UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA

ESCOLA SUPERIOR DE TECNOLOGIA

ENGENHARIA DE COMPUTAÇÃO

ANDRÉ RICARDO MARQUES ROGÉRIO

UM ESTUDO COMPARATIVO DOS PROTOCOLOS DE AUTENTICAÇÃO *PAKE*: *CPACE* E *OPAQUE*

Manaus

2022

ANDRÉ RICARDO MARQUES ROGÉRIO

UM ESTUDO COMPARATIVO DOS PROTOCOLOS DE AUTENTICAÇÃO *PAKE*: *CPACE* E *OPAQUE*

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

Orientador: Prof. Ricardo Rios Monteiro do Carmo

Manaus

2022

Universidade do Estado do Amazonas - UEA

Escola Superior de Tecnologia - EST

Reitor:

André Luiz Nunes Zogahib

Vice-Reitor:

Kátia do Nascimento Couceiro

Diretora da Escola Superior de Tecnologia:

Ingrid Sammyne Gadelha Figueiredo

Coordenadora do Curso de Engenharia de Computação:

Áurea Hileia da Silva Melo

Coordenadora da Disciplina Projeto Final:

Áurea Hileia da Silva Melo

Banca Avaliadora composta por: *Data da Defesa: 31/05/2022*

Prof. Ricardo Rios Monteiro do Carmo, Mestre (Orientador)

Prof. Carlos Maurício Seródio Figueiredo, Doutor

Prof. Fabio Santos da Silva, Doutor

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).
Sistema Integrado de Bibliotecas da Universidade do Estado do Amazonas.

M357ee Rogério, André Ricardo Marques
Um estudo comparativo dos protocolos de autenticação PAKE: CPace e OPAQUE / André Ricardo Marques Rogério. Manaus : [s.n], 2022.
56 f.: color.; 30 cm.

TCC - Graduação em Engenharia de Computação - Universidade do Estado do Amazonas, Manaus, 2022.
Inclui bibliografia
Orientador: Ricardo Rios Monteiro do Carmo

1. Segurança de dados. 2. Autenticação. 3. Protocolo de Comunicação. I. Ricardo Rios Monteiro do Carmo (Orient.). II. Universidade do Estado do Amazonas. III. Um estudo comparativo dos protocolos de autenticação PAKE: CPace e OPAQUE

Elaborado por Jeane Macelino Galves - CRB-11/463

AMAZONAS
GOVERNO DO ESTADO

FOLHA DE APROVAÇÃO

“Um estudo comparativo dos protocolos de autenticação PAKE: CPace e OPAQUE”

ANDRÉ RICARDO MARQUES ROGÉRIO

Trabalho de Conclusão do Curso de Engenharia da Computação defendido e aprovado pela banca avaliadora constituída pelos professores:

Presidente

Prof. Ricardo Rios Monteiro do Carmo, Mestre.

UNIVERSIDADE DO ESTADO DO AMAZONAS

Arguidor

Prof. Dr. Carlos Maurício Seródio Figueiredo, Doutor.

UNIVERSIDADE DO ESTADO DO AMAZONAS

Arguidor

Prof. Dr. Fabio Santos da Silva, Doutor.

UNIVERSIDADE DO ESTADO DO AMAZONAS

Manaus, 31 de maio de 2022.

Resumo

A proteção efetiva de informação e dados é uma preocupação para muitas organizações. A quantidade de informação digital está aumentando cada vez mais e garantir sua integridade tem estimulado a busca por meios efetivos de proteção. Uma forma de aumentar a proteção de informação digital é através do uso de mecanismos de autenticação. Existem vários métodos de autenticação disponíveis. Cada método possui características e requisitos, vantagens e desvantagens, e escolher qual método é mais adequado para um determinado contexto e necessidade pode se tornar uma atividade complexa e demorada. Para auxiliar nesse escolha, esse trabalho apresenta um estudo comparativo de dois protocolos de autenticação, o *CPace* e o *OPAQUE*. Para comparar o esses protocolos foram definidos dois ambientes de teste que simularam casos de uso reais. Foram definidos dois requisitos para análise, o desempenho e uso de memória. A análise comparativa dos resultados da simulação indicou que o protocolo *CPace* se mostrou mais eficiente que o *OPAQUE*. A análise qualitativa indicou O *CPace* como um protocolo de autenticação mais apropriado para uso em dispositivos de baixo desempenho e, no que diz respeito à necessidade de sistemas que lidam com dados sensíveis e que necessitam de mecanismos mais robustos de autenticação, o *OPAQUE* se mostrou mais apropriado.

Palavras Chave: Segurança de dados; autenticação; protocolo de comunicação.

Abstract

Information and data effective security is a concern of many organizations. The quantity of digital information is increasing and ensuring its integrity has stimulated the search for effective means of protection. One way to increase the protection of digital information is through the use of authentication mechanisms. There are several authentication methods available. Each method has characteristics and requirements, advantages and disadvantages, and choosing which method is best suited for a given context and need can become a complex and time-consuming activity. To assist in this choice, this work presents a comparative study of two authentication protocols, *CPace* and *OPAQUE*. To compare these protocols, two test environments were defined that simulated real use cases. For the analysis, two requirements were defined, performance and memory usage. The comparative analysis of the simulation results indicated that the *CPace* protocol was more efficient than the *OPAQUE*. The qualitative analysis indicated *CPace* as a more appropriate authentication protocol for use in low-performance devices and, concerning the need for systems that deal with sensitive data and require more robust authentication mechanisms, *OPAQUE* proved to be more appropriate.

Keywords: Data security; authentication; communication protocol.

Agradecimentos

É com muita alegria e gratidão que eu gostaria de dedicar essa seção as pessoas que foram essenciais no meu processo de escrita dessa monografia.

Um agradecimento especial ao meu professor orientador mestre Ricardo Rios, que me auxiliou desde antes do começo da escrita até o fim. O professor Ricardo me guiou na direção certa quando eu estive perdido em relação ao trabalho, além de revisar esse trabalho com muito zelo para torná-lo o melhor possível. Muito obrigado, professor!

Um agradecimento ao Daniel Bourdrez, um dos autores do protocolo *OPAQUE*, que sempre foi muito solícito desde o meu primeiro contato, mesmo sem me conhecer. Daniel me auxiliou muito no entendimento do *OPAQUE* e do *CPace*.

Gostaria de agradecer também aos meus grandes amigos Jansen Saunier e Vinícius Lopes. O Jansen foi quem inicialmente me apresentou e sugeriu que eu escrevesse sobre o protocolo *OPAQUE*. O Vinícius me auxiliou muito quando eu tive dificuldades com o *Golang*, me ajudando a resolver diversos problemas e a melhorar a arquitetura da aplicação.

Não poderia deixar de agradecer a minha família por todo o apoio durante a escrita dessa monografia, e durante toda a minha jornada na universidade. Minha mãe, Ilis, e meu pai, Ricardo, me deram todo o apoio e condição de cursar essa graduação. Meu irmão, Anderson, me acolheu quando me mudei aqui pra Manaus e me deu carona diversas vezes para a faculdade quando precisei, me permitindo não chegar atrasado nas aulas. Minha irmã, Agnes, sempre me deu forças pra continuar, ao vê-la cursando Arquitetura e se sobressaindo sempre como uma das melhores aluna de todas as turmas em que participava. Minha noiva, Luiza, que está comigo desde antes de eu fazer o vestibular para cursar essa graduação e que, recentemente aceitou meu pedido de casamento. Luiza sempre esteve presente nos meus piores momentos, me apoiando e me dando forças para não desistir da graduação, mesmo depois de reprovar duas vezes na mesma matéria.

As palavras que eu escrevo aqui são poucas para expressar a gratidão que eu sinto por todos os que estiveram presentes de alguma forma na minha graduação. Muito obrigado a todos vocês. Sem a sua ajuda, nada disso seria possível.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Em comunicação entre pares de protagonistas, é importante o conhecimento de que cada participante da comunicação possui uma identificação válida para garantir que informação sigilosa não seja compartilhada com pessoas não autorizadas. Uma das formas de se garantir a autenticidade dos pares é através da autenticação. Em sistemas computacionais, um dos tipos mais empregados de autenticação é a senha direta (WU, 1998). Nesse tipo de autenticação, o único dado disponível para o cliente da comunicação é o segredo ou senha, que é transferida através de algum meio.

Como o segredo é passado na comunicação, é possível haver uma entidade mal-intencionada capaz de interceptar essa comunicação e adquirir o segredo de um dos protagonistas. Com isso, é possível facilmente obter acesso a informação sensível.

Vários métodos podem ser empregados para mitigar esse tipo de problema. Um deles é o chamado método *Password-Authenticated Key Agreement* (*PAKE*) (HAO; OORSCHOT, 2021), através do qual a comunicação entre o par de protagonistas é viabilizada evitando a transferência de dados sensíveis. Nesse trabalho, são descritos e comparados dois protocolos *PAKEs* com o objetivo de oferecer um estudo comparativo que possa ser útil a tomada de decisão sobre que protocolos de autenticação escolher para determinadas aplicações.

1.1 Objetivo Geral

O objetivo geral desse trabalho é realizar uma análise comparativa quantitativa e qualitativa de dois protocolos que utilizam o método *PAKE* com o intuito de facilitar a escolha de um desses protocolos em aplicações reais.

1.1.1 Objetivos Específicos

Os objetivos específicos desse trabalho são:

- Investigação de protocolos de autenticação já existentes;
- Definição de algoritmos e ferramentas utilizadas para a implementação dos sistemas de autenticação utilizando os protocolos selecionados;
- Implementação dos sistemas de autenticação;
- Análise comparativa dos protocolos de autenticação;
- Estudo de viabilidade dos protocolos em aplicações reais.

1.2 Justificativa

Em sistemas de informação, como, por exemplo, sistemas Web, Correio Eletrônico, Sistemas de Banco de Dados, aplicativos móveis, dados são armazenados com acesso local ou remoto e são transferidos entre os vários participantes da comunicação. Alguns desses dados são classificados como privativos ou de acesso restrito. Para garantir a privacidade, confidencialidade, integridade e disponibilidade desses dados, um dos métodos utilizados é a autenticação.

A autenticação deve garantir que, por exemplo, uma entidade (ser humano, sistema computacional etc.) comprove que tem as credenciais necessárias e suficientes para acesso aos dados ou sistema. Há vários métodos de autenticação, mas geralmente pode-se classificá-los em três grandes classes (WU, 1998):

1. algo que o usuário sabe (AUS): senhas, números de identificação pessoal (PINs), chave segredo etc.;
2. algo que o usuário é (AUE): biometria, escaneamento de retina, identificação por voz etc.;
3. algo que o usuário tem (AUT): cartão de identificação, token eletrônico, cartões inteligentes etc.

Dentre esses, os mais utilizados são os da classe AUS, foco desse trabalho.

A Fig. 1.1 ilustra o mecanismo básico de funcionamento dos métodos AUS. Os sistemas de autenticação AUS, com algumas variações, requerem que clientes enviem um identificador e um código secreto ao servidor, que por sua vez autentica, ou seja, valida esses dados, e em seguida garante ou não acesso ao cliente.

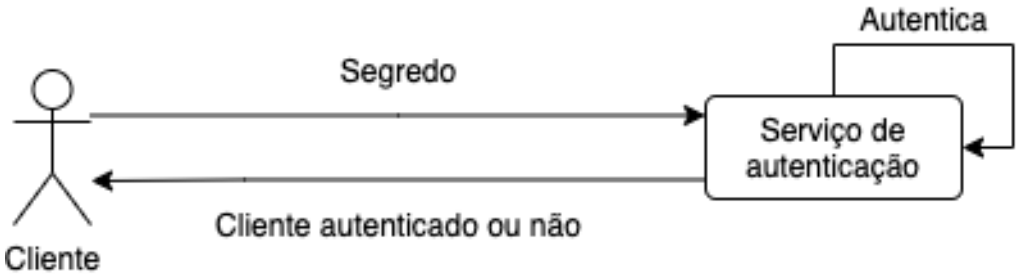

Figura 1.1: Diagrama de autenticação padrão.

