Pesquisa Booleana Cifrada usando Hardware Confiável*

Guilherme Borges, João Leitão, Henrique Domingos, e Bernardo Ferreira

NOVA LINCS & DI-FCT-UNL
g.borges@campus.fct.unl.pt, {jc.leitao, hj, bf}@fct.unl.pt

Resumo O crescente uso de serviços na Nuvem acarreta um aumento nas preocupações relativas à segurança dos sistemas que armazenam e processam dados remotamente. Soluções para proteger dados guardados na Nuvem devem procurar um balanceamento entre três dimensões: segurança, desempenho e usabilidade. Este balanceamento é particularmente relevante em grandes repositórios de dados, onde o suporte de pesquisas expressivas é uma funcionalidade crucial. Soluções propostas recentemente tendem a sacrificar desempenho e usabilidade em função de segurança, propondo sistemas pouco atractivos do ponto de vista prático. Neste artigo propomos um novo esquema de Pesquisa Booleana Confiável – PBC. O PBC suporta pesquisas booleanas sobre repositórios de texto cifrados, incluindo combinações arbitrariamente complexas de conjunções, disjunções e negações. A nossa solução alavanca num módulo de hardware confiável instalado na Nuvem, nomeadamente o Intel SGX, permitindo-nos melhorar simultaneamente as três frentes referidas acima. Tal é conseguido através da terceirização segura de computações sensíveis para a Nuvem, aproveitando o seu poder computacional e reduzindo a carga comunicacional cliente-servidor em relação a esquemas meramente criptográficos. O esquema apresentado foi implementado e avaliado experimentalmente, demonstrando a sua usabilidade e escalabilidade em relação ao estado da arte.

Palavras–chave: Segurança na Nuvem · Pesquisa Booleana · Hardware Confiável.

1 Introdução

Soluções de repositórios de dados na Nuvem estão a ser cada vez mais utilizadas por parte de utilizadores individuais e empresas, dadas as suas garantias de alta disponibilidade e elevado poder computacional por baixo custo, em comparação com soluções tradicionais. Contudo, a tendência para mover dados e computações para a Nuvem pode expôr dados sensíveis aos fornecedores desses mesmos serviços, originando fugas de dados [11,1,7]. Enquanto que, numa

* Trabalho parcialmente financiado pela UE, através do projecto LightKone (H2020 grant agreement ID 732505), e pela FCT/MCTES, através do projecto estratégico NOVA LINCS (UID/CEC/04516/2013) e do projecto #PTDC/CCI-INF/32662/2017 com financiamento FEDER.

solução tradicional, os utilizadores poderiam garantir a privacidade dos seus dados ao usar servidores locais sob o seu controlo, tal é mais difícil de garantir na Nuvem. Tais riscos são tidos como graves e, por vezes, impeditivos de uma adopção plena destes serviços [8].

Técnicas como *security at rest* [19,18] oferecem uma solução inicial para o problema. Nesta abordagem, os dados nunca são revelados em claro ao provedor da Nuvem, porém, todas as computações têm de ser executadas no cliente, incluindo operações de cifra e decifra – criando uma carga adicional na rede e comprometendo o desempenho geral do sistema com cada operação. Estas limitações tornam-se particularmente notórias em cenários com dispositivos de recursos limitados, como dispositivos móveis, dados os seus constrangimentos de comunicação, processamento e energia em comparação com soluções tradicionais.

Torna-se, assim, fundamental permitir computações sobre dados cifrados de forma eficiente, particularmente pesquisas. Estas são especialmente relevantes dado o grande volume de dados produzido hoje em aplicações baseadas na Nuvem. Técnicas como Cifras Simétricas Pesquisáveis (CSP) [22,6] suportam pesquisas sobre dados cifrados com garantias de confidencialidade, apresentando desempenho superior às técnicas anteriores. Para melhorar o desempenho das pesquisas são criados índices cifrados na Nuvem, sendo as pesquisas feitas com um *token* criptográfico gerado pelo cliente. Para aliviar o número de rondas de comunicação entre cliente e servidor, estes esquemas delegam algumas computações para a Nuvem; no entanto, tal pode revelar informação sobre os dados como, por exemplo, os identificadores de documentos retornados por uma pesquisa ou a popularidade (repetição) de cada pesquisa, comprometendo a privacidade passada e futura: se, respectivamente, uma actualização corresponde a documentos pesquisados anteriormente, e se pesquisas revelam documentos apagados anteriormente. Adicionalmente, o contínuo uso de um sistema gradualmente revela padrões de informação, abrindo a oportunidade a ataques por inferência. Deste modo, torna-se essencial mitigar estes efeitos mantendo um bom desempenho.

