Kesidis, Brooks, & Rai, 2006). COSSACK (Papadopoulos, Lindell, Mehringer, Hussain, & Govindan, 2003) and DefCOM (Mirkovic and Reiher, 2005) organize detectors at the victim side and transmit an alert to the filter or to the rate limiter that is placed at the side of the source. Chen and Song (2005) suggested a perimeter-based scheme for Internet service providers to enable a service to defence DDoS attacks for their customers. They suggested a plan according to edge routers to realize the sources of the flood off attack traffics.

Change-point detection theory is used by most of the researchers to distinguish any unusual traffic distributed through the Internet due to DDoS attacks (Blazek et al., 2001; Chen & Hwang, 2006a,b; Peng, Leckie, & Ramamohanarao, 2003; Wang, Zhang, & Shin, 2004). Because of the lack of precise statistics to explain about the prechange and postchange traffic distributions, there would be a development of a nonparametric cumulative sum (CUSUM) scheme for its low-computational intricacy (Blazek et al., 2001). The short-term behavior shifting from a long-term one is monitored by the scheme monitors. If the cumulative difference arrives at a specific threshold, there would be an attack alert. A central DDoS defense scheme was proposed by Wang et al. (2004) to check the change points of the gateway level. A similar approach was taken by Peng et al. (2003) in the source IP addresses monitoring.

Here, we are going to review two remarkable efforts, which are used more recently and they are client puzzle theory and collaboration of detection of DDoS attacks over multiple networks.

## 2.2 CLIENT PUZZLE THEORY

Much of the presented solution to overcome DDoS attacks includes enforcing servers to produce a puzzle, when the servers feel under attacks and clients must resolve these puzzles and send it again to servers as shown in Figure 2.4. Verifying the solution by servers is the next step. Generally, framework applied to explain about any certain client puzzle's scheme includes main step proposed by Jeckmans (2009) and they are: *Setup*, *PuzzleGen*, *PuzzleSol*, and *PuzzleVer*. In general, they include the algorithms applied to open the requirements for puzzles formation, produce the client puzzles, solve the client puzzles, and verify the solution of the client puzzle for the client puzzle system correspondingly.

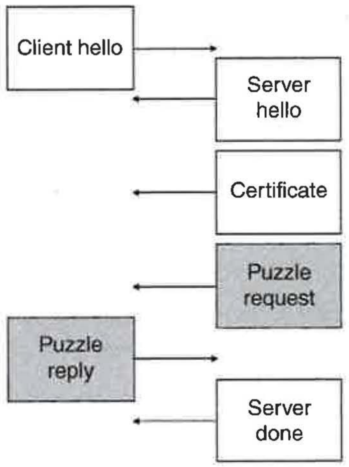

*Fig. 2.4. General client puzzle handshaking.* From Dean and Stubblefield (2001).

*Setup(k)*

It is run by the server *S*. It takes a security parameter *k* as input and output *params* as the system parameter, which would be an implicit input to the following algorithms. This system parameter is varied across different puzzle schemes.

*PuzzleGen(mk, req)*

It is run by the server *S*. It takes a server secret *mk* and additional request information *req* received from the client *C* as input and outputs a puzzle *puz* and additional information *data,* which required by the server for verification purpose. The puzzle *puz* is sent to client *C*.

*PuzzleSol(puz)*

It is run by the client *C*. It takes a puzzle *puz* that received from the server *S* and outputs a puzzle solution *sol*. The puzzle solution *sol* is sent to the server.

*PuzzleVer(data,mk, sol)*

It is run by the server *S*. It takes the puzzle information *data*, the server secret *mk,* and a puzzle solution *sol* received from the client *C* as input. It outputs 1 if *sol* is correct or 0 otherwise.

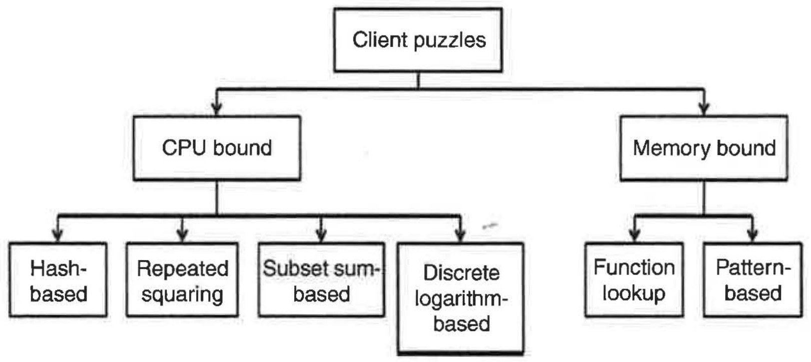

*Fig. 2.5. Classification of client puzzles schemes.* From Mirkovic and Reiher (2004).

