UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS REITOR JOÃO DAVID FERREIRA LIMA
PROGRAMA DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

João Pedro Adami do Nascimento

DESEMPENHO DE ALGORITMOS PÓS-QUÂNTICOS CANDIDATOS À PADRONIZAÇÃO NO KEMTLS HÍBRIDO

Florianópolis, Santa Catarina – Brasil
2024

João Pedro Adami do Nascimento

DESEMPENHO DE ALGORITMOS PÓS-QUÂNTICOS CANDIDATOS À PADRONIZAÇÃO NO KEMTLS HÍBRIDO

Trabalho de Conclusão de Curso submetido ao Programa de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador: Ricardo Felipe Custódio, Dr.
Coorientador: Alexandre Augusto Giron, Dr.

Florianópolis, Santa Catarina – Brasil
2024

Notas legais:

Não há garantia para qualquer parte do software documentado. Os autores tomaram cuidado na preparação desta tese, mas não fazem nenhuma garantia expressa ou implícita de qualquer tipo e não assumem qualquer responsabilidade por erros ou omissões. Não se assume qualquer responsabilidade por danos incidentais ou consequentes em conexão ou decorrentes do uso das informações ou programas aqui contidos.

Catalogação na fonte pela Biblioteca Universitária da Universidade Federal de Santa Catarina.
Arquivo compilado às 19:32h do dia 17 de dezembro de 2024.

João Pedro Adami do Nascimento
Desempenho de algoritmos pós-quânticos candidatos à padronização no KEMTLS híbrido / João Pedro Adami do Nascimento; Orientador, Ricardo Felipe Custódio, Dr.; Coorientador, Alexandre Augusto Giron, Dr. – Florianópolis, Santa Catarina – Brasil, 10 de Dezembro de 2024.
114 p.

Trabalho de Conclusão de Curso – Universidade Federal de Santa Catarina, INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico, Programa de Graduação em Ciências da Computação.

Inclui referências

1. Criptografia pós-quântica híbrida, 2. KEMTLS, 3. TLS pós-quântico, I. Ricardo Felipe Custódio, Dr. II. Alexandre Augusto Giron, Dr. III. Programa de Graduação em Ciências da Computação IV. Desempenho de algoritmos pós-quânticos candidatos à padronização no KEMTLS híbrido

CDU 02:141:005.7

João Pedro Adami do Nascimento

DESEMPENHO DE ALGORITMOS PÓS-QUÂNTICOS CANDIDATOS À PADRONIZAÇÃO NO KEMTLS HÍBRIDO

Este(a) Trabalho de Conclusão de Curso foi julgado adequado(a) para obtenção do Título de Bacharel em Ciências da Computação, e foi aprovado em sua forma final pelo Programa de Graduação em Ciências da Computação do INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico da Universidade Federal de Santa Catarina.

Florianópolis, Santa Catarina – Brasil, 10 de Dezembro de 2024.

Lúcia Helena Martins Pacheco, Dr.ª
Coordenadora do Programa de Graduação em Ciências da Computação

Banca Examinadora:

Ricardo Felipe Custódio, Dr.
Orientador
Universidade Federal de Santa Catarina – UFSC

Alexandre Augusto Giron, Dr.
Coorientador
Universidade Tecnológica Federal do Paraná – UTFPR

Lucas Perin, Dr.
Technology Innovation Institute – TII
Emirados Árabes Unidos (UAE)

Thiago Leucz Astrizi, Me.
Universidade Federal de Santa Catarina – UFSC

Dedico este trabalho aos meus melhores amigos, Rafael, Bazzi e Yan.

AGRADECIMENTOS

Agradeço ao meu orientador, Dr. Ricardo Felipe Custódio, e coorientador, Dr. Alexandre Augusto Giron, por sempre terem apoiado o meu trabalho, respeitado a minha opinião como pesquisador, e por terem aberto portas para o meu desenvolvimento profissional.

Agradeço à minha mãe, Vanessa, por nunca ter medido esforços para me proporcionar uma boa educação e uma vida confortável.

Agradeço ao meu pai, Márcio, por sempre ter ouvido minhas dores com o coração.

Agradeço aos meus amigos, Rafael, Bazzi e Yan, por encherem meu coração de amor.

“O luar clareia a noite e encanta quem a observa.
Mas somente em sua ausência, na escuridão, é possível enxergar as estrelas.”
João Pedro Nascimento

RESUMO

A confidencialidade, integridade e autenticidade são fundamentais para as comunicações pela internet na atualidade. Para garantir essas propriedades, o protocolo HTTPS foi desenvolvido, adicionando ao HTTP uma camada de criptografia por meio do TLS. Este protocolo utiliza criptografia assimétrica para estabelecer segredos compartilhados e autenticar os participantes. No entanto, avanços na computação quântica ameaçam a criptografia assimétrica tradicional, pois computadores quânticos podem quebrar seus algoritmos. Em resposta, a criptografia pós-quântica propõe algoritmos resistentes a ataques quânticos. O processo de padronização dessa nova classe de algoritmos foi iniciado pelo NIST em 2016 e encontra-se atualmente na quarta rodada, com novos candidatos a serem padronizados. Apesar dos rigorosos testes de segurança realizados durante a padronização, ainda não há confiança total nos algoritmos pós-quânticos, motivando a adoção de versões híbridas que combinam criptografia clássica e pós-quântica. Essas versões garantem segurança enquanto pelo menos um dos algoritmos empregados permanecer seguro. As principais implementações no TLS são o TLS pós-quântico híbrido e o KEMTLS híbrido. Este trabalho integra os algoritmos candidatos da quarta rodada no KEMTLS híbrido e avalia seu desempenho em um ambiente realista, comparando-o com o TLS híbrido. Os resultados indicam que o algoritmo BIKE oferece o melhor desempenho para implementação nesses protocolos. Além disso, na comparação entre KEMTLS híbrido e TLS híbrido, considerando configurações sem cache de certificados e usando BIKE, o KEMTLS híbrido apresentou melhor desempenho nos níveis de segurança 3 e 5, enquanto o TLS híbrido foi superior no nível de segurança 1.

Palavras-chaves: Criptografia pós-quântica híbrida. KEMTLS. TLS pós-quântico.

ABSTRACT

Confidentiality, integrity, and authenticity are essential for modern internet communications. The HTTPS protocol was developed to ensure these properties, adding a layer of encryption to HTTP through TLS. This protocol uses asymmetric cryptography to establish shared secrets and authenticate participants. However, advancements in quantum computing threaten traditional asymmetric cryptography, as quantum computers can break its algorithms. In response, post-quantum cryptography introduces algorithms designed to resist quantum attacks. The NIST launched the Post-Quantum Cryptography Standardization process in 2016, currently in its fourth round with new candidates under evaluation. Despite the extensive testing during the standardization process, full trust in post-quantum algorithms has not yet been established, motivating the adoption of hybrid cryptography, which combines classical and post-quantum algorithms. These hybrid versions ensure security as long as at least one component algorithm remains secure. The primary implementations in TLS are Hybrid Post-Quantum TLS and Hybrid KEMTLS. This study integrates the fourth-round candidate algorithms into Hybrid KEMTLS and evaluates their performance in a realistic environment, comparing it to Hybrid TLS. Results indicate that the BIKE algorithm offers the best performance for deployment in these protocols. Additionally, in comparing Hybrid KEMTLS and Hybrid TLS—without certificate caching and using BIKE—Hybrid KEMTLS showed better performance at security levels 3 and 5. At the same time, Hybrid TLS was superior at security level 1.

Keywords: Hybrid Post-Quantum Cryptography. KEMTLS. Post-Quantum TLS.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

HTTP	Protocolo de Transferência de Hipertexto
HTTPS	Protocolo de Transferência de Hipertexto Seguro
SSL	Protocolo *Secure Sockets Layer*
TLS	Segurança da Camada de Transporte
PSK	Chave pré-compartilhada
HKDF	Função de derivação de chave baseada em HMAC
IKM	*Input Keying Material*
RTT	*Round Trip Time*
SSH	*Secure Shell*
AES	Padrão de Criptografia Avançada
RSA	*Rivest-Shamir-Adleman*
NIST	*National Institute of Standards and Technology*
IETF	*Internet Engineering Task Force*
PQTLS	TLS pós-quântico
KEMTLS-PDK	KEMTLS com chaves públicas pré-distribuídas
KEM	Mecanismo de encapsulamento de chave
MAC	Código de autenticação de mensagem
HMAC	Código de autenticação de mensagem com chave hash
(EC)DHE	Diffie-Hellman efêmero sobre curvas elípticas ou grupos finitos
DH	Diffie-Hellman
IoT	Internet das Coisas
CRQC	Computador Quântico Criptograficamente Relevante
CDN	Rede de Distribuição de Conteúdo
GCP	*Google Cloud Platform*
ICP	Infraestrutura de Chaves Públicas
API	Interface de Programação de Aplicação

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A internet, desde seu início, esteve estreitamente ligada ao protocolo *Hypertext Transfer Protocol* (HTTP) (FIELDING; NOTTINGHAM; RESCHKE, 2022), um protocolo de comunicação utilizado para a transferência de dados na web. Com o HTTP, navegadores podem solicitar recursos de servidores, como páginas web, e exibi-los aos usuários. À medida que crescia a preocupação com a segurança dos dados transmitidos pela internet, em 1994, a empresa Netscape Communications introduziu uma camada adicional para transmissão de dados criptografados, o SSL (*Secure Sockets Layer*) (DOCS, s.d.). No momento de seu lançamento, o SSL viabilizou principalmente o surgimento de aplicações de *e-commerce*, embora seu uso não tenha se restringido a essa área.

Com o crescimento da internet e o aumento de ataques cibernéticos e roubo de informações privadas, tornou-se evidente a necessidade de uma camada de transporte segura para todas as aplicações. Nesse contexto, surgiu o HTTPS, uma extensão do HTTP que garante a confidencialidade e integridade dos dados transmitidos, além da autenticação das partes envolvidas na conexão. Para isso, é utilizado o protocolo criptográfico TLS (*Transport Layer Security*) (RESCORLA, 2018), uma versão atualizada do SSL padronizada pela *Internet Engineering Task Force* (IETF). O TLS foi inicialmente definido em 1999 e, desde então, passou por várias atualizações, estando atualmente na versão 1.3, publicada em 2018.

No TLS, de modo geral, são utilizadas duas classes de criptografia para garantir a confidencialidade e a autenticidade da comunicação: a criptografia simétrica e a criptografia assimétrica. Na criptografia simétrica, tanto o remetente quanto o destinatário compartilham a mesma chave secreta, utilizando-a para cifrar e decifrar os dados transmitidos na conexão (STALLINGS, 2013). Essa classe é usada especificamente para proteger os pacotes de dados trocados durante a comunicação.

Por outro lado, a criptografia assimétrica faz uso de chaves distintas – uma pública e outra privada – para cifragem e decifragem. No contexto do TLS, sua aplicação ocorre principalmente em algoritmos de assinatura digital e nos métodos de troca de chaves (*key exchange*), essenciais para estabelecer segredos compartilhados de forma segura em um canal público.

Nos últimos anos, o interesse e os investimentos em computação quântica aumentaram significativamente, tanto na pesquisa acadêmica quanto na indústria. A computação quântica é um campo da ciência da computação que utiliza princípios da mecânica quântica em seus processos, explorando propriedades únicas da física quântica para resolver certos problemas de forma mais eficiente do que a computação clássica (AWS, s.d.). Recentemente, empresas como a Google anunciaram ter alcan-

çado a supremacia quântica (ARUTE *et al.*, 2019), isto é, a capacidade de um computador quântico resolver um problema específico mais rapidamente do que qualquer computador clássico. Esse marco abre caminho para grandes avanços, permitindo o desenvolvimento de aplicações computacionais que anteriormente eram consideradas impraticáveis.

No entanto, a capacidade dos computadores quânticos de resolver determinados problemas computacionais de forma eficiente representa uma ameaça à criptografia contemporânea (WILTON, 2020). Na criptografia simétrica, algoritmos como o AES podem ser vulneráveis ao ataque do algoritmo quântico de Grover (GROVER, 1996), que é capaz de reduzir o tempo necessário para realizar uma busca exaustiva por uma chave simétrica, enfraquecendo a segurança do esquema criptográfico. No entanto, esse ataque não é suficiente para comprometer a criptografia simétrica, pois, além de o algoritmo de Grover ser computacionalmente custoso, uma simples alteração nos parâmetros da chave simétrica pode torná-la tão difícil de quebrar para um computador quântico quanto seria para um computador clássico (LAING; CHARLES, 2024). Assim, a criptografia simétrica não é considerada atualmente uma área sob ameaça crítica dos computadores quânticos.

A criptografia assimétrica, por outro lado, encontra-se sob uma ameaça significativa com o advento dos computadores quânticos. Os algoritmos dessa classe são baseados em problemas matemáticos, como o logaritmo discreto e a fatoração de inteiros, cuja segurança depende da dificuldade que computadores clássicos enfrentam para resolvê-los de forma eficiente. No entanto, a computação quântica, por meio do algoritmo de Shor (SHOR, 1994), é capaz de resolver esses problemas com complexidade assintótica polinomial, comprometendo assim a segurança criptográfica desses algoritmos. Isso representa uma grave ameaça para a segurança da internet, pois a criptografia assimétrica é fundamental para a maioria dos protocolos de comunicação segura em rede.

Como resposta à ameaça dos computadores quânticos, surgiu a criptografia pós-quântica, uma criptografia baseada em problemas matemáticos que não possuem solução eficiente conhecida nem em computadores clássicos e nem em computadores quânticos. Em 2016, o *National Institute of Standards and Technology* (NIST) anunciou o processo de padronização de criptografia pós-quântica (NIST, 2024), uma competição na qual algoritmos resistentes a ataques quânticos são submetidos a testes intensivos e análises rigorosas, com o objetivo de selecionar os melhores para aplicações do mundo real. Após seis anos de competição, em 2022 foram anunciados quatro vencedores[1], sendo três algoritmos de assinatura digital e um de *Key Encapsulation Mechanism* (KEM). A padronização de apenas um algoritmo de KEM levou o NIST a

[1] `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`

lançar uma quarta rodada[2] para avaliar novos candidatos a esse tipo de algoritmo.

Apesar de os algoritmos padronizados terem passado por uma série de testes e revisões durante o processo de padronização, a confiança neles ainda não foi plenamente estabelecida. Além de serem relativamente novos, diversas vulnerabilidades foram identificadas durante o processo de padronização (BEULLENS, 2022; CASTRYCK; DECRU, 2023), o que reforça essa desconfiança. Em contraste, algoritmos de criptografia clássica, como o RSA (RIVEST; SHAMIR; ADLEMAN, 1978), publicado em 1977, possuem uma robustez e confiança bem maiores, pois têm sido amplamente utilizados no mundo real há mais de 40 anos. Nesse contexto, surge a criptografia pós-quântica híbrida, que combina algoritmos clássicos e pós-quânticos para garantir segurança enquanto ao menos um dos componentes (clássico ou pós-quântico) permanecer seguro. A adoção da criptografia híbrida é recomendada para aplicações do mundo real, pois preserva a confiança nos algoritmos clássicos, ao mesmo tempo em que protege a comunicação contra futuros ataques de computadores quânticos.

A adoção de algoritmos pós-quânticos nos protocolos de comunicação segura não é trivial, pois esses algoritmos, em geral, exigem maior poder computacional e utilizam chaves significativamente maiores em comparação com os algoritmos de criptografia clássica, como o RSA (SIKERIDIS; KAMPANAKIS; DEVETSIKIOTIS, 2020). Essas características impactam diretamente o desempenho de protocolos como o TLS, tornando necessária a adaptação dos protocolos existentes ou até o desenvolvimento de alternativas mais adequadas para a criptografia pós-quântica. Nesse contexto, diversas propostas surgiram na literatura, sendo uma das mais proeminentes o TLS pós-quântico, abordado principalmente em (CROCKETT; PAQUIN; STEBILA, 2019), no qual foi desenvolvido um protótipo do TLS pós-quântico (PQTLS) e do TLS pós-quântico híbrido (*Hybrid* PQTLS).

No TLS pós-quântico, são empregados algoritmos pós-quânticos de *Key Encapsulation Mechanism* (KEM) para estabelecer as chaves secretas utilizadas na criptografia do protocolo, e algoritmos pós-quânticos de assinatura para autenticar as partes comunicantes. No TLS pós-quântico híbrido, por sua vez, algoritmos pós-quânticos são combinados com algoritmos clássicos: na troca de chaves, o algoritmo Diffie-Hellman é utilizado em conjunto com um KEM pós-quântico; para autenticação, algoritmos de assinatura clássicos (como o RSA) são combinados com algoritmos de assinatura pós-quânticos.

Enquanto o protótipo apresentado em (CROCKETT; PAQUIN; STEBILA, 2019) buscou integrar a criptografia pós-quântica no TLS sem modificar a estrutura do protocolo, outros trabalhos propuseram alterações para adaptá-lo às características dessa nova criptografia. Um exemplo é o KEMTLS, proposto em (SCHWABE; STEBILA; WIGGERS, 2020), que oferece uma alternativa ao *handshake* do TLS 1.3, substituindo

[2] `https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions`

assinaturas digitais por KEMs para a autenticação. Essa modificação visa otimizar o TLS para a criptografia pós-quântica, uma vez que algoritmos de assinatura pós-quânticos possuem chaves grandes, e sua transmissão durante o *handshake* pode impactar negativamente o desempenho do protocolo.

Com o objetivo de avaliar o impacto da criptografia pós-quântica híbrida no KEMTLS, o estudo apresentado em (GIRON *et al.*, 2023) propôs o KEMTLS híbrido (*Hybrid* KEMTLS), uma adaptação que adiciona suporte a algoritmos clássicos de KEM para serem usados em conjunto com algoritmos pós-quânticos, tanto na troca de chaves quanto na autenticação das partes.

Atualmente, os computadores quânticos não representam uma ameaça ativa à criptografia, mas já são motivo de grande preocupação. A migração da infraestrutura da internet e de outras redes de computadores para a criptografia pós-quântica é extremamente complexa e onerosa, e deve ser iniciada o quanto antes, mesmo que computadores quânticos suficientemente poderosos possam surgir apenas nas próximas décadas, segundo especialistas (MOSCA; PIANI, 2023).

Além disso, computadores quânticos apresentam uma ameaça passiva por meio de ataques retroativos, como o *store now, decrypt later* (BEVERIDGE; BUTCHER, 2023). Nesse tipo de ataque, pacotes de dados transmitidos atualmente são interceptados e armazenados para futura descriptografia, quando houver acesso a um computador quântico suficientemente poderoso. Por isso, a adoção da criptografia pós-quântica é necessária antes mesmo do advento de computadores quânticos capazes de quebrar os algoritmos atuais.

Diante desse cenário, a prototipação e experimentação da criptografia pós-quântica em protocolos de comunicação segura emergem como ferramentas essenciais para coletar dados que orientem de forma mais eficaz a migração da infraestrutura atual para uma infraestrutura pós-quântica. Esse processo permite identificar os principais desafios e avaliar o impacto dessa migração nas comunicações seguras.

Na literatura, o protocolo KEMTLS híbrido foi implementado e avaliado utilizando os algoritmos finalistas do processo de padronização do NIST. Considerando que, entre os algoritmos de KEM finalistas, apenas um foi padronizado, torna-se necessário avaliar esse protocolo com os novos algoritmos de KEM candidatos à padronização, ou seja, os algoritmos da quarta rodada. Assim, este trabalho tem como objetivo instanciar os algoritmos candidatos no KEMTLS híbrido e avaliar seu desempenho por meio de experimentos comparativos com o protocolo TLS pós-quântico híbrido.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho de conclusão de curso tem como objetivo instanciar o protocolo KEMTLS híbrido utilizando algoritmos de KEM candidatos à padronização da quarta rodada do processo de padronização da criptografia pós-quântica do NIST.

1.2.2 Objetivos Específicos

- Contextualizar a adoção da criptografia pós-quântica no protocolo TLS;
- Analisar o design e a implementação do protocolo KEMTLS híbrido;
- Avaliar o desempenho dos algoritmos candidatos à padronização;
- Integrar os algoritmos candidatos no KEMTLS híbrido;
- Mensurar o desempenho dos protocolos KEMTLS híbrido e TLS pós-quântico híbrido, instanciados com os algoritmos candidatos, em um ambiente realista;
- Mensurar o desempenho dos protocolos KEMTLS-PDK híbrido e TLS pós-quântico híbrido, com a extensão *Cached Information*, instanciados com os algoritmos candidatos, em um ambiente realista.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 TLS

O *Transport Layer Security* (TLS) é um protocolo criptográfico amplamente utilizado para garantir a confidencialidade, a autenticação e a integridade dos dados transmitidos em conexões na internet (RESCORLA, 2018). Sua principal aplicação é no protocolo HTTPS, uma extensão do *Hypertext Transfer Protocol* (HTTP) (FIELDING; NOTTINGHAM; RESCHKE, 2022), que adiciona uma camada de criptografia fornecida pelo TLS. A versão mais recente do protocolo é a 1.3, publicada em 2018.

O TLS é composto por duas camadas principais: o *TLS Record Protocol* e o *TLS Handshake Protocol*. O *TLS Record Protocol* opera na camada de transporte e tem como principal função estabelecer uma conexão segura entre as partes envolvidas na comunicação. Ele recebe os dados da aplicação, divide-os em fragmentos menores e adiciona um cabeçalho contendo informações como o tipo de conteúdo, a versão do TLS e o tamanho do fragmento. Em seguida, gera um *Message Authentication Code* (MAC) (STALLINGS, 2013) para o fragmento, anexa-o ao cabeçalho e, se necessário, realiza a cifragem do fragmento para garantir sua confidencialidade.

O *TLS Record Protocol* assegura a integridade e autenticidade dos dados aplicando um MAC a cada fragmento e protege a confidencialidade por meio da cifragem com uma chave simétrica, que é estabelecida durante o *TLS Handshake Protocol*.

2.1.1 TLS 1.3 *Handshake Protocol*

O TLS 1.3 *Handshake Protocol*, representado na Figura 1, é responsável pela negociação de parâmetros de segurança entre o cliente e o servidor e o estabelecimento das chaves de criptografia do protocolo. Ele ocorre no início de uma conexão TLS 1.3 e envolve várias etapas: negociação de algoritmos de criptografia, geração de segredo compartilhado, derivação de chaves de criptografia da conexão, autenticação do servidor, autenticação do cliente, entre outros (RESCORLA, 2018).

A fim de derivar as chaves de criptografia utilizadas para criptografar os pacotes TLS, o protocolo realiza a troca de chaves por meio do algoritmo de Diffie-Hellman (DIFFIE; HELLMAN, 1976), mais especificamente o Diffie-Hellman efêmero sobre curvas elípticas ou corpos finitos, (EC)DHE. A troca de chaves é um método de se acordar uma chave secreta entre duas entidades de maneira segura em um canal inseguro (a internet, por exemplo). O segredo obtido por este método é chamado de segredo compartilhado (pois tanto o cliente como o servidor computam o mesmo segredo) e este é utilizado para derivar as chaves de criptografia simétrica que criptografam os pacotes de dados da conexão.

Já para realizar a autenticação das entidades do protocolo (cliente e servidor),

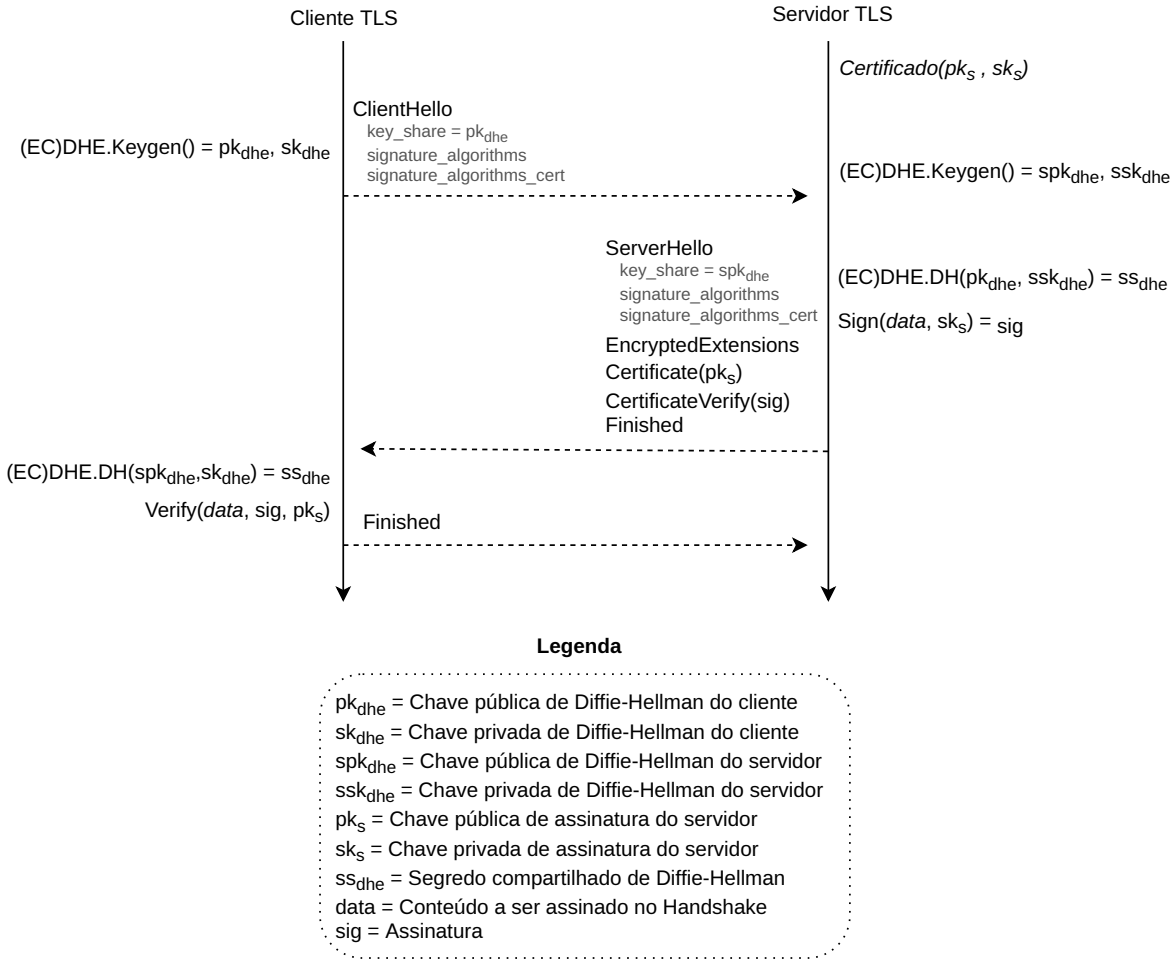


Figura 1 – Protocolo de *handshake* do TLS 1.3 (apenas autenticação do servidor)

são utilizados certificados digitais X.509 (BOEYEN *et al.*, 2008) e assinaturas digitais feitas com estes, provando desta forma que a entidade tem posse do certificado que apresenta.

O protocolo de *handshake* é baseado na troca de mensagens entre o cliente e o servidor. A primeira mensagem do protocolo é enviada pelo cliente, e se chama *ClientHello*. Esta mensagem contém, principalmente, a versão TLS suportada pelo cliente, uma lista de algoritmos criptográficos sugeridos e uma lista de extensões TLS. Algumas mensagens do *handshake* possuem suporte a extensões, que são responsáveis por prover determinadas funcionalidades ao protocolo. No *ClientHello* do TLS 1.3, as seguintes extensões são fundamentais:

- `supported_groups`: indica os algoritmos de troca de chaves suportados pelo cliente;
- `signature_algorithms`: contém a lista de algoritmos de assinatura suportados pelo cliente para serem usados no *handshake*;
- `signature_algorithms_cert`: contém a lista de algoritmos de assinatura suportados pelo cliente para serem usados na cadeia de certificados enviada pelo servidor;

- `key_share`: contém a chave pública de Diffie-Hellman do cliente, necessária para o método de troca de chaves de Diffie Hellman. Na Figura 1 esta chave está representada pelo símbolo pk_{dhe}.

Ao enviar a mensagem *ClientHello* para o servidor, este responde com a mensagem *ServerHello*, que é responsável por finalizar a primeira etapa de negociação do protocolo. A mensagem *ServerHello* contém em seu corpo, principalmente, a cifra simétrica selecionada pelo servidor a partir da lista de cifras simétricas fornecidas pelo cliente. Assim como o *ClientHello*, o *ServerHello* faz o uso de algumas extensões fundamentais para o andamento do protocolo:

- `key_share`: contém a chave pública de Diffie-Hellman do servidor, representada na Figura 1 pelo símbolo spk_{dhe}.

A partir deste momento, tanto o cliente como o servidor possuem a chave pública de Diffie-Hellman um do outro, e portanto podem executar o algoritmo de trocas de chaves de Diffie-Hellman (representado pela função `(EC)DHE.DH()` na Figura 1). Após a execução deste algoritmo, o cliente e o servidor terão computado um segredo compartilhado, ss_{dhe}, que por sua vez será derivado em uma chave de criptografia simétrica que irá criptografar as mensagens subsequentes.

A próxima mensagem enviada pelo servidor é a *EncryptedExtensions*, que, como o nome sugere, contém extensões que podem ser criptografadas, e deste modo não são necessárias para estabelecer as chaves de criptografia da conexão. Juntamente com esta mensagem, o servidor pode enviar a mensagem *CertificateRequest* a fim de solicitar ao cliente que este se autentique com um certificado.

Após estas mensagens, o protocolo entra na fase de autenticação, a qual é iniciada pelo servidor a partir do envio da mensagem *Certificate*. *Certificate* contém o certificado X.509 do servidor juntamente com a sua cadeia de certificação (autoridade certificadora emissora e autoridades certificadoras intermediárias), e é responsável por identificar o servidor. Nesta cadeia de certificação, não está contida a autoridade certificadora raiz, visto que esta deve ser distribuída de forma independente do protocolo. De modo a provar que o servidor está em posse da chave privada do certificado e consequentemente se autenticar perante o cliente, é enviada a mensagem *CertificateVerify*. Esta mensagem contém uma assinatura realizada com a chave privada (sk_s) do servidor sobre o *handshake transcript*, que corresponde ao *hash* das mensagens enviadas no *handshake*. A partir dessa assinatura, o cliente pode verificar a identidade do servidor fornecida pelo certificado.

Por fim, do lado do servidor, é gerada a mensagem *Finished*, que provê a autenticação do *handshake* e das chaves criptográficas computadas. O conteúdo desta mensagem é um HMAC (KRAWCZYK; BELLARE; CANETTI, 1997) do *handshake transcript* realizado com a chave `finished_key`, derivada do segredo `Master Secret` do *key*

schedule (Figura 2).

O cliente, ao receber a mensagem *Finished* do servidor, gera sua mensagem *Finished* da mesma forma que este e envia ao servidor, finalizando o protocolo de *handshake* do TLS 1.3.

2.1.2 *Key schedule*

O TLS 1.3 utiliza um processo de derivação de chaves chamado TLS *key schedule*, que recebe como entrada o segredo compartilhado computado no *handshake* (*Shared Secret* na Figura 2), o *handshake transcript*, e opcionalmente uma *pre-shared key* (SESSION. . . , s.d.) (*PSK* na Figura 2), e deriva este material em chaves de criptografia simétrica para serem utilizadas na criptografia da conexão. Este mecanismo é baseado em funções de HKDF-Extract e HKDF-Expand (KRAWCZYK; ERONEN, 2010) e nas funções definidas abaixo:

Código 1 – Funções adicionais do TLS *key schedule*

```
1 Derive-Secret(Secret, Label, Messages) =
2 HKDF-Expand-Label(Secret, Label,Transcript-Hash(Messages),
     Hash.length)
3
4 HKDF-Expand-Label(Secret, Label, Context, Length) =
5 HKDF-Expand(Secret, HkdfLabel, Length)
```

No Código 1, o `HkdfLabel` trata-se apenas do `Label` formatado em uma estrutura específica do protocolo.

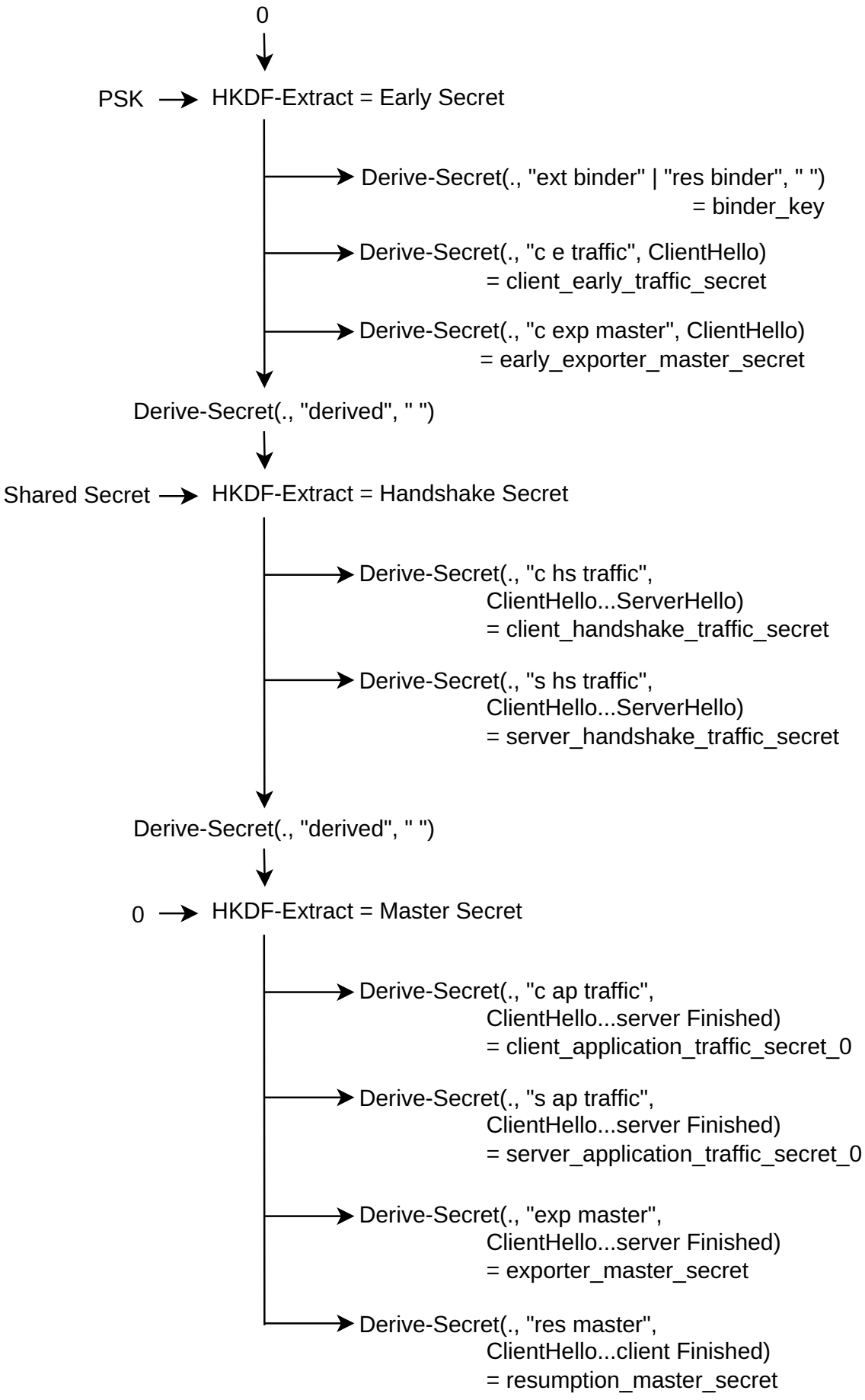


Figura 2 – *Key schedule* do TLS 1.3

No processo de derivação de chaves representado na Figura 2, uma função HKDF-Extract recebe como argumentos um *salt*[1] e *input keying material*[2] (IKM) e gera um valor de saída. No diagrama, o *salt* foi representado pela seta superior, o IKM pela seta esquerda, e a saída pela seta inferior. A saída da função HKDF-Extract é utilizada como o segredo de entrada da função Derive-Secret, que além desse segredo, recebe como entrada um label e o *handshake transcript* de determinada mensagem ou sequência de mensagens.

