



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
2.2 PROBLEMAS ECONÓMICOS	21
2.3 PROBLEMAS ÉTICOS	21
3. REFERÊNCIAS	23
4. APÊNDICE	24
4.1 IMPORTANT NOTES	24



1 Protocolo

Toda a informação presente foi obtida da documentação do Signal [1].

1.1 Primitivas

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

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

$$Combinacoes = 2^{bits} = 2^{512} = 1.340781^{154} \quad (1)$$

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

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

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

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

O que significa que seriam necessários 1500 vezes o tempo que a terra existe para poder decifrar uma *hash* efetuada 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 análise 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 == 1 or cV == 1 or Bv == 1:
        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 objetivos 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*.

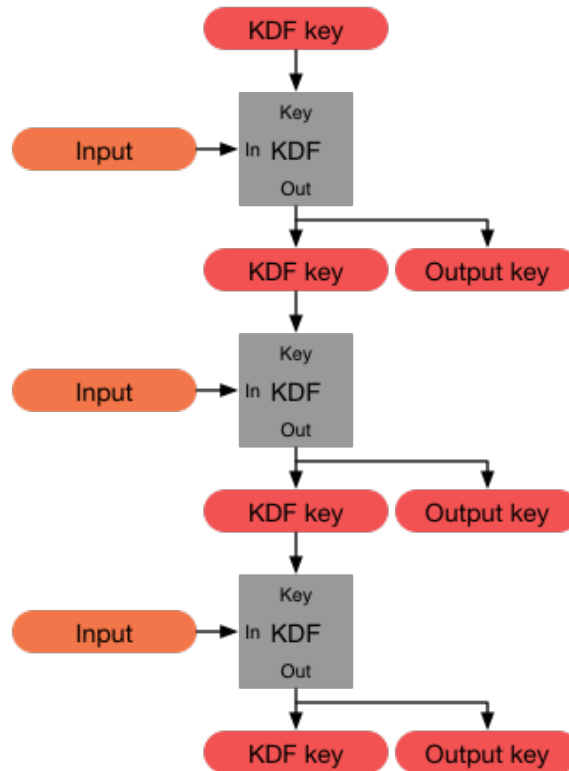


Figura 1: KDF Chain com 3 inputs

Uma *KDF Chain* apresenta as seguintes propriedades:

- **Resiliência:** Se não houver conhecimento da chave KDF, as chaves de output aparentam ser *random*. Isto mantém-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 receção 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 efetuando 2 *ratchet steps*.

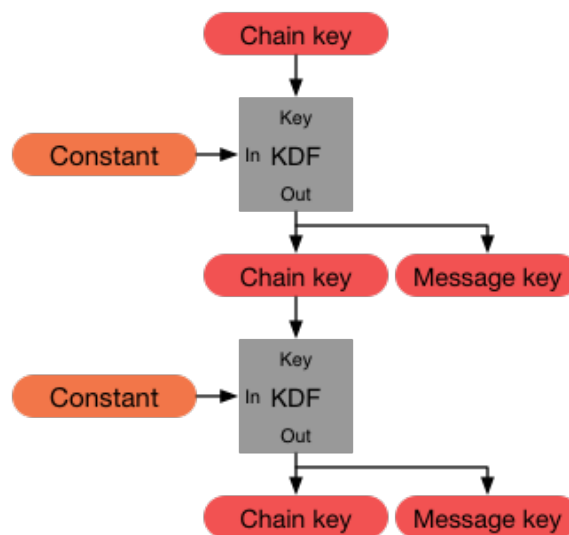


Figura 2: Symmetric-key ratchet with 2 steps

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 pública**. Quando o receptor recebe a chave pública do emissor é efetuado 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-Hellman* entre a sua *ratchet key* private e a *ratchet key* publica do Bob.

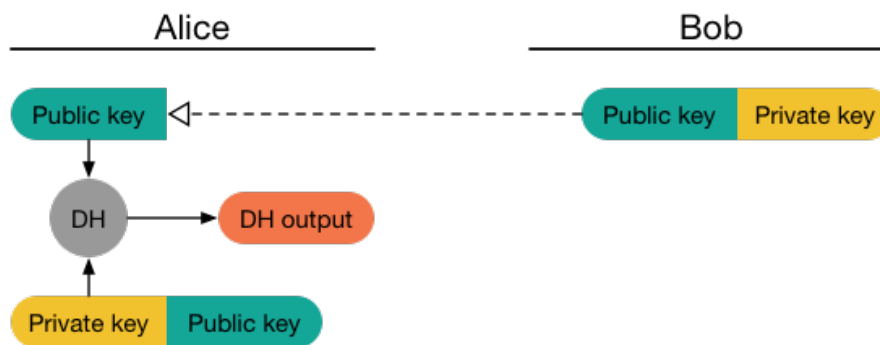


Figura 3: Mensagem enviada do Bob para a Alice

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*.

