Project: DRA FRONT END FILTER PROJECT

Title: FEF Project Final Report

Ref: DS/FMU/FEF/047 *Issue: Revision* : *2.1* *Date:* 5 June 2016

Status: Approved *Type:* Report

Keywords:

Author:

Name	*Location*	*Signature*	*Date*
R.B. Jones			

Authorisation for Issue:

Name	*Function*	*Signature*	*Date*
R. B. Jones	HAT Manager		

Abstract: This document is the final report for the DRA Front End Filter project RSRE 1C/6130, giving an overview of the achievements of the project.

Distribution: HAT FEF File

0 DOCUMENT CONTROL

0.1 Contents List

List of Tables

0.2 Document Cross References

[1] Michael J.C. Gordon and Tom F. Melham, editors. *Introduction to HOL.* Cambridge University Press, 1993.

[2] J.M. Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall, 1989.

[3] *Secure Database Technical Proposal - Amendent 1.* High Assurance Team, ICL Secure Systems, WIN01, 11th February 1992.

[4] *Secure Database Technical Proposal - Amendent 2.* High Assurance Team, ICL Secure Systems, WIN01, 22nd July 1992.

[5] DS/FMU/017. *Secure Database Technical Proposal.* High Assurance Team, ICL Secure Systems, WIN01, 21st January 1992.

[6] DS/FMU/FEF/001. *Project Overview Document.* G.M. Prout, ICL Secure Systems, WIN01.

[7] DS/FMU/FEF/002. *Errors in the Specifications.* G.M. Prout, ICL Secure Systems, WIN01.

[8] DS/FMU/FEF/003. *Formal Security Policy.* G.M. Prout, ICL Secure Systems, WIN01.

[9] DS/FMU/FEF/005. *Specifications of hide and updateState*. G.M. Prout, ICL Secure Systems, WIN01.

[10] DS/FMU/FEF/006. *Security Conjecture for the SSQL Abstract Machine.* G.M. Prout, ICL Secure Systems, WIN01.

[11] DS/FMU/FEF/014. *Specification of SSQL Semantics II.* G.M. Prout, ICL Secure Systems, WIN01.

[12] DS/FMU/FEF/015. *Proof of Security (IIe).* G.M. Prout, ICL Secure Systems, WIN01.

[13] DS/FMU/FEF/018. *Proposal for Phase 2.* G.M. Prout, ICL Secure Systems, WIN01.

[14] DS/FMU/FEF/021. *Specification of TSQL.* G.M. Prout, ICL Secure Systems, WIN01.

[15] DS/FMU/FEF/022. *SWORD Front End Architectural Model.* R.D. Arthan, ICL Secure Systems, WIN01.

[16] DS/FMU/FEF/024. *A HOL Specification of the SWORD Output Filter.* G.M. Prout, ICL Secure Systems, WIN01.

[17] DS/FMU/FEF/025. *Representation of an SSQL State as a TSQL State.* G.M. Prout, ICL Secure Systems, WIN01.

[18] DS/FMU/FEF/026. *Critical Requirements on the SWORD Query Transformations.* R.D. Arthan, ICL Secure Systems, WIN01.

[19] DS/FMU/FEF/029. *Specification of Query Transformations in HOL (II).* R.D. Arthan, ICL Secure Systems, WIN01.

[20] DS/FMU/FEF/032. *Table Computations for SWORD.* R.D. Arthan, ICL Secure Systems, WIN01.

[21] DS/FMU/FEF/034. *Phase II Proof Strategy.* R.D. Arthan, ICL Secure Systems, WIN01.

[22] DS/FMU/FEF/036. *Phase II Proof Finale.* R.D. Arthan and G.M. Prout, ICL Secure Systems, WIN01.

[23] DS/FMU/FEF/039. *Proposal and Quotation for Phase 3.* R.D. Arthan, ICL Secure Systems, WIN01.

[24] DS/FMU/FEF/046. *Phase 3 Technical Overview and Final Report.* R.D. Arthan, ICL Secure Systems, WIN01.

[25] DS/FMU/IED/USR007. *ProofPower Installation and Operation.* Lemma 1 Ltd., `http://www.lemma-one.com`.

[26] DS/FMU/IED/USR014. *ProofPower Software and Services.* Lemma 1 Ltd., `http://www.lemma-one.com`.

[27] *Security Properties of the SWORD secure DBMS Design.* Simon Wiseman, DRA.

[28] *SSQL Transformations.* Simon Wiseman, DRA, 14th January 1993.

0.3 Changes History

1.2 (14 March 1994) First draft for DRA.

Revision : *2.1* **(5 June 2016)** Final report.

Issue 2.2 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document is a report on the results of the formal specification and verification work carried out by ICL as part of the development by DRA of SWORD, a secure relational database system. ICL's work on this project is generally referred to as the "FEF project" in the rest of this document.

