ASSINATURAS ENCADEADAS: GARANTIA DE INTEGRIDADE EM PROCESSOS SAAS

Jonas Rodrigues da Rocha[1], Gustavo Bisognin[2]

Resumo: Este trabalho apresenta uma proposta de solução para o problema de confiança na integridade de dados em aplicações distribuídas no modelo *Software as a Service* (SaaS). A solução é inspirada na tecnologia *blockchain*, apresentando uma arquitetura de eventos assinados e encadeados que permite que o cliente da aplicação possa verificar a integridade dos dados gerados pelo sistema de forma independente. A prova de conceito implementada demonstrou que a arquitetura apresentada é eficaz no endereçamento do problema, com testes realizados apresentando 100% de sucesso. Os testes também demonstraram que a utilização de padrões abertos como XML juntamente com o uso de assinaturas digitais permitem construir sistemas com maior grau de transparência. Os resultados gerados no estudo permitem concluir que a arquitetura proposta é um caminho promissor, podendo ser melhorada e aplicada em aplicações em ambientes reais.

Palavras-chave: integridade de dados; assinaturas digitais; blockchain.

ABSTRACT: This work presents a proposed solution to the problem of trust in data integrity within distributed applications in the Software as a Service (SaaS) model. The solution is inspired by blockchain technology, introducing an architecture of signed and chained events that enables application clients to independently verify the integrity of data generated by the system. The proof of concept implemented demonstrated that the proposed architecture effectively addresses the problem. The tests also showed that the use of open standards like XML, combined with digital signatures, allows for the development of systems with a higher degree of transparency. The results of the study indicate that the proposed architecture is a promising approach, with potential for improvement and application in real-world environments.

Keywords: data integrity; digital signatures; blockchain.

[1]jonasrodrigues@unesc.net
[2]gustavo@unesc.net

1 INTRODUÇÃO

A adoção de soluções baseadas em Software como Serviço (SaaS) tem crescido exponencialmente tanto no setor corporativo quanto na esfera pública. Conforme destacado pela Associação Brasileira de Startups (ABSTARTUPS, 2022), cerca de 40% das startups brasileiras adotam o modelo SaaS como principal estratégia de negócio, evidenciando sua relevância no cenário tecnológico atual. As vantagens econômicas e operacionais, como a redução de custos e a diminuição da complexidade na manutenção de infraestruturas dedicadas, tornam as aplicações em nuvem cada vez mais atrativas (Taurion, 2009).

Entretanto, essa dependência crescente de serviços em nuvem levanta preocupações importantes em relação à segurança e integridade dos dados armazenados e processados. A confiança dos clientes na integridade dos dados depende diretamente da confiança depositada no fornecedor da solução. Frequentemente, os dados são armazenados em bancos de dados centralizados, controlados pelo provedor do serviço, o que pode levar a preocupações sobre possíveis alterações indevidas, sejam elas resultantes de falhas técnicas, erros humanos ou ações maliciosas (Chou, 2013).

Alterações indevidas ou perdas de dados são um risco real em soluções SaaS e podem ser causados por configurações incorretas que comprometam a segurança da aplicação ou até por atores internos com acesso privilegiado (Chou, 2013). Conforme relatório recente publicado pela Verizon (VERIZON BUSINESS, 2024), cerca de 35% dos ataques analisados contaram com atores internos. Embora o relatório trate de ataques de modo geral, não podemos desconsiderar que parte destes ataques ocorrem em soluções SaaS.

Além de agentes internos, explorações de falhas de segurança também são um problema. Como exemplo podemos citar uma importante falha de segurança em um serviço da *Transportation Security Administration* onde um ator externo por meio de acesso administrador conseguia incluir qualquer pessoa em uma lista de passageiros que permitiria transpassar alguns procedimentos de segurança dos aeroportos (Carroll, 2024).

Para mitigar tais riscos, os fornecedores de serviços SaaS adotam normas e boas práticas de segurança. A ISO/IEC 27017, por exemplo, fornece diretrizes para a implementação de controles de segurança da informação em serviços em nuvem (ABNT, 2016). Além disso, frameworks

como o OWASP são amplamente utilizados para identificar e mitigar vulnerabilidades em aplicações web (OWASP, 2024).

Em relação à integridade e imutabilidade de dados, diversas soluções têm sido propostas para abordar esse problema, incluindo o uso de bancos de dados imutáveis e blockchains privadas. Os bancos de dados imutáveis, como são gerenciados pelo próprio provedor de serviços, ainda requerem que o cliente confie na infraestrutura e nos mecanismos de segurança implementados pelo fornecedor.

Blockchains privadas, por sua vez, permitem que organizações criem redes fechadas onde os participantes são conhecidos e confiáveis. Embora ofereçam controle sobre quem pode participar e validar transações, essas blockchains ainda enfrentam desafios relacionados à centralização do controle(Buterin, 2015).

