IAA-CU-13-13-01
Authenticated uplink for the small, low-orbit student satellite NUTS

Bram Bezem & Per Kristian J. Fjellby*, Roger Birkeland†, Stig F. Mjølsnes‡
Norwegian University of Science and Technology (NTNU)

Abstract

The NTNU Test Satellite (NUTS) is a project run by the Department for Electronics and Telecommunications at NTNU. Its goal is for students to design, build and launch a double CubeSat by 2014. The satellite's primary function will be to capture images of the earth with an infrared camera. These images will be used for atmospheric observations. In this paper, we focus on the communication security aspect of such pico-satellites, especially message authentication and replay attacks. Without any means to protect against such attacks, malfeasors with the right equipment can record and replay messages to the satellite. If the command set is known, either through reverse engineering or publication, malfeasors can even generate and send legitimate commands at will. The NUTS project has chosen the CubeSat Space Protocol (CSP) as main communication protocol. This protocol lacks a replay attack prevention mechanism. We propose a scheme to improve CSP with such a mechanism. A sequence number together with a hash-based message authentication code (HMAC) is used to provide resistance against message replay attacks. A resynchronization mechanism is also implemented to account for any drifts in the sequence numbers between the satellite and ground station. We have made test-applications based on the publicly available CSP source. Using these applications, the design has been tested on a realistic, full-duplex radio link between two computers. The tests showed that the functionality added to CSP worked as designed both when no faults were introduced and when either of the nodes were reset to force a resynchronization.

*NTNU Test Satellite Group. Norway, {bezem,perkrifj}@stud.ntnu.no
†Project manager, NTNU Test Satellite. Norway, roger.birkeland@iet.ntnu.no
‡Supervising Professor, Department of Telematics. NTNU. Norway

Introduction

A typical satellite antenna system involves highly directional antennas, directing most of the radio waves toward the satellite, and not the surroundings. Recording the communication between the ground station and satellite from a location close to the antenna is still feasible. Suppose that this communication includes a message with a command instructing the satellite to take five pictures. A recorded message can be retransmitted at a later time, and if the satellite is within reach it will happily take five more pictures. This process can continue and lead to resource starvation, i.e. draining the battery or filling up the memory. It is clearly beneficial to prevent these replays and to establish some understanding of who the communicating parties are, thereby restricting access to legitimate parties.

Challa et al. as well as earlier thesis work done with the NUTS project observe that previous CubeSat projects have focused more on redundancy checks and error correction as opposed to security and authentication [4, 2]. Our starting point is the CubeSat Space Protocol (CSP), which has been chosen as the backbone protocol for communications in the NUTS mission. Ultimately, everything we propose has to fit nicely into this software suite which will be running on both the satellite and the ground station.

The NUTS project NUTS is a project run by the Department for Electronics and Telecommunications at NTNU. The goal is for students to design, build and launch a CubeSat by 2014 [1]. The satellite will be a double CubeSat and its primary function will be to capture images of the earth with an infrared camera. These images will be used for atmospheric observations.

Background

CubeSat Space Protocol

CSP is a "small network-layer delivery protocol designed for CubeSats.." [3]. It is a Danish initiative out of Aalborg University and is today integrated with `GomSpace`[1] products. The CSP core is available under a GNU Lesser General Public Licence (LGPL) on `github` under `gomspace/libcsp`.

CSP is a layered protocol, providing transport-, routing-, and interface-layers. The header is 32 bits and consists of the following fields:

PRIO(2)	SRC(5)	DST(5)	SPORT(6)	DPORT(6)	FLAGS(8)

Table 1: CSP header structure. Number of bits in parentheses.

5 bit source- and destination fields gives room for 32 unique subsystems, with 6 bit port numbers for interfacing with sub processes.

CSP provides a foundation for authenticated communication via HMAC and SHA-1 implementations. HMAC, using SHA-1 as the underlying cryptographic hash function, provides authentication via a pre-shared key.

