DocNuvem: Edição Colaborativa de Documentos na Nuvem com Garantias de Privacidade

João Rodrigues, Bernardo Ferreira, João Leitão, and Henrique Domingos

NOVA-LINCS / DI-FCT-UNL
Faculdade de Ciências e Tecnologia, 2829-516 Caparica, Portugal
jm.rodrigues@campus.fct.unl.pt,bernardof@acm.org
{jc.leitao,hj}@fct.unl.pt

Resumo Este artigo propõe uma solução inovadora para edição colaborativa e eficiente de documentos na nuvem, utilizando técnicas e métodos criptográficos homomorficos. A solução permite preservar a integridade e a privacidade dos dados e operações, sendo estas aplicadas diretamente sobre os documentos cifrados mantidos na nuvem, durante o processo de edição colaborativa. A concretização da solução e a sua avaliação mostram que em cenários comuns de edição colaborativa na nuvem é possível ultrapassar diversas limitações identificadas em soluções anteriormente propostas. Ao mesmo tempo a avaliação dos mecanismos propostos mostra que se verifica a degradação na eficiência das operações de edição é aceitável, considerando as dimensões típicas de documentos e a granularidade das operações aplicadas sobre os dados cifrados na nuvem.

Palavras-chave: Outsource de Dados e Computação, Criptografia Aplicada, Processamento de Dados Cifrados, Edição Colaborativa

1 Introdução

As plataformas em nuvem são reconhecidas pelas vantagens que oferecem. Escalabilidade, elasticidade, ubiquidade e custos ajustáveis são apenas algumas das suas vantagens, tidas também como principais características motivadoras na adoção das mesmas. Plataformas baseadas na nuvem podem ser exploradas para beneficiar diversas aplicações e serviços distribuídos, incluindo serviços de armazenamento, partilha de dados, email, alojamento de páginas web ou ferramentas colaborativas para, por exemplo, edição de documentos. Este último é especialmente interessante, pois permite a interação e sincronização de múltiplos utilizadores, encorajando um processo de produção de conteúdos mais dinâmico e com maior produtividade conjunta.

No entanto, existem algumas barreiras à adoção de soluções na nuvem que continuam a impedir a sua ampla adoção. Falta de garantias de segurança auditáveis, problemas de disponibilidade e integridade ([1,2]),

quebras de privacidade resultante da atividade de administradores de sistemas maliciosos [3,4] e de entidades externas [5], e/ou bugs nas próprias plataformas distribuídas da nuvem [6], são algumas das razões mais apontadas por organizações e utilizadores como obstáculos à adoção da nuvem. Estas barreiras tornam-se ainda mais evidentes quando está em causa a exportação de dados sensíveis ou a execução de serviços críticos, como a edição de documentos de natureza privada e sensível.

De forma a endereçar as limitações de segurança da nuvem, diferentes soluções têm sido propostas para aplicações específicas. Por exemplo, soluções para armazenamento seguro de dados na nuvem tiram partido de criptografia simétrica e sínteses, de forma a garantir a sua privacidade e integridade [7]. Soluções de email podem usar os mesmos mecanismos aliados a certificados, de forma a garantir também autenticidade das mensagens [8]. Nestas aplicações, os dados são geralmente armazenados como *backups*, acedidos individualmente pelos utilizadores. No entanto, no caso de sistemas colaborativos, e em particular editores de documentos na nuvem, os dados são editados frequentemente por múltiplos utilizadores. Para estas aplicações, soluções existentes fazem uso criptografia simétrica para proteger o conteúdo das operações colaborativas e recorrem a protocolos centralizados para consistência, baseados na teoria da causalidade, para resolver conflitos de edição [9]. No entanto, a nuvem consegue observar todos os padrões de acesso e escrita aos diferentes documentos, informação essa que pode ser usada para alavancar ataques estatísticos e quebrar a privacidade dos documentos que se esperava estarem protegidos [10].

Neste artigo apresentamos uma solução segura para edição colaborativa de documentos que resolve as limitações de sistemas anteriores, preservando a privacidade dos documentos e ofuscando padrões de acesso dos utilizadores. A solução faz uso de criptografia homomórfica, impedindo que um atacante, a operar na infra-estrutura da nuvem, controlando os canais de comunicação com o utilizador, possa inferir através da observação dos padrões de acesso (análise estatística) ou simples observação de dados e operações, informações acerca do documento em edição. Mais especificamente a solução abordada apresenta como principais contribuições:

- Um sistema de edição colaborativa que preserva a funcionalidade base existente nos sistemas de edição colaborativa não seguros, isto é, dá suporte para operações de escrita e leitura sobre documentos partilhados por um conjunto de utilizadores sem incluir mecanismos de resolução de conflitos.

– Oferece garantias de privacidade e integridade efetiva dos documentos e das operações remotas efetuadas sobre os documentos mantidos numa nuvem computacional não confiável, fazendo para isso uso de mecanismos criptográficos homomórficos.
– Adicionalmente, o sistema ofusca padrões de acesso aos documentos, tirando partido de operações de escrita/leitura indistinguíveis entre si, impedindo assim ataques estatísticos [10] e efetivamente preservando a privacidade dos dados.
– Um protótipo que demonstra a usabilidade e performance do sistema para aplicações reais, tendo em conta os *overheads* associados ao uso de criptografia homomórfica, quando comparado com outros mecanismos e soluções existentes [11].

