Università della Calabria

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Corso di Laurea Magistrale in

INGEGNERIA INFORMATICA

Relazione Progetto

di Network Security

Docente

Prof. Floriano De Rango

Studenti

Doriana Lucente 224488
Keivan Mahmoudi Rahmani 224510
Dylan Marino 224486

Anno Accademico 2021 / 2022

Indice

Introduzione

Attualmente, le reti wireless sono cresciute in modo significativo nel campo delle reti di telecomunicazioni. Le reti wireless hanno la caratteristica principale di fornire l'accesso alle informazioni senza considerare gli attributi geografici e topologici di un utente. La più utilizzata e diffusa ad oggi è la rete wireless costruita su una rete cablata. L'infrastruttura principale richiede una complessa gestione di tutto il processo e presenta dei limiti se l'infrastruttura di comunicazione non è disponibile.

Un esempio è nelle aree disastrate o per operazioni militari dove non è possibile costruire infrastrutture in modo istantaneo. Questo problema è stato risolto sviluppando un meccanismo di rete wireless ad hoc, che è noto come mobile ad hoc networks (MANET). Una MANET è una rete decentralizzata, auto-organizzata e priva di infrastrutture. Ogni nodo agisce come router per stabilire la comunicazione tra nodi su collegamenti wireless. I nodi inoltrano il pacchetto di comunicazione tra loro per trovare o stabilire il percorso di comunicazione.

Nelle reti MANET, ogni nodo si muove dinamicamente in modo arbitrario e ogni nodo può unirsi o uscire facilmente la rete, questo crea di per sé una topologia imprevedibile e sempre dinamica, che cambia nel tempo. Ci sono due questioni nate dalla natura delle MANET, cioè le prestazioni e la sicurezza. Come già detto prima, questo tipo di rete ha specifiche e caratteristiche diverse da altri tipi di reti, e questa diversità comporta ad avere dei problemi di sicurezza specifici.

Molti attacchi possono essere eseguiti in ogni layer di comunicazione. Le MANET sono più soggette a minacce fisiche rispetto alle reti cablate ed è possibile eseguire un numero ingente di attacchi, come ad esempio spoofing, intercettazioni e attacchi denial of service (DoS). La maggior parte di questi attacchi sono indirizzati al tipo di schemi del protocollo di instradamento e manomettono alcune delle loro procedure di funzionamento, sfruttando la loro implementazione e architettura insicura. Un tipico attacco man in the middle se viene applicato in una rete MANET prende il nome di BlackHole attack.

Sulla base di queste considerazioni, è opportuno chiedersi come sviluppare un protocollo di routing robusto e che tenga conto della sicurezza, che eliminerà gli attacchi esistenti nelle MANET senza rinunciare alle prestazioni complessive.

Il meccanismo di sicurezza nel protocollo di routing MANET è diviso in due categorie in base a metodo di sicurezza, ovvero meccanismo crittografico e meccanismo basato sulla fiducia (trust-based).

Vediamo velocemente in cosa consiste il meccanismo crittografico: consiste nel proteggere lo scambio di dati a pacchetto, il rilevamento del percorso e il processo di manutenzione del percorso durante il processo di comunicazione. Sono stati applicati molti tipi di algoritmi di crittografia per mettere al sicuro il pacchetto quando viene inviato. Il secondo è il meccanismo di fiducia, che stabilisce una relazione di fiducia tra i nodi prima di eseguire il processo di comunicazione ed iniziare lo scambio. Un protocollo di instradamento sicuro che fa uso del meccanismo basato sulla fiducia ha prestazioni migliori rispetto all'utilizzo del meccanismo crittografico. [1]

Nello specifico il progetto sviluppato si concentra su un particolare tipo di MANET, ovvero le VANET (Vehicular Ad Hoc Network). Le Vehicular Ad-Hoc Network o VANET sono una tecnologia che, attraverso l'utilizzo delle automobili, permette di creare una rete mobile. La caratteristica fondamentale, che le contraddistingue dalle reti mobili classiche, è l'assenza di alcuna infrastruttura predisposta: infatti è diffcile pensare che tutte le strade possano essere coperte da punti di accesso ssi per permettere la comunicazione tra i veicoli. Un tale approccio architetturale, simile a quello dei sistemi cellulari, presenterebbe sia problemi di fattibilità che di

sostenibilità economica. Per questi motivi è più sensato immaginare che i veicoli possano da soli costituire l'infrastruttura di comunicazione. [2]

VANET

Caratteristiche principali delle VANET

- *Topologia dinamica*: La velocità e la direzione dei veicoli cambiano costantemente all'interno della rete, questa caratteristica si rispecchia sulla topologia dei nodi sulla rete.

- *Connettività intermittente*: La connettività tra i dispositivi cambia molto frequentemente come la connessione tra due dispositivi lo scambio di informazioni può cessare in qualsiasi momento. Il motivo dietro le frequenti disconnessioni è dato dalla topologia molto dinamica.

- *Sensori a bordo*: supponiamo che i nodi siano dotati di sensori a bordo che sono in grado di trasmettere informazioni ad altri dispositivi o nodi.

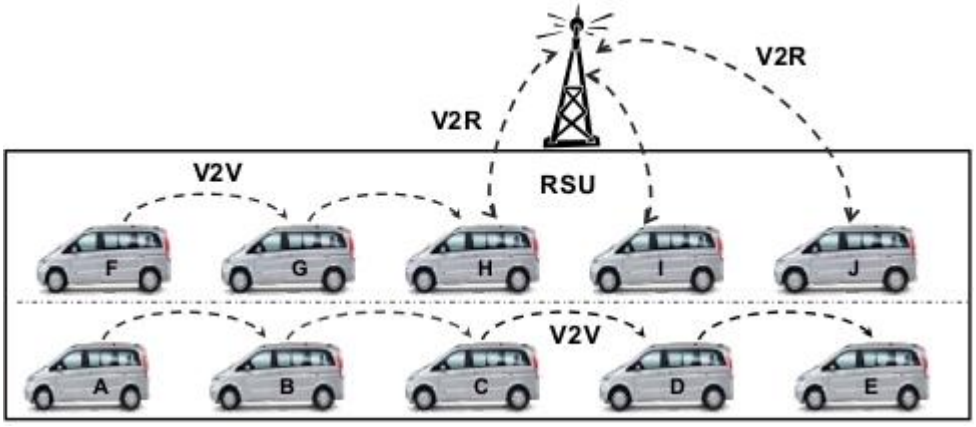
Tipologie di comunicazione

I nodi presenti nelle reti VANET possono comunicare in modi differenti, ne vediamo alcuni:

- *Comunicazione da veicolo a veicolo* (V2V): Lo scambio di dati avviene tra i diversi veicoli in modo da assistere il conducente informandolo reciprocamente con avvisi e altre informazioni importanti, come ad esempio la presenza di un'interruzione o di un incidente.

- *Comunicazione da veicolo a infrastruttura stradale* (V2I): questa comunicazione avviene tra veicoli mobili e infrastrutture fisse lungo la strada per la raccolta dei dati. Fornisce aggiornamenti relativi al rilevamento e monitoraggio ambientale come l'aggiornamento del traffico in tempo reale o l'aggiornamento del tempo.

- *Comunicazione da veicolo a banda larga cloud* (V2B): consente la comunicazione dei veicoli connessioni a banda larga come 3G/4G. Ciò migliora l'assistenza alla guida e il monitoraggio del veicolo come il cloud a banda larga, che può contenere più informazioni sul traffico e altri dati.



Protocolli di comunicazione

Lo scambio di dati tra i nodi in una VANET avviene tramite protocolli di instradamento. Questi protocolli definiscono come un pacchetto di dati verrà distribuito tra i diversi nodi. Sulla base dei mittenti e dei destinatari coinvolti, per le VANET vengono definiti due tipi di protocolli di comunicazione:

- *Unicast*: tali protocolli mirano a fornire o trasmettere dati da una fonte a una destinazione tramite un mezzo senza fili. Ci sono due modi per trasmettere i pacchetti; uno è tramite trasmissione multi-hop dove un'informazione viene trasmessa tramite i nodi vicini, facendo degli hop, ovvero delle ritrasmissioni del pacchetto da parte dei nodi intermedi. La seconda è la tecnica carry and forward in cui un pacchetto viene trasportato dal veicolo per tutto il tempo possibile e quindi trasmessa per ridurre la congestione o la ritrasmissione del pacchetto.

- *Broadcast*: i protocolli di trasmissione mirano a fornire e comunicare al maggior numero possibile di nodi. In situazioni come blocchi stradali, ingorghi, luoghi ad alta densità di traffico o situazioni di emergenza, i protocolli di trasmissione sono un must. Trasmettono pacchetti di dati a più nodi contemporaneamente. D'altra parte, uno svantaggio è il fatto che i protocolli di trasmissione aumentano anche le possibilità di ritrasmissione dei pacchetti facendo risultare il tutto in un flooding di richieste.

