Side Channel Analysis of an Automotive Microprocessor

Mark D. Hamilton†, Michael Tunstall‡, Emanuel M. Popovici†, and William P. Marnane††

†*Dept. of Microelectronic Engineering, University College Cork, Cork, IRELAND.*
m.d.hamilton@student.ucc.ie
e.popovici@ucc.ie

‡*Dept. of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol BS8 1UB, UNITED KINGDOM.*
tunstall@cs.bris.ac.uk

††*Dept. of Electrical & Electronic Engineering, University College Cork, Cork, IRELAND.*
liam@eleceng.ucc.ie

Abstract — **Automotive electronics are becoming ever more complex. The quantity and sensitivity of data that is transmitted throughout a car is expected to continue to increase in coming years. On board computers can contain information about the car and dictate how the car behaves. This means that these systems need to be secure to protect the data held within such system so that the behaviour cannot be modified by an unauthorised user. This paper outlines a method for implementing AES encryption over a CAN bus and how such a system could be attacked using Correlation Power Analysis.**

Keywords — **CAN bus, Advanced Encryption Standard (AES), Correlation Power Analysis (CPA).**

I Introduction

With the increase in complexity of car electronic systems it is likely that protecting information held within the embedded systems present within a car will become ever more important. If an adversary is able to send arbitrary messages to a microprocessor, or authenticate to a chip and modify code on the chip, the behaviour of the car may be changed. This can change the behaviour of a car which may make it dangerous to drive, and the liability of the car manufacturer in this situation is unclear.

This paper describes an implementation of sending data over an information bus, using the AES (Advanced Encryption Standard) algorithm. The paper also presents the results of a correlation power analysis that was performed on the system. A key area for use of this encryption method is in the protection of car registration information contained on an in car chip. This method can be used to encrypt any data that has to be sent over a CAN bus [4] and has no affect on other CAN messages being sent.

We detail how this system can be broken using Correlation Power Analysis (CPA) [2], a more complete version of Differential Power Analysis [5], where we derive information by computing the correlation with predicted intermediate states of the AES implementation and the instantaneous power consumption. This demonstrates that side channel attacks need to be considered when implementing a secure system in a car.

The attack described in this paper acts as a proof of concept for two attacks that could be applied to microprocessors used in automotive systems. If an adversary is able to send challenges to a chip to attempt to authenticate to the microprocessor, the attack could potentially derive the secret key used and subsequently authenticate to the microprocessor. This could allow an adversary to modify the code being run on the microprocessor. The other scenario would be to repeatedly send random messages to a chip which it would decipher. The secret key could then be derived by observing the power consumption and then send arbitrary messages to the microprocessor, and any other microprocessors on same bus, over the CAN bus. Only the second scenario is implemented but the same attack strategy would apply in both cases.

The use of power analysis is typically considered to apply to smart cards and not considered to be a

threat to other secure microprocessors. The attack described in this paper is a novel attack scenario.

The rest of this paper is organised as follows. The Advanced Encryption Standard is described in II, and Correlation Power Analysis is described in III. We describe our implementation of the AES algorithm and the chip that was designated the chip under attack in Section IV. The communication protocol and how this would fit in with the normal use of a CAN bus is described in Section V. The attack itself is described in Section VI. This is followed by our conclusions in Section VII.

Notation: The base of a value is determined by a trailing subscript, which is applied to the whole word preceding the subscript. For example, $\texttt{FE}_{16}$ is 254 expressed in base 16, $X = (x_{\ell-1}, x_{\ell-2}, \ldots, x_0)_{256}$ gives an expression for the individual bytes of X.

II Advanced Encryption Standard

The Advanced Encryption Standard (AES) [8] was introduced in 2001 as a replacement to DES [7]. This was because it became apparent that the key used in DES is too short, as an exhaustive search had become feasible with sufficient computing power. Various different sizes of keys are possible. For simplicity of presentation, only the simplest case of a 128-bit key will be considered.

Algorithm 1: Advanced Encryption Standard

Input: M, K
Output: C

$X \leftarrow M \oplus K$;
for $i \leftarrow 1$ **to** 10 **do**
  $X \leftarrow$ **ShiftRow**(X) ;
  $X \leftarrow$ **ByteSub**(X) ;
  if $i \neq 10$ **then**
    $X \leftarrow$ **MixColumn**(X) ;
  end
  $K \leftarrow$ **KeySchedule**(K) ;
  $X \leftarrow X \oplus K$;
end
$C \leftarrow X$
return C

The structure of this algorithm is given in Algorithm 1, where a message block (M) is enciphered using a key (K) to produce a ciphertext block (C). The **ShiftRow** function is a bytewise permutation of the input data. This is followed by the **ByteSub** function that is a substitution table applied to each byte of the input data. This table is an inversion over GF(2^8) followed by a bitwise permutation. The XOR with the subkey is also referred to as the **AddRoundKey** function. The **MixColumn** function is given in Algorithm 2, where $\bullet$ represents polynomial multiplication over GF(2^8) modulo the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$.

