for example. Web security mechanisms will also scan content, watching for content such as pornography, and for malicious code contained on a web page that might compromise a user’s computer.

“Condensing multi-layered protection into a single device, updated by the vendor, provides the best protection for resource-constrained companies”

Covering all your bases
It is easy to see how these functions work in unison with each other. For example, attackers often use email to send malicious URLs to users. These may be spotted by email protection functions within a unified threat management system or Internet security appliance. However, if they slip through, they will be caught by the web filtering mechanism, making it doubly hard for attackers to compromise users. Anti-virus mechanisms built into the device will also scan for malware separately, providing yet another level of protection.

Defence in depth is a crucial technique for any modern SMB that wants to protect itself against intrusion. Condensing multi-layered protection into a single device, updated by the vendor, provides the best protection for resource-constrained companies.

Modern Internet security is an exercise in probability. It is impossible to guarantee 100% security – a determined hacker may still be able to gain access to a company’s system. But the more points protection that a company covers, the more likely it is to fend off the majority of generic attacks on the Internet. Can you afford not to cover your bases?

About the author
James Harris is a product manager at network equipment manufacturer ZyXEL Communications UK. Working in the industry for the past 10 years he has held various technical and product management positions specialising in wireless, Ethernet and routing technologies.

Mitigating denial of service attacks in hierarchical wireless sensor networks
Rohan Nanda and P Venkata Krishna, School of Computing Science and Engineering, Vellore Institute of Technology, India

Due to the considerable research and development invested in new networking protocols, Wireless Sensor Networks (WSNs) have proved to be an important emerging field. WSNs are widely used in homeland security and military applications, in hospitals for medical monitoring, and in industry. However, their limited battery and power options, processing capability and memory make WSNs vulnerable to a variety of network attacks.

The use of the wireless medium is the major weakness, allowing any adversary to attack the network or compromise the nodes. The security of WSNs is an important factor because confidential data communicated between the sensor nodes needs to be protected from untrusted third parties who can misuse or modify this data by malicious means.

The aim here is to develop a key management scheme to protect the server in a hierarchical sensor network from a Denial of Service (DoS) attack. A DoS attack would make a server in a hierarchical WSN unavailable to its intended nodes or clients. The server gets saturated with external requests so that it cannot respond to legitimate requests from other nodes. Even if the server responds, the response is so slow that it is rendered effectively unavailable. DoS attacks attempt to consume the resources of the target server so that they are depleted to the point where a reset is forced. The server has the capability to handle only a fixed number of requests at a time. When the number of requests exceeds this maximum number then the time taken by server to respond to each request increases significantly. As the frequency of attack increases, the server becomes completely unavailable to service any kind of request. Thus the requests of legitimate users are denied.

The key management scheme for hierarchical WSNs, outlined here, also uses a hierarchical structure from server (base station) to cluster heads and finally to sensor nodes. Organisation across the network uses cluster formations and the establishment of a derivative key between all three hierarchical layers. Then keys are established for secure communication between sensors belonging to the same cluster head. Finally, we’ll define key management processes used to identify a DoS attack and defend the server.
Related work

There are many publications and proposals that suggest different ways of handling DoS attacks using a key management protocol. Public key cryptographic techniques prevent DoS attacks that exploit draining the battery through WSN ephemeral key establishment.\(^1\) They combine a DoS mitigation scheme with self-certified, ECC-based key generation to yield a resource-efficient security framework. The cluster adaptive rate-limiting scheme for preventing denial of sleep attack introduces cluster adaptive rate-limiting, which is based on current host-based intrusion detection techniques.\(^2\) This technique might be used to reduce energy consumption to an arbitrarily low level until a denial of sleep attack is lifted. Another scheme provides a framework to discover the number of malicious packets among a massive flood of packets by using a probabilistic approach.\(^3\) This approach achieves a better overhead than the HCF computation method.

