Ataques de Frequência em Deduplicação Cifrada na Nuvem

Rodrigo Silva, Cláudio Correia, Miguel Correia e Luís Rodrigues
{rodrigo.santos.silva, claudio.correia, miguel.p.correia, ler}@tecnico.ulisboa.pt

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

Resumo Para garantir a privacidade dos seus dados quando estes são armazenados na nuvem, os utilizadores podem optar por cifrar os ficheiros antes de os exportar. Infelizmente, sem mecanismos adicionais, o armazenamento de dados cifrados inviabiliza a concretização de técnicas de deduplicação do lado do servidor, pois dois ficheiros idênticos terão versões cifradas distintas. Neste artigo abordamos o problema de conciliar a necessidade de cifrar os dados com as vantagens da deduplicação, em particular, nas formas de atingir este objetivo evitando ataques de frequência (um ataque que permite inferir qual o conteúdo cifrado com base na frequência com que este é acedido). Neste contexto, propomos um novo mecanismo para escolher as chaves de cifra de forma robusta e eficiente, recorrendo a ambientes seguros de execução.

1 Introdução

A utilização de sistemas de armazenamento na nuvem é hoje ubíqua e tem inúmeras vantagens, mas, por outro lado, pode comprometer a privacidade dos dados dos clientes face a operadores curiosos. Uma forma de contornar esta limitação consiste em cifrar os dados antes de os exportar para a nuvem. Infelizmente, isto pode impedir o serviço de armazenamento de aplicar mecanismos que permitem reduzir de forma significativa os recursos necessários para fornecer o serviço, tal como mecanismos de deduplicação.

A deduplicação é uma técnica que permite a um serviço de armazenamento identificar ficheiros (ou apenas fragmentos destes ficheiros) duplicados e, desta forma, evitar armazenar cópias em excesso do mesmo conteúdo. Esta técnica permite obter reduções significativas no custo de armazenamento, uma vez que se verificou experimentalmente que em sistemas que armazenam grandes quantidades de dados de vários utilizadores, como Dropbox [7] e Google Drive [9], é possível encontrar quantidades elevadas de dados repetidos. Infelizmente é difícil aplicar deduplicação a dados cifrados. Se os utilizadores cifrarem os seus dados antes de os armazenarem, e usarem chaves diferentes para realizar a cifra, o mesmo fragmento irá produzir diferentes criptogramas. Isto faz com que o sistema de armazenamento não possa identificar que dados são redundantes, e por esta razão não seja capaz de aplicar deduplicação.

Deduplicação cifrada é o nome dado a técnicas que combinam cifra de ficheiros e deduplicação de dados. Normalmente, esta combinação requer alguma forma de coordenação, direta ou indireta, entre os diferentes clientes. O desafio é realizar essa coordenação de forma eficiente e que permita preservar a privacidade da informação armazenada por cada utilizador. Uma potencial vulnerabilidade das técnicas deduplicação cifrada é a exposição a ataques de frequência, um ataque que permite inferir qual o conteúdo cifrado com base na frequência com que este é acedido. Neste contexto, propomos um novo mecanismo para escolher as chaves de cifra de forma robusta e eficiente, recorrendo a ambientes seguros de execução, tal como o Intel SGX.

2 Contexto

2.1 Deduplicação

A deduplicação é uma técnica que permite reduzir o espaço de armazenamento necessário para guardar grandes conjuntos de dados através da identificação de conteúdos duplicados. Quando um cliente pretende armazenar um ficheiro, o sistema divide o ficheiro em fragmentos e de seguida, verifica se cada um destes fragmentos já foi previamente armazenado (pelo mesmo ou por outro cliente). Em caso afirmativo, é evitado o armazenamento de um novo conjunto de réplicas desse fragmento. Estudos demonstram que a deduplicação pode gerar poupanças de armazenamento significativas em um ambiente de produção, por exemplo, poupanças de 50% em armazenamento primário [13] e até 98% no armazenamento de cópias de segurança [18].

