Blockchain, Criptomoedas & Tecnologias Descentralizadas

Criptografia simétrica

Geração de números aleatórios

Prof. Dr. Marcos A. Simplicio Jr. – mjunior@larc.usp.br
Escola Politécnica, Universidade de São Paulo

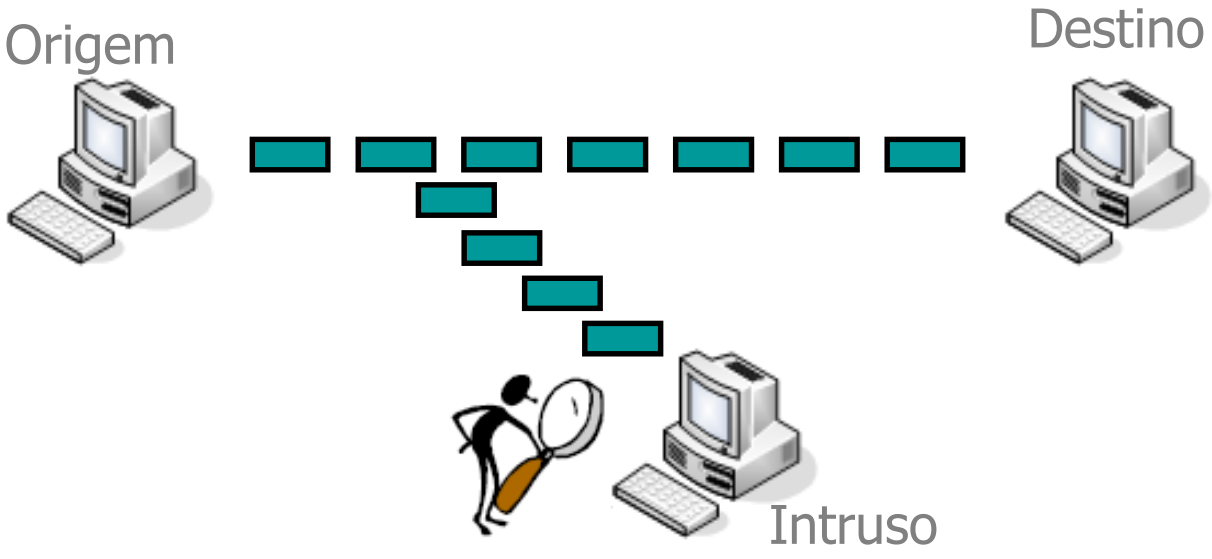
Objetivos

- Discutir os principais serviços fornecidos por algoritmos criptográficos simétricos
  - ~~Disponibilidade~~
  - Confidencialidade: **cifras**
  - Integridade: **funções de hash**
  - Autenticidade: **códigos de autenticação de mensagens**
  - ~~Irretratabilidade~~
- Discutir mecanismo auxiliar: geração de **números aleatórios**

Confidencialidade: Cifras

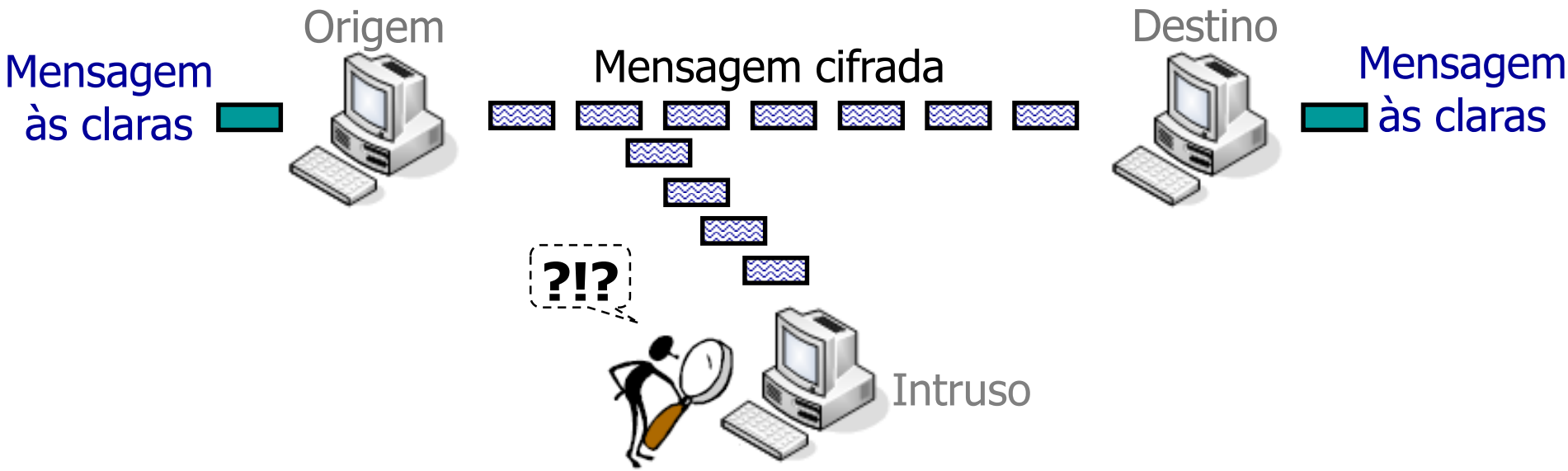
Confidencialidade

- Serviço necessário:
  - Prevenção do vazamento de informações



Confidencialidade: Cifras

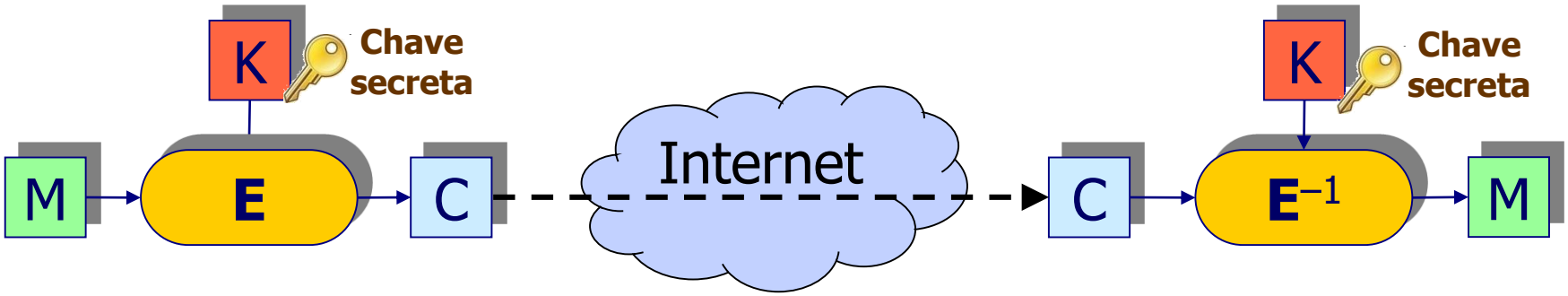
"Embaralhamento" de dados: **cifras**



"Seu saldo é R$10.000,00" ---- "Hlaafd7Y(@&fhF23%7" ----▶ "Seu saldo é R$10.000,00"

Cifra Simétrica – Definição

- Transformação matemática inversível cujo cálculo depende, no sentido direto (**cifração**) e no sentido inverso (**decifração**), de uma *mesma* informação secreta: a chave K.
  - Se K for descoberta, a confidencialidade é perdida



Cifras: Algoritmos Principais

- DES (Data Encryption Standard):
  - Blocos de 64 bits; Chaves de 56 bits
  - **Obsoleto:** aposentado em 2004 (chaves muito curtas!)