O *key schedule* é responsável por gerar os segredos do protocolo, que por sua vez são derivados em chaves de criptografia simétrica utilizadas para criptografar as mensagens e dados enviados no protocolo.

2.1.2.1 Geração das chaves de criptografia da conexão

A derivação de segredos em chaves de criptografia é feita por meio da aplicação de uma função HKDF-Expand-Label (definida no código 1) ao segredo do *key schedule* correspondente ao tipo de chave de criptografia a ser gerada. O TLS 1.3 oferece três tipos de chaves de criptografia de acordo com o contexto do protocolo no qual ela será utilizada:

1. Criptografia de dados de aplicação da etapa 0-RTT[3]: derivada do segredo `client_early_traffic_secret`;
2. Criptografia de mensagens do *handshake*: derivada do `client_handshake_traffic_secret` ou `server_handshake_traffic_secret`;
3. Criptografia de dados da aplicação pós-*handshake*: derivada do `client_application_traffic_secret_N` ou `server_application_traffic_secret_N`, onde `N` é um inteiro que corresponde ao número de vezes que o segredo já foi atualizado por meio de uma rotina de atualização de segredos.

2.1.3 TLS 1.3 Cached Information

No TLS, o servidor precisa enviar o seu certificado ao cliente durante o *handshake* para iniciar o processo de autenticação. Isto se adequa bem para casos de uso como navegação na internet, onde um cliente se comunica com uma grande variedade de servidores de forma temporária, sem necessariamente voltar a se comunicar com estes novamente em um curto intervalo de tempo. No entanto, há casos de uso específicos em que um cliente se comunica apenas com um conjunto limitado de ser-

[1] Dado randômico usado como entrada para funções de hash aplicadas a segredos (STANDARDS; TECHNOLOGY, 2020).

[2] Dado de entrada na derivação de chave com funções HKDF.

[3] Zero Round-Trip Time Resumption (RESCORLA, 2018)

vidores, ou então se comunica com determinados servidores com bastante frequência. Para estes cenários, é adequado que o cliente armazene os certificados dos servidores localmente, para que desta forma não seja necessário o envio destes durante o *handshake*. Isto ocorre quando navegadores web fazem cache de certificados de servidores frequentemente acessados, quando dispositivos IoT ou aplicativos *mobile* armazenam os certificados dos servidores fixos que se comunicam, entre outros casos. A fim de dar suporte a estes casos de uso no TLS 1.3, foi criada a extensão *Cached Information* (SANTESSON; TSCHOFENIG, 2016) para a mensagem *ClientHello*.

A extensão *Cached Information* permite que clientes informem ao servidor que já possuem o certificado deste, não sendo necessário o envio do mesmo durante o *handshake*. Para isto, é enviado por meio dessa extensão o hash da mensagem *Certificate* do servidor que o cliente possui armazenado em sua cache. O servidor, ao processar esta extensão, verifica se o hash contido na extensão corresponde ao hash da mensagem *Certificate*. Se sim, o servidor envia ao cliente uma versão modificada da mensagem *Certificate* contendo apenas o hash da mensagem. Desta forma, o certificado do servidor e a sua cadeia de certificação correspondente não são enviados no *handshake*, diminuindo a quantidade de dados transmitidos na conexão e consequentemente diminuindo a latência do protocolo.

2.2 PROTOCOLOS DE TROCA DE CHAVES

Para que duas entidades possam se comunicar de forma confidencial em um canal inseguro, é necessário que ambas tenham conhecimento de uma chave secreta de criptografia. Quando o remetente deseja enviar uma mensagem secreta ao destinatário, este utiliza a chave secreta juntamente com uma função de criptografia para gerar um texto cifrado, que é transmitido ao destinatário por meio do canal inseguro. Ao receber a mensagem, o destinatário descriptografa-a utilizando esta mesma chave secreta, obtendo desta forma o acesso ao conteúdo da mensagem.

Para que estas entidades possam acordar uma chave de criptografia, a maneira mais simples seria realizar o envio desta chave em um canal seguro. No entanto, a internet é um canal inseguro, o que impossibilita o simples envio da chave de criptografia entre as entidades. A solução para este problema é o uso de protocolos de troca de chaves (*Key exchange*), que permitem que duas partes acordem uma chave secreta compartilhada de forma segura, mesmo que estejam se comunicando através de um canal não seguro. Para isto, é utilizada a criptografia assimétrica, na qual é gerado um par de chaves que consiste em uma chave pública, que pode ser divulgada livremente, e uma chave privada, que deve ser mantida em segredo.

2.2.1 Troca de Chaves Diffie-Hellman

Protocolos de troca de chaves são extensivamente usados em protocolos de comunicação segura da internet, tal como o TLS. No TLS 1.3, o protocolo de troca de chaves utilizado é o Diffie-Hellman (DIFFIE; HELLMAN, 1976), o qual é baseado no problema do logaritmo discreto. Este problema é formulado da seguinte maneira: dada uma base g, um expoente x, e um módulo p, encontre y tal que $g^y = x \mod p$. A segurança do Diffie-Hellman reside na dificuldade de calcular o valor do expoente y quando conhecemos g, x, e p, especialmente quando p é um número primo grande. Contudo, este problema pode ser eficientemente resolvido (em tempo polinomial) por um computador quântico, conforme demonstrado em (SHOR, 1994). Por conta disto, torna-se necessário o uso de protocolos de troca de chave baseados em problemas matemáticos que não possuem solução eficiente conhecida em computadores quânticos.

2.2.2 Troca de Chaves Pós-quântica

Na criptografia pós-quântica, os problemas matemáticos que não possuem solução eficiente conhecida em computadores quânticos são instanciados em *Key Encapsulation Mechanisms* (KEMs) (TAMVADA; CELI, 2022).

KEM é um mecanismo de criptografia assimétrica utilizado para encapsular um segredo compartilhado para transmissão em um canal inseguro. Este segredo compartilhado, por sua vez, pode ser usado posteriormente para estabelecer uma comunicação segura entre as partes por meio da geração de uma chave de criptografia simétrica. O protocolo de KEM envolve as seguintes etapas:

- **Geração de Chaves:** Um par de chaves assimétricas é gerado para cada entidade participante. A chave pública é usada para encapsular um segredo compartilhado, enquanto a chave privada é usada para desencapsular e recuperar o segredo compartilhado. A parte que deseja receber uma mensagem compartilha a sua chave pública com a parte remetente. A geração de chaves pode ser definida pela função `pk, sk = KEM.Keygen()`, na qual `pk` é a chave pública e `sk` a chave privada.

- **Encapsulamento:** A parte remetente, que deseja estabelecer uma comunicação segura, gera um segredo aleatório que será usado futuramente para derivar uma chave de criptografia simétrica. Esse segredo é encapsulado (criptografado) usando a chave pública do destinatário, resultando em uma mensagem encapsulada, comumente chamada de texto cifrado. Este texto cifrado é então transmitido à parte destinatária. O encapsulamento pode ser definido pela função `ct, ss = KEM.Encaps(pk)`, na qual `ct` é o texto cifrado e `ss` é o segredo.

- **Desencapsulamento:** O destinatário, que possui a chave privada correspondente à chave pública usada no encapsulamento, desencapsula o texto cifrado recebido por meio de uma função de desencapsulamento, que recebe como argumento a chave privada deste e o texto cifrado. O resultado do desencapsulamento é o segredo compartilhado gerado pelo remetente. O desencapsulamento pode ser definido pela função `ss = KEM.Decaps(ct, sk)`.

Desta forma, é possível realizar o compartilhamento de um segredo entre duas partes de forma segura em um canal inseguro, uma vez que é necessário transmitir no canal inseguro apenas a chave pública, que é um parâmetro público, e o texto cifrado, que é um dado criptografado e portanto seguro.

2.3 COMPUTAÇÃO QUÂNTICA

A computação quântica é a ciência que estuda o processamento de informação em sistemas quânticos, tais como átomos, fótons ou partículas subatômicas (HELERBROCK, s.d.), operando de acordo com as leis probabilísticas da física quântica. Enquanto que a física clássica descreve o comportamento das partículas ao nível macroscópico, a física quântica descreve como as partículas se comportam em um nível subatômico. Ao nível subatômico, os fenômenos físicos observados diferem dos observados à nível macroscópico, dessa forma, a computação quântica busca explorar estas diferenças para realizar computações de uma maneira inédita (AWS, s.d.). Isto permite que o computador quântico possa resolver problemas complexos que são considerados intratáveis para computadores clássicos.

Enquanto um computador clássico opera por meio de circuitos que registram e controlam a passagem de corrente elétrica, em um computador quântico a informação é obtida a partir de propriedades quânticas (subatômicas), dentre as quais podemos destacar: polarização de um fóton, nível de energia de um aglomerado de átomos, *spin* de um átomo, entre outros (HELERBROCK, s.d.). A unidade básica de informação em um computador quântico é o qubit, ou bit quântico. Em contraste com os bits clássicos, que podem existir em um estado de 0 ou 1, os qubits podem existir simultaneamente em uma superposição de estados, por consequência do princípio de superposição quântica.

2.3.1 Princípios da Computação Quântica

A computação quântica é baseada nos seguintes princípios, que caracterizam um sistema quântico:

- **Superposição:** A superposição afirma que um sistema quântico existe em múltiplos estados simultaneamente. Enquanto em sistemas clássicos um objeto

encontra-se em um estado específico (por exemplo, uma luz acesa ou apagada), um sistema quântico pode estar em uma combinação de estados ao mesmo tempo. Em um estado de superposição, o qubit pode representar uma mistura de "0"e "1"ao mesmo tempo. Somente quando o qubit é medido é que ele "escolhe"um estado específico, colapsando para 0 ou 1 com determinada probabilidade.

- **Entrelaçamento:** O entrelaçamento quântico é um fenômeno em que duas ou mais partículas ficam interligadas de modo que as propriedades de uma partícula estão instantaneamente relacionadas às propriedades de outra, independente da distância física entre essas partículas (OLIVEIRA; IGNACIO DE LIMA, 2023). A partir disso, um processador quântico pode obter conclusões sobre uma partícula a partir da medição de outra partícula, e usar este fenômeno para realizar computações de forma mais eficiente que um computador clássico (AWS, s.d.).

- **Decoerência:** A decoerência afirma que um sistema quântico que originalmente se encontra em um estado de superposição ou entrelaçamento, ao interagir com o seu ambiente (como a radiação), perde essas propriedades quânticas especiais, passando a se comportar mais como um sistema clássico (COUTINHO, 2023). A decoerência é um desafio significativo na computação quântica e em experimentos quânticos, tendo em vista que interfere na capacidade de manter estados quânticos coerentes por tempo suficiente para realizar cálculos ou observações quânticas, tornando necessário o desenvolvimento de vários recursos no computador quântico que retardem a decoerência do estado.

Essas propriedades fundamentais da física quântica, oferecem vantagens significativas que tornam os computadores quânticos potencialmente mais poderosos que os computadores clássicos em certos cenários: o computador quântico tem um alto grau de paralelismo em suas computações, pois o fenômeno da superposição permite a realização de cálculos em paralelo, processando várias combinações de bits simultaneamente; o fenômeno do entrelaçamento permite uma correlação instantânea entre qubits, e isto é explorado em algoritmos quânticos para melhorar o desempenho em certos tipos de cálculos.

2.3.2 Ameaça do computador quântico

Apesar de o desenvolvimento do computador quântico ser um grande passo no desenvolvimento científico, a vantagem computacional deste computador traz grandes implicações para determinados sistemas de criptografia, principalmente a criptografia assimétrica. A segurança da criptografia assimétrica é baseada na dificuldade de encontrar soluções para determinados problemas matemáticos, como o problema do logaritmo discreto e o problema da fatoração de inteiros.

O problema do logaritmo discreto, como visto na seção 2.2.1, envolve encontrar o expoente ao qual uma base deve ser elevada para obter um determinado resultado. Em criptografia, este problema é explorado no algoritmo de troca de chaves de Diffie-Hellman.

O problema da fatoração de inteiros, por sua vez, envolve decompor um número inteiro composto em seus fatores primos. Formalmente, dado um número inteiro N, o objetivo é encontrar os números primos p e q tais que $N = p.q$. Este problema é explorado no algoritmo RSA (RIVEST; SHAMIR; ADLEMAN, 1978), amplamente utilizado em assinaturas digitais, de tal forma que no RSA, a chave pública consiste no número inteiro composto N e a chave privada envolve os fatores primos p e q.

Atualmente, em computadores clássicos, resolver o problema do logaritmo discreto e a fatoração de inteiros para números grandes é computacionalmente desafiador, uma vez que os algoritmos conhecidos para estes problemas consomem tempo exponencial de execução em relação aos parâmetros utilizados no problema. No entanto, há um algoritmo quântico (isto é, que executa em um computador quântico), capaz de resolver estes problemas de forma eficiente, com uma complexidade assintótica polinomial em relação aos parâmetros utilizados no problema. Este algoritmo quântico é o algoritmo de Shor (SHOR, 1994), e com ele é possível fatorar números inteiros compostos grandes de forma eficiente.

O algoritmo de Shor ainda não representa uma ameaça ativa para a criptografia assimétrica clássica, tendo em vista que ainda não foi desenvolvido um computador quântico criptograficamente relevante (CRQC), isto é, um computador quântico com desempenho suficiente para quebrar algoritmos de criptografia. No entanto, o algoritmo de Shor representa uma ameaça passiva para a segurança das comunicações da atualidade por meio do ataque *store now, decrypt later* (também conhecido como *harvest now, decrypt later*) (BEVERIDGE; BUTCHER, 2023). Este ataque, como o nome sugere, é baseado na captura de pacotes de dados transmitidos na rede atualmente para decifragem no futuro, quando um computador quântico eficiente estiver disponível e o algoritmo de Shor puder ser executado. Dessa forma, a confidencialidade das comunicações de hoje em dia poderá ser violada no futuro.

O tempo até que um computador quântico criptograficamente relevante seja desenvolvido é incerto, com estimativas que variam de 5 a 30 anos a partir de agora. Essa grande incerteza surge do fato de que o desenvolvimento da computação quântica está muito atrelado às descobertas que podem surgir nesta ciência, que, naturalmente, são imprevisíveis. A fim de avaliar o estado atual da ameaça do computador quântico à criptografia, a *Global Risk Institute* publica anualmente o *Quantum Threat Timeline Report*. No último *report*, *2023 Quantum Threat Timeline Report* (MOSCA; PIANI, 2023), 37 especialistas da área foram consultados a respeito da probabilidade de um CRQC ser desenvolvido nos próximos, e suas opiniões foram representadas no gráfico da Figura 3. Com base nesta pesquisa, estima-se, com uma visão otimista, que nos

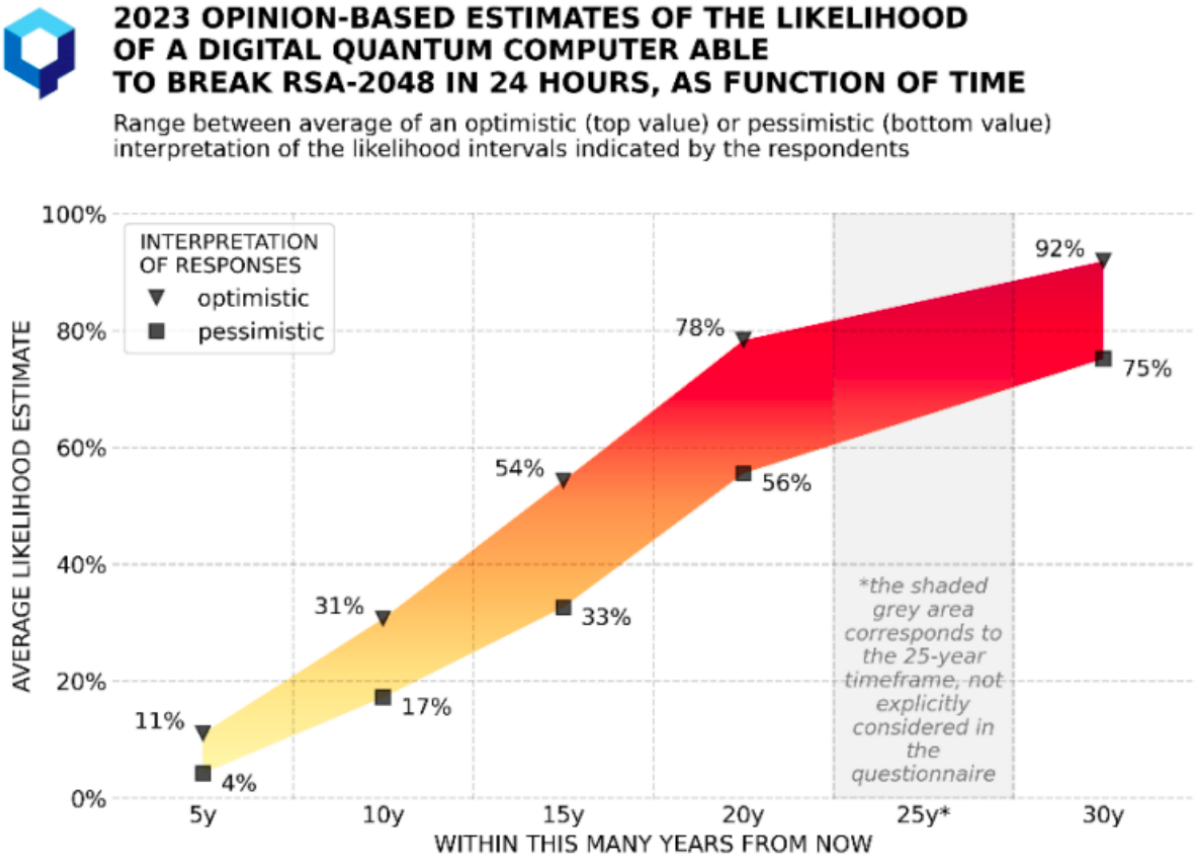


Figura 3 – Probabilidade de um computador quântico quebrar o algoritmo RSA-2048 em 24 horas nos próximos anos

Fonte: *2023 Quantum Threat Timeline Report*

próximos 15 anos há mais de 50% de chance de um CRQC ser desenvolvido. Frente à imprevisibilidade do advento do CRQC e à ameaça do ataque retroativo *store now, decrypt later*, diversos esforços surgiram a fim de desenvolver uma criptografia à prova do computador quântico, sendo o mais notável destes o processo de padronização de criptografia pós-quântica do NIST (NIST, 2024), anunciado em 2016.

2.4 CRIPTOGRAFIA PÓS-QUÂNTICA

A criptografia pós-quântica é uma criptografia desenvolvida para ser implementada em computadores clássicos com o objetivo de ser resistente contra ataques de computadores quânticos (MIT TECHNOLOGY REVIEW, s.d.). Além disto, a criptografia pós-quântica deve ser resistente a computadores clássicos também, proporcionando uma segurança igual ou superior à criptografia clássica.

Uma vez que os problemas matemáticos nos quais a criptografia clássica é baseada possuem solução eficiente em um computador quântico por meio do algoritmo de Shor (SHOR, 1994), algoritmos de criptografia pós-quântica exploram outros problemas matemáticos, que não possuem solução eficiente conhecida em um computador quântico e nem em um computador clássico. Os algoritmos pós-quânticos são classificados

de acordo com os problemas matemáticos nos quais estes se baseiam (BERNSTEIN, 2009) :

- **Lattice-based cryptography:** baseia-se em problemas difíceis relacionados a reticulados em espaços de alta dimensão, tais como encontrar o vetor mais curto em um reticulado, *Shortest Vector Problem* (SVP) (MICCIANCIO, 2005);
- **Isogeny-based cryptography:** relacionado à dificuldade de encontrar isogenias entre curvas elípticas;
- **Code-based cryptography:** baseia-se na dificuldade de decodificar mensagens que foram codificadas usando códigos de correção de erros, como o código de Goppa (BERLEKAMP, 1973);
- **Multivariate-based cryptography:** baseia-se na resolução de sistemas de equações polinomiais multivariadas;
- **Hash-based cryptography:** baseada em funções hash criptográficas.

Em 2016, foi anunciado o processo de padronização da criptografia pós-quântica do NIST (NIST, 2024), uma competição organizada pelo *National Institute of Standards and Technology* (NIST) a fim de avaliar e padronizar algoritmos de criptografia resistentes a computadores quânticos para uso em aplicações do mundo real. Este processo aceita submissões de algoritmos pós-quânticos de duas categorias: algoritmos de assinatura digital e algoritmos de KEM, utilizados para troca de chaves em protocolos criptográficos. A competição foi organizada na forma de rodadas, de modo que em cada rodada os algoritmos são submetidos a extensivos testes e criptoanálise, e somente os algoritmos suficientemente robustos avançam para a próxima rodada. Na primeira rodada[4], foram submetidos 23 algoritmos de assinatura e 59 algoritmos de KEM, dos quais apenas 3 algoritmos de assinatura e 4 de KEM chegaram à rodada final (terceira rodada[5]). Com o avanço das rodadas e estudos intensivos de criptoanálise, diversos algoritmos foram descartados do processo devido ao descobrimento de vulnerabilidades e possíveis vetores de ataques (CASTRYCK; DECRU, 2023; BEULLENS, 2022). Destes casos, destaca-se o do algoritmo de assinatura Rainbow, que teve uma vulnerabilidade descoberta na última rodada do processo (BEULLENS, 2022).

O fim da terceira rodada da competição foi marcado pelo anúncio dos algoritmos vencedores: na categoria KEM, o Kyber; na categoria de assinatura digital, CRYSTALS-Dilithium, Falcon e SPHINCS+. Contudo, o anúncio dos algoritmos vencedores não

[4] https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions

[5] https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions

marcou o fim da competição, uma vez que o NIST anunciou uma quarta rodada[6] para algoritmos de KEM, e uma nova competição[7] para algoritmos de assinatura. A quarta rodada busca padronizar novos algoritmos de KEM, dado que apenas um algoritmo de KEM foi padronizado ao fim da competição. Inicialmente, esta rodada era composta pelos algoritmos BIKE, Classic McEliece, HQC e SIKE, no entanto, o algoritmo SIKE foi desqualificado após a publicação de um ataque (CASTRYCK; DECRU, 2023).

2.4.1 Criptografia pós-quântica híbrida

Algoritmos de criptografia pós-quântica são desenvolvimentos recentes na história da criptografia, e apesar de serem submetidos à rigorosos testes e revisões no processo de padronização do NIST, ainda há margem para o descobrimento de vulnerabilidades, como foi observado ao longo das rodadas do processo de padronização, nas quais dezenas de ataques nos algoritmos candidatos foram publicados (WIKIPEDIA CONTRIBUTORS, 2024), sendo um destes na última rodada (BEULLENS, 2022). Isto é um processo natural na criptografia, dado que se espera que os algoritmos de criptografia sejam constantemente estudados e testados a fim de encontrar vetores de ataques, mesmo após décadas da sua publicação. Os algoritmos de criptografia que se mantêm seguros no decorrer dos anos provam uma robustez na sua construção, e, portanto, tornam-se confiáveis para uso até mesmo nos cenários mais críticos da segurança da informação. Algoritmos de criptografia pós-quântica ainda não atingiram a maturidade que algoritmos clássicos já possuem, tendo em vista que a sua aplicação no mundo real ainda é limitada e muito recente. Esta imaturidade provoca uma insegurança acerca da substituição dos algoritmos de criptografia clássica pelos algoritmos pós-quânticos, motivando o uso da criptografia pós-quântica híbrida.

A criptografia pós-quântica híbrida refere-se à abordagem de utilizar algoritmos criptográficos clássicos em conjunto com algoritmos pós-quânticos, visando combinar a segurança e robustez proporcionada pelos algoritmos clássicos com a resistência aos ataques quânticos oferecida pelos algoritmos pós-quânticos. O uso de criptografia pós-quântica híbrida permite uma transição gradual para a criptografia pós-quântica, diminuindo os riscos associados à transição para novos algoritmos criptográficos.

Na criptografia pós-quântica híbrida, os dois algoritmos são executados em paralelo, e as saídas produzidas por esses algoritmos são combinadas por meio de um algoritmo combinador. Este algoritmo combinador deve ser capaz de garantir a "propriedade híbrida"(STEBILA; FLUHRER; GUERON, 2023), definida da seguinte forma: o segredo resultante de um algoritmo de criptografia pós-quântica híbrida deve permanecer seguro enquanto um dos algoritmos componentes do esquema híbrido se manter seguro. Com esta propriedade, se o algoritmo pós-quântico for quebrado, o

[6] https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions

[7] https://csrc.nist.gov/Projects/pqc-dig-sig/standardization

algoritmo clássico irá garantir a segurança do esquema, uma vez que o atacante precisaria computar a saída do algoritmo clássico para obter o segredo do algoritmo híbrido, e vice-versa.

A criptografia pós-quântica híbrida pode ser utilizada em dois contextos: assinaturas digitais e troca de chaves. Um algoritmo de assinatura digital híbrido é composto por dois algoritmos de assinatura, um clássico e um pós-quântico, de tal forma que a chave (pública ou privada) corresponde à concatenação da chave do algoritmo clássico com a chave do algoritmo pós-quântico. Em um algoritmo de assinatura híbrido, os dois algoritmos de assinatura são executados com a mesma entrada (dado a ser assinado), e as assinaturas resultantes são concatenadas, obtendo dessa forma uma assinatura híbrida (CROCKETT; PAQUIN; STEBILA, 2019). Dessa forma, para que um atacante possa forjar uma assinatura híbrida, este precisa forjar a assinatura com o algoritmo clássico e com o algoritmo pós-quântico, garantindo a propriedade híbrida.

Um algoritmo de troca de chaves híbrido é composto por um algoritmo de troca de chaves clássico e um algoritmo de troca de chaves pós-quântico (STEBILA; FLUHRER; GUERON, 2023). O algoritmo pós-quântico é um KEM e o algoritmo clássico é o Diffie-Hellman, podendo este ser instanciado na forma de um KEM. A computação de um segredo compartilhado híbrido se dá por meio da execução paralela de ambos os algoritmos de troca de chaves, e consequente combinação dos segredos compartilhados obtidos de cada algoritmo. Para instanciações de algoritmos de troca de chaves híbridos no TLS, o algoritmo combinador utilizado é o *dual-PRF* (BINDEL *et al.*, 2019), um algoritmo combinador baseado no *key schedule* do TLS . Por meio deste combinador, a propriedade híbrida é garantida.

2.4.2 Adoção da criptografia pós-quântica no TLS

De modo geral, a criptografia pós-quântica foi incorporada ao TLS de duas formas: TLS pós-quântico e o KEMTLS. O TLS pós-quântico, prototipado em (CROCKETT; PAQUIN; STEBILA, 2019), busca adicionar a criptografia pós-quântica no protocolo sem alterar a estrutura e funcionamento deste, utilizando as mesmas mensagens e extensões, sendo necessária apenas a ressignificação de algumas extensões a fim de adequá-las aos algoritmos pós-quânticos. No TLS pós-quântico, a troca de chaves de Diffie-Hellman é substituída pela troca de chaves KEM, e a autenticação com assinaturas digitais é adaptada para suportar algoritmos de assinatura pós-quânticos.

O KEMTLS, proposto em (SCHWABE; STEBILA; WIGGERS, 2020), propõe uma versão alternativa do protocolo de *handshake* do TLS otimizada para a criptografia pós-quântica, na qual novas mensagens e extensões são adicionadas ao protocolo. A proposta do KEMTLS é inspirada no fato de que algoritmos de assinatura pós-quânticos, de modo geral, possuem desempenho inferior e chaves maiores em comparação a determinados algoritmos de KEM pós-quânticos. Em vista disso, o KEMTLS propõe o

uso de KEMs no processo de troca de chaves e no processo de autenticação.

2.5 TLS PÓS-QUÂNTICO

O TLS pós-quântico (*Post-quantum TLS* - PQTLS) (CROCKETT; PAQUIN; STEBILA, 2019) é um protótipo experimental do protocolo TLS com suporte à criptografia pós-quântica. Neste protótipo, a principal alteração encontra-se no protocolo de troca de chaves utilizado para estabelecer o segredo compartilhado entre as partes do protocolo. No TLS 1.3, o protocolo de troca de chaves utilizado é o Diffie-Hellman sobre corpos finitos ou curvas elípticas (RESCORLA, 2018), pelo qual, a partir do compartilhamento de parâmetros públicos do algoritmo, é possível estabelecer um segredo compartilhado de forma segura em uma rede insegura. Por conta do protocolo Diffie-Hellman ser vulnerável ao algoritmo de Shor, no TLS pós-quântico são utilizados algoritmos de KEM pós-quânticos no protocolo de troca de chaves. Para a autenticação, não foram necessárias grandes alterações na estrutura do protocolo, sendo suficiente o uso de algoritmos de assinatura pós-quântico em certificados digitais e assinaturas digitais.

Na Figura 4, encontra-se o *handshake* do TLS pós-quântico. Tal como o *handshake* do TLS 1.3, o protocolo inicia a partir da mensagem *ClientHello*, enviada pelo cliente ao servidor, juntamente com suas extensões. Como visto na seção 2.1, o TLS 1.3 faz uso das extensões `supported_groups`, `signature_algorithms` e `signature_algorithms_cert` da mensagem *ClientHello* para negociar, respectivamente, os grupos de curvas elípticas ou corpos finitos do Diffie-Hellman, os algoritmos de assinatura suportados no *handshake* e os algoritmos de assinatura suportados em cadeias de certificados digitais. Para o TLS pós-quântico não é diferente, as extensões `signature_algorithms` e `signature_algorithms_cert` serão usadas da mesma forma, no entanto, a semântica da extensão `supported_groups` foi alterada: além de poder conter os grupos de Diffie-Hellman, esta extensão irá conter os algoritmos de KEM pós-quânticos a serem negociados no *handshake*. Dado que o TLS pós-quântico encontra-se em fase de protótipo e não possui uma especificação formal ainda, esta abordagem é utilizada, apesar de não ser ideal e não seguir a semântica da extensão.

A fim de iniciar o protocolo de troca de chaves, durante a criação da mensagem *ClientHello*, o cliente realiza a geração do par de chaves de KEM efêmero, pk_e e sk_e. A chave pública obtida neste processo, pk_e é anexada ao *ClientHello* por meio da extensão `key_share`. A mensagem é então enviada ao servidor, que ao recebimento desta dá prosseguimento ao protocolo de troca de chaves KEM.

Com a chave pública do cliente em mãos, o servidor executa a função de encapsulamento `KEM.Encaps()`, obtendo como resultado o segredo compartilhado ss_e e o texto cifrado ct_e, que é anexado à mensagem *ServerHello* por meio da extensão `key_share`, e transmitido ao cliente. Novamente, a semântica da extensão não é seguida, no entanto,

por se tratar de um protótipo, esta reinterpretação da extensão é aceita.

O cliente, quando recebe o *ServerHello*, realiza o desencapsulamento do texto cifrado a partir de sua chave privada de KEM, obtendo o segredo compartilhado gerado pelo servidor e consequentemente finalizando o protocolo de troca de chaves do TLS pós-quântico. A partir deste momento, o protocolo segue o TLS 1.3: extensões adicionais são negociadas em *EncryptedExtensions*, o servidor fornece seu certificado digital X.509 pós-quântico em *Certificate*, autentica-se por meio de uma assinatura pós-quântica em *CertificateVerify* e gera a mensagem *Finished*. Deste modo, o TLS pós-quântico é definido como o TLS 1.3 que utiliza KEMs pós-quânticos no protocolo de troca de chaves e algoritmos de assinatura digital pós-quânticos para autenticação.

O *key schedule* do TLS pós-quântico é o mesmo do TLS 1.3, com a única diferença sendo o *input keying material* do segredo *Handshake Secret*, que para o caso do TLS pós-quântico é o segredo compartilhado obtido do KEM pós-quântico, e não mais o segredo compartilhado obtido em uma troca de chaves Diffie-Hellman.

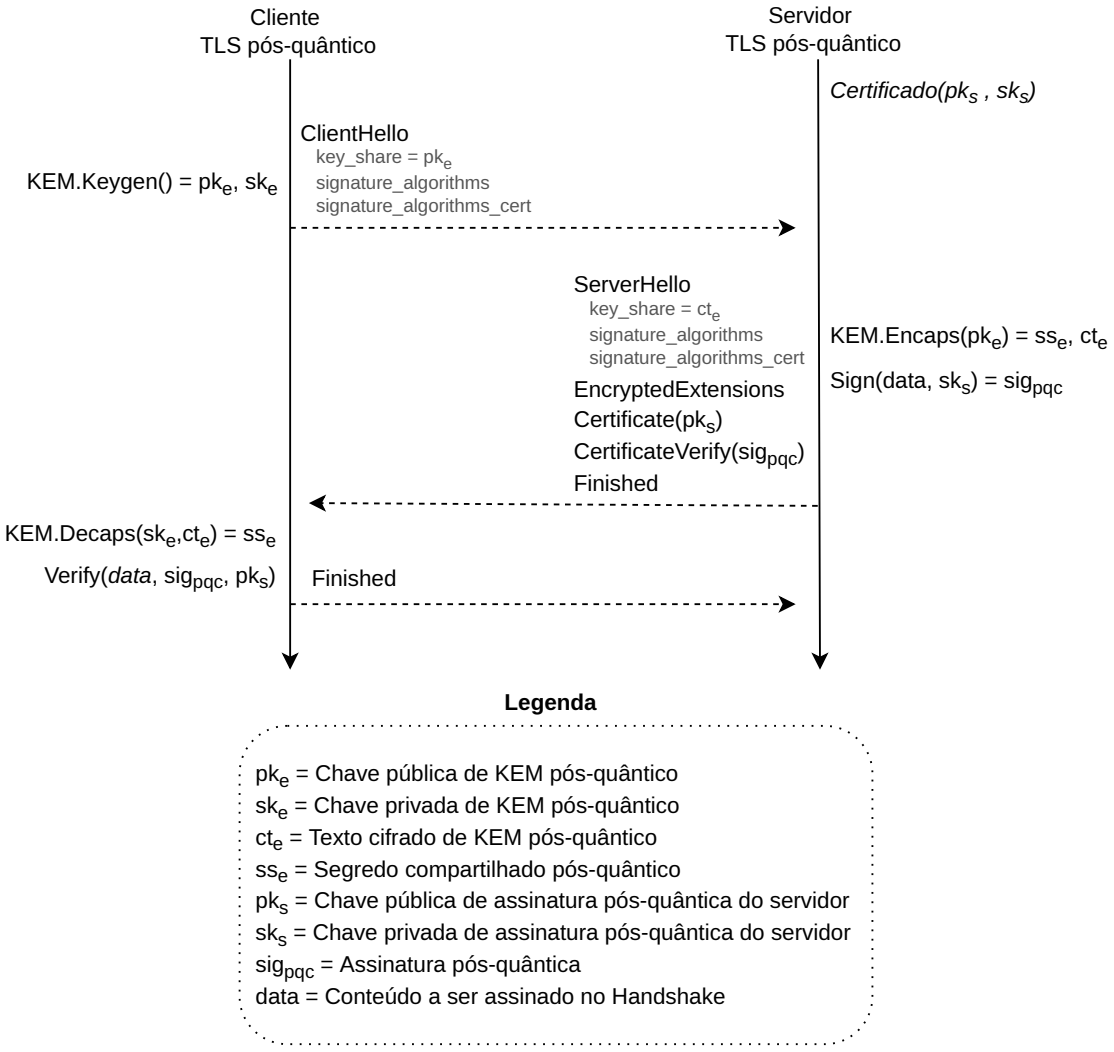


Figura 4 – Protocolo de *handshake* do TLS pós-quântico (apenas autenticação do servidor)

2.6 TLS PÓS-QUÂNTICO HÍBRIDO

O TLS pós-quântico híbrido (*Hybrid* PQTLS), inicialmente proposto em (CROCKETT; PAQUIN; STEBILA, 2019) é uma extensão do TLS pós-quântico que utiliza algoritmos clássicos em conjunto com algoritmos pós-quânticos no protocolo de troca de chaves e na autenticação das partes comunicantes.

