How to Reverse Engineer an EES Device

Michael Roe

Cambridge University Computer Laboratory,
Pembroke Street, Cambridge CB2 3QG, UK
Email: mrr@cl.cam.ac.uk

1 Introduction

In April 1993, the Clinton Administration announced a new encryption system for the protection of government and civilian telephone conversations. This proposal, called the Escrowed Encryption Standard (EES), has a number of features which led to widespread public outcry [9, 14]:

- It contains special features to enable government officials to listen in on civilian telephone conversations.
- Many of the technical details of how it will work are secret.

In this paper we will not consider the moral issues of whether it is right for a government to monitor its citizens in this way; instead, we will concentrate on the technical means which the Clinton Administration is going to use to do it.

The details of the mathematical operations used by the EES are classified. While this would be fine for a miliary cryptographic device, it is extremely undesirable in a product intended for the general commercial market (e.g. one which is to be placed inside every telephone in America):

- It is unnecessary.
- It greatly reduces public confidence in the scheme.
- The secret details are unlikely to remain secret for long.

Clearly, *something* in the system has to be secret. The goal of EES is that government agencies will be able to tap telephone calls, but no-one else will. To achieve this, the government must have, or know, something that no-one else does. This something is the *Unit Key* (KU), a cryptographic key which is different for every telephone. The basic idea is that if you know a phone's unit key you can tap it, and if you don't you can't.

In a well-designed system, this would indeed be how it worked. Everything except the actual value of the unit keys would be public, and it would be clear to everyone that phones could only be tapped by someone who had obtained a unit key through the proper channels.

Unfortunately, EES is not quite like that. As well as the unit keys, most of the technical information about how the system works is secret. This leaves a lingering doubt in people's minds that the undisclosed technical details contain a "back door" which enables phones to be tapped by someone who has not obtained proper legal authorisation.

We will describe a number of ways in which the (classified) internal workings could have been constructed so as to allow government agencies to by-pass the procedural controls, and hence decrypt EES-protected conversations without an authorising court order.

In support of the third claim (that the secret details are unlikely to remain secret for long), we will describe a number of experiments which can used to discover some of the classified internal details of an EES device.

2 Technical Overview

An overview of the EES system is shown in figure 1. Two telephone subscribers are engaged in a telephone conversation. Meanwhile, an FBI agent is intercepting the call. The possibility exists that some other person (who is not a government agent) might also have physical access to the wires that carry the telephone conversation. Briefly, the goal of EES is that the FBI agent will be able to listen in on the call, but the other attacker will not.

To prevent this other attacker from monitoring the call, the voice signal is digitised and encrypted. This encryption is carried out by a "tamper-proof" device within each telephone. Henceforth, we will refer to this "tamper-proof" encryption device as an "EES Device".

Several different models of EES device have been manufactured. The first such device, code-named "Clipper", was manufactured by Mykotronx Ltd. of Torrance, California. Unfortunately, "Clipper" was already a registered trademark of the Intergraph corporation. The Intergraph product of the same name is entirely unrelated, and has nothing to do with telephone tapping. In order to avoid perpetuating this source of confusion, we will use the term "EES device" rather than the code-name.

A block diagram of the internals of an EES device is shown in figure 2. The information used to construct this diagram came from the Escrow and Encryption Standard, FIPS 185 [16]. FIPS 185 acknowledges that the description it contains is incomplete, and asserts that "The complete specifications are classified".

Stored within the device are the unit key (KU) and the family key (KF). The unit key is different for every device, while the family key is the same in all interoperable devices.

The EES devices are used in two phases. In the first phase, the two communicating EES devices are both loaded with the same value of the session key (KS), and they agree upon an *Initialisation Vector* (IV). The session key is generated

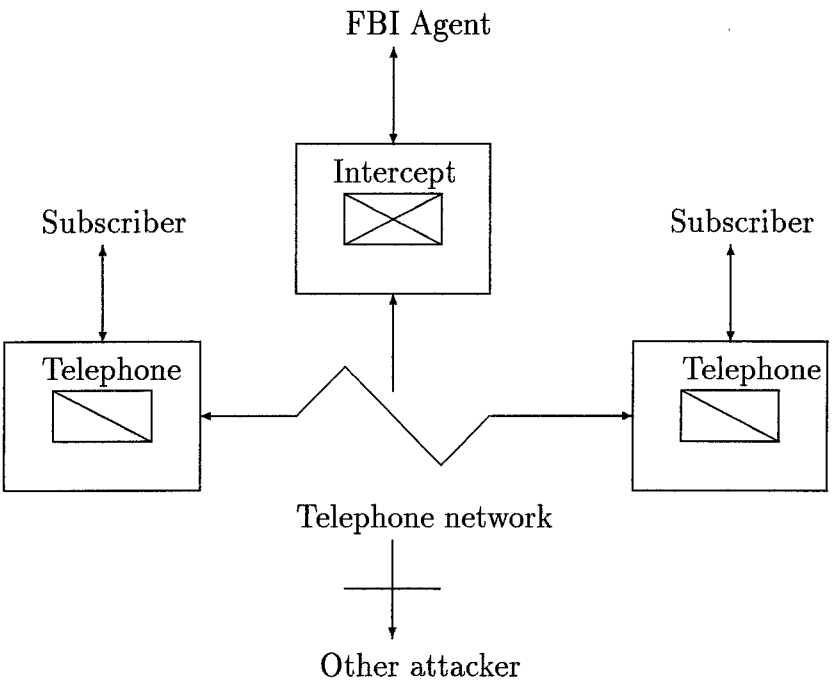

Fig. 1. Overview

by some other electronics elsewhere in the telephone; the Initialisation Vector is generated by one of the EES devices. In the second phase, the two EES devices encrypt and decrypt digitised voice using the session key and Initialisation Vector.

The procedure used in the first phase is asymmetrical between the two EES devices. A choice is made as to which of the two will generate the Initialisation Vector. We will call the device which generates the Initialisation Vector the "initiator" and the device which does not generate the Initialisation Vector key the "responder".

The "initiator" EES device is given as input an 80-bit session key (KS), and provides as output a 64-bit Initialisation Vector (IV) and a 128-bit *Law Enforcement Access Field* (LEAF). According to FIPS 185, the initiator calculates the LEAF in the following way:

1. The session key (KS) is enciphered under the unit key (KU). The mode of operation which is used for this encipherment has not been disclosed. Note, however, that it cannot be ECB mode, as ECB mode operates on 64-bit blocks and the session key is 80 bits long.
2. An Escrow Authenticator (EA) is computed. The Escrow Authenticator is

a form of cryptographic checkvalue. The size of the Escrow Authenticator, the algorithm used to compute it, and the inputs to this algorithm have not been disclosed.

