Proceedings of the International Multiconference on Computer Science and Information Technology pp. 527–533
ISSN 1896-7094

Implementation of Behaviour-Based Temporally Aware Security in Smart Cards

William G. Sirett, Konstantinos Markantonakis and Keith Mayes

Smart Card Centre**
Royal Holloway, University of London, TW20 0EX, UK
{w.g.sirett, k.markantonakis, keith.mayes}@rhul.ac.uk

Abstract. Behaviour-based security is a group of techniques used to monitor the activity of a system to identify abnormal behaviour and possibly an attack in progress. Smart cards present a constrained environment for behaviour-based security as there is no on-card source of time. A smart card applicable timestamping scheme, to provide secure time, is identified and used in a Java Card behaviour-based temporally aware security countermeasure; the functionality, implementation and operation of which is fully detailed in this work.

1 Introduction

This work identifies that the lack of an internal time source in smart cards can limit security levels. Several attacks exploit the smart card's inability to assess its own behaviour over time and restrict usage; for example: Differential Power Analysis [1, 6, 8] and Card Sharing attack [7][1]. Behaviour-based security is a group of techniques used to monitor the activity of a system to identify abnormal behaviour and possibly an attack in progress [4]. Using counters and state transition, a smart can limit unexpected sequences of behaviour but not the rate at which that behaviour occurs — the ability to assess behaviour with respect to time requires a trusted time source.

Having no internal source of time [9] is a basic limitation of smart card technology. Only three possible sources are available: an internal clock, a remote server or a neighbouring device [10]. To have an internal time keeper would require, at least, a portion of the card's chip to have a source of permanent power [3]. This work uses a timestamping scheme [11] with an on-card behaviour-based security countermeasure which is detailed in this paper. The countermeasure allows the smart card to enforce *contracts* of behaviour upon third party applications.

2 Assumptions and Requirements

(A1) Secure timestamping protocol in use,
(A2) Timestamp is represented in ISO 8601 format.
(A3) Cards have resources required to handle countermeasures,

A timestamping protocol is detailed fully in [11] and provides an ISO 8601 [5] formatted timestamp in a secure manner that is feasible to smart card operation. It is assumed that the smart card has the required resources to handle the solution: storage, computation, communication, etc.

(R1) Software based solution,
(R2) On-card solution,
(R3) Ability to track last known time,
(R4) Ability to measure frequency of events,
(R5) Ability to set up benchmarks for individual events,
(R6) Support for multiple applications,
(R7) Ability to advise third party applications to withhold services.

** This work was supported by sponsorship funding from the Smart Card Centre, founded by Royal Holloway University of London, Vodafone and G&D in 2002.

[1] Further details about these attacks and their behavioural characteristics can be found in [11].

A solution should not require modification of existing hardware to be applicable to existing smart cards and therefore be software based (R1). A smart card may be tamper resistant, not tamper proof [2], but it is more trustworthy than the device. The device can be modified and is often used as a tool of attack as shown by the card sharing attack and requires an on-card solution (R2). The remaining requirements (R3-7) address the functionality that is necessary to solve the problem. The countermeasure needs to be able to track time and to measure the frequency of events. Multiple applications need to be able to register contracts for their own behaviour. Finally, the countermeasure should be able to instruct an application to withhold its services if violating activity is detected.

3 The Proposed Architecture

This section presents an overview of the countermeasure architecture and process, details of functionality, implementation information and fulfilment of requirement analysis.

3.1 Countermeasure Architecture

Figure 1 shows the proposed architecture of the countermeasure. As the solution is an on-card solution (R1) anything outside of the smart card is not included. The usage scenario looks at addressing problems in existing infrastructures; therefore consider that the Card Acceptance Device (CAD) could be any fielded device such as a mobile device or Set-Top-Box (STB).

