Sistema de Propriedade de dados baseado em Hash

HBPS – Hash Based Property System

Introdução

Nesse paper, apresento um conceito interessante para um sistema de propriedade digital baseado em hash (portanto, suficientemente resistente a computação quântica), de grande interesse para blockchains.

O sistema também provê relativo anonimato baseado em seu próprio funcionamento interno. Não requer uma matemática extremamente complexa (ao menos relativamente), e funciona bem para a maioria dos casos.

Criado para ser uma solução inteligente à ameaça que os computadores quânticos podem representar para sistemas de blockchain, baseado na resistência dos algoritmos de hash atuais (sua força depende da resistência dos algoritmos e pode ser implementado em qualquer um, arbitrariamente, de acordo com sua necessidade).

Símbolos Utilizados

K = Chave “privada”

PK = Chave pública derivada de K por uma função de hashing

A = Pessoa interna (que estamos assumindo o papel de proprietária)

B = Pessoa externa (pode ser extrapolado para C, D, E…)

APK, BPK, CPK = Chave pública (PK) de (B, C, D, E…), para fins representativos

T = Propriedade

Proposta e Funcionamento

A ideia básica do sistema é:

A sempre possui apenas um K por T

Quando T for transacionado de alguma forma, todo seu conteúdo (independentemente do que for, seja moeda, NFT ou qualquer outra coisa) deve ser esvaziado na operação (mais sobre isso adiante)

Simplesmente isso.

A estrutura da propriedade e da prova de propriedade está assegurada sobre a unidirecionalidade dos algoritmos de hashing, podemos entender isso assim:

Para toda chave pública de A, uma única chave privada K é possível para que PK seja gerado, ou seja, ninguém pode gastar T a menos que conheça de antemão K (ou tenha passado por força bruta em todos os valores de K possíveis que poderiam gerar PK)

A prova que A possui T e está disposto a transferir para B é que ele torna público K

B, nesse ponto, pode verificar que A era realmente o destinatário de T inicialmente (pois possuía a única chave K conhecida capaz de gerar PK)

Como T foi gasto transferindo de PK para BPK, nenhuma propriedade é comprometida quando K é comprometido, e A apenas precisa gerar outro K para ser dono de algum outro T.

As vantagens desse sistema são que como A é sempre o proprietário de apenas um recurso homogêneo por K/PK, comprometer K não compromete todos os seus fundos mas apenas os de PK (que nesse momento já foram transferidos e portanto tornam PK vazio, impedindo qualquer espertinho de gastá-los por A)

Podemos listas as vantagens e desvantagens (bem como a resposta, se houver) desse sistema como:

- Resistência a computação quântica, a menos que seja possível resolver o Hash(K) em tempo computacionalmente hábil (o que acredita-se ser impossível no momento, se for possível, um novo algoritmo pode ser simplesmente usado para gerar as chaves)
- Anonimidade: Devido à necessidade constante da criação de chaves públicas e privadas para receber as propriedades digitais, vincular todos esses endereços a um único indivíduo pode ser praticamente impossível
- Devido aos endereços públicos serem, até o momento em que K é revelado, anônimos, é possível mantê-los como reservas indefinidas de dinheiro (pode-se transferir mais T para PK desde que este ainda não tenha sido gasto – o que revelaria K)

