



TECSEC<sup>®</sup> Incorporated



*Information Security and Cryptography*

CONTENTS

**Information Security ..... 2**

Some Objectives of Information Security ..... 2

Terminology ..... 2

Enforcement..... 3

**Cryptology ..... 3**

Characteristics of Cryptographic Functions ..... 3

Cryptographic Functions ..... 4

*Unkeyed Cryptographic Primitives*..... 4

*Symmetric Key Cryptographic Primitives*..... 4

*Asymmetric Key Cryptographic Primitives*..... 5

**Vulnerabilities ..... 6**

**Standards and Validations..... 7**

**References ..... 7**

## Information Security

### Some Objectives of Information Security

- *Confidentiality* or secrecy is the prevention of the disclosure of information. This can be enforced with cryptography. A small amount of data – the cryptographic keying material – is distributed secretly so that a larger amount of data – the ciphertext – can be sent over non-secure networks (*data in transit*) or stored on publicly accessible areas (*data at rest*). *Forward secrecy* is not being able to compromise future communications even if current communications are compromised. *Backward secrecy* is not being able to compromise previous communications even if current communications are compromised.
- *Access control* is the prevention of the use of information to those not specifically given permission to access that information. Access controls include: general use (i.e. execute), read, write, create, delete, modify, append, change permission, etc. Access control is *discretionary* (DAC) when those that have access can also set or change access to the data. It is *mandatory* (MAC) when an administrator controls access permissions. Access control can be *identity-based*, *role-based* (RBAC) or *content-based* (CBAC). Identity based controls specify access to objects for each individual, e.g. access control lists (ACLs). RBAC specifies access to groups of users who share a defined role. CBAC tailors access to data depending on the data itself.
- *Data separation* is the ability to keep information of varying types and levels of access in the same physical or logical location, e.g. on the same hard disc drive.
- *Cryptographic key management* involves the creation, derivation, distribution, storage, recovery, revocation and destruction of cryptographic keying material. This is very important when cryptography is used in information security systems.
- *User Identification and Authentication (I&A)* is the process of proving one's identity. User identification is based on three factors – the *knowledge* factor, the *possession* factor and the *biometrics* factors. The knowledge factor uses something known by a person, e.g. a PIN or password. The possession factor uses something possessed by a person, e.g. a smart card. The biometrics factor uses a biological or behavioral characteristic about a person, such as a written signature or a fingerprint.
- *User authentication* is the binding of a user's identity to information.
- *Non-repudiation* is the ability to prove that a user's identity is bound to information.
- *Data integrity* is the ability to detect tampering of information. *Message authentication* means data integrity, user authentication and non-repudiation.
- *Time stamping* is the marking of information with a verifiably correct date and time in a way that cannot be forged.

### Terminology

In talking about protocols, many textbooks use this terminology for communications: Two people who share information are usually called *Alice* and *Bob* (or A and B). Sometimes a third party is involved who is called *Carol*. *Dave* is there if four parties are involved. An attacker can be passive, i.e. just listening in, or intercepting the data flowing between A and B. The passive attacker is called *Eve* (for eavesdropper). An attack may be active in which case data flowing between A and B would be changed. The active attacker is called *Mallory* (for malicious attacker).

## Enforcement

Security can be implemented using:

- *Physical enforcement*: protection of information by physically removing that information from threats, e.g. placing it in a locked room. Physical security is implemented with people and policies.
- *Hardware enforcement*: protection of information by using devices to store and process that information. These devices provide resistance to tampering. An example would be a smart card.
- *Software enforcement*: protection of information by logic built into computer programs that are designed to control access. The read and write permissions used in a computer operating system are one example of software enforcement.
- *Cryptographic enforcement*: protection of information by using encryption, digital signatures and other mathematical functions applied to data. File encryption is an example of using cryptography to enforce secrecy.

## Cryptology

*Cryptography* is making unbreakable (or nearly unbreakable) codes. *Cryptanalysis* is the study of how to break them. *Cryptology* is the study of cryptography and cryptanalysis.

