# Segurança da Informação
# Uma visão da academia
**(ou, mais exatamente, de um acadêmico)**

**Prof. Dr. Marcos A. Simplicio Jr.**
Escola Politécnica da USP

## Motivação para Segurança

- **Segurança** é um **requisito não-funcional**
  - Adicionada a soluções computacionais...
  - ... e acaba seguindo **tendências** no uso de computação:
    - Ex.: Nuvem, Internet das Coisas, Big Data, sistemas inteligentes, mobilidade, descentralização (P2P)
- **Raramente é protagonista**
  - Mas ganha evidência quando **problemas** ocorrem
  - "Efeito Snowden" e casos recorrentes na mídia...

*"A maioria das pessoas não se preocupa com segurança...*
*até que ocorra uma falha de segurança"*

2

# Motivação para Segurança

- Ampla gama de **cenários**, com **necessidades distintas**

  - **Nuvem**: recursos abundantes, necessidade de isolamento de recursos, infraestrutura de terceiros
    - **Banco Inter**: https://www.tecmundo.com.br/seguranca/129811-exclusivo-vazam-dados-400-mil-clientes-banco-inter.htm


  - **IoT**: escassez de recursos, heterogeneidade, grande escala, múltiplos domínios administrativos
    - **Mirai (Botnet)**: http://computerworld.com.br/rede-mirai-ataques-virtuais-dispositivos-de-iot-se-tornam-mais-comuns

  - **Sistemas inteligentes**: grande volume de informações e vários sistemas para processá-las

    - **Cambridge Analytica** (renomeada "**Emerdata**"?) https://tecnologia.uol.com.br/noticias/redacao/2018/04/09/veja-como-saber-se-a-cambridge-analytica-roubou-seus-dados-do-facebook.htm

  - E **muitas outras**...

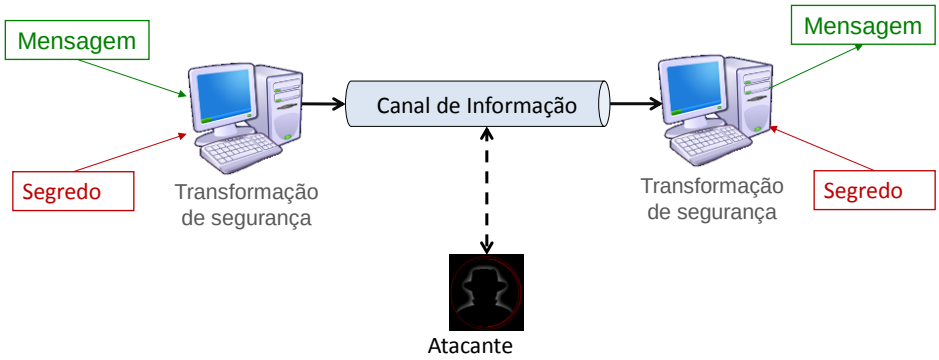


# Pesquisa em Segurança

- **Papel da pesquisa em segurança**:
  - **Antever problemas**: mais efetiva se integrada a arquitetura durante fase de **especificação**
  - **Propor soluções** adequadas
- Visão geral da área: duas visões
  - Ponto de vista do **problema**: modelos de segurança (de redes e de computadores)
  - Ponto de vista de **soluções**: voltadas a cenários mais gerais ou mais específicos

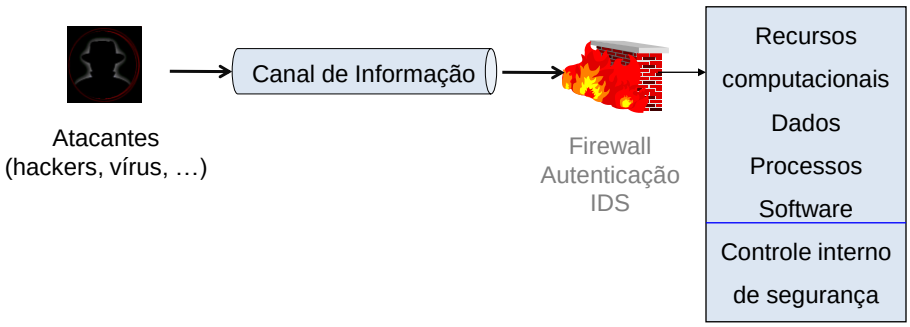
# Modelo de Segurança de Redes

(visão do problema)

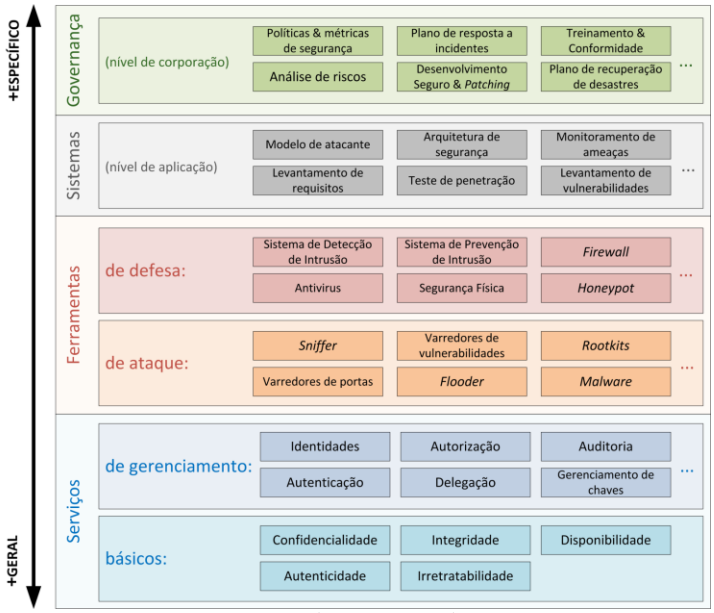



# Modelo de Segurança de Computadores

(visão do problema)



# Soluções em Segurança da Informação

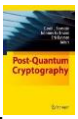


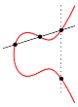

*Baseado em (Stallings, Brown, 2015)*



# LARC: Pesquisa em Segurança

- Criptografia (aplicada): projeto & análise
  - **Primitivas**



    - **Ex.: algoritmos leves** para assinatura e cifração**;** criptografia **pós-quântica; password-hashing; cifração com preservação de propriedades** (e.g., esquemas parcialmente homomórficos)...

  - **Protocolos**

    - Ex.: **mobile-banking**; **e-cash; acordo de chaves** autenticado; **deteção de trapaças** em jogos online; **comunicações veiculares;** ...
  - **Implementação eficiente**

    - Ex.: projeto e otimizações orientados a hardware **hardware-oriented**; resistência a **canais colaterais**; ...

# LARC: Pesquisa em Segurança

- Segurança de rede

  - **Sistemas distribuídos**

    - Ex.: sistemas **P2P** (e.g., blockchain); redes virtuais em ambientes de **nuvem**; coleta de dados de **saúde móvel**; detecção em tempo real de **fraudes bancárias;**...

  - Sistemas com **recursos limitados**

    - Ex.: redes de **sensores sem fios**, **redes veiculares,** e qualquer outra parte da "**Internet das Coisas"** com disponibilidade limitada de **processamento, memória e energia**...

