# Avraham Bernstein CV

Last update: 2017-06-29T11:57:00Z

Copyright © 2017 Avraham Bernstein, Jerusalem ISRAEL. All rights reserved.

License: Except where otherwise noted, this work is licensed under the Creative Commons License [CC BY-ND 4.0](https://creativecommons.org/licenses/by-nd/4.0/).

## 1.0 Contact Information & Links

**email/skype/pgp:** [Avraham.Bernstein@gmail.com](mailto:avraham.bernstein@gmail.com)

**pgp-id:** [E6647D2F](https://pgp.mit.edu/pks/lookup?op=vindex&search=0x86EFCDAEE6647D2F)

**location:** Jerusalem ISRAEL, **tz:** UTC +2:00/+3:00 [winter/summer]

**tel:IL:mobile/whatsapp:** +972.54.641-0955

**tel:US:mobile:** +1.845.402-0023

**www-home:** <http://purl.org/avraham.bernstein>

**www-linkedin:** <https://www.linkedin.com/in/avrahambernstein>

**cv:** <http://purl.org/avraham.bernstein/cv> [this file]

## 2.0 Overview

I am a computer scientist. I am a master S/W engineer and architect of industrial strength systems. After 30+ years I still feel it is the most creative profession that I could have chosen. I work at executive level. I am very innovative and creative. I have a strong background in economics. I communicate well both orally and in writing.

For the past 13 years, I worked primarily in the field of cybersecurity with a specialty in S/W obfuscation, i.e. anti-reverse engineering techniques. The target O/S were Android and IOS.

I have also worked in a broad range of other application domains where I have quickly come up to speed, and functioned at an expert level - because I am self-learner, and I know how to find and utilize mentors who are experts in the new field, and who also recognize my potential. (I am also a good mentor myself). These applications include bioinformatics, transportation, blind vision, VLSI, and factory automation. Just a few years ago, after only a month in the new (for me) field of bioinformatics, I developed a revolutionary new PCR Ct prediction algorithm. I was very fortunate that my mentor was the one who developed the original algorithm - which is the most cited in this field in Google Scholar.

Invariably I am able to apply knowledge, or "meta-knowledge", that I learned in previous fields, and apply it to the new one. A common technique that I use is to create a domain specific language (DSL) that allows me to write a formal specification of a problem. With a little ingenuity the DSL can be used to create a working command line application, and it usually forms the basis of an automated testing framework. I successfully used DSLs to develop a shop floor production control system, and I used it to describe CPU architectures.

A common programming technique that I use is meta-programming with templates that allows me to automatically generate code by creating intelligent macros regardless of the underlying target programming language.

## 3.0 Work Experience

### June 2017-present: Consultant: Cybersecurity

Developing a business model to work with small businesses that are developing apps in order to protect their apps against reverse engineering - whether or not the app is written in native code or generically in HTML and javascript.

### 2011-May 2017: Security Manager & Architect: OTT Internet Pay TV: Secure Player, @Viaccess-Orca & Sansa Security (formerly Discretix)

During the 6 years of my tenure as head of security for this OTT (over-the-top) Internet pay TV product, we had tens of millions of subscribers, while we were never successfully hacked in the field. The product was licensed for small screen devices, i.e. Android and IOS smart phones and tablets.

My job was to prevent the theft of intellectual property (aka anti-piracy), namely to protect the DRM app that we loaded into subscriber devices against reverse engineering.

Most important, I had to protect the secret keys that were stored on the subscriber devices from leaking, i.e. the subscriber keys and the content decryption keys. The core security module that we loaded into the subscriber devices was a binary (native machine code) library written in C/C++, where just the GUI wrapper was written in Java and Objective-C. Note that it is 10-100 times more difficult to reverse engineer C/C++ code compared to Java or Objective-C code. I used state-of-the-art binary software obfuscation techniques in order to outsmart state-of-the-art reverse engineering assembly code debuggers, i.e. IDA Hex-Rays.

I developed many of the security algorithms including the Android root detection algorithm which needs to be implemented using fuzzy logic.

Our major security tool vendor was Intertrust / WhiteCryption who supplied us with their SCP obfuscating compiler, and their SKB "whitebox" cryptography SDK. I worked very closely with the vendor to improve their product, because I was previously involved in the specification of a LLVM obfuscating compiler at NDS.