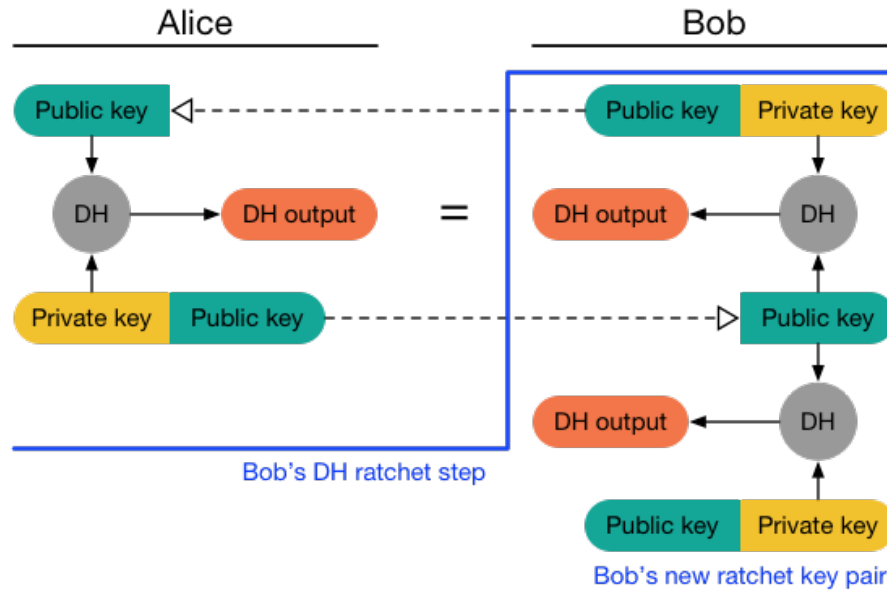


Figura 4: Mensagem enviada da Alice para o Bob após a mensagem inicial

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.

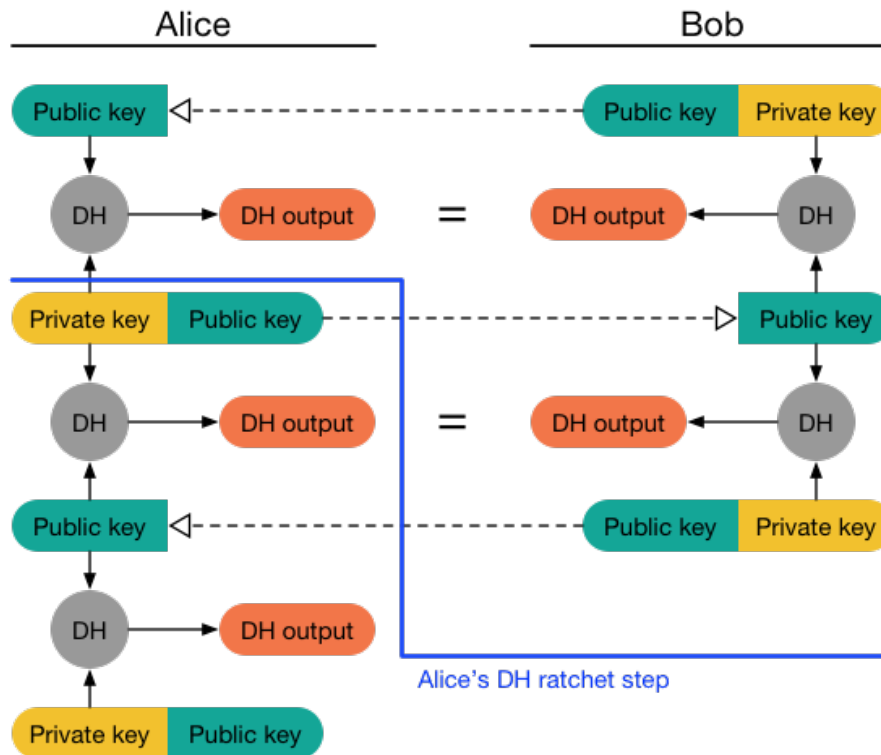


Figura 5: Nova mensagem do Bob com a Alice como destinatário

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.

