SIRS Project Presentation

Restaurants & Tourism: BombAppetit

Daniel Pereira, 99194 | Ricardo Toscanelli, 99315 | Simão Gato, 99328
Group 37

Secure Document Format

The secure document includes three operations:

- protect ( )
- check ( )
- unprotect ( )

Assumption: User and service share their respective public keys.

Note: For the implementation of the Secure Document Library, we used java + maven.

Secure Document Format: protect ( )

1. We generate a secure symmetric key using AES (Advanced Encryption Standard) with a robust 256-bit key size.
2. The voucher information is encrypted using AES in GCM mode, ensuring both confidentiality and integrity, through authenticated encryption.
3. The symmetric key is encrypted with the user's public key (2048-bit key size) using RSA with SHA256 for added security.
4. A nonce, combining a timestamp and a random number (in our case, we use the IV from the GCM mode), prevents replay attacks.
5. The entire document is signed with the restaurant owner's private key for authenticity (digital signature).

Secure Document Format: protect ( ) result


```
{
  "restaurantInfo": {
    "owner": "Maria Silva",
    "restaurant": "Dona Maria",
    "address": "Rua da Gl??ria, 22, Lisboa",
    "genre": ["Portuguese", "Traditional"],
    "menu": [
      {
        "itemName": "House Steak",
        "category": "Meat",
        "description": "A succulent sirloin grilled steak.",
        "price": 24.99,
        "currency": "EUR"
      },
      {
        "itemName": "Sardines",
        "category": "Fish",
        "description": "A Portuguese staple, accompanied by potatoes and salad.",
        "price": 21.99,
        "currency": "EUR"
      },
      {
        "itemName": "Mushroom Risotto",
        "category": "Vegetarian",
        "description": "Creamy Arborio rice cooked with assorted mushrooms and Parmesan cheese.",
        "price": 16.99,
        "currency": "EUR"
      }
    ],
    "mealVoucher": {
      "encryptedVoucher": "<...>",
      "encryptedSymmetricKey": "<...>",
      "nonce": "<...>"
    }
  },
  "signature": "<...>"
}
```

Secure Document Format: check ( )

1. Nonce verification prevents replay attacks.
2. Digital signature verification ensures the document's authenticity.
3. A boolean status indicates success or failure of verification.

Secure Document Format: unprotect ( )

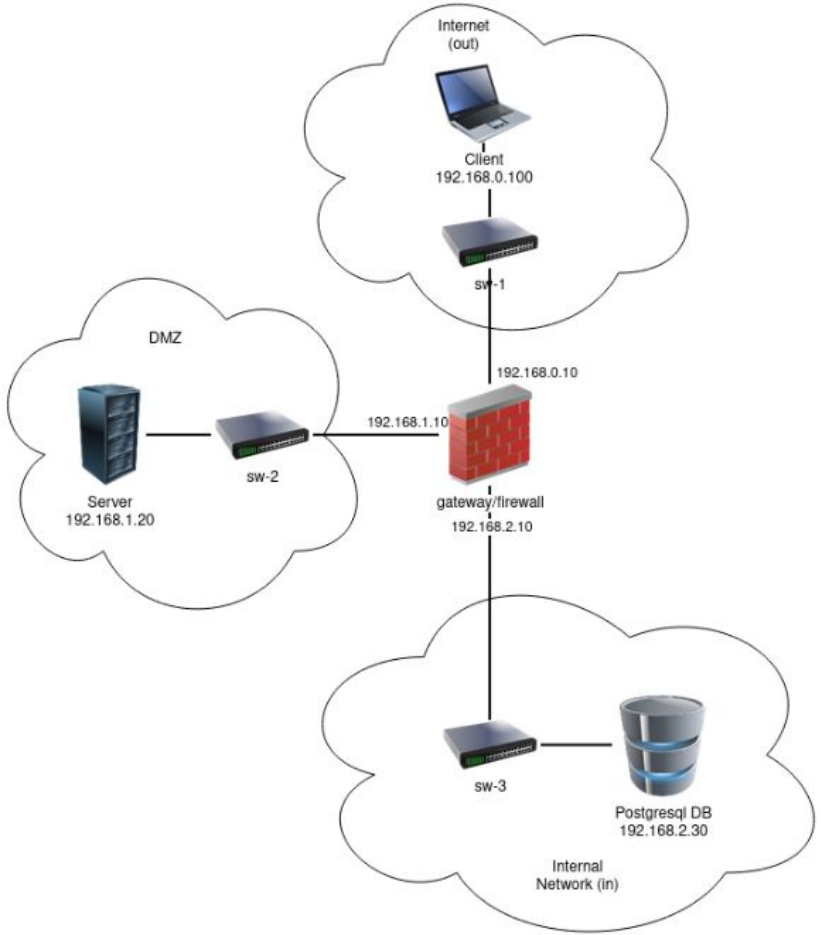
1. The symmetric key is decrypted using the user's private key.
2. Voucher information is decrypted using the decrypted symmetric key.
3. All cryptographic protection is removed, leaving the original document intact.

