**Criptografia pós-quântica**

"Os computadores quânticos podem quebrar a criptografia", sugeriu Peter Shor, professor de matemática no MIT. Era 1994, e Shor acabara de conceber um novo algoritmo. Sua descoberta desbloqueou a fatoração eficiente de inteiros, destruindo algoritmos criptográficos como o RSA, caso os computadores quânticos algum dia se tornassem realidade. Na época, o computador quântico era apenas uma teoria, um conceito de uma nova classe de computador baseada na física quântica. A ideia ainda precisava ser comprovada. Em meados de 2015, a Agência de Segurança Nacional (NSA) surpreendeu a todos ao anunciar seus planos de transição para algoritmos resistentes a quânticos (algoritmos criptográficos não vulneráveis a computadores quânticos).

Para aqueles parceiros e fornecedores que ainda não fizeram a transição para os algoritmos de curva elíptica do Suite B, recomendamos não fazer um gasto significativo para fazê-lo neste momento, mas sim preparar-se para a próxima transição para algoritmos resistentes a quânticos.

Este capítulo aborda:

* Computadores quânticos e seu impacto na criptografia
* Criptografia pós-quântica para defesa contra computadores quânticos
* Os algoritmos pós-quânticos de hoje e de amanhã

**14.1 O que são computadores quânticos e por que assustam os criptógrafos?**

Desde o anúncio da NSA, computadores quânticos têm aparecido nas notícias, com empresas como IBM, Google, Alibaba, Microsoft, Intel, entre outras, investindo recursos significativos em sua pesquisa. Mas o que são esses computadores quânticos e por que são tão assustadores? Tudo começou com a mecânica quântica (também chamada de física quântica), um campo da física que estuda o comportamento de coisas pequenas (pense em átomos e menores). Como esta é a base dos computadores quânticos, é aqui que nossa investigação começa.

*"Houve um tempo em que os jornais diziam que apenas doze homens entendiam a teoria da relatividade. Não acredito que tenha havido tal tempo. Pode ter havido um tempo em que apenas um homem entendeu, antes de escrever seu artigo. Mas depois que as pessoas leram o artigo, muitos entenderam a teoria de alguma forma, certamente mais que doze. Por outro lado, posso dizer com segurança que ninguém entende a mecânica quântica."*  
—Richard Feynman (*The Character of Physical Law*, MIT Press, 1965)

**14.1.1 Mecânica quântica, o estudo do pequeno**

Os físicos há muito pensavam que o mundo era determinístico, como nossos geradores criptográficos de números pseudorrandômicos: se você soubesse como o universo funciona e tivesse um computador grande o suficiente para calcular a "função do universo", tudo o que precisaria seria a semente (as informações contidas no Big Bang) e você poderia prever tudo a partir daí. Sim, tudo — até o fato de que, apenas 13,7 bilhões de anos após o início do universo, você estaria lendo esta linha. Nesse mundo, não há espaço para aleatoriedade. Cada decisão que você toma está predeterminada por eventos passados, mesmo os que ocorreram antes do seu nascimento.

Embora essa visão tenha intrigado muitos filósofos — "Temos mesmo livre arbítrio?" —, um campo interessante da física começou a crescer na década de 1990, desconcertando muitos cientistas desde então: a física quântica (também chamada de mecânica quântica). Acontece que objetos muito pequenos (pense em átomos e menores) tendem a se comportar de maneira bastante diferente do que observamos e teorizamos com a física clássica. Nessa escala (sub)atômica, partículas parecem se comportar como ondas às vezes, no sentido de que diferentes ondas podem se superpor, fundindo-se em uma onda maior ou se cancelando brevemente.

Uma medição que podemos realizar em partículas como elétrons é o spin. Por exemplo, podemos medir se um elétron está girando para cima ou para baixo. Até aí, nada estranho. O estranho é que a mecânica quântica afirma que uma partícula pode estar nesses dois estados ao mesmo tempo, girando para cima e para baixo. Dizemos que a partícula está em superposição quântica.

Esse estado especial pode ser induzido manualmente usando técnicas diferentes dependendo do tipo de partícula. Uma partícula pode permanecer em superposição até que a medimos; nesse caso, a partícula colapsa em apenas um desses estados possíveis (girando para cima ou para baixo). É essa superposição quântica que os computadores quânticos acabam utilizando: em vez de ter um bit que pode ser 1 ou 0, um bit quântico ou qubit pode ser ambos 0 e 1 ao mesmo tempo.

Ainda mais estranho, a teoria quântica diz que é apenas no momento da medição — e não antes — que a partícula em superposição decide aleatoriamente qual estado assumirá (cada estado tendo 50% de chance de ser observado). Se isso parece estranho, você não está sozinho. Muitos físicos não conseguiam conceber como isso funcionaria em um mundo determinístico. Einstein, convencido de que algo estava errado com essa nova teoria, uma vez disse: "Deus não joga dados." No entanto, os criptógrafos se interessaram, pois finalmente havia uma maneira de obter números verdadeiramente aleatórios! É isso que os geradores quânticos de números aleatórios (QRNGs) fazem, colocando continuamente partículas como fótons em superposição e medindo-as.

Os físicos também teorizaram como seria a mecânica quântica com objetos na nossa escala. Isso levou ao famoso experimento do gato de Schrödinger: um gato em uma caixa está simultaneamente morto e vivo até que alguém olhe dentro (o que gerou muitos debates sobre o que exatamente constitui um observador).

*"Um gato é trancado em uma câmara de aço, juntamente com o seguinte dispositivo (protegido contra interferência direta do gato): em um contador Geiger há uma pequena quantidade de substância radioativa, tão pequena que, talvez no decorrer de uma hora, um dos átomos decaia — ou talvez nenhum; se ocorrer o decaimento, o tubo do contador descarrega e, através de um relé, aciona um martelo que quebra um pequeno frasco de ácido cianídrico. Se deixarmos o sistema isolado por uma hora, diríamos que o gato ainda está vivo se nenhum átomo decaiu nesse meio tempo. O primeiro decaimento atômico o envenenaria. A função psi de todo o sistema expressaria isso ao misturar o gato vivo e morto (perdoem a expressão) em partes iguais."*  
—Erwin Schrödinger (*The Present Situation in Quantum Mechanics*, 1935)

Tudo isso é altamente contra-intuitivo para nós, pois nunca encontramos comportamentos quânticos no dia a dia. Agora, vamos adicionar ainda mais esquisitice!

Às vezes, partículas interagem entre si (por exemplo, colidindo uma com a outra) e acabam em um estado de forte correlação, onde é impossível descrever uma partícula sem as outras. Esse fenômeno é chamado de **emaranhamento quântico**, e é um dos ingredientes secretos por trás do ganho de desempenho dos computadores quânticos. Se, por exemplo, duas partículas estão emaranhadas, então, quando uma delas é medida, ambas colapsam e o estado de uma é perfeitamente correlacionado ao estado da outra. Ok, isso foi confuso. Vamos a um exemplo: se dois elétrons estão emaranhados e um deles é então medido e verificado que está girando para cima, sabemos que o outro está girando para baixo (mas não antes que o primeiro seja medido). Além disso, qualquer experimento sempre termina do mesmo jeito.

É difícil acreditar, mas ainda mais impressionante é que foi demonstrado que o emaranhamento funciona mesmo a distâncias muito longas. Einstein, Podolsky e Rosen argumentaram famosamente que a descrição da mecânica quântica era incompleta, provavelmente faltando variáveis ocultas, as quais explicariam o emaranhamento (ou seja, uma vez separadas, as partículas já saberiam exatamente qual seria o resultado de sua medição).

Einstein, Podolsky e Rosen também descreveram um experimento mental (o paradoxo EPR, nomeado a partir das iniciais de seus sobrenomes) no qual duas partículas emaranhadas são separadas por uma grande distância (pense em anos-luz de distância) e então medidas aproximadamente ao mesmo tempo. De acordo com a mecânica quântica, a medição de uma das partículas afetaria instantaneamente a outra, o que seria impossível, já que nenhuma informação pode viajar mais rápido que a luz, segundo a teoria da relatividade (daí o paradoxo). Esse estranho experimento mental foi o que Einstein famosamente chamou de "ação fantasmagórica à distância".

Posteriormente, John Bell estabeleceu uma desigualdade de probabilidades conhecida como **teorema de Bell**; o teorema, se demonstrado como verdadeiro, provaria a existência das variáveis ocultas mencionadas pelos autores do paradoxo EPR. A desigualdade foi posteriormente violada experimentalmente (muitas, muitas vezes), o suficiente para nos convencer de que o emaranhamento é real, descartando a presença de quaisquer variáveis ocultas.

