PickyFilters: Uma Abordagem Prática e Eficiente para Autenticação Anónima com Pseudónimos

Guilherme Santos, Cláudio Correia e Luís Rodrigues
{guilherme.silva.santos, claudio.correia, ler}@tecnico.ulisboa.pt

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

Resumo O problema da autenticação anónima, que permite preservar a privacidade dos clientes, tem adquirido uma importância cada vez maior, sobretudo com a proliferação de aplicações que recorrem a componentes que se executam na periferia da rede. Neste contexto, foi recentemente proposta uma nova arquitetura para suportar a autenticação anónima, denominada Privacy Keeper, que oferece garantias fortes de preservação da privacidade, incluindo a impossibilidade de fazer a ligação entre autenticações e a auditabilidade da revogação. Esta arquitetura tem a vantagem de usar primitivas criptográficas eficientes mas possui a desvantagem de utilizar listas de revogação de grande dimensão. Neste trabalho propomos e avaliamos técnicas para melhorar o desempenho do Privacy Keeper, nomeadamente permitindo que o cliente se certifique que não foi revogado e que se autentique sem ter de descarregar a totalidade da lista de revogação. Mostramos que as técnicas aqui propostas preservam as propriedades de privacidade do Privacy Keeper e aumentam a sua eficiência, reduzindo o tempo necessário para executar uma autenticação de 17 segundos para cerca de 2 milissegundos.

1 Introdução

A autenticação dos clientes é um requisito de muitas aplicações, uma vez que é comum existirem recursos ou funcionalidades que só podem ser acedidas por utilizadores autorizados. Infelizmente a necessidade de executar autenticação cria oportunidades para a recolha de dados associados ao perfil de utilização dos clientes, dados estes que podem ser usados contras os interesses dos mesmos (por exemplo, para inflacionar o preço de um serviço em função dos hábitos do utilizador). Estima-se que o acesso à informação recolhida sobre os padrões de utilização dos utilizadores é um mercado cujo valor poderá atingir os 700 mil milhões de euros em 2030 [11]. Uma forma de preservar a privacidade dos utilizadores em aplicações onde a autenticação ocorre frequentemente consiste em recorrer ao que se designa por autenticação anónima, tipicamente conseguida recorrendo ao uso de pseudónimos.

Se um cliente violar as regras de utilização do serviço, pode ser necessário impedi-lo de aceder à aplicação, o que obriga a revogar os pseudónimos anteriormente emitidos. Uma das dificuldades da autenticação anónima consiste em preservar a privacidade dos clientes durante o processo de revogação. Em

particular, o facto de dois pseudónimos aparecerem numa lista de pseudónimos revogados pode indicar que estes pertencem ao mesmo cliente, o que pode violar a privacidade do cliente, em particular se estes pseudónimos já tiverem sido utilizados ou vierem a ser utilizados pelo cliente. Neste contexto, duas propriedades relevantes do processo de autenticação são a impossibilidade de relacionar autenticações (do inglês *unlinkability*), isto é a garantia que o adversário não consegue aferir se duas autenticações distintas foram feitas pelo mesmo cliente, e a auditabilidade da revogação (do inglês *revocation auditability*) que permite ao cliente verificar se um dado pseudónimo foi revogado antes de o utilizar.

Muitos dos sistemas de autenticação anónima existentes falham em assegurar estas propriedades [4,5,10,12]. O Privacy Keeper [3] é um sistema de autenticação anónima que assegura as propriedades acima referidas. Para além disso, o Privacy Keeper baseia-se em criptografia assimétrica, que é significativamente mais eficiente que outros mecanismos criptográficos, tais como as *Zero-Knowledge Proofs*, usadas noutros sistemas que oferecem garantias semelhantes [8, 9]. Para revogar um pseudónimo, o Privacy Keeper codifica o mesmo num filtro de Bloom [2] que é usado como lista de revogação e que deve ser revelado ao cliente antes de cada autenticação. Em sistemas de larga escala, como VANETs, este filtro pode facilmente atingir gigabytes, tornando este esquema pouco prático, uma vez que transferir um filtro deste tamanho pode levar várias dezenas de segundos, e as aplicações com restrições temporais requerem latências da ordem de milissegundos (e.g., 5ms–30ms para ambientes na periferia da rede [7]).

