CSCI262 : System Security

Denial of Service Attacks (DoS)

Schedule

- What is Denial of Service attacks (DoS)?
- Flooding Attacks
- Distributed Denial of Service Attacks (DDoS)
- Reflector and Amplifier Attacks
- Defences

Denial of Service Attacks (DoS)

- Denial of service is a form of attack on the availability of some service.
  - i.e., network services (in computer security)
- NIST SP 800-61 (Computer Security Incident Handling Guide 2012)

  A denial of service (DoS) is an action that prevents or impairs the authorised use of networks, systems, or applications by exhausting resources such as central processing units (CPU), memory, bandwidth, and disk space.
- Three categories of resources to be attacked
  - Network bandwidth
  - System resources
  - Application resources

Bandwidth consumption

- Network bandwidth relates to the capacity of the network links connecting a server to the wider Internet.
- Any communication network has an upper bound on the volume of traffic at one time.
  - Denial of service by bandwidth consumption occurs when this volume of traffic is reached, and further traffic cannot be transmitted.
  - This could happen through legitimate interactions, perhaps simply meaning the link needs to be upgraded.
  - When the limit is reached existing traffic will slow, freeze, or be disconnected.
  - Such inactivity may be evidence of a DoS attack.

- Network links of organisers is connected to ISP
- Have lower capacity compared to connections within and between ISP routers
- Attackers can send a lot of ‘malicious’ traffic to the target server, overwhelming legitimate traffic, causing DoS attacks.

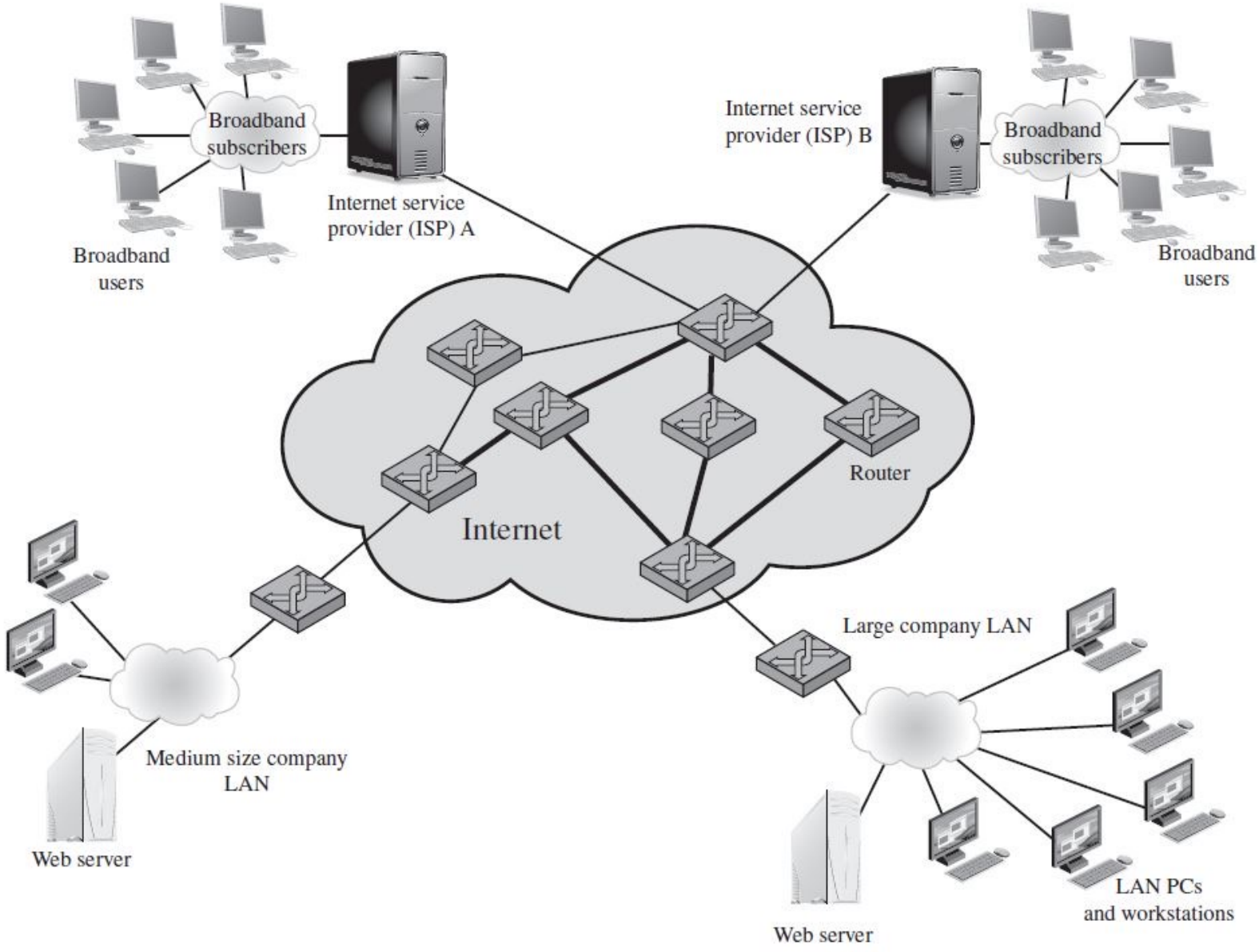


Figure 7.1 in [SB18]

Resource saturation

- Just as communication bandwidth is a DoS vulnerable resource for networks, computers also have DoS vulnerable resources; including memory, storage, and processor capacities.
- Rather than consuming bandwidth with large volumes of traffic, specific types of packets are sent that consume the limited resources available on the system.
  - These include temporary buffers used to hold arriving packets, tables of open connections, and similar memory data structures.
  - The SYN flood is a popular example of this type that uses all the available networking resources on a system.
  - Web servers are common targets for a denial-of-service attack.

System and application crash

- System or application crashes are generally easy to launch and difficult to protect against.
- Attackers can use packets whose structure triggers a bug in the system's network handling software, causing it to crash
  - The system can no longer communicate over the network until this software is reloaded
    - generally by rebooting the target system
- This is known as a **poison attack**
  - *Ping of death* attack is a typical example
  - This targeted bugs in the Windows network code that handled ICMP (Internet Control Message Protocol) echo request packets

- While small packets of data can be used to exploit vulnerabilities, if no such vulnerabilities are available we can still launch a Denial of Service attack, using “brute force”.
  - This is a difference between a quality and a quantity attack.
- We could flood a system or network with so much information that it cannot respond.
  - If a system can only handle 10 packets a second, and an attacker sends it 20 packets a second…
  - … the system may well fall over.
  - Even if it doesn’t fall over, the processing of those illegitimate packets stops or slows down the processing of legitimate packets.
  - This is also a denial of service.

Ping of Death

- Ping packets are part of the Internet Control Message Protocol (ICMP) which are part of IP (Internet Protocol).
  - ICMP is for testing connectivity to various machines on the Internet.
  - ICMP can convey status and error information.
- Ping utilises ICMP sends an ICMP echo reply.
  - The response to a ping is referred to as a pong.

Launching a “Ping of Death”

- Typically a command like the following, using a DOS window on Windows, would be used:

ping -l 65527 IP-address

where the 65527 corresponds to the amount of data to send and the IP-address specifying the target.

This shouldn’t work on your computer!

- When this was discovered (November 1996) most operating systems were vulnerable.
  - Networked hosts shouldn't be vulnerable now! ☺
- So, what happens?
- The ping of death isn't a problem with ping, or with ICMP.
  - It is a problem to do with fragmentation, reconstruction and IP handling.

