Autores: **Miguel Martins Mota; Pedro Miguel Nicolau Escaleira; Rafael dos Santos Simões**

Date: **21/04/2020**

Conteúdo

1. PROTOCOLO 2
1.1 PRIMITIVAS 2
1.2 KDF CHAINS 3
1.3 SYMMETRIC-KEY RATCHET 5
1.4 DIFFIE-HELLMAN RATCHET 5
1.5 DOUBLE RATCHET 12
1.5.1 PARA QUE SERVE DOUBLE RATCHET? 12
1.5.2 COMO FUNCIONA A DOUBLE RATCHET? 12
1.6 SERÁ O PROTOCOLO DO SIGNAL REALMENTE SEGURO? 16
1.7 O QUE TORNA O PROTOCOLO DO SIGNAL SUPERIOR? 17
1.8 DOMAIN FRONTING E SIGNAL APP 17

2. PROBLEMAS SOCIAIS, ECONÓMICOS E ÉTICOS 20
2.1 PROBLEMAS SOCIAIS 20
2.1.1 UTILIZAÇÃO DO SERVIÇO PARA ATIVIDADES ILEGAIS 20
2.1.2 UTILIZAÇÃO DE DOMAIN FRONTING 20

3. REFERÊNCIAS 22

4. APÊNDICE 23
4.1 IMPORTANT NOTES 23

1 Protocolo

Toda a informação presente foi obtida da documentação do Signal [**?**].

1.1 Primitivas

Sem contar com os algoritmos referidos em baixo que são uma grande parte do que torna Signal seguro este tambem usa as primitivas ,atualmente, mais seguras.

SHA-512 o que implica que existam , se o algoritmo *random* for perfeito:

$$Combinacoes = 2^{bits} = 2^{512} = 1.340781^{154} \tag{1}$$

Isto implica que se alguém com um computador imaginemos a fazer 2^16 cálculos em 0.2s seria 327680 por segundo:

$$Tempo = 1.340781^{154}/327680 = 4.091739^{148}s = 4.735809^{143}dias = 1.297482^{141}anos \tag{2}$$

Colocando isto em perspectiva a Terra tem $4,543 \times 10^9$ o que coloca a solução (para 1 computador) a demorar:

$$Tempo = 1.297482^{141}/4,543 * 10^9 = 1.5740434^{18} = 1500 \tag{3}$$

O que significa que seriam necessários 1500 vezes o tempo que a terra existe para poder decifrar uma *hash* efectuada com ***SHA-512***

Curve25519 é uma curva elíptica que garante 128 bits de segurança com *elliptic curve Diffie-hellman key agreement*. Este foi construído de forma a evitar potenciais *pitfalls* e para ser imune a *time attacks* (ataques com base na analise do tempo gasto para executar algoritmos criptográficos)

VXEdDSA é um algoritmo de *signing* que usa a hash *SHA-5125*. Este funciona da seguinte forma,sendo **k** uma chave privada Montgomery,**M** uma mensagem a ser assinada,**Z** uma sequência de 64 bytes aleatórios e seguros,**u** a chave publica Montgomery,**V||h||s** a assinatura a verificar(sequência de bytes em que **V** faz *encode* de um ponto e **h e s** fazem *encode* do *integer modulo q(prime number)*:

```
vxeddsa_sign(k, M, Z):
    A, a = calculate_key_pair(k)
    Bv = hash_to_point(A || M)
    V = aBv
    r = hash3(a || V || Z) (mod q)
    R = rB
    Rv = rBv
    h = hash4(A || V || R || Rv || M) (mod q)
    s = r + ha (mod q)
    v = hash5(cV) (mod 2b)
    return (V || h || s), v
```

Listing 1: Assinatura de um documento

```
vxeddsa_verify(u, M, (V || h || s)):
    if u >= p or V.y >= 2|p| or h >= 2|q| or s >= 2|q|:
        return false
    A = convert_mont(u)
    Bv = hash_to_point(A || M)
    if not on_curve(A) or not on_curve(V):
        return false
    if cA == I or cV == I or Bv == I:
        return false
    R = sB - hA
    Rv = sBv - hV
    hcheck = hash4(A || V || R || Rv || M) (mod q)
    if bytes_equal(h, hcheck):
        v = hash5(cV) (mod 2b)
        return v
    return false
```

Listing 2: Validação de assinatura de um documento

Entre outras primitivas com objectivos idênticos.

1.2 KDF chains

KDF Chains faz parte do *core* de um dos principais métodos, *Double Ratchet* (secção 1.5), necessários para manter o **Signal** seguro. Este algoritmo recebe um **segredo** e uma **chave aleatória KDF** e dados de *input* e retorna um *output* indistinguível de um *random*, isto se a chave não for conhecida. O termo *KDF Chain* é usado quando parte do *output* de uma *KDF* é usado para substituir a **chave aleatória KDF**, podendo assim esta ser usada para outro *input*. O diagrama 1 representa o processo de 3 *inputs* produzindo 3 novas *output keys*.

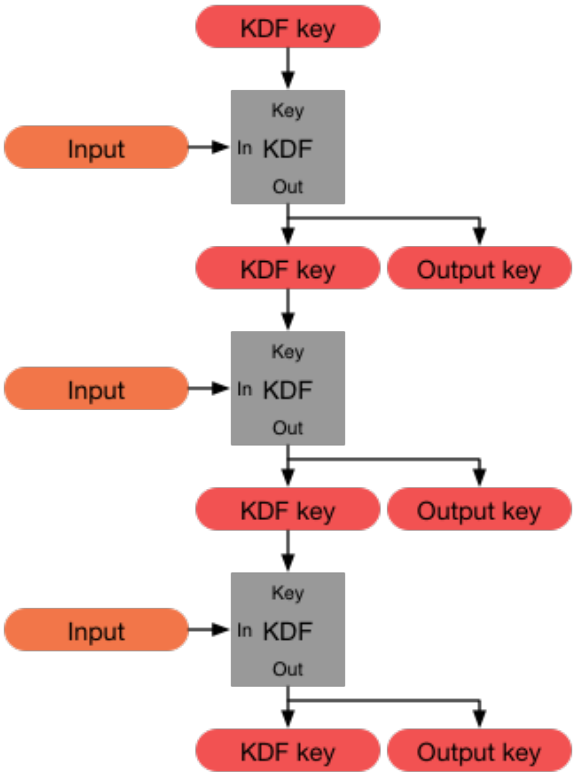

Figura 1: KDF Chain com 3 inputs

Uma *KDF Chain* apresenta as seguintes propriedades:

- **Resiliência:** Se não houver conhecimento da chave KDF, as chaves de output aparentam ser *random*. Isto mantem-se verdadeiro mesmo que o atacante controle parte dos *inputs*
- **Forward Security:** Chaves de *output* do passado aparentam ser *random* para um atacante que conheceu uma chave KDF num momento futuro.
- **Break-in recovery:** Se os inputs futuros adicionarem entropia suficiente , as chaves *output* do futuro aparentam ser *random* para um atacante que conheceu uma chave KDF num momento qualquer

Numa sessão que utilize **Double Ratchet** (secção 1.5), imaginemos entre a Alice e o Bob, cada utilizador guarda 3 chaves uma para cada KDF chain: **root chain**, **sending chain**,**receiving chain**. A chave *sending* da Alice é equivalente à chave *receiving* do Bob.