O resto do artigo está organizado da seguinte forma: Na Secção 2 apresentamos o estado da arte relacionado com este trabalho, com ênfase em que suportam atividade colaborativa; na Secção 3 apresentamos os modelos sobre os quais o nosso trabalho assenta, incluindo o modelo de sistema, modelo de adversário e modelo de dados; na Secção 4 apresentamos os componentes do sistema em maior detalhe, incluindo protocolos de suporte às principais operações; na Secção 5 descrevemos um protótipo do sistema e apresentamos uma avaliação em termos de segurança e performance; por fim, a Secção 6 concluí o artigo apresentando vetores de investigação a serem perseguidos como trabalho futuro.

2 Trabalho Relacionado

Colaboração e *Groupware* é uma área que tem tido bastante atenção por parte da comunidade cientifica nas ultimas décadas. Diferentes soluções têm sido propostas, divergindo em arquitetura (distribuída [12,13] ou centralizada [14,15,9]) e aproximação usada para resolução de conflitos (Transformação de Operações (OT) [16,17], Estruturas de Dados Comutativas (CRDTs) [13] ou Resolução de Conflitos (ACF) [18]). Referimos o leitor interessado para [19] o qual apresenta uma revisão bastante alargada do estado da arte nesta área.
Ao mesmo tempo, com a crescente popularidade de soluções "outsourced" nomeadamente soluções suportadas pelo paradigma de computação na nuvem, questões de segurança, privacidade e fiabilidade começaram a fazer parte da agenda da comunidade Colaborativa [9,11,20]. No entanto, os trabalhos existentes têm-se focado maioritariamente em questões de tolerância a falhas bizantinas e privacidade dos documentos, deixando a ofuscação de operações e suas implicações de parte. Um excelente exemplo do estado da arte é o sistema SPORC [9], uma *framework* para suporte

de aplicações colaborativas sobre servidores de terceiros não confiáveis. O SPORC suporta tolerância a falhas bizantinas e de paragem (*crash*), mantém a privacidade das operações através de criptografia simétrica e suporta diferentes tipos de aplicações colaborativas através de um protocolo de OT centralizado. No entanto, o servidor apenas é utilizado para ordenar operações, sendo os clientes a transforma-las concorrentemente. Por outro lado, de forma a detetar comportamento incorreto do servidor, todas as operações têm de ser assinadas recorrendo a criptografia de chave-publica, aumentando o *overhead* das mesmas. Adicionalmente o servidor, apesar de não ver o conteúdo das operações cifradas, consegue ver todo o padrão de acessos o que cria uma oportunidade de ataques estatísticos.

Outros trabalhos similares, apenas focados na privacidade dos documentos [11,20], intercetam todas as operações dos clientes recorrendo a uma arquitetura baseada em *middleware* e cifram o seu conteúdo antes de as enviar para a nuvem. Adicionalmente, de forma a otimizar *overheads* das operações, recorrem a *buffering* das operações e usam criptografia incremental para enviar *batches* de operações [11], ou alternativamente, dividem os documentos em blocos e usam estruturas de dados para otimizar a edição destes [20]. Porém, estes trabalhos tendem a só funcionar sobre um serviço na nuvem especifico (ex: [11] só funciona com o Google Docs).

3 Modelos

3.1 Modelo de sistema

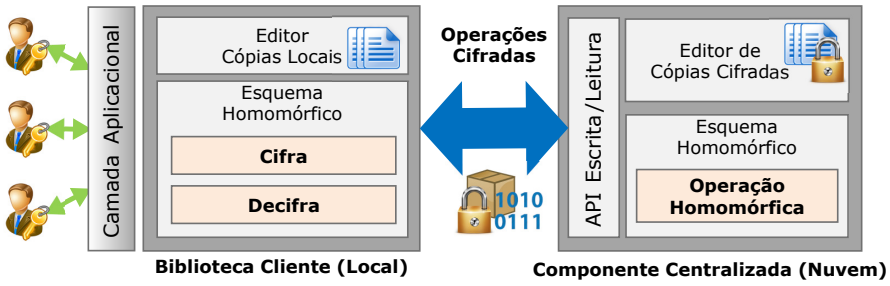

Figura 1. Modelo de sistema

O modelo do nosso sistema, ilustrado na Figura 1, inclui duas entidades principais: Uma *Componente Centralizada*, que é instanciada numa nuvem não confiável (detalhes do modelo de adversário na sec. 3.2); e várias aplicações cliente, cada uma composta por dois sub-componentes: *Camada Aplicacional* e *Biblioteca Cliente*. Na *Componente Centralizada*

são depositados um ou mais documentos sob forma cifrada. Os documentos são guardados na nuvem sob a forma de blocos de dados de tamanho fixo parametrizável, identificados por um *id* gerado aleatoriamente. Os documentos são operados de forma colaborativa por um conjunto de utilizadores (as Aplicações Cliente) que partilham um par de chaves assimétricas por cada documento. O processo de partilha de chaves não é descrito no presente artigo, no entanto pressupõe-se a existência de um processo, *online* ou *offline*, para a geração e partilha de chaves, controlado pelos utilizadores que criam os documentos.

