



PDHonline Course G425 (6 PDH)

Introduction to Digital Signatures - Part One

Instructor: Daryl S. Banks, PSM

2020

PDH Online | PDH Center

5272 Meadow Estates Drive
Fairfax, VA 22030-6658
Phone: 703-988-0088
www.PDHonline.com

An Approved Continuing Education Provider

Introduction to Digital Signatures

Part One

TABLE OF CONTENTS

1. Introduction.....	6
Abstract.....	6
Assumptions.....	7
Prerequisite Documents	7
Audience and Document Conventions.....	7
About the Author	8
2. Fundamentals	9
Encryption.....	9
Cipher Keys	9
Symmetric Cipher	10
Asymmetric Cipher.....	10
Distributing Keys	12
Message Digest.....	13
Authentication.....	13
Authentication Systems	13
Message Authentication Code	14
Bit versus Byte.....	15
Checksums	16
MD5 Hash.....	18
SHA-1 Hash.....	19
DSA.....	20
SHA1RSA.....	20
Message Signing.....	21
Message Verification	21

- 3. Algorithm Security..... 23
 - Random Numbers 23
 - Key Complexity 23
 - Validation..... 24
 - Hash Size 25
- 4. Security Methods 26
 - Password Protect..... 27
 - Message Encryption..... 28
 - Digital Signature 29
 - Applications 31
 - One-Way Hash..... 33
- 5. Certificates 34
 - Certificate Chains..... 35
 - Certificate Content 37
 - Distinguished Names 39
 - Signature Algorithm..... 39
 - Thumbprint Algorithm..... 41
 - Enhanced Key Usage 42
 - Vendor Defined Classes..... 42
- 6. Digital Identification..... 42
 - Analogy for Digital ID..... 42
 - What is a Digital ID? 43
 - Why Do I Need One?..... 45
 - How Do I Obtain One? 45
 - Self-Signed Digital IDs..... 45
 - Certificate Authorities..... 45
 - Digital IDs vs. Sealed 45
- 7. Digitally Sign and Seal 46
 - Adobe PDF..... 47
 - Manual Hashing 47
 - Using Digital ID..... 47

CAD File..... 47

 Manual Hashing..... 47

 Using Digital ID..... 47

File Delivery 48

8. Storage and Distribution 48

 Local File Server..... 48

 CAD Middleware Repositories..... 49

 Cloud Solutions..... 51

9. Summary 54

Appendix A..... 55

 Change Encryption Settings..... 56

 Remove Encryption Settings..... 57

 Sharing Certificates with Others..... 57

 Get Certificates from Other Users 57

 Verify Information on a Certificate 59

 Verify Your Own Certificate 60

 Verify information on the Certificate of a Contact..... 60

 Delete a Certificate from Trusted Identities..... 60

 Create a Self-Signed Digital ID 61

 Register a Digital ID..... 62

 Specify the Default Digital ID 62

 Change the Password and Timeout for a Digital ID..... 63

 Delete your Digital ID 64

 Protecting Digital IDs 64

 How to Protect Your Digital IDs 64

 What to do if a Digital ID is Lost or Stolen..... 65

 Smart Cards and Hardware Tokens 65

Appendix B..... 66

 Certificates with AutoCAD..... 66

 Attach a Signature to a Single File 66

 Attach Digital Signatures to Multiple Files 66

Add a Comment and Time Stamp..... 66

Digitally Sign an Encrypted Drawing..... 66

Import a Certificate..... 67

Overview of Drawings with Digital Signatures..... 67

Appendix C..... 69

 Content of a Certificate..... 69

Appendix D..... 71

 Example Hard Copy Output..... 71

Appendix E..... 72

 Florida Digital Signature Standards for Surveying and Mapping 72

Appendix F..... 74

Glossary 74

 AES Algorithm 74

 Algorithm..... 74

 Architectural Layer..... 74

 Byte..... 74

 CAD 75

 CADD..... 75

 Cloud Solutions or Cloud Computing..... 75

 Collision..... 75

 Cryptography 76

 Digital Signature..... 76

 Certificate Authority (CA)..... 76

 Checksum..... 77

 Client Server Architecture 77

 Cyclic Redundancy Check..... 77

 Encryption..... 77

 File Server..... 78

 Firewall 78

 Granularity 78

 Hash Function..... 78

Hash Chain..... 78

Message Digest..... 78

Permissions..... 79

Protocol..... 79

Public-Key Cryptography..... 79

Seed Value..... 80

Salt Value..... 80

Versioning..... 80

Workgroup..... 80

Appendix G..... 81

References..... 83

Introduction to Digital Signatures -Part One-

Daryl S. Banks, PSM

1. INTRODUCTION

Abstract

With a Digital Signature, a Digital ID is attached to a CAD file so users can work together more easily with others on projects. Recipients of drawings are provided with trustworthy information about the individual who created a drawing, and whether it was modified since it was digitally signed.

Digital Signatures provide the following benefits:

- ❖ Recipients of digitally signed drawings can be sure that the organizations or individuals who sent the drawings are trustworthy.
- ❖ A Digital Signature guarantees that a drawing's content and properties have not changed since the drawing was digitally signed.
- ❖ With a third-party Digital ID, a signed file cannot be rejected as invalid. The signer of a file cannot reject the file later by claiming the signature was copied.

A Digital Signature is not a digitized signature (an electronic scan of a signature). While a Digital Signature helps prove the author's identity and a drawing's authenticity, a digitized signature is nothing more than an electronic version of the author's own signature inserted or attached to a CAD drawing. It can be forged and copied and has no tangible security value. Digital Signature is a widely used term that uses a Public Key Infrastructure (PKI) certificate known as a Digital ID. PKI is inseparable from digital certificates. A PKI is responsible for issuing certificates, ensuring the distribution of these certificates through a directory, and validating certificates.

By the completion of this course, the you will be able to electronically sign any CAD file with your own Digital ID, and establish a drawing delivery policy at your company. Furthermore, you will learn how to purchase, install, and sign electronic documents such as Microsoft Word, Excel, Adobe PDF, and most importantly CAD drawing files. Along with the support site aesignature.com, there are additional videos, showing the step-by-step procedures for many of the topics discussed herein.

The basic instructions for signing CAD files are covered in this course. Due to its length, the author chose to exclude certain topics and prepare a second course covering signing Adobe PDF files, since technologies have been developed to automatically turn PDF into CAD vector files. There are important details to discuss regarding signing and sending PDFs securely while maintaining the sender's content, which will involve another six-hour study course.

Assumptions

Digital IDs are the focus of the seminar, but Digital IDs are laden with a few college semesters of topics like Encryption, Cryptography, and basic Computer Science. This course focuses on using Digital IDs and does not delve into the area of how to design and build your own Digital ID. Rather, this course condenses material into a six-hour continuing education course.

The topics herein are presented with typical uses of Digital Signatures being discussed, and various constituent parts are presented along the way to aid in understanding the next topic. Many of you have never used a Digital Signature to sign CAD drawings. If you have never used a Digital Signature or certificates to sign CAD drawings, spend 15 minutes viewing the videos on the support blog at aesignature.com (Banks & Banks Consulting, 2013). Click the tutorial link to understand the concept by seeing it in action. The videos will give you a basic layout of the signature process, and demonstrate the straightforward nature of signing CAD files with a Digital ID.

Prerequisite Documents

Here is a link to the video on what product to purchase at VeriSign (Symantec Now) from one of the providers <http://youtu.be/NnEhOCXmobw>. You can find the link on the tutorial section at aesignature.com. There, you will also find supplemental videos on these and more topics.

This video will show you how to download and install the free 30-day certificate or Digital ID from VeriSign. You may use this Digital ID with your own CAD software. If you purchase a Digital ID, the cost is \$20-30 USD, and expires one year from the date of purchase.

If you want to sign PDF documents with their Digital ID, you will need the *Adobe Acrobat Professional* version. You can download a 30-day trial located at Adobe.com and walk through the examples in Appendix A.

Audience and Document Conventions

During this course, ideas and terminology from the AEC industries' viewpoint are presented as much as possible. From time to time, this may not be feasible, and these departures may not be identified. However, for most of the course, the term "files," for example, indicate drawings or Computer Aided Design (CAD) files. Moreover, there will be no distinction between CAD and CADD being Computer Aided Drafting and Design.

Although a special emphasis was given to CAD files, in practice, these Digital Signature methods may be any files, including but not limited to the following list:

1. ESRI Shape Files, shp
2. Text Files ,txt
3. ASCII Files, asc
4. Word, docx
5. Excel, xls
6. Photos, jpg
7. Images, gif

8. Adobe File, pdf
9. AutoCAD, dwg

Furthermore, if the file can be serialized into a stream of bytes (a computer data construct), then a hash value can be calculated. Knowing the limitations and practical uses of these algorithms will help streamline the drawing distribution process.

The approach toward this material is biased toward the Autodesk® product line; therefore, it is important that you knowingly understand the parity between competing software companies. The standards found in one software package are shared with other software packages by industry standards set by the National Institutes for Standards, the Testing (NIST), the National Security Agency (NSA), and the well-known Internet Engineering Task Force Agency (IETA). They set and promote standards for compliance in design and software relating to applications, and a part of these standards are how public and private keys, certificates, and Digital IDs are handled over the Internet. With that stated, be assured the topics discussed herein are mature and have been used for at least a decade in one form or another. Autodesk® started using certificates after the 2000; 2000i versions of AutoCAD and product design has not changed with certificates use since.

About the Author


Daryl Banks has been a Professional Surveyor and Mapper for 12 years. He is a software developer with eight years AutoCAD.Net experience. He is currently an active licensed Autodesk® software developer and Autodesk Developer Network (ADN) member. He has over 20 years Geospatial experience with Land Development, and over ten years experience in software development. He holds a degree in Nuclear and Radiological Engineering from the University of Florida and a degree in Computer and Information Systems from the University of North Florida.

Over the past three years, Mr. Banks has developed, proprietary applications using Autodesk's Software Development Kits (SDK) RealDWG. Currently, he has two software products developed using the Autodesk API, which are both currently listed in the *Autodesk App Store* and commonly referred to as the *Exchange Store*.

Mr. Banks has worked with the ESRI GIS product line including ArcMap and ArcGIS Enterprise Server (past licensed developer), and has managed large-scale development projects using ESRI product and Autodesk Civil 3D. He has used Autodesk Map 3D and Mapguide (Web Platform) to implement Aerial mapping support on land development projects. He has helped support and integrated Map 3D and GIS databases for a large utility company in the Northwest. He has worked for Fortune 100 Engineering Companies with over 29,000 AEC Professionals, with national and international project management experience. He currently operates a geospatial consulting company specializing in custom geospatial application development and land surveying and mapping services.

2. FUNDAMENTALS

Encryption

 **Note:** *The term encryption is a two-way function (encrypt and decrypt). Stating the term another way, encryption changes information (plaintext) into unreadable, encoded messages (ciphertext). The main reason for using encryption is to hide information and recover it later.*

The term “hash” is a one-way function: only convert’s plaintext into an unrecoverable format like ciphertext or hash algorithms convert any input into ciphertext since algorithms are commonly iterative processes.

Encryption is a tool that protects secrets. Users might encrypt files on their hard drives so that the loss or theft of their computers would not compromise their data. Users also may want to encrypt network communications, especially those to their bank, doctor, or friends. This leads us to the term called a “cipher,” which encrypts or decrypts data. There are three types of ciphers:

Symmetric, or “private key.” Ciphers use a single secret key to encrypt and decrypt data. Symmetric keys can be useful in applications such as hard disk file encryption when the same person encrypts and decrypts data.

Asymmetric, or “public-key” ciphers use a pair of keys. One key is public and may be freely distributed. The other key is private and should be kept secret. Data encrypted with either key can be decrypted using the other key.

Hybrid systems use a combination of symmetric and asymmetric ciphers. In a hybrid system, asymmetric ciphers are much slower than their symmetric counterparts are. Asymmetric ciphers are used to exchange a private key called a “secret key” or a “session key.” The secret key is used with an asymmetric cipher for data encryption and decryption.

Cipher Keys

A key agreement protocol or key exchange protocol is a system where two parties can agree on a secret value. Even if someone is listening to the two parties, they can still agree on a secret value without revealing it. This is useful in situations where the two parties are likely to agree on a key that can be used to encrypt a subsequent conversation. Keys are vital to secrecy and to the algorithms that implement them.

It is easiest to think of keys in a conceptual way. First, conceptualize a cipher as a machine. To run the machine, the user sticks a key in it. Insert plain text on one side and get cipher text out the other side. Then, run the cipher in reverse to convert cipher text to plaintext. In practice, the cipher is a mathematical formula. The key is just a special number, or a few special numbers, that are used in the formula. A public key for a cipher, for example, consists of three numbers called P, G, and Y. When users encrypt data with a typical cipher, the P, G, and Y values are used mathematically to transform the plaintext into cipher text (Knudsen, 1998).

Symmetric Cipher

A Symmetric Cipher uses the same key at the sending and receiving end. Symmetric Ciphers are also called “private key” or “secret key” ciphers. Using a Symmetric Cipher can be difficult. Users must keep the key a secret and they must trust the recipient to keep the key secret. If someone else obtains the key, the user and the recipient must agree on a new key in a secure manner. For example, let’s say the drafts-person and client are using a Symmetric Cipher to exchange drawings and an intruder obtains the client’s copy of the private key. The security mechanism fails, since the intruder can sign-in with the private key. The drafts-person must determine how to get a copy of a newly created private key to the client without letting anyone else find out.

Users can encounter the same problem with the “server” and “client” parts of an application. If a user wants to keep people from snooping on the data that passes between the client and server, they can use a Symmetric Cipher. However, both the client and the server need to know the private key. If the key is discovered, their entire system is quickly insecure. To avoid such problems, users can program each client with a different private key, but the scenario of distributing keys can quickly become a distribution headache. Below is an illustration of a symmetric key cipher.

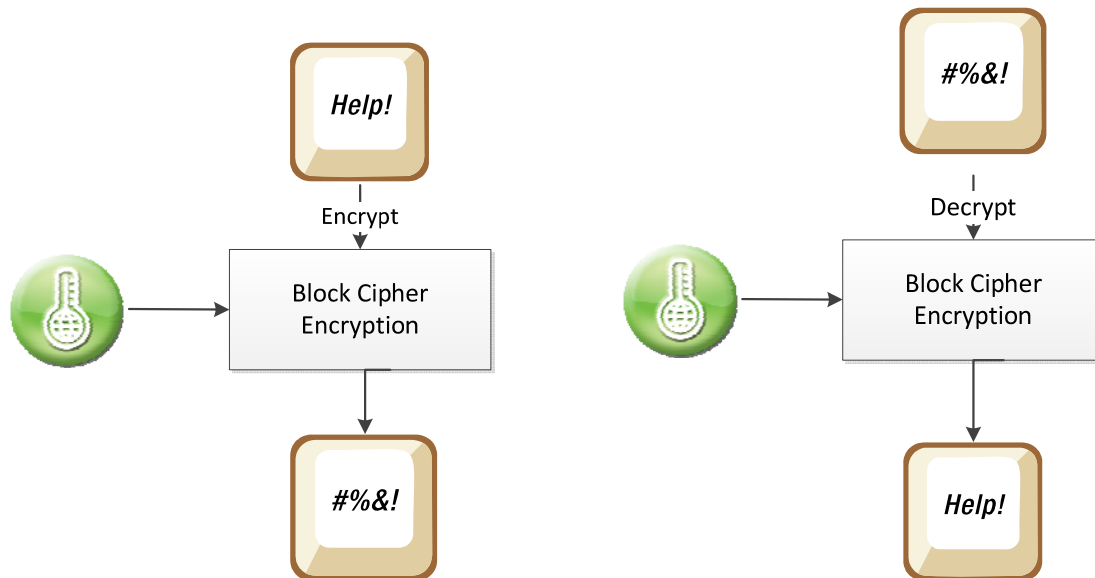


Figure 1 shows symmetric key being used to encrypt and decrypt text.


Asymmetric Cipher

⚠ Note: Message Encryption and Digital Signatures may be used with a public key certificate called a Digital ID or a Public Key Pair. It is important to understand that a custom implementation of either the Digital ID or Public Key Pairs may change which key actually signs the files. Furthermore, the focus in this course is with an implementation similar to AutoCAD. It uses the private key with the Digital ID to sign the file, while the public key is used to verify the

file. This is why the signer must distribute the public key to the recipient. Another example that illustrates the same method in signing a file is “Adobe Acrobat Digitally Signed” files. When signing using the Adobe Software, the private key signs the file, and the recipient verifies using the public key distributed by the sender. Consequently, the public key is used by the recipient (See Figures 8 and 9).

Message Encryption with Digital ID – *a sender can access the recipient's public key, which allows the sender to encrypt the message, knowing that only the recipient can decrypt the message. This time, it is the recipient's digital certificate that makes the encryption possible. As with digital signatures, the public key from the digital certificate makes the operation possible (See Figure 7).*

When you purchase a Digital ID certificate from a third party provider, you are issued a pair of keys, one private and one public. Data encrypted with your public key can only be decrypted with your private key (by you). Data encrypted by your private key works the opposite way: anyone can obtain your public key and decrypt it, and they can be sure that you were the sender since only your private key will encode information that your public key will decrypt.

 **Note:** *In this illustration with Asymmetric Cipher below, the public key cryptography does not use certificates (with PKI). Consequently, the recipient's public key encrypts, or signs the file. Likewise, the private key decrypts the file (See Figure 2).*

The shortcomings of Symmetric Ciphers are addressed by Asymmetric Ciphers, also called public key ciphers; these ciphers actually involve a public key that can be freely distributed and a private key that is secret. These keys always generate in matching pairs. Public keys are public; users can publish them in a newspaper or email them. No one can violate the user's privacy or impersonate them without their private key. The mechanism for distributing public keys, however, is a big challenge.

The example below does not implement the certificate PKI; it uses the methodology of matching key pairs. Data encrypted using a public key can be decrypted using the recipient's private key. The public key will encrypt the data, and the recipient's private key will decrypt only the data that was encrypted using the matching public key. In a few cases, the reverse process also works; data encrypted with the private key can be decrypted with the public key, as in Digital IDs. If a drafts-person wants to send a drawing to the client, he or she can encrypt a drawing using the client's public key. Only the matching private key, that should be known only to the client, can be used to decrypt the drawing.

If the intruder can intercept the drawing, it does not do the intruder any good because the drawing can be decrypted only with the client's private key. In addition, as long as the client keeps the private key secret, he or she can give the public key to anyone who wants it, even the intruder. With the public key, the intruder can send the client drawings, if he or she chooses, but cannot decode anything that others send. In particular, if the intruder uses the public key to attempt to replicate the client's private key, the process is currently mathematically infeasible to the advanced hackers, since access to million dollar computer systems to compute the values are difficult to obtain.

Asymmetric ciphers are much slower than symmetric ciphers, so they are not used to encrypt long messages or entire CAD files; they encrypt the message digest or “hash,” the product of a one-way hash transformation such as SHA-1 algorithm. As seen below in Figure 2, the sender uses the receiver’s public key to encrypt and private key to decrypt the data.

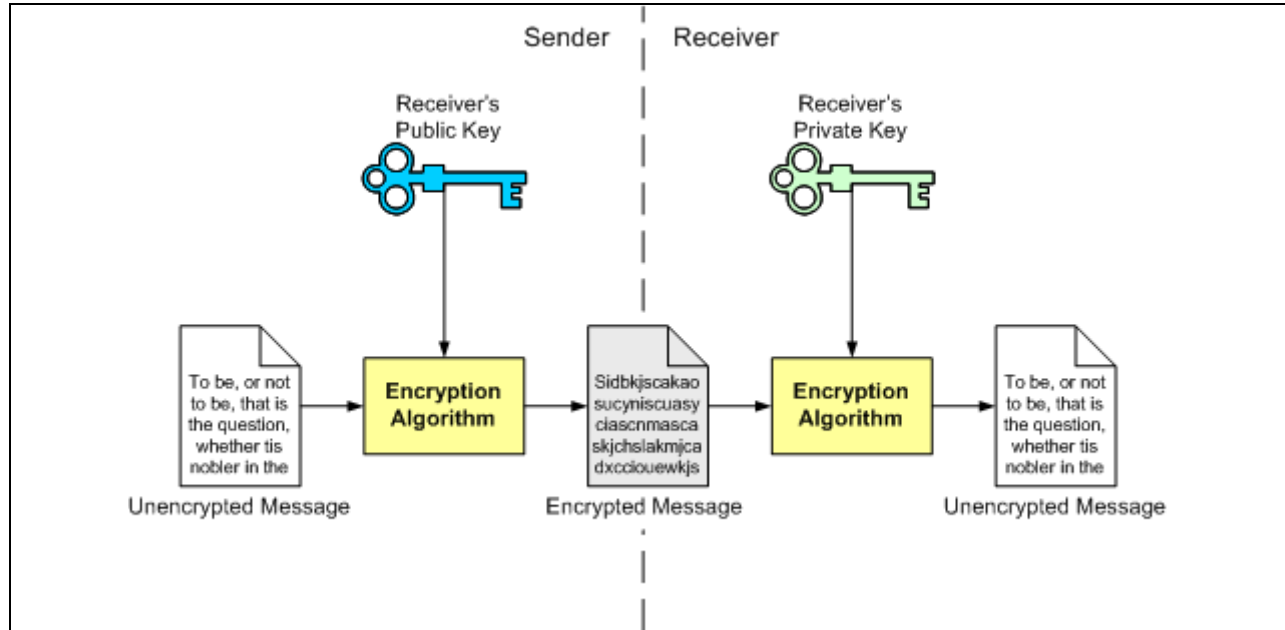


Figure 2 shows the asymmetric cipher design without certificates. (Sotomayor, 2004)

Distributing Keys

How can a drafts-person obtain the client’s public key? This can happen in several different ways. The client could post the key on a network server for the drafts-person to pick up then email it to that person. Because public keys are meant to be distributed, the client does not care if the intruder intercepts this communication. He does care, however, if the intruder gives the drafts-person a bogus key instead of the client’s genuine public key. If the intruder is successful with this trick, he or she can impersonate the client, call on him/her, and then the drafts-person is in serious trouble. However, there is a solution to this problem called “certificates.”

One of the benefits of public key cryptography is that it reduces key management because one key pair takes the place of numerous symmetric keys. This benefit is further enhanced by digital certificates, which allow public keys to be distributed and managed. Since Digital Signatures use a Public Key Infrastructure, a PKI is inseparable from digital certificates and responsible for issuing certificates, ensuring the distribution of these certificates through a directory, and validating certificates. As shown in Figure 18, the PKI structure distributes and validates the certificates that service the Digital Signature interface. As a result, the Digital Signature using certificates is dependent on the PKI for creation, distribution, and partial verification of root certificates, unless the user is implementing a custom Public Key Pair.

Message Digest

A message digest can be used to verify data integrity. A message digest is a special number calculated from a set of input data like a CAD drawing from a hash algorithm such as SHA-1 or MD5. Similarly, the depiction of text in the form of a single string of digits, created using a procedure is called a one-way hash function. The message digest, or hash values have unique output for a given input. For example, two drawings will have different message digests if they are different files. However, if the two drawings are exactly equal, then their respective message digests will be the same.

