



NRL/MR/5523--20-10,157

## Multi-Sensor Message Security

JIANWEI W. CHAO  
LAN H. TRAN  
TRANG V. MAI  
ERIC T. MAKARA  
SASTRY KOMPELLA  
JOSEPH A. MOLNAR

*Networks & Communication Systems Branch  
Information Technology Division*

October 20, 2020

**DISTRIBUTION STATEMENT A:** Approved for public release, distribution is unlimited.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. <b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b>					
1. REPORT DATE (DD-MM-YYYY) 20-10-2020		2. REPORT TYPE NRL Memorandum Report		3. DATES COVERED (From - To) 02-06-2020 – 30-09-2020	
4. TITLE AND SUBTITLE  Multi-Sensor Message Security				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER NISE	
6. AUTHOR(S)  Jianwei W. Chao, Lan H. Tran, Trang V. Mai, Eric T. Makara, Sastry Kompella, and Joseph A. Molnar				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER 1X10	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Naval Research Laboratory 4555 Overlook Avenue, SW Washington, DC 20375-5320				8. PERFORMING ORGANIZATION REPORT NUMBER  NRL/MR/5523--20-10,157	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Naval Research Laboratory 4555 Overlook Avenue, SW Washington, DC 20375-5320				10. SPONSOR / MONITOR'S ACRONYM(S)  NRL NISE	
				11. SPONSOR / MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT  DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT  This work provides details for improving upon the existing solutions for communications between radios and distributed sensors by utilizing industry standard RSA and AES encryption techniques with adaptations suitable for a diverse DSA network. The techniques described in this report provide message confidentiality, integrity, and authentication into the current DSA network paradigm through the usage of new features including RSA based key-exchange, AES key unicast/multicast encryption, a packet driven state machine, and active sensor database within the radio. These techniques have been demonstrated in a simulated software network consisting of one simulated DSA-enabled radio and up to four simulated sensors.					
15. SUBJECT TERMS  Multi-sensor      Message Security      RSA encryption AES encryption      DSA sensor      Sensor Fusion					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT  Unclassified Unlimited	18. NUMBER OF PAGES  17	19a. NAME OF RESPONSIBLE PERSON Jianwei W. Chao
a. REPORT Unclassified Unlimited	b. ABSTRACT Unclassified Unlimited	c. THIS PAGE Unclassified Unlimited			19b. TELEPHONE NUMBER (include area code) (202) 767-2186

This page intentionally left blank.

## CONTENTS

EXECUTIVE SUMMARY .....	E-1
1. INTRODUCTION AND STATE OF THE ART .....	1
1.1 Introduction .....	1
1.2 State of the Art.....	1
2. TECHNICAL APPROACH TO ENHANCE THE STATE OF THE ART .....	1
2.1 Primer on Encryption and Key Exchange .....	1
2.2 RSA Usage for DSA Networks .....	3
2.3 AES Usage for DSA Networks.....	4
2.4 Unicast Encryption .....	5
2.5 Multicast Encryption .....	5
2.6 Radio-Sensor Interface Message Definitions .....	5
2.6.1 Hello-Request with Public Key .....	6
2.6.2 Hello-Response with Public Key.....	7
2.6.3 Radio UDP Key Dissemination with Public Key .....	7
2.6.4 Radio Data Query with UDP Encryption .....	8
2.6.5 Sensor Data Response with UDP Encryption.....	8
2.6.6 Radio Subscription Request.....	8
2.6.7 Sensor Publish Response.....	9
2.6.8 Keep-Alive .....	9
2.7 Radio-Sensor Communications State-Machine .....	9
2.8 Enhanced Sensor API and Aggregation Center.....	11
3. MESSAGE PROTOCOL IMPLEMENTATION IN A SIMULATED NETWORK.....	11
3.1 Key Exchange with a Single Sensor .....	11
3.2 Hello-Request Exchange with a Single Sensor.....	12
3.3 Demonstration of Multi-Sensor Communications with Message Security.....	14
4. NEXT PHASE OF DEVELOPMENT .....	19
4.1 Messaging and Protocol Enhancements .....	19
4.1.1 Multicast Addressing with Embedded UDP Keys .....	19
4.1.2 New Entrant Discovery .....	19
4.1.3 Identity Management for Network Participation.....	20
4.1.4 Considerations for Multi-Radio and Multi-Sensor Enhancements.....	20
4.2 Integration and Experimentation with EMANE .....	20
4.3 Prototype Implementation on USRP with Sensing Capabilities.....	21
5. CONCLUSION .....	21
REFERENCES .....	22

## FIGURES

Fig. 1 – RSA Key Exchange .....	2
Fig. 2 – AES Encryption and Decryption .....	3
Fig. 3 – Public Key and UDP Key Exchange .....	4
Fig. 4 – Hello-Request and Hello-Response .....	6
Fig. 5 – SSH Identity Confirmation using PuTTY .....	7
Fig. 6 – Query and Response .....	8
Fig. 7 – Radio-Sensor State Machine.....	10
Fig. 8 – Key Exchange with RSA Encryption and Decryption.....	12
Fig. 9 – Unencrypted Hello-Request Message.....	13
Fig. 10 – Encrypted Message Demonstration .....	13
Fig. 11 – Radio Display showing Multi-Sensor Communications with Message Security .....	15
Fig. 12 – Wireshark Capture on Radio showing Multi-Sensor Communications with Message Security .	16
Fig. 13 – Display on Sensor 1 and Sensor 4 .....	17
Fig. 14 – Display on Sensor 2 and Sensor 3 .....	18

## EXECUTIVE SUMMARY

The Small Form Factor Dynamic Spectrum Access-enabled Tactical Targeting Networking Technology (SFF DSA TTNT) terminal developed in [1] is one of the first tactical radio units to implement the combination of spectrum sensing, sensor fusion, and DSA policy reasoning within a proto-production unit. The SFF DSA TTNT terminal has the ability to leverage both internal (within the terminal) and external (independent or within neighbor terminals) radio frequency (RF) spectrum sensors to monitor the spectral environment and provide information to feed the DSA reasoning engine in order to enable spectral agility and radio configuration adaptations in environments where spectrum coexistence is required. Previous work has specified and implemented a sensor interface to the radio terminal [2] but critical areas including message confidentiality, integrity, and authentication have yet to be addressed.

The newly enhanced sensor interface described in [3] enables more efficient radio-sensor communications and improved data robustness for better decision making in spectrum-agile radios. This report provides details for improving upon the existing solutions for communications between DSA-enabled radios and distributed sensors by utilizing industry standard Rivest-Shamir-Adleman (RSA) [4] and Advanced Encryption Standard (AES) [5] encryption techniques with adaptations suitable for a diverse network of DSA radios. A custom sensor message structure is utilized, and an enhanced messaging protocol is detailed to provide for message security. The techniques described in this report provide message confidentiality, integrity, and authentication into the current DSA network structure through the usage of RSA based key-exchange, AES key unicast encryption, a packet driven state machine, and a sensor Identity Database within the radio. These techniques have been implemented and demonstrated in a simulated software network consisting of one simulated DSA-enabled radio and up to four simulated sensors.

This report also outlines the next phase of development where additional messaging enhancements will be implemented, along with the integration into a real-time emulation framework, and the instantiation of a hardware prototype reference implementation. The proposed messaging improvements include leveraging multicast communications to distribute keys and data, reducing the overhead required for typical key maintenance schemes. Considerations for multi-radio, multi-sensor networks are discussed and further investigation will be conducted into the security implementation details required to realize large-scale DSA networks. The stepping stones are in place to enable researchers to rapidly iterate upon the implementation demonstrated in this report, in order to deploy these enhancements to more realistic emulation scenarios, hardware prototypes, and production radio systems.

This page intentionally left blank.

# MULTI-SENSOR MESSAGE SECURITY

## 1. INTRODUCTION AND STATE OF THE ART

### 1.1 Introduction

The Small Form Factor Dynamic Spectrum Access-enabled Tactical Targeting Networking Technology (SFF DSA TTNT) terminal developed in [1] is one of the first tactical radio units to implement the combination of spectrum sensing, sensor fusion, and DSA policy reasoning within a proto-production unit. The SFF DSA TTNT terminal has the ability to leverage both internal (within the terminal) and external (independent or within neighbor terminals) radio frequency (RF) spectrum sensors to monitor the spectral environment and provide information to feed the DSA reasoning engine in order to enable spectral agility and radio configuration adaptations in environments where spectrum coexistence is required. Previous work has specified and implemented a sensor interface to the radio terminal [2] but critical areas including message confidentiality, integrity, and authentication have yet to be addressed. This report provides details for improving upon the existing solutions for sensor-to-radio communications by utilizing industry standard encryption techniques, a custom sensor message structure [3], and new messaging protocol to provide enhanced message security.

### 1.2 State of the Art

Existing solutions in this space, including those conducted in [1] and documented in [2] have created a baseline for continued development in the communications protocols and standardization used in DSA-enabled radio networks. Specifically, the radio-to-sensor and sensor-to-sensor interfaces are of interest. The previous work [1] provided support for a single external sensor. A detailed exploration of an enhanced application programming interface (API) that supports multiple external sensors in communication with a radio terminal is documented in [3] and accompanies this report.

Current solutions have implemented a sensor interface to the terminal that leverages User Datagram Protocol (UDP) messages with specific message formatting. The use of UDP messages for sensor communications provides a starting point, but areas including message confidentiality, integrity, and authentication have yet to be addressed. Current radio-sensor communications implementations do not employ data encryption between the radio and sensors. The traffic between the radio and multiple spectrum sensors involve radio communication that is subject to monitoring and could be intercepted by an adversary. In order to protect the contents of the messages, and to prevent malicious packet injections, a scheme leveraging existing standards of encryption is proposed. Two commonly employed encryption standards for wireless communications are RSA [4] and AES [5]. While the specifics of these encryption standards are discussed in relation to sensor communications in this report, this document does not provide an in-depth exploration into the cryptographic algorithms that comprise these solutions. An interested reader is directed to the standards documentation in [4] and [5].

## 2. TECHNICAL APPROACH TO ENHANCE THE STATE OF THE ART

### 2.1 Primer on Encryption and Key Exchange

There are several encryption standards that are commonly used in production wireless communication systems. These standards, detailed in [4], [5], and [6] are well defined and well-studied to withstand many types of security attacks. This document details the implementation of select standards to provide the encryption needs for radio-sensor communications in a DSA network.



A simplified depiction of the RSA key exchange process for radio-sensor communications is shown in Fig. 1. First, the radio and sensor generate their own private/public key pairs and store them in memory. Then the radio and the sensor exchange their public keys. The radio sends its generated session key along with the sensor's public key. Throughout this report, the terms "UDP key" and "session key" may be used interchangeably. The radio uses its private key to recover the UDP key. The strength of the private key is  $2^{3081}$  operations. With the current most powerful computer, compromising the private key via brute force methods would take  $10^{996}$  years.

