Solução para habilitar conversas integras e auditáveis em aplicativos de troca de mensagens instantâneas

Andrea E. Komo[1], **Marcos A. Simplicio Jr.**[1]

[1]Escola Politécnica – Universidade de São Paulo (USP)

{aerina,mjunior}@larc.usp.br

Abstract. *Message integrity in mobile communication apps has become an important issue in recent years. In particular, since such apps are being used as corporate tools, it is not uncommon for messages thereby exchanged to be used as negotiation documents. However, by design, most of such applications do not provide any verification feature to confirm the integrity of the conversations. Indeed, analyses show that it is possible to surreptitiously modify records in apps like WhatsApp, Telegram and others. To tackle this issue, this work proposes a message structure composed of chained cryptographic hashes, thus ensuring the integrity and auditability of conversations. The solution is architecture-independent, so it can be integrated with any instant message app.*

Resumo. *A integridade das mensagens trocadas em aplicativos de comunicação instantânea tem sido uma questão importante nos últimos anos. Em particular, como esses aplicativos têm sido utilizados como ferramentas corporativas, não é incomum que as conversas neles realizadas sejam usadas como documentos para comprovar negociações. Por outro lado, a maioria desses aplicativos não apresenta um recurso de verificação para confirmar a integridade da conversas. De fato, análises mostram que é possível modificar registros de forma proposital ou acidental em aplicativos como WhatsApp, Telegram e outros, sem que a alteração seja facilmente detectada. Este trabalho propõe o uso de uma estrutura encadeada de hashes criptográficos das mensagens de forma a garantir a integridade e permitir a verificação confiável das conversas. Essa solução é independente da arquitetura do sistema e pode ser incorporada em qualquer aplicativo de mensagem.*

1. Introdução

Devido a sua facilidade de uso [Schröder et al. 2016] e popularidade, aplicativos de mensagens instantâneas (e.g.,*WhatsApp*, *Messenger*, e *Telegram*) estão sendo usados por várias entidades, inclusive no mundo corporativo. Exemplos no Brasil incluem tribunais de justiça [TJDFT 2016] e bancos federais [BB 2018]. Além de facilitar comunicações cotidianas, a eficiência desses aplicativos e a grande adesão de usuários têm influenciado empresas a adotá-los em comunicações internas, negociações, agendamentos, entre outros serviços. Estas práticas tornaram-se tão significativas que o aplicativo *WhatsApp* também criou uma versão específica para empresas, o *WhatsApp Business* [WhatsApp 2019].

Dado que aplicativos de comunicação instantânea têm deixado de ser simples redes sociais para se tornarem verdadeiras ferramentas de trabalho, é preciso garantir algumas características de segurança. Dentre os aplicativos disponíveis no mercado, nota-se

que a segurança está concentrada principalmente na confidencialidade das mensagens, na forma de criptografia fim-a-fim [WhatsApp 2016, Cohn-Gordon et al. 2017]. Em contrapartida, a integridade, autenticidade e irretratabilidade das conversas não têm recebido tanta atenção, o que muitas vezes impede a averiguação da veracidade das mensagens trocadas [Schliep et al. 2017, Schliep and Hopper 2018]. Na realidade, essa possibilidade de retratabilidade (ou, em última instância, a redução da auditabilidade) de mensagens é comumente um objetivo de projeto, visando melhorar a privacidade das comunicações mesmo em caso de captura física dos dispositivos usados para esse fim [WhatsApp 2016, Cohn-Gordon et al. 2017].

Embora a retratabilidade seja útil em vários cenários de aplicação, em alguns casos essa propriedade é indesejada. Em particular, no cenário empresarial a auditabilidade das conversas é comumente necessária para evitar fraudes. Afinal, nesses cenários as mensagens trocadas muitas vezes possuem o valor de um comprovante para os interlocutores, seja de informações fornecidas ou de pedidos realizados.