Essas soluções, embora avancem na direção de maior transparência e segurança, não eliminam a necessidade de confiança no fornecedor ou na entidade que administra a infraestrutura. Os usuários continuam sem meios eficazes para verificar, de forma independente, a integridade dos dados e dos processos aos quais estão submetidos.

No presente trabalho é proposta uma solução para uma aplicação SaaS de processos digitais que permita aos clientes da solução, verificar a integridade dos dados de forma independente. A solução se inspira na tecnologia blockchain, de forma que cada evento gerado referencia um evento anterior sendo assinado digitalmente usando certificado digital. Para verificar os dados, o sistema disponibiliza um arquivo XML com todos os eventos assinados de forma que o usuário pode verificar a integridade usando ferramentas próprias ou até ferramentas disponíveis na internet. Para avaliar a proposta, uma prova de conceito foi implementada gerando os resultados descritos nas próximas seções.

2 MATERIAIS E MÉTODOS

Este trabalho propõe uma arquitetura de sistema que permite que clientes de uma aplicação SaaS possam verificar, sem depender do fornecedor do serviço, a integridade dos dados gerados. Sendo este um trabalho de pesquisa aplicada de base tecnológica, o seu desenvolvimento foi dividido em duas etapas: pesquisa bibliográfica e proposta e validação da solução.

A pesquisa bibliográfica forneceu as bases para as duas etapas seguintes. Nesta etapa foram levantados dados referentes ao processo de

desenvolvimento de software em relação à integridade de dados e quais ferramentas e tecnologias podem ser adotadas para resolver o problema. Por fim, partindo da proposta inicial, foi descrita uma arquitetura para endereçar o problema. Esta arquitetura passou por uma avaliação exploratória por meio da implementação de uma prova de conceito para verificar sua viabilidade e funcionalidade. A solução baseia-se na criação de uma cadeia de eventos interligados, inspirada na tecnologia blockchain, onde cada evento referencia o evento anterior, sendo assinado digitalmente utilizando certificados digitais.

2.1 LEVANTAMENTO BIBLIOGRÁFICO

O levantamento bibliográfico incluiu referências nacionais e internacionais de livros, artigos, documentos normativos, documentos técnicos e sites. Durante a pesquisa não foram encontrados trabalhos correlatos que pudessem contribuir além das referências bibliográficas. Foram realizadas consultas em três repositórios acadêmicos utilizando os termos blockchain e assinaturas encadeadas. Os resultados são apresentados nas tabelas a seguir.

Tabela 1 – Resultados para a busca pelo termo blockchain.

Repositório	Filtros	Total
UNESC [3]	Ciências, engenharias e tecnologias	5
UFSC[4]	TCC Ciências da Computação	12
USP[5]	Ciências de Computação e Mat. Computacional	6

Fonte: Elaborado pelo autor.

Tabela 2 – Resultados para a busca pelo termo assinaturas encadeadas.

Repositório	Filtros	Total
UNESC	Ciências, engenharias e tecnologias	2 [6]
UFSC	TCC Ciências da Computação	0
USP	Ciências de Computação e Mat. Computacional	0

Fonte: Elaborado pelo autor.

Em complemento as referências técnicas e acadêmicas, foram utilizados relatórios de organizações privadas ou associações para obter

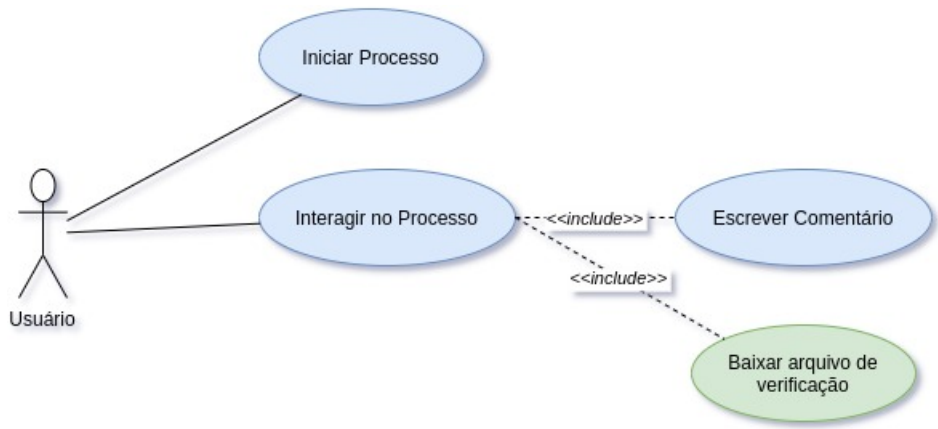
[3] Repositório Institucional da UNESC (UNESC, 2024)
[4] Repositório Institucional da UFSC (UFSC, 2024)
[5] Repositório Institucional da USP (USP, 2024)
[6] Os resultados retornados não são da área de ciências da computação

dados do mercado de SaaS e também de incidentes de segurança. Existem muitas fontes que fornecem informações sobre o uso de SaaS e também de incidentes de segurança. Esta pesquisa limitou-se a usar somente duas fontes para desenhar um cenário aproximado, deixando a cargo de pesquisas posteriores a exploração das demais fontes.