Authentication

At various stages in transactions, users want to be sure the people they contract with are really who they say they are. This process of proving identity is called “authentication.”

When users call a client on the telephone, they identify themselves by saying their name. The sound of their voice authenticates them to the person on the other end of the line. When users use an automated bank machine, their bankcard identifies them and their secret code authenticates them. Someone else using another person’s bankcard would presumably not know the code and thus cannot pretend to be the owner.

Most computer systems use a user identification and password combination for identity and authentication. Users could identify themselves using a *username* and authenticate their identities with a password. Asymmetric ciphers can replace traditional passwords and be used for authentication. Suppose a drafts-person encrypts the client’s CAD file using a private key. When the client downloads the drafts-person’s CAD file, if he or she decrypts it using the public key, the client can be sure the file is from the drafts-person because only the drafts-person’s private key could have encrypted the file in the first place.

Asymmetric ciphers are computationally expensive, meaning they are slow. Unfortunately, it is not practical to use an asymmetric cipher for entire CAD files, since the files typically range in megabytes (MB). Usually, asymmetric ciphers are used to authenticate particular participants of an exchange. The exchange itself is encrypted with asymmetric cipher using a special one-time key called a “session key.” The challenge is exchanging the session key without having anyone else find out about it. The Secure Sockets Layer (SSL) does exactly this; however, this is not within the scope of this course.

Authentication Systems

There are a few cryptographic ciphers useful for authentication. They are **message digest (hash)**, **digital signatures (custom key pair)**, and **certificates (PKI key pair)**. Message digests produce a small thumbprint of a larger set of data. Digital Signatures can be used to prove the integrity of the data. In addition, certificates are cryptographically secure containers for public keys.

Important features of designing secure websites are the “login forms.” With custom enterprise applications, login forms force users to authenticate themselves to an application before they use it. Digital signatures are used instead of passwords with enterprise-level applications within large

corporations. This is how users seamlessly access their internal email, FTP files, and chat with colleagues through internal chat programs without a password at each interface.

The challenge of building a secure application is implementing a trustworthy authentication system. There may be a small amount of overlap with previous examples, but here are a few real world examples of authentication:

- ❖ An automated bank machine – the user is identified by using a bankcard. The user is authenticated by using a personal identification number. The personal identification number is a shared secret, something that both the user and the bank possess. Presumably, the user and the bank are the only ones who know this number.
- ❖ When clients use a credit card, they identify themselves with the card. They authenticate themselves with a signature. Most store clerks never check the signature in the situation. Possession of the card is authentication enough. This is true when users order something of the telephone as well simply knowing the credit card number is proof of their identity.
- ❖ When users rent a movie from Redbox or at a store, they prove their identity with the vendor or by entering their credit card ZIP Code.
- ❖ Authentication is tremendously important in computer applications. The program or person the user communicates with may be in the next room or in the next office building. The users have none of the usual visual clues that are helpful in everyday transactions. An alternative to physical meetings would be to use Public Key Cryptography (PKC), which offers powerful tools for proving identity.

Message Authentication Code

In cryptography, a Message Authentication Code (MAC) is a short piece of information used to authenticate a drawing and to provide integrity and authenticity assurances on the drawing. Integrity assurances detect accidental and intentional drawing changes, while authenticity assurances affirm the drawing's origin.

A MAC algorithm, sometimes called a “keyed (cryptographic) hash function” (however, “cryptographic hash function” is only one of the possible ways to generate MAC values), accepts as input a secret key (symmetric key) and an arbitrary-length message or file to be authenticated, and outputs a MAC. The MAC value protects both a drawing's data integrity as well as its authenticity by allowing verifiers (who also possess the secret key) to detect any changes to the drawing's content.

An example use of a MAC is a security code that is typed in by the user of an ATM Kiosk at a Bank to access accounts. This code is attached to the message or request sent by the user. Message authentication codes attached to the message must be recognized by the receiving system in order to grant the user access. MAC keys are commonly used in electronic fund transfers to maintain information integrity.

Unlike a message digest, a MAC uses a symmetric key to create the digest value. This makes it useful for protecting the integrity of the data that is sent over an insecure network.

In Figure 3 below, the file is hashed with a secret key (known to only the sender and recipient) to produce the MAC. The file along with the MAC is transferred over the Internet to the recipient's

computer where the MAC is separated from the file. The file is computed through the same algorithm with the exact secret key to produce the second MAC value. Finally, both MAC values are compared. If there are no changes, the message is authenticated.

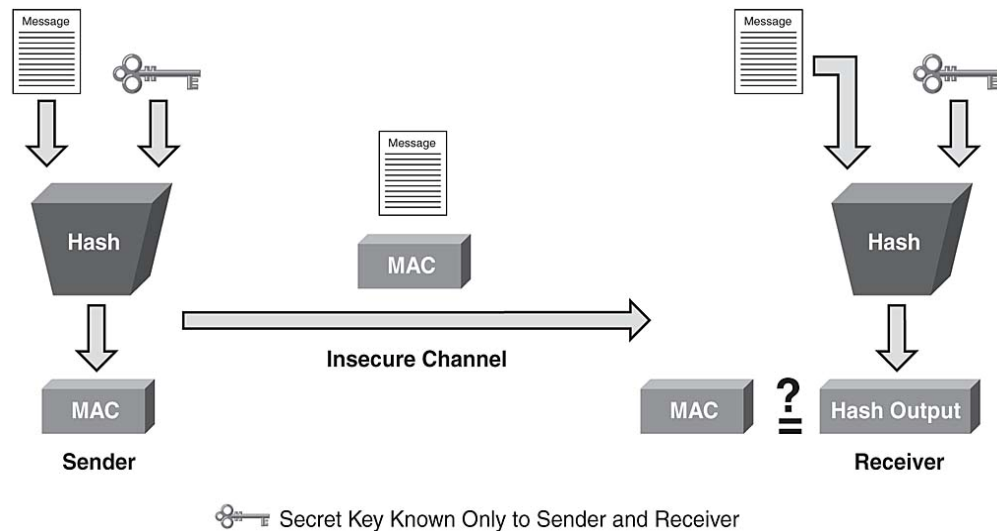


Figure 3 shows a MAC Algorithm design. (Frahim, 2010)

Bit versus Byte

Here is a quick glance of practical theory with the following terminology. A Byte is made up of 8 bits. For example, this value *9e107d9d372bb6826bd81d3542a419d6* is how many bits long? It is 32 bytes, which equals 256 bits. This will help users with the bit length information on hashes. If users would like to know more about bits and bytes, search the Glossary herein.

Algorithm Types

In mathematics and computer science, an Algorithm is a step-by-step procedure for calculations. Algorithms are mathematical mechanisms to produce encrypted data; algorithms also decrypt the encrypted data. With the cryptographic hash function such as the “Secure Hash Algorithm” (SHA-1), a hash code is produced. These hashes are unique output values in that even small changes in the source input (a change in the CAD file) drastically change the resulting output hash by the “avalanche effect.” (See “Cryptographic Hash Function” for the four main properties for review, or the “One-Way Hash”) Hashes alone can verify if a drawing’s content has changed. A term for this computation is called “hash sum,” or “checksum” values. A checksum is a fixed-size datum computed from an arbitrary block of digital data for the purpose of detecting accidental errors that may have been introduced during its transmission or storage. In reading material, the terms checksum and hash are used synonymously to refer to methods of algorithms such as SHA-1, MD5, and SHA-2. Checksums have an error detection aspect by definition like (CRC, and Adler-32). Hashes are for file integrity and not for checking files for transmission errors.

There are several checksum calculators to download and compute the MD5, SHA-1, and SHA-2 hashes of any closed and read-capable drawing. Therefore, if a drawing is opened on a computer,

then the checksum calculator cannot compute the value, since it does not have read-access to the entire file. When selecting a hash algorithm, consider these factors listed below:

- ❖ Output Key Size and Key Requirement
- ❖ The File Type or Message Complexity
- ❖ Application Performance (intended use)
- ❖ Algorithm Strength (MD4, MD5, SHA-1, DSA)

Key size affects the security of the signature and cipher. In general, the longer the key the harder will be for an attacker to decrypt the cipher text or forge a signature. Basically, longer keys have more possible values. If an attacker is trying every possible key to find the right one, a longer key gives the attacker more work. However, key size is only part of the story. Without a complex Algorithm, the cipher is destined for failure by hacker exploits.

Application Developers strive for the most secure systems. However, when security impacts performance to a degree that turns away customers, a redesign of an application may take into account the Algorithm Strength. For example, if an application backend server (hidden behind a firewall) handles all the encryption, which is transparent to the end user and is behind a company's firewall, risk versus performance may dictate implementing a less secure one-way hash like MD5, versus a larger key SHA-2 family of Algorithms, as MD5 has a slight advantage in performance.

The topic of message complexity and performance are not pertinent to this course, but understand if the source input is large and the text files being hashed (compared to small numbers such as credit card numbers and network and application speed) are important, then the hash or checksum methods utilized can have an undesirable impact on the end user.

Checksums

As shown in Figure 4 below, a checksum or hash-sum is a fixed-size datum computed from an arbitrary block of digital data for the purpose of detecting accidental errors that may have been introduced during its transmission or storage. The integrity of the data can be checked at any time by computing the checksum and comparing it with the stored one. If the checksums match, the data was likely not accidentally altered.

1. **CryptoPPInteropSign.zip**
 - MD5: 2647DE3E5E06A07F8CD05F911D75DC3B
 - SHA-1: 74DBED5386D64C9041EE66CFCF884C79F8961B0C
2. **JavaInteropSign.zip**
 - MD5: BA74E602379395177681172EFC73591E
 - SHA-1: B38D7755F6D6D9D4C6ADBE698EF77B771236AB4C
3. **CSInteropSign.zip**
 - MD5: 20A78F6E7817F523923CE2E0B21E95E9
 - SHA-1: 2D69E88935B549993D974C8A41D0091BE3547C5A

**Figure 4 shows the calculated checksum,
or hash algorithm (MD5, SHA-1) values on three different files.**

The procedure that yields the checksum from the data is called a “checksum function” or “checksum algorithm.” A good checksum algorithm will yield a different result with high probability when the data is accidentally corrupted. If the checksums match, the data has the same high probability of being free of accidental errors.

Checksum functions are related to hash functions, fingerprints, randomization functions, and cryptographic hash functions. However, each of those concepts has different applications and therefore different design goals. It is important to not use a checksum like parity bit and positional dependent checksums described below in a security related application, as these types of checksums do not have the properties required to protect data from intentional tampering. Furthermore, an emailed drawing may be corrupted or changed before reaching the client. The client now has a different drawing than was sent. The client must ask for the checksum as shown in Figure 4, or the sender must provide a paper trail for the client to compare.

“Check digits” and “parity bits” are special cases of checksums, appropriate for small blocks of data such as, Social Security numbers, bank account numbers, computer words, single bytes, etc. Various error-correcting codes are based on special checksums that not only detect common errors but also allow the original data to be recovered in certain cases.

Parity Bit

In this case, calculate the hash of a drawing before sending it to the client, so upon a returned drawing, the drawing’s calculated hash may be compared to the original hash. If they are the same, then the file has not been altered. Unlike hash algorithms, checksum calculators work on only a portion of the message, at the bit level. An example is a parity bit checksum, which is described below:

There are two variants of parity bits: even parity bit and odd parity bit. When using even parity, the parity bit is set to 1 if the number of ones in a given set of bits (not including the parity bit) is odd, making the number of ones in the entire set of bits (including the parity bit) even. If the number of ones in a given set of bits is already even, it is set to a 0. When using odd parity, the parity bit is set to 1 if the number of ones in a given set of bits (not including the parity bit) is even, keeping the number of ones in the entire set of bits (including the parity bit) odd. And when the number of set bits is already odd, the odd parity bit is set to 0. In other words, an even parity bit will be set to "1" if the number of 1s + 1 is even, and an odd parity bit will be set to "1" if the number of 1's + 1 is odd. (Cray, 2013)

7 bits of data	8 bits including parity	8 bits including parity
(count of 1 bits)	Even	Odd
0000000 (0)	00000000 (0)	10000000 (1)
1010001 (3)	11010001 (4)	01010001 (3)
1101001 (4)	01101001 (4)	11101001 (5)
1111111 (7)	11111111 (8)	01111111 (7)

Table 1 shows the example of a Parity Bit Checksum.

Position Dependent

The simple checksums described in the table above fail to detect common errors that affect many bits at once, such as changing the order of data words, or inserting or deleting words with all bits set to zero. The checksum algorithms that are most used in practice, such as Fletcher's checksum, Adler-32, and Cyclic Redundancy Checks (CRCs), address these weaknesses by considering not only the value of each word but also its position in the sequence. This feature generally increases the cost of computing the checksum.

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
PTDC0001.JPG	564,347	533,444	JPEG image	1/8/2011 1...	2C04ADED
PTDC0001.pdf	555,050	535,243	Adobe Acroba...	1/7/2013 1...	7E6A32BA
PTDC0002.JPG	625,675	597,616	JPEG image	1/8/2011 1...	4000853A
PTDC0002.pdf	616,567	599,365	Adobe Acroba...	1/7/2013 1...	515E3065
PTDC0003.JPG	635,678	599,769	JPEG image	1/8/2011 1...	81DC51AF
PTDC0003.pdf	618,283	601,407	Adobe Acroba...	1/7/2013 1...	BF34D803
PTDC0004.JPG	579,202	546,787	JPEG image	1/8/2011 1...	52915A04

Figure 5 shows the error checking checksums (circled) of typical group of files using WinZip.

MD5 Hash

The MD5 message digest algorithm was developed by Ronald Rivest in 1991. It is an updated version of MD4 earlier algorithm. It produces a 128-bit message digest value. MD5 was easily found to have specific weaknesses in its collision resistance, this normally prevents an attacker from finding two messages with the same digest. For new applications, use the SHA-1. The 128-bit (16-byte) MD5 hashes are typically represented as a sequence of 32 hexadecimal digits. The following demonstrates a 43-byte ASCII input and the corresponding MD5 hash:

MD5("The quick brown fox jumps over the lazy dog")

= 9e107d9d372bb6826bd81d3542a419d6

Even a small change in the message will, with high probability, result in a different hash, due to the avalanche effect. For example, adding a period to the end of the sentence:

MD5("The quick brown fox jumps over the lazy dog.")

= e4d909c290d0fb1ca068ffaddf22cbd0

The hash of the zero-length string is:

MD5("")

= d41d8cd98f00b204e9800998ecf8427e

The MD5 algorithm is specified for messages consisting of any number of bits; it is not limited to multiples of eight bit (octets known as bytes) as shown in the examples above. MD5 software applications such as “md5sum” might be limited to groupings of bytes, or they might not support streaming for messages of an initially undetermined length. (MD5 Hash, 2013)

SHA-1 Hash

SHA-1 stands for “Secure Hash Algorithm.” It was developed by the National Institute for Standards and Technology (NIST) in conjunction with the National Security Agency (NSA). Like MD5, SHA-1 is based on MD4. The changes made in SHA-1, however, are considerably different from changes made in MD5. Furthermore, SHA-1 produces a message digest with a value that is 160 bits long, which increases resistance to attack. There was an SHA-0, which is now obsolete. SHA and SHA-1 are now used to mean the same thing.

SHA-1 is a cryptographic hash function. The four SHA family of algorithms are structured differently and are distinguished as *SHA-0*, *SHA-1*, *SHA-2*, and *SHA-3*. SHA-1 is very similar to SHA-0, but corrects an error in the original SHA hash specification that led to significant weaknesses. The SHA-0 algorithm was not adopted by many applications; however, the SHA-2 algorithm on the other hand significantly differs from the SHA-1 hash function.

SHA-1 is the most widely used of the existing SHA hash functions, and is employed in several widely used applications and protocols. In 2005, cryptanalysts found attacks on SHA-1 suggesting that the algorithm might not be secure enough for ongoing use. NIST required many applications in federal agencies to move to SHA-2 after 2010 because of the weakness (MD5 and SHA-1) (Blackberry Developer Site). Although no successful attacks have yet been reported on SHA-2, they are algorithmically similar to SHA-1. Therefore, in 2012, NIST standardized an additional algorithm, as SHA-3, based on the results of a long-running public competition (SHA-1, 2013).

These are examples of SHA-1 digests. ASCII encoding is used for all messages.

SHA1("The quick brown fox jumps over the lazy dog")

= 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12

Even a small change in the message will, with overwhelming probability, result in a completely different hash due to the avalanche effect. For example, changing dog to cog produces a hash with different values for 81 of the 160 bits:

```
SHA1("The quick brown fox jumps over the lazy cog")
```

```
= de9f2c7f d25e1b3a fad3e85a 0bd17d9b 100db4b3
```

The hash of the zero-length string is:

```
SHA1("")
```

```
= da39a3ee5e6b4b0d3255bfef95601890afd80709
```

⚠ **Note:** Practice Exercise Cut and paste these values into the web link <https://defuse.ca/checksums.htm> to calculate and verify the above checksums.

DSA

A Digital Signature Algorithm (DSA) is a United States Federal Government standard or Federal Information Processing Standard (FIPS) for Digital Signatures. It was proposed by NIST in August 1991 for use in their Digital Signature Standard (DSS), specified in FIPS 186 (Digital Signature Standard (DSS), 1994).

Adopted in 1993, a minor revision was issued in 1996 as FIPS 186-1. It was developed by the NSA and released as a standard by NIST. It is actually a combination of DSA and SHA-1. The user can use any key size from 512 and 1024 bits, and 64-bit increments. The signature sizes spins on the input key size. Unlike MD5 and SHA-1 hash algorithms, the DSA uses an input key value known as a *seed value* (see Glossary for more information).

SHA1RSA

As shown in Appendix C, the contents of an actual certificate for a Digital ID AEC Signature Banks & Banks Consulting (2013) has a 1024-bit public key. Note – larger key sizes issued like 2048 bits are standard practice when issuing Digital ID's from CA's. In the first two attributes on the certificate, shows the two algorithms participating in this Digital ID. The first algorithm is the SHA1RSA algorithm is a hybrid hash algorithm, which uses a keyless hash SHA-1 to generate the message digest. The second part of the algorithm uses the resulting message digest as input into the RSA algorithm along with the private key as part of the Digital ID not shown. The first two attributes are "Signature Algorithm" and "Signature Hash Algorithm." As shown, the SHA-1 Hash Algorithm calculates a hash length of 160 bits or 20 bytes. Furthermore, the SHA-1 requires no input key to calculate the hash value. This initial part is called the Signature Hash Algorithm and outputs 160-bit hash, or message digest to the next phase.

The next phase is the RSA part of the algorithm. The algorithm uses the resulting 160-bit hash along with the private key as input into the RSA algorithm to sign or attach the Digital ID to the CAD file. At this point, the CAD file is signed or has a Digital ID attached. Any changes to the CAD file will invalidate the attached Digital ID and the CAD software will detach the signature completely if the drawing is saved.

Message Signing


To sign a CAD document of arbitrary size using a Digital ID, perform the following two steps:

1. Hash the CAD document resulting in 160-bit digest.
2. Encrypt the hash of the document as if it were an instance of ciphertext using the private key (See Figure 6).

In general, as users of Digital IDs, think in terms of Digital ID having a private key that is protected with a password we chose when creating the certificate (at VeriSign). This Digital ID has the user's email and name attached. It makes perfect sense to expect if anyone steals the Digital ID, he or she would need to either crack (break) the password or be given the password. As a result, in security terms, a Digital ID is safe as long as the password has not been compromised.

If you want to have a secure Digital ID and sign a CAD file, you need a CAD software package that supports Digital IDs. Follow these steps to sign a file.

1. Once a Digital ID is installed on the user's computer system, read and follow the instructions from the CAD provider to invoke the command to attach the Signature to the file, or read the Appendix B for specifics on AutoCAD.
2. Provide any recipients of the CAD file the public key certificate, which is an exported certificate without the password and private key attached. (See exporting certificates in the users Windows Internet Explorer help, or the PDF and AutoCAD example in the Appendix.)

 *There are videos to show how to install and export Digital ID's and public keys for email clients on the support site.*

Message Verification

To verify a CAD document of arbitrary size using a Digital ID, perform the following three steps:

1. Hash the document.
2. Decrypt the previously generated document hash (from step 2 of Message Signing process) using the signer's public key.
3. Verify the recovered hash from step one of the Message Verification process matching the calculated hash from step two of the Message Verification process.

In the summary of the above, you are comparing your calculated hash of the document with the signer's calculated hash of the document after you remove the signer's encryption operation, as shown in Figure 6.

Verification involves taking the Signed CAD file, separating the Digital Signature from the file and comparing the results of the two (the CAD file and Digital Signature), expecting no change if the signature is valid.

The CAD file, with the Digital Signature, is processed through the SHA-1 hash to obtain the first hash value. Then, the Digital ID uses the RSA algorithm and is processed with the public key certificate that a user provided the client to decipher the original hash value for the message digest. These two values are compared. If they are equal, then we know that there were no changes from the signer's computer to this computer.

As a result, this example illustrates why the mechanism fails when users do not distribute the public key to the users of the signed CAD file.

