Edital FAPESC 27/2021

Aplicação de algoritmos de compartilhamento de segredos e anonimização de dados para adequação de sistemas à LGPD

Santa Catarina

2021-2023

Edital FAPESC 27/2021

Aplicação de algoritmos de compartilhamento de segredos e anonimização de dados para adequação de sistemas à LGPD

Relatório técnico apresentando os desenvolvimentos e resultados obtidos na pesquisa fomentada pelo edital 27/2021 - FAPESC.

A Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina

Luciano Barreto

Santa Catarina

2021-2023

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

TI	Tecnologia da Informação
LGPD	Lei Geral de Proteção de Dados
VSS	*Verifiable Secret Sharing*
ANPD	Autoridade Nacional de Proteção de Dados do Brasil
PVSS	*Publicly Verifiable Secret Sharing*
CIO	*Chief Information Office*
DP	*Differential Privacy*
API	*Application Programming Interface*
REST	*Representational State Transfer*
LAN	*Local Area Network*
CGNAT	*Carrier-Grade Network Address Translation*
TCP	*Transmission Control Protocol*
IDS	*Intrusion Detection System*
JSON	*JavaScript Object Notation*
IQR	*Interquartile Range*
HTTP	*Hypertext Transfer Protocol*
UX	*User Experience*
IDS	*Intrusion Detection System*

Lista de símbolos

ϵ	epsilon

Sumário

Introdução

A Lei Geral de Proteção de Dados (LGPD), sancionada em 2018, estabelece um novo padrão regulatório no Brasil para a proteção de informações pessoais. Além de consolidar direitos fundamentais, a LGPD introduz princípios como a minimização de dados, finalidade, adequação e transparência. Enquanto a lei fortalece a proteção dos direitos humanos, do ponto de vista jurídico, ela também apresenta desafios técnicos significativos, especialmente quando se trata de armazenar, processar e transmitir dados em sistemas informatizados.

Neste cenário, o presente relatório, desenvolvido sob o Edital FAPESC 27/2021, aprofunda-se na aplicação de algoritmos para armazenamento e transmissão de dados pessoais de forma segura, como os de compartilhamento de segredos (esquema de Shamir, por exemplo) e técnicas de anonimização (como k-anonimato e l-diversidade). Essas abordagens são cruciais para assegurar que as organizações não apenas cumpram a LGPD, mas também implementem práticas de segurança robustas, garantindo que os dados pessoais sejam processados sem expor informações sensíveis.

A pesquisa detalhada neste documento aborda aplicações práticas em cenários comuns a muitas empresas, como sistemas de CRM, bancos de dados de RH e plataformas de e-commerce. Através de uma combinação de teoria e prática, discutimos como infraestruturas de Tecnologia da Informação (TI) podem ser adaptadas e otimizadas para atender às rigorosas demandas da LGPD, mantendo, ao mesmo tempo, alta performance e eficiência operacional.

Em conclusão, este relatório vai além de simplesmente identificar os desafios associados à LGPD. Ele fornece detalhes técnicos, apresenta soluções e serve como um guia para profissionais da área de TI, pesquisadores e organizações que buscam não apenas conformidade mas também implementação prática destes mecanismos em suas soluções tecnológicas viáveis na prática.

Proposta do Projeto

Esta seção resume a proposta de projeto enviada e retoma os conceitos necessários para estabelecer um melhor entendimento das demais seções deste relatório. As informações desta seção são somente cópias da proposta do projeto.

1 Descrição da problemática

Atualmente, todas as organizações utilizam sistemas informatizados. Seja para gerenciar vendas, disponibilizar resultados médicos online ou permitir acesso a dados acadêmicos. O denominador comum entre esses sistemas é o armazenamento de informações pessoais, como documentos, dados financeiros, registros médicos e acadêmicos, entre outros. Esses dados podem ser armazenados internamente ou em provedores de computação em nuvem, uma tendência crescente(GARTNER, 2021).

A introdução da LGPD (Lei Geral de Proteção de Dados Pessoais) (BRASIL, 2018) elevou a importância da gestão de dados pessoais. Segundo a LGPD, as organizações são responsáveis pelo tratamento desses dados. Qualquer vazamento ou exposição que prejudique os indivíduos pode resultar em sanções, processos e multas. O prejuízo é definido como qualquer exposição que viole direitos humanos. Portanto, a gestão de dados tornou-se crítica, exigindo que as organizações se adequem à LGPD e fortaleçam a confidencialidade em seus sistemas. Cumprir a LGPD beneficia as organizações e protege os direitos dos usuários.

Para garantir o armazenamento seguro de dados, várias técnicas podem ser empregadas. A encriptação, por exemplo, cifra os dados, tornando-os indecifráveis sem a chave correta (STALLINGS, 2015). O compartilhamento de segredos, por outro lado, divide informações sensíveis em várias partes, onde cada parte, isoladamente, não revela o conteúdo original (SHAMIR, 1979; SCHOENMAKERS, 1999). A anonimização é outra abordagem que despersonaliza informações, tornando difícil associá-las a indivíduos específicos (RAGHUNATHAN, 2013).

Dada a necessidade de conformidade com a LGPD e os desafios do

armazenamento de dados, é essencial explorar e implementar técnicas de armazenamento seguro. Esta pesquisa foca em técnicas de compartilhamento de segredos e anonimização, especialmente para dados pessoais. Além disso, será analisado o custo computacional desses algoritmos e a viabilidade de soluções híbridas.

O estudo de técnicas para armazenamento seguro de dados é uma área de pesquisa em expansão há anos. Embora muitas dessas técnicas apresentem limitações computacionais, tornando-as impraticáveis em cenários reais, o avanço no poder de processamento e a redução de custos dos computadores modernos abrem novas possibilidades.

A eficácia dos algoritmos de compartilhamento de segredos pode variar conforme a natureza dos dados que se busca proteger. Em sistemas onde pequenos atrasos operacionais são aceitáveis, essas técnicas podem ser implementadas com sucesso. Neste projeto, o foco recai sobre os dados sensíveis dos usuários, conforme estipulado pela LGPD. Embora nem todos os resultados sejam ideais, e nem todas as informações possam ser protegidas por esses métodos, o estudo serve como um incentivo para explorar sua aplicação em conjuntos de dados específicos.

A categorização de informações, tanto pessoais quanto sensíveis, pode ajudar a determinar quais dados serão protegidos por essas técnicas, tornando sua implementação mais viável. Ao classificar os dados, podemos dividi-los em “micro”, referindo-se a dados pessoais como nome, sobrenome e endereço, e “macro”, que se refere a informações associadas a uma pessoa, como histórico de compras online e registros médicos. Em sistemas relacionais, esses dois conjuntos de dados estão interligados, associando uma pessoa às suas informações.

Considerando essa inter-relação e as limitações das técnicas de compartilhamento de segredos, a anonimização surge como uma alternativa complementar. Essa técnica pode desvincular os dados pessoais das informações associadas, protegendo a identidade do indivíduo em caso de vazamentos. Além disso, permite determinar quais informações serão protegidas por algoritmos de compartilhamento de segredos.

Dado o risco associado ao vazamento de informações pessoais e a existência de técnicas que podem atenuar esses riscos, este projeto se justifica como uma tentativa de abordar essa demanda e oferecer soluções tangíveis para o problema.

2 Objetivos Gerais

O objetivo geral desta pesquisa é o desenvolvimento de soluções práticas para o armazenamento seguro de dados, principalmente no que tange dados pessoais e suas adequações a LGPD. Para este fim, o estudo e aplicação dos algoritmos de compartilhamento de segredos, assim como técnicas de anonimização de dados serão objetos de estudo, quanto a sua adequação e limitações diante das tecnologias atuais.

2.1 Objetivos Específicos

- Analisar a Lei Geral de Proteção de Dados;
- Buscar a definição dos dados pessoais sensíveis que são geralmente armazenados em sistemas informatizados;
- Categorizar os dados sensíveis em dados pessoais (nome, data de nascimento, RG, CPF, dentre outros) e dados pertencentes à pessoa (pedido online para uma pessoa, dados médicos de uma pessoa, dados acadêmicos de uma pessoa);
- Revisar a literatura em busca de novas abordagens para algoritmos de compartilhamento de segredos e anonimização de dados;
- Implementar abordagens existentes sobre compartilhamento de segredos e anonimização de dados;
- Analisar a viabilidade da aplicação de técnicas de compartilhamento de segredos em dados pessoais e dados pertencentes às pessoas (este para base de comparação);
- Definir limites viáveis da aplicação das abordagens para ambientes reais (tempo de resposta aceitável em sistemas);
- Implementar algoritmos híbridos entre compartilhamento de segredos e anonimização de dados;
- Analisar a viabilidade das abordagens híbridas;
- Otimizar algoritmos de compartilhamento de segredo com a criação de bases de dados de números primos;

- Analisar a viabilidade da abordagem otimizada quanto ao tempo de execução e segurança do sistema;

- Compartilhar as experiências e o conhecimento do projeto de pesquisa a comunidade.

3 Estado da Arte

Uma das soluções usuais para o armazenamento seguro de dados é o uso de encriptação. Esta técnica garante que a confidencialidade dos dados seja mantida mesmo que estes sejam expostos. Na literatura, encontramos exemplos de sistemas de armazenamento seguro de arquivos em nuvem baseados em transformações criptográficas e códigos de correção de erro (BESSANI et al., 2011; KUMAR et al., 2012; SUJANA et al., 2013). Estes buscam manter a confidencialidade dos dados de usuários em ambientes onde não se tem muito controle sobre a informação, como o comportamento honesto, mas curioso, dos provedores de computação em nuvem. Em geral, estas abordagens requerem que as informações sejam criptografadas para garantir sua confidencialidade e que as chaves criptográficas utilizadas não sejam expostas.

Outra técnica menos comum para o armazenamento seguro de dados é o compartilhamento de segredos. Uma destas técnicas, descrita por (SHAMIR, 1979), utiliza transformações (interpolações polinomiais) para dividir um segredo S em n partes, chamadas de compartilhamentos (*shares*). Cada uma destas partes do segredo não revela qualquer informação sobre o segredo S. Esta informação só pode ser reconstruída a partir da obtenção de um conjunto mínimo de t partes ($0 < t \leq n$). Assim, a posse de qualquer conjunto de k partes do segredo, com $k < t$, não permite que o segredo S seja recuperado. No entanto, partes podem ser alteradas, dificultando a reconstrução da informação.

Para lidar com cenários onde as partes estão sob ação de entidades não confiáveis, como servidores comprometidos por invasores, mecanismos de compartilhamento de segredo verificável (VSS) podem ser utilizados. Nestes mecanismos, descritos por Schoenmakers (SCHOENMAKERS, 1999), uma parte de um segredo fornecido a um combinador pode ter sua validade verificada. Estes esquemas verificáveis são essenciais para garantir que o resultado de uma recuperação de segredo não seja afetado por valores

errôneos devido a partes incorretas fornecidas por participantes maliciosos, acelerando assim a obtenção do resultado correto da computação.

Embora estas técnicas ofereçam segurança no armazenamento de dados sensíveis, existem limitações quanto à viabilidade de sua aplicação. Quanto maior o conjunto de dados a serem tratados, maior é o tempo de computação necessário, tanto para o processo de criação dos segredos (divisão em partes) quanto para a junção dos segredos (recuperação da informação).

Consciente destas limitações, Barreto (BARRETO et al., 2017; BARRETO, 2017) buscou aplicar as técnicas em um conjunto de dados delimitado: as credenciais de usuários. Estas são compostas por pequenos conjuntos de dados, como *logins*, senhas, nome e sobrenome, data de nascimento, níveis de acesso, entre outras informações de tamanho limitado. Os resultados apresentados nas pesquisas mostraram a viabilidade do uso destes algoritmos aplicados a este conjunto sensível de dados, servindo como ponto de partida para aplicação em outros conjuntos de dados, como os dados sensíveis de usuários conforme a LGPD. No entanto, somente testes práticos podem confirmar sua aplicabilidade em sistemas e tecnologias atuais.

As técnicas de anonimização de dados (RAGHUNATHAN, 2013) também têm sido utilizadas em ambientes onde o armazenamento de informações sensíveis está sujeito a vazamentos. Diferentemente da criptografia ou compartilhamento de segredos, a anonimização foca em remover a correlação de um dado com o indivíduo a quem este dado pertence. O objetivo é despersonalizar uma informação, de modo que não se consiga relacionar a informação com sua *persona* (pessoa física ou jurídica). Em (BAYARDO; AGRAWAL, 2005), são apresentados estudos da aplicação prática da técnica e sua viabilidade, focando na técnica denominada k-anonymization. Já em (DOMINGO-FERRER, 2019), é feito um estudo da aplicação das técnicas em conformidade com as Leis Gerais de Proteção de Dados Europeias, relacionando-se com os objetivos deste projeto. Em (ZHOU; PEI; LUK, 2008), são descritas técnicas aplicadas aos dados das redes sociais, uma preocupação que já existe há mais de 10 anos e que, atualmente, é foco de estudos.

A literatura sobre os temas é vasta, mas ainda há espaço para estudos de aplicações práticas, especialmente no cenário brasileiro com a recente implementação da LGPD. Comparado com regiões como a Europa, o Brasil ainda está em estágios iniciais de adaptação e implementação de práticas de proteção de dados.

Etapa 1

4 Revisão da LGPD, literatura e ferramentas

Nesta etapa do projeto os participantes deverão conhecer a LGPD e se atualizar sobre a literatura e ferramentas existentes de compartilhamento de segredos e anonimização de dados.

4.1 Metas

- Leitura e discussão com os participantes do projeto sobre a LGPD;
- Buscar a definição dos dados pessoais sensíveis que são geralmente armazenados em sistemas informatizados;
- Categorizar os dados sensíveis em dados pessoais (nome, data de nascimento, RG, CPF, dentre outros) e dados pertencentes à pessoa (pedido online para uma pessoa, dados médicos de uma pessoa, dados acadêmicos de uma pessoa);
- Buscar artigos científicos nos principais periódicos ou outras fontes de artigos indexados;
- Buscar ferramentas já implementadas para armazenamento seguro para aplicação das mesmas.

4.2 Resultados

Durante esta etapa do projeto, foi realizada uma revisão bibliográfica, na qual foram levantados artigos e livros pertinentes ao tema de pesquisa. Esse levantamento foi fundamental para embasar e direcionar os passos do estudo e desenvolvimento do projeto, garantindo que o projeto estivesse alinhado com as publicações e descobertas na área.

4.2.1 Lei Geral de Proteção de Dados Pessoais

Sobre o tema LGPD, foi realizada a leitura e estudos sobre a Lei nº 13.709 (BRASIL, 2018). Em resumo, a LGPD foi criada para proteger os

direitos fundamentais de liberdade e de privacidade dos cidadãos brasileiros, estabelecendo regras claras sobre o tratamento de dados pessoais. Esta lei se aplica a qualquer operação de tratamento de dados (geração, manipulação, transferência) realizada por pessoa física ou jurídica, independentemente do meio, do país de sua sede ou do país onde estejam localizados os dados, desde que a atividade de tratamento seja realizada no território nacional. Nesta lei, destaca-se que os dados pessoais representam a informação mais valiosa que uma empresa pode possuir e tratar. Os dados pessoais referem-se a qualquer informação relacionada a uma pessoa natural identificada ou identificável. O tratamento de dados pessoais só pode ser realizado com o consentimento explícito e informado do titular. Existem algumas exceções, como cumprimento de obrigação legal ou proteção do crédito.

A seguir, são apresentados exemplos práticos de tratamentos de dados que podem gerar infrações:

- Uma loja online coleta e utiliza o e-mail de seus clientes para enviar campanhas de marketing sem que eles tenham dado permissão explícita para isso.
- Um ex-cliente de um banco solicita a exclusão de todos os seus dados, mas o banco não atende à solicitação.
- Uma clínica médica compartilha dados de saúde de seus pacientes com empresas farmacêuticas sem uma base legal clara para essa ação.
- Uma empresa de recrutamento mantém currículos e informações de candidatos que participaram de processos seletivos há anos, mesmo sem necessidade para tal retenção.
- Um hospital compartilha informações médicas de um paciente com uma empresa de seguros sem o consentimento do paciente.
- Um e-commerce sofre um ataque cibernético e tem dados de cartões de crédito de seus clientes vazados por não ter sistemas de segurança adequados. A empresa decide não informar seus usuários sobre o ocorrido, nem reportar à ANPD (Autoridade Nacional de Proteção de Dados do Brasil).

Em caso de infrações, as sanções podem variar desde advertências até multas de até 2% do faturamento da empresa, limitada a R$ 50 milhões por infração.

Para determinarmos a que se refere os dados pessoais, utilizamos as seguintes referências:

BRASIL. Lei nº 13.709, de 14 de agosto de 2018 (BRASIL, 2018). A Lei Geral de Proteção de Dados Pessoais (LGPD) é a principal referência legal no Brasil sobre proteção de dados pessoais. Ela define dados pessoais como "informação relacionada a pessoa natural identificada ou identificável".

Pinheiro, Patricia Peck (PINHEIRO, 2020). Proteção de Dados Pessoais: Comentários à Lei n. 13.709/2018 (LGPD). São Paulo: Revista dos Tribunais. Este livro oferece uma análise detalhada da LGPD, com comentários sobre cada artigo da lei.

TJCE (CEARÁ, 2021). Dado pessoal, dado pessoal sensível e dado anonimizado.

BNDES (SOCIAL, 2021). Lei Geral de Proteção de Dados (LGPD).

Após a análise das referências e outras buscas, foi realizada uma discussão sobre as distinções entre "dados pessoais" e "dados pertencentes às pessoas". A partir das pesquisas realizadas, entendeu-se que "dados pessoais" são informações que se referem diretamente a um indivíduo identificado ou identificável. Estes podem englobar dados como nome, endereço, data de nascimento, número de telefone, endereço de e-mail, identificações como CPF e RG, dados de localização e até identificadores online, como endereços IP e *cookies*.

Por outro lado, "dados pertencentes às pessoas" representam uma categoria mais abrangente. Estes podem não identificar diretamente um indivíduo, mas ainda assim são propriedade ou foram gerados por ele. Isso inclui, por exemplo, conteúdos digitais em redes sociais, registros de compras, informações coletadas por aplicativos de saúde e *fitness*, e registros médicos como anamneses e exames.

A partir destes dados, foram geradas duas tabelas, distinguindo estes possíveis dados. Nestas tabelas são apontados o nome do dado e uma previsão da quantidade de dados que cada uma dela aproximadamente ocupa (em bytes). Estes dados podem variar de sistema para sistema, então são somente previsões. As tabelas estão descritas abaixo:

Tabela 1 – Resumo de tamanhos médios de dados pessoais

Dado	**Tamanho Médio**
Nome	50 caracteres
Sobrenome	50 caracteres
Endereço	130 caracteres
Bairro	72 caracteres
Cidade	30 caracteres
Estado	20 caracteres
RG	14 caracteres
CPF	11 caracteres
Gênero	9 caracteres
Data de nascimento	19 caracteres
Local de nascimento	100 caracteres
Telefone fixo	10 caracteres
Número de celular	11 caracteres
Latitude	12 caracteres
Longitude	12 caracteres
Retrato	500 bytes

Tabela 2 – Estimativa de tamanhos de diferentes tipos de dados

Dados	**Estimativa**
Dados médicos (por registro)	10.000 caracteres
Dados de compra (por registro)	5.000 caracteres
Laudo de exames (resultados)	3.000 caracteres
Prescrição médica (receita médica)	1.000 caracteres

4.2.2 Compartilhamento de Segredos

Sobre o tema compartilhamento de segredos foram estudados os principais algoritmos e também aqueles em que implementações existentes já puderam ser encontradas. O primeiro e mais famoso deles é o algoritmo de Shamir (SHAMIR, 1979). Este artigo apresenta um método para dividir dados S em n partes de tal forma que S seja facilmente reconstruível a partir de quaisquer k partes, mas o conhecimento de $k-1$ peças ou menos não revela absolutamente nenhuma informação sobre S. Esta técnica é fundamental para a construção de esquemas robustos de gerenciamento de chaves para sistemas criptográficos. O esquema garante que os sistemas operem de forma segura e confiável, mesmo que os entidades maliciosas destruam metade

Listagem 1 – Exemplo de execução do esquema de Shamir

```
Segredo: "Este é um segredo."

Partes
[Parte 1]: fPsS8hmO6CWp4PKyJbPw/iFtOw
[Parte 2]: QoJDhX6eoNu5viKuclOZaJU/Yw
[Parte 3]: ewolEkfT4d5lM/BvMocb89A9dg
[Parte 4]: glvR0yxCrShJofLBLTFW52qCNQ
[Parte 5]: u9O3RBUP7C2VLCAAbeXUfC+AIA
```

das partes e as violações de segurança exponham todas as partes restantes, exceto uma.

