# Impact of Security Policies on the TMN X Interface Integrity and Performance

**Ognjen Prnjat, Lionel Sacks**
University College London
Torrington Place
London WC1E 7JE, England
phone: +44 171 419 3198
email: {oprnjat | lsacks}@ee.ucl.ac.uk

**Abstract**

*This paper discusses the interconnection requirements between management systems over the TMN X interface in terms of security and integrity, as well as the relationship between these two issues. A testing approach is developed in order to assess the impact of introducing security policies in the inter-domain communication between two independent TMN OSs, on the basic functional and performance-related integrity requirements. The results and the assessment of the impact of the security policies on the X interface performance are given. The work presented here was conducted for the ACTS project TRUMPET (DGXIIIB, AC112).*

## 1. Introduction

The current telecommunications environment is characterised by the increasing level of interconnection between network operators and third party retailers in the telecommunications market. These players are encouraged to interwork so as to provide increasingly sophisticated public voice and data services to end-users and corporate customers. This inter-working is taking place not just on the level of bearer services, but also on the level of management systems. The pre-requisite for inter-working of management systems is the assurance that operation of one system will not jeopardise the correct and proper functioning and performance of another system (referred to as integrity), and that the inter-working will be secure in terms of authorised access and ability of the inter-working parties to avoid malicious human intervention.

An example inter-working scenario is given by the ACTS project TRUMPET. This project is concerned with the development of the TMN-based inter-domain management systems within the Open Network Provisioning framework, and the development of security policies for the inter-domain interaction. The TRUMPET scenario (Figure 1) involves several administratively separate players: two (or more) Public Network Operators (PNOs), a Value Added Service Provider (VASP), and a number of customers at various sites - Customer Premises Networks (CPNs) [i].

The management systems of these players form a service provisioning system for management and provision of broadband (ATM) network connections between two customers/end users. Each of the players has an independent management system under its control. CPN is an actor that has a contract agreement with the VASP regarding the use of the service by one or more authorised end-users. The VASP management system provides network connectivity to customers, on contractual basis, by utilising the resources of one or more PNOs. VASP is responsible for the service offered, and allows customers to create, modify and delete connections, thus effectively providing the “semi-permanent virtual connection” service to the customers. PNOs provide the VASP with the physical infrastructure and connectivity capabilities, by operating basic switching and transmission capabilities. The core TMN manager-agent chain [ii] consists of Xuser manager (VASP domain)---

Xuser agent (PNO)---M4 G/W (PNO)---M4 agent (PNO). The main set of security policies is applied between the Xuser manager and agent, which reside in autonomous TMN domains.

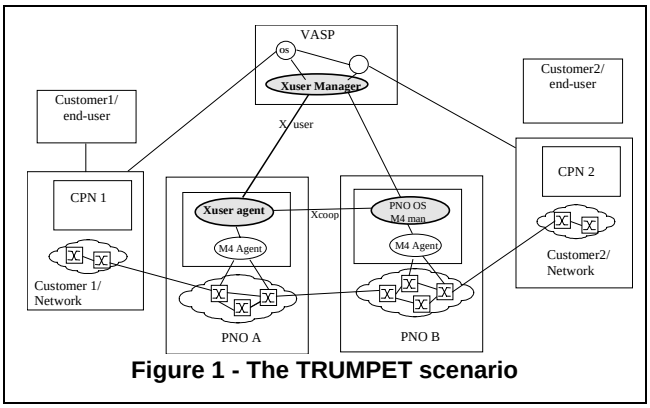

**Figure 1 - The TRUMPET scenario**

The critical point of the management systems' interconnection is the TMN X interface, such as the Xuser between the VASP and the PNO and the Xcoop between the two PNO OSs. There are two basic requirements on these interfaces: **integrity**, referring to the correct and proper operation of the interconnected management systems and the supporting communications mechanism, in terms of *functionality* and *performance* [iii]; and **security**, referring to the secure and authenticated management interactions between autonomous and possibly mutually untrusting organisations.

## 2. Interconnection requirements

Firstly, the correct and high-integrity operation is essential between the OSs over the X interface: the management systems' and communications infrastructure has to retain its correct attributes in terms of functionality and performance. The *functional* integrity attributes are the following:

**Sequencing**. Proper sequencing of actions has to be preserved over the management communications mechanism.

**Liveness**. The system needs to be live: situations such as deadlocks (system being blocked awaiting a message that cannot be emitted) and livelocks (system oscillating between a certain number of states that it cannot leave) have to be avoided, so as to maintain availability.

**Robustness**. System must be able to handle all possible states of its environment: unexpected messages, duplicate messages, *etc*.: it has to be robust and stay operational in these circumstances.

**Data Integrity**. The data exchange between two OSs has to be correct. This means that data should not be corrupted or lost by the communications stack and software, and conversely, that the performance of the communications mechanism should not be influenced by the data content. In that sense the data integrity requirement is an integrity performance requirement as well.

The *performance*-related integrity requirements, besides the above data integrity requirement, are:

**Timing / Time sensitive performance**. Timing of the operations has to stay within well-defined limits - the communications mechanism between two OSs should not jeopardise the correct timing. Increased response time may cause processes to slow down, collapse or just decrease availability, which could in turn impact the Quality of Service (QoS) offered to the customers.

