Key Escrow from a Safe Distance

Looking Back at the Clipper Chip

Matt Blaze
University of Pennsylvania
blaze@cis.upenn.edu

ABSTRACT

In 1993, the US Government proposed a novel (and highly controversial) approach to cryptography, called *key escrow*. Key escrow cryptosystems used standard symmetric- and public- key ciphers, key management techniques and protocols, but with one added feature: a copy of the current session key, itself encrypted with a key known to the government, was sent at the beginning of every encrypted communication stream. In this way, if a government wiretapper encountered ciphertext produced under a key escrowed cryptosystem, recovering the plaintext would be a simple matter of decrypting the session key with the government's key, regardless of the strength of the underlying cipher algorithms. Key escrow was intended to strike a "balance" between the needs for effective communications security against *bad guys* on the one hand and the occasional need for the *good guys* to be able to recover meaningful content from (presumably) legally-authorized wiretaps.

It didn't quite work out that way.

1. CARROTS, STICKS & ENCRYPTION

The 1990's were interesting times for cryptography. The civilian academic and commercial worlds were becoming seriously interested in this previously obscure and arcane subject, bolstered by new and exciting cryptologic advances coupled with a brave new technological landscape in which securing information was understood to be something that would soon become a very important problem. Information technology was getting inexorably faster, cheaper and better, with the notable exception of security, which seemed actually to get *worse* with every iteration of Moore's law. Cryptography, we believed (or hoped), could come to the rescue, delivering its promise of securing information carried over the increasingly insecure media of the Internet and the "information superhighways" spoken about by visionaries of the time. Cryptography, we increasingly sensed, would soon no longer be merely the esoteric stuff of spies, armies, and governments, but would become an integral part of the public information economy.

But there was a hitch.

Securing information, the government reminded us, can be a double-edged sword. In the classic model for encrypted communication, upstanding citizens *Alice* and *Bob* seek to cryptographically protect their communication from *Eve*, an evildoing eavesdropper. But in a world where transparent cryptography is built in to every telephone, computer and network node, giving the honest Alices and Bobs of the world a mechanism to keep their secrets from Eve might also give the less savory the ability to evade legal surveillance. In other words, what if Eve is occasionally the *good guy?*

And so even before the Web became synonymous with the Internet, before a single bit of encrypted SSL traffic was generated, lines were being drawn for what would become an epic battle that would preoccupy a generation of cryptographers. (And it was a bad time for that community to be preoccupied; this was the same time that the basic foundations of the of web and other critical communications technologies were designed and put into place. We've been living with the security, or lack of security, built in to that infrastructure ever since).

And the eavesdroppers had some leverage. While there were no laws in the United States preventing Alice and Bob from encrypting their communications as unbreakably as they wanted, things weren't so simple for the manufacturers of the hardware and software they might need to effectively do it. Encryption technology, it turned out, was classified as a munition, regulated under the same laws that control trafficking in rocket launchers and nuclear weapons. These laws made it legal to manufacture and sell cryptographic equipment and software domestically, but required a special license from the State Department to export any product that incorporated the technology to any foreign destination. Even making free encryption software available for download over the Internet required obtaining an arms control license.

Make strong encryption that we can't break, the government made clear, and you will never get a license to export your product to the world market. Anyone violating these rules could be prosecuted under the same laws that apply to arms traffickers and smugglers. That stick was more than large enough to discourage the industry from incorporating strong encryption into their products and standards, even as the need for it was increasingly recognized.

But in April, 1993, the government dangled a carrot next to the arms control stick: strong encryption that could be incorporated into products and that could still be freely exported. The system, called *key escrow*, aimed to provide a

ACSAC '11 Dec. 5-9, 2011, Orlando, Florida USA

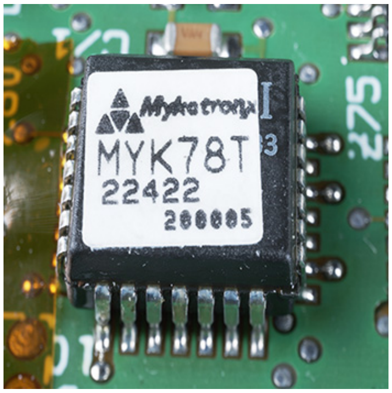

Figure 1: Mykotronx MYK-78T "Clipper" Escrowed Encryption Chip (photo courtesy of the author)

strong cipher for public use with a "back door" that could be exploited by law enforcement agents conducting a (presumably authorized) wiretap.