- Armazenamento: Devido à óbvia necessidade desse sistema de criar inúmeras chaves privadas e públicas, o armazenamento e gerenciamento dessas chaves pode ser um grande problema para contas muito movimentadas. Isso pode ser mitigado pelo seguinte fator: Apenas as chaves que atualmente possuem fundos precisam ser armazenadas pelo proprietário A, todas as outras podem ser descartadas tendo em vista que, obviamente, não possuem mais fundos (e que seria completamente insano adicionar fundos a uma chave que foi comprometida durante a transferência)
- Geração de chaves: A necessidade de criar sempre novas chaves pode ser um problema se o que é almejado é um sistema enxuto que permite gerar chaves deterministicamente, pode ser mitigado com uma chave realmente privada (chamaremos de AnonK) que gera chaves privadas K (podemos chamá-las de geradoras ou autenticadoras para maior facilidade), em uma determinada ordem de gastos (dessa maneira, deve ser possível que uma única chave ou código seja exportável, ou mesmo calculável pelo usuário de forma offline, e que permita sua importação dinâmica em outro sistema em um momento posterior, carteiras offline por exemplo, poderiam manter um registro interno de endereços criados pela chave privada e, com isso, poderem recriar as chaves independentes do dispositivo, etc…)
- Introduz algumas dificuldades ao sistema subjacente: Para que uma moeda em PK seja parcialmente gasta, seriam necessários alguns artifícios como a criação de endereços arbitrários de recebimento alheios à transação principal

Assinaturas

Um esquema de assinaturas arbitrárias com HBPS pode ser formulado através de time locks.

O que são estes time locks e como isso funcionaria?

Em primeiro lugar, torna-se necessário esclarecer o seguinte: As assinaturas deverão ter o tamanho ideal de duas vezes o da função de hashing utilizada pelo sistema, e devem ser estruturadas assim:

S = Sx:[HASH(PK sobre DADOS)] + Sy:[HASH(K sobre Sx)]

A assinatura é feita em duas etapas, primeiro, gera-se o hash da mensagem (DADOS) utilizando-se da chave pública, em seguida, gera-se o hash de Sx (hash gerado anteriormente) utilizando-se da chave privada.

Idealmente, no atual estágio de seu desenvolvimento, as assinaturas arbitrárias em um esquema HBPS time-locked devem ser tratadas como promissórias entre A e B, seja B uma pessoa ou mesmo algo como um smart-contract.

O modo como isso deve ser feito pode ser entendido dessa forma:

A deseja criar um contrato com B onde ele, dentro de um período de 3 dias, se compromete a enviar 900 moedas para o endereço vinculado, e essas moedas serão despachadas para o endereço público BPK de B, a condição é que o endereço, apesar de contê-las, não precisa dar as 900 moedas de antemão mas sim até o fim do período de tempo estipulado.

A então produz, ao longo dos 3 dias, 3 assinaturas que acompanham a promessa de render 300 moedas para a chave BPK de B, completando assim as 900 moedas estipuladas.

B pode verificar, a cada assinatura, que pelo menos 50% dela corresponde a chave pública PK, e espera que com o despacho da chave K ele possa validar perante a rede PK e suas três assinaturas.

Ao final dos 3 dias, ele então emite uma “Confirmação de assinatura” em que entrega para B sua chave P associada com sua chave PK. Tendo a chave K sido despachada, B pode verificar se as 3 assinaturas emitidas por A durante o período de tempo estipulado são mesmo válidas, tendo confirmado isso a rede assimilará a transação e a chave K/PK estará finalmente descartada.

A proposta de duas etapas da assinatura, metade PK, metade K, está relacionada com a possibilidade de um agente mal intencionado que realiza assinaturas arbitrárias durante o período de 3 dias sem o conhecimento de A. Nesse sentido, como o agente malicioso não conhece K ele não pode emitir a parte final da assinatura e, portanto, ela não pode ser verificada ao final dos 3 dias por nenhum agente interessado.

A desvantagem óbvia desse sistema é que A pode estar atualmente mentindo sobre deter a propriedade da chave PK, e estar emitindo com ela um hash inválido secundário de autenticação, nesse sentido ao final dos três dias os fundos emitidos seriam simplesmente considerados inválidos (essa perspectiva é a mesma do agente malicioso externo, mas nesse caso, é o agente principal do contrato utilizando-se de malícia). Nesse sentido, é importante tratar as assinaturas como não mais do que promissórias até o período especificado, pois embora possa ser publicamente verificável que

o endereço PK possua os fundos sendo movimentados, não é o caso que possa ser publicamente verificado por qualquer agente interessado, antes do período final, que PK é de propriedade de A.

