Criptografia (Aplicada)

2022/2023 - MSI/MCC/MERSI

Manuel Barbosa
mbb@fc.up.pt

Aula 12
TLS and Signal

Transport Layer Security (TLS)

Security Model

- Network attacker:
  - controls all infrastructure: routers, DNS, etc.
  - Eavesdrops, injects, drops, changes packages/messages.
- Examples:
  - wireless public network, e.g., coffee shop/store
  - network access in a hotel, at work
  - our own ISP …

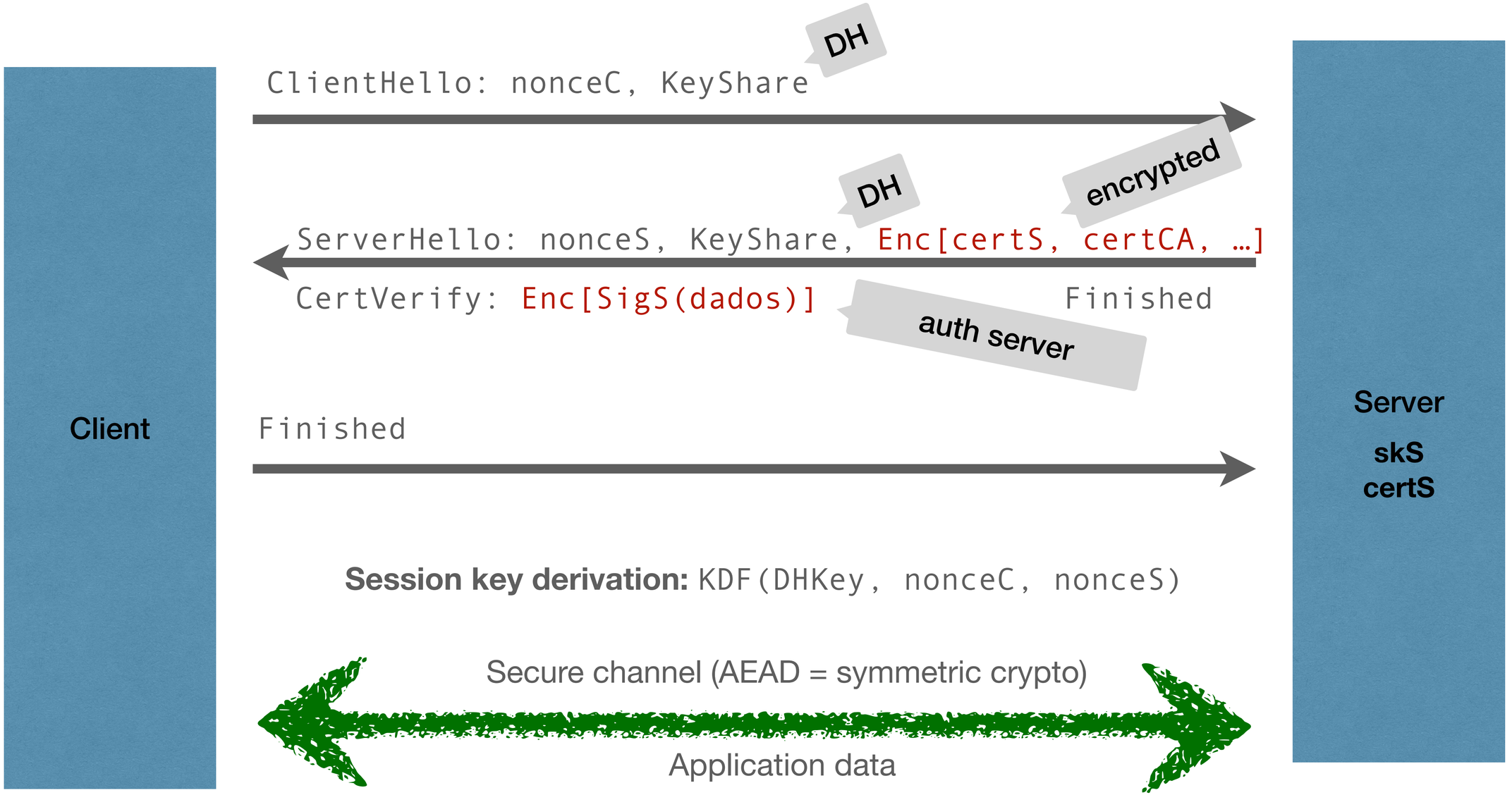
Handshake TLS1.3 = Authenticated Diffie-Hellman

