COMPARAÇÃO DE APLICATIVOS MÓVEIS DE COMUNICAÇÃO SEGURA

Resumo: Esse trabalho fará a comparação de três aplicativos móveis de comunicação, notáveis tanto pela vasta popularidade, facilidade de uso e adesão, em diferentes graus, às boas práticas de segurança – WhatsApp, Signal e Telegram –, levando-se em conta aspectos como: modelo de negócio de suas companhias; garantias de segurança prometidas; adequação ao modelo de software livre; protocolos de criptografia; e relatos de falhas de segurança, sanadas ou não. Será analisada a documentação relativa aos protocolos de segurança, bem como seus componentes individuais, e suas vulnerabilidades encontradas na literatura. Uma análise crítica desses aplicativos é especialmente relevante uma vez que são ferramentas essenciais para ativistas, jornalistas e demais indivíduos e organizações que requerem alto nível de segurança durante o trânsito de informações. O trabalho conclui com uma exposição acerca da inadequação de todos os três aplicativos enquanto candidato sólido a aplicativo padrão de troca de mensagens seguras –usandose a filosofia da distribuição de Android denominada Securegen como parâmetro–, descrevendo os problemas encontrados em cada um, e possíveis soluções a alguns desses problemas.

Palavras-chave: aplicativos, criptografia, telefonia móvel, contravigilância

Comparación de aplicaciones móviles de comunicación segura

Resumen: Este trabajo compara tres aplicaciones de comunicación móvil, notables por su gran popularidad, facilidad de uso y adhesión, a diferentes niveles, a las buenas prácticas de seguridad – WhatsApp, Signal y Telegram – teniendo en cuenta aspectos como: modelo de negocio de sus empresas; garantías de seguridad prometidas; adecuación al modelo de software libre; la documentación relativa a los protocolos de criptografía; y informes de defectos de seguridad, corregidos o no. Será analizada la documentación de los protocolos de seguridad, así como sus componentes individuales, y las vulnerabilidades encontradas en la literatura. Un análisis crítico de esas aplicaciones es especialmente relevantes ya que son esenciales a los activistas, periodistas y demás personas y organizaciones que requieren un alto nivel de seguridad durante el tránsito de información. El documento concluye con una exposición acerca de la insuficiencia de todas las tres aplicaciones como candidato sólido para aplicación estándar de cambio de mensajería segura – tomando como parámetro la filosofía de la distribución de Android llamada Securegen –, describiéndose los problemas encontrados en cada uno, y posibles soluciones a algunos de estos problemas.

Palabras clave: aplicaciones, criptografía, telefonía móvil, contravigilancia

Comparison of secure mobile communication applications

Abstract: This paper will compare three mobile message applications, notable for their huge popularity, ease of use and adherence, to varying degrees, to best safety practices – WhatsApp, Signal, and Telegram –, taking into account aspects such as: business model of their respective companies; promised security guarantees; suitability to the free software model; encryption protocols; and reports of security flaws, fixed or not. It will examine the documentation related to their security protocols – both the architecture and the individual components – and respective vulnerabilities as found on literature. A critical analysis of those applications is especially relevant since they are essential tools for activists, journalists and other individuals and organizations that require a high level of security during the exchange of information. The paper concludes that all three selected applications fail as a solid candidate for standard secure mobile message application – using the philosophy behind Android distribution called Securegen as parameter –, describing issues found in each, and possible ways to solve some of these problems.

Keywords: applications, cryptography, mobile phones, countersurveillance

LEONARDO PICCIONI DE ALMEIDA* **MÁRCIO MORETTO RIBEIRO****

INTRODUÇÃO

As denúncias feitas em junho de 2013 pelo ex-funcionário da Agência Nacional de Segurança (NSA) estadunidense, Edward Snowden, sobre o esquema de vigilância em massa efetuado pela agência e seus parceiros, trouxe à tona o debate sobre gargalos da segurança da informação na comunicação digital (Greenwald, MacAskill & Poitras, 2013).

Em particular o programa PRISM indicou que o governo dos EUA tem acesso direto, seja consentido ou não, aos dados das grandes empresas de comunicação via web (Greenwald & MacAskill, 2013).

A concentração de dados vulneráveis nos servidores dessas empresas, portanto, seria um desses gargalos. Como resposta a essa ameaça, ou como oportunidade de negócios, não tardou a surgirem aplicativos de comunicação móvel que prometiam proteger o usuário desse tipo de vigilância.