2.2 Deduplicação Cifrada

Quando os clientes cifram os dados antes de os armazenar, as oportunidades para realizar deduplicação podem ser drasticamente reduzidas. Isto porque mesmo que dois clientes armazenem o mesmo ficheiro, o texto cifrado será distinto. Designa-se por deduplicação cifrada um conjunto de técnicas que pretendem conciliar a cifra dos ficheiros com a deduplicação. A maioria destas técnicas baseia-se em mecanismos que permitem aos clientes coordenar-se, de forma implícita ou explícita, para escolherem as mesmas chaves para cifrar os mesmos ficheiros, e desta forma garantirem que fragmentos com o mesmo conteúdo resultam no mesmo texto cifrado. Uma das possíveis técnicas para atingir este fim designa-se por Message-Locked Encryption (MLE) [1] que deriva a chave de forma determinística a partir do conteúdo dos dados, normalmente através de uma função criptográfica de *hash* (p.ex., SHA-2) aplicada ao conteúdo do fragmento. No entanto, a MLE tem várias limitações. Em particular, se o serviço de armazenamento tiver acesso a um dado conteúdo, pode gerar o texto cifrado correspondente e depois identificar os clientes que possuem este conteúdo.

2.3 Análise de Frequência

Infelizmente, cifrar o conteúdo dos ficheiros pode não ser o suficiente para assegurar a privacidade dos conteúdos armazenados por um dado utilizador. Outra vulnerabilidade pode ser posta em causa se um atacante conhecer a frequência com que determinado conteúdo aparece num conjunto de dados, pois pode correlacionar este valor com a frequência com que um texto cifrado é submetido para armazenamento e, desta forma, inferir relações entre os textos cifrados e os respetivos conteúdos. Algumas técnicas de deduplicação cifrada, como o MLE, sofrem desta vulnerabilidade: uma vez que um dado conteúdo é sempre cifrado com a mesma chave, a frequência do texto cifrado é exatamente a mesma da do conteúdo original. Ou seja, o MLE permite preservar as oportunidades de aplicar deduplicação mas não oferece nenhuma proteção contra a análise de frequência.

2.4 Mediação Segura

Uma maneira de conseguir deduplicação cifrada e evitar a análise de frequência consiste em recorrer a uma entidade de confiança que coordena a atribuição de chaves de cifra a conteúdos. Sempre que pretendem cifrar um ficheiro, os clientes contactam um mediador seguro, que é responsável por indicar qual a chave que deve ser usada para cifrar cada fragmento. Dadas diversas cópias do mesmo conteúdo, a escolha das chaves pelo mediador permite separar estas cópias em diferentes conjuntos, dentro de cada conjunto as cópias são cifradas com uma dada chave (permitindo a deduplicação), e cada conjunto usa uma chave diferente (evitando a análise de frequência). A confiança no mediador pode ser conseguida usando técnicas criptográficas e/ou hardware seguro.

2.5 Ambientes de Execução Seguros

Um ambiente de execução seguro é um modo do processador que permite executar código e armazenar dados, de forma isolada do sistema operativo e dos processos de nível de utilizador. Os enclaves são ambientes de execução seguros que estão disponíveis em muitos dos processadores comuns da Intel e que são suportados por uma arquitetura denominada Software Guard Extensions (SGX) [17], a qual recorre a mecanismos de hardware como *segredos de hardware*, *atestação remota*, *armazenamento selado* e *cifra de memória*. A utilização de enclaves é uma das técnicas possíveis para concretizar mediadores seguros em sistemas de deduplicação cifrada.

O enclave depende de uma região de memória protegida por hardware chamada Enclave Page Cache (EPC) para hospedar código e dados protegidos. Um EPC compreende páginas de 4 KB, e qualquer aplicação no enclave pode usar até 128 MB. Se um enclave tiver um tamanho maior que o EPC, este cifra as páginas não utilizadas e transfere-as para a memória desprotegida, sofrendo uma penalidade de desempenho [5]. O SGX fornece duas interfaces: ECALLs, usadas por uma aplicação para invocar a funcionalidade do enclave e OCALLs, usadas pelo código do enclave para aceder a uma aplicação externa.

3 Trabalho Relacionado

Na literatura é possível encontrar diversas propostas de mediadores seguros para facilitar a deduplicação cifrada.

O Duplicateless Encryption for Simple Storage (DupLESS) [11] usa um servidor dedicado responsável por atribuir chaves de cifra a fragmentos. O seu funcionamento é inspirado no MLE, uma vez que a chave de cifra depende do *hash* do fragmento. No entanto, ao contrário do MLE básico, a chave de cifra depende também de um segredo apenas conhecido pelo mediador. Isto impede o servidor de armazenamento de conhecer qual o texto cifrado que está associado a um dado conteúdo. A confiança no mediador é obtida através de protocolos criptográficos, baseados em funções aleatórias inconscientes [15] e assinaturas cegas RSA [2,3,4], que garantem que o servidor não consegue extrair o *hash* do fragmento e que o cliente só consegue extrair a chave de cifra.