1.2 Introduction

This report is divided into sections as follows:

Section 2 describes the objectives of the formal methods work.

Section 3 describes the key achievements against the identified objectives.

Section 4 tabulates the deliverables to DRA.

Section 5 describes the results of the FEF project, particularly considering the significance of the various formal specifications and of the theorems which have been proved about them.

Section 6 comments on the cost effectiveness of the work undertaken.

Section 7 offers conclusions.

1.3 Terminology and Abbreviations

CLEF Commercial Licenced Evaluation Facility

DBMS Database Management System

FEF Front End Filter

This term is used to describe an implementation technique for SSQL. SSQL queries provided by a client user or process are transformed into SQL queries which are then executed on a standard commercial DBMS. Output from the SQL queries is filtered to remove sensitive information before it is returned to the client. In this document the phrase 'FEF project' is used to refer to ICL's work on the formal specification and verification of aspects of the FEF development being undertaken by DRA.

HAT High Assurance Team, the unit within ICL which carried out the work described in this document.

ITSEC Information Technology Security Evaluation Criteria

RDBMS Relational DBMS

SSQL Secure SQL

A variant of the SQL database query language with features supporting multilevel security. SSQL differs from SQL mainly in that there is an additional data type, *Class*, for security-classifications, which is used to classify the information held in tables and their constituent rows and columns.

SQL Structured Query Language

A standard relational query language for querying and updating a database.

TOE Target Of Evaluation

TSQL Target SQL

A variant of SQL, augmented with a data type *Class* which can be used to represent security-classifications, but which has no other special interpretation.

2 THE OBJECTIVES OF THE FORMAL METHODS WORK

The objectives of the formal methods work undertaken under contract RSRE 1C/6130 (as set out in Annex 9 to the ITT) were:

- To provide an assessment of the applicability of computerised mathematical theorem proving tools to the construction of secure systems of realistic size and complexity.
- To result, in combination with DRA in-house work, in the production of a prototype secure DBMS capable of being evaluated to a medium level of assurance.

The work has fallen into three distinct phases which had more specific objectives as follows.

2.1 Phase 1

In the first phase the primary objective was as given in the following extract from the *Secure Database Technical Proposal* [5, section 2.1]:

> *The production of a machine-checked formal proof of the security of an abstract machine modelling a secure database supporting the query language SSQL as defined by DRA in Annex 2 to the ITT.*

2.2 Phase 2

The second phase of the work was to involve formal modeling of the design of the front end filter rather than the SSQL semantics. After reviewing the requirements for Phase 2 in the light of the results of Phase 1, the following revised statement of objectives was agreed and recorded in the *Proposal for Phase 2*, [13, section 1.2]:

> *To use the proof process to discover flaws in the specifications.*
> *To evolve the specification during the course of the proof as necessary to render it satisfactory.*

The following extract from [13] amplify these objectives:

> *The main purpose of the requirement for Phase 2 in the original ITT was to verify the design of the Front End Filter. The most important (if not the only) concern was that the filter would result in a secure database.*
> *[The] proposal for Phase 2 therefore attempts to provide the most effective way of employing formal modelling and proof to improve confidence in the security of the Front End Filter.*

The specifications referred to in the *Proposal for Phase 2* are those of the Front End Filter. The relevant notion of security is that in the formal security policy [8].

2.3 Phase 3

Phase 3 was originally to comprise solely the preparation of a final report. Towards the end of phase 2 it was agreed to extend the scope of phase 3 to embrace an additional report and further technical work on multi-level security modelling. The following extended requirements for Phase 3 were identified in the *Proposal for Phase 3* [23].

> *Managerial Final Report: This is to provide a managerial overview of the achievements of the FEF project and to comment on the cost-effectiveness of the formal methods work.*
>
> *CLEF Report: This is to provide a technical overview of the formal treatment of the Front End implementation of SWORD and to comment on issues such as the effectiveness of the formal methods work in exposing security-relevant problems with the design and other issues of relevance to evaluation of the SWORD implementation.*
>
> *Multi-level Objects: DRA have carried out some research, reported in [27], into extensions of the security policy modelling for SWORD to embrace SSQL queries which are themselves treated as structured multi-level objects. This work endeavours to formulate a non-interference style security policy for systems dealing with such objects and to define a result-labelling property which is intended to capture formally certain intuitions about the desired behaviour of the SWORD implementation. However, as discussed in [27], there are some technical difficulties with the proposed definition of the result-labelling property. DRA would like ICL to give a more formal treatment of the topics discussed in [27] and to investigate how the intuitions behind the result-labelling property may best be formalised.*

This document constitutes the Managerial Final Report mentioned above.

3 ACHIEVEMENTS IN RELATION TO OBJECTIVES

The key achievements in relation to the stated objectives are summarised in sections 3.1 and 3.2 below. A more detailed description of the results may be found in Section 5. A description of the errors detected in the specifications as a result of the formal methods work is given in Section 5.5.

