Avaliação Experimental do Impacto de Cadeias de Assinaturas Digitais em Sistemas Distribuídos

Bruno F. Lovato[1], **Gabriela Stein**[1,2], **Luiz A. Rodrigues**[1] **e Elias P. Duarte Jr.**[2]

[1]UNIOESTE – Cascavel/PR – Brazil

{bruno.lovato,gabriela.stein1,luiz.rodrigues}@unioeste.br

[2]UFPR – Curitiba/PR – Brazil

elias@inf.ufpr.br

Abstract. *Distributed systems use cryptographic mechanisms to ensure the integrity, authenticity, and confidentiality of messages. Digital signature chains record the message propagation path but cause linear growth in size and computational cost. This work presents a formal model and an experimental evaluation in C++, Go, and Python, comparing the Ed25519 and RSA-2048 algorithms, with and without confidentiality (AES-GCM and ChaCha20-Poly1305). The results show that cumulative verification cost is the main scalability limitation, while Ed25519 offers better performance and lower structural growth. Confidentiality has less impact on long chains but is significant for small messages. The results emphasize the need to balance security and performance, with direct implications for fault-tolerant distributed protocols.*

Resumo. *Sistemas distribuídos utilizam mecanismos criptográficos para garantir integridade, autenticidade e confidencialidade das mensagens. Cadeias de assinaturas digitais permitem registrar o caminho de propagação das mensagens, mas introduzem crescimento linear no tamanho e no custo computacional. Este trabalho apresenta um modelo formal e uma avaliação experimental em C++, Go e Python, comparando os algoritmos Ed25519 e RSA-2048, com e sem confidencialidade (AES-GCM e ChaCha20-Poly1305). Os resultados mostram que o custo de verificação cumulativa é o principal fator limitante da escalabilidade, enquanto o Ed25519 apresenta melhor desempenho e menor crescimento estrutural. A confidencialidade tem impacto reduzido em cadeias longas, mas é relevante em mensagens pequenas. Os resultados evidenciam a necessidade de equilíbrio entre segurança e desempenho, com implicações diretas para protocolos distribuídos tolerantes a falhas.*

1. Introdução

A comunicação segura é um requisito fundamental em sistemas distribuídos modernos [Ahmed et al. 2016, Tandi 2019, Chen et al. 2023, Ziwich et al. 2005]. Nestes sistemas, mecanismos criptográficos são amplamente utilizados para garantir integridade, autenticidade e não repúdio das mensagens entre participantes. Além da verificação de mensagens individuais, diversas aplicações exigem mecanismos que comprovem a ordem temporal dos eventos, a proveniência dos dados e a participação sucessiva de múltiplas entidades ao longo de um processo distribuído [Saxena and Soh 2008].

Nesse contexto, cadeias de assinatura (*signature chains*) surgem como uma solução criptográfica natural para registrar sequências verificáveis de propagação de mensagens. Neste modelo, múltiplas assinaturas digitais são encadeadas sequencialmente, de forma que cada novo elo incorpora uma referência criptográfica ao estado previamente assinado. Essa estrutura torna computacionalmente inviável remover ou modificar elos intermediários sem invalidar toda a cadeia, preservando a integridade e a rastreabilidade dos eventos [van der Linde 2018]. Diferentemente das *hash chains* clássicas, cadeias baseadas em assinaturas utilizam criptografia de chave pública, permitindo verificação independente por qualquer participante e evitando limitações de segurança associadas a esquemas puramente baseados em *hash*, como a possibilidade de derivação de valores futuros em cenários de comprometimento de um nó [Bicakci and Baykal 2003].

Apesar de seu potencial para suportar auditoria, rastreabilidade e responsabilização em sistemas distribuídos, a literatura existente concentra-se predominantemente em aspectos criptográficos ou arquiteturais dessas cadeias. Como consequência, ainda são limitados os estudos empíricos que investigam o impacto cumulativo do encadeamento no desempenho de sistemas reais, especialmente considerando o crescimento estrutural das mensagens, o custo sequencial de verificação e a interação com mecanismos adicionais de segurança, como criptografia para confidencialidade. Essa lacuna é particularmente relevante no contexto de sistemas tolerantes a falhas, nos quais mecanismos de disseminação confiável, consenso e replicação frequentemente dependem da validação contínua de mensagens. Nestes cenários, o custo adicional imposto por cadeias de assinatura pode afetar diretamente a latência, a vazão e a escalabilidade dos protocolos, tornando essencial compreender quantitativamente esses impactos.

Este trabalho propõe um modelo formal unificado que captura o crescimento estrutural de cadeias de assinatura e sua extensão com mecanismos de confidencialidade. Além disso, apresentamos uma avaliação experimental multilinguagem, comparando C++, Go e Python, bem como diferentes algoritmos criptográficos (Ed25519 e RSA-2048) e mecanismos de confidencialidade (AES-GCM e ChaCha20-Poly1305). Diferentemente de trabalhos anteriores, esta avaliação foca explicitamente na decomposição dos custos de assinatura, verificação e criptografia, bem como no impacto cumulativo da verificação sequencial ao longo da cadeia. Adicionalmente, analisamos o crescimento do tamanho das mensagens e seus efeitos práticos na rede, incluindo a ocorrência de fragmentação em função de limites de MTU (*Maximum Transmission Unit*).