A troca de chaves para o TLS pós-quântico híbrido foi definida no *internet-draft Hybrid key exchange in TLS 1.3* (STEBILA; FLUHRER; GUERON, 2023). Neste documento, são propostas modificações no TLS 1.3 a fim de fornecer suporte à criptografia pós-quântica híbrida, tais como: negociação de algoritmos híbridos, transmissão de chaves públicas híbridas, derivação de segredos compartilhados híbridos, entre outros.

A negociação dos algoritmos híbridos é feita utilizando os mecanismos de negociação do protocolo, isto é, as extensões `supported_groups`, `signature_algorithms`, `signature_algorithms_cert`. Por mais que algoritmos híbridos sejam a composição de dois algoritmos (ou mais, em alguns casos), estes são representados como um único algoritmo nas extensões em questão, para que não seja necessário desenvolver extensões adicionais para a negociação de algoritmos híbridos.

Na troca de chaves do TLS pós-quântico híbrido, o algoritmo de KEM é executado em conjunto com o algoritmo de Diffie-Hellman. O *handshake* (Figura 5) inicia com a geração do par de chaves de Diffie-Hellman e a geração do par de chaves KEM. As chaves públicas resultantes deste processo, são concatenadas e transmitidas ao servidor por meio da extensão `key_share`. O servidor, ao processar a mensagem *ClientHello*, desconcatena as chaves públicas e executa o algoritmo de troca de chaves correspondente: para a chave pública de Diffie-Hellman, é executado o algoritmo de Diffie-Hellman `(EC)DHE.DH()`, obtendo o segredo compartilhado clássico ss_{dhe}; para a chave pública de KEM, o algoritmo de encapsulamento `KEM.Encaps()`, obtendo o segredo compartilhado pós-quântico ss_e.

A fim de permitir que o cliente execute a troca de chaves também, o servidor transmite a sua chave pública de Diffie-Hellman spk_{dhe} concatenada ao texto cifrado ct_e por meio da extensão `key_share`. O cliente, por sua vez, executa o algoritmo de Diffie-Hellman `(EC)DHE.DH()` sobre a chave spk_{dhe} e o algoritmo de desencapsulamento `KEM.Decaps()` sobre o texto cifrado ct_e, computando os mesmos segredos compartilhados que o servidor.

A criptografia pós-quântica híbrida requer que ambos os algoritmos componentes participem do processo de geração do segredo e que a combinação das saídas dos algoritmos componentes do esquema híbrido garanta a propriedade híbrida, definida na subseção 2.4.1. De modo a atender este requisito, os segredos compartilhados ss_{dhe} e ss_e são combinados por meio da concatenação e incorporados no *key schedule* como IKM da função HKDF-Extract do segredo *Handshake Secret*, conforme a Figura 6. Esta construção corresponde ao combinador *dual-PRF* (BINDEL *et al.*, 2019), que

é comprovadamente seguro e garante a propriedade híbrida.

Para a autenticação das partes do protocolo, são utilizados certificados digitais X.509 híbridos, isto é, certificados digitais assinados por um algoritmo pós-quântico híbrido de assinatura, e que contêm uma chave pública pós-quântica híbrida de assinatura. O processo de autenticação segue como no TLS 1.3: o certificado é transmitido na mensagem *Certificate*, e a assinatura híbrida realizada por este certificado sobre o *handshake transcript* é transmitida por meio da mensagem *CertificateVerify*.

Por fim, são trocadas mensagens *Finished* entre o cliente e o servidor, finalizando o protocolo de *handshake*.

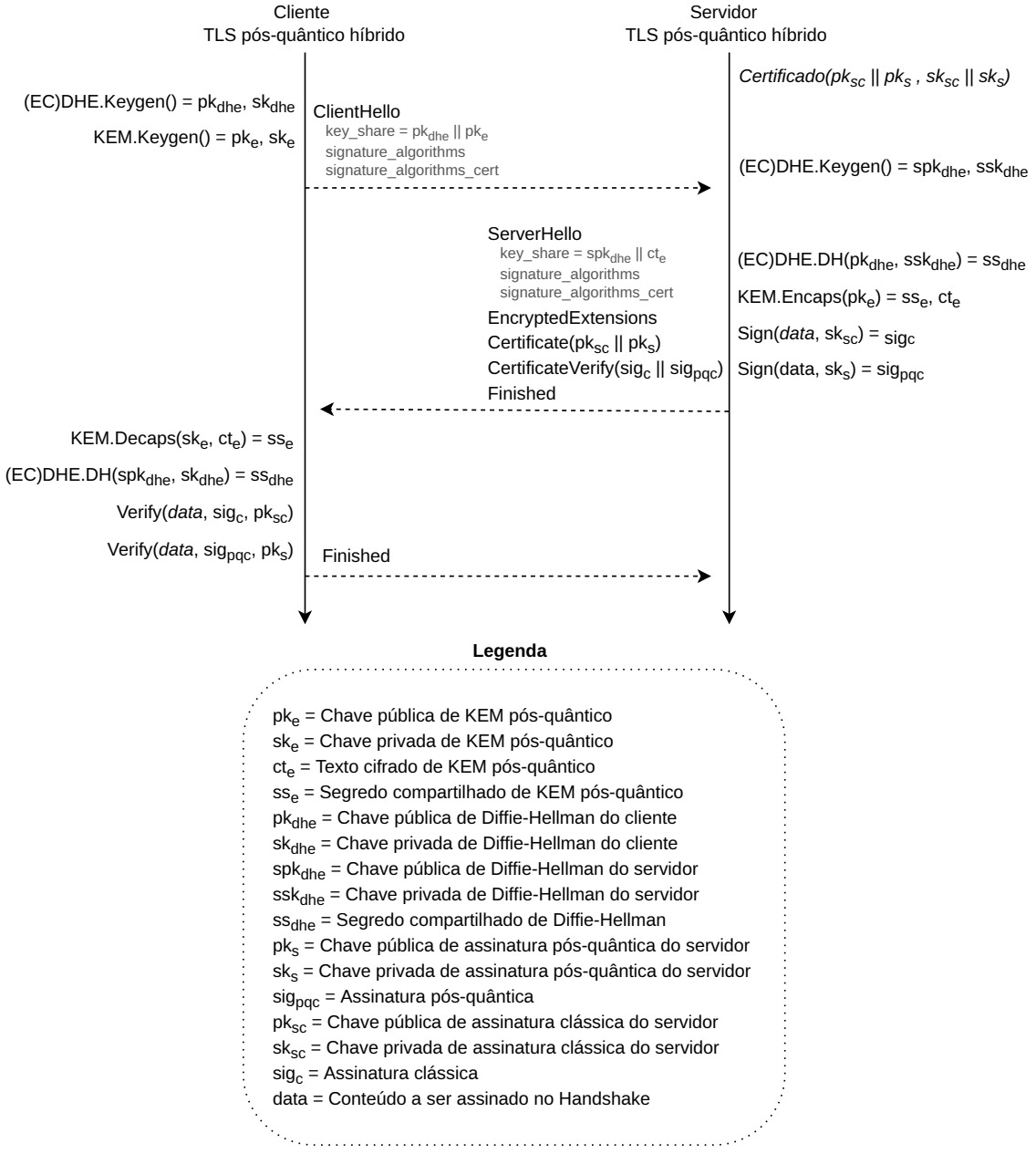


Figura 5 – Protocolo de *handshake* do TLS pós-quântico híbrido (apenas autenticação do servidor)

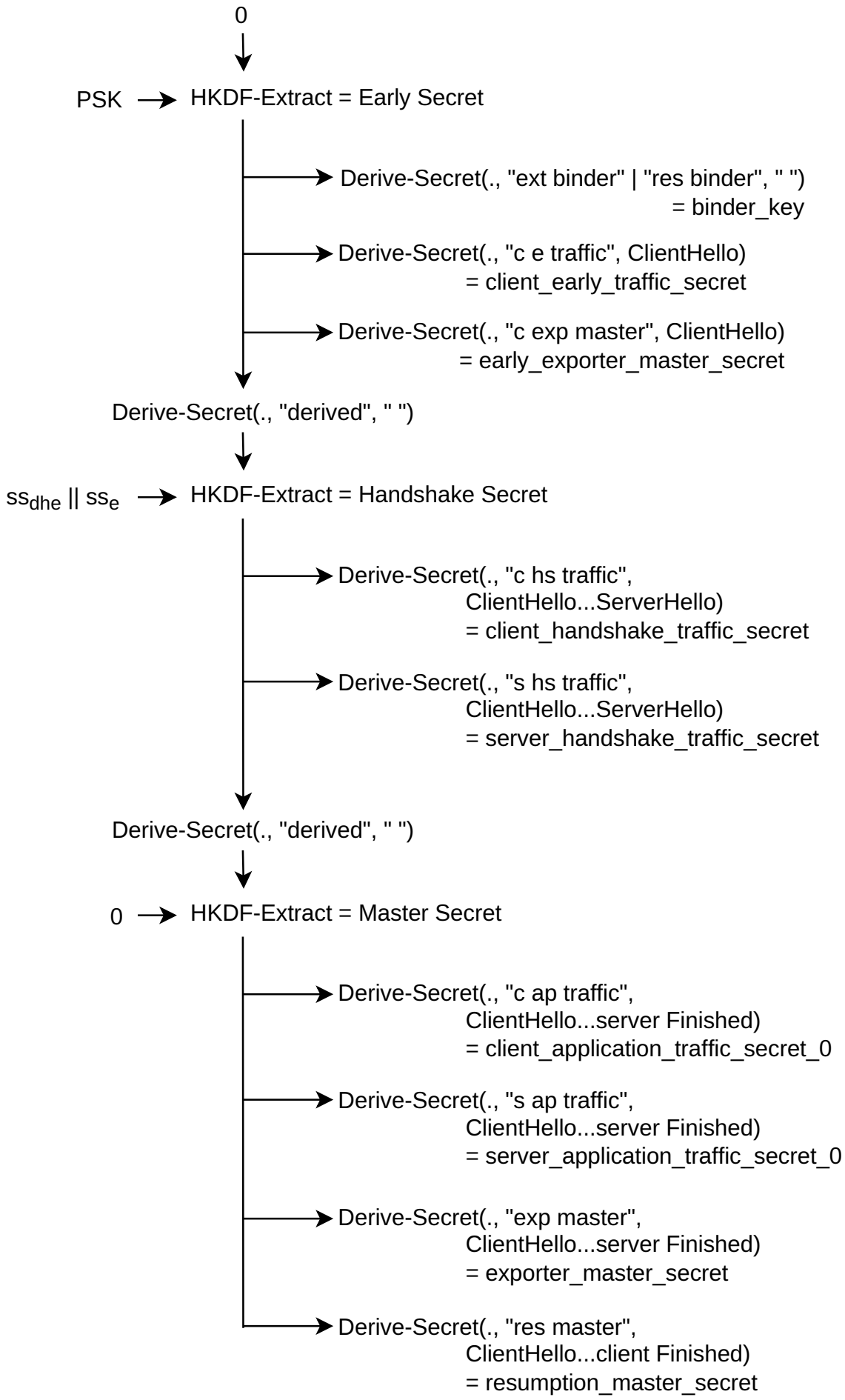


Figura 6 – *Key schedule* do TLS pós-quântico híbrido

2.7 KEMTLS

O KEMTLS surge como um protocolo alternativo ao protocolo de *handshake* do TLS 1.3, otimizado para o uso de criptografia pós-quântica (SCHWABE; STEBILA; WIGGERS, 2020). O KEMTLS faz o uso de *Key Encapsulation Mechanism* (KEMs) no protocolo de troca de chaves e no processo de autenticação do servidor (e opcionalmente o cliente), sendo este último o grande diferencial do protocolo, uma vez que no TLS e na grande maioria dos protocolos criptográficos faz-se o uso de assinaturas digitais nos fluxos de autenticação.

O KEMTLS foi modelado a partir do protocolo de handshake do TLS 1.3, portanto componentes deste protocolo, como as mensagens, extensões, segredos e chaves, são reutilizados no KEMTLS com as devidas adaptações. O KEMTLS é baseado no uso de dois pares de chave KEM: um par de chave KEM efêmero utilizado no protocolo de troca de chaves; e um par de chave KEM estático utilizado para autenticação (opcionalmente, o cliente pode ter seu par de chave KEM estático para autenticar-se perante o servidor).

O protocolo de *handshake*, representado na Figura 7, inicia com o cliente gerando um par de chaves KEM efêmero. Tal como no TLS 1.3, a primeira mensagem enviada no protocolo é o *ClientHello*, que contém na extensão `supported_groups` os algoritmos de KEM suportados pelo cliente para a troca de chaves, e na extensão `key_share` a chave pública de KEM efêmero gerada pelo cliente. Ao recebimento desta mensagem, o servidor inicia o protocolo de troca de chaves com KEMs por meio da execução da função de encapsulamento, obtendo o segredo compartilhado efêmero e o texto cifrado correspondente.

O texto cifrado gerado a partir do encapsulamento com KEM efêmero é enviado ao cliente por meio da mensagem *ServerHello*, na extensão `key_share`. O cliente, ao receber esta mensagem, realiza o desencapsulamento do texto cifrado com a chave privada de KEM efêmero, obtendo desta forma o segredo compartilhado efêmero. Este, por sua vez, é então incorporado ao *key schedule* do KEMTLS.

Em seguida, são enviadas as mensagens *EncryptedExtensions* e, opcionalmente, *CertificateRequest*, que desempenham a mesma função que no TLS 1.3. Após estas mensagens, o protocolo inicia a fase de autenticação com o envio da mensagem *Certificate* por parte do servidor. A diferença desta mensagem, em comparação com a do TLS 1.3, é que o certificado X.509 presente nesta contém a chave pública de KEM estático do servidor, e não a chave pública de assinatura do servidor, visto que no KEMTLS a autenticação é feita por meio de KEMs.

Por mais que o KEMTLS dispense o uso de algoritmos de assinatura no *handshake*, estes ainda são necessários na cadeia de certificação do certificado do servidor, isto é, nos certificados da autoridade certificadora emissora e das autoridades certificadoras intermediárias. Portanto, ainda serão realizadas operações de algoritmos

de assinatura no protocolo, mas estas serão restritas a operações de verificação de assinaturas da cadeia de certificação apenas. Isto não afeta profundamente o desempenho do protocolo, pois a operação de verificação não possui um custo computacional elevado, tal como a operação de assinatura.

Após o envio da mensagem *Certificate*, inicia-se outro protocolo KEM a fim de autenticar o servidor. Quando o cliente recebe a mensagem *Certificate*, este executa a função de encapsulamento com a chave pública estática presente no certificado, obtendo como resultado o segredo compartilhado estático e seu texto cifrado correspondente. O texto cifrado obtido desta operação é então enviado ao servidor por meio da mensagem *ClientKEMCiphertext*, que é uma mensagem criada especificamente para o protocolo KEMTLS. Juntamente a esta mensagem, o cliente envia o *Finished*, que corresponde à mesma mensagem do TLS 1.3, e provê a autenticação do *handshake*.

O servidor, ao receber a mensagem *ClientKEMCiphertext*, desencapsula o texto cifrado recebido, obtendo o segredo compartilhado estático, que é então incorporado ao *key schedule* do KEMTLS. A fim de finalizar o protocolo, o servidor responde ao cliente com a mensagem *Finished*, que, novamente, desempenha o mesmo papel que no TLS 1.3 e finaliza o *handshake*.

Diferentemente do TLS, que finaliza o handshake em uma única rodada (*round trip*), o KEMTLS necessita de duas rodadas para finalizar o *handshake*.

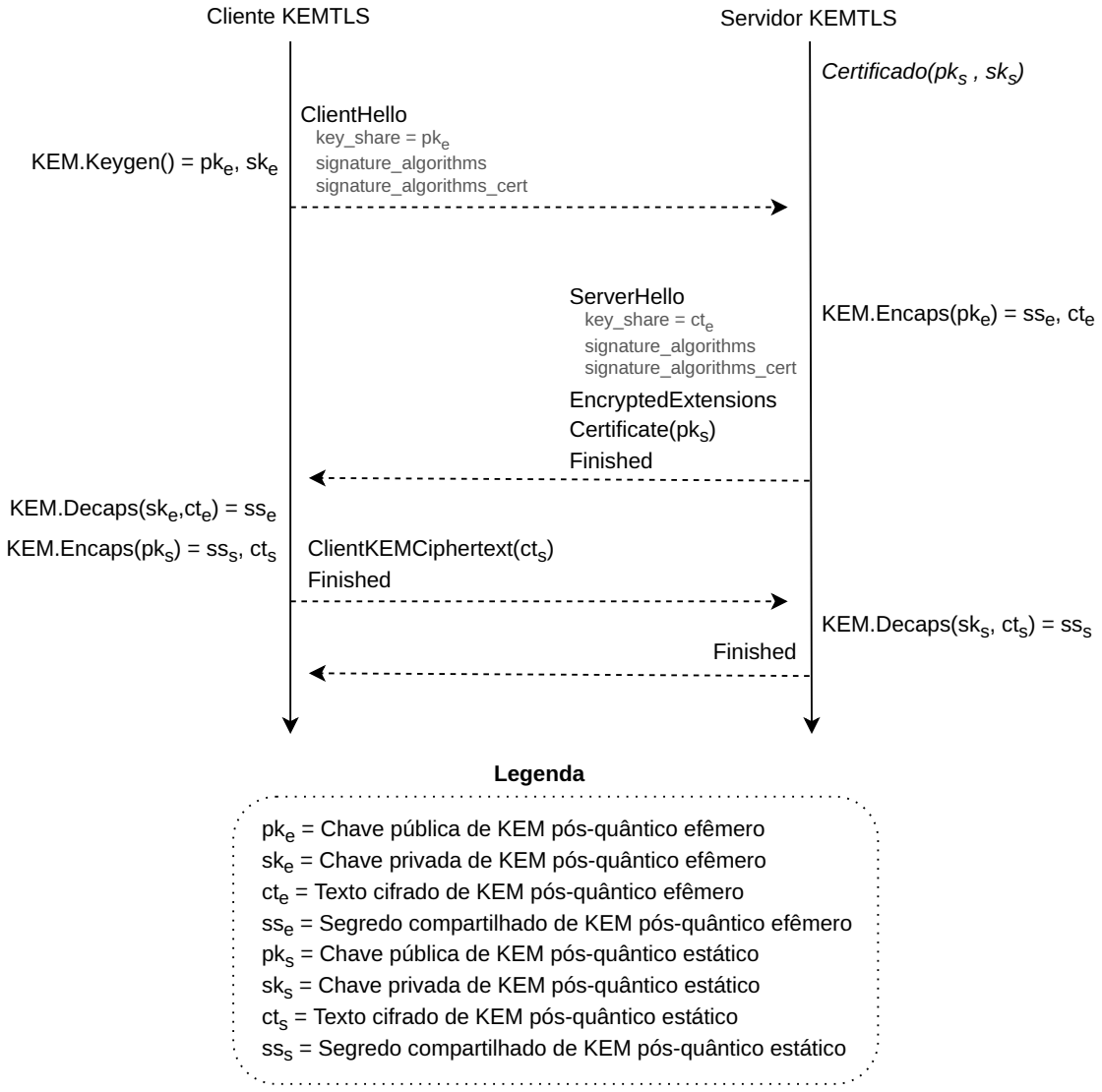


Figura 7 – Protocolo de *handshake* do KEMTLS (apenas autenticação do servidor)

2.7.1 *Key schedule*

O *key schedule* do KEMTLS, representado na Figura 8, é baseado no *key schedule* do TLS 1.3, possuindo algumas adaptações e segredos adicionais no fluxo de derivação. No KEMTLS, o segredo usado como *input keying material* da função HKDF-Extract que deriva o *Handshake Secret* será o segredo compartilhado efêmero, gerado no protocolo de troca de chaves KEM efêmero. Por conta de serem feitas duas trocas de chave KEM no KEMTLS, foi criado o segredo *Authenticated Handshake Secret*. Este segredo recebe como *input keying material* o segredo compartilhado estático e é responsável por incorporar o segredo compartilhado estático nos processos de derivação de chaves de criptografia de dados da aplicação.

Desta forma, tanto a troca de chaves KEM efêmera quanto a troca de chaves KEM estática contribuem na geração das chaves do protocolo, e apenas participantes autênticos (isto é, que possuem o segredo compartilhado estático) podem computar

estas chaves. Esta é uma grande diferença em relação ao TLS 1.3, dado que neste a chave privada do servidor não participa do processo de geração das chaves de criptografia, pois a autenticação é feita no nível do protocolo pela troca de mensagens (*Certificate* e *CertificateVerify*), enquanto que no KEMTLS ela é feita no nível criptográfico do protocolo.

Mecanismos de retomada de conexão (*resumption mechanisms*) como o 0-RTT do TLS podem ser usados no KEMTLS. Contudo, este cenário não é abordado na proposta do KEMTLS, portanto, o Early Secret é computado a partir de duas entradas nulas.

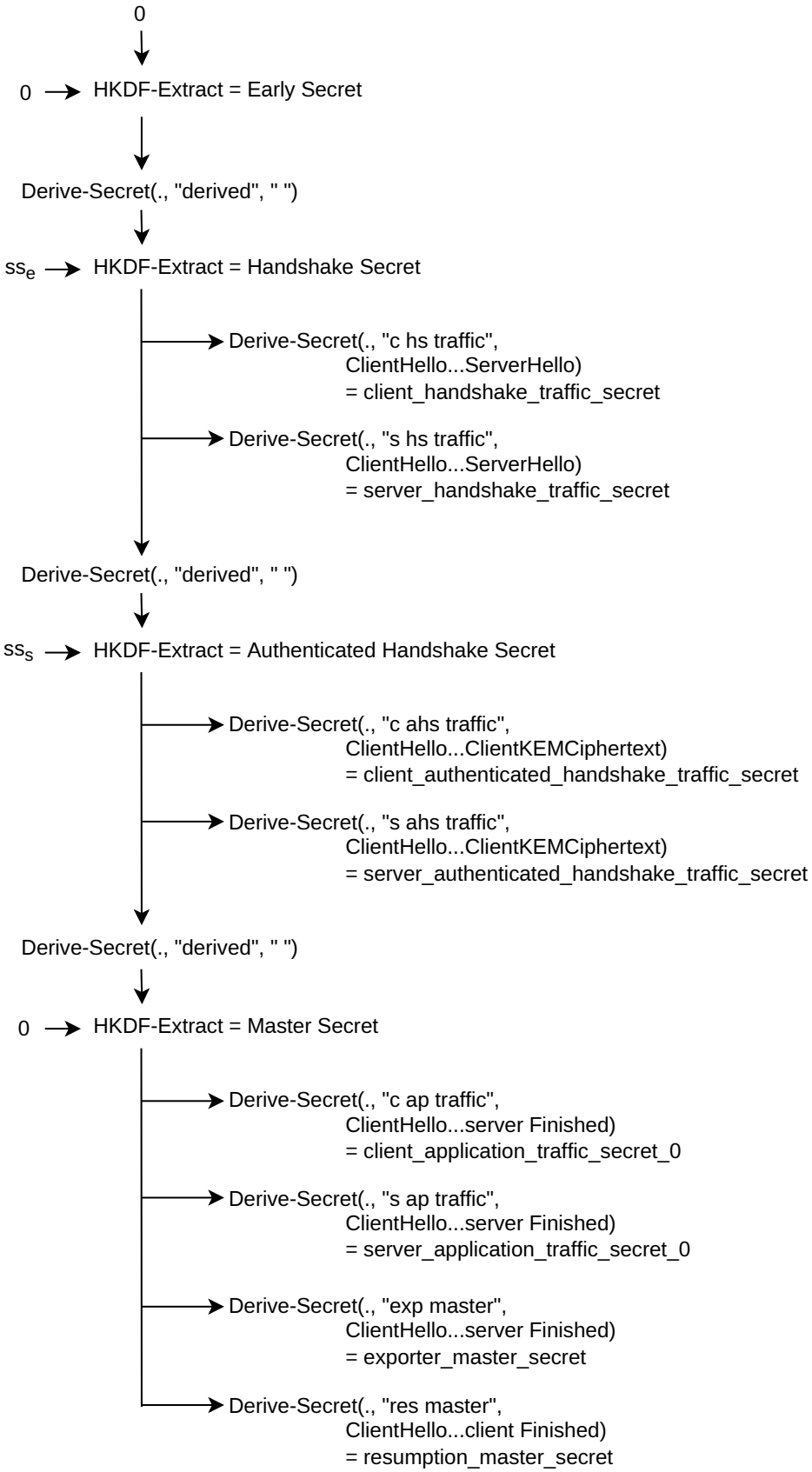


Figura 8 – *Key schedule* do KEMTLS

2.7.1.1 Autenticação Implícita e Autenticação Explícita

Por conta da troca de chaves KEM estática, que é responsável pela autenticação, participar do processo de geração de chaves do protocolo, o servidor pode se encontrar em dois estados de autenticação durante o *handshake*: autenticação implícita ou autenticação explícita.

No instante em que o cliente incorpora o segredo compartilhado estático no *key schedule* e computa as chaves derivadas do segredo *Authenticated Handshake Secret*, o servidor encontra-se implicitamente autenticado. Isto acontece pois, uma vez que as próximas mensagens serão criptografadas por chaves derivadas do segredo compartilhado estático, o servidor só irá conseguir descriptografá-las se conseguir computar o segredo compartilhado estático, o que exige que o servidor esteja em posse da chave privada estática. Desta forma, se o servidor não for uma entidade autêntica, ele não conseguirá descriptografar as mensagens e o *handshake* irá ser interrompido.

Quando o cliente recebe a mensagem *Finished* do servidor, o servidor encontra-se explicitamente autenticado, pois o HMAC contido nesta mensagem prova ao cliente que o servidor conseguiu computar corretamente o segredo compartilhado estático, e, portanto, é um servidor autêntico.

Desta forma, no KEMTLS, o servidor encontra-se autenticado implicitamente em uma rodada do *handshake* e explicitamente em duas rodadas do *handshake*.

2.8 KEMTLS-PDK

Como visto na seção 2.1.3, por meio da extensão *Cached Information* do TLS 1.3, é possível omitir a transmissão de certificados e suas respectivas cadeias de certificação no *handshake* quando o cliente já possui conhecimento dessas informações. Atualmente, esta extensão não é amplamente implementada, visto que os certificados que temos hoje em dia possuem poucos bytes, portanto, a transmissão destes durante o *handshake* não afeta significativamente o desempenho do *handshake*. Mas com o advento da criptografia pós-quântica, este cenário muda, pois chaves de algoritmos pós-quânticos possuem um tamanho grande quando comparadas às chaves de algoritmos clássicos.

A fim de fornecer suporte a estes casos de uso, foi desenvolvido o KEMTLS *with pre-distributed public keys* (SCHWABE; STEBILA; WIGGERS, 2021) (KEMTLS-PDK), que similarmente à extensão *Cached Information*, informa ao servidor que o cliente já está em posse do certificado e que não é necessário o seu envio no *handshake*. No entanto, os benefícios do KEMTLS-PDK não se restringem só a isto. Com o KEMTLS-PDK, o servidor encontra-se explicitamente autenticado perante o cliente em uma rodada do protocolo, tal como ocorre no TLS 1.3, permitindo que o servidor já possa

transmitir dados de aplicação ao cliente nesta primeira rodada. Desta forma, o protocolo KEMTLS é finalizado em uma rodada, e não duas rodadas como ocorria no KEMTLS padrão.

No *handshake* do KEMTLS-PDK (Figura 9), o cliente já possui conhecimento prévio da chave pública estática de KEM do servidor. A forma como ele obteve essa chave está fora do escopo do protocolo, ou seja, não há uma regra específica para isso, podendo o cliente ter obtido por meio de cache do navegador web, certificados de servidor pré-instalados no dispositivo IoT, certificados salvos em banco de dados de aplicações *mobiles*, entre outros. Com a posse desta chave, o cliente executa a função de encapsulamento e obtém o segredo compartilhado estático, que é incorporado ao *key schedule*, e o seu respectivo texto cifrado. Paralelamente a isto, o cliente gera um par de chaves KEM a fim de realizar a troca de chaves de KEM efêmero, que garante a propriedade de *forward secrecy* do protocolo.

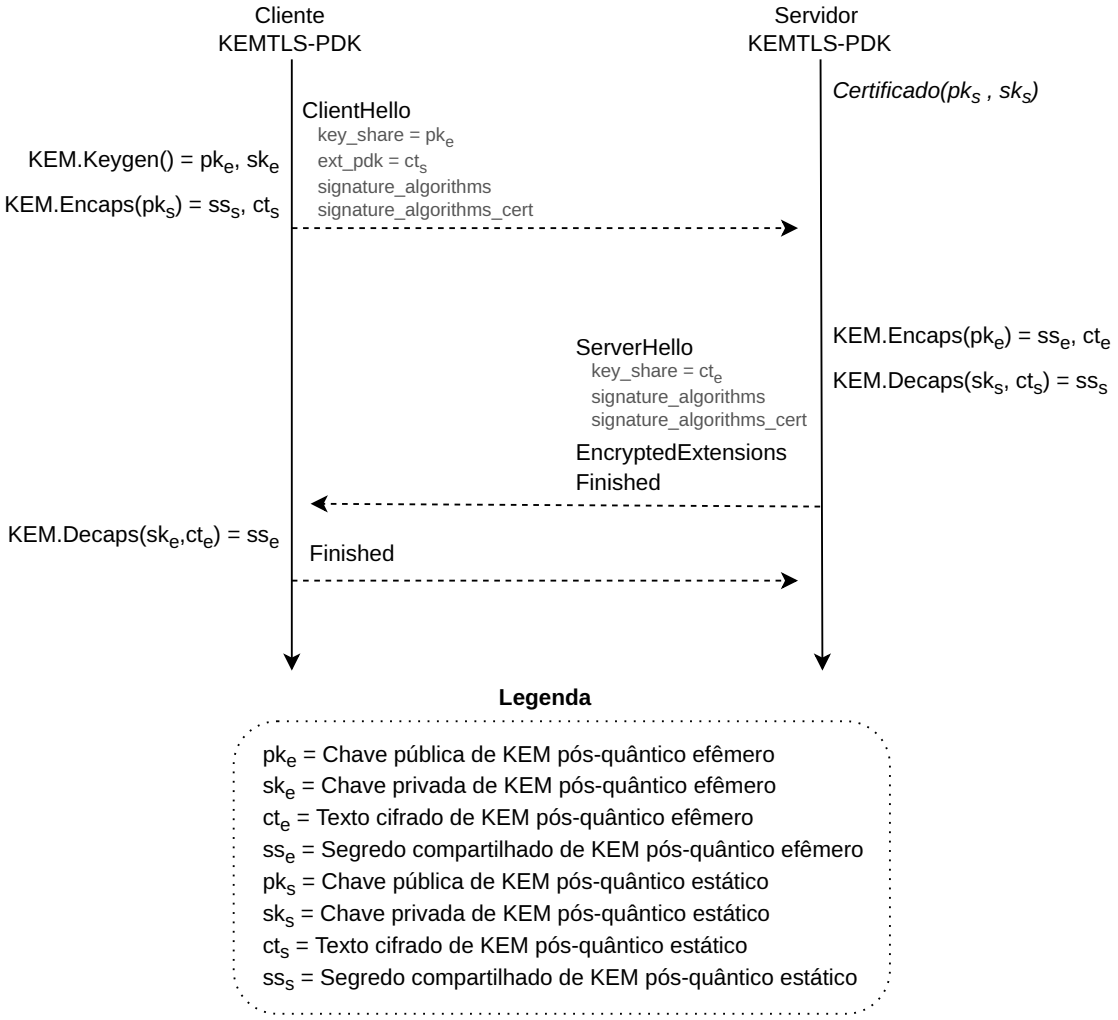


Figura 9 – Protocolo de *handshake* do KEMTLS-PDK (apenas autenticação do servidor)

Com estas operações criptográficas realizadas, o cliente envia a mensagem *ClientHello* ao servidor, contendo a chave pública de KEM efêmero na extensão `key_share` e o texto cifrado do encapsulamento na extensão `ext_pdk`. O servidor, quando recebe

esta mensagem, gera e encapsula o segredo compartilhado efêmero com a chave pública efêmera de KEM do cliente e simultaneamente desencapsula o texto cifrado com a sua chave privada de KEM estático, obtendo o segredo compartilhado estático. Feito isto, ambos os segredos são incorporados ao *key schedule* do servidor, derivando principalmente dois tipos de chaves: a chave de criptografia do *handshake*, que irá criptografar as mensagens subsequentes do protocolo; e a chave de criptografia de dados da aplicação, que irá criptografar o *payload* de dados transmitidos na conexão. Em seguida, o servidor envia as mensagens *EncryptedExtensions* e *Finished*. A partir desse momento, o servidor já pode enviar dados da aplicação ao cliente, uma vez que a chave de criptografia para isto já foi computada.

O cliente, ao receber as mensagens do servidor, desencapsula o texto cifrado enviado no *ServerHello*, tendo como resultado o segredo compartilhado efêmero. Este segredo é então incorporado ao *key schedule* e as chaves de criptografia de tráfego do protocolo são derivadas. Por fim, o cliente produz a sua mensagem *Finished* e envia ao servidor, finalizando o protocolo em uma única rodada.

O protocolo KEMTLS-PDK é de fundamental importância no cenário pós-quântico, pois ele altera a viabilidade de implantação de determinados algoritmos pós-quânticos no mundo real. Com ele, algoritmos pós-quânticos que possuem chaves públicas muito grandes podem ser utilizados no protocolo sem afetar profundamente o desempenho deste, tal como o algoritmo de KEM pós-quântico Classic McEliece. Este algoritmo possui chaves públicas com centenas de *kilobytes* de tamanho, excedendo dessa forma o tamanho limite das mensagens do TLS 1.3. Por conta disso, este algoritmo só possui implantação viável no KEMTLS-PDK, uma vez que este não requer a transmissão da chave pública.

2.8.1 Key schedule

O key schedule do KEMTLS-PDK (Figura 10) diferencia-se do key schedule do KEMTLS, uma vez que o primeiro não deriva o segredo *Authenticated Handshake Secret*. Como a chave pública de KEM estático encontra-se disponível para o cliente desde o início do protocolo, o segredo compartilhado estático é computado antes da primeira rodada e incorporado prontamente ao *key schedule* como IKM do *Early Secret*, o qual pode ser derivado em chaves de criptografia utilizadas para criptografar dados de 0-RTT.