3. The enciphered session key, the Device Identifier (DID) and the Escrow Authenticator are enciphered under the family key (KF) to form the LEAF. The mode of operation which is used for this encipherment has not been disclosed.

The "responder" EES device is given as input the session key (KS), the Initialisation Vector (IV), and the Law Enforcement Access Field (LEAF). The responder decrypts the LEAF value it is given, using the family key (KF) and the function f_3^{-1}. Part of this decrypted result is the Escrow Authenticator (EA). The responder computes the value the Escrow Authenticator *should* have using the function f_2. If the two values are not the same, the responder refuses to function (i.e. it will neither encrypt nor decrypt data).

Important details which are missing from FIPS 185 include:

- The size (in bits) of the Device Identifier (DID) and the Escrow Authenticator (EA).
- The functions f_1, f_2, f_3.
- The inputs to the function f_2.
- The inputs to the function f_4.

On reading FIPS 185, it is not apparent that there is any connection between the Initialisation Vector (IV) and the Law Enforcement Access Field (LEAF). Experiments with actual devices reveal that the value of the IV is used in the computation of the LEAF. As we shall explain later, this connection between the IV and the LEAF is fundamental to the operation of EES.

To complete the picture, the process used to generate the Unit Key and the Family Key is shown in figure 3. (This figure is derived from an article by Dorothy Denning [7]). The first escrow agent supplies KS_1, RS_1 ("Random Seed"), AI_1 ("Arbitrary Input") and KFC_1 ("Family Key Component") , while the second escrow agent supplies KS_2, RS_2, AI_2 and KFC_2. The small squares represent exclusive-or operations. The rectangles labelled $E(KCK)$ represent encipherment using KCK as a key. The key generation unit has two outputs (shown on the right of the figure). One of these outputs is given to each of the two escrow agents.

The process by which KU and KC_1 are generated are not shown on this figure. The "interim" key escrow system which is currently operational uses a variant of the key generation algorithm from annex C of ANSI X9.17 [7, 1]. In the next phase of the deployment of EES, a classified algorithm will be used for key generation.

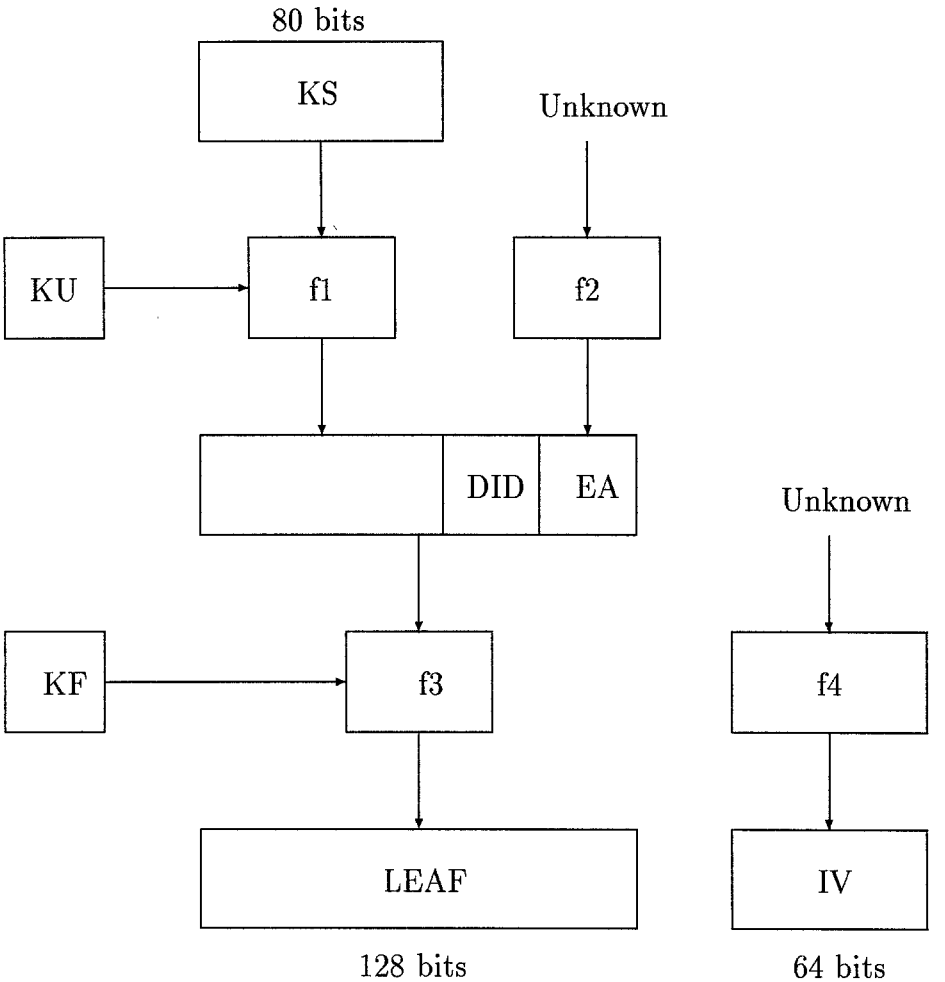

Fig. 2. Leaf Creation Method 1 (LCM1)

3 Correct use of an EES Device

An EES device does not, by itself, constitute a secure communication system. A manufacturer incorporating an EES device into a product (e.g. a telephone) must supply additional components before it will work. The design of EES makes a number of assumptions about how an EES device will be incorporated into a product, but FIPS 185 does not make these explicit. To summarise:

- Some additional means must be provided to transmit the session key (KS) between the communicating parties. The session key must be protected

