UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

MARCOS MACIEL DE CASTRO

G4DPA: UMA ARQUITETURA PARA COMPARTILHAMENTO AUDITÁVEL DE ARQUIVOS CONTENDO DADOS PESSOAIS

FORTALEZA

2024

MARCOS MACIEL DE CASTRO

G4DPA: UMA ARQUITETURA PARA COMPARTILHAMENTO AUDITÁVEL DE ARQUIVOS CONTENDO DADOS PESSOAIS

Dissertação apresentada ao Curso de Mestrado em Ciência da Computação do Programa de Pós-Graduação em Ciências da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação.

Orientador: Prof. Dr. Miguel Franklin de Castro.
Coorientador: Prof. Dr. Dário Vieira Conceição.

FORTALEZA

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C352g Castro, Marcos Maciel de.
G4DPA: Uma Arquitetura para Compartilhamento Auditável de Arquivos Contendo Dados Pessoais / Marcos Maciel de Castro. – 2024.
194 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2024.
Orientação: Prof. Dr. Miguel Franklin de Castro.
Coorientação: Prof. Dr. Dário Vieira Conceição.

1. Blockchain. 2. Contrato inteligente. 3. IPFS. 4. Proteção de dados pessoais. 5. Auditoria. I. Título.
CDD 005

MARCOS MACIEL DE CASTRO

G4DPA: UMA ARQUITETURA PARA COMPARTILHAMENTO AUDITÁVEL DE ARQUIVOS CONTENDO DADOS PESSOAIS

Dissertação apresentada ao Curso de Mestrado em Ciência da Computação do Programa de Pós-Graduação em Ciências da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação.

Aprovada em: 2024

BANCA EXAMINADORA

Prof. Dr. Miguel Franklin de Castro (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Dário Vieira Conceição (Coorientador)
École d'Ingénieurs Généraliste du Numérique (Efrei Paris)

Profa. Dra. Rossana Maria de Castro Andrade
Universidade Federal do Ceará (UFC)

Prof. Dr. Arlindo Flávio da Conceição
Universidade Federal de São Paulo (UNIFESP)

À minha esposa e aos meus filhos. Sem eles, muito provavelmente eu não teria chegado até aqui.

AGRADECIMENTOS

Primeiramente, agradeço a Deus, cuja presença constante me proporcionou força e coragem para enfrentar os desafios desta jornada.

Minha profunda gratidão ao meu orientador, Prof. Dr. Miguel Franklin de Castro, por sua orientação, paciência e contribuições valiosas, e ao meu co-orientador, Prof. Dr. Dário Vieira Conceição, pelo apoio e conselhos enriquecedores.

Ao Prof. Marciel Barros, agradeço por sua generosidade em compartilhar conhecimento; seu apoio e orientação foram cruciais para o sucesso desta pesquisa.

Aos meus colegas de trabalho, Jefferson, Herdine, Marcos Antônio, e, em especial, Robson e Marcelo, deixo meu sincero agradecimento pelo apoio e incentivo.

Aos meus pais, Francy e Antônio, sou grato pelo amor e pelos ensinamentos que me sustentaram ao longo da vida. Nada disso seria possível sem os valores que me transmitiram.

Aos meus filhos, Marcos Vinícius e Sofia, agradeço por serem minha fonte constante de alegria e inspiração. Este trabalho é também por vocês e para vocês, e espero que ele os inspire a sempre perseguirem seus sonhos com determinação.

Finalmente, à minha esposa, Elaine, minha parceira de vida, sou grato pelo amor e suporte incondicional. Este trabalho é um reflexo do nosso esforço conjunto, e compartilho com você cada conquista alcançada.

A todos os demais que, de alguma forma, contribuíram para a conclusão deste trabalho, deixo aqui meu mais sincero agradecimento.

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

"Nothing was your own except the few cubic centimeters inside your skull." (George Orwell, "Nineteen Eighty-Four", 1949.)

RESUMO

Nos últimos anos, vazamentos de dados pessoais tornaram-se frequentes, causando prejuízos materiais ou morais de difícil reparação a pessoas e organizações. Esse cenário, aliado ao aumento do armazenamento e processamento de dados pessoais em nuvem, impulsionou o desenvolvimento ou a melhoria de regulamentos de proteção de dados pessoais no mundo inteiro, como a GDPR na União Europeia e a LGPD no Brasil, tendo em vista melhor proteger a privacidade das pessoas. Com isso, é importante que organizações que processam dados pessoais busquem uma solução para evidenciar que seguem as práticas legais aplicáveis, possibilitando, inclusive, a realização de auditorias a fim de evitar sanções estatais, como a aplicação de multas. Todavia, ainda não está claro qual a melhor técnica para prover tal solução. Este trabalho tem por finalidade apresentar uma arquitetura para realização de busca e recuperação de arquivos contendo dados pessoais entre instituições distintas, sem que seja necessário armazenar os arquivos diretamente em um repositório centralizado ou expor sistemas internos dessas instituições para receber acessos vindos da internet. Ademais, propõe um mecanismo para auditar as operações de tratamento sobre compartilhamentos realizados, resguardando a privacidade dos titulares e garantindo a conformidade com normas de proteção de dados pessoais. Foi realizado um levantamento na literatura científica para identificar os principais pontos de não conformidade com esses regulamentos nos trabalhos existentes e as principais técnicas e tecnologias utilizadas. A partir dessas informações, é apresentada a arquitetura *Gateway for Data Privacy Auditing* (G4DPA), baseada em um *gateway* para intermediar as operações de registro, busca e solicitação de compartilhamento de arquivos de dados pessoais entre os controladores. O *gateway* realiza a pseudoanonimização das informações de custódia dos arquivos e das identidades das partes envolvidas antes de enviá-las a uma *blockchain* através de funções de um contrato inteligente, enquanto a transmissão do conteúdo dos arquivos é realizada através de uma rede IPFS diretamente entre os controladores. Uma prova de conceito foi implementada utilizando uma *blockchain* Ethereum, e alguns experimentos foram conduzidos em um cenário hipotético no contexto de *e-healthcare*. Por fim, os resultados dos experimentos são apresentados e analisados em relação a custos, capacidade e desempenho da solução proposta.

Palavras-chave: arquitetura; blockchain; contrato inteligente; Ethereum; IPFS; proteção de dados pessoais; GDPR; LGPD; compartilhamento de arquivos; auditoria; privacidade.

ABSTRACT

In recent years, the frequency of personal data breaches has increased significantly, causing serious material and moral damage to individuals and organizations alike. This scenario, coupled with the increase in the storage and processing of personal data in the cloud, has resulted in the development or improvement of personal data protection regulations around the world. Examples of such regulations include GDPR in the European Union and LGPD in Brazil, designed with the objective of enhancing the protection of the privacy of individuals. In this way, organizations engaged in the processing of personal data are imperative to identify an effective solution that can attest to their compliance with applicable regulations. This includes the capacity to undergo audits to avoid state-imposed sanctions, such as fines. However, it remains unclear which methodology represents the optimal approach to developing such a solution. The aim of this work is to present an architecture for searching and retrieving files containing personal data between different institutions, without the requirement of storing the files directly in a centralized repository or exposing the internal systems of these institutions to internet access. Furthermore, it proposes a mechanism for auditing the processing operations carried out on shared data, thereby safeguarding the privacy of data subjects and ensuring compliance with personal data protection regulations. A review of the scientific literature was conducted to identify the key areas of non-compliance with these regulations in existing works and the main techniques and technologies employed. Based on this analysis, the *Gateway for Data Privacy Auditing* (G4DPA) architecture is presented, in which a gateway is leveraged to facilitate the operations of registering, searching, and requesting the sharing of personal data files between controllers. The gateway pseudoanonymizes the file custody information and the identities of the parties involved before sending them to a blockchain through the functions of a smart contract, while the transmission of the file content is carried out through an IPFS network directly between the controllers. A proof of concept was implemented using an Ethereum blockchain and experiments were conducted in a hypothetical scenario in the context of e-healthcare. Finally, the results of the experiments are presented and analyzed in terms of costs, capacity, and performance to evaluate the proposed solution.

Keywords: architecture; blockchain; smart contract; Ethereum; IPFS; personal data protection; GDPR; LGPD; file sharing; auditing; privacy.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABAC	*Attribute-Based Access Control*
ABE	*Attribute-Based Encryption*
ABI	*Application Binary Interface*
AC	Autoridade Certificadora
AES	*Advanced Encryption Standard*
ANPD	Autoridade Nacional de Proteção de Dados
APD	Autoridade de Proteção de Dados
API	*Application Programming Interface*
APPI	Act on the Protection of Personal Information
AWS	Amazon Web Services
BBDSPP	*Blockchain-Based Data Sharing and Privacy Protection*
BC	*Blockchain*
BDDT	*Blockchain-based Distributed Data Transaction*
BI-IoT	*Blockchain and IPFS as a Service for IoT*
BPSDQS	*Blockchain-based Privacy-preserving and Sustainable Data Query Service*
BSSPD	*Blockchain-based security Sharing Scheme for Personal Data*
CC	Controlador Custodiante
CD	Controlador de Dados
CF/88	Constituição da República Federativa do Brasil de 1988
CID	*Content IDentifier*
CIS	Comunicações de Incidentes de Segurança
CNUCED	Conferência das Nações Unidas sobre Comércio e Desenvolvimento
CP-ABE	*Cyphertext-Policy Attribute Based Encryption*
CPA	Colorado Privacy Act
CPPA	California Consumer Privacy Act
CPRA	California Privacy Rights Act
CR	Controlador Requisitante
CSL	CyberSecurity Law
CTDPA	Connecticut Data Privacy Act
DAG	*Directed Acyclic Graph*

DeFi	*Decentralized Finance*
DHT	*Distributed Hash Table*
DID	*Decentralized Identifier*
DLT	*Distributed Ledger Technology*
DNS	*Domain Name System*
DPA	Data Protection Act
DSL	Data Security Law
ECC	*Ellyptic Curve Cryptography*
ECC	Ecocardiograma
ECDSA	*Elliptic Curve Digital Signature Algorithm*
ECG	Eletrocardiograma
EHR	*Electronic Health Record*
EMR	*Electronic Medical Record*
ENISA	European Union Agency for Cybersecurity
EUA	Estados Unidos da América
EVM	*Ethereum Virtual Machine*
FHIR	*Fast Healthcare Interoperability Resources*
FPE	*Format Preserving Encryption*
G4DPA	*Gateway for Data Privacy Auditing*
GCM	*Galois/Counter Mode*
GDPR	General Data Protection Regulation
GTW	*Gateway*
HIPAA	Health Insurance Portability and Accountability Act
HLF	*Hyperledger Fabric*
HLI	*Hyperledger Indy*
HMAC	*Hash-based Message Authentication Code*
HTTP	*Hypertext Transfer Protocol*
IC	Intervalo de Confiança
IIoT	*Industrial Internet of Things*
IoMT	*Internet of Medical Things*
IoT	*Internet of Things*
IoV	*Internet of Vehicles*

IPFS	*InterPlanetary File System*
IPLD	*InterPlanetary Linked Data*
IPNS	*InterPlanetary Name System*
IPP	*Information Privacy Principles*
ITS	*Intelligent Transportation System*
JPBC	*Java Pairing-Based Cryptography*
KASE	*Key Aggregate Searchable Encryption*
LGPD	Lei Geral de Proteção de Dados Pessoais
LSB	*Least Significant Bit*
LSSS	*Linear Secret Sharing Scheme*
NFT	*Non-Fungible Token*
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OAEP	*Optimal Asymmetric Encryption Padding*
OD	Operador de Dados
OTP	*One-Time Password*
P2P	*Peer-to-Peer*
PBFT	*Practical Byzantine Fault Toleration*
PCIM	*Patient Centric Image Management*
PDPA	Personal Data Protection Act
PE	*Predicate Encryption*
PHR	*Personal Health Record*
PIPA	Personal Information Protection Act
PIPEDA	Personal Information Protection and Electronic Documents Act
PIPL	Personal Information Protection Law
PKI	*Public Key Infrastructure*
PoC	*Proof of Concept*
POPIA	Protection of Personal Information Act
PoR	*Proof-of-Replication*
PoS	*Proof-of-Stake*
PoW	*Proof-of-Work*
PSS	*Probabilistic Signature Scheme*

RBAC	*Rule-Based Access Control*
RC	*Russian Constitution*
RDPA	Russian Data Protection Act
RM	Ressonância Magnética
RSA	Rivest-Shamir-Adleman
RSU	*Roadside Unit*
RX	Raio-X
SC	Strasbourg Convention
SC-ABSE	*Smart Contract-based Attribute-Based Searchable Encryption*
SGX	*Software Guard Extensions*
SHA	*Secure Hash Algorithm*
SPHRS	*Secure Patient Health Records Sharing*
SSA	*Secret Sharing Algorithm*
SSE	*Symmetric Searchable Encryption*
SSI	*Self-Sovereign Identity*
SSS	*Shamir Secret Sharing*
SSSA	*Shamir Secret Sharing Algorithm*
TBchain	*Three-tier architecture Blockchain*
TC	Tomografia Computadorizada
TD	Titular de Dados
TPA	The Privacy Act
TPS	Transações Por Segundo
TSM	*Trusted-Scoring Mechanism*
UCPA	Utah Consumer Privacy Act
UE	União Europeia
VANET	*Vehicular Ad-hoc Networks*
VCDPA	Virginia Consumer Data Protection Act
VPN	*Virtual Private Network*
WADO-RS	*Web Access to DICOM Objects*
WTSS	*Weighted Threshold Secret Sharing*
ZKP	*Zero Knowledge Proof*

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto e apresentação do problema

Nos últimos anos, vazamentos de dados pessoais, sejam decorrentes de ataques cibernéticos ou do compartilhamento indevido de informações particulares, tornaram-se frequentes, causando danos morais ou materiais de difícil reparação a pessoas e organizações. O escândalo da empresa Cambridge Analytica (ROSENBERG *et al.*, 2018), que utilizou dados de usuários do Facebook para criar anúncios direcionados e influenciar eleitores no pleito norte-americano de 2016, evidenciou o potencial que o uso dessas informações tem até mesmo sobre aspectos sociais, políticos e econômicos de grandes empresas e nações. Poucos anos antes, em 2013, Edward Snowden denunciou práticas de espionagem da National Security Agency (NSA) através de um programa de vigilância global sobre o tráfego de informações de pessoas, corporações e estados (STOYCHEFF, 2016). Mais recentemente, a companhia de gerenciamento de saúde Intellihartx confirmou que *hackers* roubaram informações médicas e números de seguridade social de meio milhão de pacientes (GEBREMICHAEL *et al.*, 2020).

Diante desse cenário, a União Europeia (UE) decidiu aprovar em 2016 o seu novo *regulamento para proteção de pessoas naturais em relação ao processamento de dados pessoais e ao livre movimento de tais dados*, intitulado GDPR (Parlamento Europeu, Conselho da União Europeia, 2016), em vigor desde 2018. Por sua vez, outras nações, inspiradas por esse regulamento vanguardista, publicaram suas próprias normas para proteção de dados pessoais. No Brasil, por exemplo, foi sancionada, em 2018, a Lei 13.709, denominada LGPD (Brasil, 2018). Tais regulamentações visam esclarecer regras, requisitos e instruções para implantação de controles pelas organizações, mas a introdução dessas obrigações exige mudanças consideráveis em seus processos e sistemas (TIKKINEN-PIRI *et al.*, 2018).

Além disso, com seu rápido desenvolvimento, a tecnologia de *Internet of Things* (IoT) tem-se tornado um importante modelo de negócios no cotidiano de muitas pessoas. A IoT é utilizada em diversos setores, tais como indústria, agricultura, cidades inteligentes e saúde. Aproveitar dados massivos de dispositivos IoT inteligentes pode reduzir custos, beneficiar a produção e auxiliar na tomada de decisões de negócios mais precisas. No entanto, restrições desses dispositivos, como capacidade de armazenamento e de comunicação e necessidade de baixo consumo energético, acabam por limitar o seu poder de processamento (XIE *et al.*, 2023). Com isso, muitas vezes dados pessoais são coletados e transferidos para processamento ou

armazenamento em nuvem, motivando questionamentos quanto à segurança dos procedimentos utilizados em sua manipulação (HOSSEIN *et al.*, 2019). Assim, a conformidade com a GDPR quanto ao tratamento, armazenamento e compartilhamento dos dados de usuários em nuvem passou a ser obrigatoriamente mais uma fonte de preocupação para as empresas afetadas por esse regulamento (RUSSO *et al.*, 2018).

Em regra, as organizações têm de demonstrar tal conformidade apenas se requeridas pela autoridade supervisora ou quando aventada suspeita de violação legal (KUNZ *et al.*, 2020). De todo modo, ainda não está clara qual a técnica mais adequada para que a indústria forneça soluções transparentes, eficientes e com uma boa relação custo/benefício para esse problema (WU *et al.*, 2019). Boa parte das atuais soluções para auditoria se baseiam em uma arquitetura cliente-servidor centralizada, com mecanismos de demonstração de conformidade que acabam deixando dúvidas quanto à lisura e à transparência de suas operações (TRUONG *et al.*, 2020).

1.2 Objetivos

A presente dissertação tem por finalidade propor, implementar e verificar o desempenho de uma arquitetura para realização de busca e recuperação de arquivos contendo dados pessoais entre instituições distintas, sem que seja necessário armazenar os arquivos diretamente em um repositório centralizado ou expor sistemas internos dessas instituições para receber acessos provenientes da internet. Ademais, propõe um mecanismo inovador para auditar as operações de tratamento realizadas sobre os dados de modo confiável e transparente, resguardando a privacidade dos titulares e garantindo a conformidade com normas de proteção de dados pessoais, como a GDPR e a LGPD. Adotaremos a GDPR, por se tratar de uma lei vanguardista na proteção de dados pessoais e que possui uma extensa gama de estudos baseados em suas premissas. Adicionalmente, a (Brasil, 2018) será utilizada como referência no presente trabalho a fim de fundamentar o suporte ao tratamento adequado de dados pessoais no âmbito nacional. Um glossário de termos comuns a esses normativos será apresentado na seção 3.3.1. Apesar do foco nesses dois normativos, ressalta-se que a arquitetura apresentada pode vir a ser adaptada a outras normas e regulamentos, sem prejuízos em suas características gerais de segurança.

1.3 Metodologia de desenvolvimento

A metodologia adotada nesta dissertação foi estruturada conforme o seguinte planejamento. Inicialmente, realizou-se um levantamento de artigos da literatura científica, com o objetivo de identificar trabalhos relacionados aos propósitos do estudo. Essa investigação foi organizada de modo a responder a questões de pesquisa relacionadas aos seguintes aspectos: plataformas *blockchain* utilizadas em trabalhos acadêmicos; mecanismos de armazenamento distribuído empregados para o compartilhamento de dados pessoais; técnicas criptográficas aplicadas à proteção desses dados; abordagens de computação em nuvem e seus principais pontos negativos; nível de atenção dispensado à conformidade com regulamentos de proteção de dados e à auditoria; e, por fim, pontos da GDPR e da LGPD não atendidos pelos trabalhos analisados.

Como parte da fundamentação teórica, foram sistematizados os principais conceitos previstos na GDPR e na LGPD, incluindo terminologia, princípios, direitos dos titulares e bases legais para o tratamento de dados pessoais, de modo a fornecer embasamento normativo à proposta desenvolvida.

Com base nessa fundamentação, foi concebida a arquitetura *Gateway for Data Privacy Auditing* (G4DPA), destinada ao compartilhamento de arquivos contendo dados pessoais entre diferentes instituições, em conformidade com as diretrizes estabelecidas pelas legislações mencionadas. A definição da arquitetura incluiu a identificação de entidades, a formulação de premissas, a delimitação de restrições e o desenvolvimento de um modelo de sistema, complementado por uma análise de segurança.

Posteriormente, a arquitetura foi implementada e descrita a partir de aspectos de infraestrutura, bibliotecas de *software*, mecanismos criptográficos e recursos de *hardware* disponíveis, considerando os desafios encontrados no desenvolvimento.

Por fim, para avaliar o custo e o desempenho da solução proposta, foram conduzidos quatro experimentos específicos, detalhando-se os procedimentos de testes e os resultados obtidos, que subsidiaram as análises finais.

1.4 Estrutura

No capítulo 2, descrevemos a estratégia de pesquisa bibliográfica adotada, com ênfase na metodologia utilizada para identificar, selecionar e revisar a literatura. Além disso,

para cada estudo analisado, são descritas as características relevantes em relação ao tema, e algumas tabelas comparativas são fornecidas a fim de facilitar a compreensão. No capítulo 3 são introduzidos conceitos fundamentais relacionados aos desafios inerentes ao tratamento de dados pessoais em ambientes computacionais, tais como *InterPlanetary File System* (IPFS), *blockchain* e medidas de segurança cibernética. Na seção 3.3, é feito um quadro comparativo com conceitos e equivalências entre a GDPR e a LGPD; além disso, a fim de familiarizar o leitor com essas legislações, fornecemos um resumo dos princípios e direitos previstos nesses normativos. A seguir, o capítulo 4 apresenta a solução proposta pelo presente trabalho, a arquitetura G4DPA, descrevendo, além de seu mecanismo de funcionamento, os objetivos de design, as entidades definidas, as premissas e restrições adotadas. Adicionalmente, são realizadas análises relacionadas aos aspectos de segurança e desempenho da arquitetura apresentada. A descrição da metodologia empregada nos experimentos realizados e a análise crítica dos resultados alcançados são apresentadas no capítulo 5. Por fim, no capítulo 6, são descritas as conclusões, sintetizando os principais achados e contribuições do trabalho apresentado, bem como as oportunidades de melhorias e trabalhos futuros.

2 TRABALHOS RELACIONADOS

O surgimento de leis específicas com regras para processamento de dados pessoais em vários países do mundo tem motivado pesquisas em técnicas e padrões capazes de prover conformidade legal para sistemas em diversas áreas, como *e-healthcare*, *Industrial Internet of Things* (IIoT) e *Internet of Vehicles* (IoV) (RUSSO *et al.*, 2018). A fim de avaliar o estado da arte acerca das estratégias utilizadas nesse contexto, foram investigados os desafios inerentes ao compartilhamento de dados pessoais, como tipo de armazenamento, controle de acesso e conformidade com os regulamentos legais vigentes. A metodologia adotada durante esse levantamento é descrita na seção 2.1. Já a seção 2.2 traz uma breve descrição dos trabalhos selecionados, destacando ainda suas principais contribuições e lacunas, visando direcionar futuros trabalhos que complementem a literatura.

2.1 Metodologia do levantamento bibliográfico

Embora o levantamento de trabalhos relacionados realizado neste estudo não tenha seguido todas as etapas de uma revisão sistemática conforme proposto por (KITCHENHAM; CHARTERS, 2007), ele foi inspirado nos procedimentos dessa abordagem. A metodologia empregada nesta investigação é descrita detalhadamente a seguir. Os principais fatores para a não adoção integral da sistemática foram:

a. Número limitado de participantes para a realização do processo de revisão;
b. A não utilização da técnica de *snowball* como ferramenta de descoberta de mais trabalhos, em virtude de limitação de tempo;
c. Uso de uma única base de dados.

A etapa inicial consistiu na definição dos objetivos da revisão de literatura. Este levantamento visou identificar o estado da arte em relação aos desafios enfrentados por sistemas computacionais no armazenamento, compartilhamento e garantia de privacidade no tratamento de dados pessoais, com ênfase na conformidade com regulamentos legais, como a GDPR e a LGPD. Adicionalmente, buscou-se investigar as tendências de pesquisa na área e identificar possíveis lacunas nos trabalhos existentes.

Em conformidade com essas considerações, foi elaborada a *string* de busca descrita na tabela 1 para a base de dados Scopus [1], com foco em artigos publicados a partir de 2018,

[1] https://www.scopus.com/

redigidos em inglês, no domínio da ciência da computação (`SUBJAREA(comp)`). Para eliminar materiais de mais difícil acesso, foi aplicado um filtro para artigos oriundos de anais de conferências (`SRCTYPE(p)`) ou de periódicos (`SRCTYPE(j)`), excluindo assim livros acadêmicos, publicações comerciais e relatórios técnicos. A instrução `TITLE-ABS-KEY` permite pesquisar textos específicos nos campos título, resumo e palavras-chave dos documentos. Mediante seu uso, foi possível definir que os artigos selecionados deveriam abordar a construção de uma arquitetura ou temas correlatos (plataforma, *framework*, protocolo ou padrão) e discutir questões relacionadas ao compartilhamento de dados ou informações e à proteção de dados (LGPD, GDPR, privacidade, dados pessoais ou dados sensíveis).

Como mencionado, a pesquisa foi restringida à base de dados Scopus, e isso ocorreu por duas razões. Primeiramente, o tempo disponível para a execução da pesquisa não permitiria a consulta a outros agregadores e bases de publicações científicas. Em segundo lugar, a Scopus é reconhecida como uma das mais extensas bases de dados de publicações científicas com curadoria, oferecendo ampla cobertura global e regional de revistas e periódicos, além de agregar resultados de várias outras bases e agregadores (BAAS *et al.*, 2020). Todavia, embora ela tenha sido a única base utilizada, a quantidade de resultados obtidos com as condições até aqui estabelecidas permaneceu elevada, sendo necessária uma filtragem adicional. Assim, estabelecemos que os artigos deveriam discutir aspectos da computação em nuvem, com interesse em avaliar as condições sob as quais esse paradigma é empregado ou, alternativamente, as razões para sua não utilização. Do mesmo modo, o interesse deste trabalho recai sobre os que utilizem *blockchain* e mecanismos de armazenamento distribuído (incluindo, por exemplo, tecnologias como IPFS e redes *Peer-to-Peer* (P2P)).

Um aspecto relevante da busca realizada é que ela não limitou o domínio da aplicação ou a plataforma de desenvolvimento, intencionando apurar as principais estratégias adotadas nos diferentes contextos de tratamento de dados pessoais. A tabela 2 detalha os critérios de inclusão e exclusão adotados para seleção dos trabalhos. Por sua vez, a tabela 3 apresenta a quantidade de estudos selecionados em cada fase de avaliação, a saber:

- **Fase I**: Quantidade inicial de trabalhos selecionados de acordo com a *string* de busca.
- **Fase II**: Quantidade de estudos restantes após análise do resumo.
- **Fase III**: Trabalhos selecionados após leitura completa.

Vale destacar que, apesar da *string* de busca ter retornado 69 trabalhos, apenas 65 estavam disponíveis para análise. Outro ponto relevante é que a reduzida quantidade de estudos

Tabela 1 – *String* de busca

```
PUBYEAR > 2018

AND LANGUAGE ( english )

AND SUBJAREA ( comp )

AND ( SRCTYPE ( p )
      OR SRCTYPE ( j ) )

AND ( TITLE-ABS-KEY ( architecture )
      OR TITLE-ABS-KEY ( platform )
      OR TITLE-ABS-KEY ( framework )
      OR TITLE-ABS-KEY ( protocol )
      OR TITLE-ABS-KEY ( standard ) )

AND ( TITLE-ABS-KEY ( sharing )
      OR TITLE-ABS-KEY ( shared ) )

AND ( TITLE-ABS-KEY ( lgpd )
      OR TITLE-ABS-KEY ( gdpr )
      OR TITLE-ABS-KEY ( privacy )
      OR TITLE-ABS-KEY ( “personal data” )
      OR TITLE-ABS-KEY ( “sensitive data” ) )

AND TITLE-ABS-KEY ( cloud )

AND TITLE-ABS-KEY ( blockchain )

AND ( TITLE-ABS-KEY ( “distributed storage” )
      OR TITLE-ABS-KEY ( ipfs )
      OR TITLE-ABS-KEY(“peer to peer”)
      OR TITLE-ABS-KEY(p2p) )
```

SUBJAREA (comp): restringe ao domínio da ciência da computação.
SRCTYPE (p): artigos oriundos de anais de conferências.
SRCTYPE (j): artigos oriundos de periódicos.
TITLE-ABS-KEY: pesquisa textos específicos nos campos título, resumo e palavras-chave dos documentos.
Fonte: elaborada pelo autor.

obtidos na fase I se deve às características da *string* de busca que, através do uso do operador AND, exige que cada trabalho retornado contenha pelo menos uma das palavras-chave esperadas para cada uma das áreas de interesse definidas no objetivo da revisão.

Tabela 2 – Critérios de inclusão e exclusão

Critérios de Inclusão	**Critérios de Exclusão**
Estudos que tratem pelo menos uma das áreas de interesse definidas nos objetivos da pesquisa, além do tópico privacidade.	Estudos que não detalhem aspectos importantes de suas soluções.
-	Estudos secundários.
-	Estudos que não guardem relação com compartilhamento seguro de dados pessoais ou seu armazenamento.
-	Estudos que apresentem inconsistências ou lacunas que impactem a compreensão da solução proposta.

Fonte: elaborada pelo autor.

A partir da tabela 3, observa-se que uma quantidade relativamente pequena de trabalhos foi excluída na Fase II. Isto pode ser atribuído ao grande interesse acadêmico e industrial na temática abordada, resultando em um grande número de publicações focadas nos aspectos relacionados à proteção de dados pessoais, bem como nas diversas metodologias e tecnologias aplicáveis a esse contexto, apesar de que apenas 33 dos trabalhos selecionados nesta fase realizaram algum nível de implementação ou experimentos sobre a solução proposta.

Tabela 3 – Quantidade de estudos por fase da revisão bibliográfica

Fase	**Quantidade de Estudos Selecionados**
I	65
II	55
III	37

Fonte: elaborada pelo autor.

Por fim, foi realizada a etapa de avaliação de qualidade dos trabalhos, que resultou na exclusão de 18 estudos, sendo esta a etapa final descrita em (KITCHENHAM; CHARTERS, 2007).

A partir do objetivo definido, foram determinadas as questões de pesquisa que o levantamento bibliográfico pretende responder:

RQ1: Quais são as plataformas *blockchain* utilizadas nos trabalhos e quais suas características?

RQ2: Quais mecanismos de armazenamento distribuído tem sido utilizados para o contexto de compartilhamento de dados?

RQ3: Quais mecanismos criptográficos tem sido utilizados para proteger dados pessoais compartilhados?

RQ4: Que abordagens de computação em nuvem tem sido utilizadas para compartilhamento de dados pessoais e quais pontos negativos tem sido elencados sobre essa plataforma?

RQ5: Entre os trabalhos que lidam com tratamento de dados pessoais, quantos mencionaram expressamente a necessidade de atender regulamentos legais aplicáveis sobre proteção de dados e quantos tratam de auditoria de operações?

RQ6: Se houver, quais são os principais pontos de não-conformidade com regulamentos de proteção de dados pessoais encontrados nos trabalhos, considerando a GDPR e a LGPD?

Em relação à RQ4, justifica-se esse questionamento ao observar o crescente protagonismo da computação em nuvem no cenário tecnológico contemporâneo, bem como os riscos inerentes ao tratamento de dados pessoais nesse ambiente. Tais fatores evidenciam a relevância de investigar as implicações do uso desse paradigma, como apresentado na seção 1.1.

2.2 Descrição dos Trabalhos

(NGUYEN *et al.*, 2019) utiliza uma *blockchain* Ethereum privada instanciada na Amazon Web Services (AWS), com foco em desenvolver uma solução utilizando Solidity e IPFS para armazenamento distribuído. Seu mecanismo de controle de acesso usa contratos inteligentes para realizar o compartilhamento seguro de *Electronic Health Record*s (EHRs) entre pacientes e provedores de saúde. Todavia, não há evidência de escalabilidade da solução e, apesar de registrar as transações, não aborda a remoção e atualização de dados pessoais.

(ULLAH *et al.*, 2022), por sua vez, realiza sua implementação na rede Ethereum Rinkeby, explorando o uso de Solidity como linguagem dos contratos inteligentes e IPFS para armazenar e compartilhar arquivos oriundos de dispositivos IoT, de modo distribuído e descentralizado. Entretanto, não há previsão de como revogar atributos dos usuários ou de como atualizar a política *Attribute-Based Access Control* (ABAC). Além disso, a implementação só levou em consideração os contratos inteligentes, e o valor estabelecido para o preço de *gas* em 2

Gwei não é realista. Por fim, o artigo menciona que há recompensa em criptomoedas Filecoin para os nós que armazenam os dados em IPFS, mas não aborda como isso é feito.

Também empregando uma *blockchain* Ethereum e o IPFS, (JABARULLA; LEE, 2021) propõe um projeto de *Proof of Concept* (PoC) para um sistema distribuído de gerenciamento de imagens médicas centrado no paciente, chamado de *Patient Centric Image Management* (PCIM), com o intuito de garantir a segurança e controle dos dados privados do paciente sem o uso de uma infraestrutura centralizada. Contudo, esse trabalho não previu um mecanismo para verificar se o requerente das imagens é quem realmente diz ser, e propõe bloquear o compartilhamento de um arquivo por um paciente se ele fizer isso várias vezes, a fim de evitar redundância no IPFS, o que pode causar graves prejuízos ao paciente.

(YEH *et al.*, 2022) propõe o *framework Blockchain-based Privacy-preserving and Sustainable Data Query Service* (BPSDQS) para proporcionar preservação de privacidade e acessibilidade estável dos dados em um cenário de um *Intelligent Transportation System* (ITS), sem depender de um servidor centralizado, fazendo uso de contratos inteligentes em Solidity, oráculos sobre uma *blockchain* Ethereum (utilizando a rede de testes Rinkeby) e IPFS. Além disso, emprega um mecanismo de *proxy-reencryption* e *searchable encryption* para proteger os dados e permitir que usuários possam buscar informações de tráfego sem expor suas palavras-chave de interesse. Entretanto, seu desempenho depende da disponibilidade do oráculo, que alegadamente estaria sempre online, e suas buscas são limitadas a apenas uma palavra-chave.

Ainda com foco no uso de Ethereum, Solidity e IPFS, porém com a proposta de prover uma política de controle de acesso granular e segura para atributos de dados de fala, (ZHANG; ZHAO, 2023) propõe um esquema de armazenamento distribuído em IPFS, no qual os dados são criptografados através de *Cyphertext-Policy Attribute Based Encryption* (CP-ABE) e os respectivos *Content IDentifier*s (CIDs) gerados são armazenados em um contrato inteligente Ethereum. Contudo, vale ressaltar que o funcionamento do mecanismo proposto depende de uma autoridade central honesta e confiável para inicializar os parâmetros, realizar o registro de usuários e distribuir as chaves privadas de atributos. Além disso, os pseudo-códigos dos contratos inteligentes especificados não deixam claro como as operações são realizadas.

Também com o intuito de oferecer controle granular de acesso, o estudo (ALOBADH *et al.*, 2021) propõe um esquema de autorização e controle de acesso criptografado chamado de *Smart Contract-based Attribute-Based Searchable Encryption* (SC-ABSE), integrando *Personal Health Record*s (PHRs) com *blockchain* e utilizando o IPFS para armazenamento desses registros.

Esse esquema combina CP-ABE, *Symmetric Searchable Encryption* (SSE), contratos inteligentes e IPFS. A proposta provê também um mecanismo de pesquisa de palavras-chave, proteção da privacidade dos usuários e auditoria de todas as solicitações e atividades de acesso. Todavia, este modelo não permite acesso emergencial aos arquivos dos pacientes nem prevê um mecanismo de atualização ou revogação de atributos no sistema e de exclusão de dados.

Já o artigo (HOANG *et al.*, 2020) propõe uma plataforma de compartilhamento de dados baseada no uso de uma rede *blockchain* pública e IPFS, visando proteger tanto a anonimidade de remetentes e destinatários quanto a confidencialidade dos dados em todas as atividades no sistema. A auditabilidade do acesso aos dados é garantida por meio de listas de controle de acesso ocultas, armazenadas na *blockchain* para verificação pública. O desempenho da solução foi analisado através do *deploy* dos três principais contratos inteligentes em uma rede de testes Ethereum, concluindo que o custo desses contratos é bem elevado por conta das pesadas operações realizadas *on-chain*. Além disso, algumas das operações propostas, como a assinatura em anel e o esquema de *Predicate Encryption* (PE) revogável, mostraram-se pesadas para computadores pessoais.

O artigo de (GHANI *et al.*, 2020) apresenta uma abordagem para compartilhamento de PHRs, realizando o controle de acesso através de contratos inteligentes e armazenando os arquivos utilizando IPFS, visando fornecer controle ao paciente sobre seus dados, garantir privacidade e melhorar escalabilidade. Os dados sensíveis são tratados com esteganografia para remover informações como nome, idade e endereço, sendo enviados para o IPFS pelo próprio paciente, identificado por seu endereço Ethereum. Porém, nem todos os mecanismos sugeridos são detalhados, e os PHRs, apesar de passarem por esteganografia, são armazenados às claras no IPFS. Não é previsto mecanismo de exclusão dos dados armazenados, mas apenas a revogação dos privilégios de acesso.

O trabalho de (RAJ; GHOSH, 2024) propõe um *framework* para armazenamento e compartilhamento de dados no contexto de IoT para cadeias de fornecimento, utilizando uma *blockchain* para integrar um mecanismo de controle de acesso baseado no modelo de segurança Bell-LaPadula, *proxy re-encryption* e IPFS. Para expressar as políticas de segurança multi-nível, todos os usuários e todos os objetos devem ser atribuídos a um, e apenas um, de quatro níveis de liberação (*clearance levels*) definidos: *top secret*, *secret*, *confidential* e *unclassified*, aqui em ordem do mais restrito para o menos restrito. Com base nesses níveis, é criado um conjunto de permissões através de uma matriz de controle de acesso, implementada diretamente em

um contrato inteligente. Ao receber uma requisição, o servidor *proxy* executa outro contrato inteligente para verificar se o solicitante pode acessar os dados. Em caso afirmativo, ele gera a chave de *re-encryption* com base nas chaves públicas das partes, com a qual recriptografa o arquivo e o envia ao requisitante. O IPFS é utilizado para armazenar os dados da cadeia de suprimentos de forma *off-site*, sem a necessidade de servidores em nuvem. Entretanto, o mecanismo depende de uma autoridade confiável para a inicialização do sistema criptográfico e para a geração das chaves dos usuários. A implementação do mecanismo de *proxy re-encryption* é feita em um contrato inteligente Ethereum, que, apesar de ser uma solução interessante, é muito cara. O mecanismo de revogação previsto apresenta um custo elevado, pois necessita revogar todos os usuários de uma mesma *role* e criptografar novamente todos os dados aos quais esses usuários tinham acesso. Por fim, devido ao baixo *throughput* alcançado com o uso da *blockchain* Ethereum, a solução pode apresentar problemas de escalabilidade.

O uso de *blockchain* Ethereum e IPFS também é adotado em (KUMAR; PRABHUDEVA, 2022), que propõe um *framework* de autorização de acesso e armazenamento de dados médicos de pacientes através de um algoritmo chamado *Proof-of-Replication* (PoR), implementado através de contratos inteligentes. Os dados são divididos em fragmentos (*shards*), cifrados com a chave pública do paciente e encaminhados através de uma requisição, por um contrato inteligente, a um nó de armazenamento, que os envia ao IPFS em meio de armazenamento próprio, sugerindo-se o uso de nuvem para isso. Entretanto, é importante observar que a proposta apresenta lacunas, como o detalhamento da distribuição das chaves assimétricas e do processo de decriptação do arquivo. Além disso, o artigo fixa a divisão do arquivo sempre em três partes, mesmo quando a banda disponível e o tamanho do arquivo não forem adequados a essa configuração, e o processo de revogação de chaves, realizado após cada acesso, é muito dispendioso. Também não é informado como funciona o mecanismo de controle de acesso através da *blockchain*.

O artigo de (ABOUALI *et al.*, 2021) propõe o *framework Secure Patient Health Records Sharing* (SPHRS), voltado ao compartilhamento seguro de registros de saúde, conferindo ao paciente acesso e controle sobre seus dados. O esquema utiliza contratos inteligentes de uma *blockchain* Ethereum para autenticação de usuários, IPFS para armazenamento de dados e protocolo NuCypher para proporcionar um mecanismo de gerenciamento de chaves e de *proxy re-encryption* baseado em *blockchain*. O acesso conferido a um provedor de saúde é feito por tempo determinado. Os arquivos são protegidos com criptografia simétrica, cuja chave

é criptografada pelo sistema de *proxy re-encryption*. Este trabalho, entretanto, não apresenta análise de custos computacionais, implementação, simulação ou testes de desempenho, e faz apenas uma breve discussão informal sobre a segurança do modelo proposto.

Também com foco no tratamento de dados de saúde, (KAUR *et al.*, 2024) apresenta uma abordagem distribuída para compartilhamento seguro de EHRs em larga escala, armazenando esses dados cifrados com uma política de acesso granular CP-ABE em IPFS, e utilizando contratos inteligentes Ethereum para registro de usuários. Cada usuário deve possuir um endereço Ethereum e realizar a gerência de permissões de acesso aos seus dados. As chaves do mecanismo CP-ABE são distribuídas por uma autoridade central confiável, e as funções de todos os contratos inteligentes disparam eventos, com o intuito de proporcionar rastreamento de acessos aos dados e trocar mensagens de requisição de acesso entre pacientes e visualizadores. Essas mensagens preveem a informação de um propósito de acesso, aproximando esta abordagem dos normativos legais. Os CIDs dos arquivos no IPFS são armazenados na *blockchain*, e só podem ser recuperados por usuários autorizados. Entretanto, não é mostrado no artigo como é realizada a distribuição segura das chaves criptográficas, nem foi abordada a possibilidade de exclusão de registros. Em se tratando de uma plataforma de dados de saúde, não há possibilidade de acessos emergenciais. Ademais, o esquema proposto utiliza CP-ABE para cifrar os arquivos inteiros diretamente, quando poderia ter melhor desempenho se cifrasse apenas uma chave simétrica.

O artigo de (SHREE *et al.*, 2024) apresenta uma arquitetura para armazenamento e compartilhamento de dados de *Internet of Medical Things* (IoMT), em que os dados são fragmentados com o uso do algoritmo de compartilhamento de segredos de Shamir – *Secret Sharing Algorithm* (SSA) – e os fragmentos são enviados para *clusters* IPFS separados em diferentes provedores de armazenamento em nuvem. Com isso, o comprometimento de provedores em um número inferior ao *threshold* utilizado no algoritmo não seria capaz de comprometer os dados originais, já que estes não poderiam ser reconstruídos. O registro de usuários e de dispositivos médicos inteligentes, bem como o armazenamento dos *hashes* dos fragmentos dos arquivos e o gerenciamento das permissões de acesso, é realizado através de contratos inteligentes. Os *uploads* e *downloads* de arquivos são processados por um componente central, que realiza a computação do SSA, os envios aos *clusters* IPFS e os registros dos endereços dos arquivos na *blockchain*. Como desvantagens, essa abordagem não se mostrou adequada a grandes arquivos, apresentando alto *overhead* em termos de latência de rede e altos custos das transações dos contratos inteligentes, existindo um *tradeoff* entre a sensibilidade ao tempo e a criticidade dos

dados.

Por sua vez, o trabalho de (YEH *et al.*, 2023) apresenta um mecanismo para compartilhamento de PHRs entre pacientes e médicos em conformidade com o direito ao esquecimento previsto pela GDPR, através da integração de uma *blockchain* editável (*redactable*) e uma modificação em um *cluster* IPFS que possibilita a exclusão de arquivos, proposta em um artigo anterior dos mesmos autores (YEH *et al.*, 2022). A *blockchain* editável faz uso de *hashes* camaleão para que blocos possam ser alterados sem alterar o *hash* anterior, empregando uma chave *trapdoor* previamente estabelecida. Na plataforma proposta, a confidencialidade e o controle de acesso dos médicos aos PHRs são desempenhados por um mecanismo de *proxy re-encryption*, elaborado de forma que não seja necessário que o proprietário dos dados esteja *online* o tempo inteiro. Para isso, um oráculo presumidamente "honesto mas curioso" realiza, executando em um enclave *Software Guard Extensions* (SGX), a geração de chaves de transformação em nome do paciente para o médico. Se um paciente deseja revogar a permissão anteriormente dada a um médico, uma entidade de autoridade confiável realizará a edição da *blockchain*. Do mesmo modo, se um paciente desejar excluir um de seus PHRs, ele realizará a alteração da *blockchain* e solicitará ao oráculo a exclusão dos dados do IPFS. Entretanto, essa proposta atende a GDPR apenas parcialmente, visto que não permite o acesso aos registros de saúde de um paciente em casos emergenciais, e a abordagem se mostrou cara em termos computacionais, principalmente em termos de tempo de mineração das transações.

O trabalho de (LI *et al.*, 2021) propõe a arquitetura *Blockchain-based Distributed Data Transaction* (BDDT) para IoT, fornecendo uma plataforma de negociação para produtores e consumidores de dados. Os usuários podem se beneficiar vendendo seus próprios dados, definindo antecipadamente suas próprias políticas de privacidade para determinar quais podem ser vendidos, enquanto os provedores de serviços compram esses dados para obter lucro através de análises. Todas as operações de transações de dados de IoT são registradas na *blockchain*, garantindo sua rastreabilidade e imutabilidade. O esquema emprega um *design* de dupla *blockchain*, sendo uma delas virtual, e um modelo de transação P2P é construído com contratos inteligentes. Nós de armazenamento fornecem seu próprio espaço em disco para garantir o armazenamento distribuído de dados e obtêm, por isso, uma parte do lucro das transações. Um contrato inteligente é acionado para verificar o pagamento em quantidade suficiente de criptomoeda para completar a transação de dados pelo nó consumidor. Todavia, os autores não entram no mérito da segurança criptográfica dos dados e preveem operações muito custosas para o contrato inteligente

do *framework*, deixando de abordar como elas seriam feitas de modo a minimizar o custo da solução.

O artigo de (GAO *et al.*, 2021) propõe um esquema de compartilhamento de dados centrado no usuário chamado *Blockchain-based security Sharing Scheme for Personal Data* (BSSPD), que combina *blockchain*, CP-ABE e IPFS para prover privacidade e controle de acesso granular, em que o proprietário dos dados (*data owner*) pode publicar seus dados compartilhados e definir uma política de acesso para eles, concedendo e revogando direitos de acesso aos usuários de dados (*data users*). O controle de acesso granular é alcançado através de um algoritmo CP-ABE modificado para possibilitar revogação de usuários por atributo. No esquema, o proprietário criptografa os dados compartilhados com uma chave simétrica aleatória e os salva no IPFS. Essa chave e o CID dos dados são então criptografados com os parâmetros públicos do sistema e com uma política de acesso embutida, de modo que apenas o usuário de dados cujos atributos correspondem a essa política de acesso pode baixar e decodificar a chave e o CID. A recuperação dos dados é realizada através de palavras-chave cifradas. Os rastros do proprietário e do usuário ficam armazenados na *blockchain*. Todavia, o proprietário precisa distribuir uma chave secreta para cada usuário e armazená-la na *blockchain* para o algoritmo de criptografia pesquisável, e é necessário manter grandes quantidades de índices para cada dado compartilhado, ocasionando *overhead* operacional.

(GUO *et al.*, 2021) também apresenta um *framework* de compartilhamento de dados que possibilita consultas por palavras-chave criptografadas, utilizando um esquema baseado em contratos inteligentes de uma *blockchain* e SSE com emparelhamentos bilineares, em que o proprietário de dados (*data owner*) pode incluir ou alterar novos dados e também revogar permissões. O serviço de autorização de consultas é delegado ao contrato inteligente após a primeira busca de um usuário de dados (*data user*) e, a partir desse momento, o proprietário não precisará estar *online* para fornecer novos *tokens* de busca. Um índice criptografado é construído pelo proprietário, que é armazenado junto com os documentos criptografados em um banco de dados distribuído. Para realizar consultas, um usuário de dados necessita obter sua chave de consulta através de um contrato inteligente e gerar um *token* de consulta para a palavra-chave que deseja buscar. Os proprietários mantêm uma tabela de permissões nesse contrato inteligente, de modo que podem revogar o acesso de um usuário a qualquer momento. Entretanto, a implementação realizada não abrangeu os contratos inteligentes e, com isso, não houve avaliação dos impactos dos mecanismos *on-chain* de autorização e troca de chaves.

(LI *et al.*, 2023b) apresenta um *framework* para armazenamento e compartilhamento de dados médicos de pacientes por instituições de saúde, batizado de MSNET. Sua abordagem faz uso de uma *blockchain Hyperledger Fabric* (HLF) para prover rastreabilidade e controle de acesso, e de uma rede de computação *edge* local, responsável por separar e sumarizar os dados pessoais dos pacientes e extrair *keywords* para buscas. Os dados que forem considerados pessoais são armazenados na *blockchain*, e os demais são armazenados nos servidores *edge*. Contratos inteligentes são utilizados para registrar a concordância das entidades participantes com as regras estabelecidas, controlar o compartilhamento de dados entre instituições e para aplicar um mecanismo de supervisão de uso da plataforma, incluindo punição de participantes por mau uso. É sugerido um mecanismo de gerenciamento desses contratos inteligentes através de um sistema de votação entre as instituições participantes, que podem propor alterações e votar na aprovação ou rejeição de mudanças. O Raft [2] é utilizado como mecanismo de consenso para a rede HLF. Entretanto, documentos de exames médicos não são tratados como dados pessoais, permanecendo sem criptografia ao serem armazenados, e a abordagem proposta apresenta problemas de escalabilidade em termos de espaço de armazenamento.

(TONG *et al.*, 2024) propõe a arquitetura *Blockchain-Based Data Sharing and Privacy Protection* (BBDSPP) para compartilhamento e proteção de dados em um ambiente IIoT, empregando uma abordagem criptográfica baseada em atributos combinada com um esquema *Weighted Threshold Secret Sharing* (WTSS). A autenticação é realizada por meio de *Zero Knowledge Proof*s (ZKPs) não interativas sobre os valores dos atributos. Os dados são criptografados com uma chave simétrica aleatória e armazenados em uma rede IPFS, enquanto o CID correspondente e a chave simétrica utilizada – cifrada através de um mecanismo *Attribute-Based Encryption* (ABE) – são armazenados em uma *blockchain*. Uma autoridade de atributos e uma autoridade de certificados, ambas confiáveis, interagem entre si para emitir os certificados dos usuários e especificar os seus atributos. A senha simétrica só pode ser decifrada com sucesso se os pesos dos atributos de um usuário forem maiores que um valor limite pré-estabelecido. Os autores apresentaram provas formais de corretude, autenticação, anonimidade e funcionamento do mecanismo WTSS. Entretanto, a proposta não possibilita a adição de novos atributos dinamicamente. A solução depende de autoridades confiáveis e da interação *online* entre o proprietário dos dados (*data owner*) e o visitante dos dados (*data visitor*) para funcionamento do compartilhamento. Por fim, o mecanismo de autenticação baseado em ZKPs apresenta um

[2] https://raft.github.io/

elevado custo computacional, podendo acarretar problemas de escalabilidade, e o esquema criptográfico apresenta menor eficiência quando o número de atributos é elevado.

O estudo de (HUANG *et al.*, 2024) propõe um mecanismo para compartilhamento de *Electronic Medical Record*s (EMRs) utilizando *proxy re-encryption* e uma *blockchain* HLF. Todas as mensagens trocadas recebem um *timestamp* e são assinadas por meio de *Elliptic Curve Digital Signature Algorithm* (ECDSA), de forma a garantir integridade, autenticação mútua, não-repúdio e proteger contra ataques *replay* e *sybil*. Os registros médicos são armazenados criptografados no IPFS por uma entidade chamada nuvem médica, que agrega hospitais e farmácias através da rede HLF. Todas as trocas de mensagens entre os participantes são registradas na *blockchain*, conferindo rastreabilidade à abordagem. Os pacientes são incentivados a compartilhar seus dados através do recebimento de uma criptomoeda chamada MCoin. Todavia, na abordagem proposta, os pacientes não têm soberania sobre os seus dados, pois dependem de que uma instituição médica revogue os acessos a terceiros. Quanto ao mecanismo de incentivo, apenas os pacientes são recompensados e não há previsão de como eles poderiam utilizar as criptomoedas adquiridas.

Visando solucionar problemas de compartilhamento de dados e proteção de privacidade decorrentes do rápido crescimento de dados na IIoT, (CHEN *et al.*, 2022) propõe uma arquitetura baseada na *blockchain* HLF, mas com armazenamento de dados *off-chain* para melhorar a escalabilidade do sistema. Todavia, a implementação aparenta ter sido apenas parcial, realizando o *deploy* da rede HLF e a execução dos dois contratos inteligentes projetados em Chaincode. Apesar de haver estimativas de latência, não há informação de como esses valores foram obtidos. Também não fica claro como é feita a gerência das chaves simétricas, cujo comprometimento afeta potencialmente todos os dados com elas cifrados.

Um *framework* modular para compartilhamento de EHRs com preservação de privacidade, chamado HealthBlock, é proposto em (ABDELGALIL; MEJRI, 2023). Pacientes, provedores de saúde (e.g. laboratórios) e consumidores (e.g. médicos) são identificados através de *Self-Sovereign Identity* (SSI) utilizando *Hyperledger Indy* (HLI), associando credenciais a uma chave pública do paciente e registros de saúde à sua *wallet*. Para viabilizar a delegação de acesso aos registros, é utilizada uma *blockchain* HLF, na qual o *Decentralized Identifier* (DID) e um conjunto de atributos do paciente são vinculados às chaves públicas dos consumidores delegados, por meio da assinatura digital do paciente. Os EHRs são armazenados em IPFS. Não é apresentada uma análise da segurança ou do desempenho computacional do protótipo construído. Também não é definido quem seria responsável por manter as duas *blockchains*

privadas, e não são abordados os problemas que ocorreriam no caso de comprometimento da chave privada de um consumidor. Em casos de emergência médica, seria necessário um terceiro nomeado para acesso aos dados para atender a exigências de leis de proteção de dados.

No domínio de *Vehicular Ad-hoc Network*s (VANETs), o artigo de (DWIVEDI *et al.*, 2021) apresenta uma arquitetura descentralizada baseada em *blockchain*, para compartilhar, de forma segura, informações de eventos entre *Roadside Unit*s (RSUs) sem depender de terceiros confiáveis, como servidores em nuvem. Para tanto, são utilizados o IPFS para armazenamento de eventos e um modelo próprio de *blockchain* para autenticação de veículos e RSUs. Um contrato inteligente em Ethereum é utilizado para recuperação de eventos a partir do IPFS. Entretanto, os eventos são armazenados sem criptografia, e a *blockchain* própria não foi implementada.

A arquitetura *Three-tier architecture Blockchain* (TBchain) de (LI *et al.*, 2023a) propõe uma rede *blockchain* em três camadas, com o intuito de possibilitar o armazenamento local de grandes quantidades de dados provenientes de dispositivos IoT de forma segura. As camadas estão organizadas de modo hierárquico, sendo que as transações enviadas para a TBchain passam em sequência por duas camadas de *blockchains* locais. Para alcançar imutabilidade, a última destas envia seus *hashes* para a terceira camada, correspondente à rede Ethereum. Portanto, a TBchain não pode operar como uma rede isolada, pois prevê a validação de suas transações em uma *blockchain* pública. Usuários e dispositivos IoT devem ser registrados na TBchain previamente às suas interações, com controle de acesso *Rule-Based Access Control* (RBAC), e cada usuário na TBchain deve manter sua própria *blockchain* local. Adicionalmente, o trabalho propõe uma arquitetura chamada *Blockchain and IPFS as a Service for IoT* (BI-IoT), que emprega o IPFS para armazenar os dados dos dispositivos IoT, aliviando a pressão de armazenamento sobre a *blockchain* e proporcionando integridade aos dados. Entretanto, a abordagem apresentada realiza o armazenamento dos dados às claras, ficando os arquivos acessíveis a qualquer um com o seu endereço. Não há previsão de exclusão de arquivos, e as camadas locais da TBchain acabam por dificultar o compartilhamento de dados.