A forma de lidar com isso pode ser simplesmente a de tratar as ações de A durante o período de assinatura como ações virtuais de validação futura, isto é, nenhum agente envolvido irá utilizar nenhum endereço que não deve ser comprometido no tempo D como gastador, assim ao final do período apenas aqueles participantes que puderem ser comprovados pela rede irão formar cadeias de assinaturas válidas (e portanto, provas de propriedade e transferências de propriedade). Se algum agente crucial do processo for um agente malicioso, a rede irá simplesmente tratar o que foi inserido como informações falsas e vazias de conteúdo e, nesse sentido, proteger-se de agentes maliciosos.

Os custos de transacionar falsamente na rede, por outro lado, devem ser considerados, proponho o seguinte modelo:

- Todos os participantes da interação com o contrato se comprometem a pagar uma taxa X para a rede
- Ao final do contrato, as promessas de taxas são tratadas da seguinte forma:
    - Se todos os participantes são proponentes válidos, então todos pagam suas taxas como prometido
    - Se, por outro lado a rede de assinaturas for comprometida de alguma forma, então:
        - Os proponentes válidos têm suas promessas de taxas canceladas, tendo os fundos “retornados” para algum endereço PK arbitrário de estorno (selecionado por eles, é claro)
        - Os proponentes inválidos arcam com todos os custos da transação, assim, todas as taxas são debitadas dos endereços públicos associados a eles diretamente para a rede
- Uma adição interessante, mas opcional, seriam as chamadas garantias ou “collaterals”, onde os participantes da transação reservam em um fundo terciário uma quantidade X de moedas que serão utilizadas para o pagamento das taxas ou mesmo indenizações no caso de fraude aos membros (e no caso dos valores dos endereços dos agentes maliciosos não puderem cobrir todas as taxas também). Em caso de agentes maliciosos entrarem no contrato, os fundos de seus colaterais seriam subtraídos automaticamente para a função que lhes for atribuída (como os fundos de colaterais são apenas endereços públicos PK, existe a chance de um agente malicioso enviar o endereço de um terceiro como garantia, por essa razão, os chamados “bullet addresses” devem ser utilizados – mais sobre eles a seguir).

Bullet Addresses

Endereços relampago, de tiro, ou simplesmente bullet addresses, são chaves criadas imediatamente na rede e cujos valores são imediatamente conhecidos na própria declaração dos mesmos.

Utilizados para prevenir agentes maliciosos de utilizarem-se de chaves inocentes para seus fins, eles podem até mesmo reutilizar endereços públicos (não-comprometidos) pois, na medida em que os fundos adicionados a esses endereços são de origem externa, os agentes maliciosos não podem fazer nada contra eles além de modificarem os valores perante a rede que já foram comprometidos em primeiro lugar.

Nesse sentido, se o agente malicioso C assina como colateral o endereço arbitrário PK^3, pagando-o 1000 moedas através do gasto do endereço K^1/PK^1 de sua conta, na medida em que seja revelado como malicioso as 1000 moedas serão debitadas automaticamente pela rede em uma transação validada por regras (isto é, que depende apenas do endereço público, elas são assim porque apenas podem gastar algo que for sistematicamente reconhecido como um gasto válido, e não precisam portanto comprometer a chave privada subjacente), e C terá efetivamente perdido seus fundos enquanto o usuário dono do endereço PK^3 continuará com seu saldo normalmente.

Exemplos Práticos

Com um algoritmo de mineração H, a blockchain deseja dar para A 1000 moedas

A então gera K^1, e PK^1 para receber as moedas, a blockchain registra 1000 moedas em PK^1

A tabela de endereços de A é: PK^1/K^1 (1000 moedas)

A deseja comprar de B um produto X, por 300 moedas

A deve:

- Obter o endereço BPK de B
- E, em um mesmo bloco:
  - Enviar 700 moedas para um novo PK^2 de K^2 privado
  - Enviar 300 moedas para BPK
  - Expor K^1

Ao fazer isso, B pode provar a autenticidade das moedas de A gerando PK^1 com o K^1 exposto, possuindo agora suas próprias moedas em BPK (obviamente com sua chave privada BK segura).

