

Projeto de Segurança Informática e Nas Organizações

Universidade de Aveiro

Isac Cruz (90513)

Luís Oliveira (98543)

Rui Rosmaninho (113553)



universidade
de aveiro

Conteúdo

1	Cifra	4
1.1	Introdução	4
1.2	Alterações em época de recurso	4
1.3	Cifra e decifra de pedidos	5
1.4	Cifra e decifra de respostas	6
1.5	Marcas de Verificação	6
1.6	Cifra Genérica	6
1.7	Cifra e envio no formato Json	7
1.7.1	Codificação	8
1.8	Envio de ficheiro	8
2	Repository	9
2.1	Estrutura de dados	9
2.2	Auxiliar Functions	9
2.2.1	Load Rep Key	10
2.2.2	Save State	10
2.2.3	Load State	10
2.2.4	Save Key	10
2.2.5	Load Org Private Keys	10
2.2.6	Get Valid Session Information	11
2.2.7	Validate Subject Permission	11
2.2.8	Validate Document Permission	11
2.2.9	Compute Hash	11
2.2.10	Add Org Private Keys	11
2.2.11	Load Org Private Keys	12
2.3	Anonymous API Commands	12
2.3.1	Get Rep Key	12
2.3.2	Rep Create Org	12
2.3.3	Rep List Orgs	13
2.3.4	Rep Create Session	13
2.3.5	Rep Get File	14
2.4	Authenticated API Commands	14
2.4.1	Rep Assume Role	14
2.4.2	Rep Drop Role	14
2.4.3	Rep List Roles	15
2.4.4	Rep List Subjects	15
2.4.5	Rep List Role Subjects	15
2.4.6	Rep List Subject Roles	16
2.4.7	Rep List Role Permissions	16
2.4.8	Rep List Permission Roles	16
2.4.9	Rep List Docs	17
2.5	Authorized API Commands	17
2.5.1	Rep Add Subject	17
2.5.2	Rep Suspend Subject	18
2.5.3	Rep Activate Subject	18
2.5.4	Rep Add Role	19
2.5.5	Rep Suspend Role	19
2.5.6	Rep Reactivate Role	19
2.5.7	Rep Add Permission	20

2.5.8	Rep Remove Permission	20
2.5.9	Rep Add Doc	21
2.5.10	Rep Get Doc Metadata	22
2.5.11	Rep Get Doc File	22
2.5.12	Rep Delete Doc	23
2.5.13	Rep Acl Doc	24
3	Aplication Security Verification Standats (ASVS)	25

Lista de Figuras

1.1	Fluxo de Cifra	4
1.2	Cifra Genérica	7
1.3	Ficheiro Criado	7
1.4	Cifra Genérica	8

Capítulo 1

Cifra

1.1 Introdução

Para facilitar e isolar todo o processo de cifragem foi criado um módulo paralelo à comunicação.

As mensagens são criadas e enviadas pelo cliente/servidor de forma transparente, e o módulo de cifra garante um aumento de segurança na comunicação. O módulo de cifra adiciona marcas de verificação (como marcas de verificação temporais), cifra a mensagem e envia-a para o receptor. Na recepção, a mensagem é decifrada e retornada num formato legível.

Este módulo fornece ainda um conjunto de ferramentas adicionais para operações intermédias (como cifra simétrica e assimétrica), criação e verificação de assinaturas digitais, criação e verificação de MAC's, entre outras.

As funções principais deste módulo são 'encrypt_request' e 'decrypt_request' (Seção 1.3) que cifram toda a comunicação pré-envio e pós-recepção. Será precisamente nestas funções que as marcas de verificação serão adicionadas (no envio). Cada campo do pedido é cifrado com as funções 'encrypt_easy' (Seção 1.6) e, caso não seja um ficheiro de uma organização, será codificado para ser enviado no formato json (Seção 1.7).

No caso de envio de um ficheiro de uma organização, a mensagem será assinada com um MAC para garantir a integridade. O diagrama de fluxo é apresentado na Figura 1.1

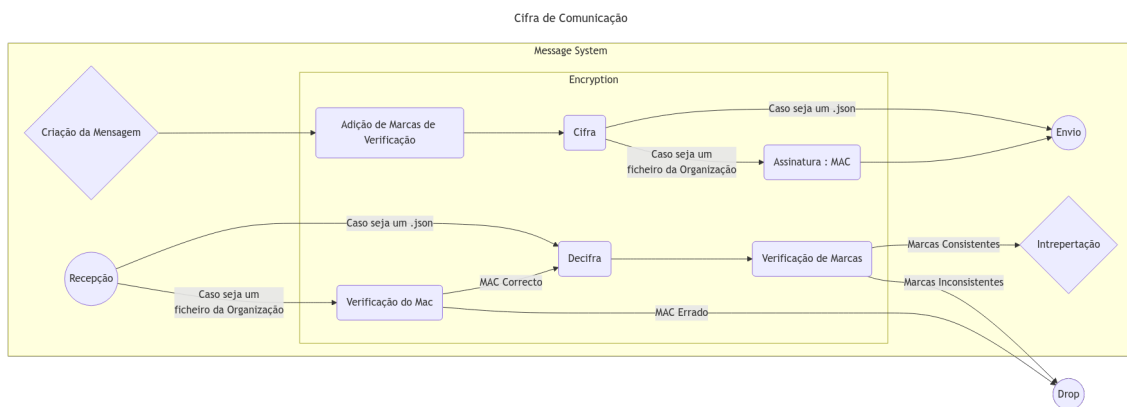


Figura 1.1: Fluxo de Cifra

1.2 Alterações em época de recurso

Durante a época de recurso, o foco foi melhorar o projeto entregue previamente. Nesse sentido, e com o objetivo de ir ao encontro do proposta dos professores, adicionámos/alterámos as funções:

- create_session
- método de como chamamos cada função
- rep_subject_credentials
- rep_suspend_role

