DOI 10.5162/ettc2022/6.1

Encryption Techniques for Test Data

Malcolm Weir[1]
[1] Ampex Data Systems Corporation, Hayward, CA, USA,
mweir@ampex.com

Abstract:
The ubiquitous need for *data security / information assurance* in test applications is no longer remotely controversial. But there exists significant confusion in the marketplace regarding the virtues and values of the various techniques and technologies. This paper discusses the standards and certifications used to protect data and explores the cost/benefit analysis of various approaches, including both traditional government-sponsored devices and commercial alternatives.

There are many overlapping *standards* for encryption, both for data in transit and data at rest. While this paper mostly focusses on data at rest applications, there are many commonalities with data communications security problems.

A key source of market confusion is the overlapping terms and acronyms, including AES, FIPS, Common Criteria, etc. This paper seeks to clarify these labels and explores the suitability of the various approaches to specific test and telemetry requirements.

Key words: Encryption, information assurance, standards, data at rest, data in transit.

Background

Data encryption has always been a desired feature for a subset of test applications, especially for military applications. Historically, when a program requires that data be encrypted, the supplier and the customer / end user are engaged in the provision of the appropriate technology e.g. “Government Off The Shelf” (GOTS) encryptors.

This is now changing in two key regards: first, for reasons that will be discussed later, encryption has become the “best practice” for all applications, regardless of who the end user happens to be, and secondly the suitability of GOTS solutions is not universal, again for a number of reasons.

With the increased demand for these technologies, there is an increased level of confusion in the industry as to what, precisely, is available and required to meet the needs of a program. This confusion leads to the critical problem that they very technology that is being deployed to protect data may not, in fact, be well suited to the task.

This paper attempts to demystify some of the terminology and processes around data encryption standards.

Introduction

It is sometimes noted that it’s very easy to invent a cipher. The only challenges are (1) to ensure that it can run in the hardware resources available at the performance required, and (2) to prove it isn’t easily cracked (defined as deciphered without the appropriate key or keys).

Obviously, those two factors are linked: there’s no point in having an uncrackable cipher that takes too long to use, nor having an extremely efficient but weak one. But as a general rule, consumers of this technology care most about the cipher’s “strength”[1], and work around the performance issues. This paper is mostly concerned with issues of “strength”.

Why Encrypt?

The benefits of encrypting data sets can be described as confidentiality, authenticity, and sometimes integrity. The first is obvious: without the key, unauthorized access requires cracking the cipher or exhaustively trying keys, both of which are (hopefully) hard. But the information

[1] Technically, in this context “strength” is a function of key size and algorithm, but common usage tends to ignore the former on the basis that comparisons assume equivalent key lengths, and thus the algorithm is the determining factor.

assurance benefits of encryption are sometimes overlooked: because only those with the correct key can create the expected data sets, accidentally or maliciously falsified data cannot be introduced between the source and destination. And integrity – the assurance that chunks of the data set are not missing or corrupted, for example – can come about in the much the same way by using encryption schemes that "chain" blocks together, so a single corrupted word will cause a ripple effect that will be easier to detect. Of the three benefits, confidentiality has been the driving rationale for test data sets.

Traditionally, the apparent need for encryption was frequently avoided by the use of physical security – vaults of tapes and dedicated, isolated telephone circuits. But by the mid-1970s, the financial services sector had started to implement distributed systems (for cash dispenser devices / "Automated Teller Machines"); these networks required technical means to make eavesdropping unproductive.

And as the world became more connected and more digital, the consequences of data leaks became more significant. Previously, the data on a misplaced reel of tape was reasonably safe from disclosure, because of the rarity of suitable drives to read the media as well as the obscurity of most data formats meant it unlikely that accidental loss would represent a significant risk. Today, though, an accidentally lost dataset or a malicious exfiltrated one are quite likely to end up in "the wrong hands", which has lead to a recognition that data previously thought mundane (such as accounting or stock control records) or data that is confidential but of very limited interest (e.g. health information) may have economic value on a "black market" sale.