Visando melhorar a segurança das comunicações em aplicativos de mensagens instantâneas, este trabalho propõe uma arquitetura de comunicação segura que garanta a integridade, autenticidade e irretratabilidade das conversas. Mais precisamente, a solução proposta engloba não apenas a proteção de cada mensagem individual, mas também a garantia de que sequências de mensagens trocadas entre os interlocutores não podem ser adulteradas. Desta forma, o esquema proposto torna as conversas realizadas passíveis de auditorias confiáveis. Basicamente, a solução utiliza cadeias de hashes criptográficos assinadas, de forma similar às utilizadas em sistemas como Git [Torvalds 2005], Bitcoin [Nakamoto 2008] e SecureTCG [Simplicio et al. 2014]. Quando relevante, as partes também podem apresentar apenas um conjunto das mensagens trocadas, permitindo sua auditoria enquanto preserva-se a privacidade do restante da conversa. Finalmente, o custo adicional de memória da solução é reduzido, consistindo no armazenamento de uma assinatura digital por usuário participando da comunicação.

O restante deste documento está organizado da seguinte forma. A Seção 2 analisa os mecanismos de segurança presentes em aplicativos de comunicação instantânea atuais. A Seção 3 detalha a arquitetura proposta neste trabalho, bem como os conceitos envolvidos. A Seção 4 descreve o aplicativo usado com prova de conceito e os resultados dos testes nele realizados. A Seção 5 conclui o trabalho e discute possíveis melhorias futuras.

2. Fundamentação teórica e trabalhos relacionados

Para garantir a integridade e autenticidade das mensagens trocadas, a maioria dos protocolos de segurança usados atualmente para comunicação instantânea não envolvem assinaturas digitais, mas apenas esquemas baseados em chaves simétricas [WhatsApp 2016, Cohn-Gordon et al. 2017], como Códigos de Autenticação de Mensagens (*Messages Authentication Codes* – MACs) e Esquemas de Cifração Autenticada (*Authenticated Encryption* – AE) [Simplicio et al. 2013]. Esse é o caso de aplicativos populares, como WhatsApp [WhatsApp 2016], Signal [Cohn-Gordon et al. 2017], Telegram [Telegram 2019a], e Threema [Rösler et al. 2018]. Por essa razão, o acesso às mensagens recebidas também permite sua manipulação, levando-se à capacidade de retratabilidade das mensagens enviadas ou recebidas. A razão é que, para manipular conteúdos locais, bastaria (1) acessar o espaço de memória onde as mensagens ficam armazenadas, (2) ma-

nipular a mensagem, e (3) recalcular o conjunto de dados de autenticação da mensagem alterada, empregando a mesma chave usada para verificar o conteúdo original. Nesse caso, embora seja possível identificar discrepâncias entre os conteúdos armazenados nos dispositivos dos usuários participantes da comunicação, não seria possível aferir quem foi o responsável pela manipulação (ou mesmo se houve modificação por mais de um usuário). Um exemplo ilustrativo do problema que essa falta de integridade pode causar é mostrado na Fig. 1, em que a ordem das mensagens em um aplicativo que simula o WhatsApp resulta em sentidos diferentes da conversa. Como se pode imaginar, essa mudança pode ser realizada propositalmente ou mesmo acidentalmente, devido a problemas de latência na comunicação [Schliep et al. 2017, Schliep and Hopper 2018].

Figura 1. Exemplo de conversa com a ordem alterada.

