An Approach for Recovering Satellites and their Cryptographic Capabilities in the Presence of SEUs and Attacks

Marcio Juliato and Catherine Gebotys
Dept. of Electrical and Computer Engineering, University of Waterloo
200 University Avenue West, Waterloo, ON, Canada
{mrjuliat,cgebotys}@uwaterloo.ca

Abstract

The problem of recovering satellites from failures or attacks, and bringing them back to an operational and safe state is crucial for satellite reliability. However, there is limited research in this area, despite the recent interest in incorporating security in satellites. This research proposes a trusted hardware module approach for recovering the satellite's cryptographic capabilities after an SEU has corrupted a cryptographic key or an attack has compromised the satellite. The proposed trusted modules are estimated to consume no more than 266mW of total power using an Altera Cyclone II FPGA. Different security levels are supported with the trusted module approach in order to meet the requirements of particular designs. Security analysis in terms of brute force attack for different types of satellite orbits is also presented. We show that the time spent in a brute force attack against the proposed system is completely independent of the computing power of an attacker, resulting in a very secure system. This research is important for addressing key recovery which is crucial for present and future satellites.

1. Introduction

The growing availability of technology around the world has posed new threats in several areas. Since countries heavily rely on commercial satellites for communication, any threat to a satellite could have the potential for putting a nation's critical infrastructures at risk [19, 21, 2]. Satellites being compromised have also been reported [10, 17, 7], clearly demanding incorporation of security into satellites. Even with the current importance of these spacecrafts, their security has relied on obscurity and uniqueness of the systems [5]. Unfortunately making satellites secure is a complex problem given their stringent constraints including low energy, low power, high reliability, and low cost.

Satellites traditionally communicate with two types of ground stations: communications stations and control stations. The latter will be referred to as "ground stations" for short. Satellite links can be divided into three groups: 1) Telemetry, tracking and control links (TT&C); 2) Data links; and 3) Cross-links. The TT&C link is important since it provides mission secrecy and reliable control of the satellite. Hence it requires both confidentiality and authentication. Thus a symmetric and asymmetric cryptosystem with cryptographic keys onboard the satellite is a necessity.

The cryptographic modules must satisfy the stringent restrictions on the satellites hardware size and weight as well as the unit's power consumption and dissipation since the satellite has a limited power supply. Furthermore, it is important to design fault tolerant hardware so that the effects of radiation (solar, cosmic) on circuits do not cause critical failures which could lead to the loss of the satellite. The fact that the system will be in orbit means that it will not be easily accessible, and as such, would have to perform for long periods of time without physical maintenance.

One of the major factors that complicate the implementation of security in the TT&C link is the presence of fault inducing radiation in space. Particles originating from space are the best documented causes of a class of errors known as Single Event Upsets (SEUs) [14] making such errors of great concern in aerospace applications. SEUs are generally a form of soft error, i.e., they are errors that occur in the circuitry of a system that can cause a bit to be flipped into an erroneous state, but are not damaging to the hardware. Typically satellites employ FPGAs due to their low volumes. In this technology a cryptographic key can get damaged by both the corruption of the FPGA configuration data [9] and directly in a storage element designed for key storage in the FPGA. Several Altera FPGAs [6] have built-in cyclic redundancy check (CRC) circuitry, which checks the internal FPGA's configuration data, and request the reconfiguration of the device if it detects any corrupted bit. Though provid-

Date: August 18, 2008.

ing a certain level of SEU resistance, this approach does not protect the user data stored or being processed within the device. As a consequence, user data must be protected by other means as Triple Modular Redundancy (TMR) [22, 3] and Hamming codes [11]. Hamming codes are especially interesting in terms of storage area when large registers are used, which is exactly the case of the cryptographic keys' registers addressed in this paper.

Although incorporation of security into satellites is complex due to the stringent constraints on power, energy, cost and reliability, an important emerging new problem to be addressed is how to recover from compromised keys, SEU damaged keys, as well as supporting new keys for higher security levels. Section 2 presents some related work, while Section 3 focuses on our approach to recover satellites from faults and attacks. In Section 4 the implementation of the trusted modules is described, and experimental results are given in Section 5.

2. Related Work

Traditionally, the proposed methodologies for the mitigation of SEUs in aerospace applications have focused on the protection of entire systems or tasks, thus consuming a large amount of resources. In [3] it is suggested the use of full Triple Modular Redundancy (TMR) and read back and reconfigure to fully protect systems from SEUs. Researchers [24] attempt to reduce the costs associated with fault mitigation techniques by only applying them to the most critical components of a design.