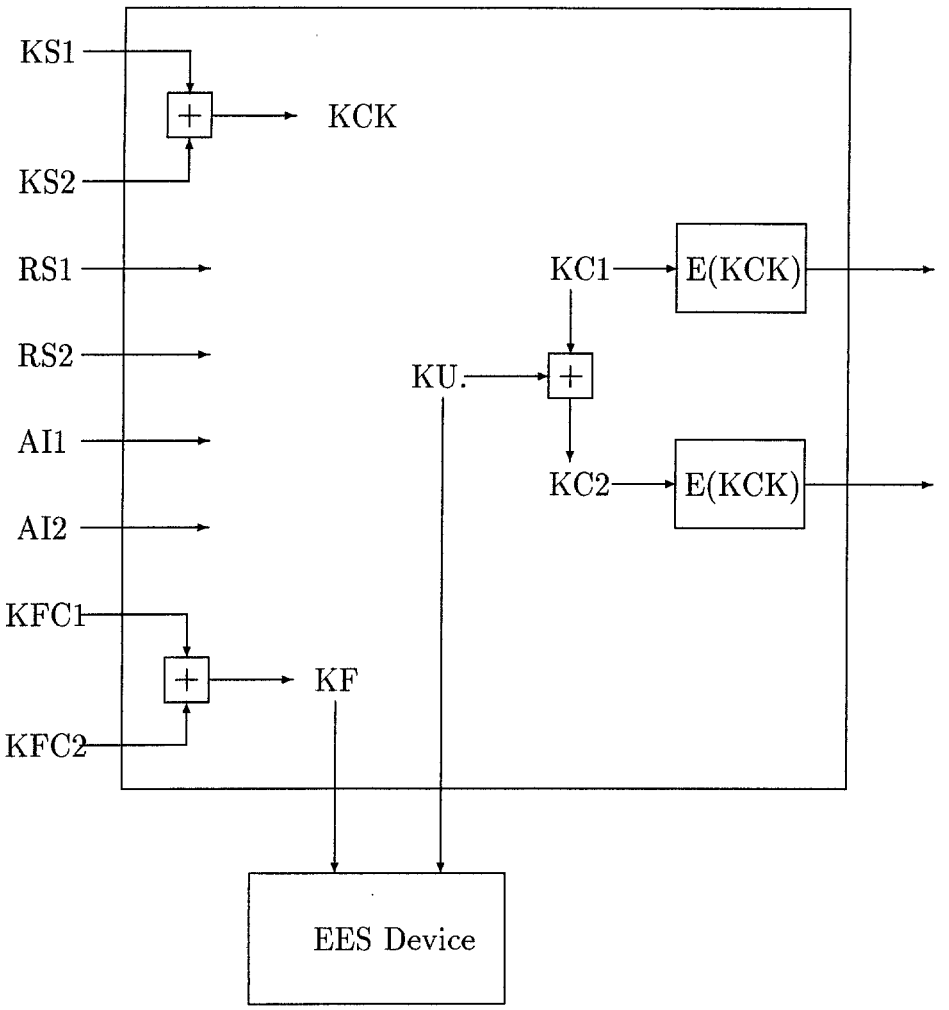

Fig. 3. EES Key Generation

against wiretapping e.g. by encrypting it. Furthermore, the EES encryption algorithm (SKIPJACK) will typically not be used for this encryption; the EES device will not encrypt any data with SKIPJACK until a session key has been exchanged, and you can't exchange a session key until you have some means of encrypting it. To break this deadlock, some other form of cryptography can be used. For example, the "Capstone" EES device provides yet another classified cryptographic algorithm, the NIST Key Exchange Algorithm (KEA) for this purpose.

- The LEAF and IV must also be transmitted between the communicating parties. For EES to work as its designers intended, these must *not* be encrypted.

If they are, the FBI will be unable to decipher an intercepted telephone call.

When it comes to incorporating cryptographic devices into products, it is often the case that Murphy's Law applies: "What can go wrong, will". This is particularly true of devices whose internals are classified, and where the person designing a system around the device is not permitted to know how to use the device correctly.

It is alleged that some of the first designers of products based on EES devices were not even told that the FBI-intercept feature was there; they were simply told that the cryptographic algorithm had a 192-bit Initialisation Vector (in fact, it has a 64-bit IV plus a 128-bit LEAF, giving a total of 192 bits).

The two most obvious ways of using an EES device will not work:

- Taking all the cryptographic variables and transmitting them (unencrypted) to the other side won't work. This results in the session key (KS) being sent in the clear, thus enabling absolutely anyone to decipher the call.
- Taking all the cryptographic variables and encrypting all of them won't work either, as it will prevent the FBI-intercept feature from operating.

4 The Skipjack Review Panel

As the internals of the EES device are classified, and as it was designed by an organisation whose activities include intercepting large numbers of telephone calls, it is reasonable to suspect the existence of some form of "back door" which enables the NSA to decipher EES-protected telephone calls without first obtaining a court order and following the proper procedures.

To allay public fears on the issue, the U.S. Government convened an "independent" panel of experts to examine the security of the cryptographic algorithm (SKIPJACK) used in EES. This review panel only examined the SKIPJACK algorithm itself [4], and did not examine the way it is used by an EES device [1]. As we will show later, even if the SKIPJACK algorithm itself is perfectly good, the way in which it is used by the EES device could introduce many security loopholes. On top of this, the results of the experiments carried out by the review panel fall far short of being a whole-hearted endorsement of SKIPJACK.

The most significant experiment carried out by the SKIPJACK review panel is the *cyclic closure test* (CCT). The objective of this test is to discover how many different possible keys there are. If the number of keys is too small, an attacker can break the algorithm by trying all possible keys. The key input to the SKIPJACK algorithm is 80 bits long, so it is clear that there are at most 2^{80} possible keys. This is a very large number, and should provide adequate security

[1] The review panel originally intended to produce a second report, which would examine the security of the way the algorithms were used. This report was never published.

for most purposes. However, the possibility remains that the number of different keys is much smaller than 2^{80}. For example, devices that implement the Data Encryption Standard (DES) [11] have a 64 bit key input, but only 56 of these bits are used (the remaining 8 bits are used as a parity check). Thus, DES has at most 2^{56} different keys, even though it has a 64 bit key input. If you are given a black box that implements DES, it is easy to determine that it has at most 2^{56} keys; a simple experiment will show that the values of 8 of the input bits have no effect on the output. Determining effective key size is not always so simple; the IBM Commercial Data Masking Facility (CDMF) [8] has only 2^{40} possible keys, and yet changing each of its 64 key input bits changes the output. In these more complex cases, the cyclic closure test is used to determine the true key size.

The cyclic closure test is a statistical test. You first choose a key size k, and then run the test several times to test the hypothesis that the number of different keys is at least 2^k. As this is a statistical test, it is very important that it is run multiple times. This is analogous to testing a coin for fairness; if it comes up heads once, this means nothing; if it comes up heads twice in row, this could be a co-incidence; if it keeps on coming up heads, something is seriously wrong.

The Skipjack review panel ran the CCT 1,000 times to test the hypothesis that there are at least 2^{40} distinct keys. These 1,000 samples gave a strong indication that SKIPJACK really does have at least 2^{40} keys. However, 2^{40} is not big enough to provide adequate security. Algorithms such as CDMF have 2^{40} keys, and they are known to be reasonably easy to break by exhaustive search.

The more interesting test is for the hypothesis that SKIPJACK has at least 2^{56} distinct keys. That is, does it have at least as many keys as DES, the 1970's algorithm which it is intended to replace. Unfortunately, the review panel only published the result of one run of the test for this hypothesis. As the CCT is a statistical test, a single run is almost meaningless. However, it is worth noting that for the one run they published, the measured cycle length (28,767,197) was shorter than the expected cycle length (168,216,976). If this happens consistently, then there is strong evidence that SKIPJACK has fewer than 2^{56} keys, rather than the 2^{80} which its proponents claim. In fact, the review team did run the test again, although they did not publish the second result [5]. The second time, the measured cycle length was also shorter than it should be.

