Segurança de Dados

Ano letivo 2025/26

Daniel Andrade
PG60242

João Fonseca
PG59787

Pedro Malainho
PG61005

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Dezembro 2025

Segurança de Dados

Ano letivo 2025/26

Relatório do Trabalho Prático
Mestrado em Cibersegurança

Coordenador João Marco Cardoso Silva

Dezembro 2025

Abstract

This report details the design, implementation, and evaluation of a Hybrid Client-Server Architecture for a secure, privacy-preserving auction system.

The system is engineered to satisfy critical security requirements, achieved through a trust-minimized model where the central Distribution/Messaging Server (DMS) acts only as a secure broker. We employ applied cryptographic techniques — specifically ECDSA Digital Signatures for authenticity and ECIES Encryption for confidentiality — to enforce:

- Anonymity of participants via a Dual-Pseudonym System.
- Integrity and Non-Repudiation of bids.
- Trusted Timestamping provided by an external authority.
- Selective Identity Disclosure to ensure identities remain concealed until the private post-auction phase.

The report details the system architecture, communication protocols, and the operational workflow, encompassing auction listing, bid submission, and winner identity revelation. We conclude with a security analysis demonstrating that the design successfully ensures all sensitive information remains protected, notably achieving Perfect Forward Secrecy (PFS), while isolating key weaknesses for future development.

Contents

List of Tables

1. Introduction

1. 1. Problem Statement

The primary challenge of this project is to design a peer-to-peer (P2P), privacy-preserving auction system that integrates applied cryptography to satisfy the following stringent security and operational requirements:

- **Anonymity**: Seller and bidder identities remain confidential throughout the auction process
- **Authenticity**: Verification that bids originate from legitimate system participants
- **Integrity**: Protection against bid tampering or modification
- **Non-repudiation**: Winning bidders cannot deny their bids
- **Trusted timestamping**: Verifiable timestamps to resolve ties (earliest bid wins)
- **Selective identity disclosure**: Mutual identity revelation between seller and winner only

These requirements must be satisfied within a robust **Hybrid Client-Server Architecture**. This system relies on a central **Distribution/Messaging Server** to handle essential tasks of peer discovery and the reliable broadcast of all auction data to connected clients. Crucially, the system also integrates a dedicated **Trusted Service** running on a separate server.

1. 2. Project Objectives

The objectives of this project are to:

- Design a peer-to-peer auction architecture that fulfills the specified security requirements.
- Implement cryptographic protocols to enforce participant anonymity, authenticity, data integrity, and selective identity disclosure.
- Develop a functional prototype, including a user interface, to demonstrate system's operation.

1. 3. Report Structure

The report is organized as follows:

- Chapter 2: System Architecture and Data Flow
- Chapter 3: Cryptographic and Security Protocols
- Chapter 4: System Implementation and Prototyping
- Chapter 5: Conclusion

2. System Architecture and Data Flow

2. 1. Hybrid Client-Server Architecture

The system is built on a **Hybrid Client-Server Architecture** featuring a central **DMS**, **Client** nodes , and a single, unified external **Trusted Service**.

2. 2. Architectural Components and Roles

2. 2. 1. Client Nodes

The Client Node is the sole point of execution for all auction participants (as a seller or bidder) and, critically, is responsible for **enforcing all cryptographic privacy and integrity** protocols.

- **Peer Authentication**: Submits a JWT and a signed certificate from the Trusted Service upon connection.
- **Security Protocol Flow**: Before any action (such as placing a bid) is sent, the Client Node executes a rigid protocol:
    1. **Trusted Ordering**: Obtains a verifiable timestamp token from the external service.
    2. **Authenticity/Integrity**: Signs the raw command with **ECDSA**.
    3. **Confidentiality**: Encrypts the payload using **ECIES** for transmission.

2. 2. 2. Distribution/Messaging Server

The DMS operates with a principle of minimal trust. It acts as a secure, stateless message broker, managing peer connections and broadcasting encrypted auction data without ever inspecting the confidential contents.

