Key Distribution/Agreement

- **Key Distribution** is the process where we assign and transfer keys to a participant
  - Out of band (e.g., passwords, simple)
  - During authentication (e.g., Kerberos)
  - As part of communication (e.g., skip-encryption)

- **Key Agreement** is the process whereby two parties negotiate a key
  - 2 or more participants

- Typically, key distribution/agreement this occurs in conjunction with or after authentication.
  - However, many applications can pre-load 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
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
  \[ y = g^x \mod p \]
- Step 3: Each principal generates the secret shared key $z$
  \[ z = y^x \mod p \]
- Perform a neighbor exchange.
Attacks on Diffie-Hellman

• This is key agreement, 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)

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

• 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
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

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”
• \( d^{-1} = e \mod \phi(n) \) or \( de \mod \phi(n) = 1 \)

1. \( p=3, \ q=11 \)
2. \( n = 3*11 = 33 \)
3. \( \phi(n) = (2*10) = 20 \)
4. \( e = 7 \mid \text{GCD}(20,7) = 1 \)
5. “Euclid’s Algorithm”
   \( d^{-1} = e \mod \phi(n) \)
   \( 1 = 7d \mod 20 \)
   \( d = 3 \)
RSA Encryption/Decryption

- Public key $k^+$ is \{e,n\} and private key $k^-$ is \{d,n\}
- Encryption and Decryption
  \[ E(k^+, P) : \text{ciphertext} = \text{plaintext}^e \mod n \]
  \[ D(k^-, C) : \text{plaintext} = \text{ciphertext}^d \mod n \]
- Example
  - Public key (7,33), Private Key (3,33)
  - Data “4” (encoding of actual data)
  \[ E(7,33,4) = 4^7 \mod 33 = 16384 \mod 33 = 16 \]
  \[ D(3,33,16) = 16^3 \mod 33 = 4096 \mod 33 = 4 \]
Encryption using private key ... 

• Encryption and Decryption
  \[ E(k^-,P) : \text{ciphertext} = \text{plaintext}^d \mod n \]
  \[ D(k^+,C) : \text{plaintext} = \text{ciphertext}^e \mod n \]

• E.g.,
  \[ E\{3,33\},4) = 4^3 \mod 33 = 64 \mod 33 = 31 \]
  \[ D\{7,33\},19) = 31^7 \mod 33 = 27,512,614,111 \mod 33 = 4 \]

• Q: Why encrypt with private key?
Digital Signatures

- Models physical signatures in digital world
  - Association between private key and document
  - ... and indirectly identity and document.
  - Asserts that document is authentic and non-reputable

- To sign a document
  - Given document d, private key k-
    - Signature $S(d) = E(k^{-1}h(d))$

- Validation
  - Given document d, signature S(d), public key k+
    - Validate $D(k^+, S(d)) = H(d)$
Cryptanalysis and Protocol Analysis

- Cryptographic Algorithms
  - Complex mathematical concepts
  - May be flawed
  - What approaches are used to prove correct/find flaws?

- Cryptographic Protocols
  - Complex composition of algorithms and messages
  - May be flawed
  - What approaches are used to prove correct/find flaws?
RSA Exponent Problems

• Small Private Exponent
  • Speeds decryption time

• However, Known Attacks Exist on Small Private Keys
  • Due to Mike Wiener, can recover private key
  • Result: If N is 1024 bits, d of private key must be at least 256 bits
  • Some workarounds are known (e.g., based on Chinese Remainder Theorem), but not proven secure

• Small Public Exponent
  • Speed signature verification time
  • Smallest possible value is 3, but recommend $2^{16} + 1$
  • Can recover M encrypted with multiple, small public keys
  • Can recover private key from small public + bits of private
Timing Attacks

• Use the timing behavior of system to extract secret

• Suppose a smartcard stores your private key
  ▸ By precisely measuring the time it takes to perform private key ops, we can recover the key
  ▸ Due to Kocher
  ▸ At most 2^n operations required, where n is the number of bits in the key

• Attack summary
  ▸ Adversary asks smartcard to generate signatures on several messages
  ▸ Recover one bit at a time starting with least significant
  ▸ Compare times to those measured offline

• Solution: blinding
Power Analysis Attacks

- Also, Discovered by Kocher
  - Power usage is higher than normal in these computations
  - Measure the timing of high power consumption
- Simple Power Analysis
  - Direct interpretation of power measurements
  - Reveals instructions executions
  - Some crypto ops may be sensitive to data, e.g., DES S-boxes
- Differential Power Analysis
  - Statistical analysis of power data correlations
- Solution: Must change the code!
Power and Timing

• What is the threat model in power/timing attacks?

• How does this conflict with the trust model?

• What is the vulnerability?