Soluções gerais & específicas

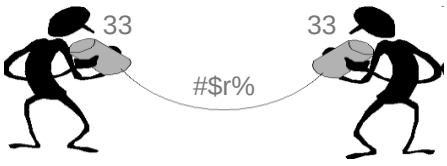
## ALGUNS EXEMPLOS

# Criptografia: pra que serve?

- Utilizando-se de técnicas criptográficas, pode-se:
  - proteger *informações armazenadas* no sistema até mesmo de pessoas que tenham acesso autorizado ao sistema.
  - proteger *informações* que estejam *em trânsito* de um sistema para outro.
  - detectar *alterações acidentais ou intencionais* nos dados.
  - verificar a *autoria* de um documento.

# Confidencialidade

- **Confidencialidade de dados**: garantia de que qualquer *informação* armazenada num sistema de computação ou transmitida via rede seja *revelada somente a usuários devidamente autorizados*.
- Observação: *informação* ≠ *dado* (representação da informação).
  - Um dado pode estar acessível a qualquer entidade e mesmo assim não revelar a informação que ele contém.



# Confidencialidade

- **Privacidade**: Garantia de que os indivíduos *controlem* ou influenciem quais *informações sobre eles* podem ser coletadas e armazenadas e *por quem* e *para quem* tais informações podem ser reveladas
  - Tem relação direta com **confidencialidade de dados** (proteção da informação), mas também envolve políticas de **uso de dados** e **confidencialidade de identidades**.

# Integridade

- Possibilidade de *verificar a consistência* da informação contida nos dados, *impedindo que seja alterada* indevidamente de *maneira imperceptível*.
- Detalhe: o serviço de integridade *não* garante que os dados não sejam alterados. A garantia efetiva é que, se os dados forem alterados sem autorização, a alteração será sempre *detectada*.

# Autenticidade

- Garantia de que a *origem* ou o *originador* de uma mensagem seja *corretamente identificado* pelo seu destinatário.
- A *verificação de autenticidade* é necessária após todo processo de identificação, seja de um usuário para um sistema, de um sistema para o usuário ou de um sistema para outro sistema.

# Irretratabilidade

- O *originador* e o *destinatário* das informações *não podem negar a sua transmissão, recepção ou posse*.
- Relacionado a *assinaturas digitais*.
  - Conceito similar a assinaturas manuais, mas com garantias matemáticas...

# Criptografia: algoritmos

- Duas famílias principais:
  - **Simétricos**: usam a mesma informação secreta (chave) conhecida apenas por remetente e destinatário, mas não por atacantes
    - Esta categoria também inclui algoritmos auxiliares, que não usam chaves secretas
  - **Assimétricos**: usam duas chaves distintas, relacionadas matematicamente. Uma chave é tornada pública (inclusive para atacantes), e a outra é conhecida apenas pelo seu dono.
- Em geral, parte mais forte do sistema:
  - Descobrir uma chave secreta/privada requer teste de ~$2^{128}$ chaves ➔ custo inviável!

# Criptografia: exemplo de ataque

- Exemplo de complexidade de ataques:
  - Suponha que ataque a algoritmo envolva testar chaves de 128 bits: $2^{128}$ possibilidades
  - Suponha também que estejam disponíveis 1 milhão ($10^6$) de super-computadores, cada um capaz de realizar 1 peta ($10^{15}$) testes por segundo
  - Ainda assim seriam necessários ~$2^{58}$ segundos para recuperar a chave...
    - Idade estimada do universo: ~$2^{59}$ segundos

$$\frac{\#\text{testes}}{\#\text{computadores x capacidade}} = \frac{2^{128}}{10^6 \times 10^{15}} \approx \frac{2^{128}}{2^{20} \times 2^{50}} = 2^{58} \text{ segundos}$$

# Computadores quânticos

- Foco de intensa pesquisa nos últimos anos
- Ameaçam criptografia clássica (qubits suficientes):
  - **Grover**: busca exaustiva ➔ algoritmos simétricos
  - **Shor**: fatoração e logaritmo discreto ➔ RSA, DH, ECC

| Algoritmo | Tipo | Seg. Clássica | Seg. Quântica | Ataque |
|---|---|---|---|---|
| AES-N | cifra | N | N/2 | Grover |
| SHA-N | hash | N (pré-imagem), N/2 (colisão) | N/2, N/3 | Grover |
| RSA-2048, RSA-3072 | Assinatura, cifração | 112, 128 | quebrado | Shor |
| ECDH-N, | Acordo de chaves | N/2 | quebrado | Shor |
| ECDSA-N | Assinatura | N/2 | quebrado | Shor |

Fontes: [NW17], [BL17]

19

# Computadores quânticos

- Tamanho necessário para ataques
  - Quebra de **RSA-n** requer ao menos **2n qubits**, mais bits adicionais para **correção de erros** [BL17]
    - Estimativa [BL17]: alguns milhões de qubits
  - Quebra de **ECC-n** requer ao menos **6n qubits** [PZ03]

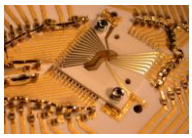
| Segurança | RSA | | ECC | |
|---|---|---|---|---|
| (bits) | Chaves (bits) | qubits | Chaves (bits) | qubits |
| 80 | 1024 | 2048 | 163 | 1000 |
| 112 | 2048 | 4096 | 224 | 1300 |
| 128 | 3072 | 6144 | 256 | 1500 |
| 192 | 4096 | 8192 | 383 | 2300 |
| 256 | 15360 | 30720 | 512 | 3000 |

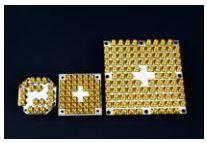
20

# Computadores quânticos

- Onde estamos: ainda longe…
  - D-WAVE: vende produtos com **milhares de qubits**
    - **Mas difícil criar computador quântico**: difícil provar vantagens com relação a computadores clássicos…
    - Ideia básica: temperatura dos qubits = instância do problema; qubits "esfriam" até valor da solução
  - 2017: John Martinis (Google) anuncia capacidade construir chip de **49 qubits** [NIST18]
  - 2018: Intel testa chip de **49 qubits** [NIST18]

*Blakestad (NIST)*





*Walden Kirsch (Intel)*

21

# Criptografia pós-quântica

- Não se conhecem algoritmos quânticos eficientes para **problema subjacente**
  - Abordagens: maioria antigas, outras novas
- **Desafios**: eficiência e análise de segurança
  - Teórica (especificação) e prática (implementação)
  - **Competição em andamento** [NIST18c]

| Algoritmo | Tipo | Uso | Chave pública (bytes) | Dados (bytes) |
|---|---|---|---|---|
| XMSS | Hash (stateful) | Assinatura | 64 | 2.5K |
| SPHINCS | Hash (stateless) | | 1056 | 41K |
| HFEv-* | Sist. Multivariados | | 500K – 1M | 32 |
| McEliece | Códigos | Cifração | 1M | 190 |
| NTRUEncrypt | Reticulados | | 1.5K – 2K | 1.5K – 2K |
| NewHope | Reticulados | Acordo de chaves | -- | 2K |
| SIDH | Isogenias | | -- | 564 |