O *framework* CareBlock é apresentado por (UPPAL *et al.*, 2021) para um contexto de IoMT, propondo a adoção de quatro *blockchains* distintas com o intuito de orquestrar um fluxo de trabalho abrangendo diagnóstico, tratamento e custeio de despesas de saúde. Essas *blockchains* são designadas para as seguintes quatro entidades: pacientes, médicos, farmacêuticos e seguradoras. Os dados de saúde provenientes de dispositivos IoMT são cifrados pelo paciente utilizando uma chave simétrica e armazenados no IPFS, enquanto o correspondente CID e

aquela chave simétrica, agora criptografada com a chave pública do profissional autorizado a acessar os dados, são registrados na *blockchain* correspondente ao tipo desse profissional. A proposta contempla o uso de uma criptomoeda denominada PlusCoin, visando viabilizar transações financeiras entre pacientes e demais intervenientes, com base em interações entre as quatro *blockchains*. Entretanto, não foi apresentada nenhuma análise de segurança do sistema proposto, observando-se também lacunas na descrição dos procedimentos de distribuição de chaves assimétricas, de revogação de acessos e de exclusão de arquivos armazenados, o que é importante considerando que os profissionais ainda detêm as chaves simétricas para decifrá-los.

(OH *et al.*, 2022) apresenta um sistema para armazenamento e compartilhamento seguros de PHRs baseado em *blockchain*, IPFS, *Key Aggregate Searchable Encryption* (KASE) e *Linear Secret Sharing Scheme* (LSSS), visando garantir que os pacientes tenham plena autoridade sobre o uso de seus registros. Ao contrário dos esquemas KASE existentes, o proposto neste trabalho possibilita, graças ao LSSS, a geração de *trapdoors* para buscas com várias palavras-chave simultaneamente, reduzindo o custo computacional quando é necessário buscar por mais de uma palavra-chave. O paciente envia seus PHRs criptografados ao IPFS, e, através de contratos inteligentes, registra o CID de seus dados e as palavras-chave associadas na *blockchain*. Como exposto, a abordagem depende de o paciente fornecer uma chave de busca agregada, o que não é conveniente em casos de emergência médica.

O trabalho de (MITTAL; GHOSH, 2023) propõe um *framework* para compartilhar dados de saúde de pacientes para um grupo de médicos que realizarão seu tratamento em conjunto. A abordagem consiste em três fases: o registro dos usuários e o armazenamento dos dados, realizados pelo paciente; a requisição e o acesso aos dados, feitos por um médico do grupo; e a comunicação dos dados entre os membros do grupo médico. Essa comunicação ocorre utilizando um mecanismo criptográfico descentralizado para gerar uma chave privada comum para o grupo. O controle de acesso aos dados é mantido através de um mecanismo de *proxy re-encryption*, em que os dados que se encontram criptografados com a chave pública do paciente passarão por uma série de operações matemáticas, de modo que o resultado possa ser, então, decifrado pela chave privada do médico requisitante. Os dados são armazenados no IPFS e os respectivos CIDs em uma *blockchain*. Todavia, a geração da chave de grupo apresenta um *overhead* computacional considerável, podendo não ser aceitável a partir de um certo número de membros, especialmente se realizada por dispositivos com recursos limitados, como IoT.

Um mecanismo para alcançar privacidade, descentralização e balanceamento de

carga em um sistema de armazenamento de dados é apresentado por (WEN; WANG, 2023), objetivando proteger a integridade dos dados e reduzir o tempo consumido em seu processo de recuperação. Um modelo de avaliação de pontuação de confiança de nós, chamado de *Trusted-Scoring Mechanism* (TSM), foi desenvolvido para reduzir as taxas de erro de acesso causadas por *hosts* de armazenamento anormais (defeituosos ou maliciosos) ou por falhas na transmissão, considerando que a falha na recuperação de qualquer segmento de arquivo piora o tempo de resposta e que um pequeno número de segmentos resulta em um tamanho maior de cada segmento de arquivo, exigindo um tempo de transmissão mais longo. A pontuação de confiança é calculada com base no status de retorno de segmentos de arquivo e estatísticas maliciosas compartilhadas por outros usuários através da *blockchain*. Já a solução de privacidade envolve armazenar múltiplos segmentos de arquivos duplicados em *hosts* não confiáveis usando tecnologias de *blockchain* e criptografia, adotando-se um mecanismo de consenso *Proof-of-Stake* (PoS) na abordagem proposta. Todavia, o trabalho não aborda o compartilhamento de dados, e os autores não apresentaram nenhuma análise de segurança. Os números considerados em uma simulação são consideravelmente menores do que os do mundo real, e não foram levados em conta os atrasos decorrentes do processamento e enfileiramento na passagem de uma rede para outra, com custos de comunicação elevados.

(SHARMA *et al.*, 2019) propõe uma arquitetura baseada em *blockchain* para IoV, com o objetivo de compartilhar dados entre veículos conectados na rede através de um ambiente confiável, destacando a possibilidade do uso da natureza das *blockchains* para transações seguras de mensagens das aplicações IoV. Essa arquitetura é formada por quatro camadas: a camada conceitual, a camada de convolução, a camada de controle e a camada de utilidade. O esquema propõe verificar e validar cada transação de dados dos sensores humanos, armazenando-as em blocos. Entretanto, não são apresentados maiores detalhes sobre o armazenamento dos dados em nuvem nem sobre os mecanismos de criptografia necessários à privacidade das informações.

Visando compartilhar dados de rotas de viagem da população a fim de conter a disseminação de COVID-19, mas garantindo a privacidade das pessoas, (LIU; WANG, 2023) usa a estrutura de cadeia federada do HLF combinada com o modelo de armazenamento do IPFS. Enquanto o texto cifrado dos dados completos da trajetória de viagem é armazenado no IPFS, a *blockchain* mantém um índice criptografado que consiste na chave *Advanced Encryption Standard* (AES) usada para criptografar esses dados, o endereço CID retornado pelo IPFS, a localização e o *timestamp* de criação do arquivo. Porém, o esquema necessita de melhorias na

taxa de transferência quando da transmissão de dados em grande escala.

(HEBBALLI *et al.*, 2023) propõe um sistema descentralizado para o gerenciamento de registros médicos, integrando as tecnologias *blockchain*, IPFS e criptografia Rivest-Shamir-Adleman (RSA). No modelo apresentado, os pacientes mantêm o controle sobre o acesso aos seus dados por meio de chaves privadas, enquanto os profissionais de saúde são responsáveis por gerar registros criptografados, que são armazenados na *blockchain* e referenciados via IPFS. O acesso por parte das seguradoras é viabilizado por meio de um mecanismo de *proxy re-encryption*, permitindo a reatribuição do acesso sem a exposição das chaves privadas dos pacientes. Para lidar com situações excepcionais, como o falecimento do paciente, o sistema adota um processo de validação baseado em *One-Time Password* (OTP). No entanto, a solução não contempla a implementação de contratos inteligentes e não discute a viabilidade de armazenar os registros diretamente na *blockchain*. A análise de desempenho foi limitada à comparação entre os algoritmos RSA e AES. O modelo proposto não demonstra compatibilidade com legislações de proteção de dados pessoais, uma vez que não contempla mecanismos para o compartilhamento de informações em casos de emergência médica, tampouco trata da exclusão ou retificação de dados.

Também na área de *e-health*, (ZHENG *et al.*, 2023) propõe um esquema de compartilhamento de EMRs baseado em endereços furtivos de *blockchain* para prevenir a correlação entre transações, cuja análise poderia comprometer a privacidade dos pacientes através de técnicas de *clustering*. Os dados dos EMRs são dessensibilizados e armazenados no IPFS, com seus respectivos CIDs mantidos na *blockchain*. O sistema de endereçamento furtivo utiliza emparelhamentos bilineares baseados em identidade, eliminando a complexidade do gerenciamento de certificados típica de uma *Public Key Infrastructure* (PKI) tradicional. Os autores fornecem uma prova formal de segurança para esse mecanismo de endereçamento. A implementação utilizando *Java Pairing-Based Cryptography* (JPBC) demonstrou que o desempenho do esquema proposto foi inferior ao da tecnologia CryptoNote [3], que também visa ocultar a identidade dos participantes em transações de *blockchain*. Contudo, o estudo limitou-se à avaliação do sistema de endereços furtivos, não implementando os contratos inteligentes nem os mecanismos de armazenamento no IPFS. Além disso, o sistema não prevê auditoria, acesso a dados em emergências médicas, nem a possibilidade de apagamento ou alteração de dados pessoais.

No artigo (KAVITHA *et al.*, 2022), os autores propõem a Stag-chain, uma arquitetura

[3] https://bytecoin.org/old/whitepaper.pdf

baseada em *blockchain* Ethereum. Ela integra uma técnica esteganográfica *Least Significant Bit* (LSB), detalhada por (BANDEKAR; SUGUNA, 2018), com o algoritmo AES e armazenamento descentralizado via IPFS para garantir comunicação segura. Para controlar a disponibilidade do arquivo esteganográfico, ele é publicado no IPFS por um período predefinido, com armazenamento do CID correspondente na *blockchain*. Após a expiração desse período, o arquivo é substituído por uma imagem comum. A implementação da arquitetura utilizou Ganache como cliente Ethereum e, embora os testes tenham mostrado que as imagens resultantes são de alta qualidade, a Stag-chain não incorpora mecanismos específicos de auditoria de operações. Além disso, o estudo não avalia aspectos de desempenho importantes como o consumo de gas dos contratos inteligentes ou o desempenho das operações no IPFS.

Em (MAINUDDIN *et al.*, 2023), os autores propõem o MedVault Nexus, um *framework* centrado no paciente para o compartilhamento eficiente e seguro de EHRs com médicos e instituições de saúde. Sua estrutura utiliza *blockchain* Ethereum, IPFS e contratos inteligentes em Solidity. Os registros médicos dos pacientes são primeiramente criptografados com os algoritmos AES e RSA e, em seguida, armazenados no IPFS. Seus CIDs e metadados correspondentes são, então, registrados na *blockchain*. O serviço Filecoin é empregado para garantir o armazenamento permanente dos dados no IPFS. O paciente compartilha seus registros com médicos e hospitais por um período predeterminado, após o qual o acesso é automaticamente revogado. As principais funcionalidades – autenticação, armazenamento, compartilhamento e acesso a dados – foram implementadas em uma aplicação *web* desenvolvida em ReactJS, e a segurança dos contratos inteligentes foi validada pela ferramenta MythX. No entanto, os testes de desempenho se limitaram aos tempos de criptografia e descriptografia, e não há auditoria direta das operações realizadas no sistema. O *framework* também não prevê a exclusão de arquivos do IPFS por solicitação do paciente, nem a possibilidade de acessos emergenciais aos dados.

Em (SINGH; RATHEE, 2023), o autor propõe um modelo de armazenamento descentralizado através de uma solução de *blockchain* Ethereum e IPFS. O fluxo operacional envolve o *upload* de arquivos para o IPFS, gerando CIDs que são armazenados em contratos inteligentes desenvolvidos em Solidity. A implementação foi testada com Ganache como *blockchain* local, e para interagir com os contratos inteligentes foi utilizado o MetaMask e uma interface *web*. Entretanto, o estudo carece de avaliações quantitativas de desempenho, análise de segurança formal, consumo de gas ou conformidade com regulamentações de proteção de dados. O autor reconhece as limitações relacionadas à revogação de atributos de usuários e à

atualização de políticas de acesso como trabalhos futuros.

Em (MALLICK *et al.*, 2024), os autores propõem um *framework* descentralizado para *Healthcare* 4.0, utilizando IoMT, *blockchain* Ethereum, computação em névoa e IPFS para mitigar as limitações dos sistemas de saúde centralizados, como alta latência, pontos únicos de falha e elevados requisitos de largura de banda. A arquitetura utiliza nós *fog* para processamento próximo aos dispositivos IoMT, o que reduz o congestionamento da rede e a sobrecarga na *blockchain* principal. Para dispositivos não confiáveis, é usado um monitor *proxy* que registra e controla suas atividades. Os dados de saúde são processados localmente pelos nós *fog*, em seguida criptografados e têm seus *hashes* armazenados na *blockchain*, enquanto os arquivos completos são enviados ao IPFS. A solução emprega contratos inteligentes em Solidity para automação de verificações e controle de acesso, utilizando assinatura digital com o algoritmo ECDSA. Foi realizada uma avaliação experimental que mostrou bom desempenho em tempos de *upload* e *download* em comparação com modelos existentes baseados em armazenamento centralizado e *blockchain* em nuvem. Contudo, os testes foram realizados em um ambiente controlado com recursos limitados (três nós *fog* e oito computadores), o que pode resultar em um comportamento diferente em implementações reais de grande escala.

Visando a preservação da privacidade em sistemas de armazenamento em nuvem, especialmente para dados educacionais, (JAIN; JAILIA, 2022) propõe um mecanismo que integra IPFS com a tecnologia *blockchain* para proporcionar um compartilhamento de dados eficiente entre proprietários de dados em nuvem e requisitantes. A arquitetura proposta é dividida em seis fases: configuração; registro de usuário; inicialização do contrato inteligente; armazenamento de dados criptografados; verificação de dados; e compartilhamento de dados baseado em *blockchain*. O IPFS é utilizado para armazenar os dados de forma distribuída, enquanto a *blockchain*, implementada com Ethereum em uma configuração privada/permissionada, gerencia o registro de arquivos criptografados e as transações de acesso por meio de contratos inteligentes. Criptografia de chave pública e *hashing* são usados para garantir segurança e integridade das informações. Embora o trabalho aponte benefícios na segurança, não aborda explicitamente bases legais de proteção de dados pessoais nem trata de como os dados podem ser excluídos. Não são especificados detalhes dos contratos inteligentes ou uma avaliação do consumo de gas.

2.3 Análise dos Trabalhos

A leitura aprofundada desses trabalhos permitiu identificar tendências e lacunas no contexto de privacidade de dados. A fim de facilitar a exposição desses resultados, foram elaboradas tabelas de apoio que sintetizam os dados extraídos dos estudos selecionados relativos a cada uma das questões de pesquisa. Em cada tabela apresentada, apenas os trabalhos que abordam as características relevantes ao contexto analisado são listados.

Dessa forma, para responder às questões de pesquisa **RQ1** e **RQ2** e ainda prover uma visão resumida dessas análises, foram elaboradas as tabelas 4 e 5. A tabela 4 apresenta um resumo das características gerais das soluções propostas, na qual foram avaliados pontos relativos à plataforma *blockchain* escolhida: i) se o tipo de acesso à rede é público, privado ou híbrido (ou seja, uma combinação de ao menos uma rede pública e ao menos uma rede privada), e ii) se a modalidade de acesso é com ou sem permissão. Já a tabela 5 relaciona o mecanismo de armazenamento utilizado na arquitetura e o tipo de acesso aos dados adotado.

A partir da tabela análise das tabelas 4 e 5, observa-se amplo uso de *blockchain*, em especial, Ethereum e HLF, como plataforma de desenvolvimento, combinado ao uso de IPFS como mecanismo de armazenamento e compartilhamento dos dados.

Dentre os principais argumentos relatados para o uso de *blockchain* destacam-se:

- o apoio ao gerenciamento de chaves e à administração de concessão de permissões de forma descentralizada;
- a rastreabilidade das operações;
- a disponibilidade, pois possibilita a execução de tarefas mesmo em caso de falhas de alguns nós;
- a imutabilidade;
- a segurança, possibilitando, inclusive, mecanismos de punição por mau comportamento na rede.

Além disso, a natureza distribuída e imutável de uma *blockchain* protege contra adulterações, enquanto o mecanismo de consenso aumenta a confiabilidade dos dados.

Segundo os estudos analisados, a combinação de *blockchain* com o uso do IPFS reduz a pressão sobre o armazenamento da *blockchain* — ou "*blockchain bloating*" —, aumentando, assim, a escalabilidade, e se mostra como alternativa viável ao armazenamento em nuvem, oferecendo um meio *offchain* seguro com redundância, tolerância a falhas e integridade criptográfica. O armazenamento de dados no IPFS também reduz custos e redundância, evitando pontos

Tabela 4 – Plataformas *blockchain* utilizadas nos trabalhos relacionados

Referência	Plataformas	Tipo de Rede	Tipo de Acesso
(ABOUALI *et al.*, 2021) (ALOBADH *et al.*, 2021) (GHANI *et al.*, 2020) (HOANG *et al.*, 2020) (JABARULLA; LEE, 2021) (KAVITHA *et al.*, 2022) (RAJ; GHOSH, 2024) (SHREE *et al.*, 2024) (ULLAH *et al.*, 2022) (YEH *et al.*, 2022) (ZHANG; ZHAO, 2023)	Ethereum	Pública	*Permissionless*
(JAIN; JAILIA, 2022) (KAUR *et al.*, 2024) (KUMAR *et al.*, 2022) (MAINUDDIN *et al.*, 2023) (MALLICK *et al.*, 2024)	Ethereum	Pública	*Permissioned*
(LI *et al.*, 2021)	Ethereum; Blockstack	Pública	*Permissionless*
(LI *et al.*, 2023a)	Ethereum; TBchain (conceitual)	Pública Privada	*Permissionless* *Permissioned*
(NGUYEN *et al.*, 2019)	Ethereum	Privada	*Permitioned*
(YEH *et al.*, 2023)	Ethereum; Bazo	Consórcio	*Permissioned*
(DWIVEDI *et al.*, 2021)	Ethereum; Conceitual	Pública Privada	*Permitionless*
(GAO *et al.*, 2021)	EOS	Pública	*Permissioned*
(ABDELGALIL; MEJRI, 2023)	HLF; HLI	Consórcio	*Permissioned*
(CHEN *et al.*, 2022) (HUANG *et al.*, 2024) (LI *et al.*, 2023a) (LIU; WANG, 2023) (TONG *et al.*, 2024)	HLF	Consórcio	*Permissioned*
(MITTAL; GHOSH, 2023) (WEN; WANG, 2023)	n/e	Consórcio	*Permissioned*
(UPPAL *et al.*, 2021)	Conceitual	Híbrida	*Permissioned*
(OH *et al.*, 2022)	Conceitual	Pública	*Permissionless*
(GUO *et al.*, 2021)	n/e	Pública	n/e
(SINGH; RATHEE, 2023) (ZHENG *et al.*, 2023)	n/e	n/e	*Permissioned*
(SHARMA *et al.*, 2019) (HEBBALLI *et al.*, 2023)	n/e	n/e	n/e

n/e: não especificado.
Fonte: elaborada pelo autor.

Tabela 5 – Mecanismos de armazenamento distribuído utilizados nos trabalhos relacionados

Referência	Mecanismo	Tipo de Acesso
(ABDELGALIL; MEJRI, 2023) (ABOUALI *et al.*, 2021) (ALOBADH *et al.*, 2021) (CHEN *et al.*, 2022) (DWIVEDI *et al.*, 2021) (GAO *et al.*, 2021) (GHANI *et al.*, 2020) (HEBBALLI *et al.*, 2023) (JABARULLA; LEE, 2021) (JAIN; JAILIA, 2022) (KAUR *et al.*, 2024) (MALLICK *et al.*, 2024) (MITTAL; GHOSH, 2023) (OH *et al.*, 2022) (RAJ; GHOSH, 2024) (SHREE *et al.*, 2024) (SINGH; RATHEE, 2023) (TONG *et al.*, 2024) (ULLAH *et al.*, 2022) (UPPAL *et al.*, 2021) (YEH *et al.*, 2022) (YEH *et al.*, 2023) (ZHANG; ZHAO, 2023) (ZHENG *et al.*, 2023)	IPFS	Público
(LIU; WANG, 2023) (NGUYEN *et al.*, 2019)	IPFS	Privado
(HUANG *et al.*, 2024) (KAVITHA *et al.*, 2022) (LI *et al.*, 2023a) (MAINUDDIN *et al.*, 2023) (YANG *et al.*, 2022)	IPFS; *Blockchain*	Público
(GUO *et al.*, 2021)	Banco de dados distribuído	Privado
(HOANG *et al.*, 2020)	Banco de dados descentralizado	Privado
(WEN; WANG, 2023) (KUMAR *et al.*, 2022) (LI *et al.*, 2023b) (LI *et al.*, 2021)	Nós descentralizados	Público
(SHARMA *et al.*, 2019)	Não	Não

Fonte: elaborada pelo autor.

Tabela 6 – Mecanismos criptográficos utilizados nos trabalhos relacionados para proteção de dados pessoais

Referência	Chave assimétrica	Chave simétrica	Assinatura digital	*Hash*	*Secret sharing*	*Proxy re-encryption*	ABE	Outros
(ABDELGALIL; MEJRI, 2023)	Sim (n/e)	Sim (n/e)	✗	✗	✗	✗	✗	
(ABOUALI *et al.*, 2021)	✗	Sim (n/e)	✗	✗	✗	Sim (nuCypher)	✗	
(ALOBADH *et al.*, 2021)	✗	AES	✗	✗	✗	✗	CP-ABE	SSE
(CHEN *et al.*, 2022)	ECC	AES	ECDSA	✗	✗	✗	✗	
(DWIVEDI *et al.*, 2021)	Sim (n/e)	Sim (n/e)	✗	✗	✗	✗	✗	
(GAO *et al.*, 2021)	✗	✗	✗	✗	✗	✗	CP-ABE	
(GHANI *et al.*, 2020)	✗	✗	✗	✗	SSS	✗	✗	Esteganografia
(GUO *et al.*, 2021)	✗	AES HMAC	✗	✗	✗	✗	✗	
(HEBBALLI *et al.*, 2023)	RSA	✗	✗	SHA-256	✗	Sim	✗	
(HOANG *et al.*, 2020)	ECC	✗	Em anel	✗	✗	Sim	CP-ABE	
(HUANG *et al.*, 2024)	Sim (n/e)	✗	ECDSA	Sim (n/e)	✗	Sim	✗	
(JABARULLA; LEE, 2021)	OpenPGP	✗	✗	✗	✗	✗	✗	
(JAIN; JAILIA, 2022)	Sim (n/e)	✗	✗	Sim (n/e)	✗	✗	✗	
(KAUR *et al.*, 2024)	✗	✗	✗	✗	✗	✗	CP-ABE (charm-crypto tool)	
(KAVITHA *et al.*, 2022)	✗	AES	✗	SHA-256	✗	✗	✗	Esteganografia
(KUMAR *et al.*, 2022)	Sim (n/e)	✗	✗	MD5	✗	✗	✗	
(LI *et al.*, 2021)	✗	Sim (n/e)	✗	✗	✗	✗	✗	
(LI *et al.*, 2023a)	✗	✗	✗	SHA-256	✗	✗	✗	
(LI *et al.*, 2023b)	Sim (n/e)	Sim (n/e)	✗	Sim (n/e)	✗	✗	✗	
(LIU; WANG, 2023)	✗	AES	✗	Sim (n/e)	✗	✗	✗	
(MAINUDDIN *et al.*, 2023)	RSA	AES	✗	SHA-256	✗	✗	✗	
(MALLICK *et al.*, 2024)	ECC	✗	ECDSA	Sim (n/e)	✗	✗	✗	
(MITTAL; GHOSH, 2023)	RSA	✗	ECDSA	✗	✗	Sim	✗	
(NGUYEN *et al.*, 2019)	RSA	✗	✗	✗	✗	✗	✗	
(OH *et al.*, 2022)	✗	✗	✗	Sim (n/e)	LSSS	✗	✗	KASE
(RAJ; GHOSH, 2024)	Sim (n/e)	✗	✗	Sim (n/e)	✗	Sim	✗	
(SHARMA *et al.*, 2019)	Sim (n/e)	✗	✗	✗	✗	✗	✗	
(SHREE *et al.*, 2024)	Sim (n/e)	✗	✗	✗	SSS	✗	✗	
(SINGH; RATHEE, 2023)	✗	✗	✗	✗	Diffie-Hellman	✗	CP-ABE	
(TONG *et al.*, 2024)	✗	Sim (n/e)	✗	Sim (n/e)	WTSS	✗	Sim (n/e)	ZKP
(ULLAH *et al.*, 2022)	Sim (n/e)	AES	✗	Sim (n/e)	ECC Diffie-Hellman	✗	✗	
(UPPAL *et al.*, 2021)	Sim (n/e)	Sim (n/e)	✗	✗	✗	✗	✗	
(WEN; WANG, 2023)	Sim (n/e)	✗	✗	Sim (n/e)	✗	✗	✗	
(YEH *et al.*, 2022)	ECC	AES	✗	✗	✗	Sim	✗	
(YEH *et al.*, 2023)	✗	✗	✗	*Chameleon hash*	✗	Sim	✗	
(ZHANG; ZHAO, 2023)	✗	✗	✗	✗	✗	✗	CP-ABE (cpabe toolkit)	
(ZHENG *et al.*, 2023)	Sim (n/e)	Sim (n/e)	✗	✗	✗	✗	✗	

n/e: não especifica.
✗: não utiliza.
Fonte: elaborada pelo autor.

únicos de falha e promovendo a descentralização. Ademais, o IPFS melhora a disponibilidade dos dados, que podem ser acessados inclusive *offline*, e garante a integridade dos registros sem depender de terceiros centralizados.

A partir dos estudos realizados, observa-se que, embora *blockchains* apresentem um *overhead* significativo devido à necessidade de armazenamento da cadeia em diversos nós, sua combinação com IPFS possibilita bom suporte ao armazenamento massivo de dados, atendendo a rigorosos requisitos de segurança, sendo essa abordagem amplamente utilizada, especialmente com Ethereum e HLF.

Já a tabela 6 responde a **RQ3** trazendo um quadro resumo dos mecanismos cripto-

Tabela 7 – Desvantagens citadas nos trabalhos relacionados sobre abordagens de nuvem

Referência	*Cloud*	Ponto único de falhas	Comprometimento da privacidade	Problemas de segurança em geral	Espalhamento de dados	Alto custo	Problemas de Confiabilidade (dependência de terceiros)
(ABDELGALIL; MEJRI, 2023)	Não usa	✗	✗	✗	✓	✗	✗
(ABOUALI *et al.*, 2021)	Não usa	✗	✓	✗	✓	✗	✗
(ALOBADH *et al.*, 2021)	Não usa	✓	✓	✓	✗	✗	✗
(CHEN *et al.*, 2022)	n/e	✓	✓	✓	✗	✗	✗
(GAO *et al.*, 2021)	Não usa	✓	✓	✗	✗	✓	✗
(GHANI *et al.*, 2020)	Não usa	✗	✗	✗	✗	✗	✓
(GUO *et al.*, 2021)	Usa (AWS)	✗	✗	✓	✗	✗	✗
(HEBBALLI *et al.*, 2023)	Não usa	✓	✓	✓	✗	✗	✗
(HOANG *et al.*, 2020)	Não usa	✓	✓	✗	✗	✗	✗
(HUANG *et al.*, 2024)	n/e	✗	✗	✗	✗	✗	✗
(JABARULLA; LEE, 2021)	Não usa	✓	✓	✗	✓	✓	✗
(JAIN; JAILIA, 2022)	Usa (n/e)	✗	✓	✓	✗	✗	✗
(KUMAR *et al.*, 2022)	Usa (AWS)	✗	✓	✓	✗	✗	✓
(LI *et al.*, 2021)	Não usa	✗	✓	✗	✗	✗	✓
(LI *et al.*, 2023a)	Não usa	✗	✗	✓	✗	✗	✗
(LI *et al.*, 2023b)	Usa (n/e)	✓	✓	✓	✗	✗	✗
(LIU; WANG, 2023)	Não usa	✓	✓	✓	✗	✗	✗
(MAINUDDIN *et al.*, 2023)	Não usa	✓	✓	✓	✗	✗	✓
(MITTAL; GHOSH, 2023)	Não usa	✗	✓	✗	✗	✗	✓
(NGUYEN *et al.*, 2019)	Usa (AWS)	✗	✗	✓	✗	✗	✗
(OH *et al.*, 2022)	Não usa	✓	✓	✗	✗	✗	✓
(RAJ; GHOSH, 2024)	Não usa	✗	✓	✗	✗	✗	✓
(SHREE *et al.*, 2024)	Usa (n/e)	✗	✓	✗	✗	✗	✗
(SINGH; RATHEE, 2023)	Não usa	✗	✓	✓	✗	✗	✗
(ULLAH *et al.*, 2022)	Não usa	✓	✗	✗	✗	✗	✓
(WEN; WANG, 2023)	Não usa	✗	✓	✗	✗	✗	✗
(YANG *et al.*, 2022)	Usa (AliCloud)	✗	✗	✓	✗	✗	✗
(YEH *et al.*, 2022)	Não usa	✓	✓	✗	✗	✗	✗
(YEH *et al.*, 2023)	Não usa	✗	✗	✗	✗	✗	✓
(ZHANG; ZHAO, 2023)	Não usa	✓	✓	✓	✓	✗	✓
(ZHENG *et al.*, 2023)	Não usa	✓	✗	✓	✗	✗	✗

n/e: não especifica.
✓: aborda como desvantagem.
✗: não aborda como desvantagem.
Fonte: elaborada pelo autor.

gráficos utilizados para proteção de dados pessoais dentre os estudos analisados, onde observa-se predominância no uso de criptografia assimétrica, simétrica e *hashing*. Alguns trabalhos, por sua vez, combinam esses mecanismos com o uso de técnicas mais recentes, como CP-ABE e ZKP.

A fim de responder a **RQ4**, a tabela 7 consolida as principais limitações no uso de computação em nuvem relatadas nos trabalhos selecionados, identificando, na coluna *cloud*, os trabalhos que fazem uso de computação em nuvem. Os principais benefícios explicitados que foram alcançados com o uso dessa tecnologia foram:

- o gerenciamento flexível de armazenamento;
- o menor esforço de manutenção da infraestrutura;
- a rapidez na administração dos dados;
- o menor custo energético;

– a conveniência;
– a escalabilidade;
– a tolerância a falhas.

Conforme delineado em **RQ5** e **RQ6**, o levantamento bibliográfico também teve como meta identificar o nível de maturidade dos estudos em relação aos direcionamentos presentes nos normativos legais vigentes relacionados à privacidade de dados. Em relação a **RQ5**, a tabela 8 resume os achados, discriminando o contexto de aplicação, a base legal mencionada e se há mecanismos de auditoria das operações de tratamento de dados pessoais. Quanto à **RQ6**, a tabela 9 destaca os requisitos legais ausentes nas arquiteturas examinadas, com foco em princípios, direitos do titular e bases legais aplicáveis. Apesar dos estudos apontarem grande preocupação com segurança e privacidade, dentre os 37 estudos analisados, apenas 5 mencionaram alguma legislação específica de privacidade. Além disso, não foi identificado qualquer trabalho que implementasse todas as exigências legais aplicáveis, especialmente em relação ao registro da base legal. Vale destacar que, dentre os trabalhos analisados, (YEH *et al.*, 2023) mostrou-se o mais aderente, necessitando apenas registrar a base legal e a finalidade, quando do registro da operação, e prever o caso de compartilhamento em situação de emergência médica.

Com base na análise dos estudos selecionados, conclui-se que, embora as soluções apresentadas ofereçam vantagens significativas em termos de segurança, integridade e descentralização dos dados, ainda persistem desafios relevantes em relação à conformidade integral com as regulamentações de privacidade de dados.

Como será detalhado no capítulo 4, este trabalho apresenta como diferencial uma abordagem de auditoria do tratamento de dados pessoais, destinada a demonstrar a conformidade das operações com legislações como a GDPR e a LGPD, oferecendo mecanismos para verificar o registro da base legal utilizada, os acessos emergenciais para proteção à vida, o exercício dos direitos de apagamento e retificação, bem como a observância dos princípios de confidencialidade e finalidade. A solução proposta utiliza uma *blockchain* em modelo tradicional, eliminando a possibilidade de alterações nos dados por *backdoors* (e.g. *hash* camaleão), mantendo compatibilidade com ambientes em nuvem e protegendo os sistemas internos das instituições contra acessos externos, uma vez que realiza a comunicação entre elas por meio de eventos na *blockchain*. Por fim, o trabalho inclui a implementação de um *framework* funcional e a descrição da infraestrutura subjacente, acompanhados de uma análise experimental do desempenho da solução.

Tabela 8 – Normativos legais nos trabalhos relacionados

Referência	Contexto	Lei	Auditoria
(GAO *et al.*, 2021) (HOANG *et al.*, 2020)	Genérico	–	✓
(GUO *et al.*, 2021) (KAVITHA *et al.*, 2022)	Genérico	–	✗
(ABOUALI *et al.*, 2021) (KUMAR *et al.*, 2022) (MAINUDDIN *et al.*, 2023) (MITTAL; GHOSH, 2023) (OH *et al.*, 2022)	*e-health*	–	✗
(ABDELGALIL; MEJRI, 2023) (GHANI *et al.*, 2020) (HEBBALLI *et al.*, 2023) (HUANG *et al.*, 2024) (KAUR *et al.*, 2024) (NGUYEN *et al.*, 2019)	*e-health*	–	✓
(ALOBADH *et al.*, 2021) (JABARULLA; LEE, 2021)	*e-health*	HIPAA	✓
(ZHENG *et al.*, 2023)	*e-health*	GDPR; HIPAA	✗
(CHEN *et al.*, 2022)	IIoT	–	✓
(TONG *et al.*, 2024)	IIoT	–	✗
(YEH *et al.*, 2023)	IoMT	GDPR	✓
(LI *et al.*, 2023b)	IoMT	GDPR; HIPAA	✓
(LI *et al.*, 2021) (RAJ; GHOSH, 2024)	IoT	–	✓
(ULLAH *et al.*, 2022) (LI *et al.*, 2023a)	IoT	–	✗
(MALLICK *et al.*, 2024)	IoMT	–	✓
(UPPAL *et al.*, 2021) (SHREE *et al.*, 2024) (SINGH; RATHEE, 2023)	IoMT	–	✗
(SHARMA *et al.*, 2019)	IoV VANET	–	✗
(DWIVEDI *et al.*, 2021) (YEH *et al.*, 2022)	ITS VANET	–	✓
(JAIN; JAILIA, 2022)	*Educational data sharing*	–	✓
(WEN; WANG, 2023)	*File storage*	–	✗
(ZHANG; ZHAO, 2023)	*Speech data sharing*	–	✓
(LIU; WANG, 2023)	*Resident travel trajectory*	–	✓

–: não aborda.
✓: sim.
✗: não.
Fonte: elaborada pelo autor.

Tabela 9 – Pontos de não-conformidade legal nos trabalhos relacionados

Referência	Registro em *logs* da base legal	Proteção à vida (acesso emergencial)	Direito ao apagamento	Princípio da confidencialidade	Direito a retificação	Princípio da finalidade
(ABDELGALIL; MEJRI, 2023)	✗	✗	✗	✓	✓	✓
(ABOUALI *et al.*, 2021)	✗	✗	✗	✓	✓	✓
(ALOBADH *et al.*, 2021)	✗	✓	✗	✓	✓	✓
(CHEN *et al.*, 2022)	✗	✓	✗	✓	✗	✓
(DWIVEDI *et al.*, 2021)	✗	✓	✓	✗	✓	✓
(GAO *et al.*, 2021)	✗	✓	✓	✓	✓	✓
(GHANI *et al.*, 2020)	✗	✓	✗	✗	✓	✓
(GUO *et al.*, 2021)	✗	✓	✓	✓	✓	✓
(HEBBALLI *et al.*, 2023)	✓	✗	✓	✓	✗	✓
(HOANG *et al.*, 2020)	✗	✗	✓	✓	✓	✓
(HUANG *et al.*, 2024)	✗	✗	✗	✓	✓	✓
(JABARULLA; LEE, 2021)	✓	✓	✓	✗	✓	✓
(JAIN; JAILIA, 2022)	✗	✗	✗	✓	✗	✗
(KAUR *et al.*, 2024)	✗	✗	✗	✓	✓	✓
(KAVITHA *et al.*, 2022)	✗	✗	✗	✓	✗	✗
(KUMAR *et al.*, 2022)	✗	✓	✓	✓	✓	✓
(LI *et al.*, 2021)	✓	✓	✗	✗	✓	✗
(LI *et al.*, 2023b)	✗	✓	✓	✓	✓	✓
(LI *et al.*, 2023a)	✗	✓	✗	✗	✓	✓
(LIU; WANG, 2023)	✗	✗	✗	✓	✗	✓
(MAINUDDIN *et al.*, 2023)	✗	✗	✗	✓	✗	✓
(MALLICK *et al.*, 2024)	✗	✓	✗	✓	✗	✓
(MITTAL; GHOSH, 2023)	✗	✗	✗	✓	✓	✓
(NGUYEN *et al.*, 2019)	✗	✓	✗	✓	✗	✓
(OH *et al.*, 2022)	✗	✗	✗	✓	✓	✓
(RAJ; GHOSH, 2024)	✗	✓	✗	✓	✓	✗
(SHARMA *et al.*, 2019)	✗	✓	✓	✓	✓	✓
(SHREE *et al.*, 2024)	✗	✗	✗	✓	✓	✓
(SINGH; RATHEE, 2023)	✓	✗	✓	✓	✓	✓
(TONG *et al.*, 2024)	✗	✓	✗	✓	✓	✓
(ULLAH *et al.*, 2022)	✗	✓	✓	✓	✓	✓
(UPPAL *et al.*, 2021)	✗	✗	✗	✓	✓	✓
(WEN; WANG, 2023)	✓	✓	✗	✓	✓	✓
(YEH *et al.*, 2022)	✗	✓	✗	✓	✓	✓
(YEH *et al.*, 2023)	✗	✗	✓	✓	✓	✗
(ZHANG; ZHAO, 2023)	✗	✓	✗	✓	✓	✓
(ZHENG *et al.*, 2023)	✗	✗	✗	✓	✓	✓

✓: trata.
✗: não trata.
Fonte: elaborada pelo autor.

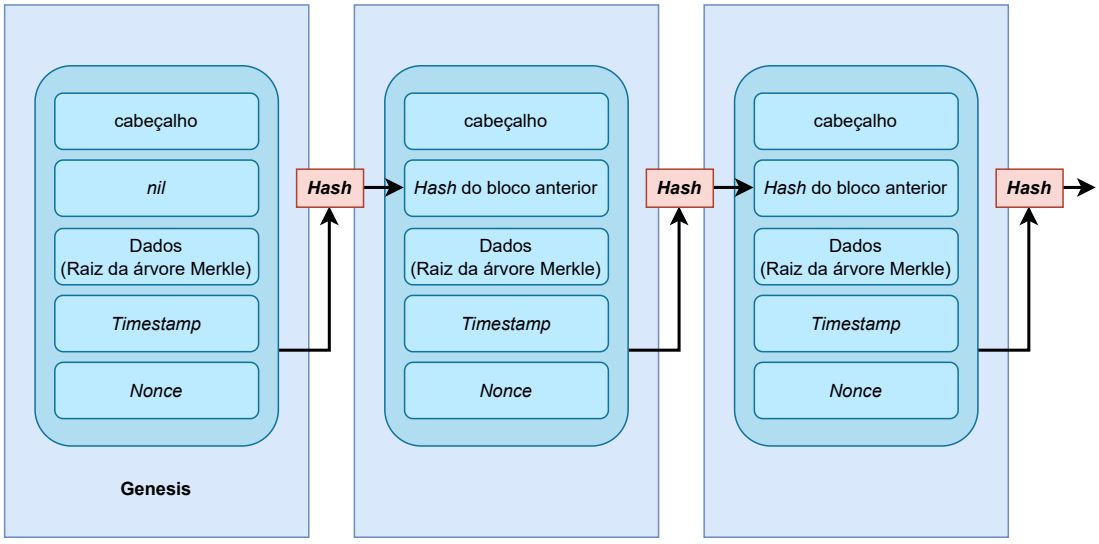
3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, as seções 3.1 e 3.2 realizam uma breve introdução aos principais conceitos e tecnologias que serão utilizados mais adiante na proposta de uma arquitetura auditável para compartilhamento de arquivos com dados pessoais entre controladores. Após, na seção 3.3, apresenta-se um breve panorama acerca da adoção de regulamentos de proteção de dados pessoais no mundo, e duas legislações centrais são abordadas de maneira esquematizada: a GDPR, regulamento europeu considerado vanguardista no que tange à proteção de dados pessoais, e a LGPD, norma brasileira influenciada pela europeia. Além de realizar um comparativo entre os termos utilizados nessas leis, são elencados os seus princípios, os direitos previstos para os titulares de dados e as bases legais que possibilitam o tratamento de dados pessoais.

3.1 Blockchain

Uma *Distributed Ledger Technology* (DLT) é um tipo de tecnologia que permite armazenar e manter um livro-razão (ou *ledger*) distribuído de forma descentralizada. Por sua vez, uma *blockchain* é uma DLT baseada em uma rede P2P e estruturada como uma cadeia sequencial e cronológica de blocos estabelecidos através de um mecanismo de consenso que os valida e replica em todos os nós da rede (BELOTTI *et al.*, 2019). A estrutura básica de uma *blockchain* pode ser vista na figura 1.

Figura 1 – Representação da estrutura de uma *blockchain*

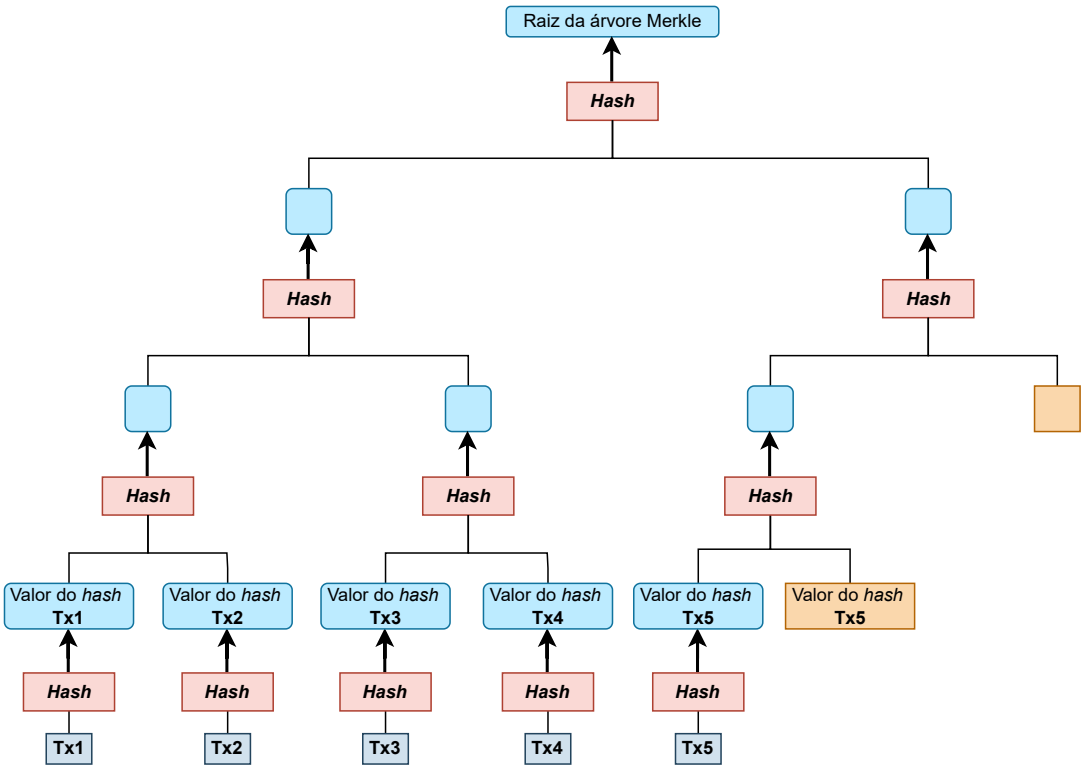


Fonte: ilustração inspirada em (BELOTTI *et al.*, 2019).

Cada bloco contém um conjunto de transações confirmadas em um determinado momento e o *hash* do bloco anterior na cadeia. Devido à resistência a colisões da função *hash*, modificações no conteúdo dos blocos mais antigos se tornam mais inviáveis à medida que a cadeia cresce, propiciando integridade e imutabilidade aos dados já escriturados (YEH *et al.*, 2023). Outras primitivas criptográficas podem ser utilizadas para prover segurança às *blockchains*, como assinaturas digitais e protocolos de PKI.

A organização das transações em uma *blockchain* baseia-se em uma árvore Merkle, que otimiza a verificação das transações em um bloco. Cada transação possui um identificador correspondente a um *hash* criptográfico. Esses identificadores são organizados em pares e processados para formar uma árvore binária cuja raiz é armazenada no cabeçalho do bloco. Esse mecanismo permite que a verificação ocorra sem a necessidade de acessar todas as transações do bloco, utilizando apenas o ramo da árvore de interesse. Caso uma transação seja adulterada, o *hash* correspondente também seria alterado, permitindo sua detecção através de um esforço computacional substancialmente menor (ALI *et al.*, 2019). A figura 2 exibe um exemplo de árvore Merkle, conforme representação ilustrada em (BELOTTI *et al.*, 2019).

Figura 2 – Exemplo de procedimento de árvore de *hash* Merkle



As transações duplicadas (com *hash*) estão marcadas em laranja.
Fonte: diagrama inspirado em (BELOTTI *et al.*, 2019).

Em uma *blockchain*, a confiança não é presumida entre os nós participantes da rede, mas sim estabelecida pela transparência e pela verificação coletiva, promovendo a segurança e a confiabilidade dos dados armazenados (ARBABI *et al.*, 2023). O novo bloco que será adicionado ao livro-razão deve ser selecionado por meio da execução de um mecanismo de consenso, que é um protocolo para assegurar a sincronização dos registros escriturados entre todos os nós quanto às transações que são válidas e que estão prestes a ser adicionadas à *blockchain* (XIAO *et al.*, 2020). Alguns dos mecanismos de consenso mais conhecidos são o *Proof-of-Work* (PoW), utilizado pelo Bitcoin (NAKAMOTO, 2008); o PoS, empregado pelo Ethereum (BUTERIN, 2014) desde 2022, a partir do "The Merge" [1]; e o *Practical Byzantine Fault Toleration* (PBFT), usado em algumas configurações de redes privadas, como o Hyperledger Fabric [2].

As *blockchains* podem ser categorizadas como *permitionless* ou *permissioned*. Uma *blockchain permitionless* permite que qualquer um se torne um participante e realize atividades como participar ativamente do mecanismo de consenso, enviar novas transações e consultar o estado do livro-razão. Os exemplos mais conhecidos atualmente desse tipo são o Bitcoin e o Ethereum. Por outro lado, em uma *blockchain permissioned*, a participação é restrita, e apenas partes com uma identidade reconhecida pré-verificada são autorizadas a ingressar na rede. Um exemplo proeminente desse tipo é o HLF.

Outra classificação das *blockchains* é quanto ao tipo de acesso, podendo ser públicas, privadas ou de consórcio. *Blockhains* públicas são redes abertas e *permitionless*, em que qualquer participante pode ler e submeter transações. Na literatura, por vezes as *blockchains* públicas e *permitionless* são consideradas equivalentes, sendo esses termos costumeiramente utilizados de forma intercambiável (ARBABI et al., 2023). *Blockchains* privadas são redes com acesso restrito e governança centralizada em uma organização, priorizando controle sobre seu uso e confidencialidade dos dados, com emprego de autorização para validadores. Por fim, *blockchains* de consórcio são redes *permissioned* regidas por um grupo de entidades, que compartilham a governança, a validação e o controle de acesso.

Embora os estados dos dados em uma *blockchain* possam mudar através da execução de transações, os registros dessas transações são únicos e imutáveis (VERA-RIVERA *et al.*, 2022). Essas transações são agrupadas em blocos, que são selecionados e replicados na rede pelo mecanismo de consenso. A quantidade total de transações em um bloco é variável, mas

[1] O termo "The Merge" refere-se à fusão da rede principal original da Ethereum (mainnet), baseada em consenso PoW, com uma *blockchain* PoS separada chamada Beacon Chain. As duas agora existem como uma única cadeia PoS.

[2] https://www.hyperledger.org/projects/fabric – Acesso em 22/06/2024.

está diretamente limitada por um valor pré-estabelecido para o tamanho máximo dos blocos (BELOTTI *et al.*, 2019). No caso do Bitcoin, esse limite é de 1 MB, enquanto no Ethereum o limite é determinado pelo valor de *gas limit*, como será visto na seção 3.1.2.

Ao fornecer governança descentralizada, imutabilidade de dados e transparência de informações, as *blockchains* impulsionam o desenvolvimento de soluções para vários setores econômicos, incluindo finanças, gerenciamento da cadeia de suprimentos e energia (VERA-RIVERA *et al.*, 2022), sendo uma solução que vem sendo explorada em diversas pesquisas, como pode ser constatado pela análise dos trabalhos relacionados, realizada no capítulo 2 .

3.1.1 Contratos inteligentes

A criação do termo contrato inteligente é creditada ao criptógrafo Nicholas "Nick" Szabo, definindo-o como "um protocolo de transação computadorizado que executa os termos de um contrato" para "minimizar exceções maliciosas e acidentais e a necessidade de intermediários confiáveis" (SZABO, 1994). Algumas *blockchains* permitem a invocação de contratos inteligentes, entre as quais a Ethereum e a HLF. Em termos técnicos, um contrato inteligente é um conjunto composto de código e variáveis de estado, ambos armazenados em uma *blockchain*, de modo que as funções do código podem ser executadas para produzir mudanças no estado dessas variáveis. Por estar armazenado em uma *blockchain*, o código não pode ser alterado e, assim, suas operações lógicas não estão sujeitas a interferências de terceiros, dispensando a necessidade de um intermediador para a execução correta de um contrato (YEH *et al.*, 2023). A tecnologia de contratos inteligentes é considerada um facilitador promissor para o compartilhamento seguro e eficiente de dados pessoais, devido às características resultantes da associação com a *blockchain*, como descentralização, *trustlessness*, imutabilidade, rastreabilidade e transparência (ARBABI *et al.*, 2023).

Entretanto, os contratos inteligentes muitas vezes não possuem maturidade suficiente e, algumas vezes, são vulneráveis a ataques de segurança (KAUR *et al.*, 2024), entre os quais vulnerabilidade de reentrância, dependência de *timestamp* e vulnerabilidade de ataque à profundidade da pilha de execução. Ferramentas para achar potenciais vulnerabilidades como Oyente (LUU *et al.*, 2016) e *frameworks* como o OpenZeppelin [3] têm sido propostas para melhorar a segurança de contratos inteligentes. Além disso, é importante ressaltar que a linha de execução de contratos inteligentes é exposta aos nós da *blockchain*, já que as transações deverão

[3] https://www.openzeppelin.com/contracts

ser validadas por todos os participantes da rede, não sendo recomendado, portanto, realizar operações envolvendo dados sensíveis através de contratos inteligentes (YEH *et al.*, 2022).

Várias linguagens são utilizadas para programação de contratos inteligentes, dependendo da *blockchain*. Solidity, por exemplo, é a linguagem utilizada para desenvolver contratos inteligentes para Ethereum, enquanto os contratos para HLF, também chamados de Chaincodes, podem ser escritos em Go, Javascript ou Java. Contratos inteligentes têm sido aplicados em diversos contextos como *Decentralized Finance* (DeFi), *Non-Fungible Token*s (NFTs), jogos, gerenciamento de documentos, gerenciamento de cadeias de suprimento, *e-healthcare* e escrituração imobiliária.

3.1.2 Ethereum

A Ethereum, proposta por Vitalik Buterin em 2014 (BUTERIN, 2014), é a primeira plataforma *blockchain* pública projetada para a execução de contratos inteligentes. Diferentemente do Bitcoin, cujo foco está predominantemente em transações financeiras, a Ethereum introduz uma infraestrutura computacional descentralizada que permite a construção de contratos inteligentes. Sua arquitetura é baseada em uma máquina de estados determinística, composta por um estado global único e pela *Ethereum Virtual Machine* (EVM), que é responsável por aplicar as mudanças de estado.

Os contratos inteligentes são escritos majoritariamente em Solidity, com restrições práticas relacionadas ao tempo de execução, ao acesso a recursos externos e à limitação de fontes de aleatoriedade. O código é compilado em bytecode e executado pela EVM de forma redundante em milhares de nós distribuídos, que validam e mantêm o estado global da rede. O mecanismo de consenso atualmente adotado é o PoS, substituindo o anterior PoW. O PoS aumentou a eficiência energética, mas ainda assim há limitações de desempenho, atualmente com um *throughput* médio de aproximadamente 15 Transações Por Segundo (TPS) e um tempo médio de produção de blocos de cerca de 12 segundos.

Para controlar o uso computacional da rede, é utilizada uma unidade para medir o esforço necessário para a execução de transações, denominada gas. A criptomoeda Ether atua como meio de pagamento para obtenção de gas, servindo como base para um mecanismo de incentivo e de proteção contra abusos. O limite de gas por bloco, atualmente fixado em 30.000.000 de unidades, define a quantidade máxima de recursos computacionais que pode ser consumida por um bloco. Transações que excedem esse limite são rejeitadas pela rede

(ANTONOPOULOS; WOOD, 2018), garantindo a segurança operacional do sistema.

O *bytecode* de um contrato inteligente Ethereum é compilado e executado como uma série de *opcodes* [4] da EVM. Há *opcodes* para operações comuns, como ADD, AND e XOR, e *opcodes* para operações específicas para o contexto de *blockchain*, como ADDRESS, BALANCE e BLOCKHASH. Para cada *opcode*, há a definição de custo em gas para sua execução, e a composição dos custos das operações individuais realizadas na execução de um contrato inteligente define o seu custo total de execução.

3.2 IPFS

O *InterPlanetary File System* (IPFS) (BENET, 2014) é uma implementação das especificações de um conjunto modular de protocolos e padrões que representa uma rede descentralizada composta por nós chamados de *peers*. Essa rede normalmente é aberta e participativa, e permite organizar e transferir dados com base no conteúdo ao invés de endereços, mas é possível configurá-la de modo privativo. Ele aborda diversos aspectos problemáticos encontrados na web atual e em seus protocolos de transferência de dados, como o *Hypertext Transfer Protocol* (HTTP), incluindo verificabilidade, resiliência, centralização, desempenho, links quebrados, soberania e propriedade de dados, armazenamento *off-chain*, software de primeiro uso local e a amarração a fornecedores. Os subsistemas do IPFS têm três responsabilidades principais em relação aos dados: representação e endereçamento, roteamento e transferência. As ideias fundamentais por trás de sua concepção incluem o uso de *hashing* criptográfico para endereçar dados de forma única, a promoção da imutabilidade de conteúdo, a persistência, permanência e *pinning* (fixação) de dados, e a participação ativa de nós na rede descentralizada (IPFS, 2023). Para ingressar na rede IPFS, um nó deve gerar um par de chaves e estabelecer contato com nós de inicialização. Cada nó é identificado pelo *hash* de sua chave pública, obviamente único, conhecido, neste contexto, como PeerID (HUANG *et al.*, 2020).

