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

Date: **21/04/2020**

Indice

1 Protocolo

1.1 KDF chains

KDF Chains faz parte do *core* de um dos principais métodos, *Double Ratchet* (secção 1.4), 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*.

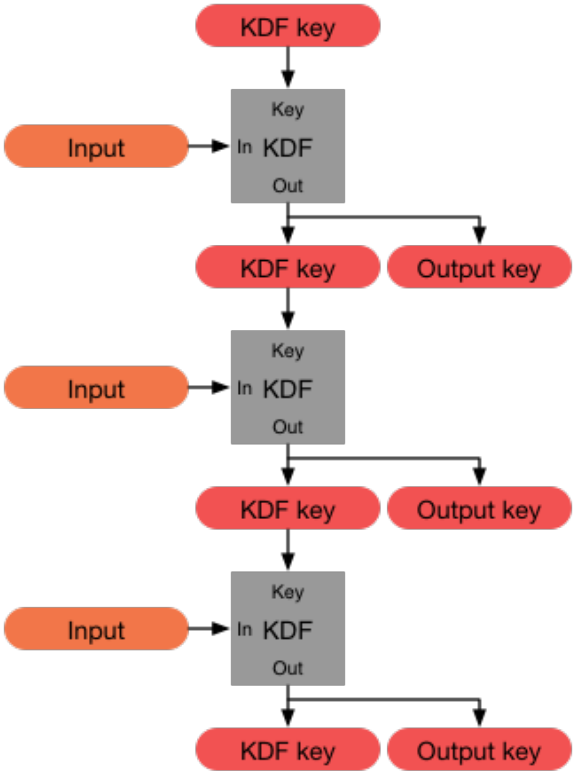

Figure 1: KDF Chain com 3 inputs

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 1.4), 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**.

1.2 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 1.1 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*.

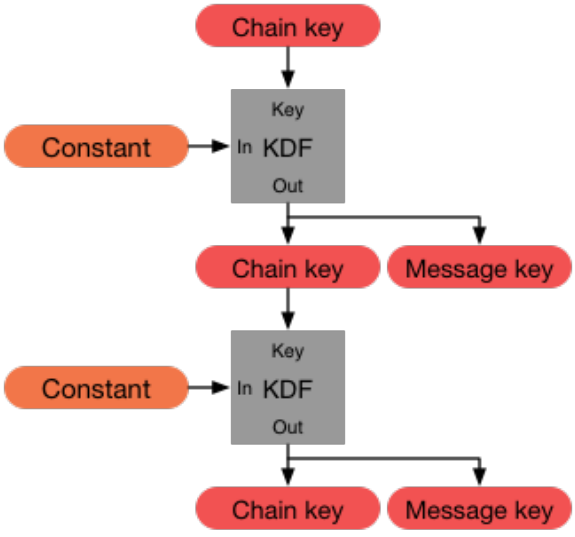

Figure 2: Symmetric-key ratchet with 2 steps

1.3 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.

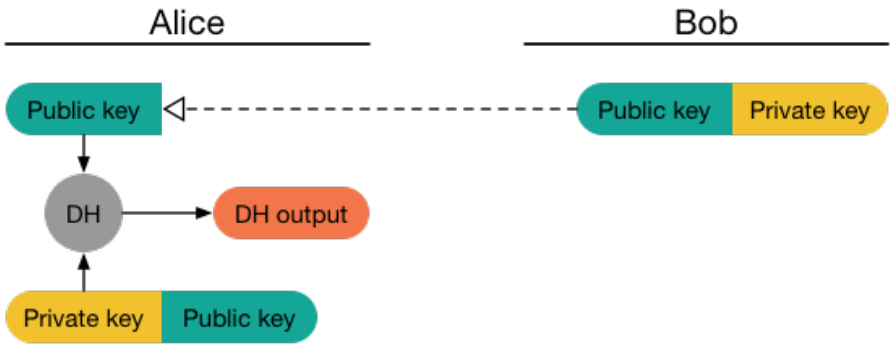

Figure 3:

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*.

