Bitcoin: Matemática Descentralizada

Resumo

O Bitcoin é muito mais do que uma simples moeda digital: é um sistema financeiro completo que funciona sem intermediários, autoridades centrais ou pontos únicos de falha. A sua robustez não vem de segredos ou de promessas, mas de problemas matemáticos extremamente difíceis de resolver, embora muito fáceis de verificar. Neste documento tento explorar, passo a passo e de forma simplificada, como a criptografia de curva elíptica, as funções de hash criptográficas e o mecanismo de prova de trabalho (Proof-of-Work) se combinam para criar um livro-razão imutável, seguro e verdadeiramente descentralizado.

O objetivo é que, mesmo quem só tem noções básicas de matemática, consiga terminar a leitura com uma compreensão sólida do porquê de o Bitcoin ser considerado "dinheiro impossível de falsificar". Vou servir-me de analogias do quotidiano, exemplos concretos e, sempre que possível, dar explicações intuitivas antes de entrar nos detalhes técnicos.

1. Introdução: Substituir confiança por matemática

Num banco tradicional depositamos o nosso dinheiro e confiamos que o banco regista corretamente o saldo, não o gasta sem autorização e não permite que outra pessoa o retire. Essa confiança baseia-se em leis, contratos e, em última instância, na reputação da instituição.

O Bitcoin elimina, por completamento, essa necessidade de confiança em terceiros. Em vez disso, qualquer pessoa pode verificar todas as regras matemáticas que dirigem o sistema. Se uma transação segue as regras, é aceite; se não segue, é rejeitada – esta é uma regra inquebrável do Protocolo Bitcoin.

Não há presidente de banco, não há regulador, mas mais importante, não há "exceções".

Os três grandes pilares matemáticos que tornam isto possível são:

1. A criptografia de chave pública, que permite provar que somos donos dos nossos bitcoin sem revelar a chave privada,
2. As funções de hash, que funcionam como selos de lacre digitais impossíveis de abrir sem partir,
3. O Proof-of-Work, que torna extremamente caro (em tempo e energia) reescrever o histórico de transações.

Abordarei agora cada um deles, começando pelo que se encontra mais próximo do utilizador comum: as funções de hash.

2. Funções de Hash: o selo digital que ninguém consegue falsificar

À semelhança das missivas importantes da antiguidade lacradas e seladas com o selo imperial ou régio, qualquer pessoa podia ver que o lacre se encontrava intacto, com o selo e que a carta não teria sido aberta, porém ninguém conseguia abrir o documento e voltar a selá-lo com o mesmo lacre, sem ter o selo original.

As funções de hash criptográficas fazem exatamente isso, mas para os dados digitais. Uma função de hash recebe uma entrada de qualquer tamanho (um caracter, uma frase, um ficheiro de vários gigabytes) e produz uma saída de tamanho fixo - no caso do Bitcoin, quase sempre 256 bits (equivalente a 64 caracteres hexadecimais), ou seja é uma transformação matemática que recebe uma entrada de tamanho arbitrário e produz uma saída de comprimento fixo, chamada *digest* ou simplesmente *hash*. No contexto do Bitcoin, esta saída tem quase sempre 256 bits (a exceção do passo intermédio de criação de endereços, cujo hash é de 160 bits), o que equivale a 64 caracteres hexadecimais.

No entanto, o que torna uma função de hash *criptográfica* não é apenas o facto de comprimir dados, mas sim o conjunto de propriedades de segurança que a tornam adequada para proteger sistemas como o Bitcoin.

Propriedades essenciais de uma função de hash criptográfica

Para que uma função de hash seja considerada segura e útil em criptografia, deve satisfazer três requisitos fundamentais, formalizados pela primeira vez nos anos 90 por investigadores como Ralph Merkle e Ivan Damgård:

1. **Determinismo (ou reproducibilidade)**

   Dada a mesma entrada, a função produz **sempre** o mesmo hash, independentemente de onde ou quando é calculada. Esta propriedade é essencial, no Bitcoin, para que todos os nós da rede cheguem exatamente ao mesmo resultado ao processar o mesmo bloco ou transação,

2. **Resistência à pré-imagem (one-way function)**

   Dado um hash h, deve ser, computacionalmente, inviável encontrar **qualquer** entrada m tal que $H(m) = h$. Em termos práticos: apresnetado o hash 1c8f2e…5a9b3f, não há forma realista de descobrir que a mensagem original era “Eu devo 10 BTC ao Bruno”.

