Key Distribution

- An important part of encryption is key management
  - Sender and receiver need the same key
    
    \[ \text{How can sender and receiver agree on a key?} \]

    \[ \text{If the key is compromised how can two distant parties agree on a new key?} \]

    - Secret key encryption is especially vulnerable

- Need methods for distribution keys reliably
Quantum Cryptography

- We know one-time-pad is provably secure, but it has problems
  *What?*

- Quantum cryptography *may* provide a plausible one-time-pad
- Based on the fact light comes from photons
- Light from a polarized filter is polarized in the filter’s axis
- If second filter applied, intensity after second filter is proportional to cosine of axis angles
  *What does this have to do with cameras? Or Jellybeans?*

Generating a One-Time Pad

- Alice has two sets of polarizing filters (four filters total)
  - **Rectilinear basis** consists of a vertical and a horizontal filter
    \[
    \begin{array}{c}
    \vert \\
    \end{array}
    \begin{array}{c}
    \_ \\
    \end{array}
    \]
  - **Diagonal basis** is the same except rotated 45 degrees
    \[
    \begin{array}{c}
    \diagup \\
    \end{array}
    \begin{array}{c}
    \diagdown \\
    \end{array}
    \]
  - *Bob has the same sets of filters*
- Alice assigns 0 or 1 to each filter in each set
  - Rectilinear basis: \[
  \begin{array}{c}
  \vert \\
  \end{array}
  \begin{array}{c}
  \_ \\
  \end{array}
  \] is 0, \[
  \begin{array}{c}
  \_ \\
  \end{array}
  \begin{array}{c}
  \vert \\
  \end{array}
  \] is 1
  - Diagonal basis: \[
  \begin{array}{c}
  \diagup \\
  \end{array}
  \begin{array}{c}
  \diagdown \\
  \end{array}
  \] is 0, \[
  \begin{array}{c}
  \diagdown \\
  \end{array}
  \begin{array}{c}
  \diagup \\
  \end{array}
  \] is 1
- Alice generates a one-time pad, for example using a RNG
  *Oh crap....*
• Alice sends the one-time pad using from the two basis at random
  – One photon polarized for the bit she wants to transmit

<table>
<thead>
<tr>
<th>Bit number</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(e)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(f)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(g)</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>?</td>
<td>1</td>
<td>x</td>
<td>?</td>
<td>?</td>
<td>0</td>
<td>x</td>
<td>?</td>
<td>x</td>
</tr>
</tbody>
</table>

- What Alice sends
- Bob's bases
- What Bob gets
- Correct basis?
- One-time pad
- Trudy's bases
- Trudy's pad

• Bob does not know which bases, randomly picks one per photon
  – If he picks correct, then he gets the correct bit
  – If he pick incorrectly, then the photon jumps to his filter or the polarization perpendicular \( \text{either way its wrong} \)
  – Therefore, some bits are correct while others are wrong

  \text{How does Bob know?}

• Bob tells Alice which bases he used, she tells which are right
  – This is done in plaintext \text{What about Trudy?}

  \text{The correct bits form the one-time pad}

• \text{Problem solved... which problem?}
Pioneering Public Key

- Key distribution has been the weak link in most cryptosystems
  - Stealing the key makes any system worthless
  - Keys must be protected from theft, yet available
- In the 1970’s, Ralph Merkle invented a secret key exchange system
  - Can be used over public lines
- First attempt at distributing keys, send a key database
  - Suppose Alice makes 1,000,000 secret keys
  - Stored in a database, each key has a serial number (identifier)
  - Alice sends Bob the database and tells the serial number to use for exchanging the secret key
    - Why isn’t this secure?

- Second attempt, send an encrypted key database (key and number)
  - Alice encrypts the database tells the encryption method
  - Bob is forced to crack the database
  - Once cracked, Bob tells Alice the serial number to use to exchange the secret key
    - Is this secure, at least more secure?