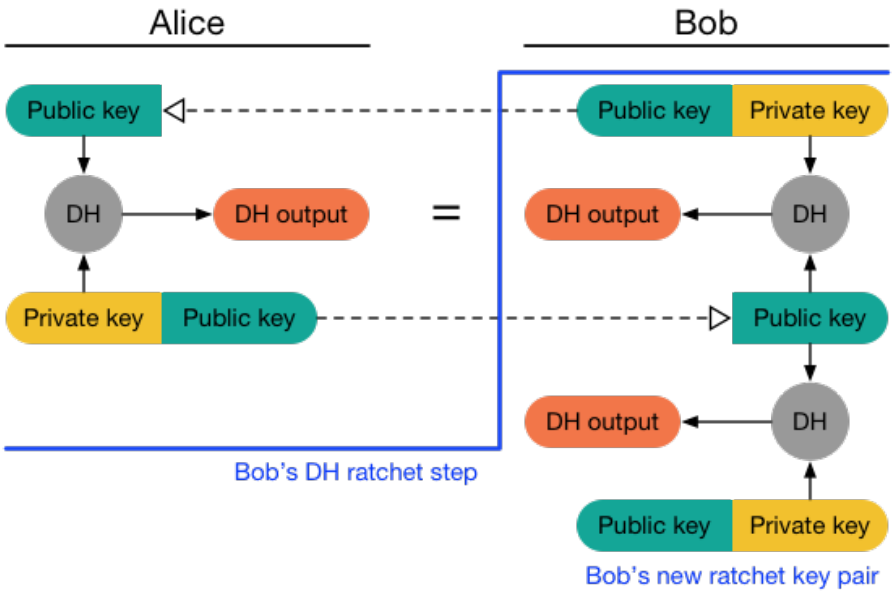

Figure 4:

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.

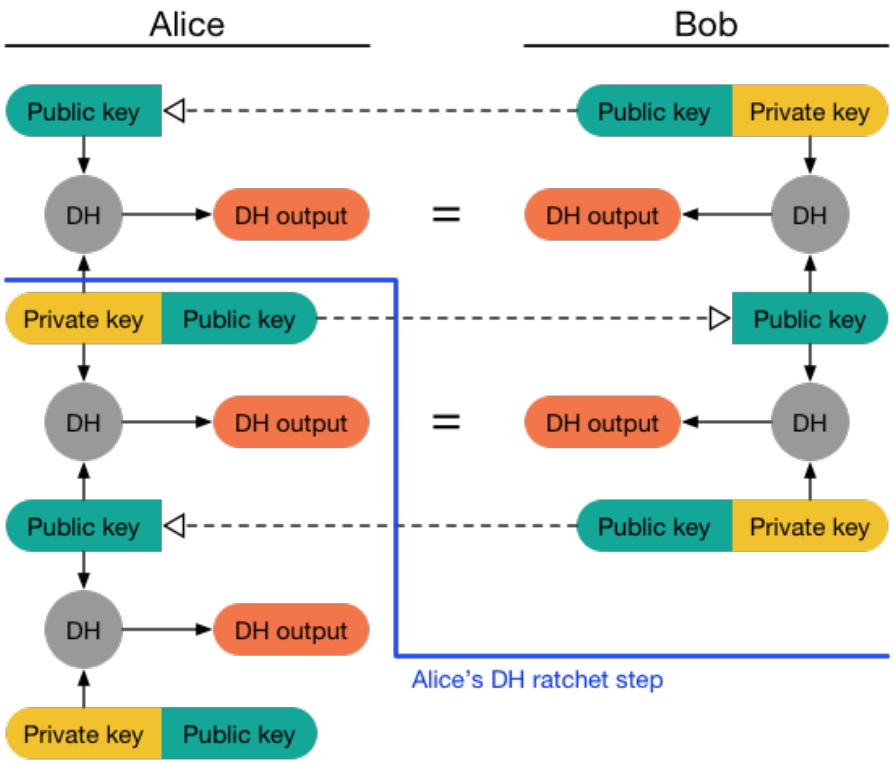

Figure 5:

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.