Por cada mensagem enviada e/ou recebida a *chain* "avança"e as suas chaves de *output* São usadas para encriptar e desencriptar as mensagens a este processo dá-se o nome de **symmetric-key ratchet**.

1.3 Symmetric-key ratchet

Todas e qualquer mensagem enviada ou recebida é encriptada usando uma **chave de mensagem única**. Esta chave única corresponde a uma chave de *output* das *KDF Chains* (secção 1.2 correspondentes. Chamemos a estas chaves *chain keys*.

Os *inputs* na KDF Chain para envio e recepção são constantes, por esta razão não fornecem *break-in recovery*. As suas chains apenas garantem que cada mensagem é encriptada com uma chave única que pode ser eliminada após encriptação ou desencriptação.Calcular a próxima chave da *chain* e de mensagem corresponde a um simples *ratchet step* no algoritmo *symmetric-key ratchet*. O diagrama em baixo (diagrama 2 demonstra a execução deste algoritmo efectuando 2 *ratchet steps*.

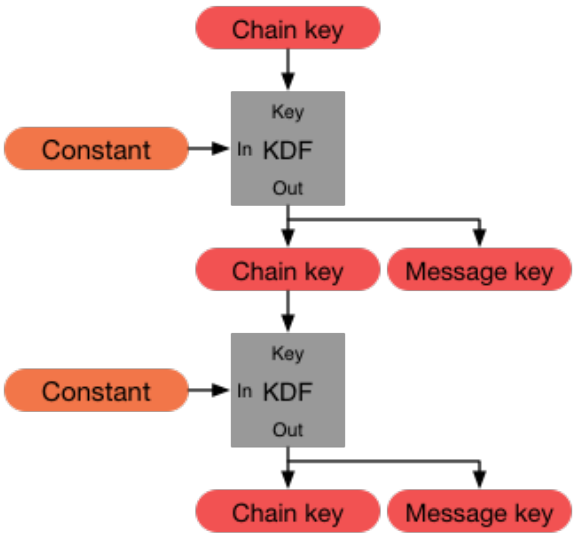

Figura 2: Symmetric-key ratchet with 2 steps

1.4 Diffie-Hellman ratchet

Até este momento mencionamos *KDF chains e Symmetric-key ratchet* ,sendo estes algoritmos necessários para o funcionamento do *Double Ratchet*, no entanto estes por si só não garantem que um atacante possuindo *receiving chain keys* ou *sending chain keys* não consiga gerar as futuras chaves e desencriptar todas as mensagens futuras. Para evitar isto o **Signal** combina *symmetric-key ratchet* com *DH Ratchet*.

Para implementar um Diffie-Hellman ratchet, cada usuário gera um par de chaves Diffie-Hellman (***ratchet key pair***). Todas as mensagens trocadas entre estes 2 usuários vão ter no header a **chave publica**. Quando o receptor recebe a chave publica do emissor é efectuado um **DH ratchet step** que consiste em substituir o actual ***ratchet key pair*** por um novo par.

Este processo leva a que ambos usuários estejam constantemente a trocar as ***ratchet key pair***, desta forma se um dos usuários for comprometido, isto é, se um atacante obter a chave privada da *ratchet* apenas fica comprometida uma mensagem e todas as mensagens anteriores e posteriores continuam seguras.

Os diagramas que se seguem,juntamente com o texto que os acompanham, demonstram mais especificamente o processo do **DH Ratchet**.

A Alice é 'inicializada' com a *ratchet key* publica do Bob. Desta forma a chave publica da Alice ainda não é conhecida pelo Bob. A Alice executa um calculo *Diffie-Helman* entre a sua *ratchet key* private e a *ratchet key* publica do Bob.

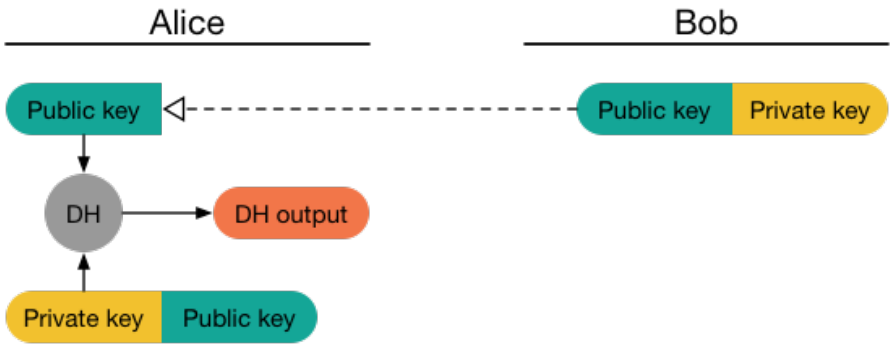

Figura 3: Mensagem enviada do Bob para a Alice

A primeira mensagem enviada pela Alice anuncia a sua *ratchet key* publica. Quando o Bob recebe uma destas mensagens ele calcula um *DH output* entre a *ratchet key* publica da Alice e a sua *ratchet key* privada, o que garante (através das propriedades do *DH*), caso não tenha havido um *middle-man* ,que seja igual ao *output* inicial da Alice. O Bob ,posteriormente, substitui o seu par de *ratchet keys* e calcula um novo *DH output*.

