

UNIVERSIDADE FEDERAL DO PAMPA

Rodrigo Bisso Machado

**SSPD-LGPD: uma Solução para Segurança
e Privacidade de Dados no cenário da Lei
Geral de Proteção de Dados**

Alegrete
2019

Rodrigo Bisso Machado

**SSPD-LGPD: uma Solução para Segurança e
Privacidade de Dados no cenário da Lei Geral de
Proteção de Dados**

Trabalho de Conclusão de Curso apresentado
ao Curso de Graduação em Ciência da Com-
putação da Universidade Federal do Pampa
como requisito parcial para a obtenção do tí-
tulo de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Diego Kreutz

Coorientador: Giulliano Paz

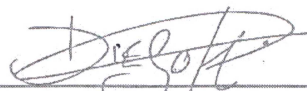
Alegrete
2019

Rodrigo Bisso Machado

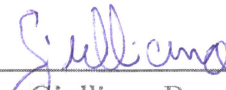
**SSPD-LGPD: uma Solução para Segurança e
Privacidade de Dados no cenário da Lei Geral de
Proteção de Dados**

Trabalho de Conclusão de Curso apresentado
ao Curso de Graduação em Ciência da Com-
putação da Universidade Federal do Pampa
como requisito parcial para a obtenção do tí-
tulo de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em 28. de novembro de 2019
Banca examinadora:



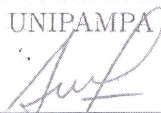
Prof. Me. Diego Kreutz
Orientador
UNIPAMPA



Giulliano Paz
Coorientador
UNIPAMPA



Prof. Dra. Andrea Sabedra Bordin
UNIPAMPA



Prof. Dr. Érico Marcelo Hoff do Amaral
UNIPAMPA

RESUMO

Notícias e relatórios de segurança sobre vazamentos de dados sensíveis e privados têm surgido com uma frequência cada vez maior. O volume e a criticidade dos vazamentos é algo que chega a ser assustador. Este trabalho tem como objetivo contribuir com o desenvolvimento de plataformas online (*e.g.*, EatStreet, Inova Digital) mais seguras através de uma solução denominada SSPD-LGDP, que é constituída de dois protocolos, um para autorização e outro para revogação de acesso aos dados. A SSPD-LGDP leva em consideração os requisitos de novas leis de proteção de dados de usuários, como a LGPD. Os protocolos da solução proposta e do protótipo implementado foram verificados formalmente utilizando a ferramenta Scyther, que não acusou nenhum erro de funcionamento em termos de garantias relacionadas à propriedades essenciais de segurança, como a confidencialidade dos dados.

ABSTRACT

Security news and reports about sensitive and private data leaks have been popping up with increasing frequency. The volume and criticality of the leaks is frightening. This work aims to contribute to the development of safer online platforms (textit eg, EatStreet, Inova Digital) through a solution called SSPD-LGDP, which consists of two protocols, one for authorization and one for revocation of access to data. SSPD-LGDP takes into account the requirements of new user data protection laws, such as LGPD. The protocols of the proposed solution and the implemented prototype were formally verified using the Scyther tool, which found no malfunctions in terms of guarantees related to essential security properties, such as data confidentiality.

LISTA DE FIGURAS

Figura 1 – Arquitetura do serviço EatStreet	16
Figura 2 – Arquitetura Cliente-Aplicação-Banco de Dados Cifrado	19
Figura 3 – Cliente-Aplicação-Banco de Dados utilizando Intel SGX	23
Figura 4 – Arquitetura com base na solução SSPD-LGPD	27
Figura 5 – Diagrama de sequência do protocolo de geração da chave do usuário . .	29
Figura 6 – Diagrama de sequência do protocolo de geração das chaves da empresa	29
Figura 7 – Diagrama de sequência do protocolo de compartilhamento de dados . .	30
Figura 8 – Diagrama de sequência do protocolo de substituição de chaves	32
Figura 9 – Diagrama de Sequência da geração das chaves da empresa utilizando GPG	33
Figura 10 – Diagrama de Sequência da versão utilizando GPG do protocolo	34
Figura 11 – Resultado da verificação formal do Protocolo 3	38
Figura 12 – Resultado da verificação formal do Protocolo 8	39
Figura 13 – Resultado da verificação formal do Protocolo 10	39

LISTA DE TABELAS

Tabela 1 – Características dos SGBDs criptográficos	20
Tabela 2 – Características dos sistemas baseados em Intel SGX	24
Tabela 3 – Quadro-resumo do estado da arte e da solução proposta	25
Tabela 1 – Geração da chave do usuário e registro no serviço	29
Tabela 2 – Geração de chaves da empresa e registro no serviço	30
Tabela 3 – Compartilhamento de dados sensíveis do usuário	31
Tabela 4 – Substituição de chaves do Usuário	31
Tabela 5 – Geração das chaves da empresa utilizando GPG	33
Tabela 6 – Protocolo de geração da chave da empresa utilizando GPG	33
Tabela 7 – Confidencialidade de dados utilizando GPG	34

SUMÁRIO

1	INTRODUÇÃO	15
2	ESTADO DA ARTE	19
2.1	Soluções com suporte nativa à criptografia	19
2.2	Soluções com segurança assistida por hardware	23
2.3	Quadro-resumo das soluções existentes	25
3	A SOLUÇÃO SSPD-LGPD	27
3.1	Requisitos	27
3.2	Protocolos de geração de chaves	28
3.3	Protocolo de compartilhamento de dados	30
3.4	Protocolo de substituição de chaves	31
4	IMPLEMENTAÇÃO	33
5	VERIFICAÇÃO FORMAL DOS PROTOCOLOS	37
6	CONCLUSÃO	41
7	DISCUSSÃO	43
7.1	As Leis sobre Proteção de Dados	43
7.2	Solução SSPD-LGPD	44
	REFERÊNCIAS	45
8	APENDICE	49

1 INTRODUÇÃO

Notícias e relatórios sobre vazamentos de dados sensíveis e privados têm surgido com uma frequência cada vez maior. As estatísticas mostram um volume assustador de dados vazados nos últimos anos (MACHADO et al., 2019). Anualmente, de 2014 a 2019, há casos registrados de vários milhões de registros vazados. Em 2018, um único vazamento de dados na Índia comprometeu os dados de mais de um bilhão de registros.

Além do número e da frequência dos vazamentos, outro aspecto a ser observado é o impacto dos dados vazados, como o re-projeto de dezenas (e até centenas) de sistemas (NG, 2019; MACHADO et al., 2019). O cenário de insegurança cibernética tornou-se tão crítico ao ponto de segurança da informação virar uma prioridade de estado. Apesar de tardiamente, governos em diferentes partes do mundo estão propondo leis e mecanismos com o intuito de tentar reverter a situação atual.

Na União Européia, entrou em vigor a *General Data Protection Regulation* (GDPR)¹ em 2016. No Brasil, a Lei Geral de Proteção de Dados (LGPD)² foi aprovada em 2018. Em ambas as regulamentações, o foco é em devolver aos usuários o controle sobre seus dados e obrigar as empresas a protegerem os dados armazenados. As multas às instituições públicas e privadas que fizerem mau uso dos dados dos usuários (*e.g.*, vazamento de dados) podem chegar a €20 milhões ou 4% do faturamento anual da empresa no caso da União Europeia e R\$50 milhões ou 2% do faturamento anual da empresa no Brasil.

No setor alimentício, um dos casos destaque de 2019 foi o vazamento de dados do aplicativo de pedidos EatStreet, que comprometeu dados pessoais e de pagamento de seus clientes (CIMPANU, 2019b; MACHADO et al., 2019). Segundo informações da página Web oficial, o aplicativo possui parceria com mais de 15 mil restaurantes em 250 cidades. O atacante conseguiu acesso não autorizado ao banco de dados em 13 de Maio de 2019, o que permitiu realizar o *download* dos dados do aplicativo (DEHAVEN, 2019). O invasor só foi descoberto e removido da rede da empresa no dia 17 de Maio. Segundo estimativas, dados de aproximadamente 6 milhões de usuários foram vazados.

A Figura 1 ilustra a arquitetura do EatStreet. Como pode ser observado, o EatStreet é um serviço que opera com dois tipos de clientes (usuários e empresas) e três tipos de atores: usuários, restaurantes e prestadores de serviço de entrega (*delivery*). Os usuários realizam pedidos online aos restaurantes. Para realizar a entrega, os restaurantes utilizam o serviço de *delivery* (terceirizado). O aplicativo, a aplicação e o banco de dados estão em azul pelo fato de trabalharem com (ou armazenarem) dados em texto plano, ou seja, sem nenhum tipo especial de criptografia ou proteção.

Outros exemplos de serviços que funcionam de forma similar ao EatStreet são o

¹ <<http://bit.ly/EUR-GDPR>>

² <<http://bit.ly/Planalto-LGPD>>

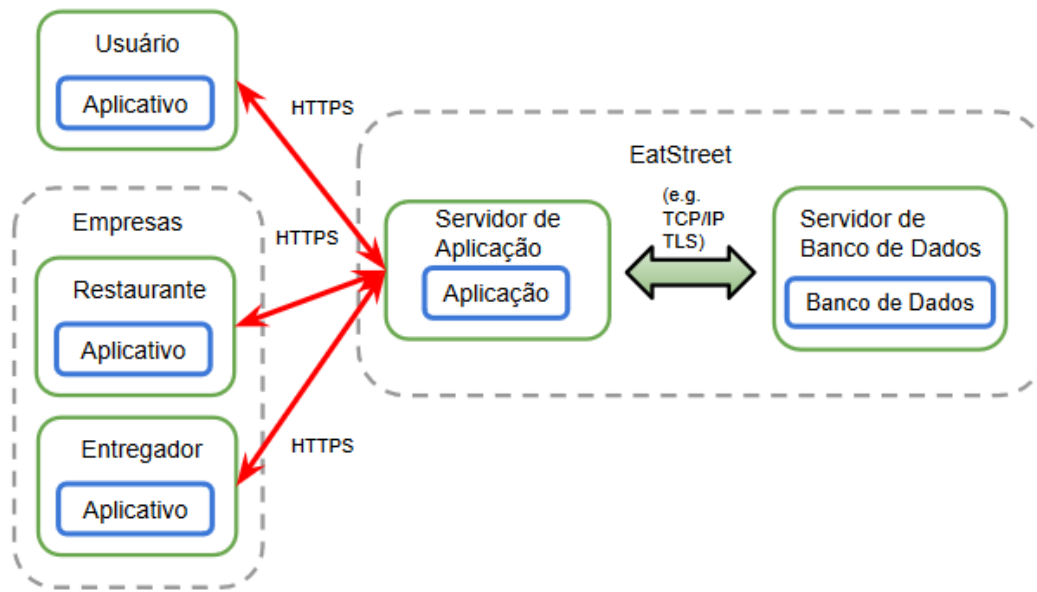


Figura 1 – Arquitetura do serviço EatStreet

Mercado Livre³, a Amazon⁴, o AliExpress⁵, e o Inova Digital (SATHELER et al., 2019). Assim como o EatStreet, estes serviços funcionam como intermediários entre os usuários e as empresas. Na absoluta maioria destas plataformas online, os dados de milhões de usuários e milhares de empresas são armazenados de forma centralizada e desprotegida (*e.g.*, sem utilização de criptografia). Por um lado, isto facilita as operações sobre os dados, como as análise de dados e a utilização de algoritmos de aprendizagem de máquina para determinar o perfil dos usuários. Por outro lado, isto facilita muito o trabalho dos hackers, ou seja, aumenta os riscos de segurança, potencializando vazamentos que comprometem dados sensíveis dos usuários.

O objeto de estudo deste trabalho é a plataforma Inova Digital (SATHELER et al., 2019). Os principais produtos da Inova Digital são documentos digitais de identificação e marketing direcionado para empresas. Como as empresas interagem com os seus clientes (*i.e.*, usuários, associados, fidelizados) através dos serviços oferecidos pela plataforma, o principal desafio da Inova Digital é proteger a confidencialidade e a privacidade dos dados dos clientes e das empresas. Além dos aspectos de segurança e privacidade, para atender ao aspecto de propriedade de dados determinado pela LGPD, a plataforma precisa oferecer também mecanismos de controle que permitem ao usuário determinar quem irá (e por quanto tempo) ter acesso aos seus dados. Buscando atender os aspectos legais e técnicos da LGPD, a pergunta que surge é: Como garantir a confidencialidade, a privacidade e o direito de acesso aos dados dos usuários e empresas em plataformas como a Inova Digital?