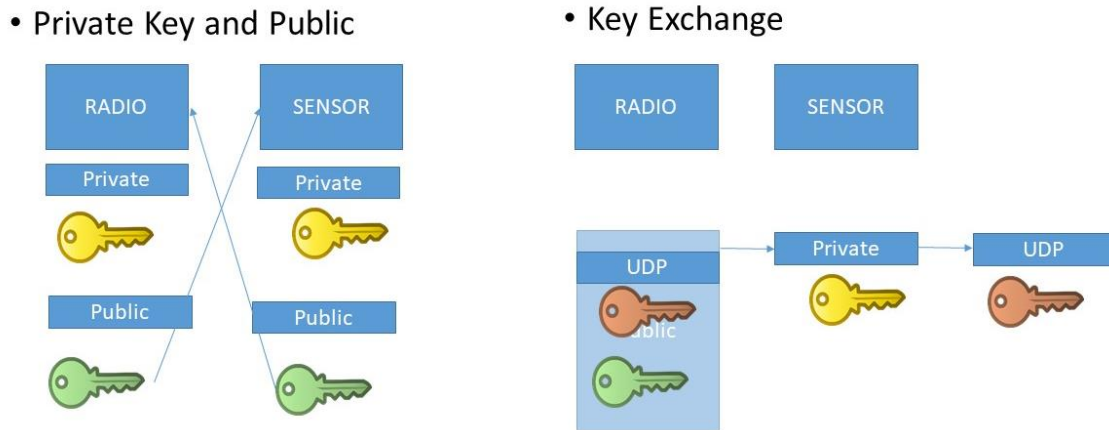


Fig. 1 – RSA Key Exchange

The algorithm described by AES [5] is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data. A depicted of the AES encryption and decryption processes is provided in Fig. 2 and illustrates the concept of the shared key. The symmetric AES key is used to protect the UDP data traffic.

In this document, a new method of embedding the session key with the data is proposed to reduce the complexity of the key maintenance. The unicast traffic uses a simplified session key exchange process to reduce the load of key maintenance in DSA-enabled radio terminals. In our proposed solution, only the radio generates the key and distributes it to all sensors. Therefore, only one key is maintained and used. In addition to the implementation of these standards, new methods are proposed to further simplify the key exchange process and key management. This design can be used to implement unicast UDP traffic protection, and extend the process into the multicast UDP traffic encryption as well.

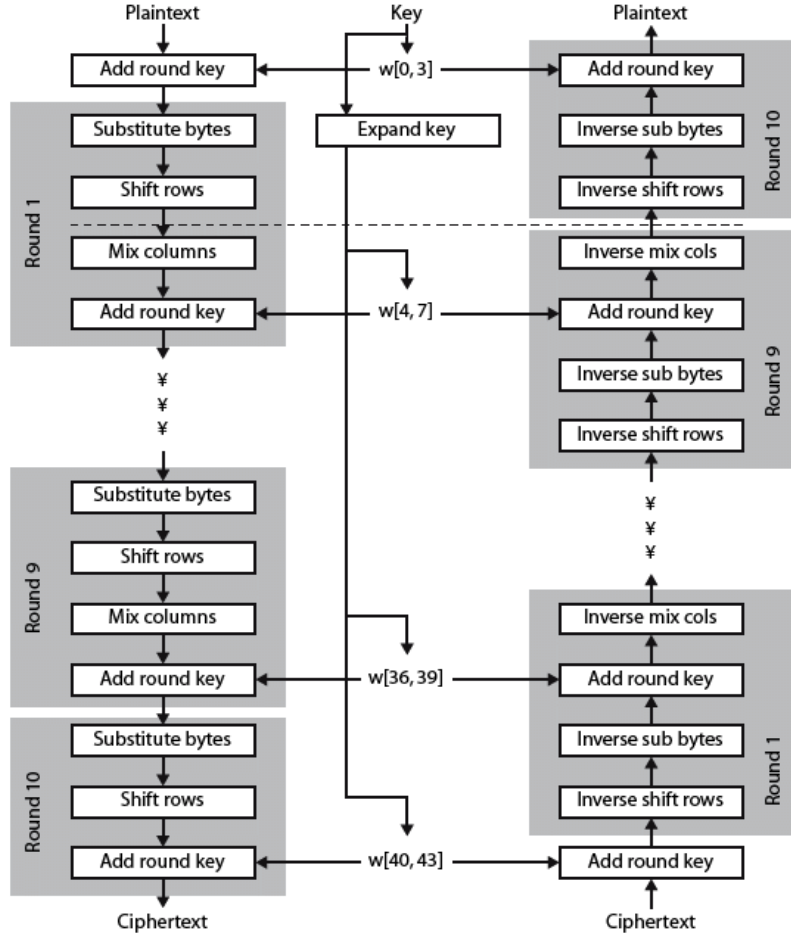


Fig. 2 – AES Encryption and Decryption

## 2.2 RSA Usage for DSA Networks

The RSA encryption is a well-known standard used to provide data security for small messages that are transmitted infrequently. RSA is best used to protect the session key (or UDP key in this document) that protects the UDP data traffic. The exchange of the session key for radio-sensor communications utilizes RSA for this purpose. Its usage is shown as blue and green lines in Fig. 3. A radio can send a Hello-Request message (optionally with a public key) to a sensor or vice-versa in order to establish initial contact. In response to a Hello-Request message, a Hello-Response message is sent back along with the sender's public key. The public key can then be used for verification and identification, detailed in 2.6.1.

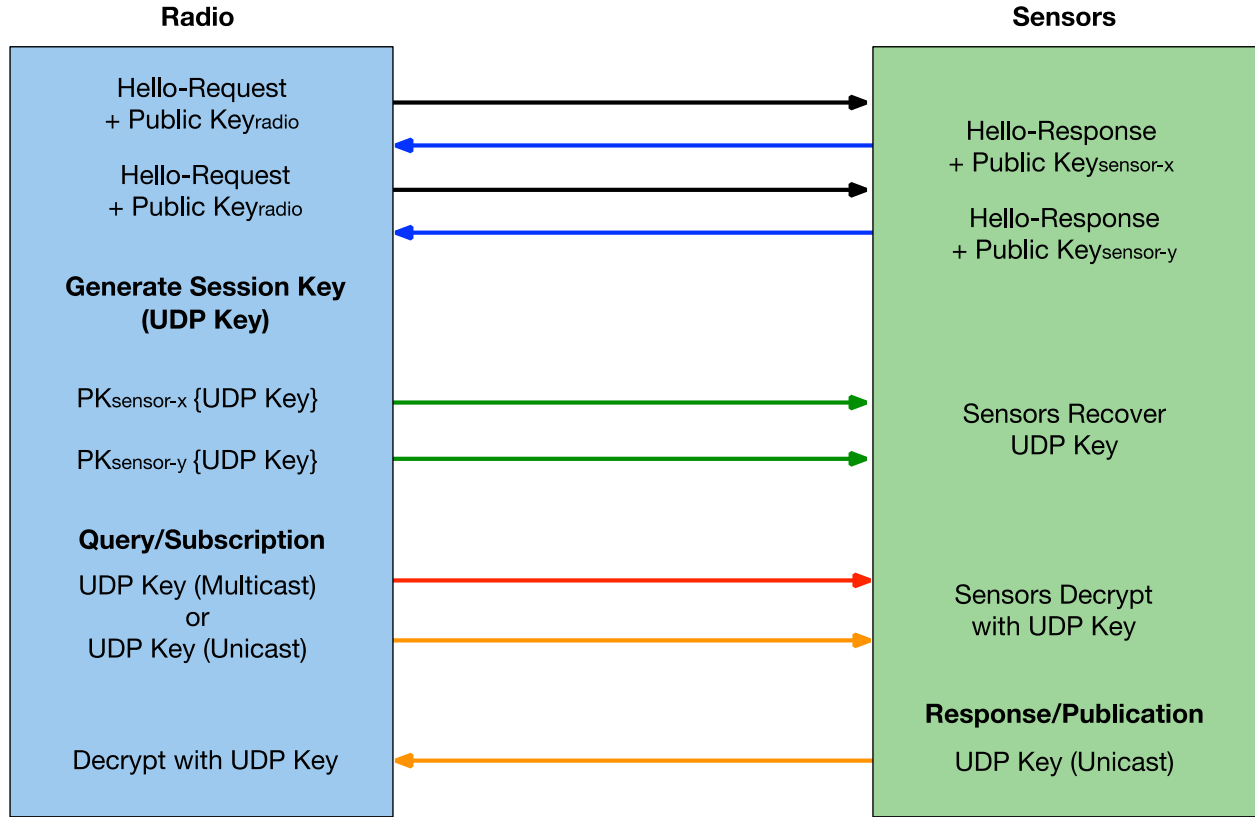


Fig. 3 – Public Key and UDP Key Exchange

In the next step, before a radio (or a sensor) sends out its UDP traffic, the UDP-key is sent to communications partners, encrypted with the intended receiver's public key. The sensor then uses its private-key to decode the packet, and thus obtain the UDP-key. This is depicted by the green lines in Fig. 3. Note that no one else but the intended receiver can decode this packet. Even if this packet is intercepted by an adversary, it cannot be decoded correctly because they do not have the private key of the intended recipient.

By design, the sender (the radio in this case) is responsible for generating the UDP-key for two purposes: 1) one key is used for all sensors which enables the possibility of unicast and multicast traffic, and 2) the same key can be used for the sensor to send encrypted data back as a response. This new design reduces the burden of key maintenance on the sensor and simplifies the key exchange. Therefore, by design, the radio becomes sole key owner. After the UDP key arrives at the sensor, the radio starts to send the encrypted UDP traffic.

### 2.3 AES Usage for DSA Networks

AES encryption is defined in [5], which details the algorithms and methodology used to encrypt and decrypt data. The standard is recommended by the National Institute of Standards and Technology (NIST) for commercial, government, and military use for secured communications. Fig. 2 illustrates the core design of AES. The encryption process uses multiple rounds of substitution, shifting, and mixing to transform the plaintext input into ciphertext output. For decryption, inverse operations are used to recover the plaintext. AES is a symmetric-key encryption scheme, because the same key is used for both encryption and

decryption. This paper details the use of AES to protect the UDP traffic of the radio and sensor. The details include key updates, key exchange, and data encryption processes.

## 2.4 Unicast Encryption

The unicast UDP traffic is encrypted with the UDP-key. This key is to be updated every 30 seconds, but the design may allow for this parameter to be configurable depending on the mission requirements. The radio sends a query to all sensors with a unicast traffic flow. Each sensor receives a query from the radio. The sensor sends a response back to radio using unicast with the same UDP key. In future implementations, the Hello-Request/Hello-Response exchange can be optimized using multicast traffic flows when multiple sensors appear in the same subnet with the radio, as described in section 4.1.1.

## 2.5 Multicast Encryption

In traditional data networks, multicast traffic is not routed by any Internet Service Provider (ISP) or network operator. However, the radios can easily support single-hop multicast, where no routing is required. This mechanism enables the possibility of using one multicast packet to query multiple sensors that are in the same subnet. Even though existing multicast encryption standards given in [6] can be used, they are impractical for the radio-sensor communications because of the significant messaging overhead and processing load in the terminal required for key maintenance. A new method that distributes the key and data is proposed in this document. The initial implementation targets unicast streams, but it can be easily extended to multicast as well. The new method can further improve the key distribution process with one UDP data stream, by piggybacking the keys with the user the traffic, such that a single multicast stream can carry both keys and data.