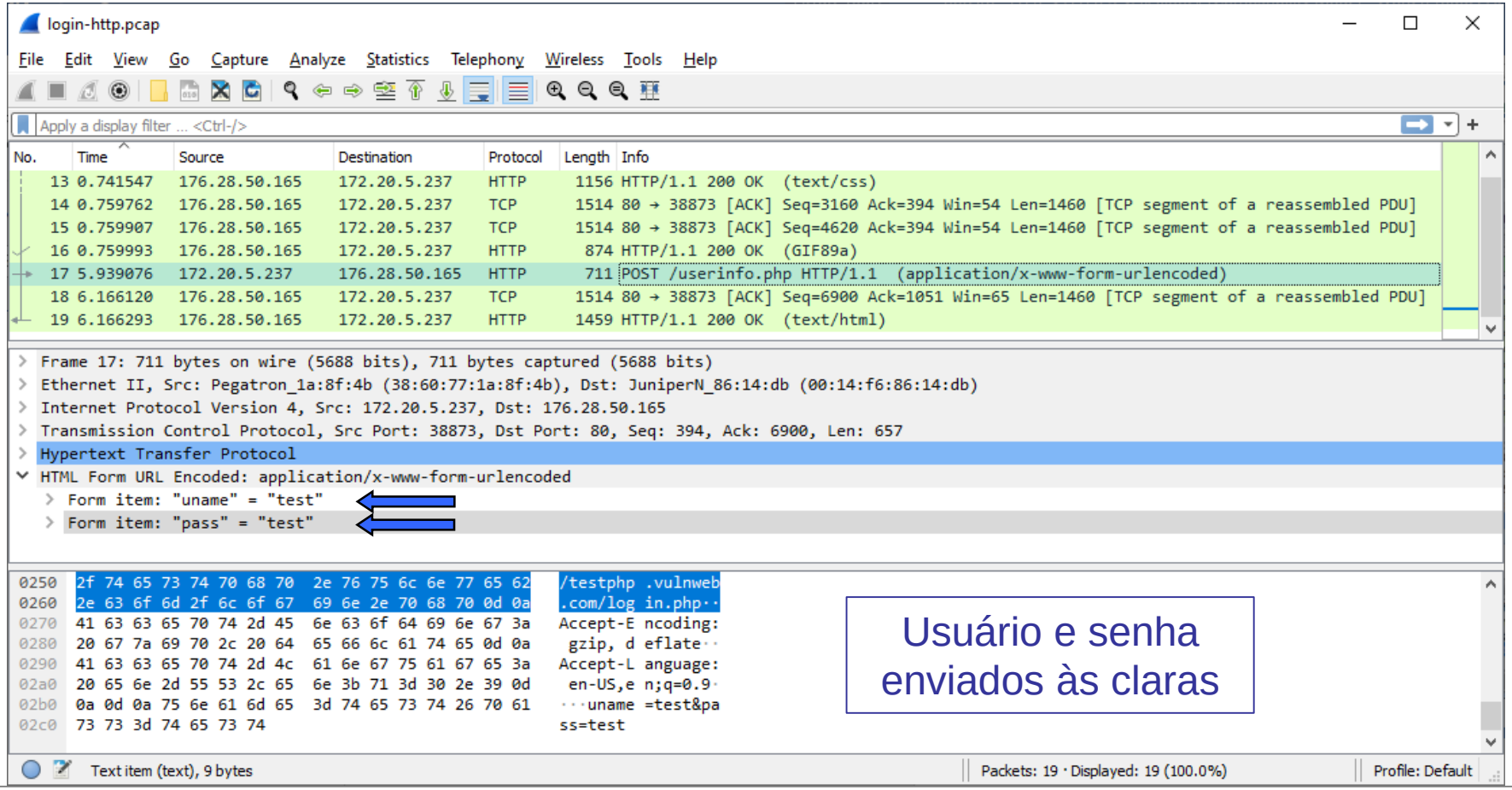
- RC4 (ArcFour):
  - Chave: tamanho variável (múltiplo de 8 bits, até 2048 bits).
  - **Legado**: antigo padrão do SSL/TLS, **aposentado** em 2015 (RFC 7465)

- 3DES (DES triplo):
  - **Legado**: tripla aplicação do DES, aproveitando implementações existentes ➔ **desaconselhado** em 2017; **aposentado** em 2023
  - Chaves: 3*56 = 168 bits (mas segurança é de ~112 bits)

- **AES (Advanced Encryption Standard):**
  - **Padrão atual** (desde 2001): vencedor de concurso público iniciado em 1997 (nome original: Rijndael)
  - Blocos de 128 bits; Chaves de 128/192/256 bits.

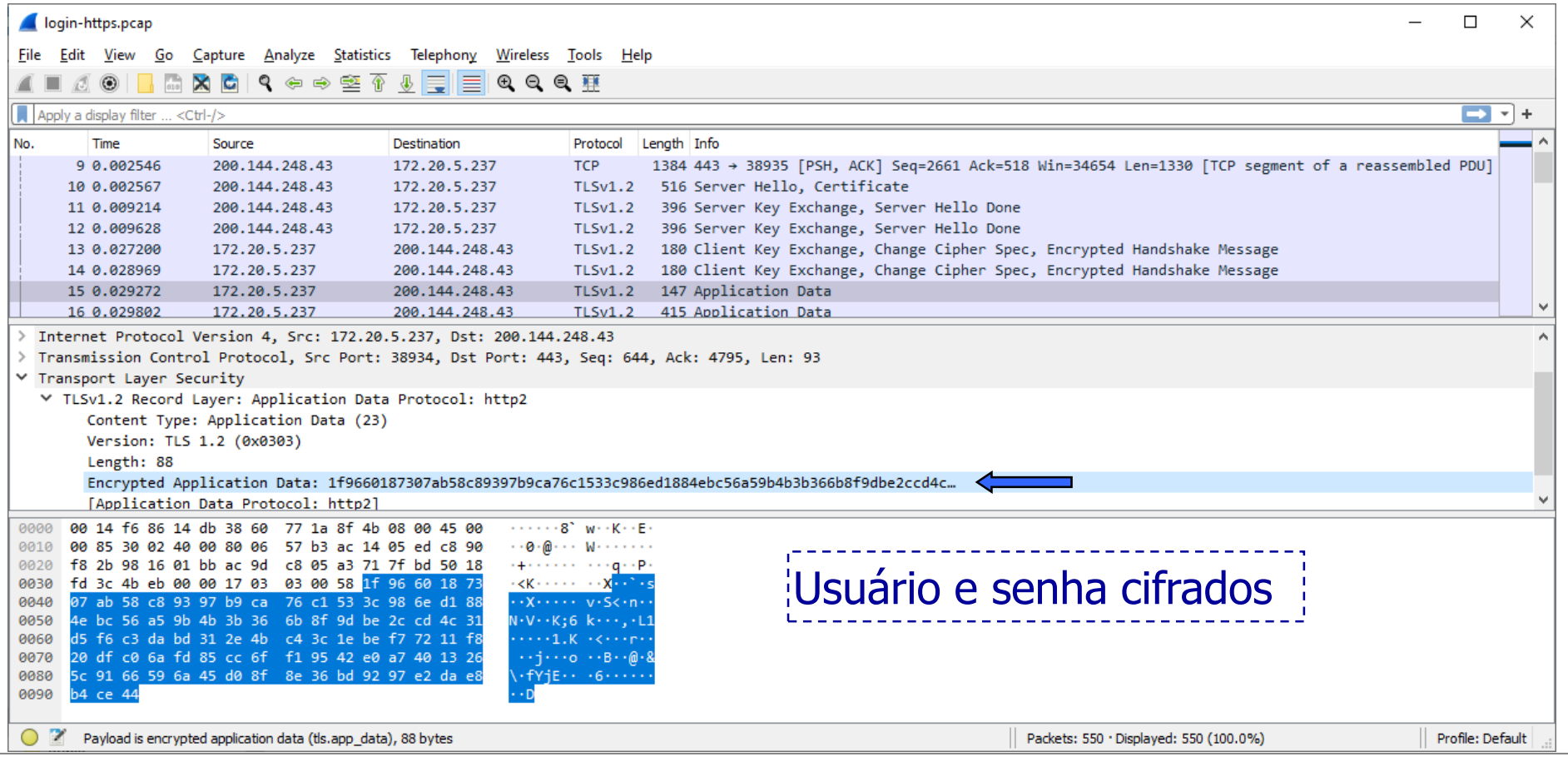
Exemplo prático: HTTP vs. HTTPS

- HTTP: dados passam em aberto na rede
  - Site de testes: http://testphp.vulnweb.com/login.php