Sicurezza: vulnerabilità e rischi nelle VANET

La sicurezza è uno dei fattori chiave da affrontare nella comunicazione veicolare. Tuttavia, data la natura dinamica dell'ambiente veicolare, è molto difficile tenere sotto controllo la rete per sfuggire a possibili attacchi. Gli attaccanti infatti possono creare problemi agli utenti della rete in diversi modi, sfruttando attacchi molto vari (DOS, Flooding, Fabrication). [3]

Vediamo quali sono i rischi principali delle reti VANET:

- I percorsi non prestabiliti o non sono comunque conosciuti da tutti i nodi della rete, inoltre l'ambiente dinamico rende difficile il controllo della topologia dei nodi.

- Quando una trasmissione è basata su collegamenti wireless, qualsiasi nodo all'interno del raggio di trasmissione messaggio può interagirvi. Quindi c'è la possibilità di attacchi passivi e/o attivi.

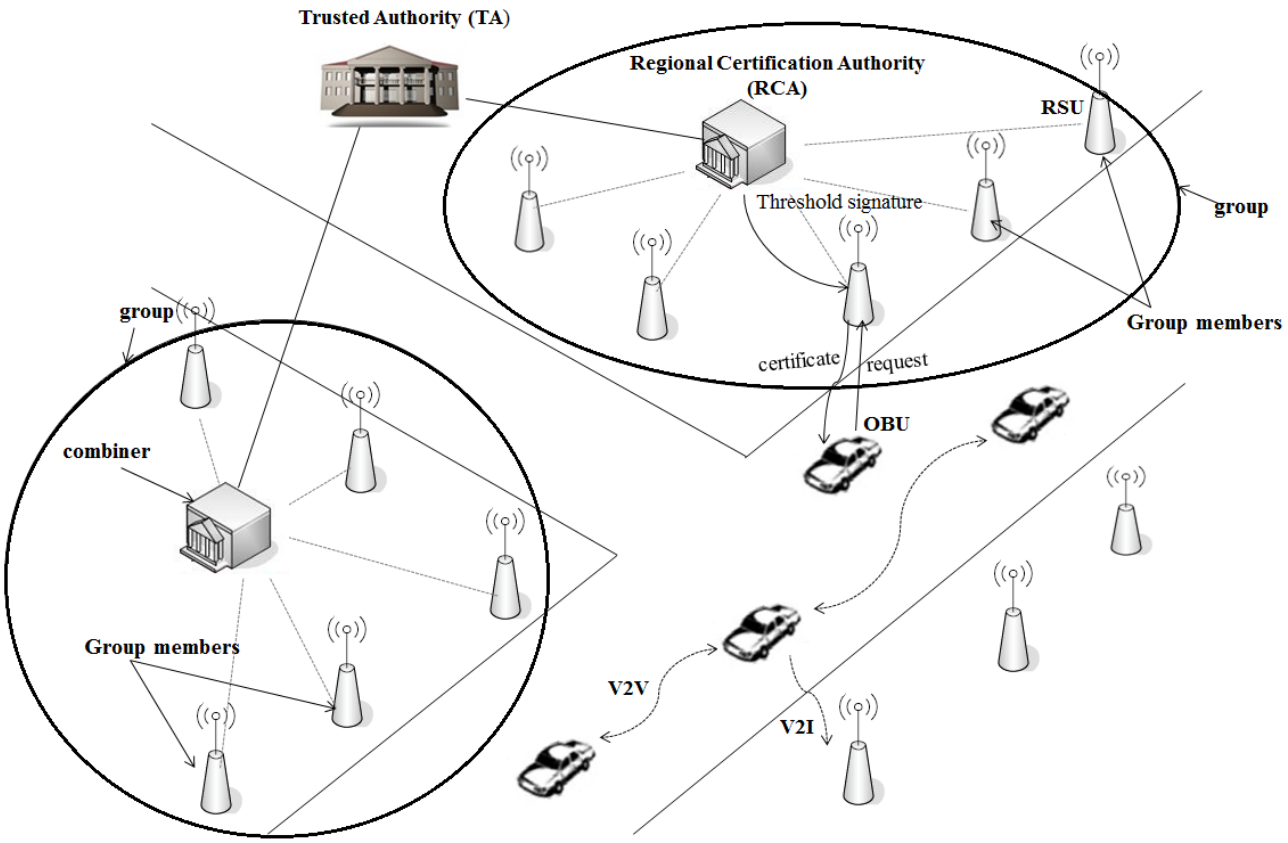
- Differenziare un comportamento dannoso da uno standard non è semplice. Nodi dannosi si trovano a inviare o scartare di proposito pacchetti. Tuttavia, un comportamento simile può avvenire anche a causa di limitazioni di larghezza di banda, nodi in mobilità e collegamenti deboli alla rete wireless. [4]

Possibili attacchi

Sulla base dei rischi appena visti possiamo elencare alcuni degli attacchi che è possibile effettuare all'interno di una VANET:

- Attacchi DoS
- Fabrication attack
- Black Hole Attack
- Flooding attack
- Attacchi all'integrità e alla riservatezza da parte dei nodi di instradamenti intermedi.
- Violazione della privacy mediante intercettazione, localizzazione e tracciamento dei veicoli.

Una possibile soluzione consiste nell'utilizzo di Certificate Authority (CA) a livello regionale che firmano i certificati dei veicoli. Ad ogni nodo presente viene associata una coppia di chiave pubblica e privata ed un certificato firmato dalla CA.



Requisiti di sicurezza nelle VANET

Vi sono due principali proprietà di sicurezza che devono essere garantite all'interno di una VANET:

- *Confidenzialità*: Si riferisce a una comunicazione riservata in cui qualsiasi utente non autorizzato non può accedere al messaggio cifrato. Solo gli utenti autorizzati possono decifrare i messaggi.

- *Integrità:* i dati o il messaggio trasmessi devono essere consegnati correttamente alla destinazione senza manomissioni da parte di un attaccante. Nella rete wireless, il concetto di integrità è importante per garantire che i dati o il messaggio siano inviati dall'utente effettivo con la sua firma e che i dati non vengono modificati da utenti non autorizzati.[5]

OMNeT++ e INET

OMNeT++

Il framework di simulazione OMNeT++ (Objective Modular Network Testbed) viene usato per analizzare sistemi con modelli composti da entità che comunicano fra loro scambiandosi messaggi. È un sistema modulare, orientato agli oggetti e basato sul linguaggio C++.

Le applicazioni principali di OMNeT sono: modellazione dei protocolli di rete, modellazione di code di rete, modellazione di sistemi multiprocessore e valutazione delle prestazioni del sistema software.

In OMNeT ritroviamo quattro tipi di file: .ned, .cc, .h, .ini e .msg.

Il file .ned definisce il modello descrivendolo come insieme di moduli, accessi, connessioni e messaggi.

I moduli programmati in C++ sono assemblati in componenti e modelli più grandi utilizzando il linguaggio NED (Network Description). Sono elementi riusabili.

I moduli possono essere semplici oppure composti . I moduli semplici sono definiti nei file .cc e .h. Mentre un esempio di modulo composto è l'intera rete.

I moduli sono collegati tramite le connessioni che vengono usate per scambiare tra loro messaggi. I messaggi definiscono gli eventi. Per accedere ai moduli vengono usati gli accessi come punti di ancoraggio per le connessioni e possono essere in, out o inout.

Nel file .ini vengono descritte le simulazioni specificandone le caratteristiche e I parametri dei moduli.

Altri file usati nella realizzazione sono .msg per descrivere I messaggi.

OMNeT++ è un softwre open source che può essere utilizzato gratuitamente per scopi accademici. Esiste anche una versione commerciale denominata OMNEST.

INET

L'INET framework può essere considerato la libreria open-source di modelli di protocollo standard di OMNeT++. È gestito dallo stesso team di OMNeT++.

Implementa la pila OSI ove per ogni livello sono presenti implementazioni di diversi protocolli.

Molti framework partono da INET e lo estendono in applicazioni specifiche.

