Aplicativo de Troca de Mensagens Instantâneas Utilizando Comunicação P2P

Andrea E. Komo[1], **Bruno O. Arakaki**[1], **Marcos A. Simplicio Jr.**[1], **Mayer R. Levy**[1]

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

{andrea.komo,bruno.oliveira.arakaki,msimplicio,mayer.levy}@usp.br

Abstract. *Mobile messaging apps based on highly centralized architectures have some security vulnerabilities. In particular, they are prone to denial-of-service if the centralized server is taken down; also, if the server is responsible for managing the users' public keys (e.g., as in WhatsApp), it can engage in Man-in-the-Middle attacks by distributing fake keys. Aiming to address both issues, this work describes messaging app built upon a peer-to-peer architecture. The proposal combines a DHT network to store the users' connection information and a PGP based web of trust to validate the users' public keys. As a result, the messaging app hereby presented creates a framework for improving the availability and security of existing, centralized messaging apps.*

Resumo. *Aplicativos de troca de mensagens instantâneas baseados em arquiteturas centralizadas apresentam algumas vulnerabilidades de segurança. Em particular, eles são propensos à negação de serviço, se o servidor central ficar indisponível; ademais, se o servidor for responsável pelo gerenciamento das chaves públicas dos usuários (e.g.,WhatsApp), este pode realizar um ataque de homem-no-meio distribuindo chaves falsas. A proposta deste trabalho combina uma rede DHT para armazenar as informações de conexões dos usuários e uma rede de confiança baseada em PGP para validar as chaves públicas. Como resultado, o aplicativo apresentado cria uma estrutura que melhora a disponibilidade e a segurança em relação aos aplicativos centralizados existentes.*

1. Introdução

Com a popularização dos *smartphones*, aplicativos de troca de mensagens instantâneas têm sido muito utilizados tanto para conversas pessoais como profissionais. No Brasil, por exemplo, a popularidade do aplicativo *WhatsApp* é tamanha que seu bloqueio por ordem judicial em 2015/16 foi bastante discutido por usuários e juristas [Barreto and Lima 2016].

Embora bastante úteis, a maioria dos aplicativos de comunicação móvel atuais segue uma arquitetura centralizada, o que costuma levar a vulnerabilidades importantes. Uma delas refere-se à criação de pontos únicos de falha lógica, que podem ser explorados em ataques de negação de serviço (*Denial-of-Service* – DoS) e, assim, afetar a disponibilidade do serviço. Outro potencial problema é que a confiança intrinsecamente depositada em um servidor central pode permitir abusos, como a sua interferência na verificação da autenticidade dos usuários [Schrittwieser et al. 2012]. Por exemplo, mesmo em aplicativos com suporte a segurança fim-a-fim, como o *WhatsApp*, os usuários obtém as chaves

públicas uns dos outros por intermédio de um servidor central, que age como a autoridade certificadora no sistema. Desta forma, ao informar uma chave errada, tal servidor torna-se capaz de realizar ataques de homem-no-meio (*Man-in-the-Middle*, ou MitM), ao menos até os usuários fazerem uma verificação dessas chaves por meio de um canal independente. Infelizmente, muitos usuários não têm consciência desses riscos e potenciais danos resultantes da exposição de suas informações pessoais, de modo que poucos têm o cuidado de fazer esse tipo de verificação de segurança [Abu-Salma et al. 2017].

Com o objetivo de mitigar essas vulnerabilidades, propõe-se neste trabalho uma arquitetura P2P (*peer-to-peer*) para construção de apps de comunicação instantânea com suporte a segurança fim-a-fim. A arquitetura proposta combina uma rede DHT (*Distributed Hash Table*) [Maymounkov and Mazières 2002], responsável pelo armazenamento das informações de conexão dos usuários, com uma rede de confiança baseada em PGP (*Pretty Good Privacy*) [Zimmermann 1995] para garantir a autenticidade dos usuários e a proteção das mensagens trocadas entre eles.

O resto deste documento está organizado como segue. A Seção 2 apresenta alguns trabalhos relacionados à segurança dos aplicativos de comunicação móvel populares atualmente. A Seção 3 descreve a fundamentação teórica que embasa a arquitetura da solução proposta, a qual é então descrita na Seção 4. A Seção 5 apresenta informações técnicas e as principais funcionalidades do aplicativo, além de especificar a demonstração planejada para o salão de ferramentas do SBSeg 2018. E a Seção 6 conclui o trabalho.