From a managerial perspective there were critical economic decisions that had to regularly be made. Overly aggressive piracy detection could generate thousands of customer support calls from bona fide subscribers where the cost of handling an individual call could be more the price of the monthly subscription fee. But weak detection algorithms could result in content piracy, or counterfeiting of subscriber credentials.

### 2016-2016: Consultant: Cybersecurity Protection of a Small Business With Extremely High Security Concerns, @Stealth

I worked with a small secretive commercial company, with approximately 15 employees, engaged in a legitimate business, whose web site was under constant attack, and whose employees could potentially face physical harm if they were "outed". The majority of the employees were researchers who had little knowledge of cybersecurity. Even the straightforward/naive way that they accessed the Internet could leave tracks that could expose them provided that their adversary were a well financed technically adept opponent - perhaps with foreign government backing. They used a WordPress web site on a private server *without* two factor authentication protection (2FA). A lot of employees worked from home, or from public Internet hot spots. They all used gmail, and most *without* 2FA. And even those who did use 2FA, used their phone number - so if their phone were lost or compromised then the extra factor wasn't very useful. Their network file system was Google Drive for business. They used Windows laptops. Most had Android phones. None of their devices were configured securely. A single compromised device, or the theft of loss of a single device, could lead to complete penetration of their network. The picture could not be much more bleak. And in order to minimize organizational resistance to change, their road to security recovery had to be a long step-by-step process.

### 2010-2010: VP R&D: Urban Traffic Route Guidance, @Telequest.co.il

TeleQuest was a startup engaged in the development of a coordinated dynamic route guidance system, in order to alleviate traffic congestion in large metropolitan areas - similar to what Waze now does.

Simulation experiments on Tokyo traffic showed that at a guided vehicle penetration rate of 50%, TeleQuest could provide a sustained average travel time reduction of ~50% for guided vehicles and ~40% for non-guided vehicles. None of the competing commercial or academic models come close, because after penetration rates of 15-20% their benefits peak and then started to decline because their simplistic opportunistic algorithms generate their own secondary traffic jams of the guided vehicles.

My job was to architect, develop, and implement algorithms, in conjunction with a team of transportation scientists and mathematicians, that were executed in a real-time traffic simulator running on tens of cloud computers, to be coordinated with GPS equipped smartphones running our S/W on tens of thousands of vehicles.

### 2009-2009: Consultant: Bioinformatics PCR Algorithm Development, @Syntezza

Within 3 months of entering the field of bioinformatics, I discovered a new much more accurate [PCR](https://en.wikipedia.org/wiki/Polymerase_chain_reaction) (DNA amplification) Ct prediction algorithm that worked particularly well in noisy environments due to PCR inhibition. I was working for a company developing a new dry [MRSA](https://en.wikipedia.org/wiki/Methicillin-resistant_Staphylococcus_aureus) detection kit, where MRSA is a deadly strain of antibiotic resistant staphylococcus bacteria that thrives in hospitals, and is the number one cause of fatal infections in hospitals. When I arrived their kit was failing. Its detection rate was only 50%, and as a result the company was facing liquidation because the investors were losing confidence. I raised their detection rate to 95+% (!) which was about 10% higher than the detection rate of their competitors - mainly from the giant multinational pharmaceutical companies. Before beginning this project, I had zero background in bioinformatics, biology, or biochemistry.

During this short period I made some important discoveries:

1. I discovered a much more powerful and much simpler algorithm to normalize data, and to reduce noise - compared to the classic Bar-Tichopad algorithm. Note that Tzachi Bar was my mentor.
2. I discovered an alternative AI (artificial intelligence) based algorithm that very accurately determined whether a sample is positive or negative (according to the Petri dish results) when the underlying data did not exhibit the flat "S" shape expected by the Bar-Tichopad algorithm.

I am in the process of patenting my algorithms.

