PERFORMANCE ANALYSIS OF THE CONFIDENTIALITY SECURITY SERVICE IN CAN

Miguel León Chávez[1], and Francisco Rodríguez Henríquez[2]

[1]Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
14 Sur y Av. San Claudio, CP 72570, Puebla, México
Tel. (52) 222 229 55 00 ext. 7213 Fax (52) 222 229 56 72
E-mail: mleon@cs.buap.mx
[2]CINVESTAV-IPN
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508, Col. San Pedro Zacatenco
México, D.F. 07300
Tel: (52) 52 55 5747 3800 ext. 6570 Fax: (52) 555 747-7002
E-mail: francisco@cs.cinvestav.mx

Abstract: This paper addresses the security service infrastructure included in the Controller Area Network (CAN), proposing the incorporation of the confidentiality service for that kind of network. Regarding security currently CAN only supports a special kind of service, namely safety, of data transfers for error detection, signaling, and self checking. Nevertheless, as CAN has become more diverse, complex and integrated into other kind of networks, it must provide higher security services, such as confidentiality, quite especially for its bus which is the most attack-prone point on CAN. Taking into account, on one hand, the security services defined by ISO and, on the other hand, the security services defined by CAN, this paper proposes to incorporate the confidentiality service to CAN based on a lightweight symmetric stream cipher algorithm, such as RC4 or A5/1 GSM. Finally, this paper presents the performance analysis of both algorithms, and recommends using RC4 because it consumes much less clock cycles that A5/1 for encrypting the CAN data frames.

Keywords: Fieldbus, Security.

1. INTRODUCTION

Controller Area Network (CAN) (Bosch, 1992; ISO 11898) is a serial communication protocol, which supports distributed real-time control, and it is used to connect engine control units, sensors, anti-skid-systems, etc. in automotive electronics, for example.
This network defines two OSI layers, the physical, and the data link (Logical Link Control and Medium Access Control) layers. At the higher layers there are several approaches such as CAL, CANOpen, DeviceNet, SDS, CANKingdom, and TT-CAN.

Up to now the security in CAN has only considered security for data transfer (i.e. error detection, error signaling, and self-checking). However, nowadays there are some legitimate security concerns about the possibility of non-authorized gaining access to the common communication channel in order to launch passive/active attacks.
As distributed systems based on CAN become more and more diverse, complex and integrated into other kind of systems, the probability of potential attacks to the security of the network increases in the same rate. Hence, the communication protocol must be

updated and enhanced in order to prevent/thwart that kind of security attacks.

This paper discusses the security in CAN according to the security services defined by ISO. The paper proposes then to incorporate the confidentiality service into the CAN protocol based on a lightweight cryptographic stream cipher, such as RC4 or A5/1 GSM. Finally, this paper presents the performance analysis of both stream cipher algorithms for different data sizes in a CAN data frame.

The remaining part of this paper is organized as follows: Section 2 presents and discusses ISO security services with regard to CAN; section 3 presents the CAN protocol; section 4 proposes the confidentiality service for the CAN protocol, and presents some theoretical results using RC4 and A5/1. Finally some future work directions and conclusions are drawn in section 5.

2. ISO SECURITY SERVICES

The Security Architecture of the OSI Reference Model (ISO 7498-2) considers five main classes of security services: authentication, access control, confidentiality, integrity and non-repudiation. These services are defined as follows: The *authentication* service verifies the supposed identity of a user or a system. The *access control* service protects the system resources against non-authorized users. The *confidentiality* service protects the data against non-authorized revelations. Confidentiality has an essential role on cryptographic systems. The *integrity* service protects the data against non-authorized modifications, insertions or deletions. The *non-repudiation* service provides certain protection against the sender of a message that refuses to be it, or against the receiver of a message that denies to have received it.

All those security services are not very likely to be useful on the context of CAN (Morris and Koopman 2003; León and Rodríguez, 2004) because of the following considerations. The authentication, of both nodes and messages, and the non-repudiation services are not needed in CAN because the CAN nodes do not make use of any information about the network configuration, e.g. node addresses. In CAN, all the messages have been assigned a unique identifier which is used as a static priority for bus access. The identifier does not indicate the destination of the message, but describes the meaning of the data, so that all the nodes in the network are able to decide by message filtering whether the data is to be acted upon by them or not. As a consequence of the concept of message filtering any number of nodes can receive and simultaneously act upon the same message. Message filtering is based upon the whole identifier, although optional mask registers may be used to select groups of identifiers to be mapped into the attached receive buffers.