2.2 CENÁRIO DE ESTUDO

Como ponto de partida foi realizado um estudo de caso de uma aplicação SaaS fictícia de processos digitais contando com dois casos de uso: iniciar um processo e interagir no processo. Iniciar um processo cria um registro na tabela de processos com o título e a descrição do processo. Ao interagir com um processo, o usuário pode inserir um comentário que gera um registro na tabela de eventos. A solução proposta acrescenta um novo caso de uso que permite que o usuário baixe o arquivo de verificação do processo. Para simplificar o estudo de caso, outros módulos necessários em uma aplicação real foram ignorados. O diagrama abaixo ilustra os casos de uso iniciais na cor azul e o novo caso de uso na cor verde.

Figura 1 - Diagrama de casos de uso da aplicação fictícia utilizada como estudo de caso. A cor verde representa o novo caso de uso que faz parte da proposta. Em azul os casos de uso pré existentes.



Fonte: Elaborado pelo autor

2.3 TECNOLOGIAS UTILIZADAS

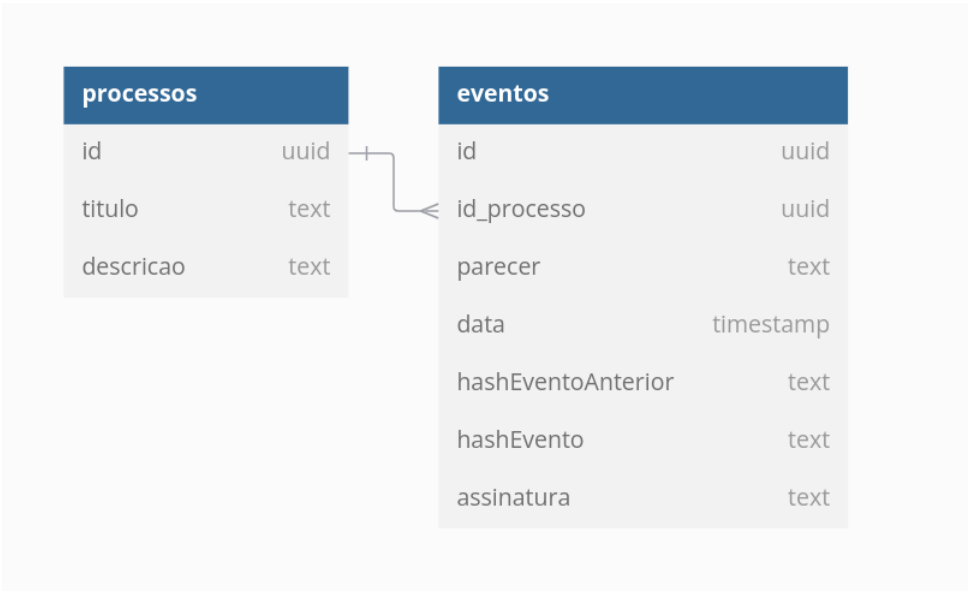
Para implementar a prova de conceito foram utilizadas diversas tecnologias como banco de dados, padrões de assinatura, certificados digitais e formatos de arquivos. As principais tecnologias e motivo de sua escolha são descritas a seguir.

2.3.1 Banco de dados

Para a implementação da prova de conceito foi escolhido o banco de dados o SQLite. A escolha foi para simplificar a infraestrutura necessária para rodar a prova de conceito. O SQLite é amplamente utilizado em ambiente de desenvolvimento por necessitar de poucos recursos de processamento e por ter configuração fácil além de bibliotecas para as mais variadas linguagens de programação sem necessidade de instalar pacotes adicionais no sistema operacional além das bibliotecas da própria linguagem.

O banco de dados da prova de conceito possui duas tabelas para armazenar os dados: processos e eventos. A tabela processos é responsável por armazenar o identificador, titulo e descrição do processo. Já a tabela eventos é responsável por armazenar as interações do usuário no processo. A figura 2 apresenta o modelo de entidade-relacionamento da estrutura de banco de dados utilizada na prova de conceito.

Figura 2 - Modelo de entidade-relacionamento da prova de conceito implementada



Fonte: Elaborado pelo autor

Na prova de conceito, o evento possui dois campos de preenchimento do usuário: processo e parecer. O campo processo é uma referência ao identificador do processo, indicando que o evento pertence ao processo referenciado. Já o campo parecer é um campo de texto onde o usuário irá registrar um comentário.

Além dos campos de preenchimento do usuário, o evento possui campos de preenchimento do sistema como *hashEventoAnterior*, que é preenchido com o *hash* gerado e assinado no evento anterior, o campo *hashEvento*, onde o sistema irá guardar o *hash* do evento e por fim o campo *signature*, que será preenchido com a assinatura do evento.

