Development of a Security Module for the Uplink of the NUTS Student Satellite

Marius Münch [(1)], Roger Birkeland [(2)]

[(1)] Master Student, Norwegian University of Technology and Science
Department of Telematics
O.S. Bragstads plass 2a
N-7491 Trondheim
Norway
Phone: +49 1520 9550988 – mariusmu@stud.ntnu.no

[(2)] PhD Student, Norwegian University of Technology and Science
Department of Electronics and Telecommunication
O.S. Bragstad plass 2a
N-7034 Trondheim
Norway
Phone: +47 73594303 - roger.birkeland@iet.ntnu.no

ABSTRACT

The Norwegian University of Science and Technology initiated a CubeSat project in 2010 under the name NTNU Test Satellite. The main goal of the NUTS project is to build a double CubeSat entirely from scratch including both hardware and software. The launch is planned for 2015 and the development of the satellite is still ongoing.
One specialty of the NUTS project is a focus on the communication security of the satellite. In order to prevent malicious use or hijacking of the satellite, authentication mechanisms for the communication between ground station and satellite are required. An authentication procedure based on Keyed-Hash Message Authentication Codes is appropriate. However, due to the threat of replay attacks, transmitted packets have to contain a fresh element.
This paper outlines a design for a modular security solution which is independent from other communication protocols used by the satellite. It is demonstrated that timestamps are a valuable option as unique element. The on board real-time clock on the satellite is used as master clock in the system. The introduction of timestamps in the system also enables session management as a complementary feature. Additionally the development of a proof of concept implementation for the proposed design is described.

1 INTRODUCTION

The NTNU Test Satellite (NUTS) project of the Norwegian University of Science and Technology (NTNU) aims to build a double CubeSat from scratch. The payload includes a camera taking pictures the Earth's atmosphere.
One additional objective of the project is to secure the uplink in order to prevent malicious use or hijacking of the satellite. The intuitive way of using encryption for the operational link is not viable due to the utilization of amateur radio frequencies for the communication on which encryption is forbidden [1]. Instead of encryption, authentication mechanisms are sufficient enough in order to secure the uplink as pointed out by [2]. Thus, keyed-hash message authentication codes (HMAC) are used as authentication mechanism for the NUTS project.
The principle idea is to append an authentication code to every message which is constructed by hashing the message together with a secret key known only by the ground station and the satellite. This approach is already implemented in the CubeSat Space Protocol (CSP) [3] which is used in the protocol stack of NUTS.
Unfortunately, the communication is still prone to replay attacks where an attacker records a sent message and replays it later on. In order to avoid this kind of attacks, recent work inside the NUTS project was dedicated to expand the CSP with sequence numbers delivering a fresh element in every message [4]. This introduced the need of a resynchronization protocol and thereby additional overhead in the communication.
This paper outlines an alternative solution which does not rely on the presence of CSP and reduces the communicational overhead. In particular, a modular security solution which uses timestamps in order to guarantee freshness for every message is presented.
The following section provides an overview over the whole system of the NUTS satellite with focus on the communicational part. The design of the proposed security solution is described in Section 3. Section 4 outlines the possibility of session management and Section 5 provides information to a proof of concept implementation. Afterwards the paper finishes with concluding remarks.

2 SYSTEM OVERVIEW

The NUTS satellite is divided into six modular components: The on-board computer (OBC), the attitude determination and control system (ADCS), a real-time clock (RTC), the electronic power system (EPS) the payload and the radio module. These components are interconnected with an I²C bus. Since the radio module works as gateway between ground segment and the satellite, it implements core routing functions. Thus, the radio requires an own microcontroller unit (MCU) for this crucial task. Beside this, the radio consists of a beacon and two transceivers communicating on the 2-meter and 70-centimeter amateur radio band. The beacon broadcasts status information via Morse code and one of the transceivers is only designated for downlink controlled by the OBC. The logical communication structure is visualized in Figure 1.
The protocol stack itself is structured in three segments. AX.25 serves as link layer protocol between base station and the satellite in the actual design but might get exchanged to a self-developed protocol in a later revision. For addressing the different modules of the satellite, CSP is used as routing protocol inside the satellite. It supports both connectionless and connection oriented operation and serves thus as both network and transport protocol. The top layer contains the application data and thereby the specific commands for the satellite. The command format itself is not defined yet and for the sake of simplicity just referred to as NUTS Messages.