Os utilizadores mantêm localmente uma cópia dos documentos a que têm acesso e sobre os quais estão ativamente a trabalhar. Desta forma é possível mascarar falhas temporárias de disponibilidade da nuvem e operar sobre os documentos de forma desligada. Os utilizadores contactam o serviço na nuvem através de uma aplicação cliente (*Camada Aplicacional*) que pode executar de forma isolada ou no navegador do utilizador. A aplicação recorre então a uma biblioteca disponibilizada como parte da nossa solução (*Biblioteca Cliente*), que é responsável por cifrar os conteúdos das operações dos utilizadores antes de serem enviadas para a nuvem (componente centralizada) e por decifrar as resposta recebidas. Esta biblioteca faz uso de primitivas criptográficas homomórficas (mais detalhes sobre estes componentes na sec. 4).

Em termos de operações, o sistema oferece três funcionalidades essenciais aos utilizadores (que depois são convertidas internamente antes de serem enviadas para a nuvem): i) edição (inserção ou remoção de conteúdos); ii) leitura (obtenção da totalidade ou frações do documento guardados na nuvem); e iii) criação de documento (criação de um espaço de trabalho colaborativo com um par de chaves assimétricas associadas). Uma edição é vista como uma alteração de um documento numa determinada posição (definida por um deslocamento relativamente ao inicio do mesmo). Uma edição pode ser composta pela remoção, adição, ou alteração de caracteres contíguos[1]. Uma leitura é vista como a recuperação de uma sequência de caracteres de tamanho variável a partir de um determinado deslocamento em relação ao inicio do documento. Finalmente, existe uma operação de criação de documento. Nesta operação, o novo documento é inicialmente criado de forma local sob a forma de um conjunto de blocos vazios de tamanho fixo. Estes blocos são então cifrados e armazenados na nuvem (note-se que o cliente mantém uma cópia local destes blocos).

[1] Note-se que, apesar de o ênfase deste trabalho incidir sobre a edição de texto, a abordagem apresentada é independente do formato do documento.

Como iremos explicar mais à frente, todas estas operações do utilizador são convertidas pela biblioteca cliente por forma a utilizarem uma interface única de acesso à componente do sistema a executar na nuvem (componente centralizada). Isto torna as diferentes operações indistinguíveis por parte da componente centralizada ou de qualquer entidade que observe a comunicação entre utilizadores e essa componente. Assume-se que os clientes se autenticam perante o sistema usando um mecanismo fora do escopo deste artigo, sendo que no entanto esse mecanismo não depende de qualquer par de chaves assimétricas utilizadas para proteger os conteúdos dos documentos. Adicionalmente, a biblioteca cliente é responsável por se registar na componente da aplicação a executar na nuvem, o que permite à ultima o envio de notificação sobre blocos de documentos sobre os quais o utilizador se encontra a trabalhar.

3.2 Modelo de adversário

O desenho do nosso sistema DocNuvem pressupõe o seguinte modelo de adversário:

Operador da Nuvem: Assumimos que os operadores ou administradores da infraestrutura de nuvem podem tentar aceder aos conteúdos dos documentos nela guardados (por curiosidade ou para aceder a informação privilegiada neles contidos). Assumimos ainda que por forma a minimizar os seus custos de operação nomeada os custos de armazenamento, o operador da nuvem pode tentar violar o seu contrato com os utilizadores, não garantindo a disponibilidade dos dados guardados na sua infraestrutura, através da remoção de blocos de dados do utilizador. Finalmente os operadores da nuvem podem ainda tentar efetuar ataques estatísticos que lhes permitam inferir informação sensível dos documentos.

Atacante Externo: Assumimos ainda a existência de atacantes externos, que não dispõem de credenciais de acesso ao serviço, e que desconhecem as chaves criptográficas utilizadas para proteger o conteúdo dos documentos. Estes atacantes podem no entanto observar o tráfego entre os clientes e utilizadores por tempo indeterminado, e efetuar ataques estatísticos alavancando toda a informação recolhida. Assumimos que o processo de autenticação utilizado pelos utilizadores perante a componente na nuvem, contemplam mecanismos de segurança para evitar ataques de homem-no-meio (*man-in-the-middle*) e de repetição (*replay attacks*).

