**Capítulo 18  
Privacidade em Blockchain**

Neste capítulo, abordaremos a privacidade, que é um tópico importante. A privacidade de dados ou privacidade da informação não é um conceito novo. A privacidade de dados refere-se à proteção de informações pessoais e sensíveis e aos direitos dos indivíduos de controlar como suas informações pessoais são coletadas, armazenadas, usadas e compartilhadas. Embora a privacidade sempre tenha sido importante, com o advento da internet e dos serviços online, proteger as informações pessoais tornou-se ainda mais crucial do que nunca. No mundo digital estabelecido, como a internet e outros serviços online, existem muitas leis e regulamentações, juntamente com políticas rigorosas de segurança da informação e controles técnicos que protegem os usuários.

No entanto, no mundo blockchain, há uma clara falta de tal rigor até o momento. Alcançar o mesmo nível de segurança que o mundo digital típico desfruta ainda é uma questão que precisa ser abordada. Neste capítulo, veremos o que é privacidade no contexto do blockchain e como podemos introduzi-la usando diferentes protocolos. Ao longo do caminho, examinaremos os seguintes tópicos:

* Privacidade e seus tipos
* Protocolos de Camada 0, Camada 1 e Camada 2 para privacidade em blockchain
* Provas de conhecimento zero, seus vários tipos, esquemas de compromisso polinomial e protocolos relevantes
* Um exemplo prático

Então, vamos começar; primeiro discutiremos a privacidade, por que ela é importante e seus vários tipos.

**Privacidade**

A privacidade no blockchain pode ser dividida em duas categorias principais com base no tipo de serviço requerido. Essas categorias são o anonimato dos usuários e a confidencialidade das transações. O anonimato está relacionado ao ocultamento da identidade do remetente ou do destinatário, enquanto a confidencialidade aborda os requisitos de ocultar os valores das transações.

A razão fundamental pela qual os blockchains não preservam a privacidade é que cada transação em um blockchain precisa ser verificada e executada por cada participante na rede. Embora essa propriedade de execução de todas as transações por todos os nós conceda aos blockchains sua poderosa propriedade de integridade e garantia de execução, isso também é uma fraqueza.

Essa fraqueza é resultado da transparência inerente nos blockchains. Como todos os dados das transações, incluindo detalhes da conta, entradas, saídas e estados, são visíveis para qualquer pessoa no blockchain, a privacidade não pode ser preservada.

Uma solução que vem à mente é que poderíamos, de alguma forma, criptografar os dados, mas se os valores estiverem ocultos, então as transações não poderão ser verificadas. O requisito fundamental é, de algum modo, combinar verificabilidade pública e confidencialidade, de modo que, mesmo que os dados estejam criptografados de alguma forma, ainda assim possam ser verificados publicamente.

Vamos agora explicar anonimato e confidencialidade, antes de aprofundarmos em o que podemos fazer para resolver esse problema.

**Anonimato**

O anonimato é desejável em situações em que a identidade dos usuários precisa ser ocultada dos outros participantes em uma rede. Isso pode se dever a exigências regulatórias, exigências empresariais, ou simplesmente à natureza sensível das transações. Por exemplo, em uma rede empresarial, pode ser desejável ocultar as negociações entre diferentes participantes da rede de outros concorrentes, para evitar fricções ou vantagem competitiva injustificada.

As propriedades de não vinculabilidade (*unlinkability*) e ser não rastreável (*untraceable*) são usadas para alcançar o anonimato. A propriedade de não rastreabilidade nos permite ocultar o rastro de uma transação de uma parte para outra em uma rede. Por outro lado, a não vinculabilidade significa que um observador é incapaz de deduzir a ligação entre transações e seus participantes, ou os relacionamentos entre os participantes das transações (remetentes e destinatários). Isso significa que, se uma terceira parte observar o fluxo de transações, ela não conseguirá ver quais entidades estão envolvidas em transações com quais outras entidades.

Existem vários blockchains que suportam privacidade de forma inerente e que são construídos com essa funcionalidade no projeto do sistema. O mais popular é o Zcash, que usa provas de conhecimento zero (*zero-knowledge proofs – ZKPs*) para alcançar o anonimato. Outros exemplos incluem Monero (<https://web.getmonero.org>), que utiliza assinaturas em anel (*ring signatures*) para fornecer serviços de anonimato.

**Confidencialidade**

A confidencialidade é um requisito absoluto em muitas indústrias, como finanças, direito e saúde. Da mesma forma, a privacidade das transações é uma propriedade altamente desejada nos blockchains. No entanto, devido à sua própria natureza, especialmente nos blockchains públicos, tudo é transparente, o que inibe seu uso em várias indústrias onde a privacidade é de importância primordial, como finanças, saúde e muitas outras.

Quando pensamos em confidencialidade, podemos dividi-la em dois requisitos opcionais adicionais: privacidade condicional e divulgação seletiva. Privacidade condicional significa que um sistema deve ter a capacidade de tornar os dados visíveis condicionalmente para uma terceira parte, como auditores, mas manter os dados ocultos de todas as outras partes, exceto daquelas que têm conhecimento das transações. Note que a privacidade incondicional também não é boa na prática, porque, se tudo estiver oculto de todos, isso pode permitir atividades criminosas como lavagem de dinheiro e financiamento ao terrorismo, sem que ninguém saiba disso. Portanto, é necessário, especialmente nas aplicações financeiras de blockchain, garantir que, em alguns casos, uma terceira parte como um regulador ou auditor possa acessar e visualizar os dados para assegurar a conformidade com os requisitos regulatórios.

O outro tipo de confidencialidade é a **divulgação seletiva**, onde um sistema deve ter a capacidade de compartilhar seletivamente alguma parte dos dados, como apenas a idade, a partir de um conjunto de dados maior que contém informações pessoais. Uma variação disso é chamada de **prova de intervalo** (*range proof*), que é a capacidade de provar que alguns dados estão dentro de um intervalo, por exemplo, minha capacidade de provar que tenho mais de 18 anos de idade ou que meu salário está entre $10.000 e $20.000.

Agora veremos algumas técnicas que podemos usar para alcançar privacidade.

**Técnicas para alcançar privacidade**

Como o blockchain é um livro-razão público de todas as transações e está abertamente disponível, torna-se fácil analisá-lo. Quando combinado com análises de tráfego, as transações podem ser vinculadas de volta aos seus endereços IP de origem, revelando possivelmente o originador de uma transação. Esta é uma grande preocupação do ponto de vista da privacidade.

No domínio do Bitcoin, é uma prática recomendada e comum gerar um novo endereço para cada transação, o que permite algum nível de não vinculabilidade. No entanto, isso não é suficiente, e várias técnicas foram desenvolvidas e usadas com sucesso para rastrear o fluxo de transações através da rede e vinculá-las de volta ao seu originador. Essas técnicas analisam blockchains usando grafos de transação, grafos de endereços e grafos de entidades, os quais facilitam a vinculação de usuários às transações, levantando assim preocupações de privacidade.