1.3 Cifra e decifra de pedidos

Os campos do pedido são todos cifrados com uma cifra híbrida utilizando `encrypt_easy` (Seção 1.6) com o algoritmo 'AES', sendo codificados os dados cujo envio precisa ser efetuado no formato json (Seção 1.7). Na decifra, é criado um objeto 'Req' cujos atributos contêm as informações relevantes do pedido: `form`, `files`, `json`. Sempre que possível, os pedidos são cifrados utilizando a chave pública do servidor.

```
1 class Req:
2     def __init__(self, data, files, json_file):
3         self.form = data
4         self.files = files
5         self.json = json_file
6
7     def __str__(self): return f>Data: {self.form}\nFiles: {self.files}\nJson: {self.json}"
8
9     @property
10    def form(self): return self._data
11
12    @form.setter
13    def form(self, value): self._data = value
14
15    @property
16    def files(self): return self._files
17
18    @files.setter
19    def files(self, value): self._files = value
20
21    @property
22    def json(self): return self._json_file
23
24    @json.setter
25    def json(self, value): self._json_file = value
26
27    def encrypt_request(rep_public_key, action, endpoint, data={}, files={}, json_file={}):
28        response = None
29        file_lisr = []
30
31        if rep_public_key != None:
32            data["previous_interaction"] = datetime.now().isoformat()
33            data = {"data" : decode_bytes2string(encrypt_easy(rep_public_key,
34                ↪ json.dumps(data).encode(), "AES"))} if not data == {} else {}
35            for key in files:
36                secret = encrypt_easy(rep_public_key, files[key].read(), "AES")
37                inner_file = io.BufferedReader(io.BytesIO(secret))
38                file_lisr.append(inner_file)
39                files[key] = inner_file
40            json_file = {"json" : decode_bytes2string(encrypt_easy(rep_public_key,
41                ↪ json.dumps(json_file).encode(), "AES"))} if not json_file == {} else {}
42        else:
43            data = {"data" : json.dumps(data)} if not data == {} else {}
44            json_file = {"json" : json.dumps(json_file)} if not json_file == {} else {}
45
46        if action == 'POST': response = requests.post(endpoint, data=data, files=files)
47        elif action == 'GET': response = requests.get(endpoint, data=data, files=files)
48
49        for file in file_lisr: file.close()
50        return response
51
52    def decrypt_request(rep_private_key, request) -> Req:
53        data = request.form["data"]
54        files = {}
55        json_file = {}
```

```

56     for file_name in request.files:
57         secret = decrypt_easy(rep_private_key, request.files[file_name].read(), "AES")
58         request.files[file_name].close()
59         files[file_name] = secret
60
61     data = json.loads(decrypt_easy(rep_private_key, encode_string2bytes(data), "AES")) if
    ↪ data != {} else {}
62     json_file = json.loads(decrypt_easy(rep_private_key, encode_string2bytes(json_file),
    ↪ "AES")) if json_file != {} else {}
63     return Req(data, files, json_file)

```

1.4 Cifra e decifra de respostas

As respostas são cifradas utilizando a chave pública do utilizador nos pedidos não anónimos. As respostas são cifradas pela função 'easy' (Seção 1.6) e codificadas pelas funções 'encode' (Seção 1.7) para envio no formato 'json'. Na chegada ao cliente, são aplicados os processos inversos para a reconstrução da resposta original. Esta parte de código não chegou a ser incluída na versão final por equívoco, mas toda a estrutura está desenhada para funcionar com esta 'feature'.

```

1 def encrypt_response(key, data: {}):
2     if key == None: return jsonify(data)
3     jsonify({
4         "data" : decode_bytes2string(encrypt_easy(key, json.dumps(data).encode(), "AES"))
5     })
6
7 def decrypt_response(key, data: {}):
8     if key == None: return data
9     if "data" in data:
10         return json.loads(decrypt_easy(key, encode_string2bytes(data["data"]), "AES"))
11

```

1.5 Marcas de Verificação

As marcas de verificação são campos adicionados ao pedido que contêm informações relevantes para a validação e verificação do pedido. Neste projeto, apenas foram utilizados marcadores temporais. No entanto, outros tipos de marcações podem ser adicionadas. Todas as marcas são adicionadas antes da cifra dos dados.

Estas têm como intuito impedir pedidos duplicados, registando e comparando no sistema a última data de um pedido.

1.6 Cifra Genérica

A cifra genérica trata-se de uma cifra híbrida que, utilizando um algoritmo definido no argumento de entrada, cria um ficheiro cifrado com uma chave aleatória, a qual é cifrada com a chave publica do servidor (Figura 1.4).

Na saída da função, é criado um ficheiro onde os primeiros 20 bits contêm o tamanho N da chave simétrica cifrada. Os N bits seguintes contêm a chave cifrada e os restantes bits pertencentes ao ficheiro cifrado (Figura 1.3). Os ficheiros criados podem ser armazenados, enviados ou incluídos num ficheiro json (Seção 1.7).

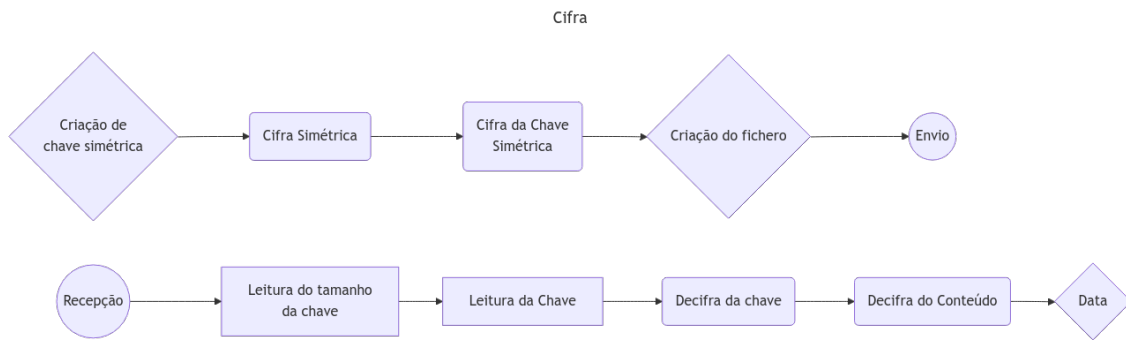


Figura 1.2: Cifra Genérica

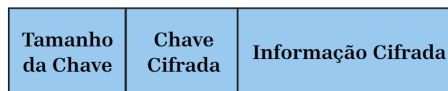


Figura 1.3: Ficheiro Criado

```

1 def encrypt_easy(public_key, data, alg_name):
2     encrypted_message, simetric_key = simetric_encryption(data, alg_name)
3     encrypted_key, size = encrypt_key(public_key, simetric_key)
4     return size + encrypted_key + encrypted_message
5
6 def decrypt_easy(private_key, data, alg_name):
7     key_lenght = int.from_bytes(data[0:KEY_SIZE], byteorder="big")
8     data = data[KEY_SIZE:]
9     encrypted_key = data[0:key_lenght]
10    data = data[key_lenght:]
11    simetric_key = decrypt_key(private_key, encrypted_key)
12    message = simetric_decryption(simetric_key, data, alg_name)
13    return message
  
```

1.7 Cifra e envio no formato Json

Para o envio de informação no formato json, é necessário a criação de informação descodificável (possível de ser convertida para caracteres). Os bits criados pelas funções de cifra nem sempre são descodificáveis. Por esta razão, foram criadas funções para o efeito. No sistema criado é possível cifrar informação sem perder a possibilidade do envio no formato mais comum.

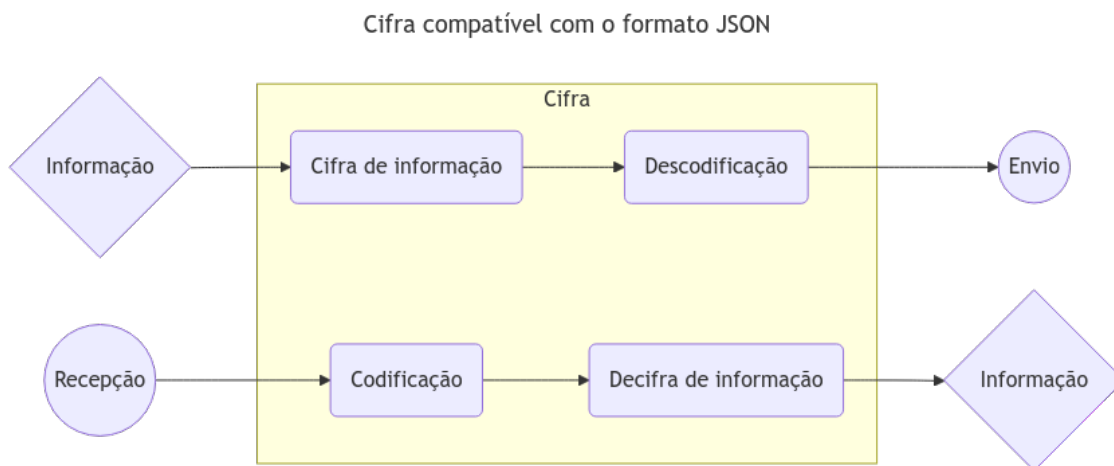


Figura 1.4: Cifra Genérica

1.7.1 Codificação

Codifica o ficheiro em base64 e, de seguida, em utf-8 para possibilitar o envio no formato de string.

```

1  def encode_string2bytes(data):
2  return base64.b64decode(codecs.encode(data, 'utf-8'))