Hoje, dizemos que a medição de partículas emaranhadas leva as partículas a se coordenarem entre si, o que contorna a previsão relativística de que a comunicação não pode ser mais rápida do que a luz. De fato, tente pensar em uma forma de usar o emaranhamento para criar um canal de comunicação, e você verá que isso não é possível. Para os criptógrafos, entretanto, a ação fantasmagórica à distância significou que poderíamos desenvolver novas maneiras de realizar trocas de chaves; essa ideia é chamada de **distribuição quântica de chaves (QKD, na sigla em inglês)**.

Imagine distribuir duas partículas emaranhadas para dois pares: que então mediriam suas respectivas partículas para começar a formar a mesma chave (pois medir uma partícula dá informações sobre a medição da outra). O conceito do QKD é ainda mais atraente devido ao **teorema da não-clonagem**, que afirma que você não pode observar passivamente tal troca e criar uma cópia exata de uma das partículas sendo enviadas nesse canal. Contudo, esses protocolos são vulneráveis a ataques triviais do tipo *man-in-the-middle (MITM)* e são meio inúteis sem já haver uma forma de autenticar os dados. Essa falha levou alguns criptógrafos como Bruce Schneier a declarar que “o QKD como produto não tem futuro”.

Isso é tudo o que direi sobre física quântica, pois isso já é demais para um livro sobre criptografia. Se você não acredita em nenhuma das coisas bizarras que acabou de ler, você não está sozinho. Em seu livro *Quantum Mechanics for Engineers*, Leon van Dommelen escreve: “A física terminou com a mecânica quântica não porque parecia a explicação mais lógica, mas porque inúmeras observações a tornaram inevitável.”

**14.1.2 Do nascimento dos computadores quânticos à supremacia quântica**

Em 1980, nasceu a ideia de computação quântica. Paul Benioff foi o primeiro a descrever o que poderia ser um computador quântico: um computador construído a partir das observações feitas nas últimas décadas da mecânica quântica. Ainda nesse mesmo ano, Paul Benioff e Richard Feynman argumentaram que essa seria a única maneira de simular e analisar sistemas quânticos, à parte das limitações dos computadores clássicos.

Somente 18 anos depois, um algoritmo quântico rodando em um computador quântico real foi demonstrado pela primeira vez pela IBM. Avançando para 2011, a D-Wave Systems, uma empresa de computadores quânticos, anunciou o primeiro computador quântico comercialmente disponível, lançando toda uma indústria em uma busca por criar o primeiro computador quântico escalável.

Ainda há um longo caminho a percorrer, e um computador quântico útil é algo que ainda não foi alcançado. O resultado mais recente, no momento da redação deste livro (2021), foi o anúncio do Google, em 2019, de ter alcançado a **supremacia quântica** com um computador quântico de 53 qubits. Supremacia quântica significa que, pela primeira vez, um computador quântico realizou algo que um computador clássico não conseguiu. Em 3 minutos e 20 segundos, realizou uma análise que levaria um computador clássico cerca de 10.000 anos para terminar. Mas, antes que você se empolgue demais, ele superou um computador clássico em uma tarefa que não era útil. Ainda assim, é um marco incrível, e só podemos imaginar aonde tudo isso nos levará.

Um computador quântico basicamente utiliza os fenômenos da física quântica (como superposição e emaranhamento) da mesma forma que computadores clássicos usam eletricidade para realizar cálculos. Em vez de bits, computadores quânticos usam **qubits**, que podem ser transformados via **portas quânticas** para defini-los em valores específicos ou colocá-los em um estado de superposição e, até mesmo, emaranhamento. Isso é um pouco semelhante ao uso de portas lógicas em circuitos de computadores clássicos. Após a conclusão de um cálculo, os qubits podem ser medidos para serem interpretados de forma clássica — como 0s e 1s. Nesse ponto, os resultados podem ser processados por um computador clássico para finalizar um cálculo útil.

Em geral, N qubits emaranhados contêm informações equivalentes a 2ᴺ bits clássicos. Mas medir os qubits ao final de um cálculo apenas fornece N números entre 0 e 1. Assim, nem sempre é claro como um computador quântico pode ajudar, e eles só têm utilidade em um número limitado de aplicações. É possível que eles se tornem mais úteis à medida que as pessoas encontrarem maneiras inteligentes de explorar seu poder.

Hoje, você já pode usar um computador quântico do conforto da sua casa. Serviços como o **IBM Quantum** (<https://quantum-computing.ibm.com>) permitem que você construa circuitos quânticos e os execute em computadores quânticos reais hospedados na nuvem. Claro, tais serviços ainda são bastante limitados no momento (início de 2021), com apenas alguns qubits disponíveis. Ainda assim, é uma experiência de explodir a mente criar seu próprio circuito e esperar que ele rode em um computador quântico real — e tudo isso gratuitamente.

**14.1.3 O impacto dos algoritmos de Grover e Shor na criptografia**

Infelizmente, como mencionei anteriormente, computadores quânticos não são úteis para todos os tipos de computação e, portanto, não são substitutos mais poderosos e diretos para nossos computadores clássicos. Mas então, para que eles servem?

Em 1994, numa época em que o conceito de computador quântico era apenas um experimento mental, Peter Shor propôs um algoritmo quântico para resolver os problemas de logaritmo discreto e fatoração. Shor teve a percepção de que um computador quântico poderia ser usado para calcular rapidamente soluções para problemas relacionados aos problemas difíceis encontrados na criptografia. Acontece que existe um algoritmo quântico eficiente que ajuda a encontrar um período tal que f(x + período) = f(x) para qualquer x dado. Por exemplo, encontrar o valor do período tal que gˣ+período = gˣ mod N. Isso, por sua vez, leva a algoritmos que podem resolver eficientemente os problemas de fatoração e logaritmo discreto, impactando efetivamente algoritmos como RSA (coberto no capítulo 6) e Diffie-Hellman (coberto no capítulo 5).

O algoritmo de Shor é devastador para a criptografia assimétrica, já que a maioria dos algoritmos assimétricos atualmente em uso depende do logaritmo discreto ou do problema de fatoração — a maior parte do que você viu ao longo deste livro, na verdade. Pode-se pensar que logaritmo discreto e fatoração ainda são problemas matemáticos difíceis e que poderíamos (talvez) aumentar o tamanho dos parâmetros de nossos algoritmos para fortalecer sua defesa contra computadores quânticos. Infelizmente, foi demonstrado em 2017, por Bernstein e outros, que embora aumentar os parâmetros funcione, seria altamente impraticável. A pesquisa estimou que o RSA poderia ser tornado resistente a quânticos aumentando seus parâmetros para 1 terabyte. Irrealista, para dizer o mínimo.

*O algoritmo de Shor destrói os fundamentos da criptografia de chave pública implantada: RSA e o problema de logaritmo discreto em campos finitos e curvas elípticas. Documentos confidenciais de longo prazo, como registros de saúde de pacientes e segredos de Estado, precisam garantir segurança por muitos anos, mas informações criptografadas hoje usando RSA ou curvas elípticas e armazenadas até que computadores quânticos estejam disponíveis serão tão fáceis de decifrar quanto mensagens criptografadas pela Enigma são hoje.*  
—PQCRYPTO: Recomendações iniciais de sistemas pós-quânticos de longo prazo (2015)

Para a criptografia simétrica, as coisas são bem menos preocupantes. O algoritmo de Grover foi proposto em 1996 por Lov Grover, como uma forma de otimizar uma busca em uma lista não ordenada. Uma busca em uma lista não ordenada de N itens leva, em média, N/2 operações com um computador clássico; levaria √N operações com um computador quântico. Um ganho considerável!

O algoritmo de Grover é uma ferramenta bastante versátil que pode ser aplicada de várias maneiras na criptografia, por exemplo, para extrair a chave simétrica de um cifrador ou encontrar uma colisão em uma função hash. Para buscar uma chave de 128 bits, o algoritmo de Grover rodaria em 2⁶⁴ operações em um computador quântico, em vez de 2¹²⁷ em um computador clássico. Esta é uma afirmação assustadora para todos os nossos algoritmos de criptografia simétrica, mas podemos simplesmente aumentar os parâmetros de segurança de 128 bits para 256 bits e isso é suficiente para conter o ataque de Grover. Assim, se você deseja proteger sua criptografia simétrica contra computadores quânticos, pode simplesmente usar SHA-3-512 em vez de SHA-3-256, AES-256-GCM em vez de AES-128-GCM, e assim por diante.

