Identity and Access Management Implementation

The implementation of access management contains its own challenges. Audits often reveal that the identity management processes used by many organizations are flawed, resulting in many users that have access permissions that they have accumulated over the years that are not aligned with their current business needs. This is a problem where privacy regulations require accountability and tracking of access permissions, and it can lead to financial penalties, security breaches, and embarrassment for the organization.

The idea of an Identity and Access Management (IAM) system is to automate the process and reduce the administrative overhead, while improving reporting and the ability to monitor the access levels granted to users. Some of the features of IAM systems include an automated process for users to request and be granted access to systems, a streamlined process for new users and for password resets.

Decentralized Access Control Management

System administrators and the local LAN administrator originally handled all access control locally. When a user needed access, their manager had to request access from the local IT staff, who would then arrange that access—often days later. This was an inefficient process but since each system and each LAN operated as a separate entity in the early days, it was really the only way to manage access.

As systems and networks merged and were integrated across departments and buildings, the local management of access became untenable. Administrators could not keep up with the workload and effectively manage access for a mobile and diverse workforce. So, automation and integration of access permissions were essential.

There are advantages to decentralized access management. Locally managed access is more responsive to local needs in that a manager can quickly have access adjusted as workloads and staffing require. Access rules and procedures are subject to local direction and oversight, and the responsibility to manage access permissions is managed by the managers closest to the users.

Centralized Access Control Management

Many organizations have migrated to a centralized access control framework, and this has many advantages. With a centralized department that manages access for all staff regardless of work location, access is managed in a consistent manner. Defined procedures can be set up that ensure access is only granted in compliance with centrally defined policies, privacy laws, and standards. This avoids the problems with decentralized access management where each local manager can do things their own way in an inconsistent and unpredictable manner.

However local offices may be frustrated with a procedure-driven centralized process that is slow and unbending. It may not be possible for a local manager to have access controls changed quickly (perhaps because a member of staff is ill) when the centralized process can only promise to address an access change request in 48 hours.

The deployment of centralized access management leads to better reporting capabilities and visibility for senior management into compliance with legislation.

The deployment of centralized access management has also led to the use of single sign on solutions that enable a user to use a single set of identification and authentication credentials to gain access to multiple systems.

Single Sign-on

Single sign on is a very generic term that refers to many very different technologies and solutions ranging from the older script-based single sign on solutions of the 1990s through to Identity as a Service (IdaaS) in the current cloud environment.

This module will look briefly at single sign on from a security perspective and therefore, will not go into extensive depth on how each solution works but will rather focus on the security advantages and disadvantages of various solutions.

As computer systems became more common in the 1990s, the problem of managing multiple IDs became apparent. The average user was managing 12–15 UserIDs, each with their own password rules and expiry dates. Today that problem is even worse by a magnitude of systems and UserIDs. So, the idea of single sign on was born. With single sign on, a user would only have to remember a single UserID and password to be granted access to all of their systems.

Early password based systems were rather insecure. Many of them actually transmitted passwords in cleartext across a network. Examples of this were Password Authentication Protocol (PAP), which was designed to allow a user to connect to a system (often to connect to their local Internet Service provider (ISP)) over a modem-based telephone line. This was understandable over a telephone based network since telephone lines were a little harder to intercept than today’s Internet, but when those services moved onto the Internet, the use of PAP continued—certainly not an acceptable solution.

Other early systems were based on scripts where a front-end program would simulate the steps a user would take to log in to an application. The user would now enter the application using the script that logged them in automatically. The risk again was that many of these scripts also passed the passwords in plaintext over the network.

We use many different single sign on products today. Many of them will store a user’s passwords for them so that the user does not have to remember all of their passwords for multiple systems. The greatest risk with these solutions is the risk of compromise of the password storage area. These password storage areas may be protected by a weak password or passphrase chosen by the user and easily compromised. We have seen many cases where a person's private data was stored by a cloud provider but easily accessed by unauthorized persons through password compromise.

Popular sign on solutions today include Federated Identity Management, web portals, OAUTH, and other similar solutions that allow a person to log in to a common application and then be able to access multiple other systems or applications based on their initial login. This uses protocols such as SAML (Security Assertion Markup Language) to pass security assertions or login credentials between enterprises that trust one another. (Since this course is careful to avoid being vendor-specific, these solutions have not been named here, but we have all seen them as we can access a website using credentials linked to an email or social media provider.)

Kerberos

Kerberos was an early single sign on solution that is built into many products today. It is an interesting example of how single sign on can work.

Kerberos uses symmetric encryption (explained in the next module) to encrypt the exchange of messages between the Users, Key Distribution Centers (KDC), and the Applications. This provides confidentiality of the communications but does not protect against other forms of attack, such as a brute force on the passwords or compromise of the data stored in the KDC.

Kerberos is an excellent example of the greatest risk associated with most single sign on solutions: the central and single point of failure, the KDC. Kerberos requires that all devices on the network use Kerberos to communicate, and if the KDC is unavailable then no one can communicate. Also, a compromise of the KDC would enable an attacker to have unlimited access to everything! This requires replication and physical protection of the KDC.

Identity as a Service (IdaaS)

The emergence of the Cloud has created many new opportunities, cost savings, and benefits for organizations and introduced new challenges related to access control and privacy. When data is stored and processed by a cloud service provider, there are some challenges with ensuring that the necessary access controls and permissions are maintained over sensitive data. This requires Service Level Agreements and ways to provide Assurance to management of compliance with laws and agreements, as we will see elsewhere in the course, but it also requires clear definitions of access controls and limits on the ability of users, administrators (especially the administrators working on behalf of the Cloud Provider), managers, and customers in regard to data access and modification.

One solution for managing access control over both cloud and non-cloud-based systems is through Identity as a Service (IdaaS). This is an Identity and Access Management (IAM) service provided by a third party to manage access permissions.

The main risks associated with identity and access management are based on improperly maintained access permissions where a user has a level of access that is not aligned with current job responsibilities, and access failure through the failure of the access control solution or compromise of the access control system.

Trust

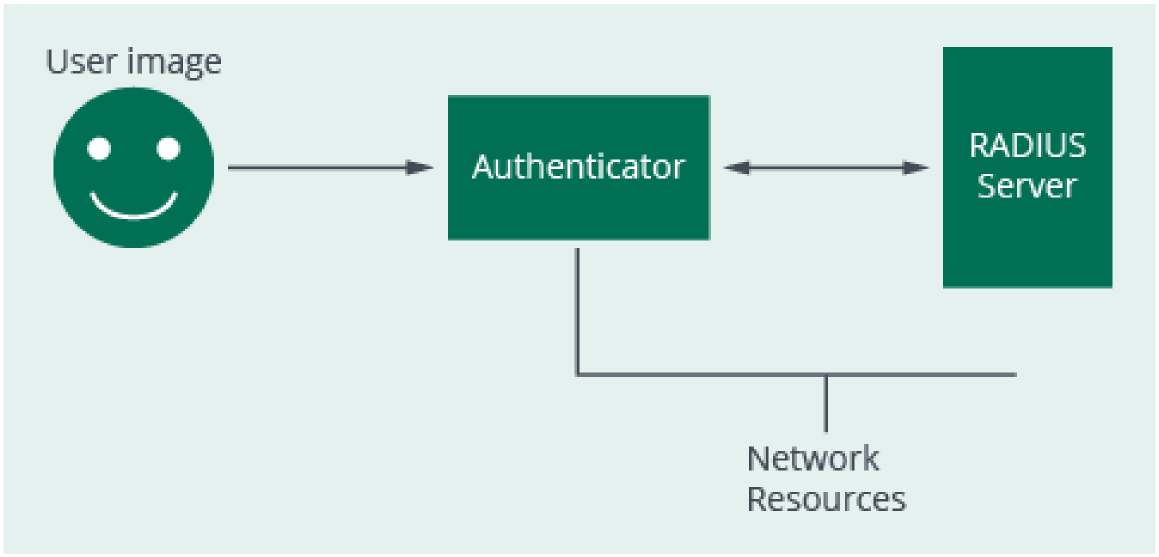
Trust is essential for many data exchanges and e-commerce transactions today. The challenge is to know whom we can trust and to establish a way to build trust. When we are communicating over networks it is difficult to prove whom we are talking to and to ensure that the data we are being provided is legal, authentic, and complete.

When we call a person on the phone, they will often answer with their name or the name of the organization we have reached. This provides us with some assurance that we have reached the right person. We do the same on the Internet. When we reach a website, such as the website of our bank, they often send us a certificate that can be used to validate we are on the correct website and not a website masquerading as the bank. The certificate also contains the public key of the bank and using that key, we know we can communicate confidentially with the bank. This process where we initiate the connection but where the responding website authenticates to me is called ‘reverse authentication’. The respondent authenticates to the originator. If the bank also asks the client that was reaching out to them for a certificate so that both parties had to exchanges certificates, this would be known as ‘mutual authentication’. With node authentication where the calling device has to provide their MAC or IP address to the called site, this is known as ‘forward authentication’. These forms of authentication allow us to establish trust and engage in e-commerce transactions with a level of assurance that we are communicating with the correct organization and not a spoofed website.