O artigo apresenta um esquema de limite (k, n) baseado em interpolação polinomial. Neste esquema, dados k pontos no plano bidimensional, existe um único polinômio $q(x)$ de grau $k-1$ que se ajusta a esses pontos. Os dados S são divididos em partes avaliando este polinômio em diferentes pontos. O conhecimento de qualquer k ou mais partes permite a reconstrução de S, enquanto $k-1$ ou menos partes não fornecem informações sobre S.

$$q(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1}$$

Onde $a_0, a_1, \ldots, a_{k-1}$ são coeficientes do polinômio.

Um exemplo do resultado final do compartilhamento de Shamir é apresentado na listagem 1.

Outro artigo revisado para o desenvolvimento do projeto foi o artigo intitulado **"A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting"** (SCHOENMAKERS, 1999). Neste artigo o autor investiga o domínio do compartilhamento de segredos em criptografia, introduzindo um novo esquema de compartilhamento de segredos publicamente verificável (**PVSS - Publicly Verifiable Secret Sharing Scheme**). Os esquemas tradicionais de compartilhamento de segredos verificáveis são criados de modo a que apenas os participantes que recebem as partes possam verificar a sua autenticidade. No entanto, o esquema PVSS quebra esta norma ao permitir que qualquer parte externa verifique a validade das ações (partes) distribuídas, aumentando a transparência e a confiabilidade.

O esquema PVSS proposto destaca-se por oferecer melhorias acen-

tuadas em duas áreas principais: eficiência e pressupostos fundamentais de intratabilidade. Quando se trata de eficiência, o esquema opera com um tempo de execução de $O(nk)$. Onde k representa um parâmetro de segurança, garantindo a robustez do esquema contra ameaças potenciais, enquanto n denota o número de participantes envolvidos no processo de compartilhamento de segredo. Um exemplo do resultado final do compartilhamento PVSS é apresentado na listagem 2.

Listagem 2 – Exemplo de execução do esquema PVSS

```
Segredo: "Este é um segredo."

Partes
[Parte 1]: aMVLafrd8raw/jJ/ctnUs7KYYcXOW6NedIfcRQvdkpdMC+bfK8YxC+
    VE0poR4Z+l4aSF/jz/phsV7wisofvoaQ
[Parte 2]: AJFKqDA9egs/OD71pXYLledSTLwa0jNdB51viUVGiHp8OK9wbbm33Y4+
    mLr7WHVB32OCFH6Q3Mwoi4p5TWHoCoE
[Parte 3]: Dnzmhs2oU5fTJ6adJI6+
    N8zhLZ81AHjNpUlWUv6RLqqrAraT03Waz0aPCMzyQwsMS3HLysreeyzIJwUIvIMB6g

[Parte 4]: PErZicg6qRnXmjnxh7mPqqNc5XsOTgtLQtXTE2OuaveRRHOTUmCRUTl/
    K2Waaw/fIE2bV4tgL12OAZc/iDEXMg
[Parte 5]: AcdKLYt6Ss3JErISzj8tv0ML0BxIHa3QcK7hozMcdWsMvSkx49SVrH6//
    Oxxu+/+yDkKnZkghoW9dlwxykzWDnQ
```

Em termos de segurança, o esquema PVSS está relacionado na suposição *Diffie-Hellman* e na sua variante decisória. Essas suposições criptográficas são amplamente reconhecidas e confiáveis no campo, garantindo que o esquema PVSS mantenha um alto nível de integridade de segurança.

Ainda sobre a técnica de segredos verificáveis foi realizada a leitura do artigo intitulado **“Publicly Verifiable Secret Sharing”** (STADLER, 1996). Além dos detalhes apresentados por (SCHOENMAKERS, 1999), este trabalho discute aplicações potenciais do PVSS, notadamente em criptossistemas de garantia e sistemas de pagamento digital com anonimato revogável. O artigo apresenta duas novas realizações de PVSS baseadas no criptossistema de *ElGamal*. Esses esquemas são projetados para compartilhar logaritmos discretos e raízes e-ésimas. O esquema de criptografia para as e-ésimas raízes permite uma prova eficiente de que um texto cifrado contém a e-ésima raiz de um determinado elemento.

Um terceiro estudo avaliado foi o artigo **“Secret sharing made**

short"(KRAWCZYK, 1993). No artigo o autor introduz uma variação do mecanismo de compartilhamento de segredos. O esquema de compartilhamento de segredos de Krawczyk, como outros esquemas de compartilhamento de segredos, divide um segredo em múltiplas partes (ou compartilhamentos) de modo que apenas um número específico dessas partes, quando combinadas, pode revelar o segredo original. No entanto, o que torna o esquema de Krawczyk único é a sua ênfase na robustez contra a trapaça. Ele se utiliza de 3 mecanismos: **Inicialização**: Um *dealer* confiável escolhe um segredo e gera compartilhamentos para cada participante; **Distribuição**: O dealer distribui as ações para cada participante. **Reconstrução**: Os participantes reúnem suas ações para reconstruir o segredo.

O esquema de Krawczyk usa interpolação polinomial, compromissos criptográficos e provas de conhecimento zero (*Zero-knowledge proof*). Para um polinômio $f(x)$ de grau t, o segredo é $f(0)$ e as ações são $f(1), f(2), \ldots$. Para verificar um compartilhamento $f(i)$, é usada uma prova de conhecimento zero. Dentre as aplicações possível do uso deste mecanismo estão principalmente *multi-party computing*, sistemas de votação seguros e protocolos criptográficos distribuídos. Um exemplo do resultado final do compartilhamento Krawczyk é apresentado na listagem 3.

Listagem 3 – Exemplo de execução do esquema de Krawczyk

```
Segredo: "Este é um segredo."

Partes
[Parte 1]: ft6LhEyxbA
[Parte 2]: 7IkGUPHsbA
[Parte 3]: 3yslMbdpbA
[Parte 4]: KQFuUyCCbA
[Parte 5]: GqNNMmYHbA
```

Além dos artigos acima descritos, outras duas variações de técnicas de compartilhamento de segredos foram utilizadas, baseadas na biblioteca *Archistar* (LORUENSER; HAPPE; SLAMANIG, 2015). Esta biblioteca inclui duas implementações de mecanismos de compartilhamento de segredos que agregam códigos de correção de erros, propostas por Rabin-Ben-Or (RABIN; BEN-OR, 1989). Em *multi-party computing* os participantes trocam informações sobre os dados a serem criptografados, e garantir a integridade destas informações trocadas é crucial. O esquema de verificação de informações

Rabin-Ben-Or aborda este desafio, permitindo aos participantes verificar a consistência das informações que recebem. O esquema Rabin-Ben-Or permite que um remetente envie informações a um destinatário para que o destinatário possa verificar sua consistência. Se o remetente tentar trapacear, o destinatário poderá detectar isso com alta probabilidade. Neste esquema um remetente compromete-se com a informação usando um esquema de compromisso criptográfico. Assim que uma informação é enviada a um destinatário, este verifica a consistência das informações recebidas em relação ao compromisso.

Assim como os demais métodos, este esquema usa interpolação polinomial. Se o remetente deseja enviar um valor v, ele escolhe um polinômio $f(x)$ de grau t tal que $f(0) = v$. O remetente envia $f(i)$ para o receptor, onde i é um ponto aleatório. O receptor verifica a consistência solicitando avaliações de $f(x)$ em diferentes pontos. A segurança depende da dificuldade de interpolar um polinômio de grau t com menos de $t + 1$ pontos. As tentativas de trapaça podem ser detectadas comparando as avaliações em diferentes pontos. Um exemplo do resultado final do compartilhamento PSS, disponível na biblioteca *Archistar*, é apresentado na listagem 4.

Listagem 4 – Exemplo de execução do esquema PSS da biblioteca

```
Segredo: "Este é um segredo."

Partes
[Parte 1]: fPsS8hmO6CWp4PKyJbPw/iFtOw
[Parte 2]: QoJDhX6eoNu5viKuclOZaJU/Yw
[Parte 3]: ewolEkfT4d5lM/BvMocb89A9dg
[Parte 4]: glvR0yxCrShJofLBLTFW52qCNQ
[Parte 5]: u9O3RBUP7C2VLCAAbeXUfC+AIA
```

4.2.3 Anonimização de dados

Outro tema estudado nesta fase foi a anonimização de dados. A anonimização de dados refere-se ao processo de proteção de informações privadas ou confidenciais, apagando ou criptografando identificadores que vinculam um indivíduo aos dados armazenados. À medida que as organizações dependem cada vez mais da análise de dados e do armazenamento digital, garantir a privacidade e a segurança dos dados pessoais tornou-se fundamental. O

anonimato ajuda as organizações a cumprir as leis de privacidade, reduzir o risco de violações de dados e manter a confiança dos seus usuários.

Para apropriação do tema, diferentes estudos foram lidos, porém uma referências principal foi utilizada: **The complete book of data anonymization: from planning to implementation** (RAGHUNATHAN, 2013). O livro investiga o domínio da anonimização de dados, com o objetivo de desmistificar o conceito e fornecer compreensões sobre sua implementação nas empresas. Apesar da crescente adoção de ferramentas de anonimização de dados, muitas organizações ainda enfrentam riscos associados ao uso indevido de dados confidenciais. Os desafios na implementação da anonimização de dados surgem de vários fatores, incluindo tecnologia, dados, processos e pessoas.

De acordo com o autor, a anonimização de dados é vista mais como uma arte do que como uma ciência por muitas empresas. A necessidade de anonimização de dados surge dos casos crescentes de vazamentos de dados, uso indevido de dados pessoais e dos custos comerciais associados. O sucesso das iniciativas de anonimização de dados muitas vezes fica aquém devido a desafios que vão desde a tecnologia até às pessoas.

O livro diferencia os desafios enfrentados pelos Diretor de Tecnologia da Informação (*CIO - "Chief Information Officer"*) e pelos profissionais da área de TI (por exemplo, arquitetos de dados). Embora os *CIOs* estejam preocupados com o sucesso geral e a execução do programa, os profissionais enfrentam desafios mais técnicos ou específicos da aplicação. O livro está estruturado em duas partes: A Parte I atende aos detalhes de interesse dos *CIOs*, discutindo a necessidade, o escopo, os desafios e os aspectos operacionais da anonimização de dados. A Parte II é adaptada para os desenvolvedores, detalhando padrões de solução, técnicas, abordagens de implementação, desafios e melhores práticas. O livro não tem como objetivo projetar ou desenvolver algoritmos de anonimização, mas serve como guia de referência para sua implementação.

Outra fonte de informação para o aprofundamento no assunto foi o livro **"Privacy-preserving data mining: models and algorithms"** (AGGARWAL; PHILIP, 2008). Neste livro, as principais técnicas de anonimização de dados foram estudadas, tais como:

1. **Mascaramento de dados**: Técnica que envolve mascarar os dados originais com conteúdo modificado (caracteres ou outros dados), mantendo o mesmo formato dos dados. Exemplos nas tabelas 3 e 4.

Tabela 3 – Dados antes do mascaramento

Nome	E-mail	CPF
João Silva	joao.silva@email.com	123.456.789-01
Maria Souza	maria.souza@email.com	456.123.789-23
Carlos Prado	carlos.prado@email.com	789.123.456-45

Tabela 4 – Dados após o mascaramento

Nome	E-mail	CPF
João Silva	xxx.xxx@email.com	123.XXX.XXX.01
Maria Souza	xxx.xxx@email.com	456.XXX.XXX-23
Carlos Prado	xxx.xxx@email.com	789.XXX.XXX-45

2. **Pseudoanonimização**: Nesta técnica se substitui identificadores privados por identificadores ou pseudônimos falsos. Isso permite que os dados sejam comparados com sua fonte sem revelar a fonte real. Exemplos nas tabelas 5 e 6.

Tabela 5 – Dados antes da pseudoanonimização

Nome	E-mail
João Silva	joao.silva@email.com
Maria Souza	maria.souza@email.com
Carlos Prado	carlos.prado@email.com

Tabela 6 – Dados após pseudoanonimização

Nome	E-mail pseudoanonimizado
João Silva	usuario123@email.com
Maria Souza	usuario456@email.com
Carlos Prado	usuario789@email.com

3. **Embaralhamento (*Shuffling*)**: Este método envolve a troca de dados entre registros, o que dificulta a compreensão dos dados originais, mas as propriedades estatísticas gerais do conjunto de dados permanecem inalteradas. Exemplos nas tabelas 7 e 8.

Tabela 7 – Dados antes do embaralhamento

Nome	Idade
João Silva	28
Maria Souza	34
Carlos Prado	45

Tabela 8 – Dados após embaralhamento

Nome	Idade
João Silva	45
Maria Souza	28
Carlos Prado	34

4. **Generalização**: Os atributos de dados são substituídos por categorias mais amplas. Exemplos nas tabelas 9 e 10.

Tabela 9 – Dados antes da generalização

Nome	Idade exata
João Silva	28
Maria Souza	34
Carlos Prado	45

Tabela 10 – Dados após generalização

Nome	Faixa etária
João Silva	20-29
Maria Souza	30-39
Carlos Prado	40-49

5. **Adição de ruído (*Noise Addition*)**: ruído aleatório é adicionado aos dados para evitar a identificação de indivíduos, mantendo a utilidade geral do conjunto de dados. Exemplos nas tabelas 11 e 12.

Tabela 11 – Dados antes da adição de ruído

Nome	Salário exato (em R$)
João Silva	3.250
Maria Souza	5.800
Carlos Prado	2.700

Tabela 12 – Dados após adição de ruído

Nome	Salário com ruído (em R$)
João Silva	3.237
Maria Souza	5.743
Carlos Prado	2.778

6. **k-anonimato (k-*anonymity*)**: k-*anonymity* é uma técnica projetada para proteger identidades individuais em conjuntos de dados. Em sua essência, o método garante que qualquer registro de dados único seja indistinguível de pelo menos $k - 1$ outros registros, tornando difícil identificar um indivíduo com base em seus atributos. Isto é atingido concentrando-se em *quasi-identifiers*, que são conjuntos de atributos que podem potencialmente, em combinação, identificar uma pessoa.

   Por exemplo, atributos como "CEP", "Data de nascimento" e "Sexo" podem, quando combinados, identificar exclusivamente um indivíduo em um conjunto de dados. No k-*anonymity* os portadores de dados (quem controla e mantém os dados de um sistema) geralmente empregam dois métodos principais: generalização e supressão. A generalização envolve a ampliação de valores de atributos específicos em categorias mais amplas. Por exemplo, substituir a idade exata "27" por um intervalo "20-29" torna mais difícil identificar um indivíduo com base na idade.

   Por outro lado, a supressão envolve a omissão de certos valores de atributos ou mesmo de registros inteiros. Isso pode ser feito para atributos que são muito exclusivos e podem facilmente levar à reidentificação. O k-*anonymity* não é infalível contra todas as ameaças externas. Os invasores com conhecimento prévio adicional ainda podem deduzir informações confidenciais, mesmo de um conjunto de dados que cumpra princípios do k-*anonymity*. Além disso, o processo para alcançar o k-*anonymity*, especialmente por meio de ampla generalização e supressão, pode diminuir a utilidade do conjunto de dados para pesquisas legítimas ou fins analíticos. Exemplos nas tabelas 13 e 14.

Tabela 13 – Dados antes do k-anonimato

Nome	Idade	Doença
João Silva	28	Gripe
Maria Souza	34	Diabetes
Carlos Prado	29	Gripe
Joana Machado	34	Diabetes
Mario Gomes	28	Resfriado

Tabela 14 – Dados após k-anonymity (com $k = 2$)

Nome	Faixa Etária	Doença
*	20-29	Gripe
*	30-39	Diabetes
*	20-29	Gripe
*	30-39	Diabetes
*	20-29	*

7. **Privacidade Diferencial (*Differential Privacy*)**: A Privacidade Diferencial (*Differential Privacy - DP*) oferece uma abordagem matematicamente fundamentada para proteger a privacidade individual em conjuntos de dados. O princípio fundamental da DP é que qualquer análise estatística não deve revelar se as informações de um indivíduo específico estão presentes no conjunto de dados.

   Para conseguir isso, uma quantidade controlada de ruído aleatório é introduzida nos resultados das consultas de dados. A magnitude e o tipo desse ruído são determinados por um parâmetro de privacidade, normalmente representado como ϵ (epsilon). O valor de ϵ serve como uma medida de perda de privacidade: um ϵ menor implica garantias de privacidade mais fortes, mas ao custo de introduzir mais ruído, o que pode reduzir a precisão ou a utilidade dos dados. A estratégia do DP reside na sua robustez.

   Mesmo que alguém tenha informações externas sobre indivíduos, o ruído adicionado pelo DP garante que não seja possível determinar de forma conclusiva a presença ou ausência de um indivíduo no conjunto de dados. No entanto, definir o ϵ correto é crucial. Muito ruído pode inutilizar os dados, enquanto pouco ruído pode comprometer a privacidade. Exemplos nas tabelas 15 e 16.

Tabela 15 – Dados antes da privacidade diferencial

Nome	Salário (em R$)
João Silva	3.250
Maria Souza	5.800
Carlos Prado	2.700

Tabela 16 – Dados após privacidade diferencial

Estatística	Valor (com ruído)
Salário médio	3.905

Etapa 2

5 Implementação, testes, revisão e avaliação das abordagens

Esta será a principal, extensa e complexa etapa do projeto. As técnicas descritas na literatura serão implementadas e aplicadas aos conjuntos de dados classificados anteriormente. Serão feitas diferentes implementações (abordagem tradicional, híbrida e otimizada). Cada uma destas implementações devem ser testadas e revisadas, para ao final serem avaliadas quanto a sua aplicação em sistemas em produção. O tempo para a implementação destas abordagens é extenso dada a complexidade imposta pela sua implementação (programação).

5.1 Metas

- Implementar algoritmo Shamir;
- Implementar algoritmo PVSS;
- Implementar de técnica descrita em outros trabalhos encontrados na literatura;
- Analisar a viabilidade da aplicação de técnicas de compartilhamento de segredos em dados pessoais e dados pertencentes às pessoas;
- Definir limites viáveis da aplicação das abordagens para ambientes reais (tempo aceitável em sistemas informatizados);
- Buscar ou criar modelos híbridos das técnicas;
- Implementar as abordagens;
- Analisar a viabilidade das abordagens híbridas;
- Otimizar algoritmos de compartilhamento de segredo com a criação de bases de dados de números primos;
- Implementar a abordagem;

- Analisar a viabilidade da abordagem otimizada quanto ao tempo de execução e segurança do sistema.

5.2 Resultados

Após a análise da literatura o projeto entrou na fase da prática, onde as abordagens foram implementadas ou refatoradas. Esta fase foi desenvolvida em dois momentos: 1) desenvolvimento, testes e avaliação dos mecanismos de compartilhamento de segredos; 2) desenvolvimento, testes e avaliação dos mecanismos de anonimização de dados. Para melhor organização deste relatório os detalhes destas metas serão descritas nas seções 6 e 7.

Os testes geraram uma vasta quantidade de resultados devido à variedade de configurações de computadores utilizados, tipos de testes, parâmetros, e repetições de cada operação para obter dados estatísticos precisos.

Neste relatório, focamos em uma seleção dessas observações para garantir clareza e concisão. Todos os dados e *scripts* em R (R Core Team, 2020) estão disponíveis em: <https://github.com/secstorproject/secstor-results> para que sejam usados em futuras análises ou comparações executadas por outros pesquisadores.

6 Compartilhamento de Segredos

Para o desenvolvimento das aplicações relacionadas ao tema compartilhamento de segredos foram utilizadas duas bibliotecas pré-existentes: *Archistar* (TECHNOLOGY, 2021), a qual implementa os trabalhos descritos em (SHAMIR, 1979), (KRAWCZYK, 1993), e duas alternativas de mecanismos de compartilhamento de segredo verificáveis (RABIN; BEN-OR, 1989); a segunda biblioteca, implementada por Bessani (NEVES, 2008)[1], a qual implementa o trabalho proposto em (SCHOENMAKERS, 1999).

6.1 Implementação da biblioteca e do *web service*

A primeira etapa do desenvolvimento ligado à esta pesquisa foi criar uma integração das bibliotecas citadas anteriormente. Para isso, uma nova biblioteca, chamada *Secstor Library* foi criada (SECSTOR, 2022d). Nesta biblioteca foi utilizado o padrão de projetos *Facade*.

O padrão *Facade* é um padrão estrutural que fornece uma interface simplificada para um subsistema mais complexo, neste caso, as bibliotecas que implementam os mecanismos de compartilhamento de segredos. Um dos principais benefícios do padrão *Facade* é que ele simplifica a interface de um sistema complexo. Os clientes não precisam se preocupar com o funcionamento interno do subsistema; eles interagem apenas com a *Facade*, que fornece uma API clara e direta. Se houverem alterações no subsistema, como alterações de métodos ou modificações de comportamento, elas poderão ser gerenciadas na *Facade* sem afetar os clientes existentes. Isso garante que o sistema permaneça flexível e adaptável a mudanças futuras.