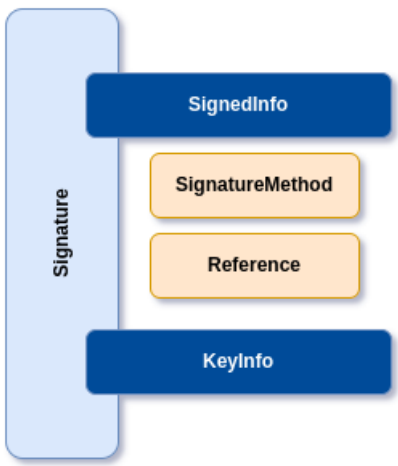
O tipo de dado escolhido como chave primária foi o *Universally Unique Identifier*(UUID). Uma das características do UUID é sua unicidade global e independente de contexto, que garante a geração de identificadores únicos independente do contexto da aplicação (Leach; Salz; Mealling, 2005). A escolha pelo UUID em oposição ao número inteiro sequencial foi pela possibilidade de criar o identificador antes de persistir o dado no banco de dados, sem a necessidade de consultar pelos registros anteriores para calcular o próximo identificador.

2.3.2 Formato de exportação

O *Extensible Markup Language* (XML) foi escolhido como formato de exportação utilizado no projeto por ser um padrão amplamente utilizado compatível com as principais linguagens de programação que permite que outras aplicações processem os dados exportados. Além do XML foi utilizado o XML *Signature*(XML-Sig), um padrão de assinaturas do *World Wide Web Consortion* (W3C) (W3C, 2013).

O XML-Sig possui uma estrutura que inclui não só a assinatura como o certificado utilizado, e permite referenciar o elemento que está sendo assinado como pode ser observado na imagem abaixo:

Figura 3 - Estrutura de dados do XML-Sig

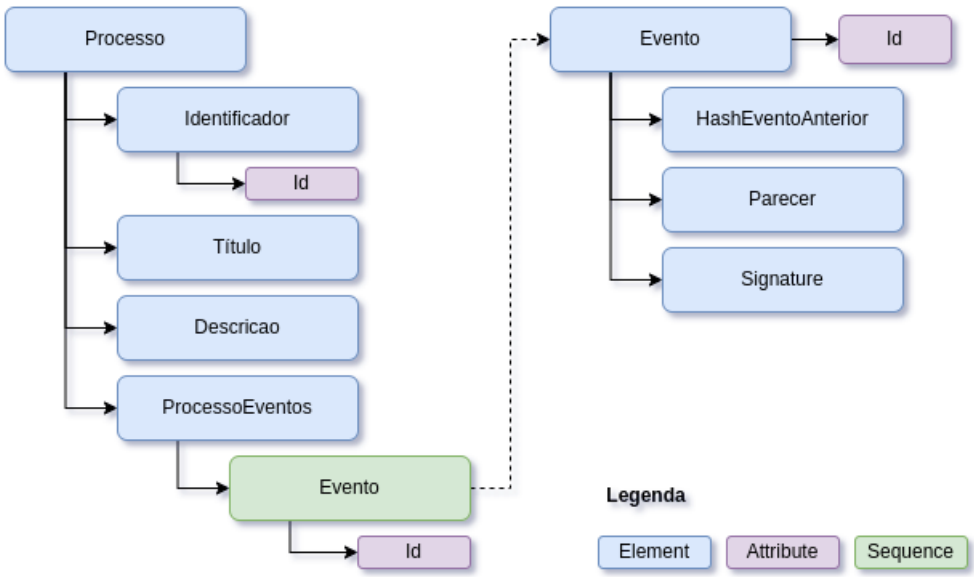


Fonte: Elaborado pelo autor

O elemento *Reference* é responsável por referenciar o elemento que está sendo assinado. Esse elemento é importante para a solução que está sendo abordada, por permitir gerar a assinatura para um elemento referenciado que, se alterado posteriormente, irá invalidar a própria assinatura. Outro elemento importante é o *KeyInfo* pois ele armazena a chave pública do certificado utilizado, permitindo a validação por terceiros apenas com o documento gerado. Para padronizar e documentar o arquivo de verificação gerado foi criado um *XML Schema Definition* (XSD).

A Definição de Esquema XML (XSD) apresentada na figura 4 descreve a estrutura de um modelo de 'Processo'. O elemento raiz Processo contém atributos e elementos aninhados para identificação (Id), título (Titulo), descrição (Descricao) e eventos associados (ProcessoEventos). Cada elemento Evento captura os detalhes de um evento específico dentro do processo, incluindo o hash anterior e o parecer. Assinaturas digitais, baseadas na especificação de Assinatura XML, são incluídas no evento para garantir a integridade dos dados.

