**Relatório**

**Key Management Service**

Disciplina: Segurança e Aplicações de Hardware Confiável

Matheus Rodrigues Bezerra

Ricardo António Araújo Martins

Rodrigo Alçada Aguiar Branco

**Introdução**

Este relatório apresenta o desenvolvimento de um projeto para a disciplina de Segurança e Aplicações de Hardware Confiável, cujo objetivo central é o desenho de um sistema de gerenciamento de chaves (Key Management Service – KMS) seguro e eficiente. O sistema proposto contempla múltiplos componentes interligados: usuários, servidores, HSMs e administradores, cada qual com papéis e níveis de confiança distintos. Enquanto usuários e HSMs são considerados implicitamente confiáveis, os administradores podem não ser totalmente confiáveis, o que impõe desafios ao desenho do sistema de confiança e à gestão das permissões.

Ao longo deste relatório, serão detalhadas as entidades envolvidas, as premissas de confiança adotadas e as estratégias propostas para garantir que o gerenciamento e o uso das chaves criptográficas ocorram de forma segura, mesmo em cenários onde nem todos os administradores podem ser plenamente confiáveis. A abordagem apresentada visa não apenas atender aos requisitos funcionais do sistema, mas também mitigar riscos e fortalecer a segurança das operações realizadas pelos usuários na nuvem.

**Entidades Necessárias – Interesses, Funções e Características**

Utilizadores: Solicitarão operações criptográficas de arquivos, isto é, cifrar, decifrar, assinar e verificar assinaturas. Eles são implicitamente confiáveis.

Administradores: Decidirão quais usuários podem usar quais HSMs, gerenciando as chaves dos mesmos. Devemos considerar os administradores como uma entidade não confiável.

Servidores: Receberão solicitações do lado do cliente e as encaminharão aos HSMs para processamento, não podendo armazenar dados referentes aos arquivos enviados.

HSMs: Armazenarão e gerenciarão as chaves dos clientes. O comportamento destes é considerado confiável. É importante ressaltar que os HSMs têm uma visão extremamente limitada de todo o sistema, mantendo apenas suas próprias chaves mestras para criptografar, descriptografar e assinar dados.

**HSM**

O HSM é a peça chave para o desenvolvimento deste sistema de forma segura e garantir que todas as funcionalidades existentes ocorram de forma confiável.

Pode-se definir um HSM como uma unidade de hardware que armazena chaves criptográficas para mantê-las privadas e, ao mesmo tempo, garantir que estejam disponíveis para aqueles autorizados a usá-las. O principal objetivo da segurança do HSM é controlar quais indivíduos têm acesso às chaves de segurança digitais de uma organização. Este componente impede que uma aplicação carregue uma cópia de uma chave privada na memória de um servidor web. Isso é útil porque, enquanto estiver a rodar um servidor, suas chaves de segurança ficam vulneráveis ​​a hackers. Se um invasor obtiver acesso ao servido, ele conseguirá localizar as chave e usá-la para acessar dados confidenciais.

**Objetivos**

O objetivo deste projeto é desenvolver um sistema para gestão de chaves, com entidades bem definidas. Além disso, foram adicionadas novas funcionalidades ao sistema, como a redundância e o registro de ocorrências (logs), que serão explicadas a seguir.

**Redundância**

No cenário atual, a necessidade de sistemas redundantes torna-se cada vez mais evidente diante do aumento das ameaças cibernéticas, da complexidade das infraestruturas digitais e da crescente dependência das organizações em relação à tecnologia. Com o avanço acelerado da digitalização e a migração massiva para ambientes em nuvem, empresas de todos os portes enfrentam riscos significativos de interrupção de serviços, perda de dados e ataques sofisticados, como ransomware, invasões à cadeia de suprimentos e até mesmo destruição física de componentes. Nesse contexto, a redundância configura-se como uma estratégia essencial para garantir a continuidade dos negócios e a resiliência operacional.