Para a biblioteca, foram executados testes de desempenho, tanto para atestar seu correto funcionamento como para determinar uma linha base para o tempo de resposta dos algoritmos de compartilhamento de segredos. Os testes buscaram os tempos de geração das partes do segredo (definido por vezes como **split**) assim como o tempo de reconstrução dos segredos (definido por vezes como **reconstruct**).

Com a biblioteca devidamente desenvolvida e testada, foi possível avançar para a próxima etapa do projeto. Utilizando o *framework Spring*

[1] Foi introduzida uma pequena otimização: os números primos são pré-gerados e armazenados em um banco de dados. Quando necessário, um primo é selecionado aleatoriamente nesse banco de dados, reduzindo a sobrecarga computacional e melhorando o tempo de resposta.

(SPRING. . . , 2023), foi desenvolvido um *web service REST* chamado *Sector API* (SECSTOR, 2022a), (SECSTOR, 2022b). Este serviço aproveita a *Secstor Library* para oferecer aos usuários uma plataforma segura e eficiente. A intenção é permitir a utilização dos algoritmos de compartilhamento de segredos de maneira que possa ser integrada por sistemas e que seja independente da linguagem de programação utilizada.

Esta implementação de *web service* compreende duas versões. A primeira, que exige autenticação, obriga o usuário a passar por um processo de registro antes de acessar os recursos, uma abordagem alinhada às melhores práticas em design de *web services* para garantir segurança. Por outro lado, a segunda versão é mais aberta, sem requerer autenticação ou registro, sendo esta a versão escolhida para a realização de nossos testes, a fim de minimizar variáveis externas.

Nos testes do *web service*, investigamos o desempenho da biblioteca quando operando como um serviço disponibilizado através da rede. Os serviços de rede apresentam particularidades no tempo de resposta, incluindo a adição de uma camada de *software* para possibilitar a operação via rede (como a integração com *framework Spring*) e os tempos de resposta inerentes das redes (LANs e Internet). Embora esses fatores possam afetar o desempenho em comparação com o uso direto da biblioteca, eles oferecem uma visão mais realista do desempenho em cenários de serviços reais. É importante notar que a disponibilização de funcionalidades por meio de serviços é uma tendência crescente no desenvolvimento de *software* moderno.

Para tentar abarcar o maior número de possibilidades de uso da biblioteca e do *web service*, um plano de testes foi gerado, como apresentado na tabela 17.

Tabela 17 – Plano de testes em Compartilhamento de Segredos

n	k	**Algorithm**	**Threads**	**Dataset Size**
3	2	css	[1, 10, 100]	[1, 3, 5]
5	3	css	[1, 10, 100]	[1, 3, 5]
6	3	css	[1, 10, 100]	[1, 3, 5]
10	5	css	[1, 10, 100]	[1, 3, 5]
3	2	krawczyk	[1, 10, 100]	[1, 3, 5]
5	3	krawczyk	[1, 10, 100]	[1, 3, 5]
6	3	krawczyk	[1, 10, 100]	[1, 3, 5]
10	5	krawczyk	[1, 10, 100]	[1, 3, 5]
3	2	pss	[1, 10, 100]	[1, 3, 5]
5	3	pss	[1, 10, 100]	[1, 3, 5]
6	3	pss	[1, 10, 100]	[1, 3, 5]
10	5	pss	[1, 10, 100]	[1, 3, 5]
3	2	pvss	[1, 10, 100]	[1, 3, 5]
5	3	pvss	[1, 10, 100]	[1, 3, 5]
6	3	pvss	[1, 10, 100]	[1, 3, 5]
10	5	pvss	[1, 10, 100]	[1, 3, 5]
3	2	shamir	[1, 10, 100]	[1, 3, 5]
5	3	shamir	[1, 10, 100]	[1, 3, 5]
6	3	shamir	[1, 10, 100]	[1, 3, 5]
10	5	shamir	[1, 10, 100]	[1, 3, 5]
3	2	pss	[1, 10, 100]	[1, 3, 5]
5	3	pss	[1, 10, 100]	[1, 3, 5]
6	3	pss	[1, 10, 100]	[1, 3, 5]
10	5	pss	[1, 10, 100]	[1, 3, 5]

Para os testes da biblioteca e do *web service* temos diferentes particularidades, mas a descrição abaixo referente a tabela 17 abarca o plano de forma geral. As particularidades serão incluídas em cada um dos itens, quando houverem.

1. n **e** k **(Colunas 1 & 2):**

   - Estas colunas representam os valores de partes totais e partes necessárias para reconstrução do segredo, respectivamente (mais detalhes sobre este parâmetros podem ser encontrando na subseção 4.2.2).

2. **Algorithm (Coluna 3):**

- Esta coluna especifica o algoritmo utilizado no teste. O mesmo se repete para cada um dos parâmetros n e k utilizados.

3. **Threads (Coluna 4):**

   - Esta coluna indica o número de processamentos ocorrendo simultaneamente durante os testes. Na biblioteca, esse número refere-se à quantidade de *threads* que são iniciadas e operadas simultaneamente, realizando as operações de **split** e **reconstruct**. No contexto do *web service*, ele simboliza a quantidade de clientes acessando o serviço ao mesmo tempo, enviando como requisições os dados para executarem o **split** e recebendo como respostas as partes do segredo, e também enviado as partes do segredos para que a operação de **reconstruct** seja executada e retornando o dado original. Para otimizar o espaço na tabela, utilizamos uma notação em forma de *array*. Em termos práticos, esses números mostram que os testes foram conduzidos com 1, 10 e 100 *threads* em paralelo, respectivamente.

4. **Dataset Size (ds) (Coluna 5):**

   - Esta coluna indica os tamanhos dos conjuntos de dados utilizados nos testes, expressos em kbytes. Para redução no tamanho da tabela foi utilizado uma notação de *array*. De forma direta estes números representam que foram realizados testes usando *datasets* de $1kb$, $10kb$ e $100kb$, respectivamente.

No domínio da ciência da computação e da tecnologia da informação, o desempenho e o comportamento dos algoritmos podem variar significativamente com base no *hardware* subjacente. Reconhecendo isso, o plano de testes, quando da tomada de tempos de resposta da biblioteca, foi abrangente em diversas configurações. Cada configuração, ou "*testbench*", representa uma combinação única de componentes de hardware, incluindo diferentes CPUs, capacidades de RAM e contagens de núcleos.

O objetivo de testar nesta base diversificada de configurações é que, no mundo real, os usuários finais possuem uma ampla variedade de configurações de *hardware*. Ao testar diferentes configurações, podemos mostrar que os algoritmos de compartilhamento de segredo apresentam diferentes desempenhos em diferentes plataforma de *hardware*. Além disso, diferentes configurações podem revelar gargalos inesperados de desempenho. Por

exemplo, um algoritmo pode funcionar bem em uma CPU de ponta, mas ter dificuldades em uma CPU de nível intermediário. Na tabela 18, a abreviatura "C/T" significa "Cores/Threads" e "SO" representa "Sistema Operacional".

Tabela 18 – Configuração dos computadores usados nos testes.

Conf.	Tipo	CPU	Freq.	C/T	RAM	SO
C1	Notebook	Intel i7-3632q	3.2Ghz	4/8	16gb	Win
C2	Desktop	Intel i3-10100t	3.8Ghz	4/8	16gb	Win
C3	Servidor[2]	Intel e2378g	5.10Ghz	8/16	32gb	Lin
C4	Desktop	AMD R7 5700x	4.6Ghz	8/16	32gb	Win
C5	Desktop	AMD R7 5700x	4.6Ghz	8/16	32gb	Lin

Para avaliar a biblioteca sob uma outra perspectiva esta foi integrada a um *web service*. Esta abordagem proporcionou uma visão mais prática de sua performance e aplicabilidade em um ambiente real. O *web service*, por sua vez, foi consistentemente hospedado na configuração **C3** (tabela 18). A escolha por esta configuração levou em consideração que o equipamento apresenta-se como a plataforma mais propícia para a hospedagem do serviço, contemplando requisitos tanto de *hardware* robusto quanto de conectividade otimizada.

O servidor, conforme detalhado na configuração **C3**, está estrategicamente localizado em um *data center* no campus. Este *data center* é equipado com características avançadas, tais como resfriamento redundante, sistemas de alimentação ininterrupta, segurança física 24/7 e conexões de rede de alta capacidade (ainda que limitadas para não comprometer o tráfego de outros serviços do campus).

Durante o processo de teste, focamos em três cenários distintos:

Tabela 19 – Detalhes dos cenários de testes da API de Compartilhamento de Segredos

Cenário	Clientes	Link Cliente	Link Servidor
T1	1 Local	1 Gbps (Dedicado)	1 Gbps (Limitado)
T2	1 Remoto	400 Mbps (Internet)	300 Mbps (Limitado)
T3	20 Locais	1 Gbps (Compartilhado)	1 Gbps (Limitado)

O termo *Link Cliente* alude à conexão de saída do cliente. Quando mencionamos um link dedicado, estamos nos referindo a uma conexão que

liga diretamente o cliente a um conjunto de *switches* que operam à mesma capacidade da interface de rede do cliente. Já no contexto de *link* compartilhado, temos múltiplos clientes que utilizam a mesma rota de saída provida por um *switch* comum. Por outro lado, quando falamos do *link* do servidor como "limitado", isso indica que, devido às políticas de rede do campus (como *firewalls* ou outros), a capacidade total da conexão não está inteiramente disponível para os testes.

No cenário **T2**, encontramos desafios ao tentar avaliar os testes com 100 *threads* simultâneas, devido a duas limitações técnicas principais:

1. O cliente estava localizado atrás de um *CGNAT - Carrier-Grade Network Address Translation*). *CGNAT* é uma técnica usada para estender o endereçamento IP, permitindo que vários dispositivos em uma rede privada compartilhem um único endereço IP público. Nesse contexto, o cliente estava limitado no número de conexões TCP simultâneas permitidas pelo seu provedor de serviços. Esse tipo de restrição pode surgir quando o provedor tenta gerenciar e conservar os endereços IP públicos disponíveis em sua rede.

2. O servidor onde o *web service* é hospedado, por outro lado, estava localizado dentro da infraestrutura de rede administrada do campus. Todas as conexões entrantes ao servidor são filtradas por um *firewall* e, além disso, monitoradas por um sistema de *IDS - Intrusion Detection System*. Este sistema, ao detectar um grande volume de requisições simultâneas provenientes de um único endereço IP, interpretava essa atividade como potencialmente maliciosa, classificando o IP como suspeito e, consequentemente, bloqueando conexões subsequentes. Embora tentássemos adicionar o endereço IP de entrada (que identifica a aplicação cliente) à lista de IPs confiáveis, o provedor do cliente periodicamente alterava seu endereço IP público. Este comportamento resultou em uma tarefa contínua e trabalhosa de manter o IP atualizado como confiável na nossa configuração.

No cenário **T3**, utilizamos 20 máquinas distintas, representando 20 clientes diferentes, localizadas no laboratório didático do campus. É importante ressaltar que esse laboratório é um espaço compartilhado. Dado o extenso tempo que a execução completa do plano demandaria, optamos por executar um subconjunto específico do plano de testes. Especificamente, os testes

foram conduzidos com os parâmetros $n = 10$, $k = 5$, $ds = 10$, e $th = 100$, este último parâmetro para cada um dos clientes.

Para os dados de entrada foram utilizados *datasets* que simulam informações pessoais como entrada para os algoritmos. Esses *datasets* foram gerados com a ajuda do site Mockaroo (MOCKAROO, 2023) e estão no formato *JSON - JavaScript Object Notation*. A quantidade de texto em cada registro varia, conforme detalhado na coluna *Dataset Size* da tabela 17. A listagem 5 apresenta um exemplo de registro resumido. Para a variação entre os tamanhos de registros, mais chaves são adicionadas e mais valores são adicionadas as chaves, principalmente à aquelas que remetem aos dados pertencentes às pessoas.

Listagem 5 – Exemplo de registro

```
        {
            "nome": "Wain",
            "sobrenome": "Mitchard",
            "email": "wmitchard0@ft.com",
            "gender": "Male",
            "ip": "111.222.205.168",
            "endereco": "5 Sutteridge Trail",
            "cidade": "Hagerstown",
            "estado": "Maryland",
            "rg": "42398438861746",
            "cpf": "03789444565",
            "data_de_nascimento": "29/1/1974",
            "local_de_nascimento": "68 Monica Crossing",
            "telefone": "(78)1238-5857",
            "celular": "(90)7 8884-1280",
            "latitude": 39.5207,
            "longitude": -77.9162,
        ...
        "biometria": "rfmdZSuElHXlzJFxJG",
        "trajetoria_clinica": "isXEYqxxYqkabpXAHx",
        "historico_pagamentos": "cBryDuYSYEvewitHrl",
        }
```

Os algoritmos produzem saídas que correspondem ao cálculo das partes (**split**), conforme detalhado na subseção 4.2.2. Posteriormente, es-

sas saídas são armazenadas e empregadas como entradas no processo de reagrupamento das partes (**reconstruct**).

6.2 Resultados alcançados

Em nossos esforços para avaliar a eficiência e o desempenho dos vários algoritmos de compartilhamento de segredos, conduzimos uma série de testes utilizando a biblioteca (SECSTOR, 2022d). O objetivo principal desses testes foi medir o tempo necessário para dividir e reconstruir segredos usando diferentes algoritmos, conjuntos de dados (*datasets*) e contagens de *threads* simultâneas, assim como diferentes parâmetros de número de partes divididas e número de partes necessárias para a reconstrução do segredo.

Em cada teste, foram processados 100 registros distintos, lidos de arquivos, de ambas as operações (**split** e **reconstruct**). Estes foram repetidos 5 vezes: para cenários com uma única *thread*, resultando em 500 operações; e em testes de estresse com 100 *threads* simultâneas, totalizando 500.000 operações, com cada *thread* executando 5 rodadas de 10.000 operações. Uma amostra dos resultados salvos para cada parâmetro está descrita na listagem 6 para uma melhor compreensão. Os resultados completos podem ser analisados utilizando os dados encontrados em (SECSTOR, 2023a).

Repare que o nome do arquivo, descrito nas legendas das listagens 6 e 7 descrevem as informações sobre os resultados que o arquivo armazena, como a operação, o nome do algoritmo, parâmetros, e demais informações do plano de testes.

Os arquivos tem como informação o número da *thread* que gerou o resultado, o identificador do registro, e os tempos. Uma particularidade para os arquivos com resultados de **reconstruct** é o parâmetro *kused* que informa o número de chaves usadas para a reconstrução do segredo.

Para cada configuração de *hardware* descrita na tabela 18 foi criada uma pasta que armazena os arquivos com os testes executados.

Para assegurar a padronização e eficiência dos testes, foi desenvolvido um cliente dedicado aos testes da API (SECSTOR, 2022c). Este cliente foi especialmente implementado para conduzir as operações de **Split** e **Reconstruct**, assim como descrito acima, e registrar os tempos de resposta. Os registros de tempo são armazenados conforme descrito as listagens 6 e 7 acima. Esta consistência na forma de registro, adotada tanto para a biblioteca quanto para o *web service*, reforça a metodologia adotada e garante uma

Listagem 6 – Arquivo: `split_css_3-2_1kB_1thread_100objects.csv`

```
thread;rgid;instance1;instance2;instance3;instance4;instance5
Thread-1;1;20,842;2,777;2,819;2,071;0,773
Thread-1;2;0,673;0,650;0,689;0,695;0,803
Thread-1;3;1,036;0,683;0,666;0,655;0,694
```

Listagem 7 – Arquivo: `reconstruct_css_3-2_1kB_1thread_100objects`

```
thread;regid;kused;instance1;instance2;instance3;instance4;instance5
Thread-1;1;3;2,477;0,157;0,154;0,221;0,243
Thread-1;2;3;0,219;0,160;0,133;0,145;0,184
Thread-1;3;3;0,144;0,180;0,152;0,163;0,150
```

comparação direta e confiável entre os resultados.

6.2.1 Limpeza de dados e tratamento de valores discrepantes

Durante a análise de desempenho dos algoritmos de compartilhamento de segredo, é imperativo assegurar a consistência, precisão e autenticidade dos dados. Valores discrepantes, que se desviam consideravelmente das demais observações, podem comprometer nossas conclusões. Para preservar a integridade de nossa análise, utilizamos dois métodos reconhecidos: o *Intervalo Interquartil (IQR)* e o *Z-score*.

Método do Intervalo Interquartil (IQR): Este método é utilizado para identificar valores atípicos, ou *outliers*, que são pontos de dados que se desviam significativamente dos demais. A identificação é baseada no intervalo definido pelos primeiro e terceiro quartis dos dados. O intervalo interquartil (IQR) é multiplicado por um fator de 1,5 para ajustar o alcance. Qualquer valor que fique fora deste intervalo ajustado é considerado um *outlier*. Este método é particularmente eficaz para conjuntos de dados com distribuições assimétricas e é robusto contra valores extremos.

Método Z-Score: Este método avalia o quão distante um ponto de dado, potencialmente um *outlier*, está da média do conjunto, em termos de desvios padrão. Geralmente, pontos com um *Z-Score* maior que 2 ou menor que -2 são classificados como *outliers*. É importante notar que este método

é sensível a valores extremos e é mais eficaz quando os dados seguem uma distribuição normal.

Após a aplicação de ambos os métodos, constatamos semelhança nas propriedades estatísticas dos resultados. Optamos pelo *Z-Score* para nossa análise por sua capacidade de equilibrar a integridade dos dados com a retenção de informações valiosas, enquanto o IQR pode ser mais conservador em certos cenários.

6.2.2 Resumo de dados e análise estatística

Após a limpeza dos dados brutos, avançamos para a etapa de sumarização dos dados a fim de extrair propriedades estatísticas essenciais. No sumário extraímos os dados da listagem abaixo:

- **Tempo Médio** (*mean_time*): Indica o tempo médio gasto para todas as entradas de um grupo (média das instâncias de testes, para cada linha), proporcionando uma visão do desempenho típico.
- **Desvio Padrão** (*sd_time*): Avalia a dispersão dos tempos dentro de um grupo. Um valor menor sugere consistência próxima à média, enquanto um maior indica maior variação.
- **Mediana** (*median_time*): Representa o tempo central quando ordenado, sendo uma métrica robusta contra valores extremos.

Estas métricas, nomeadamente média, desvio padrão e mediana, fundamentam as análises e interpretações subsequentes neste relatório.

6.3 Operação de split (Biblioteca)

Ao avaliar o desempenho de algoritmos das operações é essencial considerar várias métricas. Nesta seção os resultados serão focados na operação **split**, ou seja, no processo inicial de transformação dos dados em partes (segredos).

É importante ressaltar que o desempenho dos algoritmos, representados pelo tempo necessário para o processamento de uma operação, não é apenas determinado pela média, mas também pelos extremos (melhor e pior casos) e pelo comportamento típico (mediana). Com mais de 1.000 cenários

únicos possíveis e executados em nossos testes, devido às diferentes combinações de parâmetros, uma visão resumida torna-se indispensável. A tabela 20 apresenta este resumo. Os tempos são expressos em milissegundos (ms).

Tabela 20 – Resumo das comparações entre algoritmos

Perf.	Alg.	Conf.	n	k	ds	th	Média	Median.	Desv.
Melhor	css	C4	3	2	1	100	0.02	0.02	0.00
Melhor	krawczyk	C4	3	2	1	100	0.02	0.02	0.00
Melhor	pss	C4	3	2	1	100	0.29	0.29	0.00
Melhor	pvss	C4	3	2	1	100	4.37	4.36	0.07
Melhor	shamir	C4	3	2	1	100	0.28	0.28	0.00
Mediana	css	C3	6	3	5	10	0.10	0.10	0.01
Mediana	krawczyk	C4	5	3	10	10	0.08	0.08	0.00
Mediana	pss	C2	10	5	3	100	1.87	1.87	0.07
Mediana	pvss	C2	5	3	1	10	11.39	11.30	0.37
Mediana	shamir	C4	10	5	5	10	1.44	1.43	0.02
Pior	css	C1	6	3	10	1	0.48	0.47	0.02
Pior	krawczyk	C1	6	3	10	1	0.30	0.30	0.02
Pior	pss	C1	10	5	10	1	7.38	7.36	0.09
Pior	pvss	C1	10	5	10	1	28.92	28.69	0.81
Pior	shamir	C1	10	5	10	1	5.30	5.24	0.29

- **Algoritmo CSS:** Se destaca em uma CPU Ryzen7 5700x (**C4**), alcançando um tempo médio de apenas $0,02$ ms. No entanto, em uma CPU i7-3624qm (**C1**), o tempo médio aumenta para $0,48$ ms. Em média, em uma CPU Xeon E2378G (**C3**), apresenta $0,10$ ms.
- **Algoritmo Krawczyk:** Mostra uma resposta rápida e uniforme em todas as configurações, oscilando entre $0,02$ ms e $0,30$ ms.
- **Algoritmo PSS:** Apresenta uma variabilidade maior, com tempos que vão de $0,29$ ms até $7,38$ ms em cenários mais desafiadores.
- **Algoritmo PVSS:** Devido à sua complexidade inerente, os tempos médios variam, começando em $4,37$ ms.

