BlindRevoke: Um Mecanismo de Revogação para SSI*

Edimar Veríssimo da Silva [1], Arlindo F. da Conceição [1]

[1]Instituto de Ciência e Tecnologia (ICT – UNIFESP). Av. Cesare Monsueto Giulio Lattes, 1201 – Eugênio de Melo, São José dos Campos – SP, 12247-014

{arlindo.conceicao,edimar.verissimo}@unifesp.br

Abstract. *This work proposes BlindRevoke, a revocation mechanism for* Self-Sovereign Identity *that seeks to reduce the temporal tracking of the* Holder *by restricting revocation verification to the window explicitly authorized by the* Holder. *The solution combines a public vector in the* ledger*, temporal tokens under user control, Merkle tree proof, and* off-ledger *Bloom filters anchored by a Manifest. The proposal was implemented in Hyperledger Indy, and the results indicate initial viability in a controlled environment, with lightweight verification and low* ledger *usage in the evaluated design. The BlindRevoke architecture depends on* off-ledger *auxiliary infrastructure and may increase the credential size in scenarios with many temporal windows.*

Resumo. *Este trabalho propõe o BlindRevoke, um mecanismo de revogação para Identidade Autossoberana que busca reduzir o rastreamento temporal do titular ao restringir a verificação de revogação à janela explicitamente autorizada pelo* Holder*. A solução combina vetor público no* ledger*, tokens temporais sob controle do usuário, prova por árvore de Merkle e filtros de Bloom* off-ledger *ancorados por um Manifesto. A proposta foi implementada no Hyperledger Indy e os resultados indicam viabilidade inicial em ambiente controlado, com verificação leve e com baixo uso do* ledger *no desenho avaliado. A arquitetura do BlindRevoke depende de infraestrutura auxiliar* off-ledger *e pode aumentar o tamanho da credencial em cenários com muitas janelas temporais.*

1. Introdução

Identidades Autossoberanas (SSI, do inglês *Self-Sovereign Identity*) propõem um modelo de gestão de identidade digital no qual o indivíduo mantém controle direto sobre seus dados, reduzindo a dependência de provedores centralizados [Mazzocca et al. 2025, Sofronie et al. 2025]. Neste paradigma, a confiança é sustentada por infraestruturas descentralizadas, frequentemente baseadas em blockchain ou registros distribuídos, que atuam como âncoras transparentes e imutáveis [Mazzocca et al. 2025, Paredes-García et al. 2025, Khayer et al. 2025].

Sua base técnica combina Identificadores Descentralizados (DIDs, do inglês *Decentralized Identifiers*), que criam identidades persistentes sem permissão externa [Mazzocca et al. 2025, Dayaratne et al. 2026], e Credenciais Verificáveis (VCs, do inglês *Verifiable Credentials*) [Bernstein et al. 2025], equivalentes digitais criptograficamente verificáveis de documentos como diplomas ou licenças [Mallesh 2025,

*Este projeto foi apoiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processos 2023/00783-7 e 2025/13053-2.

Lee et al. 2026, Paredes-García et al. 2025]. Essas credenciais ampliam privacidade, portabilidade e resistência a vazamentos massivos [Lee et al. 2026, von Rauscher 2024, Paredes-García et al. 2025], sendo armazenadas em carteiras digitais sob controle do titular. Nesse modelo, há três papéis principais: o *Issuer* (emissor), que emite a credencial; o *Holder* (titular), que a armazena e apresenta; e o *Verifier* (verificador), que a verifica [Lee et al. 2026].

Apesar dessas vantagens, a revogação continua sendo um dos pontos mais sensíveis desse modelo, pois invalidar documentos perdidos, roubados ou desatualizados sem comprometer a privacidade do usuário permanece um grande desafio [Lee et al. 2026]. Em mecanismos tradicionais, como CRLs e OCSP [Wohlmacher 2000], a possibilidade de o *Verifier* consultar livremente o estado da credencial pode viabilizar vigilância indevida, rastreamento do seu ciclo de vida, inferência de informações sensíveis e correlação de apresentações, em desacordo com o princípio da não-rastreabilidade [Lee et al. 2026, Castaldo et al. 2025, Buccafurri and Licciardi 2025, Hoops et al. 2025]. Esse problema é relevante em contextos como relações de trabalho, serviços de saúde e validação de diplomas, nos quais a simples observação do status de revogação já pode revelar informações indevidas sobre o titular [Lee et al. 2026, Hoops et al. 2025, Mazzocca et al. 2025, Malamas et al. 2020, Tan-Vo et al. 2025].

As principais contribuições deste artigo são: (i) um mecanismo de revogação com autorização temporal explícita pelo *Holder*; (ii) uma implementação viável no ecossistema Hyperledger Indy; e (iii) uma avaliação inicial de custos práticos em termos de emissão, revogação, verificação e tamanho dos pacotes.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta as definições básicas das principais siglas e conceitos necessários para entender o texto; a Seção 3 apresenta os trabalhos relacionados; a Seção 4 descreve o algoritmo BlindRevoke; a Seção 5 reúne a implementação, a avaliação teórica e os resultados experimentais; a Seção 6 resume as limitações do trabalho; a Seção 7 apresenta a conclusão; e a Seção 8 registra o uso de Inteligência Artificial Generativa.

2. Definições utilizadas neste artigo

No contexto deste trabalho, Identidade Autossoberana (SSI, do inglês *Self-Sovereign Identity*) designa o modelo de identidade descentralizada em que o usuário mantém controle sobre seus dados [Lee et al. 2026]; Identificador Descentralizado (DID, do inglês *Decentralized Identifier*) é o identificador criptográfico descentralizado usado nesse ecossistema [Sporny et al. 2022]; e Credencial Verificável (VC, do inglês *Verifiable Credential*) corresponde à credencial padronizada [Mazzocca et al. 2025] pelo W3C (*World Wide Web Consortium*), organismo responsável por especificações centrais da Web e também das credenciais e identificadores descentralizados [Mazzocca et al. 2025].

A infraestrutura que sustenta esse modelo pode (mas não precisa) usar uma Tecnologia de Livro-Razão Distribuído (DLT, do inglês *Distributed Ledger Technology*) [Khayer et al. 2025], isto é, um livro-razão distribuído ou *ledger*, como o Hyperledger Indy, plataforma voltada à identidade descentralizada [Khayer et al. 2025, Paredes-García et al. 2025], e a VON Network, que é uma rede Indy usada principalmente para desenvolvimento e testes. Nesse ambiente, ATTRIB é um tipo de transação do Indy para associar atributos adicionais a um DID, enquanto *Issuer*, *Holder* e *Verifier*

representam, respectivamente, a entidade que emite a credencial, o titular que a armazena e apresenta, e a parte que a verifica [Lee et al. 2026].