Historically, cryptography involves transforming data so that it is unreadable except by those that have knowledge of the *key*. Data to be transformed is called *plaintext* or cleartext. After transformation, the result is called *ciphertext*. Transforming plaintext to ciphertext is called *encryption*; the inverse transformation from ciphertext to plaintext is called *decryption*. The functions that perform these transformations are sometimes called *cryptographic primitives*. Cryptographic primitives that use a single key are called *symmetric key*, secret key or classical cryptographic primitives.

Since the 1970s, newer cryptographic functions have appeared that use two keys. These types of functions can be used for key exchange, key transport and message authentication. Cryptographic primitives that take two keys are called *asymmetric key*, public key or public-private key primitives.

Another class of functions that are important in cryptography do not use cryptographic keys. Two examples of these functions are hash functions and pseudorandom number generators.

Symmetric, asymmetric and unkeyed cryptographic primitives can be used together to implement security objectives. The term *cryptographic function* will be used to refer to any cryptographic primitive in a general sense.

### Characteristics of Cryptographic Functions

A cryptographic function may be characterized by the set of inputs and outputs that it can accept and produce, and their size, called the *block size*. A plaintext *message* is broken into blocks and each block is encrypted. Generally, there will be some data at the end of message that will not fill a block. *Padding* or some other method of handling this last block is needed.

A cryptographic function can also be characterized by the *keyspace*, i.e. set of all keys that the function can accept, and especially by the size of the keyspace. A large keyspace is necessary, but not sufficient, for a cryptographic scheme to be secure.

Cryptographic functions and their suitability to a given problem can be evaluated against various criteria:

- **Strength.** This is measured by the number of operations required (work factor) to defeat the cryptographic functions purpose. It is not always easy to quantify the strength of a cryptographic function. A large keyspace is essential, however, it may take years of dedicated effort by many specialists before the strength of an algorithm can be attested.
- **Performance,** both in terms of speed and memory usage.
- **Ease of implementation.** This is also important for verifying correctness of the implementation. Space required for the implementation may be important in some environments.
- **Functionality.** Cryptographic functions can be combined depending on which security objective is to be solved. Different types of functions can complement each other.
- **Usage.** Different functionality is produced when a cryptographic primitive is used in different ways with different inputs.

The ways in which cryptographic functions are combined and used produce *cryptographic protocols*. Like cryptographic algorithms, it is not easy to measure the strength or correctness of cryptographic protocols.

## Cryptographic Functions

*One-way functions* are easy to compute but have the property that it is hard to figure out what the function input was, given only the function's output. *Trapdoor one-way* functions are easy to compute, and hard to invert except when additional information is known. These type of functions play a significant role in cryptography. The word "hard" means "computationally infeasible" – a meaning which changes as time and technology progress.

Cryptographic functions are broadly classified into three types of primitives:

### Unkeyed Cryptographic Primitives

There is no key for these types of functions – only input and output.

- *Cryptographic hash functions* take an arbitrary length input and produce output of a constant size. The output is used as a compact representation of the input. It should be hard to compute an input that hashes to a given value. Examples include SHA1, MD5 and RIPEMD. SHA1 produces a 160 bit (20 byte) hash. MD5 produces a 128 bit (16 byte) hash. One version of RIPEMD produces a 128 bit hash, another version produces 160 bits.
- *Pseudorandom number generators* (PRNG) are deterministic algorithms which produce a sequence of bits that are statistically independent and unbiased. The input to a PRNG is called the *seed*, which is used to initialize the algorithm. The effect of a PRNG is to use a small amount of real random data, which is usually hard to acquire, to generate a large amount of random, yet deterministic, data. The output of the PRNG should not be predictable even when all of the previous output is known.

### Symmetric Key Cryptographic Primitives

Plaintext input of a given size is transformed into (usually) the same size of ciphertext. The transformation is controlled by a key. This type of algorithm is sometimes just called a *cipher*. There are two broad classes of symmetric key cryptographic functions based upon the internal state of these functions:

- *Block cipher:* The internal state is initialized with the key and remains constant. The same plaintext block will always produce the same ciphertext block for a given key.