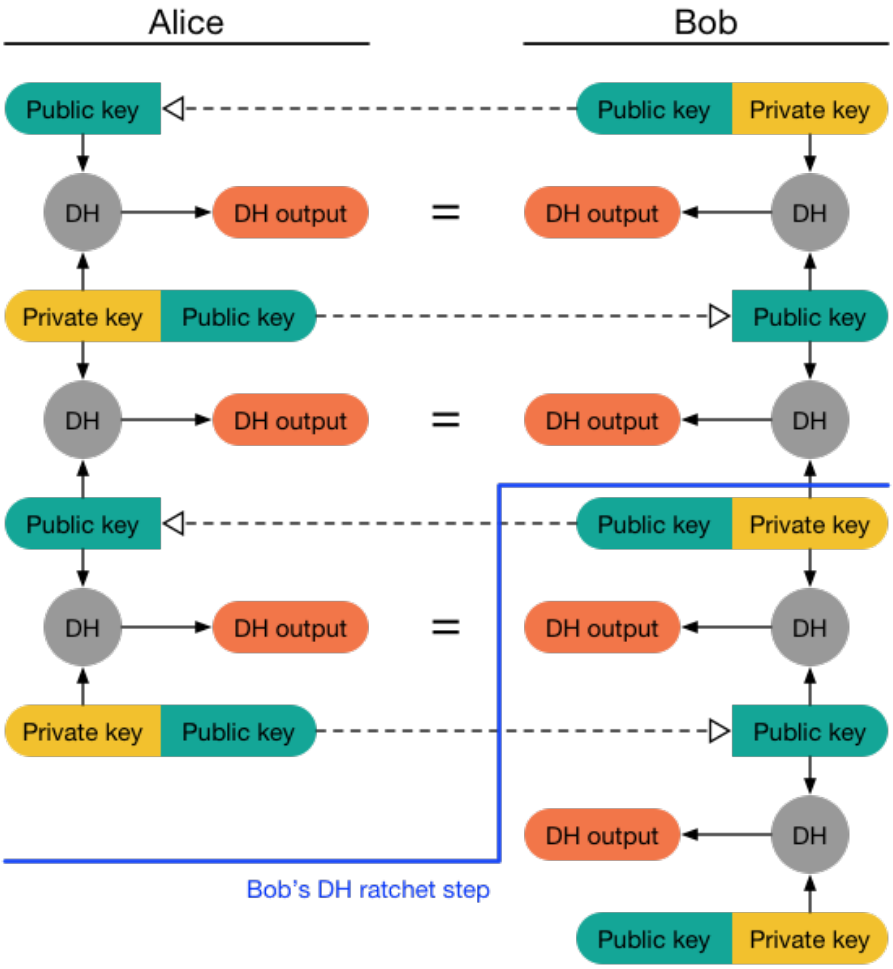

Figure 6:

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.

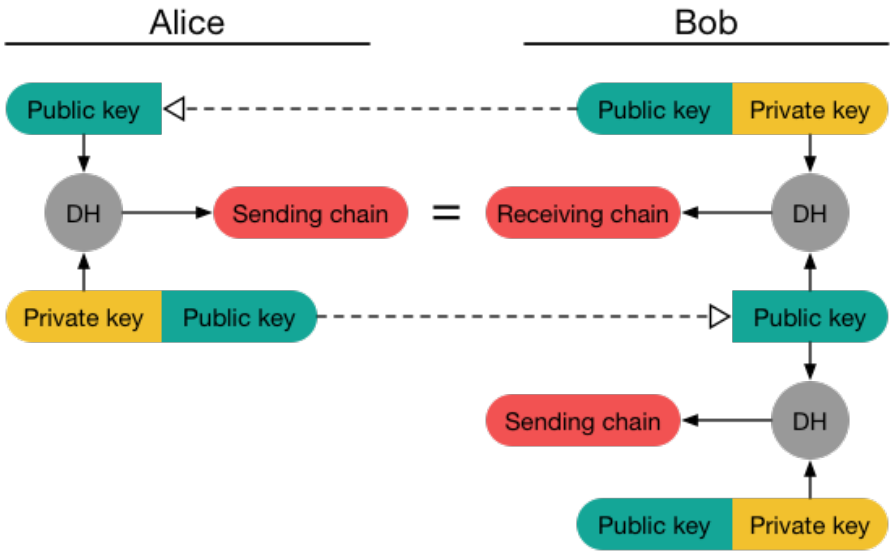

Figure 7:

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

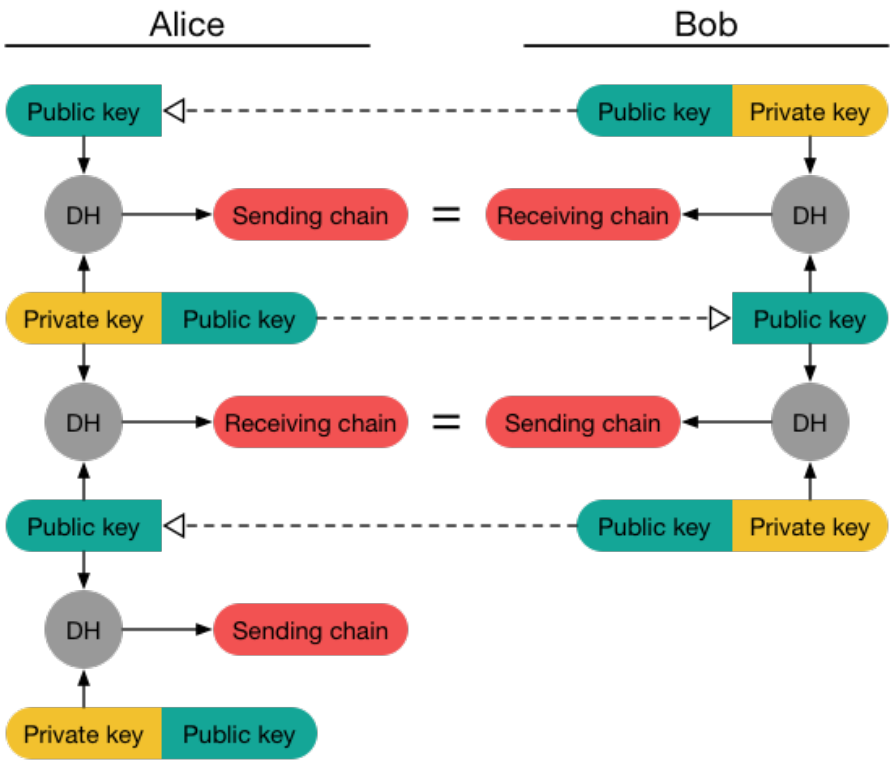

Figure 8:

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 (1.1) 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.

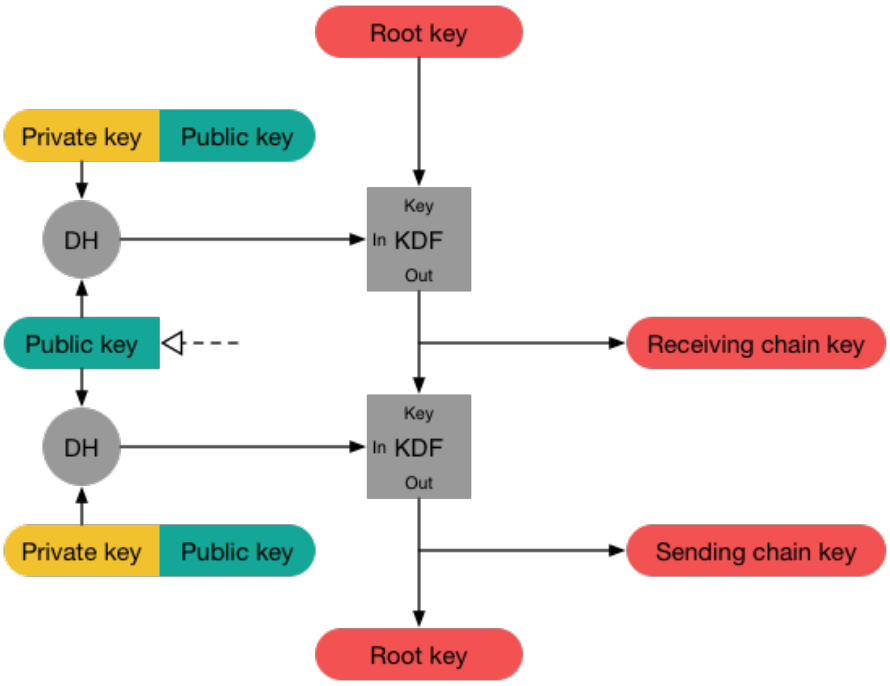

Figure 9:

1.4 Double Ratchet

1.4.1 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.

1.4.2 Como funciona a Double Ratchet?

A algoritmo de ***Double Ratchet*** é a combinação de *Symmetric-Key (sec.1.2)* com *DH Ratchets (sec. 1.3)* :

- 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 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.