Os resultados mostram que: (i) o custo de verificação cumulativa é o principal fator limitante da escalabilidade; (ii) o Ed25519 apresenta melhor desempenho e menor crescimento estrutural em comparação ao RSA-2048; e (iii) o impacto da confidencialidade é marginal em cenários com cadeias longas, mas relevante para mensagens pequenas. Estes resultados fornecem evidências quantitativas sobre limites práticos de escalabilidade e destacam aspectos fundamentais entre segurança criptográfica e desempenho em sistemas distribuídos tolerantes a falhas.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta a fundamentação teórica, apresentando os conceitos básicos de assinaturas digitais, criptografia e confidencialidade. A Seção 3 formaliza o modelo de avaliação proposto para o problema de cadeias de assinatura. A Seção 4 descreve a metodologia experimental. A Seção 5 apresenta e discute os resultados em diferentes distribuições de impacto. Os

trabalhos relacionados são apresentados na Seção 6. Por fim, a Seção 7 apresenta a conclusão e os trabalhos futuros.

2. Fundamentação Teórica

A segurança das cadeias de assinatura analisadas neste trabalho baseia-se na combinação de primitivas de criptografia assimétrica (para autenticação e não-repúdio) e criptografia simétrica (para confidencialidade). Esta seção apresenta os fundamentos conceituais e as propriedades de segurança relevantes para o modelo experimental adotado.

2.1. Assinaturas Digitais

Assinaturas digitais constituem o mecanismo criptográfico central para garantir autenticidade, integridade e não-repúdio. Diferentemente de mecanismos baseados puramente em *hash*, assinaturas digitais utilizam criptografia de chave pública, permitindo verificação independente por qualquer entidade que possua a chave pública correspondente.

Para tanto, a criptografia assimétrica emprega um par de chaves (sk, pk), onde sk é a chave privada e pk a chave pública. Uma assinatura digital consiste em um par de algoritmos `Sign` e `Verify`, de modo que:

$$\sigma = \mathsf{Sign}_{sk}(m) \qquad (1) \qquad\qquad \mathsf{Verify}_{pk}(m, \sigma) \in \{0, 1\} \qquad (2)$$

Neste trabalho foram avaliados dois algoritmos amplamente utilizados:

- RSA-2048: padrão com 2048 bits de comprimento e 256 bytes por assinatura [Moriarty et al. 2016], comumente utilizado para proteger dados, comunicações, assinaturas digitais e conexões HTTPS/VPNs.
- Ed25519: baseado em curvas elípticas Curve25519 e no algoritmo EdDSA (*Edwards-curve Digital Signature Algorithm*) [Josefsson and Liusvaara 2017]. Oferece chaves de 64 bytes e assinaturas rápidas com 64 bytes de tamanho. É normalmente utilizado no SSH moderno, criptomoedas e segurança de dados.

2.2. Criptografia Simétrica

A criptografia simétrica utiliza uma única chave secreta k compartilhada entre as partes. Um esquema de criptografia autenticada pode ser definido como:

$$\mathsf{Enc}(k, m) \to c \qquad (3) \qquad\qquad \mathsf{Dec}(k, c) \to m \qquad (4)$$

Diferentemente da criptografia de chave pública, o custo da criptografia simétrica é geralmente proporcional ao tamanho do *payload*, mas independente do número de participantes. Assim, seu impacto tende a ser mais significativo em mensagens grandes, enquanto o custo de assinaturas domina em cadeias com muitos processos.

Vale ressaltar que, embora a criptografia simétrica forneça confidencialidade eficiente, ela requer distribuição segura de chaves. Em sistemas distribuídos, isso geralmente implica o uso de um protocolo de estabelecimento de chaves (como TLS), que adiciona custo adicional de *handshake*. Neste trabalho, o foco recai sobre o impacto da criptografia de dados em si, isolando o custo estrutural do encadeamento de assinaturas.

2.3. Confidencialidade

Confidencialidade refere-se à garantia de que o conteúdo de uma mensagem permanece inacessível a entidades não autorizadas. No contexto de cadeias de assinatura, a confidencialidade não é inerente ao mecanismo de encadeamento, uma vez que assinaturas digitais fornecem integridade, autenticidade e não-repúdio, mas não ocultam o conteúdo da mensagem.

Neste estudo, foram testadas duas opções de criptografia simétrica:

- AES-GCM: cifra de blocos de 128 bits no modo GCM (Galois/Counter Mode) [Salowey et al. 2008].
- ChaCha20-Poly1305: cifra de fluxo com código de autenticação de mensagem Poly1305 [Nir and Langley 2018].

O GCM e o Poly1305 implementam um esquema AEAD (*Authenticated Encryption with Associated Data*), que fornece simultaneamente confidencialidade, integridade e autenticidade baseada na chave simétrica.

Neste trabalho, a confidencialidade é implementada como um mecanismo complementar, aplicado após a construção da cadeia de assinaturas. Seja m o *payload* original e $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ o conjunto de assinaturas que compõem a cadeia da origem até o processo k. A mensagem final antes da transmissão é $M = m \parallel \Sigma$. A aplicação de criptografia simétrica é aplicada sobre M conforme discutido na seção anterior.