Neste trabalho propomos e avaliamos uma nova biblioteca de ferramentas para melhorar o desempenho do Privacy Keeper. Nomeadamente, desenvolvemos técnicas que permitem a um cliente verificar que não foi revogado sem ter de descarregar a lista de revogação completa. Propomos duas soluções diferentes que permitem ao cliente transferir menos informação, nomeadamente, i) uma técnica suportada em assinaturas editáveis (do Inglês, *Redactable Signatures*) [6] que permite a um cliente pedir apenas algumas entradas de um filtro. 2) uma técnica suportada em filtros de Bloom hierárquicos (do Inglês, *Hierarchical Bloom Filters*) [13] que permite a um cliente transferir sequencialmente filtros mais pequenos que o original. Ambas as nossas técnicas preservam todas as propriedades de segurança necessárias, como a integridade e a autenticidade dos filtros, um desafio especialmente interessante de alcançar para a primeira técnica. O nosso componente permite ao cliente receber apenas partes mais reduzidas dos filtros para a autenticação, a razão para a nossa biblioteca se chamar PickyFilters. Através da avaliação experimental da nossa solução mostramos que esta reduz o tempo necessário para executar uma autenticação de 17 segundos para cerca de 2 milissegundos, considerando um filtro de Bloom de 1.25GB.

2 Contexto

Nos próximos parágrafos, começamos por introduzir os *filtros de Bloom*, uma das estruturas de dados mais usadas em esquemas de autenticação, assim como uma extensão dos mesmos, designada por *filtros de Bloom hierárquicos*. Por fim,

apresentamos as *assinaturas editáveis*, que permitem garantir a integridade e autenticidade sobre mensagens parciais.

2.1 Filtros de Bloom

Os filtros de Bloom [2] são uma estrutura de dados probabilista concebida para armazenar informação sobre a filiação de objetos num conjunto e testar de forma eficiente se um dado objeto pertence ao conjunto. Um filtro de Bloom é composto por um vetor de dígitos binários (bits). Para registar a presença de um objeto no conjunto, usam-se múltiplas funções de síntese (*hash*) sobre o objeto, em que cada função calcula uma posição do vetor que deve ser colocada a "1". Para testar se um objeto faz parte do conjunto, o mesmo processo é aplicado, e verifica-se se os dígitos indicados pelas funções de síntese estão a "1". Se estiverem, o objeto é dado como pertencente ao conjunto. Os filtros de Bloom, são uma estrutura probabilista, permitindo falsos positivos (mas impossibilitando a ocorrência de falsos negativos), onde um objeto é dado como pertencente ao conjunto, quando não pertence. A taxa de falsos positivos depende de vários fatores como, o tamanho do filtro, o número de elementos do conjunto e o número de funções de síntese utilizadas, sendo que um filtro maior tem capacidade para armazenar um maior número de elementos, mantendo a taxa de falsos positivos.

Em sistemas de autenticação baseados em pseudónimos, os filtros de Bloom são usualmente usados para codificar a lista de revogação e registar quais os pseudónimos revogados. Em sistemas de larga escala, devido à quantidade de pseudónimos a serem revogados e à necessidade de manter a taxa de falsos positivos baixa, o tamanho dos filtros pode chegar a valores consideráveis, tornando onerosa a transmissão frequente dos mesmos. De notar que, a taxa de falsos positivos é escolhida *a priori*, tendo em conta as necessidades do sistema, sendo as características do filtro calculadas em função da mesma.

2.2 Filtros de Bloom Hierárquicos

Os *filtros de Bloom hierárquicos* [13] são uma estrutura de dados baseada em filtros de Bloom que tenta aumentar a capacidade de escala dos filtros de Bloom originais. Estes filtros utilizam uma estrutura hierárquica composta por múltiplos filtros de Bloom com diferentes tamanhos (sendo que os filtros nas posições mais elevadas da hierarquia são mais pequenos que os filtros nas posições mais baixas), em que o registo/teste de presença é feito recorrendo ao múltiplos filtros. Tipicamente, um teste de pertença na estrutura de dados obriga a realizar, de forma sequencial, testes de presença nos vários filtros até se obter uma conclusão.

2.3 Assinaturas Editáveis

As *assinaturas editáveis* [6] possibilitam o envio de uma mensagem digitalmente assinada, apagando certas partes da mesma, mantendo a capacidade do recetor verificar as propriedades de segurança da mensagem, como a integridade e autenticidade. O processo de criação de uma assinatura editável começa por dividir a mensagem em partes removíveis. É construída uma árvore de Merkle associ-

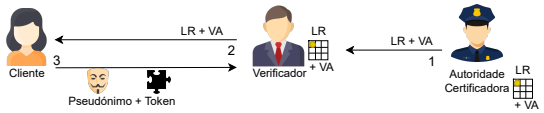


Figura 1: Autenticação no Privacy Keeper. LR é lista de revogação, e VA é parâmetro aleatório.

ando cada parte da mensagem a cada nó folha da árvore calculando a sua síntese criptográfica e depois construindo a árvore recursivamente até ao nó raiz, sendo o valor de cada nó a síntese da concatenação das sínteses dos seus nós filhos. Depois disso, o remetente da mensagem, assina digitalmente o nó raiz.