Nos casos em que os mecanismos de criptografia utilizado nas comunicações adotam o modelo cliente-servidor, em princípio pode-se verificar a ocorrência de manipulações locais ao compará-las com os dados armazenados no servidor em si. Entretanto, nem sempre isso é possível na prática, dependendo da forma como o aplicativo é projetado. Um exemplo em particular é o caso do *Telegram*, que usa criptografia cliente-servidor em conversas por padrão (i.e., quando não é habilitada a funcionalidade de ”chat secreto”). Embora o servidor de fato armazene as mensagens trocadas entre usuário, o aplicativo ainda permite manipulações de dados ao: (1) incluir entre suas funcionalidades a capacidade de deletar mensagens dos dispositivos de todos os usuários e do servidor, sejam essas mensagens enviadas pelo usuário que as está deletando ou não [Telegram 2019b]; e (2) usar um mecanismo de cifração autenticada para proteger as mensagens, permitindo que manipulações locais sejam imperceptíveis após a deleção das mensagens correspondentes no servidor [Telegram 2019a]. A Fig. 2 mostra um exemplo de como uma conversa poderia ser manipulada usando simplesmente a função de apagamento de mensagens, resultando em uma comunicação com sentido bastante diferente do original. Exemplos mais complexos envolveriam a edição das mensagens em vez de sua deleção local, de modo que elas aparecessem como apagadas apenas no servidor.

Já no caso de aplicativos que adotam criptografia fim-a-fim, a modificação de conversas é ainda de mais difícil detecção. Em particular, um dos principais mecanis-

Figura 2. Exemplo de conversa com mensagens apagadas. Fonte: Autores via aplicativo *Telegram*.

mos de segurança usados atualmente para esse propósito é o protocolo desenvolvido para uso pelo aplicativo de código aberto *Signal*, o qual foi formalmente analisado em [Cohn-Gordon et al. 2017] e é atualmente também adotado por outros aplicativos, como o WhatsApp [WhatsApp 2016]. A principal particularidade de segurança desse protocolo é o elevado grau de confidencialidade das mensagens trocadas: além de utilizar criptografia fim-a-fim, as chaves de proteção das mensagens são continuamente renovadas por meio do mecanismo conhecido como "*double ratchet*"[Marlinspike and Perrin 2016]. Como resultado desse mecanismo, cada mensagem é protegida por uma chave distinta e específica para cada par de usuários (ou, no caso de conversas em grupo, entre cada usuário e o grupo em si) [Marlinspike and Perrin 2016, Rösler et al. 2018]. Assim, descobrir a chave de uma mensagem não permite descobrir chaves de mensagens passadas, propriedade esta conhecida como *forward secrecy* ou *future secrecy* ("segurança futura", em uma tradução livre); além disso, o sistema pode se "auto-regenerar", prevenindo o comprometimento de comunicações futuras mesmo se houver vazamento de uma chave.

Embora esse tipo de mecanismo seja importante para a preservação da privacidade dos usuários, ele compromete os requisitos de um cenário corporativo, no qual comumente deseja-se ter as conversas registradas como documentos confiáveis para consultas posteriores. Afinal, dada a criptografia fim-a-fim, o sistema não permite que os servidores centrais acessem os conteúdos das mensagens, de modo que não há um ambiente à prova de manipulações do qual as mensagens originais possam ser recuperadas. Na realidade, alguns aplicativos de mensagem instantânea com segurança fim-a-fim chegam até mesmo a evitar a necessidade de um servidor central, utilizando tecnologias *peer-to-peer* para que não existam sequer registros de metadados da comunicação (e.g., os instantes de envio de mensagens) [Komo et al. 2018, Rogers et al. 2018].

No melhor do nosso conhecimento, um dos poucos trabalhos na literatura que leva em consideração requisitos de integridade de mensagens após sua recepção em aplicativos de comunicação instantânea é o protocolo proposto em [Schliep and Hopper 2018]. Nele a integridade da conversa é garantida pela combinação do protocolo de acordo de chaves, o NAXOS [LaMacchia et al. 2007], e um protocolo simétrico de cifração au-

tenticada com dados associados, o AES-GCM com vetores de inicialização aleatórios [A. Mcgrew and Viega 2004]. Para cada mensagem é gerada uma chave efêmera NAXOS com base na mensagem anterior e depois usa-se essa chave para cifrar a mensagem. Contudo, a solução também busca fornecer negação plausível, levando à propriedade retratabilidade já discutida como indesejada em cenário corporativos.