Hardware confiável moderno, disponível em processadores comuns (como Intel SGX [14] ou ARM TrustZone [3]), torna possível a execução eficiente de computações e manipulação de dados sensíveis em servidores na Nuvem sem prejuízo da confidencialidade dos mesmos. Estes módulos executam programas com garantias de isolamento em relação a qualquer outro processo a correr na mesma máquina, incluindo o sistema operativo.

Neste artigo propomos o PBC (Pesquisa Booleana Confiável), um novo esquema de CSP que suporta pesquisas booleanas de palavras-chave com número arbitrário de conjunções, disjunções, negações e parêntesis sobre documentos de texto (e.g., $A \wedge (B \vee \neg C)$). O PBC utiliza um módulo de hardware confiável na Nuvem (Intel SGX), de forma a melhorar as condições de segurança e desempenho em relação ao estado da arte. O uso deste módulo permite-nos reduzir de forma eficiente a revelação de padrões durante a execução de operações, endereçar privacidade passada e futura e verificar a integridade e resultado das operações. Tal é feito, em parte, através da delegação de computações sobre da-

dos privados para o módulo confiável na Nuvem, reduzindo o número de rondas de comunicação cliente–servidor em relação a uma versão sem o mesmo, e assim abrindo caminho ao uso do esquema por parte de clientes com dispositivos de recursos limitados.

O resto do artigo está organizado da seguinte forma: na secção 2 é apresentado o estado da arte em relação às CSP e hardware confiável; na secção 3 é apresentado o modelo de sistema, incluindo o modelo de adversário considerado; na secção 4 apresentamos o esquema PBC e os seus respectivos protocolos; na secção 5 descrevemos a implementação e avaliação experimental do sistema; finalmente, a secção 6 conclui o artigo.

2 Trabalho Relacionado

A área das Cifras Simétricas Pesquisáveis (CSP) iniciou-se com o trabalho de Song et al. [22]. Este trabalho considerava apenas pesquisas de palavras individuais sobre documentos de texto com desempenho linear. Mais tarde, Curtmola et al. [10] propuseram a adição de um índice invertido, mapeando palavras para documentos e melhorando o desempenho das pesquisas. Neste trabalho foram também introduzidas as primeiras noções de segurança formais em CSP.

Mais recentemente, Kamara et al. [16] introduziram o primeiro esquema de CSP dinâmico, permitindo a adição e remoção de palavras a documentos sem necessidade de uma re-indexação total da base de dados.

Privacidade passada e futura foram primeiro definidas por Stefanov et al. [23], sendo proposto um esquema abordando a segunda. Mais tarde, Bost et al. [5] propuseram um esquema que preservava ambas as propriedades, a troco de desempenho e armazenamento.

A maioria dos esquemas propostos foi desenhado para pesquisas de uma única palavra; porém, alguns esquemas contemplam pesquisas booleanas. Golle et al. [13] foram os primeiros a propor um esquema para pesquisas booleanas conjuntivas. Cash et al. [6] apresentaram um esquema que suporta pesquisas conjuntivas e disjuntivas com desempenho linear. Mais recentemente, Kamara e Moatz [15] propuseram o esquema IEX-2LEV, permitindo pesquisas na Forma Normal Conjuntiva (FNC). Apesar de exibir desempenho sub-linear nas pesquisas, este esquema requer armazenamento quadrático no servidor e revela bastante mais informação durante a realização de operações que esquemas com pesquisas de uma só palavra.

O conceito de computação confiável foi introduzido por Santos et al. [21] no contexto de computação na Nuvem; de acordo com este modelo é obtida uma plataforma de computação confiável na Nuvem, com garantias de atestação remota e não-manipulação de computações aí executadas. Barbosa et al. [4] expandiram esta noção com Ambientes de Execução Confiáveis (AEIs), uma abstracção que assegura o isolamento de computações dentro de ambientes não confiáveis através de primitivas de atestação remota e validação de computações executadas nesses ambientes.

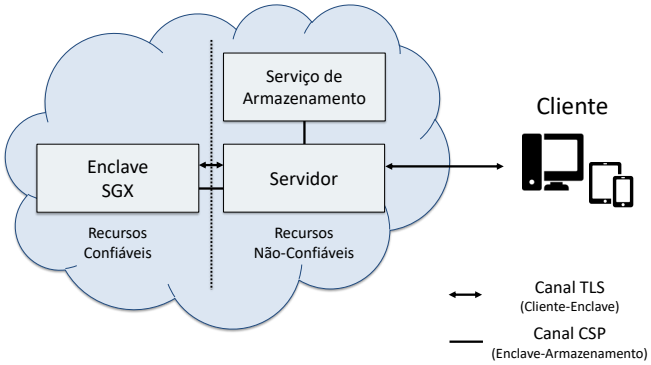


Figura 1: Arquitectura do PBC.