Uma das situações recorrentes com essa classe de métodos de autenticação é um intruso acessar os dados de autenticação. Isso normalmente é feito "ouvindo" o canal de comunicação entre cliente e servidor de autenticação para capturar dados e, em seguida, realizando ataques fazendo-se passar pelo cliente. Esse tipo de ataque, conhecido como *man-in-the-middle* (ELAKRAT; JUNG, 2018) é muito comum, e deve ser evitado.

Uma outra situação que merece atenção é quanto ao armazenamento dos códigos secretos utilizados na autenticação. Geralmente esses códigos são armazenados em bases de dados no servidor de autenticação. Esse procedimento tem se mostrado ineficaz, pois não é incomum ocorrerem falhas de segurança em servidores de autenticação, seguidas de vazamento de dados, tornando assim não somente o sistema vulnerável, mas também comprometendo a privacidade dos usuários do sistema.

Para tratar essa situação, algumas soluções, como, por exemplo, o *hashing* (AL-KUWARI; DAVENPORT; BRADFORD, 2011) de senhas antes de armazená-las em um servidor, são adotadas. *Hashing* consiste em aplicar uma função criptográfica de *hash* (*cryptographic hash function* (*CHF*), em inglês) ao dado para mapear esse dado em *bytes* criptografados e irreversíveis.

O *hashing*, entretanto, também apresenta falhas, visto que um observador da comunicação também pode conseguir acesso às informações de autenticação e realizar ataques de força bruta (DAVE, 2013), fazendo-se passar pelo usuário. Ataques de força bruta consistem em utilizar o método tentativa e erro para descobrir um segredo.

Alguns métodos de autenticação, como, por exemplo, os implementados por protocolos *PAKE*, propõem uma abordagem diferenciada para garantir autenticação mais segura. Essa classe de protocolos de autenticação propõe um método que não compartilha dados sensíveis através da comunicação cliente-servidor, mitigando assim os problemas citados anteriormente e aumentando a segurança do processo de autenticação e dos dados de acesso.

Esse trabalho apresenta duas implementações de sistemas autenticadores utilizando protocolos de autenticação *PAKE* e faz uma análise comparativa quali-quantitativa desses protocolos. Essa análise visa oferecer meios de facilitar a escolha da adoção desses protocolos de autenticação para aplicações, serviços e demais software.

1.3 Metodologia

Esse trabalho tem como propósito apresentar uma análise comparativa quali-quantitativa de dois protocolos de autenticações existentes. Para isso, pretende-se utilizar a implementação de ambos os protocolos.

A linguagem escolhida para a implementação de ambos os protocolos foi a *Golang*. A escolha dessa linguagem foi baseada na existência de implementações de referências de ambos os protocolos nessa linguagem, além da facilidade de se realizar testes e análises de desempenho nessa linguagem.

Para o versionamento de código, foi selecionado o *git* como ferramenta, utilizando o *Github*

como plataforma e servidor de armazenamento do repositório. Essa escolha foi baseada em preferência e costume com a ferramenta e a plataforma.

O tipo de pesquisa que mais se encaixa com esse estudo é a pesquisa descritiva, onde se descrevem as características de um fenômeno e, nesse caso, utiliza-se de técnicas de pesquisa, teste e análise comparativa de desempenho para chegar ao objetivo desejado.

Além disso, esse trabalho conta com uma abordagem mista quali-quantitativa, através da qual são apresentados dados baseados em métodos matemáticos, característica da abordagem quantitativa, assim como dados subjetivos, definidos pela abordagem qualitativa.

Diversos métodos podem ser utilizados em um trabalho de pesquisa. No caso desse trabalho proposto, o método utilizado é o hipotético-dedutivo, onde serão sugeridas hipóteses que venham a indicar possíveis soluções para os problemas encontrados nos sistemas de autenticação atuais.

Capítulo 2

Fundamentação Teórica

A área de segurança da informação é bem extensa e complexa. Esse capítulo apresenta definições de diversos conceitos citados nos capítulos seguintes.

2.1 Segurança da Informação e Autenticação

Informação pode ser definida como *"um conjunto de conhecimentos acumulados sobre certo tema por meio de pesquisa ou instrução."* (MELHORAMENTOS, 2021). Assim, a informação pode se apresentar em diversos formatos, como escrita em papel, exibida em filmes ou vídeos, através de conversas, ou armazenada em formato digital (R; J, 2013). A informação é um bem, do indivíduo ou da empresa, com ou sem valor monetário definido.

Em sistemas computacionais, informação é comumente conhecida como dados. Nesse trabalho é usado o termo dados para referenciar informação em meio digital.

Para garantir a continuidade de negócio e limitar o impacto de incidentes de segurança, existe a segurança da informação. Segurança da informação consiste em técnicas aplicadas para proteção de dados para garantir que somente entidades autorizadas consigam acessar dados sigilosos. A proteção e gerência de acesso a dados podem ser feitas através da comprovação da identidade e do nível de autorização do solicitante do acesso. A primeira parte dessa comprovação é feita através da autenticação.

A autenticação permite verificar a autenticidade do solicitante. Segundo Wu (WU, 1998),

pode-se classificar os tipos de autenticação em três categorias gerais:

- algo que o usuário sabe (AUS): senhas, números de identificação pessoal (PINs), chave segredo etc.;
- algo que o usuário é (AUE): biometria, varredura de retina, identificação por voz etc.;
- algo que o usuário tem (AUT): cartão de identificação, código eletrônico, cartões inteligentes etc.

Dentre esses, os mais utilizados são os da classe AUS, que são o foco desse trabalho.

2.1.1 Autenticação AUS

A autenticação AUS baseia-se em algo que o usuário sabe, ou simplesmente segredo.

Há vários métodos de autenticação AUS, como, por exemplo, autenticação baseada em *token*, chave pública, chave simétrica, autenticação via SMS e autenticação baseada em senhas e/ou *PINs*. Em sistemas digitais, é muito comum a adoção da autenticação baseada em senhas e/ou *PINs*. Esse trabalho foca nesse subconjunto de métodos de autenticação AUS.

A autenticação baseada em senhas ou PINs, ou segredo, funciona da seguinte forma: o usuário insere o segredo, provando que tem conhecimento do segredo. O serviço de autenticação é capaz então de autenticar o usuário, verificando se o segredo é válido e condiz com o esperado. Em seguida, o serviço de autenticação retorna ao cliente se o usuário deve ter acesso à informação.

A Fig. 2.1 ilustra esse método de autenticação.

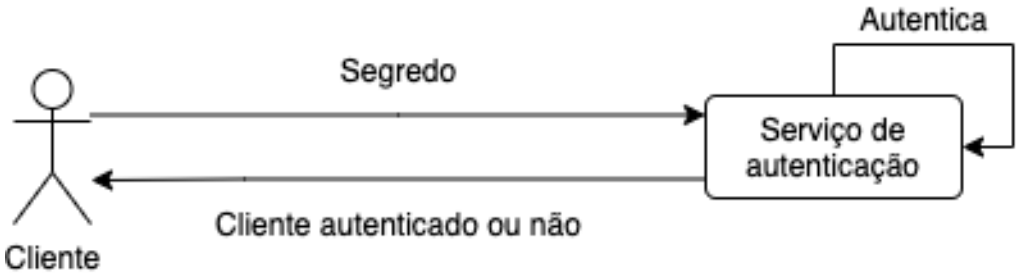

Figura 2.1: Diagrama do funcionamento de um processo de autenticação.

O funcionamento desse método de autenticação é bem abrangente, e, portanto há diversas formas de implementá-lo. Algumas dessas formas são mais seguras, implementando técnicas específicas para aumentar a segurança da comunicação e mitigar uma possível interferência externa ou ataque cibernético.

2.2 Ataques cibernéticos

Ataques cibernéticos são uma ou mais ações ofensivas que têm como alvo sistemas de informação digitais. Esses ataques podem ser realizados para causar dano a sistemas de informação, para obter acesso a dados sensíveis, para obter vantagem monetária, entre outros.

Para conseguir acesso à dados sensíveis, um agressor precisa encontrar um jeito de se passar por um usuário que tem acesso aos dados ou burlar a autenticação de alguma forma.

Alguns métodos de ataque cibernético que visam obter acesso à dados sensíveis são *backdoor*, *eavesdropping*, ataque de força bruta, ataque de pré-computação e ataque de dicionário.

2.2.1 *Backdoor*

O ataque cibernético *backdoor* consiste em contornar o serviço de autenticação, obtendo acesso direto aos dados (WYSOPAL; ENG; SHIELDS, 2010). Um *backdoor* nada mais é do que um acesso direto ao serviço sem passar pelo processo de autenticação. Essa *backdoor* muitas vezes é implementada propositalmente com o objetivo de facilitar o teste ou acesso ao dado em ambiente de desenvolvimento, mas pode acabar por ser esquecida lá.

Um agente malicioso, tendo noção da existência dessa brecha de segurança, pode aproveitá-la para obter acesso à dados sigilosos de maneira indevida.

2.2.2 *Eavesdropping*

Eavesdropping consiste em ouvir uma comunicação privada de computadores, geralmente o par cliente-servidor. Ou seja, um terceiro monitora a comunicação e obtém acesso a informações sensíveis. Esse método de ataque é especialmente eficiente em comunicações não criptografadas,

onde os dados são transferidos de maneira limpa.

Atualmente, com o grande aumento do uso do protocolo *HTTPS* (W3TECHS, s.d.) para comunicação na Internet, os ataques *eavesdropping* têm sido evitados (FOUNDATION, 2021), pois é adotada criptografia nesse tipo de comunicação. Entretanto, o risco de *eavesdropping* ainda é muito presente em outros tipos de comunicação em rede.

2.2.3 Ataque *man-in-the-middle*

Ataques *man-in-the-middle* (*MITM*) consistem em qualquer ataque em que um adversário encontra uma forma de acessar redes de comunicação e se insere na comunicação entre cliente e servidor (ELAKRAT; JUNG, 2018). Ao ser inserido como intermediário na comunicação, o adversário é capaz de enviar informações falsas se passando pelo cliente e pelo servidor, além de obter informações confidencial da comunicação, como chaves secretas e segredos.

Esses tipos de ataques podem ser muito difíceis de serem realizados, conquanto que técnicas adequadas sejam aplicadas na comunicação para protegê-la.

2.2.4 Ataque de força bruta

No ataque de força bruta, o atacante testa diversos segredos contra o servidor na esperança de descobrir o segredo correto (DAVE, 2013).

A eficácia desse método depende da robustez do segredo a ser desvendado. Por exemplo, um segredo com quatro caracteres numéricos é muito mais fácil de ser desvendado do que um segredo de quatro caracteres alfanuméricos distintos entre maiúsculos e minúsculos. Para o primeiro segredo, o total de combinações possíveis é $10^4 = 10.000$, enquanto o total de possibilidades é $62^4 = 14.776.336$ para o segundo segredo.

Nesse método o atacante assume que o segredo é pequeno e de fácil descoberta, para que se torne computacionalmente viável realizar esse tipo de ataque.

2.2.5 Ataque de dicionário

Ataque de dicionário é um tipo específico de ataque de força bruta com o qual o atacante utiliza-se de um dicionário de segredos comuns ou de segredos vazados anteriormente para realizar o ataque (NAM et al., 2009). O atacante assume, assim, que o segredo do usuário é comum, para ser facilmente lembrado, e testa contra as senhas do seu dicionário, com o objetivo de desvendar o segredo correto.

2.2.6 Ataque de pré-computação

Pré-computação consiste em realizar uma computação inicial, antes do tempo de execução do algoritmo, e gerar uma tabela de dados, ou tabela pré-computada, que pode ser usada como consulta para a execução do algoritmo. Dessa forma, ao executar o algoritmo, é possível evitar a repetição de uma computação intensiva que não depende da entrada do algoritmo.

A pré-computação pode ser utilizada em ataques para reduzir bastante o tempo necessário para se decifrar uma chave secreta, ou desvendar o segredo de um usuário. Ataques que se utilizam de tabelas pré-computadas denominam-se ataque de pré-computação.