A utilização de sistemas redundantes, como múltiplos módulos de segurança de hardware (HSMs) para criptografia, possibilita a manutenção do funcionamento dos serviços mesmo diante de falhas inesperadas em componentes críticos, evitando interrupções perceptíveis aos usuários. Tal abordagem é especialmente relevante em ambientes nos quais a disponibilidade e a integridade das informações são vitais, como nos setores financeiro, governamental e de saúde.

A arquitetura proposta prevê a utilização de múltiplos HSMs para criar redundância nas operações criptográficas, com o objetivo de garantir disponibilidade contínua e tolerância a falhas. Esse princípio proporciona os seguintes benefícios:

* **Resiliência a falhas:** Eliminação de pontos únicos de falha (SPOF), aspecto crítico para sistemas que exigem alta disponibilidade.
* **Balanceamento de carga**: Distribuição das operações criptográficas (criptografia, descriptografia, assinatura) entre os HSMs disponíveis, otimizando o desempenho e evitando gargalos.
* **Failover automático**: Em caso de falha ou indisponibilidade de um HSM, outro dispositivo assume imediatamente as funções, garantindo a continuidade das operações sem interrupções.

**Registro de Ocorrências**

A presença de um sistema de logs em uma arquitetura baseada em múltiplos HSMs é fundamental para garantir a segurança, a rastreabilidade e a conformidade operacional do ambiente. Os logs registram todas as ocorrências relevantes, como operações de criptografia e descriptografia, tentativas de acesso, alterações de configuração e falhas de autenticação, criando, assim, um histórico detalhado das atividades realizadas no sistema.

**Desenvolvimento**

**Protótipo – Características Básicas**

O projeto foi desenvolvido utilizando Node.js, versão 20.11.0, e emprega o banco de dados PostgreSQL 17. Para a realização dos testes, foi utilizada a ferramenta Postman.

**Protótipo – Entidades**

Com isso, estão esclarecidos os objetivos que o sistema deve atender. A partir dessas definições, foi desenvolvido um sistema capaz de cumprir todas essas demandas. A principal mudança foi a criação de dois tipos distintos de servidores, descritos a seguir:

1. **Servidor Principal (MainServer)**: Responsável por receber comandos dos utilizadores e administradores, realizar a autenticação dos utilizadores, enviar e-mails, armazenar e gerenciar a base de dados, além de redistribuir os pedidos de operações criptográficas aos Servidores HSM, cujas funcionalidades serão detalhadas posteriormente.
2. **Servidor HSM (HSMServer)**: Nestes servidores que estão instalados os HSMs, que é onde são realizadas as operações criptográficas solicitadas pelos utilizadores. Eles possuem comunicação exclusiva com o MainServer, ou seja, qualquer operação executada por este servidor foi previamente filtrada e encaminhada pelo MainServer. É possível implementar quantos Servidores HSM forem necessários, sendo o mínimo recomendado de dois servidores.
3. **HSMs**: Instalados nos Servidores HSM, são responsáveis pela execução das operações criptográficas, conforme as solicitações recebidas dos respectivos Servidores HSM.
4. **Administradores**: Responsáveis pela gestão dos utilizadores, incluindo a criação e exclusão de contas. Além disso, têm acesso a informações sobre a base de dados, logs de registro e o estado dos servidores.
5. **Utilizadores**: Possuem a capacidade de definir sua própria senha a partir do momento em que a conta é criada pelo administrador. Também realizam a principal função, que é realizaer operações criptográficas, tais como cifrar, decifrar, assinar e verificar assinaturas.

Os detalhes específicos de cada entidade serão explicados de forma mais detalhada nas seções seguintes deste relatório.

**Protótipo – Disposição**

O Proof-of-Concept esboçado foi um sistema que possui 3 HSM Servers, que foram denominados em hsm1, hsm2 e hsm3: possuindo o seguinte escopo:Diagrama

O conteúdo gerado por IA pode estar incorreto.