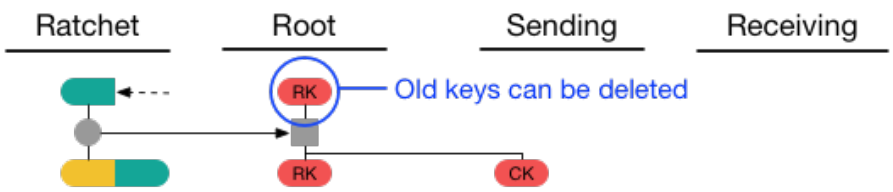

Figure 10:

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.

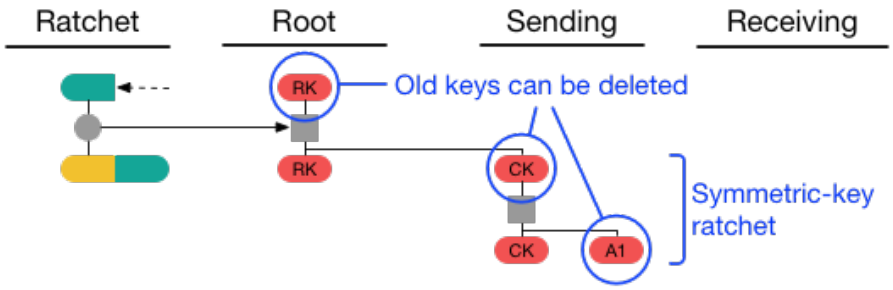

Figure 11:

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.

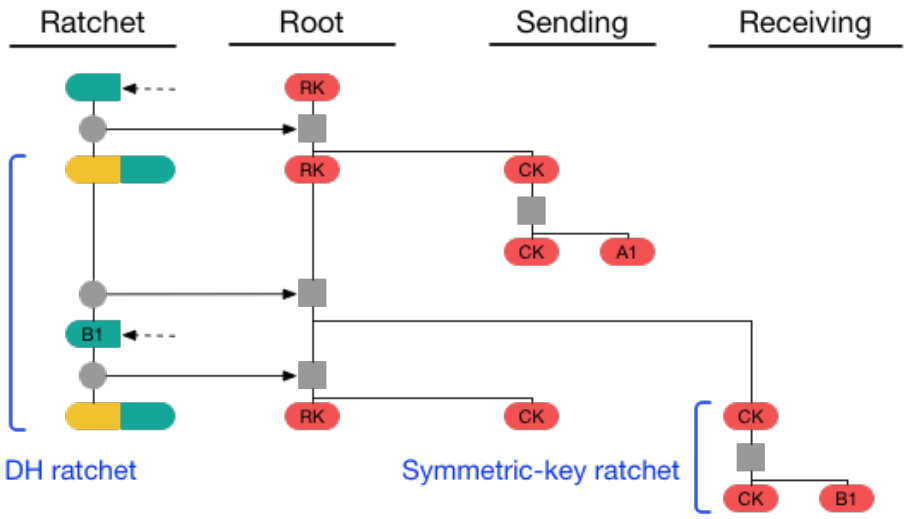

Figure 12:

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.

