CSE 484 / CSE M 584: Computer Security and Privacy

Cryptography
[Asymmetric Cryptography]

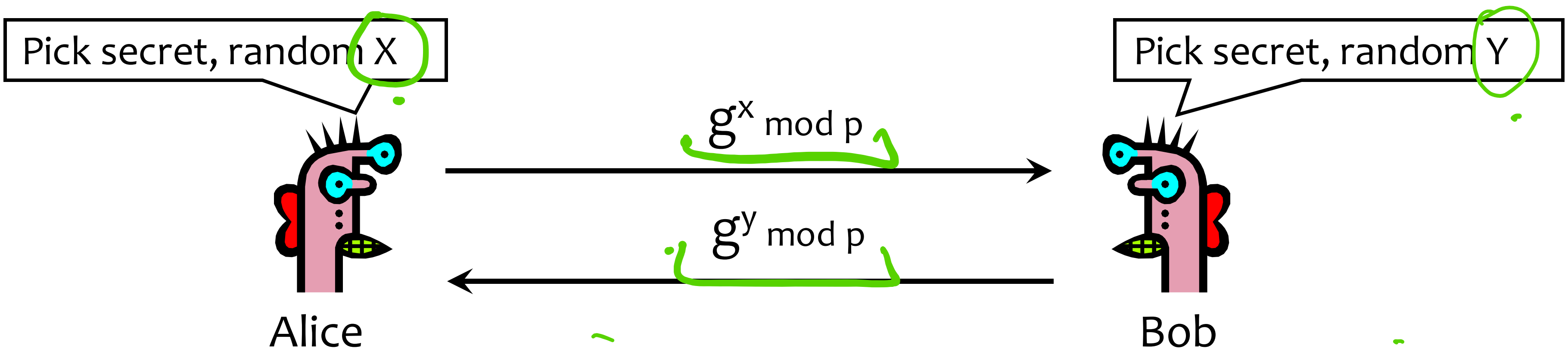
Autumn 2020

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Diffie-Hellman Key Exchange

- Alice and Bob never met and share no secrets
- Public info: p and g
  - p is a large prime, g is a **generator** of Z_p^*
    - $Z_p^*=\{1, 2 \ldots p\text{-}1\}$; a Z_p^* i such that $a=g^i$ mod p
    - Modular arithmetic: numbers "wrap around" after they reach p



Compute $k=(g^y)^x=g^{xy}$ mod p

Compute $k=(g^x)^y=g^{xy}$ mod p

Example Diffie Hellman Computation

public

$\boxed{p = 11 \quad g = 2}$

Alice

$x = 9$

Bob

$y = 4$

$g^x \bmod p$

$2^9 \bmod 11$

$\boxed{6}^4 \bmod 11$

$= 9 = K$

$g^y \bmod p$

$2^4 \bmod 11$

$\boxed{5}^9 \bmod 11$

$= 9$

$K = hash(9)$

$g^{xy} \bmod p$

$2^{9 \cdot 4} \bmod 11 = 9$

$hash(9) = K$

abc123 = AES key

Why is Diffie-Hellman Secure?

