CSE 484 / CSE M 584

# Computer Security: More Cryptography

TA: Jared Moore

jlcmoore@cs

# Logistics

- Lab 1 Final due TOMORROW (11:59pm).
- For quickest response from TAs, email all of us:

  cse484-tas@cs.washington.edu

- Check forum for some tips.
- Homework #2 out now (crypto), due on Friday, 5/5 5pm.

# Last Week Questions

- What is the difference between AES and encryption modes (CBC, CTR, etc.)?
  - AES and DES define the block cipher and can be used with various modes
- Where are password salts stored? Do we assume an attack has access to them?
  - In the same file as the password hashes. Yes.

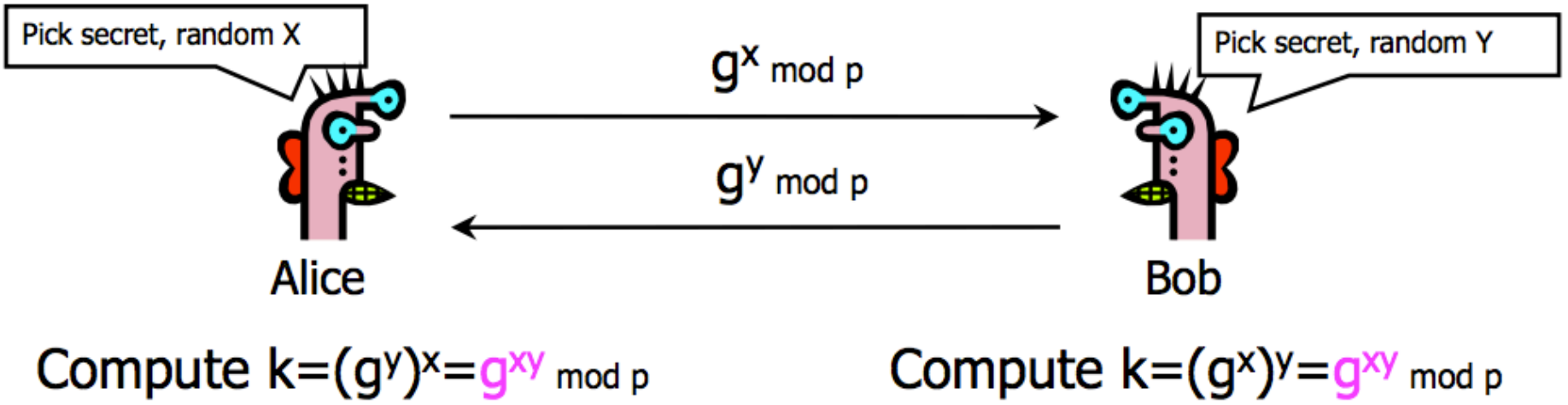
# Some Number Theory Facts

- Euler totient function $\varphi(n)$ ($n \geq 1$) is the number of integers in the [1,n] interval that are relatively prime to n
  - Two numbers are relatively prime if their greatest common divisor (gcd) is 1
  - Easy to compute for primes: $\varphi(p) = p-1$
  - Note that $\varphi(ab) = con\varphi(a)\ \varphi(b)$

- Euler's theorem: if $a \in Z_n^*$, then $a^{\varphi(n)} \equiv 1 \pmod{n}$

  $Z_n^*$: integers relatively prime to n

# DH Summary




- Public info: p (large prime) and
  g (generator of $Z_p^*$)

$Z_p^*=\{1, 2 \ldots p-1\}$; $\forall a \in Z_p^* \ \exists i$ such that $a \equiv g^i \pmod p$

# RSA Summary

- Key generation
  - Generate large primes p, q
    - Say, 1024 bits each (need primality testing, too)
  - Compute $n = pq$ and $\varphi(n) = (p-1)(q-1)$
  - Choose small e, relatively prime to $\varphi(n)$ (in $Z_n^*$)
  - Compute unique d such that $ed \equiv 1 \pmod{\varphi(n)}$
  - Public key = (e,n); private key = (d,n)
- Encryption of m: $c = m^e \bmod n$
  - Modular exponentiation by repeated squaring
- Decryption of c: $c^d \bmod n = (m^e)^d \bmod n = m$

# Why RSA Decryption Works

$e \cdot d \equiv 1 \pmod{\varphi(n)}$, thus $e \cdot d = 1 + k \cdot \varphi(n)$ for some $k$

Let m be any integer in $Z_n^*$ (not all of $Z_n$)