³ <<https://mercadolivre.com.br>>

⁴ <<https://www.amazon.com/>>

⁵ <<https://pt.aliexpress.com/>>

Levando em conta leis como a GDPR e a LGPD, os principais desafios de segurança e privacidade de dados de arquiteturas como a da EatStreet e Inova Digital são: (i) os dados são armazenados na memória RAM do servidor de aplicação; (ii) os dados são armazenados no banco de dados sem proteção (*e.g.*, cifragem para garantir confidencialidade); (iii) os usuários não tem controle sobre os seus dados, ou seja, não são eles que determinam quem irá ter acesso aos seus dados; e (iv) os usuários não possuem mecanismos para apagar os seus dados da plataforma. Neste contexto, há dois tipos de agentes maliciosos que podem explorar as vulnerabilidades dos sistemas, levando a vazamentos de dados sensíveis. Primeiro, um agente malicioso interno, isto é, alguém com acesso à infraestrutura física, onde os servidores estão instalados, ou mesmo o próprio administrador do servidor. Como constatado por relatórios de segurança, aproximadamente 50% dos incidentes de segurança são causados por agentes internos (SECURITY, 2019), como foi o caso recente da empresa Disjardins (MONTPETIT, 2019). Além disso, um administrador do sistema inexperiente pode comprometer os dados de forma involuntária, como tem ocorrido com alguma frequência na prática (NG, 2019). Outro exemplo disto é o caso do Banco Pan, no qual mais de 250GB de dados foram vazados (SOUZA, 2019a). Um administrador malicioso pode, também, facilmente comprometer os dados tanto no servidor de aplicação quanto no banco de dados.

O segundo caso é um agente malicioso externo. Esses agentes podem atacar os sistemas de diferentes formas. O caso mais comum é explorar vulnerabilidades dos servidores de aplicação, como XSS, *SQL Injection*, *Broken Authentication* e *Buffer Overflow* (OWASP, 2017), ou vulnerabilidades de aplicativos de dispositivos móveis (SOUNTHIRARAJ et al., 2014). Ao encontrar e explorar uma vulnerabilidade, o atacante pode obter acesso ao servidor de aplicação (ou mesmo o servidor do banco de dados) e provocar um vazamento de dados. Como os dados são tratados em memória no servidor de aplicação, sem nenhum tipo de proteção especial, basta o atacante realizar algum ataque de *dump* de memória ou conseguir acesso de super usuário na máquina (LI et al., 2019). Neste último caso, o atacante consegue interceptar todas as requisições diretamente na aplicação, sem precisar realizar um *dump* de memória, por exemplo.

Como forma de contribuir como o desenvolvimento de plataformas digitais, como a Inova Digital, que atendam os requisitos de novas leis como a LGPD, os objetivos deste trabalho são: (a) investigar tecnologias e soluções de sistemas de bancos de dados e mecanismos que podem ser acoplados à arquitetura do software de forma a oferecer segurança aos dados armazenados; e (b) propor uma solução para garantir a confidencialidade e a privacidade dos dados dos usuários na plataforma Inova Digital. A solução proposta, denominada SSPD-LGPD (**S**olução para **S**egurança e **P**rivacidade de **D**ados no cenário da **LGPD**), é constituída de dois protocolos: (i) autorização de acesso aos dados; e (ii) revogação de acesso aos dados.

As principais contribuições deste trabalho são:

- (c_1) Levantamento do estado da arte de banco de dados com suporte nativo a funções criptográficas;
- (c_2) Levantamento e análise do estado da arte de soluções de segurança assistidas por hardware, com atenção especial para a tecnologia Intel SGX (COSTAN; DEVADAS, 2016) (detalhes sobre a tecnologia podem ser vistos no Apêndice 8);
- (c_3) Proposta da solução SSPD-LGPD para plataformas digitais online como EatStreet, Mercado Livre e Inova Digital;
- (c_4) Especificação formal dos protocolos de autorização e revogação de acesso aos dados dos usuários;
- (c_5) Verificação formal automática dos protocolos propostos utilizando a ferramenta Scyther.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta o estado da arte. Nas Seções 3 e 4 são discutidas a arquitetura da SSPD-LGPD e um protótipo implementado, respectivamente. A verificação formal da solução proposta e do protótipo implementado é apresentada na Seção 5. Finalmente, a Seção 6 apresenta a conclusão.

2 ESTADO DA ARTE

Esta seção apresenta duas categorias de soluções existentes no que diz respeito a confidencialidade e privacidade de dados. Primeiro, na Seção 2.1 são apresentadas as soluções com suporte nativa à criptografia. Em seguida, na Seção 2.2 são discutidas as soluções de segurança assistida por hardware que utilizam a tecnologia Intel SGX.

2.1 Soluções com suporte nativa à criptografia

Para aumentar a segurança no armazenamento de dados, empresas e pesquisadores têm investido no desenvolvimento de bancos de dados com funções criptográficas, como é o caso do CryptDB (POPA et al., 2011), ZeroDB (EGOROV; WILKISON, 2016) e SQL-Cipher (ZETETIC, 2015). A principal ideia é o banco de dados armazenar as informações de forma cifrada, como ilustrado na arquitetura da Figura 2. Como os dados vem cifrados do servidor de aplicação, o agente malicioso, que tem acesso ou compromete o servidor do banco de dados, passa a ter acesso apenas aos dados cifrados. Isto, naturalmente, elimina parte dos riscos de vazamento de dados sensíveis (em texto plano) encontrados na maioria dos sistemas, que ainda seguem a arquitetura apresentada e discutida anteriormente na Figura 1.

Na arquitetura apresentada na Figura 2, o agente malicioso consegue ter acesso aos dados decifrados somente se comprometer, também, o proxy ou o servidor de aplicação. Isto porque as chaves utilizadas para cifrar e decifrar os dados ficam armazenadas no proxy, como é o caso do CryptoDB. Neste exemplo, o proxy é responsável por cifrar e decifrar, de forma transparente, os dados entre o servidor de aplicação e o banco de dados. Entretanto, os dados continuam vulneráveis a ataques no servidor de aplicação.



Figura 2 – Arquitetura Cliente-Aplicação-Banco de Dados Cifrado

O novo problema, que surge na arquitetura apresentada na Figura 2, é o gerenciamento das chaves criptográficas. Como forma de incrementar a robustez e segurança do sistemas, algumas soluções (e.g. BigQuery e Always Encrypted) adotam sistemas especializados no gerenciamento de chaves (KMS) (TURNER, 2019). A principal função de um KMS é gerenciar armazenamento, troca, utilização, destruição e substituição de chaves em um ambiente criptográfico. Em outras palavras, com um KMS ativo no sistema, as

chaves não são mais armazenadas no proxy, servidor de aplicação ou de banco de dados. A principal vantagem é o fato de o KMS ser um sistema especializado, robusto, preparado para assegurar a segurança das chaves criptográficas frente a diferentes tipos de ataques.

O risco de vazamento de dados no servidor de aplicação, onde os dados são processados em memória, pode ser mitigado através de criptografia homomórfica (GENTRY et al., 2009). A criptografia homomórfica permite a execução de instruções, como consultas SQL, sobre dados cifrados. Neste caso, apenas dados cifrados ficam armazenados na memória do servidor de aplicação, ou seja, a confidencialidade dos dados sensíveis é mantida também nessa parte da arquitetura do sistema. Entretanto, há pelo menos dois problemas. Primeiro, o conjunto de instruções é limitado e pode exigir a adaptação das aplicações. Segundo, o alto custo computacional para o processamento de consultas sobre dados cifrados, tornando a utilização inviável para a maioria dos sistemas atuais (NAEHRIG; LAUTER; VAIKUNTANATHAN, 2011). Por exemplo, uma consulta na máquina de buscas da Google pode demorar até um trilhão de vezes mais utilizando criptografia homomórfica (COONEY, 2009; GREENBERG, 2011). Uma forma de amenizar o problema do custo computacional é através da utilização de criptografia parcialmente homomórfica (MORRIS, 2013). Entretanto, isto ainda não irá resolver o problema de desempenho e, além disso, irá restringir ainda mais o número de operações que podem ser executadas sobre os dados cifrados.

Os sistemas CryptDB, ZeroDB, Always Encrypted (MICROSOFT, 2017), BigQuery (GOOGLE, 2017) e SQLCipher são exemplos de Sistemas Gerenciadores de Bancos de Dados (SGBDs) que oferecem funções criptográficas para proteção de dados. A Tabela 1 resume algumas das principais características destes sistemas.

Característica	CryptDB	ZeroDB	Always Encrypted	BigQuery	SQLCipher
KMS			Armazena chaves de cifra	Cifra e decifra os dados	
Local de cifragem	Proxy	Aplicação	Aplicação	KMS	Banco de Dados
Cifra de dados	Todo dado armazenado é cifrado	Todo dado armazenado é cifrado	Administrador decide quais itens cifrar	Administrador decide quais itens cifrar	Todo dado armazenado é cifrado
Tipo de criptografia	AES-256 e Cripto. Homomórfica	AES-256 em modo GCM	AES-256 em modo CBC	AES-256	AES-256 em modo CBC e PBKDF2
Continuidade do sistema	Descontinuado	Descontinuado	Mantido (Microsoft)	Mantido (Google)	Mantido (Zetetic)

Tabela 1 – Características dos SGBDs criptográficos

KMS. Tanto o Always Encrypted quanto o BigQuery utilizam um KMS para o gerenciamento de chaves de cifra de dados. No caso do Always Encrypted, a documentação recomenda a utilização do Azure Key Vault¹, outro serviço da Microsoft. Para utilizar o Azure Key Vault, a aplicação utiliza um driver que envia metadados da chave para o KMS. A partir dos metadados, o KMS constrói a chave e envia para a aplicação. De forma a evitar o acesso constante ao KMS, que pode causar perda de desempenho, a chave é mantida em cache pelo driver.

No BigQuery os dados armazenados são mantidos em blocos distintos. Uma chave única é utilizada para cifrar cada bloco. Em caso de atualização dos dados, a chave também é atualizada. Cada bloco é distribuído nos sistemas da Google e é replicado em formato criptográfico para backup. O sistema de backup cifra cada arquivo com uma chave própria denominada DEK, derivada de uma chave armazenada no KMS da Google. As DEK são cifradas por chaves denominadas KEK, que são armazenadas em um repositório separado, disponibilizado pelo KMS. Quando o acesso ao dado é solicitado, a DEK (cifrada) é recuperada e enviada para o KMS.

Local de cifragem. No CryptDB as chaves de cifra ficam armazenadas no proxy, cuja função é cifrar e decifrar os dados dos usuários. No ZeroDB a cifragem dos dados ocorre na aplicação, que pode estar no servidor de aplicação ou no cliente. Se a aplicação está no servidor de aplicação, então as chaves de cifra de dados ficam expostas a agentes maliciosos que venham a comprometer o servidor.

No segundo caso, a aplicação e as chaves criptográficas ficam com o usuário (e.g. no seu dispositivo móvel). Podemos dizer que, neste caso, a criptografia e a segurança dos dados (confidencialidade) passa a ser do tipo fim-a-fim. A atuação do servidor de aplicação fica (essencialmente) reduzida ao gerenciamento das requisições entre os usuários e o banco de dados.

Apesar de trazer mais segurança à arquitetura, esta alternativa é ainda pouco utilizada pelo fato de exigir mais recursos computacionais e largura de banda no lado do cliente, uma vez que as regras de negócio passam a estar no lado do cliente. Por outro lado, sob o aspecto de segurança dos dados, assumindo a segurança dos algoritmos criptográficos utilizados na aplicação, a criticidade dos vazamentos de dados, nos servidores de aplicação e banco de dados, passa a ser mínima uma vez que o atacante vai ter acesso aos dados dos usuários somente se comprometer, individualmente, as chaves criptográficas que estão armazenadas nos dispositivos dos usuários.

