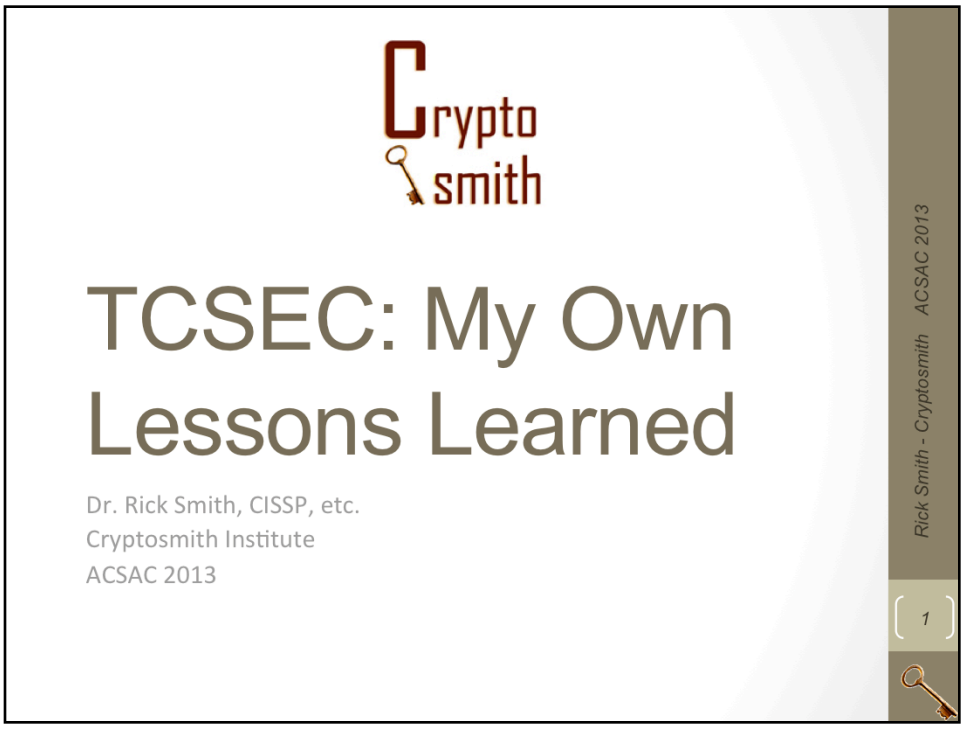
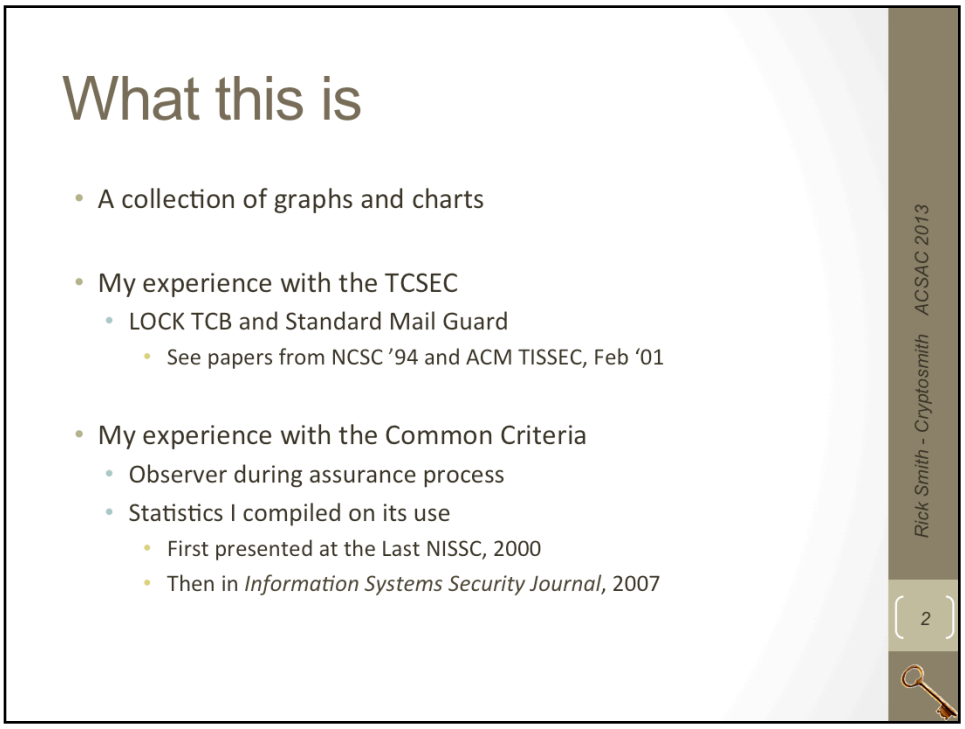


These slides and notes have also been posted on my own web site at cryptosmith.com.

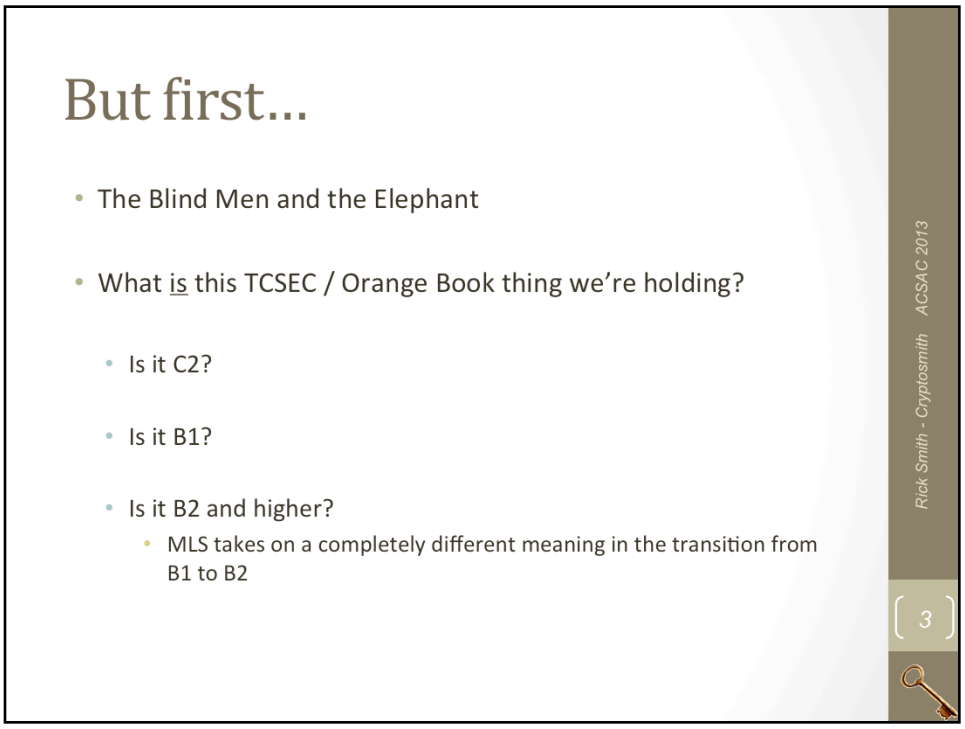


As I said by way of introduction, I'm a relative latecomer to the TCSEC community, having joined the LOCK TCB project in the early 1990s.

Earlier parts of my career introduced me to fault tolerance while writing software for an ARPANET router – we called them IMPs – running on a multiprocessor. In grad school my thesis was on fault tolerance in robotics. Then I worked on a DARPA robotics program, which dried up when Army robotics money dried up. Then I joined Secure Computing.

I worked on LOCK and on the Standard Mail Guard. After those efforts I collected statistics about the costs of high assurance development processes. Later, I provided peripheral support to a Common Criteria evaluation. Part of my work was to count the number of evaluations and look for trends, particularly in the firewall community. I continued that work in about 2007 when I published a lot of cumulative statistics about government security evaluations.

Before I talk about the numbers, let me talk a bit about the TCSEC and multilevel security.