- **Peer Authentication**: Verifies client JWTs/Certificates upon connection using the CA's public key, which is dynamically fetched from a JWK endpoint. The DMS retrieves and caches the JWK-set, extracts the appropriate public key, and uses it to validate signatures without storing any long-term trust state.
- **Cryptographic Verification**: Verifies the client's **ECDSA signature** (for authenticity) and the Trusted Timestamp signature (for time integrity) on every incoming command.
- **Auction Monitoring**: Tracks auction closing dates and determines the winning anonymous commitment based on the highest committed bid value and the trusted timestamp.

2. 2. 3. Trusted Service

The Trusted Service is the system's single external source of truth, providing two non-repudiable assurances: identity and time.

- **Certification Authority**: Signs Certificate for Clients Nodes, cryptographically linking a user's authenticated to their public key fingerprint (establishing **user legitimacy**).
- **Trusted Timestamp Authority**: Provides verifiable timestamps signed by its private key, guaranteeing trusted **tie-breaking** and **ordering of events**.

3. Security Requisites and Cryptographic Protocols

This chapter details the specific cryptographic tools and protocols implemented in the system to meet the required security properties, relying on **Elliptic Curve Cryptography** (ECC) primitives.

3. 1. Anonymity

The system achieves robust anonymity by employing a Dual-Pseudonym System, ensuring that the real-world identity is shielded by two layers of separation.

Requirement	Internal Mechanism	Broadcast Mechanism
Anonymity	Cryptographic Identifier: Permanent, non-repudiable ID used for internal state and final disclosure.	Transient Pseudonym (UUID): A temporary, unlinked name used only for public broadcast to obscure the seller/bidder.

Table 1: Anonymity Requirement

Pseudonym Generation and Use

The system uses identifiers for distinct purposes:

1. **Permanent Identifier**: This is the SHA256 fingerprint of the user's public key. It is used internally by the DMS to:
   - Bind the winning bid to a specific user for non-repudiation.
   - Perform the final identity lookup at auction closure.
2. **Transient Identifier**: This is a unique, randomly generated UUID that is created for each list/bid command. It is used exclusively in the public broadcast to notify other clients.

This separation ensures that while the DMS maintains a cryptographically secure link to the winning peer, no external entity can link the announced activity to the peer's permanent identity, thus strengthening overall anonymity.

3. 2. Confidentiality, Authenticity, Integrity, and Non-Repudiation

These four security requirements are enforced through a hybrid cryptographic protocol executed within the Client Node's send_command function.

Requirement	Mechanism	Cryptographic Justification
Confidentiality	ECIES with AES-GCM	Ensures the command payload is secured end-to-end between the client and the DMS, preventing eavesdropping.

Table 2: Confidentiality Requirement

Requirement	Mechanism	Cryptographic Justification
Authenticity	ECDSA Digital Signatures	Verification of the signature ensures that the command was created by the legitimate peer.

Table 3: Authenticity Requirement

Requirement	Mechanism	Cryptographic Justification
Integrity	ECDSA Signing & AES-GCM Authentication Tag	Dual Verification: the ECDSA signature proves the raw command was unaltered, and the AES-GCM tag proves the encrypted content was not modified.

Table 4: Integrity Requirement

Requirement	Mechanism	Cryptographic Justification
Non-repudiation	Uniqueness of Private Key	A verified ECDSA signature serves as undeniable proof that the authorized user initiated the command and cannot later deny it.

Table 5: Non-Repudiation Requirement

Hybrid Protocol Flow

1. **Confidentiality**: The client generated an **ephemeral key pair** and uses **ECDH** with the DMS's public key to derive a symmetric key (via **HKDF**). The payload is then encrypted usign **AES-256** in **GCM mode** (providing ecnryption and an Authentication Tag).
2. **Authenticity & Non-repudiation**: Separately from encryption, the client signs only the raw command string using its permanent private key with **ECDSA-SHA256**. The DMS verifies the signature after decryption, linking the action to the authenticated user.