5 Experimental Apparatus

The experiments described in the next section require access to an EES device. One way to obtain such a device would be to obtain an EES-protected telephone and dismantle it. The EES devices used in telephones are often of the surface-mounted "Quad flat pack" type and are soldered to the printed circuit board. This makes it troublesome to remove the device without destroying it. However, most university electronics laboratories will have equipment for doing this.

An alternative means of obtaining an EES device is to use the identification

smart-cards (code-name "Tessera" [2]) used in the U.S. Defense Message System. The Tessera card has the great advantage that it is designed to be plugged into a computer (via the PCMCIA socket on an IBM-compatible laptop). This means that it can be used to perform the experiments described in this section without needing the use of a soldering iron.

There are some significant differences between the facilities provided by a telephone-oriented EES device such as "Clipper" and a computer-oriented EES device such as the Tessera card:

1. Some of the devices intended for use in telephones only support one cryptographic mode of operation (typically, OFB mode). The Tessera card supports all the FIPS 81 modes [12], including Electronic Codebook Mode.
2. The Tessera card imposes an additional layer of key management protocols on top of EES. Functionally, the Tessera card can be regarded as two different devices on one chip: an EES device, and another processor which mediates all accesses to the EES device. It has been alleged that this second processor is based on an ARM 600 series macrocell. No EES device lets the user choose the Initialisation Vector; the Tessera card does not let the user choose (or even know) the session key either.
   Tessera provides two basic means of setting up a session key: a shared key can be established between two cards using the NIST KEA algorithm, or a previously saved session key can be reloaded. The operation to reload a key takes as input an 80 bit key and 16 bits of redundancy, all encrypted under a master key which is unique to the device. The Tessera card will only reload a key if the 16 bits of redundancy take on the correct value. Furthermore, the device master key is different for each card, and is not revealed to the owner of the card. Note that this makes it impossible for the owner of the device to either choose or to determine the value of a session key.

After I had started work on this paper, the Tessera cards were withdrawn and replaced with a new computer-based EES device, which was code-named "Fortezza". The change in name was because "Tessera" (like "Clipper") was already a registered trademark of another company. The Fortezza card also incorporated a technical change which was intended to prevent the LEAF-forging attack described by Matt Blaze [3]. After it has rejected several false LEAFs in succession, the Fortezza card goes into a state in which it does nothing until it is reset.

This modification does not actually prevent the LEAF-forging attack: it merely makes it slower. For example, the LEAF-forger can just reset the card each time it goes into this state. However, this is probably enough to prevent people from using LEAF-forging as a practical way to defeat the key escrow system. It was doubtful whether anyone would do this with the Tessera card, as 40 minutes is

[2] *Tessera* is the Latin word for a clay tile. Such tiles were used as primitive identity cards in the Roman empire.

a long time to wait every time you send a message. The Fortezza card makes LEAF-forging take even longer than 40 minutes, and hence it is very unlikely that anyone would use this a means to defeat the escrow system.

However, there are other reasons for creating forged LEAFs. Some of the experiments described in this paper depend upon the ability to create forged LEAFs. These experiments are still possible with the Fortezza card, but take longer. They could be redesigned to take account of the different performance characteristics of the Fortezza card. For example, in many cases there are two possible experiments, one which involves generating a large number of real LEAFs, and another which involves verifying a large number of fake LEAFs. With the Tessera card, there is no reason to choose one of these experiments over the other. With the Fortezza card, experiments which involve generating LEAFs are much quicker than experiments which involve verifying fake LEAFs.

6 Experiments

6.1 Re-Run Cyclic Closure Test

Rationale As described in the previous section, the SKIPJACK review panel decided to stop running the cyclic closure test on the SKIPJACK algorithm just as the results began to look interesting (i.e. began to provide evidence of a weakness in the algorithm). It would be interesting to continue the test, to find out if the results obtained by the SKIPJACK review panel were a statistical fluke or a real sign of weakness.

The cyclic closure test is a "black box" test; it does not require use of any information about the internals of the algorithm under test. Thus it is often possible to perform the cyclic closure test using only a hardware implementation of a secret algorithm. Unfortunately, it is not possible to perform this test using either the Tessera card or the telephone-oriented EES devices.

This experiment (or rather, non-experiment!) is included for two reasons:

- The fact that this experiment can't be performed is very significant. If the Skipjack algorithm was actually a weak algorithm, it would be very hard for users to discover this.
- This experiment might be possible with EES devices other than the Tessera card or Clipper.

Method The cyclic closure test involves finding a cycle in the following iteration:

$$x_{i+1} = E(d, x_i)$$

Where $E(d, k)$ is the result of encrypting d with key k in Electronic Codebook Mode, and d is a data block chosen at random.

The telephone-oriented EES devices *cannot* be used for this test, as they are only capable of OFB-mode encipherment with an internally generated IV. A device which supported OFB-mode with an externally-supplied IV could still be used for the cyclic closure test (simply keep the IV constant). However, the telephone-oriented devices generate a new IV at random each time they are loaded with a new key. This ruins the cyclic closure test, as the ever-changing IV will prevent the iteration from settling down into a repeating cycle.

The Tessera card also cannot be used for this test, as it does not allow the user to choose a session key. It is interesting to note that this test would be possible if the Tessera card did not put any redundancy in saved session keys. The test would be run as follows: encrypt d under the current session key, pretend that the result is a saved session key, "reload" it, and repeat. This gives the following iteration:

$$x_{i+1} = E(d, D(x_i, K_M))$$

Where K_M is the (unknown) device master key. Note that encryption and decryption are both reversible operations, so decrypting a key with K_M permutes the key space without changing its size. Thus, signs of weakness found with this modified iteration are just as valid as those found using the standard cyclic closure test.

However, the 16 bits of redundancy prevent us from using this trick; the device will usually detect that the input to the reload operation is not a genuine saved key (it will be fooled one time in 2^{16}, but this isn't often enough to be useful for the CCT).

Resources Needed This estimate of resources is somewhat hypothetical, as this experiment isn't possible for the reasons outlined above. However, if an EES device without these restrictions was obtained, the resources required would be approximately as follows.

Assuming that the true key size is 56 bits (rather than 80 as is claimed) the cyclic closure test will take on average 2×2^{28} iterations to find a cycle. Assuming that each key change takes 50ms (the Tessera card is rather slow at changing keys) then the cyclic closure test will take 310 days. To give statistically significant results, this test must be run at least three times. Thus, the total resources needed to perform this test are about 3 machine-years.