2. Trabalhos Relacionados

O *Telegram*, lançado em 2013, foi um dos primeiros aplicativos de comunicação móvel que surgiu com a proposta de permitir comunicações seguras fim-a-fim entre os usuários, um diferencial com relação a concorrentes mais antigos, como o *WhatsApp*. Embora tal mecanismo não seja habilitado por padrão, o aplicativo permite que usuários criem um "chat secreto", de modo que nem mesmo o servidor do *Telegram* tenha acesso ao conteúdo das mensagens [Telegram 2018]. Um ponto negativo, entretanto, é que o *Telegram* utiliza algoritmos de segurança de autoria própria, e não disponibiliza totalmente o código fonte para que ele seja analisado [Telegram 2018].

Buscando melhorar a segurança de seu sistema frente a concorrentes, em 2016 o *WhatsApp* incorporou por padrão o protocolo de segurança fim-a-fim Signal [Cohn-Gordon et al. 2017], que possui código aberto [WhatsApp 2016]. Porém, como o *WhatsApp* possui código fechado, não é possível verificar se de fato o protocolo foi devidamente implementado. Uma análise realizada em 2016 mostrou indícios de que a forma como o diretório de chaves públicas dos usuários é gerenciado pode permitir ataques como a cifração de dados com uma chave incorreta [Boelter 2016]. Outra crítica é que, por padrão, o aplicativo não informa os usuários de que as chaves de seus interlocutores foram alteradas, o que dificulta a detecção de eventuais ataques de MitM.

Finalmente, cabe notar que existem aplicativos de comunicação de código aberto que, embora menos populares, têm por premissa a proteção das comunicações dos usuários por meio de criptografia fim-a-fim, como *Cryptocat* [Kobeissi 2011] e *Signal* [Signal 2013]. Porém, mesmo nesses casos a arquitetura adotada ainda é centralizada, o que não elimina pontos únicos de falha lógica. Além disso, embora o código fonte desses aplicativos possa ser verificado para averiguar a inexistência de vulnerabilidades

(acidentais ou propositais), o processo de verificação das chaves públicas ainda requer participação ativa de cada usuário. Mais precisamente, toda vez que um usuário troca sua chave pública (e.g., devido à reinstalação do aplicativo), cada contato que com ele se comunique é informado para que possa verificar a validade da nova chave. Uma abordagem mais efetiva, entretanto, seria permitir que a verificação da nova chave por um ou alguns usuários pudesse ser utilizada como base para a confiança de outros usuários, como ocorre na rede de confiança PGP.

3. Fundamentação teórica

Esta seção apresenta os mecanismos que embasam a arquitetura proposta: DHT e PGP.

3.1. DHT (*Distributed Hash Table*)

Similarmente a uma tabela de hash comum, uma tabela de hash distribuída (DHT) é uma estrutura de dados para armazenamento e busca eficiente de informações organizadas na forma (`C`, `V`), onde `C` é a chave de indexação e `V` é o valor correspondente [Zhang et al. 2013]. Uma característica distintiva das DHTs, entretanto, é que o conteúdo delas é distribuído entre os vários nós que compõem a rede em vez de ficarem em um único nó. Mais precisamente, nós cujos identificadores sejam mais próximos das chaves a serem armazenadas, segundo alguma métrica de proximidade que depende do algoritmo adotado, ficam responsáveis por armazenar os dados correspondentes. Para maior disponibilidade, o armazenamento costuma ser feito com algum grau de redundância (i.e., com replicação dos dados); assim, mesmo que um nó saia da rede, as informações da DHT continuam disponíveis nos nós remanescentes.

Existem diversos algoritmos que podem ser usados para a construção de uma rede DHT, como o Kademlia [Maymounkov and Mazières 2002] e o Chord [Stoica et al. 2003]. Neste trabalho optou-se pelo Kademlia, devido a seu elevado desempenho e escalabilidade [Chávez et al. 2015]. Em particular, a busca por uma chave qualquer no Kademlia envolve a consulta (iterativa) a $O(\lg n)$ nós, onde n é o tamanho total da rede.

