fragile.

- **DCT**: più complesso, ma robusto contro compressione.

- **Spread Spectrum**: altamente resistente e adatto a scenari forensi avanzati.

La combinazione di tali approcci consente al progetto Aegis di esplorare soluzioni concrete per la tracciabilità digitale e la protezione contro la diffusione non autorizzata di contenuti sensibili, con applicazioni dirette in ambito cybersecurity e digital evidence.

3 Implementazione

La fase implementativa del progetto **Aegis** rappresenta la traduzione operativa dei concetti teorici esposti nelle sezioni precedenti. Lo scopo non è soltanto dimostrare la fattibilità del watermarking digitale, ma realizzare un sistema concreto e modulare, utilizzabile in scenari reali di cybersecurity e digital forensics.

Dal punto di vista architetturale, Aegis è stato progettato secondo un modello **client-server**, con separazione netta tra:

- componente mobile lato utente;

- componente backend forense dedicata all'elaborazione;

- interfaccia amministrativa per la verifica investigativa.

Questa distinzione segue un principio chiave di `secure software engineering`: isolare le funzioni critiche (watermark extraction e attribution) in un dominio backend controllato, minimizzando l'esposizione diretta delle logiche sensibili.

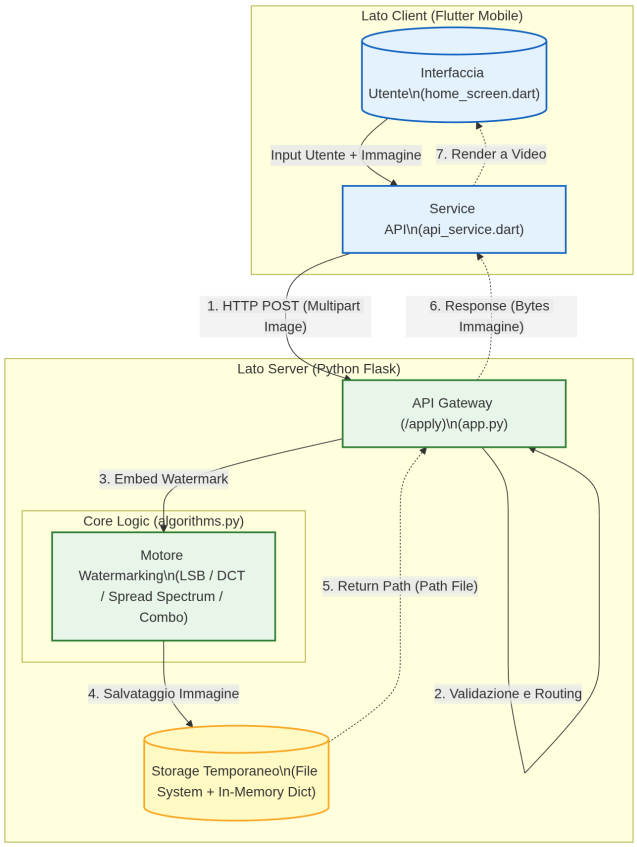

Figura 1: Architettura logica del sistema Aegis: interazione tra Client Flutter, API Gateway e Core Algoritmico.

3.1 Strumenti utilizzati

Per garantire portabilità, modularità e facilità d'uso, lo sviluppo ha impiegato tecnologie moderne e ampiamente adottate sia in ambito industriale che accademico.

Le principali scelte tecnologiche sono:

- **Flutter** per lo sviluppo cross-platform del frontend mobile;
- **Python** + **Flask** per la realizzazione del backend API forense;
- **OpenCV** e **Pillow** per la manipolazione e trasformazione delle immagini;
- **NumPy** per la gestione efficiente delle matrici e la scansione a griglia (Grid Search).

Queste tecnologie consentono di combinare efficienza operativa e sperimentazione rapida, fondamentale in un progetto orientato alla ricerca applicata.

3.1.1 Flutter

La scelta di adottare Flutter per lo sviluppo del frontend non è casuale, ma risponde a precise esigenze architetturali e prestazionali. Il framework permette infatti uno sviluppo nativo multi-piattaforma (Android/iOS) con un'unica base di codice, garantendo al contempo prestazioni elevate

grazie al proprio motore di **rendering integrato**. Inoltre, la sua capacità di gestire interfacce modulari e logiche asincrone si sposa perfettamente con la necessità di un'**applicazione reattiva e moderna**.