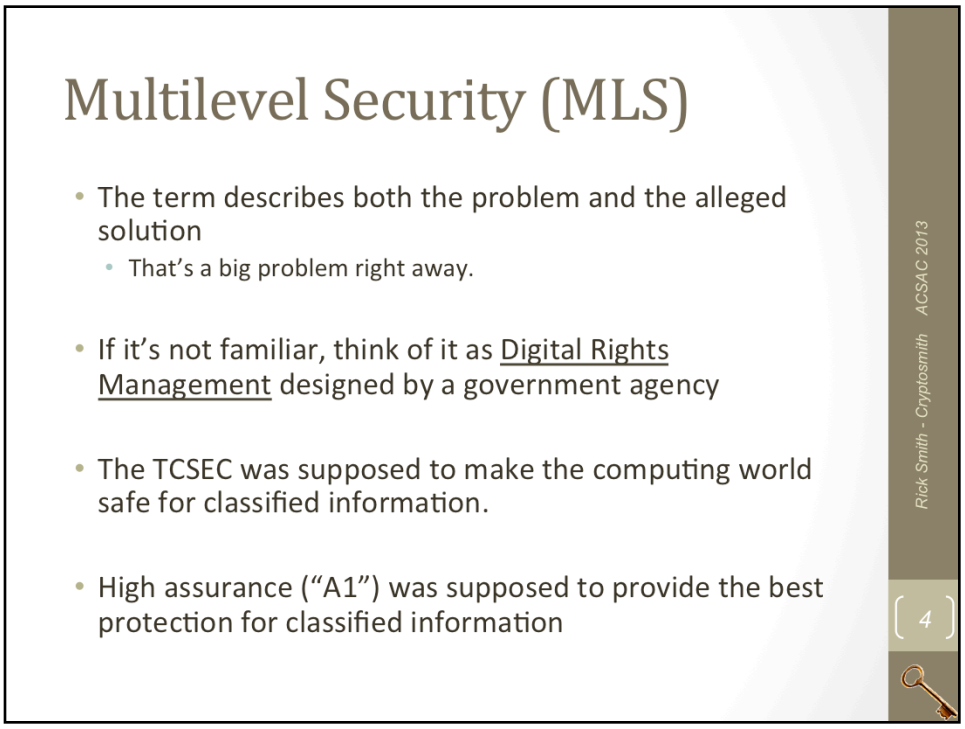


[this slide was not in the presentation, but I spoke from it]

In the C2 world, the TCSEC is mostly about functionality with minimal concern about assurance

In the B1 world, we introduce mandatory access control, but still limit our concern about assurance.

At B2 and above, assurance becomes increasingly important and increasingly challenging.

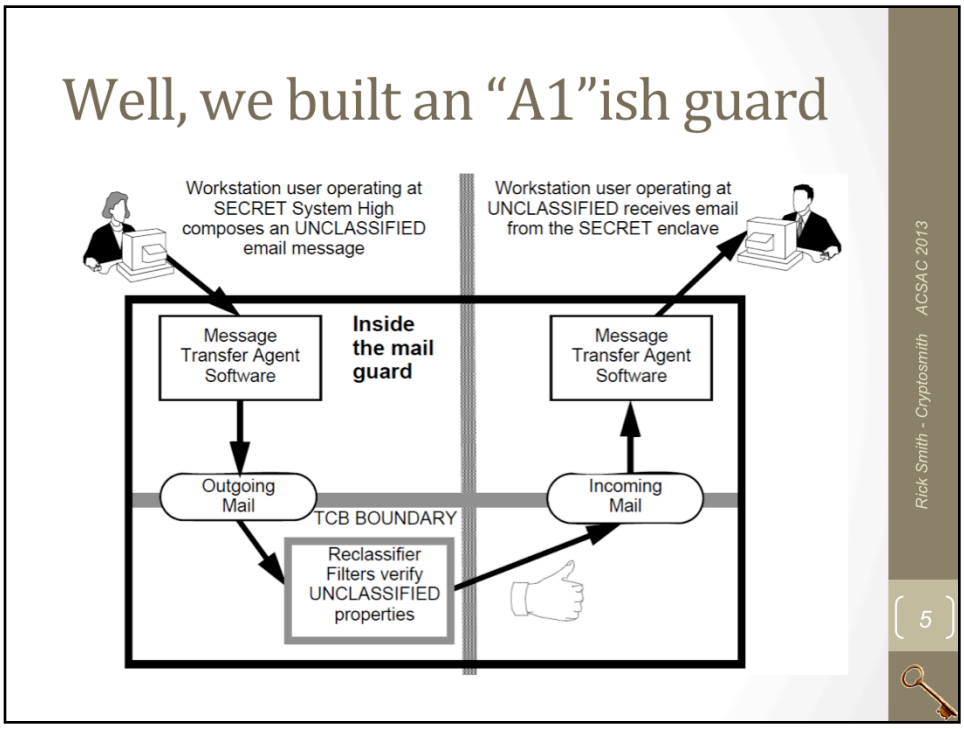


[this slide was not in the presentation, but I spoke from it]

MLS is not the only form of mandatory access control. Secure Computing tried to apply mandatory access control to firewall implementation, and other vendors (notably Cyberguard) used MLS in some of their firewall implementations. I spent a lot of time discussing the merits of such designs with other security experts in the mid 1990s. This taught me the following:

1. It is very hard to reason about MLS behavior. This may be a problem with access control in general, but it's particularly acute with MLS. It took me 6 months of work on the LOCK project before I was confident in my understanding, but I could still find myself tricked.

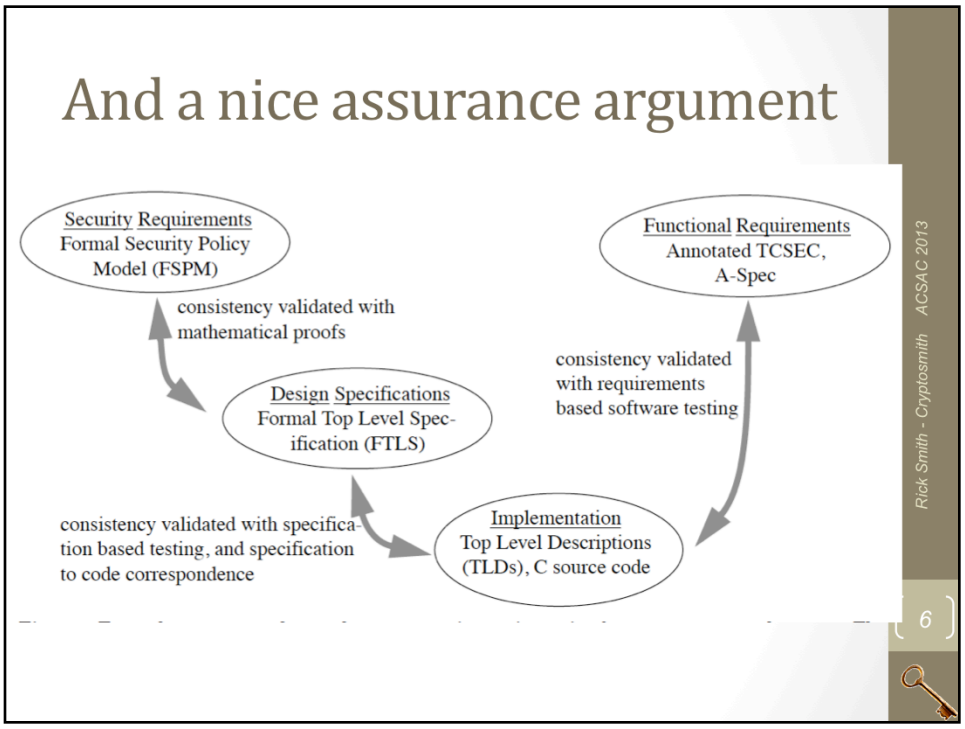
2. The semantics of MLS can change dramatically as you move between platforms, and particularly when you move between assurance levels. While it's true that there is a mathematical definition of MLS, the implementation details allow a broad range of specific and unexpected behaviors. If you relax assurance requirements, the semantics become really difficult to pin down.



This was called the "Standard Mail Guard" and I'm told that it lurked in military command centers as recently as 6 years ago. Heaven knows there may still be some out there.

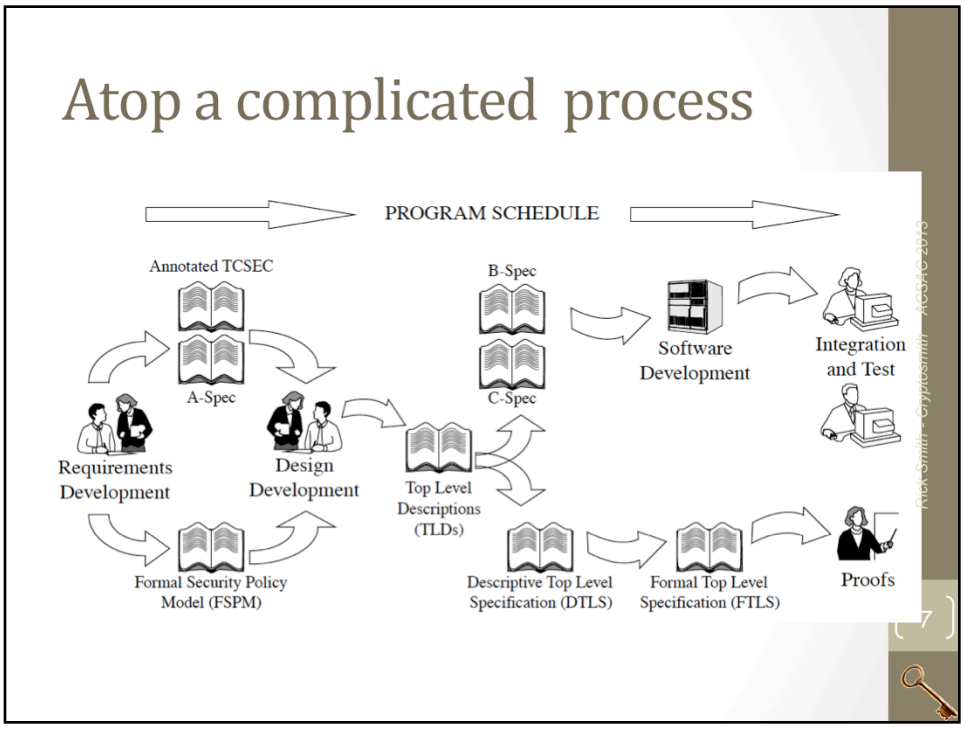
It transferred Internet-style email between classified and not-so-classified networks. This was an "MLS application" since we processed information that traversed between domains with different security classifications.

