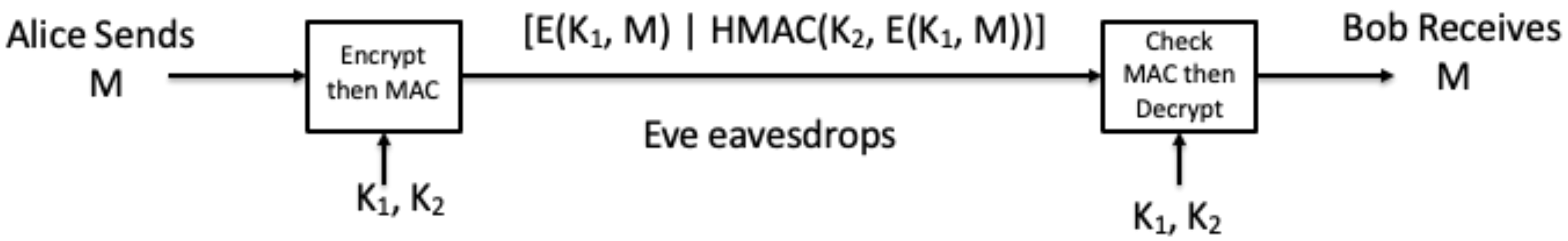
CSE 443: Introduction to Computer Security
Module: Asymmetric Cryptography

Prof. Syed Rafiul Hussain
Department of Computer Science and Engineering
The Pennsylvania State University

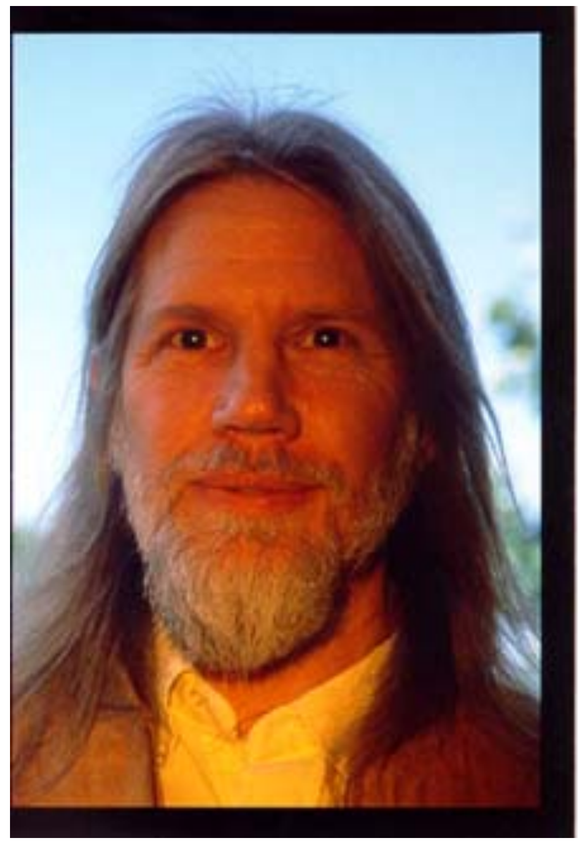
Recap of Symmetric Key Cryptography



- Without knowing K1, Eve can't read M
- Without knowing K2, Eve can't compute a valid MAC
- Problem
  - How do Alice and Bob securely share their keys?

Diffie-Hellman Key Agreement

- The DH paper really started the modern age of cryptography, and indirectly the security community
  - Negotiate a secret over an insecure media
  - E.g., "in the clear" (seems impossible)
  - Idea: participants exchange intractable puzzles that can be solved easily with additional information.

- Mathematics are very deep
  - Working in multiplicative group G
  - Use the hardness of computing discrete logarithms in finite field to make secure
  - Things like RSA are variants that exploit similar properties