Some research [12, 23, 1] proposes the use of satellites for key distribution and authentication in communication systems. However, the satellites' security is not addressed. There is limited research focusing on incorporating security in commercial satellites despite the constant growth in applications using them. This is probably due to the false assumption that satellites are out of reach of attackers. This context has led the United States General Accounting Office to publish a report [19] calling for a higher level of protection for the country's critical infrastructure. This report presents various threats, concluding that the security of commercial satellites should be more fully addressed.

In [26] an on-board security architecture for earth observation satellites is proposed. It also presents a fault-tolerant AES based on parity-based fault detection scheme and Hamming codes to mitigate SEUs or single bit faults in AES on-board implementations. In [20] a proposal is presented to generate cryptographic keys from features associated directly with the actual satellite. However, such a technique is only valid if we assume that an attacker never gains control over the satellite. Otherwise, an intruder could learn the satellite characteristics and then have enough information to easily derive future cryptographic keys.

Apart from the research outlined above, there is limited research addressing the recovery of cryptographic capabilities of satellites that have suffered faults due to SEUs or from attacks leading to compromised keys. This paper proposes an approach based on trusted modules, which can recover a satellite damaged by SEUs or even when an attacker has broken into it.

3. Satellite Recovery from Failures and Attacks

Considering the important role of commercial satellites in modern communication systems, it is crucial to maintain them functioning all the time. Attackers on the ground cannot be disregarded, given that they may try to compromise satellites' operations or even break into them. An attacker could for example, send commands to a satellite compromising its correct orbit. In this case, emergency commanding should be performed with some sort of authentication [4]. Furthermore, such an attacker can break into the satellite and modify, for example, its FPGA configuration or program memory. Therefore, satellites must quickly recover from these attacks and return to a reliable state. After that, it must safely renegotiate new cryptographic keys and re-establish a secured channel with the ground station.

SEUs can affect any key at any moment, therefore it is possible to get one or more keys corrupted simultaneously. In addition, hardware failures such as power outage and chip failure can happen due to a number of reasons (temperature, satellite position, etc). Due to these problems, it may be necessary to issue a general reset where the FPGA is reconfigured and the program memory re-initialized. Again in this scenario, a complete set of cryptographic keys can be lost. Depending on the type of the key that is lost, the satellite can adopt a specific recovery technique.

3.1. Simple Methods for Key Recovery

Depending on the type of key that is lost, the satellite could adopt a specific recovery technique. For example, in the case of losing only the public or the private keys, a point-to-point key transport technique based on symmetric encryption, as presented in [25, 13], could be used to re-establish the satellite's public and private keys. However in order to use this approach, it is assumed that the satellite always has a secured channel with the ground station. The ground station creates new private and public keys, which are sent to the satellite through the secured channel. This is worth doing since asymmetric key generation is too expensive in terms of power consumption. Likewise, the Needham-Schroeder public key protocol [16, 25, 13] could be used to address the problem of permanent damage of symmetric keys. When the ground station and the satellite possess each other's valid and authentic public keys, they

would run the Needham-Schroeder protocol to negotiate a new symmetric key. However again, this method involves public key cryptographic operations which is not suitable for the low energy/power requirements of the satellite.

More robust approaches need to be developed for satellite cryptography. For example, assumptions of a secured channel with the ground station may not hold due to a SEU-corrupted (or attacker compromised) encryption key. Furthermore low energy techniques are necessary. This paper proposes specific trusted modules which are suitable for supporting satellite recovery from corrupt keys. The architecture of the trusted modules as well as their associated threat models are outlined next.

3.2. Hash Based Approach

The first threat model to consider assumes that an attacker never gains control over the satellite, but listens to all communication (ground station–satellite, satellite–satellite) and also sends control signals to the satellite. We assume that a satellite stores hardwired information k', which is built-in during its construction and unknown to any attacker. Hence, it is assumed that the agency which built the satellite will disclose this information only to the agency that will control it. In other words, besides the manufacturer, only the ground station and the satellite know k'. Notice that k' must not be stored in the FPGA since it could get corrupted by SEUs or lost in reconfiguration.

Further, k' must never be used as a key. Once this value is discovered or leaked by any means, the satellite will never be able to re-establish a secured channel with the ground station. In fact, that is why this threat model assumes that an attacker never breaks into the satellite. Once k' is read by an attacker, he or she has learned the most important piece of information that is used to create all the future cryptographic keys.