$c^d \bmod n = (\mathbf{m^e})^{\mathbf{d}} \bmod n = \mathbf{m}^{\mathbf{1+k \cdot \varphi(n)}} \bmod n$

$= (\mathbf{m} \bmod n) * (\mathbf{m}^{\mathbf{k \cdot \varphi(n)}} \bmod n)$

Recall: Euler's theorem: if $a \in Z_n^*$, then $a^{\varphi(n)} = 1 \bmod n$

$c^d \bmod n = (\mathbf{m} \bmod n) * (\mathbf{1} \bmod n)$

$= \mathbf{m} \bmod n$

Proof omitted: True for all m in $Z_n$, not just m in $Z_n^*$

# Read the paper!

- https://people.csail.mit.edu/rivest/Rsapaper.pdf

# Sample RSA Decryption

- 26 2 15 13 7 14 13 13 1 28 14 15 13
  14 20 9 6 31 25 26 14 16 23 15 26 2 6 13 1
- p=3, q=11, n=33, e=7, d=3

- A-1 B-2 C-3 D-4 E-5 F-6 G-7 H-8 I-9 J-10 K-11 L-12 M-13 N-14 O-15 P-16 Q-17 R-18 S-19 T-20 U-21 V-22 W-23 X-24 Y-25 Z-26

# Sample RSA Decryption

- How to compute d?
  - Recall: $ed \equiv 1 \pmod{\varphi(n)}$ (where $\varphi(n) = (p-1)(q-1)$)
  - So d is inverse of e mod φ(n).
  - How to compute modular inverse?
    - Use extended Euclidean algorithm
    - … or Wolfram Alpha ☺
    - Note that this is hard if you don't know φ(n) (i.e., can't factor n).

# Public Key Crypto Summary

- Diffie-Hellman: Why is it secure?
  - Discrete log; computational DH problem; decisional DH problem are hard.
- RSA: Why is it secure?
  - Taking $e^{th}$ root is hard; Factoring is hard.