Figura 4 - Schema XSD para o arquivo de verificação



Fonte: Elaborado pelo autor

2.3.3 Assinaturas Digitais

As assinaturas digitais desempenham um papel crucial na garantia da integridade, autenticidade e não repúdio dos eventos registrados no sistema desenvolvido. Neste projeto, cada ação realizada pelos usuários gera um evento que é assinado digitalmente, assegurando que o conteúdo

não foi alterado após a assinatura e que a autoria pode ser verificada de forma confiável. Para a implementação da prova de conceito e sua validação foram utilizados certificados gerados pela ferramenta de código aberto XCA (Hohnstädt, 2024). Os certificados gerados para teste permitem o uso para assinatura digital, certificado digital e não repúdio. Esses usos permitem validar a integridade dos conteúdos assinados. Como algoritmo de assinatura, o certificado utiliza o padrão *Public-Key Cryptography Standards 1* (PKCS1) (Kaliski, 1998), que é um padrão da RSA Laboratories para criptografia de chave pública, especificamente para a aplicação do algoritmo RSA em operações de criptografia e assinatura digital. Este padrão define como utilizar o RSA de forma segura e é amplamente adotado em protocolos de segurança na internet, como SSL/TLS. Para fins de assinatura, a prova de conceito utilizou o algoritmo de hash o SHA-256 (Eastlake; Hansen, 2006).

2.3.4 Linguagem de programação

Para a implementação da prova de conceito foi escolhida a linguagem de programação Python na versão 3.10. O Python é uma linguagem de programação do tipo interpretada, de alto nível e orientada a objetos com tipagem dinâmica. Estas características somadas a popularidade (Github, 2024) da linguagem foram os pontos-chave para a decisão de usar a linguagem no projeto. O Python possui todos os módulos necessários para a implementação da prova de conceito embarcados na linguagem ou disponíveis em seu repositório de módulos, *Python Package Index* (PyPi), que podem ser facilmente instaladas utilizando a ferramenta pip.[7] Além dos módulos presentes na própria linguagem foram instalados pacotes especializados em manipulação de XML, gerenciamento de certificados e conexão com banco de dados. A tabela abaixo descreve os módulos e versões instalados bem como sua utilidade no projeto:

Tabela 3 – Pacotes python instalados para implementação da prova de conceito

Pacote	Versão	Utilidade
lxml	5.3.0	Manipulação e leitura de arquivos XML
signxml	4.0.2	Assinatura de XML
xmlsec	1.3.14	Usada para verificar as assinaturas
cryptography	43.0.1	Responsável por carregar o certificado digital

Fonte: Elaborado pelo autor.

[7] Ferramenta de linha de comando que permite instalar módulos python

2.4 SOLUÇÃO PARA O PROBLEMA DE CONFIANÇA

A solução apresentada neste trabalho se utiliza de assinaturas encadeadas para fornecer a integridade e imutabilidade esperada. Semelhante ao proposto na arquitetura do bitcoin, quando um novo processo é iniciado por um usuário da aplicação, é gerado um evento gênesis usado como referência inicial. Com um processo iniciado, cada interação no processo gera um novo evento que referencia o evento anterior relacionado aquele processo (Nakamoto, 2008).

Cada novo evento criado deve referenciar o evento anterior com exceção do evento inicial. Esta referência é criada ao preencher o campo hashEventoAnterior (figura 2) com o resumo criptográfico utilizado na assinatura do evento anterior. Dessa forma, o evento recém criado fica ligado ao evento anterior que, se for alterado, irá gerar uma inconsistência que pode ser verificada. Preenchida a referência do evento, o mesmo é convertido para a estrutura XML e então assinado. Desta forma, qualquer tentativa de alterar o evento anterior e alterar sua referência no evento posterior irá gerar uma inconsistência na assinatura digital, invalidando o processo. Após assinar o evento, o elemento *Signature* (figura3) é salvo no campo assinatura do evento e será usado posteriormente para montar o arquivo de verificação.

Ao solicitar o arquivo de verificação, o sistema irá buscar todos os eventos do processo selecionado ordenando pela data, da mais antiga para a mais recente e irá montar o arquivo de verificação XML sequencialmente, de forma que todos os eventos fiquem na mesma ordem em que foram inseridos.

2.4.1 Arquivo de verificação

O arquivo de verificação é um componente central na solução proposta, servindo como meio para que os usuários possam verificar, de forma independente, a integridade dos eventos registrados em um processo. Este arquivo é gerado no formato XML, seguindo um esquema definido em um XML Schema Definition (XSD), apresentado na Figura 4. Cada evento no arquivo XML inclui os elementos essenciais para a verificação:

a) o elemento raiz 'Processo' que representa o processo completo. Contém atributos e elementos que identificam o processo e agrupam os eventos gerados;

b) o elemento 'ProcessoEventos' é o elemento filho do ele-

mento raiz que contém os eventos gerados pelos usuários;

c) o elemento 'Evento' que representa uma interação ou ação realizada no processo e que é assinada.

