

Contents

Preface.....	vii
Overview and Goals.....	vii
Organisation and Features.....	viii
Intended Audience	x
Acknowledgements	x
1 The Software Architecture	1
1.1 Introduction.....	1
1.2 An Introduction to Software Architecture.....	2
1.2.1 The Pipe-and-Filter Model.....	3
1.2.2 The Object-Oriented Model	4
1.2.3 The Event-Based Model.....	5
1.2.4 The Layered Model.....	6
1.2.5 The Repository Model.....	6
1.2.6 The Distributed Process Model.....	7
1.2.7 The Forwarder-Receiver Model.....	7
1.3 Architecture Design Goals	8
1.4 The Object Model	9
1.4.1 User \Leftarrow Object Interaction	10
1.4.2 Action Objects.....	12
1.4.3 Data Containers.....	13
1.4.4 Key and Certificate Containers	14
1.4.5 Security Attribute Containers.....	15
1.4.6 The Overall Architectural and Object Model.....	15
1.5 Object Internals	17
1.5.1 Object Internal Details	18
1.5.2 Data Formats	20
1.6 Interobject Communications	21
1.6.1 Message Routing.....	23
1.6.2 Message Routing Implementation.....	25
1.6.3 Alternative Routing Strategies	26
1.7 The Message Dispatcher	27
1.7.1 Asynchronous versus Synchronous Message Dispatching	30
1.8 Object Reuse	31
1.8.1 Object Dependencies.....	34

1.9 Object Management Message Flow	35
1.10 Other Kernel Mechanisms.....	37
1.10.1 Semaphores	38
1.10.2 Threads.....	38
1.10.3 Event Notification	39
1.11 References.....	39
2 The Security Architecture	45
2.1 Security Features of the Architecture.....	45
2.1.1 Security Architecture Design Goals	46
2.2 Introduction to Security Mechanisms	47
2.2.1 Access Control	47
2.2.2 Reference Monitors.....	49
2.2.3 Security Policies and Models	49
2.2.4 Security Models after Bell–LaPadula.....	51
2.2.5 Security Kernels and the Separation Kernel	54
2.2.6 The Generalised TCB.....	57
2.2.7 Implementation Complexity Issues	59
2.3 The cryptlib Security Kernel.....	61
2.3.1 Extended Security Policies and Models	63
2.3.2 Controls Enforced by the Kernel.....	65
2.4 The Object Life Cycle.....	66
2.4.1 Object Creation and Destruction	68
2.5 Object Access Control.....	70
2.5.1 Object Security Implementation.....	72
2.5.2 External and Internal Object Access	74
2.6 Object Usage Control	75
2.6.1 Permission Inheritance	76
2.6.2 The Security Controls as an Expert System	77
2.6.3 Other Object Controls	78
2.7 Protecting Objects Outside the Architecture.....	79
2.7.1 Key Export Security Features	81
2.8 Object Attribute security	82
2.9 References	83
3 The Kernel Implementation	93
3.1 Kernel Message Processing.....	93
3.1.1 Rule-based Policy Enforcement.....	93
3.1.2 The DTOS/Flask Approach.....	94
3.1.3 Object-based Access Control	96
3.1.4 Meta-Objects for Access Control.....	98
3.1.5 Access Control via Message Filter Rules.....	99
3.2 Filter Rule Structure	101

3.2.1 Filter Rules	102
3.3 Attribute ACL Structure.....	106
3.3.1 Attribute ACLs.....	108
3.4 Mechanism ACL Structure.....	112
3.4.1 Mechanism ACLs.....	113
3.5 Message Filter Implementation	117
3.5.1 Pre-dispatch Filters.....	117
3.5.2 Post-dispatch Filters	119
3.6 Customising the Rule-Based Policy	120
3.7 Miscellaneous Implementation Issues	122
3.8 Performance	123
3.9 References	123
4 Verification Techniques.....	127
4.1 Introduction.....	127
4.2 Formal Security Verification.....	127
4.2.1 Formal Security Model Verification	130
4.3 Problems with Formal Verification.....	131
4.3.1 Problems with Tools and Scalability.....	131
4.3.2 Formal Methods as a Swiss Army Chainsaw	133
4.3.3 What Happens when the Chainsaw Sticks	135
4.3.4 What is being Verified/Proven?	138
4.3.5 Credibility of Formal Methods.....	142
4.3.6 Where Formal Methods are Cost-Effective.....	144
4.3.7 Whither Formal Methods?	145
4.4 Problems with other Software Engineering Methods.....	146
4.4.1 Assessing the Effectiveness of Software Engineering Techniques.....	149
4.5 Alternative Approaches.....	152
4.5.1 Extreme Programming	153
4.5.2 Lessons from Alternative Approaches	154
4.6 References	154
5 Verification of the cryptlib Kernel	167
5.1 An Analytical Approach to Verification Methods	167
5.1.1 Peer Review as an Evaluation Mechanism.....	168
5.1.2 Enabling Peer Review	170
5.1.3 Selecting an Appropriate Specification Method	170
5.1.4 A Unified Specification.....	173
5.1.5 Enabling Verification All the way Down.....	174
5.2 Making the Specification and Implementation Comprehensible	175
5.2.1 Program Cognition.....	176
5.2.2 How Programmers Understand Code.....	177
5.2.3 Code Layout to Aid Comprehension.....	180