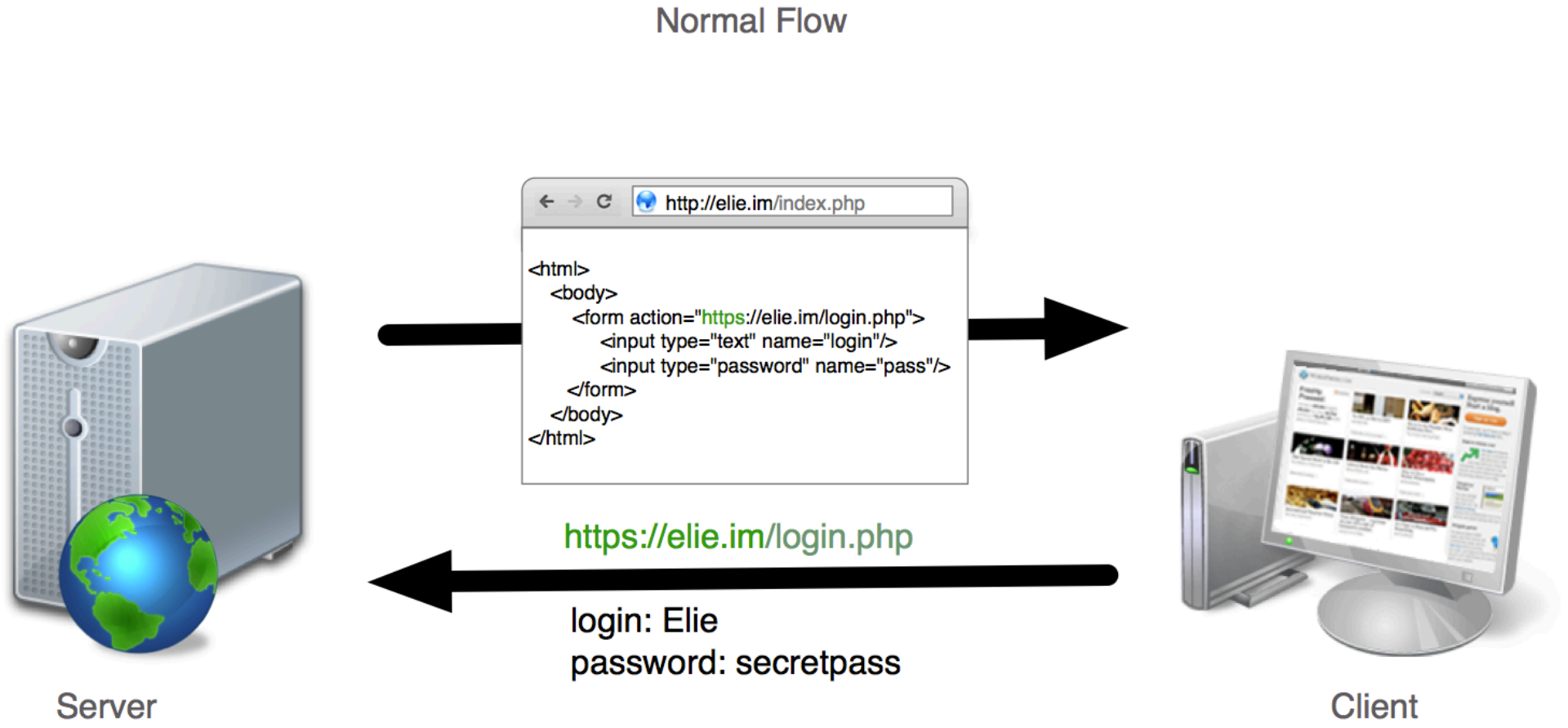
# Cryptography Summary

- Goal: Privacy
  - One-time pad
  - Block ciphers w/ symmetric keys (e.g., DES, AES)
    - Modes: EBC, CBC, CTR
  - Public key crypto (e.g., Diffie-Hellman, RSA)
- Goal: Integrity
  - MACs, often using hash functions (e.g, MD5, SHA-256)
- Goal: Privacy and Integrity
  - Encrypt-then-MAC (why?)
- Goal: Authenticity (and Integrity)
  - Digital signatures (e.g., RSA, DSS)

# Certificate Authorities

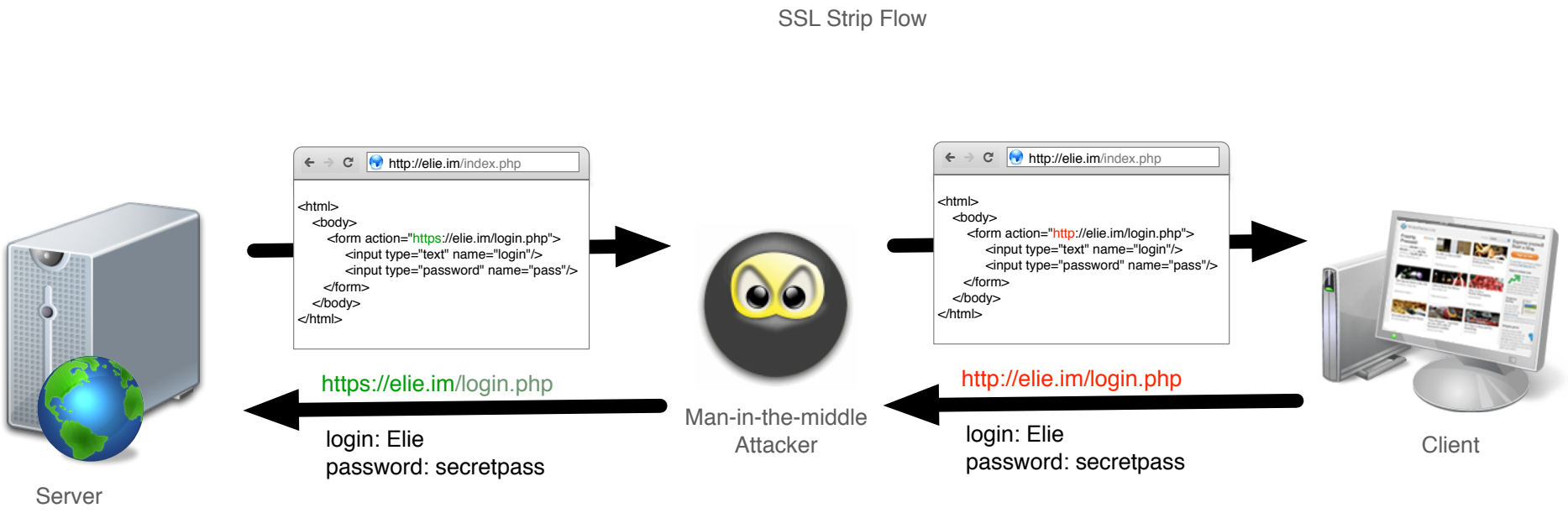
- CAs sign certificates; root CAs can authorize intermediate CAs (certificate chains).
- Problems with this model?
- Ideas for alternate solutions?
  - Examples: Perspectives (http://perspectives-project.org/), Convergence (http://convergence.io/)
    - Both rely on notary servers (chosen by the user): browser checks certificates it sees against those seen over time by trusted notaries. How does this help?