3.2. PGP (*Pretty Good Privacy*)

O PGP [Zimmermann 1995] provê mecanismos de cifração e assinatura digital por meio de criptografia de chaves públicas. Uma característica importante do sistema, entretanto, é que a autenticação das chaves dos usuários não é centralizada em uma Autoridade Certificadora totalmente confiável. Em vez disso, ela é feita de modo distribuído, por meio de uma "rede de confiança": os próprios usuários da rede verificam a autenticidade das chaves públicas, assinando os certificados uns dos outros. Cada usuário pode então atribuir um grau de confiança aos seus pares, o que indiretamente define o grau de confiança nas chaves públicas contidas nos certificados assinados por cada usuário.

Um exemplo de rede de confiança é ilustrado na Figura 1. Nessa figura, U_1 verifica as chaves públicas dos usuários U_2 a U_5, assinando os certificados correspondentes. Adicionalmente, ele atribui um nível de confiança a cada um desses usuários. Por exemplo, ao atribuir confiança total ao usuário U_5, todos os certificados assinados por ele (e.g., U_9) são automaticamente confiáveis. Quando o nível de confiança é parcial, por outro lado, são necessárias algumas assinaturas para que o certificado seja considerado válido

(e.g., as assinaturas de U_2 e U_3 sobre o certificado de U_7 permite considerá-lo válido se a confiança em U_2 e U_3 for 50%). Os níveis de confiança são atribuídos por cada usuário, de modo similar a relações de confiança em redes sociais.

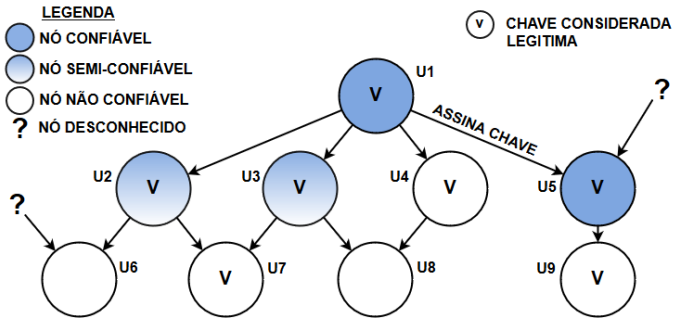

Figura 1. Modelo de Rede de Confiança baseado em PGP.

4. Arquitetura Proposta

A arquitetura do sistema tem duas partes principais: a rede de conexão entre os usuários e a estrutura de segurança criptográfica subjacente. A rede de conexão, baseada em DHT, é responsável por gerenciar as informações referentes que permitem a conexão entre os usuários. Já a camada de criptografia garante a confidencialidade, integridade e autenticidade das mensagens, bem como a identificação dos interlocutores.

4.1. Rede de conexão

Um dos papéis do servidor central em aplicativos existentes é facilitar a conexão entre os usuários. Em uma arquitetura distribuída como a proposta, entretanto, isso é feito por meio de uma rede DHT. Essa rede armazena as informações de conexão dos usuários e, assim, viabiliza o serviço de comunicação P2P.

Mais precisamente, cada usuário deve selecionar um nome de usuário `uname` único na rede. Para cada usuário, são então armazenados dois pares de chave de indexação (`C`) e valor (`V`):

1. `C` = Hash(`uname`): armazena-se em `V` a sua chave pública de domínio K_D. Isso permite a criação de "domínios protegidos", nos quais apenas o dono de K_D pode editar as informações armazenadas no nó indexado por K_D (i.e., para o qual `C` = Hash(K_D)). Isso também garante que apenas um usuário possa ser associado a cada `uname`: caso o nome já exista, a rede recusa a alteração da chave pública K_D correspondente a não ser que esse pedido de alteração seja assinado por K_D em si, de modo similar ao que ocorre no Self-certifying File System (SFS) [Mazières 2000].
2. `C` = Hash(K_D): armazena-se em `V` as informações necessárias para o estabelecimento da comunicação com o usuário e verificação de sua identidade, também na forma de um domínio protegido usando K_D. Especificamente, no protótipo utilizado é armazenado o endereço IP e porta do usuário, bem como a sua chave pública PGP e certificados correspondentes para a proteção das comunicações entre usuários. Isso dá aos usuários suporte a mobilidade, já o dono de K_D (e apenas ele) pode alterar suas informações de conexão a qualquer momento.