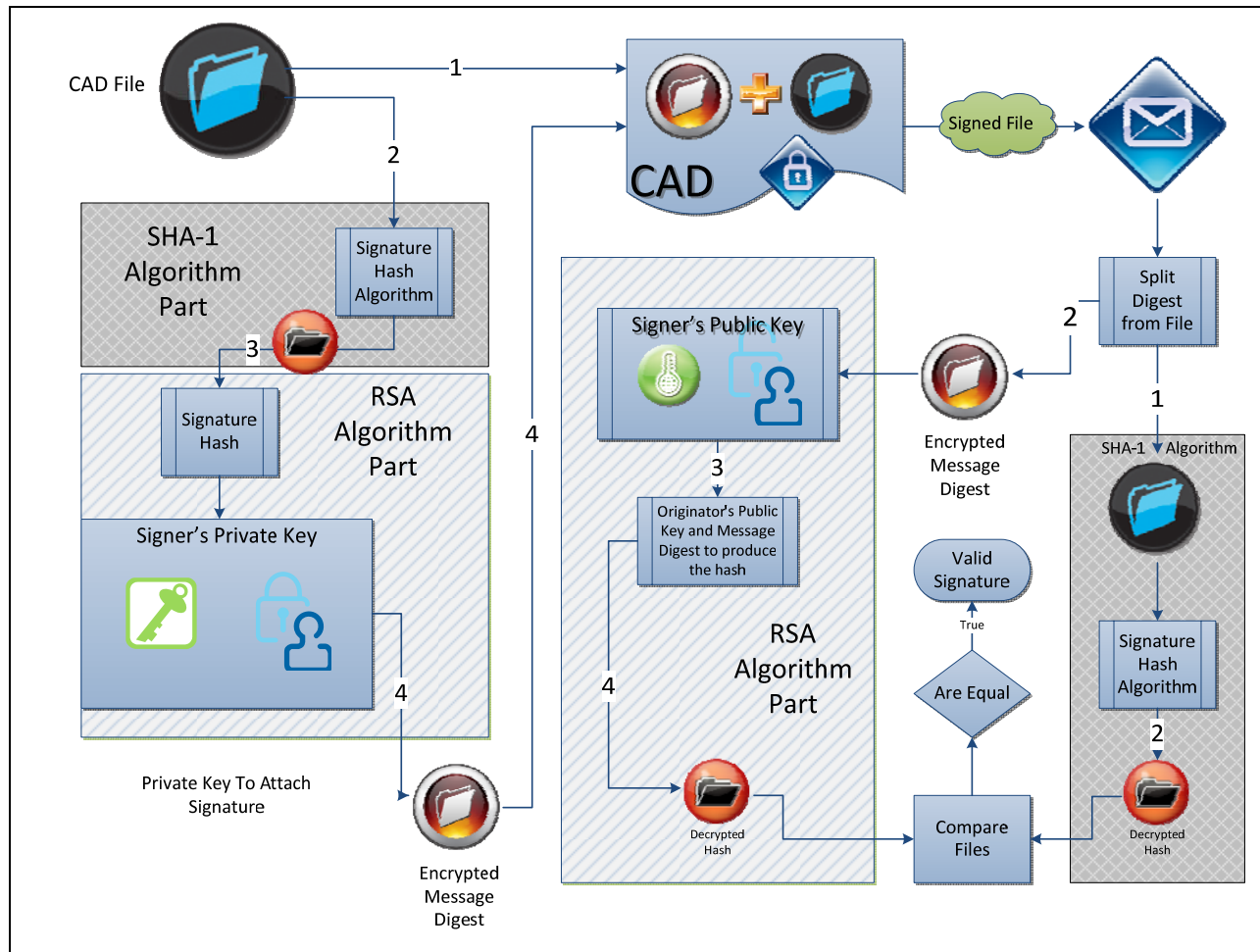


Figure 6 shows a Digital Signature being attached to a drawing and verified by the client.

3. ALGORITHM SECURITY

The material presented up to this point does not discuss the construction of the Secure Hash Algorithm SHA-1 or SHA-2. This section reveals additional details about these hash functions, and you will learn how they are selected and how they differ from other algorithms pertaining to the following list of topics:

- ❖ Randomness
- ❖ Complexity
- ❖ Validation and Testing
- ❖ Hash size

The SHA-2 hash function is implemented in certain widely used security applications and protocols, including Secure Sockets Layer (SSL), the email client Pretty Good Privacy (PGP) and Secure Sockets Hash (SSH).

The SHA-1, SHA-2, and SHA family of algorithms are the secure hash algorithms required by law for use in certain U.S. Government applications, including use within other cryptographic algorithms and protocols, for the protection of sensitive unclassified information. Also, it may be a requirement for signing drawings with a Digital ID in each user's state, as it is in Florida. SHA-1 is being retired for most government uses. These government agencies must use the SHA-2 family of hash functions for these applications after 2010 (Blackberry Developer Site, n.d.). This section compares these two algorithms, how they are validated, and the hash size.

Random Numbers

Random numbers are important for cryptography. Computers are not very good at producing truly random data. Instead, they rely on a pseudorandom number generator. A cryptographically strong pseudorandom number generator, seeded with truly random values, is defined as a random pseudorandom number generator. It does a good job of sending out unpredictable data, but if the pseudorandom number generator is not cryptographically strong or if the seed data (a key used to initialize a cryptographic device so it can accept operational keys using benign transfer techniques) is not random, the security of the application can be compromised.

If a random seed value (an initializing value) is not supplied, the algorithm may use the value of the system clock. This is a predictable seed. For example, create a random number in order to generate a cryptographic key. If an attacker knows the time the created random number was generated, or even approximately, he can guess the likely values of the random number seed. With a relatively small amount of guessing, the attacker can guess which random number seed was used. From this information, an attacker can generate the same supposedly random cryptographic key used and generated. Now the attacker can impersonate the user.

Key Complexity

A key called "mypass" has how many possible values? Using the ASCII system, there are 26 letters in the English alphabet and 10 digits being 0-9 accordingly. As a result, the possible values of a single digit are 26 + 10 per character for a total of 36 for each digit. This computes for the value of "mypass" being a length of five digits $36^5 = 60,466,176$ possible values. This

example excludes capitalized values, special characters, other character sets, or other country alphabets.

At first glance, the 60 million possible values may appear impossible to break; however, below is an illustration how many hackers crack passwords by brute force using modern PCs.

A typical PC runs about 2.0 Gigahertz (GHz), which is 10^9 Hz. Let's say the PC can perform one instruction per second. How long will it take to calculate through all possible combinations of 60 million values or instructions?

$$\frac{60 \times 10^6 \text{ instructions}}{2.0 \times 10^9 \frac{\text{cycles}}{\text{sec}} \times \frac{\text{instructions}}{1 \text{ cycle}}} = 0.03 \text{ sec or } 30 \text{ milliseconds}$$

Consequently, a modern computer can compute through 60 million values in less than one tenth of a second. Adding the complexity of capitalized letters computationally yields result 916×10^6 possible values, this computes to 458 milliseconds of computation time. Increase the password length to eight digits, and the computational time increases to 1.26 days. Finally, remove the capital letters and increase the digits to 20 as in the 160-bit as in SHA-1 output and the result is 2.1×10^{14} years.

These are idealistic examples, with no latency considerations for memory, hard drive, and bus speed. The concept of the power harnessed with modern PCs shown with this ballpark example as it pertains to security and key size is amazing. As shown in the last example, the 10^{14} years approaches the term computationally infeasible, but not impossible to duplicate.

Asymmetric ciphers and signatures have a variable key size. The user's software application design will determine the key length or have the user choose an appropriate key length. Although longer keys are more secure, they are slower. Picking the right key size is a trade-off between finding a comfortable level of security and having the application run too slowly.

Symmetric ciphers can either have a fixed or variable key length depending on the algorithm. Choosing an algorithm is a complex process. Users need to choose something that is secure enough for their application, while at the same time taking into account licensing issues, patent restrictions, and countries' import and export laws. Except for DSA, all the algorithms are free of license and patent restrictions. With the exception of the ciphering key exchange algorithms, they can also be freely exported from the United States.

Validation

For a hash function for which L is the number of bits in the message digest, finding a message that corresponds to a given message digest can always be done using a brute force search in 2^L evaluations. This is called a "Preimage Attack" (Rogaway & Shrimpton, 2009) and may or may not be practical depending on L and the particular computing environment. The second criterion, finding two different messages that produce the same message digest, known as a collision, requires on average of only $2^{L/2}$ evaluations using a "Birthday Attack" (Mihir Bellare, 2004).

To breakdown the above paragraph, it is important to understand the key concept of collisions. Collisions in the sense of algorithm creation, testing, and verification involve testing the

algorithm such as SHA-1 over many millions of iterations for identical hash outputs for a specific input or message digest. More specifically 2^L , where L is the number of bits in the message digest. 2^{32} for a message digest of length 32 bit, which are 4.29×10^9 evaluations (4.29 Billion), known as brute force attacks. This is the testing and validation evaluations where an identical output can be achieved from two different inputs or message digests (Tanenbaum, 1996).

For the signer of a drawing with their Digital ID, this means, for an attacker to duplicate a message digest from the user's signed drawing and thwart the successful transaction, they would need to perform 4.29 billion evaluations. As stated above, 2.14 billion on average to successfully create another message digest would be the same as the valid one the user signed. Consequently, this duplicate message digest would cause a fundamental breakdown in the certificate validation system. It is not that they duplicated the drawing (lines, line types, and layers); it is the intermediate process of duplicating the message hash, or digest, as the reader may recall, that is the unique output from the one-way hashed algorithm. Then, the message digest is handed to the next stage used for validating the public key certificate and Digital ID.

As PCs become more efficient, a concern about these new attacks is that they discover more efficient methods to thwart encrypted signatures like MD5, which is a relic in modern software, but is actually still used in database password hashing, since it is behind a firewall (not exposed to the public). A migration toward stronger hashes is believed to be practical. Many of the applications that use cryptographic hashes, such as password storage, are only minimally affected by a collision attack. Constructing a password that works for a given account requires a Preimage Attack (Rogaway & Shrimpton, 2009), as well as access to the hash of the original, which is most likely behind a firewall, and access may or may not be trivial. Reversing password encryption to obtain a password to try against a user's account elsewhere is not made possible by the attacks.

In the case of document signing, an attacker could not simply fake a signature from an existing document. The attacker would have to produce a pair of documents, one innocuous and one damaging, and get the private key holder to sign the innocuous document. There are practical circumstances in which this is possible, but for the average user, this scenario is highly unusual.

Hash Size

As discuss earlier, the output of an algorithm is expressed in bits or bytes. A MD5 hash has 128 bits (16 Bytes), and a SHA-0, SHA-1 both have 160 bits. The output hash size needs to be large enough to have no two output values be equal, a term known as a collision. Although most software is currently using the SHA-1 hash, the next generation SHA-2 family of algorithms (listed below) provides a larger output key.

SHA224("")	d14a028c2a3a2bc9476102bb288234c415a2b01f828ea62ac5b3e42f	224 bits or 28 bytes
SHA256("")	e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855	256 bits or 32 bytes
SHA384("")	38b060a751ac96384cd9327eb1b1e36a21fdb71114be07434c0cc7bf63f6e1da274edebfe76f65fbd51ad2f14898b95b	384 bits or 48 bytes
SHA512("")	cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e	512 bits or 64 bytes

Table 2 shows the SHA-2 family of hash functions, hash signature, and output bit length.

Function	1993	1994	2004	2005	2010	2011	2012
MD4							
MD5							
MD2							
RIPEMD							
HAVAL-128							
SHA-0							
SHA-1							
RIPEMD-160							
SHA-2 family							
SHA-3 (Keccak)							

Table 3 shows the strength of hash functions by showing the date it was broke. The dark grey areas show the year the function was broken.

4. SECURITY METHODS

Security is a vast, extensive field of study. An attempt to learn a specified field of security such as Cryptography in a six-hour course is a daunting task. In order to streamline the learning curve and narrow the subject matter for the non-computer scientist, many aspects of file security are not discussed in this course. Moreover, the main focus of digital signatures with Digital IDs hinges on the ability to use and distribute without concern for impersonation. Digital IDs are certificates purchased from third party providers, which the public key may be exported to distribute by email, FTP, or over the Internet.

Digital signatures are not the only mechanism which certificates are used, nor is it the only mechanism that Public Key Cryptography has an integral part in security. Consequently, certificates that are NOT distributed and managed by PKI (explained later), will NOT be discussed, nor will the practical uses of Digital Signatures without certificates be discussed.

Originally, it was the goal to discuss these differences, but the condensed version was over ten pages and admittedly not very useful unless designing and building a customized certificate was the intent of the reader. Moreover, the subject matter becomes joined with similar theory and practical uses, which convolutes the material into confusing examples that are not clear to the overall practice of signing CAD files. Hence, the focus of this course is limited to how to use and understand a Digital Signature managed by a certificate, also known as a Digital ID.

When signing a file over the Internet by a feature known as a Digital Signature, there are security mechanisms that use similar implementations as the Digital ID. However, the implementation of a Digital Signature that will be studied herein focuses on the subtopic of the Digital ID, which uses the PKI and is a part of the Digital Signature implementation.

There are four types of security techniques used to secure files that will be discussed. These methods are not necessarily mutually exclusive or independent of each other. They are listed below:

- ❖ Password Protect (lock)
- ❖ Message Encryption (hide)
- ❖ Digital Signatures (identify)
- ❖ One-Way Hash (verify)


The focus will be on Digital Signatures and One-Way Hash Algorithms. There are many methods and algorithms to secure files. The approach with this course is to learn the current methods actually implemented with Digital IDs and their corresponding algorithms.

Password Protect

In order to protect CAD files, one function of CAD systems is password protection. In many systems, password protection and file properties encryption are used together. In this course, the focus is on Digital Signatures with Digital IDs. However, a complete picture of different types of security would not be complete without mentioning password protection. Password protection puts a lock on the file. It protects the file in transit to the end user. The end user must know the password to open the file. However, if an attacker breaks the password, they can change the drawing, save the drawing with the original password and send it back to the originator, as if nothing changed.

It is important to understand that Password Protect does not give the full roundtrip protection and detection of changes to a CAD file. As illustrated, a hacker can break the password and impersonate the user by changing the drawing without the sender knowing. Only a Digital ID protects the file for the complete roundtrip, but it does not secure the drawing like password protection with file encryption. As a result, Password Protect functionality and Digital Signature perform two distinct functions. Password Protect locks the file. A Digital Signature verifies who sent it, and Message Encryption hides the contents. In various applications, these approaches are used together. One part verifies the sender and the other method protects the file.

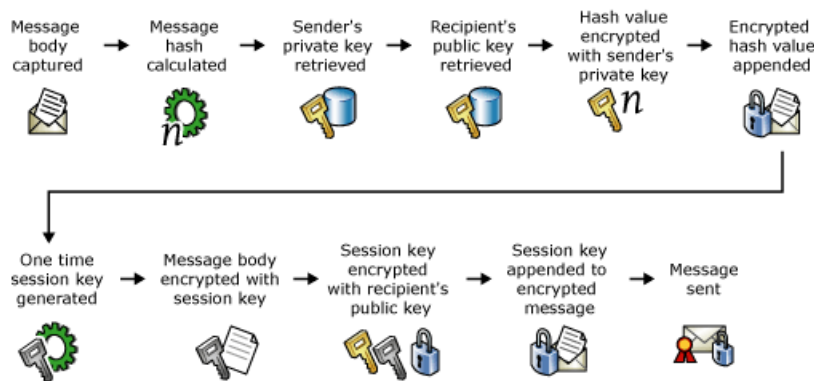
One method to help protect a password-protected document is to encrypt it inside a folder with stronger encryption mechanisms with software like “Bitser.” The software has a stronger level of encryption that can be used in coordination with a copyright or confidentiality agreement.

 Video tutorial is available on the support site for this topic.

Message Encryption

Digital Signatures with Message Encryption used with certificates completes the security in a few CAD systems Digital Signature implementation, as the Digital Signature authenticates and the Message Encryption hides the contents. With Digital Signatures, password protection may be an option but is a separate operation out of context for this topic. By using the public and private keys from the signer's Digital ID, the drawing may be signed with a Digital ID, and content hidden by a function called "encryption." The examples of this type of functionality are encrypting the drawing file properties, which does not hide the file content – only the metadata file properties.

With the purchase of a Digital ID, most email packages allow Message Encryption and Digital Signatures by importing the Digital ID. However, the Message Encryption requires the distribution of the public key to the recipient for a secure connection to be established as shown below. In order to encrypt files, the sender is trusting the recipient being a trustful individual. The recipient sends the sender their public key for the sender to encrypt the file. Upon the encrypted file's return to the recipient, the recipient's own installed private key of the matching pair will decrypt the message. Notice that Signing and Verifying with a Digital ID requires the distribution of the public key, and the sender's private key is used to encrypt or sign the data. In comparison, a digital signature with Message Encryption requires the public key of the recipient installed on the sender's computer to encrypt the data. The processes are similar in that comparison, but they differ as compared to which key does the signing or encrypting (See Figure 7).



**Figure 7 shows message encryption implemented by a Digital ID.
("How Digital Certificates are used ", 2005)**

A good written policy can be established with certain clients, designs, or internal personnel. If the CAD software does not support Message Encryption directly, then using the Digital ID installed into the email client will encrypt the contents of a correspondence. Make sure the email client encrypts the attachments along with the message. If not, then compress and encrypt the CAD files into a zipped file, which can be encrypted with a Password Protect option to decrypt the contents on the recipient's end.



Video tutorial is available on the support site for this topic.

Digital Signature

Digital Signatures are frequently used to implement electronic signatures, a term that refers to any electronic data that carries the intent of a signature, but not all electronic signatures use Digital Signatures. The focus is the specific case with PKI Certificates.

Digital Signatures are a type of asymmetric cryptography (using a public and private key exchange). For files sent through a less secure method such as email, a properly implemented Digital Signature gives the receiver reason to believe the file was sent by the originating sender. Digital Signatures are comparable to traditional handwritten signatures in many respects, but properly implemented Digital Signatures are more difficult to forge than the handwritten type. Digital Signature schemes in the sense used here are cryptographically based, and must be designed and implemented properly to be effective. Digital Signatures can also provide non-rejection, meaning the signer cannot successfully claim they did not sign a file, while also claiming their private key remains secret; furthermore, a few non-repudiation schemes offer a time stamp for the Digital Signature, so that even if the private key is exposed, the signature is valid. Digitally signed files may be anything representable as characters in a sentence. Examples include electronic mail, Word, Excel, and CAD Documents.

Digital certificates provide support to public key cryptography by providing a reliable means to distribute and access public keys. When a sender is signing a CAD file, the sender provides the public key that is associated with the private key available on the digital certificate. In turn, when the recipient is validating a Digital Signature on a file, the recipient is obtaining the public key to perform that operation from the sender's digital certificate. Figure 8 shows the following sequence of signing with the addition of the supporting elements of digital certificates.

1. CAD file, the Message is captured.
2. Hash value of the message is calculated.
3. Sender's private key is retrieved from the sender's digital certificate.
4. Hash value is encrypted with the sender's private key.
5. Encrypted hash value is appended to the message as a Digital Signature.
6. Message is sent.

To verify the signature, the following steps are required shown in Figure 9.

1. Signed CAD file, the Message is received.
2. Digital Signature containing encrypted hash value is retrieved from the message.
3. Message is retrieved.
4. Hash value of the message is calculated.
5. Sender's public key is retrieved from the sender's digital certificate.
6. Encrypted hash value is decrypted with the sender's public key.
7. Decrypted hash value is compared against the hash value produced on receipt.
8. If the values match, the message is valid.

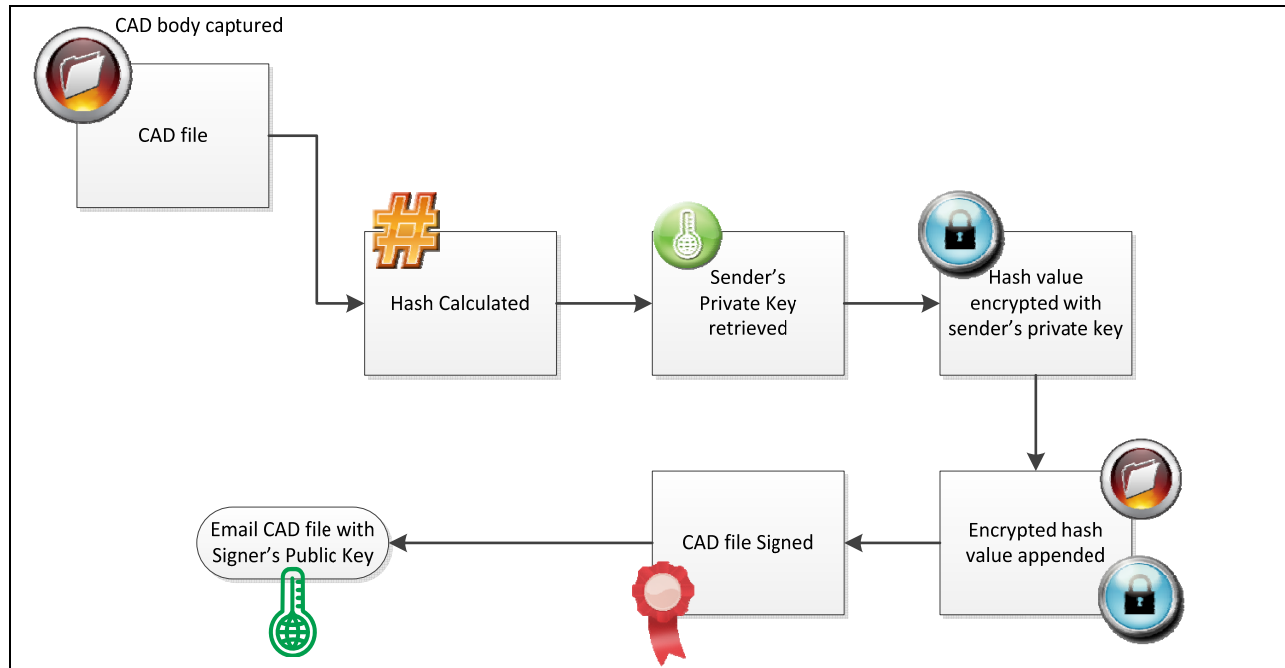


Figure 8 shows the CAD file being digitally signed by the signer.

The following figure shows the sequence of verifying a CAD file with digital certificates.

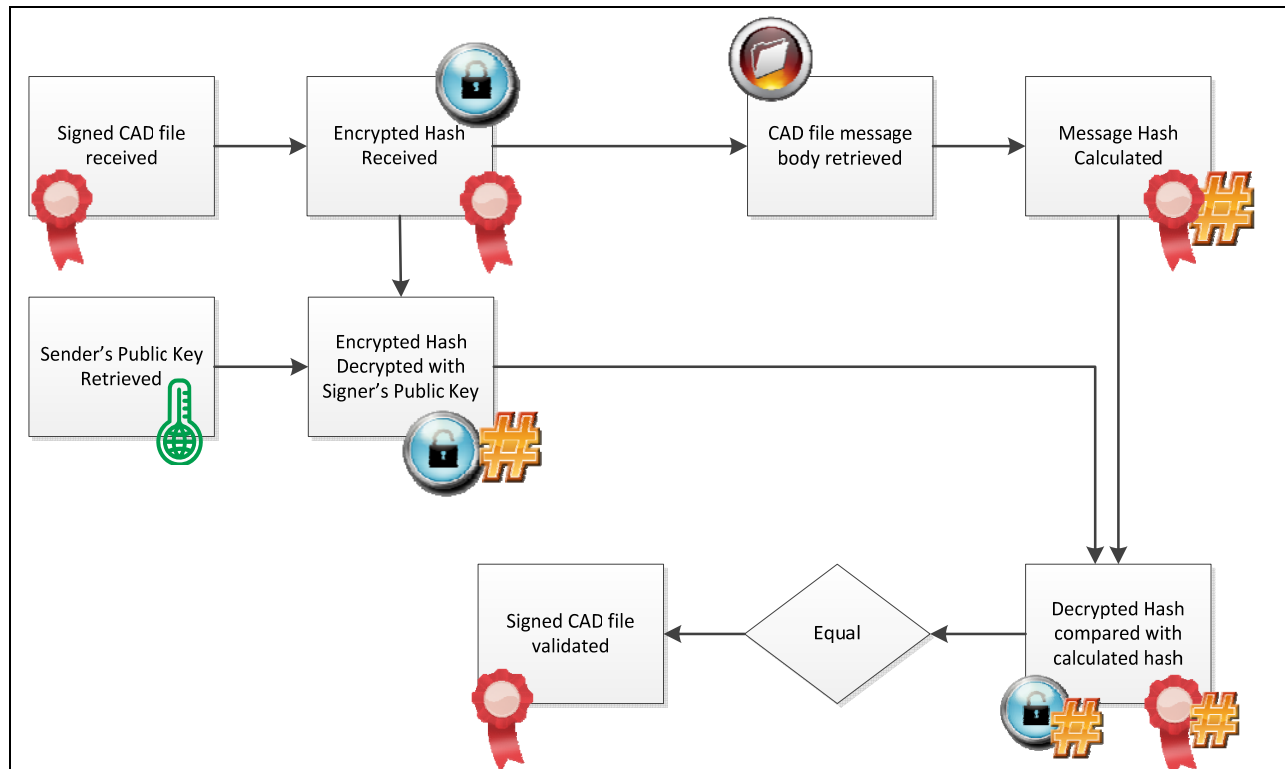


Figure 9 shows the verification process of a signed CAD file.

