LGPD Compliance: A security persistence data layer

Paulo E. B. Pitta, Elder Costa, João P. L. de Siqueira, Nilson M. Lazarin

[1]Bacharelado em Sistemas de Informação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ) – Nova Friburgo, RJ – Brazil

{paulopitta97,eldercostaoficial,joaopslorenzo,nilsonmori}@gmail.com

Abstract. *Recent data protection laws define rules for the treatment of personal information of Internet users and oblige technology companies to implement security techniques in new or pre-existing information systems. Through an integrity breach algorithm, this article addresses possible techniques for converting a clear database to a ciphered database in order to provide greater protection to databases and lower the design costs to comply with such adaptations. At the end of the experiments, it was also possible to notice a brief improvement in performance in relation to other approaches, since it uses access to volatile memory.*

Resumo. *As recentes legislações de proteção de dados definem regras para o tratamento das informações pessoais dos usuários de internet e obrigam empresas de tecnologia a implementar técnicas de segurança em sistemas de informação novos ou preexistentes. Através de um algoritmo de quebra de integridade, este artigo aborda possíveis técnicas para conversão de uma base com dados em claro para uma base cifrada de maneira a fornecer maior proteção a bancos de dados e diminuir os custos de projeto para cumprir tais adaptações. Ao fim dos experimentos, foi possível notar ainda uma breve melhora na performance perante outras abordagens, visto que esta utiliza-se de acesso à memória volátil.*

1. Introdução

Diante de recentes escândalos envolvendo a segurança dos dados de usuários de diversos serviços, governos instituíram legislações específicas para proteção de dados eletrônicos, tais como: GPRD na União Europeia e a LGPD no Brasil [Rapôso et al. 2019]. Ao enrijecer a forma como os dados devem ser tratados e armazenados, estas leis obrigam empresas a adotarem técnicas de proteção de dados. Entretanto, o custo e o tempo necessário para adequar sistemas preexistentes às novas leis pode ser uma fator inviabilizante para pequenas e médias empresas, pois, conforme [Dias et al. 2012], a segurança da informação deve ser tratada com muito cuidado, uma vez que qualquer erro pode acarretar perdas drásticas e até mesmo comprometer o funcionamento destas organizações.

Algumas propostas de uso de camadas para o armazenamento seguro de informações em banco de dados têm sido apresentadas no últimos anos. Em [Lorey et al. 2016] é utilizado o modelo TEAL (*Transparent Encryption for the database Abstraction Layer*), uma forma de aplicar criptografia na camada de abstração do banco de dados. Entretanto, a sobrecarga adicionada dificulta a adoção da técnica, tendo em vista que a cada consulta todos os dados da tabela precisam ser decifrados para comparação.

Em [Deshpane et al. 2012], por sua vez, é apresentado um *middleware* para realizar a criptografia dos dados reescrevendo os comandos *SQL*. No entanto, é utilizada uma cifra clássica de substituição polialfabética frágil à ataques de análise de frequência.

Este trabalho apresenta uma abordagem para adoção de criptografia em banco de dados de sistemas preexistentes, através da tradução de comandos *SQL* na camada de persistência de dados, utilizando criptografia simétrica para armazenar os dados em disco, além de propor um algoritmo de quebra de integridade, para persistir os dados em memória volátil, melhorando a performance do sistema em novas consultas e contribuindo com a diminuição de custos na adequação de projetos às novas leis de proteção de dados.

O artigo está estruturado da seguinte forma: A seção 2 apresenta a abordagem proposta, onde é desenvolvida a abordagem e funcionamento do algoritmo. Na seção 3, implementação e experimentos demonstram os resultados e ambiente em que fora realizado os testes. A seção 4, conclusão, expõe a opinião dos autores sobre o que fora discutido.

2. Abordagem proposta

Este trabalho apresenta uma abordagem de tradução de consultas localizado na camada de persistência de dados, capaz de criptografar um banco de dados já existente evitando que seja necessário refatorar toda a camada de persistência dos dados. Além disso, para acelerar os acessos de leitura, os dados são mantidos em memória e protegidos por um algoritmo gerador de entropia usado para alterar a integridade dos registros nas tabelas.

A conversão de uma base de dados existente em disco em uma base de dados criptografada em disco é realizada através de um *dump* da base original onde são modificadas as estruturas das tabelas para *engine Memory*; posteriormente as tabelas são duplicadas e alteradas com um prefixo indicador de criptografia e *engine InnoDB*; por fim os dados do *dump* são criptografados e armazenados nas tabelas em disco através do AES (*Advanced Encryption Standard*), uma cifra simétrica de bloco que possibilita chaves de 128, 192 ou 256bits.

Após a conversão da base de dados, é necessária uma breve adaptação na camada de persistência da aplicação, conforme apresentado na Figura 1, substituindo o recurso padrão de manipulação do banco por uma que deve ser responsável por refletir a abordagem proposta, já que quando uma determinada consulta é realizada, as tabelas utilizadas são disponibilizadas em memória, acelerando o tempo de resposta para novas consultas, bem como a funcionalidade de limpar os dados de tabelas inativas na memória.

