# Certificate chains explanation and hands-on

Setembro 2019

# Agenda

- Contexto Geral
- SSL
- TLS
- Chave publica e privada
- Certificado Digital
- Quem pode gerar um certificado?
- Cadeia de certificados
- Métodos de criptografia
- Hands-on
  - Gerando certificado com CA e Server
  - Gerando certificado com CA, Inter e Server

# SiDi

# ssl, tls e certificados _

# Contexto geral

- Necessidade de trocar informações sensíveis usando canais inseguros
- Modelo OSI não garante segurança

# SSL Context

- Protocolo de criptografia feito para prover comunicação segura através de um canal inseguro.
- Usado por web-browsers, e-mail, voip, etc.
- Versões:
  - 1.0 (nunca foi lançada)
  - 2.0 (deprecada 2011)
  - 3.0 (deprecada 2015)

# TLS Context

- Protocolo de criptografia feito para prover comunicação segura através de um canal inseguro.
- Sucessor do SSL
- Versões:
  - 1.0 (1999)
  - 1.1 (2006)
  - 1.2 (2008) – Mais usada atualmente
  - 1.3 (2018)

# Cifras (cipher suite)

- Cada cifra tem um nome único de identificação e os algoritmos que ela usa.
- Exemplo: TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - **TLS** define o protocolo pra que essa cifra é usada, geralmente é TLS
  - **RSA** indica o algoritmo de troca de chave sendo usado. Esse algoritmo é usado para determinar como o cliente e o server vão se autenticar durante o ‘handshake’
  - **3DES_EDE_CBC** indica o bloco de cifra usado para encriptar as mensagens junto com o modo de operção do bloco
  - **SHA** indica o algoritmo de autenticação que é usado para autenticar a mensagem

# Criptografia

- Origem da criptografia
- Porque é necessária?

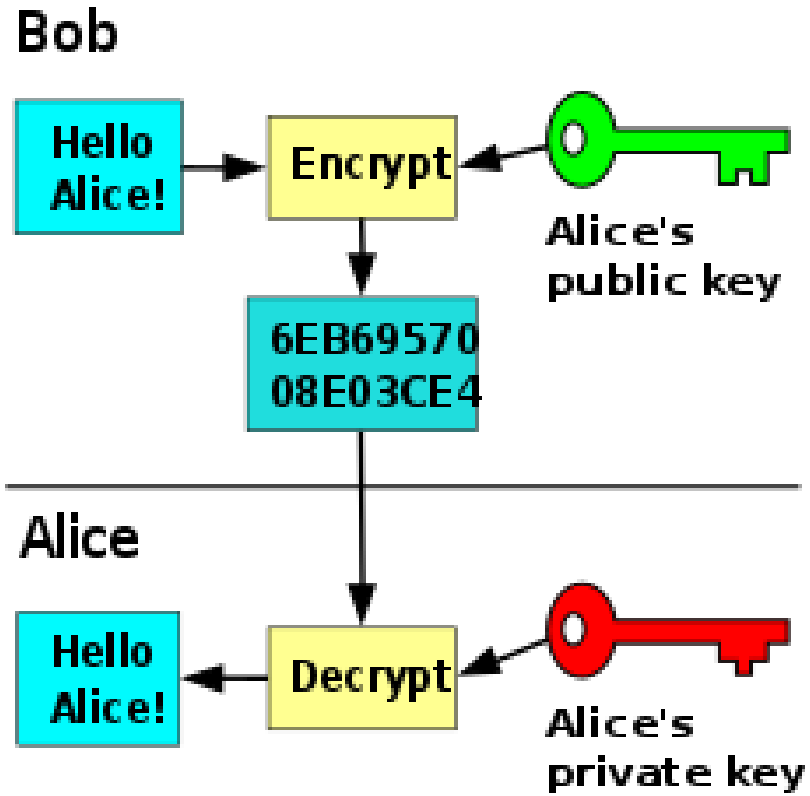
- Usos
  - Confidencialidade
    - Mantém as mensagems criptografadas somente entre os pares
  - Integridade
    - Detecção de mudança do conteúdo da mensagem
  - Autenticidade
    - Quem criou essa mensagem?

# Chave simétrica

- Conceito
- Qual o problema se a chave vazar?