2.3 Tornando a comunicação segura

Como pôde ser observado, existem diversos métodos de ataques a redes de computadores. Por isso, é necessário adotar medidas para mitigar e evitar esses tipos de ataque. O ataque de *eavesdropping*, como dito na seção 2.2.2, pode ser mitigado utilizando criptografia.

Criptografia, basicamente, consiste na construção de protocolos que permitam a execução de alguma tarefa mesmo na presença de um adversário. Um exemplo de tarefa é permitir uma comunicação segura e privada entre usuários de uma rede insegura. (CORON, 2006). Isso é feito através do obscurecimento da mensagem em um lado da comunicação para tornar a mensagem ilegível para quem não possui o segredo.

Um dos métodos mais simples, mas que exemplifica bem o funcionamento da criptografia, é a criptografia simétrica. Nesse método, os participantes em cada ponta da comunicação devem

possuir um segredo K para cifrar e decifrar a mensagem. O atacante, sem o segredo K, não é capaz interpretar a mensagem cifrada.

Um método especificamente interessante de criptografia para esse trabalho é o chamado *cryptographic hash function* (*CHF*) (AL-KUWARI; DAVENPORT; BRADFORD, 2011). *CHF* é um algoritmo que mapeia dados de tamanho arbitrário para um vetor de bits de tamanho fixo, chamado de valor *hash*.

Em particular, esse é interessante porque é uma função de via única, ou seja, é computacionalmente irreversível. Dessa forma, é possível que um par comunicador se autentique utilizando o valor *hash* previamente combinado, onde somente um sabe o segredo original.

2.3.1 Métodos de autenticação

Para garantir a integridade da comunicação e a segurança dos sistemas de informação, é essencial a definição de técnicas e métodos para mitigar ou mesmo evitar os diversos tipos de ataques cibernéticos existentes. Esses métodos podem ser implementados de diversos modos.

Em alguns casos, o segredo é enviado ao servidor como texto limpo (*clear text*). Como não é difícil de perceber, esse método provê brechas de segurança que permitem ataques de *eavesdropping* ao canal de comunicação entre o par cliente-servidor, com consequente obtenção da informação autenticadora.

Para tentar tornar o canal de comunicação um pouco menos inseguro, geralmente os dados autenticadores, em sua maioria compostos do identificador e segredo, passam pelo processo de *hashing* (CHF). Ao realizar o *hashing* do segredo, garante-se que o servidor de autenticação não armazena o segredo, somente o valor *hash* dele, evitando problemas com vazamento de dados e o tráfego do segredo em texto limpo pelos meios de comunicação vulneráveis a ataques.

Ao realizar o *hashing*, o dado é transformado em um conjunto de caracteres que não podem ser revertidos para sua forma original. Uma autenticação direta com dado autenticador em *hash* funciona da seguinte forma: o usuário entra informa o segredo e o cliente realiza o processo de *hashing* com esse dado. O cliente envia o valor *hash* gerado para o servidor. Ao receber o valor *hash*, o servidor compara o valor *hash* enviado pelo cliente com o armazenado no servidor. A

Fig. 2.2 ilustra esse processo.

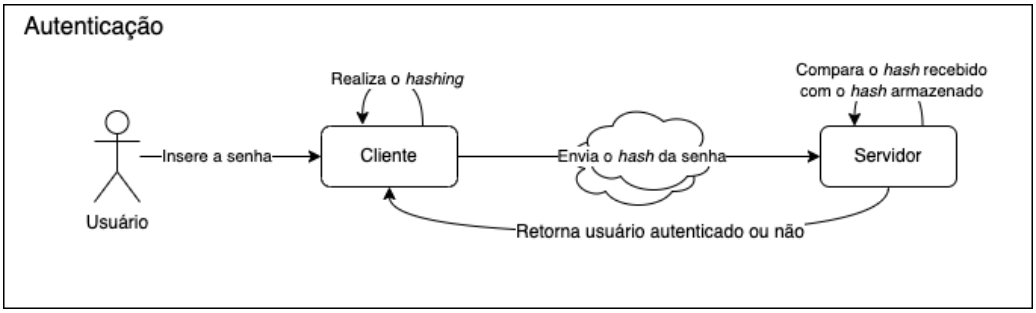

Figura 2.2: Diagrama de autenticação em servidores com *hashing*.

Com esse método, consegue-se proteger a informação de autenticação pois, mesmo que haja um terceiro ouvindo a comunicação, o valor *hash* do segredo não lhe fornece qualquer tipo de informação relevante.

2.4 Protocolos de Autenticação

Em sistemas digitais, protocolos de comunicação são definidos como um conjunto de regras que permitem que dois ou mais dispositivos possam transmitir informação via um meio físico. Protocolos de autenticação são um tipo específico de protocolo de comunicação que têm o propósito específico de permitir a transferência de dados de autenticação entre duas entidades. Desse modo, a construção e o estudo de protocolos de autenticação são crucial ao se abordar segurança da informação.

Existem diversos protocolos de autenticação disponíveis, cada um com seus casos de uso, vantagens e desvantagens. Alguns deles são: *Password Authentication Protocol* (PAP), *Extensible Authentication Protocol* (EAP), *Kerberos* e *Password-authenticated key agreement* (PAKE).

2.4.1 *Password Authentication Protocol* (PAP)

Um dos métodos mais antigos de autenticação é o *Password Authentication Protocol* (SIMPSON, 1992). A autenticação utilizando esse método é extremamente simples. O cliente envia um pacote de dados contendo as credenciais de acesso e o servidor responde com uma mensagem informando se o cliente está autenticado ou não.

Esse método de autenticação é extremamente inseguro, pois o pacote contendo as credenciais é enviado em texto livre pela rede. Assim, *PAP* é muito suscetível a ataques de *eavesdropping* (Seção 2.2.2). O método também é suscetível a ataques *man-in-the-middle*, através do qual um agente malicioso age como um interceptador da comunicação alterando os dados estabelecidos entre cliente-servidor e sucessivamente se passando tanto como o cliente quanto como o servidor.

2.4.2 *Extensible Authentication Protocol* (EAP)

O Protocolo de Autenticação Extensível (*Extensible Authentication Protocol* (*EAP*), em inglês) é um protocolo muito utilizado atualmente, principalmente para autenticação de redes Wi-Fi. O protocolo permite o transporte e a utilização de parâmetros gerados pelos métodos *EAP*. Basicamente, *EAP* fornece regras para funções comuns de autenticação e os métodos definem como é feita a negociação de autenticação entre os pares.

Alguns dos métodos *EAP* existentes são: *EAP-MD5* (VOLLBRECHT et al., 2004), *EAP-TLS* (SIMON; HURST; ABOBA, 2008), *EAP-TTLS* (FUNK; BLAKE-WILSON, 2008), *EAP-FAST* (SALOWEY et al., 2007), *EAP-LEAP* e *EAP-IKEv2* (BERSANI et al., 2008).

Como o EAP somente fornece um padrão para os métodos de autenticação, por si só não apresenta vulnerabilidades. Entretanto, as diversas implementações existentes, algumas citadas acima, apresentam algum tipo de vulnerabilidade. Por exemplo, o *EAP-LEAP* sofre de vulnerabilidade a ataques de dicionário, as implementações *EAP-TLS*, *EAP-TTLS* e *EAP-FAST* são suscetíveis a ataques *man-in-the-middle*. Além disso, os métodos *EAP-TLS* e *EAP-IKE2* podem utilizar o *PAP*, tornando esses vulneráveis a ataques de *eavesdropping* (SOTILLO, 2007).

2.4.3 *Kerberos*

Kerberos é um protocolo de autenticação desenhado com o propósito de evitar ataques de *eavesdropping*. Para isso, o *Kerberos* adota *tickets*, que permitem que módulos se comuniquem por uma rede não segura provando a sua identidade ao outro de maneira segura.

Esse método de autenticação prevê uma autenticação mútua, ou seja, ambos cliente e servidor verificam a identidade do outro. Dessa forma, é possível mitigar ataques *man-in-the-middle*.

Além disso, toda a comunicação da autenticação é criptografada, evitando o ataque de *eavesdropping*. Entretanto, *Kerberos* apresenta algumas limitações.

Esse protocolo é suscetível à ataques de dicionário e ataque de força bruta. Além disso, várias implementações do *Kerberos* contam com o uso do *DES* como algoritmo de geração de chave simétrica que, por sua vez, gera chaves fracas (NEUMAN; TS'O, 1994). O *DES* é um algoritmo de geração de chaves simétricas desenvolvido em meados de 1970. Entretanto, estudos mostraram que esse algoritmo apresenta pontos fracos quando submetidos a ataques de força bruta (DIFFIE; HELLMAN, 1977).

2.4.4 *Password-authenticated key agreement* (PAKE)

Para entender como *PAKE* funciona, será apresentado um cenário. Duas pessoas, Roberto e Ana, desejam usar um segredo, que ambos conhecem, para estabelecerem uma comunicação segura. Entretanto, nem Ana nem Roberto desejam revelar seu segredo um para o outro pois, ao fazê-lo, seu segredo estaria comprometido. Nesse caso, um terceiro seria capaz de conseguir o segredo, situação não ideal nesse cenário.

Para contornar essa situação, uma proposta é não somente autenticar Ana e Roberto, mas também criar um canal seguro de comunicação entre os dois. E, para isso, eles precisariam de uma *chave de sessão*, que fosse forte e compartilhada.

Esse método de realizar a autenticação é chamado *PAKE* (HAO; OORSCHOT, 2021). No *PAKE*, uma chave compartilhada e forte é gerada utilizando uma chave secreta, sem que essa chave seja compartilhada na comunicação. Para esse caso, a chave secreta pode ser uma chave fraca, sem causar riscos de segurança à autenticação.

Com protocolos *PAKE*, tem-se uma ótima defesa contra ataques de força bruta, ataque de dicionário e ataques *man-in-the-middle*. Um intruso não é capaz de manipular a comunicação sem saber o segredo e, se ele deseja adivinhar o segredo, ele só consegue fazer uma tentativa a cada troca de mensagens entre o par comunicante.

Além disso, os protocolos *PAKEs* garantem *forward secrecy* (HAO; RYAN, 2011). *Forward secrecy* é a segurança de que as *chaves de sessão* não serão comprometidas, mesmo que segredos

com alta longevidade usados na comunicação sejam comprometidos. *Forward secrecy* protege as comunicações passadas contra comprometimento de chaves e segredos no futuro.

De acordo com Hao, há dois tipos de protocolos *PAKE*: *balanced PAKE* (bPAKE) e *augmented PAKE* (aPAKE) (HAO; RYAN, 2011).

Nos protocolos do tipo *bPAKE*, assume-se que o par comunicante compartilha uma única senha secreta para negociar e autenticar a chave compartilhada (HAO; RYAN, 2011).

No caso de protocolos do tipo *aPAKE*, assume-se uma situação cliente-servidor, onde o servidor desconhece o segredo. Dessa forma, consegue-se evitar que um intruso que obtivesse acesso aos dados do servidor conseguisse se passar pelo cliente (HAO; RYAN, 2011).

2.5 Conclusão

De forma a mensurar e identificar as vantagens e desvantagens de cada protocolo, tornam-se necessárias análises, com o objetivo de se obter a melhor solução para um caso de uso específico. No Capítulo 3, será apresentado um estudo de protocolos do tipo *PAKE*, discorrendo sobre a escolha de dois protocolos, *CPace* e o *OPAQUE*, a serem detalhados e comparados com mais profundidade nesse trabalho.

Capítulo 3

Tipos de protocolos *PAKE*

Existe uma variedade de protocolos *PAKE*. Para se realizar um estudo comparativo, foram escolhidos dois protocolos, um de cada classe *PAKE*. Os protocolos escolhidos foram o *CPace*, da classe *bPAKE* e o *OPAQUE* da classe *aPAKE*.