5.2.4 Code Creation and Bugs.....	182
5.2.5 Avoiding Specification/Implementation Bugs	183
5.3 Verification All the Way Down	184
5.3.1 Programming with Assertions	186
5.3.2 Specification using Assertions	188
5.3.3 Specification Languages	189
5.3.4 English-like Specification Languages	190
5.3.5 Spec	192
5.3.6 Larch	193
5.3.7 ADL	194
5.3.8 Other Approaches	197
5.4 The Verification Process	199
5.4.1 Verification of the Kernel Filter Rules.....	199
5.4.2 Specification-Based Testing.....	200
5.4.3 Verification with ADL	202
5.5 Conclusion	203
5.6 References	204
6 Random Number Generation.....	215
6.1 Introduction.....	215
6.2 Requirements and Limitations of the Generator	218
6.3 Existing Generator Designs and Problems.....	221
6.3.1 The <i>Applied Cryptography</i> Generator.....	223
6.3.2 The ANSI X9.17 Generator	224
6.3.3 The PGP 2.x Generator	225
6.3.4 The PGP 5.x Generator	227
6.3.5 The /dev/random Generator	228
6.3.6 The Skip Generator	230
6.3.7 The ssh Generator	231
6.3.8 The SSLey/OpenSSL Generator.....	232
6.3.9 The CryptoAPI Generator	235
6.3.10 The Capstone/Fortezza Generator.....	236
6.3.11 The Intel Generator	238
6.4 The cryptlib Generator	239
6.4.1 The Mixing Function.....	239
6.4.2 Protection of Pool Output.....	240
6.4.3 Output Post-processing	242
6.4.4 Other Precautions	242
6.4.5 Nonce Generation	242
6.4.6 Generator Continuous Tests	243
6.4.7 Generator Verification	244
6.4.8 System-specific Pitfalls	245
6.4.9 A Taxonomy of Generators.....	248
6.5 The Entropy Accumulator.....	249

6.5.1 Problems with User-Supplied Entropy	249
6.5.2 Entropy Polling Strategy	250
6.5.3 Win16 Polling	251
6.5.4 Macintosh and OS/2 Polling	251
6.5.5 BeOS Polling.....	252
6.5.6 Win32 Polling	252
6.5.7 Unix Polling	253
6.5.8 Other Entropy Sources	256
6.6 Randomness-Polling Results.....	256
6.6.1 Data Compression as an Entropy Estimation Tool.....	257
6.6.2 Win16/Windows 95/98/ME Polling Results	259
6.6.3 Windows NT/2000/XP Polling Results	260
6.6.4 Unix Polling Results	261
6.7 Extensions to the Basic Polling Model	261
6.8 Protecting the Randomness Pool.....	263
6.9 Conclusion	266
6.10 References.....	267
7 Hardware Encryption Modules	275
7.1 Problems with Crypto on End-User Systems	275
7.1.1 The Root of the Problem	277
7.1.2 Solving the Problem	279
7.1.3 Coprocessor Design Issues.....	280
7.2 The Coprocessor	283
7.2.1 Coprocessor Hardware.....	283
7.2.2 Coprocessor Firmware	285
7.2.3 Firmware Setup	286
7.3 Crypto Functionality Implementation	287
7.3.1 Communicating with the Coprocessor	289
7.3.2 Communications Hardware.....	289
7.3.3 Communications Software	290
7.3.4 Coprocessor Session Control	291
7.3.5 Open versus Closed-Source Coprocessors.....	293
7.4 Extended Security Functionality	294
7.4.1 Controlling Coprocessor Actions.....	294
7.4.2 Trusted I/O Path	295
7.4.3 Physically Isolated Crypto	296
7.4.4 Coprocessors in Hostile Environments	297
7.5 Conclusion	299
7.6 References.....	299
8 Conclusion	305
8.1 Conclusion	305

8.1.1 Separation Kernel Enforcing Filter Rules	305
8.1.2 Kernel and Verification Co-design	306
8.1.3 Use of Specification-based Testing.....	306
8.1.4 Use of Cognitive Psychology Principles for Verification	307
8.1.5 Practical Design	307
8.2 Future Research.....	308
9 Glossary	309
Index.....	317



<http://www.springer.com/978-0-387-95387-8>

Cryptographic Security Architecture

Design and Verification

Gutmann, P.

2004, XVIII, 320 p. 56 illus., Hardcover

ISBN: 978-0-387-95387-8