A tabela 21 apresenta uma análise comparativa do desempenho da operação de **split** em diferentes configurações.

Tabela 21 – Resumo das comparações entre configurações

Conf.	Média	Mediana	Min.	Max.	Desvio
C1	4.95	1.19	0.05	28.92	7.75
C2	3.62	0.94	0.04	21.48	5.53
C3	3.12	1.07	0.04	16.19	4.29
C4	2.01	0.65	0.02	10.75	2.84

- **Configuração C1:** Esta configuração exibiu o tempo médio mais longo, cerca de $4,95$ ms. O tempo de execução variou de $0,054$ ms a $28,92$ ms. O desvio padrão de $7,75$ ms reflete a variabilidade no desempenho.
- **Configuração C2:** Com um tempo médio de $3,62$ ms, esta configuração representa um equilíbrio. Os tempos variaram entre $0,038$ ms e $21,48$ ms, e um desvio padrão de $5,53$ ms indica variabilidade moderada.
- **Configuração C3:** Esta configuração teve um tempo médio de $3,12$ ms, indicando desempenho ligeiramente mais eficiente. O tempo de execução oscilou entre $0,038$ ms e $16,19$ ms, com desvio padrão de $4,29$ ms.
- **Configuração C4:** A mais eficiente das configurações, apresentou um tempo médio de apenas $2,01$ ms. Os tempos variaram de $0,02$ ms a $10,75$ ms, com um desvio padrão notavelmente baixo de $2,84$ ms, indicando consistência.

O próximo resumo apresentado compara os diversos algoritmos de compartilhamento de segredos para a configuração **C3**. Consideramos uma ampla gama de condições, determinadas pelos parâmetros n, k, ds e th. Dada a extensa variedade de combinações possíveis, nos focamos em destacar os cenários de melhor, mediano e pior desempenho para cada algoritmo.

Note que o “melhor” desempenho refere-se ao menor tempo médio de execução, enquanto o “pior” desempenho denota o maior tempo médio registrado. O desempenho mediano, por sua vez, oferece uma visão equilibrada, indicando um resultado típico que pode ser esperado. Na tabela 22, destacamos os cenários de melhor, mediano e pior desempenho:

Tabela 22 – Resumo dos algoritmos na configuração C3

Perf.	Alg.	n	k	ds	th	Méd.	Medi.	Desv.
Melhor	css	3	2	1	100	0.04	0.04	0.00
Melhor	krawczyk	3	2	1	100	0.04	0.04	0.00
Melhor	pss	3	2	1	100	0.49	0.49	0.01
Melhor	pvss	3	2	1	100	6.33	6.32	0.08
Melhor	shamir	3	2	1	100	0.46	0.46	0.00
Mediana	css	5	3	3	1	0.10	0.09	0.03
Mediana	krawczyk	3	2	5	100	0.08	0.08	0.00
Mediana	pss	10	5	3	1	2.16	2.20	0.06
Mediana	pvss	5	3	10	1	9.84	9.78	0.21
Mediana	shamir	10	5	3	10	1.43	1.46	0.06
Pior	css	10	5	1	1	0.43	0.39	0.23
Pior	krawczyk	5	3	10	1	0.16	0.16	0.00
Pior	pss	10	5	10	1	6.85	7.02	0.23
Pior	pvss	10	5	10	1	16.20	16.30	0.48
Pior	shamir	10	5	10	100	4.67	4.83	0.25

- **Algoritmo CSS**:
  - *Melhor Desempenho*: Com os parâmetros $n = 3$, $k = 2$, $ds = 1$ e $th = 100$, registramos um notável tempo médio de $0,04$ ms.
  - *Desempenho Mediano*: Nas condições $n = 5$, $k = 3$, $ds = 3$ e $th = 1$, o tempo médio foi de $0,10$ ms.
  - *Pior Desempenho*: Configurando $n = 10$, $k = 5$, $ds = 1$ e $th = 1$, o tempo médio aumentou para $0,43$ ms.
- **Algoritmo Krawczyk**:
  - *Melhor*: Atinge um tempo médio de $0,04$ ms com os parâmetros $n = 3$, $k = 2$, $ds = 1$ e $th = 100$.
  - *Mediana*: Para $n = 3$, $k = 2$, $ds = 5$ e $th = 100$, o tempo médio foi de $0,08$ ms.
  - *Pior*: Observou-se um tempo médio de $0,16$ ms com $n = 5$, $k = 3$, $ds = 10$ e $th = 1$.

Dos dados analisados, extraímos as seguintes observações:

- O algoritmo PVSS, mesmo em seu melhor cenário, tende a apresentar tempos de execução mais longos quando comparado a algoritmos como CSS ou Krawczyk.
- Os parâmetros, em especial ds e th, têm grande influência nos resultados. Com o aumento de ds (tamanho do *dataset*), geralmente observa-se um crescimento nos tempos de execução.

Em resumo, a configuração **C3** mostrou um desempenho consistente e sólido em diferentes cenários. Ainda assim, a seleção adequada de algoritmos e seus respectivos parâmetros é essencial. Estas descobertas ressaltam a importância de selecionar algoritmos e parâmetros alinhados às demandas específicas de cada aplicação e critérios de desempenho.

6.3.1 Análise da influência das *threads* simultâneas no desempenho

Uma observação interessante foi encontrada ao analisar os resultados obtidos:

- Contrariamente à expectativa comum, o tempo médio de execução diminui ligeiramente com o aumento do número do número de *threads* simultâneas.
- A variação no tempo médio entre diferentes números de *threads* é sutil.
- O desvio padrão sugere uma variabilidade considerável nos tempos de resposta, independentemente da contagem de *threads*.

A intuição geral pode nos levar a pensar que um aumento no número de *threads* poderia introduzir mais contenção e, consequentemente, retardar o desempenho. No entanto, neste conjunto de dados, o aumento de *threads* levou a uma ligeira melhoria no desempenho. Esta tendência positiva pode ser atribuída a um gerenciamento de *threads* eficaz ou ao fato das tarefas serem intrinsecamente paralelizáveis, sem incorrer em sobrecargas significativas ou fazer uso de estruturas de dados compartilhadas.

Importante ressaltar que as diferenças de desempenho entre as configurações de 1, 10 e 100 *threads* são marginais. Isso sugere que o número de *threads* não é o principal determinante do desempenho neste contexto, e outros fatores ou potenciais gargalos podem estar influenciando o resultado.

6.4 Operação de reconstruct (Biblioteca)

Nesta seção, os resultados serão focados na operação **reconstruct**, ou seja, no processo de recomposição do segredo (dados) a partir das partes compartilhadas.

Assim como na operação de **split** temos muitos cenários possíveis que foram testados e aqui serão apresentados somente resumos. Os resultados completos e os *scripts* para a análise destes podem ser acessados no link <https://github.com/secstorproject/secstor-results>

A tabela 23 apresenta um resumo geral dos tempos da operação de **reconstruct**. Os tempos são expressos em milissegundos (ms).

Tabela 23 – Resumo das comparações entre algoritmos

Perf.	Alg.	Conf.	n	k	ku	ds	th	Média	Mdn.	Desv.
Melhor	css	C4	3	2	2	1	100	0.01	0	0.01
Melhor	krawczyk	C4	3	2	3	1	100	0.01	0	0.01
Melhor	pss	C4	3	2	2	1	100	0.02	0	0.02
Melhor	pvss	C4	3	2	3	1	100	0.07	0	0.07
Melhor	shamir	C4	3	2	3	1	100	0.01	0	0.01
Mediana	css	C4	3	2	3	10	10	0.12	0	0.12
Mediana	krawczyk	C1	3	2	3	3	100	0.06	0	0.06
Mediana	pss	C2	6	3	3	5	10	0.2	0.01	0.2
Mediana	pvss	C3	10	5	5	3	1	0.35	0.02	0.35
Mediana	shamir	C4	3	2	2	10	10	0.09	0.02	0.08
Pior	css	C1	10	5	10	10	1	1.73	0.04	1.71
Pior	krawczyk	C1	10	5	10	10	1	0.23	0.01	0.23
Pior	pss	C1	10	5	10	10	10	2.86	0.07	2.83
Pior	pvss	C1	10	5	10	10	1	1.14	0.04	1.13
Pior	shamir	C1	10	5	5	10	1	0.67	0.01	0.68

- **Algoritmo CSS:** Apresenta seu melhor desempenho na configuração **C4**, alcançando um tempo médio de apenas $0,01$ ms. No entanto, na configuração **C1**, seu pior desempenho, o tempo médio aumenta consideravelmente para $1,73$ ms. Em um cenário médio, usando a mesma configuração Ryzen7 5700x (**C4**) atinge $0,12$ ms.
- **Algoritmo Krawczyk:** Em sua melhor forma, na configuração **C4**, atinge um tempo médio de $0,01$ ms. Em contrapartida, na configuração **C1**, seu

pior desempenho, alcança $0,23$ ms. No cenário intermediário, usando uma configuração i7-3624qm (**C1**), registra $0,06$ ms.

- **Algoritmo PSS:** Seu melhor desempenho é observado na configuração **C4** com $0,02$ ms. Por outro lado, na configuração **C1**, atinge seu tempo máximo de $2,86$ ms. Em um cenário médio, numa configuração i3-10100t (**C2**), apresenta $0,2$ ms.
- **Algoritmo PVSS:** Devido à sua natureza, este algoritmo apresenta tempos médios mais elevados. Na configuração **C4**, tem seu melhor tempo de $0,07$ ms. No entanto, na configuração **C1**, seu tempo máximo é de $1,14$ ms. Em um cenário médio, na configuração **C2**, registra $0,34$ ms.
- **Algoritmo Shamir:** Na configuração **C4**, mostra um tempo médio impressionante de $0,01$ ms. Mas, na configuração **C1**, tem seu tempo mais alto de $0,67$ ms. Em uma situação intermediária, usando a configuração Ryzen7 5700x (**C4**), atinge $0,09$ ms.

A tabela 24 um resumo geral dos tempos da operação de **reconstruct** diante das várias configurações de computadores que foram testadas (as configurações estão descritas na tabela 18).

Tabela 24 – Resumo das comparações entre configurações

Conf.	Média	Mediana	Mín.	Máx.	Desvio
C1	0.36	0.21	0.021	2.86	0.37
C2	0.23	0.15	0.02	2.16	0.25
C3	0.24	0.14	0.02	2.56	0.30
C4	0.14	0.09	0.009	1.57	0.18

- **Configuração C1:** Esta configuração exibiu o tempo médio mais longo, cerca de $0,36$ ms. O tempo de execução variou de $0,021$ ms a $2,86$ ms. O desvio padrão de $0,37$ ms reflete a variabilidade no desempenho.
- **Configuração C2:** Com um tempo médio de $0,23$ ms, esta configuração representa um equilíbrio. Os tempos variaram entre $0,15$ ms e $2,16$ ms, e um desvio padrão de $0,25$ ms indica variabilidade moderada.
- **Configuração C3:** Esta configuração teve um tempo médio de $0,24$ ms, indicando desempenho ligeiramente menos eficiente. O tempo de

execução oscilou entre $0,02$ ms e $2,56$ ms, com desvio padrão de $0,30$ ms.

- **Configuração C4:** A mais eficiente das configurações, apresentou um tempo médio de apenas $0,14$ ms. Os tempos variaram de $0,009$ ms a $1,57$ ms, com um desvio padrão notavelmente baixo de $0,18$ ms, indicando consistência.

O próximo resumo apresentado compara os diversos algoritmos de compartilhamento de segredos para a configuração **C3**. Consideramos uma ampla gama de condições, determinadas pelos parâmetros n, k, $kused$, ds e th. Dada a extensa variedade de combinações possíveis, nos focamos em destacar os cenários de melhor, mediano e pior desempenho para cada algoritmo.

Note que o "melhor" desempenho refere-se ao menor tempo médio de execução, enquanto o "pior" desempenho denota o maior tempo médio registrado. O desempenho mediano, por sua vez, oferece uma visão equilibrada, indicando um resultado típico que pode ser esperado. Na tabela 25, destacamos os cenários de melhor, mediano e pior desempenho:

Tabela 25 – Resumo dos algoritmos na configuração C3

Perf.	**Alg.**	n	k	$kused$	ds	th	**Méd.**	**Medi.**	**Desv.**
Melhor	css	3	2	1	100	0.0207	0.0207	0.021	0.0006
Melhor	krawczyk	3	2	1	100	0.0172	0.0172	0.017	0.0005
Melhor	pss	3	2	1	100	0.0256	0.0256	0.026	0.0007
Mediana	css	5	3	5	100	0.0969	0.0969	0.097	0.0011
Mediana	krawczyk	10	5	3	1	0.0514	0.0514	0.0515	0.0060
Mediana	pss	5	3	3	10	0.237	0.237	0.236	0.0026
Pior	css	10	5	10	1	0.377	0.377	0.376	0.0232
Pior	krawczyk	5	3	10	10	0.161	0.161	0.161	0.0016
Pior	pss	10	5	10	10	2.56	2.56	2.54	0.115
Melhor	pvss	3	2	1	100	0.124	0.124	0.122	0.0046
Mediana	pvss	3	2	5	10	0.395	0.395	0.393	0.0051
Pior	pvss	6	3	10	10	0.978	0.978	0.975	0.0122
Melhor	shamir	3	2	1	100	0.0152	0.0152	0.015	0.0005
Mediana	shamir	6	3	3	1	0.0862	0.0862	0.068	0.0214
Pior	shamir	10	5	10	1	0.452	0.452	0.451	0.0630

- **Algoritmo CSS**:

  - *Melhor Desempenho*: Com os parâmetros $n = 3$, $k = 2$, $kused = 1$, $ds = 100$ e $th = 0.0207$, registramos um notável tempo médio de $0,0207$ ms.
  - *Desempenho Mediano*: Nas condições $n = 5$, $k = 3$, $kused = 5$, $ds = 100$ e $th = 0.0969$, o tempo médio foi de $0,0969$ ms.
  - *Pior Desempenho*: Configurando $n = 10$, $k = 5$, $kused = 10$, $ds = 1$ e $th = 0.377$, o tempo médio aumentou para $0,377$ ms.

- **Algoritmo Krawczyk**:
  - *Melhor Desempenho*: Atinge um tempo médio de $0,0172$ ms com os parâmetros $n = 3$, $k = 2$, $kused = 1$, $ds = 100$ e $th = 0,0172$.
  - *Desempenho Mediano*: Para $n = 10$, $k = 5$, $kused = 3$, $ds = 1$ e $th = 0,0514$, o tempo médio foi de $0,0514$ ms.
  - *Pior Desempenho*: Observou-se um tempo médio de 0.161 ms com $n = 5$, $k = 3$, $kused = 10$, $ds = 10$ e $th = 0,161$.

- **Algoritmo PSS**:
  - *Melhor Desempenho*: Com os parâmetros $n = 3$, $k = 2$, $kused = 1$, $ds = 100$ e $th = 0.0256$, o tempo médio foi de $0,0256$ ms.
  - *Desempenho Mediano*: Configurando $n = 5$, $k = 3$, $kused = 3$, $ds = 10$ e $th = 0.237$, o tempo médio observado foi de $0,237$ ms.
  - *Pior Desempenho*: Com $n = 10$, $k = 5$, $kused = 10$, $ds = 10$ e $th = 2.56$, o tempo médio aumentou para $2,56$ ms.

- **Algoritmo PVSS**:
  - *Melhor Desempenho*: Com os parâmetros $n = 3$, $k = 2$, $kused = 1$, $ds = 100$ e $th = 0.124$, o tempo médio foi de $0,124$ ms.
  - *Desempenho Mediano*: Configurando $n = 3$, $k = 2$, $kused = 5$, $ds = 10$ e $th = 0.395$, o tempo médio observado foi de $0,395$ ms.
  - *Pior Desempenho*: Com $n = 6$, $k = 3$, $kused = 10$, $ds = 10$ e $th = 0.978$, o tempo médio aumentou para $0,978$ ms.

- **Algoritmo Shamir**:
  - *Melhor Desempenho*: Com os parâmetros $n = 3$, $k = 2$, $kused = 1$, $ds = 100$ e $th = 0.0152$, o tempo médio foi de $0,0152$ ms.

- *Desempenho Mediano*: Configurando $n = 6$, $k = 3$, $kused = 3$, $ds = 1$ e $th = 0.0862$, o tempo médio observado foi de $0,0862$ ms.
- *Pior Desempenho*: Com $n = 10$, $k = 5$, $kused = 10$, $ds = 1$ e $th = 0.452$, o tempo médio aumentou para $0,452$ ms.

6.5 Operação de split (web service)

Para a operação de *Split*, a requisição acontece utilizando o protocolo HTTP com o método POST. O *payload* é formatado em JSON e enviada a uma rota específica, contendo os dados dos *datasets* apresentados na listagem 5. Os parâmetros essenciais, incluindo o tipo de algoritmo e os valores de n e k, são incluídos no corpo da requisição. Como resposta, o serviço retorna as partes geradas em uma estrutura JSON, semelhante a descrita também na seção 4.2.2. É importante mencionar que os tempos apontados referem-se à latência das respostas das requisições (em milisegundos).

O primeiro resumo, descrito pela tabela 26, apresenta os tempos de resposta da API quando da comparação entre os diferentes algoritmos.

Tabela 26 – Resumo das comparações entre algoritmos usando API

Perf.	Alg.	Conf.	n	k	ds	th	Média	Median.	Desv.
Melhor	css	T1	10	5	1	1	2.83	2.68	0.56
Melhor	krawczyk	T1	6	3	1	1	2.51	2.62	0.51
Melhor	pss	T1	10	5	1	1	5.6	5.49	0.74
Melhor	pvss	T1	10	5	1	1	13.5	10.7	5.41
Melhor	shamir	T1	6	3	1	1	3.74	3.32	0.92
Mediana	css	T1	6	3	10	10	4.8	4.64	1.04
Mediana	krawczyk	T1	5	3	10	100	4.78	4.72	0.84
Mediana	pss	T1	6	3	10	100	23.9	24.6	4.89
Mediana	pvss	T1	6	3	1	100	31.2	32.5	4.82
Mediana	shamir	T1	3	2	10	100	21.6	22.2	4.06
Pior	css	T2	5	3	10	1	35.8	35.7	1.01
Pior	krawczyk	T2	6	3	10	10	35.4	35.3	1.21
Pior	pss	T2	3	2	10	10	64.4	65.8	10.8
Pior	pvss	T2	3	2	10	10	64.9	65.1	2.43
Pior	shamir	T2	3	2	10	10	61.1	64.6	10.2

- **Algoritmo CSS (API):** No ambiente **T1**, este algoritmo registrou um

tempo médio de $2,83$ ms. Contudo, ao se enfrentar circunstâncias mais adversas na configuração **T2**, esse tempo subiu significativamente para $35,8$ ms.

- **Algoritmo Krawczyk (API):** De forma consistente, o algoritmo Krawczyk demonstrou eficiência na configuração **T1** com $2,51$ ms. A configuração com conexão através da internet **T2** causou um aumento para $35,4$ ms.
- **Algoritmo PSS (API):** Este algoritmo apresentou uma variação acentuada em seus tempos de resposta. Enquanto no ambiente local **T1** obteve $5,6$ ms, em condições menos ideais **T2**, registrou $64,4$ ms.
- **Algoritmo PVSS (API):** O PVSS, conhecido por sua complexidade, variou seus tempos de $13,5$ ms em ambientes ideais **T1** para $64,9$ ms sob condições desafiadoras **T2**.
- **Algoritmo Shamir (API):** O algoritmo Shamir produziu tempos que variaram de $3,74$ ms em condições locais **T1** para $61,1$ ms através da internet em **T2**.

A tabela 27 apresenta uma análise comparativa do desempenho da operação de **split** em diferentes configurações.

Tabela 27 – Resumo das comparações entre configurações utilizando API

Config.	Média	Mediana	Min.	Max.	Desvio
T1	12.7	7.70	2.51	34.7	10.1
T2	42.4	36.2	29.2	64.9	12.6
T3	31.0	20.6	11.6	58.4	22.7

- **Configuração T2:** Esta configuração mostrou um tempo médio de execução de $42,4$ ms. A variação dos tempos de resposta foi significativa, indo de $29,2$ ms até $64,9$ ms, com um desvio padrão de $12,6$ ms, que reflete uma certa dispersão nos tempos de resposta. Lembrando que nesta configuração não foram feitos testes com 100 *threads* simultâneas.
- **Configuração T3:** Com um tempo médio de $31,0$ ms, esta configuração, executada em 20 PCs em uma rede local, demonstrou um desempenho intermediário entre as três configurações. Os tempos oscilaram de $11,6$ ms a $58,4$ ms. O desvio padrão alto de $22,7$ ms sugere uma variabilidade significativa nos tempos de resposta. Lembrando que está configuração somente executou uma das configurações do plano de testes.