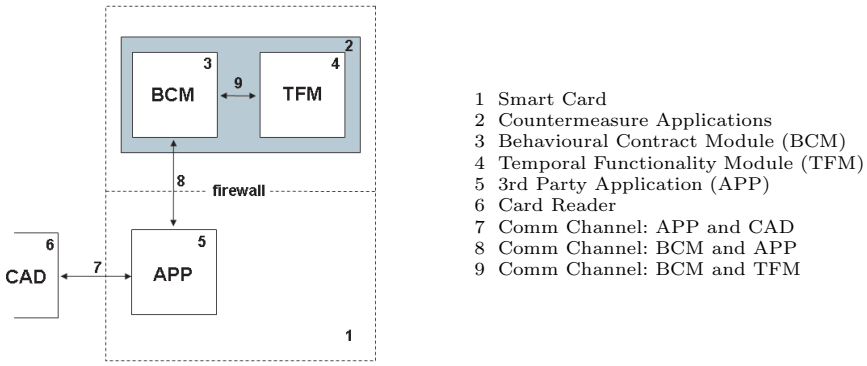

Fig. 1. Architecture of Contractual Behaviour Countermeasure

3.2 Overview of Process

The smart card application (APP) represents the application that is being monitored by the countermeasure. The countermeasure is proposed to work in the following manner (with reference to figure 1 and flow of process in figure 2):

(P1) An external communication from the card reader, via channel (7), requests a service from the APP and this request is accompanied with a timestamp and an instruction identifier.
(P2) The APP communicates this instruction request and timestamp to the Behavioural Contract Module (BCM) via channel (8).
(P3) The BCM uses the Time Functionality Module (TFM) to validate the timestamp.
(P4) The BCM, using the TFM, determines that the current timestamp is after last known time.
(P5) The BCM then checks whether this is the first request in reference to a specific instruction. If it is the first request the contract is updated and the request is permitted. If it is not the first occurrence then the contract is checked for breach of behaviour. If the contract is violated then the BCM will advise the APP to withhold services for an invalid instruction request.

Any time based functionality that is required by the BCM is provided by the TFM. The remainder of this section details the functionality of the TFM and BCM that facilitates the preceding process. Throughout this discussion the notation in table 1 is used.

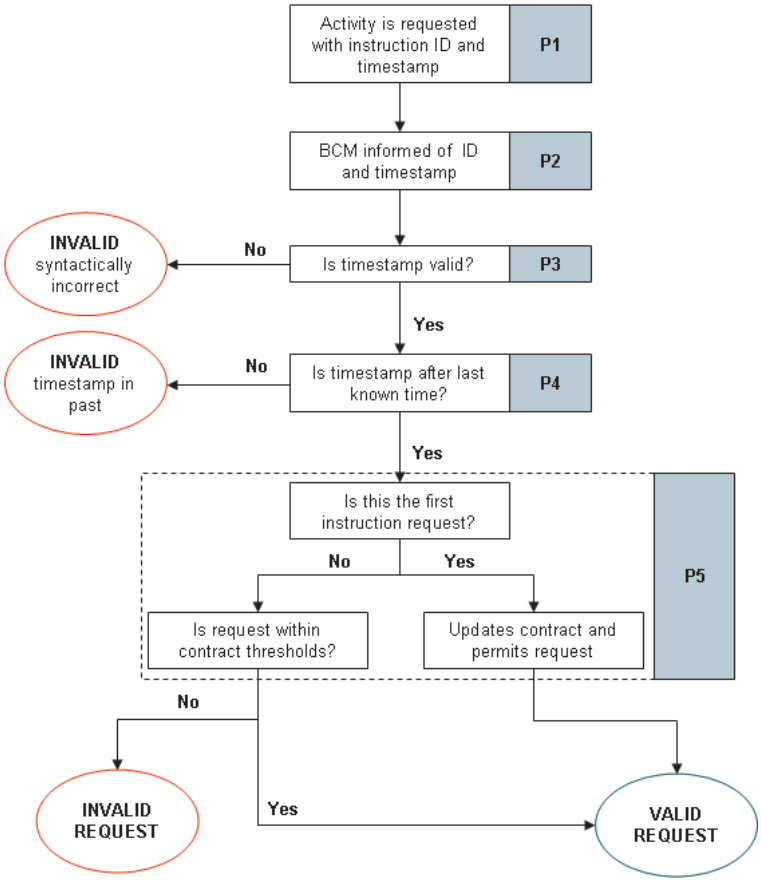

Fig. 2. Overview of Countermeasure Process

Table 1. Notation for Countermeasure Functionality

TS_t	= timestamp at time t
I_x	= interval of x
n_x	= counter of x
C	= contract
t	= current known time
0	= first known time
z	= time passed since previous known time

