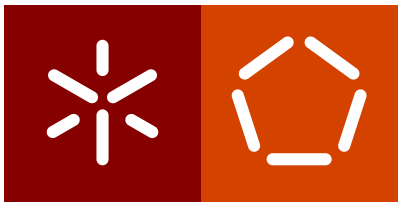
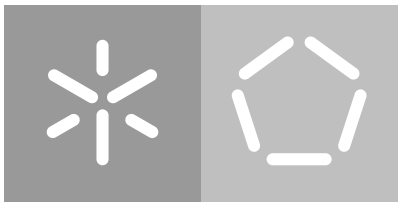


Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Carlos Carvalho Ferreira

Computação Segura Sobre Sistemas de Dados em Ambientes *Cloud*

Abril 2017



Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Carlos Carvalho Ferreira

Computação Segura Sobre Sistemas de Dados em Ambientes *Cloud*

Dissertação de Mestrado
Mestrado Engenharia Informática

Trabalho realizado sob orientação de
José Bacelar Almeida
Manuel Bernardo Barbosa

Abril 2017

AGRADECIMENTOS

Aos professores José Bacelar Almeida e Manuel Bernardo Barbosa pela orientação da dissertação, por toda a disponibilidade e por toda a ajuda dada.

Aos alunos de doutoramento Tiago Oliveira e Bernardo Portela pelos conselhos dados, tanto no trabalho prático como na escrita da dissertação explicando alguns conceitos mais complexos e sugerindo um rumo para escrita.

Aos meus pais por me suportarem e me terem dado a oportunidade de concluir este grau académico. Também tenho de lhes agradecer por terem de lidar comigo nos dias menos bons e nesta parte entra também o meu irmão.

Aos meus amigos que sempre me incentivaram a terminar a dissertação das mais diversas formas.

Por último, agradeço a todos aqueles que contribuíram para esta dissertação e que não estão mencionados.

ABSTRACT

The development of cloud services produced a massive migration of data to such services. The fact that the cloud provides a mean for users to access their data in every device (personal computer, mobile phone, tablet, etc.) just by having an Internet connection made it widely used. Using a cloud service brings several advantages to users. For example, the ease with which data is accessed and shared among different users, combined with the malleability and scalability provided by cloud storage, made cloud a target for enterprises to maintain their projects remotely and thus save money storing them in local servers.

Nevertheless, having data stored and managed remotely rises several security issues. In order to be trustable, a cloud provider must ensure that data is properly secure in terms of privacy, confidentiality and integrity. Recent attacks on largely used cloud providers (Greene, 2015) have made cloud services questionable in terms of information security, reducing the confidence placed on them.

This master thesis addresses one security challenge that cloud is facing, namely the computation over data. Computation should be made over encrypted data so that the cloud provider does not learn anything about the original data nor the result of the computation.

RESUMO

O desenvolvimento dos serviços *cloud* gerou uma migração massiva de dados para os mesmos. O facto da *cloud* fornecer um serviço acessível a partir de qualquer dispositivo (computador pessoal, telemóvel, *tablet*, etc.) através do acesso à Internet fez com que a sua utilização aumentasse. Estes serviços trazem bastantes vantagens para os utilizadores. A facilidade em aceder e partilhar os dados pelos diferentes utilizadores combinada com a maleabilidade e escalabilidade oferecidas pela *cloud*, faz com que este serviço se torne numa plataforma bastante pretendida pelas empresas para armazenar os seus projetos remotamente e, desta forma reduzir custos associados aos servidores locais.

No entanto, armazenar dados remotamente provoca o aparecimento de questões acerca de segurança e privacidade. De forma a garantir confiabilidade no serviço, o fornecedor *cloud* tem de garantir que os dados estejam seguros em termos de privacidade, confidencialidade e integridade. Ataques recentes a fornecedores de serviços *cloud* (Greene, 2015) levantaram questões em termos de segurança da informação, provocando um decréscimo na confiança depositada nestes serviços.

Esta dissertação aborda um desafio de segurança que a *cloud* está a enfrentar, a computação sobre dados. Esta computação tem de ser feita sobre dados cifrados para que o fornecedor de *cloud* não tenha acesso ao valor real da informação nem ao resultado da computação.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

ACRÓNIMOS

ACID	Atomicidade, Consistência, Isolamento e Durabilidade. 10
AES	Advanced Encryption Standard. 14, 44, 47, 48
CBC	Cipher Block Chaining. 44
CCA	Chosen-Ciphertext Attack. 20, 21
CMAC	Cipher-based Message Authentication Code. 30
CPA	Chosen-Plaintext Attack. 19, 20, 41
CRM	Customer Relationship Management. 26
DES	Data Encryption Standard. 14
HDFS	Hadoop Distributed Filesystem. 11
HGD	HyperGeometric Distribuition. 30
IND-CCA	INDistinguishability under Chosen Ciphertext Attack. 14, 17, 19, 24
IND-CCA2	INDistinguishability adaptive Chosen Ciphertext Attack. 19, 24, 25
IND-CCA3	(authenticated) INDistinguishability adaptive Chosen Ciphertext Attack. 24
IND-CPA	INDistinguishability under Chosen-Plaintext Attack. 14, 16, 17, 19, 22, 23, 33
IND-DCPA	INDistinguishability under Distinct Chosen-Plaintext Attack. 22, 33
IND-OCPA	INDistinguishability under Ordered Chosen-Plaintext Attack. 33, 35, 40, 41
INT-CTXT	Integrity of Ciphertexts. 16, 17
INT-PTXT	Integrity of Plaintexts. 16, 17
JNI	Java Native Interface. 4, 5, 43, 45
LF-PRF	Length Flexible PseudoRandom Function. 30
MACs	Message Authentication Codes. 15
MOPE	Modular Order-Preserving Encryption. 28, 43
mOPE	Mutable Order-Preserving Encoding. 28, 36, 40, 42, 43, 47, 48
NIST	National Institute of Standards and Technology. 14, 44

NoSQL	Not only SQL. 10
OPE	Order Preserving Encryption. v, 4, 9, 23, 25, 26, 28, 33–36, 41–48
ORE	Order Revealing Encryption. 26, 28, 41–43, 49
POPF-CCA	Pseudorandom Order-Preserving Function under Chosen-Ciphertext Attack. 33
RSA-OAEP	Optimal Asymmetric Encryption Padding. 21
SQL	Search Query Language. 9, 10
VIL-PRF	Variable-Input-Length PseudoRandom Function. 30
VOL-PRG	Variable-Output-Length PseudoRandom function Generator. 30

1

INTRODUÇÃO

CONTEXTUALIZAÇÃO

Nos dias de hoje, somos confrontados com problemas que, para serem resolvidos, exigem máquinas com grande capacidade de processamento o que implica um investimento avultado na aquisição de computadores. Para contornar esta limitação, existe a possibilidade de requisitar máquinas remotas capazes de efetuar uma grande quantidade de cálculos em pouco tempo, quando comparadas com os computadores utilizados no nosso dia-a-dia. Como exemplo real disso, temos a *Amazon Elastic Cloud Computing* (EC2) que disponibiliza o acesso a super-computadores de forma a corresponder às exigências do cliente.

Contudo, para que haja confiança em máquinas remotas (*cloud*), é necessário garantir a privacidade dos dados, independentemente do *service provider*. Para conseguirmos garantir que entidades não autorizadas não tenham acesso à informação, podemos recorrer à criptografia, que hoje em dia se encontra num nível seguro. É, também, importante proteger estes dados de forma a que não fiquem danificados.

Com o objetivo de combinar segurança e disponibilidade, existem algoritmos capazes de cifrar a informação sem que a possibilidade de computação sobre a mesma seja posta em causa. Ou seja, é possível haver computação sobre dados cifrados. No entanto, a segurança e a performance podem ser vistas como propriedades inversamente proporcionais, isto é, quanto maior a segurança maior será a exigência em termos da computação da informação, e por isso, a performance poderá diminuir.

O trabalho realizado nesta dissertação contribuiu para o projeto europeu *SafeCloud* [1] cujo objetivo é oferecer uma plataforma de computação segura altamente escalável.

MOTIVAÇÃO

Nos últimos anos, tem-se verificado um crescimento no interesse em serviços baseados na *cloud*. Estas plataformas são capazes de oferecer computação e armazenamento em grande

1 http://www.safecloud-project.eu/

escala sem que se perca a disponibilidade destes serviços. O custo deste serviço é baixo quando olhamos para o proveito tirado a partir destas plataformas.

Existem acontecimentos que fazem baixar a confiança dos utilizadores nestas tecnologias, tais como ataques a serviços *cloud* com a intenção de invadir a privacidade dos utilizadores (Greene, 2015). Por outro lado, os governos têm posto em prática programas de vigilância que invadem a privacidade dos dados alojados nos servidores *cloud*. Com isto, os clientes destas plataformas levantam questões em termos de segurança e privacidade, já que, não é desejável o acesso de dados pessoais por parte de entidades não autorizadas. O facto de ser um serviço remoto retira, por si só, alguma confiança ao utilizador, já que este não tem a garantia formal da localização dos dados, nem quem tem acesso aos mesmos.

Por isso, existe uma preocupação em desenvolver sistemas capazes de oferecer computação e armazenamento de forma segura e privada, isto é, conseguir realizar estas tarefas sem que a informação seja revelada a entidades não autorizadas, como administradores dos fornecedores de serviço ou outros utilizadores maliciosos. Com estas garantias, seria possível que a confiança nestes serviços aumentasse.

Com o estado da evolução da tecnologia, existe uma necessidade de gerir grandes volumes de dados. Para isso, é necessário desenvolver ferramentas que consigam garantir a privacidade dos dados em plataformas que sejam capazes de gerir quantidades enormes de informação. As bases de dados não-relacionais são plataformas que conseguem corresponder a esta exigência em termos de volume de dados. Assim, podemos juntar as propriedades de segurança a um sistema facilmente escalável.

As soluções existentes passam por utilizar algoritmos criptográficos para codificar a informação, podendo assim ser garantida a confidencialidade e/ou integridade da mesma. No contexto de sistema de dados, existe a necessidade de, por vezes, efetuar cálculos ou comparações sobre os dados presentes nas bases de dados. Caso estes dados se encontrem cifrados, será necessário que estes sejam decifrados para assim serem executados os pedidos do cliente. Este processo de decifragem terá que ocorrer do lado do cliente, já que de outra forma a privacidade dos mesmos era posta em causa. Esta solução traz transtorno ao cliente, já que desta forma transfere todas as computações para o seu lado, desperdiçando assim os recursos oferecidos pela *cloud*.

Este processo de decifragem local poderá ser evitado utilizando cifras que permitam computação sobre os dados cifrados, fazendo com que toda a computação seja efetuada do lado do servidor. Isto pode implicar um *leak* de informação, que em alguns casos, poderá não ser crucial para garantir a privacidade total da informação. Por exemplo, se o servidor executar uma *query* em que necessite de comparar dois valores, ele irá saber qual o maior desses dois valores em termos de valor real de texto limpo, pois a propriedade de ordem de um número pode ser preservada no processo de cifragem para casos como este. No entanto, nunca saberá qual o real valor da informação, já que esta não é decifrada para fazer a

comparação dos dois valores. O *leak* dos dados também pode afetar os valores armazenados, mesmo que estes não sejam usados em computações, já que a preservação da ordem é válida para os valores armazenados. Esta solução, baseada numa base de dados com valores cifrados, provoca uma diminuição na performance do sistema quando comparamos com a versão que possui valores em texto limpo. Isto deve-se ao facto da execução de *queries* sobre criptogramas ser mais custosa. Contudo, esta diminuição nunca chega a ter a proporção dos processos associados à decifragem dos dados para a computação.

Uma outra solução incide na divisão da informação, de forma a que quem tenha acesso a uma parte dos dados, não tem acesso à informação total. Aqui também será permitido processamento sobre a informação armazenada, sem que seja necessária a junção de todas as partes, através da comunicação entre os intervenientes na divisão dos dados. Esta técnica é conhecida como *secret sharing* e consiste na distribuição de um segredo entre um grupo de participantes em que cada um dos quais recebe uma parte (*share*) desse segredo. Uma *share* sozinha não é capaz de revelar qualquer informação acerca do segredo. O segredo só pode ser reconstruido quando for combinado um número suficiente de *shares*. Este número não tem necessariamente de ser igual ao número de *parties*. Esta técnica é útil, já que oferece, ao mesmo tempo, disponibilidade e confidencialidade. Dos vários esquemas de *secret sharing* existentes, o mais conhecido pertence a *Shamir* (Shamir, 1979). Aqui o segredo partilhado pelas *parties* pode ser recuperado sem que seja necessário o acesso a todas as *shares* distribuídas. Esta característica é útil, já que por vezes o acesso a alguma *share* pode ser dificultado.

OBJETIVOS

O principal objetivo desta dissertação reside na análise de várias abordagens de OPE existentes e de outras ainda não implementadas. Para o caso das técnicas já implementadas, será feita uma análise prática de resultados e performance, para além da análise teórica, enquanto que nas restantes apenas será feita uma análise teórica.

Numa primeira fase é realizado o isolamento das bibliotecas dos respetivos sistemas para que possam ser analisadas ao pormenor. Esta análise incidirá na execução destas bibliotecas de forma isolada para assim ser possível a observação dos criptogramas gerados e dos tempos de execução. Irá ser feita também uma análise ao código para assim serem percetíveis os cálculos presentes por trás dos processos de cifragem e decifragem.

Depois de testadas as bibliotecas, o passo seguinte passará por permitir uma integração das bibliotecas com o projeto global. Como o motor de base de dados utilizado no projeto é o HBase, terá que ser feita uma ligação entre este e as bibliotecas testadas, já que a base de dados HBase está escrita na linguagem JAVA e as bibliotecas encontram-se escritas na linguagem C/C++. De forma a contornar esta adversidade irá recorrer-se ao Java Native

Interface (JNI), pois esta *framework* permite que um programa escrito em JAVA possa invocar funções escritas em C/C++.

No final, tendo a certeza do correto funcionamento das bibliotecas e da respetiva invocação por parte do JAVA, as bibliotecas serão integradas no projeto *SafeCloud* de forma a serem feitos mais testes de performance e confiabilidade, desta vez num ambiente mais formal. Também será analisado o impacto da utilização do JNI.

Com isto, espera-se que a base de dados seja capaz de executar as *range queries* de forma eficaz e sem erros.

ESTRUTURA DO DOCUMENTO

Este documento encontra-se dividido em 5 capítulos, pelo que o primeiro é este, a introdução.

No próximo capítulo, estado da arte, irão ser abordados conceitos importantes para a dissertação. Será feita uma análise ao armazenamento da informação já que é necessário para um contexto real. De seguida será abordada a segurança da informação apresentando técnicas padrão para privacidade dos dados, esquemas de autenticação e normas de segurança. Para terminar o capítulo, irá ser levantado o tema da funcionalidade sobre dados cifrados.

No capítulo seguinte irão ser focadas as técnicas que permitem *range queries* sobre dados cifrados. Será usada uma abordagem teórica para a apresentação das mesmas.

A validação experimental vem a seguir e nela podemos observar os resultados obtidos e algumas conclusões acerca dos mesmos.

Por fim, serão apresentadas as conclusões acerca do trabalho realizado e novos rumos que possam ser seguidos com este tema.

2

ESTADO DA ARTE

ARMAZENAMENTO DE DADOS

Cloud

O século XXI tem sido palco de desenvolvimentos notáveis. Este avanço traz vantagens e desvantagens. Neste caso de estudo importa incidir no enorme crescimento em termos de poder de computação e de capacidade de armazenamento, e na segurança destes sistemas. Como exemplo da evolução em termos computacionais, temos, para comparação, os nossos computadores convencionais dos dias de hoje e os computadores existentes na década anterior. Esta evolução também se verificou nos supercomputadores, como podemos verificar na figura 1 .

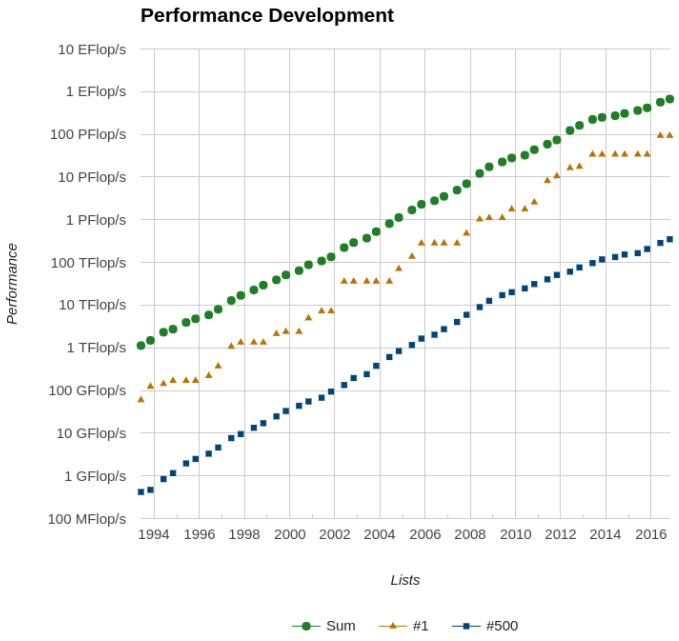


Figura 1: Crescimento dos supercomputadores ao longo dos anos. (TOP500, 2016)

A figura reflete o crescimento dos supercomputadores, ao longo de aproximadamente 20 anos, em três perspetivas, a evolução do melhor supercomputador, o crescimento do computador que se encontra na posição 500 em cada ano e o somatório de os supercomputadores presentes no *top500*, sendo que a medição dos supercomputadores é feita através do poder computacional em *Flop/s*.