Transitive trust can be defined in two ways: it is the extension of trust between two parties to areas outside or beyond the original trust relationship. If Domain A and Domain B trust each other, and Domain C and Domain B trust each other then Domain A could have transitive trust with Domain C; the other approach is used in some directory-based products where a subdirectory will inherit trust from the parent directory.

Another form of access control is based on network protection. This approach will not let a device connect to a network until it has been verified and proven to be compliant with corporate security policies. This can be done using Network Access Control (NAC) devices. When a person tries to connect a device to the network, it is placed into isolation (quarantine) until it has been checked for compliance with security baselines, such as the presence of an anti-virus product, up-to-date patches on applications and operating systems, and appropriate security settings. NAC is especially desirable with a mobile workforce that returns to the office after being connected to untrusted networks (e.g., hotels, coffee shops).

A method of controlling access to networks is through the IEEE 802.1x standard. This is a port-based network access control standard that uses Extensible Authentication Protocol over LAN (EAPOL) to enforce security policies.



The supplicant (user) provides the authenticator with their log-on credentials (username and password) using EAP. The authenticator then communicates with an authentication server using RADIUS or Diameter. The authentication server validates the credentials of the user and notifies the authenticator that access can be granted (or denied) to the requested network resources to the supplicant.

RADIUS (Remote Authentication Dial In User Service) is a networking protocol to support Authentication, Authorization, and Accounting (AAA) services for network access. It is commonly used to manage network access as a centralized access control solution.

DIAMETER was developed to overcome the limitations of RADIUS and support a mobile workforce with a reliable AAA network access solution. Another common solution used for managing network access is TACACS+ (Terminal Access Control Access Control System).

Summary

Access controls lie at the very heart of an information security program. Access controls apply to all relationships between subjects (users) and objects (resources: buildings, networks, applications, data, personnel), and it is critical that we protect those resources from improper access.

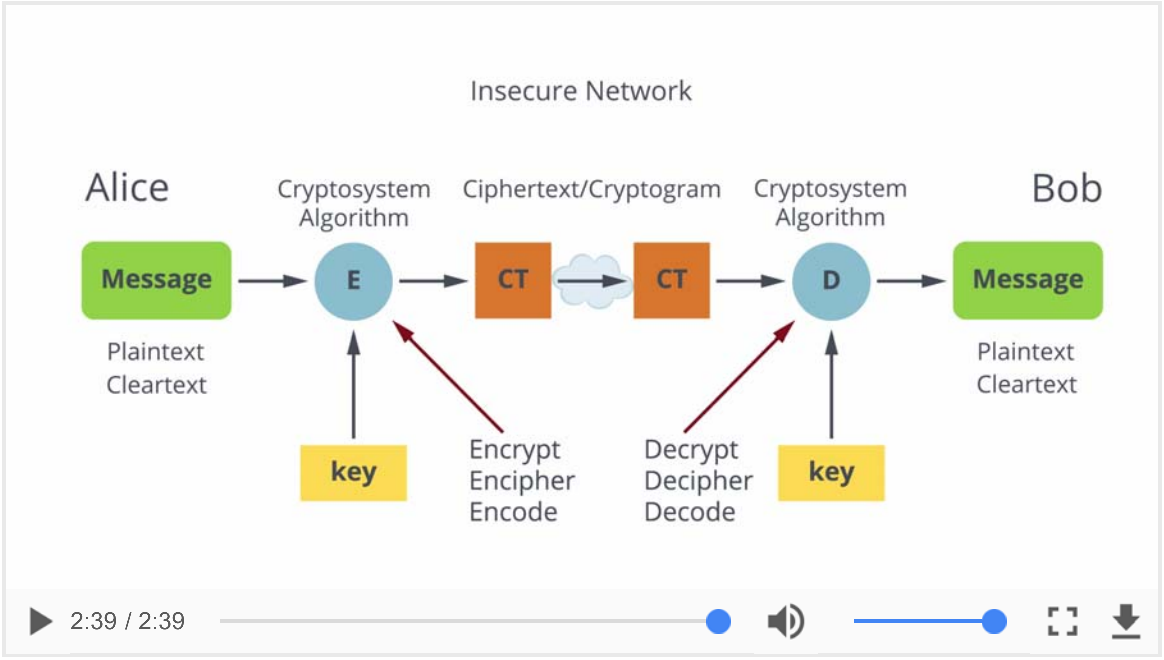
Encryption Overview

Encryption is a fascinating world of history, mathematics, creativity, and science. Encryption has been used a tool to preserve secrets (confidentiality) for thousands of years, from merchants protecting intellectual property, to military geniuses communicating with their armies.

In reality encryption is quite simple and straightforward, and this module will explain some of the basic concepts of cryptography.

Historically, cryptography was a manual process performed by the writer, but through the years it became a mechanical and electro-mechanical process using crypto devices—the Enigma machine, the M209 etc. Nowadays, cryptography is just a computer program built onto a chip or run from an application.

In this diagram, we can examine the terminology and process of cryptography.



▼ Transcript

Let's examine how cryptographic process works.

Alice wants to send a message to Bob. That message, in its natural form, is clear text, or plain text. Since sending the message to Bob will require sending the message over an insecure channel, Alice must protect the message from disclosure.

The way to protect the confidentiality of a message is to encrypt it using a cryptosystem. A cryptosystem contains an algorithm, the mathematical function, which will encrypt the message and alter the format of the message. This process is based on two primary functions. The substitution of one plain text value for another, or transposition, which is the changing of the order of the plain text letters around so the words do not make sense. Modern cryptographic algorithms often do both substitution and transposition. Transposition is also known as permutation.

Alice inserts the secret key she shares with Bob into the cryptosystem. The key is also known as the cryptovariable. The process performed in the cryptosystem is to encrypt, encipher, or encode the plain text. These are not all the same, but we don't worry about that level of detail here. The result of encrypting a plain text message using the key and the cryptosystem is cipher text of the message. This is also known as a cryptogram. Now the cipher text to the message can be sent over an insecure channel. Bob receives the cipher text of the message, and he feeds it into the same cryptosystem at his end. Bob also needs to insert the correct key in order to decrypt the message from Alice.

The decryption process is the inverse of the encrypt process. The decryption, with the correct key, would generate the plain text that Bob can read. Even if someone on the insecure network captured a copy of the cipher text, they would not be able to read or understand the message, since they would not have access to the correct key.

Cryptographic operations are based on two simple activities: substitution and transposition. Substitution is the process of replacing one value for another. An example of that process was the cipher developed by Julius Caesar. When he wrote a message, he substituted the letter he wanted with the letter three places further down the alphabet.

An example of that is:

To encrypt the message FACE

```
If the natural alphabet is:        A B C D E F G H  . . . Z

Use an alphabet shifted 3 places   D E F G H I J K  . . . C
```

The plaintext of the message would be FACE

The encrypted message would be IDFH

The secret to unlocking and reading the message would be to know the KEY—the value used to define the substitution alphabet. In this case, the key would be 3.

The other type of cryptographic operation is based on transposition, changing the position of the letters, for example, write the plaintext word FACE as CFEA. With transposition we still use the same letters, just in a different order.

Early encryption systems were usually based on either substitution or transposition, however, encryption systems of today do multiple rounds of transposition and substitution. The Advanced Encryption Standard (AES) does ten (or more) rounds of substituting and transposing the input message to generate ciphertext.

The hardest part of understanding encryption is learning the terminology. Here are the terms we will use in this course:

Plaintext/Cleartext:

> the message in its natural format

Ciphertext/Cryptogram:

> the encrypted message that could not be understood without having the ability to decrypt the message

Encrypt/Encode/Encipher:

> these are not exactly the same, but for our purposes we will use these terms without defining the differences between them. These terms describe the process of changing plaintext to ciphertext.

Decrypt/Decode/Decipher:

> the process of changing ciphertext into plaintext.

Cryptosystem:

> the device or process that is used to encrypt or decrypt a message

Algorithm:

> the mathematical process used to convert a message from plaintext to ciphertext and back again

Key:

> also known as the cryptovariable. The variable used by the algorithm during the encryption/decryption process. Often this can be a secret password, a passphrase, or pin number chosen by the person encrypting the message.

Man-in-the-Middle (MITM):

> a person that inserts himself or herself into the communications channel to intercept the communications. They may want to 'sniff' the traffic to listen in (a passive attack), or they may want to insert new data, modify the communications, delete traffic, or replay previous messages (an active attack).

Benefits of Cryptography

For thousands of years, the primary benefit of cryptography has been to ensure the confidentiality of messages being sent over an untrusted network between two people. However, in the past few decades the development of improvements in cryptography and the development of new cryptographic algorithms has brought many new benefits from the use of cryptography. The benefits of cryptography today include:

- Confidentiality
- Access control
- Integrity
- Authentication
- Non-repudiation

Each of these benefits will be examined later in the course.