   A única maneira de “inverter” é por força bruta (testar todas as entradas possíveis), o que, com 2^{256} possibilidades, é impossível mesmo com toda a energia do Sol convertida em computação durante milhares de anos.

3. **Resistência a segundas pré-imagens**

   Dada uma entrada m_1 e o seu hash $h = H(m_1)$, deve ser inviável encontrar uma entrada diferente $m_2 \neq m_1$ tal que $H(m_2) = h$. Esta propriedade, no contexto do Bitcoin, protege contra ataques em que um atacante tenta substituir uma transação válida por outra maliciosa com o mesmo hash.

4. **Resistência a colisões**

   Deve ser extremamente difícil encontrar **quaisquer** duass mensagens diferentes $m_1 \neq m_2$ tais que $H(m_1) = H(m_2)$. Pelo paradoxo do aniversário, teoricamente seriam necessárias cerca de 2^{128} tentativas para encontrar uma colisão em SHA-256 - um número tão grande que, mesmo com todos os supercomputadores do mundo a trabalhar durante a idade do universo (13,8 mil milhões de anos), a probabilidade seria quase nula.

5. **Efeito avalanche (sensibilidade extrema)**

   Uma alteração de um único bit na entrada deve alterar, em média, metade dos bits no hash. Este comportamento caótico é crucial para que pequenas mudanças (como alterar um cêntimo numa transação) invalidem completamente todos os hashes subsequentes na cadeia.

O Bitcoin utiliza principalmente a função SHA-256 (Secure Hash Algorithm 256-bit) em duplicado, criada pela NSA norte-americana, publicada como padrão aberto e estudada exaustivamente por décadas sem que tenha sido encontrada qualquer falha crítica.

Vejamos um exemplo concreto:

Entrada: “Eu devo 10 BTC ao Bruno” | SHA-256 → `1c8f2e…5a9b3f` (64 caracteres)

Se mudar apenas uma vírgula:

Entrada: “Eu devo 10 BTC, ao Bruno” | SHA-256 → `8d7e41…12f6c9` (64 caracteres, mas completamente diferente)

No Bitcoin esta propriedade é usada em quase tudo: endereços, assinaturas, ligação entre blocos e prova de trabalho. Se alguém tentar alterar uma transação antiga, todos os hashes subsequentes ficariam diferentes e a rede rejeitaria imediatamente o bloco alterado.

3. Criptografia de Curva Elíptica: a magia das chaves assimétricas

3.1 Porque é que o Bitcoin não usa RSA como os bancos?

A criptografia de chave pública permite enviar bitcoins para um endereço sem conhecer a chave privada do destinatário, garantindo que só o dono a possa gastar.

Nos bancos online, o padrão é o RSA (1977), baseado na dificuldade de fatorizar o produto de dois primos grandes — para 128 bits de segurança, precisa de chaves de 3072 bits. Satoshi Nakamoto, inteligentemente e como visionário, optou por curvas elípticas uma vez que oferecem o mesmo nível de proteção com apenas 256 bits, sendo muito mais eficientes em tamanho e velocidade. Chaves e assinaturas RSA ocupam 3 a 5 vezes mais espaço, sobredimensionando a informação das transações e, por conseguinte, a blockchain. Por outro lado, as verificações são 5-10× mais lentas em telemóveis e hardware wallets com RAM limitada.

A aritmética elíptica opera, nativamente, em registos de 32 bytes, ideal para dispositivos como Ledger ou Trezor, e para nós SPV [nós "light" que permitem aos utilizadores verificar transações numa blockchain sem descarregar a sua cópia completa.] em redes lentas.

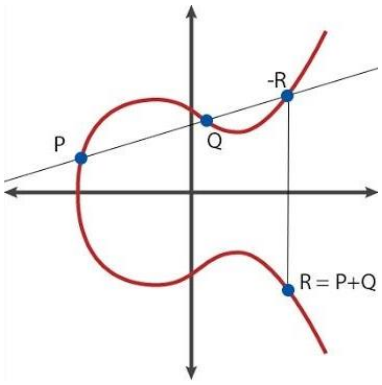
Satoshi escolheu a curva secp256k1 (domínio público, otimizada) em 2008, quando já era usada em TLS/SSH. Esta decisão pragmática tornou o Bitcoin leve, rápido e escalável sem sacrificar segurança - razão pela qual todas as principais criptomoedas a seguem.

