Implementação e comparação de *trade-offs* criptoanalíticos

Lívia dos Reis Edinger da Silva[1], Fernanda Larissa Müller[1], Yolanda Caroline Colombo[1], Frederico Schardong[1]*

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - *Campus* Rolante
*Orientador

Resumo

A segurança da informação tem papel fundamental na sociedade atual, garantindo desde a inviolabilidade de comunicações sensíveis e transações financeiras até a autenticação de entidades na Internet. Funções de *hash* são amplamente utilizadas nessas aplicações. Estas funções unidirecionais determinísticas recebem qualquer informação como entrada e produzem uma saída aleatória de tamanho fixo que não pode ser revertida a sua entrada. Esta pesquisa experimental tem como objetivo comparar empiricamente três características computacionais (quantidade de dados armazenado em disco, uso de memória e quantidade de processamento) de três técnicas de *trade-offs* criptoanalíticos que revertem o resultado de funções de *hash*. As técnicas foram implementadas em Python e comparadas em três dimensões: (i) ataque de força bruta para o processamento; e ataque de dicionário para o uso de (ii) memória e (iii) disco. Como as técnicas Hellman e Rainbow Tables são probabilísticas, ou seja, sem garantia que todos os *hashes* que fazem parte do banco de dados podem ser revertidos, foram analisados apenas os testes onde 100% dos *hashes* submetidos a estes algoritmos foram revertidos. O terceiro algoritmo, chamado de Vanilla, é o único não probabilístico, ou seja, todos os *hashes* que compõem o banco de dados sempre são reversíveis. Para realizar a verificação empírica das características computacionais destes algoritmos, cada técnica foi submetida a múltiplos testes, variando os parâmetros de configuração e utilizando o mesmo conjunto de dados de entrada (2^{17} números inteiros). Ao analisar apenas os resultados com 100% de sucesso e que utilizam 50% do espaço de disco em relação ao ataque de dicionário, pode-se comparar: (i) processamento e (ii) memória. Os resultados encontrados foram: (i) a quantidade de processamento (operações de *hash*) é de 0,0038, 0,0055 e 0,213 vezes a quantidade de processamento do ataque de força bruta, respectivamente para, Rainbow Tables, Hellman e Vanilla; e (ii) a quantidade de memória utilizada é de 3, 0,15 e 0,002 vezes o ataque de dicionário, respectivamente para Rainbow Table, Hellman e Vanilla. É possível observar que: (i) os algoritmos probabilísticos precisam de cerca de 45 vezes menos processamento que o Vanilla; e (ii) Hellman e Rainbow Table utilizam, respectivamente, cerca de 75 e 1500 vezes mais memória durante a reversão dos hashes. Conclui-se que apesar de utilizar mais operações de hash, o algoritmo Vanilla usa exponencialmente menos memória que os algoritmos probabilísticos, o que o torna competitivo em arquiteturas que privilegiam processamento em detrimento a memória como GPUs e FPGAs.

Palavras-chave: Criptografia. Funções de *hash*. *Trade-off*.