Cryptography is used in protecting sensitive data in storage or when being communicated. The use of disk encryption, encryption of data stored in databases and displayed in applications protects stored data. In addition to encryption, displayed data can also be protected by obfuscation and masking the data or using screen filters to avoid "shoulder surfing" attacks. Shoulder surfing was originally the act of peering over a person's shoulder but today the use of cameras directed at PIN pads or the screen of a user can result in the disclosure of sensitive data.

Cryptography is also used to protect data traveling across an untrusted network; there are many encryption products and protocols in use such as IPSec, SSL, TLS, and S/MIME to protect data in transit. These will be examined in more detail later in the course.

Legal Issues Related to Cryptography

Cryptography has long been considered a weapon of war as well as a commercial tool. This makes it a "Dual Use Good" and in many countries the use and distribution of cryptographic algorithms has been controlled by law. Laws related to encryption were intended to keep encryption algorithms out of the hands of criminals or enemies and often based on controls over the export of encryption or use of encryption under certain circumstances.

On the other hand, regulations today often require the use of encryption to protect sensitive data. Industry standards such as PCI-DSS (Payment Card Industry–Data Security Standards) also require that sensitive data such as payment (credit/debit) card numbers are “rendered unreadable” when stored and that access to sensitive data be restricted according to “business need-to-know”.

User Training

Most encryption systems in use today require very little user interaction. Most users are using encryption for online banking, emails, and ecommerce transactions without really even realizing that encryption is being utilized. The ease of use of crypto has made it possible for everyone to benefit from secure communications, but it has also had the effect of leading to serious breaches of sensitive data when users do not follow best practices in the deployment and use of encryption. The most important responsibility of a user is to protect the keys they use, create strong keys and not share them with other people. This is where training of users is critical to ensuring that the benefits of cryptography are realized.

Summary

In order to understand the next few modules on cryptography, it is essential to be familiar with the terminology. The security practitioner should appreciate the uses and benefits of cryptography so that it can be implemented effectively.

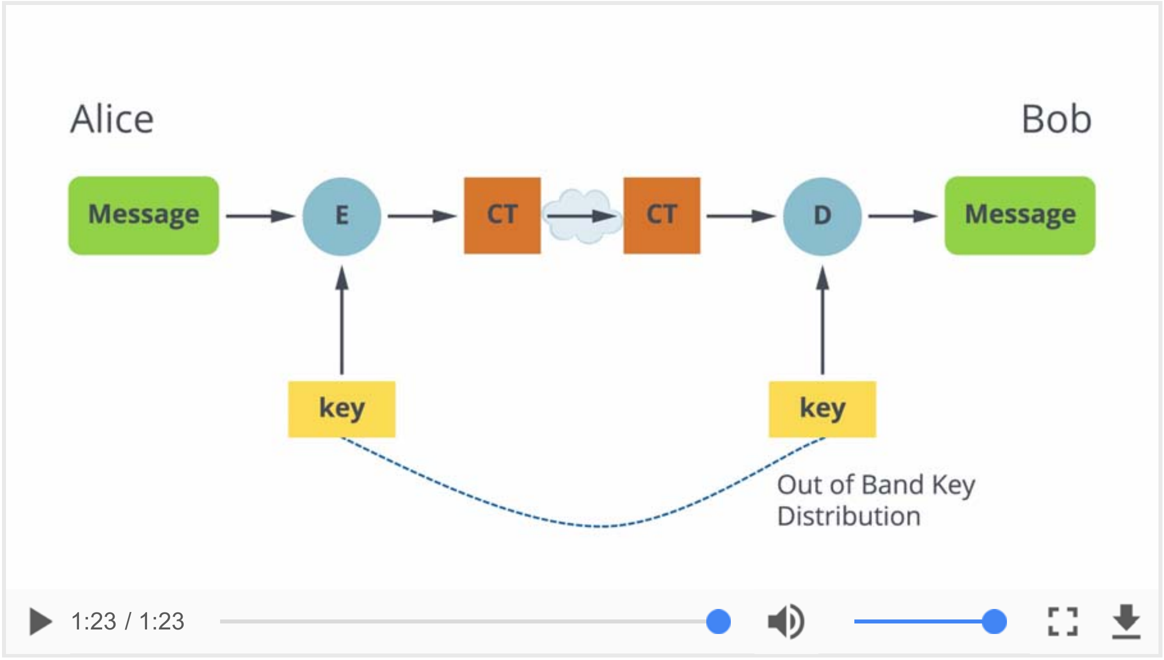
Cryptographic Algorithms

There are two primary types of cryptographic algorithms: symmetric and asymmetric. The security practitioner should be familiar with the strengths and weaknesses of each type of algorithm and how the algorithms can be used effectively.

Symmetric Algorithms

Symmetric algorithms are the traditional type of algorithm that has been in use for many years. The central characteristic of a symmetric algorithm is that it uses the same key in both the encryption and the decryption process. It could be said that the decryption process is just a mirror image of the encryption process.

Symmetric algorithms work as shown in the diagram below



▼ Transcript

Alice has a confidential message she wants to send to Bob through an untrusted channel. Alice feeds the message into the chosen cryptosystem, in this case perhaps using the advanced encryption standard or AES algorithm.

Alice feeds the symmetric key she shares with Bob into the cryptosystem. This generates ciphertext of the message. The ciphertext can be sent through the untrusted channel, and Bob receives the ciphertext. Bob feeds it into the same cryptosystem, using the same algorithm and the same key. Since the key cannot be sent over the untrusted channel, another method of key distribution is required, perhaps via courier, fax or phone. This alternate channel of key distribution is called "out of band". The message can now be decrypted. Since the same key is used in both the encryption and decryption process, this is known as symmetric incryption. Both the sender and receiver share the same single, secret key.

The same key is used for both the encryption and decryption processes. This means that the two parties communicating need to share knowledge of the same key. This type of algorithm protects data since a person that does not have the correct key would not be able to read the encrypted message. Since the key is shared, this can lead to several other challenges:

- Distribution of the key is difficult since the key cannot be sent in the same channel as the encrypted message or the man-in-the-middle would have access to the key. Sending the key through a different channel (band) than the encrypted message is called “out-of-band” key distribution. Examples of out-of-band key distribution would include sending the key via courier, fax, or phone.
- Any party with knowledge of the key can access (and therefore change) the message. This means it is difficult to prove what the initial message was or who made the changes. This brings into question the authenticity of the message.
- Each group of people that want to communicate would need to use a different key than they use for other groups. This raises the challenge of scalability—the number of keys needed grows quickly as the number of users increases. The formula for the number of key is
  $n(n - 1)/2$, where n is the number of users. An organization of 1000 employees would
  need to manage 499,500 keys.

Benefits of Symmetric Algorithms

There are many benefits to using symmetric algorithms for encryption. The main benefit is that symmetric algorithms can provide an excellent level of confidentiality and protection of sensitive data from unauthorized disclosure. The United States Government mandates the use of the symmetric algorithm-based standard of AES (Advanced Encryption Standard) for use in all government agencies to protect sensitive data.

Benefits:

- Excellent for confidentiality
- Can provide some integrity (HMAC, Keyed Hashing)
- Often freely available—many algorithms (including AES) are not patent protected
- Fast—encrypts messages relatively quickly

Disadvantages:

- Key management including scalability, key distribution
- Weak for authentication or non-repudiation

Primary Uses:

- Encrypting bulk data (backups, hard drives, portable media)
- Encrypting messages traversing communications channels (IPSec, TLS)

Examples of Symmetric Algorithms:

- DES, 3DES
- AES
- Blowfish/Twofish
- MARS
- Serpent
- RC 4, 5, 6

Other Names for Symmetric Algorithms:

- Same key

- Single key
- Shared key
- Private key
- Secret key
- Session key

The Properties of Exclusive-OR (XOR)

A central part of most encryption processes is the use of binary addition known as Exclusive-OR (XOR). Exclusive-OR is a fast mathematical operation a computer processer can perform, and it lends itself to the data encryption process.

XOR operates on the principle that when two similar binary values are added together the output will be zero. If the two binary values to be added together are different then the output will be a one.

Let’s look at an example:

```
A binary byte for lowercase ‘a’ (in ASCII) is 01100001

A key-based value is added to it of           11110101

The XOR result would be a double quote “      10010100
```

If this process was being used to encrypt the message of ‘a’, the value of ‘a’ would be substituted with a double quote in the ciphertext.

Symmetric algorithms use the XOR operation extensively to encrypt and decrypt messages because it is a fast method of substitution.

Stream versus Block Algorithms

When we talk on the phone, our voice is a stream of data being passed between the two parties. If we want to encrypt that data, then we want an encryption process that operates quickly and does not break up the conversation or cause delays in processing. For this we use a stream–based algorithm that will encrypt very quickly. An example of a stream-based algorithm is RC4 (Rivest Cipher 4), which was used in WEP, WPA, and many other encryption implementations. Stream-based algorithms are often implemented on hardware, and they provide fast, secure encryption often unnoticed by the end-users. Stream-based algorithms usually will encrypt each bit or each byte as it is transmitted using a keystream that is generated by the original key the users submitted.

Other symmetric algorithms will encrypt an entire block of data at a time. AES, for example, accepts input data in blocks of 128 bits in size. Then it performs multiple operations of transposition and substitution on that block of data as it converts it into ciphertext.