3.1 Phase 1

The original objectives took the required Phase 1 proof to be concerned with *confirming* the correctness of the specifications (w.r.t. the formal security model), while acknowledging a risk that the specifications might prove not to be correct.

In fact, the formal work proved more valuable than had been anticipated. Significant numbers of minor errors were discovered in the specifications and corrected during the formalisation, syntax checking and type checking of the specifications.

During the proof work one error was discovered which rendered the SSQL abstract machine as specified insecure. This error was corrected and the formal proofs were completed.

Thus the primary objective of Phase 1 was fully satisfied, and in addition a number of errors were detected and corrected in the original specifications.

3.2 Phase 2

The objectives of the Phase 2 work were reviewed prior to the commencement of Phase 2, and re-oriented toward using the formal specification and proof work for detecting errors in the design specifications for the Front End Filter.

Partial formal proofs of the security of the SSQL system as implemented using the Front End Filter were then undertaken.

Though the greater complexity of the security-relevant aspects of the specifications (by comparison with Phase 1) prevented completion of the relevant proofs, partial proofs were accomplished as planned. The rate of detection of errors was similar to that in Phase 1, but the proportion of substantive errors in the security of the specifications was significantly higher in Phase 2 than in Phase 1.

3.3 Phase 3

The two reports reports on overall achievements were produced, this document being one of them.

The technical component of the phase 3 work resulted in a generic formulation of a multi-level security policy. This was used to give a formal expression of the multi-level policy for SWORD along the lines discussed in [27]. Various approaches to formulating the result labelling property were considered. The approach finally chosen was checked with some proof work and appears to solve the main technical difficulties and to provide a good basis for a possible future formal treatment of an actual system design for which this type of property is required.

4 DELIVERABLES

4.1 Phase 1

In table 1 are shown the deliverables from [5, 3, 4], together with the name of the document or documents constituting each deliverable [1]:

WP	Code	Description	Document
WP1a	D1	Formal Security Policy	fef003
WP1a	D2	Specifications of 'hide' and 'update'	fef005
WP1a	D3	Specifications of SSQL semantics I	fef004
WP1a	D3	Specifications of SSQL semantics II	fef014
WP1a	D4	Specification of security conjecture	fef006
WP1b	D5	Unwinding proof scripts	fef009
WP1b	$D16_1$	Informal justification for unwinding proof	fef009
WP1c	D6	Remaining proof scripts	fef010 fef011 fef012 fef013 fef015
WP1c	$D17_1$	Informal justifications for remaining proof	fef016
WP2	D7	Proposal for Phase 2	fef018

Table 1: Phase 1 Deliverables

The following (Table 2) have also been delivered to DRA:

Description	Document
Proof Strategy	fef007
Cross Reference Index	fef008
ProofPower Theory Listings	fef017

Table 2: Additional Phase 1 Documents

4.2 Phase 2

Table 3 shows the Phase 2 deliverables, together with the description and delivery date of the material to be delivered. These are as specified in [13], amended and agreed according to a letter to DRA of 6/1/93 (ref. fef/letters/dra9).

4.3 Phase 3

The deliverables for the extended phase 3 as described in [23] are listed in table 4.

The following (Table 5) have also been delivered to DRA:

[1] As a result of an oversight in preparing the revised Phase 3 proposal, deliverable codes D16 and D17 were re-used. To disambiguate these deliverable codes they have been subscripted with the relevant phase number.

WP	Code	Description	Document
WP3	D8	Query Transformation Specifications in SML	fef019 fef020
WP3	D9	TSQL abstract machine specifications	fef021
WP3	D10	Query Transformation Specifications in HOL	fef022 fef024 fef025
WP3	D11	SSQL Implementation Model Specifications	fef028 fef029
WP3	D12	Specification of Security Propositions	fef026 fef032 fef034
WP3	D13	Phase 2 Proof Scripts	fef031 fef033 fef035
WP4	D14	Report on Phase 2 Proofs	fef036

Table 3: Phase 2 Deliverables

WP	Code	Description	Document
WP7a	D15	Managerial Final Report	fef047
WP7b	$D16_3$	CLEF Report	fef041
WP7c	$D17_3$	Formal specification of non-interference and result-labelling property for multi-level objects	fef040 fef042 fef043
WP7d	D18	Report on consistency and other proof opportunities	fef048
WP7e	D19	Report on relationship between phase 2 and phase 3 treatments	fef046

Table 4: Phase 3 Deliverables

Description	Document
Overview and Index	fef001
Cross Reference Index	fef008
Phase 3 Theory Listings	fef045

Table 5: Additional Phase 3 Documents

fef001 includes an index of all documents produced during the course of the contract.

