

THE AUSTIN PROTOCOL COMPILER

Advances in Information Security

Sushil Jajodia

Consulting editor

Center for Secure Information Systems

George Mason University

Fairfax, VA 22030-4444

email: jajodia@gmu.edu

The goals of Kluwer International Series on ADVANCES IN INFORMATION SECURITY are, one, to establish the state of the art of, and set the course for future research in information security and, two, to serve as a central reference source for advanced and timely topics in information security research and development. The scope of this series includes all aspects of computer and network security and related areas such as fault tolerance and software assurance.

ADVANCES IN INFORMATION SECURITY aims to publish thorough and cohesive overviews of specific topics in information security, as well as works that are larger in scope or that contain more detailed background information than can be accommodated in shorter survey articles. The series also serves as a forum for topics that may not have reached a level of maturity to warrant a comprehensive textbook treatment.

Researchers as well as developers are encouraged to contact Professor Sushil Jajodia with ideas for books under this series.

Additional titles in the series:

ECONOMICS OF INFORMATION SECURITY by L. Jean Camp and Stephen Lewis; ISBN: 1-4020-8089-1

PRIMALITY TESTING AND INTEGER FACTORIZATION IN PUBLIC KEY CRYPTOGRAPHY by Song Y. Yan; ISBN: 1-4020-7649-5

SYNCHRONIZING E-SECURITY by Godfried B. Williams; ISBN: 1-4020-7646-0

INTRUSION DETECTION IN DISTRIBUTED SYSTEMS:

An Abstraction-Based Approach by Peng Ning, Sushil Jajodia and X. Sean Wang
ISBN: 1-4020-7624-X

SECURE ELECTRONIC VOTING edited by Dimitris A. Gritzalis; ISBN:
1-4020-7301-1

DISSEMINATING SECURITY UPDATES AT INTERNET SCALE by Jun Li, Peter Reiher, Gerald J. Popek; ISBN: 1-4020-7305-4

SECURE ELECTRONIC VOTING by Dimitris A. Gritzalis; ISBN: 1-4020-7301-1

APPLICATIONS OF DATA MINING IN COMPUTER SECURITY, edited by Daniel Barbará, Sushil Jajodia; ISBN: 1-4020-7054-3

MOBILE COMPUTATION WITH FUNCTIONS by Zeliha Dilsun Kırıl, ISBN:
1-4020-7024-1

TRUSTED RECOVERY AND DEFENSIVE INFORMATION WARFARE by Peng Liu and Sushil Jajodia, ISBN: 0-7923-7572-6

Additional information about this series can be obtained from

<http://www.wkap.nl/prod/s/ADIS>

THE AUSTIN PROTOCOL COMPILER

by

Tommy M. McGuire
Mohamed G. Gouda
The University of Texas at Austin

Springer

eBook ISBN: 0-387-23228-1
Print ISBN: 0-387-23227-3

©2005 Springer Science + Business Media, Inc.

Print ©2005 Springer Science + Business Media, Inc.
Boston

All rights reserved

No part of this eBook may be reproduced or transmitted in any form or by any means, electronic, mechanical, recording, or otherwise, without written consent from the Publisher

Created in the United States of America

Visit Springer's eBookstore at:
and the Springer Global Website Online at:

<http://ebooks.springerlink.com>
<http://www.springeronline.com>

To Dianne Driskell.

T.M.M.

*To the memory of my
parents.*

M.G.G.

CONTENTS

Preface	xi
Acknowledgements	xiii
1 Network Protocols	1
Protocol development problems	1
Existing solutions	5
Protocol layering	6
Protocol frameworks	9
Protocol languages	10
The Austin Protocol Compiler	12
2 The Timed Abstract Protocol Notation	15
Messages and channels	16
Processes	17
Actions	18
Statements	18
Protocol style	19
Justification	20
Details of TAP	21
Message syntax	22
Process syntax	24
Action syntax	25
Statement syntax	26
Expression syntax	28
3 Execution Models of Network Protocols	31
Two Models	31
Abstract Execution Model	32
Abstract protocol state	32
Abstract protocol execution	32
Abstract faults	33

Abstract timeout behavior	34
Abstract execution of the request/reply protocol	36
The slow request/reply protocol	37
Justification	39
Concrete Execution Model	42
Concrete protocol state	42
Concrete protocol execution	42
Delayed message propagation	45
Concrete faults	45
Concrete timeout behavior	46
Local fairness	46
Concrete execution of the request/reply protocol	46
Justification	47
4 Equivalence of Execution Models	49
Protocol states	49
Equivalent protocol states	51
State transitions	52
Computations	52
Whole computations	52
Equivalent computations	53
Proof of equivalence	53
Implementation consistency	54
Event serialization	55
Event reordering	57
Implementation completeness	63
Related work	64
5 Preserving Fairness	67
Global fairness	67
Local fairness	67
Proof of fairness equivalence	68
Fairness and the Austin Protocol Compiler	68