Ao enviar a mensagem, o remetente, apaga as partes da mensagem que lhe interessam e envia as restantes partes ao recetor. Juntamente, envia também as sínteses dos nós folhas associados às partes apagadas, podendo comprimir este conjunto de sínteses usando a estrutura da árvore, enviando as sínteses de nós internos. O recetor usa a informação recebida, para reconstruir a síntese do nó raiz e verificar a assinatura digital do remetente nesse mesmo nó.

As assinaturas editáveis recorrem também a uma componente aleatória durante a construção da assinatura, que impede que o cliente recupere as partes apagadas usando ataques de força bruta.

3 Privacy Keeper

O Privacy Keeper [3] é uma técnica de autenticação que permite atingir as propriedades desejáveis em esquemas de autenticação anónima suportado unicamente por criptografia assimétrica. O esquema usa uma abordagem onde os clientes se autenticam usando um pseudónimo e um *testemunho* de não-revogação, que comprova a validade do pseudónimo usado. Este esquema contempla a existência de 3 entidades, ilustradas na Figura 1: i) os clientes, ii) os verificadores, responsáveis pelo controlo de acesso a sistemas, iii) e uma autoridade certificadora (AC), confiada tanto por clientes como por verificadores, responsável por emitir e revogar pseudónimos a clientes. O esquema divide-se em 3 fases: obtenção de pseudónimos, revogação de pseudónimos e o controlo de acesso.

3.1 Obtenção de Pseudónimos

A obtenção de pseudónimos acontece durante a entrada de um cliente do sistema, ou caso um cliente precise de mais pseudónimos para continuar as suas autenticações. O cliente estabelece uma ligação com a autoridade certificadora e envia um pedido por pseudónimos. Cada pseudónimo p tem um par de chaves assimétricas associado $\langle K^+, K^- \rangle$ e é definido pela chave pública e por uma assinatura feita com a chave privada K^-_{AC} da autoridade certificadora, $p = \langle K^+_p, \{K^+_p\}^{K^-_{AC}} \rangle$.[1]

[1] A composição do pseudónimo foi simplificada por razões de clareza e compreensão. Para uma descrição completa remete-se o leitor para [4].

3.2 Revogação de Pseudónimos

Na necessidade de remover o acesso de um cliente ao sistema, a autoridade certificadora emite uma nova da Lista de Revogação (LR) contendo informação específica deste e dos clientes anteriormente revogados. Uma LR permite a um verificador descobrir no momento de autenticação se um cliente se encontra revogado ou não do sistema. O processo de autenticação é descrito na próxima secção. Cada LR está associada a um valor único e aleatório η, que garante que cada LR é também única.

Assim, quando um cliente u é revogado do sistema a AC começa por gerar um novo η para ser usado na nova LR. Depois, a AC acede a todos os pseudónimos de u, e para cada pseudónimo gera um *testemunho* criptográfico σ, este testemunho é definido pela assinatura digital usando a chave privada do pseudónimo sobre o η, $\sigma = \{\eta\}^{K_p^-}$. Como nesta nova versão da LR o η é novo, a AC tem de também calcular os testemunhos de não-revogação para todos os outros clientes revogados até ao momento, usando o novo η. No final, todos os testemunhos σ deste conjunto são introduzidos num filtro de Bloom FB_η. Para garantir a autenticidade do novo FB_η a AC gera uma assinatura digital sobre o FB_η concatenado com o η, e publica estes elementos que definem a nova lista de revogação, $LR_{new} = \langle FB_\eta, \eta, \{FB_\eta \parallel \eta\}^{K_{CA}^-} \rangle$.

3.3 Controlo de Acesso

No momento em que um cliente u pretende autenticar-se perante um verificador, como ilustrado na Figura 1, deve escolher um pseudónimo p que ainda não tenha sido usado para se autenticar. O resto do processo pode ser dividido em duas fases: Na primeira fase o cliente deve verificar que p não se encontra revogado na atual LR; Na segunda fase o cliente deve usar p para efetivamente se autenticar perante o verificador. De seguida descrevemos estas duas fases em mais detalhe.

Consulta do Estado de um Pseudónimo De forma a alcançar a propriedade de auditabilidade da revogação, o cliente começa por pedir ao verificador a atual LR para verificar o estado de p no sistema. Para isto, depois de receber a LR e de verificar a assinatura, o cliente extrai η e gera o correspondente testemunho σ para esta LR. Por fim, o cliente verifica se σ se encontra no filtro FB_η, caso se encontre significa que p foi revogado, e o cliente deve de parar imediatamente de usar o sistema para manter a sua privacidade. Caso o testemunho de p não esteja no filtro FB_η, o cliente pode prosseguir a autenticação para a segunda fase. Note que, a auditabilidade é alcançada porque o σ_η^p é único para a LR apresentada, e não se encontra em mais nenhuma LR. O facto de um testemunho de não-revogação ser gerado especificamente para uma dada lista de revogação protege o cliente, pois caso o verificador envie uma lista desatualizada não consegue encontrar o testemunho fornecido noutras listas de revogação mais recentes.

