# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities


WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

## Interested in publishing with us?
## Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter

# Security and Privacy in Three States of Information

*Ebru Celikel Cankaya*

## Abstract

In regard to computational context, information can be in either of three states at a time: in transit, in process, or in storage. When the security and privacy of information is of concern, each of these states should be addressed exclusively, i.e., network security, computer security, and database/cloud security, respectively. This chapter first introduces the three states of information and then addresses the security as well as privacy issues that relate to each state. It provides practical examples for each state discussed, introduces corresponding security and privacy algorithms for explaining the concepts, and facilitates their implementation whenever needed. Moreover, the security and privacy techniques pertaining to the three states of information are combined together to offer a more comprehensive and realistic consideration of everyday security practices.

**Keywords:** information in transit, information in process, information in storage, network security, computer security, database security, cloud security

## 1. Introduction

The world is living in an era of information outburst thanks to rapid speed of technological developments in computational domain. Many factors contribute to the immense amount of data available online: Mobile devices are becoming more accessible to everyone with their decreasing cost, underlying network technologies are facilitating data transfer by providing faster and more reliable communication, data processing methods (such as editing, compressing) are helping customize data format, and increasing data storage capacities are turning bulk data storage more effective.

With all technology and convenience so easily reachable, an ordinary user is allured further to handle even more information by either transferring, processing, or storing them. Yet, the issues of security and privacy are often taken for granted, and unfortunately this ignorance may cause a destructive consequence by totally violating the security and privacy of information, as well as its owner.

In this chapter, three fundamental states of information, i.e., information in transit, information in process, and information in storage, are defined first. Then the chapter addresses security and privacy of information in each state by giving examples. It should be noted that as information is the processed form of data, these two terms, i.e., information and data, are used interchangeably throughout the text. The chapter then provides algorithms to achieve privacy and security of information in these three states and how they apply to particular states of information for ensuring security.

The rest of the chapter is organized as follows: In Section 2, three states of information are defined together with examples for each. In Section 3, security mechanisms, i.e., privacy and security algorithms that apply to each state of information, are discussed in detail, and examples are provided. Finally, the chapter is concluded in Section 4.

## 2. Three states of information

This section identifies three fundamental states that information can be in at a time. It is essential to distinctively identify each state, as corresponding security measures vary for each of these states. To address each separately, the information residing in either of three states at a time in a computational environment are identified. These three states are listed as information in transit, information in process, and information in storage. Information in transit refers to the status where an underlying network (wired or wireless) facilitates the transmission of data from one place (source) to another (destination). Information in process refers to the case when data is processed so that it transforms from source format to destination format. And information in storage refers to the mostly stagnant form of data that resides on a storage media for future reference. As these brief descriptions imply, information in each state has different properties than information in other states. As an example, information in transit is different from information in process and information in storage.

It should also be noted that these states may overlap to form many variations. As an example, states can form variations in pairs, such as information could be processed first and then transferred, or processed first and then stored, or stored information is retrieved for further processing, etc. Or, in a more complex format, each of the three states of information may combine in any order to form a sequence of operations, such as stored information is retrieved from database and is processed to yield new information and this new information is transferred to a remote destination.

The following subsections address each state of information in detail by describing them first and then providing examples for each state. This, later, serves as a basis for explaining security and privacy measures with respect to each state of information that is explained later in Section 3.

### 2.1 Information in transit

The first state of information is information in transit. This state refers to the situation when information handled is transferred from one place (source) to another place (destination). As depicted in **Figure 1**, in the context of information in transit state, the information residing in source side is transmitted to destination via an underlying network. The underlying network infrastructure could be of various types, such as cable network, wireless network, etc., and does not differentiate

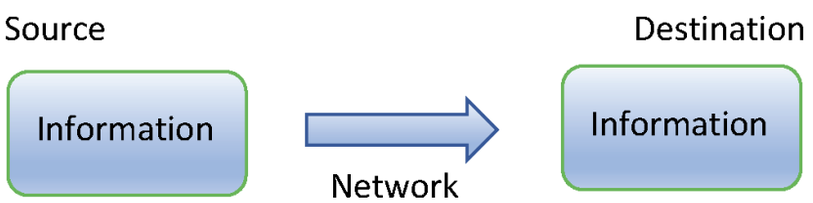


**Figure 1.**
*Information in transit.*

the type of data being transferred, as each data piece is processed in the bit level of granularity. This enables the transfer of information in various possible formats, such as plaintext, still image, movie, voice, etc.