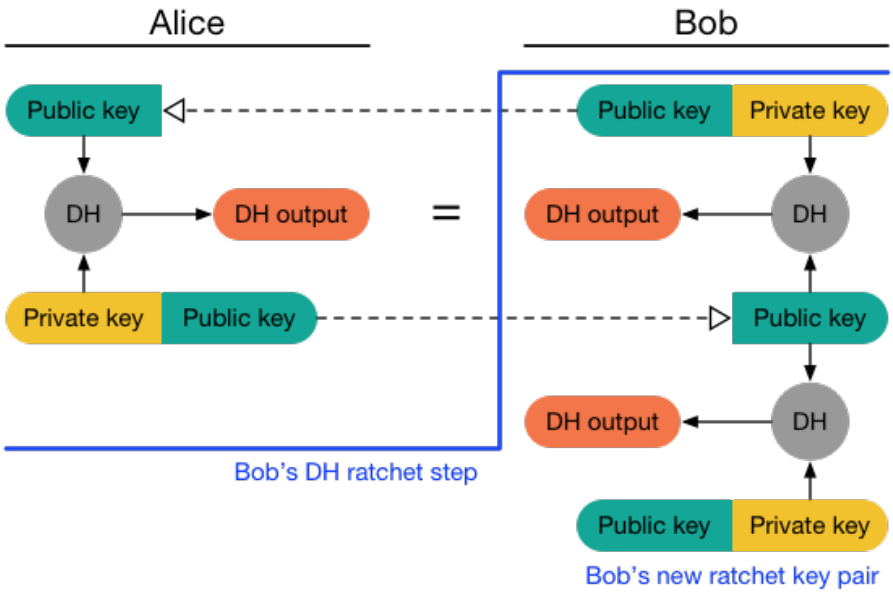

Figura 4: Mensagem enviada da Alice para o Bob após a mensagem inicial

Uma nova mensagem vinda do Bob anuncia a sua nova *ratchet key* publica. Quando a Alice receber esta mensagem vai executar um passo de *DH ratchet*, substituindo a o seu par de *ratchet keys* e derivando 2 novos *DH outputs*, um que será equivalente ao ultimo *DH output* do Bob e um novo para ser usado na próxima mensagem.

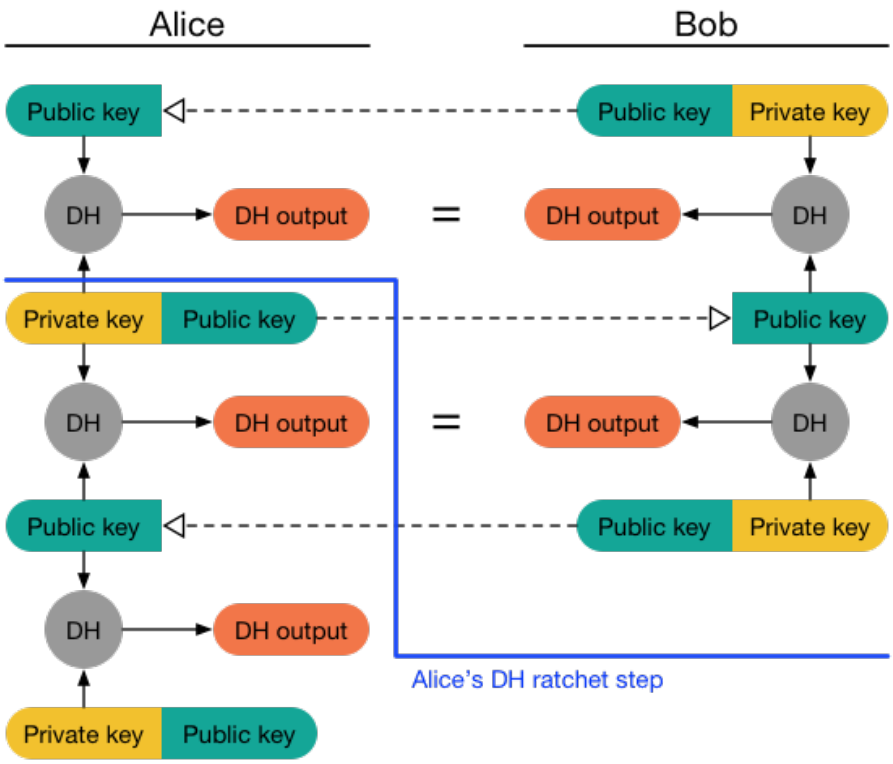

Figura 5: Nova mensagem do Bob com a Alice como destinatário

Uma mensagem nova por parte da Alice anuncia a sua nova chave publica. Eventualmente o Bob ira receber uma destas mensagens e executar um passo de *DH ratchet* e assim sucessivamente.

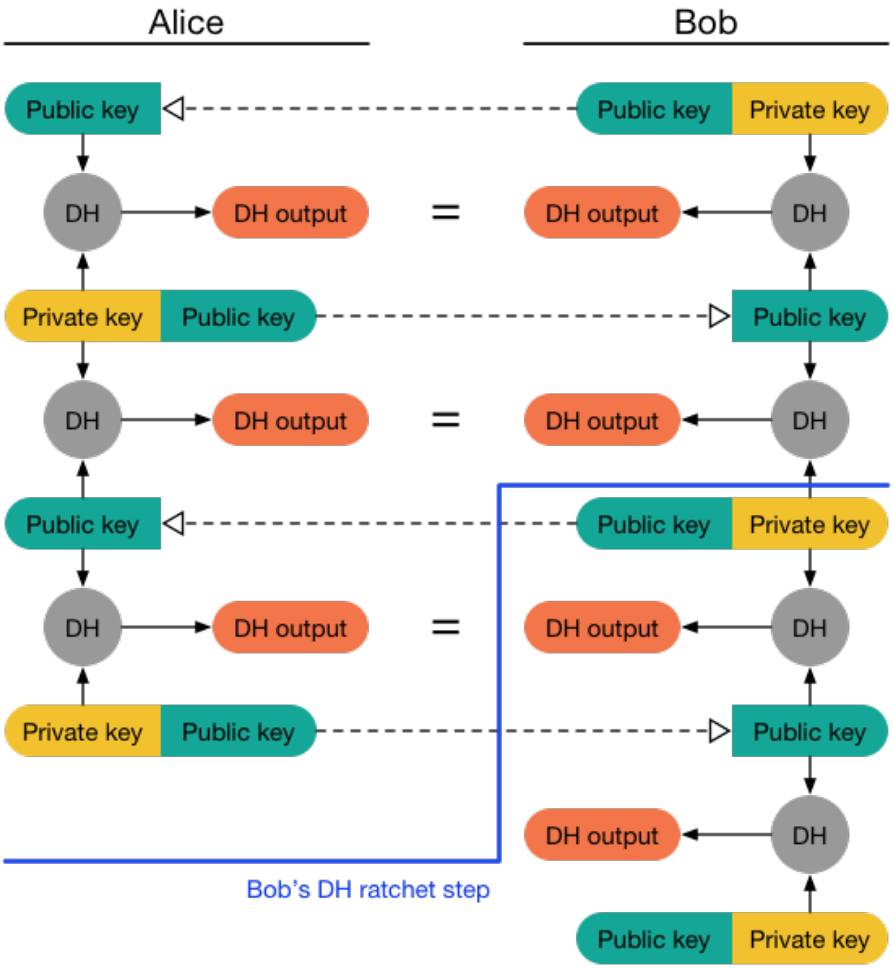

Figura 6: Nova mensagem por parte da Alice

Os *DH output* que são gerados em cada passo de *DH ratchet* São usados para derivar novas chaves de envio e recepção. A imagem em baixo revisita o primeiro passo do *DH ratchet* efectuado pelo Bob. Nesta imagem o Bob usa o seu primeiro *DH output* para derivar a sua chave de recepção que será equivalente a chave de envio por parte da Alice.