Applications

As AEC organizations move away from paper documents and blueprints with ink signatures or authenticity stamps, Digital Signatures can provide added evidence to the origin, identity, and content of electronic CAD files as well as acknowledging informed consent and approval by a the undersigned. The United States Patent and Trademark Office (USPTO) and the United States Government Printing Office (GPO) publish electronic versions of the budget, public and private laws, and congressional bills with Digital Signatures. Universities including Penn State, University of Chicago, and Stanford are publishing electronic student transcripts with Digital Signatures. It is time for the AEC Industries to start adopting the practice of signing all CAD files and PDFs that are sent electronically to clients with Digital IDs.

Authentication

Although CAD files may include information about the entity sending the file, that information may not be accurate. For example, receiving a CAD file from another employee other than the signer is more common than receiving the file from the undersigned. Here are a few basic questions to ask next time a CAD file is received:

- ❖ Who is the responsible party?
- ❖ Who either possesses the signed hard copy or digitally signed files?
- ❖ Did the sending party edit the drawing, and send it with full consent of the professional who signed the hard copies? If yes, where is the consent?
- ❖ Was the sender part of the project relating to the CAD files?
- ❖ Did the sender send the correct file version?

In an effort to not interject opinions here, let's suffice to say, everything is okay until there is a problem. The sender could eloquently answer the above questions, but if there is no assignment of responsibility with an approved method of authentication, the recipient will be in a vulnerable position.

Digital Signatures can be used to authenticate the source of the file. When ownership of a Digital Signature secret key is bound to a specific user, a valid signature proves the message was sent by that user. The importance of high confidence in sender authenticity is especially obvious in critical design deliveries. For example, suppose an engineering branch office sends instructions to the central office requesting a change in the design. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a serious mistake.

Integrity

In many circumstances, the sender and receiver of a CAD file may have a need for confidence that the message has not been altered during transmission. Although encryption hides the contents of a file, it may be possible to modify an encrypted message without understanding it. However, if a file is digitally signed, any change in the CAD file after signature will invalidate the signature. A Digital Signature is not Message Encryption. A Digital Signature only validates the signer - not Password Protecting the drawing. Furthermore, there is no efficient way to modify a file and its signature to produce a new file with a valid signature, because this is still considered to be computationally infeasible by most cryptographic hash functions, as notated in a term called “collision resistance,” which is out of scope of this seminar. However, it is defined in the Glossary.

When a client downloads a file over the Internet, he or she needs to be sure the file received is the correct one. The client needs to be assured of the file’s reliability and integrity. Many people make the following assumptions when they download a file over the Internet:

- ❖ The file is not a malicious program.
- ❖ The file has not been replaced, unbeknown to the server’s owners, by a malicious program.
- ❖ There is not another computer between the sender and the server, sending the user a different file than the one he or she wanted or modifying the file the originator sent. This is the “man-in-the-middle attack.”

These concepts would not have been an issue in the past, since electronic data was kept to a minimum. Now, most document content is electronic and stored on a server. As the movement to switch from paper deliveries to electronic deliveries continues to increase, professionals need to make the transition and keep in conformity with the professional standards as it pertains to Digital IDs, Digital Signatures, and signed and sealed documents.

An important concept to discuss is the full roundtrip verification of electronic transmission of CAD files. Most professionals trust if they have the signed and sealed prints in their cabinet, then those prints are the originals. The undersigned is not concerned with what changes were made to prints that were delivered to the client after that point in time. In the past, that was the situation for everyone. However, if the user sends those same CAD files by email, FTP, or downloaded from a server, the files sent electronically are the new “originals” because the nature of the electronic medium is different from the hard copy of signed and sealed prints. Moreover, if there were any changes to that drawing at any point along the electronic pathway, the signer has no tracking mechanism without using Digital Signatures or manual hashes. Without full verification by a Digital Signature, the signer is in a vulnerable position.

A concept has been used over the past decade to erase all traces of the company logo and signing block before sending the client a drawing. While this may give the illusion of anonymity, the sender’s digital fingerprints are everywhere. From the CAD file containing pertinent creation information to email correspondence, it would be very difficult to hide any trace of the file origin.

The term “Digital Fingerprint” may not have meaning to the average users, but electronic transactions can be traced. Likewise, the concept of electronically tracing files is certain to increase in the years to come. Undoubtedly, if Google keeps all the searches from its search engine from a decade ago and the US Government hoards all the emails that are ever sent, any anonymity was lost years ago. Digital Signatures establish a company’s reputation for legitimacy and being technologically savvy by using ideas presented in this course. They will help mitigate these types of situations, while giving the final recipient (the client) a measure of file integrity from the signer.

Non-Abandonment

Non-Abandonment, or more specifically, non-rejection of the drawing origin, is an important aspect of Digital Signatures. By definition, an entity or licensed professional that has signed a CAD drawing cannot at a later time deny having signed it. Similarly, access to the signer’s public key does not enable a fraudulent party to fake a valid signature. This topic will be explained in more detail later.

One-Way Hash

A Cryptographic Hash Function is a hash function that is defined as an algorithm that takes an arbitrary block of data and returns a fixed-size bit string. (See - Algorithm Types MD5 and SHA1 hashes for detailed description or figure below.)

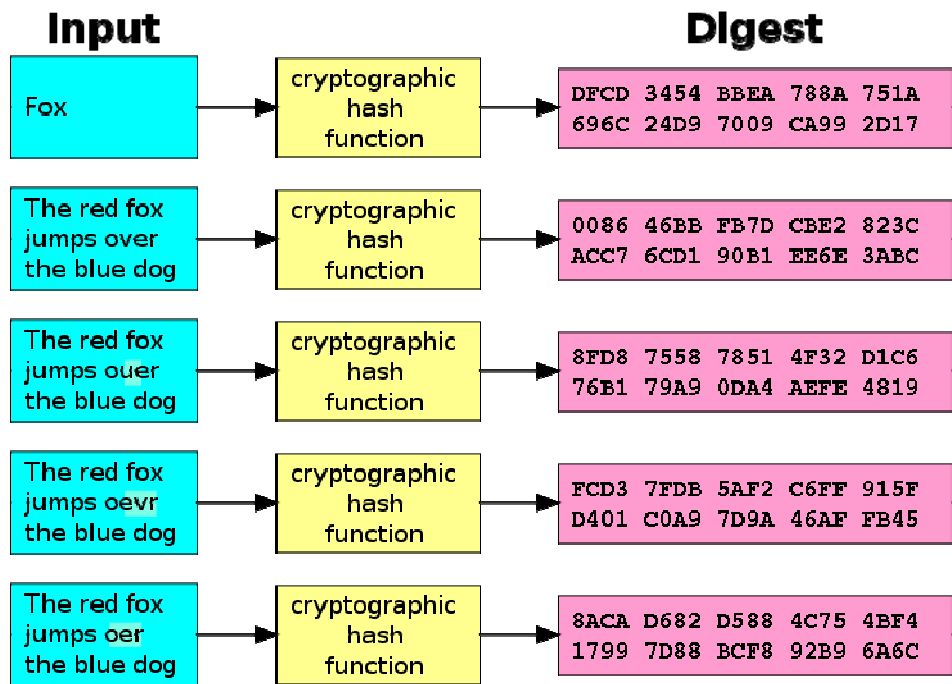


Figure 10 shows a typical one-way hash function converting text into ciphertext or digest.

A Cryptographic Hash Function such as the Secure Hash Algorithm (SHA-1) produces a hash code 160-bits in length. These hashes are unique output values in that even small changes in the

source input (a change in the CAD file) drastically changes the resulting output hash, by the avalanche effect.

A Cryptographic Hash Function is a hash function that is an algorithm that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that any (accidental or intentional) change to the drawing will change the hash value. The data to be encoded are often called the "message," and the hash value is sometimes called the message digest or simply "digests" (Cryptographic Hash Function).

The ideal Cryptographic Hash Function has four main properties:

- ❖ It is easy to compute the hash value for any given file.
- ❖ It is infeasible to generate a message that has a given hash.
- ❖ It is infeasible to modify a file without changing the hash.
- ❖ It is infeasible to find two different files with the same hash.

There are several reasons to sign such a hash (or message digest) instead of the whole document or in this case a CAD file (Digital Signatures, 2011).

For efficiency: The signature will be much shorter and thus save time since hashing is generally much faster than signing in practice.

For compatibility: Messages are typically bit strings, but particular signature schemes operate on other domains or areas. A hash function can be used to convert an arbitrary input into the proper format.

For integrity: Without the hash function, the text "to be signed" may have to be split (separated) in blocks small enough for the signature scheme to act on them directly. However, the receiver of the signed blocks is not able to recognize if all the blocks are present and in the appropriate order.

5. CERTIFICATES

Before delving into the broad topic of certificates, a gentle reminder or disclaimer is needed. When we discuss certificates in a broad sense, certain aspects may not apply to the specific case of Digital IDs. Public Key Certificates are used whenever users login to a computer system, remote desktop into another PC, or encrypt files on Windows PCs. With regards to large organizations that serve as its own Certificate Authority (CA) with rows of servers, the local IT department manages all users' access with certificates installed on their machines. This idea of "chaining" applies in that context, but it does not necessarily apply to the passing of the Digital ID (which users purchase directly from the CA). If clients pass the sender's Digital ID public key to a colleague, they will install a new certificate on their PC, which will grant them access to verify the sender's drawing, not sign or encrypt it. The chain of trust is limited to the sender's root CA, as in the image in Figure 11 that shows that an AEC Signature was purchased at VeriSign and is a Class 1 Certificate for signing files.

In the following example, the drafts-person will transfer a drawing to their client by email or another electronic transmission. If the drafts-person knows the client's public key, he or she can help the client by using something called a "certificate," issued by one person, stating the public

key of another person has a certain value. Essentially, a certificate is a signed public key. If the draft person creates the certificate by placing information about him/her and information about the third party, usually "Certificate Authority" such as "VeriSign," and the third party's public key value into a file, the drafts-person then signs the file with his or her own private key. The client can download the certificate and verify it using the drafts-person's public key. The client trusts the drafts-person, so the client also now has a trustworthy copy of the third party's public key, which he or she can use to verify files signed by the third party.

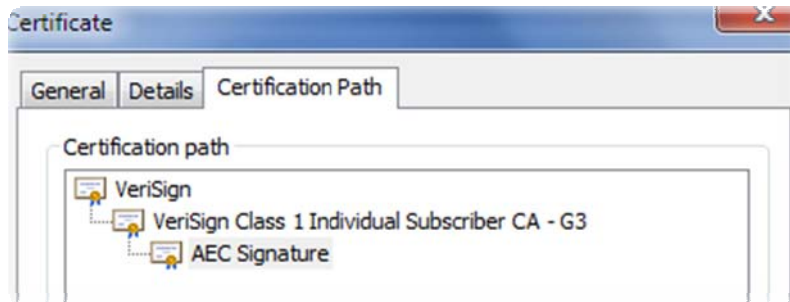


Figure 11 shows certificate chain of trust or pathway leading to the CA.

Certificate Chains

Digital certificates are verified using a chain of trust. The anchor for the digital certificate is the Root Certificate Authority (CA), that we will term the "root certificate."

A certificate authority can issue multiple certificates in the form of a tree structure. A root certificate is the top-most certificate of the tree, the private key that is used to "sign" other certificates. All certificates immediately below the root certificate inherit the trustworthiness of the root certificate. A signature by a root certificate is similar to "notarizing" a file for recording in the clerk of the court to prove identity in the physical world. Certificates further down the tree also depend on the trustworthiness of the intermediates (often known as "subordinate certification authorities").

Many software applications assume these root certificates are trustworthy on the user's behalf. For example, a web browser uses them to verify identities within secure connections. However, this implies that the user trusts the browser's publisher (i.e. Microsoft, Google, e.c.t), and have the root certificates of the CA it trusts installed along with any intermediate certificates the CA may have issued like a certificate-issuing-certificate, to faithfully verify the identity and intentions of all parties that own the certificates. This (transitive) trust in a root certificate is the usual case and is integral to the X.509 certificate chain model.

The root certificate is usually made trustworthy by a mechanism other than a certificate, such as "secure physical distribution." For example, several of the most well known root certificates are distributed through the Internet browsers by their manufacturers.

To verify a certificate, users need a public key. To verify a public key, users need a certificate. Essentially, one certificate can be verified by another, which is verified by another, and so forth. This is called “certificate chaining.” **The chain cannot be infinite, so where does it start? The certificate chain starts with a certificate whose issuer and subject are the same.** Usually, such a certificate is issued by a Certificate Authority or by a dependable institution like VeriSign.

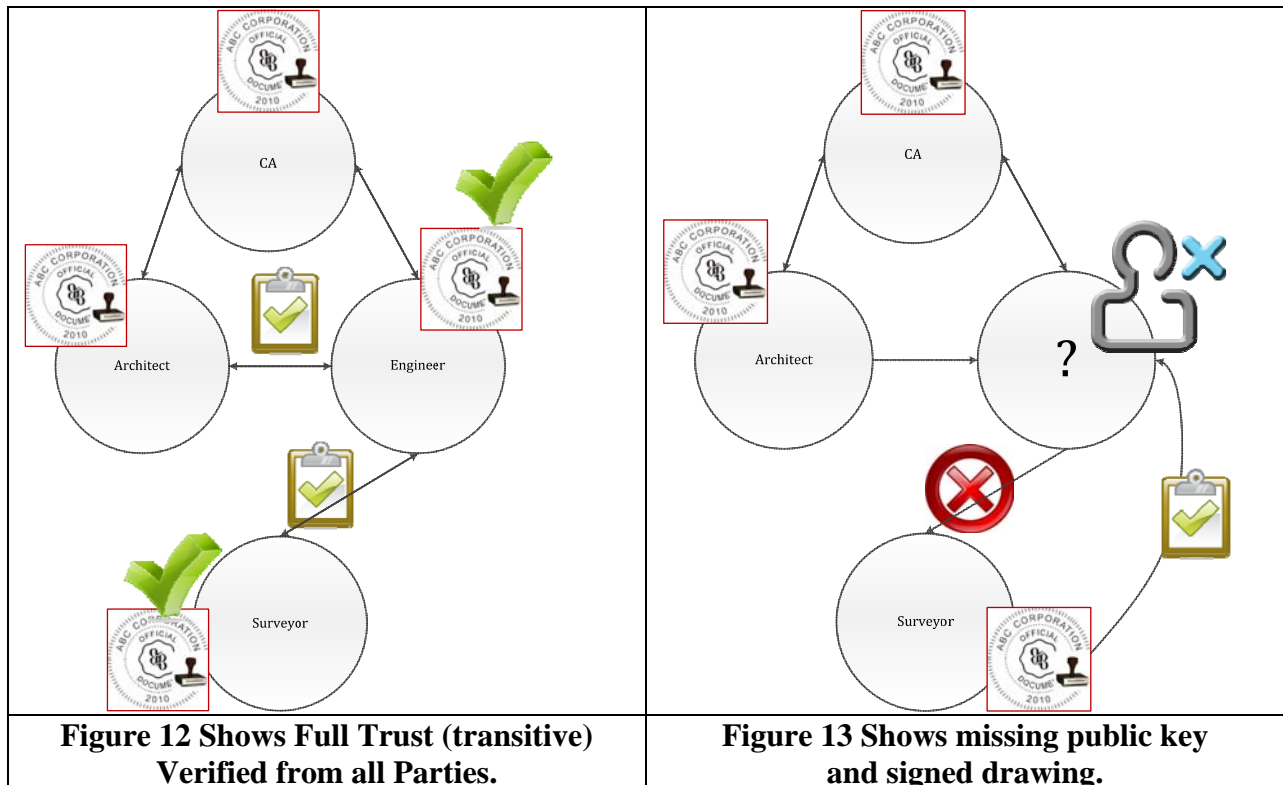
Using certificates to prove authenticity depends on a chain of certificates that ultimately terminates on a self-signed certificate. However, self-signed certificates are not secure at all. Anyone can generate a self-signed certificate, claiming to be the post office or the President of the United States. Why would users ever trust a self-signed certificate? Users can trust a self-signed certificate if they are able to verify it. One convenient way to verify certificates is a calculating message digest of the entire certificate, commonly known as a “certificate thumbprint.” To verify a thumbprint, call the people who issued the certificate and have them read off the numbers of the thumbprint. Another option is for the certifying authority to widely publish their self-signed certificate’s thumbprint, perhaps in a newspaper or magazine as well as online. If users obtain a thumbprint from several sources and they all match the thumbprint on the self-signed certificate the user possesses, the certificate is likely to be trustworthy. (See Thumbprint Algorithm for procedure to verify a certificate).

Currently, most self-signed certificates are embedded into web browsers. When users download and run a browser, it can recognize certificates issued by a dozen or so popular certifying authorities, using internal self-signed certificates from these authorities. How do users know that somebody tricky hasn’t modified the self-signed certificates as a user downloaded the browser? They don’t. If users are worried about this attack, they should verify the self-signed certificate fingerprints and the browser before they accept any certificates issued by the certifying authority.

As a practical example, let’s examine Figures 12 and 13 below. Start by looking from the bottom up because when you receive any CAD file, you are at the bottom of this transaction, as shown below. CAD software does not keep more than one certificate transaction embedded. This means once a drawing is signed by a Digital ID, if the drawing is signed again with a Digital ID from a third party, it overrides any information from the first signer. This is exactly what you want and expect. They last person to sign the drawing is the responsible party. Reviewing the figure on the left hand side, the Surveyor receives by email a public key (a clipboard with a check mark) from the Engineer along with the signed CAD drawing. The Surveyor installs the Engineer’s public key certificate from the Engineer. From this information, the Surveyor can verify the CAD file has not changed since leaving the Engineer’s computer, and the drawing was signed by the Engineer because the public key was sent along with the drawing. As a result, full verification of the source and drawing is achieved.

From Figure 9 on the right-hand side, the Surveyor is emailed an unsigned drawing. There is no public key certificate sent along with the drawing and the drawing has not been signed with a Digital ID. This is the way most transactions occur in the US. Once the drawing is sent without a Digital Signature, there is no tracking mechanism attached to the drawing. As a result, the Engineer cannot verify if the Surveyor sends the drawing back, or whether any changes were made. As a discussion in theory only (probably not good policy), in the example, the Surveyor has a Digital ID and signs the drawing. The Surveyor exports his public key certificate and emails the certificate back to the Engineer. The Engineer installs the Surveyor’s certificate, reviews and approves the content and sends the Surveyor an email approving the statement. The

Engineer cannot sign the drawing with the Surveyor's public key certificate, acting as if it were his Digital ID. There is no private key attached. Consequently, the most that can be given would be to ask the Engineer to manually calculate a manual hash on the drawing (or zipped file) using free software like Bitser, which will give the Surveyor the ability to calculate the hash on his end to verify if the files are indeed the same. As a result, using the hash can give the Surveyor partial trust, since verification of the source is not truly verified without a proper Digital ID.



Certificate Content

To verify a signature, users need the signer's public-key. So how are public keys distributed securely? The recipient could simply download the key from the server somewhere, but how are they to know they received the correct file and not a forgery? Even if they receive a valid key, how do they know that it belongs to a particular person?

Certificates answer these questions. A certificate is a statement, signed by one person, that a public key of another person has a particular value. In several ways, it's like a driver's license. The license is a document issued by a state government that matches the owner or driver's face to their name, address, and date of birth. When they purchase alcohol, the cashier can use the buyer's driver's license to prove identity and age.

Note that the license only has value because the local shops and grocery stores trust the authority of the state government. Digital Signatures have the same authentication. They need to trust the person who issued the certificate, who is known as the certificate authority.

In cryptographic terminology, a certificate associates an identity with a public key. The identity is called the “subject.” The identity that signs the certificate is the “signer.” The certificate contains information about the subject and the subject’s public key, plus information about the signer. The entire file is cryptographically signed, and the signature becomes part of the certificate. Because the certificate is signed, it can be freely distributed over insecure channels.

Simply, a certificate contains these elements:

- ❖ Information about the subject
- ❖ The subject’s public key
- ❖ Information about the issuer
- ❖ The issuer electronic signature of the above information

A “public key certificate” (also known as a “digital certificate” or “identity certificate”) is an electronic document that uses a Digital Signature to bind a public key with an identity – information such as the name of a person or an organization, their address, and so forth. The certificate can be used to verify that a public key belongs to an individual.

In a typical PKI scheme, the signature will be of a CA. In a web of trust scheme, the signature is of either the user (a self-signed certificate) or other users (“endorsements”). In either case, the signatures on a certificate are confirmations by the certificate signer that the identity information and the public key belong together.

For provable security this reliance on something external to the system has the consequence that any public key certification scheme has to rely on specific special setup assumption, such as the existence of a certificate authority. Below is a list of definitions of attributes typically found in an X.509 version of a certificate:

- ❖ **Version.** The X.509 version number.
- ❖ **Serial number.** The unique serial number that the issuing CA assigns to the certificate. The serial number is unique for all certificates issued by a given CA. This number does not change unless it is updated by a CA when renewing. It should not change for a minimum of one year.
- ❖ **Signature algorithm.** The hash algorithm that the CA uses to digitally sign the certificate.
- ❖ **Issuer.** Information regarding the CA that issued the certificate.
- ❖ **Valid from.** The beginning date for the period in which the certificate is valid.
- ❖ **Valid to.** The final date for the period in which the certificate is valid.

- ❖ **Subject.** The name of the individual, computer, device, or CA to whom the certificate is issued. If the issuing CA exists on a domain member server in a user's enterprise, this will be a distinguished name within the enterprise. Otherwise, this may be a full name and e-mail name or other personal identifier.
- ❖ **Public key.** The public key type and length associated with the certificate.
- ❖ **Thumbprint algorithm.** The hash algorithm that generates a digest of data (or thumbprint) for Digital Signatures.
- ❖ **Thumbprint.** The digest (or thumbprint) of the certificate data.
- ❖ **Friendly name.** (Optional) A display name to use instead of the name in the Subject field.
- ❖ **Enhanced key usage.** (Optional) The purposes for which this certificate can be used.