"This method forces multiple hashing calculations and numerical factors in order to falsify the first intercepted data packet and win time in broadcasting trusted message packets"

Another scheme proposes a multi-user DoS containment and signature-based broadcast authentication scheme.\(^4\) The authors also discuss Routing and Remote Access Service (RRAS), a lightweight scheme to effectively contain a DoS attack. In one scheme, the authors propose a new broadcast key management scheme for distributed WSN, which prevents a potential DoS attack where a packet has been intercepted.\(^5\) This method forces multiple hashing calculations and numerical factors in order to falsify the first intercepted data packet and win time in broadcasting trusted message packets to cover the whole network. Another new scheme discusses a remote access framework incorporating a virtual home and a Distributed Denial of Service (DDoS) defence server, which increases the difficulty of launching a low-level DDoS attack against a WSN.\(^6\) The authors propose a DoS defence approach consisting of three entities: Virtual Home; Remote Home Server; and DDoS Defence Server. Out of all these schemes, very few have used key management techniques to mitigate DoS attacks. However, key management is an integral part of a WSN even if it is not originally used for the purpose of mitigating a DoS attack.

The proposed model

The main aim of the work detailed here is to develop a new key management scheme using the concept of timestamp and delay that combats a DoS attack on WSNs. We consider the same structure of a hierarchical WSN as discussed in a previous paper on a self-enforcing and secure protocol.\(^7\) The server operates in place of the base station as the topmost root node of the hierarchy. The second hierarchy level contains the cluster heads that are responsible for co-ordination and management of nodes present under that particular cluster head. Each cluster head groups some sensor nodes under it, and these are used for various purposes depending on the application in use. The complete hierarchy is illustrated by Figure 1 where a server has three cluster heads and each cluster head controls two sensor nodes. The cluster heads can communicate with each other and with also with the server. In addition, they can communicate data to other cluster heads and the server. Similarly, the sensor nodes can communicate with other sensors of the same cluster and with the cluster head.

In our scheme each node, including the server (base station), has a unique ID and an inbuilt key (K). Before the deployment phase, each sensor node has been provided with a pseudo-random function (f) so that the sensor nodes are resilient towards outsider attacks.\(^8\) Therefore each sensor node computes its own new key by using the previous key (K\(_s\)) and its unique ID.

\[
K = f ( K_s, ID )
\]

Now all the nodes, including the server, cluster heads and sensors, have computed their keys. This key is called the original key.

Network organisation using derivative key

The base station or server identifies the cluster heads deployed under it. To choose each cluster head, the server makes use of the original key with the ID of a particular cluster head to establish communication. This is done by the derivative key calculation K’ as follows:

\[
K' = H ( K \parallel ID_i )
\]

Here, K is the original key, ID\(_i\) is the ID of the cluster head and H is the hash function which is computationally unfeasible to revert. All the cluster heads can be chosen using the derivative key.
Subsequently, the server organises the network by sending a list of sensor nodes deployed to each cluster head in a message encrypted by derivative key $K'$ as follows, as described by Hu Bai and Yang:

$$E_{K'}(M \parallel ID_{i}^{list})$$

Here, $ID_{i}^{list}$ contains the IDs of all the sensors that are deployed under that particular cluster head and $M$ is the message for sensor nodes. After receiving this message the cluster heads decrypt it using the same derivative key $K'$ to extract the information of the list of sensors. Now the cluster heads establish the same derivative key with the sensor nodes by assigning each of them a particular ID.

$$K' = H(K \parallel ID_{ij})$$

Here $ID_{ij}$ is the ID of a particular sensor node. After this step, each cluster node includes all the sensor nodes under it. Now the sensor nodes can even transmit data or messages directly to the server by using key $K'$. The sensor nodes can send a message to the server informing it of its ID. The message is encrypted using the derivative key $K'$.

$$E_{K'}(M \parallel ID_{i})$$

The encrypted message is received by the server and it decrypts it using $K'$ to retrieve the necessary information. Now, finally, the network is organised and all the nodes are using the same key $K'$ for communication.

**Cluster key usage**

After the hierarchical WSN is organised, there needs to be a secure communication between the sensor nodes and cluster heads and also between sensor nodes belonging to the same cluster head. Suppose the sensor node $S_{ij}$ wants to transmit data with $S_{i(j+1)}$ – again, $S_{i}$ encrypts the message with key $K'$ and sends it to $S_{i(j+1)}$, as in Hu et al, but with the technique described here the message uses a different key and method.

$$E_{K'}(M, ID_{ij})$$

$S_{i(j+1)}$ decrypts the message using the same key $K'$ to obtain the necessary information present in message $M$.

**Authentication check at each hierarchical level**

To secure both the server and the whole network, it is essential to check the authenticity at each hierarchical level. If there is a malicious node present among the sensors, and the server or cluster head establishes a key with it, then the adversary can easily launch attacks on the network, and it would be very difficult to combat that attack. Therefore we propose a sensor node authentication model for the given scheme.