O algoritmo SHA3-256 é uma função de *hash* criptográfico usada para produzir resumos fixos e resistentes à adulteração; Árvores de Merkle organizam hashes em estrutura hierárquica para comprovação eficiente de integridade [Tesei et al. 2023]; e o Bloom Filter é uma estrutura probabilística e compacta para testes rápidos de pertencimento a conjuntos [Kuzmanovic 2025]. No mecanismo de revogação proposto, o Vetor K é o conjunto público de valores aleatórios publicado pelo emissor para sustentar a revogação das credenciais.

3. Trabalhos relacionados

Na WebPKI (*Web Public Key Infrastructure*), infraestrutura de chaves públicas usada na Web, mecanismos clássicos de revogação ajudam a evidenciar problemas que reaparecem, sob outra arquitetura, no contexto de credenciais verificáveis. CRLs publicam listas assinadas de certificados revogados, mas sofrem com crescimento e custo de distribuição. OCSP reduz esse custo por meio de consultas online de status, mas introduz risco de rastreamento do usuário pelo padrão de consultas [Kwon et al. 2026, Wohlmacher 2000]. O CRLite, por sua vez, usa filtros de Bloom para compactar o estado de revogação e permitir verificações locais [Kwon et al. 2026]. Assim, a WebPKI é relevante para este trabalho principalmente por três lições: listas públicas têm custo de escala, consultas online podem expor metadados temporais, e estruturas probabilísticas podem reduzir custo quando combinadas com ancoragem verificável.

Esses mecanismos da WebPKI são usados aqui como analogia histórica e de privacidade, não como equivalência estrita entre certificados digitais clássicos e Credenciais Verificáveis baseadas em DID. O ponto transferido para este trabalho é o risco de rastreabilidade criado por consultas de status de revogação. Não se transfere, porém, a mesma arquitetura de autoridade, o mesmo modelo de emissão nem a relação hierárquica típica da WebPKI.

Com o avanço da Identidade Autossoberana e das infraestruturas baseadas em Tecnologia de Livro-Razão Distribuído, o problema da revogação passou a ser reavaliado sob a ótica do controle do usuário, dos Identificadores Descentralizados e das Credenciais Verificáveis [Lee et al. 2026, Sofronie et al. 2025]. Nesse cenário, o W3C padronizou a Lista de Status em Bitstring (BSL, do inglês *Bitstring Status List*), que representa o estado da credencial em uma matriz de bits comprimida e permite consulta $O(1)$ [Lee et al. 2026]. Em paralelo, trabalhos em redes veiculares propuseram substituir CRLs pesadas por registros leves em blockchain e árvores de Merkle, reduzindo drasticamente a latência de revogação sem interromper o fluxo de dados [Tesei et al. 2023].

Apesar dos avanços, a publicidade dos registros de revogação ainda favorece a possibilidade de correlação, o que motivou o uso de acumuladores criptográficos capazes de condensar conjuntos revogados e oferecer provas de não pertencimento com conhecimento zero, embora ao custo de maior complexidade computacional e atualização de *witnesses* [Lee et al. 2026, Kuzmanovic 2025]. Para mitigar essas limitações, surgiram propostas com organização por épocas e estruturas híbridas [Schumm et al. 2023], além de mecanismos voltados à proteção de metadados, como o CRSet, a LVVC e abordagens com VRFs, que buscam reduzir padrões observáveis e rastreabilidade

[Hoops et al. 2025, Lee et al. 2026, Papathanasiou and Polyzos 2024].

A literatura mais recente também tem explorado desempenho operacional e escalabilidade. *Optimistic Rollups* foram propostos para processar lotes massivos de revogações fora da cadeia principal, reduzindo custo na blockchain e mantendo validação agregada [Tan-Vo et al. 2025]. Em outra frente, técnicas de Inteligência Artificial e *Machine Learning*, como XGBoost, têm sido usadas para detecção de anomalias e mitigação proativa de fraudes no processo de revogação [Tan-Vo et al. 2025, Vikram 2023]. Há ainda propostas que combinam filtros probabilísticos com bancos locais, como SQLite, permitindo respostas determinísticas no ambiente do verificador e reservando o custo criptográfico para casos excepcionais [Kuzmanovic 2025].

O zkRevoke [Manimaran et al. 2025] propõe uma verificação contínua limitada no tempo, na qual o *Holder* define, em cada apresentação, por quantas épocas o *Verifier* poderá consultar a revogação. A solução adota uma *blacklist* pública de tokens associados a credenciais revogadas e usa provas de conhecimento zero, especificamente Groth16, para que o titular prove que seus tokens não pertencem à lista sem revelar a semente da credencial. Com isso, o *Verifier* pode checar a revogação apenas dentro do período autorizado, enquanto o *Issuer* não observa a verificação quando a lista é obtida por um registro público. O custo principal desloca-se para a geração e verificação das provas e para a atualização periódica da *blacklist* pelo *Issuer*. Em contrapartida, o mecanismo reduz a largura de banda do *Holder* em comparação com abordagens baseadas em atualização contínua de *witnesses*.

A proposta baseada em Criptografia Baseada em Identidade Hierárquica Anônima (AHIBE) [Buccafurri and Licciardi 2025] também busca limitar temporalmente o acesso do *Verifier* ao estado de revogação. Nessa abordagem, o *Holder* entrega ao verificador chaves privadas temporais e valores derivados de PRNG correspondentes às janelas autorizadas, permitindo que o *Verifier* localize e decifre apenas a informação de revogação daquele período. A privacidade decorre da anonimidade da AHIBE e do fato de o *Issuer* publicar tabelas de verificação/revogação sem receber consultas específicas de apresentação. O custo para o *Holder* é baixo no momento da apresentação, mas o *Issuer* assume tarefas periódicas de atualização criptográfica das tabelas, enquanto o *Verifier* pode precisar baixar segmentos dessas tabelas e a tabela de revogação; além disso, a solução depende de uma infraestrutura AHIBE/PKG confiável, ainda que possa ser distribuída ou baseada em criptografia de limiar, na qual a geração de chaves depende da cooperação de um subconjunto mínimo de autoridades. A Tabela 1 compara os mecanismos segundo controle temporal pelo *Holder*, custo para *Holder* e *Verifier*, peso no *ledger*, dependência *off-ledger* e tipo de garantia criptográfica predominante.

4. O Algoritmo: BlindRevoke

O BlindRevoke modela a revogação em SSI como uma consulta previamente autorizada pelo *Holder*. Em vez de permitir que o *Verifier* consulte livremente o histórico da credencial, o protocolo restringe a verificação ao intervalo para o qual recebeu material criptográfico suficiente. Para isso, a arquitetura combina um vetor público registrado no *ledger*, tokens temporais sob controle do titular, prova por árvore de Merkle e *bloom filters off-ledger* ancorados por um arquivo Manifesto. A seguir, descrevemos os elementos do protocolo e os fluxos de emissão, apresentação, verificação e revogação.