3. Arquitetura proposta

Para garantir a auditabilidade nos sistemas de troca de mensagem, propõe-se o uso de hashes criptográficos encadeados e assinados [Hu et al. 2005]. Considerando uma cadeia com $\mathcal{N}$ informações registradas, para inserir uma nova informação $\mathcal{N}+1$ é necessário que o novo registro apresente a nova informação e o hash criptográfico da informação anterior $\mathcal{N}$. A última informação é então assinada, permitindo que qualquer manipulação seja detectável. Assim, a cadeia cresce da esquerda para direita, conforme ilustrado na Fig. 3.

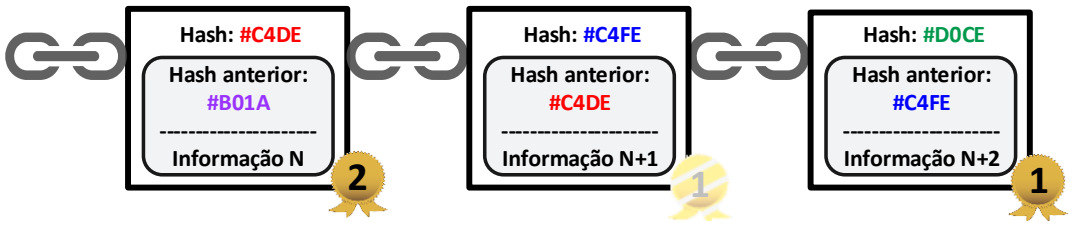

Figura 3. Estrutura de hashes encadeados. Assumindo que o usuário 1 enviou as mensagens N+1 e N+2, apenas sua última assinatura precisa ser armazenada.

Mais formalmente, tem-se a seguinte lógica para criação da cadeia de mensagens de uma conversa (note que cada $bloco_n$ é assinado digitalmente):

1. $bloco_0 = (\emptyset, M_1)$, onde $\emptyset$ é uma cadeia de bits repleta de zeros, indicando o início da comunicação;
2. $bloco_n = (h_{n-1}, M_n)$, para $n \geq 1$, onde $h_i = Hash(bloco_{n-1})$

Em razão das propriedades dos algoritmos de hash criptográfico, qualquer alteração no dado de entrada leva a alterações no hash de saída do algoritmo. Dessa forma, cria-se um vínculo sequencial que assegura a integridade de cada registro individual e também da ordem dos registros. Ao mesmo tempo, se a última assinatura da cadeia for válida, isso é suficiente para garantir a irretratabilidade da conversa completa, de modo que assinaturas anteriores do mesmo usuário podem ser descartadas. Já assinaturas de outros usuários, que enviaram mensagens anteriores, devem ser preservadas para garantir a irretratabilidade da comunicação até aquele ponto da comunicação. Ao usar tal estrutura para mensagens trocadas entre usuários, permite-se auditorias confiáveis das conversas por qualquer pessoa que tenha acesso a uma sequência de mensagens trocadas, bastando reconstruir a cadeia e verificar os hashes e a assinatura final de cada usuário.

Caso deseje-se revelar apenas um trecho de mensagens, omitindo trechos anteriores e posteriores, a verificação de integridade do conteúdo revelado ainda é possível contanto que a assinatura digital relativa à última mensagem esteja disponível. Nesse caso, apenas o hash de trechos de mensagens omitidas ficaria disponível para análise, o que não permitiria a recuperação da mensagem em si exceto por eventuais ataques de força bruta (e.g., o teste de diversas mensagens possíveis).