4.4 Electronic Deliverables

Magnetic copies of all deliverables have been supplied in a form suitable for re-running all the specification and proof processing through ProofPower, for which a makefile is supplied. A number of additional documents are mentioned in the Overview and Index [6], which are available in electronic copy only. All documents can be re-printed if required using the LaTeX software a copy of which is supplied with ProofPower, including those for which no hard copy has been supplied. They may also be viewed on screen using *dvipage*, also supplied with ProofPower.

A standard ProofPower issue tape configured for a processor nominated by DRA has also been supplied, together with a hard-copy ProofPower installation guide [25].

5 DESCRIPTION OF RESULTS

5.1 Introduction

The work under the FEF project considered in this document was commissioned and carried out in three phases. The bulk of the work was carried out between March 1992 and December 1993.

The main technical aim of Phase 1 was to produce a formal specification of the semantics of SSQL in a form suitable for processing using a computerised theorem-proving system and to provide a machine-checked proof that the SSQL semantics provided information-flow security according to an agreed formalisation of the 'No Flows Down' security policy. An important feature of the approach taken was to structure the formal specification of the SSQL semantics so that security critical aspects were clearly separated out.

The original intention for Phase 2 had been to verify that the formal model of the Front End implementation of SWORD was a 'secure refinement' (as defined in [5] Section 6.1) of the SSQL abstract machine (as specified in Phase 1) and so to demonstrate that it was secure. However, the requirements for Phase 2 were reviewed towards the end of Phase 1, and it was agreed that the complexity of the implementation was likely to make a full correctness proof prohibitively expensive. Moreover, complete informal specifications of the implementation were not expected to be available soon enough. The revised objectives for Phase 2 were therefore to use formal specification and proof to discover any security flaws in the informal specifications, and to evolve the formal specifications to eliminate any flaws discovered.

Work in Phase 3 was concerned with extended the formal analysis to capture more fully the intended benefits arising from the proposed use of labels in multi-level results, and with the provision of various reports on the work previously undertaken. On the technical side the results were to include a formalisation of a 'labelling property', together with consistency and other relevant proofs relating to this property.

The deliverables of the FEF project mainly comprise documents containing specifications or proof scripts. Specifications and theorems are expressed in ProofPower-HOL, a version of Higher Order Logic supported by ProofPower [26]. ProofPower-HOL is a formal logic suitable for specifying models of systems, and for formulating and proving theorems about such models. The concrete syntax for ProofPower-HOL is in some ways similar to the Z notation [2], but the underlying abstract syntax

and the logical system follows very closely that of HOL88 [1]. ProofPower is a tool which gives support for specification and proof in ProofPower-HOL (and other formal notations) via comprehensive programmable facilities for developing and interrogating a database of specifications and theorems. The database is organised as an extensible hierarchy of modules referred to as "theories". It also gives facilities for preparing documents containing a mixture of narrative and mathematical material. Theory listings may be automatically included in such documents, giving a precise summary of the mathematical content of specifications and of any theorems which have been proved. Most of the FEF project deliverables are documents containing both narrative and the formal material defining a theory together with the theory listing.

The results of the formal treatment from the two phases are discussed in more detail in sections 5.2 and 5.3 below.

In addition to the formal documents themselves, a key output of the work is the feedback to the DRA designers about security and other problems which were found during the course of the work. This is discussed briefly in Section 5.5. The document [7] gives a complete account.

5.2 Phase 1 Results

5.2.1 Overview

The main specification work in Phase 1 was the specification of a specific, although quite abstract, system to which the Security Model is formally applicable. This system, the SSQL Abstract Machine is intended to serve as a formal definition of the SSQL semantics. The main proof work in Phase 1 was to prove that the SSQL Abstract Machine is secure in the sense of the Security Model. The specification of the SSQL Abstract Machine is structured in such a way that the critical security features are separated out from the specification of detailed functionality. This makes clearer what is and is not essential to security and simplifies security proofs.

Sections 5.2.2 to 5.2.5 below discuss the Phase 1 results and their significance in greater detail.

5.2.2 Security Model

The Security Model[8] defines a property, *secure*, of behavioural models of systems. This is a 'non-interference' formulation in which it is required that highly classified inputs do not 'interfere' with the values of less highly classified outputs. This property is later applied to state transition models via a process of behavioural abstraction. The Security Model was actually formalised and agreed in advance of Phase 1 and was taken as the common overall statement of critical security requirements for both phases of the work.

5.2.3 SSQL Semantics

The specification of the semantics of SSQL describes how the result of an SSQL query is computed from the state of the database, covering both the response returned to the user and the new state.

The specification of the semantics is factored into three components, two of which contain all the security-relevant details and one of which contains the features which are not relevant to security. This structure is described in [5] Section 6.3.4.