AODV Protocol – RFC 3561 [Aodv e Saodv]

L'algoritmo Ad hoc On-Demand Distance Vector (AODV) permette il routing dinamico, autonomo e multihop tra nodi mobili partecipanti che desiderano stabilire e mantenere una rete ad hoc. AODV permette ai nodi mobili di ottenere rapidamente le routes per nuove destinazioni e non richiede ai nodi di mantenere rotte verso destinazioni che non sono in comunicazione attiva. AODV permette ai nodi mobili di rispondere alle interruzioni dei collegamenti e ai cambiamenti nella topologia della rete in modo tempestivo. Il funzionamento di AODV è privo di loop e offre una rapida convergenza quando la topologia della rete ad hoc cambia (tipicamente, quando un nodo si sposta nella rete). Quando i collegamenti si interrompono, AODV fa sì che l'insieme dei nodi interessati venga notificato in modo che siano in grado di invalidare i percorsi che utilizzano il collegamento perso. Una caratteristica distintiva di AODV è il suo uso di un numero di sequenza di destinazione per ogni entry di route. Il numero di sequenza della destinazione viene creato dalla destinazione per essere incluso insieme a qualsiasi informazione di route che invia ai nodi richiedenti. L'uso del numero di sequenza della destinazione assicura l'assenza di loop ed è semplice da programmare. Data la scelta tra due route verso una destinazione, un nodo richiedente è tenuto a selezionare quello con il numero di sequenza maggiore.

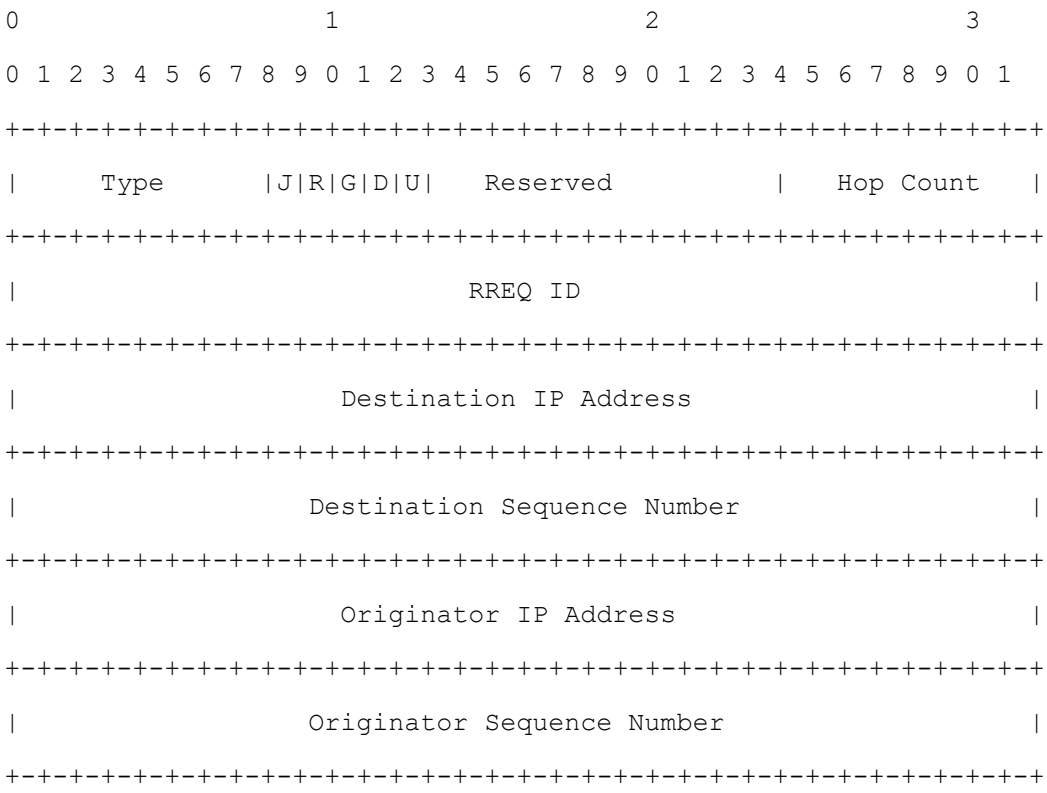
Vi sono tre tipi di messaggi nel protocollo utilizzato:

- RREQ (Route Request)
- RREP (Route Replies)
- RERR (Route Errors)

Il protocollo di routing AODV è progettato per reti mobili ad hoc con reti composte da decine a migliaia di nodi mobili. AODV può gestire tassi di mobilità bassi, moderati e relativamente alti, nonché una varietà di livelli di traffico dati. AODV è progettato per l'uso nelle reti dove i nodi possono fidarsi l'uno dell'altro, sia tramite l'uso di chiavi preconfigurate, oppure perché è noto che non esistono nodi intrusi dannosi. AODV è stato progettato per ridurre la diffusione del traffico di controllo ed eliminare l'overhead sui dati di traffico, al fine di migliorare scalabilità e prestazioni.

Vediamo nello specifico i tipi di messaggi ed il loro formato:

RREQ – Route Request

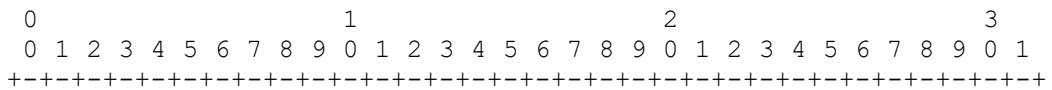


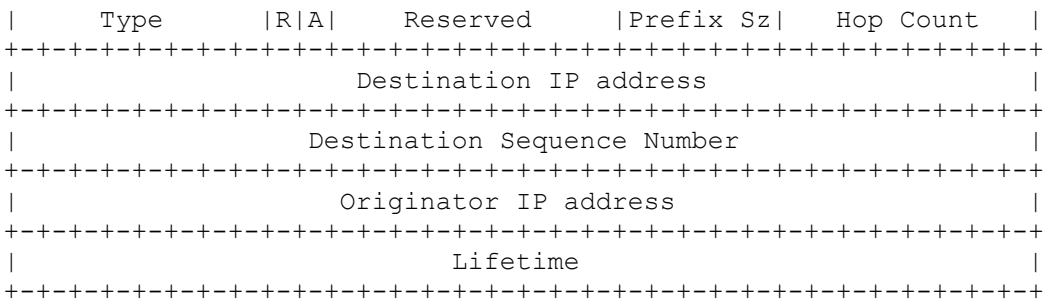
Un nodo invia un RREQ quando ha bisogno di un percorso per una destinazione e non ne ha altri disponibili. Questo può succedere se la destinazione è sconosciuta al nodo, o se in precedenza il percorso valido per la destinazione scade o è contrassegnato come non valido. il

Il campo Destination Sequence Number nel messaggio RREQ è l'ultimo noto per questa destinazione e viene copiato nell'omonimo campo nella tabella di instradamento. Se nessun numero di sequenza è noto, la flag del numero di sequenza sconosciuto viene settata. Il numero di sequenza dell'originator nel messaggio RREQ è il numero di sequenza del nodo, che viene incrementato prima dell'inserimento in un RREQ. Il campo ID RREQ viene incrementato di uno dall'ultimo ID RREQ utilizzato dal nod. Ogni nodo mantiene un solo RREQ ID.

RREP – Route Reply

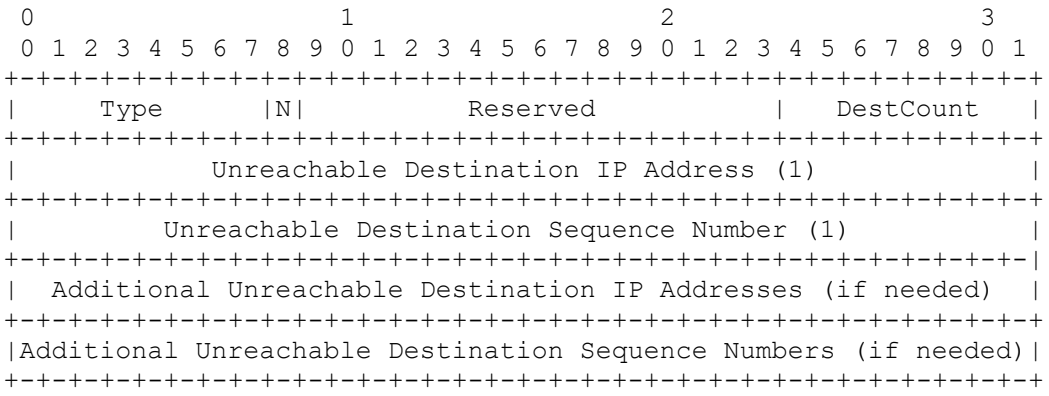
In ogni percorso il pacchetto REPLY contiene il numero di sequenza di destinazione, il
gli indirizzi IP di origine e di destinazione l'hop count e le flag di controllo. Ogni nodo intermedio
che riceve il pacchetto ROUTE REPLY, incrementa l'hop-count , ed effettua un forwarding dello stesso.





RERR – Route Error

Il messaggio RERR viene inviato ogni volta che uno o più nodi destinazione diventano irraggiungibili dai propri vicini a causa di un errore di rete.



