DSA with SHA-1 for Space Telecommand Authentication: Analysis and Results

Susanna Spinsante, *Member, IEEE,* Ennio Gambi, *Member, IEEE*

***Abstract*—The issue of securing Telecommand data communications in civil and commercial space missions, by means of properly located security services and primitives, has been debated within the Security Working Group of the Consultative Committee for Space Data Systems since several months. In the context of Telecommand transmissions, that can be vital in determining a successful operational behavior of a space system, the interest is mainly focused on authentication, more than encryption. The object of this paper is to investigate, under the perspective of computational overhead, the possible applicability of a standard scheme, Digital Signature Algorithm with SHA-1, to the authentication of Telecommand data structures, and to discuss the pros and cons related to its adoption in such a peculiar context, through numerical simulations and comparison with an alternative solution relying on the widely used MD5 hash algorithm.**

***Index Terms*—Authentication, Telecommand, Space Systems, Digital Signature Algorithm, Computational Overhead.**

I. INTRODUCTION

Civil space systems and applications are becoming more and more complex and valuable, either from an economic perspective, and as strategic assets in enabling technological development of countries and nations. Although the importance of such systems has kept growing, for several years the issue of securing space infrastructures for civil applications has been almost neglected: while military space missions, since their initial development, have implemented total security in accessing the spacecraft control systems and data, during all mission phases and under all possible operational or environmental conditions, to date, missions not falling into this category have basically relied on uniqueness and unavailability of publicly known technical details, to deter unauthorized accesses. This situation, however, is gradually changing and, since several months, the Consultative Committee for Space Data Systems (CCSDS) Security Working Group (WG) is engaged in the development of a unified security framework for space data and systems [1], [2]. Besides their economic impact, non military space systems are often crucial in supporting human activities and ensuring security against natural or man-made disasters (see COSMO-SkyMed or Cospas-Sarsat systems, as an example).

In order to issue suited methodologies for space systems security planning, several risk analyses, jointly performed by space agencies and firms, have provided useful indications about the most significant security threats, and their impact. A threat is defined as any circumstance or event having the potential to cause harm to a system, through destruction, disclosure, and modification of data, and/or through denial of service [3]. Ensuring data confidentiality (i.e. data undisclosure by unauthorized entities) and integrity (i.e. detection of unauthorized data modifications) appears to be a fundamental priority, and the CCSDS Security WG has stimulated its member agencies in determining new security recommendations for space missions. According with the basic European Space Agency (ESA) study [4], at least Telecommand (TC) authentication should be applied for all missions. In case of sensitive scientific data, Telemetry (TM) encryption is also strongly encouraged, and the full security implementation would be required by missions providing vital services, such as communication and navigation.

In the field of space data authentication, that usually deals with TC structures, discussions are ongoing in order to evaluate the applicability of general purpose authentication protocols and algorithms to space contexts. The deployment of *ad hoc* schemes may represent an interesting and challenging alternative, but this approach could contrast with the need of providing a global security concept to ensure compatibility with existing infrastructures and communication protocols, and to support interoperability among the solutions adopted by different space agencies. Moreover, while a higher level of robustness and security can be granted by solutions subjected to public verifications, *ad hoc* schemes could reveal unexpected weaknesses, like those discussed in [5] for the ESA authentication engine, now labeled as *historical*.

A preliminary trade study promoted by the CCSDS Security WG [6], and currently in progress, distinguishes two main classes of authentication/integrity algorithms: Digital Signature based schemes, and Message Authentication Code (MAC) based schemes. The former rely on the use of public key cryptography (public and private keys), the latter use a shared secret key, that can be embedded into the data before creating a check word on them (Hash based MAC, HMAC), or can be used to encrypt the check word created by a hash algorithm applied to the data (encryption based MAC). For each class of authentication schemes, the CCSDS Security WG identified a number of potential algorithms for adoption in space contexts: Digital Signature Algorithm (DSA) [7], Rivest-Shamir-Adleman (RSA), and Elliptic Curve Digital Signature Algorithm (ECDSA), in the case of digital signature based schemes; HMAC-Secure Hash Algorithm-1 (HMAC-SHA-1)

Manuscript received March 14, 2008; revised February 21, 2009.

S. Spinsante and E. Gambi are with the Department of Biomedical Engineering, Electronics and Telecommunications, Polytechnic University of Marche, via Brecce Bianche 12, Ancona, I-60131, Italy (email: s.spinsante, e.gambi@univpm.it)

1845-6421/09/8012

[8] and HMAC-Message Digest 5 (HMAC-MD5) [16], in the case of Hash based MAC; Cipher Block Chaining-MAC (CBC-MAC) [10] and Counter with Cipher Block Chaining-MAC (CCM) [11], in the case of encryption based MACs. During its Winter 2006 meeting, the Security WG confirmed the choice of DSA, and discussed the adoption of SHA-1 in non-digital signature environments. As a matter of fact, SHA-1 has recently been demonstrated to provide less than the theoretically foreseen 80 bits of security for digital signatures, and its security strength against collisions is assessed at 69 bits. Anyway, SHA-1 is currently in use in a great number of systems, for which the reduced security strength may not be of great concern.