O IPFS utiliza um esquema de endereçamento baseado em conteúdo cujos identificadores únicos, chamados de CID, são baseados em *hashes* de informações sobre o conteúdo do arquivo. A ideia por trás dos CIDs é desvincular o nome do conteúdo de sua localização de armazenamento. Essa decisão possibilita a descentralização do armazenamento, da entrega de conteúdo e da gestão de endereços, evitando a dependência de fornecedores exclusivos e eliminando a necessidade de autoridades centrais.

[4] https://ethereum.org/en/developers/docs/evm/opcodes/

Quando o conteúdo de um arquivo é adicionado ao IPFS, ele é dividido em pedaços de tamanho padrão de 256 kB, sendo que cada um recebe seu próprio CID. Uma vez que todos os pedaços tenham o seu CID definido, o IPFS constrói um *Directed Acyclic Graph* (DAG) Merkle do arquivo. Um DAG Merkle é uma estrutura de dados semelhante a uma árvore Merkle (seção 3.1), mas sem os requisitos de equilíbrio. O nó raiz combina todos os CIDs de seus nós descendentes e forma o CID final do conteúdo (a raiz). Um nó pode ter vários pais, permitindo a deduplicação de pedaços, ou seja, um mesmo conteúdo não precisa ser armazenado ou transmitido duas vezes, economizando recursos de armazenamento e largura de banda. Graças a essa estrutura, os CIDs são imutáveis e auto-certificáveis, ou seja, um conteúdo não pode ser alterado sem modificar seu respectivo CID (TRAUTWEIN *et al.*, 2022).

A publicação ou recuperação de um objeto ocorre através de um mapeamento entre um CID e um PeerID. Esses mapeamentos são indexados em uma *Distributed Hash Table* (DHT) baseada no Kademlia (MAYMOUNKOV; MAZIèRES, 2002), que é semelhante à DHT principal do BitTorrent (JIMENEZ *et al.*, 2011). Novos nós ingressam como servidores se tiverem conectividade IP pública, ou como clientes, se não forem acessíveis publicamente. Para isso, o IPFS utiliza uma técnica chamada de AutoNAT, em que um novo nó ingressa como cliente, mas solicita imediatamente que outros nós na rede iniciem conexões de volta para ele. Se três ou mais nós conseguirem se conectar, esse novo nó é então promovido a servidor DHT (TRAUTWEIN *et al.*, 2022).

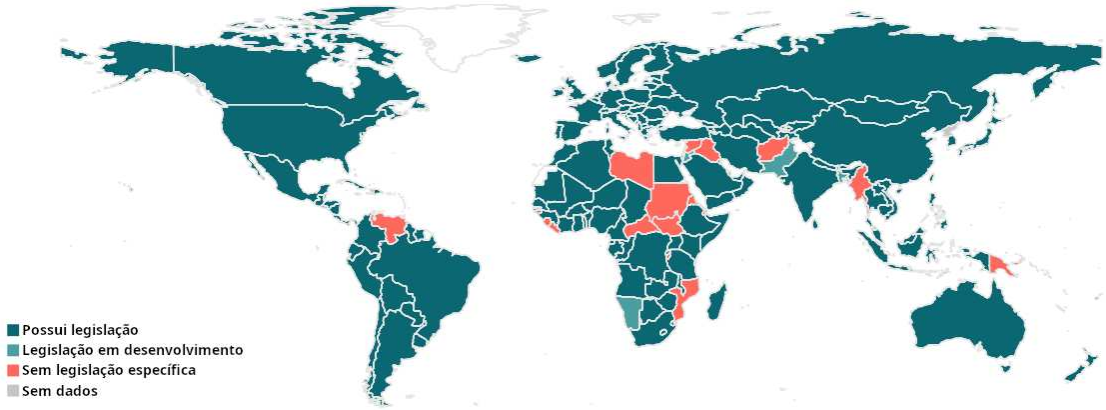
Os principais subsistemas e componentes do IPFS, conforme sua documentação oficial (IPFS, 2023), estão listados a seguir.

1. CIDs: identificam dados com base em seu conteúdo, não na localização.
2. Bitswap: protocolo para troca de blocos de dados entre pares.
3. DHTs: auxiliam no roteamento e descoberta de conteúdo.
4. DNSLink: permite vincular CIDs a domínios *Domain Name System* (DNS).
5. Sistemas de arquivos: Gerenciam armazenamento e recuperação de dados.
6. IPFS *Gateway*: fornece acesso a conteúdo IPFS através da *web* convencional.
7. *InterPlanetary Linked Data* (IPLD): estrutura de dados que permite conectar sistemas de armazenamento diversos.
8. *InterPlanetary Name System* (IPNS): sistema de nomes IPFS baseado em *blockchain*.
9. libp2p: biblioteca para construção de redes P2P.
10. DAGs Merkle: estruturas de dados para representar as informações.

Apesar de o IPFS não ser um mecanismo de armazenamento nativo para *blockchains*, é uma alternativa muito utilizada para prover armazenamento *off-chain*. Convém salientar que, caso grandes quantidades de dados fossem diretamente armazenadas na *blockchain*, haveria grande *overhead* sobre os clientes que necessitem ter uma cópia completa dos dados dos blocos (CHEN *et al.*, 2022).

3.3 Legislações sobre Privacidade de Dados

Figura 3 – Panorama mundial de legislações de proteção de dados



Fonte: extraído e traduzido de (CNUCED, 2024)

As preocupações relativas à segurança no tratamento de dados pessoais têm-se intensificado com o aumento de ataques cibernéticos, o mau uso de dados pessoais, a crescente transformação digital dos serviços e a ampliação da coleta desses dados. Em resposta, diversos países ao redor do mundo têm implementado leis rigorosas de proteção de dados. Essas regulamentações visam garantir que a coleta, armazenamento e processamento de dados pessoais sejam realizados de maneira responsável e segura, respeitando a privacidade dos indivíduos e mitigando riscos de abusos. A existência dessas regulações também aumenta a segurança jurídica, reduz os encargos administrativos e os custos de conformidade para as organizações que operam em vários países, e aumenta a confiança dos consumidores no mercado digital (DLA Piper, 2022).

A figura 3, extraída do *website* da Conferência das Nações Unidas sobre Comércio e Desenvolvimento (CNUCED), mostra uma visão geral da existência de leis de proteção de dados e privacidade ao redor do mundo. De acordo com a mesma fonte, 71% dos países adotam leis de proteção de dados; 9% têm projetos em discussão; 15% não possuem legislação; e não há informação para os 5% remanescentes. A tabela 10 mostra uma pequena relação não exaustiva

de regiões e as correspondentes normas de privacidade e proteção de dados.

Tabela 10 – Alguns regulamentos de proteção de dados no mundo

Região	Leis	Vigência	Outros regulamentos relevantes
África do Sul	POPIA	2021	
Brasil	LGPD	2020	CF/88 Lei 10.406/2002 (Código Civil) Lei 12.965/2014 (Marco Civil da Internet)
Canadá	PIPEDA	2000	
China	PIPL	2021	CSLDSL
Coreia do Sul	PIPA	2011	
EUA – Califórnia	CPPA CPRA	2020 2023	HIPAA
EUA – Colorado	CPA	2023	
EUA – Connecticut	CTDPA	2023	
EUA – Utah	UCPA	2023	
EUA – Virgínia	VCDPA	2023	
Japão	APPI	2023	
Nova Zelândia	TPA	2020	IPP
Reino Unido	DPA	2018	
Rússia	RDPA (revisão)	2015	RCSC
Singapura	PDPA (revisão)	2021	
UE	GDPR	2018	

Fonte: elaborada pelo autor a partir de (DLA Piper, 2022).

Apesar da numerosa quantidade de regulamentos existentes, o restante dessa seção destina-se a detalhar apenas a GDPR e a LGPD. A GDPR foi um dos normativos selecionados, pois, desde sua implementação em 2018, é reconhecida como a legislação mais rigorosa e influente sobre proteção de dados, servindo de modelo para outras regulamentações globais. Já a LGPD, embora também seja inspirada na GDPR, adapta seus princípios ao contexto brasileiro e estabelece diretrizes específicas para o tratamento de dados no Brasil. Considerando a posição do Brasil como uma das maiores economias da América Latina, a LGPD é crucial para a regulamentação de privacidade de dados na região. Além disso, ter estudos com foco sobre a LGPD ajuda a fomentar o interesse acadêmico nacional no tema. Ressalta-se que, apesar do foco nesses dois regulamentos, a arquitetura proposta é adaptável para outras leis, como as elencadas na tabela 10.

3.3.1 Terminologia

Para familiarizar o leitor com a terminologia utilizada no contexto de privacidade de dados pessoais, a tabela 11 oferece um quadro comparativo entre os principais conceitos da GDPR e da LGPD. Embora existam semelhanças, os conceitos apresentados não são totalmente equivalentes, pois cada legislação reflete as particularidades e necessidades jurídicas de suas respectivas jurisdições. Vale ressaltar que a tabela serve como um guia inicial para entender as diferenças e similaridades entre esses dois marcos regulatórios, mas não substitui uma análise detalhada de cada conceito em seu contexto específico de aplicação.

3.3.2 GDPR

Em vigor desde 25 de maio de 2018, a GDPR é uma das legislações de proteção de dados mais abrangentes e rigorosas do mundo. Pensada para "aumentar a certeza jurídica, reduzir o ônus administrativo e o custo de conformidade para organizações ativas em vários Estados Membros da UE, e aumentar a confiança do consumidor no único mercado digital", ela estabelece diretrizes claras sobre a coleta, processamento e armazenamento de dados pessoais e concede aos cidadãos da UE maior controle e proteção sobre suas informações pessoais, impondo obrigações a organizações que realizam tratamento de dados independentemente de se situarem ou não nos limites territoriais da UE (DLA Piper, 2022). A GDPR prevê a aplicação de multas severas àqueles que violem seus padrões de privacidade e segurança, com penalidades que chegam a dezenas de milhões de euros [5]. A fim de reforçar a relevância na conformidade com o regulamento, a figura 4 exibe a quantidade de multas aplicadas por setor, bem como o montante gasto pelas instituições a fim de quitar o débito gerado.

3.3.2.1 Princípios da GDPR

Nesta seção, serão discutidos os princípios fundamentais segundo o artigo 5 da GDPR, que constituem a base normativa para o processamento de dados pessoais. Estes princípios orientam tanto as práticas organizacionais quanto as individuais no manuseio de informações pessoais. A GDPR enumera os seguintes princípios:

1. **Licitude, justiça e transparência**: assegura que organizações informem as pessoas sobre como usarão seus dados pessoais.

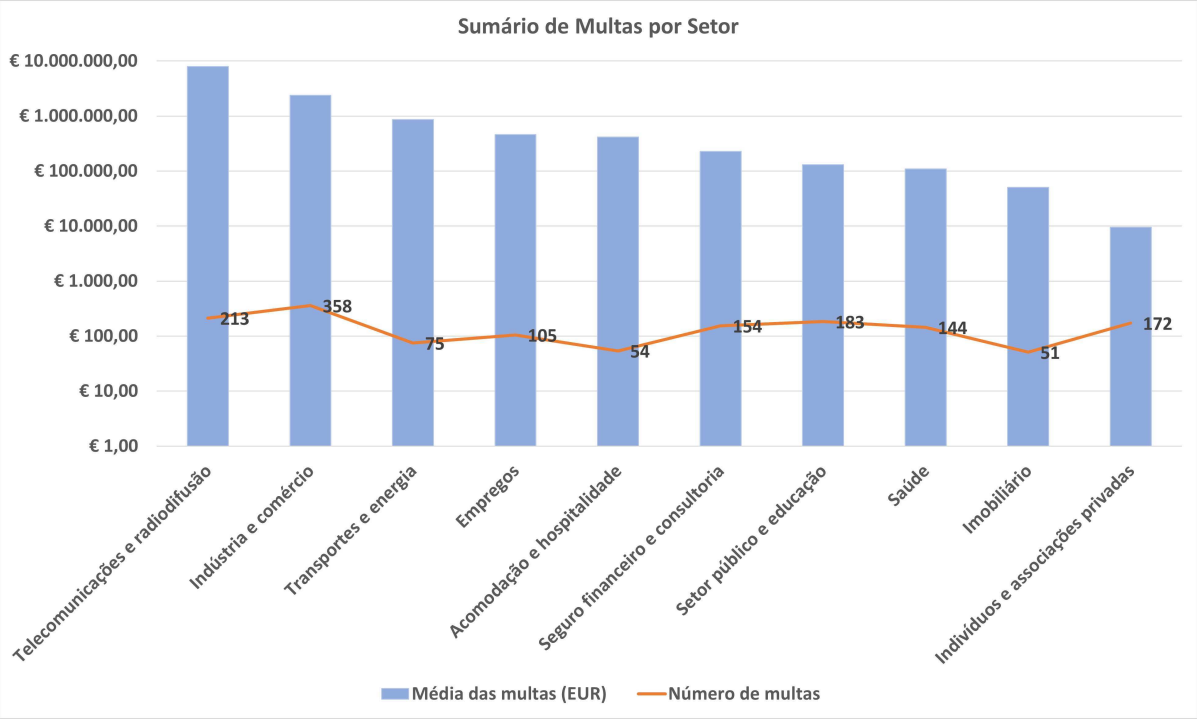
[5] https://www.enforcementtracker.com

Tabela 11 – Comparação entre conceitos da GDPR e LGPD

GDPR	LGPD	Descrição comum
Data subject	Titular de dados	Pessoa natural a quem se referem os dados pessoais que são objeto de tratamento.
Personal data	Dado pessoal	Informação relacionada a uma pessoa natural identificada ou identificável.
Processing	Tratamento	Qualquer operação realizada com dados pessoais, como coleta, armazenamento, uso, etc.
Controller	Controlador	Pessoa que determina os propósitos e meios de tratamento de dados pessoais.
Processor	Operador	Pessoa que processa dados pessoais em nome do controlador.
Consent	Consentimento	Manifestação de vontade livre, específica, informada e inequívoca pela qual o titular concorda com o tratamento de seus dados pessoais.
Genetic data, biometric data, data concerning health	Dado pessoal sensível	Dados sobre origem racial, convicção religiosa, opinião política, saúde, vida sexual, dados genéticos ou biométricos.
Pseudonymisation	Anonimização	Tratamento de dados pessoais de forma que não possam ser atribuídos a um titular específico sem informações adicionais.
Data protection officer	Encarregado	Pessoa indicada para atuar como canal de comunicação entre o controlador, os titulares dos dados e a Autoridade Nacional ou *Supervisory authority*.
Personal data breach	Não há definição explícita	Violação de segurança que leva à destruição, perda, alteração, divulgação não autorizada ou acesso aos dados pessoais transmitidos, armazenados ou de outro modo processados.
Filing system	Banco de dados	Conjunto estruturado de dados pessoais, estabelecido em um ou em vários locais, em suporte eletrônico ou físico.
Erasure or forgotten	Eliminação	Exclusão de dado ou de conjunto de dados armazenados em banco de dados, independentemente do procedimento empregado.
Restriction of processing	Bloqueio	Suspensão temporária de qualquer operação de tratamento, mediante guarda do dado pessoal ou do banco de dados.
General principle for transfers	Transferência internacional de dados	Transferência de dados pessoais para país estrangeiro ou organismo internacional.
Third party	Uso compartilhado de dados	Comunicação, difusão, transferência internacional, interconexão de dados pessoais ou tratamento compartilhado de bancos de dados pessoais por órgãos e entidades públicas e privadas.
Supervisory authority	Autoridade Nacional	Órgão da administração pública responsável por zelar, implementar e fiscalizar o cumprimento desta Lei em todo o território nacional.

Fonte: elaborada pelo autor a partir de (Parlamento Europeu, Conselho da União Europeia, 2016) e (Brasil, 2018).

Figura 4 – Sumário de multas aplicadas com fundamento na GDPR (quantidade e valor médio)



Fonte: (CMS Law, 2023)

2. **Limitação de propósito**: as organizações só podem coletar dados pessoais para fins específicos.
3. **Minimização de dados**: limita os dados coletados aos que as organizações necessitam para realizar determinado processamento.
4. **Precisão e atualização**: as organizações que coletam e processam os dados devem garantir sua exatidão e atualizá-los. Também devem alterá-los ou apagá-los a pedido de seus titulares.
5. **Limitação de armazenamento**: as organizações não reterão dados coletados por mais tempo do que o necessário.
6. **Integridade e confidencialidade**: organizações devem aplicar medidas de segurança e proteção apropriadas para proteger os dados contra roubo e acesso não autorizado.
7. **Conformidade**: controladores de dados devem demonstrar conformidade com a lei.

3.3.2.2 Direitos dos titulares na GDPR

Esta seção sintetiza os direitos dos titulares de dados conforme o capítulo 3 da GDPR. Para uma descrição detalhada, consulte (Parlamento Europeu, Conselho da União Europeia, 2016). Os seguintes artigos tratam desses direitos:

- **Artigo 12 - Transparência, informação e comunicação:** o controlador deve comunicar, de forma clara, concisa, transparente e facilmente acessível, informações acerca das operações realizadas sobre os dados pessoais.
- **Artigo 13 - Informações a serem fornecidas ao titular quando os seus dados pessoais são coletados:** estabelece as informações que devem ser fornecidas ao titular dos dados no ato da coleta.
- **Artigo 14 - Informações a serem fornecidas ao titular se os seus dados pessoais forem obtidos de outras fontes:** define as informações que devem ser fornecidas ao titular dos dados quando os dados pessoais foram obtidos de outras fontes que não diretamente dele.
- **Artigo 15 - Direito de acesso do titular dos dados:** garante ao titular o direito de obter, do controlador, confirmação de que seus dados pessoais estão sendo ou não processados, além de informações que sejam relevantes sobre esse processamento e o acesso aos próprios dados.
- **Artigo 16 - Direito de retificação:** permite ao titular obter a correção de dados pessoais imprecisos ou incompletos.
- **Artigo 17 - Direito ao apagamento (ou direito ao esquecimento):** obriga o controlador, mediante certas circunstâncias, a realizar a eliminação dos dados pessoais de um titular.
- **Artigo 18 - Direito à limitação do tratamento:** consiste no direito de obter restrições quanto ao tratamento de seus dados em situações específicas.
- **Artigo 19 - Obrigação de notificação em caso de retificação, apagamento ou limitação do tratamento de dados:** trata da obrigação do controlador de notificar os destinatários com quem os dados pessoais foram compartilhados sobre a necessidade de retificação, apagamento ou limitação do tratamento, a menos que essa notificação seja impossível ou exija um esforço desproporcional.
- **Artigo 20 - Direito à portabilidade de dados:** permite ao titular obter seus dados junto a um controlador em um formato estruturado, de uso comum e de leitura automática, para encaminhamento a outro controlador.
- **Artigo 21 - Direito de oposição:** assegura ao titular dos dados o direito de se opor a determinados tipos de tratamento de dados pessoais, incluindo a criação de perfis baseada em interesses legítimos ou para marketing direcionado.
- **Artigo 22 - Objeção a decisões automatizadas:** estabelece o direito do titular de não ser submetido a decisões baseadas exclusivamente em processamento automatizado. Entre

essas decisões estão a definição de perfil, as que venham a produzir efeitos jurídicos ou a afetar significativamente o titular.

– **Artigo 23 - Restrições:** permite restrições aos direitos dos titulares desde que previstas por lei, guardando a proporcionalidade e o respeito aos direitos fundamentais, por motivação de segurança nacional, defesa, segurança pública, investigação de crimes, interesses econômicos ou financeiros, independência judicial, monitoramento de funções públicas e direitos de terceiros.

3.3.2.3 Bases legais da GDPR para tratamento de dados pessoais

A tabela 12 apresenta resumidamente as situações em que o tratamento de dados pessoais é permitido, segundo a GDPR.

Tabela 12 – Bases legais para processamento de dados pessoais segundo o Artigo 6º da GDPR

Inciso	Base legal	Descrição
1	Consentimento	O titular deve fornecer seu consentimento para o processamento de seus dados pessoais para uma ou mais finalidades específicas.
2	Execução de contrato	O processamento é necessário para executar contrato do qual o titular é parte ou para procedimentos preliminares a ele relacionados.
3	Obrigações legais	O processamento é necessário para o cumprimento de uma obrigação legal à qual o controlador está sujeito.
4	Interesses vitais	O processamento é necessário para proteger os interesses vitais do titular dos dados ou de outra pessoa natural.
5	Interesse público	O processamento é necessário para o desempenho de uma tarefa realizada no interesse público ou no exercício de autoridade oficial investida no controlador.
6	Interesses legítimos	O processamento é necessário para atender aos interesses legítimos do controlador ou de um terceiro, exceto quando tais interesses são sobrepostos pelos interesses ou direitos e liberdades fundamentais do titular dos dados, especialmente se ele for uma criança.

Fonte: elaborada pelo autor a partir de (Parlamento Europeu, Conselho da União Europeia, 2016).

3.3.3 LGPD

Com o objetivo de proteger os direitos fundamentais de liberdade e privacidade e a livre formação da personalidade, a LGPD (Brasil, 2018) foi promulgada em 2018, entrando em vigor em agosto de 2020 e prevendo sanções a partir de agosto de 2021.

Nesse contexto, a ANPD é o órgão central de interpretação e fiscalização da LGPD no Brasil, tendo iniciado suas atividades em 2020. Ele possui atribuições que abrangem aspectos orientativos, normativos, fiscalizatórios e sancionatórios, sendo responsável por estabelecer normas e diretrizes para a implementação da LGPD em instituições públicas e privadas nos níveis federal, estadual e municipal.

De acordo com informações divulgadas pela própria ANPD, até agosto de 2023, foram recebidas 237 CISs, atualmente em análise. Entre os incidentes, destacam-se os casos relacionados a *ransomware*, que representam 97 ocorrências (ANPD, 2023). O documento também aponta para a possibilidade de ocorrência de subnotificação no Brasil, considerando que outros países apresentam números significativamente mais elevados de incidentes relatados, como 4.000 casos na França e 8.800 na Inglaterra. Um agente de tratamento de dados precisa implantar controles e medidas técnicas de segurança da informação adequadas para que possa identificar um incidente e comunicá-lo à autoridade competente aplicável e aos titulares, algo que, no Brasil, ainda está em um estágio de maturidade inicial.

Quanto às sanções, após a publicação da norma específica de dosimetria em fevereiro de 2023, foram aplicadas penalidades em 2 processos administrativos contra agentes públicos e privados, e outros 7 permanecem em análise. A tabela 13 exibe a quantidade anual de CISs recebidos a partir do ano de 2021. Já a figura 5 apresenta os números relativos aos requerimentos recebidos no mesmo período.

Tabela 13 – Quantidade de CISs recebidas pela ANPD por ano

Ano	**Comunicados no ano**	**Acumulado desde 2021**
2021	186	186
2022	275	461
2023	352	813
2024*	152	965

* dados disponíveis até o mês de junho.
Fonte: elaborada pelo autor a partir de (ANPD, 2023).

Com base nesses dados, observa-se uma tendência de aumento no número de comunicados de incidentes recebidos. Já a quantidade de requerimentos formais apresentados ao órgão mantém-se relativamente estável: desconsiderando os dois primeiros trimestres avaliados, esse volume varia entre 215 e 337 registros por semestre, com média aproximada de 250. Ainda segundo (ANPD, 2023), a expectativa é que, à medida que os agentes de tratamento compreendam e se alinhem aos padrões mínimos de segurança, haja um crescimento significativo na quantidade

Figura 5 – Requerimentos recebidos pela ANPD por período

Fonte: elaborada pelo autor a partir de (ANPD, 2023).

de notificações de incidentes de segurança no Brasil. Esse cenário reforça a importância do aprimoramento contínuo das práticas de segurança da informação e do fortalecimento da cultura de proteção de dados nas organizações.

3.3.3.1 Princípios da LGPD

Esta seção apresenta uma transcrição dos princípios fundamentais estabelecidos pela LGPD em seu artigo 6º (Brasil, 2018). Esses princípios constituem a base normativa para o tratamento de dados pessoais e orientam as práticas que organizações e indivíduos devem seguir no manuseio desse tipo de informação.

- *I - **finalidade**: realização do tratamento para propósitos legítimos, específicos, explícitos e informados ao titular, sem possibilidade de tratamento posterior de forma incompatível com essas finalidades;*
- *II - **adequação**: compatibilidade do tratamento com as finalidades informadas ao titular, de acordo com o contexto do tratamento;*
- *III - **necessidade**: limitação do tratamento ao mínimo necessário para a realização de suas finalidades, com abrangência dos dados pertinentes, proporcionais*

e não excessivos em relação às finalidades do tratamento de dados;

- *IV - **livre acesso**: garantia, aos titulares, de consulta facilitada e gratuita sobre a forma e a duração do tratamento, bem como sobre a integralidade de seus dados pessoais;*
- *V - **qualidade dos dados**: garantia, aos titulares, de exatidão, clareza, relevância e atualização dos dados, de acordo com a necessidade e para o cumprimento da finalidade de seu tratamento;*
- *VI - **transparência**: garantia, aos titulares, de informações claras, precisas e facilmente acessíveis sobre a realização do tratamento e os respectivos agentes de tratamento, observados os segredos comercial e industrial;*
- *VII - **segurança**: utilização de medidas técnicas e administrativas aptas a proteger os dados pessoais de acessos não autorizados e de situações acidentais ou ilícitas de destruição, perda, alteração, comunicação ou difusão;*
- *VIII - **prevenção**: adoção de medidas para prevenir a ocorrência de danos em virtude do tratamento de dados pessoais;*
- *IX - **não discriminação**: impossibilidade de realização do tratamento para fins discriminatórios ilícitos ou abusivos;*
- *X - **responsabilização e prestação de contas**: demonstração, pelo agente, da adoção de medidas eficazes e capazes de comprovar a observância e o cumprimento das normas de proteção de dados pessoais e, inclusive, da eficácia dessas medidas.*

3.3.3.2 Direitos dos titulares na LGPD

Esta seção elenca os direitos dos titulares de dados pessoais, conforme estabelecido pelo capítulo III da LGPD. Esses direitos conferem aos indivíduos maior autonomia na gestão de suas informações pessoais, promovendo a proteção da privacidade e assegurando o cumprimento rigoroso das disposições legais pelas organizações envolvidas no tratamento de dados.

- Artigo 18, I - **Confirmação da existência do tratamento**: direito de obter a confirmação de que seus dados pessoais estão sendo processados.
- Artigo 18, II - **Acesso aos dados**: garante a um indivíduo o acesso aos dados pessoais que lhe dizem respeito.

- Artigo 18, III - **Correção de dados**: assegura a correção de dados pessoais incompletos, inexatos ou desatualizados, mediante solicitação do titular.
- Artigo 18, IV - **Anonimização, bloqueio ou eliminação**: permite o pedido de anonimização, bloqueio ou eliminação de dados desnecessários, excessivos ou tratados em desconformidade com a lei.
- Artigo 18, V - **Portabilidade dos dados**: confere o direito ao titular de solicitar a portabilidade de seus dados pessoais para outro fornecedor de serviço ou produto, mediante regulamentação da autoridade nacional.
- Artigo 18, VI - **Eliminação dos dados tratados com consentimento**: estabelece o direito de solicitar a eliminação dos dados pessoais tratados com o consentimento do titular.
- Artigo 18, VII - **Informação sobre compartilhamento de dados**: permite ao titular obter informações sobre as entidades públicas e privadas com as quais o controlador realizou uso compartilhado de dados.
- Artigo 18, VIII - **Informação sobre a possibilidade de não consentir**: estabelece que o titular deve ser informado sobre a possibilidade de não fornecer consentimento e sobre as consequências dessa negativa.
- Artigo 18, IX - **Revogação do consentimento**: garante ao titular o direito de revogar o consentimento, mediante manifestação expressa, em procedimento gratuito e facilitado.
- Artigo 19 - **Confirmação de existência e acesso**: a confirmação de existência ou o acesso aos dados pessoais serão providenciados de forma imediata, em formato simplificado, ou em até 15 dias, se em formato claro e completo. Essas informações devem ser fornecidas em meio eletrônico ou impresso, a critério do titular.
- Artigo 20 - **Revisão de decisões automatizadas**: permite ao titular solicitar a revisão de decisões tomadas unicamente com base em tratamento automatizado de dados pessoais que afetem seus interesses, incluindo decisões destinadas a definir o seu perfil pessoal, profissional, de consumo e de crédito ou os aspectos de sua personalidade.
- Artigo 21 - **Proteção contra uso prejudicial de dados**: resguarda os titulares contra o uso prejudicial de seus dados ao exercerem seus direitos.
- Artigo 22 - **Direito à defesa judicial**: assegura que a defesa dos interesses e dos direitos dos titulares de dados poderá ser exercida em juízo, individual ou coletivamente, na forma do disposto na legislação pertinente, acerca dos instrumentos de tutela individual e coletiva.

3.3.3.3 Bases legais da LGPD para tratamento de dados pessoais

Segundo o artigo 7º da LGPD, o tratamento de dados pessoais somente poderá ser realizado nas hipóteses descritas na tabela 14.

Tabela 14 – Bases legais para processamento de dados pessoais segundo o Artigo 7º da LGPD

Inciso	Base legal	Descrição
I	Consentimento	O titular dos dados deu consentimento, de forma livre, informada e inequívoca, para o tratamento de seus dados pessoais para uma finalidade específica.
II	Cumprimento de obrigação legal	O tratamento é necessário para o cumprimento de uma obrigação legal ou regulatória pelo controlador.
III	Execução de políticas públicas	O tratamento é realizado pela administração pública para a execução de políticas públicas, previstas em leis ou regulamentos.
IV	Estudos por órgão de pesquisa	O tratamento é necessário para a realização de estudos por órgão de pesquisa, garantida, sempre que possível, a anonimização dos dados pessoais.
V	Execução de contrato	O tratamento é necessário para a execução de um contrato no qual o titular dos dados é parte, ou para procedimentos preliminares relacionados a um contrato.
VI	Exercício regular de direitos	O tratamento é necessário para o exercício regular de direitos em processo judicial, administrativo ou arbitral.
VII	Proteção da vida	O tratamento é necessário para proteger a vida ou a incolumidade física do titular dos dados ou de terceiros.
VIII	Tutela da saúde	O tratamento é necessário para a tutela da saúde, em procedimento realizado por profissionais da área da saúde ou por entidades sanitárias.
IX	Interesses legítimos	O tratamento é necessário para atender aos interesses legítimos do controlador ou de terceiros, exceto quando prevalecerem direitos e liberdades fundamentais do titular dos dados que exijam a proteção dos dados pessoais.
X	Proteção ao crédito	O tratamento é necessário para a proteção do crédito, nos termos do Código de Defesa do Consumidor.

Fonte: elaborada pelo autor a partir de (Brasil, 2018).

4 ARQUITETURA G4DPA

Neste capítulo, será apresentada uma arquitetura para prover o compartilhamento de arquivos contendo dados pessoais entre diferentes instituições, buscando seguir definições e regras expressas na GDPR e na LGPD. Essa arquitetura, chamada de G4DPA, constitui a principal contribuição deste trabalho e foi concebida para possibilitar a realização de busca e a recuperação de dados entre as instituições participantes, sem a necessidade de armazenar os arquivos diretamente em um repositório centralizado ou de expor sistemas internos dessas instituições para acessos oriundos da Internet. As operações sobre esses dados, como registro de custódia, busca e recuperação, são registradas para fins de auditoria através de um contrato inteligente em uma *blockchain*, de modo a evitar possíveis adulterações nos *logs*.

Nas próximas seções, são apresentados os objetivos de *design* que orientaram a proposta (4.1), as entidades participantes (4.2) e as premissas e restrições assumidas para o funcionamento da G4DPA (4.3 e 4.4). Em seguida, descreve-se o modelo de sistema (4.5), que fornece uma visão geral das operações de compartilhamento e, por fim, realiza-se a análise de segurança das requisições e dos dados compartilhados (4.6).

4.1 Objetivos de *design*

Como visto no capítulo 2, vários trabalhos na literatura científica recente abordam o compartilhamento de dados entre diferentes instituições e sistemas. Por conta do risco associado ao compartilhamento de dados sensíveis, governos nacionais no mundo inteiro vêm criando legislações específicas de proteção de dados pessoais, como a GDPR e a LGPD.

Para possibilitar essas transferências de dados, uma possível abordagem seria que cada instituição expusesse uma *Application Programming Interface* (API) de seu sistema legado para acesso através da internet. Contudo, essa abordagem não oferece, por si só, um mecanismo capaz de prover buscas globais sobre os dados em todas as instituições, impondo, ainda, o desafio de gerenciamento de vários *endpoints*. Além disso, a heterogeneidade tecnológica e de infraestrutura exigiria investimento em conhecimento técnico e substancial esforço operacional para adaptação e integração dos sistemas legados, com potencial aumento de custos e de riscos à segurança cibernética, dada a maior superfície exposta a ataques.

Uma outra alternativa para o problema seria o estabelecimento de *Virtual Private Network*s (VPNs) entre as instituições interessadas. Entretanto, essa abordagem provavelmente

se mostraria inviável rapidamente, já que o número de conexões a serem estabelecidas seria $\frac{n(n-1)}{2}$, sendo n o número total de instituições.

Por fim, não é trivial assegurar um mecanismo de auditoria transparente e confiável para esse problema, pois os registros armazenados em bases locais normalmente estão sujeitos a manipulação fraudulenta, e há conflito entre a publicidade do registro de auditoria e a confidencialidade dos dados. Portanto, é desejável um meio de integração entre os sistemas que não exija sua exposição direta à internet, mitigando custos e riscos decorrentes de vulnerabilidades de *software*, *hardware* ou de engenharia social.

Assim, o *design* da G4DPA leva em consideração as seguintes diretrizes:

1. Fornecer uma arquitetura padronizada para propiciar o compartilhamento seguro de dados pessoais entre instituições;
2. Proporcionar conformidade com a GDPR e a LGPD;
3. Evitar aumentar a exposição de sistemas das instituições participantes a acessos a partir da internet;
4. Minimizar os custos com a adaptação de sistemas legados; e
5. Proporcionar um mecanismo de auditoria baseado em um registro imutável.

Segundo (PAAR; PELZL, 2010), os serviços de segurança mais importantes são a confidencialidade, a integridade, a autenticação de mensagens e o não-repúdio. Adicionalmente, também são citados a autenticação, o controle de acesso, a disponibilidade, a auditoria, a segurança física e a anonimidade. A segurança física consiste na proteção de equipamentos e infraestruturas contra acessos não autorizados, desastres naturais, falhas elétricas ou outros incidentes em nível físico e, embora seja um aspecto importante da segurança da informação, ela não será tratada neste trabalho, uma vez que o foco está na concepção de uma arquitetura lógica para o compartilhamento seguro de dados pessoais. Com base nisso, a G4DPA deve apresentar as seguintes características de segurança:

Confidencialidade. A confidencialidade assegura que não haja acessos não autorizados às informações. Dados em trânsito e em repouso costumam ser protegidos utilizando criptografia, e apenas agentes autenticados e autorizados devem ter acesso às informações decifradas. Assim, mesmo em caso de vazamentos ou de espionagem, o conteúdo não será revelado.

Integridade. A integridade consiste em detectar se os dados sofreram alterações não autorizadas, em trânsito ou em repouso. Técnicas criptográficas como funções *hash* e assinaturas

digitais costumam ser empregadas para verificar se houve ou não alguma modificação nas informações.

Autenticação mútua. Em uma comunicação, a autenticação mútua refere-se à operação que compreende a autenticação do remetente pelo destinatário e, ao mesmo tempo, do destinatário pelo remetente. Em outras palavras, o destinatário deve ignorar mensagens de remetentes não autenticados e deve ser incapaz de ler mensagens se não tiver sido autenticado pelo remetente. Esse aspecto abrange tanto a autenticação das mensagens quanto a autenticação mencionada por (PAAR; PELZL, 2010), sendo considerado um mecanismo ideal para a transmissão de dados sensíveis, apesar de seus maiores custos computacionais (CHEN *et al.*, 2022; HUANG *et al.*, 2024).

Não-repúdio. Um esquema proporciona não-repúdio se é impossível ou inviável ao remetente de uma mensagem negar sua autoria. Essa característica é essencial em sistemas que possibilitam mecanismos robustos de auditoria, proporcionando a responsabilização dos usuários por suas ações.

Anonimização. A anonimização é um processo de transformação de dados pessoais para dificultar ou inviabilizar a identificação das pessoas às quais esses dados se referem. Caso haja previsão de que os dados possam ser atribuídos novamente a um indivíduo específico através do uso de informações adicionais, como uma senha, o processo é chamado de pseudoanonimização. Claramente, essas informações adicionais devem ser mantidas seguras separadamente. A GDPR prevê o uso de anonimização em seu considerando 26 e a LGPD o faz em seu artigo 13.

Controle de acesso. O controle de acesso consiste no emprego de mecanismos de autenticação e autorização para garantir que apenas usuários ou entidades previamente autorizadas possam executar determinadas ações sobre os dados ou recursos de um sistema.

Disponibilidade. A disponibilidade garante que dados e serviços estejam acessíveis aos usuários devidamente autorizados sempre que necessário.

Auditoria. A auditoria refere-se à capacidade de registrar e de verificar as ações realizadas sobre os dados, possibilitando o rastreamento de acessos e modificações. Ela é essencial para a responsabilização de usuários e para a garantia de conformidade com normas e legislações de proteção de dados, como a GDPR e a LGPD.

A disponibilidade provida pela arquitetura apoia-se na alta disponibilidade da *blockchain* (seção 3.1) e do IPFS (seção 3.2), além da possibilidade de provisionamento de múltiplos *gateways*, como será posteriormente detalhado. O controle de acesso será implementado por

meio de certificados digitais no padrão x.509 (BOEYEN *et al.*, 2008), ressalvando-se que a segurança e a gerência da emissão desses certificados não integram o escopo deste trabalho. Os registros de auditoria serão mantidos na *blockchain*, cuja característica de imutabilidade a torna particularmente adequada para esse fim. Os demais aspectos de segurança serão discutidos em maior detalhe na seção 4.6.1.

Por fim, a arquitetura deve ser resistente aos seguintes tipos de ataques, os mais comumente analisados, sob a ótica da segurança, nas soluções similares apresentadas nos trabalhos relacionados, elencados no capítulo 2:

Ataques *replay*. Os ataques *replay* podem ocorrer quando não é possível para o destinatário ou o remetente das mensagens verificar se a mensagem está sendo reutilizada ou não (PAAR; PELZL, 2010). Esse tipo de ataque envolve a captura de um pacote e sua posterior retransmissão para produzir um efeito adverso no sistema (STALLINGS, 2005).

Ataques *man-in-the-middle*. Em um ataque *man-in-the-middle*, um atacante se coloca entre o remetente e o destinatário na comunicação, atuando como servidor para o remetente e como cliente para o destinatário. As mensagens interceptadas podem então ser lidas, alteradas ou simplesmente descartadas (STALLINGS, 2005). Muitas vezes é um ataque utilizado contra sistemas que usam criptografia de chave pública, como a versão mais básica do algoritmo Diffie-Hellman, quando as chaves públicas trafegam sem proteção (PAAR; PELZL, 2010).

Ataques de personificação (*impersonation*). Um ataque de personificação ocorre quando uma entidade finge ser outra para obter privilégios ou realizar transações em nome dessa outra entidade. Normalmente, fazem uso de outros tipos de ataque conjuntamente. Por exemplo, um atacante pode capturar uma sequência de pacotes relativos a uma transação de autenticação através de *man-in-the-middle* e, utilizando *replay*, retransmiti-los para se autenticar e obter privilégios que não teria no sistema (STALLINGS, 2005).

A resistência a esses ataques será explicada mais adiante, na seção 4.6.2.

4.2 Entidades

A seguir, serão descritas as entidades definidas pela G4DPA. Para facilitar a compreensão do restante do texto, as siglas utilizadas e os seus significados são apresentados resumidamente na tabela 15.

Autoridade de Proteção de Dados (APD). A APD é a entidade responsável por inicializar o

Tabela 15 – Siglas das entidades da G4DPA e seus significados

Sigla	Significado
APD	Autoridade de Proteção de Dados
AC	Autoridade Certificadora
TD	Titular de Dados
CC	Controlador Custodiante
CR	Controlador Requisitante
GTW	*Gateway*
BC	*Blockchain*
IPFS	*InterPlanetary File System*

Fonte: elaborada pelo autor.

sistema e seus parâmetros e por supervisionar o uso de dados pessoais pelas entidades participantes do sistema de acordo com as normas legais aplicáveis, realizando o tratamento das reclamações acerca de violações. Ela representa as autoridades supervisoras (*supervisory authorities*) na GDPR (capítulo 6, artigos 51 a 59) e a ANPD na LGPD (capítulo IX, seção I, artigos 55 a 57).

Autoridade Certificadora (AC). Uma AC é uma entidade que recebe competência da APD para emitir certificados digitais que serão utilizados para identificar e autorizar os participantes do sistema. Cada participante comunicante deve possuir uma chave privada e um certificado digital que contém a respectiva chave pública, formando o par de chaves que será utilizado para criptografar, assinar e verificar as mensagens trocadas.

Titular de Dados (TD). O TD corresponde ao *data subject* da GDPR e à entidade homônima da LGPD, ambos descritos na seção 3.3.1. Ele pode realizar consultas para saber quais são os seus dados pessoais que estão compartilhados, quais são os controladores com os quais esses dados estão compartilhados e, quando aplicável, fornecer e revogar consentimento para o tratamento de dados.

Controlador Custodiante (CC). Um CC é uma entidade que possui, sob sua custódia, dados pessoais de um TD, correspondendo a um *controller* na GDPR e a um Controlador de Dados (CD) na LGPD (seção 3.3.1). Esses dados podem estar armazenados no próprio CC ou em um Operador de Dados (OD) atuando em nome do CC. Cabe ao CC informar ao *Gateway* (GTW) acerca de todos os dados pessoais que estão ou que deixam de estar sob sua custódia. São essas informações que possibilitarão buscas por esses dados.

Controlador Requisitante (CR). Uma CR é um CD (seção 3.3.1) que busca por dados pessoais de um TD junto ao GTW e que pode solicitá-los a um CC por meio do GTW. Um mesmo

CD pode ser um CR e um CC, dependendo apenas do papel que é exercido em uma interação no sistema em determinado momento.

***Gateway* (GTW).** O GTW é a entidade responsável por receber requisições de TD, CC e CR, anonimizá-las e registrá-las na BC para possibilitar buscas e prover auditoria. Além disso, ele dispara notificações através da BC para transmitir instruções de coordenação ao CC e ao CR. As mensagens trocadas com o GTW devem estar assinadas por um certificado digital devidamente emitido pela AC, e as que não estão são descartadas de imediato.

***Blockchain* (BC).** A BC (seção 3.1) é um serviço de armazenamento imutável através do qual serão registradas as informações necessárias para realizar as buscas por dados pessoais e as auditorias sobre as operações. É necessário esclarecer que nem os arquivos com dados pessoais nem seus metadados são diretamente armazenados na BC, havendo nela apenas ponteiros pseudoanonimizados para eles, e que somente a APD e o GTW possuem as informações adicionais necessárias para reverter a pseudoanonimização realizada. As entidades TD, CC e CR enviam dados para registro na BC através do GTW. Todavia, todas as entidades podem realizar operações de leitura diretamente sobre a BC. Na G4DPA, a BC também desempenha a função de meio de notificação, transmitindo instruções do GTW para as demais entidades quando necessário.

IPFS. O IPFS (seção 3.2) é um sistema para armazenamento de arquivos distribuído, que estará acessível para GTW, CC e CR. Um arquivo armazenado em um nó do IPFS fica automaticamente disponível para todas as entidades com acesso a qualquer nó desse repositório, sendo necessário apenas o conhecimento do CID correspondente.

A figura do Operador de Dados (OD), presente nos regulamentos, não é tratada como uma entidade da arquitetura. Apesar de os ODs processarem dados que estão sob custódia dos respectivos CDs, o fazem em nome destes. Ou seja, em termos legais, não há que se falar em compartilhamento de dados de um CD com seus ODs. Por essa razão, o acesso de ODs a dados pessoais não faz parte do escopo deste trabalho.

4.3 Premissas

O projeto da arquitetura G4DPA assume algumas premissas necessárias ao seu funcionamento adequado. Essas premissas são elencadas a seguir.

1. As entidades APD, AC e GTW são confiáveis, ou seja, cumprem suas atribuições ho-

nestamente. Recomenda-se que um GTW seja operado pela entidade governamental responsável pela legislação de dados pessoais que, nesta abordagem, corresponde à APD. A confiabilidade da AC é necessária para que os mecanismos que serão examinados mais adiante sejam eficazes para identificar as entidades e seus papéis.

2. A informação de que um CC possui um dado pessoal de um TD sob sua custódia deve ser, por si só, vista como um dado pessoal. Por exemplo, ela poderia revelar que uma pessoa possui uma determinada doença por ser atendida em determinada instituição de saúde, ou revelar sua participação em um contrato sigiloso. Dessa forma, essas informações necessitam ser anonimizadas, tanto nos termos da GDPR [1] quanto da LGPD [2]. Assim, as requisições de CC e CR devem ser concentradas no GTW, que realiza a pseudoanonimização de dados antes de eles serem armazenados na BC.
3. A AC realiza um processo de validação da identidade, do papel e do segmento das entidades durante o processo de emissão dos certificados, de modo que os procedimentos adotados garantem que quem faz a solicitação é realmente quem diz ser.
4. Os certificados emitidos pela AC seguem o formato padrão x.509 (BOEYEN *et al.*, 2008), mas com a adição de extensões para:

    a) Indicar unicamente a entidade à qual pertence o certificado, e.g. $id = BR\_G0001$.
    b) Indicar o tipo de entidade entre GTW ($role = gateway$), CD ($role = controller$) ou TD ($role = person$).
    c) Indicar o segmento de atuação da entidade, e.g. $segment = government$. Como visto nas seções 3.3.2.3 e 3.3.3.3, existem bases legais específicas para que intuições de alguns segmentos, como *healthcare* e governo, realizem acessos a dados pessoais em determinadas situações. Esta extensão é especialmente útil para proporcionar acesso a dados pessoais de saúde em atendimentos médicos em situações emergenciais, ou seja, para processamentos visando proteger os interesses vitais do TD.

5. As chaves privadas devem ser de conhecimento e utilização exclusiva de seus titulares. Do contrário, os mecanismos de criptografia e assinatura digital, mostrados mais adiante, não seriam capazes de fornecer a proteção adequada às mensagens e aos arquivos.
6. Os dados armazenados na BC são de acesso público para leitura e apenas o GTW pode enviar registros a ela para gravação. Enquanto a gravação na BC proporciona um meio

[1] Considerando 26.
[2] Artigo 13.

de armazenamento imutável, a leitura é aproveitada como meio de notificação para a coordenação das entidades do sistema.

7. Cada CC e CR mantém nós próprios da rede IPFS para uso exclusivo. Isso é importante para proporcionar uma remoção eficaz dos arquivos do IPFS, como mostrado mais adiante na seção 4.5.3.
8. Todos os CD e o GTW possuem seus relógios razoavelmente sincronizados. Isso é importante para o bom funcionamento do mecanismo de proteção contra ataques *replay*, como será exposto na seção 4.6.1.

4.4 Restrições

Esta seção apresenta restrições da arquitetura, esclarecendo aquilo que ela não se propõe a fazer, assim como limitações de algumas técnicas utilizadas em seu desenvolvimento.

1. Não é o objetivo desta arquitetura realizar a auditoria completa sobre toda e qualquer operação sobre dados pessoais. Esclarecendo, esta arquitetura se propõe a fornecer um esquema de busca e compartilhamento de arquivos com dados pessoais entre controladores, com evidências de atendimento às normas de proteção de dados pessoais, especialmente a GDPR e a LGPD. Com isso, organizações podem demonstrar, através de evidências, que estão sendo diligentes no cumprimento dessas normas (KUNZ *et al.*, 2020) e, adicionalmente, um TD pode rastrear seus dados e revogar consentimentos concedidos, nos termos da lei. Todavia, os dados ainda podem ser obtidos ilegalmente, por exemplo, por operadores de hardware, em níveis mais baixos de acesso, enviados por outros meios diretamente entre controladores ou mesmo fotografados em uma tela por usuários operando sistemas restritos. De todo modo, o CD deve tomar todas as medidas ao seu alcance para proteger os dados pessoais sob sua custódia [3].
2. O detalhamento dos procedimentos adotados pela AC para emissão e gerência de certificados digitais não faz parte do escopo deste trabalho.
3. Existem algumas técnicas que podem permitir, direta ou indiretamente, a reidentificação

[3] GDPR, considerando 74 (tradução livre): [1]Devem ser estabelecidas a responsabilidade e a obrigação do controlador por qualquer processamento de dados pessoais realizado pelo controlador ou em nome do controlador. [2]Em particular, o controlador deve ser obrigado a implementar medidas adequadas e eficazes e ser capaz de demonstrar a conformidade das atividades de processamento com este regulamento, inclusive a eficácia das medidas. [3]Essas medidas devem levar em conta a natureza, o escopo, o contexto e as finalidades do processamento e o risco aos direitos e liberdades das pessoas físicas.

dos titulares de dados anonimizados ou pseudoanonimizados, em especial realizando o cruzamento de informações com outras fontes de dados.

4. Para fornecer pseudoanonimização aos dados armazenados no BC e proporcionar um meio de busca de dados, há uma senha simétrica única conhecida exclusivamente pelo GTW e pela APD, devendo ser mantida de forma segura [4].

4.5 Modelo de sistema

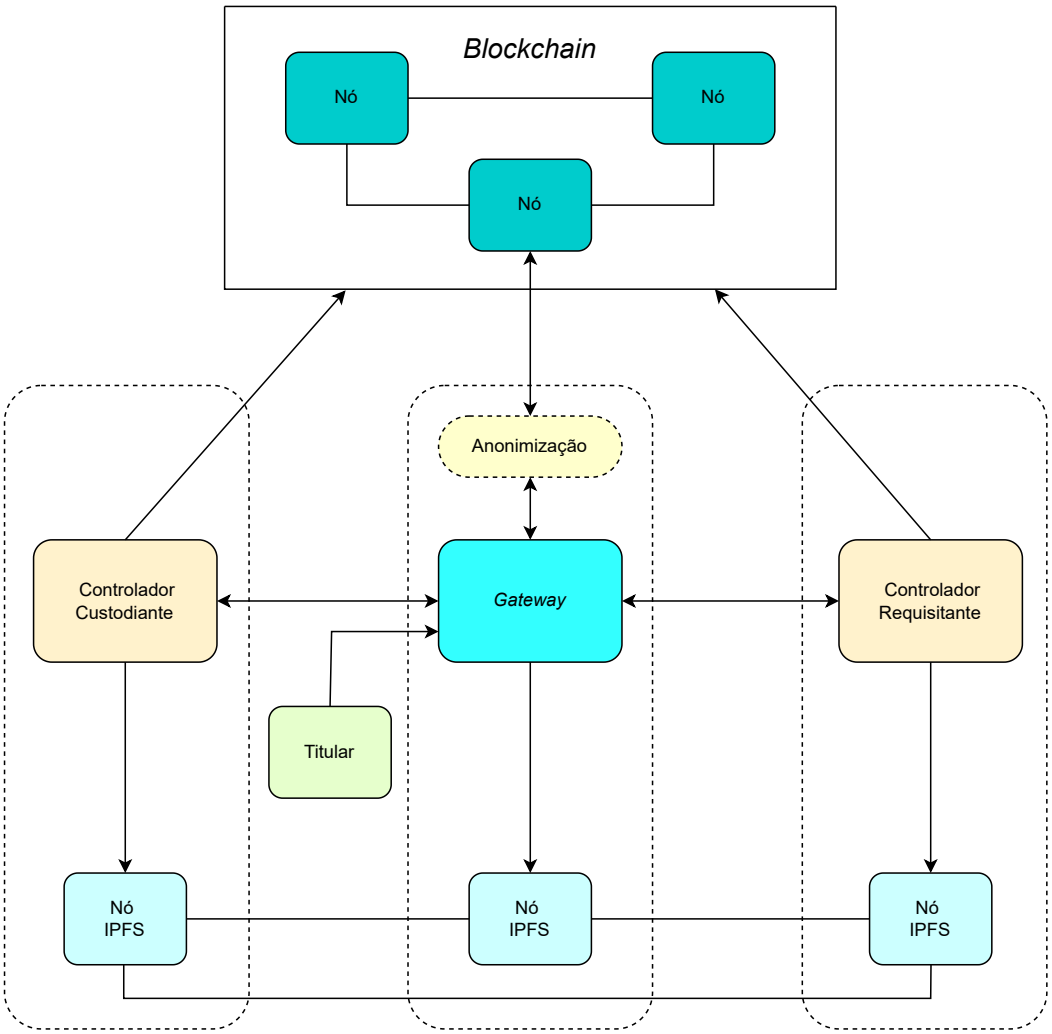
Esta seção descreve os mecanismos de operação da G4DPA que propiciam compartilhamento de arquivos com dados pessoais entre controladores. Um modelo de alto nível da arquitetura é ilustrado na figura 6.

Na G4DPA, uma **custódia** em um CC refere-se aos dados que representam a tutela desse CC sobre um arquivo com dados pessoais de um TD, devidamente fundamentada em alguma base legal. Inicialmente, o CC deve enviar ao GTW a relação de todas as suas custódias. Cada custódia compreende as seguintes informações:

1. Data de início: momento em que o CC informou ao GTW a custódia do arquivo.
2. Data de fim: momento em que o CC informou ao GTW o encerramento da custódia, ou seja, a eliminação do arquivo de suas bases.
3. Data de criação: momento em que o arquivo foi criado.
4. Identificador interno: código que identifica unicamente o arquivo para o CC.
5. Palavra-chave: um pequeno descritivo para facilitar as buscas por arquivos. Por exemplo, o arquivo pode se tratar de uma carteira de habilitação ou de um *curriculum vitae*. Exemplos na área de *healthcare* seriam um Eletrocardiograma (ECG), um Ecocardiograma (ECC), uma Ressonância Magnética (RM), um Tomografia Computadorizada (TC) e um Raio-X (RX).
6. Identificador do TD: código que indica o TD cujos dados estão contidos no arquivo.
7. Identificador do CC: código que indica o CC que detém a custódia.
8. *Hash*: uma representação única do conteúdo do arquivo através de uma *string* de tamanho fixo, obtida através de uma função matemática h para a qual, dado o valor de *hash* z, deve ser computacionalmente inviável achar x tal que $z = h(x)$ (PAAR; PELZL, 2010).

[4] Apesar de esse não ser o foco deste trabalho, o armazenamento dessa senha poderia ser realizado através de algum esquema de compartilhamento de segredo, como o *Shamir Secret Sharing Algorithm* (SSSA), de modo a evitar o comprometimento de uma senha armazenada.

Figura 6 – Modelo de alto nível da arquitetura G4DPA



Fonte: elaborada pelo autor.

9. Base legal: o dispositivo legal específico que serve de fundamento principal para o armazenamento do arquivo pelo CC. Como visto na seção 3.3, a GDPR elenca seis possíveis bases em seu artigo 6º, vide tabela 12, e a LGPD dez bases em seu artigo VII, vide tabela 14.
10. Tipo de dado: a classificação do arquivo armazenado, por exemplo, um exame de saúde, um arquivo bancário ou um contrato de seguro.
11. Modo de processamento: a indicação de como o CC realiza o processamento dos dados pessoais do arquivo, por exemplo armazenamento local, armazenamento em nuvem ou ainda por terceirização para um OD.
12. Propósito: a finalidade principal para a qual o arquivo está sendo armazenado, por exemplo,

para análise de *marketing*, compartilhamento com parceiros de negócios, tomada de decisão automatizada, transferência de dados internacional, pesquisa, atendimento eletivo de saúde ou atendimento emergencial de saúde.