Soluções de hardware recentes, tais como o Intel SGX [2,14,17] ou o ARM TrustZone [3], disponibilizam modelos que garantem as propriedades de um AEI. Tais soluções têm como vantagem a sua ubiquidade, dado estarem presentes em processadores comuns e apresentarem ferramentas de desenvolvimento que facilitam a sua integração em *software* pré-existente. Soluções recentes têm explorado a integração destes módulos de hardware em diferentes contextos, tais como a aprendizagem automática [20]. Fuhry et al. [12] propuseram um esquema de CSP usando SGX para efectuar pesquisas de intervalo, um problema relacionado, mas ainda assim fundamentalmente diferente do endereçado neste artigo.

3 Modelo de Sistema

O nosso sistema considera um modelo cliente–servidor baseado na Nuvem, como apresentado na Figura 1. O PBC é composto por quatro componentes: o Cliente (utilizador do sistema), um Enclave Intel SGX, um Servidor, que inicializa o Enclave e serve depois como intermediário para comunicações com o mesmo, e um Serviço de Armazenamento na Nuvem (serviço externo não confiável usado para persistir grandes volumes de dados, dada a memória limitada do Intel SGX [9]).

Este modelo só considera um cliente; estendê-lo para vários clientes poderia ser conseguido usando um esquema de chave pública/privada para autenticação dos clientes. Para permitir separar documentos de diferentes clientes, e controlar as respectivas permissões, o PBC poderia ser complementado com um esquema de controlo de acessos padrão no Enclave.

A principal contribuição do PBC é a de considerar um elemento de confiança do lado do servidor, responsável por computações sobre dados sensíveis; na sua ausência seria necessário adoptar mecanismos criptográficos complexos para preservar a segurança de tais computações.

Operação do PBC. O esquema PBC contempla duas fases: inicialização e operação, que descrevemos de seguida.

Na fase de inicialização, o Cliente contacta o Servidor para inicializar um Enclave PBC. Daí em diante, o Servidor encaminhará todas as comunicações Cliente-Enclave através de *ecalls* (chamadas Intel SGX para passar dados para o Enclave e executar o código do mesmo). O Cliente efectua um *handshake* TLS com o Enclave e autentica o mesmo através do mecanismo padrão do Intel SGX [2]. Nesta etapa, o Enclave inicializa também, no Serviço de Armazenamento, as estruturas necessárias para o seu funcionamento, ficando estabelecido um canal entre ambos. A segurança destas estruturas, e consequentemente do canal, é garantida por técnicas de CSP, como explicado em mais detalhe na Secção 4. A comunicação Enclave–Serviço de Armazenamento é abstraída por *ocalls* (chamadas originadas dentro do Enclave que provocam uma saída temporária do mesmo, durante a qual é feita a comunicação com o Serviço de Armazenamento).

Na fase de operação, o Cliente utiliza o PBC para adicionar e apagar palavras em documentos de texto, bem como fazer pesquisas booleanas sobre os mesmos. Nesta fase, o Cliente será responsável por manter pequenas estruturas de dados auxiliares e efectuar algum pré-processamento aos dados, antes de os enviar para o PBC. Computações mais pesadas, que requerem consultas a grandes volumes de dados, apenas ocorrem no Enclave, permitindo a existência de Clientes com baixo poder computacional e de armazenamento.

Base de Confiança e Modelo de Adversário. A base de confiança do sistema assenta no Cliente e no Enclave, sendo o Servidor, o Serviço de Armazenamento e canais de comunicação entre eles considerados não-confiáveis. Assumimos a existência de adversários totalmente maliciosos, que não só observam todos os dados e computações, como tentam também interferir com os mesmas. No entanto, não consideramos ataques de negação de serviço, relegando estes para trabalho futuro.

4 Solução

Nesta secção apresentamos o esquema PBC (Figura 2), composto por três protocolos: *Inicialização*, *Actualização* e *Pesquisa*. As funções criptográficas necessárias são uma função de *hash* com chave (por exemplo, HMAC SHA256), e uma função de cifra simétrica autenticada. O Enclave tem a capacidade de receber e enviar mensagens (receber()*e*enviar()), mediado pelo Servidor, bem como de executar operações sobre o Serviço de Armazenamento (Armazenamento.{Iniciar, Escrever, Ler}), sendo este último acedido através de uma interface do tipo dicionário, com operações escrita/leitura.