When a cluster head has to authenticate the sensors under its hierarchy it then it computes a value called $B_{pq}$ where $p$ is the cluster head and $q$ is the sensor node to be authenticated.

$$B_{pq} = H(ID_{p} \parallel T')$$

Here, $T'$ is the current timestamp of sensor $p$ and $ID_{p}$ is the unique ID of sensor $p$. The cluster head $p$ sends a message $AUT=<B_{pq}, T'>$ to sensor node $q$. After receiving the message $AUT$, the sensor node $q$ validates the timestamp if $(T'-T) \leq \Delta T$ where $T'$ is the current timestamp of sensor node $q$ and $\Delta T$ is the expected time interval of the transmission delay. If this condition is not true then authentication fails and cluster head $p$ gets to know that $q$ is a malicious sensor node and terminates the communication with sensor $q$. If the condition is true then sensor node $q$ computes:

$$B_{qp} = H(ID_{p} \parallel T)$$

Here, $T$ is the current timestamp of sensor $q$. Now sensor $q$ compares the value of $B_{qp}$ with the received value of $B_{pq}$. If $B_{pq} = B_{qp}$ holds true then cluster head $p$ knows that sensor $q$ has been authenticated successfully and now it is safe to establish a key with sensor $q$. In the same manner, the server also authenticates the cluster heads before the key is shared.

**Mitigating a DoS attack**

DoS is a highly dangerous attack that cripples the server by flooding the network with excess service requests to the server. A server can only service the request of a client if it can allocate resources to that client. A DoS attack depletes the resources of the server so that it cannot allocate the necessary resources. The server can handle a fixed amount of service requests over a given period. If the number of requests exceeds this value then the server is under DoS attack and is not able to process the requests of legitimate users or clients.

Suppose the server’s capacity to process requests has been exceeded or is on the verge of crossing the maximum number of requests per unit of time. At this stage, the server encrypts a message using the key $K'$ which contains the IP address of the server, and it broadcasts this message over the entire network but only to the cluster heads.

$$E_{K'}(IPADDR \parallel ID_{i})$$

Here, IPADDR refers to the IP address of the server and $ID_{i}$ is the unique ID of each cluster node. The cluster node decrypts the message to obtain the IP address of the server. The cluster head now needs to identify the attacker who induced the DoS attack on the network. All the requests to the server from the sensor nodes are passed through the cluster heads. Now the cluster heads again encrypt a message with the key $K'$ with a timestamp value and broadcast it to all their sensor nodes with a message:

$$E_{K'}(IPADDR \parallel M \parallel T)$$

Here the message $M$ contains the instruction for execution by the sensor nodes upon receiving the message. After receiving the message, the sensor nodes decrypt the message using the key $K'$ to obtain the contents of the message. The message $M$ contains an urgent delay function that commands the sensor nodes to delay their request to the received IP address by the value of the timestamp specified in the encrypted message. Upon receiving this message, all the sensor nodes that are currently requesting the service of the server will delay their service requests by the time specified in the timestamp. The value of the timestamp is computed by the cluster heads on the basis of the acuteness of the DoS attack. So all the sensor nodes that are not requesting service from the server will continue to process their service requests. Since each request will now be delayed by a timestamp $T$, the server recovers its depleted resources and the effect of the attack is nullified. However, if after receiving the message a node is still not slowing down the number of requests to the server then it...
means that this node has initiated the DoS attack. Now the corresponding cluster head traces the IP address of this node, encrypts it and broadcasts it over the network with the message BLOCK, which indicates to every other node to discard the packets coming from this IP address, IPATTACK.

\[ E_K = (IPATTACK \parallel BLOCK) \]

Suppose the attacker node also delays service requests when the delay timestamp packet is issued, and that after that time period is over, it again starts fetching a number of requests. In this case, the server should issue this delay timestamp packet periodically so that the attacker is compelled to reveal its identity. In another scenario, the server can block the requests from a particular cluster head by simply discarding the requests from that cluster so as to identify the cluster in which the attacker is actually present. After identifying the cluster head, the legitimate users from other clusters won’t be denied service requests, while the cluster head containing the attacker can use the delay timestamp packet to identify the attacker. Hence this scheme mitigates the DoS attack in all cases and makes the server available for legitimate users.

### Analysis of the proposed model

For evaluating the performance and efficiency of the scheme, there are some important factors, such as security and overhead.