The centerpiece of the key escrow proposal was a tamper-resistant hardware encryption module, called, in its initial version, the *Clipper Chip*. Clipper was intended as a drop-in replacement for a standard DES chip, but with a new symmetric-key cipher algorithm, called *Skipjack*, designed by the National Security Agency and using an 80 bit key. But before any two Clipper chips could communicate, they would first exchange a *Law Enforcement Access Field (LEAF)* that contained a copy of the current session key, itself encrypted with an "escrowed" *Unit Key* held by the government. Any Clipper-encrypted communication could thus be decrypted by government agents without needing to break the (presumably strong) Skipjack cipher. The agents would be able to recover the session key simply by decrypting the copy in the LEAF (which they would intercept along with the ciphertext) using their escrowed copy of the unit key. The system would, however, still be secure against *unauthorized* eavesdroppers, who presumably would lack access to the government's escrowed key database. Clipper chips (and other escrowed encryption modules in the same family) were to be manufactured, under a government license, by Mykotronx and available for sale to vendors of computers and other secure communications hardware; see Figure 1.

It was, the policymakers must have thought, a perfect solution.

2. BLOWING THE LEAF

Key escrow was not greeted by the technical community with the unreserved warm reception for which the government was perhaps hoping. Almost immediately, many objections were raised that questioned basic assumptions behind the proposal. Why should the bad guys be expected to use an encryption system that the government has announced in advance it can decrypt? How will the key escrow database be secured against unauthorized access? Why should industry adopt expensive hardware encryption (as Clipper required) just as software cryptography was becoming computationally feasible? Why should anyone trust the Skipjack cipher, an unpublished algorithm designed in secret by the NSA? And would the system reliably even solve the problem it aimed to address – ensuring government access to Clipper-encrypted traffic?

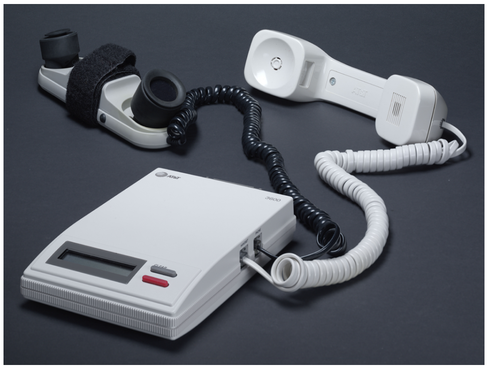

Figure 2: AT&T TSD-3600 Telephone Security Device (photo courtesy of the author)

The history of the 1990's "crypto wars" has been well-chronicled, and it is beyond the scope of this short paper to address all the various problems, theoretical, practical, and political, with key escrow as it was envisioned and as it evolved. Instead, I will offer here a personal perspective, focusing on one small battle at the beginning of this (bloodless but still damaging) "war." I am surely omitting many important episodes, details, and contributors, for which I apologize; what follows should be understood as a war story, which is at best an idiosyncratic, personal, recollection.

2.1 The AT&T Connection

AT&T (my employer at the time) was the first (and ultimately the only) company to produce a full product based on the ill-fated escrow system, but that was not their original plan. The AT&T TSD-3600D, announced in 1992, was to be a voice encryption device that could be installed in any standard wireline "POTS" telephone (between the phone base and the handset). Calls placed to other TSD-3600D-equipped telephones would be automatically digitized (at 4800bps) and encrypted using DES, making eavesdropping on the conversation (by legal or illegal means) effectively infeasible under the technology of the time. The devices weren't cheap, but were designed by the same AT&T business unit that produced the STU-III secure telephone for the government, from which it borrowed some of its design and security features. Two communicating TSDs would first perform a Diffie-Hellman key exchange (768 bit, in the first version of the product) to establish a session key, a 4 character hash of which was displayed on each unit's LCD. To detect "man-in-the-middle" attacks, users could verify (by voice over the encrypted session) that their displayed hashes matched. See Figure 2.