In a recent work [12], several evaluation criteria to assess the suitability of encryption and authentication schemes for use in Packet Telemetry and Telecommand protocols are discussed, and suited metrics suggested, based on almost theoretical analyses. The same paper reaffirms applicability of SHA-1 for HMAC, despite recent attacks reported in [13], as security proofs show that usage of SHA-1 is not affected by the attacks reported in [14]; in case digital signatures are being used by a mission, DSA must be supported.

According with such evaluations, and in order to verify through simulations and numerical tests the preliminary conclusions provided in [12], this paper investigates the joint adoption of DSA and SHA-1 in the space context, for the authentication of the TC segments and Transfer Frames transmitted from a ground station to a spacecraft. The aim of this work is to evaluate the computational impact of the authentication process on the resources globally available on board (that are costly and scarce), and its performance in presence of residual errors due to the communication channel, and eventually not (or only partially) compensated by an error correction layer. It is important to point out that hash-based schemes are considered in this paper, instead of provably secure digital signature solutions, because security risks analyses have shown that a number of context-related aspects may compensate for the reduced robustness against known or possible new attacks. Provided that an opponent should face several difficulties in accessing the space assets and the information exchanged, interest is focused on the computational efficiency of the solutions proposed, more than on their possible vulnerabilities.

The paper is organized as follows: Section II provides an overview of DSA and SHA-1 algorithms, Section III discusses the application of the two schemes to TC segments and transfer frames, whereas Section IV presents the simulations performed and the results obtained. Finally, Section V concludes the paper.

II. Digital Signature Algorithm (DSA) and Secure Hash Algorithm (SHA-1)

A digital signature of a message is a number dependent on some secret known only to the signer (i.e. his private key), and, additionally, on the content of the message being signed. It associates a message (in digital form) with some originating entity. A signature generation algorithm is a method for producing a digital signature; a verification algorithm is a method for verifying that a digital signature is authentic (i.e. it was indeed created by the specified entity).

In August 1991, the U.S. National Institute of Standards and Technology (NIST) proposed a Digital Signature Algorithm (DSA). DSA has later become a U.S. Federal Information Processing Standard (FIPS 186), called the Digital Signature Standard (DSS), the first scheme ever recognized by any government. DSA was selected for becoming the DSS, based on a number of important features: the level of security provided, the applicability of patents, the ease of export from the U.S., the impact on national security and law enforcement, and its efficiency in a number of government and commercial applications.

The Secure Hash Algorithm (SHA-1), based on MD4, was proposed by NIST for certain U.S. federal government applications. SHA-1 is a cryptographic hash function: it takes a message as input and produces an output referred to as a hash-value, or simply hash. Hash functions are used for data integrity in conjunction with digital signature schemes, in which, for several reasons, a message is typically hashed first, and then the hash-value, as a representative of the whole amount of data, is signed in place of the original message. An overview of the main features of DSA and SHA-1 is provided in the following; further details can be found in [15].

A. DSA

DSA requires, in general, a hash function $h : \{0,1\}^* \longrightarrow Z_q$, for some integer q, where Z_q is the set of integers modulo q. The DSS explicitly requires use of the Secure Hash Algorithm SHA-1 as the building hash function, that is described in the following. A necessary condition for the security of DSA is that computing logarithms in Z_q^* (the multiplicative group of Z_q, i.e. $\{a \in Z_q | gcd(a,q) = 1\}$) be computationally infeasible. This condition, however, is not sufficient.

DSA comprises several algorithms: a key generation algorithm, a DSA signature generation, and a DSA signature verification algorithm. Provided that a sender A has generated its public key (p, q, α, y), being p and q two prime numbers, α a generator of the unique cyclic group of order q in Z_p^*, $y = \alpha^a$ mod p, and a the sender's private key, the DSA signature generation and verification algorithms are as follows:

1. **DSA signature generation**. Entity A should do the following:

(a) Select a random secret integer k, $0 < k < q$
(b) Compute $r = (\alpha^k \bmod p) \bmod q$
(c) Compute $k^{-1} \bmod q$
(d) Compute $s = k^{-1} \{h(m) + ar\} \bmod q$
(e) A's signature for m is the pair (r, s)

2. **DSA signature verification**. To verify A's signature (r, s) on m, the receiver B should do the following:

(a) Obtain A's authentic public key (p, q, α, y)
(b) Verify that $0 < r < q$ and $0 < s < q$: if not, then reject the signature
(c) Compute $w = s^{-1} \bmod q$, and $h(m)$
(d) Compute $u_1 = w \cdot h(m) \bmod q$, and $u_2 = rw \bmod q$

(e) Compute $v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$
(f) Accept the signature if and only if $v = r$

The security of DSA relies on two distinct but related discrete logarithm problems. One is the logarithm problem in Z_p, where the powerful index-calculus methods applies; the other is the logarithm problem in the cyclic subgroup of order q, where the best current methods run in "square-root" time. The size of q is fixed (as per FIPS 186) at 160 bits, while the size of p can be any multiple of 64, between 512 and 1024 bits inclusive. FIPS 186 does not permit primes p larger than 1024 bits.

