Nota Técnica da Arquitetura Criptográfica

Versão descritiva e detalhada

Objetivo do documento

Este documento descreve de forma detalhada a arquitetura criptográfica adotada no Cifrei. O texto foi redigido para servir tanto como registro técnico do projeto quanto como material de compreensão para leitores interessados no tema, mesmo quando não possuam formação específica em criptografia. O foco é explicar o raciocínio da arquitetura, o fluxo operacional de cifragem e decifragem, os parâmetros que acompanham o código final, os ganhos práticos da solução e os limites naturais do modelo.

1. Introdução

O Cifrei foi concebido para resolver um problema clássico de aplicações de criptografia voltadas ao usuário comum: aumentar a segurança do processamento sem transformar a experiência de uso em algo pesado, lento ou tecnicamente intimidador. Em muitos sistemas, segurança e praticidade parecem forças opostas. Quando os parâmetros criptográficos ficam leves demais, o uso se torna agradável, mas a resistência a ataques diminui. Quando ficam agressivos demais, a proteção teórica aumenta, porém a operação se torna desconfortável e, em casos extremos, impraticável em dispositivos modestos. A arquitetura do Cifrei parte da premissa de que esse dilema não deve ser resolvido com escolhas arbitrárias, mas com uma engenharia que faça cada custo de usabilidade produzir um ganho de segurança efetivo e mensurável.

A solução adotada combina dois componentes centrais. O primeiro é o Argon2id, utilizado como função de derivação de chave. O segundo é o algoritmo AES no modo GCM, usado para realizar a criptografia autenticada da mensagem. Em termos simples, o Argon2id transforma entradas humanas em uma chave criptográfica (uma sequência de bytes adequada para uso em algoritmos de criptografia) robusta; o AES-GCM usa essa chave para cifrar a mensagem e, ao mesmo tempo, garantir que qualquer alteração indevida no conteúdo seja detectada no momento da decifragem.

Ao redor desses dois componentes existe uma camada de engenharia de produto que define o comportamento real do sistema. Essa camada inclui a calibração local por dispositivo, o registro dos parâmetros necessários no próprio código gerado e um formato de *payload* (vide abaixo) compacto pensado para não ampliar desnecessariamente o tamanho final da cifra. O resultado é uma arquitetura que

procura ser moderna sob o ponto de vista criptográfico e disciplinada sob o ponto de vista de experiência do usuário.

2. Conceitos fundamentais e glossário mínimo

Antes de descrever o fluxo completo, convém esclarecer alguns termos que aparecem com frequência ao longo deste documento. A expressão **KDF** vem do inglês *Key Derivation Function*, que pode ser traduzido como função de derivação de chave. Uma KDF recebe dados de entrada que podem ser razoáveis para seres humanos, como senhas, frases ou combinações de segredos, e os transforma em uma chave criptográfica apta para uso. O objetivo é dificultar ataques de tentativa e erro, fazendo com que cada hipótese de senha custe tempo e, dependendo do algoritmo, também memória.

A palavra ***salt*** designa um valor aleatório gerado para uma operação específica. Esse valor é combinado com a entrada do processo de derivação de chave para impedir que duas operações iguais produzam exatamente o mesmo resultado. Em linguagem prática, o *salt* ajuda a tornar cada cifragem independente, mesmo quando o usuário reaproveita credenciais humanas idênticas.

A sigla **IV** significa *Initialization Vector*, normalmente traduzida como vetor de inicialização. No caso do AES-GCM, ele é um número aleatório ou quase aleatório associado a cada operação. Sua função não é substituir a chave, mas garantir que duas mensagens cifradas sob a mesma chave não resultem, por repetição mecânica, no mesmo padrão de saída.

A sigla **AES** corresponde a *Advanced Encryption Standard*, um padrão amplamente aceito de criptografia simétrica. A expressão "simétrica" significa que a mesma chave básica, derivada previamente, é usada no processo de cifrar e de decifrar. Já a sigla GCM significa *Galois/Counter Mode*. Trata-se de um modo de operação do AES que fornece não apenas confidencialidade, mas também integridade e autenticação. Em termos práticos, isso significa que o sistema não apenas esconde a mensagem de terceiros, mas também detecta manipulações indevidas no conteúdo cifrado.