Cabe notar que, embora em princípio a chave PGP do usuário possa ser utilizada como K_D, no protótipo construído é feita essa separação por razões de conveniência, dado que K_D e a chave PGP em princípio têm propósitos distintos e, portanto, podem ter configurações distintas (e.g., datas de expiração).

Usando essa infraestrutura, quando um usuário $\mathcal{A}$ deseja comunicar-se com $\mathcal{B}$, é realizada uma busca na DHT construída via Kademlia. Ao se obter as informações de conexão do usuário $\mathcal{B}$, é possível criar uma conexão direta P2P via *socket*, de forma que toda a comunicação siga diretamente entre as partes sem a necessidade de um servidor central para repassar as mensagens. Essa abordagem contorna eventuais problemas de gargalo e protege o serviço contra possíveis ataques de DoS.

Para se comunicar com qualquer usuário da rede DHT, basta conhecer pelo menos um usuário dela. Por exemplo, isso poderia ser feito por meio do convite de um usuário que já utiliza o aplicativo, ou por meio de páginas web. Para facilitar o ingresso de novos usuários, entretanto, o protótipo construído disponibiliza também um servidor *tracker*, um artifício comumente utilizado em redes P2P para esse propósito. Assim, o *tracker* é um nó (ou conjunto de nós) com endereço IP fixo e conhecido inicialmente por todos os nós que entram na rede, pois essa informação é diretamente registrada no aplicativo em si. Embora *trackers* criem algum grau de centralização na rede, sua indisponibilidade apenas atrapalha a entrada de novos nós na rede, sem de fato prevenir tais ingressos ou afetar as comunicações entre usuários previamente registrados.

4.2. Estrutura de segurança

A rede DHT permite a remoção do servidor central, melhorando a disponibilidade e resiliência do serviço. Porém, isso não garante a segurança das comunicações em si, o que é obtido por meio de uma rede de confiança PGP cujas informações (chaves públicas e certificados) são armazenadas na DHT. Essas chaves são então utilizadas para verificar a autenticidade das partes, conforme o grau de confiança definido pelo usuário PGP. Elas também são utilizadas para a execução de um protocolo de acordo de chaves, como o próprio protocolo Signal, para garantir a segurança fim-a-fim das comunicações.

No protótipo implementado, as comunicações são protegidas usando o protocolo OpenPGP conforme definido na RFC 4880 [Callas et al. 2007], aproveitando-se as mesma bibliotecas já disponíveis para criação da rede de confiança PGP. Embora protocolos de proteção como o Signal apresentem características mais avançadas, como *forward secrecy*, a integração dessas características é deixada como trabalho futuro.

5. Protótipo desenvolvido como prova de conceito

Para a construção do protótipo da solução, foram utilizadas duas bibliotecas de código aberto: TomP2P [Bocek 2004] e Spongy Castle [Tyley 2014]. Basicamente, a biblioteca TomP2P é responsável pela criação da rede DHT, enquanto a biblioteca Spongy Castle foi usada para o gerenciamento de certificados dos usuários, assinaturas digitais e cifração das mensagens usando PGP.

A implementação foi realizada no sistema operacional Android, dado que essa plataforma é quase 6 vezes mais difundida do que o segundo maior concorrente, o iOS [Gartner 2018], além de possuir ampla documentação e uma comunidade de desenvolvedores bastante ativa. O aplicativo funciona em Android v6.0 ou superior e foi desen-

volvido no software Android Studio 2.3. Para as aplicações de suporte como o *tracker*, os protótipos e o visualizador de rede, foram utilizados Java e HTML/Javascript/CSS, desenvolvidos usando o ambiente de desenvolvimento Eclipse Mars.

A ferramenta completa, bem como instruções de instalação e o uso do aplicativo, encontram-se disponíveis no seguinte repositório do site GitHub `https://github.com/brunoarakaki/mensageiro-p2p`. A Figura 2 ilustra algumas das telas do aplicativo, que tem como principais funcionalidades:

- O aplicativo permite adicionar contatos por nome de usuário;
- O aplicativo se conecta à rede DHT usando o endereço de um *tracker* ou o endereço já conhecido dos contatos salvos no aplicativo;
- É possível iniciar uma conversa com outro usuário;
- Na conversa, é possível enviar mensagens de texto;
- Um usuário pode visualizar os outros usuários que estão conectados com ele;
- Um usuário pode assinar o certificado de outro usuário;
- Usuários que possuem seu certificado assinado por outros usuários passam automaticamente a serem confiáveis para estes usuários e para os amigos destes usuários (i.e., por simplicidade do protótipo, a rede de confiança é configurada para que haja confiança total nos nós cujos certificados sejam assinados).