Additionally, the FPGA may be reconfigured remotely and the FPGA's configuration file could also be corrupted by SEUs. Hence, it is necessary to store a minimum configuration file in a programmable read-only memory (PROM). This minimum, permanently stored capabilities include some mandatory functions for the recovery process, such as basic arithmetic operations and a hash function. The PROM should also store a minimum program to be loaded into the program memory. Even though this threat model assumes that FPGA reconfiguration may occur, addressing the reconfiguration itself is out of scope of this work.

Considering the implementation of the proposed technique in current satellites, k' could be a serial number, an ID number, or any other data that only the satellite and the ground station know, and which is readable by the FPGA. It would be better still if k' was randomly created during the satellite construction. Moreover, a bigger k' could be formed by concatenating more than one source of information. In new satellites, k' should be randomly created, be hardwired in the satellite circuitry, and be readable by the FPGA. This permanent storage of information would then be immune to SEUs for the whole lifetime of the satellite.

The recovery can be performed as follows: The ground station sends a nonce n to the satellite, which is defined as a random number used no more than once. To construct a new symmetric key k, the satellite concatenates k' with n, and computes its hash. More precisely, $k := H(k'||n)$. In addition, the ground station performs the same computation. Observe that the nonce brings freshness to the key generation, ensuring that every time n changes, so does the key k. Thus, it is possible for the satellite to frequently change the cryptographic keys. The generated key k should be long enough to match the required security level and a hash function should be chosen accordingly.

At this point, the satellite and the ground station share the same session key k, and can re-establish a secured channel. This approach is relatively simple, only requiring the computation of one hash. It is secure, given that it is infeasible for an attacker to compute k from n and assuming that k' remains unknown to the attacker. As a consequence, only the ground station and the satellite can generate the same key k since they are the only two entities knowing k'. Due to its simplicity, it can be used either in existing or new satellites, and also reused as many times as it becomes necessary by simply changing the nonce n.

3.3. Trusted Modules Approach

The next threat model makes the same assumptions of the first threat model, but with two additional assumptions: i) an attacker can break into the satellite, can modify the FPGA configuration and the program memory, and thus control the entire satellite; ii) Due to the presence of an attacker, the satellite may not always have a hash function implemented in the FPGA logic to create a new key. When more harsh and realistic scenarios are taken into account, this threat model becomes mandatory. The trusted module approach is the main strategy to protect modern satellites from failures and attacks, and as such, it is on the focus of the remaining sections of this paper.

Since an attacker can modify the satellite's untrusted computing system, it becomes impossible to recover the control over the satellite without the help of trusted modules. Thus, it is mandatory to have the trusted modules implemented with fault-tolerance and out of reach of attackers, i.e. apart from the general computing platform. In order to avoid a man-in-the-middle attack in the satellite's circuitry, it is mandatory to have a secure information path between the radio-frequency circuitry and the trusted modules. The proposed approach uses three trusted modules,

including a trusted random number generator available only to the trusted modules.

The *trusted hash and configuration module*, shown in Figure 1, is responsible for checking the integrity of some system components, i.e., the hashes of the program memory, the current FPGA configuration, and the FPGA configuration file. For example, by using the scheme presented in [8], the integrity of the untrusted storage elements can be verified. The trusted module also contains some elementary circuitry to re-initialize the program memory and to configure the FPGA from the minimal configuration file. Given that the efficient hashing technique is well studied in [8], the remainder of this will focus on the details of the trusted reset and key recovery modules.

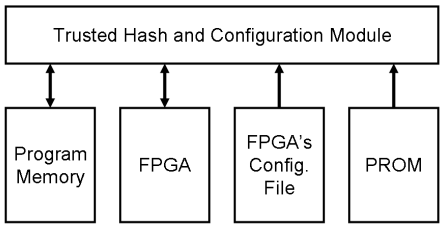

Figure 1. Trusted hash and configuration module, used to check the integrity of the satellite's components.

The *trusted key recovery module*, is responsible for recovering cryptographic keys. This module stores l keys $(k_1, k_2, ..., k_l)$ in a table, to be referred to as secrets table. The index of this table is a number n, which is b-bits long, thus leading to an address space of size 2^b. Actually, n is denoted as an one-time key recovery secret, which is sent to the satellite in the clear (i.e. no encryption is needed) by the ground station following the recovery protocol to be described in Section 3.4. If n points to a position holding a key, such a key is output. Observe that the one-time key recovery secret is never used as a key itself. Otherwise, any entity listening to the communication could use n to decrypt future encrypted messages.