3.3 Temporal Functionality Module Functionality

The TFM provides temporal based functionality for the BCM. Five key functions are required to provide the necessary functionality (with reference to the notation provided in table 1).

Validate Time: The TFM iterates through the byte array containing the timestamp to establish whether or not the timestamp is syntactically correct.

Check Last Known Time: TFM receives the latest timestamp (TS_t) from the BCM and compares it to the last known time (TS_{t-z}). If the new time is after of the last known time the last known time is updated. If not then it rejects the new stamp and returns a negative result (1).

$$\text{IF } TS_t > TS_{t-z} \text{ then VALID} \tag{1}$$

Calculate Interval: This function calculates the interval between two provided timestamps (2).

$$I = TS_t - TS_{t-z} \tag{2}$$

Compare Intervals: The TFM is provided with two intervals and returns a positive response if one timestamp is greater than another. This is used when the contract interval (I_C) is compared to the current interval (I) per event (3). The contract is violated if the current interval per event is less than the benchmark.

$$\text{IF } I > I_C \text{ then VALID} \tag{3}$$

Calculate Interval per Event: The BCM provides the TFM with an interval definition (I) and the count of events recorded (n_C) in the related contract record (C) and this function returns the calculated interval per event (I_{pe}) (4).

$$I_{pe} = I/n_C \tag{4}$$

3.4 Behavioural Contract Module Functionality

The BCM is the managing element of this countermeasure and provides three functions:

Behavioural Contract Registration: For the BCM to create a contract it requires an instruction identifier and an activity benchmark which defines the minimum time interval between instruction requests. The BCM then stores this contract interval (5) with the instruction identifier and application identifier. This is necessary as the countermeasure must be able to handle contracts for multiple applications. The application identifier can be retrieved from the runtime environment when the request comes from the APP to distinguish between client applications.

$$I_C = I \tag{5}$$

Each contract holds only six pieces of information and maintains a constant size in memory as shown in table 2. Each contract stores a 16 byte application identifier and a single byte to refer to the specific instruction. This indicates that each application can have a maximum of 0xFF or 255 instruction specific contracts. This can be increased to accommodate more instructions. An event count is stored in 4 bytes and can hold a large running count and if the particular smart card supports integer data types then 4 bytes is required to store an `int`. The last three items are two 7 byte timestamps[2] and a 7 byte interval definition and the total contract size is 42 bytes.

Table 2. Structure of Behaviour Contract

Application Identifier = 16 bytes
Instruction Identifier = 1 byte
Event Count = 4 bytes
1st Timestamp = 7 bytes
Last Timestamp = 7 bytes
Benchmark Interval = 7 bytes
Total = 42 bytes

Check Current Time: The BCM verifies that the new timestamp (TS_t) received is after the last known timestamp (TS_{t-z}) (6).

$$\text{IF } TS_t > TS_{t-z} \text{ then VALID} \tag{6}$$

Check Contract Benchmark: The role of the BCM is to validate event requests using stored contracts and this capability is provided to the APP by this function; the following process description is also given as a flow chart in figure 2.

[2] 7 byte format specified by the reduced ISO 8601 format.