Algorithm 2: The **MixColumn** Function

Input: $X = (x_0, x_1, \ldots, x_{15})_{256}$
Output: $Y = (y_0, y_1, \ldots, y_{15})_{256}$

for $i \leftarrow 0$ **to** 15 **do**
  $y_i = 2 \bullet x_i \oplus 3 \bullet x_{(i+4) \bmod 16} \oplus x_{(i+8) \bmod 16} \oplus x_{(i+12) \bmod 16}$
end
return Y

The **KeySchedule** is a function that generates a subkey from the previous subkey. The first subkey is the key with no changes, and subsequent subkeys are generated using this function. This function is described in Algorithm 3, where z is a constant that varies from one round to another.

Algorithm 3: The AES Key Schedule

Input: $X = (x_0, x_1, \ldots, x_{15})_{256}$, z
Output: $X = (x_0, x_1, \ldots, x_{15})_{256}$

for $i \leftarrow 0$ **to** 3 **do**
  $x_{i<<2} \leftarrow x_{i<<2} \oplus$
  ByteSub$(x_{(((i+1)\wedge 3)<<2)+3})$;
end
$x_0 \leftarrow x_0 \oplus z$;
for $i \leftarrow 0$ **to** 15 **do**
  if $i \neq 0 \bmod 4$ **then**
    $x_i \leftarrow x_i \oplus x_{i+1}$
  end
end
return X

A permutation is sometimes also used on the message and key on entry to the algorithm, to convert the array format to the grid format used in the specification [8]. This is an optional bytewise permutation that changes the way bytes are addressed within an implementation.

The deciphering function will not be detailed precisely here, as it is simply the inverse of Algorithm 1, further details are available in [8].

III Correlation Power Analysis

The idea of statistically treating power analysis traces was first presented to the cryptographic community in [5], and was later improved upon to in [2] to give Correlation Power Analysis (CPA). CPA is based on the relationship between the power consumption and the Hamming weight of data being manipulated at a given point in time. The differences in power consumption are potentially extremely small, and cannot be interpreted

individually, as the information will be lost in the noise incurred during the acquisition process.

Correlation Power Analysis (CPA) can be performed on any algorithm in which an intermediate operation of the form $\beta = S(\alpha \oplus K)$ is calculated, where α is known and K is the key (or some segment of the key). The function S is typically a non-linear function, usually a substitution table (referred to as an S-box), which produces an intermediate output value β.

The process of performing the attack initially involves running a microprocessor N times with N distinct message values M_i, where $1 \leq i \leq N$. The encryption of the message M_i under the key K to produce the corresponding ciphertext C_i will result in power consumption traces w_i, for $1 \leq i \leq N$, that can be captured with an oscilloscope and sent to a computer for analysis and processing.

If an attacker can predict the Hamming weight of the value of β for each of the acquired power consumption traces h_i, for $1 \leq i \leq N$, they can compute a correlation trace by calculating the correlation between h_i and the points w_i to produce a correlation trace. This gives the following calculation:

$$\rho = \frac{N \sum_{i=1}^{N} w_i h_i - \sum_{i=1}^{N} w_i \sum_{i=1}^{N} h_i}{\sqrt{N \sum_{i=1}^{N} w_i^2 - (\sum_{i=1}^{N} w_i)^2} \sqrt{N \sum_{i=1}^{N} h_i^2 - (\sum_{i=1}^{N} h_i)^2}}$$

where all operations on waveforms are conducted in a pointwise fashion, i.e. this calculation is conducted on the first point of each acquisition to produce the first point of the differential trace, the second point of each acquisition to produce the second point of the differential trace, etc. This is Pearson's correlation coefficient where ρ is computed by dividing the covariance of h_i and w_i by the product of the standard deviation of h_i and w_i.

In order to derive K, a correlation trace is produced for each value that K can take. In AES the first subkey will be treated in groups of eight bits, so 256 (i.e. 2^8) correlation traces need to be generated to test all the combinations of eight bits. The correlation trace with the largest peak will validate a hypothesis for K.

IV Implementation

The AES algorithm was implemented on a Microchip PICDEM CAN-LIN 2 development board which supports both LIN and CAN bus development. The on-board CAN bus consisted of two CAN nodes, as shown in Figure 1. One of these nodes (Node 0) has the capability to communicate with the PC through the serial port. The second node (Node 1) communicates with Node 0 over the CAN bus.