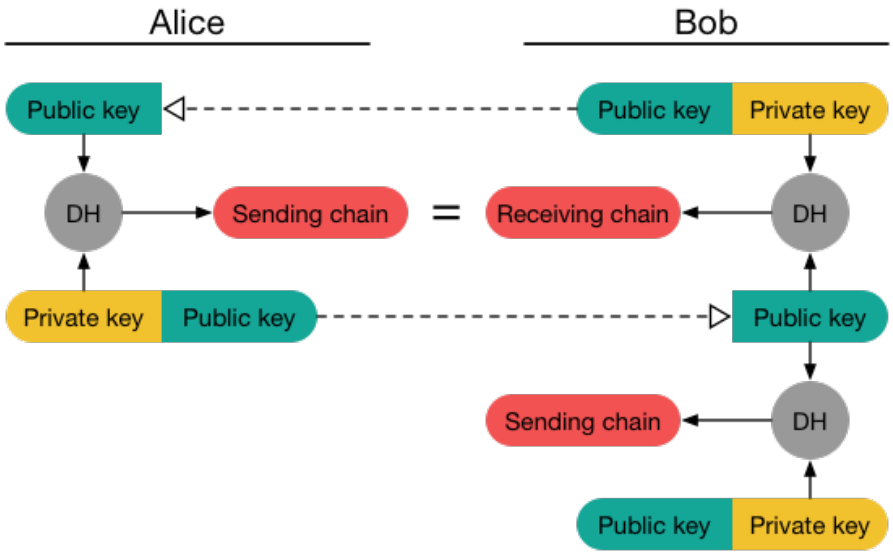

Figura 7: Mensagem inicial por parte do Bob e resposta por parte da Alice , mostrando mais especificamente as cadeias de recepção e de envio

Sendo que os usuários efectuam *"turnos"* a executar um passo do *DH Ratchet* , cada um toma turnos a introduzir novas cadeias de envio:

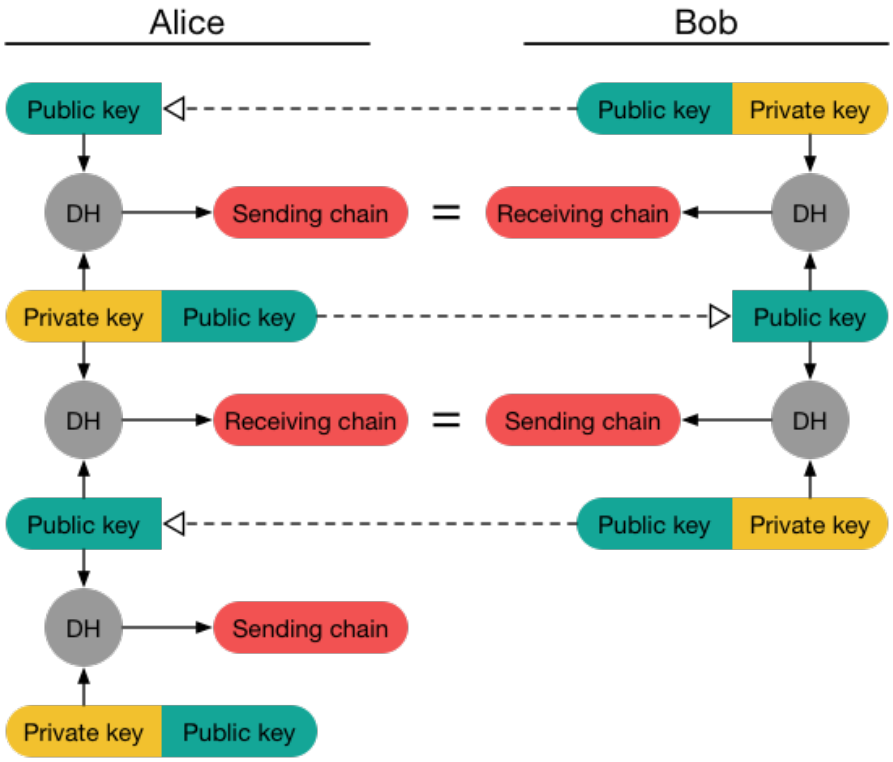

Figura 8: Nova mensagem por parte do Bob, mostrando as cadeias de receção e de envio

No entanto a imagem em cima (fig. 8), é uma simplificado. Em vez de se usar directamente os *DH outputs*, estes são usados como *KDF (1.2) inputs* para uma *root chain*, sendo os *KDF ouputs* da *root chain* usados como chaves de envio e recepção. Usar *KDF* aumenta a resiliência e *break-in recovery*.

Desta forma ,um passo completo de uma *DH Ratchet* consiste em atualizar a *root KDF chain* $2\times$ e usar as *DH output keys* como as novas chaves de envio e recepção.

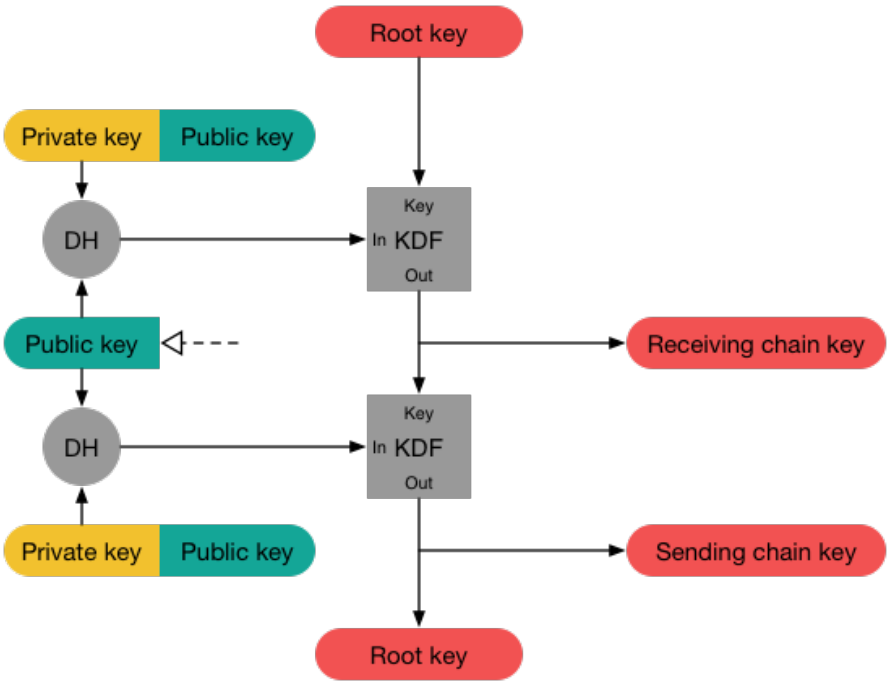

Figura 9: Imagem que demonstra o processo por de trás da geração das cadeias mencionadas nas imagens 7 e 8

1.5 Double Ratchet

1.5.1 Para que serve Double Ratchet?

O algoritmo *Double Ratchet* é um algoritmo usado por dois usuários (aplicação móvel) na troca de mensagens encriptadas baseadas numa chave secreta partilhada entre ambos. Ambos usuários derivam uma nova chave por cada mensagem de forma a que chaves anteriores não possam ser calculadas através da chaves mais recentes. Os usuários enviam valores públicos de *Diffie-Helman* juntamente com as suas mensagens. Os resultados dos cálculos de *Diffie-Helman* São misturados com a chave derivada anteriormente de modo a que chaves anteriores não possam calcular chaves mais recentes. Estas propriedades garantem protecção perante das mensagens encriptadas caso exista o comprometimento das chaves de um dos usuários.