Cifra de dados. Nos casos do Always Encrypted e BigQuery, fica a cargo dos administradores da aplicação a decisão de quais colunas das tabelas cifrar. O ideal é que todos os dados sejam cifrados, tornando mais difícil a recuperação de dados potencialmente sensíveis por parte dos agentes maliciosos. No caso de existirem conjuntos de dados não cifrados, o que ainda é muito comum na prática, um agente malicioso pode

¹ <<https://azure.microsoft.com/services/key-vault/>>

comprometer um número significativo de informações sensíveis dos usuários. No caso da empresa Blur (CIMPANU, 2019a), foram vazados dados de 2,4 milhões de usuários, incluindo informações de email, nome completo, dicas de senhas, endereços de IP e as senhas cifradas. Apenas as senhas dos usuários não foram expostas diretamente. Esta é a realidade da maioria dos sistemas, ou seja, exceto as senhas, todos os dados são armazenados em texto plano no banco de dados.

Tipo de criptografia. Todos os sistemas utilizam o algoritmo AES (DAEMEN; RIJMEN, 1999) para a cifra dos dados. O modo AES-CBC realiza operações XOR entre o bloco de dados atual e o anterior. Como o primeiro bloco não possui antecessor, é utilizado um vetor de inicialização, formado por valores pseudo-aleatórios. Estas operações são aplicadas a todos os blocos do conjunto de dados. A outra variação de utilização do AES, o modo AES-GCM, além de utilizar a cifra de bloco, utiliza *Galois field multiplication*, permitindo que o processo de cifra possa ser realizado em paralelo (STANDARDS; TECHNOLOGY, 2019).

O CryptDB utiliza também criptografia parcialmente homomórfica, que permite com que algumas das requisições possam ser executadas com os dados cifrados, tendo um custo computacional menor do que as funções completamente homomórficas. Já no caso do SQLCipher, é utilizado também o PBKDF2 para a geração da chave da tabela. O PBKDF2 utiliza uma função pseudo-aleatória para derivar chaves (MORIARTY; KALISKI; RUSCH, 2017). A força da chave derivada depende da função pseudo-aleatória utilizada. Esse algoritmo é utilizado para fortificar as chaves, a fim de tornar elas mais resistentes contra ataques de força bruta.

Continuidade do Sistema. O CryptDB e o ZeroDB foram descontinuados sem motivo aparente. No caso do ZeroDB, no GitHub (<<https://github.com/zerodb/zerodb/issues>>) dos desenvolvedores é informado que os interessados em manter o sistema devem entrar em contato com os desenvolvedores. Segundo os registros do GitHub, apenas um usuário demonstrou interesse em continuar o sistema. Porém, não houve resposta dos desenvolvedores originais do CryptDB. A edição *community* do SQLCipher ainda está ativa e recebe atualizações periódicas. O Always Encrypted e BigQuery são soluções pagas e mantidas pela Microsoft e pela Google, respectivamente.

Um dos principais desafios nos SGBDs criptográficos é o desempenho. Por exemplo, o ZeroDB chega a ser 1000 vezes mais lento que o MySQL na execução de consultas ao banco (MITTERER et al., 2018). O ZeroDB também sofre influência da rede. Resultados experimentais demonstram que uma oscilação de 5% na rede pode dobrar o tempo de execução das requisições do banco de dados. Este pode ser um sério problema para aplicações que utilizam a Internet, uma rede imprevisível e com constantes oscilações de tráfego. Outro exemplo é o CryptDB que, quando comparado com o MySQL, executa mil requisições a menos por segundo.

No que diz respeito à proteção contra agentes maliciosos, nenhum dos sistemas

oferece algum tipo de proteção para os dados no servidor de aplicação. Além disso, apenas o BigQuery não armazena as chaves de cifra no servidor de aplicação ou servidor de banco de dados. Manter as chaves de cifra nestes servidores permite que um agente malicioso recupere essas chaves realizando ataques à memória ou disco e decifre os dados dos usuários.

2.2 Soluções com segurança assistida por hardware

A Figura 3 ilustra uma arquitetura ancorada em segurança assistida por hardware, utilizando tecnologias recentes como Intel SGX, para garantir a confidencialidade dos dados. A ideia desse tipo de solução é utilizar um componente fisicamente isolado do resto do sistema, para que este possa realizar a computação de forma segura sobre os dados. Exemplos de tecnologias que oferecem este tipo de serviço são Intel SGX, TrustZone (DEVELOPER, 2019) e FPGA Security (TRIMBERGER; MOORE, 2014). Neste trabalho iremos considerar apenas soluções baseadas na tecnologia Intel SGX, uma das mais recentes e mais largamente utilizadas para o projeto de sistemas de bancos de dados seguros. Provedores de computação em nuvem, como Google e Amazon, oferecem IaaS (Infrastructure-as-a-Service) com suporte a Intel SGX.

A Figura 3 ilustra a arquitetura baseada em tecnologia como a Intel SGX. Neste caso, diferentemente das arquiteturas apresentadas nas Figuras 1 e 2, tanto os dados no servidor de aplicação quanto os dados no banco de dados são mantidos em compartimentos de memória isolados, isto é, onde nem o administrador do sistema tem acesso. No mundo SGX, estes compartimentos são denominados de enclaves.

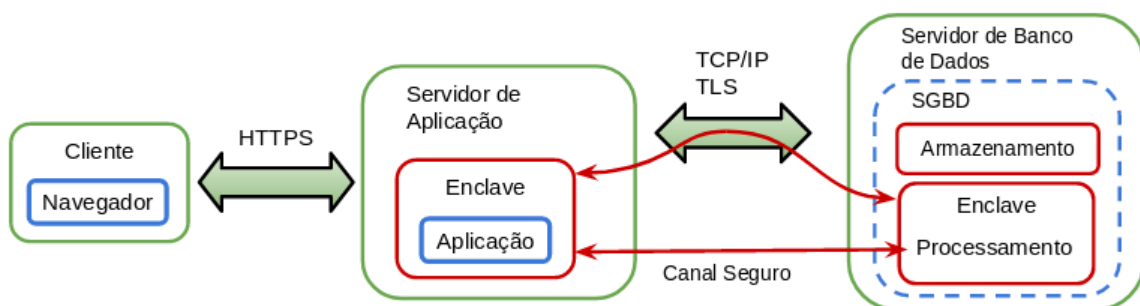


Figura 3 – Cliente-Applicação-Banco de Dados utilizando Intel SGX

O compartilhamento de chaves de cifra entre o servidor de aplicação e o servidor de banco de dados é realizado a partir da criação de um canal de comunicação entre os dois compartimentos seguros do sistema. A própria arquitetura da tecnologia SGX oferece um protocolo de atestação remota, representado pela seta vermelha inferior da Figura 3 para criar canais de comunicação seguros, isto é, sem intervenção externa. As chaves de cifra

são mantidas dentro das regiões de memória protegidas. Ao final da troca de chaves para cifrar/decifrar os dados, os servidores de aplicação e banco de dados começam a operar de forma que as requisições sejam processadas somente dentro dos compartimentos seguros (seta vermelha superior da Figura 3).

CryptSQLite (WANG et al., 2017), EnclaveDB (PRIEBE; VASWANI; COSTA, 2018), STANLite (SARTAKOV et al., 2018) e StealthDB (GRIBOV; VINAYAGAMURTHY; GORBUNOV, 2017) são exemplos de soluções que utilizam a tecnologia Intel SGX para proteção da confidencialidade dos dados. A Tabela 2 ilustra algumas das principais características destas soluções.

	CryptSQLite	EnclaveDB	STANlite	StealthDB
Enclaves	Enclaves do SGBD e Aplicação	Enclaves do SGBD e Aplicação	Enclaves do SGBD e Aplicação	Enclaves de Autenticação, Pré-Processamento e Operação e Enclave de Aplicação
SGBD	SQLite	Hekaton	SQLite	PostgreSQL

Tabela 2 – Características dos sistemas baseados em Intel SGX

Enclaves. Nas soluções CryptSQLite, EnclaveDB e STANlite são utilizados dois enclaves, um no servidor de aplicação e outro no servidor de banco de dados. Cada enclave armazena seu respectivo sistema e executa as respectivas operações das requisições dos usuários. No caso do servidor de banco de dados, o SGBD é dividido em dois, o processamento (ou *Engine*) e o armazenamento (ou *Storage*). O módulo de processamento, responsável por realizar as operações sobre os dados, é carregado para dentro do enclave, enquanto o armazenamento é mantido na região não segura do servidor, contendo somente os dados cifrados. Essa divisão é feita pela limitação de espaço dos enclaves, que possuem espaço utilizável de 90MB (WEICHBRODT; AUBLIN; KAPITZA, 2018).

O StealthDB utiliza um esquema com três enclaves no servidor de banco de dados, Autenticação (*Auth*), Pré-Processamento (*PreProcess*) e Operação (*Ops*) (GRIBOV; VINAYAGAMURTHY; GORBUNOV, 2017). Entre o enclave da aplicação e o enclave *Auth* é realizado o processo de atestação para a troca de chave de cifra de dados. Na sequência, o enclave *Auth* sela a chave e realiza o processo de atestação local com os demais enclaves no servidor de banco de dados. As chaves de cifra são enviadas para os demais enclaves, que também mantêm a chave armazenada usando o protocolo de selamento de dados.

O enclave de *Auth* é responsável pela autenticação do usuário. Caso a autenticação ocorra com sucesso, as consultas do usuário são redirecionadas para o enclave *PreProcess*, que prepara as consultas para o enclave *Ops*, responsável pela execução das consultas. O objetivo do projeto, com três enclaves, é diminuir o tamanho do enclave através do armazenando de menos código e dados dentro de cada enclave. Esta estratégia melhora o

desempenho pelo fato de reduzir a quantidade de trocas de contexto (entrada e saída do enclave) de cada enclave.

SGBD. Como o enclave tem tamanho limitado, são utilizados prioritariamente SGBDs que ocupam pouco espaço em memória. CryptSQLite e STANlite utilizam como base o SQLite, que ocupa menos de 600KB em memória e possui um bom desempenho no processamento das requisições (DRAKE, 2019). Nos casos do EnclaveDB e StealthDB, foram escolhidos o Hekaton e o PostgreSQL, respectivamente, pelos mesmos motivos dos trabalhos anteriores.

Considerando a arquitetura apresentada, podemos observar que os dados dos usuários podem ser interceptados por um agente malicioso antes de serem enviados para o enclave no servidor de aplicação. Além disso, utilizando ataques de cache, é possível recuperar dados do enclave (CHEN et al., 2018). Apesar da evolução da tecnologia Intel SGX, diferentes tipos de ataques avançados ainda são possível (MOGHIMI; EISENBARTH; SUNAR, 2018). O ataque à cache também pode ser realizado no servidor de banco de dados a fim de recuperar as chaves de cifra de dados.

2.3 Quadro-resumo das soluções existentes

A Tabela 2.3 apresenta um quadro-resumo das soluções existentes levando em consideração requisitos técnicos e legais associados a necessidades de plataformas como a EatStreet e Inova Digital e leis recentes como a LGPD. Como pode ser observado, a maioria dos requisitos está diretamente associado às exigências legais da LGPD.

		Requisitos Relacionados a LGPD				Demais Requisitos	
		Propriedade dos dados do usuário	Confidencialidade dos dados	Privacidade dos dados	Dados sensíveis e não sensíveis	Compatibilidade com SGBDs já existentes	Criptografia fim-a-fim
Soluções com suporte nativo à criptografia	CryptDB	Não Atende	Nível 2	Atende Parcialmente	Não Atende	Não Atende	Não Atende
	ZeroDB	Atende Parcialmente	Nível 3	Não Atende	Não Atende	Não Atende	Atende
	Always Encrypted	Não Atende	Nível 2	Não Atende	Atende Parcialmente	Não Atende	Não Atende
	BigQuery	Não Atende	Nível 2	Não Atende	Atende Parcialmente	Não Atende	Não Atende
Soluções com segurança assistida por hardware	SQLCypher	Não Atende	Nível 1	Não Atende	Não Atende	Não Atende	Não Atende
	CryptSQLite	Não Atende	Nível 2	Não Atende	Não Atende	Não Atende(*)	Não Atende
	EnclaveDB	Não Atende	Nível 2	Não Atende	Não Atende	Não Atende(*)	Não Atende
	STANlite	Não Atende	Nível 2	Não Atende	Não Atende	Não Atende(*)	Não Atende
Soluções por hardware	StealthDB	Não Atende	Nível 2	Não Atende	Não Atende	Não Atende(*)	Não Atende
Solução Proposta	SSPD-LGDP	Atende	Nível 4	Atende	Atende	Atende	Atende

Tabela 3 – Quadro-resumo do estado da arte e da solução proposta

O requisito de propriedade dos dados dos usuários diz respeito ao controle de acesso aos dados. Como pode ser observado na tabela, apenas a solução proposta, SSPD-LGDP, atende o requisito, assim como ocorre com outros requisitos. Mais detalhes da solução e como ela atende os requisitos podem ser visto na Seção 3. Apesar do ZeroDB utilizar criptografia fim-a-fim, o que é um requisito técnico para manter os dados sob a propriedade do usuário, este não necessariamente tem controle sobre os seus dados. No caso do ZeroDB, o usuário está a mercê do desenvolvedor do aplicativo ou sistema.

