acm
PRESS

Proceedings of the Twenty-Second Annual ACM Symposium on Principles of Distributed Computing

PODC 2003

Boston, Massachusetts, USA
July 13-16, 2003

Sponsored by
ACM Special Interest Group on Algorithms & Computational Theory
and
ACM Special Interest Group on Operating Systems

Supported by
Hewlett-Packard, IBM Research
Microsoft, Sun Microsystems Laboratories

ACM PRESS

Proceedings of the Twenty-Second Annual ACM Symposium on Principles of Distributed Computing PODC 2003

Boston, Massachusetts, USA

July 13–16, 2003

Sponsored by:
ACM Special Interest Group on Algorithms and Computational Theory
ACM Special Interest Group on Operating Systems

Supported by:
Hewlett-Packard, IBM Research
Microsoft, Sun Microsystems Laboratories

The Association for Computing Machinery, Inc.
1515 Broadway
New York, New York 10036

Notice to Past Authors of ACM-Published Articles

ACM intends to create a complete electronic archive of all articles and/or other material previously published by ACM. If you have written a work that was previously published by ACM in any journal or conference proceedings prior to 1978, or any SIG Newsletter at any time, and you do NOT want this work to appear in the ACM Digital Library, please inform permissions@acm.org, stating the title of the work, the author(s), and where and when published.

ACM ISBN: 1-58113-708-7

Additional copies may be ordered prepaid from:

ACM Order Department
P.O. BOX 11405
Church Street Station
New York, NY 10286-1405

Phone: 1-800-342-6626 (U.S.A. and Canada)
+1-212-626-0500 (All other countries)
Fax: +1-212-944-1318
E-mail: acmhelp@acm.org

ACM Order Number 536030
Printed in the U.S.A.

FOREWORD

This volume contains the 35 papers and 16 brief announcements presented at the 22nd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), which was held from July 13 to 16, 2003, in Boston, Massachusetts, USA. PODC included a special celebration honoring the work of Michael Fischer. This volume includes two talks of this celebration. This year, PODC included a Special Security Track chaired by Amir Herzberg. The goal of this track was to promote interaction between the distributed computing and the security communities. This volume contains two invited talks to this Track, as well as 8 papers and 5 brief announcements (included in the count of 35 papers and 16 brief announcements above).

The contributed papers were selected from 208 submissions to the regular presentations track and 18 submissions to the brief announcements track in electronic discussions and at a meeting of the program committee on March 29 and 30, 2003 at the Kendall Hotel in Cambridge, Massachusetts. The regular presentations were read and evaluated by the program committee, but were not formally refereed; it is expected that many of them will appear in more polished form in fully refereed scientific journals. A selection of papers will appear in a special issue of *Distributed Computing* dedicated to PODC 2003. The brief announcements were screened by the program committee based on short abstracts; it is expected that many of them will be published elsewhere (including in other conferences). The Program Committee would like to thank all the authors who submitted extended abstracts for consideration.

Partial funding for the conference was provided by Hewlett-Packard, IBM Research, Microsoft, and Sun Microsystems Laboratories. This support, together with the continuing sponsorship by the ACM SIGACT and SIGOPS, is gratefully acknowledged.

Program Committee

Marcos Aguilera, *HP Labs*
James Aspnes, *Yale*
Juan Garay, *Bell Labs*
Amir Herzberg, *Bar-Ilan U., Security Track Chair*
Markus Jakobsson, *RSA*
Dahlia Malkhi, *Hebrew U.*
Erez Petrank, *Technion*
Sergio Rajsbaum, *UNAM, Chair*
Roberto Segala, *U. Verona*

Lorenzo Alvisi, *U. Texas Austin*
Cynthia Dwork, *Microsoft*
Vassos Hadzilacos, *U. of Toronto*
Gene Itkis, *Boston U.*
Miroslaw Kutylowski, *Wroclaw U. & Signet*
Boaz Patt-Shamir, *Tel-Aviv U.*
Rajmohan Rajaraman, *Northeastern U.*
Antony Rowstron, *Microsoft*
Amin M. Vahdat, *Duke U.*