Time for Revisiting Math

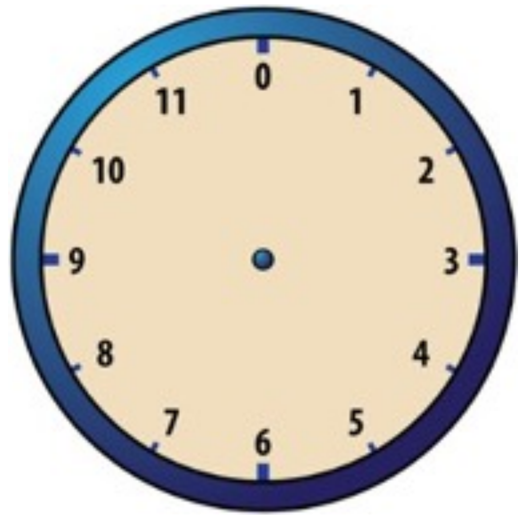
- **Divisibility:** an integer a divides b if b = ac for some integer c. This is denoted a | b
- **Prime:** an integer greater than 1 that is divisible by no positive integers other than 1 and itself
- **Greatest Common Divisor:** The GCD of two integers a and b is the largest integer n that divides both a and b
  - Denoted gcd(a, b) = n
  - Euclidean algorithm
- **Relatively prime:** Two integers a and b are relatively prime if gcd(a,b) = 1
-

Modular Arithmetic

- Clock-face arithmetic
  - Modulo 12

- Remainder arithmetic
  - Think of this in the context of integer division
  - Anything with the same remainder after division by the modulus n is considered equivalent (mod n)

  - What is 7 + 11 (mod 12)?
  - 2 * 8 (mod 12)?
  - 52 (mod 12)?



Modular Inverse

- For an integer e, the inverse modulo n is the integer d such that e*d = 1 (mod n)
  - Does not always exist!
- Examples
  - 6 * d = 1 (mod 7)
  - 5 * d = 1 (mod 9)
- Finding an inverse can be done efficiently

Euler's Totient Function

- Euler phi-function: for an integer n, $\phi(n)$ is defined as the number of positive integers that are:
  - Less than n
  - Relatively prime to n
- Multiplicative
  - For integers a and b such that gcd(a,b) = 1, $\phi(ab) = \phi(a)\,\phi(b)$
- For any prime p, $\phi(p) = p-1$
  - Example: Find $\phi(55)$

Diffie-Hellman Protocol

- For two participants p[1] and p[2]
- Setup: We pick a prime number p and a base g ($<p$)
  - This information is public
  - E.g., $p=13$, $g=4$
- Step 1: Each principal picks a private value x ($<p-1$)
- Step 2: Each principal generates and communicates a new value

$$y = g^x \bmod p$$

- Step 3: Each principal generates the secret shared key z

$$z = y^x \bmod p$$

Where y is the value received from the other party.

A protocol run ...

p=17, g=6

Step 1)

Alice picks x=4

Bob picks x=5

Step 2)

Alice's y = 6^4 mod 17 = 1296 mod 17 = 4

Bob's y = 6^5 mod 17 = 7776 mod 17 = 7

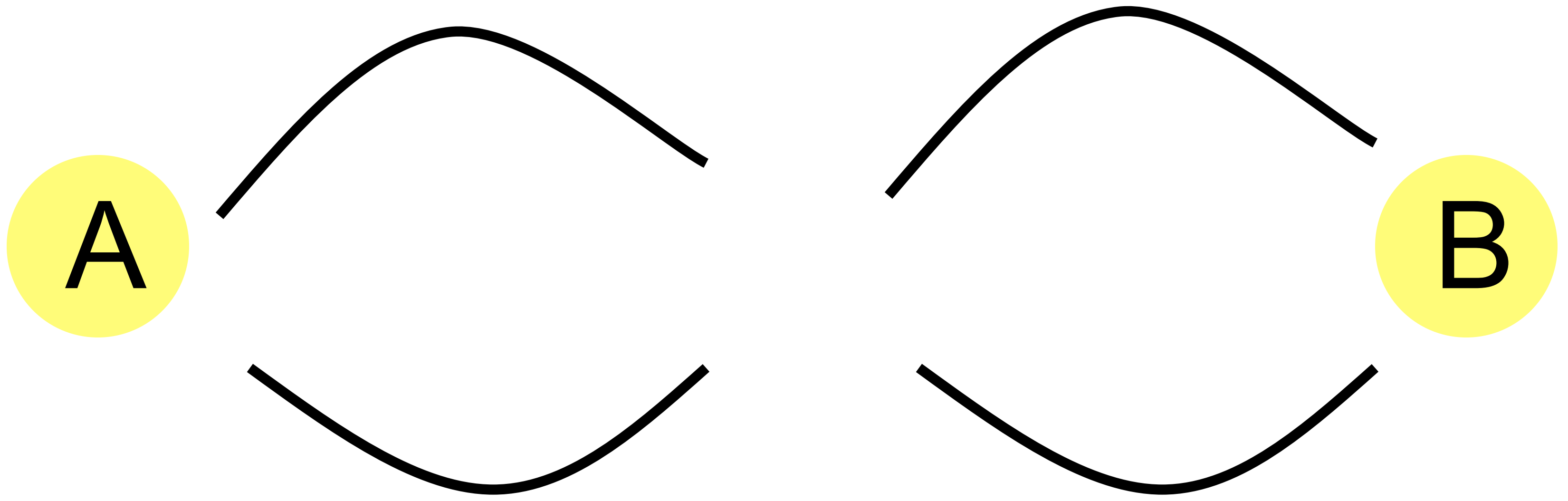
Step 3)