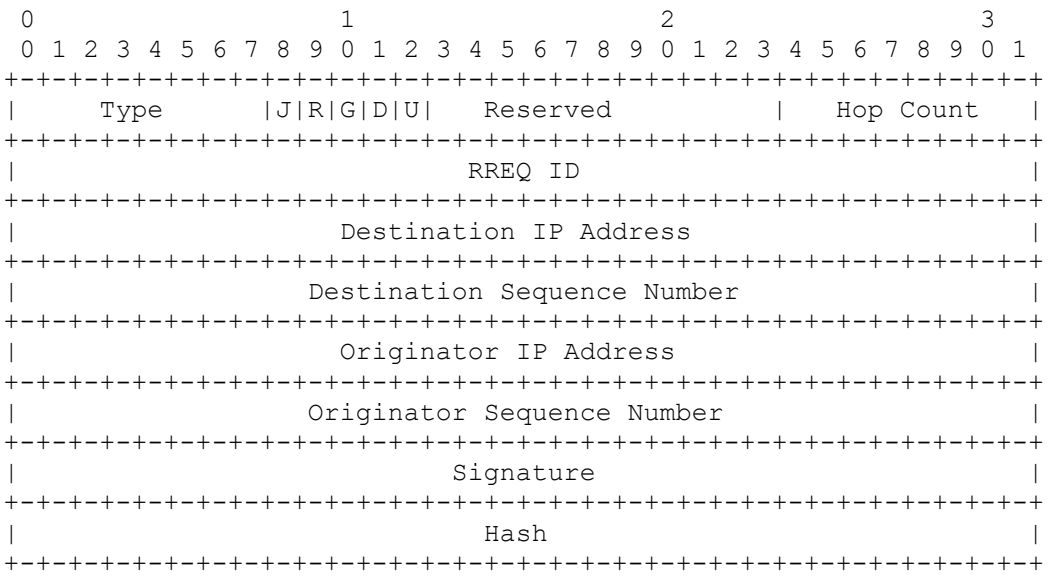
Secure Aodv

I protocolli MANET sono stati progettati senza pensare alla sicurezza, per questo protocolli come aodv necessitano dell'implementazione di un layer di sicurezza se ci si trova in una rete non fidata per evitare di incorrere in rischi. [6]

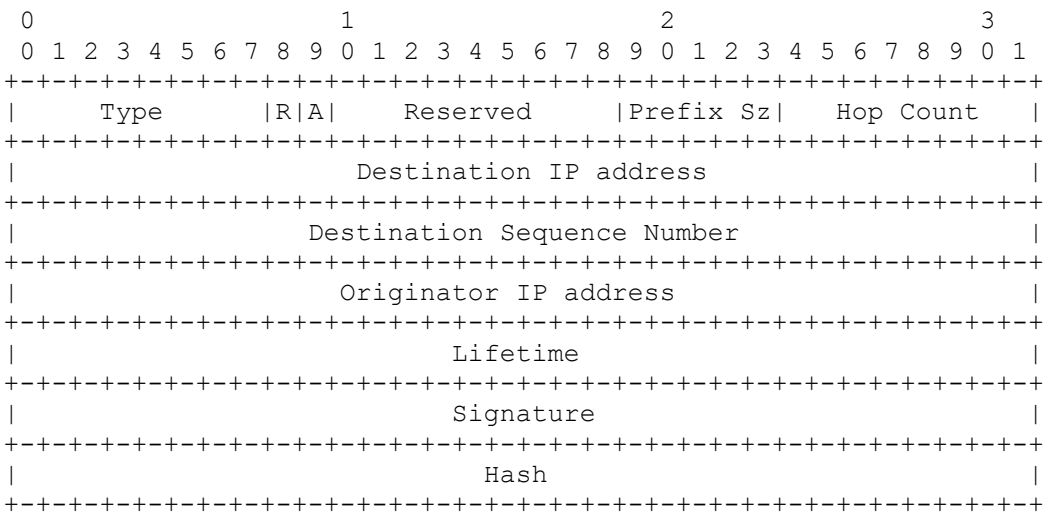
Pertanto, all'interno del progetto si è cercato di realizzare un'estensione del protocollo aodv che permettesse di rispettare i principi di integrità ed autenticazione. L'estensione proposta consiste nella modifica dei pacchetti di RREQ e di RREP e nel metodo di gestione della ricezione degli stessi e si basa sull'assunzione che ogni nodo riesca ad accedere alla chiave pubblica di tutti gli altri nodi. Il nodo mittente andrà a firmare il pacchetto tramite RSA ed inserirà anche l'hash relativo al pacchetto, mentre tutti i nodi che lo riceveranno andranno a verificare l'hash e la firma tramite la relativa chiave pubblica.[7]

I pacchetti di RREQ e di RREP saranno quindi modificati andando ad inserire al loro interno la firma e l'hash del nodo mittente in modo che gli altri nodi possano verificarne la bontà.

Illustriamo quindi di seguito la nuova struttura dei pacchetti di RREQ:

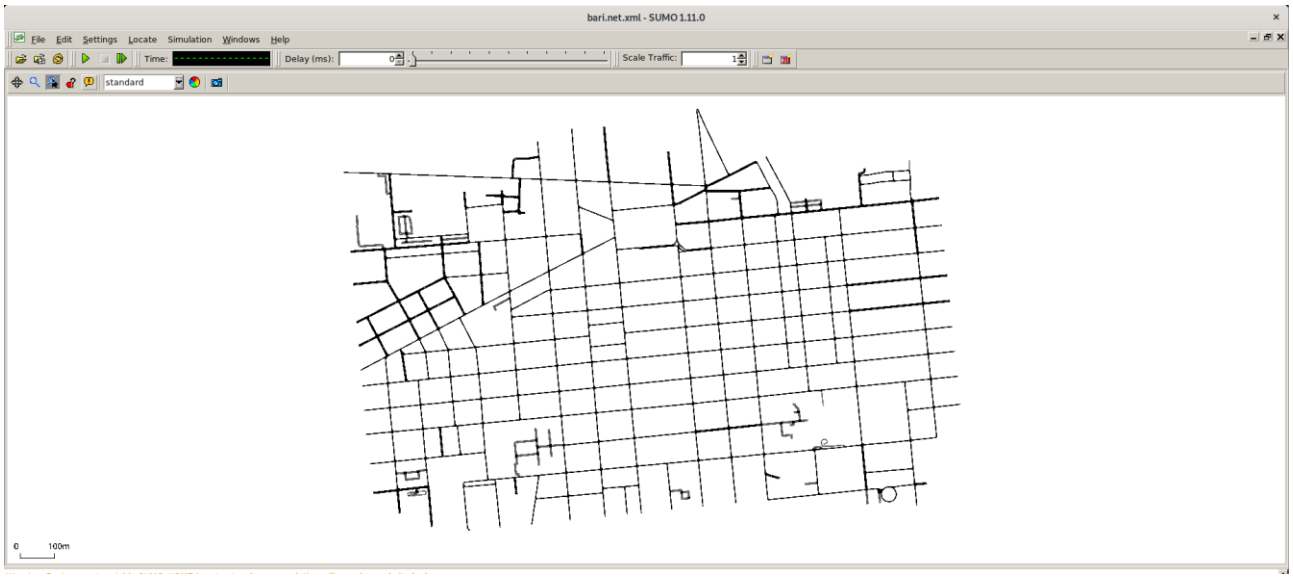


e di RREP:


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Type
R
A
Reserved
Prefix Sz
Hop Count
Destination IP address
Destination Sequence Number
Originator IP address
Lifetime
Signature
Hash

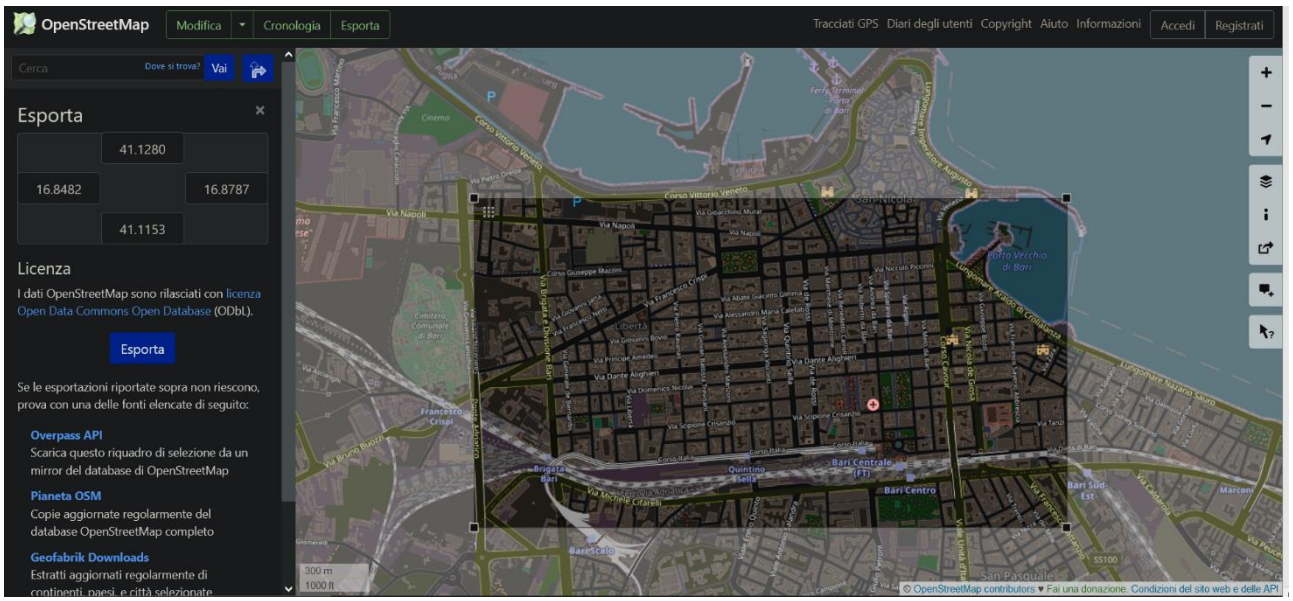
Generazione Mappa

La scelta della mappa da utilizzare per le varie simulazione è ricaduta sulla città di Bari per via delle strade molto ordinate. Tale mappa è stata poi popolata con 25 veicoli distribuiti in modo casuale.