Para resumir: a criptografia simétrica está, em grande parte, bem; a criptografia assimétrica, não. Isso é ainda pior do que parece à primeira vista: a criptografia simétrica é frequentemente precedida por uma troca de chaves, que é vulnerável a computadores quânticos. Então, este seria o fim da criptografia como a conhecemos?

**14.1.4 Criptografia pós-quântica, a defesa contra computadores quânticos**

Felizmente, isso não foi o fim do mundo para a criptografia. A comunidade rapidamente reagiu à ameaça quântica organizando-se e pesquisando algoritmos antigos e novos que não seriam vulneráveis aos ataques de Shor e Grover. O campo da **criptografia resistente a quânticos**, também conhecido como **criptografia pós-quântica**, nasceu. Existem esforços de padronização em diferentes lugares na internet, mas o esforço mais respeitado é o do NIST, que em 2016 iniciou um processo de padronização de criptografia pós-quântica.

*Parece que uma transição para a criptografia pós-quântica não será simples, pois dificilmente haverá um substituto direto para nossos algoritmos atuais de criptografia de chave pública. Um esforço significativo será necessário para desenvolver, padronizar e implantar novos sistemas criptográficos pós-quânticos. Além disso, essa transição precisa ocorrer bem antes de quaisquer computadores quânticos em larga escala serem construídos, de modo que qualquer informação que seja posteriormente comprometida por criptoanálise quântica já não seja mais sensível quando esse comprometimento ocorrer. Portanto, é desejável planejar essa transição com antecedência.*  
—Página de Criptografia Pós-Quântica do NIST (2016)

Desde que o NIST iniciou este processo, 82 candidatos se inscreveram e 3 rodadas já passaram, reduzindo a lista de candidatos para 7 finalistas e 8 finalistas alternativos (improváveis de serem padronizados, mas suficientemente únicos para serem boas opções caso um dos paradigmas usados pelos finalistas acabe sendo quebrado). O esforço de padronização do NIST busca substituir os tipos mais comuns de primitivas de criptografia assimétrica, que incluem esquemas de assinatura e criptografia assimétrica. Esta última também pode facilmente servir como uma primitiva de troca de chaves, como você aprendeu no capítulo 6.

No restante deste capítulo, abordarei os diferentes tipos de algoritmos de criptografia pós-quântica que estão sendo considerados para padronização e indicarei quais você pode utilizar hoje.

**14.2 Assinaturas baseadas em hash: Não precisam de nada além de uma função hash**

Embora todos os esquemas práticos de assinatura pareçam usar funções hash, existem maneiras de construir esquemas de assinatura que fazem uso apenas de funções hash, e nada mais. Melhor ainda, esses esquemas tendem a depender apenas da resistência à pré-imagem das funções hash e não de sua resistência a colisões. Esta é uma proposta bastante atraente, já que uma grande parte da criptografia aplicada já se baseia em funções hash sólidas e bem compreendidas.

As funções hash modernas também são resistentes a computadores quânticos, o que torna esses esquemas de assinatura baseados em hash naturalmente resistentes a quânticos. Vejamos o que são essas assinaturas baseadas em hash e como funcionam.

**14.2.1 Assinaturas de uso único (OTS) com assinaturas de Lamport**

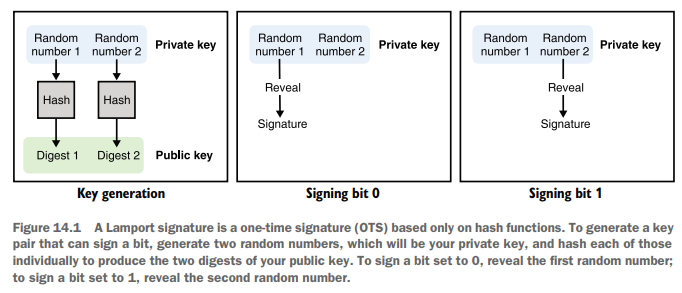
Em 18 de outubro de 1979, Leslie Lamport publicou seu conceito de assinaturas de uso único (OTS): pares de chaves que você pode usar para assinar apenas uma vez. A maioria dos esquemas de assinatura depende (em parte) de funções unidirecionais (tipicamente funções hash) para suas provas de segurança. A beleza do esquema de Lamport é que sua assinatura depende exclusivamente da segurança dessas funções unidirecionais.

Imagine que você queira assinar um único bit. Primeiro, você gera um par de chaves ao:

1. Gerar dois números aleatórios, x e y, que serão a chave privada;
2. Hashear x e y para obter dois digests h(x) e h(y), que você pode publicar como a chave pública.

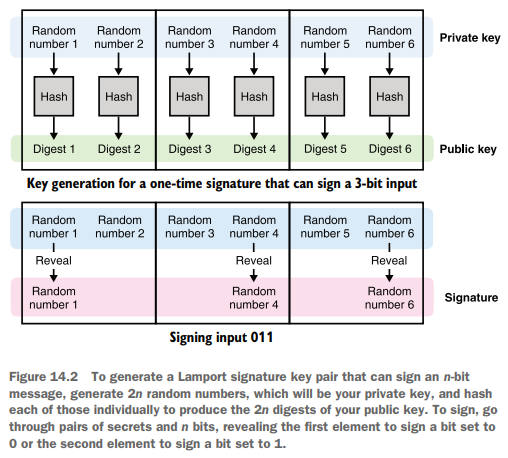
Para assinar um bit com valor 0, revele a parte x de sua chave privada; para assinar um bit com valor 1, revele a parte y. Para verificar uma assinatura, simplesmente hasheie-a para verificar se corresponde à parte correta da chave pública. Eu ilustro isso na figura 14.1.

Assinar um bit não é muito útil, você dirá. Sem problemas; uma assinatura de Lamport funciona para entradas maiores simplesmente criando mais pares de segredos, um por bit a ser assinado (veja a figura 14.2). Obviamente, se sua entrada for maior que 256 bits, você primeiro a hashearia e depois a assinaria.

\*\*Figura 14.1\*\* Uma assinatura de Lamport é uma assinatura de uso único (OTS) baseada apenas em funções hash. Para gerar um par de chaves que pode assinar um bit, gere dois números aleatórios (sua chave privada) e hasheie cada um individualmente para produzir os dois digests da sua chave pública. Para assinar um bit com valor 0, revele o primeiro número aleatório; para assinar um bit com valor 1, revele o segundo número aleatório.

Uma limitação importante deste esquema é que você só pode usá-lo uma vez; se o usar duas vezes, você acaba autorizando alguém a misturar as duas assinaturas para forjar outras assinaturas válidas. Podemos melhorar a situação de forma ingênua gerando um grande número de pares de chaves de uso único em vez de apenas um, garantindo que um par de chaves seja descartado após o uso. Isso não apenas faz sua chave pública ter o tamanho proporcional ao número de assinaturas que você imagina que pode vir a usar, como também significa que você precisa acompanhar quais pares de chaves já foram usados (ou melhor, se livrar das chaves privadas já utilizadas). Por exemplo, se você sabe que desejará assinar no máximo 1.000 mensagens de 256 bits com uma função hash de saída de 256 bits, sua chave privada e pública precisariam ter ambas 1000 × (256 × 2 × 256) bits, o que equivale a cerca de 16 megabytes. Isso é bastante para apenas 1.000 assinaturas.

A maioria dos esquemas de assinatura baseados em hash propostos hoje se baseia nos fundamentos criados por Lamport para permitir muito mais assinaturas (às vezes, uma quantidade praticamente ilimitada), chaves privadas sem estado (embora alguns esquemas propostos ainda sejam com estado) e tamanhos de parâmetros mais práticos.

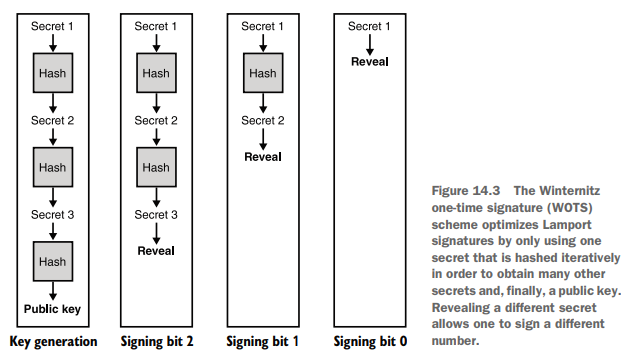


\*\*Figura 14.2\*\* Para gerar um par de chaves de assinatura de Lamport capaz de assinar uma mensagem de n bits, gere 2n números aleatórios (sua chave privada) e hasheie cada um individualmente para produzir os 2n digests da sua chave pública. Para assinar, percorra pares de segredos e bits, revelando o primeiro elemento para bits com valor 0 ou o segundo para bits com valor 1.