- When you communicate over the internet there are several layers or protocols that your traffic goes through (e.g., Physical, Data Link, IP, …).
- The size of the IP packet is limited to at most 65,535 bytes, and some of this is header information.
- There are two factors that the ping of death exploits:
  - Fragmentation and reconstruction.
    - At the “lower level” of communication (Data link layer) we don’t send IP packets.
    - The data link layer can only handle smaller lumps of data, so we fragment.
    - With the ping of death the size of reconstructed packet is greater than the IP limit.
  - IP mishandling.
    - The second problem is that the too large IP isn’t handled well.
    - Basically we get a buffer overflow, which we will look at soon.

- When computers are pinged with a packet larger than 1480 bytes the packet must be fragmented before it is sent.
- IP fragments are reassembled at destination before processing.
  - Need an offset for each fragment.
- Offset in IP header has 13 bits with 8 bytes as 1 unit, so maximum offset is (2^13 -1)*8 = 65528.
- If the last packet has a large size…
  - … a buffer overflow occurs when the target machine reassembles the packet.

Have a look at the “Invite of Death” too … (Feb 2009)
It is a quality DoS attack but against VoIP.

Classical DoS

- The simplest attacks are flooding attacks.
  - Send a whole lot of messages to overwhelm the capacity of the network connection to the target organization.
  - Other valid traffic cannot get through
  - E.g., ping flood attack:
  - Disadvantages: source address in the ICMP echo request is identified
    - Legal action can be taken in response
    - the targeted system will attempt to respond to the packets being sent, hence the attack maybe detected

Source Address Spoofing

- Use forged source address (**source address spoofing**)
- Attackers generate large volume of packets with target as the destination address but with randomly selected different source address for each packet
- The attack is now hard to be identified

SYN Spoofing

- Exploit the handshake process of a TCP connection to overflow the tables used to manage such connections
- Hence future connection requests from legitimate users fail, denying them access to the server
  - It is an attack on system resources, specifically the network handling code in the operating system.
- Let’s look at first how the handshake that TCP uses to establish a connection works in the next slide.