A introdução de confidencialidade altera o custo do sistema de forma distinta das assinaturas digitais. Enquanto o custo das assinaturas cresce aproximadamente com o número de elos N, o custo da criptografia simétrica cresce com o tamanho do *payload* S. Assim, a latência total pode ser modelada como:

$$T_{total}(N, S) = T_{sign}(N) + T_{verify}(N) + T_{enc}(S) + T_{dec}(S) \tag{5}$$

Desta forma, é possível identificar regimes distintos de dominância de custo: para grandes cadeias (N elevado) e *payload* pequeno, o custo assimétrico tende a dominar; para *payloads* extensos e cadeias curtas, o custo simétrico pode se tornar relevante.

3. Cadeias de Assinaturas

Cadeias de assinaturas digitais encadeiam assinaturas como "links" sequenciais para provar origem, integridade e encadeamento temporal de dados. São diversas suas aplicações, incluindo autenticação OTP (*One-Time Password* ou Senha de Uso Único), blockchains e roteamento seguro, entre outras.

Considere um sistema distribuído composto por N processos $\{p_1, p_2, \ldots, p_n\}$. Cada processo p_i possui um par de chaves criptográficas (pk_i, sk_i), necessário para implementar a assinatura digital da mensagem.

A seguir, são apresentados dois modelos: o primeiro utiliza assinatura digital, garantindo integridade e autenticidade apenas por meio de criptografia assimétrica; o segundo inclui a confidencialidade, exigindo o uso adicional de criptografia simétrica.

3.1. Modelo sem Confidencialidade

Sejam:

- P o *payload* da aplicação, com tamanho $|P|$ bytes;
- $H(\cdot)$ uma função *hash* criptográfica com saída de tamanho h bytes;
- $Sign_{sk_i}(\cdot)$ o algoritmo de assinatura digital;
- $Verify_{pk_i}(\cdot)$ o algoritmo de verificação da assinatura digital;
- s o tamanho da assinatura digital em bytes;
- m_d o tamanho dos metadados adicionais.

A mensagem inicial ($k = 1$) é definida como:

$$M_1 = \langle P_1, H(\perp), \sigma_1, m_d \rangle \tag{6}$$

onde:

$$\sigma_1 = Sign_{sk_1}(P_1 \parallel H(\perp)) \tag{7}$$

e $\perp$ representa a ausência de mensagem anterior.

Para $k > 1$, a mensagem encadeada é definida recursivamente como:

$$M_k = \langle P_k, H(M_{k-1}), \sigma_k, m_d \rangle \tag{8}$$

onde:

$$\sigma_k = Sign_{sk_k}(P_k \parallel H(M_{k-1})) \tag{9}$$

Em cenários onde todas as assinaturas anteriores são preservadas (cadeia acumulativa), a estrutura completa da mensagem torna-se:

$$M_k = \langle P_k, H(M_{k-1}), \sigma_1, \sigma_2, \ldots, \sigma_k, m_d \rangle \tag{10}$$

Se $|P| = P$ representa o tamanho do *payload*, o tamanho total da mensagem com profundidade k é:

$$|M_k| = P + k(s + h) + m_d \tag{11}$$

O crescimento do tamanho da mensagem é linear em função de k.

O *overhead* criptográfico relativo ao *payload* é dado por:

$$O(k) = \frac{k(s + h) + m_d}{P} \tag{12}$$

Em relação ao custo computacional, sejam:

- T_{hash} o tempo de cálculo do *hash*;

- T_{sign} o tempo de assinatura;
- T_{verify} o tempo de verificação.

O custo total para processar uma mensagem com profundidade k é:

$$T_{total}(k) = T_{hash} + T_{sign} + k \cdot T_{verify} \tag{13}$$

Portanto, tanto o tamanho quanto o custo computacional crescem linearmente com a profundidade da cadeia.

3.2. Modelo com Confidencialidade

A inclusão de confidencialidade ponta-a-ponta estende o modelo anterior, combinando assinatura digital e criptografia simétrica.

Sejam:

- $Enc_K(\cdot)$ e $Dec_K(\cdot)$ os algoritmos de criptografia simétrica (ex.: AES-GCM);
- K a chave simétrica compartilhada;
- iv o vetor de inicialização, com tamanho $|iv|$;
- tag a tag de autenticação, com tamanho $|tag|$;
- $e = |iv| + |tag|$ o *overhead* fixo da confidencialidade;
- T_{enc} o tempo de criptografia;
- T_{dec} o tempo de descriptografia.

Como no modelo sem confidencialidade, primeiro constrói-se a mensagem autenticada M_k com todas as assinaturas encadeadas. Em seguida, aplica-se criptografia:

$$C_k = \langle iv, Enc_K(M_k), tag \rangle \tag{14}$$

A verificação ocorre em duas etapas:

1. $Dec_K(C_k) \rightarrow M_k$
2. $Verify_{pk_i}(M_k), \sigma_i)$ para $i = 1..k$

O tamanho total da mensagem transmitida é:

$$|C_k| = |M_k| + e \tag{15}$$

Substituindo o modelo anterior:

$$|C_k| = P + k(s + h) + m_d + e \tag{16}$$

O crescimento continua linear em k, com deslocamento constante e.

