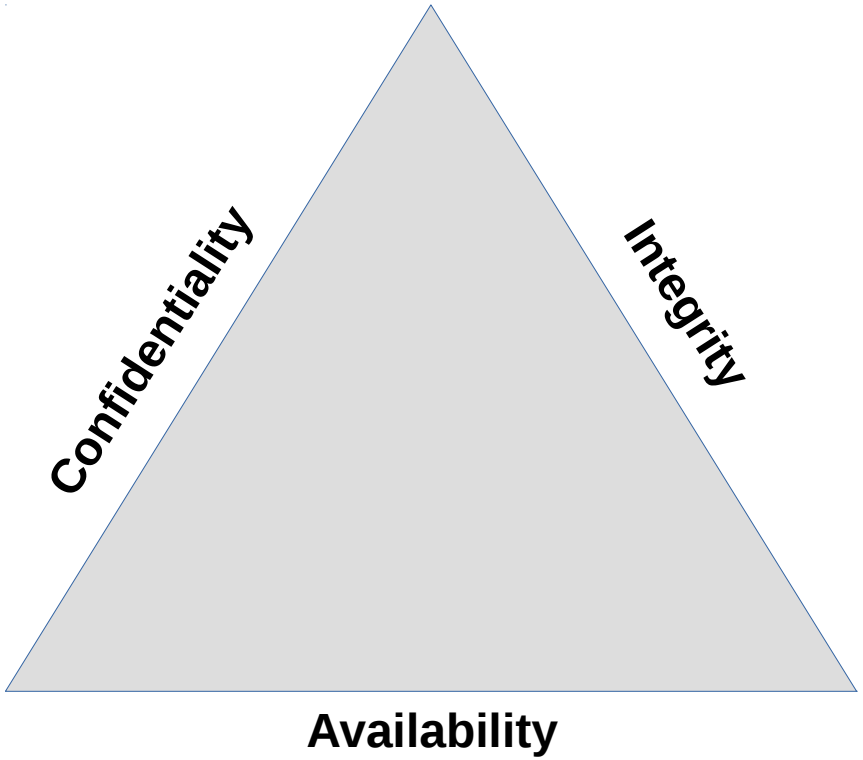
Deep Dive Cryptography
November 2018

Kris Hardy
rhardy9@cnm.edu

What We'll Learn

- Intro to Cryptography Concepts
- Hashing
- Password-based Key Deriviation
- Private-Key (Symmetric) Encryption
- Public-Key (Assymetric) Encryption

Requirements for a Secure System



CIA Triad

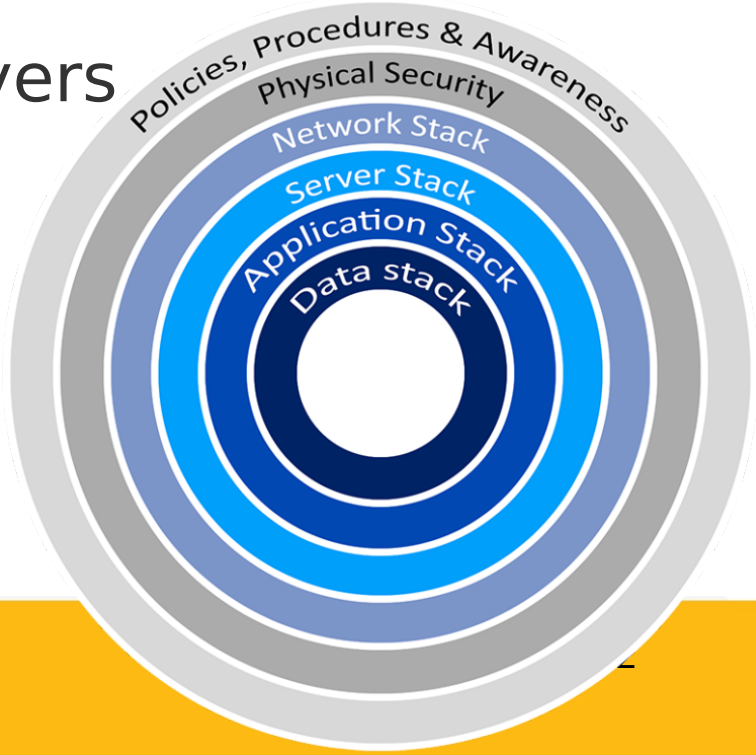
Confidentiality – Is the data/system private?