Sotto il profilo della cybersecurity, l'adozione di Flutter offre un vantaggio strutturale: consente di isolare nettamente la gestione dell'interfaccia utente dal core forense, delegando ogni operazione critica al backend. L'esperienza utente in Aegis è stata disegnata per essere **lineare e sicura**: dopo le fasi di registrazione e login, l'utente viene guidato nella selezione o acquisizione dell'immagine, che viene successivamente inviata al backend tramite API per l'analisi, fino alla visualizzazione finale dell'esito di verifica.

3.2 Librerie e dipendenze

Per garantire solide capacità di elaborazione digitale e resilienza forense, il backend Python si avvale di un ecosistema di librerie altamente specializzate.

Le dipendenze principali includono:

- `opencv-python`: per l'implementazione efficiente di trasformate e filtri sulle immagini;
- `Pillow`: essenziale per il caricamento e la gestione dei diversi formati immagine;
- `numpy`: fondamentale per l'analisi steganografica e la gestione dei blocchi DCT;
- `Flask`: scelto come framework leggero per esporre le API REST;
- `flask-cors`: per la gestione sicura delle richieste cross-origin.

Tra queste, l'utilizzo intensivo di `numpy` gioca un ruolo chiave in ottica di performance, permettendo di eseguire operazioni complesse (come la trasformata del coseno su blocchi multipli) in tempi ridotti, essenziale per garantire un'esperienza utente fluida.

3.3 Ambiente di sviluppo

L'intera architettura è stata configurata per rispondere a requisiti di riproducibilità, modularità del codice e, soprattutto, netta separazione tra la *user interface* e la *logica forense*.

Nello specifico, il backend opera in un ambiente Python dove le dipendenze sono rigorosamente definite tramite il file `requirements.txt`, favorendo un approccio conforme ai principi della *reproducible research*. Parallelamente, il frontend Flutter sfrutta un modello di sviluppo basato su Dart, che supporta agilmente sia l'emulazione locale che il deploy su dispositivi fisici per i test sul campo.

3.4 Backend

Il backend di Aegis agisce come vero e proprio motore forense del sistema. È qui che risiede la responsabilità dell'intero processo di analisi: dalla ricezione dell'immagine all'analisi steganografica, fino alla rilevazione del watermark e alla conseguente estrazione e attribuzione dei metadati.

Operativamente, questa logica è esposta tramite endpoint REST gestiti da Flask. Il sistema espone due rotte principali:

- `/apply`: riceve l'immagine pulita e restituisce quella firmata;
- `/verify`: analizza un'immagine sospetta per estrarre la firma.

La gestione delle richieste di watermarking è affidata al file `app.py`, che agisce da router smistando l'elaborazione verso la classe algoritmica appropriata, come mostrato nel Listing 1.

```python
@app.route('/apply', methods=['POST'])
def apply_watermark():
    # 1. Recupero parametri dalla richiesta (es. 'combo', 'lsb')
    technique = request.form.get('type', 'lsb')
    message = request.form.get('message', '')

    # 2. Lettura immagine da memoria
    file = request.files['image']
    # Decodifica stream di byte in matrice NumPy
    np_img = cv2.imdecode(np.frombuffer(file.read(), np.uint8), 1)

    # 3. Selezione strategia
    if technique == 'combo':
        algo = ComboWatermark()
    elif technique == 'dct':
        algo = DCTWatermark()
    else:
        algo = LSBWatermark()

    # 4. Processing e Risposta
    result_img = algo.embed(np_img, message)
    return send_file(result_img, mimetype='image/png')
```

Listing 1: Routing della richiesta di watermarking in Flask (app.py).

Per quanto riguarda la verifica, l'interfaccia amministrativa interroga il servizio inviando una richiesta POST. Ecco un esempio della chiamata client:

```javascript
fetch('http://127.0.0.1:5001/verify', {
    method: 'POST',
    body: formData
});
```

Listing 2: Chiamata client per la verifica (JavaScript).

