E2EDPS: An End-to-End Data Protection System

July Yin

july-uncurled-yin@duck.com

Abstract. In the age of digital communication, data privacy and security have become paramount concerns. This paper presents the End-to-End Data Protection System (E2EDPS), a novel protocol that leverages elliptic curve cryptography (ECC) to ensure the confidentiality and integrity of user data. By employing a unique private key-based authentication mechanism and a dual encryption strategy, E2EDPS aims to provide robust protection against unauthorized access, even in scenarios involving third-party intermediaries or cyber-criminals that found a way to access the server's database. The system is designed to be **100% strict censorship resistant**, ensuring that users can maintain control over their data without interference.

1. Introduction

The increasing reliance on cloud services and web applications has raised significant concerns regarding data privacy and security. Traditional authentication methods often expose sensitive information to potential breaches, making users vulnerable to data exploitation and censorship. E2EDPS addresses these challenges by implementing a system where user data is encrypted on the client-side before transmission, ensuring that it remains secure throughout its lifecycle.

2. Motivation

The motivation behind E2EDPS stems from the need for a secure communication framework that protects user data from unauthorized access, especially in environments where third-party services are involved. By utilizing a 64-character private key for authentication and data encryption, E2EDPS provides a streamlined approach to secure data handling. The system's **100% strict censorship resistance** ensures that users can freely express themselves and access information without fear of interference or suppression because there

services are unable to identify data to censure.

- To develop a protocol that ensures end-to-end encryption of user data.
- To implement a secure authentication mechanism based on a private key login system.
- To take profit of the user key pairs to encrypt his data in the database.
- To always encrypt and decrypt user data client-side only (browser).
- To evaluate the effectiveness and efficiency of the proposed system in real-world scenarios.
- To provide a censorship-resistant platform that empowers users to maintain control over their data.

E2EDPS is designed around a client-server architecture where the client is responsible for encrypting data before sending it to the server. The server manages the encrypted data without ever accessing it in plaintext.