No entanto, é impensável, para cada utilizador, adquirir fisicamente estas super máquinas capazes de executar informação gigantesca, já que isso implica grandes gastos e dificilmente se consegue compensar estes gastos. Para além disso, existem outros fatores que fazem com que a compra destas máquinas não seja rentável. Por um lado a utilização do *hardware* raramente poderia atingir os 100%, caso contrário os recursos disponíveis não seriam suficientes para a quantidade de informação necessária. Mas por outro lado, o facto de não atingir o máximo do seu desempenho resulta num desperdício de recursos. O cenário ideal é mesmo um serviço maleável com as necessidades de computação.

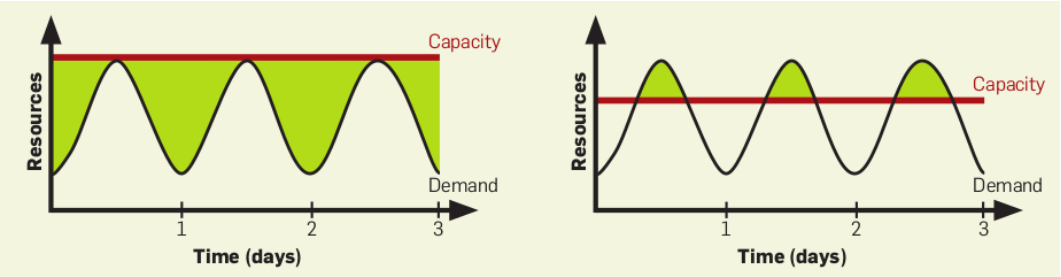


Figura 2: Despedício e escassez de recursos nas máquinas. Armbrust et al. (2010)

Na figura 2, no gráfico da esquerda, podemos observar um caso em que existe desperdício de recursos que não estão a ser utilizados representado pela sombra a verde. Por outro lado, no gráfico da direita, temos um exemplo de escassez de serviços, já que a capacidade é ultrapassada.

Para contornar esta e outras limitações, foi introduzido o conceito de *cloud*. Este serviço consiste na utilização de recursos que são geridos por empresas especializadas nesta área. Estes recursos são, normalmente, fornecidos por nós capazes de oferecer enormes quantidades de memória e grandes capacidades em termos de computação de informação. Este serviço é caracterizado por ser flexível, isto é, o cliente vai moldando a quantidade de serviço exigida consoante a utilização que necessitar. Esta caraterística pode prevenir a prática de falsos investimentos, como por exemplo, investir a mais em máquinas e utilizar apenas partes delas. Por outro lado, esta particularidade também é capaz de evitar um investimento insuficiente. Tudo depende do sucesso do cliente e das suas necessidades. Para além da elasticidade, notamos também uma sensação de recursos infinitos, já que, excetuando casos extremos, podemos sempre requisitar mais recursos. O facto deste serviço estar disponível em qualquer parte do mundo, é também uma mais-valia. Esta característica

permite, também, aceder aos serviços *cloud* a partir de qualquer dispositivo móvel, com acesso à internet, já que para usufruir dos recursos pretendidos só é necessário acesso à *Internet*. Com isto observamos que a evolução de toda a tecnologia contribui também para uma maior utilização da *cloud*.

O suporte e a transparência são outras características presentes nestes serviços. Sempre que há algum contratempo com as máquinas requisitadas, existem sempre especialistas capazes de resolver estes problemas. Posto isto, o cliente não tem de se preocupar com qualquer tipo de manutenção. No que toca a *software* podemos verificar este mesmo privilégio, já que não há qualquer preocupação no que toca a problemas de *software* relacionados com os serviços de *cloud*, tais como *bugs* ou licensas. Os *service providers*, por norma, têm disponível *software open source* como opção, mas caso o cliente opte por *software* licenciado poderá ter custos extra associados. A localização dos *data centers* responsáveis pelo serviço requisitado pelo cliente pode ser um problema em termos de disponibilidade, no entanto, este fator não pode interferir na qualidade e quantidade do serviço mesmo que os fornecedores de Internet sejam diferentes. Assim, podemos afirmar que a gestão dos *data centers* é totalmente transparente para o cliente. A configuração inicial dos servidores também pode ser evitada já que com a *cloud* temos os serviços prontos a utilizar. A aquisição de um *data center* implica uma instalação inicial que leva algum tempo e a manutenção poderá fazer com que o *data center* fique inoperável durante tempo indeterminado. Como os serviços *cloud* são taxados, maioritariamente, em função do tempo, os prejuízos poderão ser menores caso o serviço fique fora de serviço.

A *cloud* é constituída pelo *hardware* e *software* do *data center*. Existem 2 grandes tipos de *cloud*. Um deles é conhecido como *cloud* pública. Este é caracterizado por ser um serviço em que há partilha de *data centers* entre os utilizadores, isto é, uma máquina pode servir vários clientes, no entanto, os recursos pedidos pelo cliente não são afetados. Um outro tipo existente é a *cloud* privada. Aqui referimo-nos aos *data centers* internos de uma empresa ou entidade. Isto não quer dizer que tem de haver uma aquisição dos *data centers*. Apenas é acordada um compromisso de exclusividade dos *data centers*. Esta exclusividade permite um aumento da segurança, já que não há partilha de recursos entre os vários utilizadores.

Para que este serviço seja rentável existem algumas preocupações na ótica de um *service provider*. A construção destes *data centers* em localizações estratégicas de baixo custo, permitem baixar custos de electricidade, largura de banda, *software* e *hardware* em economias de larga escala. Estes fatores, combinados com multiplexagem estatística, que é capaz de aumentar a utilização dos serviços, contribuem para uma oferta razoável de recursos com custos não muito elevados para o cliente. A multiplexagem estatística é um modelo utilizado no transporte de pacotes na *Internet*.

Por vezes as empresas ou entidades trabalham com dados sensíveis, em que uma das prioridades é manter a privacidade dos mesmos. Por este motivo, é essencial que haja

confiança nos *service providers* para fazer a migração destes dados para a *cloud*. No entanto, existem fatores que fazem decrescer esta confiança. Para além dos ataques existentes a estas plataformas, há também a possibilidade do acesso à informação pela *backdoor* por parte dos administradores. Outra propriedade capaz de interferir na segurança do serviço é o nível de abstração que a *cloud* oferece para o cliente. Quanto mais baixo este for, mais exposta fica a *cloud*, e por isso, a segurança irá diminuir também.

Para combater este decréscimo existem algumas medidas capazes de aumentar a segurança da informação. Uma delas passa pela possibilidade de contratar uma outra entidade, para além do fornecedor de serviço *cloud*, capaz de ficar responsável pela segurança do sistema. Por outro lado, os próprios *service providers* também podem adotar técnicas capazes de aumentar a fiabilidade do serviço. A virtualização é uma das medidas que podem ser adotadas. É uma técnica poderosa capaz de impedir a maioria das tentativas de ataque por parte de utilizadores a outros utilizadores ou a outras insfrastruturas *cloud* subjacentes. Esta técnica também garante proteção do utilizador contra o fornecedor de serviço *cloud*. O *provider* encontra-se na camada mais baixa da *stack* do *software*, e consequentemente, conhece bem as técnicas utilizadas para garantir segurança. Uma outra técnica passa pela virtualização da memória, já que é difícil para um *provider* ter acesso ao conteúdo presente na memória de uma máquina virtual. Apesar desta garantia de segurança, nem todos os dados são virtualizáveis e nem toda a virtualização é estável. Por outro lado, para o caso de um disco rígido, não se pode dizer o mesmo quanto à dificuldade de acesso aos dados, já que a privacidade da informação aqui armazenada pode ser posta em causa sem que o disco perca a memória, ou podem existirem *bugs* de permissões capazes de colocar os dados expostos indevidamente.

Como medida de segurança *standard* temos a cifragem dos dados ao nivel do utilizador. Aqui garantimos que entidades não autorizadas, não consigam ter acesso ao valor dos dados, a não ser que possuam a *key*. No entanto, isto traz-nos problemas em termos de computação, já que a cifragem dos dados torna-os indistinguíveis e sem qualquer informação sobre o seu valor. Por exemplo, para serem executadas *queries* Search Query Language (SQL), por vezes, é necessário fazer comparações de valores ou somas. Com o objectivo de possibilitar estas operações sobre dados cifrados existem técnicas de cifragem capazes de permitir alguma computação, como cifras homomórficas e OPE. (Armbrust et al., 2010)

Bases de dados

Base de dados é um conjunto de informação organizada. A organização é feita de forma a que a informação armazenada tenha sentido e, desta forma, facilite uma pesquisa sobre os

dados. São a base do armazenamento de informação há décadas e são utilizadas sempre que é necessário armazenar alguns dados.

Esta terá de ser populada com dados para posteriormente responder a pedidos feitos por parte de um utilizador com o objetivo de obter a informação que se encontra armazenada. Para se fazer a solicitação destes dados, por vezes, é necessário mais do que um simples pedido, como o caso de comparações feitas acerca da informação alojada neste sistema de dados. Por exemplo, numa tabela onde residem várias informações de pessoas, podem ser pretendidas apenas as linhas das pessoas que possuam uma certa idade. Para isso é necessário analisar cada linha e fazer comparações em cada uma. É importante realçar os 2 principais paradigmas de sistemas de armazenamento de dados.

Um deles é o relacional. Uma base de dados relacional armazena a informação útil e, para além disso, guarda as relações existentes entre essa informação. Aqui os dados estão organizados em tabelas (ou relações) constituídas por várias colunas (cada coluna contem informação com as mesmas características) e por várias linhas (uma linha corresponde a uma entrada ou item). Cada linha possui um identificador único. Como exemplo pode-se equacionar um caso com vários clientes de uma loja. Para que os dados destes clientes sejam guardados de forma organizada sugere-se que seja feita uma tabela em que cada linha corresponde à informação de um cliente e cada coluna diz respeito ao tipo de informação (nome, telefone, idade...).

O outro paradigma é a base de dados não relacional, também conhecida como Not only SQL (NoSQL). São bases de dados vocacionadas para gerir grandes volumes de dados. Como se pode observar no nome, a principal característica das bases de dados desta natureza reside no facto de não existir relações entre a informação armazenada. Existem 3 tipos populares de bases de dados NoSQL (*key-value stores*, *Column-oriented databases* e *Document-based stores*). Tal como o nome indica, o *key-value stores* é um tipo que que guarda *values* indexados por uma *key*. É o tipo que mais se assemelha ao SQL. O tipo *Column-oriented databases* contem uma coluna extensível de dados relacionados, como alternativa ao armazenamento de conjuntos de informação numa tabela estruturada de colunas e linhas com campos uniformes para cada registo, como é utilizado em bases de dados relacionais. Por fim, o tipo *Document-based stores* guarda e organiza os dados como uma coleção de documentos, em vez de serem utilizadas tabelas estruturadas com campos uniformes para cada entrada. Neste tipo, os utilizadores podem adicionar um número variável de campos, de tamanho variável, também num documento. (Leavitt, 2010)

Neste ambiente já não se observam as propriedades Atomicidade, Consistência, Isolamento e Durabilidade (ACID). Aqui a preocupação é garantir um serviço facilmente escalável, insistindo, por isso, em características como disponibilidade e particionamento deixando de parte alguma consistência. (Mohamed et al., 2014)

Como o objetivo do projeto é a criação de uma ferramenta altamente escalável com a capacidade de gerir grandes volumes de dados, o paradigma não relacional será a escolha mais acertada. O *HBase* (George, 2011) é um exemplo deste paradigma. Para além de ser não relacional, este sistema é caracterizado por ser uma base de dados distribuída. Esta plataforma, que faz parte do projeto *open-source Apache Hadoop*, está escrita em *JAVA* e corre em cima de Hadoop Distributed Filesystem (HDFS) permitindo que algumas funcionalidades do projeto *BigTable* sejam utilizadas no *Hadoop*. Ou seja, permite que haja tolerância a falhas no armazenamento de enormes quantidades de dados esparsos (pequenas porções de informações retiradas de um conjunto gigantesco de dados irrelevantes ou do vazio, como por exemplo fazer procuras em que se pretende os 50 maiores items num grupo de 2 biliões de items ou procurar items diferentes de zero que representam menos do que 0.1% de um conjunto).

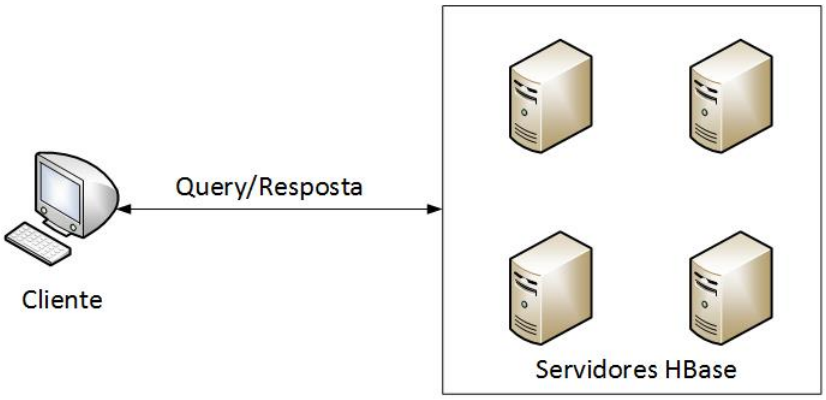


Figura 3: Esquema HBase

O *HBase* oferece funcionalidades como compressão, operações em memória e *Bloom filters* (Chang et al., 2008). Estes últimos são constituídos por uma estrutura probabilística capaz de aumentar a eficiência na pesquisa. Esta estrutura, quando questionada acerca da veracidade de um elemento fazer parte de um conjunto ou não, apenas responde "possivelmente está no conjunto"ou "definitivamente não se encontra no conjunto". As tabelas do HBase podem servir como *input* e *output* para tarefas relacionadas com o *MapReduce* executadas no *Hadoop* e podem ser acedidas através de uma *JAVA API* referente ao *HBase*. O acesso também é possibilitado através de outras *APIs* tais como *REST*, *Avro* ou *Thrift*. O armazenamento de dados é feito através da relação chave-valor orientado por coluna. Esta plataforma é indicada para operações de escrita e leitura em grandes conjuntos de dados com alto débito e baixas latências.

Figura 4: Teorema de CAP

O teorema de CAP foi apresentado em 2000, pelo professor *Eric Brewer*. Segundo este teorema, um sistema distribuído não pode ter as três propriedades representadas na figura 4 simultaneamente.

- **Consistência** Um *read* devolve o último *write* feito
- **Disponibilidade** Um pedido tem sempre uma resposta.
- **Tolerância a falhas** O sistema continua em funcionamento mesmo que haja interrupções na comunicação entre os nós

O sistema *HBase* é do tipo CP, ou seja, é um sistema capaz de garantir consistência e possui tolerância a falhas. (Han et al., 2011)

Esta plataforma é utilizada por grandes nomes na indústria tais como *Facebook* (Metz, 2010) e *Netflix* (Izrailevsky, 2011).

SEGURANÇA DA INFORMAÇÃO

A informação pode ser considerada como algo essencial e poderoso para a entidade que a mantiver. Como tal, pode ser necessário garantir que esta seja preservada em segurança, para assim não perder o valor. Para atingir esta segurança, a criptografia pode ser a chave do sucesso. Esta segurança pode implicar diferentes requisitos, tais como privacidade, autenticidade, autorização, certificação ou uma confirmação acerca da receção de uma mensagem. Nesta dissertação vai ser focada, principalmente, a privacidade/confidencialidade da informação. No entanto, a área de criptografia abrange todos os requisitos e pode ser capaz de os garantir a todos. É através da cifragem da informação que a criptografia consegue

garantir a privacidade dos dados, deixando estes totalmente impercetíveis a entidades não autorizadas. Apesar de garantir a privacidade dos dados, as cifras *standard* retiram funcionalidade sobre estes, já que num contexto de *cloud* ou base de dados, é impossível executar computações sobre criptogramas.

Standard Encryption

Na criptografia temos algumas técnicas *standard* bastante utilizadas no dia-a-dia por serem consideradas seguras. Por seguras entendemos que não são quebráveis. A quantidade de tempo que permanecem inquebráveis define a sua qualidade. Estas técnicas são conhecidas como cifras e são constituídas por funções matemáticas injetivas que transformam uma mensagem em texto limpo num criptograma. Os detalhes destas transformações são conhecidos, apenas existe uma propriedade que não é pública, a chave que pode ser caracterizada por ser simétrica ou assimétrica. Se a chave for simétrica, temos um sistema em que a cifragem e a decifragem utilizam a mesma chave, sendo nomeada de chave secreta. Num esquema de chave assimétrica, possuimos duas chaves diferentes, a chave pública que serve para cifrar a informação e a chave privada que intervém no processo de decifragem.

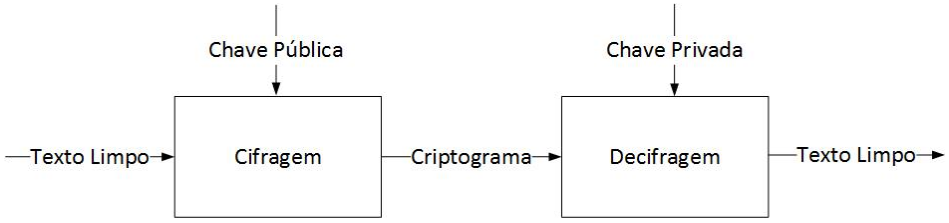


Figura 5: Esquema de uma cifra

Um tipo de cifra bastante utilizada em criptografia é conhecido como cifra por blocos. A função de cifragem é um poligrama, e trabalha sobre blocos de tamanho fixo (64/128 bits). Para além do processo normal de cifragem são necessárias tarefas extra para que os blocos sejam produzidos sempre com tamanho apropriado. Para isso, as unidades de partição e *padding* (enchimento) são responsáveis por manter os tamanhos dos blocos constantes. No processo de decifragem é aplicada aos blocos do criptograma a função inversa à cifragem, sendo o *padding* e a partição revertidos no final.