3.3 Modelo de dados

O modelo de dados da solução proposta (Fig. 2), tal como discutido brevemente na secção 3.1, é constituído por um ou mais documentos que se

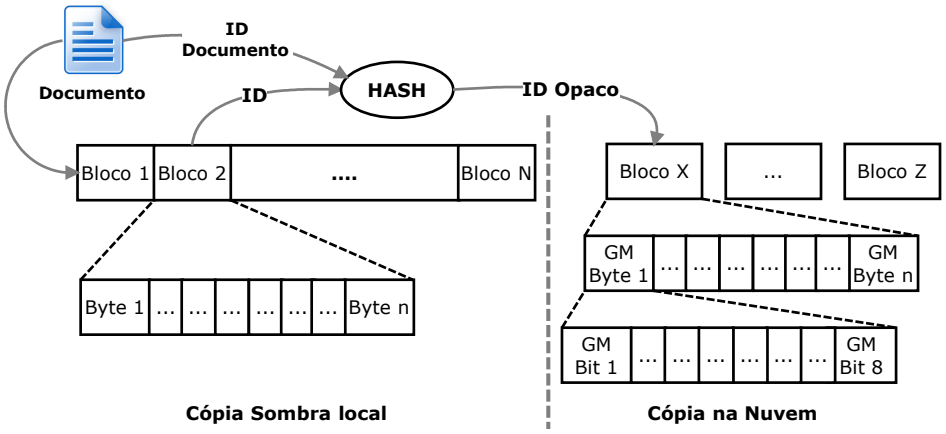

Figura 2. Modelo de dados

encontram replicados sob a forma de blocos, de tamanho fixo parametrizável, entre a componente cliente e a componente servidor a executar na nuvem. No lado do cliente é mantida uma cópia em claro dos blocos de cada documento, ordenados sequencialmente segundo um identificador único. Qualquer alteração nos blocos desta cópia é propagada para a cópia na nuvem e daí para cada um dos outros clientes, por meio de chamadas assíncronas. No momento da inicialização do sistema o cliente carrega para memória o documento da *cache* local, sendo as divergências da cópia remota carregadas por meio de um processo de sincronização. Caso não exista uma cópia em *cache* local todo o documento é descarregado e decifrado. No lado da nuvem, cada bloco é cifrado com um esquema homomórfico (detalhes na secção 4.1) e identificado através de um ID opaco. Este ID é gerado por uma função de síntese, que recebe como valor de entrada $ID\_documento||indice$, onde $ID\_documento$ é uma string que identifica univocamente um documento no sistema e o *indice* corresponde ao numero de sequência do bloco dentro do documento. Neste cenário, o ID do documento é apenas conhecido entre os utilizadores e nunca exposto à componente centralizada. Por fim, no final de cada bloco são mantidos um conjunto de bits de paridade (com dimensão parametrizável). Estes bits são atualizados pelos clientes nas escritas tomando em conta o valor anterior do bloco. Estes bits de paridade permitem a qualquer cliente efetuar~~em~~, após a leitura de um bloco, uma verificação local da integridade do bloco recuperado. No caso desta verificação falhar, o cliente pode tentar ler uma versão antiga – mas integra – desse bloco.

4 Componentes e Implementação

Tal como descrito anteriormente o sistema proposto é composto por 3 componentes, duas das quais operam do lado do cliente, enquanto a ultima é executada na nuvem. O objetivo do DocNuvem é garantir a confidencialidade tanto dos operações efetuadas pelos utilizadores como dos conteúdos manipulados por essas operações. Para tal é nos necessário recorrer a um conjunto de protocolos que alavancam criptografia homomórfica. Nesta secção descrevemos estes protocolos em maior detalhe.

4.1 Esquema Criptográfico.

Como base para o suporte de escritas e leituras remotas, garantido assim a privacidade tanto de conteúdos como operações, utilizamos o esquema homomórfico Goldwasser-Micali [21]. Este é um esquema criptográfico assimétrico e probabilístico, sendo considerado seguro no modelo IND-CPA. Este algoritmo é baseado no problema dos resíduos quadráticos de N, que por sua sua vez depende da fatorizarão de N (resultado da multiplicação de dois primos). Em particular, este esquema dispõe da seguinte propriedade que alavancamos no desenho do nosso sistema:

$$\xi(b_1) \times \xi(b_2) = \xi(b_1 \oplus b_2) \tag{1}$$

Esta expressão denota que a multiplicação de dois bits cifrados de forma probabilística é equivalente à cifra do resultado da operação de OU exclusivo ($\oplus$) desses mesmos dois bits em claro, oferecendo por isso a oportunidade de efetuar $\oplus$ sobre conteúdos cifrados. Na prática, este facto traduz-se na possibilidade de efetuar OUs exclusivos sobre dados cifrados, o que é especialmente interessante neste contexto visto que a operação de $\oplus$ possui um conjunto de propriedades bastante interessantes e sumariadas na Tabela 1.

