

Mobility Support for Geo-Encryption

Omar Al-Ibrahim, Ala Al-Fuqaha, Doug Van Dyk*, Nolen Akerman*

Department of Computer Science
Western Michigan University
Kalamazoo, MI
{oalibrah, alfuqaha} @cs.wmich.edu

Abstract— We propose a protocol to support mobility for Geo-Encryption by allowing mobile nodes to exchange movement parameters, so that a sender is able to geo-encrypt messages to a moving decryption zone that contains a mobile node’s estimated location. We also present methods for estimating the node’s movement parameters to allow for Geo-Encryption. Finally, we evaluate our model by measuring the induced overhead to the network and its performance in terms of decryption ratio.

Index Terms— geo-encryption, location-based security, GPS-based encryption

I. INTRODUCTION

GPS-based encryption (or geo-encryption) is an innovative technique that uses GPS-technology to encode location information into the encryption keys to provide location based security. GPS-based encryption adds another layer of security on top of existing encryption methods by restricting the decryption of a message to a particular location and time period. Applying this technique to a mobile environment, with a dynamically changing topology, requires a protocol to handle the distribution of movement information so that communicating hosts can keep track of each others locations. Existing GPS-based encryption techniques ([3] and [4]) have limited support for mobile nodes, therefore, we propose a protocol to support mobility for existing geo-encryption techniques to allow mobile nodes to exchange movement parameters so that a sender is able to geo-encrypt messages to a moving decryption zone that contains a mobile node’s estimated location. We also simulate this protocol to find out its performance and evaluate its scalability in a mobile network.

The paper is organized as follows: Section II introduces a model for existing geo-encryption approaches. Section III and IV give an overview on the proposed mobility model and the geo-encryption protocol. Sections V and VI show the simulation parameters, mobility file and present the results of our simulation study. Finally, we draw conclusions in Section VII.

* Stryker Medical, 3800 E. Centre Avenue, Portage, MI 49002.

II. DENNING’S MODEL OF GEO-ENCRYPTION

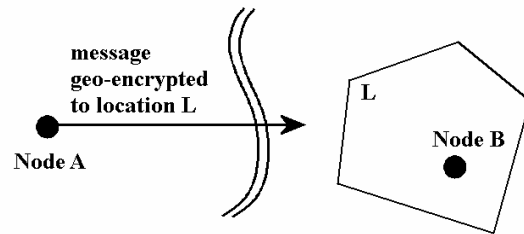


Figure 1: geo-encryption forces the receiving node (B) to be inside a pre-defined location L to be able to decrypt the message.

Denning’s model ([1], [2] and [3]) for adding security to transmissions uses location-based encryption to limit the area inside which the intended recipient can decrypt messages. Figure 1 shows the general idea of geo-encryption. A geo-locking function is employed during the encryption process to combine an encryption key with the recipient’s geographic location (L) to produce a “geo-secured” key for transmission alongside an encrypted message; the message can only be decrypted if the geo-secured key can be recovered, which can only be done if the recovering machine is physically positioned at location L. The sender also transmits parameters which define the shape of the area where decryption is permitted (the “decryption zone”), and the time period during which decryption can be accomplished.

Denning’s model is effective when the sender of a message knows the recipient’s location L and the time that the recipient will be there, and can be applied especially effectively in situations where the recipient remains stationary in a well-known location. Denning recognizes that geo-encryption is also desirable in situations where the recipient is mobile, without a pre-planned itinerary. All the sender needs is the recipient’s current/upcoming location and the time that the recipient will be there, and Denning notes that a velocity parameter (the recipient’s velocity) can be added to the geo-locking function. However, [1-3] did not address mobility support in Denning’s geo-encryption model, and therefore we propose a model to provide for mobility when using GPS-based encryption.