```

Descodifica o ficheiro de utf-8 para base64 e de base64 para binario para possibilitar a leitura no formato string do ficheiro.

```

1  def decode_bytes2string(data:bytes):
2  return codecs.decode(base64.b64encode(data), 'utf-8')

```

1.8 Envio de ficheiro

No envio de ficheiros do utilizador, é criado um MAC que é enviado após o ficheiro que confirma a autenticidade do mesmo. Este MAC é criado após a cifra do documento. Apesar de ser utilizado apenas para o envio do ficheiro, o código pode ser adaptado para incluir verificação de MAC em todas as mensagens, com relativa facilidade.

Para garantir a autenticidade do ficheiro foram utilizadas assinaturas e MAC's de forma a fazer uso seja de chaves públicas, seja de chaves privadas, de acordo com a conveniência.

Capítulo 2

Repository

Este capítulo apresenta as funcionalidades relacionadas ao repositório, detalhando as implementações que garantem a integridade e a segurança do sistema. Todas as funções descritas foram desenvolvidas com foco na validação de permissões, integridade dos dados e autenticação de sessões.

As verificações incluem:

- **Autenticação de Sessões:** Cada operação exige a validação da sessão, incluindo o uso de chaves criptográficas e a validação de interações anteriores.
- **Autorização Baseada em Permissões:** São verificadas as permissões específicas de cada 'role' associada aos utilizadores, garantindo que ações sejam realizadas apenas por sujeitos autorizados.
- **Consistência de Dados:** Os dados armazenados são verificados para evitar duplicações ou inconsistências, como adicionar utilizadores ou documentos já existentes.
- **Controlo de Acesso:** A configuração detalhada da *Access Control List* (ACL) assegura que apenas 'roles' adequadas possam aceder ou modificar documentos e permissões.
- **Roles:** Os sujeitos podem assumir uma role *manager* durante uma sessão, com verificações de permissões associadas.
- **Controlo de Roles e Permissões:**
 - **Criar e gerir Roles:** Definir roles com permissões associadas para controlar o acesso a certas funcionalidades na organização;
 - **Adicionar e remover Permissões:** Alterar permissões associadas a roles;
 - **Associar sujeitos a Roles:** Manipular associações entre sujeitos e roles dentro de uma organização.
 - **Permissões:**
 - * *ROLE_NEW*,
 - * *ROLE_DOWN*,
 - * *ROLE_UP*,
 - * *ROLE_MOD*.

Com esse modelo de validação, o sistema promove um ambiente seguro e confiável para a manutenção de utilizadores, 'roles', documentos e permissões.

2.1 Estrutura de dados

O repositório e cliente encontram-se separados, à imagem de um cenário na vida real. O cliente tem as suas pastas privadas e armazenamento de ficheiros e o repositório tem reservado a sua metadata. É utilizado ficheiros *json* em vez de uma base de dados em *SQL*.

2.2 Auxiliar Functions

As funções auxiliares foram desenvolvidas para simplificar e minimizar a repetição de código, aperfeiçoando os sistemas de segurança essenciais do sistema.

2.2.1 Load Rep Key

Carrega ou cria as chaves pública e privada do repositório, guardando-as de forma segura caso ainda não existam.

```
1 def key_loader():
2     global rep_public_key, rep_private_key
3     if not os.path.isfile("private/rep_public_key.pem") and not
4         ↪ os.path.isfile("private/rep_private_key.pem"):
5         rep_private_key, rep_public_key = repUtils.generate_credentials()
6         with open("private/rep_private_key.pem", "wb") as file:
7             ↪ file.write(repUtils.serialize_private_key(rep_private_key))
8         with open("private/rep_public_key.pem", "wb") as file:
9             ↪ file.write(repUtils.serialize_public_key(rep_public_key))
10        return
11
12    with open("private/rep_private_key.pem", "rb") as file: rep_private_key =
13        ↪ load_pem_private_key(file.read(), password=None, backend=default_backend())
14    with open("private/rep_public_key.pem", "rb") as file: rep_public_key =
15        ↪ load_pem_public_key(file.read(), backend=default_backend())
```

2.2.2 Save State

Cifra e guarda o estado atual do repositório num ficheiro seguro.

```
1 def save_state():
2     global repository
3     os.makedirs("private", exist_ok=True)
4     with open("private/repository.cifr", 'wb') as file:
5         ↪ file.write(repUtils.encrypt_easy(rep_public_key, json.dumps(repository).encode(),
6         ↪ "AES"))
```

2.2.3 Load State

Decifra e carrega o estado salvo do repositório a partir de um ficheiro seguro, caso exista.

```
1 def load_state():
2     global repository
3     with open("private/repository.cifr", 'rb') as file: repository =
4         ↪ json.loads(repUtils.decrypt_easy(rep_private_key, file.read(), "AES"))
```

2.2.4 Save Key

Cifra e guarda a chave privada de uma organização de forma segura.

```
1 def save_key(organization_name, private_key):
2     global org_private_keys
3     org_private_keys[organization_name] = private_key
4     os.makedirs("private", exist_ok=True)
5     with open("private/private.cifr", 'wb') as file:
6         ↪ file.write(repUtils.encrypt_easy(rep_public_key,
7         ↪ json.dumps(org_private_keys).encode(), "AES"))
```

2.2.5 Load Org Private Keys

Decifra e carrega as chaves privadas de todas as organizações a partir de um ficheiro seguro.

```
1 def load_org_private_keys():
2     global org_private_keys
3     with open("private/private.cifr", 'rb') as file: org_private_keys =
4         ↪ json.loads(repUtils.decrypt_easy(rep_private_key, file.read(), "AES"))
```

2.2.6 Get Valid Session Information

Valida as informações de uma sessão, garantindo que esteja ativa e associada a uma organização e utilizador válidos. Pode ainda remover a sessão, caso esta se encontre inativa por mais de trinta minutos.

```
1 def get_valid_session_information(session_file, previous_interaction):
2     organization, session_id = next(iter(json.loads(session_file).items()))
3     if organization not in repository: return {"error": "Invalid organization"}, None,
4     ↪ None
5     if datetime.fromisoformat(previous_interaction) ==
6     ↪ datetime.fromisoformat(repository[organization]['sessions'][session_id]['last_active']):
7     ↪ return {"abort": "Duplicate request detected"}, None, None
8     else: repository[organization]['sessions'][session_id]['last_active'] =
9     ↪ previous_interaction
10    if session_id not in repository[organization]['sessions']: return {"error": "Invalid
11    ↪ session ID"}, None, None
12    username = repository[organization]['sessions'][session_id]['username']
13    if not repository[organization]['subjects'][username]['status']: return {"error":
14    ↪ "User suspended"}, None, None
15    if datetime.now() >
16    ↪ datetime.fromisoformat(repository[organization]['sessions'][session_id]['last_active'])
17    ↪ + SESSION_TIMEOUT:
18        del repository[organization]['sessions'][session_id]
19        return {"error": "Session timeout"}, None, None
20    return organization, session_id, username
```