[1] GomSpace is a company established by contributors to the AAU-CubeSat mission from 2003.

Related Work

Challa et al. makes the observation that CubeSat builds are heavily resource-constrained which leaves little room for complex security suites [4]. They propose a security scheme which they have called *CubeSec* and *GndSec* that reside in the satellite and the ground station respectively. The authors claim a low-cost, low-weight, low-power, and small form factor realization of the *CubeSec* subsystem and achieving mutual authentication, confidentiality and integrity between satellite and ground station using pre-shared keys.

Problem Formulation

A replay attack is a retransmission of a previously captured command or message at a later time. *Timeliness* is a key word in this setting. Commands that are sent to a system at a time when they are not supposed to can severely interfere with a communication system, and if not handled correctly it can even bring the system to a halt. The nature of wireless communication makes eavesdropping and recording of communication between the ground station and satellite a starting point for attackers that want to compromise or abuse our system. A way to prevent these types of attacks is sought for the NUTS CubeSat mission. Since no such replay mechanisms are available for the CSP library we want to extend it to add such protection. We will also analyse and set up a test system with two radios.

Proposed solution

We considered two options for adding replay protection; sequence numbers and timestamps. When using the the first, every data packet contains a sequence number and both communicating parties keep a local counter representing this number. When a party sends a data packet, the sequence number is added to the packet and the counter is increased. At the receiver end, the number is checked against the locally stored number. The requirement is that the received sequence number must be greater or equal than the one stored locally. If it is not, the packet is considered a replay and is discarded. If for some reason the two sequence numbers should become unsynchronized, a mechanism to put them back in order is needed.

Using timestamps the communicating entities rely on synchronized clocks to decide on the timeliness of a packet. Each packet contains a timestamp. If a packet is received within a predetermined threshold, the verification is successful - otherwise the packet is dropped. We expect there to be some type of time reference in the satellite to be used for picture timestamping. However, it is a well known fact that clocks can drift. To account for this, a scheme would have to be put in place to synchronize satellite and ground station clocks. To avoid the need for synchronization, GPS receivers could be used to provide accurate clocks. In fact, the entire GPS scheme is reliant upon the accuracy of clocks in the satellites [5]. The problem is that GPS modules intended for civilian use have height and speed restrictions. Thresholds we have seen indicate an upper bound on the height and speed of 18 km and a 515 m/s [6]. These requirements are too strict for most space related activities. Applying for a military-grade GPS receiver and obtaining the proper export permissions seemed like a too daunting task. All these considerations led us to base our solution on sequence numbers.

Sequence number

Our solution gives the possibility to attach a sequence number to packets over a connection. This will ensure that every message is unique and therefore any HMAC-digest will be unique (excluding collisions). The system can then recognize and reject any previously accepted message so that the retransmission of recorded authentic messages is not possible. The presence of such a sequence number is indicated by setting a flag in the header, just as it is done with HMAC, XTEA and CRC32 in the current implementation of CSP. To allow full-duplex communication, each direction of travel needs their own sequence number. This means that each node will need to maintain a list of two sequence numbers for all possible hosts. One is used when receiving packets from the host in question and the other when sending to that host. Since CSP supports up to $2^5 = 32$ hosts, a list of 64 sequence numbers needs to be maintained by each node.