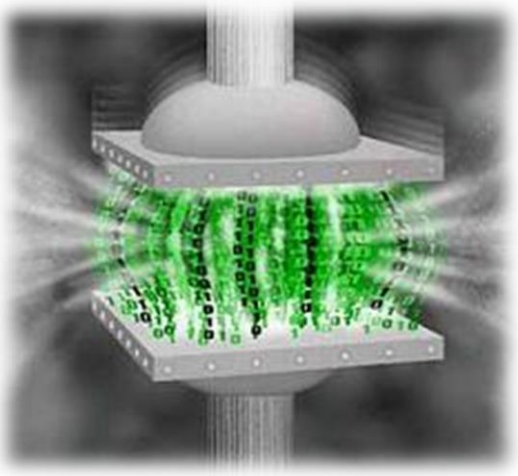


Exemplo prático: HTTP vs. HTTPS

- HTTPS: dados cifrados (túnel SSL/TLS)
  - Login em https://uspdigital.usp.br/



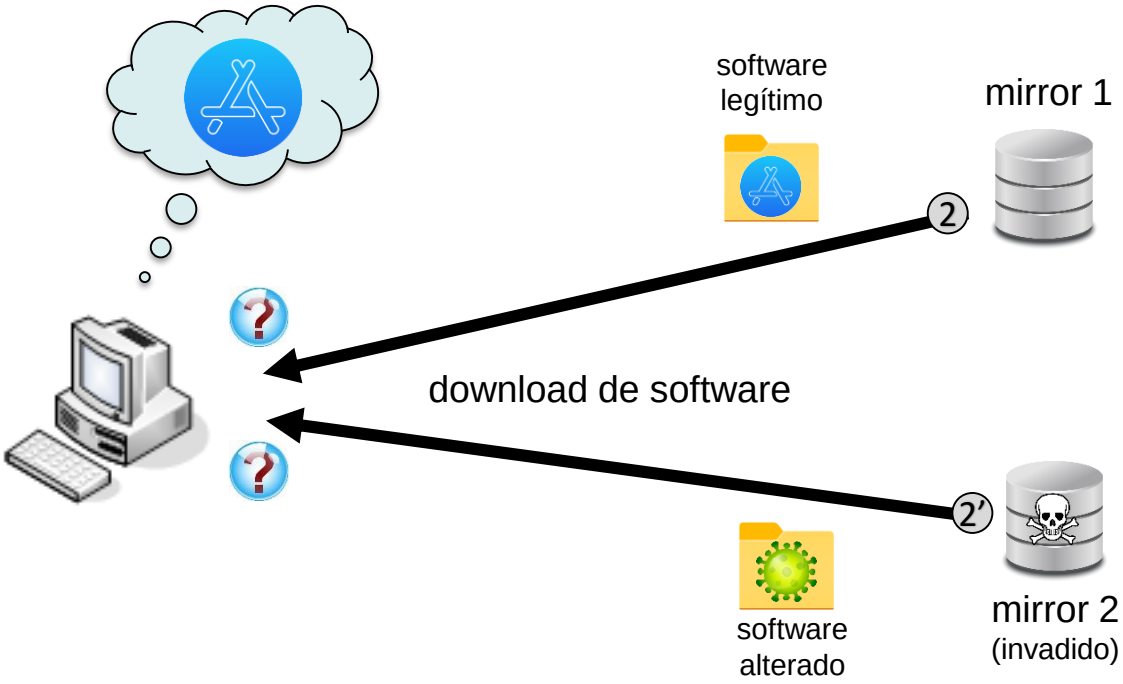
Usuário e senha cifrados



Integridade: Funções de Hash

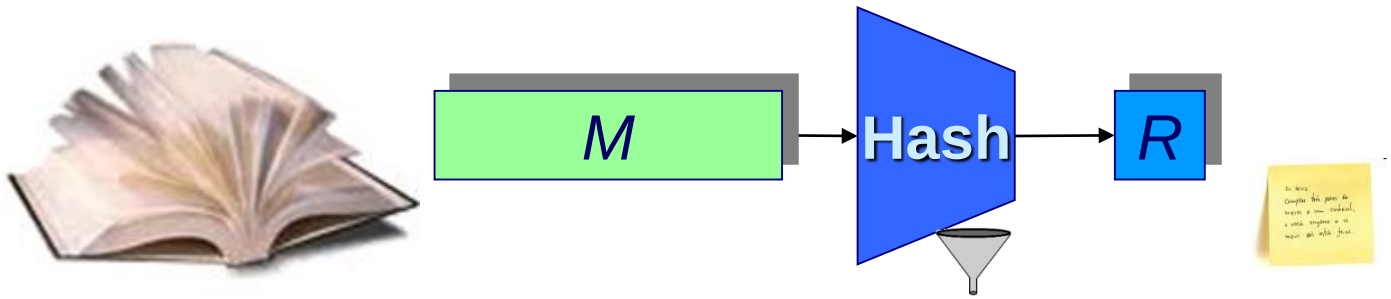
Integridade

- Serviço necessário
  - Capacidade de verificar se informação foi alterada



Integridade: Funções de Hash

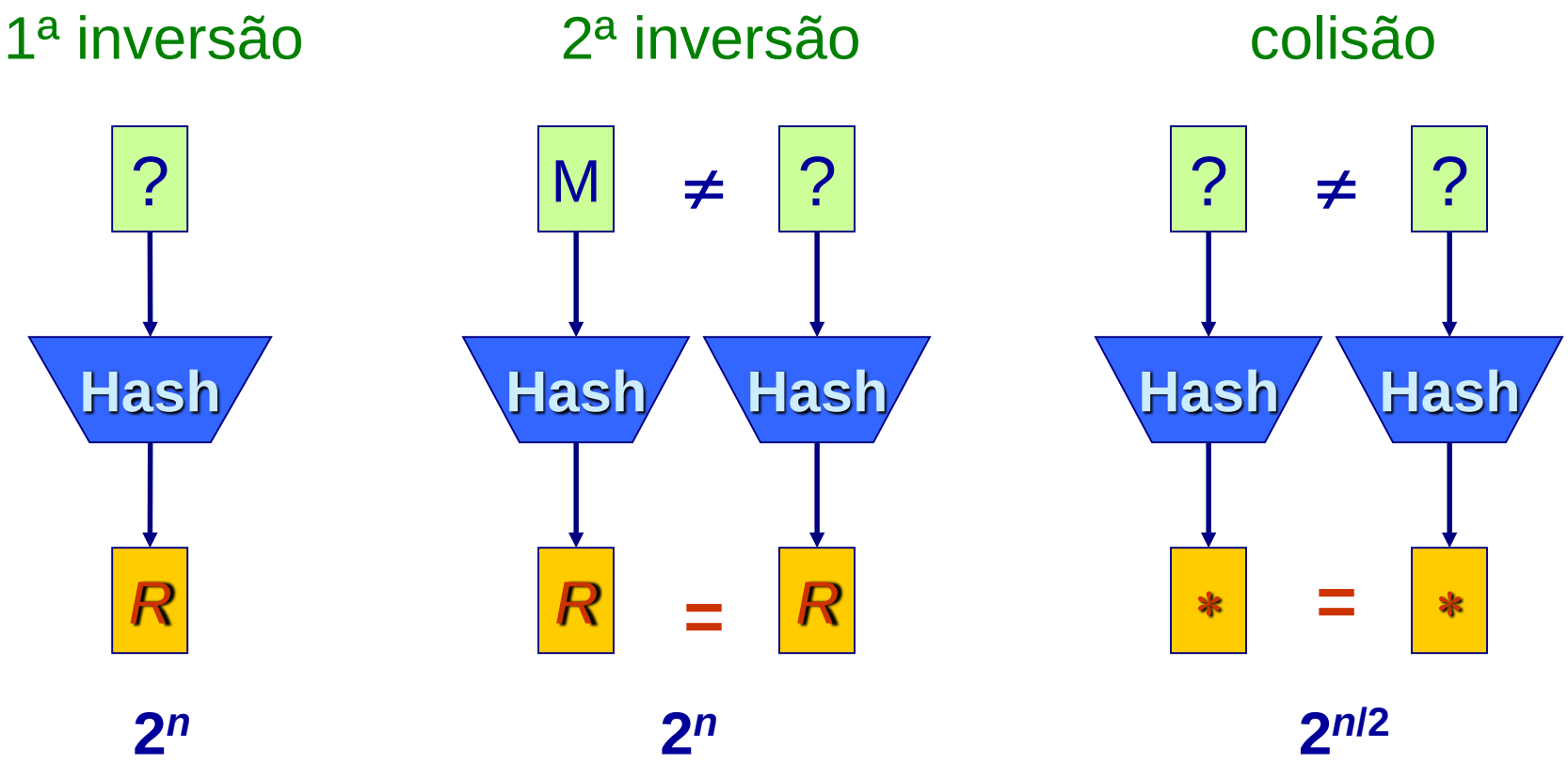
- Geram um "**resumo criptográfico**" da entrada
  - **Alterações** nos dados de entrada são **detectadas** porque elas **alteram o resumo**
  - O resumo também é chamado de "***hash***"
  - O hash tem **tamanho fixo**, e seu valor depende exclusivamente da mensagem (**não envolve** o uso de uma **chave secreta**)