2.2.7 Validate Subject Permission

Verifica se um utilizador possui as permissões necessárias para realizar uma operação específica.

```
1 def validate_subject_permission(organization, session_id, username, permission):
2     for role in repository[organization]['sessions'][session_id]['roles']:
3         role = repository[organization]['roles'][role]
4         if role['status'] and username in role['subjects'] and permission in
5         ↪ (role['permissions'] + role['acl']): return True
6     return False
```

2.2.8 Validate Document Permission

Confirma se um sujeito possui as permissões necessárias para aceder a um documento específico.

```
1 def validate_document_permission(organization, session_id, document_name, permission):
2     with open(f"metadatas/{document_name}", 'r') as metadata_file:
3         metadata = json.load(metadata_file)
4         for role in metadata['acl']:
5             if role in repository[organization]['sessions'][session_id]['roles'] and
6             ↪ permission in metadata['acl'][role]: return True
7     return False
```

2.2.9 Compute Hash

Calcula o hash SHA-256 do conteúdo fornecido.

```
1 def compute_hash(content):
2     return hashlib.sha256().update(content).hexdigest()
```

2.2.10 Add Org Private Keys

Adiciona e guarda de forma segura a chave privada de uma organização.

```

1 def add_org_private_key(org_name, private_key):
2     global org_private_keys
3     org_private_keys[org_name] = repUtils.serialize_private_key(private_key).decode()
4     os.makedirs("private", exist_ok=True)
5     with open("private/private.cifr", 'wb') as file:
6         data = repUtils.encrypt_easy(rep_public_key,
7             ↪ json.dumps(org_private_keys).encode(), "AES")
8         file.write(data)

```

2.2.11 Load Org Private Keys

Carrega todas as chaves privadas das organizações de forma segura, garantindo a uma decifragem correta e concisa.

```

1 def load_org_private_keys():
2     global org_private_keys
3     if os.path.isfile("private/private.cifr"):
4         with open("private/private.cifr", 'rb') as file:
5             data = file.read()
6             data = repUtils.decrypt_easy(rep_private_key, data, "AES")
7             org_private_keys = json.loads(data)

```

2.3 Anonymous API Commands

As funções anônimas podem ser executadas sem a criação prévia de uma sessão e não estão sujeitas a verificações. Apesar de serem anônimas, foram desenvolvidas garantindo a confidencialidade das informações trocadas do cliente para o repositório.

2.3.1 Get Rep Key

Obtém e devolve a chave pública do repositório, a fim de serem estabelecidas comunicações seguras.

```

1 @app.route('/repository/get_rep_key', methods=['GET'])
2 def get_rep_key():
3     return jsonify({"public_key": repUtils.serialize_public_key(rep_public_key).decode()})

```

2.3.2 Rep Create Org

Cria uma nova organização no repositório, atribuindo chaves criptográficas à organização e assegurando as permissões essenciais da 'role' "Managers".

```

1 @app.route('/repository/create_org', methods=['POST'])
2 def rep_create_org():
3     req = repUtils.decrypt_request(rep_private_key, request)
4     public_key = json.loads(req.files["public_key_file"])['pubkey']
5     organization = req.form['organization']
6     identifier = req.form["identifier"]
7     name = req.form["name"]
8     email = req.form["email"]
9
10    if organization in repository: return jsonify({"error": "Organization already
11    ↪ exists"}), 400
12    org_private_key, org_public_key = repUtils.generate_credentials()
13    repository[organization] = {
14        "documents": [],
15        "subjects": {
16            identifier: {
17                "name": name,
18                "email": email,
19                "pubkey": public_key,

```

```

19         "status": True
20     }
21 },
22 "roles": {
23     "Managers": {
24         "subjects": [identifier],
25         "acl": ["DOC_NEW", "SUBJECT_NEW", "SUBJECT_DOWN", "SUBJECT_UP",
26             ↪ "ROLE_ACL"],
27         "permissions": ["ROLE_NEW", "ROLE_DOWN", "ROLE_UP", "ROLE_MOD"],
28         "status": True
29     }
30 },
31 "sessions": {},
32 "public_key": repUtils.serialize_public_key(org_public_key).decode()
33 }
34
35 save_state()
36 save_key(organization, repUtils.serialize_private_key(org_private_key).decode())
37 return jsonify({"organization_public_key":
38     ↪ repUtils.serialize_public_key(org_public_key).decode()})

```

2.3.3 Rep List Orgs

Devolve a lista de organizações registadas no repositório.

```

1 @app.route('/repository/list_orgs', methods=['GET'])
2 def rep_list_orgs():
3     return jsonify({"organizations": list(repository.keys())})

```

2.3.4 Rep Create Session

Cria uma nova sessão para um utilizador numa organização, criando um identificador único para a mesma. Utiliza os parâmetros 'start_time' e 'last_active' para acompanhar a criação e a última interação da sessão com o repositório.

Estes parâmetros permitem a validação de 'timeouts', encerrando a sessão em caso de inatividade e garantindo a segurança do sistema.

```

1 @app.route('/repository/organization/sessions/create_session', methods=['POST'])
2 def rep_create_session():
3     request_data = repUtils.decrypt_request(rep_private_key, request)
4     public_key = json.loads(request_data.files["credentials_file"])[ 'pubkey' ]
5     organization = request_data.form['organization']
6     identifier = request_data.form['identifier']
7     password = request_data.form['password']
8     session_id = hashlib.sha256(public_key.replace("\n",
9     ↪ "").strip().encode('utf-8')).hexdigest()
10
11     if organization not in repository: return jsonify({"error": "Invalid organization"}),
12     ↪ 400
13     if identifier not in repository[organization][ 'subjects' ]: return jsonify({"error":
14     ↪ "Invalid username"}), 400
15     if not repository[organization][ 'subjects' ][identifier][ 'status' ]: return
16     ↪ jsonify({"error": "User suspended"}), 400
17     if session_id in repository[organization][ 'sessions' ]: return jsonify({"error":
18     ↪ "Session already active"}), 400
19
20     repository[organization][ "sessions" ][session_id] = {
21         "username": identifier,
22         "keys": [public_key],
23         "roles": [],
24         "start_time": datetime.now().isoformat(),
25         "last_active": datetime.now().isoformat()
26     }

```

```
22     save_state()
23     return jsonify({organization: session_id})
```

2.3.5 Rep Get File

Permite o download de um ficheiro do repositório a partir da sua 'file_handle', garantindo a integridade com uma assinatura digital.

```
1 @app.route('/repository/files/get_file', methods=['GET'])
2 def rep_get_file():
3     file_handle = request.form['file_handle']
4     if not os.path.exists(f"files/{file_handle}"): return jsonify({"error": "Invalid
    ↳ document"}), 400
5     with open(f"files/{file_handle}", 'rb') as file:
6         content = file.read()
7         mac = repUtils.mac_sign_easy(rep_private_key, content)
8         buff = io.BufferedReader(io.BytesIO(len(mac).to_bytes(20, byteorder='big') + mac +
    ↳ content))
9     return send_file(buff, as_attachment=True, download_name="save")
```

2.4 Authenticated API Commands

As funções autenticadas utilizam sessões criadas previamente com IDs únicos e são validadas rigorosamente. As validações incluem verificação de 'timestamps' para negar acesso a utilizadores inativos e prevenção de pedidos duplicados, garantindo também a integridade das operações.