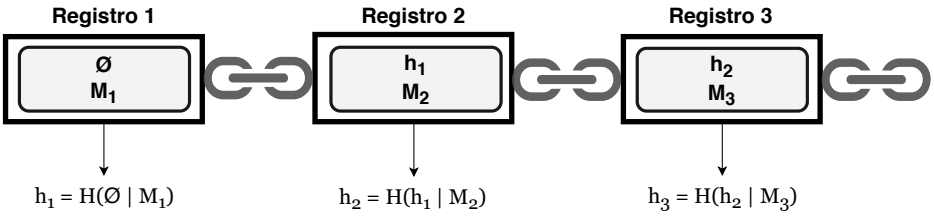

Figura 4. Representação do registro de uma conversa.

3.1. Análise

Suponha um pequeno trecho de conversa, como o ilustrado na Fig. 4. Nesse cenário de troca de mensagens instantâneas, os seguintes ataques são prevenidos pela estrutura proposta, assumindo que pelo menos um dos interlocutores é honesto.

- *Edição do conteúdo da mensagem*: Trocando a mensagem M_n por uma mensagem M'_n, onde $M_n \neq M'_n$, tem-se $h_n = H(h_{n-1}|M_n)$ e $h'_n = H(h_{n-1}|M'_n)$, de modo que $h_n \neq h'_n$. Ao calcular o hash do registro n com a mensagem editada M'_n, obtêm-se h'_n, que é diferente do valor h_n salvo no registro $n+1$ e assinado digitalmente. A Fig. 5 apresenta como exemplo um cenário em que M_2 é alterado para M'_2. Ao calcular o hash do registro 2, obtém-se h'_2, que diverge do valor h_2 salvo no registro 3 com esmagadora probabilidade, evidenciando uma alteração entre os registros 2 e 3.

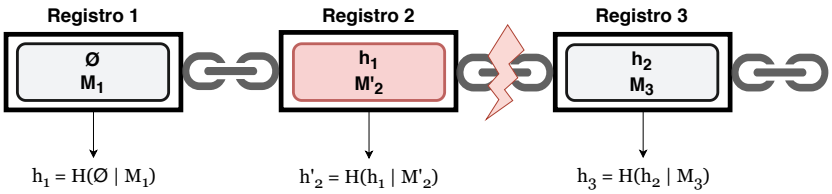

Figura 5. Conversa com uma mensagem editada.

- *Troca da ordem das mensagens*: Dada a sequência de mensagens M_{n-1}, M_n, M_{n+1} e M_{n+2}, suponha a troca de ordem entre as mensagens M_n e M_{n+1}. A alteração gera três pontos de divergência dos hashes: logo antes dos registros trocados, entre os registros trocados e logo após os registros trocados. A Fig. 6 apresenta como exemplo uma alteração de M_2 com M_3. Ao calcular e comparar os hashes da sequência, evidencia-se incompatibilidade entre os registros, conforme ilustrado.

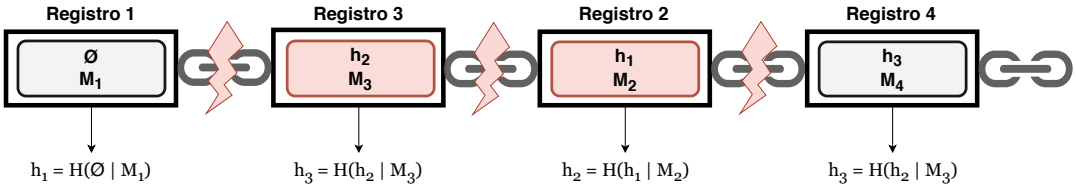

Figura 6. Conversa com a ordem da mensagens trocadas.

- *Remoção de mensagens*: Removendo a mensagem M_n da conversa, a sequência de registros saltaria de M_{n-1} para M_{n+1}. O registro M_{n-1} gera o hash h_{n-1}, enquanto o registro M_{n+1} apresenta salvo o hash h_n, assim evidenciando a alteração. A Fig. 7

apresenta como exemplo a remoção da mensagem M_2. Ao calcular o hash do registro 1, obtém-se h_1, que diverge do valor h_2 salvo no registro 3. Detecta-se, assim, uma alteração entre os registros 1 e 3.