The scenario we are considering is a chip with an unknown secret key (Node 1) has been removed from a system and is being interrogated by a second chip over the CAN bus controlled by an attacker (Node 0). To ensure that the setup of the board was as close to a real life implementation as possible, Node 1 was used to represent an in-car chip and Node 0 was used to interrogate Node 1 over the CAN bus. As all nodes in the system would be using the same Key value, only one node need be analysed to obtain the secret key. In our setup Node 0 would encipher messages that would be deciphered by Node 1.

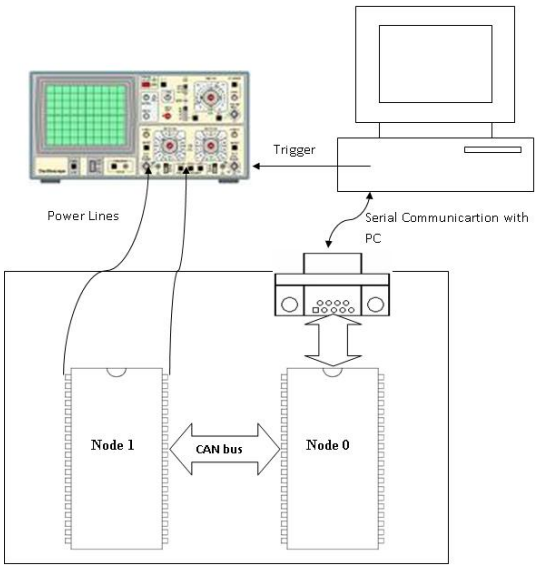

Fig. 1: The experimental setup.

V Communication Protocol

The CAN bus used in was set up so that it could handle encrypted messages, but not affect how standard CAN messages are received. The CAN bus was designed to be fast and very reliable, so decrypting every message that is sent on the CAN bus would not necessarily be the best solution. Only CAN messages that hold important information would need to be encrypted. This way the speed of the CAN bus can remain relatively unchanged. It was assumed that messages that require encryption would only be sent over the CAN bus a small percentage of the time. For this reason, these messages should have a low priority in comparison with messages that are sent frequently over the CAN bus. The CAN bus is used to transmit information that controls important devices in the car such as airbags. Delaying the speed at which such messages could be sent on the bus would not have been an acceptable result. Encrypted messages were also given specific identifiers.

The CAN (Controller Area Network) bus is an asynchronous serial communications protocol. It utilises a balanced 2-wire differential signalling in-

terface. CAN supports bit rates of up to 1 Mbit/s and uses NRZ (Non Return to Zero) encoding, with bit stuffing. CAN is a broadcast style system made up of multiple nodes where all nodes are capable of receiving and transmitting messages. Control of the bus is decided using bitwise arbitration of the identifier field of a CAN message. When a period of inactivity is detected on the bus a node may begin to send a message. If two nodes send a message at the same time then the message with the most dominant bits in the identifier field will gain control of the bus.

A standard CAN message consists of three fields which contain data. The identifier field is an 11 bit field and indicates the priority of a message and also which node/nodes are to receive the message. The message field consists of between 1 and 8 bytes of data. The third field is the data length code (DLC) this field indicates the number of bytes contained in the message field. Other fields are present but they are used for synchronisation and error detection purposes. Extended CAN protocol supports longer ID's but the standard CAN protocol is used in this paper. During the transmission of a message, every node monitors the bus and checks the ID of the message against a predefined filter. Any node whose filter indicates that the message is not to be received, disregards the remainder of the message. Any encrypted data is included in the 8 byte message field of a CAN message. Therefore two CAN messages are required to send an AES message or ciphertext. Each of these two encrypted messages was given its own identifier, in that way the order in which to recombine the messages would be known by the receiving node and would not be dependent on the order in which the messages were received.

The parts of a CAN message that require encryption are the ID, message and the DLC. The input required for the AES algorithm is 16 bytes. The maximum number of bytes that require encryption in a standard CAN message is 10, therefore the input is padded with 0's to make up 16-bytes. The input to the AES should also include a byte in position 15 which gives the length of the CAN message. Without this the receiving node would not be able to tell how much of the 16-byte output of the AES is actually a CAN message.

The ID, DLC and message fields of the message are encrypted by the AES algorithm in that order. The 16-byte output of the AES is split into two 8-byte parts. Bytes 1–8 are placed in a CAN message with ID 1 and bytes 9–16 are placed in a CAN message with ID 2. Both messages are sent when a slot is open on the CAN bus. Only nodes that are programmed to receive encrypted CAN messages will receive them.