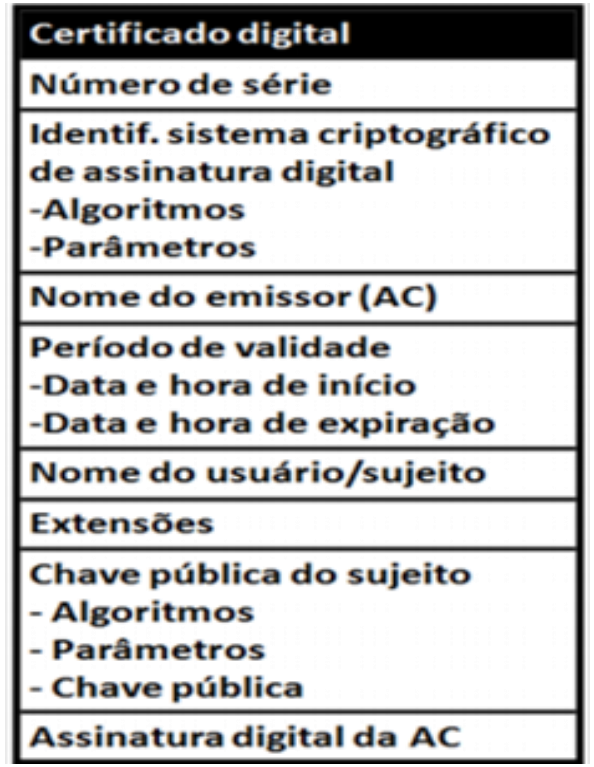
# Private/Public Keys

- Private Key
- Public Key



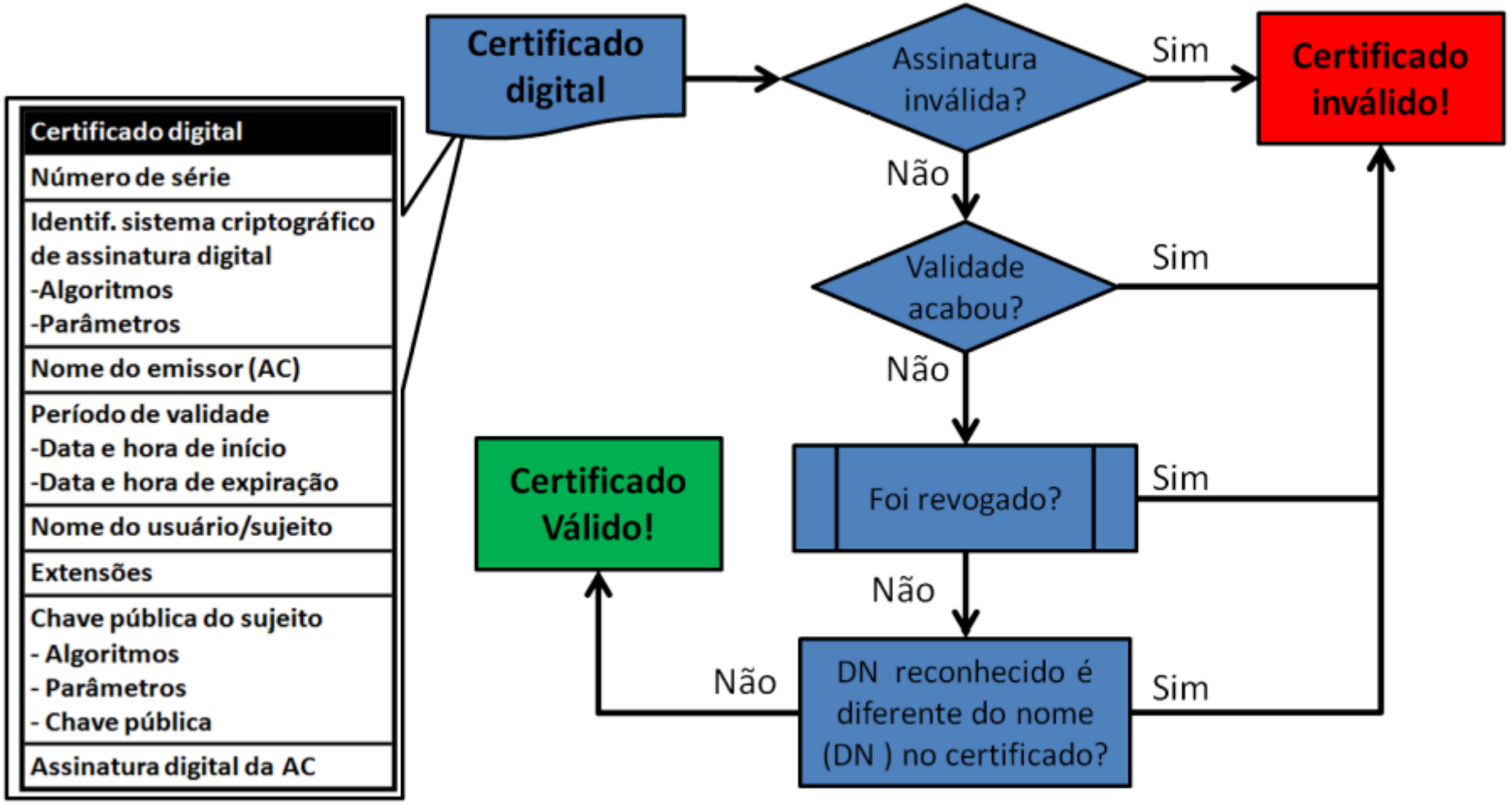
# Certificado Digital

- Para que é usado?
- O que há dentro de um certificado?
- Tipos de ataque man in de middle (roteador mudar o dns)