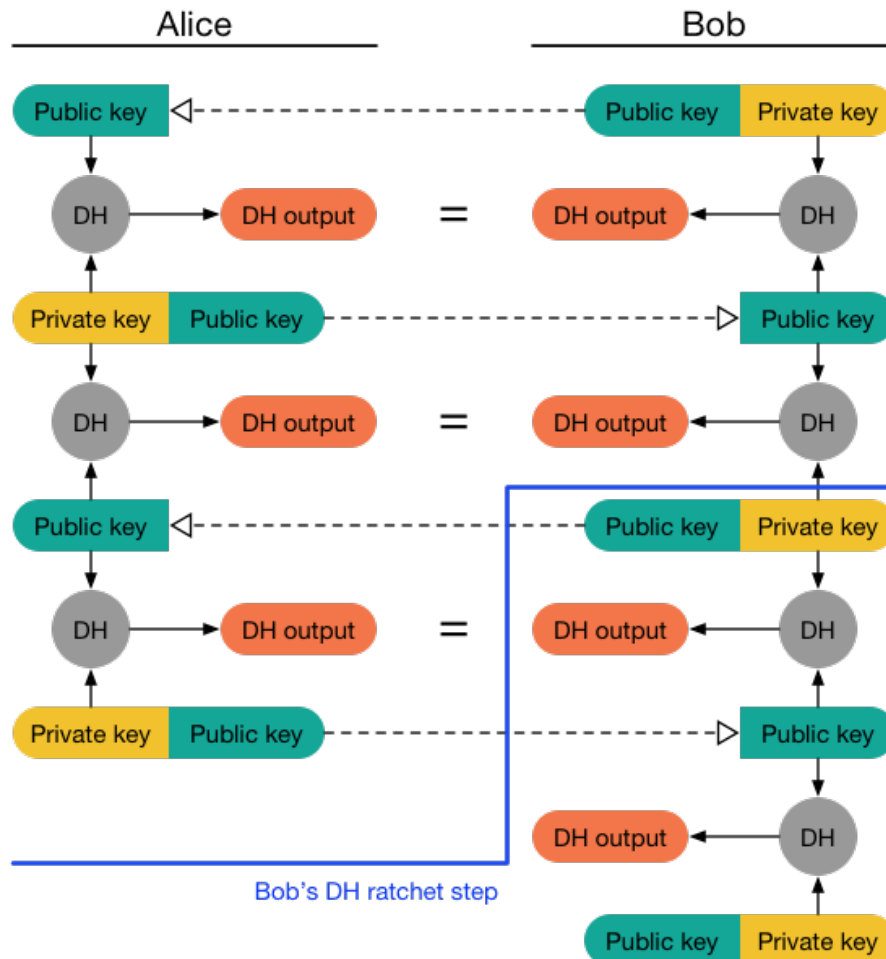


Figura 6: Nova mensagem por parte da Alice

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

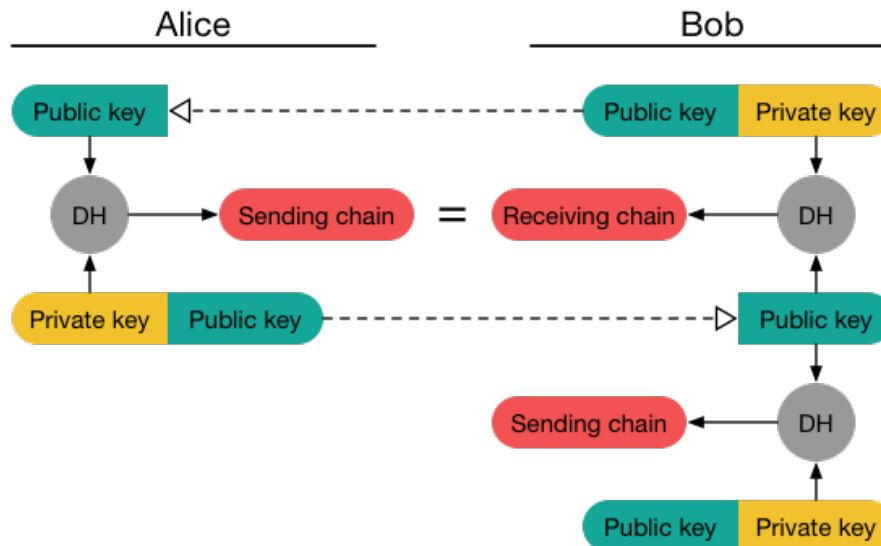


Figura 7: Mensagem inicial por parte do Bob e resposta por parte da Alice , mostrando mais especificamente as cadeias de receção e de envio