TCP three-way handshake

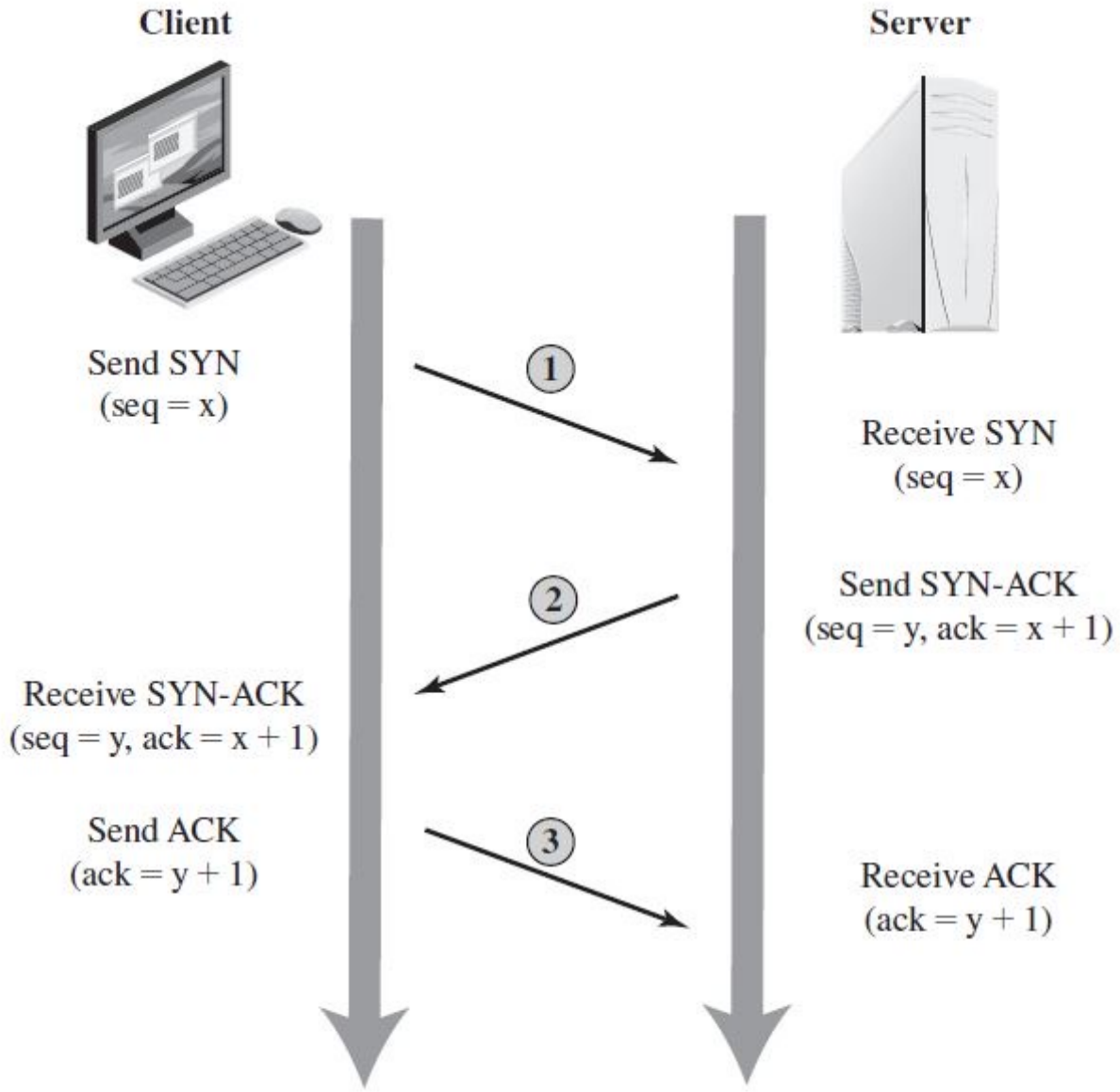


Figure 7.2 **TCP Three-Way Connection Handshake**

TCP SYN Spoofing Attack

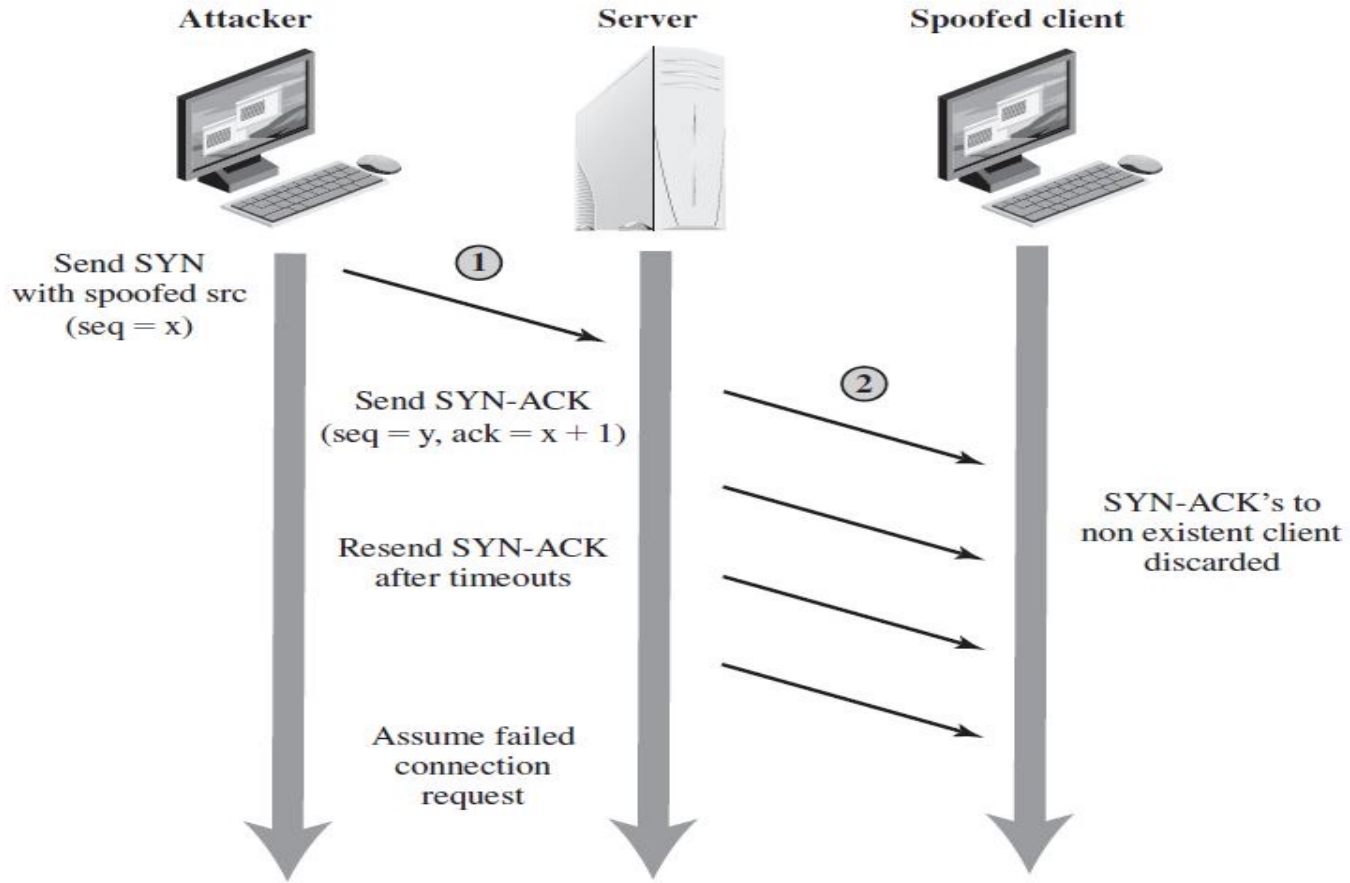


Figure 7.3 TCP SYN SpoofingAttack

A valid system at the spoofed client will respond with a reset (RST) packet to cancel this unknown connection request. The server will cancel the connection request and remove the saved information.

But if the source system is too busy or doesn't exist, there will be no reply. The server will continue to resend the SYN-ACK before finally assuming the connection request has failed and subsequently deleting the information saved.But there is a limit on how many requests can be stored and this table may be filled.

Flooding attacks

- **Aim**: overload the network capacity on some link to a server, or to overload the server's ability to handle and respond to this traffic
- These attack flood the network link the server with a torrent of malicious packets competing with
  - Hence overwhelming valid traffic flowing to the server
- Any type of network package can be used in a flooding attack
  - Typically ICMP, UPD or TCP SYN packet types

TCP SYN Flooding

- In 1996 TCP SYN flooding was used to knock out several Internet Service Providers.

  https://tools.ietf.org/html/rfc4987

- The normal TCP connection protocol is as follows;
  1. C → S: SYN
  2. S → C: SYN-ACK (and allocates buffer)
  3. C → S: ACK
- The attack involves sending lots of message 1, which half-open connections, and no message 3 responses.
  - We follow the textbook [SB18] to distinguish between TCP SYN flooding and TCP SYN spoofing on the basis of whether it's the table that's filled or the network that's filled.
    - The defences against them will be somewhat different.

Countermeasures to flooding

- TCP SYN spoofing/flooding are examples of connection depletion attacks.
- These can be launched against other protocols, including SSL.
  - The countermeasures are similar.
- **Time out**:
  - Discard half-open connections (after a time period).
- **Random dropping**:
  - Half open connections are randomly discarded when the buffer reaches a certain percentage of its capacity.
- **Client Puzzles**:
  - We will look at these soon.

- **Syncookies**: (Bernstein and Schenk)
  - Avoid dropping connections when the SYN queue fills up.
  - The Server uses a carefully constructed sequence number in the second message, the acknowledgement but discards the SYN queue entry.
    - The sequence number, a cookie, should be able to be reconstructed without needing the SYN queue entry.
  - If the Server receives a “correct” ACK from the client, the Server can reconstruct the SYN queue entry and then the connection proceeds as usual.
    - A cryptographic checksum is embedded in the server’s sequence number for verification.

Distributed DoS (DDoS)

- Since about 2000 the real problem for DoS is distributed attacks.
- Attacks are launched from multiple networked computers.
- Difficult to defend against:
  - Hard to block multiple IP addresses and still maintain the broad functionality we might need.