O uso do XSD permite que o arquivo XML de verificação seja validado sintaticamente, assegurando que todos os elementos necessários estão presentes e que os dados seguem o formato esperado. Isso é crucial para que ferramentas externas possam interpretar e verificar corretamente o arquivo. Além da estrutura definida pelo XSD, o arquivo XML incorpora assinaturas digitais para cada evento, utilizando o padrão XML-Sig. A assinatura digital é inserida dentro do elemento Signature, que é aninhado diretamente no elemento do evento correspondente. A utilização do XML-Sig dentro do arquivo XML permite que a verificação das assinaturas seja realizada de forma integrada, utilizando ferramentas compatíveis com o padrão. O usuário pode validar a integridade de cada evento e do processo como um todo, garantindo que os dados não foram alterados desde a sua assinatura.

2.4.2 Gerando Arquivo de Verificação

Para a geração do arquivo de verificação, a aplicação segue uma sequência de etapas bem definidas, de forma a criar um arquivo XML contendo todas as informações relevantes ao processo. O primeiro passo é a recuperação do processo e dos eventos do processo armazenados no banco de dados. A busca pelos eventos é ordenada pela data decrescente, mantendo a mesma ordem em que os eventos foram inseridos no sistema.

Em seguida, o sistema passa para a etapa de construção do XML. Nesta etapa ele inicia construindo o elemento raiz 'Processo' e nele adiciona os elementos de identificação como título, descrição e identificador, conforme a estrutura previamente definida no XSD. Após, o sistema fará a iteração nos eventos retornados na busca da etapa anterior e para cada evento, irá montar o XML com os dados atuais do banco e logo em seguida, adicionará o elemento 'Signature' com o valor retornado do banco. Toda esta etapa acontece em memória.

Por fim, após gerado o XML, o mesmo é escrito na pasta de exportação onde poderá ser fornecido ao solicitante.

Figura 5 - Representação do processo de exportação do arquivo de verificação

Fonte: Elaborado pelo autor

2.4.3 Verificação de Integridade

Após a geração do arquivo de verificação, o usuário tem a possibilidade de validar a integridade dos eventos em três passos: verificação de conformidade estrutural do XML, verificação de integridade das assinaturas e verificação da integridade dos eventos encadeados. A verificação estrutural do XML é feita validando o arquivo resultante com o esquema XSD fornecido. Deste modo é possível verificar se o arquivo foi gerado corretamente antes de seguir para as demais verificações.

Para verificar a integridade das assinaturas o usuário poderá optar por ferramentas online como o Validar, fornecida pelo Instituto Nacional de Tecnologia da Informação (ITI)[8]. Esta ferramenta além de verificar a integridade também permite validar a autenticidade do certificado, ideal para processos que envolvem o setor público. Além da ferramenta citada acima, qualquer outra ferramenta capaz de validar a integridade de assinaturas em arquivos XML pode ser adotada, inclusive ferramentas desenvolvidas pelo próprio usuário, o que permite um alto nível de transparência e independencia.

Por fim, para verificar o encadeamento de eventos é necessário escrever um programa ou utilizar o disponibilizado pelo fornecedor que percorra cada evento do mais recente para o mais antigo e verifique se o valor do elemento 'HashEventoAnterior' é igual ao valor do elemento 'DigestValue' do evento anterior. Este procedimento é importante, pois permite garantir que o evento anterior não foi alterado. Apesar de ser necessária uma ferramenta específica para validar, a disponibilização do XSD bem como o formato XML e sua alta compatibilidade com as linguagens de programação mais populares facilita a implementação em cenários reais, sendo possível inclusive a disponibilização de uma ferramenta com código aberto pelo próprio fornecedor.

[8]https://validar.iti.gov.br/index.html

3 RESULTADOS E DISCUSSÃO

A solução proposta validada pela prova de conceito implementada apresentou resultados positivos para resolver o problema de confiança em aplicações SaaS, especialmente em ambientes onde a integridade e a autenticidade dos dados são cruciais. Com o uso de assinaturas encadeadas, foi possível demonstrar como eventos podem ser validados em sequência, reforçando a ideia de um histórico imutável e auditável, o que é de grande valor em sistemas que precisam garantir a conformidade e a transparência.

Durante a etapa de pesquisa e levantamento bibliográfico, foi analisada a possibilidade de criar um formato de exportação próprio onde se teria maior controle. A ideia de formato próprio foi descartada em benefício de padrões abertos amplamente utilizados, o que permitiria maior transparência e também facilidade de implementação em qualquer outra linguagem. Nesse contexto, o XML se mostrou uma escolha acertada, pois além de ser um padrão aberto e amplamente utilizado, permite inclusive a validação de esquema de dados. Outro fator que pesou na escolha do XML foi a sua utilização no setor público do Brasil, com destaque para as especificações e ferramentas disponibilizadas pelo Instituto Nacional de Tecnologia da Informação que permitem a validação de arquivos XML assinados.