Note that if the true key size is greater than 56 bits, this test will not need any more time. If a cycle is not found after the expected number of iterations, then the run should be terminated. If all three runs are terminated without finding a cycle, then it is possible to reject the hypothesis that the the key size is 2^{56} or smaller.

Opportunities for parallelism Clearly, each of the runs of the cyclic closure test can be run in parallel. Hence this experiment could be performed in one year using three devices rather than three years using one device. If a very large number of devices are available, and it is desired to perform the test quickly, then it is possible to use alternative algorithms which permit a greater degree of parallelism. Some algorithms of this type are described in a paper by Michael Wiener [17].

6.2 Search for Equivalent Keys

Rationale As the cyclic closure test cannot be performed with the Tessera card, it is necessary to use some other test for effective key size. The test described in this section tests the size of the key space by searching for pairs of keys which have the same effect on all plaintexts. If the true key size is less than the claimed 2^{80} there will be many such pairs. Finding even one such pair would reveal interesting information about the internal structure of the SKIPJACK algorithm (analagous to the existence of weak keys in DES [10]). Finding many such pairs would be proof of a serious weakness in the algorithm.

This test is not quite as satisfactory as the cyclic closure test, for at least the following reasons:

- It requires at least $O(2^{28})$ 64-bit words of disc space. The cyclic closure test needs only a very small about of disc space and memory.
- We have a reason to believe that SKIPJACK will fail the cyclic closure test (the results of the Skipjack review panel). We have no reason to suppose that this alternative test will be as good at detecting SKIPJACK's weaknesses.

Method Repeat the following at least 2^{28} times:

1. Get the Tessera card to generate a random session key. (It can do this, even though it won't reveal the value of the key it has generated).
2. Use the "save key" transaction to save this key (encrypted under the device master key!) to external storage.
3. Encrypt a fixed test pattern (0, say) in ECB mode using the session key, and save the result.

The results of encrypting 0 under different keys can then be sorted in order to find matches. A match can be caused in any of the following three ways:

1. The Tessera card generated the same session key on more than one occasion. This case can be recognised by comparing the saved session keys (which are in enciphered form). If the two session keys are equal, then their enciphered forms will also be equal, as the enciphered form does not contain any randomness to prevent comparison.

2. The two keys have different effects in general, but happen to give the same result when enciphering 0.
   This case can be recognised by reloading the saved session keys, and using them to encipher other values (e.g. 1, 2, ...) The existence of such pairs of keys is to be expected, and is not a sign of weakness in the algorithm.
3. The two keys have the same effect on all (or nearly all) inputs.
   This case can be recognised by re-loading the saved session keys, and using them to encipher some randomly-chosen values. If the two keys have the same effect on a large number of randomly-chosen inputs, then this is strong evidence that they have the same effect on most inputs. If this case is encountered, it is a sign of weakness in the algorithm.

6.3 Check that IV is used as an Initialisation Vector

Rationale This experiment confirms that the "IV" parameter which is exchanged between EES devices is the same as the Initialisation Vector that the device actually uses when it encrypts or decrypts. An alternative possibility would be that "IV" is the Initialisation Vector enciphered under a key common to all EES devices.

Method One EES device (the "initiator") is used to generate an IV and LEAF for session key k_1. The initiating device is then used to encipher a sample message in OFB mode. A second EES device (the "responder") is used to decipher this message twice, in two different ways:

- Firstly, the responder is put into OFB mode and used to decipher the message.
- Secondly, the responder is put into ECB mode. The responder is used to decipher the message, with the OFB mode chaining being implemented in software external to the device. This is possible because OFB mode is defined in terms of invocations of a "black box" which provides ECB mode.

If the "IV" input to the device was not the same as the Initialisation Vector used by the device in OFB-mode chaining, then these two decipherments would give different answers.

Results This experiment has been carried out with a Tessera card by Matt Blaze [2]; both decipherments are the same. A similar experiment using CBC mode instead of OFB mode also showed that on-chip chaining and software chaining gave the same answer.

Conclusions The "IV" input really is the Initialisation Vector, and it is not enciphered. Furthermore, we can conclude that when OFB mode is selected, the device really does encipher and decipher in OFB mode (rather than some other keystream mode).

Further Remarks We can also conclude that the mode of operation is not one of the variables which are used to compute the Escrow Authenticator (EA). If the mode of operation were an input to EA, it would be impossible to use a LEAF generated by an initiator in one mode with a responder in a different mode.

However, even if the mode of operation was an input to the Escrow Authenticator, it would still be possible to perform this experiment. The reason for this is that the Initialisation Vector has no effect in ECB mode. A modified version of this experiment would involve two LEAFs, one (L_1) for OFB mode, session key k_1 and Initialisation Vector V_1, and the other (L_2) for ECB mode, session key k_1 and Initialisation Vector V_2. As the device chooses its own IV when generating a LEAF, $V_1 \neq V_2$. The software emulation of OFB mode would input k_1, L_2 and V_2 to the EES device, but would use V_1 as the real Initialisation Vector in the software emulation of chaining.

6.4 Determine whether device checks session key equals enciphered session key

Rationale A "responding" EES device is supplied with the session key via two different paths:

- Via a key management mechanism external to the device. We shall call this the "clear text session key".
- Via the LEAF, which contains the session key enciphered under the initiator's unit key and the shared family key. We shall call this the "enciphered session key".

Given these inputs, a device might do one of the following things:

- Use the clear text session key, and ignore the enciphered session key.
- Use the enciphered session key, and ignore the clear text session key.
- Check that the two versions of the key are equal, and signal an error condition if they are not equal.

If there is no "back door" in the key escrow mechanism, the responding device should not be able to use either the second or the third of these possible methods. The Unit Key is supposed to be secret and unique to every device. Hence the responding EES device should not know the unit key of the initiating device, and so should not be able to extract the session key from the LEAF.

However, the question still remains. It is theoretically possible that some of the claims made about the device are false, that in fact there is no unit key, and the responding device does check that the two values of the key are equal. It is desirable to have an experiment to show that the device does not perform this check.

Method The first EES device (the "initiator") is used to generate an IV and LEAF for a particular session key (k_1). This IV and LEAF are loaded into a second EES device (the "responder"), but with a different session key (k_2). A data block of 64 bits is encrypted using the initiating device, and the resulting ciphertext is decrypted using the responding device. For the purposes of this experiment, it does not matter which of the supported modes of operation is used for this encryption and decryption (although initiator and responder must use the same mode).

Results Most of the time, the responding device will detect that the session key has been modified. The Tessera card reveals that it has detected the tampering by refusing to perform encipherment or decipherment with the modified session key.