OU exclusivo
1. $A \oplus 0 = A$
2. $A \oplus 1 = \overline{A}$
3. $A \oplus A = 0$
4. $A \oplus B = B \oplus A$
5. $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

Tabela 1. Propriedades da operação de OU exclusivo

4.2 Interface da Componente Centralizada

Por forma a garantir a privacidade das operações efetuadas pelos utilizadores, a interface exposta pelo componente centralizado do DocNuvem

que executa na nuvem tem de ser tão simples quanto possível, e idealmente suportar uma única operação. No nosso caso decidimos relaxar este ponto, sendo que a interface desta componente suporta apenas duas operações:

$$doc_{id} \leftarrow criarDoc\{(\xi(bloco_0), bloco_0.id), ..., (\xi(bloco_n), bloco_n.id)\} \quad (2)$$

$$\xi(bloco_{result}) \leftarrow mult(\xi(bloco_{local}), bloco_{remoto}.id, bloco_{result}.id) \quad (3)$$

Desta forma, quando um utilizador pretende criar um novo documento, ele gera localmente um par de chaves assimétricas, usando a chave privada para cifrar um conjunto de blocos iniciais com todos os bits com valor 0 ($bloco_0, ..., bloco_n$). Cada bloco é identificado por um ID opaco, como definido previamente no modelo de dados (sec. 3.3).

A segunda operação é utilizada para efetuar escritas e leituras por parte do utilizador, após uma transformação da operação por parte da biblioteca cliente. A semântica interna aplicada pela nuvem é relativamente simples: dado um bloco cifrado de entrada ($bloco_{local}$), a nuvem efetua uma multiplicação com o bloco ($bloco_{remoto}$) guardado na nuvem (relembramos o leitor que uma multiplicação é equivalente a um $\oplus$ sobre o texto em claro) sendo o resultado guardado num bloco ($bloco_{result}$) e simultaneamente retornado ao cliente.

4.3 Operações do Cliente

Operação de Leitura Considerando o esquema criptográfico utilizado no desenho do nosso sistema e a interface disponibilizada pela componente centralizada na nuvem, para efetuar uma leitura a biblioteca cliente primeiro determina o identificador do bloco que pretende ler, e de seguida cifra um bloco ($bloco_{nulo}$) contendo apenas bits com o valor 0 utilizando a chave privada (por questões de eficiência um conjunto destes blocos pode ser mantido em *cache* pela biblioteca cliente). De seguida o utilizador envia um pedido para a componente centralizada do tipo *multiplicar* (equação 3) utilizando o $bloco_{nulo}$ gerado como primeiro argumento, o identificador do bloco a ler como segundo argumento, e o terceiro argumento com um índice aleatoriamente escolhido pela biblioteca cliente, de forma a ofuscar a natureza e valor real da operação. Ao receber a resposta da componente centralizada, a biblioteca cliente decifra o bloco usando a chave pública do documento e reporta o bloco à camada aplicacional após efetuar uma verificação dos seus bits de paridade.

A intuição principal desta operação é que no esquema criptográfico utilizado, e tal como explicado anteriormente, a multiplicação de um bloco arbitrário cifrado com um bloco cifrado contendo apenas bits de valor

plaintext 0 gera um novo bloco cifrado diferente de ambos, mas cujo conteúdo decifrado será igual ao do primeiro bloco.

Operação de Escrita De forma a simplificar a exposição deste mecanismo, assumimos que cada operação de escrita é reduzida à alteração do valor de um carácter num único bloco. Evidentemente várias operações de escrita de um utilizador podem ser agregadas numa única operação desde que sejam sobre o mesmo bloco de dados. Quando um utilizador modifica um carácter num bloco, essa operação é enviada para a biblioteca cliente, que irá gerar um bloco contendo todos os bits a 0, exceto os bits que se pretendem escrever. Esse bloco é então cifrado, criando assim uma mascara de bits aleatórios com o tamanho de um bloco.

Esta mascara quando aplicada sobre blocos (vazios ou previamente editados) resulta numa escrita homomórfica, ou seja, uma escrita sobre dados cifrados. Tal como podemos verificar na Figura 3 uma operação de OU exclusivo sobre dados em claro é diretamente mapeada para essa mesma operação sobre dados cifrados utilizando a operação de multiplicação tal como apresentado na Expressão 1.

Através da operação de escrita é também possível suprimir dados (caracteres), tal como definido na propriedade 3 da tabela 1. Para este fim bastará efetuar uma operação de escrita com os mesmos dados que se pretendem suprimir.

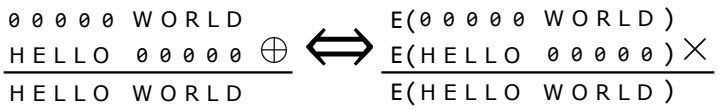

Figura 3. Exemplo de operação de escrita

Como trabalho futuro, as propriedades expostas na tabela 1 podem vir a ser exploradas para suportar funcionalidades adicionais. Nomeadamente poderão ser implementadas operações de duplicação/replicação de blocos utilizando a propriedade 1 ou até de ofuscação temporal dos padrões de acesso, implementando para isso operações nulas aplicadas de forma aleatória ao longo do tempo sobre os vários blocos que compõem o documentos.

5 Resultados e avaliação

Como prova de conceito, e de forma a apurar a usabilidade da solução proposta, desenvolvemos um protótipo em Java executando testes com foco nos mecanismos criptográficos utilizados (Micro-Benchmark) e na solução como um todo (Macro-Benchmark). Os testes sobre os mecanismos foram realizados de forma comparativa utilizando como dados de

entrada sequências continuas de caracteres aleatórios (1 byte por caráter codificado em ASCII). A solução como um todo foi testada através de um cliente evocando operações de escrita e leitura de acordo com um traço de execução de um documento de texto seguindo uma ordem de escrita aleatória, tentando assim simular um ambiente real de edição. Para ambos os testes foi utilizada uma máquina local Core i7 sendo que nos Macro-Benchmark esta maquina foi utilizada como cliente, e uma micro instância da Amazon EC2, *Data-Center* da Califórnia foi utilizada como Componente Centralizada.