Based on Figure 2.5, two main groups, CPU-bound puzzles and memory-bound puzzles can be regarded as categorization of the existing client puzzles schemes. While memory-bound puzzles need some memory lookups to be performed to solve them, CPU-bound puzzles are the puzzles that require CPU cycles to be performed when solving them. Therefore, the solving speed would deeply rely on the processor and machine memory speed.

Merkle (1978) suggested the first notion of cryptographic puzzles incorporated into network authentication protocol. Based on Merkle's idea, there are several client puzzles schemes, which are applied to frustrate DoS resource depletion types of attacks (Karame & Čapkun, 2010; Dean & Stubblefield, 2001; Feng & Kaiser, 2007a, b).

Several issues exist in performance of client puzzle, which have to be carefully weighted up as the following:

1. The price of puzzles' formation and answer confirmation used by the server has to be more costly than the price of solving the puzzle used by the client.
2. The complexity of puzzle ought to be modifiable.
3. There should be statelessness in the puzzle, so that the server may not need to keep any link state prior to the offering of puzzle key.
4. For the clients to solve the puzzle, only limited amount of time is given.

5. There should be infeasibility in precomputation attacks.
6. There is uniqueness in the puzzles, that is, by having the previous puzzles solved, it is not possible to solve new puzzles.
7. There must not be any flooding attack vulnerabilities in puzzle creation and issuing.
8. Susceptibility to any bypass issue must not exist in puzzle mechanism.
9. There must be a differentiation in puzzle issuing between legitimate users and malicious users, therefore fine users exhibit spiteful behavior in more complex puzzles.

The analysis of a client puzzles scheme can be based on four major factors:

1. *Server cost.* It consists of precomputation cost, construction cost, and verification cost. Generally, it determines the computational effort on the server's machine.
2. *Client cost.* It identifies the computational effort on the client's machine.
3. *Hardness granularity.* It measures the relationship between the workload needed to solve the puzzle and the puzzle difficulty. Fundamentally, there are three different types of hardness granularity; linear, polynomial, and exponential. The preferred granularity in puzzle scheme is the linear granularity while the worst case to deal with is the exponential granularity.
4. *Deterministic or nonparallelizability.* It represents the ability of the puzzle scheme to resist parallel computation attack. In other words, parallelizable puzzles can be distributed by coordinated adversaries to high performance computing machines, in order to perform parallel computation on the solutions of the puzzles.

Since the use of client puzzles as a mechanism to countermeasure DoS attack, most of the research has been focused on improving its security rather than developing a practical client puzzles system. The security improvement and deployment complexity of existing client puzzles schemes are not balanced. The limitations of existing client puzzles schemes can be summarized as follows:

1. Nonparallelizable is one of the important features a client puzzles scheme should have, in order to thwart the DoS attack efficiently

It can be noticed that all the hash-based puzzles suffers from this shortcoming, no matter how much they have been improved for years.

2. Probabilistic behavior of a client puzzles scheme allows an unknown amount of solutions that can be obtained at the first try of solving it. The chances of such probabilistic occurrence will increases when the number of trial increases. This is not a desirable circumstance that should occur in a real application of client puzzles.
3. Low-differentiability difficulty setting of a client puzzles system unable to distinguish legitimate users from malicious users. This makes the system unfair to those legitimate users who pay the same cost as the malicious users. For a client puzzles system to defeat DoS, the pricing of the puzzles must be fair across different kind of users.
4. Complicated deployment causes the client puzzles systems, which are not preferred in real life application. Most of the secure client puzzles systems require the adoption of a specific client software or modification of a particular protocol in order to work. Such poor adaptability leads to inconveniency problems, such as compatibility and privacy protection issues. Only robust system that can survive in today advanced technology.

## 2.3 CPU-BOUND PUZZLES

CPU-bound puzzle is the largest class of client puzzle as compared to memory-bound puzzle. It requires the clients to commit their processor's resources when solving the puzzles. In the following sections, we will first describe the algorithms of the puzzles schemes and then analyze their performance.

### 2.3.1 Hash-Based Puzzle

Hash-based client puzzle schemes require the client to reverse a cryptographic hash function, in order to solve the puzzle. When utilizing hash function in client puzzle scheme, a part of the reverse is provided so that the client is able to solve the remaining part by brute force (Juels & Brainard, 1999).