Some of the time (1 in 2^{16} for the Tessera card [3]) the responding device accepts the modified session key. In these cases it is possible to continue the experiment and to attempt to decipher the block of ciphertext. The result of this decipherment is nearly always different from the original input plaintext.

Conclusions It is possible to reject the hypothesis that the responding device only uses the enciphered session key. If the responder only used the enciphered session key, it would decrypt the ciphertext block with k_1, and hence recover the original plaintext. Instead, the responder decrypts the ciphertext block with k_2; as this ciphertext was produced by enciphering with k_1 (not k_2) this results in a block which is almost always different from the original input.

The observed behaviour can be explained by any of the following hypotheses:

- The responder does not check that the session keys are equal; it only checks the value of the Escrow Authenticator (EA).
- It does check that the session keys are equal, but never acts upon the result. This is equivalent to the above!
- The responder extracts the session key from the LEAF, computes a hash of the key which is *different* from the Escrow Authenticator (EA), and uses this other hash to check if the two keys are equal. While this is theoretically possible, it makes no sense!

Given the above experiment, and the claim that the unit keys are unique to each device, it seems likely that the responder does not check the encrypted session

key in the LEAF. Furthermore, we can conclude that the size of the Escrow Authenticator is at least 16 bits. If the Escrow Authenticator was smaller, then modified session keys would be accepted more frequently.

Further Remarks It is theoretically possible that the Escrow Authenticator is larger than 16 bits, but the responding device only checks 16 bits of it. Why would the device be designed to do this? One possible reason is as follows: by rejecting LEAFs with a bad EA, the responding device is revealing valuable information about the secret family key, KF. An attacker who does not know KF can still use a responding EES device as an oracle to provide answers to certain questions about the internals of a LEAF. Indeed, most of the experiments described in this paper are different methods of using a responder as an oracle. To prevent this type of attack on the family key, it would be better if a responding EES device always accepted a LEAF, regardless of the value of EA. However, if responding devices never checked EA, then it would be very easy to by-pass the key escrow mechanism (e.g. by not transmitting the LEAF at all).

A compromise between these positions would be for the responding devices to only check part of the EA. The devices would check enough of the EA to make it hard to bypass the key escrow system, but not enough of it to give an attacker perfect information about the LEAF.

How big should the LEAF be, and how many bits of it should be checked? Enough bits should be checked to make it inconvenient to forge an acceptable LEAF. If t is the time taken for a device to generate a single LEAF, and k is the number of bits that are checked, $t \times 2^k$ should be an inconveniently long time. A value of 50 ms for t and 16 for k results in LEAF forgery requiring 55 minutes, which is certainly inconvenient.

The Escrow Authenticator should contain enough extra bits to make it reasonably likely that a forged LEAF will not be a perfect forgery. However, the EA cannot be made very large because there is limited amount of space in the LEAF (128 bits). A LEAF size of 18 bits (with 16 of them checked) would mean that a single forged LEAF has a 75% chance of not being perfect. An 18-bit EA would ensure that anyone who forges LEAFs regularly will almost certainly create some imperfect forgeries.

If the responding devices do not check these extra bits, does it matter that they are there? It does, because there might be specialised equipment (such as the FBI intercept processors) that does check the additional bits. This would enable government officials (who have the special equipment) to detect attempts at LEAF forgery. Note that for this to work, the extra two bits do not even need to be the output of a strong hash function; it will even work if they are always zero. This would allow this function to be concealed even from people who have access to the (classified) description of the internals of the device; the specification could say "these two bits always zero (reserved for future use)" without arousing suspicion as to their true purpose. Indeed, it would be sufficient

to make the serial number two bits longer, and to arrange that no EES device is ever manufactured with the top two bits of its serial number non-zero. Then, if an intercept processor sees a LEAF containing a serial number with either of the top two bits set, it has strong evidence that the LEAF has been forged.

It is worth noting that in Dorothy Denning's original description of Clipper [6], the serial number is described as being 30 bits long. As the LEAF is 128 bits long, and the encrypted session key is 80 bits long, this would leave 18 bits for the Escrow Authenticator (EA). Subsequent descriptions of Clipper take care to avoid mentioning the length of EA [16].

6.5 Determine whether the session key affects EA

Method The first EES device (the "initiator") is used to generate an IV and LEAF for a particular session key (k_1). This IV and LEAF are loaded into a second EES device (the "responder"), but with a different session key (k_2).

Results This experiment was first performed by Matt Blaze [3]. Most of the time, the responding device will refuse to encrypt using the modified session key. Given that the responder does not make use of the session key inside the LEAF, this means that the value of the session key affects the value of EA.

6.6 Determine whether the IV affects EA

Method The first EES device (the "initiator") is used to generate an IV and LEAF for a particular session key (k_1). This session key and LEAF are loaded into a second EES device (the "responder"), with a different IV.

Results Most of the time, the responding device will refuse to encrypt using the modified IV.

Conclusions There are two possible explanations for this behaviour:

1. The value of the IV is an input to the function (f_2) used to compute EA. If the IV is modified, the responder will compute a different value of the Escrow Authenticator from that contained in the LEAF, and so will reject the LEAF.
2. The IV is an input to the function (f_3) used to encipher the LEAF. That is, IV is used as an Initialisation Vector for two different encipherments: the encipherment of the user-supplied data with the session key KS, and the encipherment of the LEAF with the family key KF.
   If the IV is modified, the responder will incorrectly decipher the LEAF, the EA contained within the incorrectly deciphered LEAF will not match the recomputed EA, and so the responder will reject the LEAF.

Further Remarks Encrypting two different messages with the same key and the same IV is usually a bad idea, as it makes some additional cryptanalytical attacks possible. However, what may be happening with EES is that two different messages (the LEAF and the user data) are enciphered with the same IV but *different* keys. This is certainly unusual, but does not seem to be cryptographically weak.

6.7 Determine whether f_2 is a checksum

Rationale If the function f_2 (used to compute EA) was one of the commonly-used error detecting codes (such as a simple checksum, a CRC, or Fletcher's algorithm), then it would be possible to produce false LEAF values in a much more efficient way that that proposed by Matt Blaze [3]. Hence, it is likely that the designers of EES made the function f_2 a cryptographically strong hash function. This experiment determines whether or not this is the case.

Method Use an "initiating" EES device to produce a LEAF (L_1) and IV (V_1) for key k_1. For each of the common error detecting codes, calculate values of the key (k_2) and IV (V_2) which give the same error detection code as k_1 and V_1. For some codes, doing this requires knowledge of the order in which k and V occur in the input to f_2; simply try all possible orderings.

Are the modified key and IV, together with the original LEAF, accepted by a "responding" EES device? If they are, repeat the experiment with different values of k_1. If the fake k_2 and V_2 are always accepted, then the function f_2 has been found.