Secure Document Format: Guarantees Provided

1. **Confidentiality**: Voucher details are accessible only to the intended user.
2. **Authenticity**: Digital signature ensures the document's origin.
3. **Integrity**: Hashing and signature verification confirm document integrity.
4. **Non-Repudiation**: The owner can't deny association with the signed document.
5. **Freshness**: Nonce guarantees document recency, preventing replay attacks.

Built Infrastructure: key components

- VM1: External user machine connecting to the application server on port 5000.
- VM2: Gateway machine separating external and internal networks with a firewall.
- VM3: Application server receiving external user connections and connected to the database server on port 5432.
- VM4: Database server accessible only via the application server.



Built Infrastructure: firewall configuration

- **Rule 1**: Allows new external connections to the application server on port 5000.
- **Rule 2**: Allows new connections from the application server to the database server on port 5432.
- **Rule 3**: Permits all previously established connections.
- Blocks everything else.

Configured Secure Channel and Key Distribution

- **Secure Channel**: TLS over gRPC for data confidentiality and integrity.
- **Key Distribution**: Users and application server share public keys mutually.
- **Assumption**: Users and application server assume a priori knowledge of each other's public keys.
- **Certificate Authority (CA)**: OpenSSL-based CA generates certificates and private keys.
- **Certificate Signing**: CA signs all certificates for secure communication.

Configured Secure Channel and Key Distribution

- **User Connection**: Clients receive server's certificate before connection, requiring trust.
- **Assumption**: In a real-world scenario, a system administrator shares/installs server certificates on user machines.
- **Server-to-Database Connection**: Certificates for both servers signed by a root certificate, establishing mutual trust.
- **Certificate Generation**: All certificates generated centrally, sent via scp to respective servers for simplicity.

Security Challenge: Challenge Overview

1. Users can now review restaurants, requiring secure review signing for authenticity.
2. Users can give vouchers to other users.

Security Challenge

- Made a simple approach to the security challenge, because our initial Secure Document Format allowed us to do that.
- **Assumption:** Users trust the server never fails and never fools. Users are confident that the reviews sent by the server are authenticated by the respective authors. Reviews are sent to the server when created.

Security Challenge

- **JSON Modification**
  - Replaced multiple "mealVoucher" entries with a single field "mealVouchers" containing a list of code-description pairs.
  - Each pair encrypted using established methods.
- **Review Addition**
  - Introducing a new "reviews" field to the JSON for user reviews.
  - Addition facilitated seamlessly; digital signature ensures non-malicious, authenticated reviews.

Security Challenge: Benefits

- **Simplicity:** Streamlined JSON structure enhances simplicity in managing vouchers and reviews.
- **Security:** Individual encryption of the vouchers ensures data integrity and confidentiality.
- **User-Friendly:** Users can easily contribute reviews without compromising system integrity.

Main Results and Conclusions

1. **Requirement Fulfillment**: Met all outlined scenario requirements, security wise.
2. **Database Schematic**: Acknowledged partial fulfillment, with future plans for refining alignment with relational database practices.
3. **Modular Service**: Envisage introducing a more modular service for efficient client-server communication, minimizing reliance on complete JSON documents.
4. **TLS Handshake Optimization**: Exploring streamlined TLS handshake with certificate provision during the process for enhanced efficiency.
5. **Individual Restaurant Keys**: Propose implementing individual keys for each restaurant, elevating security and client trust.
6. **Enhanced Frontend**: Aspire to develop a sophisticated frontend for an improved user experience.