The strength of this module relies on two fundamental features: i) the l keys are randomly distributed throughout the 2^b positions of the table, and ii) b is made big enough to become infeasible for an attacker using brute force to find a position containing a key. It is feasible to implement the proposed scheme by using a selection circuitry, as shown in Figure 2. With that, only l storage elements are required for the keys, even maintaining a huge address space. This way, up to l key recoveries can be performed by the ground station. Therefore, the greater l, the longer the satellite's protection against key losses. However, the smaller l, the less storage is needed in the satellite. For example, one could use 128-bit keys, $l = 64, 128$, or 256, and $b = 64, 128$, or 256 bits.

Another important feature of the trusted key recovery module is that, once a given key is read, it is removed from the secrets table and never used again. This also means that a given one-time key recovery secret is used only once. If reused, attackers could perform replay attacks and easily break the proposed scheme.

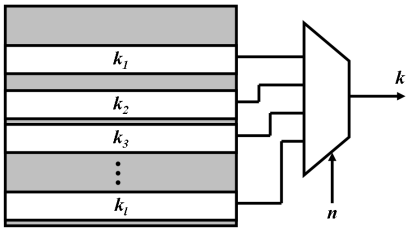

Figure 2. Table and selection circuitry used for recovering k from one-time secret n.

The *trusted reset module*, is related to the system restart. Its function is the issue of a general reset signal to bring the satellite to a reliable state from which the rest of the system can be recovered. This module can also be thought as a table, but instead of outputting a key when it receives a valid one-time reset secret s, it issues a general reset. The reset signal causes the FPGA to be reconfigured (with minimum capabilities) and the program memory to be restored. Furthermore, once a reset position is used, it is destroyed and can never be accessed again. This prevents an attacker from resetting the satellite through a replay attack using an old reset secret s.

When these three trusted modules are implemented in a satellite, it becomes possible to detect when one of its crucial components gets corrupted, bring the satellite to a reliable state, and restore its cryptographic capabilities. For that, a recovery protocol is used, whose details are shown in the next section.

3.4. Recovery Protocol

From time to time, during the normal operation of the satellite, the trusted hash module checks the integrity of the program memory, the FPGA's current configuration, and the FPGA's configuration file. The computed hashes are then sent to the ground station. Since the trusted hash module is out of reach of attackers, it can detect any data corruption on those components due to intentional and unintentional causes, e.g. SEUs and attacks. Given that the ground station knows the hashes of the FPGA configuration and program memory, a simple comparison determines whether the satellite has some crucial component corrupted. If that is the case, the ground station broadcasts a message to the communicating parties stating that the faulty satellite is not reliable, and imposes that all parties stop communicating with

it. It is up to the ground station to decide what to do next with the satellite. In some cases only a cryptographic key recovery may be necessary, whereas in more severe cases, a reset followed by a key recovery may be performed. The latter case is more complex and is exemplified below.

In order to proceed with the recovery as the entity controlling the satellite, the ground station uses the one-time secrets n for key recovery and s for the reset. As shown in Figure 3, a challenge-response protocol is used. The ground station first requests the initiation of the trusted reset protocol. Then the satellite generates a random number r and sends it to the ground station. After that, the ground station performs an exclusive-or ($\oplus$) of s and r, and responds $(s \oplus r)$ to the satellite. Finally, the satellite knows r and $(s \oplus r)$, performs $r \oplus (s \oplus r)$, and thus recovers s. For the key recovery procedure, the protocol is exactly the same, but n is used instead of s.

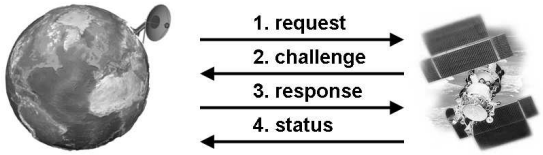

Figure 3. Recovery protocol based on challenge-response.

When the trusted reset module receives s, it indexes its secrets table. If s corresponds to a valid address, a general reset signal is issued. Next, the accessed address in the secrets table is destroyed. This means that an attacker could listen to s but could never reuse it in a replay attack to issue a new system reset. Therefore, this action is performed only once and uniquely by the holder of the one-time reset secret s. As a result of the reset signal, the FPGA and the program memory are reconfigured. After that, the satellite will send a status message to the ground station, which contains information on the success of the reset operation and also the new hashes of the FPGA configuration file and program memory.

After these steps, the satellite is in a safe state, but still does not have a secured channel with the ground station. At this point the key recovery protocol takes place. Again, the ground station requests the initiation of the protocol. The satellite generates a new random number r', and sends it to the ground station. The ground station computes $(n \oplus r')$ and sends it back to the satellite. Next, by knowing r' and $(n \oplus r')$, the satellite recovers n. Finally, the trusted key recovery module uses n to index its secrets table and outputs a cryptographic key, which is then used to establish a secured and authenticated channel with the ground station.