Integridade: Funções de Hash

- **Requisitos criptográficos fundamentais**
- (Resistência a primeira inversão) Dado um resumo R, é inviável encontrar uma mensagem M tal que R = H(M).
- (Resistência a segunda inversão) Dado um resumo R e uma mensagem M_1 tal que R = H(M_1), é inviável encontrar outra mensagem $M_2 \neq M_1$ tal que R = H(M_2).
- (Resistência a colisões) É inviável encontrar duas mensagens M_1 e M_2 tais que H(M_1) = H(M_2).

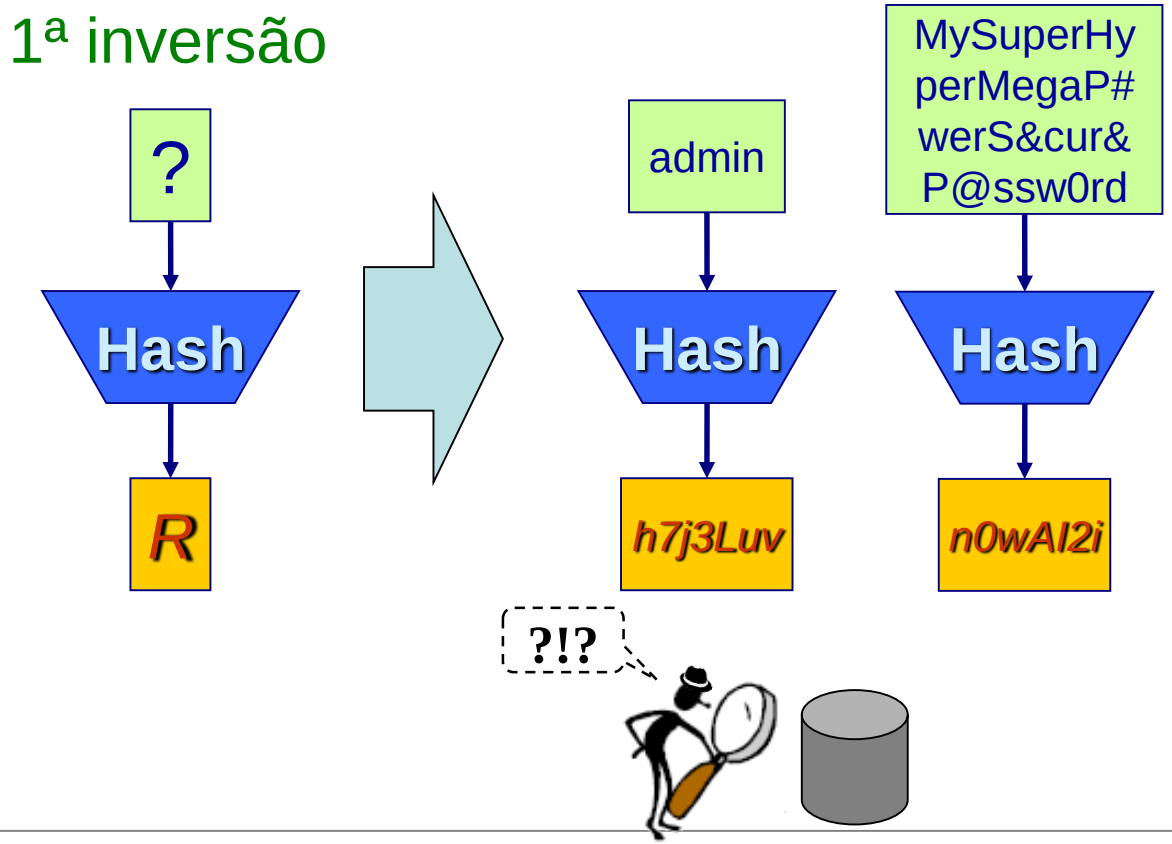
Integridade: Funções de Hash



Custo do ataque: hash de n bits

Funções de Hash: usos

Proteção de senhas em bancos de dados

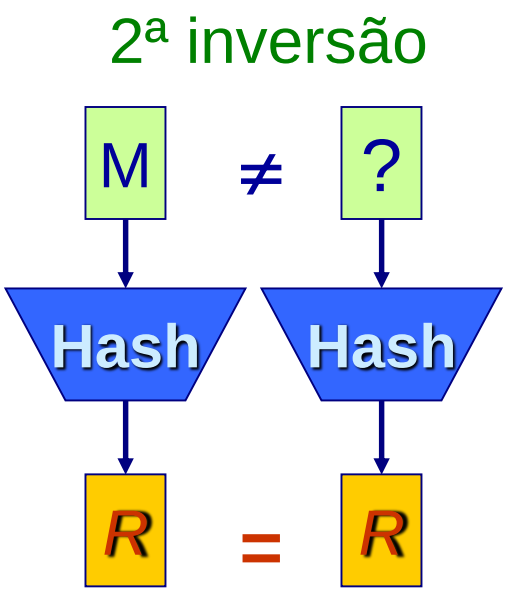


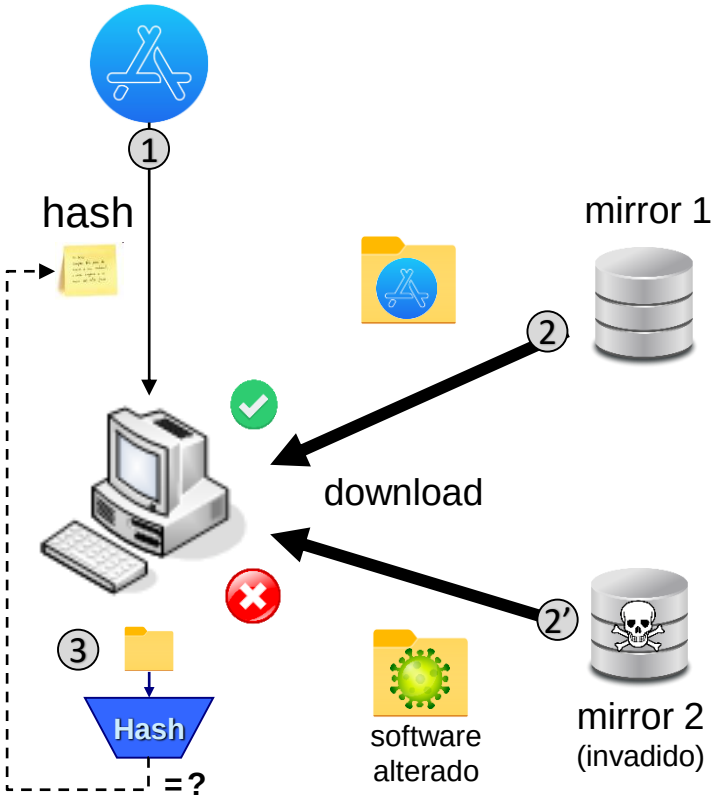
* Na prática, são usados algoritmos derivados de funções de hash: password hashing schemes (PHS)