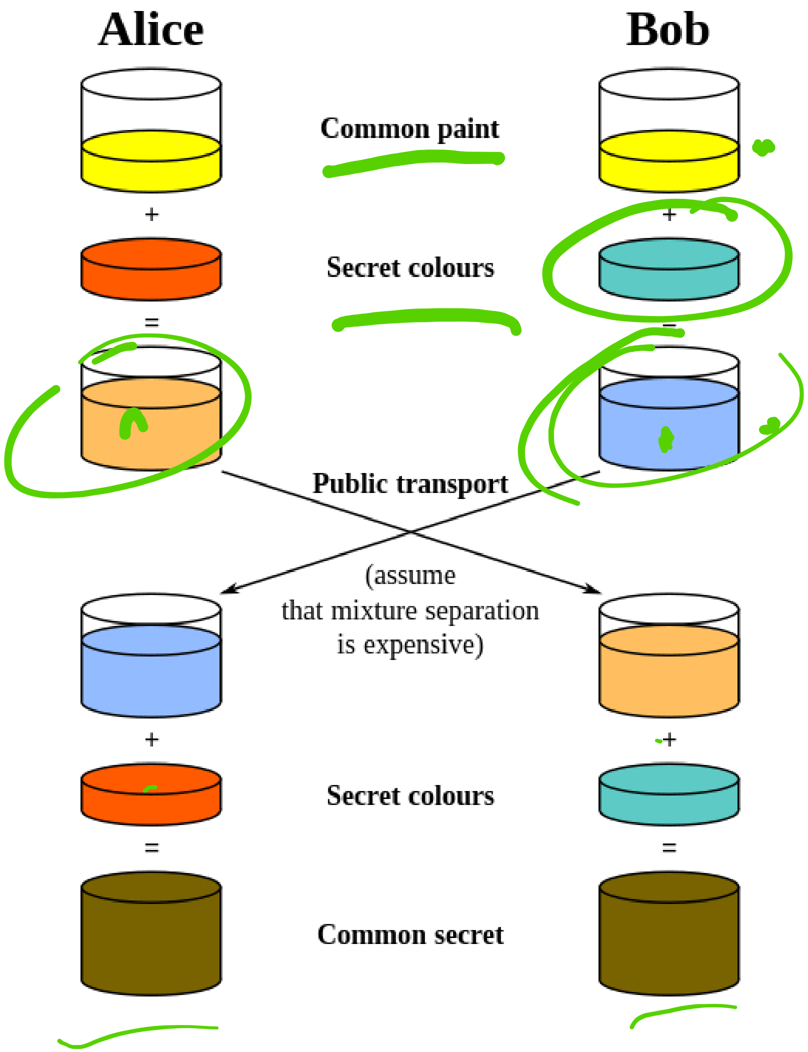
$5 = 2^{9} \bmod 11$

- Discrete Logarithm (DL) problem:
  given $g^x \bmod p$, it's hard to extract x
  - There is no known <u>efficient</u> algorithm for doing this
  - This is <u>not</u> enough for Diffie-Hellman to be secure!
- Computational Diffie-Hellman (CDH) problem:
  given g^x and g^y, it's hard to compute $g^{xy} \bmod p$

  5 6
  - … unless you know x or y, in which case it's easy
- Decisional Diffie-Hellman (DDH) problem:
  given g^x and g^y, it's hard to tell the difference between $g^{xy} \bmod p$ and $g^r \bmod p$ where r is random

"looks random"