The access control service may be implemented at the higher layers protocols (e.g. application) (León and Rodríguez, 2004). As it was mentioned before, at network configuration time, all the messages have been assigned a unique identifier. Therefore, the higher layer protocols based on CAN should provide this service by using some mechanisms of user identification, such as logging and password, to avoid access to the CAN based system configuration from non-authorized human operators.

CAN provides users with a special kind of service for data transfer, namely safety service, which includes the following procedures: error detection, error signaling, and self-checking.

For error detecting the following measures are taken into account: Monitoring (transmitters compare the bit levels to be transmitted with the bit levels detected on the bus), Cyclic Redundancy Check (CRC), Bit stuffing, and Message Frame Check.

Nevertheless, these procedures do not provide the integrity security service, which can be achieved by using cryptographic mechanisms such as one-way hash functions. Those functions accept a variable-size message as input and produce a fixed-size code, called the hash code. The hash code is a function of all the bits of the message and provides an error detection capability. A change to any bit or bits in the message results in a change on the resulting hash code.

Finally, CAN does not provide the confidentiality service. All the data transfers are made in plaintext. Therefore, in order to avoid possible passive and active attacks from intruders that have managed to gain access to the bus, CAN must instrument a data-confidentiality service via feasible encryption/decryption schemes.

To provide confidentiality, nodes may encrypt their contents using a random session key and a symmetric crypto-algorithm specially tailored for constrained environments, such as those found in CAN where the size of the messages may be from 0 to 8 bytes.

In general, strong public key cryptography is an expensive fancy solution for constrained environments. If for a given CAN based system, public key cryptography solutions become too expensive, we can still design limited security schemes for CAN at a cheaper price. For instance, we can use a security scheme based on a one-way hash function optimized for heavily constrained environments (Sarma, *et. al.*, 2002).

Encryption of the hashed message can then be achieved by using lightweight symmetric stream ciphers. In this paper we propose the usage of two lightweight stream ciphers, such as RC4 and A5/1 GSM, as the main building block needed to implant the confidentiality service for the CAN protocol.

3. CAN PROTOCOL

CAN defines the CSMA/BA (Carrier Sense Multiple Access with Bitwise Arbitration) protocol at the MAC sub-layer. The information on the bus is sent in fixed format messages of different but limited length (0-8 byes). As it was mentioned in section 2, each message has assigned a unique identifier which is used as its assigned priority. The length of the identifier is equal to 11 bits in the standard format and is equal to 29 bits in the extended format. At network operation time, when the bus is detected as idle, each CAN node begins to transmit its highest

priority message whilst monitoring the bus. The most significant bit (MSB) of the identifier field is transmitted first. The message is coded so that if a node transmits a recessive bit (logical 1) but monitors a dominant bit (logical 0) then a collision is detected. The node knows that its message is not the highest priority message in the network, stops transmitting and waits for the bus to become idle. Meanwhile, the node becomes a receiver of the message. Each receiver node performs the message filtering to determine whether the data are relevant. If so, the data are accepted, otherwise are ignored. If there is no collision in the least significant bit (LSB), the node transmits the body of the message.

An analysis that limits the response time of all the CAN messages including the lowest priority message is presented in (Tindell, *et. al.*, 1995). Other analysis providing an analytical method for computing the worst case deadline failure probability is presented in (Navet, *et. al.*, 2000). These analyses allow limiting the end-to-end delay (León and Thomesse, 2000).

The confidentiality service can be implemented in all the message transfers. However the encryption/ decryption process will affect the end-to-end delay.

As one first step to implement the confidentiality service in the CAN protocol, this paper analyses the computing cost for encrypting several CAN data frames.

4. CONFIDENTIALITY SERVICE FOR CAN

This section presents RC4 and A5/1 GSM, as well as their performance analysis for encrypting the data field of the CAN frame.

4.1 RC4

RC4 is a symmetric-key stream cipher that was developed in 1987 and kept as a trade secret by RSA Data Security, Inc. In September 1994, the algorithm was anonymously posted on the Internet, and since then it has been studied and used widely in academic circles and is now available for public analysis. RC4 is commonly used as the default cipher for SSL/TLS (Secure Sockets Layer/Transport Layer Security) connections.

RC4 uses a 2048-bit key-length needed to initialize a 256-byte state table, called the S-box. If a given application wishes to use a shorter key, then that shorter key is repeated as many times as needed to fill the 2048-bit key. Typically, RC4 is used in a mode where 16-byte (128-bit) keys are repeated sixteen times.