Arcabouço teórico

Com o objetivo de auxiliar usuários a escolher entre esses aplicativos, a Electronic Frontier Foundation (EFF, 2016) propôs sete critérios úteis, mas não necessariamente suficientes, a serem analisados. Os critérios em si são, aqui, mais relevantes do que os conceitos dados pela EFF a cada aplicativo, principalmente porque não se distingue aplicativos que implementam bem ou mal tais critérios.

O primeiro critério é a "criptografia em trânsito", que evita a decifração da mensagem, caso seja interceptada a caminho do destinatário da mensagem ou de um servidor intermediário.

O segundo critério, mais sofisticado, é a chamada "criptografia ponta a ponta", ou "criptografia fim a fim", ou seja, a inviabilidade de qualquer servidor intermediário de decifrar a mensagem. Num esquema de criptografia ponta a ponta, idealmente, apenas remetente e destinatário têm acesso às mensagens. Além de preservar a privacidade do usuário, tal atitude evita a existência de um "ponto único de falha": o acesso ao servidor intermediário por indivíduos ou organizações maliciosos pode resultar no vazamento do conteúdo de bilhões de mensagens.

O terceiro critério é a capacidade de um usuário verificar a identidade de um outro usuário. Na prática, isso significa que remetente e destinatário – mas não necessariamente terceiros – podem verificar a identidade um do outro, afastando a possibilidade de fraude.

O quarto critério é a "*perfect forward secrecy*", isso é, a impossibilidade de decifrar todas as mensagens de uma conversa caso o segredo usado para decifrar uma mensagem específico seja descoberto.

O quinto critério se refere à possibilidade de auditoria independente do código-fonte. Uma generalização desse critério é o conceito de software livre, proposto por Stallman (2015:3), que

prevê quatro direitos (as chamadas "quatro liberdades") ao usuário: o direito de utilizar o programa para qualquer fim; o direito a estudar – o que inclui auditar – o software e modificá-lo; o direto de distribuir independentemente o software; e o direito de distribuir quaisquer modificações que o usuário tenha feito no mesmo.

A possibilidade de auditoria independente permite verificar se, e como, os mecanismos de segurança são implementados. Combinada com a noção de software livre, possibilita ao usuário distribuir versões do software original que corrija quaisquer vulnerabilidades de segurança encontradas.

Completam a lista de critérios o sexto, relacionado a uma documentação abrangente dos mecanismos de segurança, e o sétimo, que avalia se houve uma auditoria recente do código.

É importante ressaltar que o placar da EFF previa apenas o cumprimento ou não dos critérios, não havendo graduação dentro de cada nível ou outra maneira de mensurar as mazelas de cada aplicativo (Zorz, 2016). Tal debilidade é, inclusive, razão pelo abandono do placar, ainda sem substituto (EFF, 2016).

No último placar atualizado, os três principais aplicativos de comunicação móvel apresentavam os seguintes conceitos: Signal apresentou conceito máximo; WhatsApp não pontuou no quinto critério; e o Telegram apresentou nota máxima em sua versão criptografada ponta a ponta, enquanto que o modo padrão do Telegram não pontuou do segundo ao quarto critérios. Contudo, como o trabalho pretende demonstrar, todos os aplicativos apresentam mazelas que o inviabilizam como solução para os sete critérios.

SIGNAL

O aplicativo Signal, originalmente chamado TextSecure, surgiu como um aplicativo de troca de SMS/MMS criptografados, mas seu foco mudou ao longo do tempo. Desde 2015, as mensagens são enviadas criptografadas ponta a ponta através de rede de telefonia celular, se o destinatário possui Signal instalado; ou através de SMS/MMS não criptografado, caso contrário (Marlinspike, 2015).

O aplicativo é mantido pela organização não governamental Open Whisper Systems (OWS), mantida por doações, financiamento governamental e, possivelmente, ganhos da venda de sua empresa antecessora, a*start-up* Whisper Systems, adquirida pelo Twitter (Greenberg, 2011).

O código-fonte, tanto dos aplicativos clientes quanto dos servidores, é disponibilizado como software livre. O aplicativo possui clientes para Android e iOS, bem como uma extensão para Google Chrome/Chromium.