Integrity – Is the data/system correct?

Availability – Is the data/system available?

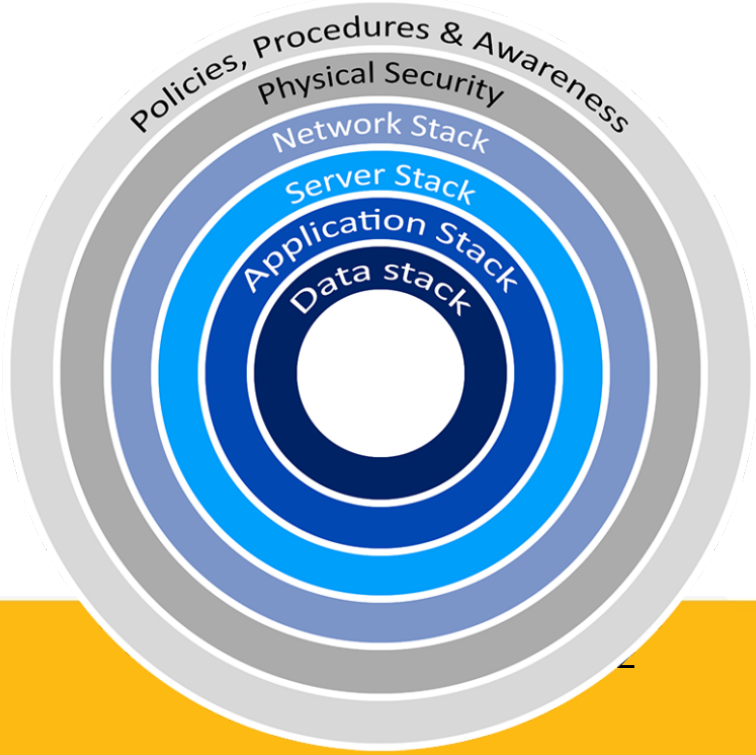
Defense in Depth (ie the Onion Model)

- No system is perfect, so security needs to be considered at each layer
- Data has to flow through various security layers
- Different controls will be used at different layers



Defense in Depth Examples

- Policy, Procedures & Awareness – Employee and user training, incidence response plans, etc.
- Physical Security – Locked facilities, badges, etc.
- Network Stack – Firewalls, VPNs, IDS/IPS
- Server Stack – OS, system firmware updates, secure configuration
- Application Stack – **Input sanitization**, User authentication
- Data Stack – Encryption, Backups



Cryptography

- The science of encrypting information to hide it from others
- **Cipher** (aka cryptographic algorithm) – The method used to encrypt, decrypt or authenticate the information
- Ciphers are hard to design correctly and hard to implement bug-free
- **NEVER ROLL YOUR OWN CRYPTO!** Use proven and expert-reviewed libraries, random number generators, etc.

When to apply cryptography

- **Data-at-rest** – Data is stored somewhere (hard drive, database, etc.) and is not being actively used and is not in RAM.
- **Data-in-transit** – Data is being moved between systems, usually over a network.
- **Data-in-use** – Data is being actively used, often in RAM.

Quick Example: Symmetric Encryption

passwords.txt

Username: khardy
Password: IL0v3Cryp40

Encrypt
Cipher: aes128
Key: MySecret

passwords.txt.aes (shown in a hex editor)

```
00000000 41 45 53 02 00 00 18 43 52 45 41 54 45 44 5F 42 59 00 61   AES....CREATED_BY.a
00000013 65 73 63 72 79 70 74 20 33 2E 31 33 00 80 00 00 00 00 00   escrypt 3.13.......
00000026 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
00000039 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
0000004C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
0000005F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
00000072 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
00000085 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
00000098 00 00 00 00 00 00 00 00 00 00 00 CE 77 4B 83 1F 25 8F 53   ............wK..%.S
000000AB 0F 3C A7 E7 15 55 3D B3 51 D8 B9 A5 F6 3A B6 A8 4A C1 E1   .<...U=.Q....:..J..
000000BE 5E 3C 99 0A CC C1 90 72 CC 74 8F 8F 7B AC 56 5C EA 87 3E   ^<.....r.t..{.V\..>
000000D1 48 49 59 B2 7D 92 63 BD 4B 2B 13 6B AB 1B 22 9B B2 28 E1   HIY.}.c.K+.k.."..(.
000000E4 8A B4 C5 77 AB E7 54 BF A1 34 5E C6 F0 69 EA 14 85 25 BE   ...w..T..4^..i...%.
000000F7 13 B4 BB D2 3B 3C F4 04 44 E9 DA 06 CC 35 E9 47 AA 00 89   ....;<..D....5.G...
0000010A 00 A1 2C C7 C0 C4 43 6B 56 70 FF 51 D0 13 61 B7 6B F5 4F   ..,...CkVp.Q..a.k.O
0000011D 09 DD 10 73 5B CA 1F B3 0E 75 F4 8E CC DB B3 8E 0B AC 84   ...s[....u.........
00000130 FF BB 87 07 1B E7 4A 33 57 69 D3 A7 80 46 B2 CE 76 FD 1D   ......J3Wi...F..v..
00000143 5A DE 0E DD 76 E9 5F AB 89 32 48 3E 34 5D 46 C4 6A         Z...v._..2H>4]F.j
```