A block mode algorithm can operate in multiple modes either as a block operation, or it can simulate a stream-based operation. The two primary block modes are ECB (Electronic Code Book) and CBC (Cipher Block Chaining). CBC is the most common mode of implementation for a symmetric block mode algorithm. The three modes of operation where symmetric block algorithms act as if they were a stream-based algorithm include CFB (Cipher FeedBack), OFB (Output Feedback), and CTR (Counter). If a person uses WPA2 (Wi-Fi Protected Access 2) for secure wireless communications, they are using AES-CCMP (Advanced Encryption Standard – Counter Cipher Block Chaining Message Authentication Code Protocol), which is a combination of the block mode operation of CBC (to prove message integrity) and the stream mode operation of CTR (to provide message confidentiality).

Session Keys

Symmetric algorithms are excellent for sending confidential communications over untrusted networks, but they are weak in other areas such as key management, non-repudiation and authentication. Therefore, symmetric algorithms are often used in combination with asymmetric algorithms as a hybrid implementation of both types of algorithms. In a hybrid implementation, the symmetric key used in encrypting the confidential message is generated and used for a single communications session and then it is discarded. For this reason, the symmetric keys used for that single use are called “session keys.”

Asymmetric Algorithms

Symmetric encryption has been in use for many thousands of years, but asymmetric encryption is relatively new, becoming publicly available only in the 1970s. Asymmetric cryptography uses a pair of keys instead of using a single key like symmetric cryptography. The pair of keys used in asymmetric are mathematically related and must always be used as a pair. One key will not work without the other key also being used. The key pair is comprised of a private key that the owner of the key pair MUST keep private and a public key that can be shared with anyone the owner wishes to share it with. The mathematics used on creating the key pair makes it simple to calculate the value of the public key if a person knows the value of the private key, but it is computationally infeasible to be able to determine the value of the private key based on the value of the public key.

Asymmetric cryptography is commonly called Public Key Cryptography because it uses public keys as well as private keys, and it is often implemented as a Public Key Infrastructure (PKI) as will be explained later in the course.

Asymmetric cryptography provides at least five primary benefits:

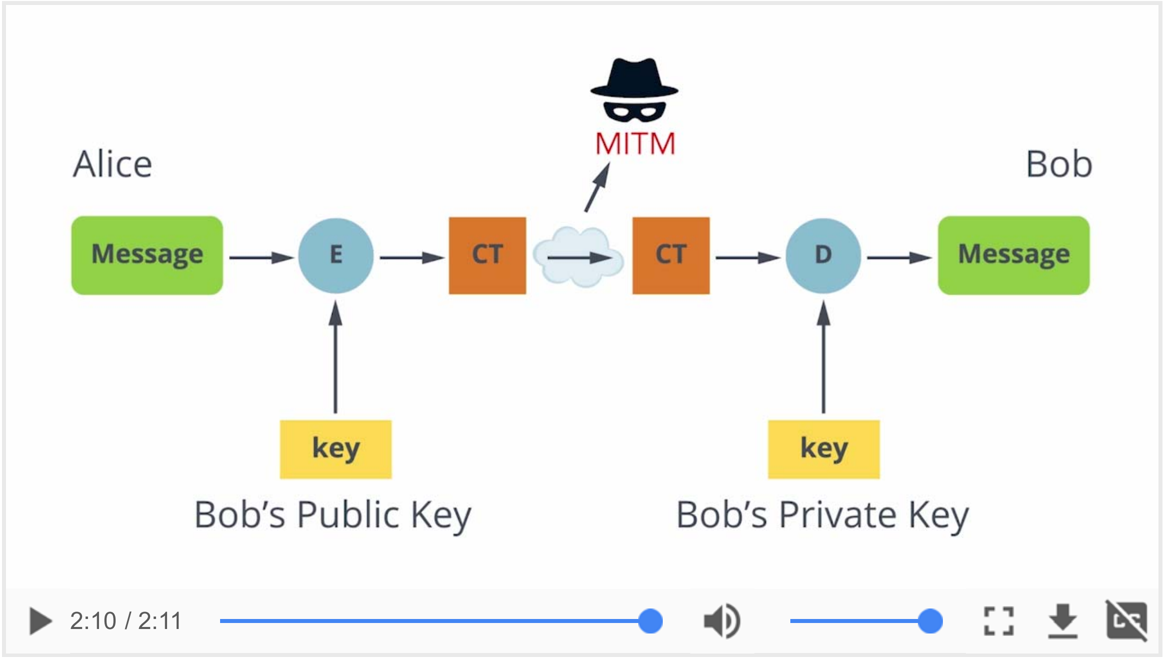
- Confidentiality
- Access Control
- Integrity
- Authentication
- Non-repudiation

Examples of Public Key Algorithms include:

- Diffie-Hellman
- RSA
- ECC

Use of Public Key Algorithms

Confidentiality:
From this diagram we can see how public key encryption can be used to send a confidential message across an untrusted channel.

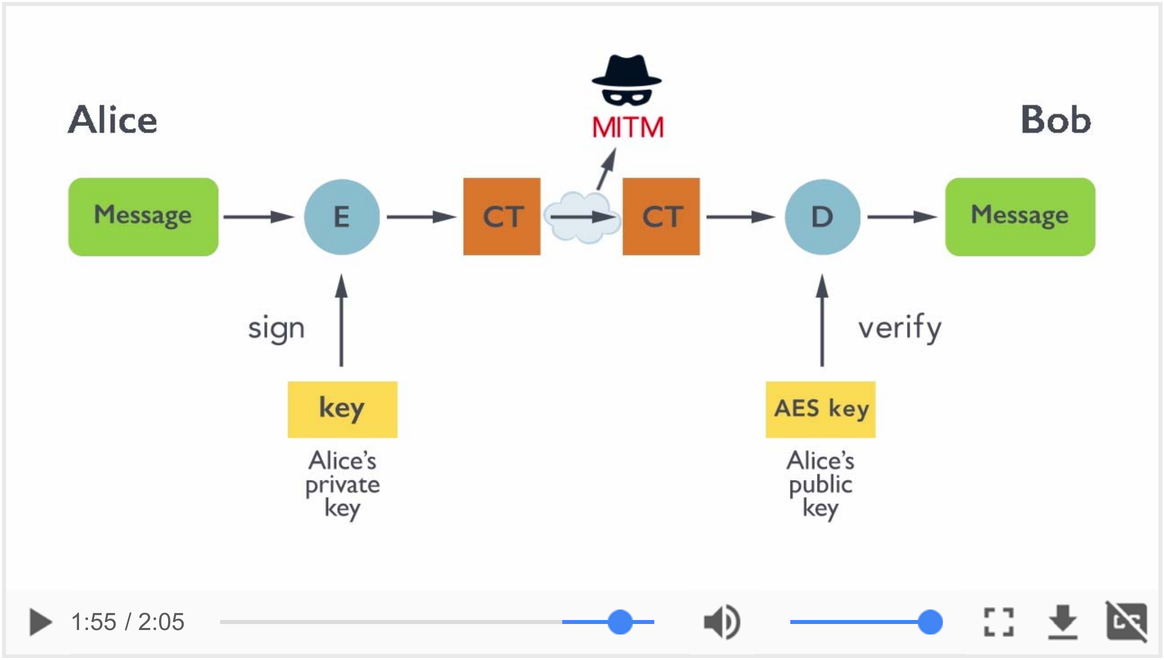


▼ Transcript

Let's look at how asymmetric encryption can be used to support the principle of confidentiality. Let's say, Alice has a sensitive message she wants to send to her friend, Bob, over an insecure channel, such as the Internet. Using asymmetric cryptography, Alice can feed that message into a cryptosystem where we will then encipher, or encrypt, that message. Alice would use Bob's public key. She may have received that public key as a certificate or some other type of key distribution mechanism. When she encrypts her message with Bob's public key, the result will be ciphertext. That ciphertext can be sent securely through that insecure medium. When Bob receives that ciphertext, he knows to feed it back into a cryptosystem. The cryptosystem will then decipher, or decrypt, that code, and it'll do it using Bob's private key. When it decrypts that ciphertext, it will then give Bob access to the plain text message that Alice wanted to send. In asymmetric cryptography, the keys always, and only, work as a pair. If a message was encrypted with a public key, it must be decrypted with a corresponding private key. Since Bob would never share his private key with anyone else, he knows this message has been sent confidentially through this insecure medium, and even if someone had captured a copy of that, what we'd call a man in the middle attack, they would be unable to read the message, supporting the principle of confidentiality.

Proof of Origin

Another excellent benefit of asymmetric algorithms is to be able to establish authentication of the sender of a message and thereby provide proof of origin for the message—the receiver knows who the message came from.