Distinguished Names

Names in X.509 certificates are not encoded simply as *common names*, such as "AEC Signature," or "Certificate Authority XYZ," or "CAD Administrator." They are encoded as distinguished names, which are a comma-separated list of name-value pairs. For example, the following could be my distinguished name:

- ❖ O=Banks & Banks Consulting
- ❖ OU=Geospatial Department
- ❖ CN=AEC Signature (could be users name here)
- ❖ E=help@banksandbanksconsulting.com

So what do "O," "OU," and "CN" mean to Digital ID users. A distinguished name can have several different attributes. The most common are the following:

- ❖ (O) Organization
- ❖ (OU) Organizational Unit
- ❖ (CN) Common Name (the user's name, or software)
- ❖ (C) Country
- ❖ (E) Email Address

Signature Algorithm

Certificate Authorities create and maintain their root (top most) level certificate. These certificates are found in the certificate store on the PC. To view installed certificates, open any web browser and navigate to the certificate settings. CAs create certificates by first generating key pairs by a process called "key generation." Key generation has two phases. The first phase is a choice of algorithm parameters which may be shared between different users of the system, while the second phase computes public and private keys for a single user. Remember, the hash algorithm SHA-1 requires no input key; it returns a hash value from the input file. The

combination of Sha1RSA algorithm requires input keys to produce the public and private output keys. (See - Sha1RSA under algorithms.)

In Appendix C, the Signature Algorithm is the hash algorithm (such as SHA-1 in this case), which has no input key. It is used to generate the 160-bit hash or message digest, which is then encrypted by the RSA part of the algorithm to use the public and private keys to either sign or verify content of a file.

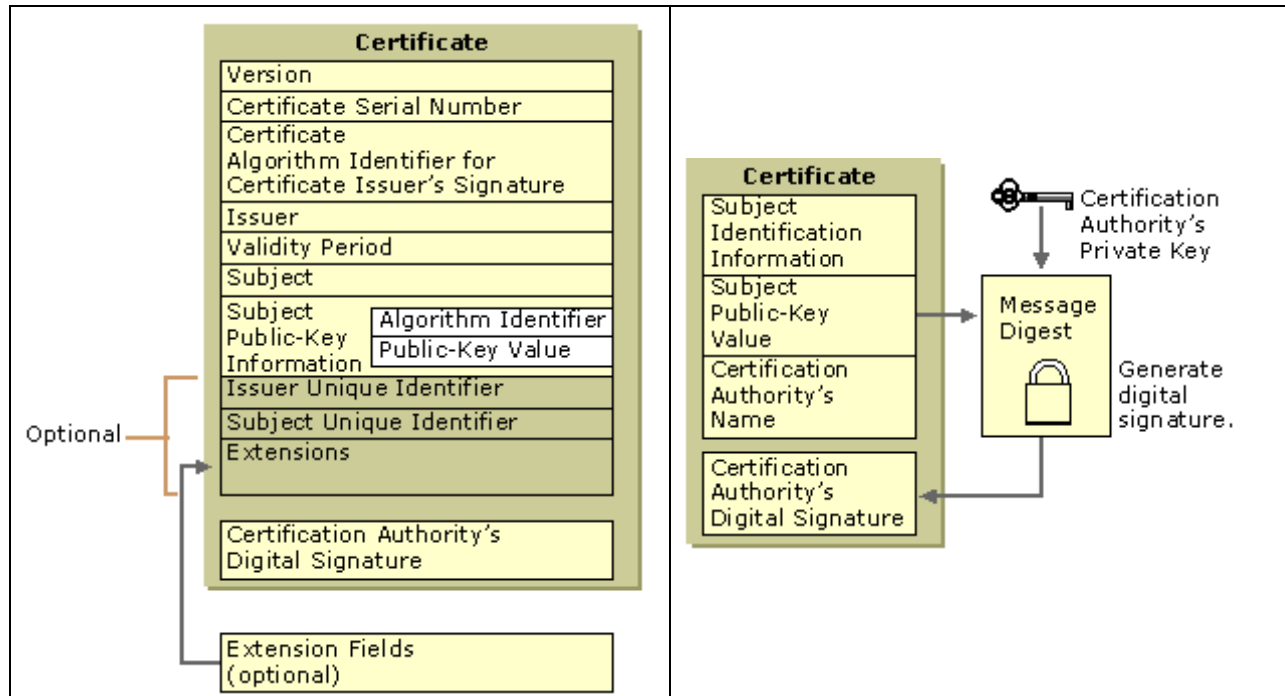
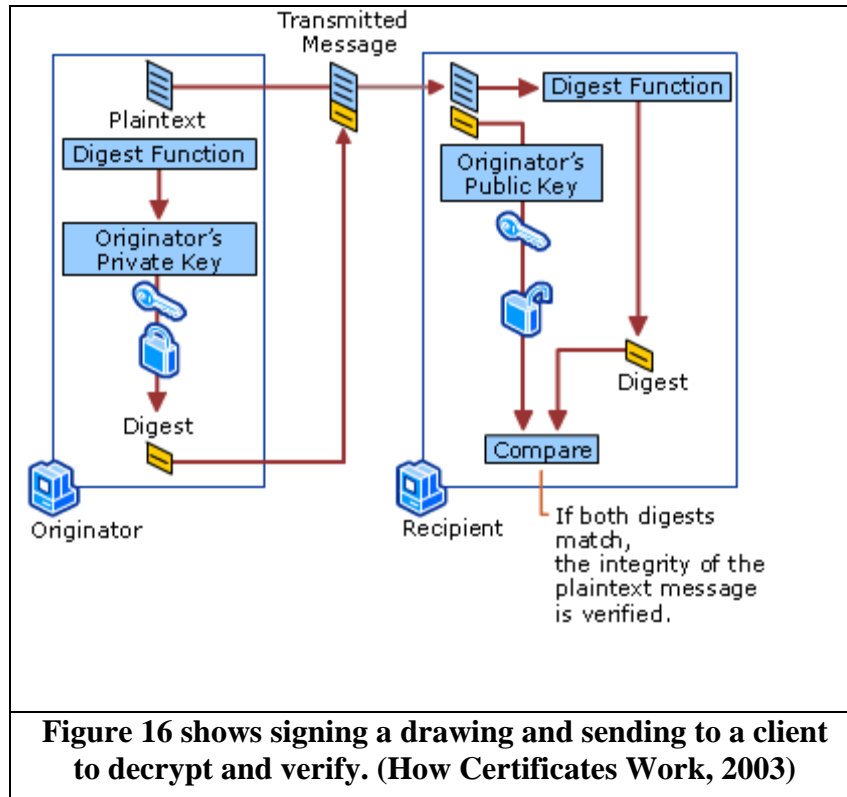


Figure 14 shows the contents of X.509 version 3 certificates. (Microsoft Technet, n.d.)

Figure 15 shows how a certificate is signed by the issuing CA. (Microsoft Technet, n.d.)

A certificate contains specific content that identifies the certificate's owner (called the “subject”) as an entity. A certificate also contains the owner's public key. Moreover, a certificate identifies the CA (called the “issuer”) that issued the certificate. A CA uses its private key to digitally sign each certificate it issues. To create a Digital Signature, the CA generates a message digest from the certificate, encrypts the digest with its private key, and includes the Digital Signature as part of the certificate. Anyone can use the message digest function and the CA's public key to verify the certificate's integrity. If a certificate becomes corrupted or someone alters it, the message digest for the altered certificate does not match the digest in the CA's Digital Signature (See - Thumbprint Algorithm). Figure 15 shows how a certificate is signed by the issuing CA.



Thumbprint Algorithm

In the Signature Algorithm paragraph, it was stated that a CA Provider's certificate can be validated. The Thumbprint and the Thumbprint Algorithm are properties of a certificate and defined as follows:

Thumbprint: The calculated hash of the certificate in DER encode X509 format.

Thumbprint Algorithm: The specific hash used to calculate the hash value, which in this instance was SHA-1.

Note: This concept is to compare the CA's root certificate installed in the local web browser to the Digital ID issued or purchased by the end user.

To calculate the hash of a CA provider's certificate and verify its Thumbprint by comparing the issued Digital IDs thumbprint, the original certificate from the CA (VeriSign) installed with the web browser must be exported into a specific Distinguished Encoding Rules (DER) encoded X509 format. Accomplish this by exporting the certificate with Internet Explorer and exclude the password on export. By excluding the password on export, the software recognizes the difference and does not include all the properties that the other format Personal Information Exchange Syntax Standard (PKCS #12) includes. The PKCS #12 standard specifies a portable format for storing or transporting a user's private keys. To verify the hash of a certificate, the password must not be included. Likewise, other properties like the Thumbprint and Thumbprint Algorithm are not included in the export as they are part of the final hash comparison.

Enhanced Key Usage

Certificates have specific functions that may be altered by the CA. If the recipient of a certificate is using the certificate for file encryption for access to files on a remote server, then the settings may be adjusted allocated permissions for file encryption, as shown on the Figures in Appendix C. Moreover, if the recipient would like to sign and encrypt the files, then permission must be given to secure email and client authentication. When using certificates to sign drawings, verify the certificates enhanced key usage for the correct settings. Below are just a few key uses:

- ❖ Server Authentication
- ❖ Time Stamping
- ❖ Code Signing
- ❖ Client Authentication
- ❖ Secure Email
- ❖ Encrypting File Systems


Vendor Defined Classes

VeriSign uses the concept of classes for different types of digital certificates:

- ❖ Class 1 for individuals, intended for email and Digital IDs.
- ❖ Class 2 for organizations, for which proof of identity is required.
- ❖ Class 3 for servers and software signing, for which independent verification and checking of identity and authority is done by the issuing certificate authority.
- ❖ Class 4 for online business transactions between companies.
- ❖ Class 5 for private organizations or governmental security.

In the case for CAD drawings, the Class 1 Certificate will enable the user to send encrypted emails and sign CAD drawings with a Digital ID or Class 1 Certificate.

6. DIGITAL IDENTIFICATION

 **Note:** *Digital Signatures using Digital IDs are presented with redundant examples to help clarify the process for the broad spectrum of readers. Remember, Digital IDs MUST contain both the private and public keys while certificates MAY contain one or both key pairs.*

Analogy for Digital ID

Considering the intended AEC audience that uses Digital Signatures is most likely less than 10%, this analogy will help visualize Digital IDs in practice.

Consider the signer of CAD files has a car and the car has two keys. One key is for the door and ignition, the master key (private key). The other key is for the trunk of the car, the spare key (public key). This example is for cars that did not have the universal key as they do now where one key fits all. If the signer has signed and sealed prints in the trunk of the car, the signer has access to both keys to the car. The master key opens all the doors, but the spare key opens only the trunk. The signer gives the spare key to the client to pick up a hard copy of signed and sealed prints in the trunk of the car. By giving the client the spare key, he does not have access to the

ignition so the client cannot steal the signer's car. The client only has access to the trunk and to the materials in the signer's trunk, which in this case are hard copies of the signed and sealed prints.

The client, by having the signer's spare key, knows these are legitimate signed drawings. The client can take the prints out of the trunk, alter them and place them back in the trunk. As a result, when the signer opens the trunk, the drawings will appear altered and a notification will be placed on the drawings about the modification. Without a private key, the client cannot sign the drawings and put them back in the trunk, appearing unchanged. If the client did have a private key of their own, they could sign the original drawings and put him back in the trunk. The car owner (original signer) can then review the drawings and receive a notification the drawing have been signed with another Digital Signature by the client and have not been changed since signing. In order for the signer to view this information, the client would need to share their public key or the key to their trunk. This is called "transitive trust scheme." By both parties sharing their public keys, a mutual trust is established.

The signer or client's car was not stolen since the spare keys did not allow access to the ignition. Also, there was mutual respect when accessing the materials in the truck of the car. In practical terms, the signer does not care who receives the public key. The public key only allows the client to validate the signer's digital thumbprint information, stating indeed the signer is verified by a third party, and the information contained in the drawing has not been altered since the last save.

What is a Digital ID?

A Digital ID or "public key certificate" (also known as a "digital certificate") is an electronic document that uses a Digital Signature to bind a public key with an identity, information such as the name of a person or an organization and their address. The certificate can be used to verify a public key belongs to an individual.

A Digital ID is like an electronic driver's license or passport that proves identity. A Digital ID usually contains the name and email address, the name of the organization that issued it, a serial number, and an expiration date. Digital IDs are used for certificate security and Digital Signatures.

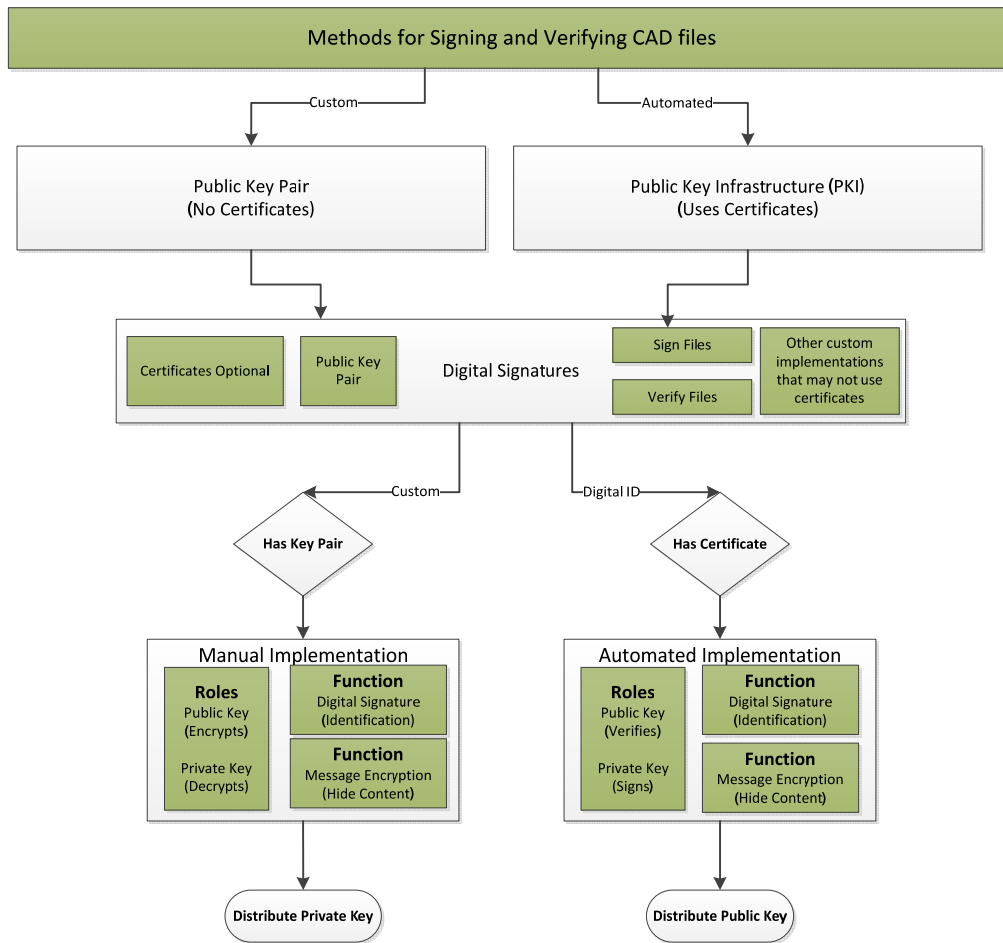


Figure 17 shows the differences in (PKI with Digital ID) and customized Public Key Pair for illustration purposes.

⚠ Note: Digital Signatures using the Public Key Pair instead of Certificates standard implementation of the PKI use the public key to encrypt and the private key to decrypt files. This operation is opposite to which key does the signing when compared to CAD systems. Other systems may be different, but Autodesk and Adobe use the opposite format, as the signer uses the sender’s private key to sign or certify the file and distribute the public key.

With Digital IDs and CAD files, the process is reversed compared with the Message Encryption process shown in Figure 17. In signing CAD files, use the private key to sign the file, and the public key to verify the signature. Without the private key, the client cannot attempt to sign the file, which is by design. It can only verify the CAD file is unchanged.

Pertaining to Message Encryption (standard PKI operation) as shown to the right of Digital ID on the flowchart above, Digital IDs contain two keys, the public key locks, or encrypts data, and the private key unlocks, or decrypts, that data. When a user signs PDFs, they use the private key to apply their Digital Signature (PKI operation with certificates). The public key is in a certificate that users distribute to others. For example, users can send the certificate to those who want to validate their signature or identity. Users store the Digital ID in a safe place because it

contains the private key that others can use to decrypt the originator's information. Export the certificate without the password to obtain a public key distributable certificate, then distribute the Digital ID public key by email, FTP, or Internet to a client.

Why Do I Need One?

Users need a Digital ID to sign a document or encrypt files, drawings, or PDFs through a public private key certificate mechanism.

How Do I Obtain One?

Users can get a Digital ID from a third-party provider (VeriSign), or they can create a self-signed Digital ID.

Self-Signed Digital IDs

Self-signed Digital IDs can be adequate for personal use or small-to-medium businesses. It should not be used for commercial or licensed professional signing and distribution. Self-signed PDF documents are not the focus of this course. The primary focus is protection of CAD drawings that could be in dispute and end up in a court of law. Furthermore, a certificate from a CA is authorized and verified by their email address that eliminates the ability for the signer to reject ownership of the drawing at that point. Self-signed certificates use should be limited to parties that have established mutual trust.

Certificate Authorities

Most business transactions require a Digital ID from a trusted third-party provider, called a "certificate authority." Because the certificate authority is responsible for verifying the user's identity to others, choose one that is trusted by major companies doing business on the Internet. A few CAs are listed below:

- ❖ Digi-Sign
- ❖ GeoTrust
- ❖ VeriSign
- ❖ DigiCert

Digital IDs vs. Sealed

An ink signature could be replicated from one document to another by copying the image manually or digitally, but to have credible signature copies that can resist scrutiny requires a significant manual or technical skill. To produce ink signature copies that resist professional scrutiny is very difficult. Today, CAD files may have the signature bound or attached. It is not recommended to bind a signature image into the CAD file. As shown in the figure below, it is not acceptable (by its absence in the figure) to attach a scanned signature, incorrectly assuming it is the Digital Signature.

Digital Signatures cryptographically bind an electronic identity to an electronic document and the Digital Signature cannot be copied to another document. Paper contracts sometimes have the

ink signature block on the last page and the previous pages may be replaced after a signature is applied. Digital Signatures can be applied to an entire document or CAD file such that the Digital Signature on the last page will indicate tampering if any data on any of the pages have been altered. This can also be achieved by manually signing all pages of the contract with ink.

Important CAD documents are signed in ink with all involved parties meeting in person, with additional identification forms other than the actual presence (i.e., driver's licenses, passports, fingerprints, etc.), and most often the presence of a respected notary that knows the involved parties. The signing usually occurs in a building that has security cameras and other forms of identification and physical security. The security that is added by this type of ink on paper signatures cannot be currently matched by digital only signatures.

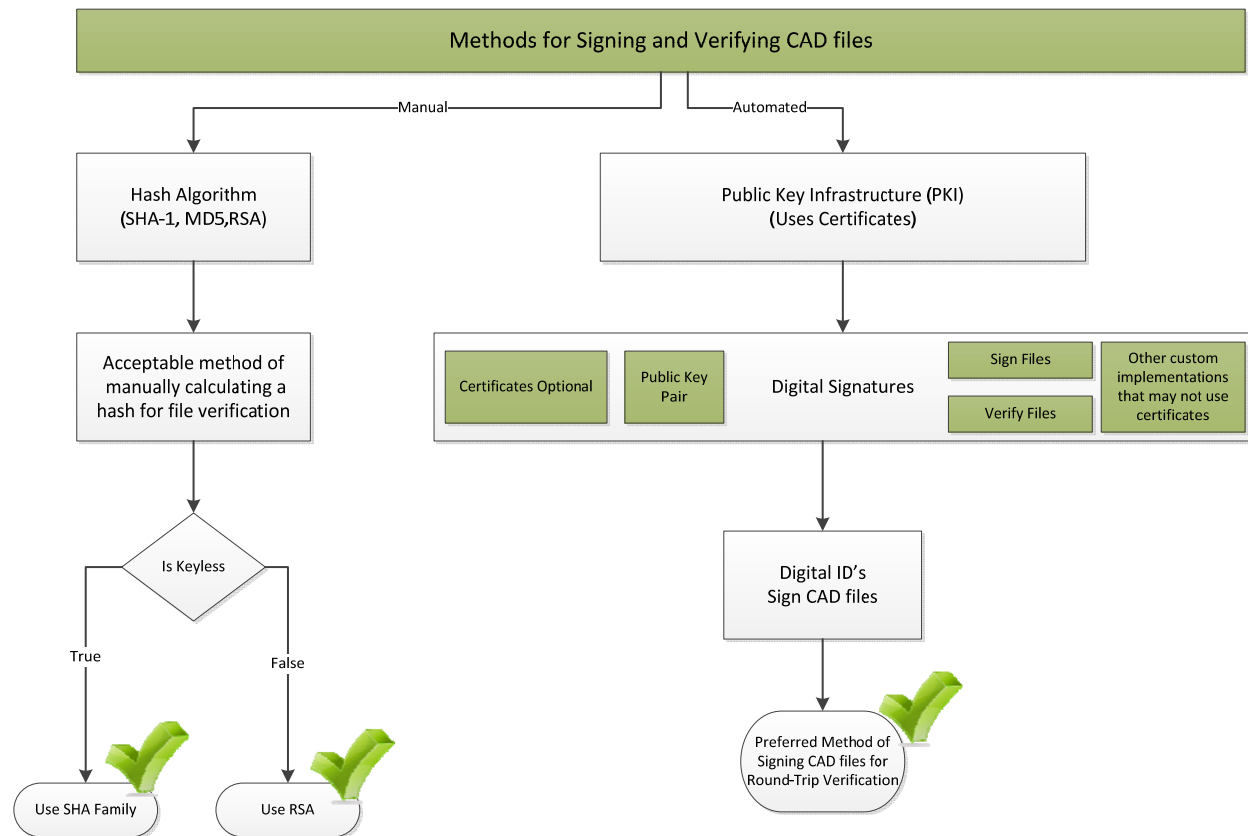


Figure 18 shows two branches (Manual Hash, Automated Digital ID) for signing and verifying CAD files.

7. DIGITALLY SIGN AND SEAL

In this section, the two methods of signing a CAD drawing are discussed below. The two methods for signing files are as follows:

- ❖ **Manual hash.** The manual hash or one-way hash examples are the SHA family or MD5 hashes.
- ❖ **Digital ID.** This specific type of Digital Signature uses certificates by the way of PKI to sign documents using automated software like Bentley® and Autodesk®.

Adobe PDF

Manual Hashing

Before signing any PDF files, it is important to cover two methods of signing any drawing or file. The first method is a straightforward method of using a hash algorithm such as the SHA-1. This is equivalent to using just the first part of the Sha1RSA algorithm as shown in the Figure 6 to generate a message digest. This is very common practice. The limiting factor is the recipient must manually calculate the same one-way hash SHA-1 on their end to compare hash values. There are other variations of algorithms that attempt to verify the content of files that have been used to verify and check for data integrity in various forms over the years:

- ❖ Checksum, Parity bits (Error Checks)
- ❖ Cycle redundancy checks (Error Checks)
- ❖ SHA-1, SHA-2,MD5 (File Integrity)

Users need the required software to be installed on their machine to calculate the hash such as Bitser® (Bitser.org, 2010) or AEC Signature®, a CAD dependent version. If there is a downside to using this type of method, it is finding software that calculates a hash that meets the professional standards with the requirements of a paper trail.