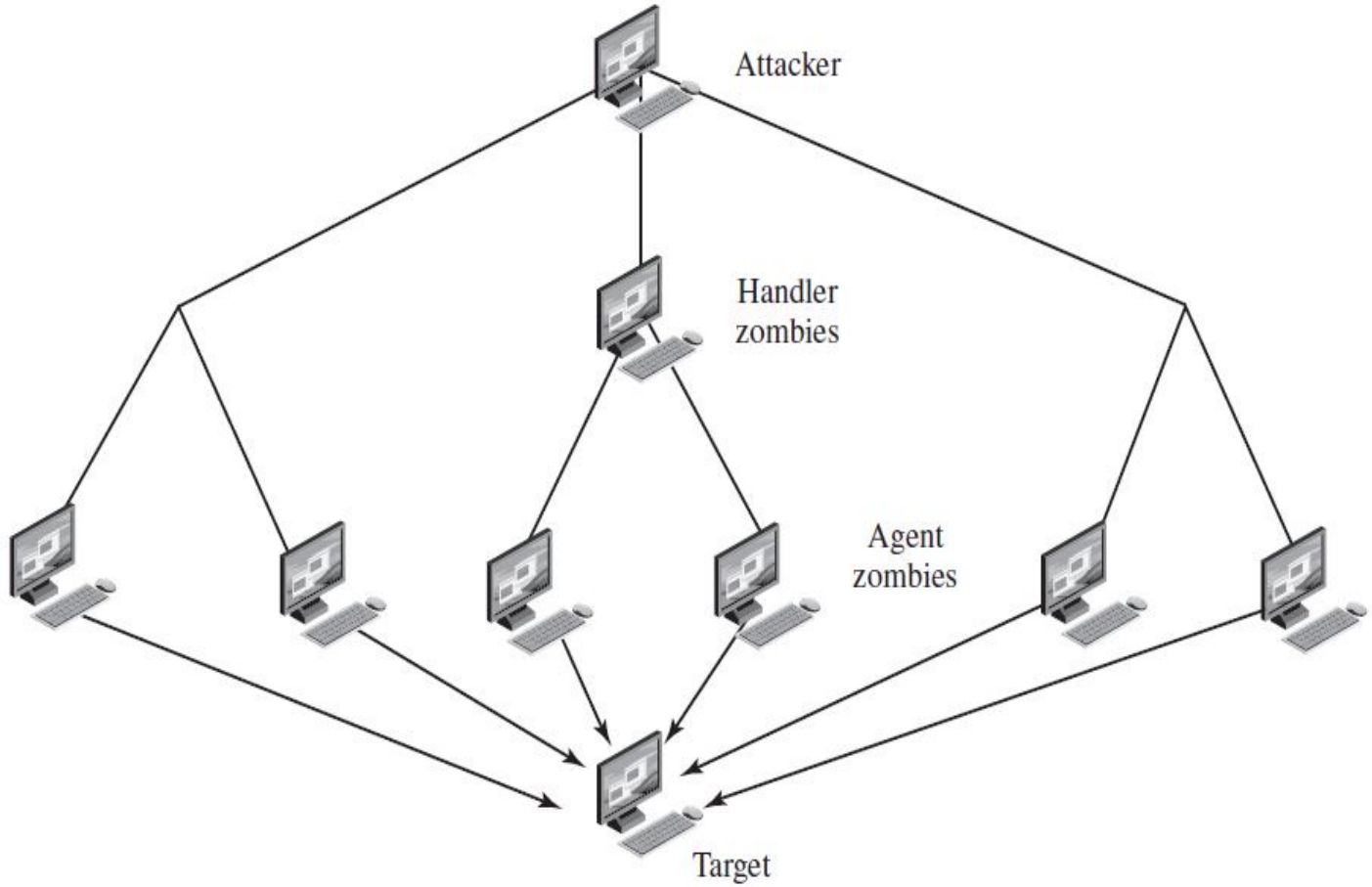


Figure 7.4 **DDoS Attack Architecture**

HTTP floods

- These are good examples of how simple a DoS, more typically a DDoS, can be to launch.
  - Each HTTP request to a website uses some resources on the web server.
  - Lots of HTTP requests → lots of resources used.
    - Do you want to stop people making HTTP requests?
- Slowloris, a specific sneaky example.
  - Sends parts of request headers…
    - … making sure not to terminate them with a blank line and sending them just frequently enough to keep the system expecting the header will finish soon.

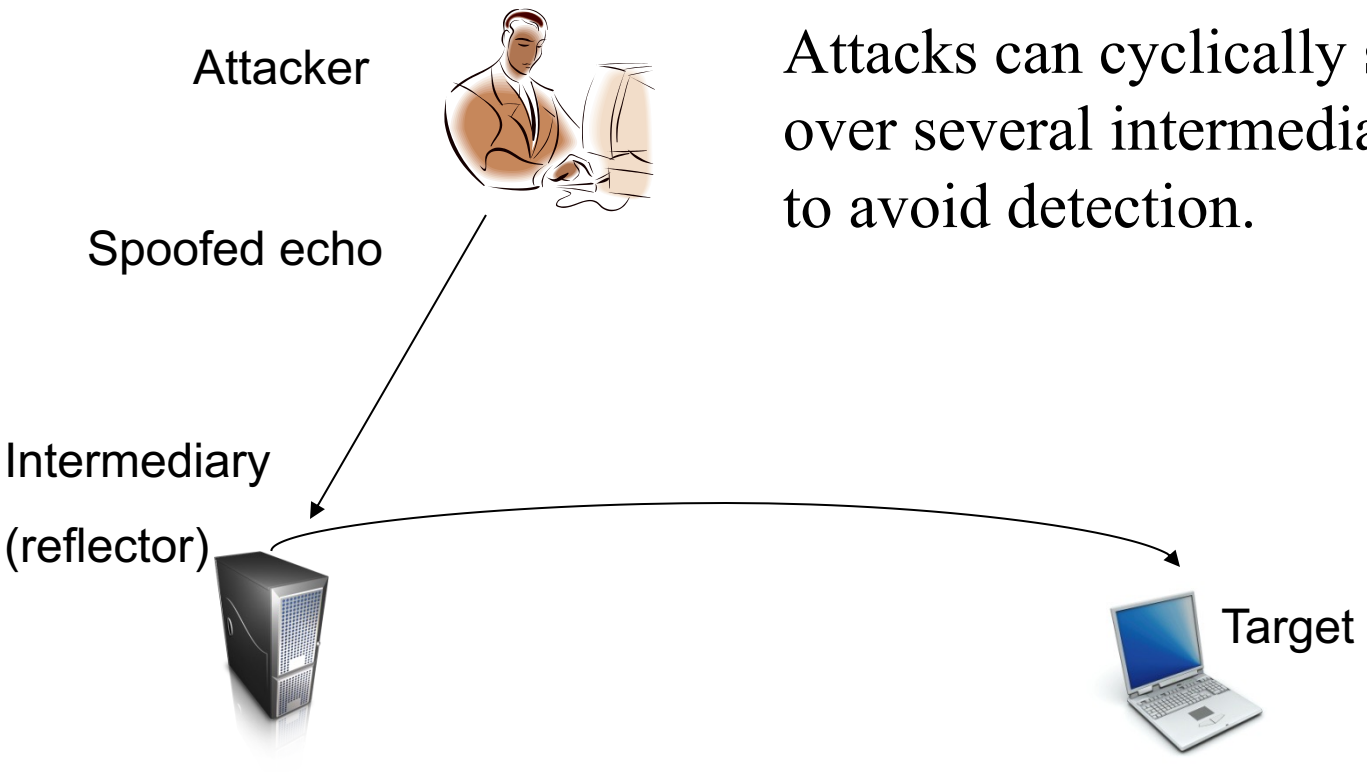
Compromise or exploitation

- In a DDOS attack we make use of multiple systems.
- But those systems are not necessarily compromised.
  - They may be, so form part of a botnet and running programs for the attacker.
  - But they may also simply be providing a service that is exploitable.

Reflection and Amplification...

- There are many legitimate services operating on the Internet that can be exploited in attacks.
  - Reflection and Amplification attacks use such uncorrupted services as intermediaries.
- In reflection attacks the attacker sends packets to a known service on the intermediary with a spoofed source address of the target system.
- The response from the intermediary is directed to the target.

Reflection Attacks



Ideally the attacker would like to use a service that creates a response packet larger than the original request.
Several UDP services have been exploited.
Attacks can cyclically spread the attack over several intermediaries, in an attempt to avoid detection.

Looping reflection: DNS reflection

- A variant creates a self-contained loop between the intermediary and the target.