O aplicativo implementa o protocolo Signal (anteriormente conhecido como Axolotl ou TextSecure), desenvolvido pelo fundador da OWS, Moxie Marlinspike, mas aberto ao público e de ampla adoção – softwares proprietários que afirmam implementar o protocolo incluem WhatsApp, Facebook Messenger e Google Allo.

Pode-se entender o protocolo Signal como uma evolução do protocolo OTR (*off-the-records*), apresentando aprimoramentos em relação a esse, como maior plausibilidade de negação (Marlinspike, 2013b) e segurança retroativa (“*future secrecy* ”) (Marlinspike, 2013a).

Arquitetura do protocolo

Para efeitos de clareza e concisão, considere um cenário em que Alice pretende iniciar uma comunicação com Bob[1].

Para poderem se comunicar em segurança, Alice e Bob precisam compartilhar um segredo, que será usada para cifrar e decifrar as mensagens. Porém, o canal de comunicação usado para a trocas desse segredo pode ser inseguro – por exemplo, grampeado. Dessa forma, Alice e Bob necessitam de um artifício para que, mesmo que o segredo não trafegue às claras pelo canal, ambos – e, necessariamente, nenhum terceiro – obtenham o mesmo segredo ao final. O mais usado desses artifícios é o protocolo de Diffie-Hellman, usado pelo Signal.

Além das chaves privadas transmitidas de maneira opaca pelo protocolo de Diffie-Hellman, é necessário que Alice e Bob tenham uma chave pública cada, únicas e disponíveis para consulta por todos. Após Alice requisitar ao servidor a chave pública de Bob, ela consegue obter três segredos distintos: um combinando a chave pública dela e a chave privada de Bob, um com a chave privada dela e a chave pública dele, e um com as chaves privadas de ambos.

Uma característica desse arranjo particular é que, uma vez que Bob possa checar a chave pública de Alice e ambos obtenham os mesmos três segredos, Bob pode se assegurar da autenticidade de Alice. Contudo, como um terceiro não pode provar que as chaves privadas enviadas por Diffie-Hellman não tenha sido forjadas, não é possível garantir que a conversa como um todo não tenha sido forjada – essa garantia é a chamada “plausibilidade de negação”.

Esses três segredos serão combinados para a produção de uma “chave raiz”. Junto à primeira chave raiz, Alice enviará uma nova chave privada, que será usada na geração da próxima chave raiz e uma “chave corrente”.

A chave corrente será usada para a geração da próxima chave corrente, bem como de uma “chave mestra”.

A partir da chave mestra, é possível gerar outros dois segredos: o primeiro será usado para cifrar a mensagem; o segundo para se gerar um MAC (*message authentication code*) a partir da cifra, usado para se garantir a autenticidade da mensagem.

Junto com sua primeira resposta a Alice, Bob enviará uma nova chave privada, que será usada para gerar a próxima chave raiz e as chaves conseguintes. Analogamente, durante toda a comunicação, novas chaves privadas serão enviadas cada vez que um dos lados enviar um conjunto de mensagens e, partindo delas, serão obtidas novas chaves raiz, etc.

Esse esquema de geração de chaves, que pode parecer à primeira vista supérfluo, é útil pois permite que, uma vez que a chave mestra seja gerada, as chaves corrente e raiz que a antecederam possam ser descartadas sem prejuízos.

Como o algoritmo de geração de chaves é irreversível, uma vez que uma chave mestra tenha sido corrompida, ela não pode ser usada para a geração de outras chaves correntes e raiz – portanto, provendo ao protocolo*future secrecy* . Portanto, para se decifrar outras mensagens, as únicas informações que Alice precisa armazenar é sua última chave privada enviada a Bob, a última chave raiz e a chave mestra das mensagens ainda não decifradas – a comunicação celular é assíncrona, de modo que as mensagens nem sempre chegam na ordem em que foram enviadas.

Além das maneiras descritas acima, a autenticidade de Alice e Bob podem ser conferidas presencialmente, através do conferimento de um*fingerprint* de suas chaves.

Primitivas criptográficas

Primitivas criptográficas são algoritmos básicos, confiáveis e bem documentados, que podem ser combinados de forma a se obter um protocolo de criptografia.

Para a troca de chaves, o protocolo Signal utiliza o protocolo de Diffie-Hellman, ou de Diffie-Hellman-Merkle. Considerando$a$ a chave privada de Alice e b a chave privada de Bob, uma troca de chaves de Diffie-Hellman consiste no envio de g e ga por Alice, seguidas pelo envio de gb por Bob, de forma que $(ga)b = (gb)^a = gab$. Se não for factível formar gab apenas com g , ga e gb , Alice e Bob podem usar gab como segredo para um esquema de criptografia simétrica (Menezes, van Oorschot & Vanstone, 1997: 515).