# Certificado Digital

## Validação



# Quem pode gerar um certificado?

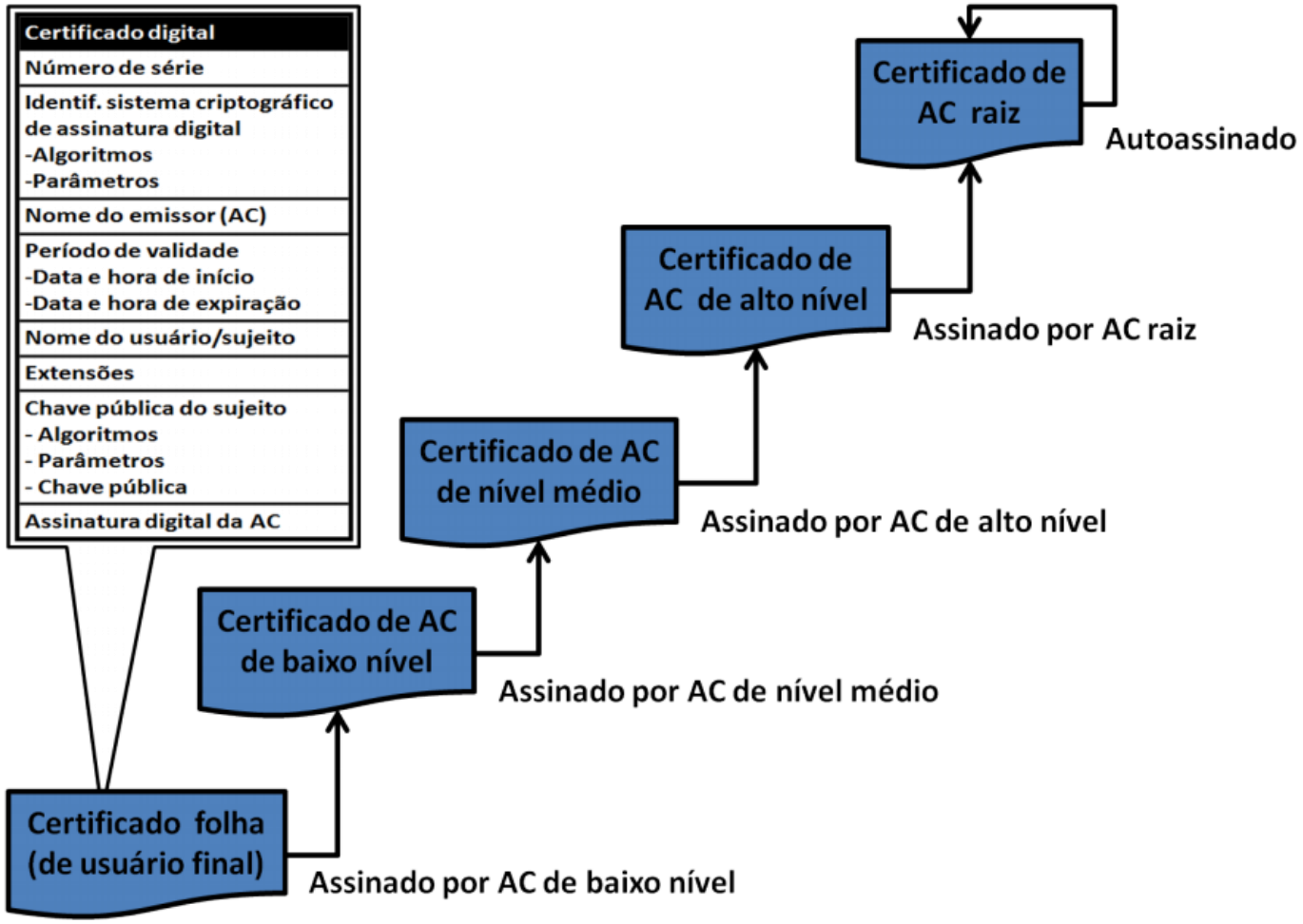
- CAs
- Self Signed

- Casos de revogação de CA
  - https://wiki.mozilla.org/CA:Symantec_Issues

# Tipos de validação de certificado

- DV – Certificado com validação de Domínio
  - Para blogs e sites pessoais
  - Valida a posse do domínio
  - Exibe o cadeado de segurança
- OV – Validação da empresa
  - Voltado para empresas e organizações sem fins lucrativos
  - Valida a posse do domínio e a organização