† Não confundir com cifração, que é usada quando se deseja que alguém autorizado (i.e., de posse de chave secreta) consiga obter entrada a partir da saída

Funções de Hash: usos

Verificação de downloads

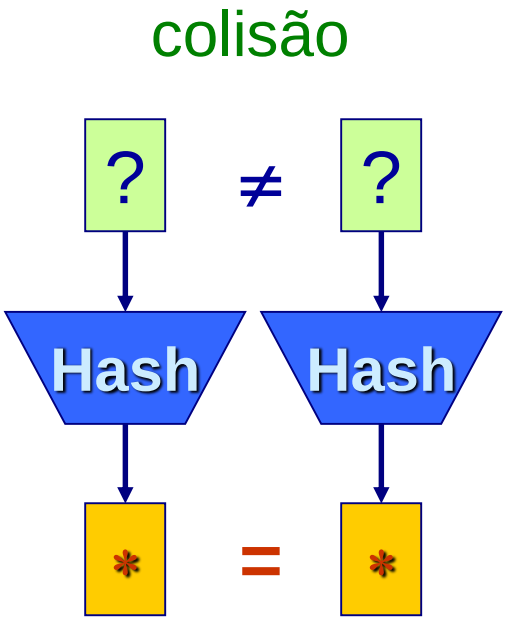


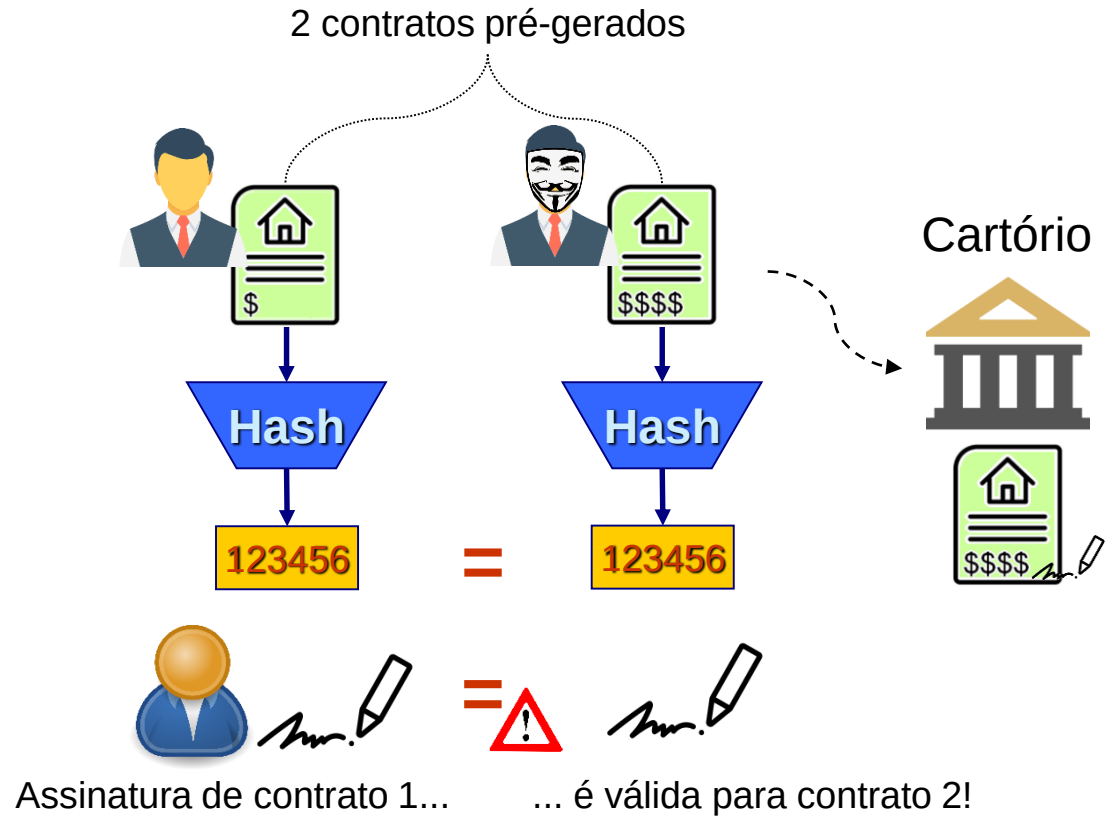


Funções de Hash: usos

Integridade de assinaturas digitais

(são feitas sobre hashes dos dados)