#### 2.3.1.1 Juels and Brainard's Hash Puzzle

In Juels and Brainard's approach (Juels and Brainard, 1999) to defend against connection depletion attacks, the cost of each connection request is a unique cryptographic problem called client puzzle. When no attack is taken place, the server accepts connection requests as usual. However, when the server observes an attack, each client that wishes to establish the connection must solve the client puzzle given.

#### 2.3.1.2 Aura et al.'s Hash Puzzle

Aura et al. (2001) pointed out the weakness of Juels and Brainard's puzzle that neglects the DoS attacks against the authentication protocols. For the improvement on the efficiency of the client puzzles, they have reduced the length of the puzzle and its solution, minimized the cost of verification of the solution, and broadcasted the puzzles if the network allowed. To make the scheme more secure, signature is used to authenticate the communication between server and client.

#### 2.3.1.3 Parallel Hash Puzzle

Parallel hash puzzle is originated from the single hash puzzle by Juels and Brainard (1999). In this scheme, a puzzle is divided into multiple smaller hash-based puzzles and a single client is asked to solve multiple hash-based puzzles in parallel. The purpose of doing this is to obtain linear puzzle hardness granularity by utilizing multiple hash puzzles in parallel where the client is responsible to find the missing bits of a pre-image of a hash function whose output is given.

#### 2.3.1.4 Hinted Hash Puzzle

Hinted hash puzzle is proposed by Feng et al. (2005) to solve the puzzle hardness granularity issue. As, how hinted hash puzzle was named, a hint will be provided along with the puzzle to be solved by the client. The hint is used to achieve linear hardness granularity, which has the same purpose of multiple hash-based puzzles proposed by Juels and Brainard. By adjusting the accuracy of the hint, which is a single value that tells the client where the answer lies, the difficulty of the puzzle is adjusted.

#### 2.3.1.5 Chained Hash Puzzle

A chained hash puzzle, proposed by Groza and Petrica (2006) consists of a number of hash-based puzzles, which may be solved only in a precise manner. In other words, the puzzles are chained together in a linear or random fashion making the puzzle solution dependent on each other based on how they are chained.

#### 2.3.1.6 Targeted Hash Puzzle

A different kind of public work functions known as targeted cryptographic hash function reversal has been proposed by Feng and Kaiser (2007a, b). For a public work function to be practical, it must fulfill four important properties, which are fast issuing, fast verification, flexible adaptability, and precomputation and replay resistance.

### 2.3.2 Repeated Squaring Puzzle

The idea of time-release crypto was proposed by Rivest et al. (1996) to hide a message that cannot be revealed by anyone even the sender until, a predetermined amount of time has passed. This idea is then applied onto client puzzles called timelock puzzles or repeated squaring puzzles in which a precise amount of time is required to solve the puzzles. To solve a repeated squaring puzzle, the client has to perform modular squaring repeatedly and the number of repetition is determined in the puzzle.

### 2.3.3 Discrete Logarithm-Based Puzzle

Waters et al. (2004) proposed an idea of outsourcing the puzzle creation and distribution to a bastion, they use a new one-way function based on the discrete logarithm problem that can replace the role of hash function in their client puzzles.

### 2.3.4 Subset Sum-Based Puzzle

Another limitation of previous puzzle schemes is lack of parallel computation attack resistant, thus susceptible to DDoS attacks. Realizing the importance of nonparallelizability property in a puzzle, Tritilanunt et al. (2007) proposed subset sum puzzles. It was claimed by Tritilanunt et al. that subset sum puzzles is a new technique that offers simple construction and low cost verification as hash-based puzzles, it also has a nonparallelizable characteristic. Basically, it is just a subset sum problem that is used to be solved as a client puzzle. The fastest algorithm at the time of this writing to solve this problem is one of the lattice reduction algorithms called the LLL algorithm

(Lenstra et al., 1982). Another intention of requiring clients to implement the LLL algorithm in solving subset sum puzzles is its nonparallelizable property. Two different LLL lattice reduction techniques that can be used to solve a subset sum problem are backtracking or brute force searching and branch and bound method. Both have been used to make a performance comparison with LLL algorithm in (Tritilanunt, 2010).