- *Stream cipher*: The internal state is initialized with the key but changes according to the length or content of previous blocks encrypted. If internal state changes are affected by previous encrypted blocks, then the stream cipher is called a *self-synchronous* stream cipher. A *synchronous* stream cipher internal state changes independent of the encrypted data.

### Block Cipher Modes

The straight use of a block cipher – just encrypt each block with the same key – is called *electronic code book* mode (ECB). To prevent identical plaintext blocks from showing the same ciphertext, feedback of the previous ciphertext block can be combined with the plaintext – a process called chaining. When the exclusive-or operation is used to combine the plaintext with the previous ciphertext block, the block cipher is said to operate in *cipher block chaining* (CBC) mode. Since there is no previous ciphertext block for the first plaintext block to be encrypted, an additional, random block is used to start the encryption. This block is called the *initialization vector* (IV).

*Cipher-feedback* mode (CFB) encrypts the previous ciphertext block (or portion of it) with the block cipher and uses the exclusive-or of this result with the current plaintext block to produce the next ciphertext block. This changes the block cipher into a self-synchronous stream cipher. If the previous output of the encryption is fed back to the block cipher instead of the previous ciphertext, then the mode is called *output feedback* mode (OFB). OFB changes the block cipher into a synchronous stream cipher. Both of these modes require an initialization vector.

### Examples of Symmetric Key Algorithms

An example of a symmetric key block cipher is the *Data Encryption Standard* (DES). It has a plaintext and ciphertext block size of 64 bits (8 bytes). The key size is 56 bits (7 bytes) but DES keys are usually 64 bits which includes the 56 bits used to initialize the algorithm and 8 bits used for parity. DES can be made stronger by using DES three times with two or three different keys, called triple DES. However, for various reasons, three key triple DES is not very much stronger than two key.

The new *Advanced Encryption Standard* (AES) will use 128 bit plaintext and ciphertext blocks. The key size is variable – either 128, 192 or 256 bits.

An example of a stream cipher is the Vernam cipher which adds a stream of characters to the plaintext to produce ciphertext. If the key stream is random, used only once and is as long as the message, then it is called a one-time pad. The one-time pad is the only unbreakable system known.

### Asymmetric Key Cryptographic Primitives

One key is used to encrypt plaintext. A different key is used to decrypt ciphertext. Together these two keys form a *key pair*. The two keys in a key pair are usually related in some way. One key may be derived from the other. When this is the case, the key that is derived is called the *public* part of the key pair, or just the public key. The other is called the *private* part of the key pair, or just the private key. If neither key can be derived from the other, then either one may be chosen as the public part.

### Confidentiality

If the public key is the encrypt key, then the asymmetric key function can be used for confidentiality. The public key is used to encrypt data that only the owner of the private key can decrypt. Secure key pairs are much larger than equivalent symmetric keys. In addition, asymmetric algorithms are slower than symmetric algorithms. Therefore, symmetric key cryptography is generally used for data encryption and asymmetric key cryptography is used to protect the symmetric keys.

Encrypting the symmetric key with a public key and sending this with the encrypted message is called *key transport*. Another asymmetric key method used to protect symmetric keys is called *key agreement*. Two parties each share public keying material which is used with private keys to create a symmetric key encryption key.

## Digital Signatures

If the public key is the decrypt key, then the asymmetric key function can be used for message authentication. This is called *digital signature* generation and verification. The private key is used to digitally sign a message. The public key is used to verify the data. For performance reasons, the data that is signed is usually not the message itself but a hash of the message.

Since asymmetric key cryptography means using information that may be published, it is important to protect the integrity of public keys. This can be accomplished by having a trusted third party – a *certificate authority* (CA) – digitally sign public keys. The signed public key along with the identity of the owner (and other information) is called a *certificate*. Managing public keys and certificates involves the use of a *public key infrastructure* (PKI).

## Examples of Asymmetric Key Algorithms

Examples of asymmetric key algorithms include RSA<sup>1</sup>, Diffie-Hellman key agreement, DSA<sup>2</sup> and ElGamal digital signatures, and elliptic curve (EC) methods. These algorithms all depend on large integer modular exponentiation.