- **Configuração T1:** Este foi o cenário com melhor desempenho dentre os três, com um tempo médio notável de apenas $12,7$ ms. Os tempos variaram de $2,51$ ms, o menor dentre todos os cenários, a $34,7$ ms. O desvio padrão de $10,1$ ms indica uma consistência razoável no desempenho desta configuração.

A tabela 28 apresenta uma análise comparativa do desempenho da operação de **split** nas configurações **T2** e **T3**, para os parâmetros parâmetros $n = 10$, $k = 5$, $ds = 10$, e $th = 100$.

Tabela 28 – Resumo das comparações entre configurações T2 e T3

Conf.	Média	Mediana	Min.	Max.	Desvio
T3	31.0	20.6	11.6	58.4	22.7
T1	17.4	22.3	4.62	30.2	11.9

- A configuração **T3**, com 20 PCs (e presumivelmente mais clientes simultâneos), teve um tempo médio de execução de $31,0$ ms.
- A configuração T1, usando apenas um computador (e presumivelmente menos carga simultânea), teve um tempo médio de $17,4$ ms.
- Considerando a carga adicional na configuração **T3**, o aumento do tempo médio de execução não foi tão pronunciado quanto poderia ser esperado. Isso sugere que a infraestrutura da configuração **T3** é robusta e pode lidar eficientemente com cargas maiores, apesar do aumento na latência.

6.6 Operação de reconstruct (web service)

De maneira similar a operação de **split**, na operação *reconstruct*, as partes juntamente com outras informações essenciais dos algoritmos são enviadas usando POST e JSON, e o sistema retorna o dado original como resposta (o mesmo registro do *dataset*).

A tabela 29 apresenta uma análise dos tempos para a operação **reconstruct**. Os tempos estão expressos em milissegundos (ms).

Tabela 29 – Resumo das comparações entre algoritmos em T1 e T2

Perf.	Alg.	Conf.	n	k	ku	ds	th	Média	Mdn.	Desv.
Melhor	css	T1	5	3	5	1	1	2.83	2.76	0.4
Melhor	krawczyk	T1	5	3	5	1	1	3.29	3.37	0.77
Melhor	pss	T1	3	2	3	5	1	3.6	3.58	0.17
Melhor	pvss	T1	10	5	5	1	1	3.6	3.52	0.27
Melhor	shamir	T1	3	2	2	5	1	2.91	2.88	0.13
Mediana	css	T1	5	3	3	5	100	5.13	5.12	0.7
Mediana	krawczyk	T1	3	2	2	5	100	5.05	5.02	0.65
Mediana	pss	T1	5	3	5	10	1	6.93	6.94	0.32
Mediana	pvss	T1	5	3	5	1	10	5.57	5.46	0.59
Mediana	shamir	T1	6	3	3	5	10	5.95	6.05	0.46
Pior	css	T2	6	3	3	10	1	77.7	77.5	3.09
Pior	krawczyk	T2	3	2	3	10	1	77.6	77.3	3.06
Pior	pss	T2	5	3	3	10	1	103	103	3.63
Pior	pvss	T2	6	3	3	10	1	75.4	75.1	3.28
Pior	shamir	T2	6	3	3	10	1	103	103	3.82

A análise dos resultados mostra:

- **Algoritmo CSS:** Atua melhor na configuração **T1**, alcançando um tempo médio de $2,83$ ms. No cenário de pior desempenho, na configuração **T2**, o tempo médio sobe para $77,7$ ms, e em um cenário mediano, na configuração **T1**, atinge $5,13$ ms.
- **Algoritmo Krawczyk:** Seu melhor desempenho é na configuração **T1**, com $3,29$ ms. Seu pior desempenho ocorre em **T2**, alcançando $77,6$ ms, e na situação mediana, atinge $5,05$ ms.
- **Algoritmo PSS:** Registra seu melhor tempo na configuração **T1** com $3,6$ ms. No entanto, seu pior desempenho, na configuração **T2**, chega a 103 ms, e seu desempenho mediano é de $6,93$ ms.
- **Algoritmo PVSS:** Mesmo sendo um algoritmo mais complexo, seu melhor desempenho é de $3,6$ ms em **T1**. Já seu pior tempo é $75,4$ ms em **T2**, e mediano é $5,57$ ms.
- **Algoritmo Shamir:** Este algoritmo tem um desempenho notável de $2,91$ ms em sua melhor configuração, **T1**. No entanto, na configuração **T2**, o tempo sobe para 103 ms, e o desempenho mediano é de $5,95$ ms.

A tabela 30 oferece um resumo geral dos tempos da operação de **reconstruct** nas diferentes configurações de computadores testadas (as configurações estão detalhadas na tabela 18).

Tabela 30 – Resumo das comparações entre configurações

Conf.	Média	Mediana	Mín.	Máx.	Desvio
T2	51.5	52.1	27.4	103	18.0
T3	9.11	7.08	6.11	13.5	3.42
T1	4.97	4.76	2.83	14.6	1.15

Com base na análise:

- **Configuração T2:** Apresentou o tempo médio mais longo de $51,5$ ms. A execução variou entre $27,4$ ms e 103 ms. O desvio padrão, estando em 18 ms, mostra uma significativa variabilidade no desempenho.
- **Configuração T3:** Esta configuração apresentou tempo médio de $9,11$ ms. A execução variou entre $7,08$ ms e $13,5$ ms. O desvio padrão, estando em $3,42$ ms, demonstrando consistência no desempenho.
- **Configuração T2:** A mais eficaz dentre as três, com um tempo médio de $4,97$ ms. A execução variou de $2,83$ ms a $14,6$ ms, com um desvio padrão relativamente baixo de $1,15$ ms, demonstrando consistência no desempenho.

A tabela 31 apresenta uma análise comparativa do desempenho da operação de **split** nas configurações **T1** e **T3**, para os parâmetros $n = 10$, $k = 5$, $ds = 10$, e $th = 100$.

Tabela 31 – Resumo das comparações entre configurações T1 e T3

Config.	Média	Mediana	Min.	Max.	Desvio
T3	31.0	20.6	11.6	58.4	22.7
T1	17.4	22.3	4.62	30.2	11.9

- **Configuração T3:** Executada em 20 PCs e com mais clientes simultâneos, teve um tempo médio de execução de $31,0$ ms. Apesar da maior carga, a latência adicional não foi tão alta quanto poderia ser esperada.

- **Configuração T1:** Usando apenas um computador e com presumivelmente menos carga simultânea, teve um tempo médio de $17,4$ ms. Embora essa configuração apresente um desempenho superior em termos de tempo médio, é importante considerar que ela também enfrenta uma carga significativamente menor em comparação com a configuração **T3**.

6.7 Comparação entre biblioteca e API

A avaliação e comparação entre a biblioteca e a API oferece uma visão essencial dos resultados. Essas comparações, que constituem parte deste estudo, proporcionam avaliações não apenas para os desenvolvedores e arquitetos de *software*, mas também para todos os leitores que podem considerar a adoção de uma das opções implementadas nestas pesquisa.

Assim como nas outras análises apresentadas, nesta seção forneceremos resumos e comparações de resultados selecionados, normalmente enfocando os melhores e piores cenários previamente identificados entre os resultados obtidos. Todos os demais resultados estão disponíveis para análises mais detalhadas.

Ao comparar a API com a biblioteca, restringimos os resultados à configuração **C3**, uma vez que o *web service* está implantado no mesmo servidor, proporcionando uma base de comparação relevante e significativa. Essas alterações buscam tornar o texto mais claro e expressivo, enfatizando a importância das comparações e esclarecendo o motivo da seleção específica da configuração **C3** para análise. Chamaremos os resultados filtrados da biblioteca de *base line (bl)*.

Na tabela 32 analisaremos o resumo geral dos tempos da *base line* com a configuração **T1** de uso da API.

Tabela 32 – Sumário comparativo entre biblioteca (B) e API (A)

Est.	Split A	Recon. A	Split B	Recon. B	Split dif.	Recon. dif.
Min.	2.51	2.83	0.04	0.02	2.47	2.81
1st Qu.	3.95	4.32	0.10	0.05	3.85	4.28
Mediana	7.70	4.76	1.07	0.08	6.63	4.68
Média	12.70	4.97	3.12	0.12	9.58	4.84
3rd Qu.	22.38	5.26	4.60	0.16	17.78	5.10
Max.	34.68	14.59	16.19	0.79	18.49	13.79

O tempo médio da API para operações de **split** ainda que pequeno (dentro da ordem dos milisegundos) é relativamente maior que o da biblioteca. Essa discrepância pode ser atribuída às sobrecargas inerentes de uma API, como latência de rede, tratamento de solicitações e potencial processamento no servidor.

Quanto a operação de **reconstruct** espera-se que o tempo médio da API seja maior que o da biblioteca. Embora a diferença possa ser menos pronunciada do que na **split** (dado que a reconstrução pode envolver menos passos computacionais), os *overheads* gerais associadas à API ainda contribuem para tempos de processamento mais longos.

6.7.1 Gráficos comparativos entre a biblioteca e API

Para proporcionar uma visualização mais clara, esta seção ilustra comparações entre a biblioteca e a API, utilizando os mesmos casos discutidos anteriormente porém utilizando gráficos. Os gráficos a seguir oferecem uma perspectiva alternativa dos dados e enfatizam algumas comparações específicas.

Cada gráfico possui dois títulos: o primeiro descreve o conteúdo do gráfico e o segundo detalha os parâmetros utilizados para sua criação. Com base em uma análise abrangente, destacamos os parâmetros associados aos melhores e piores desempenhos, referindo-se, respectivamente, aos menores e maiores tempos de resposta.

Realizamos comparações entre o tempo de resposta da biblioteca, referida como *base line (bl)*, e as configurações de testes na API. Essas comparações foram feitas de forma isolada, levando em consideração os resultados e suas características específicas.

A figura 1 ilustra a relação entre o tempo de resposta da biblioteca e as configurações **T1** e **T2** para a operação **split** no melhor caso. É evidente o impacto significativo no tempo de resposta ao se utilizar as operações via API. Adicionalmente, nota-se um acréscimo expressivo no tempo quando a API é acessada através da Internet (T2).

Já a figura 2 ilustra a operação **reconstruct** no melhor caso, onde as mesmas características do gráfico anterior puderam ser observadas, com um enfoque ainda mais claro na relação entre a execução da biblioteca e da API.

Quando comparadas as duas operações para estes casos é possível perceber que as operações de **split** tem um tempo relativamente maior que

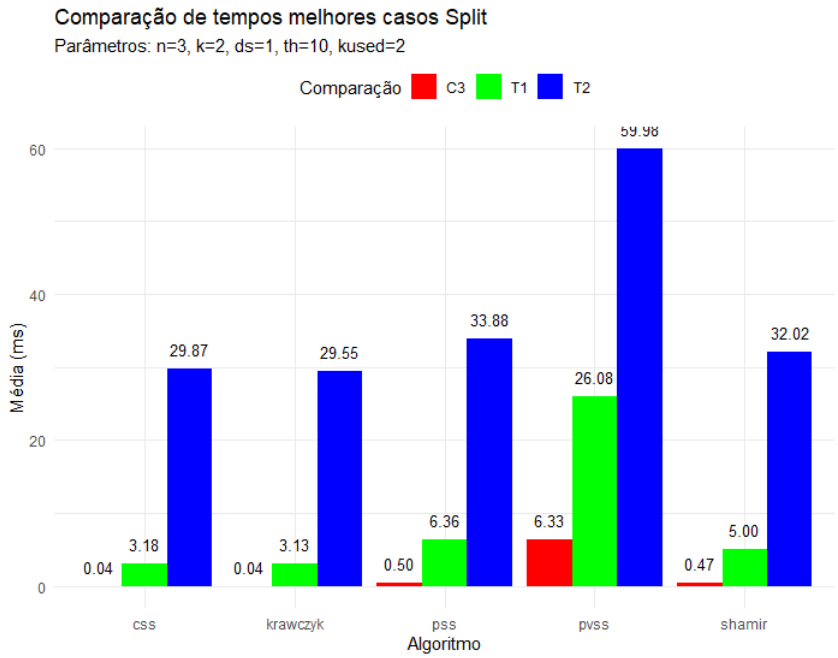

Figura 1 – Comparação de melhores casos Split

as operações de **reconstruct** na execução através da biblioteca e em alguns casos da API.

A figura 3 mostra a comparação entre o tempo de resposta da biblioteca e as configurações **T1** e **T2** para a operação **split** no cenário de pior caso. Embora o impacto da API no tempo de resposta seja notório, ao contrastar com a figura 1, observamos que os tempos finais são muito próximos em alguns casos. Isso sugere que o *overhead* e latências da API dominam o tempo de resposta total.

Já a figura 4 ilustra a operação **reconstruct** no pior caso e são observados os mesmos comportamentos.

As figuras 5 e 6 ilustram a relação entre o tempo de resposta da biblioteca e as configurações **T1** e **T3** para as operações de **split** e **reconstruct** no caso único testado em **T3**.

Neste caso já existem outras observações importantes. Para alguns algoritmos, a contenção causada pelo aumento de clientes, ainda que em rede local, é mais prejudicial no tempo de resposta. Como exemplos os algoritmos **pss** e **shamir**, para as operações de **split**

Para a operação de **reconstruct** o efeito observado anteriormente não

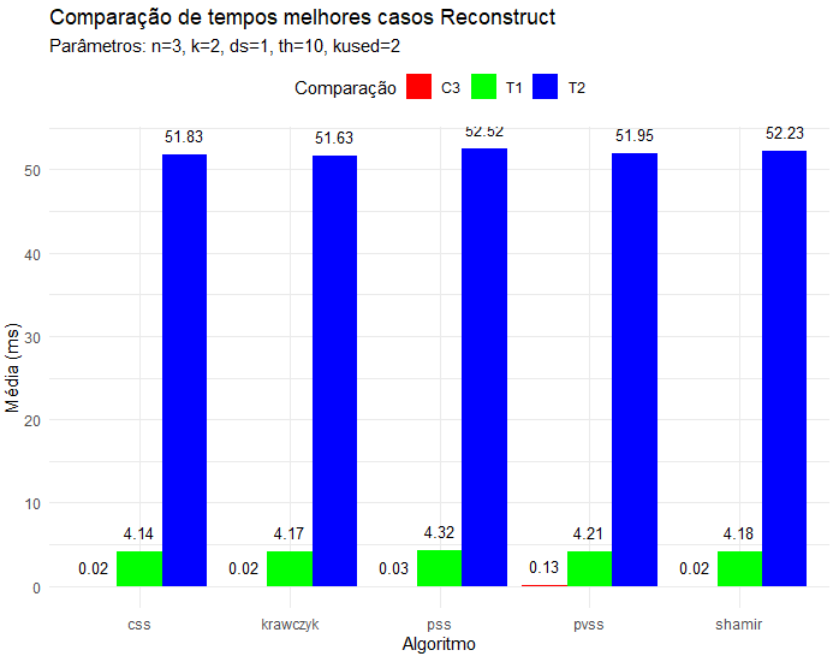

Figura 2 – Comparação de melhores casos Reconstruct

ocorre.

6.7.2 Observações gerais sobre o tempo de resposta

Em ambientes tecnológicos modernos, a eficiência e a rapidez de comunicação tornaram-se imperativos, especialmente quando se trata da interação entre sistemas. As APIs desempenham um papel central nesse cenário, funcionando como pontes de comunicação entre diferentes sistemas

Quando falamos de interação sistema-usuário, a percepção de desempenho é frequentemente influenciada pela velocidade com que as informações são retornadas. Por outro lado, na comunicação sistema-sistema, a eficiência é medida em termos absolutos. Em ambientes de microserviços, por exemplo, um serviço pode depender de múltiplas chamadas a outras APIs para completar uma única tarefa. Se cada chamada levar mais tempo do que o esperado, o efeito acumulado pode resultar em atrasos significativos. Além disso, tempos de resposta inconsistentes ou imprevisíveis podem complicar a orquestração, tornando o sistema global menos robusto e mais difícil de gerir.

A utilização de algoritmos de compartilhamento de segredos é uma prática robusta em segurança da informação, particularmente na proteção de

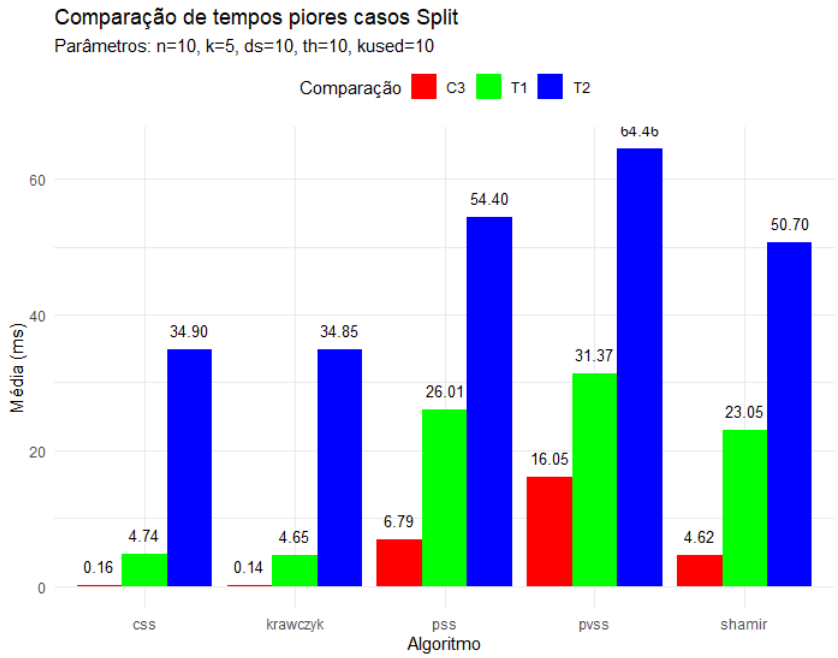

Figura 3 – Comparação de piores casos Split

dados sensíveis e na implementação de sistemas resilientes, principalmente no contexto dos sistemas atuais, onde as informações estão sendo cada vez mais armazenadas em ambientes de computação em nuvem. No entanto, como com qualquer algoritmo criptográfico, o desempenho é uma consideração importante, especialmente quando esses algoritmos são expostos como serviços por meio de APIs.

Um dos objetivos deste projeto é avaliar esses algoritmos. Embora a intenção inicial fosse uma análise geral dos algoritmos, o foco evoluiu naturalmente para a avaliação deles por meio de uma API.

Determinar tempos de resposta aceitáveis para APIs é desafiador devido à variedade de usos e à falta de padrões universais. O contexto em que a API é usada influencia as expectativas, e com a evolução tecnológica, o que era considerado rápido no passado pode ser visto como lento hoje. Além disso, informações sobre desempenho são frequentemente fragmentadas e influenciadas por considerações comerciais.

Em muitos contextos de aplicações modernas, um tempo de resposta de 30 ms para uma API, como os obtidos em nossos testes, é notavelmente eficiente. De acordo com estudos realizados por organizações como a *Nielsen*

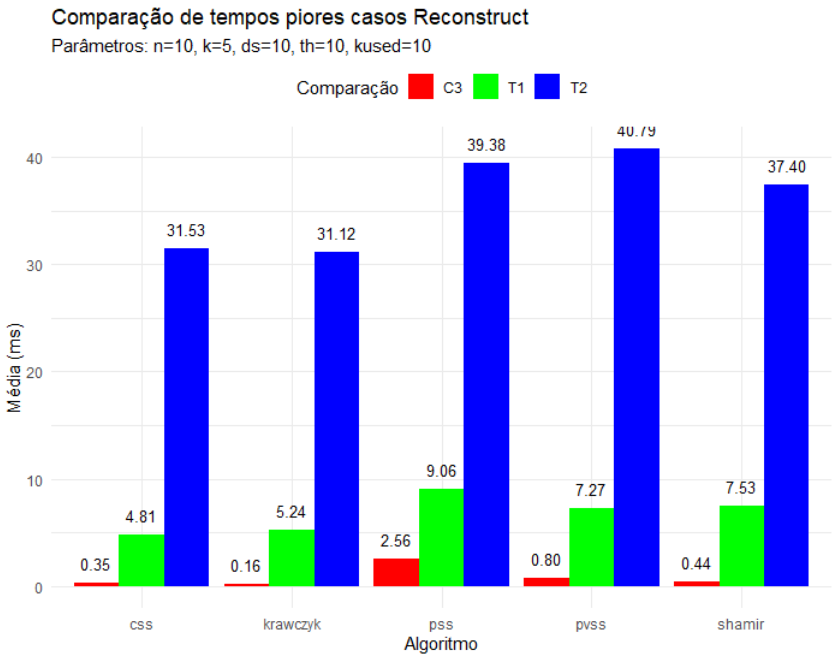

Figura 4 – Comparação de piores casos Reconstruct

Norman Group, um tempo de resposta inferior a 100 ms é percebido pelo usuário como instantâneo (GROUP, 2010). Assim, ao operar na casa dos 30 ms, a API não apenas atende, mas excede as expectativas comuns de desempenho, oferecendo uma experiência quase imediata para os usuários ou sistemas que a consomem. Embora este trabalho já tenha mais de 13 anos, é o mais utilizado como base para estudos, principalmente nas discussões no mais famoso fórum sobre programação, o *StackOverFlow* (OVERFLOW, 2022).

