**ABSTRACT**

Ad-hoc low-power wireless networks are an exciting research direction in sensing and pervasive computing. Prior security work in this area has focused primarily on denial of communication at the routing or medium access control levels. This project explores resource depletion attacks at the routing protocol layer, which permanently disable networks by quickly draining nodes’ battery power. These “Vampire” attacks are not specific to any specific protocol, but rather rely on the properties of many popular classes of routing protocols. We find that all examined protocols are susceptible to Vampire attacks, which are devastating, difficult to detect, and are easy to carry out using as few as one malicious insider sending only protocol compliant messages.

Existing systems work on secure routing attempts to ensure that adversaries cannot cause path discovery to return an invalid network path, but Vampires do not disrupt or alter discovered paths, instead using existing valid network paths and protocol compliant messages. Protocols that maximize power efficiency are also inappropriate, since they rely on cooperative node behavior and cannot optimize out malicious action.

In this project we show simulation results quantifying the performance of several representative protocols in the presence of a single Vampire. Then, we modify an existing sensor network routing protocol to provably bind the damage from Vampire attacks during packet forwarding.

**1. INTRODUCTION**

Wireless sensor networks (WSNs) promise exciting new applications in the near future, such as ubiquitous on-demand computing power, continuous connectivity, and instantly-deployable communication for military and first responders. Such networks already monitor environmental conditions, factory performance, and troop deployment, to name a few applications.

As WSNs become more and more crucial to the everyday functioning of people and organizations, availability faults become less tolerable — lack of availability can make the difference between business as usual and lost productivity, power outages, environmental disasters, and even lost lives; thus high availability of these networks is a critical property, and should hold even under malicious conditions. Due to their ad-hoc organization, wireless ad-hoc networks are particularly vulnerable to denial of service (DoS) attacks.

While these schemes can prevent attacks on the short-term availability of a network, they do not address attacks that affect long-term availability — the most permanent denial of service attack is to entirely deplete nodes’ batteries. This is an instance of a resource depletion attack, with battery power as the resource of interest.

Vampire attacks are not protocol-specific, in that they do not rely on design properties or implementation faults of particular routing protocols, but rather exploit general properties of protocol classes such as link-state, distance-vector, source routing, and geographic and beacon routing. Neither do these attacks rely on flooding the network with large amounts of data, but rather try to transmit as little data as possible to achieve the largest energy drain, preventing a rate limiting solution. Since Vampires use protocol-compliant messages, these attacks are very difficult to detect and prevent.

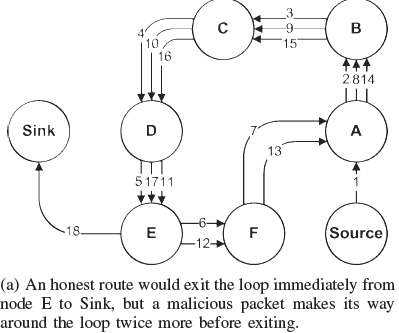


Figure 1.1

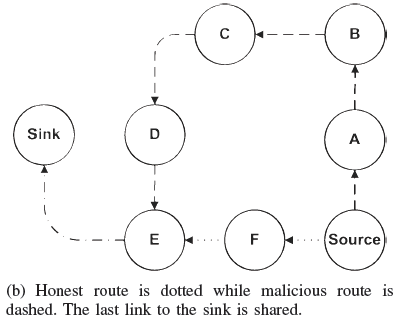


Figure 1.2

**2. WIRELESS SENSOR NETWORKS**

A **wireless sensor network (WSN)** (sometimes called a **wireless sensor and actuator network (WSAN)**) are spatially distributed autonomous sensors to *monitor* physical or environmental conditions,[[2]](https://en.wikipedia.org/wiki/Wireless_sensor_network" \l "cite_note-2) such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling *control* of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding.

**2.1. CHARACTERISTICS**

The main characteristics of a WSN include:

1. Power consumption constraints for nodes using batteries or
2. Ability to cope with node failures
3. Mobility of nodes
4. Heterogeneity of nodes
5. Scalability to large scale of deployment
6. Ability to withstand harsh environmental conditions
7. Ease of use
8. Cross-layer design

Cross-layer is becoming an important studying area for wireless communications. In addition, the traditional layered approach presents three main problems:

1. Traditional layered approach cannot share different information among different layers ， which leads to each layer not having complete information. The traditional layered approach cannot guarantee the optimization of the entire network.
2. The traditional layered approach does not have the ability to adapt to the environmental change.
3. Because of the interference between the different users, access conflicts, fading, and the change of environment in the wireless sensor networks, traditional layered approach for wired networks is not applicable to wireless networks.

So the cross-layer can be used to make the optimal modulation to improve the transmission performance, such as data-rate, energy efficiency, QoS ([Quality of Service](https://en.wikipedia.org/wiki/Quality_of_Service)), etc... Sensor nodes can be imagined as small computers which are extremely basic in terms of their interfaces and their components. They usually consist of a *processing unit* with limited computational power and limited memory, *sensors* or [MEMS](https://en.wikipedia.org/wiki/Microelectromechanical_systems) (including specific conditioning circuitry), a *communication device* (usually radio transceivers or alternatively [optical](https://en.wikipedia.org/wiki/Smart_dust)), and a power source usually in the form of a battery. Other possible inclusions are [energy harvesting](https://en.wikipedia.org/wiki/Energy_harvesting) modules,[[10]](https://en.wikipedia.org/wiki/Wireless_sensor_network#cite_note-10) secondary [ASICs](https://en.wikipedia.org/wiki/Application-specific_integrated_circuit), and possibly secondary communication interface (e.g. [RS-232](https://en.wikipedia.org/wiki/RS-232) or USB).

The base stations are one or more components of the WSN with much more computational, energy and communication resources. They act as a gateway between sensor nodes and the end user as they typically forward data from the WSN on to a server. Other special components in routing based networks are routers, designed to compute, calculate and distribute the routing tables.

**2.2. PLATFORM**

**2.2.1 HARDWARE**

One major challenge in a WSN is to produce *low cost* and *tiny* sensor nodes. There are an increasing number of small companies producing WSN hardware and the commercial situation can be compared to home computing in the 1970s. Many of the nodes are still in the research and development stage, particularly their software. Also inherent to sensor network adoption is the use of very low power methods for radio communication and data acquisition.

In many applications, a WSN communicates with a Local Area Network or [Wide Area Network](https://en.wikipedia.org/wiki/Wide_Area_Network) through a gateway. The Gateway acts as a bridge between the WSN and the other network. This enables data to be stored and processed by devices with more resources, for example, in a remotely located [server](https://en.wikipedia.org/wiki/Server_(computing)).

**2.2.2. SOFTWARE**

Energy is the scarcest resource of WSN nodes, and it determines the lifetime of WSNs. WSNs may be deployed in large numbers in various environments, including remote and hostile regions, where ad hoc communications are a key component. For this reason, algorithms and protocols need to address the following issues:

1. Increased lifespan
2. Robustness and fault tolerance
3. Self-configuration

Lifetime maximization: Energy/Power Consumption of the sensing device should be minimized and sensor nodes should be energy efficient since their limited energy resource determines their lifetime. To conserve power, wireless sensor nodes normally power off both the radio transmitter and the radio receiver when not in use.

**2.2.3. OPERATING SYSTEMS**

[Operating systems](https://en.wikipedia.org/wiki/Operating_system) for wireless sensor network nodes are typically less complex than general-purpose operating systems. They more strongly resemble embedded systems, for two reasons. First, wireless sensor networks are typically deployed with a particular application in mind, rather than as a general platform. Second, a need for low costs and low power leads most wireless sensor nodes to have low-power microcontrollers ensuring that mechanisms such as virtual memory are either unnecessary or too expensive to implement.

It is therefore possible to use embedded operating systems such as [eCos](https://en.wikipedia.org/wiki/ECos) or [uC/OS](https://en.wikipedia.org/wiki/UC/OS) for sensor networks. However, such operating systems are often designed with real-time properties.

[TinyOS](https://en.wikipedia.org/wiki/TinyOS) is perhaps the first [[11]](https://en.wikipedia.org/wiki/Wireless_sensor_network#cite_note-11) operating system specifically designed for wireless sensor networks. TinyOS is based on an [event-driven programming](https://en.wikipedia.org/wiki/Event-driven_programming) model instead of [multithreading](https://en.wikipedia.org/wiki/Thread_(computer_science)). TinyOS programs are composed of *event handlers* and *tasks* with run-to-completion semantics. When an external event occurs, such as an incoming data packet or a sensor reading, TinyOS signals the appropriate event handler to handle the event. Event handlers can post tasks that are scheduled by the TinyOS kernel some time later.