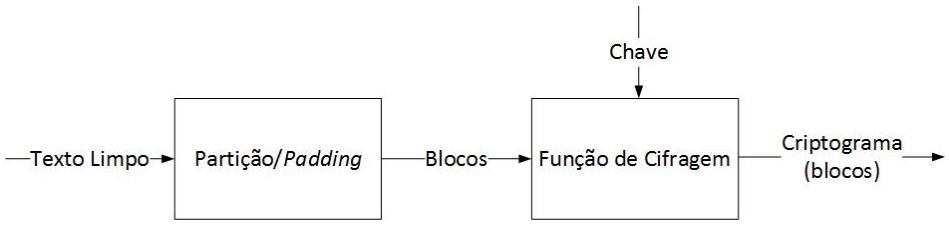


Figura 6: Esquema de uma cifra de blocos

O Advanced Encryption Standard (AES) é uma técnica considerada *standard* pela agência norte-americana National Institute of Standards and Technology (NIST). Esta cifra por blocos veio substituir o já obsoleto Data Encryption Standard (DES). Originalmente conhecido como algoritmo *Rijndael*, esta técnica foi desenvolvida por uma universidade Belga com o intuito de vencer um concurso e ficar como técnica padrão em criptografia. O *Rijndael* utiliza operações algébricas de forma inteligente, conseguindo garantir segurança através de operações complexas, mas com uma implementação eficiente tanto em *software* como em *hardware*. Este é caracterizado por ser uma cifra com tamanho de bloco variável ($N_b \times 32$ bits, $N_b = 4$, 6 e 8), sendo o tamanho de chave, também, variável ($N_k \times 32$ bits, $N_k = 4$, 6 e 8). O AES fixa o N_b de *Rijndael* em 4. O funcionamento desta cifra baseia-se no processamento de um estado 4×4 *bytes*. O AES é composto por *rounds*, sendo que o número de *rounds* depende de N_b e N_k. (Daemen and Rijmen, 1999). Cada *round* é uma rede de substituição-permutação e o estado de um *round* passa pela mesma sequência de operações:

- **AddRoundKey** Cada *byte* é combinado com a chave do *round*, derivada do algoritmo de *key schedule*
- **SubBytes** Operação não linear em que cada *byte* é substituído por um valor constante de uma *S-box*
- **ShiftRows** As linhas da matriz são rodadas de forma diferente (no AES *shifts* são 0, -1, 2 e 3 para a direita).
- **MixColumns** Cada coluna é sujeita a uma transformação linear invertível. Esta operação é equivalente a uma multiplicação de matrizes.

As duas primeiras operações visam principalmente a mistura e as duas últimas a difusão.

A criptoanálise de uma cifra é feita através na noção de ataque. Estes ataques podem classificados de várias formas tais como ataque de criptograma escolhido INDistinguishability under Chosen Ciphertext Attack (IND-CCA), texto limpo escolhido INDistinguishability under Chosen-Plaintext Attack (IND-CPA), entre outros. O objetivo é recuperar o texto limpo para um novo criptograma ou recuperar a própria chave, sendo este último mais complicado.

Funções de Hash Criptográficas

Uma função de *hash* criptográfica é um tipo de função de hash [1] com propriedades úteis no mundo da criptografia. A grande característica desta função reside no facto de ser uma função *one-way*, isto é, uma função cuja sua inversa é inviável. A única forma para que seja feita a correspondência entre um *output* e um *input*, passa por recorrer à força bruta, gerando *outputs* a partir de *inputs* escolhidos até se chegar à correspondência. Uma outra solução para atacar funções de *hash* reside na utilização de *rainbow tables* [2]. Neste tipo de funções o *input* costuma ser constituído pela mensagem e o *output* intitula-se como *digest* da mensagem.

A seguintes propriedades têm de ser verificadas numa função de *hash*

- Tem de ser determinística, ou seja, a mesma mensagem tem de gerar o mesmo *digest*
- Tem de ser eficiente, sendo que a sua computação tem de ser rápida
- Tem de ser impossível gerar mensagens a partir do seu valor de *hash*
- Uma pequena alteração na mensagem original deverá mudar o valor de *hash* de tal forma que este novo valor de *hash* não tenha qualquer relação com o antigo
- Tem de ser impossível encontrar duas mensagens com o mesmo valor de *hash*

Estas de funções de *hash* criptográficas são implementadas em aplicações que têm como objetivo oferecer segurança da informação. Estas aplicações podem gerar assinaturas digitas, Message Authentication Codes (MACs) ou outras formas de autenticação. Estas funções também podem ser usadas para mapear dados numa tabela de *hash*, identificar impressões digitais e ficheiros, identificar dados duplicados e detetar corrupção de dados.

Cifras simétricas autenticadas

Esquemas de cifras simétricas autenticadas (Bellare and Namprempre, 2000) são mecanismos de chave simétrica em que a mensagem M é transformada no criptograma C de forma a que a privacidade e autenticidade da mensagem nunca possam ser colocadas em causa. O principal objetivo de um esquema de cifragem simétrico é garantir privacidade da informação. A cifragem autenticada vem adicionar, a essa privacidade, a garantia de autenticidade das mensagens.

1 Uma função de *hash* tem como função mapear dados com tamanhos arbitrários para dados de tamanho fixo.
2 Uma *rainbow table* é uma tabela pré-computada, usada para reverter funções de *hash* criptográficas, frequentemente utilizadas para atacar *hashes* de palavras-passe

A arquitetura destes esquemas foi alvo das atenções durante alguns anos. Os primeiros esquemas de cifras simétricas autenticadas baseavam-se numa simples adição de redundância às mensagens, antes de serem cifradas. No entanto, muitos desses esquemas foram quebrados, devido à sua simplicidade. Um paradigma popular tem o nome de *generic compisition*, onde um esquema de cifragem simétrica, responsável pela privacidade, é combinado com um esquema de autenticação de mensagens, cujo objectivo individual é garantir a autenticidade das mensagens. Por exemplo, como esquema de cifragem podemos ter uma cifra por blocos CBC e para autenticação de mensagens pode ser utilizado HMAC ou CBC-MAC.

Quando se fala em integridade existem duas noções importantes. Uma delas, a Integrity of Plaintexts (INT-PTXT), garante que um criptograma, quando decifrado, origina uma mensagem cujo remetente é o responsável pela sua codificação. Este conceito é considerado um requisito de segurança natural. A outra noção, a Integrity of Ciphertexts (INT-CTXT) assegura que um criptograma não é produzido por outra entidade para além do seu remetente. Esta noção é considerada mais forte, em termos de segurança, do que a anterior.

A garantia de integridade, por si só, não garante a privacidade da informação num sistema. No caso de serem enviadas mensagens com forte autenticação MAC, se estas estiverem em texto limpo, apesar de garantida a integridade com a autenticação, o conteúdo da mensagem está completamente exposto.

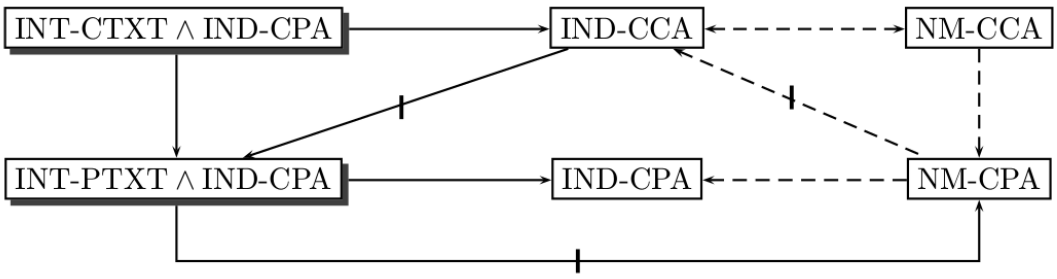


Figura 7: Relações entre os vários conceitos de segurança acerca de criptografia simétrica (Bellare and Namprempre, 2000)

A figura 7 mostra as relações existentes entre as várias noções mencionadas anteriormente. Uma "implicação" $A \rightarrow B$ significa que todo o esquema presente em A, também está presente em B. A "separação" $A \not\rightarrow B$ indica que existe, pelo menos, um esquema presente em A que não está presente em B. Apenas estão representados um pequeno conjunto de relações de forma explicita, no entanto a relação entre duas noções pode ser derivada a partir das relações que estão indicadas na figura 7. Por exemplo, IND-CCA não implica INT-CTXT $\wedge$ IND-CPA, visto que segundo o esquema da figura 7, temos INT-CTXT $\wedge$ IND-CPA $\rightarrow$ IND-CCA, ou seja, esta implicação iria contrariar o esquema. As linhas a tracejado

indicam as ligações existentes, em que a junções feitas entre a privacidade e autenticidade (INT-CTXT $\wedge$ IND-CPA e INT-PTXT $\wedge$ IND-CPA) não intervêm.

A "não-maleabilidade"não garante qualquer tipo de integridade das mensagens, isto é, a "não-maleabilidade"apenas previne a geração de criptogramas a partir de outros criptogramas, enquanto que a integridade implica que seja impossível gerar criptogramas a partir de novas mensagens, mesmo que estas não tenham qualquer relação com as mensagens subjacentes a qualquer criptograma. Há que salientar que a integridade das mensagens não implica integridade dos criptogramas.

Composição genérica

Existem várias formas de se construir esquemas de cifras simétricas autenticadas. Para este caso vamos considerar a *Generic Composition*. Esta consiste na combinação simples entre um esquema de cifragem simétrica padrão e um esquema de autenticação de mensagens.

No caso de um esquema de cifragem, a representação deste pode ser feita através das letras $\mathcal{SE}$. Ainda no mesmo esquema os algoritmos de cifragem e decifragem são representados pelas letras $\mathcal{E}$ e $\mathcal{D}$, respetivamente. Para o caso da autenticação de mensagens, a representação é feita pela sigla $\mathcal{MA}$. Neste esquema, os algoritmos *tagging* e *tag-verifying* são representados pelas letras $\mathcal{T}$ e $\mathcal{V}$, respetivamente.

Na composição genérica são combinados esquemas de cifragem e esquemas de autenticação de mensagens de forma a que sejam criados esquemas de cifragem simétrica autenticados. Estes podem ser INT-PTXT $\wedge$ IND-CPA ou INT-CTXT $\wedge$ IND-CPA, por exemplo. Esta composição é chamada de genérica, pois nos algoritmos de cifragem autenticada são sempre utilizadas caixas negras, ou seja, os algoritmos não são especificados. Em cada caso, K_e é a chave de cifragem e K_m é a chave para autenticação de mensagem. As chaves são escolhidas de forma independente uma da outra.

- *Encrypt-and-MAC (E&M)*: $\overline{\mathcal{E}}(K_e||K_m, M) = \mathcal{E}(K_e, M)||\mathcal{T}(K_m, M)$. [3] - Cifrar o texto limpo e juntar o MAC do texto limpo. A decifragem e a verificação são feitas decifrando, primeiro, o criptograma e, depois, é verificada a *tag*.

- *MAC-then-encrypt (MtE)*: $\overline{\mathcal{E}}(K_e||K_m, M) = \mathcal{E}(K_e, M||\mathcal{T}(K_m, M))$ - Anexar o MAC à mensagem e cifrar tudo junto. Os processos de decifragem e de verificação são constituídos, primeiramente pela descodificação do criptograma para se obter a mensagem e a *tag*, e por fim, pela verificação da *tag* obtida.

- *Encrypt-then-MAC (EtM)*: $\overline{\mathcal{E}}(K_e||K_m, M) = C||\mathcal{T}(K_m, C)$ em que $C = \mathcal{E}(K_e, M)$ - Cifragem da mensagem para se obter o criptograma C e depois é anexado o MAC a C. A

3 "||"indica uma operação que combina várias *strings* numa só, de tal forma que as *strings* constituintes só são recuperáveis a partir da *string* final (Se o comprimento das strings for fixo e conhecido, a concatenação será suficiente para esta operação)

decifragem e verificação são compostas pela verificação da *tag*, seguida da decifragem do criptograma.

Indistinguibilidade de criptogramas

Indistinguibilidade de criptogramas é uma propriedade que impede o adversário de fazer correspondência entre criptogramas e mensagens em texto limpo. Se falarmos de segurança num criptossistema e se apenas nos referirmos à indistinguibilidade, um criptossistema é considerado seguro se, dada uma cifragem de uma mensagem escolhida aleatoriamente a partir de um espaço de 2 mensagens, geradas pelo adversário, este não consegue adivinhar a escolha, por parte do sistema, da mensagem cifrada com uma probabilidade maior do que a escolha aleatória. Neste exemplo de 2 mensagens, a probabilidade de acertar é de $\frac{1}{2}$. Caso a probabilidade do adversário conseguir distinguir a mensagem cifrada seja maior do que $\frac{1}{2}$, então este adversário tem uma vantagem em identificar a mensagem e por isso o sistema não é considerado seguro em termos de indistinguibilidade. Com isto entende-se que um adversário não pode retirar informações extra, acerca do texto limpo, a partir do criptograma. Portanto, a probabilidade de um adversário conseguir distinguir mensagens cifradas não pode ser maior do que a probabilidade de uma escolha totalmente aleatória.

Existem várias definições dentro do ramo da indistinguibilidade. Cada uma delas possui um nível de segurança próprio. Algumas das definições existentes são IND-CPA, IND-CCA e INDistinguishability adaptive Chosen Ciphertext Attack (IND-CCA2). Qualquer uma das definições assuguram a garantia da anterior, isto é, um sistema seguro sob IND-CCA, é seguro sob IND-CPA. O mesmo se verifica para o IND-CCA2, já que um sistema seguro sob esta definição é seguro sob IND-CPA e IND-CCA. A propriedade de indistinguibilidade sob ataque de texto limpo escolhido IND-CPA é considerada uma garantia mínima obrigatória por grande parte dos criptossistemas de chave pública seguros.

De forma a generalizar as várias definições vamos considerar um esquema criptográfico que possui um gerador de chaves KG, capaz de gerar pares de chaves (K_E, K_D), um algoritmo de cifragem E e um algoritmo de decifragem D. A cifragem é sempre revertível, no entanto as chaves para cifrar e decifrar podem ser diferentes (criptografia de chave pública): $D(K_D, E(K_E, M)) = M$. Sempre que é feita a alusão a um adversário, é subentendido que as operações feitas pelo mesmo são realizadas em tempo polinomial.

Ataques sob texto limpo escolhido (CPA)

Num ataque sob texto limpo escolhido (Chosen-Plaintext Attack (CPA)), o adversário pode gerar um certo número de mensagens e obter os criptogramas correspondentes. Isto pode ser formalizado se o adversário tiver total acesso ao oráculo de cifragem, que é visto como uma "caixa negra". CPA tornou-se ainda mais importante no contexto de criptografia de

chave pública. A partir do momento em que a chave é pública, os adversários podem cifrar qualquer mensagem que pretendam. (Anderson, 2008)

Estes ataques são vulgares em vários cenários de contexto real. Isto pode ser demonstrado olhando para a história militar da Segunda Guerra Mundial. Em Maio de 1942, os criptoanalistas da marinha norte-americana descobriram que as tropas japonesas planearam um ataque na ilha *Midway*, situada no oceano Pacífico. Eles conseguiram revelar este segredo intercetando uma mensagem cifrada que continha o fragmento "AF", que acreditavam que correspondia à mensagem "Midway island". No entanto, as tentativas para convencer os responsáveis dos planeamentos foram todas fracassadas. Ninguém acreditava que *Midway* poderia ser um alvo. Posto isto, os criptoanalistas preparam um outro plano. Eles pediram às forças norte-americanas presentes em *Midway* que enviassem uma mensagem não cifrada dizendo que havia pouca água potável naquela zona. Os japoneses intercetaram a mensagem e informaram os seus superiores que "AF"tinha pouca água potável. Com isto os criptoanalistas da marinha tinham a prova de que "AF"corresponderia a *Midway* e as forças norte-americanas três porta-aviões para aquela localização. Com isto, os norte-americanos conseguiram salvar *Midway* e ainda provocaram grandes perdas às forças japonesas. Esta batalha foi considerada o ponto de viragem da guerra entre os Estados Unidos e o Japão no Pacífico.

Voltando ao tema de CPA, os criptoanalistas da marinha fizeram, exatamente, um ataque sob texto limpo escolhido. Embora de maneira indireta, eles foram capazes de "solicitar"aos japoneses que cifrassem a palavra *Midway* com o objetivo de obter informação acerca de outro criptograma gerado a partir desta palavra que seria novamente intercetado. Se o algoritmo de cifragem usado pelos japoneses fosse seguro contra ataques sob texto limpo escolhido, a estratégia dos norte americanos não teria resultado, pois os criptogramas teriam de ser diferentes (e a história poderia ser bem diferente!). Há que salientar que nunca foi intenção dos japoneses atuarem como um "oráculo de cifragem"para os Estados Unidos, e portanto foram eles a sentirem a necessidade de segurança contra CPA.