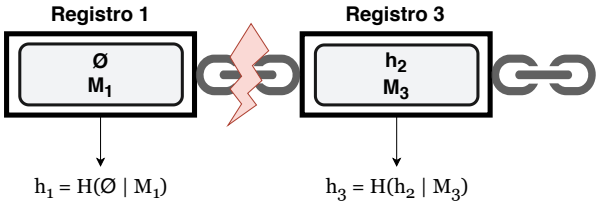

Figura 7. Conversa com uma mensagem apagada.

- *Inserção de mensagens*: Inserindo a mensagem $M_{n.5}$ na conversa, a sequência de registros seria M_n, $M_{n.5}$, e M_{n+1}. O novo registro da mensagem $M_{n.5}$ deve conter o hash h_n para manter a consistência da sequência. Porém a registro M_{n+1} também apresenta o hash h_n e não $h_{n.5}$, evidenciando a alteração. A Fig. 8 apresenta como exemplo a inserção da mensagem $M_{2.5}$. Ao comparar o hash do registro 2 com o salvo no registro 2.5, percebe-se que eles são compatíveis. Porém, o hash $h_{2.5}$ diverge do valor h_2 salvo no registro 3 evidenciando que há alguma alteração entre os registros 2.5 e 3.

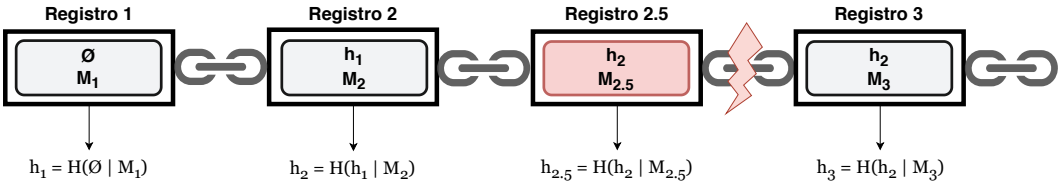

Figura 8. Conversa com uma mensagem inserida.

É relevante observar que, em todos os exemplos, a arquitetura é robusta para identificar violações de integridade da conversa, embora não possa (e não seja seu objetivo) reverter alterações detectadas. Além disso, o que garante que os hashes das mensagens alteradas não possam substituir os hashes originais é fato de que a mensagem no final da cadeia é sempre assinada digitalmente emissor daquela mensagem. Ao mesmo tempo, isso garante autenticidade e irretratabilidade às comunicações.

4. Prova de conceito e avaliação experimental

Como prova de conceito, a técnica proposta foi incorporada ao aplicativo de troca de mensagens distribuído descrito em [Komo et al. 2018]. Para isso, utilizou-se a biblioteca SpongyCastle, já presente no código do aplicativo em questão, e o pacote *java.security*. Foram usados os algoritmos SHA3-256 [NIST 2015] para o cálculo dos hashes do sistema, e o algoritmo ECDSA [Johnson et al. 2001] com a curva *secp256* [Research 2000] para assinaturas digitais, conferindo ao sistema um nível de segurança de 128 bits. A Fig. 9 mostra o modo de *debug* do sistema, em que cada mensagem é acompanhada do hash da mensagem anterior, em hexadecimal.

Um ponto importante a ser destacado é que, pelo fato do aplicativo base apresentar um estrutura distribuída de comunicação P2P, as mensagens ficam armazenadas apenas

Figura 9. Captura de tela de uma conversa no aplicativo desenvolvido.

localmente nos dispositivos dos interlocutores. Além disso, o aplicativo base usa criptografia fim-a-fim na comunicação. Assim, a privacidade dos interlocutores é garantida até que algum deles deseje a auditoria da conversa. Para isso, o usuário deve revelar as mensagens para terceiros, que então podem reconstruir a cadeia de hashes e verificar a integridade da conversa.