It was supposed to meet all requirements of the TCSEC A1 evaluation, but it never quite made it through the process. We expected the NSA to pay for it, because it was their project, but they weren't institutionally prepared to do that when the time came.



But the NSA gleefully adopted this machine because they bought into our assurance argument.

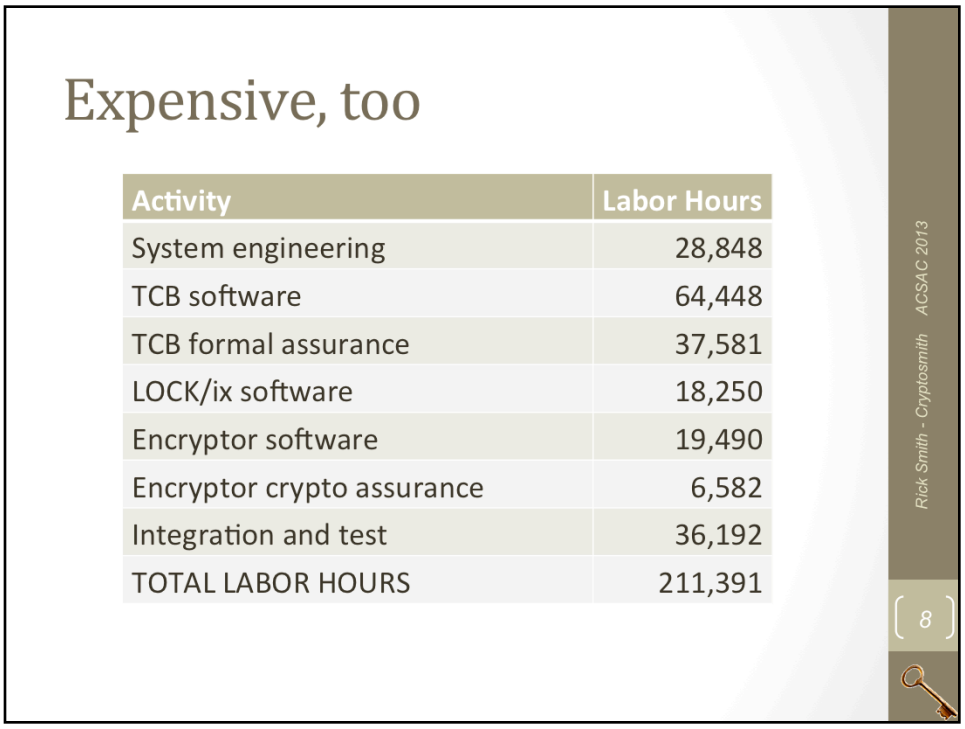
It looks terrific on paper.



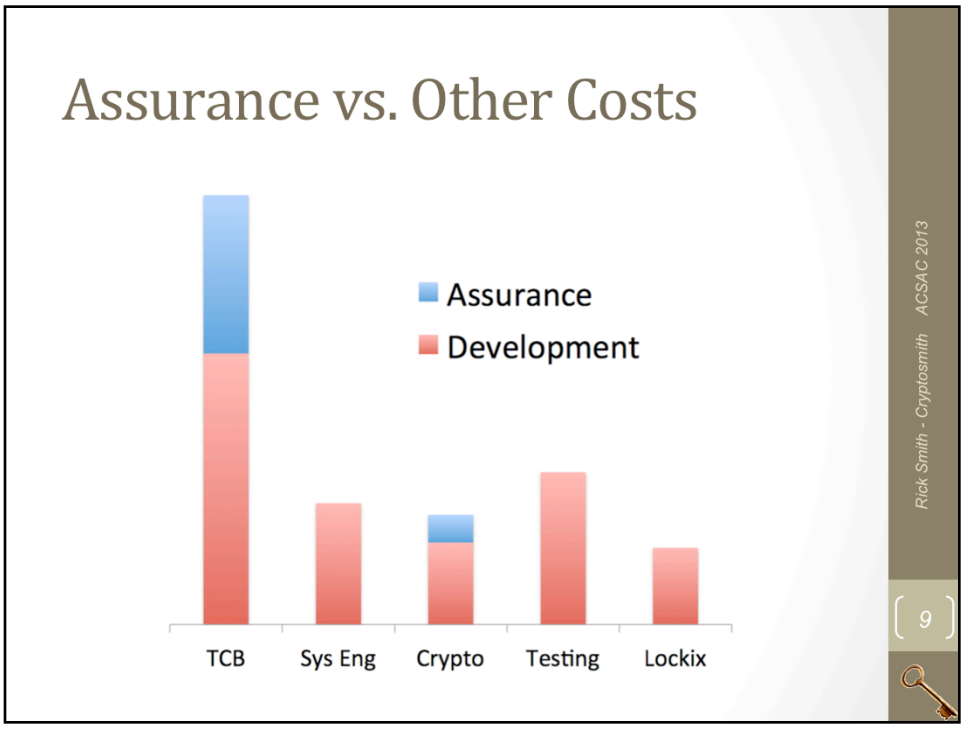
Though of course it translates into a lot of writing, testing, and documentation thereof.

This was supposed to be a classic TCSEC A1 process combined more-or-less elegantly with the DOD's STD-2167A software development (documentation) process. While the process is mostly a waterfall, there were a lot of back-flows. The consistency checks described in the previous slide would often uncover flaws in specifications, designs, or running software. We fed those back into the process and updated all affected documents. Then we repeated analyses to ensure that nothing broke.

We did our proofs, passed our tests, and ultimately shipped a product, but we never finished a 100% complete A1 evaluation. The software didn't port easily to newer hardware and the NSA didn't want to repeat the thing with more portable software. I'd say that they did SELinux instead, and flushed formal assurance down the toilet.



Now, let's face it. A high assurance process costs money. Here are the numbers for the LOCK TCB. 211 thousand labor hours with a team of lower-cost midwestern software developers. At least, given our housing prices we should have been lower cost.

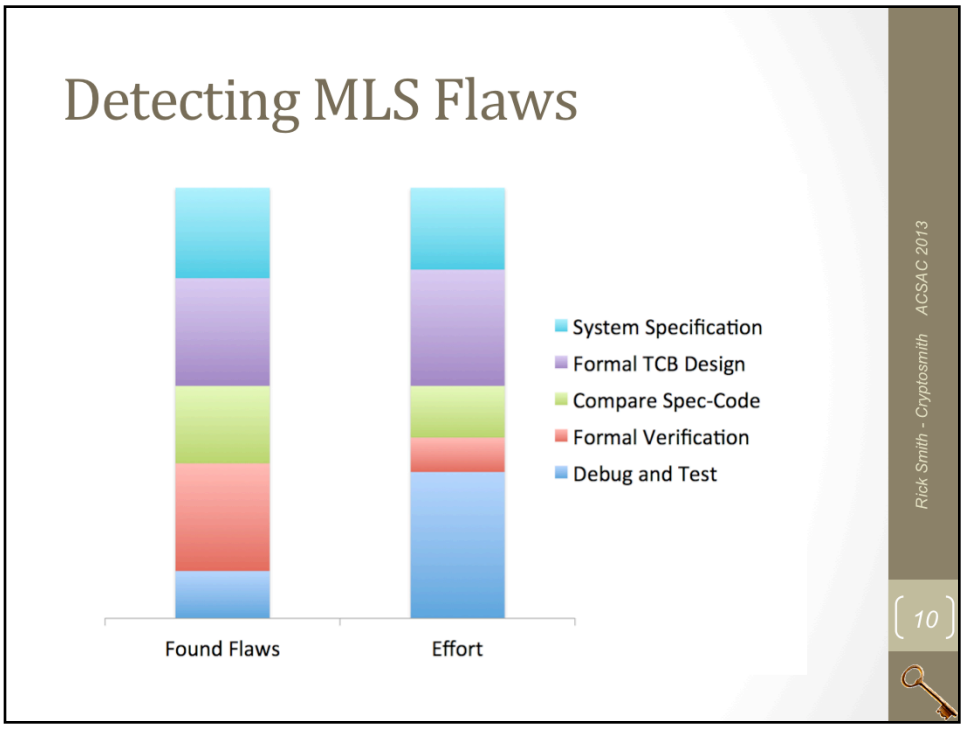


Here's a visual comparison of costs. The TCB numbers try to catch the difference between writing operating system software and writing the formal lemmas and proofs that would go along with such a thing.

