Design Goals

- **Simplicity**: easy to implement
- **Wide Applicability**: secure (dictionary attacks), works with multiple password databases
- **Efficiency**: minimize PK ops, round trips
- **Flexibility**: multiple ciphersuites
- **Extensibility**: secure data exchange with many applications
Approaches

- Client sends plaintext password to Server
- Wide compatibility (i.e. use PAM)

- Client and Server confirm knowledge of same password without exchanging it
- More secure, but fewer applications
Security Concern: Dictionary Attacks

• Online dictionary attacks
  – Adversary repeatedly authenticates to the server with guessed passwords
  – Prevent by locking accounts after too many invalid attempts

• Offline dictionary attacks
  – Using passively captured packets, repeatedly guess passwords
  – Full prevention requires use of public-key crypto
Offline Dictionary Attack Prevention

- EKE/SPEKE/SRP approaches
  - Basically perform Diffie-Hellman using password hash as the base
  - Exchange \((pw)^x\) and \((pw)^y\), compute \((pw)^{(xy)}\)
  - All suffer from IPR issues
  - Limited compatibility with backend databases
Offline Dictionary Attack Prevention

• PKI-based approach:
  – Pre-authenticate the server (i.e. certificate)
  – encrypt exchange using server’s public key
  – TLS?
  – Could also support username confidentiality
  – Works for both authentication scenarios

• Dictionary Attack *Mitigation*
  – Salt password and hash it *a lot*
  – Increases time and complexity of dictionary attack
  – What 11i’s PSK authentication does
# Feature Matrix

<table>
<thead>
<tr>
<th></th>
<th>Plaintext</th>
<th>Hashed</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKE</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>PKI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Salt</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
PKI-based Approach

• Custom PKI-based protocol
  – Light weight, could be 2 RT (+ fragmentation)
  – Similar to what EAP-PAX does for provisioning

• Use TLS-based approach
  – Much of the hard work is already done
  – Minimum of 3 RT required, assuming 2 RT TLS handshake
  – Could use a tunneled method
    • PEAP, TTLS
    • Inner method could be PAP, CHAP, etc
    • Protected data exchange already built-in
Choices

- Write our own PKI-based protocol
- Write our own TLS-based protocol
- Use tunneled method and write our own inner method
- Use tunneled method and use existing inner method

- PEAP & MSCHAPv2?
- EAP-FAST?