As the (real or imagined) value of purloined (or misplaced) data has grown, so has the default posture: it is now "best practice" to assume that data will leak, and therefore the appropriate posture is to encrypt everything to protect the underlying information.

General Types of Encryption

Broadly speaking, encryption applications relevant to test data can be broken down into two types: data-at-rest (DaR), and data-in-transit (DiT). In terms of encryption technology, the differences can sometimes be ignored, although the implementations have different characteristics and requirements. For example, with a DaR implementation, an attacker may be assumed to have a large volume of ciphertext to work against, while with a DiT system attention must be given to prevent compromising the system with "side-band" leaks (such as Radio Frequency leaks or power consumption monitoring).

As well as DaR and DiT applications, encryption algorithms can be summarized as either symmetric or asymmetric; the former is defined as using the same key to encrypt as you use for decryption, while the latter uses different keys for each operation. The vast majority of DaR applications use symmetric algorithms, while the modern internet is based on asymmetric protocols (HTTPS, SSL, etc). (The principle virtue of asymmetric algorithms is that it allows for a model where the sender and receiver need not know each other's secrets, so, for example, members of the public can communicate securely with an online store without having to be given the store's private encryption key).

While asymmetric ciphers are exceedingly useful, most test applications do not require them; about the only obvious exception are remote "phone home" telemetry systems, where widely deployed devices report to a central "mothership". While the following sections focus on symmetric "block" ciphers (as required by DaR, but also applicable to DiT), the use of asymmetric algorithms should not be completely ignored.

The First Encryption Standard

With the possible exception of Julius Caesar's cipher, the DES standard was the first openly published encryption standard. Invented by IBM and codified as the US National Institute for Standards and Technology's (NIST's) Federal Information Processing Standard (FIPS) Publication 46 (FIPS 46) in 1977, it has since been withdrawn because it is too insecure for contemporary use: with only a 56-bit key, all possible keys can be tried in a reasonable time. It also illustrates a risk in algorithm design: the rationale for some of the design decisions was opaque, leading to suspicions that the algorithm was built with a 'backdoor' known only to the designers.

One of the criticisms of the standard is that it is explicitly forbidden for use with classified information, fueling the suspicions of a backdoor – a deliberate vulnerability that would allow the National Security Agency (NSA) to read any encrypted material. It is now believed that, in fact, there was no backdoor -- to the contrary, the reason for seemingly nefarious decisions about its architecture was to protect it against a new type of attack threat which the government cryptanalysts understood but which was not at the time common knowledge – although the US Government did indeed weaken the algorithm by reducing the key size from 64 bits to 56.

To compensate for the short key size in DES, the 3rd version of the FIPS standard (FIPS 46-3 [1]) was released in 1999. This introduced a method whereby the data was looped through the DES algorithm three times, each time with a different key, making the overall key length 168 bits. This might appear to triple the strength of the encryption, but in fact, due to a specific type of cryptographic attack, it really only doubles the strength.

While technologically obsolete, Triple DES as TDEA is also known is still approved for protecting sensitive but unclassified US government data (although not recommended for new applications!).

The Advanced Encryption Standard (AES)

The workhorse of the current state-of-the-art in encryption is AES. It was the result of an open, international contest conducted by the NIST between 1997-2001. Fifteen algorithm designs were considered and evaluated for cryptographic security and performance in a variety of implementations (software, FPGA, and so on). Three technical conferences were held in the USA and Europe, involving more than 180 people from 23 countries, and featured voting by the cryptographers on the candidate algorithms.

The winner of this process was an algorithm designed by two Belgian cryptographers, a subset of their Rijndael family of ciphers – the name is derived from the surnames of the inventors. Three Rijndael ciphers make up the AES standard. Each takes a block of 128 bits of data and uses key lengths of 128, 192 and 256 bits, respectively. In the twenty years since it was published, the best technique for breaking into it has improved the number of keys one needs to "brute force" from 2256 to "just" 2252, so this is an interesting result, but not particularly useful for reading the ciphertext.