- Third attempt, send database with keys separately encrypted
  - Alice encrypts each key in the database separately
  - Bob cracks one key then tells Alice the serial number
  - Once cracked, Bob tells the serial number to use
    - Is this secure, at least more secure?
Public Key Algorithms

- In 1976 two researchers at Stanford (Diffie and Hellman) proposed a radical new cryptosystem
  - Encryption and decryption keys are different
    - What about all the methods described to-date?
  - The decryption key could not be derived from encryption key
  - The encryption key can be made public while the decryption key is held private (hence the name public key encryption)
- The catch will be to find algorithms to support this behavior

How would it work?, assume two people Alice and Bob
- Alice’s key $k_{pub}$ would be made public, while her private key $k_{pri}$ is kept secret
- If Bob needed to send a secret message to Alice then
  $$e_{k_{pub}}(p) = c$$
- Once $c$ received by Alice, she computes
  $$d_{k_{pri}}(c) = p$$

How would Alice send a secret message to Bob?
RSA

- An early public key method was RSA
  - Developed developed by Rivest, Shamir, and Adleman in 1977
  - Most widely used public-key method
- System is a block cipher based on principles in number theory
  - Modular inverse and Fermat’s Little Theorem
- Following steps identify the keys
  1. Choose two large prime numbers\(^a\), \(p\) and \(q\)
  2. Compute \(n = p \times q\) and \(z = (p - 1) \times (q - 1)\)
  3. Choose a number relatively prime to \(z\) and call it \(d\)
  4. Find \(e\) such that \(e \times d \pmod{z} = 1\)

\(^a\)Yo, we typically use \(p\) to represent "plaintext" but we’ll use \(m\) to represent "plaintext" or "message". The variable \(p\) will be used to represent a prime number. I hope this serves to confuse you. Thank You – Management

- For example...
  1. Let \(p = 7\) and \(q = 17\) (should be much larger than this...)
  2. \(n = 7 \times 17 = 119\) and \(z = (7 - 1) \times (17 - 1) = 96\)
  3. Choose \(d\) relatively prime to \(96\), \(d = 5\) works
  4. Find \(e\) such that \((e \times 5) \pmod{96} = 1\), \(e = 77\) works

\[77 \times 5 = 385 = 4 \times 96 + 1\]

- Given these numbers divide the plaintext into blocks, \(0 \leq m < n\) (therefore the message must be positive and less than \(n\))
  - To encrypt \(c = m^e \pmod{n}\)
  - To decrypt \(m = c^d \pmod{n}\)
  - Public key is \([e, n]\) and the private key is \([d, n]\)
• For example, assume the message is 9 (therefore, \( m = 9 \))
  
  – Encrypted \( e = 9^5 \pmod{119} = 25 \)
  
  – Decrypted \( m = 25^{77} \pmod{119} = 9 \)

• Another example, public key is \([3, 33]\) and the private key is \([7, 33]\)

<table>
<thead>
<tr>
<th>Message (m)</th>
<th>Ciphertext (c)</th>
<th>After decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbolic</td>
<td>Numeric</td>
<td>(m^2)</td>
</tr>
<tr>
<td>S</td>
<td>19</td>
<td>6859</td>
</tr>
<tr>
<td>U</td>
<td>21</td>
<td>9261</td>
</tr>
<tr>
<td>Z</td>
<td>26</td>
<td>17576</td>
</tr>
<tr>
<td>A</td>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>14</td>
<td>2744</td>
</tr>
<tr>
<td>N</td>
<td>14</td>
<td>2744</td>
</tr>
<tr>
<td>E</td>
<td>05</td>
<td>125</td>
</tr>
</tbody>
</table>

– Since the primes are so small, each cipher block can only be a single character (less than 33)

---

**RSA Security**

• Security is based on the difficulty of factoring large numbers
  
  – Assume the attacker could factor \( n \) (which is public)
  
  – Then could find \( p \) and \( q \), and then \( z \)
  
  – With \( z \) and \( e \), \( d \) can be found using Euclid’s algorithm

• Fortunately, factoring large numbers is difficult...
  