The crypto numbers capture the development cost and the cost of Type 1 crypto endorsement using a pre-packaged crypto module.

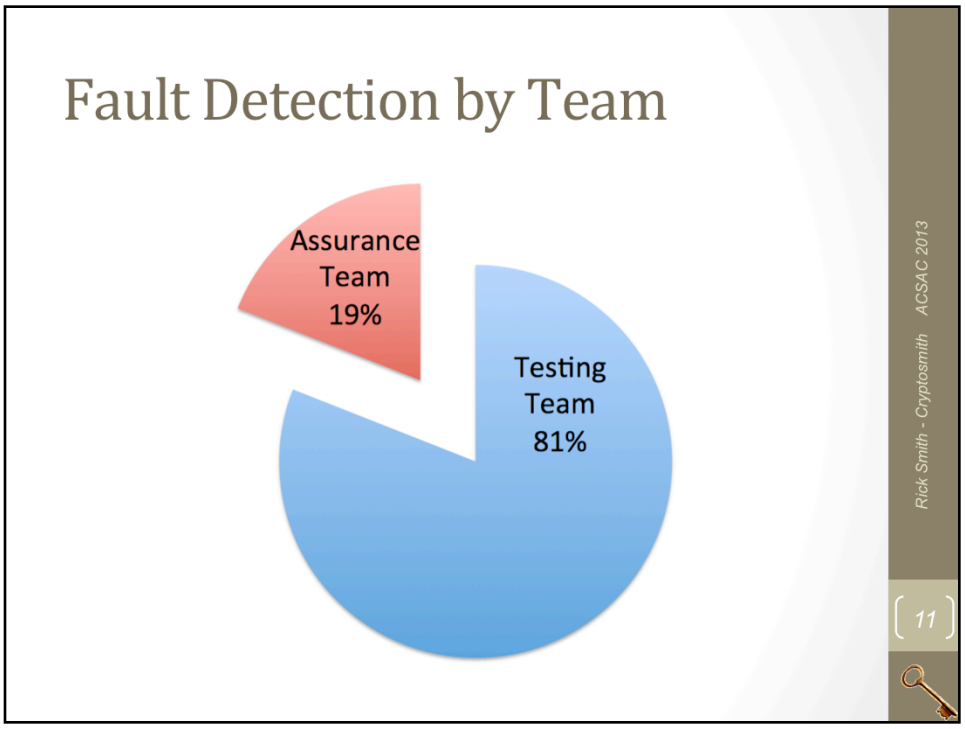
You'll notice that it takes a lot of work to do formal assurance on a working TCB.

The system engineering costs include 2167A software development documentation. Testing refers to a typical requirements-oriented testing program following 2167A documentation. "Lockix" refers to a Unix emulation package that we hosted atop the TCB so that end users would see a more-or-less familiar Unix programming and user interface.



The left-hand bar reflects every MLS flaw we found in the LOCK TCB, split out by the activity that detected it. The right-hand bar reflects the relative number of labor hours expended on each activity. Here is a brief description of the activities:

- System Specification – specifying system requirements in 2167A-style documents
- Formal TCB Design – software design specified in a special format we called “Top Level Descriptions” (TLDs), and then translated into the Gypsy formal specification language. The security policy was likewise specified in Gypsy.
- Compare Spec-Code – took the TLDs and compared their contents to the source code implementation to ensure that all elements in the spec were also in the code, and vice versa
- Formal Verification – Run mechanical proof checking to verify that Gypsy design specification was consistent with the Gypsy formal policy specification.
- Debug and Test – Conventional requirements- and design-oriented software debugging and testing.

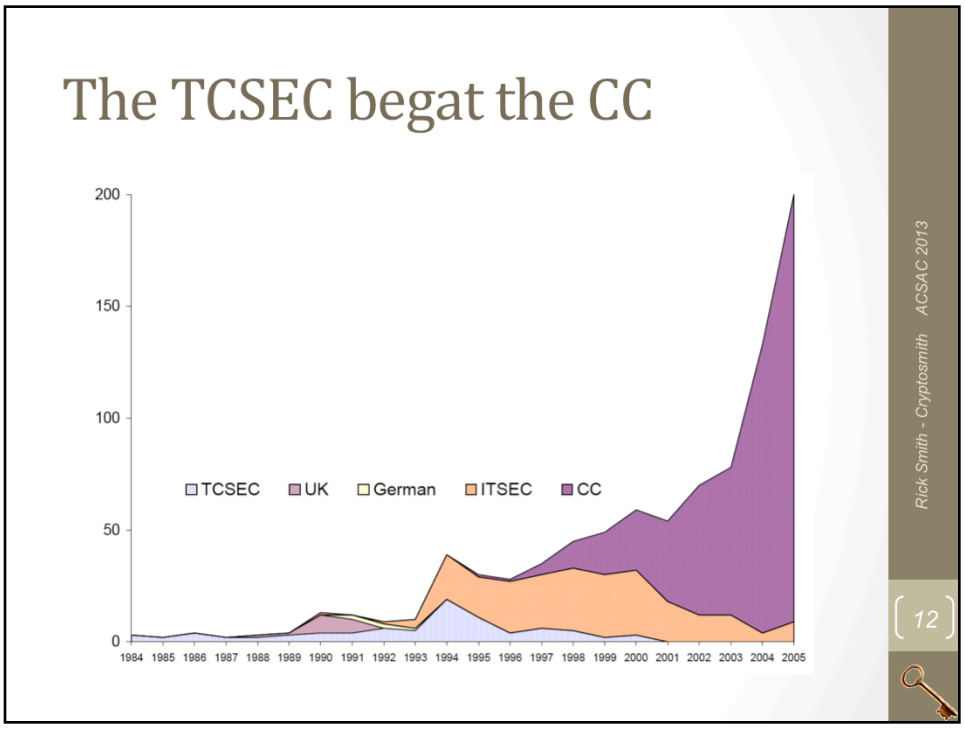


But let's not get too excited about formal assurance as a fault detection process. Like the dog taught to sniff drugs, it finds what it looks for. But it doesn't find every flaw.

This is the weakness of high assurance: practical systems typically have a lot of properties that can't be specified formally. If we focus all attention on those essential, provable properties, we end up with an unusable system.

During the Standard Mail Guard project, we implemented a guard system hosted on the LOCK TCB. This effort had to make the TCB stable and also implement appropriate application software for the guard function. This diagram presents the result of a 90-day study of flaw detection that compared the number found by the the testing team to the number found by the assurance team. Naturally the testing team found the most flaws by a large margin.

REMEMBER: This chart measures different things from the previous chart. This chart considers ALL FLAWS in the Mail Guard. The previous chart looks at MLS FLAWS ONLY in the LOCK TCB.



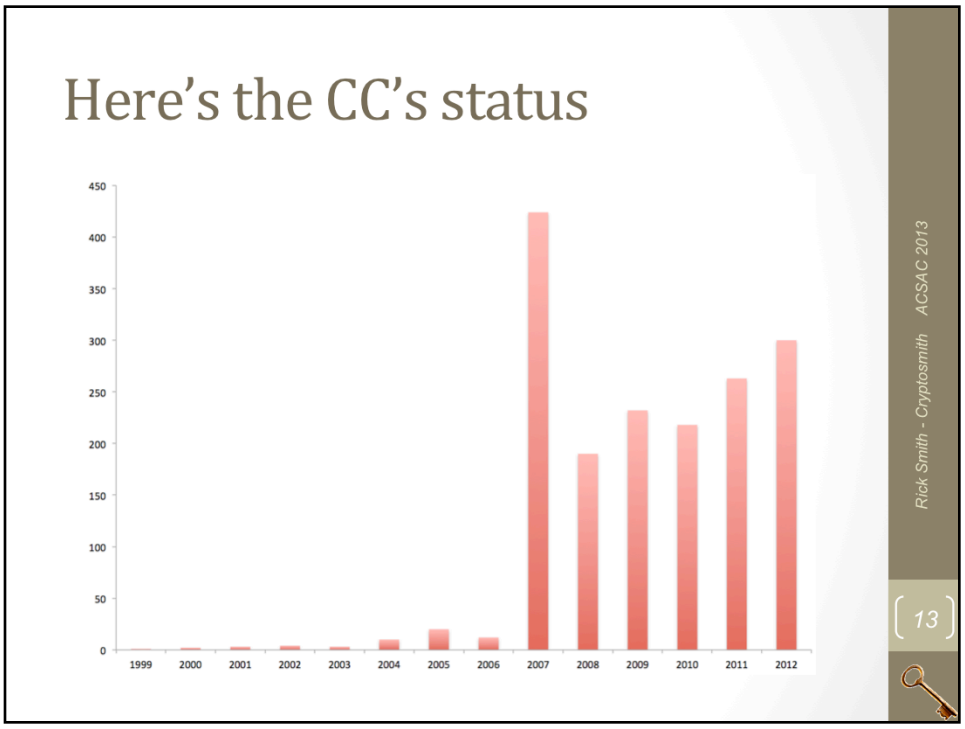
But the TCSEC is dead, so let's move on to the Common Criteria This chart shows all reported government-endorsed security product evaluations from the beginnings of the TCSEC in 1984 up to 2006.