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

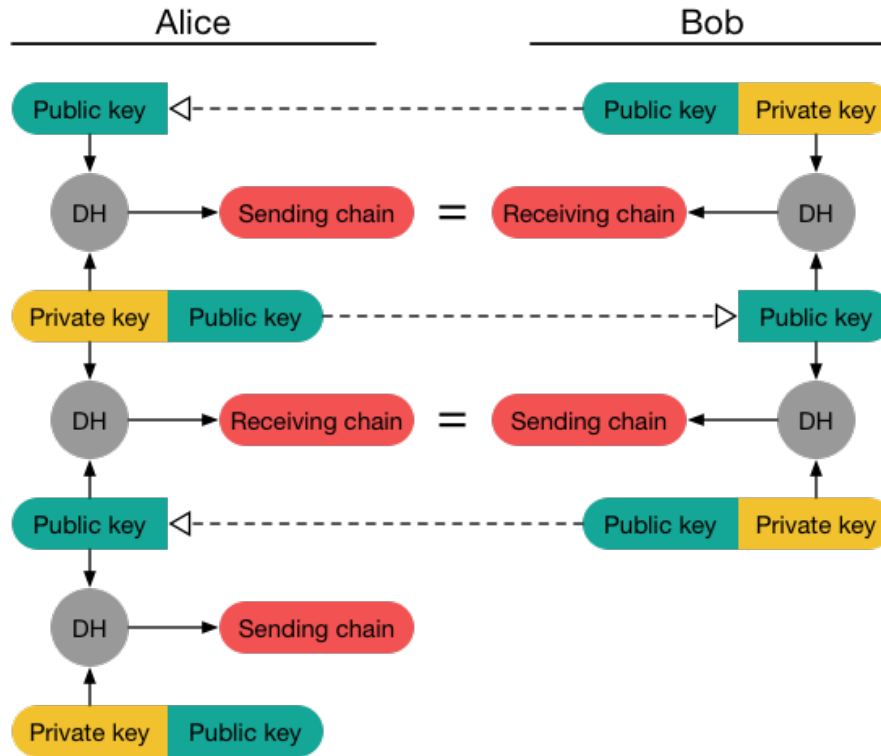


Figura 8: Nova mensagem por parte do Bob, mostrando as cadeias de receção e de envio

No entanto a imagem em cima (fig. 8), é uma simplificado. Em vez de se usar diretamente os *DH outputs*, estes são usados como *KDF (1.2) inputs* para uma *root chain*, sendo os *KDF outputs* da *root chain* usados como chaves de envio e receção. 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 receção.

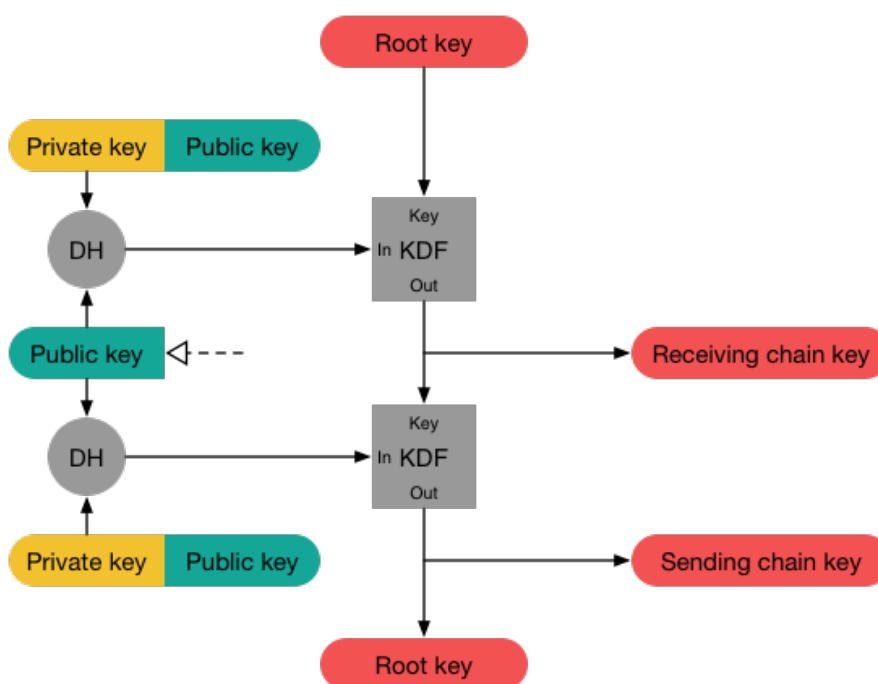


Figura 9: Imagem que demonstra o processo por de trás da geração das cadeias mencionadas nas imagens 7 e 8

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 das chaves mais recentes. Os usuários enviam valores públicos de *Diffie-Hellman* juntamente com as suas mensagens. Os resultados dos cálculos de *Diffie-Hellman* são misturados com a chave derivada anteriormente de modo a que chaves anteriores não possam calcular chaves mais recentes. Estas propriedades garantem proteção perante as 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, um passo de *Symmetric-Key ratchet* é aplicado nas cadeias de envio ou receção 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.

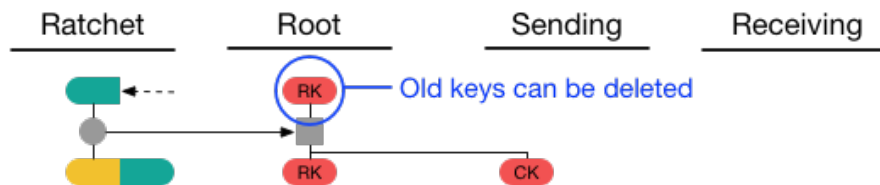


Figura 10: Primeira mensagem por parte do Bob e inicialização da chave raiz e chave de envio

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.