Autenticação de um Pseudónimo Caso o cliente chegue à conclusão de que não está revogado, pode autenticar-se no sistema enviando o seu pseudónimo p e o testemunho de não-revogação σ_η^p calculada especificamente para a lista

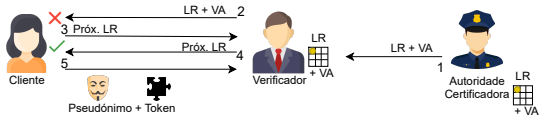


Figura 2: Autenticação com filtros de Bloom hierárquicos. LR é lista de revogação, e VA é parâmetro aleatório.

de revogação que recebeu. O verificador confirma que o testemunho recebido foi gerado corretamente e que não está presente na sua lista de revogação mais recente, se não estiver presente a autenticação tem sucesso, caso contrário falha.

4 PickyFilters

Uma desvantagem do Privacy Keeper é que obriga à transferência da lista de revogação para o cliente o que, dada a potencial elevada dimensão da mesma, pode induzir uma latência elevada no processo de autenticação. O nosso trabalho incide sobre este problema, com o objectivo de atenuar a quantidade de informação que os clientes recebem em cada autenticação, mantendo todas as propriedades de segurança inerentes ao filtro de Bloom, nomeadamente integridade e autenticidade. Propomos duas técnicas para este fim: 1) uma técnica baseada em filtros de Bloom hierárquicos, 2) uma técnica inspirada em assinaturas editáveis. Em ambas os casos, mantemos a mesmas entidades do protocolo original: clientes, verificadores e a autoridade certificadora; e as mesmas fases, com substanciais alterações nas seguintes fases: revogação de pseudónimos e controlo de acesso.

4.1 PickyFilters com Filtros de Bloom Hierárquicos

Nesta técnica, propomos que a lista de revogação seja composta por múltiplos filtros de Bloom com diferentes tamanhos, para além do filtro original, cada um contendo exatamente os mesmos testemunhos de não-revogação. Estes filtros são organizados de forma sequencial e por ordem crescente do seu tamanho.

Revogação de Pseudónimos Durante a revogação de um cliente, a autoridade certificadora cria um conjunto de n novos filtros de Bloom FBn_η e um novo valor único para o parâmetro aleatório η associado à nova lista de revogação. São calculados todos os testemunhos de não-revogação para os pseudónimos emitidos a clientes revogados até ao momento, sendo os testemunhos inseridos em todos os filtros de Bloom. Finalmente, a autoridade central assina cada um destes filtros com a sua chave privada, protegendo a autenticidade de cada filtro. Posteriormente, estes filtros e as respetivas assinaturas digitais são agregados, numa lista de revogação $LR_{new} = \langle [FB1_\eta, ..., FBn_\eta], \eta, [\{FB1_\eta \,\|\, \eta\}^{K^-_{CA}}, ..., \{FBn_\eta \,\|\, \eta\}^{K^-_{CA}}] \rangle$, que é enviada para os verificadores.

Controlo de Acesso Durante o processo de autenticação, representado na Figura 2, o cliente começa por descarregar o primeiro filtro de Bloom da lista de revogação, isto é, aquele que tem o menor tamanho, assim como o parâmetro aleatório η associado à lista de revogação. Depois, escolhe um pseudónimo p e calcula um testemunho de não-revogação σ, usando p e η. De seguida, verifica se este se encontra marcado como revogado no filtro recebido anteriormente. Como a taxa de falsos positivos de um filtro de Bloom varia com o seu tamanho e o cliente começa por receber o filtro mais pequeno, não é improvável que o cliente tenha um falso positivo neste passo. Nesse caso, o cliente irá então por transferir os restantes filtros da lista de revogação, por ordem crescente de tamanho, até chegar a um filtro em que o pseudónimo não esteja registado como revogado. Se chegar ao último filtro, e o pseudónimo estiver registado como revogado nesse filtro, o cliente assume que este foi de facto revogado e cancela a autenticação. A autenticidade de cada filtro é verificada pelo cliente usando a assinatura da autoridade certificadora. Caso o cliente chegue à conclusão de que não está revogado, pode então concluir a autenticação enviando o pseudónimo p e o respetivo testemunho ao verificador.

Esta técnica é vantajosa nos casos em que o cliente consegue confirmar que o pseudónimo não se encontra revogado através da consulta dos primeiro filtros (idealmente, recorrendo apenas ao menor filtro na maioria dos casos), podendo, no entanto, piorar a solução original caso o cliente descarregue vários filtros.