Of the two security-relevant parts one (known as 'hide' [9]) determines the constraints on read access to the database which apply at the clearance of the query submitted, and the other ('update' [9]) determines the constraints on write access to the database at the relevant clearance. In the context of these constraints the detailed semantics of the query language ('process query' [11]) can be specified in a way which does not contribute to the complexity of the security proof.

The specification of 'process query' is incomplete (that is, loose), in certain respects. Of the four kinds of query (*insert*, *delete*, *update* and *select*), it is only for *select* that full details of the functionality are specified. The specifications together contain sufficient information to establish that the SSQL Abstract Machine of Section 5.2.4 will be secure however the omitted details are filled in, as demonstrated by the completion of the proof against the formal model of the security policy.

5.2.4 SSQL Abstract Machine

To establish conformance of the system to the policy a 'behavioural model' of the system is required. This is constructed in two stages from the components of the SSQL semantics.

In the first stage a model of the target system as a state transition system is constructed. This is known as the 'SSQL Abstract Machine'.

In the second stage an operation of 'behavioural abstraction' is performed on this abstract machine to give a behavioural model of the kind required in the security model.

The SSQL Abstract Machine consists of two parts:

- The transition function.
- The initial state of the system.

The transition function is constructed from the three components of the SSQL semantics. A view of the database state is constructed which hides anything not in the user's clearance (using the 'hide' component of the semantics). The query is then processed against this view (and thus with 'no read up') using the 'process query' component. The results of the query are then applied to the database using the 'update' component of the semantics, which also computes the output to be supplied to the user. The 'update' component is responsible for ensuring that there is no 'write down'.

The initial state is loosely specified as conforming only to essential security invariants on the state.

The SSQL abstract machine is documented in [10] under the name of *SSQLam*.

5.2.5 Theorem about *SSQLam*

The following theorem is proved as the key result in Phase 1:

$$\vdash behaviours\ SSQLam \in secure$$

A theorem is expressed in ProofPower-HOL as an optional list of assumptions followed by the turnstile symbol, "$\vdash$", followed by a formula, called the conclusion of the theorem. The theorem is the assertion that the conclusion is true provided all the assumptions are. The ProofPower system ensures that the only theorems which can be computed are ones which have been derived from an

identifiable list of axioms and definitions via mathematically sound rules of reasoning. In this case, there are no assumptions, and the conclusion expresses a relationship between three named objects whose definitions form part of the Phase 1 specification work. The theorem asserts that the result of applying the behavioural abstraction operation, *behaviours*, to the SSQL Abstract Machine, *SSQLam* produces a system which belongs to the set, *secure*, of systems which conform to the Security Model.

The proof is documented in [12] under the name *secureSSQL*. The proof comprises a formal validation of the fact expressed by the above theorem, viz., that the behaviours of the abstract-state machine *SSQLam* satisfy the information-flow constraints imposed by the Security Model of Section 5.2.2.

5.3 Phase 2 Results

5.3.1 Overview

In the Phase 2 specifications, various aspects of the actual Front End implementation of SWORD were modelled with a view to doing relevant proof work to discover security problems in the design. The specifications were structured to expose security critical aspects of the design so that proof work could be done in the areas where most benefit was likely to be derived. At the top level a formal model is given of the overall system architecture. This gives rise to a decomposition of the critical security properties into properties of the top level subsystems of the actual Front End design; it also gives an opportunity to verify that the critical properties on the subsystems are sufficient for overall system security. However, these subsystems do not partition conveniently into security-critical and non-critical parts. At the next level down the interactions between the subsystems were investigated by relating them to a slightly more abstract view of query execution. At the third level, the constraints on query execution are re-expressed in terms of an explicit model which can be directly related to the SSQL syntax and which specifies precisely the security-relevant computations which are to be performed when a query is executed.

The main proof work was done at the third level, to prove that the model satisfies information-flow constraints which, intuitively, at least, bear a close relationship with the overall Security Model. Proof work was also done to give a partial proof that these information-flow constraints are sufficient to ensure conformance of the system to the security requirements. This amounts to showing that satisfaction of the information-flow constraints, together with some plausible assumptions about other aspects of the system, does entail that a system constructed according to the architectural model will conform to the Security Model. A small amount of pilot proof work was carried out at the other two levels (however, the proof work at the top-level is not further discussed below).

Sections 5.3.2 to 5.3.9 below discuss the Phase 2 results and their significance in greater detail.

5.3.2 Specification of Architecture

Specification of the architecture of a DBMS, as a construction in terms of specified subsystems. The construction itself is called *FE_SWORD*, [15], and the architectural subsystems are:

- *TSQLtf*, [14], corresponding to a "conventional DBMS", implementing *TSQL*. This may or may not be implemented in reality by a "standard commercial DBMS", and so to avoid taking a position on this issue this architectural component will here be called the "TSQL engine".

- *STP* ("SSQL Transformation Processor", [19]) which transforms queries input in SSQL to queries in TSQL to be submitted to the TSQL engine.
- *outputFilter*, [16], which sanitises results from the TSQL engine before passing them back to the user.