3.2 O que é, intuitivamente, uma curva elíptica?

Para se entender o que é uma curva elíptica, é interessante imaginar um jogo geométrico que acontece num plano especial - uma montanha-russa com uma forma muito particular: começa larga na parte de

baixo, desce para um vale suave no meio e sobe de novo dos dois lados, formando uma espécie de "U" arredondado e simétrico.

Essa forma é o que define a curva — qualquer ponto que esteja nela tem de obedecer a uma regra fixa que relaciona a sua posição horizontal com a vertical.

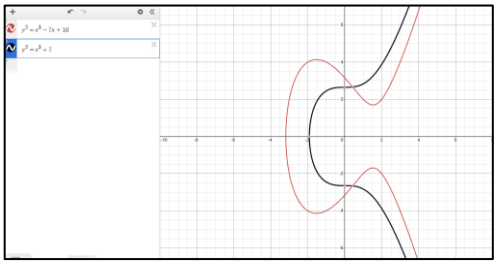


O truque mágico vem a seguir: em vez de somar pontos como fazemos com números normais, definimos uma operação especial de "adição". Escolhem-se dois pontos da curva, traçamos a reta que os liga, vemos onde essa reta volta a tocar na curva (num terceiro ponto) e, depois, refletimos esse terceiro ponto sobre o eixo horizontal para obter o resultado final da soma.

Esta operação parece complicada, mas tem regras simples e poderosas: a ordem não importa (somar A com B dá o mesmo que B com A), existe um ponto "neutro" que é como o zero (chamado ponto no infinito) e cada ponto tem um oposto que, quando somado, volta ao neutro. É exatamente como a soma normal, mas desenhada na curva.

No Bitcoin, este jogo não acontece num plano infinito com números reais, acontece num mundo finito, como um relógio gigante com um número primo enorme (com 77 digitos) de "horas". Todos os cálculos são feitos "dando voltas" nesse relógio, o que cria um conjunto fechado de pontos discretos, como pixels numa grelha invisível. A curva usada chama-se secp256k1 e foi escolhida porque é simples, rápida de calcular em computadores e não tem "portas traseiras" (back-doors) conhecidas. É este mundo finito e geométrico que permite criar chaves seguras sem precisar de autoridades centrais.

Agora, na montanha-russa com a forma $y^2 = x^3 - 7x + 10$, define-se uma operação especial de "somar" pontos da curva: traça-se uma reta que liga dois pontos, encontra-se o terceiro ponto onde a reta volta a cortar a curva e reflete-se esse ponto sobre o eixo dos xx.



Esta operação tem propriedades fantásticas: é comutativa, tem elemento neutro (o "ponto no infinito") e forma um grupo matemático.

No Bitcoin tudo isto é feito em aritmética modular, ou seja, "relógio de números" com o número primo gigantesco $p \approx 2^{256}$. E como dito acima, a curva usada é a secp256k1: $y^2 \equiv x^3 + 7 \pmod p$ com $p = 2^{256} - 2^{32} - 977$.

3.3 O segredo que protege os bitcoin: o Problema do Logaritmo Discreto Curvas Elípticas (ECDLP)

Agora que temos o "palco", a curva com a sua operação de soma especial, entra o grande segredo que protege cada bitcoin, um problema matemático que é fácil de fazer numa direção, mas praticamente impossível na outra. Tudo começa com um ponto fixo e público, chamado ponto base (vamos chamar-lhe G), que toda a rede Bitcoin conhece. A sua chave privada é simplesmente um número aleatório gigante — pensemos nele como uma senha com 77 dígitos. Para criar a chave pública, o sistema serve-se desse ponto G e "soma-o a si mesmo" exatamente esse número de vezes. É como subir uma escada: cada passo é uma adição na curva, e no final chega-se a um novo ponto, que é a sua chave pública (Q).

Este processo é rápido — o computador faz isto em frações de segundo, mesmo com números tão grandes. O problema surge quando alguém tenta fazer o caminho inverso: vê a chave pública Q e o ponto base G, e tenta descobrir quantas vezes G foi somado para chegar a tal resultado.

Este é o famoso **Problema do Logaritmo Discreto Elíptico, ou ECDLP**. Não existe atalho conhecido e a única forma é ir somando G uma a uma, testando cada possibilidade. Com números normais, seria como tentar adivinhar a combinação de um cofre com mais combinações do que átomos no universo observável. Mesmo com os supercomputadores mais avançados do mundo a calcular durante milhões de anos, é uma tarefa impossível. É esta assimetria brutal — fácil multiplicar (somar muitas vezes), impossível dividir (descobrir o número de somas) — que garante que só quem conhece a chave privada consegue gastar os bitcoins.