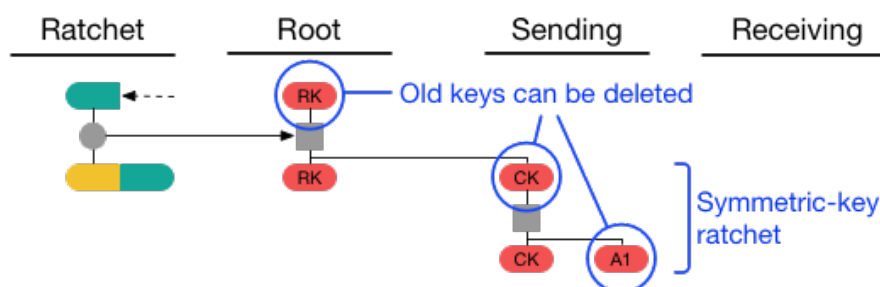


Figura 11: Demonstração de um passo na symmetric-key ratchet após receção da primeira mensagem

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

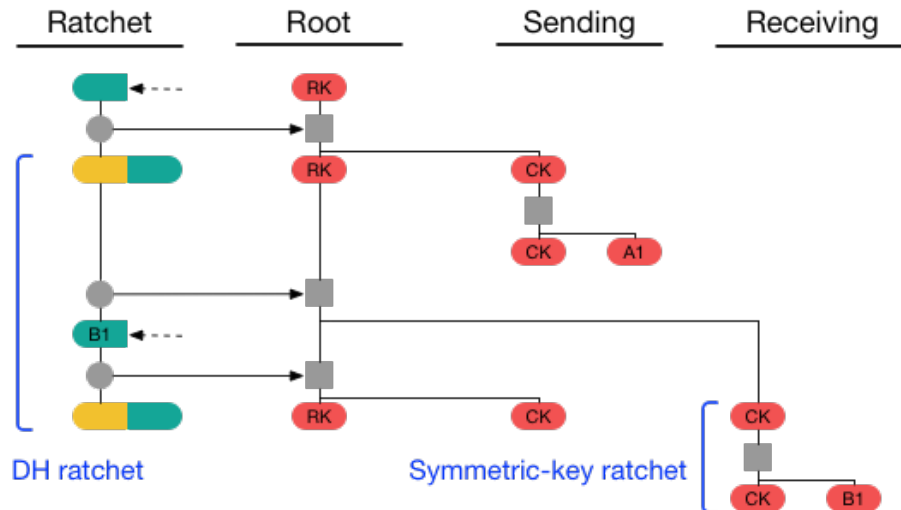


Figura 12: Demonstração de um passo na symmetric-key ratchet após receção da segunda mensagem vinda do Bob levando à geração de uma chave de receção