AES is the only block cipher in the Commercial National Security Algorithm Suite (which replaced the former "Suite B") list. As such it is listed by the NSA as being suitable, when used with 256-bit keys, to protect up to TOP SECRET information (although this list says nothing about the implementation, only the algorithm; see the next section on FIPS 197 [2]).

All AES algorithms operate on a block of 128 bits, that is 16 bytes; to encrypt larger quantities of data, the algorithm must be applied to successive 16-byte pieces. This creates a new problem: if you use the same key for each piece, then the output ciphertext will show where those 16-byte pieces are duplicated (even if you don't know what the plain text actually is), which can provide a lot of information about the plaintext. To address this, AES features several "modes", the simplest of which is the "electronic code book" (ECB) mode, where you use the same key for each block. This can yield the unfortunate results shown in the picture in Figure 1. A straightforward enhancement is to include a counter merged into the process, so that each block is encrypted with different settings (the AES CTR mode). Other modes use different ways to perturb the process; which one is "best" depends entirely on the application. For data at rest encryption two modes are often used: CBC (Cipher Block Chaining) and XTS (Xor–encrypt–Xor-based tweaked-codebook mode with ciphertext stealing), the latter using two equal-sized keys[2].

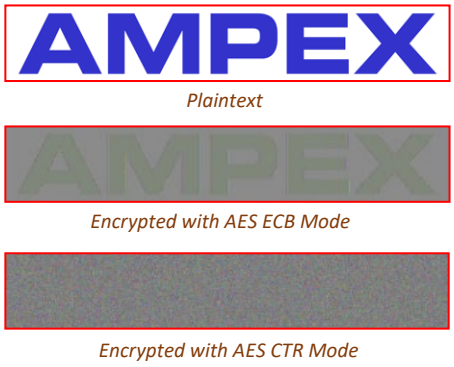

Figure 1 AES Modes

Compared to other algorithms, one significant advantage of AES is that modern CPUs often contain either special instructions or complete special-purpose subsystems to perform (or assist with) the operation. Examples include Intel's "AES-NI" instructions and NXP's "AES Execution Unit". These hardware features not only improve performance but make it harder for an attacker to perform "side-channel" or "timing" attacks where information about the process can be deduced by monitoring the operation: the special hardware subsystems act as "black boxes" which conceal the details of the operation.

It should be remembered that these modes are mechanisms for applying the AES algorithm, which itself always remains the same: a "black box" that takes the same 128 bits of data and a

[2] *The use of two keys is not to increase the strength of the encryption per se, rather it makes it very hard to extract any information even given a huge amount of encrypted data. Using the two 256-bit keys provides no more than the strength of one 257-bit key: if it takes a certain time to try 2^{256} possibilities to get the first key, then it will take the same amount of time to guess the second, and $2^{256} + 2^{256} = 2^{257}$!*

key of 128, 192 or 256 bits in length and produces a 128-bit encrypted output.

FIPS Publication 197 (FIPS 197)

The document that codifies AES encryption – that is, defines the algorithm is FIPS 197. Related to the publication, certifications assuring compliance with FIPS 197 are issued via the Cryptographic Algorithm Validation Program (CAVP). While it is possible to implement an AES architecture, one can only be deemed "certified" after having gone through the CAVP process and awarded a FIPS 197 certificate.

While FIPS 197 improved upon its predecessor, DES, it is likely the FIPS 197 standard will be updated in the future with changes in how AES is used or implemented to increase the strength of the protection. This is not unlike how FIPS 46-3 introduced Triple DES.

In normal usage, when an algorithm is referred to as FIPS 197, the implication is that the algorithm has been certified by an accredited laboratory to conform to the standard. NIST has a Cryptographic Algorithm Validation Program (CAVP) that defines the test suite that an implementation of AES must pass, and then the certified implementation will be recorded on the NIST website. So, for example, certificate "AES 2408" was issued to Intelliprop, Inc. for their AES-XTS implementation (an FPGA core), and the NIST website indicates their implementation is FIPS 197 certified for key lengths of 128 and 256 bits (but evidently not 192 bits).