▼ Transcript

Let's look at how asymmetric cryptography can be used to support proof of origin. Let's say, for example, Alice has a message she wants to send to Bob. She needs to prove that message came from her. For example, a contract or an order for a product. She has that message and she encrypts that message using a crypto system based on asymmetric key cryptography and she would use her own private key to encrypt that message. That creates ciphertext of that message. She can send that ciphertext through an insecure, unreliable channel and when Bob receivers that ciphertext, he will feed it into the crypto system to decrypt or decode, decipher that message. He will have to use Alice's public key. When he does so, he will then have a copy of the message that Alice sent. With asymmetric key cryptography, we know that the keys always and only work as a pair. The only key that can be used to decrypt this message is Alice's public key. Therefore, Bob knows it had to have been encrypted with Alice's private key. It had to have come from Alice.

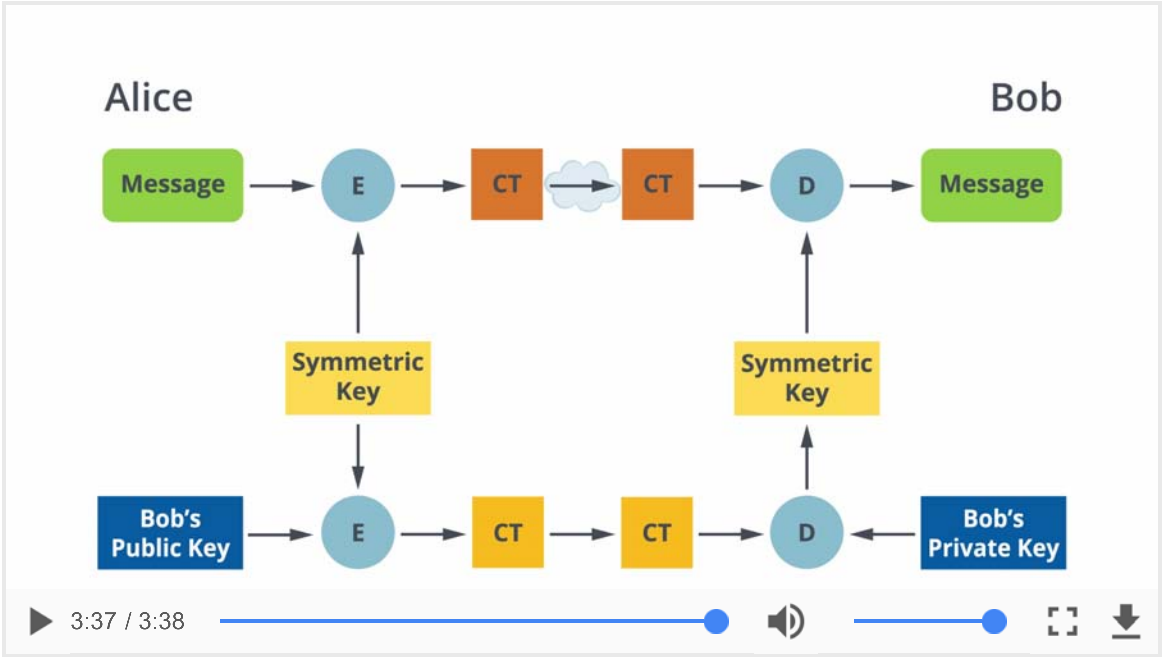
In this case, even if we had a man in the middle attack, where the person captured a copy of that message, they would be able to read it if they had Alice's public key, but they could not alter that message and even though it did not provide confidentiality, it does provide and support the concept of proof of origin.

Disadvantages of Asymmetric Algorithms:

- SLOW—many times more computationally intensive than symmetric algorithms

Hybrid Encryption Implementations

So, we have two types of encryption algorithms: symmetric algorithms that provide fast and confidential transmission of data but have a disadvantage of difficult key distribution and scalability; and asymmetric algorithms that can transmit a message confidentially but are really slow. The solution is to use the strengths of each type of algorithm to support the other type. This requires the use of symmetric algorithms to encrypt a message quickly and the use of asymmetric algorithms to send the symmetric key to the recipient that is needed to decrypt the secure message at the far end of the communications channel. We can see how a hybrid system works here:



▼ Transcript

We saw how asymmetric encryption can be used to send a confidential message between two parties by encrypting the confidential message with the receiver's, that is Bob's, public key. The problem is that asymmetric encryption is incredibly slow and not practical for encrypting a large message. On the other hand, we know that symmetric encryption is ideal to quickly encrypt large messages and maintain confidentiality of the message, but symmetric encryption is plagued by the problem of key management and, specifically, key distribution. Since both parties, Alice and Bob, would need a copy of the symmetric key, there needs to be a secure way to send the symmetric key from Alice to Bob without anyone else getting a copy. This is where we can use the advantage of each type of encryption to work together in a hybrid model. Let's use symmetric encryption to provide confidentiality of the message, and asymmetric encryption to provide for a secure key distribution, to send the symmetric key confidentially from Alice to Bob.

Alice starts with a plaintext of the message that she wishes to send to Bob. Alice feeds that plaintext of the message into a cryptosystem that uses a symmetric algorithm such as the Advanced Encryption Standard, or AES. Alice's system selects a secret value to be used as the symmetric AES key. This key is then used to encrypt the message. This generates ciphertext of the confidential message Alice wants to send to Bob. This ciphertext can now be sent across an insecure network.

When Bob receives the ciphertext, he will feed it into a cryptosystem based on the same symmetric algorithm, but in order to decrypt the ciphertext, Bob requires the symmetric key that Alice used to encrypt the message. In order to securely send the symmetric key to Bob, we need to go back to what happened in the cryptosystem when Alice encrypted the message.

The cryptosystem sent the symmetric key it used to encrypt the message to another process that will now encrypt that symmetric key using asymmetric encryption. To send the symmetric key confidentially to Bob, we know that we would encrypt it with Bob's public key. This creates ciphertext of the symmetric key. The ciphertext of the symmetric key can now be appended to the ciphertext of the message. We will send the encrypted message and the encrypted symmetric key together. This is known as a digital envelope. The ciphertext of the symmetric key can now be decrypted using Bob's private key. Using the symmetric key, the ciphertext of the message can now be decrypted.

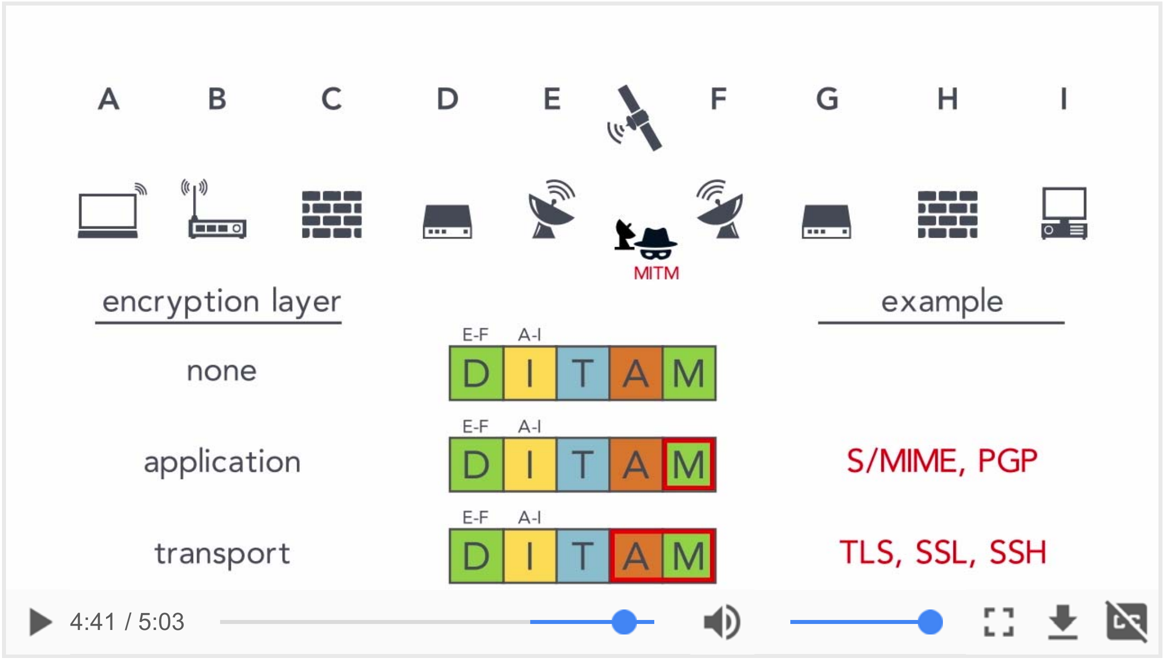
Since the symmetric key is small (probably only 128 bits in length), it can be encrypted quickly by the asymmetric algorithm. This allows the sender to send both the encrypted message and the encrypted symmetric key together to the recipient in what is known as a “digital envelope.”

In fact, we see that most deployments of asymmetric algorithms are used to support an implementation of symmetric algorithms, or for proof of origin as will be examined in more detail later in the course.