Sempre que um CD passar a tutelar uma nova custódia, deve encaminhar essa informação ao GTW. Ao receber uma requisição de registro da nova custódia, o GTW realiza a pseudoanonimização de suas informações e, em seguida, as encaminha para armazenamento na BC. Como visto na seção 3.1, graças aos algoritmos de consenso empregados na BC, o registro distribuído é consistente em todos os nós e, assim, é possível executar várias instâncias do GTW concorrentemente, de modo a distribuir entre elas a carga de usuários.

Quando um controlador de dados deseja acessar um arquivo com dados de um TD, assume o papel de CR. Ele envia uma requisição de busca ao GTW, informando o identificador do TD, o período em que o arquivo deve ter sido criado, a palavra-chave desejada, a base legal para o seu acesso, o tipo do dado e o propósito de seu acesso. Antes mesmo de realizar a busca, o GTW realiza uma requisição à BC para armazenar um evento para registrar a solicitação de busca pelo CR. Após isso, a BC e o GTW realizam a busca de acordo com as permissões do CR. Os dados são retornados da BC para o GTW, que, por sua vez, realiza mais uma filtragem, enviando para o CR uma lista com as custódias encontradas e os respectivos ponteiros criptografados, limitando os resultados à data de criação do arquivo.

De posse dessas informações, o CR analisa as datas das custódias retornadas e solicita os arquivos desejados ao GTW, informando, mais uma vez, a base legal, o tipo de dado e o seu propósito, além do ponteiro criptografado para a custódia solicitada e uma chave pública aleatória que será utilizada para gerar uma chave simétrica compartilhada através do algoritmo Diffie-Hellman. A requisição é processada pelo GTW, desanonimizada e encaminhada à BC, para registro das seguintes informações:

1. Data de início: momento em que a requisição da custódia foi registrada na BC.
2. Data de fim: momento em que a requisição da custódia foi finalizada, seja porque o arquivo foi efetivamente compartilhado ou porque o CC negou o compartilhamento.
3. Recusa de compartilhamento: indica se o compartilhamento da custódia foi negado pelo CC. Essa negativa faz com que o GTW dispare um evento na BC, informando a APD que houve uma solicitação rejeitada. De posse dessa informação, APD pode iniciar um processo de investigação sobre a ocorrência e aplicar as punições aplicáveis ao CC, se a negativa for improcedente, ou ao CR, se procedente. Isso é importante para garantir

os acessos tempestivos aos dados, especialmente nos casos de proteção à saúde (seções 3.3.2.3 e 3.3.3.3).

4. Ponteiro para a custódia: registro indicando a localização das informações da custódia solicitada.
5. Identificador do CR: código que indica o CR que solicitou o compartilhamento do arquivo.
6. Base legal: o dispositivo legal específico que serve de fundamento para a solicitação de compartilhamento para CR.
7. Tipo de dado: a classificação do arquivo solicitado.
8. Propósito: a finalidade principal para a qual o arquivo está sendo requisitado.

Para oferecer um mecanismo de auditoria para terceiros, as entidades que transacionam com o GTW sempre recebem um recibo para as transações realizadas. Esse recibo, apesar de não ser capaz de identificar diretamente um registro na BC, pode ser decifrado pela APD com uma senha simétrica conhecida apenas por ela e pelo GTW, caso seja aventada alguma possibilidade de fraude por algum usuário do sistema.

Tabela 16 – Serviços oferecidos pelo contrato inteligente `G4DPA-SC`

Serviço	Chamador direto	Chamador indireto
`IsMemberOfGroup`	`G4DPA-SC`	✗
`AddControllerToGroup`	AC	✗
`RemoveControllerFromGroup`	AC	✗
`AddCustody`	GTW	CC
`CloseCustody`	GTW	CC
`RegisterSearch`	GTW	CR
`SearchCustody`	GTW	CR
`AddRequestSharing`	GTW	CR
`NotifySharingRequest`	GTW	✗
`NotifyRequester`	GTW	✗
`NotifyDeleteDataFromIPFS`	GTW	✗
`DenySharing`	GTW	CC
`EndRequest`	GTW	CR
`ListAllCustodiesOfDataOwner`	GTW	TD
`ListAllCustodiesFromDataControllerOfDataOwner`	GTW	CC, TD

✗: Não há.
Fonte: elaborada pelo autor.

A seguir, são elencados os recursos fornecidos pela G4DPA.

Inicialização do sistema. Para inicializar o sistema, a APD cria o contrato inteligente `G4DPA-SC`

na blockchain, oferecendo serviços como registro das custódias e compartilhamento de arquivos. A tabela 16 apresenta uma lista dos serviços disponibilizados pelo contrato. Entretanto, de modo a pseudoanonimizar os dados armazenados das custódias e prover busca sobre esses dados, esses serviços só estarão acessíveis através do GTW, com exceção dos serviços de credenciamento, acessíveis apenas à AC. Continuando a inicialização, a APD providencia a disponibilização pública de pelo menos um GTW. Para isso, ela emite, junto a AC, um certificado especial contendo a extensão $role = gateway$ para cada GTW, como visto na seção 4.3, e realiza a configuração e o provisionamento do GTW.

Credenciamento de TD, CC e CR. De modo a interagir com o GTW, as entidades TD, CC e CR necessitam se credenciar para uso do sistema. Para tanto, providenciam a emissão, junto à AC, de seus certificados digitais, que serão utilizados para comprovação de suas identidades e de seus segmentos de atuação. Após validar a identidade e os dados do solicitante, a AC emite o certificado assinado, contendo a chave pública e as extensões $role$ e *segment* adequadas, conforme indicado na seção 4.3. A seguir, a AC envia uma transação para a BC, informando que, a partir daquele momento, o solicitante faz parte do segmento de atuação que constará no certificado emitido. Isso é feito para que a *blockchain* contenha as informações necessárias à auditoria das operações realizadas. Por exemplo, caso um médico tenha seu registro profissional suspenso, não poderia estar realizando acessos a dados de saúde de pacientes durante a suspensão.

Autenticação de TD, CC, CR e GTW. Um aspecto importante da arquitetura proposta é a autenticação das partes envolvidas, pois serve como base para os mecanismos de controle de acesso e de auditoria. A G4DPA propõe uma abordagem de autenticação não interativa através do uso de assinaturas digitais. O GTW somente processa uma mensagem se tiver sido assinada por um certificado devidamente emitido pela AC, descartando-a, caso contrário. O mecanismo de criptografia e assinatura de mensagens será abordado em maiores detalhes na seção 4.5.1.

Notificações para CC e CR. O GTW recebe requisições diretamente de CC e CR, mas não há previsão de que CC e CR recebam requisições diretas de outras entidades, em conformidade com a diretriz de não exposição de APIs para acessos oriundos da internet, elencada na seção 4.1. Dessa feita, a transmissão de informações do GTW para CC e CR só é realizada de duas formas: i) através de uma resposta a uma requisição direta feita por CC ou CR ao GTW; ou ii) através da emissão de um evento pela BC. Caso seja emitido um evento,

ele só contém os dados necessários para informar uma das outras duas entidades de que o GTW necessita transmitir para ela uma instrução de coordenação. A partir do recebimento desse evento, a entidade deve realizar uma requisição direta ao GTW para consultar os detalhes adicionais, que só são fornecidos caso a requisição autentique sua identidade corretamente. Uma relação dos eventos que podem ser emitidos pela BC é mostrada na tabela 17.

Tabela 17 – Eventos emitidos pelo contrato inteligente `G4DPA-SC`

Evento	Destinatário	Quando
`LogNotifySharingRequest`	CC	CR solicita o compartilhamento de um arquivo
`LogNotifyDataSent`	CR	CC disponibilizou o arquivo solicitado no IPFS
`LogNotifyDeleteDataFromIPFS`	CC	CR informa que o arquivo foi decifrado com sucesso

Fonte: elaborada pelo autor.

Registro de custódias. A custódia de um arquivo contendo dados pessoais de um TD deve ser registrada na plataforma pelo CC através do GTW. Os arquivos permanecem armazenados pelo CC em sua infraestrutura, mas as informações de sua custódia estão pseudoanonimizadas na BC, permitindo a realização de operações de busca e solicitação de compartilhamento através do GTW. Qualquer CC devidamente identificado pode registrar uma custódia no sistema.

Busca por custódias. Quando um CR deseja acessar arquivos com dados pessoais de um TD armazenados em um CC, é necessário primeiro descobrir quais são as custódias desses arquivos. Para isso, deve ser realizada uma busca através do GTW informando os parâmetros desejados, como identificador do TD, data de início de custódia e tipo de dado. Isso porque, apesar de os dados armazenados na BC serem públicos, são armazenados anonimizados, não sendo possível ao CR realizar buscas diretamente. Ao receber a requisição de busca, o GTW executa uma transação para registrar essa operação na BC, para auditoria. Em seguida, o GTW envia a transação de busca propriamente dita e a BC realiza a busca sobre os dados do contrato inteligente com base nas permissões do CR. A partir dos resultados, o GTW procede à criptografia dos índices que indicam quais são as custódias antes de encaminhá-los em resposta ao CR. Se isso não fosse feito, o CR passaria a ter conhecimento do valor pseudoanonimizado do identificador do TD a partir da leitura da BC, comprometendo a segurança. Além desse índice criptografado, também são retornados os dados úteis para a busca, como a data de início da custódia e o tipo

de informação, para que o CR possa filtrar quais custódias são mais relevantes para sua necessidade.

Compartilhamento de arquivos. A partir da resposta recebida para a busca, o CR seleciona a custódia desejada. Por exemplo, ele pode optar pela custódia mais recente de um certo tipo de dado. Feito isso, o CR envia uma requisição ao GTW para que o arquivo desejado seja compartilhado. O GTW envia uma transação para execução de uma função na BC que realiza o registro dessa solicitação, e outra transação para emitir um evento de notificação para o CC que detém a custódia, de modo a avisá-lo de que há uma solicitação de compartilhamento de arquivo a ser atendida. O mecanismo de compartilhamento de arquivos será explicado em maiores detalhes na seção 4.5.2.

Cadastro prévio de base legal para acessos. Um TD pode autorizar que um CR possa acessar seus dados que estejam sob custódia de qualquer CC, desde que registre seu consentimento previamente na BC através do GTW. Além disso, um CC pode registrar na BC, também através do GTW, a existência de um contrato firmado com um CR para compartilhamento de dados de um TD. Após validação do registro do contrato pelo TD, o CR pode obter arquivos custodiados pelo CC relativos a esse TD. Note que, enquanto no primeiro caso o CR pode obter arquivos do TD a partir de qualquer CC, no segundo só pode obtê-los do CC participante do contrato, e apenas se validado pelo TD.

Encerramento de custódias. Um CC pode apagar um arquivo sob sua custódia por solicitação do TD ou por alguma questão legal ou contratual. Nesse caso, ele deve, imediatamente antes de apagá-lo, registrar o encerramento da custódia, de modo que ela não será mais listada em buscas. Desse modo, como a informação sobre a custódia é mantida na BC, uma auditoria pode verificar, para um determinado período, se o CC esteve de fato responsável pelo arquivo, bem como avaliar a conformidade das operações de busca e de compartilhamento realizadas sobre essa custódia, incluindo quando ocorreram e com quais partes envolvidas.

Revogação de credenciais. Em alguns casos, pode ser necessário revogar credenciais por motivos como perda de licença para o exercício de uma atividade ou comprometimento do certificado digital associado. Para isso, a AC deve incluir o certificado digital afetado em sua lista de certificados revogados e chamar a função `revokeCredentials` do contrato inteligente `G4DPA-SC`, registrando, na BC, que essa credencial está inativa a partir desse momento, mas mantendo-a para fins de auditoria. Isso garante que o GTW possa verificar

que o certificado foi revogado e passe a recusar requisições com ele assinadas.

Listagem de custódias para um TD. O artigo 15 da GDPR e o artigo 18, II da LGPD preveem o direito de o TD obter do CD os seus dados pessoais por ele tratados, conforme apontado nas seções 3.3.2.2 e 3.3.3.2, respectivamente. Dessa maneira, o TD, mediante requisição ao GTW, pode obter a relação de custódias referentes à sua pessoa, vez que essa relação é mantida na BC. Para isso, todavia, o TD deve estar credenciado para uso do sistema e, portanto, assinando sua requisição com o seu certificado digital devidamente emitido pela AC. Uma vez com a relação de suas custódias, o TD pode também requisitar os arquivos relacionados através do GTW, que deverão ser encaminhados pelos CCs detentores das respectivas custódias.

Solicitação de esquecimento. O direito ao esquecimento é um direito do TD previsto no artigo 17 da GDPR, vide seção 3.3.2.2, e no artigo 18, VI, da LGPD, vide seção 3.3.3.2. Na G4DPA, o TD pode encaminhar uma requisição ao GTW solicitando a eliminação de seus dados pessoais de um CC. Caso isso ocorra, o GTW deve encerrar as custódias do TD e, ao mesmo tempo, notificar o CC de que ele deve realizar, de modo imediato, a exclusão dos arquivos relacionados a essas custódias. Apesar de não haver meio de comprovar diretamente a eliminação, a APD pode proceder a auditorias nos sistemas do CC e verificar se a solicitação do TD foi de fato atendida.

Solicitação de retificação. Outro direito do TD previsto na GDPR em seu artigo 16 e na LGPD em seu artigo 18, III, é o direito à retificação de dados incorretos ou desatualizados, como visto nas seções 3.3.2.2 e 3.3.3.2. Caso o TD envie uma requisição de retificação ao GTW, deverá juntar o arquivo com a informação correta. A custódia do arquivo incorreto é encerrada, e o CC é notificado de que deve proceder à retificação do arquivo em seu sistema. Uma vez que tenha realizado a correção, o CC deve então informar nova custódia ao GTW. Novamente, a APD pode auditar os sistemas do CC verificando o atendimento tempestivo ou não da solicitação de retificação do TD.

Realização de auditoria. Como mencionado nos últimos itens, a APD deve realizar auditorias sobre as operações realizadas pelas entidades participantes. Em especial, as operações de solicitação de esquecimento e de retificação somente podem ser totalmente garantidas mediante auditorias internas nos sistemas dos CCs. Para tanto, as senhas utilizadas pelo GTW nos mecanismos de criptografia e anonimização também são conhecidas pela APD, de modo que ela é capaz de decifrar os recibos das operações dos CDs e saber a quais

partes se refere cada custódia armazenada na BC.

4.5.1 Criptografia e assinatura de mensagens e autenticação mútua

Assinaturas digitais são comumente realizadas utilizando algoritmos de criptografia assimétrica, em que, de forma simplificada, o signatário cifra a mensagem com sua chave privada para que só possa ser lida se decifrada com sua chave pública. Todavia, esses algoritmos são computacionalmente intensivos, não sendo, portanto, a opção mais adequada para criptografar um grande volume de dados (PAAR; PELZL, 2010). Por essa razão, costuma-se realizar as assinaturas digitais apenas sobre o *hash* dos dados.

Já para proporcionar sigilo sobre os dados, os algoritmos de criptografia assimétrica costumam ser usados simplesmente para transmitir uma pequena mensagem contendo uma chave simétrica de sessão, em uma abordagem chamada de criptografia híbrida (SMART, 2016). Nesse caso, a criptografia simétrica é aplicada sobre os dados, e apenas a respectiva chave é cifrada com a chave pública do destinatário.

A G4DPA adota um esquema de criptografia híbrida para assegurar a confidencialidade e a integridade dos dados trocados em todas as requisições realizadas por TDs, CCs e CRs ao GTW, bem como em suas respostas. Essa abordagem incorpora ideias apresentadas em (DAVIS, 2001), que propõem a criação de uma dependência entre as camadas interna e externa de criptografia quando a assinatura digital e a criptografia de chave pública são combinadas (*encrypt & sign*). Tal dependência permite a detecção de modificações maliciosas na camada externa, fortalecendo a segurança do protocolo de troca de mensagens. A notação adotada ao longo deste trabalho para representar operações criptográficas é a mesma do artigo original e é apresentada na tabela 18.

Tabela 18 – Notação utilizada para representar operações criptográficas

Notação	**Significado**
(A, a)	Par de chaves, em que A denota a chave pública e a indica a chave privada
$cert_A$	Certificado digital contendo a chave pública A de um usuário
$\{M\}^K$	A mensagem M é cifrada com a chave K (simétrica ou pública, dependendo do contexto)
$\{M\}^{-K}$	A mensagem M é decifrada com a chave K (simétrica ou pública, dependendo do contexto)
$\{M\}^k$	A mensagem M é cifrada com a chave k (privada)
$\{M\}^{-k}$	A mensagem M é decifrada com a chave k (privada)
$\#M$	*Hash* da mensagem M

Fonte: elaborada pelo autor com base em (DAVIS, 2001).

Um exemplo de ataque comum a um modelo mais ingênuo de criptografia e assinatura seria o seguinte. Suponha que Alice tenha enviado para Eva uma mensagem "*gossip*" cifrada com a chave pública E de Eva, juntamente com a assinatura digital dessa mensagem, cifrando o *hash* de *gossip* com a chave privada a de Alice:

$$Alice \rightarrow Eva : \{ \{ gossip \}^{E}, \{ \#gossip \}^{a} \} \tag{4.1}$$

Ao receber a mensagem, Eva resolve forjar o envio de *gossip* de Alice para Bob. Para tanto, basta que Eva decifre a mensagem com sua chave privada e e, em seguida, a cifre com a chave pública B de Bob. Para todos os efeitos, Bob seria levado a crer que Alice foi a responsável pelo envio de *gossip*:

$$Eva \rightarrow Bob : \{ \{ gossip \}^{B}, \{ \#gossip \}^{a} \} \tag{4.2}$$

A fim de explicar o esquema de criptografia e assinatura utilizado nesta arquitetura, suponha agora que Alice deseja enviar uma mensagem *msg* com segurança para Bob e que ela conhece o certificado digital $cert_B$ contendo a chave pública de Bob. Já Bob não tem conhecimento da chave pública A de Alice, mas confia no certificado da autoridade certificadora que assinou o certificado $cert_A$ de Alice. Inicialmente, Alice gera uma chave de sessão K, simétrica e aleatória, e a utiliza para produzir a mensagem criptografada msg':

$$msg' = \{ msg \}^{K} \tag{4.3}$$

Alice precisa transmitir msg' para Bob. Mas, para que ele seja capaz de decifrar a mensagem, Alice também deve transmitir a chave K. Então, Alice usa B para cifrar K, e realiza a assinatura digital do *hash* de K usando a. Para que Bob possa obter a chave simétrica K, ela também envia $cert_A$:

$$M = \{ cert_A, \{K\}^{B}, \{\#K\}^{a} \} \tag{4.4}$$

Segundo (DAVIS, 2001), essa seria uma construção ingênua com a técnica de criptografar e assinar, possibilitando ataques como o exemplificado anteriormente. Uma das

formas de corrigir esse problema é criptografar algo que identifique o remetente junto com a chave K (4.5) e, em seguida, assinar K por fora do texto cifrado (4.6):

$$X = \{ \{ \#cert_A, K \}^B, \#K \} \tag{4.5}$$

$$sign_a(X) = \{ X \}^a \tag{4.6}$$

Com essa modificação, a mensagem completa que Alice enviaria para Bob seria:

$$M' = \{ cert_A, X, sign_a(X), msg' \} \tag{4.7}$$

Ou, em sua forma expandida:

$$M' = \{ cert_A, \{ \{ \#cert_A, K \}^B, \#K \}, \{ \{ \#cert_A, K \}^B, \#K \}^a, \{ msg \}^K \} \tag{4.8}$$

Essa alteração entrelaça a camada criptográfica externa à camada interna, impedindo que um atacante substitua a assinatura de Alice. Além disso, a criptografia de $\#cert_A$ junto a K com a chave pública B comprova que foi realmente Alice quem realizou a operação de criptografia, uma vez que a assinatura também contém $\#cert_A$. A assinatura externa prova, ainda, que Alice de fato produziu msg', já que a chave K corresponde ao texto cifrado com a criptografia assimétrica. Assim, com essa construção, Bob pode verificar que Alice é a autêntica remetente da mensagem, e Alice tem a garantia de que apenas Bob terá acesso a seu conteúdo. Ademais, por conta da assinatura digital, Alice e Bob têm a garantia de que *msg* será recebida íntegra. Maiores detalhes dessa construção são fornecidos em (DAVIS, 2001). Mais adiante, na seção 4.6, será mostrado como essa construção contribui para os objetivos elencados na seção 4.1.

4.5.2 Mecanismo de compartilhamento de arquivos

Esta seção detalha o processo por meio do qual o CR solicita e obtém arquivos com dados pessoais de um TD na G4DPA. Esse processo é dividido em quatro fases: a solicitação do arquivo; a disponibilização do arquivo; a obtenção do arquivo; e a finalização da solicitação. Para cada fase, são apresentados os diagramas de sequência dos passos realizados e os diagramas de

coreografia em BPMN (OMG, 2010), a fim de melhor explicar as interações entre os componentes da arquitetura durante o processo de compartilhamento. A tabela 19 apresenta a relação de símbolos utilizados na explanação das fases.

Para a descrição dos passos em cada fase, suponha que o CR realizou uma busca por dados de um TD. A partir das informações recebidas, o CR deseja obter o arquivo referenciado pela custódia $\mathscr{C}$. Todas as requisições de CC e CR para o GTW e as respectivas respostas são realizadas por meio do mecanismo apresentado na seção 4.5.1. Desse modo, sempre que um passo tratar de uma dessas requisições ou de suas respostas, estará implícito que a mensagem será confidencial, estará íntegra e que seu destinatário e remetente estarão autenticados mutuamente. Caso haja algum problema com essas características da mensagem, ela será descartada e a ação solicitada rejeitada.

Outra consideração importante é que os dados enviados do GTW para a BC estarão pseudoanonimizados, sendo a pseudoanonimização revertida pelo GTW quando do recebimento dos dados de volta da BC.

Tabela 19 – Simbologia utilizada na explanação do mecanismo de compartilhamento

Símbolo	**Definição**
$\mathscr{C}$	Custódia pseudoanonimizada
DH_R	Chave pública do CR para troca de chaves através do algoritmo Diffie-Hellman
dh_R	Chave privada do CR para troca de chaves através do algoritmo Diffie-Hellman
DH_C	Chave pública do CC para troca de chaves através do algoritmo Diffie-Hellman
dh_C	Chave privada do CC para troca de chaves através do algoritmo Diffie-Hellman
S	Chave de sessão compartilhada entre o CR e o CC, obtida através do algoritmo Diffie-Hellman
$\mathscr{F}$	Identificador interno pseudoanonimizado, referente ao arquivo sendo compartilhado
F	Arquivo sendo compartilhado
F'	Arquivo sendo compartilhado, criptografado com a chave de sessão S
$\mathscr{E}_1$	Evento para notificar o CC de que foi solicitado um compartilhamento de arquivo
$\mathscr{E}_2$	Evento para notificar o CR de que um arquivo foi compartilhado via IPFS
$\mathscr{E}_3$	Evento para notificar o CC de que deve remover um arquivo do IPFS
$\mathscr{I}_R$	Identificador de coordenação para o CR acompanhar o processo de compartilhamento
$\mathscr{I}_C$	Identificador de coordenação para o CC acompanhar o processo de compartilhamento
$\mathscr{P}$	Código de prova de acesso do CR ao arquivo compartilhado

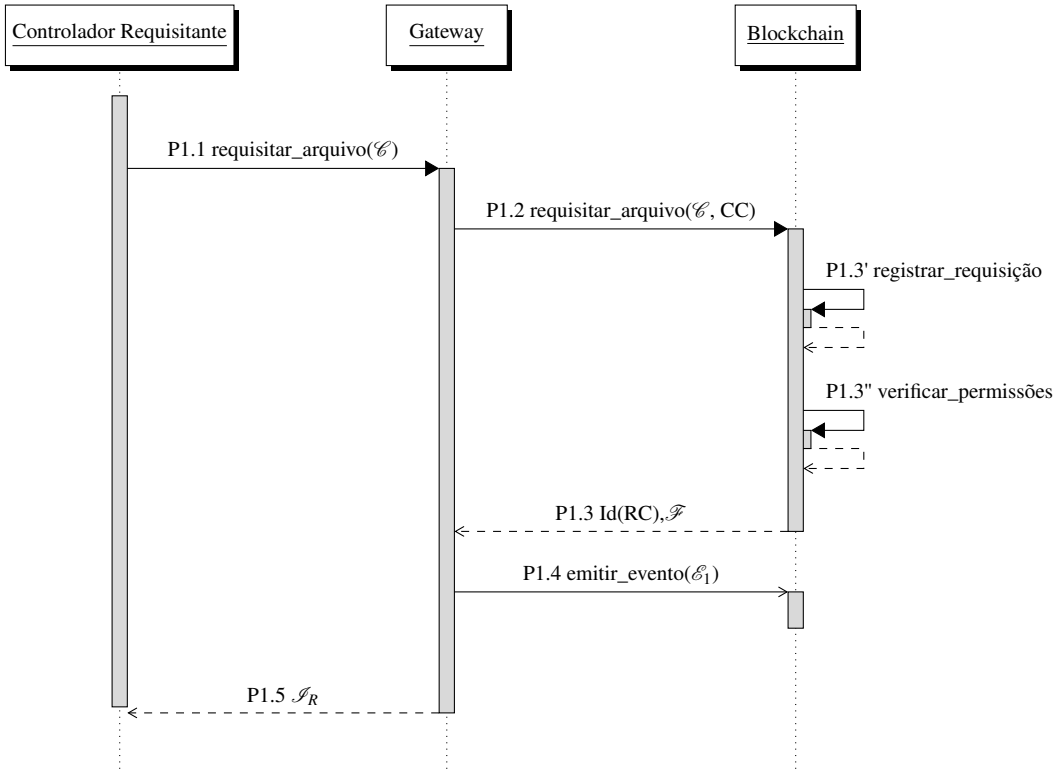
Fonte: elaborada pelo autor.

Fase 1 – Solicitação do arquivo

Na fase de solicitação do arquivo, o CR solicita ao GTW um arquivo referenciado pela custódia $\mathscr{C}$. O CR desconhece a identidade do CC que registrou essa custódia, e o GTW irá reverter a anonimização dos dados e encaminhar o evento $\mathscr{E}_1$ para o CC, convocando-o a compartilhar o arquivo.

Os passos realizados nesta fase são enumerados e detalhados a seguir. As figuras 7 e 8 exibem os diagramas de sequência e de coreografia relativos à fase de solicitação de arquivo, respectivamente.

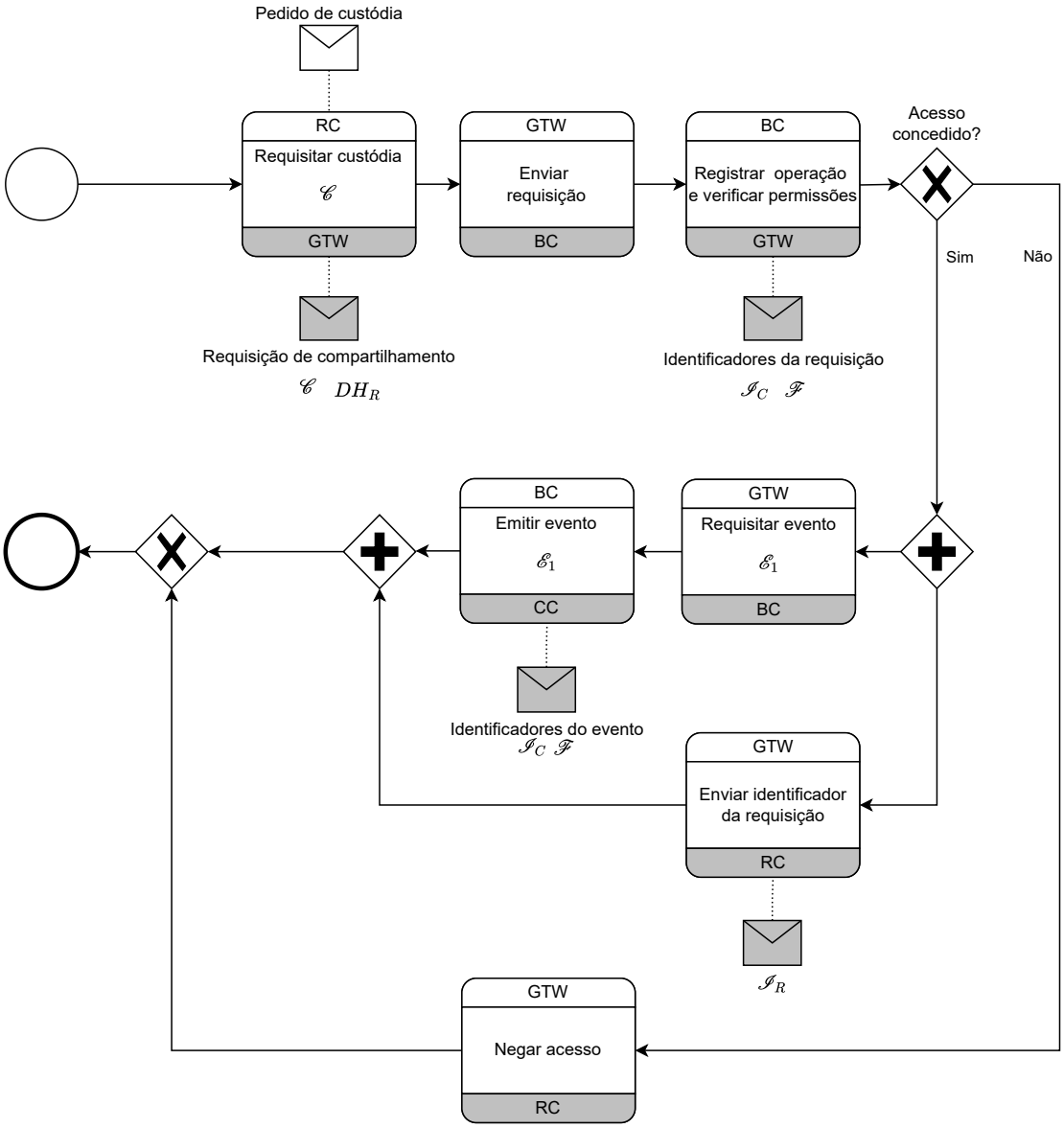
Figura 7 – Diagrama de sequência da Fase 1 – Solicitação do arquivo



Fonte: elaborada pelo autor.

P1.1. O CR envia ao GTW uma requisição de compartilhamento do arquivo referenciado pela custódia $\mathscr{C}$, junto com a chave pública do par de chaves (DH_R, dh_R). Este par é gerado de forma aleatória para cada requisição de compartilhamento e será utilizado para estabelecer uma chave secreta compartilhada entre o CR e o CC através do algoritmo Diffie-Hellman para troca de chaves (DIFFIE; HELLMAN, 1976). A chave privada do par permanece em segredo com o CR.

Figura 8 – Diagrama de coreografia da Fase 1 – Solicitação do arquivo



Fonte: elaborada pelo autor.

P1.2. O GTW envia para a BC uma transação indicando que o CR solicitou o compartilhamento da custódia $\mathscr{C}$.

P1.3. A BC realiza o registro dessa solicitação para fins de auditoria, e verifica se há alguma base legal que fundamente o acesso de CR a $\mathscr{C}$. Essa base deve ter sido previamente registrada na BC, podendo ser, por exemplo, um consentimento do TD ou um contrato entre o CR e o CC, ou pode também ser uma situação em que o CR possui prerrogativa de acesso, como no caso de um ente governamental (certificado com a extensão $segment = government$) ou de uma entidade médica (certificado com a extensão $segment = healthcare$) prestando um atendimento de saúde emergencial. Se houver tal base, o GTW recebe o identificador do CC que registrou $\mathscr{C}$ e o identificador interno $\mathscr{F}$ do arquivo, ambos valores pseudoanonimizados. Não havendo tal base, ele deve negar o acesso do CR ao arquivo referenciado por $\mathscr{C}$.

P1.4. Para que o CC saiba que ele deve compartilhar um arquivo, o GTW envia à BC uma transação para a emissão de um evento $\mathscr{E}_1$, contendo o identificador do CC e um identificador aleatório $\mathscr{I}_C$ gerado pelo GTW, que o utilizará para coordenar as ações do CC neste processo de compartilhamento. Da mesma forma, o CC usará $\mathscr{I}_C$ para reportar ao GTW suas ações ao longo das iterações seguintes deste compartilhamento.

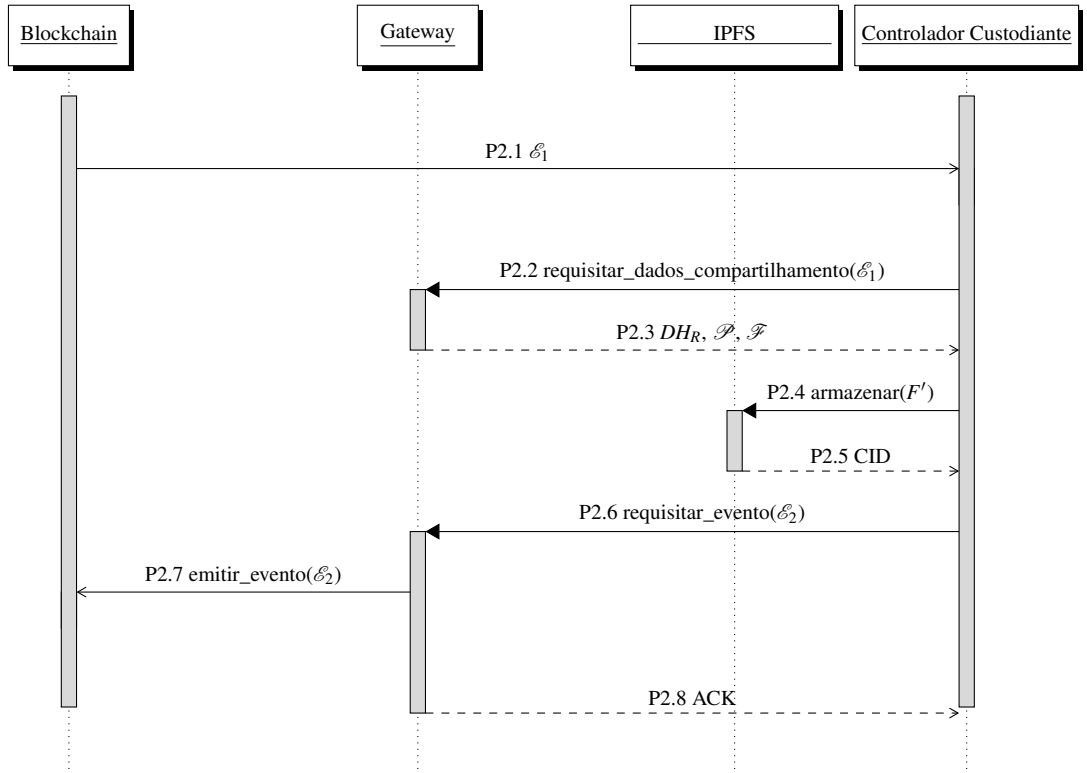
P1.5. De maneira similar, o GTW também gera um identificador $\mathscr{I}_R$ para coordenar as ações do CR, que será usado também para que o CR acompanhe o processo de compartilhamento e realize reportes. $\mathscr{I}_C$ e $\mathscr{I}_R$ são identificadores distintos para evitar que CC e CR possam utilizar essa informação para identificar um ao outro. O GTW responde ao CR informando o valor $\mathscr{I}_R$.

Fase 2 – Disponibilização do arquivo

Na fase de disponibilização do arquivo, o CC encontra-se escutando eventos do tipo $\mathscr{E}_1$ que contenham o seu identificador, e, por essa via, é notificado pela BC. A partir dessa notificação, o CC recupera, a partir do GTW, o identificador do arquivo que deve ser compartilhado, que será, então, armazenado no nó IPFS privativo do CC. Quando informado pelo CC de que o arquivo encontra-se disponível para recuperação a partir do IPFS, o GTW solicita à BC a emissão de um evento $\mathscr{E}_2$ para cientificar o CR.

Os passos desta fase estão representados no diagrama de sequência da figura 9, e são listados e explicados a seguir. A figura 10 exibe as interações entre os componentes envolvidos.

Figura 9 – Diagrama de sequência da Fase 2 – Disponibilização do arquivo



Fonte: elaborada pelo autor.

P2.1. O CC verifica que foi emitido o evento $\mathscr{E}_1$ de solicitação de compartilhamento através de seu identificador.

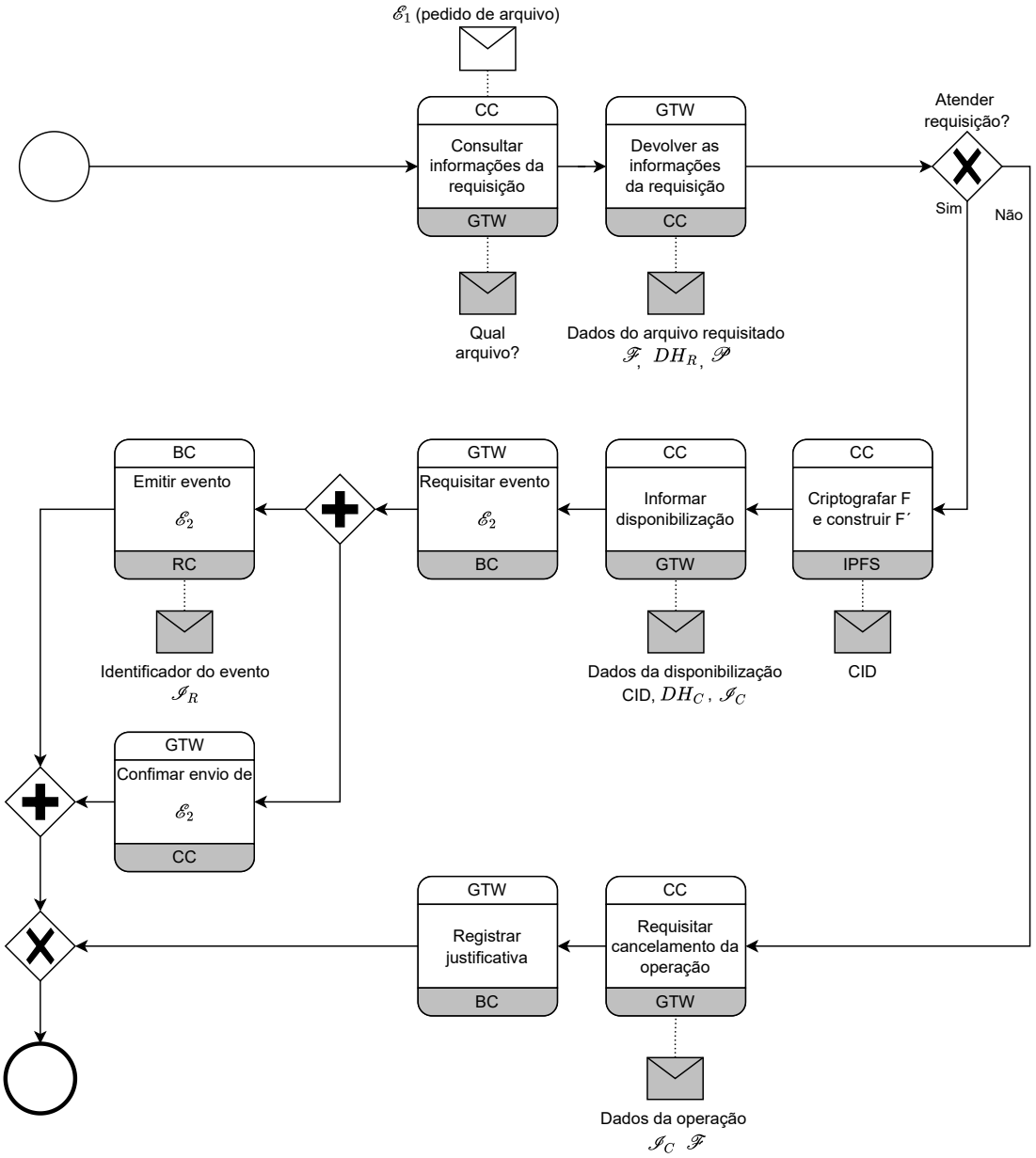
P2.2. A partir do evento $\mathscr{E}_1$, o CC envia uma requisição com o identificador $\mathscr{I}_C$ ao GTW para obter as informações acerca do compartilhamento de arquivo solicitado.

P2.3. Em resposta, o GTW envia ao CC o identificador interno do arquivo $\mathscr{F}$, a chave pública DH_R e um código aleatório $\mathscr{P}$ que será utilizado mais adiante como prova de que o CR acessou um arquivo compartilhado pelo CC.

P2.4. A partir do identificador interno $\mathscr{F}$, o CC localiza o arquivo F a ser compartilhado e gera o seu par de chaves (DH_C, dh_C). A chave de sessão S, compartilhada com CR, é então calculada a partir de dh_C e DH_R. CC gera o arquivo $F' = \{ \mathscr{P}, \{ F \}^S \}$, contendo a prova de compartilhamento, a ser apresentada mais adiante pelo CR ao GTW, e o arquivo F criptografado com a chave compartilhada. Por fim, o CC armazena F' em um nó IPFS em seu domínio.

P2.5. O nó IPFS sob domínio do CC retorna o CID correspondente ao arquivo F' que foi armazenado.

Figura 10 – Diagrama de coreografia da Fase 2 – Disponibilização do arquivo



Fonte: elaborada pelo autor.

P2.6. O CC envia uma requisição para o GTW informando que o arquivo solicitado foi disponibilizado no IPFS. A requisição contém o identificador $\mathscr{I}_C$, a chave pública DH_C e o CID do arquivo.

P2.7. O GTW envia à BC uma transação para a emissão de um evento $\mathscr{E}_2$, contendo o identificador do CR e o identificador de acompanhamento do processo de compartilhamento $\mathscr{I}_R$.

P2.8. O CC recebe uma resposta de confirmação do GTW.

Fase 3 – Obtenção do arquivo

Nesta fase, o CR recebe um evento $\mathscr{E}_2$ com seu identificador a partir da escuta aos eventos emitidos pela BC. O CR, então, solicita ao GTW o endereço do arquivo compartilhado, e tenta realizar o seu *download* a partir de seu nó IPFS e, em seguida, comprova que baixou o arquivo apresentando $\mathscr{P}$ ao GTW, recebendo as informações necessárias para decifrar o arquivo.

Os detalhes dos passos desta fase estão descritos a seguir, estando representados no diagrama de sequência da figura 11. Já a figura 12 retrata as interações entre os componentes.

P3.1. O CR verifica que foi emitido o evento $\mathscr{E}_2$ com o seu identificador, indicando que há um novo arquivo com ele compartilhado.

P3.2. A partir do evento $\mathscr{E}_2$, o CR envia uma requisição com o identificador $\mathscr{I}_R$ ao GTW para obter as informações acerca do arquivo que foi compartilhado.

P3.3. O GTW envia o CID do arquivo em resposta à requisição do CR.

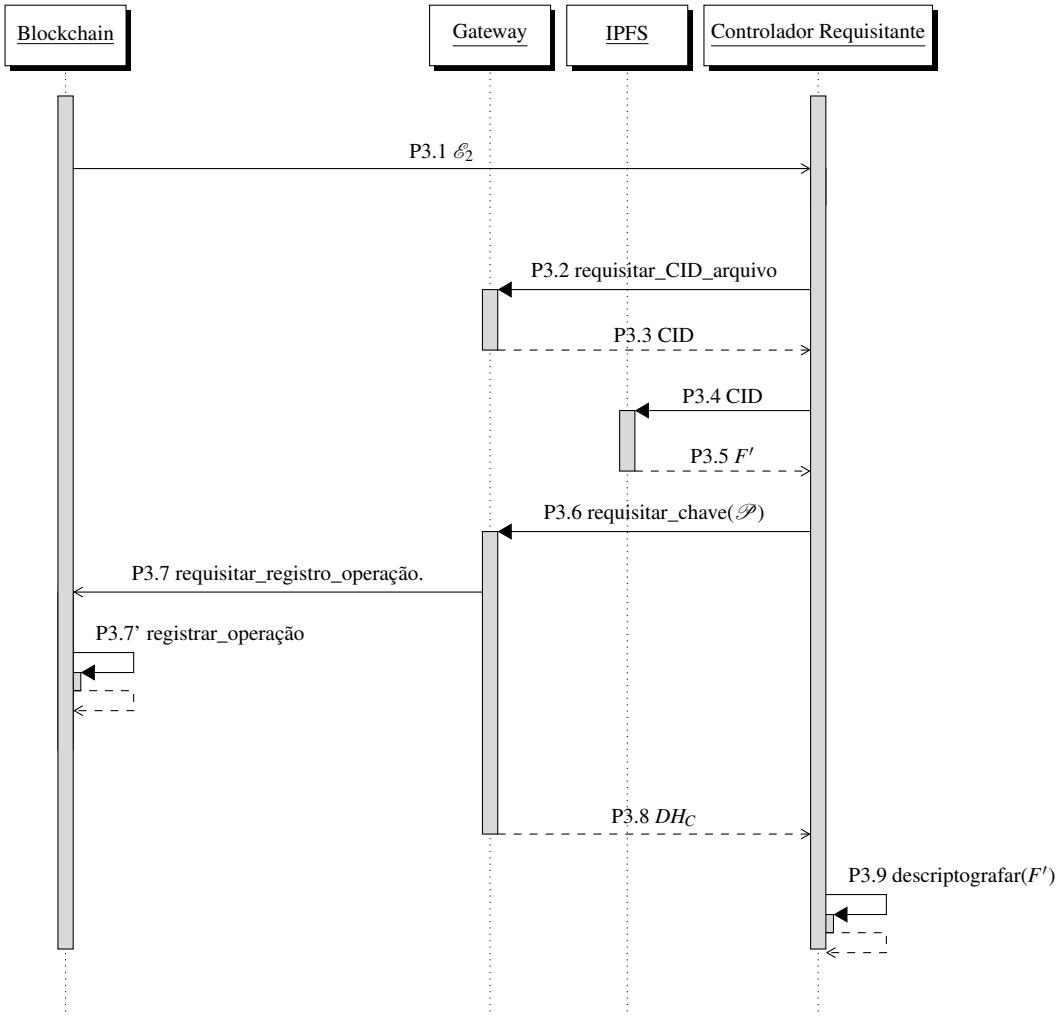
P3.4. O CR solicita o arquivo com o endereço CID ao nó do IPFS em seu domínio.

P3.5. O IPFS devolve o arquivo F' ao CR.

P3.6. De posse do arquivo F', o CR obtém o código de comprovação $\mathscr{P}$ e o arquivo criptografado $\{ F \}^S$. Para obter a chave compartilhada S, é necessário obter a chave pública do CC, que é de conhecimento do GTW. Assim, CR envia uma requisição ao GTW informando o identificador $\mathscr{I}_R$ e o código de comprovação $\mathscr{P}$.

P3.7. O GTW verifica se o código de comprovação $\mathscr{P}$, recebido de CR, está correto. Se não estiver, ele registrará uma não conformidade na BC. Estando correto o código, ele envia à BC uma transação indicando que CR recebeu o arquivo referido pela custódia $\mathscr{C}$.

Figura 11 – Diagrama de sequência da Fase 3 – Obtenção do arquivo



Fonte: elaborada pelo autor.

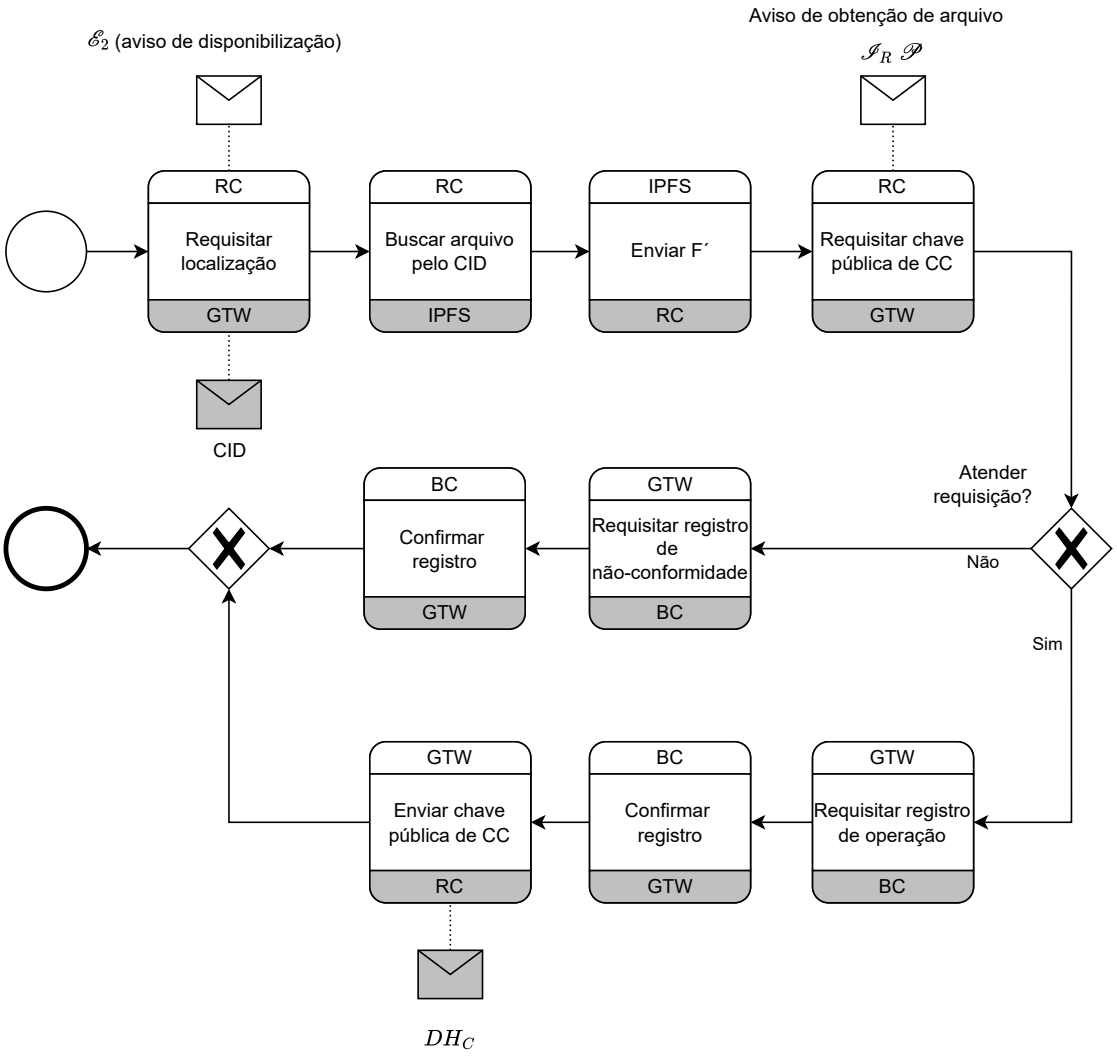
P3.8. O GTW responde ao CR com a chave pública DH_C do CC.

P3.9. O CR calcula a chave compartilhada S a partir de $\{ dh_R, DH_C \}$ e, em seguida, obtém o arquivo $F = \{ \{ F \}^S \}^{-S}$.

Fase 4 – Finalização da solicitação

Durante a fase de finalização, o GTW verificará que o CR recebeu o arquivo correto e, após, coordenará as ações do CC e do CR para realizar a remoção dos arquivos dos seus nós IPFS, o que será apresentado na seção 4.5.3. O CR recebe a diretiva para remover os arquivos através da própria resposta à sua requisição, enquanto que o GTW solicita à BC a emissão de um evento $\mathscr{E}_3$ para notificar o CC. Por fim, o GTW realiza a verificação de exclusão do arquivo do IPFS através de consulta ao seu nó.

Figura 12 – Diagrama de coreografia da Fase 3 – Obtenção do arquivo



Fonte: elaborada pelo autor.

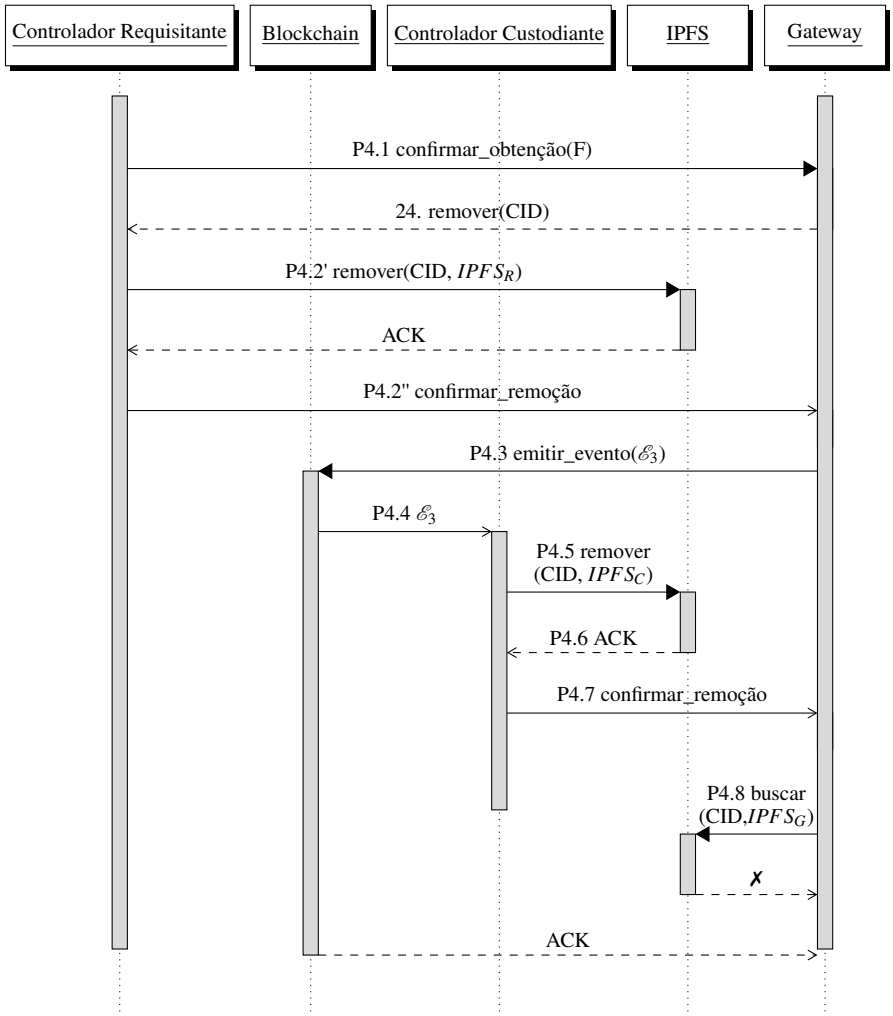
Os passos desta fase são detalhados a seguir, podendo ser observados no diagrama de sequência da figura 13. Ademais, as interações entre os componentes arquiteturais envolvidos podem ser analisadas na figura 14.

P4.1. O CR calcula #F e encaminha uma requisição para o GTW informando $\mathscr{I}_{\mathscr{R}}$ e #F.

P4.2. O GTW verifica se o valor de #F é o mesmo que consta na custódia $\mathscr{C}$. Se for, é porque o arquivo que o CR recebeu é o mesmo que o CC possui em custódia. O GTW retorna uma mensagem para o CR orientando a remoção imediata de F' de seu nó IPFS.