Para classificar as soluções com relação a confidencialidade dos dados, foram definidos quatro níveis:

Nível 1 (N1): os dados são cifrados somente no banco de dados;

Nível 2 (N2): os dados são cifrados no servidor de aplicação e no banco de dados;

Nível 3 (N3): os dados são cifrados na aplicação antes de serem enviados ao banco de dados; e

Nível 4 (N4): os dados são cifrados e decifrados somente na aplicação do usuário.

O ideal é que as soluções estejam todas no nível 4, garantindo o máximo de confidencialidade aos dados do usuário. Entretanto, como pode ser observado, a maioria das soluções está apenas no nível 2, ou seja, os dados são cifrados somente no servidor de aplicação.

Com relação à privacidade dos dados, as soluções existentes não definem regras para compartilhamento dos dados com terceiros. Além disso, não são utilizadas funções de hashes criptográficas para dados de identificação do usuário.

Com relação à separação de dados sensíveis e não sensíveis, somente o Always Encrypted e o BigQuery permitem que o administrador do sistema selecione quais dados serão cifrados. Entretanto, a cifragem dos dados ocorre no servidor de aplicação e o administrador pode interferir no processo, ou seja, não é controlado pelo usuário final do serviço.

As soluções que utilizam segurança assistida por hardware são propostas pensando em compatibilidade com diferentes sistemas de banco de dados. Porém, como estão atreladas a tecnologias como Intel SGX, existe um fator limitante quanto à escolha do SGBD, que é o tamanho ocupado em memória. Um enclave SGX tem um espaço de memória bastante restrito. As primeiras gerações de máquinas equipadas com a tecnologia SGX possuem um limite de aproximadamente 80MB por enclave. Além disso, na maioria dos casos, existe a necessidade de re-implementação de partes do sistema para suportar outro banco de dados.

3 A SOLUÇÃO SSPD-LGPD

A Figura 4 ilustra a arquitetura da Inova Digital suportada pela solução SSPD-LGPD. Primeiro, os dados sensíveis dos usuários são armazenados de forma cifrada nos servidores de aplicação e banco de dados da Inova Digital. Segundo, o usuário possui controle sobre quem irá acessar os dados. Na prática, isto significa que o usuário irá determinar se a Empresa A, por exemplo, poderá ter (ou não) acesso aos seus dados. Cada empresa é obrigada a realizar uma solicitação explícita de acesso aos dados do usuário.

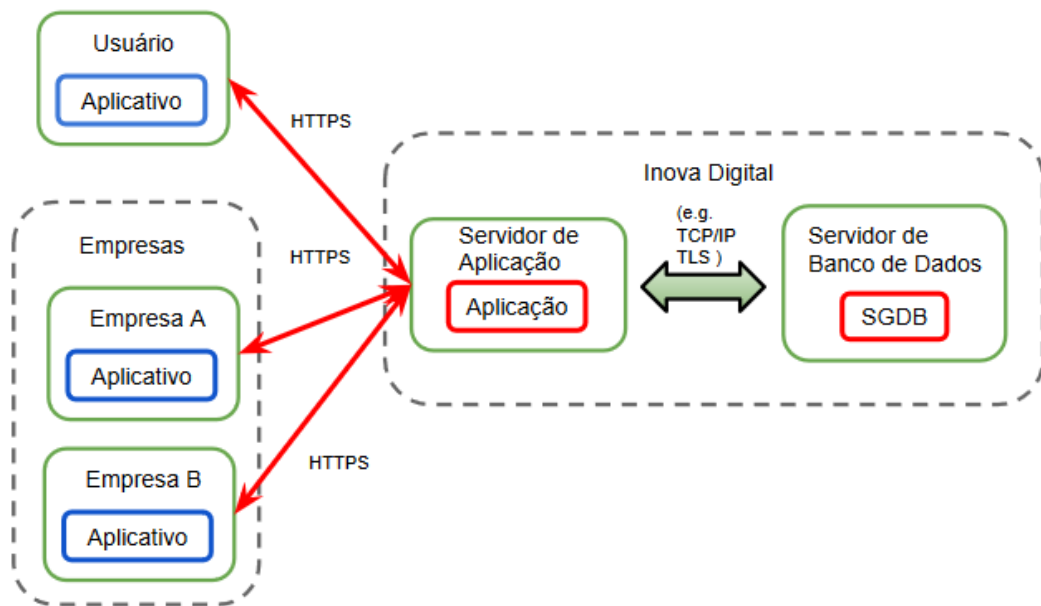


Figura 4 – Arquitetura com base na solução SSPD-LGPD

As próximas seções apresentam os requisitos e os protocolos da SSPD-LGPD. A solução é composta por dois protocolos básicos. O primeiro é o protocolo de autorização de acesso e compartilhamento de dados dos usuários. O segundo protocolo é destinado a remoção de acesso ou simples atualização de chave secreta utilizada para manter a confidencialidade dos dados do usuário.

3.1 Requisitos

Propriedade dos dados do usuário. De acordo com a LGPD, o usuário é quem decide quais empresas e por quanto tempo terão acesso aos seus dados. Isto significa que o usuário pode, a qualquer momento, revogar o acesso aos seus dados a uma empresa X qualquer.

Confidencialidade e privacidade dos dados. Os dados do usuário devem ser armazenados de forma segura, ou seja, sem comprometer a sua confidencialidade e privacidade em caso de algum eventual vazamento de dados. Na prática, isto significa utilizar algo-

ritmos de criptografia e identificadores não rastreáveis. Um exemplo de identificador não rastreável é a hash criptográfica do CPF ao invés do número. As funções hash criptográficas fornecem a propriedade de irreversibilidade por construção, ou seja, a partir da hash criptográfica não é possível chegar-se ao número CPF. Para um usuário externo ou atacante, a hash criptográfica nada mais é que um conjunto de caracteres pseudo-aleatórios sem significado.

Dados sensíveis e não-sensíveis. Dados não-sensíveis são informações que já são públicas ou que não revelam informações sensíveis dos usuários (*e.g.*, números de telefones que constam em guias telefônicos). O servidor de aplicação passa a realizar operações sobre dados considerados não-sensíveis apenas. Isto faz com que o servidor de aplicação deixe de ser um elo fraco de segurança na arquitetura. Já os dados sensíveis são cifrados na origem, no próprio dispositivo do usuário, e armazenados de tal forma no servidor de dados da plataforma. O usuário terá que explicitamente compartilhar a chave de cifra dos dados com todas as empresas que terão acesso aos dados.

Compatibilidade com SGBDs já existentes. Como os dados sensíveis serão cifrados na aplicação do usuário, o SGBD não precisa suportar funções criptográficas, o que não impacta o desempenho do serviço, como ocorre nas arquiteturas discutidas nas Seções 2.1 e 2.2. Para responder às requisições dos clientes, basta o banco de dados deixar identificável os campos nas respectivas tabelas. Com exceção de um dado para identificação do usuário, todas as informações podem ser armazenadas de forma cifrada no banco de dados. Isso permite a utilização de SGBDs existentes, como MySQL, PostgreSQL, IBM DB2, Oracle e SQLite.

Criptografia fim-a-fim. Os dados sensíveis trocados entre usuários e empresas devem atender os requisitos de confidencialidade e privacidade especificados pela LGPD. As informações compartilhadas entre usuários são decifradas somente nas aplicações dos usuários, o que caracteriza uma solução de criptografia fim-a-fim. Para que isso ocorra, é necessário um sistema de chaves compartilhadas ou chave pública/privada. O ideal é que a geração das chaves seja controlada pelos usuários, sem envolver outros componentes do sistema. Para um agente malicioso recuperar as chaves de cifra, ele terá de comprometer os usuários individualmente.

3.2 Protocolos de geração de chaves

O primeiro passo da solução SSPD-LGPD é a geração da chave simétrica do usuário. A Figura 5 e o Protocolo 1 representam o processo geração de sua chave e registro do usuário no serviço.

No Protocolo 1, a aplicação do usuário gera uma chave secreta simétrica SK (linha 1). A geração da chave é realizada utilizando dados únicos do usuário em conjunto com um valor pseudo-aleatório (*e.g.* obtido a partir de uma função como $URANDOM$). Esta chave é utilizada para cifrar (representado pela função E no protocolo) os dados do

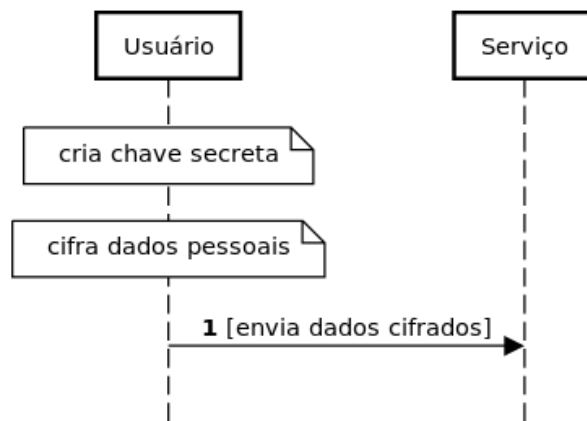


Figura 5 – Diagrama de sequência do protocolo de geração da chave do usuário

usuário. Na linha 2, o usuário envia para o serviço uma mensagem contendo a operação de adicionar o seu registro ao banco de dados (*ADD*), o seu identificador *id_cliente* (e.g., hash criptográfica do CPF), os seus dados cifrados ($E(\text{dados})$), os seus dados públicos ou dados não-sensíveis (*dados_pub*), um *nonce* (para evitar ataques de *replay*) e uma assinatura da mensagem (*HMAC*). O serviço recebe a mensagem, processa e armazena os dados do usuário.

-
- | | |
|----------------------------------|---|
| 1. Usuário | $SK \leftarrow$ Cria chave secreta |
| 2. Usuário \rightarrow Serviço | $[ADD, id_usuario, E(\text{dados}), \text{dados_pub}, \text{nonce}]$, HMAC |
-

Protocolo 1 – Geração da chave do usuário e registro no serviço

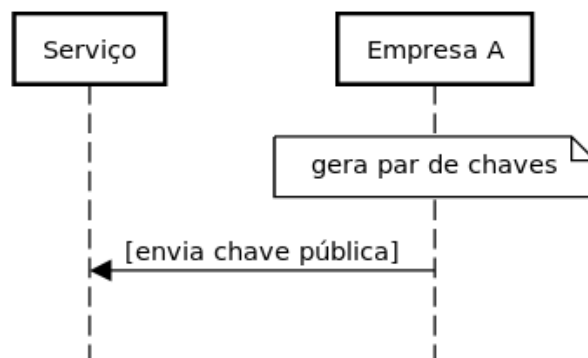


Figura 6 – Diagrama de sequência do protocolo de geração das chaves da empresa

O processo de registro da empresa no serviço é descrito pelo Protocolo 2. O esquema de chaves utilizado pela empresa é diferente do usuário. Aqui são geradas duas chaves, uma pública e outra privada (linha 1). Uma cópia da chave pública (P_k) é enviada para o serviço (linha 2). A mensagem da Empresa contém o comando *POST_KEY*, o

identificador da empresa ($id_empresa$) a chave pública P_k , um *nonce* e uma assinatura *HMAC*.

