Relatório do Projeto de Criptografia

Prova de Merkle

Aluno: Rodrigo Abreu Alves de Freitas Mota
Professor: Ruy Queiroz

Introdução

A árvore de Merkle é uma estrutura para organizar as informações de hashes de uma sequência de bytes garantindo a integridade desta sequência e certa capacidade de rastreio de inconsistências. A árvore é organizada como uma binary tree, com cada node contendo um digest obtido a partir de uma hash function aplicada a combinação (soma) dos digest dos seus dois filhos (caso não seja uma folha). As folhas da árvore de merkle possuem os hashes dos dados originais.

Usando esses hashes é possível gerar uma prova de que determinado elemento está em uma posição específica desse conjunto de dados sem a necessidade de acessar diretamente esse conjunto de dados. Usando apenas uma quantidade exponencialmente menor de dados é possível garantir esta integridade. Essa prova criptográfica é conhecida como prova de Merkle.

Montagem da árvore

Divide-se os dados originais em subsequência de bytes, sendo a quantidade dessas subsequências N = 2^n (ou seja, uma potência de 2). Para cada subsequência aplica-se uma hash function, obtendo N digests, que serão as folhas da árvore. Após isso, agrupa-se pares consecutivos de nós (1 e 2, 3 e 4, 5 e 6, …), soma cada um desses pares, e, por fim, aplica-se a hash function ao resultado, que será o digest dos parent nodes dos nodes atuais. Teremos N/2 = 2^(n-1) nodes resultantes, e os mesmos passos anteriores são repetidos para gerar os parent nodes desses N/2 novos nodes.

Após repetir esse processo n vezes, teremos 1 único node, que será o raiz. Este deve ser publicado e confiado.

Geração da prova

Podemos provar que a sequência de bytes x é a subsequência na posição pos dentro de uma sequência maior, usando apenas n nodes da árvore, apenas os necessários para reconstruir o digest do node raiz.

Como x e pos são informações da prova, sabemos também o digest de x e sabemos a posição da sua dupla, que são os bytes em pos+1 ou pos-1 (dependendo se pos é par ou ímpar). Com o digest desses dois podemos gerar o digest do parent node. Com o digest do

node que é par deste parent node podemos gerar o digest do node acima destes dois últimos. Desta forma, se conseguimos o digest de um node na posição p da altura h da árvore e nos for fornecido o digest do par dele, podemos obter o digest do node na posição p/2 da altura h+1 da árvore (considerando as folhas na menor altura). Portanto, é fácil ver que precisamos de um digest para cada altura.

Como é praticamente impossível encontrar um x' diferente de x tal que hash_function(x) == hash_function(x'), é extremamente difícil fornecer os n-1 hashes da prova de forma que o digest do node raiz seja reconstruído.

Verificação da prova

Aplica-se a hash function recursivamente a soma dos hashes recebidos na prova. Se o resultado final for igual ao digest do raiz de node pode-se dizer que o elemento procurado está na posição sugerida da sequência de bytes.

Algumas aplicações

Integridade de bancos de dados distribuídos new sql, pois com a árvore de Merkle pode-se verificar qual node tem o digest diferente fazendo um BFS começando no node raiz, desta forma, não é necessário seguir pelos descendentes dos nodes com digest corretos. Poupando tempo de processamento exponencialmente para o caso de apenas uma folha inconsistente.

Criptomoedas usam provas de Merkle para otimizar a consulta de transações em blocos.

Versionamento de arquivos como o git podem usar para garantir a integridade dos arquivos e procurar qual arquivo foi modificado.