Diffie-Hellman Algorithm

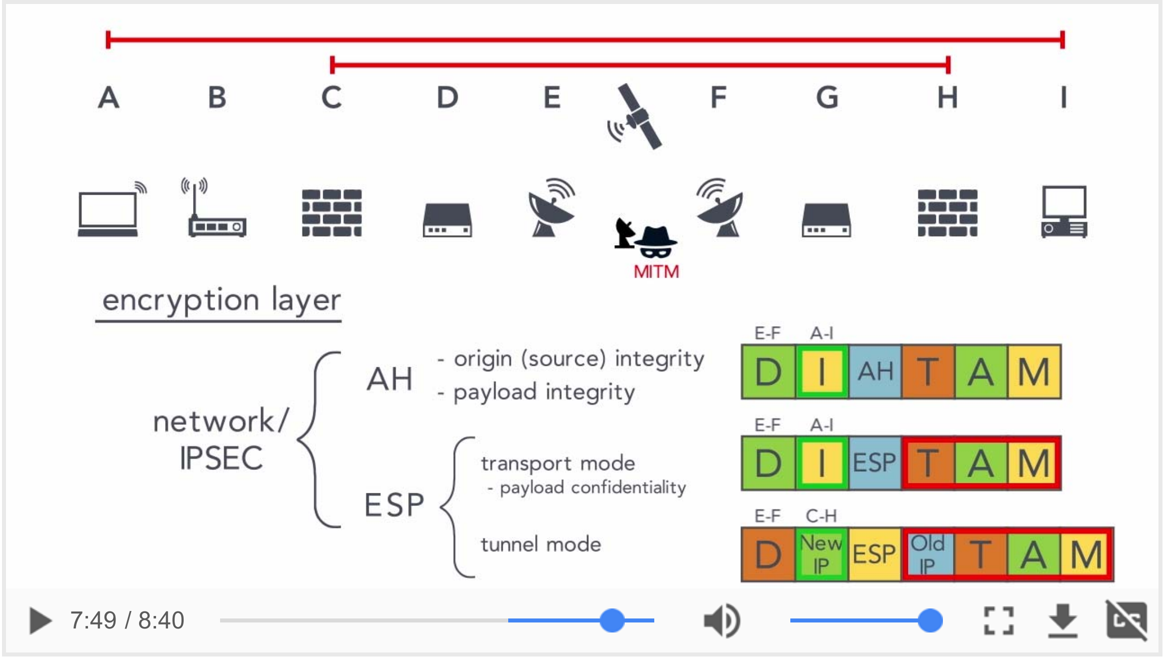
The Diffie-Hellman (D-H) algorithm is used differently than RSA or ECC. As seen in the previous diagram, RSA and ECC are often used to encrypt and transmit a symmetric key to the recipient. D-H is used instead to negotiate the value of the symmetric key between the sender and receiver. Instead of the sender of a message choosing the symmetric key to be used as done with the other asymmetric algorithms, with D-H each participant exchanges their public key that is then used to negotiate the value of the symmetric key. This is a common practice with an implementation of IPSec.

Video: Encryption



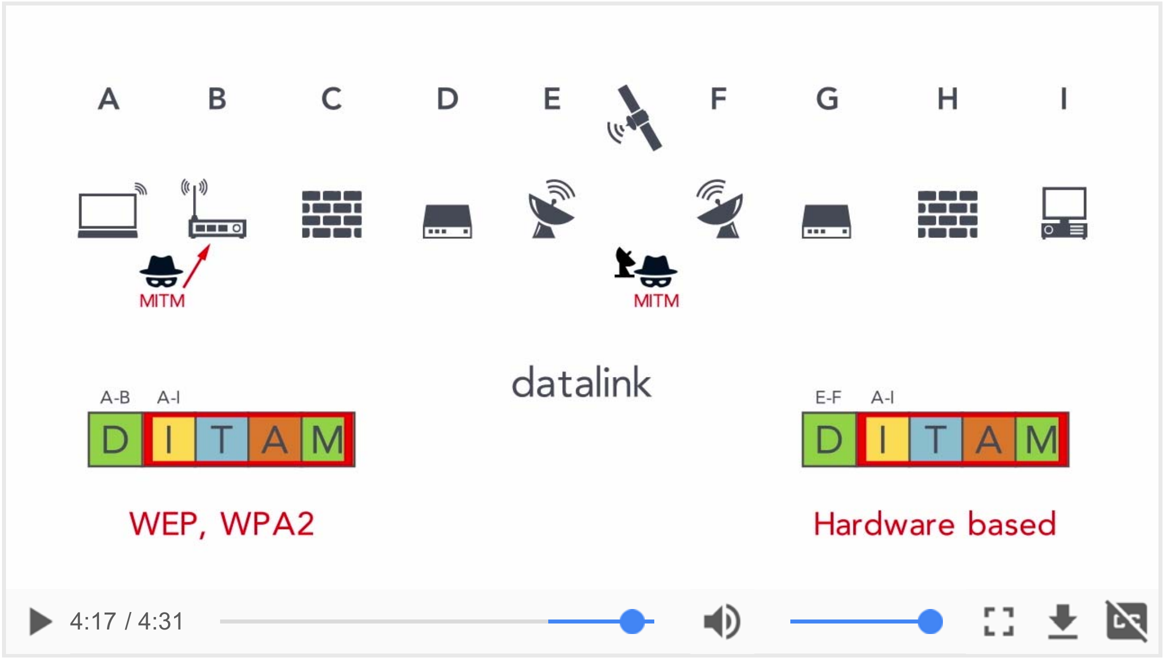
▼ Transcript

Let's take a look at where we can do encryption at all of the layers of the OSI stack and some of the encryption options we have. We see here a simple network, a person with a laptop talking through a wireless access point through a firewall through a router to then a router that is routed up to a satellite uplink picked up by a ground station that router point F, another router, a firewall, and finally the host server at the far end. We know that any communications off of a satellite can be picked up over a very large satellite footprint area. And if we had, for example, a man-in-the-middle attack here where someone was intercepting that communication, what would they actually see? The packet they would see would look like this. We would have the message or the data itself. In front of that would be our application header. Then we'd have our TCP header, our IP header, and our data link header. The data link header in this case which showed that the data is originating at point E and going to point F. Our IP header would show that traffic is originating in point A and ending at point I. So, our man-in-the-middle would be able to see everything about this communication, who was talking to whom, which network link is currently on and monitoring, the type of traffic, and of course the data itself. This would be unacceptable if we had some form of confidential communications. So, what do we need to do? We need to look at the ways we can encrypt this traffic to go across this untrusted network. We can encrypt at the application layer. For example, we could use products like Secure Multi-purpose Internet Mail Extension or PGP as an example. And if we do that, what would our packet look like now? Our packet instead would look like this. If we're continuing to monitor this point, our data link header would still be the same, our IP header would still be the same. We'd have our transport header, our application header, and then we would have the data itself with the exception now that the data itself would be encrypted. So that the person monitoring the traffic here knows who's talking to whom, they know the type of traffic it is, is it FTP or SMTP for example, but they could not see the actual content of the message. And, that's how we encrypt at that layer. We could also encrypt at the transport layer. Now at the transport layer, we have a number of encryption options we could use. The most obvious being transport layer security or TLS. Formally of course called SSL, in it's old iteration the idea of course of secure socket layer. Now, you could argue with me on this, but it's not worth it. Because we also have in here as well we could put in Secure Shell and Secure Shell two. As I say, you can argue over that. maybe I should show it at a higher level, but that's a debatable point and not really important for us at this point in time. What would our packet look like if we're using TLS? With TLS, we would still have our same data link header information. Traffic is going from point E to point F. We would still have the correct IP address information. We have our transport header, and beyond that we would have our application header and the message or data we're sending. In this case with TLS, we will encrypt everything including the application header. So, when you're doing online banking and using TLS, then that person who is monitoring the traffic could still see that you are talking to the bank. They could see you're using TCP, but they would not be able to see the actual application header information or the actual data you're sending back and forth.