**14.2.2 Chaves menores com as assinaturas de uso único de Winternitz (WOTS)**

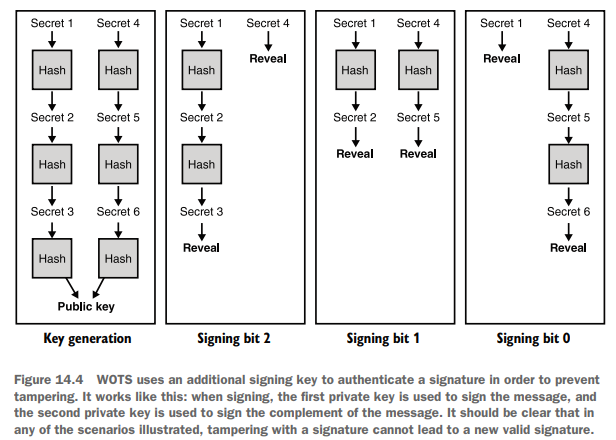
Alguns meses após a publicação de Lamport, Robert Winternitz, do Departamento de Matemática de Stanford, propôs publicar *hashes* de *hashes* de um segredo h(h(...h(x))) = hʷ(x) em vez de publicar múltiplos *digests* de múltiplos segredos, a fim de otimizar o tamanho de uma chave privada (veja a figura 14.3). Este esquema é chamado de **assinatura de uso único de Winternitz (WOTS)** em homenagem ao autor.

Por exemplo, escolhendo w = 16, você pode assinar 16 valores diferentes ou, em outras palavras, entradas de 4 bits. Você começa gerando um valor aleatório x que serve como sua chave privada e o hasheia 16 vezes para obter sua chave pública, h¹⁶(x). Agora imagine que você quer assinar os bits 1001 (9 em base 10); você publica a nona iteração do *hash*, h⁹(x). Eu ilustro isso na figura 14.3.



\*\*Figura 14.3\*\* O esquema de assinatura de uso único de Winternitz (WOTS) otimiza as assinaturas de Lamport usando apenas um segredo que é \*hasheado\* iterativamente para obter muitos outros segredos e, finalmente, uma chave pública. Revelar um segredo diferente permite assinar um número diferente.

Tire alguns minutos para entender como este esquema funciona. Você vê um problema com ele?  
Um problema importante é que este esquema permite falsificações de assinatura. Imagine que você veja a assinatura de alguém para o bit 1001, que seria h⁹(x), segundo nosso exemplo anterior. Você pode simplesmente hasheá-la novamente para obter outras iterações como h¹⁰(x) ou h¹¹(x), o que lhe daria uma assinatura válida para os bits 1010 ou 1011. Isso pode ser contornado adicionando uma pequena tag de autenticação após a mensagem, a qual você também teria que assinar. Eu ilustro isso na figura 14.4. Para se convencer de que isso resolve o problema de falsificação, tente forjar uma assinatura a partir de outra assinatura.



\*\*Figura 14.4\*\* O WOTS usa uma chave de assinatura adicional para autenticar uma assinatura e impedir manipulações. Funciona assim: ao assinar, a primeira chave privada é usada para assinar a mensagem e a segunda chave privada é usada para assinar o complemento da mensagem. Deve ser claro que, em qualquer um dos cenários ilustrados, manipular uma assinatura não pode levar a uma nova assinatura válida.

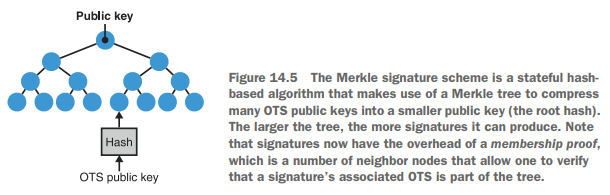
**14.2.3 Assinaturas de múltiplos usos com XMSS e SPHINCS+**

Até agora, você viu maneiras de assinar coisas usando apenas funções hash. Enquanto as assinaturas de Lamport funcionam, elas têm tamanhos de chave grandes, então o WOTS as melhorou reduzindo o tamanho das chaves. Ainda assim, ambos os esquemas não escalam bem, pois ambos são assinaturas de uso único (se reutilizar um par de chaves, você quebra o esquema), e assim, seus parâmetros aumentam linearmente de tamanho dependendo do número de assinaturas que você acha que precisará.

Alguns esquemas que toleram o reuso de um par de chaves para algumas assinaturas (em vez de apenas uma) existem. Esses esquemas são chamados de **assinaturas de poucos usos (FTS — Few-Time Signatures)** e irão falhar, permitindo falsificações de assinatura se reutilizados muitas vezes. As FTS dependem de baixas probabilidades de reutilizar a mesma combinação de segredos de um conjunto de segredos. Esta é uma pequena melhoria sobre as assinaturas de uso único, permitindo uma redução do risco de reutilização de chaves. Mas podemos fazer melhor.

Qual é uma técnica que você aprendeu neste livro que comprime muitas coisas em uma só? A resposta são as **árvores de Merkle**. Como você deve lembrar do capítulo 12, uma árvore de Merkle é uma estrutura de dados que fornece provas curtas para perguntas como "meus dados estão neste conjunto?". Na década de 1990, o mesmo Merkle que propôs as árvores de Merkle também inventou um esquema de assinatura baseado em funções hash que comprime um número de assinaturas de uso único em uma árvore de Merkle.

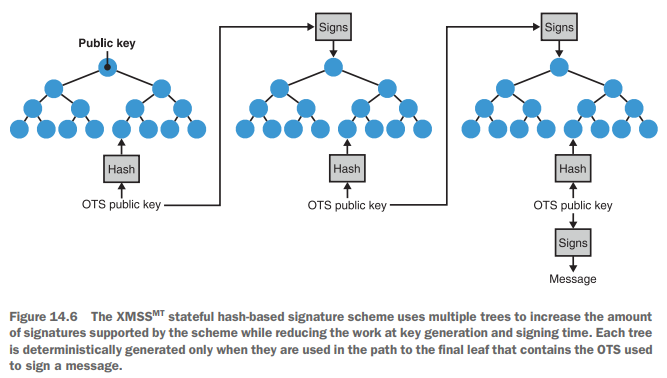
A ideia é bastante direta: cada folha da sua árvore é o *hash* de uma assinatura de uso único, e o *hash* da raiz pode ser usado como chave pública, reduzindo seu tamanho ao tamanho de saída da sua função hash. Para assinar, você escolhe uma assinatura de uso único que ainda não foi usada e então a aplica conforme explicado na seção 14.2.2. A assinatura consiste na assinatura de uso único junto com a prova de Merkle de que ela pertence à sua árvore de Merkle (todos os nós vizinhos). Este esquema é obviamente com estado, pois deve-se ter cuidado para não reutilizar uma das assinaturas de uso único na árvore. Eu ilustro isso na figura 14.5.



\*\*Figura 14.5\*\* O esquema de assinatura de Merkle é um algoritmo baseado em hash com estado que utiliza uma árvore de Merkle para comprimir muitas chaves públicas OTS em uma chave pública menor (o \*hash\* da raiz). Quanto maior a árvore, mais assinaturas ela pode produzir. Note que as assinaturas agora têm o overhead de uma prova de pertencimento, que é um número de nós vizinhos que permitem verificar que a OTS associada pertence à árvore.

O **XMSS (Extended Merkle Signature Scheme)**, padronizado na RFC 8391, buscou transformar em produção as assinaturas de Merkle ao adicionar uma série de otimizações ao esquema de Merkle. Por exemplo, para produzir um par de chaves capaz de assinar N mensagens, você deve gerar N chaves privadas OTS. Embora a chave pública agora seja apenas o *hash* da raiz, você ainda precisa armazenar N chaves privadas OTS. O XMSS reduz o tamanho da chave privada armazenada gerando determinísticamente cada OTS na árvore usando uma semente e a posição da folha na árvore. Assim, você só precisa armazenar a semente como chave privada, em vez de todas as chaves privadas OTS, e pode rapidamente regenerar qualquer par de chaves OTS a partir de sua posição na árvore e da semente. Para acompanhar qual folha/OTS foi usada por último, a chave privada também contém um contador que é incrementado cada vez que é usada para assinar.