O custo total para processar uma mensagem confidencial com profundidade k é:

$$T^{conf}_{total}(k) = T_{hash} + T_{sign} + T_{enc} + T_{dec} + k \cdot T_{verify} \tag{17}$$

Caso cada nó verifique todas as assinaturas anteriores, o custo cresce linearmente em k.

O overhead total relativo ao *payload* é:

$$O_{conf}(k) = \frac{k(s+h) + m_d + e}{P} \tag{18}$$

4. Metodologia Experimental

As implementações foram desenvolvidas em C++, Go e Python[1], utilizando bibliotecas criptográficas consolidadas em cada ecossistema, para garantir confiabilidade, reprodutibilidade e comparabilidade. Em Python, foi empregada a biblioteca `cryptography`, baseada em OpenSSL; em C++, utilizou-se OpenSSL 3.x por meio da API EVP; e em Go, foram utilizados exclusivamente pacotes padrão do `crypto/*`.

Foram adotados os mesmos esquemas criptográficos em todas as linguagens: Ed25519 e RSA-2048 para assinaturas digitais, e AES-256-GCM ou ChaCha20-Poly1305 para confidencialidade. A uniformização dos parâmetros criptográficos assegura que diferenças de desempenho observadas decorram do ambiente de execução e das características das linguagens, e não das primitivas utilizadas.

4.1. Cenários de Teste

Os experimentos foram estruturados a partir da combinação de três dimensões principais: confidencialidade, tamanho do *payload* e número de processos na cadeia.

A confidencialidade foi avaliada em dois modos: (i) sem criptografia adicional, utilizando apenas assinaturas digitais, e (ii) com criptografia simétrica via AES-256-GCM ou ChaCha20-Poly1305 e aplicada ao *payload* antes da assinatura.

O *payload* foi variado entre 8B e 1024B, contemplando desde mensagens de controle até cargas mais representativas. Já o número de processos foi escalonado de 1 a 8, permitindo analisar o impacto do crescimento da cadeia em cenários de diferentes escalas.

A combinação desses parâmetros define um espaço experimental tridimensional, possibilitando avaliar o comportamento do sistema sob diferentes condições operacionais. Embora o número de processos seja limitado, o comportamento observado é consistente com o modelo analítico, permitindo extrapolação para cenários maiores, especialmente em topologias estruturadas.

4.2. Métricas Avaliadas

A latência total média corresponde ao tempo desde a geração do *payload* até a verificação final na cadeia. Para análise detalhada, foram medidas separadamente as latências de assinatura e verificação. Nos cenários com confidencialidade, também foram mensuradas as latências de criptografia e descriptografia.

O tamanho efetivo transmitido inclui o *payload*, as assinaturas acumuladas e, quando aplicável, os campos adicionais da criptografia autenticada (como *nonce* e *tag*). Essa métrica permite avaliar o crescimento estrutural das mensagens. Adicionalmente,

[1]Código-fonte e resultados no GitHub https://github.com/BrunoLovato41

foi analisada a fragmentação potencial, considerando uma MTU de 1.500 bytes, a fim de identificar cenários em que o tamanho da mensagem excede esse limite.

Cada experimento foi executado 30 vezes para o cálculo das resultados. Os dados completos com desvio padrão e intervalo de confiança estão no repositório.

4.3. Ambiente de Execução de Experimentos

Os experimentos foram conduzidos em um notebook Lenovo IdeaPad L340, com processador Intel Core i5-9300H (4 núcleos, 8 threads, até 4,10 GHz), 16 GB de memória RAM DDR4 e sistema operacional Linux (kernel 6.17). O processador possui suporte a instruções criptográficas por hardware, como AES-NI e AVX2, relevantes para otimização de operações simétricas.

5. Resultados Experimentais

Esta seção apresenta a análise dos resultados obtidos a partir da combinação dos parâmetros experimentais descritos anteriormente. A avaliação considera o impacto do número de processos, do tamanho do *payload*, da linguagem de implementação, do algoritmo de assinatura e da presença de confidencialidade.

5.1. Latência

A Tabela 1 apresenta os tempos médios de verificação e assinatura considerando um *payload* de 8 bytes. Para os demais tamanhos de mensagem, o comportamento é similar. Os resultados completos estão disponíveis em https://github.com/BrunoLovato41.

Observa-se, inicialmente, que tanto a latência de verificação quanto a de assinatura crescem monotonicamente com o número de processos, refletindo o comportamento cumulativo das cadeias de assinatura. Este crescimento é aproximadamente linear, o que confirma empiricamente que o custo total do sistema é proporcional ao tamanho da cadeia, isto é, ao número de participantes.

A comparação entre algoritmos revela diferenças significativas. Em todos os cenários, o custo de assinatura utilizando RSA é substancialmente superior ao observado com Ed25519. Para oito processos, por exemplo, a assinatura em RSA atinge aproximadamente 4,82 ms em C++, 10,27 ms em Go e 8,48 ms em Python, enquanto Ed25519 permanece abaixo de 0,50 ms em todas as linguagens. Isso representa ganhos que variam de uma a duas ordens de magnitude, dependendo da linguagem e da configuração. Esse comportamento evidencia que o algoritmo criptográfico é o principal fator determinante do desempenho das cadeias de assinatura.