▼ Transcript

Let's continue looking at how we can do encryption at the various layers of the OSI, or TCPIP stack. We can move down to the IP layer. At the network layer, we're going to use IPSEC, or IP Security. A part of the IP version six project that was separated off because of the need to provide network layer security. IPSEC operates in two different ways. First of all, with Authentication Header or AH for short. With Authentication Header, we will actually take our packet. And it'll look like this. We'll have our data link header. If we're monitoring between E and F, it'll show the data link addresses of those two adjacent points. We'll have our IP header, A to I for example. But then we are going to insert here an Authentication Header. Then we will have our transport header, our application header, and the actual payload, the message content we're sending. What does the Authentication Header do? The Authentication Header authenticates, or validates, the information in the IP header. So it says, this information here in the IP header is correct. We know it truly is coming from point A, and we know it has not been altered because of a checksum or integrity checks within the IP header. So the benefits of Authentication Header are exactly what its name says. We have here origin integrity. You could call that source integrity. And we have then payload integrity. We have authenticated the information in the IP header. Now in order to set this up, we would also have to set up something we'll call a Security Association, or SA for short. And the Security Association would be, well, you could call it almost a certificate, or a way to set up that associated relationship between the two endpoints of this communication. In the essay, we would identify this communication session with an SA, say number five or number six as an index for it. We would also identify the type of protocol we're using, such as AH. And we would then identify the source IP address so that we have a trusted relationship between the two endpoints of this communication. We have integrity of the packet. It has not been altered by anybody on the way by. And it has not been spoofed. It did not come from point Q pretending to be point A, for example. So that's a benefit of Authentication Header. But most people that use IPSEC are much more familiar with ESP, Encapsulating Security Payload. And with ESP, we're going to look at the two main methods of transmitting or communicating using ESP. First of all, we'll have what we call transport mode. And in transport mode, our packet will look like this. We'll still have our data link header as always. We'll have our IP header. And then, we'll insert our ESP header. Then, our transport header, our application header, and the actual payload itself. What we would do with ESP in transport mode is everything we did with Authentication Header. We'll authenticate the IP header. So, we will prove origin or source integrity, and payload integrity. It has not been changed or altered. But in addition to that, we will now add in payload confidentiality. And we will encrypt everything from the transport header in, so that when we're using IPSEC, and if you're using, say for example, an IPSEC virtual private network, a person who was monitoring your traffic could still see who was talking to whom. A is talking to I. It's still going over this data link. But they would not be able to see the type of traffic nor the content of the message. So that's ESP in transport mode. And what we've done with this is we've actually set up end to end encryption. We've encrypted the traffic right from point A right through to point I. But there's another thing we'll often do. Very often with IPSEC, we're gonna set up network to network encryption. We're gonna set up a tunnel between say two firewalls. The firewall from the branch office to the firewall in the head office. Or, from a user with a laptop in a hotel room to the VPN concentrator at the firewall. So we didn't encrypt right to the end. And this is what we'll call, therefore, tunnel mode. In tunnel mode, our packet will now look like this. We will still have our data link header. Then, we'll have a new IP header. And let's say that we're gonna describe the tunnel from C to H. The new IP header would show the IP addresses of the two ends of the tunnel, from C to H. Then we would have our ESP header. Then our old IP header, our transport application, and our message. So what would this do? This would take our traffic that when it reached point C would now be encapsulated, or wrapped into a separate tunnel, until it reached point H. At point H, the tunneling would be removed and would go back then to its original IP address. We would have provided confidentiality of everything, including the original IP header. So the difference being now that a person monitoring traffic at point E would only be able to see traffic as going from C to H, from the branch office to the head office. But they would not see who in the branch office was talking to whom in the head office. We would also have, of course, integrity of that new IP header. So we knew that this tunnel truly did originate at C, not at point T pretending to be point C. So we have source and origin integrity, payload integrity, payload confidentiality, and we also have some traffic flow confidentiality as well. Now, both tunnel and transport mode can be used with Authentication Header as well. However, because we're not encrypting anything, there's no value really in setting up a tunnel usually with Authentication Header. So this is how IPSEC or IP Security operates.



▼ Transcript

We've looked at how we provide encryption at various layers of the OSI, or a TCPIP stack. Most of this was end-to-end type of encryption. But now, we're gonna look at how do we encrypt and protect data between two adjacent points across the network. Two adjacent points is called a datalink; a link between router D and router E, or especially for example, a link between a satellite ground point at E and a ground point at F. If this was a satellite link from, say, London to New York, we know that the traffic being transmitted from London could easily be picked up in Philadelphia, Boston, and Washington D.C., as well as many other places. So, what do we do? At the datalink, we wanna protect data between those two adjacent points, and very often, here is where we'll use hardware-based encryption. This is a stream of data that flows between those two points and hardware based encryption is excellent for streaming information. All of the information that hits E will automatically be encrypted before it gets and before it says transmitted until it's received at F, and our packet would look like this: We have our datalink header that shows that traffic is going from point E to point F. Then, everything from within that point is encrypted. For example, the IP header that shows traffic going from A to I. The transport header, the application header, and the message or payload itself. So, a man in the middle attack, picking up communications from this satellite, would only be able to see that traffic is going from London to New York. Nothing else would be visible to the person at that point. Now, this is important that we have the datalink header as still unencrypted, because a ground station in Boston, when it sees traffic that's routed towards New York, will say, oh that's not for me, and would just ignore it, because it would not even have the symmetric key being used to encrypt the data from London to New York. We also have another place where we'll do datalink encryption, and that is between a laptop and a wireless access point. Here's where we use encryption products. We normally used, or initially used, WEP, WPA2 for example today, and what do we do in that case? Our frame would now look like this. Our datalink header is A to B. Our IP header: A to I. Our transport application and the message itself. And a person who is intercepting the traffic, transmitting wirelessly between these two points, would again only be able to see that a laptop was talking to wireless access point, but would not be able to see who that laptop was really trying to talk to. What we'd see then, is the encryption is removed at the wireless access point, which means, if we need to have encrypted communication, we would have to re-encrypt over each of the datalinks, and this is what we often will see. We'll see then, encryption put on, taken off, put on again for the next link, and so on. The point of compromise is still if a person is able to compromise any of the actual nodes themselves, they could actually intercept the traffic at that point, or is momentarily decrypted. A lot of this is based on encrypting very sensitive links where it could be very susceptible to monitoring.

Summary

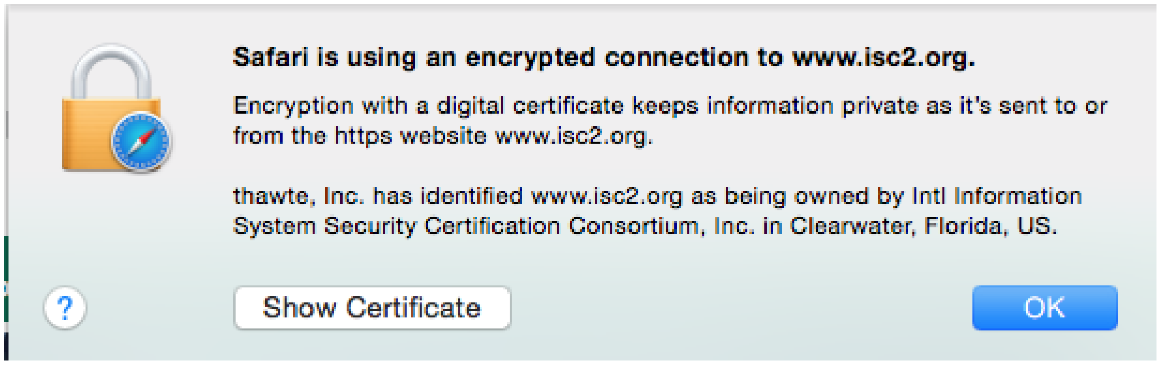
The security practitioner should have an understanding of the concepts, terminology, strengths, and weaknesses of cryptography as just described. While there is much more fascinating knowledge that can be examined in cryptography, the level of knowledge described here is important to ensure that cryptography can be implemented effectively within the organization.

Public Key Infrastructure (PKI) Overview

A major challenge associated with network communications is trust. How can a user know whether or not the website they are visiting is who it says it is? Is a user really on their bank's website or the website of someone pretending to be the bank, hoping that a victim will share their banking details?

An excellent solution to establishing trust is PKI. In the previous module we saw how asymmetric cryptography can be used to authenticate the sender of a message, now we can see how it can also be used to authenticate the recipient of the message.

(ISC)^2 needs to provide a trusted website to its members. For this reason, (ISC)^2 created an asymmetric key pair. As we know, they keep the private key secret, but the public key can be public—everyone can have a copy of it. The question for all of us is whether it is truly the public key of (ISC)^2 or just someone pretending to be (ISC)^2 and sending us their spoofed public key instead. To avoid this situation, (ISC)^2 went to a Certificate Authority (CA) named Thawte and provided them with a copy of (ISC)^2's public key. Thawte was then requested to issue a certificate to (ISC)^2 that proved the public key in the certificate belonged to (ISC)^2. Thawte then validated that the person applying for the certificate was authorized to obtain a certificate on behalf of (ISC)^2 and that (ISC)^2 was a legitimate organization entitled to the certificate.


Safari is using an encrypted connection to www.isc2.org.

Encryption with a digital certificate keeps information private as it's sent to or from the https website www.isc2.org.

thawte, Inc. has identified www.isc2.org as being owned by Intl Information System Security Certification Consortium, Inc. in Clearwater, Florida, US.

Workshop—PKI Certificates

Visit the website of (ISC)^2 and answer the following questions about the certificate that (ISC)^2 has on their website.

1. What standard was used to create this certificate?
2. What is the purpose and uses permitted for the certificate?
3. What is the length of (ISC)^2 public key?
4. What is a CRL and what it is used for?
5. What are the permitted uses for the (ISC)^2 public key—note that this is NOT the same as the uses of the certificate?
6. What is the purpose of the SHA256 value at the end of the certificate?

Registration Authority

In some cases, an applicant for a certificate will not go directly to a Certificate Authority, instead they may apply to a local representative of the CA called a Registration Authority (RA). The RA does not issue a certificate, but it validates the application and passes the application for a certificate to the CA.

Establishing Trust

When a CA issues a certificate, it signs the certificate. This proves that the certificate did indeed come from the CA and that the certificate has not been altered by anyone. As will be seen in the next module, a digital signature validates the source and integrity of a document.

