



**TÉCNICO**  
LISBOA

# GrooveGalaxy

Grupo A42

- José Cruz nº 99260
- Miguel Pato nº 110833
- Simão Silva nº 99329

# Points of Discussion

- Secure Document Format implementation
- Built infrastructure
- Secured channels
- Key distribution
- Security challenge implementation
- Main results and conclusions

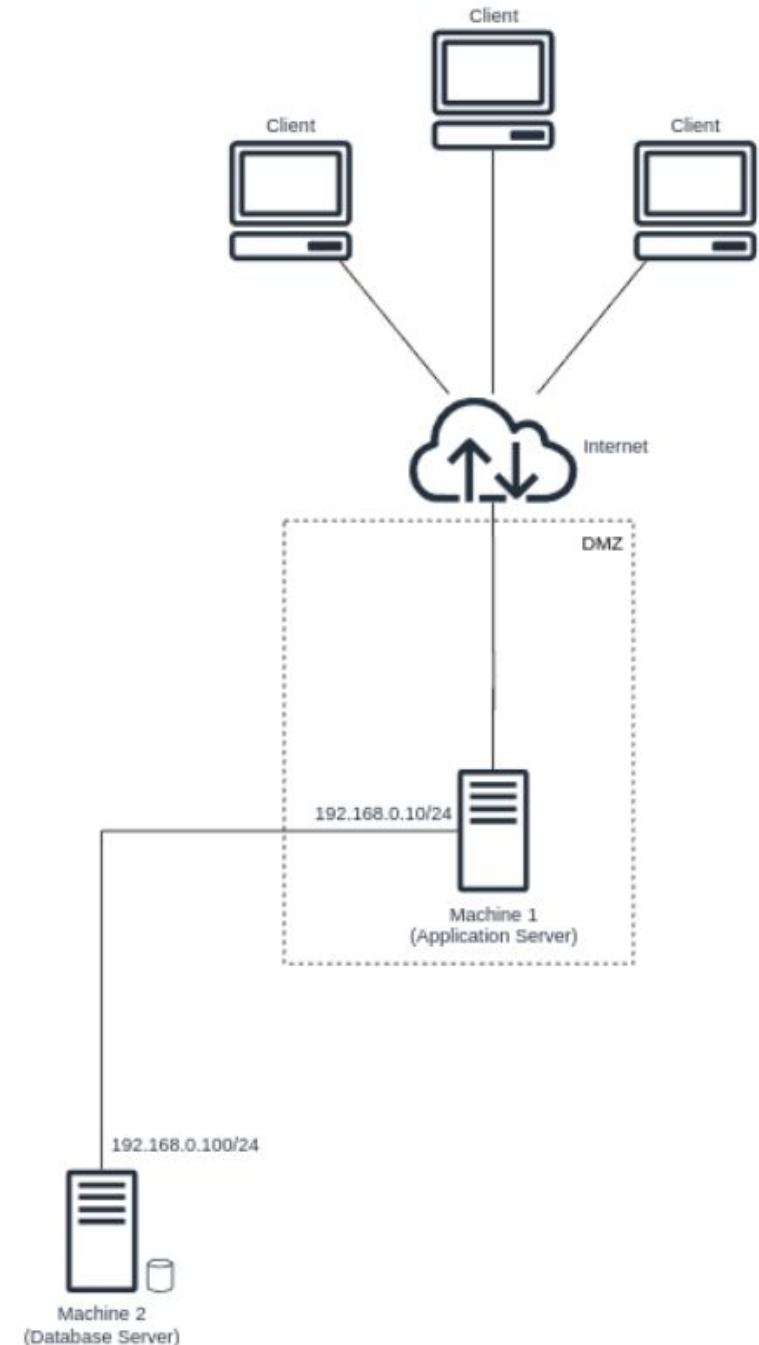
# Secure Document Format Implementation

- Only the audio (field “audioBase64”) is encrypted.
  - The encryption algorithm used was AES with CTR block encryption and zero padding.
  - The encryption is made using a temporary key sent encrypted by a permanent key to the client.
- The document is sent as whole in “bytes” format.
- The digital signature used to authenticate the server is made from the content plus the IV used to encrypt plus the encrypted temporary key, also the one used to encrypt.
  - The secure hash algorithm used was SHA-256.
- For freshness we use a nonce, composed of a random number and a timestamp.

```
{
  "media": {
    "mediaInfo": {
      "owner": "Bob",
      "format": "WAV",
      "artist": "Alison Chains",
      "title": "Man in the Bin",
      "genre": ["Grunge", "Alternative Metal"]
    },
    "mediaContent": {