III. THE PROPOSED MODEL

The proposed model (described more fully in [5]) is based on the geo-encryption scheme in [1] in which both the sender and the receiver are mobile, without preplanned itineraries, and can securely deliver their current locations to one another whenever necessary. In order to do this, each mobile node that will be receiving geo-encrypted messages needs to inform potential sender nodes about its intended movement in order for a sender node to estimate the mobile node's expected location at any point in time. This is done by sending information regarding the mobile node's movement, which we call mobility parameters, to the sender via a sequence of message exchanges. The details of the message exchanges are described more fully in [5].

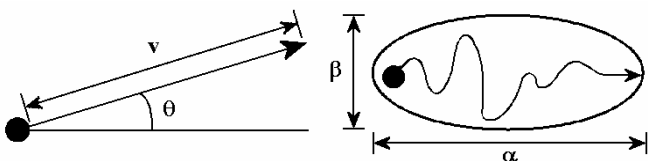


Figure 2: Diagram illustrating the four mobility parameters: velocity, direction, speed maneuverability, and breadth maneuverability.

Let A be a mobile station (the moving *agent*) and let B be a stationary (*base*) station in a network using Denning-style geo-locking for an added layer of security. In our model, the geo-locking function takes shape, time, velocity, direction, and two maneuverability parameters. The shape parameters define an ellipse as the decryption zone. An ellipse is suitable for the shape of our decryption zone because it has a length and breadth, *and* when both are equal, the ellipse becomes a circle that provides uniform coverage in all directions. (A rectangle also has a length and breadth, but when both are equal, it forms a square, with non-uniform coverage.) The time parameter specifies the period during which decryption is possible. When A is in motion, B will need to calculate a time parameter that represents a *future* time when A will actually be in the decryption zone when a geo-encrypted message arrives for decipherment at A.

Figure 2 shows the four mobility parameters that a mobile node uses to advertise its movement information. The velocity parameter, v , describes the recipient's speed. This is the average speed at which the recipient is expected to travel. Velocity (v) is determined from observing the distance traveled during a specified time unit. In [5], we describe a mechanism for a mobile node to measure its velocity using the GPS. The direction parameter, θ , describes the direction in which the recipient is traveling and is measured as the positive angle between the positive side of the Equator circle and the velocity vector on a geographic coordinate system.

The first maneuverability parameter, α , is an indication of how frequently the moving recipient might need to change speeds while traveling to the new destination (how much

leeway, in terms of speed changes, that should be built into the size of the decryption zone). This speed-maneuverability parameter influences the length of the ellipse-shaped decryption zone. For travel on a commercial airliner at a fixed speed, this factor would be small, while for travel on a highway, where unexpected delays might crop up, it would probably be larger.

The second maneuverability parameter, β , defines how much the moving recipient might deviate from a straight line while traveling to a new destination (how much "wiggle room" is deemed necessary, based on an assessment of the terrain being followed). This parameter affects the breadth of the ellipse-shaped decryption zone, and we refer to it as the breadth-maneuverability parameter. On a straight highway, or in an airplane, the breadth maneuverability factor will probably be small. But in mountainous terrain, or while traveling on horseback or on a winding river, the breadth maneuverability factor will be larger. It may make more sense in the real world for a mobile station to define large maneuverability values for a single, long, winding and unpredictable move, rather than frequently computing a series of small directional and speed changes while following a crooked trail.

The proposed model attempts to achieve the following goals: 1) capture the locations of other mobile stations within a given decryption region with high probability, 2) keep stations locations secret from rivals, and 3) permit the stations to be as mobile and maneuverable as possible. These three goals are in conflict and thus we need to make certain compromises in the design. In our simulations, we show the relationships between these factors and evaluate the performance and scalability.

IV. PROTOCOL OVERVIEW

Our protocol builds on top of existing wireless multi-hop routing protocols, thus we will not address the routing issues of mobile multi-hop networks. We will use DSR [6] in our simulations. Our protocol will handle the communication of movement information between mobile nodes and the updating of this information whenever nodes move unexpectedly. It aims to allow mobile nodes to communicate their movement information accurately while at the same time reducing the overhead on the network.

When a mobile node, say MB1, wishes to communicate with another node MB2, it broadcasts a message to discover a route to MB2. When such a route exists, node MB1 will receive a route reply specifying the sequence of hops to reach node MB2. In our model, the source node will also need to know the position of the destination node.