  – Rivest states that factoring a 200 digit number requires 4 billion years of computer time
  
  – As computers become faster, just use larger \( p \) and \( q \) values

  *What is an alternative attack on factoring?*

*Early versions of Netscape were insecure although they used RSA, why?*
Some Background for RSA

• Fermat discovered that given a prime $p$ and any positive integer $m$

$$m^{p-1} \pmod{p} = 1$$

– For example if $m = 7$ and $p = 11$, then

$$7^{11-1} \pmod{11} = 7^{10} \pmod{11} = 1$$

• Euler discovered a similar relationship, if $p$ and $q$ are primes, then let

$$n = p \cdot q$$

$$m^{(p-1)(q-1)} \pmod{n} = 1$$

Note, $m$ and $n$ must also be relatively prime

– For example if $m = 38$, $p = 11$, and $q = 5$, then $n = p \cdot q = 55$ and

$(p-1)(q-1) = 40$, then

$$38^{40} \pmod{55} = 1$$

• Let’s change Euler’s rule by multiplying both sides by $m$,

$$m^{(p-1)(q-1)+1} \pmod{n} = m$$

– So we can raise an integer $m$ to the power $[(p-1)(q-1) + 1]$, then modulo $n$ and get $m$ back
Why RSA works... Kinda

- Consider encrypting a message \( m \),
  - Select two prime numbers \( p \) and \( q \)
  - Find two relatively prime numbers \( e \) and \( d \) such that
    \[ e \cdot d = (p - 1)(q - 1) + 1 \]
  - Encrypt, \( m^e \pmod{n} = c \)
  - Decrypt, \( c^d \pmod{n} = m \)

- Why does this work? Decrypting \( c \) can be rewritten as
  \[ c^d \pmod{n} = (m^e)^d \pmod{n} = m^{e \cdot d} \pmod{n} \]
  recall \( e \cdot d = (p - 1)(q - 1) + 1 \), so let’s substitute
  \[ m^{(p-1)(q-1)+1} \pmod{n} = m \]
  based on Euler’s discovery (well kinda, I did take a shortcut or two)

RSA Security

<table>
<thead>
<tr>
<th>What the attacker wants to do</th>
<th>How to accomplish it</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find ( m )</td>
<td>Must know ( d )</td>
</tr>
<tr>
<td>Find ( d )</td>
<td>Must know ( e ) and ( (p - 1)(q - 1) )</td>
</tr>
<tr>
<td>( e ) is known, so find ( (p - 1)(q - 1) )</td>
<td>Must know ( p ) and ( q )</td>
</tr>
<tr>
<td>Find ( p ) and ( q )</td>
<td>Must factor ( n )</td>
</tr>
</tbody>
</table>

- Security is based on the difficulty of factoring \( n \), a large number
Other Public-Key Algorithms

- The first public key algorithm was by Merkle and Hellman, 1978
  - Owner has a large number of objects, each with a weight
  - Owner encodes the message by selecting a set of objects
  - Total weight and the complete list of objects made public
    
    *What is this based on?*

- Merkle thought the system was secure and offered $100 to break
  - Shamir broke it, became $100 richer

- Merkle made changes offered $1000 to break
  - Rivest broke it, became $1000 richer

Management of Public Keys

- An important question is *how do you obtain public keys?*

*Publish on your web-site?*

```
1. GET Bob's home page
2. Fake home page with $E_T$
3. $E_T$(Message)
4. $E_T$(Message)
```

*What about a dedicated key distribution center?*
Certificate Authority

- Certificate Authority (CA) does not distribute keys
  - Certifies the public key belongs to people, companies, etc...