Additionally, the ground station can use the new secured channel to send new program and FPGA configuration files to the satellite. This may be an important step, since attacks or failures may have occurred due to bugs or security holes in the previous system's embedded hardware or software. Later on, the ground station broadcasts a message allowing other parties to resume their communications with the satellite. Although a worst case scenario has been described, the recovery can be simpler depending on the type of the failure. For example, it may be necessary only to recover a cryptographic key, and the satellite reset may not be needed.

One may argue that it would be much faster and cheaper not to use the challenge-response protocol, and instead of that, send s and n directly to the satellite. We consider that a valid argument, however the challenge-response protocol is used to increase the security of the system by taking into account the time t spent in the communications between the ground station and the satellite. More precisely, the communication time t (in seconds) is determined by $t = d/c$, where d is the distance (in Km) between Earth and the satellite, and c is the speed of light in vacuum (299,792.458 Km/s).

This point becomes much clearer when observed from the perspective of an attacker. Without the challenge-response protocol, an attacker could perform a concurrent attack by sending an (invalid) one-time secret after the other. In fact, the delay between two trials would be the time spent by the satellite to process the one-time secret, which is performed quite quickly. Now, considering the challenge-response protocol, an attacker cannot perform a concurrent attack and is imposed to exchange 3 messages with the satellite, thus spending $3t$ seconds in communication time between two trials. The status message can be temporarily discarded by the attacker.

Table 1. Protocol delay

Satellite Type	Distance d from Earth (Km)	Protocol Delay (3 messages) (s)
LEO	80 (minimum)	$2.40\text{x}10^{-3}$
MEO	1,700 (minimum)	$5.10\text{x}10^{-2}$
GEO	35,700 (minimum)	1.07
Mars	78,338,750 (average)	$2.35\text{x}10^{3}$

Table 1 shows the protocol delay, considering the exchange of 3 messages, for different kinds of satellites [18], e.g. Low Earth Orbit (LEO), Medium Earth Orbit (MEO), and Geosynchronous Orbit (GEO) satellites. An estimate of the average communication time with a satellite orbiting Mars [15] is also included given the interests of many space agencies on that planet. A ground station that knows the one-time secrets would spend $2.4ms$ on the challenge-response protocol while recovering a LEO satellite. If the satellite was orbiting Mars, it would spend 2350s (about 39 minutes). We are assuming that these communication delays are acceptable for ground stations in most cases, and is worth doing because of the extra security that it brings to the system. Consequently, one-time secrets can use fewer bits, thus saving valuable implementation area.

4. Trusted Modules Implementation

The implementation of the trusted modules consists of four components: control unit, address counter, secrets and keys table, and compare secret unit. The internal architecture of the trusted modules is shown in Figure 4. The control unit is responsible for receiving the one-time secret and for managing its evaluation. The whole process takes four clock cycles to: access the keys/secrets table, compare the stored secret with the received one, destroy the used secret, and increment the address counter. Even though the trusted reset and key recovery modules have different functions, their internal architectures are very similar. The only difference, represented by the dashed lines in Figure 4, is the additional circuitry for managing the cryptographic keys within the trusted key recovery module.

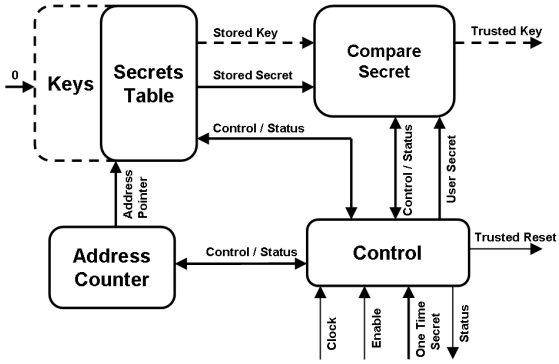

Figure 4. Trusted modules architecture.

The secrets table stores l secrets, each of them b bits wide, which are protected against errors by the use of hamming codes. Actually, all secrets are stored in an encoded form, by adding p parity bits to each storage element. When the memory is read, an internal Hamming decoder corrects a potential bit flip caused by SEUs, and sends the secret to the secret compare unit. The implementation of the trusted reset module considers secrets of 64, 128 and 256 bits, which require the addition of 7, 8, and 9 parity bits, respectively. In the case of the trusted key recovery module, the secrets and keys are stored together in the same storage element. This saves some parity bits in its Hamming encoded form, thus requiring only 8, 8, and 9 parity bits, respectively to encode the 64, 128 and 256-bit secrets along with the 128-bit keys. Hence, the total memory bits required to store the secrets is given by $l * (b + p)$, and $l * (128 + b + p)$ to store the secrets along with the keys.