The multicast UDP traffic is encrypted with the UDP-key. This key is to be updated every 30 seconds, but the design may allow for this parameter to be configurable depending on the mission requirements. This innovation is made possible due to the fact that the UDP-key is controlled and originated in the radio, so no extra key maintenance is required. The UDP-key are distributed the same way as the unicast UDP-keys. This feature cannot be fully implemented at present since standard networking devices do not support the forwarding of multicast traffic without significant customization. The multicast messaging implementation is left to a future effort, described in section 4.1.1.

## 2.6 Radio-Sensor Interface Message Definitions

As depicted in Fig. 3, there are many types of packets transmitted between the radio and the sensor(s). The various packet types are defined here to provide a baseline definition of the messaging protocol and to describe the actions undertaken when a packet is received. For this section, the terms message and packet may be used interchangeably. These radio-sensor interactions are illustrated Fig. 4 and Fig. 6. A Hello-Request message is sent and received by its destination, where a Hello-Response message is returned to the original sender. When a Query is sent, a response is returned, with optional keep-alive. When a subscription is sent, a periodically publication is returned. The details are explained in the following sections.

The identity management (using public key) is carried out during the handshake (Hello-Request and Hello-Response). The method is very similar to the popular Unix/Linux Secure Shell (SSH) utility [12]. The details are explained in 2.6.1.

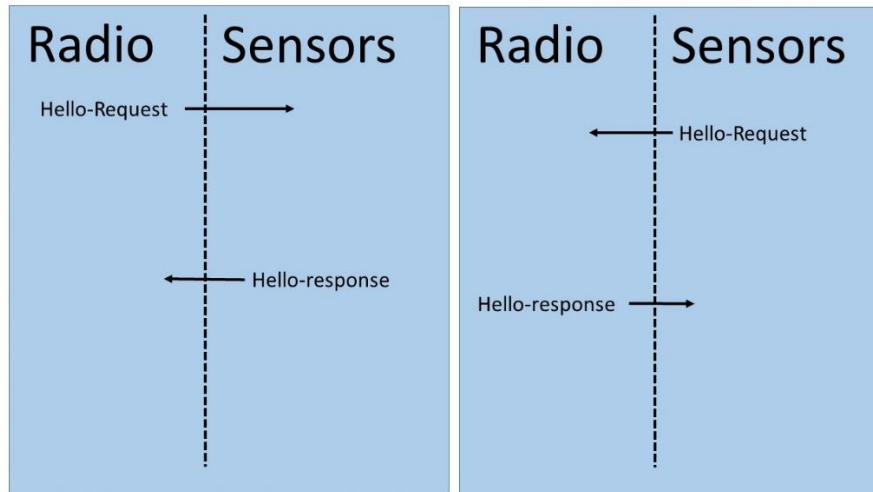


Fig. 4 – Hello-Request and Hello-Response

### 2.6.1 Hello-Request with Public Key

When the radio or sensor starts up (or a after restart), a Hello-Request packet is sent to all members of the Aggregation Sensors List [3]. The Hello-Request packet is the initial mechanism that a radio uses to discover a sensor, or vice-versa. In Fig. 4, note that the initial Hello-Request packet can be initiated by either the radio or the sensor. The other party then sends a Hello-Response in response. In similar handshake implementations, these messages are called Client-Hello and Server-Hello. However, this implementation does not have dedicated client or server roles, only radio and sensors, and therefore the general terms Hello-Request and Hello-Response are used instead to describe the handshake process where either party can initiate the communication.

The radio and sensor(s) are required to communicate with each other using defined Internet Protocol (IP) addresses in the Aggregation Sensors List. In the Hello-Request message, a public key is sent to provide identity verification. After receiving the Hello-Response packet, the public key is checked against a list (the Identity Database) of known IP addresses with its associated public keys. If the public key matches the entry in the list, the Hello-Request is accepted and a Hello-Response is returned containing the sender's own public key. The Aggregation Center's [3] Identity Database is updated as a result of the reception of Hello-Response packets. If it does not match the entry in the list or does not exist in the list at all, a user confirmation/rejection is required to proceed.

This implementation is similar to that of the popular SSH protocol demonstrated with the PuTTY [13] client utility. The user confirmation is demonstrated in Fig. 5. The Identity Database is stored in a protected hidden directory of the Aggregation Center, similar to SSH implementations. When a new IP is confirmed, it is added to the database for future reference. The confirmation/rejection can also be configured for auto-confirmation/auto-rejection in an unattended radio/sensor. The current implementation is set to auto-confirm, meaning all entries are accepted.

Note that the Aggregation Sensors List contains all sensors with which the radio will try to connect, while the Active List (details in 2.6.8) contains the current active IP addresses that are currently communicating with the radio. The Identity Database is used to verify and maintain the IP address with its associated public key for identification verification. The implementation of using unknown IPs is left to be implemented in the enhanced discovery mode of the future version. The Hello-Request packet is designed to use unicast addressing, but can be modified to use single-hop multicast for discovery purposes in the future.

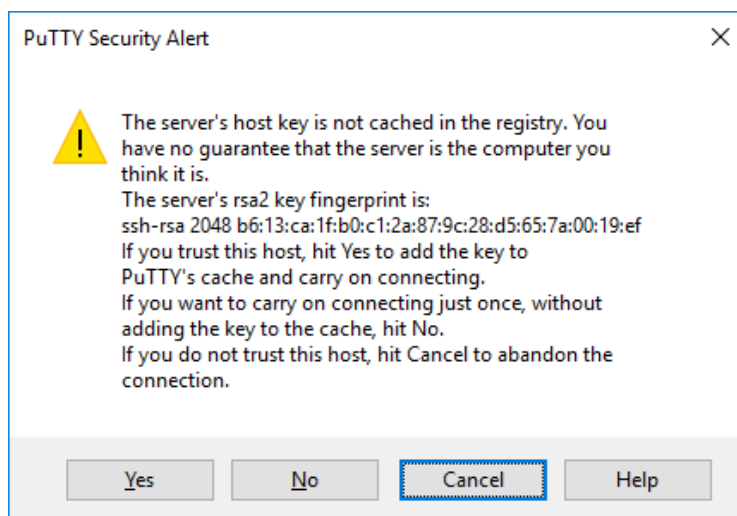


Fig. 5 – SSH Identity Confirmation using PuTTY

### 2.6.2 Hello-Response with Public Key

Upon reception of a Hello-Request packet, the receiving sensor (or radio) sends back a Hello-Response packet with its own public key. This public key is used for key-exchange purposes and provides a method of identity verification. The public key is checked with an Identity Database in a protected hidden directory using methodology described in 2.6.1. Once the connection is accepted, the IP is entered into the Active List for future interactions. The current implementation is set to auto-confirm, meaning all connections are accepted.

The pair of Hello-Request and Hello-Response messages can be used in radio-sensor or sensor-radio discovery as shown in Fig. 4. It is also can be used as a mechanism for reconnection. For example, if a sensor restarted in the middle of the process, upon restart, the sensor sends a Hello-Request to the radio, and the radio responds with a Hello-Response. The radio then puts the sensor back into the Active List, and the process is resumed as normal. This is done via a state-machine, instead of sequential-machine, to use the Hello-Request/Hello-Response mechanism for discovering and reconnecting as shown later in Fig. 7. A sensor or a radio can accept Hello-Request or Hello-Response messages at any time and use these packets to maintain its Active List and Identity Database.

### 2.6.3 Radio UDP Key Dissemination with Public Key

The radio is responsible for generating the UDP-key (session key) and sending it to a destination sensor by encrypting it with the sensor's public key and transporting via a unicast UDP message. Once the sensor receives this packet, its private key is used to recover the UDP-key. By design, the radio is responsible for updating the UDP-key every 30 seconds, or another update interval as configured for the mission. The radio maintains a list of active sensors and will only send the UDP-key to sensors that are currently in the list. If the sensor doesn't receive UDP-Key update, or if it has not made contact with radio for more than 30 seconds, it is considered disconnected, thus removed from the Active List. To rejoin, the sensor is required to reconnect using the Hello message exchange. The Active List is maintained by the Hello-Request/Hello-Response mechanism described in 2.6.2 and the keep-alive mechanism described in 2.6.8. In the event that a new Hello-Request is used to rejoin the network, the sensor is re-entered into the Active List. The sensor's UDP key is disseminated immediately so that the query (Section 2.6.4) or the subscription (Section 2.6.6) can be honored by the sensor.

#### 2.6.4 Radio Data Query with UDP Encryption

The radio sends the Radio Data Query packet periodically (configurable; default to 1 second) to sensors in the Active List to ask for radio frequency (RF) spectrum sensor data. This packet is unicast to sensors and is protected with the UDP-key. This packet is only sent to sensors in the radio's Active List. If the response (2.6.5) is not received within 30 seconds, the sensor is removed from Active List, as that sensor will no longer have that latest UDP key disseminated by the radio, nor it will query again. In the case where the response is delayed more than 30 seconds, the keep-alive packet (as in 2.6.8) is sent from the sensor to the radio to maintain the entry in the Active List.

Fundamentally, this process is different from the common publish/subscribe (pub/sub) implementation which implies automated periodic updates (publication) from the sensor to the subscribing radio. This query will trigger only one response from the sensor. Therefore, if repeated responses are required, periodic queries need to be sent. For initial development and proof of concept, the Query/Response mechanism is currently implemented in a manner similar to Pub/Sub (2.6.6) but will be modified accordingly for future implementations. Additionally, this packet can be modified to be sent with multicast addressing to reduce traffic volume in future implementations.

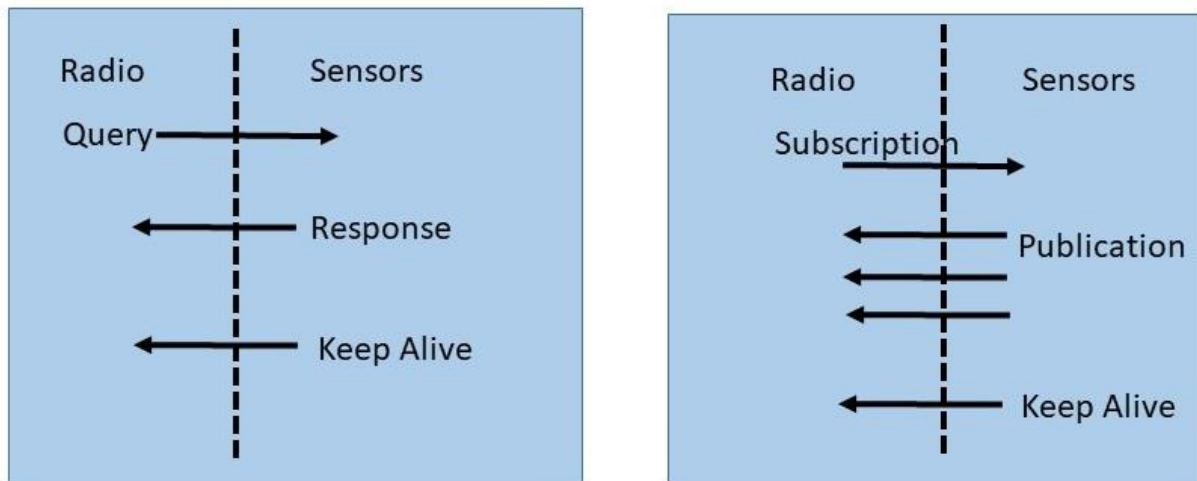


Fig. 6 – Query and Response

#### 2.6.5 Sensor Data Response with UDP Encryption

Once the sensor receives the Radio Data Query message from the radio, it must reply with the Sensor Data Response packet via unicast IP address using the UDP-key (also known as session key given in 2.6.3). The radio decrypts this packet with its current UDP-key. The contents of the Sensor Data Response packet are detailed in [3].