Para se garantir que não seja factível obtergab a partir de g , ga e gb , é necessário que g , a e b pertençam a um conjunto de números em que o problema da fatoração seja computacionalmente difícil – no caso do Signal, o conjunto utilizado é a curva elíptica Curve25519, proposta por Bernstein (2006).

A geração de chaves usa ora HMAC, ora HKDF, a depender da quantidade de chaves a serem geradas a cada passo. HMAC (*hash-based MAC*) é um algoritmo originalmente utilizado para geração de MACs, mas que pode ser usado para gerar entropia. HKDF (*HMAC-based key derivation function*) é um algoritmo que utiliza HMAC para geração de chaves seguras de tamanho arbitrário. Ambos os algoritmos são irreversíveis e considerados confiáveis tanto por desenvolvedores quanto por acadêmicos.

O HMAC e, consequentemente, o HKDF do protocolo Signal está baseado no uso do algoritmo SHA-256, também conhecida como SHA-2, desenvolvida pela NSA e recomendada pelo*National Institute of Standards and Technology* estadunidense (NIST, 2012). Atualmente, não há ataques conhecidos contra o algoritmo completo, mas apenas contra algumas simplificações.

As mensagens são cifradas e decifradas utilizando a cifra de bloco AES (*Advanced Encryption Standard*), também conhecida como *Rijndael* – outro padrão NIST (2001) –, em modo contador (Diffie & Hellman, 1979). Atualmente, não há ataques conhecidos com ganhos significativos sobre a força bruta. Para a geração de MACs, também é utilizado o algoritmo HMAC.

O protocolo Signal utiliza a consagrada tática*encrypt-then-mac* – isso é, cifrar a mensagem para, em sequência, obter um MAC da mensagem cifrada. Segundo Bellare & Namprempre (2000), essa estratégia comprovadamente providencia autenticidade, indistinguibilidade e não maleabilidade à mensagem cifrada.

WHATSAPP

WhatsApp é o aplicativo de troca de mensagens mais popular do mundo, somando mais de um bilhão de usuários. Desde 2014, o aplicativo pertence ao gigante das redes sociais Facebook. Possui versões para todas as principais plataformas para aparelhos celulares, bem como algumas legadas, e também versão para navegadores.

Seu código-fonte é fechado e proprietário, e o aplicativo lucra através da exploração da privacidade dos usuários, repassando dados, inclusive, para sua companhia mãe.

O WhatsApp implementa o protocolo Signal desde abril de 2016 (Marlinspike, 2016). Sua implementação do protocolo, supostamente, é bastante parecida com a do aplicativo Signal (WhatsApp, 2016). Contudo, é impossível analisar satisfatoriamente a implementação, uma vez que o código fonte do aplicativo é fechado.

Diferenças em relação à implementação no aplicativo Signal serão tratadas na subseção seguinte.

Protocolo

Na implementação do WhatsApp, Bob deve carregar um lote de "pré-chaves" no servidor do WhatsApp. Cada pré-chave é descartada após um único uso e, como cabe a Bob ativamente reenviar outro lote de pré-chaves cada vez que o lote anterior tenha se esgotado, é possível que, por determinados períodos de tempo, o servidor contenha nenhuma de suas pré-chaves.

Ao iniciar as trocas de chave com Bob, Alice recebe, além das chaves pública e privada de Bob, uma de suas pré-chaves, se disponível. Se Alice não receber nenhuma pré-chave de Bob, a geração da primeira "chave raiz" ocorre tal como na implementação do aplicativo Signal. Em caso contrário, a pré-chave é utilizada, junto com os outros três segredos compartilhados, para a geração da chave raiz.

TELEGRAM

O Telegram é um aplicativo sem fins lucrativos, mantido pela fortuna de seu fundador, o multimilionário russo Pavel Durov, também fundador e ex-CEO da rede social VK (originalmente,

Vkontakte). No entanto, o aplicativo não rejeita a possibilidade de sobreviver de doações ou funcionalidades pagas após o fim das reservas monetárias de Durov.