Further, if an unrecoverable error is found in the stored secret or key, e.g. two bit flips, the control unit is informed. Consequently, the address counter is incremented and the ground station instructed to use the next one-time secret. One-time secrets are indexed sequentially through a pointer generated by the address counter unit. This unit consists of a fault-tolerant ($log_2 l$)-bit counter, allowing for l resets/key recoveries. It accounts with a Hamming encoder/decoder pair to keep the counter in an encoded form, so that it can correct one bit flip and detect two bit flips.

Once the stored secret is read from the secrets table, it is sent to the compare secret unit. Then, the compare secret unit performs a bit-wise comparison between the stored secret and the one-time secret under test. From the comparison results, the control unit determines whether or not to issue a reset signal (a key, in the case of the key recovery module). If the one-time secret was "guessed" correctly, the stored secret is zeroed in the secrets table (along with its corresponding key, in the case of the key recovery module). Finally, the control unit increments the address counter, thus preparing the trusted module for the next recovery process.

In case of an error in the address counter, its value can be reset by the control unit. In the sequence, the control unit executes a series of counter increments, until it detects that the secret coming from the secrets table is different of zero. At this point, it knows that the address pointer has been recovered to its correct position. Likewise, the control unit follows this procedure to recover the address counter after FPGA reconfigurations. Despite this work shows the implementation of the secrets table using the FPGA's RAM blocks, an actual implementation in a satellite should use flash memory, so that its contents are not lost in an eventual FPGA reconfiguration.

5. Experimental Results and Discussion

Our experimental results are based on the several implementations using an Altera CycloneII EP2C35F672C6 FPGA. The synthesis of the trusted modules are presented along with an analysis of brute force attacks. All the synthesis were performed using QuartusII version 7.2.

Table 2. Trusted Reset Module: Implementation Results

Secret (bits)	# of Resets	Area (LEs)	Mem. (bits)	Freq. (MHz)	Dynamic/Total Power (mW)
64	64	502	4608	62.24	10.78 / 137.95
	128	507	9216	63.50	11.03 / 137.83
	256	512	18432	63.03	12.25 / 139.36
128	64	887	8768	59.01	14.83 / 151.01
	128	892	17536	58.97	16.17 / 153.44
	256	901	35072	58.83	16.80 / 153.48
256	64	1642	17024	57.14	17.37 / 173.95
	128	1647	34048	56.06	19.28 / 177.18
	256	1654	68096	54.18	20.56 / 177.19

As shown in Table 2, the simplest implementation of the trusted reset module uses 64-bit secrets and can issue up to 64 reset signals. In this configuration, the module occupies 502 logic elements (LEs), operates at 62.24MHz, and uses 4608 memory bits for the secrets table. Besides, it consumes 10.78mW of dynamic power and 137.95mW of total power. The total power consumption includes I/O

power dissipation and the static power dissipation (about 81mW for the trusted reset modules). In its most secure version, using 256-bit secrets and being able to issue up to 256 reset signals, the module occupies 1654 LEs, runs at 54.18MHz, and uses 68096 memory bits. Further, it consumes 20.56mW of dynamic power and 177.19mW of total power. Furthermore, compared to the simplest version, the more secure version consumes 1.91 times more dynamic power and occupies 3.3 times more area.

Table 3. Trusted Key Recovery Module: Implementation Results (128-bit keys)

Secret (bits)	# of Keys	Area (LEs)	Mem. (bits)	Freq. (MHz)	Dynamic/Total Power (mW)
64	64	1213	12864	56.73	21.34 / 209.36
	128	1219	25728	55.34	20.92 / 208.35
	256	1224	51456	58.33	23.33 / 209.81
128	64	1674	17024	55.78	29.81 / 231.25
	128	1686	34048	55.19	29.97 / 230.14
	256	1691	68096	57.57	31.96 / 231.49
256	64	2356	25216	52.97	38.05 / 264.60
	128	2356	50432	51.75	39.23 / 265.30
	256	2372	100864	50.73	40.65 / 266.02

The results shown in Table 3, refer to the trusted key recovery modules working with 128-bit keys. In its simplest version, the module uses 64-bit secrets and can issue up to 64 keys. For that, it occupies 1213 LEs, 12864 memory bits, and operates at 56.73MHz. It has dynamic power consumption of 21.34mW, whereas its total power consumption is 209.36mW. Again, the static power dissipation of all trusted key recovery modules is about 81mW.