- Assume Pluf wants others to communicate with him securely
  - Pluf goes to the CA with his public key, passport, etc...
  - CA issues certificate, signed with private key and hashed

```
I hereby certify that the public key 1999AB8FE303857393CAB9478294
belongs to Nirre Pluf, 932 Manchester Hall, Winston-Salem, NC 27109
SHA-2 hash of the above signed with the CA’s private key
```

- Pluf posts the certificate on his web-site

*What is the difference? The advantage?*

---

X.509

- X.509 is a standard format for certificates
  - Approved by the ITU and adopted by the IETF (RFC 3280)

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Which version of X.509</td>
</tr>
<tr>
<td>Serial number</td>
<td>This number plus the CA’s name uniquely identifies the certificate</td>
</tr>
<tr>
<td>Signature alg.</td>
<td>The algorithm used to sign the certificate</td>
</tr>
<tr>
<td>Issuer</td>
<td>X.500 name of the CA</td>
</tr>
<tr>
<td>Validity period</td>
<td>The starting and ending times of the validity period</td>
</tr>
<tr>
<td>Subject name</td>
<td>The entity whose key is being certified</td>
</tr>
<tr>
<td>Public key</td>
<td>The subject’s public key and the ID of the algorithm using it</td>
</tr>
<tr>
<td>Issuer ID</td>
<td>An optional ID uniquely identifying the certificate’s issuer</td>
</tr>
<tr>
<td>Subject ID</td>
<td>An optional ID uniquely identifying the certificate’s subject</td>
</tr>
<tr>
<td>Extensions</td>
<td>Many extensions have been defined</td>
</tr>
<tr>
<td>Signature</td>
<td>The certificate’s signature (signed by the CA’s private key)</td>
</tr>
</tbody>
</table>
Public Key Infrastructure

- PKI consists of components required for public key distribution
  - Assume **trust anchors** exist and known in advance
    What are the trust anchors?

  - Several models exist to provide a scalable and secure solution

- **Monopoly model**
  - Only one CA and everyone knows the public key What are the advantages and disadvantages?

- **Monopoly plus Registration Authorities (RA)**
  - CA selects other organizations as RA’s to check identities and obtain/vouch for public keys What is the advantage over a monopoly? Disadvantages?

- **Oligarchy** *(currently used by browsers)*
  - There is a set of trust anchors, can be added and deleted
    What is the advantage and disadvantages?

- **Anarchy** *(currently used by PGP)*
  - Users are responsible for configuring trust anchors How?

  - Some web-sites have databases you can deposite certificates
    What is the advantage over a monopoly? Disadvantages?
Hierarchical PKI

- Use a hierarchy of Regional Authorities (RA) and CA’s
  - Root authorizes RA, who authorize CA’s

- Suppose Alice needed Bob’s public key
  - She gets a certificate signed by CA-5, contains Bob’s key
  - Alice does not know CA-5 and asks for credentials

  - CA-5 responds with certificate signed by RA-2
  - CA-5 certificate contains CA-5’s public key

    So what? What can she verify? Is Alice any better off?

    What if she does not know RA-2?

    What about root’s public key?

- The hierarchy forms a chain of trust or certification path
PKI Directories

• Another issue is where the certificates are stored

• Have each user store his or her certificate?
  What are the advantages and disadvantages?

• Have DNS store certificates?
  What are the advantages and disadvantages?

PKI Revocation

• There may be a reason to revoke a certificate
  Give an example?

• CA can periodically issue a Certificate Revocation List (CRL)
  – Certificates have expiration, so the list may not be too long
    What are some other issues with using a CRL?

    How are CRL managed?

    What is a reasonable certificate lifetime?
PKI Risks

- PKI seems like a reasonable solution for security
  - Appears to be a scalable approach
- Carl Ellison and Bruce Schneier have described risks of PKI
  - Can you really trust the CA, and for what?
  - How to protect the private key?
  - Can you verify the root keys?
  - How do you know you are requesting the right certificate?

*What is the common theme to these (and other) concerns?*


---

Diffie-Hellman Key Exchange

- Let’s go back to secret-key cipher
  - Assume that Alice and Bob need to transmit messages
  - Furthermore, assume a secret key has not been established
  - A key exchange technique is needed...
- Diffie-Hellman exchange allows strangers to establish a key
  - Uses similar mathematical principles as RSA
- The exchange algorithm is
  - Alice and Bob agree on two large prime numbers $n$ and $g$
  - Where $\frac{n-1}{2}$ is also prime (other conditions apply to $g$)
  - These numbers ($n$ and $g$) can be public
  - Alice now picks a random number $x$ and keeps it secret
  - Bob does the same, pick a secret random $y$
Alice starts the exchange by sending Bob \([n, g, g^x \pmod{n}]\)