      "lyrics": [
        "Trapped in a world, a box of my own",
        "Container whispers, in this space alone",
        "Echoes of silence, in the walls I confide",
        "A man in the box, with nowhere to hide",
        "Chained by thoughts, in a silent uproar",
        "Searching for keys, to unlock the door"
      ],
      "audioBase64": "UklGRiQIAAAWQVZAAWQVZFZm10IBAAAA"
    }
  }
}
```

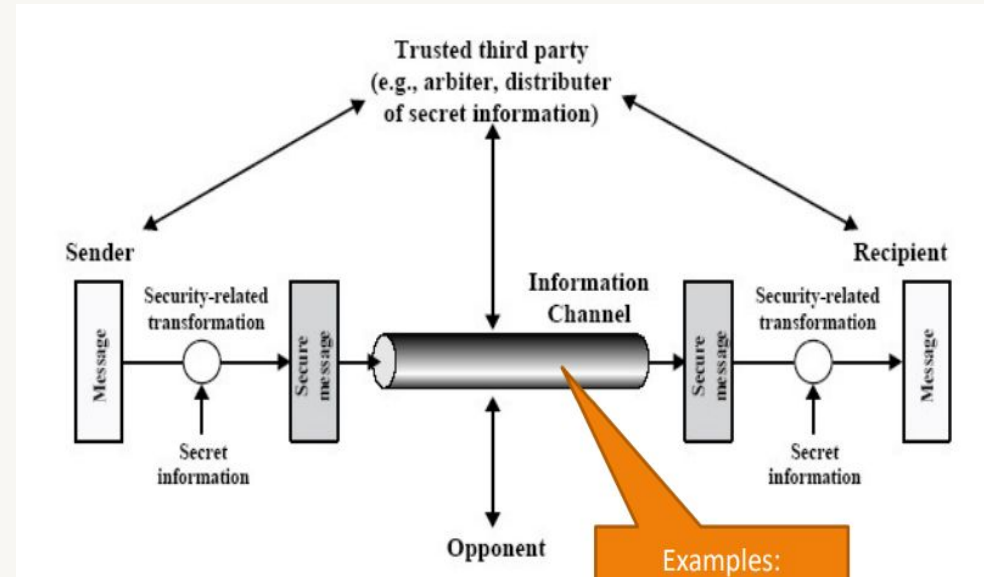
# Built Infrastructure

- Database server:
  - Authentication and TLS implemented by MySQL.
  - All the data in the database supposedly encrypted, also using MySQL already provided encryption.
- Application server:
  - Deals with business logic and communicates with the clients and database server
- Clients:
  - Connect to the application server through the internet.
  - Cannot be trusted



# Secured Channels

- Both the channels between the clients and the application server and this last and the database implement TLS/SSL.
  - This allows for encryption of the transmitted content.
  - Also provides authentication
    - The clients and the application server authenticate themselves to each other.
    - The database and the application server do this as well.



