6.033 Computer System Engineering
Spring 2009

Nickolai Zeldovich

high-level ideas for today:
    public key authentication
    certificate authorities
    session keys
    network protocols and attacks: be explicit

Authentication, Authorization
==========

--- board 1 ---

setting: client and server sending messages over the internet
    want to achieve: authentication, authorization, confidentiality

last time: how to authenticate requests given a shared secret
    stock trading example: "buy 100 shares of google"
    password, hash the request and password together

```
 [pw]           [pw]
    C --- I --- S

     <----------
     m, H(m + pw)   m: "GOOG shares cost $100"

     ---------->
     m, H(m + pw)   m: "buy 100 shares of GOOG"
```

--- board 2 ---

now: what to do if we don't have a shared secret ahead of time?
    e.g. stock quotes coming from web server

    want a pair of functions SIGN and VERIFY

```
                       Shared           Public
                        key           key

      SIGN(m, K1) -> sig      H(m+K1)           m^e mod N
      VERIFY(m, sig, K2)      sig==H(m+K2)      sig^d mod N == m

                       MAC         Signature
```

    SLIDE: RSA reminder

      TELL STUDENTS that both of these examples are crippled
          want to use HMAC and better public-key signature schemes
          these are just for your intuition

    formalize previous password-based authentication: MAC
    new: different sign, verify keys
    key property of public-key signatures: others can verify but not generate!

attacks on SIGN, VERIFY for authentication [right side of board?]
    crypto attacks

```
      figure out the key or find ways to violate crypto's guarantees
    authentication
      modification
      reordering
      extending
      splicing

    for example RSA as-described is vulnerable to multiplying messages
      also need to take care to condense string into an integer

--- board 3 ---

what does it even mean to authenticate someone in this case?
    how do you know what public key to use for someone?
    idea: split up into authenticating a name, and then trusting a name
    someone is going to manage names for us

    symmetric
      S <-> PW <-> trust

    asymmetric
      S <-> K1 <-> name <-> trust
            \----V----/
            trust someone to tell us these bindings

--- board 4 ---

talk to a server, get a response and signature with server's key
    now talk to a key-naming server to ask it, what's this guy's name?
    key-naming server responds, along with its own signature
    must trust key-naming server ahead of time
    otherwise might need to talk to another key-naming server...

      SIGN("GOOG is $100", K1), K2

    [C knows K2_ca]

      C <--------- S
      ^
      |
      \----> CA
      "Who is K2?"
      SIGN("K2 is quotes.nasdaq.com", K1_ca)

--- board 5 ---

idea: don't actually need to query the key-naming server all the time
    give key-naming server's signed response to original server
    typically called a "certificate"

 [K2_ca]
    C <--------------- S

      m: "GOOG is $100"
      SIGN(m, K1_q), K2_q
      SIGN("K2_q is quotes.nasdaq.com", K1_ca)
```

delegation: certificates might be chained
      otherwise everyone has to talk to the root certificate authority
    this is roughly how DNSSEC works
    user trusts the top-level certificate

      SIGN("K2_q is quotes.nasdaq.com", K1_nasdaq), K2_nasdaq
      SIGN("K2_nasdaq is *.nasdaq.com", K1_ca)

THE ENTIRE CHAIN HAS TO BE TRUSTED!
    and infact the weakest possible way to construct the chain is the problem

how would one of these CAs decide who the next guy is?
    used to be that you have to go to a town office and get a business license
      can specify any name you want, can probably find some town that's lax
    now it's just verified by email on your domain registration
      one popular CA will send email to webmaster@domain.com to ask if it's OK

SLIDE: easy to spoof this process, even for well-known companies!
    reiterate: weakest link in the chain is what matters

SLIDE: rarely-used mechanisms are a disaster for security
    Verisign designed a certificate-revocation list for this purpose
    not part of the usual workflow, so never tested!
    indeed, when time came to use it, noone knew how to find this CRL!

in practice users would think that a secure website was secure, end of story
    didn't bother thinking about what server they were actually talking to
    new kind of certificates now that involve "extended validation"
    pops up a green bar showing the name of the company that was validated

alternative: blindly trust & remember keys
    vulnerable first time around
    SSH
    add a certificate exception in a browser

--- board 6 ---

ok, now have public keys for both parties, what next?
    asymmetric encryption/signatures are computationally-expensive
    most protocols use public keys to establish symmetric session key
      then use symmetric encryption or MACs for subsequent messages

session key protocol: denning-sacco
    A wants to talk with B, need to establish symmetric session key
    1. A -> S: A, B
    2. S -> A: SIGN({A, Ka}, Ks), SIGN({B, Kb}, Ks)
    3. A picks a session key Kab
      A -> B: SIGN({A, Ka}, Ks), ENCRYPT(SIGN(Kab, Ka), Kb)
    4. A and B communicate using Kab to do symmetric encryption/MACs
      (usually not a good idea to use same key for two purposes, so
       maybe Kab is actually two keys, one for encrypt, one for MAC)

--- board 7 --- [to the right?]

desired goals

1. secrecy: only A, B know Kab
2. authentication: A, B know other's identity

is this true?
  1. A knows Kab, and the only person that can decrypt msg 3 is B
  2. A knows the only person that can decrypt Kab must be B
     B knows the only person that could have sent Kab was A (signed!)

turns out the protocol is broken!
recipient can't really infer the party on the other end is A!
suppose B was malicious (e.g. A visited a malicious web site)
can impersonate A to some other party (e.g. another web site)
  .. get {C:Kc}_Ks
  3. B -> C:
      SIGN({A, Ka}, Ks), ENCRYPT(SIGN(Kab, Ka), Kc)
what recipient can infer is that A signed this message, but so what?
  it wanted to use Kab to talk to someone, but not necessarily us (rcpt)!