**Throughput/Rate**. The communications mechanism has to support a certain throughput, and the rate of signal exchange should not in any way impact the operation of applications.

If the system either looses its liveness, or if sequencing is distorted, or if the system starts performing outside its time limits, or if it cannot deal with unexpected messages, or if data exchange is corrupted, the system performance is degrading, and the functionality might be lost. In other words, the integrity of that system is at stake.

Second interconnection requirement on the X interface is security: the management data exchange between two parties must be secure from the intentional attack, and authenticated for correct identity of entities communicating over the X interface. Basic security requirements are as follows:

**Authentication** refers to the mutual recognition of the communicating parties. Both parties must authenticate themselves to each other.
**Access control** to managed objects insures that the managing party can access only a certain set of objects on the agent party side, according to the contract established.
**Data integrity**: data must be protected against modification, insertion and repetition.
**Confidentiality**: data content must not be disclosed, while in transit, to unauthorised parties.

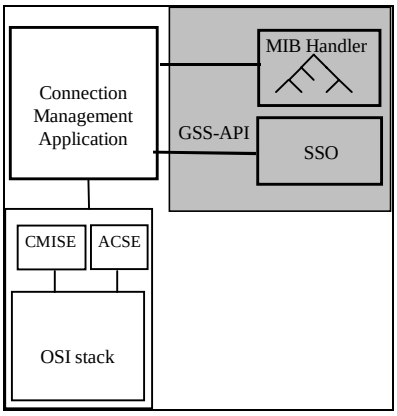

**Figure 2 - TRUMPET security architecture**

TRUMPET consortium developed a security policy for the X interface [iv], based on GSS-API [v] authentication mechanisms, and implementing mutual authentication, access control, and data integrity. Security mechanisms, supporting the developed security policy, are implemented in such a fashion that they are conceptually used as part of the communications stack (CMIS [vi]) by the management applications, while effectively the security mechanisms are implemented above the stack, thus being an OSI layer-7 add-on feature to the TMN applications, as shown on Figure 2. This means that security may be turned on and off.

## 3. Testing requirements and methodology

TRUMPET project required a testing methodology [vii] that could establish that integrity and security requirements are both fulfilled. These requirements might also be conflicting, for several reasons: some security mechanisms require a pre-transaction overhead, availability of specific information at both sides of the association, *etc*. Lack of synchronisation and coherency of security-related information, or the introduction of significant security overhead, might disrupt the basic operation, and degrade the performance of the management system - it can jeopardise its integrity status. Even if the functionality is preserved, a possible decrease of the performance/throughput of the management system can prove to be a costly drawback of the introduction of security policies.

Thus, the first aim of testing is to establish that security and integrity requirements are not opposing, *i.e.*, that the behaviour of the interaction has not changed after the implementation of the security policies, and that the performance is not significantly affected. This requires the system behaviour to be perceived in a concrete way so as to establish the baseline stand-alone system behaviour that might be compared to the system behaviour when the security policies are introduced. The second aim of testing in TRUMPET is the proof that the security policies themselves are correctly implemented, and function both under normal circumstances and security breaches. This paper focuses on the testing of the impact of introduction of security policies on the functional and performance integrity requirements and as such is not interested in assessing the features and possible flaws in the security mechanisms *per se.* There are 3 phases within this aspect of testing:
**Phase 1** - the basic behaviour of the management applications' interaction over the Xuser interface must be established, in terms of management infrastructure and support object functionality.
**Phase 2** - this behaviour must be exercised over the communications mechanism to ensure proper functioning of the communications mechanism (without security) and satisfactory performance.
**Phase 3** - it must be shown that the introduction of security mechanisms within different domains does not jeopardise the integrity requirement - the proper and correct functioning of the management system communication infrastructure. Introduction of security mechanism should not push the system outside its time limits, distort the sequencing of the messages, cause deadlock or

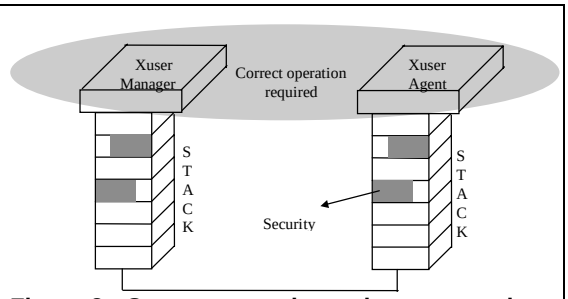


**Figure 3 - Correct operation to be preserved**

livelock situations, *etc*. The basic *behaviour* of the interaction over the X interface established in phase 1 must be *preserved*. Also, performance must be satisfactory. These two factors would imply that the basic level of communications integrity is preserved.

Figure 3 illustrates this. Functionality of the management infrastructure (stack) should not change after the introduction of security (shaded boxes), which is conceptually done within the stack. If the behaviour of the interaction of two management entities can be established, and if it can be shown that this behaviour has not changed after deploying security in the stack, the task is accomplished. Also, performance of the communications mechanism, observed during phases 2 and 3, can give a level of understanding of the impact of introducing security in the X-interface implementation.