The APP provides an instruction identifier and a current timestamp (TS_t) to the BCM. The BCM checks for an existing contract, first retrieving the application identifier from the runtime environment and updating given elements of information. It updates the last recorded event timestamp and, if it is the first occurrence, updates the first recorded event timestamp (TS_0) (7). Additionally it also increments the contract event counter (n_C) (8). The interval per event (I_{pe}) is calculated using the *calculate interval per event* function of the TFM (3.3). The BCM then validates the instruction request by checking the resulting interval against the contracted interval (I_C) using the TFM *compare interval* function (9).
If the threshold is greater than the current calculated interval then the instruction request is in breach of contract and the APP is advised to refuse the request. Else the request is valid and the APP should perform the requested activity.

$$\text{IF } n_C = 0 \text{ THEN } TS_0 := TS_t \tag{7}$$

$$n_C := n_C + 1 \tag{8}$$

$$\text{IF } I_{pe} > I_C \text{ then VALID} \tag{9}$$

4 Implementation Details

To validate our proposal a proof of concept model was constructed based on readily available open source tools to run a simulated smart card environment. The chosen development language was Java and the Java Card platform [13]. The model comprises three elements: APP, BCM and TFM. The APP communicates with the BCM which in turn uses the TFM. The BCM class is part of a that instantiates a class called TFM which provides the time functionality. The countermeasure package also contains an interface class and a contract class. The interface class allows other Java Card Applets to use the BCM's services by providing details of the BCM's public API and the contract object is an underlying class that is used by the BCM.

5 Requirement Fulfilment

The proposed countermeasure is software based and affects only the smart card, fulfilling (R1) and (R2). The requirements defined various functional features required of the solution; (R3) stated that the last known time was to be tracked and used during operation which is governed by the TFM and employed by the BCM. The BCM allows multiple applications to register for contracts on multiple events and advise the withdrawal of application services. This fulfils (R5), (R6) and (R7). Finally, (R4) requires that the solution can measure the frequency of events on-card. The TFM provides the ability to perform this action and the BCM puts that information into context.

6 Operation

Table 3 shows the contents of a behaviour contract and how the values changed during operation.

Events 4 and 8 show that when the current interval was shown to be in violation of the contract the card's services were withheld. These results show that if card usage breaches contracted behaviour then the services of the smart card should be denied. These breaches are expected in each of the example attacks briefly mentioned in the introduction.

Figure 3 shows the operation information as a chart. The dashed line represents the contract benchmark interval which remains constant, and the solid line shows the varying current interval as calculated by the BCM in relation to specific requests for activity. When the solid line breaks the dashed line the contract is violated and the BCM instructs the APP to withhold services. The exception is during the first request for service; at this point the calculated current interval is below that of the contract interval but is ignored as the BCM initialises the contract and last known time.

7 Conclusions

This work identifies that, while some limited behaviour-based security is possible on current smart cards, time based analysis of behaviour is not possible without a trusted source of time; either internal or

Table 3. Contract State During Operation — Decimal

n_C	TS_0	TS_t	I_C	I_{pe}	Result
1	2005-07-06 15:51:30	2005-07-06 15:51:30	16 s	0 s	valid
2	2005-07-06 15:51:30	2005-07-06 15:52:30	16 s	30 s	valid
3	2005-07-06 15:51:30	2005-07-06 15:52:31	16 s	20 s	valid
4	**2005-07-06 15:51:30**	**2005-07-06 15:52:32**	**16 s**	**15 s**	**invalid**
5	2005-07-06 15:51:30	2005-07-06 15:53:32	16 s	24 s	valid
6	2005-07-06 15:51:30	2005-07-06 15:53:33	16 s	20 s	valid
7	2005-07-06 15:51:30	2005-07-06 15:53:34	16 s	17 s	valid
8	**2005-07-06 15:51:30**	**2005-07-06 15:53:35**	**16 s**	**15 s**	**invalid**