Ataques sob texto limpo escolhido não são perigosos apenas para aplicações militares. Existem vários casos em quem o adversário consegue influenciar as mensagens que são cifradas por uma entidade honesta, apesar de ser bastante incomum existirem casos em que o adversário tem total controlo da escolha da informação a ser cifrada. Considerando o caso em que temos vários servidores a comunicarem entre si de forma segura, cifrando a informação. Contudo, as mensagens trocadas entre os servidores são baseadas em pedidos, internos ou externos, que eles receberam, que por sua vez, feitos por utilizadores que podem ser considerados como adversários do sistema. Portanto, estes adversários conseguem influenciar a informação que é cifrada por estes servidores. Assim, estes sistemas têm que se precaver destes ataques e protegerem-se contra ataques sob texto limpo escolhido. (Katz and Lindell, 2008)

Ataques sob criptograma escolhido (CCA)

Chosen-Ciphertext Attack (CCA) é um modelo de ataque em que o adversário pretende obter informação parcial acerca de um criptossistema. O adversário escolhe um criptograma e consegue obter a mensagem decifrada correspondente a esse criptograma, sem conhecer a chave que foi usada neste processo. Existe a possibilidade de serem submetidos um ou mais criptogramas para o oráculo de decifragem. Através deste processo, o adversário pode conseguir descobrir a chave secreta usada.

Existem vários esquemas que são vulneráveis a Ataques sob criptograma escolhido, CCA. Um deles é o criptossistema de El Gamal (ElGamal, 1985) que é semanticamente[4] seguro contra CPA, no entanto, é vulnerável a CCA. Outro esquema vulnerável a CCA é o *plain RSA*. Esta vulnerabilidade pode ser demonstrada através do exemplo em que o adversário deseja saber a decifragem $m \equiv c^d \pmod{n}$ de um criptograma c. Este, pode escolher um inteiro aleatório s e pedir a decifragem "inocente"da mensagem $c' \equiv s^e c \mod n$. A partir da resposta $m' \equiv (c')^d$, é fácil recuperar a mensagem original, já que $m \equiv m's^{-1} \pmod{n}$. Outra vulnerabilidade atualmente desvendada é o facto do bit menos significante de RSA ser tão seguro como toda a mensagem. Em particular, existe um algoritmo que consegue decifrar o criptograma se existir um outro algoritmo que consiga prever o bit menos significativo de uma mensagem, fornecendo apenas o criptograma e a chave pública. Consequentemente, não é necessário para um adversário conseguir a mensagem completa decifrada num CCA. Se forem revelados alguns bits por cada criptograma escolhido pode ser suficiente para que o ataque resulte. (Bleichenbacher, 1998)

Se um sistema for vulnerável a este tipo de ataques, tem de haver uma preocupação extra para evitar situações em que o adversário consiga decifrar criptogramas escolhidos por ele, ou seja, negar o acesso ao oráculo de decifragem. Esta preocupação pode ser bastante complicada, uma vez que criptogramas escolhidos de forma parcial podem provocar ataques subtis. Uma forma de dar a volta a esta vulnerabilidade passa por assinar as mensagens e assim provar a identidade do autor das mensagens. Isto faz com que só sejam permitidos ataques quando não é utilizado *hashing* na mensagem a ser assinada. Optimal Asymmetric Encryption Padding (RSA-OAEP) ou *Cramer-Shoup* são exemplos de esquemas seguros contra este tipo de ataques.

Este tema e o anterior serão mais aprofundados nas secções seguintes.

IND-CPA

A ideia básica por trás de um ataque *chosen-plaintext* encontra-se na permissão dada ao adversário em cifrar várias mensagens escolhidas *on-the-fly* de forma adaptativa. O adversário

4 Segurança semântica: De acordo com esta definição, um adversário não pode ter qualquer tipo de informação acerca de uma mensagem cifrada. Esta garantia apenas é válida quando estamos perante um adversário completamente passivo.

$\mathcal{A}$ pode interagir livremente com um *encryption oracle*, que é visto como uma "caixa negra"que cifra mensagens escolhidas por $\mathcal{A}$. Este processo é feito usando uma chave secreta k que permanece incógnita para o adversário. Quando $\mathcal{A}$ interage com o oráculo fornecendo a mensagem m em texto limpo como *input*, o oráculo retorna o criptograma $c \leftarrow Enc_k(m)$ como resposta. Para *Enc* ser considerada uma função pseudo-aleatória, o oráculo usa novos *coins* aleatórios cada vez que responde a uma *query*.

A definição de segurança exige que $\mathcal{A}$ não pode conseguir distinguir a cifragem de duas mensagens arbitrárias, mesmo no caso em que $\mathcal{A}$ tem acesso ao oráculo de cifragem.

Neste caso, voltamos ao exemplo dado anteriormente. O adversário gera 2 mensagens com o mesmo comprimento. O sistema criptográfico escolhe, aleatoriamente, uma das mensagens para cifrar. O adversário tenta adivinhar qual das mensagens foi escolhida, a partir do criptograma que é retornado pelo sistema.

A segurança neste modelo é expressa como a máxima vantagem de um adversário no seguinte jogo:

- O adversário recebe uma chave pública que vai tentar atacar
- Quando estiver preparado, o adversário vai escolher duas mensagens em texto limpo (x_0, x_1)
- O adversário recebe um criptograma correspondente a uma das mensagens (x_b)
- O adversário tem de adivinhar o valor de b em que $b \in 0, 1$

Se o adversário tiver acesso ao oráculo de cifragem, ele pode cifrar as mensagens que pretender e assim comparar com o criptograma que foi cifrado pelo sistema. Isto faz com que o adversário consiga acertar no criptograma com a probabilidade de 1, caso o algoritmo seja determinístico. Consideremos o caso em que o adversário gera duas mensagens, m_0 e m_1, e recebe do oráculo um criptograma referente a uma das duas mensagens geradas, $c \leftarrow Enc_k(m_b)$. Como o adversário tem acesso ao oráculo $Enc_k(\cdot)$, ele pode cifrar as próprias mensagens obtendo assim $c_0 \leftarrow Enc_k(m_0)$ e $c_1 \leftarrow Enc_k(m_1)$. Desta forma, pode comparar c_0 e c_1 com o c devolvido pelo criptossistema. Se $c = c_0$, então $b = 0$ e se $c = c_1$, então $b = 1$. No entanto, não há nenhum esquema criptográfico determinístico que pode ser seguro contra IND-CPA. Em vez disso, um esquema seguro contra IND-CPA tem de ser probabilístico. Ou seja, o oráculo necessita de usar *fresh coins* aleatórios, cada vez que é executada uma *query*, para que os criptogramas gerados sejam sempre distintos. Desta forma, o adversário não consegue fazer a correspondência entre as mensagens e o criptograma dado pelo criptossistema, no caso do exemplo dado anteriormente.

O conceito principal introduzido por este cenário é o de *polynomial bound*. A segurança do sistema é definida pela probabilidade de acertar na mensagem cifrada em tempo útil.

O facto das mensagens possuírem o mesmo comprimento, previne de casos em que o adversário compara o comprimento dos criptogramas. (Katz and Lindell, 2008)

A vantagem de um adversário neste tipo de ataques pode ser definida pela seguinte fórmula:

$$Adv_{\mathcal{SE}}^{ind-cpa}(A) = Pr[Exp_{\mathcal{SE}}^{ind-cpa-1}(A) = 1] - Pr[Exp_{\mathcal{SE}}^{ind-cpa-0}(A) = 1] \quad (1)$$

IND-DCPA

O conceito de INDistinguishability under Distinct Chosen-Plaintext Attack (IND-DCPA) foi proposto (Bellare et al., 2004) com o intuito de definir a segurança de esquemas de cifragem simétricos determinísticos. A principal razão para a necessidade deste conceito é o facto de um algoritmo criptográfico simétrico revelar a igualdade de mensagens. Adaptando a situação anterior ao IND-DCPA, temos o adversário A limitado a submeter *queries* distintas para o oráculo. Caso contrário, estamos perante um ataque trivial. Isto é, supondo que A submete as *queries* $(m_0^1, m_1^1), ..., (m_0^q, m_1^q)$, é exigido que $m_b^1, ...m_b^q$ sejam todas distintas para $b \in 0, 1$. (Bellare et al., 2004)

A vantagem do adversário A pode ser enunciada da seguinte forma:

$$Adv_{\mathcal{MA}}^{ind-dcpa}(A) = Pr[Exp_{\mathcal{MA}}^{ind-dcpa-1}(A) = 1] - Pr[Exp_{\mathcal{MA}}^{ind-dcpa-0}(A) = 1] \quad (2)$$

IND-OCPA

Como qualquer criptograma gerado por um esquema OPE revela informações acerca da ordem do valor do texto limpo, existe a necessidade de estender a definição de IND-CPA. Em particular, existirá uma maior exigência nas *queries* feitas pelo adversário A para ter o mesmo *"order pattern"*. Ou seja, esta definição exige que os criptogramas de uma sequência de mensagens não pode revelar mais nada a não ser a ordem dos mesmos.

Qualquer *query* (m_0, m_1) que seja submetida ao oráculo satisfaz a seguinte propriedade: $|m_0| = |m_1|$. As mensagens têm de ter o mesmo *"order pattern"*, isto é, $m_0^i < m_0^j$ se e só se $m_1^i < m_1^j$ para todo $1 \leq i, j \leq q$. (Boldyreva et al., 2009) Para um adversário A, a sua vantagem num esquema OPE pode ser definida pela seguinte fórmula:

$$Adv_{OPE}^{ind-ocpa}(A) = Pr[Exp_{OPE}^{ind-ocpa-1}(A) = 1] - Pr[Exp_{OPE}^{ind-ocpa-0}(A) = 1] \quad (3)$$

IND-CCA

Nesta definição, para além do adversário conseguir submeter mensagens com o objetivo de as cifrar, também podem ser submetidos criptogramas para serem decifrados. Isto significa

que o adversário consegue cifrar e decifrar todo o tipo de mensagens antes de obter o criptograma que ele tem de adivinhar.

A segurança neste modelo é expressa como a máxima vantagem de um adversário no seguinte jogo:

- O adversário tem acesso a um oráculo onde pode submeter criptogramas para serem decifrados
- Quando estiver preparado, o adversário vai escolher dois criptogramas (x_0, x_1)
- O adversário recebe a mensagem em texto limpo, correspondente a um dos criptogramas (x_b)
- O adversário tem de adivinhar o valor de b em que $b \in 0, 1$

IND-CCA considera a possibilidade de várias interações entre o sistema e o adversário, o que implica que a segurança não vai enfraquecendo com o tempo. Outra consequência do IND-CCA é a não-maleabilidade, ou seja, o criptograma não pode ser alterado para ser decifrado em $M' \neq M$. (Katz and Lindell, 2008)

Fórmula:

$$Adv_{\Pi}^{ind-cca}(A) = Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)} \implies 1] - Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\$|\cdot|)} \implies 1] \quad (4)$$

IND-CCA2

Esta definição acaba por ser idêntica à anterior, no entanto existe uma ligeira diferença. Neste caso, o adversário pode cifrar e decifrar após receber o C. No entanto, não pode decifrar o C.

- O adversário tem acesso a um oráculo onde pode submeter criptogramas para serem decifrados
- Quando estiver preparado, o adversário vai escolher dois criptogramas (x_0, x_1)
- O adversário recebe a mensagem em texto limpo, correspondente a um dos criptogramas (x_b)
- O adversário pode submeter mais criptogramas para o oráculo
- O adversário tem de adivinhar o valor de b em que $b \in 0, 1$

IND-CCA2 sugere que decifrar criptogramas depois de conhecer o criptograma C consegue dar uma vantagem significativa em alguns esquemas criptográficos, desde que os pedidos para decifrar sejam influenciados pelo criptograma recebido. (Bleichenbacher, 1998)

Fórmula:

$$Adv_{\Pi}^{ind-cca2}(A) = Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)} \implies 1] - Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\$|\cdot|),\mathcal{D}_K(\cdot)} \implies 1] \quad (5)$$

IND-CCA3

A definição é igual à apresentada anteriormente, IND-CCA2, execepto num aspecto: Aqui, o adversário não pode decifrar criptogramas. Em vez disso, este tem acesso a um oráculo que retorna sempre $INVALID$.

Vamos considerar o adversário A um algoritmo com a acesso a um oráculo e $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ um esquema criptográfico. A medida da vantagem (authenticated) INDistinguishability adaptive Chosen Ciphertext Attack (IND-CCA3) pode ser definida pela seguinte fórmula:

$$Adv_{\Pi}^{ind-cca3}(A) = Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)} \implies 1] - Pr[A^{\mathcal{E}_K(\$|\cdot|),\perp(\cdot)} \implies 1] \quad (6)$$

Tal como se sucede em IND-CCA2, o oráculo $\mathcal{E}_K(.)$, com um input M, retorna o criptograma $\mathcal{E}_K(M)$, e o oráculo $\mathcal{E}_K(\$|\cdot|)$, com um input M, retorna o criptograma de $|M|$ bits aleatórios. O oráculo $\mathcal{D}_K(\cdot)$, com um input C retorna o texto limpo $\mathcal{D}_K(C)$. O oráculo $\perp(\cdot)$ retorna $INVALID$ para qualquer input que lhe seja dado.

Para esta e para as definições anteriores, assume-se sempre que o adversário produz *queries* bem estruturadas, não repete uma *query* e nunca solicita a decifragem do criptograma C, sendo este o resultado da última cifragem feita. As 2 primeiras suposições são genéricas, enquanto que a terceira previne uma distinção trivial por parte do adversário.

(Shrimpton, 2004)

De notar que as técnicas *standard* cumprem todos os requisitos em termos de segurança, no entanto não permitem computação provocando assim limitações na área de computação remota. Para contornar esta adversidade existem técnicas capazes de preservar algumas propriedades no processo de cifragem, como é o caso do OPE. Desta forma, as *queries* são executadas, apenas, do lado do servidor, sem expor a informação a entidades externas.

FUNCIONALIDADE SOBRE DADOS CIFRADOS

De forma a contornar o facto da *standard encryption* não permitir computação, foram criadas técnicas de cifragem capazes de contornar este problema, tais como o OPE. Este esquema, de carácter determinístico, é capaz de preservar a ordem numérica do texto limpo nos criptogramas. A história desta técnica é longa e vai até aos tempos da Primeira Guerra Mundial onde eram utilizadas comunicações em código. As mensagens eram cifradas e decifradas de acordo com um livro em que continha o texto limpo e o criptograma correspondente. Tanto os criptogramas como as correspondências em texto limpo eram organizadas por ordem

numérica ou alfabética. Desta forma, apenas era necessária uma cópia destas informações para se proceder à cifragem e/ou decifragem. (Bauer, 2013)

No entanto, só em 2004 é que foi feita uma proposta formal de OPE (Agrawal et al., 2004), cujo principal objetivo era oferecer uma solução capaz de resolver os problemas relacionados com a performance quando eram executadas *range queries* sobre dados cifrados, já que até aqui era necessário proceder-se à decifragem da informação para que estas *queries* fossem executadas. Para além disso, com esta técnica, uma base de dados presente num servidor remoto não confiável poderia conseguir realizar comparações de carácter numérico ou alfabético com criptogramas sem os decifrar, não pondo em causa a privacidade da informação. Era possível aliar a segurança com a eficiência.

Para além de oferecer *range queries* eficientes, OPE também contribui para que a indexação e o processamento de *queries* sejam realizados com a mesma exatidão e eficiência que se pode observar em bases de dados com texto limpo, visto que para uma *query* apenas é necessário cifrar a e b e o servidor consegue localizar os dados desejados em tempo logarítmico através de estruturas baseadas em árvores. Depois da primeira publicação, OPE foi sugerido para ser utilizado em agregação de informação cifrada em sensores de rede (Girao et al., 2005). Também foi recomendado o uso desta técnica como ferramenta para aplicar técnicas de processamento de sinal na proteção de conteúdo multimédia (Erkin et al., 2007). Esta técnica também é usada no contexto de servidores de *email*, aplicações *web*, Customer Relationship Management (CRM) *software*, entre outras. (Kaur and Singh, 2013)

Em termos de análise de segurança, o cenário ideal para um esquema OPE passa por não revelar mais informação para além da ordem do valor em texto limpo, já que esta "fuga"de informação é necessária para que as *range queries* sejam executadas sobre os criptogramas. Existiram várias propostas de OPE em que eram tidos em conta, apenas adversários com formas específicas no roubo da informação e, portanto não eram dadas garantias para adversários mais gerais. A primeira proposta que encarou o problema de forma rigorosa, (Boldyreva et al., 2009), demonstrou que era impensável alcançar o cenário ideal em esquemas OPE, sobre certas suposições. Em consequência disto, os autores deste esquema, demonstraram, mais tarde, que este algoritmo revelava pelo menos metade dos bits do texto limpo (Boldyreva et al., 2011). Para além desta, foram apresentadas, posteriormente, propostas que garantiam total segurança para esquemas OPE. Exemplo disso, temos a *CryptDB* (Popa et al., 2011), que apresenta uma técnica OPE com as garantias ideias de segurança (Popa et al., 2013). Isto quer dizer que os criptogramas apenas revelam a ordem do texto limpo, mantendo todas as outras informações privadas. Outra proposta é o Order Revealing Encryption (ORE) (Boneh et al., 2015). Esta, mais recente, apresenta também grandes garantias de segurança no que toca a técnicas OPE.

Tal como as outras técnicas criptográficas, OPE segue a seguinte propriedade:

$$m = Dec(K, Enc(K, m)) \tag{7}$$

Isto significa que uma mensagem m cifrada com a chave K, quando decifrada com a mesma chave tem de corresponder à mensagem original sem qualquer alteração. A seguinte propriedade apenas é seguida pelas técnicas OPE:

$$Enc(K, m1) > Enc(K, m2) \quad iff \quad m1 > m2 \tag{8}$$

Ou seja, a ordem numérica da mensagem em texto limpo corresponde à ordem numérica das cifras, permitindo assim que possam ser realizadas comparações sem decifrar a informação. (Boldyreva et al., 2009)

CONCLUSÃO

Neste capítulo foi apresentado, inicialmente, o funcionamento do armazenamento de dados, desde as infraestruturas e intermediários até à organização da informação. O armazenamento de informação será a base para por em prática os esquemas criptográficos.