### 2.3.5 Modified Time-Lock Puzzle

The traditional time-lock puzzle was first proposed by Rivest et al. (1996). The term time-lock is used by Rivest to describe the primer property of the puzzles in which a predetermined amount of time is required before the puzzles are available to be solved. It is also known as repeated squaring puzzles, as it was first employed in a client puzzle scheme, which a number of repeated squaring needs to be performed to solve it. There are some improvements that have been done on time-lock puzzles scheme by Feng and Kaiser (2010) to make it a more practical and secure scheme. The design of modified time-lock scheme separates the costs into puzzle verification cost and puzzle issuance cost. Basically, it is the combination of single hash puzzle and time-lock puzzle. The single hash puzzle will adopt in the puzzle issuer and the time-lock puzzle is responsible for the puzzle solver. This scheme is designed in this way because single hash puzzle offers linear hardness granularity and cheaper puzzle creation cost, which makes it the most suitable candidate as a puzzle issuer. As for time-lock puzzle, it is deterministic in nature and has parallelization resistance, which is the only scheme that satisfies the requirements as a puzzle solver. In summary, the modified time-lock puzzle has joined the strength and eliminated the shortcomings of single hash puzzle and time-lock puzzle.

## 2.4 SUMMARY OF CPU-BOUND PUZZLES

Cryptographic hard problems such as discrete logarithm problem, Diffie-Hellman problem, subset sum problem, and RSA problem have played a vital role in client puzzle scheme. Similarly, calculate the inversion of a one-way function such as cryptographic hash function, which is also considered as a mathematical hard problem, has been utilized to form the basis for most of the CPU-bound puzzles schemes. It can be observed that the first hash puzzles proposed by Juels and Brainard (1999) has some limitations, which then leads to a lot of research done to improve their scheme. All of these improvements are tailored, according to the properties of a good puzzle scheme.

First, the hardness granularity can be improved by distributing multiple client puzzles for the client to solve in parallel (Juels and Brainard, 1999). However, the best to granularity is by providing a range of possible solutions for the client to search in Feng et al. (2005) when the server's cost is taken into consideration. A stateful puzzle scheme might cause the server to be overloaded by the puzzle information. To make a puzzle scheme becomes stateless, the puzzle relevant information should not be stored in the server, and instead, they are sent to the client and have the client to send them back together with the solution. This must be done carefully without sending the information that could breach the security of the scheme. Even though most of the limitations in hash puzzle have been countered, the probabilistic nature of hash function causes it vulnerable to parallelization attack until today. Repeated squaring puzzle is one of the schemes that can resist parallel computation attack. Due to its high-storage cost and high-computational cost resulted from storing RSA modulus and calculating modular exponentiations respectively, it is considered impractical. Subset sum-based puzzle can utilize LLL algorithm's nonparallelizability to defend against parallelization attack. Nonetheless, subset sum-based puzzle suffers from being stateful and has a hardness granularity that is polynomial in nature. Last, modified time-lock puzzle has the best overall performance. It simply combines the hash-based puzzle scheme and repeated squaring puzzle scheme into one so that the advantages from both schemes are inherited, at the same time eliminating their disadvantages.

## 2.5 MEMORY-BOUND PUZZLES

Due to the great disparity in processing speed than in memory speed in today's computing hardware, memory-bound puzzles are introduced to make the puzzle solving less dependent on hardware. Primarily, the client is required to perform some look-ups in a precomputed database.

### 2.5.1 Function Look-Up Puzzle

Abadi et al. (2005) proposed a family of moderately hard memory bound functions that can be used to perform searching on a look-up table, which in turn increase the speed of solving a client puzzle. In Abadi's definition, a memory-bound function works on the ineffectiveness of the caches available in a machine. This can be done by having the memory-bound function behaves randomly, while accessing locations in a large region of the memory. In this situation, memory latency is used to measure the performance of a machine instead of memory throughput, which is less uniform. In short, a good memory-bound function will have a uniform cost across systems from high-end to low-end.

### 2.5.2 Pattern-Based Puzzle

A new memory bound puzzle construction based on heuristic search for sliding tile problem (Loyd & Gardner, 1959) using a look-up table called pattern database or heuristic table was proposed by Doshi et al. (2006). Basically, sliding tile problem is about finding a path to slide the tiles, which have been arranged in a specific pattern on a grid from a starting state to the goal state. The path used is not necessary to be the optimal path. In this situation, existing algorithms that were used to obtain the shortest path, they cannot be used always to find the solution. Optimal solutions to the $4 \times 4$ sliding tile problem can be acquired by using pattern databases introduced by Culberson and Schaeffer (1996). The main purpose behind incorporating pattern databases into the client puzzle scheme is that they consume more memory rather than processor resource, when search time is reduced desirably. Thus, puzzle-solving process is memory-bound.