O aplicativo modificado para incorporar o mecanismo proposto foi testado entre um Samsung Galaxy J2 Prime e um Xiaomi Redmi Note 7 com sistemas operacionais *Android* versões 6 e 9, respectivamente, ambos conectados em uma rede sem fio local, usando um roteador TP–Link TL–WR542G. Foi então analisado no dispositivo Xiaomi Redmi Note 7 o tempo médio para assinar e verificar mensagens usando o ECDSA que ficaram respectivamente em 0,286 $\pm$ 0,196 ms e 0,723 $\pm$ 0,194 ms. Adicionalmente, foi então analisada a latência nas mensagens com a inserção dos hashes e assinaturas, capturando medidas numéricas de tempo desde o envio da mensagem até sua recepção, para mensagens com 5, 50, 100, 200, 500 e 1000 caracteres. Para cada tamanho de mensagem, foram realizados 10 envios, levando aos tempos médios de propagação e desvios padrão mostrados na Tabela 1.

Analisando os números obtidos, constata-se que, independentemente do número de caracteres contidos na mensagem, os tempos de propagação médios são próximos a 2 segundos em tempo absoluto, e próximos de 1 segundo em tempo relativo ao aplicativo original, algo que seria aceitável em um aplicativo de mensagens instantâneas.

5. Considerações finais

Este trabalho propõe o uso da técnica de hashes criptográficos encadeados para prover auditabilidade a conversa nos aplicativos de troca de mensagens instantâneas. O resultado é uma solução simples e de baixo custo computacional, que pode ser incorporada a diversos aplicativos existentes, sejam eles centralizados (e.g. WhatsApp, Telegram, Signal) ou distribuídos (e.g., Briar).

Tabela 1. Propagação das mensagens: médias, medianas e desvios padrão.

Tamanho da mensagem (#caracteres)	Tempo médio (ms)		Tempo mediano (ms)		Desvio padrão	
	app original	app modificado	app original	app modificado	app original	app modificado
5	1149,5	1893,6	1077,5	1871,5	230,0	119,3
50	1075,7	2387,9	1091,0	2014,0	41,5	801,4
100	1055,0	2895,0	1058,0	1932,0	37,1	1634,4
200	1089,8	2483,5	1087,5	1782,0	29,2	1490,6
500	1076,0	2002,2	1074,0	1939,0	31,7	162,3
1000	1072,4	1845,4	1069,5	1836,5	36,2	73,5

Os testes realizados com a prova de conceito desenvolvida indicam que, mesmo com o aumento de processamento para a geração e verificação de hashes e assinaturas, a latência total das mensagens mantém-se em níveis que não afetam significativamente a usabilidade do sistema. Como trabalho futuro, propõe-se incorporar a estrutura de cadeia de hashes em aplicativos de código aberto de ampla utilização, como Signal ou Briar, para validação da viabilidade da proposta e verificação do seu impacto computacional.

***Agradecimentos**: Este trabalho teve apoio da CAPES (Código de Financiamento 001), do CNPQ (Bolsa de Produtividade 301198/2017-9), da FAPESP (projeto 13/25977-7), da LG Electronics e da Ripple.*

Referências

[A. Mcgrew and Viega 2004] A. Mcgrew, D. and Viega, J. (2004). The galois/counter mode of operation (gcm).

[BB 2018] BB (2018). Clientes do BB podem acessar suas contas usando o WhatsApp [online]. `www.bb.com.br/pbb/pagina-inicial/imprensa/n/58382/`.

[Cohn-Gordon et al. 2017] Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., and Stebila, D. (2017). A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 451–466.

[Hu et al. 2005] Hu, Y.-C., Jakobsson, M., and Perrig, A. (2005). Efficient constructions for one-way hash chains. In *Proc. of the 3rd Int. Conf. on Applied Cryptography and Network Security (ACNS'05)*, pages 423–441, Berlin, Heidelberg. Springer-Verlag.