### 2004-09: Security Research & Business Development, @Cisco-NDS

1. Background task was doing security code reviews. At a basic level this involved enforcing the use of textbook safe programming practices, e.g. protection against stack smashing, wiping memory before freeing it, validating user input, etc. At a higher level, it involved checking how well the program was protected against reverse engineering.
2. We came to the conclusion it was far too difficult for the average programmer to be concerned about both producing correct code and to protect it against reverse engineering. I was part of the specification team for an obfuscating compiler, to be implemented using the open source LLVM infrastructure.
3. I was technical lead and writer of a proposal to China CCC TV to protect their internal IP-TV broadcasts of the 2008 Beijing Olympics from being leaked outside of China. The IOC allows the host country to receive the feed for free, but the owner of the feed MSNBC received billions of dollars of royalties. Therefore if China were incapable of protecting the feed from leaking, MSNBC would cut off their internal feed.
4. I wrote business proposals to use NDS security technology to protect computer games and to protect printer ink cartridges from reverse engineering.
5. I became an expert in the use of virtual machines (VM) technology to crack DRM and cryptographic security schemes. I used QEMU to crack crypto techniques developed by the world's best commercial crypto team (at NDS) by breaking the random number generator.
6. I arranged for seminars to be given by world class hackers and security researchers.
7. I gave a presentation to senior management about the huge development costs associated with fixing bugs. Nearly 25% of their R&D manpower was devoted to fixing bugs! I discovered the extent of these costs by data mining their ClearQuest bug tracking database. And I also suggested numerous relatively easy to implement techniques that would greatly reduce the number of bugs.

### 2003-04: CTO: Blind Accessibility H/W & S/W To Enable Viewing of Digital Images & Maps, @Virtouch

1. The flagship product was the [VTPlayer tactile mouse](http://www.avrahambernstein.com/resources/vtplayer.png). It had 2 finger pads on top - each with an array of 4x4 refreshable braille pins.
2. In default mode, it converted screen pixels to an 8-bit gray scale with say a threshold of 128 to control whether a pin is up or down. By slowly moving the mouse across the screen every pixel can be felt in the cursor's path.
3. The problem was blind people are not comfortable with mice because they can't see the cursor. I added vibrating pin patterns and tonal audio cues to indicate the cursor position. However the solution was not good enough. Anyway the mouse was extremely expensive (~$350) given our low sales volume.
4. Therefore I abandoned the mouse in favor of an off-the-shelf graphics tablet and stylus combined with audio cues. A tablet computer or smart phone are also good.
5. It was a great success because the blind could inherently feel their position on the tablet. And we rectangularly mapped the tablet into the physical screen or the current window.
6. XY position audio cues were based upon 2 musical instruments, e.g. piano and flute, mapped to 8 octaves of a major scale. Each of the 56x56 quadrants (note a scale has 7 notes) produced a different note combination that they could easily discern. The 8-bit gray value of the underlying pixel was indicated by 256 levels of volume.
7. My most important conceptual breakthrough was using XML and SVG to provide semantic/contextual information, e.g. via MapML. Maps today are made of layers, e.g. roads, cities, sites of interests, topography, polical borders, etc. I provided filters to declutter the maps so only specified layers would be selected. Also the granularity was selectable too, e.g. topographic isolines of 500m. By clicking the user received contextual semantic data associated with the cursor position.
8. I was awarded a large EU FP6 matching grant to conduct further research - administered the Israeli gov Chief Scientist office - but the investors balked.

### 1999-2002: Manager Cable Network Tools S/W Development Group, @Vyyo

1. I managed a team of 3-6 programmers.
2. I designed a NMS and MIB for cable modems and routers, and I designed the associated embedded SNMP agents.
3. I designed a hybrid IP connection for cable modems where there was no physical cable upstream channel. Instead the upstream channel used a telephone modem (ATA), while the downstream channel used the cable modem. Head end network equipment for both interfaces was supplied by the cable operator. My solution was to dynamically modify the arp table of the edge router. For typical surfing, the effective downstream rate was as fast as a pure cable solution. The company applied for a patent.
4. I greatly improved the efficiency of the laboratory modem speed stress testing by a factor of 10-100 by using a "steepest descent" search algorithm instead of a binary search algorithm. Reduced testing time per modem from hours to minutes.
5. Designed a virtual testing lab with 64K modems and 512K PCs via multiplexing the physical connections. The test lab had only 256 physical cable modems, 4 physical PCs with 8 network connections each, 1 cable router, and 2 24-port layer-2 programmable switches. By dynamically editing the PC MAC addresses, and by dynamically editing the MAC filters on the network switches, I was able to stress test the router into believing that it faced 64K modems, and the modems into believing that each one was shared by 16 PCs.

### 2000-2000: Consultant: Network Management System for [FSO](https://en.wikipedia.org/wiki/Free-space_optical_communication) Devices, @MRV / JOLT

### 1999-1999: Consultant: Win32 Network Driver Software, @Inex/Zamir

### 1998-1998: Consultant: GCC Compiler Port for a 128-Core Stack Machine, @Fourfold

### 1996-1997: Consultant: S/W Architect & Implementation of Conoscopic Interferometer Workstation, @Optimet

The workstation S/W was implemented in TCL. TCL verbs were added to control the various hardware components, i.e. the probe, the XY jig, the video camera, etc. By using TCL, the workstation could be controlled via a GUI, or by scripts that allowed automation in a factory environment.