Fonte: [BL17]

22

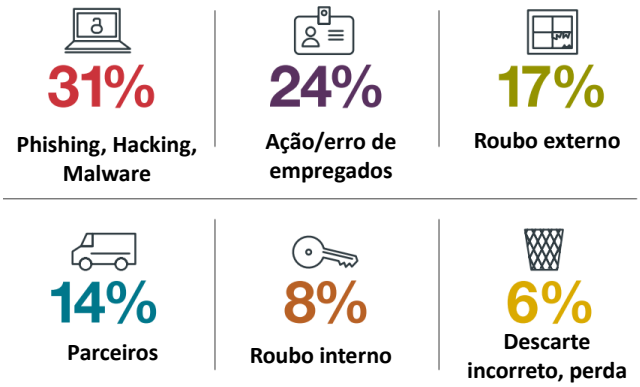
# Cybersegurança vs. Humanos

- Humano: o elo mais fraco de qualquer sistema


23

# Cybersegurança vs. Humanos

- Incidentes de segurança (2016)
  - Obs.: ataques externos com frequência têm origem em problemas internos (e.g., roubo de credenciais)




Fonte: [BH16]

24

# Cybersegurança vs. Humanos

- Constribuições da pesquisa em segurança:
  - **Treinamento** & **ferramentas automáticas**
- Alguns exemplos:
  - Geração e proteção de senhas
  - Identificação de phishing
  - Software para cifração de dados.

25

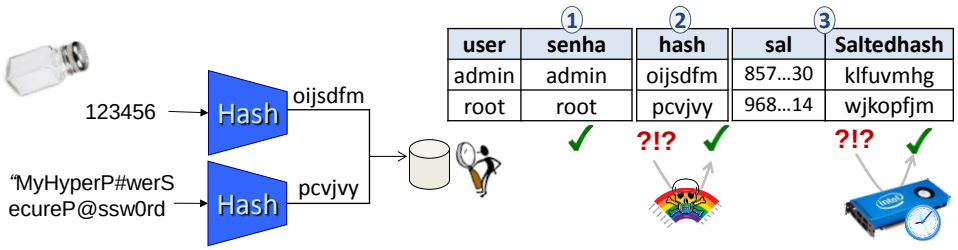
# Proteção de senhas


- Mesmo em cenário com **boa política de segurança**…
  - Senhas não são enviadas às **claras pela rede**; ausência de **keyloggers** no sistema; senhas **alfanuméricas**
- … ainda é possível quebrar senhas por meio de **força bruta:** ~40 bits de entropia em média (Florencio and C. Herley, 2007)
  - **Online**: diversos testes junto a servidor
  - **Offline**: após roubar base de dados/dispositivo
- Proteção:
  - **Online**: bloqueio (temporário) de usuário, CAPTCHAs
  - **Offline**: elevar **custo computational** de cada teste
    ➔ **evita execução de vários testes em paralelo**

26

# Proteção de senhas

- Dispositivo/base de dados **armazena**:
  1. Senha **às claras** ➔ custo do ataque = **zero**...
  2. **Hash (senha)** ➔ custo do ataque = **download de tabela** pré-computada (e.g., rainbow table) ou **serviço web gratuito**...
  3. **Hash (senha, sal)** ➔ custo do ataque = **1 hash/teste**
     - Poucos **us** em PCs modenos; pode ser feito em **paralelo**
     - Clustes de GPUs: >$10^{12}$ hashes/h ➔ quebra de senhas alfanuméricas de **8-caracteres em** 5.5h... (https://securityledger.com/2012/12/new-25-gpu-monster-devours-passwords-in-seconds/)






# Proteção de senhas

- **Password hashing** (com sal)
  - **Custos configuráveis**: *t* segundos com *m* megabytes de RAM; aumento expressivo de tempo se ataque usar menos memória
  - Configuração ideal: custo **imperceptível para usuário legítimo**, mas **relevante para atacantes realizando múltiplos testes**
    - Ex.: *t* = 1s, *m* = 1GiB para autenticação local (ou remota se execução puder ser transferida para clientes)
    - Ex.: *t* = 100 ms, *m* = 20 MiB para autenticação no lado do servidor
      - ~1000 autenticações/s ➔ uso de ~2 GiB

*Configurável

| | 1 núcleo de processamento | | 1000 núcleos de processamento | |
|---|---|---|---|---|
| | testes/s | Memória total | testes/s | Memória total |
| 1 hash | > 1000 | < 1 KiB | > 1.000.000 | poucos KiB |
| PBKDF/bcrypt | 1 | < 1 KiB | 1000 | poucos KiB |
| Lyra2/Argon2 | 1 | 1 GiB | 8 | 8 GiB |

**Limita paralelismo de clusters (ex.: de GPUs)**

# Proteção de senhas

- **Geração de senhas**
  - **Usuário**: frase fácil de lembrar, difícil de adivinhar.
    - Ex.: “Esta era uma boa senha… até eu mostrá-la em sala de aula” ➔ “Ee1bs…Aemesda :p”
    - Ex.: “Quem ri por último, é porque não entendeu a piada”: “Qrp’u, ‘epneap.”
  - **Sistema**: políticas adequadas
    - **Não limitar tamanho** máximo: PHS(senha, sal) tem tamanho fixo…
    - **Não limitar caracteres especiais**: PHS(senha, sal) pode ser convertido para Base64 sem riscos a banco de dados
    - **Não exigir troca** frequente de senhas [NIST18b]: leva a anotações e substituições simples (e.g., “senha+mês”)
    - **Usar crackers** para testar força de senhas no sistema

29

# Cybersegurança vs. Humanos

- **Identificação de phishing**
  - De ataques mais simples…

“Detectamos uma atividade suspeita na sua conta: Compra no valor de R$2500,00 em Lojas Americanas. Acesse http://www.bradesco.com.br/ se não reconhecer essa transação e desejar bloqueá-la.”

**Usar barra de endereços/histórico**

  - … aos mais complexos e efetivos.

“Você foi sorteado para testar o novo iPhone 171. Para confirmar, acesse sua conta e informe seu Apple ID: https://www.apple.com/”

“Detectamos uma tentativa de acesso a sua conta a partir do IP 95.31.18.11 (Moscou/Rússia). Caso não reconheça esse acesso, bloqueie-o respondendo a esta mensagem com o código de 6 dígitos que lhe será enviado em instantes. Equipe GMail.”