The tiny light-blue line along the bottom shows all TCSEC evaluations.

The UK and Germany both fielded active evaluation programs, and both were folded into the ITSEC program in the early 1990s. The big orange band that smothers the TCSEC evaluations shows the growth of ITSEC evaluations in the 1990s. We'll see more about that in a minute.

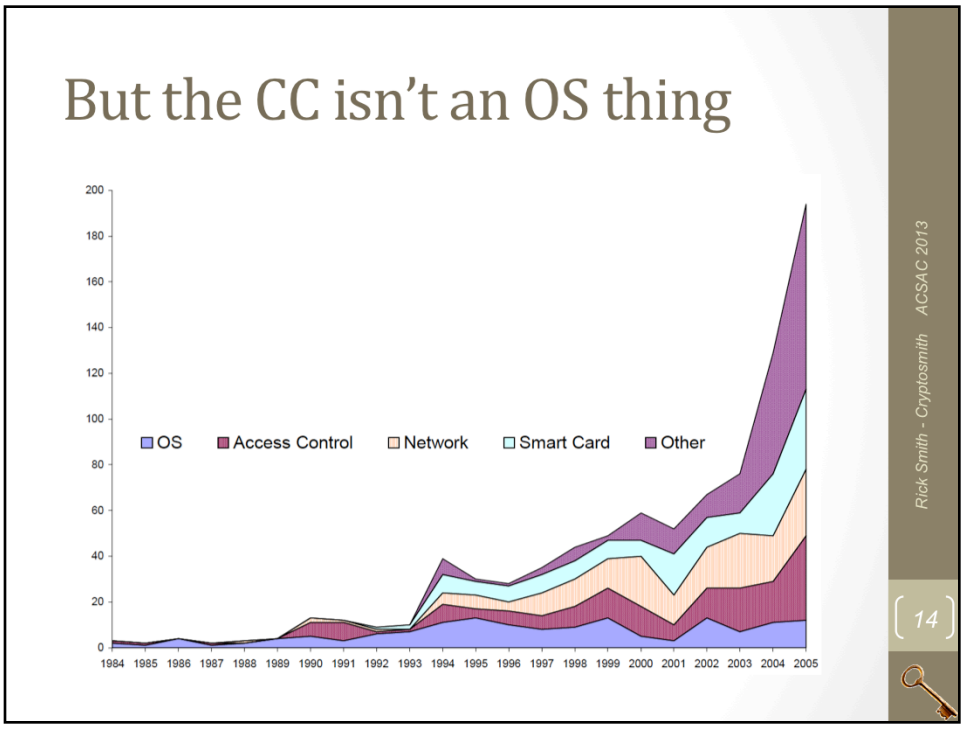
The big purple triangle on the right shows the rapid growth of Common Criteria evaluations following their introduction and acceptance by national governments.

Now, I haven't collected detailed records since 2006, but here's what I found on the Common Criteria Portal, an apparently authoritative web site.



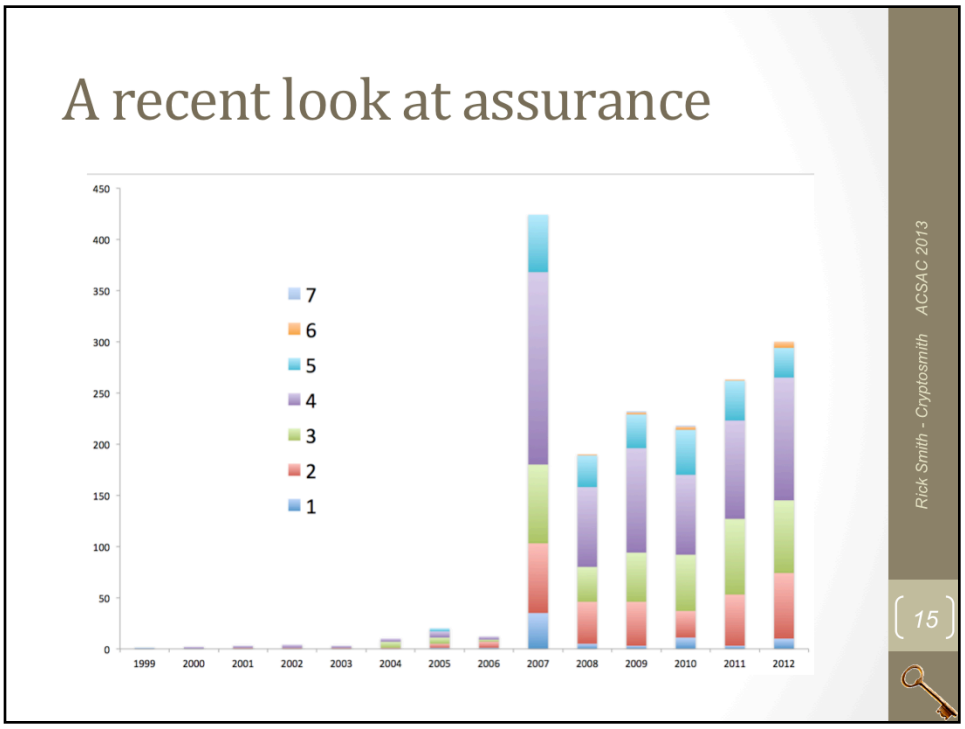
A casual examination didn't show me why there was a surge in 2007, but note that the level has been 200-300 evaluations a year since then.

So it's continuing at about the same rate as before.



Originally, the TCSEC was intended to evaluate operating systems. It was soon adapted to other, more specific software, including network devices, access control packages, and database managers.

As the other evaluation criteria arose, they were used to evaluate a whole range of additional products, notably smart cards and, in Europe, odometers for commercial shipping applications.

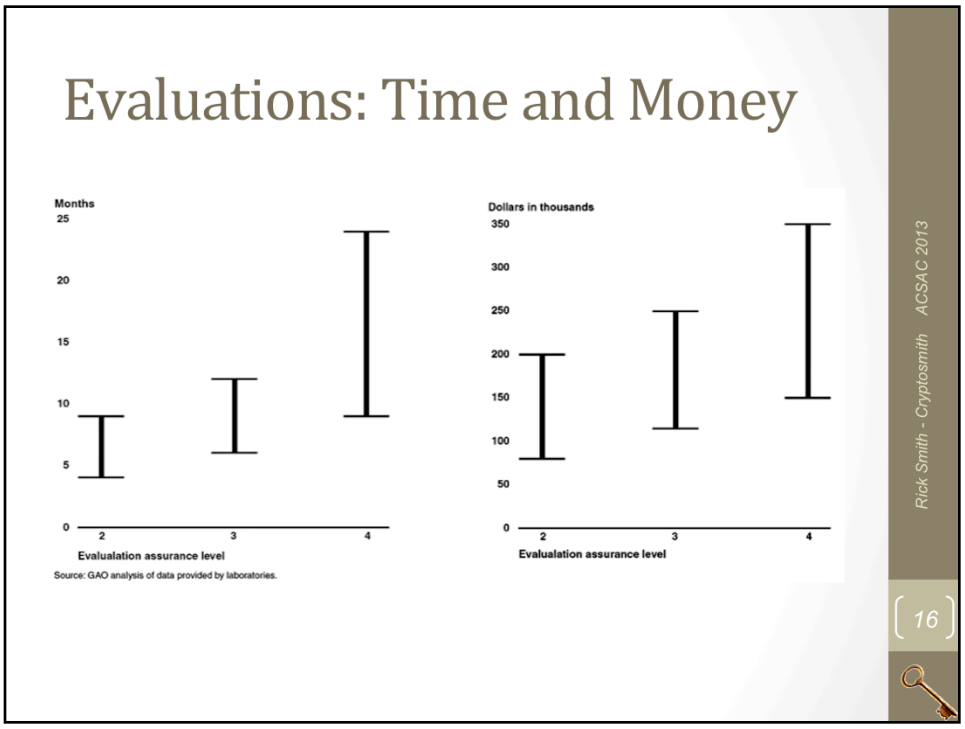


So here's a look at how assurance relates to recent Common Criteria evaluations.