O protocolo de *Inicialização* recebe como argumento um parâmetro de segurança λ e inicia o canal seguro entre Cliente e Enclave. Quando o Cliente se conecta ao Servidor, este inicializa o Enclave PBC e, doravante, reencaminha todo o tráfego TCP do Cliente para esse Enclave (linhas 6 e 9). O Cliente e o Enclave negoceiam uma ligação TLS. No Cliente, um dicionário de contadores W irá associar cada palavra única no repositório a um inteiro, representando o número de operações de escrita efectuadas sobre cada palavra. No lado do

```
Inicialização(λ)
    Cliente:
 1: k_F ← GerarChave(λ)
 2: W ← [ ]
 3: Servidor.inicializar()
    Servidor:
 4: Enclave.inicializar()
    Cliente:
 5: SGX ← TLS.começarHandshake(Enclave)
    Servidor:
 6: Encaminhar tráfego TCP para o Enclave
    Enclave:
 7: TLS.acabarHandshake()
    Cliente:
 8: SGX.autenticar()
    Servidor:
 9: Encaminhar tráfego TCP para o Enclave
    Enclave:
10: nDocs ← 0
11: k_E ← GerarChave(λ)
12: Armazenamento.Iniciar(I)

Actualização(op, w, id)
    Cliente:
 1: k_w ← HMAC(k_F, w)
 2: c ← W[w]
 3: if c = ⊥ then
 4:     c ← 0
 5: else
 6:     c ← c + 1
 7: end if
 8: SGX.enviar({op, id, c, k_w})
 9: W[w] ← c
    Servidor:
10: Encaminhar tráfego TCP para o Enclave.
    Enclave:
11: receber({op, id, c, k_w})
12: l ← HMAC(k_w, c)
13: id* ← Symm.Enc(k_E, (l, op, id))
14: Armazenamento.Escrever(I, l, id*)
15: if id > nDocs then
16:     nDocs ← nDocs + 1
17: end if

Pesquisa(q)
    Cliente:
 1: {w̄, φ} ← ProcessarPesquisaBooleana(q)
 2: C ← [ ]
 3: for all w ∈ w̄ do
 4:     k_w ← HMAC(k_F, w)
 5:     c ← W[w]
 6:     C ← C ∪ {k_w, c}
 7: end for
 8: SGX.enviar({C, φ})
    Servidor:
 9: Encaminhar tráfego TCP para o Enclave.
    Enclave:
10: receber({C, φ})
11: Q ← [ ]
12: for all {k_w, c} ∈ C do
13:     L ← [ ]
14:     for all c_i ← 0 . . . c do
15:         l ← HMAC(k_w, c_i); L ← L ∪ l
16:     end for
17:     Q ← Q ∪ {k_w, L}
18: end for
19: Q ← Nivelar(Q); Q ← PermutaçãoAleatória(λ, Q)
20: D′ ← Armazenamento.Ler(I, Q)
21: D ← [ ]
22: for all id* ∈ D′; l′ ∈ Q′ do
23:     (l, op, id) ← Symm.Dec(k_E, id*); Verificar(l, l′)
24:     D ← D ∪ {op, id}
25: end for
26: Q′ ← Juntar(Q, D)
27: R ← Resolver(φ, Q′, nDocs)
28: enviarCliente(R)
    Servidor:
29: Encaminhar tráfego TCP para o Cliente
    Cliente:
30: R ← SGX.receber(R)
```

Figura 2: Esquema PBC composto pelos protocolos *Inicialização*, *Actualização* e *Pesquisa*.

Enclave é inicializado o contador nDocs, que mantém o número de documentos existente na base de dados. Este será usado na resolução de pesquisas booleanas no protocolo *Pesquisa*. O Enclave é ainda responsável por inicializar o Serviço de Armazenamento, representado por um dicionário I. O protocolo de *Inicialização* é ainda responsável pela inicialização das chaves criptográficas: k_F e k_w, a chave do HMAC e da função de cifra simétrica autenticada Symm, respectivamente.

