Public Key Infrastructure (PKI)

Policies, procedures, hardware, software, people:
- Digital certificates: create, distribute, manage, store, revoke

This is a big, big, endeavor:
- Lots of planning

Also refers to the binding of public keys to people and devices:
- The certificate authority (CA)
- It's all about trust

Symmetric encryption:

A single key, shared key:
- Encrypt with the key
- Decrypt with the same key
- If it gets out, you'll need another key

Secret Key algorithm:
- A shared secret

Doesn't scale very well:
- Can be challenging to distribute

Very fast to use:
- Less overhead than asymmetric encryption
- Often combined with asymmetric encryption

Asymmetric encryption:
Public key cryptography:
- Two or more mathematically related keys

Private key:
- Keep this one private

Public key:
- Anyone can see this key
- Give it away

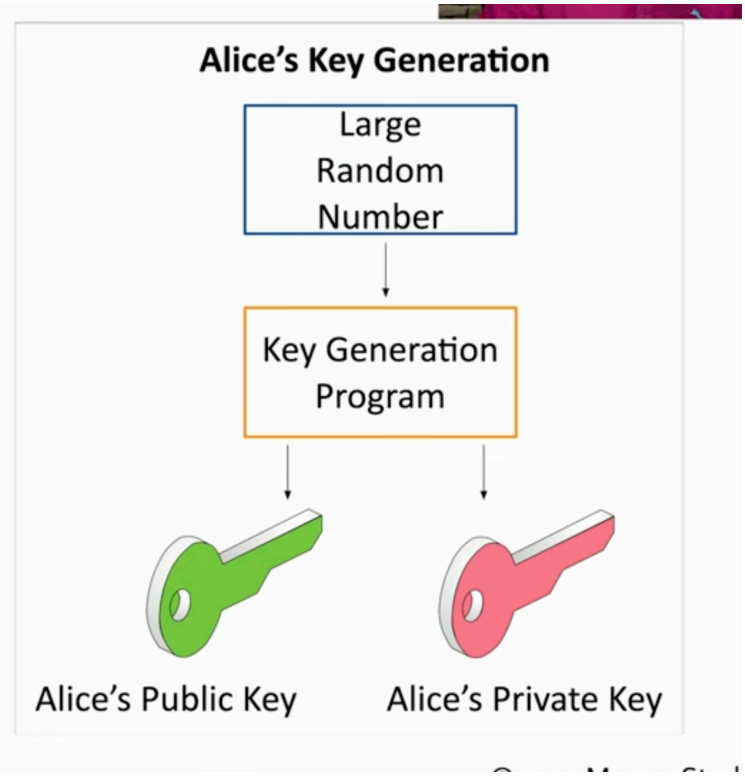
The private key is the only key that can decrypt data encrypted with the public key:
- You can't derive the private key from the public key.

The key pair:

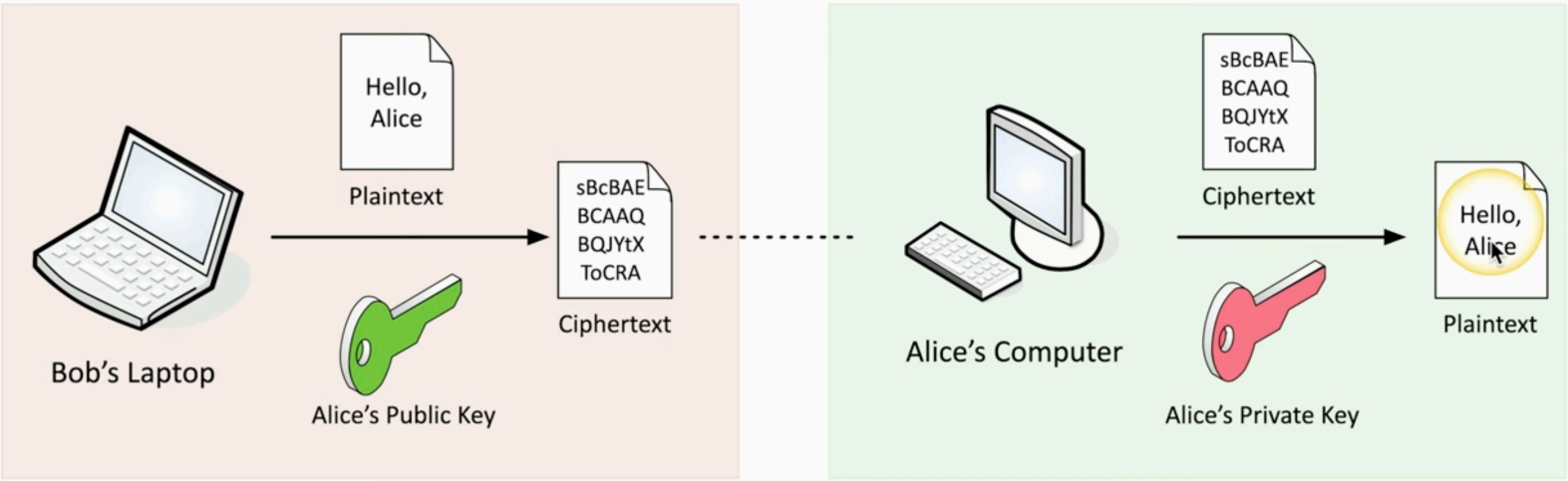
Asymmetric encryption:
- Public key cryptography

Key generation:
- Build both the public and private key at the same time
- Lots of randomization
- Large prime numbers
- Lots and lots of math

EVERYONE can have the public key:
- ONLY ALICE has the PRIVATE key.