The construction also requires a state-representation function, specified as *reprState* in [17].

5.3.3 Specification of Security-Critical Properties

Specification of security-critical properties of the architectural construction and subsystems. These are documented in [15] under the names:

subsys_secure
subsys_secureA
subsys_secureB
subsys_secureC
subsys_secureD
subsys_secureE

5.3.4 Subsystem Lemmas

The statement of a collection of lemmas, which may be called the 'subsystem' lemmas. Taken together, these lemmas assert that the security-critical properties (Section 5.3.3 above) are true of the construction and subsystems of the architecture.

Thus a proof of security assuming the lemmas will be a proof of the correctness of the architectural structuring as a first design-step. Since the lemmas are stated in terms of specifications of the individual subsystems of the architecture, they represent proof-obligations which are carried forward to later design-steps. The subsystem lemmas are documented in [21] under the names:

Architecture_Secure
Subsys_SecureA
Subsys_SecureB
Subsys_SecureC
Subsys_SecureD
Subsys_SecureE

5.3.5 Specification of *FE_SWORD_SYSTEM*

FE_SWORD_SYSTEM is a DBMS; it is the architectural construction applied to the architectural subsystems. It is defined in [21].

5.3.6 Proof of *Theorem1*

Theorem1, stated and proved in [21], is

$$
\begin{array}{l}
\quad Architecture\_Secure, \\
\quad Subsys\_SecureA, \\
\quad Subsys\_SecureB, \\
\quad Subsys\_SecureC, \\
\quad Subsys\_SecureD, \\
\quad Subsys\_SecureE \\
\vdash\ FE\_SWORD\_SYSTEM\_secure
\end{array}
$$

Here there are 6 assumptions, namely the subsystem lemmas described in Section 5.3.4 above. The conclusion of the theorem here is the boolean term $FE\_SWORD\_SYSTEM\_secure$. This is defined in [21] to be equivalent to $FE\_SWORD\_SYSTEM \in secure$, (and so the theorem has a similar form to the main theorem, *secureSSQL*, of Phase 1, discussed in Section 5.2.5). This result shows that the architecture specified is correct with regard to the Security Model. In other words, the security-critical requirements on the subsystems and the construction have been correctly identified in that they are indeed sufficient to ensure the security of the constructed DBMS.

5.3.7 Specification of Execution Model

What is called the 'Execution Model' represents a conceptual step in the development process following on from the Architecture definition. The Execution Model may be described as a specification of a new constraint on the architectural subsystems *TSQLtf* and *STP*. The purpose of the constraint is to provide an alternative to the somewhat intractable semantics of TSQL for use in the further formal modelling.

The Execution Model is documented in [18] through the specification of three new objects: EM_1, *compile* and *upd*. EM_1 is a constant, while *compile* and *update* are variables (representing subsystems whose detailed specification is not available). EM_1 is a construction function for composing the two subsystems *compile* and *upd*. *compile* models compilation a query to give a formal representation of a database operation; *upd* models the use of the result of *compile* to update a database state. EM_1 composes the two subsystems by compiling and executing a query, using *update* to modify the database state or "outputting" the result of a select query as appropriate. A loose specification for *compile* and *upd* is given by a definition of correctness relative to the TSQL semantics:

$$
\quad Correct\_Compile \ = \{(compile,\ upd) \mid EM_1\ compile\ upd = TSQLtf\}
$$

5.3.8 Reduction of Proof-Obligations: *Theorem4*

Further development, on the familiar recursive pattern, consists of a design-decomposition of a subsystem into components, together with further correctness-proof work to reduce the subsystem proof-obligations to component proof-obligations.

In this case, the subsystems in question are *TSQLtf* and *STP*, taken together, and the components in question are EM_1, *compile* and *upd*. Use is made of the definitions of the Execution Model to show that the lemma *Subsys_SecureE* can be reduced to three lemmas, called:

EM_SecureE
Correct_Compile_OkSTP
TableComputationsSecure

These lemmas are documented in [21], and the relevant theorem is stated and proved in [21] as *Theorem4*:

$$\begin{array}{l} \quad Architecture\_Secure, \\ \quad Subsys\_SecureA, \\ \quad Subsys\_SecureB, \\ \quad Subsys\_SecureC, \\ \quad Subsys\_SecureD, \\ \quad EM\_SecureE, \\ \quad Correct\_Compile\_OkSTP, \\ \quad TableComputationsSecure \\ \vdash \quad FE\_SWORD\_SYSTEM\_secure \end{array}$$

which may be compared with *Theorem1* above. The approach here is that of so-called "partial proofs", in which a partial attack is made on a verification problem by successively attacking the assumptions of an initial theorem (typically vacuous: $P \vdash P$). The goal being to replace each assumption of the initial theorem with zero or more assumptions which are more plausible, or which constrain fewer or simpler components of the system.