Neste documento, a palavra ***payload*** é usada como sinônimo técnico do que, na experiência do usuário do Cifrei, aparece como o código final produzido pela cifragem. Esse código é a representação serializada de tudo que será necessário para reproduzir a operação de decifragem: versão de formato, parâmetros mínimos do processo de derivação de chave, *salt*, IV e texto cifrado autenticado.

3. Materiais de entrada e modelo lógico do Cifrei

No modelo operacional do Cifrei, a cifragem nasce da combinação de três elementos percebidos pelo usuário: a mensagem em texto aberto, a chave e a frase segredo. A mensagem é o conteúdo que se deseja proteger. A chave e a frase segredo são fatores humanos de entrada. Embora, do ponto de vista da interface, ambas possam parecer apenas campos de texto, a arquitetura as trata como componentes distintos de um material de derivação que será convertido em chave criptográfica.

O processo de cifragem pode ser descrito em três etapas principais:

1. Inicialmente, a frase segredo e a chave fornecidas pelo usuário são combinadas e utilizadas como entrada para o algoritmo Argon2id, juntamente com um valor aleatório denominado *salt* e um conjunto de parâmetros utilizados para dificultar a ação de atacantes *(hackers* que pretendem

"quebrar" a criptografia). Esse processo gera uma chave criptográfica, representada como uma sequência de bytes adequada para uso em algoritmos de criptografia.

2. Em seguida, essa chave criptográfica é utilizada no algoritmo AES no modo GCM (Galois/Counter Mode), juntamente com a mensagem a ser protegida e um vetor de inicialização (IV) aleatório. O resultado dessa operação é o texto cifrado, acompanhado de um mecanismo de autenticação que garante a integridade dos dados.
3. Por fim, todas as informações necessárias para a decifragem — incluindo o *salt*, o IV, os parâmetros do Argon2id e o próprio texto cifrado — são organizadas em uma estrutura compacta, formando o código final que pode ser armazenado ou transmitido.

É importante notar que a mensagem em si não participa da geração da chave derivada. A mensagem só entra em cena depois que a chave robusta já foi produzida. Essa separação é conceitualmente importante. A arquitetura não mistura a mensagem ao processo de derivação de chave para evitar ambiguidades lógicas e para permitir que a segurança da cifra dependa, de forma principal, dos segredos fornecidos pelo usuário e dos parâmetros aleatórios de cada operação. Assim, o fluxo correto deve ser entendido da seguinte forma: a chave e a frase segredo, combinadas segundo a lógica interna do aplicativo e temperadas por um *salt* aleatório, alimentam o Argon2id; o resultado desse processo é a chave criptográfica utilizada pelo AES-GCM para cifrar a mensagem.

Em razão disso, quando se afirma informalmente que o sistema usa "mensagem, chave, frase segredo, *salt* e IV para gerar o código", essa afirmação está correta apenas se entendida como descrição do fluxo completo e não como descrição do mecanismo de derivação de chave. Em termos estritamente técnicos, o papel da mensagem e o papel do material de derivação são distintos. A mensagem é objeto da cifra; a chave e a frase segredo são insumos da derivação da chave criptográfica.

4. Argon2id: o coração da derivação de chave

O Argon2id é uma função moderna de derivação de chave desenhada para dificultar ataques de força bruta em larga escala. Seu diferencial não está apenas em exigir tempo de processamento para cada tentativa, mas também em exigir memória de maneira relevante. Isso importa porque atacantes modernos não contam apenas com processadores tradicionais; eles podem empregar placas gráficas ou hardware especializado para testar um grande número de hipóteses rapidamente. Algoritmos que consomem memória de forma mais intensa tornam esse tipo de paralelização muito mais caro.