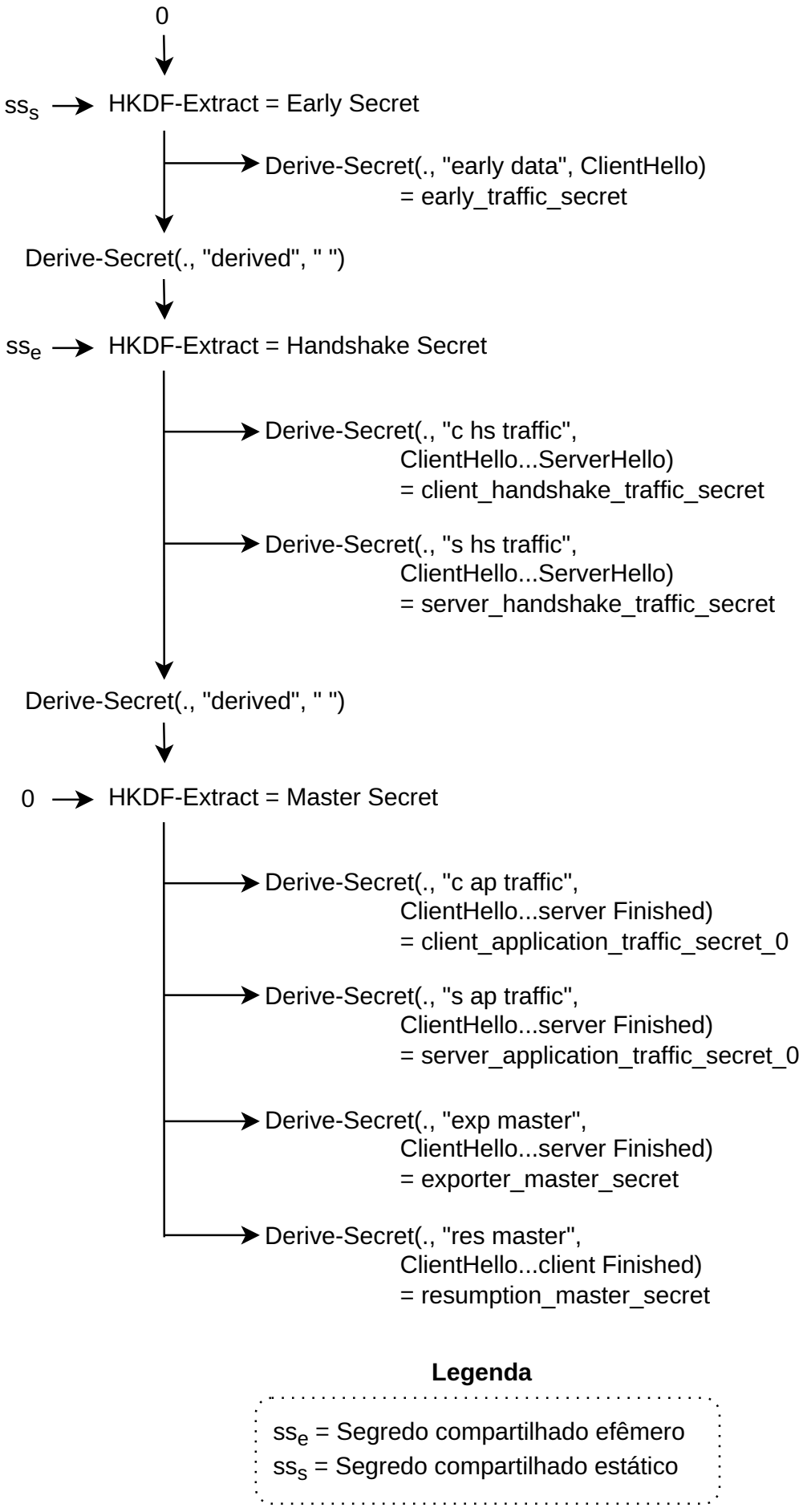


Figura 10 – *Key schedule* do KEMTLS-PDK

3 PROPOSTA

Apesar de o processo de padronização da criptografia pós-quântica do NIST ter finalizado em 2022, uma nova rodada do processo foi anunciada com o intuito de padronizar novos algoritmos de KEM.

Os algoritmos padronizados até o momento são, em sua maioria, baseados em problemas matemáticos de reticulados. Para esta nova rodada, portanto, buscou-se padronizar algoritmos de KEM baseados em problemas matemáticos diferentes, com o objetivo de ter uma maior diversidade de algoritmos pós-quânticos padronizados. Isto é importante pois, pelo fato de a criptografia pós-quântica ser uma criptografia emergente, novas descobertas nessa área podem expor vulnerabilidades destes algoritmos e abrir portas para ataques, como foi observado durante o processo de padronização (BEULLENS, 2022; CASTRYCK; DECRU, 2023). Os algoritmos candidatos à padronização são: BIKE, HQC, SIKE, e Classic McEliece. BIKE, HQC e Classic McEliece são KEMs baseados em problemas de códigos corretores de erro (*code-based cryptography*), e SIKE é um KEM baseado em isogenias (*isogeny-based cryptography*). Destes quatro, apenas três se mantêm no processo de padronização, uma vez que SIKE foi desqualificado após a publicação de um ataque a este (CASTRYCK; DECRU, 2023).

Por serem baseados em problemas matemáticos diferentes, os algoritmos candidatos possuem um desempenho bastante diferente em comparação com o KEM padronizado, Kyber, tornando necessária a avaliação da viabilidade de implantação e desempenho destes novos algoritmos em protocolos criptográficos.

No trabalho anterior (GIRON *et al.*, 2023), publicado em 2023, foi desenvolvido o KEMTLS híbrido. Neste trabalho, o protocolo em questão foi instanciado com os algoritmos finalistas de KEM do processo de padronização, e teve seu desempenho avaliado junto ao protocolo TLS pós-quântico híbrido. Este trabalho de conclusão de curso se propõe a expandir o trabalho anterior, instanciando os algoritmos candidatos à padronização no KEMTLS híbrido, a fim de avaliar seu desempenho em um ambiente realista. Além disso, busca-se comparar a latência do KEMTLS híbrido com a do TLS pós-quântico híbrido, identificando dessa forma o protocolo com melhor desempenhos nas conexões.

3.1 KEMTLS HÍBRIDO

O KEMTLS híbrido (Hybrid KEMTLS), desenvolvido em trabalho anterior (GIRON *et al.*, 2023), é uma extensão do protocolo KEMTLS que adiciona suporte à criptografia pós-quântica híbrida por meio da incorporação de algoritmos clássicos de KEM nas operações criptográficas do protocolo.

Na incorporação de KEM clássicos ao protocolo, são utilizados esquemas de *Hy-*

brid Public Key Encryption (HPKE) (BARNES *et al.*, 2022) para instanciar o algoritmo de Diffie-Hellman na forma de um KEM. Nestes esquemas, o algoritmo de Diffie-Hellman é combinado a uma função de derivação de chave HKDF. Desta forma, um KEM clássico de Diffie-Hellman é definido, informalmente, da seguinte forma:

Código 2 – Pseudocódigo de um KEM baseado em Diffie-Hellman

```
function Keygen() {
  pkR, skR = DH.GenerateKeyPair()
  return pkR, skR
}

function Encaps(pkR) {
  // pkR é a chave pública do destinatário

  // Gera par de chaves Diffie-Hellman
  skE, pkE = DH.GenerateKeyPair()

  // Executa o algoritmo de Diffie-Hellman
  dh = DH.Diffie-Hellman(skE, pkR)

  kem_context = concat(pkE, pkR)

  // Deriva o segredo gerado pelo Diffie-Hellman no segredo
     compartilhado do KEM
  // por meio do mecanismo "extract-then-expand" do HKDF
  shared_secret = HKDF.ExtractAndExpand(dh, kem_context)

  // Retorna o segredo compartilhado e a chave pública de
     Diffie-Hellman
  return shared_secret, pkE
}

function Decaps(pkE, skR) {
  // pkE é a chave pública do remetente
  // skR é a chave privada do destinatário

  // Executa o algoritmo de Diffie-Hellman
  dh = DH.Diffie-Hellman(skR, pkE)

  // Extrai a chave pública correspondente de skR
  pkRm = getPublicKey(skR)

```

```
35    kem_context = concat(pkE, pkR)
36
37    // Deriva o segredo gerado pelo Diffie-Hellman no segredo
          compartilhado do KEM
38    // por meio do mecanismo "extract-then-expand" do HKDF
39    shared_secret = HKDF.ExtractAndExpand(dh, kem_context)
40
41    return shared_secret
42  }
```

O *handshake* do KEMTLS híbrido (Figura 11) segue o *handshake* do KEMTLS, sem alterações na estrutura do protocolo, no entanto, agora para cada operação de KEM, há uma operação de KEM pós-quântico e uma operação de KEM clássico. A troca de chaves efêmera com KEMs híbridos segue o *internet-draft Hybrid key exchange in TLS 1.3* (STEBILA; FLUHRER; GUERON, 2023): a negociação de KEMs híbridos é feita utilizando a extensão `supported_groups`; as chaves públicas de KEM geradas pelo cliente são concatenadas e transmitidas pela extensão `key_share` do *ClientHello*; os textos cifrados de KEM gerados pelo servidor são concatenados e transmitidos pela extensão `key_share` do *ServerHello*; e, por fim, os segredos compartilhados obtidos da troca de chaves híbrida são concatenados e incorporados ao *key schedule*.

Para a troca de chaves estática, responsável por autenticar o servidor, a negociação dos algoritmos de KEM híbrido se dá pela extensão `signature_algorithms` das mensagens *ClientHello* e *ServerHello*, na qual novos identificadores foram criados para os algoritmos de KEM híbridos. Apesar de esta ser uma extensão de negociação de algoritmos de assinatura e não de KEMs, esta abordagem evita a criação de novos mecanismos de negociação no protocolo, e, dado o caráter experimental do protocolo, esta abordagem é utilizada mesmo não sendo ideal. Na transmissão das chaves públicas de KEM estático, são utilizados certificados X.509 pós-quânticos híbridos de KEM, isto é, certificados digitais assinados por um algoritmo de assinatura pós-quântico híbrido e que a chave pública corresponde à concatenação de uma chave pública de KEM clássico com uma chave pública de KEM pós-quântico.

Ao receber a mensagem *Certificate* contendo o certificado híbrido do servidor, o cliente realiza as operações de encapsulamento para cada chave pública, obtendo os textos cifrados e segredos compartilhados respectivos a cada componente do esquema híbrido. Os textos cifrados são concatenados e transmitidos ao servidor pela mensagem *ClientKEMCiphertext*, e os segredos compartilhados são concatenados e incorporados ao *key schedule*. A partir deste momento, o servidor encontra-se implicitamente autenticado, portanto, o cliente já pode fazer o envio de dados de aplicação, pois

apenas um servidor autêntico[1] conseguirá computar a chave de criptografia necessária para decifrar a mensagem, conforme discutido na subseção 2.7.1.1.

Na finalização do protocolo, são trocadas mensagens *Finished*, geradas da mesma forma que no TLS 1.3, que autenticam o *handshake* e finalizam o protocolo. Com o recebimento da mensagem *Finished* do servidor, o servidor encontra-se explicitamente autenticado, pois apenas um servidor autêntico conseguiria computar o segredo compartilhado estático, e, por conseguinte, computar o HMAC contido na mensagem *Finished* . Assim como o KEMTLS, o KEMTLS híbrido necessita de duas rodadas (*round trip*) para finalizar o *handshake*.

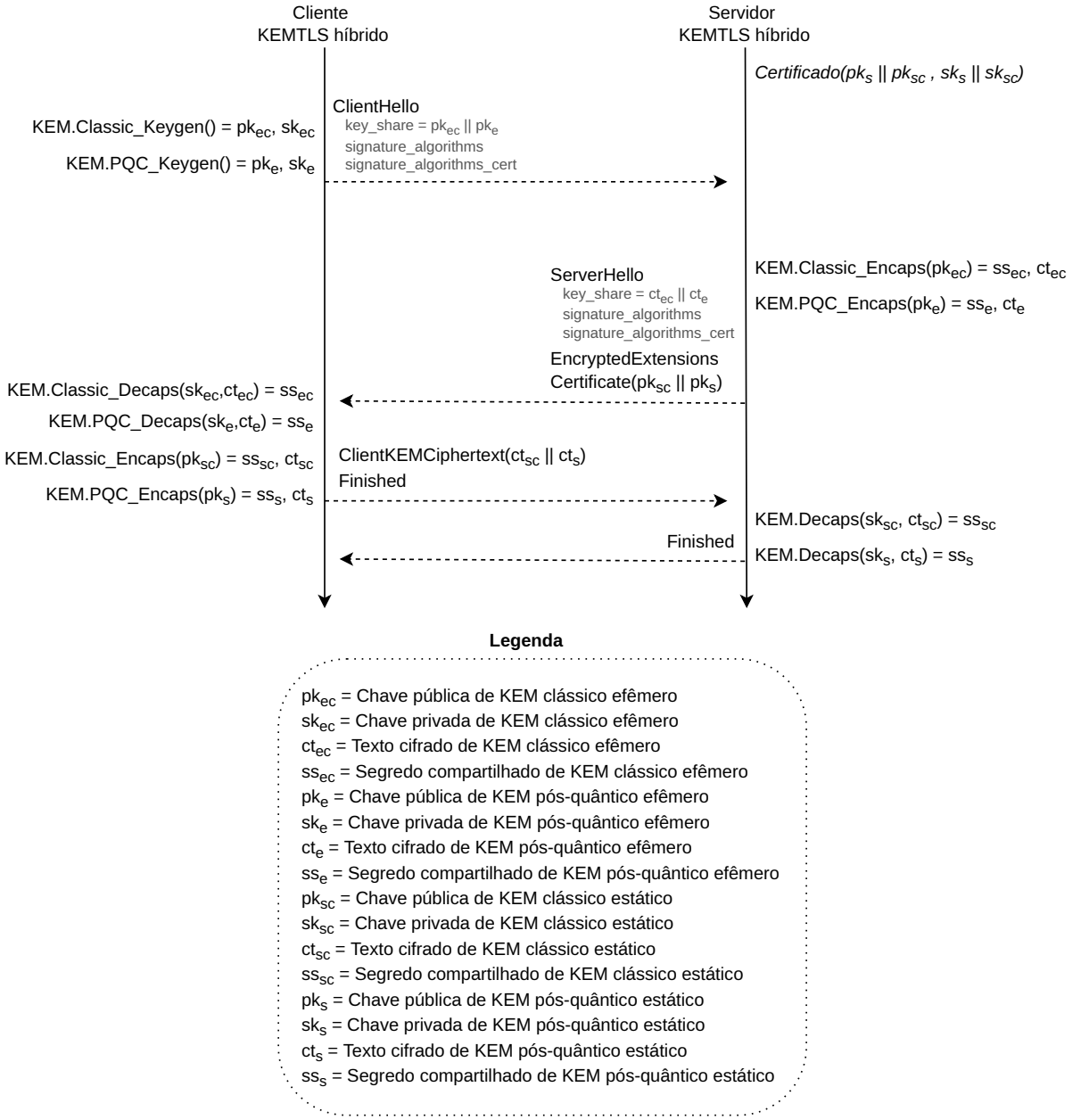


Figura 11 – Protocolo de *handshake* do KEMTLS híbrido (apenas autenticação do servidor)

[1] Servidor autêntico, neste contexto, é aquele que está em posse da chave privada correspondente à chave pública apresentada no certificado.

3.1.1 Key schedule

A fim de incorporar a troca de chaves KEM clássica e a troca de chaves KEM pós-quântica no processo de derivação das chaves de criptografia do protocolo, os segredos compartilhados clássicos e pós-quânticos são concatenados e servidos como IKM para as funções de HKDF-Extract correspondentes (Figura 12). Esta construção, inspirada no *key schedule* do TLS pós-quântico híbrido, corresponde ao combinador criptográfico *dual-PRF* (BINDEL *et al.*, 2019), e, portanto, garante a propriedade híbrida. Desta forma, as chaves de criptografia do protocolo permanecerão seguras enquanto um componente do KEM híbrido se manter seguro.

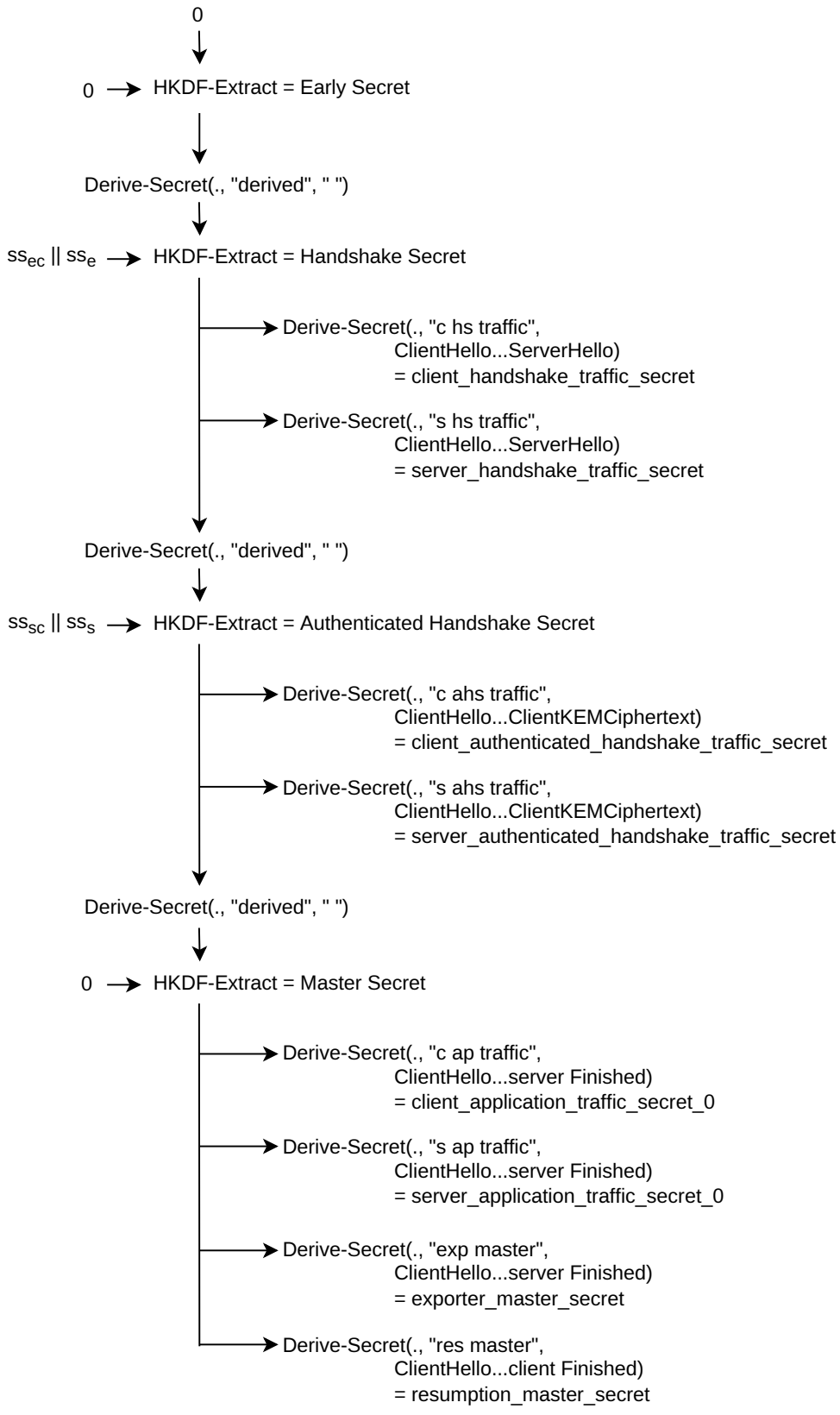


Figura 12 – *Key schedule* do KEMTLS híbrido

3.1.2 KEMTLS-PDK Híbrido

No trabalho anterior, (GIRON *et al.*, 2023), o suporte à criptografia pós-quântica híbrida também foi adicionado ao KEMTLS-PDK, obtendo-se o KEMTLS-PDK híbrido.

Para dar suporte a KEMs híbridos, o KEMTLS-PDK híbrido usou a mesma abordagem que o KEMTLS híbrido: a troca de chaves efêmera segue o *internet-draft Hybrid key exchange in TLS 1.3*; a negociação de KEMs híbridos é feita utilizando as extensões `supported_groups` e `signature_algorithms`; chaves públicas e textos cifrados de KEMs são concatenados para transmissão; e a autenticação é feita utilizando certificados X.509 pós-quânticos híbridos de KEM.

O *handshake* do KEMTLS-PDK híbrido (Figura 13) segue o *handshake* do KEMTLS-PDK, no entanto, para cada operação de KEM do protocolo da especificação original, são realizadas duas operações de KEM, uma com o KEM clássico e outra com o KEM pós-quântico. Assim como no KEMTLS híbrido, não foram necessárias alterações estruturais no protocolo.

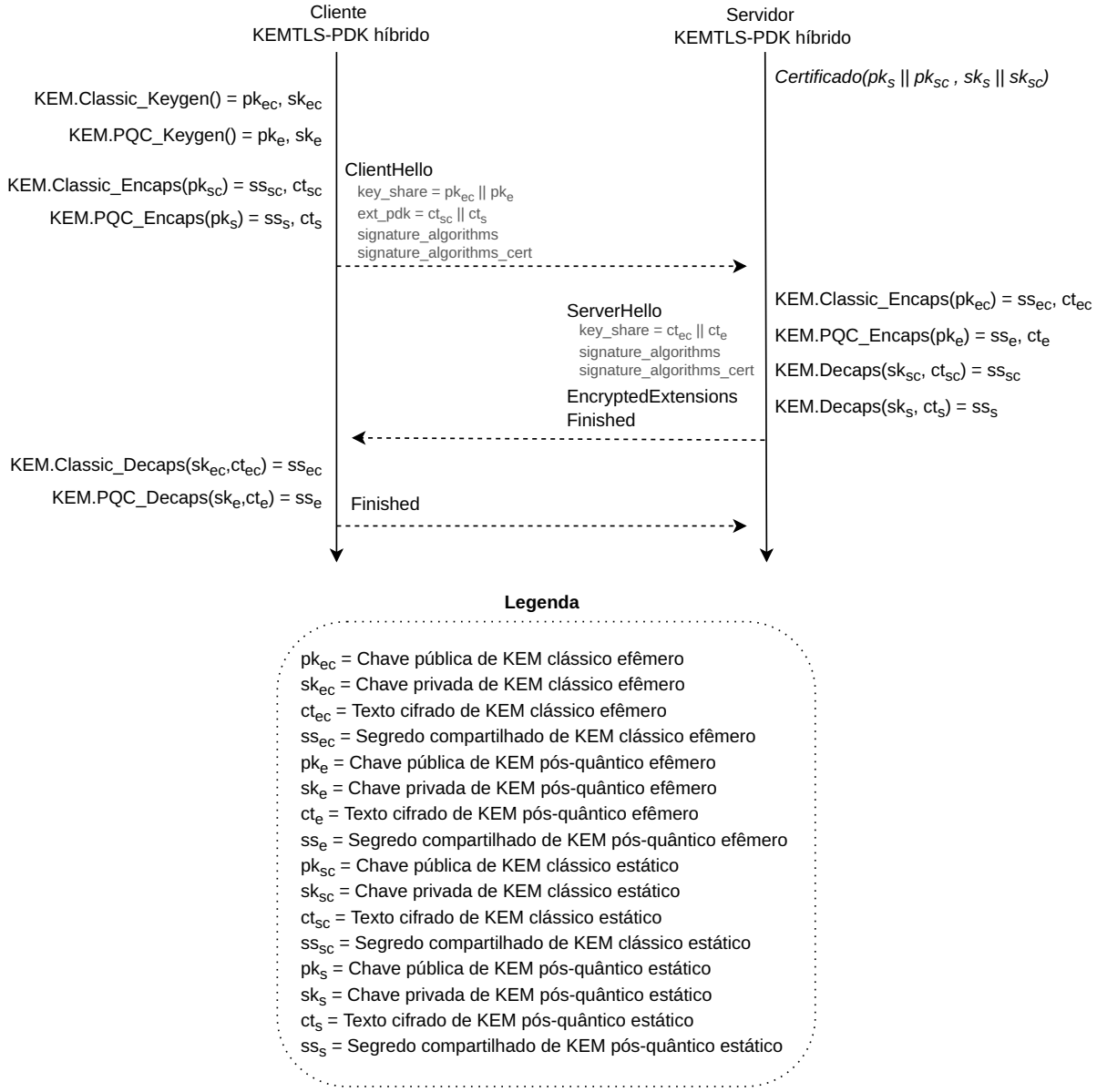
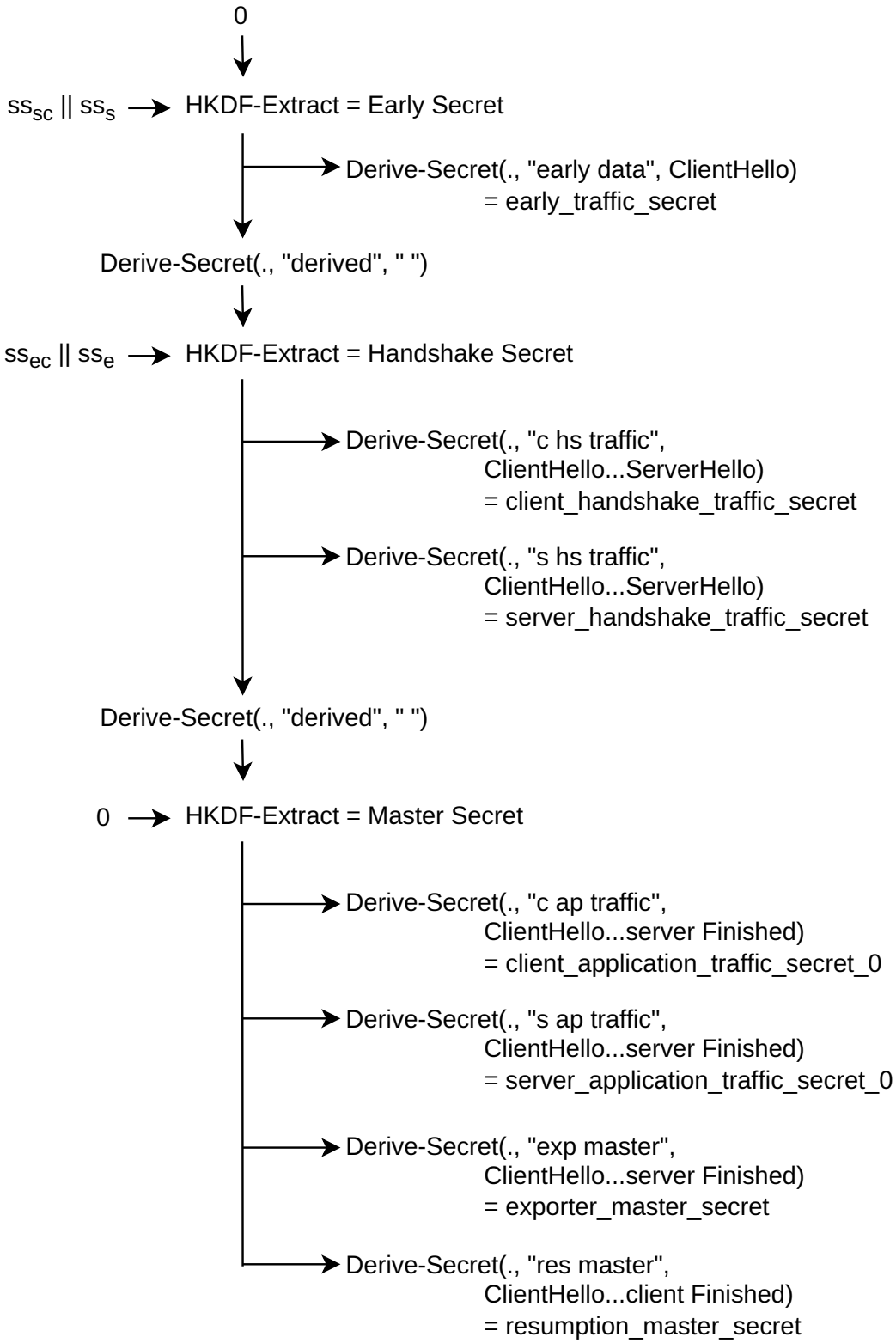


Figura 13 – Protocolo de *handshake* do KEMTLS-PDK híbrido (apenas autenticação do servidor)

3.1.2.1 Key schedule

O *key schedule* do KEMTLS-PDK híbrido (Figura 14) possui a mesma estrutura do *key schedule* do KEMTLS-PDK, sendo a única alteração o *input keying material* usado para os segredos *Early Secret* e *Handshake Secret*: o IKM do *Early Secret* é a concatenação do segredo compartilhado estático clássico com o pós-quântico e o IKM do *Handshake Secret* é a concatenação do segredo compartilhado efêmero clássico com o pós-quântico.

Esta construção corresponde ao combinador criptográfico *dual-PRF* (BINDEL *et al.*, 2019), o que garante a propriedade híbrida, definida na subseção 2.4.1.



Legenda

ss_{ec} = Segredo compartilhado de KEM clássico efêmero
ss_{sc} = Segredo compartilhado de KEM clássico estático
ss_e = Segredo compartilhado de KEM pós-quântico efêmero
ss_s = Segredo compartilhado de KEM pós-quântico estático

Figura 14 – *Key schedule* do KEMTLS-PDK híbrido

3.1.3 Implementação

A implementação do KEMTLS híbrido, publicada no repositório `https://github.com/JPADN/go-hybrid-kemtls`, foi realizada na biblioteca padrão da linguagem de programação *Go*, a partir de uma adaptação de uma implementação do KEMTLS nesta mesma biblioteca, publicada no trabalho (CELI *et al.*, 2021).

A implementação original do KEMTLS em *Go* conta com os algoritmos pós-quânticos de KEM Kyber512 e SIKEp434, implementados nativamente em *Go* na biblioteca CIRCL (*Cloudflare Interoperable Reusable Cryptographic Library*) (CLOUDFLARE, s.d.). A biblioteca CIRCL não possui a implementação de todos os algoritmos do processo de padronização do NIST, limitando dessa forma o escopo dos experimentos aos poucos algoritmos pós-quânticos disponíveis na biblioteca. Por conta disso, para a implementação do KEMTLS híbrido, optou-se por integrar a biblioteca *liboqs* (PROJECT, s.d.[a]) ao projeto.

A biblioteca *liboqs* é uma biblioteca de código-fonte aberto desenvolvida pela *Open Quantum Safe* (STEBILA; MOSCA, 2017) que contém a implementação de uma vasta gama de algoritmos pós-quânticos. Dado que a linguagem de programação desta biblioteca é C, foi utilizada a biblioteca *liboqs-go* (PROJECT, s.d.[b]), uma biblioteca *wrapper*[2] que disponibiliza a API da *liboqs* para a linguagem Go. Por meio da *liboqs-go*, foi possível incorporar algoritmos de KEM e assinatura pós-quânticos à biblioteca padrão do *Go*.

Para este trabalho, para cada algoritmo candidato, buscou-se incorporar uma versão do algoritmo para cada nível de segurança do NIST (MOODY, 2018) dentre 1, 3 e 5 . Os níveis de segurança do NIST significam que o algoritmo é tão difícil de "quebrar"(usando busca exaustiva) quanto o AES-128, para o nível 1, o AES-192, para o nível 3, e o AES-256, para o nível 5. Os seguintes algoritmos de KEM pós-quânticos foram incorporados ao KEMTLS híbrido:

- **Nível de segurança 1:** HQC-128, BIKE-L1, Classic-McEliece-348864;
- **Nível de segurança 3:** HQC-192, BIKE-L3, Classic-McEliece-460896;
- **Nível de segurança 5:** HQC-256, BIKE-L5, Classic-McEliece-6688128.

A fim de incorporar os algoritmos de KEM clássicos ao projeto do KEMTLS híbrido, utilizou-se a biblioteca CIRCL, a qual possui a implementação do protocolo HPKE para o algoritmo de Diffie-Hellman sobre curvas elípticas. No KEMTLS híbrido, o KEM clássico foi instanciado com as seguintes curvas elípticas, recomendadas pelo NIST em (STANDARDS; TECHNOLOGY, 2023): P-256, P-384 e P-521.

[2] Bibliotecas *wrapper*, através do mecanismo de *foreign function interface*, adaptam uma interface já existente, escrita em uma linguagem de programação, para uma interface compatível em outra linguagem de programação.

Apesar de no protocolo do KEMTLS híbrido ser utilizado apenas KEMs, algoritmos de assinatura ainda são necessários na cadeia de certificação do certificado do cliente ou servidor. Por conta disso, foram implementados algoritmos de assinatura pós-quânticos híbridos. Para o componente clássico do algoritmo híbrido, foi utilizado o algoritmo ECDSA (*Elliptic Curve Digital Signature Algorithm*) (STANDARDS; TECHNOLOGY, 2023) e sua implementação da biblioteca padrão do Go para as seguintes curvas elípticas: P-256, P-384 e P-521. Já para o componente pós-quântico, foram utilizados os seguintes algoritmos de assinatura pós-quânticos da biblioteca *liboqs*: no nível de segurança 1, Dilithium 2; no nível de segurança 3, Dilithium 3; e no nível de segurança 5, Dilithium 5. Dentre os algoritmos de assinatura padronizados, o Falcon e o Dilithium são os algoritmos com melhor desempenho conforme as métricas de desempenho publicados pela *open quantum safe*[3]. Apesar de o Dilithium e o Falcon possuirem desempenho semelhante, optou-se por utilizar o Dilithium nos algoritmos de assinatura pós-quântico híbridos, pois este possui parâmetros para os níveis de segurança 1, 3 e 5, enquanto que o Falcon só possui parâmetros para os níveis de segurança 1 e 5.

3.1.3.1 Algoritmos Integrados

Na implementação, os seguintes algoritmos de KEM híbridos foram integrados:

- **Nível de segurança 1:** `P256_HQC_128`, `P256_BIKE_L1`, `P256_Classic_McEliece_348864`;
- **Nível de segurança 3:** `P384_HQC_192`, `P384_BIKE_L3`, `P384_Classic_McEliece_460896`;
- **Nível de segurança 5:** `P521_HQC_256`, `P521_BIKE_L5`, `P521_Classic_McEliece_6688128`.

O nome dos algoritmos é composto de duas partes, a primeira parte indica a curva elíptica utilizada pelo algoritmo de Diffie-Hellman do KEM clássico: o prefixo `P256_` corresponde à curva elíptica P-256; `P384_` à curva P-384; e `P521_` à curva P-521. A segunda parte do nome indica o algoritmo de KEM pós-quântico.

Para os processos que envolvem assinaturas digitais, os seguintes algoritmos de assinatura pós-quânticos híbridos foram integrados:

- **Nível de segurança 1:** `P256_Dilithium2`;
- **Nível de segurança 3:** `P384_Dilithium3`;
- **Nível de segurança 5:** `P521_Dilithium5`.

Novamente, o nome dos algoritmos é composto por duas partes. A primeira parte indica a curva elíptica utilizada pelo algoritmo de ECDSA e a segunda parte indica o algoritmo de assinatura pós-quântico.

[3] Disponível em `https://openquantumsafe.org/benchmarking/visualization/speed_sig.html`

Tanto nos algoritmos híbridos de KEM como nos de assinatura, os algoritmos clássicos sempre são pareados com algoritmos pós-quânticos que possuem o mesmo nível de segurança do NIST.