Em relação à verificação, embora os valores sejam inferiores aos de assinatura, também se observa crescimento com o número de processos. No caso do Ed25519, a verificação permanece abaixo de 1 ms, enquanto no RSA, os valores são maiores e apresentam maior variação entre linguagens. Ainda assim, o custo de verificação tende a ser menos dominante que o de assinatura no tempo total.

A análise por linguagem revela diferenças relevantes de desempenho. A implementação em C++ apresenta comportamento mais equilibrado, com bom desempenho tanto para RSA quanto para Ed25519. A linguagem Go destaca-se pelo excelente desempenho com Ed25519, apresentando os menores tempos de assinatura e verificação

Tabela 1. Tempos médios (ms) de verificação e assinatura, sem confidencialidade para *payload* de 8 bytes com menores valores em negrito.

N	C++				Go				Python			
	RSA		Ed25519		RSA		Ed25519		RSA		Ed25519	
	Verify	Sign	Verify	Sign	Verify	Sign	Verify	Sign	Verify	Sign	Verify	Sign
1	0,00	0,65	0,00	0,06	0,00	1,46	0,00	**0,03**	0,00	0,83	0,00	0,05
2	0,06	1,83	0,14	0,12	**0,04**	2,77	**0,07**	**0,05**	0,16	**1,47**	0,13	0,11
3	0,08	**1,82**	0,29	0,18	**0,09**	4,15	**0,12**	**0,08**	0,23	2,14	0,27	0,17
4	**0,12**	**2,50**	0,43	0,24	0,15	5,67	**0,19**	**0,11**	0,48	2,82	0,41	0,23
5	**0,16**	**3,19**	0,55	0,27	0,20	7,16	**0,24**	**0,13**	0,48	3,49	0,53	0,28
6	**0,21**	**3,81**	0,70	0,36	0,24	8,48	**0,30**	**0,15**	0,41	3,98	0,67	0,35
7	**0,21**	**4,32**	0,84	0,41	0,28	9,54	**0,36**	**0,19**	0,67	4,78	0,81	0,41
8	**0,22**	**4,82**	0,98	0,47	0,33	10,27	**0,42**	**0,22**	1,87	8,48	0,95	0,49

nesse algoritmo. No entanto, no caso do RSA, Go apresenta os maiores custos de assinatura, indicando possível maior *overhead* na implementação dessa primitiva. Já Python apresenta maior latência em praticamente todos os cenários, especialmente em alta cardinalidade, refletindo o impacto do ambiente interpretado e da gestão de memória.

Outro aspecto importante observado é que a assinatura domina o custo total das cadeias, especialmente no caso do RSA. Isso implica que, em sistemas distribuídos que utilizam cadeias de assinatura, o gargalo principal não está apenas na verificação das mensagens recebidas, mas também na geração de novas assinaturas a cada etapa.

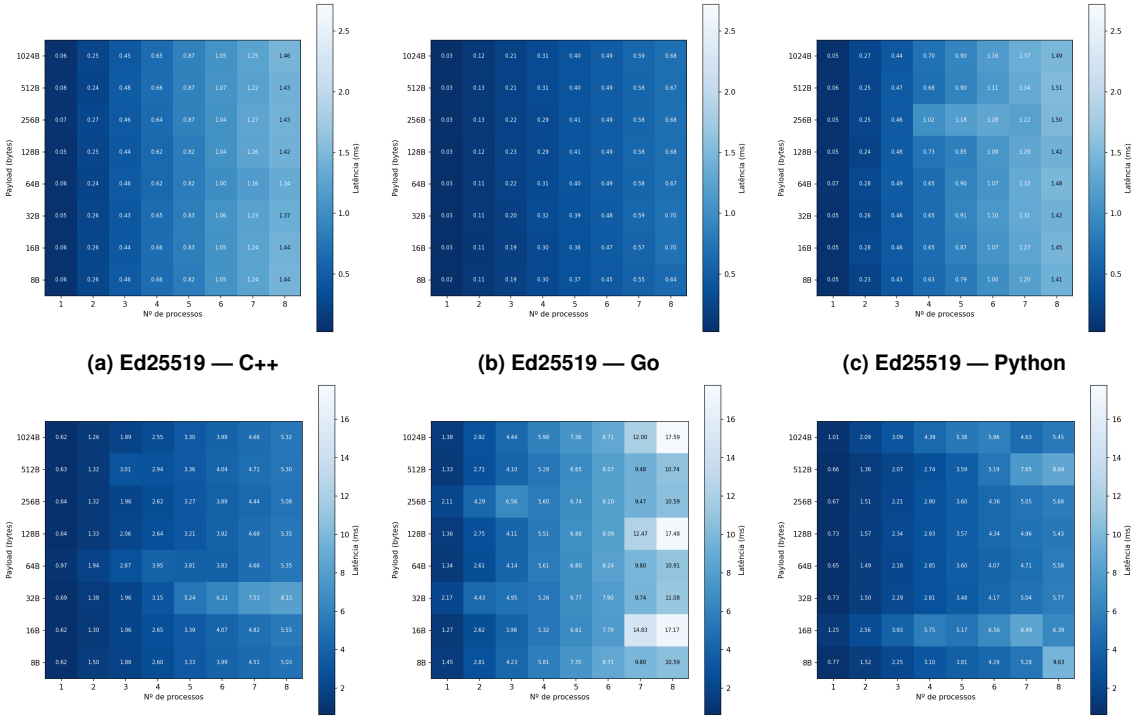
Do ponto de vista da escalabilidade, os resultados indicam que o uso de Ed25519 torna-se significativamente mais adequado para sistemas com grande número de participantes, mantendo a latência em níveis baixos mesmo com o crescimento da cadeia.