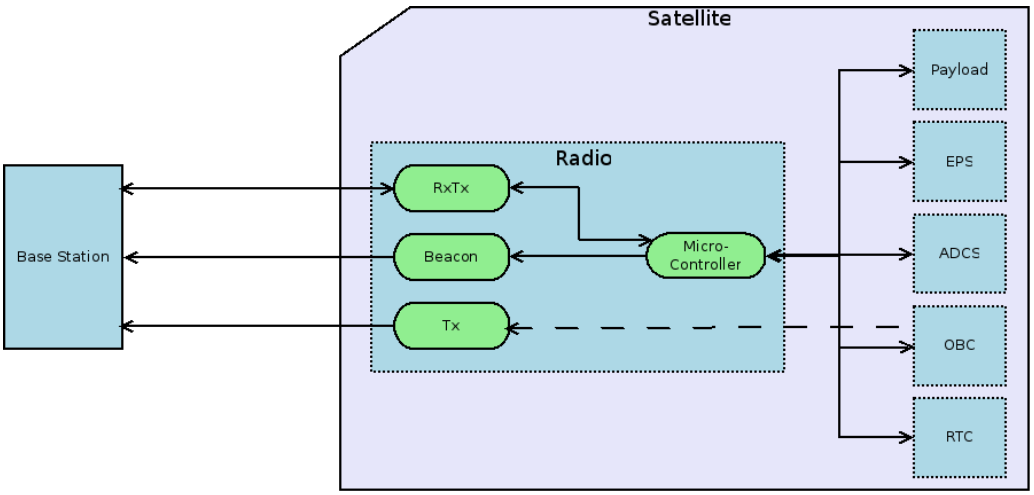

Figure 1. Schematic overview of the communication structure

3 DESIGN OF THE SECURITY MODULE

This section has two purposes. On the one hand, the design process itself is mirrored in a bottom-to-top approach. On the other hand, we support the design decision with reasons.
The overall focus of the design lies on the security functionalities for the satellite and the solution is custom-made for the system described in the previous section. Thus, the main parts of the security solution are settled down in the satellite and specifications for the base station are kept minimalized.

3.1 Design Goals

In order to develop a suitable security solution design goals have to be clearly specified. The top level goal is to provide a secured uplink which is only accessible for authenticated parties. More specifically, it is desired that the ground station authenticates itself to the satellite. This shall be realized with the help of HMACs. Additionally, every message has to contain a unique, and thus, fresh element in order to prevent replay attacks.
As second level goal the special challenges resulting from the space environment should be taken into account for the solution. This implies that the communicational overhead resulting from the security functionalities shall be minimized in order to keep the link budget small. Furthermore, the designed solution should be small, simple and computational cheap for the satellite side since it has to run efficiently on the satellites MCU. In addition to that, the possibility of bit flips induced by cosmic radiation has to be kept in mind.
Moreover, a not formal specified goal is mentioned at this point. An optimal security solution would be as resistant as possible to changes in the communicational part of the system since the development of the NUTS satellite is still on-going.

3.2 Placement of the Security Functionalities

One of the most critical design decisions is where to locate the security functionalities. The term location is ambiguous here and refers to both the satellite modules and the protocol stack.
The choice of the module is rather easy since the radio MCU is a bottleneck. It has to validate all incoming frames and implements the full protocol stack. Thus, it is reasonable to integrate the security functionalities inside the radio MCU in order to have a centralized solution. This approach simplifies the implementation since only a single module inside the satellite has to be altered.

