International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016 [ICSSCCET 2016]

| | | | |
|---|---|---|---|
| **ISBN** | 978-81-929866-6-1 | **VOL** | 02 |
| **Website** | icssccet.org | **eMail** | icssccet@asdf.res.in |
| **Received** | 25 – February – 2016 | **Accepted** | 10 - March – 2016 |
| **Article ID** | ICSSCCET071 | **eAID** | ICSSCCET.2016.071 |


# IP Based Software Puzzles using AES Encryption on The Client Side to Countermeasure DOS Attacks

**S Ganapathi Ammal[1], S Vimala[2], K ArunKarthick[3], S JaganRaja[4], R Surya[5]**

[1]Assistant Professor, Department of Information Technology, Karpagam Institute of Technology, Coimbatore

[2]Assistant Professor, [3,4,5] UG Scholar, Department of Information Technology, Selvam College of Technology, Namakkal

**Abstract:** *Information forensics and security is the practice of defending information from unauthorized user. Denial-of-service (DoS) attacks make a machine (or) network resource unavailable to its intended users. Such as the occurrence of indefinitely interrupt (or) suspended service of a host by using built in Graphics processing unit (GPU).Software puzzle scheme is proposed for defeating GPU-inflated DoS attacks. But it has to spend lots of time for generating the puzzles on server. It is to be decreasing the server's efficiency and resources. In Proposed system to implementing a new method based on generating the puzzles on client side. This increases the server efficiency and also the user is evaluated with their IP address. The puzzles are changed randomly based on the user performance. Traffic on the network is also reduced and it allows more users to connect with the system.*

**Index Words:** *Software puzzle, Code Obfuscation, GPU Programming, Denial of Service (DoS), AES*

## 1. INTRODUCTION

A denial of service (DoS) attack is an incident in which a user or organization is deprived of the services of a resource they would normally expect to have. In a distributed denial-of-service, large numbers of compromised system attack a single target. [1]Although a DoS attackdoes not usually result in the theft of information or other security loss, it can cost the target person or company a great deal of time and money. Typically, the loss of service is the inability of a particular network service, such as e-mail, to be available or the temporary loss of all network connectivity and services. A denial of service attack can also destroy programming and files in affected computer systems. In some cases, DoS attacks have forced Web sites accessed by millions of people to temporarily cease operation.

RSA is a cryptosystem for public-key encryption, and is widely used for securing sensitive data, particularly when being sent over an insecure network such as the Internet. RSA [2] derives its security from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy, but determining the original prime numbers from the total factoring is considered infeasible due to the time it would take even using today's super computers. The public and the private key-generation algorithm is the most complex part of RSA cryptography. Two large prime numbers, p and q, are generated. A modulus n is calculated by multiplying p and q. This number is used by both the public and private keys and provides the link between them. Its length, usually expressed in bits, is called the key length. The edoesn't have to be a secretly selected prime number as the public key is shared with everyone. The private key consists of the modulus n and the private exponent d, which is calculated using the Extended Euclidean algorithm to find the multiplicative inverse with respect to the quotient of n. As discussed, the security of RSA relies on the computational difficulty of factoring large integers.

As computing power increases and more efficient factoring algorithms are discovered, the ability to factor larger and larger numbers



**Cite this article as:** S Ganapathi Ammal, S Vimala, K ArunKarthick, S JaganRaja, R Surya. "IP Based Software Puzzles using AES Encryption on The Client Side to Countermeasure DOS Attacks". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 373-379. Print.

also increases. Encryption strength is directly tied to key size, and doubling key length delivers an exponential increase in strength, although it does impair performance. RSA keys are typically 1024- or 2048-bits long, but experts believe that 1024-bit keys could be broken in the near future, which is why government and industry are moving to a minimum key length of 2048-bits.

The light has to be analysed that threat adversaries may have a way to achieve resource inflation, where they use resources or techniques that valid clients may not have implemented. If this inflation technique is good enough, then the arguments that disparity is acceptable begin to break down, as increasingly large numbers of valid clients are unable to muster the resources needed to obtain service. Of course, in many cases it is sufficient for the valid clients just to use the inflation technique themselves, thus levelling the playing field.

However, in other cases this may not be possible or desirable. For instance, for bandwidth-based schemes, a client may increase their bandwidth resources by either deliberately misusing the wireless MAC layer, or ignoring congestion control back-off rules at the transport layer. This enables them to gain extra bandwidth at the expense of other hosts.