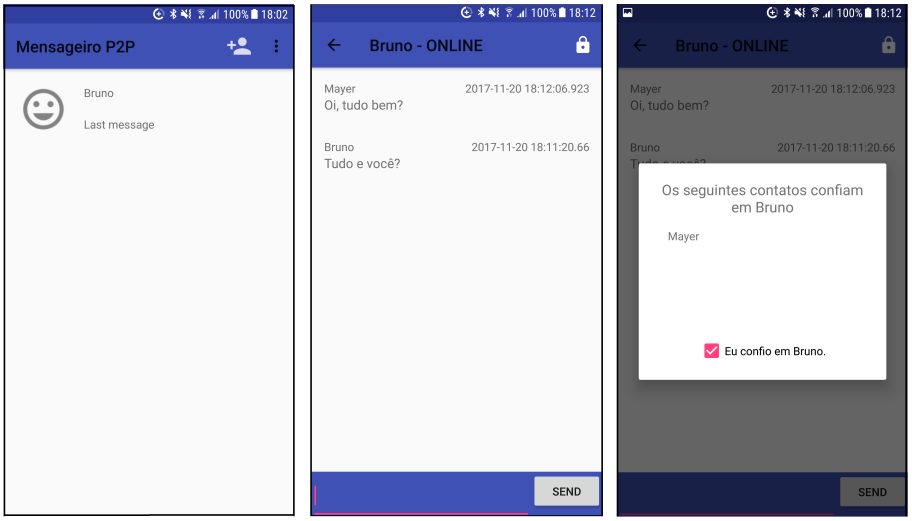

Figura 2. Estados das tela do aplicativo.

5.1. Demonstração

Para mostrar o funcionamento do protótipo, os seguintes requisitos devem ser satisfeitos:

- Todos os dispositivos devem ter conexão direta. Isto é, todos os dispositivos devem estar conectados em uma mesma rede, por exemplo Wi-Fi, ou devem possuir IPs públicos. A razão é que a versão preliminar do aplicativo não tem suporte a redes com NAT (*Network Address Translation*).
- Assim como os dispositivos, o *tracker* também é um nó da rede. Ele deve estar em execução e com um endereço de IP acessível na rede dos dispositivos.

Para a sequência da apresentação, são necessários pelo menos três dispositivos que atendam aos requisitos de sistema descritos na Seção 5. A demonstração proposta segue os seguintes passos:

1. Criar o usuário no aplicativo. Ao fazer isso, é iniciada uma tentativa de conexão automática com o *tracker*.
2. Adicionar um contato pelo nome do usuário.
3. Iniciar uma conversa com o contato adicionado para verificar o funcionamento correto do mecanismo de comunicação.
4. Assinar o certificado de outro usuário e verificar a rede de confiança. Para essa verificação, são necessários pelo menos três dispositivos para simular a situação apresentada na Figura 1.
5. Revogar a assinatura do certificado de outro usuário e verificar novamente a rede de confiança.
6. (Opcional) Capturar pacotes na rede, para verificar que as mensagens estão de fato cifradas.
7. (Opcional) Tentativa de sobrescrever dados na rede DHT, para verificação de que apenas o dono da chave de domínio correspondente pode fazê-lo.

6. Conclusão

Aplicativos móveis para comunicações instantâneas são atualmente bastante populares no Brasil e no mundo. Entretanto, como tais soluções costumam adotar uma estrutura centralizada em uma entidade controladora, a segurança real desses aplicativos depende em última instância da disponibilidade e honestidade desse servidor central.

A solução proposta neste documento tem como objetivo mitigar essas vulnerabilidades de segurança por meio da adoção de uma arquitetura descentralizada. O sistema proposto combina uma DHT para a busca de usuários e PGP para verificação de suas chaves públicas. O resultado é uma solução com elevada resistência a censura, além de permitir que usuários controlem o grau de confiança atribuído às chaves públicas de seus contatos para evitar tentativas de escutas em suas comunicações.