#### 2.6.6 Radio Subscription Request

The radio sends the Radio Subscription Request message to a sensor with unicast addressing. This message specifies how often the sensor should send responses back to the radio. The radio can also send new requests at-will with updated response periods to change the message rate for sensor transmissions. This is different from the Query in that the Subscription Request only needs to be sent once, and the sensor will periodically send its sensory information back with Publish Response messages, discussed in 2.6.7.

### 2.6.7 Sensor Publish Response

The sensor sends the Sensor Publish Response message to the radio periodically via unicast as specified in the Subscription Request message it received from the radio. If the radio does not receive the Publish Response message within 30 seconds, the sensor will be removed from Active List. In the case where the messaging period is greater than 30 seconds, the keep-alive packet (as in 2.6.8) is sent to keep the sensor's entry in the Active List.

### 2.6.8 Keep-Alive

The radio maintains the Active List that removes an inactive IP address after 30 seconds without communication. When a sensor is idling and has nothing to send, it is required to send the Keep-Alive packet to the radio every 10 seconds in order to be kept in the Active List. This period is designed to be less than 30 seconds in case the best-effort UDP messages are not received successfully over the wireless channel. If the Keep-Alive packet is not received by the radio in time, the protocol dictates that a new handshake needs to be re-established via the Hello-Request and Hello-Response message exchange.

If a sensor has not provided, or the radio has not successfully received, one of the following messages within the past 30 seconds, it is removed from the Active List:

- Sensor Data Response
- Sensor Publish Response
- Keep-Alive

Some network conditions that could result network disconnections and require reconnections may include:

- The sensor and/or radio has lost power and rebooted
- The sensor and/or radio have traveled outside of communications range
- Any of the handshake processes were interrupted due to corrupted packets or bit errors
- The sensor configuration has changed (identity, IP address, etc.)

When communications are restored, the sensor or radio can send the Hello-Request, the partner will respond with a Hello-Response. The state-machine implementation in the radio allows connections to be reestablished.

## 2.7 Radio-Sensor Communications State-Machine

In order for the radio to be able to maintain the processing of packets from sensors throughout the message exchange process, a state-machine is defined as shown in Fig. 7. The state-machine is essential to keeping the radio-sensor interactions in a state that either the sensor or the radio can start, stop, or restart at any time independently. The Hello-Request and Hello-Response messages in Sections 2.6.1, 2.6.2 and Fig. 4 explained this with respect to messaging, however, the packet processing engine needs to keep its state transition clearly defined so that new packets and new sensor requests can be processed without confusion. The state-machine diagram depicts the logic that has been implemented in the software simulation network. A Base State is defined, with control-flow blocks that allow the sensor (or radio) to determine which steps in the message exchange have been completed, and what to do if certain messages do not arrive.

Three different lists: 1) Aggregation Sensors List, 2) Active List, and 3) Identity Database are implemented. The Aggregation Sensors List contains the list of sensors that are to be connected at start up. It contains the list of sensors to which the radio sends its initial Hello-Request packets. The Active List is a subset of the Aggregation Sensors List that is updated periodically and adjusted depending on connectivity with sensors.



The Identity Database is used for verifying sensor identification information from the sensors' Hello-Request and Hello-Response packets against a list of sensor IPs and associated public keys.

When the radio starts, it consults the Aggregation Sensors List to send it Hello-Request packets. The sensors who responded with Hello-Responses using the correct public key matching the Identity Database entries are added to the Active List.

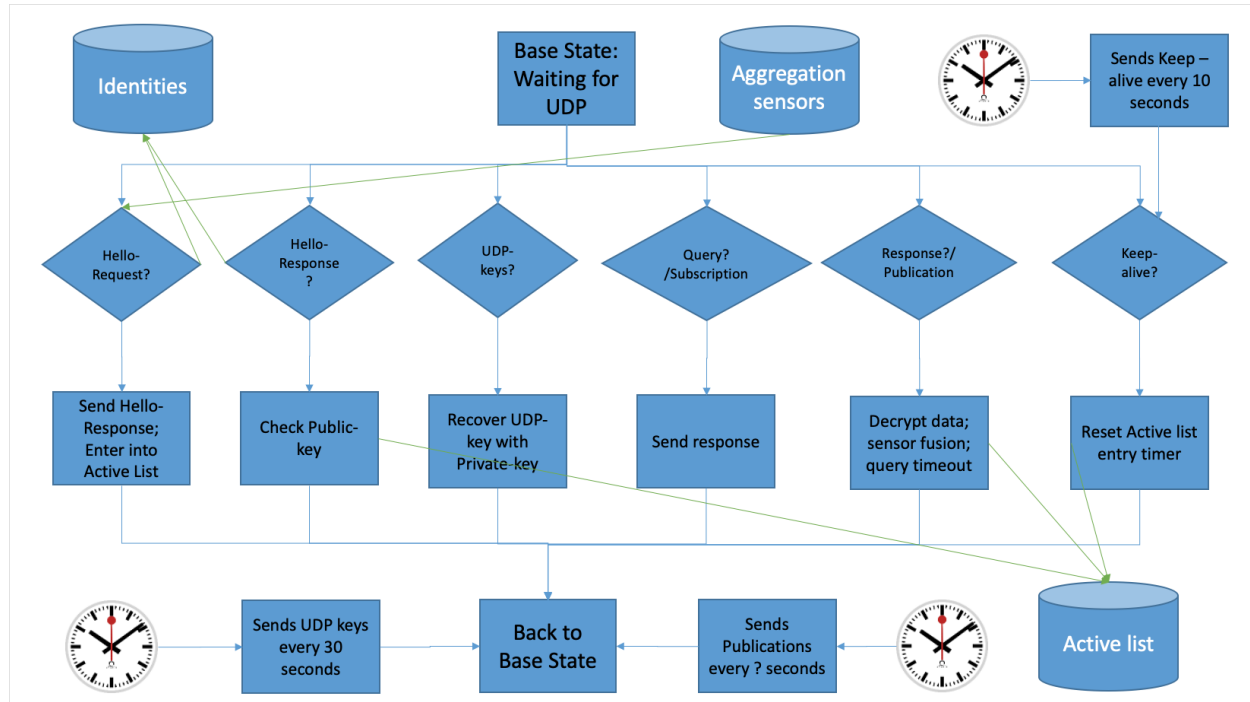


Fig. 7 – Radio-Sensor State Machine

It can be seen in Fig. 7 that the Aggregation Sensors List is consulted by the radio's Hello-Request/Hello-Response process. All members in the list will receive a Hello-Request packet, and respond with a Hello-Response packet. The public key and the IP address combination in the Hello-Response is checked against the Identify Database (upper left in Fig. 7). If it matches, then the sensor's IP is entered into the Active List (lower right). The radio's Active List of sensors is shown with connections to blocks that indicate messages that impact the list, such as the Keep-Alive and the Hello-Request/Hello-Response exchange. If a sensor does not send the Data Response, Publish Response, or Keep-Alive packets to the radio in 30 seconds, or the radio does not receive those packets, the sensor is declared disconnected and is removed from the Active List. In this event, the sensor is apparently not in operation or lost its network connectivity. Once a sensor is reconnected to the network, a new Hello-Request is sent and checked by the radio. If it is allowed, all the processes are resumed (including a UDP key update, subscription request, and etc.), and the sensor has rejoined the radio's network.

The timeout clock is visible at the top-right of Fig. 7, which is a critical innovation to keep the radio-sensor network current and the sensor entries updated in the Active List. When a sensor is idle, or has nothing to send, a Keep-Alive is sent every 10 seconds, which keeps the sensor in the Active List.

The timer on the lower left enables periodic UDP key refreshes every 30 seconds for improved security. Encrypted UDP traffic can be subjected to brute force attacks to obtain the content of the data. With newer hardware and technology, an update of the UDP key every 30 seconds can help prevent malicious activity.

The state-machine also maintains an Identity Database to accept and reject connections. This list is similar to that of common SSH implementations. The Identity Database contains the sensors' IPs and associated public keys in a protected hidden directory on the radio and sensors. The IPs of the Hello-Response packets are used as an index to look into the Identity Database, the public key in the look-up table is checked against the received public key. If it is a match, the sensor IP is entered into Active List. If the index is not found or there is a public key mismatch, it will require a user confirmation or rejection to proceed.

## **2.8 Enhanced Sensor API and Aggregation Center**

A rudimentary application programming interface (API) for radio-sensor communications was developed under the SFF DSA TTNT Program, documented in [1] and [2]. The API enabled a single external sensor to communicate with a DSA-enabled radio terminal to provide distributed sensing information to aid the DSA reasoner within the terminal.

The current work provides enhancements to this API. The messaging security that is the subject of this report has been developed and implemented in software simulation prototypes. The messaging security is implemented in conjunction with the API enhancements in the simulation environment. The simulation environment enables experimentation with the expanded API for multiple sensor connections and extension to the data content fields. The resulting implementation forms the basis for the Aggregation Center – an improved data aggregation method for multiple sensor architectures. These developments are detailed in an accompanying report [3].

## **3. MESSAGE PROTOCOL IMPLEMENTATION IN A SIMULATED NETWORK**

The radio-sensor logic state-machine depicted in Fig. 7 was implemented in software to create simulated sensors and a simulated DSA-enabled radio communicating over the enhanced interface. The details of the software development to implement the enhanced API are included in [3]. The secure messaging protocols were also implemented in these simulators to provide a baseline mechanism of analysis and verification of the encryption scheme and message exchange. This section provides a brief overview of the initial verification tests conducted on the software implementation.