## 2.6 SUMMARY OF MEMORY-BOUND PUZZLES

Existing memory-bound puzzles schemes do not eliminate the parallelization attack efficiently. All of them only offer polynomial hardness granularity. Due to their probabilistic behavior, the puzzle can be solved at first try. Their designs concern only the memory latency or memory speed by assuming an average memory size on clients' hardware. There is a possibility that some clients with huge memory size can store more pre-computed databases for solving the puzzle. So, attempt to slow down the clients by forcing them to renew their databases only work on those with smaller memory size.

## 2.7 COMPARISON OF EXISTING CLIENT PUZZLES SCHEMES

The abbreviations that describe the elements to be compared are presented in Table 2.1. The performance overview of existing client puzzles scheme is shown in Table 2.2. Both tables are adapted from Jeckmans (2009) with minor modifications made. Targeted hash puzzle scheme is added to the original Table 2.2. The puzzle cost is divided into puzzle creation cost and puzzle verification cost. The communication cost is approximated as the total cost of the puzzle scheme. Parallel computation resistance describes the ability of the scheme to defend against parallelization attack. Hardness granularity represents the relationship between the workloads needed to solve the puzzle and the puzzle difficulty with linear being better and exponential being worst. All schemes will have either deterministic nature or probabilistic nature. The comparison of existing client puzzles scheme can be summarized in the subsequent section. Most schemes are probabilistic in nature and vulnerable to the parallel computation attack. This indicates that these schemes will fail to defend against DoS and DDoS attack, which is launched by high-end machines.

Table 2.1 Abbreviations of Comparative Element

| Header | | Operation | |
|---|---|---|---|
| DN | Deterministic nature | H | Hash function |
| PC | Puzzle creation | M | (Modular) multiplication |
| PY | Puzzle verification | E | Modular exponentiation |
| PR | Parallel computation resistance | C | Checksum function |
| HG | Hardness granularity | F | One-way many-to-one function |
| LS | Long-term storage | Size | |
| SS | Short-term storage | k | Security parameter |
| CC | Communication cost | h | Hash value |
| | | r | Modulus of RSA |
| Value | | | |
| l | Puzzle difficulty | s | Maximum puzzle difficulty |
| n | Number of puzzles | t | Maximum number of puzzles |

Table 2.2 Comparison of Existing Client Puzzles Schemes

| **Client Puzzle Schemes** | **PC** | **PV** | **LS** | **SS** | **CC** |
|---|---|---|---|---|---|
| Hash-based-Juel | *2H* | *H* | *k* | – | *3h+2k+2k^2* |
| Hash-based-Aura | – | *H* | – | *3k* | *3k* |
| Parallel hash | *2n*H* | *n**H | *k* | – | *3n*h+2k+2k^2* |
| Hinted hash | *2H* | *H* | *k* | – | *3h+2k+2k^2* |
| Chained hash | *2n*H* | – | *k* | *n*h* | *3n*h* |
| Targeted hash | – | *H* | *k* | – | *h+k* |
| Discrete logarithm | *E* | – | *k* | *k* | *3k* |
| Repeated squaring | *M* | *2E* | – | *3r* | *3r* |
| Subset-sum | *H+1*M* | – | *s*h+k* | *h* | *3h* |
| Modified time-lock | *2H* | *2E* | – | *3r* | *3r* |
| Function look-up | (1−1)*F+C* | – | – | *k* | *3k* |
| Pattern-based | 1**C+H* | *H* | (*t*+1)*k* | – | *h*+(2*n*+1+1)*k* |
| **Client Puzzle Schemes** | **PR** | **HG** | **DN** | | |
| Hash-based-Juel | no | exponential | no | | |
| Hash-based-Aura | no | exponential | no | | |
| Parallel hash | no | polynomial | no | | |
| Hinted hash | no | linear | no | | |
| Chained hash | some | polynomial | no | | |
| Targeted hash | no | linear | no | | |
| Discrete logarithm | no | linear | no | | |
| Repeated squaring | yes | linear | yes | | |
| Subset-sum | yes | polynomial | yes | | |
| Modified time-lock | yes | linear | yes | | |
| Function look-up | some | polynomial | no | | |
| Pattern-based | some | polynomial | no | | |

In order to prevent parallelization attack, the puzzle should be constructed in such a way that its workload cannot be distributed. Only the hash-based puzzles and modified time-lock puzzles schemes offer inexpensive server cost. We could observe that most schemes have cheap puzzle creation cost but expensive puzzle verification cost. Another observation is the improvement of hash-based puzzle in linearity. A hash-based puzzle can have a linear hardness granularity by providing some hints to the client to search for the solution. This has been done with the hinted hash puzzles. Most schemes require short-term storage and long-term storage space to keep track of some puzzle states. Puzzle scheme that requires short-term storage is a stateful scheme. As long as storage is needed in the scheme, the server's storage could be exhausted by attackers.

