05/05/25 Notes

CIA
C | Confidentiality
I | Integrity
A | availability

Risk - likelihood and impact a threat will will exploit vulnerability

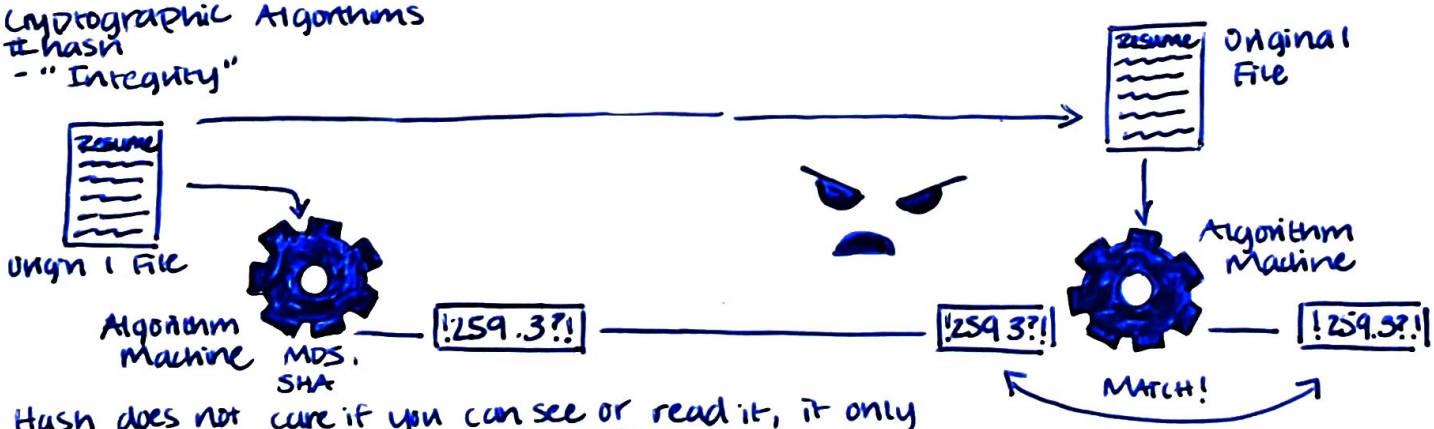
- Threat: harmful event
  - → man-made
  - → natur
- Vulnerability: weakness
- Likelihood: % probability (exploit)
- Impact: $$$ (always measured in $)

Compliance = "Law" and "Regulations"
GDPR = EU privacy
PCI DSS = "credit card"

Security Controls = "Counter Measure"
BIA, BCP, DRP

In cryptography, people spends their life to encrypt something that is NOT meant to be broken "secret writing"

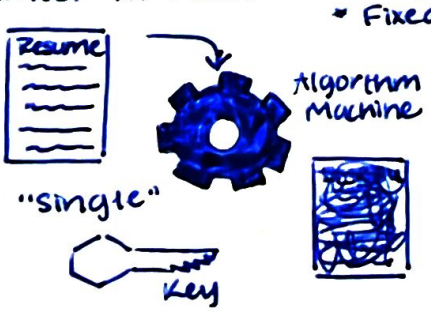
Cryptographic Algorithms
#hash
- "Integrity"



Hash does not care if you can see or read it, it only cares if you are going to mess with it!
Algorithm to remember for hash:
1) MD5
2) SHA
   - SHA 28
   - SHA 256
   - SHA 512

- NOT Reversible
- Fixed Size

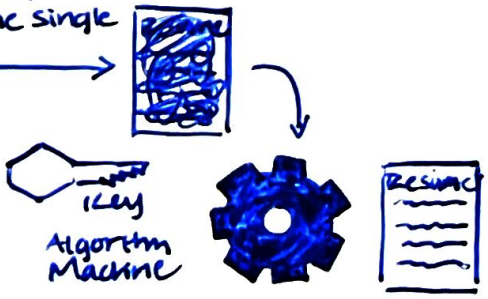
ALGORITHM # MD5/SHA
1). Symmetric: RC5, AES, 3DES, DES
2). Asymmetric: RSA, D.H, ECC



Symmetric key = "single"
- requires a key
  1). RC5
  2). AES
  2). DES/3DES
- private key
- encryption
- confidentiality

- Method starts with receiver!. Not sender

How does recepient get copy of the single key?