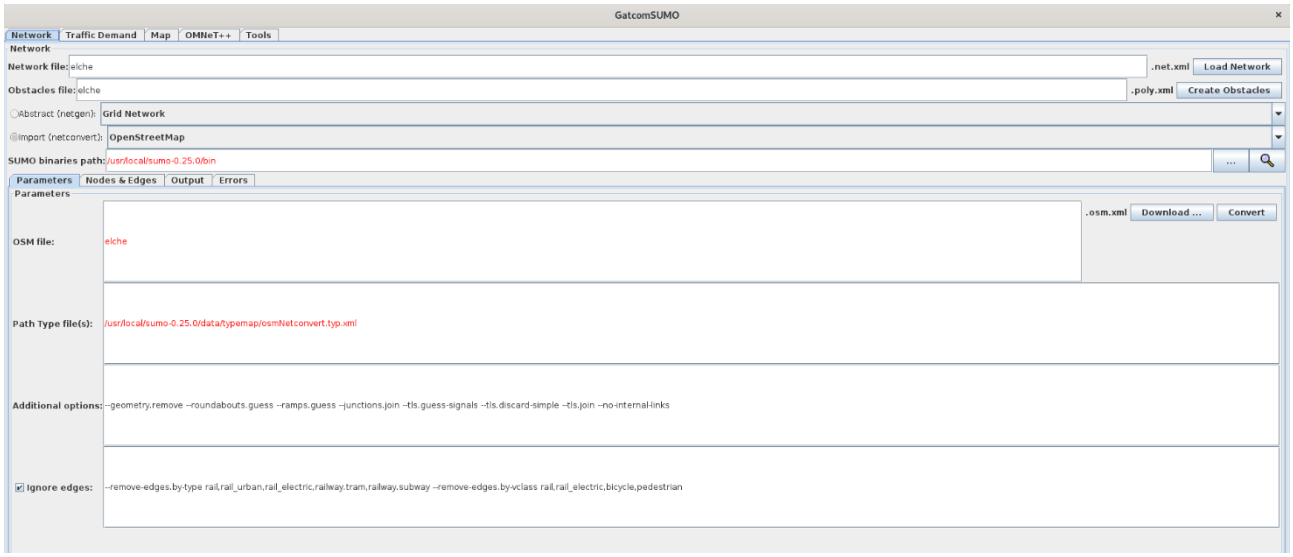


Per la realizzazione della mappa è stato sfruttato:

- il sito OpenStreetMap per prelevare le coordinate dell'area di interesse come mostrato di seguito:



- Il tool GatcomSUMO per la generazione di tutti i file necessari per utilizzare una mappa all'interno di una simulazione su Omnet++



In particolare, all'interno del tool è anche possibile definire quanti veicoli inserire nella simulazione e successivamente creare in modo random i percorsi che dovranno seguire all'interno della simulazione.

Implementazione SecureAodv

La realizzazione del layer di sicurezza è stata realizzata a partire dalla classe AodvBase del modulo inet aggiungendo funzionalità grazie alla libreria Crypto++. In particolare, le modifiche inserite sono state le seguenti:

- Un metodo *GenKeyPair* per la realizzazione della chiave pubblica e chiave privata:

```
void GenKeyPair(std::string my_ip)

{
    std::string myIpBase64;
    CryptoPP::StringSource ss(my_ip, true,new CryptoPP::Base64Encoder(new Cryp-
toPP::StringSink(myIpBase64)));

    AutoSeededRandomPool rng;
    InvertibleRSAFunction privkey;
    privkey.Initialize(rng, 1024);

    std::string filename = "/home/veins/workspace.omnetpp/NetSecPro-
ject/src/keys/privkey_"+ myIpBase64 +".txt";
    Base64Encoder privkeysink(new FileSink(filename.c_str()));
    privkey.DEREncode(privkeysink);
    privkeysink.MessageEnd();
    RSAFunction pubkey(privkey);



    filename="/home/veins/workspace.omnetpp/NetSecProject/src/keys/pub-
key_"+ myIpBase64 +".txt";
    Base64Encoder pubkeysink(new FileSink(filename.c_str()));
    pubkey.DEREncode(pubkeysink);
    pubkeysink.MessageEnd();

}
```

Poiché abbiamo l'esigenza di creare una coppia di chiavi per ogni nodo, è stato scelto di chiamare i file con la codifica base64 dell'IP del nodo in modo da potergli accedere con semplicità.
Questo metodo viene chiamato al termine della funzione *initialize* in modo da poter avere da subito disponibile la coppia di chiavi.

```
void SecureAodv::initialize(int stage)
{
    ...

    GenKeyPair(getSelfIPAddress().str());
}
```

- Un metodo *verify* per la verifica di un messaggio tramite la sua chiave pubblica

```
bool verify(int length, std::string message, std::string signature-
Base64, std::string originator_ip) {
    std::string originatorIpBase64;
    CryptoPP::StringSource st(originator_ip, true,new CryptoPP::Base64En-
coder(new CryptoPP::StringSink(originatorIpBase64)));

    std::string filename = "/home/veins/workspace.omnetpp/NetSecPro-
ject/src/keys/pubkey_"+ originatorIpBase64 +".txt";
    ifstream f(filename.c_str());
    if(f.fail()) return false;
    std::string signature;
    CryptoPP::StringSource ss(signatureBase64, true,
            new CryptoPP::Base64Decoder(new CryptoPP::StringSink(signature)));

    CryptoPP::ByteQueue bytes;
    FileSource file2(
            filename.c_str(),
            true, new Base64Decoder);
    file2.TransferTo(bytes);
    bytes.MessageEnd();
    RSA::PublicKey pubKey;
    pubKey.Load(bytes);

    RSASSA_PKCS1v15_SHA_Verifier verifier(pubKey);
    bool result = verifier.VerifyMessage((const byte*) message.c_str(),
            message.length(), (const byte*) signature.c_str(), length);

    return result;
}
```

Che viene chiamato nei metodi *handleRREP* e *handleRREQ* per la verifica della signature.

```
 //Verify packet signature

std::string signature = rrep->getSignature();
bool result = verify(len, messageBase64, signature,rrep->getOriginato-
rAddr().str());


//if not verified insert address into blacklist

if (true == result) {
    cout << "Message Verified" << endl;
} else {
    EV_INFO << "blacklist for node " << rrep->getOriginato-
rAddr().str().c_str() << endl;
```

```
    blacklist.insert({rrep->getOriginatorAddr(),simTime()});
    cout << "Message Verification Failed" << endl;
    return;
```

- Un metodo *sign* per la firma dei pacchetti tramite la chiave privata:

```
std::string sign(std::string message, std::string my_ip) {
    AutoSeededRandomPool rng;
    CryptoPP::ByteQueue bytes;

    std::string myIpBase64;
    CryptoPP::StringSource st(my_ip, true,new CryptoPP::Base64Encoder(new Cryp-
toPP::StringSink(myIpBase64)));

    std::string filename = "/home/veins/workspace.omnetpp/NetSecPro-
ject/src/keys/privkey_"+ myIpBase64 +".txt";
    FileSource file(
            filename.c_str(),
            true, new Base64Decoder);
    file.TransferTo(bytes);
    bytes.MessageEnd();
    RSA::PrivateKey privateKey;
    privateKey.Load(bytes);

    RSASS<PKCS1v15, SHA1>::Signer signer(privateKey);
    byte* signature = new byte[signer.MaxSignatureLength()];
    if ( NULL == signature) {
        return "";
    }

    size_t length = signer.SignMessage(rng, (const byte*) message.c_str(),
            message.length(), signature);
    len = length;
    std::string sig(reinterpret_cast<const char *>(signature), length);
    std::string messageBase64;
    CryptoPP::StringSource ss(sig, true,
            new CryptoPP::Base64Encoder(
                    new CryptoPP::StringSink(messageBase64)));
    return messageBase64;
}
```