Durov decidiu fundar o Telegram após se negar, repetidas vezes, a entregar dados de oposicionistas políticos, usuários do VK, a autoridades russas (Salomão, 2015). A criptografia utilizada pelo Telegram, supostamente, seria usada como forma de tornar os dados de seus usuários opacos à inteligência russa e de outros países. Desde 2014, Durov vive exilado em São Cristóvão e Névis, no Caribe.

O aplicativo possui clientes, oficiais ou não oficiais, para quase todas as maiores plataformas, sejam*desktop* ou móvel, e também para navegadores. Os clientes oficiais são disponibilizados como software livre. O código do servidor, no entanto, é fechado – sua publicação não é considerada prioridade, atualmente.

O Telegram utiliza um protocolo próprio, o MTProto, desenvolvido por funcionários da empresa sem treinamento prévio em criptografia, de maneira que o protocolo sofreu algumas alterações ao longo de sua curta história, corrigindo problemas assim que encontrados. O Telegram oferece prêmios em dinheiro para quem conseguir quebrar a criptografia do aplicativo, mas as regras do concurso são consideradas muito restritas (Hornby, 2013).

O protocolo MTProto vem em dois sabores: com ou sem criptografia ponta a ponta. Por padrão, as conversas do Telegram não apresenta criptografia de ponta a ponta, mas apenas durante trânsito. No caso de conversas em grupo, criptografia ponta a ponta não é atualmente disponível.

Arquitetura do protocolo

O protocolo MTProto (Telegram, 2016a; 2016b) é considerado significativamente mais simples que o Signal, mas também mais heterodoxo. A explicação a seguir considera a versão criptografada ponta a ponta, teoricamente mais segura.

Para iniciar a conversa, Alice e Bob começam a compartilhar um segredo, enviando suas chaves secretas, de maneira opaca, utilizando-se o protocolo de Diffie-Hellman.

O MAC da mensagem é calculado a partir da mensagem. A partir do MAC e do segredo compartilhado, é gerada outra chave, que será usada para cifrar a mensagem. Então, a mensagem cifrada e seu MAC são enviadas para o canal.

Para prover*perfect forward secrecy* , uma nova troca de chaves de Diffie-Hellman ocorre a cada uma semana ou cem mensagens, o que ocorrer primeiro, exceto se a chave não tenha sido usada nenhuma vez.

A única forma de se conferir a identidade de Alice ou Bob é através de verificação de*fingerprint* , presencialmente.

Primitivas criptográficas

Diferentemente do protocolo Signal, que utiliza uma curva elíptica, o gerador utilizado pelo MTProto para a troca de chaves de Diffie-Hellman é um gerador de números primeiros de alta ordem. Há relatos de manipulação pela NSA na implementação de geradores de números primos (Adrian et al., 2015), o que, na prática, tornaria o uso de Diffie-Hellman pouco efetivo.

O MAC é gerada utilizando-se o SHA-1, um antecessor do SHA-2, atualmente considerado vulnerável (Schneier, 2004). Contudo, o uso original do SHA-1 é cifragem, e não geração de MAC, o que torna incerto a relevância das vulnerabilidades conhecida – tal qual a descrita em Stevens (2012).

A geração da chave de cifragem a partir do segredo compartilhado por Diffie-Hellman utiliza uma função de derivação de chaves desenvolvida pelos próprios desenvolvedores do protocolo, sendo pouco estudada e praticamente não utilizada por nenhum outro protocolo.

A cifragem também utiliza AES. Contudo, diferentemente do protocolo Signal, ela utiliza um modo bem mais raro, o IGE (*Infinite Garble Extension*). Até por ser raramente usado e estudado, não há vulnerabilidades conhecidas em relação à forma como o protocolo utiliza IGE.

Uma característica particular do modo IGE é que a alteração da mensagem cifrada necessariamente causará alterações na decifração da mensagem, o que pode ser utilizado como indício da autenticidade da mensagem. Contudo, esse uso do modo IGE não apresenta todas as propriedades comprovadas pelo paradigma*encrypt-then-mac* , não adotado pelo MTProto.

CONCLUSÃO

Securegen (2016) é uma distribuição Android, focada em três pilares valiosos para aqueles que necessitam comunicar-se seguramente em dispositivos móveis: software livre, uso de criptografia forte e máximo respeito à privacidade do usuário.

Ao analisar os três aplicativos apresentados, contudo, pode-se perceber que nenhum deles adota satisfatoriamente esses três pilares.