Previous work on the NUTS project [2] lead to a suggested size of 16 bits which we deemed adequate, taking into consideration the approximate rate of sent packets and the expected lifetime of the satellite. A size of 16 bits allows 65536 unique packets to be sent. The sequence numbers will only be used on the uplink when sending commands, and on the acknowledgement packets for those command-packets. Depending on the application layer protocol, several commands can be sent per packet. We do not believe the average number of packets sent per pass will exceed 10 over the lifetime of the satellite. Calculating with an average of ten packets per pass and five passes per day gives us over three and a half years of operation. The lifetime is estimated to be between six months and two years, thus we consider a length of 16 bits enough with a decent safety margin. Another factor we took into consideration is the overhead our scheme would introduce. By overhead we mean metadata that must be transferred together with user data, and thereby negatively impacting the net throughput. Due to our design choice not to add it to the header, the sequence number will only add overhead when it is enabled on a connection. We assume it will only be used on the uplink where the user data consists of application layer data with commands and their arguments destined for components in the satellite. There, data throughput is not a big concern, while authentication is. The downlink is where the pictures from the satellite will be transmitted, our recommendation would be to not use sequence numbers on this connection because it will negatively impact bandwidth, which is a major concern. A lot of effort is being put into data compression and image averaging [7] in order to minimize the amount of data that has to be sent down to earth. The large data volumes would also mean that a large number of packets would be sent, thus requiring a longer sequence number. The next step up in unsigned integers would be 32 bit allowing over four billion unique packets, but having a twice as large impact on throughput.

Resynchronization

In order to make the system as reliable as possible there must be a way for nodes to resynchronize the sequence numbers. The two copies of the sequence number can become unsynchronized for several reasons. The satellite's orbit height makes it more prone to influence from cosmic and solar radiation. This influence might cause restarts or corrupt data in memory. In this case we do not want the sequence number to restart at zero, as that would allow messages from the previous sequence to be replayed. We will require that at least one of the nodes is able to store the sequence number in a reliable way. Using the resynchronization protocol it can share that number with the other node. It

is automatically initiated when a threshold for errors is met, so that is does not require user intervention. The packets are sent using the same link as the main data connections, but on a separate socket using a reserved port. This makes them distinguishable from regular packets. The packets are simple in structure merely containing the two sequence numbers concerning the hosts in question.

As part of the resynchronization process the parties are notified when they are unsynchronized. The initiating party sends their sequence numbers. The responding party makes the decision of what both parties' sequence numbers should be set to. It chooses the highest of the received number and its own copy, to ensure that the sequence number always is increasing. The initiating party also checks that the number it receives in the response is larger than the number it has stored at that time. This means a sequence number is never used twice. It is important that the sequence numbers are stored securely. A bit flip in the memory of the satellite can cause the sequence number to increase radically. For example, a bit flip in the most significant bit in the sequence number would lead to a jump of half the possible space.

Security Analysis

Attacking the MAC

Here we identify and analyse ways to attack our design. We look at the following scenarios:

1. Modify a previously sent message for an attacker to run a command of his choosing.
2. Replay a message previously sent from the ground station to the satellite.

#1 We assume that the attacker has recorded a message and has analysed it at bit level to identify the structure, or has obtained the packet structure by other means. The source code will be made public and no proprietary codecs are used on the radio, so this should be feasible. We see two major obstacles. If one alters the message contents by which the HMAC calculation is based upon, the subsequent verification in the satellite will fail. For it not to fail, the attacker would have to find an input message that produces the same HMAC tag as the original message. This is known as a collision, and if the MAC is strong, finding such collisions is computationally infeasible. If one should succeed in finding such a pair, the likelihood if it resembling the command structure of a CSP packet is highly unlikely.

#2 This attack is just a replay of a previous message, meaning that no modification to the packet is done. The packet will have a valid HMAC tag, so the sequence number check will decide if this is a replay or not. Since the satellite will require an increase in the sequence number, thus the packet will be discarded if it is a replay attempt. The attacker could however delay the transmission, i.e. by jamming the satellite so that no commands are received and thereby no increase to the satellite sequence number is made. If the ground station sends messages while the satellite is jammed and an attacker records these messages, they can be replayed after the jamming is stopped. The commands are authentic since they are sent by the ground station, but the delay could cause potential problems, for example that the camera is triggered when it should not be, or similar. Using some form of scheduling of the commands could mitigate this problem. Say that

the command instructs the satellite to take 5 pictures at a given time. Then you have limited the possibility of issuing untimely commands. If a valid command is received after it is intended to be executed it should be discarded by the scheduler.