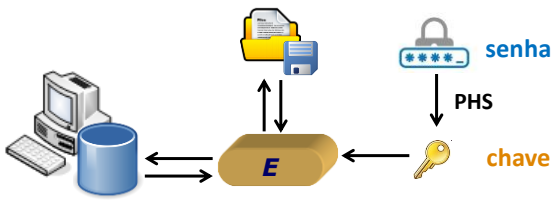
**Entender mecanismos de segurança**

**Recomendado**: https://cartilha.cert.br/

30

# Cybersegurança vs. Humanos

- Uso de software para cifração de dados: proteção contra roubo/apreensão de dispositivos
  - Ex.: Truecrypt, Veracrypt,
    - Suportam volumes ocultos: “plausible deniability”
  - Ex. (nuvem) Cryptsync




31

# Soluções de propósito específico

- Engenharia de segurança: proposta e análise
  - “Quando o TLS não basta”
- Principais desafios

  - **Modelagem**: definição de **requisitos realistas** e razoavelmente **completos**

  - **Segurança**: coerência com **modelo de ameaças** e uso de algoritmos/protocolos adequados; **fatores humanos**

  - **Eficiência**: **recursos** computacionais

  - **Escalabilidade**: **número de nós** e considerações sobre **múltiplos domínios** administrativos
- Exemplo: **Jogos P2P**

32

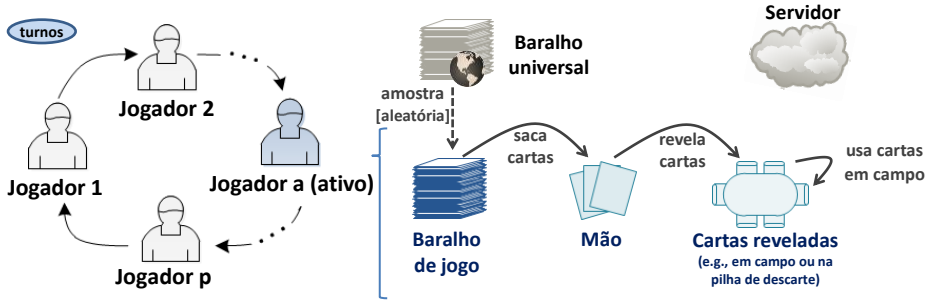
# Exemplo: Jogos digitais

- Contexto específico:
  - Jogos de cartas colecionáveis (TCG) **peer-to-peer**
  - **Arquitetura de jogo distribuída** (sem intervenção de servidor confiável): menos custo, mais escalabilidade
- Considerações práticas:
  - Alguns trabalhos de IC/TCC e mestrado
  - Mercado da ordem de **U$100 Bi anuais**; crescente participação no **tráfego da Internet** (2% em 2015)
  - Costuma atrair estudantes ☺

33

# Exemplo: Jogos digitais

- SecureTCG: cenário
  - Baralho de jogo: 1 por jogador, personalizado
    - Possivelmente após amostragem aleatória do conjunto de cartas existentes
    - Permanece confidencial até o momento da partida

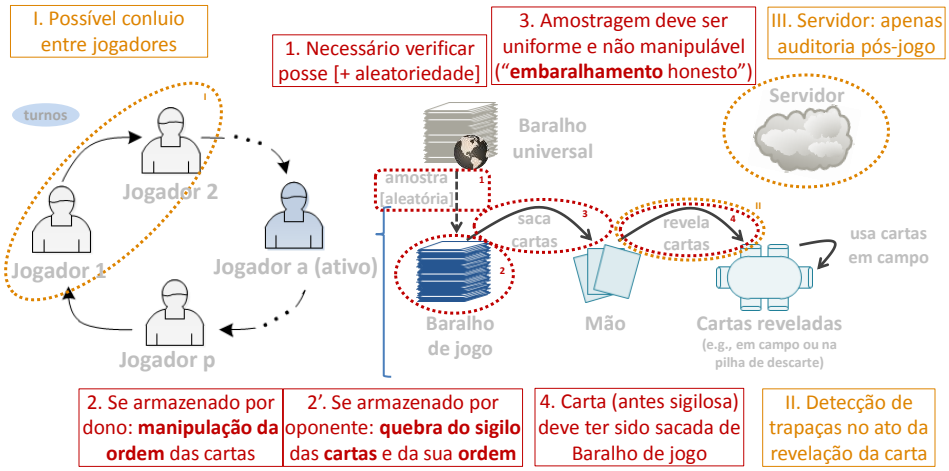



  - Servidor não interfere na partida (P2P), mas pode auditá-la a posteriori

34

# Exemplo: Jogos digitais

- SecureTCG: requisitos/ameaças (<u>trapaças</u>)



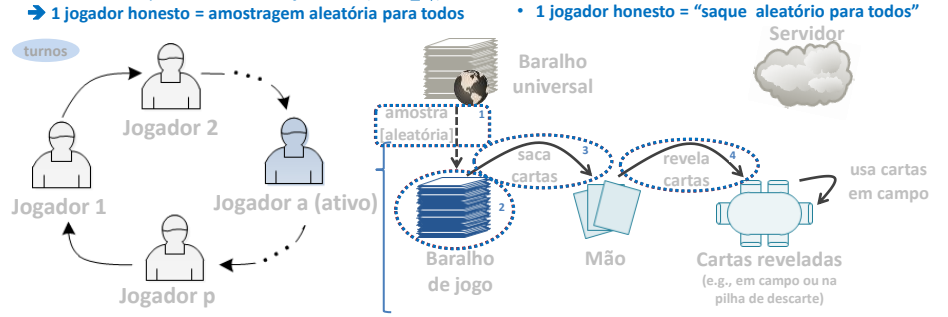

35

# Exemplo: Jogos digitais

- SecureTCG: soluções

1. Amostragem aleatória: função pseudoaleatória
   - Semente: XOR de contribuições de todos os jogadores $P_i$
   - *Commitment* prévio da contribuição: Hash( seed_$P_i$ )
   - ➔ **1 jogador honesto = amostragem aleatória para todos**

3. Índice aleatório: hash chain
   - XOR de contribuições de links de todos os jogadores
   - 1 link revelado por saque: próximo índice imprevisível
   - *Commitment* da chain ➔ impede manipulação
   - **1 jogador honesto = "saque aleatório para todos"**




2. Baralho armazenado por dono
   - *Commitment* probabilístico dos valores e ordem: Hash (rand, ID_carta) ➔ **impede manipulação**

4. **Consistência**: verificar se carta revelada
   a. **Estava no baralho**: *commitment* das cartas
   b. **Foi sacada**: *commitment* dos índices

Não envolve servidor, previne conluio e detecta trapaça quando carta é revelada

36

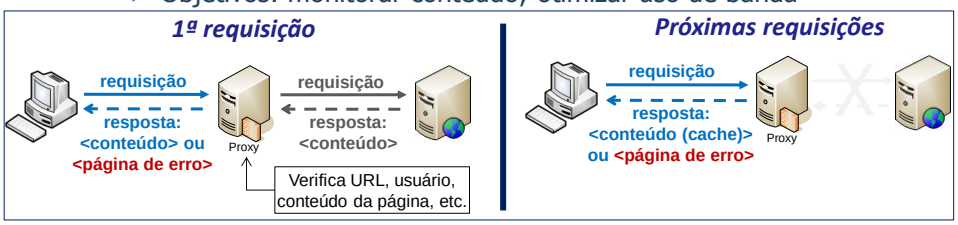
Privacidade na (Deep) Web

# APÊNDICE

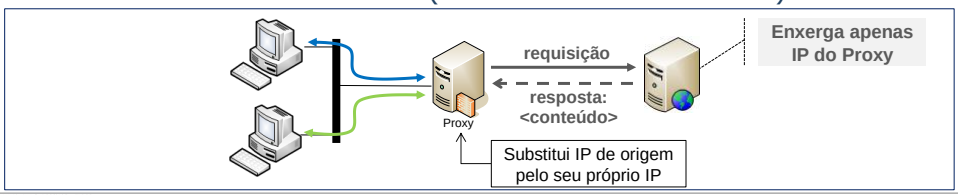


## Proxies e privacidade

- Empresas costumam usar **proxies**
  - Objetivos: monitorar conteúdo; otimizar uso de banda