### *3.1 Phase 1 - Specifying the X interface behaviour*

Behaviour of the X interface is specified through the Test-MIB (Management Information Base), a set of test-objects on the agent side. Concept of the Test-MIB can be used to implement basic and finite behaviour visible to any party acting through an X interface. The name Test-MIB has been chosen since the focus is on manager/agent interaction, *i.e.*, manager controlling a set of objects through an agent - this set representing the shared management knowledge. Test-MIB can be implemented as a set of managed object classes that have some typical behaviour, accommodating the integrity requirements defined above. Such behavioural envelope can be presented to any management application as a simulation of the basic X-interface behaviour.

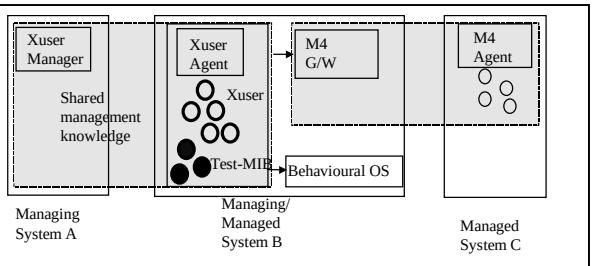


**Figure 4 - Test-MIB and the manager-agent chain**

Position of the Test-MIB in the TRUMPET manager-agent chain is shown on Figure 4 (extended from [ii]). Hollow objects represent objects of the Xuser MIB, *i.e.* the information model provided by system B (in our case PNO) to system A (VASP). In practice, some operations performed by system B on the managed objects in its MIB (on behalf of the system A) will involve further operations on the objects in system C, while others will not (depending on the actual MIB configuration). The idea of the Test-MIB (dark objects) is to implement a few objects that, instead of representing the actual MIB, just imitate the behaviour of system B's information model. As shown on Figure 4, manager calls on the Test-MIB do not propagate further as those on the Xuser do. These calls are processed in the 'behavioural' OS. The behaviour of the Test-MIB objects is just a superset of the possible behaviour of the Xuser. Thus, Test-MIB can be used at any time, without affecting the operation of the `live` system.

Typical way of representing behaviour is using the Finite State Machine (FSM) approach. This approach models the behaviour as a set of states that the system can be in. Transitions between states occur as a response to external stimuli, such as the FSM receiving a particular signal (in our case, CMIS M-Get, M-Set, M-Action call) whilst being in a correct state, or as a response to internal stimuli, such as a particular operation in the FSM being completed or an internal timer timing out.

Inconsistencies in the operation of a system modelled by an FSM are likely to occur when a system is in a state where it cannot respond to a particular input signal. This can lead to a malfunction of

the system, since no behaviour is specified as a response to that input signal. Now, sender of that signal might be expecting the receiving system to be in a state in which it would be if it did respond to the input. Also, if a system is in a state, or oscillating between a number of states, waiting for signal that can never be received, its operation is corrupt: system is in state of deadlock or livelock respectively.

If a system is designed so as to avoid all possible inconsistencies, it is robust to failure. This is optimistic, especially for time-dependent system where transitions between states depend on correct timing of external and internal stimuli. In any real-time system of considerable level of complexity, a wide range of problems will arise and correct modelling of behaviour is of paramount importance.

As seen from the above discussion, two main issues in behavioural modelling can be identified as:

- possibility of a system to deal correctly with input signals while being in a particular state.
- timing of operations performed by the system and timing of transitions between the states.

These two factors were taken as a baseline for the abstraction of Test-MIB objects' behaviour. Five basic behavioural patterns were established. These are shown in Table 1, compared to the integrity requirements they are targeting to depict and the corresponding concrete behaviour of the Xuser.

| Behavioural pattern | Integrity requirement | Example corresponding Xuser behaviour |
|---|---|---|
| Various time delays | Time-sensitive performance | Modification of the connection parameters which takes variable amounts of time |
| Rate-critical behaviour | Throughput – rate | Set of successive operations on the Xuser |
| Various values of input signals | Data integrity | Various values of parameters of the Xuser data |
| Time-critical behaviour | Timing-sequencing-liveness | Set of concurrent operations on the Xuser (case with multiple managers) |
| Sequencing of operations performed on the Test-MIB | Correct sequencing | Well-defined sequence of operations needed to reserve a connection |

**Table 1 - Behavioural patterns**

Implementation of the Test-MIB is shown on Figure 5. The behaviour of the Test-MIB objects is provided to the Xuser Agent through an Application Programming Interface (API). This behavioural code can be seen as an independent TMN OS ('a behavioural' OS). The behavioural bit of code is thus independent of the platform, and accessed through the API via defined operations. Test-MIB objects (defined in GDMO [viii] /ASN.1) implement a set of actions which represent calls to the behavioural API. So if a CMIS [vi] M-Action call is received, it is translated into a call to the behavioural API. This can be also done for other CMIS calls, such as M-Set and M-Get.