PODC 2003 BEST STUDENT PAPER AWARD WINNER

"Constant-Time Distributed Dominating Set Approximation"
by Fabian Kuhn and Roger Wattenhofer

MICHAEL FISCHER CELEBRATION

Fischer Celebration Chair

Nancy Lynch, *MIT*

In honor of his 60th birthday, the 22nd ACM Symposium on Principles of Distributed Computing featured a series of lectures illustrating and celebrating the impact of the work of Michael Fischer. Mike Fischer is one of the pioneers of the field of distributed computing theory. He is an author of about two dozen papers in this area, and one of the original founders of the PODC conference. His contributions to distributed computing theory include algorithms and impossibility results for problems such as mutual exclusion, resource allocation, consensus, global snapshots, and reliable communication in unreliable networks. His contributions to other areas, including sequential and parallel algorithms and security protocols, have been equally significant.

I am grateful to the speakers, Leslie Lamport, Albert Meyer, and Rebecca Wright for their participation in the celebration.

Nancy Lynch
Cambridge, Massachusetts, July 2003

2003 EDSGER W. DIJKSTRA PRIZE IN DISTRIBUTED COMPUTING

Sponsored by SIGACT, SIGOPS, AT&T Labs, Compaq, IBM, Intel, and Sun Microsystems.

The Edsger W. Dijkstra Prize in Distributed Computing was created to acknowledge an outstanding paper on the principles of distributed computing, whose significance and impact on the theory and/or practice of distributed computing has been evident for at least a decade. This year's award was presented at PODC 2003 to the following paper.

> Maurice Herlihy, "Wait-Free Synchronization," *ACM Transactions on Programming Languages and Systems,* vol. 13, no. 1, pages 124–149, January 1991.

Members of the selection committee were Yehuda Afek, Michael Merritt, Sergio Rajsbaum (Chair), and Sam Toueg. Vassos Hadzilacos and Nir Shavit describe the winning paper's contributions as follows.

Vassos Hadzilacos writes: In this paper Herlihy developed a beautiful and useful theory of fault-tolerant computation in distributed systems where asynchronous processes communicate by accessing shared objects of arbitrary types. He showed that objects of different types can differ widely in their ability to support fault-tolerant computations, and defined a hierarchy that classifies objects according to that ability. He also proved the universality of consensus, a fundamental result that facilitates this classification of objects and highlights the central role of the consensus problem in fault-tolerant computing. The paper builds on surprisingly diverse work by other researchers (Lamport and Peterson, who pioneered the notion of wait freedom; Fischer, Lynch and Paterson, who proved the impossibility of consensus in asynchronous message-passing systems and pioneered bivalency proofs; and Loui and Abu Amara, who demonstrated that certain specific strong synchronization primitives can help us solve consensus), but it unifies that work and extends it in truly novel and unexpected ways.

Herlihy's paper has been extremely influential in shaping the theory of distributed computing. It has also been influential in practice, by providing solid justification for modern multiprocessors to support in hardware universal synchronization primitives such as compare-and-swap rather than weaker primitives such as fetch-and-add.

Nir Shavit writes: This paper laid the structural foundation for the area of multiprocessor synchronization, and introduced two of its most fundamental notions: the Wait-free Hierarchy and the Universality of Consensus. When I try to think of a parallel to Dr. Herlihy's discoveries in other (albeit broader, arguably more important) scientific fields, the concept that comes to mind is Dimitri Mendeleyev's discovery of the Periodic Table of the Elements. In a manner similar to Mendeleyev's choice of Atomic Weight, a property introduced by others, to form a hierarchy among the elements, Herlihy chose the "solvability of consensus," a property introduced by Fischer, Lynch, and Paterson, as the defining property that forms a hierarchy among wait-free and lock-free concurrent objects. The effect of Dr. Herlihy's paper on researchers in our specialized field was similar, if not in scale, then in spirit: it introduced structure and order where formerly chaos and intuition reined, and it invigorated numerous scientists to set on a quest to extend the results and fill in the gaps.

