Análise da Criptografia Ponta-a-Ponta: Um Estudo sobre o Protocolo Signal e suas implementações

Luiz Antônio Lima de Freitas Leite[1], **Max José Lobato Pantoja Junior**[1], **Wesley Pontes Barbosa**[1]

[1]Instituto de Ciências Exatas e Naturais (ICEN) – Universidade Federal Pará
Belém, PA – Brasil

{luiz.freitas.leite,max.junior,wesley.pontes.barbosa}@icen.ufpa.br

Abstract. *[REVISAR] This paper analyzes the Signal Protocol, the de facto standard for secure messaging. We explore its evolution from the "Axolotl"ratchet, named after the self-healing salamander, to its current state. Technical concepts such as the Double Ratchet Algorithm and X3DH are detailed. We present case studies on WhatsApp, discussing its closed-source implementation allowed by specific commercial agreements and the controversy surrounding its 2021 privacy policy update regarding metadata collection. Finally, we examine the Matrix protocol and its Olm/Megolm libraries, which adapt Signal's concepts for federated environments and high-performance group chats.*

Resumo. *[REVISAR] Este artigo analisa o Protocolo Signal, o padrão de fato para mensagens seguras. Exploramos sua evolução desde o algoritmo "Axolotl", nomeado em referência à salamandra regenerativa, até seu estado atual. Conceitos técnicos como o algoritmo Double Ratchet e X3DH são detalhados. Apresentamos estudos de caso sobre o WhatsApp, discutindo sua implementação de código fechado permitida por acordos comerciais específicos e a controvérsia em torno da atualização de sua política de privacidade em 2021 referente à coleta de metadados. Por fim, examinamos o protocolo Matrix e suas bibliotecas Olm/Megolm, que adaptam os conceitos do Signal para ambientes federados e grupos de alta performance.*

1. Introdução

[MAX] [melhorar contextualização historica para o surgimento da criptografia ponta-aponta] A segurança em comunicações digitais evoluiu de modelos baseados em criptografia de transporte (como TLS/SSL) para a Criptografia Ponta-a-Ponta (E2EE), onde nem mesmo o provedor do serviço tem acesso ao conteúdo. O protagonista dessa mudança é o Protocolo Signal.

1.1. História e o Nome Axolotl

[MAX] [melhorar explicação sobre a historia especifica do signal, deixe a historia mais direta e relacione o anfibio com o a auto-cura do protocolo (quem nem o cachorro do inferno no Kerberos)] O protocolo foi desenvolvido pela *Open Whisper Systems*, liderada por Moxie Marlinspike e Trevor Perrin. Inicialmente, o mecanismo central de criptografia não se chamava "Signal", mas sim **Axolotl**.

O nome foi escolhido em homenagem ao axolote (*Ambystoma mexicanum*), uma salamandra aquática conhecida por sua impressionante capacidade de auto-regeneração. Essa analogia biológica referia-se à propriedade de "auto-cura"(*self-healing*) do protocolo: se uma chave de sessão for comprometida por um atacante, o algoritmo rotaciona as chaves automaticamente na próxima mensagem, "curando"a segurança da conversa e impedindo que o atacante decifre mensagens futuras [Perrin and Marlinspike 2016].

Em 2016, para simplificar a nomenclatura e evitar confusões de marcas registradas, o nome foi oficialmente alterado para **Signal Protocol**, unificando a marca do aplicativo e da tecnologia subjacente.

2. Conceitos Básicos

[MAX] [foca em explicar todas as propriedades relevantes pro protocolo signal, verifica se vale a pena adicionar aqui tambem o Sealed Sender que o whatsapp nao usa (ou nao usa mais)] Para compreender o funcionamento do Protocolo Signal, é necessário definir certas propriedades criptográficas fundamentais que ele visa garantir:

- **Sigilo Perfeito Encaminhado (Forward Secrecy):** Garante que, se a chave privada de um usuário for roubada hoje, as mensagens trocadas no passado permaneçam seguras. Isso é obtido através da geração de chaves de sessão efêmeras que são descartadas após o uso.
- **Segurança Pós-Comprometimento (Post-Compromise Security):** Também conhecida como "Break-in Recovery". Refere-se à capacidade do protocolo de restabelecer a segurança após um comprometimento temporário das chaves, através da atualização contínua do material criptográfico.
- **Diffie-Hellman (DH):** Um método que permite a duas partes, que não têm conhecimento prévio uma da outra, estabelecerem conjuntamente uma chave secreta compartilhada em um canal inseguro.

3. O Protocolo Signal

[LA] [definir melhor os elementos necessarios para descrever o protocolo e, principalmente, fazer um diagrama do protocolo em funcionamento] O Protocolo Signal combina o algoritmo de acordo de chaves X3DH e o algoritmo Double Ratchet. O código é aberto e auditável, hospedado no GitHub da organização *Signal Messenger*, dividido em bibliotecas para o protocolo (`libsignal`), clientes (Android, iOS e Desktop) e servidor.

3.1. O Algoritmo Double Ratchet

O coração do protocolo é o Double Ratchet (Catraca Dupla). Ele gerencia a rotação contínua das chaves de criptografia de mensagens. O termo "catraca"implica que o processo avança em uma direção e não pode ser revertido (garantindo o *Forward Secrecy*). O algoritmo utiliza duas camadas de derivação de chaves:

1. **Diffie-Hellman Ratchet:** Ocorre quando há troca de mensagens. As partes renovam suas chaves públicas DH, alterando a "raiz"da cadeia de chaves. Isso fornece a propriedade de auto-cura.
2. **Symmetric-Key Ratchet (Hash):** Ocorre para cada mensagem enviada dentro da mesma sessão DH. Uma função de derivação de chaves (KDF) gera uma nova chave de mensagem a partir da anterior.