Em caso de falha em alguma fase das validações, é retornado um erro apropriado.

2.4.1 Rep Assume Role

Permite um utilizador assumir uma 'role' específica dentro de uma organização, durante a sua sessão.

O sistema previne a assunção de 'roles' já assumidas, ou que o utilizador não tem acesso, ou que não existem.

```
1 @app.route('/repository/organization/roles/assume_role', methods=['POST'])
2 def rep_assume_role():
3     request_data = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
    ↳ get_valid_session_information(request_data.files["session_file"],
    ↳ request_data.form["previous_interaction"])
5     if not username: return jsonify(organization), 400 # In this case, 'organization' is
    ↳ an error message
6     role = request_data.form["role"]
7
8     if role not in repository[organization]["roles"]: return jsonify({"error": "Invalid
    ↳ role"}), 400
9     if role in repository[organization]['sessions'][session_id]['roles']: return
    ↳ jsonify({"error": "Role already assigned"}), 400
10    if username not in repository[organization]['roles'][role]['subjects']: return
    ↳ jsonify({"error": "Invalid permission"}), 400
11
12    repository[organization]['sessions'][session_id]['roles'].append(role)
13    save_state()
14    return jsonify({"success": True})
```

2.4.2 Rep Drop Role

Remove uma 'role' específica da sessão de um utilizador.

O sistema previne a remoção de 'roles' não assumidas durante a sessão, ou que não existam.

```

1 @app.route('/repository/organization/roles/drop_role', methods=['POST'])
2 def rep_drop_role():
3     request_data = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
6     ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8     ↪ an error message
9     role = request_data.form["role"]
10
11     if role not in repository[organization]['sessions'][session_id]['roles']: return
12     ↪ jsonify({"error": "Invalid role"}), 400
13
14     repository[organization]['sessions'][session_id]['roles'].remove(role)
15     save_state()
16     return jsonify({"success": True})

```

2.4.3 Rep List Roles

Lista todas as 'roles' associadas à sessão de um utilizador.

```

1 @app.route('/repository/organization/list_roles', methods=['GET'])
2 def rep_list_roles():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
6     ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8     ↪ an error message
9
10     return jsonify({"roles": repository[organization]['sessions'][session_id]['roles']})

```

2.4.4 Rep List Subjects

Lista os sujeitos (utilizadores) de uma organização e o seu estado ativo/inativo, podendo ser feita uma filtragem a partir de um 'username' específico.

```

1 @app.route('/repository/organization/list_subjects', methods=['GET'])
2 def rep_list_subjects():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
6     ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8     ↪ an error message
9     identifier = request_data.form.get("identifier")
10
11     if identifier and identifier in repository[organization]["subjects"]: return
12     ↪ jsonify({identifier: repository[organization]["subjects"][identifier]["status"]})
13     return jsonify({subject: repository[organization]["subjects"][subject]["status"] for
14     ↪ subject in repository[organization]["subjects"]})

```

2.4.5 Rep List Role Subjects

Lista os sujeitos associados a uma 'Role' específica numa organização.

```

1 @app.route('/repository/organization/roles/list_role_subjects', methods=['GET'])
2 def rep_list_role_subjects():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
6     ↪ request_data.form["previous_interaction"])

```

```

5     if not username: return jsonify(organization), 400 # In this case, 'organization' is
      ↪ an error message
6     role = request_data.form["role"]
7
8     if role not in repository[organization]["roles"]: return jsonify({"error": "Invalid
      ↪ role"}), 400
9
10    return jsonify({"roles": repository[organization]['roles'][role]['subjects']})

```

2.4.6 Rep List Subject Roles

Lista todas as 'roles' associadas a um sujeito específico numa organização.

```

1  @app.route('/repository/organization/subjects/list_subject_roles', methods=['GET'])
2  def rep_list_subject_roles():
3      request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4      organization, session_id, username =
      ↪ get_valid_session_information(request_data.files["session_file"],
      ↪ request_data.form["previous_interaction"])
5      if not username: return jsonify(organization), 400 # In this case, 'organization' is
      ↪ an error message
6      identifier = request_data.form["identifier"]
7
8      if identifier not in repository[organization]["subjects"]: return jsonify({"error":
      ↪ "Invalid subject"}), 400
9
10     return jsonify({"roles": [role for role in repository[organization]['roles'] if
      ↪ identifier in repository[organization]['roles'][role]['subjects']]})

```

2.4.7 Rep List Role Permissions

Lista todas as permissões associadas a uma 'role' específica numa organização.

```

1  @app.route('/repository/organization/roles/list_role_permissions', methods=['GET'])
2  def rep_list_role_permissions():
3      request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4      organization, session_id, username =
      ↪ get_valid_session_information(request_data.files["session_file"],
      ↪ request_data.form["previous_interaction"])
5      if not username: return jsonify(organization), 400 # In this case, 'organization' is
      ↪ an error message
6      role = request_data.form["role"]
7
8      if role not in repository[organization]['roles']: return jsonify({"error": "Invalid
      ↪ role"}), 400
9
10     return jsonify({"permissions": repository[organization]['roles'][role]['permissions']
      ↪ + repository[organization]['roles'][role]['acl']})
11

```

2.4.8 Rep List Permission Roles

Lista todas as 'roles' que contêm uma permissão específica numa organização.

```

1  @app.route('/repository/organization/roles/list_permission_roles', methods=['GET'])
2  def rep_list_permission_roles():
3      request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4      organization, session_id, username =
      ↪ get_valid_session_information(request_data.files["session_file"],
      ↪ request_data.form["previous_interaction"])
5      if not username: return jsonify(organization), 400 # In this case, 'organization' is
      ↪ an error message

```

```

6     permission = request_data.form["permission"]
7
8     if permission not in ["DOC_NEW", "SUBJECT_NEW", "SUBJECT_DOWN", "SUBJECT_UP",
9         ↪ "ROLE_ACL", "ROLE_NEW", "ROLE_DOWN", "ROLE_UP", "ROLE_MOD"]: return
10    ↪ jsonify({"error": "Invalid permission"}), 400
11
12    return jsonify({"roles": [role for role in repository[organization]['roles'] if
13        ↪ permission in (repository[organization]['roles'][role]['permissions'] +
14        ↪ repository[organization]['roles'][role]['acl'])]})