**Overhead:** The computational overhead is mainly imposed by the derivative key calculation, authentication check at each hierarchical level and the emergency message broadcast when the DoS attack occurs. Due to the high efficiency of the pseudo-random function, the computational overhead before deployment is negligible. The time complexity of the hashing functions is estimated to be \(O(1)\), due to which the overhead of the entire scheme is highly reduced. In total, we have used four hash function computations, two of which are used in derivative key calculation and the other two in authentication checks at each hierarchical level. Although authentication at each hierarchical level incurs some computational overhead, because of the verification of the nodes, it is an indispensable part of the scheme that enhances the overall security. In case of a DoS attack, the main aim is to keep the server working so that its resources are not compromised, so it is essential to broadcast the delay timestamp packet, which also incurs a computational time overhead.

**Security:** From the start of the scheme, where the pseudo-random function keeps on generating fresh keys until the authentication check, the scheme provides resilience towards network security attacks and maintains data integrity and privacy within the network. The derivative key, established after the random function, is symmetrically organised throughout the network, which makes it very difficult for an adversary to find a weak point in the network. The authentication check at each hierarchical level further shields the scheme by verifying the identity of the sensor nodes with which the key is going to be established. The steps taken during a DoS attack, such as delaying the service requests of clients and discarding the packets of the attacker, play an integral role in successfully defending the server against the adversary. Due to authentication checks and the derivative key, node compromise also doesn’t yield any secure information about the keys of other nodes. Even in the case when the attacker is not identified, application of the alternative strategy defeats the DoS attack on the server.

### Conclusion

We have proposed a key management scheme for hierarchical wireless sensor networks that not only defends the network against DoS attacks but also maintains confidentiality, integrity and authenticity of data transmitted between sensor nodes. It uses a derivative key that is derived symmetrically between the hierarchy of sensor nodes. This key enables the successful data communication between sensors, cluster heads and server. It also authenticates the sensor nodes with cluster heads and cluster heads with the server by using the timestamp method. Moreover, the timestamp delay packet plays a key role in mitigating a DoS attack. This scheme also provides a lot of scope for future development.

### About the authors

Rohan Nanda is a B.Tech CSE student at the Vellore Institute of Technology (VIT), Tamil Nadu, India. He is carrying out his research under Dr P Venkata Krishna. His research interests include wireless sensor networks security, key management and DoS attacks. He has published two papers on sensor networks security in IEEE International conferences.

Dr P Venkata Krishna is professor and division leader of computer networks at VIT. He has authored more than 80 research papers in various national and international journals and conferences. He is a reviewer of journals such as IEEE Trans on Mobile Computing, IET Journal on Communications and the International Journal of Security and Networks. He is the editor for the International Journal of Systems, Cybernetics and Informatics.

### References

1. Arazi, O; Qi, H; Rose, D. ‘A public key cryptography method for denial

Cloud computing: new challenges and opportunities

Richard Morrell and Akash Chandrashekar, Red Hat

We are witnessing a shift in the cloud computing and virtualisation landscapes as a new model of security arises in response to the demand for clarity into how to harness the consumption of elastic computing resources. If we look back at traditional server-based and hosting provision for security, it was very much belts and strong vendor-supported braces that allowed customers to have a perimeter-based security that enclosed their assets and provided assurance. How is this changing as a result of the move to a cloud-based model? Is the emphasis on assurance and privacy, especially in a multi-tenant environment in an outsourced location, changing the way we are approaching security?

Cultural change

Many organisations have already embraced virtual hosting and have used datacentres under contract for a decade or more to provide simple web-based hosting or clustered web application hosting. Does moving these into the cloud enforce a change in culture? What are the pressures to regulate the industry in the post-WikiLeaks and LulzSec hacking scandal world?

One of the most interesting aspects of the movement of Service Oriented Architecture (SOA) application development and hosting from corporate and on-premise firewalls to outsourced cloud providers is its effect on the way the audit model works and the way it can be assessed as part of an overall corporate governance model. The use of SAS 70 (II) certification as a standard across cloud deployments is a start but not sufficient on its own to provide assurance of a cloud provider’s ability to provide security controls that reflect customers’ needs. SAS 70 (II) is not and cannot be taken as applicable to the needs of enterprise customers using cloud computing as it is more of an accounting standard than a standard related to virtualised elastic computing environments. The reality is that customers would benefit from a more transparent and practical regulatory framework, one that takes into account governance, risk and control.