O Tunable Encrypted Deduplication (TED) [12] utiliza uma arquitetura semelhante mas acrescenta mecanismos para controlar o número de vezes que um dado conteúdo é cifrado com a mesma chave. Para este efeito, o mediador mantém um registo de quantas vezes uma mesma chave já foi atribuída a um dado fragmento, mudando a chave quando necessário.

A utilização de ambientes de execução seguros para executar o mediador permite evitar a utilização de algoritmos criptográficos complexos para assegurar a confiança no mediador, a qual é assegurada diretamente pelo *hardware*. Em particular, os mecanismos de atestação permitem que os clientes confirmem qual é o código que está a ser executado pelo mediador, e os restantes mecanismos de segurança do enclave asseguram que a execução do mediador está protegida.

O SGXDedup [16] é, na sua essência, uma variante do DupLESS, que usa a confiança no SGX para evitar os protocolos criptográficos mais pesados do DupLESS. Apesar de introduzir vários melhoramentos em relação ao DupLESS, o SGXDedup sofre das mesmas vulnerabilidades do sistema em que se baseou, em particular no que se refere aos ataques de análise de frequência.

O SGX Enabled Secure Deduplication (S2Dedup) [14] é também uma solução baseada em ambientes de execução seguros, mas que mantém uma tabela com as frequências dos fragmentos dentro do enclave, de forma a assegurar que uma chave não é reutilizada mais do que um pré-determinado número de vezes para o mesmo conteúdo. Uma limitação do S2Dedup é que, devido à memória limitada do enclave, não consegue manter um histórico completo da frequência do acesso a todos os objectos. Desta forma, pode ser obrigado a mudar de chaves mais vezes do que o estritamente necessário, limitando a eficiência da deduplicação.

4 Arquitetura da Solução Desenvolvida

O nosso sistema é inspirado nos trabalhos anteriores e baseia-se na concretização de um mediador seguro, que se executa num enclave do servidor.

4.1 Hipóteses

Assumimos as seguintes condições: (i) a integridade e privacidade dos dados enviados pelo canal de comunicação entre o cliente e servidor são protegidas pelo uso de operações criptográficas; (ii) a comunicação entre o cliente e o enclave é segura; (iii) não existe conluio entre o cliente e o servidor de armazenamento; (iv) os ficheiros receita (explicados mais à frente) estão armazenados de forma segura, pois são protegidos com a chave do cliente; (v) o enclave garante a privacidade e integridade do código e dos dados lá mantidos.

Consideramos um adversário honesto mas curioso, ou seja, que não modifica o protocolo do sistema mas que tem acesso a um conjunto de dados auxiliares e que observa a distribuição de frequência dos *hashes* dos fragmentos.

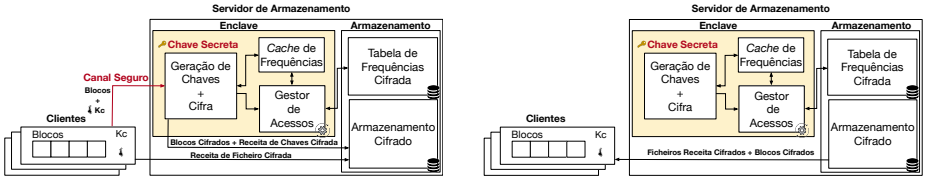
Este adversário tem como objetivo identificar o conteúdo original dos fragmentos cifrados no servidor de armazenamento observando os acessos no servidor (comparando a frequência de acessos de uma dada entrada à frequência de um dado fragmento), os acessos aos próprios dados, e os acessos aos metadados do sistema, especificamente o ponto de entrada para a tabela cifrada no servidor (apresentada nos próximos parágrafos).

4.2 Componentes e Interações

Tal como nos trabalhos anteriores, assumimos que os clientes podem realizar escritas e leituras dos seus dados da forma que desejarem. Isto é feito estabelecendo uma ligação segura com o enclave presente no servidor de armazenamento, que é responsável por cifrar os dados e colocá-los em armazenamento cifrado. O enclave irá escolher a chave para cifrar os dados, tendo em conta a frequência de cada um dos fragmentos [12,14]. Descrevemos agora com mais detalhe a operação de envio e leitura de um ficheiro no nosso sistema.