Tabela 1. Comparação qualitativa entre mecanismos de revogação de credenciais verificáveis

Mecanismo	**Controle/ Flexibilidade Temporal pelo Holder**	**Custo Computacional (Holder)**	**Custo Computacional/ Banda (Verifier)**	**Peso no *Ledger***	**Dependência *Off-Ledger***
BSL [Lee et al. 2026]	Baixo	Baixo	Baixo–Médio	Moderado	Alta
Acumuladores [Lee et al. 2026]	Baixo (salvo ZKP)	Alto (atualização de *witness*)	Médio–Alto	Muito baixo	Alta
LVVC [Lee et al. 2026]	Médio	Alto	Baixo	Muito baixo	Altíssima
CRSet [Hoops et al. 2025]	Baixo	Baixo	Baixo–Médio	Moderado	Baixa–Média
AHIBE [Buccafurri and Licciardi 2025]	Alto	Muito baixo	Médio (banda)	Baixo–Médio	Moderada
zkRevoke [Manimaran et al. 2025]	Alto	Médio (ZKP)	Médio	Moderado	Baixa–Média
BlindRevoke (proposta)	**Alto**	**Médio**	**Baixo–Médio**	**Baixo**	**Moderada–Alta**

Na Tabela 1, soluções como BSL e CRSet privilegiam simplicidade operacional, mas oferecem menor controle temporal ao *Holder*; acumuladores e LVVC reduzem o peso no *ledger*, porém deslocam custo para provas, *witnesses* ou infraestrutura auxiliar. AHIBE e zkRevoke são os pares mais próximos por darem alto controle temporal ao titular: AHIBE usa chaves temporais e anonimidade criptográfica, com maior custo recorrente no *Issuer* e banda no *Verifier*; zkRevoke usa tokens por época e provas de conhecimento zero, deslocando custo para provas e atualização de *blacklist*. O BlindRevoke busca um ponto intermediário, com baixo peso no *ledger*, verificação local eficiente e garantias baseadas em compromissos temporais e filtros de Bloom ancorados.

A Figura 1 resume as etapas de emissão, autorização temporal e verificação, mostrando como o *Verifier* valida a prova de Merkle, consulta o *ledger* e usa o Manifesto para localizar o *bloom filter* da janela autorizada. Não é necessário que o *Issuer* disponibilize uma API para permitir a verificação de revogação. Para maior privacidade, o *Issuer* pode disponibilizar os arquivos estáticos e o *Verifier* pode ter um software semelhante à API para consultar localmente o status de uma credencial.

A Tabela 2 resume a notação do protocolo, separando dados assinados na credencial, dados publicados no *ledger* e material sob controle do *Holder*.

Tabela 2. Notação básica do protocolo BlindRevoke

Símbolo / elemento	**Significado**	**Natureza**	**Estado após a emissão**
`seed` (atributo de controle gravado na credencial)	identificador aleatório de 256 bits da credencial, usado na derivação de REVOG_i	público e assinado	mantido na credencial
`start_time`, `unit_of_time`, `time_window`	parâmetros que definem o particionamento temporal da credencial em janelas	públicos e assinados	mantidos na credencial
K	vetor público do emissor, registrado no *ledger*, usado como base para derivar compromissos temporais	público e persistente	mantido no *ledger*
tmp_i	valor aleatório associado à janela i, usado na construção de L_i	privado	entregue ao *Holder*
L_i	compromisso temporal verificável da janela i	derivado e verificável	reconstruível a partir do material da janela
$L = \{L_1, \ldots, L_n\}$	conjunto de compromissos temporais da credencial	derivado	organizado em árvore de Merkle
S_i	sequência específica de 32 elementos distintos de K, mais tmp_i	privado	entregue ao *Holder*
S	conjunto dos materiais brutos de todas as janelas	privado	entregue ao *Holder*
T	forma compactada de S, usada na implementação para reduzir armazenamento	privado	entregue ao *Holder*
`root_merkle_L`	raiz da árvore de Merkle construída sobre L	público e assinado	mantido na credencial
REVOG_i	chave de revogação da janela i, consultada no Bloom filter	derivada	calculada quando necessário
Manifesto	índice mestre que referencia os Bloom filters *off-ledger* e cuja integridade é ancorada no *ledger*	público	URL e *hash* ancorados no *ledger*

Como sintetiza a Tabela 2, os atributos de controle e `root_merkle_L` são públicos e assinados na credencial; K e a âncora do Manifesto permanecem públicos no *ledger*. Os parâmetros tmp_i, S, S_i e T compõem o material temporal sob controle do *Holder*, usado para autorizar a consulta de revogação apenas nas janelas explicitamente reveladas.