- Historically the TLS handshake used mostly:
  - session key transport from client to server (RSA)
  - implicit server authentication => correct use of secret key
- Today the advantages of DH are clear:
  - much more efficient using elliptic curves
  - perfect forward secrecy:
    - long-term keys are used for digital signatures and not session key transport
    - compromising long-term key does not compromise past key agreement executions

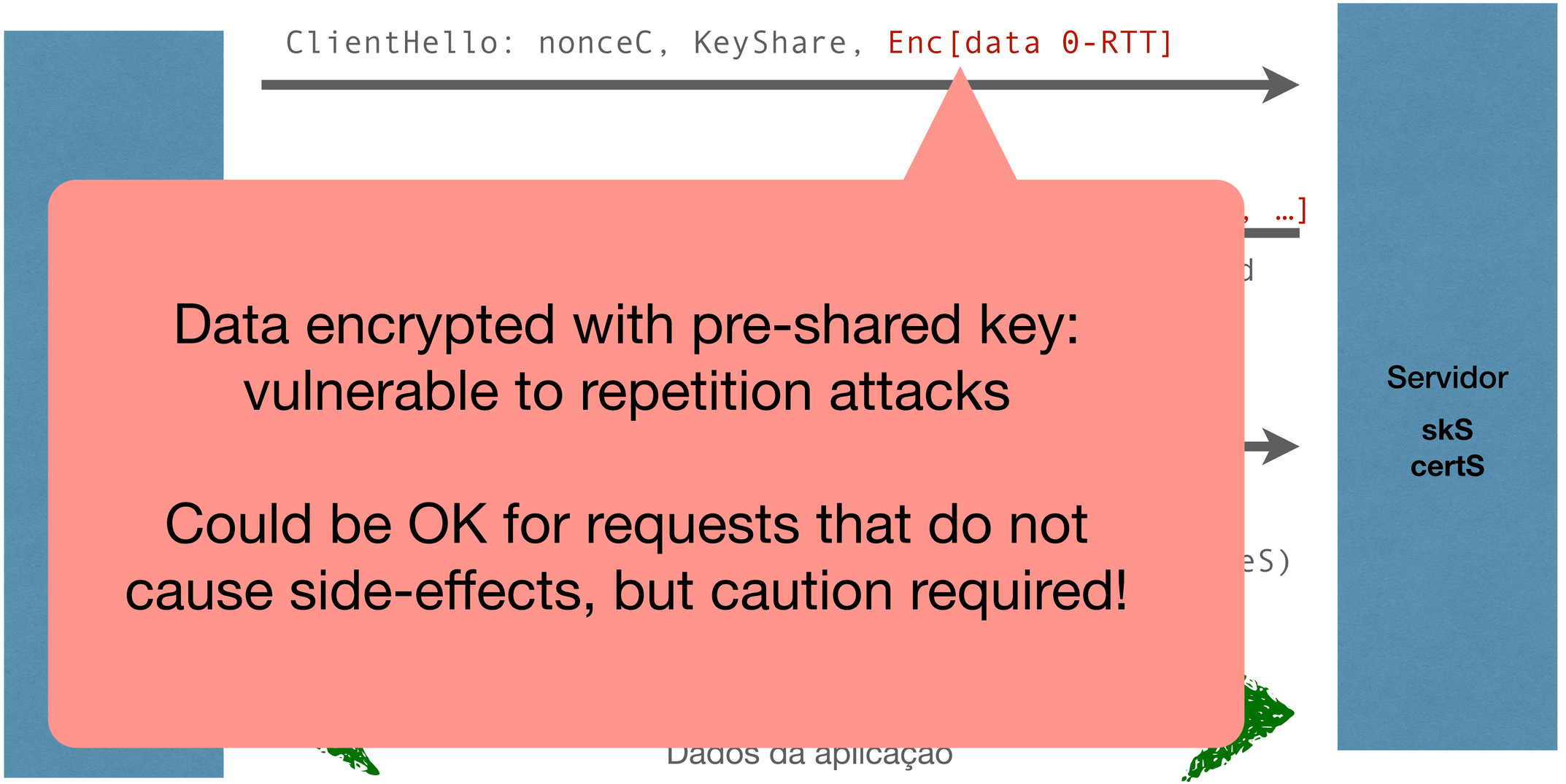
TLS requires PKI

- Server authenticates the DH exchange using a digital signature (client signature is optional)
- How does the client know which key to verify server signature?
  - Server sends public-key certificate
  - Client validates public-key certificate (pre-installed root CAs) =>
    - domain name must match certificate subject
    - it is possible to use wildcards in the certificate for the leftmost component, e.g., *.a.com
  - Client uses the public key to verify the server's signature
- Recall: client usually not authenticated => server could be talking to anyone

Handshake TLS 1.3



Handshake TLS 1.3: Otimization (caution)



Data encrypted with pre-shared key:
vulnerable to repetition attacks

Could be OK for requests that do not cause side-effects, but caution required!

Integration TLS/HTTP

- HTTP messages are payloads transmitted inside the secure channel (hidden)
- Problems/solutions:
  - web proxy: proxy needs to know HTTP header to establish connection
    - client can send the domain name before the encrypted client hello message