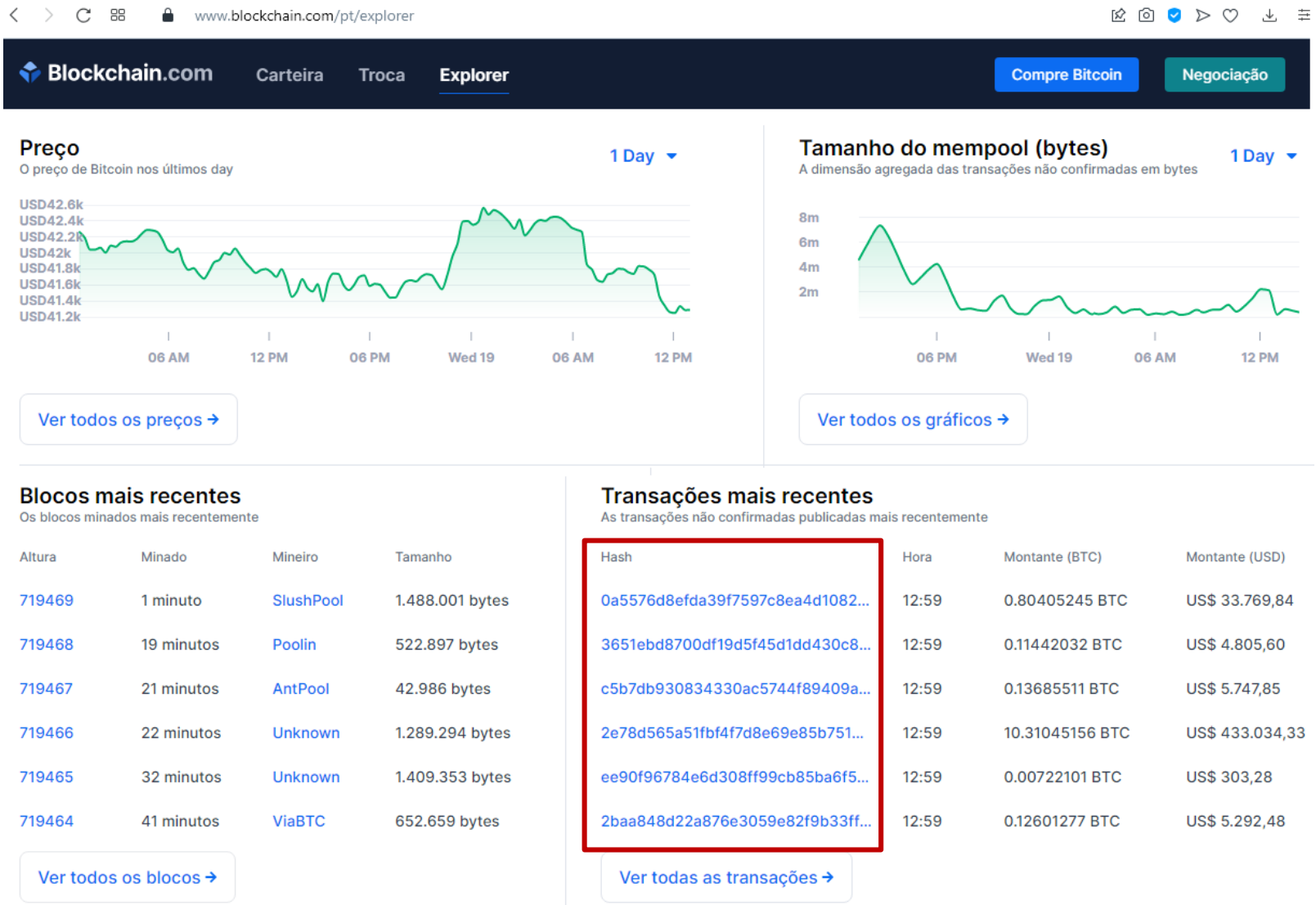




Integridade: Funções de Hash

- Família MD:
  - MD2, MD4 e MD5: hashes de 128 bits
    - Completamente quebrada (Wang et al., 2004)
- Família SHA
  - SHA-0: hashes de 160 bits
    - Não recomendado: colisão em 2^{39} passos x 2^{80} projetado
  - SHA-1: hashes de 160 bits
    - Não recomendado: desde 2010, para assinaturas
    - Segurança: colisões em 2^{60} passos x 2^{80} projetado
  - **SHA-2**: Hash de X bits, para X=224, 256, 384 ou 512
    - Paliativo atual: baseados no SHA-1, mas hash grande dificulta ataques
  - **SHA-3**: hashes de 224, 256, 384 e 512 bits
    - Concurso público finalizado em 2012: **Keccak**

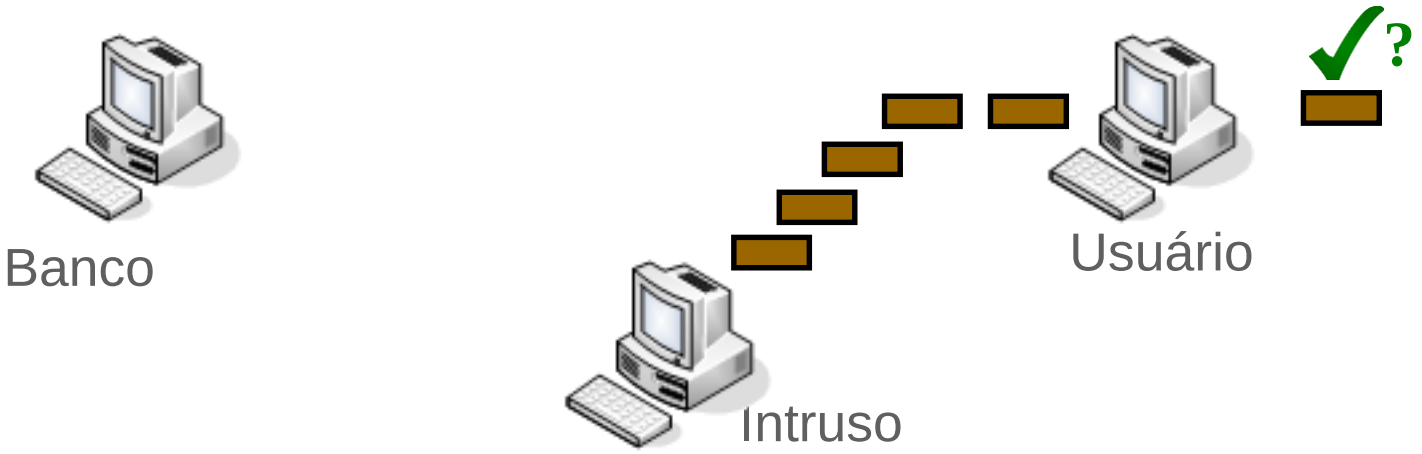
Exemplo prático: Bitcoin



Autenticidade: Códigos de Autenticação de Mensagens (MAC)

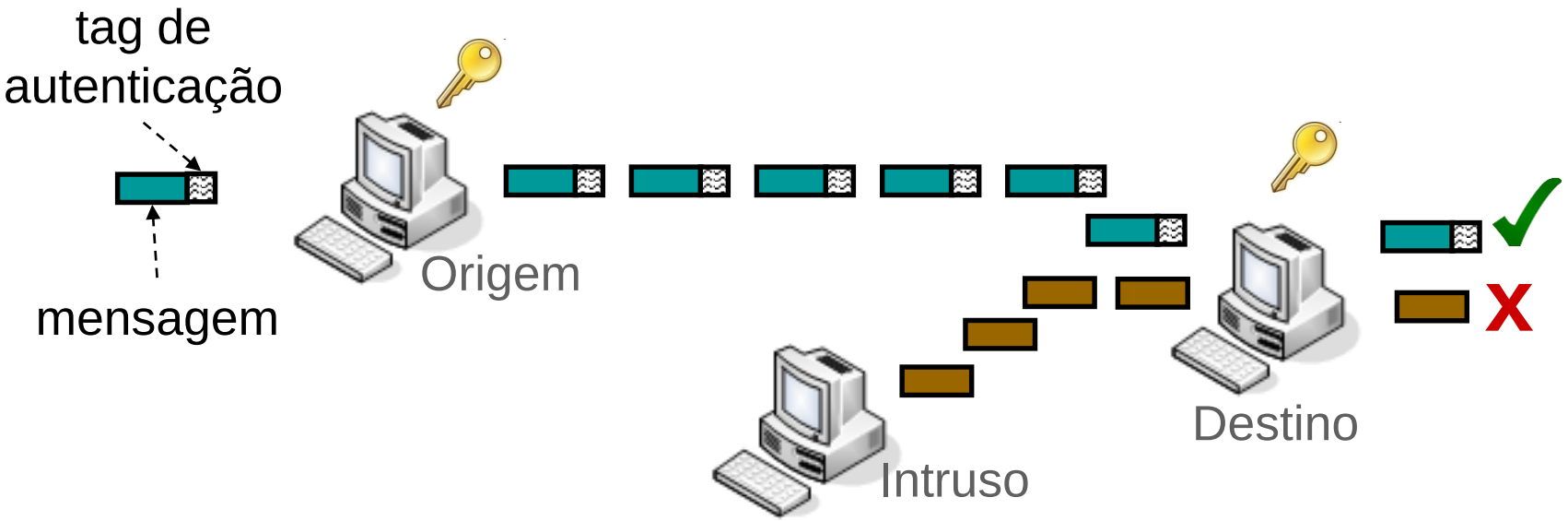
Autenticidade

- Serviço necessário:
  - Capacidade do receptor em verificar quem é o emissor da mensagem