Diversas pesquisas em UX (estudos referentes à experiência de um usuário em um sistema) e desempenho de sistemas têm ressaltado a importância de tempos de resposta rápidos. De acordo com um relatório da Google, a probabilidade de abandono aumenta à medida que o tempo de carregamento se estende além de 3 segundos (GOOGLE, 2022). Mesmo que essa métrica seja frequentemente citada no contexto de carregamento de páginas da web, ela ilustra a impaciência intrínseca dos usuários e sistemas em ambientes digitais. Operando a 30 ms, a API está bem posicionada para evitar tais problemas e para maximizar a satisfação e eficiência.

No entanto, é crucial reconhecer que diferentes condições de teste podem influenciar os tempos de resposta. Testes sob cargas de trabalho mais intensas ou sob condições de internet variáveis podem resultar em tempos

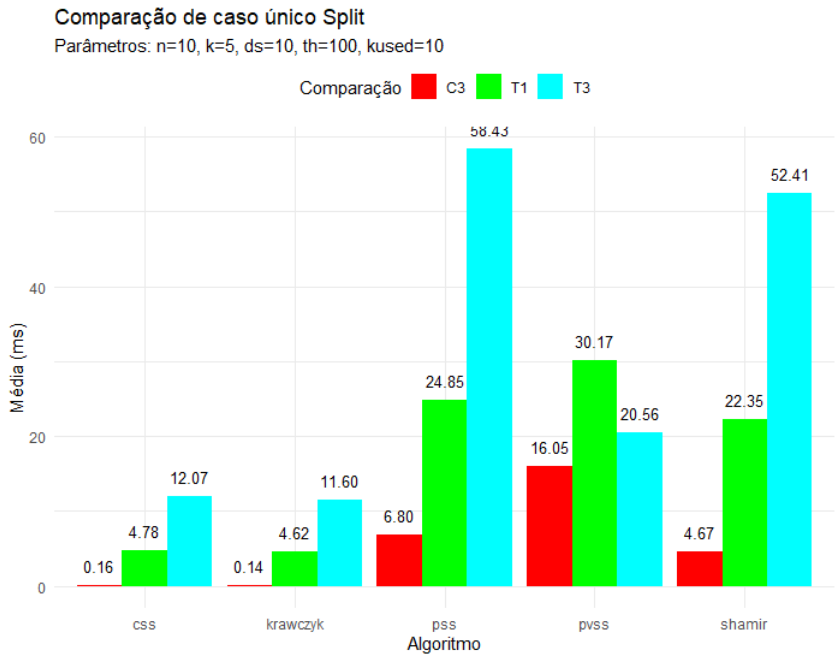

Figura 5 – Comparação caso único Split

de resposta mais longos. Assim como em qualquer estudo ou avaliação, é desafiador abranger todas as possíveis variações e cenários. Novas pesquisas e testes são essenciais para continuar refinando nosso entendimento e para identificar áreas de melhoria.

6.8 Análise do uso de CPU do Servidor Dell R250

Graças ao apoio fundamental e ao fomento fornecido pela Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC), foi possível a aquisição de um servidor de alta performance, especificamente um Dell PowerEdge R250.

Este servidor é amplamente reconhecido por sua eficiência e adequação a pequenas e médias empresas, sendo um equipamento do tipo rack que se destaca pela sua adaptabilidade e potência. A ideia principal foi testar a implementação em um servidor acessível a empresas que tenham interesse em implantar estes mecanismos de segurança em suas soluções tecnológicas, mas que não fossem impeditivas ou necessitassem de um recurso computacional com curso muito elevado.

Configuração do Servidor

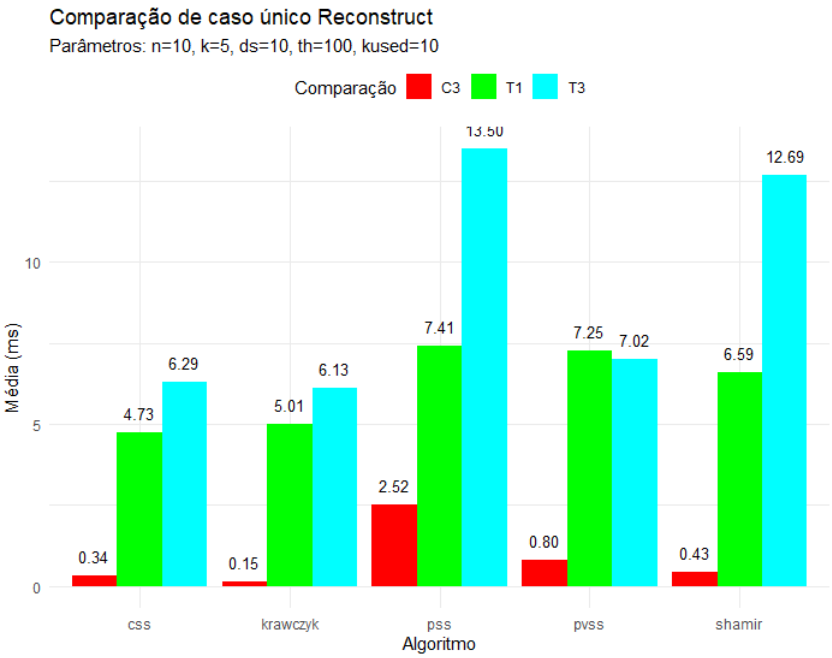

Figura 6 – Comparação caso único Reconstruct

- **Modelo do Servidor:** Dell PowerEdge R250
- **Processador:** Intel Xeon-E2378G
- **Memória RAM:** 32 GB

6.8.1 Metodologia dos Testes

Cenário de Teste: Foram executadas múltiplas instâncias simultâneas de operações de compartilhamento de segredos, tanto no teste da biblioteca quanto do *web service*, cujos resultados já foram apresentados nas seções anteriores.

Monitoramento de Recursos: Uso da CPU, distribuição de carga pelos núcleos, consumo de memória, e I/O.

Resultados Observados

- **Uso da CPU:** Notavelmente, em todos os testes, a utilização da CPU se manteve abaixo de 50% em todos os núcleos. Esta eficiência indica uma gestão de carga de trabalho é relativamente baixa para o poder computacional do servidor.

- **Distribuição de Carga:** A carga de trabalho pareceu foi parcialmente distribuída entre os núcleos do processador, evitando picos de uso em núcleos individuais, o que sugere uma paralelização razoável das tarefas.
- **Memória e I/O:** A memória RAM e as operações de I/O não foram gargalos, indicando que a configuração de 32 GB foi mais do que suficiente para as tarefas executadas e que a velocidade de I/O está em harmonia com as demandas de processamento.

7 Anonimização de Dados

Após a implementação bem-sucedida da *Secstor Library* e do *web service REST* detalhados na seção anterior, a pesquisa avançou para a segunda etapa: a anonimização de dados. Enquanto a *Secstor Library* e o *Web Service REST* anterior oferecem uma plataforma segura para o compartilhamento de segredos, a introdução de métodos de anonimização, chamado de '*de-dd*[3] apresenta outro método para a garantia de segurança no tratamento de dados, garantindo que os dados pessoais dos usuários sejam protegidos, mesmo em cenários de potenciais vazamentos de dados ou ataques maliciosos.

Assim como já descrito os seção 6.1, o objetivo inicial foi a avaliação das abordagens através de códigos locais (através de uma biblioteca), porém para atender as demandas e arquiteturas dos sistemas atuais, as bibliotecas foram transformadas também em um *web service*.

A seguir, revisamos os mecanismos que foram desenvolvidos, acompanhados de exemplos de sua aplicação:

- **Generalização**: A generalização consiste em substituir dados específicos por informações mais gerais ou por categorias mais amplas. Este processo diminui o nível de detalhes presentes no conjunto de dados. Por exemplo, um campo que informa a idade "27 anos" pode ser generalizado para "entre 20-30 anos".

- **Hashing:** Nesta técnica, a informação é convertida em um *hash*, transformando os valores originais, sejam eles números ou textos, em uma sequência de tamanho fixo. É frequentemente usado para armazenar senhas, mas também pode ser aplicado em outros conjuntos de dados para transmissão e validação subsequente.

- **Mascaramento:** O mascaramento oculta partes específicas dos dados, como informações pessoais, números de cartões de crédito ou outros dados sensíveis. Pode ser total, onde toda a informação é oculta, ou parcial, onde parte da informação original é mantida e a outra ocultada. É usado em situações em que certos dados precisam ser protegidos ao serem revelados publicamente, como por exemplo, os dígitos de um CPF em um documento.

[3] DE-IDentification

- **Null-out:** Este método substitui os dados por valores nulos ou vazios. É útil quando certos campos não são necessários ou contêm informações sensíveis que não devem ser compartilhadas.

- **Perturbação:** A perturbação adiciona uma pequena variação aleatória aos dados. Em dados numéricos, isso pode envolver a adição ou subtração de pequenos valores; para datas, pode significar adicionar ou subtrair dias. É comumente usado para manter propriedades estatísticas sem revelar informações específicas.

- **Pseudoanonimização:** A pseudoanonimização substitui identificadores reais por pseudônimos ou identificadores falsos. Isso permite que os dados sejam relacionados entre si sem expor as informações originais. É frequentemente aplicado em contextos onde a identificação precisa ser ocultada, mas a relação entre os dados deve ser mantida para análises subsequentes.

- **Troca ou Permutação (*Swaping*):** A técnica de troca envolve a permutação de valores entre registros. Por exemplo, as idades em um conjunto de dados podem ser reorganizadas de modo que não se alinhem mais aos registros originais, mantendo, contudo, a distribuição geral das idades. Esse método auxilia na anonimização de dados, garantindo que os valores sejam corretos, mas não se relacionem diretamente aos registros originais, preservando as características estatísticas do conjunto.

Na primeira etapa deste segmento da pesquisa, semelhante ao descrito na seção 6.1, objetivou-se desenvolver uma biblioteca para implementar os mecanismos de anonimização de dados anteriormente mencionados. Como resultado, foi criada a biblioteca **de-id (anonymizer)** (SECSTOR, 2023b).

Para garantir a eficácia da biblioteca, realizaram-se testes de corretude e desempenho. Estes avaliaram tanto sua funcionalidade quanto os tempos de execução. Os testes utilizaram os mesmos *datasets* descritos na seção 6.1 e apresentados na listagem 5. Os dados de saída, após o processo de anonimização, variaram conforme diferentes configurações, determinadas por parâmetros ou *payloads*, dependendo do contexto.

Após a validação da biblioteca, o próximo passo foi desenvolver um *web service*. Com o auxílio do *framework Django* (FOUNDATION, 2022), um *web service REST* foi construído e chamado de *de-id* (SECSTOR, 2023c).

Este serviço incorpora a biblioteca *anonimyzer*, proporcionando uma plataforma segura e eficaz para os usuários. O objetivo é oferecer um recurso de anonimização de dados que possa ser integrado por diferentes sistemas, independentemente da linguagem de programação.

O *web service* adota duas abordagens: síncrona e assíncrona:

Abordagem Síncrona: Neste modo, o cliente envia uma solicitação e aguarda pela resposta do servidor. O servidor processa os dados de maneira sequencial, retornando apenas quando concluído. Embora esta abordagem possa levar a tempos de espera mais longos, especialmente com grandes volumes de dados, sua natureza sequencial pode simplificar a implementação e gestão.

Abordagem Assíncrona: Nesta abordagem, o cliente pode prosseguir com outras tarefas após enviar uma solicitação, sem esperar pelo término do processamento. O servidor, por sua vez, responde assim que a tarefa é concluída. Isso pode resultar em uma resposta mais rápida ao usuário, mesmo que o processamento continue em segundo plano. Tal abordagem tende a ser mais escalável e responsiva, especialmente sob alta demanda.

Semelhante ao *web service* para compartilhamento de segredos, é necessária autenticação para acessar este serviço.

Durante os testes deste *web service*, examinou-se o desempenho da biblioteca em um ambiente de rede. Os serviços de rede possuem peculiaridades que podem influenciar os tempos de resposta, como a integração com o *framework Django* e os próprios tempos de resposta das redes (LANs e Internet). Apesar destes fatores poderem alterar o desempenho em relação ao uso direto da biblioteca, eles proporcionam uma visão mais realista em cenários práticos. Notou-se que a integração com o *framework* apresentou um desempenho mais promissor em comparação aos testes de estresse da biblioteca, onde a contenção de *threads* foi um gargalo significativo nos tempos, diferente do que ocorreu na implementação das técnicas de compartilhamento de segredo. Na tentativa de verificar esta observação, algumas hipóteses foram levantas: mal gerenciamento da aplicação para avaliar a biblioteca, o gerenciamento de *threads* em Python pode não ser tão eficiente para o caso. Ao final das discussões focamos nos resultados da API, pois para a oferta do serviço este é o caminho mais adequado no desenvolvimento de sistemas atuais.

Para abordar amplamente as potencialidades de uso da biblioteca e do

web service, elaboramos um plano de testes, conforme detalhado na tabela 33. Nesta fase, decidimos simplificar o plano, dado que o anterior produziu um volume elevado de resultados, muitos dos quais não foram integralmente incorporados no relatório devido a restrições de espaço. Apesar de termos reduzido a quantidade de testes, ainda obtivemos um volume considerável de dados. Embora apenas uma parcela desses resultados tenha sido analisada detalhadamente e apresentada neste relatório, todos estão disponíveis para consulta e análise no repositório referenciado (SECSTOR, 2023a).

Tabela 33 – Plano de testes em anomimização de dados

Parâm./*Payload*	**Threads**	**Dataset Size (em kb)**	**Requisição**
1	5	[1, 3, 5, 10]	[NA, *Sync*, *Assync*]
2	5	[1, 3, 5, 10]	[NA, *Sync*, *Assync*]
3	5	[1, 3, 5, 10]	[NA, *Sync*, *Assync*]

Para os testes da biblioteca e do *web service* temos diferentes particularidades, mas a descrição abaixo referente a tabela 33 abarca o plano de forma geral. As particularidades serão incluídas em cada um dos itens, quando houverem.

1. **Parâmetro/*Payload* (Coluna 1):**
    - Esta coluna determina o parâmetro usado na biblioteca ou o *payload* para o *web service*. Cada opção estabelece o mecanismo a ser adotado, os campos a serem anonimizados e o respectivo tipo de anonimização, além de seus parâmetros específicos.
2. **Threads (Coluna 2):**
    - Esta coluna representa o número de processamentos executados simultaneamente durante os testes. Embora fosse possível variar amplamente esse número, o que resultaria em diferentes desempenhos, optamos por usar apenas 5 *threads* simultâneas. A decisão pelo número 5 foi tomada considerando que alguns servidores começam a recusar um alto volume de conexões simultâneas, o que é identificado como *flooding*[4].

[4] *Flooding* é uma técnica de ataque cibernético que sobrecarrega um sistema com tráfego ou solicitações excessivas

3. **Dataset Size (Coluna 3):**

    - Esta coluna indica os tamanhos dos conjuntos de dados utilizados nos testes. Para redução no tamanho da tabela foi utilizado uma notação de *array*. De forma direta estes números representam que foram realizados testes usando datasets de $1kb$, $3kb$, $5kb$ e $10kb$, respectivamente.

4. **Requisição (Coluna 4):**

    - Esta coluna indica o tipo de requisição. Para os testes na biblioteca este parâmetro não é utilizado, pois as execuções ocorrer diretamente no dispositivo. Já para os testes no *web service* estas podem ser síncronas ou assíncronas, como já descrito anteriormente.

Da mesma forma que nos testes sobre a biblioteca e *web service* de compartilhamento de segredos, diferentes configurações de computadores foram utilizadas. Aqui limitamos as configurações de execução da biblioteca somente 3 configurações, porém, para a execução do *web service* os cenários foram diversificados.

Tabela 34 – Configuração dos computadores usados nos testes.

Conf.	Tipo	CPU	Freq.	C/T	RAM	SO
C1	Desktop	Intel i3-10100t	3.8Ghz	4/8	16gb	Win
C2	Desktop	AMD R7 5700x	4.6Ghz	8/16	32gb	Win
C3	Servidor[5]	Intel e2378g	5.10Ghz	8/16	32gb	Lin
Amazon EC2	Servidor	2.5 GHz	2	1gb	Lin	

Os cenários abaixo descrevem, principalmente, o ambiente de execução do *web service*. Para os testes descritos no cenário **T1** e **T2**, o serviço foi executado no servidor descrito na configurações **C3**. Para os demais foi utilizado o serviço de nuvem da *Amazon*, onde o *web service* foi executado em uma instância do tipo *t3*. O servidor, conforme detalhado na configuração **C3**, está estrategicamente localizado em um *data center* no campus.

Durante o processo de teste, focamos em três cenários distintos:

Tabela 35 – Detalhes dos cenários de testes da API

Cenário	Clientes	Link Cliente	Link Servidor
T1	1 Local	1 Gbps (Dedicado)	1 Gbps (Limitado)
T2	1 Remoto	400 Mbps (Internet)	300 Mbps (Limitado)
T3	1 Remoto	400 Mbps (Internet)	Variável (Amazon)
T4	1 Remoto	100 Mbps compartilhada (Wifi)	Variável (Amazon)

O termo *Link Cliente* alude à conexão de saída do cliente. Quando mencionamos um *link* dedicado, estamos nos referindo a uma conexão que liga diretamente o cliente a um conjunto de *switches* que operam à mesma capacidade da interface de rede do cliente. Já no contexto de *link* compartilhado, temos múltiplos clientes que utilizam a mesma rota de saída provida por um *switch* comum ou uma conexão *Wifi*. Por outro lado, quando falamos do *link* do servidor como "limitado", isso indica que, devido às políticas de rede do campus (como *firewalls*), a capacidade total da conexão não está inteiramente disponível para os testes.

No cenário **T2**, **T3** e **T4** encontramos as mesmas dificuldades relacionadas a *CGNat* já descritas neste relatório. Os demais detalhes dos resultados dos testes, tal como os *datasets* de entrada e a limpeza estatística pós obtenção dos resultados são os mesmos apresentados na tabela 5, seção 6.2 e 6.2.1.

O número de instâncias de testes também foram os mesmos descritos no início da subseção 6.2. A exceção foi somente no formato dos arquivos contendo os tempos de respostas, uma vez que o número de parâmetros para o testes dos mecanismos de anonimização são distintos dos algoritmos de compartilhamento de segredos. As listagens 8 (biblioteca) e 9 (API). Os nomes dos arquivos descrevem os parâmetros utilizados e o conteúdo dos arquivos os tempos obtidos para cada teste.

Listagem 8 – Arquivo: `anonimized_data_5_threads_1kB_100objects_dataset parameters_1`

```
1 thread;registro;tempo1;tempo2;tempo3;tempo4;tempo5
2 thread-1;1;22,090;54,268;109,922;127,805;135,864
3 thread-3;1;86,323;90,183;118,779;92,086;110,804
4 thread-4;1;99,287;118,154;115,595;94,316;108,830
```

Listagem 9 – Arquivo: anonimized_data_5_threads_1kB_100objects_dataset parameters_1

```
thread;registro;tempo1;tempo2;tempo3;tempo4;tempo5
thread-1;1;13,516;10,587;37,805;16,155;20,970
thread-2;1;15,922;25,626;21,232;28,557;18,207
thread-3;1;14,541;36,472;15,123;23,899;22,825
```

7.1 Resultados da biblioteca (*Anonymizer)*

Nesta seção os resultados da biblioteca nas diferentes configurações e parâmetros são descritos. Como já destacado anteriormente, é importante ressaltar que o desempenho dos algoritmos, representados pelo tempo necessário para o processamento de uma operação, não é apenas determinado pela média, mas também pelos extremos (melhor e pior casos) e pelo comportamento típico (mediana).

A tabela 36 apresenta um resumo geral dos tempos de execução dos mecanismos, sem separar as configurações ou tamanhos de *datasets*. Os tempos são expressos em milissegundos (ms).