A solução pelo WhatsApp é, talvez, a hipótese mais facilmente descartada. Por ser software fechado, não é possível uma auditoria pública do código e, se fosse necessário, correção do mesmo por qualquer usuário. Não é possível falar de comunicação móvel segura sem apelar para as liberdades do software. O WhatsApp também é notavelmente débil em relação à privacidade do usuário, coletando muito mais metadados dos usuários e mensagens do que o aplicativo Signal (Lee, 2016), e pouco se sabe sobre como o Facebook utiliza os mesmos.

O Signal, por sua vez, é campeão do respeito à privacidade do usuário, e possui o protocolo mais sólido e consolidado – praticamente não apresenta alterações desde 2014, quando o suporte a mensagens SMS/MMS criptografadas foi removido. Problemas de implementação existem,

mas são raros e, quando descobertos, rapidamente resolvidos. Exemplos passados incluem um*bug* que fazia com que as mensagens fossem mandadas em texto plano, ou seja, sem uso de criptografia (OWS, 2014a), e outro que permite a leitura das mensagens decifradas na tela de bloqueio do aparelho (OWS, 2014b).

Seu uso enquanto software livre, todavia, é prejudicado: apesar de exemplarmente disponibilizar seu código como software livre tanto o código do servidor quanto do cliente – situação única entre os três –, sua instalação na plataforma Android requer o uso de serviços fornecidos pelo Google, o que obriga seus usuários nessa plataforma a aceitar os nebulosos termos de serviço daquela companhia, inviabilizando, por exemplo, sua distribuição junto a projetos como o Securegen.

Uma solução possível para esse impasse seria uma versão do protocolo Signal que substituísse o uso dos serviços do Google nessa implementação por outras tecnologias. Tentativa nesse sentido incluíram o LibreSignal (n.d.), que utilizava o padrão aberto WebSocket para comunicação. Contudo, seu desenvolvimento foi abandonado em definitivo.

Apesar de o código do servidor Telegram ser fechado, ele fornece uma bem documentada API pública, que permite o desenvolvimento de clientes não oficiais com relativa facilidade. O código de seus clientes oficiais são abertos e livres.

O protocolo do Telegram, apesar de simples, é bastante heterodoxo: sua escolha por primitivas obscuras ou obsoletas, a decisão de não usar o paradigma*encrypt-then-mac* e, no caso da derivação de chaves, praticamente "reinventar a roda" usando um algoritmo próprio, sem qualquer justificação teórica ou empírica, em vez de uma primitiva criptográfica em que se pudesse confiar, são motivos mais que suficientes para requerer cautela a qualquer usuário. Apesar de o uso do protocolo MTProto não ter ainda sido descartado para comunicação segura, pouco há para se confiar em suas decisões de projeto.

Devido a relativa imaturidade do protocolo, ainda é comum se encontrar vulnerabilidades no protocolo, com reações distintas por parte do Telegram. Dois ataques teóricos, descritos por Jakobsen & Orlandi (2015), que comprometem a indistinguibilidade do texto cifrado (IND-CCA), foi desconsiderado pela equipe de engenheiros, porque ainda não afeta a segurança prática do protocolo. Nas palavras do próprio Telegram (2016a):

> Um funcionário do correio pode escrever 'Haha' (usando tinta invisível!) no lado de fora de um envelope selado que ele entrega para você. Isso não impede o envelope de ser entregue, isso não permite-o alterar o conteúdo do pacote, e isso não permite-o ver o que há do lado de dentro.

O próprio esquema de*fingerprints* do Telegram, utilizado para garantir a identidade dos usuários, já sofreu um ataque teórico, descrito por Rad (2015). O ataque partia das premissas de que, como o *fingerprint* é apenas um "resumo" da chave, e não a chave em si, é muito mais fácil encontrar colisões entre *fingerprints* do que entre chaves – 2^{-65} contra 2^{-128}. Desde então, o Telegram alterou a forma como obtém os *fingerprints* (Telegram, 2015), tornando o ataque

infactível perante o poder computacional atual. Ataques dessa natureza, causados pela má escolha de primitivas pelo Telegram, têm chances altas de se repetirem, o que poderiam ter sido evitado pela formulação de um protocolo mais ortodoxo, a princípio.