# Secured Communication Between Client and Application Server

No.	Time	Source	Destination	Protocol	Length	Info
537	272.307821388	192.168.1.50	192.168.1.10	TLSv1.2	132	Application Data
538	272.313097805	192.168.1.10	192.168.1.50	TLSv1.2	105	Application Data
539	272.314504799	192.168.1.50	192.168.1.10	TLSv1.2	105	Application Data
540	272.356241196	192.168.1.10	192.168.1.50	TCP	66	5001 → 43310 [ACK] Seq=40 Ack=106 Win=505 Len=0 TSval=2159875745 TSecr=27957372...
542	272.516731435	192.168.1.10	192.168.1.50	TLSv1.2	101	Application Data
543	272.519507393	192.168.1.10	192.168.1.50	TLSv1.2	120	Application Data
544	272.519945237	192.168.1.50	192.168.1.10	TCP	66	43310 → 5001 [ACK] Seq=106 Ack=129 Win=501 Len=0 TSval=2795737499 TSecr=2159875...
545	272.521198406	192.168.1.50	192.168.1.10	TLSv1.2	105	Application Data
546	272.523871731	192.168.1.10	192.168.1.50	TCP	66	5001 → 43310 [ACK] Seq=129 Ack=145 Win=505 Len=0 TSval=2159875911 TSecr=2795737...
547	272.523872074	192.168.1.10	192.168.1.50	TLSv1.2	98	Application Data
548	272.525769083	192.168.1.10	192.168.1.50	TLSv1.2	105	Application Data
549	272.526779898	192.168.1.50	192.168.1.10	TCP	66	43310 → 5001 [ACK] Seq=145 Ack=200 Win=501 Len=0 TSval=2795737506 TSecr=2159875...

Exemplo de um pedido  
do cliente ao servidor  
(TLS)



# Secured Communication Between Application Server and Database

58	25.584281309	192.168.0.100	192.168.0.10	TLSv1.3	99 Application Data
59	25.585871533	192.168.0.10	192.168.0.100	TLSv1.3	109 Application Data
60	25.586234904	192.168.0.100	192.168.0.10	TLSv1.3	99 Application Data
61	25.603323068	192.168.0.10	192.168.0.100	TLSv1.3	1039 Application Data
62	25.630423595	192.168.0.100	192.168.0.10	TLSv1.3	99 Application Data
63	25.632753612	192.168.0.10	192.168.0.100	TLSv1.3	109 Application Data
64	25.633046887	192.168.0.100	192.168.0.10	TLSv1.3	99 Application Data
65	25.635297752	192.168.0.10	192.168.0.100	TLSv1.3	165 Application Data
66	25.648489444	192.168.0.100	192.168.0.10	TLSv1.3	99 Application Data
67	25.650270708	192.168.0.10	192.168.0.100	TLSv1.3	109 Application Data
68	25.650584634	192.168.0.100	192.168.0.10	TLSv1.3	99 Application Data
69	25.652431476	192.168.0.10	192.168.0.100	TLSv1.3	275 Application Data
70	25.680820509	192.168.0.100	192.168.0.10	TLSv1.3	99 Application Data
71	25.683644472	192.168.0.10	192.168.0.100	TLSv1.3	109 Application Data
72	25.683962385	192.168.0.100	192.168.0.10	TLSv1.3	99 Application Data
73	25.686521513	192.168.0.10	192.168.0.100	TLSv1.3	259 Application Data
74	25.703981009	192.168.0.100	192.168.0.10	TLSv1.3	99 Application Data
75	25.705828598	192.168.0.10	192.168.0.100	TCP	74 39918 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1399958785 TS...
76	25.705855387	192.168.0.100	192.168.0.10	TCP	74 22 → 39918 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=15...
77	25.706599118	192.168.0.10	192.168.0.100	TCP	66 39918 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1399958786 TSecr=1575136290
78	25.707529000	192.168.0.10	192.168.0.100	SSHv2	87 Client: Protocol (SSH-2.0-JSCH_0.2.13)
79	25.707548603	192.168.0.100	192.168.0.10	TCP	66 22 → 39918 [ACK] Seq=1 Ack=22 Win=65152 Len=0 TSval=1575136292 TSecr=13999587...
80	25.741308093	192.168.0.100	192.168.0.10	SSHv2	106 Server: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u2)
81	25.742115377	192.168.0.10	192.168.0.100	TCP	66 39918 → 22 [ACK] Seq=22 Ack=41 Win=64256 Len=0 TSval=1399958822 TSecr=1575136...
82	25.746002175	192.168.0.100	192.168.0.10	SSHv2	1178 Server: Key Exchange Init
83	25.746966619	192.168.0.10	192.168.0.100	TCP	66 39918 → 22 [ACK] Seq=22 Ack=1153 Win=64128 Len=0 TSval=1399958827 TSecr=15751...
84	25.750586289	192.168.0.10	192.168.0.100	SSHv2	890 Client: Key Exchange Init
85	25.752313214	192.168.0.10	192.168.0.100	SSHv2	122 Client: Elliptic Curve Diffie-Hellman Key Exchange Init
86	25.752442433	192.168.0.100	192.168.0.10	TCP	66 22 → 39918 [ACK] Seq=1153 Ack=902 Win=64384 Len=0 TSval=1575136337 TSecr=1399...
87	25.756158203	192.168.0.10	192.168.0.100	TCP	66 44294 → 3306 [ACK] Seq=3625 Ack=4280 Win=64128 Len=0 TSval=1399958836 TSecr=1...
88	25.760488676	192.168.0.100	192.168.0.10	SSHv2	614 Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
89	25.764137748	192.168.0.10	192.168.0.100	SSHv2	82 Client: New Keys
90	25.765238221	192.168.0.10	192.168.0.100	SSHv2	150 Client:
91	25.765511487	192.168.0.100	192.168.0.10	TCP	66 22 → 39918 [ACK] Seq=1701 Ack=1002 Win=64384 Len=0 TSval=1575136350 TSecr=139...