Che viene chiamato nei metodi *createRREQ* e *createRREP* per la creazione della signature.

```
//sign packet

std::string signature = sign(messageBase64,getSelfIPAddress().str());
rreqPacket->setSignature(signature.c_str());
```

- La creazione di un hash tramite SHA256 da inserire nei pacchetti nei metodi *createRREQ* e *createRREP*:

```
    //hashing packet

    CryptoPP::HexEncoder encoder(new CryptoPP::FileSink(std::cout));

    std::string digest;

    CryptoPP::SHA256 hash;
    hash.Update((const byte*)message.data(), message.size());
    digest.resize(hash.DigestSize());
    hash.Final((byte*)&digest[0]);

    rreqPacket->setHash(digest.c_str());
```

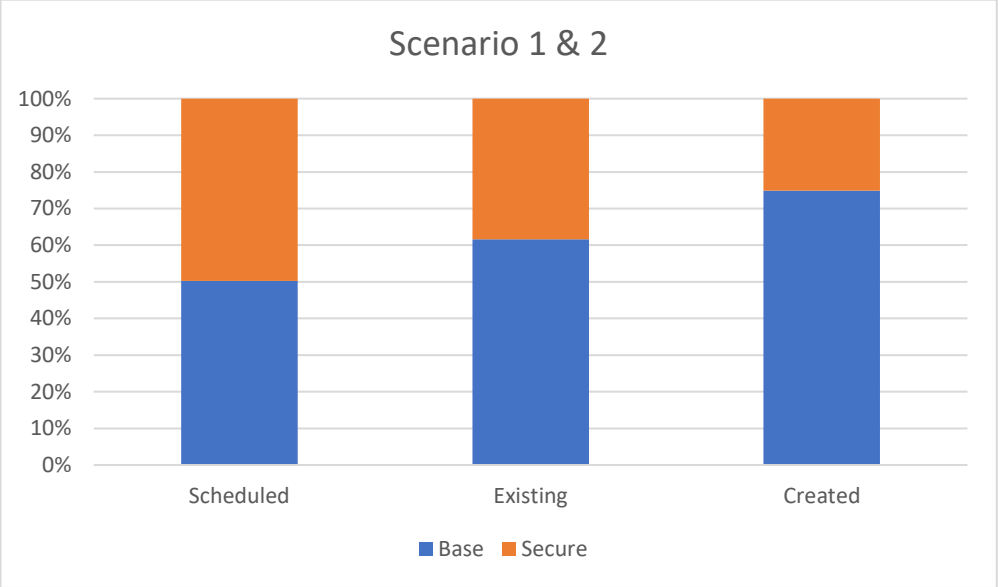
- La verifica dell'hash nei metodi di *handleRREQ* e *handleRREP*:

```
// verify packet hash

    SHA256 hash;
    std::string digest = rrep ->getHash();
    hash.Update((const byte*)message.data(), message.size());
    bool verified = hash.Verify((const byte*)digest.data());

    if (verified == true)
        std::cout << "Verified hash over message" << std::endl;
    else{
        std::cout << "Failed to verify hash over message" << std::endl;
        return;
    }
```

Scenario 1 & 2 – AodvBase e SecureAodv

Gli scenari 1 e 2 prevedono delle simulazioni molto semplici in cui la totalità di nodi è formata da nodi Base o Secure. In particolare, con lo scenario 2 si vuole mostrate come l'applicazione del layer di sicurezza non intacca il funzionamento del protocollo ma solo, in parte, con un lieve calo delle prestazioni. È possibile vedere questo fenomeno facendo un' analisi dei pacchetti:

Scenario 1 & 2		
	Base	Secure
Scheduled	97	96
Existing	3440	2140
Created	45375	15228



Questi scenari, benché molto semplici, risultano particolarmente importanti in quanto fanno da base per la costruzione di tutti gli altri.

Scenario 3 & 4 – Base e Secure con Fabrication

In questi scenari si vuole evidenziare il comportamento del protocollo aodv base e secure in presenza di nodi malevoli.
In particolare, questi scenari riguardano un attacco di tipo Fabrication che mira a modificare la struttura topologica della rete.
Questo attacco consiste nella "fabbricazione" di pacchetti RREP contenti un numero di hop molto alto indipendentemente dal tipo di RREQ: il nodo malevolo alla ricezione della RREQ non andrà ad inoltrare la RREQ ai suoi vicini ma risponderà immediatamente con una RREP indicando che il nodo desiderato si trova ad una distanza molto elevata, andando così a modificare la topologia dei nodi.

Per quanto concerne l'implementazione sono stati modificati quindi i metodi di *handleRREQ* e *createRREP* nel modo seguente:

```
void FabricationRSA::handleRREQ(const Ptr<RreqSec>& rreq, const L3Ad-
dress& sourceAddr, unsigned int timeToLive)
{
    ...

    if (true) { // indiependentemente dall'ip nella rreq
        EV_INFO << "I am the destination node for which the route was re-
quested" << endl;

        // create RREP
        auto rrep = createRREP(rreq, destRoute, reverseRoute, sourceAddr);

        // send to the originator
        sendRREP(rrep, rreq->getOriginatorAddr(), 255);

        return;    // discard RREQ, in this case, we do not forward it.
    }

    ...
}
```

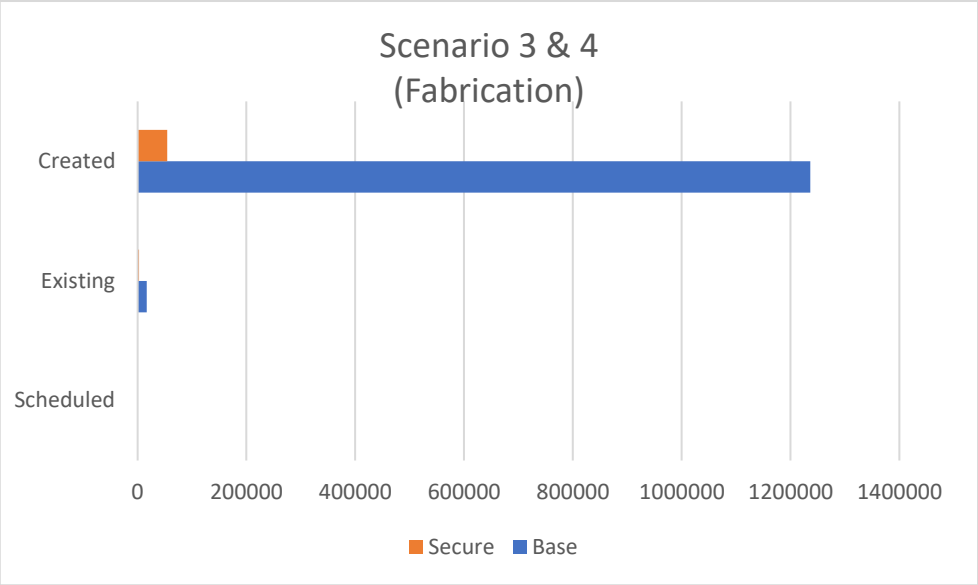
Nella *handleRREQ* indipendentemente dall'IP di destinazione andiamo a creare una RREP ed evitiamo di inoltrare la richiesta.

```
const Ptr<RrepSec> FabricationRSA::createRREP(const Ptr<RreqSec>& rreq, IRoute *de-
stRoute, IRoute *originatorRoute, const L3Address& lastHopAddr)
{
    ...

    if (true) {  //in ogni caso

        if (!rreq->getUnknownSeqNumFlag() && sequenceNum + 1 == rreq->getDest-
SeqNum())
            sequenceNum++;

        rrep->setDestSeqNum(sequenceNum);

        rrep->setHopCount(1000);
.
        rrep->setLifeTime(myRouteTimeout.trunc(SIMTIME_MS));
    }
    ...
}
```

Nella *createRREP* invece andiamo a settare l'HopCount a 1000 invece che a 0.

Passando ai dati possiamo vedere come l'applicazione del layer di sicurezza fa si che i pacchetti dei nodi malevoli vengano droppati comportando una riduzione importante del numero di pacchetti scambiati e permettendo alla rete di funzionare in modo corretto.