A, por outro lado, agora possui a seguinte tabela de endereços:

- K^1/PK^1 (Zero moedas) Comprometido
- K^2/PK^2 (700 moedas) Seguro

Desse modo, A consegue dividir suas moedas com B, sem comprometê-las perdendo seu endereço principal.

Recomendações

Para uma implementação inicial recomendo os seguintes parâmetros:

- SHA512 como algoritmo de hashing
- Uma chave principal sempre privada geradora de todas as outras (para facilitar a portabilidade). A chave deve ser imprevisível de todas as formas para não comprometer K^n
- Transações atreladas para que sempre seja o caso que um endereço PK seja esvaziado instantaneamente

Experimental

Proposta de multi-assinaturas finitas baseadas em um único hash

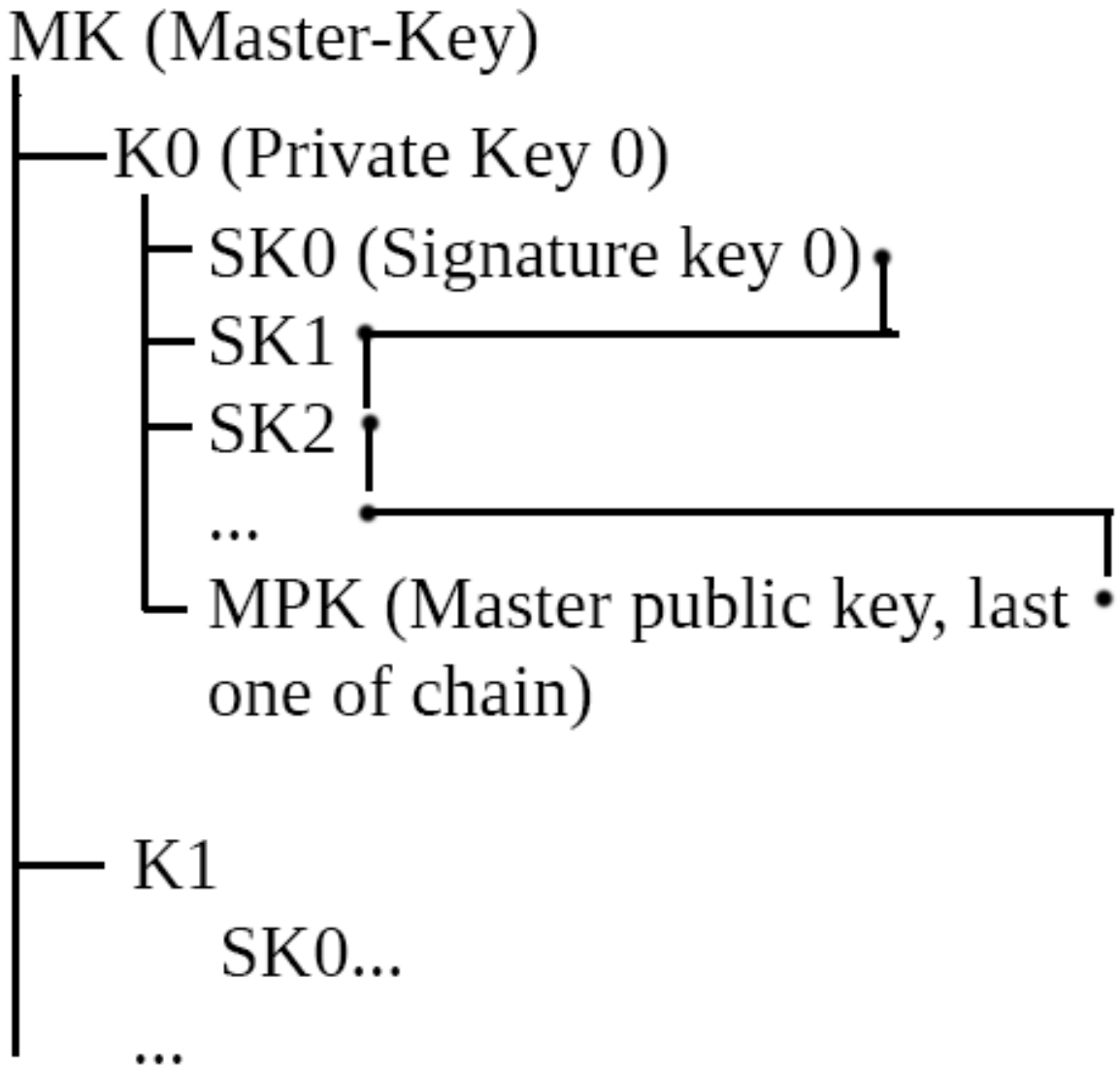
Finite Hash-Based Multi-Signature, FHBMS