A operação de envio de dados está representada na Figura 1a. Um cliente que deseje enviar um ficheiro F, primeiro, divide o ficheiro em vários fragmentos (de tamanho fixo, p.ex., 4KB), e constrói um ficheiro receita [12] com base na ordem dos fragmentos no ficheiro. Em seguida, o cliente cifra o ficheiro receita com a chave de cliente Kc (única por cliente) e envia para o servidor de armazenamento. O cliente depois envia todos os fragmentos, e a chave de cliente através do canal de comunicação seguro com o enclave. O enclave gera uma chave para cada fragmento com base na sua frequência, num parâmetro t que indica o número máximo de cópias iguais que podem ser cifradas com a mesma chave, e um segredo conhecido apenas pelo enclave, e cifra cada fragmento com a sua respetiva chave. Após a geração de todos os criptogramas, o enclave constrói uma receita de chaves [12], que contém as chaves para todos os fragmentos que foram cifrados. Cifra essa receita com a chave do cliente Kc, e armazena os fragmentos cifrados e o ficheiro receita cifrado no sistema de armazenamento.

Para ler um ficheiro (ver Figura 1b), um cliente recupera a receita do ficheiro cifrada e a receita de chaves cifrada do sistema de armazenamento, bem como os fragmentos cifrados. De seguida, decifra os ficheiros receita com a chave de cliente



(a) Escrita (b) Leitura

Figura 1: Acesso ao servidor

Kc. Finalmente o cliente decifra cada fragmento cifrado usando a respetiva chave e reconstrói o ficheiro com base na receita do ficheiro.

De forma a prevenir ataques de frequência, é necessário armazenar no enclave uma cache com a frequência de cada fragmento, ou seja, o número de vezes que já foi escrito. Devido ao limite de memória dos enclaves (128 MB) [8], não é possível armazenar a frequência de todos os fragmentos dentro da cache do enclave, p.ex., utilizando SHA-256 é possível obter 2^{256} *hashes* diferentes. Considerando 100MB disponíveis para a cache do enclave, apenas é possível guardar aproximadamente 2M entradas no enclave, uma vez que cada entrada ocupa 42 octetos (incluindo a *hash*, um contador de frequência, e uma referência para a entrada respectiva na tabela de frequências). Em sistemas de armazenamento reais, estas 2M entradas podem representar 5% do número de fragmentos existentes. Sistemas como o S2Dedup [14] têm este tipo de cache, no entanto, quando o limite de memória é atingido a tabela é reiniciada, impedindo o uso da deduplicação para os fragmentos dos quais se perdeu informação.

De forma a ultrapassar este problema, o nosso sistema mantém a frequência de todos os fragmentos, mesmo quando o limite de memória do enclave é atingido. O servidor contém uma segunda tabela de frequências cifrada, fora do enclave, cujos conteúdos apenas podem ser decifrados pelo enclave. O enclave irá consultar esta tabela (como se observa na Figura 1a) quando a memória máxima é atingida, e não é possível armazenar a frequência de novos fragmentos.

Um problema que pode surgir ao utilizar uma tabela numa zona não confiável é que o servidor é capaz de observar o acesso do enclave às entradas desta tabela, conseguindo inferir os conteúdos através da frequência de acessos [10]. Para mitigar este problema, a nossa solução inclui um novo componente interno ao enclave, designado *Gestor de Acessos*. Este componente concretiza políticas de despejo da cache e políticas de acesso à tabela de frequências, com o objetivo de mascarar a frequência de acessos a cada entrada.

4.3 Garantias de Privacidade

De forma a quantificar o nível de privacidade que o nosso componente *Gestor de Acessos* é capaz de oferecer à nossa tabela na zona não confiável, decidimos seguir uma variante do critério de preservação de privacidade usado em PraDa [6], designado por α-*privacy*. Este critério assegura que existem sempre pelo menos

$\frac{1}{\alpha}$ fragmentos com a mesma frequência. No nosso trabalho, consideramos uma generalização deste critério, designado por (α, δ)-*privacy*, que assegura que, dado um fragmento, acedido com uma frequência f, existem sempre pelo menos $\frac{1}{\alpha}$ fragmentos que são acedidos com uma frequência no intervalo $[f-\delta, f+\delta]$. Note-se que quando $\delta = 0$ a (α, δ)-*privacy* é equivalente à α-*privacy*. Esta generalização permite capturar cenários onde a frequência de um dado fragmento não pode ser estimada com exatidão.