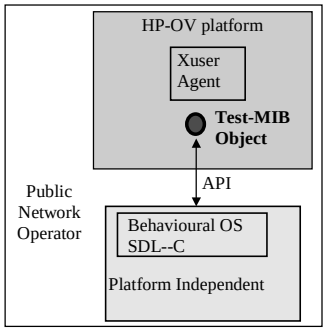


**Figure 5 - Test-MIB implementation**

Behaviour at the specification stage is done using UML [ix] State Charts, and the Specification and Description Language (SDL) [x], while implementation is done in the C language, since both manager and agent applications in TRUMPET were implemented using C.

### *3.2 Phase 2 - Testing without security*

Once the basic behaviour of the Test Objects at the agent side is established, the next step is to develop a set of test-cases initiated from the other side of the X-interface so as to test the defined behaviour over the stack.

The testing configuration shown on Figure 6 classifies, according to the conformance testing methodology [xi], as the remote testing method. The only Point of Control and Observation (PCO) is situated on the Xuser manager's API (marked in black). This test-API provides an interface similar to the CMIS M-Action interface, which enables the test M-Action calls to be invoked on the Test-MIB objects on the agent side.

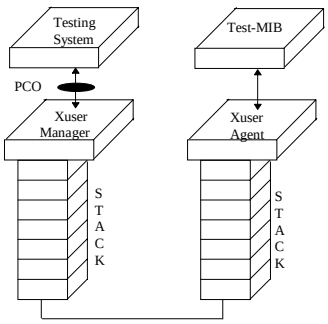

**Figure 6 - Testing configuration**

A set of test-cases targeted to exercise the Test-MIB behaviour over the management support machinery and the stack is specified in the Tree and Tabular Combined Notation - TTCN [xii] language. Behavioural part of test-cases is derived from the Test-MIB (Table 1) specification: when running the tests, both the ordering and the timing of the events to be observed on the PCO can be established and measured. These two dimensions of the test-cases give an observational framework for the comparison of secured and unsecured cases. Running the test cases ensures the proper and correct functionality of the communications stack; and it allows measurement of the time-related performance parameters, such as delay. Thus, quantifiable integrity-related information, based on functionality and performance, can be gathered.

### *3.3 Phase 3 - Testing with security*

Having tested the correct functioning and gathered data of management infrastructure performance, the next step is to prove that the introduction of security mechanisms does not impact the integrity of the inter-domain interaction. The system should still perform within its time limits, and its behaviour should stay the same - liveness should be maintained, sequencing of particular operations should not change, data integrity should be preserved, and performance should not degrade significantly.

The TRUMPET security mechanism [iv] consists of a set of security functionalities added underneath the management functionality. The management calls are thus expanded to the level of the secure management calls. The testing configuration now has the structure as shown on Figure 7.

The proof that the stack functionality and performance has not changed with the introduction of security is conducted by executing the test-cases from the phase 2 testing step (postulating that the security is transparent to the management application, which is the case in TRUMPET), and observing the verdicts and analysing the results. Behaviour (ordering of events at the PCO) needs to stay the same as in the unsecured phase, and the system should not block. Although the ordering of the events must be the same, the timing can be different, since the use of security mechanism is expected to introduce delays. Only if all the test cases established in the phase 2 are passed successfully, it can be stated that the behaviour of the Xuser did not change with the introduction of the security mechanism.

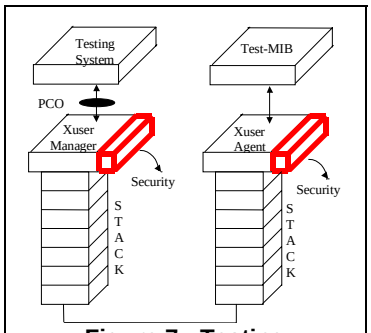

**Figure 7 - Testing configuration (with added security)**

## 4. Testing and performance results

TRUMPET manager (VASP) and agent (PNO) applications have both been implemented as single-threaded, and they communicate in a fully blocking fashion over the X interface. Hence, the performance integrity requirement of the throughput/rate is not applicable, as well as the functional

integrity requirement of sequencing. Thus, only two behavioural patterns from Table 1 were implemented: Test-MIB exhibited various time delays in the first case, and was made to be sensitive to the values of the input signals in the second. These behavioural patterns were implemented in the behavioural OS accessible through the behavioural API of one Test-MIB object (Figure 5).

Test cases implemented had two purposes: measurement of time-sensitive performance of manager-agent communication both with and without security deployed; and the proof of the robust operation and correct functionality. The functionality, in terms of liveness and data integrity, was preserved both in secured and unsecured case: data was not corrupted or lost by the communications stack and software. Moreover, the system was live at all times, independent on the data size or content.

Time-sensitive performance was assessed on different levels. Delay introduced by the communications mechanism during the establishment of the management association, and the delay when performing management operations, were measured. Next, the delays exhibited when performing management operations, which take various amounts of time to be completed by the agent, were measured. The rationale was to detect the possible agent-delay dependent jitter. Finally, the effect of different lengths of arguments of management operations on the performance of the communications mechanism in terms of the string-length dependent delays was recorded. Results of these measurements are presented in the following sections.