A node receiving a message must first pass it through a filter as it does with all CAN messages. If the message is not stopped by the filter and the ID matches that of an encrypted message then, the node places the 8 byte message field in a register. At some point the node should receive the second encrypted message with the other ID. The node places the 8 byte message in the other half of the register and begins the decryption process. Only on receiving both messages does the node attempt to decrypt the message. Once the decryption process is finished, the node should have access to the ID, DLC and the 8 byte message that had been encrypted.

VI Attacking a Node

As the decryption of a message occurs before a receiving node checks the message ID against its filter, all one must know is the ID of encrypted messages. The attack was set up so that a node controlled by a PC sent 1000 encrypted messages to a node being interrogated. The power lines of this node were monitored as it was decrypting the messages. As the attacker has knowledge of the plaintext being decrypted information on the secret key being used may derived.

Correlation Power Analysis is based on the linear relationship between the Hamming weight of data being manipulated and the instantaneous power consumption. The attacker must know the data that is being decrypted and also have access to the power lines of the chip performing the decryption. For our approach, the power lines of node 1 were monitored and node 0 was used to send 1000 messages to over the CAN bus to node 1. The power trace of node 1 is recorded for the period in which it is decoding the message (see Figure 1).

This was achieved by the PC sending the data to Node 0, setting a trigger on the oscilloscope. The oscilloscope then acquires the power consumption trace of Node 1 while it is decrypting the data. The power traces can then be analysed with their corresponding data. An adversary may identify the time at which the encryption is taking place by monitoring when a message is sent across the CAN bus. As soon as Node 1 receives both encrypted CAN messages it will begin the decryption process. To speed up the process of synchronising the traces, an output pin on the PIC was used to show when the encryption process was occurring. This would not be possible in practice for an attacker but using synchronisation software would give the same result, this would, however, takes more time to successfully attack a chip.

The microprocessor that was used in our experiments was a Microchip PIC18F458. This is a low-power microprocessor and therefore the power analysis is not as straight forward as it would be for

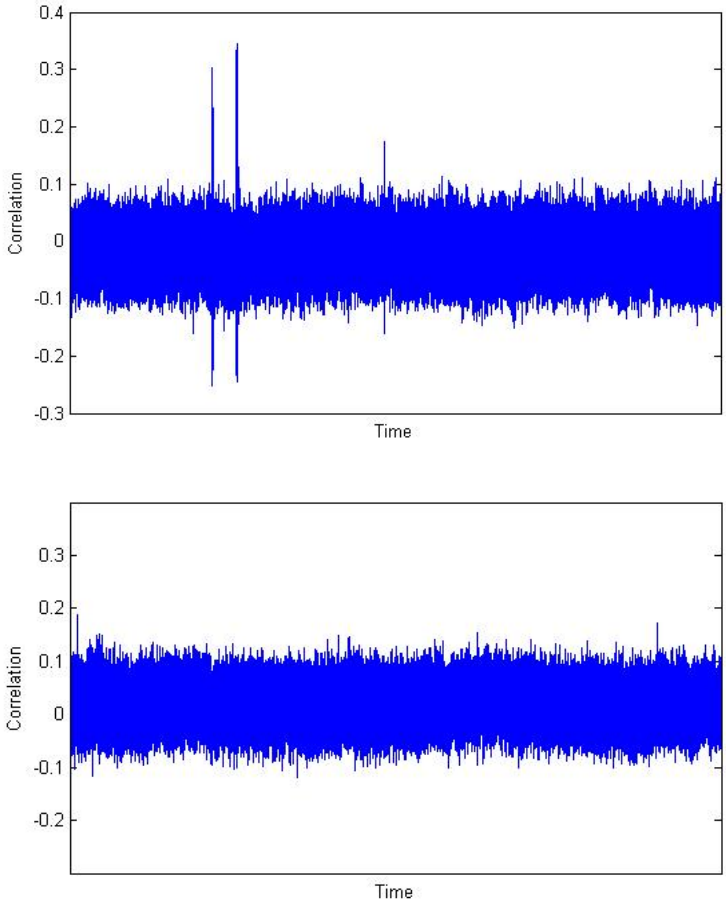

Fig. 2: The correlation trace for the correct key guess (upper) and the next most significant incorrect hypothesis (lower).

a normal microprocessor. The microprocessor was clocked by the development board at 25 MHz. It has been demonstrated that lowering clock speeds can improve the quality of the results of a CPA [6]. However, this was not changed to keep the attack as close as possible to an attack scenario, especially as countermeasures are being proposed to inhibit the modification of the clock speed [9]. Each of the transitions introduced a peak in the power consumption followed by a damped oscillation (as shown in Figure 3), and the magnitude of this peak varied considerably from one acquisition to another. Before these traces were used to derive any information they were, therefore, filtered with a low-pass filter with a cut off threshold set to 250 MHz. This removed these oscillations and any other high frequency noise that was acquired.