**Protótipo – Comunicação**

Para comunicação externa, todos os acessos, tanto de usuários quanto de administradores, ao Main Server ocorrem por meio de canais HTTPS, garantindo a proteção dos dados em trânsito contra interceptação e ataques do tipo man-in-the-middle. Por outro lado, a comunicação entre o Main Server e os HSM Servers é realizada via HTTP, devido a limitações de projeto. Contudo, foi implementada uma restrição que permite a esses servidores receberem requisições exclusivamente provenientes do Main Server.

**Protótipo – Base de Dados**

Para o Main Server, foi criada uma base de dados em PostgreSQL com o objetivo de armazenar dados de autenticação e registro de logs. A base de dados contém as seguintes tabelas:

**Users**

Esta tabela tem como finalidade armazenar os dados dos utilizadores, permitindo a realização de ações de autenticação e o encaminhamento de funções para os servidores HSM. Os principais campos desta tabela são:

1. **Id:** Número de identificação.
2. **Name:** String única do utilizador
3. **Email:** String com o endereço de e-mail do utilizador, que será utilizado para definição da password.
4. **Password:** String que fica armazenado o hash da password do utilizador.
5. **Salt:** Valor aleatório utilizado junto com a senha para aumentar a segurança do hash da senha.
6. **Role\_id:** Identificador da função do usuário no sistema, isto é, se for 1, é um utilizador e se for 2, é um administrador.
7. **Status:** Indica o status do usuário (ativo ou deletado).
8. **Created\_by:** Para fins de auditoria, fica registrado que administrador criou este utilizador.
9. **Server\_name\_1:** Nome do primeiro HSM server que este utilizador possui chaves criptográficas.
10. **Key\_Reference\_1:** Id de referência das chaves deste utilizador no server\_name\_1.
11. **Server\_name\_2:** Nome do segundo HSM server que este utilizador possui chaves criptográficas.
12. **Key\_Reference\_2:** Id de referência das chaves deste utilizador no server\_name\_2.

**Logs**

Esta tabela tem como objetivo armazenar todas as ações requisitadas ao Main Server, bem como as informações relativas às suas respectivas respostas. Essa abordagem foi adotada para evitar, por exemplo, a necessidade de armazenar os ficheiros enviados, permitindo, ainda assim, o registro das ocorrências processadas pelo servidor. Dessa forma, a tabela possui os seguintes campos:

1. **Id:** Número de identificação.
2. **UserId:** Referência ao utilizador que fez esta requisição.
3. **Endpoint:** Tipo de requisição que o utilizador ou administrador fez.
4. **Ip\_address:** endereço de IP da requisição.
5. **Status:** Status da resposta.
6. **Created\_at:** Instante da ocorrência.

**Outras tabelas**

Também existe a tabela **Roles**, responsável por armazenar os valores referentes aos tipos de utilizador, como "Utilizador" e "Administrador". Além disso, há a tabela **Verification\_Codes**, que é utilizada para guardar os códigos de verificação necessários para que o utilizador possa definir a sua palavra-passe.

**Protótipo – Funções**

**Autenticação**

Exceto para a ação “User – Set Password”, todas as demais ações disponíveis no servidor requerem autenticação. Para isso, é necessário realizar uma requisição POST para o endpoint /auth/login, utilizando o seguinte formato para o corpo da requisição:

Uma imagem contendo Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

Em caso de autenticação bem-sucedida, é retornado um Bearer Token contendo os dados de userId e a validade do referido token. Este token deve ser informado no campo Authorization em todas as requisições HTTP/HTTPS. A validade do token foi configurada para apenas 1 hora, considerando-se que a grande maioria dos usuários não utilizará o servidor mais do que algumas vezes durante uma sessão.

**Criar utilizadores**

Esta função é exclusiva para uso de administradores, que enviará ao servidor os seguintes campos:

Uma imagem contendo Texto

O conteúdo gerado por IA pode estar incorreto.

A partir deste pedido, o sistema terá o seguinte workflow:

Texto

O conteúdo gerado por IA pode estar incorreto.

Este código de verificação será utilizado pelo utilizador para definir a sua password posteriormente. Optou-se pelo envio do código por e-mail ao utilizador, a fim de evitar qualquer influência do administrador na escolha da password, seguindo o princípio de que este pode não ser considerado totalmente confiável.