Though **Figure 1** illustrates information being transmitted from source to destination, more than usual, the roles change and the source becomes destination and the destination becomes source. This is because of the inherently full duplex property of communication in most of the times.

It should also be noted that if information in transit is considered only in isolation, then it should not change the format of data being transferred. This explains why in **Figure 1** both sender and destination sides label information exactly the same way. And to guarantee this unchanging property of data, integrity mechanisms are inherently built in network systems that facilitate information in transit. If a more complex system is designed and implemented by combining information in transit and information in process states, then labeling of each information entity will have to be modified accordingly. In particular, these entities will label information as $\text{Information}_1$ and $\text{Information}_2$ on each side so as to indicate the changing content of the information itself. This phenomenon is explained further in Section 2.2 and is depicted later in **Figure 2**.

Sending a memo that has text, pictures, and videos in it as part of a business operation from company headquarters to a branch office is an example of the case where information is transmitted from source (the company headquarters) to destination (the branch office).

While improvements in technology are facilitating the way information can be transferred, this brings along the issue of providing the security and privacy of information transferred. These issues are addressed in the security and privacy section that follows.

## 2.2 Information in process

This state focuses on how one operates on data to change its form. It is very common in computational operations today to process data such that it no longer possesses its original format. As an example, one may compress data so that it occupies less space, another may encrypt it so that it becomes unintelligible to unintended third parties, yet another combines compression and encryption so as to benefit from combined effects of the two.

Information in process is represented graphically in **Figure 2**, where a series of operations are applied to input data to yield the output data. The expectation in the end of the process in this figure is that the output ($\text{Information}_2$) will be different from input ($\text{Information}_1$). This is a fundamental difference from **Figure 1**, where information on each side was expected to be the same.

In the context of information in process, take, for example, the process of encrypting text. The input data will be a legible text, such as a sentence, a document, etc., whereas the output generated will be an unintelligible sequence of characters, as is deliberately meant to be by the process. The process itself could be one single operation as simple as circularly shifting characters a certain amount to


**Figure 2.**
*Information in process.*

the left or right or, in the hope to achieve sophisticated complexity, a sequence of operations, such as applying a complex math-based prediction model on previous data to calculate the overall end of year benefit for a company.

The concern with information in process is that the reliability of data should be preserved at all times so that the resulting data can be trusted. This involves authentication and reliability of input and output data, as well as the trustability of the process itself, each of which are discussed later in the Security Mechanisms for Three States of Information subsection below.

### 2.3 Information in storage

Information in storage state refers to the case when information rests in a storage media of choice for making it available in the future. **Figure 3** depicts the last state of information which is called information in storage.

The expectation at this state is to ensure that the information remains intact in the storage and moreover no unauthorized party can access to it. These efforts involve authentication of the users who can access the information, as well as integrity of information being stored. The requirement to keep the data intact, i.e., prevent its integrity while in storage, explains why the information is labeled the same in both user and storage sides in **Figure 3**. This is a similar requirement that was mentioned while explaining the information in transit, as was depicted in **Figure 1**. Yet, as information in process inherently changes the format of data, the labeling of information on each side of the process was different for information in process (see **Figure 2**).

As for the storage media, various options are possible: Information can be stored on a local database, or on cloud, particularly when local storage is not sufficient to accommodate large amount of data.

Once stored, information usually is accessed again for retrieval purposes, explaining the full duplex form of communication between the user and storage.

As an example, a person can store family vacation pictures on his cell phone and then may decide to store them on a DVD for allocating more space on his cell phone storage.