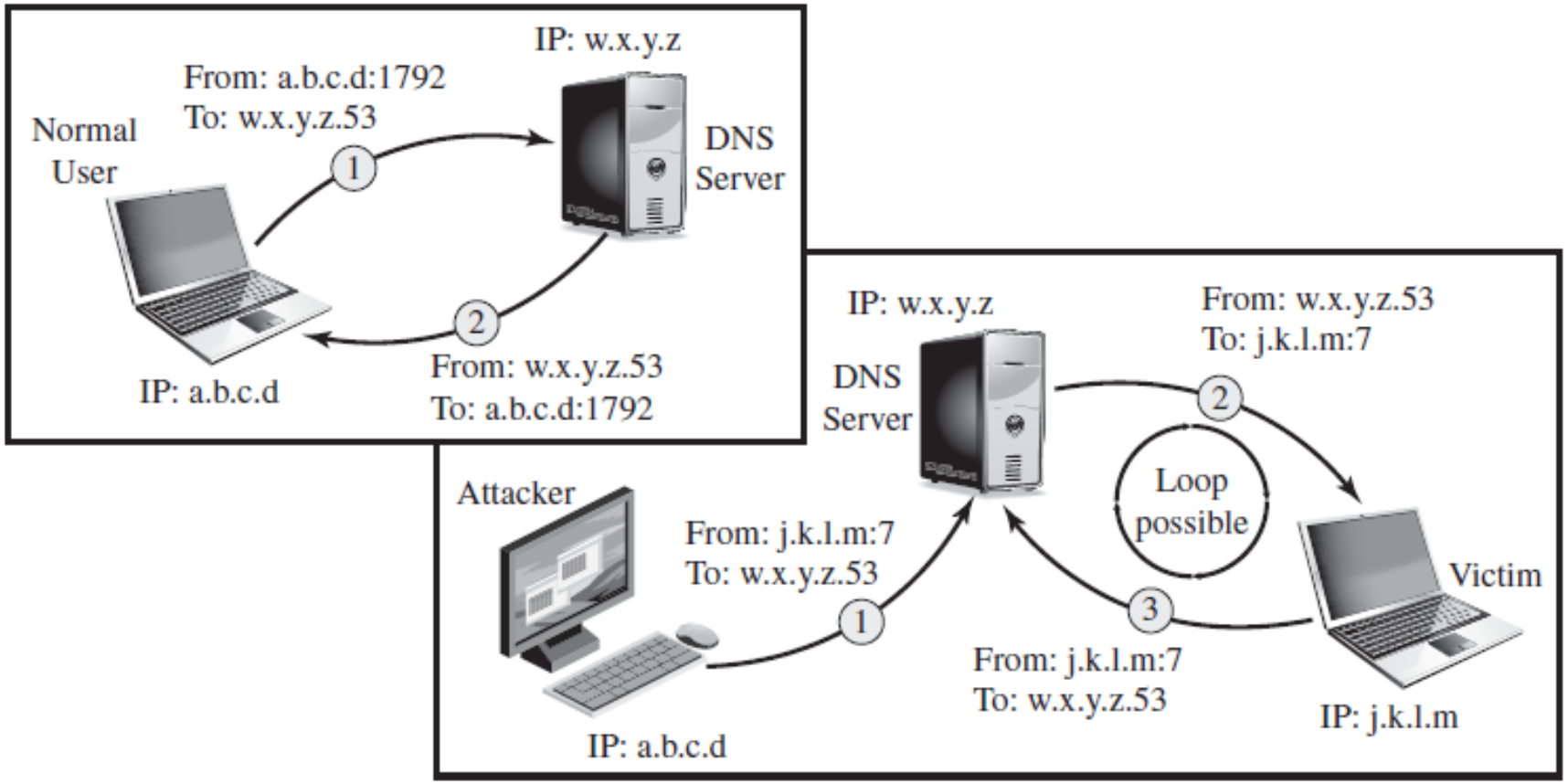
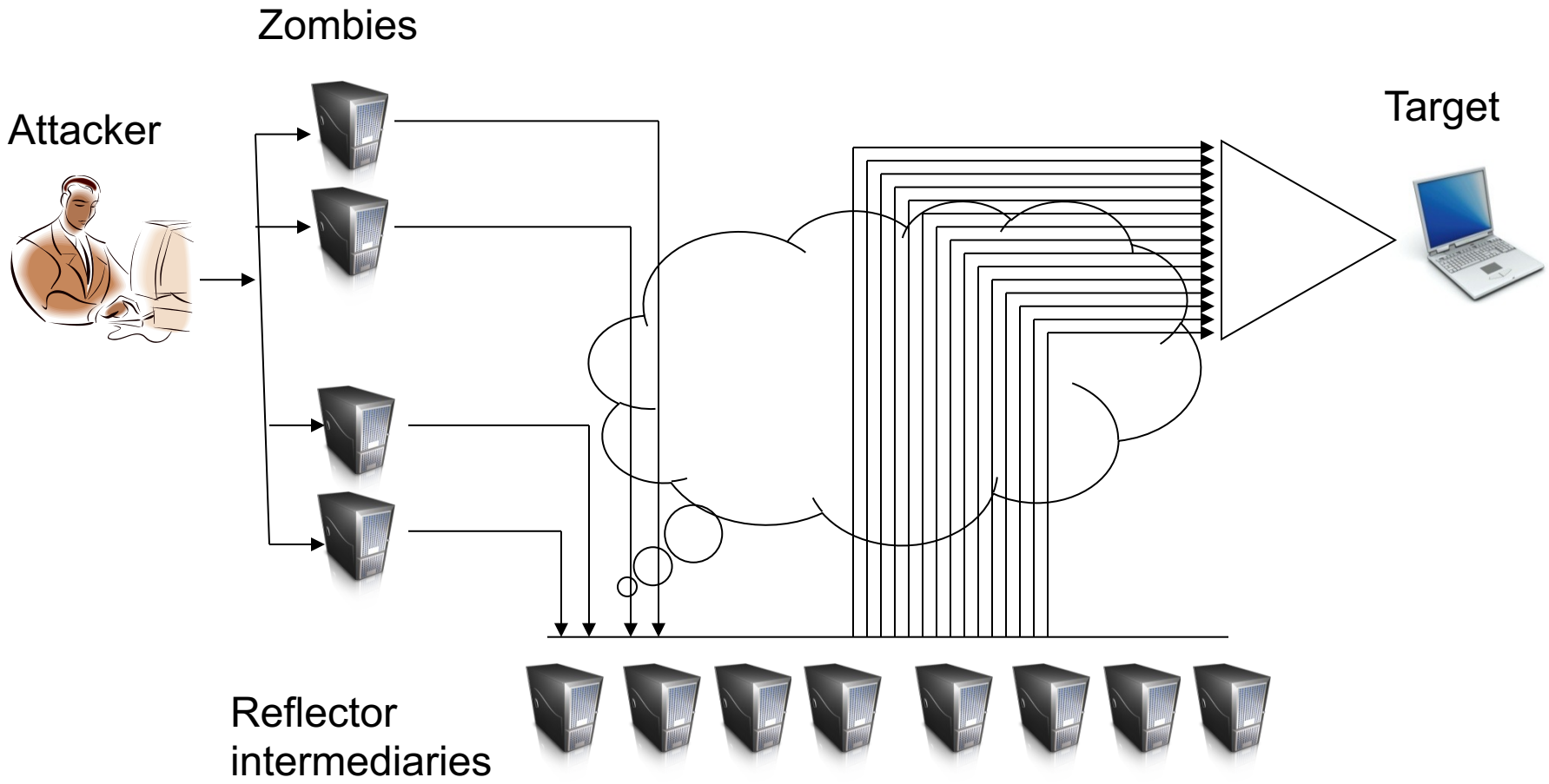


Figure 7.6 **DNS Reflection Attack**

Amplification attacks

Each initial attack packet produces multiple responses.
This could be, for example, by attacking a broadcast address.
Here we have DDoS and reflecting …



DNS Amplification Attacks

- Use DNS requests with spoofed source address being the target.
- Exploit DNS behavior to convert a small request to a much larger response.
  - 60 byte request to 512 - 4000 byte response.
- Attacker sends requests to multiple well connected servers, which flood the target.
  - A moderate flow of request packets is sufficient…

Defenses against DoS

- **Attack prevention and preemption (before the attack):** mechanisms to enable the victim to endure attack attempts without denying service to legitimate clients
  - Filters are implemented to limit the ability of systems to send packets with spoofed source addresses.
  - Access control mechanisms are implemented
  - The filters must be applied to traffic before it leaves the ISP's network, or even at the point of entry to their network, to identify some packets with spoofed source addresses
- **Attack detection and filtering (during the attack):** mechanisms to detect the attack as it begins and respond immediately, to minimize the impact of the attack
- **Attack source traceback and identification (during and after the attack):** attempt to identify the source of the attack as a first step in preventing future attacks.
- **Attack reaction (after the attack):** This is an attempt to eliminate or curtail the effects of an attack.