4.2 PickyFilters com Assinaturas Editáveis

Nesta técnica, tiramos partido do facto de que o cliente pode saber de antemão as características do filtro de Bloom que codifica a lista de revogação, como o tamanho e o número de funções de síntese, e o parâmetro aleatório η associado à lista. Usando esta informação, o cliente pode calcular o seu testemunho de não-revogação e saber quais as entradas do vetor que necessita de verificar para perceber se um dado pseudónimo foi revogado. Propomos então que o cliente, ao invés de transferir o filtro inteiro, descarregue apenas segmentos do filtro, de forma a obter todas as entradas do vetor de que necessita.

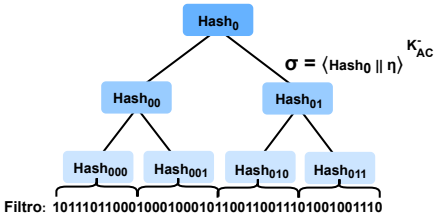


Figura 3: Assinatura editável de um filtro. AC é autoridade certificadora, η é parâmetro aleatório e σ assinatura digital.

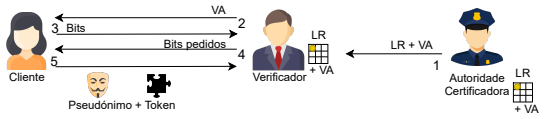


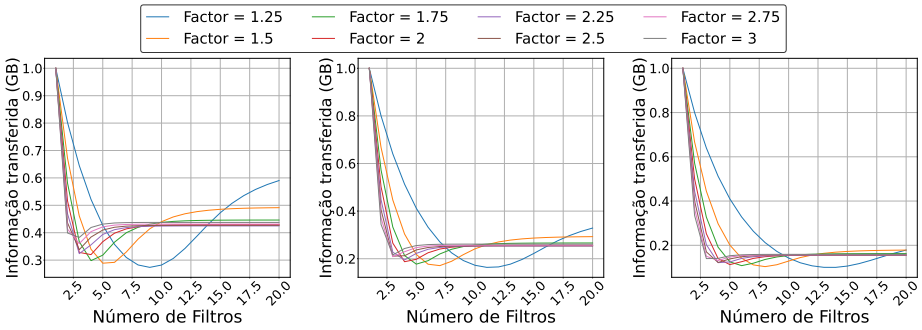
Figura 4: Autenticação com assinaturas editáveis. LR é lista de revogação, e VA é parâmetro aleatório.

Revogação de Pseudónimos Ao transferir apenas partes da lista de revogação, o cliente não consegue verificar a integridade da lista usando uma assinatura digital comum, como proposto no Privacy Keeper. Como forma a ultrapassar este problema, sugerimos que a entidade central assine a lista de revogação usando uma técnica inspirada nas assinaturas editáveis.

Na nossa solução, depois da criação da lista de revogação $LR_{new} = \langle FB_\eta, \eta \rangle$, o filtro de Bloom é dividido em múltiplos segmentos de tamanho configurável, e a síntese criptográfica de cada segmento é associada aos nós filhos de uma árvore de Merkle. Os restantes nós da árvore são, então, gerados recursivamente fazendo a síntese da concatenação das sínteses do nós filhos. Por fim, a autoridade central gera uma assinatura digital σ usando a concatenação entre a síntese raiz e do parâmetro aleatório η associado à lista de revogação, conforme ilustrado na Figura 3. Os elementos gerados formam a nova lista de revogação, $LR_{new} = \langle FB_\eta, \eta, \sigma \rangle$, que é distribuída aos verificadores, que depois constroem a árvore binária usando o mesmo processo.

Controlo de Acesso Na fase de controlo de acesso, exibida na Figura 4, o verificador começa por enviar ao cliente o parâmetro aleatório η da sua lista de revogação mais recente. O cliente escolhe um pseudónimo p que ainda não tenha usado e gera um testemunho de não-revogação, usando p e η. Conhecendo a parametrização do filtro de Bloom que codifica a lista de revogação, o cliente pede apenas os segmentos relevantes para verificar o seu estado de autenticação ao verificador. O verificador responde enviando estes segmentos e as sínteses necessárias para que o cliente consiga reconstruir o caminho, desde os nós folha, associados aos segmentos pedidos, até ao nó raíz.

Ao receber esta informação, o cliente certifica-se que não está revogado usando os segmentos do filtro de Bloom que recebeu, verifica a assinatura σ da entidade certificadora, depois de calcular a síntese do nó raíz, recursivamente, usando a informação enviada pelo verificador. A integridade dos segmentos do filtro de Bloom que o cliente recebeu, está salvaguardada pela estrutura da árvore binária. Caso um verificador malicioso modificasse algum pedaço, essa mudança refletir-se-ia em algum nó superior da árvore devido às propriedades da árvore de Merkle. A alteração afetaria necessariamente a síntese do nó raíz, pelo que o cliente conseguiria aperceber-se da alteração ao verificar a assinatura da autoridade certificadora nesse mesmo nó. Caso o cliente chegue à conclusão de que não se encontra revogado, termina a autenticação, enviando ao verificador o seu pseudónimo p e o testemunho correspondente.