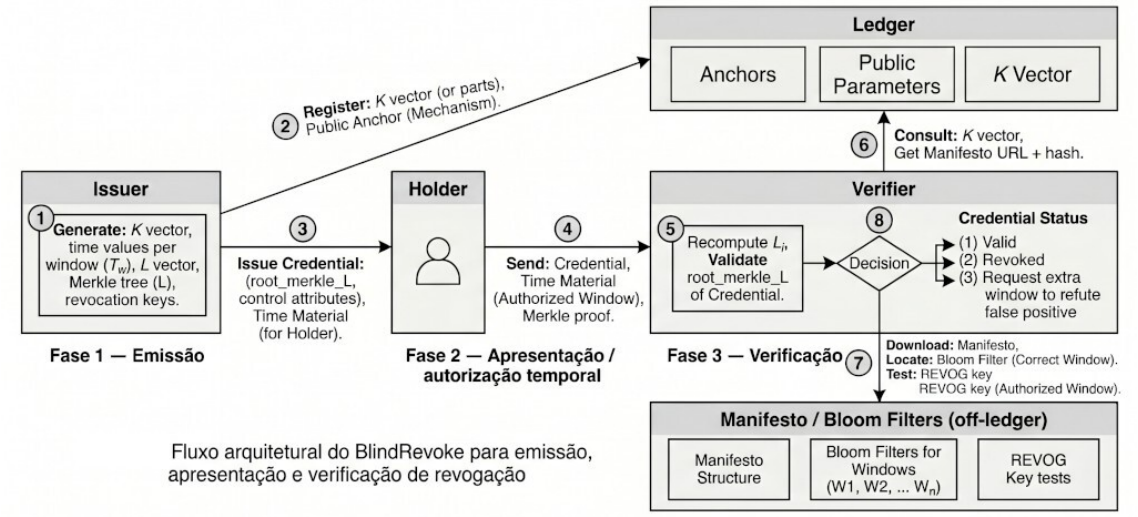

Figura 1. Fluxo arquitetural do algoritmo BlindRevoke

Observação: Passo 1: o *Issuer* gera o vetor K, os valores temporais por janela, o vetor L, a árvore de Merkle e as chaves de revogação. Passo 2: o *Issuer* registra no *ledger* o vetor K, ou suas partes, e a âncora pública do mecanismo. Passo 3: o *Issuer* emite a credencial ao *Holder* com `root_merkle_L`, atributos de controle e material temporal. Passo 4: o *Holder* envia ao *Verifier* a credencial, o material temporal da janela autorizada e a prova de Merkle. Passo 5: o *Verifier* recompõe L_i e valida a raiz `root_merkle_L` assinada na credencial. Passo 6: o *Verifier* consulta o *ledger* para obter o vetor K, a URL do Manifesto e seu *hash*. Passo 7: o *Verifier* baixa o Manifesto, localiza o *bloom filter* correto e testa a chave REVOG da janela autorizada. Passo 8: o *Verifier* decide se a credencial está válida, revogada ou se precisa de janela extra para refutar falso positivo.

4.1. Modelo de ameaça e garantias

O modelo de ameaça considera, em primeiro lugar, um *Verifier* curioso ou malicioso. O *Verifier* curioso segue o protocolo, mas tenta extrair mais informação que a necessária, como consultar janelas passadas, futuras ou não autorizadas. O *Verifier* malicioso pode ainda tentar reutilizar material recebido, alterar a ordem dos elementos revelados ou correlacionar apresentações. O BlindRevoke mitiga esses ataques ao exigir que cada consulta seja vinculada ao material temporal revelado pelo *Holder* e à prova de Merkle correspondente.

Também consideramos um *Issuer* honesto-mas-curioso, infraestrutura *off-ledger* indisponível ou maliciosa e observadores de rede. Em conluio com o *Verifier*, a infraestrutura *off-ledger* pode observar metadados como IP, horário de download, Manifesto solicitado, filtro acessado e repetição de consultas, ainda que não consiga alterar silenciosamente o conteúdo validado pelo *hash* ancorado no *ledger*. Assim, a ancoragem protege integridade e detecção de adulteração, mas não garante disponibilidade nem anonimato de acesso.

Ficam fora do escopo desta versão o comprometimento da carteira do *Holder*, a quebra de SHA3-256 ou das assinaturas da credencial, o conluio total entre *Issuer* e *Verifier*, anonimato completo de rede e disponibilidade permanente da infraestrutura *off-ledger*. Assim, as garantias avaliadas concentram-se em limitar a verificação de revogação às janelas explicitamente autorizadas e em detectar adulterações no material público usado na verificação.

4.2. Definições

Para funcionar, o BlindRevoke exige atributos informativos e atributos de controle, como resume a Tabela 3. Os atributos de controle não substituem os dados da credencial: adi-

cionam metadados criptográficos para particionamento temporal, prova de integridade e consulta de revogação, devendo estar previstos no esquema usado na emissão.

Tabela 3. Estrutura de uma credencial revogável

	Atributos	Descrição do atributo	Valor do Atributo
Atributos Informativos	A1	Descrição 1	Valor 1
	A2	Descrição 2	Valor 2
	A3	Descrição 3	Valor 3
	A4	Descrição 4	Valor 4
Atributos de Controle	A5	`seed`	32 bytes aleatórios (256 bits)
	A6	`start_time`	DD/MM/YYYY (date)
	A7	`unit_of_time`	1 = seconds; 2 = minutes; 3 = hours; 4 = days; 5 = weeks; 6 = months; 7 = years; 8 = decades
	A8	`time_window`	Unidade de tempo
	A9	`root_merkle_L`	32 bytes (um *hash*)

Os atributos de controle `seed`, `start_time`, `unit_of_time`, `time_window` e `root_merkle_L` devem estar presentes em todas as credenciais revogáveis. Esses atributos são públicos, assinados pelo emissor e usados para particionar a credencial em janelas temporais e permitir a validação do material revelado pelo *Holder*.

4.3. O Vetor K

O vetor K é composto por 1024 valores aleatórios de 256 bits, publicados pelo emissor no *ledger*. Esses valores **não são secretos**; eles funcionam como uma base pública verificável a partir da qual são derivados os compromissos temporais usados pelo protocolo. O vetor K é publicado uma única vez por emissor.

Cada janela temporal do protocolo é construída a partir de uma sequência específica de 32 elementos distintos extraídos de K. Como esses valores são concatenados antes da aplicação da função *hash*, tanto a escolha quanto a ordem dos elementos influenciam diretamente o resultado. O número de arranjos possíveis é dado por $(1024!/(1024-32)!) = 1{,}309 \times 10^{96}$, o que torna impraticável a busca exaustiva pelos subconjuntos utilizados, especialmente quando essa estrutura é combinada aos valores temporários tmp_i e à aplicação de SHA3-256. Assim, o papel de K não é atuar como segredo, mas como base pública verificável para a derivação dos compromissos temporais usados pelo protocolo.

4.4. O Vetor L

O vetor L é formado pelos compromissos temporais da credencial. Para cada janela temporal i, o emissor seleciona uma sequência de 32 elementos distintos do vetor K, não necessariamente em ordem crescente, e a combina com um valor aleatório temporário tmp_i. O compromisso L_i é então calculado por:

$$L_i = Hash(k_{i,1} \mid k_{i,2} \mid \cdots \mid k_{i,32} \mid tmp_i) \tag{1}$$

A coleção $\{L_1, L_2, L_3, \ldots, L_n\}$ define, portanto, as janelas temporais nas quais o *Holder* poderá autorizar a consulta do estado de revogação da credencial. Esses valores são organizados em uma árvore de Merkle, cuja raiz `root_merkle_L` é incorporada à credencial como atributo de controle assinado pelo emissor.

Para cada janela i, o emissor também entrega ao *Holder* o material necessário para reconstruir o respectivo compromisso temporal, aqui representado por:

$$S_i = [k_{i,1}, k_{i,2}, \ldots, k_{i,32}, tmp_i] \tag{2}$$

e, de forma agregada,

$$S = [S_1, S_2, S_3, \ldots, S_n] \tag{3}$$

Assim, L_i funciona como o compromisso verificável da janela i, enquanto S_i corresponde à informação que permite ao *Holder* autorizar sua revelação ao *Verifier* no momento da apresentação. Em um cenário com 244 janelas temporais, por exemplo, o vetor L ocupa 7808 bytes, enquanto o vetor S possui custo significativamente maior. Esse custo aparece principalmente na emissão e no armazenamento local pelo *Holder*; na apresentação, apenas as janelas autorizadas precisam ser reveladas, e sua compactação é tratada na seção de implementação por meio do vetor T.