Such modifications may require special administrative privileges or access to specialized software and can create DoS or congestion [14] in the network so valid clients are not likely to use them. In such cases, it is hoped that the inflation threat is not too severe or that other measures, like detection of such behaviour, can limit its effectiveness. In other cases, the issue may be more border line and it may be debatable whether valid clients should use the same inflation tactics as attackers.

In all three cases, however, it is prudent to have some way of analysing the threat to determine whether the valid nodes should match the inflation technique, ignore it because its impact is small, or change to different currency-based schemes. We consider a range of resource inflation strategies and attempt to access their likely effectiveness. These include: cloud computing (outsourcing of puzzles), multi-core parallel computations (enabled by many new processors), the use of Graphical Processing Units (GPUs), and the bandwidth inflation techniques mentioned above.

Among these we find that GPUs pose the greatest threat and require the most immediate attention. We show that attackers can use cheap and widely available GPUs to inflate their ability to solve typical cycle exhaustion puzzles by more than a 600x factor. While adaptive puzzle schemes may be able to tolerate a disparity of 40x, they fail when faced with a 600x disparity. In this range, adaptively making the resource demands higher does not help, since adversaries are so much richer than valid clients that the latter are 'priced out of the market' and fail to get service in a reasonable time.

On the other hand, to consider some of the drawbacks valid clients might experience if they use their GPUs to solve DoS puzzles. The paper makes three primary contributions. First, we introduce and analyse the concept of resource inflation as a 'thinking out of the box' approach to defeating DoS countermeasures. This includes a simple quantitative model to assess the impact of resource inflation. We argue that currency-based schemes should always be analysed with this model to estimate their resilience against resource inflation. Second, we illustrate a series of resource inflation attacks on existing DoS defence mechanisms to underscore the significance and implications of this threat. Third, while the general idea of using GPUs to solve puzzles may seem obvious to some, how to implement and measure a full range of strategies is not obvious and comprised the bulk of the work has to be done.

## 2. Resource Inflation Using GPUs

Graphics Processing Units (GPUs) [3] are dedicated devices used for rendering, manipulating, and displaying computer graphics. Modern GPUs are in general very efficient at processing large amounts of data in parallel. Unlike the modern CPU which is designed to efficiently optimize the execution of single threaded (or not highly multi-threaded) programs using complex out-of-order execution strategies, a modern GPU's efficiency comes from executing massively data-parallel programs. These are algorithms which perform simple operations on a large number of data points in parallel.

This is often referred to as Single Instruction Multiple Data (SIMD) [10] programming. Recently, there has been significant interest in using GPUs' efficiency at executing data parallel algorithms for non-graphical computation. This paradigm is known as General Purpose GPU (GPGPU) computes or Stream Computing in the hands of non-graphics programmers. There exist multiple tool-chains that support GPGPU programming [10] [12].

Two of the most popular ones are the Brook/Compute Abstraction Layer (CAL) from AMD and Compute Unified Device Architecture (CUDA) from Nvidia. GPGPUs have been used in improving the speed of various programs such as Folding@Homecomputational chemistry as well as various other fields. Recently, GPUs were also used for finding MD5 chosen-prefix collisions. Logical view of GPGPUs Architecture mentioned in Fig 2.1. Each engine in turn consists of a number of thread-processors.

For instance, AMD's HD4850 contains 800 thread-processor instances. Each thread processor unit has access to its own general purpose registers and can also access the GPU's memory. The thread dispatcher manages various threads and is invoked by the client

**Cite this article as:** S Ganapathi Ammal, S Vimala, K ArunKarthick, S JaganRaja, R Surya. "IP Based Software Puzzles using AES Encryption on The Client Side to Countermeasure DOS Attacks". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 373-379. Print.

on the CPU. It must be noted that threads in GPUs are not as analogous to processor threads on a general purpose computer. A GPU-based thread is a very lightweight thread that can be started with minimum overhead. All the GPU-based threads (within a single SIMD engine block) need to execute the same instruction (possibly on different input data) for maximum efficiency. This granularity uses a parallelism exploited between plaintext blocks only.