Dito isto, há um limite de quantas OTS você pode manter em uma árvore de Merkle. Quanto maior a árvore, mais tempo levará para regenerar a árvore a fim de assinar mensagens (já que você precisa regenerar todas as folhas para produzir uma prova de Merkle). Quanto menor a árvore, menos chaves privadas OTS precisam ser regeneradas ao assinar, mas isso obviamente derrota o propósito: estamos de volta a um número limitado de assinaturas. A solução é usar uma árvore menor, onde as OTS em suas folhas não são usadas para assinar mensagens, mas sim para assinar o *hash* da raiz de outras árvores de Merkle de OTS. Isso transforma nossa árvore inicial em uma **hiperárvore — árvore de árvores** — e é uma das variantes do XMSS chamada **XMSSMT**. Com o XMSSMT, apenas as árvores envolvidas no caminho de uma OTS precisam ser regeneradas, com base na mesma técnica. Eu ilustro isso na figura 14.6.



\*\*Figura 14.6\*\* O esquema de assinatura baseado em hash com estado XMSSMT usa múltiplas árvores para aumentar a quantidade de assinaturas suportadas pelo esquema enquanto reduz o trabalho na geração de chaves e no momento da assinatura. Cada árvore é gerada determinísticamente somente quando é usada no caminho até a folha final que contém a OTS usada para assinar uma mensagem.

Observe que o fato de o XMSS e o XMSSMT serem com estado pode não ser um problema em algumas situações, mas não é uma propriedade desejável, em geral. Ter que manter controle de um contador é contraintuitivo, pois não é esperado dos usuários de esquemas de assinatura convencionais. Essa mudança de prática pode levar à reutilização de OTS (e, assim, à falsificação de assinaturas) em caso de uso indevido. Por exemplo, reverter um sistema de arquivos a um estado anterior ou usar a mesma chave de assinatura em múltiplos servidores pode induzir o uso do mesmo caminho na hiperárvore duas vezes para assinar uma mensagem.

Para resolver uma das maiores desvantagens do XMSS (sua necessidade de estado) e expor uma interface semelhante aos esquemas de assinatura aos quais estamos acostumados, o esquema de assinatura **SPHINCS+** foi proposto como parte da competição de criptografia pós-quântica do NIST. O esquema de assinatura sem estado SPHINCS+ amplia o XMSSMT com três grandes mudanças:

* **Assinar a mesma mensagem duas vezes leva à mesma assinatura.** De forma semelhante ao EdDSA (coberto no capítulo 7), o caminho usado na hiperárvore é derivado de forma determinística, com base na chave privada e na mensagem. Isso garante que assinar a mesma mensagem duas vezes leve à mesma OTS e, assim, à mesma assinatura; e, como a chave privada é usada, os atacantes também não conseguem prever qual caminho você usará para assinar as mensagens deles, caso você, de alguma forma, assine mensagens de terceiros.
* **Usar mais árvores.** O XMSSMT evita reutilizar a mesma OTS mantendo registro de qual OTS foi usada por último. Como o objetivo do SPHINCS+ é evitar manter um estado, ele precisa evitar colisões ao escolher um caminho de forma pseudorrandômica. Para isso, o SPHINCS+ simplesmente usa uma quantidade muito maior de OTS, reduzindo a probabilidade de reutilizar a mesma duas vezes. Como o SPHINCS+ também usa uma hiperárvore, isso se traduz em mais árvores.
* **Usar assinaturas de poucos usos (FTS).** Como a segurança do esquema se baseia na probabilidade de reutilizar o mesmo caminho duas vezes, o SPHINCS+ também substitui a OTS final usada para assinar mensagens pela FTS mencionada anteriormente. Assim, reutilizar o mesmo caminho para assinar duas mensagens diferentes ainda não contribui diretamente para quebrar o esquema de assinatura.

Enquanto o SPHINCS+ está sendo considerado para padronização na competição de criptografia pós-quântica do NIST, ele não é o principal candidato. O SPHINCS+ não apenas é lento, como suas assinaturas são grandes em comparação com as alternativas propostas (como as baseadas em reticulados, sobre as quais você aprenderá mais adiante neste capítulo). Os esquemas baseados em hash com estado, como o XMSS, oferecem maior velocidade e tamanhos de assinatura melhores (menos de 3 KB em comparação com o mínimo de 8 KB do SPHINCS+). (Em termos de tamanhos de chave pública, ambos os esquemas fornecem tamanhos semelhantes aos esquemas de assinatura pré-quânticos como ECDSA e Ed25519.) Devido aos tamanhos de parâmetros mais realistas e à segurança bem compreendida, o XMSS é recomendado como um padrão inicial pelo NIST no SP 800-208, “Recomendação para esquemas de assinatura baseados em hash com estado.”

A seguir, vamos dar uma olhada em duas outras maneiras de construir primitivas criptográficas resistentes a quânticos. Um aviso gentil: elas são bem mais pesadas em matemática!

**14.3 Chaves e assinaturas menores com criptografia baseada em reticulados**

Um grande número de esquemas de criptografia pós-quântica é baseado em **reticulados**, uma estrutura matemática que você aprenderá nesta seção. A própria competição de criptografia pós-quântica do NIST elegeu esquemas baseados em reticulados para metade de seus finalistas. Isso torna a **criptografia baseada em reticulados** o paradigma mais provável de vencer e se tornar um padrão do NIST. Nesta seção, falarei sobre dois algoritmos baseados em reticulados: **Dilithium**, um esquema de assinatura, e **Kyber**, uma primitiva de criptografia de chave pública. Mas antes disso, vejamos o que são reticulados.

**14.3.1 O que é um reticulado?**

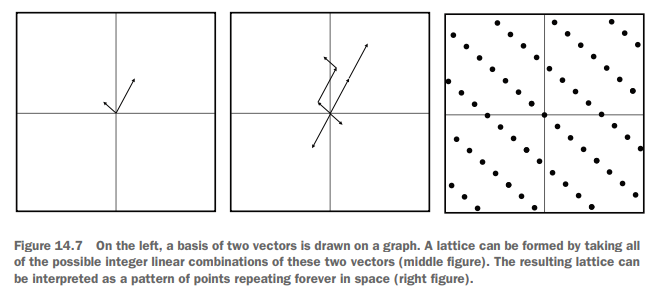
Primeiro, *baseado em reticulado* provavelmente não significa o que você pensa que significa. Pegue o RSA (coberto no capítulo 6), do qual dizemos que é baseado no problema de fatoração. Isso não significa que usamos fatoração no RSA, e sim que a fatoração é como se ataca o RSA, e como a fatoração é difícil, dizemos que o RSA é seguro. É a mesma coisa com os sistemas criptográficos baseados em reticulados: **reticulados são estruturas que possuem problemas difíceis**, e esses sistemas criptográficos são seguros enquanto esses problemas continuarem difíceis.

Dito isso, o que é um reticulado? Bem, é como um espaço vetorial, mas com inteiros.  
Se você não se lembra o que é um espaço vetorial, é o conjunto de todos os vetores que podem ser criados usando:

* **Uma base** — um conjunto de vetores; por exemplo, (0,1) e (1,0).
* **Uma operação entre vetores** — os vetores podem ser somados; por exemplo, (0,1) + (1,0) = (1,1).
* **Uma operação escalar** — um vetor pode ser multiplicado por um escalar; por exemplo, 3 × (1,2) = (3,6).

Em nosso exemplo, o espaço vetorial contém todos os vetores que podem ser expressos como uma combinação linear da base, o que se traduz em qualquer vetor que possa ser escrito como a × (0,1) + b × (1,0) para quaisquer escalares a e b. Por exemplo, 0,5 × (0,1) + 3,87 × (1,0) = (3,87, 0,5) está no nosso espaço vetorial, assim como 99 × (0,1) + 0 × (1,0) = (0,99), e assim por diante.

Um **reticulado** é um espaço vetorial onde todos os números envolvidos são inteiros.  
Sim, em criptografia, gostamos de inteiros. Eu ilustro isso na figura 14.7.