### **3.1 Key Exchange with a Single Sensor**

Before the data communication can be encrypted properly, the radio and sensor need to exchange their RSA public keys. Then the public key can be used to send the UDP (session) key in secret. This is done with a Hello-Request and a Hello-Response exchange as described in 2.6.1 and 2.6.2. Fig. 8 Fig. 8 – Key Exchange shows the UDP keys are distributed (as in 2.6.3) between the radio and the sensor using the public key. In the figure, the UDP packets encrypted with AES.MODE\_CBC, and 30 seconds later a new key is originated from and radio and distributed to the server. The RSA encryption and decryption is shown and highlighted with red color text.

```

lantran - dsalad71@dsalad71: ~/src/SpectrumMask - ssh -Y dsalad71@192.168.1.101 - 133x39
server_string =

**** New key = 'This is a key123', AES.MODE_CBC, 'This is an IV456' 1601305536.83 30
Report_sensing_result $$$$$
('strInput', 'MHBBBHHIIHBBHHIHHHHHdfIhhhb')
(' ciphertext = ', '\x128\xaf\xac\x1b\xeb0\x12d\x5f4\x3e26\x26\x12'\xf0\xbfK\x13=\xb30\xf8~o\x9e\xaa\xfa\n\xe26\xfcy\x93\x9c\x1
5p\x89\xfe\x12\x1d82@\x00I4,\xccCaG\x8b\x1c\xa3!\xee\x05;}Uy1\xa8\xa7\xa9a\xc2a\xf0\xa2\xf9\x9f\xdf\xed\x92')
*** updateRate msec = ***
1000
Check time here ...
Time for report
Waiting radio for UDP key ...
waiting to receive Key from radio
received "encrypted_message=('yx\rBUB\x95Q\xb1\xef\x5b5\xa2d\xde\x3f3\xa1q\x83V\xd4J\xd3#\xed\xfd\xddu\x0b\xfb\xeb\xea\x1b\x81\xd1\xd4c
\x006\xbf\x1f\xa2)\x9fa\x16\x96\xb6<I\xb1\x1b\x7f\x7d7M\x99z@e\x8d\xbdZ\xd1\x8b\xa9\x97d\xe3;dL\xeb!\x11\xe0\xd6h\xd6\x5c7\xa1
8\x1ahty\x1f1\x02\x12d@\xd0C\rh;\xc27\xea\x8e\x00T\x9b\x94jF5\xa6d>K0\x03\xdb\x85Kf\x97\x97~##',)" from ('192.168.
9.103', 5005)
Received:
Encrypted message = ('yx\rBUB\x95Q\xb1\xef\x5b5\xa2d\xde\x3f3\xa1q\x83V\xd4J\xd3#\xed\xfd\xddu\x0b\xfb\xeb\xea\x1b\x81\xd1\xd4c\x006\x5
f\x1f\xa2)\x9fa\x16\x96\xb6<I\xb1\x1b\x7f\x7d7M\x99z@e\x8d\xbdZ\xd1\x8b\xa9\x97d\xe3;dL\xeb!\x11\xe0\xd6h\xd6\x5c7\xa1
8\x1ahty\x1f1\x02\x12d@\xd0C\rh;\xc27\xea\x8e\x00T\x9b\x94jF5\xa6d>K0\x03\xdb\x85Kf\x97\x97~##',)
('UDP general key from radio = ', "'Ka0nCRKxATjVbKGg, AES.MODE_CBC, 'txDMvjLPjMrB1RBb'")
!!!! New key = 'Ka0nCRKxATjVbKGg, AES.MODE_CBC, 'txDMvjLPjMrB1RBb'
State machine = 5
**** New key = 'Ka0nCRKxATjVbKGg, AES.MODE_CBC, 'txDMvjLPjMrB1RBb' 1601305537.83 31
Report_sensing_result $$$$$
('strInput', 'MHBBBHHIIHBBHHIHHHHHdfIhhhb')
(' ciphertext = ', '\x13\x7eT\xdd=\xcd\x9a\xa0\xbd8\x9f0N'\x1b\xfb\x80\x16r\x16\x0e\x00\x8c\xa5/\xf7\x13\xbeT\xfbm\x1b\xa97\xb5\xe2\x1
1\xe5(\x0e\xbb+\xfe\x15\x08\xfe\x91\x1f;o\x16\x03\x1bo'\xfdf:\x81>\x03\x92\tC\x7f\xae\xff\x07\xa7\xfc\x8f\x8c\xa9\xb9\xa1\xce\x1f
7\x05")
*** updateRate msec = ***
1000
Check time here ...
Time for report
State machine = 5
**** New key = 'Ka0nCRKxATjVbKGg, AES.MODE_CBC, 'txDMvjLPjMrB1RBb' 1601305538.83 32
Report_sensing_result $$$$$

```

Fig. 8 – Key Exchange with RSA Encryption and Decryption

### 3.2 Hello-Request Exchange with a Single Sensor

The communications protocol was implemented first between a single radio simulator and sensor simulator. The sensor provided information as defined in the enhanced API specification [3] to the simulated radio. First, the communications were tested without encryption. Fig. 9 shows the Wireshark [7] packet inspection utility being used to analyze one of the sensor messages. The screenshot shows the implementation is using the UDP format and a message was exchanged between two hosts with different (but known) IP addresses.

The example message payload is highlighted, and is shown to be in plaintext, meaning a party intercepting this message could easily decipher the meaning. The plaintext message of “Hello” is visible in the figure to show a Hello-Request Message. This demonstration was used to ensure that the messages were being transmitted as expected between the simulated sensor and radio, before adding in extra complexity. The Hello-Request and Hello-Response packets do not need encryption, because in the RSA implementation, the public key is meant to be viewed in the clear. Only the corresponding private key can decrypt the data (seen in Fig. 8 with red text) which is encrypted by the public key.

Fig. 10 shows the Wireshark utility being used to observe the payload of the encrypted messages. Again, the UDP transport is verifiable in the tool. In the figure, it is shown that the message payload is no longer readable, and is depicted as set of garbled characters with no obvious meaning. In the lower windows, the message payload had been encrypted and is no longer be deciphered because peeping Wireshark does not have the appropriate key.

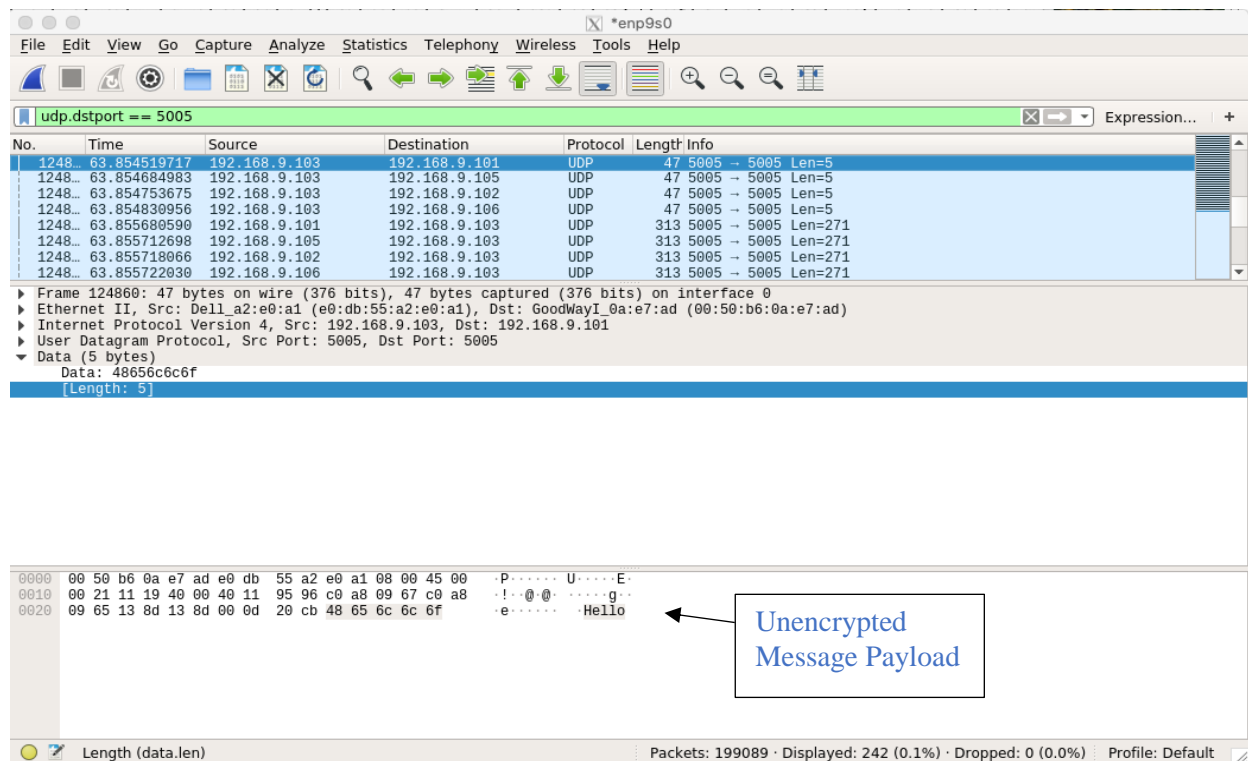


Fig. 9 – Unencrypted Hello-Request Message

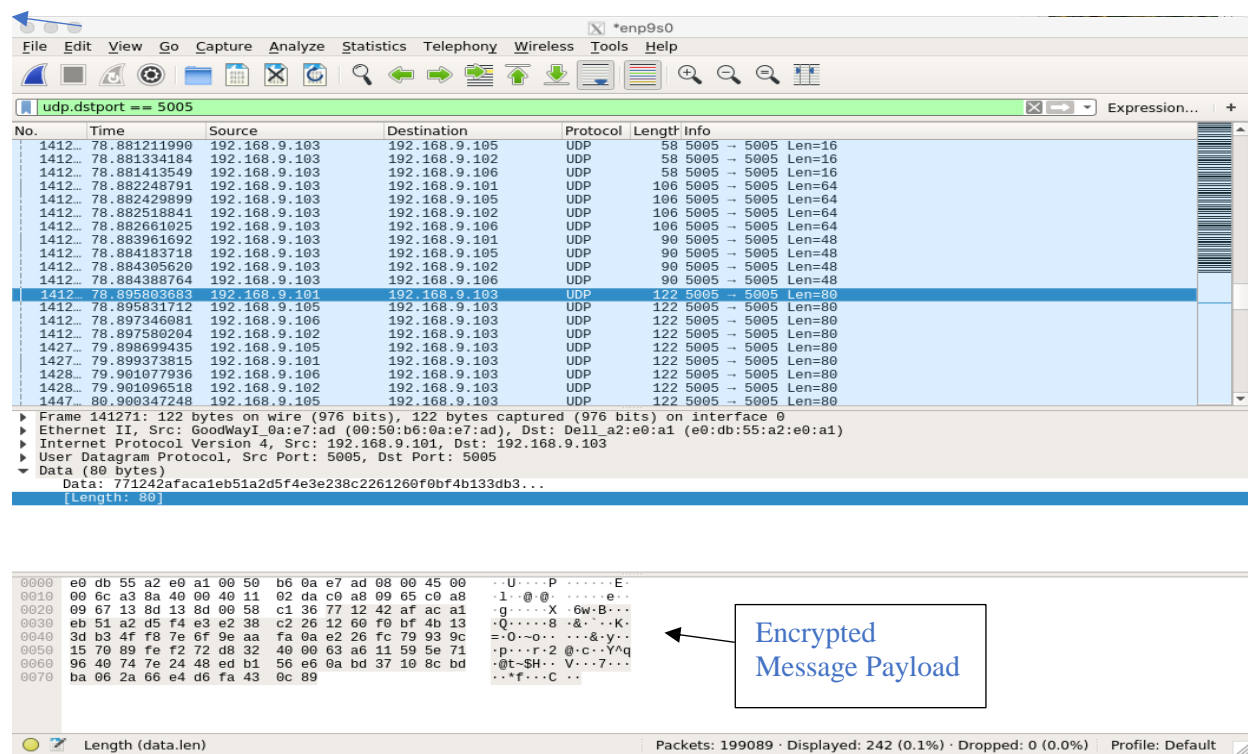


Fig. 10 – Encrypted Message Demonstration

### **3.3 Demonstration of Multi-Sensor Communications with Message Security**

Once the communication was verified between a single simulated sensor and simulated radio, the network was expanded to include four sensors, all providing information updates to the simulated radio. The screenshots in Fig. 11 to Fig. 14 show the running simulation network in action, the screenshot are taken at different locations to show the interactions between the radio and sensors.