B. SHA-1

Though based on MD4, SHA-1 presents a number of differences from it:

- SHA-1 hash-value is 160 bits, and five (vs. four) 32-bit chaining variables are used;
- The compression function has four rounds instead of three, and each round has 20 steps instead of 16;
- Within the compression function, each 16-word message block is expanded to an 80-word block, by a process whereby each of the last 64 of the 80 words is the XOR (eXclusive OR, or sum module 2) of 4 words from earlier positions in the expanded block. These 80 words are then input one-word-per-step to the 80 steps;
- The core MD4 step is modified;
- SHA-1 uses four non-zero additive constants, whereas MD4 uses three constants, only two of which are non-zero. The byte ordering used for converting between streams of bytes and 32-bit words in the official SHA-1 specification is big-endian; this differs from MD4 which is little-endian.

A detailed algorithmic description of SHA-1 can be found in [15]. Compared to 128-bit hash functions, the 160-bit hash-value of SHA-1 provides increased security against brute-force attacks. SHA-1 and RIPEMD-160 presently appear to be of comparable strength; both are considered stronger than MD5. In SHA-1, a significant effect of the expansion of 16-word message blocks to 80 words, in the compression function, is that any two distinct 16-word blocks yield 80-word values which differ in a larger number of bit positions, significantly expanding the number of bit differences among message words input to the compression function. The redundancy added by this preprocessing evidently adds strength to the overall function.

Besides testing DSA with SHA-1, in this paper we also provide some results about DSA with MD5 [16] chosen as the hash algorithm; this way it is possible to compare two different solutions, with different degree of robustness and security, but also requiring a different amount of computational power.

C. MD5

MD5 is a widely used cryptographic hash function that generates a 128-bit hash value. Since its initial proposal, MD5 has been adopted in a great variety of applications, especially in the software world, even if in 1991 a security flaw was found, which was later confirmed by further research activities, that led to the promotion of different hash algorithms like SHA-1.

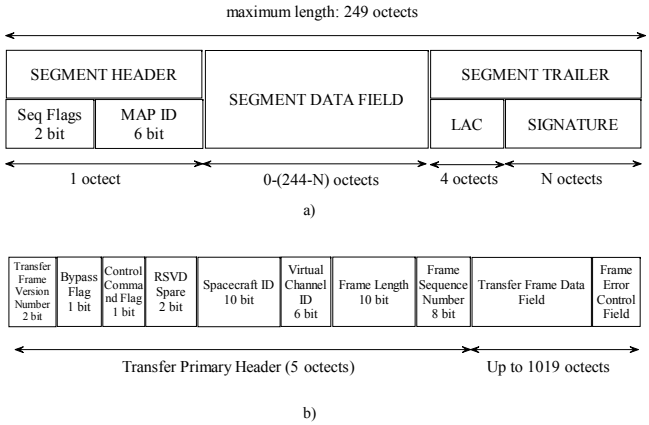

Fig. 1. Structure of: a) the Authenticated TC segment, b) the generic TC Transfer frame used for simulation purposes

MD5 processes a variable-length input message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512 bits; if necessary, the message may be padded so that its length results to be divisible by 512. The core MD5 algorithm operates on a 128-bit state, divided into four 32-bit words initialized to certain fixed constants. Each 512-bit message block operated by the MD5 engine modifies the state variable. The processing of a message block consists of four similar rounds, each one composed of 16 similar operations based on a non-linear function F, modular addition, and left rotation. There are four possible versions of function F; a different one is used at each round.

As requested to any hash function developed for cryptographic applications, even a small change in the input message results in a completely different hash, due to the avalanche effect produced by the MD5 processing.

By simulating both the application of MD5 and SHA-1 with DSA, we will show in the following that the computational effort required to implement DSA makes the selection between SHA-1 and MD5 of minor detail under this point of view; different features, like robustness and security, must be taken into account as a criterion for a proper selection of the most suited hash scheme, according with the level of risk and required security of each specific system or application.