A parte final do nome, "id", indica que o algoritmo combina propriedades de duas variantes históricas do Argon2. De um lado, ele preserva características que o tornam mais adequado diante de certos cenários de observação lateral do processamento; de outro, mantém um comportamento robusto diante de ataques de tentativa e erro em ambiente *offline*. Na prática do Cifrei, isso o transforma em uma escolha particularmente boa para um aplicativo que roda em navegadores e precisa defender conteúdos salvos ou copiados para fora do ambiente da aplicação.

O Argon2id trabalha com parâmetros ajustáveis. Entre os mais relevantes estão o custo de memória, que define aproximadamente quanta memória será usada na operação; o custo de tempo, que determina quantas passagens internas serão realizadas; e o grau de paralelismo, que regula como o trabalho pode ser repartido internamente. O Cifrei não trata esses parâmetros como números abstratos para fins

acadêmicos. Ele os utiliza como controles práticos de segurança e de usabilidade. Em vez de impor um parâmetro fixo universal para todos os dispositivos, o aplicativo mede localmente qual custo é razoável para o *hardware* em uso e conserva esse resultado para operações futuras.

O efeito dessa decisão é duplo. De um lado, o sistema evita desperdiçar segurança em dispositivos mais capazes, já que não fica limitado a um patamar baixo pensado para o pior aparelho possível. De outro, o sistema não penaliza desnecessariamente os dispositivos mais modestos, pois não os obriga a executar o mesmo custo de derivação definido por um computador muito mais potente. Essa flexibilidade não compromete a interoperabilidade entre dispositivos, porque os parâmetros efetivamente usados em cada cifra acompanham o próprio código final.

5. Calibração local por tempo e teto global

A calibração dinâmica implementada no Cifrei foi concebida como uma solução prática para o problema de heterogeneidade dos dispositivos. Em vez de assumir que todos os aparelhos respondem da mesma forma, o aplicativo realiza um teste local para estimar quais parâmetros do Argon2id conduzem a um tempo de execução adequado. O alvo do projeto situa-se, em regra, entre aproximadamente 400 e 700 milissegundos por operação, admitindo-se tempos maiores, até um limite de tolerância, em dispositivos mais modestos ou em cenários excepcionais.

Esse intervalo não é meramente estético. Ele representa uma escolha de produto. Tempos muito baixos poderiam significar que o processo de derivação ficou barato demais para um atacante. Tempos excessivos, por sua vez, aumentariam fricção, consumo de bateria e sensação de lentidão sem garantir, na prática, um salto proporcional de proteção. A calibração busca exatamente o ponto em que o custo imposto ao usuário ainda é aceitável e, ao mesmo tempo, o custo imposto ao atacante já é relevante.

Uma vez encontrada uma configuração adequada, o resultado é armazenado localmente no dispositivo. Isso significa que o usuário não precisa recalibrar a cada uso. A calibração volta a ser necessária apenas se a informação local desaparecer ou se, por decisão de produto futura, a política de custo for revisada. A presença de um teto global completa o modelo. Mesmo quando um dispositivo muito potente poderia suportar parâmetros muito mais altos, o aplicativo evita que a cifra criada nesse aparelho se torne excessivamente desconfortável de abrir em dispositivos mais modestos. Em outras palavras, a calibração maximiza a segurança local dentro de limites compatíveis com a interoperabilidade real do produto.

6. AES-GCM: confidencialidade, integridade e autenticação

Depois de produzida a chave criptográfica por meio do Argon2id, a mensagem é cifrada usando o algoritmo AES no modo GCM. Essa escolha é relevante porque o AES-GCM não oferece apenas um embaralhamento do conteúdo; ele fornece o que se chama de criptografia autenticada. A confidencialidade impede que terceiros leiam a mensagem sem a chave correta. A integridade garante que mudanças indevidas na saída cifrada sejam detectadas. A autenticação, nesse contexto, significa que a operação de decifragem só terá êxito se todos os elementos necessários estiverem corretos e consistentes. Se, por exemplo, o usuário informar a chave, o código ou a frase segredo com um único e mínimo erro de digitação, a decifragem nem roda.