5 Estratégias de Prevenção do Ataque de Frequência

Como referido anteriormente, a nossa estratégia para impedir os ataques de frequência reside na utilização do componente *Gestor de Acessos*, que medeia a interação entre a cache de frequência residente no enclave e a tabela de frequências armazenada no exterior, e cuja operação se baseia na combinação de um mecanismo de dispersão de acessos que é aplicado sempre que o enclave acede à tabela de frequência cifrada que se encontra na zona não confiável, e um mecanismo de despejo da cache, que é utilizado para substituir entradas na cache quando esta atinge o tamanho máximo. De seguida, descrevemos algumas das políticas que podem ser concretizadas por estes mecanismos.

5.1 Políticas de Dispersão de Acessos

O mecanismo de troca de informação entre a cache do enclave e a tabela cifrada no servidor pode ser realizada seguindo várias abordagens. Consideramos três políticas: *Acesso Direto*, *Acesso k-Anónimo*, e *Acesso por Grupos*.

- *Acesso Direto:* O componente *Gestor de Acessos* acede diretamente à entrada da tabela que pretende. Esta abordagem tem um bom desempenho a nível computacional e de armazenamento, pois é apenas consultada a entrada desejada no servidor.
- *Acesso k-Anónimo:* Ao invés de aceder diretamente à entrada que pretende na tabela, o componente *Gestor de Acessos* consulta K entradas (a entrada pretendida e $K-1$ aleatórias), por uma ordem também aleatória. Permitindo que fragmentos pouco frequentes passem a ser consultados mais vezes.
- *Acesso por Grupos:* Consiste em armazenar as entradas do servidor em grupos. Para obter a entrada desejada do servidor, o componente *Gestor de Acessos* indica o seu grupo, não o seu *hash*. Desta forma, o servidor irá enviar todas as entradas presentes nesse grupo, e não consegue perceber qual a entrada que foi requisitada. O grupo de uma dada entrada é dado por:

$$\mathit{Hashcode}(h) \mod N$$

em que N é o número de grupos, e *Hashcode* é uma função que retorna um inteiro dado um *hash* h (p.ex., $h[0]*31^{n-1}+h[1]*31^{n-2}+...+h[n-1]$, em que *h[0]* é o primeiro carácter de h, e n é o número de caracteres). Este esquema assume uma distribuição uniforme das entradas pelos grupos, e permite que

fragmentos com acessos menos frequentes partilhem o mesmo grupo com os fragmentos com acessos frequentes, aumentando assim a frequência de acesso dos primeiros.

5.2 Políticas de Despejo de Cache

Quando é necessário armazenar a frequência de um fragmento mas a cache do enclave já está cheia, é necessário despejar uma das entradas, de forma a armazenar a nova dentro do enclave. Para este efeito, consideramos 4 políticas de despejo: *Aleatório* (escolhe um elemento presente na cache do enclave de forma aleatória para despejar), *Acedido Há Mais Tempo* (escolhe o elemento que foi acedido há mais tempo na *cache* do enclave), *Menos Frequente* (escolhe o elemento com menor frequência presente na *cache* do enclave), e *Menos Acedido no Exterior* (escolhe o elemento com menos acessos à tabela cifrada no servidor, que esteja presente na cache do enclave).

5.3 Combinação de Políticas

As diferentes políticas de dispersão de acessos podem ser combinadas com as diferentes políticas de despejo da cache, permitindo 12 modos de operação distintos do componente *Gestor de Acessos*. Foram avaliadas as 12 combinações possíveis, e os resultados obtidos são apresentados na secção seguinte.

6 Avaliação Experimental

Nesta secção avaliamos as poupanças de armazenamento da nossa solução, e analisamos as garantias de privacidade nos acessos à tabela cifrada no servidor.

Para avaliarmos a distribuição das frequências de acesso à tabela cifrada do servidor, foi analisado um conjunto de dados suscetível a ataques de frequência, que contém um número de fragmentos com uma frequência elevada. Este conjunto artificial de dados foi obtido usando um gerador de dados que segue uma distribuição Zipfiana (ver Figura 2) com 10K fragmentos distintos e 100M acessos. A cache presente no enclave consegue armazenar 500 entradas, logo só consegue armazenar 5% das frequências dos fragmentos no enclave (zona verde).