passwords.txt.aes


```
00000000 41 45 53 02 00 00 18 43 52 45 41 54 45 44 5F 42 59 00 61   AES....CREATED_BY.a
00000013 65 73 63 72 79 70 74 20 33 2E 31 33 00 80 00 00 00 00 00   escrypt 3.13.......
00000026 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
00000039 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
0000004C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
0000005F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
00000072 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
00000085 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...................
00000098 00 00 00 00 00 00 00 00 00 00 00 CE 77 4B 83 1F 25 8F 53   ............wK..%.S
000000AB 0F 3C A7 E7 15 55 3D B3 51 D8 B9 A5 F6 3A B6 A8 4A C1 E1   .<...U=.Q....:..J..
000000BE 5E 3C 99 0A CC C1 90 72 CC 74 8F 8F 7B AC 56 5C EA 87 3E   ^<.....r.t..{.V\..>
000000D1 48 49 59 B2 7D 92 63 BD 4B 2B 13 6B AB 1B 22 9B B2 28 E1   HIY.}.c.K+.k.."..(.
000000E4 8A B4 C5 77 AB E7 54 BF A1 34 5E C6 F0 69 EA 14 85 25 BE   ...w..T..4^..i...%.
000000F7 13 B4 BB D2 3B 3C F4 04 44 E9 DA 06 CC 35 E9 47 AA 00 89   ....;<..D....5.G...
0000010A 00 A1 2C C7 C0 C4 43 6B 56 70 FF 51 D0 13 61 B7 6B F5 4F   ..,...CkVp.Q..a.k.O
0000011D 09 DD 10 73 5B CA 1F B3 0E 75 F4 8E CC DB B3 8E 0B AC 84   ...s[....u.........
00000130 FF BB 87 07 1B E7 4A 33 57 69 D3 A7 80 46 B2 CE 76 FD 1D   ......J3Wi...F..v..
00000143 5A DE 0E DD 76 E9 5F AB 89 32 48 3E 34 5D 46 C4 6A         Z...v._..2H>4]F.j
```


Decrypt
Cipher: aes128
Key: MySecret

passwords.txt

Username: khardy
Password: IL0v3Cryp40

Hashing

hash(“sha256”, “mypassword”) => 89e01536ac207279409d4de1e5253e01f4a1769e696db0d6062ca9b8f56767c8

- A function which performs one-way encryption, but not decryption.
- Useful for situations where you don’t want or need the plaintext, but need to know if something is different.
- Examples:
  - Signatures
  - Storing passwords (you don’t want the password, but need to know if it is correct)
  - Verify that a file’s contents haven’t changed

Hashing Algorithms

- **MD5** – Message Digest. Obsolete.
- **SHA-1** – Secure Hash Algorithm. Vulnerable to collisions, obsolete.
- **SHA-2** – Includes SHA-224, SHA-256, SHA-384, SHA-512. Creates a hash of the named bit-length. (SHA-512 creates 512-bit hashes)
- **SHA-3** – New. Named in 2012 by NIST.
- **HMAC** – Hashed Message Authentication Code. Hash-based system to detect message tampering using a Message Authentication Code (MAC).

Hashing Algorithms

- **RIPEMD** – RACE Integrity Primitives Evaluation Message Digest. 128-bit version is vulnerable.
- **RIPEMD-160** – Strengthened RIPEMD algorithm that creates a 160-bit hash. RIPEMD-256 and RIPEMD-320 are available, but don’t significantly increase the hash strength.