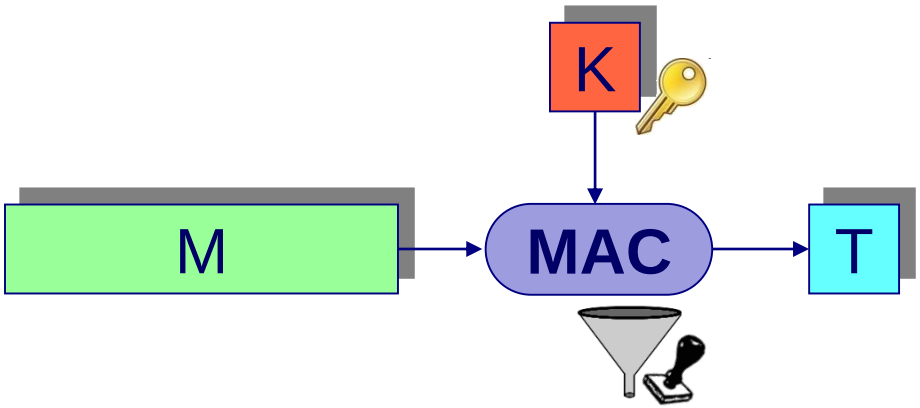
Autenticidade: estratégia básica

- Usar ***resumo criptográfico dependente de chave***
  - Apenas origem e destino conhecem a chave e conseguem calcular resumo corretamente
  - Também garante integridade (alteração na mensagem detectada, como no caso das funções de hash)



Códigos de Autenticação

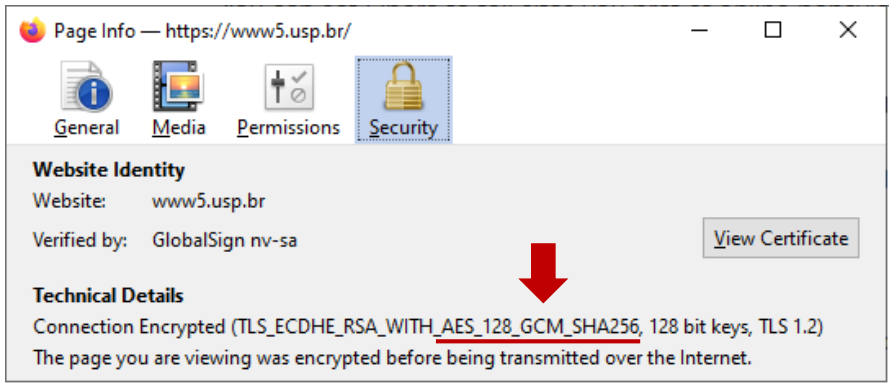
- Message Authentication Code (**MAC**):
  - Cria resumo que é anexado à mensagem, permitindo detectar alterações (**integridade**) e garantir a **autenticidade** do remetente.
- Resumo: "tag (etiqueta) de autenticação"
  - Depende da **mensagem** e também de uma **chave secreta** conhecida por remetente e destinatário.

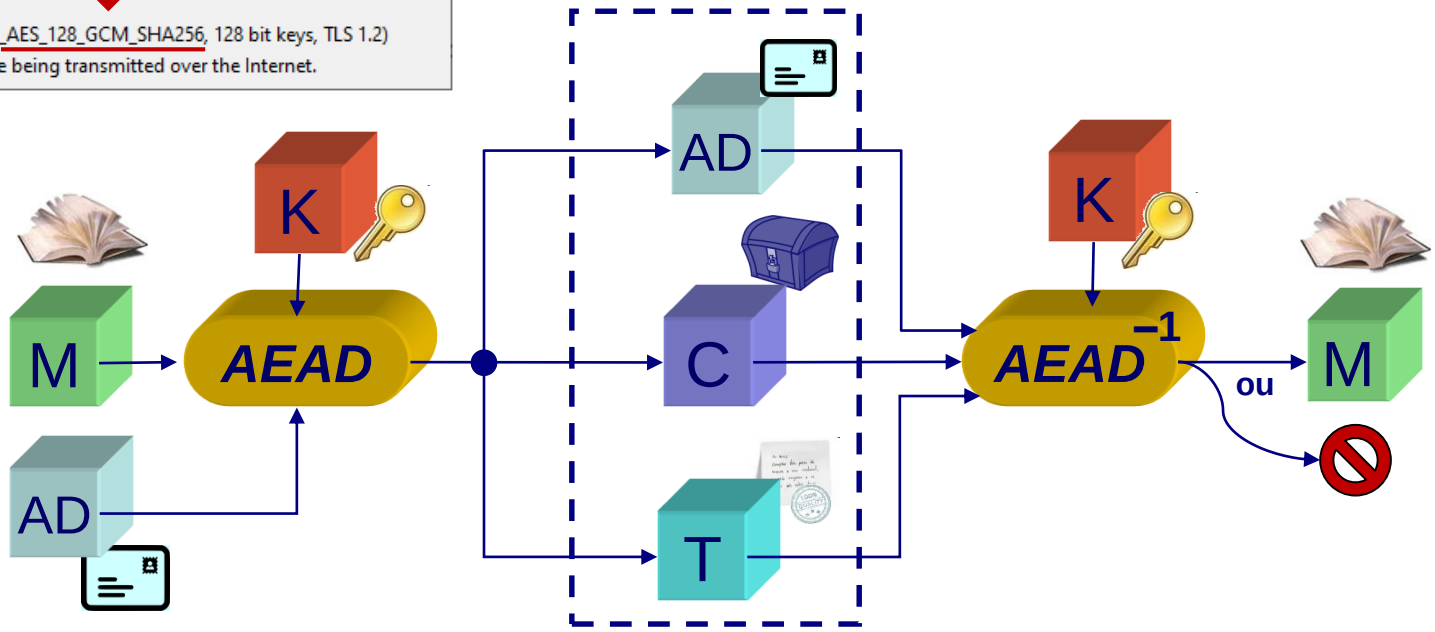


Autenticidade: Algoritmos

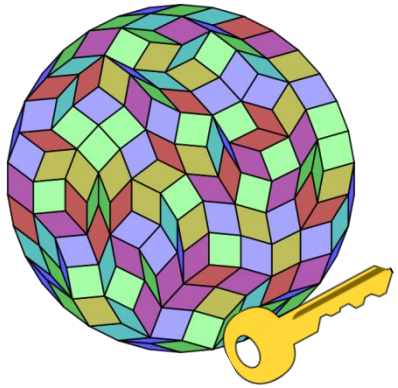
- Baseados em cifras de bloco:
  - **CMAC** (NIST SP 800-38B).
  - Pró: tamanho de código (reusam cifras de bloco).
- Baseados em funções de *hash*:
  - **HMAC** (FIPS 198).
  - Pró: desempenho (funções de *hash* puras).
- Combinados com cifras
  - **AEAD**: *Authenticated Encryption with Associated Data* (confidencialidade de parte dos dados)
    - Exemplos tradicionais: **GCM, CCM, EAX**
  - **Concurso** finalizado em 2018 (Caesar): (http://competitions.cr.yp.to/caesar-submissions.html)
    - Ascon (mais leve), AEGIS-128 & OCB (alto desempenho), Deoxys-II (defesa em profundidade: e.g., não requer nonces)

Exemplo prático: TLS





- Dado ***confidencial*** (C) e ***autenticado*** (T)
  - AD: dados associados (enviados às claras, autenticados)
  - Serviços: confidencialidade, integridade e autenticidade (cifra simétrica e algoritmo de MAC internos a AEAD)



Geração de chaves: números aleatórios

Estudo de caso: Netscape

- Netscape 1.x (1995).
- Dois estudantes de Berkeley descrevem como quebrar a segurança do navegador, recuperando chaves usadas em sessões seguras (HTTPS) em **25 s**.
- Chaves pequenas?
  - Não, chaves de 128 bits (tamanho atual !!!)
- Pergunta: como isso é possível?