It should be noted that the distinction among these three states of information, namely, transit, process, and storage, is not crystal clear. This is because almost all the time these states are utilized in an overlapped manner. Formally, if the set of states is $S = \{s_1, s_2, s_3\}$, then all possible permutations of these states generate the following permutation set $P = \{\emptyset, s_1, s_2, s_3, s_1s_2, s_2s_1, s_1s_3, s_3s_1, s_2s_3, s_3s_2, s_1s_2s_3, s_1s_3s_2, s_2s_1s_3, s_2s_3s_1, s_3s_1s_2, s_3s_2s_1\}$. It should also be noted that it is permutations, not combinations, as the order matters when combining each of the three states. An example regarding the three states of information could be given as follows: Compressing data first and then emailing it to destination should be considered a different operation than emailing the data first and then compressing it at the destination.

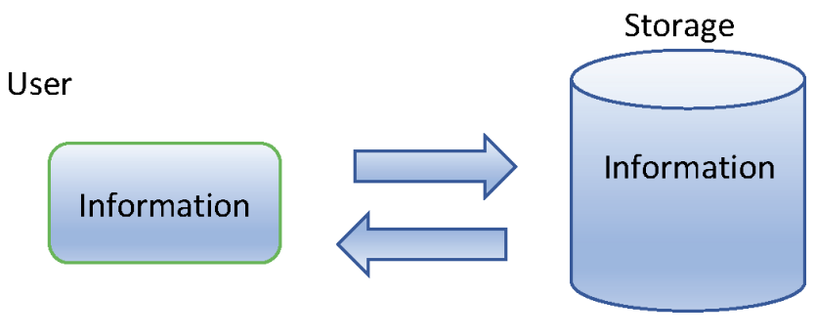


**Figure 3.**
*Information in storage.*

| Permutation | State | Description |
|---|---|---|
| 1 | Ø | No operation (trivial case) |
| 2 | $s_1$ | Transit |
| 3 | $s_2$ | Process |
| 4 | $s_3$ | Storage |
| 5 | $s_1s_2$ | Transit-process |
| 6 | $s_2s_1$ | Process-transit |
| 7 | $s_1s_3$ | Transit-storage |
| 8 | $s_3s_1$ | Storage-transit |
| 9 | $s_2s_3$ | Process-storage |
| 10 | $s_3s_2$ | Storage-process |
| 11 | $s_1s_2s_3$ | Transit-process-storage |
| 12 | $s_1s_3s_2$ | Transit-storage-process |
| 13 | $s_2s_1s_3$ | Process-transit-storage |
| 14 | $s_2s_3s_1$ | Process-storage-transit |
| 15 | $s_3s_1s_2$ | Storage-transit-process |
| 16 | $s_3s_2s_1$ | Storage-process-transit |

**Table 1.**
*Permutation of information states.*

**Table 1** presents all possible permutations of these three states applied either in isolation, or combined in pairs, or combined in triplets based on the underlying scenario each represents. The states listed in **Table 1** map to generic states $s_1$, $s_2$, and $s_2$ of the formal definition given above as follows: $s_1$ = information in transit, $s_2$ = information in process, and $s_3$ = information in storage.

Take, for example, the case where the husband retrieves previous year's tax records from his hard disk and emails them to his wife, and then the wife processes them to create this year's tax amount to be paid to the government. All three states of information are employed in this scenario in this particular order, storage-transit-process, corresponding to line 11 of **Table 1**. To make it even more complex, imagine the wife submits the tax form for this year via online submission and then stores it in her disk. This would mean expanding the order of states to include "transit-storage" operations in addition to what already exists in line 11 of the table.

## 3. Security mechanisms for three states of information

In this section, each of the three states of information described in Section 2 is addressed, regarding the security features each state requires. In particular, Section 3.1 focuses on the security of information in transit state and hence introduces and explains network security. Section 3.2 concentrates on the state of information in process and explains the security measures, algorithms, and their implementations defined under the title of computer security. And Section 3.3 analyzes the security of information in storage state and provides information pertaining to database as well as cloud security.

