News from the CryptoCards Team

Welcome to the first edition of the IBM CryptoCards Newsletter. This edition discusses the latest news from the CryptoCards software/firmware development team.

CCA Release 5.4.33

Available for 4767 customers - Linux or Windows

The latest release of CCA, 5.4.33, is now available for IBM 4767-002 on x86 systems running RHEL Server 7.5, SLES 12.3, or Windows 2016 Server.

Here is a summary of changes for CCA release 5.4.33:

- Three-key (192-bit) Triple-DES keys are added to strengthen security for operations such as data encryption, PIN processing, and key wrapping.
- Limited ISO Format 4 (ISO-4) AES PIN blocks as defined in the ISO 9564-1 standard.
- Directed keys, whose objective is to generate and derive many different AES key pairs with different key usages from one key diversification key (KDK).
- Wrapping and unwrapping DES and TDES keys using an AES Key Block Protection Key (TR-31 key block version ID, or method, "D") according to ISO 20038.

EP11 Release 2.0

Available for CEX5S and CEX6S customers

The latest release of EP11, 2.0, is now available for IBM CEX5S and CEX6S customers on IBM z13 or z14 servers. Supported operating systems include:

- RHEL Server (64-bit),
- SLES (64-bit): 11 SP4, 12, 12 SP1, 12 SP2, 12 SP3, and
- Ubuntu (64-bit): 16.04.05, 18.04.

Summary of changes for EP11 2.0:

- Adds new API for targeting cards and domains, allowing for unified target creation and target groups.
- Extends exported user interfaces in ep11.h and ep11adm.h.
- New EP11 TKE daemon. The daemon now implements an authentication method for the communication between TKE and daemon. The Linux user needs to be added to the ep11tke group to work on a TKE. This feature can be disabled per configuration option.
- Adds documentation to the EP11 structure document about the EP11 Support Program.

New software requirements:

- OpenSSL 1.0.x or 1.1.x is required for the new EP11 TKE daemon.

PCI PTS HSM Certification

IBM CEX6S / CCA 6.0 certification achieved

The IBM Crypto Express 6S (CEX6S) is IBM's fastest and most secure hardware security module (HSM). As of January 14, 2019, with IBM's Common Cryptographic Architecture (CCA) version 6.0, the CEX6S has achieved certification under the Payment Card Industry (PCI) PIN Transaction Security (PTS) HSM program.

The Payment Card Industry Data Security Standard (PCI DSS) requirements for protection of account data apply broadly to businesses in the financial services and retail banking industry and specifically require HSMs for protection of cryptographic keys that protect cardholder data (Requirement 3.5.3).

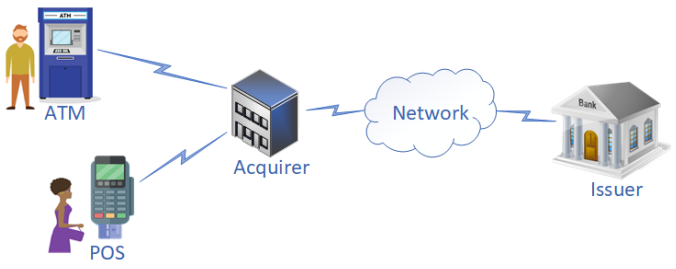
For more information, see the IBM Systems Z Security Roundtable Blog: http://ibmsystemsmag.com/blogs/z-security-roundtable/january-2019/ibm-cex6s/

Is 2-key Triple DES really broken for banking applications?

Today, most cryptographers recommend using the AES (Advanced Encryption Standard) algorithm when symmetric cryptography is required. AES is strong, well-vetted, and standardized by a number of respected bodies. However, many applications that have been using Triple-DES (TDES) continue to use that algorithm, because it will take years for them to fully migrate to AES. This includes the payment card systems used worldwide in the banking industry. The recommendation to stop using TDES – and the withdrawal of some of the related standards – has many people worried about security. But how real is the risk in this environment?

Why haven't they switched to AES yet?

Let's think about why it is so hard to quickly change these systems from TDES to AES. Consider how many different elements have to work together in the payments system. There are many different entities whose systems must work together: physical stores with point-of-sale (POS) terminals and systems, online merchants with web sites, ATM providers, acquirers who receive the transactions initiated by the customers, networks that route the transactions from the acquirers to the banks that issued the cards and hold the accounts, and those issuers themselves who must approve the transactions. There are other parties as well, who are involved indirectly: auditors, key management systems, security evaluation and certification companies, EMV card personalization service providers, and others.



Each of these companies has computer systems running software to perform their part of the transaction processing, as well as related functions like key management and secure backup. To further complicate matters, in today's world these systems are a mix of servers owned and operated by the companies themselves, and cloud systems that are operated by third parties. Each of the companies also has specialized hardware to secure the transaction processing and the cryptographic keys that are used. This includes Hardware Security Modules (HSMs), ATMs, POS terminals, EMV smart cards that are given to their customers, smart cards used in key management, specialized administrative and key management devices, and others. All of the software and hardware at all of these companies must work together in order to securely process transactions.

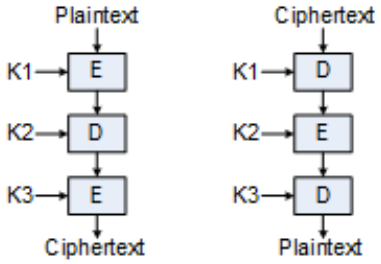
If one company in this system unilaterally decides to change from TDES to AES, the system stops working. They will not be able to work with the others who send cryptographically-protected data to them, nor with those who receive the protected data they create. The only way for such a change to work is if all parties who must communicate with each other make the change at the same time, switching from TDES to AES simultaneously. This is a very complex undertaking. It means revising and testing software, and either upgrading or buying new and expensive equipment. It means careful testing to make sure the parties' systems work together after all hardware and software are upgraded. It means carefully verifying that no security weaknesses have been introduced, particularly if some parts of a system use older algorithms while others use the newer ones.