1.5.2 Como funciona a Double Ratchet?

A algoritmo de ***Double Ratchet*** é a combinação de *Symmetric-Key (sec.1.3)* com *DH Ratchets (sec. 1.4)* :

- Quando uma mensagem é enviada ou recebida, uma passo de *Symmetric-Key ratchet* é aplicado nas cadeias de envio ou recepção para derivar a nossa chave da mensagem.
- Quando uma nova chave *ratchet public* é recebida, um passo de *DH ratchet* é efetuado antes da *symmetric-key ratchet* para substituir as chaves da cadeia (*chain keys*)

No diagrama 10 ,em baixo, a Alice inicializou o processo com a chave *ratchet public* e um segredo partilhado

(equivalente à chave raiz inicial). Na inicialização a Alice gera um novo par de chaves *ratchet* e envia o *DH output* para a raiz do *KDF* de forma a que sejam calculadas uma nova chave raiz e uma nova chave de envio.

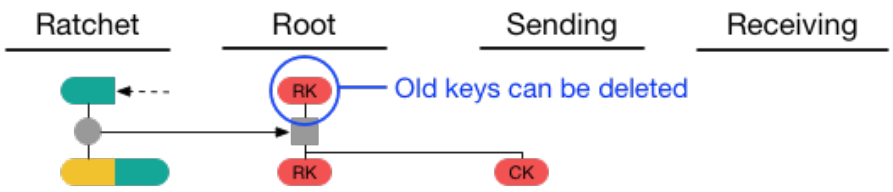

Figura 10: Primeira mensagem por parte do Bob e inicializacao da chave raiz e chave de envio

Quando a Alice envia a sua primeira mensagem, ela aplica um passo *symmetric-key ratchet* à sua cadeia de chaves de envio, resultando numa nova chave para mensagens (cada chave deste género possuirá um *label* perante a mensagem que estas encriptam/ desencriptam). A nova chave da cadeia é guardada enquanto a chave para mensagem e a antiga chave de cadeia podem ser eliminadas.

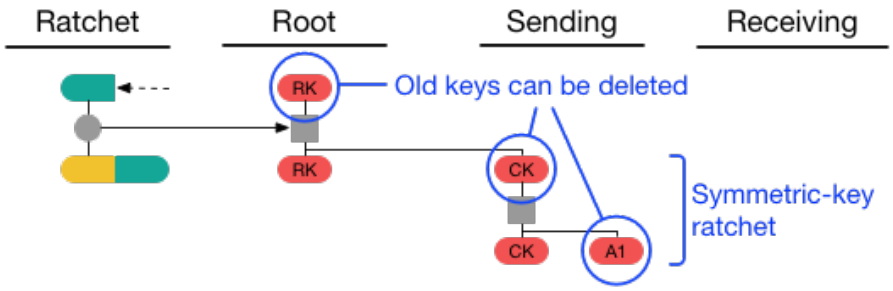

Figura 11: Demonstração de um passo na symmetric-key ratchet após receção da primeira mensagem

Se posteriormente a Alice receber uma mensagem vinda do Bob, esta irá conter uma nova chave *rachet public*. A Alice aplica um passo de *DH ratchet* de forma a derivar novas chaves de envio e recepção. Posteriormente aplica um passo *symmetric-key ratchet* na cadeia de recepção para obter a chave de recepção para a mensagem vinda do Bob.

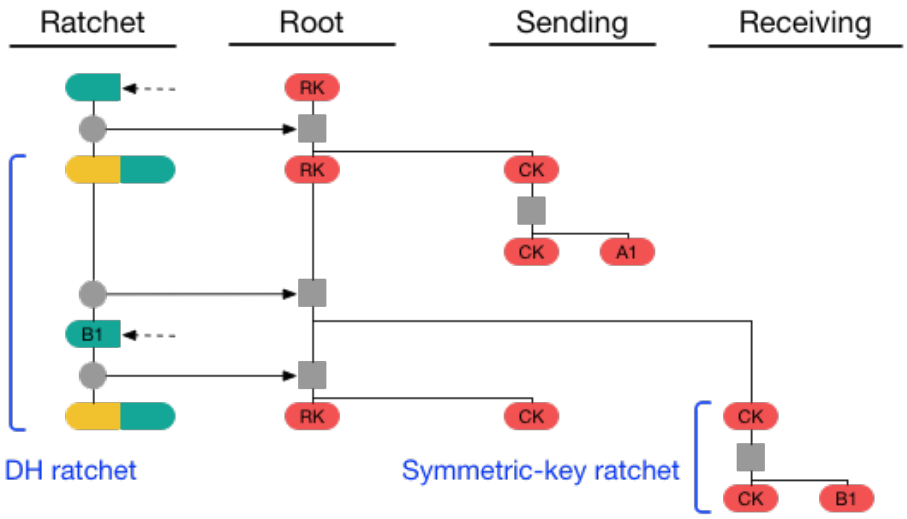

Figura 12: Demonstração de um passo na symmetric-key ratchet após receção da segunda mensagem vinda do Bob levando à geração de uma chave de receção

Suponhamos que a Alice envia uma mensagem A2, recebe uma mensagem B2 (com a chave *rachet public* antiga), depois envia a mensagem A3 e A4. A cadeia de envio da Alice faz 3 passos enquanto a sua cadeia de recepção apenas andou 1 passo.