Análise de (in)segurança

- Baixa **aleatoriedade** das chaves de sessão!
  - Chaves geradas a partir do relógio do sistema (precisão de µs), sem acúmulo entre ativações
  - Conhecendo minuto da criação da sessão HTTPS: menos de 60 milhões de chaves possíveis
    - Segurança de cerca de 2^{26}, não 2^{128}
- Geração de chaves segura: fontes de **entropia**

  - Ex. (**físicas**): relógio, ruído térmico
  - Ex. (**comportamentais**): estatísticas de rede, pastas temporárias (Firefox 3.5), posição do mouse (VeraCrypt)
  - Soluções de **sistema**: “SecureRandom” (Java), “/dev/random” (Unix)

Geradores pseudo-aleatórios

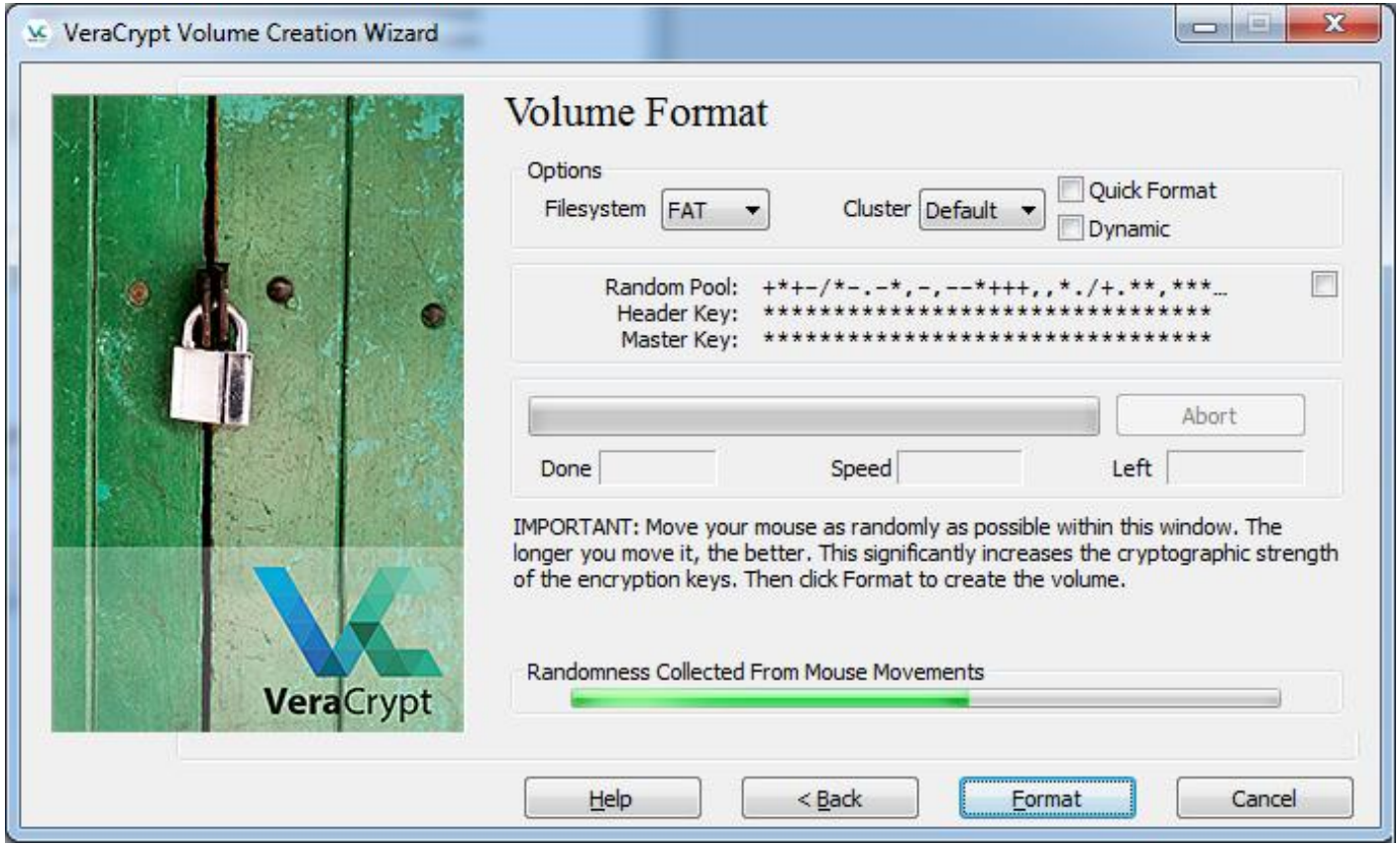
- Evitam necessidade de capturar entropia "bruta" repetidamente
  - Na prática: ganhos de desempenho

- Basicamente:
  - Coleta-se entropia bruta (de várias fontes!) para criar/atualizar **semente** de tamanho adequado.
    - Semente deve ser **mantida secreta**
  - Usa-se algoritmo determinístico para gerar uma sequência **"indistinguível" de bits aleatórios**.

- Algoritmos padrão: NIST-SP800-90A-Rev 1
  - **Revisão removeu Dual_EC_DRBG** (backdoor da NSA)

Entropia: exemplo prático

- Veracrypt: acúmulo de entropia fornecida pelo usuário (movimento do mouse)



Entropia: contra-exemplos

- Baixa entropia é recorrente na literatura… ☹
  - Debian OpenSSL (2008): chaves dependentes apenas de process-id e arquitetura de hardware
  - Urna eletrônica brasileira (2012): recuperação da ordem dos votos registrados na urna (RDV)
  - Chaves RSA geradas por dispositivos de rede (2012): repetição de números aleatórios (fator primo compartilhado), permitindo recuperação da chave privada.
    - Problema observado também em estudos em 2013 (smart cards), 2015 (servidores HTTPS) e 2017 (chaves Tor)
  - “Brain-wallet” (2015+): chaves privadas geradas a partir de senhas, facilitando roubo de criptomoedas

- D. Aranha, M. Karam, A. Miranda, F. Scarel. (In)segurança do voto eletrônico no Brasil / Vulnerabilidades no software da urna eletrônica brasileira. In: Cadernos Adenauer 1/2014: Justiça Eleitoral, 117-133, 2014. URL: https://sites.google.com/site/dfaranha/pubs/aranha-karam-miranda-scarel-12-pt
- N. Heninger, 2017. Random number generation done wrong. Crypto Experts -- wr0ng 2017. URL: https://cryptoexperts.com/wr0ng2017/slides-wrong/Wr0ng_2017_Nadia_Heninger.pdf
- E. Lacey, 2019. “A 'Blockchain Bandit' Is Guessing Private Keys and Scoring Millions”. Wired, Apr 2019. URL: https://www.wired.com/story/blockchain-bandit-ethereum-weak-private-keys/

Blockchain, Criptomoedas & Tecnologias Descentralizadas

Criptografia simétrica

Geração de números aleatórios

Prof. Dr. Marcos A. Simplicio Jr. – mjunior@larc.usp.br
Escola Politécnica, Universidade de São Paulo

Apoio financeiro: d24n.org <https://d24n.org> ripple impact | university blockchain research initiative

Referências

- W. Stallings, L. Brown “Computer Security Principles and Practice – 2nd/3rd/4th edition”. Prentice-Hall, ISBN: 0-13-277506-9. 2011/2015/2018.
  - Em português: W. Stallings, L. Brown. “Segurança de Computadores - Princípios e Práticas” (2ª Ed), Elsevier, 2014
- W. Stallings: “Cryptography and Network Security” (6th/7th Ed.), Prentice-Hall 2013/2016.
  - Em português: W. Stallings: “Criptografia e Segurança de Redes” (6ª Ed.), Pearson-Prentice-Hall (2014).
- S. Wykes. Criptografia Essencial: A Jornada do Criptógrafo, 1a ed. Elsevier, 2016.
- A. Narayanan, J. Bonneau, E. Felten. "Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction". Princeton University Press, 2016. ISBN: 0691171696. Available: https://d28rh4a8wq0iu5.cloudfront.net/bitcointech/readings/princeton_bitcoin_book.pdf?a=1