3. 3. Trusted Timestamping and Time Integrity

The system uses the external TSA to provide verifiable time tokens for irrefutable event ordering.

Requirement	Mechanism	Cryptographic Justification
Trusted Timestamping	TSA Digital Signature	The time value is signed by the TSA's private key, guaranteeing that the time is authentic and accurate at the moment the TSA processed the request.

Table 6: Trusted Timestamping Requirement

The DMS verifies the signature using the TSA's pre-loaded public key to establish a reliable basis for all chronological events, such as bid tie-breaking.

3. 4. Selective Identity Disclosure

This protocol guarantees that only the seller and the winning bidder access each other's full identities, and only after the auction closes.

- Mechanism: The DMS's auction_monitor thread identifies the anonymous winner and seller, creates a unique, transient private SocketIO room, and moves only those two parties into it.
- Protocol: The DMS then emits a private notification, facilitation the exchange of their identities username.

4. System Implementation and Prototyping

This chapter details the system flow, focusing on the state management logic within the DMS.

4. 1. Client Authentication and Authorization

The system enforces a multi-step authentication process, with the Trusted Service acting as the controlling CA.

X.509 Certificate Generation:

The X.509 certificate acts as the initial proof of identity binding. The client submits only its public key to the CA's /api/sign-csr endpoint.

- **TS Role**: The Trusted Servive uses the client's provided public key to dynamically build a new certificate.

- **Subject Control**: The CA hardcodes the Subject details (e.g., Commom Name: gruposete.mcs.uminho.pt) into the certificate, ensuring that all peer certificates conform to a single, predefined organizational standard.

- **Trust Establishment**: The CA signs this newly constructed certificate, binding the clients public key to the controlled Subject details and establishing verifiable trust.

4. 1. 1. JWT creation and Identity Provisioning

The JWT is the **session authorization token**. The client sends its newly signed certificate to the CA's /api/issue endpoint. The CA:

- Extracts the public key from the validated certificate and calculates the anonymous peer_sub (SHA256 fingerprint).
- Issues a JWT containing the peer_sub in the sub claim, signed by the CA's private key. The client uses this token to connect and authenticate with the DMS.

4. 1. 2. JWK and JWT Validation

The DMS validates the client's identity using the JWK standard. The DMS retrieves the CA public key from the well-known/ca-public endpoint. Upon connection, the DMS uses this public key to perform a two-layer verification:

1. **JWT Validation (Authorization)**: The DMS uses the CA Public Key to verify the JWT's signature, confirming the token's authenticity and extracting the client's peer_sub identifier.
2. **Certificate Validation (Identity Binding)**: The DMS verifies the signature on the X.509 certificate using the same CA Public Key, ensuring the client's public key is legitimately bound to the CA's trust chain.

4. 2. Auction State Management

The DMS manages the entire auction state in a dictionary named auction_items.

- **Concurrency Control**: A threading.Lock() object (items_lock) is utilized to ensure thread-safe access and modification of the shared auction_items structure, preventing race conditions during concurrent bid placements.
- **Anonymous Data Structure**: Each auction item is an instance of the Item dataclass, which tracks its status using only anonymous identifiers: **seller** (the peer_sub of the listing user) and **buyer** (the peer_sub of the highest bidder).

4. 3. Listing an Item

The ListCommand manages the creation of a new auction item.

- **Command Reception**: The DMS decrypts and verifies the ECDSA signature of the /list command.
- **Item Creation**: A new Item object is instantiated.
  - The anonymous peer_sub of the client is saved as the seller.
  - The **verified trusted timestamp** supplied with the command is recorded as the listing_timestamp.
  - The specified closing_date is parsed and stored.
- **Broadcasting**: The new item is added to auction_items under a generated unique ID, and a someone_listed event containing the anonymous item details is broadcast to all connected clients via SocketIO.

4. 4. Placing a bid

The BidCommand handles the core bidding logic, incorporating security checks.