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

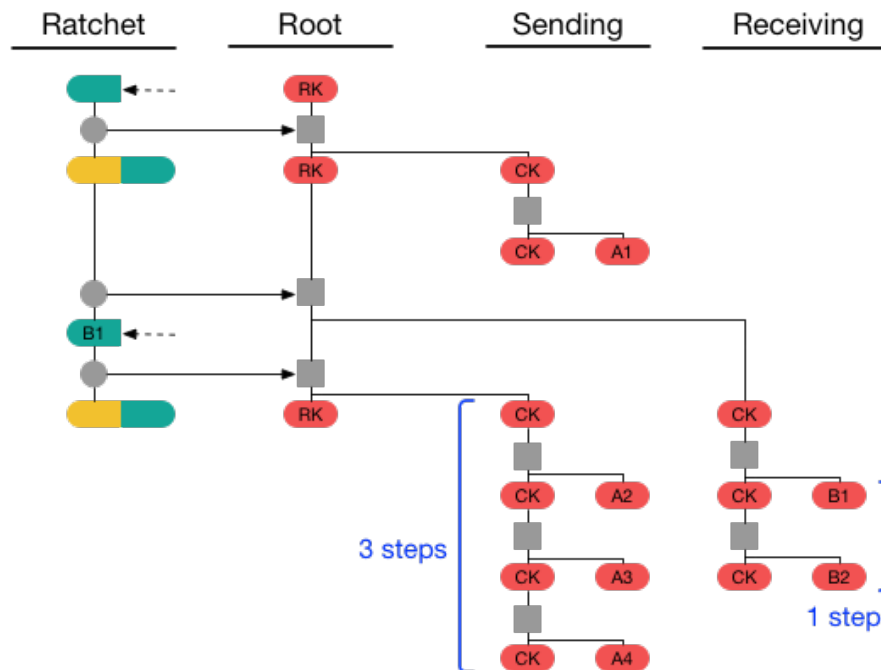


Figura 13: Demonstração de alguns casos específicos do Double-Ratchet

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

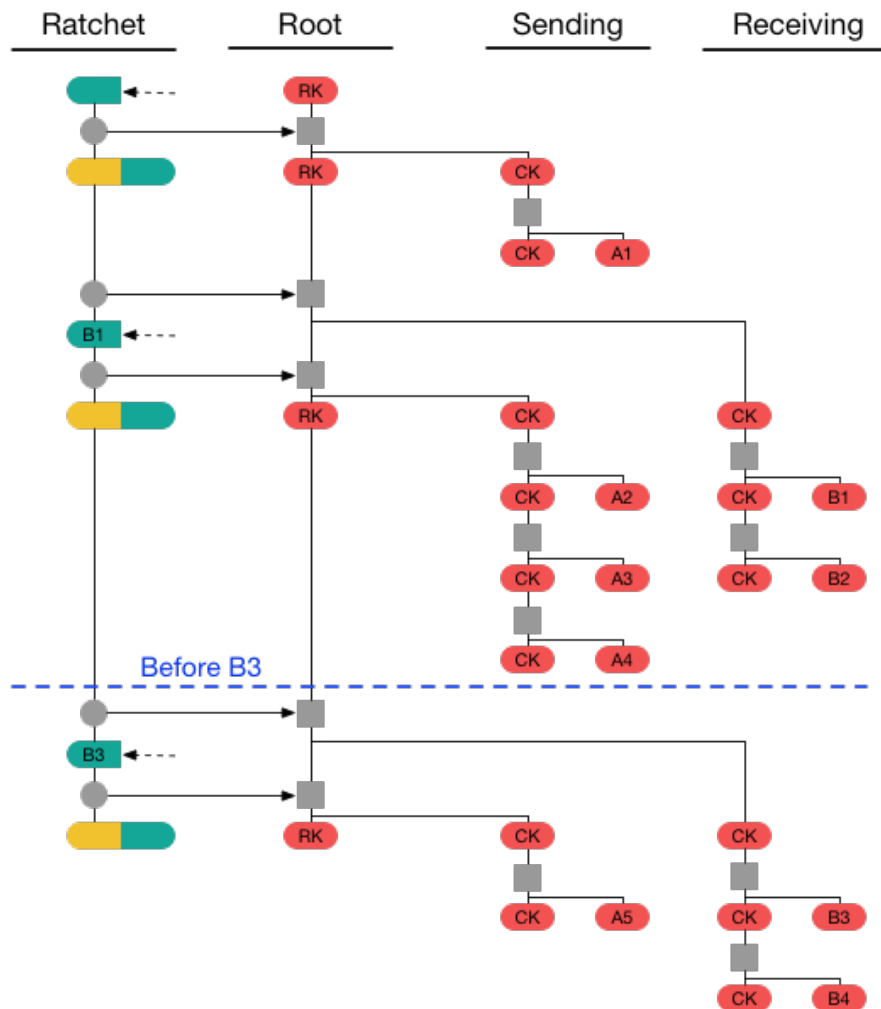


Figura 14: Estado final da cadeia

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 sólido 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 políticas de grandes países [2]), pois o Signal é **Open-Source**, permitindo que qualquer pessoa veja o código e detete 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 ativado por defeito.



Excluindo o que foi mencionado em cima, o Signal ainda proporciona uma excelente documentação [1] 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 detetar 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** [3].
- 2016,Outubro, investigadores de *University of Oxford,Queensland University of Technology and McMaster University* publicam uma analise formal do protocolo concluído que o protocolo era **criptograficamente seguro** [4].
- 2017,Julho, investigadores de *Ruhr University Bochum* durante uma nova análise detetam uma falha puramente teórica, uma vez que esta ao acontecer seria automaticamente detetada, voltando a **validar a declaração anterior** [5].

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 detetar 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 *WhatsApp*, o Signal não guarda dados (ou metadados que por vezes ainda são mais importantes) do cliente e se este quiser privacidade o Signal garante-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 detetar 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 mantêm-se guardados num servidor caso sejam apagadas.

Além de tudo o referido anteriormente o Signal encontra-se ativamente 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 políticas, que pretendem obter mais informação sobre a população ou dos cidadãos do seu país. Desta



forma , alguns países como por exemplo o Egito (e.g Egito bloqueia todas as comunicações Signal [6]), bloqueiam o tráfego de aplicações como o Signal , consequentemente reduzindo a privacidade das pessoas.

O Signal entevem neste aspecto , mais uma vez procurando devolver a privacidade aos cidadãos, usando um mecanismo chamado **Domain Fronting**. Este mecanismo ,como referido no artigo [7], 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 tráfego 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 [8],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* [9] [7]). 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 [8]), mas deixou uma marca que demonstra que ,internamente,o seu protocolo visa e procura manter a privacidade das pessoas.