The two keys in an RSA key pair are independent, therefore RSA can be used for either confidentiality or authentication depending on which key is chosen as the public key. Diffie-Hellman key agreement is used for confidentiality. DSA and ElGamal, both of which are based on the Diffie-Hellman algorithm, are used for authentication. Elliptic curve methods are analogous to Diffie-Hellman and DSA, but do the math operations differently. EC has the property that for an equivalent level of security, EC keys can be much smaller than Diffie-Hellman keys.

## Vulnerabilities

Security built with cryptography can fail in many subtle ways. Security is like a chain, i.e. a system's security is only as strong as its weakest link. The same can be said for cryptography. Four areas of potential failure are:

1. *Design*. Designing cryptographic algorithms and protocols is an art as much as it is a science. Novices (and even experts) in cryptography frequently create algorithms that, although they seem to be very complicated and secure, are successfully attacked immediately by experts. A cryptographic algorithm or protocol must undergo years of scrutiny by the experts before anyone has an idea of how secure it really is. When using cryptography, it is usually best to stick with standard published algorithms and methods.
2. *Implementation*. Only those who have been living under a rock for the past twenty years are not aware of the fact that computer programs have bugs. Bugs in cryptographic software are dangerous because they open up the system to attacks. Very thorough testing, including third part beta testing, is necessary. Laboratories are available to certify that cryptographic systems have met a certain standard.
3. *Installation*. Many computer systems come configured out of the box for minimum security. Defaults are usually set at the lowest security level and security features are turned off. It is up to the administrator to add security by changing the configuration of the system. This can be a major job in a complex computer or communication systems. Hackers frequently know more about a system than busy administrators. Anything missed can open up vulnerabilities. From a security standpoint, it is better to have maximum security turned on from the initial installation of a product.

---

<sup>1</sup> From Rivest, Shamir and Adelman.

<sup>2</sup> Digital Signature Algorithm.

Another problem is that there may be bugs or vulnerabilities in a product that are not discovered until after installation. Bug fixes and patches must be applied but frequently are not, either because the administrators didn't know about it or just failed to keep up.

4. *Usage.* If a user can bypass security and make his life easier, he will. Good application design must prevent these shortcuts, but even more importantly, must not allow security features to get in the way of normal work. In addition to preventing bad things, security must also be able to detect and inform users when bad things happen.

These areas of vulnerability must be kept in mind when building cryptographic systems.

## Standards and Validations

Cryptographic standards are available for compatibility concerns, due diligence, and to provide a known level of security. Cryptographic standards are concerned with algorithms, protocols and methods. Cryptographic standards are published by the American National Standards Institute (ANSI), the National Institute for Standards and Technology (NIST), and the International Standards Organization (ISO). Ad hoc standards are also published by the Internet Engineering Task Force and by RSA.

ANSI standards exist for cryptographic algorithms, protocols and other security related topics, such as biometrics. The US financial industry must use products that meet these standards.

NIST maintains a number of cryptographic standards, and coordinates validation programs for many of those standards. The Cryptographic Module Validation (CMV) Program encompasses validation testing for cryptographic modules and algorithms. All of the tests under the CMV Program are handled by third-party laboratories that are accredited as Cryptographic Module Testing (CMT) laboratories by the National Voluntary Laboratory Accreditation Program (NVLAP).

The Common Criteria (CC) project has created an international standard (ISO 15408) for evaluation of information security. The CC presents information security requirements for products under distinct categories of functional requirements and assurance requirements. In the US, the NIST and NSA are the sponsoring organizations for the development of the CC. The NIST and NSA have created the National Information Assurance Partnership (NIAP). The goal of NIAP is to establish cost-effective testing, evaluation, and certification programs. Vendors that want CC certification for their products in the US can use NIAP.

IETF standards are informational. However, most, if not all of the protocols that the Internet uses are those published by the IETF. Another popular set of standards being used are the Public-Key Cryptography Standards (PKCS) published by RSA Laboratories.