The 1000 traces were than treated by calculating the correlation between the points of the acquisitions and a predicted byte within the algorithm, as described in Section III. The point that we chosen was the output of the first ByteSub operation when the ciphertext sent to Node 1 was being deciphered. 256 correlation traces were generated, i.e.

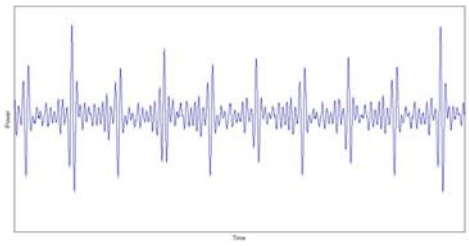

Fig. 3: An example power trace.

one for each of the possible hypotheses for the first byte of the last subkey. The two traces with the largest correlations are plotted in Figure 2. The largest correlation is present in the upper trace, which corresponds to the correlation for the correct key guess. The lower trace corresponds to the correlation trace for an incorrect guess with the largest correlation. It can be seen that the correlation traces are adequate to determine one key byte. In order to recover an entire secret key an attacker would need to repeat this process for each

of the sixteen ByteSub operations.

There are two different power supply pins on the Microchip PIC18F458, and the same information could be seen by acquiring the power consumption of either pin.

VII Conclusion

This paper outlines a strategy for applying side channel analysis to an automotive system. The system was designed to represent an in-car chip capable of securing data being sent across a CAN bus. The attack analyses the power consumption in an attempt to acquire the value of the secret key used by a microprocessor. We have demonstrated that that a secret key can be derived with 1000 random messages sent across a CAN bus for the microprocessor to decrypt, because of the linear relationship between the data being processed and the instantaneous power consumption. The attack that is described in this paper targeted the first ByteSub function of the decryption algorithm. We have shown that CPA can successfully be applied to an automotive system. We have also shown that a CPA attack can be used against a low power microprocessor if appropriate filtering and synchronisation techniques are used. Correlation values are less then would be seen for a normal microprocessor because the chip used was a low power microprocessor. However, the correlation result for the correct key guess was distinctly larger then the next highest guess.

The countermeasures required to prevent Correlation Power Analysis are well defined and are typically included in smart card implementations of cryptographic algorithms. However, these countermeasures are not typically included in automotive microprocessors since side channel analysis is not considered a threat. The most common countermeasure is Boolean masking is where the data is manipulated in such a way that the value present in memory is always masked (XORed) with a random value that will change each time an algorithm is computed, which removes any correlation between the Hamming weight and the instantaneous power consumption. Some ideas for implementing this countermeasure were proposed in [3], and an example of this sort of implementation applied to block ciphers can be found in [1].

References

[1] M.-L. Akkar and C. Giraud. An implementation of DES and AES secure against some attacks. In C. K. Koç, D. Naccache, and C. Paar, editors, *Cryptogaphic Hardware and Embedded Systems — CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer-Verlag, 2001.

[2] E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems — CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer-Verlag, 2004.

[3] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards approaches to counteract power-analysis attacks. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer-Verlag, 1999.

[4] Robert Bosch GmbH. Bosch CAN specification version 2.0. `http://www.semiconductors.bosch.de/pdf/can2spec.pdf`, 1991.

[5] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

[6] S. Mangard, N. Pramstaller, and E. Oswald. Successfully attacking masked AES hardware implementations. In J. R. Rao and B. Sunar, editors, *Cryptogaphic Hardware and Embedded Systems — CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer-Verlag, 2005.

[7] NIST. *Data Encryption Standard (DES) (FIPS–46-3)*. National Institute of Standards and Technology, 1999.

[8] NIST. *Advanced Encryption Standard (AES) (FIPS–197)*. National Institute of Standards and Technology, 2001.

[9] F. Vater and S. Peter amd P. Langendörfer. Combinatorial logic circuitry as means to protect low cost devices against side channel attacks. In D. Sauveron, K. Markantonakis, A. Bilas, and J.-J. Quisquater, editors, *Information Security Theory and Practices 2007 — Smart Cards, Mobile and Ubiquitous Computing Systems — WISTP 2007*, volume 4462 of *Lecture Notes in Computer Science*, pages 244–253. Springer-Verlag, 2007.