O modo GCM produz, junto com o texto cifrado, uma autenticação criptográfica incorporada ao resultado. Por isso, quando o aplicativo informa erro de verificação ou falha de autenticação, o significado técnico é que a operação não conseguiu reproduzir, com a mesma chave e os mesmos parâmetros, o estado esperado da cifra. Em termos de uso, isso pode ocorrer por quatro razões principais: a chave ou a frase segredo foram informadas de modo incorreto; o código foi alterado ou corrompido; o *salt* ou o IV foram lidos de modo errado; ou algum parâmetro do processo não corresponde ao originalmente usado. A vantagem do AES-GCM é que ele detecta essas situações sem precisar expor detalhes sensíveis sobre qual parte exata falhou.

O IV gerado em cada operação tem papel central nesse processo. Ele não é um segredo, mas precisa ser único para cada cifragem sob uma mesma chave. Ao acompanhar o código final, ele permite que a decifragem reconstrua a operação corretamente. Como o IV é renovado a cada cifragem, duas mensagens iguais não tendem a produzir saídas idênticas quando submetidas à mesma chave derivada. Esse detalhe reduz a exposição de padrões repetitivos que poderiam ser explorados por observadores externos.

7. Passo a passo operacional da cifragem

O fluxo de cifragem pode ser descrito em etapas sucessivas.

1. O usuário informa a mensagem, a chave e a frase segredo.
2. O aplicativo consulta localmente se já possui parâmetros de calibração confiáveis para aquele dispositivo. Se ainda não possuir, realiza uma calibração inicial para estimar um custo adequado do Argon2id e registra o resultado localmente.
3. O sistema gera um *salt* aleatório para aquela operação específica.
4. A chave e a frase segredo, juntamente com esse *salt* e com os parâmetros definidos, são convertidas em um material criptográfico adequado ao Argon2id.
5. O Argon2id executa a derivação de chave, produzindo uma chave criptográfica robusta.
6. O sistema gera um IV para a operação de AES-GCM.
7. A mensagem em texto aberto é cifrada com o AES-GCM usando a chave derivada e o IV recém-gerado.
8. Este passo consiste em reunir, em um mesmo formato, a versão do protocolo, os parâmetros mínimos usados pelo KDF, o *salt*, o IV e o resultado da cifragem autenticada.
9. O último passo é a serialização desse conjunto em um código compacto, que pode ser copiado, salvo ou transmitido pelo usuário. O ponto essencial é que esse código carrega consigo tudo o que será necessário para reproduzir a operação mais tarde, exceto, naturalmente, a chave e a frase segredo, que continuam pertencendo ao usuário.

8. Passo a passo operacional da decifragem

A decifragem começa quando o usuário apresenta ao aplicativo três elementos: o código final, a chave e a frase segredo.

Em primeiro lugar, o sistema lê a estrutura do código e extrai dela a versão do formato, os parâmetros mínimos do KDF, o *salt*, o IV e o texto cifrado autenticado.

Em segundo lugar, a aplicação reconstrói o material de derivação a partir da chave, da frase segredo, do *salt* e dos parâmetros recuperados.

Em terceiro lugar, o Argon2id é executado novamente, agora não com base na calibração local preferida do dispositivo atual, mas com base nos parâmetros que vieram dentro do próprio código. Essa distinção é decisiva. O aparelho pode ter sido calibrado para um custo diferente, mas a cifra só será aberta corretamente se a derivação for refeita exatamente com os parâmetros da operação original. Em quarto lugar, a chave derivada é usada para tentar a decifragem pelo AES-GCM com o IV extraído do código.

Se todos os elementos estiverem corretos e intactos, a mensagem em texto aberto é recuperada com êxito. Se houver qualquer divergência relevante, como uma frase segredo incorreta, uma chave errada, um trecho corrompido do código ou parâmetros incompatíveis, o AES-GCM rejeitará a operação e a aplicação informará falha de verificação ou erro de autenticação. Esse comportamento é desejável, porque evita a apresentação de saídas parcialmente corrompidas e ajuda a manter a integridade lógica do sistema.