-
1. Empresa A Gera par de chaves assimétricas
 2. Empresa A → Serviço [*POST_KEY*, $id_empresa$, P_k , *nonce*], HMAC
-

Protocolo 2 – Geração de chaves da empresa e registro no serviço

3.3 Protocolo de compartilhamento de dados

O diagrama de sequência da Figura 7 representa o protocolo de compartilhamento de dados entre usuário e empresa. Este processo é descrito em detalhes pelo Protocolo 3 Para poder visualizar os dados do usuário, a empresa precisa realizar um pedido de acesso aos dados ao serviço (linha 1), que irá notificar o usuário (linha 2). Caso o usuário autorizar a empresa a ter acesso aos seus dados, ele irá requisitar a chave pública P_k da empresa para o serviço (linha 3 e linha 4). A chave secreta SK é cifrada com a chave pública P_k e enviada para o serviço (linha 5). Finalmente, o serviço envia a chave SK e os dados do usuário para a empresa (linhas 6 e 7), respectivamente. As mensagens das linhas 6 e 7 são respostas ao pedido realizado pela empresa da linha 1. Finalmente, a empresa decifra os dados do usuário utilizando a chave SK (linha 8).

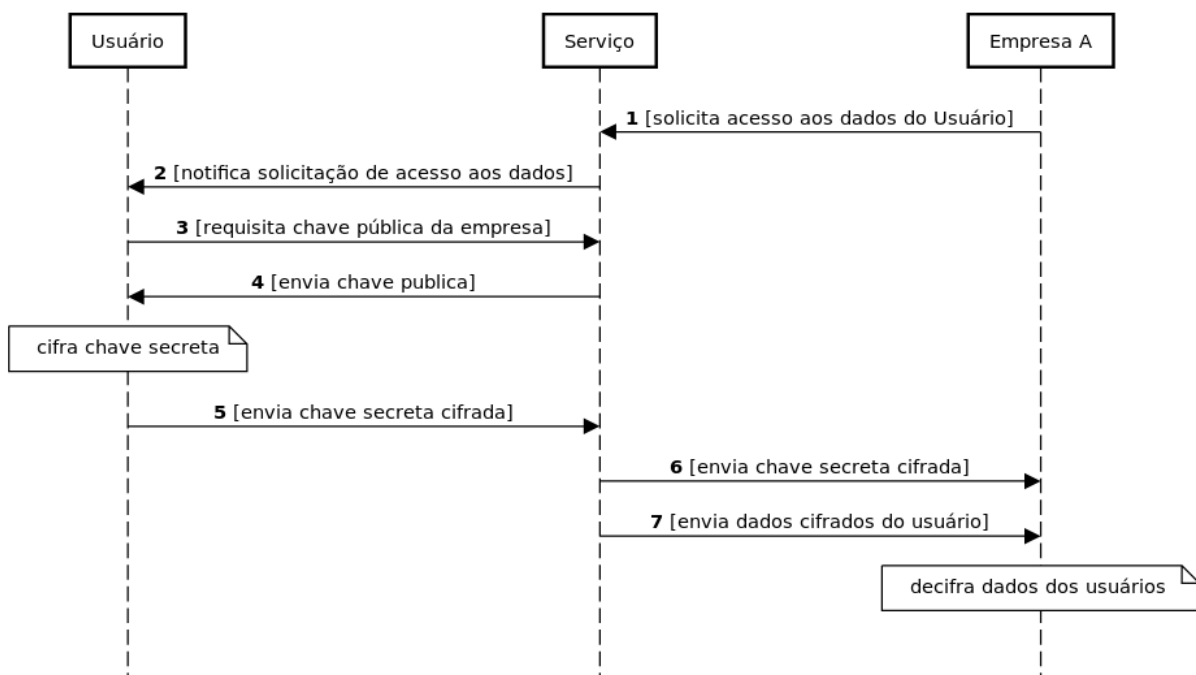


Figura 7 – Diagrama de sequência do protocolo de compartilhamento de dados

1. Empresa A → Serviço	[<i>ACCESS</i> , <i>id_empresa</i> , <i>id_usuario</i> , <i>nonce</i>], HMAC
2. Serviço → Usuário	[<i>NOTIFY</i> , <i>id_servico</i> , <i>id_empresa</i> , <i>nonce</i>], HMAC
3. Usuário → Serviço	[<i>GET_PK</i> , <i>id_usuario</i> , <i>id_empresa</i> , <i>nonce</i>], HMAC
4. Serviço → Usuário	[<i>REPLY</i> , <i>id_servico</i> , P_k , <i>nonce</i>], HMAC
5. Usuário → Serviço	[<i>REPLY</i> , <i>id_usuario</i> , $E_{P_k}(SK)$, <i>nonce</i>], HMAC
6. Serviço → Empresa A	[<i>REPLY</i> , <i>id_servico</i> , $E_{P_k}(SK)$, <i>id_usuario</i> , <i>nonce</i>], HMAC
7. Serviço → Empresa A	[<i>REPLY</i> , <i>id_servico</i> , $E(dados)$, <i>dados_pub</i> , <i>id_usuario</i> , <i>nonce</i>], HMAC
8. Empresa A	Decifra dados do usuário

Protocolo 3 – Compartilhamento de dados sensíveis do usuário

3.4 Protocolo de substituição de chaves

Para atender os requisitos de propriedade e confidencialidade de dados, é necessário também um protocolo de revogação de chaves secretas do usuário. O diagrama de sequência da Figura 8 e o Protocolo 4 detalham o protocolo de substituição de chaves do usuário. No Protocolo 4, o usuário requisita todos os seus dados cifrados do serviço (linhas 1 e 2). Então, o usuário decifra os dados utilizando a chave atual (linha 3), gera uma nova chave secreta (linha 4), re-cifra os dados com a nova chave e envia os dados cifrados para o serviço (linha 5). Além dos dados cifrados, é enviado também uma lista com as empresas autorizadas a ter acesso aos dados do usuário (linha 6). A partir deste momento, para cada empresa na lista, o usuário cifra sua chave secreta com a chave pública da empresa e envia para o serviço (linha 7) e notifica a empresa (linha 8). A empresa então requisita os dados e a nova chave secreta do usuário (linha 9). Por fim, o serviço envia para a empresa a nova chave secreta e os dados do usuário cifrados. A empresa deve atualizar a chave secreta local.

1. Usuário → Serviço	[<i>GET_DADOS</i> , <i>id_usuario</i> , <i>nonce</i>], HMAC
2. Serviço → Usuário	[<i>REPLY</i> <i>id_servico</i> , $E_{SK}(dados)$, <i>nonce</i>], HMAC
3. Usuário	Decifra dados
4. Usuário	$SK \leftarrow$ Gera nova chave secreta
5. Usuário → Serviço	[<i>UPDATE</i> , <i>id_usuario</i> , $E_{SK}(dados)$, <i>dados_pub</i> , <i>nonce</i>], HMAC
6. Usuário → Serviço	[<i>UPDATE</i> , <i>id_usuario</i> , <i>lista_empresas</i> , <i>nonce</i>], HMAC
	{Para cada empresa na lista de autorizadas}
7. Usuário → Serviço	[<i>UPDATE</i> , <i>id_usuario</i> , $E_{P_k}(SK)$, <i>nonce</i>], HMAC
8. Serviço → Empresa A	Notifica alteração da chave secreta
9. Empresa A → Serviço	[<i>REQUEST</i> , <i>id_empresa</i> , <i>id_usuario</i> , <i>nonce</i>], HMAC
10. Serviço → Empresa A	[<i>REPLY</i> , <i>id_servico</i> , <i>id_usuario</i> , $E_{P_k}(SK)$, <i>nonce</i>], HMAC
12. Serviço → Empresa A	[<i>REPLY</i> , <i>id_servico</i> , <i>id_usuario</i> , $E_{SK}(dados)$, <i>nonce</i>], HMAC

Protocolo 4 – Substituição de chaves do Usuário

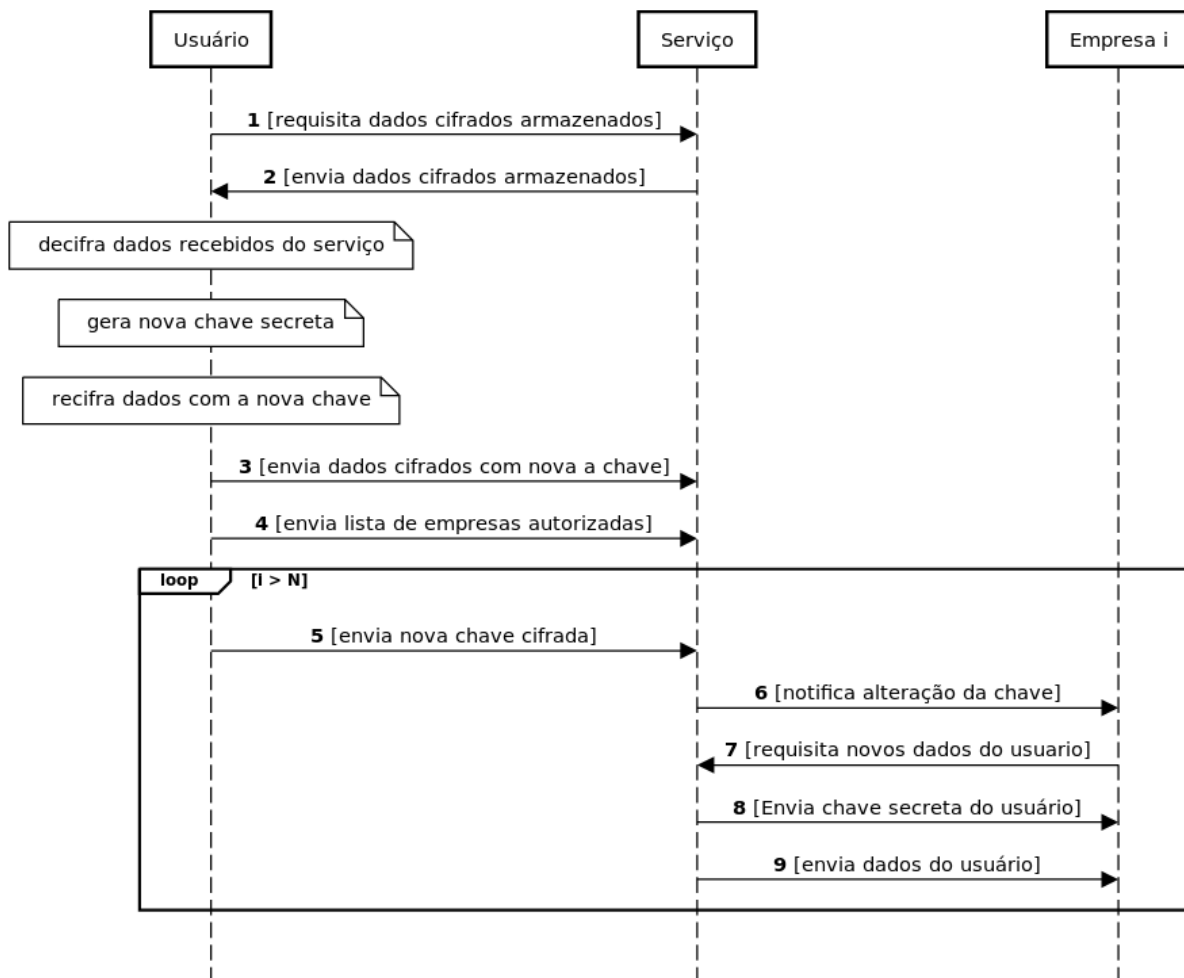


Figura 8 – Diagrama de sequência do protocolo de substituição de chaves

Com o Protocolo 4 o usuário consegue manter o controle sobre quem tem acesso aos seus dados. Ao substituir a chave secreta, o usuário pode revogar o acesso de seus dados para uma ou mais empresas. Este recurso é crucial para atender a requisitos legais como os pontos pelas leis GDPR e LGPD.

4 IMPLEMENTAÇÃO

O diagrama de sequência da Figura 10 e o Protocolo 7 e o representam uma implementação do Protocolo 3 utilizando o serviço de chaves GnuPG (ou GPG), que é uma implementação completa e livre do padrão OpenPGP¹ (também conhecido como PGP). Como pode ser observado, há um serviço adicional, que é o Serviço-GPG. Ao invés de agregar o gerenciamento das chaves no serviço da Inova Digital, as chaves públicas ficam apenas nos repositórios públicos GPG, os quais representam um serviço público já consolidado. Isto reduz a complexidade de implementação e a superfície de ataque do serviço Inova Digital. O Protocolo 6 é uma adaptação do Protocolo 2 para a utilização de chaves GPG. A chave pública gerada é publicada (linha 2) no Serviço-GPG e somente o identificador da chave (*keyid*), é enviado para o Serviço (linha 3).