Apesar dos pontos positivos, o formato XML trouxe desafios. O primeiro deles é em relação à montagem do XML. Como a montagem é feita parte com dados vindas do banco e montadas em tempo real, e parte dos dados já previamente montados e salvos no banco de dados (assinatura), ao montar o arquivo, qualquer alteração de formatação pode gerar falsa inconsistência. Esse foi o principal desafio. Nos primeiros testes realizados, a ferramenta de edição de códigos estava formatando o arquivo gerado quando era aberto. O arquivo ficava com a indentação correta, porém, ao passar pelo validador, todas as assinaturas ficavam inconsistentes. Após algumas tentativas o problema da indentação foi percebido e o editor configurado para não formatar automaticamente. Isso preveniu a inconsistência, porém o arquivo XML gerado estava sem as indentações convencionadas.

Outro desafio encontrado foi em relação ao encadeamento dos eventos. A ideia abordada cria utiliza o resumo criptográfico do evento anterior e adiciona como parte do próprio evento, criando assim um encadeamento. Em um teste onde um evento foi removido, ao passar o arquivo de verificação em uma ferramenta de validação, o mesmo não acusou incon-

sistência. Isso acontece por que o resumo criptográfico do evento seguinte ao evento removido não foi alterado, já que ele guarda em campo próprio o resumo criptográfico do evento anterior, mesmo ele não existindo. Este problema pode ser resolvido fazendo a assinatura do evento atual juntamente com o evento anterior. Desta forma, os eventos seriam criptograficamente encadeados e qualquer alteração ou falta de algum evento acusaria inconsistência.

3.1 TESTES REALIZADOS

Foram realizados testes para validar a confiabilidade da solução proposta implementada na prova de conceito. Para verificar a integridade das assinaturas foi utilizada a ferramenta *Verify XML Digital Signature*, disponibilizada online[9]. Além da ferramenta citada acima, foi implementada como parte do projeto uma ferramenta para validar a integridade das assinaturas e também o encadeamento delas. Em relação aos testes realizados, foi utilizado em todas as assinaturas o mesmo certificado. O certificado foi criado pelo próprio acadêmico com as características necessárias para assinaturas digitais. Isso, no entanto, não afetou os resultados dos testes realizados, pois a proposta é validar a integridade dos eventos criptograficamente e não a autenticidade da assinatura.

Para fins de padronização, cada teste realizado iniciou com um processo contendo três eventos. Os processos e eventos foram criados manualmente e mantidos no banco de dados a medida que eram testados, simulando desta forma um ambiente mais próximo do real. No total foram gerados 8 processos para atender os cenários:

a) ao verificar um processo que não sofreu alterações indevidas não deve apresentar inconsistências;

b) quando um evento do processo for deletado do banco de dados a verificação deve apresentar inconsistência;

c) quando algum campo de um evento do processo for alterado indevidamente, a verificação deve apresentar inconsistência

Para os cenários onde foram criadas inconsistências propositais, as inconsistências foram aplicadas ao primeiro evento em um teste e no segundo evento no teste seguinte. Nas inconsistências onde um dado do

[9] https://tools.chilkat.io/xmlDsigVerify.cshtml

evento foi alterado, foram alterados somente o campo 'parecer' e 'hashEventoAnterior', sendo uma alteração de um tipo em cada teste.

Em nenhum dos testes foram alterados mais do que um evento e também não foram realizados testes de alteração no próprio processo. Além disso, cada teste foi validado tanto na ferramenta online quanto na ferramenta implementada no projeto, gerando os resultados consolidados nos quadros abaixo:

Tabela 4 – Resultados da verificação de integridade do processo em cada cenário utilizando a ferramenta *Verify XML Digital Signature*

Cenário	Testes	Sucesso	Falha
Processo íntegro	2	2	0
Evento indevidamente removido	2	0	2
Campo 'eventoAnterior' indevidamente alterado	2	2	0
Campo 'parecer' indevidamente alterado	2	2	0

Fonte: Elaborado pelo autor.

Tabela 5 – Resultados da verificação de integridade do processo em cada cenário utilizando a ferramenta implementada

Cenário	Testes	Sucesso	Falha
Processo íntegro	2	2	0
Evento indevidamente removido	2	2	0
Campo 'parecer' indevidamente alterado	2	2	0
Campo 'eventoAnterior' indevidamente alterado	2	2	0

Fonte: Elaborado pelo autor.

Do total de testes realizados pôde-se observar que a ferramenta própria implementada obteve o acerto de 100%, demonstrando a viabilidade da proposta. Por outro lado, ao utilizar a ferramenta online, a taxa de sucesso foi de apenas 80%. Isso por que, como já mencionado anteriormente, a simples verificação de integridade das assinaturas não é capaz de validar o encadeamento, sendo esperado que nos casos onde um evento foi deletado a verificação daria um falso positivo.

4 CONCLUSÃO

Este trabalho apresentou uma solução que permite aos clientes de uma aplicação SaaS verificar a integridade dos dados de forma independente. A abordagem focada no uso de assinaturas digitais encadeadas

demonstrou ser capaz de prover a transparência e a confiança na integridade dos dados.