When the US government learned of AT&T's plans to market the TSD, it worried that criminals might use the devices to thwart wiretaps. Plans for a new escrowed encryption system – with a wiretap backdoor – were hurriedly drawn up by the NSA, and AT&T was persuaded to replace the regular (non-escrowed) DES-based encryption scheme in the original TSD product with one based on the new system, which became known as the Clipper chip. In 1993, when Clipper was announced, a new Clipper-based TSD, dubbed the TSD-3600E, was announced at the same time. As incentive for AT&T's cooperation, the government agreed to purchase a significant quantity of Clipper-equipped TSD-3600Es, which sold for over $1000 each. The original unescrowed DES-based TSD-3600D models were recalled by AT&T and quickly disappeared from the market.

When key escrow – and AT&T's involvement in it – was made public, I was just starting my career as a cryptography and security researcher in AT&T's Bell Laboratories division. My colleagues and I, like most members of the civilian cryptography research community, learned about the escrow scheme from the *New York Times,* and we were as skeptical as anyone of the security and practicality of the government's plan. Working for a company that was so prominently involved in what seemed like such a technically ill-advised project was a bit uncomfortable, but it also had its advantages. It was easier to get questions answered, to sort out how this technology was supposed to work. There might even be an opportunity to do some interesting research on the subject. After some poking around, I managed to get hold of a pair of first generation TSD-3600s, but this was less useful than I had hoped, especially given how infrequently I needed to have sensitive telephone conversations. The real breakthrough came when a group from NSA visited the Labs to brief us and answer our questions, which was especially helpful given the dearth of solid publicly released information on the technology. My colleague Steve Bellovin and I both took notes.

As the NSA meeting was breaking up, we asked, half-jokingly, if they'd mind if we posted a summary of what they'd told us to Usenet. To my great surprise, they enthusiastically agreed (evidently they were as eager to get details out as we were to learn them). Steve and I compared notes, and a few days later we posted a short writeup to the `sci.crypt` newsgroup.

In writing the summary, we were careful to stick to the facts, avoiding needlessly inflammatory commentary on the wisdom of key escrow or on whether the NSA should be trusted. This must have come as something of a relief to the NSA's readers of `sci.crypt`, flooded as it was at the time with relentless criticism of the Clipper program and of the government's intentions. A week later, the NSA invited me to come down to their headquarters and R&D facility at Ft. Meade, MD.

To make a long story short, I ended up bringing home samples of a next generation key escrow device. This was a PCMCIA card, code-named *Tessera*[1], intended for secure PC email and other data encryption applications. The Tessera card was based on a version of the key escrow chip, called Capstone, that added some public key exchange and digital signature features to its interface but was otherwise similar to the Clipper chip in its functionality. The NSA people asked only that I play with it and perhaps find interesting applications. I asked if I'd be able to publish my results, and, again to my surprise, they agreed.

2.2 Oracles and Rogues

As a research platform, the Tessera PCMCIA card offered a significant advantage over the Clipper-based TSD-3600 product: an open API with direct access to the encryption chip's functional interface. I could simply connect the card to a reader on my computer and write software to send data directly to and interrogate results directly from the Capstone chip (which included all the functions of the Clipper chip). This would be a much easier way to experiment with key escrow than the other alternative available to me, which involved removing the Clipper chip from a TSD-3600, reverse-engineering its pinout, and building and debugging an interface to it for my computer. With the PCMCIA card, all that was already done.

So I could get right to work. Which, of course, raised the question, to work on what, exactly? What questions would be interesting to ask about key escrow?