Qualquer pessoa pode verificar a matemática do Bitcoin, mas ninguém consegue quebrá-la. É a base de toda a segurança do sistema criado por Satoshi Nakamoto: um cadeado que qualquer um pode ver, mas só o dono da chave consegue abrir.

No fundo, o protocolo define um ponto base G de ordem enorme *n* (também ≈ 2^{256}). A **chave privada** é simplesmente um número aleatório *k* entre 1 e n−1. **A chave pública Q** calcula-se multiplicando G por *k* (ou seja, somando G a si próprio k vezes). Fazer *k* × G = Q é fácil e rápido, porém o inverso — sabendo Q e G, descobrir *k* — é tentar solucionar o ECDLP, um problema que, com os melhores algoritmos conhecidos (Pollard's rho), exigiria cerca de 2^{128} operações mesmo em supercomputadores.

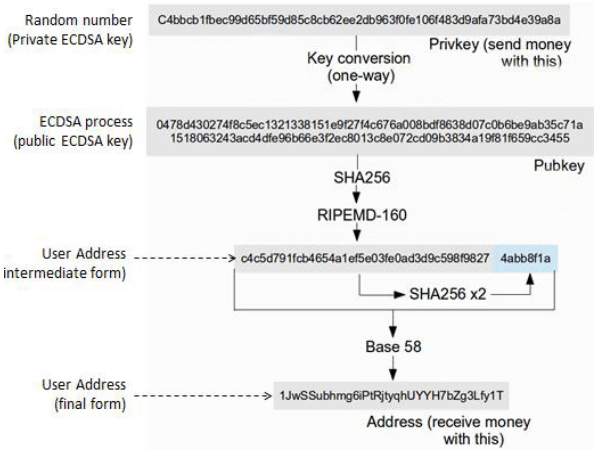
3.4 Um exemplo pedagógico com números pequenos

Para percebermos o conceito sem nos perdermos em números gigantes, vamos usar uma curva minúscula e um ponto base G = (13,9) de ordem pequena.
Se a chave privada for *k* = 7, a chave pública será Q = 7 × G = (15,11). Numa curva destas é fácil descobrir k por força bruta, mas na curva real do Bitcoin o grupo tem ≈ 2^{256} elementos, é como tentar acertar num grão de areia específico em todas as praias do planeta.

4. Como se gera um endereço Bitcoin (passo a passo)

A geração de um endereço Bitcoin é um processo unidirecional e determinista que transforma uma chave privada aleatória num identificador público curto e seguro. Cada passo aplica transformações criptográficas irreversíveis, garantindo que a partir da chave privada, qualquer pessoa possa obter, exatamente, o mesmo endereço. E a partir do endereço, é impossível recuperar a chave privada (devido ao ECDLP e à resistência à pré-imagem das hashes).

Erros de digitação são detetados, automaticamente, através do checksum e o formato final é compacto e legível por humanos (Base58Check).



1. Escolhe-se 256 bits verdadeiramente aleatórios, a chave privada *k*.
2. Calcula-se Q = *k* × G (chave pública, geralmente 65 bytes).
3. Aplica-se SHA-256 à chave pública.
4. Aplica-se RIPEMD-160 ao resultado → 20 bytes (160 bits – a exceção).
5. Adiciona-se o byte de versão (0x00 na rede principal) e um checksum de 4 bytes (para detetar erros de digitação).
6. Codifica-se tudo em Base58Check → endereço que começa por 1 ou 3
(Ex: 1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa).

O duplo hash torna colisões praticamente impossíveis (mesmo com o paradoxo dos aniversários, seriam necessárias ≈ 2^{80} tentativas).

5. Assinaturas Digitais ECDSA: provar posse sem revelar o segredo

O protocolo ECDSA (Elliptic Curve Digital Signature Algorithm) permite que o detentor de uma chave privada autentique o controlo sobre fundos associados a um endereço Bitcoin, sem nunca expor o segredo subjacente. Quando um utilizador pretende gastar bitcoins, constrói uma transação que referencia saídas anteriores não gastas (UTXOs) e especifica novos destinatários; para desbloquear cada entrada, deve anexar uma assinatura digital que prove conhecimento da chave privada correspondente ao hash da chave pública embutido no script de bloqueio.