Alice's z = 7^4 mod 17 = 2401 mod 17 = *4*

Bob's z = 4^5 mod 17 = 1024 mod 17 = *4*

Attacks on Diffie-Hellman

- This is key exchange, not authentication.
  - You really don't know anything about who you have exchanged keys with
  - The man in the middle …



  - Alice and Bob think they are talking **directly** to each other, but Mallory is actually performing two separate exchanges
- You need to have an authenticated DH exchange
  - The parties sign the exchanges (more or less)
  - See Schneier for a intuitive description

Public Key Cryptography

- Public Key cryptography
  - Each key pair consists of a public and private component: k^+ (public key), k^- (private key)

$$\mathbf{D}(k^+, \mathbf{E}(k^-, p)) = p$$

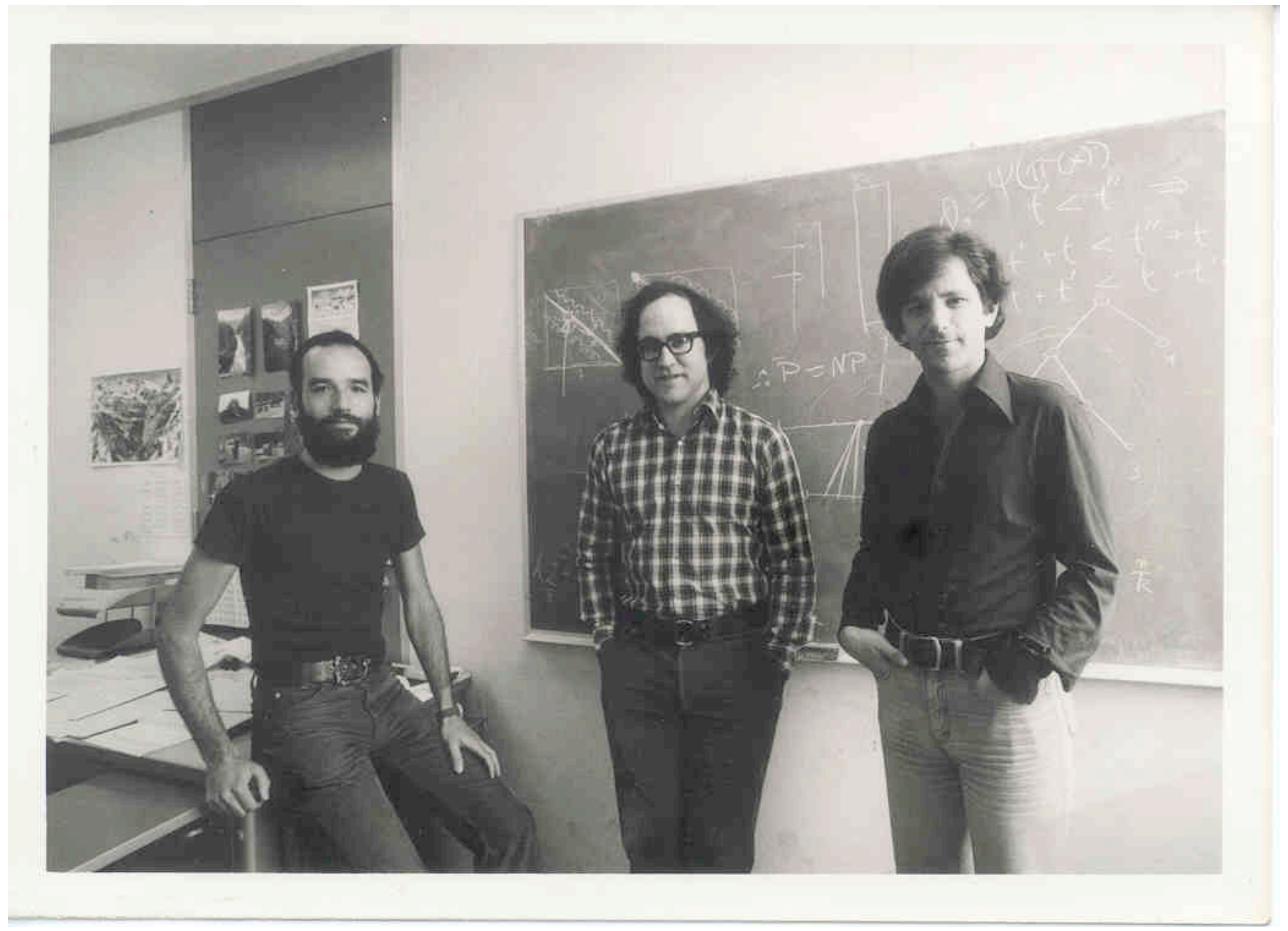
$$\mathbf{D}(k^-, \mathbf{E}(k^+, p)) = p$$