At the time, most of the questions and criticisms of the government's key escrow proposal were either political ("why should we trust the government to hold our keys?") or required access to classified information ("how does the Skipjack cipher work?"). But I decided to start with a question that the hardware might be helpful in answering: how do key escrow devices enforce the escrow features? That is, how do they prevent someone from using the Skipjack cipher without sending the data fields that enable a government eavesdropper to recover the key?

I found that the system included a number of safeguards to frustrate some of the most obvious ways to defeat the key escrow field, called the LEAF ("Law Enforcement Access Field").

The details of the LEAF were classified. We knew it was a 128 bit structure containing enough information for law enforcement recovery of the session key with the cooperation of the agencies holding the escrowed unit key database. We were told that the LEAF package contained a 32 bit unique unit identifier (the serial number of the chip that generated the LEAF), the current 80 bit session key (encrypted with the device's unit key) and a 16 bit LEAF checksum. The entire structure was encrypted with a "family key" to produce the final LEAF package. All cryptographic operations in LEAF creation were based on symmetric (secret) key techniques using the Skipjack cipher and other (unspecified) algorithms. The family key was global, shared by all interoperable key escrow devices (but not published). The Skipjack algorithm, the family key, the encryption modes used to encrypt the unit key and the LEAF message, and the details of the checksum algorithm were not made public (and all were protected against reverse engineering by Clipper's tamper-resistant hardware)[2]. Externally, the LEAF was presented as an opaque 128 bit package.

To decrypt escrowed traffic, a law enforcement agency first must intercept the LEAF and the traffic itself using some sort of data wiretapping technology. The LEAF could then be decrypted with the family key, revealing the chip serial number, the unit key-encrypted session key and the LEAF checksum. The target's serial number would then be provided by the agents to two "key escrow agencies," which would each return a "share" of the escrowed unit key associated with the given serial number. The two unit key shares would then be combined (by bitwise exclusive-or) to produce the full unit key, which the law enforcement agency could then use to decrypt the session key. This session key could in turn decrypt the actual intercepted traffic.

[1] *Tessera* was an unfortunate name. It turned out to be a registered trademark of a company that made PCMCIA cards and that wanted nothing to do with key escrow. The NSA eventually had to change the code name to *Fortezza.*

[2] In 1998, after Clipper was abandoned, the NSA declassified and published the Skipjack algorithm. I believe it was, and remains, the only NSA-designed symmetric-key encryption algorithm ever publicly released.

The key escrow system thus relied on the availability of the LEAF along with the encrypted traffic. To force applications to send the LEAF on the same channel as the traffic, key escrow devices would not decrypt data until they had received a valid LEAF for the current session key. Presumably, the chips on each end would perform various integrity checks on received LEAFs prior to accepting them.

To provide a convenient application interface for LEAF management, the devices generated and loaded LEAFs as part of the process of generating and loading the initialization vectors (IVs) for each cryptographic session. The Clipper and Capstone chips provided *generateIV* and *loadIV* functions that operated on 192 bit parameters instead of the usual 64 bits. This "IV" parameter was actually a two part structure containing the standard 64 bit IV concatenated with the 128 bit encrypted LEAF. The *loadIV* operation would fail if the LEAF did not pass an integrity check.

Most details of the LEAF creation method, encryption modes, and data structures, beyond those mentioned above, were classified and were therefore unknown to me. In particular, the key escrow standard did not specify the mechanism that enforced the transmission of the correct LEAF as part of the ciphertext stream. However, I was able to perform a number of simple experiments on my Tessera card to confirm and expand what we knew about the LEAF's internal structure and the way it was used. I found:

- LEAF integrity was verified via redundancy in its internal checksum field. In general, attempts to load an incorrect LEAF failed. This must have been due entirely to the checksum field and not through direct verification of the unit ID or encrypted session key fields; the receiving chip could not confirm the correctness of those fields since it would have no way of knowing the unit ID or unit key of its peer. Therefore, the LEAF must have been testable based entirely on session information known to the receiver (such as the cleartext session key and IV) and that must have been included in the LEAF checksum computation.