Diffie-Hellman: Conceptually



Common paint: p and g

Secret colors: x and y

Send over public transport:
g^x mod p
g^y mod p

Common secret: g^{xy} mod p

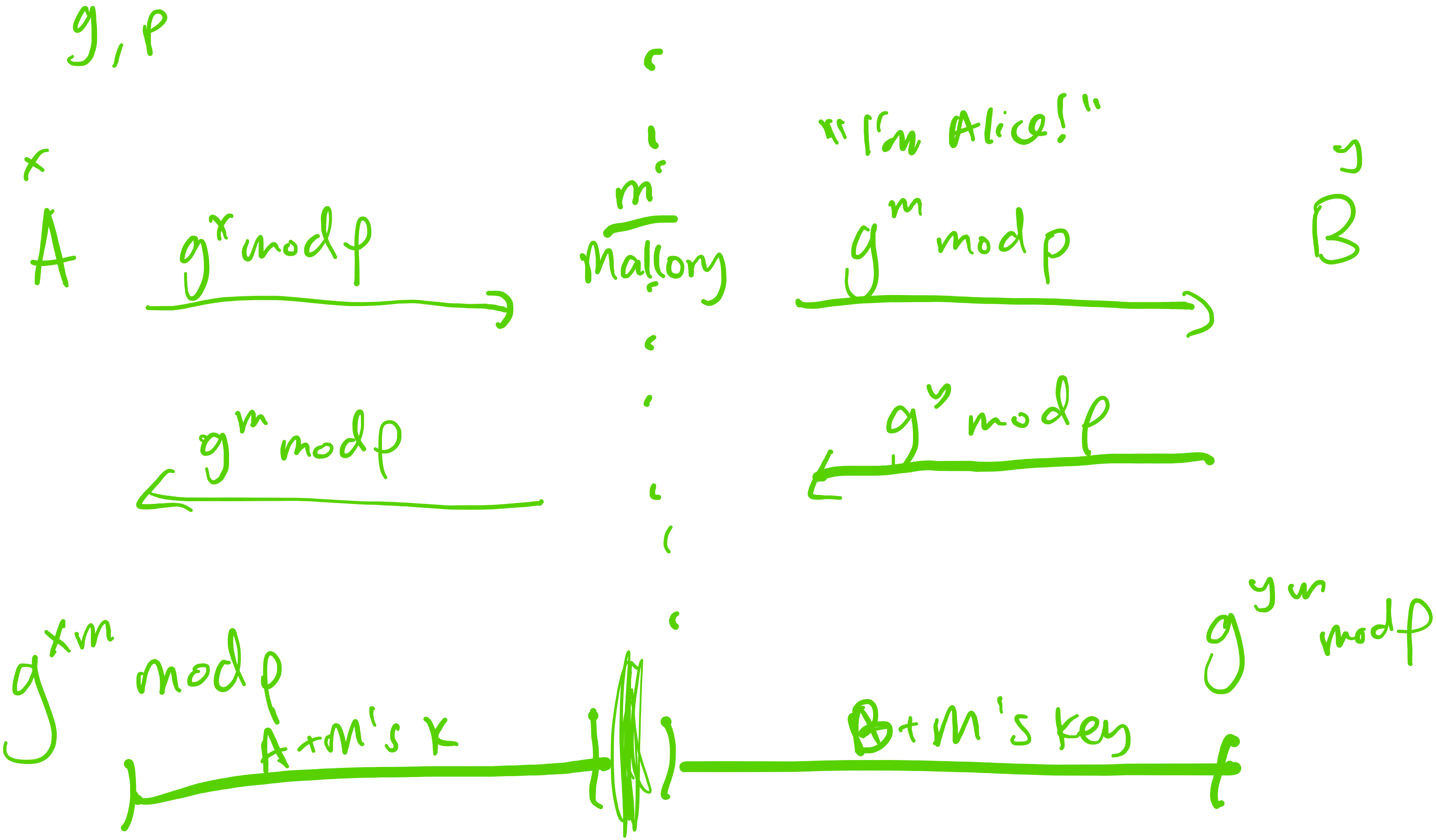
[from Wikipedia]

Diffie-Hellman Caveats

- Assuming DDH problem is hard (depends on choice of parameters!), Diffie-Hellman protocol is a secure key establishment protocol against passive attackers
  - Common recommendation:
    - Choose p=2q+1, where q is also a large prime
    - Choose g that generates a subgroup of order q in Z_p*
  - Eavesdropper can't tell the difference between the established key and a random value
  - In practice, often hash g^{xy} *mod p*, and use the hash as the key
  - Can use the new key for symmetric cryptography
- Diffie-Hellman protocol (by itself) does not provide authentication (against active attackers)
  - Person in the middle attack (also called "man in the middle attack")

"MITM"

Person In The Middle Attack



More on Diffie-Hellman Key Exchange

- Important Note:
  - We have discussed discrete logs modulo integers
  - Significant advantages in using elliptic curve groups
    - Groups with some similar mathematical properties (i.e., are "groups") but have better security and performance (size) properties

Stepping Back: Asymmetric Crypto

- We've just seen session key establishment
  - Can then use shared key for symmetric crypto
- Next: public key encryption
  - For confidentiality
- Then: digital signatures
  - For authenticity

block ciphers (AES-CBC)

hash | MAC

A

A_{pub}, A_{priv}

Encrypt(msg, B_{pub}) = C

B

$\boxed{B_{pub}}$, B_{priv}

Decrypt(C, B_{priv})