Next, users must have full read access to the file, and the file cannot be opened at the time of the signing since the software will not have access to the entire file. There are exceptions to this rule with bit-locking software that retains less access to the file at one time, which allows the file to be read by other software. As a general practice, the file should always be closed to calculate a hash, unless a CAD package, such as AutoCAD, allows such practices.

Once users calculate the hash value, they have a unique value for the PDF. If it is changed by a client, the hash will be different. (See Checksums Figure 5 for an example output.)

Using Digital ID

The second method of using a Digital Signature for PDFs that support using a Digital ID is included in the Appendix A to sign with a certificate. Make sure the user sends the public key to the recipients or they will not be able to verify the signed drawing or file, as the file will be unreadable due to the security restrictions.

CAD File

Manual Hashing

The hash algorithm method is the same as the hash signing a PDF. Once users get the hash value, they have a unique value for the PDF. If a client changes it, the resulting calculated hash will be different than the original value. The recipient will need to recalculate the hash manually to validate the file.

Using Digital ID


The second method of implementing a digital signature for CAD file support with Digital IDs, users can follow instructions in Appendix B. Make sure users send the public key to the recipients or they will not be able to compute the drawing's authenticity. The file will be readable, but due to the security restrictions of the unidentifiable certificate, the clients do not have confirmation the file has not changed from the sender's computer to their computers. Try this exercise with a colleague!

It is important to understand the difference in security architectures; manually calculated hashes help to determine if a file has changed, but the computed hash must be shown to the end user to compare with hash value on their end. A drawback to this second method is the end user must calculate the hash again on their end to compare values.

The Digital Signature protects the document for the complete round trip. This means the user signed the drawing with their Digital ID and passed his or her public key certificate to the client. The client installed the certificate and opened the drawing. They received notification the file has not changed since the last save and the signer's identity is listed on the certificate. If the end user changes anything with the file, the certificate is rendered invalid. Therefore, if it is emailed back to the user, it will show an invalid drawing. As a result, the user has protection for the complete electronic transmission of the file.

File Delivery

After the CAD file is digitally signed, there needs to be a delivery policy. This means, at a minimum, there should be a paper trail showing the Company Name, Signer's Name and Contact Information, along with the hash value used, if any. A good approach would be to zip (compress files into one package) the user's signed files into a compress zip file, which may be protected with a password. Then, email the zipped contents to the client. The client would need to know the password. The password can be intelligently embedded into a copyright and confidentiality agreement text file attached to the email, meaning the client must read the agreement to get the password to open the zipped file.

 **Note:** Remember to include any public key certificates if a Digital ID was used! A public key certificate is not the sender's entire Digital ID. It is an exported certificate without the password. This allows the recipient to install the certificate by double clicking on the file; otherwise, they would need to know the sender's private password when installing the certificate, which defeats the entire purpose of the certificate architecture.

A free software to password protect with encrypting drawings into a zipped file is Bitser. The software may be downloaded at Bitser.org.



Video tutorial is available on the support site for this topic.

8. STORAGE AND DISTRIBUTION

Local File Server

Storage of CAD files on a local or network file servers are common. Many backup systems are very cost effective and easy to manage. If users decide to save the drawing contents onto a file

server, there should be an access policy in place. Access policies restrict or allow access to a folder or file. The folder should have “read only” rights, meaning once files are placed into the server they cannot be overridden by a newer version of the same file. Since the server is a simple system of file rights management, special care needs to be taken that no mistake overrides the file contents.

By setting the folder to default or “read-only,” the user can copy the file to their local directories but not move the file to override the original content. They would need to rename the file or insert it into a new subfolder.

In the example below, this server architecture places the file into a backend (limited user access) server to have “read-write” access and pushes synchronized files into a group of remote, read-only servers for different offices to use the data. The end users do not have the ability to change any of the files on the read-only server.

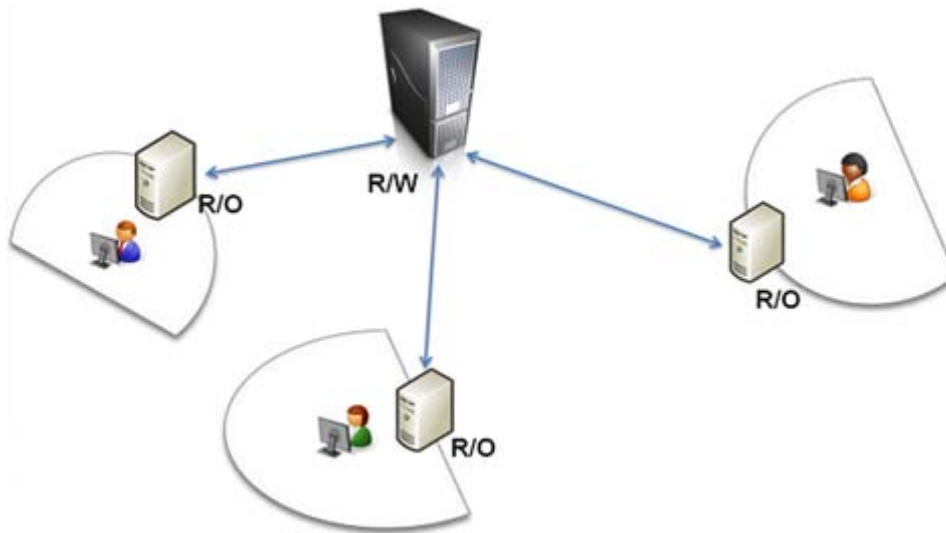


Figure 20 shows the read only server architecture or design.

CAD Middleware Repositories

By centralizing the storage of, conceptually, all engineering data and related documents, users can reduce time organizing files, avoid costly mistakes, and more efficiently release and revise designs. As product designs advance and become more complicated, the middleware products protect designers and engineers from unintentional overwriting of designs.

- ❖ CAD integration
- ❖ Collaborative Design
- ❖ Find Data Quickly
- ❖ Copy and Reuse Data
- ❖ Revision Management
- ❖ Administration and Configuration
- ❖ Share Data to the Cloud
- ❖ Scalability for Multiple Sites

- ❖ Automated Release
- ❖ Exchange Data with Enterprise Business Systems

Middleware products reside on either standalone server machines or on local machines (not recommended). The standard architecture is they exist on an architectural layer between the database layer and the file store layer. They tend to fulfill the limitations by inserting a management system between architectural layers that ordinary file servers cannot perform. As with the Autodesk Vault, it is actually designed to be a standalone system with an included database management system.

The design of the middleware system gives the drafts-person the ability to share and collaborate on multiple parts of the project at the same time. With the following functionality, such as check-out for edit, check-out for read-only, and check-in, gives the drafts-person and CAD Manager finer granularity with permissions on any project. Sewer Design and Drainage Design on Subdivision One can be designed and managed simultaneously by different groups, companies, or agencies.

Policy controls are needed to limit and access to a signed file. Good policy allows for adequate access to materials, while keeping the drawing's integrity. It is probably not a good practice to allow editing by everyone once a file is signed; however, a process called "versioning," an integral part of the middleware system, allows continuous editing of drawings, but at specific times, a snapshot of the drawing is obtained. This would be an excellent place to create a snapshot of the file, after signing with the sender's Digital ID.

In the example middleware server architecture below, servers could be located in the next room, county, or country. The primary objective is 100% access and server uptime. If a hurricane or earthquake hit a region, having servers distributed across time zones will ensure data availability; this approach has a seemingly familiar *cloud* aspect to the architecture.

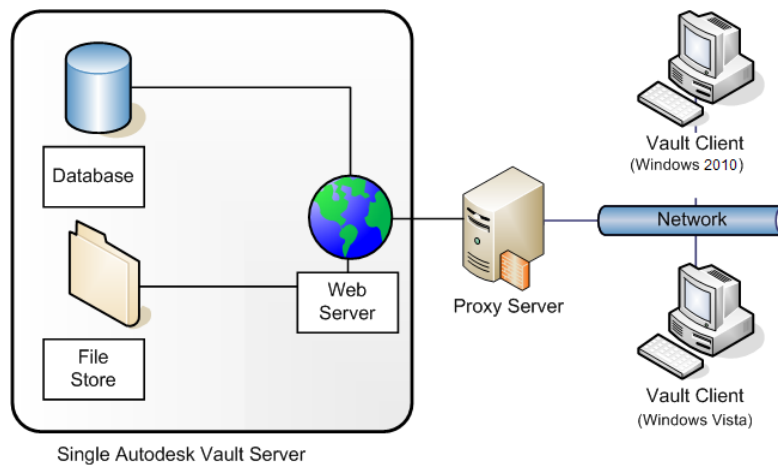


Figure 21 shows the Autodesk Vault typical architecture.

Cloud Solutions

Wikipedia defines Cloud computing as the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software, and computation.

In a cloud computing structure, there is a significant workload shift. Local computers, PCs, Tablets, and Mobile Devices no longer have to do all the heavy lifting when it comes to running large applications. The network of computers that make up the cloud handles them instead. Hardware and software demands on the user's side decline, a term call "Thin Client." An important, limited aspect is the user's device needs to be able to run the cloud computing system's software, which can be as simple as a web browser, and the cloud's network takes care of the rest.

There is a shift that will take a few years to see from an end user's perspective. Currently, most of the CAD software is installed locally. CAD systems are currently available and can be purchased as a service via the Cloud. What does this mean with regards to the cost for maintaining the hardware, software, and connections that are entirely on the host of the cloud service? The end user will connect with a lower powered device. Once the command is sent to draw a line, for example, the shared memory and network resources in the cloud combine and focus to perform that function; thus, this results in entire cloud CAD Systems being maintained by the software providers, which will perform at or better than current response times of a traditional desktop PC.

There's a good chance readers have already used selected forms of cloud computing. If they have an e-mail account with a Web-based e-mail service like Hotmail, Yahoo! Mail or Gmail, readers have had a little experience with cloud computing. Instead of running an e-mail program on their computer, users log in to a Web e-mail account remotely. The software and storage for the user's account doesn't exist on their computer. It is on the service's computer cloud.

In the Figure 22 below, along with traditional server storage, the apps on the user's Tablet (i.e., Kindle, Ipad, and Nexus) are distributed via the cloud. With distributing CAD drawings on the cloud, the sender avoids the typical firewall nightmares where clients do not receive their files. Emailing sending files by FTP, clients' files may never reach their destination due to security restrictions. Now, Autodesk has embedded a system of cloud space for each user to upload and distribute drawings via the cloud. All they do is email the provided link to the intended recipient. This would be an excellent place to have a client download the file after a Digital Signature was attached.

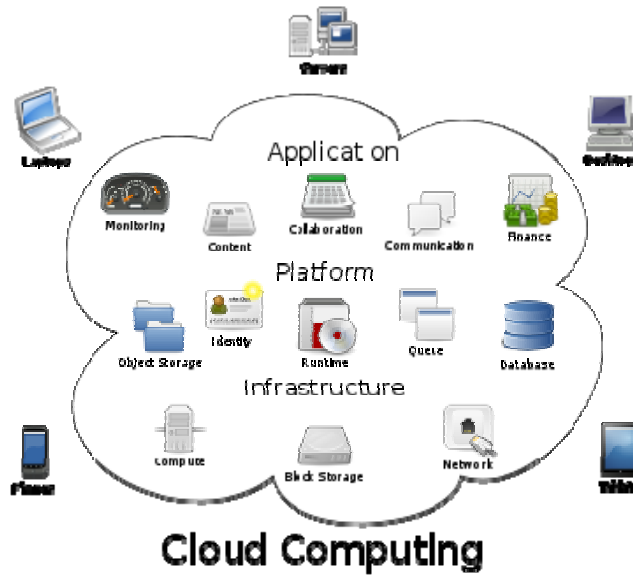


Figure 22 shows the cloud design.

Digital Signature Guidelines

As a last thought on materials and methods pertaining to Digital IDs, is the adherence to the state standards of practice. In Florida, for Surveying and Mapping, the standard of practice is shown in Appendix E. There are a few items under Subsection 3 that are of interest to every reader of this course and are shown below:

(3) An electronic signature is a digital authentication process attached to or logically associated with an electronic document and shall carry the same weight, authority, and effect as an original signature and raised seal. The electronic signature, which can be generated by using either public key infrastructure or signature dynamics technology, must be as follows:

- (a) Unique to the person using it;
- (b) Capable of verification;
- (c) Under the sole control of the person using it;
- (d) Linked to a document in such manner that the electronic signature is invalidated if any data in the document are changed.



Note: *An additional note on these ideas as presented is they are meant as a starting point that will not be ambiguous to the intent and purpose of digitally signed drawings. They are meant to serve as guidelines, and are NOT a standard of modifying certificate settings for professionals across disciplines and regions of the United States.*

The four sub-items listed above are attempting to specify what constitutes a Digital Signature that meets the state requirements. These sub-items start with an ability to uniquely identify the person who signed the document. Remember, the Digital ID is verified by the purchasers email

address for Class 1 Certificate. As a result, a bad example would be if the signer created the Digital ID with a subject name of “My Company Name, Inc.” or “Fred Smith Construction, Inc.,” since there is ambiguity of who is the actual holder of the Digital ID. These above examples may be misconstrued to mean the administrator and secretary along with any other professionals can sign documents for the company. They very well may have the right to perform signatures on behalf of the company, but the intent of sub-item “a” is the responsible professional’s name, including a middle initial, since that may eliminate confusion of same names. A good example would be “John E. Smith III” as the modified subject with an email address of John.Smith@FredSmithConstruction.com.

An idea that may be suitable for most professionals is “John E. Smith, PE 0001123,” as seen on the figure on the right hand side below, which appends the license number to the name. In the certificate fields below, the license number is in the certificate description property fields. The friendly name has the professional’s name instead of *Persona non-validated*, meaning the person was not validated; only the email was validated. The Digital ID purchased has editable properties such as the friendly name and description. These are good places to add the license number, name of the company and its location if it is a satellite office.

Items “b” and “c” describe the method used must be able to be verify if the file has changed. As described earlier, either method of computing hashes manually or using a Digital ID with the built in PKI to validate and sign files is acceptable. Where the difficulty may arise, is establishing sole ownership with the manual hash method, since there is no public or private key. It may not be acceptable method by itself; however, manual hash calculators (Bitser.org) may be used as an additional step to post process files inside a folder then encrypt the folder contents. Finally, email the compressed, encrypted and signed CAD files to the client.

Item “d” begins with the word, “Linked.” This may seem like semantics since both the manual hash and a Digital ID with PKI can verify changes in the drawings, but there are questions the professional must answer. Is the method linked to the file? How is the value linked to the file? The terminology seems to suggest that the only mechanism that meets these requirements is the Digital ID using PKI since the message digest is attached to the file itself.

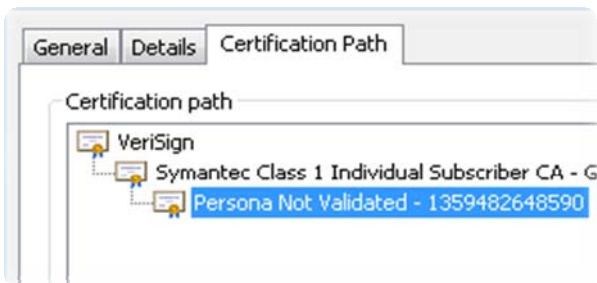


Figure 23 shows original subject of an issued certificate.

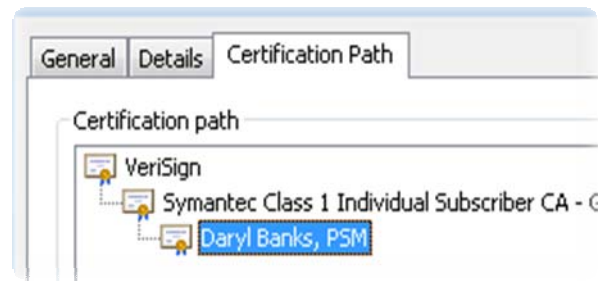


Figure 24 shows revised friendly name and description properties of the certificate.

9. SUMMARY

As CAD Platforms continue to evolve, the electronic transmission and review of drawings are here to stay. Many state professional standards governing these practices have been adopted. This course briefly discussed one state standard for Surveying and Mapping in Florida. Although brief, the point was to show the need for standards when using Digital IDs and the need for a paper trail as a matter of record keeping. A Digital ID is nothing more than a certificate that uses a public key architecture to sign and verify the contents of a file. To understand how a certificate is designed, a few concepts were presented such as Algorithms, Symmetric and Asymmetric Keys, and Encryption. The complex details of these concepts were abstracted to focus on the use more than the design perspective. Figures presented help illustrate the use of Digital IDs and the need to share the signer's public key for verification of the distributed drawing.

Two common methods to verify a drawing depend on whether the built-in mechanism of Digital IDs was implemented or the manual method of calculating the hash value on both ends of the transmission was chosen to verify the drawing's contents were not altered. Although a Digital ID or manual hash was sufficient alone to verify the file, the requirement for the added paper trail turns the two-step command into a multi-step process to prepare and send the drawings.

A final discussion was presented on the storage of the signed drawings. Although a local server storage was sufficient for small organizations, other options exist for 24-hour, real-time access to content via middleware and cloud-based solutions for clients and field personnel.

Hopefully, with this understanding of Digital Signatures, you will pursue more knowledge on this subject and evaluate other software packages to deliver excellent, professional and technically savvy content. Remember, if any of the content is unclear, visit the support site listed earlier for additional examples, materials, and tutorials on these topics and more.

If this course was interesting and you wish to know more about Adobe Portable Document Files (PDF), then purchase the second part of this series on Introduction to Digital IDs with PDFs. The topics will concentrate on implementing a clear strategy when signing and emailing PDF documents including the following list of items:

- ❖ Creating and Using Self-Signed Certificates
- ❖ Creating a customized Digital Signature for PDF
- ❖ Encrypting with a Self-Signed Digital ID
- ❖ Encrypting and Signing with a 3rd Party Digital ID
- ❖ Understanding how PDF converts to CAD formats
- ❖ How to prevent the PDF from being reverse engineered
- ❖ Create a roaming policy for tracking client usage

APPENDIX A

Adobe Portable File

Certificate Security

The following items are directly from the help file located in the Adobe Portable File Software. Included are the relevant items relating to Digital IDs. Furthermore, to sign with a Digital Certificate with PDFs, you will need the Professional Version of Adobe Acrobat to sign with a certificate.

Once you obtain the software, a certificate obtained by purchasing a Digital ID from VeriSign may be used to encrypt documents and to verify a Digital Signature of an output CAD file into PDF. A Digital Signature assures recipients that the document came from you. Encryption ensures that only the intended recipient can view the contents. A certificate stores the public key component of a Digital ID. When you secure a PDF using a certificate, you specify the recipients and define the file access level for each recipient or group. For example, you can allow one client to sign with their certificate and fill forms and another to edit text or remove pages. You can choose certificates from your list of trusted identities, files on disk or the Windows certificate store, which is native to Windows Operating Systems. Always include your certificate in the recipient list so that you can open the document later.

Adobe makes a disclaimer about encrypting documents using certificates from third party Digital IDs. If the certificate is lost or stolen, the issuing authority can replace it. If a self-signed Digital ID is deleted, all PDFs that were encrypted using the certificate from that ID are inaccessible forever. So, be cautious using self-signed certificates. You do not want to be unable to open the document three years down the road.

Encrypt a PDF or PDF Portfolio with a Certificate

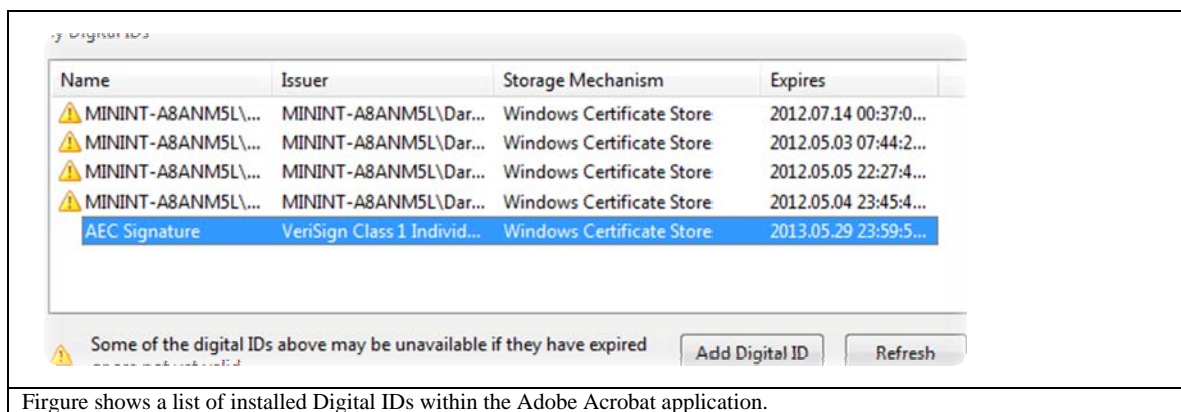
To encrypt many PDFs, use “Action Wizard” (File > Action Wizard) to apply a predefined sequence. Alternatively, select under Security Method Certificate Security. You can also save your certificate settings as a security policy and reuse it to encrypt PDFs.



1. For a single PDF or a component PDF in a PDF Portfolio, open the PDF. For a PDF Portfolio, open the PDF Portfolio and choose View > Portfolio > Cover Sheet.

2. Choose “Tools > Protection > Encrypt > Encrypt with Certificate.” If you don’t see the Protection panel, see the instructions for adding panels at Task Panes.
3. At the prompt, click “Yes.”
4. In the Certificate Security Settings dialog box, select the document components to encrypt.
5. From the Encryption Algorithm menu, choose a rate of encryption, and then click “Next.”
6. The encryption algorithm and key size are version-specific. Recipients must have the corresponding version (or later) of Acrobat or Reader to decrypt and read the document.
7. If you select 128-bit AES, recipients must have Acrobat 7 or later or Reader 7 or later to open the document.
8. If you select 256-bit AES, Adobe Acrobat 9 or later or Adobe Reader 9 or later is required to open the document.
9. Create a recipient list for the encrypted PDF. Always include your own certificate in the recipient list so that you are able to open the document later.
10. Click “Search” to locate identities in a directory server or in your list of trusted identities.
11. Click “Browse” to locate the file that contains certificates of trusted identities.
12. To set printing and editing restrictions for the document, select “Recipients” from the list, and then click “Permissions.”
13. Click “Next” to review your settings, and then click “Finish.”

When a recipient opens the PDF or PDF Portfolio, the security settings you specified for that person are used.



Change Encryption Settings

1. Do one of the following:
 - o For a single PDF or a component PDF in a PDF Portfolio, open the PDF.
 - o For a PDF Portfolio, open the PDF Portfolio and choose “View > Portfolio > Cover Sheet.”
2. Select “Tools > Protection > More Protection > Security Properties.”
3. Click “Change Settings.”