- LEAF checksum computation included (implicitly or explicitly) the current IV. The LEAF changed whenever a new IV was generated even when the session key remained the same. Since the IV was not included directly as one of the LEAF fields, the only field it could affect would be the LEAF checksum. Furthermore, receiving devices would refuse to load a LEAF with the wrong IV.

- LEAF checksum computation must have included the cleartext of the current session key. Attempting to load an otherwise valid LEAF (and corresponding IV) from a previous session key failed. It was therefore not possible to "re-use" a LEAF generated from an old session key, even though such a LEAF would itself be internally consistent.

- The LEAF integrity check included every bit of the LEAF. Attempts to load an otherwise valid LEAF with a single bit inverted anywhere in the 128 bit structure always failed.

- The LEAF encryption method diffused its input across the entire 128 bit structure. The LEAF structure or encryption mode was apparently not exactly as specified in publicly released documents. Generating a new IV for a given session key caused changes across the entire LEAF. Since the Skipjack cipherblock size was 64 bits, encryption of the LEAF would have to involve at least two block encryption operations. Since the IV affected only the checksum, and the checksum appeared at the end of the LEAF structure in public documents, we could conclude that at least one of the following was true:

  - The LEAF was encrypted with a non-standard mode in which cleartext in "late" blocks affects the early ciphertext.
  - The LEAF was encrypted with a standard forward-chaining or stream mode but the checksum appears in the first cipherblock of the LEAF.
  - The LEAF was encrypted with a standard forward-chaining or stream mode but the current session IV was itself used to initialize it.

- The LEAF checksum was, in fact, 16 bits long. A brute-force search of the LEAF space for a valid LEAF required about 2^{16} operations.

That last point turned out to be interesting. It meant that it was possible to use 2^{16} Clipper or Capstone chip operations as an "oracle" to generate apparently valid, acceptable LEAFs for the current IV and session key that would actually be useless for escrowed decryption.

So the safeguards that required transmission of a valid LEAF weren't very strong after all. With only access to the chip's standard interface, one could easily create a "rogue" device that could happily interoperate with legitimate escrowed peers, enjoy the use of the strong Skipjack cipher, but be impervious to the key escrow back door. The only thing stopping you was a 16 bit exhaustive search, a very low barrier even in 1993.

In April, 1994, I wrote a paper about all this, "Protocol Failure in the Escrowed Encryption System". I circulated it to a few colleagues, and, not wanting to blindside anyone, also sent a copy to my contacts at NSA. They were, I must say, extremely good natured about it.

Eventually I submitted the paper to the upcoming 1994 ACM *Computer and Communications Security Conference*, which would be held in November. But some time in May, someone (I never found out who) sent a copy to John Markoff, the technology reporter at the *New York Times* who had broken the key escrow story the previous year. He called me to tell me he was writing a story about my paper for his paper, and wondered if I had any comment.

2.3 No Such Thing As Bad PR?

After the *Times* called, it occurred to me that I was in what could be considered an uncomfortable position, an employee of the research division of the same company in whose product I was finding flaws. And it was all based on a controversial wiretapping system created by a secretive government intelligence agency. And now the *New York Times* was about to write about it. And there I was, right at the center of the story. It seemed like a good time to involve management.

I feared that the company might not be completely delighted with my discoveries, or with my writing a paper on the subject. And indeed, executives in parts of AT&T couldn't understand why some kid in the troublemaking, out-of-control research lab would even think that it was a good idea to publish such things. But the Bell Labs management shined. They actively defended the importance of publishing and fully supported me as a member of the public research community, no matter the effect it might have on sales of the TSD or the company's relationship with the government. Our job as scientists, they argued, was to tell the truth. I was never prouder to work there.

Eventually, Markoff called me to let me know that his story would be running in the *Times* the next day. But when I got my copy of the paper, I couldn't find any mention of it. It was only later that I noticed the story in the one place I didn't look: the top of the front page, under the headline *Flaw Found in Federal Plan for Wiretapping.* Apparently cryptography was bigger news than I thought. More likely, it was a slow news day.