- Public keys are distributed (typically) through public key certificates
  - Anyone can communicate secretly with you if they have your certificate
  - E.g., SSL-based web commerce

RSA (Rivest, Shamir, Adelman)

- A dominant public key algorithm
  - The algorithm itself is conceptually simple
  - Why it is secure is very deep (number theory)
  - Use properties of exponentiation modulo a product of large primes

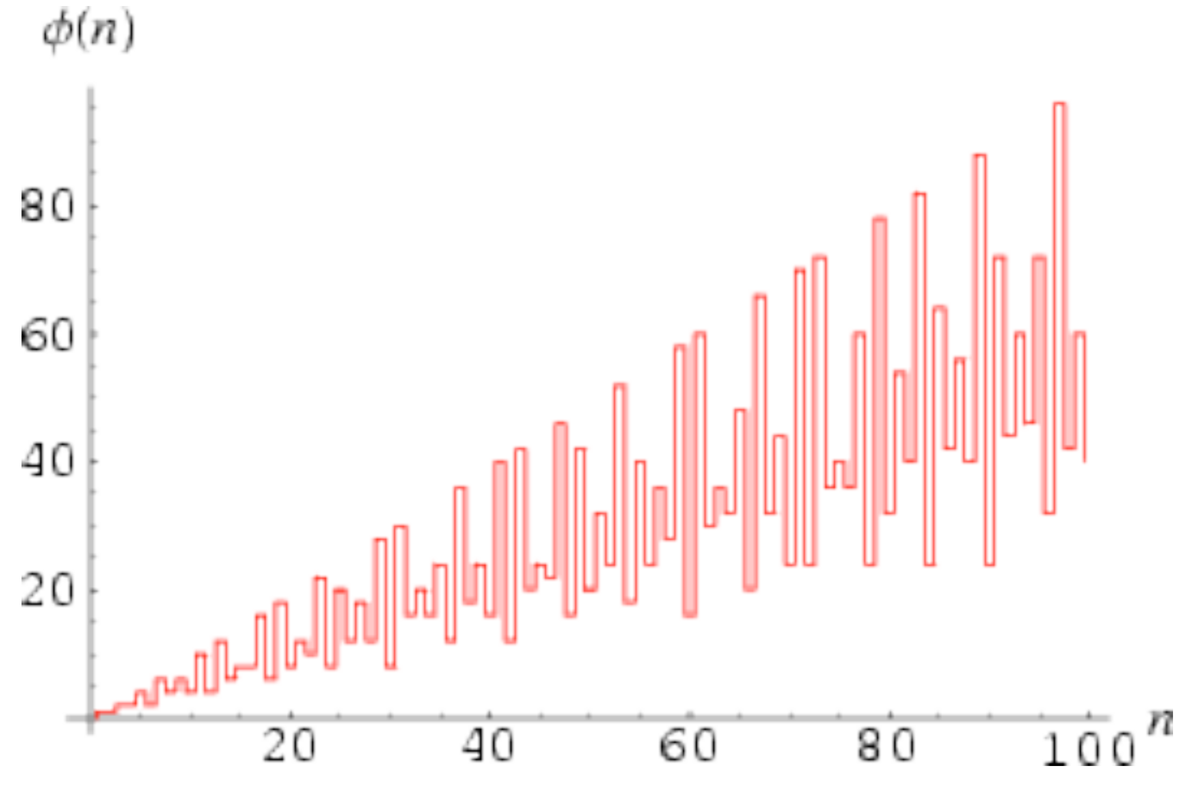
"A method for obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, Feb., 1978 21(2) pages 120-126.