9. O que exatamente vai dentro do código final

Do ponto de vista do usuário, o código final é uma sequência compacta de caracteres. Do ponto de vista técnico, ele é a serialização de um pequeno conjunto de campos indispensáveis para a operação posterior de decifragem. Em linhas gerais, o código incorpora a identificação da versão de formato, os parâmetros necessários para refazer a derivação de chave, o *salt*, o IV e o resultado do AES-GCM. Em uma implementação compacta, esses dados são empacotados com o mínimo possível de redundância textual.

Esse desenho resolve duas necessidades ao mesmo tempo. Em primeiro lugar, torna a cifra autossuficiente, porque tudo que é necessário para a reconstrução da operação acompanha a própria saída. Em segundo lugar, evita a inflação desnecessária do tamanho do código, já que a arquitetura não precisa carregar descrições verbosas em formato textual quando a própria versão de protocolo já determina a maior parte das convenções internas. Em outras palavras, o código não é apenas uma mensagem cifrada; ele é uma pequena cápsula de reconstrução do processo criptográfico.

10. Compatibilidade entre dispositivos e independência da calibração local

Um dos pontos mais delicados em arquiteturas com custo dinâmico é a compatibilidade entre dispositivos. Se uma cifra fosse criada em um aparelho potente com parâmetros agressivos e depois aberta em um aparelho mais fraco usando apenas a calibração local do segundo dispositivo, a derivação de chave não reproduziria o mesmo resultado. O Cifrei evita esse erro ao colocar no próprio código os parâmetros efetivamente usados na operação original.

Por isso, a calibração local do dispositivo atual serve principalmente para criar novas cifras, não para reinterpretar cifras antigas de modo arbitrário. Ao decifrar, o aplicativo obedece ao que veio dentro do código. Esse princípio garante a interoperabilidade do sistema. Uma cifra produzida em um computador pode ser aberta em um telefone, e uma cifra produzida em um telefone pode ser aberta em um computador, desde que o usuário apresente a mesma chave e a mesma frase segredo e desde que o código não tenha sido alterado.

11. Modelo de ameaça e limites da arquitetura

O principal cenário de defesa do Cifrei é o ataque offline contra o código final. Imagine-se um atacante que obtenha acesso ao código de uma cifra e tente descobrir, por tentativa e erro, a combinação correta de chave e frase segredo. Nesse cenário, o papel do Argon2id é encarecer cada hipótese de ataque, exigindo tempo e memória para cada tentativa. O papel do AES-GCM é garantir que nenhuma tentativa de decifragem apresente um falso positivo silencioso sem a autenticação correta do conteúdo.

Há, contudo, limites naturais que nenhum white paper sério deveria omitir. Se o usuário escolher uma frase segredo muito fraca, previsível ou reutilizada em excesso, a segurança final do sistema diminui. Se o dispositivo estiver comprometido por software malicioso, a proteção do algoritmo não impede, por si só, que dados digitados sejam capturados antes mesmo da cifragem. Do mesmo modo, o sistema não substitui a necessidade de boas práticas operacionais, como cuidado com engenharia social, proteção de credenciais e integridade do ambiente de execução.

A arquitetura também não deve ser interpretada como uma garantia metafísica contra todo tipo de análise futura. O que ela oferece é uma combinação muito robusta de técnicas contemporâneas, adequada ao estado atual da prática em aplicações client-side. Seu mérito está em impor custo relevante ao ataque realista mais plausível para esse modelo de produto, sem exigir do usuário comum uma experiência hostil ou excessivamente técnica.

12. Comparação conceitual com modelos anteriores e alternativas comuns

O *salt*o qualitativo do Cifrei pode ser compreendido pela comparação conceitual com arquiteturas baseadas em funções mais antigas de derivação de chave. Em modelos centrados em PBKDF2, por exemplo, a resistência do sistema é aumentada principalmente pelo número de iterações, isto é, pelo custo em tempo de processamento. O Argon2id acrescenta uma dimensão muito importante: o custo em memória. Isso dificulta ataques paralelos em hardware especializado de maneira mais eficiente do que o simples aumento linear de iterações.