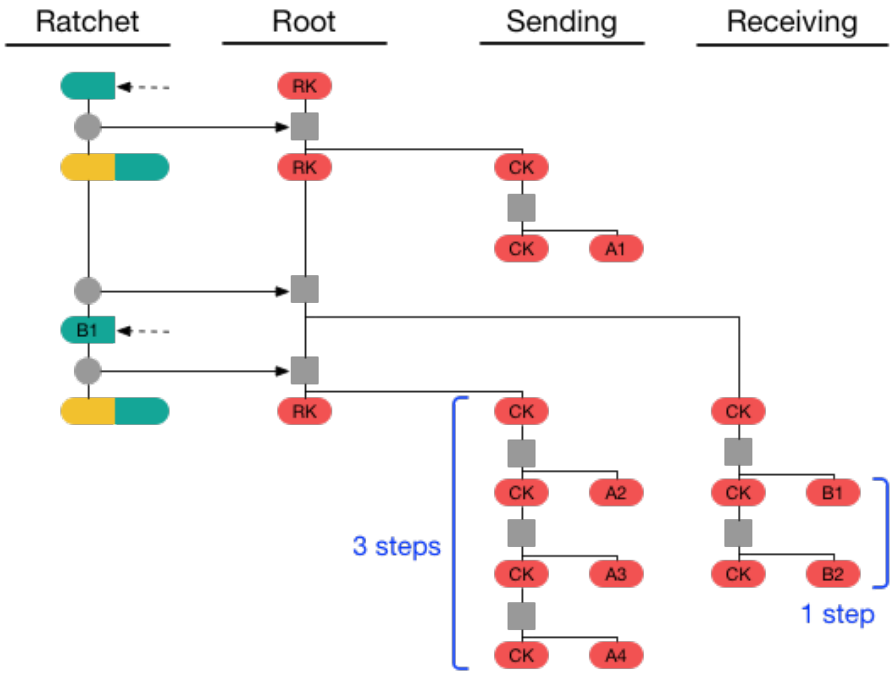

Figura 13: Demonstração de alguns casos específicos do Double-Ratchet

Se ,posteriormente, receber a mensagem B3 e B4 com a próxima chave *rachet* do Bob e em seguida enviar a mensagem A5 o seu estado final será como identificado no diagrama 14.

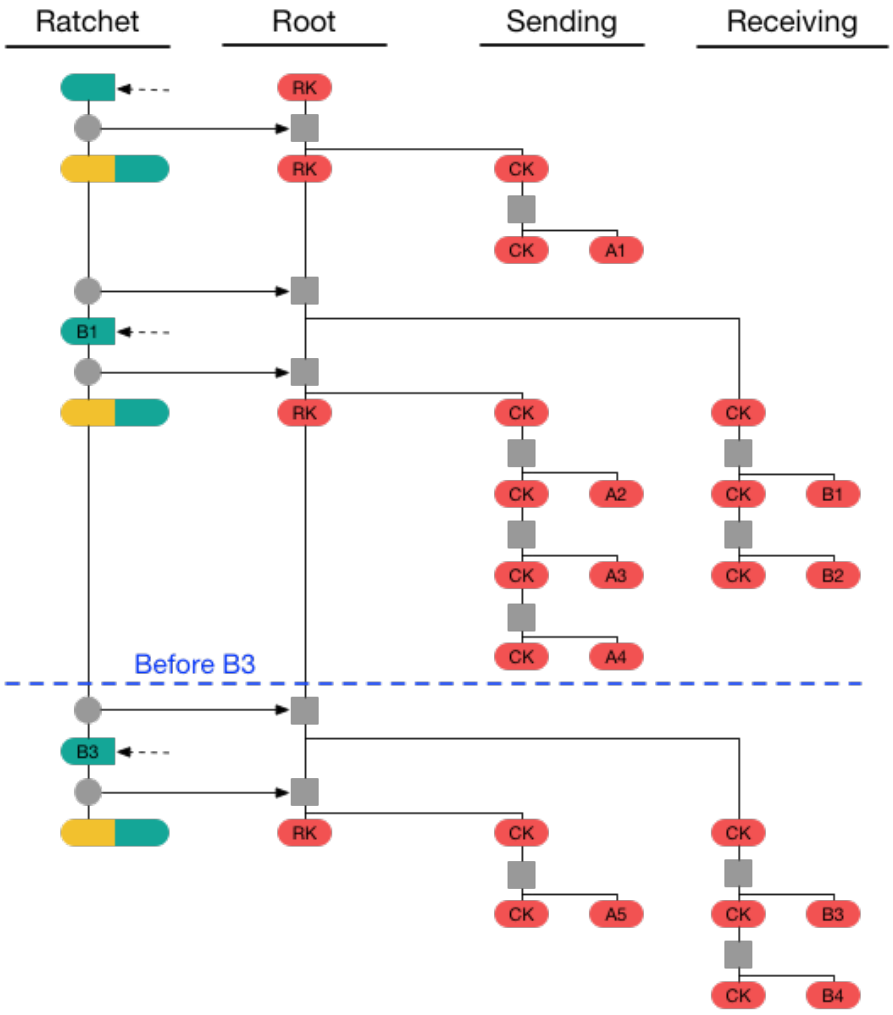

Figura 14: Estado final da cadeia

1.6 Será o protocolo do Signal realmente seguro?

Perante tudo o que foi mencionado nos protocolos usados pelo Signal, rapidamente poderíamos considerar que sim, o Signal seria um protocolo bastante seguro. No entanto, sem qualquer acesso ao *source-code*, não conseguimos validar realmente o quão solido e seguro é o Signal. Aqui entra uma das grandes razões para o ***Signal*** ser o protocolo e aplicação mais segura de todas as existentes (sendo até usado por entidades politicas de grandes países [**?**]), pois o Signal é ***Open-Source***, permitindo que qualquer pessoa veja o código e detecte erros no mesmo, o que até aos dias de hoje ainda não aconteceu. O protocolo utilizado pelo Signal foi tão popular,seguro e tão bem recebido que ficou a ser um protocolo por si mesmo tendo sido este adaptado em 2018 por grandes empresas como **WhatsApp**,**Facebook Messenger**,**Skype** e **Google Allo**, apesar disso em alguns destes o protocolo Signal não se encontra activado por defeito.

Excluindo o que foi mencionado em cima, o Signal ainda proporciona uma excelente documentação [**?**] onde explica detalhadamente todo o processo por detrás do seu protocolo o que permite que qualquer engenheiro valide o protocolo sem olhar para o código , podendo detectar lacunas antes de olhar para o *source-code*. Por outro lado o Signal tem passado por varias auditorias sem qualquer registo de falhas a nível de segurança.

Algumas auditorias e estudos feitos ao Signal:

- 2014,Outubro, investigadores de *Ruhr University Bochum* publicam uma análise do protocolo **validando a sua segurança** [].
- 2016,Outubro, investigadores de *University of Oxford,Queensland University of Technology and McMaster University* publicam uma analise formal do protocolo concluido que o protocolo era **criptograficamente seguro** [].
- 2017,Julho, investigadores de *Ruhr University Bochum* durante uma nova análise detectam uma falha puramente teórica, uma vez que esta ao acontecer seria automaticamente detectada, voltando a **validar a declaração anterior** [].

1.7 O que torna o protocolo do Signal superior?

Tendo sido adaptado pelas grandes empresas, como mencionado em cima, não se consideraria um protocolo pouco robusto mas ,para realçar uma das suas grandes vantagens é o facto de ser ***Open-Source*** o que permite que uma grande comunidade de pessoas com gosto por ***infosec*** possam ver o *source-code* e detectar falhas sem ser por tentativa e erro através de uma interface (como mencionado anteriormente). Enquanto o Signal é *Open-Source* e sabemos que a comunicação é realmente ***end-to-end encrypted*** , e se duvidarmos podemos facilmente ver o *source-code* e confirmar.