### *4.1 Association delays*

Mean value of the time taken to perform a management association without authentication was 197.7 ± 14.2 ms, based on the data sample of 200 values, ranging between 176.4 and 309.1 ms. Mean value of the time taken to perform the management association with mutual authentication was 2719.0 ± 385.1 ms, based on the data sample of 200 values, ranging between 2216.4 and 4625.3 ms. Thus, the average delay introduced by the mutual authentication deployed with the TRUMPET security package is 2521.3 ms. The top curve of Figure 8 represents the association delays with authentication, and the bottom one the association delays without authentication. It can be seen that the association delays for the secured management association are not just considerably higher, but also that the fluctuations around the mean value are an order of magnitude larger: the secure association establishment introduces an arbitrary delay. This feature of the TRUMPET security package could present a drawback in its use in the time-sensitive applications, where the time constraints are rigorously defined.

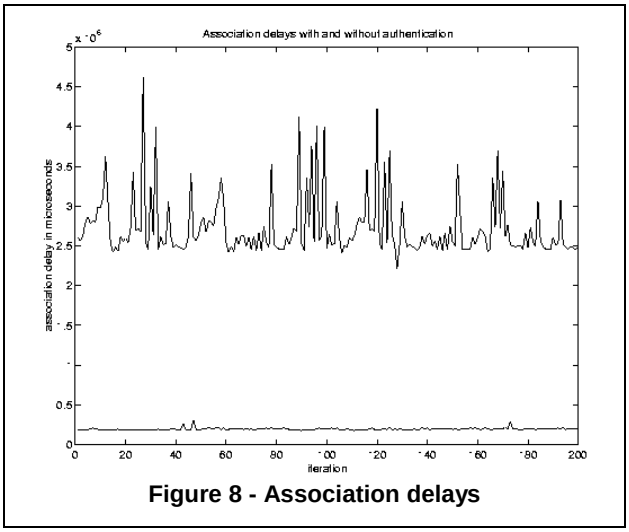

**Figure 8 - Association delays**

### *4.2 Operation delays*

Mean value of the time taken to perform an unsecured management operation (M-Action, with the simple one string parameter) was 68.7 ± 5.1 ms, based on the data sample of 500 values, ranging between 61.2 and 119.1 ms. Mean value of the time taken to perform the secured management operation was 110.7 ± 88 ms, based on the data sample of 500 values, ranging between 84.4 and 1082.9 ms. Thus, the average delay introduced by the TRUMPET security package when

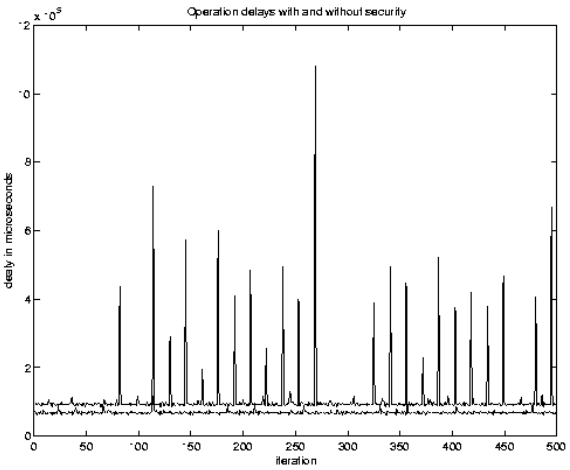


**Figure 9 - Operation delays**

performing management operations is 42 ms. The top curve of Figure 9 represents the secured management operation delays, and the bottom one the operation delays without security. Delays for the secured management operations are higher, as well as fluctuations around the mean value of the delay: the use of security package introduces arbitrary delays. The use of the TRUMPET security package in the time-sensitive applications might pose the risk to integral operation by causing these arbitrary delays. Arbitrary delays in turn could cause timeouts, affect the concurrent operation and potentially cause data incoherence, loss of liveness or similar.

### *4.3 Delay-sensitive performance*

Next, the delays exhibited when performing unsecured management operations, which take various amounts of time to be completed by the agent, were measured. The aim was to detect agent-delay dependent jitter. The agent (*i.e.*, the Test-MIB) was implemented so as to exhibit 500 delays, in the increments of 10,000 microseconds. The overall delay was measured on the test-manager side. The expected delay to be measured was thus equal to the sum of the mean delay for the management operation, plus the Test-MIB in-built delay. Figure 10 depicts the actual delay measured versus the simulated delay: the actual delay follows the simulated, apart from the 5% of the samples of the actual delay, which fall well off the mark. Figure 11 shows the delay introduced by the communications mechanism, as the function of the increasing agent-simulated delay. The 5% of the out-of-band values are not shown. Most of the values fall in the band of the average management operation delay discussed in section 4.2, with the average delay slightly higher than that for the non-delayed management operation: 96.5 as compared to 68.7 ms. Standard deviations compare more drastically, with 128,810 versus 5,1 ms. These discrepancies are due to the 5% of the samples falling considerably out of band.