4 AVALIAÇÃO

4.1 OBJETIVOS

De modo a avaliar a instanciação dos algoritmos candidatos à padronização no KEMTLS híbrido, um conjunto de experimentos foi executado sobre a implementação. Estes experimentos buscam atender os seguintes objetivos:

- Mensurar o desempenho dos algoritmos candidatos em KEMs híbridos;
- Avaliar o impacto dos algoritmos candidatos no KEMTLS híbrido;
- Comparar o desempenho entre os protocolos pós-quânticos híbridos KEMTLS híbrido e TLS pós-quântico híbrido instanciados com os algoritmos candidatos;
- Determinar se o KEMTLS híbrido possui melhor desempenho que o TLS pós-quântico híbrido;
- Determinar o algoritmo candidato com melhor desempenho em protocolos pós-quânticos híbridos.

4.2 METODOLOGIA

Para a avaliação do impacto dos algoritmos candidatos no KEMTLS híbrido, foram conduzidos experimentos utilizando duas máquinas virtuais, uma atuando como cliente e a outra como servidor. Nestas máquinas virtuais, foi instanciado o KEMTLS híbrido e o TLS pós-quântico híbrido, a fim de poder comparar o desempenho do KEMTLS híbrido com outro protocolo pós-quântico híbrido. Em ambos os protocolos, apenas a autenticação do servidor foi habilitada, por conta deste ser o cenário mais comum na navegação na internet.

O algoritmo Kyber, já padronizado pelo NIST na rodada anterior, não foi incluído nos experimentos, pois possui um desempenho significativamente superior aos algoritmos da rodada 4 em termos de latência, conforme observado nas métricas de desempenho publicadas pela *open quantum safe*[1]. Dessa forma, focou-se na avaliação dos candidatos da rodada 4 a fim de obter a melhor configuração dentre eles.

Uma infraestrutura de chaves públicas (ICP) fictícia foi criada para emitir os certificados digitais utilizados nos experimentos. Esta ICP é composta por uma AC raiz e uma AC emissora, ambas certificadas por um algoritmo de assinatura pós-quântico híbrido. O algoritmo de assinatura das ACs foi escolhido de modo a possuir um nível de segurança correspondente ao nível da hierarquia em que a AC se encontra: `P521_Dilithium5` para a AC raiz e `P384_Dilithium3` para a AC emissora. Nos experimentos, o cliente foi

[1] Disponível em `https://openquantumsafe.org/benchmarking/visualization/speed_kem.html`

configurado para confiar na AC raiz. Por conta disso, durante a conexão, apenas é transmitido o certificado folha do servidor e o certificado da AC emissora, para que deste modo o cliente possa estabelecer confiança na cadeia de certificação.

Foram realizados três experimentos, numerados de 0 a 2.

Experimento 0: Inicialmente, buscou-se mensurar o desempenho dos algoritmos que foram integrados no KEMTLS híbrido (subseção 3.1.3.1) isoladamente, isto é, sem estarem instanciados no protocolo. Para os KEMs, foi medido o tempo de execução de cada algoritmo para cada operação de KEM: geração de chaves, encapsulamento, desencapsulamento. Para os algoritmos de assinatura, foi medido o tempo de execução de cada algoritmo para as operações: assinatura do `handshake_transcript`, simulando dessa forma a assinatura realizada no KEMTLS híbrido e TLS; e verificação de assinatura. Este experimento inicial teve como objetivo evidenciar o custo computacional dos algoritmos integrados considerando a latência adicional que a biblioteca *wrapper liboqs-go* adiciona ao desempenho dos algoritmos da biblioteca *liboqs*, conforme apontado em (CELI *et al.*, 2021).

Experimento 1: No primeiro experimento, foram executados 1000 *handshakes* para cada algoritmo de KEM no KEMTLS híbrido e no TLS pós-quântico híbrido. Como no KEMTLS são utilizados KEMs tanto para troca de chaves como para autenticação, utilizou-se o mesmo algoritmo para ambos os processos. No TLS, o algoritmo de KEM foi utilizado apenas na troca de chaves, e para a autenticação foram utilizados os algoritmos `P256_Dilithium2`, `P384_Dilithium3` e `P521_Dilithium5`, de acordo com o nível de segurança do KEM. O KEM Classic McEliece foi excluído deste experimento pelo fato de possuir uma chave pública de tamanho superior ao tamanho limite estabelecido para pacotes TLS. Para fins de comparação, buscou-se agrupar os resultados dos experimentos pelo nível de segurança dos algoritmos utilizados na conexão, resultando nas seguintes configurações:

Tabela 1 – Configurações do experimento 1

Protocolo	Nível	Troca de Chaves	Autenticação	Abreviatura
KEMTLS híbrido	1	P256_HQC_128	P256_HQC_128	L1-HQC-HQC
		P256_BIKE_L1	P256_BIKE_L1	L1-BIKE-BIKE
	3	P384_HQC_192	P384_HQC_192	L3-HQC-HQC
		P384_BIKE_L3	P384_BIKE_L3	L3-BIKE-BIKE
	5	P521_HQC_256	P521_HQC_256	L5-HQC-HQC
		P521_BIKE_L5	P521_BIKE_L5	L5-BIKE-BIKE
TLS pós-quântico híbrido	1	P256_HQC_128	P256_Dilithium2	L1-HQC-DIL
		P256_BIKE_L1	P256_Dilithium2	L1-BIKE-DIL
	3	P384_HQC_192	P384_Dilithium3	L3-HQC-DIL
		P384_BIKE_L3	P384_Dilithium3	L3-BIKE-DIL
	5	P521_HQC_256	P521_Dilithium5	L5-HQC-DIL
		P521_BIKE_L5	P521_Dilithium5	L5-BIKE-DIL

Experimento 2: O segundo experimento buscou avaliar o desempenho dos protocolos pós-quânticos híbridos em cenários em que há *cache* de certificados. Para isto, foram executados 1000 *handshakes* para cada algoritmo de KEM no KEMTLS-PDK híbrido e no TLS pós-quântico híbrido com a extensão *Cached Information*. Para a troca de chaves, em ambos os protocolos, foram utilizados os algoritmos BIKE e HQC. Classic McEliece foi novamente excluído da troca de chaves por conta do tamanho de sua chave pública. Para a autenticação no KEMTLS-PDK híbrido, no entanto, o Classic McEliece foi incluído, dado que sua chave pública não precisa ser transmitida no *handshake*, uma vez que há cache de certificados. Para a autenticação do TLS pós-quântico híbrido, são novamente utilizados algoritmos de assinatura pós-quânticos híbridos. Neste experimento, as seguintes configurações foram aplicadas aos protocolos:

Tabela 2 – Configurações do experimento 2

Protocolo	Nível	Troca de Chaves	Autenticação	Abreviatura
KEMTLS-PDK híbrido	1	P256_HQC_128	P256_HQC_128	L1-HQC-HQC
		P256_BIKE_L1	P256_BIKE_L1	L1-BIKE-BIKE
		P256_HQC_128	P256_Classic_McEliece_348864	L1-HQC-CM
		P256_BIKE_L1	P256_Classic_McEliece_348864	L1-BIKE-CM
	3	P384_HQC_192	P384_HQC_192	L3-HQC-HQC
		P384_BIKE_L3	P384_BIKE_L3	L3-BIKE-BIKE
		P384_HQC_192	P384_Classic_McEliece_460896	L3-HQC-CM
		P384_BIKE_L3	P384_Classic_McEliece_460896	L3-BIKE-CM
	5	P521_HQC_256	P521_HQC_256	L5-HQC-HQC
		P521_BIKE_L5	P521_BIKE_L5	L5-BIKE-BIKE
		P521_HQC_256	P521_Classic_McEliece_6688128	L5-HQC-CM
		P521_BIKE_L5	P521_Classic_McEliece_6688128	L5-BIKE-CM
TLS pós-quântico híbrido (*Cached Information*)	1	P256_HQC_128	P256_Dilithium2	L1-HQC-DIL
		P256_BIKE_L1	P256_Dilithium2	L1-BIKE-DIL
	3	P384_HQC_192	P384_Dilithium3	L3-HQC-DIL
		P384_BIKE_L3	P384_Dilithium3	L3-BIKE-DIL
	5	P521_HQC_256	P521_Dilithium5	L5-HQC-DIL
		P521_BIKE_L5	P521_Dilithium5	L5-BIKE-DIL

4.2.1 Implementação

Os experimentos realizados foram implementados no repositório *hybrid-kemtls-tests*, disponível em `https://github.com/JPADN/hybrid_kemtls_tests`.

4.2.2 Ambiente

Para avaliar o impacto dos algoritmos candidatos no KEMTLS híbrido, buscou-se mensurar o desempenho deste em um ambiente realista, onde não há controle sobre as diversas variáveis que podem afetar uma rede, tais como perda de pacotes, variação na latência, congestionamentos de rede, etc. Para isto, foi utilizada a plataforma de *cloud computing Google Cloud Platform* (GCP). Nesta plataforma, para os experimentos 1 e 2, foram alocadas duas máquinas virtuais geograficamente distantes, uma atuando

como cliente da conexão e outra como servidor. O tipo de máquina virtual utilizado foi o N2 (2 vCPU e 8 GB de memória RAM), recomendada para uso em servidores *web* de médio tráfego e microsserviços em contêineres, de acordo com a documentação[2]. O experimento 0, por se tratar de um experimento local, foi executado em apenas uma destas máquinas virtuais.

A alocação de máquinas virtuais geograficamente distantes buscou simular a latência média das conexões realizadas no dia-a-dia da internet. Por meio de um experimento prático com o programa *ping* do Linux, foi medida a latência das conexões do computador do autor a servidores de aplicações frequentemente acessadas pela população, tais como *YouTube* e *Instagram*. Dessas medições, foi observada uma latência média de 32 ms para o *YouTube*, servido pela CDN do *Google*, e uma latência média de 41 ms para o *Instagram*, servido pela CDN do *Facebook*. A fim de obter uma latência similar nos experimentos, optou-se por instanciar uma máquina virtual em São Paulo - Brasil (região *southamerica-east1* do GCP), atuando como cliente da conexão, e outra em Santiago - Chile (região *southamerica-west1* do GCP), atuando como servidor da conexão. A conexão entre estas duas regiões é de 50 ms, próxima da latência média observada no mundo real.

4.3 MÉTRICAS DE AVALIAÇÃO

Os experimentos 0, 1 e 2 foram avaliados com base nas seguintes métricas de avaliação.

4.3.1 Tempo de execução de algoritmos

A fim de mensurar o desempenho dos KEMs híbridos e algoritmos de assinatura pós-quânticos híbridos no experimento 0, utilizou-se como métrica de avaliação o tempo de execução médio dos algoritmos de KEMs e assinatura digital e o tamanho em bytes dos objetos criptográficos transmitidos por estes (chave pública, assinatura, etc.).

Na obtenção do tempo de execução médio, cada algoritmo foi executado 1000 vezes e o tempo de execução das seguintes operações foi mensurado:

- **Algoritmos de KEM:**
  - Geração de chaves;
  - Encapsulamento;
  - Desencapsulamento.

[2] Documentação da família de máquinas virtuais N2 da GCP disponível em `https://cloud.google.com/compute/docs/general-purpose-machines?hl=pt-br`.

- **Algoritmos de assinatura digital:**
  - Assinatura;
  - Verificação.

No experimento 0, os seguintes objetos criptográficos tiveram seu tamanho mensurado em bytes:

- **Algoritmos de KEM:**
  - Chave pública;
  - Texto cifrado.
- **Algoritmos de assinatura digital:**
  - Chave pública;
  - Assinatura.

4.3.2 *Time to send app data*

4.3.2.1 Experimento 1

No experimento 1, buscou-se avaliar a latência do *handshake* para os protocolos KEMTLS híbrido com o TLS pós-quântico híbrido. A métrica mais comum utilizada na literatura é mensurar o tempo de handshake, que corresponde ao tempo decorrido entre a primeira mensagem do protocolo, o *ClientHello*, e a última mensagem do protocolo, o *Finished*. No entanto, esta métrica não é justa para fins de comparação destes protocolos, uma vez que o KEMTLS híbrido leva duas rodadas para finalizar o protocolo, enquanto o TLS pós-quântico híbrido leva apenas uma rodada. Apesar disso, em ambos os protocolos, o cliente pode enviar dados de aplicação em uma única rodada, pois basta uma rodada para que o cliente tenha as chaves de criptografia da conexão computadas. Desta forma, a métrica de avaliação usada no experimento 1 é o tempo decorrido do começo do *handshake* até o momento em que o cliente pode enviar dados de aplicação, que coincide com o instante em que a mensagem *Finished* do cliente é enviada. Esta métrica foi nomeada como *time to send app data*. Para o experimento 1, esta métrica está ilustrada na Figura 15.

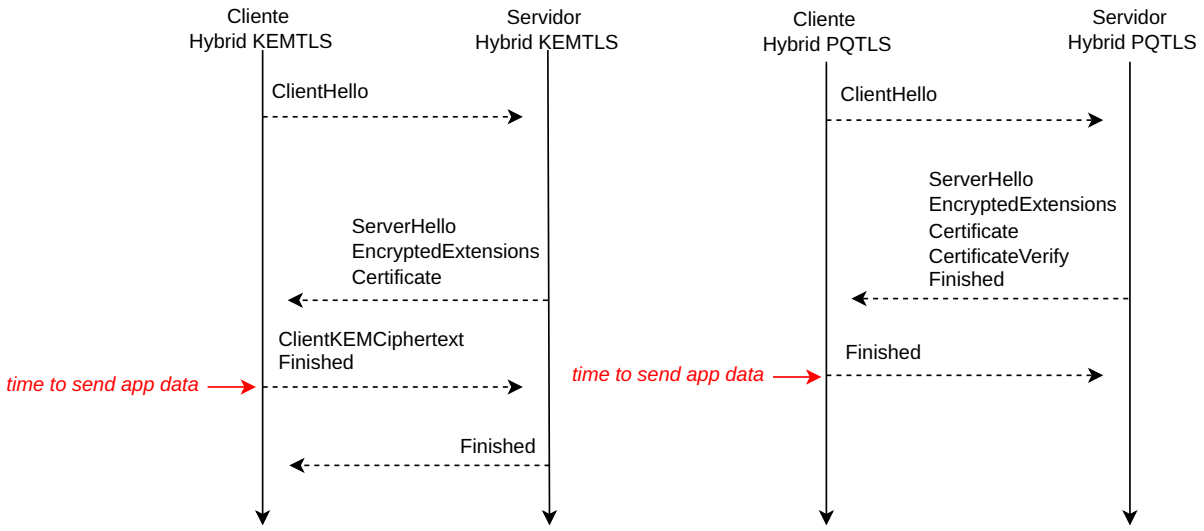


Figura 15 – Métrica de avaliação *time to send app data* nos protocolos KEMTLS híbrido e TLS pós-quântico híbrido

4.3.2.2 Experimento 2

No experimento 2, ambos os protocolos avaliados finalizam o *handshake* em uma única rodada, pois a cache de certificados dispensa a rodada adicional existente no *handshake* do KEMTLS híbrido. Por conta disso, o tempo decorrido do começo do *handshake* até o momento em que o cliente pode enviar dados de aplicação, referenciado como *time to send app data*, corresponde ao tempo de *handshake* de ambos os protocolos.

Desta forma, a métrica de avaliação utilizada no experimento 2 é a mesma do experimento 1, o *time to send app data*, no entanto, vale ressaltar que neste experimento, esta métrica engloba o *handshake* completo. Para o experimento 2, esta métrica está ilustrada na Figura 16.

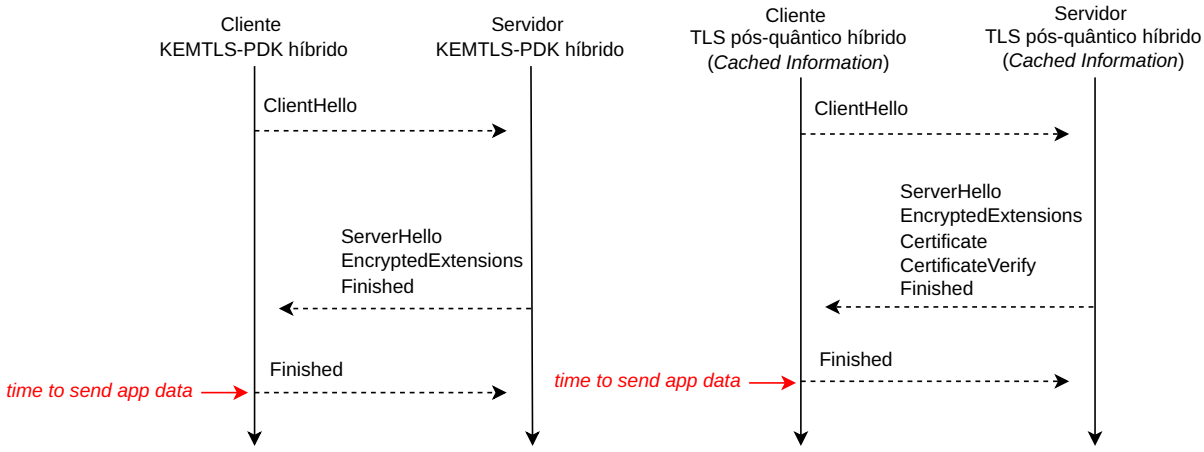


Figura 16 – Métrica de avaliação *time to send app data* nos protocolos KEMTLS-PDK híbrido e TLS pós-quântico híbrido (*Cached Information*)

4.3.3 Custo de autenticação

Os experimentos 1 e 2 tratam-se de experimentos comparativos entre os protocolos pós-quânticos híbridos KEMTLS híbrido e TLS pós-quântico híbrido. O *time to send app data* de cada protocolo está diretamente relacionado às operações criptográficas realizadas durante o handshake. Em ambos os protocolos, a troca de chaves é realizada da mesma forma, por meio de uma troca de chaves KEM efêmera, e, portanto, o impacto do algoritmo de troca de chaves é o mesmo nos dois protocolos. A autenticação, no entanto, difere entre os dois protocolos, tendo em vista que o KEMTLS híbrido utiliza KEMs estáticos para autenticação e o TLS pós-quântico híbrido utiliza assinaturas digitais. Desta forma, mapeou-se o custo de autenticação das configurações utilizadas nos experimentos 1 e 2.

O custo de autenticação refere-se ao tempo gasto pelo algoritmo de autenticação e a quantidade de bytes transmitidos no fluxo de autenticação contido na métrica *time to send app data*. O tempo gasto pelo algoritmo corresponde à soma dos tempos de execução presentes nas tabelas 8 e 10 para as operações criptográficas realizadas. A quantidade de bytes transmitidos corresponde à soma dos tamanhos presentes nas tabelas 9 e 11 para os objetos criptográficos transmitidos.

4.3.3.1 Experimento 1

No experimento 1, o custo de autenticação do KEMTLS híbrido, mapeado na Tabela 3, envolve uma operação de encapsulamento e a transmissão de uma chave pública de KEM. É importante notar que neste experimento a métrica *time to send app data* não engloba a operação de desencapsulamento com o KEM estático, evidenciada pelas Figuras 15 e 11. Já o custo de autenticação do TLS pós-quântico híbrido, mapeado na Tabela 4, envolve as operações de assinatura e verificação, e a transmissão de uma chave pública e uma assinatura.

Tabela 3 – Custo de autenticação do KEMTLS híbrido no experimento 1

Configuração	Custo de tempo (ms)	Custo de transmissão (bytes)
L1-HQC-HQC	4,30	2314
L1-BIKE-BIKE	0,17	1606
L3-HQC-HQC	13,38	4619
L3-BIKE-BIKE	1,00	3180
L5-HQC-HQC	37,93	7378
L5-BIKE-BIKE	13,88	5255

Tabela 4 – Custo de autenticação do TLS pós-quântico híbrido no experimento 1

Configuração	Custo de tempo (ms)	Custo de transmissão (bytes)
L1-HQC-DIL e L1-BIKE-DIL	0,28	3873
L3-HQC-DIL e L3-BIKE-DIL	12,47	5450
L5-HQC-DIL e L5-BIKE-DIL	20,93	7464

4.3.3.2 Experimento 2

Para o experimento 2, o custo de autenticação do KEMTLS-PDK híbrido (Tabela 5) envolve as operações de encapsulamento e desencapsulamento, e a transmissão de um texto cifrado. Observa-se, portanto, que o custo de autenticação do KEMTLS-PDK híbrido difere do custo do KEMTLS híbrido, abordado no experimento 1. Isto se dá, pois, enquanto no KEMTLS híbrido a métrica *time to send app data* engloba apenas a autenticação implícita do servidor, esta mesma métrica no KEMTLS-PDK híbrido engloba a autenticação explícita do servidor.

O custo de autenticação do TLS pós-quântico híbrido com *Cached Information* (6) envolve as mesmas operações que o TLS pós-quântico híbrido, assinatura e verificação, no entanto, apenas a transmissão da assinatura está incluída em seu custo, uma vez que o cliente já possui a chave pública em sua *cache*.

Tabela 5 – Custo de autenticação do KEMTLS-PDK híbrido no experimento 2

Configuração	Custo de tempo (ms)	Custo de transmissão (bytes)
L1-HQC-HQC	10,76	4498
L1-BIKE-BIKE	0,81	1638
L1-HQC-CM e L1-BIKE-CM	0,26	161
L3-HQC-HQC	33,64	9075
L3-BIKE-BIKE	3,67	3212
L3-HQC-CM e L3-BIKE-CM	1,64	253
L5-HQC-HQC	86,21	14554
L5-BIKE-BIKE	32,09	5287
L5-HQC-CM e L5-BIKE-CM	27,49	341

Tabela 6 – Custo de autenticação do TLS pós-quântico híbrido com *Cached Information* no experimento 2

Configuração	Custo de tempo (ms)	Custo de transmissão (bytes)
L1-HQC-DIL e L1-BIKE-DIL	0,28	2494
L3-HQC-DIL e L3-BIKE-DIL	12,47	3399
L5-HQC-DIL e L5-BIKE-DIL	20,93	4737

4.4 SUMÁRIO DE AVALIAÇÃO

A Tabela 7 contém um sumário de avaliação dos experimentos 0, 1 e 2.

Tabela 7 – Sumário de avaliação dos experimentos

Exp.	**Ambiente**	**Métricas de avaliação**	**Protocolos comparados**
0	Uma máquina virtual N2	Tempo de execução dos algoritmos	Nenhum
1	2 máquinas virtuais N2 (SP, Brasil - Santiago, Chile)	*Time to send app data* Custo de autenticação	KEMTLS híbrido e TLS pós-quântico híbrido
2	2 máquinas virtuais N2 (SP, Brasil - Santiago, Chile)	*Time to send app data* Custo de autenticação	KEMTLS-PDK híbrido e TLS pós-quântico híbrido (*Cached Information*)

4.5 RESULTADOS

4.5.1 Experimento 0

O experimento 0 foi dividido em duas partes: avaliação dos KEMs híbridos, e avaliação dos algoritmos de assinatura pós-quânticos híbridos.

4.5.1.1 KEMs híbridos

Na Tabela 8, encontram-se os tempos de execução médios de cada KEM híbrido integrado nos protocolos.

Tabela 8 – Tempo de execução médio das operações de KEM em milissegundos

Nível	KEM	Ger. Chave	Encaps.	Decaps.
1	P256_HQC_128	2,10	4,30	6,46
1	P256_BIKE_L1	0,30	0,17	0,64
1	P256_Classic_McEliece_348864	48,47	0,12	0,14
3	P384_HQC_192	6,72	13,38	20,26
3	P384_BIKE_L3	1,22	1,00	2,67
3	P384_Classic_McEliece_460896	170,14	0,80	0,84
5	P521_HQC_256	18,62	37,93	48,28
5	P521_BIKE_L5	8,89	13,88	18,21
5	P521_Classic_McEliece_6688128	235,69	13,93	13,56

Para as operações de encapsulamento e desencapsulamento, o algoritmo híbrido com menor tempo de execução foi o híbrido Classic McEliece, chegando a possuir um tempo de execução médio inferior a 1 ms para os níveis de segurança 1 e 3. Em segundo lugar, considerando um ordenamento pelo tempo de execução, temos o híbrido BIKE, com tempos de execução relativamente próximos ao híbrido Classic McEliece. Em último lugar, temos o híbrido HQC, com tempos médios consideravelmente superiores em relação aos outros algoritmos avaliados. Essa disparidade é maior no nível 1, em que o híbrido HQC chega a ser quarenta vezes mais lento que o híbrido Classic McEliece para a operação de desencapsulamento.

Apesar de o híbrido Classic McEliece ter um excelente desempenho para as operações de encapsulamento e desencapsulamento, a operação de geração de chaves possui um tempo de execução extremamente alto, chegando a casa das centenas de milissegundos. Diferentemente do Classic McEliece, o híbrido com BIKE mantém um tempo de execução baixo para a geração de chaves, similar aos tempos de encapsulamento e desencapsulamento. O híbrido com HQC continua tendo um desempenho inferior ao com BIKE, no entanto, para a geração de chaves, possui um desempenho superior ao Classic McEliece.

Quando se trata de protocolos de comunicação em rede, a avaliação do impacto de um algoritmo de criptografia não se restringe ao seu tempo de execução, sendo

necessário também mensurar a quantidade de bytes transmitidos pelo algoritmo na conexão. Em um protocolo de KEM, dois componentes públicos são transmitidos na conexão: a chave pública do KEM e o texto cifrado. A Tabela 9 contém, para cada KEM híbrido avaliado, o tamanho em bytes de chave pública e do texto cifrado.

Tabela 9 – Tamanho em bytes da chave pública e do texto cifrado para os KEMs

Nível	KEM	Chave pública	Texto cifrado
1	P256_HQC_128	2314	4498
1	P256_BIKE_L1	1606	1638
1	P256_Classic_McEliece_348864	261185	161
3	P384_HQC_192	4619	9075
3	P384_BIKE_L3	3180	3212
3	P384_Classic_McEliece_460896	524257	253
5	P521_HQC_256	7378	14554
5	P521_BIKE_L5	5255	5287
5	P521_Classic_McEliece_6688128	1045125	341

O algoritmo híbrido com menor chave pública é o híbrido BIKE, com tamanhos na casa dos *kilobytes*. Em segundo lugar, o híbrido HQC, com chaves públicas com cerca de 1 KB de tamanho maior que as do BIKE. O algoritmo híbrido com maior chave pública foi o Classic McEliece, que, para o nível de segurança 5, possui uma chave pública com tamanho na casa dos *megabytes*.

No que diz respeito ao texto cifrado, o Classic McEliece se destacou com um texto cifrado bastante menor em relação aos outros algoritmos avaliados. O BIKE vem logo em seguida, com textos cifrados de *kilobytes* de tamanho. E por fim, o HQC, com textos cifrados cerca de três vezes maiores que o BIKE.

4.5.1.2 Algoritmos de assinatura pós-quânticos híbridos

A Tabela 10 contém o tempo de execução médio das operações de assinatura e verificação para cada algoritmo de assinatura pós-quântico híbrido integrado nos protocolos.

Tabela 10 – Tempo médio de execução dos algoritmos de assinatura pós-quânticos híbridos em milissegundos

Nível	Algoritmo de Assinatura	Assinatura	Verificação
1	P256_Dilithium2	0,14	0,14
3	P384_Dilithium3	4,34	8,13
5	P521_Dilithium5	7,23	13,7

Nos protocolos avaliados, dois componentes públicos do algoritmo de assinatura são transmitidos na conexão: a chave pública e a assinatura. A Tabela 11 contém os tamanhos em bytes da chave pública e da assinatura para cada algoritmo de assinatura pós-quântico híbrido integrado.

Tabela 11 – Tamanho em bytes da chave pública e da assinatura dos algoritmos de assinatura pós-quânticos híbridos integrados

Nível	Algoritmo de Assinatura	Chave pública	Assinatura
1	P256_Dilithium2	1379	2494
3	P384_Dilithium3	2051	3399
5	P521_Dilithium5	2727	4737

4.5.2 Experimento 1

4.5.2.1 Representação dos resultados

Os resultados do experimento encontram-se nos gráficos 17, 18 e 19. Nestes gráficos, cada coluna representa o *time to send app data* médio do protocolo para a configuração descrita no eixo x. Para facilitar a comparação entre os protocolos, as colunas foram agrupadas pelo algoritmo de troca de chaves utilizado: o grupo da esquerda agrupa as configurações onde a troca de chaves é feita com o híbrido HQC, e o grupo da direita o híbrido BIKE.

A dispersão dos dados amostrados foi representada pela barra de erro no topo da coluna, que corresponde ao desvio padrão dos dados.

4.5.2.2 Análise dos resultados

Para todos os níveis de segurança, o híbrido BIKE desempenhou melhor que o híbrido HQC, o que é esperado, uma vez que o tempo de execução do híbrido BIKE é consideravelmente menor para todas as operações criptográficas, assim como o tamanho de sua chave pública e texto cifrado, conforme as tabelas 8 e 9.

Quanto à comparação entre os protocolos, no nível 1 (Figura 17), ambos os protocolos tiveram desempenho equivalente para o híbrido HQC, enquanto que para o híbrido BIKE o TLS pós-quântico híbrido demonstrou uma vantagem significativa, com um *time to send app data* 20 ms menor em relação ao KEMTLS híbrido. Este resultado vai contra o resultado esperado, uma vez que o custo de autenticação do L1-BIKE-BIKE é inferior ao custo do L1-BIKE-DIL.

Para o nível 3 (Figura 18), novamente ambos os protocolos tiveram um desempenho equivalente para o híbrido HQC, no entanto, neste nível de segurança, o KEMTLS híbrido superou o desempenho do TLS pós-quântico híbrido para o híbrido BIKE, apresentando um *time to send app data* médio 15 ms menor. Ao comparar o custo de autenticação das configurações, observa-se que o L3-BIKE-BIKE custa 1 ms de tempo de execução e 3180 bytes para transmissão, enquanto o L3-BIKE-DIL custa 12,47 ms de tempo de execução e 5450 bytes para transmissão, o que justifica o desempenho superior do L3-BIKE-BIKE.

No nível mais alto de segurança, nível 5 (Figura 19), o TLS pós-quântico híbrido teve uma latência menor que o KEMTLS híbrido para o híbrido HQC, justificável pelo

custo de tempo de autenticação inferior do L5-HQC-DIL, de 20,93 ms, em comparação ao L5-HQC-HQC, de 37,93 ms, uma vez que ambas as configurações transmitem um custo de transmissão muito próximo. Já para o híbrido BIKE, ambos os protocolos tiveram um desempenho muito próximo, com o KEMTLS híbrido com um *time to send app data* levemente inferior ao TLS pós-quântico híbrido, o que está de acordo com o custo de autenticação de cada protocolo: L5-BIKE-BIKE possui um custo de autenticação de 13,88 ms e 5255 bytes, levemente inferior ao do L5-BIKE-DIL, de 20,93 ms e 7674 bytes.

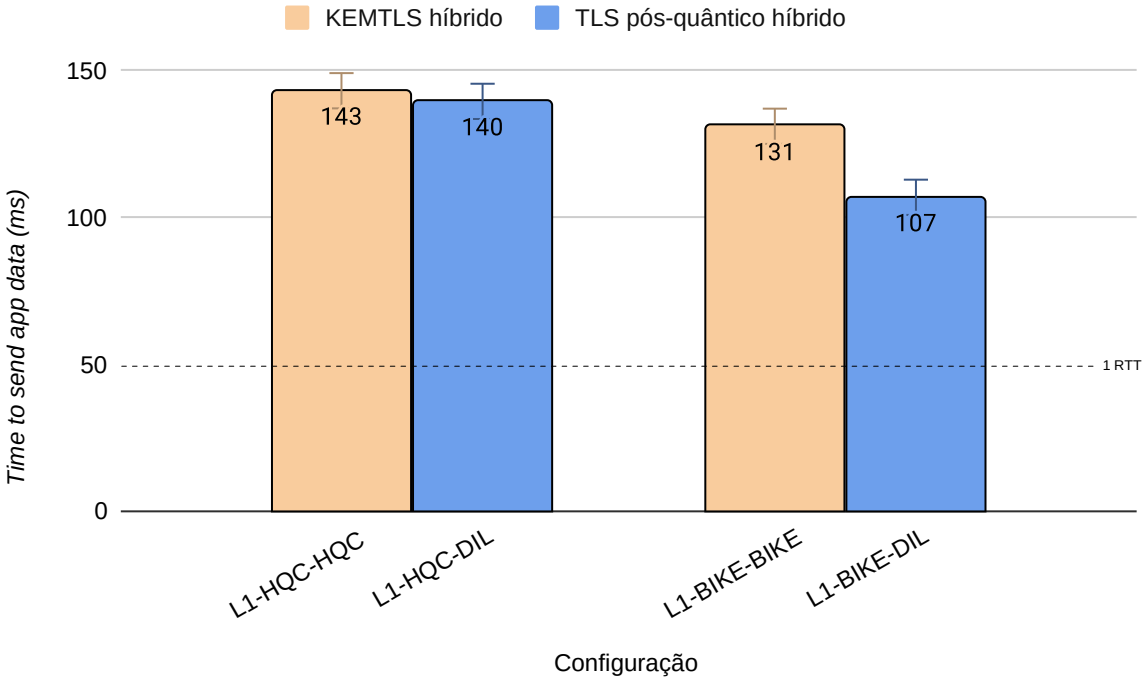


Figura 17 – *Time to send app data* médio do experimento 1 para o nível de segurança 1

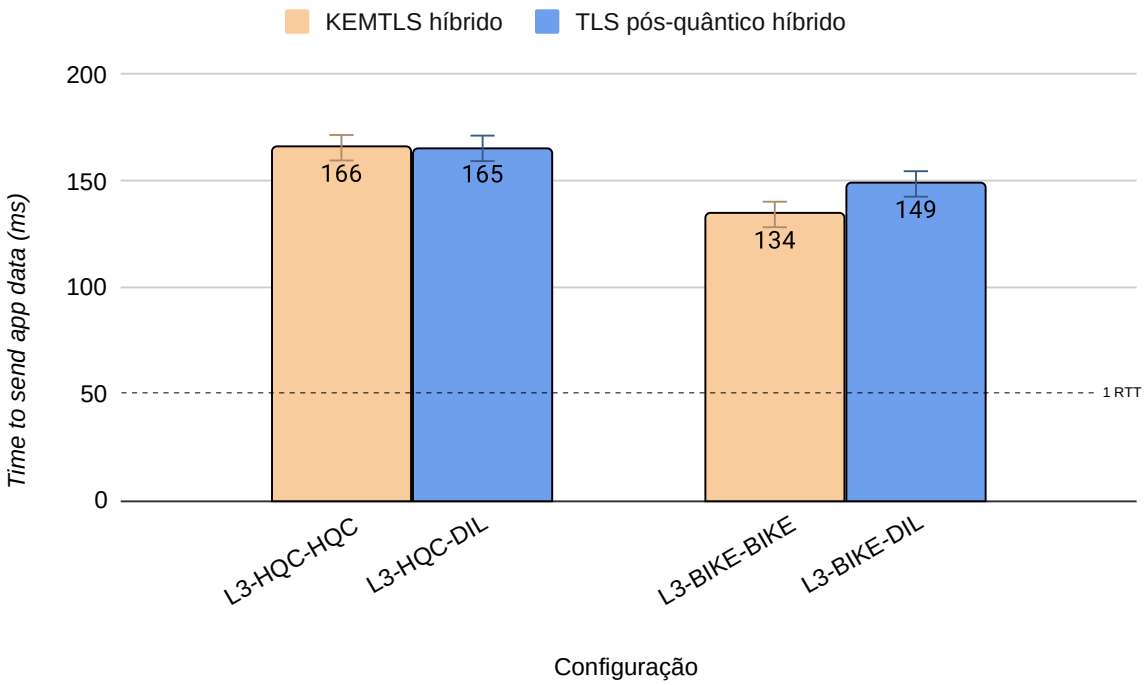


Figura 18 – *Time to send app data* médio do experimento 1 para o nível de segurança 3

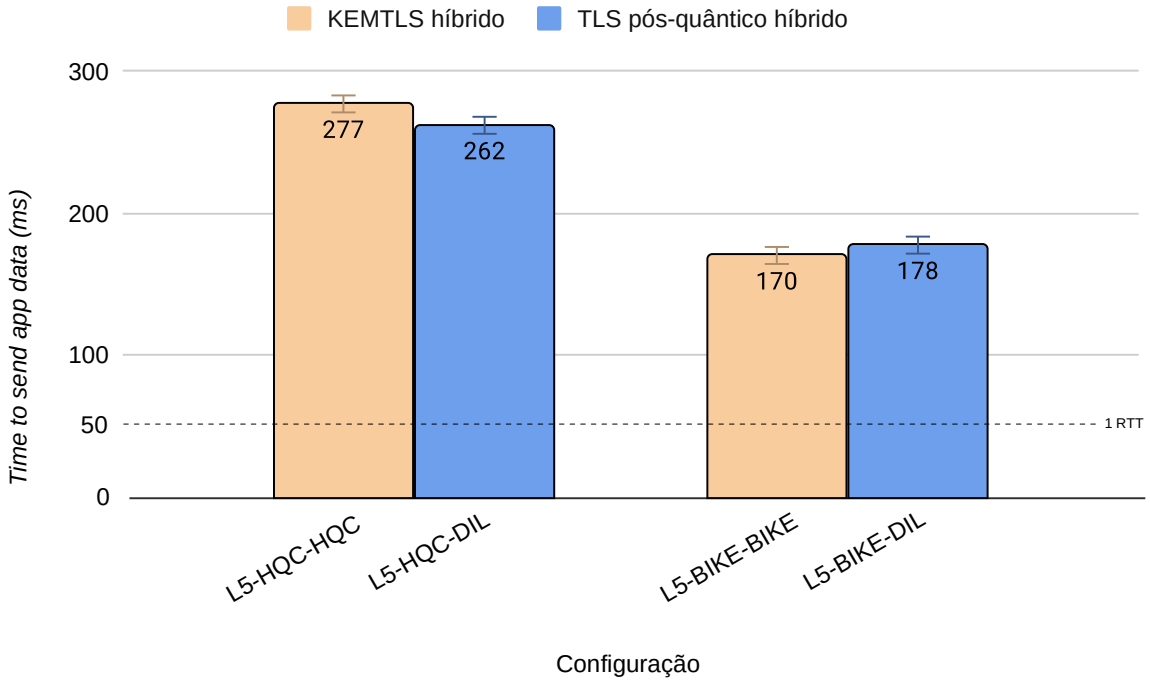


Figura 19 – *Time to send app data* médio do experimento 1 para o nível de segurança 5

4.5.3 Experimento 2

4.5.3.1 Representação dos resultados

Os resultados do experimento 2 encontram-se nos gráficos 20, 21 e 22. A interpretação dos gráficos é a mesma do experimento 1, conforme subseção 4.5.2.1. Diferentemente do experimento 1, no experimento 2 o KEMTLS-PDK híbrido foi avaliado com dois tipos de configuração, uma em que o algoritmo de KEM estático é o mesmo que o de KEM efêmero, e outra em que o algoritmo de KEM estático é o Classic McEliece. Por conta disso, cada agrupamento de colunas possui duas colunas para o KEMTLS-PDK híbrido.