## References

The first 10 references are ANSI standards. These may be bought from the ABA at <http://webstore.ansi.org>. The FIPS documents are available from NIST at <http://csrc.nist.gov>. These particular ANSI and FIPS standards were chosen because of their importance to TECSEC<sup>®</sup> Incorporated's Constructive Key Management<sup>®</sup> (CKM<sup>®</sup>) technology. Reference [19] is probably the most comprehensive reference on cryptography, though it is not a textbook. It contains an excellent bibliography, though it is a bit dated. A good book for programmers and those implementing cryptography is reference [21]. It also contains a large bibliography. Reference [20] is an introduction to elliptic curve techniques as well as an implementation guide for programmers and is recommended for those wanting more information on elliptic curve cryptography. Reference [22] is a college level textbook on cryptography. Reference [18] is a CD-ROM containing the papers from all of the CRYPTO and EUROCRYPT conferences from 1981 to 1997 plus a printed index (460 pages!) to those papers. It has a wealth of information, generally at the most advanced levels of cryptography, that cannot be found anywhere else. There are fewer references available on cryptanalysis, however, it is covered admirably in [16] and [17].

For more references see [21] and especially [19].

- [1] ANSI X9.30-1997 (Part 1). "Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry – Part 1: The Digital Signature Algorithm (DSA)", ASC X9 Secretariat – American Bankers Association. 1997.
- [2] ANSI X9.30-1997 (Part 2). "Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry – Part 2: The Secure Hash Algorithm (SHA) (Revised)", ASC X9 Secretariat – American Bankers Association. 1997.
- [3] ANSI X9.31-1998. "Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA).", ASC X9 Secretariat – American Bankers Association. 1998.
- [4] ANSI X9.42-2000. "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry: Transport of Symmetric Algorithm Keys Using RSA", draft, 2000.
- [5] ANSI X9.44-1994. "Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography", ASC X9 Secretariat – American Bankers Association. 1994.
- [6] ANSI X9.52-1998. "Triple Data Encryption Algorithm Modes of Operation", ASC X9 Secretariat – American Bankers Association. 1998.
- [7] ANSI X9.65-2000. "Triple Data Encryption Algorithm (TDEA), Implementation Guide", draft, 2000.
- [8] ANSI X9.69-1999. "Framework for Key Management Extensions", ASC X9 Secretariat – American Bankers Association. 1999.
- [9] ANSI X9.73-2000. "Cryptographic Message Syntax", draft, 2000.
- [10] ANSI X9.84-2000. "Biometrics Management and Security for the Financial Services Industry", draft, 2000.
- [11] FIPS PUB 46-3, "Data Encryption Standard", Federal Information Processing Standards Publication 46-3, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia. 1999.
- [12] FIPS PUB 140-1, "Security requirements for cryptographic modules," Federal Information Processing Standards Publication 140-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia. 1994.
- [13] FIPS PUB 180-1, "Secure hash standard," Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia. April 17, 1995.
- [14] FIPS PUB 186-2, "Digital signature standard," Federal Information Processing Standards Publication 186, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia. 2000.
- [15] D. E. Knuth. *The Art of Computer Programming – Seminumerical Algorithms*, volume 2. Addison-Wesley, Reading, Massachusetts, 2<sup>nd</sup> edition, 1981.
- [16] A. G. Konheim. *Cryptography, A Primer*. John Wiley & Sons, New York, New York, 1981.
- [17] I. J. Kumar. *Cryptology – System Identification and Key-Clustering*. Aegean Park Press, Laguna Hills, California, 1997.

- [18] K. S. McCurley, C. D. Ziegler (Eds.). *Advances in Cryptology – Electronic Proceedings and Index of the CRYPTO and EUROCRYPT Conferences, 1981 – 1997*. Springer-Verlag, Berlin, Heidelberg, 1998.
- [19] A.J. Menezes, P.C. van Oorshot, S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, Florida, 1997.
- [20] M. Rosing. *Implementing Elliptic Curve Cryptography*. Manning, Greenwich, Connecticut, 1999.
- [21] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, New York, 2<sup>nd</sup> edition, 1996.
- [22] D.R. Stinson. *Cryptography: Theory and Practice*. CRC Press, Boca Raton, Florida, 1995.