After the comparison among the existing client puzzles schemes, modified time-lock puzzle is found to be the only scheme that exhibits good overall performance. Therefore, it is chosen as the scheme to be implemented in this project.

Because of the application of client puzzles as a countermeasure against DoS attack, instead of developing a practical client puzzles system, most of the investigation has been centered on improving its security. There is any balance in the improvement of security and operation complexity of existing client puzzles schemes.

In general, client puzzle scheme cannot detect the attacker source and also imposes both server and client heavy unnecessary computation. This is the inherent weakness property of client puzzle.

## 2.8 COLLABORATION OF DETECTION OVER MULTIPLE NETWORKS

An algorithm was proposed by Chen, Hwang, and Ku (2007) in which a distributed method is proposed to identify DDoS flooding attacks at the traffic flow level. The defense system is appropriate for performance over the core networks operated by ISPs. Some traffic fluctuations are detectable at Internet routers or at the gateways of edge networks at the early stage of a DDoS attack. There is a development of a *distributed change-point detection* (DCD) architecture applying *change aggregation trees* (CAT). The chief idea includes detecting abrupt traffic changes over multiple network domains at the initial time.

A new mechanism, called CAT is applied by the algorithm that proposes a DCD architecture.

The flooding traffic that is sufficiently large to crash the victim machine through communication buffer overflow, disk exhaustion,

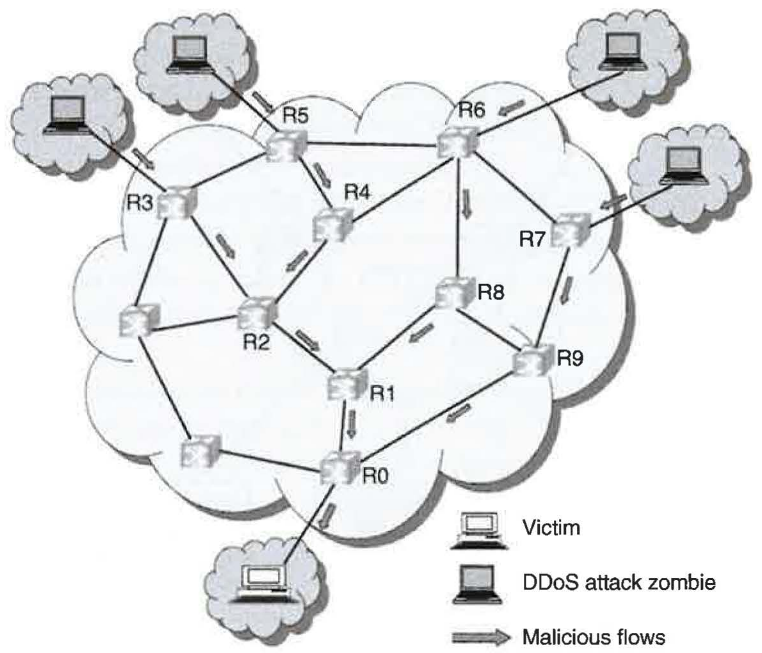

*Fig. 2.6. A large number of zombies generate traffic superflow by DDoS flooding attacks toward a common destination as victim host.* From Chen et al. (2007).

connection link saturation, and so forth. Figure 2.6 illustrates a flooding attack that is initiated from four zombies. The *attack-transit routers* (ATRs) identify the abnormal surge of traffic at their I/O ports. The attraction for the victim is by the end router *R0* in Figure 2.6. All the attack flows from the superflow homing toward the end router.

A superflow includes all packets destined for the similar network domain from all possible source Internet protocol (IP) addresses and uses a variety of protocols like transmission control protocol (TCP) or user datagram protocol (UDP), etc.

Briefly, Chen et al. (2007) developments may be classified in four technical features, as the quick vision and evidences of which are given in following segments:

1. *Traffic anomaly detection at the superflow level.* Monitoring Internet traffic at routers on individual flows is identified by a 5-tuple: source