Depois disto, foi demonstrado o *standard* das técnicas criptográficas e os padrões de segurança. Foram apresentadas as vertentes de privacidade e autenticidade. Os esquemas mais poderosos existentes são considerados seguros, no entanto não permitem funcionalidade sobre os dados cifrados. A secção seguinte apresenta formas de conseguir funcionalidade sobre dados cifrados, mais propriamente, executar *range queries* sobre os mesmos, abdicando um pouco da performance observada nas cifras *standard*.

3

APLICAÇÕES PRÁTICAS PARA PESQUISAS SOBRE DADOS CIFRADOS

Neste capítulo serão apresentados algoritmos capazes de executar *range queries* sobre dados cifrados, sem os decifrar. A primeira técnica a ser apresentada foi proposta por Boldyreva em 2009, o OPE. Depois serão abordadas as técnicas Modular Order-Preserving Encryption (MOPE) e Mutable Order-Preserving Encoding (mOPE), que foram lançadas em 2011 e 2013, respetivamente. Por fim será apresentada uma proposta recente, de 2015, a ORE.

ORDER-PRESERVING SYMMETRIC ENCRYPTION (OPE)

Este esquema (Boldyreva et al., 2009) foi o primeiro a tratar do problema de *range queries* sobre dados cifrados de forma rigorosa. Os seus autores demonstraram que é inviável alcançar a segurança ideal em OPE, sobre certas suposições implícitas (Boldyreva et al., 2011). Posto isto, eles ofereciam garantias mais fracas em termos de segurança e mais tarde foi demonstrado que existiam fugas de informação em pelo menos metade dos bits do texto limpo.

A cifragem e decifragem deste esquema consiste num mapeamento baseado numa distribuição probabilística hipergeométrica, que passo a explicar na secção seguinte.

HyperGeometric Distribuition

De forma a mostrar a relação entre uma distribuição hipergeométrica e uma função *order-preserving* vamos considerar um gráfico em que o eixo do x é composto por inteiros de 1 até M e o eixo do $y = f(x)$ possui inteiros de 1 até N. Dado S, um conjunto de M inteiros distintos pertencentes a $[N]$, pode ser construída uma função *order-preserving* de $[M]$ para $[N]$ mapeando cada $i \in [M]$ com o iésimo elemento mais pequeno de S. Assim, uma combinação *M-out-of-N* corresponde a uma função de *order-preserving*. Por outro lado, considera-se uma função de *order-preserving* de $[M]$ para $[N]$. A imagem desta função f define conjunto de M objetos distintos de $[N]$, por isso uma função de *order-preserving* corresponde a uma combinação única *M-out-of-N*.

Posto isto, chega-se a esta igualdade, definida para $M, N \in \mathbb{N}$ para qualquer x, $x+1 \in [M]$, $y \in [N]$:

$$Pr_{f \xleftarrow{\$} OPF_{[M],[N]}}[f(x) \leq y < f(x+1] = \frac{\binom{x}{y}.\binom{N-y}{M-x}}{\binom{N}{M}} \tag{9}$$

Considerando o seguinte modelo de bolas e sacos: imaginando o caso em que temos 100 bolas num saco, das quais 20 são pretas e as restantes 80 são brancas. Se tirarmos 1 bola do saco, é intuitivo que a probabilidade dessa bola ser preta é 20%. Imaginemos agora que eram retiradas y bolas do saco sem reposição. O número de bolas pretas retiradas seria X e este segue uma distribuição hipergeométrica. Este X é definido pela seguinte equação em que x é a probabilidade pretendida ($X = x$), N é o número total de bolas, M é o número de bolas pretas e y a amostra de bolas a serem retiradas:

$$P_{HG}(x; N, M, y) = \frac{\binom{x}{y}.\binom{N-y}{M-x}}{\binom{N}{M}} \tag{10}$$

O cálculo do coeficiente binomial é dado pela seguinte equação:

$$\binom{x}{y} = \frac{y!}{x!(y-x)!} \quad for\ 0 \leq x \leq y \tag{11}$$

Intuitivamente, as equações 9 e 10 implicam que possa ser construída uma função de *order-preserving* f de $[M]$ para $[N]$ como uma experiência envolvendo N bolas, das quais M são pretas. Retirando as bolas do saco de forma aleatória e sem reposição, se a $y$ésima bola retirada for preta, então o ponto menos *unmapped* do domínio é mapeado por y em f. É evidente que esta experiência é ineficiente para ser realizada diretamente. No entanto, foi usada a distribuição hipergeométrica para projetar procedimentos que fazem *lazy sample* (função que está na base das funções de cifragem e de decifragem da informação) de forma eficiente e recursiva a uma função de *order-preserving* e o seu inverso.

Em 1985, foi publicado um algoritmo eficiente capaz de gerar valores aleatórios de acordo com a distribuição hipergeométrica (Kachitvichyanukul and Schmeiser, 1985). Este algoritmo é caracterizado por ser exato, isto é, não é uma aproximação por uma distribuição relacionada. É importante referir que este algoritmo é determinístico, ou seja, tem sempre o mesmo *output* para o mesmo *input*. O *output* deste algoritmo corresponde ao número de bolas pretas no nosso exemplo, seguindo uma distribuição hipergeométrica.

No contexto da criptografia, podemos afirmar que o conjunto de todas as bolas corresponde ao espaço de criptogramas e o espaço de mensagens é representado pelo número de bolas pretas.

TapeGen

De forma a ser fornecida toda a aleatoriedade para o algoritmo de HyperGeometric Distribuition (HGD), foi criada a função *TapeGen*, que tanto fornece *input* para a função de HGD, como influencia o *output* das funções de cifragem e de decifragem. A função *TapeGen* também é variável em termos de tamanho de *input* e *output*. Foi proposto uma Length Flexible PseudoRandom Function (LF-PRF) que usa uma Variable-Input-Length Pseudo-Random Function (VIL-PRF), F e um Variable-Output-Length PseudoRandom function Generator (VOL-PRG), G. Foi provado, também, que se F for uma VIL-PRF e se G for um VOL-PRG, então TapeGen é uma LF-PRF. A seguinte equação define o TapeGen:

$$TapeGen(1^l, K, x) = G(1^l, F(K, x)) \tag{12}$$

Nesta equação, o 1^l é o *generator output length*, K é uma chave escolhida uniformemente e aleatoriamente a partir do espaço de chaves e x é o *input*. Os autores recomendam que F seja instanciado com Cipher-based Message Authentication Code (CMAC) e que G seja instanciado com AES128 em *counter-mode*, por exemplo. Na equação seguinte podemos observar uma instanciação concreta do TapeGen:

$$TapeGen(K, x) = AES128CTR(CMAC(K, x), T) \tag{13}$$

Em que o T é uma *string* com tamanho l.

Cifragem

Vamos agora observar a fase de cifragem deste modelo. O pseudo-código é apresentado no algoritmo 1. A letra D representa o domínio do texto limpo enquanto que R corresponde ao domínio dos criptogramas. O número que vai ser cifrado é representado pela letra m. Visto que o algoritmo é recursivo, vamos considerar o momento de primeira chamada de função, para explicação do mesmo.

Na linha 1, a cardinalidade dos conjuntos $\mathcal{D}$ e $\mathcal{R}$ é atribuída às variáveis M e N, respetivamente. De seguida, na linha 2 calcula-se o valor mínimo dos conjuntos $\mathcal{D}$ e $\mathcal{R}$, decrementa-se uma unidade e os valores obtidos são guardados nas varáveis r e d, por esta ordem. Feito isto, calcula-se o valor de y somando a variável r com metade do valor de N. Seguindo o exemplo anteriormente dado, vamos assumir que N tem o valor de 100 e que as bolas são numeradas de 1 a 100. Consequentemente, o valor de y será de 50. Como no exemplo anterior o número de bolas pretas era de 20, o M terá este valor e por isso vamos avançar para a linha 9, já que não vai entrar no *if* por ter um valor diferente de 1. Agora, irá ser usada a função *TapeGen* para gerar um *token*, *cc*, que vai ser usado na função HG.

Tal como foi explicado na secção 3.1, a função HG recebe como argumentos o número de elementos do maior conjunto, o número de elementos do conjunto menor e o número de extrações. De forma a aumentar o nível de abstração, as varáveis anteriormente dadas, N e M são substituídas por $\mathcal{D}$ e $\mathcal{R}$.

Será agora calculada a amostra por parte da função HG e será atribuída à variável x. A função será invocada da seguinte forma, para este exemplo: $HG((1,\ldots,100),(1,\ldots,20),50;cc)$. Neste caso o resultado será de 10 com 20% de probabilidade, 9 e 11 com 17.5% de probabilidade e assim sucessivamente seguindo a distribuição hipergeométrica. Para a mesma chave K e o mesmo número m, será gerado o mesmo *token cc*. Como a função HG é de caráter determinística, o x terá sempre o mesmo valor para diferentes execuções e para o mesmo K e m. Isto permite que o processo de cifragem seja determinístico.

O algoritmo fará agora uma comparação entre o m e o x. Se o m for menor ou igual a x, os domínios de texto limpo e criptogramas serão redefinidos para $\{d+1,\ldots,x\}$ e $\{r+1,\ldots,y\}$ e a função de cifragem será chamada novamente para estes novos intervalos. Caso o m seja maior do que x, o $\mathcal{D}$ e o $\mathcal{R}$ serão modificados de forma a que armazenem os intervalos $\{x+1,\ldots,d+M\}$ e $\{y+1,\ldots,r+N\}$. As chamadas recursivas da função de cifragem só irão terminar quando o domínio do espaço de mensagens apenas contiver 1 elemento, que corresponderá ao m.

Antes de retornar o criptograma final, será gerado um *token* de forma aleatória que será usado para escolher um valor dos elementos restantes do conjunto $\mathcal{R}$. Este valor corresponderá ao resultado do processo de cifragem.

Algorithm 1 Encryption HGD - $Enc_K^{HG}(\mathcal{D},\mathcal{R},m)$

$M \leftarrow |\mathcal{D}|; N \leftarrow |\mathcal{R}|$
$d \leftarrow min(\mathcal{D}) - 1; r \leftarrow min(\mathcal{R}) - 1$
$y \leftarrow r + \lceil N/2 \rceil$
if $|\mathcal{D}| = 1$ **then**
    $cc \xleftarrow{\$} TapeGen(K, 1^{l_R}, (\mathcal{D},\mathcal{R},1||m))$
    $cc \xleftarrow{cc} \mathcal{R}$
    return c
end if
$cc \xleftarrow{\$} TapeGen(K, 1^{l_1}, (\mathcal{D},\mathcal{R},0||y))$
$x \xleftarrow{\$} d + HG(MN, y-r; cc)$
if $m \leq x$ **then**
    $\mathcal{D} \leftarrow \{d+1,\ldots,x\}$
    $\mathcal{R} \leftarrow \{r+1,\ldots,y\}$
else
    $\mathcal{D} \leftarrow \{x+1,\ldots,d+M\}$
    $\mathcal{R} \leftarrow \{y+1,\ldots,r+N\}$
end if
return $Enc_K^{HG}(\mathcal{D},\mathcal{R},m)$

Decifragem

O processo de decifragem é idêntico ao seu processo inverso apresentado anteriormente, principalmente no facto de também ser processada uma pesquisa binária no domínio, de maneira a encontrar a mensagem em texto limpo correspondente ao criptograma. O pseudo-código encontra-se no algoritmo 2.

Na linha 12, e comparando com a linha 10 do processo de cifragem, o criptograma c é comparado com y para selecionar a direção da pesquisa, direita ou esquerda. De facto, é intuitivo perceber que os passos da pesquisa sobre o domínio serão os mesmos que os observados no algoritmo de cifragem.

Avançando para a última iteração, na linha 4, se o domínio do conjunto de mensagens apenas contiver um elemento, a decifragem chega ao fim. O único elemento presente neste conjunto corresponde ao m. No entanto, antes deste valor ser retornado, é feita uma última verificação. É atribuído ao w o criptograma gerado a partir do m descoberto e é feita uma comparação com o c inicialmente dado.

Algorithm 2 Decryption HGD

1: $Dec_K^{HG}(\mathcal{D}, \mathcal{R}, c)$
2: $M \leftarrow |\mathcal{D}|; N \leftarrow |\mathcal{R}|$
3: $d \leftarrow min(\mathcal{D}) - 1; r \leftarrow min(\mathcal{R}) - 1$
4: $y \leftarrow r + \lceil N/2 \rceil$
5: **if** $|\mathcal{D}| = 1$ **then** $m \leftarrow min(\mathcal{D})$
6: $\quad cc \xleftarrow{\$} TapeGen(K, 1^{l_R}, (\mathcal{D}, \mathcal{R}, 1||m)$
7: $\quad w \xleftarrow{\$} \mathcal{R}$
8: $\quad$ **if** $w = c$ **then return** m
9: $\quad$ **elsereturn** $\perp$
10: $\quad$ **end if**
11: **end if**
12: $cc \xleftarrow{\$} TapeGen(K, 1^{l_1}, (\mathcal{D}, \mathcal{R}, 0||y))$
13: $x \xleftarrow{\$} d + HG(MN, y - r; cc)$
14: **if** $c \leq y$ **then**
15: $\quad \mathcal{D} \leftarrow \{d + 1, \ldots, x\}$
16: $\quad \mathcal{R} \leftarrow \{r + 1, \ldots, y\}$
17: **else**
18: $\quad \mathcal{D} \leftarrow \{x + 1, \ldots, d + M\}$
19: $\quad \mathcal{R} \leftarrow \{y + 1, \ldots, r + N\}$
20: **end if**
21: **return** $Dec_K^{HG}(\mathcal{D}, \mathcal{R}, c)$

Garantias de segurança

Este esquema não vai ao encontro da segurança ideal. Para esta ser alcançada seria necessário satisfazer as noções *standard* de segurança como IND-CPA. Para além deste algoritmo ser determinístico, também permite fugas de informação acerca de relações de ordem numérica, sobre o texto limpo. Por exemplo, um adversário contra um esquema OPE que executa 2 pares de *queries* com ordem oposta consegue, trivialmente, quebrar IND-CPA, já que os criptogramas têm a mesma ordem que os textos não cifrados. No entanto, apesar de não conseguir alcançar esta noção de segurança, a intenção é conseguir a melhor noção de segurança possível, com uma abordagem ao caso de *deterministic public-key encription* (Bellare et al., 2007), (Boldyreva et al., 2008), (Bellare et al., 2008), *on-line ciphers* (Bellare et al., 2001) e *deterministic authenticated encryption* (Rogaway and Shrimpton, 2007).

Uma abordagem para dar a volta a este problema é tentar enfraquecer a definição de IND-CPA. Para o caso da *deterministic authenticated encryption*, isto foi feito por (Bellare et al., 2004), que formaliza uma noção chamada IND-DCPA. Este conceito foi, consequentemente, aplicado em códigos de autenticação de mensagens (Bellare, 2006). Visto que esta cifragem é caracterizada por ser determinística, a igualdade dos textos limpos será posta em causa. IND-DCPA não permite que o adversário, utilizado na experiência IND-CPA, faça *queries* à sua *left-right-encryption-oracle* na forma de $(x_0^1, x_1^1), \ldots, (x_0^q, x_1^q)$ tal que $x_0^1, \ldots, x_0^q$ serão todos distintos e $x_1^1, \ldots, x_1^q$ também são todos distintos entre si. Esta noção foi generalizada com o termo INDistinguishability under Ordered Chosen-Plaintext Attack (IND-OCPA), questionando acerca das relação em vez de satisfazer as mesma relações de ordem.

Em vez de tentar restringir, ainda mais, o adversário na definição de IND-OCPA, uma outra alternativa passa pela abordagem às funções pseudo-aleatórias (PRFs) ou às permutações (PRPs), exigindo que nenhum adversário consiga distinguir entre o acesso oráculo ao algoritmo de cifragem do esquema e o objeto "ideal"correspondente. Neste caso, este último é uma função de *order-preserving* aleatória com o mesmo domínio e contra-domínio. Desde que as funções de *order-preserving* sejam injetivas, também faz sentido focar na noção de segurança mais forte que adicionalmente permite o acesso ao oráculo por parte do adversário ao algoritmo de decifragem ou à função inversa, respetivamente. Esta noção tem a designação de Pseudorandom Order-Preserving Function under Chosen-Ciphertext Attack (POPF-CCA).

Uma outra propriedade verificada neste esquema é a distância entre valores. Nas comparações de ordem feita entre os criptogramas, para além de ser conhecido qual o maior criptograma, é possível saber se um criptograma é muito ou pouco maior do que o outro. Isto porque o mapeamento é feito num domínio limitado e através da análise dos bits dos criptogramas consegue-se saber a diferença entre os mesmos. Por exemplo, se tivermos os seguintes criptogramas 11001 e 11000, sabemos que os valores estão próximos já que só diferem no

último bit. Para o caso de termos os seguintes criptogramas 11001 e 00100, sabemos que estão distantes já que o bit da esquerda diz-nos que um se encontra na segunda metade do domínio e o outro na primeira metade.

A primeira versão da *CryptDB* usava esta técnica para cifrar a informação cujo objetivo final era executar *range queries*. Na *CryptDB*, este algoritmo foi melhorado usando árvores de procura binária AVL para *batch encryption*, como por exemplo, *loads* da base de dados, tornando, assim, o sistema mais eficiente.