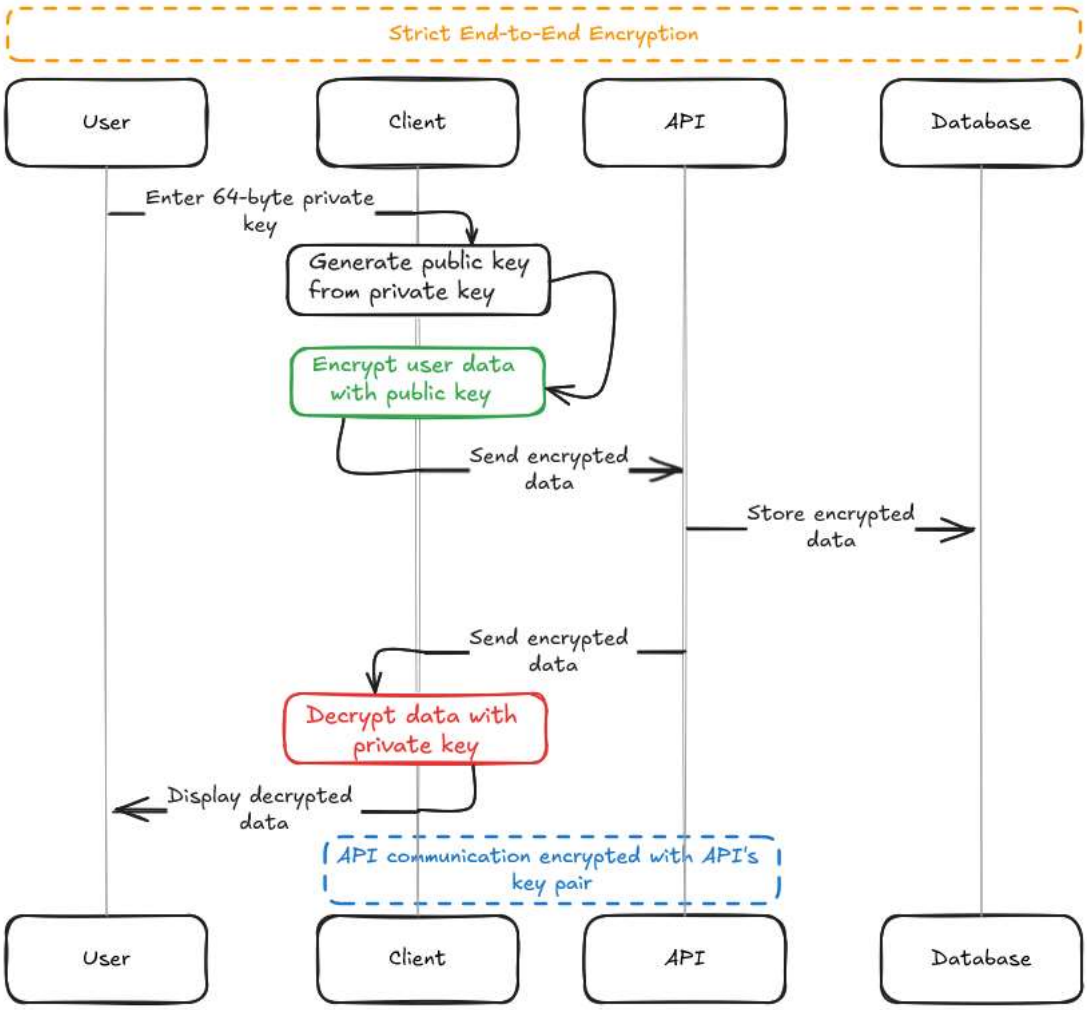
3. Keys Generation

The system generates a random private key, which serves as the foundation for all cryptographic operations. This key is unique to each user and is used for both signing messages and decrypting data. The private key is generated using a secure random number generator, ensuring that it is unpredictable and unique.

From the private key, a corresponding public key is derived using the secp256k1 elliptic curve on the browser side. When a user logs in, the browser sends the hashed version of the private key (hashed client-side) to the server, ensuring that the raw private key is never transmitted. The server checks if this hashed private key exists in its database to verify user existence and responds with its own public key.

Data Encryption and Decryption

Data is encrypted on the client-side using the public key before transmission. Upon receiving the data, the server stores it securely without decrypting it. The user's private key is also utilized to encrypt sensitive information such as their name or any other personal data before it is stored in the database. This information can later be decrypted client-side when needed.



4. Keys Generation

Login Transaction Flow

When a user attempts to log in, they provide their private key, which undergoes hashing on the client side to create a secure representation that can be sent to the server without exposing the actual key. This process ensures that even if an attacker intercepts this request, they cannot retrieve the actual private key.

Request Structure

```
{
    // other fields here
    "e2edps": {
        "clientPrivateKeyHash": "128 chars",
        "clientPublicKey": "130 chars"
    }
}
```

The client-side hashing of the private key adds an extra layer of security, as only hashed values are stored on the server. This mechanism significantly reduces the risk of key compromise and unauthorized access.

Response Structure

```
{
    // other fields here
    "token": "...", // example token
    "e2edps": {
        "serverPublicKey": "130 chars"
    }
}
```

Upon successful verification, the server responds with a session token and its own public key. This public key is used for encrypting subsequent API requests, ensuring that all communication remains secure.