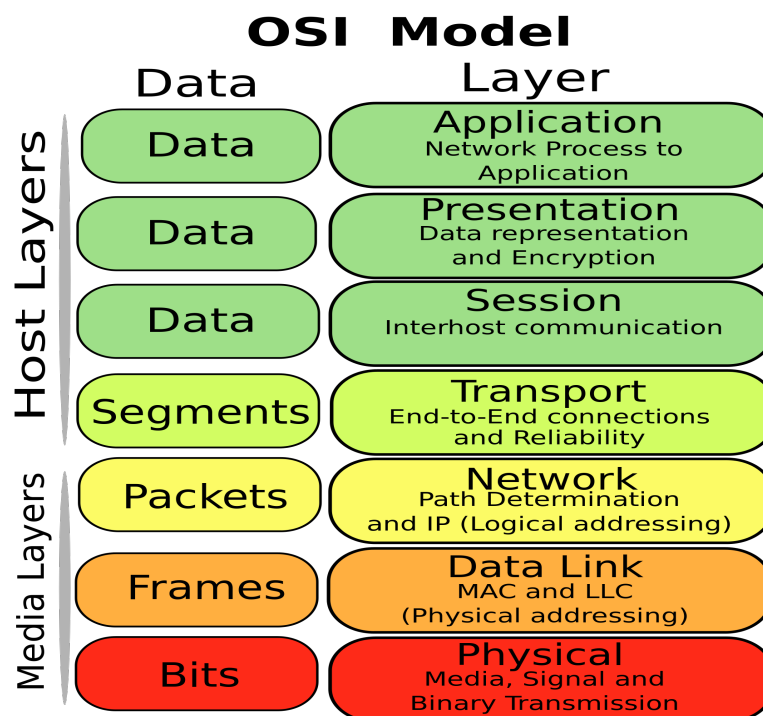


Figura 15: Figura representativa das camadas OSI, dando ênfase à camada aplicacional [10].

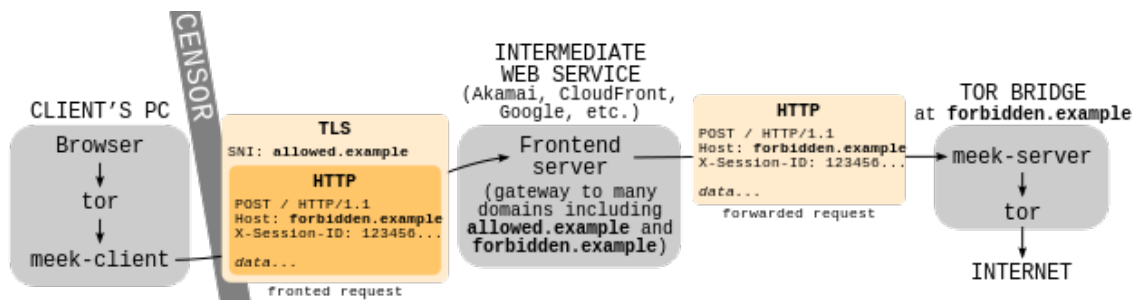


Figura 16: Funcionamento do Domain Fronting [11].



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 público alvo desta. Neste caso, o público alvo do *Signal* é qualquer pessoa que pretende realizar comunicações seguras e secretas *online*. Tendo isso em consideração, temos pessoas como "maníacos" da cibersegurança, políticos, 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 dum 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 [12]. 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 conjunto 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* [13].

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 cidadãos 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 [14].

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 facilmente 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*, em 2018 a *Google* e a *Amazon* impossibilitaram a utilização deste método usando os seus serviços, tornando inacessível o uso do *Signal* e outras aplicações que garantem comunicações seguras nos países descritos [15]. É claro que quem o queira continuar a fazer, o pode consumir aproveitando-se, por exemplo, duma *VPN*. Contudo, este tipo de ferramentas são mais disseminadas por pessoas informadas do ramo da informática, pelo que para um "cidadão comum" pode não ser o mais evidente.

Apesar dos esforços que este género de registes tem feito para limitar a disseminação de informação nos seus países, têm felizmente continuado a haver esforços de serviços como o *Signal* para continuar a haver fontes de divulgação de conhecimento onde este é escasso.

2.2 Problemas económicos

Como já explicado no início deste relatório, o *Signal* pertence a uma fundação sem fins lucrativos, pelo que não possui qualquer modelo de negócio, ou pelo menos um claro. A sua principal ambição é criar o ambiente mais seguro possível para as comunicações realizadas pelos seus utilizadores. Por esse motivo, os problemas económicos que ele pode criar advém mais do mercado que tiram à sua concorrência do que do próprio modelo de negócio.