O processo inicia-se com a computação de um hash da transação (excluindo as assinaturas, para evitar circularidade), designado por *z*, que serve como mensagem a assinar. Em seguida, gera-se um nonce efémero k_e — um inteiro aleatório único e secreto entre 1 e n−1 (ordem da curva) — cuja qualidade criptográfica é crítica: qualquer previsibilidade ou reutilização compromete, imediatamente, a chave privada, como demonstrado em incidentes históricos de carteiras Android em 2013.

A assinatura propriamente dita é gerada em duas etapas principais:

1: multiplica-se o nonce pelo ponto base G, obtendo-se o ponto $R = k_e \cdot G$; a coordenada x de R é reduzida módulo n para produzir o valor r (se r = 0, rejeita-se e regenera-se k_e).
2: calcula-se $s = k_e^{-1} \cdot (z + r \cdot k) \bmod n$, onde k^{-1} é o inverso modular do nonce e *k* a chave privada; novamente, se s = 0, repete-se o processo. O par ordenado (r, s) constitui a assinatura compacta (tipicamente 70–72 bytes em DER), que é inserida no scriptSig da transação.

Esta construção garante que apenas quem conhece *k* pode produzir um par válido, pois a equação subjacente explora a dificuldade do ECDLP: o verificador, sem acesso a k_e ou k, apenas confirma a consistência matemática.

A verificação é efetuada por qualquer nó da rede de forma determinista e eficiente. Recebida a transação, extrai-se z (o mesmo hash), a chave pública Q do endereço e a assinatura (r, s). Computam-se dois escalares: $u_1 = z \cdot s^{-1} \bmod n$ e $u_2 = r \cdot s^{-1} \bmod n$. Em seguida, calcula-se o ponto $P = u_1 \cdot G + u_2 \cdot Q$ e verifica-se se a coordenada x de P, reduzida módulo n, coincide com r. Se a igualdade se verificar, a assinatura é aceite e a transação considerada válida; caso contrário, é rejeitada.

Este procedimento não requer aleatoriedade nem segredos por parte do verificador, dependendo apenas da integridade matemática do grupo da curva e da resistência à falsificação do esquema — propriedades exaustivamente validadas desde a padronização NIST em 2000 e amplamente adotadas em protocolos como TLS e SSH.

6. Proof-of-Work e Árvores de Merkle: Matemática do consenso e verificação eficiente

O mecanismo de Proof-of-Work (PoW) transforma o consenso distribuído num problema de otimização estocástica sobre a função SHA-256 aplicada duas vezes (SHA-256d) ao cabeçalho de bloco de 80 bytes, que inclui o hash do bloco anterior, a raiz Merkle das transações, o timestamp, o campo de dificuldade compactada (bits) e o nonce de 32 bits.

Matematicamente, o minerador deve encontrar um nonce tal que

$$H = \text{SHA-256}(\text{SHA-256}(\text{cabeçalho})) < T$$

onde o target T é derivado da dificuldade D por

$$T = \frac{2^{224}}{D} \quad \text{(aproximadamente)}$$

impondo uma probabilidade de sucesso por tentativa $p \approx \frac{T}{2^{32}}$.

Com hashrate global em novembro de 2025 superior a 1 000 EH/s, o processo segue uma distribuição de Poisson com taxa média de 1 bloco a cada 600 segundos, ajustada a cada 2016 blocos pela fórmula

$$D_{n+1} = D_n \times \min\left(4, \max\left(\frac{1}{4}, \frac{t_{\text{real}}}{2016 \times 600}\right)\right)$$

garantindo estabilidade temporal apesar de variações exponenciais no poder computacional (computação quântica).

Esta construção torna a reversão de blocos um problema de recuperação de energia gasta, com custo de ataque de 51 % estimado em $O(D \times t)$ onde t é o número de blocos a reescrever, explorando a irreversibilidade prática da função de hash unidirecional. A vantagem matemática aqui é a simplicidade do teste: qualquer telemóvel pode verificar um bloco em milissegundos, bastando para isso dois hashes SHA-256 e uma comparação de 256 bits, sem necessidade de hardware especializado, permitindo que carteiras móveis confirmem pagamentos com a mesma segurança que um nó completo.