Scenario 3 & 4		
	Base	Secure
Scheduled	123	100
Existing	16932	2874
Created	1236446	54618


Scenario 3 & 4
(Fabrication)
Created
Existing
Scheduled
0
200000
400000
600000
800000
1000000
1200000
1400000
Secure
Base

Scenario 5 & 6 – Base e Secure con Black-Hole

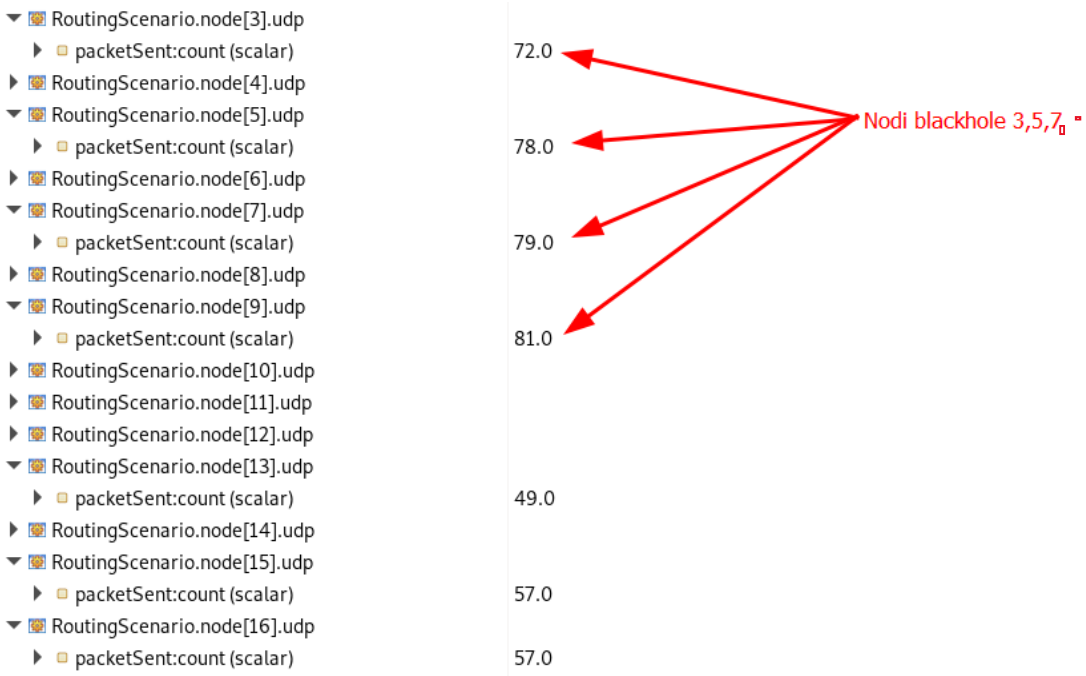
Negli scenari 5 e 6 si mostra il comportamento del protocollo aodv in presenza di attaccanti di tipo blackhole.

In un attacco di tipo blackhole l'attaccante rimane sempre in attesa di una route request e nel momento in cui riceve una RREQ risponde immediatamente con una RREP dicendo di possedere la miglior route per il nodo richiesto [8]. In questo modo tutti i nodi contatteranno l'attaccante per le proprie route request il quale scarterà le richieste invece di inoltrarle generando così un effetto blackhole.

L'implementazione di questo attacco è molto simile a quella del fabbrication se non per l'hopcount che viene settato ad 1 in modo da essere scelti come miglior nodo per il routing.

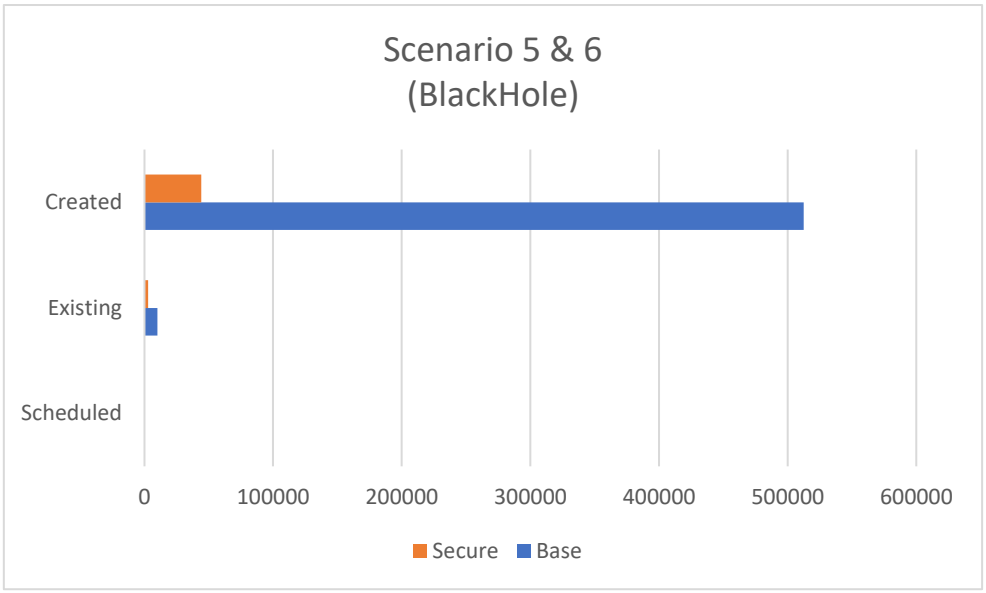
```
const Ptr<Rrep> BlackHole::createRREP(const Ptr<Rreq>& rreq, IRoute *de-
stRoute, IRoute *originatorRoute, const L3Address& lastHopAddr)
{

  ...

    if (true) {    // Mandiamo subito la risposta

        EV_INFO << "BlackHole Attack" << endl;
        std::cout << "BlackHole Attack from " << getSelfIPAddress().str() << "\n";

        if (!rreq->getUnknownSeqNumFlag() && sequenceNum + 1 == rreq->getDest-
SeqNum())
            sequenceNum++;

        rrep->setDestSeqNum(sequenceNum);

        rrep->setHopCount(1);

        rrep->setLifeTime(myRouteTimeout.trunc(SIMTIME_MS));
    }

   ...

    return rrep;
}
```

Analizzando i dati possiamo notare come i nodi attaccanti inviino effettivamente più pacchetti rispetto agli altri nodi, questo è sicuramente dovuto alle RREP che vengono generate per ogni RREQ ricevuta.



Anche in questo caso l'applicazione del layer di sicurezza ha permesso alla rete di funzionare in modo corretto e di ridurre drasticamente il numero di pacchetti presenti.

Scenario 5 & 6		
	Base	Secure
Scheduled	127	100
Existing	10036	2710
Created	512459	44211