The notion of wait-freedom can be attributed to Lamport and others in the 70s and 80s, and was a subject of great interest in the distributed computing community for several years prior to Dr. Herlihy's presentation of his paper. What was new in "wait-free synchronization" was that it pondered wait-freedom in its most general form: "Given two concurrent objects X and Y, does there exist a wait-free implementation of X by Y?" The paper provided us with many answers, but most importantly with a tool, the solvability of consensus, a tool that helped us fit together the many answers we had, and showed us for the first time how to make use of them in classifying the power of concurrent objects. It also showed us consensus's power by introducing the notion of the Universality and Universal Constructions, considered by many to be the "Turing machines" of wait-free computation. Last but not least, it is a beautifully written paper, presenting its ideas with just the right combination of formalism and simplicity, so much so that undergraduates can understand most of it after one reading.

* * *

The following is a brief historical perspective of the paper written by the author himself.

"Wait-Free Synchronization" was originally about programming language design. I did my Ph.D. work in Barbara Liskov's group at MIT. The group focused on the design of Argus, a language for fault-tolerant distributed computing that pioneered the notion of atomic transactions as a programming language abstraction mechanism.

For reasons of efficiency and modularity, Argus allowed objects to provide their own specialized atomicity (that is, synchronization and recovery) mechanisms. Unfortunately, such mechanisms don't necessarily compose. A transaction that operates on multiple objects may not be atomic, even if its effects are atomic when restricted to any individual object. In his thesis, Bill Weihl introduced the notion of a *local atomicity property*, a restriction of atomicity that supports composition by ensuring that if each object individually satisfies that property, then any transaction over multiple objects are atomic.

It is the nature of local atomicity properties that transactions must sometimes *block* waiting for one another. Atomicity properties can be evaluated by comparing the circumstances under which they force transactions to block.

When I arrived at CMU, shared-memory multiprocessors were starting to emerge as an active research area, so I started thinking about how one would design programming language abstractions for such an environment. Eventually, Jeannette Wing and I developed the notion of *linearizability*, a correctness condition that specifies the interface between a shared object and its concurrent users.

Linearizability differs from local atomicity in several curious ways. First, linearizability, is itself a local property, unlike atomicity and certain alternative shared-memory correctness conditions. Second, linearizability, unlike any kind of transactional atomicity property, *never* requires one process to block waiting for another. This observation had an implication which I was slow to explore: if mutual exclusion by locking is not a logical requirement, then is it really necessary?

One day, while looking for warblers in a Pennsylvania forest, it occurred to me that it might be instructive to devise a way to construct a wait-free FIFO queue using only read/write registers. In the next few weeks, I came up with a sequence of increasingly complicated incorrect solutions. The problems seemed to cluster around the difficulty of ensuring that two concurrent dequeuing threads would each choose different items from the front of the queue.

Like a detective in a *film noir*, I became troubled by the notion that I had seen something like this somewhere before. Naturally, I was well aware of the Fischer-Lynch-Patterson (FLP) result showing that consensus is impossible in an asynchronous message-passing system where processes can fail. I incorrectly (but perhaps not unreasonably) had assumed that FLP was a specialized technical result about the futility of constructing fault-tolerant commitment protocols for distributed transactions.

Rereading the FLP tech report, I suddenly realized that a shared FIFO queue can solve two-process consensus, while it was known that read/write registers can't. As an undergraduate, I had studied Mathematics, where it is commonplace to use simple properties to show that two complex things are not isomorphic (for example, genus for surfaces, homology for complexes, and so on). As a result, once I was aware of consensus, it seemed quite natural to use consensus numbers to identify inequivalent synchronization primitives.