Requirements for Public Key Encryption

- Key generation: computationally easy to generate a pair (public key PK, private key SK)
- Encryption: given plaintext M and public key PK, easy to compute ciphertext $C=E_{PK}(M)$
- Decryption: given ciphertext $C=E_{PK}(M)$ and private key SK, easy to compute plaintext M
  - Infeasible to learn anything about M from C without SK
  - Trapdoor function: Decrypt(SK,Encrypt(PK,M))=M

Some Number Theory Facts

- Euler totient function $\overset{\text{phi}}{\varphi(n)}$ ($n \geq 1$) 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 $\varphi(ab) = \varphi(a)\,\varphi(b)$ if a & b are relatively prime

p, q primes

$\varphi(pq) = \varphi(p)\,\varphi(q)$

RSA Cryptosystem [Rivest, Shamir, Adleman 1977]

- Key generation:
  - Generate large primes p, q
    - Say, 1024 bits each (need primality testing, too)
  - Compute **n**=pq and Π(**n**)=(p-1)(q-1) $= \varphi(pq) = \varphi(p)\varphi(q)$
  - Choose small **e**, relatively prime to Π(n)
    - Typically, **e=3** or **e=2^{16}+1=65537**
  - Compute unique **d** such that ed ≡ 1 mod Π(n)
    - Modular inverse: d ≡ e^{-1} mod Π(n)
  - Public key = (e,n); private key = (d,n)
- Encryption of m: c = m^e mod n
- Decryption of c: c^d mod n = $(m^e)^d$ mod n = m

(p-1)(q-1)

How to compute?

brute force
d=1

- extended Euclidean alg

- Wolfram Alpha

Why RSA Decryption Works (FYI)

$e*d=1 \bmod phi(n)$, thus $e*d=1+k*phi(n)$ for some k

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

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

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

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

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

$= m \bmod n$

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

Why RSA Decryption Works (FYI)

- Decryption of c: $c^d \bmod n = (m^e \bmod n)^d \bmod n = (m^e)^d \bmod n = m$
- Recall $\mathbf{n}=pq$ and $\varphi(\mathbf{n})=(p-1)(q-1)$ and $ed \equiv 1 \bmod \varphi(n)$

- Chinese Remainder Theorem: To show $m^{ed} \bmod n \equiv m \bmod n$, sufficient to show:
  - $m^{ed} \bmod p \equiv m \bmod p$
  - $m^{ed} \bmod q \equiv m \bmod q$

- If $m \equiv 0 \bmod p \rightarrow m^{ed} \equiv 0 \bmod p$

- Else $m^{ed} = m^{ed-1}m = m^{k(q-1)(p-1)}m = m^{h(p-1)}m$ for some k, and h=k(q-1). Why? Recall how d was chosen and the definition of mod.
- Fermat Little Theorem: $m^{(p-1)h}m \equiv 1^h m \bmod p \equiv m \bmod p$

Why is RSA Secure?

$ed \equiv 1 \bmod \varphi(n)$

- RSA problem: given c, n=~~pq~~, and e such that $\gcd(e, \varphi(n))=1$, find m such that $m^e=c \bmod n$
  - In other words, recover m from ciphertext c and public key (n,e) by taking e^{th} root of c modulo n
  - There is no known efficient algorithm for doing this

$\varphi(n) = \varphi(p)\varphi(q)$

- Factoring problem: given positive integer n, find primes $p_1, \ldots, p_k$ such that $n=p_1^{e_1}p_2^{e_2}\ldots p_k^{e_k}$

- If factoring is easy, then RSA problem is easy (knowing factors means you can compute d = inverse of e mod (p-1)(q-1))
  - It may be possible to break RSA without factoring n -- but if it is, we don't know how