III. Application of DSA to TC segments and Transfer frames

The security algorithms briefly outlined in the previous section have been applied to authenticate TC segment data structures and general TC Transfer frames, defined according with CCSDS specifications. The format of authenticated TC segments and TC Transfer frames, used also for simulation purposes, is depicted in Fig. 1.

The TC segment [17] header comprises two fields: a Sequence Flags field of 2 bits, and a Multiplexed Access Point (MAP) ID field, 6 bit long. The total length of the TC segment header is consequently fixed to 1 octect. The segment data field can have a variable length between 0 and (244-N) octects,

where N represents the total length, in octects, of the signature applied to the TC segment. The signature is part of the segment trailer, together with a Logical Authentication Channel (LAC) field of 4 octects. The maximum length of a TC segment, from which the upper limit on the length of the data field is derived, is fixed to 249 octects.

General TC Transfer frames [18] comprise several fields: a Flags field of 16 bits, a frame header of 24 bits, and a data field of up to 1019 octects (including an optional Frame Error Control field). The Flags field includes 10 bits of Spacecraft ID, whose values have been defined by CCSDS; the Spacecraft ID is obviously fixed for a given mission. The Frame Sequence Number of 8 bits in the frame header must be updated at each frame transmission within the same communication session; once the value 255 is reached, this field will restart from the value 0. Following the Transfer Frame field, that carries the actual payload data, we will add an N-byte field carrying the signature of each TC transfer frame, obtained by the application of DSA with SHA-1 or MD5.

For each TC segment, the SHA-1 and MD5 hash values are computed over all the fields, with the obvious exception of the Signature field. Then, the result of DSA applied to the hash-value is stored in the Signature field and transmitted as the TC authentication tag. Part of the LAC field (30 bits) acts as a counter against so-called replay attacks: for each transmitted TC, the value of the counter is increased, so that the receiver can detect duplicated TCs. As the actual content of TC segments is usually classified, random values are used for simulation purposes, even if, in real systems, most TCs differ only in some bits, and the concrete structure of a telecommand is bound to very strong regulations. At the receiver, the hash-values are re-computed on the received data; by means of the sender's public key, the corresponding (local) DSA signature is compared to the received Signature. If the two signatures are equal, then the TC segment authenticity is verified, and the Telecommand is accepted for further processing, otherwise it is discarded. In a similar way, TC Transfer frames are processed by SHA-1 and MD5 hash algorithms applied over all the frame fields; the resulting signature computed by means of DSA is appended as a tail end to each frame, and verified at the receiver according with the same procedure described above.

Following the DSA specification, each TC segment and frame is authenticated by means of a secret key, i.e. the sender's private key, whereas the TC signature verification is performed by the receiver, using the sender's public key. The sender's private and public keys are defined according with a mathematical relationship that makes all the system work properly. As any asymmetric encryption scheme foresees, no secret key needs to be securely shared between the two parties; the sender's public key must be known at the receiver, but it can be transmitted in the clear form. This may be a valuable feature in the space context, where the key pair may be subjected to frequent changes due to different mission phases; the delivery of the public key in a clear form greatly simplifies key management.

Obviously, in order to increase the security and robustness of the system, it's a universally recognized good practice to periodically change the keys used, for example according with

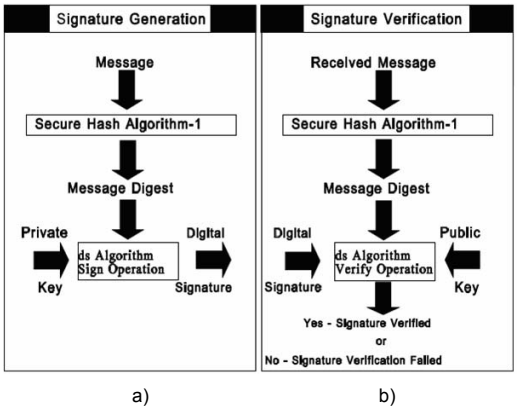

Fig. 2. Functional diagram of the authentication processor: a) sender, b) receiver

the frequency and the duration of the communication sessions between the ground station (the sender) and the spacecraft (the receiver), or on the basis of the amount of data globally exchanged. This should be accomplished by means of a proper key management infrastructure, that is always a core element of any asymmetric scheme; however, this is not discussed in this paper, as we assume that such an infrastructure is already available for use in the context of interest. At present, several research activities are ongoing on this topic, see [19] as an example. Fig. 2 shows the functional diagram of the DSA/SHA-1 authentication processor that has been implemented for simulation purposes. In the signature generation phase, the message (i.e. the TC segment) is processed by the SHA-1 generator block, and the digest obtained is passed to the Digital Signature generation algorithm that uses the sender's private key. By this way, the TC segment is authenticated and signed before transmission. At the receiving end, the signature verification process requires to compute the SHA-1 digest on the received TC segment; then, the DSA signature received is verified by the public key associated to the sender's private key. The received digest is then compared to the locally generated one: if they are equal, the TC segment is accepted as valid. When using DSA with MD5, the SHA-1 processing block is simply replaced by the MD5 engine.