Outro ataque do Telegram, este aqui ainda não resolvido e com claras implicações práticas, ocorre porque a vasta maioria dos usuários do aplicativo não utilizam autenticação em dois passos. Como, no Telegram, a autenticação entre aparelhos ocorre a partir de envio de mensagens SMS, alguém com poder sobre a rede SMS – como governos e companhias de telefonia – poderiam interceptar um SMS de autenticação requisitado por Alice, ou até se passar por ela, e ter acesso a todas as suas conversas Rad (2015). Esse ataque pode ser contornado com autenticação de dois passos, mas pouco foi feito para se encorajar tal prática.

É preciso ponderar, ainda, quão prudente é lançar o código de um protocolo em pleno desenvolvimento ao público, onde pode ser acessado e estudado por indivíduos e órgãos mal intencionados.

Outro problema do Telegram é referente à grande quantidade de metadados enviados aos seus servidores, por padrão, que podem ser utilizados para rastrear um usuário, comprometendo sua privacidade (Flisbäck, 2015).

Contudo, suas boas vindas em relação ao desenvolvimento de aplicações derivadas torna o Telegram um candidato para aplicações que pretendem melhorar a segurança dos usuários. Alguns problemas que impedem um uso mais abrangente do usuário é o fato de que, por padrão, as conversas são criadas sem criptografia ponta a ponta e autodestruição – o que seria perfeitamente possível de se consertar no cliente –, a indisponibilidade de conversas criptografadas ponta a ponta para grupos e a não obrigatoriedade da autenticação em dois passos.

Conclui-se, dessa forma, que nenhum dos três aplicativos cumprem, no estado atual, todos os requisitos postos pelo Securegen. No entanto, soluções para alguns desses problemas podem ser achadas. O campo de aplicativos móveis seguros ainda necessitam muito da solução de muitos impasses colocados pelo*status quo* atual.

NOTAS

* Aluno do bacharelado em Sistemas de Informação da EACH-USP (leopiccionia@gmail.com).

** Professor doutor da EACH-USP (marciomr@usp.br).

1. Esta seção é baseada em OWS (2015).

REFERÊNCIAS

Adrian, D. et al. (2015). Imperfet Forward Secrecy: How Diffie-Hellman Fails in Practice. In*Proceeding of the 22nd ACM SIGSAG Conference on Computer and Communications Security* , pp. 5-17. ACM. Retrieved from https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf

Bellare, M., & Namprempre, C. (2000). Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In*Advances in Cryptology – ASIACRYPT 2000* , pp. 531-545. Springer Berlin Heidelberg. Retrieved from http://cseweb.ucsd.edu/ mihir/papers/oem.pdf

Bernstein, D. J. (2006). Curve25519: new Diffie-Hellman speed records. In*Proceedings of PKS 2006, to appear* . Retrieved from http://cr.yp.to/papers.html#curve25519

Diffie, W., & Hellman, M. E. (1979). Privacy and authentication: An introduction to cryptography.*Proceeding of the IEEE* , 67(3), 397-427.

Electronic Frontier Foundation. (2016).*Secure Messaging Scorecard* . Retrieved from https://www.eff.org/node/82654

Flisbäck, O. (2015).*Stalking anyone on Telegram* . Retrieved from https://oflisback.github.io/telegram-stalking/

Greenberg, A. (2011, Nov 28). Twitter Acquires Moxie Marlinspike's Encryption Startup Whisper Systems.*Forbes* . Retrieved from http://www.forbes.com/sites/andygreenberg/2011/11/28/twitter-acquires-moxie-marlinspikes-encryption-startup-whisper-systems

Greenwald, G., & MacAskill, E. (2013, Jun 07). NSA Prism program taps in to user data of Apple, Google and Others.*The Guardian* . Retrieved from https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data

Greenwald, G., MacAskill E., & Poitras, L. (2013, June 11). Edward Snowden: the whistleblower behind NSA surveillance revelations.*The Guardian* . Retrieved from https://www.theguardian.com/world/2013/jun/09/edward-snowden-nsa-whistleblower-surveillance

Hornby, T. (2013, Dec 19). Telegram's Cryptanalysis Contest. *Crypto Fails* [Blog post]. Retrieved from http://www.cryptofails.com/post/70546720222/telegrams-cryptanalysis-contest

Jakobsen, J. B., & Orlandi, C. (2015).*A practical cryptanalysis-of-the Telegram messaging protocol* . Master Thesis. Aarhus University. Retrieved from http://cs.au.dk/ jakjak/master-thesis.pdf