As each state handles information differently, the security requirements per state differ accordingly. For this reason, subsections below elaborate security concerns with regard to each state of information and present various techniques to provide the security of information in each of these states.

## 3.1 Network security

Different network infrastructures are probable today. The decision to make on which network infrastructure to employ depends on the needs for speed, reliability, the type of data being transferred, and the technology available where the source and destination are. The most common network infrastructures are TCP/IP and UDP communications. Each of these communication protocols utilize a layered architecture to process transmitted data. Essentially, they use similar fields for uniquely identifying the data being transmitted (such as source IP address, destination IP address, source port number, destination port number, flags, and other fields).

The fundamental security services that are expected to be provided by network security are confidentiality, integrity, availability (known as CIA principles), and authentication. In the following subsections, each of these services is discussed, and examples of tools and mechanisms used to provide these services are given.

### *3.1.1 Confidentiality*

Confidentiality refers to the service where secrecy of the data being transmitted is preserved at all times. It is an essential requirement to ask from a network communication as otherwise the trust is violated irreparably.

The means that to provide confidentiality service is to employ encryption, i.e., encoding data so as to make it unintelligible to unintended third parties.

Based on the number of encryption keys used in an encryption system, two types of encryption are possible: symmetric encryption and asymmetric encryption. In a cryptosystem where C is the ciphertext, P is the plaintext, K is the secret key, E is the encryption algorithm, and D is the decryption algorithm, the symmetric encryption and decryption processes are formulized as seen in Eqs. (1) and (2), respectively.

$$\text{Symmetric encryption scheme encryption process: } C = E_K(P) \tag{1}$$

$$\text{Symmetric encryption scheme decryption process: } P = D_K(C) \tag{2}$$

In symmetric encryption, each communicating party uses the same secret key to encrypt the transmitted content. The need to use a secret key in symmetric encryption raises the question of how to provide the secrecy of this key itself. The solution is to employ a different key (named as session key) each time a new transmission takes place. Surely this adds on to the complexity of the scheme, but this is the inevitable result of the tradeoff between secrecy and complexity in security systems. The most common symmetric encryption algorithms are DES [1] and AES [2] with key sizes of 128 bits and 256 bits, respectively.

Take, as an example, the case where n users communicate with each other by using symmetric encryption scheme. To achieve a secure communication, each entity will have to ensure that his/her communication will not be revealed to other entities in the system. This will lead to a requirement to employ $\frac{nx(n-1)}{2}$ total number of symmetric keys in the system. Known as the $n^2$ problem [3], the size of symmetric keys becomes quadratically complex and is problematic particularly for large n. To alleviate the $n^2$ problem, a new type of encryption called asymmetric encryption has been introduced.

In asymmetric encryption, each communicating party possesses a key pair {privateKey, publicKey} and employs this pair for communicating with any of the

other parties in the system. Therefore, for communication among n users, the need for total number of keys is 2n only, a dramatic reduction as compared to the symmetric key encryption. A common asymmetric algorithm is RSA that is widely used in cryptographic platforms.

Another classification for cryptographic algorithms is based on how the plaintext bits are processed: stream cipher and block cipher. In stream cipher, as the name implies, the plaintext bits are processed individually to encrypt, while block cipher bits are processed in blocks of multiples of 8 during encryption. The inherent nature of these encryption methods introduces a tradeoff between the two: Stream cipher is faster and has the advantage of low error propagation, yet it is more prone to malicious bit injection as they can go undetected very easily. The block cipher performs slower, as it has to incur the extra cost of block forming each time. It also cannot prevent error propagation in blocks. Yet, the block cipher is secure against malicious bit injection.

Yet another classification for cryptographic algorithms is based on how the plaintext is transformed into ciphertext. According to this classification, one can perform a substitution cipher or transposition (permutation) cipher. In substitution cipher, plaintext bits are transformed into ciphertext bits by employing a single alphabet (hence called monoalphabetic cipher) or multiple alphabets (hence called polyalphabetic cipher). In a monoalphabetic cipher, the key size is fixed and is reused for the entire plaintext encryption, which leads to a 1:1 mapping between plaintext and ciphertext letters. As an example, for encrypting a given plaintext in English with length n, if the encryption key is designated as K = 5, then the encryption algorithm could be written as seen in **Figure 4**.