P4.3. O GTW envia à BC uma transação para a emissão de um evento $\mathscr{E}_3$, contendo o identificador do CC e o identificador de acompanhamento do processo de compartilhamento $\mathscr{I}_C$.

Figura 13 – Diagrama de sequência da Fase 4 – Finalização da solicitação



Fonte: elaborada pelo autor.

P4.4. O CC percebe que foi emitido o evento $\mathscr{E}_3$ com o seu identificador.

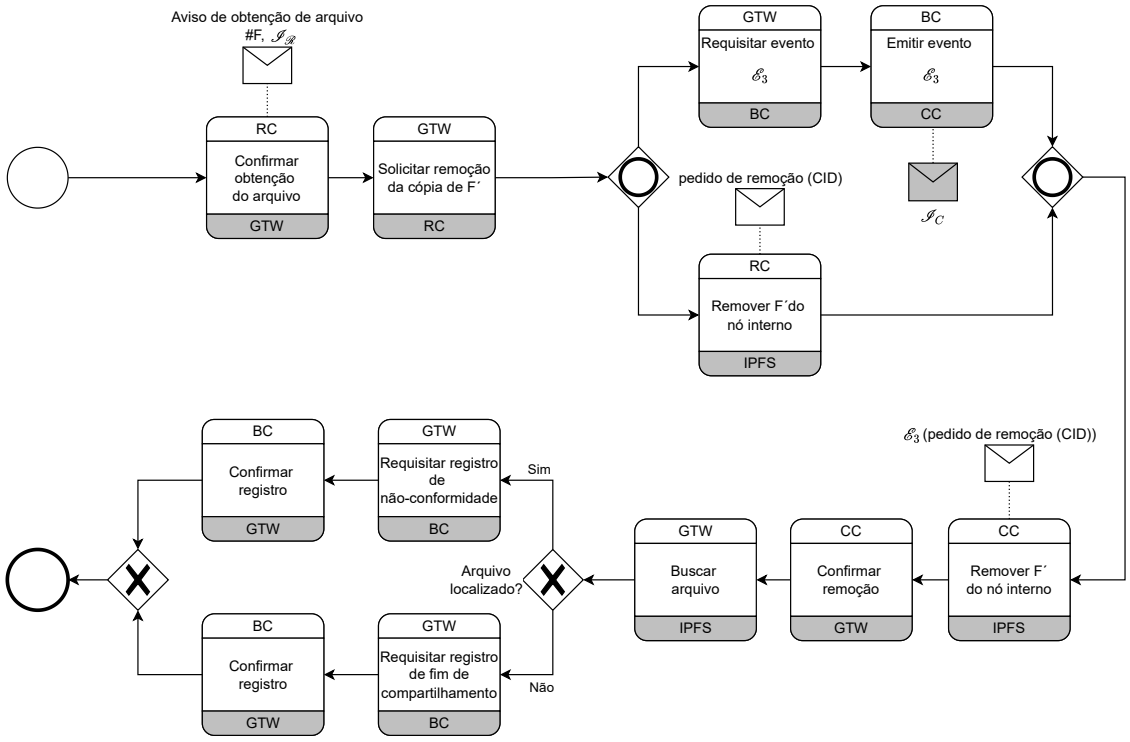
P4.5. O CC envia uma solicitação de remoção do arquivo ao seu nó IPFS.

P4.6. O nó IPFS do CC confirma a remoção.

P4.7. O CC envia ao GTW uma requisição informando que o arquivo foi removido do seu nó IPFS.

P4.8. Para confirmar que CC e CR removeram o arquivo de seus nós IPFS, o GTW realiza uma busca pelo arquivo F' através do CID em seu próprio nó IPFS. Caso não seja encontrado, o arquivo foi de fato removido. Do contrário, será gerada uma não conformidade. Por fim, o encerramento da requisição é registrado na BC.

Figura 14 – Diagrama de coreografia da Fase 4 – Finalização da solicitação



Fonte: elaborada pelo autor.

4.5.3 Mecanismo de remoção de arquivos do IPFS

Como exposto na seção 3.2, não há, no IPFS, um mecanismo próprio e nativo que possibilite a exclusão de um arquivo da rede a partir do seu CID. Entretanto, se não for realizado o *pinning* do arquivo em um nó do IPFS, ele será excluído na próxima execução do coletor de lixo desse nó. Arquivos que sofreram *pinning*, por outro lado, não são excluídos pelo coletor de lixo e estão disponíveis para acesso a qualquer outro nó da rede a partir do seu CID.

Considerando o funcionamento assíncrono do mecanismo de compartilhamento de arquivos a partir das notificações por eventos na BC, é necessário garantir que CR tenha acesso ao arquivo compartilhado por CC através do IPFS, mesmo que o coletor de lixo execute. Portanto, o CC deve inicialmente realizar o *pinning* do arquivo, evitando sua remoção nos ciclos de coleta de lixo.

Todavia, para evitar que arquivos já desnecessários causem inchaço nos nós, elevando os custos de operação da solução, e ainda para evitar que dados pessoais, embora criptografados, estejam disponíveis para acesso por qualquer um a partir do CID, com possível prejuízo à segurança, é interessante que seja feita a limpeza dos arquivos que já foram obtidos pelo CR e

que, portanto, não necessitam mais estar disponíveis no IPFS. Para tanto, são necessários dois passos: i) remover o *pinning* do arquivo do nó IPFS e, em seguida, ii) executar o coletor de lixo.

Uma primeira opção seria remover o *pinning* quando necessário, e aguardar a execução periódica automática do coletor de lixo. Apesar de alguns clientes IPFS apresentarem um período padrão para isso (e.g. o Kubo realiza um ciclo a cada uma hora), não existe exigência no protocolo IPFS de que a coleta de lixo seja realizada. Dessa feita, esta arquitetura sugere que a coleta de lixo seja executada também sob demanda, e, após sua realização, haja o reporte ao GTW de forma que ele possa, ao final, verificar se o arquivo foi removido da rede.

4.6 Análise de segurança

Esta seção apresenta uma análise informal sobre como o mecanismo de compartilhamento de arquivos proposto na G4DPA atende aos objetivos de segurança elencados na seção 4.1. Para isso, suponha que o CR requereu o compartilhamento de um arquivo F, e que essa requisição foi atendida pelo CC seguindo a abordagem proposta.

A análise será dividida em duas partes, apresentadas a seguir: a primeira, tratando da segurança das requisições do CR e do CC ao GTW e das respectivas respostas; a segunda, tratando da segurança do conteúdo do arquivo que é solicitado pelo CR.

4.6.1 Análise de segurança das requisições

Esta seção apresenta uma análise de segurança das requisições que o CR e o CC enviam ao GTW na arquitetura G4DPA e das respectivas respostas, considerando os objetivos de segurança listados na seção 4.1, cuja linguagem e símbolos também serão aqui utilizados. Todas essas requisições e respostas ocorrem através do mecanismo apresentado na seção 4.5.1, a partir do qual serão delineadas as considerações a seguir.

Confidencialidade. Como visto na equação 4.8, o conteúdo da requisição (msg) é enviado criptografado com uma chave simétrica K. Essa chave segue na mesma requisição cifrada com a chave pública B. Portanto, apenas Bob é capaz de obter a chave K e decifrar corretamente o conteúdo da requisição.

Integridade. Suponha que alguma parte da mensagem M' na equação 4.7 foi alterada sem o conhecimento de Alice e de Bob. Caso a alteração tenha sido feita em $cert_A$, seria necessário alterar o valor de X pela equação 4.5. A alteração de X também seria necessária

se msg' fosse modificada, já que ela corresponde a uma mensagem cifrada com a chave K (equação 4.3). Em ambos os casos, como apenas Alice possui acesso à sua chave privada a, o atacante não conseguiria calcular corretamente a assinatura digital na equação 4.6. Portanto, não é possível realizar modificações na requisição que sejam indetectáveis para Bob e Alice e, assim, o mecanismo oferece garantia de verificação de integridade.

Autenticação mútua. Para evidenciar que o mecanismo apresentado na seção 4.5.1 proporciona autenticação mútua, deve ser mostrado que Bob sabe que foi Alice quem enviou a mensagem para ele, e que Alice sabe que apenas Bob será capaz de acessar a mensagem. Por contradição, suponha que a mensagem não tenha sido enviada a Bob por Alice. Nesse caso, seja Eva quem enviou a mensagem para Bob. Como os certificados são públicos, Eva tem acesso a $cert_A$ para construir a mensagem M' da equação 4.7, mas não tem acesso à chave privada a e, dessa forma, Eva não conseguiria produzir a assinatura da equação 4.6. Assim, a mensagem não poderia ter sido enviada a Bob por alguém que não fosse Alice. Por outro lado, novamente por contradição, suponha que Eva é capaz de abrir a mensagem enviada por Alice. Nesse caso, Eva consegue decifrar a mensagem da equação 4.3, ou seja, Eva tem acesso à chave K. Como se observa na equação 4.8, K encontra-se sempre criptografada com chave B. Todavia, como Eva não tem acesso à chave privada b, ela não seria capaz de obter a chave K. Como Bob é o único que possui conhecimento de b, também é o único que possui acesso a K e, por conseguinte, ao conteúdo da mensagem. Portanto, apenas Bob pode acessar a mensagem enviada por Alice. Assim, conclui-se que o mecanismo proposto propicia autenticação mútua entre remetente e destinatário.

Não-repúdio. Suponha que Alice enviou uma mensagem para Bob e tenta negar sua autoria. Já foi mostrado que a mensagem está íntegra. Pela equação 4.7, X está criptografado com a chave privada a de Alice, da qual apenas ela tem conhecimento. Em outras palavras, X está assinado digitalmente por Alice. Portanto, apenas Alice poderia calcular a assinatura digital $sign_a(X)$, e, assim, não há como negar sua autoria.

Resistência a ataques à segurança das requisições

Ataques *replay*. O mecanismo de troca de mensagens apresentado na seção 4.5.2 utiliza um *nonce* e um *timestamp* para evitar que a mensagem possa ser reenviada posteriormente por Eva e recebida como válida por Bob, provocando efeitos indesejados no sistema. Seja M_A uma

mensagem que Bob recebeu de Alice no *timestamp* t_B e que Eva conseguiu espionar esse envio e obter M_A. Sejam, ainda, t_A e n_A o *timestamp* e o *nonce* contidos em M_A, respectivamente. Se Bob aceitou M_A é porque $t_B - t_A \leq \tau$, sendo τ o período de tempo máximo para aceitação da requisição. Suponha que Eva reproduza o envio de M_A para Bob, que recebe o *replay* no *timestamp* t_E. Como a mensagem goza de integridade, Eva não conseguiu alterar os valores t_A e n_A em M_A. Se $t_E - t_A \leq \tau$, então n_A existe no conjunto de *nonces* que Bob já recebeu de Alice, e a mensagem seria rejeitada. Caso contrário, se $t_E - t_A > \tau$, a mensagem seria descartada por extrapolar o tempo limite para aceitação da requisição. Portanto, Eva não é capaz de realizar *replay* de mensagens enviadas de Alice para Bob.

Ataques *man-in-the-middle*. Ainda que Eva seja capaz de inspecionar e retransmitir as mensagens trocadas entre Alice e Bob, ela não será capaz de acessar o conteúdo decifrado da mensagem nem alterará a mensagem sem que isto seja detectado por Bob, devido à confidencialidade, integridade e autenticação mútua. Assim, conclui-se que o mecanismo de envio de requisições da G4DPA oferece proteção contra ataques *man-in-the-middle*. Isso é especialmente importante devido à troca de chaves públicas para estabelecimento da chave S através do algoritmo Diffie-Hellman [5], pois Eva poderia forjar uma chave pública e enviá-la para Bob como se fosse Alice e, ao mesmo tempo, forjar outra chave pública e enviá-la para Alice como se fosse Bob. Caso isso ocorresse, Eva seria capaz de estabelecer uma chave de sessão com Alice e outra chave de sessão com Bob, podendo acessar o conteúdo de todas as mensagens trocadas, além de alterá-las e retransmiti-las.

Ataques de personificação (*impersonation*). Uma vez que as mensagens trocadas entre CR e GTW e entre CC e GTW não sejam vulneráveis a ataques *replay* e a ataques *man-in-the-middle*, somente seria possível a Eva realizar um ataque de personificação caso ela tivesse acesso à chave privada de Alice ou de Bob, o que não é admissível segundo premissa elencada na seção 4.3. Portanto, o mecanismo é resistente a ataques de personificação.

4.6.2 Análise de segurança do arquivo compartilhado

Prosseguindo com a análise informal de segurança, os objetivos de segurança listados na seção 4.1 serão analisados em relação ao arquivo que é encaminhado do CC para o CR no processo de compartilhamento.

[5] "O Diffie-Hellman é suscetível a um ataque *man-in-the-middle* e, portanto, requer alguma forma de autenticação das partes comunicantes" (PAAR; PELZL, 2010).

Confidencialidade. Suponha que um bisbilhoteiro E tenha tido acesso ao conteúdo do arquivo compartilhado F. Como o arquivo é criptografado ainda no domínio do CC e descriptografado apenas no domínio do CR, infere-se que ele foi maliciosamente capturado em trânsito, ou seja, entre CC e o IPFS, entre o IPFS e o CR ou diretamente a partir do IPFS. Portanto, F se encontrava criptografado com a chave de sessão S. Assumindo que o algoritmo de criptografia simétrica utilizado é seguro, se E teve acesso ao conteúdo de F é porque E obteve S. Como S também não trafega fora dos domínios do CC e do CR, E precisaria ter acesso a pelo menos uma das chaves privadas dh_R ou dh_C para calcular S, mas essas chaves são mantidas seguras sob domínio do CR e do CC, respectivamente. Portanto, E não conseguiria obter a chave S. Ademais, como mostrado na seção 4.6.1, as requisições e respostas entre CC e CR e GTW são protegidas pelo mecanismo apresentado na seção 4.5.1 e, assim, são resistentes a ataques *man-in-the-middle*. Dessa feita, conclui-se que um bisbilhoteiro E não pode ter acesso ao conteúdo de um arquivo compartilhado F, e, assim, que o mecanismo de compartilhamento de arquivos fornece confidencialidade.

Integridade. Seja c_1 o conteúdo do arquivo F no momento em que a sua custódia foi registrada pelo CC, e c_2 o conteúdo do arquivo que o CR recebeu através do processo de compartilhamento de dados. Suponha $c_1 \neq c_2$. Nesse caso, o CR poderia detectar essa diferença comparando $\#c_1$ e $\#c_2$, sendo o primeiro conhecido por constar na resposta à busca por custódias e o segundo por poder ser calculado a partir de c_2. Assim, o CR pode verificar que o arquivo recebido corresponde ao arquivo inicialmente registrado em custódia pelo CC. Além disso, após c_1 ser cifrado pelo CC, só poderá ser decifrado com a senha compartilhada S. Portanto, o CC sabe que o CR só poderá decifrar o arquivo corretamente se seu conteúdo estiver íntegro.

Autenticação mútua. O arquivo F' encontra-se criptografado com a chave S, que é calculada usando o algoritmo Diffie-Hellman inicialmente pelo CC com as chaves DH_R e dh_C no passo P2.4 da fase 2, e posteriormente pelo CR usando as chaves DH_C e dh_R no passo P3.9 da fase 3. Como visto na seção 4.6.1, o mecanismo de troca de mensagens oferece autenticação mútua e também as protege de ataques *man-in-the-middle* e, com isso, as chaves públicas DH_R e DH_C, que são transportadas por essas mensagens, têm a garantia de serem autênticas do CR e glsCC, respectivamente. Ademais, as chaves privadas dh_R e dh_C são geradas exclusivamente para uma determinada solicitação de arquivo e presume-se que são de conhecimento exclusivo do CR e do CC, respectivamente. Sendo assim, o CR e o CC sabem que apenas o outro será capaz de calcular S. Portanto, além de o CC, apenas o CR é capaz de decifrar F'. Do mesmo modo,

somente o CC, além de o CR, possui a chave para produzir F'. Assim, remetente e destinatário de um arquivo compartilhado através da G4DPA gozam de autenticação mútua.

Não-repúdio. Suponha que o CR recebeu um arquivo que, em princípio, corresponda à custódia C, mas que o CC argumente que não foi ele quem informou essa custódia ou, ainda, que não foi ele quem adicionou o correspondente arquivo ao IPFS para o compartilhamento. Ocorre que, quando a custódia fora encaminhada ao GTW, o valor $\#F$ foi armazenado pseudo-anonimizado na BC. Assim, quando o CR calcula o *hash* do arquivo após a descriptografia e encaminha esse valor ao GTW para verificação, segundo os passos P4.1 e P4.2 da fase 4, se essa verificação for bem-sucedida, significa que o arquivo corresponde ao referenciado pela custódia $\mathscr{C}$, informada pelo CC. Ainda, como visto no item anterior, a chave S somente é de conhecimento do CC e do CR e, assim, além do CR, apenas o CC seria capaz de ter realizado essa operação de criptografia. Portanto, não há como o CC negar a custódia e o compartilhamento do arquivo F.

Resistência a ataques à segurança dos arquivos compartilhados

Ataques *man-in-the-middle*. Neste caso, o atacante E deveria se posicionar em algum ponto do caminho do arquivo entre o CC e o CR, ou seja, entre o CC e o seu nó IPFS, ou entre os nós IPFS, ou, por fim, entre o CR e o seu nó IPFS. Como os arquivos são cifrados e decifrados nas pontas, internamente aos domínios do CC e do CR, seria necessário comprometer a chave criptográfica aleatória S. Como visto na seção 4.6.1, as requisições e respostas entre CC, CR e GTW são protegidas pelo mecanismo da seção 4.5.1, permanecendo imunes a ataques *man-in-the-middle* e, por conseguinte, assegurando ao Diffie-Hellman proteção contra esse vetor. Portanto, como a chave S não trafega às claras, sendo calculada apenas nas pontas, mesmo que E obtenha o arquivo, não consegue ter conhecimento de seu conteúdo. Ainda, caso E forjasse e conseguisse transmitir outro arquivo ao destino, a requisição do CR ao IPFS levaria a erro, pois, como visto na seção 3.2, o CID é criado com base no conteúdo do arquivo F'. Ademais, o conteúdo do arquivo decifrado F também é validado através dos passos P4.1 e P4.2 da fase 4. Assim, a transmissão do arquivo está protegida de ataques *man-in-the-middle*.

4.6.3 Outras propriedades de segurança da G4DPA

Rastreabilidade. Como visto na seção 3.1, uma *blockchain* é um mecanismo que promove a imutabilidade de seus registros, que é uma característica muito desejável para a realização de auditorias. Todas as operações promovidas pela G4DPA são realizadas de modo a armazenar o solicitante na BC, e os registros nunca são removidos, mas apenas mudam para um estado inativo. Ainda que fossem removidos, estariam disponíveis para verificação em um estado anterior da BC.

Anonimização. Todos os dados enviados para a BC pelo GTW são pseudoanonimizados, dificultando a inferência de novas informações através do conhecimento das relações entre CCs, CRs e TDs, em atendimento à previsão do considerando 26 da GDPR e do artigo 13 da LGPD. Além disso, o GTW responde às requisições do CC e do CR com recibos criptografados, de modo que, sem que sejam decifrados, não permitem que os dados relacionados armazenados na BC sejam identificados, como as custódias – também cifradas –, que identificariam tanto o TD quanto o CC que as tutelam.

5 EXPERIMENTOS EM UM CENÁRIO DE *E-HEALTHCARE*

Este capítulo descreve os experimentos conduzidos com o objetivo de avaliar a implementação da G4DPA em um cenário hipotético. Esse cenário é brevemente descrito na seção 5.1 e, a seguir, a seção 5.2 apresenta uma visão geral dos experimentos sobre ele realizados. A seção 5.3 detalha as tecnologias utilizadas e as principais decisões tomadas na implementação. Por fim, as seções 5.4, 5.5, 5.6 e 5.7 descrevem em mais detalhes os objetivos e a metodologia aplicada em cada experimento, bem como os resultados e as respectivas análises.

5.1 Descrição do cenário

O cenário adotado consiste no compartilhamento de arquivos contendo dados de saúde de pacientes entre instituições de atendimento médico. Apesar de hipotética, trata-se de uma situação plausível e realista.

Instituições que utilizam o protocolo *Fast Healthcare Interoperability Resources* (FHIR) [1] para gerenciar dados clínicos podem representar e armazenar exames de imagem por meio de recursos como ImagingStudy [2], DocumentReference [3], Binary [4] e ImagingSelection [5]. Esses recursos permitem que os dados sejam incorporados diretamente por meio de codificação em base64 ou armazenados em serviços externos, referenciados por elementos do tipo Endpoint [6], como serviços RESTful DICOMweb [7], incluindo *Web Access to DICOM Objects* (WADO-RS)[8].

Em situações de emergência médica, costuma ser muito importante que o profissional de saúde tenha acesso a exames prévios do paciente. Como apresentado na seção 3.3, a GDPR e a LGPD permitem o acesso a esses dados para proteção de interesses vitais, independentemente do consentimento do titular. Contudo, esse acesso depende de o paciente portar os exames em mídia digital ou saber onde foram realizados, desde que haja mecanismos que possibilitem busca e acesso remoto entre instituições. A G4DPA busca proporcionar um desses mecanismos ao registrar exames realizados em diferentes instituições, oferecendo funcionalidades auditáveis de busca e compartilhamento em conformidade com os normativos legais.

[1] https://hl7.org/fhir/
[2] https://www.hl7.org/fhir/imagingstudy.html
[3] https://www.hl7.org/fhir/documentreference.html
[4] https://www.hl7.org/fhir/binary.html
[5] https://www.hl7.org/fhir/imagingselection.html
[6] https://www.hl7.org/fhir/endpoint.html
[7] https://www.dicomstandard.org/using/dicomweb
[8] https://www.dicomstandard.org/using/dicomweb/retrieve-wado-rs-and-wado-uri

5.2 Visão geral dos experimentos

A avaliação da G4DPA foi organizada em quatro experimentos, cada um voltado à análise de um aspecto crítico da solução. Assim, o primeiro experimento investigou o consumo de gas nas execuções do contrato inteligente, permitindo identificar funções mais custosas e possíveis variações de custo. O segundo analisou o tempo necessário para realizar operações com arquivos no IPFS, considerando inclusão, recuperação e remoção de arquivos com tamanhos distintos e diferentes níveis de concorrência, de modo a avaliar a influência desses fatores no desempenho. O terceiro mediu o tempo de busca de custódias registradas na BC sob perspectivas com diferentes números de controladores concorrentes, fornecendo indicadores de escalabilidade. Por fim, o quarto experimento examinou o desempenho do compartilhamento de arquivos entre controladores, que é a funcionalidade finalística da arquitetura, que envolve operações criptográficas e transmissão de dados pelo IPFS. Os resultados obtidos nesses experimentos, apresentados e analisados nas seções seguintes, permitem avaliar a eficiência e a viabilidade da abordagem proposta frente a diferentes condições de carga e operação.

Com o objetivo de garantir a robustez dos resultados, cada experimento foi executado 100 vezes de forma independente. Esse número de repetições é superior ao mínimo de 30 comumente recomendado pela literatura (JAIN, 1991; LAW; KELTON, 2000), que se baseia no teorema central do limite. Este teorema assegura que, ao se coletar um número suficientemente grande de amostras independentes de uma mesma população, a distribuição da média amostral tende a uma distribuição normal, característica fundamental para a validade dos intervalos de confiança. Ao adotar 100 repetições, não apenas superamos essa recomendação, como também contribuímos para a estabilidade das estimativas e para a obtenção de intervalos de confiança de 99% mais estreitos, reforçando a confiabilidade das conclusões em cada cenário experimental.

Os experimentos foram conduzidos em um computador pessoal com os seguintes recursos de hardware: um processador AMD Ryzen 5 1600, contando com 6 núcleos e 12 *threads* de execução, lançado em 2017; 16 GB de memória RAM DDR4 2400MHz; e uma unidade SSD de 120 GB. Todo o código escrito para a prova de conceito está disponível no repositório Git do projeto (CASTRO, 2024).

5.3 Descrição dos componentes da implementação

Os componentes de infraestrutura básica da implementação foram executados em contêineres Docker através do orquestrador Docker Compose. A tabela 20 mostra a relação do *software* utilizado, as respectivas versões e o papel desempenhado.

Tabela 20 – *Software* utilizado para implementação da infraestrutura

Software	**Versão**	**Papel**
Docker	27.1.1	Motor de contêineres
Docker Compose	2.29.1	Orquestrador de contêineres
Kubo	0.29.0	Implementação do protocolo IPFS
Geth	alltools-v1.14.3	Implementação do protocolo da *blockchain* Ethereum
InfluxDB	2.7	Banco de dados baseado em séries temporais para monitoramento do desempenho do Geth
Prometheus	v2.52.0	Banco de dados para monitoramento e emissão de alertas acerca do desempenho do Geth
Grafana	10.1.10	Apresentação de *dashboards* com gráficos e visualizações de dados do InfluxDB e do Prometheus

Fonte: elaborada pelo autor.

Implementação do IPFS. O Kubo [9] foi configurado para prover o serviço IPFS através de uma rede privada. Nesse modo de funcionamento, é necessário definir um nó responsável pelo procedimento de *bootstrap*, a partir do qual os outros nós tomam conhecimento dos outros *peers* em operação na rede, e uma chave, sem a qual não é possível fazer parte dessa rede. Os nós também foram configurados com o perfil `flatfs` [10], que corresponde ao modo de armazenamento de dados no IPFS mais confiável e testado atualmente, além de minimizar o uso de memória e proporcionar a execução do mecanismo de coleta de lixo o mais rápido possível — o que é importante nesta arquitetura —, apesar de não ser o modo que proporciona a importação de dados mais rápida.

Implementação da *blockchain*. A Ethereum foi escolhida para a implementação tanto por ter sido a *blockchain* mais utilizada nos trabalhos relacionados, conforme constatado na tabela 4, como por ser bastante versátil, podendo ser utilizada tanto uma rede pública existente, como a Mainnet, ou ser configurada como uma rede privada. Considerando os altos custos de execução da rede Ethereum Mainnet e também a dificuldade de obter o volume de *tokens* necessários para o desenvolvimento desta implementação utilizando uma rede pública de testes, como a

[9] https://github.com/ipfs/kubo
[10] https://docs.ipfs.tech/how-to/default-profile/#available-profiles

Sepolia [11], o Geth [12] foi configurado para funcionar localmente simulando uma cadeia para desenvolvimento, através da opção `--dev`. Para tentar se aproximar das condições da rede Mainnet, a produção de blocos foi definida para ocorrer a cada 12 segundos. O bloco gênesis foi configurado com todas as atualizações até o *fork* Cancun. O limite de gas por bloco foi definido como `0x1c9c380` (ou 30.000.000) de unidades. Por fim, foram criadas algumas contas com 100 ETH cada para a realização dos experimentos.

Monitoramento da *blockchain*. Para monitorar o comportamento da rede Ethereum durante o desenvolvimento e os testes da implementação, foram coletados dados através de duas formas:

1. Envio de informações do Geth para um serviço InfluxDB [13];
2. Coleta de métricas pelo Prometheus [14] por um *endpoint* configurado no próprio Geth.

A partir disso, *dashboards* foram configurados no Grafana [15] utilizando essas duas fontes de dados, possibilitando a visualização do desempenho da *blockchain* de forma simplificada.

Infraestrutura de chaves públicas. A criação de uma AC e dos certificados digitais para entidades da arquitetura, conforme os requisitos indicados na seção 4.3, foi realizada através do OpenSSL [16]. Uma autoridade certificadora raiz foi criada e, a partir desta, emitido um certificado para uma autoridade certificadora intermediária. Por sua vez, a partir da autoridade intermediária, foram emitidos um certificado para o GTW e vários certificados para CDs e TDs.

Codificação dos componentes. Todo o código escrito para configuração da infraestrutura está disponível no repositório do projeto (CASTRO, 2024). O GTW e o código para simular o comportamento dos CDs foram escritos na linguagem de programação Go, versão 1.22.5, e as principais bibliotecas utilizadas estão listadas na tabela 21.

Serialização dos dados. Os dados das requisições feitas pelos CDs ao GTW e das respectivas respostas, como visto na seção 4.5.1, foram serializados utilizando a versão 3 do Protocol Buffer [17], que é um formato para intercâmbio de dados estruturados proposto pelo Google. Foram definidas as possíveis mensagens que seriam utilizadas em cada uma das fases do mecanismo de compartilhamento de arquivos, como visto na seção 4.5.2. Para gerar o código Go para serializar e desserializar as mensagens, foi utilizada a ferramenta `protoc` na versão `libprotoc 27.0`.

[11] https://ethereum.org/pt/developers/docs/networks/
[12] https://geth.ethereum.org/
[13] https://github.com/influxdata/influxdb
[14] https://prometheus.io/
[15] https://grafana.com/oss/grafana/
[16] https://openssl-library.org/
[17] https://protobuf.dev/programming-guides/proto3/

Tabela 21 – Principais bibliotecas utilizadas

Biblioteca	Versão	Finalidade
`crypto/aes`	Biblioteca padrão	Criptografia simétrica com AES
`crypto/cipher`	Biblioteca padrão	Modos padrão de criptografia em bloco, usado para o modo GCM
`crypto/ecdh`	Biblioteca padrão	Algoritmo Diffie-Hellman com curvas elípticas
`crypto/rand`	Biblioteca padrão	Gerador criptograficamente seguro de números aleatórios
`crypto/rsa`	Biblioteca padrão	Algoritmo de chave pública RSA
`crypto/sha256`	Biblioteca padrão	Algoritmo de *hash* SHA-256
`crypto/x509`	Biblioteca padrão	Subconjunto de definições do padrão x.509
`github.com/ethereum/go-ethereum`	v1.14.3	Interação com contratos inteligentes em redes Ethereum
`github.com/gin-gonic/gin`	v1.10.0	*Framework* web para a linguagem Go
`github.com/ipfs/boxo`	v0.19.0	Interação com serviços IPFS
`github.com/ipfs/go-cid`	v0.19.0	Implementação da especificação CID
`github.com/ipfs/kubo`	v0.28.0	Interação com serviços IPFS
`gitlab.com/ubiqsecurity/ubiq-fpe-go`	v0.0.0-20240501180808	Algoritmo de criptografia simétrica FPE-FF1, para anonimização
`google.golang.org/protobuf`	v1.34.1	Mecanismo de serialização por Protocol Buffers

Fonte: elaborada pelo autor.

Criptografia de dados em trânsito. Uma vez serializados, os dados das requisições e respostas são criptografados com o algoritmo AES-256 (NIST, 2023) no modo GCM (DWORKIN, 2007) para cifra em bloco, correspondendo à operação de criptografia simétrica indicada na equação 4.3. Os *hashes* da equação 4.5 foram calculados através do algoritmo SHA-256 (NIST, 2015), e a assinatura digital realizada na equação 4.6 foi obtida empregando o algoritmo RSA-PSS (RIVEST *et al.*, 1978; BELLARE; ROGAWAY, 1996), com esquema de preenchimento OAEP (BELLARE; ROGAWAY, 1995), conforme (MORIARTY *et al.*, 2016). Nos códigos a seguir, `Message` é a estrutura que representa os dados da requisição ainda em texto claro. Uma vez carregados os dados nessa estrutura, serão empregadas as técnicas criptográficas mencionadas, gerando uma estrutura do tipo `DPMessage`.

```
type Message struct {
    Timestamp   uint64
    Certificate x509.Certificate
    Data        []byte
}

type DPMessage struct {
        Timestamp   uint64
```

```
        Certificate []byte
        Secret []byte
        SymmetricKeyDigest []byte
        EncryptedData []byte
        Signature []byte
}
```

Criptografia de dados em repouso. Quanto aos arquivos transmitidos via IPFS (armazenados ao menos temporariamente, portanto), eles são criptografados com o algoritmo AES-256 no modo GCM através de uma chave compartilhada secreta. Essa chave, por sua vez, é gerada por um protocolo via algoritmo Diffie-Hellman (DIFFIE; HELLMAN, 1976), com utilização da curva elíptica Curve25519 (LANGLEY *et al.*, 2016).

Pseudoanonimização de dados pessoais. Por fim, uma das técnicas sugeridas para pseudoanonimização pela European Union Agency for Cybersecurity (ENISA) é o emprego de um algoritmo do tipo *Format Preserving Encryption* (FPE) (ENISA, 2019). Dessa feita, na implementação realizada, os dados são anonimizados pelo GTW usando o algoritmo FPE-FF1, conforme a atual recomendação do National Institute of Standards and Technology (NIST) (DWORKIN, 2019).

Contrato inteligente. O contrato inteligente foi escrito em Solidity, versão 0.8.26 e compilado com `solc` na mesma versão. A ferramenta `abigen`, versão 1.14.2, foi usada para gerar o código *Application Binary Interface* (ABI) e o código Go para interagir com as funções do contrato inteligente. Abaixo, segue uma versão resumida do contrato inteligente G4DPA-SC, omitindo parâmetros, enumerações e conteúdo das funções, deixando apenas o seu esqueleto de forma a oferecer uma visão resumida. A versão completa pode ser examinada no apêndice A. A título de informação, o *deploy* desse contrato na *blockchain*, com as configurações apresentadas anteriormente, consumiu 3.654.370 unidades de gas.

```
struct Custody {
    uint64 startTs;
    uint64 endTs;
    bytes8 internalId;
    bytes8 creationTs;
    bytes8 keyword;
    bytes16 dataControllerId;
```

```
    bytes16 dataOwnerId;
    bytes32 dataHash;
    LawfulBase lawfulBase;
    DataKind dataKind;
    ProcessingMode processingMode;
    Purpose purpose;
}

struct Consent {
    uint64 idxCustody;
    uint64 fromTs;
    uint64 untilTs;
    bytes16 dataOwnerId;
    bytes16 dataCustodianId;
    bytes16 dataRequesterId;
    uint64 withdrawnSince;
}

struct Request {
    uint64 startTs;
    uint64 endTs;
    bool deniedByCustodian;
    uint64 custodyIdx;
    bytes16 dataRequesterId;
    LawfulBase lawfulBase;
    DataKind dataKind;
    Purpose purpose;
}

event LogNotifySharingRequest(...);
event LogNotifyDataSent(...);
event LogNotifyDeleteDataFromIPFS(...);
```

```
contract G4DP {
    Custody[] public custodies;
    mapping(bytes16 => uint64[]) public dataControllerCustodies;
    mapping(bytes16 => uint64[]) public dataOwnerCustodies;

    Request[] public requests;
    mapping(bytes16 => uint64[]) public dataRequesterRequests;
    mapping(bytes16 => uint64[]) public dataControllerRequests;
    mapping(bytes16 => uint64[]) public dataOwnerRequests;

    mapping(Group => mapping(bytes16 => Member[])) public memberOf;

    function addCustody(...) public returns (uint64) {}
    function registerSearch(...) public {...}
    function searchCustody(...) public view returns (...) {...}
    function addRequestSharing(...) public returns (...) {...}
    function notifySharingRequest(...) public {...}
    function notifyDataSent(...) public {...}
    function notifyDataDecrypted(...) public {...}
    function endRequestSharing(...) public returns (bool) {}
}
```

5.4 Experimento 1: consumo de gas nas execuções do contrato inteligente

Ao apresentar soluções que empregam contratos inteligentes em *blockchains* Ethereum e que foram implementadas para avaliação, a literatura costuma analisar os custos de execução de suas funções em termos de consumo de gas (DWIVEDI *et al.*, 2021; ULLAH *et al.*, 2022; JABARULLA; LEE, 2021; LI *et al.*, 2021; GHANI *et al.*, 2020; ZHANG; ZHAO, 2023; KUMAR *et al.*, 2022; KAUR *et al.*, 2024; SHREE *et al.*, 2024; RAJ; GHOSH, 2024). Essa métrica está diretamente relacionada aos custos financeiros de execução e também se relaciona com o *throughput* da solução, pois uma vez atingido o limite máximo de gas de um bloco,

a execução da função somente poderá ocorrer a partir da mineração do próximo bloco. As principais funções do contrato inteligente `G4DPA-SC` são apresentadas a seguir, acompanhadas de uma breve descrição de suas respectivas funcionalidades.

AddCustody: função utilizada por uma instituição de saúde para registrar, na BC, a existência de um arquivo contendo dados pessoais sob sua custódia. Nesse registro, são especificados a base legal aplicável, o tipo de dado pessoal, a forma de armazenamento e o propósito de sua manutenção.

CloseCustody: função invocada por um CC para registrar, na BC, que deixou, a partir de um instante específico, de manter sob sua custódia um certo arquivo.

RegisterSearch: função mediante a qual um CR efetua a busca por arquivos contendo dados pessoais de determinado tipo, associados a um TD, devendo registrar a base legal que autoriza a busca e o acesso, bem como o propósito de utilização.

AddRequestSharing: função mediante a qual um CR inicia a solicitação de compartilhamento de um arquivo cuja existência foi previamente identificada, por meio da função `RegisterSearch`, sob a custódia de um determinado CC.

NotifySharingRequest: função utilizada pelo CR para registrar, na BC, a emissão de um evento $\mathscr{E}_1$ direcionado a um CC, indicando a existência de uma requisição de compartilhamento de arquivo pendente.

EndRequestSharing: função que registra, na BC, a conclusão bem sucedida do processo de compartilhamento de um arquivo contendo dados pessoais de um TD, requisitado por um CR a um CC.

NotifyDataSent: função utilizada pelo CC para emitir, na BC, um evento $\mathscr{E}_2$ direcionado a um CR, informando que o arquivo solicitado encontra-se disponível por meio do IPFS.

NotifyDataDecrypted: função utilizada pelo CR para notificar, através de um evento $\mathscr{E}_3$ na BC, que o arquivo foi devidamente recebido e decriptografado. A partir dessa notificação, o CC deve proceder à exclusão dos dados enviados do IPFS.

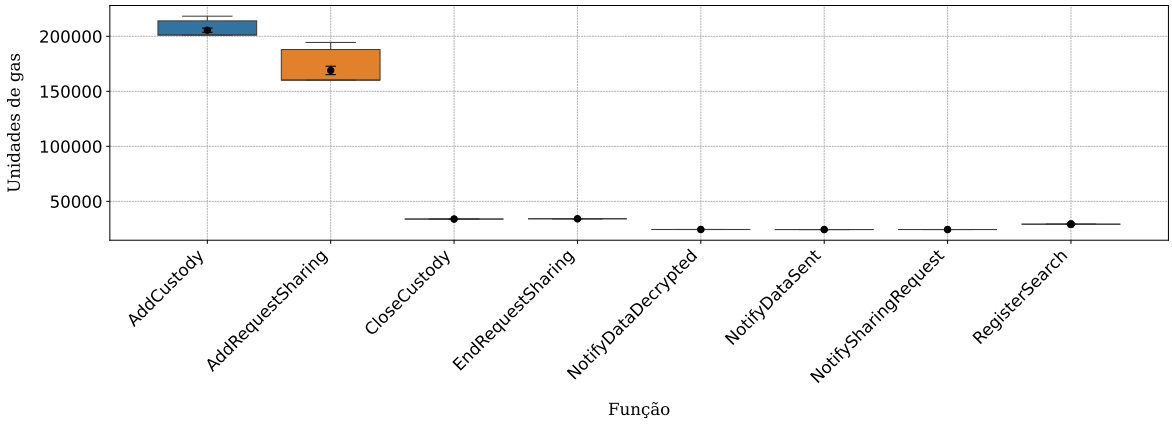
Este experimento consistiu em executar cada uma dessas funções e registrar o correspondente consumo total de gas. Isso foi repetido 100 vezes, conforme motivação feita na seção 5.2, e os resultados numéricos obtidos constam na tabela 22. Observa-se que as funções `AddCustody` e `AddRequestSharing` foram as mais custosas e as que apresentaram uma variação mais relevante de custo. As demais funções apresentaram pouquíssima variação no consumo de gas. Os dados da tabela estão resumidos no gráfico da figura 15.

Tabela 22 – Resultados numéricos do experimento 1

Função	Consumo mínimo de gas	Consumo máximo de gas	Média	Mediana	Desvio padrão	Amplitude do IC (99%)	IC inferior	IC superior
`AddCustody`	201.150	218.274	205.547	201.174	7443	3835	203.630	207.464
`AddRequestSharing`	160.205	194.405	168.988	160.205	14.915	7684	165.146	172.830
`CloseCustody`	33.928	33.940	33.940	33.940	1	1	33.940	33.940
`EndRequestSharing`	34.172	34.184	34.184	34.184	1	1	34.184	34.184
`NotifyDataDecrypted`	24.427	24.439	24.439	24.439	1	1	24.439	24.439
`NotifyDataSent`	24.385	24.397	24.397	24.397	1	1	24.397	24.397
`NotifySharingRequest`	24.406	24.418	24.418	24.418	1	1	24.418	24.418
`RegisterSearch`	29.235	29.259	29.257	29.259	5	3	29.256	29.258

Fonte: elaborada pelo autor.

Figura 15 – *Box-plot* do consumo de gas pelas funções do contrato inteligente



Fonte: elaborada pelo autor.

A maior variação observada nas funções `AddCustody` e `AddRequestSharing` deve-se ao fato de que, na implementação realizada, essas funções operam sobre algumas estruturas: um *array* dinâmico onde os dados são de fato armazenados, e mapas dos identificadores dos CDs e TDs para *arrays* que contêm as posições do *array* de armazenamento. Essa separação é feita por dois motivos:

1. Em Solidity, os *arrays* são estruturas sequenciais indexadas, que permitem acesso direto a qualquer elemento por meio de seu índice de forma eficiente (*O(1)*). No entanto, operações de busca por um determinado valor requerem varredura linear (*O(n)*), a menos que o conteúdo esteja ordenado e seja implementado manualmente um algoritmo de busca mais eficiente. Todavia, realizar a ordenação dinamicamente no estado da *blockchain* utilizaria muitas operações de escrita para movimentar os dados, podendo encarecer sobremaneira a execução da função;
2. Os mapas são estruturas de acesso rápido e excelentes para busca, mas que não permitem enumerar seu conteúdo, ao menos até a versão atual da linguagem Solidity.

Por sua vez, as demais funções ou operam realizando atualizações de variáveis de

estado que podem ser localizadas na *blockchain* de modo simples, usando as estruturas mantidas pelas duas funções anteriores, ou emitindo eventos indexáveis na *blockchain*, para notificar CDs.

Com base nas estatísticas obtidas, é possível estimar que, repetindo as condições do experimento, há uma chance de 99% de o consumo de gas das funções estar entre os valores IC mínimo e máximo. Por outra perspectiva, considerando a configuração da *blockchain* indicada na seção 5.3, i.e. um limite para o consumo de gas por bloco de 30.000.000 e um tempo de produção de bloco de 12 segundos, cada função pode alcançar os valores de *throughput* máximos teóricos indicados na tabela 23, supondo que todas as execuções alcancem os valores máximos e que o bloco seja composto apenas por execução dessas funções.

Tabela 23 – *Throughput* máximo teórico das funções do contrato inteligente G4DPA-SC

Função	**Consumo de gas máximo estimado**	***Throughput* máximo teórico (TPS)**
`AddCustody`	207.464	12
`AddRequestSharing`	172.830	14
`CloseCustody`	33.940	74
`EndRequestSharing`	34.184	73
`NotifyDataDecrypted`	24.439	102
`NotifyDataSent`	24.397	102
`NotifySharingRequest`	24.418	102
NotifySharingRequestRegisterSearch	29.258	85

Fonte: elaborada pelo autor.

5.5 Experimento 2: tempo para desempenhar operações de leitura e escrita no IPFS

Os trabalhos relacionados que fizeram uso de IPFS e implementaram provas de conceito analisaram, com frequência, o desempenho de operações de leitura e gravação de arquivos (JABARULLA; LEE, 2021; YEH *et al.*, 2022; KUMAR *et al.*, 2022; SHREE *et al.*, 2024; YANG *et al.*, 2022; YEH *et al.*, 2023). Desse modo, o segundo experimento tem por objetivo analisar os tempos das operações de gravação, leitura e remoção de arquivos no contexto abordado neste trabalho, ou seja, utilizando o IPFS como meio para realizar a transmissão de arquivos de um CC para um CR, excluindo os arquivos logo após sua conclusão.

Embora a figura 6 apresente três nós IPFS, este experimento requer apenas dois: os nós do CC e do CR, responsáveis pela execução das operações de gravação, leitura e remoção mencionadas anteriormente. O desempenho dessas operações foi avaliado em relação a três variáveis:

1. Operação realizada, podendo ser:

a) `add`: adição (gravação) de um novo arquivo ao nó do CC;

b) `delete`: remoção dos arquivos do nó do CC;

c) `retrieve`: obtenção (leitura) de um novo arquivo através do nó do CR.

2. Tamanho do arquivo, assumindo valores de 1 MB, 10 MB e 100 MB;
3. Quantidade de operações conduzidas concorrentemente, podendo ser 1, 5, 25 e 125.

A escolha dos tamanhos de arquivo fundamenta-se em dois aspectos principais. Primeiro, tais dimensões são representativas de arquivos comumente utilizados em aplicações de *e-healthcare*, como relatórios clínicos, exames digitalizados ou mesmo conjuntos de registros de um TC ou RM. Segundo, a variação em diferentes ordens de magnitude possibilita avaliar o impacto do tamanho do arquivo sobre o desempenho das operações, possibilitando que os resultados sejam observados tanto em cenários típicos quanto em limites superiores mais exigentes.

Já a quantidade de operações concorrentes foi definida de forma multiplicativa, a fim de representar cenários de carga progressivamente maior, com o intuito de analisar a escalabilidade do sistema em diferentes níveis de concorrência, com um conjunto reduzido, porém representativo, de pontos experimentais.

Dessa maneira, considerando as possibilidades para as três variáveis, cada uma das 36 combinações possíveis foi conduzida em 100 rodadas separadas, conforme explicação na seção 5.2, obtendo o tempo médio de cada operação para cada tamanho de arquivo e quantidade de operações concorrentes por rodada. Para a realização desses testes, foram gerados arquivos binários com conteúdo aleatório, considerando-se que o conteúdo em si não exerce influência significativa sobre a execução das operações, desde que mantida a aleatoriedade. Conforme descrito na seção 3.2, o IPFS divide cada arquivo em blocos (*chunks*) de 256 kB, os quais podem eventualmente se repetir entre diferentes arquivos. Assim, é importante que tais repetições ocorram de forma igualmente aleatória. Além disso, a utilização de um mesmo arquivo em todos os testes poderia levar o IPFS a não realizar a transmissão efetiva do CC para o CR, uma vez que este último já disporia localmente dos blocos correspondentes durante a recuperação.

A tabela 24 reúne os resultados do experimento 2 para as operações `add`, `delete` e `retrieve`, incluindo métricas de tempo de execução, dispersão e intervalos de confiança sob diferentes tamanhos de arquivos e níveis de concorrência.

Tabela 24 – Resultados numéricos do experimento 2

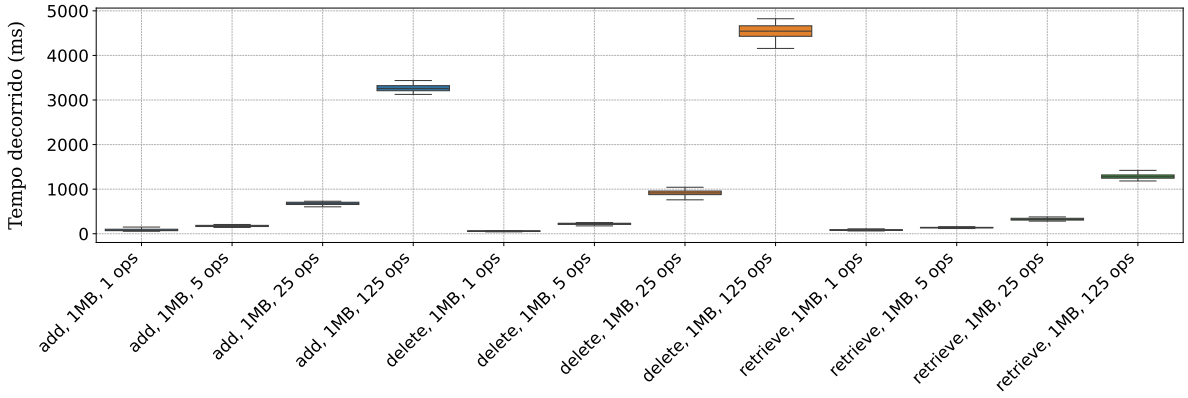
Operação	Tamanho do arquivo (MB)	Quantidade de operações concorrentes	Tempo mínimo (ms)	Tempo máximo (ms)	Média (ms)	Mediana (ms)	Desvio padrão (ms)	Amplitude do IC (99%)	IC inferior	IC superior
add	1	1	56	234	92	74	39	20	82	102
	1	5	141	430	190	169	54	13	183	196
	1	25	604	1082	695	685	68	9	691	700
	1	125	3125	3803	3277	3256	109	17	3268	3285
	10	1	151	442	196	175	57	29	181	211
	10	5	476	825	546	542	41	10	541	551
	10	25	2399	3169	2642	2627	153	17	2634	2650
	10	125	12.097	66.898	13.673	12.691	6839	325	13.511	13.836
	100	1	1050	1649	1161	1141	84	43	1139	1183
	100	5	4028	29.066	4673	4416	2475	568	4389	4957
	100	25	20.785	68.508	25.200	23.087	8443	866	24.767	25.633
delete	1	1	29	423	80	60	57	29	65	94
	1	5	152	432	237	224	56	20	227	247
	1	25	657	1123	906	916	79	33	890	923
	1	125	3804	8089	4563	4544	406	77	4524	4601
	10	1	46	457	102	78	71	37	83	120
	10	5	163	1647	262	246	145	36	244	280
	10	25	835	1215	1064	1072	76	34	1047	1081
	10	125	4820	5709	5302	5312	177	76	5264	5340
	100	1	81	244	140	138	20	10	135	145
	100	5	385	661	493	487	51	17	485	502
	100	25	1973	4076	2389	2357	241	43	2367	2410
retrieve	1	1	63	384	94	80	45	23	82	106
	1	5	117	313	144	136	28	8	140	148
	1	25	283	705	334	321	52	8	330	338
	1	125	1184	6212	1368	1279	503	26	1355	1380
	10	1	484	710	545	541	38	20	535	555
	10	5	993	1418	1056	1045	61	18	1047	1065
	10	25	2564	3130	2675	2649	99	16	2668	2683
	10	125	11.723	51.637	13.659	12.829	5343	268	13.525	13.792
	100	1	4285	4913	4495	4492	96	50	4470	4520
	100	5	9592	27.706	10.226	9894	2198	505	9973	10.478
	100	25	25.334	59.031	28.223	27.624	4457	458	27.994	28.452

Fonte: elaborada pelo autor.

Com o intuito de ampliar a compreensão dos resultados apresentados na tabela 24, apresentam-se a seguir *box-plots* que ilustram a distribuição dos tempos de execução. As figuras estão organizadas de acordo com três enfoques: variação do tamanho do arquivo, quantidade de operações concorrentes e tipo de operação realizada.

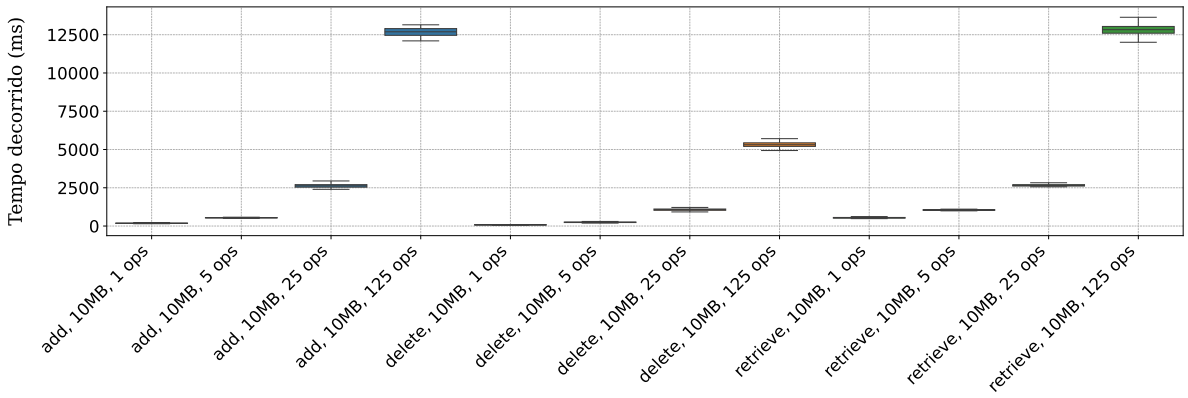
Dessa forma, as figuras 16, 17 e 18 apresentam a distribuição dos tempos de execução das operações de `add`, `delete` e `retrieve` em função do tamanho do arquivo, avaliando o impacto de arquivos de 1 MB, 10 MB e 100 MB, respectivamente, no desempenho das funções, considerando diferentes níveis de concorrência.

Figura 16 – *Box-plot* da duração das funções do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes, para tamanhos de arquivo de 1 MB



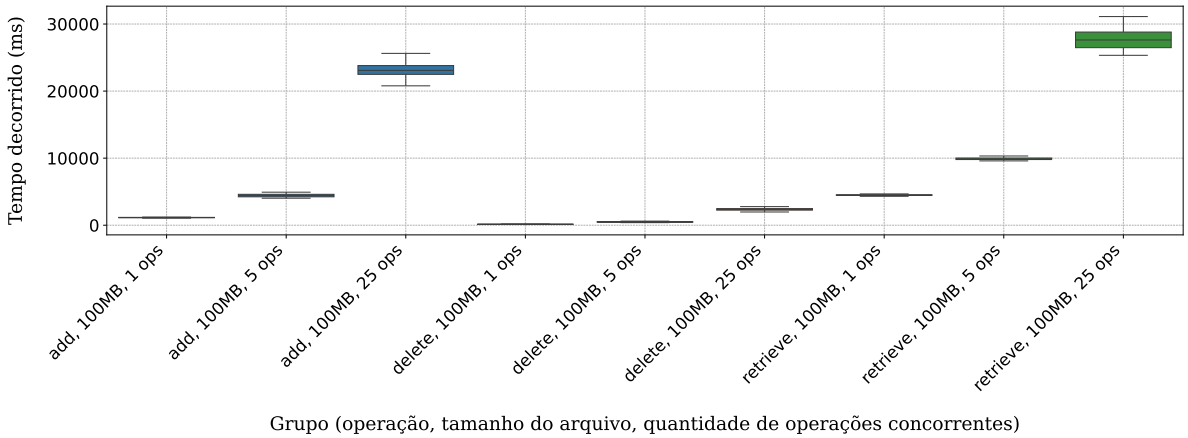
Fonte: elaborada pelo autor.

Figura 17 – *Box-plot* da duração das funções do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes, para tamanhos de arquivo de 10 MB



Fonte: elaborada pelo autor.