The screenshot in Fig. 11 shows the running simulation network in action, from the radio's point of view, including several sensor messages that are being processed. The sensor messages were color-coded to aid development. It can be seen in the Fig. 11 depicting the radio's terminal, that there are four colors of messages (red, cyan, pink, and purple) that represent different sensors. The figure shows that the radio has established communications with Sensor 1 (red), Sensor 4 (cyan), Sensor 3 (pink) and Sensor 2 (purple), and has received the sensory data updates from each one. The data has gone through successful decryption process to recover the plaintext. All four sensors were demonstrated to successfully communicate with the radio.

```

lantran — dsa@dsasensor: ~/dsa-sensor — ssh -Y dsa@192.168.9.103 — 93x59
strInput **** HHBBBHHIIHBBHHIHHIhhHHdffiHhbb 76 80
m = 133
Report from first sensor
Delay = -0.012675 loss = 0.00 m1 = 34 m = 133
(64, 4, 120, 130, 6, 1001, 10, 170, 18000000, 70, 5, 90, 10, 13, 3, 5, 251, 11000000, -54, -50, 3, 6, 1601305540.838233, 38.8220100402832, -77.02478790283203, 9, -90, 90, -180, -119)
Report_sensing_result $$$$
(' data address = ', '\x83\x97f<{\x8c\xe5\xdc\x99\x9d\x8d\x96\xfbwsC\xd0.\x92\xad\xc0\xd3\x
a7\xcb\Cxca\xe9\xa5^t\x7f\xeb\xd5\xd7\xcd\xecB\x0wT\xda\xaf\xce\x8f\xcb\xbd\xfa\x98\x84D\x
f\x11\xfc|<\xc8\xe2\xf8\xce\xb4\x01\xcd@\xddk\x96\xa9e\xebV\xedNuj3n\xca\x03', ('192.168.9.10
5', 5005))
(' cyber data = ', '\x00@\x00\x04x\x82\x8c\x03\xec\x00\x00\x00\xa0\x00\x00\x00\xaa\x01\x12\x
a8\x80\x00\xbe\x050\x00\n\x00\r\x00\x00\x00\x03\x00\x05\x00\xfb\x00\xa7\xd8\xc0\xff\xca\xff\x
e\x00\x03\x00\x06A\xd7\xdc~\xf15\xa5\xcaB\x1b@\xb3\xc2\x9a\x04{\x00\x00\x00\t\xff\xa6\x00Z\xff
L\x88\x00\x00\x00\x00')
strInput **** HHBBBHHIIHBBHHIHHIhhHHdffiHhbb 76 80
m = 134
Report from fourth sensor
Delay = -0.013928 loss = 0.00 m4 = 34 m = 134
(64, 4, 120, 130, 140, 1004, 160, 170, 18000000, 190, 5, 48, 10, 13, 3, 5, 251, 11000000, -54, -50, 3, 6, 1601305540.838244, 38.81318283081055, -77.00875091552734, 9, -90, 90, -180, -120)
Report_sensing_result $$$$
(' data address = ', '\xf5\xd3\xd76\xdcy-0\xeb\x01J}F]KZ\xc2_K\x1a&\xedQ?;\x1d\xd8c\n7\x89D8
\xc4\xa4\xe3\xa2\xed\t\xcbCo\xaa\xd3d\xb7\x00l5@\x90\x05\xffB\xb6e#\xe3\x89\x91|\xb2\x1b\x9dC
k\x079[\xc7\x06\x8b'\x1e\x1r\x9c\x04\x0c', ('192.168.9.106', 5005))
(' cyber data = ', '\x00@\x00\x04x\x82\x8c\x03\xeb\x00\x00\x00\xa0\x00\x00\x00\xaa\x01\x12\x
a8\x80\x00\xbe\x050\x00\n\x00\r\x00\x00\x00\x03\x00\x05\x00\xfb\x00\xa7\xd8\xc0\xff\xca\xff\x
e\x00\x03\x00\x06A\xd7\xdc~\xf17nBB\x1bIJ\xc2\x9a\x14\xab\x00\x00\x00\x06\xff\xa6\x00Z\xffL
x88\x00\x00\x00\x00')
strInput **** HHBBBHHIIHBBHHIHHIhhHHdffiHhbb 76 80
m = 135
Report from third sensor
Delay = -0.009289 loss = 0.00 m3 = 34 m = 135
(64, 4, 120, 130, 140, 1003, 160, 170, 18000000, 190, 5, 48, 10, 13, 3, 5, 251, 11000000, -54, -50, 3, 6, 1601305540.860001, 38.821571350097656, -77.04036712646484, 6, -90, 90, -180, -120)
Value state machine = 6
update key counter = 4
Wait for sensor report message
**** New key = 'KaOnCRKxATjVbKGg, AES.MODE_CBC, 'txDMvj1PjWrBlRbb' 1601305540.87
Report_sensing_result $$$$
(' data address = ', '\x94\xf6\xc1\xd1u\xca\xbb\xa0C\xe0\xf9A3j\x0f\xba\xd0m\r\x13\r\x0b3\xfc
\xc9r\x87\x91\xbfG\nHp\x0b1Q7\x14\x9b\xe4\xe1n\xdb\xc2\x08\xde\x9bS\t\x8c\x97R\x0f6Y\xdb\xd3\x
f7\x0fZ\xefJ\x0d9_\xf0\x0b5\x980\x86\x0c\x920\xfd"\xbbb>\xe2\xe3\xd1\x19\xee\x99', ('192.168.9.1
02', 5005))
(' cyber data = ', '\x00@\x00\x04x\x82\x8c\x03\xea\x00\x00\x00\xa0\x00\x00\x00\xaa\x01\x12\x
a8\x80\x00\xbe\x050\x00\n\x00\r\x00\x00\x00\x03\x00\x05\x00\xfb\x00\xa7\xd8\xc0\xff\xca\xff\x
e\x00\x03\x00\x06A\xd7\xdc~\xf17nBB\x1b7\x0c2\x9a\n\x0b4\x00\x00\x00\x06\xff\xa6\x00Z\xffL\x8
8\x00\x00\x00\x00')
strInput **** HHBBBHHIIHBBHHIHHIhhHHdffiHhbb 76 80
m = 136
Report from second sensor
Delay = -0.011356 loss = 0.00 m2 = 34 m = 136
(64, 4, 120, 130, 140, 1002, 160, 170, 18000000, 190, 5, 48, 10, 13, 3, 5, 251, 11000000, -54, -50, 3, 6, 1601305540.860001, 38.804168701171875, -77.02090454101562, 6, -90, 90, -180, -120)
Report_sensing_result $$$$
(' data address = ', '\x83\x97f<{\x8c\xe5\xdc\x99\x9d\x8d\x96\xfbwsC\xd0.\x92\xad\xc0\xd3\x

```

Fig. 11 – Radio Display showing Multi-Sensor Communications with Message Security



Fig. 12 shows a screenshot at the radio interface where the Wireshark utility is used to capture and analyze network packets. The Wireshark window in Fig. 12 was used to verify that every message was successfully encrypted, similar to the results shown previously in Fig. 10. The multiple IP addresses shown demonstrate multi-sensor communications in action. In the same figure, it also shown that the contents are encrypted.