O protocolo *Actualização* é usado para adicionar ou remover termos de documentos, e recebe como argumento op (operação de adicionar ou remover), w (o termo a actualizar) e id (identificador do documento a alterar). O protocolo apresentado é definido em função de um único termo, sendo trivial a sua extensão para operações sobre múltiplos termos. O Cliente inicia o algoritmo gerando, com um HMAC, uma chave k_w sobre w (linha 1 do protocolo), o que permite ocultar o tamanho de w ao enviá-lo pela rede. O Cliente verifica e incrementa o valor do contador em W para o termo, enviando o tuplo $\{\mathsf{op}, \mathsf{id}, c, \mathsf{k}_w\}$ para o Enclave. No Enclave é gerado o par chave/valor a colocar em I; a chave l é um

HMAC gerado em função de k_w e c, o que garante a geração de uma entrada única em I, pois, para cada k_w, nunca há dois valores c iguais. O valor id^* é um tuplo cifrado $\{l, \mathsf{op}, \mathsf{id}\}$, que é usado no protocolo *Pesquisa* para resolver a pesquisa booleana, garantindo a integridade do par através da repetição do valor da chave l no tuplo. O par é então enviado para o Armazenamento, sendo incrementado o valor de nDocs caso a operação tenha incidido num novo documento. Notamos que uma operação de apagar não corresponde a uma remoção do termo no Armazenamento, mas sim a uma adição desse termo com $op = apagar$. No processamento de uma pesquisa todas as operações sobre um termo serão retornadas, sendo considerada a última operação para cada par termo/documento. Tal permite tornar operações de apagar indistinguíveis das de adicionar.

Por fim, descrevemos o protocolo *Pesquisa*, que recebe uma pesquisa booleana q. O Cliente cria um dicionário C de pares (k_w, c) para cada termo de q. Para cada par (k_w, c), o Enclave gera então os pares (l, id^*), de forma similar ao protocolo *Actualização*. Os pares gerados são depois solicitados ao Armazenamento por ordem aleatória (linhas 19 a 21); ao receber cada par é feita a decifra do valor e a verificação da integridade do mesmo (linha 23). O Enclave guarda as respostas num dicionário Q' cujas chaves correspondem aos termos da pesquisa e os valores aos documentos onde cada termo se encontra. A pesquisa é assim resolvida usando operações de conjuntos sobre Q'. Para realizar negações torna-se necessário ter conhecimento do conjunto total de documentos; para o efeito é utilizada a variável nDocs; assim, são solicitados os documentos do termo a negar ao Armazenamento e depois o conjunto resultante negado em função de $nDocs$. Finalmente, o Enclave devolve o resultado ao Cliente pelo canal seguro.

4.1 Análise de Segurança

No resto desta secção iremos abordar informalmente a segurança do PBC, incidindo primeiro em cada componente e considerando o Cliente como seguro por definição. No Enclave, garantias de confidencialidade e integridade são dadas pelo Intel SGX [2,14,17]. Para mais, as garantias de atestação do Intel SGX permitem ao Cliente assegurar que a versão do PBC a correr no Enclave é correcta.

O PBC oferece garantias de verificação nos dados guardados no Serviço de Armazenamento. Cada par etiqueta/valor contém, no valor, a etiqueta respectiva; deste modo, o Enclave pode garantir a correspondência entre os mesmos ao consultá-los. Para mais, o valor é cifrado com uma cifra autenticada, permitindo a sua repudiação em caso de manipulação. Deste modo, o PBC garante que os valores armazenados externamente são confiáveis.

O canal TLS Cliente–Enclave garante a integridade e confidencialidade das comunicações entre ambos. Ao fazer uma actualização é escondido o tamanho da palavra a adicionar (dado que é enviada como um HMAC de tamanho fixo); numa pesquisa é revelado o número de termos dessa pesquisa, mas não o seu formato booleano. No canal Enclave–Serviço de Armazenamento, o tráfego é composto por pares chave/valor cifrados, cuja autenticidade e integridade é verificada no Enclave. Como tal, o PBC apenas revela os padrões de acesso ao Armazenamento a um adversário, isto é, que valores foram acedidos numa consulta.

Por fim, o PBC garante ainda privacidade passada e futura. A privacidade passada é baseada na definição de Bost et al. [5]: uma adição da palavra w e a sua subsequente eliminação são indistinguíveis em pesquisas com essa mesma palavra. No PBC, actualizações (adições e eliminações) são efectivamente iguais para o Serviço de Armazenamento. Ao fazer uma pesquisa sobre w, o PBC solicita todas as entradas referentes a essa palavra, efectuando o processamento relativo ao tipo de operação no Enclave, sendo estas portanto indistinguíveis para um adversário. A privacidade futura requer que pesquisas antigas não possam ser co-relacionadas com novas actualizações. Dado que cada entrada no Armazenamento é independente de todas as outras, e na operação de pesquisa apenas são solicitadas entradas (isto é, o Servidor não recebe nenhum *token* ou chave criptográfica que pudesse alavancar no futuro), podemos afirmar que o esquema preserva igualmente a privacidade futura.