La risposta fornita dal server non è un semplice feedback tecnico, ma assume valenza forense.

3.4.1 Pipeline forense

Il processo di verifica segue una pipeline rigorosa che può essere schematizzata nei seguenti passaggi:

1. Parsing e validazione del file immagine ricevuto.
2. Conversione e normalizzazione del formato per uniformare l'analisi.
3. Analisi parallela alla ricerca di watermark LSB e DCT.
4. Recupero della firma tramite scansione a griglia (Grid Search) e unione dei risultati.
5. Generazione dell'output investigativo finale.

3.5 L'Algoritmo proprietario "Aegis Combo" (DCT + Spatial Redundancy)

La strategia sviluppata, denominata "Aegis Combo", implementa un approccio a difesa profonda basato su due livelli complementari:

- **Livello Fragile (LSB):** garantisce l'integrità del dato originale. È estremamente veloce e permette di verificare se il file è bit-per-bit identico all'originale.
- **Livello Robusto (DCT con Grid Search):** garantisce la sopravvivenza della firma anche in caso di compressione, ridimensionamento o ritaglio (crop).

A differenza degli approcci classici che si affidano esclusivamente a codici di correzione errore su posizioni fisse, Aegis adotta una strategia di *Spatial Redundancy* e *Synchronization Search* per contrastare le trasformazioni geometriche:

1. **Tiling Infinito:** la firma (composta da mittente, destinatario e marker univoci `###`) viene ripetuta ciclicamente sull'intera superficie dell'immagine. Questo assicura che, anche se l'immagine viene ritagliata mantenendo solo una piccola porzione (crop), statisticamente almeno una copia completa della firma sarà presente nel frammento residuo.
2. **Grid Search (Sliding Window):** il problema principale del ritaglio è la perdita di sincronizzazione con la griglia 8x8 della DCT. Per risolvere questo problema, l'algoritmo di estrazione non effettua una singola lettura statica, ma esegue una scansione a forza bruta intelligente ("Grid Search"): vengono testati molteplici offset di allineamento, spostando la finestra di lettura pixel per pixel lungo gli assi X e Y, fino a individuare la sincronizzazione corretta che rivela il watermark.

L'implementazione concreta di questa strategia ibrida è visibile nella classe `ComboWatermark`. Come mostrato nel Listing 3, l'algoritmo applica sequenzialmente la trasformata DCT e successivamente la codifica LSB, creando un doppio livello di protezione.

```
class ComboWatermark:
    def __init__(self):
        self.lsb = LSBWatermark() # Livello Fragile
        self.dct = DCTWatermark() # Livello Robusto

    def embed(self, image, text):
        # 1. Applicazione Watermark Robusto (DCT)
        # Resiste a compressione e modifiche
        temp_img = self.dct.embed(image, text)

        # 2. Applicazione Watermark Fragile (LSB)
        # Permette la verifica di integrit  bit-exact
        final_img = self.lsb.embed(temp_img, text)

        return final_img
```

Listing 3: Metodo embed della classe ComboWatermark (algorithms.py).

Questa combinazione rende il watermark resistente anche a modifiche geometriche asimmetriche (come lo screenshot parziale), poiché l'algoritmo "cerca" attivamente la griglia corretta in cui la firma risulta leggibile, senza dipendere da coordinate assolute.

Un aspetto critico di questa fase, in un'ottica di *security-by-design*, riguarda la gestione dell'input: immagini malevole o malformate potrebbero infatti rappresentare un vettore d'attacco, rendendo necessaria una validazione stringente.

3.6 Frontend

Il frontend mobile, realizzato in Flutter, ha il compito di rendere accessibile il sistema mascherando la complessità sottostante, senza però esporre direttamente la logica forense. Attraverso schermate intuitive per il login, la dashboard, l'invio dell'immagine e la consultazione dei risultati, l'utente interagisce con il sistema in modo fluido.