Asymmetric encryption:



Step 1: Bob creates a message and has Alices public key
Step 2: The key creates a cipher text that only Alice can unlock with her private key

Step 3: Alice receives the cipher text and her private key unlocks the cipher
Step 4: Alice has decrypted the encrypted message sent by bob

Key escrow:

Someone else holds your decryption keys:
- your private keys are in the hands of a 3rd party
- This may be within your own organization

This can be a legitimate business arrangement:
- A business might need access to employee information
- Government agencies may need to decrypt partner data

Controversial?
- Of course
- But may still be required

Encrypting Data

Protect data, on storage devices:
- SSD, hard drive, USB drive, cloud storage, etc.
- This is data at rest

Full-disk and partition/volume encryption:
-Bitlocker, FileVault, etc.

File encryption:
- EFS (Encrypting File System),third party utilities

Database encryption:

Protecting Stored data:
- And the transmission of that data

Transparant encryption:
- Encrypt all database information with a symmetric key

Record-level encryption:
- Encrypt individual columns
- Use separate symmetric keys for each column

026e711644050	0105fa2b539a0	98fd11fd1f1d764	d76415
c66429f11	2ce10d074b	9dac504b31e	cf070802c81
9d6af8956	d5be49	0c2d8a898	f3ee5646ba5
33dea7fa	8bd8f6f6c	83a7bb35	1e3ed8fa5b
6d7a5b0	34c990	5428c430	813b240d38

—>

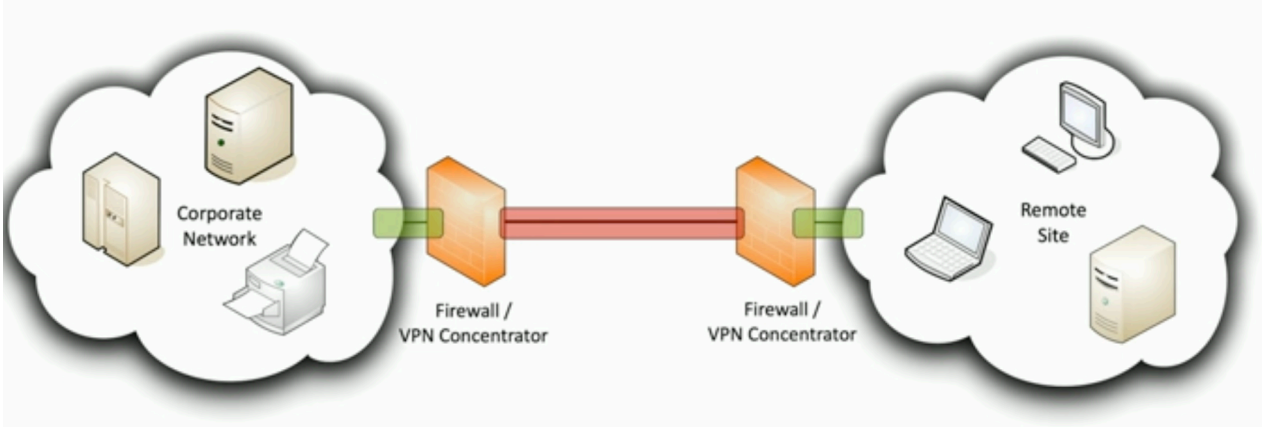
Employee ID	First Name	Last Name	d76415
1001	George	Hammond	cf070802c81
1002	Daniel	Jackson	f3ee5646ba5
1003	Jack	O'Neill	1e3ed8fa5b
1004	Samantha	Carter	813b240d38

Transport encryption:

Protect data traversing the network:
- You're probably doing this now

Encrypting in the application:
- Browsers can communicate using https

VPN (Virtual Private Network):
- Encrypts all data transmitted over the network
- Client-based VPN using SSL/TLS
- Site-to-Site VPN using IPsec