Tabela 36 – Resumo das comparações entre configurações

Perf.	th	ds	Param.	Conf.	Req.	Méd	Median.	Dev.
Melhor	5	1	1	C2	NA	53.7	53.2	9.81
Melhor	5	3	2	C2	NA	100	97.4	35.2
Melhor	5	10	3	C2	NA	88.1	88.9	23.6
Mediana	5	10	1	C1	NA	113	111	17.4
Mediana	5	1	2	C1	NA	199	179	89.1
Mediana	5	3	3	C1	NA	190	188	26.6
Pior	5	5	1	C3	NA	137	133	52.5
Pior	5	10	2	C3	NA	439	435	105
Pior	5	1	3	C3	NA	228	222	73.7

- **Configuração C2 (Melhor):** Esta configuração, utilizando uma CPU Ryzen7 5700, apresentou tempos médios de execução variando de $53,7$ ms a 100 ms. A variabilidade foi notável, especialmente quando comparada ao desvio padrão, que chegou a $35,2$ ms em uma das execuções.
- **Configuração C1 (Mediana):** Utilizando uma CPU i3-10100t, os tempos médios de execução variaram de 113 ms a 199 ms. A execução com

um *dataset* de tamanho 1 e parâmetro 2 apresentou a maior variabilidade, com um desvio padrão de $89,1$ ms.

- **Configuração C3 (Pior):** Com uma CPU Xeon E2378g, os tempos médios foram mais elevados, variando de 137 ms a 439 ms. O desvio padrão também foi mais alto, indicando uma maior variabilidade nos tempos de execução.

A tabela 37 apresenta um resumo geral dos tempos das execuções diante das várias configurações de computadores que foram testadas.

Tabela 37 – Resumo das comparações entre configurações

Conf.	Média	Mediana	Mín.	Máx.	Desvio
C3	264	225	133	439	130
C1	167	189	111	212	41.3
C2	84.1	90.2	53.7	112	23.2

- **Configuração C3:** Esta configuração exibiu um tempo médio de 264 ms. O tempo de execução variou de 133 ms a 439 ms. O desvio padrão de 130 ms reflete a variabilidade no desempenho.

- **Configuração C1:** Com um tempo médio de 167 ms, esta configuração teve tempos que variaram entre 111 ms e 212 ms. O desvio padrão de $41,3$ ms indica variabilidade moderada.

- **Configuração C2:** Esta configuração, sendo a mais eficiente das três, apresentou um tempo médio de apenas $84,1$ ms. Os tempos variaram de $53,7$ ms a 112 ms, com um desvio padrão de $23,2$ ms, indicando consistência.

A tabela 38 apresenta um resumo dos tempos das operações para a configuração *C3* em diferentes performances.

Tabela 38 – Resumo das comparações de performance para a configuração C3

Performance	ds	Param.	Média	Desvio	Mediana
Melhor	10	1	133	50.4	131
Melhor	3	2	421	111	417
Melhor	10	3	224	75	220
Mediana	1	1	134	52.6	131
Mediana	1	2	435	105	433
Mediana	3	3	225	76.2	221
Pior	5	1	137	52.5	133
Pior	10	2	439	105	435
Pior	1	3	228	73.7	222

- **Performance Melhor:** Nas melhores performances, os tempos médios foram de 133 ms, 421 ms e 224 ms para os conjuntos de parâmetros *ds* e *par/pay* (10,1), (3,2) e (10,3) respectivamente.

- **Performance Mediana:** Nas performances medianas, os tempos médios foram de 134 ms, 435 ms e 225 ms para os conjuntos de parâmetros *ds* e *par/pay* (1,1), (1,2) e (3,3) respectivamente.

- **Performance Pior:** Nas piores performances, os tempos médios foram de 137 ms, 439 ms e 228 ms para os conjuntos de parâmetros *ds* e *par/pay* (5,1), (10,2) e (1,3) respectivamente.

7.1.1 Visualização gráfica dos resultados (biblioteca)

Para uma representação mais intuitiva dos resultados, foi gerado o gráfico da figura 7 abaixo. Este gráfico ilustra os tempos de resposta e suas médias correspondentes para diferentes tamanhos de *datasets*. A legenda "parpay" refere-se aos diversos parâmetros empregados na biblioteca. Neste contexto, o termo "parpay" é uma nomenclatura genérica adotada no *dataframe* de análise, podendo representar tanto parâmetros quanto *payloads*, dependendo se o contexto é de biblioteca ou API.

Interpretação dos Gráficos de Linha para o Dataset da Biblioteca

- **Variação de Desempenho**: Existe uma variação significativa no desempenho entre as diferentes configurações. Por exemplo, a configuração **C1** tende a ter tempos de execução mais longos em comparação com outras configurações para a maioria dos tamanhos de *datasets*.

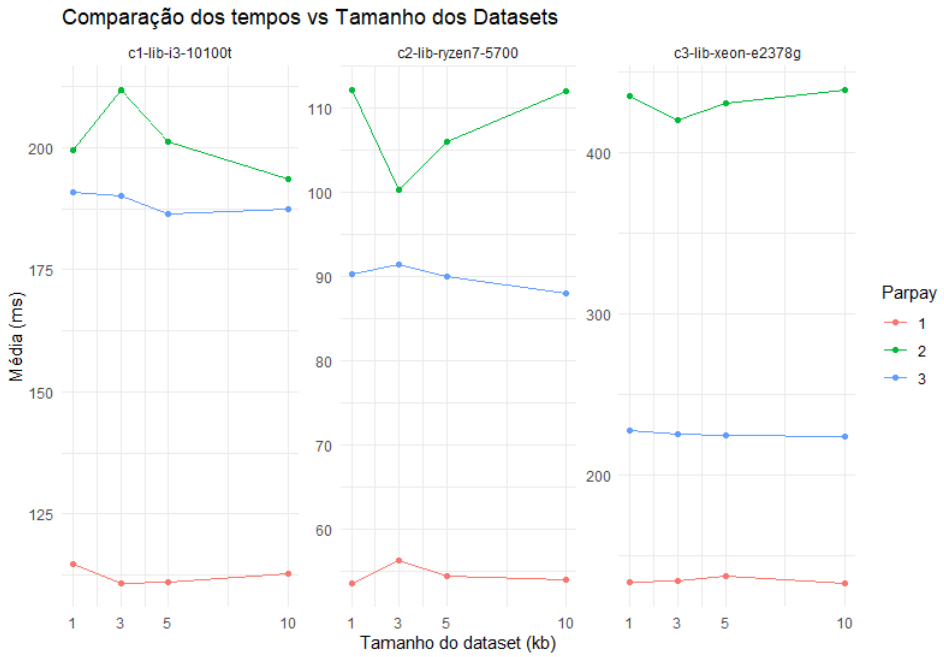

Figura 7 – Tempos de respostas de todas as configurações

- **Impacto do Tamanho do *Dataset***: Em algumas configurações, como C1 e C2, o tempo médio de execução aumenta com o tamanho do *dataset*. No entanto, em outras configurações, como **C3**, o tempo de execução parece ser bastante constante, independentemente do tamanho do *dataset*.

- **Influência do parpay**: Em quase todas as configurações, diferentes valores de *parpay* produzem diferentes trajetórias de desempenho. Em algumas configurações, um *parpay* mais alto leva a um tempo de execução mais longo, enquanto em outras o efeito é o oposto.

7.2 Resultados do web service (de-id)

No *web service*, os dados a serem anonimizados são enviados por meio de uma requisição HTTP utilizando o método POST. Essa requisição pode ser síncrona ou assíncrona. O *payload* é formatado em JSON e direcionado a uma rota específica para cada tipo de requisição. Ele contém as informações dos *datasets* conforme ilustrado na listagem 5, bem como os parâmetros essenciais para a operação, como o tipo de anonimização, campos e outros parâmetros relacionados.

O próximo resumo, descrito pela tabela 39, apresenta os tempos de resposta da API ao comparar diferentes configurações.

Tabela 39 – Resumo das comparações usando API

Perfor.	th	ds	Payl.	Config.	Req.	Méd.	Mediana	Desv.
Melhor	5	3	1	T1	Assínc.	15.4	15.1	1.29
Melhor	5	1	2	T1	Assínc.	15.5	15.2	1.18
Melhor	5	1	3	T1	Assínc.	15.9	15.6	1.25
Mediana	5	5	1	T3	Assínc.	353	353	9.58
Mediana	5	10	2	T1	Sínc.	204	204	1.68
Mediana	5	10	3	T1	Sínc.	176	176	3.17
Pior	5	10	1	T4	Assínc.	1198	1202	75.4
Pior	5	10	2	T4	Sínc.	1222	122.	91.7
Pior	5	10	3	T4	Sínc.	1624	1622	76.6

- **Melhor Performance (API):** Nas condições ideais do ambiente **T1** e em modo assíncrono, os tempos médios foram consistentemente baixos, variando entre $15,4$ ms e $15,9$ ms.
- **Performance Mediana (API):** A operação assíncrona no ambiente **T3** registrou 353 ms, enquanto em **T1**, os modos síncrono e assíncrono variaram entre 176 ms e 204 ms. Aqui os tempos de resposta já começam a não ser tão interessantes como observamos na subseção 6.7.2
- **Pior Performance (API):** O ambiente **T4**, possivelmente com conexões mais instáveis ou sobrecarregadas, exibiu tempos significativamente mais altos, variando de 1.198 ms a 1.624 ms, tanto em modos síncronos quanto assíncronos. Aqui os tempos de resposta já são inaceitáveis se a relação for sistema-sistema, de acordo com o já observado na subseção 6.7.2.

A tabela 40 apresenta um resumo geral dos tempos das execuções diante das várias configurações de computadores que foram testadas.

Tabela 40 – Resumo das comparações entre configurações

Conf.	Média	Mediana	Mín.	Máx.	Desvio
T4	1189	1156	1104	1624	113
T3	443	396	353	778	115
T1	88.3	60.9	15.4	204	79.7
T2	69.3	53.6	37.9	133	34.1

- **Configuração T4:** Esta configuração exibiu um tempo médio de 1.189 ms. O tempo de execução variou de 1.104 ms a 1.624 ms. O desvio padrão de 113 ms reflete a variabilidade no desempenho.
- **Configuração T3:** Com um tempo médio de 443 ms, esta configuração teve tempos que variaram entre 353 ms e 778 ms. O desvio padrão de 115 ms indica uma variabilidade considerável.
- **Configuração T1:** Apresentando um tempo médio de 88.3 ms, esta configuração teve tempos que variaram de 15.4 ms a 204 ms. O desvio padrão de 79.7 ms sugere uma amplitude substancial nos tempos registrados.
- **Configuração T2:** Esta configuração, com um tempo médio de 69.3 ms, mostrou-se eficiente. Os tempos variaram de 37.9 ms a 133 ms, e um desvio padrão de 34.1 ms indica consistência.

A tabela 41 apresenta um resumo dos tempos das operações para a configuração *C3* em diferentes performances.

Tabela 41 – Resumo das comparações de performance para a configuração T2

Performance	ds	Payl.	Média	Desvio	Mediana
Melhor	1	1	37.9	1.23	38.0
Melhor	1	2	38.0	1.13	37.9
Melhor	1	3	38.6	1.61	38.5
Mediana	10	1	38.8	2.93	38.5
Mediana	5	2	112	36.3	119
Mediana	5	3	41.0	3.55	39.8
Pior	10	1	87.2	145	63.3
Pior	1	2	118	38.5	119
Pior	10	3	133	15.7	130

- **Performance Melhor:** Nas melhores performances, os tempos médios foram de 37.9 ms, 38.0 ms e 38.6 ms para os conjuntos de parâmetros *ds* e *payload* (1,1), (1,2) e (1,3) respectivamente.
- **Performance Mediana:** Nas performances medianas, os tempos médios foram de 38.8 ms, 112 ms e 41.0 ms para os conjuntos de parâmetros *ds* e *payload* (10,1), (5,2) e (5,3) respectivamente.

- **Performance Pior:** Nas piores performances, os tempos médios foram de 87.2 ms, 118 ms e 133 ms para os conjuntos de parâmetros *ds* e *payload* (10,1), (1,2) e (10,3) respectivamente.

7.2.1 Visualização gráfica dos resultados (web service)

Nesta seção, apresentamos gráficos que oferecem informações adicionais sobre os resultados dos testes realizados no *web service*.

A figura 8 apresenta os tempos dos diferentes cenários com relação ao envio de requisições síncronas e assíncronas.

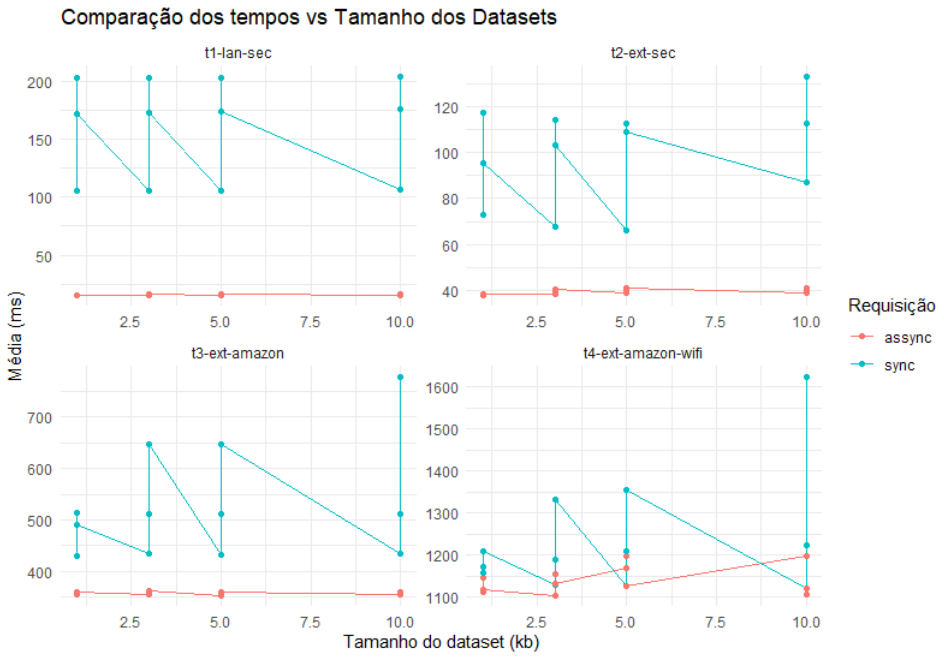

Figura 8 – Comparação de tempos em requisições síncronas e assíncronas

Interpretação do gráfico comparativo entre requisições síncronas e assíncronas

- **Variação entre configurações**: Assim como no gráfico da biblioteca, vemos uma variação significativa no desempenho entre as diferentes configurações de API.
- **Tipos de requisições**: As requisições assíncronas (“assync”) tendem a ter tempos de execução muito mais curtos do que as solicitações síncronas (“sync”) em todas as configurações e tamanhos de *dataset*.

- **Efeito do tamanho do *Dataset***: Em algumas configurações, o tempo médio de execução aumenta com o tamanho do *dataset*, especialmente para solicitações síncronas. Por exemplo, para a configuração **T1**, o tempo de execução síncrono aumenta substancialmente à medida que o tamanho do *dataset* cresce.
- **Links externos (internet)**: As configurações que fazem uso da internet, como **T2**, **T3** e **T4**, tendem a ter tempos de execução mais longos, especialmente para solicitações síncronas. Isso pode ser devido a latências de rede ou outros fatores associados a conexões externas.
- **Consistência nas solicitações assíncronas**: As linhas representando solicitações assíncronas são relativamente planas em todas as configurações, indicando que o tempo de execução assíncrono é bastante consistente, independentemente do tamanho do *dataset*.

Já as figuras 9, 10, 11 apresentam as comparações de tempos para a configuração **T2**, separados por *payload*.

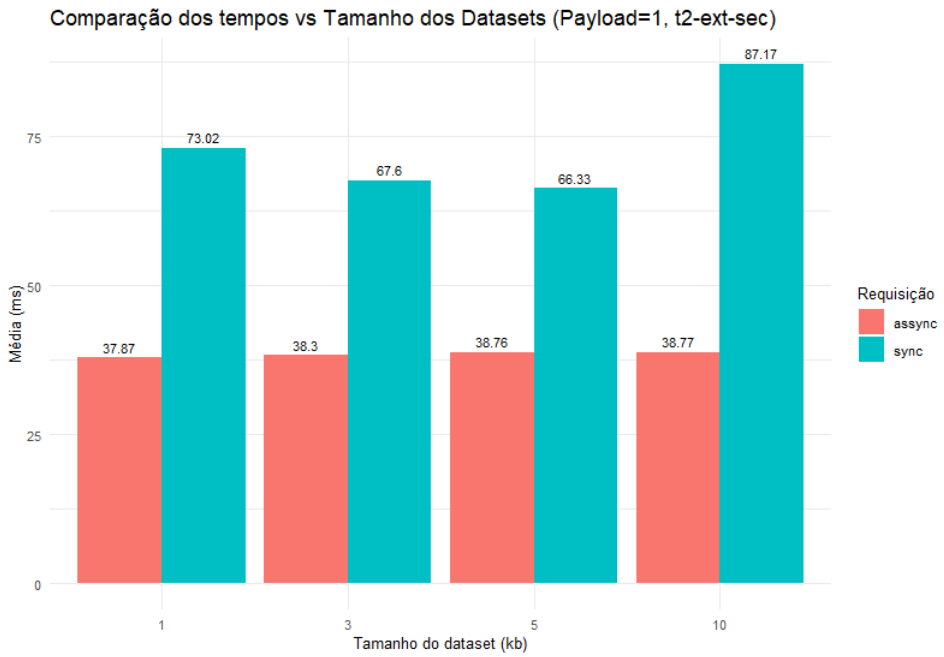

Figura 9 – Comparação sync/assync para T2 - P1

No gráfico da figura 9 é possível perceber, como nos gráficos anteriores, que o tempo médio de execução para solicitações assíncronas é

consistentemente mais baixo do que para solicitações síncronas em todos os tamanhos de *dataset*. Para solicitações assíncronas, o tempo de execução é bastante estável, independentemente do tamanho do *dataset*. Já para solicitações síncronas, há um aumento notável no tempo de execução à medida que o tamanho do *dataset* aumenta.

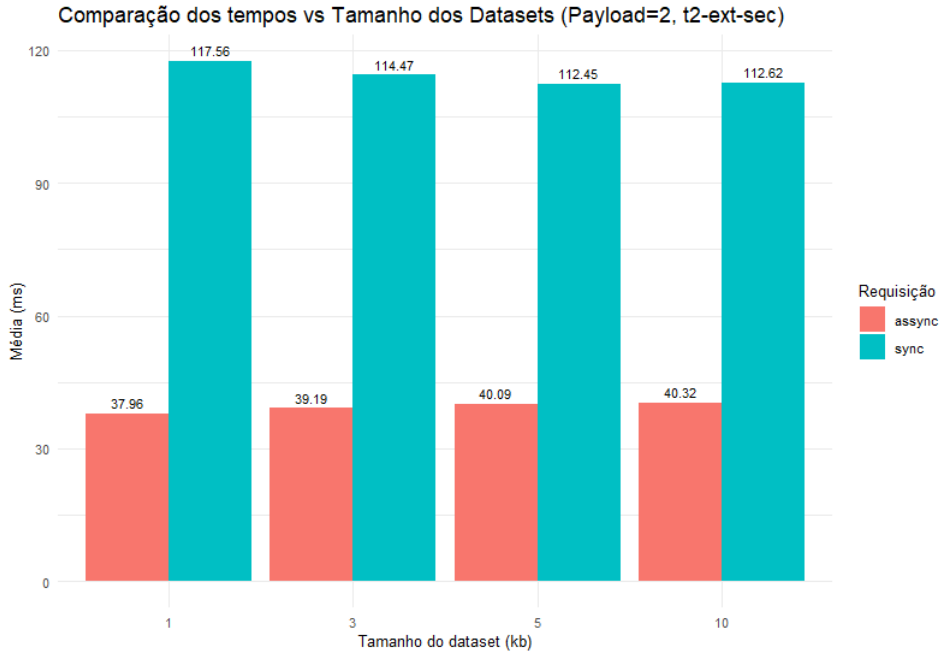

Figura 10 – Comparação sync/assync para T2 - P2

Novamente, como apresentado no gráfico da figura 10, as solicitações assíncronas são mais rápidas em comparação com as síncronas. A diferença no tempo de execução entre solicitações assíncronas e síncronas parece ser mais pronunciada no *payload* 2, especialmente para *datasets* maiores.

A tendência é semelhante às observadas para os valores de *payload* anteriores no gráfico da figura 11. O tempo de execução para solicitações síncronas continua a aumentar com o tamanho do *dataset*, enquanto as solicitações assíncronas permanecem consistentes.