Sendo o seu principal oponente o *WhatsApp*, pode-se facilmente entender que, apesar do *Signal* ter sido lançado para o público em 2014, desde então não parece ter tirado "grande terreno" ao seu rival, sendo que este possui aproximadamente mil milhões e meio de utilizadores ativos mensalmente e o *Signal* possuir mais de 10 milhões de *downloads* no *Google Play* (não existe quaisquer dados públicos das estatísticas da utilização do serviço, pelo que se torna fazer uma comparação de outra forma). Mesmo em relação ao *Telegram*, que possui cerca de 200 milhões de utilizadores mensais ativos, outra aplicação que oferece *E2E*, é fácil entender que apresenta uma muito menor quantidade de utilizadores [16].

Conclui-se assim, que duma forma geral, este serviço não tem uma presença acentuada no mercado, pelo que não pode apresentar problemas económicos graves no grande panorama.

Contudo, pelo facto de não apresentar um modelo económico usual, isso pode traduzir-se em problemas económicos internos. Pelo facto do serviço pertencer a uma associação sem fins lucrativos, torna-se impraticável gerar dinheiro da forma típica, sobrevivendo apenas através de doações. Para além de isso implicar a existência duma equipa de programadores bastante reduzida (falamos aqui dos trabalhadores pagos, não da comunidade *open source*), é também impossível, por exemplo, financiar ações judiciais que possam haver contra a aplicação. Por esse motivo é que à data da criação deste documento o *Signal* anuncia uma possível cessão de funções em solo americano caso a lei *EARN IT* seja aprovada, já que não possui quaisquer meios económicos que suportem possíveis ações judiciais dos seus utilizadores, ao contrário das empresas concorrentes, que possuem um modelo financeiro bem estruturado.

2.3 Problemas éticos

Como o *Signal* é um meio seguro para fazer comunicações, os seus problemas éticos vão muito de encontro aos problemas éticos do uso de encriptação no geral. Como discutido nos problemas sociais, este tipo de serviços são normalmente usados para praticar crimes, como abuso de menores ou planeamento de ataques



terroristas. Contudo, é também através deles que muitos cidadãos possuem uma ferramenta para disseminar informação, sendo por isso um meio importante para a existência de pensamento crítico livre em países onde não o é permitido. Será que o bom compensa o mau no uso desta aplicação? Claramente, não existe uma resposta correta, mas podemos fazer várias constatações. Como já falado antes, quem quer cometer este tipo de crimes, não necessita do *Signal* para o fazer, há métodos menos convencionais fora do controlo do governo que funcionam da mesma maneira. Contudo, o que se pode dizer no caso de pessoas que vivem num regime de opressão? Pessoas que nunca tiveram o acesso que nós, num país democrático, temos desde sempre à tecnologia, e que por isso não têm conhecimentos suficientes para saber sequer da existência de outros meios para fazer comunicações seguras e anónimas que não uma simples aplicação de telemóvel?

Não podemos dar uma resposta com toda a certeza de se é ou não moralmente correta a existência duma ferramenta deste tipo, mas podemos dar a nossa opinião dados os argumentos anteriores. Para nós, apesar do mal que advém da existência dela, a privacidade é um direito importante que, enquanto pessoas, temos e que não deve ser reduzido ou posto de parte por haver quem o use para atividades maliciosas.



3 Referências

- [1] Signal organization. Signal protocol information.
- [2] Laurens Cerulus. Eu commission to staff: Switch to signal messaging app, 2020.
- [3] Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Jörg Schwenk, and Thorsten Holz. How secure is textsecure? In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 457–472. IEEE, 2016.
- [4] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 451–466. IEEE, 2017.
- [5] Paul Rösler, Christian Mainka, and Jörg Schwenk. More is less: on the end-to-end security of group chats in signal, whatsapp, and threema. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 415–429. IEEE, 2018.
- [6] Farid Farid. No signal: Egypt blocks the encrypted messaging app as it continues its cyber crackdown, 2016.
- [7] Artem Rukavytsia. Domain fronting in a nutshell, 2019.
- [8] Moxie Marlinspike. Looking back on the front, 2018.
- [9] Matt Burges. This is why russia’s attempts to block telegram have failed, 2018.
- [10] Ian Ames. The osi model, 2019.
- [11] Pierluigi Paganini. Apt29 group used domain fronting to evade detection long before these techniques were widely known, 2017.
- [12] Committee on the Judiciary. Graham, blumenthal, hawley, feinstein introduce earn it act to encourage tech industry to take online child sexual exploitation seriously, 2020.
- [13] Riana Pfefferkorn. The earn it act is here. surprise, it’s still bad news., 2020.
- [14] Joshua Lund. 230, or not 230? that is the earn it question., 2020.
- [15] Moxie Marlinspike. A letter from amazon, 2018.
- [16] Nsoor Iqbal. Whatsapp revenue and usage statistics (2020), 2020.



4 Apêndice

4.1 Important Notes