Comunicação com  
MySql (TLS)

Upload do áudio via  
SFTP

# Key Distribution

- The client and the application server share a secret (symmetric) key from the beginning.
  - This symmetric key is permanent (stored securely in java keystores).
  - It is used to encrypt a **temporary key** (stored in the database).
- The document is encrypted using a symmetric temporary key
  - The temporary keys are renewed all at the same time by the database server.
- The client also has the application server public key (not in a keystore) from the get go to verify the digital signature of received documents.
- The server has its own private key (not in a keystore) to sign digital signatures.



# Certificate Distribution

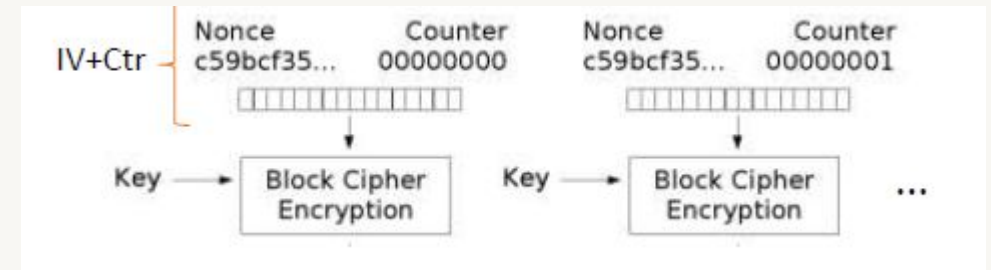
- The clients and application server possess two java keystores.
  - One containing the certificate of application server, allowing it to authenticate itself towards the clients.
  - Another containing client certificates (each client only has its own) so that the clients can authenticate themselves towards the application server.

This combination allows for the clients and application server to ensure that they are talking to each other and not a malicious attacker.

- The database already has a certificate, made available by MySQL.

# Security Challenge Implementation

- In order to allow for playback of a music from the middle of it we used **Counter** mode for our symmetric block encryption keys.
- To implement the concept of family sharing, we made it so that the temporary keys (used to encrypt the documents) would be a family key, also temporary, shared by all the users of a family.
  - This change is invisible to the clients.



# Main Results and Conclusions

- We were able to secure the documents and the communication channels.
- Unfortunately, our clients do not store the song documents in a confidential way.
- We also did not setup the firewalls in the application server and the database, which would have improved security