5 Implementação e Avaliação Experimental

Implementámos um protótipo do PBC em C\C++, recorrendo ao SDK disponibilizado para o Intel SGX. O protótipo desenvolvido contém cerca de 6 mil linhas de código. O HMAC descrito na secção anterior foi instanciado por um HMAC de SHA256 e a função Symm pela cifra simétrica autenticada XSalsa20 com MAC Poly1305. O Serviço de Armazenamento foi implementado com um mapa padrão da biblioteca *standard* do C++.

A nossa avaliação experimental pretende aferir a latência e escalabilidade das operações de actualização e pesquisa, bem como o peso de cada participante do protocolo nas mesmas. Adicionalmente, pretendemos medir a influência de vários factores em operações de pesquisa, nomeadamente a quantidade de termos, o seu formato e a selectividade dos termos, isto é, a percentagem de artigos retornados por cada termo pesquisado. Por fim, comparamos o desempenho do PBC com o estado-da-arte [15].

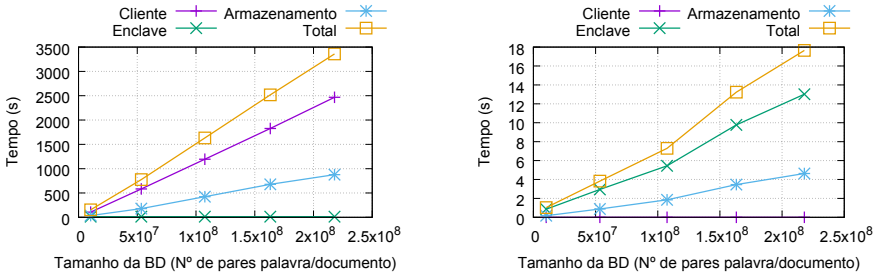
5.1 Banco de testes

A avaliação foi executada num Intel NUC com processador i3 7100U e 8GB de RAM. Dada a limitação de memória desta máquina, o Serviço de Armazenamento na Nuvem foi instanciado numa *workstation* na rede local, com CPU AMD Opteron 6272 e 64GB de RAM.

Os testes usaram cerca de um terço do *dataset* da Wikipedia inglesa, correspondendo a 200 milhões de pares termo/documento e 21 GB de dados não comprimidos; as pesquisas efectuadas foram compostas pelos termos mais populares da língua inglesa (por exemplo, *time*, *person*, *year* ou *way*), combinadas em pesquisas conjuntivas e disjuntivas, com número variável de negações.

5.2 Resultados

Os resultados serão apresentados por latência em função do crescente tamanho do repositório de dados, em termos de pares termo/documento, num intervalo



(a) Desempenho do protocolo *Actualização*. (b) Desempenho do protocolo *Pesquisa* para uma conjunção de cinco termos.

Figura 3: Desempenho dos protocolos *Actualização* e *Pesquisa* em tempo total e particionado entre Cliente, Enclave e Armazenamento.

entre 10 e 220 milhões de pares, e com uma amostragem feita aproximadamente de 50 em 50 milhões. Os resultados correspondem a uma média de 50 execuções.

Desempenho por Participante. Os primeiros resultados inferidos relacionam-se com o desempenho de cada participante (Cliente, Enclave e Serviço de Armazenamento) nos protocolos de *Actualização* e *Pesquisa*.

Na Figura 3a apresentamos o protocolo de actualização, podendo-se observar um aumento linear da latência com o tamanho da operação. A maior fatia de processamento ocorre no Cliente, já que o mesmo é responsável pelo pré-processamento dos artigos e remoção de termos vazios e repetições. O Enclave apenas executa duas computações criptográficas, pelo que apresenta uma latência muito baixa em relação aos outros componentes.

Na pesquisa (Figura 3b), a maior fatia de processamento ocorre no Enclave, já que este recebe e processa a totalidade dos documentos contendo os termos pesquisados, efectuando adicionalmente operações de conjuntos sobre esses documentos. O Cliente apenas executa uma computação criptográfica e um acesso a um dicionário, pelo que a sua latência é constante em função de uma operação singular, e linear considerando um lote de operações.