Para garantir a segurança dos dados em memória esta abordagem propõe um algoritmo de quebra de integridade, pois, algoritmos convencionais de criptografia são limitados a determinados tipos de campos, tais como: *binary, blob, string*. O algoritmo fornece um *token* de 256bits usado para alterar os dados armazenados na tabela em memória, através da fórmula $Dado_{ofuscado} = (Dado_{emclaro} + Token)\ mod\ N$, onde: N é o limite do campo de dados. Dessa forma é possível que uma data, submetida ao algoritmo de quebra de integridade, gere uma nova data também válida para armazenamento em campo do tipo *date*.

Pode-se observar, na Figura 2, o funcionamento do algoritmo utilizado para quebrar a integridade dos dados. Este algoritmo é baseado em uma Cifra de Feis-

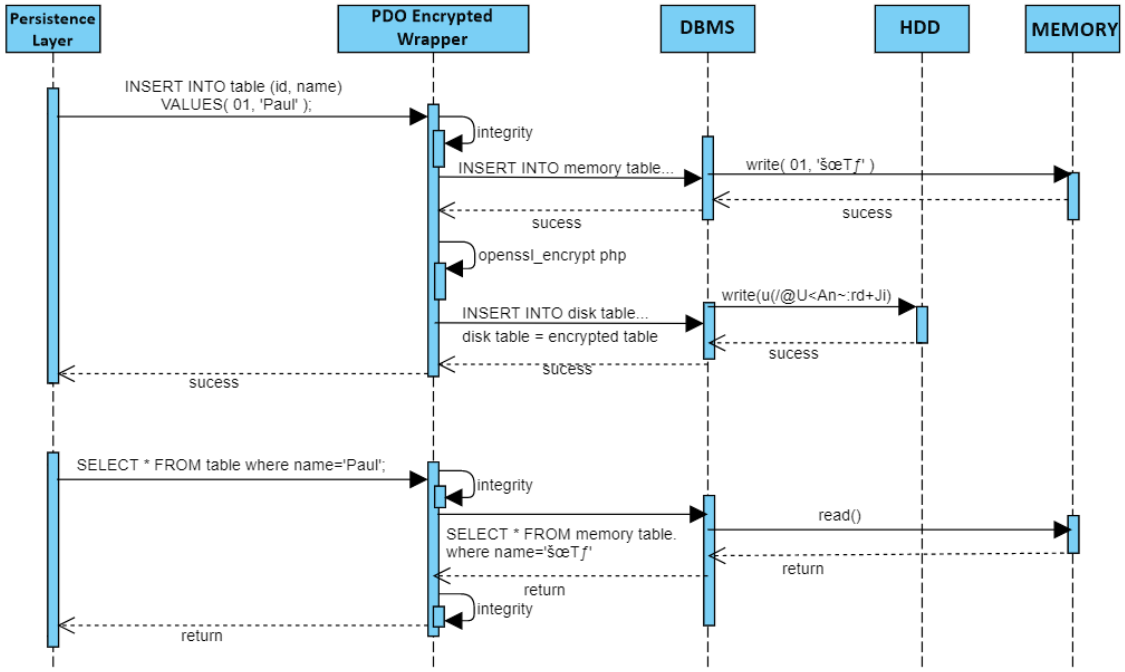

Figura 1. Funcionamento da abordagem proposta

tel de 10 rodadas. Como entrada ele recebe uma semente de qualquer tamanho e uma chave de 128bits. A chave é submetida ao KeySchedule do algoritmo AES [Rijmen and Daemen 2001] onde são geradas 10 subchaves, uma para cada rodada.

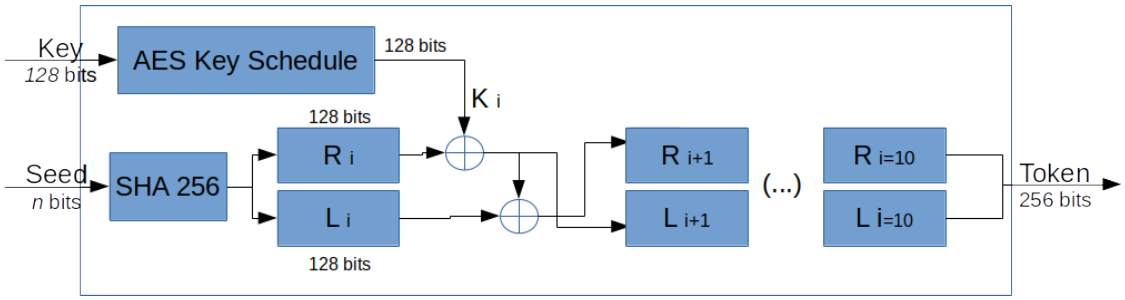

Figura 2. Gerador de entropia

3. Implementação e Experimentos

A abordagem apresentada foi implementada na linguagem PHP e encontra-se disponibilizada no GitHub[1]. Possui orientação a objetos, tendo classes e métodos com suas responsabilidades para cifrar e decifrar dados e objetos, realizar a conversão da base de dados, bem como outra classe responsável por atuar na camada de persistência sendo chamada ao invés da PDO padrão, que tem seu uso encapsulado na classe em questão.