```

2.4.9 Rep List Docs

Lista os documentos pertencentes a uma organização, com filtros opcionais por criador e data.

```

1  @app.route('/repository/organization/documents/list_docs', methods=['GET'])
2  def rep_list_docs():
3      request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4      organization, session_id, username =
5      ↪ get_valid_session_information(request_data.files["session_file"],
6      ↪ request_data.form["previous_interaction"])
7      if not username: return jsonify(organization), 400 # In this case, 'organization' is
8      ↪ an error message
9      identifier = request_data.form.get("identifier", None)
10     date = request_data.form.get("date", None)
11
12     if identifier and not date: return jsonify({ "documents": [document for document in
13         ↪ os.listdir('metadatas') if document in repository[organization]['documents'] and
14         ↪ json.load(open(f"metadatas/{document}"))['creator'] == identifier]})
15     if not identifier and date: return jsonify({ "documents": [document for document in
16         ↪ os.listdir('metadatas') if document in repository[organization]['documents'] and
17         ↪ json.load(open(f"metadatas/{document}"))['create_data'] == date]})
18     if identifier and date: return jsonify({ "documents": [document for document in
19         ↪ os.listdir('metadatas') if document in repository[organization]['documents'] and
20         ↪ json.load(open(f"metadatas/{document}"))['creator'] == identifier and
21         ↪ json.load(open(f"metadatas/{document}"))['create_data'] == date]})
22     return jsonify({ "documents": [document for document in os.listdir('metadatas') if
23         ↪ document in repository[organization]['documents']]})

```

2.5 Authorized API Commands

Os comandos autorizados abrangem várias operações fundamentais que servem de manutenção ao repositório seguro, utilizando permissões e controle de acessos para além das sessões previamente usadas.

2.5.1 Rep Add Subject

Adiciona um novo sujeito ao repositório, prevenindo a adição de sujeitos já adicionados.

```

1  @app.route('/repository/organization/subjects/add_subject', methods=['POST'])
2  def rep_add_subject():
3      request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4      organization, session_id, username =
5      ↪ get_valid_session_information(request_data.files["session_file"],
6      ↪ request_data.form["previous_interaction"])
7      if not username: return jsonify(organization), 400 # In this case, 'organization' is
8      ↪ an error message
9      identifier = request_data.form["identifier"]
10     name = request_data.form["name"]
11     email = request_data.form["email"]
12     public_key = json.loads(request_data.files["credentials_file"])['pubkey']
13
14     if not validate_subject_permission(organization, session_id, username, 'SUBJECT_NEW'):
15         ↪ return jsonify({"error": "Permission denied"}), 400

```

```

12     if identifier in repository[organization]['subjects']: return jsonify({"error": "User
    ↪ already exists"}), 400
13
14     repository[organization]['subjects'][identifier] = {
15         "name": name,
16         "email": email,
17         "pubkey": public_key,
18         "roles": [],
19         "status": True
20     }
21     save_state()
22     return jsonify({"success": True})

```

2.5.2 Rep Suspend Subject

Suspende um sujeito específico, impedindo a suspensão no caso do sujeito já se encontrar suspenso.

```

1 @app.route('/repository/organization/subjects/suspend_subject', methods=['POST'])
2 def rep_suspend_subject():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
    ↪ request_data.form["previous_interaction"])
6     if not username: return jsonify(organization), 400 # In this case, 'organization' is
    ↪ an error message
7     identifier = request_data.form['identifier']
8
9     if not validate_subject_permission(organization, session_id, username,
    ↪ 'SUBJECT_DOWN'): return jsonify({"error": "Permission denied"}), 400
10    if identifier not in repository[organization]['subjects']: return jsonify({"error":
    ↪ "Invalid username"}), 400
11    if not repository[organization]['subjects'][identifier]['status']: return
    ↪ jsonify({"error": "Subject already suspended"}), 400
12
13    repository[organization]['subjects'][identifier]['status'] = False
14    save_state()
15    return jsonify({"success": True})

```

2.5.3 Rep Activate Subject

Reativa um sujeito, garantindo que este se encontrava previamente suspenso.

```

1 @app.route('/repository/organization/subjects/activate_subject', methods=['POST'])
2 def rep_activate_subject():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
    ↪ request_data.form["previous_interaction"])
6     if not username: return jsonify(organization), 400 # In this case, 'organization' is
    ↪ an error message
7     identifier = request_data.form['identifier']
8
9     if not validate_subject_permission(organization, session_id, username, 'SUBJECT_UP'):
    ↪ return jsonify({"error": "Permission denied"}), 400
10    if identifier not in repository[organization]['subjects']: return jsonify({"error":
    ↪ "Invalid username"}), 400
11    if repository[organization]['subjects'][identifier]['status']: return
    ↪ jsonify({"error": "Subject already active"}), 400
12
13    repository[organization]['subjects'][identifier]['status'] = True
14    save_state()
15    return jsonify({"success": True})

```

2.5.4 Rep Add Role

Criar uma nova 'role' numa organização, sem permissões associadas.

```
1 @app.route('/repository/organization/roles/add_role', methods=['POST'])
2 def rep_add_role():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5         ↪ get_valid_session_information(request_data.files["session_file"],
6         ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8         ↪ an error message
9     role = request_data.form["role"]
10
11     if not validate_subject_permission(organization, session_id, username, 'ROLE_NEW'):
12         ↪ return jsonify({"error": "Permission denied"}), 400
13     if role in repository[organization]["roles"]: return jsonify({"error": "Role already
14         ↪ exists"}), 400
15
16     repository[organization]['roles'][role] = {
17         "subjects": [],
18         "acl": [],
19         "permissions": [],
20         "status": True
21     }
22     save_state()
23     return jsonify({"success": True})
```

2.5.5 Rep Suspend Role

Suspende uma 'role' existente numa organização, verificando o 'status' da mesma previamente.

Correção: Previne a suspensão da 'role' "Managers".

```
1 @app.route('/repository/organization/roles/suspend_role', methods=['POST'])
2 def rep_suspend_role():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5         ↪ get_valid_session_information(request_data.files["session_file"],
6         ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8         ↪ an error message
9     role = request_data.form["role"]
10
11     if role == "Managers": return jsonify({"abort": "Role cannot be suspended"}), 400
12     if not validate_subject_permission(organization, session_id, username, 'ROLE_DOWN'):
13         ↪ return jsonify({"error": "Permission denied"}), 400
14     if role not in repository[organization]["roles"]: return jsonify({"error": "Invalid
15         ↪ role"}), 400
16     if not repository[organization]["roles"][role]['status']: return jsonify({"error":
17         ↪ "Role already suspended"}), 400
18
19     repository[organization]['roles'][role]['status'] = False
20     save_state()
21     return jsonify({"success": True})
```

2.5.6 Rep Reactivate Role

Reativa uma role existente numa organização, verificando o 'status' da mesma previamente.

```
1 @app.route('/repository/organization/roles/reactivate_role', methods=['POST'])
2 def rep_reactivate_role():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
```

```

4      organization, session_id, username =
    ↪ get_valid_session_information(request_data.files["session_file"],
    ↪ request_data.form["previous_interaction"])
5  if not username: return jsonify(organization), 400 # In this case, 'organization' is
    ↪ an error message
6  role = request_data.form["role"]
7
8  if not validate_subject_permission(organization, session_id, username, 'ROLE_UP'):
    ↪ return jsonify({"error": "Permission denied"}), 400
9  if role not in repository[organization]["roles"]: return jsonify({"error": "Invalid
    ↪ role"}), 400
10 if repository[organization]["roles"][role]['status']: return jsonify({"error": "Role
    ↪ already active"}), 400
11
12 repository[organization]['roles'][role]['status'] = True
13 save_state()
14 return jsonify({"success": True})