4.5. O processo de emissão de credenciais

No processo de emissão, o emissor primeiro define a validade da credencial e, a partir dela, o número de janelas temporais de revogação que serão suportadas (se a credencial não tiver validade, basta omitir os atributos de controle). Para cada janela i, o emissor seleciona 32 elementos do vetor K, gera o valor temporário tmp_i e calcula o compromisso correspondente L_i. Em seguida, organiza o conjunto $L = \{L_1, L_2, L_3, \ldots, L_n\}$ em uma árvore de Merkle e incorpora a raiz `root_merkle_L` à credencial como atributo de controle assinado. O emissor, então, entrega ao *Holder* a credencial e o material necessário para autorizar consultas de revogação nas janelas previstas, preservando com o titular o controle sobre quando esse material será revelado ao *Verifier*.

Após a emissão, o emissor pode descartar os vetores auxiliares que não sejam necessários para o processo de revogação, mantendo apenas os dados estritamente requeridos para registrar futuras revogações. Com isto queremos dizer que o emissor não precisa manter o vetor L ou os valores temporários tmp_i para realizar a revogação, mas apenas o vetor K e a estrutura de janelas (chaves REVOG já calculadas), o que reduz significativamente a carga de armazenamento e processamento do emissor.

4.6. O processo de verificação de credenciais

No processo de verificação, o *Holder* apresenta ao *Verifier* a credencial, o material correspondente à janela temporal autorizada e a respectiva prova de Merkle. Com esses elementos, o *Verifier* recalcula o compromisso temporal L_i da janela apresentada e recompõe a raiz da árvore, verificando se o resultado coincide com o atributo `root_merkle_L` assinado na credencial. Em seguida, o *Verifier* consulta no *ledger* o vetor público K do emissor para confirmar que os 32 valores informados pertencem ao conjunto válido do protocolo e são distintos entre si. A verificação não exige que esses valores estejam em ordem crescente; ao contrário, a ordem apresentada faz parte do compromisso criptográfico da janela, pois a inversão ou permutação dos elementos altera completamente o resultado da função de *hash*. Assim, o *Verifier* deve preservar a sequência $k_{i,1}, k_{i,2}, \ldots, k_{i,32}$ recebida do *Holder* ao recalcular L_i e REVOG_i.

Após essa validação, o *Verifier* obtém no *ledger* o endereço e o *hash* do arquivo Manifesto, recupera sua versão completa e, por meio dele, localiza o *bloom filter* aplicável à credencial. A chave de revogação da janela i é então calculada a partir desta equação:

$$\mathrm{REVOG}_i = \mathrm{Hash}(Seed \mid time_i \mid k_{i,1} \mid k_{i,2} \mid \cdots \mid k_{i,32}) \tag{4}$$

em que $time_i$ representa o identificador temporal da janela autorizada e $Seed$ representa o identificador da credencial. O *Verifier* só consegue efetuar a consulta da janela para a qual recebeu do *Holder* o material necessário à reconstrução do respectivo compromisso temporal.

Dessa forma, o protocolo restringe a verificação de revogação à janela explicitamente autorizada pelo *Holder*, impedindo, no modelo proposto, consultas arbitrárias sobre janelas passadas, futuras ou não reveladas. A análise da probabilidade de falsos positivos do *bloom filter* e o uso de janelas adicionais de confirmação são apresentados na Seção 4.8.

4.7. O arquivo Manifesto

O arquivo Manifesto funciona como índice mestre da infraestrutura *off-ledger* de revogação: em vez de registrar diretamente os *bloom filters* no *ledger*, o *Issuer* grava, via transação ATTRIB associada ao seu DID, apenas a URL e o *hash* do Manifesto, permitindo que o *Verifier* recupere esse arquivo, identifique os filtros relevantes e confira sua integridade com base no *hash* público ancorado no *ledger*. Dessa forma, a arquitetura separa armazenamento e integridade, oferecendo um mecanismo mínimo de versionamento e consistência entre o conteúdo externo e sua âncora pública, o que reduz a carga sobre a blockchain e mantém a revogação leve e modular; por outro lado, embora essa dependência externa não comprometa a integridade do mecanismo, ela ainda pode afetar sua disponibilidade operacional. Esses riscos podem ser mitigados operacionalmente com arquivos estáticos cacheáveis, espelhos ou CDN (*Content Delivery Network*), download antecipado dos filtros, consulta local e uso de Tor ou proxy quando o cenário exigir maior proteção de metadados.

4.8. Estatísticas sobre filtros de Bloom

No BlindRevoke, os filtros de Bloom mantêm a verificação leve e local. Com filtro padrão de 2 MB ($m = 16.777.216$ bits) e limite operacional de 95%, cada filtro suporta 345.238 entradas, ou cerca de 945 credenciais com 365 janelas cada. A Tabela 4 mostra que dobrar o filtro dobra aproximadamente a capacidade de revogação.

Tabela 4. Capacidade estimada de filtros de Bloom para diferentes tamanhos

Filtro de Bloom (tamanho)	Capacidade (95%)	Número de credenciais para revogar (considerando 365 janelas por credencial)
2 MB	345.238	945
4 MB	690.477	1.891
8 MB	1.380.953	3.783
16 MB	2.761.906	7.566
32 MB	5.523.813	15.133

A adoção de filtros de Bloom não visa apenas economizar espaço, mas também viabilizar consultas rápidas mesmo em dispositivos com recursos limitados. Operacionalmente, a minimização de falsos positivos combina quatro decisões: dimensionar o filtro por m, k e n; impor limite conservador de ocupação; rotacionar filtros via Manifesto quando esse limite é atingido; e solicitar janelas extras de confirmação quando houver indicação de revogação. Nesse contexto, o efeito probabilístico relevante é assimétrico: uma credencial revogada não é aceita como válida, mas uma credencial válida

pode, com baixa probabilidade, ser classificada como revogada, comportamento cuja estimativa segue a formulação clássica da taxa de falso positivo (PFP) de filtros de Bloom [Christensen et al. 2010].

$$p_{FP}(m,k,n) = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \tag{5}$$

Neste trabalho, $m = 16.777.216$, $n = 345.238$ e $k = 32$, resultando em $P_{FP} \approx 7{,}38 \times 10^{-11}$, ou cerca de 1 falso positivo em 13,55 bilhões de consultas. A Tabela 5 quantifica essa política: cada janela extra acrescenta uma consulta independente ao filtro e reduz exponencialmente a chance de classificar uma credencial válida como revogada; por isso, o algoritmo adota 10 janelas adicionais como parâmetro conservador.