Similarly, the corresponding decryption algorithm for monoalphabetic cipher can be written as seen in **Figure 5**.

Monoalphabetic cipher is straightforward to implement. The risk is once the key is known, it will lead to decryption of the entire ciphertext promptly, violating the requirement for security. As for deciphering a suspected to have been monoalphabetically encrypted text, one can use the source language (such as English) n-gram statistics to reveal the key size. The oldest known encryption algorithm Caesar cipher is a monoalphabetic cipher with a key size of 3. Once statistical analysis is used to decrypt monoalphabetic cipher, the letter frequencies of ciphertext will reflect the letter frequencies of source language, only with "substituted" letters this time. As an example, if key size is 4 for a monoalphabetic cipher, then the

```
monoalphabeticEncrypt(P[], K)
for i=1 to n do {
       C[i] = P[i] + K;
       print (C[i]);
}
```

**Figure 4.**
*Monoalphabetic cipher encryption.*

```
monoalphabeticDecrypt(C[], K)
for i=1 to n do {
 P[i] = C[i] - K;
       print (P[i]);
}
```

**Figure 5.**
*Monoalphabetic cipher decryption.*

plaintext letter "A" will be encrypted into ciphertext letter "E," ciphertext letter "B" will be encrypted into plaintext letter "F," etc. Therefore, the frequency of the ciphertext letter "E" will be similar to the frequency of source language letter "A," the frequency of the ciphertext letter "F" will be similar to the frequency of source language letter "B," etc. This is a very important clue in helping one decrypt the ciphertext correctly and fast.

In an effort to strengthen the security of monoalphabetic cipher, polyalphabetic encryption has been introduced. As the name implies, in polyalphabetic cipher, there is an M/N mapping between plaintext letters and the ciphertext letters. This means that a plaintext letter "A" may be encoded into cipher letter "Y" once, and into ciphertext letter "F" another time, and into ciphertext letter "K" yet another time, etc. Therefore, there is no one fixed key size which will make decryption harder and more time-consuming. This will also cause a phenomenon called uniform letter frequencies for the ciphertext letters, which will be very close to uniform distribution of letters, rather than reflecting the natural language statistical characteristics (where letters "A," "E," and "T" occur the most frequent, whereas letters "Z" and "Q" occur the least frequent for English). Deciphering a ciphertext with uniform letter frequencies is very hard as compared to deciphering a ciphertext that possesses the underlying source language frequencies.

In the case of a polyalphabetic cipher, more complex deciphering techniques such as Kasiski test [4] and index of coincidence [5] should be used. Both these techniques employ statistical analysis on ciphertext partitions formed by reordering ciphertext letters in varying lengths to reveal a meaningful plaintext correspondence, in the hope to find a most probable key size. The ultimate polyalphabetic encryption algorithm is called the Vernam Cipher (also called one-time pad) [6] and uses a key size as large as plaintext to obstruct key size computation and prevent the facilitating contribution of n-gram statistics.

The transposition (permutation) cipher uses the same letters as the plaintext while encrypting the text. As an example, in rail cipher where key size is K, the plaintext letters are combined by right shifting each letter K positions to form the ciphertext. Therefore, the ciphertext letter frequencies will remain exactly the same as the plaintext letter frequencies. This is a very important discovery that will help determine whether a transposition cipher or a substitution cipher was used in the first place.

Regardless of the type of encryption algorithm, confidentiality service for network security is provided by applying the encryption cipher of choice. For a system with symmetric encryption, encryption algorithm for the sender side will be as seen in Eq. (3), and decryption algorithm for the receiver side will be as seen in Eq. (4):

$$\text{Symmetric encryption sender side } C = E_{K_1}(P) \tag{3}$$

$$\text{Symmetric decryption receiver side } P = D_{K_1}(C) \tag{4}$$

For a system with asymmetric encryption, each entity will use a pair of keys {privateKey, publicKey}. Hence, encryption algorithm for the sender side will be as seen in Eq. (5), and decryption algorithm for the receiver side will be as seen in Eq. (6):

$$\text{Asymmetric encryption sender side } C = E_{ReceiverPublicKey}(P) \tag{5}$$

$$\text{Asymmetric decryption receiver side } P = D_{ReceiverPrivateKey}(C) \tag{6}$$

More security services are needed to provide a comprehensive security for a communication network. In the following section, these additional services are introduced.