7.3 Análise do uso de CPU do Servidor Dell R250

Assim como descrito na seção 7.3, observamos o uso de CPU e memória do servidor durante os testes desta etapa do projeto. Foram executadas múltiplas instâncias simultâneas de operações de anonimização de dados,

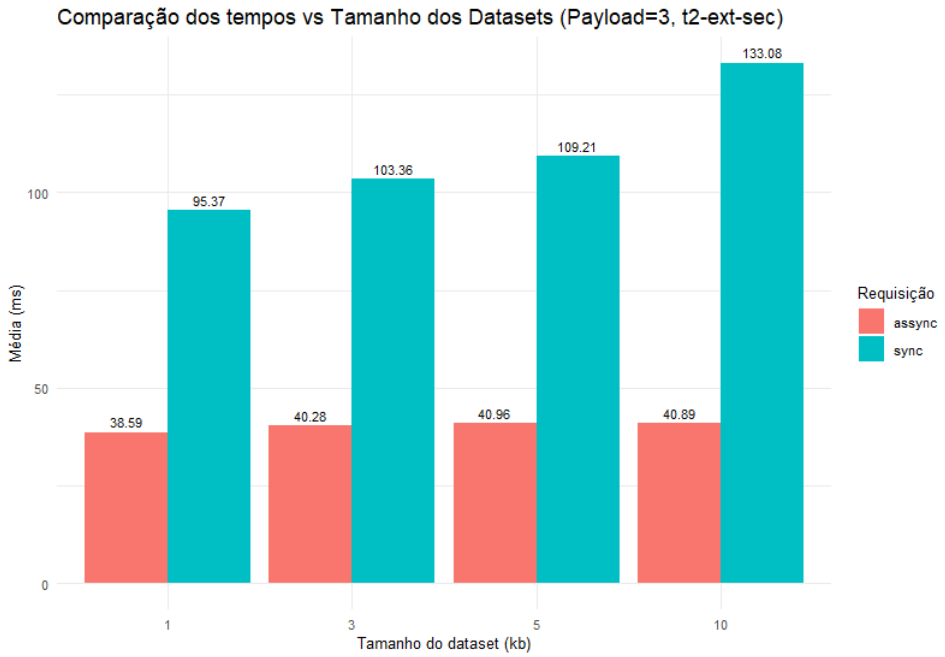

Figura 11 – Comparação sync/assync para T2 - P3

tanto no teste da biblioteca quanto do *web service*, cujos resultados já foram apresentados nas seções anteriores.

O resultado foi semelhante ao apresentado quando da execução dos testes sobre os mecanismos de compartilhamento de segredos. Menos que 50% de uso de CPU foi observado. Os demais recursos como memória e I/O também não apresentaram aumento de uso significativo.

Etapa 3

8 Compartilhamento da experiência obtida com o projeto

Nesta etapa o projeto deverá estar finalizado e seus resultados deverão ser apresentados para a comunidade, tanto do ponto de vista técnico quanto do ponto de vista de pesquisa aplicada.

8.1 Metas

- Compartilhar os códigos-fonte em repositórios de *software* livre;
- Escrita de artigo ou relatório técnico;
- Seminários sobre a experiência e seus resultados.

8.2 Resultados

Os resultados do projeto foram disseminados de três maneiras distintas:

1. **Código Fonte:** O código fonte dos projetos desenvolvidos foi disponibilizado em um repositório público, permitindo o acesso por outros pesquisadores interessados em dar continuidade à pesquisa, bem como instituições que desejem implementar os mecanismos de segurança desenvolvidos em suas soluções tecnológicas.

    - **Secstor Secret Sharing Library**
      Secstor, Project. 2022.
      Disponível em: <https://github.com/secstorproject/secstor-library>
    - **Secstor Secret Sharing API**
      Secstor, Project. 2022.
      Disponível em: <https://github.com/secstorproject/secstor>
    - **Secstor Secret Sharing API - No Auth**
      Secstor, Project. 2022.
      Disponível em: <https://github.com/secstorproject/secstor-no-auth>

- **Secstor Secret Sharing API Fetcher**
  Secstor, Project. 2022.
  Disponível em: <https://github.com/secstorproject/secstor-api-fetcher>
- **Secstor Anonymizer Library**
  Secstor, Project. 2023.
  Disponível em: <https://github.com/secstorproject/secstor-anonymizer_lib_fetcher>
- **Secstor Anonymizer Library Fetcher**
  Secstor, Project. 2023.
  Disponível em: <https://github.com/secstorproject/secstor-anonymizer_lib_fetcher>
- **Secstor DE-IDentification API**
  Secstor, Project. 2023.
  Disponível em: <https://github.com/secstorproject/secstor-de-id>
- **Secstor DE-IDentification API Fetcher**
  Secstor, Project. 2023.
  Disponível em: <https://github.com/secstorproject/secstor-de-id_api_fetcher>
- **Análise dos Resultados do Projeto Secstor**
  Secstor, Project. 2023.
  Disponível em: <https://github.com/secstorproject/secstor-results>

2. **Apresentação de Poster:** Durante a Semana Nacional de Tecnologia do Campus Canoinhas (SNCT 2023).

3. **Palestra:** No Workshop de Informática do Campus Canoinhas, realizado em 2023.

4. **Disponibilização temporária dos serviços**: Os serviços foram disponibilizados temporariamente, até o fim do projeto, nos seguintes endereços:

   - **Secstor Secret Sharing API** - <http://sector.canoinhas.ifsc.edu.br:8080/>
   - **Secstor De-ID API** - <http://sector.canoinhas.ifsc.edu.br:40123/>
   - **Secstor De-ID API** - <https://api.de-id.com.br/>

Metas Não Alcançadas

9 Considerações revisadas sobre implementação de técnicas híbridas

No projeto inicial, considerávamos a possibilidade de combinar técnicas de compartilhamento de segredos com métodos de anonimização de dados, vislumbrando um cenário promissor com tal integração. Acreditávamos que essa fusão poderia maximizar os benefícios de ambas as estratégias, resultando em uma solução robusta para a privacidade e segurança dos dados. No entanto, ao aprofundarmos nosso estudo nas especificidades de cada técnica, identificamos que sua combinação é mais desafiadora do que inicialmente antecipado. Por isso, foi necessário revisar essa abordagem. Os pormenores de nossas observações são detalhados a seguir.

9.1 Natureza dos dados no compartilhamento de segredos

O compartilhamento de segredos, em sua essência, é um método para dividir um segredo S em n partes, de modo que apenas um limite específico t dessas partes possa reconstruir o segredo original. Matematicamente, isso pode ser representado como:

$$S = f(s_1, s_2, \ldots, s_t) \tag{1}$$

onde s_i são os compartilhamentos (partes) individuais do segredo.

Um aspecto crucial do compartilhamento de segredos é que partes individuais s_i não são estruturados de uma maneira que retenha a semântica dos dados originais. São valores essencialmente aleatórios que, quando combinados da maneira certa, revelam o segredo. Esta falta de estrutura significa que os dados, uma vez passados através de um esquema de compartilhamento de segredos, perdem o seu formato e contexto originais.

9.2 Anonimização de dados e suas restrições

A anonimização de dados, por outro lado, visa transformar os dados de tal forma que os pontos de dados individuais não possam ser rastreados até entidades específicas, mantendo ao mesmo tempo a utilidade geral dos dados. Um princípio fundamental da anonimização de dados é que, uma vez anonimizados, os dados não devem ser desanonimizados, visando a proteção da privacidade individual.

Vamos considerar um conjunto de dados D que foi anonimizado para D'. A função de transformação pode ser representada como:

$$D' = g(D) \tag{2}$$

onde g é a função de anonimização (mascaramento, embaralhamento, generalização, etc).

O desafio surge quando consideramos o processo inverso. Para que a anonimização de dados seja eficaz:

$$D \neq g^{-1}(D') \tag{3}$$

Ou seja, o conjunto de dados original D não deve ser recuperável de D' usando o inverso da função de anonimização g^{-1}.

9.3 Integração das abordagens

Dada a natureza dos dados no compartilhamento de segredos e as restrições da anonimização dos dados, a integração das duas técnicas apresenta desafios inerentes:

- Como os dados gerados através do compartilhamento de segredos perdem sua estrutura, a aplicação de técnicas de anonimização de dados nesses dados torna-se algo não trivial. A anonimização depende de dados estruturados para manter a utilidade e ao mesmo tempo garantir a privacidade.

- Por outro lado, se considerarmos a aplicação de técnicas de compartilhamento de segredos em dados já anonimizados, encontraremos um obstáculo. A essência da anonimização de dados é que os dados não

devem ser retornamos na sua forma original. No entanto, o compartilhamento de segredos, por definição, permite a reconstrução dos dados originais (o segredo) quando o número limite de compartilhamentos é combinado. Isto vai contra o princípio da anonimização irreversível.

É importante ressaltar que, esta observação não reflete necessariamente uma falha de planejamento, mas sim destaca a meticulosidade de nossa abordagem investigativa. Nossa dedicação em buscar soluções práticas e seguras nos levou à reflexão de que, no momento atual, a ideia de implementações híbridas entre compartilhamento de segredos e anonimização de dados pode ser mais problemática do que benéfica.

Na ciência, erros são mais do que inevitáveis; são essenciais. Em vez de representarem falhas, eles destacam limites do nosso conhecimento e nos incentivam a reavaliar nossos rumos. Muitas descobertas revolucionárias surgiram de experimentos inesperados ou hipóteses erradas. Assim, erros são fundamentais para o progresso científico, moldando e enriquecendo nossa compreensão contínua.

Considerações Finais

10 Visão geral do relatório

Ao longo deste relatório, exploramos a aplicação de algoritmos de compartilhamento de segredos e anonimização de dados com o objetivo de adequar sistemas à LGPD. Através das diversas etapas do projeto, foi possível compreender a importância e os desafios associados à proteção de dados pessoais, especialmente no contexto atual em que a privacidade se tornou uma preocupação central.

Na primeira etapa, realizamos uma revisão aprofundada da LGPD, bem como da literatura e ferramentas relevantes. Esta revisão forneceu uma base sólida para as etapas subsequentes, garantindo que as abordagens adotadas estivessem alinhadas com as melhores práticas e regulamentações vigentes.

A segunda etapa focou na implementação, testes, revisão e avaliação das abordagens selecionadas. Os testes realizados ofereceram informações valiosas sobre o desempenho e a eficácia das técnicas de compartilhamento de segredos e anonimização. Além disso, a comparação entre diferentes algoritmos e configurações permitiu identificar as melhores estratégias para diferentes cenários de uso.

Finalmente, na terceira etapa, compartilhamos a experiência obtida com o projeto, destacando as metas alcançadas, os desafios enfrentados e os resultados obtidos. Esta etapa também abriu caminho para trabalhos futuros, onde planejamos condensar as informações deste relatório e elaborar artigos para apresentação em simpósios e conferências.

Em conclusão, este projeto não apenas contribuiu para a compreensão das técnicas de proteção de dados, mas também forneceu ferramentas práticas para ajudar as organizações a se adaptarem à LGPD. À medida que a legislação e as expectativas em torno da privacidade continuam a evoluir, é essencial que continuemos a pesquisa e o desenvolvimento nesta área, garantindo que os sistemas estejam sempre em conformidade e que os dados dos indivíduos estejam protegidos.

10.1 Trabalhos Futuros

Com base no conteúdo do relatório, identificamos algumas possíveis direções para trabalhos futuros:

1. **Estudos de Caso em Setores Específicos**: Seria interessante aplicar as técnicas de compartilhamento de segredos e anonimização de dados em setores específicos, como saúde ou finanças, e analisar os desafios e benefícios específicos desses setores.

2. **Avaliação de Conformidade com Outras Regulamentações**: Além da LGPD, existem outras regulamentações de privacidade em diferentes regiões, como o GDPR na Europa. Um estudo comparativo sobre como as técnicas se alinham a diferentes regulamentações seria valioso.

3. **Treinamento e Workshops**: Com base no conhecimento adquirido durante este projeto, poderiam ser organizados workshops ou sessões de treinamento para empresas e profissionais interessados em se adequar à LGPD.

4. **Integração com Tecnologias Emergentes**: Explorar como as técnicas de compartilhamento de segredos e anonimização podem ser integradas com tecnologias emergentes, como *blockchain* ou inteligência artificial, para criar soluções de privacidade mais robustas.

5. **Avaliação de Desempenho em Ambientes de Grande Escala**: Embora este relatório tenha abordado testes e avaliações, um estudo mais aprofundado sobre o desempenho dessas técnicas em ambientes de grande escala ou soluções distribuídas, como grandes bancos de dados ou sistemas em nuvem, seria benéfico.

Agradecimentos

Gostaríamos de expressar nossa profunda gratidão à Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC) pelo apoio financeiro concedido através do edital 27/2021. Este fomento foi fundamental para a realização desta pesquisa, permitindo a aquisição de um servidor essencial para a execução dos testes e hospedagem dos *web services* desenvolvidos. Além disso, a contribuição da FAPESC possibilitou a concessão de bolsas de iniciação científica para dois alunos, enriquecendo sua formação

acadêmica e proporcionando uma experiência valiosa no mundo da pesquisa. Também foi possível contar com o serviço de um programador para desenvolvimento complementar, garantindo a qualidade e eficiência dos sistemas implementados.

A presença e o apoio da FAPESC foram cruciais para o sucesso deste projeto, e estamos sinceramente gratos por sua confiança e investimento em nossa pesquisa. Esperamos que os resultados alcançados possam contribuir significativamente para o avanço científico e tecnológico de nosso estado e país.

Referências

AGGARWAL, C. C.; PHILIP, S. Y. *Privacy-preserving data mining: models and algorithms*. [S.l.]: Springer Science & Business Media, 2008. Citado na página 29.

BARRETO, L. *Controle de autenticação tolerante a intrusões em federações de clouds*. Tese (Tese (Doutorado em Engenharia de Automação e Sistemas)) — Universidade Federal de Santa Catarina, [S. l.], 2017. Citado na página 19.

BARRETO, L. et al. Secure storage of user credentials and attributes in federation of clouds. In: *Proceedings of the Symposium on Applied Computing*. [S.l.: s.n.], 2017. p. 364–369. Citado na página 19.

BAYARDO, R. J.; AGRAWAL, R. Data privacy through optimal k-anonymization. In: *21st International conference on data engineering (ICDE'05)*. [S.l.]: IEEE, 2005. p. 217–228. Citado na página 19.

BESSANI, A. et al. Depsky: Dependable and secure storage in a cloud-of-clouds. In: *European Systems Conference (EuroSys'11)*. [S.l.: s.n.], 2011. Citado na página 18.

BRASIL. *Lei nº 13.709, de 14 de agosto de 2018. Dispõe sobre a proteção de dados pessoais e altera a Lei nº 12.965, de 23 de abril de 2014 (Marco Civil da Internet)*. Brasília, DF: Presidência da República, 2018. Acessado em: 27 jul. 2021. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709.htm>. Citado 3 vezes nas páginas 15, 21 e 23.

CEARá, T. de Justiça do Estado do. *LGPD - Dados Pessoais*. 2021. TJCE. Disponível em: <https://www.tjce.jus.br/lgpd/lgpd-dados-pessoais/>. Citado na página 23.

DOMINGO-FERRER, J. Personal big data, gdpr and anonymization. In: *International Conference on Flexible Query Answering Systems*. [S.l.]: Springer, Cham, 2019. p. 7–10. Citado na página 19.

FOUNDATION, D. S. *Django Documentation*. 2022. Version 3.2. Disponível em: <https://docs.djangoproject.com/>. Citado na página 70.

GARTNER. *Forecasts Worldwide Public Cloud End-User Spending to Grow 18% in 2021*. 2021. Acessado em: 29 jul. 2021. Disponível em: <https://www.gartner.com/en/newsroom/press-releases/2021-04-21-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-23-perce Citado na página 15.

GOOGLE. *The Need for Mobile Speed*. 2022. Acessado em 03 de outubro de 2023. Disponível em: <https://blog.google/products/admanager/the-need-for-mobile-speed/>. Citado na página 65.

GROUP, N. N. *Website Response Times*. 2010. Acessado em 03 de outubro de 2023. Disponível em: <https://www.nngroup.com/articles/website-response-times/>. Citado na página 65.

KRAWCZYK, H. Secret sharing made short. In: SPRINGER. *Annual international cryptology conference*. [S.l.], 1993. p. 136–146. Citado 2 vezes nas páginas 27 e 37.

KUMAR, A. et al. Secure storage and access of data in cloud computing. In: *2012 International Conference on ICT Convergence (ICTC)*. [S.l.: s.n.], 2012. p. 336–339. Citado na página 18.

LORUENSER, T.; HAPPE, A.; SLAMANIG, D. Archistar: towards secure and robust cloud based data sharing. In: IEEE. *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. [S.l.], 2015. p. 371–378. Citado na página 27.

MOCKAROO. *Mockaroo - Random Data Generator and API Mocking Tool*. 2023. Accessado: 30/08/2023. Disponível em: <https://www.mockaroo.com/>. Citado na página 43.

NEVES, B. A. *PVSS Java*. 2008. Acessado em: 29/08/2023. Disponível em: <https://navigators.di.fc.ul.pt/software/jitt/src/jss-0.1.tar.gz>. Citado na página 37.

OVERFLOW, S. *Stack Overflow - Where Developers Learn, Share, and Build Careers*. 2022. Acessado em 03 de outubro de 2023. Disponível em: <https://stackoverflow.com/>. Citado na página 65.

PINHEIRO, P. P. *Proteção de dados pessoais: Comentários à lei n. 13.709/2018-lgpd*. [S.l.]: Saraiva Educação SA, 2020. Citado na página 23.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2020. Disponível em: <https://www.R-project.org/>. Citado na página 36.

RABIN, T.; BEN-OR, M. Verifiable secret sharing and multiparty protocols with honest majority. In: *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. [S.l.: s.n.], 1989. p. 73–85. Citado 2 vezes nas páginas 27 e 37.

RAGHUNATHAN, B. *The complete book of data anonymization: from planning to implementation*. [S.l.]: CRC Press, 2013. Citado 2 vezes nas páginas 15 e 19.

RAGHUNATHAN, B. *The complete book of data anonymization: from planning to implementation*. [S.l.]: CRC Press, 2013. Citado na página 29.

SCHOENMAKERS, B. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: *Advances in Cryptology (CRYPTO99)*. [S.l.: s.n.], 1999. v. 1666, n. i, p. 148–164. Citado 5 vezes nas páginas 15, 18, 25, 26 e 37.

SECSTOR, P. *Secstor Secret Sharing API*. [S.l.]: Secstor Project, 2022. <https://github.com/secstorproject/secstor>. Citado na página 38.

SECSTOR, P. *Secstor Secret Sharing API - No Auth*. [S.l.]: Secstor Project, 2022. <https://github.com/secstorproject/secstor-no-auth>. Citado na página 38.

SECSTOR, P. *Secstor Secret Sharing API Fetcher*. [S.l.]: Secstor Project, 2022. <https://github.com/secstorproject/secstor-api-fetcher>. Citado na página 44.

SECSTOR, P. *Secstor Secret Sharing Library*. [S.l.]: Secstor Project, 2022. <https://github.com/secstorproject/secstor-library>. Citado 2 vezes nas páginas 37 e 44.

SECSTOR, P. *Análise dos Resultados do Projeto Secstor*. [S.l.]: Secstor Project, 2023. <https://github.com/secstorproject/secstor-results>. Citado 2 vezes nas páginas 44 e 72.

SECSTOR, P. *Secstor Anonymizer Library*. [S.l.]: Secstor Project, 2023. <https://github.com/secstorproject/secstor-anonymizer_lib_fetcher>. Citado na página 70.

SECSTOR, P. *Secstor DE-IDentification API*. [S.l.]: Secstor Project, 2023. <https://github.com/secstorproject/secstor-de-id>. Citado na página 70.

SHAMIR, A. How to share a secret. *Communications of the ACM*, v. 22, n. 11, p. 612–613, 1979. Citado 4 vezes nas páginas 15, 18, 24 e 37.

SOCIAL, B. N. de Desenvolvimento Econômico e. *Lei Geral de Proteção de Dados (LGPD)*. 2021. BNDES. Disponível em: <https://www.bndes.gov.br/wps/portal/site/home/transparencia/lgpd>. Citado na página 23.

SPRING Framework. 2023. <https://spring.io>. Acessado em 26 de setembro de 2023. Citado na página 38.

STADLER, M. Publicly verifiable secret sharing. In: SPRINGER. *International Conference on the Theory and Applications of Cryptographic Techniques*. [S.l.], 1996. p. 190–199. Citado na página 26.

STALLINGS, W. *Criptografia e segurança de redes: princípios e práticas*. 6. ed. [S.l.]: Pearson, 2015. Citado na página 15.

SUJANA, B. et al. Secure framework for data storage from single to multi clouds in cloud networking. *International Journal of Emerging Trends and Technology in Computer Science*, v. 2, n. 2, 2013. Citado na página 18.

TECHNOLOGY, A. Austrian Institute of. *Archistar*. [S.l.]: Austrian Institute of Technology, 2021. <https://github.com/Archistar>. Citado na página 37.

ZHOU, B.; PEI, J.; LUK, W. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM Sigkdd Explorations Newsletter*, v. 10, n. 2, p. 12–22, 2008. Citado na página 19.