Concluding remarks If an attacker were to get access to the keys used in the authentication scheme, the whole system is compromised. An attacker can then create messages with a sequence number that is large enough and recompute the HMAC with the key. Protecting the keys is therefore paramount.

Attacking the resynchronization

CSP sets connection options when the connection is first created. If the sequence numbers on the ground station and satellite become skewed, packets are discarded. If we were to send the resynchronization packets over the same connection, these packets would also be dropped. The way we solved this was to make both nodes listen on a port we reserved for sequence number resynchronization. When a node initiates a resynchronization, a new connection is established on the resynchronization port with the socket option of sequence number disabled. In essence this means that resynchronization packets does not contain any means to support replay detection, so previously captured resynchronization packets may be recorded and resent at a later time. The goal of this section is to go through different attack scenarios to see what an attacker has to work with in order to compromise our system through the resynchronization procedure.

We assume for this section that the attacker is in possession of a valid resynchronization request packet. This packet could be obtained by recording a transmission of such a packet from either the satellite or the ground station. It is also possible to construct such a packet bit by bit. Let us first examine the case where an attacker initiates a resynchronization by transmitting a valid resynchronization packet. The resynchronization request packet could be sent to either the ground station or the satellite. If the packet is valid, where valid in this sense means that it is a correctly structured CSP resynchronization packet, the receiving side will process the packet. The receiver will then select the sequence number with the highest numerical value as the new sequence number, based on a comparison of the sequence number received in the resynchronization request packet and the number stored locally. A response will then be sent containing the sequence number to be used. Upon receiving this message, the attacker can make the same check as the receiver and determine the new sequence number. The sequence number is now synchronized between the attacker and either the satellite or the ground station. Since the attacker does not know the shared secret key used for HMAC he is not able to send packets that will pass the message authentication check on the authenticated channel, following the same reasoning as the previous section. The fact that the attacker has interfered with the sequence number mechanism will have implications for legitimate operations when the satellite comes within radio distance of the ground station. If the sequence numbers have been altered, a new resynchronization is required in order to resume operations on the authenticated channel. If the attacker simply replays a resynchronization request he has heard before, the damage done is limited. Since the packet is a replay it is probable that the sequence number included in the resynchronization request is used before. Therefore the new chosen sequence number will remain unchanged from the ground station/satellite perspective, and no resynchronization is required for legitimate communications to resume. Things get more interesting if the attacker has the possibility of generating his own

resynchronization request, and especially inserting a sequence number of his choosing. Remember that the resynchronization scheme is based on selecting the highest sequence number as the new sequence number. If an attacker is able to introduce large jumps, the sequence number space will be used up fast. One possible solution is to disallow very large jumps in the sequence number. The problem is where to draw the line. Ultimately you could end up with a situation where the sequence numbers differ by so much that such a large jump is required in order to resynchronize, and you effectively end up with a livelock situation. The best solution we have come up with for this problem is to apply HMAC on the resynchronization link as well. An attacker could still send replays of legitimate resynchronization sequences, but not create custom requests. The replays will have no effect since the system selects the highest of the received sequence number and its own, thereby rejecting the replayed attempt. The attacker is not able to create his own request because they will fail under the HMAC verification. We have not focused on key generation and key handling, but we believe that the use of different keys for message authentication and resynchronization authentication is good practice.

Testing

The test-setup consisted of two computers and two radios. Both computers were running `soundmodem`, a software package that uses the computer's sound-card to send AX.25 packets to a radio. `soundmodem` can use a variety of modulation types including Audio Frequency-shift Keying (AFSK), which we chose for our testing purposes. A baud-rate of 1200 was used for the test-setup. We changed the `csp_rdp_packet_timeout` and `csp_rdp_ack_timeout` parameter to 3000 and 3000/4 respectively from 1000 and 1000/4, to cope with the limited link speed. The existing CSP implementation already had the required support for interfacing with the virtual serial link set up by `soundmodem`. Both radios were configured to use separate frequencies for transmission and reception, simulating the up- and down-link of the satellite.