For the placement in the protocol stack two fundamental approaches are coming into consideration. The information for the security solution could be either placed inside one of the used protocols as enhancement or in the form of an own protocol between the existing layers.
The first approach is rather difficult to handle. The used protocols are AX.25, CSP and the NUTS messages. The enhancement of AX.25 is rather inadvisable due to the ground segment. It is firmly integrated in the Linux-kernel which implies that kernel modifications for the base station would be necessary. In addition to that, AX.25 is widely used and a modification here would eliminate the possibility to listen to other amateur radio based satellites.
An extension to CSP appears reasonable at the first glance since it provides already HMAC functionalities. Thus, only a fresh element would have to be added in the protocol itself to deal with the threat of replays attacks. Unfortunately, this solution brings problems due to the modularity of the NUTS satellite system itself. Either every module would have to run a modified version of CSP or both the radio MCU and the ground segment would have to share a customized CSP interpreter which adds and removes the security related information. Both solutions would toughen up the general code maintenance inside the project. In addition to that it would be more recommendable to contribute to CSP itself instead of altering the code of a third party project.
Finally, NUTS messages are not defined yet and a requirement for security information is inappropriate since the radio module does not have to touch the application layer for messages designated to other modules. An enhancement here would only be reasonable for a decentralized security solution which would be against the decision of a solution integrated inside the radio MCU.
Consequently, the presented security solution makes use of an own security layer. The most reasonable location for this is between AX.25 and CSP due to the fact that the radio operates with both of them while the other modules only implementing CSP and the NUTS messages. This means that the security solution can be modular and is only responsible for validating the authenticity of incoming frames and removing the corresponding header. The logical structure of a received frame is shown in Figure 2 in order to visualize the altered protocol stack.

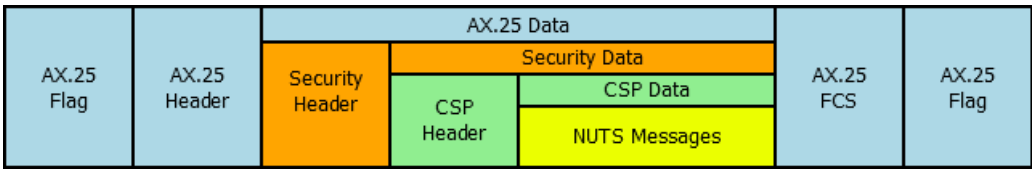

Figure 2. Logical view of frames inside the radio MCU

A minor benefit resulting from the modularity of such a security solution is that it is independent from the other protocols. Additionally, it could be discarded without major effort in the case that the general NUTS specifications are changing.

3.3 Fresh Element

Every message has to contain a fresh element in order to prevent replay attacks. The first consideration was the usage of sequence numbers for the NUTS project. These are numbers increasing with every sent message. However, this requires additional resynchronization mechanisms since satellite and base station can easily get out of synchronization due to link errors or bit flips caused by cosmic radiation. Hence, the overall system gets rather complicated.
In order to demonstrate a possible alternative, we developed a system making use of timestamps. The most challenging part hereby is to have synchronized clock in both ground segment and satellite. It is not feasible to use an extern master clock due to the satellite's location in space [3]. Thus, either the ground segment or satellite clock has to serve as master clock for the system. Since it is easier to adjust the time inside the base station than in the satellite, it makes sense to utilize the

RTC of the satellite as master clock.
However, the base station has to know the satellite time. Therefore, the satellite has to transmit its time in a reliable way. The proposed solution to this is to include the satellite time inside the beacon status information. Since the beacon broadcasts the status information in Morse code in a periodic manner, the base station is able to resynchronize its clock at every pass of the satellite according to the received time.

3.4 Validation

The most important part for the security solution is the validation of the received messages. The satellite has to be able to identify messages from unauthenticated parties including replayed ones. Therefore, both received HMAC and timestamp have to be validated by the satellite.
Since HMAC validation includes hashing of the received message it is computational more expensive than timestamp validation. Thus, the timestamp should be validated first.
The signal runtime for sending the satellite time to the base station and for sending a message to the satellite is causing an almost static time drift. Thus, the satellite has to distract the expected time drift from a received timestamp taking both runtimes for uplink and beacon downlink into account. Afterwards the satellite has to verify that the received timestamp is inside valid time window. To achieve this, the satellite gets the actual time from the RTC and checks if the timestamp from the received message matches the satellite time. Since it is not possible to predict the uplink and downlink time with perfect accuracy for all cases, there has to be a certain tolerance in which timestamps are treated as valid. In order to make the timestamp validation more clear, Algorithm 1 provides pseudocode for the timestamp validation.