(a) Taxa de falsos positivos = 0.1%

(b) Taxa de falsos positivos = 0.01%

(c) Taxa de falsos positivos = 0.001%

Figura 5: Variação de Parâmetros usando Filtros de Bloom Hierárquicos

5 Avaliação

Avaliámos experimentalmente ambas as técnicas propostas no PickyFilters, codificando cada técnica em C++. Desenvolvemos um cliente e um verificador usando duas máquinas Intel NUC10i7FNB. As máquinas têm um processador Intel i7-10710U, 16 GB RAM e Ubuntu 22.04 LTS. Analisámos o impacto da variação dos diversos parâmetros de ambas as soluções na quantidade de informação transmitida durante a autenticação. Avaliámos também ambas as técnicas medindo a latência média do processo de autenticação, e comparámo-las com a latência média do Privacy Keeper. Usámos o algoritmo Ed25519 [1] para gerar assinaturas digitais determinísticas.

5.1 PickyFilters com Filtros de Bloom Hierárquicos

Como descrito anteriormente, a nossa técnica baseada em filtros de Bloom hierárquicos usa diversos filtros com diferentes tamanhos, onde existem vários parâmetros que influenciam diretamente a quantidade de informação a ser transferida na autenticação. Assim, avaliamos os seguintes parâmetros: 1) *factor* de redução entre cada filtro, começando no filtro maior/original até ao mais pequeno. 2) *número de filtros* com diferentes tamanhos usados na hierarquia, ou seja, quantas vezes reduzimos o filtro maior pelo factor escolhido. 3) *taxa de falsos positivos* que aceitamos no maior filtro da hierarquia. Na Figura 5, variamos estes parâmetros e calculamos a quantidade esperada de informação relativa a filtros de Bloom que é necessário transferir no momento da autenticação para cada uma destas configurações, fixando o tamanho do maior filtro a 1GB.

Vale a pena destacar que calcular a quantidade esperada de informação transferida não é trivial, uma vez que depende do número de filtros que o cliente tem de transferir antes de se autenticar. Para avaliar cada conjunto de parâmetros, calculámos a quantidade de informação transferida para cada cenário (em cada cenário, o cliente necessita de descarregar um número distinto de filtros antes de

terminar o processo de auditoria), e calculámos o valor esperado da quantidade de informação transferida pelo cliente usando as taxas de falsos positivos de cada filtro de Bloom para calcular a probabilidade de cada cenário.

É possível observar que, quando existe apenas um filtro, a informação necessária para transferir é sempre 1GB, que representa o filtro original. Contudo, quando criamos mais filtros, a quantidade média de informação transmitida decresce rapidamente para menos de metade a partir de um número de filtros superior a 5. Além disso, observamos que quanto menor é o factor de redução menor é a quantidade esperada de informação transmitida em cada autenticação, sendo no entanto necessário aumentar o número de filtros usados para atingir o mínimo de informação transmitida.

Observamos que a informação transferida diminui quando também reduzimos a taxa de falsos positivos. A justificação deriva do facto de que a redução da taxa se reflete também na taxa dos filtros mais pequenos, uma vez que todos os filtros terão taxas de falsos positivos menores e um maior número de clientes autenticar-se-á nos primeiros filtros. Note que, para diminuir a taxa de falsos positivos com o tamanho do filtro fixo, é preciso reduzir o número de itens no filtro.

Uma boa configuração para uma taxa de falsos positivos de 0.001% seria escolher um fator de 2 com 4 filtros, sendo este um ponto que minimiza a informação a ser transmitida na rede, mantendo um número razoável de filtros. A nossa técnica reduz drasticamente a informação transferida, em particular, transferindo apenas cerca de 10% da quantidade de informação original em média.

5.2 PickyFilters com Assinaturas Editáveis

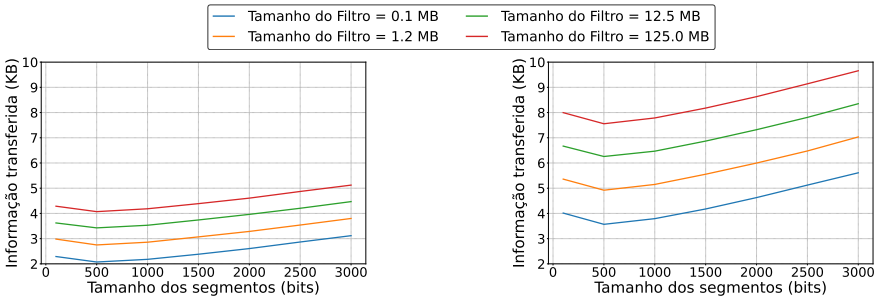
A nossa técnica baseada em Assinaturas Editáveis, tem duas estruturas principais, o filtro de Bloom, e uma árvore de Merkle, construída a partir do filtro. A informação transferida durante a autenticação é constituída, sobretudo, por segmentos do filtro de Bloom e por várias sínteses da árvore de Merkle.