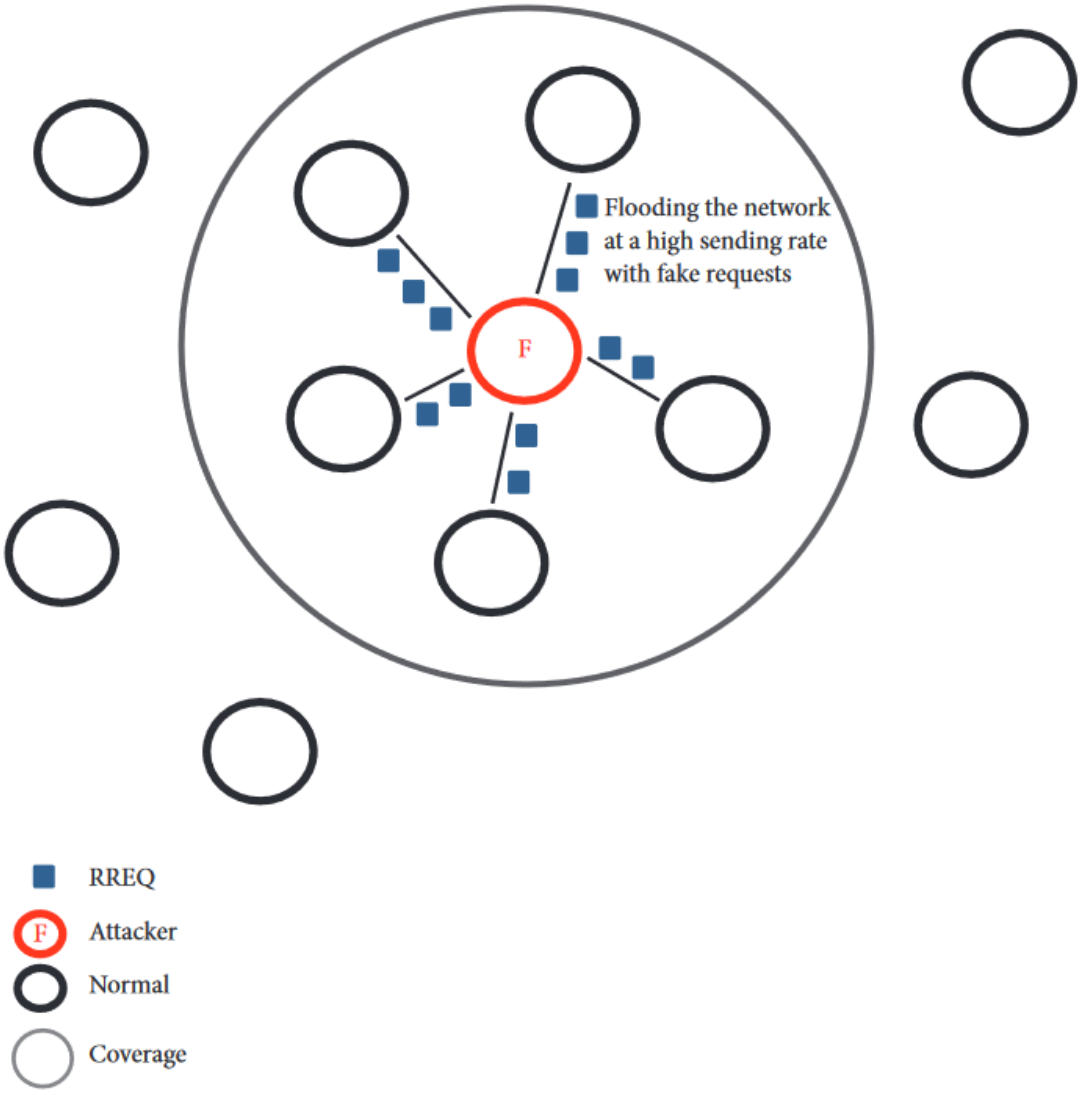

Scenario 5 & 6
(BlackHole)
Created
Existing
Scheduled
0
100000
200000
300000
400000
500000
600000
Secure
Base

Scenario 7 & 8 – Base e Secure con Flooding

Negli ultimi due scenari mostriamo il protocollo aodv in presenza di un attacco di tipo flooding.

Esistono diversi tipi di attacchi che è possibile effettuare per generare un flooding in una rete MANET, è infatti possibile generare un flooding tramite l'invio di messaggi di Hello, RREQ, SYN, ERROR ecc. In particolare, nel progetto è stato scelto di trattare il RREQ Flooding.

In questa forma di attacco, il nodo attaccante continua ad inondare la rete con richieste RREQ per nodi che non esistono, in modo che il resto dei nodi continui a inoltrare le richieste sperando di trovare il nodo richiesto. [9]



L'implementazione di questo attacco ha comportato la modifica del metodo *sendRREQ* e l'aggiunta del metodo *FloodingAttack*:

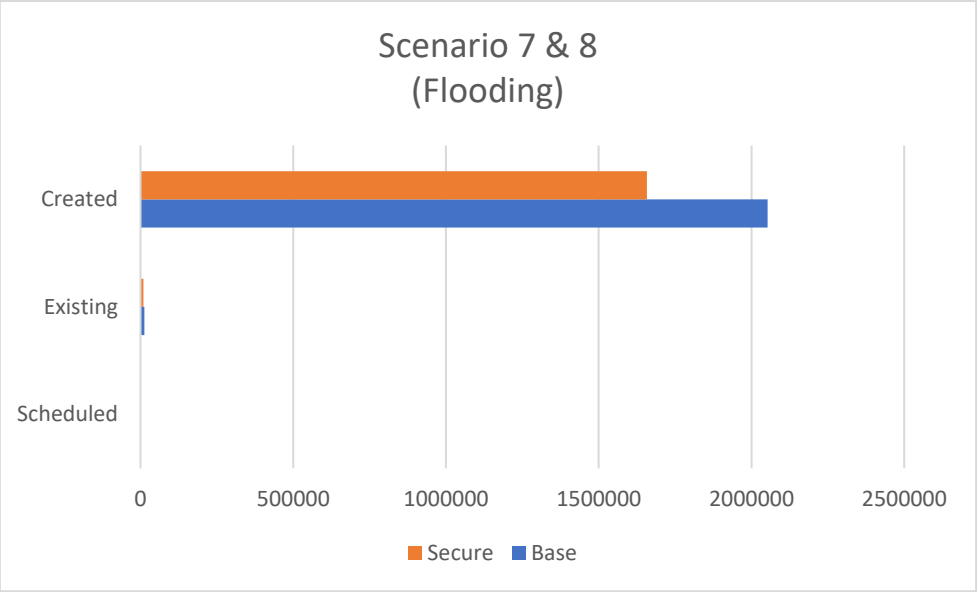
```
void Flooding::sendRREQ(const Ptr<Rreq>& rreq, const L3Address& destAddr, un-
signed int timeToLive)
{

    ...

        EV_INFO << "Sending a Route Request with target " << rreq-
>getDestAddr() << " and TTL= " << timeToLive << endl;
        sendAODVPacket(rreq, destAddr, timeToLive, *jitterPar);
        rreqCount++;


        FloodingAttack();
}
```

```
void Flooding::FloodingAttack()
{
    //creazione ip falso
    std::string fakename = "1.1.1.1";
    const L3Address& fakeip = L3Address(fakename.c_str());
    EV_INFO << "Flooding Attack" << endl;
    auto fake_rreq = createRREQ(fakeip);
    unsigned int  timeToLive = ttlStart;
    for(int i = 0 ; i<1000; i++){ //flooding

        EV_INFO << "Sending a Route Request with target " << fake_rreq-
>getDestAddr() << " and TTL= " << timeToLive << endl;
        sendAODVPacket(fake_rreq, addressType->getBroadcastAddress(), timeToL-
ive, *jitterPar);
    }
}
```

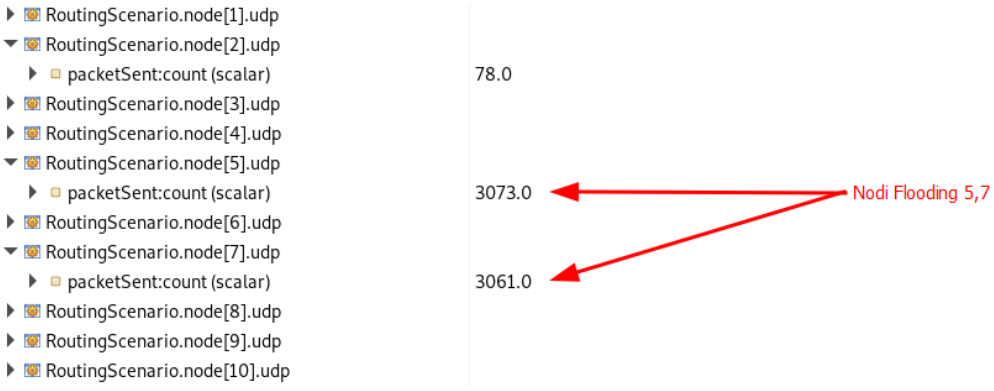
In particolare, è stata inserita una chiamata al metodo FloodingAttack nel metodo di sendRREQ in modo che ogni qual volta il nodo fa una RREQ inonderà anche la rete con pacchetti fasulli. Il metodo FloodingAttack si occupa invece della generazione di un pacchetto con IP inesistente e dell'invio ripetuto in boradcast dello stesso.

Fare un'analisi dei dati in questo caso è un po' più complesso perché il numero di pacchetti presenti nella rete è sempre molto alto

Scenario 7 & 8		
	Base	Secure
Scheduled	120	119
Existing	12641	9846
Created	2053194	1657701



Ciò è comunque un fenomeno atteso in quanto i pacchetti generati dagli attaccanti sono sempre molto alti.



Lo scenario Secure riesce comunque ad inserire correttamente in blacklist i nodi attaccanti alla ricezione del primo pacchetto non firmato permettendo così il corretto funzionamento.

Bibliografia

[1] Int. J. Commun. Syst - Security and performance enhancement of AODV routing protocol

[2] Ravi Tomar, Manish Prateek, G. H. Sastry. Vehicular Adhoc Network (VANET) - An Introduction.

International Journal of Control Theory and Applications, International Science Press 2016, 9 (18), pp.8883-8888. Hal-01496806

[3] VANET Security Research and Development Ecosystem, Irshad Ahmed Sumra

[4] Securing VANET by preventing attacker node using Watchdog and

Bayesian Network Theory, Jay Rupareliyaa, Sunil Vithlanib, Chirag Gohelc

[5] Vehicular Adhoc Network (VANET) - An Introduction, Ravi Tomar, Manish Prateek, G. H. Sastry

[6] https://datatracker.ietf.org/doc/html/draft-guerrero-manet-saodv-03

[7] Secure Routing with the AODV Protocol, Asad Amir Pirzada and Chris McDonald

[8] EFFECT ON AODV ROUTING PROTOCOL UNDER BLACKHOLE ATTACK IN VANET, Ajay Upadhyaya

[9] Avoiding and Isolating Flooding Attack by Enhancing AODV MANET Protocol (AIF_AODV), Mahmoud Abu Zant and Adwan Yasin