Essa escolha foi deliberada. Em 2019, o *Crypto Forum Research Group* (*CFRG*) iniciou um processo seletivo de protocolos *PAKE*, denominado *CFRG PAKE Selection Process 2019*.

O *CFRG* é um fórum criado para discussão e revisão de usos de mecanismos criptográficos (IRTF, 2014). O *CFRG PAKE Selection Process 2019* teve como objetivo estimular o desenvolvimento de protocolos do tipo *PAKE* e selecionar, ao final, dois protocolos como recomendação da *Internet Engineering Task Force* (*IETF*) (SULLIVAN, 2019). Ao final do processo seletivo, os protocolos selecionados foram o *CPace* e o *OPAQUE*.

Por serem os protocolos *PAKE* recomendados pela *IETF* e especialmente por serem protocolos de classe *PAKEs* diferentes, viu-se como relevante a análise comparativa desses protocolos.

3.1 *Balanced PAKE*

O protocolo *PAKE* assume que as duas partes comunicantes contêm uma informação secreta simétrica (HAO; RYAN, 2011). Ou seja, considerando uma comunicação cliente-servidor, tanto o cliente quanto o servidor devem conhecer o segredo.

Os requerimentos típicos (HAO; RYAN, 2011) para que um protocolo possa ser considerado

bPAKE são:

1. resistência a ataques de dicionário *offiline* —nenhuma informação de verificação do segredo deve ser vazada à um adversário;

2. resistência a ataques de dicionário *online* —limitar ataques *online* a somente uma tentativa de senha por execução do protocolo;

3. segurança de chave de sessão —previne que uma chave de sessão comprometida comprometa outras sessões;

4. *forward secrecy* —produz uma chave de sessão que continua segura, mesmo que o segredo seja descoberto no futuro.

O par comunicante, com acesso ao segredo, consegue estabelecer uma comunicação segura. Entretanto, como citado na seção 2.3.1, existem problemas associados ao armazenamento de segredo no servidor, tornando-o suscetível à vazamento de dados. Mesmo que o servidor armazene o segredo em forma de *hash*, um intruso ainda é capaz de obter alguma informação valiosa desse vazamento.

A grande vantagem dos protocolos *bPAKE* está atribuída a sua característica não opinativa, ou seja, não há um único tipo de comunicação nessa classe de protocolos. Com *bPAKE*, é possível estabelecer comunicações seguras e autenticadas entre cliente e servidor ou até entre cliente e cliente (HAO; RYAN, 2011). Além disso, tendem a ser mais leves e menos complexos do que a alternativa.

Os principais protocolos *bPAKE* são o *CPace* (ABDALLA; HAASE; HESSE, 2022), o *Encrypted Key Exchange (EKE)* (BELLOVIN; MERRITT, 1992) e o *Strong password-only authenticated key exchange (SPEKE)* (JABLON, 1996a).

O *EKE* é um protocolo de autenticação da classe *bPAKE* que promete segurança contra ataques de dicionário. *EKE* foi um marco no avanço dos protocolos *PAKEs* (HAO; RYAN, 2011). *EKE* apresenta um método de autenticação que pode ser facilmente modificado. Sendo assim, diversos protocolos *EKE* foram desenvolvidos. Entretanto, estudos demonstraram que, apesar da grande variedade, esses protocolos são suscetível à ataques de dicionário. Em seu

artigo, Tang e Chen descrevem as fraquezas dos protocolos *EKE-U* e *EKE-M* (TANG; CHEN, 2005). Jablon e David foram capazes de encontrar vulnerabilidades no protocolo *DH-EKE* (JABLON, 1996b). Patel oferece algoritmos de ataque força bruta utilizando teoria de números que se aproveitam das vulnerabilidades dos protocolos *RSA-EKE*, *DH-EKE* e *ElGamal-EKE*.

O *SPEKE* é um protocolo de autenticação *bPAKE* semelhante ao *EKE*, mas que utiliza restrições diferentes. As restrições do *SPEKE* são geradas a partir do segredo, enquanto as do *EKE* são fixas. Essas restrições geradas apresentam riscos. Ao realizar um ataque de força bruta, um adversário é capaz de testar múltiplos segredos em uma única tentativa (HAO; RYAN, 2011).

Tendo em vista as claras vulnerabilidades de ambos os protocolos, será focado o *CPace* como protocolo da classe *bPAKE* de referência.

3.1.1 CPace

O *CPace* é um protocolo de autenticação do tipo *bPAKE* onde ambas as partes comunicantes compartilham um segredo para derivar uma chave compartilhada, sem expor o segredo para ataque de dicionário *offline* (ABDALLA; HAASE; HESSE, 2022).

O *CPace* foi especificamente desenhado com o foco em eficiência em dispositivos limitados, bem como na mitigação de problemas futuros com adversários utilizando computação quântica para quebra do logaritmo de curvas elípticas (ABDALLA; HAASE; HESSE, 2022).

3.1.1.1 Fluxo do Protocolo

O *CPace* é um protocolo que conta com duas rodadas de autenticação, quando são feitas duas trocas de mensagens, aqui denominadas *flows*, entre os pares. Uma rodada pode ser definida como um passo que todos os participantes podem executar sem depender do outro (HAO; OORSCHOT, 2021).

Na Fig. 3.1 é apresentado um esquema do método de autenticação do *CPace*. Nesse diagrama, é assumido o par comunicante como sendo **A** e **B**. Essa escolha foi feita de maneira deliberada, pois considera-se que o par comunicante pode ser tanto cliente-servidor quanto

cliente-cliente.

O funcionamento é relativamente simples. Inicialmente, A computa e envia a B sua chave pública (Ya) e, opcionalmente, seu identificador (ADa). B faz o mesmo, computando sua chave pública (Yb) e enviando a A a chave e, opcionalmente, seu identificador (ADb). Ambos agora podem derivar suas chaves de sessão intermediária ($ISKs$). O par então verifica se as ISK derivadas são as mesmas. Se são as mesmas, a comunicação está devidamente protegida e autenticada (ABDALLA; HAASE; HESSE, 2022).

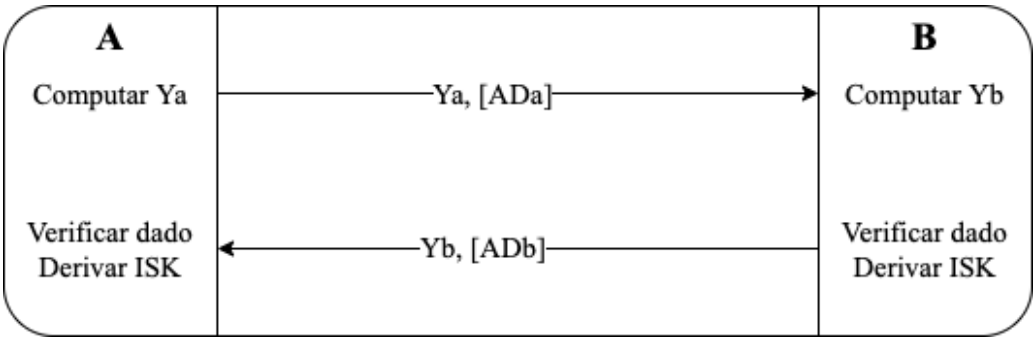

Figura 3.1: Fluxo de comunicação do protocolo *CPace*.

3.2 *Augmented PAKE*

Os protocolos *PAKE* do tipo *augmented* contêm, geralmente, um esquema mais focado para o uso cliente-servidor. Como melhoria de protocolos do tipo *bPAKE*, essa classe de protocolos adicionam mais um requerimento, a resistência a comprometimento de servidor (HAO; RYAN, 2011). Ou seja, um adversário não deve ser capaz de assumir a identidade de um cliente em caso de vazamento de dados do servidor. Esses esquemas tendem a ser mais complexos que os *bPAKE*, tanto em questão de implementação quanto de computação, mas fornecem mais segurança devido ao requisito adicional.

Os principais protocolos do tipo *aPAKE* são o *Secure Remote Password Protocol (SRP)* (WU, 1998), o *Augmented-EKE (AEKE)* (BELLOVIN; MERRITT, 1993) e o *OPAQUE* (BOURDREZ et al., 2022).

O *SRP* é um protocolo do tipo *aPAKE* que apresenta resistência à ataques de dicionários, além de *forward secrecy*. O *SRP* também garante que os segredos também são armazenados de

maneira criptografadas, o que adiciona uma segurança a mais em relação a outros protocolos (WU, 1998). O *SRP* é usado atualmente em algumas aplicações comerciais, como *1Password*, *iCloud Keychain*, entre outros (SHERMAN et al., 2020). Apesar da grande segurança do *SRP*, patentes impedem o livre uso desse protocolo. A patente do *SRP* foi realizada em Setembro de 2014, aceita em Dezembro de 2016 e só expirará em Dezembro de 2034.

O *AEKE* é uma extensão do protocolo *EKE* para torná-lo *aPAKE*. Esse protocolo preserva as principais propriedades de segurança do *EKE*. Entretanto, esse protocolo apresenta algumas vulnerabilidades. Um adversário que conseguir acesso ao arquivo de segredos cifrados é capaz de se camuflar como servidor para o cliente, além de conseguir realizar ataques de dicionário contra os segredos cifrados (BELLOVIN; MERRITT, 1993). Portanto, esse protocolo é vulnerável contra acesso indevido ao servidor.

Por conta da patente do *SRP* limitar o uso desse protocolo, e por conta das vulnerabilidades apresentadas pelo *AEKE*, o protocolo *OPAQUE* será focado como protocolo da classe *aPAKE* de referência.

3.2.1 OPAQUE

O *OPAQUE* é um protocolo do tipo *aPAKE* que também é seguro contra ataques de pré-computação. Nesse protocolo, o segredo é totalmente opaco ao servidor. Ou seja, o servidor nunca tem conhecimento do segredo, até mesmo no processo de registro. Além disso, *OPAQUE* é extensivo, permitindo que o cliente recupere informações arbitrárias do servidor somente utilizando o seu segredo. (BOURDREZ et al., 2022).

O *OPAQUE* é composto de três funcionalidades: uma função pseudo-aleatória esquecida (*oblivious pseudo random function* (*OPRF*), em inglês), um mecanismo de recuperação de senha e um protocolo de troca de mensagens autenticada (*authenticated key exchange* (*AKE*), em inglês) (BOURDREZ et al., 2022). *AKE* consiste, basicamente, na troca de chaves de sessão e autenticação das partes comunicantes em um protocolo de troca de chaves (DIFFIE; OORSCHOT; WIENER, 1992).

3.2.1.1 Fluxo do Protocolo

Assim como o *CPace*, o *OPAQUE* conta com duas rodadas, tanto no passo de cadastro quanto no de autenticação. O passo de cadastro é necessário pois o segredo nunca é transmitido na comunicação. Tanto o passo de cadastro quanto o de autenticação necessitam de três trocas de mensagens (*flows*) entre os pares para serem realizados.

No passo de cadastro, é feita a computação de dados que devem ser utilizados para recuperar as credenciais de autenticação no servidor. A recuperação dessa credencial de autenticação só pode ser feita com o conhecimento do segredo do cliente (BOURDREZ et al., 2022).

Durante o passo de autenticação é feita a recuperação dessas credenciais utilizando o segredo do cliente e, em seguida, essas credenciais são usadas como dados de entrada do protocolo de comunicação *AKE*, para que seja realizada a troca de mensagens (BOURDREZ et al., 2022).

3.2.1.1.1 Fluxo de Cadastro

O fluxo de cadastro no *OPAQUE* conta com três mensagens. Na primeira mensagem, o cliente computa um valor escalar da *OPRF* denominado *blind* e um segundo valor *request*, utilizando o seu segredo. A *request* é então enviada ao servidor na requisição de cadastro. O servidor, de posse da informação da *request*, gera uma resposta de cadastro, que é enviada ao cliente. O cliente, recebendo a resposta de cadastro, computa suas credenciais, denominadas *record*, e as envia ao servidor. O servidor então realiza o último passo, que consiste em armazenar as credenciais em um banco de dados, junto com o identificador do cliente para futura autenticação. A Fig. 3.2 ilustra o fluxo de cadastro no protocolo *OPAQUE*.