Essas técnicas baseiam-se na ideia de que cada transação na rede Bitcoin (ou em qualquer blockchain público) é registrada publicamente e pode ser vinculada a outras transações, endereços e entidades. Por exemplo, a análise de grafo de transações envolve a criação de um grafo de todas as transações e a identificação de padrões de transações entre endereços. A análise de grafo de endereços envolve a criação de um grafo de todos os endereços e suas transações associadas. Esse tipo de análise pode ajudar a identificar os relacionamentos entre diferentes endereços e o fluxo de fundos entre eles.

As técnicas mencionadas anteriormente podem ser ainda enriquecidas usando informações publicamente disponíveis (por exemplo, endereços Bitcoin de usuários em fóruns públicos da internet) sobre transações e vinculando-as aos usuários reais. Existem analisadores de blocos (*block parsers*) de código aberto disponíveis que podem ser usados para extrair informações de transações, saldos e scripts do banco de dados da blockchain.

Nesta seção, descreveremos diferentes técnicas para alcançar privacidade, incluindo anonimato e confidencialidade.

Assim como soluções de escalabilidade, também podemos dividir as soluções de privacidade em três categorias, com base na camada dentro da pilha de arquitetura do blockchain na qual operam:

* **Métodos de Camada 0**, ou métodos de camada de rede, onde o mecanismo para alcançar privacidade opera no nível da rede.
* **Métodos de Camada 1**, também chamados de métodos *on-chain*, onde o próprio protocolo do blockchain é aprimorado para alcançar privacidade.
* **Métodos de Camada 2**, ou métodos *off-chain*, onde mecanismos que existem fora do blockchain principal são usados para alcançar privacidade no blockchain.

**Camada 0**

Essas soluções são métodos no nível da camada de rede que fornecem anonimato utilizando **TOR** e **I2P**. Esses métodos nos permitem ocultar as identidades das partes envolvidas.

**Tor**

Tor, *The Onion Router*, é um software que permite comunicações anônimas. Mais informações sobre o Tor estão disponíveis aqui: <https://www.torproject.org>. Podemos usar o Tor para permitir comunicação anônima em redes blockchain de criptomoedas. Tor é uma escolha comum para permitir comunicação anônima e tem sido usado em Monero, Verge e no Bitcoin para fornecer anonimato.

Para Bitcoin, veja: <https://en.bitcoin.it/wiki/Tor>. O Monero também pode ser executado com o Tor (<https://web.getmonero.org/>). Verge é outro exemplo de criptomoeda que faz uso do Tor para ofuscação de IP.

**I2P**

**I2P**, *Invisible Internet Project*, é uma rede anônima construída sobre a internet. Ela permite comunicação peer-to-peer resistente à censura. É usada no Monero para fornecer serviços de anonimato. Mais informações sobre o I2P estão disponíveis aqui: <https://geti2p.net/en/>. Monero e Zcash suportam transações anônimas. O Monero faz uso de assinaturas em anel, endereços furtivos e transações confidenciais, enquanto o Zcash usa zk-SNARKs.

Como muitas das técnicas descritas nas seções seguintes podem ser usadas nas Camadas 1 e 2 — por exemplo, ZKPs — não estamos distinguindo entre essas camadas. Em vez disso, listaremos apenas essas soluções e discutiremos para que elas podem ser usadas. Além disso, as soluções para anonimato e confidencialidade também se sobrepõem, e as técnicas usadas para alcançar confidencialidade também são mais ou menos aplicáveis para alcançar anonimato. No entanto, separamos a Camada 0, a camada de rede, e descreveremos algumas das abordagens no nível de rede para alcançar anonimato, antes de apresentar uma variedade de mecanismos de privacidade aplicáveis às Camadas 1 e 2.

**Ofuscação por Indistinguibilidade**

Essa técnica criptográfica pode servir como uma solução milagrosa para todos os problemas de privacidade e confidencialidade nos blockchains, mas a tecnologia ainda não está pronta para implementações em produção. **Ofuscação por indistinguibilidade** (*Indistinguishability Obfuscation – IO*) permite a ofuscação de código, que é um tópico de pesquisa muito ativo em criptografia. Se aplicada aos blockchains, a IO pode servir como um mecanismo de ofuscação inquebrável que transforma contratos inteligentes em uma caixa-preta onde o comportamento do código ofuscado é indistinguível. Em termos mais simples, a funcionalidade interna do contrato inteligente é totalmente oculta.

A ideia chave por trás da IO é o que os pesquisadores chamam de “*quebra-cabeça multilinear*”, que basicamente ofusca o código do programa misturando-o com elementos aleatórios, e se o programa for executado conforme pretendido, produzirá a saída esperada. No entanto, qualquer outra maneira de executá-lo fará com que o programa produza dados aleatórios e sem sentido.

Em outras palavras, a IO torna computacionalmente inviável para um atacante distinguir entre duas execuções diferentes do programa, mesmo que o atacante tenha acesso completo ao código do programa e ao comportamento de entrada/saída.

Em outras palavras, a IO fornece uma maneira de proteger a privacidade da implementação de um programa, tornando impossível para um atacante determinar os detalhes exatos de como o programa é executado ou quais entradas ele recebeu.

Este artigo de pesquisa está disponível em: <https://eprint.iacr.org/2013/451.pdf>.

**Criptografia homomórfica**

Este tipo de criptografia permite que operações sejam realizadas sobre dados criptografados. Imagine um cenário em que os dados são enviados para um servidor em nuvem para processamento. O servidor os processa e retorna a saída sem saber nada sobre os dados que foram processados. Esta também é uma área madura para pesquisa, e a **criptografia homomórfica totalmente funcional**, que permite todas as operações sobre dados criptografados, ainda não é totalmente implantável em produção; no entanto, já houve progressos significativos neste campo. Uma vez implementada em blockchains, ela pode permitir o processamento de transações criptografadas sem a necessidade de descriptografá-las, o que permitirá, por si só, a privacidade e a confidencialidade das transações.

Por exemplo, os dados armazenados no blockchain podem ser criptografados usando criptografia homomórfica, e cálculos podem ser realizados sobre esses dados sem a necessidade de descriptografá-los, proporcionando assim um serviço de privacidade sobre blockchains. Este conceito também foi implementado em um projeto chamado **Enigma**, que está disponível online em: <https://www.media.mit.edu/projects/enigma/overview/>, pelo Media Lab do MIT. O Enigma é uma rede ponto a ponto que permite a várias partes realizar cálculos sobre dados criptografados sem revelar nada sobre os dados. A pesquisa está disponível em: <https://crypto.stanford.edu/craig/>.

**Computação multipartidária segura**

O conceito de **computação multipartidária segura** (*secure multiparty computation*) não é novo e é baseado na noção de que os dados são divididos em múltiplas partições entre as partes participantes, sob um mecanismo de compartilhamento secreto, que então realiza o processamento real sobre os dados sem a necessidade de reconstruir os dados em uma única máquina.