**Deletar utilizadores**

Esta função é exclusiva para uso de administradores e tem como objetivo enviar ao servidor o parâmetro **id**, alterando o status do respectivo utilizador para ‘deleted’, sem, contudo, removê-lo da base de dados. Dessa forma, caso a função seja utilizada de maneira indevida por um administrador, o único impacto possível será impedir que o utilizador realize operações com o servidor, o que pode ser facilmente revertido, sem perda de dados.

**Outras Funções do Administrador**

1. **Get Users**: Esta função é responsável por listar todos os usuários cadastrados no sistema.
2. **Get Logs**: Esta função permite consultar os logs do sistema com diversos filtros, aceitando parâmetros opcionais via query string, como userId, endpoint, status, startDate e endDate.
3. **Clean Logs**: Função responsável pela limpeza de logs antigos do sistema, requerendo uma data limite (beforeDate) no corpo da requisição, mas não permite limpar logs mais recentes que 2 meses, pois entende-se que 2 meses impediria o administrador apagar logs de ocorrências maliciosas.
4. **Get Status**: Função complexa que monitora o estado geral do sistema: coletando informações do servidor principal (MainServer), além de fazer ping em cada servidor HSM registrado e registrando se ele está online ou offline.

**Set Password**

Função destinada exclusivamente a novos utilizadores. Após a criação do utilizador pelo administrador, é enviado um e-mail contendo um código de verificação. O utilizador deverá então submeter um pedido ao Main Server utilizando o endpoint “/user/set-password/:code”, onde ":code" corresponde ao código de verificação recebido. No corpo da requisição, deve ser incluído o campo ‘newpwd’, no qual o utilizador define a sua nova palavra-passe. Após o envio do pedido, ocorre o seguinte fluxo de trabalho:

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Portanto, após o utilizador definir uma password, o Main Server envia a 2 HSM Servers, escolhidos de forma aleatória, pedidos para gerar novos conjuntos de chaves, e como resposta, recebe um keyId de cada HSM Server e adicionará estas informações para a base de dados, e finalmente, retorna ao utilizador que a definição de password foi bem-sucedida.

**Assinar ficheiros**

Esta função é exclusiva para utilizadores e permite assinar ficheiros, o algoritmo utilizado é o SHA-256 e o RSA utilizando a chave privada do utilizador, garantindo integridade e autenticidade. Para gerar a assinatura, deve ser enviado o ficheiro que está para ser assinado, são chamados os dois HSMs que o utilizador tem chaves criptográficas registradas, cada um retornando uma assinatura distinta, com o MainServer unindo as duas posteriormente, como pode ser visto abaixo:

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Dessa forma, garante-se que, caso algum dos servidores HSM utilizados pelo usuário esteja fora do ar, ainda será possível validar a assinatura. Por motivos lógicos, a assinatura é atribuída ao valor do userId presente no token de autenticação, com o objetivo de evitar fraudes nas assinaturas.

**Verificar Assinaturas**

Esta função é exclusiva para utilizadores e permite verificar se a assinatura de um ficheiro foi realmente efetuada por determinado utilizador. Para a execução desta função, devem ser enviados o ficheiro, a assinatura que se pretende verificar e o nome do utilizador que declara ser o autor do ficheiro, seguindo a seguinte estrutura:

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

O workflow que será mostrado irá supor que o hsm1 tenha sido destruído para que assim, seja demonstrado como que funcionaria nesta hipótese:

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Assim, consegue-se perceber a importância dessa arquitetura, possibilitando que mesmo com algum HSM Server comprometido, as operações do sistema conseguiriam contornar a situação e conseguir prover a funcionalidade ao utilizador.