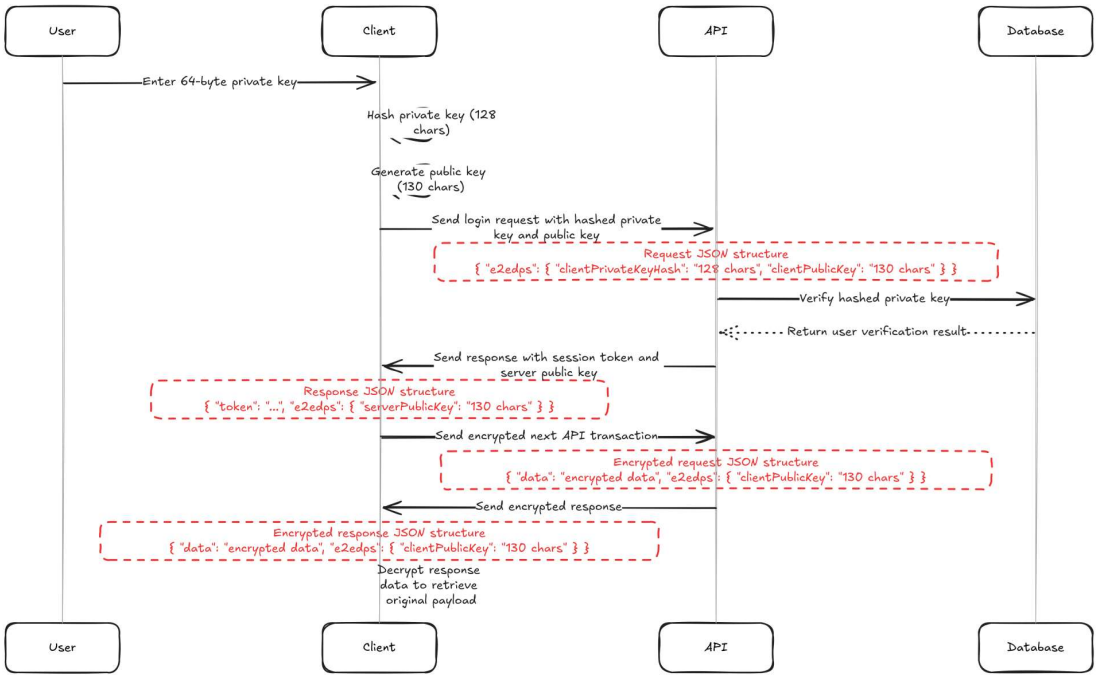
Encrypted API Transactions

Once logged in, subsequent API requests are also secured using encryption:

Encrypted Transaction Structure

```
{
    "data": "encrypted data",
    "e2edps": {
        "clientPublicKey": "130 chars" // or save in session
    }
}
```

The client's public key is either included in the request or stored in the server session, allowing the server to encrypt its responses. After decrypting this data on the client side, users will find their original JSON payload intact.



5. Security measures

E2EDPS employs several security measures to ensure data protection:

End-to-End Encryption

By encrypting data on the client-side, E2EDPS ensures that sensitive information is never exposed to the server or any third-party services. This approach prevents unauthorized access and maintains the confidentiality of user data throughout its lifecycle.

Key Management

The use of a unique private key for each user minimizes the risk of key compromise. Additionally, hashing of the private key adds an extra layer of security since only hashed values are stored on the server. This design ensures that even if the server is compromised, the actual private keys remain protected.

Robust Authentication

The implementation of a private key login system with a strong 64-character private key serves as an effective authentication mechanism, making it difficult for attackers to gain unauthorized access. The hashing of the private key before transmission further enhances the security of the authentication process.

Protection Against Reverse Proxies

A significant advantage of E2EDPS is its ability to mitigate risks associated with reverse proxies like Cloudflare. These services often decrypt incoming traffic before re-encrypting it for backend servers, potentially exposing sensitive information in clear text during this process. However, with E2EDPS:

- The raw private keys are never sent over the network.
- Data remains encrypted until it reaches its intended recipient (the client), thus preventing reverse proxies from accessing sensitive information during transmission.
- Even if decrypted at any point by intermediaries, they only see hashed representations or encrypted payloads rather than clear text.

This design effectively enhances privacy and security against potential vulnerabilities introduced by third-party services.

Censorship Resistance

E2EDPS is designed to be **100% strict censorship resistant**, ensuring that users can freely express themselves and access information without fear of interference or suppression. By encrypting data on the client-side and maintaining control over their private keys, users can bypass censorship attempts and maintain their right to free speech and information access.

6. Conclusion

The End-to-End Data Protection System (E2EDPS) represents a significant advancement in secure data handling practices. By integrating elliptic curve cryptography and a unique private key-based authentication system, E2EDPS provides a robust framework for protecting user data against unauthorized access while ensuring that all communications are encrypted end-to-end. The system's **100% strict censorship resistance** empowers users to maintain control over their data and freely express themselves without fear of interference.

Future work will focus on optimizing system performance and exploring additional cryptographic techniques to enhance security. The team is committed to continuously improving E2EDPS to meet the evolving needs of users in the digital age.

References

[1] Boneh, D., & Shoup, V. (2017). *A Graduate Course in Applied Cryptography*.

[2] NIST. (2019). *Recommendation for Key Management: Part 1 - General*. NIST Special Publication 800-57.

[3] Rivest, R., Shamir, A., & Adleman, L. (1978). *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM.

[4] Elliptic Curve Digital Signature Algorithm (ECDSA). (n.d.). *Cryptography Research Group, University of California, San Diego*.

[5] Diffie, W., & Hellman, M. (1976). *New Directions in Cryptography*. IEEE Transactions on Information Theory.