Many CAs operate on a hierarchical trust model. At the top of the hierarchy is a root CA and everyone trusts the root. The lower level CAs are validated by the root and even if one organization uses a certificate issued by one CA, it can trust the certificates from other CAs that are under the same root CA. If the CA used by an organization is not under the same root CA, then there must be cross-certification between the CAs that they agree to accept the certificates issued by another CA.

Another method of establishing trust in certificates is through a "web of trust." A web of trust is based on trust between peers instead of a hierarchy. Alice does not know who Bob is and, therefore, she cannot be certain she has Bob's public key. But Alice does know Sam, and Sam knows Bob. So, Sam verifies for Alice that she truly has Bob's public key. Now Alice knows that any message she encrypts with Bob's public key can only be read by Bob, using his private key. In a web of trust, each person can act as a validator for others.

Integrity

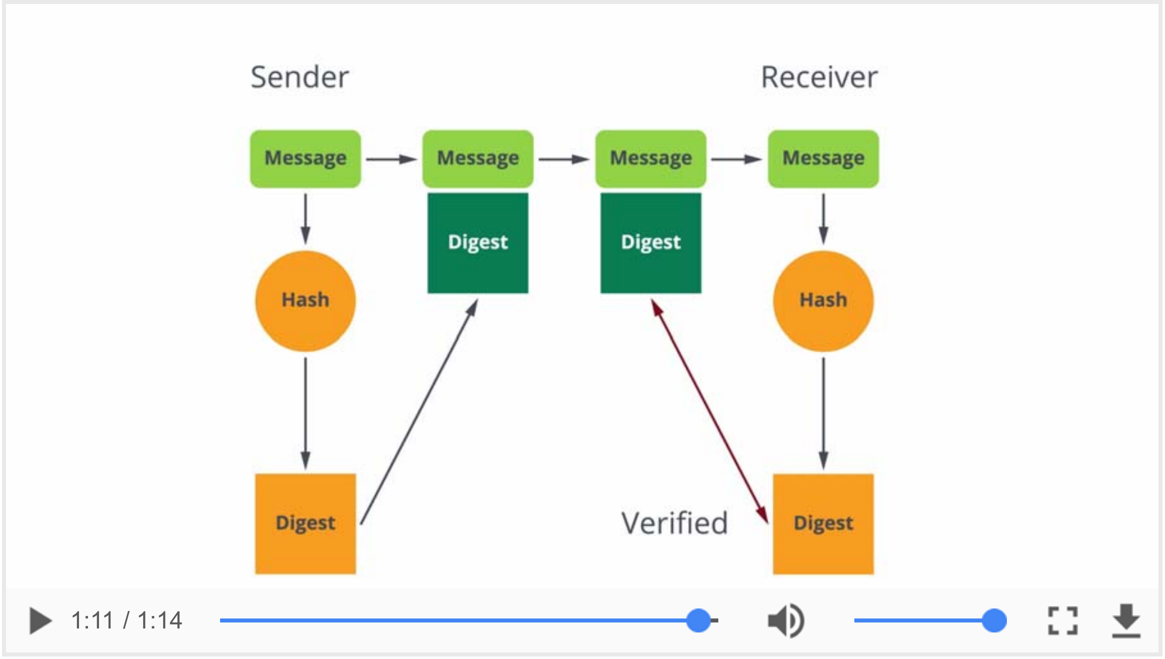
An important part of secure communications is integrity. People sending data across a network need to be sure that the data has not been altered in transit. The alteration of data may be accidental as the result of noise or a loss of network connectivity, or the alteration may be intentional where a hacker is attempting to modify the data and disrupt communications. There have been numerous ways to prove the integrity of communications over the years including:

- Parity bits
- Checksums
- Cyclic Redundancy Checks (CRC)
- Message Authentication Codes (MAC and HMAC)
- Hashing

Hashing is the most common method of ensuring message integrity today using protocols such as:

- MD5
- SHA1, SHA-256, SHA2 (512)
- Keccak (SHA3 draft)

The operation of a hashing algorithm is seen here:



▼ Transcript

Hashing algorithms are used to prove the integrity of a message. They operate by running the entire message through a hashing algorithm. The hash function computes a fixed-length hash of the message. The hash is often called a "digest" or "fingerprint."

The sender then appends the digest to the message. The message and the attached hash can now be sent to the receiver. The receiver now has the message, but to verify that the message was not changed during transmission or during storage, the receiver runs the received message through the same hash function as the sender used.

This generates a digest of the received message. If the digest of the received message is the same as the digest sent by the sender, then the receiver has confidence that the received message is the same as the one sent by the sender.

The problem with a simple hash function like this is that it does not protect against a malicious attacker that would be able to change both the message and the hash/digest.

Digital Signatures

A digital signature is not the same as a digitized signature. A digitized signature is the scan of a person's autograph, a digital signature is a signed hash of a message.

We know that a hash is a good way to ensure integrity of a message and that encrypting a message with a private key provides proof of origin, but to encrypt the entire message with a private key would be far too slow. So, the solution is to encrypt the hash of the message with the private key of the sender of the message. This process will prove the integrity and source of the message—we now know who the message came from and that the message is authentic—it has not been changed or altered. This provides non-repudiation.

It is important to note that a digital signature does not encrypt the message. The use of a digital signature does not provide message confidentiality, according to the Digital Signature Standard (DSS), it only addresses message integrity and proof of origin.

Fundamental Key Management Concepts Overview

The compromise of most cryptographic implementations is not due to weaknesses in the algorithms, instead it is due to problem with key management. This is often a human problem when people share keys, choose weak keys, do not destroy old keys, or store keys insecurely. A major part of breaking the code for the Enigma machine was the work of the Polish mathematician Marian Rejewski. He was able to determine the order of the letters on the cipherdisks used in the Enigma by gaining access to a month's worth of old encryption keys.

Kerckhoff's Principles

The Dutch mathematician, Auguste Kerckhoff, wrote six principles for cryptography in the 1880s. One of the most important principles was that a cryptographic algorithm did not need to be a secret to be secure. The work of Claude Shannon expanded on that principle to reinforce the concept that the strength of a cryptosystem is dependent on the secrecy of the key, and the cryptographic system should be secure even if everything about the system is known except the key. Claude Shannon further wrote that we should always assume that our adversaries have a copy of our ciphertext and know our algorithm. These principles emphasize the need to protect cryptographic keys. Keys are the secret that protects and unlocks our data, and a compromise of the keys will result in the loss or disclosure of our data.

Key Creation

The generation or creation of cryptographic keys should be a secure trusted process. The keys are generated from the key space, which is the entire body of key values available to be chosen. For example, a pin number that is four digits long has a key space of 10,000 possible pin numbers (based on the fact that a four digit pin number can be any value from 0000–9999). If the pin is five digits long the possible number of keys is 100,000. This demonstrates the value of longer keys, as a longer key is more resistant to guessing.

People often fall into the trap of choosing keys from a small part of the key space, perhaps they even just increment the key each time they are forced to change it from Winter16 to Winter17. Such patterns can make the compromise and guessing of the keys much simpler.

Keys, like passwords, should not be based on known values such as birth dates or telephone numbers.

Cryptoperiod

The cryptoperiod is the length of time the key is valid before it has to be changed. Most organization realize that the longer a key is in use, the more likely that it will be compromised. Most organizations will set a maximum time that a key or password is in use before it must be changed.

Key History

Organizations and individuals are often required to keep a copy of old keys so that that can access old encrypted files such as backups. This requires the organization to develop a secure way to store old keys and prevent their compromise or their loss. The loss of an encryption key will almost certainly result in the loss of the data in the encrypted file as well. In some jurisdictions, it is even required by law to retain a copy of old keys in case law enforcement or some other regulatory body requires access to older encrypted files.

Key Distribution

Keys must be sent securely between parties that are communicating. There are many methods of exchanging or negotiating keys ranging from Diffie-Hellman to a personal courier. The challenge is to ensure that keys are not compromised by a man-in-the-middle attack during the distribution of the keys. As seen earlier, we can ensure the validity and ownership of public keys by having them distributed in a certificate.

Key Storage

Keys must be kept in a secure location. There are many examples where a system was compromised because passwords were written down and stored under a keyboard or in the desk calendar. Private keys for banks, for example, are kept in a hardware security module (HSM) that can store the keys in a way that makes them nearly impossible to compromise. Other key storage solutions include USB drives, smartcards, tokens, password wallets, and trusted third parties.

Key Recovery

The loss of a key will usually mean the loss of the encrypted data, so organizations must take steps to ensure that keys will be available when required. There are several methods of key recovery available including multi-party (split knowledge), dual control, and key escrow. Some organizations will require employees to provide a copy of their encryption keys to a trusted department that retains a copy of all keys. This would be known as key escrow, but this does bring into question the ability to prove non-repudiation since there is more than one person that has a copy of the encryption keys. To prevent any one person from knowing or having access to another person's key, the organization may implement the principle of split knowledge where no one person has access to an entire key, and it requires two people working together to assemble the key (dual control).

Summary

The security practitioner should be familiar with the various types of encryption algorithms and their strengths and weaknesses. This allows the effective deployment and operation of encryption in the organization. The protection of keys is central to the secure operation of any cryptographic implementation.