Tabela 5. Probabilidade residual de falso positivo com janelas extras

Janelas Extras	Consultas Totais	Probabilidade de erro	Chances de se encontrar um falso positivo
1	2	$5{,}447 \times 10^{-21}$	1 em $1{,}836 \times 10^{20}$
2	3	$4{,}020 \times 10^{-31}$	1 em $2{,}487 \times 10^{30}$
3	4	$2{,}967 \times 10^{-41}$	1 em $3{,}370 \times 10^{40}$
4	5	$2{,}190 \times 10^{-51}$	1 em $4{,}566 \times 10^{50}$
5	6	$1{,}616 \times 10^{-61}$	1 em $6{,}187 \times 10^{60}$
6	7	$1{,}193 \times 10^{-71}$	1 em $8{,}382 \times 10^{70}$
7	8	$8{,}805 \times 10^{-82}$	1 em $1{,}136 \times 10^{81}$
8	9	$6{,}498 \times 10^{-92}$	1 em $1{,}539 \times 10^{91}$
9	10	$4{,}796 \times 10^{-102}$	1 em $2{,}085 \times 10^{101}$
10	11	$3{,}540 \times 10^{-112}$	1 em $2{,}825 \times 10^{111}$

Observação: A coluna "Consultas Totais" indica o número de consultas ao *bloom filter* para verificar a revogação, somando a janela autorizada e as janelas extras.

Se o falso positivo persistir após as janelas extras previstas, o caso deve ser tratado operacionalmente por nova apresentação com janelas adicionais ou, em último caso, por reemissão da credencial.

Embora alguns autores [Christensen et al. 2010] observem que a fórmula clássica não é exata em todos os cenários, seu erro relativo diminui em filtros maiores e com razão m/n favorável. Como o mecanismo proposto usa filtros grandes e ocupação conservadora, essa aproximação foi considerada adequada para o dimensionamento e coerente com os resultados experimentais do protótipo.

5. Implementação e Avaliação

Esta seção apresenta a validação prática e analítica do mecanismo BlindRevoke. Inicialmente, detalham-se os aspectos de implementação sobre a plataforma Hyperledger Indy e as estratégias de armazenamento. Na sequência, são abordadas as propriedades teóricas de segurança e privacidade do protocolo, seguidas pela avaliação experimental dos tempos de execução e do tamanho dos pacotes transmitidos.

5.1. Implementação

A implementação no Hyperledger Indy segue uma estratégia pragmática para contornar o limite de armazenamento do *ledger* sem perder verificabilidade: o vetor K é

gravado uma única vez por meio de transações ATTRIB vinculadas ao DID do emissor. Como esse vetor tem cerca de 32 KB e excede o limite aproximado de 4 KB por transação, ele é particionado e registrado sequencialmente na blockchain, consumindo 11 transações devido à expansão da codificação em base 64 (esses valores foram observados empiricamente no protótipo implementado sobre a VON Network local, e podem variar conforme configuração da rede Indy); em testes na rede VON Network local, esse processo levou cerca de 40 segundos, mas ocorre apenas na fase inicial de configuração. Já a revogação em si permanece *off-ledger*, de modo que a blockchain atua apenas como âncora leve, prática recomendada para contornar gargalos de armazenamento [Mazzocca et al. 2025, Lee et al. 2026], ao armazenar a URL e o *hash* do arquivo Manifesto, que referencia externamente os filtros de Bloom.

Essa arquitetura reduz a carga no *ledger* e permite verificação local de revogação com baixo custo, usando primitivas padronizadas como SHA3-256 e estruturas probabilísticas amplamente disponíveis (*Bloom filters*), evitando mecanismos mais complexos como PKG, AHIBE e acumuladores criptográficos [Buccafurri and Licciardi 2025, Hoops et al. 2025, Lee et al. 2026, Schumm et al. 2023]. O crescimento do material temporal é mitigado pelo vetor T, que compacta S de cerca de 252 KB para aproximadamente 18 KB ao substituir valores por índices relativos a K; assim, a credencial verificável somada aos vetores L e T totaliza cerca de 25 KB em um cenário com 244 janelas e segue viável mesmo com 1000 janelas. Esse custo ocorre na emissão e no armazenamento local, enquanto a apresentação pode usar entrega parcial de janelas. Em conjunto, essas escolhas combinam baixo peso no *ledger*, verificação local rápida e manutenção modular via Manifesto, ainda que com dependência de infraestrutura auxiliar *off-ledger* para distribuição dos filtros.

5.2. Avaliação teórica

A robustez e a privacidade do vetor L decorrem do fato de que cada compromisso temporal L_i combina uma sequência específica de elementos públicos de K com um valor aleatório secreto tmp_i, tornando impraticável sua reconstrução sem o material autorizado pelo *Holder*; ao mesmo tempo, o protocolo limita a exposição de metadados temporais porque o *Verifier* recebe apenas a janela autorizada e a prova mínima de Merkle necessária para validá-la, sem acesso às demais janelas, enquanto janelas extras de confirmação ajudam a reduzir o impacto de falsos positivos do *bloom filter*. Além disso, a revogação minimiza a custódia de informações privadas pelo emissor, pois a chave REVOG_i pode ser calculada sem depender de tmp_i, permitindo registrar a revogação nos filtros a partir da janela corrente até a final e preservar a verificabilidade pública da credencial.

5.3. Avaliação experimental

Esta seção apresenta testes automatizados em ambiente controlado sobre Hyperledger Indy, com VON Network e API do *bloom filter*. A avaliação mediu emissão, revogação, verificação e tamanho dos pacotes para credenciais revogáveis com 3 atributos, em notebook 12th Gen Intel Core i5-12450H, 16GB de RAM e Linux Mint 22.2 Zara. Para reprodução dos testes: `https://github.com/yugi386/info_congressos/blob/main/reproduzir_testes.pdf`.

A Tabela 6 mostra que mais janelas aumentam precisão temporal, mas ampliam o tempo de emissão e o armazenamento no *Holder*.

Tabela 6. Emissão da Credencial Revogável

Janelas de Tempo	Tempo para Emitir (ms)	Tamanho da credencial (Kbytes)
100	73,357	78,847
365	174,47	198,759
1000	401,524	486,047
5000	1.839,269	2.295,641
10000	3.660,759	4.557,942

A emissão permanece abaixo de 1 segundo até 1000 janelas. A Tabela 7 mostra que revogações precoces exigem mais escritas no *bloom filter*, pois todas as janelas posteriores também são marcadas como revogadas.