MODULAR ORDER-PRESERVING ENCRYPTION (MOPE)

Este esquema (Boldyreva et al., 2011) é uma extensão ao esquema anteriormente apresentado, que aumenta a segurança adicionando um *offset* modular secreto a cada valor, antes do processo de cifragem. Seja j o *offset* a ser adicionado, e x a mensagem a ser cifrada temos: $MOPE(x) = OPE(x + j)$, sendo o OPE o esquema apresentado em 3.1. Este *offset* é o mesmo para todas as mensagens e o seu valor é gerado aleatoriamente. Desta forma, a localização dos pontos referentes aos dados são escondidos, já que o *offset* altera a mesma. No entanto, não é possível disfarçar a distância entre os pontos.

Este esquema também veio introduzir novas definições de segurança para OPE. A primeira é mencionada como *Window One-Wayness*. Esta adaptação é mais forte e mais genérica no que toca à definição padrão de *one-wayness* [1]. Para $1 \leq r \leq M$ e $z \geq 1$, é dado ao adversário um conjunto de z criptogramas de mensagens (uniformemente) aleatórias e é pedido ao adversário que adivinhe o intervalo de tamanho r em que está inserida a mensagem subjacente. Caso $r = 1$, a definição vai ao encontro da definição já existente de *one-wayness* (para vários criptogramas). A vantagem do adversário contra o esquema de cifragem simétrico $\mathcal{SE}_{[M],[N]}$ é:

$$Adv^{r,z-wow}_{[M],[N]}(A) = Pr[Exp^{r,z-wow}_{\mathcal{SE}_{[M],[N]}}(A) = 1] \tag{14}$$

em que o ensaio $Exp^{r,z-wow}_{\mathcal{SE}_{[M],[N]}}(A) = 1$ é definido da seguinte forma:

$Exp^{r,z-wow}_{\mathcal{SE}_{[M],[N]}}(A)$
$K \xleftarrow{\$} \mathcal{K}; m \xleftarrow{\$} Comb^{[M]}_z; c \leftarrow \mathcal{E}nc(K, m)$
$(m_L, m_R) \xleftarrow{\$} A(c)$
Return 1 se $(m_R - m_L) \mod M + 1 \leq r$ e se existir $m \in \mathbf{m}$ tal que qualquer $m \in [m_L, m_R]$ ou $(m_L > m_R$ e $m \in [m_L, M] \cup [1, m_R])$
Return 0 noutro caso

1 Uma função *one-wayness* é caracterizada por ser facilmente computada e dificilmente revertida

A outra definição é conhecida como *Window Distance One-Wayness* e inside no facto do OPE expor a distância entre as mensagens. Para se perceber até que ponto esta exposição é feita, o adversário tenta adivinhar o intervalo de tamanho r que corresponde à distância entre dois das z mensagens aleatórias, em que $1 \leq r \leq M$ e $z \geq 2$. Se $r = 1$ o adversário terá de adivinhar o exato valor entre 2 dos z criptogramas. A vantagem do adversário contra o esquema de cifragem simétrico $\mathcal{SE}_{[M],[N]}$ é:

$$Adv^{r,z-wdow}_{[M],[N]}(A) = Pr[Exp^{r,z-wdow}_{\mathcal{SE}_{[M],[N]}}(A) = 1] \tag{15}$$

em que o ensaio $Exp^{r,z-wdow}_{\mathcal{SE}_{[M],[N]}}(A) = 1$ é definido da seguinte forma:

$Exp^{r,z-wdow}_{\mathcal{SE}_{[M],[N]}}(A)$
$K \xleftarrow{\$} \mathcal{K}; m \xleftarrow{\$} Comb^{[M]}_z; c \leftarrow \mathcal{E}nc(K,m)$
$(d_1, d_2) \xleftarrow{\$} A(c)$
Return1 se $(d_2 - d_1) + 1 \leq r$ e se existir $m_i, m_j \in \mathbf{m}$ distintos com $m_j - m_i \mod M \in [d_1, d_2]$
Return 0 noutro caso

MUTABLE ORDER-PRESERVING ENCODING (MOPE)

Com o objectivo de combater as falhas verificadas em propostas anteriores, foi criado o primeiro esquema de OPE com segurança ideal. A prioridade desta plataforma é fazer com que os criptogramas não revelem mais nenhuma informação para além da ordem dos valores do texto limpo. Foi o primeiro esquema que conseguiu alcançar o cenário ideal, no entanto, é afetado em termos de eficiência.

Esta tecnologia consegue alcançar um cenário ideal e rigoroso em termos de garantias de segurança. Estas garantias são definidas como IND-OCPA (Bellare et al., 2007), e asseguram que os adversários não conseguem mais nenhuma informação acerca do criptograma para além da ordem dos valores do texto limpo, que é o mínimo exigido para o OPE.

Este esquema é designado por mOPE devido à "mutabilidade"dos criptogramas verificada e é utilizada a palavra *enconding* em vez de *encryption* para enfatizar o afastamento perante o modelo *standard* de cifragem.

Intuitivamente, mOPE funciona sobre uma árvore balanceada de procura contendo todos os valores cifrados pela aplicação. A codificação do valor em termos de *order-preserving* é o *path* a partir da raiz até ao valor nessa árvore de procura. Portanto, se x é menor do que y, o *path* para x estará à esquerda do *path* para y. Este *path* é representado por uma codificação binária em que o valor codificado do *path* aumenta da esquerda para a direita na árvore. A árvore de procura encontra-se no mesmo servidor não-confiável que

armazena os dados cifrados, e os clientes de confiança cifram os valores inserindo-os na árvore usando um protocolo interativo. O tamanho do *path* corresponde á profundidade da árvore. De forma a prevenir que este *path* não tome proporções excessivamente longas, mOPE faz rebalanceamento da árvore. Isto faz com que a localização dos criptogramas seja atualizada já que o *path* pode ser modificado com o rebalanceamento, no entanto é garantido que apenas um número pequeno de valores já cifrados atualiza os criptogramas para cada novo valor codificado.

mOPE está relacionado com os esquemas criptográficos capaz de executar *range queries* sobre dados cifrados, no entanto existe uma diferença quando comparado com estes esquemas. O mOPE não preserva a ordem dos valores em texto limpo no criptograma. Em vez disso, este esquema cifra, separadamente, os valores referentes à informação e os valores alusivos às *queries* e fornece um algoritmo capaz de fornecer a ordem entre o valor da *query* e o valor da *informação*, por cada pedido. Este algoritmo nunca fornece informação acerca da comparação entre duas *queries* ou entre dois valores. Idealmente, não fornece qualquer tipo de outra informação. O facto dos criptogramas não preservarem a ordem por eles próprios faz com que esquemas como este não possam ser usados sobre *software* com o objetivo de usar *range queries* sobre dados cifrados (OPE) sem uma adaptação prévia.

Modelo

Este esquema assenta num modelo cliente servidor em que o cliente é de confiança, já que é o dono da informação, ao contrário do servidor que não é considerado confiável. Para que haja segurança e computação ao mesmo tempo, apenas uma propriedade pode ser revelada, a ordem real dos valores. Esta garantia tem de prevalecer nos dois modelos de segurança, o modelo ativo e o passivo. Como exemplo de um cenário ideal de OPE, podemos considerar a cifragem dos valores 56, 20, 30, 5 e 60. Uma forma perfeita de codificar estes valores era cifrá-los de maneira a que os criptogramas gerados fossem 4, 2, 3, 1 e 5, respetivamente. Desta forma era garantido que a ordem dos criptogramas prevalecia e que nada mais do que isso era revelado. No entanto, esta forma de cifragem não revela preocupação acerca de futuras cifragens de valores, já que se o próximo valor a ser cifrado for, por exemplo, o número 25, não consegue ter um criptograma que preserve a sua ordem, neste exemplo.

De forma a contornar este problema, os autores desta tecnologia providenciam ao cliente uma leitura dos valores anteriormente cifrados de forma a levar as comparações para o lado de cliente, revelando ainda menos informação ao servidor acerca dos valores reais. É garantido que o número destas interações não ultrapasse o logaritmo do número total de codificações.

Arquitetura do sistema

A arquitetura base desta plataforma é constituída por *B-tree's*. A escolha recaiu para esta estrutura, já que com *B-tree's* qualquer operação feita nas árvores (*insert*, *delete* ou *lookup*) tem um custo logarítmico para o pior caso. Para além disso, são permitidas atualizações de criptogramas de forma eficiente com base em resumos de transformações concisas e é facilitada a verificação caso um servidor malicioso tenha produzido respostas corretas.

No entanto, a explicação do funcionamento deste esquema foi feita, pelos autores, utilizando árvores binárias de procura de modo a simplificar a exposição. Estas árvores estão organizadas da seguinte forma: cada nodo contém à sua esquerda os nodos com valores inferiores e à sua direita nodos com valores superiores. Estes valores referidos correspondem ao valor do número em texto limpo. Todas as técnicas utilizadas extendem-se para árvores binárias de modo simples.

Cada nodo da árvore possui um criptograma cifrado de forma determinística. A organização dos nodos na árvore é feita consoante o valor dos criptogramas em texto limpo, isto é, a posição do nodo na árvore é definida através do valor do número em texto limpo. Conforme podemos observar na imagem 8, os nodos à esquerda são sempre menores do os nodos à direita. Na imagem temos os valores em texto limpo, mas estes não estão presentes na árvore no contexto real, apenas servem para facilitar a compreensão do processo da árvore.

De forma a evitar a decifragem dos números do lado do servidor e a fazer inserções corretas é feita uma interação com o cliente a cada inserção. Imaginemos que era pretendido inserir-se o número 55 na árvore da imagem 8. No primeiro passo o cliente pede o criptograma presente no nodo da raiz da árvore. É retornado o valor "x93d12a"e o cliente decifra este criptograma obtendo o valor 32. Feita a comparação entre o 32 e o 55 é pedido ao servidor o filho direito do nodo em que está presente o 32. É retornado o criptograma "x27716c"que dá origem ao número 69 em texto limpo. Como 55 é menor que 69, é pedido ao servidor o nodo filho que se encontra à esquerda. Como não existe qualquer nodo à esquerda, o servidor informa o cliente do sucedido. Desta forma, o cliente consegue saber o *path* onde tem de inserir o número 55. Cifra-o de forma determinística e ordena a inserção na posição correta. Nesta interação o servidor apenas tem informação acerca do resultado das comparações feitas entre o valor a inserir e os criptogramas presentes na árvore, não conseguindo qualquer tipo de outra informação acerca do valor a ser inserido ou sobre os criptogramas presentes nas árvores.

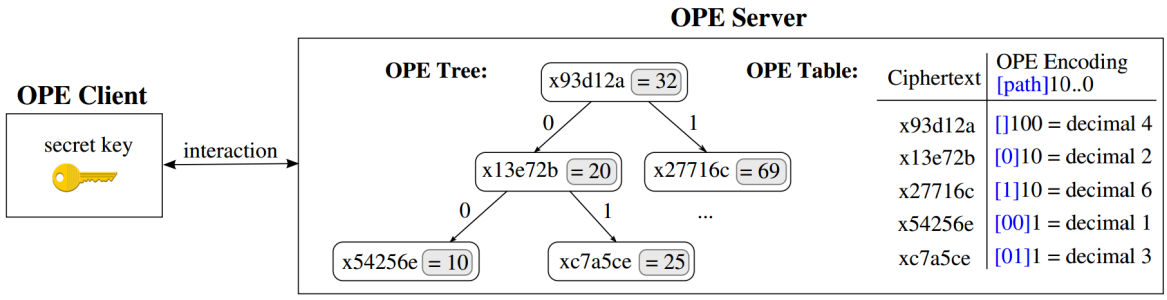


Figura 8: Codificação das árvores na *CryptDB* (Popa et al., 2013)

Codificação binária

A codificação binária é feita através do caminho feito desde a raiz da árvore até ao nodo em questão. Através deste será possível fazer comparação entre os valores pretendidos da árvore. Para que isto seja possível, é necessário arranjar uma forma de dar valores ao caminho. Assim, cada vez que o caminho da esquerda é tomado para chegar ao nodo, dá-se o valor de 0, por outro lado, se o caminho da direita for o escolhido é dado o valor de 1. Chegando ao nodo é feita uma concatenação de todos os bits resultantes das escolhas feitas ao longo do caminho até ao nodo. Por exemplo, na figura 8, o *path* correspondente ao valor 10 é 00, que convertido para decimal dá origem ao valor 0. Já o 25 tem o caminho correspondente de 01, que em decimal tem o valor de 1. Ora, podemos observar que a ordem é conservada na codificação binária.

Neste modelo podemos observar uma adversidade, os nodos não se encontram todos à mesma profundidade. Isto provoca uma diferença em termos de comprimentos nos caminhos. Por exemplo, a raiz da árvore (32) não possui qualquer valor em termos de caminho, e sendo assim, é impossível fazer-se comparações entre caminhos. De forma a contornar esta adversidade, foi estabelecido um comprimento padrão para todos os caminhos de todos os nodos da árvore (32 ou 64 bits). Para que isto possa ser cumprido por todos os nodos, é acrescentado um sufixo a todos os caminhos:

$$OPEencoding = [path]10...0 \tag{16}$$

É acrescentada uma quantidade necessária de zeros até que o *path* fique com o comprimento estabelecido. Voltando ao exemplo da 8, seja $m = 5$ o comprimento do caminho, nos casos do 10 e do 25, a ordem não é alterada. Para este caso o caminho do 10 fica 00100, que em decimal dá origem ao valor 8. No caso do 25, temos agora o caminho 01100, que corresponde ao valor 24, que é maior do que 8. Para o valor da raiz da árvore temos o caminho 10000, que corresponde ao valor 32, que é efetivamente maior que os 2 anteriores. Com isto fica provado que a ordem é preservada.

Balanceamento da árvore

Para que a profundidade da árvore tenha sempre valores logarítmicos em relação ao número total de valores armazenados, é necessário efetuar balanceamentos ocasionais. Para o caso de se verificar um nodo com demasiados filhos, pode ser feita uma divisão no ramo em que o filho deste nodo ganha um nodo filho (que é o antigo pai). Na figura 9 podemos ver um exemplo simples que ajuda a perceber o balanceamento feito. Se existir, ainda, um desequilíbrio na árvore, este balanceamento propaga-se pelos nodos filhos até ser alcançado o equilíbrio.

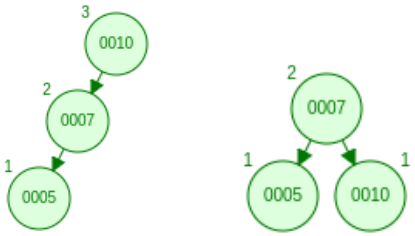


Figura 9: Balanceamento de árvores

Este balanceamento pode provocar uma alteração no caminho de um nodo, e consequentemente, uma alteração na codificação de um valor. Por este motivo é necessário que haja uma atualização nos valores de codificação armazenados no servidor. Existe uma técnica capaz de reduzir o custo desta atualização. Esta tem o nome de *transformation summary* e permite que sejam efetuadas operações de balanceamento concisas em $O(\log n)$ e identifica com precisão o alcance dos valores afetados. Com isto, as atualizações de codificações podem ser efetuadas, de forma eficiente, numa passagem sobre o alcance dos valores afetados.

Por outro lado, com este balanceamento é garantido que o comprimento das codificações nunca atinjam comprimentos muito grandes, já que o comprimento do caminho até um nodo corresponde à profundidade desse nodo e este balanceamento consegue, em alguns casos, reduzir esta profundidade.

Codificação obsoleta

O balanceamento pode fazer com que o valor de codificação fique obsoleto. Por exemplo, um valor pode ficar obsoleto, quando, primeiramente, é pedido o valor de codificação de um determinado nodo e depois disso são efetuadas operações que envolvem *inserts* e balanceamentos da árvore. Desta forma, o caminho para o nodo pode ser alterado e, portanto, a codificação também é alterada. Todas as computações executadas com este valor incorreto podem dar origem a erros.

De forma a prevenir estas situações foi criada uma tabela para mapear os criptogramas e os valores de codificação correspondentes. Um exemplo para uma *OPE Table* pode ser visto

na figura 8. Sempre que há uma inserção ou um balanceamento, esta tabela é atualizada com os novos valores de codificação. Esta atualização apenas é útil para o servidor, já que o cliente não tem acesso nem necessita das codificações, apenas insere e pede criptogramas. Com isto, o servidor é capaz de executar operações OPE sem necessitar de recorrer ao cliente.

Garantias de segurança

Este esquema é IND-OCPA. O mOPE alcança o cenário ideal apresentado em (Boldyreva et al., 2009), ou seja, nenhumas informações acerca do valor real inserido são expostas a não ser as propriedades de ordem. Com o intuito de demonstrar a segurança do sistema, vamos considerar qualquer adversário Adv e duas sequências de valores $v = (v_1, ..., v_n)$ e $w = (w_1, ..., w_n)$. O adversário tem acesso a todas as interações feitas entre o cliente e o servidor. Os criptogramas enviados são cifrados com um esquema de cifragem DET, ou seja, as garantias de segurança deste esquema assumem que as cifragens são computacionalmente indistinguíveis de valores aleatórios. Para o caso da cifragem dos valores de v e w a aprendizagem do adversário pode basear-se em várias propriedades. Para isso, procede-se indutivamente sobre o número de valores a serem cifrados. O caso base dá-se quando nenhum valor foi cifrado ainda e, nesta altura, o adversário não possui qualquer informação acerca dos valores. Após i iterações o adversário obtém a mesma informação para ambos os casos (v e w), e mesmo após $i + 1$ iterações, não há nada de novo em termos de aprendizagem para o adversário.

Focando agora na parte da codificação, existem 2 resultados possíveis. O primeira dá-se quando o valor da codificação já se encontra na tabela. Como v e w têm a mesma relação de ordem (já que $v_i = v_j \iff w_i = w_j$), então o w também já irá ter o valor de codificação na tabela. Como não há interações entre o cliente e o servidor, não existem mais nenhuma informação que o adversário tenha acesso.