- **Verification Checks**: Before updating the state, the DMS performs three critical checks:
  - **Existence**: Verifies the item ID exists in auction_items.
  - **Value**: Calls item.is_valid_bid() to ensure the new bid_value is higher or equal than the current item.highest_bid or item.minimum_bid.

- **Time Integrity and Tie Resolution**: The DMS performs two chronological checks using the verified trusted timestamp ts_obj from the TSA:

1. **Anti-Retroactive Check**: It verifies that ts_obj is newer than the item's listing_timestamp (item_ts < ts_obj). This is the base check to prevent retroactive bids.

2. Tie Resolution Rule: If an incoming bid's value is equal to the current item.highest_bid, the existing bid is retained, as the winning bid is always the one with the older (earlier) verified trusted timestamp. A newer bid cannot overturn an existing bid of the same value.

- **State Update**: If all checks pass, the following fields are updated under the mutual exclusion lock:
  - item.buyer is updated to the bidder's anonymous peer_sub.
  - item.highest_bid is updated to the new value.
  - The bid's **trusted timestamp** is appended to item.biding_timestamp.
- **Broadcast**: An anonymous someone_bid event is broadcast to all clients, allowing them to update their local view of the item's highest bid.

4. 5. Auction Closure and Identity Disclosure

The Selective Identity Disclosure protocol is managed by the auction_monitor thread.

- **Closure Determination**: The auction_monitor periodically checks the trusted timestamp from the CA against the closing_date of all active items.
- **Identification**: Once an auction closes, the DMS uses the stored anonymous IDs (item.seller and item.buyer) to look up the corresponding client's session information in the clients dictionary.
- **Private Room Creation**: The DMS dynamically creates a unique, private SocketIO room (auction_<item_name>) and moves the winning seller's and buyer's SIDs into this room.
- **Notification**: A private auction_closed notification is emitted only to the members of this private room. At this stage, the DMS facilitates the exchange of the respective users' username, fulfilling the selective disclosure requirement.

4. 6. Utility Commands and Time Simulation

The system provides utility commands for querying state and specialized API endpoints for testing auction closure logic.

4. 6. 1. Utility and Query Commands

The DMS implements simple commands that read the auction_items state:

- /allitems (AllItemsCommand): Retrieves and sends a dictionary of all active auction items to the requesting client.
- /myitems (MyItemsCommand): Filters auction_items to return only those where the item.seller matches the requesting client's authenticated peer_sub, demonstrating the use of the anonymous identifier for state retrieval.

4. 6. 2. Time Simulation Commands

The Trusted Service (app.py) includes API endpoints to facilitate testing of the time-sensitive auction logic:

- /api/settime: Allows a user to set a simulated date (e.g., fast-forwarding to test closing auctions)
- /api/resettime: Resets the simulation, causing the /api/timestamp endpoint to return the true current date and time.

5. Security Analysis and Evaluation

This chapter validates the system's compliance with established security requirements and assesses the performance overhead introduced by the complex cryptographic protocols.

5. 1. Cryptographic Security Analysis

- **Confidentiality and Perfect Forward Secrecy**

The use of **ECIES** with ephemeral key pairs ensures robust **Confidentiality**. Critically, the use of ephemeral keys achieves **PFS**. This means that if the DMS's long-term private key were ever compromised in the future, past transaction payloads would remain secure, as the unique, per-transaction ephemeral session keys are no longer available for decryption.

- **Trusted Timestamping**

Time integrity is guaranteed by externalizing the time source to the TSA. The DMS's verification of the TSA's digital signature using the public key ensures that the time used for critical functions (such as tie-breaking and auction closure) is authentic and immutable, effectively preventing time-based attacks.

- **Integrity and Non-Repudiation**

1. **Integrity**: The ECDSA signature proves the raw command was unaltered. Furthermore, the AES-GCM Authentication Tag (generated during encryption) provides a strong cryptographic assurance that the encrypted data was not modified during transit.