Once the S-box is initialized with the key, the RC4 algorithm enters in a loop that updates the S-box and generates a byte of pseudo-random keystream. That pseudo-random keystream is XOR-ed with the plaintext message to produce the ciphertext.

Thus, the RC4 algorithm can be broken into two phases: initialization and operation. In the first phase the 256-byte state table *S* is populated using the key *K* as a seed. Once the state table is setup, it continues to be modified in a regular pattern as data is encrypted. In the second phase the algorithm outputs a ciphertext which is the input message XOR-ed byte by byte with the values of the pseudo-random keystream. As long as the same initial S-box is used, both of them encryption and decryption processes are completely symmetric.

4.2 A5/1 GSM

A5 is a stream cipher algorithm used by GSM to protect the over-the-air privacy of telephone conversations. The original algorithm was renamed A5/1 and there exist several versions: A5/0 (no encryption at all), A5/1, A5/2 (weaker than A5/1), and A5/3 (the newest version based on Kasumi block cipher).

The approximate design of A5/1 was leaked in 1994, and its exact design was reverse engineered in 1999.

A GSM conversation is sent as a sequence of frames every 4.6 milliseconds. Each frame contains 114 bits representing the digitized A to B communication. Each conversation can be encrypted by a new session key K. For each frame, K is mixed with a publicly known frame counter F_n, and the result serves as the initial state of a generator which produces 228 pseudo random bits, which are XOR-ed by the two parties with the 114+114 bits of the plaintext to produce the ciphertext (Biryukov, *et. al.*, 2000).

A5/1 uses three linear feedback shift registers, which are clocked in a stop/go fashion using the following majority rule: Each register has a single clocking tap; each clock cycle, the majority function of the clocking taps is calculated and only those registers whose clocking taps agree with the majority bit are actually clocked.

In the first step, the registers are zeroed and then clocked for 64 cycles (ignoring the stop/go control). During this period each bit of K (from LSB to MSB) is XOR-ed in parallel into the LSB's of the three registers.

In the second step, the registers are clocked for 22 additional cycles (ignoring the stop/go control). During this period the successive bits of F_n (from LSB to MSB) are again XOR-ed in parallel into the LSB's of the three registers.

In the third step, the registers are clocked for 100 additional clock cycles with the stop/go clock control but without producing any output.

In the fourth step, the registers are clocked for 228 additional clock cycles with the stop/go clock control in order to produce the 228 output bits. At each clock cycle, one output bit is produced as the XOR of the MSB's of the three registers.

4.3 Performance analysis of RC4 and A5/1 in CAN

The RC4 and A5/1 GSM algorithms have been programmed in C code, and then we have obtained the assembler code for the Intel MCS®96 microcontroller family, such as the 87C196CB with integrated CAN 2.0 serial interface, and up to 16MHz operation.

Then the time required for encrypting the data field of the CAN frames has been calculated, with a key size of 8 bytes and with the identifier field as the frame counter F_n. This calculation has been made adding the number of clock cycles of each assembler

instruction during the phases/steps of both algorithms for different data size (1-8 bytes). Table 1 shows these calculations.

Table 1 Clock cycles required for executing the RC4 and A5/1 algorithms in assembler code of the 87C196CB.

Bytes	RC4	A5/1
1	120,942	1,026,284
2	121,124	1,064,149
3	121,646	1,100,152
4	122,256	1,138,002
5	122,898	1,173,074
6	123,556	1,210,949
7	124,226	1,246,021
8	124,904	1,282,983

These clock cycles can be multiplied by the system clock frequency to obtain the time required for encrypting/decrypting the data field of the CAN frame. For example, if the clock frequency is 16 MHz then the encryption time, for the RC4 algorithm, goes from 7.5 ms to 7.8 ms for 1 byte to 8 bytes, respectively, and the encryption time for the A5/1 algorithm goes from 64.1 ms to 80.1 ms for 1 byte to 8 bytes, respectively.

The results shown in Table 1 are due to different facts:

1. Software/hardware implementation

As it was mentioned before both algorithms have been programmed in C code.

However, A5/1 GSM was specifically designed for an efficient implementation in hardware, since it uses three linear feedback shift registers and four XOR gates.

On the other hand, RC4 is based on the use of a random permutation, and the cipher can be expected to run very quickly in software (Stallings, 2003).

2. Initialization phase

Most of the clock cycles for ciphering the data field of the CAN frames are consumed during the initialization phase. For example, RC4 takes in this phase from 99.28% to 95.49% for 1 byte to 8 bytes, respectively.