The observation concerning the universality of consensus was the result of my irritated reaction to a colleague's comment that the impossibility of consensus by read/write registers implies that there would be no point in studying wait-free synchronization, since the read/write memory model, with its elegant and well-developed logics, is the only one worthy of study. Now, everything I knew about programming I had learned "on the street", since I took no Computer Science classes as an undergraduate, and I knew that real programmers routinely (even then) used instructions such as test-and-set and compare-and-swap, which had no place in the *bien-pensant* academic models of the time. The universality of consensus thus emerged from considering how one might use compare-and-swap instructions to implement a wait-free concurrent object.

In late 1987, I wrote the conference version of "Wait-Free Synchronization", and glancing at the conference calendar, submitted it to STOC, where it was rejected without comment. I resubmitted it to PODC, where it received a warmer reception. In retrospect, I am indeed fortunate that the paper appeared in PODC, since the PODC community has provided unstinting personal and scholarly support. I am very grateful.

CONFERENCE ORGANIZATION

Steering Committee

Elizabeth Borowsky, *Boston Coll.*
Soma Chaudhuri, *Iowa State U.*
Keith Marzullo, *UCSD*
Yoram Moses, *Technion*
Gil Neiger, *Intel MRL, Chair*
Sergio Rajsbaum, *UNAM*
Nir Shavit, *Tel-Aviv U.*

General Chair

Elizabeth Borowsky, *Boston Coll.*

Program Chair

Sergio Rajsbaum, *UNAM*

Local Arrangements

Mark Moir, *Sun Labs*

Treasurer

Soma Chaudhuri, *Iowa State U.*

Publicity Chair

Victor Luchangco, *Sun Labs*

SPONSORS

ACM Special Interest Group on Algorithms and Computational Theory
ACM Special Interest Group on Operating Systems

SUPPORTED BY

Hewlett-Packard, IBM Research, Microsoft, Sun Microsystems Laboratories

The Program Committee would like to thank the following colleagues for their assistance in evaluating the submissions:

Michel Abdalla
Ittai Abraham
Mustaque Ahamad
Dorit Aharonov
James Anderson
Felipe Araujo
Anish Arora
N. Asokan
Giuseppe Ateniese
Hagit Attiya
Omar Bakr
Roberto Baldoni
Boaz Barak
Stefano Basagni
Amos Beimel
Steve Bellovin
Josh Benaloh
Michael Bender
Daniel Bickson
Carlo Blundo