```

2.5.7 Rep Add Permission

Adiciona sujeitos ou permissões a uma 'role' específica numa organização, assegurando previamente a existência de ambos.

```

1  @app.route('/repository/organization/roles/add_permission', methods=['POST'])
2  def rep_add_permission():
3      request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4      organization, session_id, username =
    ↪ get_valid_session_information(request_data.files["session_file"],
    ↪ request_data.form["previous_interaction"])
5  if not username: return jsonify(organization), 400 # In this case, 'organization' is
    ↪ an error message
6  role = request_data.form["role"]
7  identifier = request_data.form["identifier"]
8
9  if not validate_subject_permission(organization, session_id, username, 'ROLE_MOD'):
    ↪ return jsonify({"error": "Permission denied"}), 400
10 if role not in repository[organization]["roles"]: return jsonify({"error": "Invalid
    ↪ role"}), 400
11
12 if identifier.isalnum():
13     if identifier not in repository[organization]['subjects']: return
    ↪ jsonify({"error": "Invalid username"}), 400
14     if identifier in repository[organization]["roles"][role]['subjects']: return
    ↪ jsonify({"error": "Role already assigned"}), 400
15     repository[organization]["roles"][role]['subjects'].append(identifier)
16     save_state()
17     return jsonify({"success": True})
18
19 if identifier in repository[organization]["roles"][role]['permissions']: return
    ↪ jsonify({"error": "Permission already exists"}), 400
20 if identifier in ["ROLE_NEW", "ROLE_DOWN", "ROLE_UP", "ROLE_MOD"]:
    ↪ repository[organization]["roles"][role]['permissions'].append(identifier)
21 elif identifier in ["DOC_NEW", "SUBJECT_NEW", "SUBJECT_DOWN", "SUBJECT_UP",
    ↪ "ROLE_ACL"]: repository[organization]["roles"][role]['acl'].append(identifier)
22 else: return jsonify({"error": "Invalid permission"}), 400
23 save_state()
24 return jsonify({"success": True})

```

2.5.8 Rep Remove Permission

Remove sujeitos ou permissões de uma 'role' específica numa organização, assegurando a existência dos mesmos previamente.

Para além das funcionalidades básicas, o sistema retorna um aviso, com a possibilidade de forçar o comando, caso o utilizador tente retirar uma permissão crítica associada a si mesmo (p.

ex: retirar "ROLE_MOD" da 'role' associada ao mesmo utilizador).

```
1 @app.route('/repository/organization/roles/remove_permission', methods=['POST'])
2 def rep_remove_permission():
3     request_data = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
6     ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8     ↪ an error message
9     role = request_data.form["role"]
10    identifier = request_data.form["identifier"]
11    force_flag = int(request_data.form["force_flag"])
12
13    if role not in repository[organization]["roles"]: return jsonify({"error": "Invalid
14    ↪ role"}), 400
15    if not validate_subject_permission(organization, session_id, username, 'ROLE_MOD'):
16    ↪ return jsonify({"error": "Permission denied"}), 400
17
18    if identifier.isalnum():
19        if role == "Managers" and len(repository[organization]["roles"][role]['subjects'])
20        ↪ == 2: return jsonify({"error": "Role must have at least 2 members"}), 400
21        if identifier not in repository[organization]["roles"][role]['subjects']: return
22        ↪ jsonify({"error": "Invalid username"}), 400
23        repository[organization]["roles"][role]['subjects'].remove(identifier)
24        if not force_flag and not validate_subject_permission(organization, session_id,
25        ↪ username, 'ROLE_MOD'):
26            repository[organization]["roles"][role]['subjects'].append(identifier)
27            return jsonify({"abort": "Use f/force to remove user admin privileges"}), 400
28        save_state()
29        return jsonify({"success": True})
30
31    if identifier == "ROLE_MOD" and role == "Managers": return jsonify({"error":
32    ↪ "Permission cannot be removed from role"}), 400
33    if identifier in repository[organization]["roles"][role]['permissions']:
34    ↪ repository[organization]["roles"][role]['permissions'].remove(identifier)
35    elif identifier in repository[organization]["roles"][role]['acl']:
36    ↪ repository[organization]["roles"][role]['acl'].remove(identifier)
37    else: return jsonify({"error": "Invalid permission"}), 400
38    if not force_flag and not validate_subject_permission(organization, session_id,
39    ↪ username, 'ROLE_MOD'):
40        repository[organization]["roles"][role]['permissions'].append(identifier)
41        return jsonify({"abort": "Use f/force to remove user admin privileges"}), 400
42    save_state()
43    return jsonify({"success": True})
```

2.5.9 Rep Add Doc

Adiciona um novo documento ao repositório associado a uma organização, garantindo a sua integridade, armazenando de forma segura os parâmetros críticos da 'metadata' referentes ao documento.

```
1 @app.route('/repository/organization/documents/add_doc', methods=['POST'])
2 def rep_add_doc():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
6     ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8     ↪ an error message
9     document_name = request_data.form["document_name"]
10    file_content = request_data.files['file']
11    metadata = request_data.form['metadata']
12
13    if not validate_subject_permission(organization, session_id, username, 'DOC_NEW'):
14    ↪ return jsonify({"error": "Permission denied"}), 400
```

```

11     if document_name in repository[organization]['documents'] or
12     ↪ os.path.exists(f"files/{metadata['file_handle']}"): return jsonify({"error": "File
13     ↪ already exists"}), 400
14     os.makedirs(f"files/", exist_ok=True)
15     os.makedirs(f"metadatas/", exist_ok=True)
16     with open(f"files/{metadata['file_handle']}", 'wb') as file: file.write(file_content)
17     with open(f"metadatas/{document_name}", 'w') as metadata_file:
18         json.dump({
19             "document_handle": None,
20             "create_date": datetime.now().isoformat(),
21             "creator": username,
22             "file_handle": metadata['file_handle'],
23             "acl": {"Managers": ["DOC_ACL", "DOC_READ", "DOC_DELETE"]},
24             "deleter": None,
25             "key":
26                 ↪ repUtils.decode_bytes2string(repUtils.encrypt_easy(load_pem_public_key(repository[organization]
27                 ↪ backend=default_backend()), repUtils.encode_string2bytes(metadata['key'])),
28                 ↪ "AES")),
29             "algorithm": metadata["algorithm"],
30         }, metadata_file)
31
32     repository[organization]['documents'].append(document_name)
33     save_state()
34     return jsonify({"success": True})

```

2.5.10 Rep Get Doc Metadata

Obtém a 'metadata' de um documento específico associado a uma organização, decifrando os parâmetros necessários, mas assegurando a transmissão segura dos dados para o cliente.

```

1  @app.route('/repository/organization/documents/get_doc_metadata', methods=['GET'])
2  def rep_get_doc_metadata():
3      request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4      organization, session_id, username =
5      ↪ get_valid_session_information(request_data.files["session_file"],
6      ↪ request_data.form["previous_interaction"])
7      if not username: return jsonify(organization), 400 # In this case, 'organization' is
8      ↪ an error message
9      document_name = request_data.form["document_name"]
10
11      if not validate_document_permission(organization, session_id, document_name,
12      ↪ 'DOC_READ'): return jsonify({"error": "Permission denied"}), 400
13      if document_name not in repository[organization]['documents']: return
14      ↪ jsonify({"error": "Invalid document"}), 400
15
16      with open(f"metadatas/{document_name}", 'r') as metadata_file:
17          metadata_file = metadata_file.read()
18
19          content = repUtils.encrypt_easy(repository[organization][username]["pubkey"],
20          ↪ metadata_file.encode(), "AES")
21          mac = repUtils.mac_sign_easy(rep_private_key, content)
22          s = len(mac).to_bytes(20, byteorder='big')
23
24          buff = io.BufferedReader(io.BytesIO(s + mac + content))
25          return send_file(buff, as_attachment=True, download_name="save")