2. **Non-Repudiation**: The client's ECDSA signature provides undeniable proof that the user — the sole holder of the associated private key — authorized the raw command.

5. 2. Performance Evaluation

The use of ECC is efficient, but the multi-step, sequential protocol introduces measurable overhead.

Latency Overhead

Every client command incurs high latency due to the sequential, required steps:

- **TSA Request**: Network latency associated with retrieving the external trusted timestamp is the **primary expected bottleneck**.

- **Cryptography**: Sequential execution of ECDSA Signing and ECIES Key Exchange/Encryption.

DMS Throughput

The system's maximum throughput is limited by the computational load on the DMS. Each incoming command requires performing three computationally intensive checks:

1. ECIES Decryption (to retrieve the command).

2. Client ECDSA Signature Verification (for authenticity).

3. TSA Signature Verification. This load is compounded by the use of a mutual exclusion lock (items_lock), which serializes state transitions to preserve data integrity, further limiting concurrency under heavy bidding load.

5. 3. Limitations and Future Work

System Limitations

- **Single Point of Trust** (TSA): The Trusted Service is a centralized point of failure. Its compromise would invalidate all identity and time assurances, and its failure would halt new client certification and time integrity checks.
- **Resource Exhaustion Attack**: The DMS's requirement to perform computationally expensive signature verification on every message makes it susceptible to a Denial-of-Service (DoS) attack, where an attacker floods the server with invalid signatures, consuming excessive CPU cycles.

Proposed Future Enhancements

- **Decentralized Time Source**: The TSA role could be migrated to a decentralized system, such as a public **blockchain** or a **Verifiable Delay Function** (VDF), to distribute trust and remove the single point of failure.
- **Zero-Knowledge Proofs** (ZKP): Integrating ZKPs would allow the DMS to mathematically verify a bid's validity (e.g., bid is greater than current highest bid) without having to decrypt the actual bid value. This would maximize confidentiality even from the server.

6. Conclusion

The implemented Hybrid Client-Server Auction System successfully validates the feasibility of conducting secure, anonymous transactions in a distributed environment while minimizing the trust required of the central messaging server (DMS).

The design rigorously addresses core security requirements through the strategic use of Elliptic Curve Cryptography (ECC):

- **Integrity and Non-Repudiation** are guaranteed by mandatory **ECDSA signatures** on every command.
- **Confidentiality and Perfect Forward Secrecy (PFS)** are enforced via **ECIES encryption** using ephemeral keys, ensuring data security even if the DMS's long-term key is compromised.
- **Anonymity** is achieved through a **Dual-Pseudonym System**, using a permanent, cryptographic peer_sub for internal accountability and a transient, random pseudonym for public broadcasts, effectively preventing external activity correlation.

The system's reliance on the **Trusted Service** (TS) for verifiable identity and time integrity is a powerful feature, but it introduces the central limitation of a single point of trust and the primary **network latency bottleneck** via the TSA request. Future work, particularly the integration of **Zero-Knowledge Proofs** (ZKP) for bid validation and the migration of the time service to a decentralized authority, will further reduce the trust burden on the central components and enhance system resilience, paving the way for truly trust-minimized, confidential auction platforms.

7. Acronyms

AES-GCM: Advanced Encryption Standard - Galois/Counter Mode

API: Application Program Interface

CA: Certificate Authority

CPU: Central Processing Unit

DMS: Distribution/Messaging Server

ECC: Elliptic Curve Cryptography

ECDH: Elliptic Curve Diffie-Hellman

ECDSA: Elliptic Curve Digital Signature Algorithm

ECIES: Elliptic Curve Integrated Encryption Scheme

HKDF: HMAC-based Key Derivation Function

JWT: JSON Web Token

JWK: JSON Web Key

P2P: Peer-to-Peer

PFS: Perfect Forward Secrecy

SID: Session ID

TS: Trusted Service

TSA: Time-Stamping Authority

UUID: Universally Unique Identifier

VDF: Verifiable Delay Function

ZKP: Zero-Knowledge Proofs