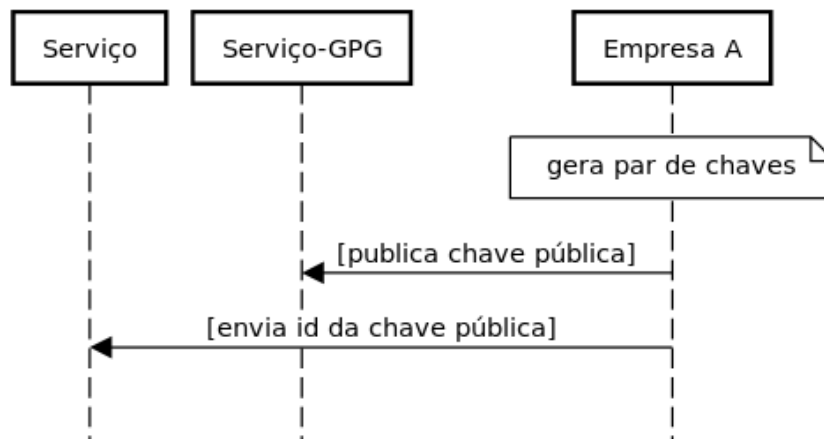


Figura 9 – Diagrama de Sequência da geração das chaves da empresa utilizando GPG

Tabela 5 – Geração das chaves da empresa utilizando GPG

1. Empresa A	Gera par chaves GPG
2. Empresa A → Servidor-GPG	Publica chave pública
5. Empresa A → Serviço	[<i>PK_ID id_empresa, keyid, nonce</i>], HMAC

Tabela 6 – Protocolo de geração da chave da empresa utilizando GPG

Como pode ser observado no Protocolo 7, há poucas diferenças em relação a versão original do Protocolo 3. A separação dos serviços leva a inclusão de duas novas mensagens, nas linhas 5 e 6, respectivamente. Estas mensagens são implementadas utilizando os clientes e protocolos já existentes do serviço de chaves GPG, como o `gpg2` da GnuPG². Após o recebimento da notificação de acesso aos dados, o usuário recupera a P_k da empresa do serviço GPG (linhas 5 e 6).

¹ <<https://tools.ietf.org/html/rfc4880>>

² <<https://gnupg.org/>>

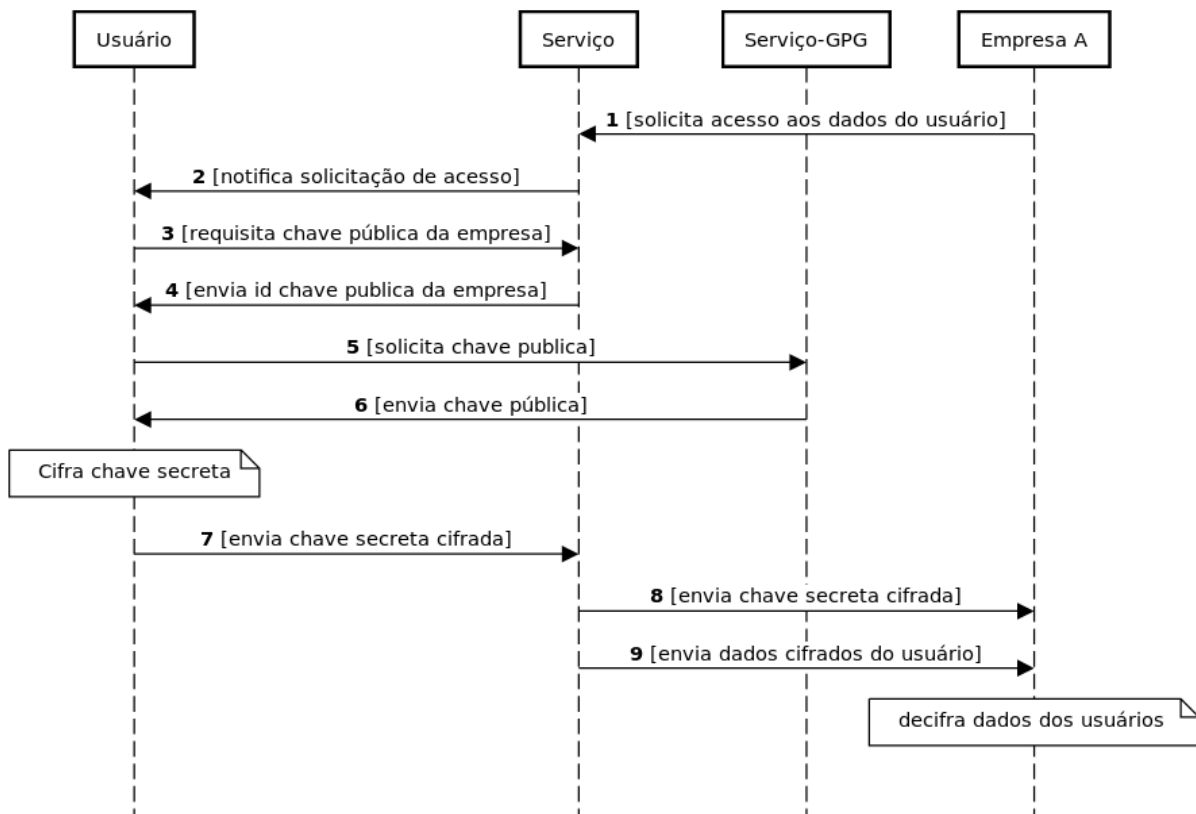


Figura 10 – Diagrama de Sequência da versão utilizando GPG do protocolo

1. Empresa A → Serviço	$[ACCESS, id_empresa, id_usuario, nonce]$, HMAC
2. Serviço → Usuário	$[NOTIFY, id_servico, id_empresa, nonce]$, HMAC
3. Usuário → Serviço	$[GET, id_usuario, id_empresa, nonce]$, HMAC
4. Serviço → Usuário	$[REPLY, id_servico, keyid, nonce]$, HMAC
5. Usuário → Serviço-GPG	Requisita chave pública da empresa
6. Serviço-GPG → Usuário	Envia chave pública
7. Usuário → Serviço	$[REPLY, id_usuario, E_{p_k}(SK), nonce]$, HMAC
8. Serviço → Empresa A	$[REPLY, id_servico, E_{p_k}(SK), id_usuario, nonce]$, HMAC
9. Serviço → Empresa A	$[REPLY, id_servico, E(dados), id_usuario, nonce]$, HMAC
10. Empresa A	decifra dados do usuário

Protocolo 7 – Confidencialidade de dados utilizando GPG

Um protótipo do protocolo foi implementado com a linguagem de programação Python na versão 3.7. Para a comunicação com os servidores do serviço GPG, foi utilizada a biblioteca GnuPG³. Esta biblioteca implementa as funções de cifra da chave secreta e a comunicação com os servidores GPG. Para os testes, foi utilizado o serviço de

³ <<https://docs.red-dove.com/python-gnupg/>>

gerenciamento de chaves da Ubuntu⁴.

⁴ <<https://keyserver.ubuntu.com>>

5 VERIFICAÇÃO FORMAL DOS PROTOCOLOS

Um dos maiores desafios no desenvolvimento de protocolos de segurança é garantir a corretude do protocolo, do projeto até a implementação. O uso de ferramentas especializadas é um passo importante na verificação dos protocolos ainda em processo de projeto (JENUARIO et al., 2019). Neste trabalho, a ferramenta Scyther (CREMERS, 2006) foi utilizada para verificar a corretude dos protocolos propostos e implementados.

A ferramenta Scyther utiliza uma linguagem própria para realizar a verificação automática de protocolos. A seguir, são apresentadas as principais semânticas operacionais da ferramenta.

Termos atômicos. Os termos atômicos servem para definir os dados que serão utilizados no protocolo (e.g. variáveis). Exemplos: (a) *var*: define variáveis para armazenar os dados recebidos de uma comunicação; (b) *const*: define constantes geradas para cada instanciação de uma função; (c) *fresh*: representa variáveis que serão iniciadas com valores pseudo-aleatórios.

Chaves assimétricas. A estrutura de chaves públicas é predefinida pela ferramenta, sendo $sk(X)$ a chave privada e $pk(X)$ a respectiva chave pública de X . A representação da cifra de um dado m utilizando a chave pública é $\{m\}_{pk(X)}$.

Tipos predefinidos. Determinam o comportamento de uma função ou termo. Eis os três principais tipos predefinidos. **Agent**: é utilizado para criar um agente que irá interagir nas comunicações. **Function**: é um tipo de agente que define um termo como sendo uma função. **Nonce**: é considerado o tipo padrão para termos.

Tipos de eventos. Os eventos de envio e recebimento de uma mensagem são definidos como **send** e **recv**, respectivamente. Para cada evento de envio, existe um evento de recebimento correspondente (e.g. **send_1** e **recv_1**)

Eventos de afirmação. O evento **claim** é utilizado em especificações de funções para modelar propriedades de segurança. Na prática, após afirmar que uma variável é secreta, a ferramenta irá verificar se a afirmação procede durante a execução do protocolo, ou seja, verificar se somente as partes interessadas possuem acesso ao dado secreto.

A ferramenta utiliza uma linguagem própria para a realização da verificação. Os Algoritmos 1, 2 e 3 representam os Protocolos 3, 8 e 10, respectivamente. As chaves pk e sk são declaradas globalmente e são atribuídas automaticamente à EmpresaA pela ferramenta. O comando **inversekeys** (linha 3) determina que as chaves pk e sk são inversas. Ao cifrar com a chave pública, só é possível decifrar com a chave privada. A **const HMAC** é definida para ser utilizada como assinatura das mensagens. A chamada para a função **protocol** marca o início da especificação dos protocolos.

As Figuras 11, 12 e 13 apresentam as saídas da verificação formal dos três protocolos. O objetivo da verificação formal era verificar a confidencialidade dos dados armazenados no serviço, ou seja, um atacante que comprometer o serviço não deve ser capaz de decifrar os dados armazenado. Como pode ser observado nas figuras, o *status* **Ok** indica

Algorithm 1 Protocolo 3 na semântica da ferramenta Scyther

```

const pk: Function;
secret sk: Function;
inversekeys (pk,sk);
secret sKey: Function;
const HMAC;
protocol CDC(Usuario,EmpresaA,Servico){

  role Usuario{
var idEmpresa, idServoco : Nonce;
fresh dados, nonce, idUsuario: Nonce;
send_1(Usuario, Servico, HMAC(idUsuario, dadosKey(Usuario), nonce));
recv_5(Servico, Usuario, HMAC(idServico, idEmpresa, nonce));
send_6(Usuario, Servico, HMAC(idUsuario, idEmpresa, nonce));
recv_7(Servico, Usuario, HMAC(idServico, pk(EmpresaA), nonce));
send_8(Usuario, EmpresaA, HMAC(idUsuario, sKeypk(EmpresaA), nonce));
}

  role EmpresaA{
var idCliente, idServico, dados : Nonce
fresh idEmpresa, nonce : Nonce
send_2(EmpresaA, Servico, HMAC(idEmpresa, pk(EmpresaA), nonce));
recv_3(Servico, EmpresaA, HMAC(idServico, idUsuario, nonce));
send_4(EmpresaA, Servico, HMAC(idEmpresa, idUsuario, nonce));
recv_9(Servico, EmpresaA, HMAC(idServico,
sKeypk(EmpresaA), idUsuario, nonce));
recv_10(Servico, EmpresaA, HMAC(idServico, dadosKey(Usuario), idUsuario, nonce));
}

  role Servico{
var dados, nonce, idUsuario : Nonce
fresh idServico : Nonce
recv_1(Usuario, Servico, HMAC(idUsuario, dadosKey(Usuario), nonce));
recv_2(EmpresaA, Servico, HMAC(idEmpresa, pk(EmpresaA), nonce));
send_3(Servico, EmpresaA, HMAC(idServico, idUsuario, nonce));
recv_4(EmpresaA, Servico, HMAC(idEmpresa, idUsuario, nonce));
send_5(Servico, Usuario, HMAC(idServico, idEmpresa, nonce));
recv_6(Usuario, Servico, HMAC(idUsuario, idEmpresa, nonce));
send_7(Servico, Usuario, HMAC(idServico, pk(EmpresaA), nonce));
recv_8(Usuario, Servico, HMAC(idUsuario, sKeypk(EmpresaA), nonce));
send_9(Servico, EmpresaA, HMAC(idServico, sKeypk(EmpresaA), idUsuario, nonce));
send_10(Servico, EmpresaA, HMAC(idServico, dadosKey(Usuario), idUsuario, nonce));
claim(Servico, Secret, dados);
}
}

```

que não há nenhuma falha nos respectivos protocolos.