This is meant to simulate a Geo-encryption model in a mobile network. Node MB1 will need to keep a table of the positions of the nodes it intends to send data to. The table will be kept current with position update messages from these nodes. Each node can obtain and send its coordinates to correspondent nodes using a position update message. The

destination node MB2, upon receipt, will map its true position to the intended location of the message using the *shape parameters* (discussed in [1]). If the result of the mapping matches the location sent by MB1, the decryption will be considered successful else the message will be dropped. We will simplify this model to test if the node is within a given square centered on the received coordinates.

V. SIMULATION MODEL

We decided to make the position update aspect of the protocol be reactive similarly to DSR routing. To simulate geo-encryption using DSR we made the following assumptions:

1. Authentication is present and free: the establishment of communication between two nodes is assumed authenticated and anytime encryption fails the authenticated relationship is assumed reestablished automatically by the Position Update Message.
2. Decryption of the message is assumed part of another layer. Our simulations are only concerned with position updates; if the message contains the right position parameters x and y to a certain tolerance \pm Tolerance, the message is considered decrypted.
3. Flow state in DSR was disabled to allow every message to carry its source route.
4. DSR control messages like route discovery messages are not subject to encryption and are not tested for decryption.

When a message is received by its destination node, if none of its protocol flags are set (RouteRequest, RouteReply, PositionUpdate), the node gets its own real X and Y values (GPS) and compares them to the x and y values in the packet header. If $x \in [X_{real} \pm \text{Tolerance}]$ and $y \in [Y_{real} \pm \text{Tolerance}]$ the message is considered decrypted otherwise the message is considered not decrypted and the packet is passed to GeoHandler, a function that responds to decryption failures. GeoHandler is also called if the message is decrypted but its coordinates are not within half the tolerance ($x \in [X_{real} \pm \text{Tolerance}/2]$ and $y \in [Y_{real} \pm \text{Tolerance}/2]$) this is meant to preempt future decryption failures.

GeoHandler constructs a message, puts the real X and Y values in the x and y fields, sets the Position Update Flag and sends the message on the reverse route that the received message arrived on.

When a message is received and it has its Position Update Flag set, it is not tested for decryption but is used to update the table entries corresponding to the source node that sent the message. This makes our Geo encryption protocol totally on demand and should only have overhead when decryption failures occur.

To evaluate the protocol we added lines to the trace file indicating the following events:

- A message was successfully decrypted.

- A message failed decryption. These would give a metrics of the protocol performance.
- A Position Update Message was sent.
- A Position Update Message was received. These would allow us to gauge the overhead of the protocol on regular DSR.

We used a subset of bus routes from the Seattle area [7] as our mobility file. First we plotted the data for one of the files at our disposal and, after proper unit conversion, determined a $1500 \times 1500 \text{ m}$ area that presented dense traffic. We then selected the movements of buses within that area during a 15 minute period (simulation target time). Furthermore we excluded from that selection the buses that came too close to the edge of our area to avoid sudden node disappearances during the simulation. And finally, from the remaining data, we selected the 50 buses with the most number of updates in the given period (see Figure 3). From that we created an initial position file and movements file with *TCL* commands to include in our simulation file. We also included different pause times between movement updates to create several movement files with decreased mobility; the pause times are 10, 25, 50, 75, 100, 200, 400, 650 and 900 seconds. Movements updates that exceeded 900 seconds were purged from the files, thus in effect the 900 second pause time corresponds to zero mobility when all the nodes stay at their original positions.