[LiteOS](https://en.wikipedia.org/wiki/LiteOS) is a newly developed OS for wireless sensor networks, which provides UNIX-like abstraction and support for the C programming language.

[Contiki](https://en.wikipedia.org/wiki/Contiki) is an OS which uses a simpler programming style in C while providing advances such as [6LoWPAN](https://en.wikipedia.org/wiki/6LoWPAN) and [Protothreads](https://en.wikipedia.org/wiki/Protothreads).

**2.2.4. ONLINE COLLABORATIVE SENSOR DATA MANAGEMENT PLATFORMS**

Online collaborative sensor data management platforms are on-line database services that allow sensor owners to register and connect their devices to feed data into an online database for storage and also allow developers to connect to the database and build their own applications based on that data. Such platforms simplify online collaboration between users over diverse data sets ranging from energy and environment data to that collected from transport services. Other services include allowing developers to embed real-time graphs & widgets in websites; analyze and process historical data pulled from the data feeds; send real-time alerts from any data stream to control scripts, devices and environments.

The architecture of the Wikisensing system [[12]](https://en.wikipedia.org/wiki/Wireless_sensor_network#cite_note-12) describes the key components of such systems to include APIs and interfaces for online collaborators, a middleware containing the business logic needed for the sensor data management and processing and a storage model suitable for the efficient storage and retrieval of large volumes of data.

**2.3. APPLICATIONS**

**2.3.1. AREA MONITORING**

Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. A military example is the use of sensors detect enemy intrusion; a civilian example is the [geo-fencing](https://en.wikipedia.org/wiki/Geo-fence) of gas or oil pipelines.

**2.3.2. HEALTH CARE MONITORING**

The medical applications can be of two types: wearable and implanted. Wearable devices are used on the body surface of a human or just at close proximity of the user.[[5]](https://en.wikipedia.org/wiki/Wireless_sensor_network#cite_note-5) The implantable medical devices are those that are inserted inside human body. There are many other applications too e.g. body position measurement and location of the person, overall monitoring of ill patients in hospitals and at homes. Body-area networks can collect information about an individual's health, fitness, and energy expenditure.

**2.3.3. ENVIRONMENTAL/EARTH SENSING**

There are many applications in monitoring environmental parameters, examples of which are given below. They share the extra challenges of harsh environments and reduced power supply.

**2.3.4. AIR POLLUTION MONITORING**

Wireless sensor networks have been deployed in several cities to monitor the concentration of [dangerous gases for citizens](https://en.wikipedia.org/wiki/Air_pollution). These can take advantage of the ad hoc wireless links rather than wired installations, which also make them more mobile for testing readings in different areas.

**2.3.5. FOREST FIRE DETECTION**

A network of Sensor Nodes can be installed in a forest to detect when a [fire](https://en.wikipedia.org/wiki/Forest_fire) has started. The nodes can be equipped with sensors to measure temperature, humidity and gases which are produced by fire in the trees or vegetation. The early detection is crucial for a successful action of the firefighters; thanks to Wireless Sensor Networks, the fire brigade will be able to know when a fire is started and how it is spreading.

**2.3.6. LANDSLIDE DETECTION**

A [landslide](https://en.wikipedia.org/wiki/Landslide) detection system makes use of a wireless sensor network to detect the slight movements of soil and changes in various parameters that may occur before or during a landslide. Through the data gathered it may be possible to know the occurrence of landslides long before it actually happens.

**2.3.7. WATER QUALITY MONITORING**

[Water quality](https://en.wikipedia.org/wiki/Water_quality) monitoring involves analyzing water properties in dams, rivers, lakes & oceans, as well as underground water reserves. The use of many wireless distributed sensors enables the creation of a more accurate map of the water status, and allows the permanent deployment of monitoring stations in locations of difficult access, without the need of manual data retrieval.

**2.3.8 NATURAL DISASTER PREVENTION**

Wireless sensor networks can effectively act to prevent the consequences of [natural disasters](https://en.wikipedia.org/wiki/Natural_disaster), like floods. Wireless nodes have successfully been deployed in rivers where changes of the water levels have to be monitored in real time.

**2.3.9. MACHINE HEALTH MONITORING**

Wireless sensor networks have been developed for machinery condition-based maintenance (CBM) as they offer significant cost savings and enable new

functionality.

Wireless sensors can be placed in locations difficult or impossible to reach with a wired system, such as rotating machinery and untethered vehicles.

**2.3.10. DATA LOGGING**

Wireless sensor networks are also used for the collection of data for monitoring of environmental information, this can be as simple as the monitoring of the temperature in a fridge to the level of water in overflow tanks in nuclear power plants. The statistical information can then be used to show how systems have been working. The advantage of WSNs over conventional loggers is the "live" data feed that is possible.

**2.3.11. WATER/WASTE WATER MONITORING**

Monitoring the quality and level of water includes many activities such as checking the quality of underground or surface water and ensuring a country’s water infrastructure for the benefit of both human and animal. It may be used to protect the wastage of water.

**2.3.12. STRUCTURAL HEALTH MONITORING**

Wireless sensor networks can be used to monitor the condition of civil infrastructure and related geo-physical processes close to real time, and over long periods through data logging, using appropriately interfaced sensors.

**2.4. DISTRIBUTED SENSOR NETWORK**

If a centralized architecture is used in a sensor network and the central node fails, then the entire network will collapse, however the reliability of the sensor network can be increased by using a distributed control architecture. Distributed control is used in WSNs for the following reasons:

1. Sensor nodes are prone to failure,
2. For better collection of data,
3. To provide nodes with backup in case of failure of the central node.

There is also no centralized body to allocate the resources and they have to be self-organized.

**3. VAMPIRE ATTACKS**

We define a Vampire attack as the composition and transmission of a message that causes more energy to be consumed by the network than if an honest node transmitted a message of identical size to the same destination, although using different packet headers. We measure the strength of the attack by the ratio of network energy used in the benign case to the energy used in the malicious case, i.e. the ratio of network-wide power utilization with malicious nodes present to energy usage with only honest nodes when the number and size of packets sent remains constant.

Vampire attack means creating and sending messages by malicious node which causes more energy consumption by the network leading to slow depletion of node’s battery life.

There are two types of vampire attacks. They are,

1. Attack on Stateless Protocols
2. Attack on State full Protocols
   1. **STATELESS PROTOCOLS:**
3. Same as source routing protocol.
4. Source node specifies entire route to destination within packet header.
5. Intermediaries don’t make independent forwarding decisions.
   1. **STATE FULL PROTOCOLS:**
6. Nodes are aware of their topology, state, forwarding decisions.
7. Nodes make local forwarding decisions on that stored state.
8. Two important classes are: link state and distance –vector.
   1. **TYPES OF ATTACKS ON STATELESS PROTOCOLS:**
9. Carousel attack
10. Stretch attack

**3.3.1. CAROUSEL ATTACK:**

In this attack, an adversary sends a packet with a route composed as a series of loops, such that the same node appears in the route many times. This strategy can be used to increase the route length beyond the number of nodes in the network, only limited by the number of allowed entries in the source route.

1. Adversary sends packets with routes composed of a series of loops.
2. Exploits limited verification of message headers at forwarding nodes.
3. Used to increase the route length beyond no of nodes in network
4. Theoretical limit: energy usage increase by a factor of O (λ), where λ is the maximum route length.

The below figure 3.1 demonstrates the carousel attack.

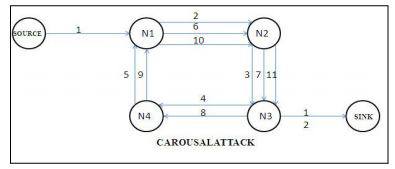


Figure 3.1

**3.3.2. STRETCH ATTACK:**

Another attack in the same vein is the stretch attack, where a malicious node constructs artificially long source routes, causing packets to traverse a larger than optimal number of nodes. An honest source would select the route Source → F → E → Sink in figure 1.2, affecting four nodes including itself, but the malicious node selects a longer route, affecting all nodes in the network. These routes cause nodes that do not lie along the honest route to consume energy by forwarding packets they would not receive in honest scenarios.

Adversary constructs artificially long routes traversing every node in the network.