Formas Normais Conjuntiva e Disjuntiva. Neste teste (Figura 4) comparamos o desempenho de quatro pesquisas diferentes, variando o formato da pesquisa e o seu tamanho: pesquisas na forma normal conjuntiva (FNC) – com uma e três conjunções (respectivamente quatro e oito termos) e pesquisas na forma normal disjuntiva (FND) – com uma e três disjunções. Analisando os resultados, concluimos que, por um lado, a forma da pesquisa não afecta o desempenho da mesma, já que o número de documentos a requisitar e processar da base de dados é igual; por outro lado, o tamanho da pesquisa influencia a latência, visto que a presença de mais termos implica reaver mais entradas da base de dados.

Selectividade da Pesquisa. Neste teste efectuámos várias pesquisas de um termo, cada uma retornando uma percentagem do repositório diferente, entre 0,2 e 30%.

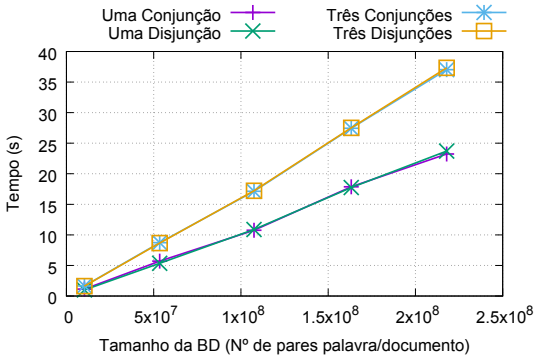


Figura 4: Comparativo entre pesquisas na FNC e na FND, com número variável de termos.

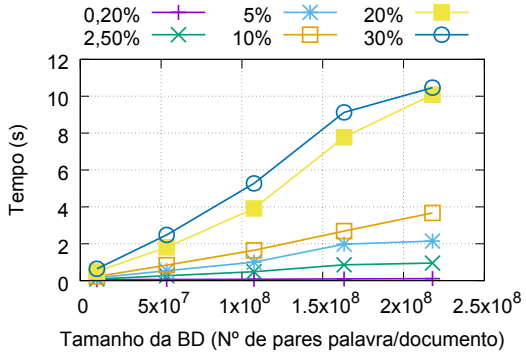


Figura 5: Desempenho relativo à selectividade dos termos.

Como demonstra a Figura 5, uma maior selectividade implica uma maior latência para a pesquisa. Tal é justificado pelo facto de que o protocolo implica reaver todos os documentos da base de dados que contêm essa palavra; uma palavra mais popular implicará um maior número de acessos à base de dados.

Comparação com o Estado da Arte. Na Tabela 1 comparamos a nossa solução com o esquema IEX-2LEV [15] através do código-fonte aberto dos autores. Dado que o esquema IEX-2LEV requer uma grande quantidade de RAM, executámos os testes para ambos os esquemas na máquina AMD Opteron 6272 com 64GB de RAM, com um modo simulado para o SGX. Os testes foram corridos com uma base de dados de até 56 mil pares, não nos tendo sido possível executar testes no IEX-2LEV com mais documentos.

Executámos as operação *Actualização* (chamada *Setup* no IEX-2LEV) e *Pesquisa* em ambos, tendo chegado à conclusão de que o desempenho do PBC é vastamente superior ao estado da arte. Tal pode ser explicado pela complexidade criptográfica do esquema IEX-2LEV, dado que o mesmo não considera uma base

Tamanho da BD	**Actualização**		**Pesquisa**	
(Nº pares termo/doc)	PBC	IEX-2LEV	PBC	IEX-2LEV
9 793	0,151	5 143	0,004	12
27 446	0,423	15 568	0,021	173
56 238	0,862	29 274	0,061	216

Tabela 1: Comparação de desempenho entre PBC e IEX-2LEV [15]. Tempos em segundos, pesquisa com oito termos.

de confiança do lado do servidor. Por conseguinte, e para manter garantias fortes de confidencialidade (que, ainda assim, são mais fracas que do que as oferecidas pelo PBC), cria estruturas de dados pouco eficientes, as quais crescem exponencialmente com o número de pares termo/documento na base de dados. No PBC, as estruturas de dados do repositório crescem linearmente.

6 Conclusão

Neste artigo propusemos um novo esquema de CSP híbrido, combinando primitivas criptográficas comuns com uma solução de hardware confiável, assim permitindo a delegação de computações para a Nuvem e melhorando o estado da arte em três frentes: segurança, desempenho e expressividade.

Como demonstrado pela nossa avaliação experimental, a solução atinge um desempenho superior ao estado da arte, fornecendo garantias de segurança fortes e suportando pesquisas booleanas de formato arbitrário.