No segundo resultado, temos o valor v_i sem valor de codificação na tabela. Neste caso existirão trocas de informação entre servidor e cliente de forma a definir a posição do valor na árvore (optar pela direita ou esquerda, em cada nodo). Estas decisões serão as únicas informações a que o adversário tem acesso.

Em ambos os casos, o adversário obtém a mesma informação para ambos os valores dos dois conjuntos, ou seja, não tem qualquer informação que permita fazer uma distinção entre os criptogramas do conjunto v e w.

ORDER REVEALING ENCRYPTION (ORE)

ORE (Boneh et al., 2015) é uma técnica recente, caracterizada por ser uma generalização do original OPE. No contexto de OPE lidamos com criptogramas numéricos que preservam a ordem do valor do texto limpo para serem feitas comparações diretas com os criptogramas respondendo às *range queries* feitas. No entanto, com esta técnica, existem algumas mudanças relativamente ao OPE. Com o ORE, os criptogramas não têm uma forma específica e a comparação é feita através de uma função que recebe 2 criptogramas e retorna o resultado da comparação feita segundo os valores em texto limpo, mas sem os decifrar.

Com este esquema é alcançada segurança contra ataques IND-OCPA.

O esquema ORE é um tuplo de algoritmos $\Pi = (ORE.Setup, ORE.Encrypt, ORE.Compare)$ que são definidos no domínio $\mathcal{D}$ com as seguintes propriedades:

- $ORE.Setup(1^\lambda) \rightarrow sk$ a chave sk é gerada através desta função que recebe um parâmetro de segurança, λ.
- $ORE.Encrypt(sk, m) \rightarrow ct$ Dada um chave sk e uma mensagem m, a função $ORE.Encrypt$ gera um criptograma ct.
- $ORE.Compare(ct_1, ct_2) \rightarrow b$ Este algoritmo retorna o resultado da comparação feita entre os dois criptogramas, ct_1 e ct_2. O *output* é o seguinte: $b \in 0, 1$.

Nesta técnica não observamos nenhuma função de decifragem. Uma solução capaz de resolver esta omissão, reside na junção de de um criptograma da mensagem seguro sobre CPA, à cifragem em ORE. Assim pode-se proceder à decifragem direta do criptograma anexado obtendo a mensagem pretendida.

Esta técnica difere do OPE na forma dos criptogramas. Enquanto que a construção de um criptograma em OPE depende da chave, da mensagem e da aleatoriedade dada, em ORE o bit anterior também interfere na cifragem da mensagem. Como exemplo, temos o caso de dois criptogramas cifrados, $c_1 = 11001100$ e $c_2 = 10001100$. Em OPE através da localização destes dois criptogramas no espaço dos criptogramas podemos ter uma pequena noção do valor dos criptogramas e da distância dos mesmo. Em ORE apenas temos informação acerca do valor em texto limpo através dos dois primeiros bits. O resto dos bits são cifrados com a influência do *bit* anterior fazendo com que o seu real valor não corresponda diretamente ao valor do criptograma. Consequência disto, é a necessidade da existência de uma função própria capaz de executar a comparação entre os dois criptogramas.

CONCLUSÃO

Neste capítulo foram apresentadas técnicas que permitem funcionalidade sobre dados cifrados, mais propriamente, técnicas que permitem a execução de *range queries* sem decifrar

a informação. A técnica OPE, foi a primeira a tratar do problema de forma séria e com segurança razoável. Esta técnica é caracterizada por permitir funcionalidade direta sem que hajam alterações do lado da base de dados, já que os criptogramas preservam a ordem das mensagens em texto limpo. Os mesmo autores lançaram a glsmope11, que é uma adaptação do OPE. Como foram feitas pequenas alterações apenas, esta técnica não foi estudada com grande profundidade. Em 2013, surgiu uma técnica diferente do OPE, o mOPE. Neste esquema, já foram feitas grandes transformações trazendo novas implicações. Mudanças no esquema de base de dados e existência de comunicação constante são exigências para que esta técnica funcione de forma correta. Por fim foi estudada a técnica ORE. Uma técnica recente ainda, que demorou a estabilizar e por isso, também foi feito um estudo superficial acerca da mesma.

4

VALIDAÇÃO EXPERIMENTAL

Neste capítulo será demonstrado o trabalho desenvolvido em termos práticos. As técnicas MOPE e ORE apenas possuem uma abordagem teórica, pelo que não serão demonstrados resultados acerca destes esquemas. A justificação para esta omissão reside na semelhança de MOPE e OPE, pelo que os resultados não iriam diferenciar de forma a serem tiradas conclusões. No caso do ORE, o facto deste esquema ser recente, tirou a possibilidade de serem feitos testes com estas técnicas. No entanto, serão apresentados os resultados acerca dos testes feitos às técnicas OPE e mOPE.

Estas bibliotecas foram integradas com um sistema *key-value storage*, o *HBase*. Como este motor de base de dados se encontra na linguagem *JAVA*, a comunicação entre as bibliotecas e o sistema de dados é feita através do JNI, visto que estas bibliotecas se encontram implementadas na linguagem *C* por razões de eficiência. Esta integração entre as bibliotecas e o sistema de dados consiste na definição de interfaces JNI, sendo que esta *framework* provoca um pequeno decréscimo, em termos de performance, no sistema. Por isso, será avaliado o custo computacional provocado pelo JNI. De forma a conseguir-se quantificar este custo, foram realizados testes com uma interface simples em *JAVA* que, através do JNI, recorre às bibliotecas para fazer cifragens e decifragens de mensagens. Para se comparar os resultados obtidos com esta interface, foram realizados testes, sem qualquer componente em *JAVA*, ou seja, foi criada uma uma outra interface utilizando a mesma linguagem das bibliotecas. Desta forma, consegue-se avaliar o *overhead* provocado pelo JNI.

Ambas as técnicas, OPE e mOPE, foram submetidas a testes idênticos, na mesma máquina. As especificações e o ambiente de testes podem ser observados na tabela 2.

Processador	Intel® Core™ i7-4720HQ Quad Core 6M Cache 2.6 - 3.6 GHz
Memória RAM	8GB DDR3 1600MHz
Sistema Operativo	Ubuntu 16.04.2 LTS
Compilador C/C++	5.4.0
Compilador JAVA	1.8.0_131
Versão openssl	1.0.2g
Versão NTL	9.6.4

Tabela 2: Especificações da máquina de teste

De forma a avaliar a perda de performance, em relação a esquemas que não permitem funcionalidade, foram realizados testes com *AES-CBC* para que se consiga ter uma base para comparação. Para a contagem do tempo foi utilizada a biblioteca *time.h* em *C/C++* e em *JAVA* recorreu-se à função *currentTimeMillis()* do *System* para fazer a contagem do tempo.

Com o intuito de tornar os resultados mais realistas, foram utilizados diferentes tamanhos de mensagens e criptogramas, e foram feitas várias execuções, sendo que o valor da mediana foi a referência usada nos resultados demonstrados. As mensagens usadas no processo de cifragem não foram geradas aleatoriamente, já que assim é possível ter um maior controlo sobre estas e gerar as mesmas mensagens para os vários esquemas. A mensagem usada para o processo de cifragem foi incrementada em cada iteração para assim não serem repetidas as cifragens da mesma mensagem e cada execução era constituída por 1000 iterações (1000 cifragens e 1000 decifragens).

IMPLEMENTAÇÕES OPEN-SOURCE

Standard Encryption (AES-CBC)

O AES, apresentado na secção 2.2.1, é uma cifra por blocos, considerada *standard* pela agência norte-americana NIST. Por este motivo, esta técnica foi usada como esquema base para a comparação dos resultados obtidos com técnicas que permitem funcionalidade sobre dados cifrados. O modo de operação da cifra por blocos é o Cipher Block Chaining (CBC). Neste modo, o primeiro bloco é combinado com um vetor de inicialização aleatório. Os blocos seguintes estão sempre dependentes do criptograma gerado no bloco anterior a si.

Para a cifragem e decifragem dos dados foi usada a biblioteca *openssl*, tornando, assim possível a utilização da cifra *AES-CBC*. Esta biblioteca não é intuitiva, pelo que houve um esforço extra para que esta funcionasse neste contexto. Houve a preocupação de ser feita uma interface genérica e coesa com as restantes técnicas testadas. A utilização da interface do *openssl* exige o cumprimento de alguns passos. Primeiramente começou-se por criar um contexto criptográfico e gerar um vetor de inicialização. Feito isto, procedeu-se à inicialização, em que é indicada a cifra a ser utilizada pela interface e é dado o contexto criado anteriormente e o vetor de inicialização. Concluída a inicialização da operação (de cifragem ou decifragem, dependendo da função), a interface está pronta a cifrar/decifrar o texto limpo/criptograma pretendido. Por fim, é dado o passo final que corresponde à libertação da memória alocada e verificação do sucesso da operação.

OPE

Esta técnica consiste no mapeamento de uma mensagem no espaço de criptogramas, de modo a preservar a ordem original. Para a implementação desta técnica, foi isolado o código presente na *CryptDB*, onde na primeira versão da mesma, este esquema era utilizado para tratar das *range queries* sobre dados cifrados. Para além do isolamento do código de cifragem, foi também isolado o algoritmo *sampling* do *HGD*, presente no OPE, e comparado com o *paper* original (Kachitvichyanukul and Schmeiser, 1985). Para além de facilitar a compreensão do código, os testes feitos com o *HGD* permitiram também um maior conhecimento acerca do mapeamento feito e do comportamento da função hipergeométrica. Para se conseguir lidar com grades números de forma precisa, esta técnica recorre à biblioteca NTL [1]. O facto do código presente da *CryptDB* não ser modular, dificultou a tarefa de isolamento pois foi exigido um trabalho extra na compreensão e modulação do código. A adaptação feita no âmbito desta dissertação, fez com que o código desenvolvido possa ser adaptado para outras plataformas, de forma simples. Esta adaptação consistiu sobretudo na resolução de dependências e na transformação para um sistema mais genérico.

```
ZZ
OPE::encrypt(const ZZ &ptext) {
    ope_domain_range dr =
        search([&ptext](const ZZ &d, const ZZ &) { return ptext < d; });

    auto v = sha256::hash(StringFromZZ(ptext));
    v.resize(16);

    blockrng<AES> aesrand(aesk);
    aesrand.set_ctr(v);

    ZZ nrange = dr.r_hi - dr.r_lo + 1;
    return dr.r_lo + aesrand.rand_zz_mod(nrange);
}
```

No código acima apresentado, temos a função responsável pelo processo de cifragem. É feita uma pesquisa com o objetivo de mapear a mensagem num domínio de criptogramas. Feita esta pesquisa, através da função *search*, é adicionado um fator aleatório de forma a tornar mais segura a cifragem. É garantido que o valor aleatório somado não provoque um resultado final fora do intervalo em que o criptograma foi mapeado.

Para calcular o valor da função hipergeométrica, referente ao intervalo dado, é invocada a função *HGD*, cuja declaração pode ser observada de seguida:

1 http://www.shoup.net/ntl/

```
HGD(rgap, ndomain, nrange-ndomain, prng);
```

```
if (go_low(d_lo + dgap, r_lo + rgap))
    return lazy_sample(d_lo, d_lo + dgap - 1, r_lo, r_lo + rgap - 1, go_low,
        prng);
else
    return lazy_sample(d_lo + dgap, d_hi, r_lo + rgap, r_hi, go_low, prng);
```

O mapeamento é feito de forma recursiva (código em cima), em que os domínios se dividem a meio até atingir o caso de paragem (código em baixo). Dependendo do valor de domínio e de *range* é feita a escolha da direita ou da esquerda do domínio do criptograma, para serem usados na próxima iteração, invocando novamente a mesma função com novos valores.

```
if (ndomain == 1)
    return ope_domain_range(d_lo, r_lo, r_hi);
```

Quando chegamos ao caso de paragem, é devolvido um objeto com todos os valores necessários para gerar o criptograma final.

Para o funcionamento correto do código acima descrito, é necessário inicializar o objecto, dando-lhe as informação necessárias para esta tarefa (chave e tamanhos de mensagem e criptogramas). De seguida, as mensagens podem ser geradas na forma de *string*, no entanto, é necessário converter para o tipo ZZ, que é característico da biblioteca NTL. Feito isto, pode-se proceder à cifragem e decifragem de mensagens. Estes passos estão representados na seguinte porção de código:

```
int main(int argc, char** argv) {
  OPE * ope = new OPE(string(key, AES_KEY_BYTES), ptextsize, ctextsize);
  ZZ dataZZ = ZZFromString(data);
  ZZ enc = ope->encrypt(dataZZ);
  ZZ dec = ope->decrypt(enc);
}
```

mOPE

Como no OPE, o código para esta técnica foi retirado da plataforma *CryptDB*. Foram aplicadas modificações de forma a funcionar de forma independente. O paradigma relativo a

este esquema indica um ambiente cliente-servidor, sendo que o servidor mantém o estado e, armazena os criptogramas e a árvore relativa à ordem das mensagens. O cliente limita-se a fazer pedidos e a responder ao servidor com as indicações acerca do caminho para percorrer a árvore. Por este motivo, foi simulado, de forma prática, um ambiente cliente-servidor para ir ao encontro do paradigma. Esta implementação foi feita através de um sistema cliente-servidor, em que a comunicação é feita através de *sockets* em *C*.

A função de cifragem (código abaixo) baseia-se numa procura na árvore presente no servidor. Caso, não seja encontrado o valor pretendido na árvore, é feita uma inserção deste mesmo valor na árvore e é feita uma nova procura do mesmo, para assim ser retornado o criptograma. Também se pode observar a forma de como é feita a travessia na árvore. O lado do cliente é o responsável pelas comparações feitas entre os valores presentes na árvore e valor que se pretende cifrar. Todos os *reads* e *writes* são referentes às comunicações entre cliente e servidor.

```
(Cliente)
uint64_t encrypt(V pt) const {
    uint64_t v = 0;
    uint64_t nbits = 0;
    try
    {
        for (;;)
        {
            msg.action = 'l'; msg.v = v; msg.nbits = nbits;
      r = write(addr, &msg, sizeof(struct args));
            r = read(addr, &res_lookup, sizeof(uint64_t));

            V xct = res_lookup;
            V xpt = block_decrypt(xct);

            if (pt == xpt)
                break;

            if (pt < xpt)
                v = (v << 1) | 0;
            else
                v = (v << 1) | 1;

            nbits++;
        }
    }
    catch (ope_lookup_failure &)
    {
        msg.v = v; msg.nbits = nbits; msg.ct = block_encrypt(pt); msg.action =
            'i';
        r = write(addr, &msg, sizeof(struct args));
```

```
        r = read(addr, &res_insert, sizeof(uint64_t));
        return encrypt(pt);
    }
    throw_c(nbits <= 63);
    return (v << (64 - nbits)) | (1ULL << (63 - nbits));
}
```

O código abaixo representado refere-se à inicialização e, cifragem e decifragem dos dados, na parte do cliente. A função *encrypt* está representado no código acima.

```
(Cliente)
int main(int argc, char** argv) {
  OpeTrees ot(key);
  ct = ot.encrypt(pt);
  dec = ot.decrypt(ct);
}
```

Em baixo temos o modo de funcionamento do servidor que se encontra à espera de mensagens por parte do cliente para assim proceder à realização das tarefas pedidas pelo servidor. A função *process_msg_server* é responsável por diferenciar os pedidos feitos pelo cliente e executar as tarefas pretendidas.

```
(Servidor)
int main(int argc, char** argv) {
  ope_server<uint64_t> ope_serv;
  connected_socket = connect_client();

  while (1){
      r = read(connected_socket, &msg, sizeof(struct args));
      if (r == 0)
          break;

    process_msg_server(msg, &ope_serv, connected_socket);
    bzero(&msg, sizeof(char));
    }
}
```

INTEGRAÇÃO COM O HBASE

De forma a conseguir integrar as bibliotecas com o *HBase*, foi necessário recorrer à *framework* JNI e à ferramenta *maven*.

O facto das bibliotecas de cifragem e decifragem dos dados estarem escritas na linguagem *C*, exige uma forma de comunicação entre o sistema de dados, escrito em *JAVA* e as bibliotecas. O JNI é uma *framework* que faz a integração entre o código escrito na linguagem *JAVA* com código escrito noutras linguagens, tais como *C/C++*. Esta ferramenta previne situações, como esta, em que as bibliotecas estão escritas noutra linguagem em relação à aplicação original.

Foi, então, usado o JNI para ser desenvolvida uma interface capaz de garantir a comunicação entre o sistema de dados e as bibliotecas. Esta interface é constituída pelas principais funções necessárias para executar tarefas como inicialização de objetos e, cifragem e decifragem de mensagens. Para além da criação da interface, houve também uma adaptação das bibliotecas para que todas as funcionalidades necessárias estejam disponíveis na interface. No início existiram algumas dificuldades no que toca ao tipo de dados usados para a partilha de variáveis, já que os esquemas usados tinham algumas particularidades. No caso do OPE, é usada uma biblioteca para tratar grandes números, o NTL. A biblioteca OPE retorna o resultado da cifragem e decifragem num tipo de dados característico do NTL, no entanto o NTL apenas está disponível para C/C++. Foi, então necessário a conversão das mensagens (texto limpo e criptogramas) para *String* de forma a que possam ser lidas e tratadas do lado do *JAVA*. Para além da conversão para *String*, todas as mensagens enviadas para o lado *JAVA* eram submetidas sobre a forma de *byte array* de forma a criar uma aplicação genérica. Para o caso do mOPE a grande dificuldade residiu em enviar o objeto que contem a árvore para o lado do *JAVA* e na forma de armazenamento deste mesmo objeto. Foi utilizado o conceito de objeto genérico para o armazenamento da árvore do lado do *JAVA*. Para o envio é utilizado o mesmo conceito, sendo utilizado o tipo *jobject*, característico do JNI, para a comunicação entre as linguagens.