As outras ferramentas estão fechadas ao publico e não permitem acesso ao código desta forma, apenas podemos confiar que as entidades se estão a comportar e não estão a fazer nada de errado com os nossos dados. Em cima disto como falado em cima o Signal tem sofrido bastantes auditorias e estas não encontraram qualquer problema com a aplicação.

Ao contrario do *WhatApp*, o Signal não guarda dados (ou metadados que por vezes ainda são mais importantes) do cliente e se este quiser privacidade o Signal guarante-a enquanto o *WhatsApp* ,sendo a aplicação mais utilizada , não o faz o que põe em causa a privacidade de milhões de usuários. O *WhatsApp* apesar de usar o protocolo seguro e open-source do Signal pode ,*behind the curtains*, estar a fazer algo mais que desconhecemos.O *WhatsApp* efetua um *load* de todos os nossos contactos do telemóvel para os seus servers remotos de forma a detectar quem tem ou não WhatsApp enquanto o Signal usa a tecnologia ***SGX*** da Intel (regiões protegidas da memória).

Comparativamente ao *Telegram* ,para além da distinção que o *Telegram* não é *open-source* e o Signal é , o Signal perante as analises feitas possui um algoritmo de segurança muito superior ao do Telegram. O Signal apaga **completamente**,ênfase no completamente, as mensagens passado algum tempo, enquanto noutras apps, *Telegram* inclusive, os records mantém-se guardados num servidor caso sejam apagadas.

Além de tudo o referido anteriormente o Signal encontra-se activamente a procurar manter a privacidade dos seus clientes mesmo que sejam impostas regras pelos países onde estes vivem.

1.8 Domain Fronting e Signal App

Em grande parte a sociedade procura ter privacidade ,no entanto, no outro lado existem organizações ,muitas vezes politicas, que pretendem obter mais informação sobre a população ou dos cidadãos do seu pais. Desta

forma , alguns paises como por exemplo o Egito (e.g Egito bloqueia todas as comunicações Signal [**?**]), bloqueiam o tráfego de aplicações como o Signal , consequentemente reduzindo a privacidade das pessoas.

O Signal entrevem neste aspecto , mais uma vez procurando devolver a privacidade aos cidadãos, usando um mecanismo chamado ***Domain Fronting***. Este mecanismo ,como referido no artigo [**?**], funciona fazendo *bypass* aos métodos de censura que bloqueiam através de *DPI,DNS Filtering e IP Blocking* , sendo que estes dependem de grandes *CDN's*. Desta forma o *Domain Fronting* não passa de uma maneira de fazer o tráfico parecer que é gerado por um *domain* válido, isto apenas é possível pela forma como as *CDN's* modernas funcionam, tudo isto acontece na *application layer da camada OSI (figura 15)*. Por exemplo, o **Domain A** e o **Domain B** pertencem ao mesmo *CDN*, no entanto o **A** encontra-se bloqueado mas o **B** não. Assim a ideia principal do *Domain Fronting* é criar um pacote com o **Domain B** no *SNI Header* e o **Domain A** no *HTTP Header*. Uma vez que o *SNI* não é encriptado no *TLS Protocol* este não sera bloqueado pelas autoridades, mas quando o pedido chega ao CDN e lê o HTTP Host Header com o *Domain A* este será encaminha para o *Domain A*(fig. 16).

Como podemos ver o Signal tenta sempre possibilitar e manter a privacidade dos seus usuários ,no entanto este método deixa de funcionar se ,como mencionado no artigo do Signal [**?**],se bloquearem todos os sites de um *CDN* especifico para se bloquear o site que se pretende censurar (e.g Tentativa da Russia a bloquear o *Telegram* bloqueiam todo o tráfego de uma *CDN* [**?**] [**?**]). No final de contas o Signal perdeu ,nos paises em que foi censurado, pois as *CDN's* bloquearam este tipo de comportamento (e.g Amazon Services bloqueia Signal [**?**]), mas deixou uma marca que demonstra que ,internamente,o seu protocolo visa e procura manter a privacidade das pessoas.

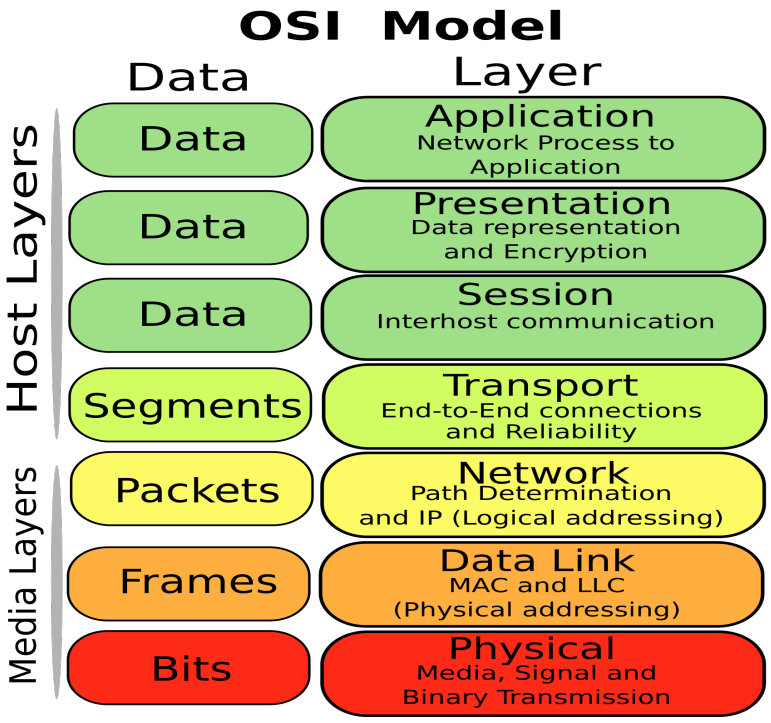

Figura 15: Figura representativa das camadas OSI, dando enfase à camada aplicacional

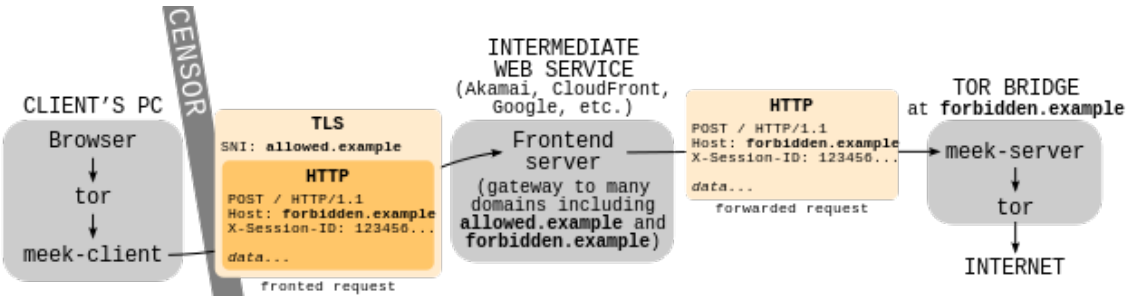

Figura 16: Funcionamento do Domain Fronting

2 Problemas Sociais, Económicos e Éticos

2.1 Problemas Sociais

Antes de poder haver qualquer discussão dos impactos sociais que uma aplicação deste género apresenta, é necessário perceber qual o publico alvo desta. Neste caso, o publico alvo do *Signal* é qualquer pessoa que pretende realizar comunicações seguras e secretas *online*. Tendo isso em consideração, temos pessoas como "maníacos"da cibersegurança, politicos, terroristas, foras da lei e qualquer um que pretenda fazer algo de forma mais escondida dos olhares mais atentos.