See [SB18, Section 7.6 and 7.7] for more details

Protection against DDOS …

- Cloudflare offers protection against a variety of DDOS attacks.

  https://www.cloudflare.com/ddos-hub/

Puzzles

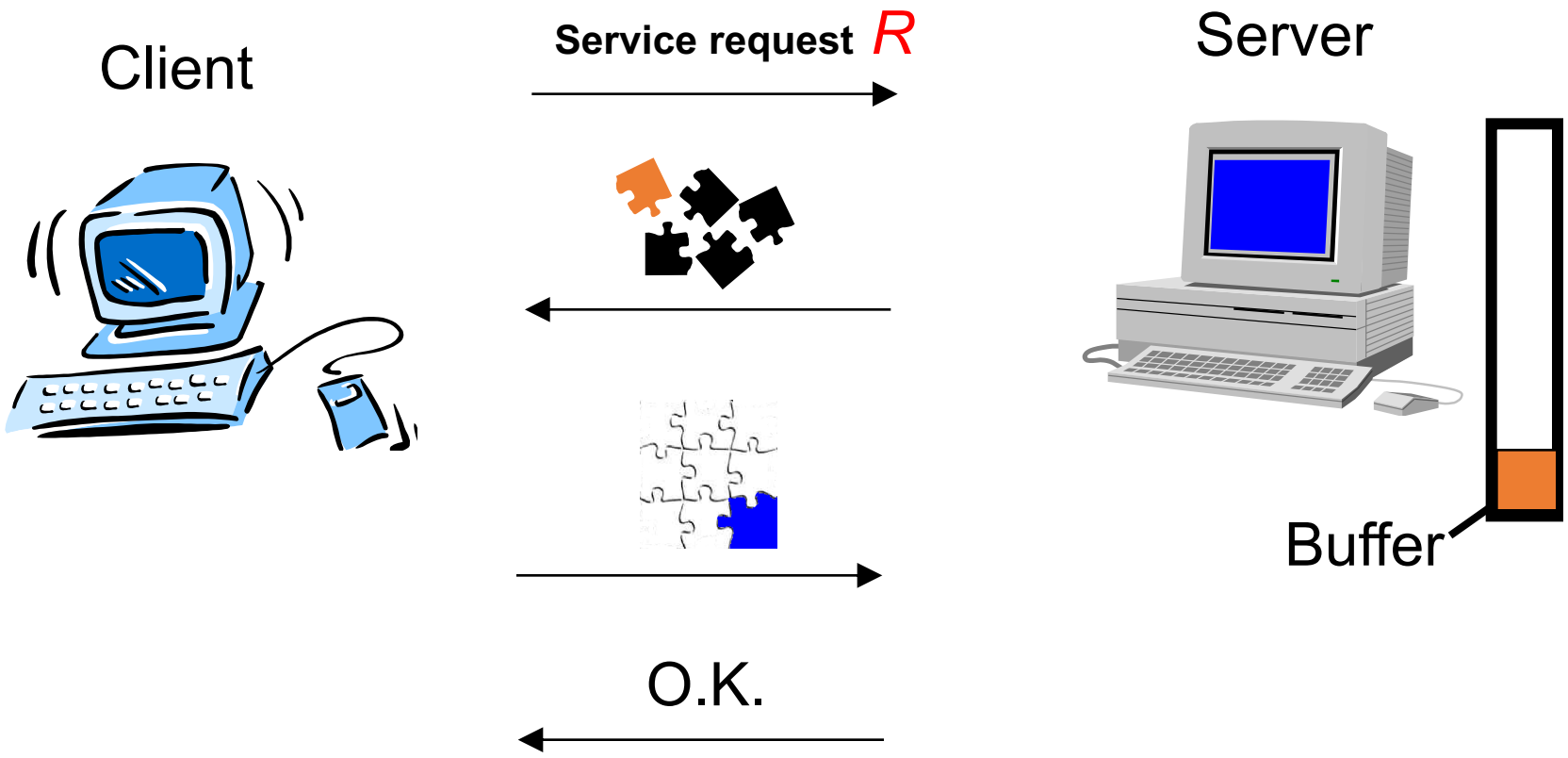
Puzzle activation

- When there is no evidence of a denial of service attack taking place, the server accepts connections normally.
- When an attack on the server is detected, perhaps through an intrusion detection system, the server accepts connections selectively using puzzles.
- Juels and Brainard (1999) proposed the use of Client puzzles.
  - They presented client puzzles in the context of addressing the SYN Flooding attack, more generally for “connection depletion attacks”.

Using puzzles

- To each client, that is, initiated connection, a unique **client puzzle** is proposed.
  - A client puzzle could be a cryptographic problem formulated using time and a server secret.
  - The client needs to submit the correct solution to gain a connection.
  - The idea is that only a live user, not a zombie machine for example, will actively work on answering the problem appropriately.
- We need to be sure that very little work is required before the appropriate response is received.
  - Thus generating the puzzle shouldn’t be difficult.
  - Solving the puzzle shouldn’t be too tough either!

The client puzzle protocol



From Juels and Brainard

Client Puzzle Requirements

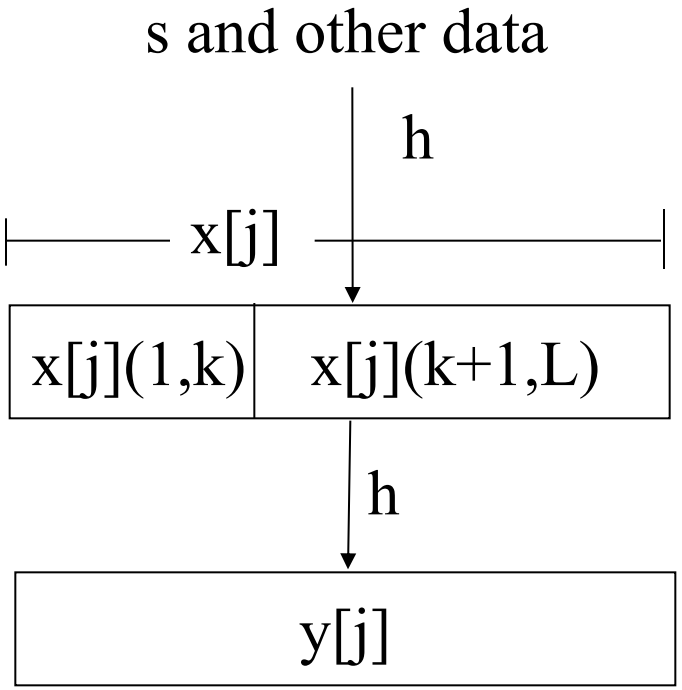
- One particular need associated with many other approaches is that of client-side software to solve client puzzles.
  - This can be built in browsers.
  - It could be available as a plug in.
  - For distributed systems, such as local networks, the storage can be centralised.
- This shouldn’t be considered too much of a problem.

Puzzles and sub-puzzles

- We are going to follow the Client Puzzle Protocol proposed by Juels and Brainard (1999).
  - We are not going to look at all the details.
    - In particular we aren't going to look at their proofs regarding the bounds on the success of an attacker.
- One particularly important aspect though, is the flexibility and scalability of the Client Puzzle approach.
- This is, in part, due to treating a client puzzle P as a number of independent sub-puzzles.
  - Sub-puzzles may have different difficulties.
  - Let us denote the j^{th} sub-puzzle in P by P[j].
  - For a particular client puzzle we have m sub-puzzles.

- A sub-puzzle is constructed as follows:
  - x[j] is a bit-string obtained by hashing a server secret s and a set of service parameters.
    - The hash value is of length L.
  - x[j] is hashed to give y[j] = h(x[j]).
  - The sub-puzzle consists of the pair:

    (x[j](k+1,L), y[j])



- The solution is the missing part of x[j], such that the completed x′[j] satisfies h(x′[j])=y[j].

x[j](1,k) is one solution to the sub-puzzle P[j]

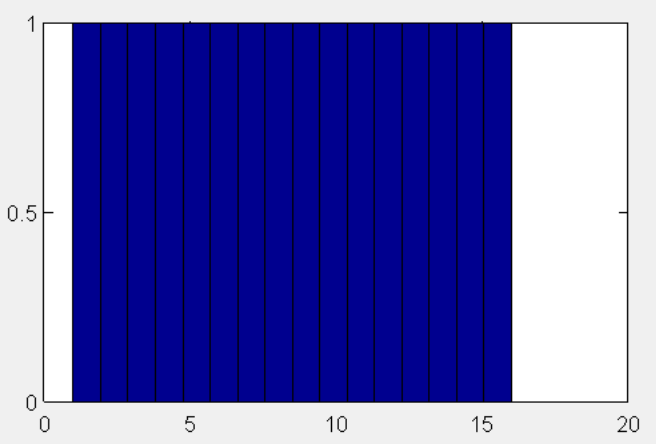
How difficult is the puzzle?