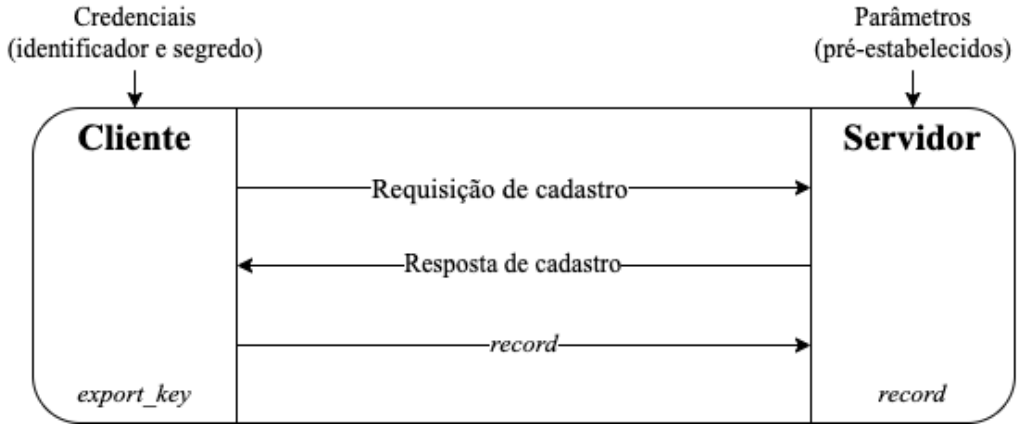

Figura 3.2: Fluxo de cadastro do protocolo *OPAQUE*.

3.2.1.1.2 Fluxo de Autenticação

O fluxo de autenticação, por sua vez, é definido por três trocas de mensagens: *ke1*, *ke2* e *ke3*, onde *ke* significa troca de chave (*key exchange*, em inglês). Essa parte do fluxo do protocolo conta com o uso do protocolo *AKE* para que a troca de mensagens esteja devidamente protegida de adversários.

As mensagens *ke1* e *ke2* incluem a troca de parte das chaves enviadas pelo cliente ao servidor e do servidor ao cliente, sucessivamente. A mensagem *ke3* provê a autenticação propriamente dita e *forward secrecy*, conforme requisito dos protocolos da classe *PAKE* (BOURDREZ et al., 2022). A Fig. 3.2 ilustra o fluxo de autenticação no protocolo *OPAQUE*.

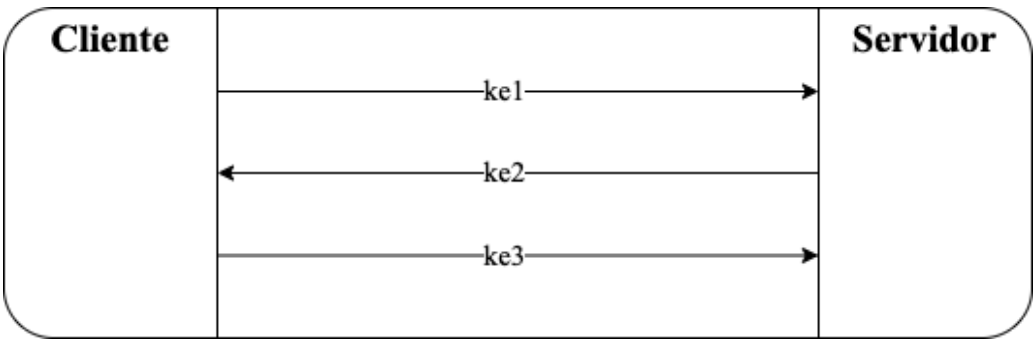

Figura 3.3: Fluxo de autenticação do protocolo *OPAQUE*.

3.3 Conclusão

Esse capítulo apresentou em mais detalhes as classes de protocolo *PAKE* existentes, destacando o *CPace* e o *OPAQUE*. O próximo capítulo trata da arquitetura e funcionamento dos protocolos *CPace* e *OPAQUE*.

Capítulo 4

Arquitetura e Funcionamento

Em relação a outros tipos de protocolos de autenticação, os protocolos do tipo *PAKE* adicionam alguns requisitos para aumentar ainda mais a segurança. Todos os protocolos *PAKE* precisam apresentar os seguintes requisitos: resistência a ataques de dicionário *offiline*, limitar ataques *online* a somente uma tentativa de senha por execução do protocolo, garantia de segurança da chave de sessão e prover *forward secrecy*. Além disso, os protocolos da classe *aPAKE* também devem ser resistentes ao comprometimento do servidor.

O *CPace* e o *OPAQUE*, protocolos de classe *bPAKE* e *aPAKE*, respectivamente, apresentam esses requisitos. Pode-se argumentar que, por apresentar um requisito a mais, o protocolo *OPAQUE* é, consequentemente, melhor que o *CPace*. Entretanto, uma argumentação desse tipo é rasa, pois não considera alguns fatores importantes como desempenho, complexidade de implementação, flexibilidade de uso do protocolo, entre outros.

Para que se possa realizar um estudo comparativo mais aprofundado de ambos os protocolos, é importante medir e avaliar seus desempenhos. Portanto, foi necessária a implementação de uma aplicação que simula uma situação de cadastro e autenticação cliente-servidor para cada um dos protocolos.

Nesse capítulo, são apresentados os requisitos funcionais, não funcionais e a arquitetura da aplicação implementada, assim como dos protocolos *CPace* e *OPAQUE*.

4.1 Aplicação de Teste

Para que seja possível testar ambos os protocolos, a aplicação de teste precisa ser capaz de cadastrar novos usuários, armazenar credenciais de usuários e autenticar usuários cadastrados.

Em uma aplicação real, o uso de um banco de dados seria essencial para o funcionamento da aplicação. Entretanto, a presença uma base de dados adicionaria complexidade que, por consequência, tornaria menos precisos os resultados dos testes. Portanto, a aplicação foi idealizada utilizando a própria memória em tempo de execução, ou seja, as credenciais foram armazenadas em uma variável no servidor.

Apesar dessa simplificação, adotou-se o uso de banco de dados no texto dessa monografia, pois esse é o cenário que mais se adéqua ao mundo real.

Para atingir os requisitos, a aplicação foi pensada como uma aplicação cliente-servidor usual, utilizando o *HTTP* como protocolo de comunicação. A parte do cliente da aplicação inicia os processos de cadastro e autenticação fazendo requisições *POST* ao servidor e a parte do servidor escuta requisições em rotas pré-estabelecidas, armazena dados de credenciais e autentica o usuário.

4.1.1 Rotas

As rotas das requisições foram definidas de acordo com a necessidade do protocolo em questão. O *CPace*, por exemplo, necessita apenas de uma rota de cadastro, `/registration`, enquanto o *OPAQUE* necessita de duas rotas para cadastro, `/registration-init` e `/registration-finalize`.

As tabelas Tab. 4.1 e Tab. 4.2 ilustram as rotas pré-estabelecidas para o *CPace* e para o *OPAQUE*, além de discorrer brevemente sobre suas funções.

Rota	Função
/registration	Cadastra o usuário na base de dados.
/authentication-init	Inicia a autenticação do usuário. Retorna alguns dados para o cliente processar e enviar ao servidor.
/authentication-finalize	Finaliza o processo de autenticação. Retorna se o usuário está devidamente autenticado.

Tabela 4.1: Tabela de rotas pré-estabelecidas do *CPace*

Rota	Função
/registration-init	Inicia o processo de cadastro do usuário. Retorna alguns dados que o cliente deve processar para prosseguir.
/registration-finalize	Finaliza o processo de cadastro do usuário. Retorna se o usuário foi cadastrado.
/auth-init	Inicia a autenticação do usuário. Retorna alguns dados para o cliente processar e enviar ao servidor.
/auth-finalize	Finaliza o processo de autenticação. Retorna se o usuário está devidamente autenticado.

Tabela 4.2: Tabela de rotas pré-estabelecidas do *OPAQUE*

4.2 CPace

4.2.1 Cadastro

Apesar do protocolo *CPace* não contar com uma etapa de cadastro, é necessário que um cadastro ocorra para que a autenticação aconteça. O cadastro foi idealizado como uma simples troca de mensagens entre o cliente e o servidor. O cliente envia o seu identificador e senha por meio da rota /registration, como pode ser observado na Fig 4.1. O servidor armazena os dados recebidos e retorna ao cliente se ele foi devidamente cadastrado.

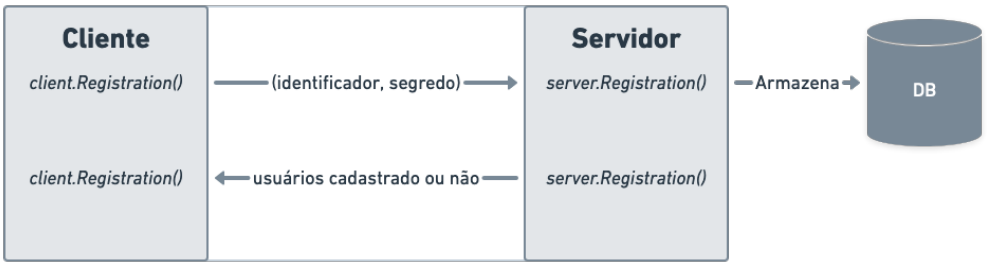

Figura 4.1: Arquitetura de cadastro do protocolo *CPace*.

4.2.2 Autenticação

O processo de autenticação é iniciado pelo cliente interessado em ser autenticado. O cliente cria um contexto `c` com seu identificador `username` e o identificador do servidor `ServerId`. Em seguida, o cliente gera a `msgA`, utilizando `c` e o segredo do usuário `password`. Por último, é feita uma chamada *HTTP* do tipo *POST* à rota `/authentication-init` do servidor, passando como parâmetro de corpo a `msgA`.

O servidor, ao receber `msgA` e o identificador do usuário, busca pelo usuário na base de dados. Se não for encontrado, o usuário não é autenticado. Se for encontrado, o servidor então cria um contexto `c`. Utilizando o contexto `c` gerado, o segredo do usuário e a `msgA`, o servidor gera dois valores: a `msgB`, que deve ser enviada ao cliente e o `keyB`, que é armazenado em uma variável intermediária para uso na finalização da autenticação.

O cliente, recebendo `msgB`, é capaz de gerar sua chave `keyA`. Essa chave é enviada ao servidor através da rota `/authentication-finalize`. Se, `keyA` e `keyB` forem iguais, o cliente e o servidor derivaram a mesma chave e, por consequência, ambos conhecem o segredo. Se as chaves forem diferentes, as informações para gerar a chave não foram iguais e o cliente não deve ser autenticado. A Fig. 4.2 demonstra o processo de autenticação no protocolo *CPace*.

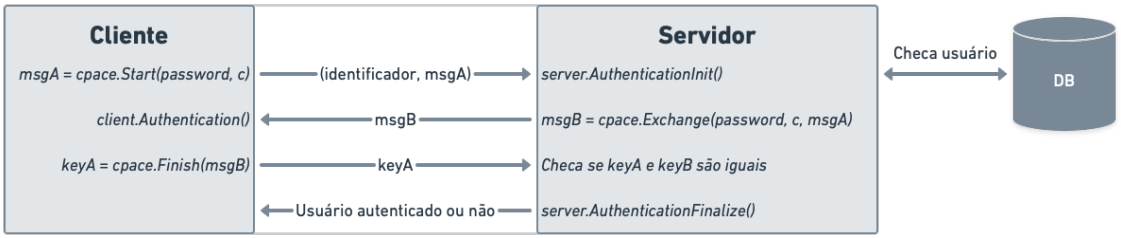

Figura 4.2: Arquitetura de autenticação do protocolo *CPace*.

4.3 OPAQUE

4.3.1 Cadastro

Diferentemente do *CPace*, a definição do *OPAQUE* conta com uma etapa de cadastro.