Bob responds to Alice by sending \([g^y \pmod{n}]\)

Alice then calculates \((g^y \pmod{n})^x \pmod{n}\)

Similarly Bob calculates \((g^x \pmod{n})^y \pmod{n}\)

Both operations yield \(g^{xy} \pmod{n}\), which is the secret key

- Of course Trudy has seen all the messages
  - She knows \(g\) and \(n\)
  - But she cannot determine \(x\) and \(y\), no practical algorithm exists for computing discrete logarithms modulo on large primes...

- Example key exchange
  - Assume \(n = 47\) and \(g = 3\)
  - Alice picks \(x = 8\) and Bob picks \(y = 10\), both are secret
  - Alice computes \(3^8 \pmod{47} = 28\) and sends \([47, 3, 28]\)
  - Bob sends Alice \([17]\)
  - Alice computes \(17^8 \pmod{47} = 4\)
  - Bob computes \(28^{10} \pmod{47} = 4\)
  - Therefore the secret key is 4 (OK, it’s a bit small...)

- Trudy has seen all the messages
  - Must solve the equation \(3^x \pmod{47} = 28\) (from Alice’s first message)
  - If she determines \(x = 8\), she can then compute \(17^8 \pmod{47} = 4\)
  - Values are so small, just use exhaustive search for \(x\), therefore you should use large numbers...
Trudy’s Revenge

- Despite the elegance of Diffie-Hellman, there is a problem
  - Bob receives the $[47, 3, 28]$, how does he know it’s from Alice?

- Assume Trudy can intercept the messages from Bob and Alice

  - While Alice and Bob are selecting their $x$ and $y$, Trudy selects $z$
  - Alice sends first message 1 to Bob
  - Trudy intercepts, sends Bob message 2 with her $z$
  - Trudy then sends Alice message 3 with her $z$
  - Trudy intercepts message 4

  - Now everyone does the modular arithmetic
  - Alice computes $g^{xz} \pmod n$, so does Trudy for messages to Alice
  - Bob computes $g^{yz} \pmod n$, so does Trudy for messages to Bob

  How does Trudy use these keys?

- This is called the bucket brigade attack or the (wo)man-in-the-middle attack
  - Similar to a hijack attack
Steganography

- There are two general methods for transmitting secrets
  - Cryptography - Info scrambled and reconstituted using a key
  - Stenography - Secret encoded in another message such that it is unseen by the casual user

- Steganography has been in use for many centuries
  - Greeks in 4th century BC
  - Francis Bacon and Shakespeare
  - German microdot during WWII

- Steganography and the authorship of Shakespearean plays
  - Hidden texts have been found in Shakespearean plays
  - They point to Francis Bacon as the true author of the plays

Steganography and Digital Media

- Recently steganography has seen a revival
  - Digital media provides many opportunities to hide info
  - Digital media typically has an accuracy greater than necessary
  - Can downgrade and make room for secrete information
  - Examples include images, music, and video

What about executables?

- Consider an image that consists of a matrix of pixels
  - Assume each pixel is 24 bits (8 for red, green, and blue)
  - Use two Least Significant Bits (LSB) of each color
  - This would provide 9 bits of information per pixel
- The resulting image is downgraded (distorted)

- Other steganography methods exist
  - Frequency encoding
  - Adding information in media header, PS for example

```
%!PS
% Creator: Adobe Illustrator(TM) 5.0
%For: (Al Pierno) (Hadel Studio)
```

- TCP/IP headers, not all the bits are used

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Version</td>
<td>Type of service</td>
<td>Total length</td>
</tr>
<tr>
<td></td>
<td>Identification</td>
<td>[0][1] Fragment offset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time to live</td>
<td>Protocol</td>
<td>Header checksum</td>
</tr>
<tr>
<td></td>
<td>Source address</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Destination address</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Options (0 or more words)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- Disk fragmentation

- Discovering hidden data
  - Very difficult without the original media

*Steganography can be used legitimately, give an example.*