Figura 18 – *Box-plot* da duração das funções do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes, para tamanhos de arquivo de 100 MB

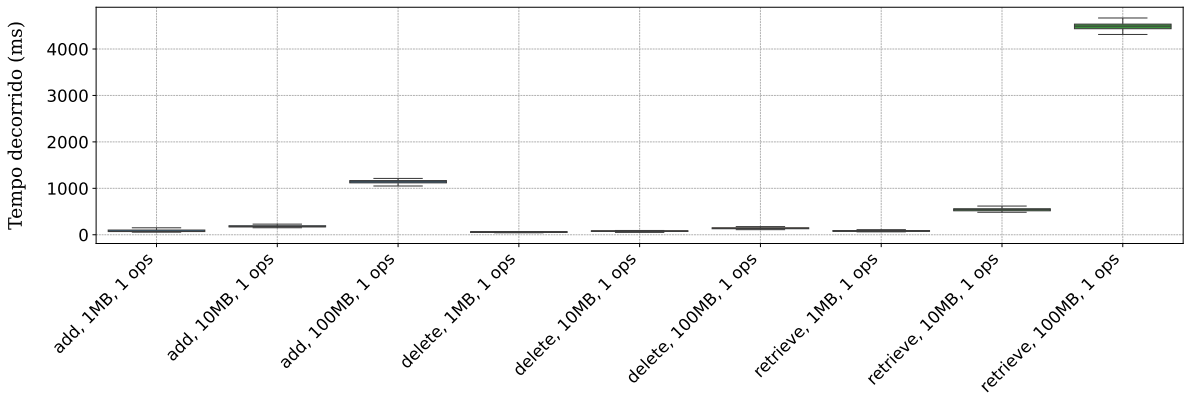


Fonte: elaborada pelo autor.

A análise dessas figuras evidencia que o aumento do tamanho dos arquivos resulta em crescimento significativo dos tempos médios e medianos de execução. Esse efeito é mais pronunciado nas operações `add` e `retrieve`, que chegam a apresentar valores bastante elevados em cenários com arquivos de 100 MB. `Delete`, em contrapartida, manteve-se como a operação mais eficiente e estável, exibindo tempos consistentemente menores.

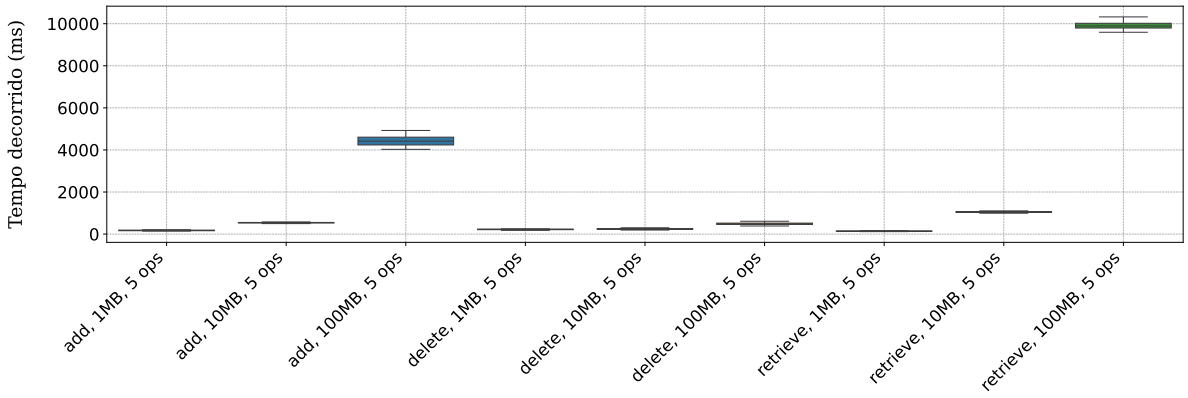
Por sua vez, as figuras 19, 20, 21 e 22 destacam a influência do número de operações concorrentes sobre os tempos de execução, visando observar a forma como a concorrência afeta a variabilidade e os valores centrais das distribuições para cada função analisada.

Figura 19 – *Box-plot* da duração das funções do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes, considerando 1 operação concorrente



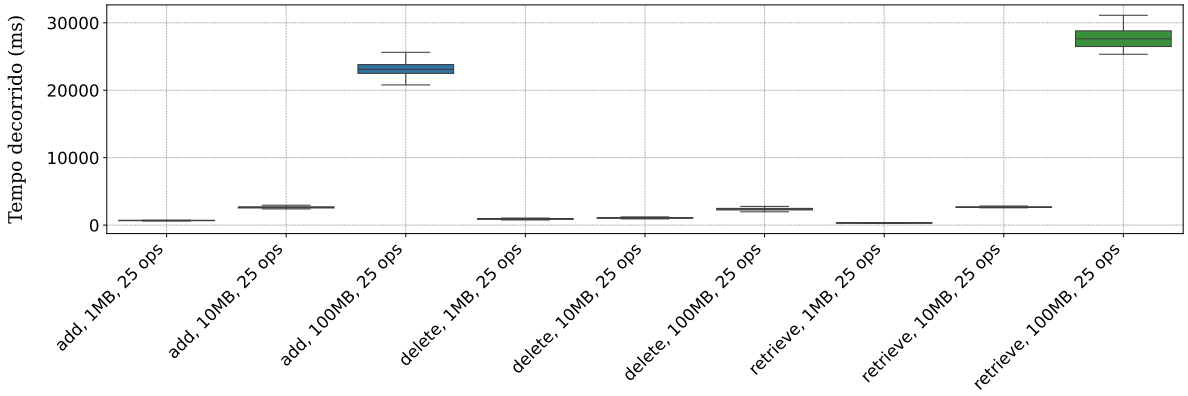
Fonte: elaborada pelo autor.

Figura 20 – *Box-plot* da duração das funções do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes, considerando 5 operações concorrentes



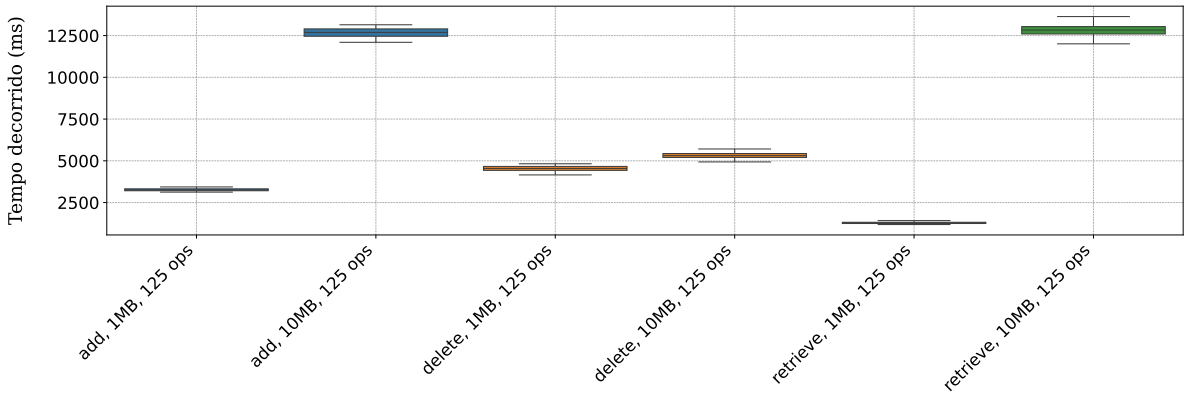
Fonte: elaborada pelo autor.

Figura 21 – *Box-plot* da duração das funções do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes, considerando 25 operações concorrentes



Fonte: elaborada pelo autor.

Figura 22 – *Box-plot* da duração das funções do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes, considerando 125 operações concorrentes

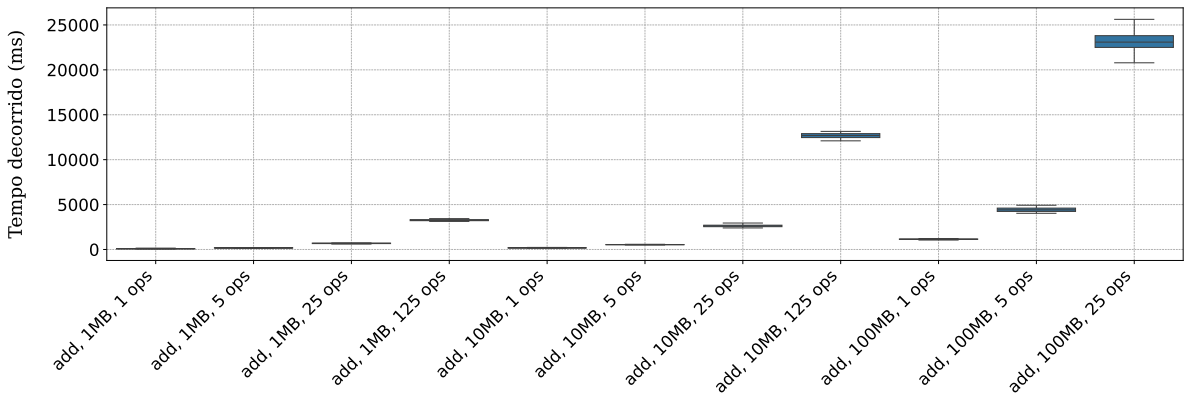


Fonte: elaborada pelo autor.

Como pode ser observado nessas figuras, o crescimento da concorrência eleva substancialmente os tempos de execução e amplia a variabilidade dos resultados. Esse comportamento é mais severo para as operações de `add` e `retrieve`, que apresentam dispersões elevadas em cenários com 25 ou 125 operações concorrentes. `Delete` novamente manteve desempenho mais previsível, mesmo sob maior carga, reafirmando seu caráter mais estável em comparação às demais operações.

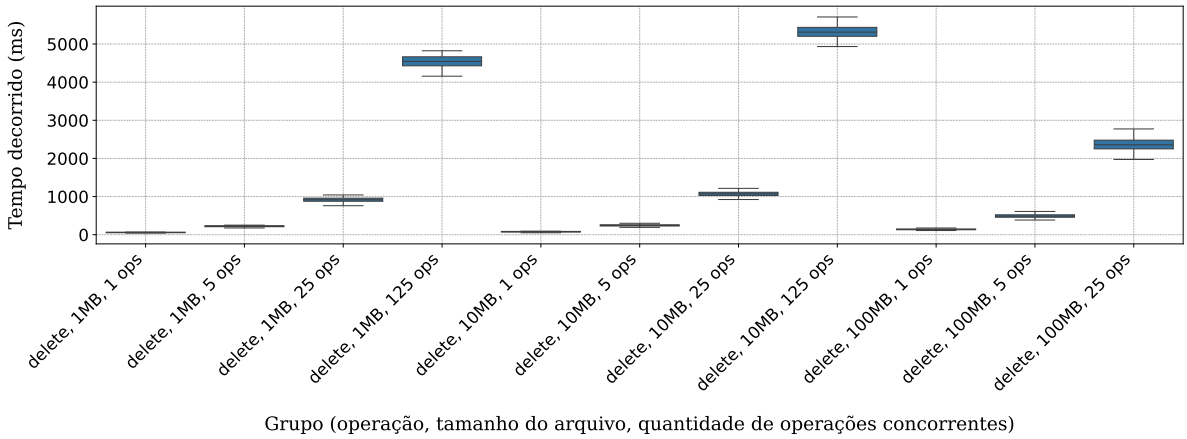
Por fim, as figuras 23, 24 e 25 apresentam o comportamento das operações `add`, `delete` e `retrieve`, visando analisar as diferenças de desempenho entre as funções, considerando simultaneamente o tamanho do arquivo e o grau de concorrência, e evidenciar quais são mais estáveis ou mais suscetíveis à degradação sob carga.

Figura 23 – *Box-plot* da duração da função `add` do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes



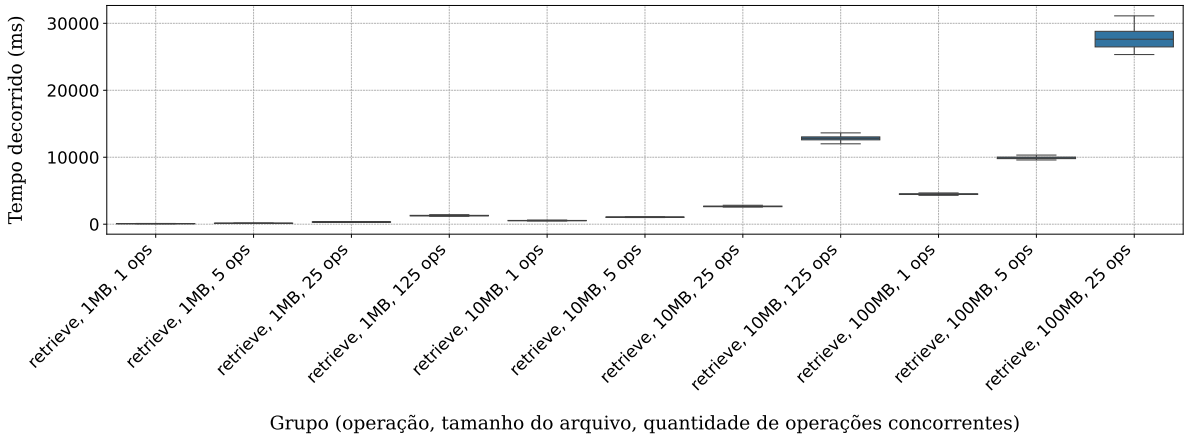
Fonte: elaborada pelo autor.

Figura 24 – *Box-plot* da duração da função `delete` do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes



Fonte: elaborada pelo autor.

Figura 25 – *Box-plot* da duração da função `retrieve` do experimento 2 por tamanho do arquivo e quantidade de operações concorrentes



Fonte: elaborada pelo autor.

Essas figuras permitem comparar diretamente as três operações analisadas. Os resultados confirmam que `delete` é a operação menos sensível a variações de carga, apresentando menor dispersão nos diferentes cenários. `Add` e `retrieve`, por outro lado, revelam comportamento mais instável: `add` sofre degradação acentuada em cargas maiores, enquanto `retrieve` combina aumento expressivo nos tempos de execução com forte variabilidade, sobretudo em arquivos grandes e maior concorrência.

Após a análise dos tempos de execução, a tabela 25 apresenta as taxas de *throughput* obtidas no experimento 2 para as operações `add`, `delete` e `retrieve`, considerando diferentes combinações de tamanho de arquivo e quantidade de operações concorrentes. São reportados os valores mínimos e máximos tanto da taxa individual, correspondente ao desempenho de

uma única operação, quanto da taxa global, que reflete o processamento conjunto de todas as operações executadas concorrentemente. Esses resultados ajudam a avaliar a escalabilidade do sistema sob distintas cargas de trabalho, fornecendo uma visão complementar às métricas de tempo apresentadas anteriormente.

Tabela 25 – Resultados numéricos de *throughput* das operações do experimento 2

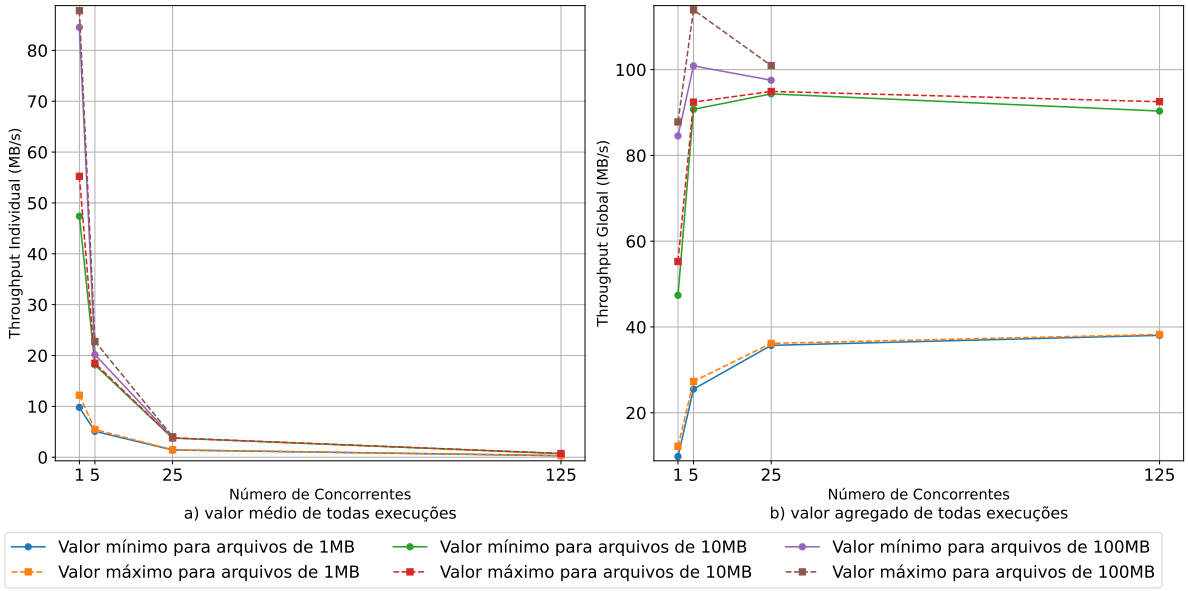
Operação	Tamanho do arquivo (MB)	Quantidade de operações concorrentes	Taxa individual mínima (MB/s)	Taxa individual máxima (MB/s)	Taxa global mínima (MB/s)	Taxa global máxima (MB/s)
	1	1	9,80	12,20	9,80	12,20
	1	5	5,10	5,46	25,51	27,32
	1	25	1,43	1,45	35,71	36,18
	1	125	0,30	0,31	38,05	38,25
	10	1	47,39	55,25	47,39	55,25
`add`	10	5	18,15	18,48	90,74	92,42
	10	25	3,77	3,80	90,34	94,91
	10	125	0,72	0,74	90,34	92,52
	100	1	84,53	87,80	84,53	87,80
	100	5	20,17	22,78	100,87	113,92
	100	25	3,90	4,04	97,53	100,94
	1	1	10,64	15,38	10,64	15,38
	1	5	4,05	4,41	20,24	22,03
	1	25	1,08	1,12	27,09	28,09
	1	125	0,22	0,22	27,17	27,63
	10	1	83,33	120,48	83,33	120,48
`delete`	10	5	35,71	40,98	178,57	204,92
	10	25	9,25	9,55	231,27	238,78
	10	125	1,87	1,90	234,08	237,46
	100	1	689,66	740,74	689,66	740,74
	100	5	199,20	206,19	996,02	1030,93
	100	25	41,49	42,25	1037,34	1056,19
	1	1	9,43	12,20	9,43	12,20
	1	5	6,76	7,14	33,78	35,71
	1	25	2,96	3,03	73,96	75,76
	1	125	0,72	0,74	90,58	92,25
	10	1	18,02	18,69	18,02	18,69
`retrieve`	10	5	9,39	9,55	46,95	47,76
	10	25	3,73	3,75	93,18	93,70
	10	125	0,73	0,74	90,63	92,42
	100	1	22,12	22,37	22,12	22,37
	100	5	9,54	10,03	47,72	50,14
	100	25	3,51	3,57	87,87	89,30

Fonte: elaborada pelo autor.

De modo a oferecer uma perspectiva visual complementar aos resultados dessa tabela, as figuras 23, 24 e 25 apresentam o *throughput* das operações `add`, `delete` e `retrieve`

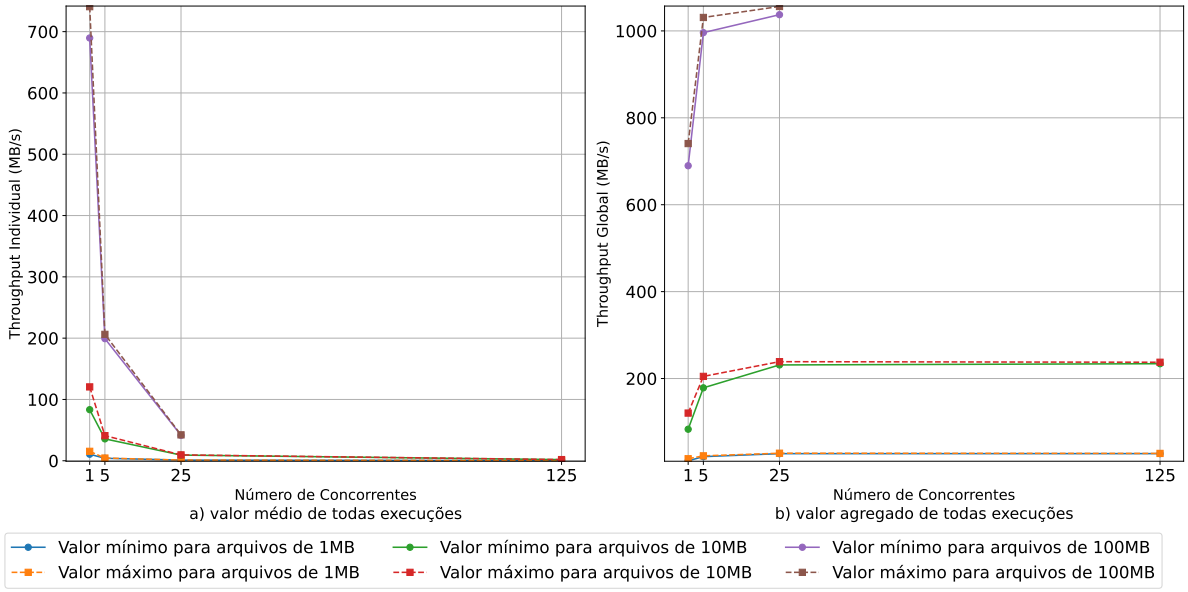
no experimento 2.

Figura 26 – Limites inferior e superior para o *throughput* da operação `add` no experimento 2, considerando o tamanho do arquivo e a quantidade de operações concorrentes



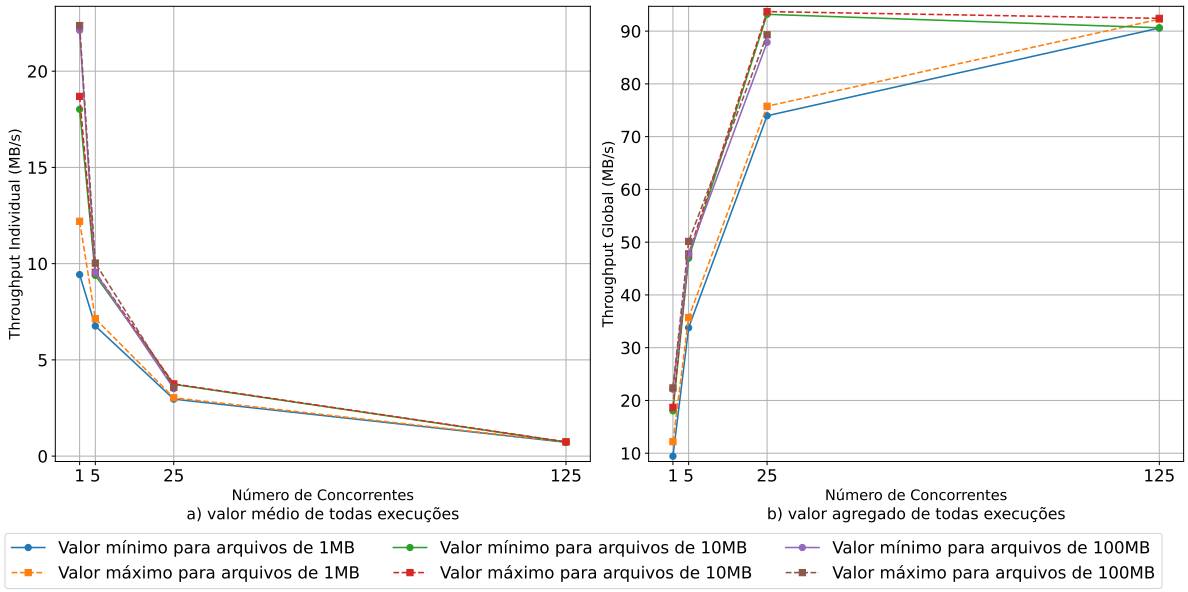
Fonte: elaborada pelo autor.

Figura 27 – Limites inferior e superior para o *throughput* da operação `delete` no experimento 2, considerando o tamanho do arquivo e a quantidade de operações concorrentes



Fonte: elaborada pelo autor.

Figura 28 – Limites inferior e superior para o *throughput* da operação `retrieve` no experimento 2, considerando o tamanho do arquivo e a quantidade de operações concorrentes



Fonte: elaborada pelo autor.

De acordo com essas figuras, observa-se que, de forma geral, o *throughput* global cresce quando se aumentam os tamanhos dos arquivos, enquanto o *throughput* individual tende a diminuir em razão do maior tempo médio de execução de cada operação. A elevação da concorrência também promove ganhos no *throughput* global, mas apenas até determinado ponto, já que, em cargas mais elevadas, o sistema atinge a saturação dos recursos computacionais disponíveis e o desempenho agregado deixa de crescer, podendo inclusive sofrer degradação em função do aumento do *i/o wait*. Esse comportamento se manifesta de forma clara nas operações `add` e `retrieve`, que apresentam perda de eficiência sob condições de carga intensa, ao passo que `delete`, mais uma vez, se mostra mais estável, preservando taxas elevadas e consistentes mesmo em cenários de alta concorrência. Esses resultados indicam que, embora a concorrência possa elevar a taxa de processamento agregado, há limites claros de escalabilidade impostos pela infraestrutura computacional utilizada nos testes.

5.6 Experimento 3: tempo de busca de custódias

A busca por custódias é uma das funcionalidades mais importantes providas pela G4DPA, sendo um dos diferenciais em relação às propostas de outros trabalhos relacionados. No cenário hipotético descrito neste capítulo, é ela que vai possibilitar ao médico verificar se o paciente possui registros anteriores que sejam úteis ao seu atendimento.

Para executar o experimento, inicialmente foram adicionadas 10.000 custódias através de chamadas ao GTW, que, por sua vez, realizou as correspondentes chamadas à função `addCustody` do contrato inteligente `G4DPA-SC`. Os campos de cada custódia foram gerados aleatoriamente de modo a haver registrados, ao final, 50 custódias com a palavra-chave "ECC" para cada um de 200 TDs.

Uma vez carregados os dados na BC, buscas foram realizadas por CRs da área de saúde, ou seja, que apresentam ao GTW certificados digitais com as extensões $role = controller$ e $segment = healthcare$, indicando em suas requisições a base legal de proteção dos interesses vitais do TD, conforme indicado nas tabelas 12 e 14 das seções 3.3.2.3 e 3.3.3.3, respectivamente.

O experimento foi conduzido em 5 etapas, cada uma com 100 rodadas, número este explicado na seção 5.2. Em cada etapa, a quantidade de CRs concorrentes foi fixada em uma das possibilidades 1, 5, 25, 125 e 625. Assim, foi obtido o tempo de cada requisição e calculado o tempo médio por busca, agrupando-o pela quantidade de CRs concorrentes. A tabela 26 apresenta os resultados obtidos, que são resumidos em seguida no gráfico da figura 29.

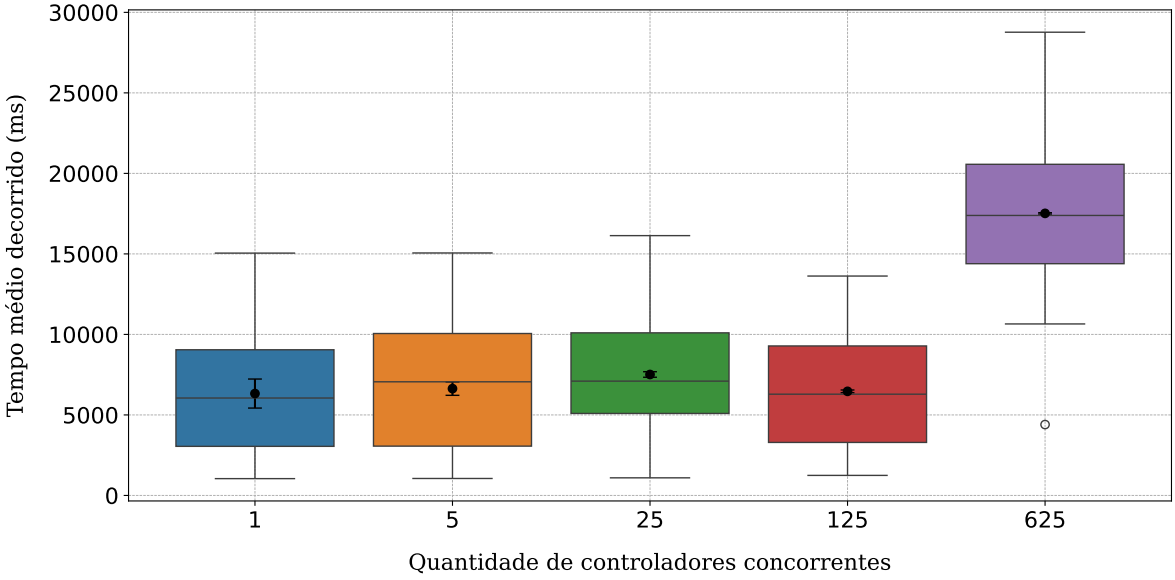
Tabela 26 – Resultados numéricos do experimento 3

CDs concorrentes	Tempo mín. (ms)	Tempo máx. (ms)	Média (ms)	Mediana (ms)	Desvio padrão (ms)	Amplitude do IC (99%)	IC inferior	IC superior
1	1044	15.049	6328	6050	3499	1803	5426	7229
5	1053	15.058	6633	7057	3628	833	6217	7050
25	1093	16.135	7512	7099	3346	343	7340	7683
125	1245	13.625	6464	6289	3590	171	6379	6550
625	4402	28.768	17.517	17.388	4003	83	17.475	17.558

Fonte: elaborada pelo autor.

Como pode ser observado, o tempo médio de uma busca se mantém aproximadamente o mesmo em todos os valores até 125 CRs concorrentes. Já ao empregar 625 CRs, o tempo médio praticamente triplica. Isso ocorre por conta do tempo de produção dos blocos. Uma transação somente é confirmada quando o bloco em que ela foi minerada é produzido. A periodicidade de produção de blocos foi configurada como a cada 12 segundos, como informado no início deste capítulo. As transações que não puderam ser processadas no bloco corrente permanecem pendentes para processamento nos blocos subsequentes. Assim, o gráfico da figura 29 está indicando que, repetidas as condições do experimento, há 99% de chance de o processamento das 625 requisições de busca ser realizado através de transações que ocorrem ao longo de 2 blocos.

Figura 29 – *Box-plot* do tempo de busca de custódias por quantidade de CRs concorrentes



Fonte: elaborada pelo autor.

5.7 Experimento 4: tempo de compartilhamento de custódias

O quarto e último experimento teve por objetivo analisar o desempenho da principal e mais complexa funcionalidade oferecida pela G4DPA, o compartilhamento de arquivos entre CDs, detalhada na seção 4.5.2. A ideia foi executar requisições de compartilhamento de arquivos de 1 MB concorrentemente entre controladores. A escolha deste tamanho de arquivo considerou, inicialmente, a ideia de testar o desempenho do GTW e da BC. As funções criptográficas sobre os arquivos, assim como sua transmissão através do IPFS, não ocorrem utilizando recursos do GTW diretamente, mas utilizando recursos dos controladores. Isso contribui para a melhor escalabilidade da solução, uma vez que o processamento e o armazenamento acabam distribuídos entre os próprios CDs.

Neste experimento, considerando a limitação do ambiente computacional disponível, dois nós IPFS serão utilizados. Os identificadores dos CDs são sequenciais, e se o identificador for par, usará um dos nós IPFS; se for ímpar, usará o outro. As requisições de compartilhamento são enviadas do controlador identificado por i para o controlador identificado por $i+1$, exceto o último, que enviará a notificação para o CD de menor identificador. Desse modo, será formado um anel, e cada CD envia e recebe exatamente uma requisição e uma notificação de compartilhamento, respectivamente, ou seja, ora atua como CR, ora como CC. O desempenho do compartilhamento foi avaliado na perspectiva da quantidade de operações de compartilhamento conduzidas concorrentemente, podendo ser 2, 5, 25 e 125.

Devido ao maior consumo de recursos computacionais deste experimento em relação ao experimento 3 e à limitação do *setup* disponível para os testes, restringiu-se a maior quantidade de operações realizadas de maneira concorrente a 125, considerando que, neste caso, os arquivos são lidos e criptografados pelos controladores, consumindo mais memória e produzindo mais operações de entrada e saída, especialmente considerando que os nós IPFS estão concorrendo pelos mesmos recursos de processamento, memória e *i/o* do disco. De todo modo, cada uma das possibilidades foi experimentada em 100 rodadas de forma separada, como motivado na seção 5.2. O tempo médio de cada compartilhamento completo foi registrado fase por fase e, uma vez concluídas todas as rodadas, foi calculado o tempo médio de compartilhamento por fase e por quantidade de operações concorrentes de compartilhamento. Ainda, como no experimento 2, foram utilizados arquivos binários com conteúdo aleatório, pelos mesmos motivos mencionados quando de sua análise.

Em preparação para execução do experimento, foram adicionadas através de chamadas ao GTW (e a `addCustody`, por consequência) 50 custódias para cada controlador dos 125 possíveis, totalizando 6250 custódias. Os campos de cada custódia foram gerados do mesmo modo que no experimento 3. Uma vez carregados os dados na BC, foram realizadas as requisições de compartilhamento por cada um dos CDs, sendo uma por rodada, de um total de 100 rodadas. Os resultados numéricos deste experimento estão listados na tabela 27.

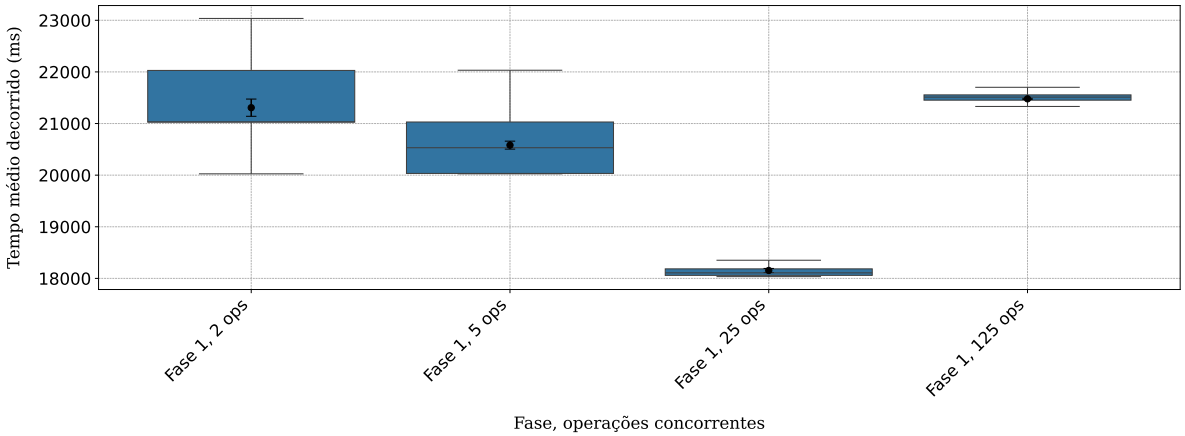
Tabela 27 – Resultados numéricos do experimento 4, para arquivo de 1 MB com até 125 operações concorrentes

Fase	Operações concorrentes	Tempo mín. (ms)	Tempo máx. (ms)	Média (ms)	Mediana (ms)	Desvio padrão (ms)	Amplitude do IC (99%)	IC inferior	IC superior
1	2	14.032	23.036	21.307	21.031	913	334	21.140	21.474
	5	20.026	24.029	20.581	20.531	634	154	20.504	20.658
	25	13.041	23.047	18.153	18.107	729	79	18.114	18.193
	125	18.619	21.720	21.479	21.512	302	22	21.468	21.489
2	2	12.106	16.154	12.381	12.149	547	201	12.280	12.482
	5	12.166	13.217	12.244	12.201	198	47	12.221	12.268
	25	12.594	12.916	12.693	12.685	67	9	12.689	12.698
	125	12.369	13.023	12.618	12.593	135	14	12.611	12.626
3	2	24.191	36.891	24.690	24.245	1404	515	24.432	24.947
	5	24.279	25.253	24.527	24.472	219	89	24.482	24.571
	25	24.561	25.061	24.809	24.816	73	15	24.801	24.817
	125	26.687	27.179	26.847	26.840	87	30	26.832	26.862
4	2	42	1331	81	70	128	47	58	105
	5	109	176	143	144	12	7	140	147
	25	373	1157	560	547	100	19	550	569
	125	3216	3828	3539	3533	109	39	3520	3559

Fonte: elaborada pelo autor.

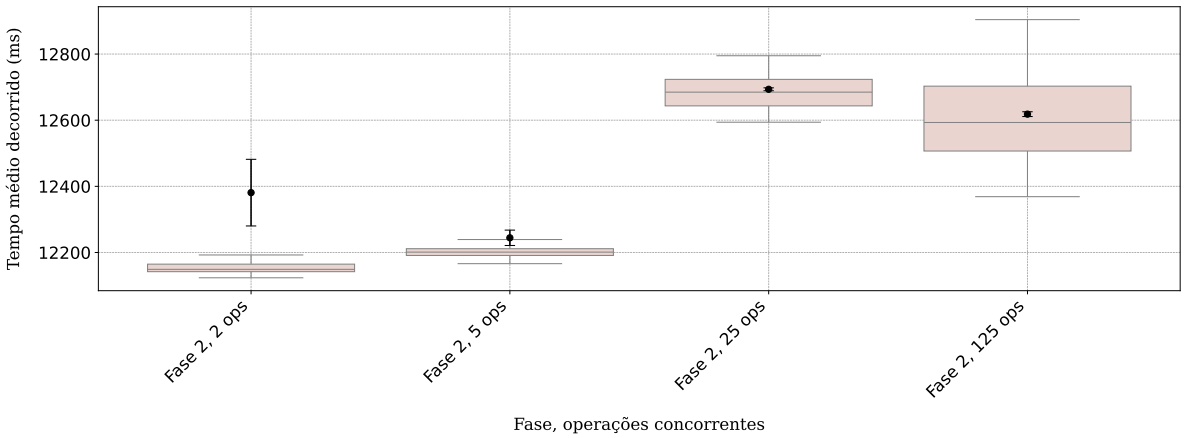
As figuras 30, 31, 32 e 33 apresentam gráficos *box-plot* para tempo de duração das fases 1, 2, 3 e 4, respectivamente, em função da quantidade de operações concorrentes de CRs, que podem ser 2, 5, 25 ou 125.

Figura 30 – *Box-plot* do tempo de duração da fase 1 por quantidade de CRs concorrentes



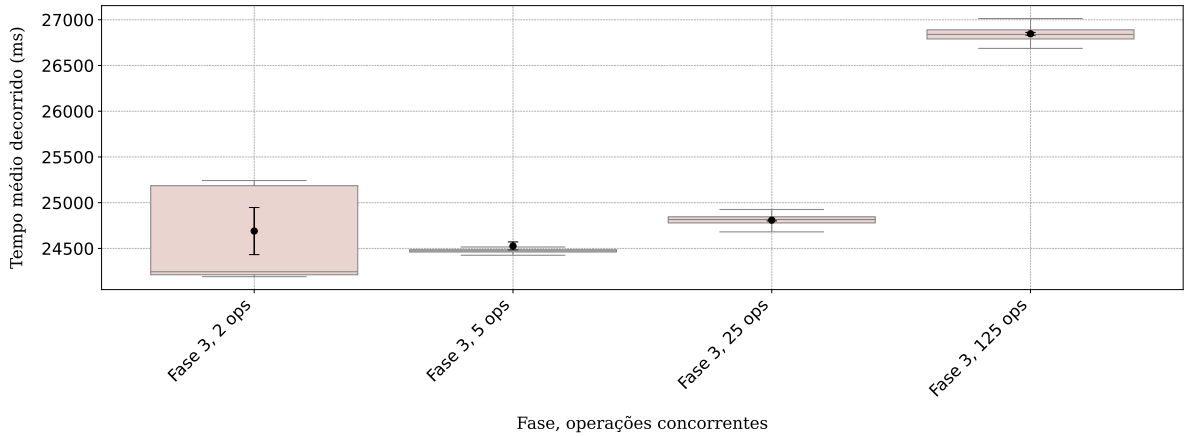
Fonte: elaborada pelo autor.

Figura 31 – *Box-plot* do tempo de duração da fase 2 por quantidade de CRs concorrentes



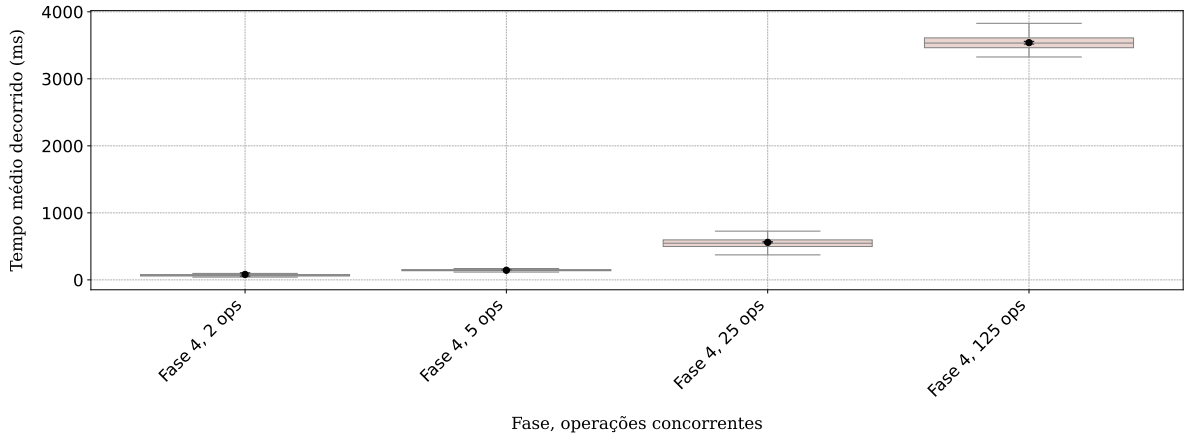
Fonte: elaborada pelo autor.

Figura 32 – *Box-plot* do tempo de duração da fase 3 por quantidade de CRs concorrentes



Fonte: elaborada pelo autor.

Figura 33 – *Box-plot* do tempo de duração da fase 4 por quantidade de CRs concorrentes

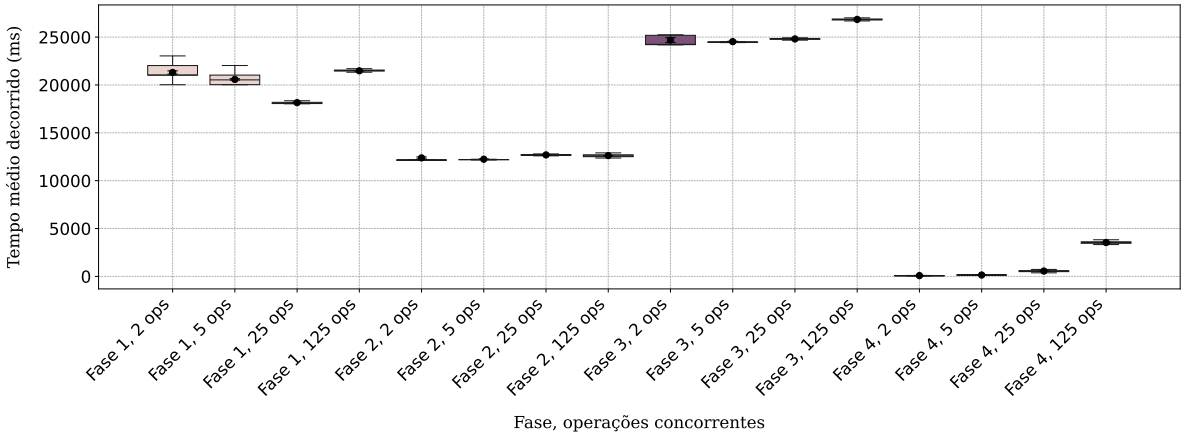


Fonte: elaborada pelo autor.

De modo comparativo, a figura 34 consolida as quatro figuras anteriores para o tempo decorrido nas fases 1, 2, 3 e 4, considerando 2, 5, 25 ou 125 operações concorrentes. Como se observa, não há uma grande variação produzida com o incremento da quantidade de operações concorrentes em cada fase, o que é um indício de que o GTW apresenta uma razoável escalabilidade. Considerando os tempos de transmissão de um arquivo através do IPFS vistos no experimento 2, que são menores que os reais por não levarem em conta a latência da rede, o tempo do processo de compartilhamento pode compensar o maior tempo necessário para transmitir arquivos maiores. Em outras palavras, a transmissão de arquivos muito pequenos através dessa implementação apresenta *overheads* maiores do que a transmissão de grandes arquivos. A ideia é que o tempo de transmissão do arquivo pela rede através do IPFS acabe sendo absorvido pelo tempo de espera pela produção dos blocos da rede Ethereum, o que acaba

limitando o tempo do processo de compartilhamento.

Figura 34 – *Box-plot* do tempo de duração de todas as fases por quantidade de CRs concorrentes



Fonte: elaborada pelo autor.

6 CONSIDERAÇÕES FINAIS

Neste capítulo, apresentam-se as considerações finais do trabalho. A seção 6.1 apresenta as principais contribuições, a seção 6.2 expõe as limitações do trabalho, a seção 6.3 discute as conclusões decorrentes da pesquisa e a seção 6.4 indica oportunidades de pesquisas futuras relacionadas ao tema.

6.1 Principais contribuições

As principais contribuições deste trabalho são apresentadas a seguir, de forma resumida.

1. Foi apresentada a arquitetura G4DPA, concebida para o compartilhamento de arquivos contendo dados pessoais em conformidade com a GDPR e a LGPD (CASTRO *et al.*, 2022). Entre seus componentes estão um *gateway* HTTP, responsável por receber requisições e anonimizar dados pessoais; uma *blockchain*, utilizada para armazenar metadados anonimizados de busca e para emitir notificações por meio de eventos; e nós *IPFS*, empregados para a transmissão efetiva dos arquivos.
2. Foi desenvolvido um mecanismo de troca de mensagens que assegura confidencialidade, integridade, autenticação mútua e não repúdio, permitindo que controladores de dados comuniquem-se com o *gateway* sem a necessidade de login ou transmissão de credenciais pela rede.
3. Foi proposto um mecanismo de auditoria, implementado sobre *blockchain*, que possibilita verificar se as operações realizadas sobre dados pessoais estão em conformidade com legislações de proteção de dados. Esse mecanismo registra informações adicionais, como a identidade pseudoanonimizada do titular, a identificação do controlador custodiante e do requisitante, bem como a base legal e o propósito do processamento. Diferentemente da maior parte dos trabalhos correlatos, esta proposta inclui explicitamente o registro da base legal e do propósito, ampliando o suporte à conformidade regulatória.
4. Por fim, destaca-se a funcionalidade de busca por custódias, considerada um dos principais diferenciais da G4DPA em relação a trabalhos existentes. Tal funcionalidade permite que, em cenários práticos, como o atendimento médico, seja possível identificar registros prévios de pacientes, localizar arquivos sob custódia de diferentes controladores e viabilizar seu eventual compartilhamento, sempre em conformidade com legislações de proteção de

dados e com suporte a auditoria.

6.2 Limitações

A seguir, são apontadas as principais limitações deste trabalho.

1. A busca bibliográfica restringiu-se à base Scopus em razão de limitações de tempo, e o procedimento não seguiu integralmente todas as etapas de uma revisão sistemática conforme proposto por (KITCHENHAM; CHARTERS, 2007), o que prejudicou a cobertura da busca pela literatura relevante. Apesar disso, a base Scopus é reconhecida como uma das mais extensas bases compreendendo publicações científicas com curadoria, oferecendo ampla cobertura global e regional de revistas e periódicos, além de agregar resultados de várias outras bases e agregadores (seção 2.1);
2. Presume-se que APD, AC e GTW são entidades confiáveis e cumprem suas atribuições honestamente, o que reduz o escopo do modelo de ameaças, podendo causar subestimativas dos riscos associados (seção 4.3);
3. A arquitetura requer uma PKI para emissão e revogação de certificados digitais das entidades comunicantes, de modo que a segurança das funcionalidades depende da robustez dessa infraestrutura e de uma adequada gestão do ciclo de vida dos certificados (seções 4.3 e 4.1);
4. Cada entidade necessita de um nó IPFS próprio, mantido sob seu domínio, para o funcionamento do mecanismo de exclusão de arquivos (seções 4.5 e 4);
5. Os experimentos foram conduzidos com uma *blockchain* Ethereum em modo de desenvolvimento em um único computador. Nesse caso, as latências entre as entidades são mínimas e geralmente diferentes da realidade, já que os contêineres executam em um mesmo *host*, e houve alguns testes que causaram sobrecarga de operações de *i/o*, limitando a análise. Além disso, não houve teste sobre *blockchains* de consórcio ou outras plataformas (seções 5.2 e 5.3).

6.3 Discussão

Boa parte dos trabalhos relacionados investigados não apresentou evidência de atenção em relação à conformidade com os regulamentos de proteção de dados pessoais. Apesar de ser relativamente novo, esse tema tem se tornado cada vez mais ubíquo. Dessa feita, é importante

que os trabalhos acadêmicos apresentem cuidado com o atendimento a esses normativos. Há também trabalhos que estranhamente armazenam dados pessoais diretamente na *blockchain*. Como visto, isso fere os direitos dos titulares de dados na GDPR e na LGPD, pois impede a exclusão e a retificação desses dados mediante solicitação do titular e, ainda, implica na disponibilidade dessa informação por todos os nós que processam as transações da *blockchain*. O único caso visto na literatura investigada que contorna essa situação é o artigo de (YEH *et al.*, 2023), que apresenta uma *blockchain* editável, mas que ainda é uma tecnologia iniciando o seu desenvolvimento e, a princípio, não apresenta, em contrapartida, o mesmo grau de imutabilidade de uma *blockchain* tradicional.

Como mencionado anteriormente, a Ethereum foi a *blockchain* escolhida para a prova de conceito deste trabalho por ter sido constatado que ela foi a mais utilizada nos trabalhos relacionados, e é versátil para permitir a configuração em rede pública (Mainnet) ou privada. Todavia, conforme os resultados dos experimentos, a utilização da G4DPA com uma rede pública Ethereum pode ser muito onerosa, considerando os custos de gas das funções do contrato inteligente `G4DPA-SC`, a depender da aplicação. Sendo esse o caso, é mais interessante configurar uma rede privada Ethereum, ou, ainda, avaliar a utilização de outras plataformas *blockchain*, como o HLF.

Outra questão sobre o uso de Ethereum diz respeito à latência e ao *throughput*. Como os blocos têm tamanho limitado e são produzidos atualmente a cada 12 segundos em média, há um limite para a quantidade de transações executadas por unidade de tempo. Obviamente, quanto menor o consumo de gas do contrato inteligente, melhor. Todavia, ainda assim há que se ter cuidado com as variações de custo dos *tokens* Ethereum (ETH). Isso porque, durante o funcionamento da rede, há variações no gasPrice, que é o parâmetro de conversão entre unidades de gas e ETH. Além disso, o preço do ETH também é flutuante em relação a moedas como dólar e real. Assim, em períodos de grande utilização da Ethereum, as transações podem ser mais caras e mais demoradas.

A arquitetura apresentada se mostrou funcional e viável para uso prático, sendo necessário, porém, escolher uma plataforma *blockchain* adequada ao caso de uso. Se os custos de uso da rede Ethereum Mainnet forem considerados elevados, é possível implantar uma rede privada Ethereum, restando necessário, porém, definir mecanismos de consenso, incentivo a mineradores, políticas de implantação, etc.

6.4 Trabalhos futuros

A pesquisa atual estabelece uma base para diversas investigações futuras no campo de blockchain e segurança de dados pessoais. A seguir, são apresentadas algumas direções técnicas promissoras para trabalhos futuros:

1. Uma possível solução para otimizar a escalabilidade e a eficiência da blockchain na arquitetura proposta, aumentando o *throughput* de transações e a diminuição dos custos de operação, seria utilizar uma *blockchain* de camada 2, ou seja, uma *blockchain* sobre *blockchain*.
2. Uma comparação entre a implementação atual, utilizando Ethereum, e uma adaptação utilizando a plataforma HLF seria interessante, de modo a expandir a compatibilidade de uso da arquitetura para outros contextos, avaliando desempenho, custo e segurança e identificando as vantagens e desvantagens através de experimentação.
3. Vários trabalhos relacionados utilizaram técnicas criptográficas avançadas, como ABE, *proxy-reencryption*, criptografia homomórfica e ZKP. Uma possibilidade seria substituir algumas das primitivas criptográficas utilizadas neste trabalho por algumas destas, visando melhorar a sua aplicabilidade, seu desempenho ou seu custo.
4. Investigar algoritmos que aprimorem a técnica de pseudoanonimização empregada, como esquemas de compartilhamento de segredo (e.g., SSSA), para mitigar o risco de comprometimento de uma senha armazenada.
5. Para a utilização desta arquitetura em uma plataforma Ethereum privada, é necessário realizar a implantação de nós de forma a alcançar segurança e latência adequada. Para isso, é necessário avaliar algoritmos de consenso e definir modelos de incentivo e punição para os participantes da rede.
6. Como visto, a segurança dos contratos inteligentes é um aspecto fundamental para a integridade das aplicações sobre plataformas blockchain. Assim, estudos de ferramentas, frameworks e metodologias para identificação e mitigação de vulnerabilidades são muito importantes para uma boa codificação, proporcionando segurança à execução dos contratos inteligentes.
7. Os experimentos deste trabalho foram realizados sobre recursos computacionais bastante limitados. A execução dos experimentos em um ambiente mais próximo do mundo real, com GTWs replicados, aplicações dos CDs e nós IPFS distante geograficamente, e

dispondo de recursos computacionais independentes, possibilitará uma melhor visualização sobre o desempenho e os custos de execução de um sistema baseado nesta arquitetura.

REFERÊNCIAS

ABDELGALIL, L.; MEJRI, M. Healthblock: A framework for a collaborative sharing of electronic health records based on blockchain. **Future Internet**, Multidisciplinary Digital Publishing Institute, [*S. l.*], v. 15, mar. 2023. ISSN 19995903.

ABOUALI, M.; SHARMA, K.; AJAYI, O.; SAADAWI, T. Blockchain framework for secured on-demand patient health records sharing. *In:* **2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)**. Nova Iorque, Estados Unidos: Institute of Electrical and Electronics Engineers, 2021. p. 0035–0040.

ALI, M. S.; VECCHIO, M.; PINCHEIRA, M.; DOLUI, K.; ANTONELLI, F.; REHMANI, M. H. Applications of blockchains in the internet of things: A comprehensive survey. **IEEE Communications Surveys & Tutorials**, [*S. l.*], v. 21, n. 2, p. 1676–1717, 2019.

ALOBADH, H.; YASIN, S. M.; UDZIR, N. I.; NINGGAL, M. I. H. Blockchain-based access control scheme for secure shared personal health records over decentralised storage. **Sensors**, Multidisciplinary Digital Publishing Institute, [*S. l.*], v. 21, abr. 2021. ISSN 14248220.

ANPD. **Balanço 3 anos**. 2023. Acesso em: 19 nov. 2023. Disponível em: https://www.gov.br/anpd/pt-br/documentos-e-publicacoes/anpd_balanco_tres_anos.pdf.

ANTONOPOULOS, A.; WOOD, G. **Mastering Ethereum: Building Smart Contracts and DApps**. [*S. l.*]: O'Reilly Media, 2018. ISBN 9781491971949. Disponível em: https://github.com/ethereumbook/ethereumbook/tree/develop?tab=readme-ov-file.

ARBABI, M. S.; LAL, C.; VEERARAGAVAN, N. R.; MARIJAN, D.; NYGåRD, J. F.; VITENBERG, R. A survey on blockchain for healthcare: Challenges, benefits, and future directions. **IEEE Communications Surveys & Tutorials**, [*S. l.*], v. 25, n. 1, p. 386–424, 2023.