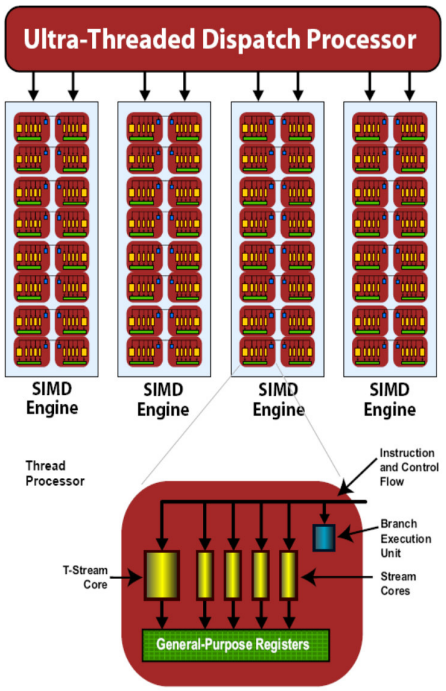


Fig2.1: Logical view of GPU Architecture

All of such threads can run simultaneously on the stream processors mentioned in Fig 2.1 as long as there are enough stream processors on the GPU card. In case the data elements are larger than the number of stream processors, the Thread Dispatcher manages the available processors between the various threads. For instance, consider the following snippet of code written in AMD's stream computing language [13]:

```
kernel void

sum(float a<>, float b<>, out float c<>)

{

        c = a + b;

}
```

When two threads running on a thread processor in an SIMD engine need to execute different instructions, the GPU will ensure that only one of the threads executes at any given time.

## 3. Basic GPU-Inflated DoS Attack

In order to elaborate software puzzle, we recap its rival GPU-inflated DoS attack [11] [12] in advance. When a client wants to obtain a service, she sends a request to the server. After receiving the client request, the server responds with a puzzle challenge x. If the client is genuine, she will find the puzzle solution y directly on the host CPU, and send the response (x, y) to the server. However, by using the similar mechanism in accelerating calculation with GPU, a malicious user who controls the host will send the challenge x to GPU and exploit the GPU resource to accelerate the puzzle-solving process.

**Cite this article as:** S Ganapathi Ammal, S Vimala, K ArunKarthick, S JaganRaja, R Surya. "IP Based Software Puzzles using AES Encryption on The Client Side to Countermeasure DOS Attacks". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 373-379. Print.

## 4. Other Resource Inflated Threads

To study other avenues for resource inflation [5] on puzzle-based mechanisms and bandwidth based mechanisms. On puzzle-based mechanisms, the use of multi-core processors and cloud computing for resource inflation. In effect, the time taken to solve the puzzle is reduced from't' ona single threaded implementation to (t=n) + c on an n-core processor, where c is a small synchronization overhead. In an analysis in a factor of 38 x speed-up is provided between resources constrained Nokia 6620and a PC with a Xeon 3.2GHz processor. It appears that the puzzle solving algorithm on the Xeon processor was not parallelized. The benefits of running multiple threads for solving has hreversal puzzles by implementing it on an Intel Q6600 processor.

## 5. AES

Advanced Encryption Standard, a symmetric block cipher introduced in 2001 by NIST[6], encrypts and decrypts plaintext and cipher text blocks using a 128-bit, 192-bit, or 256-bit key size. Its calculation unit is 1 byte. This cipher executes the iteration of the same round, for which the number of iterations depends on the key size. We selected 128-bit key length AES, which consists of 10 rounds. Each round consists of four transformations: Sub Bytes, Shift Rows, Mix Columns, and Add Round Key. The final round differs slightly from the other rounds: it does not include Mix Columns. As described in this paper, AES is implemented based on optimized ANSI C source code provided as a part of Open SSL, the open source toolkit for SSL/TLS. Its algorithm defines round processes combined into a transformation simply using a lookup table called "T-box" and exclusive-or operation. Letting a be the round input, which is divided into each 32 bits, the round output e is represented as

$$ej = T0[a0;j] \oplus T1[a1;j+1] \oplus T2[a2;j+2] \oplus T3[a3;j+3] \oplus kj;$$

Where T0, T1, T2, and T3 are lookup tables and kjis the j-th column of a round key. This algorithm includes only four lookup table transformations and four exclusive-or operations. Furthermore, AES hassome modes such as Electric Code Book (ECB) and Cipher Block Chaining (CBC).

## 6. AES encryption implementation on CUDA GPU

This section presents discussion of the granularity of parallel processing and memory allocation to design parallelized AES on a CUDA GPU. Granularity signifies a task size to dispatch to a processor. Granularity is an effect of the parallel AES algorithm design which represents how to parallelize the AES algorithm. The memory allocation strategy of CUDA is important because CUDA has some different types of memory systems, as described earlier. Characteristics of each memory system mutually differ to quite a degree.

### 6.1 Granularity of parallel processing

Granularities of four types are defined to parallel AES algorithm [8].