1. Causes packets to traverse larger than optimal no of nodes
2. Causes nodes that doesn’t lie on optimal path to process packets
3. Theoretical limit: energy usage increase of factor O (min (N, λ)), where N is the number of nodes in the network and λ is the maximum path length allowed.
4. Potentially less damaging per packet than the carousel attack, as the no of hops per packet is bounded by the number of network nodes.

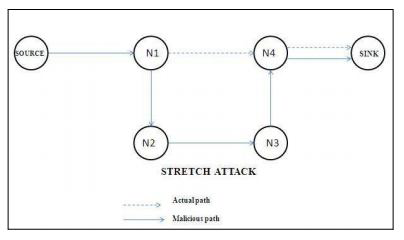


Figure 3.2

* 1. **TYPES OF ATTACKS ON STATE FULL PROTOCOLS:**

1. Directional antenna attack
2. Malicious Discovery attack

**3.4.1. DIRECTIONAL ANTENNA ATTACK**

1. Energy can be wasted by restarting packet in various parts of network
2. Using a directional antenna adversaries can deposit packets in arbitrary parts of the network.
3. Consumes energy of nodes that would not have had to process the original packet.
4. Half Wormhole attack – as a directional antenna constitutes a private communication channel.
5. Packet leashes cannot prevent this attack as they are not to protect against malicious message sources only intermediaries.

**3.4.2. MALICIOUS DISCOVERY ATTACK**

1. Also known as spurious route discovery.
2. Falsely claims that a link is down or claim a new link to non-existent node.
3. More serious when nodes claim a long distance route has changed.
4. Trivial in open networks and in closed networks: repeatedly announce and withdraw routes.
5. Theoretical energy usage increase of a factor of O (N) per packet.

**4. PROPOSED SYSTEM**

To avoid the vampire attacks, we study that topology discovery and packet forwarding which will takes the message to the destination where attacks may not happen.

**4.1. CLEAN-SLATE SENSOR NETWORK ROUTING:**

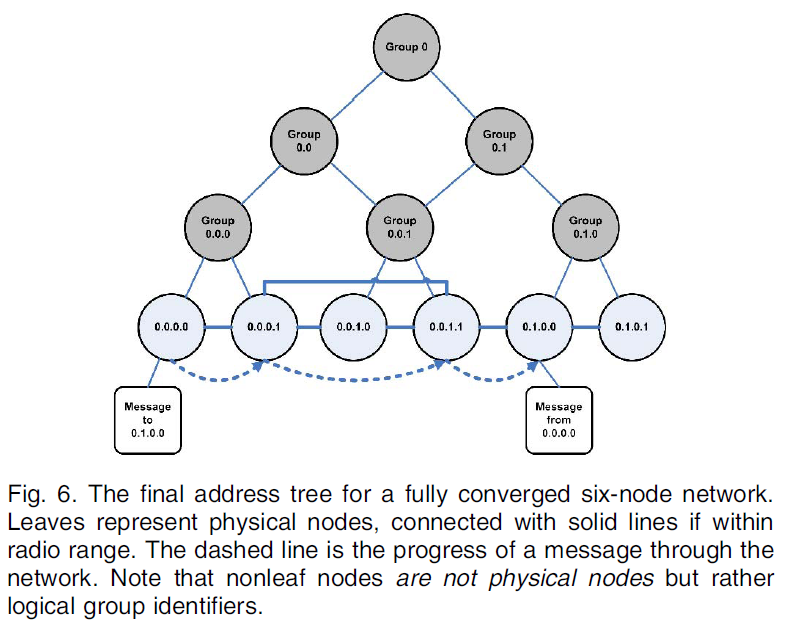
The original version of the protocol, although designed for security, is vulnerable to Vampire attacks. PLGP consists of a topology discovery phase, followed by a packet forwarding phase, with the former optionally repeated on a fixed schedule to ensure that topology information stays current. (There is no on-demand discovery.) Discovery deterministically organizes nodes into a tree that will later be used as an addressing scheme. When discovery begins, each node has a limited view of the network — the node knows only itself. Nodes discover their neighbors using local broadcast, and form ever-expanding “neighborhoods,” stopping when the entire network is a single group. Throughout this process, nodes build a tree of neighbor relationships and group membership that will later be used for addressing and routing.

1. **PLGP**: a clean-slate secure sensor network routing protocol by Parno et al.
2. The original version of the protocol is vulnerable to Vampire attacks.
3. PLGP consists of a topology discovery phase, followed by a packet forwarding phase.
4. Discovery deterministically organizes nodes into a tree that will later be used as an addressing scheme
   1. repeated on a fixed schedule
   2. Discovery deterministically organizes nodes into a tree that will later be used as an addressing scheme
   3. When discovery begins, each node has a limited view of the network—the node knows only itself. Nodes discover their neighbors using local broadcast, and form ever expanding “neighborhoods,” stopping when the entire network is a single group. Throughout this process, nodes build a tree of neighbor relationships and group membership that will later be used for addressing and routing.

**4.1.1. PLGP**

At the end of discovery, each node should compute the same address tree as other nodes.

All leaf nodes in the tree are physical nodes in the network, and their virtual addresses correspond to their position in the tree.

****

**4.1.1. PLGP FORMING GROUPS AND ADDRESSING**

Each node starts as its own group of size one, with a virtual address 0. Nodes who overhear presence broadcasts form groups with their neighbors. When two individual nodes (each with an initial address 0) form a group of size two, one of them takes the address 0, and the other becomes 1.

Like individual nodes, each group will initially choose a group address 0, and will choose 0 or 1 when merging with another group. Each group member prepends the group address to their own address, e.g., node 0 in group 0 becomes 0.0, and node 0 in group 1 becomes 1.0, and so on. Each time two groups merge, the address of each node is lengthened by 1 bit. Implicitly, this forms a binary tree of all addresses in the network, with node addresses as leaved.

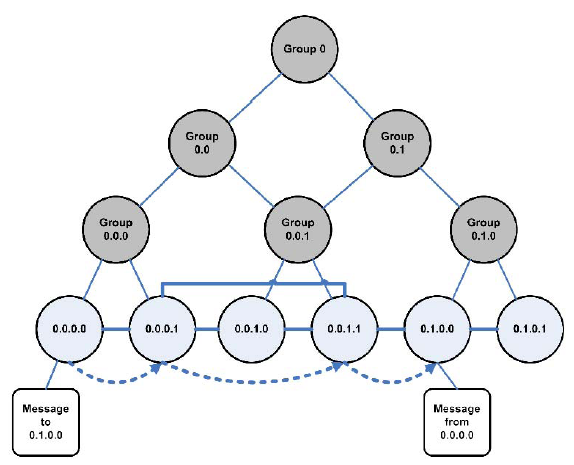


Figure 4.1

When larger groups merge, they both broadcast their group IDs (and the IDs of all group members) to each other, and proceed with a merge protocol identical to the two-node case.

Groups that have grown large enough that some members are not within radio range of other groups will communicate through “gateway nodes,” which are within range of both groups.

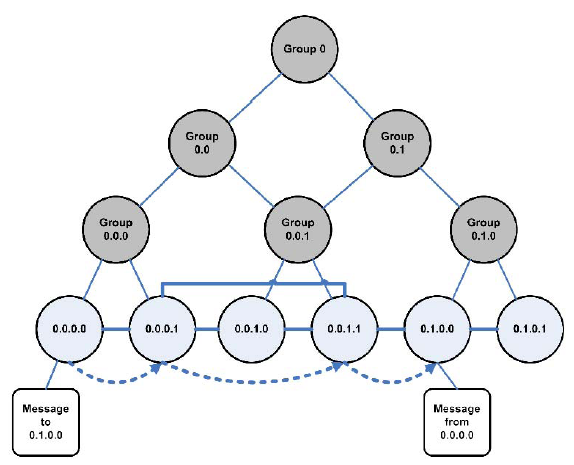


Figure 4.2

Every node within a group will end up with a next-hop path to every other group, as in distance vector. Topology discovery proceeds in this manner until all network nodes are members of a single group. By the end of topology discovery, each node learns every other node’s virtual address, public key, and certificate, since every group members knows the identities of all other group members and the network converges to a single group.