La comunicazione con il server è mediata da un service layer dedicato (`api_service.dart`), che si occupa di inviare le richieste HTTP e gestire le risposte JSON. Sotto il profilo della sicurezza, tale architettura *thin client* offre un vantaggio strategico: poiché nessuna logica di estrazione del watermark risiede nel client, il backend rimane l'unica *fonte di verità* investigativa, riducendo drasticamente il rischio che il metodo forense possa subire **reverse engineering** analizzando l'applicazione mobile.

Il metodo `applyWatermark` si occupa di incapsulare l'immagine e i metadati in una richiesta `Multipart`, simulando l'invio di un form dati complesso.

```dart
Future<Uint8List?> applyWatermark(File img, String type, String msg) async {
    var uri = Uri.parse("$baseUrl/apply");
    var request = http.MultipartRequest('POST', uri);

    // Metadati: tipo algoritmo e messaggio segreto
    request.fields['type'] = type;
    request.fields['message'] = msg;

    // Stream dei byte dell'immagine
    request.files.add(await http.MultipartFile.fromPath(
      'image',
      img.path,
    ));

    var response = await request.send();
    // Se 200 OK, convertiamo lo stream di risposta in byte
    if (response.statusCode == 200) {
        return await response.stream.toBytes();
    }
    return null;
}
```

Listing 4: Invio dell'immagine tramite MultipartRequest in Dart.

3.7 Considerazioni di sicurezza implementativa

Benché il prototipo attuale sia funzionale alla sperimentazione, l'eventuale passaggio a un ambiente di produzione reale richiederebbe l'implementazione di controlli di sicurezza addizionali. Tra questi, risulterebbero prioritari l'autenticazione forte sugli endpoint (via JWT o OAuth2), meccanismi di rate limiting per prevenire **attacchi DoS**, una validazione ancora più rigorosa dei file caricati, la cifratura del traffico tramite HTTPS, la gestione di audit log per garantire la catena di custodia digitale e una corretta politica di privacy. Tali accorgimenti sono fondamentali affinché un sistema nato per proteggere contenuti sensibili non diventi, paradossalmente, un punto di vulnerabilità.

3.8 Flusso Utente e Interfaccia

L'applicazione mobile è stata progettata per guidare l'utente attraverso un flusso operativo sicuro e lineare, minimizzando le possibilità di errore umano.

Le funzionalità principali accessibili via interfaccia (UI) sono:

- **Autenticazione:** l'accesso è consentito solo previa registrazione e login validato dal server, garantendo che ogni operazione sia attribuibile a un'identità verificata (User Code).
- **Secure Sending:** l'utente seleziona un'immagine dalla galleria locale e specifica il destinatario. In questa fase, il sistema inibisce l'invio verso utenti inesistenti interrogando il database in tempo reale. È inoltre possibile selezionare la strategia di watermarking desiderata tramite un apposito selettore.
- **Inbox e Accettazione:** i destinatari visualizzano le immagini in arrivo in una sezione dedicata ("Pending"). In ottica di non ripudiabilità, il destinatario deve esplicitamente "Accettare" l'immagine per poterla scaricare; l'azione di accettazione applica una seconda firma digitale (quella del ricevente) sull'immagine, vincolando il file al destinatario.
- **Verifica:** una sezione dedicata permette l'upload di immagini sospette per verificare, tramite chiamata al backend forense, la presenza e l'integrità dei watermark applicati.

4 Fase di test

Nel contesto della cybersecurity e della digital forensics, la validazione non è un optional. A differenza delle applicazioni tradizionali, Aegis non deve limitarsi a funzionare in condizioni ideali, ma deve dimostrare affidabilità nella verifica del watermark, robustezza contro le trasformazioni reali e resilienza anche in scenari avversari, garantendo sempre la ripetibilità del risultato investigativo.

La strategia di testing adottata per Aegis si concentra su una **verifica funzionale** condotta tramite l'interfaccia amministrativa e test di robustezza.

4.1 Pagina di verifica forense

Per consentire un controllo diretto e manuale delle capacità investigative del sistema, Aegis include una **pagina amministrativa** dedicata alla verifica del watermark, implementata in `HTML/JavaScript`.

Questa interfaccia rappresenta un modulo essenziale in un contesto forense, poiché permette ad un analista umano di caricare contenuti sospetti e ottenere un responso immediato riguardo la presenza di firme nascoste.