RSA Key Generation

- Pick two large primes p and q
- Calculate n = pq
- Pick e such that it is relatively prime to phi(n) = (q-1)(p-1)
  - "Euler's Totient Function"
- d ~= e^{-1} mod phi(n)
  or

  de mod phi(n) = 1



1. p=3, q=11

2. n = 3*11 = 33
3. phi(n) = (2*10) = 20
4. e = 7 | GCD(20,7) = 1

"Euclid's Algorithm"

5. d * 7 mod 20 = 1
   d = 3

RSA Encryption/Decryption

PennState

- Public key k^+ is {e,n} and private key k^- is {d,n}
- Encryption and Decryption

$$E(k+,P) : \text{ciphertext} = \text{plaintext}^e \bmod n$$

$$D(k-,C) : \text{plaintext} = \text{ciphertext}^d \bmod n$$

- Example
  - Public key (7,33), Private Key (3,33)
  - Data "4" (encoding of actual data)

  - $E(\{7,33\},4) = 4^7 \bmod 33 = 16384 \bmod 33 = 16$
  - $D(\{3,33\},16) = 16^3 \bmod 33 = 4096 \bmod 33 = 4$

Encryption using private key …

- Encryption and Decryption

$$E(k^-,P) : \text{ciphertext} = \text{plaintext}^d \bmod n$$

$$D(k^+,C) : \text{plaintext} = \text{ciphertext}^e \bmod n$$

- E.g.,
  - $E(\{3,33\},4) = 4^3 \bmod 33 = 64 \bmod 33 = 31$
  - $D(\{7,33\},19) = 31^7 \bmod 33 = 27{,}512{,}614{,}111 \bmod 33 = 4$

- Q: Why encrypt with private key?

Why does RSA work?

- Difficult to find $\phi(n)$ or d using only e and n
- Finding d equivalent difficulty to factoring p*q
  - Classical problem worked on for centuries; no known reliable fast method
  - Example: Took 18 months to factor a 200 digit number into its 2 prime factors
- It is feasible to encrypt and decrypt because
  - It is possible to find large primes
  - It is possible to find coprimes and their inverses
  - Modular exponentiation is feasible

{e,n} is public information
If you could factor n into p*q, then
Could compute $\phi(n) = (p-1)(q-1)$
Could compute $d = e^{-1} \bmod \phi(n)$
Would know the private key {d,n}!

“Textbook” RSA and Security

- What we’ve just seen is known as “textbook” RSA
- RSA must be used with proper padding to prevent certain attacks (including chosen plaintext attacks)
- As we’ve used it here, NO integrity!
- RSA keys can be of any length
  - The current recommendation is that important keys should be at least 2048-bits in length
  - 1024 bit keys are ok for most uses, but you should feel nervous about them

Attacks Against RSA

- Brute force: try all possible private keys
  - Can be defeated by using a large enough key space (e.g., 1024 bit keys or larger)
- Mathematical attacks
  - Factor n (possible for special cases of n)
  - Determine d directly from e, without computing $\phi(n)$
- At least as difficult as factoring n

Probable-Message Attack (using {eon})

- Encrypt all possible plaintext messages
- Try to find a match between the ciphertext and one of the encrypted messages
  - Only works for small plaintext message sizes
- Solution: pad plaintext message with random text before encryption
- PKCS#1v1 specifies this padding format:
-