**Cifrar**

Esta função é exclusiva para utilizadores e permite cifrar ficheiros para o utilizador que for desejado, para conseguir efetuar este comando é necessário enviar o ficheiro e qual utilizador (name) poderá decifrar este ficheiro, a criptografia adotada neste modelo foi o RSA sendo uma criptografia assimétrica. O workflow é bastante similar ao que ocorre na assinatura de ficheiros, como pode ser visto a seguir:

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Com este modelo, de cifra permite criar cifras para qualquer utilizador que exista no sistema.

**Decifrar**

Esta função é exclusiva para utilizadores e permite decifrar ficheiros, o utilizador precisa apenas enviar a cifra, e a partir do token de autenticação o Main Server encontra o id de utilizador, o fluxograma desta função funciona da seguinte forma:

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Desta forma, garantimos a confidencialidade de um ficheiro e que apenas o destinatário requerido no momento de cifrar o ficheiro conseguirá decifrá-lo.

**Trabalhos Futuros**

Mesmo com um projeto com implementações bastante consideráveis, o grupo que desenvolveu este protótipo acredita que seria possível implementar mais funcionalidades, como:

1. Utilizar um HSM no próprio Main Server para lidar com assinaturas de tokens, e-mails e no HTTPS, garantindo uma camada extra de autenticidade ao protótipo.
2. Os HSM Servers poderiam também servir como backup da base de dados do Main Server, possibilitando que se o Main Server fosse fisicamente destruído e não ficasse mais operável, facilmente conseguiria ser reposto por qualquer outro servidor configurável.
3. Adicionar Plano de Contingência para a perda do Main Server que fosse extremamente rápido e eficaz pois é o único servidor que ficar fora do ar, todo o sistema fica fora do ar, devido ao fato de guardar apenas o back-end, a base de dados e detalhes de configuração, isto seria completamente viável.
4. Adicionar um plano de Contingência para a perda de algum HSM Server, mesmo com a perda de algum HSM Server, não necessariamente as funcionalidades são perdidas, porém é necessário lidar com questões de redistribuir as chaves de utilizadores entre os HSM que ainda restam a funcionar.
5. Como o sistema de envio de e-mails já está implementado, poderia ser mais-valia adicionar mais funcionalidades que exploram isto, como ao cifrar um ficheiro para um utilizador, ele pode receber um e-mail com este ficheiro cifrado.
6. Cogitou-se em armazenar as passwords dos utilizadores com cifras obtidas com operações do HSM, porém percebeu-se que não faria sentido criar algo desta forma considerando que ao realizar um hash com salt já implicaria numa segurança aceitável.
7. Implementar criptografia simétrica, o projeto acabou utilizando apenas criptografia assimétrica, mas por questões de eficiência, o ideal é utilizar criptografia simétrica, como o AES, que é consideravelmente mais rápido, esta questão não foi abordada durante o desenvolvimento porque o RSA já estava a correr bem e, por isso, foi decidido dar atenção a outras questões.
8. Comunicação em HTTPS entre o Main Server e os HSMs Servers, isto não foi imposto por causa de limitações na implementação.

**Conclusão**

Finalmente, acredita-se que o protótipo criado é capaz de cumprir os requisitos exigidos pelo tema do projeto, que era o de criar um Key Management Service (KMS) garantindo a segurança permitida pelo uso de HSMs e seguindo boas práticas, como a principal delas, as chaves privadas presentes nos HSMs em momento algum saem do HSM. Quanto aos objetivos estabelecidos pelo grupo, a redundância já está a funcionar e cumpre u sua função de lidar com a condição do caso de algum HSM ser completamente perdido e seus dados ficarem irrecuperáveis, mas acreditamos que é possível melhorar a eficácia deste quesito. e quanto a obtenção de logs para auditoria, já acreditamos que se consegue guardar informações relevantes sem comprometer a privacidade e informações sensíveis dos utilizadores, como informações sobre ficheiros. O projeto tem bom aspecto e consegue funcionar como era proposto no início do desenvolvimento.

**Referências**

<https://www.fortinet.com/resources/cyberglossary/hardware-security-module>

<https://www.entrust.com/resources/learn/what-are-hardware-security-modules>

<https://utimaco.com/data-protection/gp-hsm>