For the A5/1 algorithm, this phase (steps 1, 2 and 3) takes from 99.61% for 1 byte to 97.56% for 8 bytes.

The initialization phase is required by RC4 in each encryption because it is strongly recommended that no two messages are encrypted with the same key. Otherwise the message can usually be broken. If the two encrypted messages are XOR-ed together, the result is the XOR of the original messages (Dawson and Nielsen, 1996).

Due to the fact that the overhead introduced by the initialization phase is too large for both algorithms, we propose to include the concept of a *session key*. This way, we propose to open a new session (and thus to generate a new key) each time that the CAN protocol is initialized by the application. As long as the session is still active that same session key will be used to encrypt all CAN frames.

This feature will require a specific protocol among the parties involved in order to solve issues related to key generation and management. Such issues include: key generation and renovation; opening and closing sessions, etc.

5. CONCLUSION

This paper has discussed the OSI security services regarding the Controller Area Network protocol. The paper has shown that some security services are not needed in the context of CAN, such as authentication, and non-repudiation.

The access control service may be implemented at the upper layers.

Even though CAN provides users with a special kind of safety service for data transfers, this paper recommends the usage of cryptography mechanisms to provide the integrity service, such as hash functions.

Finally, this paper has presented the performance analysis of two lightweight cryptographic stream ciphers, RC4 and A5/1 GSM, in order to implement the confidentiality service into the CAN protocol. The RC4 and A5/1 algorithms have been programmed in C code, and translated to assembler code for the Intel MCS®96 microcontroller family. In this way, we have computed the clock cycles required for encrypting/decrypting the data field of the CAN frames, with this result it is possible to compute the time required for encryption/decryption according to the clock frequency of the microcontroller been used.

It can be noted that the encryption/decryption process hardly affect the end-to-end delay of the CAN protocol. Nevertheless, in any application there must exist a tradeoff between real-time and security constraints on the network, and the developer must decide what level of tradeoff is required by the application.

On the other hand, RC4 has the characteristic that no two messages must be encrypted with the same key. However we may relax this condition due to the already mentioned huge overhead associated to the generation of new keys each time that a frame has to be transmitted. Taking this design decision will imply the definition of a specific session protocol and this is our future research work, as well as the performance analysis using the A5/3 cipher.

ACKNOWLEDGMENTS

This research project has been partially supported by the CONACyT project 45306.

REFERENCES

Biryukov, A., A. Shamir and D. Wagner (2000). Real Time Cryptanalysis of A5/1 on a PC. In *Fast Software Encryption Workshop*, April 10-12, New York City.

Bosch, R. GmbH. (1992). CAN Protocol Specification V2.0 (A, B).

Dawson, E., and L. Nielsen (1996). Automated Cryptanalysis of XOR Plaintext Strings. *Criptologia*, **vol.** XX, No. 2.

ISO 7498-2 (1989). International Organization for Standardization. Information processing systems-Open Systems Interconnection-Basic Reference Model - Part 2: Security Architecture.

ISO 11898 (1992). International Organization for Standardization. Road Vehicles-Interchange of Digital Information-Controller Area Network (CAN) for High Speed Communication.

León, M., and J.P. Thomesse (2000). Fieldbuses and Real-Time MAC Protocols. In *4th IFAC International Symposium on Intelligent Components and Instruments (SICICA'2000)*, Buenos Aires, Argentina, pp 51-56.

León, M., and F. Rodríguez (2004). SDL Specification of a Security Architecture for the IEC 61158. In *11th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'2004)*, Salvador da Bahia, Brazil, April 5-7.

Morris, J., and P. Koopman (2003). Critical Message Integrity over Shared Network. In *5th IFAC Conference on Fieldbus Systems and their Applications*, Aveiro, Portugal, July 7-8, pp 145-151.

Navet, N., Y-Q. Song, and F. Simonot (2000). Worst-case deadline failure probability in real-time applications distributed over controller area network. *Journal of Systems Architecture*, **vol.** 46, pp 607-617.

Sarma, S.E., S.A. Weis and D.W. Engels (2002). Low Cost RFID and the Electronic Product Code. In *Cryptographic Hardware and Embedded Systems-CHES, Lecture Notes in Computer Sciences*, Springer-Verlag, Heiselberg, Germany.

Stallings, W. Cryptography and Network Security: Principles and Practice. Prentice Hall, Third Ed., 2003.

Tindell, K., A. Burns and A. Wellings (1995). Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, **vol.** 3, No. 8, pp 1163-1169.