  - Mas também melhoram privacidade da navegação na Internet devido a NAT (Network Address Translation)

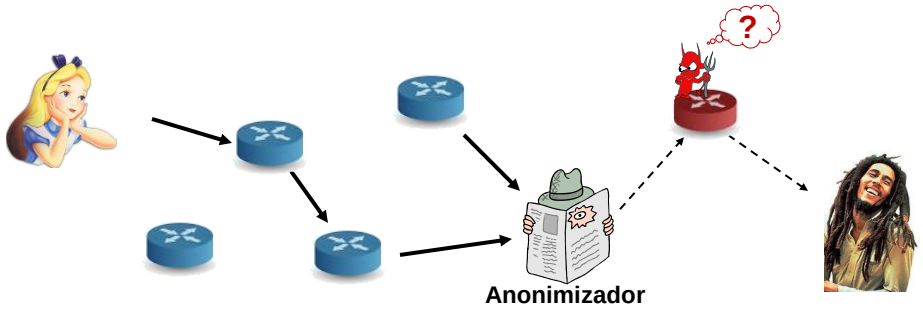


## Proxies e privacidade

- Proxies web: fazem um serviço semelhante
  1. Proxy acessa página pedida, com IP do próprio Proxy
  2. Página acessada responde normalmente (preferências, como linguagem, são aquelas configuradas pelo Proxy)
  3. Proxy entrega resultado para usuário dentro de seu próprio site (conteúdo HTML do site gerado dinamicamente)
- Conexões normalmente são cifradas (TLS)
  - Podem também incluir recursos extras, como esteganografia
  - Exemplo: https://hide.me/en/proxy
- Vamos generalizar (e melhorar!) a ideia
  - Afinal, o Proxy pode registrar requisições, ou ser tirado do ar

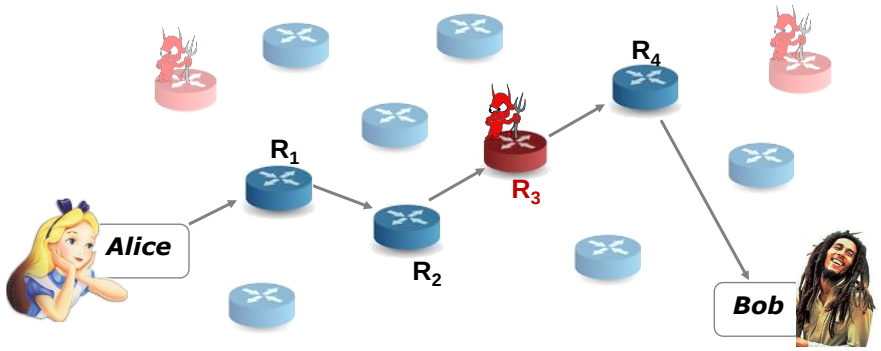


## Generalização: roteamento aleatorizado




- Disfarça origem das mensagens fazendo roteamento aleatório
  - Técnicas populares: Crowds, Freenet, Onion routing
  - Roteadores não têm certeza se a origem aparente de uma mensagem é de fato seu originador ou outro roteador

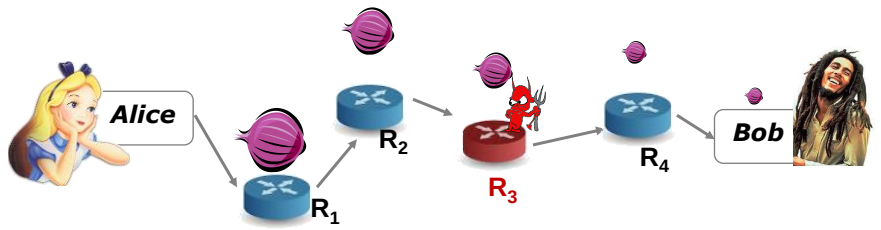
## Onion Routing




- Origem escolhe uma sequência aleatória de roteadores
  - Alguns roteadores são honestos, outros são controlados por atacantes
  - Origem decide a comprimento do caminho



## Onion Routing




cria = $\{R_2, \{R_3, \{R_4, \{B, \{M\}_{pk(B)}\}_{pk(R4)}\}_{pk(R3)}\}_{pk(R2)}\}_{pk(R1)}$

***Info de roteamento de cada link é cifrada com a chave pública (pk) do roteador***

***Cada roteador descobre apenas a identidade do próximo roteador***

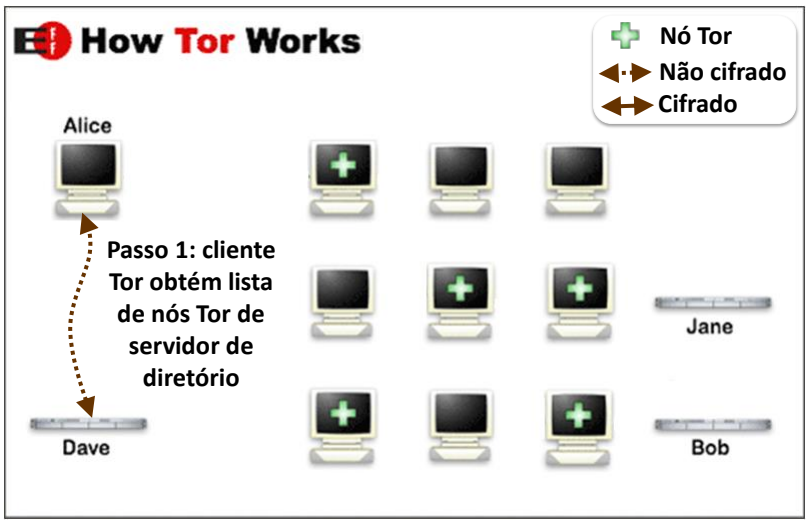
***Apenas destinatário acessa mensagem M***

# Tor

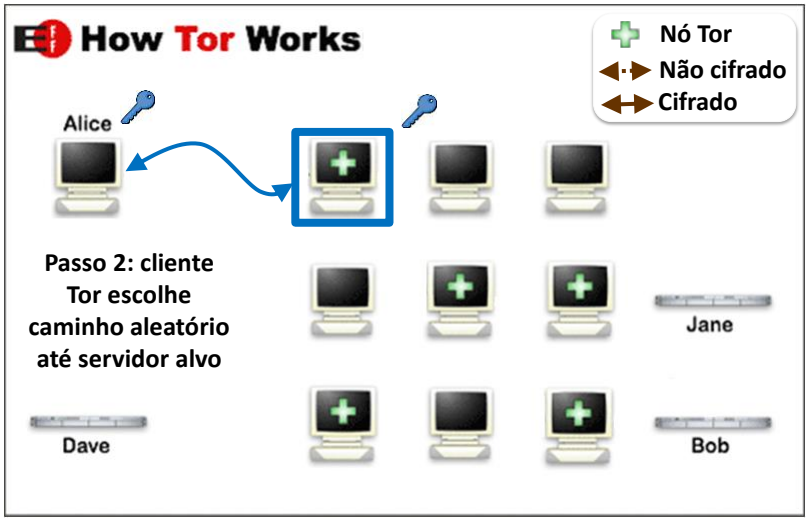
- Segunda geração do onion routing
  - http://tor.eff.org
  - Desenvolvido by Roger Dingledine, Nick Mathewson and Paul Syverson
  - Projetado especificamente para comunicações na Internet que requerem baixa latência
- Ativo desde Outubro de 2003
  - Diversos nós espalhados pelo mundo todo
  - Milhares de usuários
  - Clientes de "fácil uso" (plugins, Tor Browser)
  - Navegação anônima e gratuita