La pagina consente:

- la selezione di un file immagine locale;
- l'invio dell'immagine al backend tramite richiesta HTTP POST;
- la visualizzazione dei risultati forensi estratti.

L'invocazione dell'endpoint di verifica avviene tramite il seguente meccanismo:

```
fetch('http://127.0.0.1:5001/verify', {
    method: 'POST',
    body: formData
});
```

Il backend risponde con un oggetto JSON contenente attributi di valore investigativo, tra cui:

- `technique`: metodologia di watermark, tra le tre disponibili, rilevata;
- `sender`: identificativo del soggetto mittente;
- `receiver`: identificativo del destinatario vittima;
- `raw`: firma grezza estratta.

Output Forense

Quando il sistema rileva una firma valida, genera un report in formato JSON strutturato (Listing 5), che costituisce l'evidenza digitale dell'attribuzione.

Listing 5: Esempio di payload JSON restituito durante la verifica forense.

```
{
  "status": "DETECTED",
  "timestamp": "2026-01-30T10:45:00Z",
  "evidence": {
    "technique": "DCT_Combo_GridSearch",
    "sender_id": "USR_0001",
    "receiver_id": "USR_0002",
    "confidence_score": 0.98
  },
  "raw_watermark": "USR_0001<->USR_0002###AegisSig"
}
```

Questo formato standardizzato permette l'integrazione dei risultati con eventuali altri strumenti di SIEM o di analisi investigativa automatizzata.

Questo output costituisce un elemento centrale del forensic watermarking, poiché consente di supportare l'attribuzione in scenari di **diffusione illecita**.

4.2 Testing di robustezza del watermark

Un watermark forense, per essere utile in ambito reale, deve risultare resistente non solo a manipolazioni accidentali, ma anche a tentativi intenzionali di rimozione da parte di un avversario.

Le principali trasformazioni considerate includono:

- compressione JPEG;
- resizing e downsampling;
- introduzione di rumore artificiale;
- filtri di smoothing e sharpening;
- cropping parziale dell'immagine.

In particolare, il sistema Aegis combina tecniche differenti proprio per garantire un trade-off tra `invisibilità del watermark`, `robustezza a trasformazioni comuni` e `resilienza a manipolazioni avversarie`.

L'approccio DCT risulta intrinsecamente più robusto rispetto a LSB, poiché opera nel dominio delle frequenze e sopravvive a molte trasformazioni di compressione.

Inoltre, l'utilizzo della **ridondanza spaziale** (Tiling) consente di recuperare firme anche in presenza di ritagli significativi (cropping), aumentando la probabilità di estrazione corretta anche da porzioni parziali dell'immagine. Di seguito si riporta una sintesi dei risultati sperimentali ottenuti sottoponendo le immagini marcate a diversi stress test:

Tipo di Attacco	**Strategia LSB**	**Strategia DCT (Aegis Combo)**
Nessuna modifica	**Rilevato**	**Rilevato**
Compressione JPEG (90%)	Perso	**Rilevato**
Compressione JPEG (70%)	Perso	**Rilevato**
Ritarglio (Crop 30%)	Perso	**Rilevato (Grid Search)**
Rumore (Noise)	Perso	**Rilevato**
Screenshot (Scaling)	Perso	Non Rilevato (v. Limiti)

Tabella 1: Matrice di robustezza delle tecniche implementate.

5 Conclusioni

Il progetto Aegis si configura come una risposta concreta alle sfide poste dalla diffusione non autorizzata di immagini sensibili, coniugando i principi della cybersecurity con le metodologie della digital forensics. L'obiettivo non si limita alla semplice protezione del dato, ma si estende alla creazione di un ecosistema capace di **scoraggiare abusi e facilitare le indagini**.

Attraverso l'adozione di tecniche avanzate di forensic watermarking, il sistema trascende la marcatura statica dei contenuti digitali per abilitare una reale attribuzione investigativa. Le proprietà fondamentali introdotte dal modello includono:

- tracciabilità della catena di distribuzione;
- non ripudiabilità delle azioni di invio e ricezione;
- robustezza del marchio rispetto alle manipolazioni più comuni.