  - Exibe o cadeado de segurança
- EV – Validação estendida
  - Para sites de comércio eletrônico
  - Valida a posse do domínio e o mais elevado nível de autenticação empresarial
  - Exibe o cadeado de segurança, o nome da empresa e a barra verde

# Cadeia de certificados



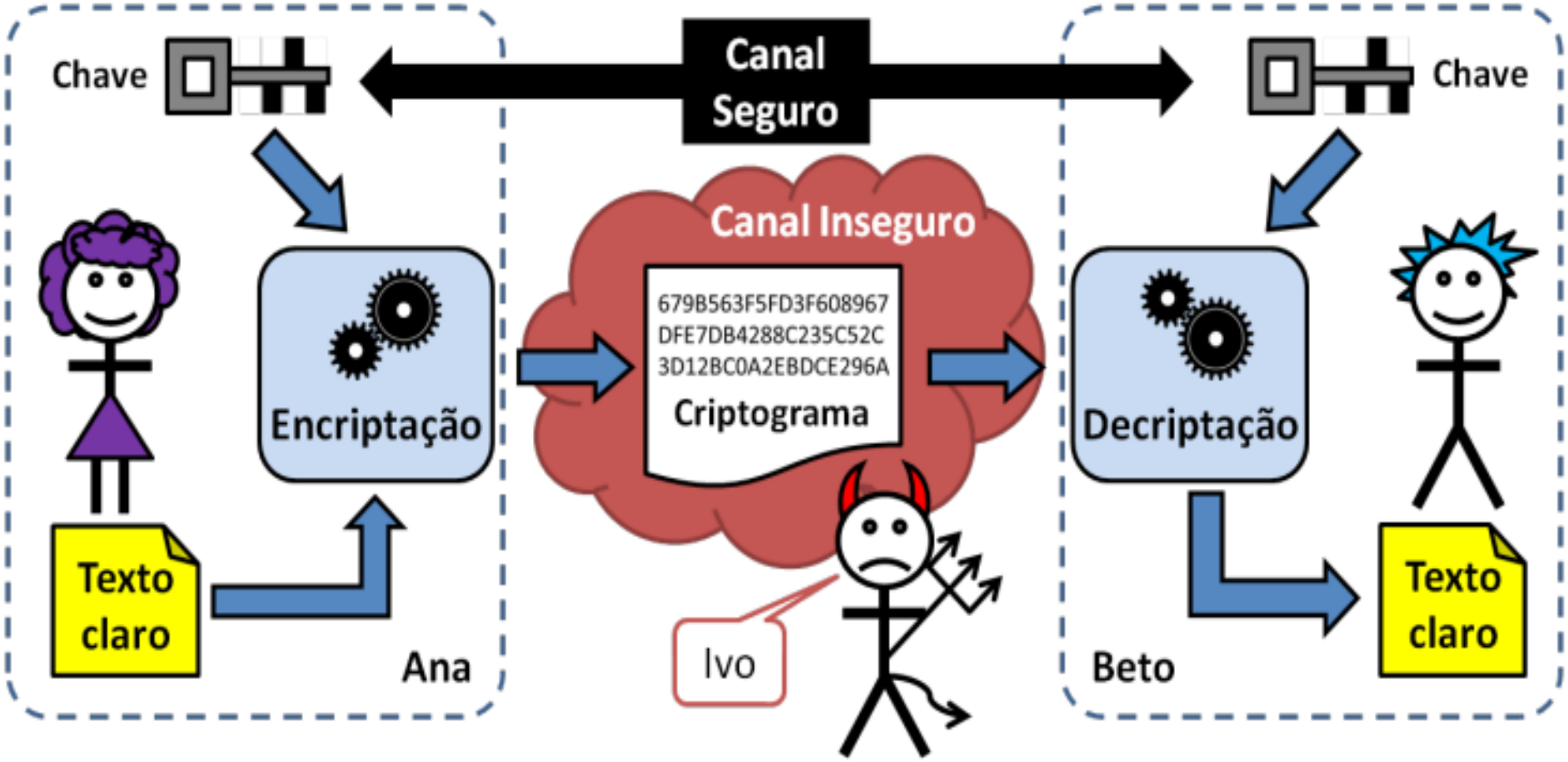
# Key Usage

- Key usage
  - DigitalSignature
  - NonRepudiation
  - KeyEncipherment
  - DataEncipherment
  - KeyAgreement
  - KeyCertSign
  - CRLSign
  - EncipherOnly, DecipherOnly
- Extended Key Usage extension