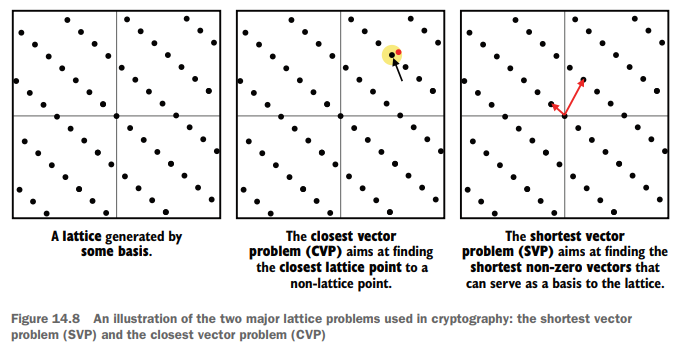


\*\*Figura 14.7\*\* À esquerda, uma base de dois vetores é desenhada em um gráfico. Um reticulado pode ser formado ao se tomar todas as combinações lineares inteiras possíveis desses dois vetores (figura do meio). O reticulado resultante pode ser interpretado como um padrão de pontos se repetindo infinitamente no espaço (figura da direita).

Existem vários problemas bem conhecidos como difíceis no espaço dos reticulados, e para cada um desses problemas, temos algoritmos para resolvê-los. Esses algoritmos são frequentemente os melhores que conseguimos pensar, mas isso não significa que sejam eficientes ou mesmo práticos. Assim, os problemas são considerados difíceis **pelo menos até que uma solução mais eficiente seja encontrada**. Os dois problemas mais conhecidos são os seguintes (ambos ilustrados na figura 14.8):

* **Problema do vetor mais curto (SVP — Shortest Vector Problem)** — Responde à pergunta: qual é o vetor não nulo mais curto em seu reticulado?
* **Problema do vetor mais próximo (CVP — Closest Vector Problem)** — Dado um ponto fora do reticulado, encontra o ponto mais próximo a ele dentro do reticulado.

Geralmente, usamos algoritmos como o **LLL** (algoritmo de Lenstra–Lenstra–Lovász) ou o **BKZ** (algoritmo de Block-Korkine-Zolotarev) para resolver ambos os problemas (CVP pode ser reduzido ao SVP). Esses são algoritmos que **reduzem a base de um reticulado**, ou seja, tentam encontrar um conjunto de vetores mais curtos do que os dados originalmente e que consigam produzir exatamente o mesmo reticulado.



\*\*Figura 14.8\*\* Uma ilustração dos dois principais problemas de reticulado usados na criptografia: o problema do vetor mais curto (SVP) e o problema do vetor mais próximo (CVP).

**14.3.2 Aprendizado com erros (LWE), uma base para criptografia?**

Em 2005, Oded Regev introduziu o problema de **aprendizado com erros (LWE — Learning With Errors)**, que se tornou a base para muitos esquemas criptográficos, incluindo alguns dos algoritmos deste capítulo. Antes de prosseguir, vejamos do que se trata o problema LWE. Comecemos com as seguintes equações, que são combinações lineares dos mesmos inteiros s₀ e s₁:

* 5s₀ + 2s₁ = 27
* 2s₀ + 0s₁ = 6

Sabemos que, usando o algoritmo de eliminação de Gauss, podemos aprender rápida e eficientemente quais são os valores de s₀ e s₁, desde que tenhamos equações suficientes. Agora, o interessante é que se adicionarmos um pouco de ruído a essas equações, o problema se torna muito mais difícil:

* 5s₀ + 2s₁ = 28
* 2s₀ + 0s₁ = 5

Embora provavelmente não seja tão difícil descobrir a resposta com mais equações ruidosas, o problema se torna difícil uma vez que aumentamos o tamanho dos números envolvidos e a quantidade de variáveis sᵢ.

Essencialmente, este é o problema LWE, embora frequentemente enunciado com vetores em vez de números isolados. Imagine que você tem um vetor secreto s com coordenadas módulo algum número grande. Dado um número arbitrário de vetores aleatórios aᵢ de mesmo tamanho e os cálculos aᵢs + eᵢ, onde eᵢ é um pequeno erro aleatório, você consegue encontrar o valor de s?

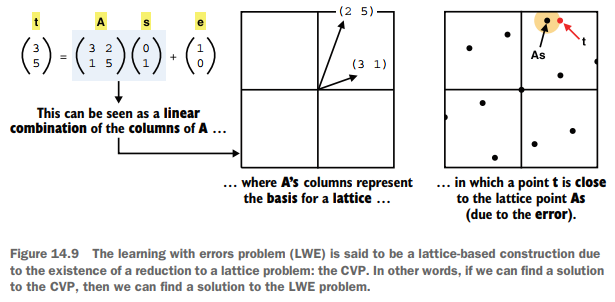
**NOTA:** Para dois vetores v e w, o produto vw pode ser calculado usando um **produto escalar**, que é a soma do produto de cada par de coordenadas. Vejamos um exemplo: se v = (v₀, v₁) e w = (w₀, w₁), então vw = v₀ × w₀ + v₁ × w₁.

Por exemplo, se eu usar o segredo s = (3,6) e fornecer os vetores aleatórios a₀ = (5,2) e a₁ = (2,0), obtenho de volta as equações com as quais iniciei o exemplo. Como eu disse anteriormente, esquemas baseados em reticulados na verdade não fazem uso dos reticulados em si; ao contrário, eles são considerados seguros se o SVP continuar difícil (para alguma definição de “difícil”). A redução só pode ser vista se escrevermos as equações anteriores em forma matricial, como mostrado na figura 14.9.

Essa forma matricial é importante, pois a maioria dos esquemas baseados em LWE são expressos e mais fáceis de explicar nessa forma. Reserve alguns minutos para revisar multiplicação de matrizes. Além disso, caso você não tenha notado, usei algumas convenções notacionais comuns que são bastante úteis para ler equações que envolvem matrizes e vetores: ambos são escritos em negrito, e matrizes sempre com letras maiúsculas. Por exemplo, **A** é uma matriz, **a** é um vetor e **b** é apenas um número.

**NOTA:** Existem várias variantes do problema LWE (por exemplo, ring-LWE ou module-LWE), que são basicamente o mesmo problema, mas com coordenadas em diferentes tipos de grupos. Essas variantes são frequentemente preferidas devido à compactação e otimizações que possibilitam. A diferença entre as variantes de LWE não afeta as explicações que seguem.

Agora que você sabe o que é o problema LWE, vamos aprender sobre algumas criptografias pós-quânticas baseadas nele: a **Suíte Criptográfica para Reticulados Algébricos (CRYSTALS)**. Convenientemente, o CRYSTALS abrange duas primitivas criptográficas: uma troca de chaves chamada **Kyber** e um esquema de assinatura chamado **Dilithium**.



\*\*Figura 14.9\*\* O problema de aprendizado com erros (LWE) é dito ser uma construção baseada em reticulados devido à existência de uma redução para um problema de reticulados: o CVP. Em outras palavras, se podemos encontrar uma solução para o CVP, então podemos encontrar uma solução para o problema LWE.

**14.3.3 Kyber, uma troca de chaves baseada em reticulados**

Dois esquemas finalistas do NIST estão intimamente relacionados: **CRYSTALS-Kyber** e **CRYSTALS-Dilithium**, que são candidatos do mesmo time de pesquisadores e ambos baseados no problema LWE. O **Kyber** é uma primitiva de criptografia de chave pública que pode ser usada como uma primitiva de troca de chaves, a qual explicarei nesta seção. O **Dilithium** é um esquema de assinatura, o qual explicarei na próxima seção. Também observe que, como esses algoritmos ainda estão em evolução, descreverei apenas as ideias e intuições por trás de ambos os esquemas.

Primeiro, vamos assumir que todas as operações acontecem em um grupo de inteiros módulo um número grande **q**. Vamos também assumir que os erros e as chaves privadas são amostrados (escolhidos uniformemente ao acaso) de um intervalo pequeno centrado em 0, que chamaremos de intervalo de erro. Especificamente, o intervalo de erro é o intervalo [-B, B], onde B é muito menor que q. Isto é importante, pois alguns termos precisam ser pequenos o bastante para serem considerados erros.

Para gerar a chave privada, basta gerar um vetor aleatório **s**, onde cada coeficiente está no intervalo de erro. A primeira parte da chave pública é uma lista de vetores aleatórios **aᵢ** do mesmo tamanho, e a segunda parte é a lista associada de produtos escalares com ruído **tᵢ = aᵢs + eᵢ mod q**. Este é exatamente o problema LWE que aprendemos anteriormente.  
Importante para o que segue: podemos reescrever isso com matrizes:

**t = A s + e**

onde a matriz **A** contém os vetores aleatórios **aᵢ** como linhas, e o vetor de erro **e** contém os erros individuais **eᵢ**.