Para além de ser feita a ponte entre as linguagens, foi também necessário ligar estes esquemas com o projeto global. Foi, então, criado um projeto *maven* [2] para cada esquema, de forma a ser possível integrar as bibliotecas com o motor de base de dados. *Maven* é uma ferramenta usada para fazer *build* em projetos de grande escala *JAVA* de forma automática. Este projeto criado disponibiliza as funções essenciais para o funcionamento dos esquemas, possibilitando a invocação das mesmas por parte de outros projetos *maven* integrados. Este sistema de dados (*HBase*) foi fornecido pelo *HASLab* [3]. A junção do *maven* com o *jni* no mesmo projeto foi um processo complexo, já que não é comum juntar estas duas tecnologias. Para a execução de aplicações que usem a *framework jni*, são necessárias

2 https://maven.apache.org/
3 http://haslab.uminho.pt/

algumas particularidades na compilação e execução. Na compilação é necessário a geração de um ficheiro *.so (shared object)* no lado do C, para que o *JAVA* consiga aceder às bibliotecas. No lado *JAVA* é necessário a utilização do *javah* para que seja gerado um ficheiro *header* da linguagem C, necessário para a comunicação entre as duas linguagens.

Por fim, e para garantir o funcionamento em várias plataformas, foi feito um *docker* [4]. Nele foram introduzidos todos os *softwares* necessário para o correto funcionamento da aplicação, bem como a instalação de dependências.

DISCUSSÃO DE RESULTADOS

Overhead JNI

Como é expectável, esta *interface* irá provocar um decaimento em termos de perfomance do sistema. No entanto, como podemos comprovar, esse decaimento é tolerável e não afeta muito a performance sistema.

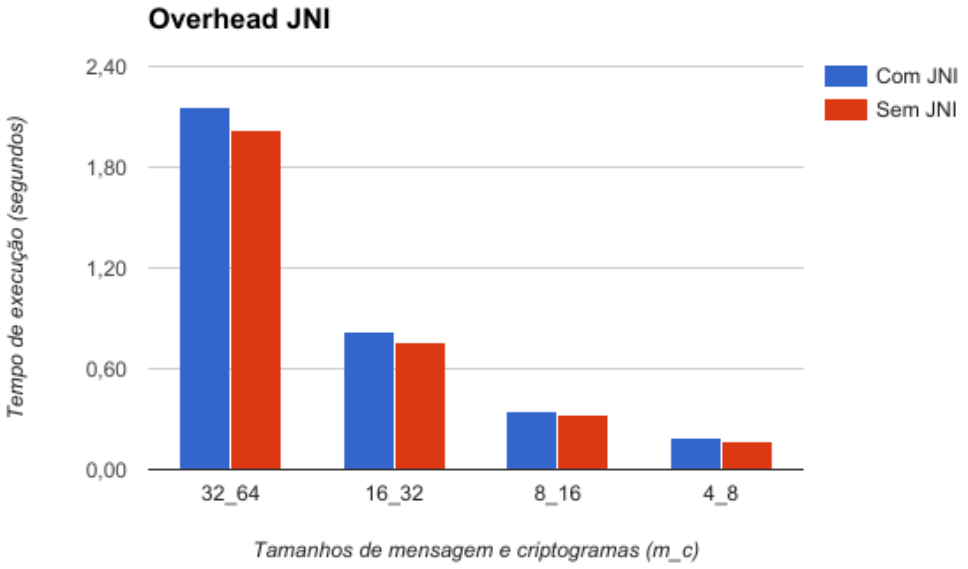


Figura 10: Execução da biblioteca OPE sem o JNI e com o JNI

Este aumento na ordem dos 10% de tempo deve-se a vários fatores:

- Os métodos nativos em *C/C++* não foram compilados para correr na *JVM*. As bibliotecas foram compiladas, primeiramente, para gerar uma biblioteca dinâmica.

4 *Docker* é uma ferramenta capaz de criar uma camada de abstração de um sistema operativo virtual em que são indicados todas as versões do *software* necessário e dependências para o funcionamento da aplicação em questão

- A transferência de dados entre as linguagens são lineares ao tamanho dos mesmos.
- Se for passado um objeto ou se for necessário uma *callback*, o código nativo *C* poderá estar a fazer chamadas à *JVM*.
- As *strings* em *JAVA* estão codificadas. O acesso ou a criação das mesmas pode ser dispendioso.

Comparação de performance

Como as técnicas são diferentes, existirá, também, diferenças nos tempos de execução. É, por isso, fundamental analisar estes dados e perceber o porquê dos mesmos.

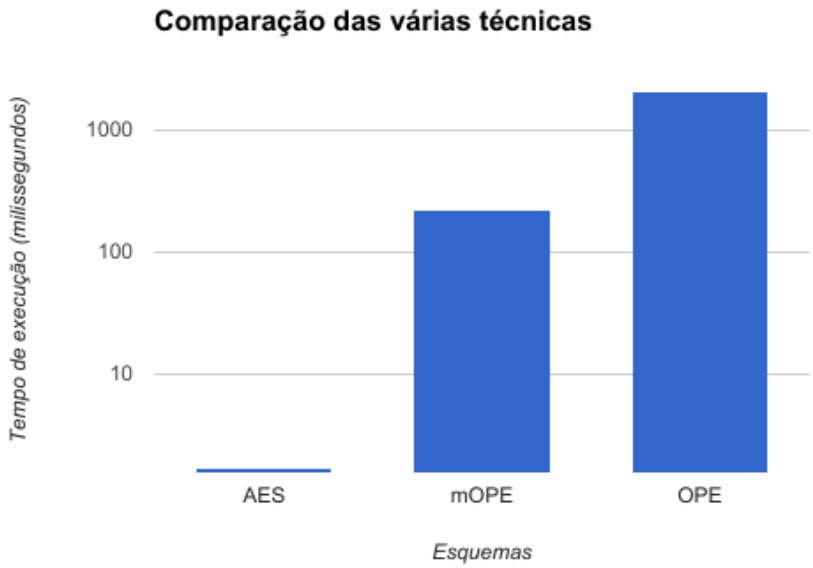


Figura 11: Comparação das diferentes técnicas em termos de tempo de execução

Como era expectável, uma cifra que não permitem funcionalidade sobre dados cifrados obtém melhores resultados do que as restantes, já que a existência de funcionalidade sobre dados cifrados retira eficiência ao sistema.

Quando comparamos as 2 cifras que permitem funcionalidade podemos ver que o mOPE é cerca de 10 vezes mais rápido que o OPE. O esquema mOPE possui várias comunicações entre cliente e servidor para tomar as decisões acerca do caminho a percorrer na árvore, no entanto, nestes resultados não são considerados tempos de latência, que possam existir no contexto real, relativos à comunicação.

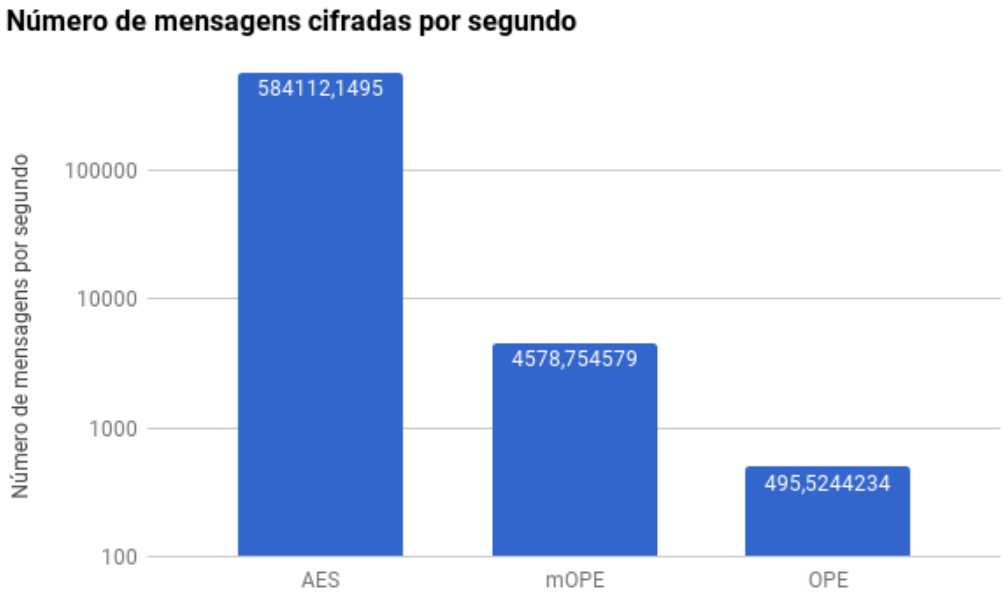


Figura 12: Comparação da capacidade de cifragem por unidade de tempo

Na figura 12, pode ser vista uma outra perspetiva em termos de comparação de performance. Embora, a ordem entre os esquemas seja a mesma, pode ser observado o número de mensagens que um esquema pode conseguir cifrar, e assim, calcular se esse número de mensagens é suficiente para as necessidades do sistema em que o esquema pode ser introduzido.

A performance das duas técnicas foi analisada, no entanto existem outros aspetos, tais como a disponibilidade do servidor para manter estado e a capacidade dos canais de comunicação, que importa considerar na escolha de um esquema para um sistema.

5

CONCLUSÃO

Foram abordadas, teórica e praticamente, várias técnicas pragmáticas capazes de permitir funcionalidade em dados cifrados. Foi feita uma avaliação técnica e uma análise acerca da implementação existente. Depois da adaptação dos esquemas e da criação de um ambiente homogéneo em cada uma das técnicas, foram tiradas conclusões acerca da viabilidade do sistema e do tempo de execução em termos de cifragem e decifragem de dados.

Apesar dos resultados experimentais ditarem que umas técnicas são mais eficientes no processo de cifragem do que outras, a escolha do esquema de cifragem para um sistema não se pode basear apenas neste fator. Como se observou no capítulo anterior, a técnica mOPE obteve melhores resultados em termos de performance, no entanto esta técnica tem outras implicações. Uma delas reside no facto de ter bastante comunicação quando comparada com OPE (apenas solicita a cifragem de uma mensagem e recebe o criptograma). Outra particularidade que caracteriza o esquema mOPE é o facto de exigir a manutenção de um estado. Para além disso, o balanceamento das árvores também é outro procedimento não avaliado, visto que é impossível prever os dados que vão ser inseridos e, consequentemente, o número de balanceamentos que esses dados podem originar. Segundo os autores (Popa et al., 2013), em média, são actualizados entre 2 e 4 nodos por cada cifragem, provocando um decréscimo do *throughput* na ordem dos 15%. Por isso, a escolha de uma técnica para um sistema está totalmente dependente do ambiente em que este irá estar e dos recursos dados pelo mesmo.

A existência de funcionalidade sobre dados cifrados é algo viável com os recursos disponíveis nos dias de hoje. No entanto, esta área ainda é vista como dispendiosa para o cliente, e no ponto de vista deste, a segurança não é 100% garantida. Esta insegurança deve-se ao facto de este tema, ainda, ser bastante recente, no que toca a implementações. Por isso, é necessário um amadurecimento deste tipo de esquemas para ser alcançada estabilidade e assim, ser conquistada a confiança dos utilizadores. Só assim pode ser padronizado e consequentemente aumentar a utilização destas técnicas. Uma outra contrariedade consiste no *trade-off* existente entre performance e funcionalidade, isto é, o cliente tem de optar por deixar de parte alguma performance para obter funcionalidade nos dados. Exemplo disso são os tempos verificados com o esquema AES que não permite funcionalidade, mas é eficiente,

tendo registado os melhores tempos de execução. Os tempos verificados nas cifras que permitem funcionalidade acabam por perder alguma performance, mas em contra partida permitem funcionalidade sobre dados cifrados.

O trabalho desenvolvido conseguiu provar que a funcionalidade sobre dados cifrados, para além ser uma prática viável, é algo que pode trazer bastantes vantagens ao cliente, tais como implementar segurança sem modificar os sistemas de dados existentes nos dias de hoje. A criação de um ambiente genérico para a cifragem dos dados, no âmbito desta dissertação, também facilita a implementação dos esquemas em qualquer sistema de dados, já que problemas como a diferença de linguagens entre o sistema de dados e as bibliotecas ou a integração com o sistema de dados podem ser ultrapassados de forma eficaz.

TRABALHO FUTURO

O trabalho futuro passa pelo estudo e implementação de novas técnicas funcionais, como o ORE. Desta forma, podem ser encontradas técnicas capazes de ter melhores resultados em termos de performance. Este estudo é, também, importante pois pode permitir a descoberta de novas formas de permitir funcionalidade sobre dados cifrados.

Outro ponto interessante passa pela melhoria do sistema já existente em termos de performance. Para isso, testes com novos tipos de dados ou melhoramentos dos algoritmos de cifragem são passos importantes para a descoberta de algumas lacunas a nível de eficiência.

A aplicação destas técnicas noutros sistemas de dados, para além do *HBase*, seria também um tópico pertinente no contexto desta dissertação. Para além de ser provada a generalização do código desenvolvido, poderia ser descoberto um sistema capaz de executar as *queries* de forma mais rápida.

Todos os pontos enumerados podem, de alguma forma, contribuir para o desenvolvimento do tema principal desta dissertação, a computação sobre dados cifrados.

BIBLIOGRAFIA

Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574. ACM, 2004.

Ross Anderson. *Security engineering*. John Wiley & Sons, 2008.

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, apr 2010. ISSN 0001-0782. doi: 10.1145/1721654.1721672. URL http://doi.acm.org/10.1145/1721654.1721672.

Friedrich L Bauer. *Decrypted secrets: methods and maxims of cryptology*. Springer Science & Business Media, 2013.

Mihir Bellare. New proofs for nmac and hmac: Security without collision-resistance. In *Annual International Cryptology Conference*, pages 602–619. Springer, 2006.

Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 531–545. Springer, 2000.

Mihir Bellare, Alexandra Boldyreva, Lars Knudsen, and Chanathip Namprempre. Online ciphers and the hash-cbc construction. In *Annual International Cryptology Conference*, pages 292–309. Springer, 2001.

Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Breaking and provably repairing the ssh authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):206–241, 2004.

Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *Annual International Cryptology Conference*, pages 535–552. Springer, 2007.

Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *Annual International Cryptology Conference*, pages 360–378. Springer, 2008.

Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs# 1. In *Annual International Cryptology Conference*, pages 1–12. Springer, 1998.

Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Annual International Cryptology Conference*, pages 335–359. Springer, 2008.

Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'neill. Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 224–241. Springer, 2009.

Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Annual Cryptology Conference*, pages 578–595. Springer, 2011.

Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 563–594. Springer, 2015.

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2): 4, 2008.

Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael. 1999.

Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

Zekeriya Erkin, Alessandro Piva, Stefan Katzenbeisser, Reginald L Lagendijk, Jamshid Shokrollahi, Gregory Neven, and Mauro Barni. Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing. *EURASIP Journal on Information Security*, 2007:17, 2007.

Lars George. *HBase: the definitive guide*. "O'Reilly Media, Inc.", 2011.

Joao Girao, Dirk Westhoff, and Markus Schneider. Cda: Concealed data aggregation for reverse multicast traffic in wireless sensor networks. In *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, volume 5, pages 3044–3049. IEEE, 2005.

Tim Greene. Biggest data breaches of 2015. *Network*, 10:14, 2015.

Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE, 2011.

Yury Izrailevsky. Nosql at netflix. *The Netflix Tech Blog*, 2011. URL http://techblog.netflix.com/2011/01/nosql-at-netflix.html.

Voratas Kachitvichyanukul and Burce Schmeiser. Computer generation of hypergeometric random variates†. *Journal of Statistical Computation and Simulation*, 22(2):127–145, 1985.

Jonathan Katz and Yehuda Lindell. Introduction to modern cryptography: principles and protocols. In *Cryptography and network security*, chapter 2, pages 47–109. Chapman & Hall/CRC, Boca Raton, 2008.

Manpreet Kaur and Rajbir Singh. Implementing encryption algorithms to enhance data security of cloud in cloud computing. *International Journal of Computer Applications*, 70 (18), 2013.

Neal Leavitt. Will nosql databases live up to their promise? *Computer*, 43(2):12–14, 2010.

Cade Metz. Facebook: Why our 'next-gen' comms ditched mysql. *The Register*, 2010. URL http://www.theregister.co.uk/2010/12/17/facebook_messages_tech/.

Mohamed A Mohamed, Obay G Altrafi, and Mohammed O Ismail. Relational vs. nosql databases: A survey. *International Journal of Computer and Information Technology (ISSN: 2279–0764) Volume*, 2014.

Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 85–100. ACM, 2011.

Raluca Ada Popa, Frank H Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 463–477. IEEE, 2013.

Phillip Rogaway and Thomas Shrimpton. Deterministic authenticated-encryption a provable-security treatment of the key-wrap problem. *Advances in Cryptology — EUROCRYPT '06*, 2007.

Adi Shamir. *How to Share a Secret*, volume 22. ACM, New York, NY, USA, nov 1979. doi: 10.1145/359168.359176. URL http://doi.acm.org/10.1145/359168.359176.

Tom Shrimpton. A characterization of authenticated-encryption as a form of chosen-ciphertext security. *IACR Cryptology ePrint Archive*, 2004:272, 2004.

TOP500. Performance development, 2016. URL https://www.top500.org/statistics/perfdevel/. [Online; accessed April 29, 2017].

NB: place here information about funding, FCT project, etc in which the work is framed. Leave empty otherwise.