Na primeira etapa, o usuário que deseja se cadastrar deve informar seu identificador e segredo desejado. O cliente gerará um *RegistrationRequest* `rReq` utilizando o segredo do usuário. O identificador e o `rReq` são enviados ao servidor pela rota `/registration-init` para que o cadastro ocorra.

Ao receber uma requisição de cadastro, o servidor é responsável por gerar um identificador da credencial do usuário `credId`. Além disso, nesse passo o servidor é responsável por gerar uma *seed* para o cliente, ou utilizar uma *seed* pré-existente, que será necessária no processo de recuperação das credenciais no futuro. O servidor, de posse da sua chave *AKE* `pks`, do `credId` gerado, da *seed* e da `rReq`, gera uma *RegistrationResponse* `rRes` que é, em seguida, enviada ao cliente como resposta.

No recebimento da resposta do servidor, o cliente pode continuar para a finalização do cadastro. O cliente gera então um *RegistrationRecord* `rRec` utilizando seu identificador, o `rRes`, e o identificador do servidor. O `rRec` é enviado ao servidor por meio da rota `/registration-finalize` para que suas credenciais sejam devidamente armazenadas na Base de Dados.

De posse do `RegistrationRecord`, o servidor é capaz de criar uma nova entrada para esse usuário na Base de Dados para autenticação futura. Os dados armazenados são o identificador do usuário `ClientIdentity`, o identificador da credencial `credId` e o `RegistrationRecord`. A

Fig. 4.3 ilustra o processo de cadastro no *OPAQUE*.

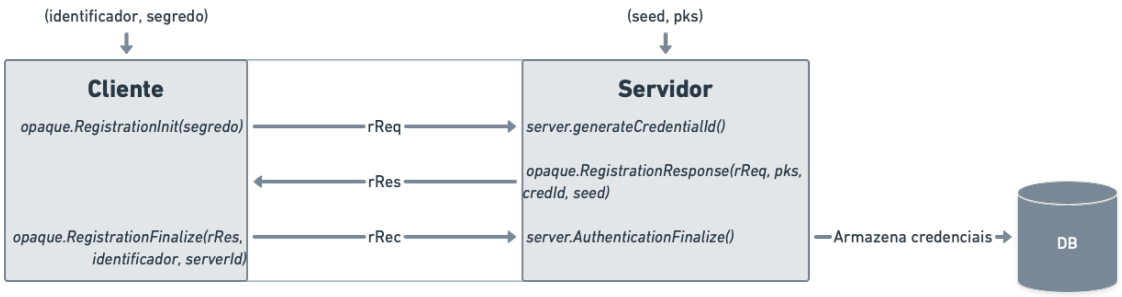

Figura 4.3: Arquitetura de cadastro do protocolo *Opaque*.

4.3.2 Autenticação

Como citado na seção 3.2.1, para o processo de autenticação é utilizado um protocolo de troca de chaves *AKE*. A rodada de autenticação é realizada através de três mensagens: `ke1`, `ke2` e `ke3`. A autenticação do protocolo de troca de chaves é realizada utilizando as credenciais criadas na etapa de cadastro. Ao final da autenticação, o cliente é capaz de provar que conhece o segredo e cliente e servidor concordam em uma chave secreta mútua (BOURDREZ et al., 2022).

A primeira etapa da autenticação é feita pelo cliente ao realizar uma requisição *POST* à rota `/auth-init`. Inicialmente, o cliente gera seus dados de autenticação `authInit` utilizando seu identificador e o segredo. O `authInit` é enviado serializado ao servidor no corpo da requisição de início da etapa de autenticação. Essa etapa é denominada `ke1`.

O servidor, ao receber o `authInit`, desserializa o dado e o utiliza para encontrar o cliente na base de dados. Encontrando o cliente, o servidor é capaz de gerar a segunda troca de mensagens `ke2`, utilizando o `ke1`, as credenciais do cliente encontradas na base de dados, o identificador, chave secreta e chave pública do servidor e a *seed* do *OPRF*. O `ke2` é então serializado e enviado ao cliente como resposta da requisição.

Na etapa seguinte, o cliente é capaz de gerar `ke3` utilizando a resposta do servidor `ke2` desserializada, o identificador do servidor e o identificador do usuário. Em seguida, o cliente gera uma chave de sessão `sessionKey` e a envia juntamente com `ke3` para o servidor pela rota

`/auth-finalize`.

De posse dos dados enviados pelo cliente, o servidor é capaz de finalizar o processo de autenticação. Para isso, o servidor verifica se o `ke3` é válido. Em sendo válido o servidor é então capaz de gerar uma chave de sessão `serverSessionKey` e compará-la à `sessionKey` recebida do cliente. Se forem iguais, o cliente tem conhecimento do segredo e portanto está devidamente autenticado. A Fig. 4.4 ilustra o funcionamento da rodada de autenticação do *OPAQUE*.

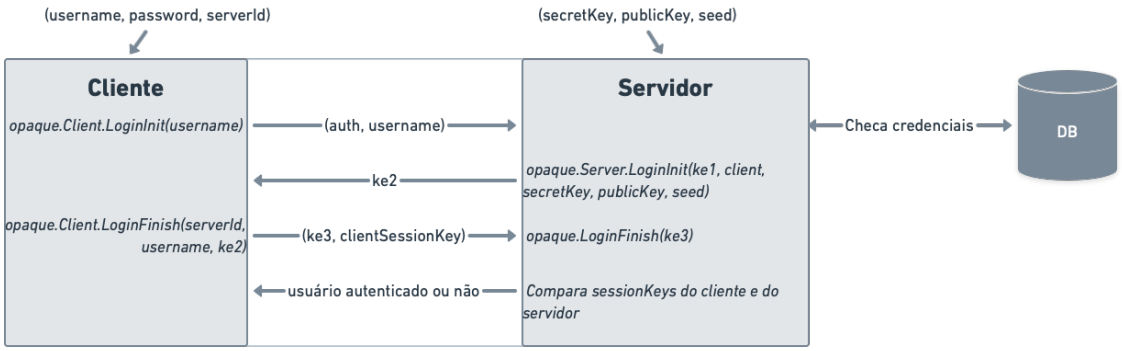

Figura 4.4: Arquitetura de autenticação do protocolo *Opaque*.

4.4 Conclusão

Nesse capítulo foram apresentadas as arquiteturas e requisitos de cada protocolo e da aplicação de teste. No Capítulo 5, serão descritas as tecnologias utilizadas na implementação, as dificuldades encontradas, os resultados obtidos e uma análise crítica de ambos os protocolos.

Capítulo 5

Implementações e Resultados

Para realizar os testes de desempenho dos protocolos, foram implementadas duas aplicações com as arquiteturas descritas no Capítulo 4. Esse Capítulo descreve detalhes de implementação de cada aplicação, assim como apresenta os resultados obtidos nos testes.

5.1 Implementações

Para realizar as implementações das aplicações, a linguagem escolhida foi a *Golang* (GO, s.d.b). *Golang* é uma linguagem desenvolvida pela Google®. Ela é uma linguagem estaticamente tipada, que contém segurança de memória (GO, 2014), coleta de lixo (GO, s.d.a), tipagem estruturada (GO, s.d.a) e concorrência do tipo CSP (GO, s.d.a).

A decisão por essa linguagem foi tomada por dois principais motivos: a existência de bibliotecas de ambos os protocolos nessa linguagem e o grande suporte à testes e análise de desempenho de algoritmos.

O *OPAQUE* contém implementações em diversas linguagens como a implementação de referência em *Sage* (*Python*) (WOOD; BOURDREZ; LEWI, 2022), em *Rust* (LEWI et al., 2022), *Golang* (BOURDREZ, 2020), entre outras. O *CPace* apresenta a implementação de referência em *Sage* (HAASE; HESSE, 2022), implementações em *Rust* (DENIS; PESTANO, 2021) e em *GoLang* (VALSORDA, 2021).

Uma das bibliotecas padrões do *Golang* é a chamada *testing* (TESTING. . . , 2022) que provê

diversas funcionalidades para testes e análise de desempenho de algoritmos de maneira simples e confiável. A existência dessa biblioteca e a facilidade em usá-la tiveram um grande peso na escolha pelo *Golang* como linguagem.

5.1.1 Arquitetura de Software e Funcionamento

Para não comprometer os testes, as aplicações possuem a mesma arquitetura de chamada de requisição e armazenamento e recuperação de dados. O cliente é definido como um conjunto de funções que realizam chamadas *POST* ao servidor por *HTTP* e o servidor é um ouvinte dessas chamadas, realizando as operações necessárias e retornando uma resposta de acordo com a definição do protocolo.

A arquitetura de software das duas aplicações implementadas são as mesmas. Tem-se um módulo da aplicação denominada cliente e outra servidor. Os módulos funcionam através de interface de linhas de comando (*command-line interface (CLI)*, em inglês). A escolha por *CLI* foi feita pela baixa complexidade de se produzir aplicações desse tipo, além do baixo uso de recursos de *hardware*, como processamento e memória.

O módulo cliente é uma aplicação que realiza chamadas *HTTP* do tipo *POST* ao servidor, computando dados de acordo com a especificação de cada protocolo. O módulo do servidor é composto de três partes: o ouvinte *HTTP* (*listener*, em inglês), o protocolo de autenticação e a base de dados. O *listener* fica sempre em execução, esperando novas chamadas *HTTP* em rotas pré-definidas, o protocolo de autenticação realiza as computações necessárias para o cadastro e autenticação no servidor e a base de dados armazena as credenciais dos usuários cadastrados. A Fig. 5.1 ilustra a arquitetura de *software* das aplicações.

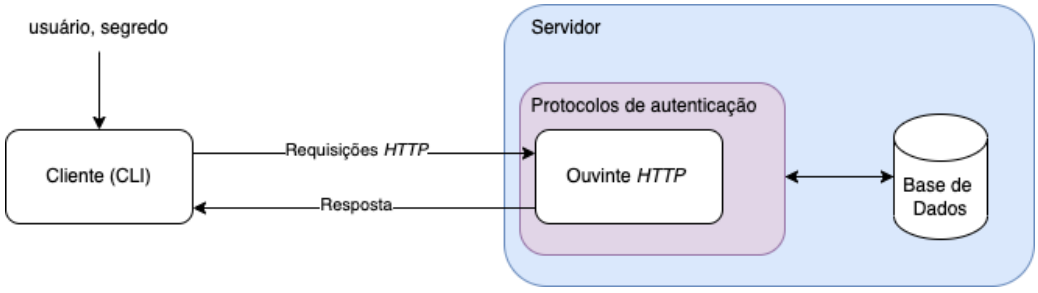

Figura 5.1: Arquitetura de software das aplicações implementadas.

A arquitetura de diretórios do projeto foi definida de acordo com a *GoLang Standards* (QUEST, 2021). O `cmd` contém os executáveis do cliente e do servidor, `internal/app` contém todas as funcionalidades do cliente e do servidor, assim como definição de banco de dados (`db`) e configurações do protocolo (`opaque` ou `cpace`) e `test` contém os algoritmos de análise de desempenho da aplicação. A Fig. 5.2 ilustra a arquitetura de diretórios da aplicação de teste do *OPAQUE*.

```
├── cmd/
│   ├── client/
│   │   └── client.go
│   └── server/
│       └── server.go
├── docs/
├── internal/
│   ├── app/
│   │   ├── client/
│   │   │   ├── authentication.go
│   │   │   ├── client.go
│   │   │   └── registration.go
│   │   ├── db/
│   │   │   └── db.go
│   │   ├── opaque/
│   │   │   └── config.go
│   │   └── server/
│   │       ├── authentication.go
│   │       ├── server.go
│   │       └── registration.go
│   └── utils/
│       └── utils.go
├── test/
│   └── main_test.go
└── vendor
```

Figura 5.2: Arquitetura de diretórios da aplicação cliente-servidor do protocolo *OPAQUE*.

As aplicações foram desenvolvidas usando as bibliotecas disponíveis de cada protocolo. Para o caso do *OPAQUE*, foi feita a escolha pela implementação `opaque-go` de Daniel Bourdrez (BOURDREZ, 2020), um dos autores do protocolo *OPAQUE*. Para o *CPace*, foi utilizada a biblioteca `go-cpace-ristretto255`, de Filippo Valsorda (VALSORDA, 2021).