What is the risk, until we stop using TDES?

Rest assured that the banking industry *is* in the process of working toward these algorithm changes, but it will take some time before there is a large-scale switch to AES. So, until then, how susceptible are these TDES-based systems to attack? That requires thinking a bit about the cryptography, and how it is used in payment applications.

There are two different types of Triple-DES, which differ in their cryptographic strength. The term "Triple-DES" means that the algorithm uses single-DES three times,

with three 8-byte DES[1] keys. This is shown in the figure below, where "E" means to encrypt using single-DES with an 8-byte key, and "D" means to decrypt in a similar manner. The left-hand part of the figure shows TDES encryption, while the right-hand part shows decryption.



With 3-key TDES, keys K1, K2, and K3 are each different, randomly-generated keys and the TDES key is the concatenation of these three keys. With 2-key TDES, keys K1 and K3 are *the same*, while key K2 is different. A 2-key TDES key is the concatenation of K1 and K2. As you would expect, 3-key TDES is cryptographically stronger than 2-key, but not by as much as you might expect. Payment systems most commonly use the 2-key type of TDES.

ISO, the International Organization for Standardization, has a good document which explains the strengths of 2-key and 3-key TDES. ISO TR 14742 is titled *Financial services — Recommendations on cryptographic algorithms and their use*. Regarding 2-key TDES, it says this:

> 2-key Triple DES … has effective strength $2^{\min(112,\ 120-t)}$ where 2^t is the number of plaintext ciphertext pairs available to an attacker.

The same formula can be found in the paper *On the security of 2-key triple DES* by Chris Mitchell. What we see here is that the strength of 2-key TDES is *highly dependent on the number of plaintext-ciphertext pairs that are known*. If you know a large number of unencrypted data input values (plaintext) and the corresponding encrypted data output values (ciphertext), then it is easier to attack the algorithm. However, if you think carefully about the way TDES is most commonly used in payment systems, you realize that there are generally *no* plaintext-ciphertext pairs, and thus the algorithm is still quite strong. Let's look at some examples.

PIN encryption: A customer's PIN (Personal Identification Number) is encrypted within the device where it is typed in. This is generally an ATM or a POS terminal in a store. These devices are required to meet the criteria for a Secure Cryptographic Device (SCD)[2], and the unencrypted PIN never appears outside of those secure hardware devices. Thus, the plaintext is never available to an attacker who is monitoring transactions on a network or has access to the host system where they are processed. In addition, there are no ways for an attacker to have their own chosen PIN values encrypted in order to create known plaintext-ciphertext pairs.

Key Management: This is the process of handling cryptographic keys: generation, storage, import and export, and other processes. Banking standards require that keys must *never* appear in unencrypted form outside of a Secure Cryptographic Device (SCD). The permitted ways for a key to get into the system are to have it generated within an SCD, or to have it imported from another system in encrypted form, or to have it entered in multiple cleartext key parts using dual control / split knowledge techniques such that no single person has information about the value of any part of the final key. These methods ensure that a key is never available outside of an SCD, except in encrypted (wrapped) form. Thus, no plaintext is available to an attacker. Furthermore, technical and procedural controls prevent an attacker from entering their own chosen plaintext keys into the system as a way to create known plaintext-ciphertext pairs.

Message Authentication (MAC): Many transactions involve computation of message authentication codes, which are cryptographically-generated check values computed over a data string. While MAC algorithms use encryption algorithms "under the covers", they are designed so that no complete

[1] The keys are 8-bytes long, but only 56 bits contain key material. The remaining 8 bits are used for parity.

[2] For example, see ANSI X9.97 or ISO 13491.

ciphertext blocks ever appear in the MAC result. For example, the common TDES CBC-MAC algorithm only outputs a truncated subset of the final block of CBC encryption ciphertext, typically the leftmost 4 bytes (32 bits). Since so much of the ciphertext information is discarded, an attacker does not truly have plaintext-ciphertext pairs.

Card Security Codes: Most credit cards carry a security code, often printed on the back near the signature panel. These codes are used to verify that the person making a purchase actually has the card in their possession. Like other processes mentioned in this article, the security codes are computed using cryptography, but they do not output the direct result of any TDES operation on the cardholder data that is their input. The output is not the ciphertext of any plaintext input values, and thus no plaintext-ciphertext pairs are available to an attacker.

Unique Key Per Transaction: Many transactions in the payments system use some type of key derivation, such that every transaction uses a different key. Examples include the DUKPT (Derived Unique Key Per Transaction) method that is common in POS terminals, and the derivation of transactions keys with EMV smart cards. When a key is only used for a single transaction, it is clear that an attacker cannot collect enough plaintext-ciphertext pairs to attack that key, even if both plaintext and ciphertext are available.

Conclusion

What we have seen here is that the best attacks against 2-key TDES require collecting large numbers of plaintext-ciphertext pairs, but that most use cases in the banking payments system do not offer the attacker those pairs. As a result, 2-key TDES is much less susceptible to attack in these environments than it would be in a "generic" one where TDES is used for general-purpose data encryption.

This does not mean that the industry should focus any less on the quickest possible migration to AES. However, it does show that the risk of cryptographic attacks is less serious in these systems than some information would lead you to believe.

Todd W. Arnold
Senior Technical Staff Member
IBM Cryptographic Coprocessor Development

If you have questions or comments about this newsletter, or suggestions for future issues, please send email to crypto@us.ibm.com.