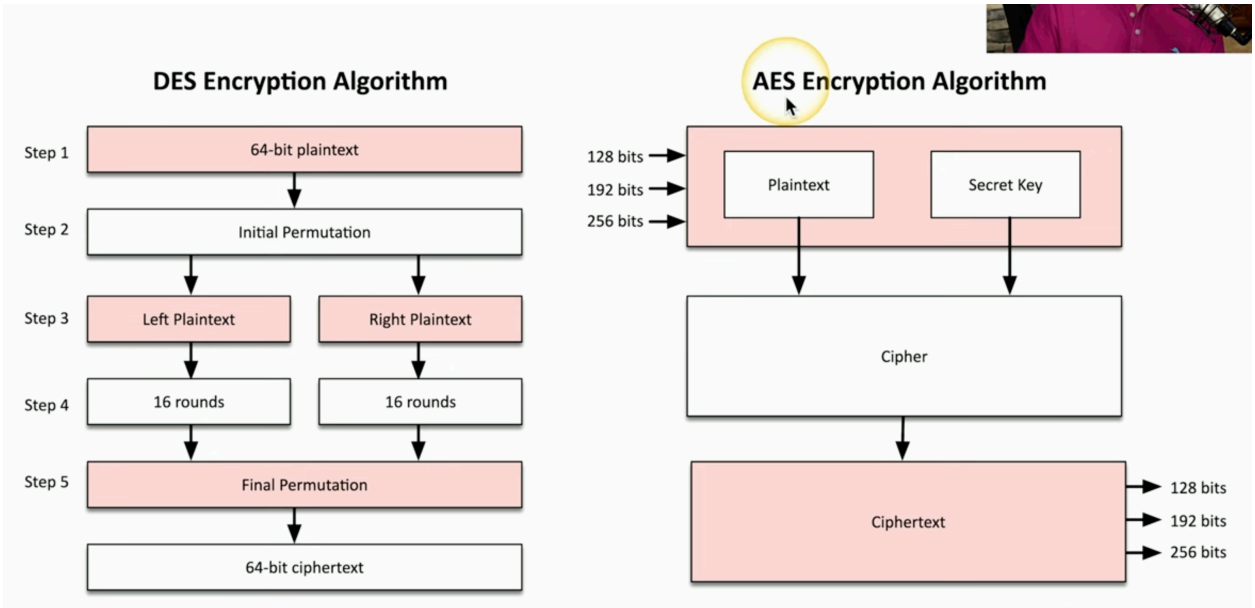


Encryption algorithms:

There are many, many different wats to encrypt data:
- The proper "formula" must be used during encryption and decryption

Both sides decide on algorithm before encrypting the data
- The details are often hidden from the end user

There are advantages and disadvantages between algorithms:
-Security level, speed, complexity and implementation, etc.



Professor Messer November 23rd-24th, 2025
Security+ 1.4

Cryptographic keys:

There's very little that isn't known about the cryptographic process:
- The algorithm is usually a known entity
- The only thing you don't know is the key

The key determines the output:
- Encrypted data
- Hash value
- Digital signature

KEEP YOUR KEY PRIVATE!:
- Its the only thing protecting your data

Key lengths:

Larger keys tend to be more secure:
- Prevent brute force attacks
- Attackers can try every possible key combo

Symmetric encryption:
- 128bit or larger symmetric keys are common
- These numbers get larger and larger as time goes on

Asymmetric encryption:
- Complex calculations of prime numbers
- Larger keys than symmetric encryption
- Common to see key lengths of 3,072 bits or longer

Key stretching:

A weak key is a weak key:
- By itself, its not very secure

Make a weak key stronger by performing multiple processes:
- Hash a password. Hash the hash of the password. And continue
- Key stretching, key strengthening

Brute force attacks would require reversing each of those hashes:
- The Attacker has to spend much more time, even though the key is small

Key Exchange

A logistical challenge:
- How do you share an encryption key across an insecure medium without physically transferring key

Out of band key exchange:
- Dont send the symmetric key over the 'net
- Telephone, courier, in-person, etc.

In band key exchange:
- Its on the network
- Protect the key with additional encryption
- Use asymmetric encryption to deliver a symmetric key

Professor Messer November 23rd-24th, 2025
Security+ 1.4

Real time encryption/ decryption:

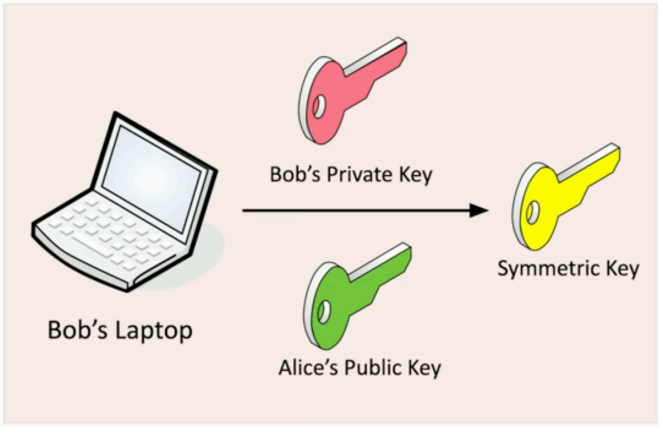
There's a need for fast security:
- Without compromising the security part

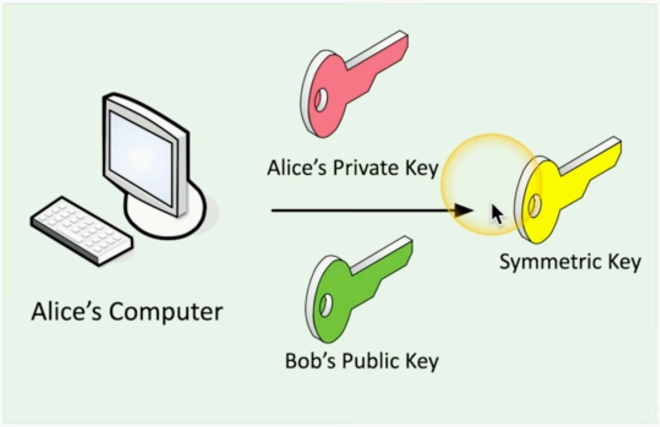
Share symmetric session key using asymmetric encryption:
- Client encrypts random (symmetric key) with a servers public key
- The server decrypts this shared key and uses it to encrypt data
- This is the session key

Implement session keys carefully:
- Need to be changed often (ephemeral keys)
- Need to be unpredictable

Symmetric key from asymmetric keys:

Use public and private key cryptography to create a symmetric key:
- Math is powerful





Step 1: Alice creates symmetric key
[Symmetric Key: "abc123"]

Step 2: Alice encrypts it with Bob's public key
Bob's Public Key → [Encrypted: "xj#9k2@"]

Step 3: Sent over network (safe even if intercepted!)
Network → [Encrypted: "xj#9k2@"]

Step 4: Bob decrypts with his private key
Bob's Private Key → [Symmetric Key: "abc123"]

Step 5: Both have same symmetric key, use for communication
Alice ←→ [Fast AES encryption] ←→ Bob

Encryption technologies

Trusted Platform module(TPM):

A specification for cryptographic functions:
- Cryptography hardware on a device

Cryptographic processor:
- Random number generator, key generator

Persistent memory:
- Unique keys burned in manufacturing

Versatile memory:
- Storage keys, hardware config information
- Securely store BitLocker keys

Password protected:
- No dictionary attacks

Hardware Security Module(HSM):

Used in large environments:
- Clusters, redundant power
- Securely store thousands of cryptographic keys

High end cryptographic hardware:
- Plug in card or separate hardware device

Key backup:
- Secure storage in hardware

Cryptographic accelerators:
- Offload that CPU overhead from other devices

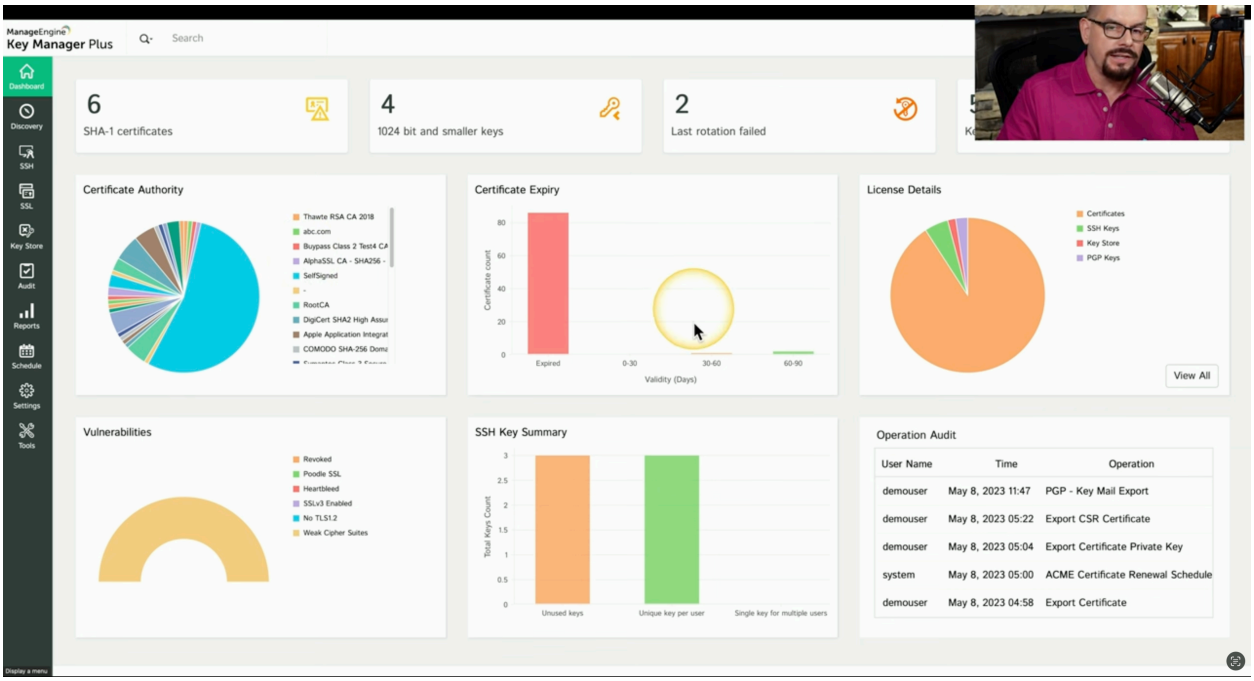
Key management system:

Services are everywhere:
- On-premises, cloud-based
- Many different keys for many different services

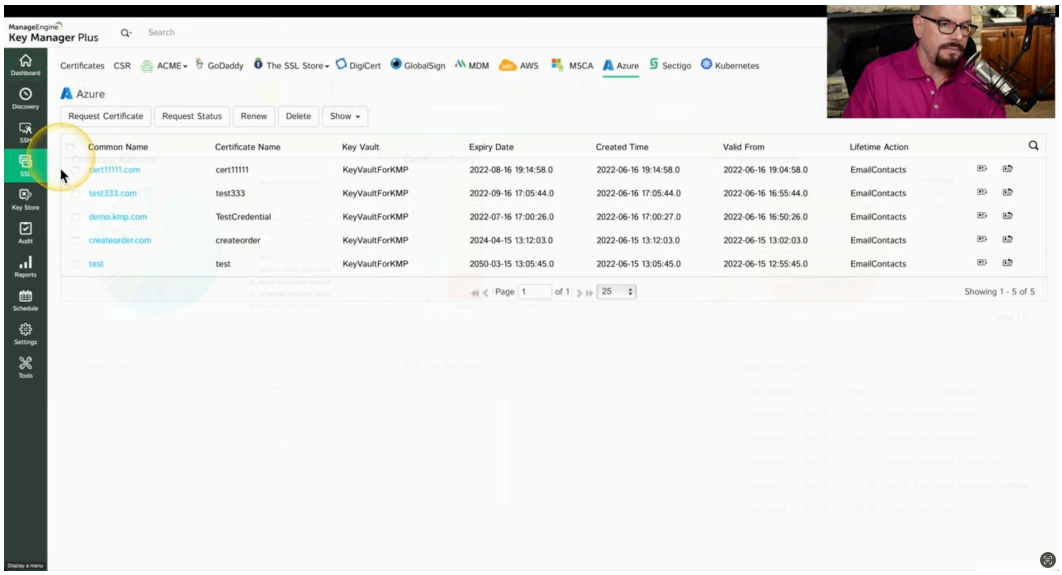
All key management from one console:
- Create keys for specific service or cloud provider(SSL/TLS, SSH, etc)
- Associate keys with specific users
- Rotate keys on regular intervals
- Log key use and important events