This setup was then used to transmit packets over a channel protected by HMAC and sequence numbers, using simple test applications written by us. One was termed the 'server' - it would merely listen to incoming connections and receive packets, similar to the satellite waiting for and receiving commands. The other was termed the 'client' - it would attempt to connect, if successful send several packets of dummy data and disconnect. Both programs would start using sequence number 0 every time they are stopped and restarted.

We had four different scenarios we wished to test over the full-duplex radio.

1. *Normal use of sequence numbers, by running both programs over a period of time.*
   Sequence number were successfully added to both on the data packets from the client to the server and the ACK packets in return. Verification by the receiver succeeded, and all counters were increased as they should.

2. *Resynchronization by letting them run and then restart the 'client' so it suddenly loses its sequence number.*
   First the communication proceeded as desired like in scenario 1. Then once the client was reset, the new packets it sent were rejected as they did not have the expected sequence number. The server notified the client after the configured threshold was reached. The communication continued after a successful resynchronization of the sequence numbers.

3. *Resynchronization by letting them run and then restart the 'server' so it suddenly loses its sequence number.*
   The same as scenario 2 except the server was reset.

4. *Resynchronization when both sides lose the sequence number.*
   Again, the nodes behaved as expected. Communication was resumed without problems but with all counters at zero after both nodes were reset, this is as expected since the test-applications were designed to restart at zero.

Further work

As we previously explained the sequence numbers need to be stored securely to survive reboots and cosmic radiation while in space. An infrastructure to manage and provide secure storage of the numbers has to be designed and implemented. This is also important for the cryptographic keys, but these can be stored in read-only memory (ROM) since they will not change. ROM is more resistant to influence from external forces since it is designed for a single write. Simply sending up several back-up keys that can be used if the others are compromised add extra security at the cost of more permanent use of memory. The sequence numbers change regularly and can therefore not be stored in ROM. An option might be to make copies and use a voting system to determine which are valid.

Cryptographic key management is another issue. There needs to be a way to use different keys for different purposes, like a particular key only used for resynchronization. These then can be switched with other securely stored keys in case an attacker manages to copy them. Alternatively, methods for generating temporary keys to be used for HMAC in both resynchronization and normal operation could be used. Then there is one master-key not in general use, but only for key derivation/generation.

Conclusion

The issue of replay attacks is not new to the communication security world, but as our paper points out, the field of Cube-Satellites have had more focus on error correcting codes and redundancy checks. In this paper we have described the steps we have taken to extend CSP with replay protection - through sequence numbers, message authentication algorithms, and resynchronization. We have looked into what security it provides and implemented a proof-of-concept prototype on the `libcsp` library.

Further work is still required before the system can be classified as flight-ready, but we believe that our design provides a strong base to resist replay attacks.

References

[1] Roger Birkeland. *NUTS Mission Statement*. June 2011. http://nuts.cubesat.no/upload/2012/01/20/nuts-1_mission.pdf

[2] Sandesh Prasai. *Access control of NUTS uplink*. Master's Thesis, Norwegian University of Science and Technology. July 2012.

[3] Gomspace, *Network-Layer delivery protocol for CubeSats and embedded systems*. http://www.gomspace.com/documents/GS-CSP-1.1.pdf. June 28, 2011

[4] Obulapathi N. Challa, Gokul Bhat, Dr. Janise Mcnair, *CubeSec and GndSec: A Lightweight Security Solution for CubeSat Communications*. University of Florida, Department of Electircal and Computer Engineering. 26^{th} Annual AIAA/USU Conference on Small Satellites.

[5] G. Xu, *Gps: Theory, Algorithms and Applications*. Springer Verlag, 2003.

[6] M. D'Errico, *Distributed Space Missions for Earth System Monitoring*. Space Technology library. Springer, 2013.

[7] M. Bakken, *Signal processing for communicating gravity wave images from the NTNU Test Satellite*. Master's Thesis, Norwegian University of Science and Technology. July 2012