### *3.1.2 Integrity*

While confidentiality focuses on the secrecy of data, integrity overlooks this concern and is primarily concerned about the trustworthiness data, i.e., the data is not changed by unauthorized entities. Consider a banking transaction where 1000$ is transferred from one account to another. The identity of sender and customer may be known to those who are processing this transaction. So, confidentiality is not a concern. Yet, the fact that the 1000$ amount should be transferred fully and correctly is the upmost concern and should be addressed via integrity.

In a network setting, integrity is achieved by implementing a hash algorithm on data to be transmitted. A hash algorithm H is an encoding function that takes an input (plaintext) P of any size to yield an output D (digest) of fixed size as seen in Eq. (7):

$$D = H(P) \tag{7}$$

Implementing a hash algorithm on input data generates a message digest, which is then transmitted to the receiver along with the plaintext. It should be noted that for a cryptographic system where integrity is the only concern, confidentiality is disregarded and plaintext can be sent in the clear. Having received the plaintext, the receiver applies the same hash function (that should have been agreed upon beforehand) on the plaintext and compares the computed digest with what has been received. If they match, it means that the integrity of data has remained intact. Otherwise the data has been compromised in transit.

In order for hash function to perform properly, it should have all two of the following properties:

- One-way property: given a digest $D = H(P)$, calculating plaintext $P = H^{-1}(D)$ should be computationally difficult. To accomplish this goal, one-way functions exploiting computationally hard problems (such as discrete logarithm problem) are designated as hash functions.

- Weak collision property: for two different plaintexts as $P_1$ and $P_2$, where $D_1 = H(P_1)$ and $D_2 = H(P_2)$, the probability of $D_1 = D_2$ should be very small. It should be noted that regardless of plaintext size, the digest has a fixed length. So, a collision is more likely to occur than is actually anticipated. Once this is the case, techniques such as double hashing, linked list, etc. are applied to avoid collision with an overhead of extra process and/or space.

Several hash functions having strong hash properties exist to provide integrity of data. Examples of such functions are SHA [7], MD5 [8], etc.

### *3.1.3 Availability*

For confidentiality to provide secrecy, and for integrity to provide trustworthiness of data, availability of this data should be guaranteed first. In network security context, availability is concerned about the entities being available at all times throughout communication. Those entities are information pertaining to the sender and receiver, the data itself that is being sent, and the metadata that is uniquely identifying the data under operation.

Attacks targeting fields of network protocol headers (IP, TCP, MAC headers) may disrupt availability. Denial of service (DoS) is a common type of network security attack that exploits the three-way handshake protocol between a sender and a receiver. Ideally, the three-way handshake should start by step 1 where the sender sends the TCP header with SYN flag field set along with a sender sequence number. Upon receipt of this request, in step 2 the receiver—if available—informs its availability by sending back the TCP header with ACK flag field set, the sender sequence number + 1, and the receiver sequence number to the sender. Finally, in step 3, the sender sends the TCP header with ACK field set, sender sequence number + 1, and the receiver sequence number + 1. Only then the actual communication can start.

When the three-way handshake is compromised, a rogue sender bombards a victim receiver by sending him too many "half-open" connections, i.e., step 1 of the three-way handshake protocol with SYN flag fields set in each with a different sender sequence number. The victim receiver tries to respond to each SYN request by sending back an ACK + SYN response addressing each independent sender sequence number, but as the rogue sender deliberately never sends back an ACK response to these responses, the victim gets overwhelmed shortly as its half-open connection buffer overflows. The solution to DoS attack is to limit and advertise the buffer size for each entity in the network.