On the other hand, its most secure version uses 256-bit secrets and allows for 256 key recoveries. This module utilizes 2372 LEs, 100864 memory bits, and runs at 50.73MHz. Further, it has dynamic and total power consumption of 40.65mW and 266.02mW, respectively. Moreover, this module occupies 1.95 times more area, and consumes 1.9 more dynamic power than its simplest version.

Table 4. Trusted Reset and Key Recovery: Processing Times

Secret (bits)	# of Keys/Resets	Reset Proc. Time (ns)	Key Recovery Proc. Time (ns)
64	64	64.27	70.51
	128	62.99	72.28
	256	63.46	68.58
128	64	67.79	71.71
	128	67.83	72.48
	256	67.99	69.48
256	64	70.00	75.51
	128	71.35	77.29
	256	73.83	78.85

Moving to the processing time analysis, one may observe through Table 4 that all modules are quite efficient. The slowest and the fastest trusted reset module have processing times of, respectively, 73.83ns and 64.27ns. Similarly, the slowest trusted key recovery module executes a key recovery in 78.85ns, while the fastest one performs the same operation in 70.51ns.

In order to determine the strength of the modules against brute force attacks, we must also to take into account the communication protocol delay. Because of the trusted modules' processing times are quite small in face of the protocol delay, the former can be disregarded in the determination of the total time spent in a brute force attack. Additionally, the total number of trials that an attacker must perform in a brute force attack is 2^b, where b is the number of bits of the one-time secret. Moreover, an attacker can perform no better than wait for the end of each protocol run to launch another trial. Thus, the total time (in seconds) spent in a brute force attack is $2^b * 3t$, where $3t$ is the time spent exchanging 3 protocol messages as given by Table 1. The total time (in years) spent in a brute force attack against the trusted modules are presented in Table 5. Since these times are completely independent of the computational power of the attacker, it can be concluded that the proposed schemes are very secure.

Table 5. Brute Force Attack

Secret (bits)	Brute Force Attack (years)			
	LEO	MEO	GEO	Mars
64	$1.40\text{x}10^{9}$	$2.99\text{x}10^{10}$	$6.27\text{x}10^{11}$	$1.38\text{x}10^{15}$
128	$2.59\text{x}10^{28}$	$5.51\text{x}10^{29}$	$1.16\text{x}10^{31}$	$2.54\text{x}10^{34}$
256	$8.82\text{x}10^{66}$	$1.87\text{x}10^{68}$	$3.94\text{x}10^{69}$	$8.64\text{x}10^{72}$

The less secure system is obtained when 64-bit secrets are used in conjunction with LEO satellites. In this case, a brute force attack would take $1.40\text{x}10^{9}$ years. However, the farther the satellite is, the more secure the scheme becomes for the same secret size. For instance, a satellite implementing the same modules, but orbiting Mars, would raise the total time of a brute force attack to $1.38\text{x}10^{15}$ years. On the other hand, if the construction constraints of a satellite allows for the use of more hardware area, 256-bit secrets could be used. As a result, a brute force attack would become in the order of 10^{57} harder. For example, it would take $8.82\text{x}10^{66}$ for an attacker to break such a system implemented in LEO satellites, where as this number would increase to $8.64\text{x}10^{72}$ years for satellites orbiting Mars.

As can be computed from Table 2, this increased security level of using 256-bit secrets does not lead to a big increase in the implementation area, memory bits, and dynamic power consumption, when compared to 64-bit secrets. Actually, the trusted reset modules would occupy 3.23 times more LEs, 3.69 times more memory bits, and consume 1.67 times more dynamic power. Similarly, from Table 3, the trusted key recovery modules would occupy 1.94 times more LEs, 1.96 times more memory bits, and consume 1.74 times more dynamic power. Although larger secret sizes leads to higher levels of security without increasing too much area and power consumption, the security level achieved with 64-bit secrets may be enough for most applications.

6. Conclusions

This paper provides a set of techniques to recover satellites that suffer attacks or severe failures caused by SEUs. Several implementations of the trusted modules using various levels of security and number of recoveries are presented using an Altera Cyclone II FPGA. For instance, a trusted reset module working with 256-bit one-time secrets and allowing for 256 satellite resets, utilizes 1654 LEs and 68096 memory bits. It issues a reset signal in only $73.83ns$ when operating at 54.18MHz, while its dynamic and total power consumption is, respectively, 20.56mW and 177.19mW.

