Public key encryption algorithms

Two inter-related requirements:

1. need \( d_B(\cdot) \) and \( e_B(\cdot) \) such that \( d_B(e_B(m)) = m \)

2. need public and private keys for \( d_B(\cdot) \) and \( e_B(\cdot) \)

RSA: Rivest, Shamir, Adelson algorithm

RSA: Choosing keys

1. Choose two large prime numbers \( p, q \) (e.g., 1024 bits each)
2. Compute \( n = pq, \ z = (p-1)(q-1) \)
3. Choose \( e \) (with \( e < n \)) that has no common factors with \( z \). (\( e, z \) are “relatively prime”).
4. Choose \( d \) such that \( ed - 1 \) is exactly divisible by \( z \). (in other words: \( ed \mod z = 1 \)).
5. Public key is \((n, e)\). Private key is \((n, d)\).

RSA: Encryption, decryption

0. Given \((n, e)\) and \((n, d)\) as computed above
1. To encrypt bit pattern, \( m \), compute \( c = m^e \mod n \) (i.e., remainder when \( m^e \) is divided by \( n \))
2. To decrypt received bit pattern, \( c \), compute \( m = c^d \mod n \) (i.e., remainder when \( c^d \) is divided by \( n \))

m = (m^e \mod n)^d \mod n

RSA example:

Bob chooses \( p=5, q=7 \). Then \( n=35, z=24 \).
\( e=5 \) (so \( e, z \) relatively prime).
\( d=29 \) (so \( ed - 1 \) exactly divisible by \( z \)).

<table>
<thead>
<tr>
<th>encrypt:</th>
<th>letter</th>
<th>m</th>
<th>m^e</th>
<th>c = m^e \mod n</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>l</td>
<td>12</td>
<td>1524832</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>decrypt:</th>
<th>c</th>
<th>c^d</th>
<th>m = c^d \mod n</th>
<th>letter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>17</td>
<td>48196857210675091509141825223072000</td>
<td>12</td>
<td>l</td>
</tr>
</tbody>
</table>
RSA: Why — part 1

\[ m = (m^e \mod n)^d \mod n \]

Number theory result: If \( p, q \) prime, \( n = pq \), then

\[ x^y \mod n = x^{(p-1)(q-1)} \mod n \]

\[ (m^e \mod n)^d \mod n = m^{ed} \mod n \]

\[ = m^{ed} \mod (p-1)(q-1) \mod n \]

(using number theory result above)

\[ = m \]

RSA: Why — Part 2

- Why is the message secure?
- Recovering the message requires \( d \) and \( n \)
- \( n \) is public (part of public key)
- Therefore, how can \( d \) be obtained?
- Nobody knows, in spite of centuries of searching, how to factor faster than exhaustive search
- Obtaining \( d \) is equivalent to factoring \( n \)
- With large \( n \), factoring is infeasible

Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol ap1.0: Alice says “I am Alice”

Protocol ap2.0: Alice says “I am Alice” and sends her IP address along to “prove” it.

Failure scenario??

Failure scenario??
**Authentication: another try**

**Protocol ap3.0:** Alice says “I am Alice” and sends her secret password to “prove” it.

**I am Alice, password**

Failure scenario?

**Authentication: yet another try**

**Protocol ap3.1:** Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.

**I am Alice, encrypt(password)**

Failure scenario?

**Authentication: yet another try**

**Goal:** avoid playback attack

**Nonce:** number (R) used only once in a lifetime

**ap4.0:** to prove Alice “live”, Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key

**Nonce:** number (R) used only once in a lifetime

**ap4.0:** to prove Alice “live”, Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key

**Failures, drawbacks?**

**Authentication: ap5.0**

**ap4.0 requires shared symmetric key**

- problem: how do Bob, Alice agree on key
- can we authenticate using public key techniques?

**ap5.0:** use nonce, public key cryptography
**ap5.0: security hole**

*Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)*

Need "certified" public keys (more later ...)

Bob sends doc, \( X \)
\[\text{encrypted using } e_y\]

Alice decrypts \( e_y(X) \)
\[\text{recovered } X\]

Trudy decrypts \( e_y(X) \)
\[\text{recovered } X\]
\[\text{encrypted using } e_x\]
\[\text{towards } e_y(X) \text{ to Alice}\]