Embora o protótipo desenvolvido como prova de conceito apresente algumas limitações (e.g., ausência de suporte a NAT), ele permite verificar as principais funcionalidades da arquitetura proposta. Em particular, ele permite o estabelecimento de conexões entre usuários móveis e a troca de mensagens com segurança fim-a-fim. Como trabalho futuro, planeja-se expandir esse protótipo para que ele possa ser usado mais facilmente como base para a construção de aplicativos competitivos com os existentes atualmente. Para isso, os próximos passos principais são a inclusão do protocolo Signal para a comunicação entre nós e a incorporação de mecanismos para contornar NATs, como *Hole Punching* [Srisuresh et al. 2008].

Agradecimentos: Este trabalho foi em parte financiado pelo CNPQ (Bolsa de Produtividade 301198/2017-9) e pela FAPESP (projeto 13/25977-7).

Referências

[Abu-Salma et al. 2017] Abu-Salma, R., Sasse, M., Bonneau, J., Danilova, A., Naiakshina, A., and Smith, M. (2017). Obstacles to the Adoption of Secure Communication Tools. In *IEEE Symposium on Security and Privacy (IEEE S&P)*, pages 137–153.

[Barreto and Lima 2016] Barreto, I. and Lima, M. (2016). Marco Civil da Internet: Análise das Decisões Judiciais que Suspenderam o Aplicativo WhatsApp no Brasil – 2015-16. *Rev. de Direito, Governança e Novas Tecnologias*, 2(2).

[Bocek 2004] Bocek, T. (2004). TomP2P: A P2P-based high performance key-value pair storage library. Disponível em: https://tomp2p.net/. Acesso em nov/2017.

[Boelter 2016] Boelter, T. (2016). WhatsApp Retransmission Vulnerability. Disponível em: https://tobi.rocks/2016/04/whats-app-retransmission-vulnerability/. Acesso em: jun/2018.

[Callas et al. 2007] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and Thayer, R. (2007). RFC 4880: OpenPGP Message Format. https://tools.ietf.org/html/rfc4880.

[Chávez et al. 2015] Chávez, A. G. M., Cortés, E. P., and Guerrero, M. L. (2015). A performance comparison of Chord and Kdemlia DHTs in high churn scenarios. *Peer-to-Peer Networking and Applications*, 8(5):807–821.

[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 *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 451–466.

[Gartner 2018] Gartner (2018). Gartner Says Worldwide Sales of Smartphones Recorded First Ever Decline During the Fourth Quarter of 2017. Disponível em: https://www.gartner.com/newsroom/id/3859963. Acesso em: jul/2018.

[Kobeissi 2011] Kobeissi, N. (2011). Cryptocat. https://crypto.cat/.

[Maymounkov and Mazières 2002] Maymounkov, P. and Mazières, D. (2002). Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *1st Int. Workshop on Peer-to-Peer Systems (IPTPS'01)*, pages 53–65. Springer.

[Mazières 2000] Mazières, D. (2000). *Self-certifying File System*. PhD thesis, MIT.

[Schrittwieser et al. 2012] Schrittwieser, S., Kieseberg, P., Leithner, M., Mulazzani, M., and Huber, M. (2012). Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications. In *NDSS*. Internet Society.

[Signal 2013] Signal (2013). Signal. https://signal.org/.

[Srisuresh et al. 2008] Srisuresh, P., Ford, B., and Kegel, D. (2008). RFC 5128 - State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs). Technical report, IETF. http://www.rfc-editor.org/rfc/rfc5128.txt.

[Stoica et al. 2003] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32.

[Telegram 2018] Telegram (2018). Telegram FAQ. https://telegram.org/faq.

[Tyley 2014] Tyley, R. (2014). Spongy castle - a repackage of Bouncy Castle for Android. Disponível em: https://rtyley.github.io/spongycastle/. Acesso em nov/2017.

[WhatsApp 2016] WhatsApp (2016). WhatsApp Encryption Overview. Disponível em: https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf. Acesso em: mai/2018.

[Zhang et al. 2013] Zhang, H., Wen, Y., Xie, H., and Yu, N. (2013). A Survey on Distributed Hash Table (DHT): Theory, Platforms, and Applications.

[Zimmermann 1995] Zimmermann, P. (1995). Building in Big Brother. chapter Pretty Good Privacy: Public Key Encryption for the Masses, pages 93–107. Springer.