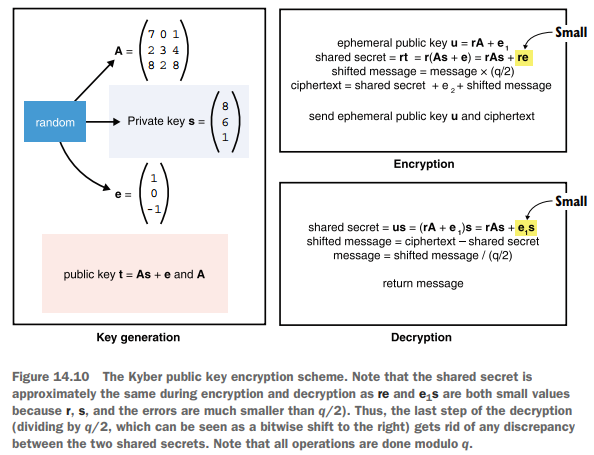
Para realizar uma troca de chaves com o Kyber, criptografamos uma chave simétrica de 1 bit (sim, um único bit!) com o esquema. Isto é semelhante ao mecanismo de encapsulamento de chave do RSA que você viu no capítulo 6. Os quatro passos a seguir mostram como a criptografia funciona:

1. Gerar um vetor de chave privada efêmero **r** (onde os coeficientes estão no intervalo de erro) e sua chave pública efêmera associada **rA + e₁** com algum vetor de erro aleatório **e₁**, usando a matriz **A** do outro par como parâmetro público. Observe que a multiplicação matricial é feita à direita, o que envolve multiplicar o vetor **r** com as colunas de **A**, em vez de computar **A r** (uma multiplicação do vetor **r** com as linhas de **A**). É um detalhe, mas necessário para o funcionamento da etapa de decriptação.
2. Deslocamos nossa mensagem para a esquerda multiplicando-a por **q/2**, para evitar que pequenos erros afetem nossa mensagem. Observe que **q/2 módulo q** normalmente significa **q** multiplicado pelo inverso de 2 módulo **q**, mas aqui significa simplesmente o inteiro mais próximo de **q/2**.
3. Computamos um segredo compartilhado com o produto escalar de nossa chave privada efêmera e a chave pública do outro par.
4. Criptografamos a mensagem (deslocada) somando-a ao segredo compartilhado, bem como a um erro aleatório **e₂**. Isso produz um texto cifrado.

Após realizar os passos, podemos enviar tanto a chave pública efêmera quanto o texto cifrado ao outro par. Após receber ambos, o par pode seguir estes passos para decifrar a mensagem:

1. Obter o segredo compartilhado computando o produto escalar de sua chave secreta com a chave pública efêmera recebida.
2. Subtrair esse segredo compartilhado do texto cifrado (o resultado contém a mensagem deslocada e um pouco de erro).
3. Deslocar a mensagem de volta para onde estava originalmente dividindo-a por **q/2**, removendo efetivamente o erro.
4. A mensagem será 1 se estiver mais próxima de **q/2** do que de 0; caso contrário, será 0.

Claro, 1 bit não é suficiente, então os esquemas atuais empregam diferentes técnicas para superar essa limitação. Recapitulei os três algoritmos na figura 14.10.



\*\*Figura 14.10\*\* O esquema de criptografia de chave pública Kyber. Observe que o segredo compartilhado é aproximadamente o mesmo durante a criptografia e a decriptação, pois \*\*r e\*\* e \*\*e₁ s\*\* são ambos valores pequenos (já que \*\*r\*\*, \*\*s\*\* e os erros são muito menores que \*\*q/2\*\*). Assim, a última etapa da decriptação (dividir por \*\*q/2\*\*, o que pode ser visto como um deslocamento bit a bit à direita) elimina qualquer discrepância entre os dois segredos compartilhados. Todas as operações são feitas módulo \*\*q\*\*.

Na prática, para uma troca de chaves, a mensagem que você criptografa na chave pública do outro par é um segredo aleatório. O resultado é então derivado de forma determinística a partir do segredo e da transcrição da troca de chaves, que inclui a chave pública do par, sua chave efêmera e o texto cifrado.

Os parâmetros recomendados para o Kyber resultam em chaves públicas e textos cifrados de cerca de 1 kilobyte, o que é muito maior que os esquemas pré-quânticos que usamos, mas ainda dentro do viável para a maioria dos casos de uso. Embora o tempo dirá se podemos reduzir ainda mais o overhead de comunicação desses esquemas, parece que, até agora, **pós-quântico rima com tamanhos maiores**.

**14.3.4 Dilithium, um esquema de assinatura baseado em reticulados**

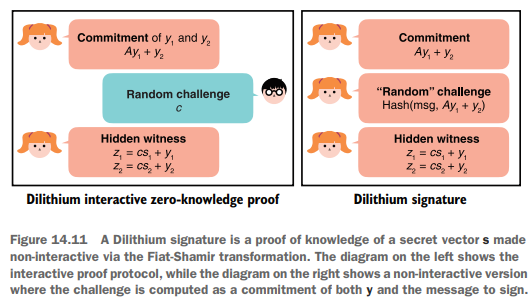
O próximo esquema que explicarei, o **Dilithium**, também é baseado em LWE e é o candidato-irmão do Kyber. Assim como outros tipos de assinaturas que vimos (como a assinatura de Schnorr no capítulo 7), o Dilithium é baseado em uma **prova de conhecimento de zero conhecimento** que é tornada não interativa via o truque de **Fiat-Shamir**.

Para a geração de chaves, o Dilithium é semelhante ao esquema anterior, exceto pelo fato de manter o erro como parte da chave privada. Primeiro geramos dois vetores aleatórios que servem como chave privada, **s₁** e **s₂**, e então computamos a chave pública como **t = A s₁ + s₂**, onde **A** é uma matriz obtida de forma semelhante ao Kyber. Note que consideramos o erro **s₂** como parte da chave privada porque precisamos reutilizá-lo sempre que assinamos uma mensagem (ao contrário do Kyber, onde o erro pode ser descartado após a geração de chaves).

Para assinar, criamos um **protocolo sigma** e então o convertemos em uma prova de zero conhecimento não interativa via a transformação de Fiat-Shamir, de forma semelhante a como o protocolo de identificação de Schnorr é convertido em uma assinatura de Schnorr no capítulo 7.  
O protocolo interativo funciona assim:

1. O provador faz um *commit* de dois vetores aleatórios, **y₁** e **y₂**, enviando **A y₁ + y₂**.
2. Após receber esse *commit*, o verificador responde com um desafio aleatório **c**.
3. O provador então calcula os dois vetores **z₁ = c s₁ + y₁** e **z₂ = c s₂ + y₂**, e os envia ao verificador somente se eles forem valores pequenos.
4. O verificador verifica se **A z₁ + z₂ – c t** e **A y₁ + y₂** são os mesmos valores.

O truque de Fiat-Shamir substitui o papel do verificador no passo 2, fazendo com que o próprio provador gere o desafio a partir de um *hash* da mensagem a ser assinada e do valor comprometido **A y₁ + y₂**. Eu recapitulo esta transformação na figura 14.11, usando um diagrama semelhante ao do capítulo 7.



\*\*Figura 14.11\*\* Uma assinatura Dilithium é uma prova de conhecimento de um vetor secreto \*\*s\*\*, tornada não interativa via a transformação de Fiat-Shamir. O diagrama à esquerda mostra o protocolo de prova interativa, enquanto o diagrama à direita mostra a versão não interativa onde o desafio é calculado como um compromisso tanto de \*\*y\*\* quanto da mensagem a ser assinada.

Novamente, esta é uma grande simplificação do esquema de assinatura. Muitas outras otimizações são usadas na prática para reduzir os tamanhos das chaves e das assinaturas. Normalmente, essas otimizações tentam reduzir quaisquer dados aleatórios gerando-os determinísticamente a partir de um valor aleatório menor, e reduzir dados não aleatórios comprimindo-os via métodos personalizados (não necessariamente via algoritmos de compressão conhecidos). Existem também várias otimizações adicionais possíveis devido à estrutura única do LWE.

No nível de segurança recomendado, o Dilithium oferece assinaturas de cerca de **3 KB** e chaves públicas de menos de **2 KB**. Obviamente, isso é muito mais do que as chaves públicas de 32 bytes e assinaturas de 64 bytes dos esquemas pré-quânticos, mas também é muito melhor do que as assinaturas baseadas em hash sem estado. É bom ter em mente que esses esquemas ainda são relativamente novos, e é possível que algoritmos melhores sejam descobertos para resolver o problema LWE, o que potencialmente aumentaria os tamanhos de chaves públicas e assinaturas. Também é possível que sejam encontradas técnicas melhores para reduzir o tamanho desses parâmetros. De modo geral, é provável que a resistência quântica sempre venha com um custo em termos de tamanho.