6.1 Análise das Poupanças de Armazenamento

O objetivo de ter uma segunda tabela de frequências é armazenar as frequências de todos os fragmentos, maximizando o ganho de deduplicação. Para verificarmos o ganho de deduplicação, simulámos a poupança de armazenamento para o conjunto de dados apresentados na Figura 2 quando é utilizada a solução S2Dedup (com $t = 350$, quando a frequência de um fragmento atinge t uma nova chave é usada para o cifrar [12]) e a nossa solução (com $t = 350$). O conjunto de dados analisados representa aproximadamente 381.5 GB (fragmentos de tamanho 4KB). Estas simulações foram feitas variando o tamanho da cache.

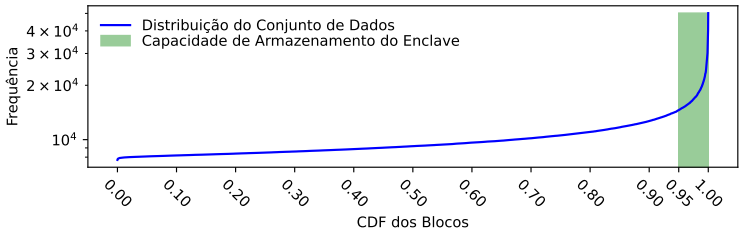


Figura 2: Distribuição de frequência dos fragmentos no conjunto de dados

Tabela 1: Poupança de armazenamento no S2Dedup e na nossa solução

Percentagem de entradas que podem ser armazenadas no enclave		5%	50%	95%	100%
Poupança de armazenamento	S2Dedup	2.7%	29.3%	70.1%	99.74%
	Nossa Solução	99.74%	99.74%	99.74%	99.74%

Como o S2Dedup precisa de reiniciar a tabela de frequências dentro do enclave sempre que alcança o seu tamanho máximo, o S2Dedup está limitado pelo tamanho do enclave para oferecer poupanças de armazenamento através da deduplicação. Por outro lado, o nosso sistema tira partido de uma tabela na zona não confiável, capaz de tolerar esta limitação, assim é capaz de aproveitar ao máximo a deduplicação de fragmentos. Como é observável na Tabela 1 a nossa solução oferece sempre o máximo de poupança de armazenamento, enquanto o S2Dedup tem muita dificuldade em aplicar deduplicação quanto mais pequena for a memória do enclave. Apenas é capaz de poupar 2.7% de armazenamento quando o enclave tem capacidade de manter 5% das entradas.

6.2 Análise dos Acessos ao Armazenamento não Confiável

Para avaliar a informação que um servidor malicioso pode extrair através dos acessos à sua tabela cifrada, foi simulada a execução de cada uma das estratégias para o conjunto de dados apresentado anteriormente. Além do tipo de estratégia utilizada para consultar a tabela do servidor, quisemos entender se as políticas de despejo da cache influenciam de alguma forma o número de acessos. A Figura 3 mostra os resultados obtidos.

A distribuição de frequências do conjunto de dados analisados está representada a azul (a mesma que está apresentada na Figura 2), de forma a retratar o efeito das políticas na frequência de acessos. A distribuição dos acessos à tabela cifrada, com o uso das estratégias *Acessos Diretos*, *Acessos K-Anónimos* e *Acessos por Grupos*, está representada a vermelho, laranja e verde, respetivamente.

Quando aplicada a política de despejo de cache *Aleatório*, as estratégias *Acesso Direto* e *Acesso k-Anónimo* demonstram o mesmo tipo de acessos que a distribuição original, porém, os acessos são mais acentuados usando *Acesso k-Anónimo*, pois o número de acessos torna-se superior ao original. A estraté-

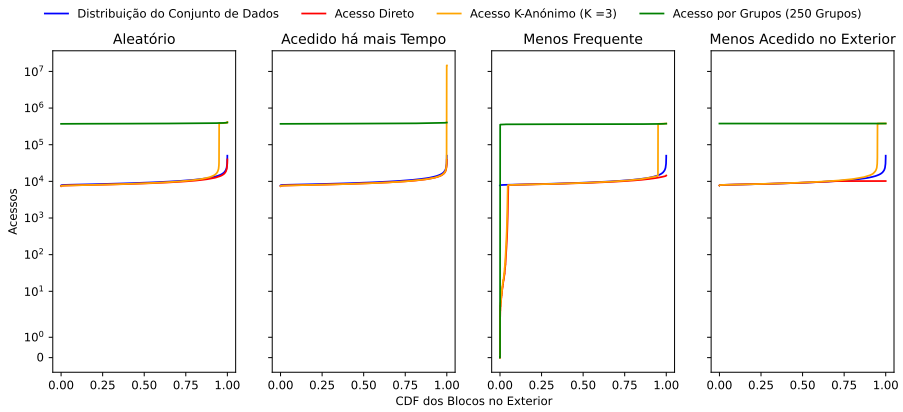


Figura 3: Distribuição dos acessos à tabela cifrada do servidor com diferentes políticas de cache para cada abordagem