The color scale goes from EAL 7 on top to EAL 1 on the bottom.

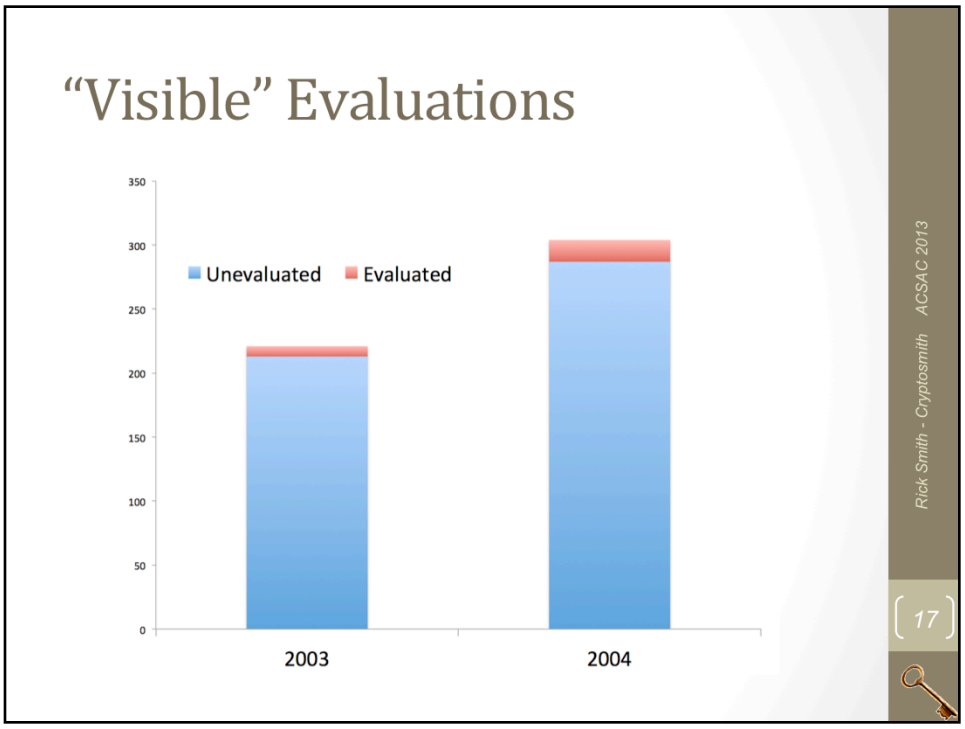
And before assurance fanatics get too excited, note that the giant purple band represents EAL 4, which some consider barely tolerable assurance.

There are a depressingly small number of higher assurance evaluations.



This graph comes from a 2006 GAO report (GAO 06-392) that looked at security evaluations as implemented by US government agencies and programs.

The “moderate assurance” EAL 4 evaluation can take “only $150,000 and 9 months” or as much as $350,000 and 25 months.



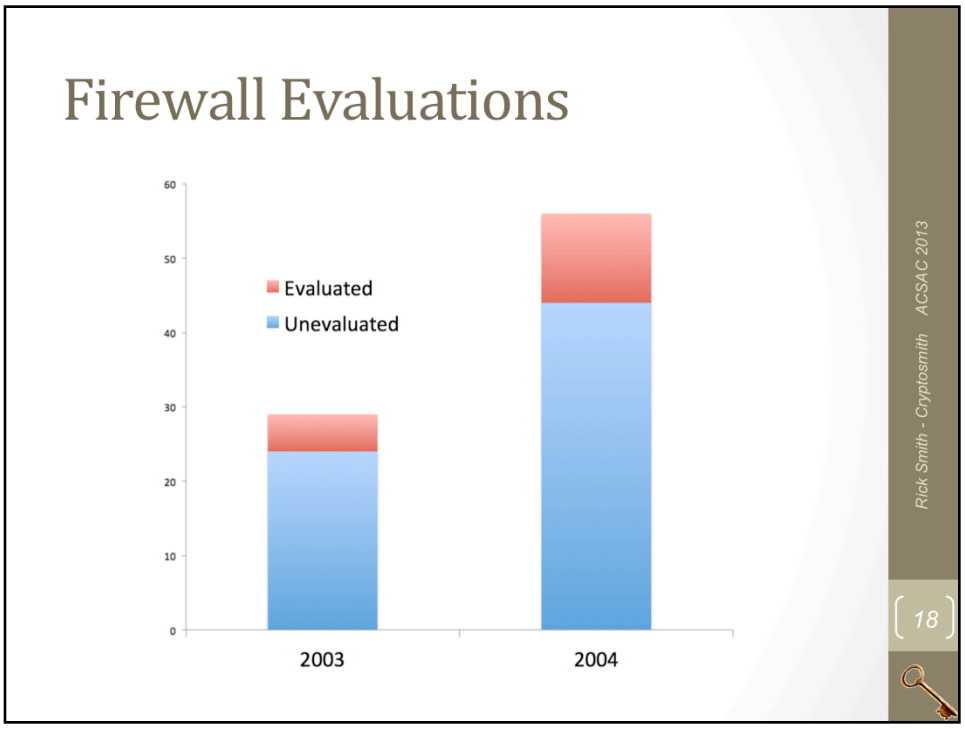
Here's an interesting question: how many "security products" are there, compared to the number that are evaluated?

I'm not sure there's a practical way to measure that, since it's hard to say what constitutes a distinct product, and at what point is it worth counting.

I used product review information from security magazines in 2003 and 2004 to try to estimate how many products actually get evaluated.

As you can see, only a few percent of security products reviewed were evaluated products – around 3% to 6%

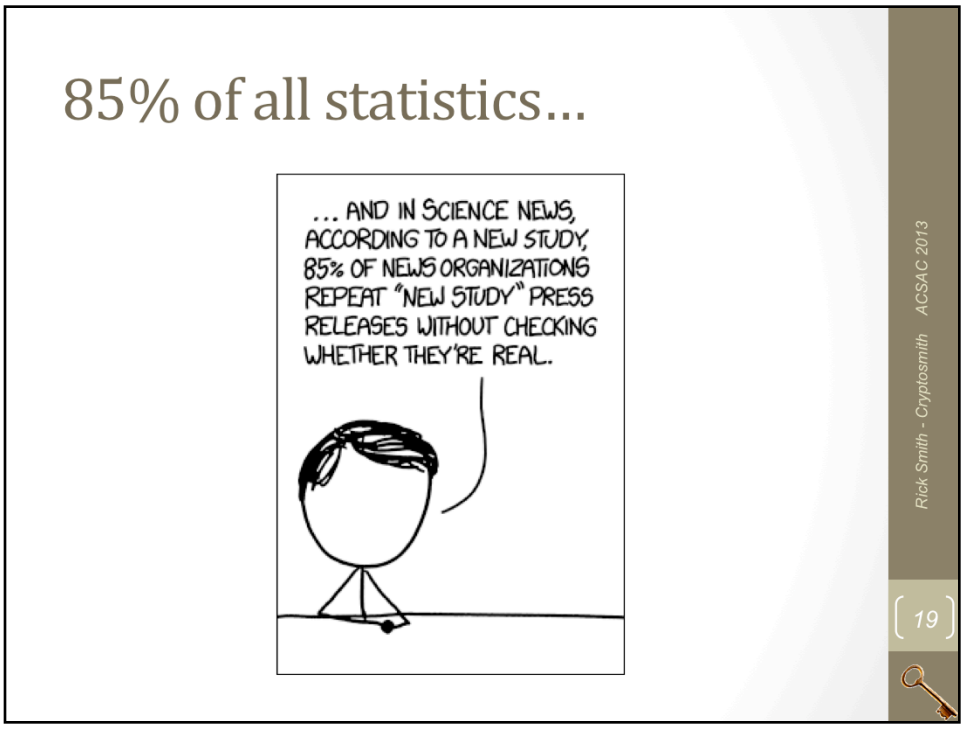
Interestingly, a lot of evaluated products were not reviewed in security magazines – maybe 10% of such products. This is probably because a lot of European evaluated products are very specialized and wouldn't appeal to a broad IT market.



Things seemed a bit clearer in the firewall community, at least in '03 and '04. There were between a dozen and two dozen firewalls evaluated each year. It seems that between a third and a fifth of the evaluated firewalls were actually reviewed in a security magazine.

Thus we have something closer to 20% of all firewall products being evaluated.

While this suggests that a lot of firewall vendors have bought into getting evaluations, it indicates that the majority were still avoiding evaluations.