Lee, M. (2016, Jun 22). Battle of the Secure Messaging Apps: How Signal Beats WhatsApp.*The Intercept* . Retrieved from https://theintercept.com/2016/06/22/battle-of-the-secure-messaging-apps-how-signal-beats-whatsapp/

LibreSignal. (n.d.). LibreSignal: the truly private and Google-Free messenger for Android.*GitHub.* Retrieved from https://github.com/LibreSignal/LibreSignal

Marlinspike, M. (2013a, Nov 26). Advanced cryptographic ratcheting [Blog post].*Open Whisper Systems* . Retrieved from https://whispersystems.org/blog/advanced-ratcheting/

Marlinspike, M. (2013b, Jul 27). Simplifying OTR deniability [Blog post].*Open Whisper Systems* . Retrieved from https://www.whispersystems.org/blog/simplifying-otr-deniability/

Marlinspike, M. (2015, Mar 06). Saying goodbye to encrypted SMS/MMS [Blog post].*Open Whisper Systems.* Retrieved from https://whispersystems.org/blog/goodbye-encrypted-sms/

Marlinspike, M. (2016, Apr 05). WhatsApp's Signal Protocol Integration is now complete [Blog post].*Open Whisper Systems* . Retrieved from https://whispersystems.org/blog/whatsapp-complete/

Menezes, A. J., van Oorschot, P. C, & Vanstone, S. A. (1997).*Handbook of Applied Cryptography* . US: CRC Press.

National Institute of Standards and Technology. (2001).*Advanced Encryption Standard (AES)* . Springfield, VA. Retrieved from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

National Institute of Standards and Technology. (2012).*Secure Hash Standard (SHS)* . Gaithersburg, MD. Retrieved from http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

Rad, A. (2015, Jan 09).*A 264Attack On Telegram, and Why a Supper Villain Doesn't Need It To Read Your Telegram Chats* [Blog post]. Retrieved from https://www.alexrad.me/discourse/a-264-attack-on-telegram-and-why-a-super-villain-doesnt-need-it-to-read-your-telegram-chats.html

Salomão, K. (2015, Dec 20). Pavel Durov, o polêmico fundador do aplicativo Telegram.*Exame* . Retrieved from http://exame.abril.com.br/negocios/pavel-durov-o-polemico-fundador-do-aplicativo-telegram/

Schneier, B. (2004, Aug 19). Opinion: Cryptanalysis of MD5 and SHA: Time for a new standard [Blog post].*Computerworld.* Mountain View, CA. Retrieved from http://www.computerworld.com/article/2566208/security0/opinion--cryptanalysis-of-md5-and-sha--time-for-a-new-standard.html

Securegen. (2016). Página inicial. Retrieved from http://securegen.org/

Stallman, R. M. (2015).*Free Software, Free Society* (3rd ed.). Boston, MA: Free Software Foundation.

Stevens, M. (2012).*Attacks on Hash Functions and Applications* . Amsterdam. Retrieved from https://marc-stevens.nl/research/papers/PhD Thesis Marc Stevens - Attacks on Hash Functions and Applications.pdf

Open Whisper Systems. (2014a). Forward SMS not being encrypted when not touching text.*GitHub* . Retrieved from https://github.com/WhisperSystems/Signal-Android/issues/1073

Open Whisper Systems. (2014b). Reading “Encrypted” Messages without Password.*GitHub* . Retrieved from https://github.com/WhisperSystems/Signal-Android/issues/1716

Open Whisper Systems. (2015). ProtocolV2.*GitHub* . Retrieved from https://github.com/JavaJens/TextSecure/wiki/ProtocolV2

Telegram. (2015, Apr 08).*Active Sessions and Two-Step Verifications* [Blog post]. Retrieved from https://telegram.org/blog/sessions-and-2-step-verification

Telegram. (2016a).*FAQ for the Technically Inclined* . Retrieved from https://core.telegram.org/techfaq

Telegram. (2016b).*MTProto Mobile Protocol* . Retrieved from https://core.telegram.org/mtproto

WhatsApp. (2016).*WhatsApp Encryption Overview* . Retrieved from https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

Zorz, Z. (2016, Aug 11). How the EFF was pushed to rethink its Secure Messaging Scorecard.*Help Net Security.* Retrieved from https://www.helpnetsecurity.com/2016/08/11/eff-secure-messaging-scorecard/