gia *Acesso por Grupos* consegue alterar a frequência de acessos dos fragmentos, porém aumenta o número de acessos.

Para as estratégias *Acesso Direto* e *Acesso K-Anónimo*, a política *Acedido há mais Tempo* não faz uma boa gestão dos elementos que são despejados, pois tem apenas em conta a ordem pelo qual os fragmentos são acedidos. A estratégia *Acesso por Grupos* mantém o comportamento anterior.

A política de despejo *Menos Frequente* garante que são mantidos dentro do enclave os elementos mais frequentes, no entanto, os elementos que são menos frequentes irão ser sempre despejados, e sempre que sejam necessários, irá ser preciso realizar um acesso ao servidor. Nas estratégias *Acesso Direto* e *Acesso k-Anónimo*, isto reduz o número de acessos a alguns elementos. A estratégia *Acesso por Grupos* apresenta uma redução no número de acessos, mas continua a exibir um número de acessos muito superior às restantes abordagens.

Finalmente, a política de despejo *Menos Acedido no Exterior* é a que melhor consegue normalizar a frequência de acessos a cada elemento na tabela do servidor. Isto é possível observar especialmente com as estratégias *Acesso Direto* e *Acesso por Grupos*, que mais se aproximam de uma linha reta. Este efeito não é fácil de obter com a abordagem *Acesso K-Anónimo* porque apesar de despejar os elementos menos acedidos no exterior, os seus acessos são aleatórios, fazendo com que alguns elementos continuem a ser acedidos muitas mais vezes.

6.3 Análise das Garantias de Privacidade

De forma a observar a (α, δ)-*privacy* oferecida por cada combinação, foi aplicado o critério apresentado na Secção 4.3 ao conjunto de dados analisado, para diferentes valores de δ. Os resultados obtidos são apresentados na Figura 4.

Observa-se que o valor de α está compreendido entre $1/N$ e 1, em que N é o número de fragmentos distintos. Quanto maior o valor de $1/\alpha$ e menor o valor de δ, maior a segurança oferecida pela combinação.

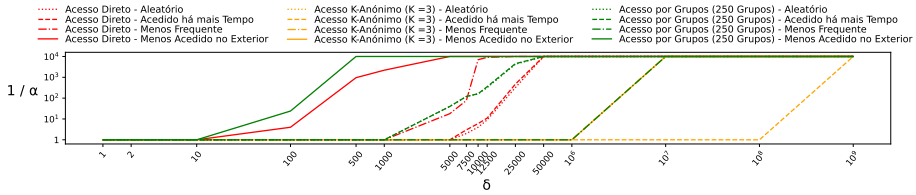


Figura 4: Garantias de privacidade oferecidas por cada combinação

A combinação *Acessos por Grupos* com a política de despejo *Menos Acedido no Exterior* é a que consegue a melhor (α, δ)-*privacy*. Com δ igual a 100, existe para cada fragmento, pelo menos 24 outros fragmentos ($24(\alpha, \delta)$-*privacy*) com uma frequência com diferença máxima de 100. E com δ igual a 500, todos os 10K fragmentos tem uma diferença máxima de frequência de 500 entre eles.

A combinação *Acesso Direto* com a política *Menos Acedidos no Exterior* é a segunda melhor combinação, atingindo $10\text{K}(\alpha, \delta)$-*privacy*, com δ igual a 5K.

As outras combinações conseguem obter (α, δ)-*privacy* mas para valores de δ muito superiores, o que torna mais fácil distinguir os fragmentos entre eles, tornando o uso destas combinações menos seguro.