4.5.3.2 Análise dos resultados

Assim como no experimento 1, as configurações com o híbrido BIKE na troca de chaves tiveram um desempenho superior às configurações com híbrido HQC, para todos os níveis de segurança.

Para o nível de segurança 1 (Figura 20), considerando o híbrido HQC na troca de chaves (agrupamento de colunas à esquerda), o KEMTLS-PDK híbrido e TLS pós-quântico híbrido obtiveram um *time to send app data* equivalente para as configurações L1-HQC-CM e L1-HQC-DIL, apesar de o L1-HQC-CM possuir um custo de transmissão consideravelmente menor. Considerando o híbrido BIKE na troca de chaves (agrupamento de colunas à direita), todas as configurações obtiveram um *time*

to send app data equivalente de cerca de 80 ms, uma vez que todas as configurações possuem uma autenticação com custo de tempo inferior a 1 ms. Novamente, apesar de a configuração L1-BIKE-CM possuir um custo de transmissão consideravelmente menor, de 161 bytes, isto não impactou significativamente o *time to send app data*.

Quanto ao nível de segurança 3 (Figura 21), considerando o híbrido HQC na troca de chaves, o KEMTLS-PDK híbrido, com a configuração L3-HQC-CM, teve um tempo de execução levemente menor que TLS pós-quântico híbrido, com a configuração L3-HQC-DIL. A diferença no custo de autenticação justifica isto: L3-HQC-CM custa 1,64 ms e 253 bytes, enquanto que o L3-HQC-DIL 12,47 ms e 3399 bytes. Para as configurações com o híbrido BIKE na troca de chaves, o KEMTLS-PDK híbrido novamente tem um tempo de execução levemente menor, para ambas as configurações L3-BIKE-BIKE e L3-BIKE-CM, que possuem um desempenho equivalente entre si levando em conta o desvio padrão do experimento.

No nível de segurança 5 (Figura 22), o TLS pós-quântico híbrido teve um desempenho superior ao KEMTLS-PDK híbrido em todas as configurações. Para as configurações em que o híbrido HQC é usado na troca de chaves, enquanto as configurações L5-HQC-CM e L5-HQC-DIL tiveram um desempenho próximo, a configuração L5-HQC-HQC obteve um *time to send app data* cerca de 100 ms maior que estas, por conta do seu custo de autenticação de 86,21 ms e 14554 bytes. Já para configurações em que o híbrido BIKE é usado na troca de chaves, a configuração do KEMTLS-PDK híbrido que mais se aproximou do TLS pós-quântico híbrido foi a L5-BIKE-BIKE, com custo de autenticação de 32,09 ms e 5287 bytes, ao invés da configuração L5-BIKE-CM, com custo de 27,49 ms e 341 bytes. Este resultado é justificável quando observamos o custo de tempo similar entre as duas configurações e consideramos a natureza imprevisível de um experimento realista. No entanto, esperava-se que o custo de transmissão consideravelmente inferior do L5-BIKE-CM tivesse um impacto maior no *time to send app data*.

O experimento 2 evidencia que o desempenho do protocolo está fortemente relacionado ao tempo de execução dos algoritmos, enquanto que a quantidade de bytes transmitidos aparenta não ter tanto impacto, uma vez que as configurações que utilizam o híbrido Classic McEliece na autenticação, apesar de possuírem um custo de transmissão baixíssimo, não desempenharam melhor que as configurações que utilizam o híbrido BIKE na autenticação, que possui um custo de tempo similar, mas um custo de transmissão significativamente maior.

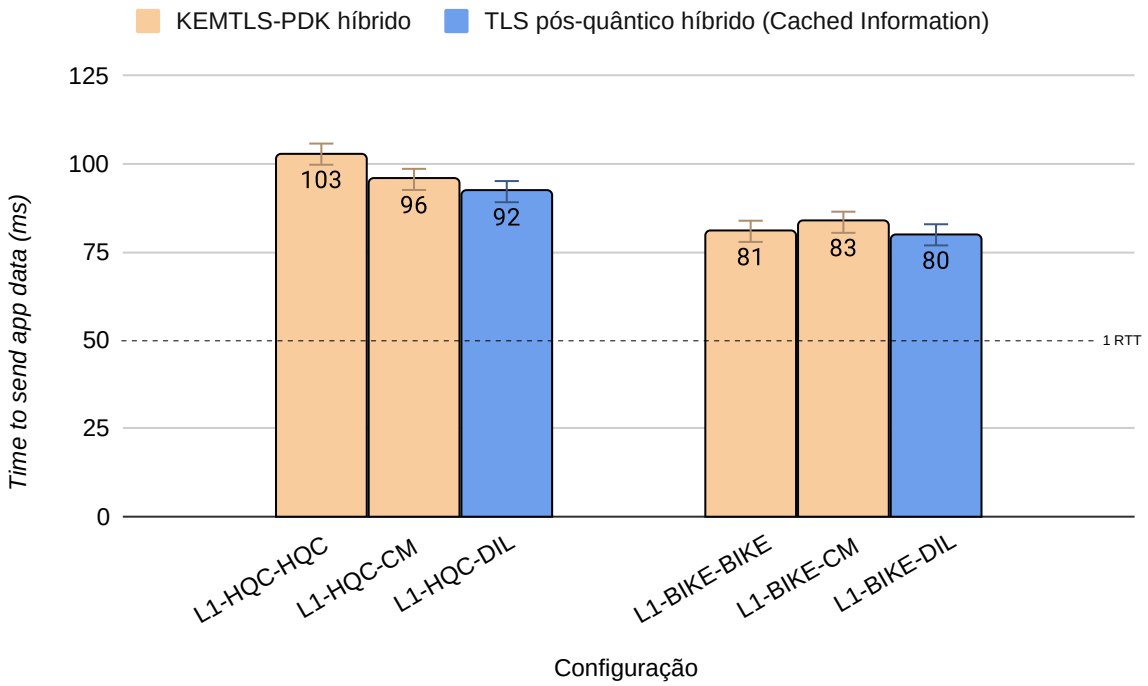


Figura 20 – *Time to send app data* médio do experimento 2 para o nível de segurança 1

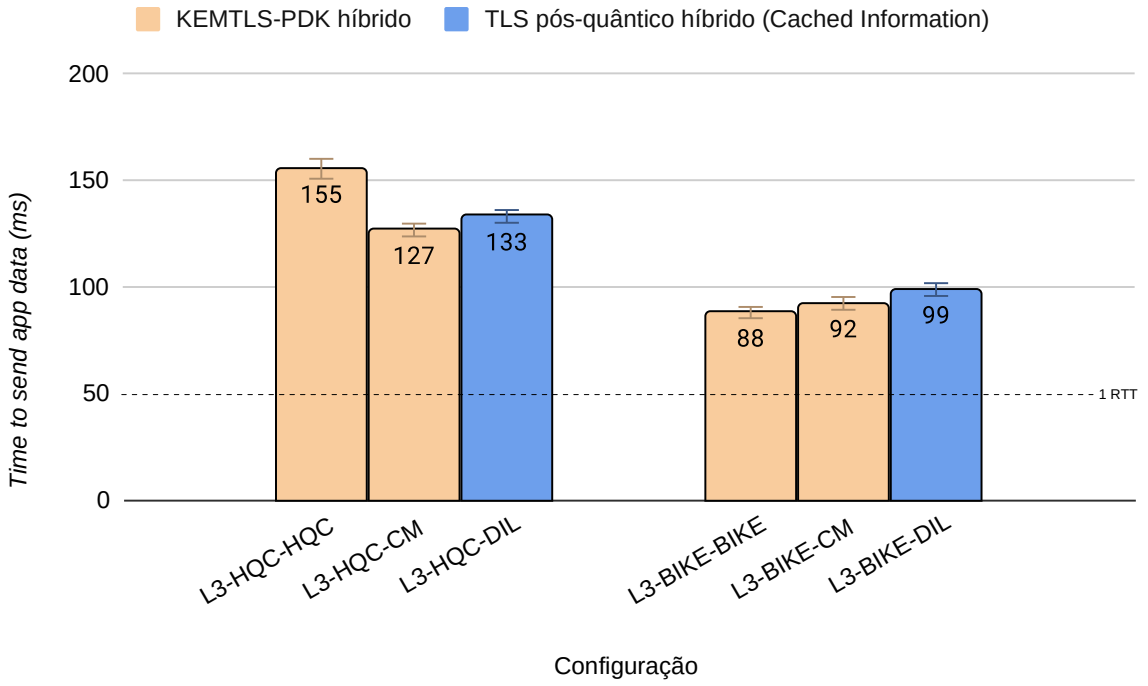


Figura 21 – *Time to send app data* médio do experimento 2 para o nível de segurança 3

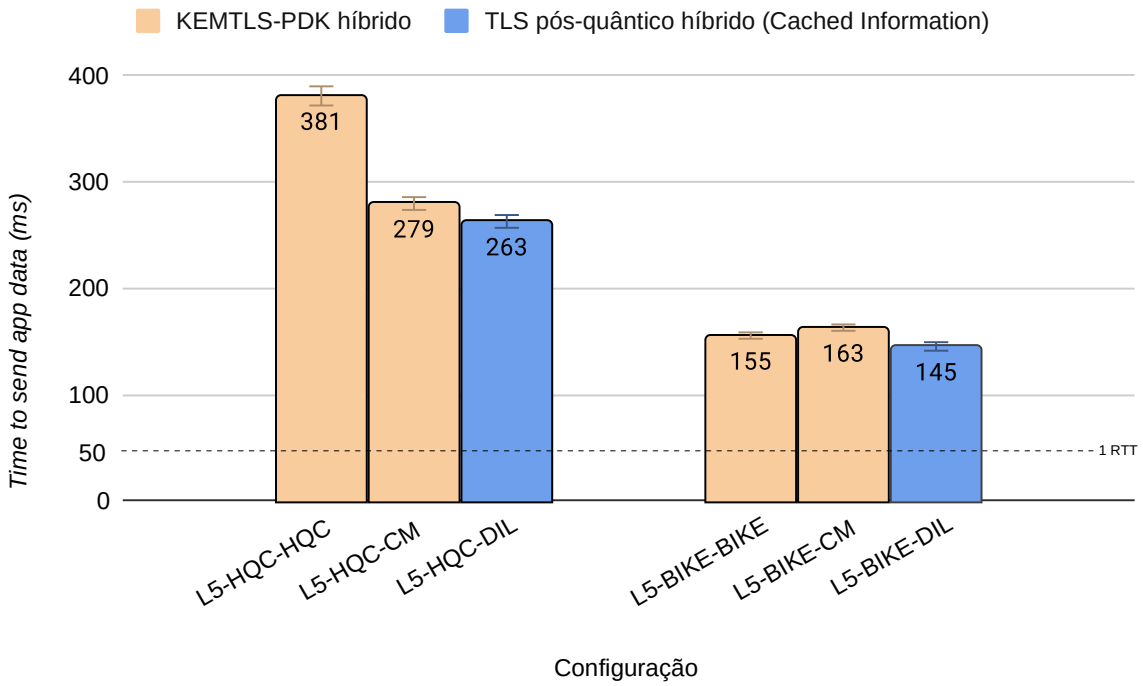


Figura 22 – *Time to send app data* médio do experimento 2 para o nível de segurança 5

4.5.4 Conclusão dos experimentos

A partir da análise dos resultados do experimento 0, considerando uma troca de chaves KEM efêmera em que as operações de geração de chaves, encapsulamento e desencapsulamento são executadas, observa-se que o híbrido BIKE é o algoritmo com melhor desempenho, pois possui o menor tempo de execução e requer a menor quantidade de bytes transmitidos. Para troca de chaves KEM estática, em que apenas as operações de encapsulamento e desencapsulamento são executadas, o híbrido Classic McEliece é o algoritmo com menor tempo de execução. No entanto, por possuir uma chave pública que pode chegar a tamanhos na casa dos *megabytes*, seu uso só é adequado quando há *cache* da chave pública.

Para os experimentos 1 e 2, conclui-se que o algoritmo pós-quântico de KEM candidato à padronização com melhor desempenho para instanciação nos protocolos TLS pós-quântico híbrido e KEMTLS híbrido é o BIKE, dado que as instanciações dos protocolos com este algoritmo obtiveram um *time to send app data* consideravelmente menor em comparação às instanciações com o HQC, especialmente no nível de segurança 5, no qual a diferença foi de aproximadamente 100 ms em ambos os experimentos.

No que diz respeito aos protocolos pós-quânticos híbridos, considerando as instanciações com o algoritmo candidato de melhor desempenho (BIKE), sob o ponto de vista da métrica *time to send app data*, a seguinte listagem atribui o protocolo com

melhor desempenho (*time to send app data* menor) para cada nível de segurança:

- **Nível de segurança 1:**
  - Sem *cache* de certificados: TLS pós-quântico híbrido;
  - Com *cache* de certificados: TLS pós-quântico híbrido e KEMTLS-PDK híbrido (desempenho equivalente).
- **Nível de segurança 3:**
  - Sem *cache* de certificados: KEMTLS híbrido;
  - Com *cache* de certificados: KEMTLS-PDK híbrido.
- **Nível de segurança 5:**
  - Sem *cache* de certificados: KEMTLS híbrido;
  - Com *cache* de certificados: TLS pós-quântico híbrido.

Importante ressaltar que desempenhar melhor, neste contexto, significa ter uma latência menor para o envio de dados de aplicação, e não ter um *handshake* mais rápido.

5 CONSIDERAÇÕES FINAIS

O processo de padronização da criptografia pós-quântica do NIST teve seu primeiro grupo de algoritmos padronizados anunciado em 2022, um algoritmo de KEM e três algoritmos de assinatura. A padronização de um único algoritmo de KEM motivou o anúncio de uma quarta rodada, composta por apenas algoritmos de KEM. Enquanto o algoritmo de KEM padronizado é baseado em problemas matemáticos sobre reticulados (*lattice-based cryptography*), os algoritmos da quarta rodada são baseados em problemas de códigos corretores de erro (*code-based cryptography*). Esta diferença implica em um desempenho consideravelmente diferente entre o algoritmo padronizado e os algoritmos da quarta rodada, o que torna necessária a reavaliação dos protocolos pós-quânticos híbridos instanciados com estes algoritmos.

Buscando avaliar a viabilidade de implantação e o desempenho dos algoritmos da quarta rodada em protocolos pós-quânticos híbridos, este trabalho instanciou o protocolo KEMTLS híbrido com os algoritmos pós-quânticos de KEM candidatos à padronização da quarta rodada, mensurou o desempenho do protocolo KEMTLS híbrido e KEMTLS-PDK híbrido com estes algoritmos em um ambiente realista e comparou-os com o protocolo TLS pós-quântico híbrido sob as mesmas condições e algoritmos.

A avaliação do KEMTLS híbrido com os algoritmos candidatos foi feita em um ambiente realista por meio de uma infraestrutura de *cloud*. Os servidores desta infraestrutura foram escolhidos de modo a terem uma capacidade computacional similar à de um servidor web comum e uma latência de conexão similar à latência observada em conexões do dia-a-dia. A métrica de avaliação utilizada foi a *time to send app data* que mede o tempo decorrido desde o início do *handshake* até o momento em que o cliente pode enviar dados de aplicação ao servidor.

O desempenho dos protocolos pós-quânticos híbridos foi avaliado por meio de dois experimentos: experimento 1, no qual foram avaliadas as versões base dos protocolos, sem *cache* de certificados; e experimento 2, no qual foram avaliadas as versões destes protocolos com *cache* de certificados. Em ambos os experimentos, em todos os níveis de segurança, as configurações com o algoritmo BIKE na troca de chaves e autenticação foram as que obtiveram menor latência, portanto, conclui-se que o BIKE é o algoritmo candidato com melhor desempenho para instanciação no KEMTLS híbrido e no KEMTLS-PDK híbrido.

No que diz respeito à comparação entre o desempenho do KEMTLS híbrido com o TLS pós-quântico híbrido, nenhum protocolo se destacou em relação ao outro, uma vez que ambos apresentaram um *time to send app data* similar, de modo geral. Em cenários em que não há *cache* de certificados, o TLS pós-quântico híbrido é o protocolo com melhor desempenho no nível de segurança 1, enquanto que no nível de segurança 3 e 5, o KEMTLS híbrido possui um desempenho superior. Em cenários onde há *cache*

de certificados, ambos os protocolos possuem desempenho equivalente no nível de segurança 1, no nível de segurança 3 o KEMTLS-PDK supera o desempenho do TLS pós-quântico híbrido, e no nível de segurança 5 é o TLS pós-quântico híbrido quem desempenha melhor.

Além destes resultados, os experimentos realizados evidenciaram que o tempo de execução de um algoritmo de KEM afeta significativamente o desempenho do protocolo KEMTLS híbrido, enquanto que o tamanho dos objetos criptográficos transmitidos (chave pública e texto cifrado) não possui um impacto significativo.

5.1 TRABALHOS FUTUROS

O presente trabalho avaliou o desempenho do KEMTLS híbrido com algoritmos candidatos sob o ponto de vista do cliente em um cenário tradicional de navegação na *web*, utilizando a métrica *time to send app data*, máquinas virtuais com configurações de servidores *web* de médio porte, sem autenticação de cliente, e considerando conexões com baixa perda de pacotes e latência relativamente baixa. Desta forma, para trabalhos futuros, indica-se avaliar este protocolo com os algoritmos candidatos em outros cenários que também precisarão transicionar para a criptografia pós-quântica.

Uma possível continuação deste trabalho seria avaliar o protocolo com os algoritmos candidatos sob o ponto de vista do servidor. Neste contexto, é possível realizar experimentos de teste de carga HTTP do servidor, nos quais múltiplos clientes concorrentemente enviam requisições HTTP a um servidor *web* que deve atendê-las, servindo algum conteúdo como uma página *web*. Com estes experimentos, é possível avaliar a capacidade de um servidor *web* de operar sob alta demanda. Neste cenário, aspectos como a quantidade de memória RAM utilizada pelos algoritmos e a quantidade de bytes transmitidos na rede possuem um impacto significativo no desempenho do servidor, portanto, resultados diferentes dos obtidos neste trabalho podem ser encontrados.

Outro cenário que não foi avaliado foi KEMTLS híbrido (instanciado com algoritmos candidatos) com autenticação mútua, com e sem *cache* de certificados. Por se tratar de um cenário que contém um fluxo adicional de autenticação, algoritmos candidatos com desempenho excepcional na autenticação (tal como o Classic McEliece quando há cache de certificados) podem se destacar.

Por fim, outro possível cenário para avaliação são redes IoT. Estas redes são compostas por dispositivos de baixo poder computacional conectados geralmente por redes sem fio, caracterizadas por uma latência maior e uma alta probabilidade de perda de pacotes. Este cenário possui requisitos diferentes de desempenho do que do cenário *web* que foi avaliado no presente trabalho, e, portanto, requer uma avaliação distinta, com outras métricas e abordagens.

REFERÊNCIAS

ARUTE, Frank *et al.* Quantum Supremacy using a Programmable Superconducting Processor. **Nature**, v. 574, p. 505–510, 2019. Disponível em: <https://www.nature.com/articles/s41586-019-1666-5>. Citado na p. 17.

AWS. **O que é computação quântica?** [*S.l.*: *s.n.*]. https://aws.amazon.com/pt/what-is/quantum-computing/. Acessado em 16/03/2024. Citado nas pp. 16, 29, 30.

BARNES, Richard *et al.* **Hybrid Public Key Encryption**. [*S.l.*]: RFC Editor, fev. 2022. RFC 9180. (Request for Comments, 9180). DOI: 10.17487/RFC9180. Disponível em: <https://www.rfc-editor.org/info/rfc9180>. Citado na p. 51.

BERLEKAMP, E. Goppa codes. **IEEE Transactions on Information Theory**, v. 19, n. 5, p. 590–592, 1973. DOI: 10.1109/TIT.1973.1055088. Citado na p. 33.

BERNSTEIN, Daniel J. Introduction to post-quantum cryptography. *In:* **Post-Quantum Cryptography**. Edição: Daniel J. Bernstein, Johannes Buchmann e Erik Dahmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. P. 1–14. ISBN 978-3-540-88702-7. DOI: 10.1007/978-3-540-88702-7_1. Disponível em: <https://doi.org/10.1007/978-3-540-88702-7_1>. Citado na p. 33.

BEULLENS, Ward. Breaking Rainbow Takes a Weekend on a Laptop. *In:* DODIS, Yevgeniy; SHRIMPTON, Thomas (Ed.). **Advances in Cryptology – CRYPTO 2022**. Cham: Springer Nature Switzerland, 2022. P. 464–479. Citado nas pp. 18, 33, 34, 50.

BEVERIDGE, Iain; BUTCHER, Dave. **Harvest Now, Decrypt Later – Fact or Fiction?** [*S.l.*: *s.n.*], 2023. https://www.entrust.com/blog/2023/11/harvest-now-decrypt-later-fact-or-fiction/. Acessado em 17/03/2024. Citado nas pp. 19, 31.

BINDEL, Nina *et al.* Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange. *In:* DING, Jintai; STEINWANDT, Rainer (Ed.). **Post-Quantum Cryptography**. Cham: Springer International Publishing, 2019. P. 206–226. Citado nas pp. 35, 38, 54, 57.

BOEYEN, Sharon *et al.* **Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**. [*S.l.*]: RFC Editor, mai. 2008. RFC 5280.

(Request for Comments, 5280). DOI: 10.17487/RFC5280. Disponível em: <https://www.rfc-editor.org/info/rfc5280>. Citado na p. 22.

CASTRYCK, Wouter; DECRU, Thomas. An Efficient Key Recovery Attack on SIDH. *In:* HAZAY, Carmit; STAM, Martijn (Ed.). **Advances in Cryptology – EUROCRYPT 2023**. Cham: Springer Nature Switzerland, 2023. P. 423–447. Citado nas pp. 18, 33, 34, 50.

CELI, Sofía *et al.* Implementing and Measuring KEMTLS. *In:* LONGA, Patrick; RÀFOLS, Carla (Ed.). **Progress in Cryptology – LATINCRYPT 2021**. Cham: Springer International Publishing, 2021. (Lecture Notes in Computer Science), p. 88–107. DOI: 10.1007/978-3-030-88238-9_5. Disponível em: <https://kemtls.org/publication/measuring-kemtls/>. Citado nas pp. 59, 63.

CLOUDFLARE. **CIRCL - Cloudflare Interoperable Reusable Cryptographic Library**. [*S.l.*: *s.n.*]. https://github.com/cloudflare/circl. Acessado em 18/06/2024. Citado na p. 59.

COUTINHO, Thiago. **Computação Quântica: tecnologia que opera em máquinas extremamente complexas!** [*S.l.*: *s.n.*], 2023. https://www.voitto.com.br/blog/artigo/computacao-quantica. Acessado em 17/03/2024. Citado na p. 30.

CROCKETT, Eric; PAQUIN, Christian; STEBILA, Douglas. **Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH**. [*S.l.*: *s.n.*], 2019. Cryptology ePrint Archive, Paper 2019/858. https://eprint.iacr.org/2019/858. Disponível em: <https://eprint.iacr.org/2019/858>. Citado nas pp. 18, 35, 36, 38.

DIFFIE, Whitfield; HELLMAN, Martin E. New Directions in Cryptography. **IEEE Transactions on Information Theory**, v. 22, n. 6, p. 644–654, 1976. Citado nas pp. 21, 28.

DOCS, MDN Web. **Evolution of HTTP**. [*S.l.*: *s.n.*]. https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP. Acessado em 18/03/2024. Citado na p. 16.

FIELDING, Roy T.; NOTTINGHAM, Mark; RESCHKE, Julian. **HTTP Semantics**. [*S.l.*]: RFC Editor, jun. 2022. RFC 9110. (Request for Comments, 9110). DOI: 10.17487/RFC9110. Disponível em: <https://www.rfc-editor.org/info/rfc9110>. Citado nas pp. 16, 21.

GIRON, Alexandre Augusto *et al.* Post-quantum Hybrid KEMTLS Performance in Simulated and Real Network Environments. *In:* ALY, Abdelrahaman; TIBOUCHI, Mehdi (Ed.). **Progress in Cryptology – LATINCRYPT 2023**. Cham: Springer Nature Switzerland, 2023. P. 293–312. Citado nas pp. 19, 50, 56.

GROVER, Lov K. A fast quantum mechanical algorithm for database search. *In:* PROCEEDINGS of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996. (STOC '96), p. 212–219. DOI: 10.1145/237814.237866. Disponível em: <https://doi.org/10.1145/237814.237866>. Citado na p. 17.

HELERBROCK, Rafael. **O que é computação quântica?** [*S.l.*: *s.n.*]. https://brasilescola.uol.com.br/o-que-e/fisica/o-que-e-computacao-quantica.htm. Acessado em 16/03/2024. Citado na p. 29.

IBM. **Session resumption with a pre-shared key**. [*S.l.*: *s.n.*]. https://www.ibm.com/docs/en/sdk-java-technology/8?topic=handshake-session-resumption-pre-shared-key. Acessado em 19/05/2024. Citado na p. 24.

KRAWCZYK, Dr. Hugo; BELLARE, Mihir; CANETTI, Ran. **HMAC: Keyed-Hashing for Message Authentication**. [*S.l.*]: RFC Editor, fev. 1997. RFC 2104. (Request for Comments, 2104). DOI: 10.17487/RFC2104. Disponível em: <https://www.rfc-editor.org/info/rfc2104>. Citado na p. 23.

KRAWCZYK, Dr. Hugo; ERONEN, Pasi. **HMAC-based Extract-and-Expand Key Derivation Function (HKDF)**. [*S.l.*]: RFC Editor, mai. 2010. RFC 5869. (Request for Comments, 5869). DOI: 10.17487/RFC5869. Disponível em: <https://www.rfc-editor.org/info/rfc5869>. Citado na p. 24.

LAING, Thalia; CHARLES, Tommy. **Anticipating the Quantum Threat to Cryptography**. [*S.l.*: *s.n.*], 2024. https://threatresearch.ext.hp.com/anticipating-the-quantum-threat-to-cryptography/. Acessado em 16/02/2024. Citado na p. 17.

MICCIANCIO, Daniele. Shortest Vector Problem. *In:* **Encyclopedia of Cryptography and Security**. Edição: Henk C. A. van Tilborg. Boston, MA: Springer US, 2005. P. 569–570. ISBN 978-0-387-23483-0. DOI: 10.1007/0-387-23483-7_392. Disponível em: <https://doi.org/10.1007/0-387-23483-7_392>. Citado na p. 33.

MIT TECHNOLOGY REVIEW. **O que é criptografia pós-quântica?** [*S.l.*: *s.n.*]. https://mittechreview.com.br/o-que-e-criptografia-pos-quantica/. Acessado em 17/03/2024. Citado na p. 32.

MOODY, Dustin. **Let's Get Ready to Rumble- The NIST PQC Competition**. [*S.l.*: *s.n.*], 2018. https://csrc.nist.gov/CSRC/media/Presentations/Let-s-Get-Ready-to-Rumble-The-NIST-PQC-Competiti/images-media/PQCrypto-April2018_Moody.pdf. Citado na p. 59.

MOSCA, Dr. Michele; PIANI, Dr. Marco. **2023 Quantum Threat Timeline Report**. [*S.l.*], 2023. Disponível em: <https://globalriskinstitute.org/publication/2023-quantum-threat-timeline-report/>. Citado nas pp. 19, 31.

NIST. **Post-Quantum Cryptography**: Overview. [*S.l.*: *s.n.*], 2024. https://csrc.nist.gov/projects/post-quantum-cryptography. Acessado em 16/02/2024. Citado nas pp. 17, 32, 33.

OLIVEIRA, Danilo; IGNACIO DE LIMA, Bruno. **O que é e como funciona o Emanharamento Quântico?** [*S.l.*: *s.n.*], 2023. https://olhardigital.com.br/2023/10/24/ciencia-e-espaco/o-que-e-e-como-funciona-o-emanharamento-quantico/. Acessado em 16/03/2024. Citado na p. 30.

PROJECT, Open Quantum Safe. **liboqs**. [*S.l.*: *s.n.*]. https://github.com/open-quantum-safe/liboqs. Acessado em 18/06/2024. Citado na p. 59.

PROJECT, Open Quantum Safe. **liboqs-go: Go bindings for liboqs**. [*S.l.*: *s.n.*]. https://github.com/open-quantum-safe/liboqs-go. Acessado em 18/06/2024. Citado na p. 59.

RESCORLA, Eric. **The Transport Layer Security (TLS) Protocol Version 1.3**. [*S.l.*]: RFC Editor, ago. 2018. RFC 8446. (Request for Comments, 8446). DOI: 10.17487/RFC8446. Disponível em: <https://www.rfc-editor.org/info/rfc8446>. Citado nas pp. 16, 21, 26, 36.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 21, n. 2, p. 120–126, fev. 1978. Acessado em 17/03/2024. ISSN 0001-0782. DOI: 10.1145/359340.359342. Disponível em: <https://doi.org/10.1145/359340.359342>. Citado nas pp. 18, 31.

SANTESSON, Stefan; TSCHOFENIG, Hannes. **Transport Layer Security (TLS) Cached Information Extension**. [*S.l.*]: RFC Editor, jul. 2016. RFC 7924. (Request for Comments, 7924). DOI: 10.17487/RFC7924. Disponível em: <https://www.rfc-editor.org/info/rfc7924>. Citado na p. 27.

SCHWABE, Peter; STEBILA, Douglas; WIGGERS, Thom. More efficient post-quantum KEMTLS with pre-distributed public keys. *In:* BERTINO, Elisa; SHULMAN, Haya; WAIDNER, Michael (Ed.). **Computer Security – ESORICS 2021**. Cham: Springer International Publishing, set. 2021. (Lecture Notes in Computer Science), p. 3–22. DOI: 10.1007/978-3-030-88418-5_1. Disponível em: <https://kemtls.org/publication/kemtlspdk/>. Citado na p. 46.

SCHWABE, Peter; STEBILA, Douglas; WIGGERS, Thom. Post-Quantum TLS Without Handshake Signatures. *In:* PROCEEDINGS of the 2020 ACM SIGSAC Conference on Computer and Communications Security. Virtual Event, USA: Association for Computing Machinery, 2020. (CCS '20), p. 1461–1480. DOI: 10.1145/3372297.3423350. Disponível em: <https://doi.org/10.1145/3372297.3423350>. Citado nas pp. 18, 35, 41.

SHOR, P.W. Algorithms for quantum computation: discrete logarithms and factoring. *In:* PROCEEDINGS 35th Annual Symposium on Foundations of Computer Science. [*S.l.*: *s.n.*], 1994. P. 124–134. DOI: 10.1109/SFCS.1994.365700. Citado nas pp. 17, 28, 31, 32.

SIKERIDIS, Dimitrios; KAMPANAKIS, Panos; DEVETSIKIOTIS, Michael. Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH. *In:* PROCEEDINGS of the 16th International Conference on Emerging Networking EXperiments and Technologies. Barcelona, Spain: Association for Computing Machinery, 2020. (CoNEXT '20), p. 149–156. DOI: 10.1145/3386367.3431305. Disponível em: <https://doi.org/10.1145/3386367.3431305>. Citado na p. 18.

STALLINGS, William. **Cryptography and Network Security: Principles and Practice**. 6th. USA: Prentice Hall Press, 2013. ISBN 0133354695. Citado nas pp. 16, 21.

STANDARDS, National Institute of; TECHNOLOGY. **Digital Identity Guidelines**. [*S.l.*], 2020. DOI: 10.6028/NIST.SP.800-63-3. Citado na p. 26.

STANDARDS, National Institute of; TECHNOLOGY. **Digital Signature Standard (DSS)**. [*S.l.*], 2023. DOI: 10.6028/NIST.FIPS.186-5. Citado nas pp. 59, 60.

STEBILA, Douglas; FLUHRER, Scott; GUERON, Shay. **Hybrid key exchange in TLS 1.3**. [*S.l.*], set. 2023. Work in Progress. Disponível em: <https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/09/>. Citado nas pp. 34, 35, 38, 52.

STEBILA, Douglas; MOSCA, Michele. Post-quantum Key Exchange for the Internet and the Open Quantum Safe Project. *In:* AVANZI, Roberto; HEYS, Howard (Ed.). **Selected Areas in Cryptography – SAC 2016**. Cham: Springer International Publishing, 2017. P. 14–37. Citado na p. 59.