```

2.5.11 Rep Get Doc File

Obtém um documento específico associado a uma organização, assim como a sua 'metadata', assegurando a transmissão segura dos mesmos.

```

1  @app.route('/repository/organization/documents/get_doc_file', methods=['GET'])
2  def rep_get_doc_file():

```



```

3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
6     ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8     ↪ an error message
9     document_name = request_data.form["document_name"]
10    info = request_data.form["info"]
11
12    if not validate_document_permission(organization, session_id, document_name,
13    ↪ 'DOC_READ'): return jsonify({"error": "Permission denied"}), 400
14    if document_name not in repository[organization]['documents']: return
15    ↪ jsonify({"error": "Invalid document"}), 400
16
17    if info == "meta":
18        with open(f"metadatas/{document_name}", 'r') as metadata_file:
19            metadata_file = json.loads(metadata_file)
20            return jsonify({"res": metadata_file})
21
22    if info == "file":
23        with open(f"files/{metadata_file['file_handle']}", 'r') as file:
24            content = file.read()
25            mac = repUtils.mac_sign_easy(rep_private_key, content)
26            s = len(mac).to_bytes(20, byteorder='big')
27
28            buff = io.BufferedReader(io.BytesIO(s + mac + content))
29            return send_file(buff, as_attachment=True, download_name="save")

```

2.5.12 Rep Delete Doc

Remove um documento através da destruição da associação do ficheiro com a sua 'metadata', e não propriamente o conteúdo do documento. Além disso, remove ainda o documento da lista de documentos da organização ao qual estava associado.

```

1 @app.route('/repository/organization/documents/delete_doc', methods=['POST'])
2 def rep_delete_doc():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5     ↪ get_valid_session_information(request_data.files["session_file"],
6     ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8     ↪ an error message
9     document_name = request_data.form["document_name"]
10
11    if not validate_document_permission(organization, session_id, document_name,
12    ↪ 'DOC_DELETE'): return jsonify({"error": "Permission denied"}), 400
13    if document_name not in repository[organization]['documents']: return
14    ↪ jsonify({"error": "Invalid document"}), 400
15
16    if os.path.isdir('metadatas') : os.makedirs("metadatas", exist_ok=True)
17
18    with open(f"metadatas/{document_name}", 'r') as metadata_file:
19        metadata = json.load(metadata_file)
20        file_handle = metadata["file_handle"]
21        metadata["file_handle"] = None
22    with open(f"metadatas/{document_name}", 'w') as metadata_file:
23        json.dump(metadata, metadata_file)
24    repository[organization]['documents'].remove(document_name)
25    save_state()
26    return jsonify({"file_handle": file_handle})

```

2.5.13 Rep Acl Doc

Altera as permissões de controlo de acesso (ACL) de um documento específico, protegendo a permissão "DOC_ACL" da 'role' "Managers".

```
1 @app.route('/repository/organization/documents/acl_doc', methods=['POST'])
2 def rep_acl_doc():
3     request_data : Req = repUtils.decrypt_request(rep_private_key, request)
4     organization, session_id, username =
5         ↪ get_valid_session_information(request_data.files["session_file"],
6         ↪ request_data.form["previous_interaction"])
7     if not username: return jsonify(organization), 400 # In this case, 'organization' is
8         ↪ an error message
9     document_name = request_data.form["document_name"]
10    operation = int(request_data.form["operation"])
11    role = request_data.form["role"]
12    permission = request_data.form["permission"]
13
14    if not validate_document_permission(organization, session_id, document_name,
15        ↪ 'DOC_ACL'): return jsonify({"error": "Permission denied"}), 400
16    if document_name not in repository[organization]['documents']: return
17        ↪ jsonify({"error": "Invalid document"}), 400
18
19    with open(f"metadatas/{document_name}", 'r') as metadata_file:
20        metadata = json.load(metadata_file)
21        if role not in metadata['acl']: return jsonify({"error": "Invalid role"}), 400
22        if operation:
23            if permission in metadata['acl'][role]: return jsonify({"error": "Permission
24                ↪ already exists"}), 400
25            metadata['acl'][role].append(permission)
26        else:
27            if permission == "DOC_ACL" and role == "Managers": return jsonify({"error":
28                ↪ "Permission cannot be removed from role"}), 400
29            if permission not in metadata['acl'][role]: return jsonify({"error": "Invalid
30                ↪ permission"}), 400
31            metadata['acl'][role].remove(permission)
32
33    if os.path.exists("metadatas/"): os.makedirs("metadatas/", exist_ok=True)
34    with open(f"metadatas/{document_name}", 'w') as metadata_file:
35        json.dump(metadata, metadata_file)
36    save_state()
37    return jsonify({"success": True})
```

Capítulo 3

Application Security Verification Standards (ASVS)

De acordo com o que foi proposto, tendo em conta o ASVS 4.0.3, foram escolhidos padrões de verificação dos conteúdos do capítulo 6 do ASVS, que descrevem padrões referentes a cifragem armazenada.

1. 6.1.1 : Os ficheiros pessoais estão cifrados, excepto a 'metadata'. No entanto, o sistema está preparado para cifrar essa informação (com alterações mínimas). (não verificado)
2. 6.1.2 e 6.1.3 : todos os ficheiros não referentes a 'metadata' são cifrados.(verificado)
3. 6.2.1 : Apesar dos erros serem manuseados na sua maioria, não é possível garantir proteção contra Padding Oracle attacks. (não verificado)
4. 6.2.2 : Cifras programadas por defeito são utilizadas. (não verificado)
5. 6.2.3 : (não verificado)
6. 6.2.4 : É possível trocar algoritmos e tamanhos de chave com certa facilidade no código, mas não é uma função disponível para o utilizador. (discutível)
7. 6.2.5 : Apesar de algoritmos menos seguros serem possíveis de serem utilizados, por defeito, o programa utiliza algoritmos mais seguros. (verificado)
8. 6.2.6 : Os números aleatórios criados são adaptados para cada algoritmo. Porém, apesar da chance de repetição ser baixa, não é garantidamente nula. (não verificado)
9. 6.2.7 : Autenticação com MAC é utilizada apenas para transmissão de ficheiros. MIC's são utilizados para os ficheiros guardados. (verificado)
10. 6.2.8 : (não verificado)
11. 6.3.1 : É utilizado OS.random() do python que garante imprevisibilidade na criação de números aleatórios. (verificado)
12. 6.3.2 : (não verificado)
13. 6.3.3 : (não verificado)
14. 6.4.1 : (não verificado)
15. 6.4.2 : (não verificado)

Dos 16 padrões abordados 6 são cumpridos pelo sistema criado.