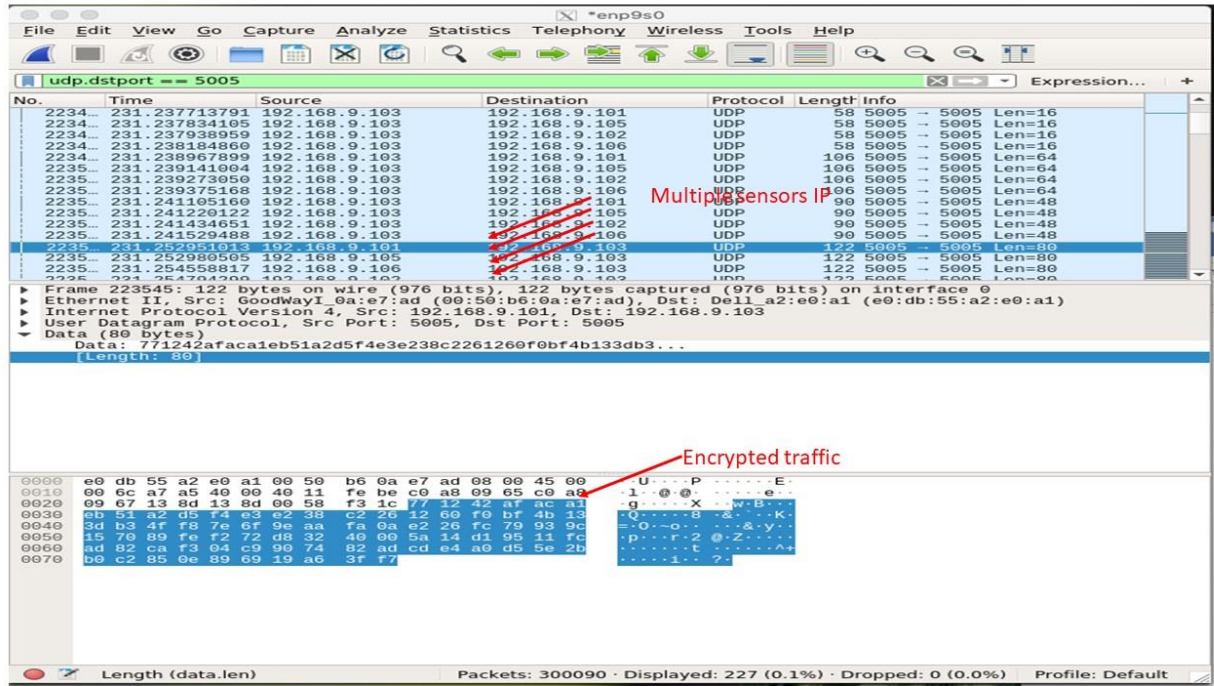
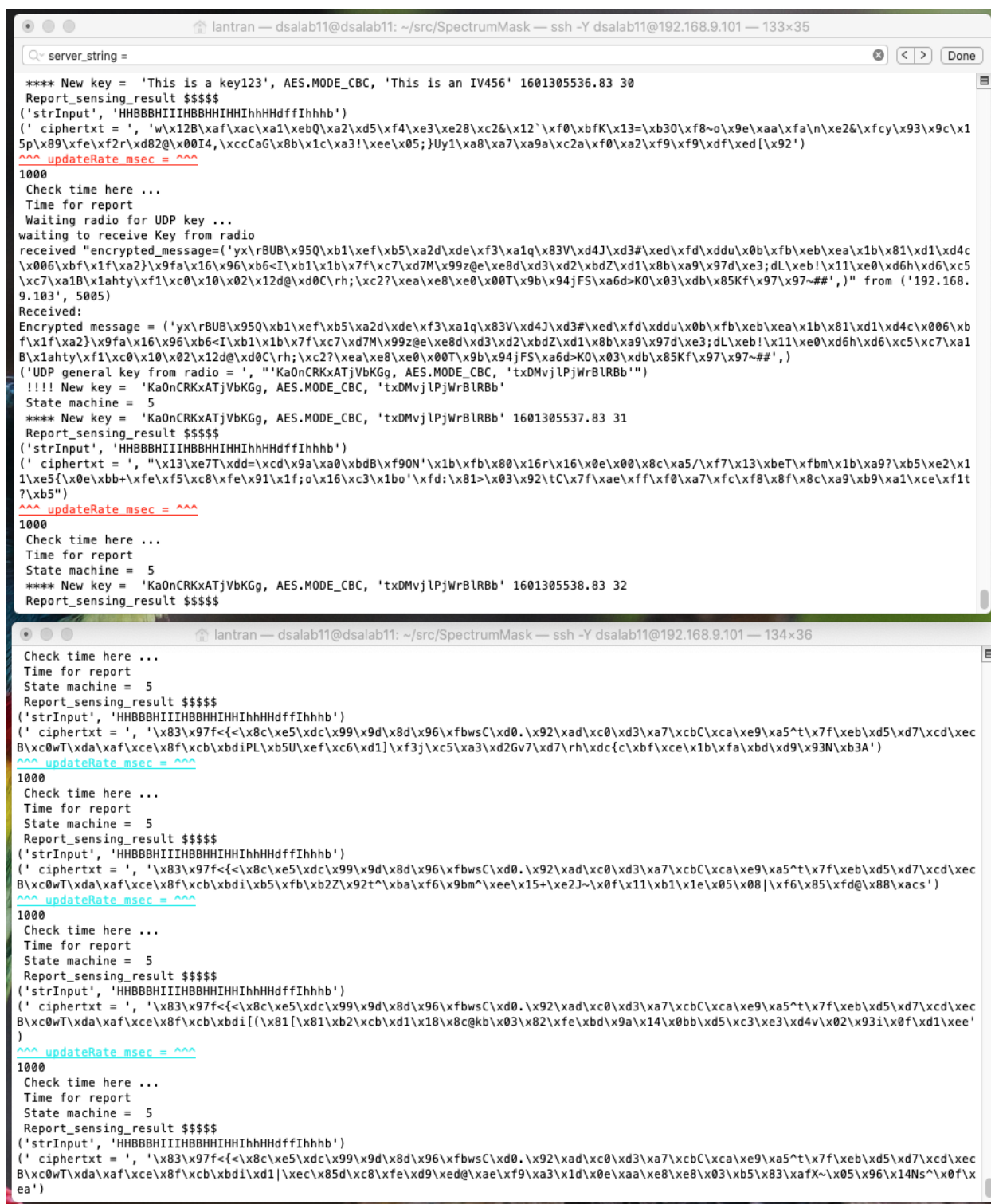


Fig. 12 – Wireshark Capture on Radio showing Multi-Sensor Communications with Message Security

Fig. 13 and Fig. 14 are screenshots showing a terminal window for each of the four sensors. Each sensor is identifiable by a unique ID and a unique color of text so that the message can be easily identified during development. The color scheme matches the color scheme shown previously in the radio terminal windows: Sensor 1 (red), Sensor 4 (cyan), Sensor 3 (pink) and Sensor 2 (purple).

Each of the figures shows the sensor checking for time and sending the information back periodically (as a Publish Response message specified in 2.6.7). These figures also show the sensory information to be sent to the radio before and after encryption. Before encryption is noted with “strInput” and after the encryption is noted with “ciphertxt”. In the upper part of the Fig. 13, the red sensor (1), also shows the key exchange process with the radio noted with “New Key” as specified in 2.6.3.



The image displays two terminal windows from a remote session. The top window shows a sequence of events for a sensor, including key generation, reporting, and receiving encrypted data. The bottom window shows a similar sequence, focusing on the receipt and processing of encrypted messages.

```

lantran — dsalab11@dsalab11: ~/src/SpectrumMask — ssh -Y dsalab11@192.168.9.101 — 133x35
server_string =

**** New key = 'This is a key123', AES.MODE_CBC, 'This is an IV456' 1601305536.83 30
Report_sensing_result $$$$
('strInput', 'HHBBBHHIIHBBBHHIHHHHHdfffIhhhb')
(' ciphertxt = ', '\x12B\xaf\xac\xal\xebQ\xad\x5\x4\x3\x28\x26\x12'\xf0\xbfK\x13=\xb30\xf8~o\x9e\xaa\xfa\n\xe2&\xfcy\x93\x9c\x1
5p\x89\xfe\x2r\x2d\x82@\x00I4,\xccCaG\x8b\x1c\xa3!\xee\x05;}Uy1\xa8\xa7\xa9a\xc2a\xf0\xa2\xf9\xf9\xdf\xed[\x92')
^^^ updateRate_msec = ^^
1000
Check time here ...
Time for report
Waiting radio for UDP key ...
waiting to receive Key from radio
received "encrypted_message=('y\rBUB\x95Q\xb1\xef\xb5\xa2d\xde\xf3\xa1q\x83V\xd4J\xd3#\xed\xfd\xddu\x0b\xfb\xeb\xea\x1b\x81\xd1\xd4c
\x006\xbf\x1f\xa2}\x9fa\x16\x96\xb6<I\xb1\x1b\x7f\xc7\xd7M\x99z@e\xed\x8d\x3\x2d\xbdZ\xd1\x8b\xa9\x97d\xe3;dL\xeb!\x11\xe0\xd6h\xd6\xc5
\xc7\xa18\x1ahty\xf1\xc0\x10\x02\x12d@\xd0C\rh;\xc27\xea\xe8\xe0\x00T\x9b\x94jFS\xa6d>K0\x03\xdb\x85Kf\x97\x97~##',)" from ('192.168.
9.103', 5005)
Received:
Encrypted message = ('y\rBUB\x95Q\xb1\xef\xb5\xa2d\xde\xf3\xa1q\x83V\xd4J\xd3#\xed\xfd\xddu\x0b\xfb\xeb\xea\x1b\x81\xd1\xd4c\x006\x
f\x1f\xa2}\x9fa\x16\x96\xb6<I\xb1\x1b\x7f\xc7\xd7M\x99z@e\xed\x8d\x3\x2d\xbdZ\xd1\x8b\xa9\x97d\xe3;dL\xeb!\x11\xe0\xd6h\xd6\xc5\xc7\xa1
B\x1ahty\xf1\xc0\x10\x02\x12d@\xd0C\rh;\xc27\xea\xe8\xe0\x00T\x9b\x94jFS\xa6d>K0\x03\xdb\x85Kf\x97\x97~##',)
('UDP general key from radio = ', '"KaOnCRKxATjVbKGg, AES.MODE_CBC, 'txDMvjLPjWrB1RBb'")
!!!! New key = 'KaOnCRKxATjVbKGg, AES.MODE_CBC, 'txDMvjLPjWrB1RBb'
State machine = 5
**** New key = 'KaOnCRKxATjVbKGg, AES.MODE_CBC, 'txDMvjLPjWrB1RBb' 1601305537.83 31
Report_sensing_result $$$$
('strInput', 'HHBBBHHIIHBBBHHIHHHHHdfffIhhhb')
(' ciphertxt = ', '\x13\xe7T\xdd=\xcd\x9a\xa0\xbdB\x90N'\x1b\xfb\x80\x16r\x16\x0e\x00\x8c\xa5/\xf7\x13\xbeT\xfbm\x1b\xa97\xb5\xe2\x1
1\xe5\x0e\xbb+\xfef\x5\x80\xfe\x91\x1f;o\x16\xc3\x1bo'\xfd:\x81>\x03\x92\tC\x7f\xae\xff\xf0\xa7\xfc\x8f\x8c\xa9\xb9\xa1\xce\x1f1t
7\xb5")
^^^ updateRate_msec = ^^
1000
Check time here ...
Time for report
State machine = 5
**** New key = 'KaOnCRKxATjVbKGg, AES.MODE_CBC, 'txDMvjLPjWrB1RBb' 1601305538.83 32
Report_sensing_result $$$$

lantran — dsalab11@dsalab11: ~/src/SpectrumMask — ssh -Y dsalab11@192.168.9.101 — 134x36
Check time here ...
Time for report
State machine = 5
Report_sensing_result $$$$
('strInput', 'HHBBBHHIIHBBBHHIHHHHHdfffIhhhb')
(' ciphertxt = ', '\x83\x97f<{\xc8c\xe5\xdc\x99\x9d\x8d\x96\xfbwsC\xd0.\x92\xad\xcd\x03\xa7\xcbC\xca\x9e9\xa5^t\x7f\xeb\xd5\xd7\xcd\xec
B\xcd0wT\xda\xaf\xce\x8f\xcb\xbd1PL\xb5U\xef\xcc6\x1d]\xf3j\x5\xa3\x2d6v7\x7d\rh\xdc{\c\xbf\xce\x1b\xfa\xbd\x9d\x93N\xb3A')
^^^ updateRate_msec = ^^
1000
Check time here ...
Time for report
State machine = 5
Report_sensing_result $$$$
('strInput', 'HHBBBHHIIHBBBHHIHHHHHdfffIhhhb')
(' ciphertxt = ', '\x83\x97f<{\xc8c\xe5\xdc\x99\x9d\x8d\x96\xfbwsC\xd0.\x92\xad\xcd\x03\xa7\xcbC\xca\x9e9\xa5^t\x7f\xeb\xd5\xd7\xcd\xec
B\xcd0wT\xda\xaf\xce\x8f\xcb\xbd1\xb5\xfb\x22\x92t^\xba\xfb\x9bm^xee\x15+\xe2j~\x0f\x11\xb1\x1e\x05\x08|\xf6\x85\xfd@\x88\xacs')
^^^ updateRate_msec = ^^
1000
Check time here ...
Time for report
State machine = 5
Report_sensing_result $$$$
('strInput', 'HHBBBHHIIHBBBHHIHHHHHdfffIhhhb')
(' ciphertxt = ', '\x83\x97f<{\xc8c\xe5\xdc\x99\x9d\x8d\x96\xfbwsC\xd0.\x92\xad\xcd\x03\xa7\xcbC\xca\x9e9\xa5^t\x7f\xeb\xd5\xd7\xcd\xec
B\xcd0wT\xda\xaf\xce\x8f\xcb\xbd1|\x81[\xb1\xb2\xcb\x1d1x18\x8c\xkb\x03\x82\xfe\xbd\x9a\x14\x0bb\xd5\x83\xe3\x2d4v\x02\x93i\x0f\x1d1\xee'
)
^^^ updateRate_msec = ^^
1000
Check time here ...
Time for report
State machine = 5
Report_sensing_result $$$$
('strInput', 'HHBBBHHIIHBBBHHIHHHHHdfffIhhhb')
(' ciphertxt = ', '\x83\x97f<{\xc8c\xe5\xdc\x99\x9d\x8d\x96\xfbwsC\xd0.\x92\xad\xcd\x03\xa7\xcbC\xca\x9e9\xa5^t\x7f\xeb\xd5\xd7\xcd\xec
B\xcd0wT\xda\xaf\xce\x8f\xcb\xbd1d1|\xec\x85d\x8c\xfe\x9d\xed\xae\x9f\x9a3\x1d\x0e\xaa\xe8\xe8\x03\xb5\x83\xafX~\x05\x96\x14Ns^\x0f\x
ea')

```

Fig. 13 – Display on Sensor 1 and Sensor 4





Fig. 14 – Display on Sensor 2 and Sensor 3

## 4. NEXT PHASE OF DEVELOPMENT

Building upon the work documented in this report, the next phases of development include establishing a more enhanced simulation environment, integration with a real-time emulation framework with realistic scenarios, and deployment on a prototype reference hardware implementation. Additionally, further enhancements to the messaging scheme and communications protocols have been identified and will be implemented.