A saída produzida após o processamento também é compartilhada entre as partes. Várias partes realizam o cálculo mutuamente sem revelar suas entradas. Somente a saída da computação é revelada.

Uma plataforma de computação multipartidária segura chamada **Enigma** foi proposta em 2015. O artigo está disponível aqui: <https://web.media.mit.edu/~guyzys/data/enigma_full.pdf>.

**Confidencialidade assistida por hardware confiável**

Plataformas de computação confiável podem ser usadas para fornecer um mecanismo pelo qual a confidencialidade de uma transação pode ser alcançada em um blockchain, por exemplo, usando **Intel Software Guard Extensions (SGX)**, que permite que código seja executado em um ambiente protegido por hardware chamado *enclave*. Uma vez que o código seja executado com sucesso no enclave isolado, ele pode produzir uma prova, chamada *quote*, que pode ser atestada pelos servidores em nuvem da Intel.

Há uma preocupação de que confiar na Intel resulte em algum nível de centralização e não esteja em linha com o verdadeiro espírito da tecnologia blockchain. No entanto, essa solução tem seus méritos e, na realidade, muitas plataformas já usam chips da Intel, então confiar na Intel pode ser aceitável por alguns em certos casos.

Se essa tecnologia for aplicada a contratos inteligentes, então, uma vez que um nó tenha executado o contrato inteligente, ele pode produzir uma prova de correção, demonstrando a execução bem-sucedida, e os outros nós apenas precisarão verificá-la. Essa ideia pode ser estendida ainda mais usando qualquer **Ambiente de Execução Confiável (TEE – Trusted Execution Environment)**, que pode fornecer a mesma funcionalidade de um enclave e está disponível até mesmo em dispositivos móveis com Comunicação por Campo de Proximidade (NFC – Near Field Communication) e um elemento seguro. Por exemplo, **Ekiden** é uma plataforma que faz uso do TEE do Intel SGX para executar contratos inteligentes enquanto preserva a confidencialidade. Mais informações sobre o Ekiden estão disponíveis aqui: <https://arxiv.org/pdf/1804.05141>.

**Protocolos de mistura**

Um **protocolo de mistura**, ou *mixer*, é um serviço que permite aos usuários preservar sua privacidade misturando suas moedas com as de outros usuários.

Esses esquemas são usados para fornecer anonimato às transações de criptomoedas. Nesse modelo, é usado um provedor de serviço de mistura (um intermediário ou uma carteira compartilhada). Os usuários enviam moedas para essa carteira compartilhada como um depósito, e então a carteira compartilhada pode enviar outras moedas (do mesmo valor depositado por outros usuários) ao destino. Os usuários também podem receber moedas que foram enviadas por outros por meio desse intermediário. Dessa forma, o vínculo entre saídas e entradas não existe mais, e a análise do grafo de transações não será capaz de revelar a relação real entre remetentes e destinatários:

<IMAGEM>

**Coinjoin** é um exemplo de protocolo de mistura, onde duas transações são unidas para formar uma única transação, mantendo as entradas e saídas inalteradas. A ideia central por trás do Coinjoin é construir uma transação compartilhada que seja assinada por todos os participantes. Essa técnica melhora a privacidade de todos os participantes envolvidos nas transações.

Coinjoin é uma técnica usada para anonimizar transações em Bitcoin misturando-as de forma interativa. A ideia baseia-se na formação de uma única transação a partir de múltiplas entidades, sem causar nenhuma alteração nas entradas e saídas. Ela remove o vínculo direto entre remetentes e destinatários, o que significa que um único endereço não pode mais ser associado a transações, o que poderia levar à identificação de usuários.

O Coinjoin requer cooperação entre várias partes dispostas a criar uma única transação misturando pagamentos. Portanto, deve-se observar que, se qualquer participante no esquema Coinjoin não cumprir o compromisso feito de cooperar na criação de uma transação única — ao não assinar as transações conforme exigido — isso pode resultar em um ataque de negação de serviço (DoS).

Nesse protocolo, não há necessidade de uma terceira parte confiável única. Este conceito é diferente de um serviço de mistura, que atua como uma terceira parte confiável ou intermediária entre os usuários de Bitcoin e permite o embaralhamento das transações. Esse embaralhamento de transações resulta na prevenção de rastreamento e vinculação dos pagamentos a um usuário específico.

Vários serviços de mistura de terceiros também estão disponíveis, mas, se um serviço for centralizado, ele representa a ameaça de rastreamento do mapeamento entre remetentes e destinatários. Isso porque o serviço de mistura conhece todas as entradas e saídas. Além disso, misturadores totalmente centralizados apresentam o risco de que os administradores do serviço roubem as moedas.

Diversos serviços, com graus variados de complexidade, como **CoinShuffle**, **Coinmux** e **Darksend** no Dashcoin, estão disponíveis, baseando-se na ideia de transações Coinjoin (mistura). O **CoinShuffle** é uma alternativa descentralizada aos serviços de mistura tradicionais, pois não requer uma terceira parte confiável.

No entanto, os esquemas baseados em Coinjoin possuem algumas fraquezas, mais notavelmente a possibilidade de lançar um ataque DoS por usuários que se comprometeram a assinar as transações inicialmente, mas agora não fornecem sua assinatura, atrasando ou interrompendo completamente as transações conjuntas.

**CoinSwap**

**CoinSwap** é um mecanismo de privacidade baseado na ideia de trocas atômicas (*atomic swaps*). As trocas atômicas permitem que duas partes troquem moedas sem exigir uma terceira parte confiável. O CoinSwap também pode ser usado para trocas entre blockchains diferentes (*cross-chain swaps*). O CoinSwap funciona utilizando uma terceira parte no fluxo da transação e também requer canais de comunicação privados entre todas as partes. Dessa forma, os endereços do remetente e do destinatário não podem ser vinculados. Essa terceira parte recebe fundos do remetente e os envia ao destinatário.

A técnica aqui é que a terceira parte paga os fundos ao destinatário usando uma fonte de fundos totalmente diferente, desconectando assim o vínculo entre o remetente e o destinatário. Essa desconexão resulta em uma transação não vinculável e, portanto, proporciona privacidade. O remetente usa transações multiassinadas (normalmente 2 de 2 — remetente e terceira parte) para permitir que as transações sejam gastas, enquanto o destinatário também requer transações multiassinadas (normalmente 2 de 2 — destinatário e terceira parte) para que as transações sejam gastas. Em outras palavras, o remetente e a terceira parte assinarão a saída da transação para enviar o Bitcoin à terceira parte, enquanto o destinatário e a terceira parte assinarão a saída da transação para enviar o Bitcoin ao destinatário.

O CoinSwap usa transações bloqueadas por hash (*hash-locked transactions*) em que uma pré-imagem do hash é necessária para desbloquear a transação. O uso de transações bloqueadas por hash impede que a terceira parte roube o Bitcoin. Mais informações sobre o CoinSwap podem ser encontradas no link a seguir, onde ele foi originalmente proposto por Gregory Maxwell: <https://bitcointalk.org/index.php?topic=321228>.