  - virtual hosts: same IP, multiple DNS => how does server know which certificate to use?
    - old solution: client-hello includes server domain name
    - TLS1.3 tries to hide domain name for privacy (encrypted certificates)
      - future solution: domain name encrypted under public key provided by DNS server

Integration TLS/HTTP

- Why not use HTTPS for all traffic?
  - some years ago => performance
  - today => AES-NI => HW acceleration => no excuse
- Since 2018 browsers tend to tag HTTP sites as insecure
  - e.g., visual indication, alarm when password is sent, etc.

TLS/HTTPS in browsers

- Debatable whether visual indicators are effective:
  - do we confirm the domain name?
  - do we confirm we have a secure connection?
- What if we click on a link http://www.paypal.com?
  - correct server policy => redirect from http to https
  - but => MiM could connect on our behalf to https://www.paypal.com
  - This is called SSL Strip Attack (modern browsers signal this => does user notice?)
  - solution => possible to push to browser a flag that indicates all future connections to this domain name should use HTTPS (Strict Transport Security) => cleared with cache

TLS/HTTPS in browsers

- Never forget Man in the Middle attacks:
  - Only protection is an authenticated server public key
- Corrupt or hacked CA has catastrophic impact:
  - several examples TurkTrust (2013), Indian NIC (2014), WoSign (2016)
  - signed false certificates for Google, Yahoo, etc.
- Can only be fixed by eliminating certificates from operating system distributions