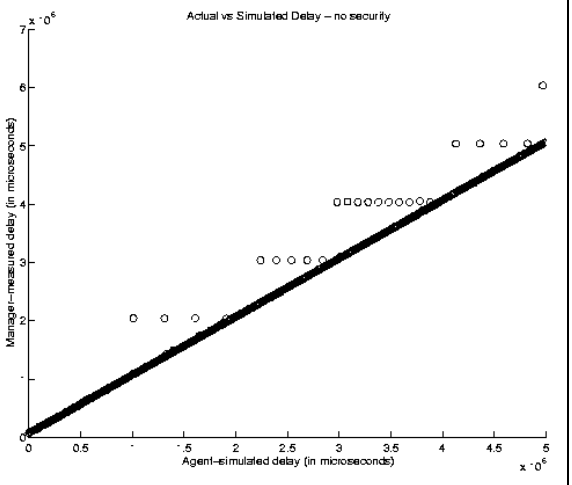


**Figure 10 - Actual vs. simulated delay, no security**

Next, the experiment was repeated with security switched on. The actual delay corresponded to the simulated, apart from the 5% of the samples which fall well off the mark, the same phenomenon as for the unsecured management operation measurements. Figure 12 shows the extra delay introduced by the communications mechanism (overall delay, but normalised), as the function of the increasing agent-simulated delay (the 5% of the out-of-band measurements are not shown). Most of the values are agent-delay independent, and fall in the band of the average secured management operation delay (section 4.2), with the average delay slightly higher than that for the non-delayed secured management operation: $117.2 \pm 105.1$ as compared to $110.7 \pm 88$ ms.

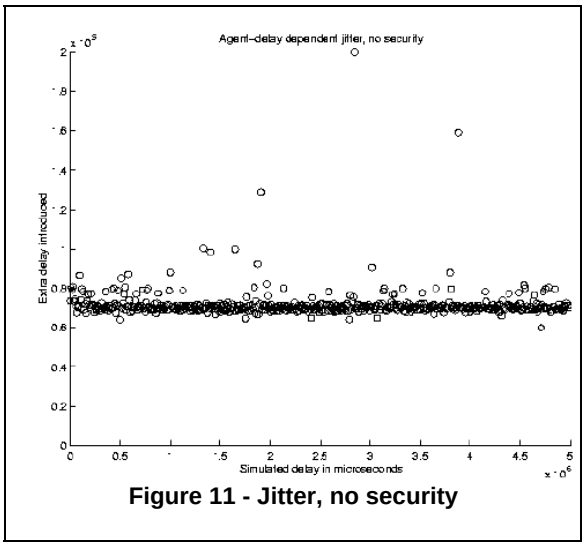


**Figure 11 - Jitter, no security**

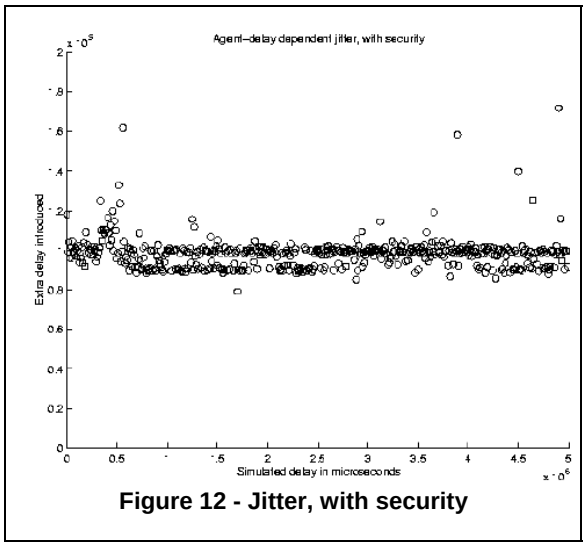


**Figure 12 - Jitter, with security**

Overall, it can be noted that the agent-simulated delay does not significantly influence the time-sensitive performance of the management communications mechanism, both with and without security. In this case, the use of TRUMPET security package does not introduce potential risks to integral operation, when compared to the unsecured operations.

### *4.4 Argument-dependent performance*

Next, effect of different operation argument lengths on the time-sensitive performance of the communications mechanism, in terms of the string-length dependent delays, was measured. The test case was run 1000 times, with string lengths 1-1000, both over the secured and unsecured management communications channel. The data integrity was preserved in both cases: the communications mechanism did not corrupt the data, and the liveness was preserved. The delays exhibited by the communications mechanism were recorded.

Delays for the unsecured management operations are shown on Figure 13: the delay increases steadily with the string length. However, there is a steep increase in the delay as the string length increases from 440 to 444: the step taking just 4 iterations, the delay increased by roughly 120%. This is most likely due to the memory allocation on the HP Open-View machine on which the manager-agent implementation was running.

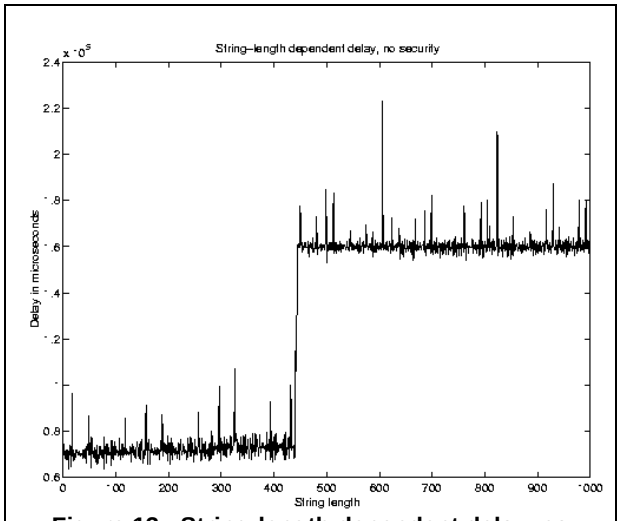


**Figure 13 - String-length dependent delay, no security**

Comparing the fixed string-length (Figure 9, section 4.2) and increasing string-length (Figure 13) delays, it can be seen that mean delay values before the step are comparable, as expected.