Dal punto di vista tecnico, il valore aggiunto del progetto risiede nella combinazione di approcci complementari come LSB e DCT. L'integrazione di queste tecniche con strategie di sincronizzazione spaziale (Grid Search) rappresenta un significativo passo avanti rispetto ai tradizionali sistemi di watermarking statico.

5.1 Limiti sperimentali: il problema della Desincronizzazione di Scala

Sebbene l'implementazione attuale di Aegis dimostri una notevole resilienza contro il ritaglio puro (*cropping*) grazie alla tecnica del Tiling e della Grid Search traslazionale, i test sul campo hanno evidenziato criticità nel recupero della firma da **screenshot** o fotografie di schermi. La causa risiede nella natura rigida della trasformata DCT (8×8 pixel): quando un'immagine viene "screenshottata" o zoomata, subisce un ricampionamento che altera la dimensione dei blocchi. Attualmente, l'algoritmo esplora solo la traslazione (offset X/Y) ma non la scala, per mantenere performance in tempo reale.

5.2 Sviluppi futuri

In coerenza con la visione progettuale iniziale, il percorso evolutivo di Aegis prevede l'implementazione di strategie avanzate di difesa a strati non ancora integrate nel prototipo attuale:

- **Prevenzione attiva (DRM e Policy):** sviluppo di un meccanismo di "visualizzazione effimera" lato client. Le immagini non verrebbero salvate come file standard nella galleria, ma gestite come contenitori crittografati visualizzabili solo *in-app* per un tempo limitato (modello "one-time view"), con blocco nativo degli screenshot tramite le API `FLAG_SECURE` di Android/iOS.

- **Evoluzione in "AIgis" (Deep Learning):** per superare i limiti della DCT evidenziati (scaling), si prevede l'integrazione di reti neurali (CNN) addestrate per riconoscere il watermark come feature visiva persistente, permettendo l'estrazione anche da foto angolate o deformate.

- **Scansione Proattiva (OSINT):** implementazione di agenti autonomi di ricerca (crawler) incaricati di scansionare fonti pubbliche e repository online per rilevare copie non autorizzate delle immagini protette, allertando automaticamente l'utente vittima (Rilevamento e Risposta).

- **Autenticazione Forte:** integrazione di firme crittografiche asimmetriche per garantire che ogni watermark sia inconfutabilmente legato all'identità digitale del mittente (Non ripudiabilità forte).

In prospettiva, Aegis possiede il potenziale per evolvere da **proof-of-concept** a **strumento operativo completo**, offrendo un modello di cybersecurity forense applicata che sia al contempo efficace, etico e tecnicamente riproducibile.

Riferimenti bibliografici

[1] N. Ahmed, T. Natarajan e K.R. Rao. «Discrete Cosine Transform». In: *IEEE Transactions on Computers* C-23.1 (1974), pp. 90–93. DOI: 10.1109/T-C.1974.223784.

[2] Mauro Barni et al. «A DCT-domain system for robust image watermarking». In: *Signal processing* 66.3 (1998), pp. 357–372.

[3] Ingemar Cox et al. «Digital watermarking». In: *Journal of Electronic Imaging* 11.3 (2002), pp. 414–414.

[4] Ingemar J Cox et al. «Secure spread spectrum watermarking for multimedia». In: *IEEE transactions on image processing* 6.12 (1997), pp. 1673–1687.

[5] Jiri Fridrich. «Applications of data hiding in digital images». In: *ISSPA'99. Proceedings of the Fifth International Symposium on Signal Processing and its Applications (IEEE Cat. No. 99EX359)*. Vol. 1. IEEE. 1999, 9–vol.

[6] Deshpande Neeta, Kamalapur Snehal e Daisy Jacobs. «Implementation of LSB Steganography and Its Evaluation for Various Bits». In: *2006 1st International Conference on Digital Information Management*. 2007, pp. 173–178. DOI: 10.1109/ICDIM.2007.369349.

[7] R. Pickholtz, D. Schilling e L. Milstein. «Theory of Spread-Spectrum Communications - A Tutorial». In: *IEEE Transactions on Communications* 30.5 (1982), pp. 855–884. DOI: 10.1109/TCOM.1982.1095533.