Hybrid Cryptosystems

- In practice, public-key cryptography is used to secure and distribute session keys.
- These keys are used with symmetric algorithms for communication
- Sender generates a random session key, encrypts it using receiver's public key and sends it
- Receiver decrypts the message to recover the session key
- Both encrypt/decrypt their communications using the same key
- Key is destroyed in the end

$$\text{Alice} \xrightarrow{[E(K_{B+}, k) \mid E(k, m)]} \text{Bob}$$

Bob's pubic / private key pair is (B^+, B^-)

k is the session key, sometimes called the ephemeral key

Digital Signatures

- A digital signature serves the same purpose as a real signature
  - It is a mark that only the sender can make
  - Other people can easily recognize it belonging to the sender
- Digital signatures must be:
  - Unforgeable: If Alice signs message M with signature S, it is impossible for someone else to produce the pair (M, S).
  - Authentic: If Bob receives the pair (M, S) and knows Alice's public key, he can check ("verify") that the signature is really from Alice

How can Alice sign a digital document?

- Digital document: M
- Since RSA is slow, hash M to compute digest h(M)
- Signature: $Sig_{k-}(M) = E_{k-}(h(M)) = (h(M))^d \bmod n$
  - Since only Alice knows k-, only she can create the signature
- To verify: Verify $(M, Sig_{k-}(M))$
  - Bob computes h(M) and compares it with $D_{k+}(Sig_{k-}(M))$
  - Bob can compute $D_{k+}(Sig_{k-}(M))$ since he knows k+ (Alice's public key)
  - If and only if they match, the signature is verified (otherwise, fails)
- Note: M is not computable directly from $Sig_{k-}(M)$

Alice's public / private key pair is (A^+, A^-)

Alice $\xrightarrow{[E(K_{B+}, k) \mid E(k, m \mid Sig(K_{A-}, m))]}$ Bob

Bob's pubic / private key pair is (B^+, B^-)

Birthday Attack and Signatures

- Since signatures depend on hash functions, they also depend on the hash function's collision resistance
- Don't use MD5 or SHA1
-

Properties of digital signature

- No forgery possible: No one can forge a message that is purportedly from Alice
- Authenticity check: If you get a signed message you should be able to verify that it's really from Alice
- No alteration/Integrity: No party can undetectably alter a signed message
- Provides authentication, integrity, and non-repudiation (cannot deny having signed a signed message)
-

Non-Repudiation

- Which offers non-repudiation, and why?
  - HMAC: [m | HMAC(k, m)]
  - Digital Signature: [m | Sigk-(m)]

The symmetric/asymmetric key

- Symmetric (shared) key systems
  - Efficient (Many MB/sec throughput)
  - Difficult key management
    - Kerberos
    - Key agreement protocols
- Asymmetric (public) key systems
  - Slow algorithms (so far …)
  - Easy key management
    - PKI - public key infrastructures
    - Webs of trust (PGP)

Important principles

- Don’t design your own crypto algorithm
  - Use standards whenever possible
- Make sure you understand parameter choices
- Make sure you understand algorithm interactions
  - E.g. the order of encryption and authentication
    - Turns out that authenticate then encrypt is risky
- Be open with your design
  - Solicit feedback
  - Use open algorithms and protocols
  - Open code? (jury is still out)

Common issues that lead to pitfalls

- Generating randomness
- Storage of secret keys
- Virtual memory (pages secrets onto disk)
- Protocol interactions
- Poor user interface
- Poor choice of key length, prime length, using parameters from one algorithm in another

Review: secret vs. public key crypto.

- Secret key cryptography
  - Symmetric keys, where A single key (k) is used is used for E and D

$$D(\ k, E(k, p)\ ) = p$$

- All (intended) receivers have access to key
- Note: Management of keys determines who has access to encrypted data
  - E.g., password encrypted email
- Also known as symmetric key cryptography

- Public key cryptography

– Each key pair consists of a public and private component:

k^+ (public key), k^- (private key)

$$D(\ k^-, E(k^+, p)\ ) = p$$

$$D(\ k^+, E(k, ^-p)\ ) = p$$

- Public keys are distributed (typically) through public key certificates

– Anyone can communicate secretly with you if they have your certificate

– E.g., SSL-base web commerce

A really good book on the topic

- The Code Book, Simon Singh, Anchor Books, 1999