Isso não é tudo o que existe em criptografia pós-quântica; a competição de criptografia pós-quântica do NIST tem uma série de outras construções baseadas em diferentes paradigmas. O NIST anunciou que um padrão inicial seria publicado em 2022, mas espero que o campo continue evoluindo rapidamente, pelo menos enquanto computadores quânticos continuarem a ser vistos como uma ameaça legítima. Embora ainda haja muitas incertezas, isso também significa que há muito espaço empolgante para pesquisa. Se isso lhe interessar, recomendo dar uma olhada nos relatórios do NIST (<https://nist.gov/pqcrypto>).

**14.4 Eu preciso entrar em pânico?**

Para resumir: computadores quânticos são um grande problema para a criptografia, caso se tornem realidade. Qual é a lição aqui? Você precisa largar tudo o que está fazendo e migrar imediatamente para algoritmos pós-quânticos? Bem, não é tão simples assim.

Pergunte a qualquer especialista e você receberá respostas diferentes. Para alguns, estamos a 5 ou 50 anos de distância; para outros, isso nunca acontecerá. Michele Mosca, diretor do Instituto de Computação Quântica, estimou “uma chance de 1/7 de quebrar RSA-2048 até 2026 e uma chance de 1/2 até 2031”, enquanto Mikhail Dyakonov, pesquisador do CNRS na França, declarou publicamente:

*"Poderemos algum dia controlar mais de 10³⁰⁰ parâmetros variáveis continuamente que definem o estado quântico de um sistema desses? Minha resposta é simples. Não, nunca."*

Embora os físicos — não os criptógrafos — saibam melhor, ainda assim podem estar incentivados a promover suas próprias pesquisas para obter financiamento. Como não sou físico, direi apenas que devemos permanecer céticos em relação a afirmações extraordinárias, ao mesmo tempo em que nos preparamos para o pior. A questão não é “Vai funcionar?”, mas sim “Vai escalar?”.

Existem muitos desafios para que computadores quânticos escaláveis (que possam destruir a criptografia) se tornem realidade; o maior deles parece ser o nível de ruído e erros, que é difícil de reduzir ou corrigir. Scott Aaronson, cientista da computação da Universidade do Texas, coloca da seguinte forma:

*"Você está tentando construir um navio que permaneça o mesmo navio, mesmo quando cada tábua dele apodrece e precisa ser substituída."*

Mas e quanto ao que a NSA disse? É preciso lembrar que a necessidade de confidencialidade do governo geralmente excede as necessidades de indivíduos e empresas privadas. Não é absurdo pensar que o governo possa querer manter certos dados ultrassecretos classificados por mais de 50 anos. Ainda assim, isso intrigou muitos criptógrafos (veja, por exemplo, *"A Riddle Wrapped In An Enigma"*, de Neal Koblitz e Alfred J. Menezes), que se perguntaram por que deveríamos nos proteger de algo que ainda não existe ou que talvez nunca existirá.

De qualquer forma, se você está realmente preocupado e a confidencialidade de seus ativos precisa permanecer por longos períodos, não é absurdo — e é relativamente fácil — aumentar os parâmetros de todos os algoritmos criptográficos simétricos que você está usando.  
Dito isso, se você realiza uma troca de chaves para obter uma chave AES-256-GCM, essa parte assimétrica ainda é vulnerável a computadores quânticos, e proteger apenas a criptografia simétrica não será suficiente.

Para a criptografia assimétrica, ainda é cedo demais para saber o que é seguro de usar. Melhor aguardar o término da competição do NIST para obter mais criptoanálise e, consequentemente, mais confiança nesses novos algoritmos.

*Atualmente, existem vários sistemas criptográficos pós-quânticos que foram propostos, incluindo sistemas baseados em reticulados, sistemas baseados em códigos, sistemas multivariados, assinaturas baseadas em hash e outros. No entanto, para a maioria dessas propostas, mais pesquisas são necessárias para obter maior confiança em sua segurança (particularmente contra adversários com computadores quânticos) e para melhorar seu desempenho.*  
— Chamada de propostas de Criptografia Pós-Quântica do NIST (2017)

Se você é impaciente e não consegue esperar pelo resultado da competição do NIST, uma coisa que pode fazer é usar simultaneamente um esquema atual e um esquema pós-quântico em seu protocolo. Por exemplo, você pode coassinar mensagens usando Ed25519 e Dilithium, ou seja, anexar a mensagem com duas assinaturas de dois esquemas de assinatura diferentes. Se, eventualmente, o Dilithium for quebrado, um atacante ainda teria que quebrar o Ed25519; e se os computadores quânticos se tornarem realidade, o atacante ainda teria a assinatura Dilithium que não conseguiria forjar.

**NOTA:** Foi exatamente isso que o Google fez em 2018, e novamente em 2019, com a Cloudflare, experimentando um esquema híbrido de troca de chaves em conexões TLS entre uma pequena porcentagem de usuários do Google Chrome e servidores tanto do Google quanto da Cloudflare. O esquema híbrido foi uma mistura de X25519 e uma troca de chaves pós-quântica (New Hope em 2018, HRSS e SIKE em 2019), onde a saída de ambas as trocas de chaves eram misturadas no HKDF para produzir um único segredo compartilhado.

Finalmente, vou reforçar que as assinaturas baseadas em hash são bem estudadas e bem compreendidas. Embora apresentem certo overhead, esquemas como **XMSS** e **SPHINCS+** podem ser utilizados hoje, e o XMSS já possui padrões prontos (RFC 8391 e NIST SP 800-208).

**Resumo**

* Computadores quânticos são baseados em física quântica e podem oferecer uma aceleração não desprezível para certos cálculos.
* Nem todos os algoritmos podem rodar em um computador quântico e nem todos conseguem competir com um computador clássico. Dois algoritmos notáveis que preocupam os criptógrafos são:
  + **O algoritmo de Shor**, que pode resolver eficientemente o problema de logaritmo discreto e de fatoração. Ele quebra a maior parte da criptografia assimétrica atual.
  + **O algoritmo de Grover**, que pode buscar eficientemente uma chave ou valor em um espaço de 2¹²⁸ valores, impactando a maioria dos algoritmos simétricos com segurança de 128 bits. Aumentar os parâmetros de segurança para 256 bits é suficiente para frustrar os ataques quânticos.
* O campo da criptografia pós-quântica busca encontrar novos algoritmos criptográficos para substituir as primitivas assimétricas atuais (por exemplo, criptografia assimétrica, trocas de chaves e assinaturas digitais).
* O NIST iniciou um esforço de padronização de criptografia pós-quântica em 2016. Atualmente, há sete finalistas e o esforço está na rodada final de seleção.
* Assinaturas baseadas em hash são esquemas de assinatura que utilizam apenas funções hash. Os dois principais padrões são XMSS (com estado) e SPHINCS+ (sem estado).
* A criptografia baseada em reticulados é promissora, pois oferece chaves e assinaturas menores. Dois candidatos promissores são baseados no problema LWE: **Kyber** (para criptografia assimétrica e troca de chaves) e **Dilithium** (para assinaturas).
* Outros esquemas pós-quânticos existem e estão sendo propostos na competição do NIST, incluindo aqueles baseados em códigos, isogenias, criptografia simétrica e polinômios multivariados. A competição está prevista para terminar em 2022, o que ainda deixa espaço para novos ataques ou otimizações.
* Não está claro quando (ou se) computadores quânticos serão eficientes o suficiente para destruir a criptografia.
* Se você precisa proteger dados por longos períodos de tempo, deveria considerar:
  + Atualizar o uso de algoritmos simétricos para parâmetros com segurança de 256 bits (ex.: AES-256-GCM no lugar de AES-128-GCM, SHA-3-512 no lugar de SHA-3-256).
  + Utilizar esquemas híbridos que misturam algoritmos pós-quânticos e pré-quânticos (ex.: assinar sempre com Ed25519 e Dilithium; realizar trocas de chave com X25519 e Kyber).
  + Utilizar assinaturas baseadas em hash como XMSS e SPHINCS+, que já são bem estudadas e compreendidas. O XMSS já possui padrões estabelecidos e aprovados pelo NIST.