TAMVADA, Goutam; CELI, Sofía. **Deep dive into a post-quantum key encapsulation algorithm**. [*S.l.*: *s.n.*], 2022. https://blog.cloudflare.com/post-quantum-key-encapsulation. Acessado em 16/03/2024. Citado na p. 28.

WIKIPEDIA CONTRIBUTORS. **NIST Post-Quantum Cryptography Standardization — Wikipedia, The Free Encyclopedia**. [*S.l.*: *s.n.*], 2024. https://en.wikipedia.org/w/index.php?title=NIST_Post-Quantum_Cryptography_Standardization&oldid=1212758724. [Acessado em 17/03/2024]. Citado na p. 34.

WILTON, Robin. **Fact Sheet: Quantum Physics and Computing**: Does quantum computing put our digital security at risk? [*S.l.*: *s.n.*], 2020. https://www.internetsociety.org/resources/doc/2020/does-quantum-computing-put-our-digital-security-at-risk. Acessado em 16/02/2024. Citado na p. 17.

Apêndices

APÊNDICE A – CÓDIGO FONTE

O código fonte está disponível nos seguintes repositórios:
https://github.com/JPADN/go-hybrid-kemtls
https://github.com/JPADN/hybrid_kemtls_tests

APÊNDICE B – ARTIGO SBC

Desempenho de algoritmos pós-quânticos candidatos à padronização no KEMTLS híbrido

João Pedro Adami do Nascimento

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brazil

Abstract. *Confidentiality, integrity, and authenticity in internet communications are guaranteed by the HTTPS protocol, which uses TLS to add a layer of encryption. However, quantum computing threatens traditional asymmetric algorithms, motivating the development of post-quantum cryptography. Since 2016, NIST has been leading the standardization of these algorithms, currently in the fourth round. Due to the immaturity of these new algorithms, hybrid versions that combine classical and post-quantum cryptography, have been adopted. This work analyzes the performance of post-quantum algorithms in the Hybrid KEMTLS, comparing it to the Hybrid Post-quantum TLS. The results show that BIKE is the algorithm with the best performance.*

Resumo. *A confidencialidade, integridade e autenticidade nas comunicações na internet são garantidas pelo protocolo HTTPS, que usa TLS para adicionar uma camada de criptografia. Entretanto, a computação quântica ameaça algoritmos assimétricos tradicionais, motivando o desenvolvimento da criptografia pós-quântica. Desde 2016, o NIST lidera a padronização desses algoritmos, atualmente na quarta rodada. Devido à imaturidade destes novos algoritmos, versões híbridas, que combinam criptografia clássica e pós-quântica, foram adotadas. Este trabalho analisa o desempenho de algoritmos pós-quânticos no KEMTLS híbrido, comparando-o ao TLS pós-quântico híbrido. Os resultados mostram que o BIKE é o algoritmo com melhor desempenho.*

1. Contextualização

A internet, desde seu início, esteve estreitamente ligada ao protocolo *Hypertext Transfer Protocol* (HTTP) [Fielding et al. 2022], um protocolo de comunicação utilizado para a transferência de dados na web. Com o crescimento da internet e o aumento de ataques cibernéticos e roubo de informações privadas, tornou-se evidente a necessidade de uma camada de transporte segura para todas as aplicações. Nesse contexto, surgiu o HTTPS, uma extensão do HTTP que garante a confidencialidade e integridade dos dados transmitidos, além da autenticação das partes envolvidas na conexão. Para isso, é utilizado o protocolo criptográfico TLS (*Transport Layer Security*) [Rescorla 2018].

No TLS, de modo geral, são utilizadas duas classes de criptografia para garantir a confidencialidade e a autenticidade da comunicação: a criptografia simétrica e a criptografia assimétrica.

Nos últimos anos, o interesse e os investimentos em computação quântica aumentaram significativamente, tanto na pesquisa acadêmica quanto na indústria. Recentemente, empresas como a Google anunciaram ter alcançado a supremacia quântica

[Arute et al. 2019], isto é, a capacidade de um computador quântico resolver um problema específico mais rapidamente do que qualquer computador clássico. Esse marco abre caminho para grandes avanços, permitindo o desenvolvimento de aplicações computacionais que anteriormente eram consideradas impraticáveis.

No entanto, a capacidade dos computadores quânticos de resolver determinados problemas computacionais de forma eficiente representa uma ameaça à criptografia contemporânea [Wilton 2020], especialmente a criptografia assimétrica. Os algoritmos de criptografia assimétrica são baseados em problemas matemáticos, como o logaritmo discreto e a fatoração de inteiros, cuja segurança depende da dificuldade que computadores clássicos enfrentam para resolvê-los de forma eficiente. No entanto, a computação quântica, por meio do algoritmo de Shor [Shor 1994], é capaz de resolver esses problemas com complexidade assintótica polinomial, comprometendo assim a segurança criptográfica desses algoritmos.

Como resposta à ameaça dos computadores quânticos, surgiu a criptografia pós-quântica, uma criptografia baseada em problemas matemáticos que não possuem solução eficiente conhecida nem em computadores clássicos e nem em computadores quânticos. De modo à padronizar os algoritmos pós-quânticos para uso no mundo real, o *National Institute of Standards and Technology* (NIST) anunciou o processo de padronização de criptografia pós-quântica [NIST 2024] em 2016. Atualmente este processa encontra-se na quarta rodada, onde estão sendo avaliados algoritmos de KEM candidatos à padronização.

Algoritmos de criptografia pós-quântica são desenvolvimentos recentes na história da criptografia, portanto ainda não foi estabelecida a confiança necessária para substituir os algoritmos utilizados hoje em dia por algoritmos pós-quânticos. Nesse contexto, surge a criptografia pós-quântica híbrida, que combina algoritmos clássicos e pós-quânticos para garantir segurança enquanto ao menos um dos componentes (clássico ou pós-quântico) permanecer seguro.

A adoção da criptografia pós-quântica no TLS foi realizada por meio de duas propostas, principalmente: o TLS pós-quântico [Crockett et al. 2019] e o KEM-TLS [Schwabe et al. 2020]. De modo À incorporar a criptografia pós-quântica híbrida, extensões destes protocolos foram desenvolvidas: o TLS pós-quântico híbrido [Crockett et al. 2019] e o KEMTLS híbrido [Schwabe et al. 2020].

Atualmente, os computadores quânticos não representam uma ameaça ativa à criptografia, mas já são motivo de grande preocupação. A migração da infraestrutura da internet e de outras redes de computadores para a criptografia pós-quântica é extremamente complexa e onerosa, e deve ser iniciada o quanto antes, mesmo que computadores quânticos suficientemente poderosos possam surgir apenas nas próximas décadas, segundo especialistas [Mosca and Piani 2023]. Além disso, computadores quânticos apresentam uma ameaça passiva por meio de ataques retroativos, como o *store now, decrypt later* [Beveridge and Butcher 2023].

Diante desse cenário, a prototipação e experimentação da criptografia pós-quântica em protocolos de comunicação segura emergem como ferramentas essenciais para coletar dados que orientem de forma mais eficaz a migração da infraestrutura atual para uma infraestrutura pós-quântica. Esse processo permite identificar os principais desafios e avaliar o impacto dessa migração nas comunicações seguras.

Na literatura, o protocolo KEMTLS híbrido foi implementado e avaliado utilizando os algoritmos finalistas do processo de padronização do NIST. Considerando que, entre os algoritmos de KEM finalistas, apenas um foi padronizado, torna-se necessário avaliar esse protocolo com os novos algoritmos de KEM candidatos à padronização, ou seja, os algoritmos da quarta rodada. Assim, este trabalho tem como objetivo instanciar os algoritmos candidatos à padronização no KEMTLS híbrido e avaliar seu desempenho por meio de experimentos comparativos com o protocolo TLS pós-quântico híbrido.

1.1. TLS

O *Transport Layer Security* (TLS) é um protocolo criptográfico amplamente utilizado para garantir a confidencialidade, a autenticação e a integridade dos dados transmitidos em conexões na internet [Rescorla 2018]. O TLS é composto por duas camadas principais: o *TLS Record Protocol* e o *TLS Handshake Protocol*. O *TLS Record Protocol* opera na camada de transporte e tem como principal função criptografar os dados transmitidos na conexão utilizando algoritmos de criptografia simétrica. Já o TLS *Handshake Protocol*, é responsável pela negociação de parâmetros de segurança entre o cliente e o servidor, estabelecimento das chaves de criptografia do protocolo e autenticação das partes comunicantes.

A fim de estabelecer um segredo para gerar as chaves de criptografia utilizadas para criptografar os pacotes TLS, o handshake executa uma troca de chaves Diffie-Hellman [Diffie and Hellman 1976]. O segredo obtido por este método é chamado de segredo compartilhado e este é utilizado para derivar as chaves de criptografia simétrica do protocolo por meio de um processo de derivação de chaves chamado *key schedule*. Já para realizar a autenticação das entidades do protocolo são utilizados certificados digitais X.509 [Boeyen et al. 2008] e assinaturas digitais feitas com estes, provando desta forma que a entidade tem posse do certificado que apresenta.

Dessa forma, a confidencialidade estabelecida pelo TLS, e a autenticação das partes comunicantes, depende da segurança do algoritmo de troca de chaves Diffie-Hellman e do algoritmo de assinatura digital, respectivamente, ambos algoritmos de criptografia assimétrica.

1.2. TLS 1.3 Cached Information

No TLS, o servidor precisa enviar o seu certificado ao cliente durante o *handshake* para iniciar o processo de autenticação. A fim de prover suporte para cenários onde o cliente já possui o certificado do servidor previamente, foi criada a extensão *Cached Information* [Santesson and Tschofenig 2016]. Por meio desta extensão, clientes TLS informam ao servidor que já possuem o certificado deste, não sendo necessário o envio do mesmo durante o *handshake*.

1.3. Ameaça do computador quântico

Apesar de o desenvolvimento do computador quântico ser um grande passo no desenvolvimento científico, a vantagem computacional deste computador traz grandes implicações para determinados sistemas de criptografia, principalmente a criptografia assimétrica. A segurança da criptografia assimétrica é baseada na dificuldade de encontrar soluções para determinados problemas matemáticos, como o problema do logaritmo discreto e o problema da fatoração de inteiros.

Atualmente, em computadores clássicos, resolver o problema do logaritmo discreto e a fatoração de inteiros para números grandes é computacionalmente desafiador, uma vez que os algoritmos conhecidos para estes problemas consomem tempo exponencial de execução em relação aos parâmetros utilizados no problema. No entanto, há um algoritmo quântico (isto é, que executa em um computador quântico), capaz de resolver estes problemas de forma eficiente, com uma complexidade assintótica polinomial em relação aos parâmetros utilizados no problema. Este algoritmo quântico é o algoritmo de Shor [Shor 1994], e com ele é possível fatorar números inteiros compostos grandes de forma eficiente.

O algoritmo de Shor ainda não representa uma ameaça ativa para a criptografia assimétrica clássica, tendo em vista que ainda não foi desenvolvido um computador quântico criptograficamente relevante (CRQC), isto é, um computador quântico com desempenho suficiente para quebrar algoritmos de criptografia. No entanto, o algoritmo de Shor representa uma ameaça passiva para a segurança das comunicações da atualidade por meio do ataque *store now, decrypt later* (também conhecido como *harvest now, decrypt later*) [Beveridge and Butcher 2023]. Este ataque, como o nome sugere, é baseado na captura de pacotes de dados transmitidos na rede atualmente para decifragem no futuro, quando um computador quântico eficiente estiver disponível e o algoritmo de Shor puder ser executado. Dessa forma, a confidencialidade das comunicações de hoje em dia poderá ser violada no futuro.

Frente à imprevisibilidade do advento do CRQC e à ameaça do ataque retroativo *store now, decrypt later*, diversos esforços surgiram a fim de desenvolver uma criptografia à prova do computador quântico, sendo o mais notável destes o processo de padronização de criptografia pós-quântica do NIST [NIST 2024], anunciado em 2016.

1.4. Criptografia pós-quântica

A criptografia pós-quântica é uma criptografia desenvolvida para ser implementada em computadores clássicos com o objetivo de ser resistente contra ataques de computadores quânticos [MIT Technology Review]. Além disto, a criptografia pós-quântica deve ser resistente a computadores clássicos também, proporcionando uma segurança igual ou superior à criptografia clássica.

Uma vez que os problemas matemáticos nos quais a criptografia clássica é baseada possuem solução eficiente em um computador quântico por meio do algoritmo de Shor [Shor 1994], algoritmos de criptografia pós-quântica exploram outros problemas matemáticos, que não possuem solução eficiente conhecida em um computador quântico e nem em um computador clássico.

Em 2016, foi anunciado o processo de padronização da criptografia pós-quântica do NIST [NIST 2024], uma competição organizada pelo *National Institute of Standards and Technology* (NIST) a fim de avaliar e padronizar algoritmos de criptografia resistentes a computadores quânticos para uso em aplicações do mundo real. Este processo aceita submissões de algoritmos pós-quânticos de duas categorias: algoritmos de assinatura digital e algoritmos de KEM, utilizados para troca de chaves em protocolos criptográficos. A competição foi organizada na forma de rodadas, de modo que em cada rodada os algoritmos são submetidos a extensivos testes e criptoanálise, e somente os algoritmos suficientemente robustos avançam para a próxima rodada. Com o avanço

das rodadas e estudos intensivos de criptoanálise, diversos algoritmos foram descartados do processo devido ao descobrimento de vulnerabilidades e possíveis vetores de ataques [Castryck and Decru 2023, Beullens 2022]. Destes casos, destaca-se o do algoritmo de assinatura Rainbow, que teve uma vulnerabilidade descoberta na última rodada do processo [Beullens 2022].

O fim da terceira rodada da competição foi marcado pelo anúncio dos algoritmos vencedores: na categoria KEM, o Kyber; na categoria de assinatura digital, CRYSTALS-Dilithium, Falcon e SPHINCS+. Contudo, o anúncio dos algoritmos vencedores não marcou o fim da competição, uma vez que o NIST anunciou uma quarta rodada[1] para algoritmos de KEM, e uma nova competição[2] para algoritmos de assinatura. A quarta rodada busca padronizar novos algoritmos de KEM, dado que apenas um algoritmo de KEM foi padronizado ao fim da competição. Inicialmente, esta rodada era composta pelos algoritmos BIKE, Classic McEliece, HQC e SIKE, no entanto, o algoritmo SIKE foi desqualificado após a publicação de um ataque [Castryck and Decru 2023].

1.5. Criptografia pós-quântica híbrida

Algoritmos de criptografia pós-quântica são desenvolvimentos recentes na história da criptografia, e apesar de serem submetidos à rigorosos testes e revisões no processo de padronização do NIST, ainda há margem para o descobrimento de vulnerabilidades, como foi observado ao longo das rodadas do processo de padronização, nas quais dezenas de ataques nos algoritmos candidatos foram publicados [Wikipedia contributors 2024]. Algoritmos de criptografia pós-quântica ainda não atingiram a maturidade que algoritmos clássicos já possuem, tendo em vista que a sua aplicação no mundo real ainda é limitada e muito recente. Esta imaturidade provoca uma insegurança acerca da substituição dos algoritmos de criptografia clássica pelos algoritmos pós-quânticos, motivando o uso da criptografia pós-quântica híbrida.

A criptografia pós-quântica híbrida refere-se à abordagem de utilizar algoritmos criptográficos clássicos em conjunto com algoritmos pós-quânticos, visando combinar a segurança e robustez proporcionada pelos algoritmos clássicos com a resistência aos ataques quânticos oferecida pelos algoritmos pós-quânticos. O uso de criptografia pós-quântica híbrida permite uma transição gradual para a criptografia pós-quântica, diminuindo os riscos associados à transição para novos algoritmos criptográficos.

Na criptografia pós-quântica híbrida, os dois algoritmos são executados em paralelo, e as saídas produzidas por esses algoritmos são combinadas por meio de um algoritmo combinador. Este algoritmo combinador deve ser capaz de garantir a ”propriedade híbrida”, definida em [Stebila et al. 2023].

A criptografia pós-quântica híbrida pode ser utilizada em dois contextos: assinaturas digitais e troca de chaves. Um algoritmo de assinatura digital híbrido é composto por dois algoritmos de assinatura, um clássico e um pós-quântico, de tal forma que a chave (pública ou privada) corresponde à concatenação da chave do algoritmo clássico com a chave do algoritmo pós-quântico. Em um algoritmo de assinatura híbrido, os dois algoritmos de assinatura são executados com a mesma entrada (dado a ser assinado), e

[1] `https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions`

[2] `https://csrc.nist.gov/Projects/pqc-dig-sig/standardization`

as assinaturas resultantes são concatenadas, obtendo dessa forma uma assinatura híbrida [Crockett et al. 2019]. Dessa forma, para que um atacante possa forjar uma assinatura híbrida, este precisa forjar a assinatura com o algoritmo clássico e com o algoritmo pós-quântico, garantindo a propriedade híbrida.

Um algoritmo de troca de chaves híbrido é composto por um algoritmo de troca de chaves clássico e um algoritmo de troca de chaves pós-quântico [Stebila et al. 2023]. O algoritmo pós-quântico é um KEM e o algoritmo clássico é o Diffie-Hellman, podendo este ser instanciado na forma de um KEM. A computação de um segredo compartilhado híbrido se dá por meio da execução paralela de ambos os algoritmos de troca de chaves, e consequente combinação dos segredos compartilhados obtidos de cada algoritmo. Para instanciações de algoritmos de troca de chaves híbridos no TLS, o algoritmo combinador utilizado é o *dual-PRF* [Bindel et al. 2019], um algoritmo combinador baseado no *key schedule* do TLS . Por meio deste combinador, a propriedade híbrida é garantida.

1.6. Adoção da criptografia pós-quântica no TLS

De modo geral, a criptografia pós-quântica foi incorporada ao TLS de duas formas: TLS pós-quântico e o KEMTLS. O TLS pós-quântico, prototipado em [Crockett et al. 2019], busca adicionar a criptografia pós-quântica no protocolo sem alterar a estrutura e funcionamento deste, utilizando as mesmas mensagens e extensões, sendo necessária apenas a ressignificação de algumas extensões a fim de adequá-las aos algoritmos pós-quânticos. No TLS pós-quântico, a troca de chaves de Diffie-Hellman é substituída pela troca de chaves KEM com algoritmos de KEM pós-quânticos, e a autenticação com assinaturas digitais é adaptada para suportar algoritmos de assinatura pós-quânticos.

O KEMTLS, proposto em [Schwabe et al. 2020], propõe uma versão alternativa do protocolo de *handshake* do TLS otimizada para a criptografia pós-quântica, na qual novas mensagens e extensões são adicionadas ao protocolo. O KEMTLS propõe o uso de KEMs no processo de troca de chaves e no processo de autenticação. Dessa forma, o KEMTLS é baseado na execução de duas trocas de chaves KEM, a troca de chaves KEM efêmera, responsável por gerar o segredo compartilhado efêmero, e a troca de chaves KEM estática, responsável por autenticar o servidor e gerar o segredo compartilhado estático, e por fim, ambos os segredos compartilhados são incorporados ao *key schedule*. Desta forma, tanto a troca de chaves KEM efêmera quanto a troca de chaves KEM estática contribuem na geração das chaves do protocolo, e apenas participantes autênticos (isto é, que possuem o segredo compartilhado estático) podem computar estas chaves.

1.7. Adoção da criptografia pós-quântica híbrida no TLS

A fim de adicionar suporte para a criptografia pós-quântica híbrida no TLS, foram desenvolvidos os protocolos TLS pós-quântico híbrido e KEMTLS híbrido.

O TLS pós-quântico híbrido é uma extensão do TLS pós-quântico foi desenvolvida em [Crockett et al. 2019], na qual são utilizados algoritmos clássicos em conjunto com algoritmos pós-quânticos no protocolo de troca de chaves e na autenticação das partes comunicantes, conforme definido no *internet-draft Hybrid key exchange in TLS 1.3* [Stebila et al. 2023]. Na troca de chaves do TLS pós-quântico híbrido, o algoritmo de KEM é executado em conjunto com o algoritmo de Diffie-Hellman, e os segredos resultantes de cada algoritmo de troca de chave são concatenados e incorporadorados ao *key*

schedule do protocolo, gerando dessa forma as chaves de criptografia simétricas do protocolo Para a autenticação das partes do protocolo, são utilizados certificados digitais X.509 híbridos, isto é, certificados digitais assinados por um algoritmo pós-quântico híbrido de assinatura, e que contêm uma chave pública pós-quântica híbrida de assinatura.

O KEMTLS híbrido (Hybrid KEMTLS) foi desenvolvido em trabalho anterior [Giron et al. 2023] e será abordado com mais detalhes na seção 1.8.

1.8. Trabalho anterior

O KEMTLS híbrido (Hybrid KEMTLS), desenvolvido em trabalho anterior [Giron et al. 2023], é uma extensão do protocolo KEMTLS que adiciona suporte à criptografia pós-quântica híbrida por meio da incorporação de algoritmos clássicos de KEM nas operações criptográficas do protocolo.

Na incorporação de KEM clássicos ao protocolo, são utilizados esquemas de *Hybrid Public Key Encryption* (HPKE) [Barnes et al. 2022] para instanciar o algoritmo de Diffie-Hellman na forma de um KEM. Nestes esquemas, o algoritmo de Diffie-Hellman é combinado a uma função de derivação de chave HKDF, resultando em um KEM clássico.

O *handshake* do KEMTLS híbrido (Figura 1) segue o *handshake* do KEMTLS, sem alterações na estrutura do protocolo, no entanto, agora para cada operação de KEM, há uma operação de KEM pós-quântico e uma operação de KEM clássico. A troca de chaves efêmera com KEMs híbridos segue o *internet-draft Hybrid key exchange in TLS 1.3* [Stebila et al. 2023]. Na transmissão das chaves públicas de KEM estático, são utilizados certificados X.509 pós-quânticos híbridos de KEM, isto é, certificados digitais assinados por um algoritmo de assinatura pós-quântico híbrido e que a chave pública corresponde à concatenação de uma chave pública de KEM clássico com uma chave pública de KEM pós-quântico.

A fim de incorporar a troca de chaves KEM clássica e a troca de chaves KEM pós-quântica no processo de derivação das chaves de criptografia do protocolo, os segredos compartilhados clássicos e pós-quânticos são concatenados e servidos como IKM para as funções de HKDF-Extract correspondentes (Figura 2). Esta construção, inspirada no *key schedule* do TLS pós-quântico híbrido, corresponde ao combinador criptográfico *dual-PRF* [Bindel et al. 2019], e, portanto, garante a propriedade híbrida. Desta forma, as chaves de criptografia do protocolo permanecerão seguras enquanto um componente do KEM híbrido se manter seguro.

No trabalho anterior, [Giron et al. 2023], o suporte à criptografia pós-quântica híbrida também foi adicionado ao KEMTLS-PDK, obtendo-se o KEMTLS-PDK híbrido. Para dar suporte a KEMs híbridos, o KEMTLS-PDK híbrido usou a mesma abordagem que o KEMTLS híbrido: a troca de chaves efêmera segue o *internet-draft Hybrid key exchange in TLS 1.3* e a autenticação é feita utilizando certificados X.509 pós-quânticos híbridos de KEM.

No trabalho em questão, o KEMTLS híbrido foi avaliado com os algoritmos da terceira rodada do processo de padronização do da criptografia pós-quântica do NIST, que eram os algoritmos finalistas em 2022. São eles:

- KEM: Kyber, NTRU e SABER
- Assinatura: Dilithium, Falcon e SPHINCS+

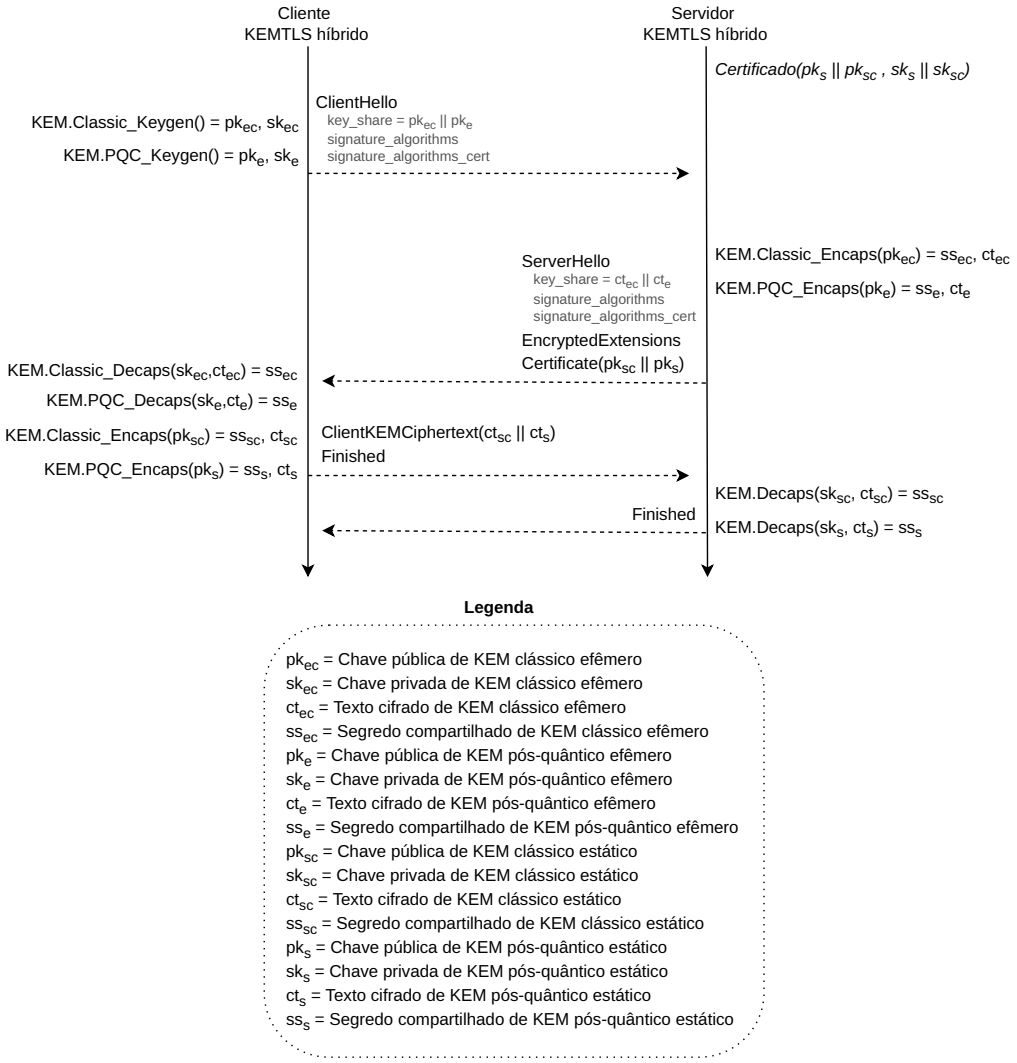


Figura 1. Protocolo de *handshake* do KEMTLS híbrido (apenas autenticação do servidor)

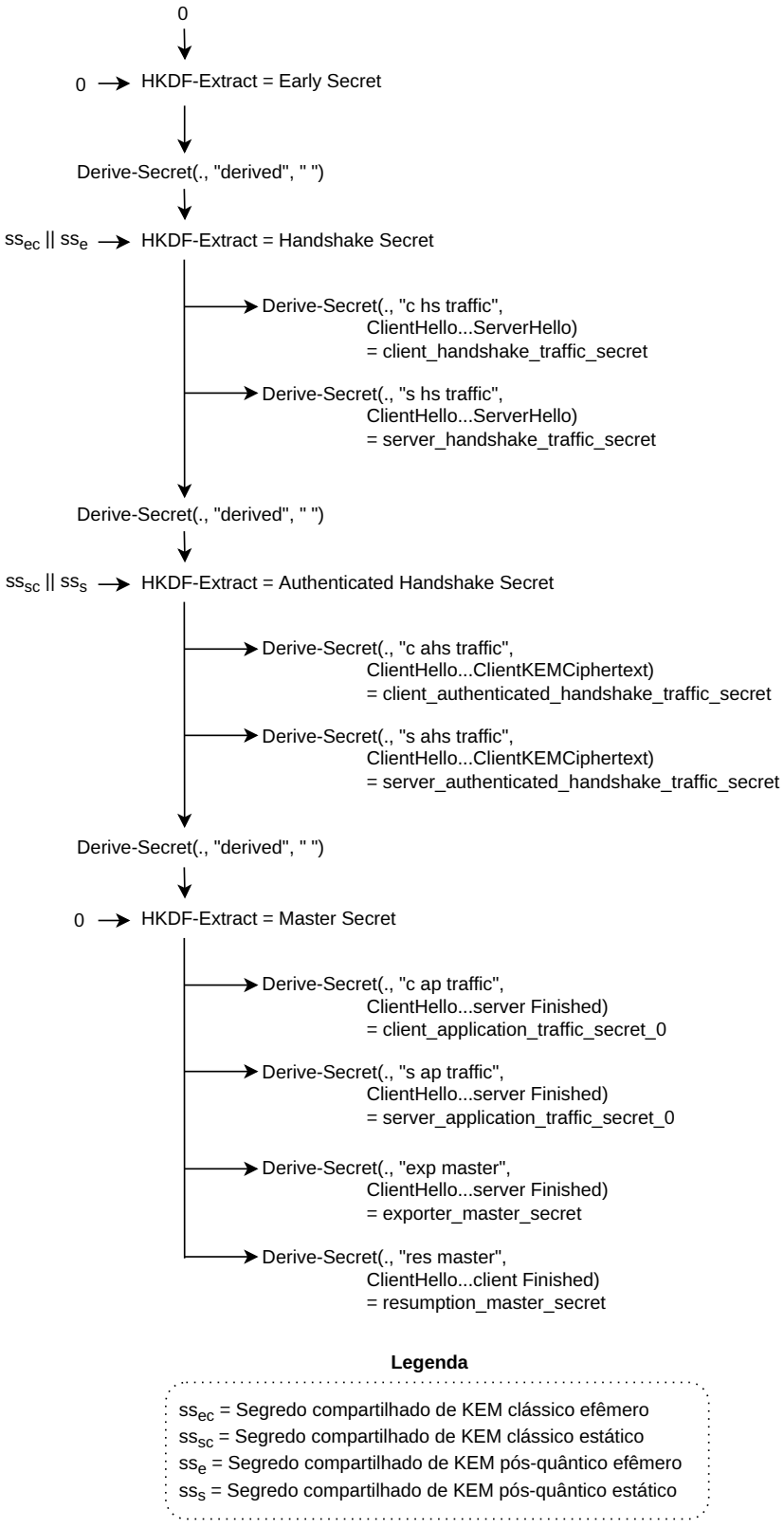


Figura 2. *Key schedule* do KEMTLS híbrido

2. Proposta

Este trabalho se propõe a expandir o trabalho anterior [Giron et al. 2023], instanciando os algoritmos candidatos à padronização da rodada 4 do processo de padronização da criptografia pós-quântica do NIST no KEMTLS híbrido, a fim de avaliar seu desempenho em um ambiente realista. Além disso, busca-se comparar a latência do KEMTLS híbrido com a do TLS pós-quântico híbrido, identificando dessa forma o protocolo pós-quântico híbrido com melhor desempenho.

2.1. Implementação

Para a instanciação dos algoritmos candidatos à padronização no KEMTLS híbrido, optou-se por integrar a biblioteca *liboqs* [Project a] ao projeto.

A biblioteca *liboqs* é uma biblioteca de código-fonte aberto desenvolvida pela *Open Quantum Safe* [Stebila and Mosca 2017] que contém a implementação de uma vasta gama de algoritmos pós-quânticos. Dado que a linguagem de programação desta biblioteca é C, foi utilizada a biblioteca *liboqs-go* [Project b], uma biblioteca *wrapper*[3] que disponibiliza a API da *liboqs* para a linguagem Go. Por meio da *liboqs-go*, foi possível incorporar algoritmos de KEM e assinatura pós-quânticos ao KEMTLS híbrido.

Para este trabalho, para cada algoritmo candidato, buscou-se incorporar uma versão do algoritmo para cada nível de segurança do NIST [Moody 2018] dentre 1, 3 e 5 :

- **Nível de segurança 1:** HQC-128, BIKE-L1, Classic-McEliece-348864;
- **Nível de segurança 3:** HQC-192, BIKE-L3, Classic-McEliece-460896;
- **Nível de segurança 5:** HQC-256, BIKE-L5, Classic-McEliece-6688128.

A fim de incorporar os algoritmos de KEM clássicos ao projeto do KEMTLS híbrido, utilizou-se a biblioteca CIRCL, a qual possui a implementação do protocolo HPKE para o algoritmo de Diffie-Hellman sobre curvas elípticas. No KEMTLS híbrido, o KEM clássico foi instanciado com as seguintes curvas elípticas, recomendadas pelo NIST em [of Standards and Technology 2023]: P-256, P-384 e P-521.

2.2. Algoritmos integrados

Na implementação, os seguintes algoritmos de KEM híbridos foram integrados:

- **Nível de segurança 1:** P256_HQC_128, P256_BIKE_L1, P256_Classic_McEliece_348864;
- **Nível de segurança 3:** P384_HQC_192, P384_BIKE_L3, P384_Classic_McEliece_460896;
- **Nível de segurança 5:** P521_HQC_256, P521_BIKE_L5, P521_Classic_McEliece_6688128.

O nome dos algoritmos é composto de duas partes, a primeira parte indica a curva elíptica utilizada pelo algoritmo de Diffie-Hellman do KEM clássico: o prefixo P256_

[3]Bibliotecas *wrapper*, através do mecanismo de *foreign function interface*, adaptam uma interface já existente, escrita em uma linguagem de programação, para uma interface compatível em outra linguagem de programação.

corresponde à curva elíptica P-256; P384_ à curva P-384; e P521_ à curva P-521. A segunda parte do nome indica o algoritmo de KEM pós-quântico.

Para os processos que envolvem assinaturas digitais, os seguintes algoritmos de assinatura pós-quânticos híbridos foram integrados:

- **Nível de segurança 1:** P256_Dilithium2;
- **Nível de segurança 3:** P384_Dilithium3;
- **Nível de segurança 5:** P521_Dilithium5.

Novamente, o nome dos algoritmos é composto por duas partes. A primeira parte indica a curva elíptica utilizada pelo algoritmo de ECDSA e a segunda parte indica o algoritmo de assinatura pós-quântico.

3. Avaliação

De modo a avaliar a instanciação dos algoritmos candidatos à padronização no KEMTLS híbrido, um conjunto de experimentos foi executado sobre a implementação. Estes experimentos buscam atender os seguintes objetivos:

- Mensurar o desempenho dos algoritmos candidatos em KEMs híbridos;
- Avaliar o impacto dos algoritmos candidatos no KEMTLS híbrido;
- Comparar o desempenho entre os protocolos pós-quânticos híbridos KEMTLS híbrido e TLS pós-quântico híbrido instanciados com os algoritmos candidatos;
- Determinar se o KEMTLS híbrido possui melhor desempenho que o TLS pós-quântico híbrido;
- Determinar o algoritmo candidato com melhor desempenho em protocolos pós-quânticos híbridos.

3.1. Metodologia

Para a avaliação do impacto dos algoritmos candidatos no KEMTLS híbrido, foram conduzidos experimentos utilizando duas máquinas virtuais, uma atuando como cliente e a outra como servidor. Nestas máquinas virtuais, foi instanciado o KEMTLS híbrido e o TLS pós-quântico híbrido, a fim de poder comparar o desempenho do KEMTLS híbrido com outro protocolo pós-quântico híbrido. Em ambos os protocolos, apenas a autenticação do servidor foi habilitada, por conta deste ser o cenário mais comum na navegação na internet.

Foram realizados três experimentos: experimento 0, experimento 1 e experimento 2.