Existem vários parâmetros da solução que impactam a quantidade de informação que é transferida durante a autenticação. Estes parâmetros incluem: 1) *número de funções de síntese*, que influencia diretamente o número de segmentos do filtro que são enviados ao cliente e, 2) *tamanho dos segmentos*, em que o filtro é divido, uma vez que o seu tamanho determina o número de segmentos do filtro que serão associados aos nós folha da árvore, influenciando o tamanho da árvore de Merkle e o número de sínteses que são enviadas ao cliente.

Na Figura 6, variamos estes parâmetros e medimos a quantidade média de informação que o verificador envia ao cliente no momento da autenticação.

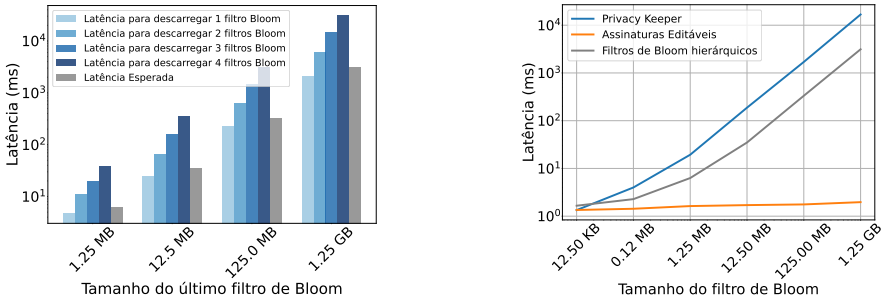
É percetível que aumentar o número de funções de síntese do filtro, aumenta a quantidade de informação transmitida na rede, devido ao facto de o cliente receber mais segmentos quanto maior o número de funções de síntese e também por ser necessário enviar mais sínteses para que o cliente possa calcular os caminhos desde os nós folha até ao nó raíz, recursivamente.

A quantidade de informação transferida durante a autenticação atinge o seu mínimo, considerando os tamanhos dos filtros e o número de funções de síntese estudados, quando o tamanho dos segmentos é 500 dígitos binários. É possí-



(a) Número de funções de síntese = 5 (b) Número de funções de síntese = 10

Figura 6: Variação de Parâmetros usando Assinaturas Editáveis



(a) Latência usando filtros hierárquicos (b) Latência média

Figura 7: Latência da Autenticação

vel compreender que ao ter segmentos mais curtos, a árvore de Merkle associada será maior e mais sínteses serão enviadas ao cliente, e por outro lado, caso tenhamos segmentos grandes, mais informação referente aos mesmos será enviada ao cliente. Sendo os 500 dígitos binários o valor ótimo entre estes dois parâmetros.

Concluí-se através desta experiência que uma boa configuração seria dividir o filtro de Bloom em segmentos de 500 dígitos binários. No pior dos cenários, caso o filtro tivesse 10 funções de síntese, a solução permitiria reduzir a informação enviada ao cliente de 125MB para 7KB. É de salientar que isto representa uma redução significativa da quantidade de informação original (concretamente, 0.006% do tamanho original).

5.3 Filtros Hierárquicos vs Assinaturas Editáveis

Nesta secção avaliamos a latência da autenticação de ambas as técnicas propostas e comparamo-las com a latência da autenticação do Privacy Keeper. Com o objetivo de fazer uma comparação justa entre ambas as técnicas, escolhemos configurações que minimizem a informação trocada no momento da autenticação entre verificadores e clientes, de acordo com a análise apresentada anteriormente.

Para a avaliação de todos os sistemas, desenvolvemos um conjunto de testes onde medimos a latência média da autenticação de um cliente, variando o tamanho do filtro de Bloom que implementa a lista de revogação. No caso da técnica baseada em filtros de Bloom hierárquicos, este tamanho corresponde ao tamanho do último e maior filtro. Para a avaliação do PickyFilters com filtros de Bloom hierárquicos, usámos a seguinte configuração: factor de redução de 2, portanto, cada filtro tem metade do tamanho do filtro seguinte, fixámos o número de filtros do conjunto a 4, com 5 funções de síntese em cada filtro. Variando a latência do último filtro, medimos a latência da autenticação em cada um dos 4 cenários possíveis (em cada cenário, o cliente descarrega um número distinto de filtros). Com as 4 latências, calculámos a latência esperada de cada autenticação, tendo em conta a latência para cada um dos 4 cenários e a taxa de falsos positivos de cada filtro, que apresentamos na Figura 7a. Para o cálculo da latência esperada considerámos uma taxa de falsos positivos de 0.01% para o último filtro de Bloom, que utilizámos para calcular as taxas de falsos positivos para os outros filtros. Note que apenas esta técnica tem a sua eficiência dependente da taxa de falsos positivos aceitável na lista de revogação. Usamos a latência esperada obtida para a comparação com as outras duas técnicas. Para a avaliação da técnica baseada em Assinaturas Editáveis fixámos o tamanho dos segmentos a 500 dígitos binários e o número de funções de síntese a 5, medindo a latência da autenticação.