Paolo Boldi
Costas Busch
Levente Buttyán
Christian Cachin
Miguel Castro
Jiangzhuo Chen
Gregory Chockler
Jacek Cichon
Michele Colajanni
Manuel Costa
Lenore Cowen
Ronald Cramer
Artur Czumaj
Paolo D'Arco
Angela Dalton
Nenad Dedic
Carole Delporte-Gallet
Mario Di Raimondo
Juergen Dingel
Allen Downey
Hugues Fauconnier
Serge Fehr
Christof Fetzer
Faith Fich
Matthias Fitzi
Cormac Flanagan
Eric Fleury
Matthew Franklin
Michael Freedman
Yun Fu
Eli Gafni
Martin Gagne
Cyril Gavoille
Maciej Gebala
Rosario Gennaro
George Giakkoupis
Seth Gilbert
Virgil Gligor
Marcin Gogolewski
Philippe Golle
Marcin Gomulkiewicz
Ganesh Gopalakrishnan
Maciej Grzeskowiak
Rachid Guerraoui
Liang Guo
Shai Halevi
Timothy Harris
Maurice Herlihy
Ted Herman
Kris Hildrum
Chun-Yun Hsiao
Yuval Ishai
Jan Iwanik
Sitaram Iyer
Oleg Izmerly
Trent Jaeger
Sushil Jajodia
Stanislaw Jarecki
Prasad Jayanti
Lujun Jia
Shudong Jin
Yuh-Jzer Joung
Ari Juels
Tomasz Jurdzinski
Frans Kaashoek
Michal Karonski
Marcin Karpinski
Idit Keidar
Anne-Marie Kermarrec
Angelos Keromytis
Emanuel Kieronski
Marcin Kik
Mike Kistler
Marek Klonowski
Wojciech Kordecki
Guy Kortsarz
Dejan Kostic
Hugo Krawczyk
Antonin Kucera
Daniel Kucner
Shay Kutten
Anukool Lakhina
Riccardo Lancellotti
Mikel Larrea
Fabrice LeFessant
Ronny Lempel
Xiaozhou Li
Ninghui Li
Li Li
Vincenzo Liberatore
Mark Lillibridge
Guolong Lin
Yehuda Lindell
Andrzej Lingas
John Linn
Michele Loreti
Zvi Lotker
Victor Luchangco
Breno de Madeiros
Jan-Willem Maessen
Tal Malkin
Yishay Mansour
Ofer Margo
Jean-Philippe Martin
Keith Marzullo
Laurent Massoulie
Ibrahim Matta
Marios Mavronikolas
Robert McNerney
Roie Melamed
Stephan Merz
Mark Moir
Yoram Moses
Dalit Naor
Jeff Napper
Gil Neiger
Mikhail Nesterenko
Kobi Nissim
Damian Niwinski
Guevara Noubir
Rafail Ostrovsky
Katarzyna Paluch
Sarvar Patel
Elan Pavlov
Fernando Pedone
Andrzej Pelc
David Peleg
Adrian Perrig
Srdjan Petrovic
Benny Pinkas
Marek Piotrow
Jim Plank
Greg Plaxton
Giuseppe Prencipe
Roberto de Prisco
Bill Pugh
Yuval Rabani
David Ratacjzak
Sylvia Ratnasamy
Michel Raynal
Robbert van Renesse
Patrick Reynolds
Tom Rodeheffer
Liam Roditti
Luis Rodrigues
Bartlomiej Rozanski
Eric Ruppert
Marek Rusinkiewicz
Scott Russell
Wojciech Rutkowski
Matt Schmid
Yaron Sella
Hovav Shacham
Gauri Shah
Ilya Shanayderman
Marc Shapiro
Yuval Shavitt
Vitaly Shmatikov
Alex Shvartsman
Shakhar Smorodinky
Grzegorz Stachowiak
Scott Stoller
Gideon Stupp
Ravi Sundaram
Michael Szydlo
Anat Talmy
Roberto Tamassia
Tamir Tassa
Srikanta Tirthapura
Francisco Torres-Rojas
Yiannis Tsiounis
Gene Tsudik
Mark Tuttle
Arun Venkataramani
Krishnamurthy Vidyasankar
Berthold Voecking
Jennifer Welch
Susanne Wetzel
Jedrzej Wierzejewski
William Winsborough
Pawel Wlaz
Avishai Wool
Rebecca Wright
Peng Xie
Yue Yang
Bulent Yener
Jian Yin
Adam Young
Haifeng Yu
Moti Yung
Pawel Zalewski
Jan Zatopianski
Marcin Zawada
Li Zhang
Zheng Zhang
Ben Zhao
Lidong Zhou
Sencun Zhu
Grazyna Zwozniak

TABLE OF CONTENTS

Invited Presentations

Regular Presentations

Session 1

Session 2

Session 3

Session 4

Session 5

Session 6

Session 7

Session 8

Session 9

Session 10

Session 11

Session 12

Invited Presentations

Cryptography and Competition Policy – Issues with 'Trusted Computing'

Ross Anderson
Cambridge University
Computer Laboratory
JJ Thomson Avenue
Cambridge CB3 0FD, England
Ross.Anderson@cl.cam.ac.uk

ABSTRACT

The most significant strategic development in information technology over the past year has been 'trusted computing'. This is popularly associated with Microsoft's 'Palladium' project, recently renamed 'NGSCB'. In this paper, I give an outline of the technical aspects of 'trusted computing' and sketch some of the public policy consequences.