Figura 1. Diagrama conceitual do Double Ratchet: A KDF Chain deriva chaves de mensagem, enquanto o DH Ratchet atualiza a KDF Chain.

Essa arquitetura garante que cada mensagem seja criptografada com uma chave única que nunca é reutilizada.

4. Estudos de Caso e Implementações

[WESLEY] [explicar como o whatsapp usa o signal, o fato de ser codigo fechado impede de entrar em detalhes muito profundos, mas]

4.1. WhatsApp: Privacidade vs. Metadados

O WhatsApp é a implementação mais popular do Protocolo Signal. Embora o protocolo seja licenciado sob a GPLv3 (que exigiria que softwares derivados também fossem de código aberto), o WhatsApp mantém seu código fechado (proprietário).

Isso é possível porque o WhatsApp não simplesmente copiou o código do repositório público; eles firmaram um acordo comercial específico com a Open Whisper Systems em 2014 para integrar a biblioteca *libsignal* em sua infraestrutura antes que as obrigações estritas da GPL fossem aplicadas à sua base de código final [Marlinspike 2016].

4.1.1. A Polêmica dos Metadados (2021)

Embora o conteúdo das mensagens no WhatsApp seja protegido pela mesma criptografia do Signal, o modelo de negócios da Meta (controladora do WhatsApp) baseia-se na exploração de **metadados**.

Em janeiro de 2021, o WhatsApp atualizou seus Termos de Serviço e Política de Privacidade, gerando reação global e migração em massa para o Signal e Telegram. A atualização explicitou a coleta de dados que não são protegidos pela criptografia ponta-a-ponta. Conforme a seção "Informações que coletamos"da política [WhatsApp Inc. 2021]:

- **Dados de Atividade:** Tempo, frequência e duração das interações.
- **Grafo Social:** Quem fala com quem (remetente e destinatário).
- **Informações do Dispositivo:** Modelo de hardware, sistema operacional, nível de bateria, força do sinal e identificadores únicos.
- **Endereço IP:** Usado para estimar a localização geral.

Ao contrário do aplicativo Signal, que desenhou seu sistema para minimizar o conhecimento do servidor (usando técnicas como *Sealed Sender*), o WhatsApp centraliza metadados, permitindo a construção de perfis comportamentais detalhados dos usuários, mesmo sem ler o conteúdo textual das mensagens.

4.2. Matrix e Olm/Megolm

O Matrix é um padrão aberto para comunicação descentralizada e federada. Diferente do Signal (que é centralizado), o Matrix permite que servidores diferentes conversem entre si. Para garantir segurança nesse ambiente, o Matrix desenvolveu a biblioteca **Olm**.

Olm é uma implementação do algoritmo Double Ratchet do Signal, escrita em C++ e adaptada para a arquitetura do Matrix. No entanto, o algoritmo original do Signal é "pesado"para grupos grandes (complexidade $O(N^2)$), pois exige o estabelecimento de sessões individuais par-a-par.

Para resolver isso, o Matrix introduziu o **Megolm**. O Megolm sacrifica a propriedade de "auto-cura"imediata (o *break-in recovery* é mais lento) em troca de escalabilidade massiva para grupos com milhares de usuários. No Megolm, cada participante cria uma sessão de saída ("outbound") e compartilha a chave da catraca com os outros participantes via canais seguros Olm [The Matrix.org Foundation 2024]. Isso permite criptografia eficiente em ambientes federados sem quebrar as propriedades fundamentais de confidencialidade.

5. Conclusão

A evolução do algoritmo Axolotl para o onipresente Protocolo Signal redefiniu a segurança na internet. A tecnologia provou ser robusta contra adversários estatais e corporativos no que tange à confidencialidade do conteúdo.

Entretanto, a análise das implementações do WhatsApp e Matrix revela que o protocolo é apenas uma peça do quebra-cabeça. No caso do WhatsApp, a segurança criptográfica convive com a vigilância de metadados sancionada pelos termos de uso. No caso do Matrix, vemos a adaptação do protocolo (Megolm) para garantir soberania de dados através da federação. Conclui-se que a "privacidade"é um conceito mais amplo que a "criptografia", dependendo tanto da matemática dos algoritmos quanto da ética das implementações.

Referências

Marlinspike, M. (2016). Whatsapp's signal protocol integration. `https://signal.org/blog/whatsapp-complete/`. Acesso em: 15 jun. 2024.

Perrin, T. and Marlinspike, M. (2016). The double ratchet algorithm. Technical report, Signal. Acesso em: 15 jun. 2024.

The Matrix.org Foundation (2024). Matrix Specification - E2E Encryption Implementation. `https://spec.matrix.org/latest/client-server-api/#end-to-end-encryption`. Descrição das bibliotecas Olm e Megolm.

WhatsApp Inc. (2021). WhatsApp Privacy Policy Updates. `https://www.whatsapp.com/legal/updates/privacy-policy`. Termos atualizados em Janeiro de 2021. Acesso em: 15 jun. 2024.