Em relação à confidencialidade, os resultados mostram que o custo associado, tanto para criptografia quanto para descriptografia, é desprezível quando comparado às operações de assinatura e verificação. Em todos os cenários avaliados, os tempos observados para algoritmos simétricos, como AES e ChaCha20, situam-se na ordem de microssegundos, não impactando significativamente a latência total das cadeias de assinatura.

Esse comportamento é consistente ao longo das diferentes linguagens de programação e dos números de processos analisados, evidenciando que o uso de criptografia simétrica não representa um fator limitante para a escalabilidade do sistema quando o tamanho das mensagens é pequeno. Mesmo com o aumento do número de participantes na cadeia, o custo adicional introduzido pela confidencialidade permanece praticamente constante e várias ordens de magnitude inferior ao custo das operações assimétricas.

Por fim, a Figura 1 apresenta o mapa de calor da latência média total considerando o cenário sem mecanismos de confidencialidade. Observa-se que o principal fator determinante da latência é o número de processos, evidenciado pelo gradiente dominante ao longo do eixo horizontal. À medida que novos processos são adicionados à cadeia, a latência cresce de forma aproximadamente linear. Esse comportamento é esperado, uma vez que cada processo adicional implica a execução de operações de verificação das assinaturas anteriores e a geração de uma nova assinatura, resultando em um custo cumulativo ao longo da cadeia.

Por outro lado, o impacto do tamanho do *payload* mostra-se significativamente menor. Para tamanhos menores (até 128 bytes), a variação de latência é praticamente imperceptível, indicando que o custo das operações criptográficas assimétricas domina o



(a) Ed25519 — C++ **(b) Ed25519 — Go** **(c) Ed25519 — Python**

(d) RSA-2048 — C++ **(e) RSA-2048 — Go** **(f) RSA-2048 — Python**

Figura 1. Latência média total sem confidencialidade.

tempo total. Mesmo para *payloads* maiores (512 e 1024 bytes), o aumento de latência é moderado quando comparado ao impacto do crescimento do número de processos.

Além disso, a homogeneidade das cores ao longo do eixo vertical para valores fixos de processos reforça que o custo marginal associado ao aumento do *payload* é relativamente baixo. Isso sugere que, dentro dos limites avaliados, o sistema apresenta boa estabilidade em relação ao volume de dados, mantendo desempenho consistente para diferentes tamanhos de mensagem.

De forma geral, embora C++ e, principalmente, Python apresentem uma dispersão de tempo significativa, Go demonstra um desvio padrão consistentemente baixo, sugerindo uma estabilidade de tempo de execução superior (veja dados completos no GitHub).

5.2. Crescimento do Tamanho das Mensagens

A Tabela 2 apresenta o crescimento do tamanho das mensagens ao longo da cadeia de assinaturas, considerando um *payload* inicial de 8 bytes e cenários com e sem confidencialidade. Observa-se um crescimento linear em função do número de processos, resultante da adição cumulativa de assinaturas digitais a cada elo da cadeia.

Este comportamento é independente da linguagem de implementação, uma vez que o tamanho das mensagens é determinado exclusivamente pelo algoritmo de assinatura utilizado. Assim, trata-se de uma propriedade estrutural do modelo de encadeamento, e não da plataforma de execução.

A diferença entre os cenários “Aberto” e “Cripto” permanece constante ao longo de toda a cadeia, indicando que o uso de criptografia simétrica introduz um *overhead* fixo,

Tabela 2. Tamanho total das mensagens (bytes) ao longo da cadeia de assinaturas, considerando *payload* inicial de 8 bytes e cenários sem (Aberto) e com confidencialidade (Cripto).

N	RSA		Ed25519	
	Aberto	Cripto	Aberto	Cripto
1	264	292	72	100
2	520	548	136	164
3	776	804	200	228
4	1.032	1.060	264	292
5	1.288	1.316	328	356
6	1.544	1.572	392	420
7	1.800	1.828	456	484
8	2.056	2.084	520	548

independentemente do número de processos.

Comparando os algoritmos, observa-se que o RSA apresenta crescimento mais acentuado devido ao maior tamanho de suas assinaturas, enquanto o Ed25519 mantém um aumento mais moderado. Como consequência, o impacto estrutural do encadeamento é significativamente maior no caso do RSA.

Este crescimento tem implicações diretas na camada de rede. Considerando um MTU típico de 1.500 bytes, verifica-se que, para RSA, o limite é ultrapassado a partir de seis processos, mesmo com *payload* inicial reduzido. A partir desse ponto, ocorre fragmentação de pacotes, o que pode introduzir *overhead* adicional e impactar negativamente a latência e a confiabilidade da comunicação. Para Ed25519, por outro lado, os valores permanecem abaixo do MTU em todos os cenários analisados.