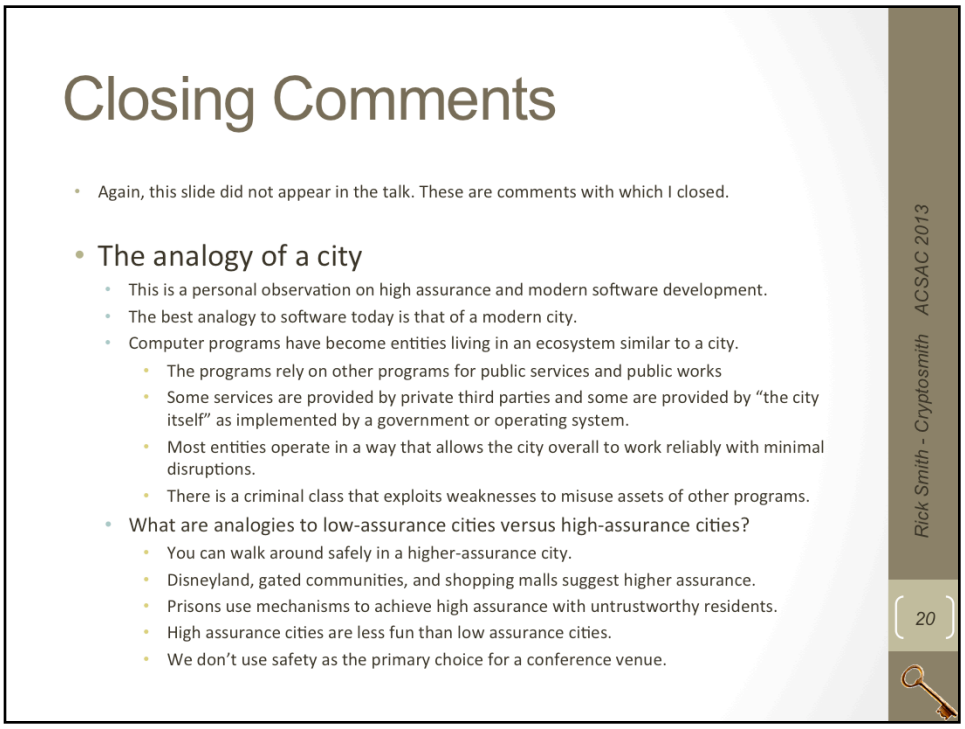


So, yes, it's hard to tell what these numbers mean.

And thanks to Randal Munro and XKCD for that last slide.

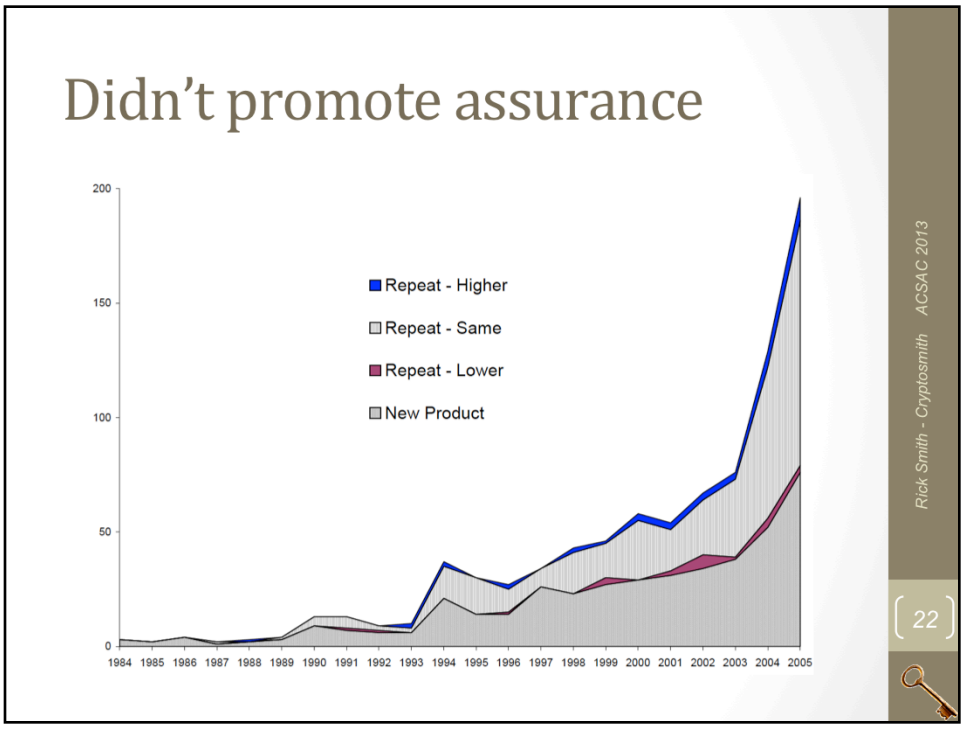
Most papers referenced in this talk may be found here: http://www.cryptosmith.com/archives/61

A list of citations appears at the end.



[This reflects comments I made at the end of my talk]

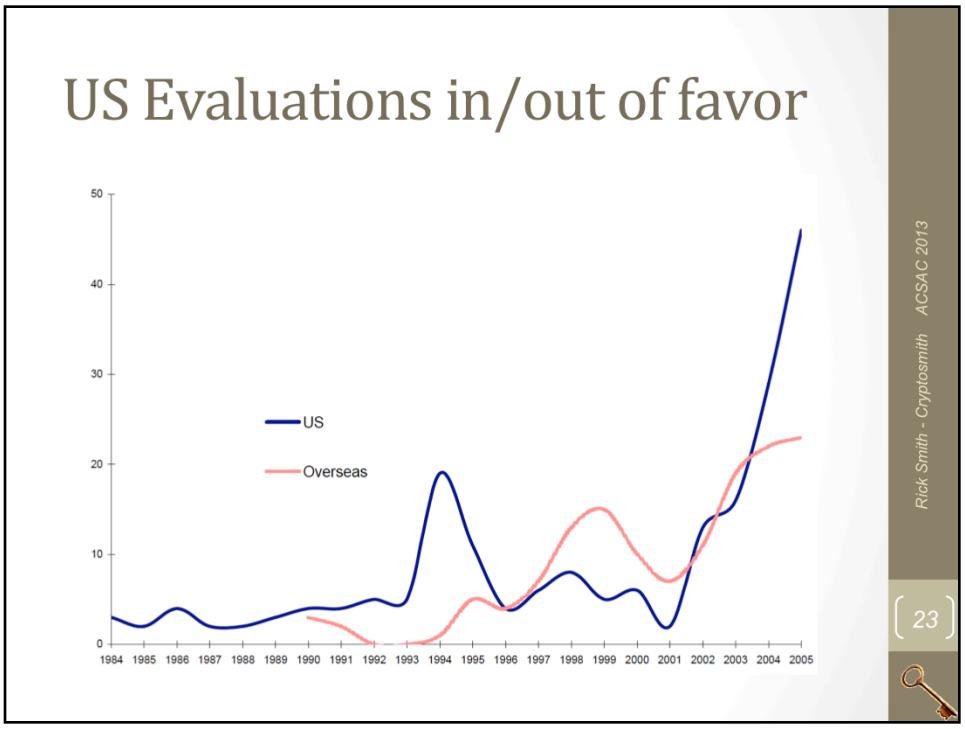
The following are charts and graphs that I prepared for my presentation and then decided not to include.



The TCSEC introduced the C-B-A 3-2-1 sequence to entice or lure vendors to seek higher levels of assurance for their evaluated products.

Those narrow colored lines embedded among the gray show how few products actually changed their assurance level between 1984 and 2006.

The gray wedge on the bottom represents "new" products.