IV. Simulations and Results

The DSA/SHA-1 and DSA/MD5 authentication processors have been implemented on a microcontroller, with the aim of evaluating their computational requirements and performances, when applied to TC segments and Transfer frames simulated according with the previous discussion. The device selected for experimental evaluations is a dsPIC by Microchip [20], based on a modified Harvard 16 bit architecture, with up to 40 MIPS (Million Instructions Per Second) processing speed. The device is equipped with a flash program memory of 256 kbytes, a 30720 byte RAM, and 85 I/O (Input/Output) ports. By means of a proper Integrated Development Environment and a USB communication interface, the generation of TC segments and frames, their authentication and verification have been implemented through a program written in the C language, and executed by the programmable device. The adoption of a common and specific platform for testing different algorithms should ensure a fair comparison among their performance.

A. Application of SHA-1 and MD5 to TC segments and Transfer frames

As a first result, we are interested in evaluating the variation of the computational resources required by the authentication processor, according with the length of the data to be authenticated. The SHA-1 function generates an output value of 160 bits (i.e. 20 octects), MD5 outputs a 128-bit hash (i.e. 16 octects); in both cases the DSA is applied with a 1024 bit key, to produce a final signature of 40 octects. From this configuration, it follows that the length of the simulated TC segments Data Field can vary between 0 and 204 octects. In the case of TC Transfer frames, several lengths of the Transfer Frame Data field have been simulated, ranging from 100 to 1000 octects, by step of 100 octects. Each simulation has been repeated as many times as needed to get reliable results.

Preliminary tests have shown that the number of instruction cycles required for DSA execution is far greater than the number of cycles required to compute only the SHA-1 or MD5 hash on the input TC segments or frames. For this reason, in order to effectively evaluate the relationship between the length of the TC segments and frames, and the number of operations performed by the dsPIC, we first considered the hash computation processes only. The average results obtained on TC segments by the application of SHA-1 and MD5 are shown in Fig. 3: we can observe a step behavior, that can be explained by resorting to the hash algorithms functional architecture. As shown, when processing TC segments having the maximum admitted length of 204 octects, the number of instruction cycles requested by SHA-1 is around $18 \cdot 10^3$, i.e. an average computation time of 0.45 ms per TC; this value decreases to $9 \cdot 10^3$ instruction cycles, i.e. around 0.2 ms per TC segment, when the MD5 hash algorithm is applied.

SHA-1 processes the input data by 512-bit blocks; each time the amount of input data is higher than an integer multiple of this value, the number of operations performed by the algorithm increases of a fixed amount. Moreover, for each range of input data lengths corresponding to the same number of instruction cycles performed by the SHA-1 processor, a subrange exists, where the number of cycles executed is locally higher. This is due to the padding procedure applied by SHA-1 to get an amount of the input data that is a multiple of 512 bits. The amount of padding needed decreases as the length of the input data field increases, so that additional operations due to padding are performed only for the length values in each subrange. This effect should be carefully evaluated in the space context, where computational resources are in general considered scarce. A similar step behavior happens also in the case of MD5, that adds a padding to the input message in order to get an integer number of 512-bit blocks. With respect to SHA-1, it is possible to see a reduced amount of instruction cycles required (almost -40%) at a parity of the input message Data field length, and also a memory occupation that decreases by 20% with respect to SHA-1. On the other hand, MD5 shows a reduced avalanche effect, if compared to SHA-1, which makes its overall strength questionable.

If we consider the joint application of SHA-1 or MD5 with DSA, in the case of TC segments with Data field length

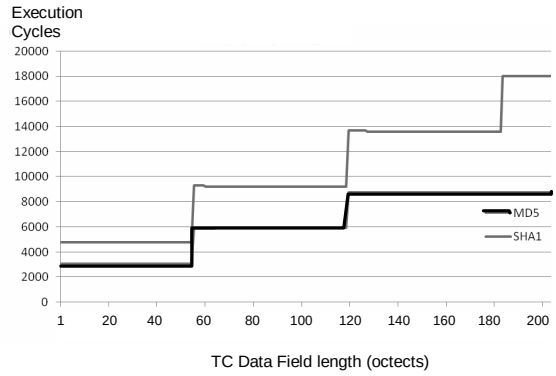

Fig. 3. SHA-1 and MD5 instruction cycles for different TC segment Data Field lengths

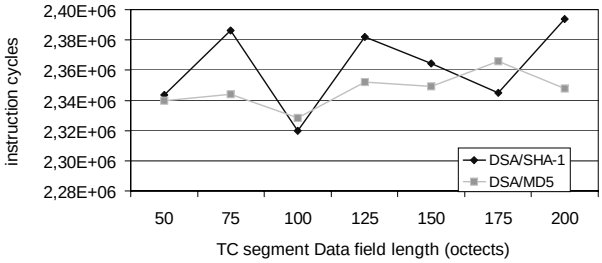

Fig. 4. Instruction cycles required by DSA with SHA-1/MD5, for different TC segment Data Field lengths

varying from 50 to 200 bytes, the number of instruction cycles required grows about two orders of magnitude, as depicted in Fig. 4, thus confirming the strong dominance of DSA over the hash algorithm, from the point of view of computational resources needed.