1. INTRODUCTION

Customers of the computing and communications industries are getting increasingly irritated at ever more complex and confusing prices. Products and services are sold both singly and in combinations on a great variety of different contracts. New technology is making 'bundling' and 'tying' strategies ever easier, while IT goods and services markets are developing so as to make them ever more attractive to vendors. These trends are now starting to raise significant issues in competition policy, trade policy, and even environmental policy.

Ink cartridges for computer printers provide a good example. Printer prices are increasingly subsidised by cartridge sales: the combination of cheap printers and expensive cartridges enables vendors to target high-volume business users and price-sensitive home users with the same products. The level of cross-subsidy used to be limited by the availability of refilled cartridges, and cartridges from third-party aftermarket vendors. However, many printer cartridges now come with chips that authenticate them to the printer, a practice that started in 1996 with the Xerox N24 (see [5] for the history of cartridge chips). In a typical system, if the printer senses a third-party cartridge, or a refilled cartridge, it may silently downgrade from 1200 dpi to 300 dpi, or even refuse to work at all. An even more recent development is the use of expiry dates. Cartridges for the HP BusinessJet 2200C expire after being in the printer for 30 months, or 4.5 years after manufacture [3] – which has led to consumer outrage [4].

This development is setting up a trade conflict between the USA and Europe. Printer maker Lexmark has sued Static Control Components, a company making compatible cartridges and components, alleging that their compatible authentication chips breach the Digital Millennium Copyright Act [7, 6]. On February 27, 2003, Judge Karl Forester ordered Static Control to stop selling cartridges with chips that interoperate with Lexmark's printers pending the outcome of the case. "The court has no trouble accepting SCC's claim that public policy generally favors competition," wrote Judge Forester. "The court finds, however, that this general principle only favors legitimate competition. Public policy certainly does not support copyright infringement and violations of the DMCA in the name of competition." So it would now appear that US law protects the right of vendors to use such market barrier technologies to tie products and control aftermarkets.

However, the European Parliament has approved a "Directive on waste electrical and electronic equipment" with the opposite effect. It is designed to force member states to outlaw, by 2006, the circumvention of EU recycling rules by companies who design products with chips to ensure that they cannot be recycled [8]. The scene looks set for yet another trade war between the USA and Europe. Which side should economists and computer scientists support?

Varian argues that tying printers to cartridges may be not too objectionable from a policy viewpoint [9]:

> The answer depends on how competitive the markets are. Take the inkjet printer market. If cartridges have a high profit margin but the market for printers is competitive, competition will push down the price of printers to compensate for the high-priced cartridges. Restricting after-purchase use makes the monopoly in cartridges stronger (since it inhibits refills), but that just makes sellers compete more intensely to sell printers, leading to lower prices in that market. This is just the old story of "give away the razor and sell the blades."

However, tying in other industries may well be:

> But if the industry supplying the products isn't very competitive, then controlling after-purchase behavior can be used to extend a monopoly from

PODC'03, July 13–16, 2003, Boston, Massachusetts, USA.
ACM 1-58113-708-7/03/0007.

one market to another. The markets for software operating systems and for music and video content are highly concentrated, so partnerships between these two industries should be viewed with suspicion. Such partnerships could easily be used to benefit incumbents and to restrict potential entrants.

In a growing number of industries, technical typing mechanisms based on cryptography, or at least on software that is tiresome to reverse engineer, are being used to control aftermarkets:

- Mobile phone manufacturers often earn more money on batteries than on the sales of the phones themselves, so have introduced authentication chips into the batteries. A mobile phone may refuse to recharge an alien battery, and may turn up the RF transmitter power to drain it as quickly as possible. In Morotola's case, battery authentication was represented as a customer safety measure when it was introduced in 1998 [10];
- Carmakers are using data format lockout to stop their customers getting repairs done by independent mechanics. In the case of the writer's own car, for example, the local garage can do a perfectly adequate 10,000 mile service, but does not have the software to turn off the nagging 'service due' light on the dashboard. Congress is getting upset at such practices [12];
- Computer games firms have been using market barrier tricks for years. As with printers, the business strategy is to subsidise sales of the actual consoles with sales of the cartridges (or more recently, CDs) containing the software. Sales of accessories, such as memory cards, are also controlled, and there have been lawsuits invoking the DMCA against unlicensed accessory vendors. As with printers, laws are diverging; for example, it is legal to defeat the Sony Playstation's copy protection and accessory control mechanisms in Australia, but not in Canada [11].