Algorithm 1. Pseudocode for timestamp validation

```
function validate_timestamp(timestamp t_rec){
     t_sat = rtc_get_timestamp();
     t_rec = t_rec - downlink_time - uplink_time;
if((t_rec > t_sat+tolerance ) || (t_rec < t_sat-tolerance))
     return failure;
else
     return success;
}
```

On the other hand, the HMAC validation is rather simple to explain. The satellite has to construct the HMAC of the received message and verify that it is the same as the received one. The construction of an HMAC is standardized in [5]. Given a hash function *H*, a secret key *k*, a message *m* and two padding constants *opad* and *ipad*, a HMAC is constructed as shown in Equation 1.

$$HMAC(k,m) = H(k \text{ xor } opad \,||\, H(k \text{ xor } ipad \,||\, m)) \quad (1)$$

4 SESSION MANAGEMENT

Due to the use of timestamps a logical session management can easily be introduced. One session is hereby one pass of the satellite. Since the base station at the NTNU is the only ground segment which shall have an uplink access to the satellite, it would be logical to disallow the access in the time between passes. Therefore, session management can be used to reduce the overall attack surface and tie the attacker to a geographical location.

We developed an exemplary session management procedure in order to analyse the capabilities and overhead of a solution utilizing session management. The sketched session management procedure requires the satellite to track four additional variables: *session_start*, *session_end*, *t_saved* and *session_active*. When a new message is received it has to be checked if a session is already initiated or not which is saved in the *session_active* variable. In the case of an initiated session it has to be verified that the received timestamp *t_recv* is higher than the last received timestamp *t_saved*. Additionally, the satellite has to verify that the timestamp is inside the boundaries of the actual session which are provided by *session_start* and *session_end*.
In the case that no session is active the satellite has to verify that an initiation of a session is possible. Since the duration between two passes is nearly constant, the session management functionality has to verify that the difference between the fresh received and the last valid timestamp is higher than the passing interval. As soon a session is initiated the values for *session_start* and *session_end* have to be set.
The detachment of an existing session is done separately in the task being responsible to provide the time to the beacon. The reason is that the satellite time is retrieved periodically here and it can easily be checked if this time is outside the specified session boundaries.
The full session management procedure is visualized as flowchart in Figure 3.