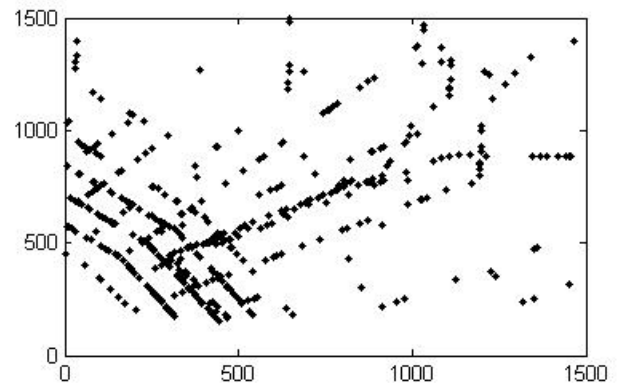


Figure 3: Final simulation data from a subset of bus routes in a $1500 \times 1500 \text{ m}$ area.

We used *ns-2.28* for our simulations, running DSR with flow state disabled and the modifications described above.

For each one of the mobility files we performed three runs with 10 sender 10 receivers, 20 sender 20 receivers and 30 sender 30 receivers. Every sender sent data at a Constant Bit Rate (CBR) of 4 packets per seconds with a packet size of 256 bytes.

For each run we recorded the decryption ratio and the protocol overhead. We measured our decryption ratio as the ratio of successfully decrypted messages amongst those that were received. Thus the ratio does not reflect the delivery ratio of DSR.

We measure the protocol overhead for position updates as the ratio of position update messages to the total number of data messages (CBR, decrypted or that were not received). A better measure would have been the ratio of generated position update messages to the total number of messages on the network including DSR messages as this is a modification to DSR.

VI. SIMULATION RESULTS

With the tolerance fixed to 10 m and running for 10 , 20 and 30 senders and receivers respectively, we note that the general trend is, as expected, that the decryption ratio falls with an increase in mobility (i.e. the decryption ratio increases with bigger pause times). See Figure 4. This is due to the fact that higher mobility means that nodes move more often away from their perceived positions at the sending nodes. As a result more messages are not decrypted.

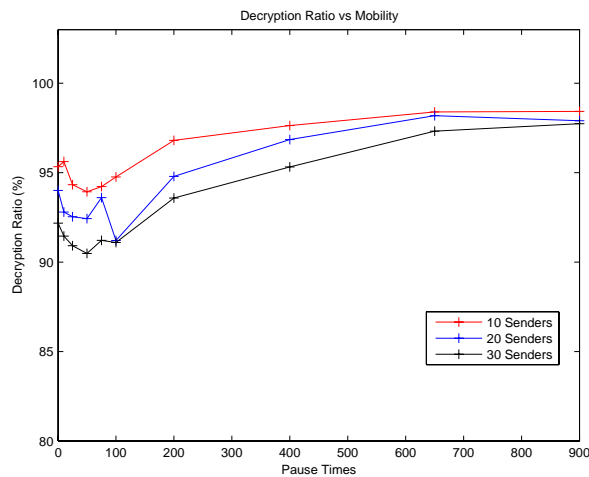


Figure 4: Decryption ratio vs. mobility for 10, 20, and 30 Constant Bit Rate (CBR) sources.

Also from Figure 4, we can see that an increase in CBR senders and receivers results in a decrease in decryption. That can be explained by the increased delay in message delivery due to increased network congestion. If a message is buffered often along the way, it allows time to the destination node to move further away from its current position and thus increase the chance of a decryption failure. This is confirmed by the fact that at low mobility, the gap between the ratios of the three cases is reduced. A noticeable feature of the graph is also the fast dip for pause times between 0 and 100 seconds which might be a result of using DSR as our routing protocol.

On the other hand, overhead decreases with increased pause times. See Figure 5. This behavior is typical of a protocol that is reactive to movement. If there is no movement then there is no need for movement updates. The overhead does not quite go to zero at zero mobility because in our simulation, nodes do not initially know the positions of their destinations but have to learn them by sending the default coordinates $(-1, -1)$

and causing a position update message.