Fig. 5 shows the amount of instruction cycles required by DSA with SHA-1 and MD5, when applied to TC Transfer frames of length varying from 100 to 1000 octects. On average, DSA with SHA-1 requires higher computational resources than DSA with MD5, especially for frame lengths greater than 800 bytes. However, there may be some cases, like the one corresponding to a TC Transfer frame length of 400 bytes in the figure, in which the situation is reversed, and DSA with MD5 may need more computational cycles than DSA with SHA-1. These conditions, together with the great variability observed in the results obtained through simulations, are basically due to the DSA signature generation process: when the selected random integer k has a big value, the amount of instruction cycles performed grows dramatically. This makes DSA impact dominant over the hash algorithm applied, under the point of view of computational complexity.

B. Authentication in presence of residual channel errors

In a second set of experimental tests, we are interested in evaluating the effects of residual errors, due to the communi-

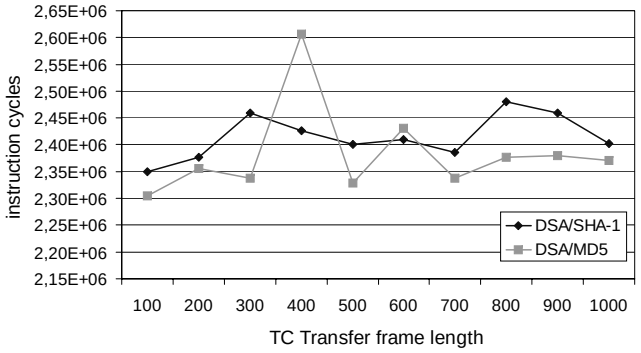

Fig. 5. Instruction cycles required by DSA with SHA-1/MD5, for different TC Transfer frame lengths

TABLE I
AVERAGE PERCENT CORRUPTION RATES: TC SEGMENT LENGTH = 50, 100, 200 OCTECTS; AWGN CHANNEL

p_B	10^{-4}	10^{-5}	10^{-6}
Data (50)	0.848	0.132	0.008
Data (100)	4.532	0.9	0.032
Data (200)	11.54	2.252	0.104
Signature (50)	2.96	0.592	0
Signature (100)	2.808	0.624	0.036
Signature (200)	2.816	0.628	0.052
Data + Signature (50)	0.032	0.028	0
Data + Signature (100)	0.16	0	0
Data + Signature (200)	0.356	0.012	0

cation channel, on the correct verification of the TC segments and Transfer frames at the receiver. First of all, we assume a simplified scenario, in which no error correction strategies are applied to the TC segments during transmissions. It is known that, on the contrary, robust Forward Error Correction (FEC) techniques are usually adopted in real world space communications. However, in the case of harsh environments (like Deep Space missions), it is possible that residual errors are not completely detected or corrected by the FEC algorithms. Such errors could affect the signature verification process at the receiver, thus causing the rejection of valid TCs, or, in the worst case, the validation of malicious TCs.

First we consider an Additive White Gaussian Noise (AWGN) communication channel, characterized by sparse errors that determine an average bit error probability p_B of 10^{-6}, 10^{-5}, and 10^{-4}. These values of p_B are applied to sequences of 5000 TC segments of different lengths, ranging from 50 to 100 to 200 octects; p_B's of 10^{-5} and 10^{-4} have been simulated in the case of TC Transfer frames of length varying from 200 to 1000 octects by steps of 200 bytes. Each simulation is run five times, in order to ensure the significance of the results obtained.

For each simulated communication session, we consider the number of TC segments and Transfer frames that are corrupted in their Data field only, in their Signature field only, and in both their Data and Signature fields. In this last case, we also verify if the corrupted Signature corresponds to the DSA/SHA-1 Signature computed over the corrupted Data: this would represent the worst case for the scenario under test, as the received Telecommand would be accepted as authentic, even if corrupted by errors on the channel, or even by malicious attacks by an opponent.

The results obtained through these simulations are reported in Table I for the TC segments; samples of the resulting performances, for each TC segment length, are also shown in Fig. 6. Similar behaviors have been obtained by simulations in the case of TC Transfer frames.