**4.2. TOPOLOGY DISCOVERY:**

Discovery begins with a time-limited period during which every node must announce its presence by broadcasting a certificate of identity, including its public key (from now on referred to as node ID), signed by a trusted offline authority. Each node starts as its own group of size one, with a virtual address 0. Nodes who overhear presence broadcasts form groups with their neighbors. When two individual nodes (each with an initial address 0) form a group of size two, one of them takes the address 0, and the other becomes 1. Groups merge preferentially with the smallest neighboring group, which may be a single node. We may think of groups acting as individual nodes, with decisions made using secure multiparty computation.

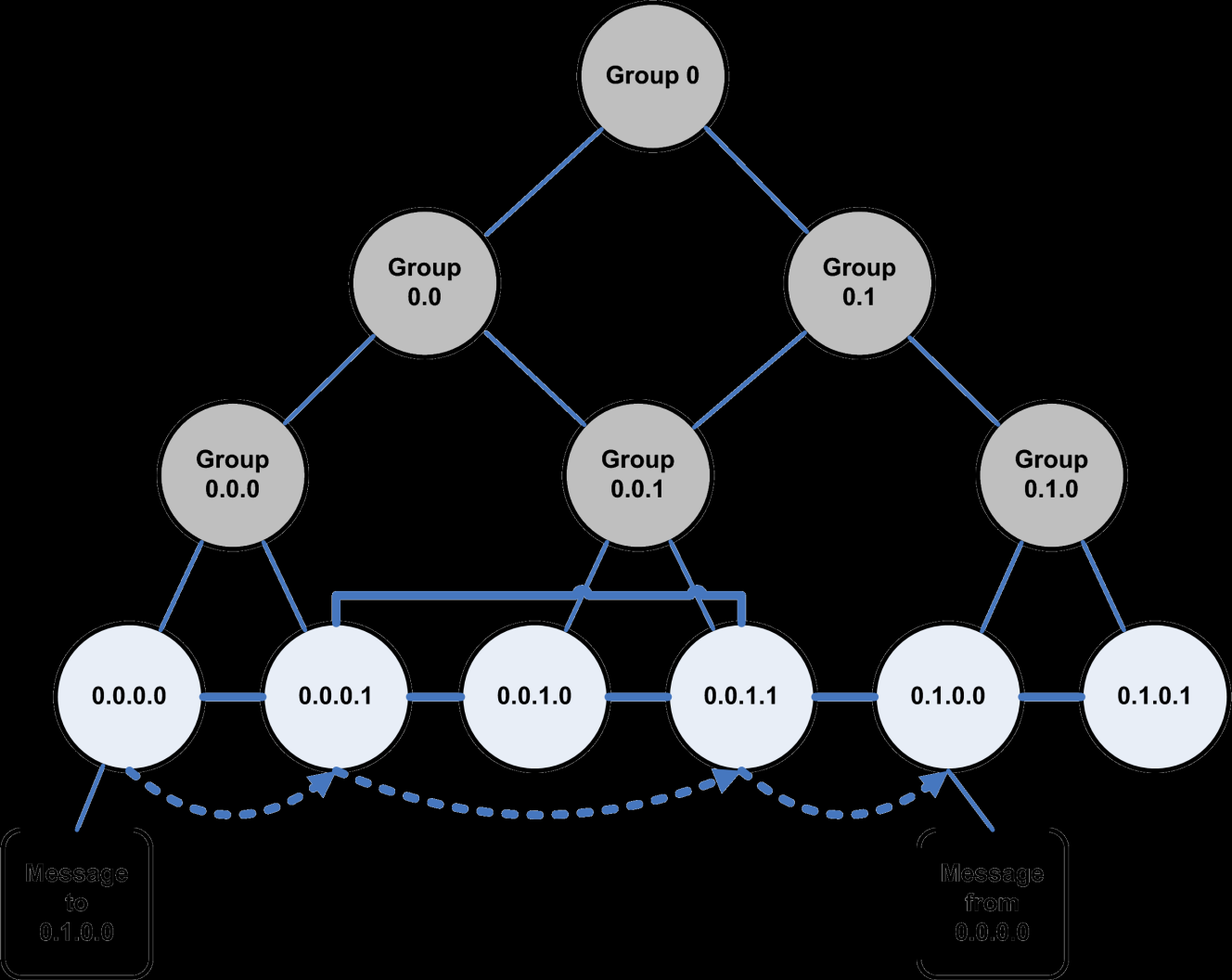


Figure 4.1

**4.3. PACKET FORWARDING**

During the forwarding phase, all decisions are made independently by each node. When receiving a packet, a node determines the next hop by finding the most significant bit of its address that differs from the message originator’s address.

Thus every forwarding event shortens the logical distance to the destination, since node addresses should be strictly closer to the destination.

In PLGP, forwarding nodes do not know what path a packet took, allowing adversaries to divert packets to any part of the network, even if that area is logically further away from the destination than the malicious node. This makes PLGP vulnerable to Vampire attacks.

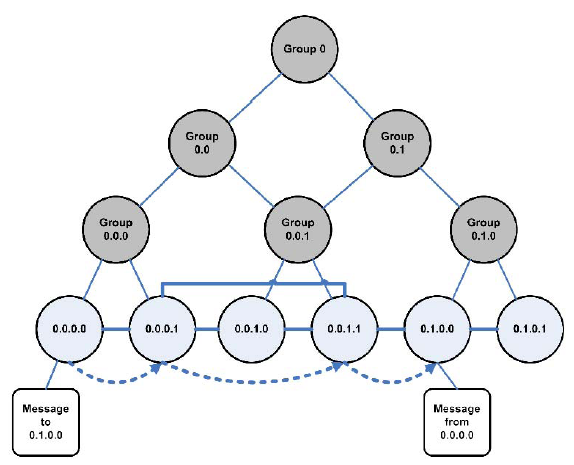


Figure 4.2

An honest node has no way to tell that the packet it just received is moving away from the destination; the only information available to the honest node is its own address and the packet destination address, but not the address of the previous hop (who can lie).

**4.4. PROVABLE SECURITY AGAINST VAMPIRE ATTACKS**

**4.4.1. NO-BACKTRACKING PROPERTY:**

Satisfied for a given packet if and only if it consistently makes progress toward its destination in the logical network address space.

More formally:

No-backtracking is satisfied if every packet p traverses the same number of hops whether or not an adversary is present in the network.

Case 1: L is honest L--🡪..hops..-->D

Case 2: L is Malicious L--🡪..hops..-->D

1. Same # of Hops
2. Same network wide energy utilization
3. is independent of the actions of malicious nodes

No-backtracking implies Vampire resistance

PLGPdoes not satisfy **No-backtracking property**:

PLGP differs from other protocols in that packets paths are further bounded by a tree, forwarding packets along the shortest route through the tree that is allowed by the physical topology. Since the tree implicitly mirrors the topology (two nodes have the same parent if and only if they are physical neighbors, and two nodes sharing an ancestor have a network path to each other), and since every node holds an identical copy of the address tree, every node can verify the optimal next logical hop.

However, this is not sufficient for no-backtracking to hold, since nodes cannot be certain of the path previously traversed by a packet.

Adversaries can always lie about their local metric, and so PLGP is still vulnerable to directional antenna/wormhole attacks, which allow adversaries to divert packets to any part of the network.

**4.4.2. PROPOSE PLGP WITH ATTESTATIONS (PLGPA):**

1. Add a verifiable path history to every PLGP packet
2. The resulting protocol, PLGP with attestations (PLGPa) uses this packet history together with PLGP’s tree routing structure so every node can securely verify progress, preventing any significant adversarial influence on the path taken by any packet which traverses at least one honest node.
3. These signatures form a chain attached to every packet, allowing any node receiving it to validate its path. Every forwarding node verifies the attestation chain to ensure that the packet has never traveled away from its destination in the logical address space.

**4.4.2.1. PLGPA SATISFIES NO-BACKTRACKING**

1. Since all messages are signed by their originator, messages from honest nodes cannot be arbitrarily modified by malicious nodes wishing to remain undetected. Rather, the adversary can only alter packet fields that are changed en route (and so are not authenticated), so only the route attestation field can be altered, shortened, or removed entirely.
2. To prevent truncation, which would allow Vampires to hide the fact that they are moving a packet away from its destination, use one-way signature chain construction which allow nodes to add links to an existing signature chain, but not remove links, making attestations append only.
3. so we define the hop count of a packet as follows:

Definition: The hop count of packet p, received or forwarded by an honest node, is no greater than the number of entries in p’s route attestation field, plus 1.