The most challenging part of the project was providing the tools for interpreting the results - especially when an *ad hoc* object was viewed.

The result of a raw scan produced millions of XYZ points. My task was to provide 3D graphic tools to give the results semantic meaning, i.e. to give "life" to the object.

The core 3D graphic S/W package that I used was OpenGL. I designed a [domain specific language (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language) wrapper that allowed a researcher to manipulate the millions of facets. An especially important technique was to assign different colors to regions according to filtering criteria, e.g. like a topographic map. And since TCL does not efficiently deal with huge arrays, I integrated TCL math extensions that could efficiently manipulate huge arrays.

The project was a major success.

### 1994-95: CEO & CTO: S/W Architect & Implementation of US DOD Mil-Spec Automated Testing System, @Pitkha Outsourcing / Elop

1. The company product was the weapon fire control system for the Night Hawk laser guided missiles.
2. My task was to create an architecture and implementation of the automated test procedures dictated by the US DOD. The test specification documentation was about 1 meter high.
3. Additionally there would be many ad hoc tests that would be required during the development process.
4. The test equipment included external voltage regulators, external heating and cooling equipment, vibrators, etc.
5. The workstation controlling the tests was a Windows NT PC.
6. Instead of creating a monolithic test program in C/C++, I created a BASIC-like [domain specific language (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language) with special drivers for controlling the various pieces of H/W.
7. The test specification manual was being regularly revised, plus I needed the flexibility to allow the system engineers to write their own scripts without delving into the C code. And I wanted to avoid having the client call me every time he needed to implement a minor/trivial change that he could easily learn to do himself.
8. The major tests were presented via a GUI which in fact emitted DSL script.
9. The system worked as planned.
10. Unsupervised tests ran successfully for up to 72 hours (over holidays).
11. Post mortem: The first version of TCL was released around the time of this project *pre -Internet*. Once I learned about TCL, I realized it would have been the ideal platform for creating the test environment. Instead of me having to create my own BASIC-like language with control structures and variable handling, I could have relied upon TCL, and simply added custom primitives for the various pieces of H/W.

### 1991-94: CEO & CTO: S/W Architect & Implementation of S/W Toolchain For DSPG PINE CPU, @Pitkha Outsourcing / DSPG

I was the software architect of a clock accurate DSP CPU simulator along with a complete software development toolchain, i.e. a debugger, C compiler, assembler and linker. Note that the system was developed just before the GNU Compiler Collection framework reached maturity (i.e. v2.95).

This system enabled complete working applications to be developed *before* the chip became physically available. This system reduced the client's time-to-market by 6-12 months. My technological breakthrough was my design of a [domain specific language (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language) that described the CPU architecture which automatically generated the source code for the toolchain that enabled the system to be automatically rebuilt within an hour in the face of almost daily changes to the VLSI architecture - especially the pipeline.

Another important application for this toolchain was to compare the results of assembly code test programs used to verify the VHDL VLSI specification against the same test code run on the simulator. The simulator executed tests for up to 72 hours (over holidays) without any failures.

### 1989-90: S/W Architect: Shop Floor Production Control (SFPC) System: BARI II, @DEC / Iscar

1. Iskar Matkash in Tefen IL is a fully automated factory that produces thousands of different cutting blades using a sintering process. The raw materials go through many stages of operations. In many cases after undergoing intermediate processing, the partially processed material can still be diverted to multiple final products - similar to stem cells. The factory contains hundreds of automated workstations, stands, stacks, guided vehicles, and conveyor belts. Pallets can be automatically move from one stand on a workstation to the next, or to temporarily place a pallet on a storage stand or a stack stand.
2. My task was to create a computer program that automatically operated/orchestrated the factory.
3. When my co-architect and I started this project we had zero background in industrial engineering. We were supplied with a mentor who brought us up to speed.
4. Eventually after months of discussions we created an architecture that was a textbook object oriented taxonomy - a "factory object kingdom". The top level object was a "production instruction".
5. We defined the attributes and methods associated with each object.
6. We created a descriptive, i.e. *non-procedural*, [domain specific language (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language) that was designed to be used by the factory engineer.
7. I wrote the language manual.
8. We used the language to configure the factory. We created a GUI interface wrapper for the language which emitted CLI script.
9. We mapped the language to a relational database.
10. We created an implementation architecture in Pascal. Given the inherent OO nature of the architecture, C++ would have been a better implementation language choice - but the project management refused.
11. After 18 calendar months, and 4 man-years later, the factory ran perfectly!