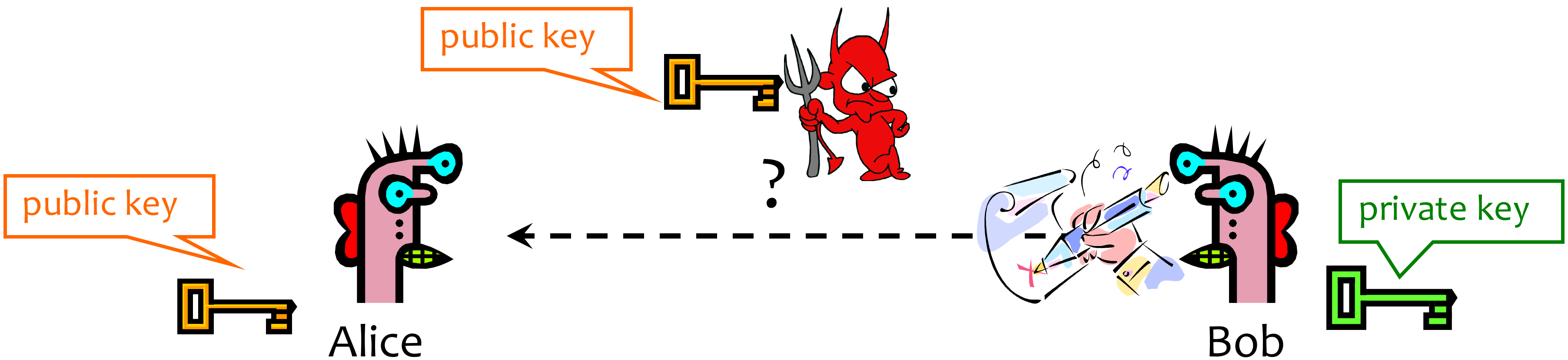
msg tag
msg' tag'
MAC: hash(msg, key)

RSA Encryption Caveats

- Encrypted message needs to be interpreted as an integer less than n
- Don’t use RSA **directly** for privacy – output is deterministic! Need to pre-process input somehow
- Plain RSA also does not provide integrity
  - Can tamper with encrypted messages

In practice, OAEP is used: instead of encrypting M, encrypt M□G(r) ; r□H(M□G(r))
- r is random and fresh, G and H are hash functions

Digital Signatures: Basic Idea



Given: Everybody knows Bob's public key
Only Bob knows the corresponding private key

Goal: Bob sends a "digitally signed" message

1. To compute a signature, must know the private key
2. To verify a signature, only the public key is needed

RSA Signatures

- Public key is **(n,e)**, private key is **(n,d)**
- To sign message m: $s = m^d \bmod n$
  - Signing & decryption are same **underlying** operation in RSA
  - It's infeasible to compute **s** on **m** if you don't know **d**
- To verify signature s on message m:

  verify that $s^e \bmod n = (m^d)^e \bmod n = m$
  - Just like encryption (for RSA primitive)
  - Anyone who knows **n** and **e** (public key) can verify signatures produced with d (private key)
- In practice, also need padding & hashing
  - Standard padding/hashing schemes exist for RSA signatures

DSS Signatures

- Digital Signature Standard (DSS)
  - U.S. government standard (1991, most recent rev. 2013)
- Public key: $(p, q, g, y=g^x \bmod p)$, private key: x
- Security of DSS requires hardness of discrete log
  - If could solve discrete logarithm problem, would extract x (private key) from $g^x \bmod p$ (public key)

- Again: We've discussed discrete logs modulo integers; significant advantages to using elliptic curve groups instead.

Cryptography Summary

- Goal: Privacy
  - Symmetric keys:
    - One-time pad, Stream ciphers
    - Block ciphers (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, SHA-256)
- Goal: Privacy and Integrity
  - Encrypt-then-MAC
- Goal: Authenticity (and Integrity)
  - Digital signatures (e.g., RSA, DSS)

Want More Crypto?

- Some suggestions:
  - CSE 490C (Rachel Lin):
    https://courses.cs.washington.edu/courses/cse490c/20au/
  - Stanford Coursera (Dan Boneh):
    https://www.coursera.org/learn/crypto