In terms of a hierarchy of quality, it is perfectly possible to have an AES implementation that works and is functionally correct, but without a NIST certification one cannot be objectively confident of that correctness; in effect, the certification is objective proof that the implementation is a correct interpretation of the standard.

FIPS Publication 140 (FIPS 140, FIPS 140-2, FIPS 140-3)[3]

Complimentary to FIPS 197, FIPS Publication 140 "Security Requirements for Cryptographic Modules" (FIPS 140) [3] covers the pieces surrounding the actual encryption. When working together as sub-component of a system, these pieces are often referred to as a "module" and consists of things like a micro-controller, encryptors, and a supporting storage on the same circuit board, or a software library with clearly defined interfaces.

In this broader context, a FIPS 140 module includes some very straightforward concepts and some more abstract ideas. The straightforward concepts include physical security requirements like "how can attempts to physically interfere with the module be detected?" and "how can accidental or deliberate interference result in the module 'failing safe' and refusing to function?" The more abstract ideas are things like the characterization of the interfaces into the module and the functional roles, services and authentication provided. In general, these concepts pull in other standards, so while the FIPS 140 document is quite short, by the time the rest has been incorporated, it becomes a very extensive standard.

To understand this better, a brief description of how encryption modules tend to be architected is in order. It is common to consider the DaR encryption on storage devices as matching the illustration in Figure 2: unencrypted ("plaintext") data is fed into an algorithm, together with a suitable key, and the resulting encrypted data is stored on the device. As similar approach can of course be used for DiT.

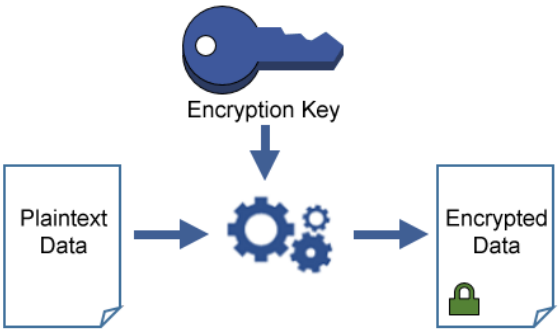

Figure 2 Simple Encryption

This is, of course, perfectly functional, and usable, but it has limitations; probably most significant of those is the fact that one, and only one, key can decrypt the data. This may sound like a good idea, until one realizes that it means that every authorized user must have that one single key, which becomes problematic when there's a need to revoke access to just one of those users or the key gets compromised: the only option is to decrypt all the data with the original key and then re-encrypt it with a new one.

[3] Note: the "dash" after a FIPS publication number is the major version indicator, so "FIPS 140-2" is the se9cond major version of the standard, and "140-3" is the third. There are also minor revisions within version, so there are three revisions of FIPS 140-2, usually indicated by the publication date.

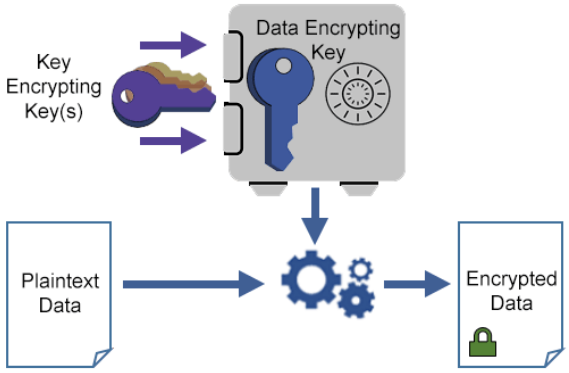

Figure 3 Key Encrypting Keys

But there is a straightforward alternative: instead of the user passing in the key used to encrypt the data, they pass in a key that is used to unlock the key that's protecting it, as illustrated in Figure 3. In this arrangement, multiple copies of the "data encrypting key" (DEK) can be stored on the storage device, with each copy encrypted with its own "key encrypting key" (KEK). Any one of the KEKs can be used to access a copy of the DEK, and the DEK is then used to access the data.