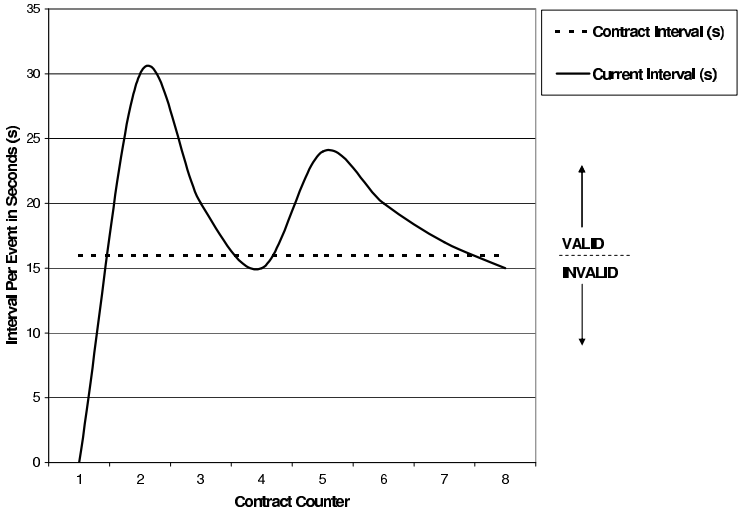

Fig. 3. Contract State During Operation

external to the smart card. This work defined the assumptions and requirements of an on-card behaviour-based temporally aware security countermeasure which allows a smart card to monitor its behaviour rate with respect to time. Using timestamps in this manner is a simple but innovative solution and could provide smart card with a greater capacity for self defence in the future. The functionality of the proposed solution was fully presented and a practical implementation was realised. Full analysis is beyond the scope of this work and published elsewhere [12].

References

1. M. Aigner and E. Oswald: *Power analysis tutorial,* Institute for Applied Information Processing and Communication, University of Technology Graz — Seminar, 2000.
2. R. Anderson and M. Kuhn. *Tamper resistance — a cautionary note,* In: The 2nd USENIX Workshop on Electronic Commerce Proceedings, Oakland, California, pages 1–11. USENIX Association, November 1996.
3. V. Cordonnier, A. Watson, and S. Nemchenko. *Time as an aid to improving security in smart cards,* In: 7th Annual Working Conference on Information Security Management and Small Systems Security, pages 131–144. Kluwer Academic Press, London, 1999. Amsterdam, The Netherlands.
4. D. Geer. *Behavior-based network security goes mainstream,* Computer, **39**(3):14–17, March 2006. IEEE.
5. ISO. *ISO 8601 Data elements and interchange formats – Information interchange – Representation of dates and times.* `http://www.iso.org` 3rd edition, 2004.
6. P. Kocher, J. Jaffe, and B. Jun. *Differential power analysis,* In: M. Wiener, editor, Advances in Cryptology CRYPTO '99, volume **1666** of Lecture Notes in Computer Science, pages 388–397. Springer-Verlag, 1999.

7. M. Kuhn. *Attack on pay-TV access control systems, December 1997,* Security seminar talk, University of Cambridge, Computer Laboratory, London, UK. `http://www.cl.cam.ac.uk/~mgk25`
8. T. S. Messerges, E. A. Dabbish, and R. H. Sloan. *Investigations of power analysis attacks on smartcards,* In: 1st USENIX Workshop on Smartcard Technology (Smartcard '99), pages 151–162. USENIX, May 1999. Chicago, Illinois, USA.
9. W. Rankl and W. Effing. *Smart Card Handbook*. John Wiley & Sons, Ltd, 3rd edition, 2003. ISBN: 0470856688.
10. L. Rousseau. *Secure time in a portable device,* Proceedings of 3rd Gemplus Developer Conference, Paris, France, 2001. Gemplus.
11. W. G. Sirett. *Analysis, Implementation, and Deployment of Behaviour-Based Temporally Aware Security in Smart Cards*. Phd thesis, Department of Mathematics, Royal Holloway, University of London, September 2006.
12. W. G. Sirett, K. Markantonakis, and K. Mayes. *Temporally aware behaviour-based security in smart cards,* In: 2006 International Conference on Computational Intelligence and Security (CIS '06), IEEE Xplore. IEEE, Nov 2006. Guangzhou, China.
13. Sun Miscrosystems Inc. *Runtime Environment Specification; Java Card Platform, Version 2.2.1* `http://java.sun.com` 2003.