The significance of this theorem is that assumption *Subsys_SecureE* of *Theorem1* has been shown to be reducible to a condition, *EM_SecureE* on EM_1, provided that:

- *STP* is related in a certain way to *compile*, the relationship being that specified in the definition of *Correct_Compile_OkSTP*. In essence, this says the the input-output behaviour of a select query at any state could equally well be computed by interpreting the query in the state according to a set of rules captured in the definition of a set of computations called *TableComputations*. Subject to various modelling assumptions discussed in detail in [20], the details of the definition of this set are based closely on the semi-formal definition of the query transformations specified informally in [28]. In a certain sense, *TableComputations* embodies a description of SSQL in terms of a relational algebra augmented with security features.
- All the computations in the set *TableComputations*, satisfy certain natural information-flow constraints, as specified in the definition of *TableComputationsSecure*.

5.3.9 *Theorem5*

Further proof work results in reducing *Theorem4* above to *Theorem5* below, which is documented in [22]. This result is "the best partial proof of the overall system security" achieved under the FEF project.

$$\begin{array}{l} \quad Architecture\_Secure, \\ \quad Subsys\_SecureA, \\ \quad Subsys\_SecureB, \\ \quad Subsys\_SecureC, \\ \quad Subsys\_SecureD, \\ \quad View_t\_secureE, \\ \quad outputFilter\_secureE, \end{array}$$

Correct_Compile_OkSTP,
⊢ *FE_SWORD_SYSTEM_secure*

Theorem5, in comparison with *Theorem4*, represents progress in the following respects:

- The assumption *TableComputationsSecure* has disappeared, since it has been proved as a theorem, as documented in [22]. This indicates that the bottom level of proof work as discussed in Section 5.3.1 is complete — the SSQL relational algebra satisfies the information-flow constraints imposed upon it.
- The assumption *EM_SecureE* has been replaced by two new assumptions $View_t$*_secureE* and *outputFilter_secureE*. This indicates partial progress on the second level of proof work and reflects a reduction of the assumption *EM_SecureE* of *Theorem4* to conditions, $View_t$*_secureE* and *outputFilter_secureE* on two of the constituents of EM_1.

5.4 Phase 3 Results

The main technical aim of phase 3 was to clarify by formal specification and proof the ideas of the DRA paper [27]. This paper is concerned with the support which SWORD provides for database queries and results which are themselves structured multi-level objects. There are two main aspects to this: the multi-level security policy and the result labelling property. The multi-level security policy is intended to generalise the security policy which was used in phases 1 and 2. The result labelling property is intended to give a formal description of the intentions behind the classification information appearing within the tables which are the outputs from queries to SWORD. In essence, the idea is that the classification associated with a field in a table should indicate to the client the classification of the data in the field. While the intuitions behind this idea are clear in simple cases, it turns out that of the required property is somewhat tricky to define in general (as is discussed in the postscript to [27].

In phase 3 the multi-level security policy was addressed at two levels: at an abstract level, a "generic" policy was formulated and a few of its theoretical properties were investigated; at a more concrete level, a high-level architectural model of SWORD was described. The architectural model also served to give a framework for defining the result labelling property.

A more detailed discussion of the phase 3 technical work is given in the document [24], which also serves as an introduction to the specifications produces during phase 3. [24] also discusses the technical conclusions which can be drawn from the phase 3 work.

5.5 Issues Detected

The main issues with the SSQL and Front End designs which were detected during the project are described and discussed in [7]. For present purposes these may conveniently be classified and summarised as follows::

- **Syntax and type checking errors:** the bulk of the design was conveyed to ICL in a semi-formal notation; transcription of this into a machine-checked formal language revealed errors such as misspellings of names, failure to pass arguments correctly, and omission of auxiliary

definitions. While none of these problems cause the system as specified to be demonstrably insecure, they are likely to prevent a proof of security from going through, and make a proper judgement about security difficult. In some cases the correction is obvious and the problem would therefore be unlikely to result in an insecure implementation, but in other cases it is less clear how the problem should be resolved. In these cases there is a greater risk that a misunsdertanding might arise which would cause the implementation to be insecure.

This type of problem is detected automatically as the design is transcribed and checked using the ProofPower tool.

Approximately 30 problems of this class were reported in Phase 1 and about the same number in Phase 2.

- **Incompleteness:** for example, security of the query transformations of [28] relies on the system not loading into its symbol tables information about directories and tables which the client is not cleared to see; this was not made explicit in the design material. The detection of this type of error depends on whether an object is just referenced and not defined (in which case the tool will flag the omission) or whether an object is inappropriately given a dummy definition (in which case the significance of the omission may not become apparent until proof is attempted).