In this way, if one KEK gets compromised or lost, an administrator can simply erase the copy of the DEK that is encrypted with that particular KEK, leaving the rest untouched. And of course, an outdated KEK can be updated by simply decrypting the DEK using the old KEK, and then re-encrypting it using a new one – a simple operation involving a few bytes instead of gigabytes!

An additional benefit of this approach is that the DEK – the key that is protecting the user's data – need never leave the storage device. This means that "multi-factor" arrangements, in which you need two or more distinct and independent KEKs to unlock the DEK, and each KEK is provided by a different mechanism or "factor", can be crafted so that unless both factors (i.e., both KEKs) are simultaneously compromised, the data remains safe – and the pieces only come together within the controlled environment of the storage device protecting the integrity of the separate "factors"!

A slight variant of this idea is that, instead of the DEK being stored in the storage device, one or more KEKs are, and then the encrypted DEK is provided to be decrypted by the KEK(s) within the device. This approach ensures that the storage device cannot be "tricked" into giving up the DEK (as it simply doesn't have it), yet security is preserved as the DEK isn't usable unless it's "correctly" encrypted by the KEK(s) in the storage device.

So, getting back to FIPS 140: where a KEK/DEK architecture (Figure 3) is being used, the DEK must be generated using an appropriately random "deterministic random number generator" (DRNG) algorithm, an approved list of which is provided (FIPS 140, Annex C, which calls out NIST SP800-90B [4], amongst others). Next the approved modes of AES operation are detailed (NIST SP800-38E [5], for example, defining how AES-XTS must be used).

Then there are algorithms (key derivation functions, KDF's) that convert weak (human-grade) passwords into acceptably strong keys. The idea here is that although it is straightforward to "brute force" a short text password, it is also easy to lock the module after a certain number of failed attempts – and that lockdown can be either temporary, preventing additional efforts for some period, or irretrievably permanent, by destroying the hidden DEK. Instead, if the password is first converted into a strong key, then backdoor attacks that nullify the password-checking logic (i.e. make any password appear to be "correct") will be useless: one would still need the key that was created by applying the password together with some fixed (but hidden) constants through the KDF algorithm.

There is also another very significant capability associated with a FIPS 140 certification: the use of "message authentication" services ("HMAC": Keyed-Hash Message Authentication Code). These provide a mechanism by which the system can validate that a particular arbitrary-sized stream of bytes (a "message") has not been tampered with. The fundamental idea is that one can use a "hash algorithm" (e.g., "SHA-256") that creates a checksum of the message, and then cryptographically signs it using the private part of a public key cryptography key pair, so that any change to the message will change the checksum and it is impossible to update the signature without the secret, private key. This approach is used in the Ampex TSEM FIPS 140 module to validate the contents of the memories storing the FIPS 197 FPGA bitstream using firmware code in a secure microcontroller that is itself "signed" in the same way.

Both the hash algorithm and the HMAC are defined by their own FIPS standards: FIPS 180-4 "Secure Hash Standard (SHS)" [6] and FIPS 186-4 "Digital Signature Standard (DSS)" [7], respectively. And the digital signature standard defines the specifics of the acceptable public key cryptography schemes.

Since "one size rarely fits all", FIPS 140 defines levels of increasing security, with "Level 3" is theoretically more secure than "Level 2". It is important to note that these levels are not related

to the version “dash” number, so that it is appropriate to comment that FIPS 140-2 Level 2 is the most common certified level! As is usually the case with these types of collections of disparate requirements, many implementations might qualify for a higher level in some areas but are “held back” in others. So, for example, the logical protection mechanisms (e.g., code-signing) might warrant a higher level, but the physical anti-tamper protections might not. It should be noted that some anti-tamper methods are reasonably easy to achieve but impose consequences for the product development; Level 3 physical security can be achieved by “potting” all the hardware in epoxy!

One important characteristic of a module’s certification is how and where the boundary between it and the rest of the system is drawn. A very tightly drawn boundary reduces the elements that must be certified (and so potentially reduces the overall security / value of the module), while a broad brush will include pieces (of software, usually) that become subject to the restrictions of certification, so that any updates to that software will require updating the certificate.

