IBM Common Cryptographic Architecture:

By using the IBM® CCA application programming interface (API), you can obtain cryptographic services from the CCA.

CCA products provide a variety of cryptographic processes and data-security techniques.

Your application program can call verbs (sometimes called services) to perform the following functions:

Data confidentiality: Encrypt and decrypt information, typically using the AES or DES algorithms in Cipher Block Chaining (CBC) mode to enable data confidentiality.

Data integrity: Hash data to obtain a digest, or process the data to obtain a Message Authentication Code (MAC) or keyed hash MAC (HMAC), that is useful in demonstrating data integrity.

Nonrepudiation: Generate and verify digital signatures using either the RSA algorithm or the ECDSA algorithm, to demonstrate data integrity and form the basis for nonrepudiation.

Authentication: Generate, encrypt, translate, and verify finance industry personal identification numbers (PINs) and American Express, MasterCard, and Visa card security codes with a comprehensive set of finance-industry-specific services.

Key management: Manage the various AES, DES, ECC, and RSA keys necessary to perform the mentioned cryptographic operations.

You use the CCA security API to access a common cryptographic architecture.

Figure provides a conceptual framework for positioning the CCA security API, which you use to access a common cryptographic architecture. Application programs make procedure calls to the CCA security API to obtain cryptographic and related I/O services. You can issue a call to the CCA security API from essentially any high-level programming language. The call, or request, is forwarded to the cryptographic services access layer and receives a synchronous response. Your application program loses control until the access layer returns a response after processing your request.

The runtime software can be divided into the following categories:

- Service-requesting programs, including application and utility programs.
- The security API, an agent function that is logically part of the calling application program or utility.
- The cryptographic services access layer: an environment-dependent request routing function, key-storage support services, and device driver to access one or more hardware cryptographic engines.
- The cryptographic engine software that gives access to the cryptographic engine hardware.
  The cryptographic engine is implemented in the hardware of the CEX*C coprocessor. Security-sensitive portions of CCA are implemented in the cryptographic engine software running in the protected coprocessor environment.
- Utility programs and tools provide support for administering CCA secret keys, interacting with CCA managed symmetric and public key cryptography key storage, and configuring the software support.

Reference:
https://www.ibm.com/support/knowledgecenter/en/linuxonibm/com.ibm.linux.z.wskc.doc/wskc_c_ch1ccafuncover.html

Feige–Fiat–Shamir identification scheme

In cryptography, the **Feige–Fiat–Shamir identification scheme** is a type of parallel zero-knowledge proof developed by Uriel Feige, Amos Fiat, and Adi Shamir in 1988. Like all zero-knowledge proofs, it allows one party, Peggy, to prove to another party, Victor, that she possesses secret information without revealing to Victor what that secret information is. The Feige–Fiat–Shamir identification scheme, however, uses **modular arithmetic** and a parallel verification process that limits the number of communications between Peggy and Victor.