Finally, the measurements were repeated with the security mechanisms switched on. The delays for the secured management operations are shown on Figure 14, superimposed on the non-secured management operation delays of Figure 13. Delay increases steadily as the string length increases. However, there is a gradual increase in the delay as the string length increases from 400 to 440: the step taking 40 iterations, the delay increased by

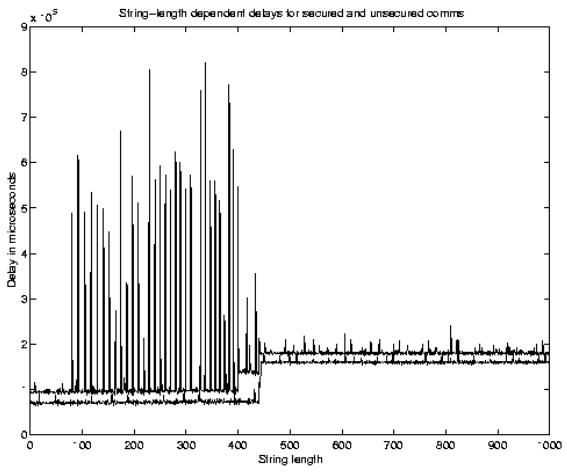


**Figure 14 - String-length dependent delays (secured and unsecured communications)**

roughly 40%. Before the step increase, the fluctuations around the mean delay value are as drastic as for the fixed-length parameter secured management operation delays discussed in section 4.2. However, after the step increase in the delay, the fluctuations settle down, becoming comparable with the unsecured operations case from Figure 13.

In both cases, the delays increase steadily as the string length increases. Both manifest a drastic delay increase when the string length reaches 440. Apart from that, both compare with their fixed string-length counterparts of Figure 9; but, for the secure management operations, the dramatic fluctuations around the mean value settle down after the string-length increase.

## 5. Discussion

In the light of the increasing level of interconnection between autonomous management systems, the key issue is that of the ability of the interconnected systems to interwork not just securely, but also in a fully integral way in terms of functionality and performance. This paper discussed the testing methodology developed for the ACTS project TRUMPET. In the TRUMPET context, the aim is to test and verify the proper functioning of the inter-domain management communications mechanism and infrastructure supporting the Xuser interface both with and without security features implemented. The second aim in this context was to quantify the performance of the communications mechanism in both of these cases, allowing the measurement of the impact of introduction of security features in the implementation of the TMN X interface and the supporting CMIS-based stack.

The testing approach developed here is not necessarily limited to the TRUMPET testing arena. This approach considered two basic aspects of integrity: *functionality* and *performance*. Thus, it can be used as a stand-alone integrity policy focusing on the integrity of communications mechanisms and infrastructure supporting interactions of specific distributed management applications. This policy can be exercised prior to the interconnection of management applications operating in autonomous domains. It can be also used to test the impact of the introduction of any additional functionality to the stack and any change in the components implementing the services inside the stack.

The Test-MIB implementation allows players involved in the interconnection to avoid exposing their real resources and particular information models during testing of the communications mechanism and management infrastructure. Instead, operators/service providers could use the Test-MIB and the ‘behavioural’ OS as a test-bed which aims to provide a superset of the possible behaviours exhibited by the shared information model (X interface). This behaviour is independent of the level of management system interaction. The management applications taking part in the interconnection can be involved, using TMN terminology, in service level interactions (as in TRUMPET), as well as network level interactions. This approach successfully distinguishes the stack and infrastructure testing from the application-specific testing, (which would include the full, detailed testing of the information models and their particular behavioural aspects mapped 1:1 with these particular information models [xiii]). Testing results would thus guarantee the required level

of integrity of the communications stack and infrastructure - possible integrity behavioural problems would be restricted to the incorrect specification of the *particular* X interface information models.

Application of this approach to TRUMPET Xuser interface testing was restricted due to the fully synchronous, blocking manager-agent communications. Thus, a full set of behaviours was not developed. The test cases implemented had two purposes: to prove the robust operation and correct functionality; and to measure the time-sensitive performance of the manager-agent communications. Functionality, in terms of liveness and data integrity, was preserved both with and without security mechanisms deployed. The data was not corrupt or lost by the communications stack and software. Moreover, the system was live at all times, independent on the data size or content.