1. When any node receives a message, it checks that every node in the path attestation 1) has a corresponding entry in the signature chain, and 2) is logically closer to the destination than the previous hop in the chain. This way, forwarding nodes can enforce the forward progress of a message, preserving no-backtracking.
2. If no attestation is present, the node checks to see if the originator of the message is a physical neighbor. Since messages are signed with the originator’s key, malicious nodes cannot falsely claim to be the origin of a message, and therefore do not benefit by removing attestations.
3. Since no-backtracking guarantees packet progress, and PLGPa preserves no-backtracking, it is the only protocol discussed so far that provably bounds the ratio of energy used in the adversarial scenario to that used with only honest nodes to 1, and by the definition of no-backtracking PLGPa resists Vampire attacks.
4. This is achieved because packet progress is securely verifiable can modify to allow for limited backtracking (-backtracking, as opposed to original 0-backtracking scheme), which provides some leeway in the way no-backtracking is verified, allowing a certain amount of total backtracking per packet within a security parameter (Case of reaching a dead-end path).

**4.5. PERFORMANCE CONSIDERATIONS:**

1. PLGPa includes path attestations which increase the size of every packet, incurring penalties in terms of bandwidth use, and thus radio power. Adding extra packet verification requirements for intermediate nodes also increases processor utilization, requiring time, and additional power.
2. There is nothing to be gained in completely no adversarial environments, but in the presence of even a small number of malicious nodes, the increased overhead becomes worthwhile when considering the potential damage of Vampire attacks.
3. In total, the overhead on the entire network of PLGPa (over PLGP) when using 32-bit processors or dedicated cryptographic accelerator is the energy equivalent of 90 additional bytes per packet, or a factor O(xλ), whereλ is the path length between source and destination and x is 1.2-7.5, depending on average packet size (512 and 12 bytes, respectively).
4. Even without dedicated hardware, the cryptographic computation required for PLGPa is tractable even on 8-bit processors, although with up to a factor of 30 performance penalty, but this hardware configuration is increasingly uncommon.

**4.6. SECURING THE DISCOVERY PHASE**

Enforce rate limits in a number of ways, such as neighbor throttling or one-way hash chains [14]. We can also optimize discovery algorithms to minimize our window of vulnerability. If a network survives the high risk discovery period, it is unlikely to suffer serious damage from Vampires during normal packet forwarding.

1. An attack in discovery phase:
   1. Malicious nodes can use directional antennas to masquerade neighbors to any or all nodes in the network, and therefore look like a group of size one, with which other groups will try to preferentially merge. Merge requests are composed of the requested group’s ID as well as all the group members’ IDs, and the receiving node will flood this request to other group members.
   2. Other groups will issue merge requests, which the Vampire can deny. In PLGP, denials are only allowed if another merge is in progress, so if we modify the reject message to include the ID of the group with which the merge is in progress (and a signature for nonrepudiation), these messages can be kept and replayed at the end of the topology discovery period, detecting and removing nodes who incorrectly deny merge requests. Vampires could maintain the illusion that it is a neighbor of a given group. Since join events require multiparty computation and are flooded throughout the group, this makes for a fairly effective attack.
2. The bound we can place on malicious discovery damage in PLGPa is still unknown. Moreover, if we can conclude that a single malicious node causes a factor of k energy increase during discovery (and is then expelled), it is not clear how that value scales under collusion among multiple malicious nodes.