**TumbleBit**

O protocolo **TumbleBit** foi introduzido em 2016. O TumbleBit é totalmente compatível com o protocolo do Bitcoin. Trata-se de um protocolo de pagamentos fora da cadeia (*off-chain*), anônimo, rápido e com *unlinkability*, que permite que as partes transfiram fundos por meio de uma terceira parte ou intermediário não confiável, chamado de *tumbler*. Neste protocolo, até mesmo o *tumbler* é incapaz de desanonimizar os pagadores e recebedores envolvidos em um pagamento. Ele envolve o uso de dois protocolos de troca justa que impedem qualquer atividade maliciosa, como trapaça por participantes ou pelo *tumbler*. O TumbleBit se baseia em um protocolo chamado **solucionador de quebra-cabeça RSA** (*RSA puzzle solver*), que permite a um pagador fazer pagamentos ao *tumbler*. A menos que o *tumbler* resolva esse quebra-cabeça RSA, ele não pode reivindicar nenhum Bitcoin pago pelo pagador. Outro protocolo de troca justa, chamado **protocolo de promessa de quebra-cabeça** (*puzzle-promise protocol*), é usado entre o *tumbler* e o recebedor para reivindicar o pagamento.

O TumbleBit consiste em três fases, conforme listado aqui:

1. **Fase de depósito em garantia (escrow)**, onde todos os canais de pagamento são configurados.
2. **Fase de pagamentos**, onde os pagadores transferem fundos.
3. **Fase de retirada (cash-out)**, onde pagadores e recebedores encerram os canais de pagamento.

Mais informações sobre o TumbleBit podem ser encontradas aqui: <https://eprint.iacr.org/2016/575.pdf>. Uma implementação de prova de conceito do TumbleBit está disponível aqui: <https://github.com/BUSEC/TumbleBit>.

Uma inovação recente é a introdução do **TumbleBit++**, que é um protocolo de mistura aprimorado baseado no TumbleBit. Ele fornece anonimato e confidencialidade dos valores das transações combinando transações confidenciais e um *hub* de pagamento anônimo centralizado e não confiável. Mais informações sobre o TumbleBit++ estão disponíveis aqui: <https://link.springer.com/chapter/10.1007/978-3-030-31919-9_21>.

Agora discutiremos o **Dandelion**, que fornece anonimato inerente e é implementado por meio da reformulação da camada de rede do protocolo do Bitcoin.

**Dandelion**

Além das abordagens mencionadas anteriormente, uma proposta recente chamada **Dandelion** também foi apresentada. O Dandelion (sua versão aprimorada é chamada **Dandelion++**) é uma proposta que visa tornar as transações na rede do Bitcoin intraceáveis. Este protocolo permitirá que transações anônimas ocorram na rede Bitcoin, em oposição a transações pseudônimas, onde um adversário, usando métodos de análise de rede, pode rastrear a transação até seu nó de origem. Consequentemente, o adversário pode então descobrir o endereço IP original do remetente da transação.

Podemos dizer que o Dandelion é um mecanismo para fornecer anonimato inerente às transações em Bitcoin porque, uma vez implementado, a camada P2P do protocolo de rede do Bitcoin será modificada de tal forma que rastrear transações até seu nó de origem e IP se tornará extremamente difícil.

A desanonimização de usuários de Bitcoin é um problema conhecido, e diversos artigos de pesquisa estão disponíveis sobre esse tema. Por exemplo, o artigo *Deanonymization of clients in Bitcoin P2P network* está disponível em: <https://arxiv.org/pdf/1405.7418.pdf>.

Uma proposta de melhoria do Bitcoin está disponível em:  
<https://github.com/bitcoin/bips/blob/master/bip-0156.mediawiki>.  
O artigo original da proposta do Dandelion está disponível em:  
<https://arxiv.org/pdf/1701.04439.pdf>.  
O artigo de pesquisa sobre o Dandelion++ está disponível em:  
<https://arxiv.org/pdf/1805.11060.pdf>.

Na rede Bitcoin, o mecanismo de propagação epidêmica (*epidemic flooding mechanism*, também conhecido como **protocolo de fofoca – Gossip protocol**) é utilizado para propagação de transações. Sobre esse mecanismo, há um método relativamente eficaz chamado **difusão**, que é utilizado para prover algum nível de anonimato. Nesse protocolo, cada nó introduz atrasos independentes e exponenciais para espalhar transações aos seus vizinhos. Esse esquema resulta na redução da simetria do protocolo epidêmico, o que torna a análise da rede mais difícil e menos confiável. No entanto, esse esquema é previsível e, portanto, não fornece garantias suficientes de anonimato.

O Dandelion propõe um novo mecanismo de roteamento compatível com versões anteriores. Nesse protocolo:

* Primeiro, é construído um grafo de privacidade (*privacy graph*, também chamado de **conjunto de anonimato**). Esta fase é chamada de **fase de construção do grafo privado**. Trata-se de um subgrafo da rede P2P existente do Bitcoin, e cada nó seleciona um subconjunto de seus pares de saída nesta fase. É nesse momento que a linha aleatória de nós é selecionada.
* As mensagens (transações) são então roteadas por esse grafo de privacidade durante a **fase do Caule – Stem phase**. É aqui que a mensagem é propagada por uma linha aleatória de nós.
* Finalmente, há a **fase da Flor – Fluff phase**, onde as mensagens são roteadas (transmitidas) para toda a rede por difusão.

Em resumo, o protocolo Dandelion é composto por uma fase de anonimato e uma fase de propagação. Na fase de anonimato, esse protocolo espalha uma mensagem ao longo de uma linha aleatória por um número aleatório de saltos. Após esse passo, a mensagem é transmitida para toda a rede usando difusão. Com essa combinação de seleção de caminho aleatório e difusão, o protocolo Dandelion fornece garantias de anonimato quase ideais.

Esse protocolo pode ser visualizado utilizando o seguinte diagrama:

<IMAGEM>

**Transações confidenciais**

As transações confidenciais fazem uso de **compromissos de Pedersen** (*Pedersen commitments*) para fornecer confidencialidade. Esquemas de compromisso permitem que um usuário se comprometa com um determinado valor enquanto o mantém em segredo, com a capacidade de revelá-lo posteriormente. Esquemas de compromisso podem ser construídos simplesmente usando funções de hash criptográficas. Duas propriedades precisam ser satisfeitas para projetar um esquema de compromisso: **vinculação** (*binding*) e **ocultação** (*hiding*).

A vinculação garante que o compromissário não consiga alterar o valor escolhido depois de comprometido, enquanto a propriedade de ocultação garante que qualquer adversário não seja capaz de descobrir o valor original ao qual o compromissário se comprometeu.