Tabela 7. Tempo de Revogação da Credencial pelo Emissor

Janelas	Cenário	Janela de Revogação	Tempo de revogação (ms)	Chaves escritas no bloom filter
100	`revoked_early`	0	70,57	110
100	`revoked_middle`	50	69,176	60
100	`revoked_late`	99	62,776	11
365	`revoked_early`	0	89,192	375
365	`revoked_middle`	182	79,461	193
365	`revoked_late`	364	62,947	11
1000	`revoked_early`	0	144,27	1010
1000	`revoked_middle`	500	101,282	510
1000	`revoked_late`	999	62,151	11
5000	`revoked_early`	0	409,238	5010
5000	`revoked_middle`	2500	226,723	2510
5000	`revoked_late`	4999	85,892	11
10000	`revoked_early`	0	713,226	10010
10000	`revoked_middle`	5000	393,613	5010
10000	`revoked_late`	9999	77,058	11

Ainda assim, a revogação permanece na escala de milissegundos. A Tabela 8 mostra mediana, P95 e janelas consultadas, evidenciando que a verificação não percorre linearmente todas as janelas.

A latência cresce com o total de janelas, mas permanece em milissegundos. As Tabelas 9 e 10 comparam envio completo e entrega parcial, cenário recomendado por revelar apenas as janelas necessárias.

Na Tabela 9, a apresentação completa cresce rapidamente. Com entrega parcial, como mostra a Tabela 10, o pacote passa a depender das janelas reveladas, não do total de janelas da credencial.

Desse modo, no ambiente experimental avaliado, o BlindRevoke mostrou-se viável e eficiente para os cenários testados, reduzindo os custos de comunicação quando se utiliza entrega parcial de janelas. Contudo, apresenta um compromisso importante: o número de janelas temporais aumenta o tamanho da credencial e dos pacotes de apresentação. Por este motivo, o horizonte temporal e a política de entrega devem ser decisões centrais no planejamento do projeto.

6. Limitações deste trabalho

Embora os resultados indiquem viabilidade prática, a avaliação ainda possui escopo controlado. Os experimentos foram executados em ambiente automatizado sobre Hyperledger Indy, com credenciais de 3 atributos e foco em emissão, revogação, verificação

Tabela 8. Tempo total de latência de verificação de revogação (10 execuções)

Janela de tempo	Cenário	Janela de revogação	Mediana (ms)	P95 (ms)	Janelas consultadas
100	not_revoked	-	87,97	100,588	18
100	revoked_early	0	53,683	62,565	12
100	revoked_middle	50	44,853	53,416	11
100	revoked_late	99	18,853	26,264	8
365	not_revoked	-	109,659	137,634	20
365	revoked_early	0	84,952	95,917	16
365	revoked_middle	182	28,839	36,092	10
365	revoked_late	364	29,416	34,116	10
1000	not_revoked	-	146,08	169,982	21
1000	revoked_early	0	127,504	146,873	18
1000	revoked_middle	500	119,884	124,048	17
1000	revoked_late	999	51,076	59,83	11
5000	not_revoked	-	407,933	427,59	24
5000	revoked_early	0	413,395	431,292	24
5000	revoked_middle	2500	406,02	420,617	23
5000	revoked_late	4999	208,975	216,452	14
10000	not_revoked	-	724,525	746,017	25
10000	revoked_early	0	762,777	789,853	26
10000	revoked_middle	5000	722,821	734,169	25
10000	revoked_late	9999	403,297	422,139	15

Tabela 9. Tamanho do pacote completo de Apresentação

Janelas válidas	Total de janelas	Tempo de criação (ms)	Payload apresentação (kb)	Envelope criptografado (kb)
1	11	14,402	25,852	35,586
10	20	14,898	40,992	55,777
100	110	19,799	197,640	264,641
365	375	28,133	688,784	919,497
1000	1010	48,347	1.892,615	2.524,607
5000	5010	212,058	10.069,330	13.426,895
10000	10010	464,091	20.621,162	27.496,002
25000	25010	1.196,151	52.726,678	70.303,359

Tabela 10. Tamanho do Pacote de Apresentação com Entrega Parcial de Janelas

Janelas válidas	Janelas entregues ao Verificador	Tempo de criação (ms)	Payload apresentação (kb)	Envelope criptografado (kb)
1	1	14,603	10,151	14,652
10	1	15,233	10,208	14,729
100	1	14,434	10,297	14,849
365	3	14,779	14,056	19,862
1000	10	21,931	27,255	37,462
5000	50	29,737	108,919	146,347
10000	100	40,813	214,305	286,864
25000	250	89,292	534,903	714,327

e tamanho dos pacotes. Essa limitação é mitigada parcialmente pela reprodutibilidade dos testes e deve ser tratada, em trabalhos futuros, com ambientes distribuídos, múltiplos emissores/verificadores e credenciais mais diversas.

Do ponto de vista arquitetural, o mecanismo depende de infraestrutura *off-ledger* para distribuição do Manifesto e dos filtros de Bloom. A integridade é mitigada pelo *hash* ancorado no *ledger*, enquanto disponibilidade e metadados podem ser reduzidos com cache, espelhos/CDN, download antecipado e consulta local. O crescimento do material do *Holder* pode ser mitigado pelo vetor T, pela entrega parcial de janelas e por granularidade temporal adaptada ao caso de uso: credenciais curtas podem usar janelas mais estreitas, enquanto credenciais longas tendem a exigir janelas mais agregadas.

Por fim, o uso de filtros de Bloom introduz falsos positivos residuais. O impacto é mitigado por limite de ocupação, rotação de filtros via Manifesto, janelas extras de confirmação e reemissão da credencial em casos excepcionais. Permanecem como trabalhos futuros a análise formal de conluio, múltiplos verificadores, *rollback* malicioso e vazamento de metadados fora do núcleo criptográfico.

7. Conclusão

Este trabalho apresentou o BlindRevoke, um mecanismo de revogação para SSI concebido para restringir a consulta ao estado de revogação à janela temporal expressamente autorizada pelo *Holder*. Ao articular um vetor público ancorado no *ledger*, compromissos temporais estruturados em árvore de Merkle e filtros de Bloom *off-ledger* referenciados por Manifesto, o modelo reduz a possibilidade de consultas irrestritas ao histórico da credencial e, consequentemente, mitiga riscos de correlação e rastreamento temporal, sem abrir mão da verificabilidade e da auditabilidade.

Do ponto de vista prático, a proposta mostrou viabilidade em ambiente controlado sobre Hyperledger Indy, com tempos aceitáveis de emissão, revogação e verificação, além de baixo peso no *ledger* e checagem local rápida. Ao mesmo tempo, os resultados também evidenciam os principais *trade-offs* da arquitetura: dependência de infraestrutura auxiliar *off-ledger* e crescimento do tamanho da credencial e dos pacotes de apresentação em cenários com muitas janelas temporais. Como continuidade, trabalhos futuros podem aprofundar a avaliação em ambientes distribuídos e de larga escala, bem como avançar na análise formal de segurança e em mecanismos de mitigação para cenários de conluio entre verificadores e falhas na infraestrutura *off-ledger*.