**5. UML DIAGRAMS FOR THE PROPOSED SYSTEM:**

**5.1. USE-CASE DIAGRAM**

****

**5.2. CLASS DIAGRAM**

****

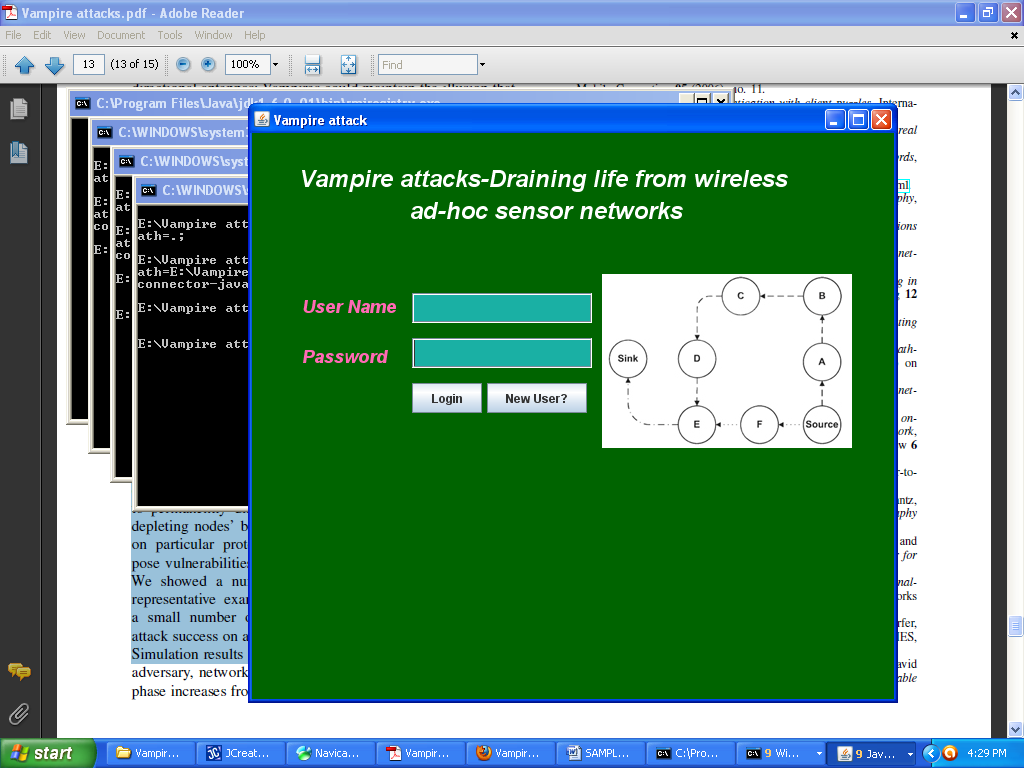
**5.3. SEQUENCE DIAGRAM:**

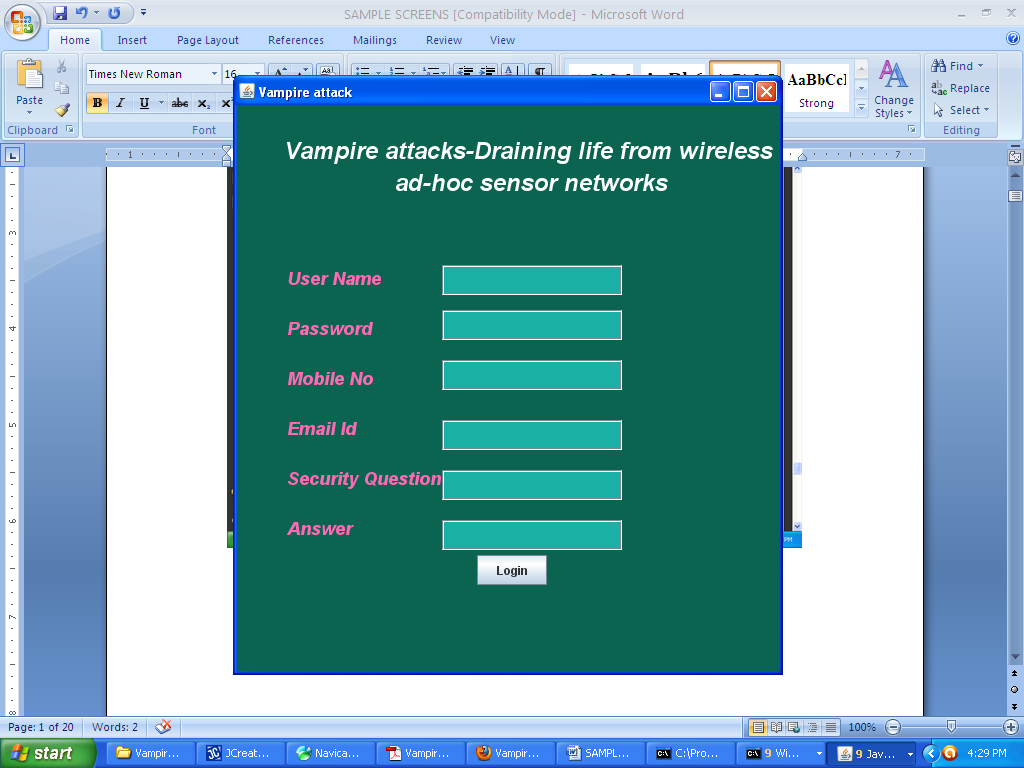
****

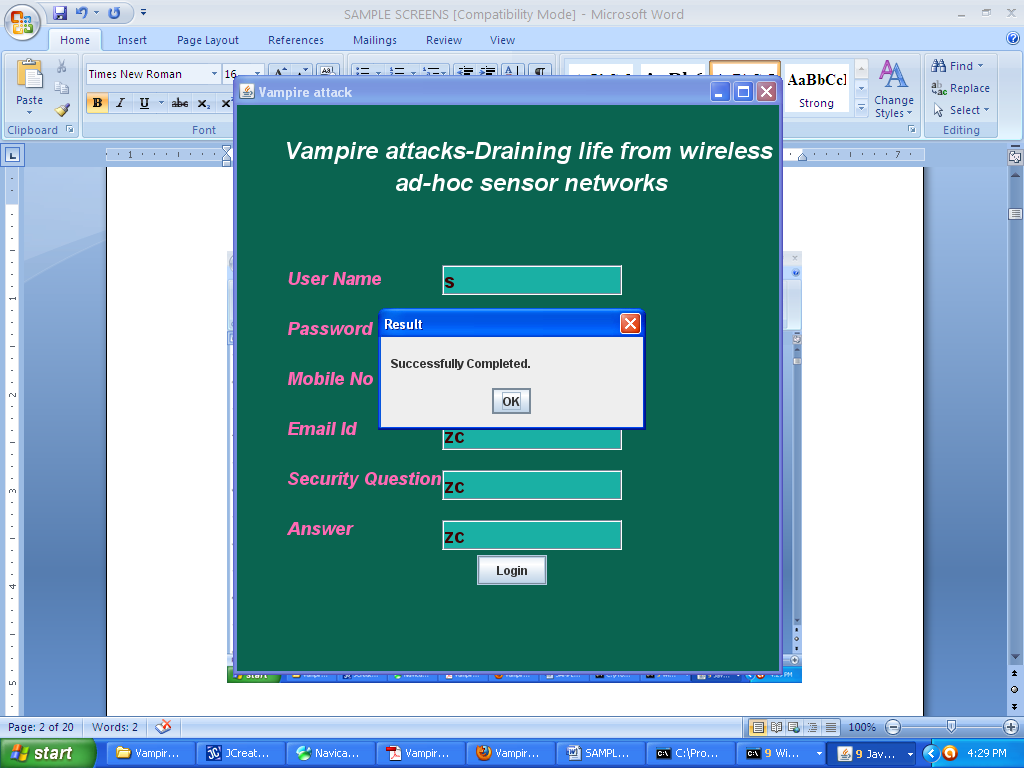
**5.4. ACTIVITY DIAGRAM:**

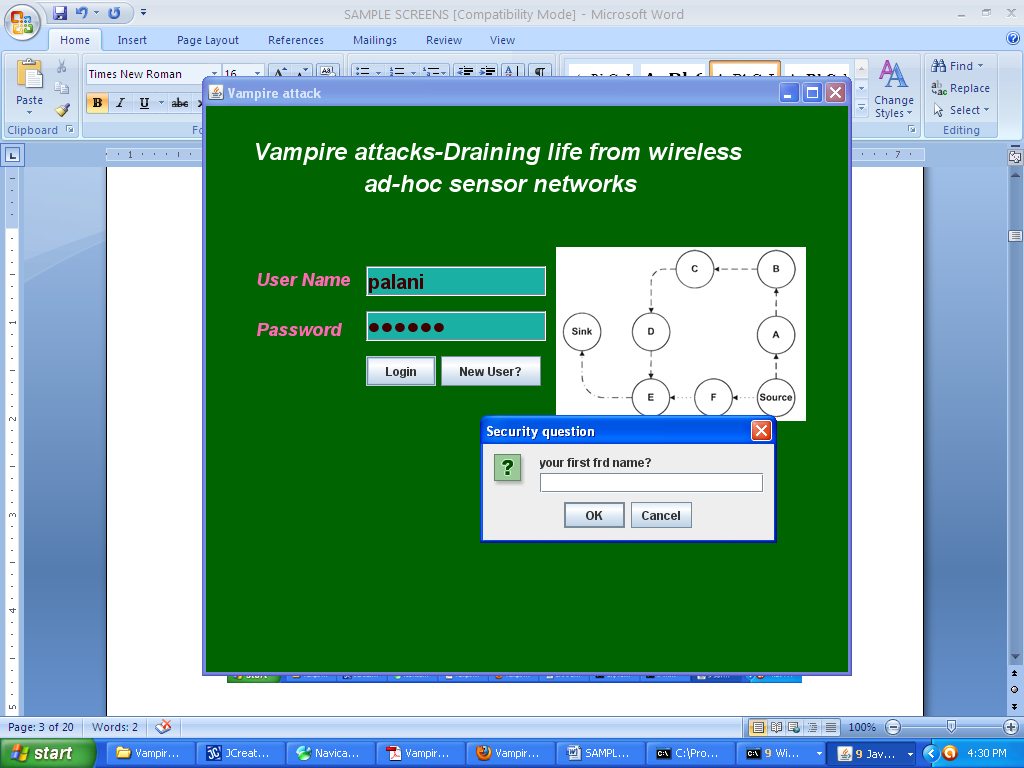
****

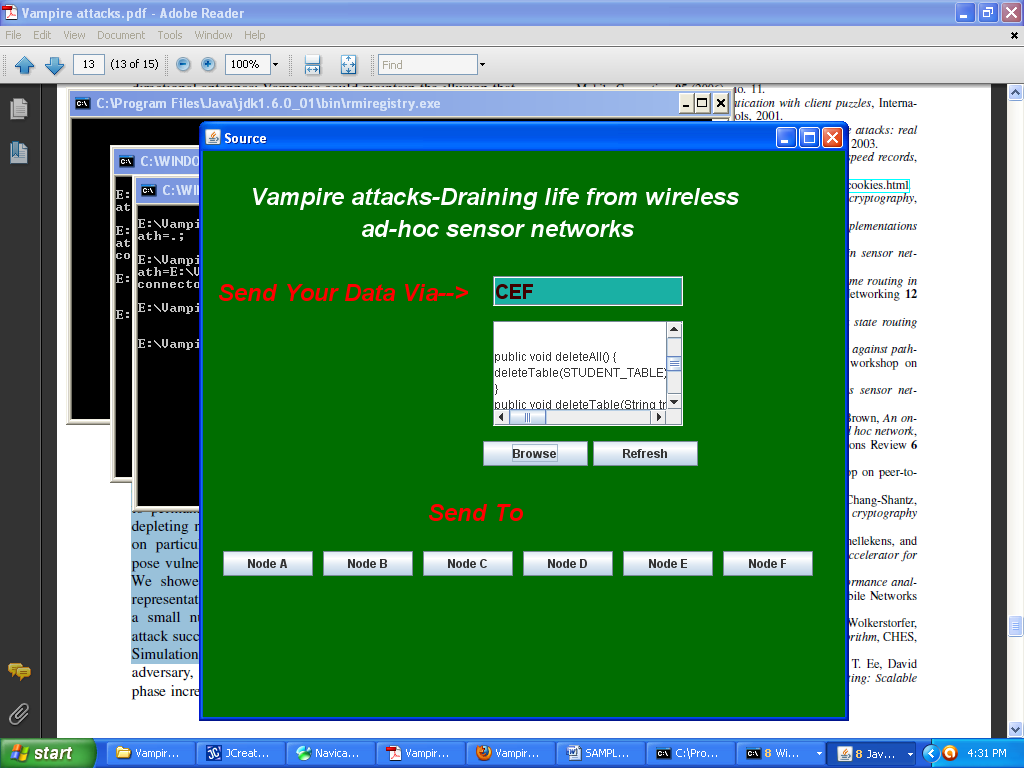
**6. SCREENSHOTS:**

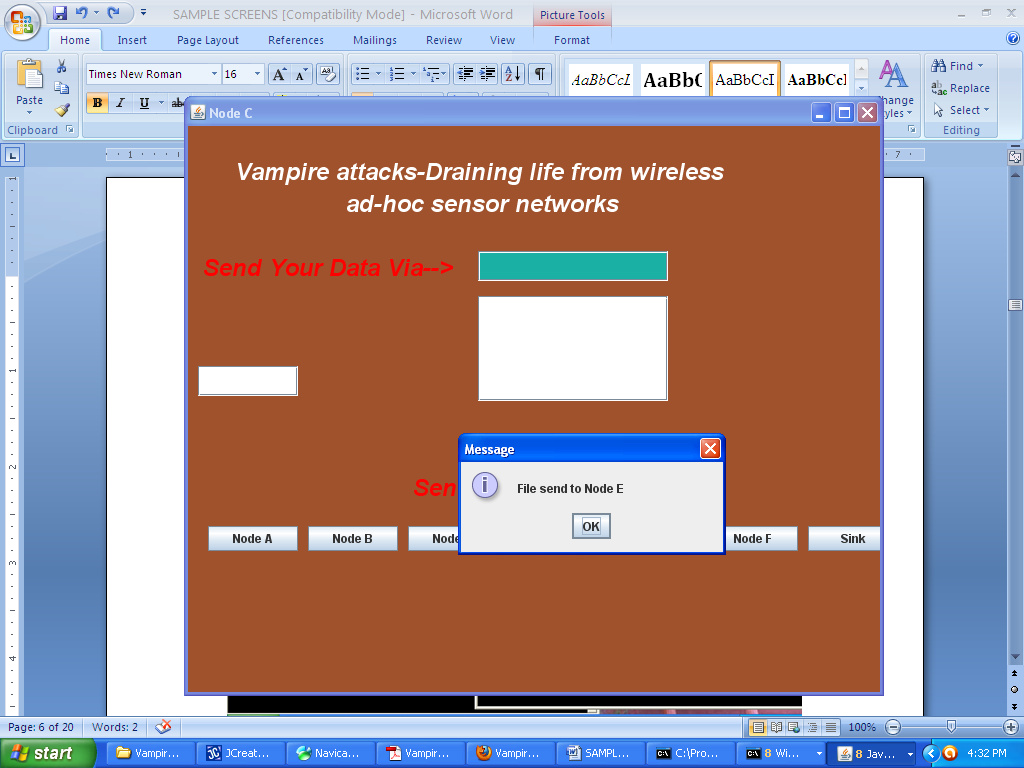
****

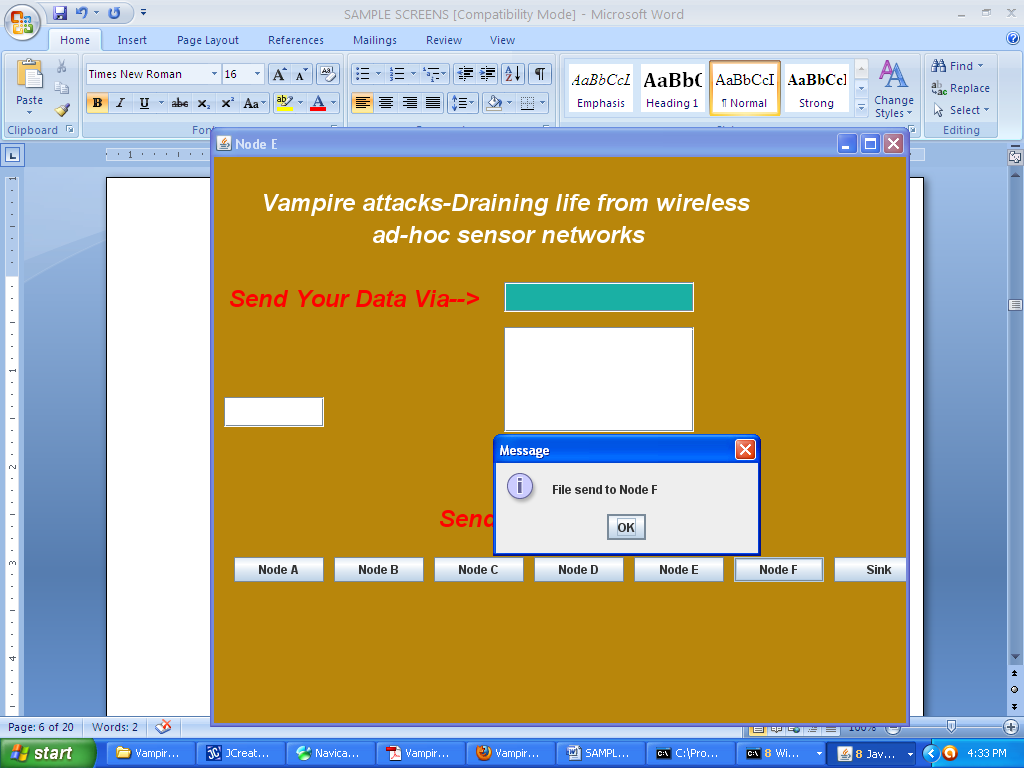
****

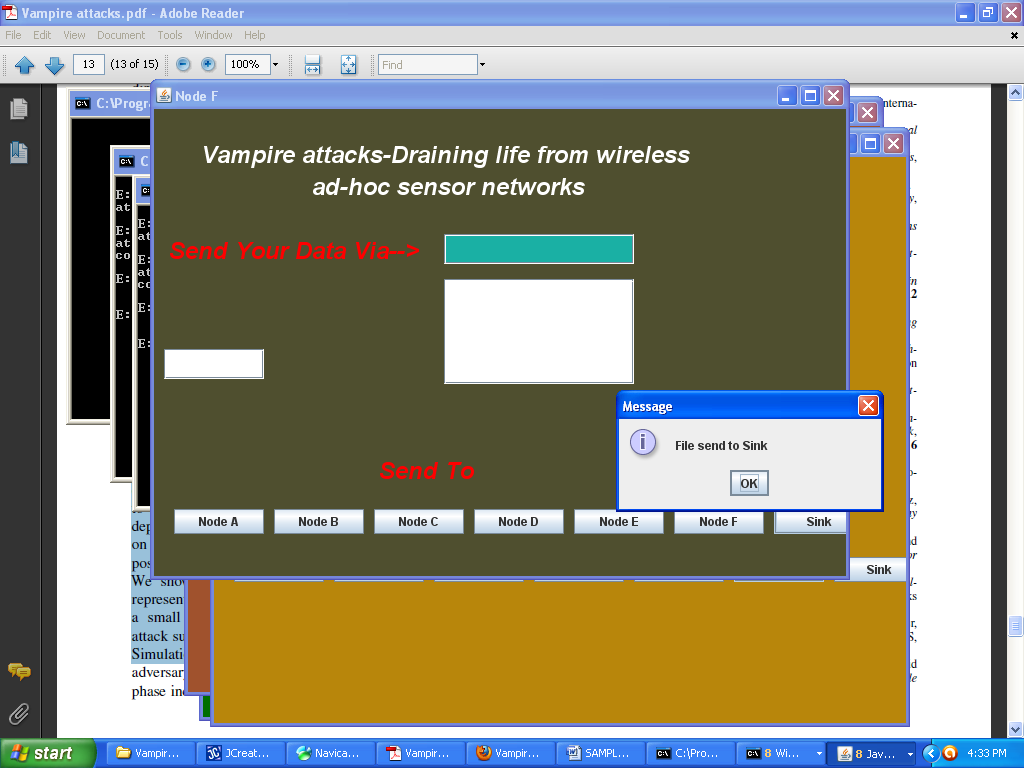
****

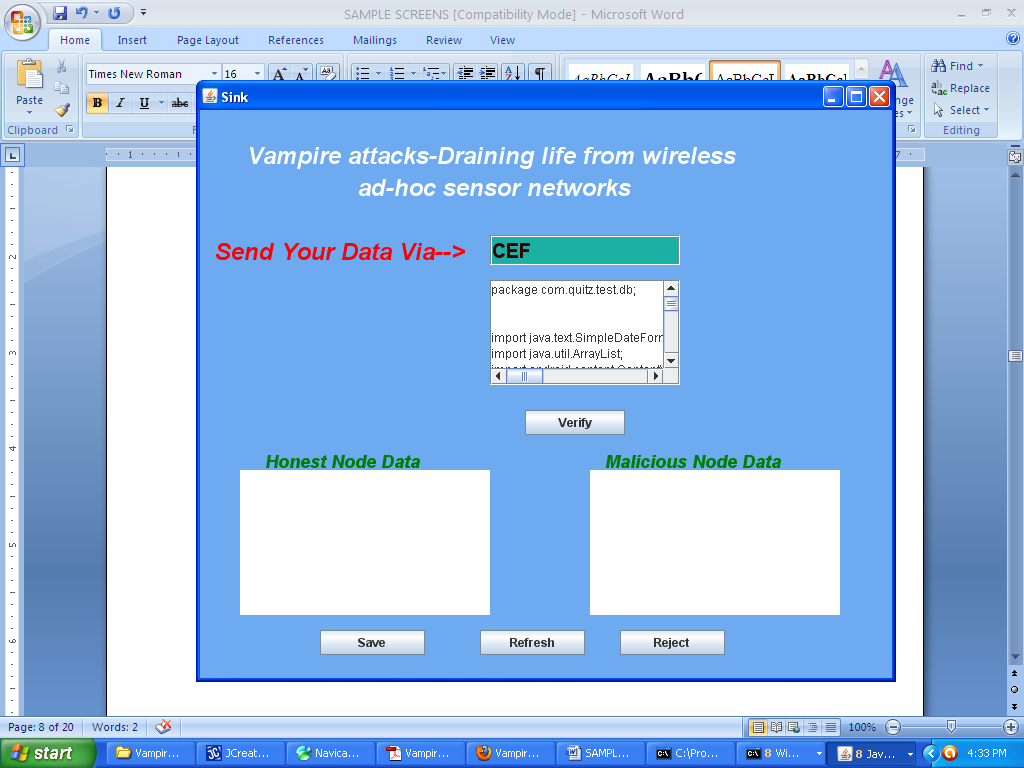
****

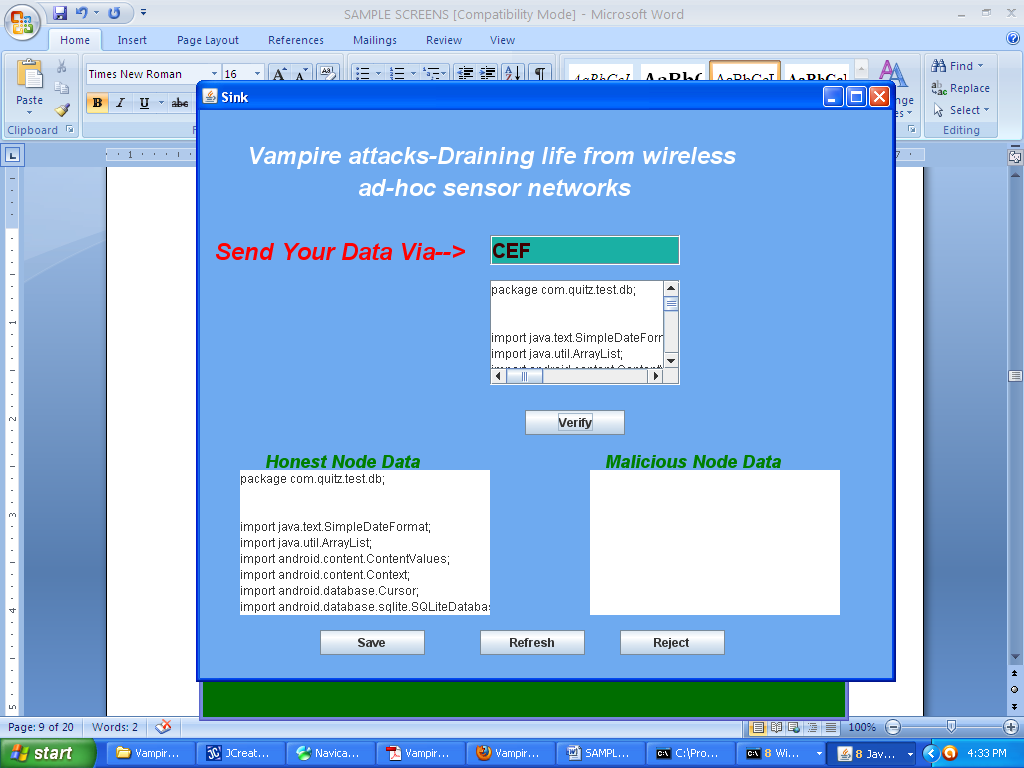
****

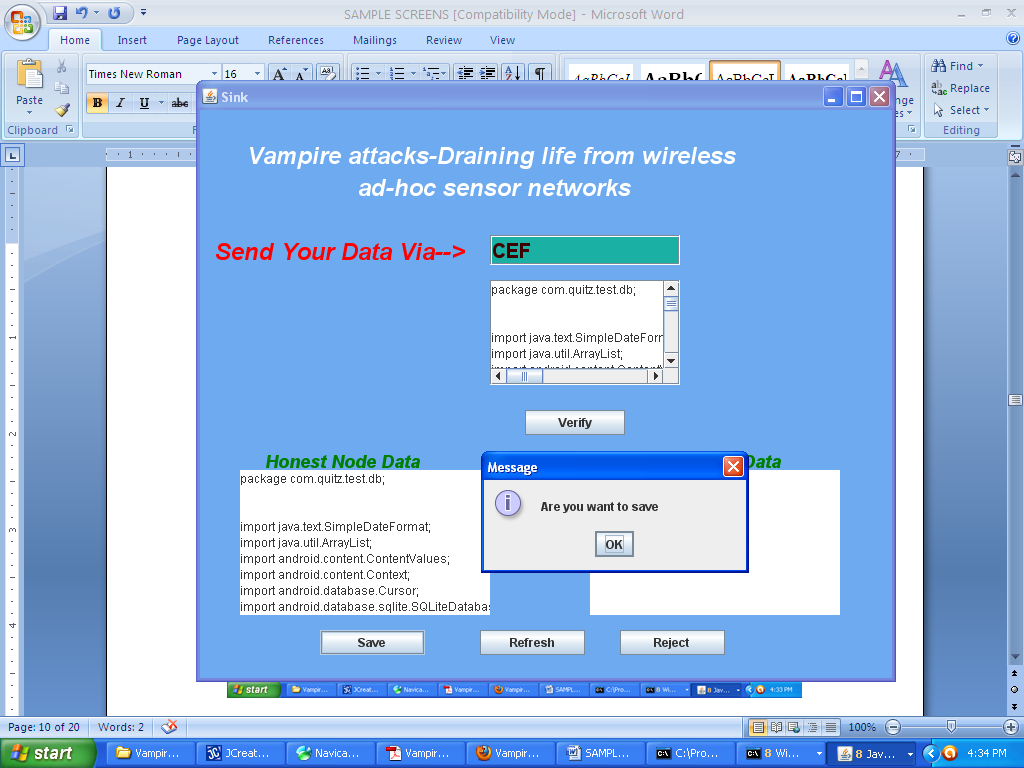
****

****

****

****

****

****

**7. CONCLUSION**

1. This project defined Vampire attacks, a new class of resource consumption attacks that use routing protocols to permanently disable ad hoc wireless sensor networks