Professor Messer
Security+ 1.4

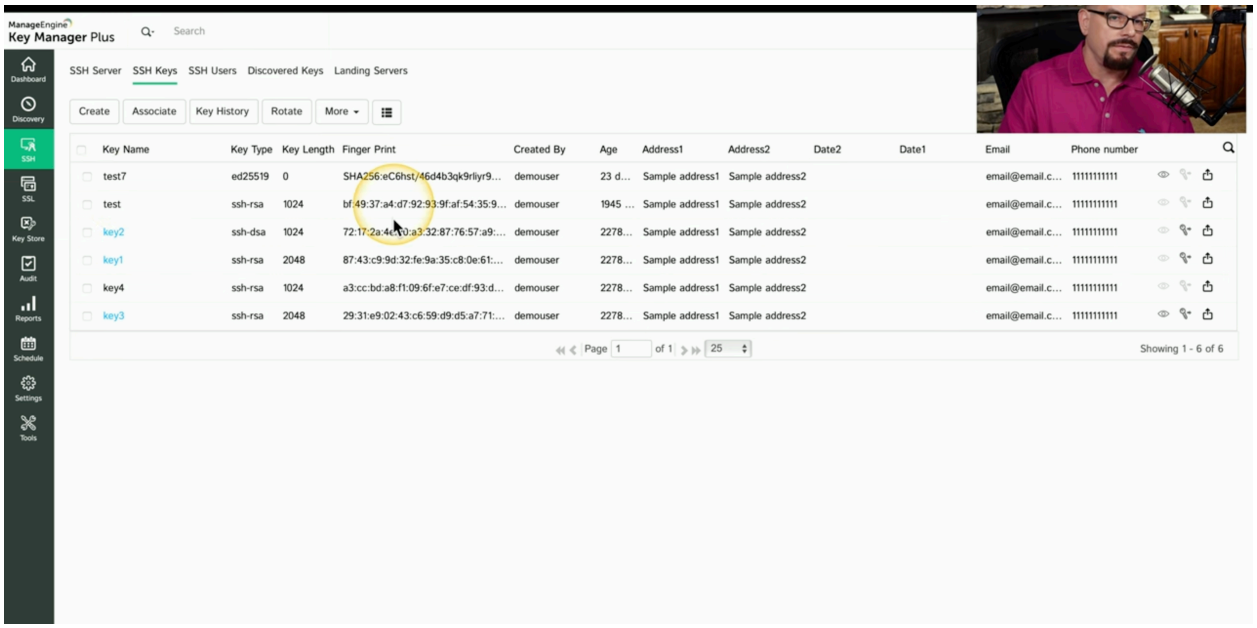
November 23rd-24th, 2025



Dashboard of key management system



Keys that have been created and what server they are associated with



Ssh console communication with details and where the key might be used

You can also create reports on the repots tab and analyze how the keys are being used and how often they are being used

Keeping Data private:

Our data is located in different places:
- Mobile phones, laptops, etc
- The most private data is often physically closest to us

Attackers are always finding new techniques
- Its a race to stay one step ahead.

Our data is changing constantly:
- How do we keep this data protected?

Secure Enclave:
A protected are for our secrets:
- Often implemented as a hardware processor
- Isolated from the main processor
- Many different technologies and names

Provides extensive security features:
- Has its own boot ROM
- Monitors the system boot process
- True random number generator
- Real -time memory encryption
- Root cryptographic keys
- Performs AES encryption in hardware

Obfuscation

The Process of making something unclear;
- Its now much more difficult to understand

But its not impossible to understand:
- If you know how to read it

Hide information in plain sight:
- Store payment information without storing a credit card number

Hide information inside of an image:
- Steganography

Steganography:

Greek for concealed writing:
- Security through obscurity

Message is invisible:
- But its really there

The covertext:
The container or document or file

Common steganography techniques:

Network based:
- Embed messages in TCP packets

Use an image:
- embed the message in the image itself

Invisible watermarks;
- Yellow dots on printers

Audio steganography:
- Modify the digital audio file
- Interlace a secret message within the audio
- Similar technique to image steganography

Video steganography:
- Sequence of images
- Use of image steganography on a larger scale
- Manage the signal to noise ratio
- Potentially transfer much more information