Um compromisso de Pedersen é um tipo de esquema de compromisso informacionalmente oculto e computacionalmente vinculante, sob a suposição do logaritmo discreto. Compromissos de Pedersen permitem operações adicionais e preservam a propriedade comutativa nos compromissos, o que os torna particularmente úteis para fornecer confidencialidade em transações de Bitcoin. Em outras palavras, eles suportam criptografia homomórfica parcial de valores, o que significa que, usando esquemas de compromisso, podemos ocultar valores de pagamento em uma transação de Bitcoin. Esse conceito já foi implementado no Projeto Elements:  
<https://elementsproject.org/features/confidential-transactions>.

**MimbleWimble**

O esquema **MimbleWimble** foi proposto anonimamente no canal IRC do Bitcoin e desde então ganhou muita popularidade. O MimbleWimble estende a ideia de transações confidenciais e Coinjoin, o que permite a agregação de transações sem exigir nenhuma interatividade. No entanto, ele não suporta o uso da linguagem de scripts do Bitcoin, juntamente com várias outras funcionalidades do protocolo padrão do Bitcoin. Isso o torna incompatível com o protocolo atual do Bitcoin. Portanto, ele pode ser implementado como uma **sidechain** do Bitcoin ou como uma criptomoeda alternativa. No entanto, ele foi implementado no **Litecoin** em maio de 2022.

Esse esquema pode abordar simultaneamente problemas de privacidade e escalabilidade. Os blocos criados usando a técnica MimbleWimble não contêm transações como nos blockchains tradicionais do Bitcoin; em vez disso, esses blocos são compostos por três listas: uma lista de entradas, uma lista de saídas e algo chamado de **excedentes** (*excesses*), que são listas de assinaturas e diferenças entre saídas e entradas. A lista de entradas basicamente faz referência às saídas antigas, e a lista de saídas contém saídas de transações confidenciais.

Os blocos criados usando o esquema MimbleWimble são verificáveis pelos nós utilizando assinaturas, entradas e saídas para assegurar a legitimidade do bloco. Em contraste com o Bitcoin, as saídas das transações no MimbleWimble contêm apenas chaves públicas, e a diferença entre as saídas antigas e novas é assinada por todos os participantes envolvidos nas transações.

**ZkLedger**

Este livro-razão distribuído fornece privacidade de transação, incluindo confidencialidade do valor da transação e dos endereços do remetente e do destinatário. Ele utiliza **provas de conhecimento zero não interativas (NIZKs)** e **protocolos Sigma** para alcançar isso. No entanto, este protocolo não é escalável porque o tamanho das transações cresce linearmente com o número de partes envolvidas.

**Criptografia baseada em atributos**

A **criptografia baseada em atributos (ABE – Attribute-Based Encryption)** é um tipo de criptografia de chave pública que fornece simultaneamente confidencialidade e controle de acesso. A ideia principal por trás deste esquema é que a chave privada de um usuário e o seu texto cifrado dependem dos atributos do usuário, como sua localização, fuso horário ou seu papel na organização. Isso significa que a descriptografia só é possível quando não apenas a chave privada está disponível, mas também os atributos correspondentes estão presentes.

**Assinaturas anônimas**

Assinaturas anônimas são tipos de assinaturas digitais nas quais as assinaturas não revelam a identidade do signatário. Existem principalmente dois esquemas disponíveis para assinaturas anônimas: **assinaturas de grupo** e **assinaturas em anel (ring signatures)**.

As **assinaturas de grupo** permitem que um conjunto de signatários forme um grupo gerenciado por um gerente de grupo. Cada membro do grupo recebe uma chave de assinatura de grupo fornecida pelo gerente. Essa chave de assinatura de grupo permite que cada membro do grupo assine mensagens anonimamente em nome do grupo. O gerente de grupo é capaz de identificar quem assinou a mensagem, enquanto entidades externas não conseguem. Embora as assinaturas de grupo possam funcionar bem, uma das limitações deste esquema é que o gerente do grupo pode identificar os usuários. Esse problema foi abordado com as assinaturas em anel.

As **assinaturas em anel** permitem que um conjunto de signatários forme um grupo (um anel) de membros. Cada membro deste grupo é capaz de assinar mensagens em nome do anel. Diferentemente das assinaturas de grupo, **não há gerente de grupo** nas assinaturas em anel, de modo que **ninguém é capaz de identificar os signatários**.

Embora todas essas técnicas sejam úteis e forneçam muitas propriedades desejáveis de anonimato, também há um problema com o uso malicioso dessa tecnologia. Imagine se o anonimato for utilizado por criminosos envolvidos em lavagem de dinheiro ou na venda de drogas ilegais usando criptomoedas anônimas. Seria quase impossível rastrear atividades ilegais até seu originador se o anonimato absoluto for alcançado. Imagine uma variante do mercado **Silk Road** (<https://en.wikipedia.org/wiki/Silk_Road_(marketplace)>) usando uma criptomoeda totalmente anônima e confidencial. Como isso poderia ser rastreado e interrompido?

Até agora discutimos técnicas criptográficas genéricas e alguns exemplos para permitir a privacidade no blockchain. No entanto, ainda não tocamos no conceito importante de **contratos inteligentes privados**. Blockchains programáveis, começando com o Ethereum e agora com cadeias como Solana e Polkadot, todos suportam contratos inteligentes. É desejável alcançar privacidade também no nível do contrato inteligente, em vez de apenas no nível da transação padrão. Pense em um contrato ERC-20 onde todas as operações, como transferências, verificação de saldo e outras, são realizadas com total privacidade. Existem várias propostas para isso também, algumas das quais discutiremos a seguir.

**Zether**

**Zether** permite um mecanismo de transação privada que oferece confidencialidade e anonimato. Ele é implementado na blockchain do Ethereum, mas, em teoria, pode ser implementado em qualquer blockchain programável. Zether é implementado como um contrato inteligente no qual o ether é depositado e transformado em um token Zether.

Outras propostas incluem:

* **Zexe** (<https://eprint.iacr.org/2018/962.pdf>)
* **PPChain** (<https://ieeexplore.ieee.org/document/9199417>)
* **Hawk** (<https://ieeexplore.ieee.org/document/7546538>)
* **ZKLedger** (<https://dci.mit.edu/zkledger>)

**Privacidade usando protocolos de Camada 2**

A privacidade usando Camada 2 e **canais de estado** (*state channels*) também é possível, simplesmente porque todas as transações são executadas fora da cadeia (*off-chain*), e a blockchain principal não vê as transações, exceto pelo estado final de saída e dados mínimos de transação (representando transações), o que garante anonimato e confidencialidade. No entanto, observe que, geralmente, a privacidade **não é o objetivo principal dos protocolos de Camada 2** — eles se concentram mais em fornecer escalabilidade. Mesmo assim, a Camada 2 pode ser usada para oferecer anonimato e confidencialidade se o conhecimento zero for usado não apenas como uma prova de computação correta (integridade), mas também como uma **medida de confidencialidade**. Isso é realizado no projeto **Aztec** (<https://aztec.network>).

