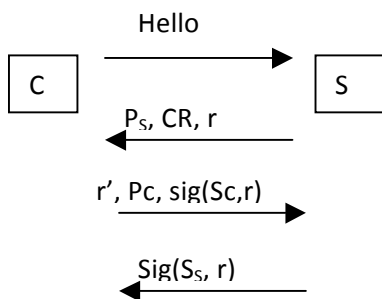
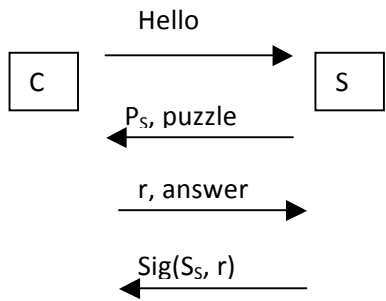


Client Puzzles for DoS

SSL



$P=e,N$

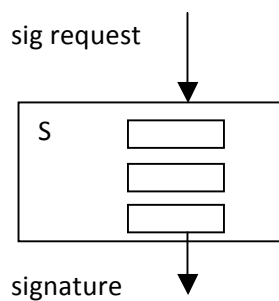
$S=d,N$, where d is large

$y = \text{sig}((d,N),x) = x^d \text{ mod } N$

$\text{ver}((e,N), x, y) = y^e = x \text{ mod } N$

- Assume server can do about 100 sig/sec
- Client work
 - Almost nothing
- Server work
 - 1 RSA sig

Number of RSA request



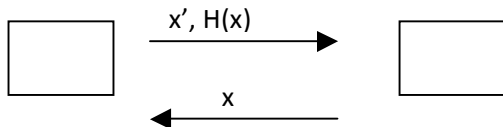
50 (request/sec) → 100 queue size = 0

100 → 50 queue fill, slower

- Computing is more expensive than verifying
- Cannot ask server to pick small exponent because it is vulnerable to other attack
- Attack: if you allow to choose your public private key pair, you can swap the exponent and allow you to generate signature fast and server verification slow

Puzzle also used in SPAM

- Need to upgrade SMTP server, can be deploy one at a time
- Attacker user botnet, does not care about additional computation

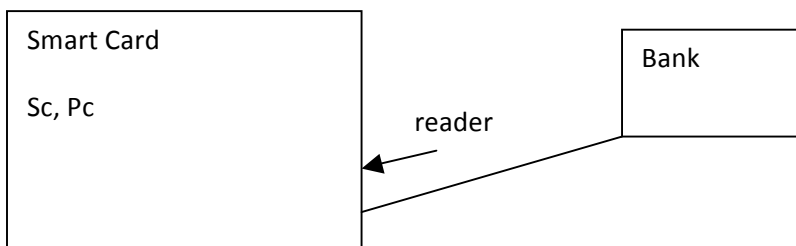


- Where x' is x with l lsb set to 0
- e.g. $l = 16$ bits
- suppose botnet size about 2^{17}
- server can perform about 2^{10} signatures per second
- require puzzle difficulty: 2^7 seconds

Side Channel Attacks

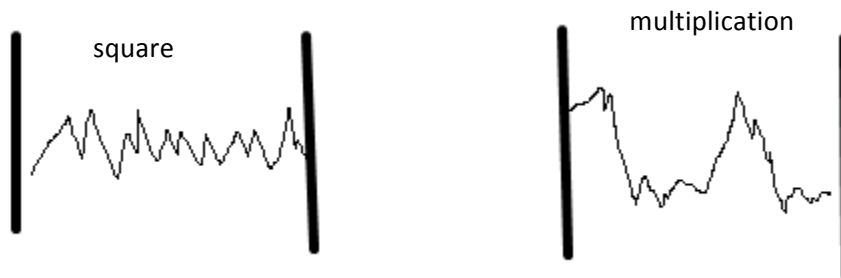
- any information channel other than the explicit channel
 - time
 - power
 - temperature
 - disk usage
 - sound
 - light/EMR
 - cache miss
 - CPU load
- Covert channel deliberately leak information to communication
- Side channel leak information by accident

Card perform signature = $\text{sig}(Ss,m)$



```

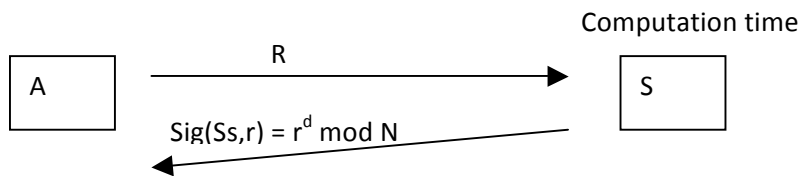
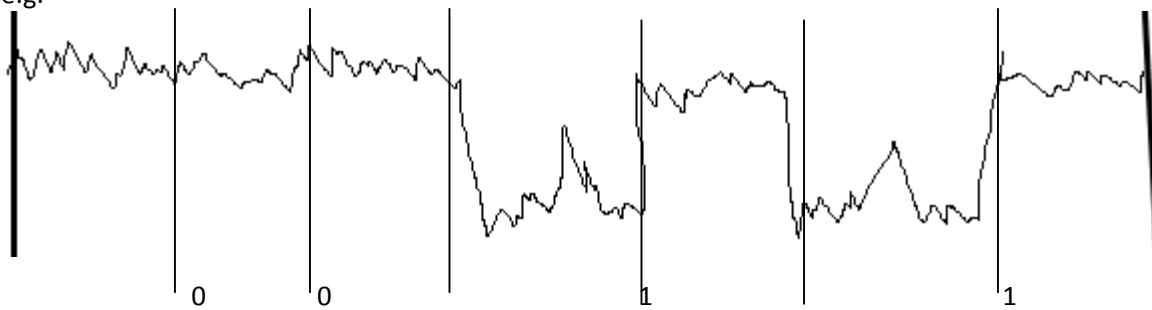
sig(Sc, m) = md mod N
mod (m, d, N) {
    acc=1
    for l = |d|-1 to 0
        acc = acc2 mod N
        if di = 1
            acc * m mod N
        else
            dummy = acc * m mod N // fix with 33 to 50% slower
    return acc
}
    
```



Power consumption of computing RSA

Power Analysis Attack

e.g.



Measured time = computation + rtt
 = computation + noise

Simple Executing Trace of mod N

Several
hundreds
more
operations

{
 acc = 1
 acc = acc² //ignore
 acc = acc * m
 acc = acc²
 acc = acc²
 acc = acc * m
 ...
 ...

equals to about average operation time
x # of remaining operation = test

If the bit is 1 “acc = acc * m” executed

take mfast]_{i=1}^s and mslow]_{i=1}^s

get tfast avg and tslow = avg

if tfast = tslow

 then d_{|d|-1} = 0

 else d_{|d|-1} = 1