05/05/25 Notes

CIA
C | Confidentiality
I | integrity
A | availability

Risk - likelihood and impact a threat will will exploit vulnerability

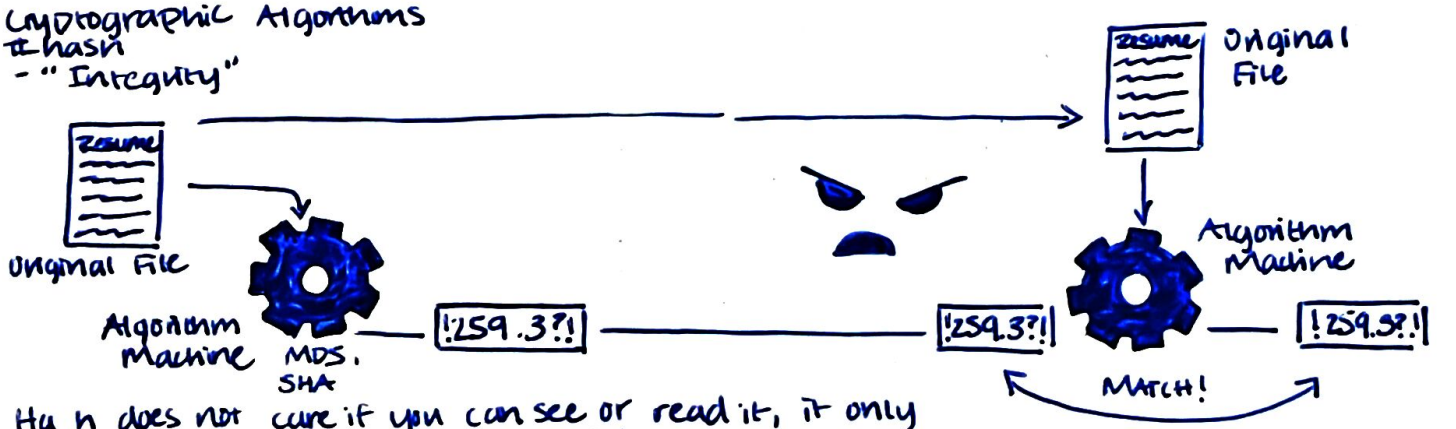
- Threat: harmful event
  - → man-made
  - → natur
- Vulnerability: weakness
- Likelihood: % probability (exploit)
- Impact: $$$ (always measured in $)

Compliance = "Law" and "Regulations"
GDPR = EU privacy
PCI DSS = "credit card"

Security Controls = "Counter Measure"
BIA, BCP, DRP

In cryptography, people spends their life to encrypt something that is NOT meant to be broken "secret writing"

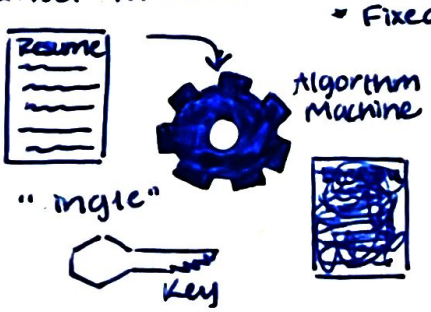
Cryptographic Algorithms
#hash
- "Integrity"



Hash does not care if you can see or read it, it only cares if you are going to mess with it!
Algorithm to remember for hash:
1). MD5
2). SHA
- SHA 28
- SHA 256
- SHA 512

- NOT Reversible
- Fixed Size

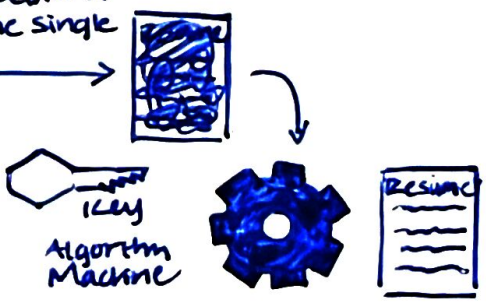
ALGORITHM # MD5/SHA
1). Symmetric: RC5, AES, 3DES, DES
2). Asymmetric: RSA, DH, ECC



Symmetric Key = "single"
- requires a key
  1). RC5
  2). AES
  2). DES/3DES
- private key
- encryption
- confidentiality

- Method starts with receiver! Not sender

How does recepient get copy of the single key?