8. Uso de Inteligência Artificial Generativa

Neste trabalho, ferramentas de inteligência artificial foram empregadas no apoio à organização de texto, na elaboração de testes automatizados (com uso do ChatGPT 5.5) e na criação da Figura 1 (com uso do Gemini 3.1 Pro). Todo o conteúdo produzido com esse suporte foi posteriormente revisado e validado pelos autores antes da redação final.

Referências

Bernstein, G., Burnett, D. C., Chadwick, D., Cohen, G., Jones, M. B., Longley, D., and Sporny, M. (2025). Verifiable credentials overview. W3C Group Note, 24 September 2025. World Wide Web Consortium (W3C). Disponível em: `https://www.w3.org/TR/vc-overview/`. Último acesso: 23/04/2026.

Buccafurri, F. and Licciardi, C. (2025). Towards privacy-preserving revocation of verifiable credentials with time-flexibility. In *2025 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 516–521, Venice, Italy.

Castaldo, L., Cortese, G., Izzo, S., and Balsamo, F. (2025). Electronic attestation of attributes extended validation services. In *3rd International Workshop on Trends in Digital Identity (TDI 2025)*, pages 16–25, Bologna, Italy. CEUR-WS.org. CEUR Workshop Proceedings, vol. 3396. Disponível em: `https://ceur-ws.org/Vol-3968/paper1.pdf`.

Christensen, K., Roginsky, A., and Jimeno, M. (2010). A new analysis of the false positive rate of a bloom filter. *Information Processing Letters*, 110(21):944–949.

Dayaratne, T., Rudolph, C., and Yu, J. (2026). Enhancing security and resilience in der integration: A self-sovereign identity approach for smart inverters in virtual power plants. *Distributed Ledger Technologies*. Just Accepted, May 2026.

Hoops, F., Gebele, J., and Matthes, F. (2025). Crset: Private non-interactive verifiable credential revocation. arXiv preprint arXiv:2501.17089v3. Disponível em: `https://arxiv.org/pdf/2501.17089`.

Khayer, B., Mirzaei, S., Alavizadeh, H., and Salehi Shahraki, A. (2025). Blockchain for secure iot: A review of identity management, access control, and trust mechanisms. *IoT*, 6:65.

Kuzmanovic, D. (2025). Design and implementation of efficient credential revocation system. Disponível em: `https://files.ifi.uzh.ch/CSG/staff/schumm/extern/theses/BA-D-Kuzmanovic.pdf`.

Kwon, H., Kim, D., Kim, H., Hahn, C., and Hur, J. (2026). Certificate revocation in the tls ecosystem: A survey. *ACM Computing Surveys*, 58(7):Article 178, 36 pages.

Lee, Y., Shin, H., and Choi, D. (2026). A survey on credential revocation and did deactivation in self-sovereign identity systems. *IEEE Access*, 14:16089–16115.

Malamas, V., Kotzanikolaou, P., Dasaklis, T. K., and Burmester, M. (2020). A hierarchical multi blockchain for fine grained access to medical data. *IEEE Access*, 8:134393–134412.

Mallesh, A. (2025). Immutable identity ledger: Revolutionizing secure credential management with blockchain technology. *International Research Journal of Modernization in Engineering Technology and Science*, 7(3).

Manimaran, P., Raikwar, M., Garrett, T., da Conceição, A. F., Jehl, L., and Vitenberg, R. (2025). zkrevoke: Configurable untraceability for verifiable credentials using zkps. Proceedings on Privacy Enhancing Technologies Symposium (PETS 2026): `https://petsymposium.org/popets/2026/popets-2026-0086.php`.

Mazzocca, C., Acar, A., Uluagac, S., Montanari, R., Bellavista, P., and Conti, M. (2025). A survey on decentralized identifiers and verifiable credentials. *IEEE Communications Surveys & Tutorials*, 27(6):3641–3671.

Papathanasiou, A. M. and Polyzos, G. C. (2024). Privacy-preserving revocation of verifiable credentials with verifiable random functions. In *2024 International Conference on Information Networking (ICOIN)*, pages 391–394, Ho Chi Minh City, Vietnam.

Paredes-García, D., Fernández-Carrasco, J. Á., López, J. A. M., Vasquez-Correa, J. C., Yoldi, I. J., Moreno-Acevedo, S. A., González-Docasal, A., Irazusta, H. A., Muniain, A. Á., and Loinaz, Y. d. D. (2025). Siberia: A self-sovereign identity and multi-factor authentication framework for industrial access. *Applied Sciences*, 15:8589.

Schumm, D., Mukta, R., and Paik, H.-y. (2023). Efficient credential revocation using cryptographic accumulators. In *2023 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 127–134, Athens, Greece.

Sofronie, M., Brînzea, A., Bratu, A., Aciobăniţei, I., and Pop, F. (2025). A bidirectional bridge for cross-chain revocation of verifiable credentials in segregated blockchains. *Algorithms*, 18:734.

Sporny, M., Longley, D., Sabadello, M., Reed, D., Steele, O., and Allen, C. (2022). Decentralized identifiers v1.0: Core architecture, data model, and representations. W3C Recommendation, 19 July 2022. World Wide Web Consortium (W3C). Disponível em: `https://www.w3.org/TR/did-1.0/`. Último acesso: 23/04/2026.

Tan-Vo, K. et al. (2025). Optimizing academic certificate management with blockchain and machine learning: A novel approach using optimistic rollups and fraud detection. *IEEE Access*, 13:168135–168159.

Tesei, A., Lattuca, D., Luise, M., Pagano, P., Ferreira, J., and Bartolomeu, P. C. (2023). A transparent distributed ledger-based certificate revocation scheme for vanets. *Journal of Network and Computer Applications*, 212:103569.

Vikram, S. (2023). Enhancing credential security in distributed manufacturing: Machine learning for monitoring and preventing unauthorized client certificate sharing. *JAAFR - Journal of Advance and Future Research*, 1(7):7–15. Disponível em: `https://rjwave.org/jaafr/papers/JAAFR2307002.pdf`.

von Rauscher, C. E. (2024). Design and implementation of privacy-preserving mechanisms for metadata and public keys in blockchain-based self-sovereign identities. Disponível em: `https://files.ifi.uzh.ch/CSG/staff/schumm/extern/theses/BA-C-vonRauscher.pdf`.

Wohlmacher, P. (2000). Digital certificates: a survey of revocation methods. In *Proceedings of the 2000 ACM Workshops on Multimedia (MULTIMEDIA '00)*, pages 111–114, New York, NY, USA. Association for Computing Machinery.