### 4.1 Messaging and Protocol Enhancements

In the process of designing and implementing the new features described in this document, additional avenues of improvement were discovered to augment the radio capabilities, especially with the intention of deployment to large-scale DSA networks.

#### 4.1.1 Multicast Addressing with Embedded UDP Keys

The key exchange process can be further enhanced to use multicast UDP data streams. This aids in network traffic optimization by using a multicast addressing approach where the same data stream can tag the data and the key with different headers. This innovation is a safe mechanism to distribute UDP-keys alongside the data since the UDP-key can only be decrypted by intended recipients (sensors). It can reduce the repeated transmission of unicast traffic if the sensors are located within one hop distance to the radio. This scheme is elegant in its simplicity as a mechanism to distribute keys without the large burden of key maintenance, which is suitable for a network of diverse DSA-enabled radio terminals with system processing and storage limitations.

#### 4.1.2 New Entrant Discovery

The current method for radios and sensors to connect, as described in this report, is by using pre-coordinated IP addresses via an Aggregation Sensors List [3]. In many circumstances, new sensors may come into the picture without prior arrangement with the DSA-enabled radio. It is impractical, and likely impossible, to configure all known sensor addresses beforehand and coordinate this information across all possible DSA-enabled radios in the network. A method for the radio to discover new sensors is a necessary capability for the future, especially in large scale networks containing high numbers of radios and sensors. There are various ways to handle new entrant sensors and radios into a DSA network. The new entrant discovery mechanism differs from the reconnection and rejoining mechanisms described in 2.6.1 Keep-Alive. A few promising solutions are briefly described in the following sub-sections.

##### 4.1.2.1 Multicast Hello

The Hello-Request packet can be modified to be sent with multicast addressing so that the sensors within one hop distance can send the Hello-Response message. Due to the fact that most of the routers and radios do not forward multicast, one hop multicast is the best result that can be achieved in the near term, without employing more complex technical solutions. Using multicast can be seen as discovering an unknown unicast IP by leveraging a known multicast group. If the radio doesn't know the unicast IP address of a sensor, it can use the known multicast group address to send Hello-Request packets to all multicast group members. Previously undiscovered sensors, who are subscribed to the group, can reply to the radio via unicast to provide their unicast IP addresses.

#### 4.1.2.2 *Sensor Neighbor Address Sharing*

The sensor and radio can also exchange their lists of known sensors (the Aggregation Sensors List) so that once a radio connects to the sensor, more sensors can be shared with the radio. The sensor can send its list of sensors to the radio, where the radio can update its list with the new information.

#### 4.1.3 *Identity Management for Network Participation*

An Identity Database is already implemented and described in 2.6.1. Here, more consideration is given to those radios without physical layer protection, such as radios without transmission security (TRANSEC) capabilities. Modern adversaries have the capabilities to perform radio Medium Access Control (MAC) address spoofing or IP address spoofing to maliciously join the DSA radio network. A hostile sensor or a radio could pretend to be a member of the community if enough packets were intercepted during the existing messaging scheme. This vulnerability can be amplified if the radio and sensor communication is carried over a public network. The current implementation accepts new users in two ways: 1) match IP with MAC address or match public keys with preloaded checklist or 2) confirmation/rejection action (as in 2.6.1). However, this is not enough to fully prevent spoofing. More work is needed to go beyond the current RSA-based method to prevent illegitimate network participants.

#### 4.1.4 *Considerations for Multi-Radio and Multi-Sensor Enhancements*

This report covers single radio's implementation that communicates to multiple sensors. If multiple radios are networked to multiple sensors, many other factors need to be considered. The UDP key dissemination will become more complex, which will increase the demand for key management as well. This problem requires more research into options such as: keeping multiple keys at the sensor, electing a key-master for the key generation, or letting the sensor generate the key. This problem will be addressed in future work.

### 4.2 **Integration and Experimentation with EMANE**

To supplement the modeling and simulation efforts, new capabilities can be evaluated in a dynamic emulation environment and integrated with modern tactical communications waveform models. The Extendable Mobile Ad-hoc Network Emulator (EMANE) [8] is an ideal emulation framework for prototyping DSA software code and examining policy effects in a realistic environment, without the requirement for physical hardware platforms. Previous work has been completed within the SFF DSA TTNT Program to enable the radio nodes with DSA policy reasoning capabilities, spectrum sensing, sensor fusion, and remote policy distribution capabilities.

The EMANE emulation framework allows for the creation of dynamic scenarios where multiple radios can be instantiated and coexist in a single environment. It can be used with its own emulated physical (PHY) layer to observe DSA policy compliance such as those with respect to boundary conditions, grace periods, policy loading times, and RPD, among others. The DSA-TTNTv7 waveform model is a classified Government Off the Shelf (GOTS) product that currently employs DSA code that can be built upon and expanded to add capabilities. Additionally, the EMANE framework allows the DSA-enabled radio terminal to leverage both emulated sensors (within EMANE) and physical sensor hardware via an Ethernet connection. Live radio frequency (RF) signals, such as those from signal generator or production-grade radio platform can be injected into the emulation via the hardware sensor and used as evidence for DSA reasoner decision making within the scenario. The EMANE integration will allow for rapid iteration of the sensor API code and communications protocol, and provides a rich framework to conduct both static and dynamic tests in a controlled emulation environment. Successful implementations can be transitioned to the Multifunctional Information Distribution System Joint Tactical Radio System Terminals (MIDS-JTRS) TTNT radios.

### 4.3 Prototype Implementation on USRP with Sensing Capabilities

Once the radio-sensor communications code and technologies have been tested and matured through modeling, simulation, and integration with EMANE, a prototype implementation shall be developed. The combination of other critical DSA functions, such as sensor interfacing [2], sensor information aggregation [2], and sensor data fusion [9], along with the enhanced DSA reasoning engine [10] will be deployed into a reference hardware implementation, potentially leveraging a platform such as the Universal Software Radio Peripheral (USRP) model X310 developed by Ettus Research. [11] The USRP X310 is a high-performance, scalable software defined radio (SDR) platform for designing and deploying next generation wireless communications systems. The USRP reference implementation will enable expanded laboratory and field testing with a configurable DSA-enabled radio platform that can be tested with other DSA-enabled devices such as the SFF DSA TTNT terminal, and other platforms currently in development. Coexistence and interoperability testing with a diverse set of DSA-enabled radios will provide valuable insights into the differences between different reasoning engines and the ways they process and react to policy rules.

## 5. CONCLUSION

This report has provided details for improving upon the existing solutions for radio-sensor communications by utilizing industry standard encryption techniques, a custom sensor message structure, and enhanced messaging protocol to provide message security. A new method has been proposed to improve the key distribution process with one single UDP data stream. In addition to the implementation of industry-standard encryption protocols, new methods are proposed further simplify the key exchange process and key management. These techniques have been demonstrated in a simulated software network consisting of one simulated DSA-enabled radio and up to four simulated sensors.

The newly enhanced interface described in [3] enables more efficient radio-sensor communications and improved data robustness for better decision making in spectrum-agile radios. The techniques described in this report provide message confidentiality, integrity, and authentication into the current DSA network paradigm through the usage of new features including RSA based key-exchange, AES key unicast/multicast encryption, a packet driven state machine, and active sensor database within the radio. The next steps and future work have been outlined and discussed in this report. The stepping stones are in place to enable researchers to rapidly iterate upon the implementation demonstrated in this report, in order to deploy these enhancements to more realistic emulation scenarios, hardware prototypes, and production radio systems.

## REFERENCES

1. Final Program Report for the Dynamic Spectrum Access (DSA) Small Form Factor (SFF) Tactical Targeting Network Technology (TTNT) Radio Terminal Program, Rockwell Collins and Shared Spectrum Company, January 2020
2. Interface Control Document (ICD) for the Dynamic Spectrum Access (DSA) for a Small Form Factor (SFF) Tactical Targeting Network Technology (TTNT) Radio External Sensor, Rockwell Collins and Shared Spectrum Company, October 2019
3. L. Tran, T. Mai, E. Makara, J. Molnar, S. Kompella, M. Dillon, "Enhanced API for Multi-Sensor Aggregation," NRL Code 5524, NRL/MR/5524--20-10,159, September 2020
4. IETF RFC 8017, "PKCS #1: RSA Cryptography Specifications Version 2.2." Available: <https://tools.ietf.org/html/rfc8017>
5. FIPS 197, "Advanced Encryption Standard (AES)", Available: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>
6. RFC 3740, "The Multicast Group Security Architecture", Available: <https://tools.ietf.org/html/rfc3740>
7. Wireshark, Available: <https://www.wireshark.org/>
8. "Extendable Mobile Ad-hoc Network Emulator (EMANE)", NRL, Available: <https://www.nrl.navy.mil/itd/ncs/products/emane>
9. M. Dillon, C. Pici, S. Kompella, J. Molnar, L. Tran, "Multi-Sensor Correlation and Fusion for Spectrum Occupancy", NRL/FR/5524-20-10,157, September 2020
10. J. Chao, E. Makara, J. Molnar, S. Kompella, "Policy Enhancement For Improved DSA Reasoning", NRL/FR/5523--20-10,402, September 2020
11. "USRP X310", Ettus Research, Available: <https://www.ettus.com/all-products/x310-kit/>
12. IETF RFC 4253, "The Secure Shell (SSH) Transport Layer Protocol", Available: <https://tools.ietf.org/html/rfc4253>
13. S. Tatham, "PuTTY", Available: <https://www.putty.org/>