Privacy

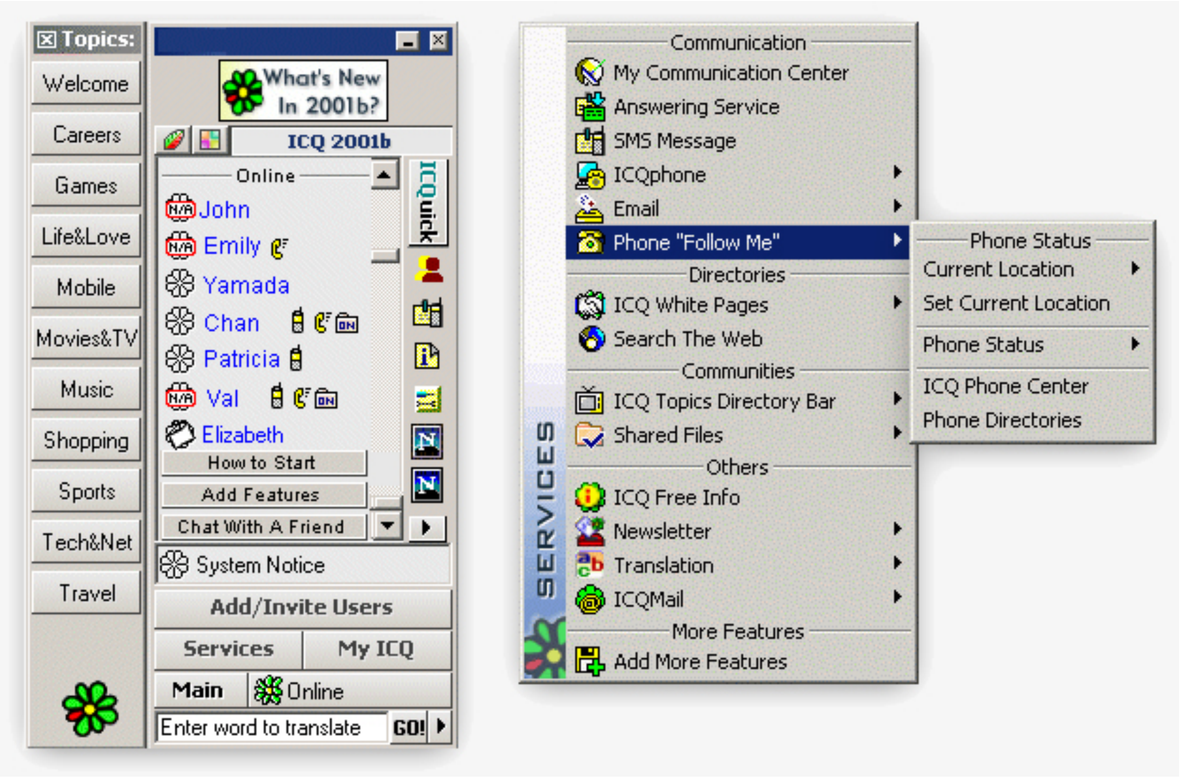
- TLS reveals a lot of meta information (endpoints, size and number of messages, etc.)
- Traffic analysis ofter permits inferring:
  - application with which one is interacting
  - what operations one is doing
- TOR provides some protection, but can be removed by sufficiently powerful attacker

Secure Messaging
(extra: não sai para teste)

Secure Messaging

- Ano 2004:
  - Existiam já mecanismos de comunicação por instant messaging, geralmente centralizados
    - AIM, MSN, ICQ, …
  - Começaram a aparecer os primeiros overlays que pretendiam garantir “segurança” contra adversários externos e o próprio servidor.
  - Esses overlays eram baseados nas tecnologias de chave pública dos anos 90: S/MIME, PGP/GPG.

I Seek You (ICQ)



Off the Record Messaging

- Ano 2004: Off the Record Messaging (OTR)
  - Requisitos de segurança e privacidade inovadores:

"*We argue that **not only must encryption be used to hide the contents of the conversation**, but also, the encryption must provide **perfect forward secrecy** to protect from future compromises.*
*Additionally, **authentication must be used** to ensure that the person on the other end is who they claim to be.*
*However, the **authentication mechanism must offer repudiation**, so that the communications remain personal and unverifiable to third parties.*
Only with these properties can privacy similar to real-world social communications be achieved."