A cautionary tale as to why FIPS 140 certification (or equivalent) can be valuable comes from security researchers who created attacks that defeated the encryption on several non-FIPS commodity storage devices (Meijer & van Gastel, 2019 [8]). Their attacks included loading modified firmware into the target devices, having instrumented them to identify how the firmware was intended to work. So, for example, they created firmware that would always believe the supplied password was correct, no matter what. It is possibly tempting fate to assert that, had the drives been FIPS 140 certified, their attacks would have failed, but it is certainly true that it would have been much harder to gain access to the data.

Common Criteria Certification (CC, NIAP)

While the FIPS 140 certification process provides assurance of a solid solution for many applications, it is intrinsically an American (USA and Canada) framework. To put another way, the only authorities issuing certificates are the US and the Canadian governments.

This introduces obvious issues for non-American applications: does using the FIPS approach implicate exportability (ITAR, etc)? How can a non-American user (particularly sovereign users) be assured that there was no interference with the evaluation? (And regardless of the likelihood of that happening, the issue is the ability to assert that it could not have happened; certification is always trying for absolute assurance, not just reasonable conclusions).

The solution for both American and non-American users lies with the Common Criteria for Information Technology Security Evaluation (referred to as Common Criteria or CC). This is an international framework for providing security certification a system. In the USA, the responsible body for CC efforts is the National Information Assurance Partnership (NIAP), which is operated by the NSA.

The international members of the framework are (currently) a total of 31 countries, slightly more than half of which are “certificate producers” with the rest being “certificate consumers”. Producer nations run a full scheme including certifying labs to evaluate products. Consumer nations agree to accept certificates from the “producing” nations. A program based in a “consumer” country that wants a certified product would simply outsource the certification to a producer nation (Indonesia outsourcing to Australia, for example).

Common Criteria stands in contrast with FIPS 140, as the latter is concerned solely with cryptographic systems, while CC can be applied to any type of system. The two schemes are very closely related, and indeed up until 140-2 the FIPS standard explicitly called out requirements from the Common Criteria standard (those requirements haven’t gone away but are now separately and explicitly listed in FIPS 140-3).

Common Criteria is concerned with the security functions of a product as a whole, which obviously includes cryptography (overlapping with FIPS 140), auditing and logging, access controls, administrative roles, and so on. For a data storage device, there is a lot of commonality between the CC DaR and the FIPS 140 requirements, but enough variability (e.g., on the drawing of a FIPS 140 boundary) to keep CC separate from the FIPS certification.

For DaR applications, there are five potentially relevant CC protection profiles (PP): two for full-disk encryption, two for file-based encryption, and one for USB flash drives. The two full-disk encryption protection profiles boast a “collaborative” tag – they are collaborative PPs (cPPs), not just PPs – indicating that they’ve been developed with a larger group of contributors than just the US government. The two full-disk cPPs are for the “Encryption Engine” (the module that does the encryption) and then

for the "Authorization Acquisition", which handles key management.[4]

The Encryption Engine cPP [9] defines how the data must be encrypted; slightly bizarrely, it references the ISO/IEC standard (18033-3) [10] for AES rather than FIPS 197, even though the NSA's CNSA list references the latter! The Authorization Acquisition cPP [11] is significant mostly because it is an entirely separate standard, allowing the two functions to be separated and even provided by two distinct suppliers.

One of the ramifications of the CC authentication certification for DaR is that it *must* contain the totality of the key lifecycle, from key generation through key transport to loading the key into the encryption engine. This leads to the somewhat paradoxical situation that a gold-standard, NSA-generated secret key *cannot* be used in CC (or in related standards, such as CSfC).

In the context of DaR and DiT, a standalone CC certification is rarely required, as FIPS 140 provides a similar level of assurance in a more narrowly focused standard; of course, when dealing with other security accreditations, CC is more commonly mandated. The value of the CC DaR/DiT certification is not inherent in the validation itself, but because the "next layer" (e.g. CSfC, see below) uses CC certificates as building blocks to practical, approved solutions.