Further Remarks With the Tessera card, it is impossible for the user to select the session key, so this experiment is restricted to changing the Initialisation Vector. Furthermore, if the Initialisation Vector is also used as the IV in the decipherment of the LEAF (as suggested in section 6.6) then it will not be possible to derive useful information about f_2 by changing IV. This experiment is only really informative with EES devices that allow the user to select a session key.

6.8 Determine whether DID or encrypted session keys affects EA

Method With a fixed value of the session key (k_1) and a fixed value of the IV (V_1), try random values of the LEAF until two LEAF values (L_1 and L_2) are found that will be accepted by a responding device. Then the following two equations hold:

$$f_3^{-1}(L_1) = (E'_1, D'_1, f_2(k_1, V_1, E'_1, D'_1, \ldots))$$

$$f_3^{-1}(L_2) = (E_2', D_2', f_2(k_1, V_1, E_2', D_2', \ldots))$$

One in 2^{16} LEAF values will be accepted, so this can be done in a few hours. It is highly likely that these two fake LEAF values will contain different encrypted session keys. That is, $E_1' \neq E_2'$.

With one fake LEAF (say L_1), keep the IV fixed at V_1 and find other values of the session key that will be accepted by a responding device. Call these k_i'.

$$\forall i : f_3^{-1}(L_1) = (E_1', D_1', f_2(k_i', V_1, E_1', D_1', \ldots))$$

Check to see whether these k_i are also accepted by a device given the other fake LEAF. If they are always accepted, then the following equation holds:

$$\forall i : f_3^{-1}(L_2) = (E_2', D_1', f_2(k_i', V_1, E_2', D_2', \ldots)$$

That is, changing the values of the Device ID (DID) and the encrypted session key from (E_1', D_1') to (E_1', D_2') does not change the set of keys (k') for which the value of f_2 is constant.

If f_2 were a simple function, this fact would not allow us to draw any conclusions about its inputs. For example, f_2 might be a checksum of all the inputs (including E and D), with $E_1'+D_1' = E_2'+D_2'$. However, if f_2 is a collision-free hash function, it would follow that E and D could not be inputs to f_2!

Note that this experiment will still work even if the IV is used in the decipherment of the LEAF (as suggested in section 6.6).

6.9 Determine whether encrypted session keys affects EA

Method With two different session keys (k_1 and k_2), use an initiating device to generate (IV, LEAF) pairs until an IV collision is found. That is, we have two LEAFs generated by the same device (with the same DID) with the same IV but different session keys.

Repeat this process until many such pairs are found. Eventually, a pair will be found where k_1 is accepted with the second LEAF as well as the first. Call these two LEAFs L_1 and L_2.

Then find session keys k_i such that k_i is accepted with L_1 and V_1. If these keys are also accepted with L_2 and V_2, then the enciphered session key is probably not used in the computation of EA. If these keys are usually rejected, then the enciphered session key is used in the computation of EA.

Note that this experiment will still work even if the IV is used in the decipherment of the LEAF (as suggested in section 6.6).

Resource Requirements Assuming that IVs are randomly generated, about 2^{32} IVs will need to be generated before an IV collision is found. If it takes significantly fewer or more attempts to find a collision, this is very interesting:

- If it takes longer, then the initiating device must have some memory; perhaps the IV is the result of encrypting a counter with a key stored inside the device. If this is the case, then this experiment cannot be completed.
- If a collision is found sooner, then the IV is not as random as it should be. Perhaps a few bits of the IV are being used to leak part of the session key (or even the unit key!) to an eavesdropper? This would be very bad!

Assume that it does indeed take 2^{32} attempts to find a collision. One in 2^{16} of these collisions will have the additional property that k_1 is accepted with both IVs. Thus the total number of trials needed is less than 2^{48}. (It is less because there are economies of scale when searching for multiple collisions).

This is undoubtedly an expensive and time-consuming experiment. However, it only needs to be done once *for the algorithm*. The discussions of DES breaking machines usually hypothesise an attacker being prepared to do an $O(2^{56})$ search for each key she wishes to break. Here, it is only necessary to do an $O(2^{48})$ search *once*; this will reveal a (classified) fact about every EES device in existence.

7 Other Observations

As far as we know, there has never been a good description of the internals of an EES device in the unclassified literature. Attempts to reverse engineer these devices have led to a better description of their internal workings than has previously been available. This in turn led to some new observations on the security of the scheme [13].

7.1 Forgery

The telephone-oriented EES devices only support Output Feedback Mode. This mode does not provide integrity or authentication, although someone who had not examined the scheme in detail might be fooled into thinking that it does. The fact that OFB mode is unsuitable for integrity or authentication has been known for at least ten years [15]. The new observation is that a workable key escrow system would need to have authentication as well as confidentiality.

Consider the following scenario. A person is on trial for a criminal offence, and the only evidence for the prosecution is an intercept of a telephone call that was protected by an EES device. The prosecution shows that the ciphertext can be decrypted into intelligible speech by an intercept processor loaded with the escrowed copy of the defendant's telephone's unit key. The content of the telephone conversation is clearly incriminating. However, the defendant denies

that they made the telephone call. The compression algorithms used in EES telephones distort speech, so it is hard to recognise the voice. The prosecution argues that since only the defendant's escrowed key will decipher the call, it must be the case that it was made from the defendant's telephone.

The defence argues that the call has been forged in the following manner. Given the ciphertext (from wiretapping) and the plaintext (either as the output from an intercept processor, or from a bug in the same room as the telephone) it is possible to exclusive-or them together to recover the key stream. This key stream can then be exclusive-ored with an entirely different plaintext to produce a forged ciphertext. This forged ciphertext will be converted into the forged plaintext when fed into the intercept processor.

Based on this argument, juries may well be persuaded to disregard evidence from EES intercepted telephone calls. As the stated purpose of EES is to aid law-enforcement, this is a serious blow to its credibility.

Alternatively, all this "cryptographic magic" may convince a jury that EES provides absolute proof that a person made a particular call. This carries a grave risk of innocent people being convicted on the basis of falsified evidence.

Note that access to the escrowed keys is *not* needed to make a forgery. Anyone with access to the output from an intercept processor can make such a forgery, regardless of whatever physical and procedural controls are used to restrict access to the actual keys.