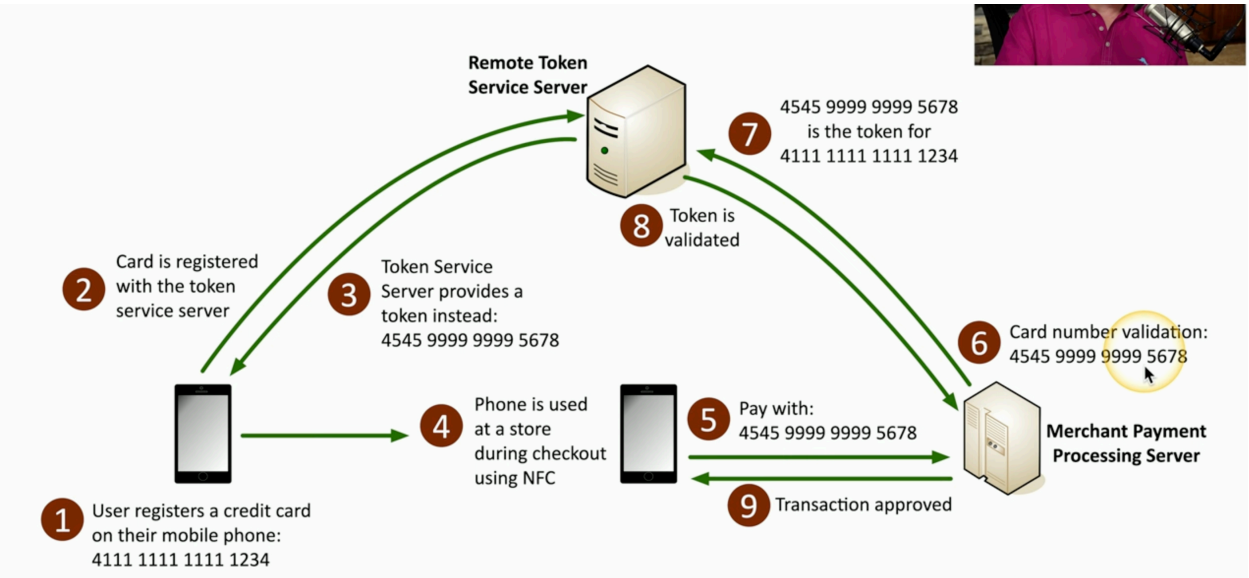
Tokenization - type of steganography

Replace sensitive data with a non-sensitive placeholder:
- SSN 266-12-1112 is now 691-61-8539

Common with credit card processing:
- Use temporary token during payment

- An attacker capturing the card numbers can't use them later

This isn't encrypting or hashing:
- The original data and token aren't mathematically related



The tokenization process with the example of using a credit card.

All that's happening with this is that when you register a credit card number the remote token service server receives that card number and creates a different token number instead of your actual credit card number but the remote token service server knows that your real credit card is tied to that "fake" card number. So when you go buy something the "fake" card number is used and the remote token service server knows the real card number tied to the fake card number it issued and validates the payment. The remote token service server is basically wrapping your real card number with a different number to protect it against attackers.

Data Masking:

Data obfuscation:
- Hide some of the original data

Protects PII:
- And other sensitive data

May only be hidden from view:
- The data may still be intact in storage
- Control the view based on permission

Many different techniques:

- Substituting, shuffling, encrypting, masking out, etc.

Obfuscation Use Cases:

Defensive (Good):
- Tokenization: Protect payment data
- Data Masking: Privacy protection
- Code obfuscation: Protect intellectual property

Offensive (Bad):
- Malware obfuscation: Evade antivirus
- Steganography: Data exfiltration
- Script obfuscation: Hide malicious commands
- URL obfuscation: Phishing attacks

Key Exam Differences:

Technique	Reversible?	Needs Key?	Mathematical?
Tokenization	Yes (token service)	No	No
Encryption	Yes	Yes	Yes
Hashing	No	No	Yes
Data Masking	Maybe	No	No
Steganography	Yes (if method known)	No	No

Remember:
- Obfuscation = HIDE (not encrypt)
- Tokenization ≠ Encryption (no math relationship)
- Steganography = Hidden in plain sight
- Data Masking = Control what people see

Hashing an digital signatures

Hashes:

Represent data as a short string of text:
- A message digest, a fingerprint

One way trip:
- Impossible to recover the original message from the digest
- Used to store passwords / confidentiality

Verify downloaded document is the same as the original:
- Integrity

Can be a digital signature:
- Authentication, non-repudiation and integrity

A hash example:

SHA256 hash:
- 256 bits / 64 hexadecimal

My name is Professor Messer:
- long string of numbers and letters

My name is Professor Messer!:
- Completely different string of letters and numbers both being 256 bits long

Collision:

Hash functions:
- Take an input of any size
- Create a fixed size string
- Message digest, checksum

The hash should be unique:
- Different inputs should never create the same hash
- If they do, its a collision

MD5 has a collision problem:
- Found in 1996
- Dont use MD5 for anything important

Practical Hashing:

Verify a downloaded file:
- Hashes may be provided on the download site
- Compare the downloads file hash with the posted hash value

Password storage;
- Instead of storing the password, store a salted hash
- Compare hashes during the authentication process
- Nobody ever knows your actual password

Adding some salt:
Salt:
- Random data added to a password when hashing

Every user gets their own random salt:
- The salt is commonly stored with the password

Rainbow tables won't work with salted hashes:
- Additional random values added to the original password

This slows things down for the brute force attack strategy:
- It doesn't completely stop the reverse engineering

Salting the hash:

Each user gets a different random hash:
- The same password creates a different hash.