Na Figura 7b, fazemos a comparação das latências esperadas (e a sua evolução com o aumento do tamanho do filtro de Bloom) usando as duas técnicas propostas e o Privacy Keeper original (que transfere sempre a lista de revogação na sua totalidade). A nossa avaliação experimental mostra que, para filtros de Bloom de 1.25GB, a latência da autenticação é, em média, $\pm 17s$ no Privacy Keeper original, $\pm 3s$ com filtros de Bloom hierárquicos e $\pm 2ms$ com Assinaturas Editáveis. O Picky Filters com Assinaturas Editáveis apresenta pois o melhor desempenho, reduzindo de forma significativa a latência de autenticação em aproximadamente $99,99\%$ comparado com o Privacy Keeper original.

6 Conclusões

Neste trabalho abordamos o problema de assegurar de forma eficiente a auditabilidade da revogação. Em trabalhos anteriores, esta propriedade era assegurada enviando ao cliente a lista de revogação completa, para que este pudesse verificar o seu estado de revogação antes de se autenticar no sistema. Este requisito pode levar uma autenticação a demorar até $17s$, algo inaceitável em aplicações com interação humana. Propomos duas técnicas que permitem transferir menos informação, alcançando assim, uma latência de autenticação aceitável e prática. Mostramos experimentalmente, que uma das técnicas reduz o tempo necessário para autenticação de forma muito significativa (de $17s$ para $2ms$).

Agradecimentos: Este trabalho foi suportado pela Fundação para a Ciência e a Tecnologia (FCT) por fundos nacionais através dos projectos INESC-ID UIDB/50021/2020, e o SmartRetail (financiado pela IAPMEI com ref. C6632206063-00466847).

Referências

1. Bernstein, D., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of cryptographic engineering **2**(2), 77–89 (2012)
2. Bloom, B.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM pp. 422–426 (1970)
3. Correia, C.: Low-Latency Privacy-Preserving Access to Edge Storage. Ph.D. thesis, Instituto Superior Tecnico, Universidade de Lisboa (Jul 2024)
4. Correia, C., Correia, M., Rodrigues, L.: Using range-revocable pseudonyms to provide backward unlinkability in the edge. In: Proceedings of the ACM Conference on Computer and Communications Security. Copenhagen, Denmark (2023)
5. Haas, J., Hu, Y.C., Laberteaux, K.: Efficient certificate revocation list organization and distribution. IEEE Journal on Selected Areas in Communications **29**(3), 595–604 (2011)
6. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Topics in Cryptology — CT-RSA 2002. pp. 244–262. Springer (2002)
7. Ricart, G.: A city edge cloud with its economic and technical considerations. In: Proceedings of the International Workshop on Smart Edge Computing and Networking. Kona (HI), USA (Jun 2017)
8. Tsang, P., Au, M., Kapadia, A., Smith, S.: Blacklistable anonymous credentials: blocking misbehaving users without TTPs. In: Proceedings of the 14th ACM conference on Computer and Communications Security. pp. 72–81 (2007)
9. Tsang, P., Au, M., Kapadia, A., Smith, S.: PEREA: Towards practical TTP-free revocation in anonymous authentication. In: Proceedings of the 15th ACM conference on Computer and Communications Security. pp. 333–344 (2008)
10. Tsang, P., Kapadia, A., Cornelius, C., Smith, S.: Nymble: Blocking misbehaving users in anonymizing networks. IEEE Transactions on Dependable and Secure Computing **8**(2), 256–269 (2009)
11. United States Government Accountability Office: Vehicle data privacy. (2017), https://www.gao.gov/assets/gao-17-656.pdf
12. Verheul, E., Hicks, C., Garcia, F.: IFAL: Issue first activate later certificates for V2X. In: Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P). pp. 279–293. IEEE (2019)
13. Zhu, Y., Jiang, H., Wang, J.: Hierarchical Bloom filter Arrays (HBA): a novel, scalable metadata management system for large cluster-based storage. In: Proceedings of the IEEE International Conference on Cluster Computing. pp. 165–174. San Diego (CA), USA (2004)