Commercial Solutions for Classified (CSfC)

CSfC is a program run by the NSA which uses a pair of layered, Common Criteria certified encryption products to create a solution that may be used to secure National Security Information. The stated rationale for using two products, with the second encrypting the output of the first, is that this mitigates deficiencies that might exist in the implementation of either. However, one might recall the aforementioned "triple DES" exists to provide a security boost over regular DES, so it is not unreasonable to assume that the layering provides some additional security, even if that isn't the public rationale for it.

Unlike FIPS 140 and CC in isolation, CSfC is specifically designed to secure data at the levels needed for the most sensitive of information and is recognized by the US government for that purpose; unlike "Type 1", it is also designed for non-governmental use, such as by finance or healthcare organizations.

CSfC provides requirements for solutions via Capability Packages (CPs). The "Data-at-Rest CP" [12] defines several implementation architectures, such as a software layer on top of a hardware one, or two full-disk software layers, or file-based software on top of full-disk, and so on. The most recent version of the DaR CP also supports solutions using two hardware designs.

To help ensure that the same vulnerability does not exist in both layers, the CSfC philosophy requires that each layer must be produced by different vendors (or, in the case of corporate mergers and acquisitions, demonstrably different teams within the same company). Under this principle of diversity, there must be at least two, and possibly three, organizations involved in a CSfC solution: one each to produce the encryption implementations, and optionally a third to serve as an integrator of the other two.

From a functional standpoint, CSfC solutions are as good as the traditional US Government "Type 1" approach, but with significantly increased versatility. First, CSfC implementations are not "controlled cryptographic items (CCI)", which facilitates (and reduces the cost of) logistics and handling and particularly international/export applications. Second, key handling concepts can be tailored to the specific application and mission requirements.

Key handling with legacy "Type 1" systems is based entirely on the NSA's "one size fits all" model: all keys are generated by the NSA, and distributed through the appropriate secure channels, before being loaded into the encryption device, typically using a "Secure Key Loader". This approach is fine when being used within the US/NSA sphere of influence, it is naturally impractical for commercial and sovereign international customers.

By contrast, with a CSfC device, keys must be "organically" created within the device (or the ecosystem for the device). This inherently creates significant flexibility for the design of mechanisms to deliver keys to the encryptors.

[4] Prior to 2010, CC evaluated products according to Evaluation Assurance Levels (EALs), of which there are 7. The lower four (EAL1 through EAL4) were process-based evaluations, meaning most any system could be evaluated and certified whether or not it met a particular function or purpose (which were termed "robustness" evaluations). While this obviously has value, it is at odds with the sponsoring governments' goal of qualifying functionally similar products in a comparable and repeatable manner. The revised approach (since 2010) substitutes the old robustness evaluations with strict compliance with defined protection profiles for the lower four levels, and then retained the semiformal and formal design analysis for the upper levels only once a protection profile has been validated.

Over a longer term, by contrast with “Type 1” solutions, CSfC requires periodic recertification through the NSA (or equivalent), and there is always the possibility that, during that process, new or modified requirements may become mandated with possible budget implications. While this may appear a significant and justifiable concern, real-world cybersecurity – obviously including cryptography – demands regular software / firmware updates to protect against newly identified vulnerabilities. The old model, where a single product certification survives the life of the program, cannot be sustained in the contemporary “connected world”; the well-publicized attacks on certain Intel CPUs (“Meltdown”[5] and “Spectre”[6]) are examples of unexpected problems that cannot be ignored.

The value of the CSfC program can be summarized by the following: at least one foreign government has used the CSfC “recipe” (that is, the CP) together with products locally certified to Common Criteria PP standards.

The integrity of CSfC can be illustrated by noting that, using a properly certified and structured CSfC solution, the NSA will approve the use of industry standard WiFi network and internet bridges to carry US Top Secret information.

Test Data Applications

Several features common to many test applications lend themselves to the use of FIPS or Common Criteria.