Experimento 0: Inicialmente, buscou-se mensurar o desempenho dos algoritmos que foram integrados no KEMTLS híbrido isoladamente, isto é, sem estarem instanciados no protocolo. Para os KEMs, foi medido o tempo de execução de cada algoritmo para cada operação de KEM: geração de chaves, encapsulamento, desencapsulamento. Para os algoritmos de assinatura, foi medido o tempo de execução de cada algoritmo para as operações: assinatura e verificação de assinatura.

Experimento 1: No primeiro experimento, foram executados 1000 *handshakes* para cada algoritmo de KEM no KEMTLS híbrido e no TLS pós-quântico híbrido. Como

no KEMTLS são utilizados KEMs tanto para troca de chaves como para autenticação, utilizou-se o mesmo algoritmo para ambos os processos. No TLS, o algoritmo de KEM foi utilizado apenas na troca de chaves, e para a autenticação foram utilizados os algoritmos P256_Dilithium2, P384_Dilithium3 e P521_Dilithium5, de acordo com o nível de segurança do KEM. O KEM Classic McEliece foi excluído deste experimento pelo fato de possuir uma chave pública de tamanho superior ao tamanho limite estabelecido para pacotes TLS. Para fins de comparação, buscou-se agrupar os resultados dos experimentos pelo nível de segurança dos algoritmos utilizados na conexão, resultando nas seguintes configurações:

Tabela 1. Configurações do experimento 1

Protocolo	Nível	Troca de Chaves	Autenticação	Abreviatura
KEMTLS híbrido	1	P256_HQC_128	P256_HQC_128	L1-HQC-HQC
		P256_BIKE_L1	P256_BIKE_L1	L1-BIKE-BIKE
	3	P384_HQC_192	P384_HQC_192	L3-HQC-HQC
		P384_BIKE_L3	P384_BIKE_L3	L3-BIKE-BIKE
	5	P521_HQC_256	P521_HQC_256	L5-HQC-HQC
		P521_BIKE_L5	P521_BIKE_L5	L5-BIKE-BIKE
TLS pós-quântico híbrido	1	P256_HQC_128	P256_Dilithium2	L1-HQC-DIL
		P256_BIKE_L1	P256_Dilithium2	L1-BIKE-DIL
	3	P384_HQC_192	P384_Dilithium3	L3-HQC-DIL
		P384_BIKE_L3	P384_Dilithium3	L3-BIKE-DIL
	5	P521_HQC_256	P521_Dilithium5	L5-HQC-DIL
		P521_BIKE_L5	P521_Dilithium5	L5-BIKE-DIL

Experimento 2: O segundo experimento buscou avaliar o desempenho dos protocolos pós-quânticos híbridos em cenários em que há *cache* de certificados. Para isto, foram executados 1000 *handshakes* para cada algoritmo de KEM no KEMTLS-PDK híbrido e no TLS pós-quântico híbrido com a extensão *Cached Information*. Para a troca de chaves, em ambos os protocolos, foram utilizados os algoritmos BIKE e HQC. Classic McEliece foi novamente excluído da troca de chaves por conta do tamanho de sua chave pública. Para a autenticação no KEMTLS-PDK híbrido, no entanto, o Classic McEliece foi incluído, dado que sua chave pública não precisa ser transmitida no *handshake*, uma vez que há cache de certificados. Para a autenticação do TLS pós-quântico híbrido, são novamente utilizados algoritmos de assinatura pós-quânticos híbridos. Neste experimento, as seguintes configurações foram aplicadas aos protocolos:

Tabela 2. Configurações do experimento 2

Protocolo	Nível	Troca de Chaves	Autenticação	Abreviatura
KEMTLS-PDK híbrido	1	P256_HQC_128	P256_HQC_128	L1-HQC-HQC
		P256_BIKE_L1	P256_BIKE_L1	L1-BIKE-BIKE
		P256_HQC_128	P256_Classic_McEliece_348864	L1-HQC-CM
		P256_BIKE_L1	P256_Classic_McEliece_348864	L1-BIKE-CM
	3	P384_HQC_192	P384_HQC_192	L3-HQC-HQC
		P384_BIKE_L3	P384_BIKE_L3	L3-BIKE-BIKE
		P384_HQC_192	P384_Classic_McEliece_460896	L3-HQC-CM
		P384_BIKE_L3	P384_Classic_McEliece_460896	L3-BIKE-CM
	5	P521_HQC_256	P521_HQC_256	L5-HQC-HQC
		P521_BIKE_L5	P521_BIKE_L5	L5-BIKE-BIKE
		P521_HQC_256	P521_Classic_McEliece_6688128	L5-HQC-CM
		P521_BIKE_L5	P521_Classic_McEliece_6688128	L5-BIKE-CM
TLS pós-quântico híbrido (*Cached Information*)	1	P256_HQC_128	P256_Dilithium2	L1-HQC-DIL
		P256_BIKE_L1	P256_Dilithium2	L1-BIKE-DIL
	3	P384_HQC_192	P384_Dilithium3	L3-HQC-DIL
		P384_BIKE_L3	P384_Dilithium3	L3-BIKE-DIL
	5	P521_HQC_256	P521_Dilithium5	L5-HQC-DIL
		P521_BIKE_L5	P521_Dilithium5	L5-BIKE-DIL

3.1.1. Ambiente

Os experimentos realizados foram implementados no repositório *hybrid-kemtls-tests*, disponível em `https://github.com/JPADN/hybrid_kemtls_tests`.

Para avaliar o impacto dos algoritmos candidatos no KEMTLS híbrido, buscou-se mensurar o desempenho deste em um ambiente realista. Para isto, foi utilizada a plataforma de *cloud computing Google Cloud Platform* (GCP). Nesta plataforma, para os experimentos 1 e 2, foram alocadas duas máquinas virtuais do tipo N2 [4] geograficamente distantes, uma atuando como cliente da conexão e outra como servidor. A latência entre as duas máquinas foi de 50 ms.

3.1.2. Métrica de avaliação

No experimento 0, utilizou-se como métrica de avaliação o tempo de execução médio dos algoritmos de KEMs e assinatura digital e o tamanho em bytes dos objetos criptográficos transmitidos por estes (chave pública, assinatura, etc.).

Já a métrica de avaliação usada nos experimentos 1 e 2 é o tempo decorrido do começo do *handshake* até o momento em que o cliente pode enviar dados de aplicação, que coincide com o instante em que a mensagem *Finished* do cliente é enviada. Esta métrica foi nomeada como *time to send app data*.

Custo de autenticação:

A fim de auxiliar a análise dos resultados dos experimentos 1 e 2, foi mensurado o custo de autenticação para cada configuração avaliada nos protocolos. O custo de

[4]Documentação da família de máquinas virtuais N2 da GCP disponível em `https://cloud.google.com/compute/docs/general-purpose-machines?hl=pt-br`

autenticação refere-se ao tempo gasto pelo algoritmo de autenticação e a quantidade de bytes transmitidos no fluxo de autenticação contido na métrica *time to send app data*. O tempo gasto pelo algoritmo corresponde à soma dos tempos de execução presentes nas tabelas 7 e 9 para as operações criptográficas realizadas. A quantidade de bytes transmitidos corresponde à soma dos tamanhos presentes nas tabelas 8 e 10 para os objetos criptográficos transmitidos.

Tabela 3. Custo de autenticação do KEMTLS híbrido no experimento 1

Configuração	Custo de tempo (ms)	Custo de transmissão (bytes)
L1-HQC-HQC	4,30	2314
L1-BIKE-BIKE	0,17	1606
L3-HQC-HQC	13,38	4619
L3-BIKE-BIKE	1,00	3180
L5-HQC-HQC	37,93	7378
L5-BIKE-BIKE	13,88	5255

Tabela 4. Custo de autenticação do TLS pós-quântico híbrido no experimento 1

Configuração	Custo de tempo (ms)	Custo de transmissão (bytes)
L1-HQC-DIL e L1-BIKE-DIL	0,28	3873
L3-HQC-DIL e L3-BIKE-DIL	12,47	5450
L5-HQC-DIL e L5-BIKE-DIL	20,93	7464

Tabela 5. Custo de autenticação do KEMTLS-PDK híbrido no experimento 2

Configuração	Custo de tempo (ms)	Custo de transmissão (bytes)
L1-HQC-HQC	10,76	4498
L1-BIKE-BIKE	0,81	1638
L1-HQC-CM e L1-BIKE-CM	0,26	161
L3-HQC-HQC	33,64	9075
L3-BIKE-BIKE	3,67	3212
L3-HQC-CM e L3-BIKE-CM	1,64	253
L5-HQC-HQC	86,21	14554
L5-BIKE-BIKE	32,09	5287
L5-HQC-CM e L5-BIKE-CM	27,49	341

Tabela 6. Custo de autenticação do TLS pós-quântico híbrido com *Cached Information* no experimento 2

Configuração	Custo de tempo (ms)	Custo de transmissão (bytes)
L1-HQC-DIL e L1-BIKE-DIL	0,28	2494
L3-HQC-DIL e L3-BIKE-DIL	12,47	3399
L5-HQC-DIL e L5-BIKE-DIL	20,93	4737

3.2. Resultados

3.2.1. Experimento 0

O experimento 0 foi dividido em duas partes: avaliação dos KEMs híbridos, e avaliação dos algoritmos de assinatura pós-quânticos híbridos.

Na Tabela 7, encontram-se os tempos de execução médios de cada KEM híbrido integrado nos protocolos.

Tabela 7. Tempo de execução médio das operações de KEM em milissegundos

Nível	KEM	Ger. Chave	Encaps.	Decaps.
1	P256_HQC_128	2,10	4,30	6,46
1	P256_BIKE_L1	0,30	0,17	0,64
1	P256_Classic_McEliece_348864	48,47	0,12	0,14
3	P384_HQC_192	6,72	13,38	20,26
3	P384_BIKE_L3	1,22	1,00	2,67
3	P384_Classic_McEliece_460896	170,14	0,80	0,84
5	P521_HQC_256	18,62	37,93	48,28
5	P521_BIKE_L5	8,89	13,88	18,21
5	P521_Classic_McEliece_6688128	235,69	13,93	13,56

A Tabela 8 contém, para cada KEM híbrido avaliado, o tamanho em bytes de chave pública e do texto cifrado.

Tabela 8. Tamanho em bytes da chave pública e do texto cifrado para os KEMs

Nível	KEM	Chave pública	Texto cifrado
1	P256_HQC_128	2314	4498
1	P256_BIKE_L1	1606	1638
1	P256_Classic_McEliece_348864	261185	161
3	P384_HQC_192	4619	9075
3	P384_BIKE_L3	3180	3212
3	P384_Classic_McEliece_460896	524257	253
5	P521_HQC_256	7378	14554
5	P521_BIKE_L5	5255	5287
5	P521_Classic_McEliece_6688128	1045125	341

A partir da análise das tabelas 7 e 8, considerando uma troca de chaves KEM efêmera em que as operações de geração de chaves, encapsulamento e desencapsulamento são executadas, observa-se que o híbrido BIKE é o algoritmo com melhor desempenho, pois possui o menor tempo de execução e requer a menor quantidade de bytes transmitidos. Para troca de chaves KEM estática, em que apenas as operações de encapsulamento e desencapsulamento são executadas, o híbrido Classic McEliece é o algoritmo com menor tempo de execução. No entanto, por possuir uma chave pública que pode chegar a tamanhos na casa dos *megabytes*, seu uso só é adequado quando há *cache* da chave pública.

A Tabela 9 contém o tempo de execução médio das operações de assinatura e verificação para cada algoritmo de assinatura pós-quântico híbrido integrado nos protocolos.

Tabela 9. Tempo médio de execução dos algoritmos de assinatura pós-quânticos híbridos em milissegundos

Nível	Algoritmo de Assinatura	Assinatura	Verificação
1	P256_Dilithium2	0,14	0,14
3	P384_Dilithium3	4,34	8,13
5	P521_Dilithium5	7,23	13,7

Nos protocolos avaliados, dois componentes públicos do algoritmo de assinatura são transmitidos na conexão: a chave pública e a assinatura. A Tabela 10 contém os tamanhos em bytes da chave pública e da assinatura para cada algoritmo de assinatura pós-quântico híbrido integrado.

Tabela 10. Tamanho em bytes da chave pública e da assinatura dos algoritmos de assinatura pós-quânticos híbridos integrados

Nível	Algoritmo de Assinatura	Chave pública	Assinatura
1	P256_Dilithium2	1379	2494
3	P384_Dilithium3	2051	3399
5	P521_Dilithium5	2727	4737

3.2.2. Experimento 1

Os resultados do experimento encontram-se nos gráficos 3, 4 e 5. Nestes gráficos, cada coluna representa o *time to send app data* médio do protocolo para a configuração descrita no eixo x. Para facilitar a comparação entre os protocolos, as colunas foram agrupadas pelo algoritmo de troca de chaves utilizado: o grupo da esquerda agrupa as configurações onde a troca de chaves é feita com o híbrido HQC, e o grupo da direita o híbrido BIKE. A dispersão dos dados amostrados foi representada pela barra de erro no topo da coluna, que corresponde ao desvio padrão dos dados.

Para todos os níveis de segurança, o híbrido BIKE desempenhou melhor que o híbrido HQC, o que é esperado, uma vez que o tempo de execução do híbrido BIKE é consideravelmente menor para todas as operações criptográficas, assim como o tamanho de sua chave pública e texto cifrado, conforme as tabelas 7 e 8.

Quanto à comparação entre os protocolos, no nível 1 (Figura 3), ambos os protocolos tiveram desempenho equivalente para o híbrido HQC, enquanto que para o híbrido BIKE o TLS pós-quântico híbrido demonstrou uma vantagem significativa, com um *time to send app data* 20 ms menor em relação ao KEMTLS híbrido. Este resultado vai contra o resultado esperado, uma vez que o custo de autenticação do L1-BIKE-BIKE é inferior ao custo do L1-BIKE-DIL.

Para o nível 3 (Figura 4), novamente ambos os protocolos tiveram um desempenho equivalente para o híbrido HQC, no entanto, neste nível de segurança, o KEMTLS híbrido

superou o desempenho do TLS pós-quântico híbrido para o híbrido BIKE, apresentando um *time to send app data* médio 15 ms menor. Ao comparar o custo de autenticação das configurações, observa-se que o L3-BIKE-BIKE custa 1 ms de tempo de execução e 3180 bytes para transmissão, enquanto o L3-BIKE-DIL custa 12,47 ms de tempo de execução e 5450 bytes para transmissão, o que justifica o desempenho superior do L3-BIKE-BIKE.

No nível mais alto de segurança, nível 5 (Figura 5), o TLS pós-quântico híbrido teve uma latência menor que o KEMTLS híbrido para o híbrido HQC. Já para o híbrido BIKE, ambos os protocolos tiveram um desempenho muito próximo, com o KEMTLS híbrido com um *time to send app data* levemente inferior ao TLS pós-quântico híbrido, o que está de acordo com o custo de autenticação de cada protocolo: L5-BIKE-BIKE possui um custo de autenticação de 13,88 ms e 5255 bytes, levemente inferior ao do L5-BIKE-DIL, de 20,93 ms e 7674 bytes.

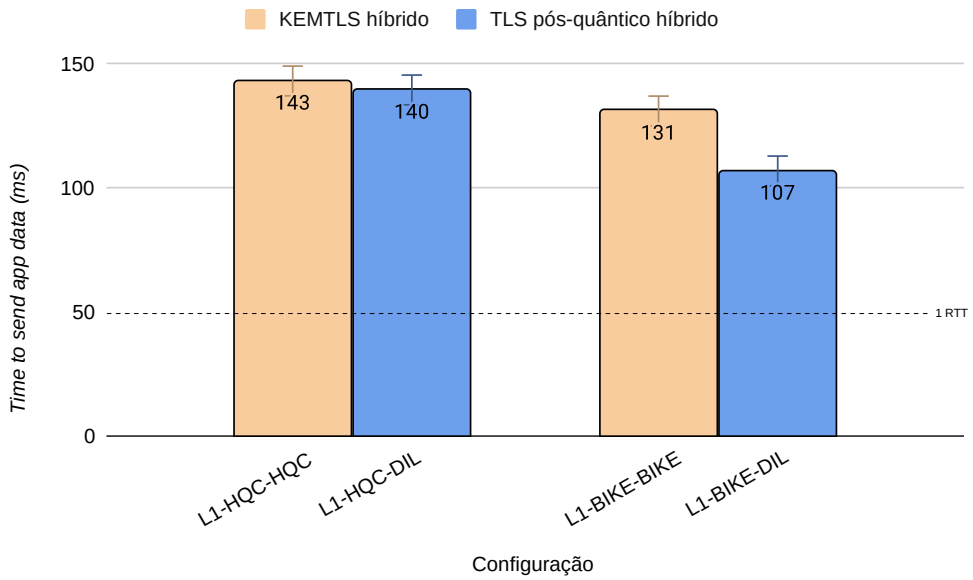


Figura 3. *Time to send app data* médio do experimento 1 para o nível de segurança 1

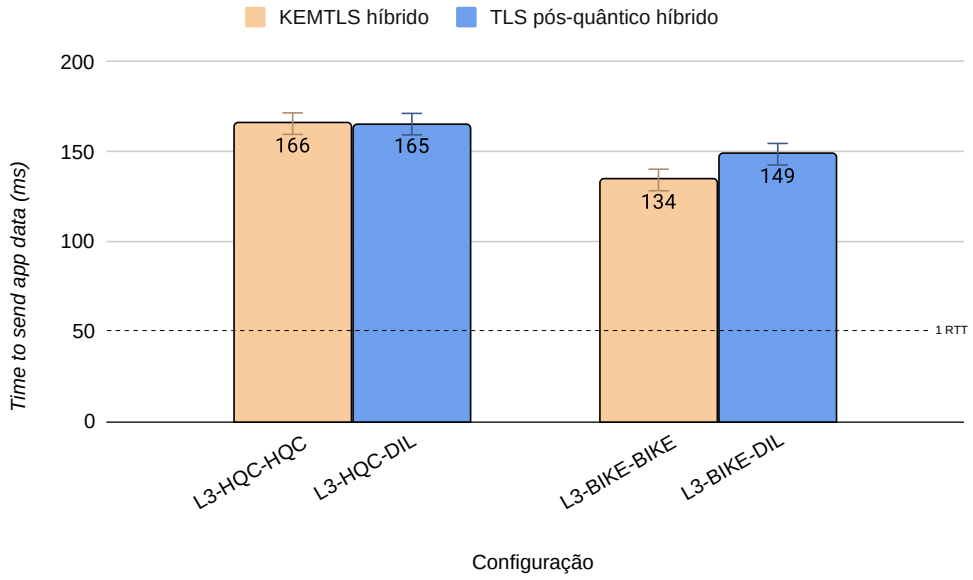


Figura 4. *Time to send app data* médio do experimento 1 para o nível de segurança 3

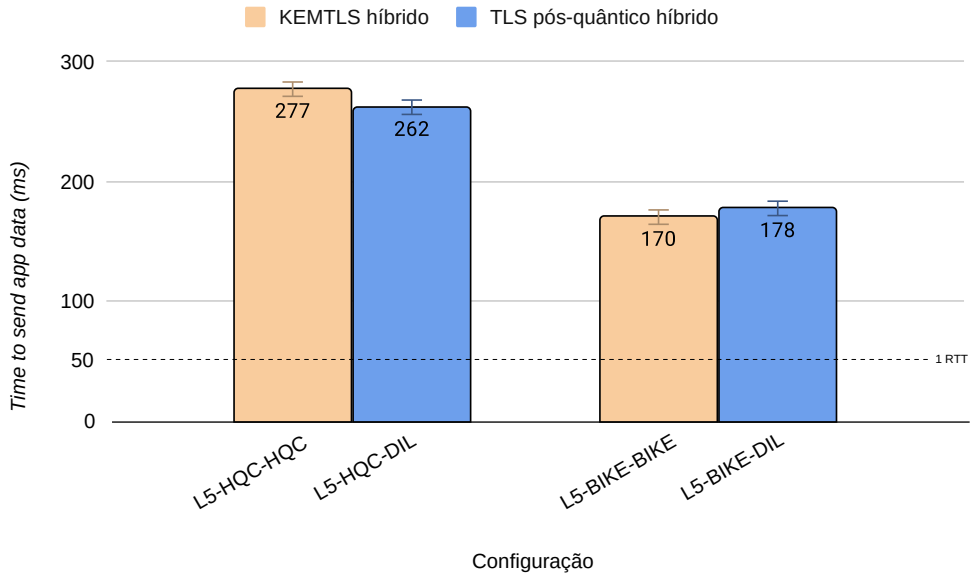


Figura 5. *Time to send app data* médio do experimento 1 para o nível de segurança 5

3.2.3. Experimento 2

Os resultados do experimento 2 encontram-se nos gráficos 6, 7 e 8. A interpretação dos gráficos é a mesma do experimento 1. Diferentemente do experimento 1, no experimento 2 o KEMTLS-PDK híbrido foi avaliado com dois tipos de configuração, uma em que o algoritmo de KEM estático é o mesmo que o de KEM efêmero, e outra em que o algoritmo de KEM estático é o Classic McEliece. Por conta disso, cada agrupamento de colunas possui duas colunas para o KEMTLS-PDK híbrido.

Assim como no experimento 1, as configurações com o híbrido BIKE na troca de chaves tiveram um desempenho superior às configurações com híbrido HQC, para todos os níveis de segurança.

Para o nível de segurança 1 (Figura 6), considerando o híbrido HQC na troca de chaves (agrupamento de colunas à esquerda), o KEMTLS-PDK híbrido e TLS pós-quântico híbrido obtiveram um *time to send app data* equivalente para as configurações L1-HQC-CM e L1-HQC-DIL, apesar de o L1-HQC-CM possuir um custo de transmissão consideravelmente menor. Considerando o híbrido BIKE na troca de chaves (agrupamento de colunas à direita), todas as configurações obtiveram um *time to send app data* equivalente de cerca de 80 ms, uma vez que todas as configurações possuem uma autenticação com custo de tempo inferior a 1 ms. Novamente, apesar de a configuração L1-BIKE-CM possuir um custo de transmissão consideravelmente menor, de 161 bytes, isto não impactou significativamente o *time to send app data*.

Quanto ao nível de segurança 3 (Figura 7), considerando o híbrido HQC na troca de chaves, o KEMTLS-PDK híbrido, com a configuração L3-HQC-CM, teve um tempo de

execução levemente menor que TLS pós-quântico híbrido, com a configuração L3-HQC-DIL. A diferença no custo de autenticação justifica isto: L3-HQC-CM custa 1,64 ms e 253 bytes, enquanto que o L3-HQC-DIL 12,47 ms e 3399 bytes. Para as configurações com o híbrido BIKE na troca de chaves, o KEMTLS-PDK híbrido novamente tem um tempo de execução levemente menor, para ambas as configurações L3-BIKE-BIKE e L3-BIKE-CM, que possuem um desempenho equivalente entre si levando em conta o desvio padrão do experimento.

No nível de segurança 5 (Figura 8), o TLS pós-quântico híbrido teve um desempenho superior ao KEMTLS-PDK híbrido em todas as configurações. Para as configurações em que o híbrido HQC é usado na troca de chaves, enquanto as configurações L5-HQC-CM e L5-HQC-DIL tiveram um desempenho próximo, a configuração L5-HQC-HQC obteve um *time to send app data* cerca de 100 ms maior que estas, por conta do seu custo de autenticação de 86,21 ms e 14554 bytes. Já para configurações em que o híbrido BIKE é usado na troca de chaves, a configuração do KEMTLS-PDK híbrido que mais se aproximou do TLS pós-quântico híbrido foi a L5-BIKE-BIKE, com custo de autenticação de 32,09 ms e 5287 bytes, ao invés da configuração L5-BIKE-CM, com custo de 27,49 ms e 341 bytes. Este resultado é justificável quando observamos o custo de tempo similar entre as duas configurações e consideramos a natureza imprevisível de um experimento realista. No entanto, esperava-se que o custo de transmissão consideravelmente inferior do L5-BIKE-CM tivesse um impacto maior no *time to send app data*.

O experimento 2 evidencia que o desempenho do protocolo está fortemente relacionado ao tempo de execução dos algoritmos, enquanto que a quantidade de bytes transmitidos aparenta não ter tanto impacto, uma vez que as configurações que utilizam o híbrido Classic McEliece na autenticação, apesar de possuírem um custo de transmissão baixíssimo, não desempenharam melhor que as configurações que utilizam o híbrido BIKE na autenticação, que possui um custo de tempo similar, mas um custo de transmissão significativamente maior.

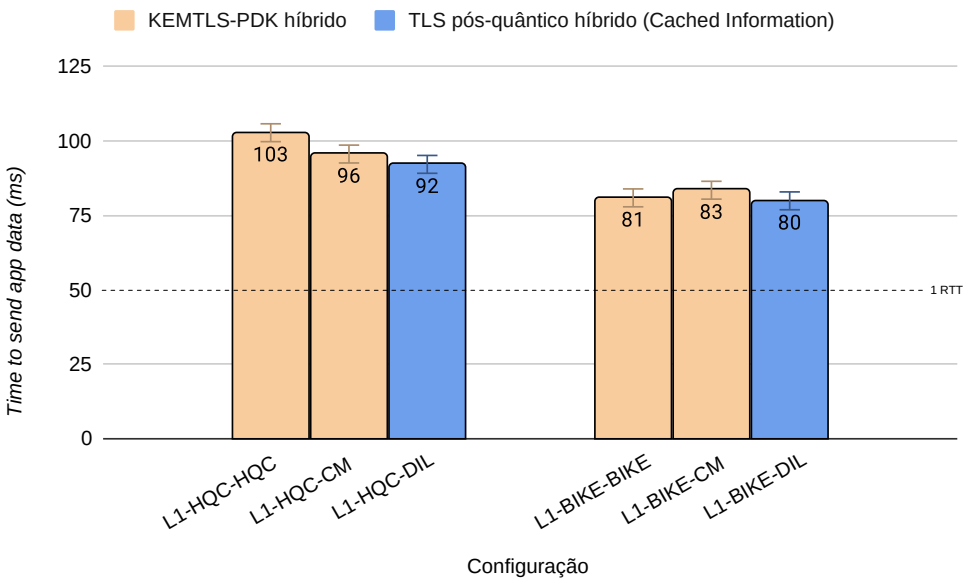


Figura 6. *Time to send app data* médio do experimento 2 para o nível de segurança 1

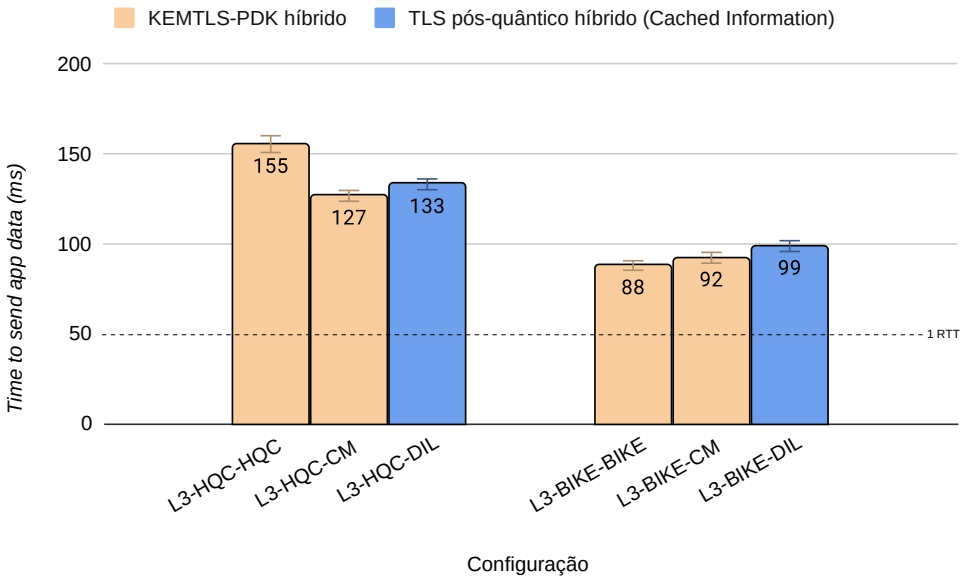


Figura 7. *Time to send app data* médio do experimento 2 para o nível de segurança 3

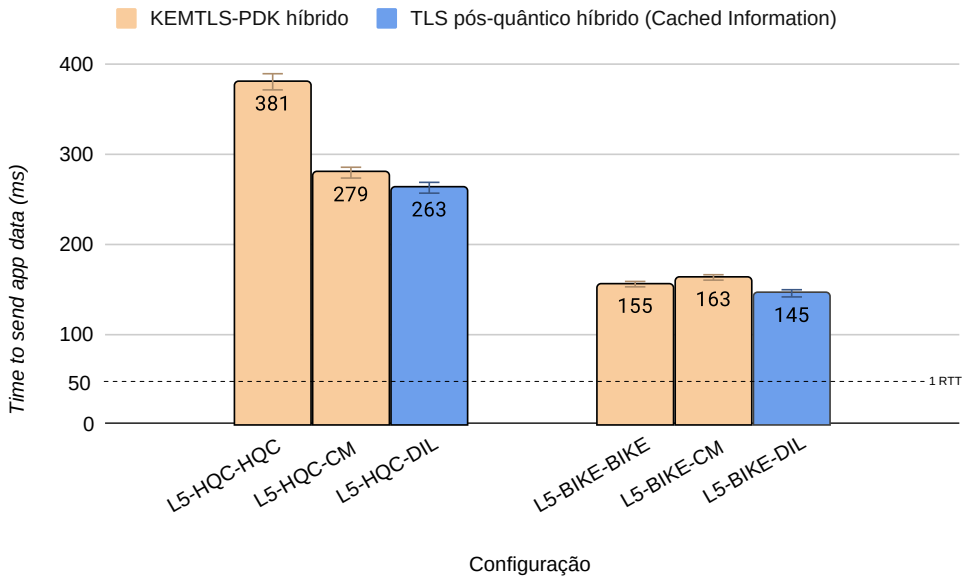


Figura 8. *Time to send app data* médio do experimento 2 para o nível de segurança 5

4. Considerações finais

A análise dos resultados dos experimentos 1 e 2, evidencia que em todos os níveis de segurança, as configurações com o algoritmo BIKE na troca de chaves e autenticação foram as que obtiveram menor latência, portanto, conclui-se que o BIKE é o algoritmo candidato com melhor desempenho para instanciação no KEMTLS híbrido e no KEMTLS-PDK híbrido.

No que diz respeito à comparação entre o desempenho do KEMTLS híbrido com o TLS pós-quântico híbrido, nenhum protocolo se destacou em relação ao outro, uma vez que ambos apresentaram um *time to send app data* similar, de modo geral. Em cenários em que não há *cache* de certificados, o TLS pós-quântico híbrido é o protocolo com melhor desempenho no nível de segurança 1, enquanto que no nível de segurança 3 e 5, o KEMTLS híbrido possui um desempenho superior. Em cenários onde há *cache* de certificados, ambos os protocolos possuem desempenho equivalente no nível de segurança 1, no nível de segurança 3 o KEMTLS-PDK supera o desempenho do TLS pós-quântico híbrido, e no nível de segurança 5 é o TLS pós-quântico híbrido quem desempenha melhor.

Além destes resultados, os experimentos realizados evidenciaram que o tempo de execução de um algoritmo de KEM afeta significativamente o desempenho do protocolo KEMTLS híbrido, enquanto que o tamanho dos objetos criptográficos transmitidos (chave pública e texto cifrado) não possui um impacto significativo.

Referências

Arute, F. et al. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510.

Barnes, R., Bhargavan, K., Lipp, B., and Wood, C. A. (2022). Hybrid Public Key Encryption. RFC 9180.

Beullens, W. (2022). Breaking rainbow takes a weekend on a laptop. In Dodis, Y. and Shrimpton, T., editors, *Advances in Cryptology – CRYPTO 2022*, pages 464–479, Cham. Springer Nature Switzerland.

Beveridge, I. and Butcher, D. (2023). Harvest now, decrypt later – fact or fiction? https://www.entrust.com/blog/2023/11/harvest-now-decrypt-later-fact-or-fiction/. Acessado em 17/03/2024.

Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., and Stebila, D. (2019). Hybrid key encapsulation mechanisms and authenticated key exchange. In Ding, J. and Steinwandt, R., editors, *Post-Quantum Cryptography*, pages 206–226, Cham. Springer International Publishing.

Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S., and Cooper, D. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280.

Castryck, W. and Decru, T. (2023). An efficient key recovery attack on sidh. In Hazay, C. and Stam, M., editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 423–447, Cham. Springer Nature Switzerland.

Crockett, E., Paquin, C., and Stebila, D. (2019). Prototyping post-quantum and hybrid key exchange and authentication in tls and ssh. Cryptology ePrint Archive, Paper 2019/858. https://eprint.iacr.org/2019/858.

Diffie, W. and Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.

Fielding, R. T., Nottingham, M., and Reschke, J. (2022). HTTP Semantics. RFC 9110.

Giron, A. A., do Nascimento, J. P. A., Custódio, R., Perin, L. P., and Mateu, V. (2023). Post-quantum hybrid kemtls performance in simulated and real network environments. In Aly, A. and Tibouchi, M., editors, *Progress in Cryptology – LATINCRYPT 2023*, pages 293–312, Cham. Springer Nature Switzerland.

MIT Technology Review. O que é criptografia pós-quântica? https://mittechreview.com.br/o-que-e-criptografia-pos-quantica/. Acessado em 17/03/2024.

Moody, D. (2018). Let's get ready to rumble- the nist pqc competition. https://csrc.nist.gov/CSRC/media/Presentations/Let-s-Get-Ready-to-Rumble-The-NIST-PQC-Competiti/images-media/PQCrypto-April2018_Moody.pdf.

Mosca, D. M. and Piani, D. M. (2023). 2023 quantum threat timeline report. Technical report, Global Risk Institute.

NIST (2024). Post-quantum cryptography. https://csrc.nist.gov/projects/post-quantum-cryptography. Acessado em 16/02/2024.

of Standards, N. I. and Technology (2023). Digital signature standard (dss). Technical Report FIPS 186-5, U.S. Department of Commerce.

Project, O. Q. S. liboqs. https://github.com/open-quantum-safe/liboqs. Acessado em 18/06/2024.

Project, O. Q. S. liboqs-go: Go bindings for liboqs. https://github.com/open-quantum-safe/liboqs-go. Acessado em 18/06/2024.

Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446.

Santesson, S. and Tschofenig, H. (2016). Transport Layer Security (TLS) Cached Information Extension. RFC 7924.

Schwabe, P., Stebila, D., and Wiggers, T. (2020). Post-quantum tls without handshake signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 1461–1480, New York, NY, USA. Association for Computing Machinery.

Shor, P. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134.

Stebila, D., Fluhrer, S., and Gueron, S. (2023). Hybrid key exchange in TLS 1.3. Internet-Draft draft-ietf-tls-hybrid-design-09, Internet Engineering Task Force. Work in Progress.

Stebila, D. and Mosca, M. (2017). Post-quantum key exchange for the internet and the open quantum safe project. In Avanzi, R. and Heys, H., editors, *Selected Areas in Cryptography – SAC 2016*, pages 14–37, Cham. Springer International Publishing.

Wikipedia contributors (2024). Nist post-quantum cryptography standardization — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=NIST_Post-Quantum_Cryptography_Standardization&oldid=1212758724. [Acessado em 17/03/2024].

Wilton, R. (2020). Fact sheet: Quantum physics and computing. https://www.internetsociety.org/resources/doc/2020/does-quantum-computing-put-our-digital-security-at-risk. Acessado em 16/02/2024.