5.1 Funcionalidade

Ao contrário dos serviços de edição colaborativa online, a nossa solução não permite que o provedor da nuvem tenha acesso ao conteúdo dos documentos. Por essa razão, funcionalidades como pesquisa, tradução e correção ortográfica remota sobre o conteúdo não poderão ser suportadas pelo provedor na nuvem. Não obstante tais funcionalidades poderão ser facilmente suportadas pela aplicação cliente. O protótipo desenvolvido também não suporta funcionalidades como formatação de documentos ou inclusão de objetos não textuais como é o caso de imagens. Além disso o protótipo desenvolvido não possui um mecanismo para resolução de conflitos em edições concorrentes. No entanto é possível que dois utilizadores editem de forma colaborativa e em simultâneo o mesmo documento, desde que o façam em deslocamentos distintos. Como trabalho futuro pretendemos conceber um protocolo fiável para gestão destes mesmos conflitos.

5.2 Otimizações

Resultados preliminares demonstraram que o uso de criptografia da chave pública, mais especificamente o uso do esquema homomórfico Goldwasser-Micali, tem uma baixa performance em contextos de cifra e decifra de conteúdos de volumes consideráveis. Para endereçar este problema, foram incluídas algumas otimizações no protótipo inicial de forma a garantir a usabilidade do sistema. Adicionalmente, podemos considerar que os documentos de texto são de tamanho limitado, como acontece em editores populares como o Google Docs, que apenas suporta documentos com no máximo 1.024.000 caracteres [2]. Não obstante, não existem limitações ou *overheads* de performance relacionados com o tamanho dos documentos editados, sendo que o *overhead* recai diretamente sobre o espaço necessário para armazenar esse mesmo documento na forma cifrada.

[2] *Google Docs, Sheets, and Slides size limits*, http://goo.gl/NnHjQt

Listas pré-computadas de valores cifrados Por forma a mitigar os tempos necessários no processo de cifra, durante o processo de instanciação do sistema o cliente gera duas listas contendo valores de bits cifrados. Como só existem dois valores possíveis para um bit (0 ou 1), são geradas duas listas deste tipo. Estas listas possibilitam não só acelerar o processo de cifra, pela substituição de forma aleatória de bits em claro por valores pré-cifrados, como possibilitam que seja endereçado o problema da expansão por meio de métodos de indexação simples.

Mapeamento para valores em claro Outra das otimizações utilizada consiste em manter em memória um mapa que relaciona os valores dos bits cifrados para os valores em claro correspondentes. Desta forma, é possível acelerar o processo de decifra de dados através de simples consultas nesta mesma estrutura. Durante o processo de cifra esta estrutura é preenchida, sendo posteriormente utilizada no processo de decifra. Não obstante, esta otimização apresenta apenas vantagens em dados armazenados de forma estacionária, ou seja, dados não re-editados (por ex., um documento arquivado) aos quais não tenham sido aplicadas operações homomórfica (de natureza probabilística) preservando desta forma o determinismo dos valores cifrados. Por outro lado, podemos facilmente antever cenários nos quais grande parte dos blocos dos documentos não são editados e por essa razão esta otimização poderá ser utilizada como forma de acelerar o processo de decifra.

5.3 Micro-Benchmark

Para analisar o *overhead* introduzido pelo uso de criptografia homomórfica quando comparado com o uso de criptografia simétrica tradicional, executámos um *benchmark*, averiguando qual o diferencial de latência no processo de cifra, aplicação da operação homomórfica (OU Exclusivo) e decifra por caráter, com e sem as otimizações introduzidas. Para estes testes, implementámos e utilizámos o esquema Goldwasser-Micali (com chaves de 1024 bits) comparando com os algoritmos criptográficos AES, com chave de 256 bits e 3DES, com chave de 168bits, disponiveis na biblioteca criptográfica *Bouncy Castle*[3].

Na Tabela 2 podemos observar os valores de performance temporal de cifra, decifra e operação homomórfica por carácter (1 byte) para os mecanismos, nas vertentes simples e otimizada. Através duma rápida inspeção verificamos que os algoritmos de cifra simétrica são, em várias ordens de grandeza, mais eficientes que os algoritmos de cifra assimétrica puros, tal como utilizados na versão não otimizada. Com as otimizações

[3] Legion of the Bouncy Castle, https://www.bouncycastle.org

	Operação	Média (por caráter)
Não-Otimizado	Cifra	0.13ms
	Decifra	13.16ms
	Operação Homomórfica	0.0009ms
Otimizado	Cifra	0.0001ms
	Decifra	0.0001ms
	Operação Homomórfica	0.0001ms
AES-256	Cifra	0.000000329ms
	Decifra	0.000000065ms
3DES	Cifra	0.00000023ms
	Decifra	0.000000092ms