7.2 Masquerade

Another attack on the EES protocols has been found by Moti Yung and Yair Frankel. Their attack has some similarities with the forgery attack described in the previous section, in that it shows the need for authentication as well as confidentiality in an escrow system. Their attack proceeds as follows. Suppose Alice initiates a conversation with Bob, and Bob later initiates a conversation with Chris. If Bob gets to choose his session key (KS) with Chris, then Bob can re-use the session key and IV he shared with Alice. Furthermore, Bob can give Chris a copy of Alice's LEAF, rather than a new LEAF created by Bob. This has two unfortunate consequences:

- If they only have a wiretap warrant for Bob (and not Alice or Chris), the FBI may experience a certain amount of difficulty in tapping the call between Bob and Alice; the enciphered session key inside the LEAF Bob sends will not be decipherable with Bob's escrowed unit key.
- Bob may be able to incriminate Alice. Bob (knowing that Chris's phone is tapped) could say something incriminating in his call to Chris. When the FBI decipher the LEAF Bob sent (to determine who called Chris), they will find Alice's Device Identifier. Furthermore, when they have obtained a wiretap warrant for Alice and an escrowed copy of Alice's unit key, they will discover

that decipherment of the contents of the LEAF with Alice's unit key yields the session key, and that decipherment of the message with this session key yields the incriminating plaintext. This can then be used to convince a jury that Alice is guilty.
- Alternatively, Alice actually makes the incriminating telephone calls to Chris. When Alice is caught and prosecuted, Alice's lawyer claims that Alice is an innocent victim of an attack carried out by Bob (as described above).

While the forgery attack can be prevented by using a different mode of operation, this attack works against all modes. However, the key management system used to exchange session keys can be chosen so as to prevent this attack (e.g. by preventing either party from choosing what the session key will be). The Escrowed Encryption Standard (FIPS 185) makes no mention of how session keys are to be exchanged, so a manufacturer can make a product which is vulnerable to this attack whilst still being compliant with FIPS 185.

7.3 The Role of the IV

Suppose that Alice and Bob frequently talk to each other using EES telephones, and that their conversations are reasonably short (a few minutes, say). Furthermore, suppose that Alice wishes to defeat the key escrow mechanism without Bob knowing. To do this, Alice can build a modified EES telephone that uses Matt Blaze's LEAF-forging attack [3]. This attack has the disadvantage that it takes Alice a long time to forge a LEAF for each call — 40 minutes with a Tessera card, and even longer with a Fortezza card.

Alice might try to optimise this attack by computing a forged LEAF once, and re-using this forged LEAF for each of her telephone conversations with Bob. What are the problems with this optimisation?

If she uses the same key too often, Alice increases her vulnerability to attacks based on differential or linear cryptanalysis. However, if SKIPJACK is safe against these attacks for one long phone call, then it is also safe against these attacks for a series of short phone calls with the same key. So this isn't a problem.

It's also true that if this re-used session key is ever compromised (e.g. by physical means, such as a bug inside Bob's telephone), then *all* of Alice and Bob's conversations are compromised. However, it is a design assumption of EES that the FBI can no longer afford to do this sort of close-in monitoring, and so Alice feels safe from this too.

What really stops Alice from using this optimisation is that the Initialisation Vector affects the value of the LEAF. To re-use a forged LEAF, Alice must re-use the Initialisation Vector as well as the session key, as these are both cryptographically bound to the LEAF. Re-using the same key and IV renders Alice vulnerable to easy cryptanalytical attacks, particularly as the telephones use

OFB mode for encipherment. The exclusive-or of two ciphertext conversations will be the exclusive-or of the plaintexts (the keystream cancels out), and there is enough redundancy in digitised speech to enable both plaintexts to be recovered from their exclusive-or.

Note that this is very dependent on the mode of operation that is used. If CBC mode was used instead of OFB mode, Alice could successfully re-use a LEAF. To do this, Alice would tell Bob the same IV each time, but would actually use a different IV. As CBC mode is resynchronising, this works; Bob hears a few milliseconds of noise at the start of a call (too little to arouse suspicion), and then the call proceeds normally. OFB mode is not resynchronising. If Alice tries this trick with OFB mode, Bob will hear nothing but static.

Hence, we observe that tying the IV to the LEAF increases the security of the system against attempts to use unescrowed keys; but that this additional protection is only obtained with some modes of operation (e.g. OFB mode). However, OFB mode was a bad choice of mode for reasons explained in the previous section. We conclude that a version of EES which worked (in so far as such a thing is possible at all!) would have had to use a mode of operation which supports non-repudiation and isn't resynchronising.

Acknowledgements

I would like to thank Steve Kent for explaining the physical tamper-proofing used by the Clipper Chip; Dorothy Denning for providing additional information on the experiments carried out by the SKIPJACK Review Panel; Matt Blaze for providing additional information on his experiments with the Tessera card; and Robert Morris for practical advice.

References

1. American Bankers' Association. *ANSI X9.17-1985: Financial Institution Key Management (Wholesale)*, 1985.
2. M. Blaze. Personal communication, December 1994.
3. M. Blaze. Protocol failure in the escrowed encryption standard. In *Second ACM Conference on Computer and Communications Security*, pages 59 – 67, November 1994.
4. E. F. Brickell, D. E. Denning, S. T. Kent, D. P. Maher, and W. Tuchman. *SKIPJACK Review — Interim Report — The SKIPJACK Algorithm*, June 1993.
5. D. E. Denning. Personal communication, August 1993.
6. D. E. Denning. *The Clipper Chip : A Technical Summary*, April 1993.
7. D. E. Denning and M. Smid. Key escrowing today. *IEEE Communications Magazine*, 32(9), September 1994.
8. D. Johnson, S. Matyas, A. Le, and J. Wilkins. Design of the commercial data masking facility data privacy algorithm. In *Proceedings of the first ACM Conference on Computer and Communications Security*, November 1993.

9. S. Landau. *Codes, Keys and Conflicts: Issues in U.S. Crypto Policy.* ACM U.S. Public Policy Committee, June 1994.
10. C. H. Meyer and S. M. Matyas. *Cryptography: a new dimension in computer data security.* John Wiley and Sons, 1982.
11. National Bureau of Standards. *Federal Information Processing Standard — Publication 46: Data Encryption Standard,* 1977.
12. National Bureau of Standards. *Federal Information Processing Standard — Publication 81: DES Modes of Operation,* 1977.
13. M. Roe and M. Lomas. Forging a clipper message. *Communications of the ACM,* 37(12):12, December 1994.
14. William Safire. Sink the clipper chip. In *New York Times,* 14th February 1994.
15. (UK) Government Communications Headquarters. *Review of DEA-1,* October 1985.
16. (U.S.) National Institute of Standards and Technology. *Federal Information Processing Standards Publication 185 — Specifications for the Escrowed Encryption Standard,* February 1994.
17. Michael Wiener and Paul van Oorschot. Parallel collision search with application to hash functions and discrete logarithms. In *Second ACM Conference on Computer and Communications Security,* pages 210 – 217, November 1994.