A escolha por tecnologias abertas, como XML e XML-Sig, se mostraram positivas ao por permitir compatibilidade e facilidade de validação por ferramentas de terceiros. A utilização de um formato de exportação padronizado amplia a aplicabilidade do sistema e possibilita a replicação da solução em outras plataformas, ampliando seu potencial de adoção.

A implementação da prova de conceito evidenciou os desafios e do uso de XML, especialmente no que diz respeito à manipulação de arquivos. Problemas como formatação automática feita por alguns editores demonstram que é necessário cuidados ao manipular estes arquivos quando assinados.

Outro desafio encontrado está relacionado ao encadeamento dos eventos. A abordagem implementada utiliza o resumo criptográfico do evento anterior e o inclui como parte do evento atual, estabelecendo assim uma cadeia. Essa abordagem se mostrou eficaz usando a ferramenta de verificação implementada na pesquisa, mas não passou em verificações de ferramentas de terceiro.

Os resultados obtidos com esses testes, bem como os problemas apontados podem ser tópico de pesquisa para trabalhos futuros. A exploração de abordagens alternativas para o encadeamento dos eventos pode trazer ainda mais aplicabilidade para a proposta. Além disso, trabalhos futuros podem expandir a pesquisa realizada, utilizando estrutura de dados mais complexas e aplicando em cenários reais.

O trabalho apresentado contribuiu com a discussão de um tema que é relevante e pouco explorado em pesquisas acadêmicas, ressaltando a importância de novos trabalhos direcionados ao tema.

5 REFERÊNCIAS

Referências

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 27002**: Tecnologia da informação – técnicas de segurança – código de prática para controle de segurança da informação com base na abnt nbr iso/iec 27002 para serviços em nuvem. Rio de Janeiro, 2016.

ASSOCIAÇÃO BRASILEIRA DE STARTUPS. **Mapeamento de Startups Brasil 2022**. [S.l.], 2022. Disponível em: <https://abstartups.com.br/wp-content/uploads/2023/01/Mapeamento-de-Startups-Brasil-2022.pdf>. Acesso em: 2024-08-27.

BUTERIN, V. **On Public and Private Blockchains**. 2015. Disponível em: <https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains>. Acesso em: 2024-10-20.

CARROLL, I. **Bypassing airport security via SQL injection**. 2024. Disponível em: <https://ian.sh/tsa>. Acesso em: 2024-08-23.

CHOU, T.-S. Security threats on cloud computing vulnerabilities. **International Journal of Computer Science and Information Technology**, v. 5, p. 79–88.

EASTLAKE, D.; HANSEN, T. **US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)**. 2006. RFC 4634. Disponível em: <https://datatracker.ietf.org/doc/html/rfc4634>. Acesso em: 2024-10-20.

Github. **The Most Popular Programming Languages**. 2024. Disponível em: <https://github.blog/news-insights/octoverse/octoverse-2024/#the-most-popular-programming-languages>. Acesso em: 2024-10-28.

HOHNSTäDT, C. **X Certificate and Key Management**. 2024. Disponível em: <https://github.com/chris2511/xca>. Acesso em: 2024-05-25.

KALISKI, B. **PKCS #1: RSA Cryptography Specifications Version 2.0**. 1998. RFC 2437. Disponível em: <https://www.rfc-editor.org/rfc/rfc2437>. Acesso em: 2024-10-20.

LEACH, P. J.; SALZ, R.; MEALLING, M. H. **A Universally Unique IDentifier (UUID) URN Namespace**. RFC Editor, 2005. RFC 4122. (Request for Comments, 4122). Disponível em: <https://www.rfc-editor.org/info/rfc4122>.

NAKAMOTO, S. **Bitcoin: A Peer-to-Peer Electronic Cash System**. 2008. White paper. Disponível em: <https://bitcoin.org/bitcoin.pdf>.

OWASP. **About OWASP**. 2024. Disponível em: <https://owasp.org/about>. Acesso em: 2024-08-23.

TAURION, C. **Cloud Computing: computação em nuvem. Transformando o mundo da tecnologia da informação**. 1. ed. Rio de Janeiro: Brasport, 2009.

UNIVERSIDADE DE SÃO PAULO. **Repositório da Produção USP**. [S.l.], 2024. Disponível em: <https://repositorio.usp.br/>. Acesso em: 2024-08-23.

UNIVERSIDADE DO EXTREMO SUL CATARINENSE. **Repositório Institucional da UNESC**. [S.l.], 2024. Disponível em: <http://repositorio.unesc.net>. Acesso em: 2024-08-23.

UNIVERSIDADE FEDERAL DE SANTA CATARINA. **Repositório Institucional da UFSC**. [S.l.], 2024. Disponível em: <https://repositorio.ufsc.br>. Acesso em: 2024-08-23.

VERIZON BUSINESS. **2024 Data Breach Investigations Report**. [S.l.], 2024. Disponível em: <https://www.verizon.com/business/en-gb/resources/reports/2024/dbir/2024-dbir-data-breach-investigations-report.pdf>.

W3C. **XML Signature Syntax and Processing Version 1.1**. [S.l.], 2013.