4. Do any of the following then click “Next.”
 - To encrypt different document components, select that option.
 - To change the encryption algorithm, choose it from the menu.
5. Do any of the following:
 - To check a trusted identity, select the recipient, and then click “Details.”
 - To remove recipients, select one or more recipients, and then click “Remove.” Do not remove your own certificate unless you do not want access to the file using that certificate.
 - To change permissions of recipients, select one or more recipients, and then click “Permissions.”
6. Click “Next” then click “Finish.” Click “OK” to close the Document Properties dialog box, and save the document to apply your changes.


Remove Encryption Settings

1. Do one of the following:
 - For a single PDF or a component PDF in a PDF Portfolio, open the PDF.
 - For a PDF Portfolio, open the PDF Portfolio and choose “View > Portfolio > Cover Sheet.”
2. Select “Tools > Protection > Encrypt > Remove.”
3. If prompted, type the permissions password. If you don’t know the permissions password, contact the author of the PDF.

Sharing Certificates with Others

Businesses that use certificates for secure workflows often store certificates on a directory server that participants can search to expand their list of trusted identities.

When you receive a certificate from someone, you can add it to your list of trusted identities. You can set your trust settings to trust all Digital Signatures and certified documents created with a specific certificate. You can also import certificates from a certificate store, such as the Windows certificate store. A certificate store often contains numerous certificates issued by different certification authorities.

 **Note:** *Third-party security providers usually validate identities by using proprietary methods or, they integrate their validation methods with Acrobat. If you use a third-party security provider, see the documentation for the third-party provider.*

Get Certificates from Other Users

Certificates that you receive from others are stored in a list of trusted identities. This list resembles an address book and enables you to validate the signatures of these users on any documents you receive from them. Once you receive a certificate from a trusted client usually all the steps needed to install is to double click the file.

Request a Certificate from Another User

1. Do one of the following:
 - a. In Acrobat, choose “Tools > Sign & Certify > More Sign & Certify > Manage Trusted Identities.”
 - b. In Reader, choose “Edit > Protection > Manage Trusted Identities.”



Note: *If you don't see the “Sign & Certify” or “Protection” panel, see the instructions for adding panels at Task Panes.*

2. Click “Request Contact.”
3. Type your name, e-mail address, and contact information.
4. To allow other users to add your certificate to their list of trusted identities, select “Include My Certificates.”
5. Select either “Email Request” or “Save Request As A File.” Then click “Next.”
6. Select the Digital ID file to use, and then click “Select.”
7. Do one of the following:
 - a. If the “Compose Email” dialog box appears, type the e-mail address of the person you're requesting a certificate from, and click “Email.” Send the e-mail message that appears, with the attached certificate, in the default e-mail application.
 - b. If the “Export Data As” dialog box appears, specify a name and location for the file, click “Save,” and then click “OK.”

Add a Certificate from e-Mail

When a contact sends a certificate to you in e-mail, it is displayed as an import/export methodology file attachment.

1. Double-click the e-mail attachment, and then click “Set Contact Trust” in the dialog box that appears.
2. On the “Trust” tab of the “Import Contact Settings” dialog box, select “Trust Options.”
 - Select “Use This Certificate as a Trusted Root” only if it is required to validate a Digital Signature. Once you make a certificate a trust anchor, you prevent revocation checking on it (or any certificate in the chain).
 - To allow actions that can be a security risk, click “Certified Documents” then select the options you want to allow:
 - a. Dynamic Content.
 - b. Includes FLV and SWF files as well as external links.
 - c. Embedded High Privilege JavaScript.
 - d. Trusts embedded scripts.
 - e. Privileged System Operations.
 - f. Includes networking, printing, and file access.
3. Click “OK” to view the import details, and then click “OK” again.

Add a Certificate from a Digital Signature in a PDF

You can safely add a certificate to your trusted identities from a signed PDF by first verifying the thumbprint with the originator or the certificate.

1. Open the PDF containing the signature.
2. Open the signature panel, and select the signature in the “Signatures” panel.
3. On the Options menu, click “Show Signature Properties,” and then click “Show Certificate.”
4. If the certificate is self-signed, contact the originator of the certificate to confirm that the thumbprint values on the “Details” tab are correct. Trust the certificate only if the values match the values of the originator.
5. Click the “Trust” tab, click “Add to Trusted Identities,” and click “OK.”
6. In the “Import Contact Settings” dialog box, specify trust options, and click “OK.”

Import Certificates using the Windows Certificate Wizard (Windows only)

If you use the Windows certificate store to organize your certificates, you can import certificates using a wizard in Windows Explorer. To import certificates, identify the file that contains the certificates, and determine the file location.

1. In Windows Explorer, right-click the “Certificate” file and choose “Install PFX.”
2. Follow the onscreen instructions to add the certificate to the Windows certificate store.
3. If you are prompted to validate the certificate before installing it, note the MD5 digest and SHA1 digest values (thumbprint). Contact the originator of the certificate to confirm that the values are correct before you trust the certificate. Click “OK.”

Associate a Certificate with a Contact

If you have a contact that is not associated with a certificate, or you want to change the certificate associated with a contact, follow these steps. A contact must have at least one valid certificate to exchange encrypted PDFs.

1. Do one of the following:
 - In Acrobat, choose “Tools > Sign & Certify > More Sign & Certify > Manage Trusted Identities.”
 - In Reader, choose “Edit > Protection > Manage Trusted Identities.”



Note: If you don't see the “Sign & Certify” or “Protection” panel, see the instructions for adding panels at Task Panes.

2. Select the contact, and click “Details.”
3. Click “Associate Certificate.”
4. Select a certificate, and click “OK.” Click “OK” again.

Verify Information on a Certificate

The Certificate Viewer dialog box provides user attributes and other information about a certificate. When others import your certificate, they often want to check your “thumbprint” information against the information they receive with the certificate. (The thumbprint refers to the MD5 digest and SHA1 digest values.) You can check certificate information for your Digital ID files or the ID files that you import.

The “Certificate Viewer” dialog box provides the following information:

- Certificate validation period
- Intended use of the certificate
- Certificate data, such as the serial number and public key method

You can also check if the certificate authority has revoked the certificate. Certificates are typically revoked when an employee leaves the company or when security is compromised in some way.

Verify Your Own Certificate

1. Do one of the following:
 - In Acrobat, choose “Tools > Protection > More Protection > Security Settings.”
 - In Reader, choose “Edit > Protection > Security Settings.”



Note: *If you do not see the “Protection” panel, see the instructions for adding panels at Task Panes.*

2. Select your Digital ID, and then click “Certificate Details.”

Verify information on the Certificate of a Contact

1. Do one of the following:
 - In Acrobat, choose “Tools > Sign & Certify > More Sign & Certify > Manage Trusted Identities.”
 - In Reader, choose “Edit > Protection > Manage Trusted Identities.”

Note: If you don’t see the “Sign & Certify” or “Protection” panel, see the instructions for adding panels at Task Panes.

2. Select the contact, and click “Details.”
3. Select the certificate name, and click “Show Certificate.”

Delete a Certificate from Trusted Identities

1. Do one of the following:
 - In Acrobat, choose “Tools > Sign & Certify > More Sign & Certify > Manage Trusted Identities.”

- In Reader, choose “Edit > Protection > Manage Trusted Identities.”



Note: *If you do not see the “Sign & Certify” or “Protection” panel, see the instructions for adding panels at Task Panes.*

2. Choose “Certificates” from the “Display” menu.
3. Select the certificate, and click “Delete.”

Create a Self-Signed Digital ID

Sensitive transactions between businesses generally require an ID from a certificate authority rather than a self-signed one.

1. Do one of the following:
 - In Acrobat, choose “Tools > Sign & Certify > More Sign & Certify > Security Settings.”
 - In Reader, choose “Edit > Protection > Security Settings.”



Note: *If you do not see the “Sign & Certify” or “Protection” panel, see the instructions for adding panels at Task Panes.*

2. Select “Digital IDs” on the left, and then click the “Add ID” button.
3. Select the option “A New Digital ID I Want To Create Now” then click “Next.”
4. Specify where to store the Digital ID and click “Next.”

New PKCS#12 Digital ID File

Stores the Digital ID information in a file that has the extension .pfx in Windows and .p12 in Mac OS. You can use the files interchangeably between operating systems. If you move a file from one operating system to another, Acrobat still recognizes it.

Windows Certificate Store (Windows only)

Stores the Digital ID to a common location from where other Windows applications can also retrieve it.

5. Type a name, email address, and other personal information for your Digital ID. When you certify or sign a document, the name appears in the “Signatures” panel and in the “Signature” field.
6. (Optional) To use Unicode values for extended characters, select “Enable Unicode Support” then specify “Unicode” values in the appropriate boxes.
7. Choose an option from the “Key Algorithm” menu. The 2048-bit RSA option offers more security than 1024-bit RSA, but 1024-bit RSA is more universally compatible.
8. From the “Use Digital ID For” menu, choose whether you want to use the Digital ID for signatures, data encryption, or both.

9. Type a password for the Digital ID file. For each keystroke, the password strength meter evaluates your password and indicates the password strength using color patterns. Reconfirm your password.

You can export and send your certificate file to contacts that can use it to validate your signature.



Note: *Make a backup copy of your Digital ID file. If your Digital ID file is lost or corrupted, or if you forget your password, you cannot use that profile to add signatures.*

Register a Digital ID

To use your Digital ID, register your ID with Acrobat or Reader.

1. Do one of the following:
 - o In Acrobat, choose “Tools > Protection > More Protection > Security Settings.”
 - o In Reader, choose “Edit > Protection > Security Settings.”



Note: *If you do not see the “Protection” panel, see the instructions for adding panels at Task Panes.*

2. Select “Digital IDs” on the left.
3. Click the “Add ID” button.
4. Select “My Existing Digital ID From” and choose one of the following options:

A File

Select this option if you obtained a Digital ID as an electronic file. Follow the prompts to select the Digital ID file, type your password, and add the Digital ID to the list.

A Roaming Digital ID Stored On a Server

Select this option to use a Digital ID that is stored on a signing server. When prompted, type the server name and URL where the roaming ID is located.

A Device Connected to This Computer

Select this option if you have a security token or hardware token connected to your computer.

5. Click “Next” follow the onscreen instructions to register your Digital ID.

Specify the Default Digital ID

To avoid being prompted to select a Digital ID each time your sign or certify a PDF, you can select a default Digital ID.

1. Do one of the following:
 - In Acrobat, choose “Tools > Protection > More Protection > Security Settings.”
 - In Reader, choose “Edit > Protection > Security Settings.”



Note: *If you do not see the “Protection” panel, see the instructions for adding panels at Task Panes.*

2. Click “Digital IDs” on the left, and then select the Digital ID you want to use as the default.
3. Click the “Usage Options” button, and choose a task for which you want the Digital ID as the default. To specify the Digital ID as the default for two tasks, click the “Usage Options” button again and select a second option.

A check mark appears next to selected options. If you select only the signing option, the “Sign” icon appears next to the Digital ID. If you select only the encryption option, the “Lock” icon appears. If you select only the certifying option, or if you select the signing and certifying options, the “Blue Ribbon” icon appears.

To clear a default Digital ID, repeat these steps, and deselect the usage options you selected.

Change the Password and Timeout for a Digital ID

Passwords and timeouts can be set for PKCS #12 IDs. If the PKCS #12 ID contains multiple IDs, configure the password and timeout at the file level.



Note: *Self-signed Digital IDs expire in five years. After the expiration date, you can use the ID to open, but not sign or encrypt, a document.*

1. Do one of the following:
 - In Acrobat, choose “Tools > Protection > More Protection > Security Settings.”
 - In Reader, choose “Edit > Protection > Security Settings.”



Note: *If you do not see the Protection panel, see the instructions for adding panels at Task Panes.*

2. Expand “Digital IDs” on the left, select “Digital ID Files,” and then select a “Digital ID” on the right.
3. Click the “Change Password” button. Type the old password and a new password. For each keystroke, the password strength meter evaluates your password and indicates the password strength using color patterns. Confirm the new password, and then click “OK.”
4. With the ID still selected, click the “Password Timeout” button.
5. Type the password, and click “OK.”

Be sure to back up your password in a secure place. If you lose your password, either create a new self-signed Digital ID and delete the old one, or purchase one from a third-party provider.

Delete your Digital ID

When you delete a Digital ID in Acrobat, you delete the actual PKCS #12 file that contains both the private key and the certificate. Before you delete your Digital ID, ensure that it is not in use by other programs or required by any documents for decrypting.



Note: *You can only delete self-signed Digital IDs that you created in Acrobat. A Digital ID obtained from another provider cannot be deleted.*

1. Do one of the following:
 - In Acrobat, choose “Tools > Protection > More Protection > Security Settings.”
 - In Reader, choose “Edit > Protection > Security Settings.”



Note: *If you do not see the “Protection” panel, see the instructions for adding panels at Task Panes.*

2. Select “Digital IDs” on the left, and then select the “Digital ID” to remove.
3. Click “Remove ID” then click “OK.”

Protecting Digital IDs

By protecting your Digital IDs, you can prevent unauthorized use of your private keys for signing or decrypting confidential documents. Ensure that you have a procedure in place in the event your Digital ID is lost or stolen.

How to Protect Your Digital IDs

When private keys are stored on hardware tokens, smart cards, and other hardware devices that are password- or PIN-protected, use a strong password or PIN. Never divulge your password to others. If you must write down your password and store it in a secure location. Contact your system administrator for guidelines on choosing a strong password. Keep your password strong by following these rules:

- Use eight or more characters.
- Mix uppercase and lowercase letters with numbers and special characters.
- Choose a password that is difficult to guess or hack, but that you can remember without having to write it down.
- Do not use a correctly spelled word in any language, as they are subject to “dictionary attacks” that can crack these passwords in minutes.
- Change your password on a regular basis.
- Contact your system administrator for guidelines on choosing a strong password.

To protect private keys stored in P12/PFX files, use a strong password and set your password timeout options appropriately. If using a P12 file to store private keys that you use for signing, use the default setting for password timeout option. This setting ensures that your password is always required. If you are using your P12 file to store private keys that are used to decrypt documents, make a backup copy of your private key or P12 file. You can use the backed up private key of P12 file to open encrypted documents if you lose your keys.

The mechanisms used to protect private keys stored in the Windows certificate store vary depending on the company that has provided the storage. Contact the provider to determine how to back up and protect these keys from unauthorized access. In general, use the strongest authentication mechanism available and create a strong password or PIN when possible.

What to do if a Digital ID is Lost or Stolen

If your Digital ID was issued by a certificate authority, immediately notify the certificate authority and request the revocation of your certificate. In addition, you should not use your private key.

If your Digital ID was self-issued, destroy the private key and notify anyone to whom you sent the corresponding public key (certificate).

Smart Cards and Hardware Tokens

A “smart card” looks like a credit card and stores your Digital ID on an embedded microprocessor chip. Use the Digital ID on a smart card to sign and decrypt documents on computers that can be connected to a smart card reader. Some smart card readers include a keypad for typing a Personal Identification Number (PIN).


Similarly, a “security hardware token” is a small, keychain-sized device that you can use to store Digital IDs and authentication data. You can access your Digital ID by connecting the token to a USB port on your computer or mobile device.

If you store your Digital ID on a smart card or hardware token, connect it to your device to use it for signing documents.

APPENDIX B

Certificates with AutoCAD

Personally Sign Drawings

 *Note: This topic was taken from the help files in the AutoCAD 2013 version. Information has been removed for the sake of conciseness. To use certificates or password protect your CAD file, use the command “securityoptions” to bring up the security settings for the drawing. Specific videos are available on aecsignature.com.*

When you attach a Digital Signature to a file, anyone who views the file is notified if modifications were made after you signed it. Modifications invalidate a Digital Signature.

Obtain a Digital ID

To attach a Digital Signature to a file, you must have a Digital ID (certificate), which is issued by a certificate authority. A Digital ID identifies either an individual or an organization.

Attach a Signature to a Single File

When you attach a Digital Signature to a file, you are helping to ensure that recipients of the file are notified of modifications.

Attach Digital Signatures to Multiple Files

When you attach a Digital Signature to a set of files, you are helping to ensure that anyone viewing the files knows about any changes that are made to the files after you signed them.

Add a Comment and Time Stamp

You can add a comment and time stamp to a Digital Signature.

Digitally Sign an Encrypted Drawing

You can attach a password and a Digital Signature to a drawing file. The password must be attached first.

To attach a Digital Signature to a file, you must have a Digital ID (certificate), which is issued by a certificate authority. A Digital ID identifies either an individual or an organization.

A Digital ID contains a name, serial number, expiration date, and other information that certifies the Digital Signature. From a certificate authority, you can obtain Digital IDs with a security level of Low, Medium, or High.

Low. Use a Low security level if you want to sign multiple files quickly. A Digital Signature is automatically attached to all file types that are valid for a Digital Signature.

Medium. Use a Medium security level if you want to be informed when an application is trying to create a signature with your Digital ID. You are notified each time a Digital Signature containing your Digital ID is attached to a file.

High. Use a High security level if the signature is very sensitive and you do not want your computer to be misused in the signing of a file. You are prompted for a password each time you sign a file.

You can set an option to automatically attach your signature to a drawing whenever you save it. You also can sign many drawings at once, in a batch, and you can sign transmittal packages of drawings.

When you attach a Digital Signature to a file, you are helping to ensure that recipients of the file are notified of modifications.

You can set an option for a signature to be attached after you save a file. A single Digital Signature can be attached per file.

If you plan to protect a set of drawings with a password and a signature, first add the passwords then sign the files in a batch. Adding a password to a signed drawing, or otherwise changing a signed drawing, invalidates the signature.

You can attach a password and a Digital Signature to a drawing file. The password must be attached first.

Modifications to files, including the adding of passwords, invalidate their Digital Signatures.

Import a Certificate

To request the root certificate, contact the organization or individual who attached a Digital Signature to the current file.

When you receive the root certificate, click “Start” menu (Windows) “Settings ► Control Panel ► Internet Options.”

1. In the “Internet Properties” dialog box, Content tab, click “Certificates.”
2. In the “Certificate Manager,” click Import to run the “Certificate Manager Import Wizard.”
3. Follow the on-screen instructions to obtain a root certificate for the current file.

Overview of Drawings with Digital Signatures

A Digital Signature identifies an individual or an organization through a Digital ID (certificate), and enables you to validate the file. Validating a file is especially important when you are working on collaborative projects or transmitting files over the Internet. You can validate a Digital Signature in either the program or Windows Explorer.

Using the Digital Signatures feature, you can obtain the following information about signed files:

- ❖ Whether the file was changed since signed.
- ❖ Whether the signers are who they claim to be.

- ❖ Whether the signers can be traced (thus preventing forgery).

A Digital Signature is determined to be invalid for the following reasons:

- ❖ The file was corrupted when the Digital Signature was attached.
- ❖ The file was corrupted in transit.
- ❖ The Digital ID is no longer valid.

If you want to retain valid Digital Signature status, do not add a password or otherwise modify or save a file that already has a Digital Signature attached. Make modifications, such as adding passwords, before signing the file. Signature information does not remain intact if you modify, save, or export drawing data.

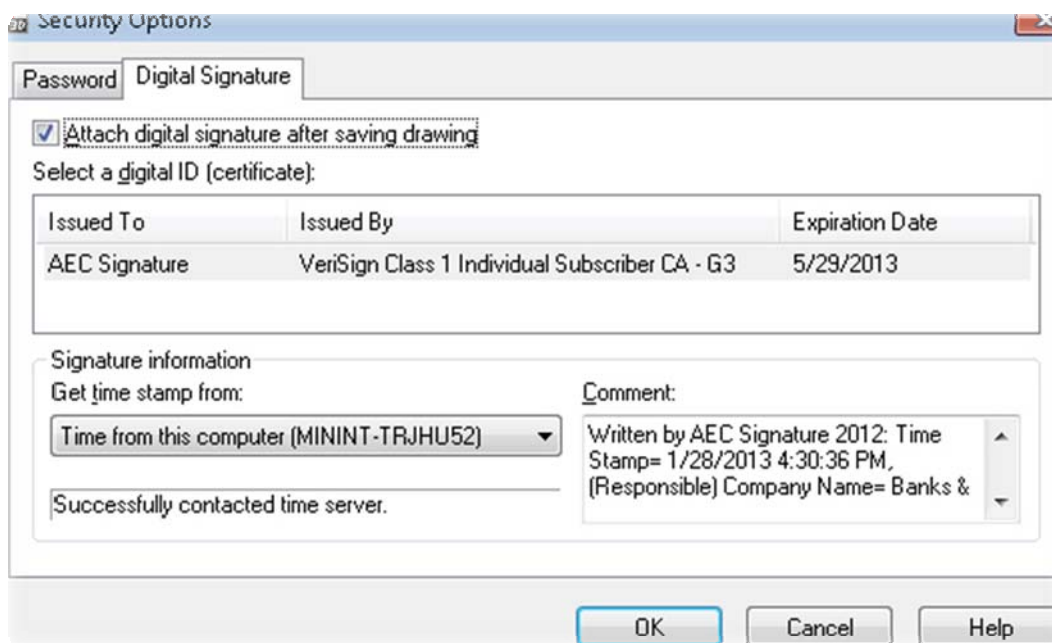


Figure 1 shows the Digital ID setting in AutoCAD invoked by the *securityoptions* command.

APPENDIX C

Content of a Certificate

Please refer to the Glossary for the definitions.

Attribute	Values
Signature Algorithm	sha1RSA
Signature Hash Algorithm	sha1
Issuer	CN = VeriSign Class 1 Individual Subscriber CA - G3 OU = Persona Not Validated OU = Terms of use at https://www.verisign.com/rpa (c)09 OU = VeriSign Trust Network O = VeriSign, Inc. C = US
Valid From	Thursday, May 24, 2012 7:00:00 PM
Valid To	Wednesday, May 29, 2013 6:59:59 PM
Subject	E = help@banksandbanksconsulting.com CN = AEC Signature OU = Digital ID Class 1 - Microsoft Full Service OU = Persona Not Validated OU = www.verisign.com/repository/RPA Incorp. by Ref.,LIAB.LTD(c)98 OU = VeriSign Trust Network O = VeriSign, Inc.
Public Key	30 81 89 02 81 81 00 b9 62 dc 9c 0e 34 3f ee d9 2b a5 98 1b 6a 37 2c 79 16 f9 74 2d e9 00 76 ce 9c 00 62 7c b5 b4 54 d0 a8 58 3d e8 bc 68 ed 00 ec 46 51 4d e2 13 66 04 98 19 b0 16 13 15 1c 06 9e 58 ae 3e d0 2d c8 df 42 12 c1 9a aa 82 dd 58 e5 22 2b 78 6f 45 9a 47 99 99 56 cd 10 80 90 a2 08 67 2e bc 2a b5 27 e9 9a 9c ef 38 ee 5b fc d2 5a 3e fc 10 ce e6 fc 30 8b 11 c6 b8 0c df 37 8d 52 9b 6e e8 1e 42 f1 02 03 01 00 01

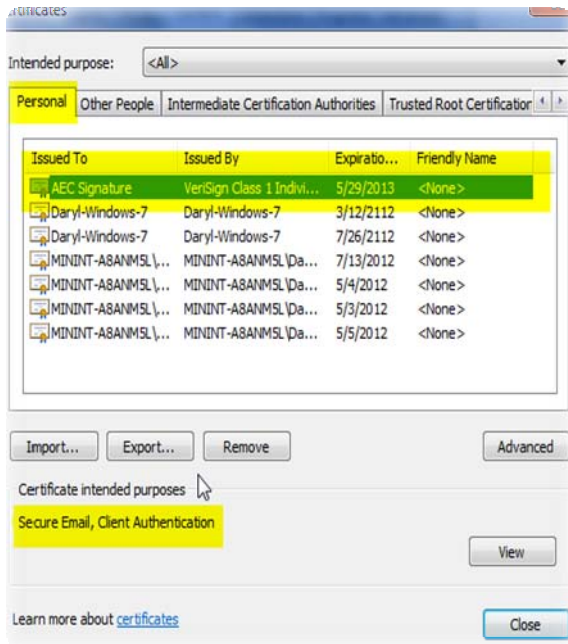


Figure shows an installed certificate used for signing CAD files and sending secure email.