#### 6.1.1 6 Bytes/Thread

Using the parallelizing method of 16 bytes/thread means that each thread processes each plain text block consisting of 16 bytes independently. This implementation presents advantages of lower over head than other granularities requiring no synchronization and no shared data between threads. This granularity uses a parallelism exploited between plaintext blocks only.

#### 6.1.2 8 Bytes/Thread and 4 Bytes/Thread

Granularity at 8 bytes/thread processes one plaintext block with two threads. It presents a plain text block divided into two threads and two plaintext blocks processed by four processors simultaneously. This method exploits parallelism between plaintext blocks and inner plaintext processing.

#### 6.1.3 1byte/Thread

It is absolutely better to process AES encoding with at least a 32-bit operating unit because the AES encoding algorithm used in these studies is optimized for 32–bit processing. However, it is able to process 1 byte data by a thread. 1 byte/thread means that 16 threads process a plaintext block in a coordinated manner.

**Cite this article as:** S Ganapathi Ammal, S Vimala, K ArunKarthick, S JaganRaja, R Surya. "IP Based Software Puzzles using AES Encryption on The Client Side to Countermeasure DOS Attacks". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 373-379. Print.

## 7. Overview of Resource Inflation Threats to DoS Countermeasures

To perform an extensive experimental case study on resource inflation attempts on puzzle-based mechanisms. First, we investigate the feasibility of using GPUs for improving puzzle-solving speeds as GPUs are particularly efficient in handling large numbers of similar operations in parallel. Second, we study the feasibility of speeding up puzzle solving by leveraging multiple processors in a multi-core processor [10]. Third, we consider the use of cloud computing facilities for parallelizing, and hence speeding up puzzle solving capabilities. We also explore resource inflation possibilities against bandwidth-based mechanisms. In particular, we focus on three classes of resource inflation possibilities. First, we study the possibility and implications of attacks at transport layer, and in particular those that ignore or exploit congestion control mechanisms. Second, we consider the effect of greedy MAC-layer behaviour by an attacker that tries to gain a larger share of the bandwidth by violating the protocol in subtle ways. Third, we briefly discuss how using more than one interface for network connection can help an attacker inflate its resources.

## 8. Framework of Software Puzzle

In order to defeat the GPU-inflated DoS attack [1], we extend data puzzle to software puzzle. At the server, the software puzzle scheme has a code block warehouse W storing various software instruction blocks. Besides, it includes two modules: generating the puzzle C0x by randomly assembling code blocks extracted from the warehouse; and obfuscating the puzzle C0x for high security puzzle C1x.The code block warehouse W [9] stores compiled instruction blocks {bi}, e.g., in Java bytecode, or C binary code. The purpose to store compiled codes rather than source codes is to save server's time; otherwise, the server has to take extra time to compile source codes into compiled codes in the process of software puzzle generation.

## 9. Software Puzzle Generation

In order to construct a software puzzle [7], the server has to execute three modules: puzzle core generation, puzzle challenge generation, software puzzle encrypting/obfuscating. Once a software puzzle C1x is created at the server side and compiled into the Java class file C1x.class, it will be delivered to the client who requests for services over an insecure channel such as Internet, and run at the client's side. Fig 9.1 shows the puzzle generation architecture to describe how the puzzle was generated.

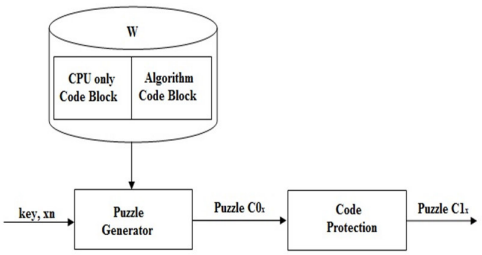


Fig 9.1: Puzzle Generation Architecture

## 10. Security Analysis

Software puzzle aims to prevent GPU from being used in the puzzle-solving process based on different instruction sets and real-time environments between GPU and CPU. Conversely, an adversary may attempt to deface the software puzzle scheme by simulating the host on GPU, cracking puzzle algorithm, re-producing GPU-version puzzle, or abusing the access priority in puzzle-solving. If an attacker is able to run a CPU simulator over GPU environment, the software puzzle can be executed on GPU directly. However, this simulator-based attack may be impractical in accelerating the puzzle-solving process.