- As we noted the solution to the sub-puzzle P[j] consists of the k missing bits of x[j], that is x[j](1,k), or something else that can be concatenated with x[j](k+1,L) such that the result hashes to y[j].
  - We know there is at least one solution, and likely exactly one.
- Assuming the hash function isn't broken with respect to pre-image resistance, the best we can do is use brute force, and the computational hardness of the sub-puzzle is equivalent to the hardness of searching a space of size 2^k.
  - Each test requires one hash, and the expected number required is $\sim 2^{k-1}$.

- The solution to the complete puzzle P consists of the m different k-bit solutions to all of the component sub-puzzles.
  - m and k are joint security parameters governing the overall difficulty of the problem.
- The expected cost of completing P is ~$m*2^{k-1}$, and is at worst ~$m*2^{k}$.
  - We won't ever need 2^{k} tests for a k-bit puzzle of this sort since there must be a solution and if the first 2^{k}-1 fail the last must be the answer so we don't need to hash again.

M puzzles → Decreased standard deviation

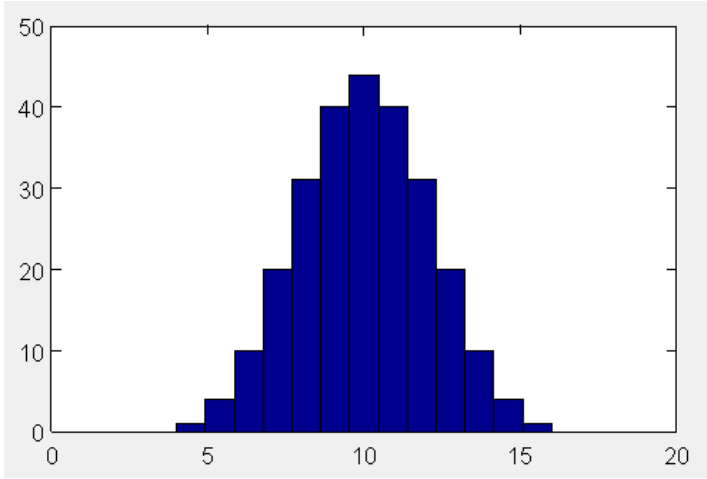
- A set of sub-puzzles with a total expected difficulty equal to that of a single puzzle, will have a smaller standard deviation that the single puzzle distribution.
- Consider a single puzzle (m=1) with k=4.
  - Expected number of hashes $\sim m*2^{k-1}=2^3=8$.
  - Assume we test the last value too then we have a uniform distribution of the number of tests...
- Compare this with m=4 and k=2, with expected number of hashes $\sim m*2^{k-1}=2^3=8$.
  - The distribution looks quite different...



- Single puzzle, k=4.
  - Distribution to the left.
- Expected hashes : **8.5**.
  - Average of the numbers 1 to 16.
- Sample standard deviation: ~4.7610
- Population standard deviation: ~4.6098

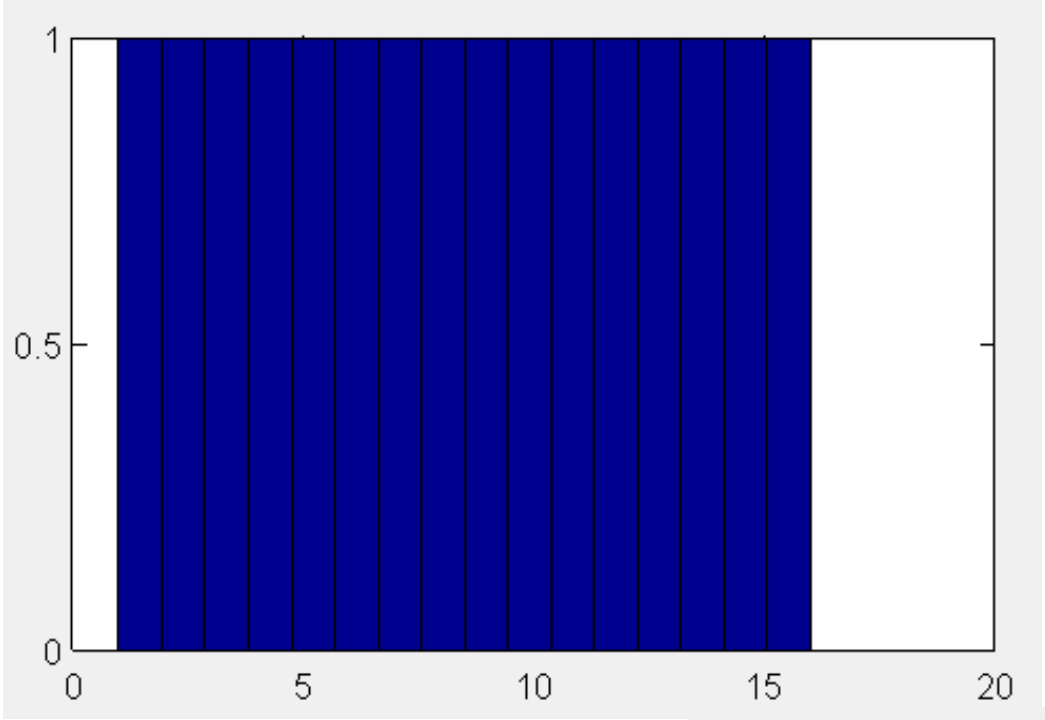
Denominators …
Sample: n-1, population: n

- Four puzzles, k=2 each.
- Expected hashes: 10.
  - **2.5** per puzzle.
  - We need at least 4, at least one per puzzle.
- Sample Standard deviation: ~2.2404
- Population Standard deviation: ~2.2361



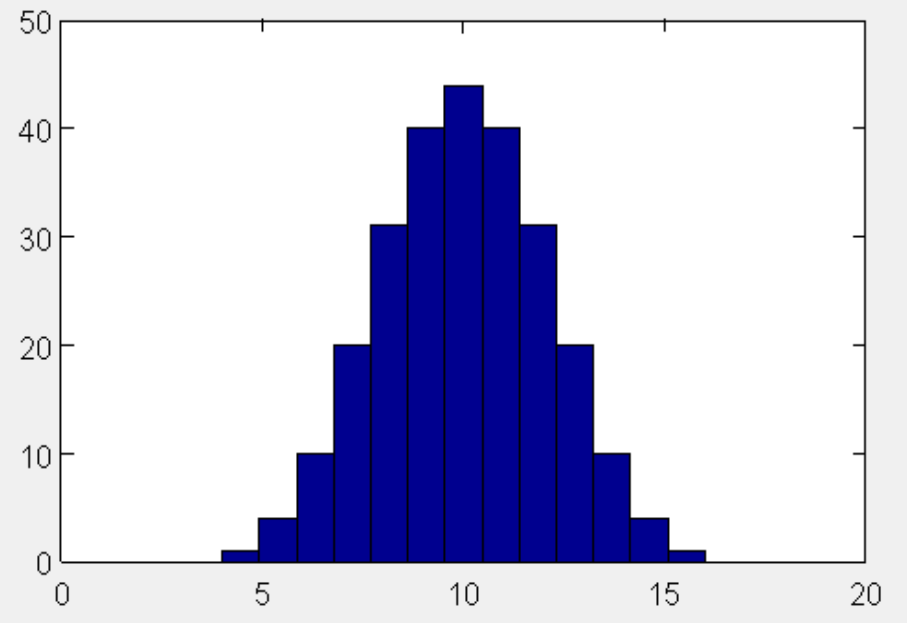
- See the Law of Large Numbers (repeated experiments), and the Central Limit Theorem (combining distributions) → normal.