# SSL Strip Attack



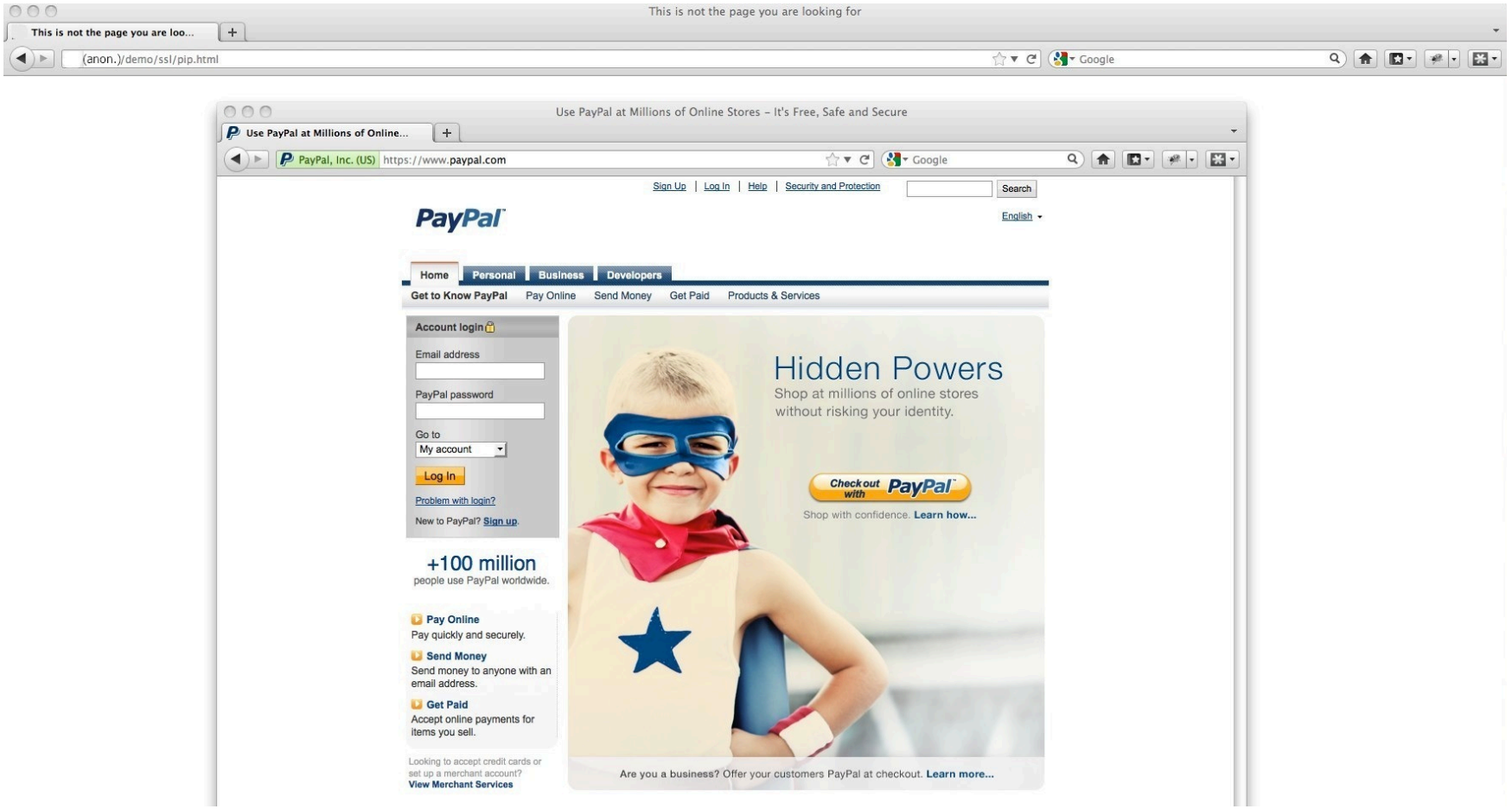

[Figures thanks to Elie Bursztein. See also http://www.thoughtcrime.org/software/sslstrip/.]