**Gerenciadores de privacidade**

**Gerenciadores de privacidade** fornecem um mecanismo que garante a confidencialidade das transações. Estes são componentes fora da cadeia (*off-chain*) que substituem a carga útil da transação por um índice hash, de tal forma que **apenas os participantes da transação** conseguem encontrar a carga útil criptografada correspondente representada pelo índice hash. Outras partes que não participam da transação simplesmente ignoram o hash. Este conceito foi discutido no **Capítulo 16**, sobre **Blockchain Empresarial** (*Enterprise Blockchain*).

Além da escalabilidade e da privacidade, há vários outros aspectos gerais de segurança que precisam ser abordados no blockchain. Descreveremos esses tópicos de segurança geral no próximo capítulo, **Capítulo 19 – Segurança em Blockchain**.

Existem duas abordagens quando se trata de soluções de escalabilidade e privacidade:

1. A primeira é o desenvolvimento de **linguagens específicas de domínio (DSLs – Domain Specific Languages)**, que podem ser usadas para escrever programas que podem ser verificados por um verificador na Camada 1.
2. A segunda são as **máquinas virtuais de conhecimento zero (ZK VMs)**, que permitem que computações (programas) sejam verificadas na Camada 2.

DSLs exigem que os desenvolvedores aprendam uma nova linguagem, enquanto ZK VMs são compatíveis com ferramentas e linguagens convencionais já em uso. Uma vantagem chave das DSLs é que a prova ZK ou circuito gerado é agnóstico à cadeia subjacente, o que facilita seu uso em qualquer cadeia, desde que escrevamos um verificador para essa cadeia. As ZK VMs, no entanto, têm algum risco de **dependência de fornecedor** (*vendor lock-in*).

**Privacidade usando conhecimento zero**

Quando pensamos em privacidade, a primeira coisa que vem à mente é **criptografia**, e podemos imaginar várias maneiras de alcançar privacidade, as quais descrevemos a seguir:

1. **Podemos possivelmente criptografar transações** para que os dados sejam confidenciais, mas essa abordagem é impraticável por dois motivos:
   * Primeiro, a verificação pública das transações não é possível, já que os valores estão ocultos.
   * Segundo, mesmo que algum esquema de criptografia seja utilizado, o gerenciamento de chaves se torna um grande problema devido ao grande número de chaves exigidas.
2. A **segunda abordagem** é o uso de **esquemas de compromisso**, que são uma solução bastante promissora. Podemos substituir o valor/montante da transação por um compromisso com o valor, depois nos comprometemos com esse valor e o publicamos na blockchain.
   * Novamente, o problema é que, como o valor está oculto, a transação não é verificável publicamente.
   * Além disso, é possível comprometer-se com um valor negativo.
3. A **solução** é anexar **provas de conhecimento zero (ZKPs)** ao compromisso, que provam duas coisas:
   * **Compromisso de intervalo** (*range commitment*), que garante que o número é positivo e está dentro de um intervalo válido.
   * **Compromisso de valor** (*value commitment*), que oculta o valor real da transação.  
     Compromissos de intervalo são mais difíceis de fazer, enquanto os de valor são relativamente mais simples, pois se traduzem em uma equação linear simples, mais fácil de lidar.

Vamos agora explorar **compromissos criptográficos**, um primitivo criptográfico essencial para construir protocolos de conhecimento zero no blockchain.

**Compromissos criptográficos**

Um **esquema de compromisso criptográfico** é um protocolo em duas fases composto por dois algoritmos: um algoritmo de compromisso e um algoritmo de verificação:

* Commit(m, r) -> c
* Verify(m, c, r) -> aceitar ou rejeitar

Na primeira fase, chamada de **fase de compromisso**, o remetente envia ao receptor uma string de compromisso c.  
Alice executa o algoritmo commit para se comprometer com uma mensagem m usando uma aleatoriedade secreta r e produz uma string de compromisso c. Essa string c é enviada para Bob.

Na segunda fase, chamada de **fase de abertura**, o receptor abre o compromisso para verificar se o remetente realmente se comprometeu com a mensagem m e não trapaceou.

Isso é verificado usando o algoritmo verify. Esse algoritmo recebe três valores — mensagem m, compromisso c e aleatoriedade r — e, com base na saída, o verificador **aceita ou rejeita** o compromisso.  
Uma vez que o compromisso é aberto, qualquer pessoa pode verificar publicamente se ele foi aberto corretamente.

Podemos pensar em compromissos criptográficos como um **análogo digital** ou equivalente criptográfico de envelopes lacrados.  
Há duas propriedades de segurança dos compromissos criptográficos: **ocultação** (*hiding*) e **vinculação** (*binding*).

* A propriedade de ocultação garante que a mensagem comprometida não seja revelada antes da fase de abertura.  
  Em outras palavras, o compromisso c não revela nada sobre a mensagem.
* A propriedade de vinculação garante que, uma vez que o valor seja comprometido, ele não possa ser alterado depois.  
  Em outras palavras, uma vez que o remetente se comprometeu com um valor, ele **não pode** abrir o compromisso com uma mensagem diferente m que o algoritmo verify também aceite.

Existem dois tipos de esquemas de compromisso com base na força das propriedades de segurança:

* **Esquemas de compromisso padrão**, que protegem contra um receptor em tempo polinomial probabilístico (*PPT*) e um remetente com poder ilimitado. Tais esquemas são **computacionalmente ocultos** e **informacionalmente vinculantes**.
* **Esquemas de compromisso perfeitos**, que protegem contra um remetente PPT e um receptor com poder ilimitado. Tais esquemas são **informacionalmente ocultos** e **computacionalmente vinculantes**.

Esquemas de compromisso que são **simultaneamente informacionalmente ocultos e vinculantes não existem**.

Há vários modelos de esquemas de compromisso, incluindo esquemas baseados em hash, compromissos de Pedersen e esquemas de compromisso polinomial. Um tipo de esquema de compromisso chamado **compromisso de Pedersen** é muito útil devido às suas propriedades homomórficas. Vamos primeiro ver como o compromisso de Pedersen é construído e depois veremos como sua propriedade homomórfica funciona.

O esquema funciona em três fases: configuração pública (*setup*), compromisso (*commit*) e abertura (*open*).

**Fase de configuração (Setup):**

1. Seja 𝔾 um grupo cíclico finito de ordem prima.
2. Obtenha dois elementos g,h∈𝔾g, h \in 𝔾g,h∈G, onde
   * ggg é um elemento fixo conhecido do grupo (chamado de gerador),
   * e hhh é um elemento aleatório uniformemente escolhido de 𝔾.
3. Seja qqq a ordem de 𝔾, que é exponencialmente grande.