O funcionamento dessa proposta depende do reconhecimento sistêmico da mesma, opera sob a suposição de que o conjunto de assinaturas PK^n inferiores a chave pública mestra MPK irão ter validade sob a premissa de gerarem elas mesmas MPK iterativamente. Isto é, se uma chave interior PK^n puder, sob uma premissa iterativa, gerar MPK, então MPK deve ser a proprietária legítima do que quer que estiver sendo atribuído. MPK portanto se torna a recebedora e todas as PK^n “chaves de transferência publica-privada”. Isto é, desde que um PK^n anterior seja possível de ser emitido, tal que PK^n possa gerar de alguma forma MPK, então PK^n será considerado uma assinatura válida para o gasto de MPK.

O problema principal com essa abordagem é que, necessariamente, MPK terá um número finito de assinaturas que deverão ser exclusivamente utilizadas, uma chave inferior PK^n emitida para MPK não deve ser de forma alguma reconhecida pelo sistema como válida uma vez que já tiver sido emitida.

Considerando a pressuposição necessária, esse sistema de multi-assinatura é inferior ao sistema de assinaturas únicas K/PK pois nesse caso PK está vinculado apenas a K, e portanto o sistema não tem escolha além de aceitá-lo como prova definitiva sobre PK.

Idealmente, esse esquema deve ser gerado da seguinte forma:



Como podemos ver no esquema acima, idealmente, a chave privada mestra MK deve gerar as chaves privadas base K0, K1, K2…

As chaves privada base, por sua vez, devem derivar delas as chaves de assinatura SK^n, que devem, por sua vez, derivar $SK^{n!}$, etc… até que a chave pública mestra MPK seja derivada (isto é, o ponto final da cadeia)

MPK deve ser possível de ser provada por qualquer SK^n, e qualquer SK^n deve ser provável através de SK^{n-1}, etc… até que tenha atingido K.

A quantidade de chaves SK^n derivadas de K até MPK determinam o número de assinaturas possíveis pra MPK, e o tempo de geração de MPK será equivalente ao tempo necessário para gerar as n assinaturas possíveis de serem feitas para K/MPK

O tempo de verificação de uma assinatura SK^n será igual ao número de iterações necessárias até que MPK seja derivado, no pior dos casos, o tempo será igual ao tempo de geração de MPK.

A prova consiste em verificar se em algum ponto da cadeia de derivações K/SK^n irá derivar MPK, se isso for o caso, então é improvável que K/SK^n tenham sido criadas arbitrariamente por algum agente malicioso (contanto que MPK seja apenas recipiente), e portanto a assinatura única K/PK^n terá sido demonstrada válida em relação ao container de valor MPK.

É claro, o esquema de multi-assinaturas depende da integridade do sistema em que reside, se for um servidor, é preciso que nenhum agente malicioso possa mudar suas regras internas, e que elas sejam bem estabelecidas. Se for uma blockchain, of course, é preciso que pelo menos 51% dos nós estejam comprometidos com as regras também, algo que obviamente é necessário para uma blockchain funcionar em primeiro lugar então não deve ser um problema absurdo.