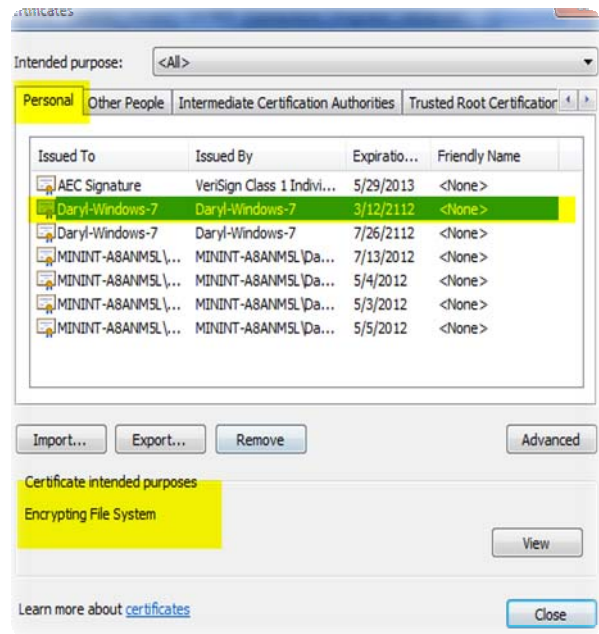


Figure shows an installed certificate used for file encryption only.

APPENDIX D

Example Hard Copy Output

<p>ELECTRONIC SIGNATURE FILE VERSION 2012</p> <p>Private Key File ***NOT FOR DISTRIBUTION***</p> <p>Date Created: 1/15/2013</p> <p>Company Name: Place Company Name Here. Business Information: Geospatial Company, Surveying and Mapping, Software Development</p> <hr/> <p>Profession: Surveying and Mapping</p> <hr/> <p>Individual Name: John Surveyor , License Number: PSM XXXXXXXX Address: XXXX Main Street Florida XXXXX</p> <p>Signer Validation Codes:</p> <p>MD5 = xMpCOKC5I4INzFCab3WEmw== SHA1 = NWoZK3kTsexUV00Ywo1G5jlUKKs= COMPUTER ID = 6A5FEBFD03E17D4FBFF MAC ID = B8:AC:6F:42:C1:7D 00:50:56:C0:00:01 00:50:56:C0:00:08</p> <hr/> <p>Original File Information</p> <hr/> <p>***KEEP THIS FILE FOR YOUR RECORDS***</p> <p>Drawing Name: C:\Civil 3D Projects\JOBS\16175\Drawings\AECSignatureV12\20130115_101529AM\16175.dwg</p> <p>***AUTOGENERATED DWG FILE FOR DISTRIBUTION*** *** DISTRIBUTE YOUR PUBLIC KEY ONLY ***</p> <hr/> <p>File Electronic Signature Code: 85627968557</p> <hr/> <p>Generated File Output: C:\Civil 3D Projects\JOBS\16175\Drawings\AECSignatureV12\20130115_101529AM\16175_Private_Key_File.txt</p> <p>Generated DWG File: C:\Civil 3D Projects\JOBS\16175\Drawings\AECSignatureV12\20130115_101529AM\16175_Private_Key_File_Signed.dwg</p> <p>User Defined Encryption (SHA1): 36eee26ad1ea9d3ba64d4ea08fae867686ff664c</p> <p>Printed Signature, Sign Below:</p> <hr/> <p>Signer's Name Here , License Number: XXXXXXX</p>

APPENDIX E

Florida Digital Signature Standards for Surveying and Mapping

5J-17.062 Procedures for Signing and Sealing Electronically Transmitted Plans, Specifications, Reports or Other Documents. (BPR, 2010).

(1) Information stored in electronic files representing plans, specifications, plats, reports, or other documents that must be sealed under the provisions of Chapter 472, F.S., shall be signed, dated, and sealed by the professional surveyor and mapper in responsible charge.

(2) A license holder may use a computer generated representation of his or her seal on electronically conveyed work; however,

The final hard copy documents of such surveying or mapping work must contain an original signature and raised seal of the license holder and date or the documents must be accompanied by an electronic signature as described in this section. A scanned image of an original signature shall not be used in lieu of an original signature and raised seal or electronic signature. Surveying or mapping work that contains a computer generated seal shall be accompanied by the following text or similar wording: “The seal appearing on this document was authorized by [Example: Leslie H. Doe, P.E. 0112 on (date)]” unless accompanied by an electronic signature as described in this section.

(3) An electronic signature is a digital authentication process attached to or logically associated with an electronic document and

shall carry the same weight, authority, and effect as an original signature and raised seal. The electronic signature, which can be generated by using either public key infrastructure or signature dynamics technology, must be as follows:

- (a) Unique to the person using it;
- (b) Capable of verification;
- (c) Under the sole control of the person using it;

(d) Linked to a document in such manner that the electronic signature is invalidated if any data in the document are changed.

(4) Alternatively, electronic files may be signed and sealed by creating a “signature” file that contains the surveyor and mapper’s name and PSM number, a brief overall description of the surveying and mapping documents, and a list of the electronic files to be sealed. Each file in the list shall be identified by its file name utilizing relative Uniform Resource Locators (URL) syntax described in the Internet Architecture Board’s Request for Comments (RFC) 1738, December 1994, which is hereby adopted and incorporated by reference by the Board and can be obtained from the Internet Website: <ftp://ftp.isi.edu/in-notes/rfc1738.txt>. Each file shall have an authentication code defined as an SHA-1 message digest described in Federal Information Processing Standard Publication 180-1 “Secure Hash Standard,” 1995 April 17, which is hereby adopted and incorporated by reference by the Board and can be obtained from the Internet

Website: <http://www.itl.nist.gov/fipspubs/fip180-1.htm>. A report shall be created that contains the surveyor and mapper's name and PSM number, a brief overall description of the surveyor and mapper documents in question and the authentication code of the signature file. This report shall be printed and manually signed, dated, and sealed by the professional surveyor and mapper in responsible charge. The signature file is defined as sealed if its authentication code matches the authentication code on the printed, manually signed, dated and sealed report. Each electronic file listed in a sealed signature file is defined as sealed if the listed authentication code matches the file's computed authentication code.

APPENDIX F

Glossary

All definitions contained herein have been referenced from Wikipedia Encyclopedia.

AES Algorithm

AES is based on a design principle known as a substitution-permutation network, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification *per se* is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

Algorithm

In mathematics and computer science, an algorithm is a step-by-step procedure for calculations. Algorithms are used for calculation, data processing, and automated reasoning.

Architectural Layer

The following four layers are the most common layers in a logical multilayered architecture for an information system with an object-oriented design:

1. Presentation Layer (a.k.a. UI Layer, View Layer, Presentation Tier.)
2. Application Layer (Service Layer or GRASP Controller Layer)
3. Business Layer (a.k.a Business Logic Layer (BLL))
4. Infrastructure Layer (Data access layer, Persistence layer, Logging, Networking, and other Services, which are required to support a particular Business Layer.)

Sometimes there is no explicit distinction between the Business Layer and the Application Layer, e.g., the Application Layer is considered as being a part of the Business Layer. On the other hand, it is also possible to even further divide the Application/Business Layers into more layers. For example, if the Model View Presenter pattern is used, then you can consider the Presenter Layer as being a layer between the User Interface Layer and the Application Layer.

Byte

A byte is a unit of digital information in computing and telecommunications that most commonly consists of eight bits. Historically, a byte was the number of bits used to encode a single character of text in a computer and for this reason it is the basic addressable element in many computer architectures. The size of the byte has historically been hardware dependent and no definitive standards existed that mandated the size. The *de facto* standard of eight bits is a convenient power of two permitting the values 0 through 255 for one byte. With ISO/IEC 80000-13, this common meaning was codified in a formal standard. Many types of applications use variables representable in eight or fewer bits, and processor designers optimize for this common usage. The popularity of major commercial computing architectures have aided in the ubiquitous acceptance of the 8-bit size.

The term octet was defined to explicitly denote a sequence of 8 bits because of the ambiguity associated at the time with the term byte.

CAD

Computer Aided Drafting

CADD

Computer Aided Drafting and Design

Cloud Solutions or Cloud Computing

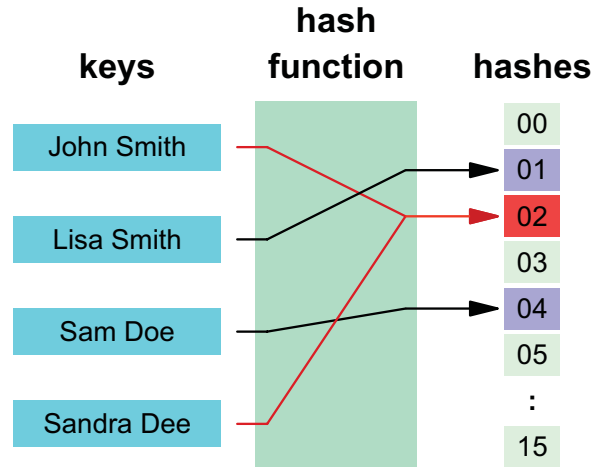
Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation.

In the business model using software as a service, users are provided access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications. “Software as a Service” (SaaS) is sometimes referred to as “on-demand software” and is usually priced on a pay-per-use basis. SaaS providers generally price applications using a subscription fee. By reducing costs associated with maintaining hardware and software locally, SaaS has benefits over traditional desktop applications that require additional employees to manage.

Collision

Some hash functions may map two or more keys to the same hash value, causing a collision. Such hash functions try to map the keys to the hash values as evenly as possible because collisions become more frequent as hash tables fill up.

In terms of certificates, if two different message digests are processed by SHA-1 and a hashing algorithm produces the same result, then there is a “collision.” The diagram below illustrates a collision at hash “02” by John and Sandra’s keys.



Cryptography

Cryptography prior to the modern age was effectively synonymous with *encryption*, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message shared the decoding technique needed to recover the original information only with intended recipients, thereby precluding unwanted persons to do the same. Since World War I and the advent of the computer, the methods used to carry out cryptology have become increasingly complex and its application more widespread.

Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure; theoretical advances (e.g., improvements in integer factorization algorithms) and faster computing technology require these solutions to be continually adapted. There exist information-theoretically secure schemes that probably cannot be broken even with unlimited computing power - an example is the one-time pad - but these schemes are more difficult to implement than the best theoretically breakable, but computationally, secure mechanisms.

Digital Signature

A Digital Signature is not a digitized signature (electronic scan of signature). While a Digital Signature helps prove your identity and a drawing's authenticity, a digitized signature is nothing more than an electronic version of your own signature inserted or attached to a CAD drawing. It can be forged and copied, and has no tangible security value. Digital Signature is a common term that uses a public key certificate known as a Digital ID. You can either create a self-signed certificate, or purchase a Digital ID or certificate from a CA.

Certificate Authority (CA)

In cryptography, certificate authority, or certification authority, (CA) is an entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate. This allows others (relying parties) to rely upon signatures or assertions

made by the private key that corresponds to the public key that is certified. In this model of trust relationships, a CA is a trusted third party that is trusted by both the subject (owner) of the certificate and the party relying upon the certificate. CAs are characteristic of many Public Key Infrastructure schemes.

Checksum

A checksum or hash sum is a fixed-size datum computed from an arbitrary block of digital data for the purpose of detecting accidental errors that may have been introduced during its transmission or storage. The integrity of the data can be checked at any later time by re-computing the checksum and comparing it with the stored one. If the checksums match, the data was likely not accidentally altered.

Client Server Architecture

The model assigns one of two roles to the computers in a network: Client or server. A “server” is a computer system that selectively shares its resources; a “client” is a computer or computer program that initiates contact with a server in order to make use of a resource. Data, CPUs, printers, and data storage devices are some examples of resources.

This sharing of computer resources is called “time-sharing,” because it allows multiple people to use a computer (in this case, the server) at the same time. Because a computer does a limited amount of work at any moment, a time-sharing system must quickly prioritize its tasks to accommodate the clients.

Clients and servers exchange messages in a request-response messaging pattern - the client sends a request, and the server returns a response. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. All client-server protocols operate in the application layer.

Whether a computer is a client, a server, or both, it can serve multiple functions. For example, a single computer can run web server and file server software at the same time to serve different data to clients making different kinds of requests. Client software can also communicate with server software on the same computer

Cyclic Redundancy Check

A Cyclic Redundancy Check (CRC) (Ewing, 2010) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short “check value” attached, based on the remainder of a polynomial division of their contents; on retrieval, the calculation is repeated and corrective action can be taken against presumed data corruption if the check values do not match.

Encryption

Encryption is the process of encoding messages (or information) in such a way that eavesdroppers or hackers cannot read it.

File Server

In computing, a file server is a computer attached to a network that has the primary purpose of providing a location for shared disk access, i.e., shared storage of computer files (such as documents, sound files, photographs, movies, images, databases, etc.) that can be accessed by the workstations that are attached to the same computer network. The term “server” highlights the role of the machine in the client–server scheme, where the clients are the workstations using the storage. A file server is not intended to perform computational tasks and does not run programs on behalf of its clients. It is designed primarily to enable the storage and retrieval of data while the computation is carried out by the workstations.

Firewall

A firewall can either be software-based or hardware-based and is used to help keep a network secure. Its primary objective is to control the incoming and outgoing network traffic by analyzing the data packets and determining whether it should be allowed through or not, based on a predetermined rule set. A network's firewall builds a bridge between the internal network or computer it protects, upon securing that the other network is secure and trusted, usually an external (inter)network, such as the Internet, that is not assumed to be secure and trusted

Granularity

Granularity is the extent to which a system is broken down into small parts, either the system itself or its description or observation. It is the extent to which a larger entity is subdivided. For example, a yard broken into inches has finer granularity than a yard broken into feet.

Coarse-grained systems consist of fewer, larger components than fine-grained systems; a coarse-grained description of a system regards large subcomponents while a fine-grained description regards smaller components of which the larger ones are composed.

The terms granularity, coarse, and fine are relative when comparing systems or descriptions of systems.

Hash Function

A hash function is any algorithm or subroutine that maps large data sets of variable length to smaller data sets of a fixed length. For example, a person's name, having a variable length, could be hashed to a single integer. The values returned by a hash function are called hash values, hash codes, hash sums, checksums or simply hashes.

Hash Chain

A hash chain is the successive application of a cryptographic hash function to a piece of data. In computer security, a hash chain is a method to produce many one-time keys from a single key or password. For non-repudiation a hash function can be applied successively to additional pieces of data in order to record the chronology of data's existence.

Message Digest

A cryptographic hash function is a hash function; that is, an algorithm that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that any (accidental or intentional) change to the data will (with very high probability) change the hash value. The data to be encoded are often called the "message," and the hash value is sometimes called the message digest or simply digests.

Permissions

There are three specific permissions on Unix-like systems that apply to each class:

1. The *read* permission, which grants the ability to read a file. When set for a directory, this permission grants the ability to read the names of files in the directory (but not to find out any further information about them such as contents, file type, size, ownership, permissions, etc.)
2. The *write* permission, which grants the ability to modify a file. When set for a directory, this permission grants the ability to modify entries in the directory. This includes creating files, deleting files, and renaming files.
3. The *execute* permission, which grants the ability to execute a file. This permission must be set for executable binaries (for example, a compiled C++ program) or shell scripts (for example, a Perl program) in order to allow the operating system to run them. When set for a directory, this permission grants the ability to access file contents and meta data info if its name is known, but not list files inside the directory (unless *read* is set).

The effect of setting the permissions on a directory (rather than a file) is "one of the most frequently misunderstood file permission issues".

When permission is not set, the rights it would grant are denied. Unlike ACL-based systems, permissions on a Unix-like system are not *inherited*. Files created within a directory will not necessarily have the same permissions as that directory.

Protocol

In the natural sciences a protocol is a predefined written procedural method in the design and implementation of experiments.

A security protocol (cryptographic protocol or encryption protocol) is an abstract or concrete protocol that performs a security-related function and applies cryptographic methods.

A protocol describes how the algorithms should be used. A sufficiently detailed protocol includes details about data structures and representations, at which point it can be used to implement multiple, interoperable versions of a program.

Public-Key Cryptography

Public-key cryptography refers to a cryptographic system requiring two separate keys, one of which is secret and one of which is public. Although different, the two parts of the key pair are mathematically linked. One key locks or encrypts the plaintext, and the other unlocks or decrypts the ciphertext. Neither key can perform both functions by itself. The public key may be

published without compromising security, while the private key must not be revealed to anyone not authorized to read the messages.

Seed Value

An initial starting value introduced at the beginning calculation of an Algorithm to reduce dependency of starting values on the System Clock, since the Windows System Clock values are not truly random. By using a random seed value, it reduces the weakness of using a clock value.

Salt Value

A value introduced to cryptographically harden, or reduce dependency on iterative static key uses more than one time. By introducing a random value at a point in the iterative calculation, an Algorithm output is more difficult to reproduce or hack. In other words, salt values introduce more heterogeneous behaviors in the result of the homogeneous algorithm, hence the name of salt.

Versioning

Software versioning is the process of assigning either unique version names or unique version numbers to unique states of computer software. Within a given version number category (major, minor), these numbers are generally assigned in increasing order and correspond to new developments in the software. At a fine-grained level, revision control is often used for keeping track of incrementally different versions of electronic information, whether or not this information is actually computer software.

Workgroup

Microsoft operating systems in the same workgroup may allow each other access to their files, printers, or Internet connection. Members of different workgroups on the same local area network and TCP/IP network can directly access shared resources in workgroups to which they are joined AND other workgroups on the same physical network as evidenced by later versions of windows like XP and Win-7.

APPENDIX G

References

- Digital Signature Standard (DSS)*. (1994, May 19). Retrieved 1 21, 2013, from NIST:
<http://www.itl.nist.gov/fipspubs/fip186.htm>.
- How Certificates Work*. (2003, March 28). Retrieved Feb 25, 2013, from Microsoft Technet:
[http://technet.microsoft.com/en-us/library/cc776447\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc776447(v=ws.10).aspx)
- How Digital Certificates are used for Digital Signatures and Message Encryption*. (2005, 5 19). Retrieved 2 23, 2013, from Microsoft Technet: [http://technet.microsoft.com/en-us/library/bb123848\(v=exchg.65\).aspx](http://technet.microsoft.com/en-us/library/bb123848(v=exchg.65).aspx)
- Digital Signatures*. (2011, Apr). Retrieved Feb 28, 2013, from Wikipedia:
http://en.wikipedia.org/wiki/Digital_signature
- MD5 Hash*. (2013, Feb 21). Retrieved Feb 25, 2013, from Wikipedia:
<http://en.wikipedia.org/wiki/Md5>
- SHA-1*. (2013, Feb 24). Retrieved Feb 25, 2013, from Wikipedia:
<http://en.wikipedia.org/wiki/Sha1>
- Banks & Banks Consulting. (2013). *AEC Signature*. Retrieved Feb 2, 2013, from AEC Signature Blog: <http://www.aecsignature.com>
- Bitser.org. (2010). *Bitser*. Retrieved Feb 27, 2013, from Bitser: <http://www.bitser.org/>
- Blackberry Developer Site. (n.d.). *Recommendation on the use of Hash Algorithms*. Retrieved Feb 25, 2013, from Blackberry Developer Site:
http://developer.blackberry.com/native/documentation/bb10/com.qnx.doc.crypto/topic/c_dg_recommend_hash_algorithm.html
- BPR. (2010, Dec 21). *5J-17.051 Minimum Technical Standards: General Survey-Florida*. Retrieved Feb 28, 2013, from Minimum Technical Standards:
www.800helpfla.com/psm/pdfs/5J-17051.pdf
- Cray, S. R. (2013, 1). *Parity Bit*. Retrieved 2 21, 2013, from Wikipedia:
http://en.wikipedia.org/wiki/Parity_bit
- Cryptographic Hash Function*. (n.d.). Retrieved Feb 28, 2013, from Wikipedia:
http://en.wikipedia.org/wiki/Cryptographic_hash_function
- Ewing, G. (2010, March). *Reverse-Engineering a CRC Algorithm*. Retrieved 2 21, 2013, from <http://www.cosc.canterbury.ac.nz/greg.ewing>:
<http://www.cosc.canterbury.ac.nz/greg.ewing/essays/CRC-Reverse-Engineering.html>
- Frahim, Q. H. (2010). *SSL Remote Access VPNs (Network Security)*. Indiana: Cisco Press.
- Knudsen, J. (1998). *Java Cryptography*. California: O'Reilly.

- MD5 and SHA-1*. (n.d.). Retrieved Feb 25, 2013, from Blackberry Developer Site:
http://developer.blackberry.com/native/documentation/bb10/com.qnx.doc.crypto/topic/c_dg_md5_sha1.html
- Microsoft Technet. (n.d.). *Digital Certificates*. Retrieved Feb. 23, 2013, from Microsoft:
technet.microsoft.com/en-us/library/cc962029.aspx
- Mihir Bellare, T. K. (2004). *Hash Function Balance and Its Impact on Birthday Attacks*. pp401–418. EUROCRYPT.
- NIST. (2005, May 15). *Hash Policy*. Retrieved Feb 25, 2013, from National Institutes of Standards and Testing: <http://csrc.nist.gov/groups/ST/hash/policy.html>
- Rogaway, P., & Shrimpton, T. (2009, June). *Preimage Attack*. Retrieved 2 21, 2013, from Wikipedia: http://en.wikipedia.org/wiki/Preimage_attack.
- Sotomayor, B. (2004). *Public Key Cryptography, The Globus Toolkit 3 Programmer's Tutorial*. Retrieved Feb 23, 2013, from <http://gdp.globus.org>: <http://gdp.globus.org/gt3-tutorial/multiplehtml/ch10s03.html>
- Tanenbaum, A. (1996). *Computer Networks*. New Jersey: Prentice-Hall, Inc.