In contrast, a trusted key recovery module allowing for 256 key recoveries and using 256-bit one-time secrets, occupies 2372 LEs and 100864 memory bits. This module can recover a key in $78.85ns$ when operating at 50.73MHz, with dynamic and total power consumption of 40.65mW and 266.02mW, respectively.

A brute force attack against these modules would take $8.82\text{x}10^{66}$ years, when they are applied to LEO satellites. Since the time spent in a brute force attack is completely independent of the computing power of the attackers, the proposed scheme can be considered very secure. In summary, this research addresses the problem of bringing satellites to a safe state after key losses, major failures or attacks, and also securely restoring their cryptographic capabilities. This research is important for support the incorporation of security into satellites.

References

[1] M. Arslan and F. Alagoz. Security issues and performance study of key management techniques over satellite links. In *11th Intenational Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, pages 122–128, 2006.

[2] S. Bain. The increasing threat to satellite communications. *Online Journal of Space Communication*, 6, November 2003.

[3] P. Blain, C. Carmichael, E. Fuller, and M. Caffrey. Seu mitigation techniques for Virtex FPGAs in space applications. In *MAPLD Proceedings*, 1999.

[4] CCSDS. The application of CCSDS protocols to secure systems. Informational Report 350.0-G-2, CCSDS, January 2006.

[5] CCSDS. Security threats against space missions. Informational Report 350.1-G-1, CCSDS, October 2006.

[6] A. Corporation. *Error Detection and Recovery Using CRC in Altera FPGA Devices*. Altera Corporation, January 2007. AN 357.

[7] S. Fleming. Hacker infiltrates military satellite. Online: The Register, March 1999. http://www.theregister.com/1999/03/01/hacker_infiltrates_military_satellite/.

[8] B. Gassend, G. Suh, D. Clarke, M. van Dijk, and S. Devadas. Caches and hash trees for efficient memory integrity verification. In *Ninth International Symposium on High-Performance Computer Architecture (HPCA 2003)*, pages 295–306, 2003.

[9] P. Graham, M. Caffrey, J. Zimmerman, P. Sundararajan, E. Johnson, and C. Patterson. Consequences and categories of SRAM FPGA configuration SEUs. In *MAPLD Proceedings*, 2003.

[10] T. L. Group. Hackers seize UK military satellite. Online, March 1999. http://www.landfield.com/isn/mail-archive/1999/Mar/0001.html.

[11] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2):147–160, April 1950.

[12] I. Ingemarsson and C. Wong. Encryption and authentication in on-board processing satellite communication systems. *IEEE Transactions on Communications*, 29(11):1684–1687, November 1981.

[13] A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[14] G. Messenger and M. Ash. *Single Event Phenomena*. Springer, 2006.

[15] NASA. Solar system exploration webpage. Online, February 2008. http://solarsystem.nasa.gov/planets.

[16] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[17] T. Newspaper. British hackers attack MoD satellite, March 1999. http://www.telegraph.co.uk/connected/main.jhtml?xml=/connected/1999/03/04/ecnhack04.xml.

[18] U. of Concerned Scientists (UCS). UCS satellite database. Online, February 2008. http://www.ucsusa.org/global_security/space_weapons.

[19] U. S. G. A. Office. Critical infrastructure protection: Commercial satellite security should be more fully addressed. Technical Report GAO-02-781, United States General Accounting Office, 2002.

[20] E. Papoutsis, G. Howells, A. Hopkins, and K. McDonald-Maier. Key generation for secure inter-satellite communication. In *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*, pages 671–681. IEEE Computer Society, 2007.

[21] K. Poulsen. Satellites at risk of hacks. Online: SecurityFocus, October 2002. http://www.securityfocus.com/news/942.

[22] D. Pradhan. *Fault-Tolerant Computer System Design*. Prentice-Hall, 1996.

[23] A. Roy-Chowdhury, J. Baras, M. Hadjitheodosiou, and S. Papademetriou. Security issues in hybrid networks with a satellite component. *IEEE Wireless Communications*, 12(6):50–61, 2005.

[24] P. Samudrala, J. Ramos, and S. Katkoori. Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs. *IEEE Transactions on Nuclear Science*, 51:2957–2969, 2004.

[25] D. Stinson. *Cryptography Theory and Practice*. CRC Press, 3rd edition, 2005.

[26] T. Vladimirova, R. Banu, and M. Sweeting. On-board security services in small satellites. In *MAPLD Proceedings*, 2005.