Modified Fiat-Shamir  
(another zero knowledge)

- Alice establishes public key \(<n,v>\)
  - \(n = p \times q\)
  - \(v\) is a number for which only Alice knows sq root mod \(n\)
  - \(v\) generated by picking random \(s\), and squaring mod \(n\)
  - \(s\) is her private key
- For Bob to authenticate Alice:
  - Alice chooses \(k\) random numbers \(r_1...r_k\). For each, she sends \(r_i^2 \mod n\) to Bob
  - Bob chooses random subset and tells Alice. This is set\(_1\). The rest is set\(_2\).
  - Alice sends \(sr_i \mod n\) for each # of set\(_1\), and sends \(r_i \mod n\) for set\(_2\).
  - Bob squares Alice's replies mod \(n\).
    - For those in set\(_1\), the square should be \(vr_i^2 \mod n\).
    - For set\(_2\), the square should be \(r_i^2 \mod n\)
- Security comes from difficulty of finding root \(mod n\) of a number.
F-S cont.

Why does this work?
- \( s^2 = v \)
- If Trudy impersonates Bob
  - she can create her own \( r_i \)'s, and therefore \( r_i^2 \)'s
  - can't generate \( sr_i \), because she doesn't know the root of \( v \)

Why do we need \( R \)?
- If Trudy overhears Alice being authenticated by Bob, she gets some pairs \( <r_i^2, sr_i> \)
- Trudy can then try to impersonate Alice to Fred. However:
  - she needs to generate \( r_i^2 \)'s.
  - if Fred picks one for set 1, Trudy better have used an overheard pair
  - if Fred picks one for set 2, Trudy better have made up her own
    - given \( sr_1 \), should could not generate \( r_1 \)
- Same probabilities as with graph isomorphism

Why Is This Good?
- RSA is zero knowledge, or almost.
  - expensive!
- However, Fiat-Shamir requires only:
  - Alice has 45 multiplies (assuming 30 \( r_i \)'s)
  - work for Bob the same
Hybrid encryption

- Public-key encryption is “slow”
- Encrypting “block-by-block” would be inefficient for long messages

- Hybrid encryption gives the functionality of public-key encryption at the (asymptotic) efficiency of private-key encryption!

Enc = public-key encryption scheme
Enc’ = private-key encryption scheme
Security

- If public-key component and private-key component are secure against chosen-plaintext attacks, then hybrid encryption is secure against chosen-plaintext attacks.

Extension

- How should hybrid encryption be done when sending the same message to multiple recipients (e.g., email encryption)?
Signature schemes

Basic idea

• A signer publishes a public key \( pk \)
  • As usual, we assume everyone has a correct copy of \( pk \)
• To sign a message \( m \), the signer uses its private key to generate a signature \( \sigma \)
• Anyone can verify that \( \sigma \) is a valid signature on \( m \) with respect to the signer’s public key \( pk \)
  • Since only the signer knows the corresponding private key, we take this to mean the signer has “certified” \( m \)
• Security: no one should be able to generate a valid signature other than the legitimate signer
Typical application

- Software company wants to periodically release patches of its software
  - Doesn’t want a malicious adversary to be able to change even a single bit of the legitimate patch

Solution:
- Bundle a copy of the company’s public key with initial copy of the software
- Software patches signed (with a version number)
- Do not accept patch unless it comes with a valid signature (and increasing version number)

Signatures vs. MACs

- Could MACs work in the previous example?
  - Computing one signature vs. multiple MACs
  - Managing one key vs. multiple keys
  - Public verifiability
  - Transferability
  - Non-repudiation

  Not obtained by MACs!