3. POSTSCRIPT

Clipper and key escrow eventually faded away. While my paper may have helped accelerate its demise, key escrow would not have been likely to succeed even if the Clipper escrow mechanism had been more robust than it was.

Fundamental problems with the government's vision for key escrow made it inherently unworkable, regardless of the implementation details. First, of course, was the problem of securing a growing database of end-user keys, a very attractive target for unauthorized eavesdroppers who might seek to intercept escrowed traffic themselves. Then there was the economic problem: communications cryptography was, by the 1990's, becoming an essentially zero-marginal-cost technology, something that could often be implemented in software more easily than by adding specialized hardware. But Clipper required the use of hardware cryptography, taking something that was becoming inherently cheap and turning it back into something expensive. The market would ultimately never accept this, even if the trust issues and technical problems could have been worked out.

Over the next few years there were attempts to revive key escrow under various new proposed schemes (the name eventually changed to "key recovery"). By the end of the decade, however, the government gave up. The export rules – the government's main leverage in promoting key escrow – were relaxed to allow mass-market products to use strong cryptography without a special license, and eventually, cryptography started to become integrated into more and more products, software, and protocols. Key escrow was finally dead.

It's probably worth asking whether this was a good thing. Law enforcement, after all, warned that unfettered access to cryptography would make it easier for criminals, spies, and terrorists to cover their tracks.

Fortunately, the worst fears of law enforcement haven't come to pass. Every year since the Nixon administration, the federal government has issued a report on legal wiretaps, giving the number of intercepts, the types of crimes being investigated and other statistics. Since 2002, the report has included another statistic: the number of cases in which encryption encountered on a legal wiretap prevented law enforcement from getting the evidence it was seeking.

With the increasing proliferation of eavesdrop-thwarting encryption built in to our infrastructure since export laws were relaxed a decade ago, we might expect law enforcement wiretap rooms to have become quiet, lonely places.

But maybe not. The latest wiretap report identifies a total of just six (out of 3194) cases last year in which encryption was encountered, and this prevented recovery of evidence a grand total of *zero* times.

What's going on here? Shouldn't all this encryption be affecting government eavesdroppers at least a little bit more than the wiretap report suggests? Do the police know something about cryptanalysis that the rest of us don't, enabling them to effortlessly decrypt criminal messages in real time without batting an eye? Is AES (the federally-approved algorithm that won an open international competition for a new standard block cipher in 2001) part of an elaborate conspiracy to lull us into a sense of complacency while enabling the government to secretly spy on us? Perhaps, but the likely truth is far less exciting, and ultimately, probably more comforting.

The answer is that faced with encryption, capable investigators in federal and local law enforcement have done what they have always done when new technology comes around: they've adapted their methods in order to get their work done. Widespread encryption, rather than shutting down police wiretaps, has actually pushed them in a more reliable – and accountable – direction.

This is because while traffic encryption is highly effective at preventing wholesale, *un-targeted* interception, it does remarkably little to prevent *targeted* government eavesdropping in the complex architectures of modern computing and communication technologies. Yes, today's encryption algorithms are believed to be effectively secure in practice, in the sense that they make it infeasible for even an adversary with the resources of a government to obtain cleartext from ciphertext without access to the key. But government eavesdroppers doesn't have to limit themselves to that scenario for their wiretap targets. They can instead exploit the reality that the cleartext (or the keys to decrypt it) for almost all encrypted traffic today is typically available, somewhere, on a general-purpose computer that is exposed to government access, either explicitly or through surreptitious means. And as systems become more sophisticated and incorporate more features, the exposure of cleartext and keys to third party access tends to increase correspondingly. All without Clipper chips or any other kind of key escrow systems, mandated or not.

If only we had understood that in 1993. We could have saved ourselves a quite a bit of trouble, and maybe spent a bit more time actually making things more secure.