**Fase de compromisso (Commit):**

1. O comprometedor compromete-se com um número m∈Zqm \in \mathbb{Z}\_qm∈Zq​.
2. O comprometedor escolhe um inteiro aleatório z∈Zqz \in \mathbb{Z}\_qz∈Zq​.
3. Envia o compromisso ccc ao receptor, onde c=gm⋅hzc = g^m \cdot h^zc=gm⋅hz.

**Fase de abertura (Open):**

1. Para abrir o compromisso ccc, o comprometedor envia (m,z)(m, z)(m,z).
2. O receptor verifica se c=gm⋅hzc = g^m \cdot h^zc=gm⋅hz.

O compromisso de Pedersen é **perfeitamente oculto** e **computacionalmente vinculante**. Compromissos de Pedersen são **homomórficos aditivos**. Suponha que tenhamos duas strings de compromisso, c1c\_1c1​ e c2c\_2c2​, que são compromissos para as mensagens m1m\_1m1​ e m2m\_2m2​, respectivamente. Se multiplicarmos esses dois compromissos, o resultado será um novo compromisso c3c\_3c3​ para a soma m3m\_3m3​ de m1m\_1m1​ e m2m\_2m2​. Isso significa que, sem conhecer os valores originais m1m\_1m1​ e m2m\_2m2​ (as mensagens comprometidas originalmente), o receptor pode construir um compromisso para a **soma dos valores comprometidos**, ou seja, um compromisso c3c\_3c3​ para m3m\_3m3​, onde m3=m1+m2m\_3 = m\_1 + m\_2m3​=m1​+m2​, e c3c\_3c3​ pode ser aberto como m3m\_3m3​ usando as informações de abertura correspondentes.

Um adversário de tempo polinomial probabilístico (**PPT**) tem acesso a um algoritmo aleatório de tempo polinomial para quebrar a segurança do sistema. Podemos pensar nele como uma entidade com recursos computacionais limitados, capaz de quebrar o sistema apenas se tiver acesso a um algoritmo PPT.  
Algoritmos de tempo polinomial são aqueles que rodam em um tempo “razoável” em função do tamanho da entrada. Mais formalmente, um algoritmo é de tempo polinomial se executa em O(nc)O(n^c)O(nc), onde ccc é uma constante. Um algoritmo PPT é um algoritmo de tempo polinomial **randomizado**, ou seja, que possui acesso a uma fonte de aleatoriedade (por exemplo, lançar moedas).

Compromissos de Pedersen são comumente usados, com algumas modificações, em vários blockchains que preservam a privacidade, incluindo **Monero** e **Zcash**. Observe que **compromissos de Pedersen, sozinhos, não conseguem criar uma solução de privacidade completa**, mas servem como um **componente** em uma solução maior que inclui **provas de conhecimento zero (ZKPs)**, que discutiremos a seguir.

**Provas de conhecimento zero (Zero-knowledge proofs)**

A primeira prova de conhecimento zero foi inventada por **Goldwasser, Micali e Rackoff** em 1985. Argumentos sucintos e transparentes oriundos de PCP foram introduzidos por **Kilian** e **Micali** em 1992 e 1994, respectivamente. Essas provas são os primeiros **SNARKs**, mas eram impraticáveis devido ao tempo de prova (prover) ser inviável; no entanto, as provas eram **curtas em tamanho** e **rápidas de verificar**. Por muito tempo, os ZKPs foram considerados impraticáveis, mas, em 2013, ocorreu um avanço que introduziu tempo linear de prova com tamanho de prova constante. Isso foi ainda mais aprimorado em 2016. Ambos os artigos estão disponíveis aqui:

* GGPR13: <https://eprint.iacr.org/2012/215>
* Groth16: <https://eprint.iacr.org/2016/260>

Em 2013, foi introduzido o **Pinocchio**, o primeiro ZK SNARK prático. Também em 2013, surgiu o **tinyRAM**, a primeira **ZK VM**. Em 2018, foi lançado o primeiro **ZK rollup** por **Barry Whitehat**, a linguagem **Circom** da **iden3**, e o primeiro **ZK-STARK** pela **Starkware**. Em 2019, foi introduzido o **PLONK**, um marco importante que melhorou a usabilidade dos SNARKs. Sua configuração é **universal e atualizável**, o que significa que a configuração precisa ser executada apenas uma vez e pode ser usada para qualquer programa, em vez de exigir uma configuração separada para cada novo programa, como no esquema anterior (por exemplo, no Zcash). Além disso, o PLONK usa **compromissos polinomiais** que são intercambiáveis com qualquer outro esquema de compromisso, para alcançar diferentes melhorias. Isso significa que o **esquema de prova** e o **esquema de compromisso** são separados, podendo-se usar um compromisso mais eficiente. O PLONK usa **KGZ10 (Kate)**, mas pode ser trocado por **FRI** ou outro esquema.

O acrônimo **FRI** significa *Fast Reed-Solomon IOP of Proximity*, e **IOP** significa *Interactive Oracle Proof*. O protocolo FRI garante que um polinômio comprometido tenha grau limitado.

O FRI permite que o SNARK se torne **transparente** e **resistente a ataques quânticos**, enquanto mantém o mesmo esquema de prova, ou seja, o PLONK.  
Os esquemas anteriores, como **Marlin** e **Sonic**, eram mais complexos e custosos.  
Depois, em 2020, foram desenvolvidos o **Cairo ZKVM** e o **Halo2**.  
Em 2021, surgiram o **Circom 2.0** e o desenvolvimento do **Mina**.  
Em 2022, muitos projetos foram lançados, incluindo o **Plonky2**, o **ZKSNARK mais rápido até o momento**, juntamente com diversos outros projetos, como **ZKEVMs**, incluindo **Scroll** e **zkSync**.

Não há sinais de que esse desenvolvimento esteja desacelerando ou sequer se estabilizando; mais e mais pesquisas e desenvolvimentos estão sendo conduzidos por equipes entusiasmadas, e essa tendência deve crescer ainda mais com inovações no horizonte. Como já se acredita amplamente que ZK complementa muito bem os blockchains, **o futuro do blockchain será fortemente orientado ao conhecimento zero**.

A primeira criptomoeda a implementar com sucesso ZKPs (especificamente **ZK-SNARKs** – *Succinct Non-Interactive Argument of Knowledge*) para garantir privacidade no blockchain foi o **Zcash**. A mesma ideia pode ser implementada no Ethereum usando contratos inteligentes e também em outros blockchains. O artigo de pesquisa original está disponível em:  
<https://eprint.iacr.org/2013/879.pdf>

Outro excelente artigo está disponível em:  
<http://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf>

Há outro tipo de ZKP chamado **zk-STARK** (*zero-knowledge Succinct Transparent Argument of Knowledge*), que é uma **melhoria sobre os zk-SNARKs** no sentido de que:

* os zk-STARKs consomem muito **menos largura de banda e armazenamento**;
* **não exigem uma configuração inicial confiável** (trusted setup), considerada controversa e necessária para os zk-SNARKs;
* fundamentalmente, eles são SNARKs, mas **sem configuração confiável**;
* além disso, os zk-STARKs são muito mais **rápidos**, pois não utilizam curvas elípticas nem dependem de funções de hash.