Off the Record Messaging

- Mecanismos clássicos não satisfaziam todos os requisitos:
  - Usam assinaturas digitais para autenticação, o que torna as mensagens não repudiáveis
  - Se não usam assinaturas digitais para autenticação são completamente inseguros contra Man-in-the-Middle
  - Muitas vezes, comprometendo uma chave de longa duração, o adversário consegue recuperar todas as mensagens trocadas (transporte de chaves)

Off the Record Messaging

- Ideias chave para perfect-forward-secrecy (*ratcheting*):
  - DH + Assinaturas para handshake inicial => chaves simétricas iniciais
  - DH re-keying sob autenticação simétrica $k_{ij} = H(g^{x_i y_j})$

$$
\begin{aligned}
A \rightarrow B: &\quad g^{x_1} \\
B \rightarrow A: &\quad g^{y_1} \\
A \rightarrow B: &\quad g^{x_2}, E(M_1, k_{11}) \\
B \rightarrow A: &\quad g^{y_2}, E(M_2, k_{21}) \\
A \rightarrow B: &\quad g^{x_3}, E(M_3, k_{22})
\end{aligned}
$$

Descarta-se logo que possível k_{ij}

Descarta-se logo que possível x, y, g^x e g^y

Off the Record Messaging

- Ideia chave para repúdio:
  - DH + Assinaturas para handshake inicial
  - Alguém que registe todo o trace tem uma prova? => Apenas do handshake
  - A chave de MAC é calculada como $H(K_{enc})$
  - Quando a chave K_{enc} muda, a chave de MAC é divulgada!
  - Porquê => qualquer mensagem autenticada poderia ser feita por qualquer um!

Signal

- Vários protocolos de messaging seguro com “end-to-end-encryption” surgiram nos últimos anos, que protegem contra a “curiosidade” do próprio servidor.

- Os grandes fabricantes de software e fornecedores de serviços adoptaram esse standard de segurança, sob pena de perderem utilizadores.

- O mais proeminente é talvez o Signal, que tem as suas próprias aplicações, e é também adotado pelo Whatshapp e pelo Facebook Messenger.

2E9 utilizadores!!!

Estrutura do Signal

- A comunicação tem de ser possível mesmo com uma das parties off-line
  - Implica utilizar o servidor como buffer e, inicialmente, como canal que autentica utilizadores
- O bootstrap é feito quando um utilizador regista uma identity key (chave de assinatura de longa duração) no servidor
  - autenticação com base no telemóvel

Estrutura do Signal

- O acordo de chaves tem 3 partes
  - Handshake inicial: extended tripple Diffie-Hellman (X3DH)
  - Asymmetric ratchet (quando recebemos DH fresco): recalcula-se chave de sessão com mistura de DH antigo e DH novo
  - Symmetric ratchet (quando não recebemos DH fresco): recalcula-se chave de sessão com base em hashing
- Cada mensagem é protegida com uma chave diferente (perfect forward secrecy)
- Introduz-se um novo objetivo de segurança: post-compromise security, que permite recuperar segurança mesmo se o estado interno do protocolo for revelado.

(Figuras de https://eprint.iacr.org/2016/1013.pdf)