Number of possible cases needing that many hashes



Number of hashes needed

Number of possible cases needing that many hashes



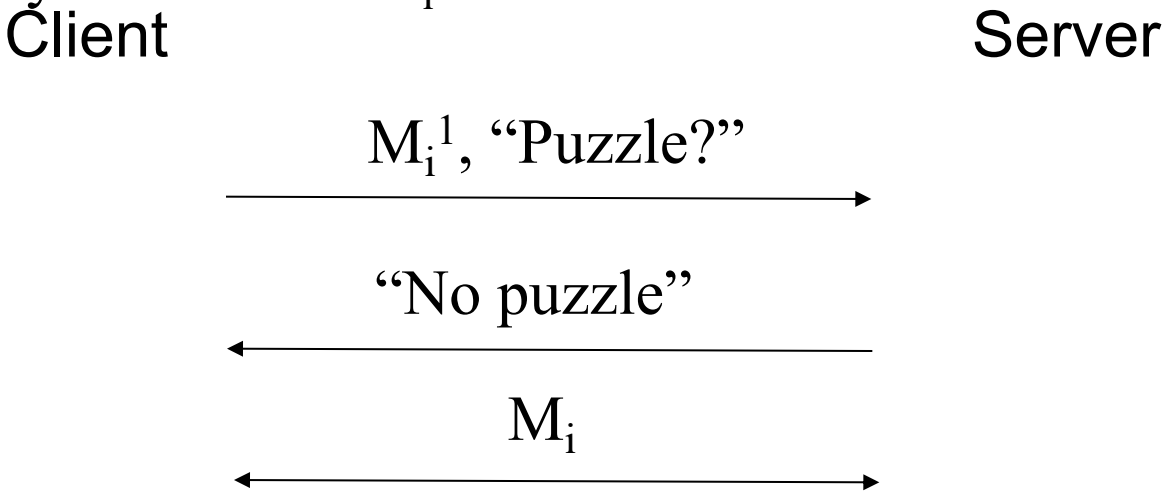
Number of hashes needed

The distributions …

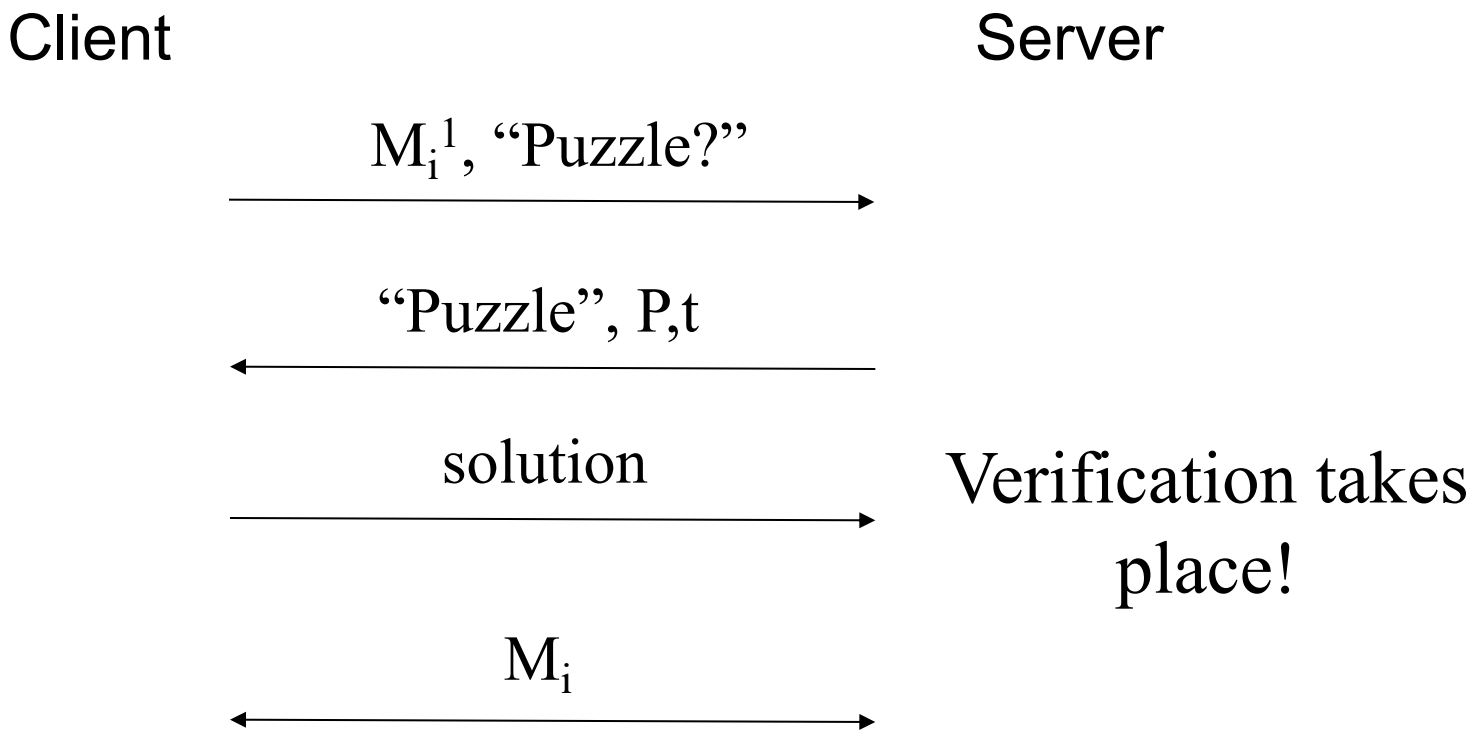
#hashes needed	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Cases for m=1, k=4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cases for m=4, k=2	0	0	0	1	4	10	20	31	40	44	40	31	20	10	4	1

Back to the puzzle itself: What's in the initial information?

- To begin the Client submits:
  - A message M_i^1, which is the 1st message of the i^{th} run of a protocol **M** to that server.
    - This information will be used in generating the puzzle.
  - Possibly a request to be informed if there is a puzzle to be solved?
- If operations are as normal the Server replies no and continues the protocol.
  - The Server may record that M_i is allowed.



- If puzzles are active the server calculates a series of sub-puzzles with $x[j]=h(s,t,M_i^1,j)$, where:
  - s is the server secret.
  - t is a timestamp:
    - It can provide integrity regarding the time.



So what puzzle information is stored?

- The solution from the client is:

$$(\{x[j](1,k) : 0...j...m-1\}, M_i^1, t)$$

- The puzzles themselves are effectively stateless, so the answer is nothing. ☺
  - The solution contains all the information the server needs other than their own server secret.
  - Verification involves the server re-calculating x[j], and comparing the current time against t.

Overcoming Puzzles

- The generation of puzzles cannot be too computationally demanding, since the generation is carried out when clients initial a communication.
- A rogue client sends some Hello messages:
  - For each, the Server has to prepare a puzzle. ☹
  - The rogue doesn't answer, but keeps sending new "Hello" messages…
- It is difficult to prevent this kind of attack when it is launched as a distributed DoS. ☹
- There is also always the problem of balancing how long a puzzle should take against the resources of the entity to solve the puzzle.