BAAS, J.; SCHOTTEN, M.; PLUME, A.; CôTé, G.; KARIMI, R. Scopus as a curated, high-quality bibliometric data source for academic research in quantitative science studies. **Quantitative Science Studies**, [*S. l.*], v. 1, n. 1, p. 377–386, fev. 2020. ISSN 2641-3337. Disponível em: https://doi.org/10.1162/qss_a_00019.

BANDEKAR, P. P.; SUGUNA, G. C. LSB based text and image steganography using AES algorithm. *In:* **2018 3rd International Conference on Communication and Electronics Systems (ICCES)**. Coimbatore, Tamil Nadu, Índia: Institute of Electrical and Electronics Engineers, 2018. p. 782–788.

BELLARE, M.; ROGAWAY, P. Optimal asymmetric encryption. *In:* SANTIS, A. D. (Ed.). **Advances in Cryptology — EUROCRYPT'94**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. p. 92–111. ISBN 978-3-540-44717-7.

BELLARE, M.; ROGAWAY, P. The exact security of digital signatures-how to sign with RSA and rabin. *In:* MAURER, U. (Ed.). **Advances in Cryptology — EUROCRYPT '96**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. p. 399–416. ISBN 978-3-540-68339-1.

BELOTTI, M.; BOžIć, N.; PUJOLLE, G.; SECCI, S. A vademecum on blockchain technologies: When, which, and how. **IEEE Communications Surveys & Tutorials**, [*S. l.*], v. 21, n. 4, p. 3796–3838, 2019.

BENET, J. **IPFS - Content Addressed, Versioned, P2P File System**. 2014. Pré-*print* arXiv:1407.3561. Disponível em: http://arxiv.org/abs/1407.3561.

BOEYEN, S.; SANTESSON, S.; POLK, T.; HOUSLEY, R.; FARRELL, S.; COOPER, D. **Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**. [*S. l.*]: RFC Editor, 2008. RFC 5280. (Request for Comments, 5280). Disponível em: https://www.rfc-editor.org/info/rfc5280.

Brasil. **Lei N° 13.709, de 14 de agosto de 2018 - Lei geral de Proteção de Dados Pessoais (LGPD)**. 2018. 59–64 p. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709compilado.htm.

BUTERIN, V. **Ethereum Whitepaper**. 2014. Acesso em: 21 nov. 2023. Disponível em: https://ethereum.org/en/whitepaper/.

CASTRO, M. **G4DPA: código fonte (versão v1.0.0)**. 2024. Disponível em: https://doi.org/10.5281/zenodo.17180824.

CASTRO, M. de; PEREIRA, M.; CASTRO, M. de. Uma arquitetura baseada em blockchain para auditoria de conformidade com regulamentos de proteção de dados. *In:* **Anais do XXII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. Santa Maria, Rio Grande do Sul, Brasil: Sociedade Brasileira de Computação, 2022. p. 390–395. Disponível em: https://sol.sbc.org.br/index.php/sbseg/article/view/21684.

CHEN, C.-L.; YANG, J.; TSAUR, W.-J.; WENG, W.; WU, C.-M.; WEI, X. Enterprise data sharing with privacy-preserved based on hyperledger fabric blockchain in IIoTs application. **Sensors**, [*S. l.*], v. 22, n. 3, 2022. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/22/3/1146.

CMS Law. **GDPR Enforcement Tracker Report: Numbers and Figures**. 2023. Acesso em: 11 abr. 2023. Disponível em: https://cms.law/en/int/publication/gdpr-enforcement-tracker-report/numbers-and-figures.

CNUCED. **Data Protection and Privacy Legislation Worldwide**. 2024. Acesso em: 20 jun. 2024. Disponível em: .https://unctad.org/page/data-protection-and-privacy-legislation-worldwide.

DAVIS, D. Defective sign & encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML. *In:* **Proceedings of the 2001 USENIX Annual Technical Conference (USENIX ATC 01)**. Boston, Massachusetts, Estados Unidos: USENIX Association, 2001. p. 65–78. Disponível em: https://www.usenix.org/conference/2001-usenix-annual-technical-conference/defective-sign-encrypt-smime-pkcs7-moss-pem-pgp.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. **IEEE Transactions on Information Theory**, [*S. l.*], v. 22, n. 6, p. 644–654, 1976.

DLA Piper. **Data Protection Laws of the World**. 2022. Acesso em: 16 nov. 2023. Disponível em: https://www.dlapiperdataprotection.com/system/modules/za.co.heliosdesign.dla.lotw.data_protection/functions/handbook.pdf?country=all.

DWIVEDI, S. K.; AMIN, R.; VOLLALA, S. Blockchain-based secured IPFS-enable event storage technique with authentication protocol in VANET. **IEEE/CAA Journal of Automatica**

Sinica, Institute of Electrical and Electronics Engineers, [*S. l.*], v. 8, p. 1913–1922, dez. 2021. ISSN 23299274.

DWORKIN, M. **Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC**. [*S. l.*]: National Institute of Standards and Technology, 2007. SP 800-38D. (Security and Privacy, 800-38D). Disponível em: https://doi.org/10.6028/NIST.SP.800-38D.

DWORKIN, M. **Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption**. [*S. l.*]: National Institute of Standards and Technology, 2019. SP 800-38G Rev. 1. (Security and Privacy, 800-38G). Disponível em: https://doi.org/10.6028/NIST.SP.800-38Gr1-draft.

ENISA. **Pseudonymisation techniques and best practices**. 2019. Acesso em: 25 jun. 2024. Disponível em: https://www.enisa.europa.eu/publications/pseudonymisation-techniques-and-best-practices.

GAO, H.; MA, Z.; LUO, S.; XU, Y.; WU, Z. Bsspd: A blockchain-based security sharing scheme for personal data with fine-grained access control. **Wireless Communications and Mobile Computing**, Hindawi Limited, [*S. l.*], v. 2021, 2021. ISSN 15308677.

GEBREMICHAEL, T.; LEDWABA, L. P. I.; ELDEFRAWY, M. H.; HANCKE, G. P.; PEREIRA, N.; GIDLUND, M.; AKERBERG, J. Security and privacy in the industrial internet of things: Current standards and future challenges. **IEEE Access**, [*S. l.*], v. 8, p. 152351–152366, 2020.

GHANI, A.; ZINEDINE, A.; MOHAJIR, M. el. A blockchain-based secure PHR data storage and sharing framework. *In:* **2020 6th IEEE Congress on Information Science and Technology (CiSt)**. Agadir e Essaouira, Marrocos: Institute of Electrical and Electronics Engineers, 2020. p. 162–166.

GUO, Y.; WANG, S.; HUANG, J. A blockchain-assisted framework for secure and reliable data sharing in distributed systems. **Eurasip Journal on Wireless Communications and Networking**, Springer Science and Business Media Deutschland GmbH, [*S. l.*], dez. 2021. ISSN 16871499.

HEBBALLI, A. K.; J, B.; AGARWAL, A.; CHALLA, M. Securing medical data records using blockchain in a cloud computing environment. *In:* **2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)**. Bhilai, Madhya Pradesh, Índia: Institute of Electrical and Electronics Engineers, 2023. p. 1–5.

HOANG, V.-H.; LEHTIHET, E.; GHAMRI-DOUDANE, Y. Privacy-preserving blockchain-based data sharing platform for decentralized storage systems. *In:* **2020 IFIP Networking Conference (Networking)**. Paris, França: [*S. n.*], 2020. p. 280–288.

HOSSEIN, K. M.; ESMAEILI, M. E.; DARGAHI, T.; KHONSARI, A. Blockchain-based privacy-preserving healthcare architecture. *In:* **2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)**. Edmonton, Alberta, Canadá: Institute of Electrical and Electronics Engineers, 2019. p. 1–4.

HUANG, D. C.; LIU, L. C.; DENG, Y. Y.; CHEN, C. L.; ZENG, K. W. A hyperledger fabric-based EMR sharing mechanisms with proxy re-encryption and IPFS. **Cluster Computing**, Springer, [*S. l.*], 2024. ISSN 15737543.

HUANG, H.; LIN, J.; ZHENG, B.; ZHENG, Z.; BIAN, J. When blockchain meets distributed file systems: An overview, challenges, and open issues. **IEEE Access**, [*S. l.*], v. 8, p. 50574–50586, 2020.

IPFS. **IPFS Documentation Website - Concepts**. 2023. Acesso em: 16 mar. 2023. Disponível em: https://docs.ipfs.tech/concepts.

JABARULLA, M. Y.; LEE, H. N. Blockchain-based distributed patient-centric image management system. **Applied Sciences (Switzerland)**, Multidisciplinary Digital Publishing Institute, [*S. l.*], v. 11, p. 1–20, jan. 2021. ISSN 20763417.

JAIN, M.; JAILIA, M. Blockchain-based data sharing approach considering educational data. **International Journal of Information Security and Privacy**, [*S. l.*], v. 16, p. 1–20, jan. 2022.

JAIN, R. **The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling**. New York: John Wiley & Sons, 1991. ISBN 978-0471503361.

JIMENEZ, R.; OSMANI, F.; KNUTSSON, B. Sub-second lookups on a large-scale kademlia-based overlay. *In:* **2011 IEEE International Conference on Peer-to-Peer Computing**. Quioto, Japão: Institute of Electrical and Electronics Engineers, 2011. p. 82–91.

KAUR, J.; RANI, R.; KALRA, N. Attribute-based access control scheme for secure storage and sharing of EHRs using blockchain and IPFS. **Cluster Computing**, Springer, [*S. l.*], v. 27, p. 1047–1061, fev. 2024. ISSN 15737543.

KAVITHA, V.; SRUTHI, G.; THOSHINNY, B.; RIDUVARSHINI, S. Stagchain – a steganography based application working on a blockchain environment. *In:* **2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC)**. Coimbatore, Tamil Nadu, Índia: Institute of Electrical and Electronics Engineers, 2022. p. 674–681.

KITCHENHAM, B.; CHARTERS, S. **Procedures for Performing Systematic Reviews**. Keele, Inglaterra: Keele University, 2007.

KUMAR, H.; PRABHUDEVA. An authorization framework for preserving privacy of big medical data via blockchain in cloud server. **International Journal of Advanced Computer Science and Applications**, [*S. l.*], v. 13, 2022.

KUMAR, P.; KUMAR, R.; GARG, S.; KAUR, K.; ZHANG, Y.; GUIZANI, M. A secure data dissemination scheme for IoT-based e-health systems using AI and blockchain. *In:* **GLOBECOM 2022 - 2022 IEEE Global Communications Conference**. Rio de Janeiro, Brasil: Institute of Electrical and Electronics Engineers, 2022. p. 1397–1403.

KUNZ, I.; CASOLA, V.; SCHNEIDER, A.; BANSE, C.; SCHüTTE, J. Towards tracking data flows in cloud architectures. *In:* **2020 IEEE 13th International Conference on Cloud Computing (CLOUD)**. [*S. l.*]: Institute of Electrical and Electronics Engineers, 2020. p. 445–452.

LANGLEY, A.; HAMBURG, M.; TURNER, S. **Elliptic Curves for Security**. [*S. l.*]: RFC Editor, 2016. RFC 7748. (Request for Comments, 7748). Disponível em: https://www.rfc-editor.org/info/rfc7748.

LAW, A. M.; KELTON, W. D. **Simulation Modeling and Analysis**. 3rd. ed. Boston, Massachusetts, Estados Unidos: McGraw-Hill, 2000. ISBN 978-0070592926.

LI, H.; PEI, L.; LIAO, D.; WANG, X.; XU, D.; SUN, J. BDDT: use blockchain to facilitate IoT data transactions. **Cluster Computing**, Springer, [*S. l.*], v. 24, p. 459–473, mar. 2021. ISSN 15737543.

LI, Y.; YU, Y.; WANG, X. Three-tier storage framework based on TBchain and IPFS for protecting IoT security and privacy. **ACM Transactions on Internet Technology**, Association for Computing Machinery, [*S. l.*], v. 23, ago. 2023. ISSN 15576051.

LI, Z.; ZHANG, J.; ZHANG, J.; ZHENG, Y.; ZONG, X. Integrated edge computing and blockchain: A general medical data sharing framework. **IEEE Transactions on Emerging Topics in Computing**, Institute of Electrical and Electronics Engineers, [*S. l.*], 2023. ISSN 21686750.

LIU, F.; WANG, P. A novel privacy protection method of residents' travel trajectories based on federated blockchain and interplanetary file systems in smart cities. **PeerJ Computer Science**, [*S. l.*], v. 9, 2023. Disponível em: https://api.semanticscholar.org/CorpusID:260258902.

LUU, L.; CHU, D.-H.; OLICKEL, H.; SAXENA, P.; HOBOR, A. Making smart contracts smarter. *In:* **Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security**. Viena, Áustria: Association for Computing Machinery, 2016. (CCS '16), p. 254–269. ISBN 9781450341394. Disponível em: https://doi.org/10.1145/2976749.2978309.

MAINUDDIN, M.; RAHMAN, M. M.; SAJID, Z. B.; ANIK, M. A. R. Blockchain-enabled ipfs cloud solution for storing and sharing patient data securely and efficiently. *In:* **2023 6th International Conference on Electrical Information and Communication Technology (EICT)**. Khulna, Bangladesh: Institute of Electrical and Electronics Engineers, 2023. p. 1–6.

MALLICK, S.; LENKA, D.; TRIPATHY, P.; RAO, D.; SHARMA, S.; RAY, N. A lightweight, secure, and scalable blockchain-fog-iomt healthcare framework with ipfs data storage for healthcare 4.0. **SN Computer Science**, [*S. l.*], v. 5, jan. 2024.

MAYMOUNKOV, P.; MAZIèRES, D. Kademlia: A peer-to-peer information system based on the XOR metric. *In:* **First International Workshop on Peer-to-Peer Systems**. Berlin, Heidelberg: Springer-Verlag, 2002. (IPTPS '01), p. 53–65. ISBN 3540441794.

MITTAL, S.; GHOSH, M. A three-phase framework for secure storage and sharing of healthcare data based on blockchain, IPFS, proxy re-encryption and group communication. **Journal of Supercomputing**, Springer, [*S. l.*], abr. 2023. ISSN 15730484.

MORIARTY, K.; KALISKI, B.; JONSSON, J.; RUSCH, A. **PKCS #1: RSA Cryptography Specifications Version 2.2**. [*S. l.*]: RFC Editor, 2016. RFC 8017. (Request for Comments, 8017). Disponível em: https://www.rfc-editor.org/info/rfc8017.

NAKAMOTO, S. **Bitcoin: A Peer-to-Peer Electronic Cash System**. 2008. Acesso em: 21 nov. 2023. Disponível em: https://assets.pubpub.org/d8wct41f/31611263538139.pdf.

NGUYEN, D. C.; PATHIRANA, P. N.; DING, M.; SENEVIRATNE, A. Blockchain for secure EHRs sharing of mobile cloud based e-health systems. **IEEE Access**, [*S. l.*], v. 7, p. 66792–66806, 2019.

NIST. **Secure Hash Standard (SHS)**. [*S. l.*]: National Institute of Standards and Technology, 2015. FIPS 180-4. (Federal Information Processing Standards, 180-4). Disponível em: https://doi.org/10.6028/NIST.FIPS.180-4.

NIST. **Advanced Encryption Standard (AES)**. [*S. l.*]: National Institute of Standards and Technology, 2023. FIPS 197. (Federal Information Processing Standards, 197). Disponível em: https://doi.org/10.6028/NIST.FIPS.197-upd1.

OH, J.; LEE, J. Y.; KIM, M. H.; PARK, Y.; PARK, K. S.; NOH, S. K. A secure personal health record sharing system with key aggregate dynamic searchable encryption. **Electronics (Switzerland)**, Multidisciplinary Digital Publishing Institute, [*S. l.*], v. 11, out. 2022. ISSN 20799292.

OMG. **Business Process Model And Notation**. 2010. Acesso em: 14 jul. 2024. Disponível em: https://www.omg.org/spec/BPMN.

PAAR, C.; PELZL, J. **Understanding Cryptography: A Textbook for Students and Practitioners**. [*S. l.*]: Springer Berlin Heidelberg, 2010. ISBN 9783642041006.

Parlamento Europeu, Conselho da União Europeia. **Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)**. 2016. 1–88 p. Disponível em: https://eur-lex.europa.eu/eli/reg/2016/679.

RAJ, R.; GHOSH, M. A blockchain based lightweight and secure access control framework for IoT-enabled supply chain. **Peer-to-Peer Networking and Applications**, Springer, [*S. l.*], 2024. ISSN 19366450.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, Association for Computing Machinery, Nova Iorque, Estados Unidos, v. 21, n. 2, p. 120–126, fev. 1978. ISSN 0001-0782. Disponível em: https://doi.org/10.1145/359340.359342.

ROSENBERG, M.; CONFESSORE, N.; CADWALLADR, C. **How Trump Consultants Exploited the Facebook Data of Millions**. 2018. Disponível em: https://www.nytimes.com/2018/03/17/us/politics/cambridge-analytica-trump-campaign.html.

RUSSO, B.; VALLE, L.; BONZAGNI, G.; LOCATELLO, D.; PANCALDI, M.; TOSI, D. Cloud computing and the new EU general data protection regulation. **IEEE Cloud Computing**, [*S. l.*], v. 5, n. 6, p. 58–68, 2018.

SHARMA, S.; GHANSHALA, K. K.; MOHAN, S. Blockchain-based internet of vehicles (IoV): An efficient secure ad hoc vehicular networking architecture. *In:* **2019 IEEE 2nd 5G World Forum (5GWF)**. Dresden, Alemanha: Institute of Electrical and Electronics Engineers, 2019. p. 452–457.

SHREE, S.; ZHOU, C.; BARATI, M. Data protection in internet of medical things using blockchain and secret sharing method. **Journal of Supercomputing**, Springer, [*S. l.*], v. 80, p. 5108–5135, mar. 2024. ISSN 15730484.

SINGH, A.; RATHEE, G. A decentralized data sharing model using blockchain with fine grained access control. *In:* **2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)**. Greater Noida, Uttar Pradesh, Índia: Institute of Electrical and Electronics Engineers, 2023. p. 13–18.

SMART, N. P. **Cryptography Made Simple**. [*S. l.*]: Springer International Publishing, 2016. ISBN 9783319219356.

STALLINGS, W. **Cryptography and Network Security Principles and Practices, Fourth Edition**. [*S. l.*]: Prentice Hall, 2005. ISBN 9780131873162.

STOYCHEFF, E. Under surveillance: Examining facebook's spiral of silence effects in the wake of nsa internet monitoring. **Journalism and Mass Communication Quarterly**, [*S. l.*], v. 93, n. 2, p. 296 – 311, 2016.

SZABO, N. **Smart Contracts**. 1994. Acesso em: 20 mai. 2024. Disponível em: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html.

TIKKINEN-PIRI, C.; ROHUNEN, A.; MARKKULA, J. EU general data protection regulation: Changes and implications for personal data collecting companies. **Computer Law & Security Review**, [*S. l.*], v. 34, n. 1, p. 134–153, 2018. ISSN 0267-3649. Disponível em: https://www.sciencedirect.com/science/article/pii/S0267364917301966.

TONG, W.; YANG, L.; LI, Z.; JIN, X.; TAN, L. Enhancing security and flexibility in the industrial internet of things: Blockchain-based data sharing and privacy protection. **Sensors**, Multidisciplinary Digital Publishing Institute, [*S. l.*], v. 24, fev. 2024. ISSN 14248220.

TRAUTWEIN, D.; RAMAN, A.; TYSON, G.; CASTRO, I.; SCOTT, W.; SCHUBOTZ, M.; GIPP, B.; PSARAS, Y. Design and evaluation of IPFS : A storage layer for the decentralized web. *In:* **ACM SIGCOMM 2022 Conference (SIGCOMM '22)**. Amsterdã, Países Baixos: Association for Computing Machinery, 2022.

TRUONG, N. B.; SUN, K.; LEE, G. M.; GUO, Y. GDPR-compliant personal data management: A blockchain-based solution. **IEEE Transactions on Information Forensics and Security**, [*S. l.*], v. 15, p. 1746–1761, 2020.

ULLAH, Z.; RAZA, B.; SHAH, H.; KHAN, S.; WAHEED, A. Towards blockchain-based secure storage and trusted data sharing scheme for IoT environment. **IEEE Access**, Institute of Electrical and Electronics Engineers, [*S. l.*], v. 10, p. 36978–36994, 2022. ISSN 21693536.

UPPAL, S.; KANSEKAR, B.; MEHER, P.; MINI, S.; TOSH, D. CareBlocks: A blockchain-based health information sharing framework for medical IoT. *In:* **2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)**. Noida, Uttar Pradesh, Índia: Institute of Electrical and Electronics Engineers, 2021. p. 928–933.

VERA-RIVERA, A.; HOSSAIN, E.; HUSSEIN, A. R. Exploring the intersection of consortium blockchain technologies and multi-access edge computing: Chronicles of a proof of concept demo. **IEEE Open Journal of the Communications Society**, [*S. l.*], v. 3, p. 2203–2236, 2022.

WEN, Y. F.; WANG, C. P. Data privacy mechanisms development and performance evaluation for personal and ubiquitous blockchain-based storage. **Journal of Supercomputing**, Springer, [*S. l.*], v. 79, p. 19636–19670, nov. 2023. ISSN 15730484.

WU, Z.; WILLIAMS, A. B.; PEROULI, D. Dependable public ledger for policy compliance, a blockchain based approach. *In:* **2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)**. Dallas, Texas, Estados Unidos: Institute of Electrical and Electronics Engineers, 2019. p. 1891–1900.

XIAO, Y.; ZHANG, N.; LOU, W.; HOU, Y. T. A survey of distributed consensus protocols for blockchain networks. **IEEE Communications Surveys & Tutorials**, v. 22, n. 2, p. 1432–1465, 2020.

XIE, Y.; HUANG, K.; YUAN, S.; LI, X.; LI, F. Versatile remote data checking scheme for cloud-assisted internet of things. **IEEE Internet of Things Journal**, [*S. l.*], PP, p. 1–1, jan. 2023.

YANG, H.; LI, Y.; ZHOU, H.; ZHAO, Y.; SONG, L. A research on the sharing platform of wild bird data in yunnan province based on blockchain and interstellar file system. **Sensors**, Multidisciplinary Digital Publishing Institute, [*S. l.*], v. 22, set. 2022. ISSN 14248220.

YEH, L. Y.; HSU, W. H.; SHEN, C. Y. GDPR-compliant personal health record sharing mechanism with redactable blockchain and revocable IPFS. **IEEE Transactions on Dependable and Secure Computing**, Institute of Electrical and Electronics Engineers, [*S. l.*], 2023. ISSN 19410018.

YEH, L. Y.; SHEN, N. X.; HWANG, R. H. Blockchain-based privacy-preserving and sustainable data query service over 5G-VANETs. **IEEE Transactions on Intelligent Transportation Systems**, Institute of Electrical and Electronics Engineers, [*S. l.*], v. 23, p. 15909–15921, set. 2022. ISSN 15580016.

ZHANG, Q.; ZHAO, Z. Distributed storage scheme for encryption speech data based on blockchain and IPFS. **Journal of Supercomputing**, Springer, [*S. l.*], v. 79, p. 897–923, jan. 2023. ISSN 15730484.

ZHENG, J.; HUANG, X.; ODOOM, J.; XIANG, Y. A privacy-aware electronic medical record sharing scheme based on blockchain and identity-based cryptography. *In:* **2023 International Conference on Blockchain Technology and Information Security (ICBCTIS)**. Los Alamitos, Califórnia, Estados Unidos: Institute of Electrical and Electronics Engineers, 2023. p. 94–100. Disponível em: https://doi.ieeecomputersociety.org/10.1109/ICBCTIS59921.2023.00022.

APÊNDICE A – CONTRATO INTELIGENTE G4DP-SC

```
pragma solidity ^0.8.26;

// Enums

enum LawfulBase {
    Consent, // 0, GDPR Art. 6, 1, a
    Contract, // 1, GDPR Art. 6, 1, b
    LegalObligation, // 2, GDPR Art. 6, 1, c
    VitalInterests, // 3, GDPR Art. 6, 1, d
    PublicTask, // 4, GDPR Art. 6, 1, e
    LegitimateInterests //5,  GDPR Art. 6, 1, f
}

enum DataKind {
    Essn, // 0
    Name, // 1
    BirthDate, // 2
    Address, // 3
    Email, // 4
    HealthData // 5
}

enum ProcessingMode {
    LocalStorage, // 0
    Cloud, // 1
    DataProcessorOutsourcing // 2
}

enum Purpose {
    Storage, // 0
    MarketingAnalysis, // 1
    SharingWithBusinessParties, // 2
    AutomatedDecisionMaking, // 3
    InternationalDataTranfer, // 4
    Research, // 5
    ElectiveHealthCare, // 6
    EmergencyHealthCare // 7
}

enum Group {
    Healthcare, // 0
    Government, // 1
    Industry, // 2
    Commerce // 3
}

```

```
47  // Structs
48
49  struct Custody {
50      uint64 startTs;
51      uint64 endTs;
52      bytes8 internalId;
53      bytes8 creationTs;
54      bytes8 keyword;
55      bytes16 dataControllerId;
56      bytes16 dataOwnerId;
57      bytes32 dataHash;
58      LawfulBase lawfulBase;
59      DataKind dataKind;
60      ProcessingMode processingMode;
61      Purpose purpose;
62  }
63
64  struct Consent {
65      uint64 idxCustody;
66      uint64 fromTs;
67      uint64 untilTs;
68      bytes16 dataOwnerId;
69      bytes16 dataCustodianId;
70      bytes16 dataRequesterId;
71      uint64 withdrawnSince;
72  }
73
74  struct Request {
75      uint64 startTs;
76      uint64 endTs;
77      bool deniedByCustodian;
78      uint64 custodyIdx;
79      bytes16 dataRequesterId;
80      LawfulBase lawfulBase;
81      DataKind dataKind;
82      Purpose purpose;
83  }
84
85  struct ErasureRequest {
86      uint64 nonce;
87  }
88
89  struct Erasure {
90      uint64 nonce;
91  }
92
93  struct RectificationRequest {
94      uint64 nonce;
```

```
95  }
96
97  struct Rectification {
98      uint64 nonce;
99  }
100
101 struct Member {
102     uint64 inactiveSince;
103 }
104
105 // Events
106
107 event LogCustody(uint64);
108 event LogRemoveCustody(bool);
109 event LogRequestSharing(bool, uint64);
110 event LogNotifySharingRequest(string indexed _dataControllerIdStr, uint64
    ↪  _sharingRequestWaitId);
111 event LogNotifyDataSent(string indexed _dataRequesterIdStr, uint64 _requestWaitId);
112 event LogNotifyDeleteDataFromIPFS(string indexed _dataControllerIdStr, uint64
    ↪  _requestWaitId);
113 event LogSearchCustody(
114     bytes16 indexed _dataRequesterId,
115     bytes16 indexed _dataOwnerId,
116     bytes16 _dataControllerId,
117     bytes8 sinceTs,
118     bytes8 untilTs,
119     bytes8 keyword,
120     LawfulBase _lawfulBase,
121     DataKind _dataKind,
122     Purpose _purpose
123 );
124 event LogEndRequest(bool);
125
126 // Contracts
127
128 contract G4DP {
129
130     // State variables
131
132     Custody[] public custodies;
133     mapping(bytes16 => uint64[]) public dataControllerCustodies;
134     mapping(bytes16 => uint64[]) public dataOwnerCustodies;
135
136     Request[] public requests;
137     mapping(bytes16 => uint64[]) public dataRequesterRequests;
138     mapping(bytes16 => uint64[]) public dataControllerRequests;
139     mapping(bytes16 => uint64[]) public dataOwnerRequests;
140
```

```
141     mapping(Group => mapping(bytes16 => Member[])) public memberOf;
142
143     // Functions
144
145     function isMemberOfGroup(bytes16 _dataControllerId, Group _group) public view
        ↪  returns (bool) {
146         uint64 length = uint64(memberOf[_group][_dataControllerId].length);
147         if (length > 0 && memberOf[_group][_dataControllerId][length-1].inactiveSince
            ↪  == 0) {
148             return true;
149         }
150
151         return false;
152     }
153
154     function addControllerToGroup(bytes16 _dataControllerId, Group _group) public
        ↪  returns (bool) {
155         if(isMemberOfGroup(_dataControllerId, _group)) {
156             return false;
157         }
158         memberOf[_group][_dataControllerId].push(Member(0));
159         return true;
160     }
161
162     function removeControllerFromGroup(bytes16 _dataControllerId, Group _group) public
        ↪  returns (bool) {
163         uint64 length = uint64(memberOf[_group][_dataControllerId].length);
164         if(isMemberOfGroup(_dataControllerId, _group)) {
165             memberOf[_group][_dataControllerId][length-1].inactiveSince =
                ↪  uint64(block.timestamp);
166         }
167
168         return false;
169     }
170
171     function addCustody(Custody memory _custody) public returns (uint64) {
172         uint64 idx = uint64(custodies.length);
173         _custody.startTs = uint64(block.timestamp);
174         _custody.endTs = 0;
175         custodies.push(_custody);
176         dataControllerCustodies[_custody.dataControllerId].push(idx);
177         dataOwnerCustodies[_custody.dataOwnerId].push(idx);
178         emit LogCustody(idx);
179         return idx;
180     }
181
182     function isCloseableCustody(uint64 _idxCustody, bytes16 _dataControllerId) public
        ↪  view returns (bool) {
```

```
183         if (_idxCustody < custodies.length && custodies[_idxCustody].dataControllerId
            ↪  == _dataControllerId && custodies[_idxCustody].endTs == 0) {
184             return true;
185         }
186         return false;
187     }
188
189     function closeCustody(uint64 _idxCustody, bytes16 _dataControllerId) public
        ↪  returns (bool) {
190         if (isCloseableCustody(_idxCustody, _dataControllerId)) {
191             custodies[_idxCustody].endTs = uint64(block.timestamp);
192             emit LogRemoveCustody(true);
193             return true;
194         }
195         emit LogRemoveCustody(false);
196         return false;
197     }
198
199     function registerSearch(
200         bytes16 _dataRequesterId,
201         bytes16 _dataOwnerId,
202         bytes16 _dataControllerId,
203         bytes8 _sinceTs,
204         bytes8 _untilTs,
205         bytes8 _keyword,
206         LawfulBase _lawfulBase,
207         DataKind _dataKind,
208         Purpose _purpose) public {
209             emit LogSearchCustody(_dataRequesterId, _dataOwnerId, _dataControllerId,
                ↪  _sinceTs, _untilTs, _keyword, _lawfulBase, _dataKind, _purpose);
210     }
211
212     function searchCustody(
213         bytes16 _dataRequesterId,
214         bytes16 _dataOwnerId,
215         bytes16 _dataControllerId,
216         LawfulBase _lawfulBase,
217         DataKind _dataKind,
218         Purpose _purpose
219     ) public view returns (uint64[] memory, Custody[] memory) {
220         // have to count before list
221         _dataControllerId = _dataControllerId;
222         uint64 j;
223         if (_lawfulBase == LawfulBase.VitalInterests && _dataKind ==
            ↪  DataKind.HealthData && _purpose == Purpose.EmergencyHealthCare) {
224             if (isMemberOfGroup(_dataRequesterId, Group.Healthcare)) {
225                 for (uint64 i = 0; i < dataOwnerCustodies[_dataOwnerId].length; i++) {
```

```
226                     if (custodies[dataOwnerCustodies[_dataOwnerId][i]].endTs == 0 &&
                        ↪  custodies[dataOwnerCustodies[_dataOwnerId][i]].dataKind ==
                        ↪  DataKind.HealthData) {
227                         j += 1;
228                     }
229                 }
230             }
231         }
232
233         uint64[] memory indexes = new uint64[](j);
234         Custody[] memory cs = new Custody[](j);
235         j = 0;
236         if (_lawfulBase == LawfulBase.VitalInterests && _dataKind ==
            ↪  DataKind.HealthData && _purpose == Purpose.EmergencyHealthCare) {
237             if (isMemberOfGroup(_dataRequesterId, Group.Healthcare)) {
238                 for (uint64 i = 0; i < dataOwnerCustodies[_dataOwnerId].length; i++) {
239                     if (custodies[dataOwnerCustodies[_dataOwnerId][i]].endTs == 0 &&
                        ↪  custodies[dataOwnerCustodies[_dataOwnerId][i]].dataKind ==
                        ↪  DataKind.HealthData) {
240                         indexes[j] = dataOwnerCustodies[_dataOwnerId][i];
241                         cs[j] = custodies[dataOwnerCustodies[_dataOwnerId][i]];
242                         j += 1;
243                     }
244                 }
245             }
246         }
247
248         return (indexes, cs);
249     }
250
251     function addRequestSharing(Request memory _request) public returns (bool, uint64)
        ↪  {
252         if (_request.lawfulBase == LawfulBase.VitalInterests && _request.dataKind ==
            ↪  DataKind.HealthData && _request.purpose == Purpose.EmergencyHealthCare) {
253             if (isMemberOfGroup(_request.dataRequesterId, Group.Healthcare)) {
254                 if (custodies[_request.custodyIdx].endTs == 0 &&
                    ↪  custodies[_request.custodyIdx].dataKind == DataKind.HealthData) {
255                     uint64 idx = uint64(requests.length);
256                     _request.startTs = uint64(block.timestamp);
257                     _request.endTs = 0;
258                     requests.push(_request);
259                     dataRequesterRequests[_request.dataRequesterId].push(idx);
260                     dataControllerRequests[custodies[_request.custodyIdx].
261                         dataControllerId].push(idx);
262                     dataOwnerRequests[custodies[_request.custodyIdx].dataOwnerId].
263                         push(idx);
264                     emit LogRequestSharing(true, idx);
265                     return (true, idx);
```

```
266                 }
267             }
268         }
269
270         emit LogRequestSharing(false, 0);
271         return (false, 0);
272     }
273
274     function notifySharingRequest(string calldata _dataControllerIdStr, uint64
    ↪ _sharingRequestWaitId) public {
275         emit LogNotifySharingRequest(_dataControllerIdStr, _sharingRequestWaitId);
276     }
277
278     function notifyRequester(string calldata _dataRequesterIdStr, uint64
    ↪ _requestWaitId) public {
279         emit LogNotifyDataSent(_dataRequesterIdStr, _requestWaitId);
280     }
281
282     function notifyDeleteDataFromIPFS(string calldata _dataControllerIdStr, uint64
    ↪ _requestWaitId) public {
283         emit LogNotifyDeleteDataFromIPFS(_dataControllerIdStr, _requestWaitId);
284     }
285
286     function denySharing(bytes16 _dataControllerId, uint64 _idxRequest) public returns
    ↪ (bool)  {
287         if (custodies[requests[_idxRequest].custodyIdx].dataControllerId ==
        ↪ _dataControllerId && requests[_idxRequest].endTs == 0) {
288             requests[_idxRequest].endTs = uint64(block.timestamp);
289             requests[_idxRequest].deniedByCustodian = true;
290
291             return true;
292         }
293
294         return false;
295     }
296
297     function endRequest(bytes16 _dataRequesterId, uint64 _idxRequest) public returns
    ↪ (bool) {
298         if (requests[_idxRequest].dataRequesterId == _dataRequesterId &&
        ↪ requests[_idxRequest].endTs == 0) {
299             requests[_idxRequest].endTs = uint64(block.timestamp);
300             requests[_idxRequest].deniedByCustodian = false;
301
302             emit LogEndRequest(true);
303             return true;
304         }
305
306         emit LogEndRequest(false);
```

```
307             return false;
308         }
309
310         function listAllCustodiesOfDataOwner(
311             bytes16 _dataOwnerId
312         ) external view returns (Custody[] memory) {
313             uint64 j = 0;
314             for (uint64 i = 0; i < dataOwnerCustodies[_dataOwnerId].length; i++) {
315                 if (custodies[dataOwnerCustodies[_dataOwnerId][i]].endTs == 0) {
316                     j += 1;
317                 }
318             }
319
320             Custody[] memory cs = new Custody[](j);
321             j = 0;
322             for (uint64 i = 0; i < dataOwnerCustodies[_dataOwnerId].length; i++) {
323                 if (custodies[dataOwnerCustodies[_dataOwnerId][i]].endTs == 0) {
324                     cs[j] = custodies[dataOwnerCustodies[_dataOwnerId][i]];
325                     j += 1;
326                 }
327             }
328
329             return cs;
330         }
331
332     }
```