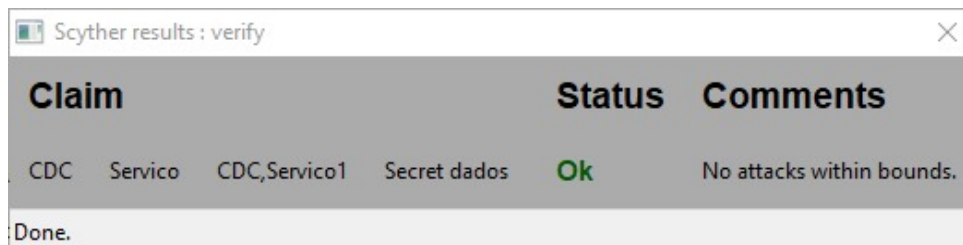


Figura 11 – Resultado da verificação formal do Protocolo 3

Algorithm 2 Protocolo 8 na semântica da ferramenta Scyther

```

const pk: Function;
secret sk: Function;
inversekeys (pk,sk);
const HMAC;
hashfunction H;

protocol SUB(Usuario,EmpresaA,Servico,KUF){
role Kuf{
fresh random: Nonce;
send_3(KUF,Usuario,sKey(h(random)));
}

role Usuario{
var idEmpresa, random, idServocp : Nonce;
fresh dados, nonce, idUsuario: Nonce;
send_1(Usuario, Servico, HMAC(idUsuario,
nonce));
recv_2(Servico, Usuario, HMAC(idServico, da-
dossKey(Usuario), nonce));
recv_3(KUF, Usuario, sKey(H(random)));
send_4(Usuario, Servico, HMAC(idUsuario, da-
dossKeyup, nonce));
}
}

role EmpresaA{
var notify : Nonce
recv_5(Servico, EmpresaA, HMAC(notify));
}

role Servico{
var dados, nonce, idUsuario : Nonce
fresh notify, idServico : Nonce
recv_1(Usuario, Servico, HMAC(idUsuario,
nonce));
send_2(Servico, Usuario, HMAC(idServico, da-
dossKey(Usuario), nonce));
recv_4(Usuario, Servico, HMAC(idUsuario, da-
dossKeyUp, nonce));
send_5(Servico, EmpresaA, HMAC(notify));
claim(Servico, Secret, dados);
}
}

```

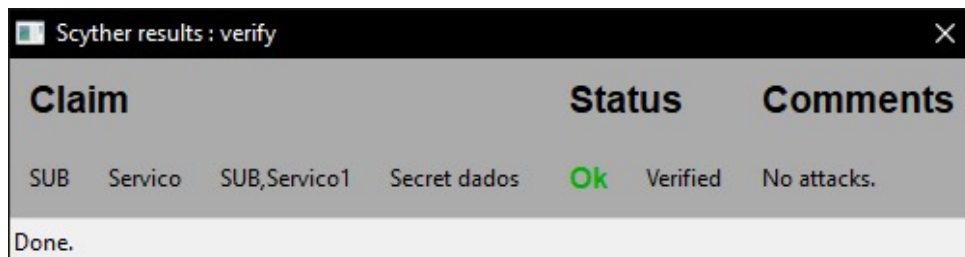


Figura 12 – Resultado da verificação formal do Protocolo 8

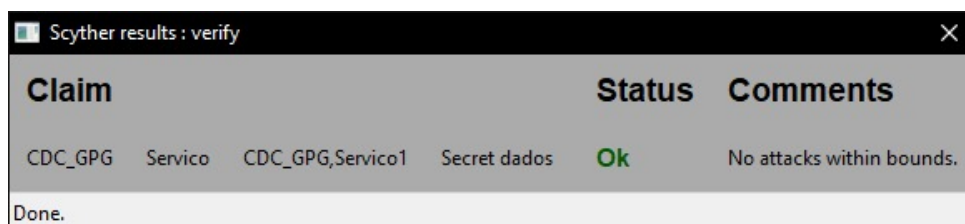


Figura 13 – Resultado da verificação formal do Protocolo 10

Algorithm 3 Protocolo 10 na semântica da ferramenta Scyther

```

const pk: Function;
secret sk: Function;
inversekeys (pk,sk);
secret sKey: Function;
const HMAC;

protocol CDC-GPG(Usuario,EmpresaA,Servico,ServicoGPG){
role ServicoGPG{
recv_2(EmpresaA, ServicoGPG,pk(EmpresaA));
recv_9(Usuario, ServicoGPG, HMAC(pk(EmpresaA)));
send_10(ServicoGPG, Usuario, pk(EmpresaA));
}

    role Usuario{
var idEmpresa, keyID, idServico : Nonce;
fresh dados, nonce, idUsuario: Nonce;
send_1(Usuario, Servico, HMAC(idUsuario, dadosKey(Usuario), nonce));
recv_6(Servico, Usuario, HMAC(idServico, idEmpresa, nonce));
recv_7(Usuario, Servico, HMAC(idUsuario, idEmpresa, nonce));
recv_8(Servico, Usuario, HMAC(idServico, keyID, nonce));
send_9(Usuario, ServicoGPG, HMAC(pk(EmpresaA), dados, nonce));
recv_10(ServicoGPG, Usuario, pk(EmpresaA));
send_11(Usuario, Servico, HMAC(idUsuario, sKeypk(EmpresaA), nonce));
}

    role EmpresaA{
var idCliente, idServico, dados : Nonce
fresh idEmpresa, nonce, keyID : Nonce
send_2(EmpresaA, ServicoGPG, pk(EmpresaA));
send_3(EmpresaA, Servico, HMAC(idEmpresa, keyID, nonce));
recv_4(Servico, EmpresaA, HMAC(idServico, idUsuario, nonce));
send_5(EmpresaA, Servico, HMAC(idEmpresa, idUsuario, nonce));
recv_12(Servico, EmpresaA, HMAC(idServico, sKeypk(EmpresaA), idUsuario, nonce));
recv_13(Servico, EmpresaA, HMAC(idServico, dadossk(Usuario), idUsuario, nonce));
claim(Servico, Secret, dados);
}
}
}

```

6 CONCLUSÃO

Neste trabalho foram analisados os aspectos e requisitos de segurança de plataformas como a EatStreet e a Inova Digital. Durante o processo, foram levados em conta também os requisitos derivados de leis recentes como a LGPD. A partir dos requisitos mapeados, foi proposta uma solução técnica denominada SSPD-LGPD, que é composta essencialmente por dois protocolos, um para a autorização e outro para a revogação de acesso a dados de usuários.

A Solução SSPD-LGPD possibilita aos usuários um controle sobre seus dados pessoais. Os usuários podem conceder e revogar os acessos das empresas de forma fácil e ágil. Os dados dos usuários estão decifrados somente na aplicação do usuário. O serviço passa a armazenar os dados de forma cifrada. Isso exige que os atacantes comprometam os dispositivos dos usuários.

Foi implementado um protótipo da SSPD-LGPD utilizando a linguagem Python (versão 3.7) e a biblioteca GnuPG, que implementa funções do sistema de chaves GnuPG. Os protocolos da solução proposta e da implementação foram verificados formalmente utilizando a ferramenta Scyther. Os resultados da verificação indicam que a solução é robusta o suficiente a ponto de garantir a confidencialidade dos dados de usuários de plataformas online como a EatStreet e a Inova Digital.

O próximo passo (trabalho futuro) é integrar a SSPD-LGPD na plataforma online Inova Digital. Espera-se que este trabalho possa contribuir com a proteção de dados pessoais no contexto atual, ou seja, levando em consideração leis como a LGPD e os constantes ataques cibernéticos que levam a vazamentos de dados cada vez mais frequentes.

7 DISCUSSÃO

7.1 As Leis sobre Proteção de Dados

A União Europeia (EU) pôs em vigor, em 2016, uma nova regulamentação para a proteção de dados pessoais, a *General Data Protection Regulation* 2016/679 (GDPR). A GDPR é um marco legal para a proteção e privacidade de dados de todos os cidadãos da EU e do Espaço Econômico Europeu (EEE), tornando a proteção de dados pessoais um direito fundamental, assim como a liberdade. Inspirada na GDPR, em 2018, foi sancionada a Lei Geral de Proteção de Dados Pessoais (LGPD), nº 13.709, a qual entrará em vigor em agosto de 2020. Tanto a GDPR quanto a LGPD visam proteger e fortalecer a privacidade, dando um maior controle aos cidadãos sobre seus dados pessoais e determinando como devem acontecer a coleta e o tratamento desses dados por terceiros.

Dados pessoais, segundo ambas as leis, são informações que possam identificar, direta ou indiretamente, uma pessoa natural, como CPF, RG e nome completo. Além disso, dados não pessoais como profissão, localização e endereço IP podem se tornar dados pessoais, se utilizados em conjunto com outros dados, para identificar uma pessoa natural. Dados pessoais sensíveis são informações que podem violar a intimidade, honra e imagem das pessoas naturais, como origem racial e étnica, convicções religiosas, políticas e filosóficas, dados genéticos e biométricos e dados referentes à saúde e vida sexual.

Ambas as leis determinam que deverão responder às regulamentações toda e qualquer empresa, pública ou privada, ou pessoa, física ou jurídica, que: armazene ou trate dados pessoais em seu território; a coleta e tratamento de dados tenha como objetivo oferecer ou fornecer serviços em seu território; e colete e manipule dados de seus cidadãos, independente da nacionalidade ou localização das empresas e dados. As multas por violação das regulamentações podem chegar a 20 milhões de euros ou 4% do faturamento anual da pessoa jurídica envolvida, no caso da GDPR, e 2% do faturamento anual da pessoa jurídica ou 50 milhões de reais, pela LGPD.

Ainda no Brasil, além da LGPD, o governo estuda a possibilidade de criação de uma Lei Geral de Segurança Cibernética (LGSC) (SOUZA, 2019b). A lei deve ser apresentada ao Congresso Nacional ainda em 2019. Segundo o coronel Arthur Pereira Sabbat, representante do Gabinete de Segurança Institucional, pelo menos 70 milhões de brasileiros foram vítimas de crimes cibernéticos em 2018. Este é mais um sinal de que a segurança e a privacidade de dados está se tornando uma prioridade de estado.

Tanto a GDPR quanto a LGPD estipulam que a coleta e o tratamento de dados pessoais, sensíveis ou não, se darão apenas mediante autorização explícita do titular dos dados, ou seja, a quem os dados referem-se. Os termos de uso deverão ser sucintos e explícitos, informando com qual finalidade, por quanto tempo e quais empresas e serviços terão acesso aos dados. A autorização de utilização de dados poderá ser cancelada facilmente e a qualquer momento, assim como a modificação e deleção dos dados pessoais. Em suma,

os dados pessoais pertencem única e exclusivamente aos seus titulares, cabendo a estes a decisão de utilização, deleção e comercialização.

7.2 Solução SSPD-LGPD

O foco da solução proposta é adicionar as camadas de propriedade, confidencialidade e privacidade dos dados dos usuários em aplicações do mesmo domínio que Inova Digital e EatStreet, seguindo o que é determinado pela LGPD. A solução é projetada para ser compatível com qualquer sistema de banco de dados. Isto permite maior flexibilidade na utilização da solução. O mesmo vale para o sistema de chaves públicas. O gerenciamento de chaves públicas pode ser realizado por um terceiro (*e.g.*, GPG), como apresentado na Seção 4.

A solução SSPD-LGPD visa ao usuário um controle maior sobre seu dados pessoais. Cabe ao usuário decidir quais empresas terão acesso aos seu dados. Porém, a solução proposta não permite ao usuário definir quais dados cada empresa terá acesso.