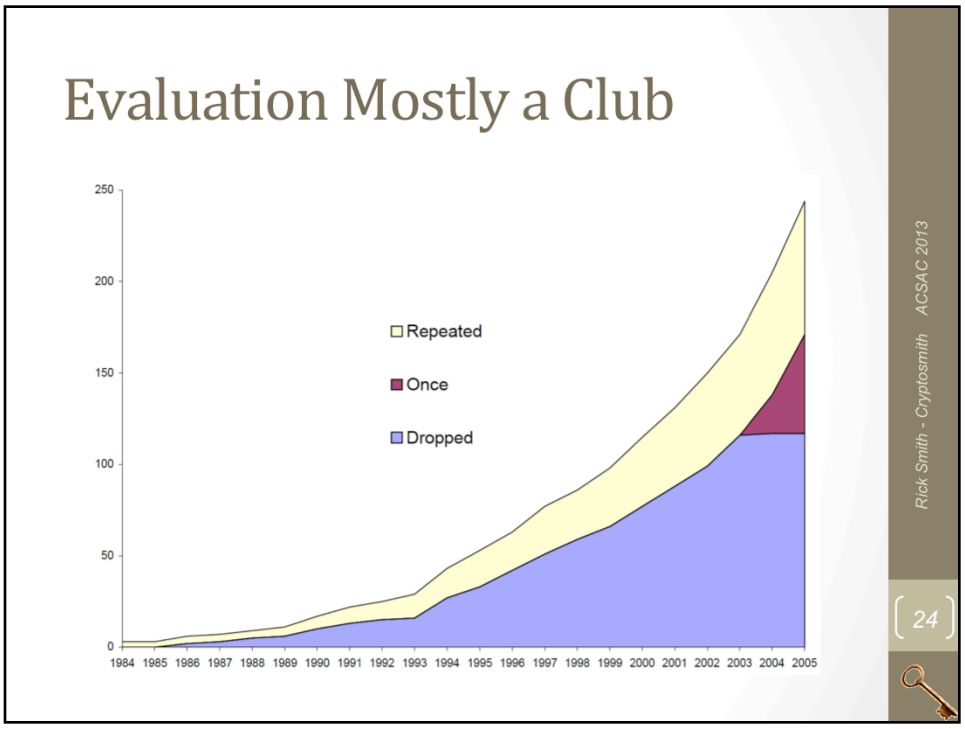


This is actually old news, but it's a relevant bit of history.

This looks at all product evaluations by vendors in the United States. The blue line shows product evaluations performed in the United States, while the pink line shows evaluations performed overseas on US products.

I bring this up because there tends to be a nationalistic element to product evaluations, partly for national defense reasons, and partially to generate work for domestic workers instead of foreign workers.

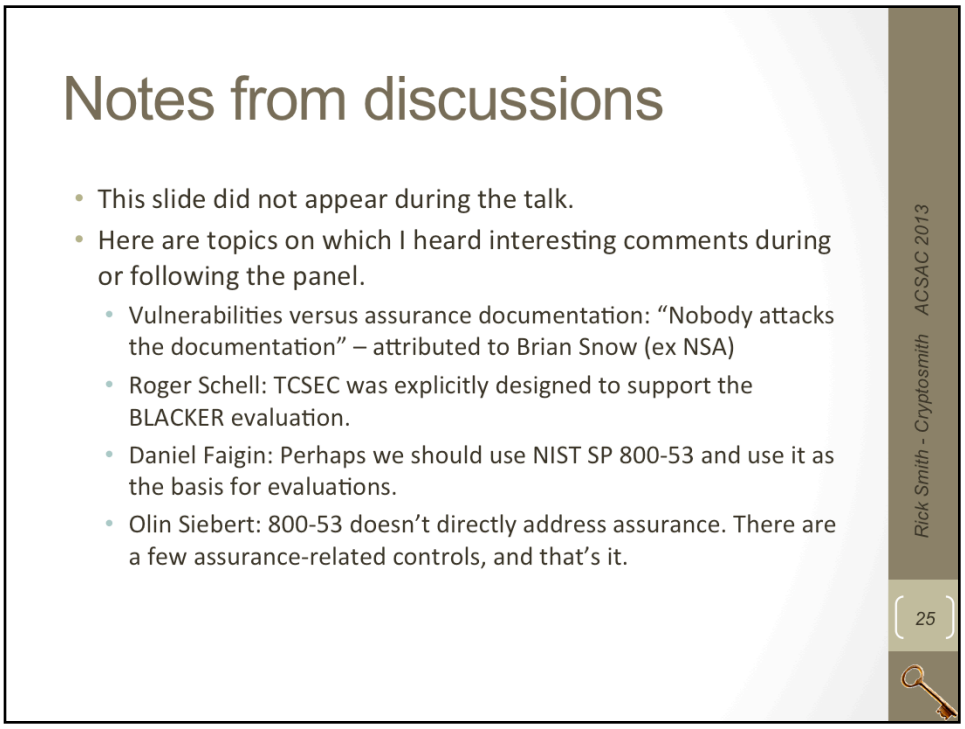
Note that product evaluations weren't consistently performed in the vendor's home country. While there was a big spike in US evaluations in 1994, there was a multi-year spike in foreign evaluations that peaked in 1999. Not all evaluations returned home by 2005, though a lot of US evaluations took place once our government recognized CC evaluations.



Now, here's another chart, and I'm sorry that I can't report on more recent data. This shows the cumulative number of evaluations grouped by vendor, taking into account mergers as much as I could. The big blue triangle indicates vendors and their products who tried the evaluation thing but didn't keep at it.

The cream colored frosting on top shows the product evaluations from vendors who have completed two or more evaluations as of 2005

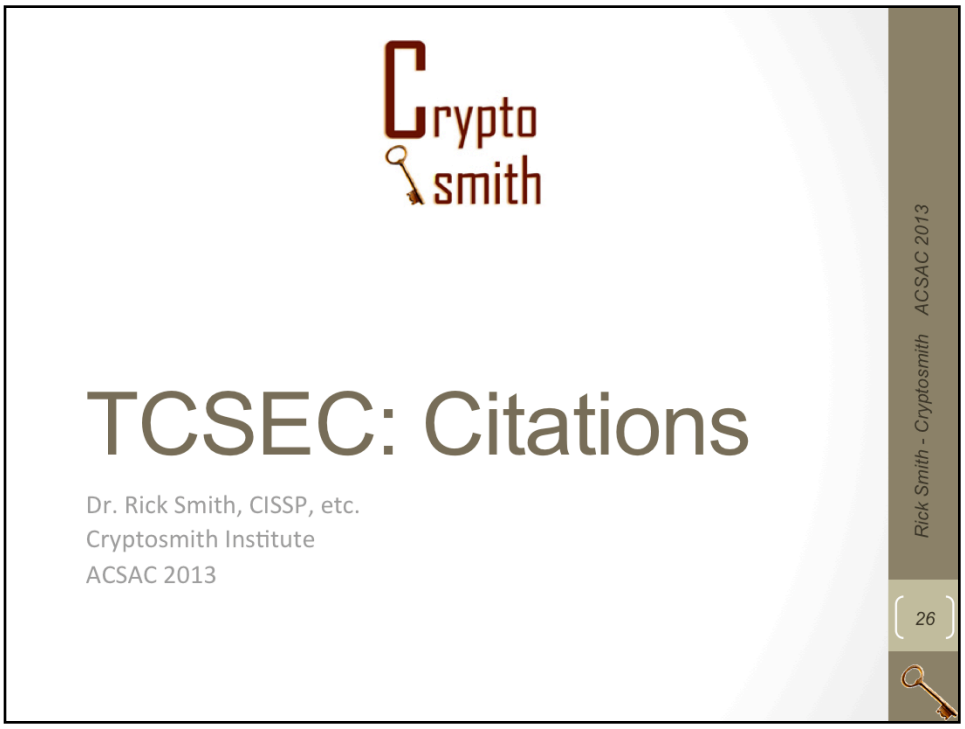
This shows that some vendor organizations buy into the pain and expense of evaluations: they develop the corporate infrastructure, their marketing people get used to promoting it, and it works its way into the product costs. Such enterprises do tend to keep evaluating products, while others may try it once and give it up as too much cost for too little benefit.



Roger Schell and I also spoke briefly about the assurance numbers. He and I have a dramatically different view of the TCSEC and the role of A1-style assurance in theory and practice. Schell is undoubtedly more of an authority than I am. It is also clear, however, that he clings to a very specific point of view.

During the panel, Schell said a properly evaluated system should never require security patching. He gave two examples: BLACKER and the evaluated version of Multics. I have no information about BLACKER, but I used Multics systems that were indeed fixed and improved over time. Schell and a couple of NSA veterans argued that there was at least one Multics system at the agency that ran for 20 years without any patching. The unstated (and unsubstantiated) implication was that the unpatched system remained secure.

When Schell talked to me personally about the LOCK statistics, he said that the numbers were completely different from what he saw on other A1 projects, notably BLACKER. It wasn't clear what he saw, and he has no documentation to support his observation. His conclusion, however, was that LOCK had not in fact performed the A1 process correctly. I can not agree with his conclusion. Published and peer-reviewed materials about the LOCK program likewise contradict Schell's conclusion.



GAO, “Information assurance: national partnership offers benefits, but faces considerable challenges,” (Report GAO 06-392). Washington, DC: United States Government Accountability Office, 2006.

Smith, “Trends in Security Product Evaluations" (PDF), Information Systems Security 16 (4), 2007.

Smith, “Cost profile of a highly assured, secure operating system” ACM Transactions on Information and System Security (TISSEC), 2001

Smith, “Historical Survey of Security Product Evaluations" (PDF),” Proc. 22nd National Information Systems Security Conference, 2000.

Smith, “Constructing a High Assurance Mail Guard" (PDF), Proc. 17th National Computer Security Conference, 1994.

Find my papers here: http://www.cryptosmith.com/archives/61