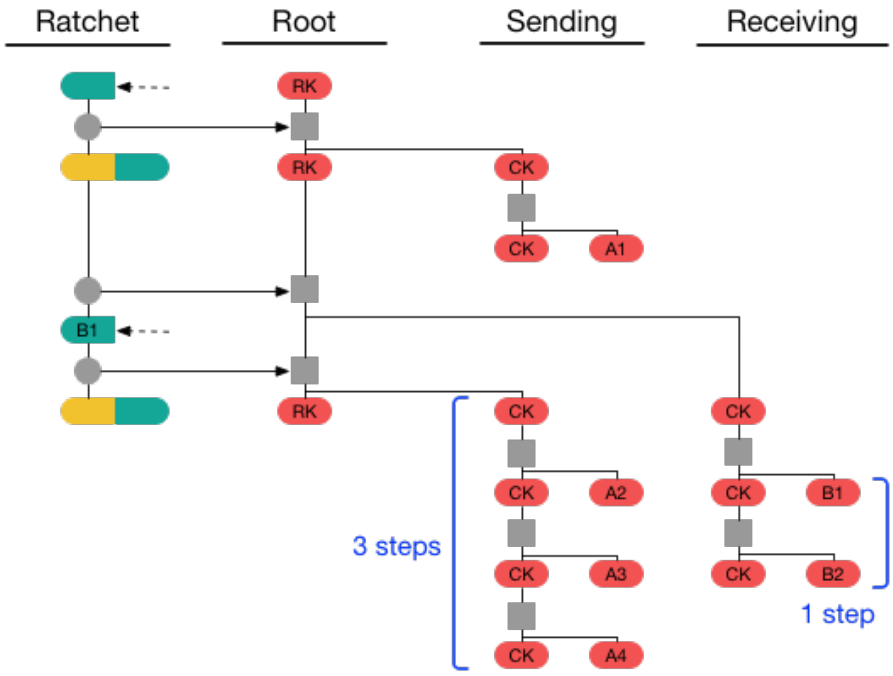

Figure 13:

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.

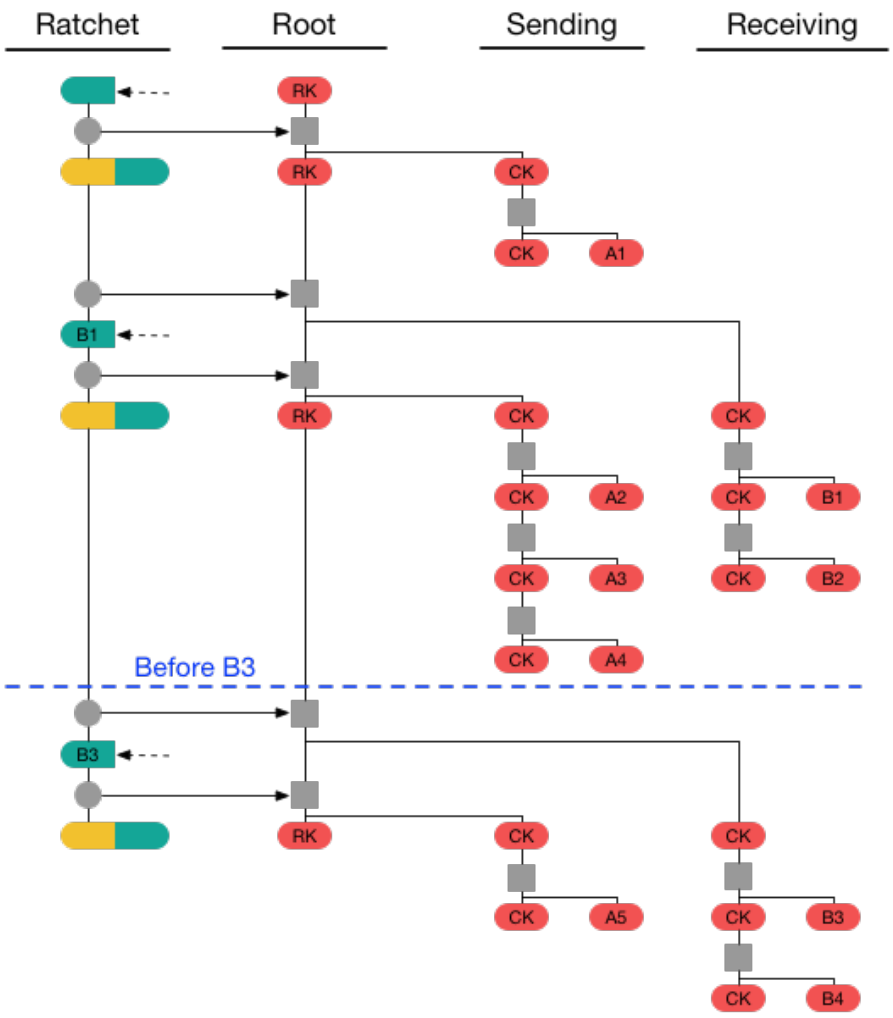

Figure 14:

1.5 O que torna o protocolo do Signal superior?

FALAR DO QUE ESTA NA WIKEPEDIA NA PARTE DE ENCRYPTION PROTOCOLS ADICIONAR UM BCD DAS PRIMITIVAS USADAS PARA ENCRIPTACAO

2 Referências

[1] Simon K., *Digital trends 2019: Every single stat you need to know about the internet* [Online], Available online [25/02/2019]: `https://thenextweb.com/contributors/2019/01/30/digital-trends-2019-every-single-stat-you-need-to-know-about-the-internet`

30, jan. 2019.

3 Appendix

3.1 Important Notes