Para trabalho futuro consideramos estender o esquema para suportar pesquisas com relevância, retornando um número fixo de documentos mais relevantes ao Cliente, ao invés de todos os correspondentes à pesquisa. Tal permitirá não só melhorar o desempenho do esquema para repositórios de tamanho elevado, como também ocultar o tamanho da resposta, que poderá indicar se uma pesquisa se refere a palavras mais ou menos populares, isto é, que aparecem em mais ou menos documentos. Também consideramos adicionar uma solução de armazenamento padrão na Nuvem, como Redis ou Cassandra, que permitirão conjugar armazenamento na memória e no disco, bem como a possibilidade de distribuir o mesmo entre várias Nuvens. Tornar o sistema robusto contra ataques de negação de serviço é igualmente um caminho de investigação possível, podendo ser consideradas soluções com vários enclaves em Nuvens diferentes, como forma de distribuir carga e aumentar a disponibilidade da nossa solução na presença de ataques do género.

Referências

1. Amir, W.: Hackers Leak 36 Million+ MongoDB Accounts. https://tinyurl.com/mongo-leak (jun 2016)

2. Anati, I., Gueron, S., Johnson, S., Scarlata, V.: Innovative Technology for CPU Based Attestation and Sealing. HASP '13 (2013)
3. ARM: ARM Security Technology: Building a Secure System using TrustZone Technology ARM (2009), https://tinyurl.com/armtrustzonereport
4. Barbosa, M., Portela, B., Scerri, G., Warinschi, B.: Foundations of hardware-based attested computation and application to SGX. In: EURO S&P'16. pp. 245–260 (2016)
5. Bost, R., Minaud, B., Ohrimenko, O.: Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives. In: CCS'17. ACM (2017)
6. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M.C., Steiner, M.: Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In: CRYPTO'13. pp. 353–373. Springer (2013)
7. Chirgwin, R.: Two Million Recordings of Families Imperiled by Cloud-Connected Toys' Crappy MongoDB. https://tinyurl.com/register-db-leak (feb 2017)
8. Cloud Security Alliance: The Treacherous 12 - Top Threats to Cloud Computing + Industry Insights (2017), https://tinyurl.com/treacherous12
9. Costan, V., Devadas, S.: Intel sgx explained. Tech. rep., Cryptology ePrint Archive, Report 2016/086, 2016. https://eprint.iacr.org/2016/086 (2016)
10. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: Proceedings of the 13th ACM CCS. pp. 79–88 (2006)
11. Duckett, C.: Australian Red Cross Apologises for Massive Data Leak. https://tinyurl.com/red-cross-data-leak (oct 2016)
12. Fuhry, B., Bahmani, R., Brasser, F., Hahn, F., Kerschbaum, F., Sadeghi, A.R.: Hardidx: practical and secure index with sgx. In: IFIP DBSec. pp. 386–408. Springer (2017)
13. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: ACNS. pp. 31–45 (2004)
14. Hoekstra, M., Lal, R., Pappachan, P., Phegade, V., Del Cuvillo, J.: Using Innovative Instructions to Create Trustworthy Software Solutions. In: HASP' 13. ACM, New York, NY, USA (2013)
15. Kamara, S., Moataz, T.: Boolean Searchable Symmetric Encryption with Worst-Case Sub-Linear Complexity. In: EUROCRYPT'17. IACR (2017)
16. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Proceedings of the 19th ACM CCS. pp. 965–976. ACM (2012)
17. McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C.V., Shafi, H., Shanbhogue, V., Savagaonkar, U.R.: Innovative Instructions and Software Model for Isolated Execution. ACM, New York, NY, USA (2013)
18. MongoDB: Encryption at Rest (2018), https://tinyurl.com/mongo-encryption
19. MySQL: MySQL Enterprise Transparent Data Encryption (2018), https://www.mysql.com/products/enterprise/tde.html
20. Ohrimenko, O., Schuster, F., Fournet, C., Mehta, A., Nowozin, S., Vaswani, K., Costa, M.: Oblivious Multi-Party Machine Learning on Trusted Processors. In: Proceedings of the 25th USENIX Security (2016)
21. Santos, N., Gummadi, K.P., Rodrigues, R.: Towards trusted cloud computing. In: HotCloud'09. USENIX Association (2009)
22. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the 21st S&P. pp. 44–55. IEEE (2000)
23. Stefanov, E., Papamanthou, C., Shi, E.: Practical Dynamic Searchable Encryption with Small Leakage. In: Proceedings of the 21th NDSS (2014)