by depleting nodes’ battery power. These attacks do not depend on particular protocols or implementations, but rather expose vulnerabilities in a number of popular protocol classes.
2. This project showed a number of proof-of-concept attacks against representative examples of existing routing protocols using a small number of weak adversaries, and measured their attack success on a randomly generated topology of 30 nodes. Simulation results show that depending on the location of the adversary, network energy expenditure during the forwarding phase increases from between 50 to 1,000 percent. Theoretical worst case energy usage can increase by as much as a factor of O (N) per adversary per packet, where N is the network size.
3. This project proposed defenses against some of the forwarding-phase attacks and described PLGPa, the first sensor network routing protocol that provably bounds damage from Vampire attacks by verifying that packets consistently make progress toward their destinations.
4. The project have not offered a fully satisfactory solution for Vampire attacks during the topology discovery phase, but suggested some intuition about damage limitations possible with further modifications to PLGPa.

**8. REFERENCES**

1. Imad Aad, Jean-Pierre Hubaux, and Edward W. Knightly, Denial of service resilience in ad hoc networks, MobiCom, 2004.
2. Gergely Acs, Levente Buttyan, and Istvan Vajda, Provably secure ondemand source routing in mobile ad hoc networks, IEEE Transactions on Mobile Computing 05 (2006), no. 11.
3. Tuomas Aura, Dos-resistant authentication with client puzzles, International workshop on security protocols, 2001.
4. John Bellardo and Stefan Savage, 802.11 denial-of-service attacks: real vulnerabilities and practical solutions, USENIX security, 2003.