05/05/25 Notes

CIA
C | Confidentiality
I | Integrity
A | availability

Risk - likelihood and impact a threat will will exploit vulnerability

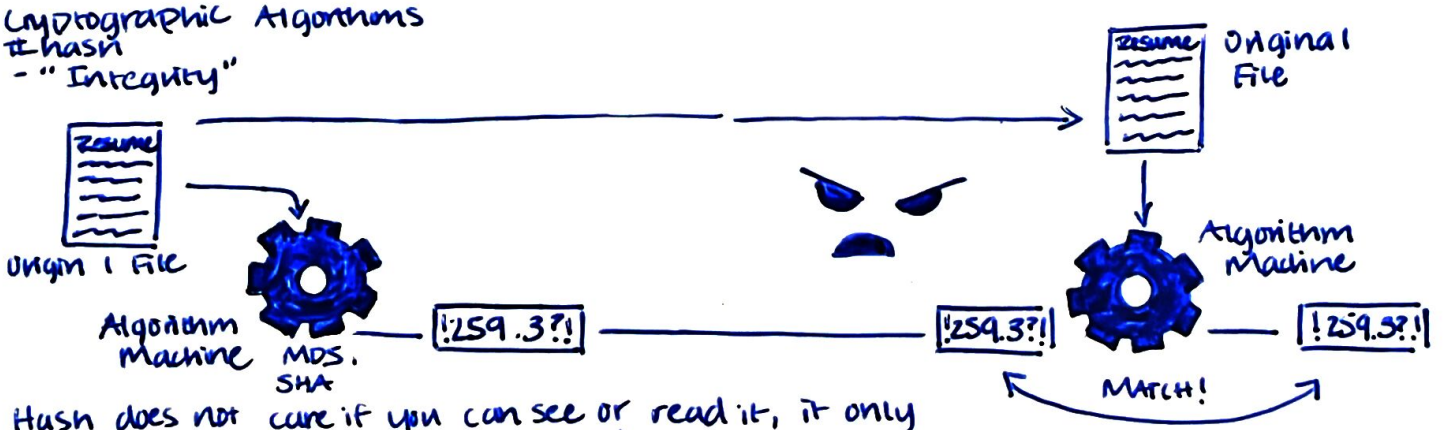
- Threat: harmful event
  - → man-made
  - → natur
- Vulnerability: weakness
- Likelihood: % probability (exploit)
- Impact: $$$ (always measured in $)

Compliance = "Law" and "Regulations"
GDPR = EU privacy
PCI DSS = "credit card"

Security Controls = "Counter Measure"
BIA, BCP, DRP

In cryptography, people spends their life to encrypt something that is NOT meant to be broken "secret writing"

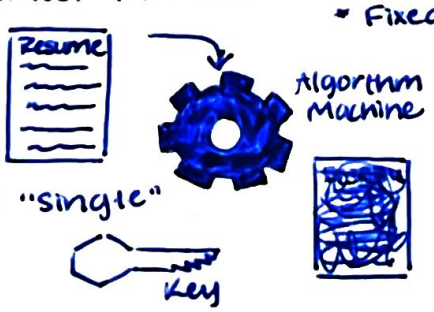
Cryptographic Algorithms
#hash
- "Integrity"



Hash does not care if you can see or read it, it only cares if you are going to mess with it!
Algorithm to remember for hash:
1) MD5
2) SHA
   - SHA 28
   - SHA 256
   - SHA 512

- NOT Reversible
- Fixed size

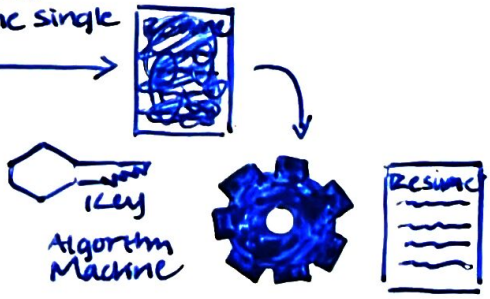
ALGORITHM # MD5/SHA
1). Symmetric: RC5, AES, 3DES, DES
2). Asymmetric: RSA, D.H, ECC



Symmetric key = "single"
- requires a key
  1). RC5
  2). AES
  2). DES/3DES
- private key
- encryption
- confidentiality

- Method starts with receiver! Not sender

How does recepient get copy of the single key?