## 11. Experimental Evaluation

SSL/TLS protocol is the most popular on-line transaction protocol, and an SSL/TLS server performs an expensive RSA decryption operation for each client connection request, thus it is vulnerable to DoS attack. Our objective is to protect SSL/TLS server with software puzzle against computational DoS attacks, particularly GPU-inflated DoS attack. As a complete SSL/TLS protocol includes many rounds, we use RSA decryption step to evaluate the defence effectiveness in terms of the server's time cost for simplicity. Assume the time to perform one RSA decryption bet0, and the time to generate and verify one software puzzle be ts(Note that t0 >ts, otherwise, software puzzle is useless). The genuine clients spend less time in waiting for the services. Hence, a good strategy is to initiate the software puzzle defence if the number of requests is beyond a threshold; otherwise, no defence is required because service

**Cite this article as:** S Ganapathi Ammal, S Vimala, K ArunKarthick, S JaganRaja, R Surya. "IP Based Software Puzzles using AES Encryption on The Client Side to Countermeasure DOS Attacks". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 373-379. Print.

is satisfactory for all clients. Fig 11.1 shows that software puzzle can increase the service quality significantly in terms of the percentage of served customers.

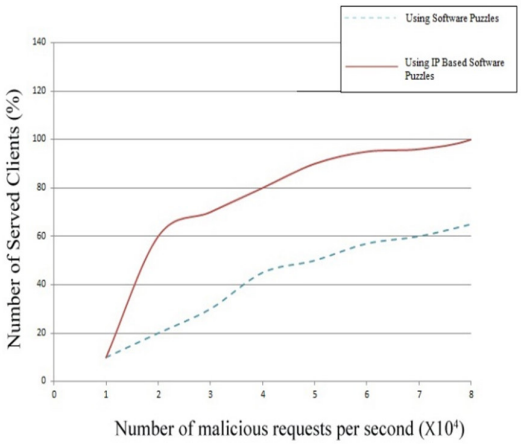


Fig 11. 1 Service Efficiency comparison of server with Software puzzles and IP Based Software Puzzles

The graph also shows the difference between performances of server efficiency by using both the Software Puzzles and also the IP based software puzzles. Software puzzle scheme performs 15,000 requests per second. In IP based Software puzzles the server is expected to perform nearly 50,000 requests per second. It performs

To demonstrate the effectiveness of IP Based Software puzzle, let's compare the cost of the participants.

## 11.1 Server Cost

If the server-client system adopts software puzzle, the CPU time spent in the server is

- time t1 for preparing the initial puzzle C0x;
- time t2 for converting C0x into software puzzle C1x;
- time t3 for puzzle package generation;
- timet4 for verifying the client answer.

In the countermeasure, the server has to send the software puzzle package to the client. The package is merely 120; 000 bits on average, hence, the server is able to serve 12*109=120000 = 105 users assuming the network bandwidth is 12Gbps. Indeed, the service capacity can be increased if the puzzle core is constructed from random and lightweight function. Thus, the bandwidth DoS attack threat is small. In other words, the present scheme can increase the defence capability against time-DoS attack, without sacrificing the defence capability against space-DoS attack.

In order to verify the response (˜x; ˜y), the server has to store the corresponding (x; y) into the storage S, which is about 128+16=144 bits, or 18 bytes. In order to remove a long-time open request so as to prevent memory exhaustion, each result is kept for some time only, e.g., 1 minute. Thus, given that there are 15,000 requests per second, the storage for the server is merely 18*15000*60 = 1.62*107 bytes, or about 16M bytes, which is very small for a server.

## 11.2 Client Cost

In order to be served by the server, a client has to solve the software puzzle by trial and error. For each trial, the client has to run the software puzzle. In the experiments, the client takes 2 seconds to try only 2,000 keys for finding the solution y because a newly loaded class has to run the load Class(), get Method() and invoke() which are very slow in the present JVM. To enable bigger search space, the new class is reconstructed with a batch of trial solutions so as to amortize the re-loading time, e.g., when the new class includes puzzle code for 36 trials, the client is able to test 11,918 keys within 2 seconds, while the communication cost is merely increased 30% with jar package.

**Cite this article as:** S Ganapathi Ammal, S Vimala, K ArunKarthick, S JaganRaja, R Surya. "IP Based Software Puzzles using AES Encryption on The Client Side to Countermeasure DOS Attacks". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 373-379. Print.

### 11.3 Attacker Cost

The attacker has two choices to solve the software puzzle. One is to solve the puzzle as a normal client does. Obviously, the attacker has no advantage over the normal client in this case. In other words, the IP based software puzzle achieves its goal.