The time sensitive performance was assessed on different levels. Delay introduced by the communications mechanism during the establishment of the management association, and the delay introduced by the communications mechanism when performing management operations were measured. Average delay introduced during association by mutual authentication deployed in the TRUMPET security package is roughly 2.5 seconds. The fluctuations around the mean value of the association delay are an order of magnitude larger for the secure management association establishment. Average delay introduced by the TRUMPET security package when performing management operations is 42 ms. The fluctuations around the mean value for the management operations are also an order of magnitude larger for the secured management operations. Hence, the use of the TRUMPET security package in the time-sensitive applications might pose a risk to integral operation by causing the arbitrary delays which could affect the concurrent operation and cause data incoherence, loss of liveness or similar.

Delays exhibited when performing management operations that take various amounts of time to be completed by the agent were also measured so as to detect the possible agent-delay dependent jitter. The agent-simulated delay did not significantly influence the time-sensitive performance of the management communications mechanism, both with and without security. In this case, the use of TRUMPET security package does not introduce potential risks to integral operation.

Finally, the effect of different lengths of arguments of management operations on the performance of the communications mechanism, in terms of the string-length dependent delays, was recorded. In both secured and unsecured cases, delays increase with the string length. Both manifest a drastic delay increase when the string length exceeds 400, most likely due to the memory allocation on the HP Open-View machine on which the manager-agent implementation was running. Besides this, both compare with their fixed string-length counterparts. However, for secure management operations, dramatic fluctuations around the mean value settle down after the string-length increase.

## 6. Conclusions and future work

Overall, it can be pointed out that the TRUMPET security package introduces arbitrary delays during secure management association establishment and when performing secure management operations. The aim of this experiment was not to diagnose in detail the cause of such performance degradation: the goal was to show how complex inter-domain interactions might appear sensitive to the introduction of additional sophisticated features. The arbitrary delays appear to be an inherent feature of the security package, and as such might have a severe impact on the integral operation of concurrent, real-time management applications communicating in an asynchronous fashion. Arbitrary delays might cause timeouts in either manager or agent applications, which otherwise would not appear. The timeouts could cause unforeseen changes of state information in either management application, which could in turn lead to possible livelock situations. The arbitrary

delays could also affect the coherence of the management data. Thus, the core management applications using the TRUMPET security package might need to be re-engineered, taking into account the performance degradation, so as to retain highly integral operations. The time-outs would need to be redesigned, and data coherence policies would need to be enforced.

In conclusion, it has to be emphasised that the integrity, security and performance requirements are closely inter-linked. As such, they must be considered throughout the system development lifecycle, starting from requirements capture down to implementation, so as to avoid not just possible inconsistencies in system operation, but also the need to re-engineer applications post-facto [xiv].

As part of the future work, the testing and performance measurement approach presented in this paper can be developed further. Firstly, the Test-MIB and the test-cases, since based on high-level UML and SDL specifications, can be expanded so as to be applicable in the emerging CORBA and JAVA management environments. Moreover, a more thorough analysis of possible behaviours of different information models can be considered so as to allow a complete superset of behaviours to be developed in order to aid the testing of actual applications, in different scenarios of inter-domain interaction. This policy would give a level of assurance of correct and proper functioning of the communications mechanism and management infrastructure to both parties involved in the interconnection. The second benefit of this policy is that it gives a quantitative notion of the integrity features of the management communications mechanism, in terms of time-critical performance aspects of its operation.

## 7. References


[i] L. Sacks, O. Prnjat, M. Wittig, M. M. Kande, B. Bushnan, S. Mazaher, C. Autant, “TRUMPET Service Management Architecture”, EDOC’98 Workshop Proceedings, San Diego, CA, November 1998.

[ii] ITU-T Recommendation M.3010, “Principles for a Telecommunications Management Network”, 1992.

[iii] K. Ward, “Impact of Network Interconnection on Network Integrity”, British Telecommunications Engineering, January 1995, vol. 13, pp. 296-303.

[iv] F. Gagnon, L. Sacks, L. Strick, J. Olnes, “A Security Architecture for TMN Inter-Domain Management”, IS&N ’97 Conference Proceedings, May 1997.

[v] RFC 2078, “Generic Security Service Application Program Interface”, Version 2, 1997.

[vi] ISO/IEC 9595, ITU X.710, “Management Information Service Definition - Common Management Information Service Definition”.

[vii] O. Prnjat, L. Sacks, H. Hegna, “Testing Integrity vs. Security Requirements on the TMN X Interface”, EUNICE ’98 Network Management Workshop Proceedings, Munich, Germany, September 1998.

[viii] ISO/IEC 10165-4, ITU X.722, “Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects”.

[ix] Rational Software Corporation, Unified Modelling Language, http://www.rational.com

[x] CCITT, Specification and Description Language SDL, June 1994; ITU-T, Z.100, 1993.

[xi] ISO/IEC 9646, “Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 1: General Concepts“, 1992.

[xii] ISO/IEC 9646, “Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN)“, 1992.

[xiii] R. Eberhardt, S. Mazziotta, D. Sidou, “Design and Testing of Information Models in a Virtual Environment”, IM'97, The 5th IFIP/IEEE International Symposium on Integrated Network Management, San Diego, USA, May 1997.

[xiv] O. Prnjat, L. Sacks, “Integrity Methodology for Interoperable Environments”, IEEE Communications, Special Issue on Network Interoperability, May 1999, Vol. 37, No. 5, pp 126-139.