6	The Austin Protocol Compiler	71
	Architecture of the compiler	72
	Message handling	73
	TAP processes	74
	APC runtime interfaces	77
	Initializing and executing the runtime system	77
	Invoking C functions from TAP	79
	Message functions	79
	Architecture of the runtime system	80
	Implementation of the concrete execution model	81
7	Two examples	85
	The secret exchange protocol	85
	Hop integrity	86
	Implementation of the secret exchange protocol	87
	Behavior of the secret exchange protocol	93
	The accelerated heartbeat protocol	95
	Implementation of the accelerated heartbeat protocol	98
	Behavior of the accelerated heartbeat protocol	102
8	A DNS Server	107
	The authoritative DNS server	110
	Implementation performance	116
	Latency	118
	Throughput	120
	Overhead	121
	Performance of the Austin Protocol Compiler	122
9	Concluding Remarks	125
	Summary	125
	Future directions	126
	Enhancements	127
	Alternative compiler back ends	127
	Alternative runtime systems	128
	Bibliography	129
	Index	133

PREFACE

There are two groups of researchers who are interested in designing network protocols and who cannot (yet) effectively communicate with one another concerning these protocols. The first is the group of protocol verifiers, and the second is the group of protocol implementors.

The main reason for the lack of effective communication between these two groups is that these groups use languages with quite different semantics to specify network protocols. On one hand, the protocol verifiers use specification languages whose semantics are abstract, coarse-grained, and with large atomicity. Clearly, protocol specifications that are developed based on such semantics are easier to prove correct. On the other hand, the protocol implementors use specification languages whose semantics are concrete, fine-grained, and with small atomicity. Protocol specifications that are developed based on such semantics are easier to implement using system programming languages such as C, C++, and Java.

To help in closing this communication gap between the group of protocol verifiers and the group of protocol implementors, we present in this monograph a protocol specification language called the Timed Abstract Protocol (or TAP, for short) notation. This notation is greatly influenced by the Abstract Protocol Notation in the textbook *Elements of Network Protocol Design*, written by the second author, Mohamed G. Gouda. The TAP notation has two types of semantics: an abstract semantics that appeals to the protocol verifiers and a concrete semantics that appeals to the protocol implementors group.

More significantly, we show in this monograph that the two types of semantics of TAP are equivalent. Thus, the correctness of a TAP specification of some protocol, that is established based on the abstract semantics of TAP, is maintained when this specification is implemented based on the concrete semantics of TAP. The equivalence between the abstract and concrete semantics of TAP suggests the following three-step method for developing a correct implementation of a protocol:

1. Specify the protocol using the TAP notation.

2. Verify the correctness of the specification based on the abstract semantics of TAP.
3. Implement the specification based on the concrete semantics of TAP.

To aid in step 3 of this method, we developed the Austin Protocol Compiler (or APC, for short) that takes as input a TAP specification of some protocol and produces as output C-code that implements this protocol based on the concrete semantics of TAP. The design of the Austin Protocol Compiler is one of the main features of this monograph.

This monograph is primarily directed towards protocol designers, verifiers, reviewers, and implementors. It is also directed towards graduate students who are interested in designing, verifying, and implementing network protocols.

The authors wish to express their thanks to their friends and colleagues at the Department of Computer Sciences at The University of Texas at Austin for their encouragement and support.

The Austin Protocol Compiler software, including the compiler, runtime system, and the examples from this book, is available from the Austin Protocol Compiler home page¹.

¹<http://www.cs.utexas.edu/users/mcguire/software/apc/>

ACKNOWLEDGEMENTS

The authors would like to thank Lorenzo Alvisi, Michael D. Dahlin, Mootaz Elnozahy, and Aloysius K. Mok for their suggestions which have improved this monograph.

Tommy M. McGuire would like to thank his friends and coworkers in UTCS and elsewhere for their support: Kay Nettle, Fletcher Mattox, John Chambers, Stephanie Tomlinson, Dan Machold, Cyndy Matuszek, Toren Smith, Joe Trent, Scott Sutcliffe, Chris McCraw, Tony Bumpass, Casey Cooper, Pat Horne, Chris Kotrla, Matt Larson, Bart Phillips, Carol Hyink, Lewis Phillips and his ex-boss, Patti Spencer. Without their patience, this work would not have been completed. He is also grateful for the support and encouragement of his family.

Mohamed G. Gouda is grateful to his parents from whom he inherited his moral pursuit and work ethics. His mother, an art teacher and a school principal in Cairo, was born on June 29, 1917 and passed away on September 10, 2002. His father, a language teacher and an education official in Cairo, was born on April 1, 1916 and passed away on June 3, 1996. This monograph is dedicated to their living and loving memory.

T.M.M.

M.G.G.

Austin, TX

July, 2004