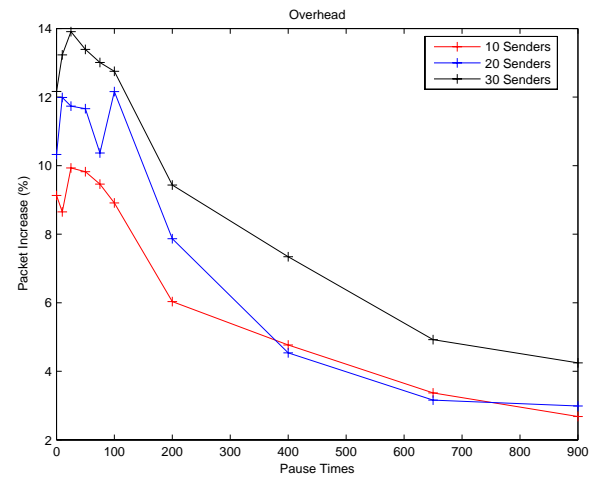


Figure 5: Protocol overhead vs. mobility for 10, 20, and 30 CBR sources

We can also see that in general, overhead increases with an increase in CBR senders and receivers. This must follow from the drop in decryption ratio discussed previously as every failed decryption or near failed (not within half tolerance) generates an update message and thus increases the overhead of the protocol. By the same token, the initial increase in overhead for small pause times compared to 0 is a result of the equivalent behavior of the decryption ratio in Figure 4.

We do not consider the drop of the overhead of the 20 CBR below that of 30 CBR, to be significant. We believe it to be an artifact that would disappear if we ran more simulations.

For our second series of simulations, we used our base case of 10 senders and 10 receivers to test the effect of changing the tolerance. First we compared the decryption ratio of our reference tolerance of 10 meters to that of a more restrictive tolerance of 3 meters. The results over the usual pause times going from 0 to 900 seconds can be seen in Figure 6.

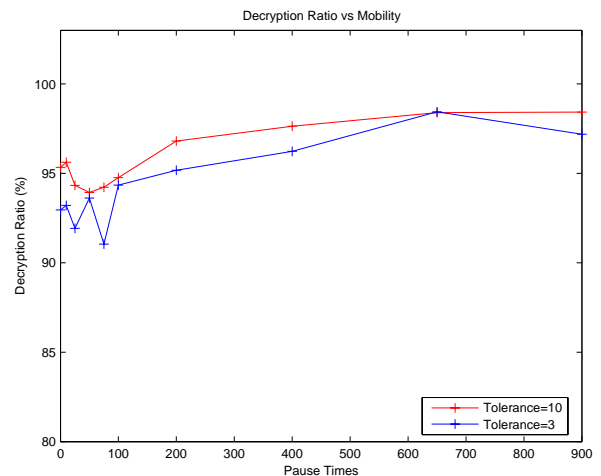


Figure 6: Decryption ratio vs. mobility for a tolerance of 3 and 10.

The value of the tolerance sets the size of the decryption square. Thus a more restrictive tolerance will result in a smaller decryption zone and lower decryption ratios as seen in the figure.

This trend can be further seen in Figure 7 where we show the results of increasing the tolerance through 5, 10, 20, 50, 70 and 100 meters for the maximum mobility case and with 10 CBR sources. As expected, decryption increases with more slack tolerances, that ratio would not reach 100% though due to the need to establish the knowledge of the initial positions.

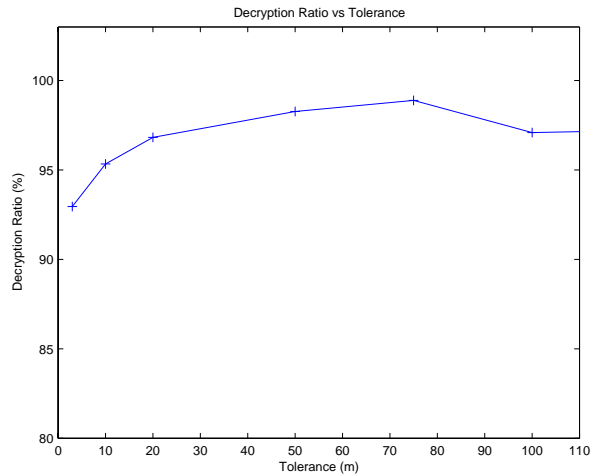


Figure 7: Decryption ratio vs. tolerance for 10 CBR sources

In a similar behavior, the overhead drops rapidly with increased tolerance down to the minimum required for the initial position messages and remains constant after a certain point. That behavior is mostly dependent on the range of motion of the buses during the simulation. If it does not exceed half of the given tolerance then no movement updates are required for the life of the simulation.