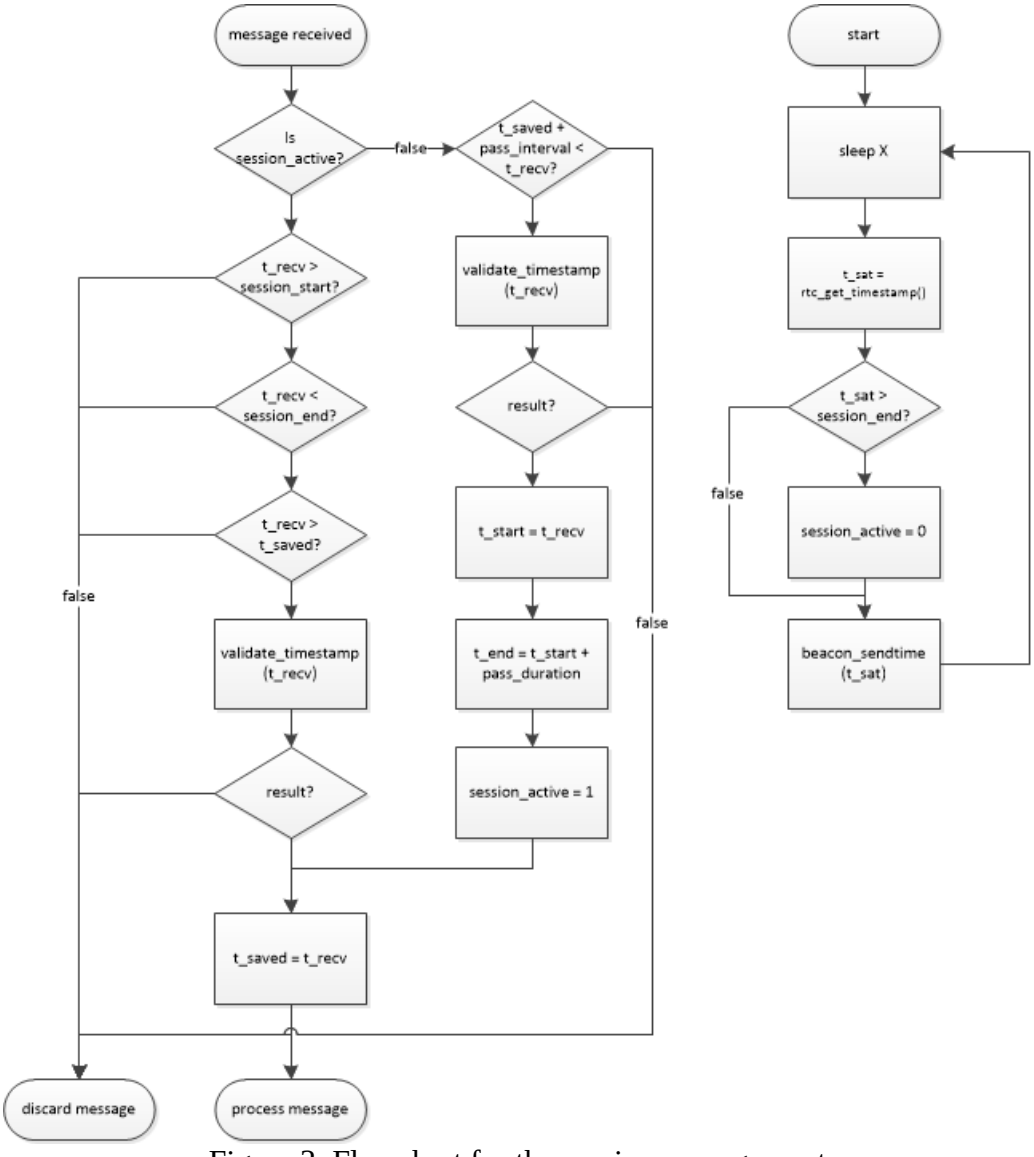

Figure 3. Flowchart for the session management

5 PROOF-OF-CONCEPT IMPLEMENTATION

The proposed design was implemented as proof of concept on an AVR UC3-A3 Xplained evaluation kit which uses the same MCU as used in the radio module. The evaluation kit was connected with the DS3231SN clock over an I²C bus. This clock is also used by the satellite and is suitable for the lower earth orbit and the project with an operating temperature range between -45 °C to +85 °C and an accuracy of ±3.5 ppm [6]. Since hardware for the beacon was not available during the development of the security solution, a software side simulation of the beacon was written for the UC3-A3. The simulation retrieves the time from the RTC and emulates the beacon output via an RS232 connection. Likewise, the uplink was emulated over this connection. Therefore, a realistic representation of the link properties is not included in the proof of concept implementation. However, this is not a problem since it shall only be demonstrated that the proposed design is working under laboratory conditions.
As third-party software FreeRTOS serves as real-time operation system in order to be able to utilize different tasks. Additionally, PolarSSL is used to provide cryptographic hash functions and HMAC functionalities. As specific hash function MD5 was chosen for the proof of concept implementation. This decision is based on better performance results [7], less overhead and the fact that MD5 is still reasonable secure for HMAC usage [8].
To demonstrate an exemplary security layer, a protocol header was developed for the proof of concept implementation. The header contains a magic value to identify the beginning of the security header, a field for the timestamp, a field for the HMAC and a field for the length of the following data as shown in Table 1. The HMAC is constructed over the security header and data in order to authenticate the timestamp as well. Thus, the HMAC field is filled with 16 zero bytes for construction and validation of the HMAC.

Table 1. Security header for the proof of concept implementation

Field	Magic Value	Timestamp	HMAC	Data Length
Size [bytes]	1	4	16	2

The implementation itself is done in C and is thus procedural. Two continuously running tasks for FreeRTOS were developed. One is responsible for simulating the uplink and accepting incoming messages. It calls a function for the security validation as soon a message is received. The other task is responsible for providing the security prerequirements which is retrieving the time from the RTC and sending it to the simulated beacon output. These functionalities are divided in three different C-files and the schematic overview is given in Figure 4.
The base station part of the system is represented in a simple Perl script running on a PC. It takes messages specified by the user as input and waits for the time provided by the simulated beacon signal. Afterwards it encapsulates the message in the security header, constructs the HMAC and sends the packet to the UC3-A3.
This proof of concept was successfully tested. First of all, correct packages were properly identified and validated by the security module. Additionally, we inserted malformed packets on purpose which were rejected in all cases. In order to simulate different attack scenarios, packets with invalid HMACs, data length values and timestamps were constructed. While the first two always led to a discard of the packet, invalid timestamps which were close to the actual time were still accepted. This behaviour results from the necessary timestamp validation tolerance described in Section 3.4. Thus, a theoretical threat for a replay attack does still exist. However, the time window for such an attack is rather small and lies in a range of a few seconds. Furthermore, a real world attacker has to receive a message first before he can replay it which minimizes the attack window even more.