# Tor: resumo



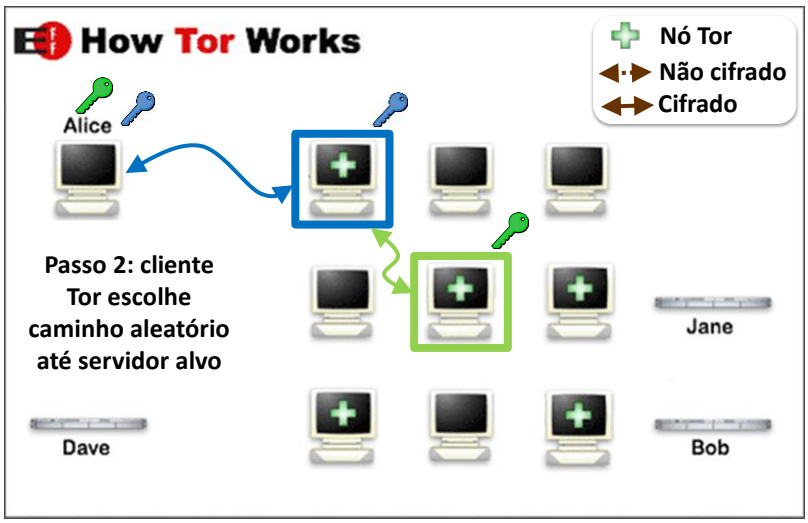
## Tor: resumo




Chave simétrica de sessão estabelecida entre cliente Onion Router #1: circuito inicial

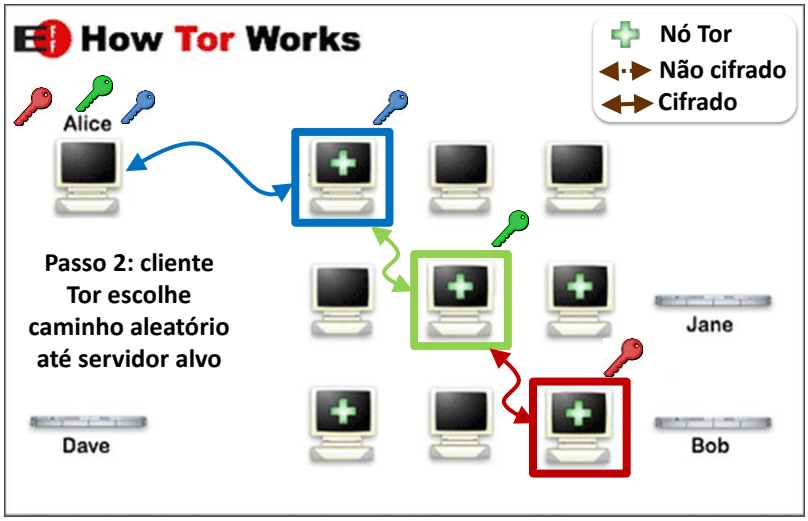


## Tor: resumo




Cliente estende circuito, estabelecendo nova chave simétrica de sessão com Onion Router #2 (tunelamento via onion router #1)

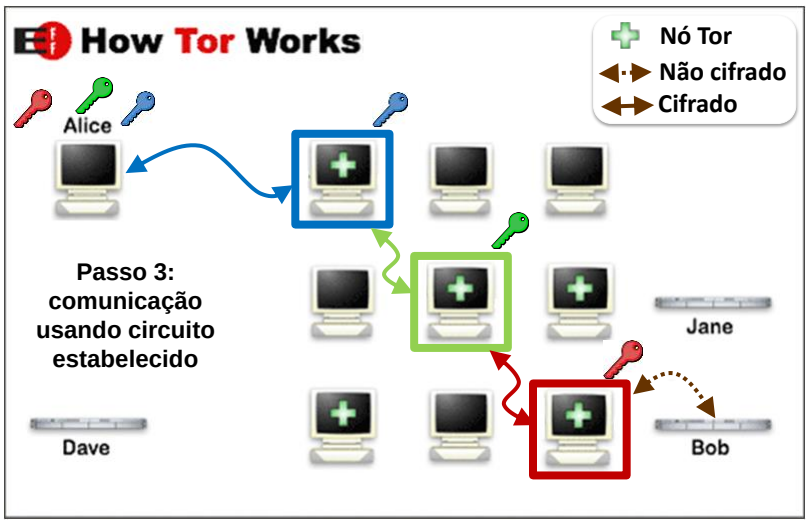
## Tor: resumo




Cliente estende circuito, estabelecendo nova chave simétrica de sessão com Onion Router #3 (tunelamento via onion routers #1 e #2)

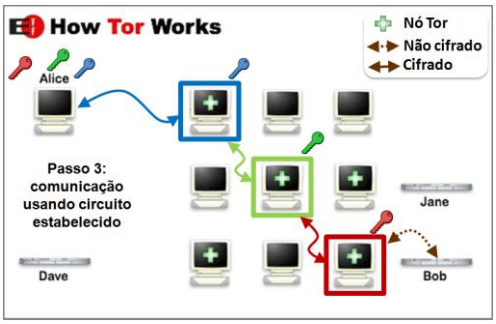


## Tor: resumo

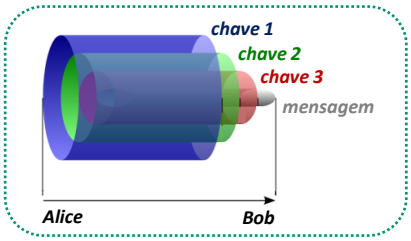



Cliente acessa destino final via circuito estabelecido: destino vê apenas Onion Router #3

## Tor: resumo

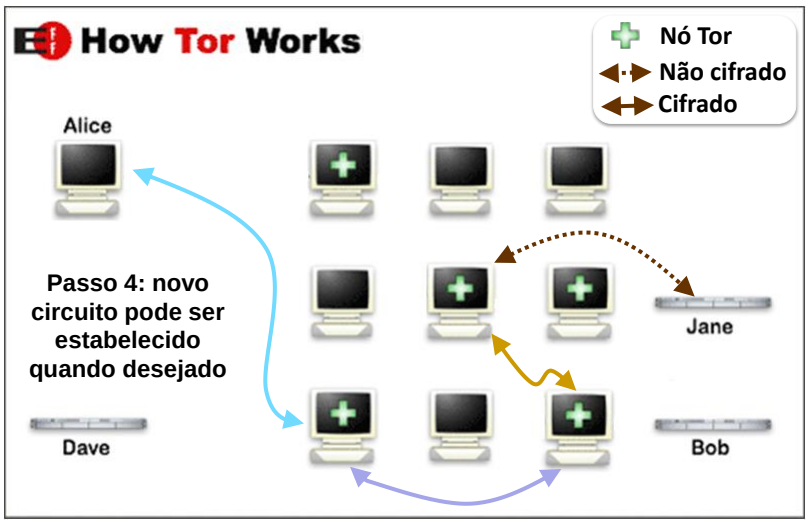



Formato das mensagens Tor saindo da origem:






## Tor: resumo



## Tor: características

- "Perfect Forward secrecy"

  - No onion routing, um nó poderia gravar mensagens e depois comprometer a chave privada de todos os nós até o destino
  - No Tor, são criadas chaves de sessão, que são removidas após uso e, assim, não podem ser comprometidas

- Alguns serviços:

  - Diversos fluxos TCP podem **compartilhar** um mesmo circuito
  - Verificação de **integridade dos dados** antes da saída da rede
  - Nós confiáveis atuam como **servidores de diretório**: listas de roteadores conhecidos assinadas digitalmente
  - Pontos de encontro (rendezvous) e **serviços escondidos**: anonimato dos servidores



## Servidores com localização oculta


- Objetivo: servidor na Internet com as seguintes características:

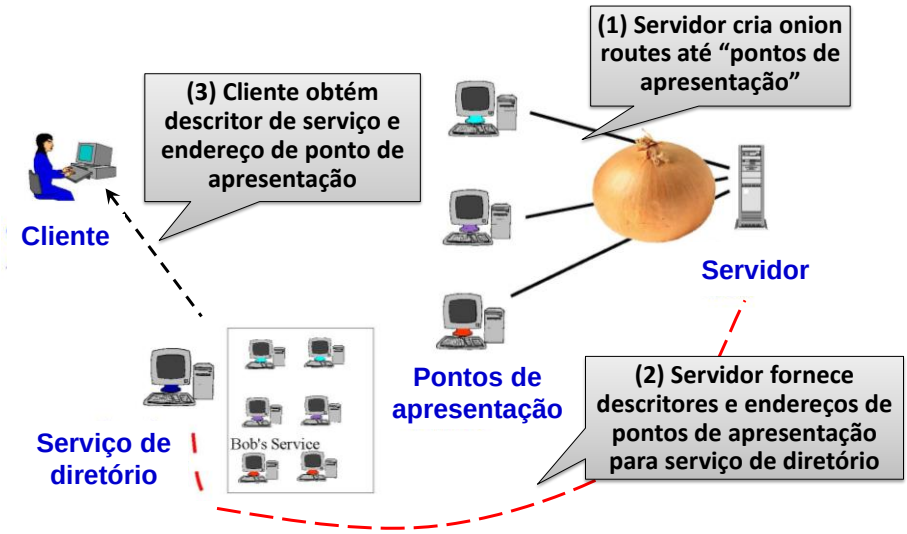

  - **Disponibilidade**: acessível por qualquer pessoa de qualquer lugar,

  - **Privacidade**: pessoas que acessam servidor não sabem onde ele está ou quem o controla

- **Resultado**: servidor resistente a censura

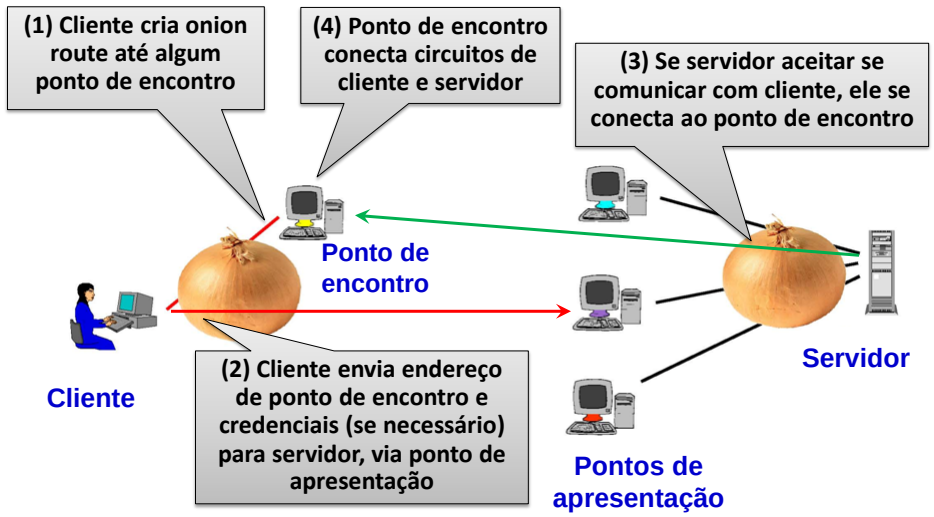

  - Capaz de resistir a ataques de negação de serviço: serviço distribuído, com acesso controlado
  - Resistente a captura física: não se sabe onde está o servidor físico!

## Criando um servidores com localização oculta






## Usando servidor com localização oculta



## Ataques à rede Tor

- Ataques passivos
  - Não é tão difícil saber se um nó está executando protocolo Tor: o difícil é saber com quem ele está se comunicando
- Ataques ativos
  - DDoS, controle de um nó da rede Tor
- Ataques aos diretórios
  - Destruição ou subversão de servidores de diretório
- Pontos de encontro
  - Ataque a pontos de encontro ou pontos de apresentação



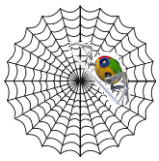
## Tor, Web e "Deep Web": Indexação

- **Web Crawling**: **indexação** automática de páginas Web
  - Usado, por exemplo, para construir a base de dados de **sites de busca**
- **Web crawlers** (ou Web spiders): programas de computador que **automatizam indexação**
  - Visitam páginas e indexam texto visível e metadados
  - Seguem hiperlinks encontrados, continuando "navegação" pela Web e descobrindo novos sites
  - Podem executar indefinidamente, identificando modificações em páginas já visitadas.