A análise dos dados nos demais cenários indica que esse comportamento se mantém para diferentes tamanhos de mensagem. O aumento do *payload* desloca linearmente o tamanho total das mensagens, antecipando o ponto de violação do MTU no caso do RSA, mas não altera o padrão de crescimento nem a diferença relativa entre os algoritmos. Em particular, para *payloads* maiores (e.g., 512 bytes e 1.024 bytes), a fragmentação ocorre com menos processos, reforçando que o limite de MTU é rapidamente atingido em cadeias mais longas.

5.3. Impacto das Cadeias de Assinaturas em Diferentes Topologias

O impacto do uso de cadeias de assinaturas digitais em sistemas distribuídos depende fortemente da topologia de comunicação adotada pelo sistema. Como cada elo da cadeia adiciona uma assinatura e requer verificação das assinaturas anteriores, o custo computacional e o crescimento estrutural da mensagem podem afetar de maneira distinta diferentes arquiteturas de rede.

Em topologias lineares, o modelo de cadeia de assinaturas se adapta naturalmente à estrutura do sistema. Nestes cenários, o crescimento do custo de verificação acompanha o comprimento da cadeia e permanece proporcional ao número de participantes. Como cada mensagem percorre um único caminho, o impacto na escalabilidade tende a ser previsível e relativamente controlado.

Em topologias em malha ou redes peer-to-peer, o impacto pode ser mais significativo. Nessas arquiteturas, mensagens frequentemente são disseminadas por múltiplos caminhos e podem alcançar um grande número de nós. Se cada retransmissão adicionar

uma nova assinatura, o crescimento da cadeia pode tornar-se rapidamente elevado, aumentando tanto o custo de verificação quanto o tamanho da mensagem. Esse comportamento pode limitar a eficiência de protocolos de disseminação, especialmente em sistemas com alta cardinalidade.

Em arquiteturas hierárquicas, como árvores, o comportamento pode variar dependendo da posição dos nós na estrutura. Nós próximos à raiz da árvore podem receber mensagens com maior número de assinaturas acumuladas, resultando em maior custo de verificação. Entretanto, o impacto total pode ser distribuído entre diferentes subárvores. Nesse contexto, cadeias de assinaturas podem fornecer garantias de integridade sequencial e rastreabilidade das operações ao longo da hierarquia, ao custo de aumento progressivo do tamanho da mensagem.

Uma topologia com grande potencial nesse contexto é o hipercubo ou seu correspondente virtual VCube [Duarte Jr et al. 2023]. Em um hipercubo de dimensão d, o número de nós é $N = 2^d$, e a distância máxima entre dois nós é $d = \log_2 N$. Quando cadeias de assinaturas são utilizadas em hipercubos, cada etapa de comunicação adiciona um novo elo à cadeia. Como o número de saltos necessários para alcançar qualquer nó cresce apenas de forma logarítmica, o comprimento máximo da cadeia também passa a ser limitado por $O(\log N)$ em processos de *broadcast* ou agregação estruturada [Jeanneau et al. 2017, Ruchel et al. 2024]. Isso representa uma vantagem significativa em relação a topologias lineares, nas quais o comprimento da cadeia cresce proporcionalmente ao número total de participantes. Além disso, a estrutura regular do hipercubo permite distribuir uniformemente o custo de verificação entre os nós do sistema. Cada nó participa de no máximo d interações ao longo de um processo de disseminação completo, o que contribui para manter o custo computacional global equilibrado mesmo em sistemas com grande número de participantes.

6. Trabalhos Relacionados

O conceito de cadeias de assinatura foi formalizado por Saxena e Soh [Saxena and Soh 2008], que introduziram o paradigma de *one-way signature chaining* como um mecanismo para registrar sequências verificáveis de interações entre múltiplos participantes. Nesse modelo, cada participante adiciona uma assinatura digital dependente do estado anterior, formando uma estrutura resistente à modificação de elos intermediários.

Extensões desse modelo foram exploradas em diferentes contextos de segurança. Bicakci e Baykal [Bicakci and Baykal 2003] propõem o uso de cadeias de assinatura como alternativa a cadeias de *hash* em mecanismos de autenticação baseada em senha de uso único (OTP), destacando maior flexibilidade e resistência a comprometimento de servidores. Van der Linde [van der Linde 2018] investiga o uso de cadeias de assinaturas em cenários pós-quânticos, integrando assinaturas de uso único a estruturas semelhantes a blockchains.

No contexto de sistemas distribuídos, cadeias de assinatura têm sido utilizadas para proveniência de dados e auditoria de execução. Ahmed et al. [Ahmed et al. 2016] propõem um esquema baseado em assinaturas agregadas para registrar transformações sucessivas em dados distribuídos, reduzindo o tamanho das provas criptográficas. De forma semelhante, Chen et al. [Chen et al. 2023] exploram o uso de cadeias de assina-

tura em protocolos de disseminação de blocos, permitindo rastrear rotas de propagação e incentivar comportamentos cooperativos na rede.

Apesar desses avanços, a maioria dos trabalhos concentra-se em aspectos criptográficos, modelos de segurança ou aplicações específicas, sem avaliar de forma sistemática o impacto cumulativo das cadeias de assinatura no desempenho de sistemas distribuídos. Em particular, há uma lacuna na literatura quanto à análise experimental que considere simultaneamente (i) o crescimento estrutural das mensagens, (ii) o custo sequencial de verificação, (iii) a influência de diferentes algoritmos criptográficos e (iv) o comportamento em diferentes linguagens de programação.