Up till now, vendors wanting to introduce barrier technologies to control aftermarkets typically had to design them from scratch. It is hard to get security designs right first time – especially when the designers are new to information security technology – so most early designs were easily circumvented [1]. The legislative environment is uneven and unpredictable, as the above examples show. There are often major political issues, especially in industries that are already concentrated and exposed to regulation. So there are significant risks and costs associated with these barrier technologies, and they are by no means ubiquitous.

That may be about to change dramatically. The introduction of so-called 'trusted computing' will make it straightforward for all sorts of vendors to tie products to each other, to lock applications and data on different platforms, and to tie down licences for the software components of systems to particular machines. This is likely to usher in a significant change in the way in which many of the information goods and services industries do business, and may spill over into may traditional industries too. First, we need a brief overview of 'trusted computing'. (For more detail, see the Trusted Computing FAQ at [2].)

2. TRUSTED COMPUTING

In June 2002, Microsoft announced Palladium, a version of Windows implementing 'trusted computing' and due for release in 2004. In this context, 'trusted' means that software running on a PC can be trusted by third parties, who can verify that a program running on a machine with which they are communicating has not been modified by the machine's owner. Programs will also be able to communicate securely with each other, and with their authors. This opens up a number of interesting new possibilities.

The obvious application is digital rights management (DRM): Disney will be able to sell you DVDs that will decrypt and run on a Palladium platform, but which you won't be able to copy. The music industry will be able to sell you music downloads that you won't be able to swap. They will be able to sell you CDs that you'll only be able to play three times, or only on your birthday. This will be controversial; other applications will be less so. For example, trusted computing platforms can host games where cheating is much harder, or auction clients which can be trusted to follow a set of agreed rules – which will make it significantly easier to design many types of auction [13].

Palladium built on the work of the Trusted Computing Platform Alliance (TCPA) which included Microsoft, Intel, IBM and HP as founder members. The TCPA specification, version 1.0, was published in 2000, but attracted little attention at the time. Palladium was claimed to use TCPA version 1.1 which supports some extra hardware features, and the next generation of Pentium processors from Intel (the 'LaGrande' series), which offer an extra memory protection mode: the idea is that since many existing untrusted applications run with administrator privilege, that is in ring 0 of the processor, upgrading security without replacing all these applications requires yet another protected memory mode, called 'curtained memory', so that small parts of trusted software can run with extra privilege that gives them access to cryptographic keys. TCPA has recently been formally incorporated and relaunched as the 'Trusted Computing Group' [14].

The TCPA/TCG specifications set out the interface between the hardware security component (the 'Fritz chip'), which monitors what software and hardware are running on a machine, and the rest of the system, which includes the higher layers of software and the means by which the Fritz chips in different machines communicate with each other. Fritz's role in the 'trusted' ecology is to assure third parties that your machine is the machine you claim it to be, and that it is running the software that you claim it to be.

2.1 Terminology

There is some difficulty in finding a suitable name for the subject matter of this paper. Neither 'TCPA' nor 'Palladium' will really do. For a while, when public criticism of TCPA built up, Microsoft pretended that Palladium and TCPA had nothing to do with each other; this pretence was then abandoned. But as criticism of Palladium has increased in turn, Microsoft renamed it NGSCB, for 'Next Generation Secure Computing Base' [15]. Presumably this isn't the final name, and in any case it's a bit of a mouthful. We might refer to the project as 'trusted computing' but that has evoked principled opposition; Richard Stallman, for example, prefers 'treacherous computing' as the real purpose of the technology is to remove effective control of a PC from