Password	Hash
dragon	a9c43be948c5cabd56ef2bacffb77cdaa5eec49dd5eb0cc4129cf3eda5f0e74c
dragon+ gsEVx	35172b0b7c9c3002bbf02908b3f330dde5f5eda42b0b4d553ae0759eb25686c8
dragon+ LTBkP	b9909221fbbda70bea27644b84443ddb68f78ede2253fa8f6409fd8b7602599a
dragon+ HTsBK	cf1c25063093411faefb09198356de6775955b5ff6e0fe9aab665a3e95d11e25
dragon+ MnNEo	99658da1af957af6f26790189f0a1957018c0962361580c5bd5fc50d26b71579

Professor Messer
Security+ 1.4

November 23rd-24th, 2025

Digital signatures:

Prove the message was not changed:
- Integrity

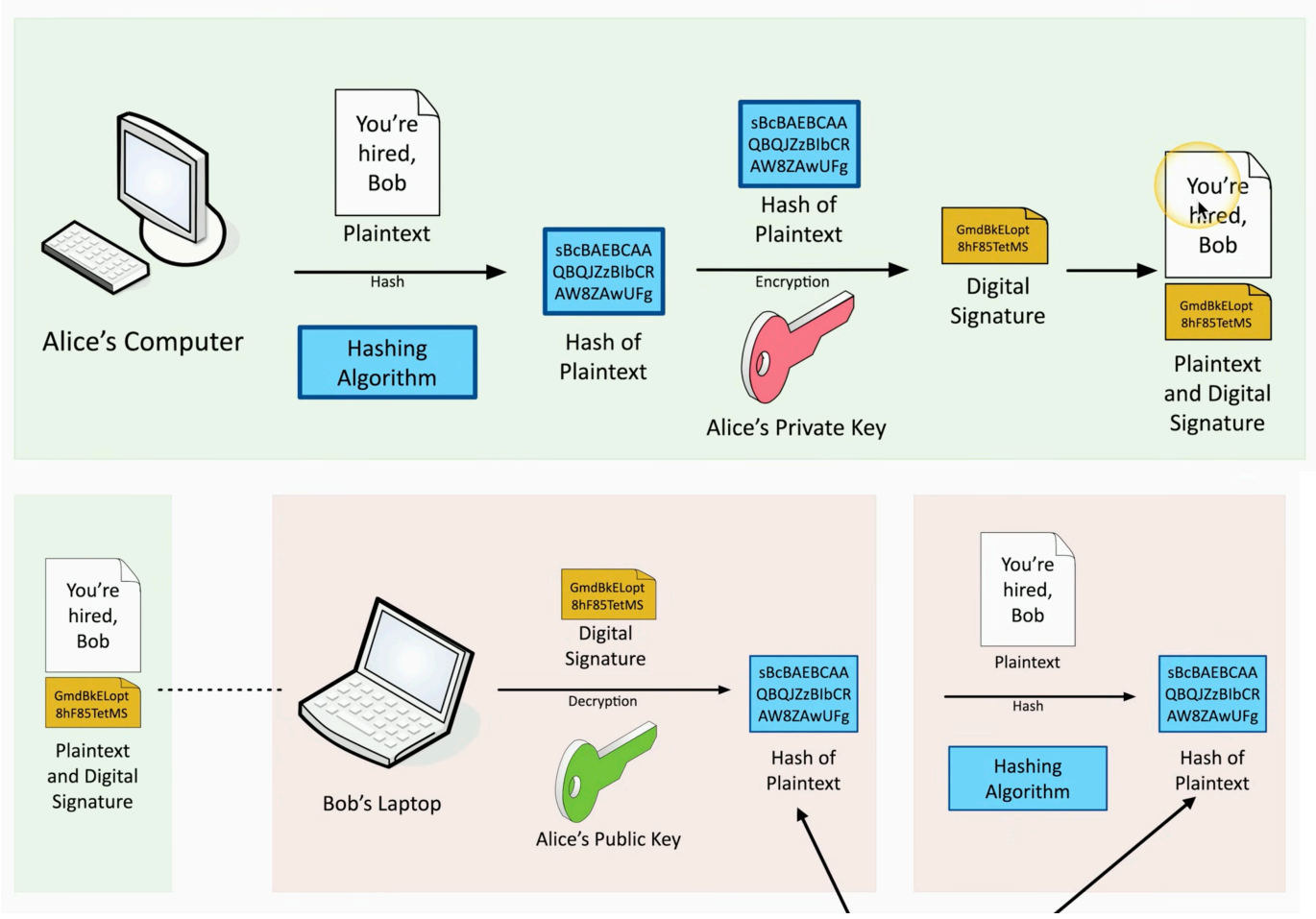
Prove the source of the message:
- Authentication

Make sure the signature isn't fake:
-Non-repudiation

Sign with the private key;
- The message doesn't need this to be encrypted
- Nobody else can sign this

Verify with the public key:
- any change in the message will invalidate the signature

Creating a digital signature:



Professor Messer November 23rd-24th, 2025
Security+ 1.4

Summary of creating a digital signature:

Alice creates a plaintext document with a text on it. The hashing algorithm is activated to create that specific plaintext document a hash or a very long string of letters and numbers. The hash associated with that plain text is then encrypted using Alices private key and is then digitally signed by Alice saying this document is from me (non-repudiation) and hasn't been altered (integrity). The digital signature is then associated with the plaintext document where it is then sent to bob. Bob receives the plaintext document with the digital signature and starts to decrypt the HASH of the plaintext Alice wrote with Alices public key which everyone has access to. After the decryption process of the hash of the plaintext, Bob already had access to the document and however not until after the hashing algorithm finished decrypting the hash of that message was Bob able to see if the hashes matched to ensure integrity, authentication and non-repudiation.