Este trabalho diferencia-se ao fornecer uma avaliação experimental abrangente desses fatores, incluindo a decomposição dos custos de assinatura, verificação e criptografia, bem como a análise do impacto do crescimento das mensagens em limites práticos de rede, como a MTU.

7. Conclusão

Este trabalho apresentou uma avaliação experimental do impacto de cadeias de assinaturas digitais em sistemas distribuídos, considerando diferentes linguagens de programação, algoritmos criptográficos, tamanhos de *payload* e a presença de mecanismos de confidencialidade. A análise contemplou não apenas métricas tradicionais de desempenho, como latência, mas também a decomposição dos custos de assinatura, verificação e criptografia, bem como o crescimento estrutural das mensagens.

Os resultados demonstram que o principal fator limitante das cadeias de assinatura é o crescimento linear associado ao número de processos, tanto em termos de custo computacional quanto de tamanho das mensagens. Em particular, a verificação cumulativa das assinaturas e a geração de novos elos impõem um custo proporcional ao comprimento da cadeia, impactando diretamente a latência total.

Entre os algoritmos avaliados, o Ed25519 apresentou desempenho superior e melhor escalabilidade em comparação ao RSA-2048, especialmente em cenários com maior número de processos. Esse comportamento está associado ao menor custo de assinatura e ao tamanho reduzido das assinaturas, o que também contribui para um crescimento mais controlado das mensagens. A confidencialidade introduz um custo adicional mensurável, porém significativamente inferior ao das operações assimétricas, não constituindo fator limitante para a escalabilidade do sistema. Por outro lado, o crescimento linear da cadeia pode levar à violação do limite de MTU em redes IP, mais rápido no caso do RSA.

Como trabalhos futuros, pretende-se investigar a aplicação das cadeias de assinaturas na construção de sistemas distribuídos tolerantes a falhas bizantinas. Concretamente, o objetivo é investigar a aplicação do diagnóstico baseado em comparações [Ziwich and Duarte 2016] estendido com cadeias de assinaturas para obter estratégias eficazes de tolerância a falhas bizantinas. Em particular é objetivo explorar o uso de topologias estruturadas como o VCube para reduzir o comprimento efetivo das cadeias e mitigar o custo de verificação cumulativa. Esses sistemas devem então ser testados em ambientes distribuídos reais, incorporando latência de rede e variabilidade de comunicação, bem como avaliar técnicas de otimização, como assinaturas agregadas e o impacto de primitivas pós-quânticas.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Referências

Ahmed, I., Khan, A., Khan, M. S., and Ahmed, M. (2016). Aggregated signatures for chaining: A secure provenance scheme. In *IEEE Trustcom/BigDataSE/ISPA*, pages 2012–2017.

Bicakci, K. and Baykal, N. (2003). Improving the security and flexibility of one-time passwords by signature chains. *Turkish Journal of Electrical Engineering and Computer Sciences*, 11(3):223–236.

Chen, P., Li, Z., Ling, X., and Wang, J. (2023). Accelerated gossip protocol for incentivizing block propagation. In *IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 170–175.

Duarte Jr, E. P., Rodrigues, L. A., Camargo, E. T., and Turchetti, R. C. (2023). The missing piece: a distributed system-level diagnosis model for the implementation of unreliable failure detectors: Ep duarte et al. *Computing*, 105(12):2821–2845.

Jeanneau, É., Rodrigues, L. A., Arantes, L., and Duarte Jr, E. P. (2017). An autonomic hierarchical reliable broadcast protocol for asynchronous distributed systems with failure detection. *Journal of the Brazilian Computer Society*, 23(1):15.

Josefsson, S. and Liusvaara, I. (2017). Edwards-Curve Digital Signature Algorithm (EdDSA). RFC 8032.

Moriarty, K., Kaliski, B., Jonsson, J., and Rusch, A. (2016). PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017.

Nir, Y. and Langley, A. (2018). ChaCha20 and Poly1305 for IETF Protocols. RFC 8439.

Ruchel, L. V., de Camargo, E. T., Rodrigues, L. A., Turchetti, R. C., Arantes, L., and Duarte Jr, E. P. (2024). Scalable atomic broadcast: A leaderless hierarchical algorithm. *Journal of Parallel and Distributed Computing*, 184:104789.

Salowey, J., Choudhury, A., and McGrew, D. (2008). AES Galois Counter Mode (GCM) Cipher Suites for TLS. RFC 5288.

Saxena, A. and Soh, B. (2008). One-way signature chaining: A new paradigm for group cryptosystems. *Intl'l J. of Information and Computer Security*, 2(3):268–296.

Tandi, M. (2019). Digital signature chains for secure document exchange in erp-based procurement workflows. Int'l J. of Advanced Engineering and Emerging Technologies.

van der Linde, W. (2018). Post-quantum blockchain using one-time signature chains. Master's thesis, Radboud University. Master's Thesis.

Ziwich, R. P., Duarte, E., and Albini, L. C. P. (2005). Distributed integrity checking for systems with replicated data. In *11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, volume 1, pages 363–369. IEEE.

Ziwich, R. P. and Duarte, E. P. (2016). A nearly optimal comparison-based diagnosis algorithm for systems of arbitrary topology. *IEEE Transactions on Parallel and Distributed Systems*, 27(11):3131–3143.