As reasonable and expected, the average percent corruption rate is higher for that field representing the greatest part of the TC segment. Thus, in the case of TC segments that are 50 octects long, the 40 octects of the Signature field correspond to the 80% of the total amount of data, and the Signature field is the one showing the highest corruption rate. Moving to 200-octect long TC segments, the situation is reversed: the Data field constitutes the 80% of the overall TC segment length, and it shows the highest percent corruption rate. In all the situations examined, the incidence of the case corresponding to the joint corruption of the Signature and Data fields is quite negligible; in each of these cases, the corrupted Signature does not correspond to the Signature computed over the corrupted Data field, i.e. the system does not show collisions.

In a last run of simulations, we consider the transmission of authenticated TC segments over a burst channel. Again, we assume no error correction strategies are applied to the data. Though different from the channel error probability in the case of sparse errors, we can define an input error probability p_B also in this case, as the average number of bit in error, because of bursts, over the total number of bits simulated. Fig. 7 shows the percent amounts of corrupted TC segments in the case of 200-octect long TCs: as a preliminary evaluation, we can say that the system performs better in the case of a burst channel, with respect to an AWGN channel, at a parity of the average bit error probability. Again, no collisions have been revealed when both the Data and Signature fields are corrupted. A common feature, in all the situations examined, is the fixed length of the authentication tag: this causes a higher incidence of the authentication overhead when shorter TCs are transmitted. Alternative and more recent solutions could be, on the contrary, able to provide authentication tag lengths tunable with the TC length.

V. CONCLUSION

This paper provided some numerical figures related to the performance evaluation of the Digital Signature Algorithm with SHA-1 or MD5 as the hash engine, when applied to the authentication of Telecommand data structures, segments or Transfer frames, in space contexts. In order to test the amount of computational resources required by each security scheme, we proposed their implementation on a commercial dsPIC; this could be seen also as a prototypical realization. The DSA with SHA-1 authentication scheme, which seems to emerge as the favourite solution within Space Agencies, has proved to be robust in presence of residual errors on the channel; this is a valuable issue, especially in harsh space environments. Further developments of this work will concern the implementation of

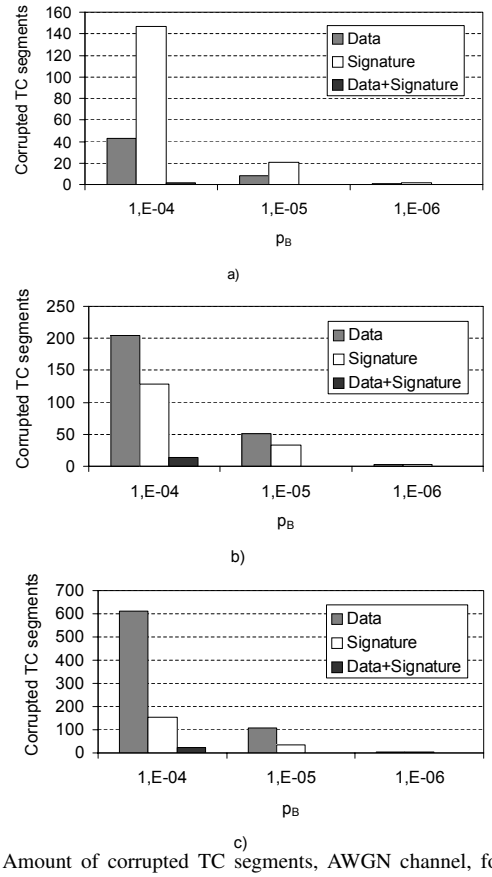

Fig. 6. Amount of corrupted TC segments, AWGN channel, for different lengths: a) 50 octects, b) 100 octects, c) 200 octects

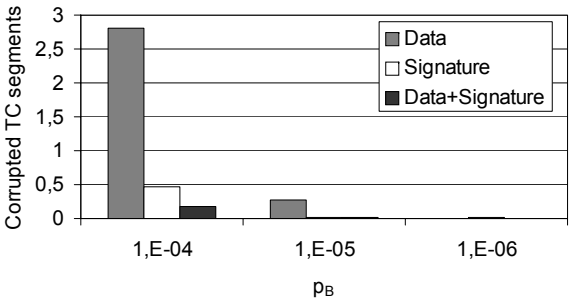

Fig. 7. Average percent corruption rates of TC segments, length = 200 octects, burst channel

alternative schemes on the same hardware platform and their thorough comparison.

REFERENCES

[1] I. Aguilar Sanchez: *Securing ESA Space Missions: Some Conclusions after Recent Generic and Project-Specific Security Activities*, in Proc. of 4th ESA International Workshop on Tracking, Telemetry and Command Systems for Space applications TTC 2007, Darmstadt (Germany), 11-14 Sep. 2007, pp. 753-760.