5.1.2 Dificuldades Encontradas

Alguns desafios surgiram durante o processo de desenvolvimento das aplicações, especialmente na aplicação do *OPAQUE*. No processo de desenvolvimento, foram encontradas algumas discrepâncias na documentação da implementação, além de um *bug*. Nos dois casos, o autor da biblioteca foi informado e os problemas foram rapidamente corrigidos.

Outro desafio com a implementação da aplicação do *OPAQUE* foi quanto à complexidade. O *OPAQUE* é um protocolo que conta com duas etapas, uma de cadastro e uma de autenticação contendo, três trocas de mensagens cada. Para cada troca de mensagens, uma, duas ou três operações eram necessárias para o correto fluxo do protocolo. É necessária muita atenção pois o erro em uma operação causa um efeito em cascata de erros nas outras operações, o que dificulta o rastreio.

A implementação da aplicação do *CPace* foi mais rápida, em parte pela menor complexidade do protocolo e em parte por já ter muita coisa pronta da aplicação do *OPAQUE* que pôde ser reaproveitada.

5.2 Resultados

Com as implementações finalizadas, foram realizadas análises de comparativas dos protocolos *CPace* e *OPAQUE*. Foram verificados e comparados o desempenho e o uso de memória de cada uma das soluções.

5.2.1 Análise Quantitativa

As análises comparativas foram realizadas utilizando o pacote *testing* da *API* do *Golang* (TESTING. . . , 2022).

Para maior precisão e redução de ruído, o teste de desempenho foi executado 10.000 vezes para cada protocolo. Dessa maneira, foi possível obter resultados com baixo desvio padrão, o que implica que houve pouca interferência de outros processos na realização dos testes e, portanto, pode-se afirmar que os dados são precisos.

O comando utilizado para executar a análise comparativa, com seus respectivos parâmetros, foi:

```
go test <file-path> -bench=.  -benchmem -benchtime=10000x
```

A análise comparativa de desempenho e memória foi feita considerando o tempo necessário para cadastrar e autenticar um usuário no protocolo.

5.2.1.1 Desempenho

O *CPace* foi, de longe, o protocolo mais rápido dessa comparação. Em um processo médio de cadastro e autenticação, o *CPace* necessitou de 669.000ns (0.000669s) por operação, com um desvio padrão aproximado de 25.000ns. O *OPAQUE* necessitou de um tempo médio aproximado de 109.711.270ns (1,097 s) por operação, com um desvio padrão de 661.000ns. Ou seja, nos testes realizados, o *CPace* foi, em média, 163 vezes mais rápido que o *OPAQUE*.

O gráfico dos tempos obtidos (Fig. 5.3) demonstra essa grande diferença de tempo por operação em cada um dos protocolos. Como pode-se notar, o *OPAQUE* demora muito mais para ser executado. No protocolo *OPAQUE*, as tarefas de criptografia são distribuídas entre o cliente e o servidor. Ou seja, o cliente e o servidor necessitam realizar computações expressivas. Isso pode ser uma vantagem no caso de custo de processamento em servidor, por exemplo, mas também pode ser uma desvantagem por haver um peso computacional ao dispositivo do cliente, ou seja, existe um requisito mínimo para uso da aplicação.

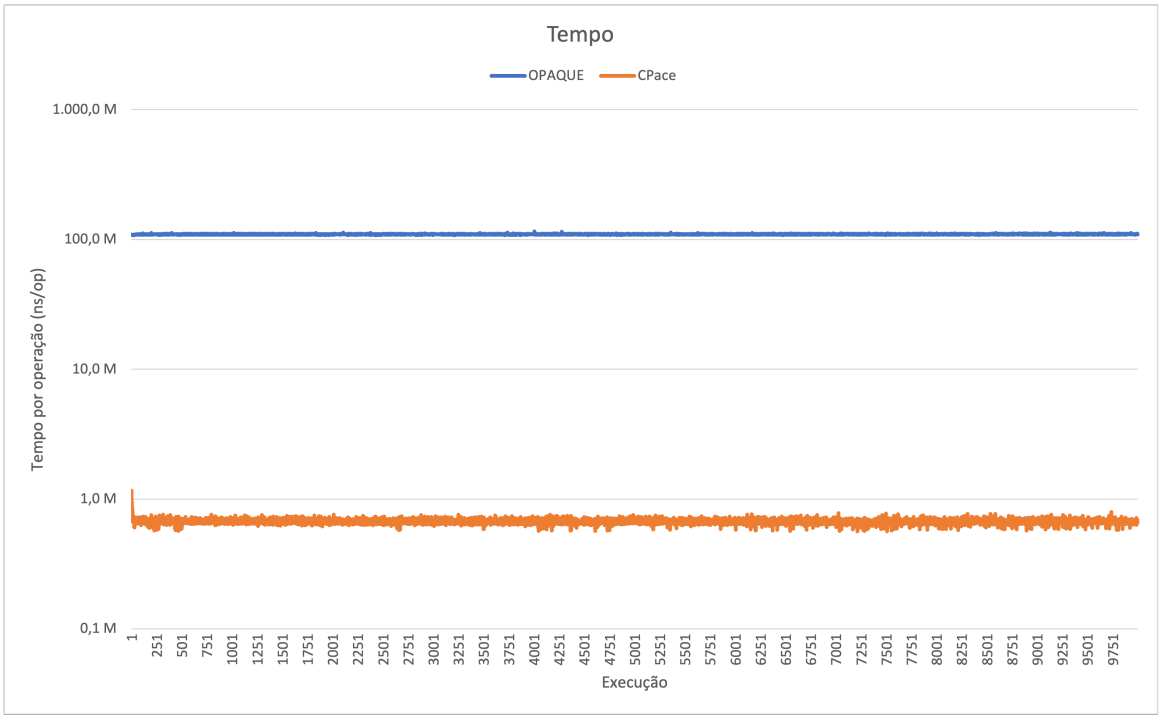

Figura 5.3: Comparação de tempo por operação entre o *CPace* e o *OPAQUE*.

5.2.1.2 Memória

Quanto ao uso de memória do sistema, o *CPace* também se mostrou muito mais eficiente que o *OPAQUE*, utilizando, em média, 46.568 *bytes* (0,046 *MB*) por operação, contra 67.247.740 *bytes* (67,24 *MB*) do *OPAQUE*. O desvio padrão para o uso de memória do *CPace* foi de 12.267 *bytes*, enquanto o do *OPAQUE* foi de 25.377 *bytes*.

A Fig. 5.4 apresenta a diferença de uso de memória por operação entre os protocolos. O *OPAQUE*, sendo um protocolo mais complexo, utiliza muito mais memória que o *CPace*. Isso é um fator que pode ser crucial na escolha de um protocolo para uma solução de autenticação. Se o servidor ou o cliente estão em dispositivos que não têm uma quantidade expressiva de memória, a execução do *OPAQUE* pode se tornar muito custosa. Antagonicamente, o *CPace* necessita de pouca memória por operação, fator que reforça o desenho do protocolo a ser utilizado em dispositivos limitados.

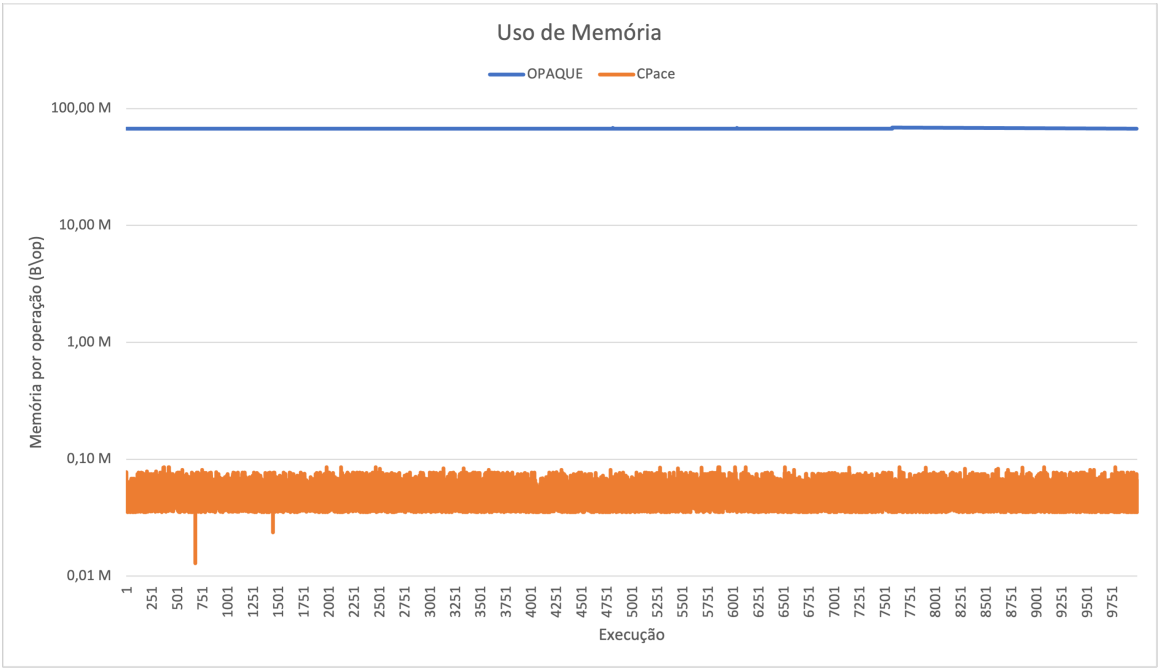

Figura 5.4: Comparação de memória utilizada por operação entre o *CPace* e o *OPAQUE*.

5.2.2 Análise Qualitativa

Por serem protocolos de categorias diferentes, cada protocolo apresenta vantagens e desvantagens. De modo geral, os protocolos *aPAKE*, classe do *OPAQUE*, são uma melhoria em relação

aos protocolos do tipo *bPAKE*, classe do *CPace*. Como explicado na seção 3.2, os protocolos *aPAKE* contam com um requisito a mais, que é a segurança em caso de comprometimento do servidor.

Para algumas aplicações, como uma aplicação *web*, essa segurança pode ser essencial. Entretanto, para outras nem tanto. Em um cenário hipotético de uma rede de sistemas embarcados por exemplo, se o comprometimento de um único dispositivo significar o comprometimento de todos os outros, esse requerimento adicional pode não ser o suficiente para impedir um adversário de comprometer o sistema por completo. Nesse caso, um protocolo do tipo *bPAKE* pode ser utilizado.

Em um segundo cenário que utiliza um único servidor robusto para a autenticação de diversos clientes, a garantia de segurança dos outros usuários em caso de comprometimento do servidor pode se tornar um requisito essencial da solução implementada.

Em relação à complexidade de implementação, o *OPAQUE*, é um protocolo muito mais complexo de ser implementado. O *OPAQUE* conta com alguns requisitos a mais, como o *AKE* e a *OPRF*, para ser implementado, além da necessidade de definição de uma etapa de cadastro para garantir o correto funcionamento do protocolo.

O *OPAQUE* é um protocolo muito mais modular e aberto a modificações que o *CPace*. Isso pode ser uma grande vantagem. O *OPAQUE* é definido por três funcionalidades principais: a *OPRF*, um mecanismo de recuperação de chaves e um protocolo *AKE* (BOURDREZ et al., 2022). Qualquer uma dessas funcionalidades pode ser alterada para se obter um protocolo de autenticação específico. Por isso, pode ser possível construir protocolos de desempenho melhor que o que foi utilizado nesse trabalho, tanto em relação à tempo quanto ao uso de memória. Além disso, a modularidade do *OPAQUE* permite separar a execução de cada módulo em processos e até dispositivos separados.

O *CPace* apresenta outra vantagem sobre o *OPAQUE*. Por ser um protocolo do tipo *bPAKE*, ele pode ser utilizado em comunicação cliente-cliente e cliente-servidor. O *OPAQUE*, entretanto, foi desenvolvido como um protocolo somente para comunicação cliente-servidor.