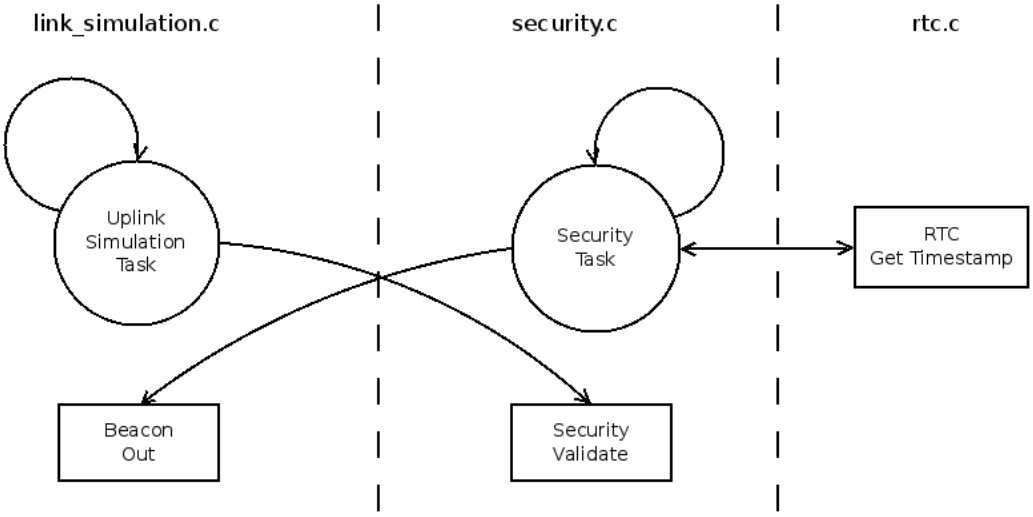

Figure 4. Schematic overview of the proof of concept implementation

6 CONCLUSION

The developed security solution demonstrates mainly two matters of fact usable for the NUTS project: the usage of a modular security solution not utilizing other protocols and the usage of timestamps for replay protection is realizable. The project would benefit from a modular solution due to removed complexity and software overhead in other modules than the radio. Timestamps based on the satellite time are a feasible solution for clock synchronization and are suitable as replay prevention even though a small attack window is still open.
Although the usage of the session management appears plausible, it is discarded for the NUTS project. The proposed session management adds unnecessary complexity to the security module and allows more surface for failures.
The design itself was successfully implemented and tested in a simplified environment outside the satellite. The NUTS project aims to implement the presented solution on the final satellite.
Thus, future work includes a redeployment of the system in a more realistic environment in order to eliminate unforeseen failures. In addition to that, the solution should be formally verified to ensure its robustness. If the design successfully withstands this additional analysis it will be integrated to the satellite. In this case different tests can be done once the satellite is in operation in order to make real-world statements about the capabilities of the presented solution.

7 REFERENCES

[1] ITU, *Radio regulations articles*, Article 25.2, Geneva, Switzerland, 2012.

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

[3] GomSpace, *CubeSat Space Protocol - A small network-layer delivery protocol designed for Cubesats*, Aalborg, Denmark, http://www.libcsp.org/ [Online – accessed 10-April-2014].

[4] Bezem B. and Fjellby P.K.J., *Authenticated uplink for the small, low orbit student satellite NUTS*, Project Work, Norwegian University of Science and Technology, Trondheim, Norway, 2012.

[5] Bellare M., Krawczyk H. and Canetti R., *HMAC: Keyed-Hashing for Message Authentication*, RFC 2104, Internet Engineering Task Force, 1997.

[6] Maxim Integrated, *DS3231 - Extremely Accurate I2C-integrated RTC/TCXO/Crystal*, San Jose, California, 2013.

[7] Münch M., *Development of a Security Module for the Uplink of the NUTS Student Satellite* Project Work, Norwegian University of Science and Technology, Trondheim, Norway, 2014.

[8] Stallings, *W. Cryptography and Network Security: Principles and Practice*, Prentice Hall, 2011