2.1.1 Utilização do serviço para atividades ilegais

Um dos problemas sociais derivados da utilização do *Signal* que ocorre logo é a sua utilização para atividades ilegais. Recente à data de criação deste documento, é levado ao congresso Americano a proposta duma lei, *Eliminating Abusive and Rampant Neglect of Interactive Technologies* ou *EARN IT*, que prevê que qualquer *website* combata ativamente abusos a menores feitos pelos seus utilizadores usando as *features* do mesmo [1]. Nela, é prevista a criação de "boas práticas", que minimizem ou ponham um termo no abuso de menores na *internet*, por entidades superiores, sendo que serviços *online* que prestem serviço nos Estados Unidos são obrigados a seguirem esse conjuntos de práticas. Apesar desta lei não indicar especificamente a não utilização de mecanismos de segurança que permitam utilizadores comunicar anonimamente, torna-se bastante claro que, se a ela for aceite, irão certamente haver "boas práticas"que irão passar pela existência de *backdoors* em serviços que usem encriptação para proteger as comunicações, tal como referido por Riana Pfefferkorn, *Associate Director of Surveillance and Cybersecurity* do Centro para Internet e Sociedade de Stanford, no *blog post* [2].

Claramente o serviço apresentado neste relatório seria um dos alvos mais óbvios das consequências desta lei, já que ao possuir *E2E*, permite aos abusadores de menores ter um meio facilitado para praticarem este tipo de atos sem nenhuma autoridade desconfiar, nem mesmo a empresa por detrás do *Signal*. Apesar deste crime ser um problema social sério, em que abusadores devem ser localizados e devidamente punidos, é necessário pensar noutros possíveis problemas que este tipo de censuras criam. Um deles é obviamente o perigo que uma lei desta dimensão causa na segurança das comunicações *online*,na liberdade de expressão. A primeira, porque existindo um *backdoor*, é uma falha extrema de segurança e não se pode considerar uma comunicação segura uma comunicação onde seja minimamente possível por alguém obter o seu conteúdo e a segunda porque é mais um passo para o estado saber e controlar a forma como os cidadão do seu país interagem *online*, algo que é obviamente muito perigoso e que dá mais poder ao estado do que aquele que ele necessita.

Para além disso, é necessário entender que o impedimento de encriptação neste tipo de redes sociais e a existência de *backdoors* acessíveis às entidades judiciais só torna o trabalho destes criminosos mais difícil, mas não impossível, já que estes podem usar outros métodos menos convencionais para fazer as suas comunicações sem qualquer interferência das autoridades. Ou seja, este tipo de medidas só afeta a segurança das pessoas que comunicam na *internet* e não afetaria de forma severa estes delinquentes [3].

A conclusão de tudo isto é que o *Signal* ameaça deixar de prestar funções em solo Americano se a lei for passada, pelo que, como concluído no parágrafo anterior, só prejudica a segurança dos internautas.

2.1.2 Utilização de domain fronting

Em países, como o Irão ou Egito, onde é feita censura do que os seus cidadãos podem pesquisar na *internet*, serviços como o *Signal* ou o *WhatsApp* são usualmente bloqueados por uma *Firewall* do governo. Claramente

estes regimes aplicam este tipo de medidas de forma a moldar fácilmente as ideias e instrução da sua sociedade, de forma a obter um melhor controlo sobre as mesmas e reduzir ou eliminar completamente possíveis protestos ou tentativas de colapso contra regimes autocráticos (como o Egito) e ditatoriais (como o Irão).

Apesar dos esforços do *Signal* se manter ativo nestes países através da técnica de *domain fronting*,

- EARN IT Act (child abuse law to implement backdoors on social messaging apps like signal)
  - `https://www.pcmag.com/news/messaging-app-signal-threatens-to-dump-us-market-if-anti-encryption-bill`
  - por um lado, abusos de menores é um problema sério, que tem de ser reduzido ou extreminado de alguma meneira
  - contudo, não é com este tipo de leis, já que abusadores podem sempre usar outras aplicações mais "não convencionais"
  - outro problema é o facto deste tipo de leis reduzirem a liberdade da população duma sociedade: "But other lawmakers say they're against the bill, citing its potential to be abused. "This terrible legislation is a Trojan horse to give Attorney General Barr and Donald Trump the power to control online speech and require government access to every aspect of Americans' lives,"said Senator Ron Wyden (D-Oregon) last month."
  - esta lei obviamente traz coimas, coisa que empresas grandes como o facebook ou google conseguem suportar, mas que o signal não consegue, porque é apenas um projeto duma associação sem fins lucrativos
  - `https://signal.org/blog/earn-it/`
  - vai tudo contra liberdade de expressão
  - contra a lei "Section 230 of the Communications Decency Act", que dita que qualquer aplicação ou website não é responsável pelo conteudo que os seus utilizadores publicam na internet
- não é uma aplicação intrusiva -> não tem anúncios
- não possui um marketing forte como outros rivais de grandes empresas, pelo que sofre pela impopularidade
- problema de mensagens com limite de tempo (auto destroem-se)
- utilização de domain fronting `https://signal.org/blog/looking-back-on-the-front/`, o que torna possivel usar uma aplicação com este tipo de poder num país onde são banidas (por exemplo, no Irão, o WhatsApp está banido)
- o protocolo tem sido usado por outras aplicações de concorrência, instigando a comunicação segura na internet -> pontos fortes e fracos
- problemas com a implementação de backdoors também na Australia `https://qz.com/1497092/the-signal-encrypted-app-service-wont-comply-with-australias-assistance-and-access-bill/`

3 Referências

[1] Committee on the Judiciary. Graham, blumenthal, hawley, feinstein introduce earn it act to encourage tech industry to take online child sexual exploitation seriously, 2020.

[2] Riana Pfefferkorn. The earn it act is here. surprise, it's still bad news., 2020.

[3] Joshua Lund. 230, or not 230? that is the earn it question., 2020.

4 Apêndice

4.1 Important Notes