APÊNDICE B – CÓDIGO-FONTE PRINCIPAL DO GATEWAY

```
1
2  package main
3
4  import (
5          "crypto/rsa"
6          "crypto/x509"
7          "flag"
8          "fmt"
9          "log"
10         "net/http"
11         "strconv"
12         "sync"
13         "time"
14
15         "br.marcosmc/g4dpa/dpanonymization"
16         "br.marcosmc/g4dpa/dpantireplay"
17         "br.marcosmc/g4dpa/dpconfig"
18         "br.marcosmc/g4dpa/dpcrypto"
19         "br.marcosmc/g4dpa/dpeth"
20         "br.marcosmc/g4dpa/dpidentification"
21         "br.marcosmc/g4dpa/dpipfs"
22         "br.marcosmc/g4dpa/dpmessages/dpproto"
23         "br.marcosmc/g4dpa/dpmetrics"
24         "br.marcosmc/g4dpa/dputil"
25
26         "google.golang.org/protobuf/proto"
27
28         "github.com/gin-gonic/gin"
29         "github.com/ipfs/kubo/client/rpc"
30 )
31
32 type cmdLineArguments struct {
33         gatewayCrtFile     *string
34         gatewayPrivKeyFile *string
35         rootCACrtFile      *string
36         subCACrtFile       *string
37         ethHost            *string
38         ethHttpPort        *string
39         ethWsPort          *string
40         contractAddress    *string
41         ipfsUrl            *string
42         ipfsHttpTimeout    *time.Duration
43 }
44
45 type commonCertificates struct {
46         gatewayRsaPrivKey    *rsa.PrivateKey
```

```
47          gatewayX509Cert      *x509.Certificate
48          rootCertPool         *x509.CertPool
49          intermediateCertPool *x509.CertPool
50  }
51
52  type commonConnections struct {
53          ethConn    *dpeth.EthereumConnection
54          ipfsRpcApi *rpc.HttpApi
55  }
56
57  type actionFunc func(*dpcrypto.Message, *commonCertificates, *commonConnections)
    ↪  (*proto.Message, error)
58
59  type sharing struct {
60          idxRequest          uint64
61          dataRequesterId     string
62          dataControllerId    string
63          requesterSharingId  uint64
64          controllerSharingId uint64
65          internalId          uint64
66          requesterDHPubKey   [32]byte
67          controllerDHPubKey  [32]byte
68          sharingProof        [32]byte
69          cid                 string
70          datahash            [32]byte
71  }
72
73  var sharings = struct {
74          sync.RWMutex
75          counterForRequester  uint64
76          counterForController uint64
77          dataRequester        map[uint64]*sharing
78          dataController       map[uint64]*sharing
79  }{
80          dataRequester:  make(map[uint64]*sharing),
81          dataController: make(map[uint64]*sharing),
82  }
83
84  func main() {
85          log.SetFlags(log.Ldate | log.Ltime | log.Lmicroseconds | log.Lshortfile)
86          gin.SetMode(gin.ReleaseMode)
87
88          arguments := parseCmdLine()
89
90          certs, conn, gatewayIdentification, err :=
            ↪  initializeGatewayVariables(arguments)
91          if err != nil {
92                  log.Println("error: could not initialize gateway variables")
```

```
93                  log.Fatalln(err)
94          }
95
96          fmt.Printf("Gateway Identification: %s\n", gatewayIdentification)
97
98          if dpconfig.GatewayMetricsEnabled {
99                  dpmetrics.Initialize()
100         }
101
102         router := gin.Default()
103
104         post(dpconfig.CustodyAddRoute, addCustodyAction, certs, conn, router)
105         post(dpconfig.CustodyCloseRoute, closeCustodyAction, certs, conn, router)
106         post(dpconfig.CustodySearchRoute, searchCustodyAction, certs, conn, router)
107         post(dpconfig.CustodyRequestSharingRoute, requestSharingCustodyAction, certs,
            ↪  conn, router)
108         post(dpconfig.SharingDetailRequestRoute, sharingDetailsAction, certs, conn,
            ↪  router)
109         post(dpconfig.SharingDataSentRoute, sharingDataSentAction, certs, conn,
            ↪  router)
110         post(dpconfig.WhatDataWasSharedRoute, whatDataWasSharedAction, certs, conn,
            ↪  router)
111         post(dpconfig.DataWasGrabbedRoute, dataWasGrabbedAction, certs, conn, router)
112         post(dpconfig.DataWasDecryptedRoute, dataWasDecryptedAction, certs, conn,
            ↪  router)
113         post(dpconfig.DataWasDeletedRoute, dataWasDeletedAction, certs, conn, router)
114
115         if dpconfig.GatewayMetricsEnabled {
116                 router.GET(dpconfig.NewExecutionRoute, func(ctx *gin.Context) {
117                         experiment :=
                        ↪  ctx.Query(dpconfig.ExecutionRouteParamExperiment)
118                         if experiment == "" {
119                                 ctx.JSON(http.StatusBadRequest, gin.H{"msg":
                                ↪  "experiment name is required"})
120                         }
121                         round, err :=
                        ↪  strconv.ParseUint(ctx.Query(dpconfig.ExecutionRouteParamRounds),
                        ↪  10, 0)
122                         if err != nil {
123                                 ctx.JSON(http.StatusBadRequest, gin.H{"msg": "could
                                ↪  not parse round"})
124                         }
125
126                         dpmetrics.NewExecution(experiment, uint(round))
127
128                         ctx.JSON(http.StatusOK, gin.H{"msg": "round " +
                        ↪  strconv.FormatUint(round, 10) + " of experiment " +
                        ↪  experiment + " created"})
```

```
129                 })
130
131                 router.GET(dpconfig.ExecutionReportRoute, func(ctx *gin.Context) {
132                         buffer, err := dpmetrics.CSV()
133                         if err != nil {
134                                 ctx.JSON(http.StatusInternalServerError, gin.H{"msg":
                                    ↪  "could not generate csv"})
135                         }
136
137                         ctx.Header("Content-Description", "File Transfer")
138                         ctx.Header("Content-Transfer-Encoding", "binary")
139                         ctx.Header("Content-Disposition", "attachment;
                            ↪  filename="+dpconfig.ReportFilename)
140
141                         ctx.Data(http.StatusOK, "application/octet-stream",
                            ↪  buffer.Bytes())
142                 })
143         }
144
145         router.Run(":" + dpconfig.GatewayPort)
146 }
147
148 func parseCmdLine() *cmdLineArguments {
149         arguments := &cmdLineArguments{}
150
151         arguments.gatewayCrtFile = flag.String("gateway.crt",
        ↪  dpconfig.DefaultGatewayCrt, "path to the gateway certificate file")
152         arguments.gatewayPrivKeyFile = flag.String("gateway.key",
        ↪  dpconfig.DefaultGatewayPrivKey, "path to the gateway private key file")
153         arguments.rootCACrtFile = flag.String("ca.root", dpconfig.DefaultRootCACrt,
        ↪  "path to the root ca certificate file")
154         arguments.subCACrtFile = flag.String("ca.intermediate",
        ↪  dpconfig.DefaultSubCACrt, "path to the intermediate ca certificate file")
155         arguments.ethHost = flag.String("eth.host", dpconfig.DefaultGatewayEthHost,
        ↪  "ethereum node hostname")
156         arguments.ethHttpPort = flag.String("eth.http.port",
        ↪  dpconfig.DefaultGatewayEthHttpPort, "ethereum http api port")
157         arguments.ethWsPort = flag.String("eth.ws.port",
        ↪  dpconfig.DefaultGatewayEthWsPort, "ethereum web socket api port")
158         arguments.contractAddress = flag.String("contract.address",
        ↪  dpconfig.ContractAddress, "g4dpa contract address")
159         arguments.ipfsUrl = flag.String("ipfs.url", dpconfig.DefaultClientIpfsUrl,
        ↪  "url of ipfs api")
160         arguments.ipfsHttpTimeout = flag.Duration("ipfs.timeout",
        ↪  dpconfig.DefaultClientIpfsHttpTimeout, "timeout of the http client used by
        ↪  ipfs client")
161
162         flag.Parse()
```

```
163
164         return arguments
165 }
166
167 func initializeGatewayVariables(arguments *cmdLineArguments) (*commonCertificates,
    ↪  *commonConnections, *dpidentification.Identification, error) {
168         var err error
169
170         rootCertPool, err := dputil.GetCertPool(*arguments.rootCACrtFile)
171         if err != nil {
172                 log.Println("error: could not build root ca cert pool")
173                 return nil, nil, nil, err
174         }
175
176         intermediateCertPool, err := dputil.GetCertPool(*arguments.subCACrtFile)
177         if err != nil {
178                 log.Println("error: could not build intermediate ca cert pool")
179                 return nil, nil, nil, err
180         }
181
182         gatewayX509Cert, err :=
         ↪  dputil.ReadX509CertificateFromFile(*arguments.gatewayCrtFile)
183         if err != nil {
184                 log.Println("error: could not read gateway certificate from file")
185                 return nil, nil, nil, err
186         }
187
188         gatewayRsaPrivKey, err :=
         ↪  dputil.ReadRsaPrivKeyFromFile(*arguments.gatewayPrivKeyFile)
189         if err != nil {
190                 log.Println("error: could not read gateway private key from file")
191                 return nil, nil, nil, err
192         }
193
194         ethConn, err := dpeth.NewEthConn(dpconfig.GatewayAccountEthPrivKey,
         ↪  *arguments.ethHost, *arguments.ethHttpPort, dpconfig.ContractAddress)
195         if err != nil {
196                 log.Println("error: could not initialize ethereum connection")
197                 return nil, nil, nil, err
198         }
199
200         ipfsRpcApi, err := rpc.NewURLApiWithClient(*arguments.ipfsUrl,
         ↪  &http.Client{Timeout: *arguments.ipfsHttpTimeout})
201         if err != nil {
202                 log.Println("error: could not open ipfs connection")
203                 return nil, nil, nil, err
204         }
205
```

```
206         certs := &commonCertificates{
207                 gatewayRsaPrivKey:    gatewayRsaPrivKey,
208                 gatewayX509Cert:      gatewayX509Cert,
209                 rootCertPool:         rootCertPool,
210                 intermediateCertPool: intermediateCertPool,
211         }
212
213         conn := &commonConnections{
214                 ethConn:    ethConn,
215                 ipfsRpcApi: ipfsRpcApi,
216         }
217
218         gatewayIdentification :=
            ↪ dpidentification.GetIdFromCertificate(gatewayX509Cert)
219
220         return certs, conn, gatewayIdentification, nil
221 }
222
223 func post(route string, action actionFunc, certs *commonCertificates, conn
    ↪ *commonConnections, router *gin.Engine) {
224         router.POST(route, func(ctx *gin.Context) {
225                 statusCode, wiredProtobResponse, err := doPostActions(certs, conn, ctx,
                    ↪ action)
226                 if err != nil {
227                         log.Printf("error: could not process post on route %s\n%s",
                            ↪ route, err.Error())
228                         errMsg := statusCodeToErrorMessage(statusCode)
229                         ctx.JSON(statusCode, gin.H{"error": errMsg})
230                         return
231                 }
232
233                 ctx.Data(http.StatusOK, "application/octet-stream",
                    ↪ wiredProtobResponse)
234         })
235 }
236
237 func statusCodeToErrorMessage(statusCode int) string {
238         var errMsg string
239         switch statusCode {
240         case http.StatusBadRequest:
241                 errMsg = "missing mandatory parameters"
242         case http.StatusUnauthorized:
243                 errMsg = "unauthorized client"
244         case http.StatusInternalServerError:
245                 errMsg = "server failure"
246         }
247         return errMsg
248 }
```

```
249
250 func doPostActions(certs *commonCertificates, conn *commonConnections, ctx
    ↪  *gin.Context, actionFunc actionFunc) (int, []byte, error) {
251         msg := ctx.PostForm(dpconfig.MsgParameterName)
252
253         if msg == "" {
254                 errMsg := "could not retrieve parameter"
255                 log.Printf("error: %s\n", errMsg)
256                 return http.StatusBadRequest, nil, fmt.Errorf("%v", errMsg)
257         }
258
259         openedMsg, err := dpcrypto.GetPlainMessage([]byte(msg),
          ↪  certs.gatewayRsaPrivKey, certs.rootCertPool, certs.intermediateCertPool)
260         if err != nil {
261                 log.Println("error: could not open dpmessage")
262                 return http.StatusUnauthorized, nil, err
263         }
264
265         protobResponse, err := actionFunc(openedMsg, certs, conn)
266         if err != nil {
267                 log.Println("error: could not do specific post action")
268                 return http.StatusInternalServerError, nil, err
269         }
270
271         wiredProtobResponse, err := proto.Marshal(*protobResponse)
272         if err != nil {
273                 log.Println("error: could not wire response")
274                 return http.StatusInternalServerError, nil, err
275         }
276
277         dpMessage := dpcrypto.Message{
278                 Timestamp:   uint64(time.Now().UnixNano()),
279                 Certificate: *certs.gatewayX509Cert,
280                 Data:        wiredProtobResponse,
281         }
282
283         encryptedDpMessage, err :=
          ↪  dpMessage.EncryptAndSign(openedMsg.Certificate.PublicKey.(*rsa.PublicKey),
          ↪  certs.gatewayRsaPrivKey, dpconfig.RandomSymmetricKeySize)
284         if err != nil {
285                 log.Println("error: could not protect response message before
                  ↪  sending")
286                 return http.StatusInternalServerError, nil, err
287         }
288
289         return http.StatusOK, encryptedDpMessage.Serialize(), nil
290 }
291
```

```
292 func addCustodyAction(plainMsg *dpcrypto.Message, certs *commonCertificates, conn
    ↪  *commonConnections) (*proto.Message, error) {
293         protobCustody := &dpproto.CustodyAddRequest{}
294         err := proto.Unmarshal(plainMsg.Data, protobCustody)
295         if err != nil {
296                 log.Println("error: could not parse custody")
297                 return nil, err
298         }
299
300         senderIdentification :=
            ↪  dpidentification.GetIdFromCertificate(&plainMsg.Certificate)
301         if !senderIdentification.IsController() {
302                 log.Println("error: sender is not autorized to register data
                    ↪  custodies")
303                 return nil, fmt.Errorf("sender of custody is not a data controller")
304         }
305         if uint8(protobCustody.DataKind) == uint8(dpproto.DataKindType_HealthData) &&
            ↪  !senderIdentification.IsHealthcareKind() {
306                 log.Println("error: sender is not autorized to register healthcare
                    ↪  data custodies")
307                 return nil, fmt.Errorf("sender of custody is not a healthcare data
                    ↪  controller")
308         }
309
310         nonceValid := dpantireplay.TestNonce(
311                 dpidentification.GetIdFromCertificate(&plainMsg.Certificate).GetName(),
312                 int64(plainMsg.Timestamp),
313                 dpconfig.NonceTTL,
314         )
315         if !nonceValid {
316                 log.Println("error: nonce not valid")
317                 return nil, fmt.Errorf("nonce sent is not valid for the sender")
318         }
319
320         if len(protobCustody.Keyword) > 8 { // max. 8 characters
321                 log.Println("error: keyword must have at most 8 characters")
322                 return nil, fmt.Errorf("invalid keyword size")
323         }
324
325         custody := &dpanonymization.Custody{
326                 StartTs:          0,
327                 EndTs:            0,
328                 InternalId:       protobCustody.GetInternalId(),
329                 CreationTs:
                    ↪  time.Unix(int64(protobCustody.GetCreationTimestamp()), 0),
330                 Keyword:          protobCustody.GetKeyword(),
331                 DataControllerId: plainMsg.Certificate.Subject.CommonName,
332                 DataOwnerId:      protobCustody.GetDataOwnerId(),
```

```
333                 DataHash:          protobCustody.GetHash(),
334                 LawfulBase:        uint8(protobCustody.GetLawfulBase()),
335                 DataKind:          uint8(protobCustody.GetDataKind()),
336                 ProcessingMode:    uint8(protobCustody.GetProcessingMode()),
337                 Purpose:           uint8(protobCustody.GetPurpose()),
338         }
339
340         plainIndex, err := dpeth.AddCustody(custody, conn.ethConn)
341         if err != nil {
342                 log.Println("error: could not send custody to blockchain")
343                 return nil, err
344         }
345
346         encryptedIndex, err :=
        ↪  dpcrypto.EncryptAesGcm(dputil.Uint64ToBytes(*plainIndex),
        ↪  []byte(dpconfig.GatewaySymmetricKey))
347         if err != nil {
348                 log.Println("error: could not encrypt index")
349                 return nil, err
350         }
351
352         serverMessage := "ok"
353         custodyAddResponse := dpproto.CustodyAddResponse{
354                 Receipt:       encryptedIndex,
355                 ServerMessage: &serverMessage,
356         }
357
358         var protobResponse proto.Message = &custodyAddResponse
359         return &protobResponse, nil
360 }
361
362 func closeCustodyAction(plainMsg *dpcrypto.Message, certs *commonCertificates, conn
    ↪  *commonConnections) (*proto.Message, error) {
363         protobCloseReq := &dpproto.CustodyCloseRequest{}
364         err := proto.Unmarshal(plainMsg.Data, protobCloseReq)
365         if err != nil {
366                 log.Println("error: could not parse close custody request")
367                 return nil, err
368         }
369
370         nonceValid := dpantireplay.TestNonce(
371                 dpidentification.GetIdFromCertificate(&plainMsg.Certificate).GetName(),
372                 int64(plainMsg.Timestamp),
373                 dpconfig.NonceTTL,
374         )
375         if !nonceValid {
376                 log.Println("error: nonce not valid")
377                 return nil, fmt.Errorf("nonce sent is not valid for the sender")
```

```
378         }
379
380         idxCustody, err := dpcrypto.DecryptAesGcm(protobCloseReq.Receipt,
            ↪  []byte(dpconfig.GatewaySymmetricKey))
381         if err != nil {
382                 log.Println("error: could not decrypt receipt")
383                 return nil, err
384         }
385
386         ok, err := dpeth.CloseCustody(plainMsg.Certificate.Subject.CommonName,
            ↪  dputil.BytesToUint64(idxCustody), conn.ethConn)
387         if err != nil {
388                 log.Println("error: could not close custody on the blockchain")
389                 return nil, err
390         }
391
392         serverMessage := "ok"
393         custodyCloseResponse := dpproto.CustodyCloseResponse{
394                 Ok:            ok,
395                 ServerMessage: &serverMessage,
396         }
397
398         var protobResponse proto.Message = &custodyCloseResponse
399         return &protobResponse, nil
400 }
401
402 func searchCustodyAction(plainMsg *dpcrypto.Message, certs *commonCertificates, conn
    ↪  *commonConnections) (*proto.Message, error) {
403         pbSearchCustody := &dpproto.CustodySearchRequest{}
404         err := proto.Unmarshal(plainMsg.Data, pbSearchCustody)
405         if err != nil {
406                 log.Println("error: could not parse search custodies message")
407                 return nil, err
408         }
409
410         nonceValid := dpantireplay.TestNonce(
411                 dpidentification.GetIdFromCertificate(&plainMsg.Certificate).GetName(),
412                 int64(plainMsg.Timestamp),
413                 dpconfig.NonceTTL,
414         )
415         if !nonceValid {
416                 log.Println("error: nonce not valid")
417                 return nil, fmt.Errorf("nonce sent is not valid for the sender")
418         }
419
420         if pbSearchCustody.Keyword != nil && len(*pbSearchCustody.Keyword) > 8 {
421                 log.Println("error: keyword could not have more than 8 characters")
422                 return nil, fmt.Errorf("invalid keyword size")
```

```
423         }
424
425         indexes, custodies, err := dpeth.SearchCustody(
426                 plainMsg.Certificate.Subject.CommonName,
427                 pbSearchCustody.DataOwnerId,
428                 pbSearchCustody.GetDataControllerId(),
429                 pbSearchCustody.GetSince(),
430                 pbSearchCustody.GetUntil(),
431                 pbSearchCustody.GetKeyword(),
432                 uint8(pbSearchCustody.LawfulBase),
433                 uint8(pbSearchCustody.DataKind),
434                 uint8(pbSearchCustody.Purpose),
435                 conn.ethConn,
436         )
437         if err != nil {
438                 log.Println("error: could not search custodies on the blockchain")
439                 return nil, err
440         }
441
442         var responseElements []*dpproto.CustodySearchResponseElement
443         for i, c := range custodies {
444                 // se c atende aos parâmetros since, until e keyword da busca, pois
                    ↪  esses parâmetros são apenas registrados na blockchain (o filtro é
                    ↪  feito pelo gateway)
445                 if pbSearchCustody.GetKeyword() == "" || pbSearchCustody.GetKeyword()
                    ↪  == c.Keyword {
446                         if pbSearchCustody.GetSince() == 0 ||
                            ↪  pbSearchCustody.GetSince() <= uint64(c.CreationTs.Unix())
                            ↪  {
447                                 if pbSearchCustody.GetUntil() == 0 ||
                                    ↪  pbSearchCustody.GetUntil() >=
                                    ↪  uint64(c.CreationTs.Unix()) {
448                                         encryptedIndex, err := dpcrypto.EncryptAesGcm(
449                                                 dputil.Uint64ToBytes(indexes[i]),
450                                                 []byte(dpconfig.GatewaySymmetricKey),
451                                         )
452                                         if err != nil {
453                                                 log.Println("error: could not encrypt
                                                    ↪  index")
454                                                 return nil, err
455                                         }
456                                         element :=
                                            ↪  dpproto.CustodySearchResponseElement{
457                                                 StartTs: c.StartTs,
458                                                 Receipt: encryptedIndex,
459                                         }
460                                         responseElements = append(responseElements,
                                            ↪  &element)
```

```
461                                 }
462                         }
463                 }
464         }
465
466         serverMessage := "ok"
467         custodySearchresponse := dpproto.CustodySearchResponse{
468                 Custodies:     responseElements,
469                 ServerMessage: &serverMessage,
470         }
471
472         var protobResponse proto.Message = &custodySearchresponse
473         return &protobResponse, nil
474 }
475
476 func requestSharingCustodyAction(plainMsg *dpcrypto.Message, certs *commonCertificates,
    ↪  conn *commonConnections) (*proto.Message, error) {
477         pbRequestSharing := &dpproto.ReqSharingRequest{}
478         err := proto.Unmarshal(plainMsg.Data, pbRequestSharing)
479         if err != nil {
480                 log.Println("error: could not parse request sharing message")
481                 return nil, err
482         }
483
484         nonceValid := dpantireplay.TestNonce(dpidentification.GetIdFromCertificate(
485                 &plainMsg.Certificate).GetName(),
486                 int64(plainMsg.Timestamp),
487                 dpconfig.NonceTTL,
488         )
489         if !nonceValid {
490                 log.Println("error: nonce not valid")
491                 return nil, fmt.Errorf("nonce sent is not valid for the sender")
492         }
493
494         idxCustody, err := dpcrypto.DecryptAesGcm(pbRequestSharing.IdxCustody,
        ↪  []byte(dpconfig.GatewaySymmetricKey))
495         if err != nil {
496                 log.Println("error: could not decrypt idxcustody")
497                 return nil, err
498         }
499
500         requestSharing := &dpanonymization.RequestSharing{
501                 StartTs:           0,
502                 EndTs:             0,
503                 DeniedByCustodian: false,
504                 CustodyIdx:        dputil.BytesToUint64(idxCustody),
505                 DataRequesterId:   plainMsg.Certificate.Subject.CommonName,
506                 LawfulBase:        uint8(pbRequestSharing.GetLawfulBase()),
```

```
507                 DataKind:          uint8(pbRequestSharing.GetDataKind()),
508                 Purpose:           uint8(pbRequestSharing.GetPurpose()),
509         }
510
511         ok, idxReq, err := dpeth.AddRequestSharing(requestSharing, conn.ethConn)
512         if err != nil {
513                 log.Println("error: could not request sharing on the blockchain")
514                 return nil, err
515         }
516
517         encryptedIndex, err := dpcrypto.EncryptAesGcm(dputil.Uint64ToBytes(*idxReq),
            ↪  []byte(dpconfig.GatewaySymmetricKey))
518         if err != nil {
519                 log.Println("error: could not encrypt index")
520                 return nil, err
521         }
522
523         dataControllerId, err :=
            ↪  dpeth.GetDataControllerId(dputil.BytesToUint64(idxCustody), conn.ethConn)
524         if err != nil {
525                 log.Println("error: could not retrieve data controller info")
526                 return nil, err
527         }
528
529         requesterIdentification :=
            ↪  dpidentification.GetIdFromCertificate(&plainMsg.Certificate)
530
531         sharingProof, err := dputil.RandByteSlice(32)
532         if err != nil {
533                 log.Println("error: could not generate sharing proof")
534                 return nil, err
535         }
536
537         internalIdDecrypted, dataHashDecrypted, err :=
            ↪  dpeth.GetInternalIdAndHash(dputil.BytesToUint64(idxCustody), conn.ethConn)
538         if err != nil {
539                 log.Println("error: could not get internal id and hash from
                    ↪  blockchain")
540                 return nil, err
541         }
542         internalId := dputil.BytesToUint64(internalIdDecrypted[:])
543
544         shr := &sharing{
545                 idxRequest:          *idxReq,
546                 dataRequesterId:     requesterIdentification.GetName(),
547                 dataControllerId:    string(dataControllerId[:]),
548                 requesterSharingId:  0,
549                 controllerSharingId: 0,
```

```
550                 internalId:          internalId,
551                 requesterDHPubKey:   [32]byte(pbRequestSharing.DhRequesterPubKey),
552                 controllerDHPubKey:  [32]byte{},
553                 sharingProof:        [32]byte(sharingProof),
554                 cid:                 "",
555                 datahash:            [32]byte(dataHashDecrypted),
556         }
557
558         // ##############################################################################
559         sharings.Lock()
560         {
561                 shr.controllerSharingId = sharings.counterForController
562                 sharings.counterForController = sharings.counterForController + 1
563
564                 shr.requesterSharingId = sharings.counterForRequester
565                 sharings.counterForRequester = sharings.counterForRequester + 1
566
567                 sharings.dataController[shr.controllerSharingId] = shr
568                 sharings.dataRequester[shr.requesterSharingId] = shr
569         }
570         sharings.Unlock()
571         // ##############################################################################
572
573         err = dpeth.NotifySharingRequest(string(dataControllerId[:]),
        ↪  shr.controllerSharingId, conn.ethConn)
574         if err != nil {
575                 log.Println("error: could not inform data controller about sharing
                ↪  request")
576                 return nil, err
577         }
578
579         serverMessage := "ok"
580         pbReqSharingResponse := dpproto.ReqSharingResponse{
581                 Ok:            ok,
582                 Receipt:       encryptedIndex,
583                 Request:       shr.requesterSharingId,
584                 ServerMessage: &serverMessage,
585         }
586
587         var protobResponse proto.Message = &pbReqSharingResponse
588         return &protobResponse, nil
589 }
590
591 func sharingDetailsAction(plainMsg *dpcrypto.Message, certs *commonCertificates, conn
    ↪  *commonConnections) (*proto.Message, error) {
592         protobDetails := &dpproto.SharingNotificationDetailRequest{}
593         err := proto.Unmarshal(plainMsg.Data, protobDetails)
594         if err != nil {
```

```
595                 log.Println("error: could not parse sharing detail request")
596                 return nil, err
597         }
598
599         nonceValid := dpantireplay.TestNonce(
600                 dpidentification.GetIdFromCertificate(&plainMsg.Certificate).GetName(),
601                 int64(plainMsg.Timestamp),
602                 dpconfig.NonceTTL,
603         )
604         if !nonceValid {
605                 log.Println("error: nonce not valid")
606                 return nil, fmt.Errorf("nonce sent is not valid for the sender")
607         }
608
609         // testar se quem enviou a solicitação realmente é quem deveria
610         custodianId := dpidentification.GetIdFromCertificate(&plainMsg.Certificate)
611         fmt.Println(custodianId)
612         fmt.Println(
613                 sharings.dataController[protobDetails.NotificationIndex].dataControllerId,
614         )
615         fmt.Println(custodianId.IsController())
616         if custodianId.GetName() !=
        ↪  sharings.dataController[protobDetails.NotificationIndex].dataControllerId
        ↪  || !custodianId.IsController() {
617                 log.Println("error: authorization failed")
618                 return nil, fmt.Errorf("error: http requester was not identified as
                    ↪  the controller associated to notification")
619         }
620
621         // ############################################################################
622         sharings.RLock()
623         shr := sharings.dataController[protobDetails.NotificationIndex]
624         sharings.RUnlock()
625         // ############################################################################
626
627         sharingProof, err := dputil.RandByteSlice(dpconfig.RandomSharingProofSize)
628         if err != nil {
629                 log.Println("error: could not generate \"sharing proof\"")
630                 return nil, err
631         }
632         msgWithSharingProof := dpcrypto.Message{
633                 Timestamp:   uint64(time.Now().UnixNano()),
634                 Certificate: *certs.gatewayX509Cert,
635                 Data:        sharingProof,
636         }
637         dpmsgWithSharingProof, err := msgWithSharingProof.EncryptAndSign(
638                 plainMsg.Certificate.PublicKey.(*rsa.PublicKey),
639                 certs.gatewayRsaPrivKey,
```

```
640                 dpconfig.RandomSymmetricKeySize,
641         )
642         if err != nil {
643                 log.Println("error: could not encrypt and sign message containing
                    ↪  \"sharing proof\" to controller")
644                 return nil, err
645         }
646
647         serverMessage := "ok"
648         shareDetailsResponse := dpproto.SharingNotificationDetailResponse{
649                 InternalId:        shr.internalId,
650                 DhRequesterPubKey: shr.requesterDHPubKey[:],
651                 EncryptedProof:    dpmsgWithSharingProof.Serialize(),
652                 ServerMessage:     &serverMessage,
653         }
654
655         var protobResponse proto.Message = &shareDetailsResponse
656         return &protobResponse, nil
657 }
658
659 func sharingDataSentAction(plainMsg *dpcrypto.Message, certs *commonCertificates, conn
    ↪  *commonConnections) (*proto.Message, error) {
660         protobSent := &dpproto.SharingDataSentRequest{}
661         err := proto.Unmarshal(plainMsg.Data, protobSent)
662         if err != nil {
663                 log.Println("error: could not parse sharing data sent request")
664                 return nil, err
665         }
666
667         nonceValid := dpantireplay.TestNonce(
668                 dpidentification.GetIdFromCertificate(&plainMsg.Certificate).GetName(),
669                 int64(plainMsg.Timestamp),
670                 dpconfig.NonceTTL,
671         )
672         if !nonceValid {
673                 log.Println("error: nonce not valid")
674                 return nil, fmt.Errorf("nonce sent is not valid for the sender")
675         }
676
677         // ##############################################################################
678         sharings.Lock()
679         shr := sharings.dataController[protobSent.NotificationIndex]
680         shr.cid = protobSent.Cid
681         shr.controllerDHPubKey = [32]byte(protobSent.DhCustodianPubKey)
682         sharings.Unlock()
683         // ##############################################################################
684
685         // testar se quem enviou a solicitação realmente é quem deveria
```

```
686         custodianId := dpidentification.GetIdFromCertificate(&plainMsg.Certificate)
687         if custodianId.GetName() !=
            ↪  sharings.dataController[protobSent.NotificationIndex].dataControllerId ||
            ↪  !custodianId.IsController() {
                    log.Println("error: authorization failed")
                    return nil, fmt.Errorf("error: http requester was not identified as
                    ↪  the controller associated to notification")
            }

            serverMessage := "ok"
            dataSentResponse := dpproto.SharingDataSentResponse{
                    ServerMessage: &serverMessage,
            }

            err = dpeth.NotifyDataSent(shr.dataRequesterId, shr.requesterSharingId,
            ↪  conn.ethConn)
            if err != nil {
                    log.Println("error: could not notify requester")
                    return nil, err
            }

            var protobResponse proto.Message = &dataSentResponse
            return &protobResponse, nil
    }

    func whatDataWasSharedAction(plainMsg *dpcrypto.Message, certs *commonCertificates,
    ↪  conn *commonConnections) (*proto.Message, error) {
            protobWhatData := &dpproto.WhatDataWasSentRequest{}
            err := proto.Unmarshal(plainMsg.Data, protobWhatData)
            if err != nil {
                    log.Println("error: could not parse what data was shared request")
                    return nil, err
            }

            nonceValid := dpantireplay.TestNonce(
                    dpidentification.GetIdFromCertificate(&plainMsg.Certificate).GetName(),
                    int64(plainMsg.Timestamp),
                    dpconfig.NonceTTL,
            )
            if !nonceValid {
                    log.Println("error: nonce not valid")
                    return nil, fmt.Errorf("nonce sent is not valid for the sender")
            }

            // testar se quem enviou a solicitação realmente é quem deveria
            requesterId := dpidentification.GetIdFromCertificate(&plainMsg.Certificate)
            if requesterId.GetName() !=
            ↪  sharings.dataController[protobWhatData.NotificationIndex].dataRequesterId
            ↪  || !requesterId.IsController() {
```

```
728                 log.Println("error: authorization failed")
729                 return nil, fmt.Errorf("error: http requester was not identified as
                    ↪  the requester associated to notification")
730         }
731
732         // ##############################################################################
733         sharings.RLock()
734         shr := sharings.dataRequester[protobWhatData.NotificationIndex]
735         sharings.RUnlock()
736         // ##############################################################################
737
738         serverMessage := "ok"
739         whatDataResponse := dpproto.WhatDataWasSentResponse{
740                 Cid:           shr.cid,
741                 ServerMessage: &serverMessage,
742         }
743
744         var protobResponse proto.Message = &whatDataResponse
745         return &protobResponse, nil
746 }
747
748 func dataWasGrabbedAction(plainMsg *dpcrypto.Message, certs *commonCertificates, conn
    ↪  *commonConnections) (*proto.Message, error) {
749         protobDataWasGrabbed := &dpproto.WhatDataWasSentRequest{}
750         err := proto.Unmarshal(plainMsg.Data, protobDataWasGrabbed)
751         if err != nil {
752                 log.Println("error: could not parse what data was shared request")
753                 return nil, err
754         }
755
756         nonceValid := dpantireplay.TestNonce(
757                 dpidentification.GetIdFromCertificate(&plainMsg.Certificate).GetName(),
758                 int64(plainMsg.Timestamp),
759                 dpconfig.NonceTTL,
760         )
761         if !nonceValid {
762                 log.Println("error: nonce not valid")
763                 return nil, fmt.Errorf("nonce sent is not valid for the sender")
764         }
765
766         // testar se quem enviou a solicitação realmente é quem deveria
767         requesterId := dpidentification.GetIdFromCertificate(&plainMsg.Certificate)
768         if requesterId.GetName() !=
        ↪  sharings.dataController[protobDataWasGrabbed.NotificationIndex].dataRequesterId
        ↪  || !requesterId.IsController() {
769                 log.Println("error: authorization failed")
770                 return nil, fmt.Errorf("error: http requester was not identified as
                    ↪  the requester associated to notification")
```

```
771         }
772
773         // ##############################################################################
774         sharings.RLock()
775         shr := sharings.dataRequester[protobDataWasGrabbed.NotificationIndex]
776         sharings.RUnlock()
777         // ##############################################################################
778
779         // marcar o compartilhamento como finalizado; de agora em diante, não
            ↪  conformidades lançarão eventos na blockchain
780         ok, err := dpeth.EndRequestSharing(shr.dataRequesterId, shr.idxRequest,
            ↪  conn.ethConn)
781         if err != nil || !ok {
782                 log.Println("error: could not register end of request sharing")
783                 return nil, err
784         }
785
786         dataWasGrabbedResponse := dpproto.DataWasGrabbedResponse{
787                 DhCustodianPubKey: shr.controllerDHPubKey[:],
788         }
789
790         var protobResponse proto.Message = &dataWasGrabbedResponse
791         return &protobResponse, nil
792 }
793
794 func dataWasDecryptedAction(plainMsg *dpcrypto.Message, certs *commonCertificates,
    ↪  conn *commonConnections) (*proto.Message, error) {
795         protobDataWasDecrypted := &dpproto.DataWasDecryptedRequest{}
796         err := proto.Unmarshal(plainMsg.Data, protobDataWasDecrypted)
797         if err != nil {
798                 log.Println("error: could not parse what data was shared request")
799                 return nil, err
800         }
801
802         nonceValid := dpantireplay.TestNonce(
803                 dpidentification.GetIdFromCertificate(&plainMsg.Certificate).GetName(),
804                 int64(plainMsg.Timestamp),
805                 dpconfig.NonceTTL,
806         )
807         if !nonceValid {
808                 log.Println("error: nonce not valid")
809                 return nil, fmt.Errorf("nonce sent is not valid for the sender")
810         }
811
812         // testar se quem enviou a solicitação realmente é quem deveria
813         requesterId := dpidentification.GetIdFromCertificate(&plainMsg.Certificate)
814         if requesterId.GetName() !=
            ↪  sharings.dataController[protobDataWasDecrypted.NotificationIndex].dataRequesterId
            ↪  || !requesterId.IsController() {
```

```
815                 log.Println("error: authorization failed")
816                 return nil, fmt.Errorf("error: http requester was not identified as
                    ↪  the requester associated to notification")
817         }
818
819         // ##############################################################################
820         sharings.RLock()
821         shr := sharings.dataRequester[protobDataWasDecrypted.NotificationIndex]
822         sharings.RUnlock()
823         // ##############################################################################
824
825         if shr.datahash != [32]byte(protobDataWasDecrypted.DataHash) {
826                 log.Println("error: wrong hash value")
827                 return nil, fmt.Errorf("hash value sent by requester does not match
                    ↪  hash value stored in blockchain")
828         }
829
830         // enviar notificação para apagar dados do IPFS
831         err = dpeth.NotifyDataDecrypted(shr.dataControllerId, shr.controllerSharingId,
        ↪  conn.ethConn)
832         if err != nil {
833                 log.Println("error: could not notify custodian")
834                 return nil, err
835         }
836
837         serverMessage := "you must imediatelly delete data from IPFS"
838         dataWasDecryptedResponse := &dpproto.DataWasDecryptedResponse{
839                 ServerMessage: &serverMessage,
840         }
841
842         var protobResponse proto.Message = dataWasDecryptedResponse
843         return &protobResponse, nil
844 }
845
846 func dataWasDeletedAction(plainMsg *dpcrypto.Message, certs *commonCertificates, conn
    ↪  *commonConnections) (*proto.Message, error) {
847         protobDataWasDeleted := &dpproto.DataWasDeletedRequest{}
848         err := proto.Unmarshal(plainMsg.Data, protobDataWasDeleted)
849         if err != nil {
850                 log.Println("error: could not parse what data was shared request")
851                 return nil, err
852         }
853
854         nonceValid := dpantireplay.TestNonce(
855                 dpidentification.GetIdFromCertificate(&plainMsg.Certificate).GetName(),
856                 int64(plainMsg.Timestamp),
857                 dpconfig.NonceTTL,
858         )
```

```
859         if !nonceValid {
860                 log.Println("error: nonce not valid")
861                 return nil, fmt.Errorf("nonce sent is not valid for the sender")
862         }
863
864         // testar se quem enviou a solicitação realmente é quem deveria
865         requesterId := dpidentification.GetIdFromCertificate(&plainMsg.Certificate)
866         if requesterId.GetName() !=
            ↪  sharings.dataRequester[protobDataWasDeleted.NotificationIndex].dataControllerId
            ↪  || !requesterId.IsController() {
867                 log.Println("error: authorization failed")
868                 return nil, fmt.Errorf("error: http requester was not identified as
                    ↪  the requester associated to notification")
869         }
870
871         go func() error {
872                 // ###################################################################
873                 sharings.RLock()
874                 shr := sharings.dataRequester[protobDataWasDeleted.NotificationIndex]
875                 sharings.RUnlock()
876                 // ###################################################################
877
878                 cid, err := dpipfs.ExistsOnIPFS(shr.cid, conn.ipfsRpcApi)
879                 if err != nil {
880                         log.Println("error: could not verify if cid existed on ipfs")
881                         return err
882                 }
883
884                 var serverMessage string
885
886                 if cid != nil {
887                         serverMessage = fmt.Sprintf("Attention: cid %s was not delete
                            ↪  from IPFS!", shr.cid)
888                         log.Println(serverMessage)
889                 } else {
890                         serverMessage = "ok"
891                         // ################################
892                         sharings.Lock()
893                         delete(sharings.dataController, shr.controllerSharingId)
894                         delete(sharings.dataRequester, shr.requesterSharingId)
895                         sharings.Unlock()
896                         // ################################
897                 }
898
899                 return nil
900         }()
901
902         serverMessage := "message received"
```

```
903         dataWasDecryptedResponse := &dpproto.DataWasDecryptedResponse{
904                 ServerMessage: &serverMessage,
905         }
906
907         var protobResponse proto.Message = dataWasDecryptedResponse
908         return &protobResponse, nil
909 }
910
```

APÊNDICE C – CÓDIGO-FONTE PRINCIPAL DE INTERAÇÃO COM A BLOCKCHAIN

```
1
2  package dpeth
3
4  import (
5          "context"
6          "crypto/ecdsa"
7          "fmt"
8          "log"
9          "math/big"
10         "runtime"
11         "sync"
12
13         "br.marcosmc/g4dpa/dpanonymization"
14         "br.marcosmc/g4dpa/dpconfig"
15         "br.marcosmc/g4dpa/dpcrypto"
16         "br.marcosmc/g4dpa/dpmetrics"
17         "br.marcosmc/g4dpa/dputil"
18         "br.marcosmc/g4dpa/solidity/dpcontract"
19         "github.com/ethereum/go-ethereum"
20         "github.com/ethereum/go-ethereum/accounts/abi/bind"
21         "github.com/ethereum/go-ethereum/common"
22         "github.com/ethereum/go-ethereum/core/types"
23         "github.com/ethereum/go-ethereum/crypto"
24         "github.com/ethereum/go-ethereum/ethclient"
25         "github.com/ethereum/go-ethereum/event"
26 )
27
28 const fileInternalIdSize = 8
29 const custodianIdSize = 16
30 const ownerIdSize = 16
31 const requesterIdSize = 16
32 const keywordSize = 8
33 const hashSize = 32
34
35 type MutexNonce struct {
36         value       uint64
37         lastPending uint64
38         mutex       sync.RWMutex
39 }
40
41 func (nonce *MutexNonce) getValue(client *ethclient.Client, ctx context.Context,
   ↪  fromAddress common.Address) (uint64, error) {
42         pending, err := client.PendingNonceAt(ctx, fromAddress)
43         if err != nil {
```

```
44                  log.Println("error")
45                  return 0, err
46          }
47
48          var ret uint64
49          {
50                  if pending != nonce.lastPending {
51                          nonce.value = pending
52                          nonce.lastPending = pending
53                  }
54
55                  ret = nonce.value
56                  nonce.value++
57          }
58
59          return ret, nil
60  }
61
62  type EthereumConnection struct {
63          fromAddress       common.Address
64          client            *ethclient.Client
65          auth              *bind.TransactOpts
66          g4dpContract      *dpcontract.G4dp
67          suggestedGasPrice *big.Int
68  }
69
70  func NewEthConn(accountPrivKey string, host string, port string, contractAddress
    ↪  string) (*EthereumConnection, error) {
71          privateKey, err := crypto.HexToECDSA(accountPrivKey)
72          if err != nil {
73                  log.Println("error")
74                  return nil, err
75          }
76
77          client, err := ethclient.Dial(fmt.Sprintf("http://%s:%s", host, port))
78          if err != nil {
79                  log.Println("error")
80                  return nil, err
81          }
82          defer client.Close()
83
84          chainID, err := client.ChainID(context.Background())
85          if err != nil {
86                  log.Println("error")
87                  return nil, err
88          }
89
90          auth, err := bind.NewKeyedTransactorWithChainID(privateKey, chainID)
```

```
91          if err != nil {
92                  log.Println("error")
93                  return nil, err
94          }
95
96          g4dpContract, err := dpcontract.NewG4dp(common.HexToAddress(contractAddress),
            ↪  client)
97          if err != nil {
98                  log.Println("error")
99                  return nil, err
100         }
101
102         publicKey := privateKey.Public()
103         publicKeyECDSA, ok := publicKey.(*ecdsa.PublicKey)
104         if !ok {
105                 log.Println("error")
106                 return nil, fmt.Errorf("error casting public key to ECDSA")
107         }
108         fromAddress := crypto.PubkeyToAddress(*publicKeyECDSA)
109
110         suggestedGasPrice, err := client.SuggestGasPrice(context.Background())
111         if err != nil {
112                 log.Println("error")
113                 return nil, err
114         }
115
116         ethConn := EthereumConnection{
117                 fromAddress:       fromAddress,
118                 client:            client,
119                 auth:              auth,
120                 g4dpContract:      g4dpContract,
121                 suggestedGasPrice: suggestedGasPrice,
122         }
123
124         return &ethConn, nil
125 }
126
127 func newG4dpSession(ethConn *EthereumConnection) (*dpcontract.G4dpSession, error) {
128         nonce, err := mxNonce.getValue(ethConn.client, context.Background(),
            ↪  ethConn.fromAddress)
129         if err != nil {
130                 log.Println("error")
131                 return nil, err
132         }
133
134         session := &dpcontract.G4dpSession{
135                 Contract: ethConn.g4dpContract,
136                 CallOpts: bind.CallOpts{
```

```
137                         Pending: false,
138                 },
139                 TransactOpts: bind.TransactOpts{
140                         From:     ethConn.auth.From,
141                         Signer:   ethConn.auth.Signer,
142                         GasLimit: dpconfig.GatewayEthFunctionGasLimit,
143                         GasPrice: ethConn.suggestedGasPrice,
144                         Nonce:    new(big.Int).SetUint64(nonce),
145                 },
146         }
147
148         return session, nil
149 }
150
151 var mxNonce MutexNonce
152
153 func newG4dpCallerSession(ethConn *EthereumConnection) (*dpcontract.G4dpCallerSession,
    ↪  error) {
154         session := &dpcontract.G4dpCallerSession{
155                 Contract: &ethConn.g4dpContract.G4dpCaller,
156                 CallOpts: bind.CallOpts{
157                         Pending: false,
158                 },
159         }
160
161         return session, nil
162 }
163
164 func DeployG4dp() (*common.Address, error) {
165         privateKey, err := crypto.HexToECDSA(dpconfig.GatewayAccountEthPrivKey)
166         if err != nil {
167                 log.Println("error")
168                 return nil, err
169         }
170
171         client, err := ethclient.Dial(dpconfig.DefaultGatewayEthHttpUrl)
172         if err != nil {
173                 log.Println("error")
174                 return nil, err
175         }
176         defer client.Close()
177
178         chainID, err := client.ChainID(context.Background())
179         if err != nil {
180                 log.Println("error")
181                 return nil, err
182         }
183
```

```
184         auth, err := bind.NewKeyedTransactorWithChainID(privateKey, chainID)
185         if err != nil {
186                 log.Println("error")
187                 return nil, err
188         }
189
190         contractAddress, tx, _, err := dpcontract.DeployG4dp(auth, client)
191         if err != nil {
192                 log.Println("error")
193                 return nil, err
194         }
195
196         log.Printf("Deploy cost: %d\n", tx.Cost().Div(tx.Cost(), tx.GasPrice()))
197
198         return &contractAddress, nil
199 }
200
201 func fnName() string {
202         pc, _, _, ok := runtime.Caller(1)
203         if !ok {
204                 return "?"
205         }
206
207         fn := runtime.FuncForPC(pc)
208         if fn == nil {
209                 return "?"
210         }
211
212         return fn.Name()
213 }
214
215 func AddCustody(custody *dpanonymization.Custody, _ethConn *EthereumConnection)
    ↪  (*uint64, error) {
216         anonymCustody, err := dpanonymization.AnonymizeCustody(custody,
        ↪  []byte(dpconfig.GatewaySymmetricKey))
217         if err != nil {
218                 log.Println("error: could not anonymize custody")
219                 return nil, err
220         }
221
222         mxNonce.mutex.Lock()
223         session, err := newG4dpSession(_ethConn)
224         if err != nil {
225                 log.Println("error")
226                 return nil, err
227         }
228         tx, err := session.AddCustody(*anonymCustody)
229         mxNonce.mutex.Unlock()
```

```
230         if err != nil {
231                 log.Println("error")
232                 return nil, err
233         }
234
235         // Cuidado! se configurarmos o tempo de produção de bloco pelo geth (em modo
            ↪  dev) como zero, as operações serão executadas na EVM sem produção de
            ↪  blocos,
236         // ou seja, a espera abaixo nunca terminará!
237         if dpconfig.WaitTxFinish {
238                 recp, err := bind.WaitMined(context.Background(), _ethConn.client, tx)
239                 if err != nil {
240                         log.Println("error")
241                         return nil, err
242                 }
243
244                 // a transação falhou. Nesta versão, não vamos reenviá-la, isso ficará
                    ↪  a cargo do cliente, enquanto for síncrono (ou seja, tiver wait).
245                 if recp.Status == 0 {
246                         log.Println("error")
247                         return nil, fmt.Errorf("transaction failed")
248                 }
249
250                 logCustody, err :=
                    ↪  _ethConn.g4dpContract.ParseLogCustody(*recp.Logs[0])
251                 if err != nil {
252                         log.Println("error")
253                         return nil, err
254                 }
255
256                 if dpconfig.GatewayMetricsEnabled {
257                         functionName := fnName()
258                         nonce := tx.Nonce()
259                         dpmetrics.AddMetric(&dpmetrics.Metric{
260                                 FunctionName: &functionName,
261                                 Nonce:        &nonce,
262                                 TxGasUsed:    &recp.GasUsed,
263                                 FileSize:     nil,
264                                 Duration:     nil,
265                         })
266                 }
267
268                 return &logCustody.Arg0, nil
269         } else {
270                 return nil, nil
271         }
272 }
273
```

```
274 func CloseCustody(_dataControllerId string, _idxCustody uint64, _ethConn
    ↪  *EthereumConnection) (bool, error) {
275         dataControllerIdAnonym, err :=
            ↪  dpcrypto.EncryptFpeFF1([]byte(dputil.SpacePad(_dataControllerId,
            ↪  custodianIdSize)), []byte(dpconfig.GatewaySymmetricKey))
276         if err != nil {
277                 log.Println("error: could not anonymize controllerId")
278                 return false, err
279         }
280
281         mxNonce.mutex.Lock()
282         session, err := newG4dpSession(_ethConn)
283         if err != nil {
284                 log.Println("error")
285                 return false, err
286         }
287         tx, err := session.CloseCustody(_idxCustody,
            ↪  [custodianIdSize]byte(dataControllerIdAnonym))
288         mxNonce.mutex.Unlock()
289         if err != nil {
290                 log.Println("error")
291                 return false, err
292         }
293
294         if dpconfig.WaitTxFinish {
295                 recp, err := bind.WaitMined(context.Background(), _ethConn.client, tx)
296                 if err != nil {
297                         log.Println("error")
298                         return false, err
299                 }
300
301                 // a transação falhou. Nesta versão, não vamos reenviá-la, isso ficará
                    ↪  a cargo do cliente, enquanto for síncrono (ou seja, tiver wait).
302                 if recp.Status == 0 {
303                         log.Println("error")
304                         return false, fmt.Errorf("transaction failed")
305                 }
306
307                 logCloseCustody, err :=
                    ↪  _ethConn.g4dpContract.ParseLogRemoveCustody(*recp.Logs[0])
308                 if err != nil {
309                         log.Println("error")
310                         return false, err
311                 }
312
313                 if dpconfig.GatewayMetricsEnabled {
314                         functionName := fnName()
315                         nonce := tx.Nonce()
```

```
316                         dpmetrics.AddMetric(&dpmetrics.Metric{
317                                 FunctionName: &functionName,
318                                 Nonce:        &nonce,
319                                 TxGasUsed:    &recp.GasUsed,
320                                 FileSize:     nil,
321                                 Duration:     nil,
322                         })
323                 }
324
325                 return logCloseCustody.Arg0, nil
326         } else {
327                 return false, nil
328         }
329 }
330
331 func registerSearch(
332         _dataRequesterIdAnonym []byte,
333         _dataControllerIdAnonym []byte,
334         _dataOwnerIdAnonym []byte,
335         _sinceTs []byte,
336         _untilTs []byte,
337         _keyword []byte,
338         _lawFulBase uint8,
339         _dataKind uint8,
340         _purpose uint8,
341         _ethConn *EthereumConnection,
342 ) error {
343         mxNonce.mutex.Lock()
344         session, err := newG4dpSession(_ethConn)
345         if err != nil {
346                 log.Println("error")
347                 return err
348         }
349         tx, err := session.RegisterSearch(
350                 [16]byte(_dataRequesterIdAnonym),
351                 [16]byte(_dataOwnerIdAnonym),
352                 [16]byte(_dataControllerIdAnonym),
353                 [8]byte(_sinceTs),
354                 [8]byte(_untilTs),
355                 [8]byte(_keyword),
356                 _lawFulBase,
357                 _dataKind,
358                 _purpose,
359         )
360         mxNonce.mutex.Unlock()
361         if err != nil {
362                 log.Println("error")
363                 return err
```

```
364         }
365
366         if dpconfig.WaitTxFinish {
367                 recp, err := bind.WaitMined(context.Background(), _ethConn.client, tx)
368                 if err != nil {
369                         log.Println("error")
370                         return err
371                 }
372
373                 // a transação falhou. Nesta versão, não vamos reenviá-la, isso ficará
                    ↪  a cargo do cliente, enquanto for síncrono (ou seja, tiver wait).
374                 if recp.Status == 0 {
375                         log.Println("error")
376                         return fmt.Errorf("transaction failed")
377                 }
378
379                 if dpconfig.GatewayMetricsEnabled {
380                         functionName := fnName()
381                         nonce := tx.Nonce()
382                         dpmetrics.AddMetric(&dpmetrics.Metric{
383                                 FunctionName: &functionName,
384                                 Nonce:        &nonce,
385                                 TxGasUsed:    &recp.GasUsed,
386                                 FileSize:     nil,
387                                 Duration:     nil,
388                         })
389                 }
390
391                 return nil
392         } else {
393                 return nil
394         }
395 }
396
397 func SearchCustody(
398         _dataRequesterId string,
399         _dataOwnerId string,
400         _dataControllerId string,
401         _since uint64,
402         _until uint64,
403         _keyword string,
404         _lawFulBase uint8,
405         _dataKind uint8,
406         _purpose uint8,
407         _ethConn *EthereumConnection,
408 ) ([]uint64, []dpanonymization.Custody, error) {
409         session, err := newG4dpCallerSession(_ethConn)
410         if err != nil {
```

```
411                 log.Println("error")
412                 return nil, nil, err
413         }
414
415         dataRequesterIdAnonym, err :=
            ↪  dpcrypto.EncryptFpeFF1([]byte(dputil.SpacePad(_dataRequesterId,
            ↪  requesterIdSize)), []byte(dpconfig.GatewaySymmetricKey))
416         if err != nil {
417                 log.Println("error: could not anonymize requesterId")
418                 return nil, nil, err
419         }
420
421         var dataControllerId string
422         if _dataControllerId == "" {
423                 dataControllerId = string(0x00000000000000000000000000000000)
424         }
425
426         dataControllerIdAnonym, err :=
            ↪  dpcrypto.EncryptFpeFF1([]byte(dputil.SpacePad(dataControllerId,
            ↪  custodianIdSize)), []byte(dpconfig.GatewaySymmetricKey))
427         if err != nil {
428                 log.Println("error: could not anonymize controllerId")
429                 return nil, nil, err
430         }
431
432         dataOwnerIdAnonym, err :=
            ↪  dpcrypto.EncryptFpeFF1([]byte(dputil.SpacePad(_dataOwnerId, ownerIdSize)),
            ↪  []byte(dpconfig.GatewaySymmetricKey))
433         if err != nil {
434                 log.Println("error: could not anonymize ownerId")
435                 return nil, nil, err
436         }
437
438         sinceAnonym, err := dpcrypto.EncryptFpeFF1(dputil.Uint64ToBytes(_since),
            ↪  []byte(dpconfig.GatewaySymmetricKey))
439         if err != nil {
440                 log.Println("error: could not anonymize since_date")
441                 return nil, nil, err
442         }
443
444         untilAnonym, err := dpcrypto.EncryptFpeFF1(dputil.Uint64ToBytes(_until),
            ↪  []byte(dpconfig.GatewaySymmetricKey))
445         if err != nil {
446                 log.Println("error: could not anonymize until_date")
447                 return nil, nil, err
448         }
449
450         keywordAnonym, err := dpcrypto.EncryptFpeFF1([]byte(dputil.SpacePad(_keyword,
            ↪  keywordSize)), []byte(dpconfig.GatewaySymmetricKey))
```

```
451         if err != nil {
452                 log.Println("error: could no        t anonymize keyword")
453                 return nil, nil, err
454         }
455
456         err = registerSearch(dataRequesterIdAnonym, dataControllerIdAnonym,
            ↪  dataOwnerIdAnonym, sinceAnonym, untilAnonym, keywordAnonym, _lawFulBase,
            ↪  _dataKind, _purpose, _ethConn)
457         if err != nil {
458                 log.Println("error: could not register search")
459                 return nil, nil, err
460         }
461
462         indexes, custodiesAnonym, err :=
            ↪  session.SearchCustody([requesterIdSize]byte(dataRequesterIdAnonym),
            ↪  [ownerIdSize]byte(dataOwnerIdAnonym),
            ↪  [custodianIdSize]byte(dataControllerIdAnonym), _lawFulBase, _dataKind,
            ↪  _purpose)
463         if err != nil {
464                 log.Println("error")
465                 return nil, nil, err
466         }
467
468         custodies := make([]dpanonymization.Custody, len(custodiesAnonym))
469
470         for k, v := range custodiesAnonym {
471                 c, err := dpanonymization.DeanonymizeCustody(&v,
                    ↪  []byte(dpconfig.GatewaySymmetricKey))
472                 if err != nil {
473                         log.Println("error: could not deanonymize custody")
474                         return nil, nil, err
475                 }
476
477                 custodies[k] = *c
478         }
479
480         return indexes, custodies, nil
481 }
482
483 func AddRequestSharing(_requestSharing *dpanonymization.RequestSharing, _ethConn
    ↪  *EthereumConnection) (bool, *uint64, error) {
484         anonymRequestSharing, err :=
            ↪  dpanonymization.AnonymizeRequestSharing(_requestSharing,
            ↪  []byte(dpconfig.GatewaySymmetricKey))
485         if err != nil {
486                 log.Println("error: could not anonymize requestSharing")
487                 return false, nil, err
488         }
```

```
489
490         mxNonce.mutex.Lock()
491         session, err := newG4dpSession(_ethConn)
492         if err != nil {
493                 log.Println("error")
494                 return false, nil, err
495         }
496         tx, err := session.AddRequestSharing(*anonymRequestSharing)
497         mxNonce.mutex.Unlock()
498         if err != nil {
499                 log.Println("error")
500                 return false, nil, err
501         }
502
503         if dpconfig.WaitTxFinish {
504                 recp, err := bind.WaitMined(context.Background(), _ethConn.client, tx)
505                 if err != nil {
506                         log.Println("error")
507                         return false, nil, err
508                 }
509
510                 // a transação falhou. Nesta versão, não vamos reenviá-la, isso ficara
                ↪  a cargo do cliente, enquanto for síncrono (ou seja, tiver wait).
511                 if recp.Status == 0 {
512                         log.Println("error")
513                         return false, nil, fmt.Errorf("transaction failed")
514                 }
515
516                 LogRequestSharing, err :=
                ↪  _ethConn.g4dpContract.ParseLogRequestSharing(*recp.Logs[0])
517                 if err != nil {
518                         log.Println("error")
519                         return false, nil, err
520                 }
521
522                 if dpconfig.GatewayMetricsEnabled {
523                         functionName := fnName()
524                         nonce := tx.Nonce()
525                         dpmetrics.AddMetric(&dpmetrics.Metric{
526                                 FunctionName: &functionName,
527                                 Nonce:        &nonce,
528                                 TxGasUsed:    &recp.GasUsed,
529                                 FileSize:     nil,
530                                 Duration:     nil,
531                         })
532                 }
533
534                 return LogRequestSharing.Arg0, &LogRequestSharing.Arg1, err
```

```
535         } else {
536                 return false, nil, err
537         }
538 }
539
540 func EndRequestSharing(_dataRequesterId string, _idxRequestSharing uint64, _ethConn
    ↪  *EthereumConnection) (bool, error) {
541         dataRequesterIdAnonym, err :=
            ↪  dpcrypto.EncryptFpeFF1([]byte(dputil.SpacePad(_dataRequesterId,
            ↪  requesterIdSize)), []byte(dpconfig.GatewaySymmetricKey))
542         if err != nil {
543                 log.Println("error: could not anonymize requesterId")
544                 return false, err
545         }
546
547         mxNonce.mutex.Lock()
548         session, err := newG4dpSession(_ethConn)
549         if err != nil {
550                 log.Println("error")
551                 return false, err
552         }
553         tx, err := session.EndRequest([16]byte(dataRequesterIdAnonym),
            ↪  _idxRequestSharing)
554         mxNonce.mutex.Unlock()
555         if err != nil {
556                 log.Println("error")
557                 return false, err
558         }
559
560         if dpconfig.WaitTxFinish {
561                 recp, err := bind.WaitMined(context.Background(), _ethConn.client, tx)
562                 if err != nil {
563                         log.Println("error")
564                         return false, err
565                 }
566
567                 // a transação falhou. Nesta versão, não vamos reenviá-la, isso ficara
                    ↪  a cargo do cliente, enquanto for síncrono (ou seja, tiver wait).
568                 if recp.Status == 0 {
569                         log.Println("error")
570                         return false, fmt.Errorf("transaction failed")
571                 }
572
573                 logEndRequest, err :=
                    ↪  _ethConn.g4dpContract.ParseLogEndRequest(*recp.Logs[0])
574                 if err != nil {
575                         log.Println("error")
576                         return false, err
```

```
577                 }
578
579                 if dpconfig.GatewayMetricsEnabled {
580                         functionName := fnName()
581                         nonce := tx.Nonce()
582                         dpmetrics.AddMetric(&dpmetrics.Metric{
583                                 FunctionName: &functionName,
584                                 Nonce:        &nonce,
585                                 TxGasUsed:    &recp.GasUsed,
586                                 FileSize:     nil,
587                                 Duration:     nil,
588                         })
589                 }
590
591                 return logEndRequest.Arg0, nil
592         } else {
593                 return false, nil
594         }
595 }
596
597 func NotifySharingRequest(dataControllerIdStr string, sharingRequestWaitId uint64,
    ↪  ethConn *EthereumConnection) error {
598         mxNonce.mutex.Lock()
599         session, err := newG4dpSession(ethConn)
600         if err != nil {
601                 log.Println("error")
602                 return err
603         }
604         tx, err := session.NotifySharingRequest(dataControllerIdStr,
        ↪  sharingRequestWaitId)
605         mxNonce.mutex.Unlock()
606         if err != nil {
607                 log.Println("error")
608                 return err
609         }
610
611         if dpconfig.WaitTxFinish {
612                 recp, err := bind.WaitMined(context.Background(), ethConn.client, tx)
613                 if err != nil {
614                         log.Println("error")
615                         return err
616                 }
617
618                 // a transação falhou. Nesta versão, não vamos reenviá-la, isso ficara
                ↪  a cargo do cliente, enquanto for síncrono (ou seja, tiver wait).
619                 if recp.Status == 0 {
620                         log.Println("error")
621                         return fmt.Errorf("transaction failed")
```

```
622                 }
623
624                 if dpconfig.GatewayMetricsEnabled {
625                         functionName := fnName()
626                         nonce := tx.Nonce()
627                         dpmetrics.AddMetric(&dpmetrics.Metric{
628                                 FunctionName: &functionName,
629                                 Nonce:        &nonce,
630                                 TxGasUsed:    &recp.GasUsed,
631                                 FileSize:     nil,
632                                 Duration:     nil,
633                         })
634                 }
635
636                 return nil
637         } else {
638                 return nil
639         }
640 }
641
642 func NotifyDataSent(dataRequesterIdStr string, requestWaitId uint64, ethConn
    ↪  *EthereumConnection) error {
643         mxNonce.mutex.Lock()
644         session, err := newG4dpSession(ethConn)
645         if err != nil {
646                 log.Println("error")
647                 return err
648         }
649         tx, err := session.NotifyRequester(dataRequesterIdStr, requestWaitId)
650         mxNonce.mutex.Unlock()
651         if err != nil {
652                 log.Println("error")
653                 return err
654         }
655
656         if dpconfig.WaitTxFinish {
657                 recp, err := bind.WaitMined(context.Background(), ethConn.client, tx)
658                 if err != nil {
659                         log.Println("error")
660                         return err
661                 }
662
663                 // a transação falhou. Nesta versão, não vamos reenviá-la, isso ficara
                    ↪  a cargo do cliente, enquanto for síncrono (ou seja, tiver wait).
664                 if recp.Status == 0 {
665                         log.Println("error")
666                         return fmt.Errorf("transaction failed")
667                 }
```

```
668
669                 if dpconfig.GatewayMetricsEnabled {
670                         functionName := fnName()
671                         nonce := tx.Nonce()
672                         dpmetrics.AddMetric(&dpmetrics.Metric{
673                                 FunctionName: &functionName,
674                                 Nonce:        &nonce,
675                                 TxGasUsed:    &recp.GasUsed,
676                                 FileSize:     nil,
677                                 Duration:     nil,
678                         })
679                 }
680
681                 return nil
682         } else {
683                 return nil
684         }
685 }
686
687 func NotifyDataDecrypted(dataControllerIdStr string, requestWaitId uint64, ethConn
    ↪  *EthereumConnection) error {
688         mxNonce.mutex.Lock()
689         session, err := newG4dpSession(ethConn)
690         if err != nil {
691                 log.Println("error")
692                 return err
693         }
694         tx, err := session.NotifyDeleteDataFromIPFS(dataControllerIdStr,
          ↪  requestWaitId)
695         mxNonce.mutex.Unlock()
696         if err != nil {
697                 log.Println("error")
698                 return err
699         }
700
701         if dpconfig.WaitTxFinish {
702                 recp, err := bind.WaitMined(context.Background(), ethConn.client, tx)
703                 if err != nil {
704                         log.Println("error")
705                         return err
706                 }
707
708                 // a transação falhou. Nesta versão, não vamos reenviá-la, isso ficará
                  ↪  a cargo do cliente, enquanto for síncrono (ou seja, tiver wait).
709                 if recp.Status == 0 {
710                         log.Println("error")
711                         return fmt.Errorf("transaction failed")
712                 }
```

```
713
714                 if dpconfig.GatewayMetricsEnabled {
715                         functionName := fnName()
716                         nonce := tx.Nonce()
717                         dpmetrics.AddMetric(&dpmetrics.Metric{
718                                 FunctionName: &functionName,
719                                 Nonce:        &nonce,
720                                 TxGasUsed:    &recp.GasUsed,
721                                 FileSize:     nil,
722                                 Duration:     nil,
723                         })
724                 }
725
726                 return nil
727         } else {
728                 return nil
729         }
730 }
731
732 func GetDataControllerId(idxCustody uint64, ethConn *EthereumConnection) (string,
    ↪  error) {
733         session, err := newG4dpCallerSession(ethConn)
734         if err != nil {
735                 log.Println("error")
736                 return "", err
737         }
738
739         c, err := session.Custodies(new(big.Int).SetUint64(idxCustody))
740         if err != nil {
741                 log.Println("error")
742                 return "", err
743         }
744
745         dataControllerId, err := dpcrypto.DecryptFpeFF1(c.DataControllerId[:],
        ↪  []byte(dpconfig.GatewaySymmetricKey))
746         if err != nil {
747                 log.Println("error: could not deanonymize controller id")
748                 return "", err
749         }
750
751         return dputil.SpaceUnpad(string(dataControllerId)), nil
752 }
753
754 func GetInternalIdAndHash(idxCustody uint64, ethConn *EthereumConnection)
    ↪  ([fileInternalIdSize]byte, [hashSize]byte, error) {
755         session, err := newG4dpCallerSession(ethConn)
756         if err != nil {
757                 log.Println("error")
```

```
758                 return [fileInternalIdSize]byte{}, [hashSize]byte{}, err
759         }
760
761         c, err := session.Custodies(new(big.Int).SetUint64(idxCustody))
762         if err != nil {
763                 log.Println("error")
764                 return [fileInternalIdSize]byte{}, [hashSize]byte{}, err
765         }
766
767         internalId, err := dpcrypto.DecryptFpeFF1(c.InternalId[:],
        ↪  []byte(dpconfig.GatewaySymmetricKey))
768         if err != nil {
769                 log.Println("error: could not deanonymize internal id")
770                 return [fileInternalIdSize]byte{}, [hashSize]byte{}, err
771         }
772
773         dataHash, err := dpcrypto.DecryptFpeFF1(c.DataHash[:],
        ↪  []byte(dpconfig.GatewaySymmetricKey))
774         if err != nil {
775                 log.Println("error: could not deanonymize data hash")
776                 return [fileInternalIdSize]byte{}, [hashSize]byte{}, err
777         }
778
779         return [fileInternalIdSize]byte(internalId), [hashSize]byte(dataHash), nil
780 }
781
782 func ListenEvents() error {
783         client, err := ethclient.Dial(dpconfig.DefaultGatewayEthWsUrl)
784         if err != nil {
785                 log.Println("error")
786                 return err
787         }
788         defer client.Close()
789
790         query := ethereum.FilterQuery{
791                 Addresses: []common.Address{
792                         common.HexToAddress(dpconfig.ContractAddress),
793                 },
794         }
795
796         logs := make(chan types.Log)
797
798         sub, err := client.SubscribeFilterLogs(context.Background(), query, logs)
799         if err != nil {
800                 log.Println("error")
801                 return err
802         }
803
```

```
804         for {
805                 select {
806                 case err := <-sub.Err():
807                         log.Println("error")
808                         return err
809                 case vLog := <-logs:
810                         log.Println(vLog)
811                 }
812         }
813 }
814
815 func ListenForSharingRequestEvents(dataControllerId string, client *ethclient.Client,
    ↪  contractAddress string) (event.Subscription, chan
    ↪  *dpcontract.G4dpLogNotifySharingRequest, error) {
816         g4dpContract, err := dpcontract.NewG4dp(common.HexToAddress(contractAddress),
        ↪  client)
817         if err != nil {
818                 log.Println("error: could not instantiate contract")
819                 return nil, nil, err
820         }
821
822         opts := bind.WatchOpts{
823                 Start:   nil,
824                 Context: nil,
825         }
826
827         sink := make(chan *dpcontract.G4dpLogNotifySharingRequest)
828         subscription, err := g4dpContract.WatchLogNotifySharingRequest(&opts, sink,
        ↪  []string{dataControllerId})
829         if err != nil {
830                 log.Println("error: could not subscribe to sharing request events")
831                 return nil, nil, err
832         }
833
834         return subscription, sink, nil
835 }
836
837 func ListenForDataSharedEvents(dataRequesterId string, client *ethclient.Client,
    ↪  contractAddress string) (event.Subscription, chan
    ↪  *dpcontract.G4dpLogNotifyDataSent, error) {
838         g4dpContract, err := dpcontract.NewG4dp(common.HexToAddress(contractAddress),
        ↪  client)
839         if err != nil {
840                 log.Println("error: could not instantiate contract")
841                 return nil, nil, err
842         }
843
844         opts := bind.WatchOpts{
```

```
845                 Start:   nil,
846                 Context: nil,
847         }
848
849         sink := make(chan *dpcontract.G4dpLogNotifyDataSent)
850         subscription, err := g4dpContract.WatchLogNotifyDataSent(&opts, sink,
            ↪  []string{dataRequesterId})
851         if err != nil {
852                 log.Println("error: could not subscribe to data shared events")
853                 return nil, nil, err
854         }
855
856         return subscription, sink, nil
857 }
858
859 func ListenForDeleteFromIPFSEvents(dataControllerId string, client *ethclient.Client,
    ↪  contractAddress string) (event.Subscription, chan
    ↪  *dpcontract.G4dpLogNotifyDeleteDataFromIPFS, error) {
860         g4dpContract, err := dpcontract.NewG4dp(common.HexToAddress(contractAddress),
            ↪  client)
861         if err != nil {
862                 log.Println("error: could not instantiate contract")
863                 return nil, nil, err
864         }
865
866         opts := bind.WatchOpts{
867                 Start:   nil,
868                 Context: nil,
869         }
870
871         sink := make(chan *dpcontract.G4dpLogNotifyDeleteDataFromIPFS)
872         subscription, err := g4dpContract.WatchLogNotifyDeleteDataFromIPFS(&opts, sink,
            ↪  []string{dataControllerId})
873         if err != nil {
874                 log.Println("error: could not subscribe to data shared events")
875                 return nil, nil, err
876         }
877
878         return subscription, sink, nil
879 }
880
881 func AddControllerToGroup(_dataRequesterId string, _group uint8, ethConn
    ↪  *EthereumConnection) error {
882         requesterId, err :=
            ↪  dpcrypto.EncryptFpeFF1([]byte(dputil.SpacePad(_dataRequesterId, 16)),
            ↪  []byte(dpconfig.GatewaySymmetricKey))
883         if err != nil {
884                 log.Println("error")
```

```
885                 log.Fatalln(err)
886         }
887
888         mxNonce.mutex.Lock()
889         session, err := newG4dpSession(ethConn)
890         if err != nil {
891                 log.Println("error")
892                 return err
893         }
894         tx, err := session.AddControllerToGroup([requesterIdSize]byte(requesterId),
        ↪  _group)
895         mxNonce.mutex.Unlock()
896         if err != nil {
897                 log.Println("error")
898                 return err
899         }
900
901         if dpconfig.WaitTxFinish {
902                 recp, err := bind.WaitMined(context.Background(), ethConn.client, tx)
903                 if err != nil {
904                         log.Println("error")
905                         return err
906                 }
907
908                 // a transação falhou. Nesta versão, não vamos reenviá-la, isso ficará
                ↪  a cargo do cliente, enquanto for síncrono (ou seja, tiver wait).
909                 if recp.Status == 0 {
910                         log.Println("error")
911                         return fmt.Errorf("transaction failed")
912                 }
913
914                 log.Println("AddControllerToGroup gas usage: " + fmt.Sprintf("%d",
                ↪  recp.GasUsed))
915
916                 return nil
917         } else {
918                 return nil
919         }
920 }
921
922
```