O artigo original de pesquisa sobre zk-STARKs está disponível aqui:  
<https://eprint.iacr.org/2018/046.pdf>

**Bulletproofs** são provas curtas de conhecimento zero não interativas. Elas **não exigem configuração confiável**. Esse esquema permite que um provador comprove que um número criptografado está dentro de um intervalo de números **sem revelar nenhuma outra informação** sobre esse número. Mais informações sobre Bulletproofs estão disponíveis aqui:  
<https://electroneropulse.org/public/doc/Bulletproof%20RingCT.pdf>

A seguir está uma rápida comparação das técnicas mencionadas:

| **Tipo / Propriedade** | **Tamanho da Prova** | **Tempo do Prover** | **Tempo de Verificação** | **Requer Configuração Confiável?** |
| --- | --- | --- | --- | --- |
| **SNARKs** | Pequeno (~288 B) | Médio (~2.3 s) | Pequeno (~10 ms) | **Sim** |
| **STARKs** | Grande (40–50 KB) | Pequeno (~1.6 s) | Médio (~16 s) | **Não** |
| **Bulletproofs** | Médio (<1 KB) | Grande (~30 s) | Grande (~1100 s) | **Não** |

Cobrimos ZKPs nos **Capítulos 21 e 4**. Aqui, focaremos mais em **ZKPs não interativos** e aprofundaremos os **SNARKs**.

Em geral, há duas classes de argumentos de conhecimento zero: **protocolos sigma** e **SNARKs**.

**Nota:** “prova” e “argumento” são dois termos que às vezes são usados como sinônimos, mas têm diferenças sutis.  
Nos *S\*ARKs*, o “AR” significa *argument*, indicando que **não são provas completas**.  
Provas de conhecimento zero são de fato provas, enquanto *arguments* são “argumentos”.  
A diferença entre um argumento e uma prova está relacionada à força da propriedade de *solidez* (*soundness*) no protocolo:

* **Argumentos** são seguros contra um provador limitado computacionalmente (tempo polinomial).
* **Provas** são seguras contra um provador ilimitado (infinitamente poderoso).  
  Assim, provas são **informacionalmente seguras**, enquanto argumentos são **computacionalmente seguros**.

**Protocolos Sigma**

Os **protocolos Sigma** são ZKPs interativos simples, com **três fases**: compromisso, desafio e resposta. Visualmente, um protocolo Sigma pode ser representado como segue:

1. O **prover** envia um **compromisso** ao verificador.
2. O **verificador** envia de volta um **desafio aleatório**.
3. O **prover** responde com uma **resposta baseada no segredo e no desafio**.
4. O **verificador** valida a prova.

Esse protocolo é chamado de “Sigma” por causa da forma do fluxo de mensagens entre as partes, que se assemelha à letra grega Σ.

Este protocolo pode ser convertido em uma versão **não interativa** utilizando o **heurístico de Fiat-Shamir**. A **heurística de Fiat-Shamir** substitui o desafio aleatório do verificador por um **hash da transcrição até o momento**, tornando o protocolo completamente não interativo. Esse processo é amplamente utilizado em muitas construções ZKP.

**SNARKs**

**SNARK** significa **Succinct Non-interactive Argument of Knowledge** (Argumento sucinto não interativo de conhecimento). Os SNARKs possuem várias propriedades desejáveis:

* **Sucedem como argumentos não interativos**, ou seja, requerem apenas uma mensagem entre provador e verificador.
* **São sucintos**, ou seja, as provas são **curtas** e podem ser **verificadas rapidamente**.
* **São argumentos**, portanto seguros contra provadores com poder computacional limitado.
* **São de conhecimento**, ou seja, demonstram que o provador realmente conhece o valor ou solução para o qual está fornecendo a prova.

SNARKs são construídos sobre circuitos aritméticos e requerem **tradução de programas para representações de circuitos**. Isto é necessário para que as provas possam ser geradas e verificadas eficientemente. Essa conversão de lógica de programas para circuitos aritméticos é feita usando **linguagens de circuito**, como **Circom**, **Noir**, **Leo** e outras.

**Aplicações de SNARKs em blockchains**

Os SNARKs são extremamente úteis em contextos de blockchain por várias razões:

* **Privacidade**: permitem transações privadas onde remetente, destinatário e montantes são ocultos.
* **Escalabilidade**: permitem que grandes lotes de transações sejam processados fora da cadeia e apenas uma prova seja colocada on-chain (ZK rollups).
* **Verificação eficiente**: a verificação de provas SNARK é rápida e econômica em termos de gás, mesmo para grandes computações.
* **Compressão de dados**: um SNARK pode provar a validade de uma computação massiva com apenas alguns bytes.

Como mencionado anteriormente, **Zcash** é um exemplo de criptomoeda que utiliza SNARKs para prover privacidade completa, enquanto **zkSync**, **StarkNet**, **Polygon zkEVM** e **Scroll** usam SNARKs para **escalabilidade e privacidade**.

**Conclusão**

Neste capítulo, exploramos os aspectos da **privacidade e anonimato em blockchains**. Discutimos a importância da privacidade, especialmente em aplicações financeiras e comerciais.  
Discutimos também vários métodos para alcançar a privacidade, incluindo:

* **Moedas privadas**, como **Monero** e **Zcash**, que utilizam técnicas como **assinaturas em anel**, **endereços furtivos**, **transações confidenciais**, **compromissos de Pedersen** e **ZKPs**.
* **Protocolos de mistura**, como **Coinjoin**, **CoinShuffle**, **TumbleBit** e **CoinSwap**.
* **Protocolos baseados em hardware confiável**, como **SGX** e **Ekiden**.
* **Propostas de privacidade na camada de rede**, como **Dandelion e Dandelion++**.
* **Privacidade em contratos inteligentes** por meio de tecnologias como **Zether**, **Hawk**, **Zexe**, **Aztec** e **ZKLedger**.
* **Compromissos criptográficos**, especialmente **compromissos de Pedersen**.
* **Provas de conhecimento zero**: **SNARKs**, **STARKs**, **Bulletproofs**, protocolos **Sigma** e a evolução da tecnologia de conhecimento zero.

Observamos como a privacidade em blockchain não é apenas um desafio técnico, mas também **um equilíbrio delicado entre transparência e confidencialidade**, necessário para adoção em larga escala em setores regulamentados.

Vimos também que, embora a privacidade seja desejável, ela pode ser **abusada para atividades ilícitas**, o que requer atenção ao desenhar sistemas que preservem a privacidade mas ainda possam operar dentro de marcos legais aceitáveis.

Finalmente, à medida que **tecnologias baseadas em ZK** continuam amadurecendo, espera-se que a **privacidade se torne uma característica fundamental dos blockchains modernos**, possibilitando **escala, segurança e soberania de dados** ao mesmo tempo.