### 1988-88: S/W Architect & Implementation: Quadriplegic PC Accessibility, @Cubital

I designed and implemented a system that enabled quadriplegics to access computers with CRT video displays *before the era of inexpensive voice recognition technology, tablets, and smartphones*.

The hardware we used for this project was a standard light-pen (an obsolete point-and-select input device that determines its position via the CRT's scan point) combined with custom telescopic optics that extended its normal maximum range to the screen from 10 mm to 800 mm, which we attached to the user's head with a headband, along with a standard accessibility sip-and-puff switch.

The software had 2 modes of operation. In learning mode, a virtual keyboard covered about 80% of the screen. (With 6/6 hindsight, I should have patented the virtual keyboard concept !). When the light-pen was aimed at a key then it would be illuminated. Puffing on an illuminated key caused the virtual key stroke to be entered into the operating system. In normal mode, the virtual keyboard was no longer visible. Instead a single key would pop up at the position on the underlying hidden virtual keyboard where the light pen was aimed, leaving 95% of the screen real estate available for the "real" application.

This system enabled our first client, Shulamit Gabai, a former school teacher who was stricken with polio in all of her limbs, to type 30 characters per minute, and to become a book editor for a major publishing house.

I implemented the system in the Prolog programming language, just in order to learn the language.

### 1987-88: Senior Programmer: 3D Printer: Solider, @Cubital

### 1986-1986: Programmer: Automated Leather Sewing Workstation, @Orisol

### 1985-1985: Programmer & VAX/VMS Sysadmin: Hebrew/English Word Processor: Glyph, @John Bryce

### 1983-84: Real-time Programmer: 1. Data Collection System, and 2. Radar for Lavi Fighter Plane, @DSI / Elta

I wrote the real-time O/S kernel for the 8080 data collection terminal. At that time, no off-the-shelf product was available.

### 1982-83: Junior Programmer & IBM CP/CMS Sysadmin, @Mitre Corp, McLean VA

Major programs were written in APL.

### 1980-82: Economist & Programmer: Simulation of Electric Utility Power Systems, @JWAA, Washington DC

I did simulations of electric power generation and transmission systems where the results were presented at Public Utility Commission rate hearings. Relied both on my programming and economic expertise.

### 1977-78: Economist & Intervenor at ECAP'77 Ontario Hydro Costing and Pricing Hearings, @Ontario Energy Board, Toronto Canada

I presented Ontario Hydro's marginal cost pricing (aka peak load or time-of-day pricing) case which the government of Ontario forced them to withdraw due to political pressure from the metal refining industry which at that time was using electric blast furnaces. I received extensive "secret" support from the disgruntled economists who wrote the submission. Even though the Energy Board eventually refused to accept the principle of marginal cost pricing, I was the first public interest "intervenor" to ever be awarded costs. And I published an op-ed describing the issues at the hearing in the Globe and Mail - Canada's newspaper of record.

## 4.0 Formal Education

### 1978-79: York University, Toronto Canada

MA Economics. Minor: Applied mathematics, and engineering.

### 1976-76: University of Toronto: Rotman School of Management (MBA Program)

No degree. Credits applied to York University (above).

### 1973-76: University of Toronto: Faculty of Arts & Sciences

BA. Major: Economics and mathematics.

## 5.0 Spoken Languages

English (5/5). Hebrew (4/5). French (2/5).

## 6.0 Programming Languages

Language knowledge in order of expertise, based upon my current frequency of usage:

1. C, Tcl, bash + Posix text utilities, e.g. awk, sed, etc.
2. C++, Python, make, markdown, pandoc, HTML5, CSS, Jinja2 macro/template preprocessor
3. flex, bison, javascript, Java, yaml, json
4. Forth, Prolog, Lisp, Fortran, APL, SVG, XML Schema, Relax NG, XSLT

Note that I write compilers and domain specific languages, so learning a new language takes me only a few days.

O/S knowledge in order of expertise, based upon my current frequency of usage:

1. Linux
2. Android
3. IOS
4. Win32