Blockchain Technology

A distributed ledger:
- Keep track of individual transactions

Everyone on the blockchain network maintains the ledger:
- Records and replicates to anyone and everyone

Many practical applications:
- Payment process
- Digital identifcation
- Supply chain monitoring
- Digital voting

The Blockchain process:

Step 1:

A transaction is requested

The transaction could be any digital transaction from transferring bitcoins, medical records Data backups, or transferring house information.

Step 2:

The transaction is sent to every computer (or node) in a decentralized network to be verified.

Step 3:

The verified transaction is added to a new block of data containing other recently verified transactions.

Step 4:

A secure code (a hash) is calculated from the previous blocks of transaction data in the blockchain. The hash is added to the new block of verified transactions.

Step 5:

The block of data is added to the end of the blockchain, which is then updated to all nodes in the network for security.

The transaction is complete.

Notes:

If any blocks are altered, its hash and all following hashes in the chain are automatically recalculated. The altered block chain will no longer match the chains stored by the rest of the network, and will be rejected.

Certificates:

Digital Certificates:

A public key certificate:
- Binds a public key with a digital signature
- And other details about the key holder

A digital signature adds trust:
- PKI uses certificate authorities for additional trust
- Web of trust adds other users for additional trust

Certificate creation can be built into the OS:
- Part of Windows Domain services
- Many 3rd party options

What's in a digital certificate:

X.509:
- Standard format

Certificate details:
- serial number
- Version
- Signature Algorithm
- Issuer
- Name of cert holder
- Public key
- Extensions
- And more..

Root of Trust;

Everything associated with IT security requires trust:
- A foundational characteristic.

How to build trust from something unknown?
- Someone/something trustworthy provides their approval

Professor Messer November 23rd-24th, 2025
Security+ 1.4

Refer to the root of trust:
- An inherently trusted component
- Hardware, software, firmware, or other component
- Hardware security module (HSM), secure enclave, certificate authority

Certificate Authorities

You connect to a random website:
- Do you trust it/

Need a good way to trust an unknown entity:
- Use a trusted 3rd party
- An Authority

Certificate Authority (CA) has digitally signed the website certificate:
- You trust the CA, therefore you trust the website
- Real time verification

Third party certificate authorities:

Built- in to your browser:
- Any browser

Purchase you web site certificate:
- It will be trusted by everyone's browser

CA is responsible for vetting the request:
- They will confirm the certificate owner
- Additional verification information may be required by the CA

Certificate signing requests

Step 1:
Create a key pair, then send the public key to the CA to be signed
- A certificate signing request or (CSR)

Step 2:

The CA validates the request:
- Confirms DNS emails and website ownership

Step 3:

CA digitally signs the cert:
- Returns to the applicant

Private certificate authorities;

You are your own CA:
- Build it in-house
- Your devices must trust the internal CA

Needed for medium to large organizations:
- Many web servers and privacy requirements

Implement as part of your overall computing strategy
- Windows Certificate Services, OpenCA

Self Signed certificates;

Internal certificates dont need to b signed by a public CA:
- Your company is the only one going to use it
- No need to purchase trust for devices that already trust you

Build your own CA:
- Issue your own certificates signed by your own CA

Install the CA certificate/trusted chain on all devices:
- They'll now trust any certificates signed by your internal CA
- Works exactly like a certificate you purchased

Wildcard certificates;
Subject Alternative Name (SAN)
- Extension to an X.509 certificate
- Lists additional identification information
- Allows a certificate to support many different domains

Wildcard domain:
- Certificates are based on the name of the server
- A wildcard domain will apply to all server names in a domain

Key revocation

Certificate revocation list (CRL):
- Maintained by the Certificate Authority (CA).
- Can contain many revocations in a large file.

Many different reasons:
- Changes all the time

April 2014 - CVE-2014-0160
- Heartbleed
- OpenSSL flaw put the private key of affected web servers at risk
- Open SSL was patched, every web server certificate was replaced
- Older web certificates were moved to the CRL.

OCSP stapling:

Online Certificate Status Protocol:
- Provides Scalability for OCSP checks

The CA is responsible for responding to all client OCSP requests:
- This may not scale well

Instead have the certificate holder verify their own status:
- Status information is stored on the certificate holders server

OCSP status is "stapled " into the SSL/TLS handshake:
- Digitally signed by the CA

Getting revocation details to the browser:

OCSP (Online Certificate Status Protocol):
- The browser can check certificate revocation

Messages usually sent to an OCSP responder via HTTP:
- Easy to support over internet links
- More efficient than downloading a CRL

Not al browser/apps support OCSP:
- Early internet Explorer versions did not support OCSP
- Some support OCSP, but don't bother checking.