[Johnson et al. 2001] Johnson, D., Menezes, A., and Vanstone, S. (2001). The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63.

[Komo et al. 2018] Komo, A. E., Arakaki, B. O., Simplicio Jr., M. A., and Levy, M. R. (2018). Aplicativo de troca de mensagens instantâneas utilizando comunicação P2P. In *Anais Estendidos do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 65–72, Porto Alegre, RS, Brasil. SBC.

[LaMacchia et al. 2007] LaMacchia, B., Lauter, K., and Mityagin, A. (2007). Stronger security of authenticated key exchange. In Susilo, W., Liu, J. K., and Mu, Y., editors, *Provable Security*, pages 1–16, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Marlinspike and Perrin 2016] Marlinspike, M. and Perrin, T. (2016). The double ratchet algorithm [online]. whispersystems.org/docs/.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[NIST 2015] NIST (2015). *FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. National Institute of Standards and Technology.

[Research 2000] Research, C. (2000). Sec 2: Recommended elliptic curve domain parameters, version 1.0. http://www.secg.org/SEC2-Ver-1.0.pdf.

[Rogers et al. 2018] Rogers, M., Saitta, E., Grote, T., Dehm, J., Erlingsson, E., Tyers, B., and Grigg, J. (2018). Briar. https://briarproject.org/.

[Rösler et al. 2018] Rösler, P., Mainka, C., and Schwenk, J. (2018). More is less: On the end-to-end security of group chats in Signal, WhatsApp, and Threema. In *IEEE European Symposium on Security and Privacy (Euro S&P)*, pages 415–429.

[Schliep and Hopper 2018] Schliep, M. and Hopper, N. (2018). End-to-end secure mobile group messaging with conversation integrity and deniability. Cryptology ePrint Archive, Report 2018/1097. https://eprint.iacr.org/2018/1097.

[Schliep et al. 2017] Schliep, M., Kariniemi, I., and Hopper, N. (2017). Is Bob sending mixed signals? In *Proc. of the 2017 on Workshop on Privacy in the Electronic Society*, WPES '17, pages 31–40, New York, NY, USA. ACM.

[Schröder et al. 2016] Schröder, S., Huber, M., Wind, D., and Rottermanner, C. (2016). When signal hits the fan: On the usability and security of state-of-the-art secure mobile messaging. In *Proc. of the 1st European Workshop on Usable Security*.

[Simplicio et al. 2013] Simplicio, M., Oliveira, B., Margi, C., Barreto, P., Carvalho, T., and Näslund, M. (2013). Survey and comparison of message authentication solutions on wireless sensor networks. *Ad Hoc Networks*, 11(3):1221–1236.

[Simplicio et al. 2014] Simplicio, M., Santos, M., Leal, R., Gomes, M., and Goya, W. (2014). SecureTCG: a lightweight cheating-detection protocol for P2P multiplayer online trading card games. *Security and Communication Networks*, 7(12):2412–2431.

[Telegram 2019a] Telegram (2019a). MTProto mobile protocol [online]. core.telegram.org/mtproto.

[Telegram 2019b] Telegram (2019b). Telegram FAQ [online]. telegram.org/faq.

[TJDFT 2016] TJDFT (2016). Portaria Conjunta 67 de 08/08/2016. Tribunal de Justiça do Distrito Federal e dos Territórios. Disponível: www.tjdft.jus.br/publicacoes/publicacoes-oficiais/portarias-conjuntas-gpr-e-cg/2016/portaria-conjunta-67-de-08-08-2016.

[Torvalds 2005] Torvalds, L. (2005). Git. http://www.git-scm.com/.

[WhatsApp 2016] WhatsApp (2016). WhatsApp encryption overview [online]. www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf.

[WhatsApp 2019] WhatsApp (2019). Aplicativo WhatsApp Business [online]. www.whatsapp.com/business/.