Em comparação com funções amplamente conhecidas no contexto de armazenamento de senhas, como o bcrypt, o Argon2id oferece uma flexibilidade maior de parametrização e um desenho mais moderno para os objetivos de derivação resistente a ataques offline em ambientes contemporâneos. No contexto do Cifrei, o ganho prático não está em um slogan tecnológico, mas no fato de que o custo imposto ao atacante se torna mais caro de escalar, sobretudo quando combinado com uma política de calibração realista por dispositivo.

13. Segurança, praticidade e racional de produto

A arquitetura do Cifrei foi construída sobre uma ideia simples, mas exigente: qualquer sacrifício de praticidade só se justifica se produzir um ganho de segurança efetivo; e qualquer concessão em segurança só é aceitável quando preserva usabilidade sem desperdiçar proteção real. Esse raciocínio explica a escolha por uma faixa-alvo de tempo para a derivação de chave, a existência de um teto global e a decisão de armazenar localmente a calibração. O produto não busca a configuração mais dura em abstrato; ele busca a melhor configuração concreta para o uso real.

Esse ponto é importante porque sistemas criptográficos, quando mal calibrados, podem acabar punindo o usuário sem elevar de forma relevante o custo do ataque. O Cifrei procura evitar esse erro. Seu desenho parte da experiência efetiva de cifrar e decifrar em dispositivos modernos, incluindo telefones de bom nível e computadores pessoais convencionais, e transforma essa experiência em parâmetro de engenharia. O resultado é um sistema que tenta ser sério sem ser áspero, e técnico sem se tornar inacessível.

14. Conclusão

A arquitetura criptográfica do Cifrei pode ser resumida como a união de uma derivação de chave moderna, baseada em Argon2id, com uma cifra autenticada robusta, baseada em AES-GCM, articuladas por uma camada de engenharia orientada à interoperabilidade e à experiência do usuário. A chave e a frase segredo fornecidas pelo usuário não são tratadas como chave criptográfica pronta; elas são transformadas, junto com um *salt* e parâmetros específicos da operação, em uma chave criptográfica forte. Essa chave é então empregada para cifrar a mensagem e proteger sua integridade. O resultado é encapsulado em um código compacto, suficientemente rico para permitir a decifragem posterior em qualquer dispositivo compatível.

Do ponto de vista técnico, o modelo é sólido porque reconhece a diferença entre derivar uma chave e cifrar uma mensagem, entre calibrar um dispositivo e reconstruir uma operação antiga, entre tornar o sistema lento e torná-lo realmente caro para um atacante. Do ponto de vista do produto, ele é coerente porque transforma essas distinções em uma experiência de uso simples. O Cifrei não se limita, portanto, a trocar algoritmos; ele reorganiza o fluxo criptográfico do aplicativo em torno de uma arquitetura mais madura, mais explicável e mais consistente com o estado atual das boas práticas em segurança aplicada.

Apêndice A. Resposta curta à pergunta central: "o que acontece com meus dados quando eu cifro?"

Quando o usuário clica para cifrar, a aplicação não usa diretamente o texto da chave ou da frase segredo como chave criptográfica. Primeiro, ela gera um *salt* da operação e consulta os parâmetros de custo apropriados. Depois, transforma chave e frase segredo em uma chave criptográfica forte por meio do Argon2id. Em seguida, gera um IV e usa essa chave criptográfica para cifrar a mensagem pelo AES-GCM. Por fim, reúne em um único código o que será necessário para refazer a operação no futuro: versão, parâmetros mínimos, *salt*, IV e texto cifrado autenticado.

Quando o usuário clica para decifrar, o aplicativo faz o caminho inverso. Ele lê o código, extrai os parâmetros, refaz exatamente a mesma derivação de chave com a chave e a frase segredo apresentadas e tenta abrir a cifra com o AES-GCM. Se tudo coincidir, a mensagem reaparece. Se qualquer elemento estiver errado ou adulterado, a operação falha de maneira segura.