Tabela 2. Micro-Benchmark para o mecanismo homomórfico utilizado comparativamente com algoritmos de cifra simétrica standard

descritas na secção anterior, consegue-se reduzir essa diferença substancialmente. Não obstante, e tendo em conta o contexto de edição de documentos que se pretende endereçar, os valores de performance obtidos são aceitáveis do ponto de vista da usabilidade. É importante relembrar que a performance do processo de decifra na versão otimizada depende do conhecimento prévio dos valores em claro dos bits cifrados. Na prática tal limitação torna-se factual na edição remota de documentos, na qual a aplicação de uma operação homomórfica resulta em novos valores (probabilísticos/não determinísticos), desconhecidos da biblioteca cliente, a qual terá de aplicar a operação criptográfica de decifra.

5.4 Macro-Benchmark

De forma a avaliar a performance do sistema apresentado, definimos três cenários distintos envolvendo três versões do sistema desenvolvido:

1. **Cenário em claro.** A partir do sistema DocNuvem, tal como apresentada no presente artigo, foi criada uma versão na qual todos os dados são armazenados e recuperados em claro sendo que a operação de escrita sobre os documentos é aplicada pelo servidor que gere os dados em claro. A operação de escrita, à semelhança da solução DocNuvem, é baseada na operação de OU exclusivo sobre os dados depositados.
2. **Cenário com criptografia simétrica.** Partindo da solução base foram feitas algumas alterações de modo a que o sistema suporta-se o armazenamento de dados cifrados de forma simétrica (utilizando AES com chave de 256 bits), o que obriga ao armazenamento dos dados no lado do servidor de forma estacionária. De forma a dar suporte a operações de escrita sobre os documentos, estes são descarregados do servidor, decifrados, operados, cifrados novamente e só então enviados de volta para os servidor.

3. **Cenário com criptografia homomórfica.** Neste cenário foi aplicado o sistema tal como descrito nas secções anteriores. Ou seja, neste cenário é utilizada uma esquema com propriedades homomórficas, mais especificamente o esquema Goldwasser-Micali com uma chave de 1024 bits, o qual permite que os dados armazenados de forma cifrada no servidor sejam escritos homomorficamente, e por isso de forma privada.

Estes cenários foram testados utilizando um traço de execução com cerca de 200 operações de escrita e 200 de leitura (aproximadamente 50 carateres por operação) de um documento com cerca de 10.000 carateres no total. Nestes cenários, o tamanho dos blocos operados é de 512 bytes [4].

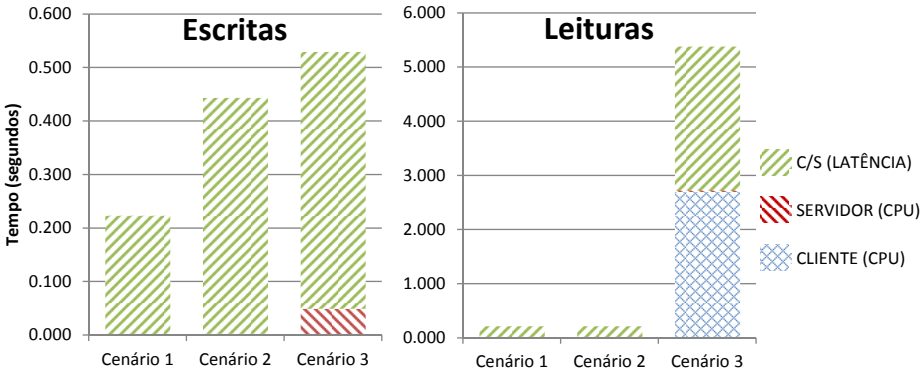

Figura 4. Macro-Benchmark para o traço de execução de um documento com escritas e leituras

A figura 4 representa os valores médios para as operações de escrita/leitura definidas no traço de execução. Tal como seria de esperar, o cenário base 1 apresenta, de forma geral, performances em tudo superiores aos restantes cenários. No entanto podemos verificar que a latência, à semelhança dos restantes cenários, é predominante relativamente aos tempos de processamento do cliente e servidor para operações de escrita e leitura.

No cenário 2 o tempo de processamento agrava-se de forma pouco significativa, tendo em vista o cenário analisado. Apesar de visualmente impercetível, neste cenário a latência numa operação de escrita é sensivelmente o dobro da mesma latência numa operação de leitura. Esta relação justifica-se pelo facto de uma escrita envolver sempre uma leitura de dados de forma a que estes possam ser escritos localmente, isto é, resultando em duas chamadas ao servidor (uma leitura e uma escrita).

[4] Esta dimensão foi selecionada atendendo ao compromisso entre o número de operações e o tamanho das mesmas

No cenário 3, correspondente à solução DocNuvem, verificamos que as latências de comunicação no processo de escrita são semelhantes ás do cenário 2. No que refere aos tempos de processamento do servidor, e tal como esperado devido ao uso de operações homomórficas, encontramos uma ligeira agravante. No processo de leitura, envolvendo decifras não otimizadas, verificamos tempos de processamento no cliente consideráveis. As latências de comunicação são justificadas pela expansão espacial da cifra, levando a que cada bloco de 512 bytes expanda para um bloco de 1 Megabyte, enviado no corpo das chamadas feitas ao servidor.