Comunicação Offline

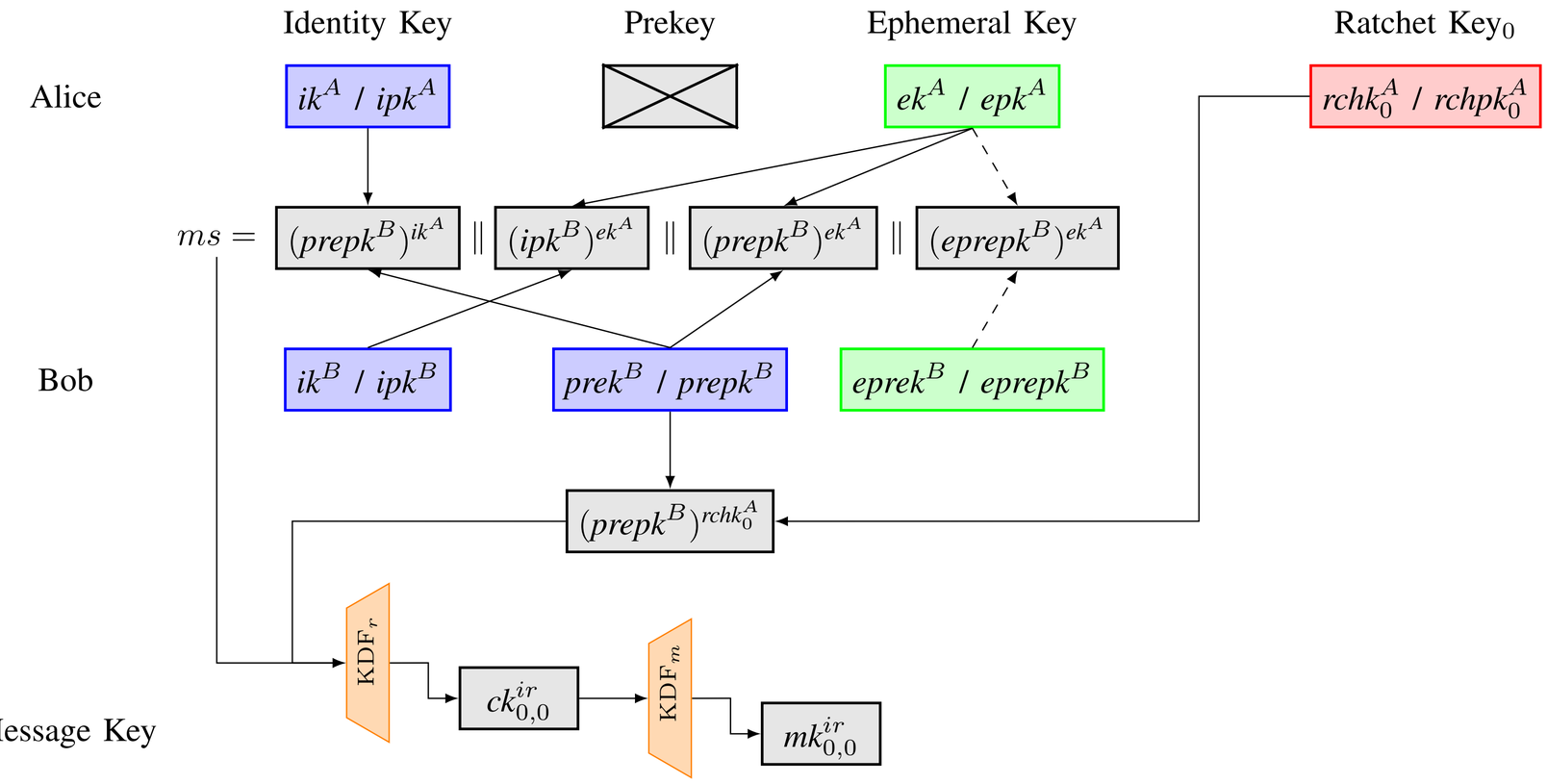
- Para garantir comunicação quando um contacto está offline:
  - todos os contactos submetem regularmente um número significativo de chaves DH efémeras para o servidor.
- Quando queremos enviar uma mensagem a um contacto:
  - podemos pedir uma dessas chaves ao servidor, para termos um elemento fresco para o ratcheting assimétrico ou handshake inicial.

Registo

- Cada recetor regista o seguinte conjunto de chaves públicas:
  - Uma identity key Kid (DH longo prazo + assinatura)
  - Uma signed DH prekey de médio prazo assinada com Kid (partilhadas por todos os emissores)
  - Chaves DH de curto prazo, descartadas pelo servidor uma vez entregues a um emissor
- Em resumo:
  - existem chaves DH de curta e média duração para acautelar ataques MitM localizados no tempo => necessário interceptar ambas
  - existem chaves de assinatura de longa duração => necessário intervir no início para ser possível lançar ataques MiM sobre chaves DH de média duração