# SSL Strip Attack



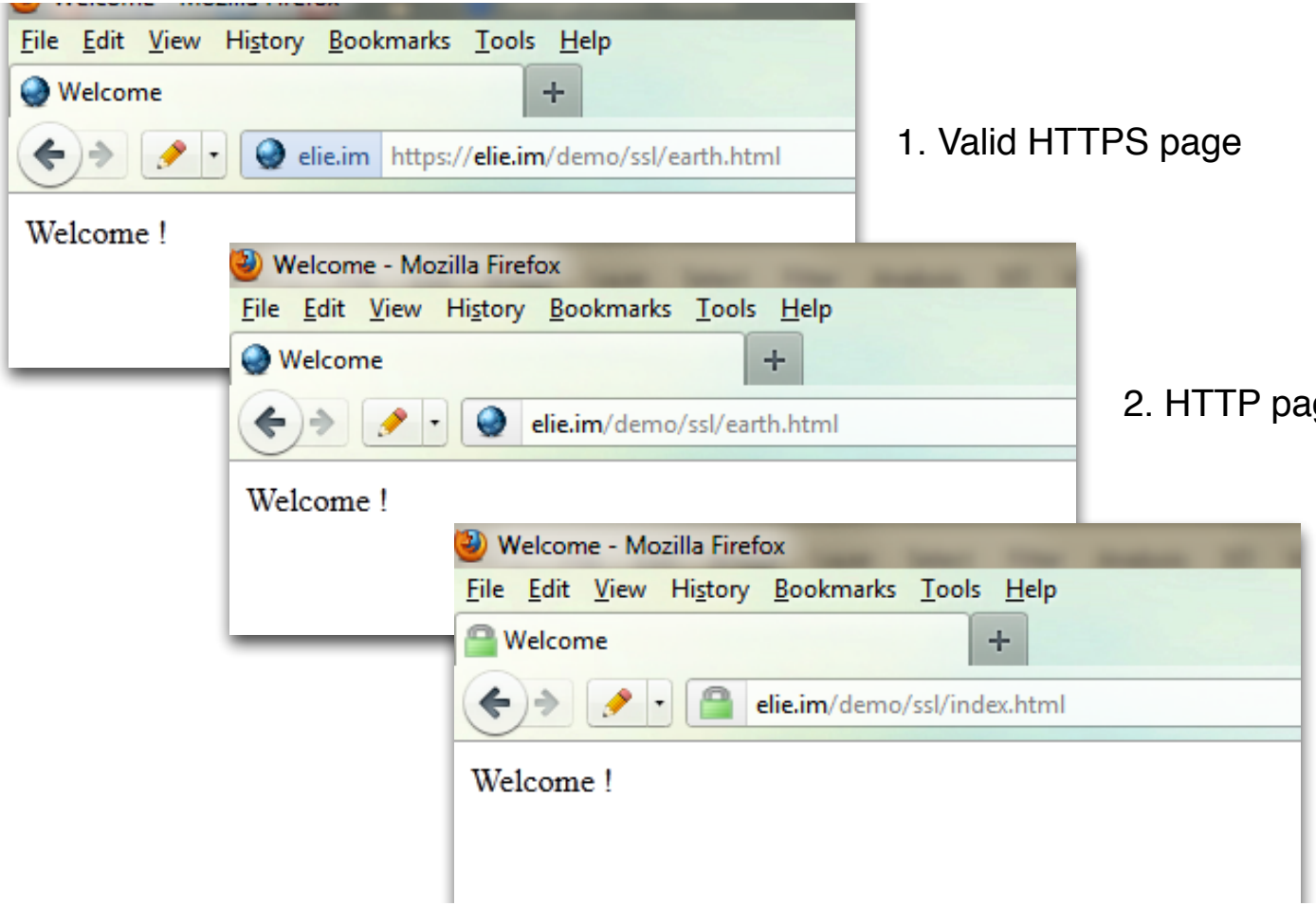

[Figures thanks to Elie Bursztein. See also http://www.thoughtcrime.org/software/sslstrip/.]

# SSL User Interface Attacks




[Figures thanks to Elie Bursztein]

# SSL User Interface Attacks




1. Valid HTTPS page

2. HTTP page

3. HTTP page
+ Lock spoofing

[Figures thanks to Elie Bursztein]

# SSL User Interface Attacks


[Figures thanks to Elie Bursztein]