Integrada ao PoW, a árvore de Merkle organiza as transações do bloco numa estrutura binária completa onde cada nó *não-folha* é o hash SHA-256d dos seus filhos, culminando na raiz incluída no cabeçalho. Esta árvore permite provas de inclusão com complexidade logarítmica:

Para *N* transações, uma prova requer apenas $\lceil \log_2 N \rceil$ hashes (tipicamente 10–14 em blocos reais), reduzindo a verificação SPV (Simplified Payment Verification) de $O(N)$ para $O(\log N)$ em espaço e tempo.

A raiz Merkle vincula, probabilisticamente, todas as transações ao PoW, de modo que qualquer alteração numa *folha* propaga um efeito avalanche através da árvore, invalidando o hash do cabeçalho e exigindo novo trabalho — unindo assim a integridade local das transações à dificuldade global do consenso num único framework matemático rigoroso e verificável.

Esta eficiência logarítmica é crucial para dispositivos móveis: uma carteira SPV descarrega apenas o cabeçalho (80 bytes) e o caminho Merkle (~300–400 bytes), totalizando menos de 1 KB por bloco, em vez de centenas de MB do bloco completo, permitindo verificação segura em redes 4G/5G com bateria limitada e armazenamento reduzido, enquanto mantém a confiança total na matemática do protocolo.

7. Resistência a computadores quânticos: o que realmente muda?

Os computadores quânticos são a única ameaça conhecida que poderia, em teoria, quebrar partes importantes da criptografia do Bitcoin. O algoritmo de Shor, executado num computador quântico suficientemente grande e estável, resolveria o ECDLP em tempo polinomial, isto é, descobriria chaves privadas a partir de chaves públicas em horas ou dias em vez de milhões de anos. Isso afetaria diretamente o ECDSA e, portanto, todas as carteiras cujas chaves públicas já tenham sido reveladas (P2PKH antigas e endereços reutilizados).

O algoritmo de Grover, por outro lado, aceleraria a busca de nonces no Proof-of-Work, reduzindo a dificuldade efectiva de 2^{256} para aproximadamente 2^{128} — ainda um número astronomicamente grande e, com consumo energético realista, completamente inviável durante séculos.

A comunidade Bitcoin acompanha de perto o progresso quântico. Estima-se que ainda faltem décadas para computadores quânticos úteis em grande escala, tempo mais do que suficiente para fazer uma soft fork que migre todas as assinaturas para algoritmos pós-quânticos tais como Lattice-based ou Hash-based signatures, já em estudo.

O Proof-of-Work, esse, sobreviveria, largamente, intacto uma vez que o algoritmo de Grover apenas oferece uma aceleração quadrática — procura exaustiva em 2^{256} possibilidades para cerca de 2^{128} operações, o que, mesmo com milhares de qubits estáveis e consumo energético equivalente a nações inteiras, manteria o custo de mineração ou de um ataque de 51% proibitivamente elevado por séculos. Por outro lado, o ajuste dinâmico da dificuldade continuaria a escalar o alvo proporcionalmente ao hashrate quântico, preservando o intervalo de 10 minutos entre blocos e a segurança económica do protocolo sem necessidade de alterações fundamentais.

Conclusão

O Bitcoin não funciona por magia nem por "fé fundamentalista". Funciona porque substituiu a confiança humana por problemas matemáticos que ninguém, até hoje, conseguiu resolver de forma eficiente: o logaritmo discreto elíptico na curva secp256k1, a resistência à pré-imagem e colisão do SHA-256 e a pura brute-force do Proof-of-Work.

Compreender estes fundamentos permite separar riscos reais dos medos infundados!

No fundo, o Bitcoin é matemática pura em ação, ao serviço do Homem e isso é o que o torna tão poderoso. Cada transação que fazemos e cada bitcoin que guardamos encontram-se protegidos por cálculos que ninguém, nem governos, nem hackers, nem supercomputadores conseguem quebrar sem gastar uma energia absurda ou esperar milhões de anos.

É como se tivéssemos um cofre que qualquer pessoa pode ver, mas só o proprietário consegue abrir, porque a chave está escondida num problema que a própria natureza dos cálculos torna impossível de resolver.

É emocionante pensar que, pela primeira vez na história, temos dinheiro que não depende de bancos, de promessas ou de confiança cega, mas depende só de números, de lógica, de regras que qualquer um pode verificar no seu telemóvel.

Por essa razão, enquanto a matemática estiver do "nosso" lado, o Bitcoin continua a ser o ativo mais seguro, justo e livre que alguma vez foi criado. E isso, sim, dá vontade de acreditar no futuro.