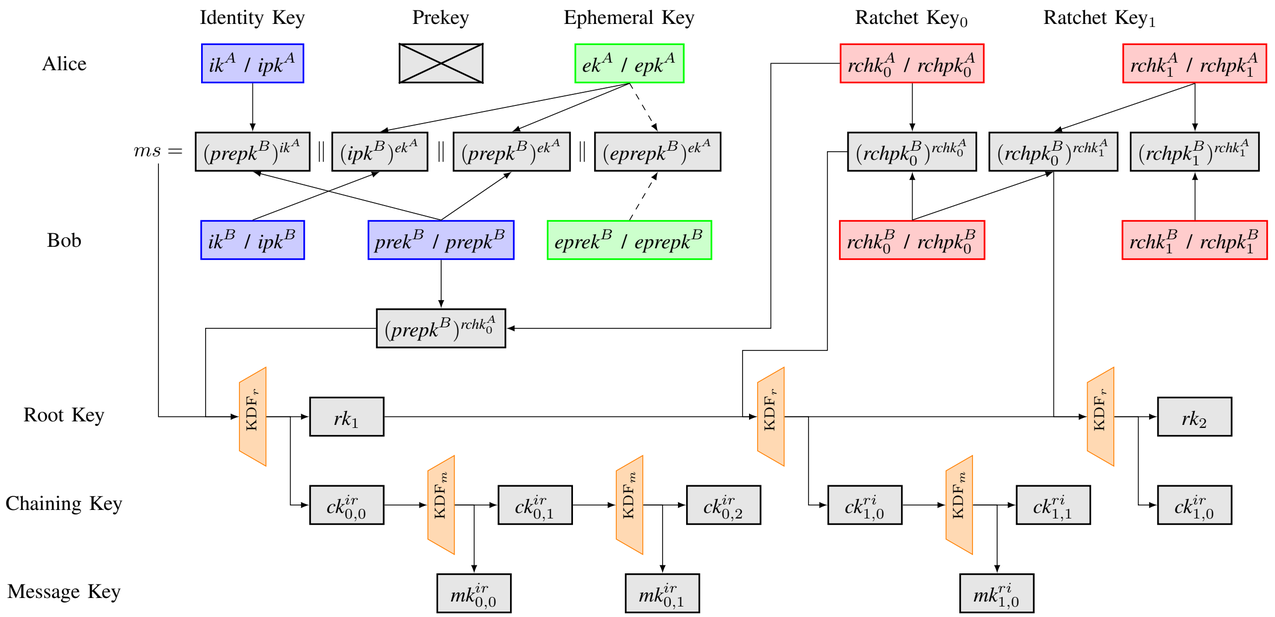
Tripple Handshake

- Perspectiva Alice:
- Mensagem enviada identifica chave efémera utilizada
- Observar como se combinam vários Diffie-Hellman:
  - concatenam-se várias chaves da forma g^{xy}
  - calcula-se o hash (KDFr) => chave de cadeia ck
  - outro hash (KDFm) => chave para uma mensagem mk



Ratcheting

- Depende das mensagens transmitidas e recebidas:
- Quando transmitimos várias mensagens seguidas, usamos ratchet simétrico.
- Quando recebemos um novo valor DH do destinatário, então fazemos ratchet assimétrico.
- Destroem-se as chaves de transmissão (mk) e guarda-se a última chave de cadeia (ck)



- Como gerimos o “esquecimento” do lado do receptor?
- As mensagens podem chegar desordenadas, portanto temos de nos lembrar das mk passadas até recebermos uma mensagem cifrada com essa chave.
- Se avançarmos a chain, todas as mensagens intermédias estão protegidas!

Pressupostos e garantias

- Recebemos o idk inicial por canal autêntico (servidor honest but curious)
- Recebemos DH do primeiro triple handshake por canal autêntico.
- Isto garante confidencialidade e autenticidade do master secret.
- O esquecimento das chaves garante perfect forward secrecy.
- Se houver corrupção total do estado, podemos recuperar segurança (post compromise security) se houver um ratchet assimétrico que o adversário não controla poderemos recuperar segurança (talvez temporariamente)
- Note-se que corrupção total do estado implica que idk já não é autentica, e portanto novas sessões serão vulneráveis a MitM.
- Não tem como objectivo a deniability, mas seria possível usar qualquer coisa como no OTR.