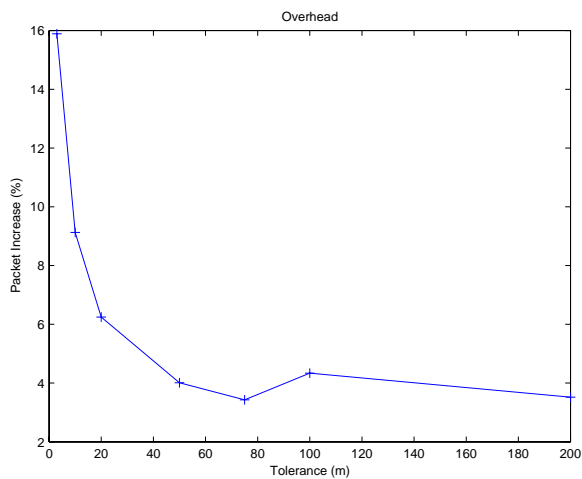


Figure 8: Protocol overhead vs. Tolerance

The exponential shape can be explained by the possibility that a lot of buses have a limited range of motion and the slightest increase in tolerance is sufficient to cover their entire

or most of their path and make updates unnecessary while a few buses have larger ranges of motion and will be affected only by a large increase in the tolerance.

VII. CONCLUSION AND FUTURE WORK

Using geo-encryption adds a significant layer of security to network transmissions. Mobile networks should be able to take advantage of this technique. We believe that our model serves a source for further work on location-based security for mobile networks. In our proposal, mobile nodes which stray from their advertised locations can reestablish a secure status within the network at the same time do not sacrifice the secrecy of their locations. We evaluated a simplified version of the Geo-encryption protocol by measuring the induced overhead to the network and its decryption performance by simulating a modified DSR protocol using *ns-2* under selected scenarios.

Our results proved some of our expectations about decryption decline with increase in mobility and an equivalent increase in overhead. We also saw some results we did not predict such as the decrease in decryption ratio with an increase of network traffic due to increased message queuing delay. And furthermore we need to make further simulations with different mobility files and under different scenarios to get a better idea of the protocol behavior, confirm our preliminary results and further clarify and explain some aspects of the above plots.

Finally we can point certain steps to improve the performance of the protocol as next position prediction at the sender or the receiver based on history of movement, or the sending of movement parameters such as speed and direction by the receiver to the sender. And for improved security we can extend our protocol to a multi-hop encryption scheme that would require the sender to have knowledge of the position of all the forwarding nodes.

REFERENCES

- [1] D. E. Denning, "Geo-encryption," June, 2004, unpublished.
- [2] D. Denning and P. MacDoran, "Location-based authentication: grounding cyberspace for better security," in *Computer Fraud and Security*, np.: Elsevier Science Ltd, 1996.
- [3] L. Scott and D. Denning, "Geo-encryption: using GPS to enhance data security." *GPS World*, 1 Apr. 2003.
- [4] Trimble Information Services, "Powering the transformation of location data into location information," [Online document] 2002.
- [5] Ala Al-Fuqaha, Omar Al Ibrahim, Joe Baird, "A Mobility Model for GPS-Based Encryption", *IEEE GlobeCom* 2005.
- [6] David B. Johnson and David A. Maltz, "Dynamic source routing in ad hoc wireless networks", In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181. Kluwer Academic Publishers, 1996.
- [7] Jorjeta G. Jetcheva, Yih-Chun Hu, Santashil PalChaudhuri, Amit Kumar Saha, and David B. Johnson. "Design and Evaluation of a Metropolitan Area Multitier Wireless Ad Hoc Network Architecture", *Proceedings of the 5th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2003)*, pp. 32-43, IEEE, Monterey, CA, October 2003