6 Conclusões

Neste artigo apresentámos uma solução de edição colaborativa de documentos na nuvem, baseada em criptografia homomórfica, que garante a confidencialidade e integridade dos dados e das operações. Quando comparada com o estado da arte [9,11,20], verificamos que a nossa solução é a primeira a aliar a cifra dos dados à ofuscação de padrões de acesso aos mesmos, protegendo assim de forma efetiva a confidencialidade dos documentos operados numa base computacional não confiável, face a ataques estatísticos. A avaliação da solução proposta revela que há um preço a pagar pela ofuscação de padrões de acesso aos documentos. Porém, acreditamos que com a nossa solução é ainda possível criar um sistema de edição colaborativa sem comprometer a sua usabilidade pratica. Por outro lado, existem otimizações adicionais que poderão ser aplicadas de forma a reduzir as latências verificadas, como por exemplo, a utilização de técnicas de compressão.
Como trabalho futuro, pretendemos desenvolver mecanismos para enriquecer a privacidade oferecida aos utilizadores, impedindo a um atacante inferir que um utilizador se encontra a aceder ao serviço. Para esse fim, pretendemos recorrer ao uso de ruído, ou seja clientes fictícios/inativos a gerarem operações vazias sobre o sistema. Pretendemos também endereçar limitações existes no prototipo atual nomeadamente em termos de suporte a acessos concorrentes e a integração do prototipo existente num serviço disponível na Internet, através, por exemplo, do uso de um *browser plugin*.

Referências

1. Gaudin, S.: Google outage takes down Gmail, Docs, Calendar. https://tinyurl.com/k8rj5v4
2. Reed, B.: Major outage takes down multiple Google services. http://bgr.com/2014/01/24/google-gmail-outage/
3. Chen, A.: GCreep: Google Engineer Stalked Teens, Spied on Chats. http://gawker.com/5637234 (2010)
4. Rushe, D.: Google: don't expect privacy when sending to Gmail. http://tinyurl.com/kjga34x (2013)
5. Greenwald, G., MacAskill, E.: NSA Prism program taps in to user data of Apple, Google and others. http://tinyurl.com/oea3g8t (2013)
6. Kincaid, J.: Google Privacy Blunder Shares Your Docs Without Permission. https://tinyurl.com/ybdyzcl
7. Kamara, S., Lauter, K.: Cryptographic Cloud Storage. In: Work. Real-Life Cryptogr. Protoc. Stand. (2010) 136–149
8. Rodrigues, J.a., Ferreira, B., Domingos, H.: TMS: A Trusted Mail Repository Service using Public Storage Clouds. In: MW4NG'13, ACM (2013)
9. Feldman, A.J., Zeller, W.P., Freedman, M.J., Felten, E.W.: SPORC: Group Collaboration using Untrusted Cloud Resources. In: OSDI. Volume 10. (2010) 337–350
10. Islam, M.S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In: NDSS. (2012)
11. Huang, Y., Evans, D.: Private editing using untrusted cloud services. In: Second Int. Work. Secur. Priv. Cloud Comput., IEEE (2011) 263–272
12. Sun, C.: Undo as concurrent inverse in group editors. ACM Trans. Comput. Interact. **9**(4) (2002) 309–361
13. Preguica, N., Marques, J.M., Shapiro, M., Letia, M.: A Commutative Replicated Data Type for Cooperative Editing. ICDCS (June 2009) 395–403
14. Nichols, D.A., et al.: High-latency, low-bandwidth windowing in the Jupiter collaboration system. In: Proc. 8th Annu. ACM Symp. User interface Softw. Technol., ACM (1995) 111–120
15. Wang, D., Mah, A., Lassen, S.: Google wave operational transformation. Whitepaper, Google Inc (2010)
16. Sun, C., Ellis, C.: Operational transformation in real-time group editors: issues, algorithms, and achievements. In: CCSCW, ACM (1998) 59–68
17. Sun, D., Sun, C.: Context-Based Operational Transformation in Distributed Collaborative Editing Systems. IEEE Trans. Parallel Distrib. Syst. **20**(10) (October 2009) 1454–1470
18. Shapiro, M., Bhargavan, K.: The Actions-Constraints approach to replication: Definitions and proofs. Tech. peport MSR-TR-2004-14. Microsoft Res. (2004)
19. Ignat, C.L., et al.: A Comparison of Optimistic Approaches to Collaborative Editing of Wiki Pages. In: Int. Conf. Collab. Comput. Networking, Appl. Work. (2007)
20. Yeh, S.C., Su, M.Y., Chen, H.H., Lin, C.Y.: An efficient and secure approach for a cloud collaborative editing. J. Netw. Comput. Appl. **36**(6) (November 2013) 1632–1641
21. Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Comput. Syst. Sci. **28** (1984) 270–299