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 $r_i^{2s}$
    • can't generate $sr_i$, because she doesn't know the root of $v$

• Why do we need set$_2$?
  • 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. W/ a set$_2$:
    • she needs to generate $r_i^{2s}$
    • 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 $sr_i$, should could not generate $r_i$
  • Same probabilities as with graph isomorphism

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
  \[\text{Not obtained by MACs!}\]
Functional definition

- Key-generation algorithm: randomized algorithm that outputs \((pk, sk)\)
- Signing algorithm:
  - Takes a private key and a message, and outputs a signature: \(\sigma \leftarrow \text{Sign}_{sk}(m)\)
- Verification algorithm:
  - Takes a public key, a message, and a signature and outputs a decision bit: \(b = \text{Vrfy}_{pk}(m, \sigma)\)
- Correctness: for all \((pk, sk)\), \(\text{Vrfy}_{pk}(m, \text{Sign}_{sk}(m)) = 1\)

Security?

- Analogous to MACs
  - Except that adversary is given the signer’s public key
- \((pk, sk)\) generated at random; adversary given \(pk\)
- Adversary given \(\sigma_1 = \text{Sign}_{sk}(m_1), \ldots, \sigma_n = \text{Sign}_{sk}(m_n)\) for \(m_1, \ldots, m_n\) of its choice
- Attacker “breaks” the scheme if it outputs a forgery; i.e., \((m, \sigma)\) with:
  - \(m \neq m_i\) for all \(i\)
  - \(\text{Vrfy}_{pk}(m, \sigma) = 1\)
“Textbook RSA” signatures

- Public key \((N, e)\); private key \((N, d)\)
- To sign message \(m \in \mathbb{Z}_N^*\), compute \(\sigma = m^d \mod N\)
- To verify signature \(\sigma\) on message \(m\), check whether \(\sigma^e = m \mod N\)
- Correctness holds…

- …what about security?

Security of textbook RSA sigs?

- Textbook RSA signatures are \textit{not} secure
  - Easy to forge a signature on a random message
    - Choose random number \(\sigma\)
    - Define \(m = \sigma^e \mod N\)
    - It verifies!
  - Easy to forge a signature on a chosen message, given one signatures on messages of adversary’s choice
    - Assume wants to sign \(m\)
      - Finds \(m_1, m_2\) such that \(m_1 \cdot m_2 = m\)
      - Gets \(\sigma_1, \sigma_2\) \(\mod N\) \(\text{(see a couple slides back)}\)
      - \(\sigma = \sigma_1 \cdot \sigma_2 \mod N\) also is the same as \(m^e \mod N\)
        - \(m_1^e \cdot m_2^e = (m_1 \cdot m_2)^e = m^e\)
  - How to fix?
    - hashes
IEEE Keys visible on website for month

Authentication
Requirements for Authentication System

• Secure
  • Eavesdropper out of luck
• Reliable
  • Especially important for centralized schemes
• Transparent
  • User should not be aware of authentication (besides password)
• Scalable

Authentication Overview

■ Scenario:
  ■ Large set of principals attached to an open channel (e.g. internet)
  ■ Each principal repeatedly
    ■ attempts to start connection (session) w/ a specified principal
    ■ upon establishment, exchanges msgs
    ■ closes
■ Authentication is about:
  ■ When principal A assumes it has connected to B,
    ■ A is indeed talking to B
  ■ When principal A assumes confidentiality/integrity of msg exchange,
    ■ msgs are private and intact
■ Principal can be human or executing program
  ■ Programs can use high-quality secrets (eg, $2^{64}$)
  ■ Humans use low-quality secrets (eg, maybe $2^{32}$), cannot do crypto
Attacks

- Authentication cannot handle all attacks, eg
  - Overrun (take over) of human principal
  - Overrun memory while program doing login authentication

- Network-based attacks (in order of increasing difficulty)
  - Spoofing messages (sources)
    - C at n_c sends msgs w/ sender network address n_A
  - Eavesdropping: observing msgs in the channel
    - Very easy in WLANS and LANS because of broadcast
    - Not easy in wired point-to-point links
      - Tap router ports
      - Compromise route computation
  - Intercepting msgs, changing them, resending
    - Very easy in WLANS and LANS because of broadcast
    - Not easy in wired point-to-point links

End-Host Based Attacks

- Principal C pretends to be A on computer n_A
  - online password guessing
- Read data on hard disk of n_A or A
  - Obtain old keys, password files
  - Obtain current keys, password files
  - Offline password guessing on encrypted passwords
- Overrun computer n_A
  - while A is not at n_A
  - while A is at n_A
- Read data in memory of n_A while A executing
- Overrun a program
  - mail client, web browser
  - network servers (FS, time-of-day, auth servers..)
Three Types of Network Authentication

• Password-based authentication
  • Authenticating by showing a secret password to remote peer
  • Always vulnerable to eavesdropping and/or online password guessing

• Address-based authentication
  • by using a physically-secured terminal/computer

• Crypto-based authentication
  • show evidence of a secret key to remote peer
  • does not expose the secret to the peer (or network)

Password Authentication

I'm Alice, password is “fiddlesticks”

• Problem
  • Eavesdropping

• Issues:
  • Off- vs On-line password guessing

• Storage
  • plaintext
  • encrypted
  • hashes
Address-Based Authentication

• “Not who you are, but where”
  • /etc/hosts.equiv (check it out on a mac)
  • .rhosts
    • per-user list of <account, computer> tuples to allow
• Attacks
  • safe from eavesdropping
  • bad when foo, bar list each other, foo broken into
  • network address spoofing
    • broadcast networks!

Network Address Spoofing

• Basic
  • everyone can read on broadcast networks (LAN, wireless)
  • can set your “source” IP address to anything you want
• On network path between Alice and dst
  • re-direct traffic by sending routing msgs to routers
    • these might be authenticated
  • source routing
    • Trudy marks her packets
      • <Alice, Trudy, dst>
    • dst will send back w/ the same path
      • Trudy doesn't forward
Crypto Authentication

• Everything we’ve talked about before, but how w/ passwords?
• Passwords → keys
  • hashing: fast but small keys, structure problems
  • decryption: use password to decrypt a better key

Eavesdropping and Database Reading

I’m Alice

R

R signed w/ Alice’s private key

If I read Bob’s database, can I impersonate Alice to Bob?
Eavesdropping and Database Reading

Alice, “fiddle”

knows hash of password
computes hash of “fiddle” and compares

- Eavesdropping?
- Database reading?

Eavesdropping and Database Reading

I’m Alice

R

K_{alice}\{R\}

- Eavesdropping?
- Database reading?
- Protecting against both is hard w/o public key.
Lamport’s Hash

Alice knows: 
<n, hash^n(“fiddle”)> 

Workstation knows: 
1) compares H(x) to hash^n(“fiddle”) 
2) stores <n-1, x>

Eavesdropping? 
Database reading?