05/05/25 Notes

CIA
C | confidentiality
I | integrity
A | availability

Risk - likelihood and impact a threat will will exploit vulnerability

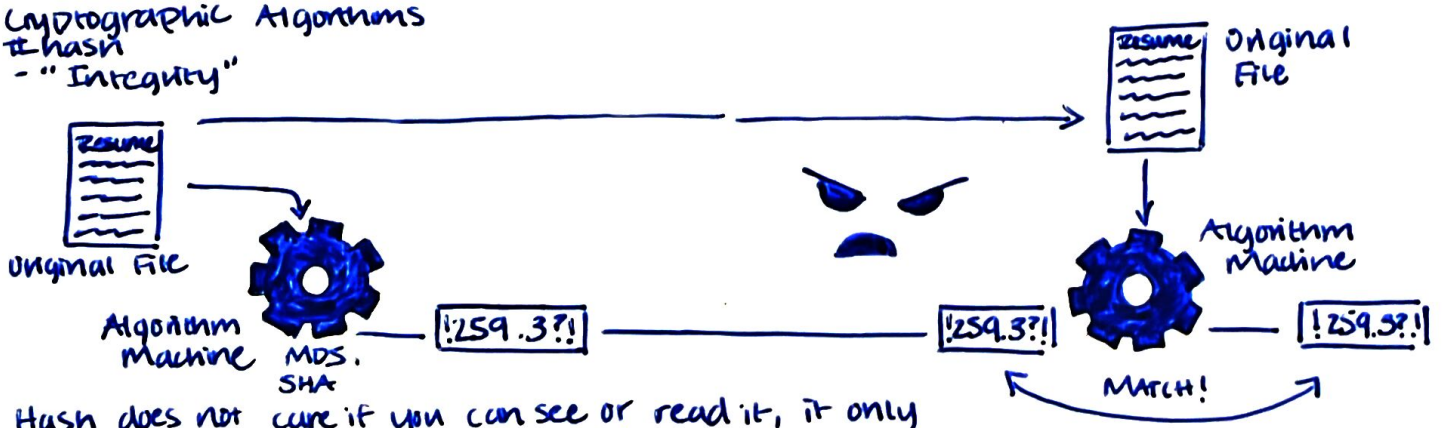
- Threat: harmful event
  - → man-made
  - → natur
- Vulnerability: weakness
- Likelihood: % probability (exploit)
- Impact: $$$ (always measured in $)

Compliance = "Law" and "Regulations"
GDPR = EU privacy
PCI DSS = "credit card"

Security Controls = "Counter Measure"
BIA, BCP, DRP

In cryptography, people spends their life to encrypt something that is NOT meant to be broken "Secret writing"

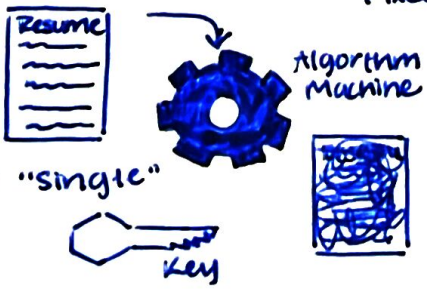
Cryptographic Algorithms
#hash
- "Integrity"



Hash does not care if you can see or read it, it only cares if you are going to mess with it!
Algorithm to remember for hash:
1) MD5
2) SHA
   - SHA 28
   - SHA 256
   - SHA 512

* NOT Reversible
* Fixed Size

ALGORITHM # MD5 / SHA
1). Symmetric: RC5 AES, 3DES DES
2). Asymmetric: RSA, D H, Ecc



Symmetric Key = "single"
* requires a key
  1). RC5
  2). AES
  2). DES/3DES
* private key
* encryption
* confidentiality

* Method starts with eceiver !. Not sender

How does receipient get copy of the single key?