[2] D. Fischer, M. Merri, T. Engel: *Introducing a Generic Security Extension for the Packet TM/TC Protocol Stack*, in Proc. of 4th ESA International Workshop on Tracking, Telemetry and Command Systems for Space applications TTC 2007, Darmstadt (Germany), 11-14 Sep. 2007, pp. 761-768.

[3] Consultative Committee for Space Data Systems: "The Application of CCSDS Protocols to Secure Systems," Report Concerning Space Data System Standards, CCSDS 350.0-G-2, Green Book, Washington, D.C., Aug. 2005.

[4] ESA Space Operations Center, Contract No. 17462/03/D/CS, "Encryption for Space, Present Scenario, Performance and Software Efficient Applications," Final Report, July 2004.

[5] F. Chiaraluce, E. Gambi, S. Spinsante: *Efficiency Tests Results and New Perspectives for Secure Telecommand Authentication in Space Missions: Case-Study of the European Space Agency*, ETRI Journal, Vol. 27, No. 4, pp. 394 - 404, Aug. 2005.

[6] H. Weiss (NASA/JPL/SPARTA): "Authentication/Integrity Algorithm Issues White Paper", CCSDS Authentication Algorithm Trade Study, White Paper, Aug. 2005 (available at: http://public.ccsds.org/sites/cwe/sea-sec/default.aspx).

[7] National Institute of Standards and Technology (NIST): "Digital Signature Standard (DSS)", Federal Information Processing Standards Publication (FIPS PUB 182-6), Jan. 2000.

[8] National Institute of Standards and Technology (NIST): "Secure Hash Standard", Federal Information Processing Standards Publication (FIPS PUB 180-1), Apr. 1995.

[9] Internet Engineering Task Force (IETF): "The MD5 Message-Digest Algorithm", RFC 1321, Apr.1992 (available at: http://www.faqs.org/rfcs/rfc1321.html).

[10] National Institute of Standards and Technology (NIST): "DES Modes of Operation", Federal Information Processing Standards Publication (FIPS PUB 181), Dec. 1980.

[11] R. Housley, D. Whiting, N. Ferguson: *Counter with CBC-MAC (CCM) - AES Mode of Operation*, Submission to NIST (available at: http://csrc.nist.gov/encryption/modes/proposedmodes/ccm/ccm.pdf).

[12] A. Adelsbach, C. Harpes, G.P. Calzolari, S. Zatti, and D. Fischer: *Practical Evaluation of cryptographic Configurations for Packet TM/TC*, in Proc. of 4th ESA International Workshop on Tracking, Telemetry and Command Systems for Space applications TTC 2007, Darmstadt (Germany), 11-14 Sep. 2007, pp. 650-657.

[13] X. Wang, D. Feng, X. Lai, and H. Yu: *Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD*, Report 2004/199, Cryptology ePrint Archive, 2004.

[14] M. Bellare: *New Proofs of NMAC and HMAC: Security without Collision-Resistance*, Advances in Cryptology - Crypto 2006, LNCS 4117, Springer-Verlag, 2006.

[15] A. Menezes, P. van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996 (available at: www.cacr.math.uwaterloo.ca/hac).

[16] R. Rivest: *The MD5 Message-Digest Algorithm*, IETF Network Working Group, RFC 1321, April 1992.

[17] Consultative Committee for Space Data Systems: "TC Synchronization and Channel Coding", CCSDS 231.0-R-1 Red Book, Washington, D.C., USA, Jul. 2002.

[18] Consultative Committee for Space Data Systems: "TC Space Data Link Protocol", CCSDS 232.0-B-1 Blue Book, Washington, D.C., USA, Sep. 2003.

[19] C. Gorecki, A. Weigl, R. Rathje: *Key Synchronization for TT&C Networks*, Proc. of 4th ESA International Workshop on Tracking, Telemetry and Command Systems for Space applications TTC 2007, Darmstadt (Germany), 11-14 Sep. 2007, pp. 681-685.

[20] Microchip dsPIC33FJ256GP710 DSC, www.microchip.com.

Susanna Spinsante was born in Osimo, Italy, in 1976. She received her Laurea degree (summa cum laude) in Electronic Engineering in 2002 from the Università di Ancona, Italy and, in 2005, the Ph.D. in Electronic Engineering and Telecommunications at the Università Politecnica delle Marche. Her main research interests are related to security and encryption aspects in communication networks, multimedia applications over IP, coding and audio/video applications. She is a member of IEEE.

Ennio Gambi was born in Ancona, Italy in 1961. He received his Laurea degree in Electronic Engineering from the Università di Ancona, Italy, in 1986. In 1992 he joined, as a researcher, the Dipartimento di Elettronica ed Automatica of the Università di Ancona, where he is currently the lecturer for the course of Telecommunication Systems. He is presently working on radio communication systems, multiple access techniques, video signal processing. He is a member of IEEE.