# métodos de criptografia_

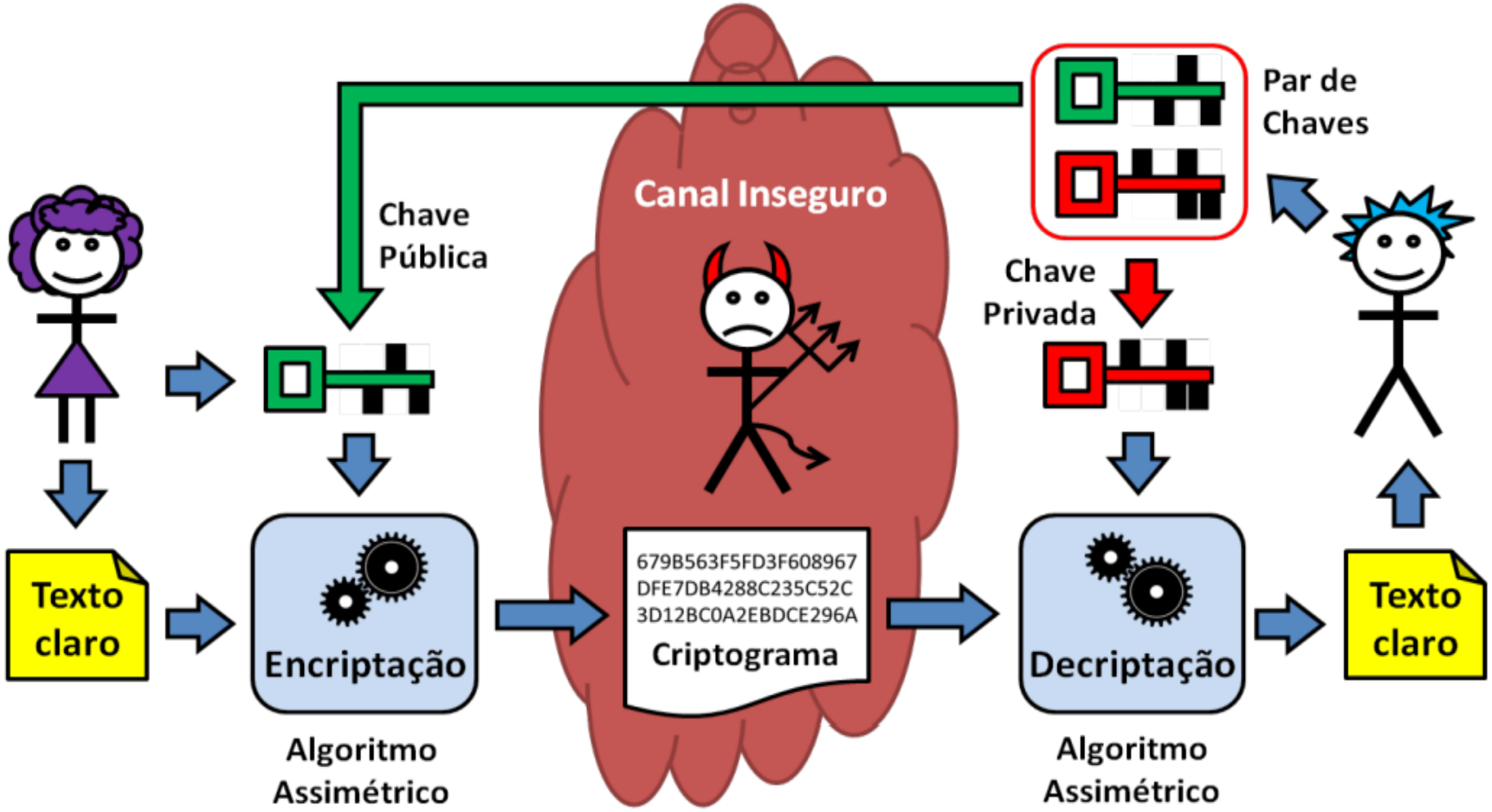
# Métodos de criptografia

## Criptografia simétrica



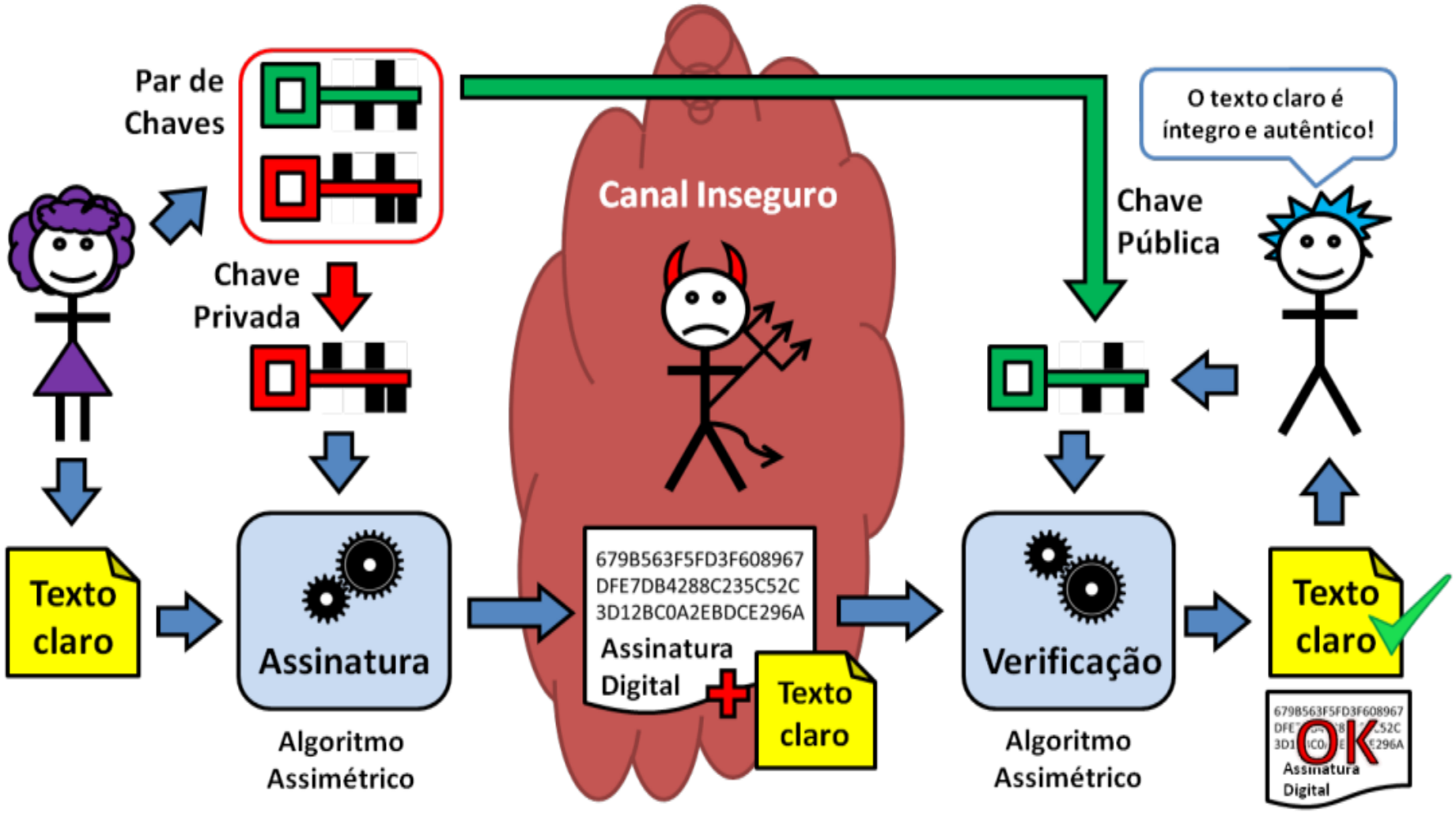
# Métodos de criptografia

Criptografia assimétrica para segredos



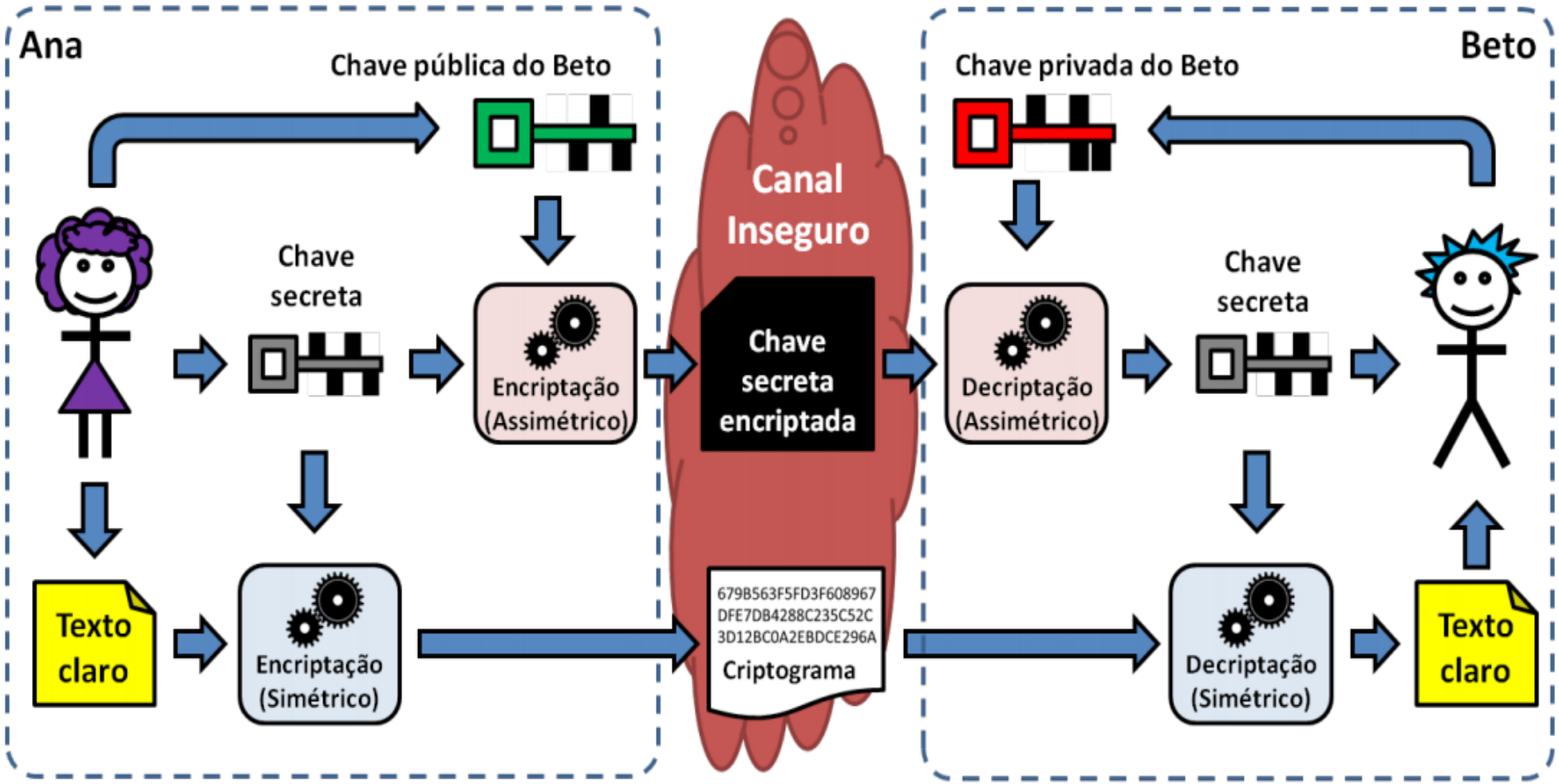
# Métodos de criptografia

## Criptografia assimétrica para autenticidade



# Métodos de criptografia

Criptografia hibrida (simétrica + assimétrica)



# hands on _

# OpenSSL

- OpenSSL é uma implementação de código aberto dos protocolos de SSL e TLS.
- Implementa funções básicas de criptografia para trabalhar com certificados digitais

# Repositório

- https://github.com/felipetortella/ap-pucc-certificate

# Criando seu self-signed certificate

- Criar chave privada
  - openssl genrsa –aes256 –out rootCA.key
    - *aes (advanced encryption standard) 256bits*
- Criar certificado root (CA)
  - openssl req –x509 –new –nodes –key rootCA.key –days 3650 –out rootCA.pem
    - req
    - x509
      - Padrão para infraestruturas de chaves públicas
      - Formato dos certificados digitais
    - days
      - Data de duração do certificado
    - Nodes
      - no DES, openssl não vai encriptar a chave privada no certificado