Para a realização dos testes, foi utilizado um VPS (*Virtual Private Server*) CPU: 4, Clock: 2.2GHz, HD: 160GB SSD, RAM: 8GB SO: Debian 9. A base de testes utilizada foi a *Employees* disponível no site do MySQL[2]. Foram realizadas 256 baterias de testes,

[1]https://github.com/paulopitta97/aes-crypt-php
[2]https://dev.mysql.com/doc/employee/en/employees-installation.html

em quatro bases de dados de tamanho distinto, contendo 6 tipos de *queries*, sendo elas tipo: INSERT; DELETE; UPDATE; SELECT ALL; SELECT ID; SELECT JOIN. Foram analisados quatro indicadores:

- O tempo médio para execução de cada tipo de *querie* no banco de dados;
- O tempo médio gasto para conversão de uma base de dados;
- O tempo gasto para decifrar e disponibilizar em memória toda uma base de dados (up Memory);
- O consumo de memória RAM para manter a toda uma base na memória;

O método proposto neste artigo foi comparado com uma implementação que faz uso dos recursos criptográficos do próprio MariaDB através das funções AES_ENCRYPT e AES_DECRYPT, para retratar o uso de um sistema que realiza essas operações a cada busca no banco de dados. Os resultados dos experimentos demonstram que a abordagem proposta, possui um melhor desempenho em operações de INSERT, DELETE e UPDATE em tabelas com mais de 16mil tuplas. Isso é dada a característica do modelo, visto que essas operações são realizadas em memória e em disco. Por sua vez, as operações de SELECT (ALL, ID e JOIN) possuem um desempenho muito melhor ao se utilizar o modelo proposto neste trabalho. Os resultados são apresentados na tabela abaixo:

		AES_ENCRYPT() e DECRYPT()				MÉTODO PROPOSTO			
Qtd de Tuplas		1k	16k	32k	64k	1k	16k	32k	64k
em milisegundos	Insert	6,8	24	40,3	78,3	9,2	10,7	12,0	15,7
	Delete	3,9	24,1	47	91,3	7,0	23,6	39,3	68,2
	Update	3,8	22,3	43	84,7	5,1	21,8	37,1	66,6
	Select All	19	127,2	256,1	623,7	7,2	50,9	98,9	194,9
	Select Id	2,6	12,6	25,1	47,7	3,1	3,2	3,1	3,0
	Select Join	2834	332851	1319987	5225300	9,4	96,7	190,4	386,7
	up Memory	-	-	-	-	498,9	7329	14273	28980
	Destroy	-	-	-	-	4,1	6,0	6,3	8,0
	Convert	-	-	-	-	650	8056	15994	31233
Memory (MB)		455	504	537	634	645	689	722	747

4. Conclusão

Este trabalho apresentou uma abordagem para adoção de técnicas criptográficas em banco de dados já existentes, contribuindo com a diminuição de custos de projeto para adequação às legislações de proteção de dados. A grande vantagem do método proposto, em relação a outras abordagens é a performance melhorada, através da manutenção das tabelas em memória volátil. Foi implementada uma biblioteca que atua como *middleware* para tradução de consultas localizada na camada de persistência de dados responsável por gerenciar as tabelas em disco e em memória, evitando modificação da camada de persistência de um software em produção ao encapsular a implementação original.

Os experimentos realizados demonstram que esta proposta é promissora, pois, através do principio de localidade temporal, os dados são mantidos em memória e novos acessos de leitura são otimizados, equilibrando segurança e performance. Dessa forma, busca-se contribuir para que sistemas existentes possam se adequar as novas legislações de proteção de dados, diminuindo o custo de projeto e refatoração.

Referências

Deshpane, A., Patil, A., Joshi, S., and Botara, S. (2012). *DBCrypto: A Database Encryption System using Query Level Approach*. International Journal of Computer Applications. Vomue 45 - N 08.

Dias, J. M. F., de Cássia M. C. Rodrigues, R., and Pires, D. F. (2012). *A Segurança de Dados na Computação em Nuvens nas Pequenas e Médias Empresas*. RESIGeT - Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica. Vol. 02, Mr. 1.

Lorey, K., Buchann, E., and Böhm, K. (2016). *TEAL: Transparent Encryption for the Database Abstraction Layer*. Fórum CAiSE16 na 28 Conferência Internacional de Engenharia de Sistemas de Informação, Ljubljana, Eslovrênia.

Rapôso, C. F. L., de Lima, H. M., de Oliveira Junior, W. F., Silva, P. A. F., and de Souza Barros, E. E. (2019). Lgpd-lei geral de proteção de dados pessoais em tecnologia da informação: Revisão sistemática. *RACE-Revista da Administração*, 4:58–67.

Rijmen, V. and Daemen, J. (2001). Advanced encryption standard. *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pages 19–22.