Other examples of network attacks targeting availability can be listed as SYN guessing, IP spoofing (by impersonating a legitimate entity), covert channels (by embedding secret data on unused fields of network protocol headers), messing up fragmentation of packages so that they will not defragment correctly in the receiving entity, etc.

A common protection against availability attacks is to use firewalls. They protect systems from outside entities by enforcing strict rules only to allow packets with particular properties (IP number, port number field, etc.).

#### *3.1.4 Authentication*

Authentication focuses on whether the originator of data really is who he claims to be. It is directly associated with trustworthiness, which makes it closely related to integrity. In other words, authentication is origin integrity.

To provide authentication in network systems, digital signature is used. Digital signature is an encoding function that is similar to a hash function, with the additional property of having a key. The key should belong to the originator, as an evidence for proving his identity. A hash function with a key is called message authentication code (MAC). Several MAC algorithms exist, HMAC [9] being the most popular.

In a digital signature scheme, asymmetric encryption is used, where encoding and decoding are named as signing and verifying, respectively. Eqs. (8) and (9) list the signing and verifying algorithms for digital signature.

$$\text{Signing algorithm: } C = S_{SenderPrivateKey}(P) \tag{8}$$

$$\text{Verification algorithm: } P = V_{SenderPublicKey}(C) \tag{9}$$

ElGamal algorithm is a commonly used digital signature scheme. It provides varying levels of authentication based on the choice of key size.

### 3.2 Computer security

The concentration on computer security is on secrecy and privacy of data during processing. Therefore, those security tools and mechanisms introduced for

network security also apply to computer security. In particular the following three security services are needed to deliver a comprehensive security service while processing data:

- Cryptography to help with confidentiality
- Hash functions to provide integrity
- Digital signatures to help with providing authentication

Take, as an example, a banking transaction in which 1.5% interest is added to customer's bank account. It is very important that identity of this account holder should be kept confidential. So, an encryption tool should be incorporated to provide confidentiality. Moreover, the balance amount should remain intact throughout transaction (except when the resulting balance is calculated), and this requirement can be met via hash algorithms. Furthermore, the system should assure at all times that the transaction belongs to the actual owner of the account, but no other account holder. So, a digital signature scheme should also be employed to prevent repudiation, hence to provide authentication.

### 3.3 Database/cloud security

As being one of the probable states information can reside in, storing data usually involves one of two media: database or cloud.

#### *3.3.1 Database security*

When data is stored on a database, the security measures that should be considered comprise of the following:

- Encrypting the data stored so as to provide confidentiality.
- Compressing data so as to occupy less space—additionally, compression offers augmented security as it shuffles data. A large selection of compression algorithms is available. The choice on which algorithm to choose depends on data type (for text data, a lossless compression algorithm should be utilized, while for image files, voice and video lossy compression can be tolerated), space requirements, speed, and compression rate (how much the compression algorithm reduces the input size).
- Prevent against SQL injection attack so that input forms for data entry cannot be exploited to potentially damaging SQL instructions. As an example, if a database system is vulnerable against SQL injection attack, an attacker can "inject" a rogue statement (such as dropping a database table) to execute, immediately following a biased statement (e.g., as simple as "1 = 1") that will always yield true. A suggested method to prevent against SQL injection attack is to use data sanitization (so that data entry will not allow some characters such as apostrophe, etc.), or using stored procedures to execute instructions, rather than exposing forms for easy injection. There exist several forms of SQL injection, and cross-site scripting (XSS) is one of them.
- Secure the database physically so that data can be protected against access from unintended third parties.

### *3.3.2 Cloud security*

Based on NIST's cloud security definition [10], cloud provides one of the three fundamental services:

- Infrastructure as a service (IaaS) allows subscribers to execute any application and OS on the hardware and resources (abstracted via hypervisors) made available by the cloud. Some examples of IaaS type are Amazon Web Services (AWS), Google Compute Engine (GCE), Microsoft Azure, Rackspace, and Cisco Metapod.