  Some 13 problems of this class were reported in Phase 1 and 6 in Phase 2. Perhaps 5 or 6 of the 7 Phase 2 problems were directly relevant to security in that a security check was omitted from the design in a way which would lead directly to a violation of the Security Model if reflected in the implementation.

- **Algorithmic errors:** for example, security of the overall system is crucially dependent on clearances of fields within derived tables being computed in a way which respects the possible flow of information into the field. The functional requirements on SWORD mean that some of these computations are quite subtle. This type of error, if it is security-relevant, is certain to be exposed during a full proof of security, and has a good chance of being exposed by partial proof work as carried out in Phase 2 if the area of attack is carefully chosen.

  1 problem of this type was discovered in Phase 1 and 5 were discovered in Phase 2. All but one of these problems were discovered during the proof work rather than during formulation of the specifications, and all would lead directly to a violation of the Security Model if reflected in the implementation.

- **Modelling errors:** these are errors where the assumptions used to model a system themselves give rise to problems which would not actually arise in an implementation: an example in the FEF work was in the Phase 2 model of select query execution (*TableComputations*), which, in a sense, tries to recover from errors which cause execution to be aborted (or never initiated) in the implementation.

  Only 1 significant problem of this type arose (viz., the above one).

6 COST EFFECTIVENESS

This final report is required to address the cost effectiveness of the work undertaken on the FEF project.

Objective measures of cost effectiveness are difficult to establish in this context, partly because of the lack of comparative data, and partly becuase the SWORD project is still under way. Full information on the impact of the formal treatment is therefore not yet available.

Since the work was undertaken on a fixed price basis, additional costs arising were absorbed by the contractor, except for costs arising from changes to the requirements. By comparison with the expectations at the time of placing the contract, we believe that the project has exceeded expectations, in terms of the benefit realised. It is therefore reasonable to argue that the cost-effectiveness of the work has exceeded expectations at the time of placing the contract.

The remainder of this section adds detail by mentioning some of the ways in which the benefits realised exeeded expectations.

6.1 Phase 1

During Phase 1 the required formal machine checked verification of the specifications of SSQL was achieved. Thus 'computerised mathematical theorem proving' was shown to be applicable in this system of 'realistic size and complexity'.

Careful structuring of the specifications was however necessary to ensure that the required proofs were of manageable complexity. The specification and proof work were undertaken in HOL using ProofPower, by experience personnel at ICL supported by senior consultants with more extensive experience. If any of these crucial elements had been different then costs might have been substantially greater. For example, at the time of bidding for the contract ICL believed that the cost of the proofs required in Phase 1 would have been greater by a factor of at least 10 if the proposed restructuring of the specifications had not been achievable.

During the course of the Phase 1 work many errors were detected and corrected. The proof work therefore contributed to significant improvements in the quality of the specifications. Had these specifications been originally developed using fully formal notations and appropriate supporting tools, this benefit would have been realised probably within the cost of the original specification work.

6.2 Phase 2

When the complexity of the specifications involved in the Phase 2 work was clear full formal proofs of correctness were judged to be unrealisable within reasonable costs, and more selective formal analysis was therefore undertaken.

The revised objective for this phase was to use the formal modelling work to discover flaws in the specifications, and by this means to evolve the specifications as necessary to eliminate any flaws discovered.

The work was successful in discovering a number of problems and in general the errors found were more significant than those found in Phase 1.

6.3 Phase 3

The technical work in phase 3 was essentially research into certain issues in formal modelling of security-relevant properties of a system such as SWORD. The cost-effectiveness of this research can therefore only be conjectured by considering its likely relevance in a future work on systems similar to SWORD.

Certain parts of the phase 3 formal specifications and proofs would be directly reusable in an appropriate context: e.g., the "generic" formulation of the security policy. The technical approach used in other parts of the work could also be exploited by using it as a paradigm in applications where the formal material was not directly reusable. In particular, the treatment of the result labelling property could be adapted if it was required to capture formally the significance of classification labels like those which appear within SWORD tables.

7 CONCLUSIONS

Under the FEF contract the SWORD multi-level secure RDBMS has been subject to extensive formal modelling to validate the security aspects of the design, including the completion of a number of formal proofs about aspects of the specification and design.

The FEF contract has achieved its original objectives and has yielded benefits beyond those anticipated.

The following conclusions are suggested:

- That the use of formal methods involving the development of machine checked formal proofs is feasible on projects of this kind.
- That great care should be taken in the structuring of specifications to improve the clarity with which they address critical issues. This reduces the costs and increases the benefits obtained from formal proof development.
- That careful modelling of selected aspects of the system can enable useful proof work even where the overall complexity would make less subtle approaches prohibitively expensive.
- That greater benefits are realisable if formal languages and tools are introduced at the earliest possible stage in the development process. In particular, use of formal notations and tools during specification and design where appropriate rather than as a post-hoc exercises would be likely to be more cost effective.