Um ponto a ser levantado é o custo computacional envolvido nas operações de cifra/decifra de dados. Estas operações são realizadas pelas aplicações do usuário e empresa. Por conta disto, não existe um ponto central onde estas operações são realizadas, o que torna o custo computacional quase irrelevante.

Um problema é a perda da chave do usuário. A chave do usuário é armazenada de forma decifrada somente pela aplicação do usuário. Caso esta chave seja perdida (*e.g.*, o aparelho seja furtado ou danificado), é impossível recuperar no estágio atual da solução. Como definido pelo Protocolo 4, é necessário recuperar os dados armazenados pela aplicação e decifrar utilizando a chave antiga, para então os dados serem cifrados pela nova chave e enviados de volta para a aplicação a fim de substituir a chave.

REFERÊNCIAS

- CHEN, G. et al. Sgxpectre attacks: Stealing intel secrets from sgx enclaves via speculative execution. **arXiv preprint arXiv:1802.09085**, 2018. Citado na página 25.
- CIMPANU, C. **Data of 2.4 million Blur password manager users left exposed online**. 2019. <<https://zd.net/35CNeOj>>. Citado na página 22.
- CIMPANU, C. **EatStreet food ordering service discloses security breach**. 2019. Disponível em: <<https://zd.net/2NYbR1E>>. Citado na página 15.
- COONEY, M. **IBM touts encryption innovation**. 2009. <<https://www.computerworld.com/article/2526031/ibm-touts-encryption-innovation.html>>. Citado na página 20.
- COSTAN, V.; DEVADAS, S. Intel sgx explained. **IACR Cryptology ePrint Archive**, v. 2016, n. 086, p. 1–118, 2016. Citado 2 vezes nas páginas 18 e 49.
- CREMERS, C. J. F. **Scyther: Semantics and verification of security protocols**. [S.l.]: Eindhoven University of Technology Eindhoven, Netherlands, 2006. Citado na página 37.
- DAEMEN, J.; RIJMEN, V. Aes proposal: Rijndael. **AES proposal**, 1999. Citado na página 22.
- DEHAVEN, K. **EatStreet Data Breach Risks Customers' Information**. 2019. Disponível em: <<https://bit.ly/35nm9hQ>>. Citado na página 15.
- DEVELOPER, A. **ARM TrustZone**. 2019. <<https://developer.arm.com/ip-products/security-ip/trustzone>>. Citado na página 23.
- DRAKE, M. **SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems**. 2019. <<https://do.co/34jcvfX>>. Citado na página 25.
- EGOROV, M.; WILKISON, M. Zerodb white paper. **arXiv preprint arXiv:1602.07168**, 2016. Citado na página 19.
- GENTRY, C. et al. Fully homomorphic encryption using ideal lattices. In: **Stoc**. [S.l.: s.n.], 2009. p. 169–178. Citado na página 20.
- GOOGLE. **BigQuery**. 2017. <<https://cloud.google.com/bigquery/docs/>>. Citado na página 20.
- GÖTZFRIED, J. et al. Cache attacks on intel sgx. In: **ACM. Proceedings of the 10th European Workshop on Systems Security**. [S.l.], 2017. p. 2. Citado na página 49.
- GREENBERG, A. **DARPA Will Spend \$20 Million To Search For Crypto's Holy Grail**. 2011. <<https://bit.ly/34m5MSq>>. Citado na página 20.
- GRIBOV, A.; VINAYAGAMURTHY, D.; GORBUNOV, S. Stealthdb: a scalable encrypted database with full sql query support. **arXiv preprint arXiv:1711.02279**, 2017. Citado na página 24.
- INTEL, I. R. **Software Guard Extensions SDK for Linux OS**. 2016. Citado na página 50.

- JENUARIO, T. et al. Verificação Automática de Protocolos de Segurança com a ferramenta Scyther. In: **4o Workshop Regional de Segurança da Informação e de Sistemas Computacionais**. Alegrete-RS, Brasil: [s.n.], 2019. Disponível em: <<http://errc.sbc.org.br/2019/wrseg/papers/jenuario2019verificao.pdf>>. Citado na página 37.
- LI, C. et al. Mimosa: Protecting private keys against memory disclosure attacks using hardware transactional memory. **IEEE Transactions on Dependable and Secure Computing**, IEEE, 2019. Citado na página 17.
- MACHADO, R. et al. Vazamentos de Dados: Histórico, Impacto Socioeconômico e as Novas Leis de Proteção de Dados. In: **4o Workshop Regional de Segurança da Informação e de Sistemas Computacionais**. Alegrete-RS, Brasil: [s.n.], 2019. Disponível em: <<https://bit.ly/37ih5gw>>. Citado na página 15.
- MICROSOFT. **Always Encrypted**. 2017. <<https://bit.ly/2XKjhc8>>. Citado na página 20.
- MITTERER, M. et al. An experimental performance analysis of the cryptographic database zerodb. In: ACM. **Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems**. [S.l.], 2018. p. 6. Citado na página 22.
- MOGHIMI, A.; EISENBARTH, T.; SUNAR, B. Memjam: A false dependency attack against constant-time crypto implementations in sgx. In: SPRINGER. **Cryptographers' Track at the RSA Conference**. [S.l.], 2018. p. 21–44. Citado na página 25.
- MOGHIMI, A.; IRAZOQUI, G.; EISENBARTH, T. Cachezoom: How sgx amplifies the power of cache attacks. In: SPRINGER. **International Conference on Cryptographic Hardware and Embedded Systems**. [S.l.], 2017. p. 69–90. Citado na página 49.
- MONTPETIT, J. **Personal data of 2.7 million people leaked from Desjardins**. 2019. Disponível em: <<https://www.cbc.ca/news/canada/montreal/desjardins-data-breach-1.5183297>>. Citado na página 17.
- MORIARTY, K.; KALISKI, B.; RUSCH, A. **PKCS# 5: password-based cryptography specification version 2.1**. [S.l.], 2017. Citado na página 22.
- MORRIS, L. Analysis of partially and fully homomorphic encryption. **Rochester Institute of Technology**, p. 1–5, 2013. Citado na página 20.
- NAEHRIG, M.; LAUTER, K.; VAIKUNTANATHAN, V. Can homomorphic encryption be practical? In: ACM. **Proceedings of the 3rd ACM workshop on Cloud computing security workshop**. [S.l.], 2011. p. 113–124. Citado na página 20.
- NG, A. **Thanks to Equifax breach, 4 US agencies don't properly verify your data**. 2019. <<https://cnet.co/2OmaJ8y>>. Citado 2 vezes nas páginas 15 e 17.
- OWASP. **The Ten Most Critical Web Application Security Risks**. 2017. <<https://bit.ly/2Okfm2Yf>>. Citado na página 17.
- POPA, R. A. et al. Cryptdb: protecting confidentiality with encrypted query processing. In: ACM. **Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles**. [S.l.], 2011. p. 85–100. Citado na página 19.

PRIEBE, C.; VASWANI, K.; COSTA, M. Enclavedb: A secure database using sgx. In: IEEE. **2018 IEEE Symposium on Security and Privacy (SP)**. [S.l.], 2018. p. 264–278. Citado na página 24.

SARTAKOV, V. et al. Stanlite—a database engine for secure data processing at rack-scale level. In: IEEE. **2018 IEEE International Conference on Cloud Engineering (IC2E)**. [S.l.], 2018. p. 23–33. Citado na página 24.

SATHELER, G. et al. **InovaDigital – Uma Plataforma de Identificação, Gestão, Marketing Digital e Inovação**. 2019. Citado na página 16.

SECURITY, H. N. **Human error still the cause of many data breaches**. 2019. <<https://bit.ly/34fFjpL>>. Citado na página 17.

SOUNTHIRARAJ, D. et al. Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps. In: CITESEER. **In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS’14)**. [S.l.], 2014. Citado na página 17.

SOUZA, R. de. **Exclusivo: vazam mais de 200 GB de documentos de bancos brasileiros**. 2019. <<https://bit.ly/2Dei43p>>. Citado na página 17.

SOUZA, R. de. **Para frear cibercrime, Brasil estuda criar Lei Geral de Segurança Cibernética**. 2019. <<http://bit.do/fbukU>>. Citado na página 43.

STANDARDS, N. I. of; TECHNOLOGY. **Block Cipher Techniques**. 2019. <<https://csrc.nist.gov/projects/block-cipher-techniques/bcm>>. Citado na página 22.

TRIMBERGER, S. M.; MOORE, J. J. Fpga security: Motivations, features, and applications. **Proceedings of the IEEE**, IEEE, v. 102, n. 8, p. 1248–1265, 2014. Citado na página 23.

TURNER, D. M. **What is Key Management? a CISO Perspective**. 2019. <<https://www.cryptomathic.com/news-events/blog/what-is-key-management-a-ciso-perspective>>. Citado na página 19.

WANG, Y. et al. Cryptsqlite: Protecting data confidentiality of sqlite with intel sgx. In: IEEE. **2017 International Conference on Networking and Network Applications (NaNA)**. [S.l.], 2017. p. 303–308. Citado na página 24.

WEICHBRODT, N.; AUBLIN, P.-L.; KAPITZA, R. sgx-perf: A performance analysis tool for intel sgx enclaves. In: ACM. **Proceedings of the 19th International Middleware Conference**. [S.l.], 2018. p. 201–213. Citado na página 24.

ZETETIC, L. **SQLCIPHER**. 2015. <<https://www.zetetic.net/sqlcipher/documentation/>>. Citado na página 19.

8 APENDICE

Intel SGX

Intel SGX é uma extensão da arquitetura do conjunto de instruções x86 presente nas gerações mais recentes dos processadores Intel (COSTAN; DEVADAS, 2016). SGX usa hardware específico para possibilitar a execução segura de contêineres chamados enclaves. A região de memória que reside no enclave é protegida contra todo tipo de acesso externo, incluindo acesso privilegiado do sistema operacional ou hypervisor. Cada enclave é isolado e suporta selamento de dados (*sealing*, atestação local e remota).

Enclaves SGX são identificados unicamente pelo *Enclave Control Structure*, que contém um digest de 256 bits de um log criptográfico gravado no processo de compilação do enclave. Esta informação é utilizada para reproduzir uma chave de selamento gerada pelo hardware. A chave de selamento depende do digest e da chave do hardware, o que significa que os dados selados (cifrados) só podem ser decifrados pelo enclave que os selou. Isto garante um armazenamento seguro (i.e. confidencial e onde a integridade pode ser verificada) dos dados e código dos enclaves.

Intel SGX possui dois protocolos de atestação, um local e outro remoto. A atestação local possibilita que um enclave E1 autentique o programa que está executando em outro enclave E2. Para realizar a atestação, é utilizada uma cópia assinada do *digest* da selagem dos dados do E2 e o protocolo Diffie-Hellman para auxiliar na autenticação da integridade de um programa de P1 num enclave E2.

A atestação remota é implementada usando o processo de atestação local em conjunto com um par de enclaves de autoria da Intel (um enclave de provisionamento e um enclave de referência). O enclave de provisionamento requisita uma chave de atestação da Intel e armazena-a de forma selada, limitando o acesso e derivação de chaves aos enclaves de autoria da Intel. O enclave de referência requer a chave de atestação, verifica a medição usando atestação local e assina a medida junto com uma mensagem opcional. O par medida-mensagem, chamado de *quote*, pode ser verificado usando o *Intel Attestation Service*.

Apesar da segurança fornecida pelos enclaves, estes não estão livres de ataques. Através de ataques à cache, é possível recuperar dados do enclave (MOGHIMI; IRAZOQUI; EISENBARTH, 2017) (GÖTZFRIED et al., 2017). Nos casos demonstrados, é necessário permissão de *root* ao sistema para realizar os ataques, algo viável para um administrador de sistema malicioso. Outro fator limitante para a aplicação do SGX em sistemas é de que tanto o servidor de aplicação quanto o servidor de banco de dados precisam ter suporte para a tecnologia, algo que nem sempre é viável, dependendo da aplicação. Além de problemas de segurança, a tecnologia apresenta limitações com relação à desempenho. Os enclaves são limitados a um espaço de memória de 90MB. As aplicações precisam ser otimizadas devido a problemas de desempenho do SGX. As únicas linguagens suportadas pelo SGX são C e C++, sendo que as funções de interfaces estão

limitadas à linguagem C (INTEL, 2016).