- Platform as a service (PaaS) allows subscribers to create their custom applications on the cloud. The cloud makes itself available to its customers by providing tools such as a DBMS, OS, system software, and applications. The examples of PaaS type can be listed as Apache Stratos, Windows Azure, OpenShift, Heroku, Google App Engine, and AWS Elastic Beanstalk.

- Software as a service (SaaS) subscribers sign a service agreement for this service to execute cloud-owned online applications. Some common examples of SaaS type of cloud service are listed as follows: GoToMeeting, Salesforce, Dropbox, Cisco WebEx, Google Apps, and Concur.

Regardless of the type of service cloud provides, security of the cloud should consider boundary security that builds itself on the layered architecture of hypervisors. Moreover, hardware boundary and abstraction boundary definitions are protections should be offered regarding whether the cloud itself is private or public.

It is dramatically important for cloud to isolate individual customers' data so that they will not interfere with other customers' data. So, anonymity is a primary concern. A straightforward technique to provide anonymity is by incorporating encryption.

Moreover, it should be computationally infeasible to extract summary data that will bring together multiple subscribers' data pool, as this may lead to unfair and unethical advantage. As an example, personal healthcare data stored for multiple healthcare providers should not be analyzed easily to extract a conclusion that persons inhabiting in a particular region are more prone to a particular disease as this may cause people from this region be charged higher by insurance companies.

Though security measures are classified and analyzed separately, due to the complex nature of information systems, handling information most of the time involves multiple aspects of security at the same time. For this reason, complex information security systems have been developed and are widely used. Some examples of such systems are Pretty Good Privacy (PGP) [11] for data encryption, integrity, and authentication, Kerberos for secure key distribution, and many more [12].

## 4. Conclusions

This section introduces the three fundamental states of information, namely, information in transit, information in process, and information in storage, and then discusses the security measures pertaining to each state of information. In particular, network security methods to provide security of information in transit, computer security methods to provide security of information in process, and database/cloud security to provide security of information in storage are introduced and discussed in detail.

As the computer technology evolves in rapid speed, security measures will be extended by either improving the existing algorithms or adding more algorithms to the suit.

IntechOpen

IntechOpen

## Author details

Ebru Celikel Cankaya
Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA

*Address all correspondence to: exc067000@utdallas.edu

IntechOpen

© 2020 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

## References

[1] Coppersmith D. The data encryption standard (DES) and its strength against attacks. IBM Journal of Research and Development. May 1994;**38**(3):243-250

[2] Daemen J, Risjemn V. The Design of Rijndael. New York, USA: Springer-Verlag; 2002. p. 1

[3] Bishop M. Computer Security: Art and Science. Boston, MA, USA: Addison-Wesley; 2003

[4] Pommerening K. Kasiski's test: Couldn't the repetitions be by accident? Journal Cryptologia. 2006;**30**(4):346-352

[5] Friedman WF. The Index of Coincidence and its Applications in Cryptanalysis. Walnut Creek, CA, USA: Aegean Park Press; 1987

[6] Vernam G. Cipher printing telegraph systems: For secret wire and radio telegraphic communications. IEEE. February 1926;**45**(2):109-115

[7] NIST SHA-3 Standard. Permutation-Based Hash and Extendable-Output Functions. 2015. DOI: 10.6028/NIST.FIPS.202

[8] Guzman LB, Sison AM, Medina RP. MD5 secured cryptographic hash value. In: International Conference on Machine Learning and Machine Intelligence, September 2018. pp. 54-59

[9] NIST. The Keyed-Hash Message Authentication Code (HMAC). Available from: https://csrc.nist.gov/publications/detail/fips/198/1/final

[10] NIST SP 800-145. The NIST Definition of Cloud Computing. 2011. Available from: http://csrc.nist.gov/publications/PubsSPs.html#800-145

[11] Choi YB, Hunter ND. Design, release, update, repeat: The basic process of a security protocol's evolution. International Journal of Advanced Computer Science and Applications. 2017;**8**(1):355-360

[12] Stallings W. Cryptography and Network Security. 7th ed. New York, USA: Pearson; 2016