- Criar chave privada do certificado
  - openssl genrsa –aes256 –out private.key
- Criar CSR (Certificate Signing Request)
  - openssl req –new –key private.key –out private.csr
- Sign CSR with rootCA
  - openssl x509 -req –in private.csr –CA rootCA.pem –CAkey rootCA.key –CAcreateserial –out private.crt –days 3650
- Change certificate to tomcat server be able to read it
  - openssl pkcs12 –export –in private.crt –inkey private.key –out private.p12
- Use java keytool to generate JKS
  - keytool –importkeystore –srckeystore private.p12 –srcstoretype PKCS12 -destkeystore –private.jks –deststoretype JKS

# Hands on com certificado intermediário

## Gerando o certificado intermediário

- Já temos nosso rootCA.pem
- Converter nosso .pem para .crt que vai ser usado no intermediário
  - openssl x509 –outform der –in rootCA.pem –out rootCA.crt
- Gerar o inter.csr
  - openssl req –new –newkey rsa:2048 –sha256 –nodes –out inter.csr –keyout inter.key –config inter.cfg
- Assinar nosso intermediário com o rootCA
  - openssl x509 –req -CA rootCA.pem –CAkey rootCA.key –CAcreateserial -in inter.csr –out inter.crt –extfile inter.cfg –extensions v3_ca
- Gerar o .pem do inter
  - openssl x509 –outform PEM –in inter.crt –out inter.pem

# Hands on com certificado intermediário

Gerando o certificado do servidor com o Intermediário

- Gerar a private key e o csr usando o server.cfg
  - openssl req -new -newkey rsa:2048 -sha256 -nodes -out private.csr -keyout private.key -config server.cfg
- Assinar o .crt com o CA intermediário
  - openssl x509 -req -CA inter.pem -CAkey inter.key -CAcreateserial -days 2880 -in private.csr -out private.crt -extfile server.cfg -extensions v3_req
- Juntar o CA root e o CA intermediário em um arquivo só
  - cat rootCA.pem inter.pem > bundle.pem
- Exportar .p12 com a cadeia inteira de certificados
  - openssl pkcs12 -export -in private.crt -inkey private.key -out private.p12 -chain -CAfile bundle.pem

# Tipos de extensão dos certificados

- At its core an X.509 certificate is a digital document that has been encoded and/or digitally signed according to RFC 5280.
- **Encodings (also used as extensions)**
  - .DER = The DER extension is used for binary DER encoded certificates. These files may also bear the CER or the CRT extension.  Proper English usage would be “I have a DER encoded certificate” not “I have a DER certificate”.
  - .PEM = The PEM extension is used for different types of X.509v3 files which contain ASCII (Base64) armored data prefixed with a “—— BEGIN …” line.
- **Common Extensions**
  - .CRT = The CRT extension is used for certificates. The certificates may be encoded as binary DER or as ASCII PEM. The CER and CRT extensions are nearly synonymous.  Most common among *nix systems
  - CER = alternate form of .crt (Microsoft Convention) You can use MS to convert .crt to .cer (.both DER encoded .cer, or base64[PEM] encoded .cer)  The **.cer** file extension is also recognized by IE as a command to run a MS cryptoAPI command (specifically rundll32.exe cryptext.dll,CryptExtOpenCER) which displays a dialogue for importing and/or viewing certificate contents.
  - .KEY = The KEY extension is used both for public and private PKCS#8 keys. The keys may be encoded as binary DER or as ASCII PEM.

# Self signed para testes rápidos

https://letsencrypt.org/docs/certificates-for-localhost/

```
openssl req -x509 -out localhost.crt -keyout localhost.key -newkey rsa:2048 -nodes -sha256 -subj '/CN=localhost' -extensions EXT -
config <( \ printf "[dn]\nCN=localhost\n[req]\ndistinguished_name =
dn\n[EXT]\nsubjectAltName=DNS:localhost\nkeyUsage=digitalSignature\nextendedKeyUsage=serverAuth")
```

# show certificate on spring boot _

# LINKS _

# Referências


- https://searchsecurity.techtarget.com/definition/digital-certificate
- https://www.comodo.com/resources/small-business/digital-certificates.php
- https://letsencrypt.org/docs/certificates-for-localhost
- TLS v1.2 Protocol Detailed Flow
  - https://tls.ulfheim.net/
- Introdução à criptografia para administradores de sistemas com TLS, OpenSSL e Apache mod_ssl.
  - Alexandre Braga (UNICAMP)
  - Ricardo Dahab (UNICAMP)
- https://www.ibm.com/support/knowledgecenter/en/SSKTMJ_9.0.1/admin/conf_keyusageextensionsandextendedkeyusage_r.html
- https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/nss_tech_notes/nss_tech_note3


# Obrigado,

@segueosidi


- fl.tortella@sidi.org.br
- r.cadaval@sidi.org.br