The second choice is that the attacker's host simulates the software puzzle and converts the software puzzle into the GPU version. In this case, GPU can quickly solve the puzzle in parallel, but the conversion process takes almost the same time as the first choice. This gives the attacker no incentive to perform the conversion.

## 12. Conclusion and Future Work

An open problem is how to construct the client-side software puzzle so as to save the server time for better defence performance. By initialising the puzzles on the client side saves the server time to use for other resources. The user is evaluated with the IP address, whether the user is returning user and authenticated user. If the user is an authenticated user, usual puzzles are generated. When an unauthorised user tries to attack the server simultaneously then the puzzles are changed randomly that prevents the resource from the attackers. To evaluate the effect of code de-obfuscation, this is related to the technology advance of code obfuscation. It reduces the traffic in the network and allows more authenticated users to use the server resources. It provides a secure transmission of data. In the present IP based software puzzle, the server doesn't have to spend time in constructing the puzzle. Although this paper focuses on GPU-inflation attack, its idea can be extended to thwart DoS attackers which exploit other inflation resources such as Cloud Computing. For example, suppose the server inserts some anti-debugging codes for detecting Cloud platform into software puzzle, when the puzzle is running, the software puzzle will reject to carry on the puzzle-solving processing on Cloud environment such that the Cloud-inflated DoS attack fails.

References


1. Yongdong Wu, Zhigang Zhao, FengBao and Robert H. Deng "Software Puzzle: A Countermeasure to Resource-Inflated Denial-of-Service Attack" Published in: Information Forensics and Security, IEEE Transactions on (Volume: 10, Issue:1),Mar 2015.
2. B. Barak, O. Goldreichy, R. Impagliazzoz, S. Rudich, A. Sahai, S. Vadhank, and K. Yang, "On the (Im)possibility of Obfuscating Programs," CRYPTO, LNCS 2139, pp.1-18, 2001
3. Christos Douligeris, and Aikaterini Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," Computer Networks 44, pp.643-666, 2004.
4. David Kahn, "The Code breakers: the story of secret writing," Scribners, second edition, pp.235, 1996.
5. David Keppel, Susan J. Eggers and Robert R. Henry. "A Case for Runtime Code Generation". Technical Report 09-02-1996
6. Z. Gao and N. Ansari. "Differentiating malicious ddos attack traffic from normal tcp flows by proactive tests." Communications Letters, IEEE, 10(11):793–795, November 2006
7. James E. Smith, and Ravi Nair, "Virtual machines: versatile platforms for systems and processes," Morgan Kaufmann Publishers, pp.19, 2005.
8. John Black, and Phillip Rogaway, "Ciphers with Arbitrary Finite Domains," CT-RSA, LNCS 2271, pp.114-130, 2002.
9. A. Juels, and J. Brainard, "Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks," Network and Distributed System Security Symposium, pp. 151-165, 1999.
10. Keisuke Iwai, Hashirimizu Yokosuka Kanagawa, Nakameguro Meguroku, Takakazu Kurokawa. "Acceleration of AES encryption on CUDA GPU". International Journal of Networking and Computing – www.ijnc.org ISSN 2185-2839 (print) ISSN 2185-2847 (online) Volume 2, Number 1, pages 131–145, January 2012
11. S. Khanna, S. S. Venkatesh, O. Fatemieh, F. Khan, and C. A. Gunter. Adaptive selective verification In INFOCOM '08: IEEE Conference on Computer Communications, Phoenix, AZ, April 2008. IEEE.
12. Ravinder Shankesi, Omid Fatemieh, and Carl A. Gunter "Resource Inflation Threats to Denial of Service Countermeasures" University Of Illinois
13. Ronald L Rivest, Adi Shamir and David A_ Wagner "Time locks puzzles and timed-release Crypto" Revised March 2008.
14. Timothy J. McNevin, Jung-Min Park, and R. Marchany, "pTCP: A Client Puzzle Protocol for Defending against Resource Exhaustion Denial of Service Attacks," TR-ECE-04-10, Virginia Tech, Oct. 2004.
15. Xiao Feng Wang, and Michael K. Reiter, "Mitigating Bandwidth- Exhaustion Attacks using Congestion Puzzles," ACM Conference on Computer and Communications Security, pp.257-267, 2004.


**Cite this article as:** S Ganapathi Ammal, S Vimala, K ArunKarthick, S JaganRaja, R Surya. "IP Based Software Puzzles using AES Encryption on The Client Side to Countermeasure DOS Attacks". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 373-379. Print.