# Tor, Web e "Deep Web": Indexação

- "Deep web": conteúdo web não indexado por máquinas de busca tradicionais
  - Conteúdo **gerado dinamicamente** (ex.: via Javascript)
  - Conteúdo para o qual **não existem links** em sites já indexados
  - Conteúdo em **sites privados ou de acesso restrito**
    - Exigem login ou acesso via canal específico (ex.: redes Tor ou Freenet)
  - Texto embutido em **arquivos multimídia**

*"surface web"*



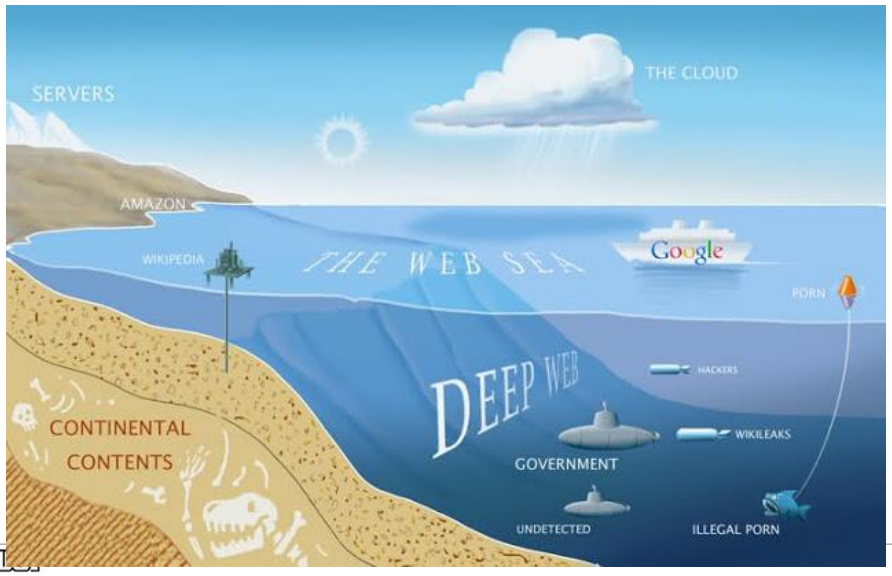
*(parte da) "deep web"*

*não é lido por web crawlers*



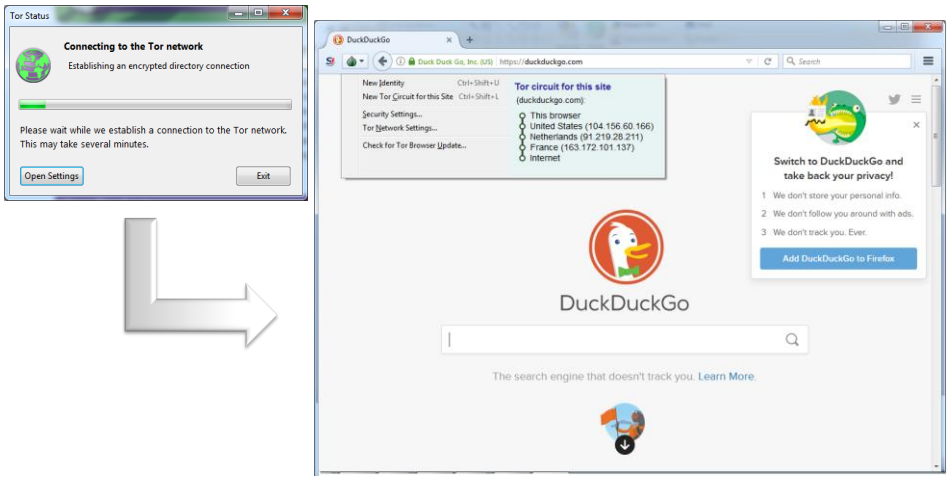

# Tor, Web e "Deep Web": Indexação

***Termo às vezes usado (de modo simplista) como "conteúdo ilegal"***



## Tor Browser: teste você mesm@!

- Página oficial: https://www.torproject.org/






# Referências


- [BH16] BakerHostetler, 2016. Is Your Organization Compromise Ready? 2016 Data Security Incident Response Report. URL: http://f.datasrvr.com/fr1/516/11618/BakerHostetler_2016_Data_Security_Incident_Response_Report.pdf
- [BL17] D. Bernstein, T. Lange (2017) Post-quantum cryptography - dealing with the fallout of physics success. URL: https://eprint.iacr.org/2017/314
- [NIST18] NIST, 2018. The Race to Build a Quantum Computer. URL: https://www.nist.gov/topics/physics/introduction-new-quantum-revolution/race-build-quantum-computer
- [NIST18b] NIST, 2018. NIST Special Publication 800-63B -- Digital Identity Guidelines. URL: https://pages.nist.gov/800-63-3/sp800-63b.html
- [NIST18c] NIST, 2018. Project -- Post-Quantum Cryptography. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography
- [NW17] R. Niederhagen, M. Waidner (2017) Practical Post-Quantum Cryptography. URL: https://www.sit.fraunhofer.de/fileadmin/dokumente/studien_und_technical_reports/Practical.PostQuantum.Cryptography_WP_FraunhoferSIT.pdf
- [PZ03] J. Proos, C. Zalka (2003). Shor's discrete logarithm quantum algorithm for elliptic curves. URL: https://arxiv.org/abs/quant-ph/0301141

# Segurança da Informação
# Uma visão da academia
**(ou, mais exatamente, de um acadêmico)**

**Prof. Dr. Marcos A. Simplicio Jr.**
Escola Politécnica da USP

Phishing

# APÊNDICE

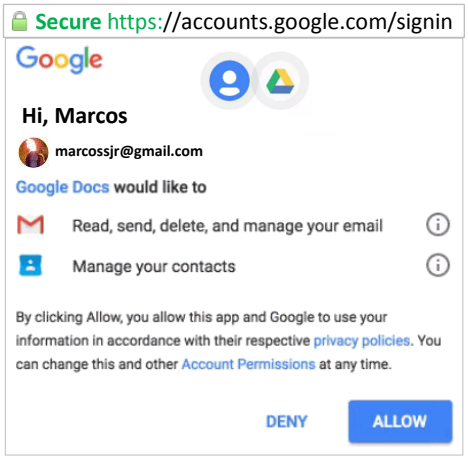
# Exemplo: Phishing?

- E parece ter algo errado no seguinte cenário?

João convidou você para editar o seguinte documento:

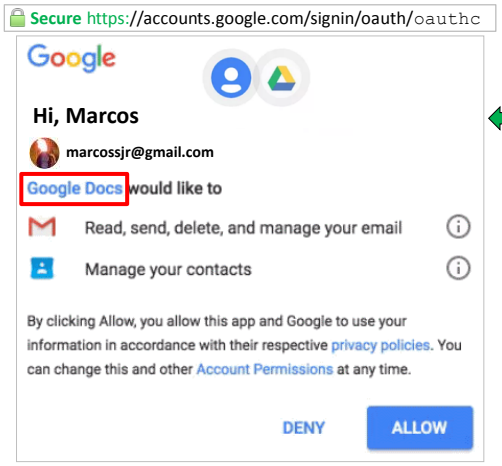
Open in Docs

**Desconfiado? Um pouco mais de detalhes…**



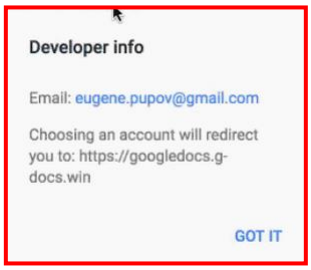

# Exemplo: Phishing?

- E parece ter algo errado no seguinte cenário?




É a página do Google: o certificado é válido

São suas informações cadastradas no Google




**Não parece ser o Google Docs…**

# Exemplo: Phishing?

- E parece ter algo errado no seguinte cenário?
  - Sim: é um caso de phishing explorando o processo de autenticação via Oauth do Google
    - (2017): mais de 1 milhão de usuários afetados
    - Acesso ao e-mail do usuário (e outras informações pedidas)
  - O “Google Docs” é um aplicativo criado pelo atacante, não o serviço do Google