A tabela Tab. 5.1 apresenta uma lista de característica, ressaltando as vantagens e desvan-

tagens de cada protocolo.

Funcionalidades	OPAQUE	CPACE
Complexidade de implementação	Alta	Média
Pares comunicantes	Cliente-Servidor	Cliente-Cliente; Cliente-Servidor
Modular	X	
Maior segurança em caso de comprometimento do sistema	X	

Tabela 5.1: Tabela comparativa entre *OPAQUE* e *CPace*.

5.3 Considerações

A diferença de desempenho e uso de memória entre os dois protocolos é evidente, chegando a ser de ordem de magnitudes diferentes. Alguns dos fatores que ocasionam essa diferença são:

1. A implementação do *OPAQUE* utilizada conta com muito mais segurança pois todos os dados são serializados antes de serem enviados à outra parte, e desserializado quando chega no destino. A implementação do *CPace* não conta com esse grau de segurança adicional, que seria importante ser implementada para uso em uma aplicação real;

2. O *OPAQUE* possui um passo de cadastro complexo, enquanto o *CPace* deixa a critério do desenvolvedor como será realizado o cadastro. Nessa comparação, o cadastro do *CPace* foi feita como uma simples troca de mensagens entre o cliente e o servidor, o que é extremamente leve em relação ao cadastro do *OPAQUE*;

5.4 Conclusão

A partir da implementação e dos testes, foi possível realizar uma análise comparativa entre os dois protocolos *PAKE* recomendados pela *IETF*. Pode-se perceber que a escolha por um protocolo não é fácil, mas a análise das vantagens e desvantagens dos protocolos facilita essa escolha. O próximo capítulo conclui esse trabalho, apresentando a quais casos de uso cada protocolo melhor se adapta.

Capítulo 6

Conclusões

Ao se falar em solução de problemas, especialmente em computação, é comum ouvir-se a frase ”não existe bala de prata”. Ou seja, não há uma única solução ótima para todos os problemas. Cada problema deve ser estudado para que seja encontrada uma solução apropriada. Algo que funciona para um problema específico pode não necessariamente funcionar para outro.

Essa frase e esse entendimento são essenciais nesse trabalho. Através dos estudos realizados, identificou-se que cada protocolo de autenticação escolhido tem vantagens e desvantagens e, consequentemente, possuem propósitos distintos.

Se há a necessidade de um protocolo de autenticação seguro para sistemas mais simples, e que seja mais fácil de escalar, o *CPace* é a solução desejada. O *CPace*, além de ser mais rápido e consumir menos memória, é mais simples de ser implementado. Por outro lado, quando é necessário um protocolo de autenticação que seja mais robusto e seguro, e com boa modularidade para personalizações do produto, o *OPAQUE* é a escolha ideal.

Esse trabalho descreve em detalhes as característica dos protocolos *CPace* e *OPAQUE*, destacando suas qualidades e fraquezas de forma comparativa. Além disso, a análise quantitativa apresentada exalta a diferença de desempenho e uso de memória em aplicações reais, que de fato auxilia na escolha por um protocolo em desenvolvimento de sistemas.

6.1 Trabalhos futuros

Nesse trabalho, foram comparados dois protocolos de autenticação, o *CPace* e o *OPAQUE*, de classes diferentes. Um trabalho futuro interessante seria uma comparação do protocolo *OPAQUE* variando os três módulos constituintes do protocolo. Por exemplo, alterando a *OPRF* e o protocolo *AKE* seria possível obter variações significantes de desempenho e uso de memória?

Referências Bibliográficas

ABDALLA, M.; HAASE, B.; HESSE, J. *CPace, a balanced composable PAKE*. [S.l.], 2022. Acessado em: 23 de abr. de 2022. Disponível em: <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-cpace-05>.

AL-KUWARI, S.; DAVENPORT, J. H.; BRADFORD, R. J. *Cryptographic Hash Functions: Recent Design Trends and Security Notions*. 2011. Cryptology ePrint Archive, Report 2011/565. <https://ia.cr/2011/565>.

BELLOVIN, S. M.; MERRITT, M. Encrypted key exchange: Password-based protocols secure against dictionary attacks. 1992.

BELLOVIN, S. M.; MERRITT, M. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. [S.l.: s.n.], 1993. p. 244–250.

BERSANI, F. et al. *The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method*. RFC Editor, 2008. RFC 5106. (Request for Comments, 5106). Disponível em: <https://www.rfc-editor.org/info/rfc5106>.

BOURDREZ, D. *Bytemare/opaque: GO implementation of opaque, the asymmetric password-authenticated key exchange protocol*. 2020. Acessado em: 30 de abr. de 2022. Disponível em: <https://github.com/bytemare/opaque>.

BOURDREZ, D. et al. *The OPAQUE Asymmetric PAKE Protocol*. [S.l.], 2022. Acessado em: 30 de abr. de 2022. Disponível em: <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-opaque-08>.

CORON, J. *What is cryptography?* [S.l.]: Institute of Electrical and Electronics Engineers - IEEE, 2006.

DAVE, K. T. Brute-force attack 'seeking but distressing'. *Int. J. Innov. Eng. Technol. Brute-force*, Citeseer, v. 2, n. 3, p. 75–78, 2013.

DENIS, F.; PESTANO, G. *CPace-Ristretto255, a balanced PAKE*. 2021. Acessado em: 20 maio de 2022. Disponível em: <https://github.com/jedisct1/rust-cpace>.

DIFFIE, W.; HELLMAN, M. E. Special feature exhaustive cryptanalysis of the nbs data encryption standard. *Computer*, IEEE, v. 10, n. 6, p. 74–84, 1977.

DIFFIE, W.; OORSCHOT, P. C. V.; WIENER, M. J. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, v. 2, n. 2, p. 107–125, jun. 1992. ISSN 1573-7586. Disponível em: <https://doi.org/10.1007/BF00124891>.

ELAKRAT, M. A.; JUNG, J. C. Development of field programmable gate array–based encryption module to mitigate man-in-the-middle attack for nuclear power plant data communication network. *Nuclear Engineering and Technology*, v. 50, n. 5, p. 780–787, 2018. ISSN 1738-5733. Disponível em: <https://www.sciencedirect.com/science/article/pii/S173857331730565X>.

FOUNDATION, E. F. *HTTPS Everywhere FAQ*. 2021. Acessado em: 17 maio de 2022. Disponível em: <https://www.eff.org/https-everywhere/faq\#what-does-https-everywhere-protect-me-against>.

FUNK, P.; BLAKE-WILSON, S. *Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)*. RFC Editor, 2008. RFC 5281. (Request for Comments, 5281). Disponível em: <https://www.rfc-editor.org/info/rfc5281>.

GO. *The Go Memory Model*. 2014. Internet. Acessado em: 20 maio de 2022. Disponível em: <https://go.dev/ref/mem>.

GO. *Frequently Asked Questions (FAQ)*. s.d. Internet. Acessado em: 20 maio de 2022. Disponível em: <https://go.dev/doc/faq>.

GO. *The Go Programming Language*. s.d. Internet. Acessado em: 20 maio de 2022. Disponível em: <https://go.dev>.

HAASE, B.; HESSE, J. *CPace, a balanced composable PAKE*. 2022. Acessado em: 20 maio de 2022. Disponível em: <https://github.com/cfrg/draft-irtf-cfrg-cpace>.

HAO, F.; OORSCHOT, P. C. van. Sok: Password-authenticated key exchange–theory, practice, standardization and real-world lessons. *Cryptology ePrint Archive*, 2021.

HAO, F.; RYAN, P. Y. A. Password authenticated key exchange by juggling. In: CHRISTIANSON, B. et al. (Ed.). *Security Protocols XVI*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 159–171. ISBN 978-3-642-22137-8.

IRTF. *IRTF Crypto Forum Research Group (CFRG)*. 2014. Acessado em: 17 maio de 2022. Disponível em: <https://irtf.org/cfrg>.

JABLON, D. P. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, ACM New York, NY, USA, v. 26, n. 5, p. 5–26, 1996.

JABLON, D. P. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, ACM New York, NY, USA, v. 26, n. 5, p. 5–26, 1996.

LEWI, K. et al. *An implementation of the OPAQUE password-authenticated key exchange protocol*. 2022. Acessado em: 20 maio de 2022. Disponível em: <https://github.com/novifinancial/opaque-ke>.

MELHORAMENTOS. *DICIONÁRIO Brasileiro da Língua Portuguesa*. [s.n.], 2021. Internet. Acessado em: 5 de nov. de 2021. Disponível em: <https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/informa\\%C3\\%A7\\%C3\\%A3o/>.

NAM, J. et al. An off-line dictionary attack on a simple three-party key exchange protocol. *IEEE Communications Letters*, v. 13, n. 3, p. 205–207, 2009.

NEUMAN, B.; TS'O, T. Kerberos: an authentication service for computer networks. v. 32, p. 33–38, 1994.

QUEST, K. *Golang-Standards/Project-layout*. 2021. Acessado em: 30 de abr. de 2022. Disponível em: <https://github.com/golang-standards/project-layout/blob/master/README_ptBR.md>.

R von S.; J van N. From information security to cyber security. 2013.

SALOWEY, J. A. et al. *The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)*. RFC Editor, 2007. RFC 4851. (Request for Comments, 4851). Disponível em: <https://www.rfc-editor.org/info/rfc4851>.

SHERMAN, A. T. et al. Formal methods analysis of the secure remote password protocol. In: *Logic, Language, and Security*. [S.l.]: Springer, 2020. p. 103–126.

SIMON, D.; HURST, R.; ABOBA, D. B. D. *The EAP-TLS Authentication Protocol*. RFC Editor, 2008. RFC 5216. (Request for Comments, 5216). Disponível em: <https://www.rfc-editor.org/info/rfc5216>.

SIMPSON, W. A. *PPP Authentication Protocols*. RFC Editor, 1992. RFC 1334. (Request for Comments, 1334). Acessado em: 23 maio de 2022. Disponível em: <https://www.rfc-editor.org/info/rfc1334>.

SOTILLO, S. *Extensible Authentication Protocol (EAP) Security Issues*. 2007.

SULLIVAN, N. *Crfg Proposed PAKE Selection Process*. 2019. Acessado em: 17 maio de 2022. Disponível em: <https://mailarchive.ietf.org/arch/msg/cfrg/-J43ZsPw2J5MBC-k8y6--kJJtZk/>.

TANG, Q.; CHEN, L. Weaknesses in two group diffie-hellman key exchange protocols. *Cryptology ePrint Archive*, 2005.

TESTING Package - Go. Google, 2022. Acessado em: 07 maio de 2022. Disponível em: <https://pkg.go.dev/testing>.

VALSORDA, F. *FiloSottile/go-cpace-RISTRETTO255: An experimental go implementation of the CPACE pake, instantiated with the RISTRETTO255 group.* 2021. Acessado em: 30 de abr. de 2022. Disponível em: <https://github.com/FiloSottile/go-cpace-ristretto255>.

VOLLBRECHT, J. et al. *Extensible Authentication Protocol (EAP)*. RFC Editor, 2004. RFC 3748. (Request for Comments, 3748). Disponível em: <https://www.rfc-editor.org/info/rfc3748>.

W3TECHS. *Usage statistics of default protocol HTTPS for websites*. s.d. Acessado em: 18 maio de 2022. Disponível em: <https://w3techs.com/technologies/details/ce-httpsdefault>.

WOOD, C.; BOURDREZ, D.; LEWI, K. *The OPAQUE Asymmetric PAKE Protocol*. 2022. Acessado em: 20 maio de 2022. Disponível em: <https://github.com/cfrg/draft-irtf-cfrg-opaque>.

WU, T. The secure remote password protocol. p. 97–111, 1998.

WYSOPAL, C.; ENG, C.; SHIELDS, T. Static detection of application backdoors. *Datenschutz und Datensicherheit-DuD*, Springer, v. 34, n. 3, p. 149–155, 2010.