6.4 Discussão

Observa-se que a estratégia *Acesso por Grupos* é a que consegue manter as frequências de acesso dos fragmentos próximas umas das outras, independentemente da política de cache utilizada, tornando mais difícil distinguir fragmentos. Isto acontece porque os fragmentos menos frequentes podem ser colocados no mesmo grupo que os fragmentos mais frequentes, alterando a frequência de acesso destes, tornando mais difícil realizar ataques de análise de frequência. No entanto, esta estratégia é computacionalmente intensiva, aumentando o número de acessos em mais de uma ordem de grandeza. Com a política *Menos Acedidos no Exterior*, a estratégia *Acesso Direto* também consegue manter as frequências de acesso dos fragmentos próximas umas das outras, mas com um número de acessos mais baixo. Porém, o uso desta política de despejo implica um uso maior da memória dentro do enclave, o que pode causar problemas de desempenho.

7 Conclusões

Neste trabalho apresentamos uma solução baseada em Intel SGX que é capaz de suportar deduplicação enquanto mantém fortes níveis de privacidade, evitando ataques de análise de frequência. Neste contexto, propomos e comparamos diversas combinações de estratégias e políticas de cache para prevenir os ataques de análise de frequência.

Agradecimentos: Agradecimentos a Rafael Soares pela troca de ideias e ajuda durante o desenvolvimento deste trabalho. Este trabalho foi suportado pela FCT – Fundação para a Ciência e a Tecnologia, através da bolsa 2020.05270.BD e do projecto NGSTORAGE (financiado pelo OE com a ref. PTDC/CCI-INF/32038/2017) e por fundos nacionais através do projeto UIDB/50021/2020.

Referências

1. Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In: International Conference on the Theory and Applications of Cryptographic Techniques. Athens, Greece (May 2013)
2. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. Journal of Cryptology **16**, 185–215 (2003)
3. Camenisch, J., Neven, G., et al.: Simulatable adaptive oblivious transfer. In: International Conference on the Theory and Applications of Cryptographic Techniques. Barcelona, Spain (May 2007)
4. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO. Santa Barbara, CA, USA (Jan 1983)
5. Correia, C., Correia, M., Rodrigues, L.: Omega: a secure event ordering service for the edge. In: International Conference on Dependable Systems and Networks. Valencia, Spain (Jun 2020)
6. Dong, B., Liu, R., Wang, W.: PraDa: Privacy-preserving data-deduplication-as-a-service. In: International Conference on Conference on Information and Knowledge Management. Shanghai, China (Nov 2014)
7. Dropbox: Available at https://dropbox.com/, accessed: 2022-07-18
8. El-Hindi, M., Ziegler, T., Heinrich, M., Lutsch, A., Zhao, Z., Binnig, C.: Benchmarking the second generation of intel SGX hardware. In: International Conference on Management of Data. Philadelphia, USA (Jun 2022)
9. Google Drive: Available at https://www.google.com/drive/, accessed: 2022-07-18
10. Jurado, M., Smith, G.: Quantifying information leakage of deterministic encryption. In: International Conference on Cloud Computing Security Workshop. London, United Kingdom (Nov 2019)
11. Keelveedhi, S., Bellare, M., Ristenpart, T.: Dupless: Server-aided encryption for deduplicated storage. In: Security Symposium USENIX Security). Washington, D.C. (Aug 2013)
12. Li, J., Yang, Z., Ren, Y., Lee, P., Zhang, X.: Balancing storage efficiency and data confidentiality with tunable encrypted deduplication. In: European Conference on Computer Systems. Heraklion, Greece (Apr 2020)
13. Meyer, D., Bolosky, W.: A study of practical deduplication. ACM Transactions on Storage **7**, 1–20 (2012)
14. Miranda, M., Esteves, T., Portela, B., Paulo, J.a.: S2Dedup: SGX-enabled secure deduplication. In: International Conference on Systems and Storage. Haifa, Israel (Jun 2021)
15. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. Journal of the ACM (JACM) **51**, 231–262 (2004)
16. Ren, Y., Li, J., Yang, Z., Lee, P.P., Zhang, X.: Accelerating encrypted deduplication via SGX. In: USENIX Annual Technical Conference. Remotely (Jul 2021)
17. SGX101: Overview - SGX 101. Available at https://sgx101.gitbook.io/sgx101/sgx-bootstrap/overview, accessed : 2022-07-18
18. Wallace, G., Douglis, F., Qian, H., Shilane, P., Smaldone, S., Chamness, M., Hsu, W.: Characteristics of backup workloads in production systems. In: International conference on File and Storage Technologies. San Jose, CA (Feb 2012)