First, test articles are frequently heavily constrained in terms of Size, Weight and Power (SWaP). This can pose insurmountable challenges with integrating GOTS / legacy devices with the required capabilities. By contrast, FIPS/CC solutions can include software implementations, which can be scaled to fit the physical constraints of the application.

Second, test applications tend to involve numerically small numbers of systems: the largest pools of flight test recorders number in the scale of a few dozen units, which makes the time and expense of a “from the ground up” encryption solution much harder to justify. But if the virtues of certification can be obtained by judicious selection of key components (e.g. by using a FIPS 140 certified SSD in place of an uncertified one), then the cost differential becomes marginal and the schedule impact trivial.

Third, it is sometimes said (partially in jest) that “encryption is easy, but key handling is a challenge”. Using GOTS solutions or similar tends to include a rigid key handling policy, deviation from which (e.g. to using “test keys”) has unknown consequences: if the test keys all have the form ‘123456’, then it’s fair to say that security will be compromised! More seriously, because open certification processes like those of FIPS and Common Criteria allow system designers to make informed, intelligent decisions about the consequences of changes to the expected application. As an example, it is possible that a designer of a DiT solution might conclude that only FIPS 197 is required on the test article, with the other parts that would make up a FIPS 140 system being distributed to ground-support equipment or other certification efforts.

Conclusion

With the maturation of programs like FIPS and Common Criteria, the commercial and international market can have confidence that "government quality” (i.e. US Government quality) solutions can be implemented without recourse in cost, confidentiality, or schedule to independent developers and integrators, and end users can have confidence that their solutions really do provide the features and benefits that they expect.

References

[1] National Institute of Standards and Technology,FIPS PUB 46-3 “Data Encryption Standard (DES)”, 25 October 1999. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf

[2] National Institute of Standards and Technology,FIPS PUB 197 “Advanced Encryption Standard (AES)”, 26 November 2001. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.197

[3] National Institute of Standards and Technology,FIPS PUB 140-3 “Security Requirements for Cryptographic Modules”, 22 March 2019. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.140-3

[4] M. Turan, E. Barker, J. Kelsey, K. McKay, M. Baish and M. Boyle,SP 800-90B “Recommendation for the Entropy Sources Used for Random Bit Generation”, January 2018. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-90B

[5] CVE-2017-5754 is the official reference to Meltdown. CVE is the Standard for Information Security Vulnerability Names maintained by MITRE.

[6] CVE-2017-5753 and CVE-2017-5715 are the official references to Spectre. CVE is the Standard for Information Security Vulnerability Names maintained by MITRE.

[5] M. Dworkin,SP 800-38E “Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices”, January 2010. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-38E

[6] National Institute of Standards and Technology,FIPS PUB 180-4 “Secure Hash Standard (SHS)”, August 2015. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.180-4

[7] National Institute of Standards and Technology,FIPS PUB 186-4 “Digital Signature Standard (DSS)”, July 2013. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.186-4

[8] C. Meijer and B. van Gastel, “Self-Encrypting Deception: Weaknesses in the Encryption of Solid State Drives”, in *IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 2019. Available: https://doi.org/10.1109/SP.2019.00088

[9] National Information Assurance Partnership, “collaborative Protection Profile for Full Drive Encryption - Encryption Engine”, 1 February 2019. [Online]. Available: https://www.niap-ccevs.org/MMO/PP/CPP_FDE_EE_V2.0E.pdf

[10] International Standards Organization,ISO/IEC 18033-3:2005 “Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers”, July 2005. [Online]. Available: https://www.iso.org/standard/37972.html

[11] National Information Assurance Partnership, “collaborative Protection Profile for Full Drive Encryption - Authorization Acquisition”, 1 February 2019. [Online]. Available: https://www.niap-ccevs.org/MMO/PP/CPP_FDE_AA_V2.0E.pdf

[12] National Security Agency Central Security Service, „Data-at-Rest Capability Package V5.0“,18 November 2020. [Online]. Available: https://www.nsa.gov/Portals/75/documents/resources/everyone/csfc/capability-packages/Data-at-Rest%20Capability%20Package%20v5.0.pdf