Hashing in PHP

- hash() - Hashes a value
- hash_equals() - Compares a two hashes in a way to avoid timing attacks
- hash_algos() - Lists available algorithms

Key Stretching

- A feature of some cryptographic algorithms that derive strong keys from weaker passwords.
- Example: Password-based Key Derivation Functions (**PBKDF**)
  - Runs repetitive hashing on a password to create a repeatable, high-entropy key
  - Common Algorithms: PBKDF2, Argon2

Key Stretching Algorithms

- **BCRYPT** – Derives keys using the Blowfish cipher and salting, and adds an iteration count to provide adaptive strength.
  - Adaptive function – Part of an algorithm that allows it to increase in strength as computing power increases.
- **PBKDF2** – Password-based Key Derivation Function 2
  - Uses a passphrase and salt, and applies an HMAC to the input iteratively to provide adaptive strength.

PBKDF2 in PHP

- hash_pbkdf2()
- hash_equals()
- password_hash()
- password_verify()

Terms

- **Plaintext** – The unencrypted information that needs to be protected
- **Ciphertext** – The resulting information after encryption that is created by the cipher
- **Cryptanalysis** – Attempting to break the cryptography by analyzing the plaintext, ciphertext and/or cipher

Cryptographic Security Concerns

- **Collision** – When two different inputs result in the same output.
- **Key Strength** – Usually, the longer the key, the stronger the encryption. Each cipher has its own key length.
  - DES=56-bits AES-128=128-bits
- **Security through Obscurity** – Hiding things rather than properly securing them. DO NOT RELY ON OBSCURITY!

Symmetric Algorithms Example Uses

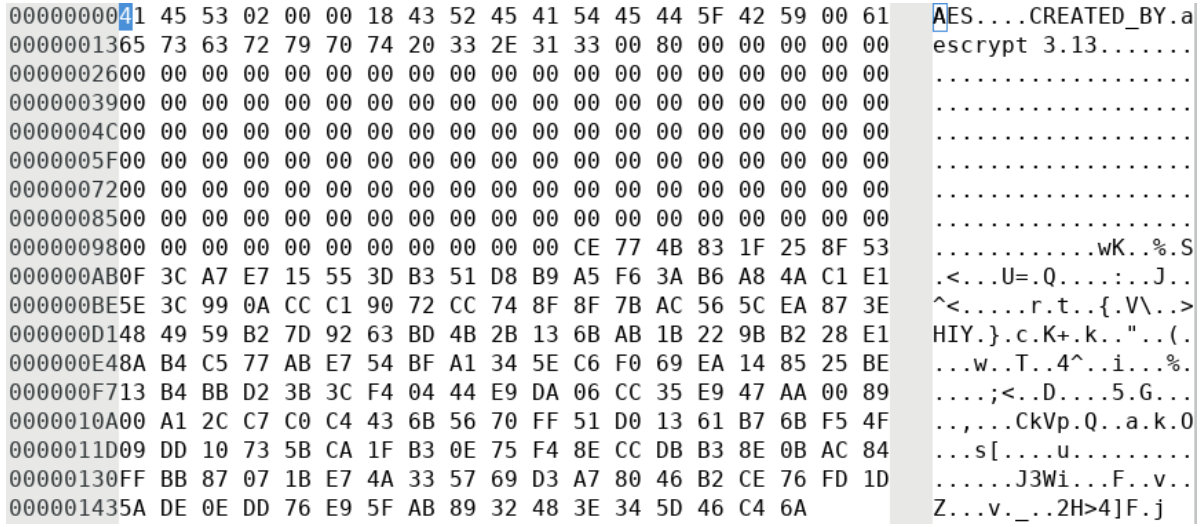
- You want to save a list of passwords for your favorite websites on your computer.
- You want to share data with someone that you can give a password to by another method (phone, in-person, etc.)

Symmetric Algorithms

passwords.txt

Username: khardy
Password: IL0v3Cryp40

Encrypt
Cipher: aes128
Key: MySecret



- The **same key** is used to encrypted AND decrypt.
- Only store and share the key using secure means.
  - What does secure mean? It depends on the situation.
- **Key Exchange** becomes critical. Consider how you will share keys with other parties.

Symmetric Cryptographic Algorithms

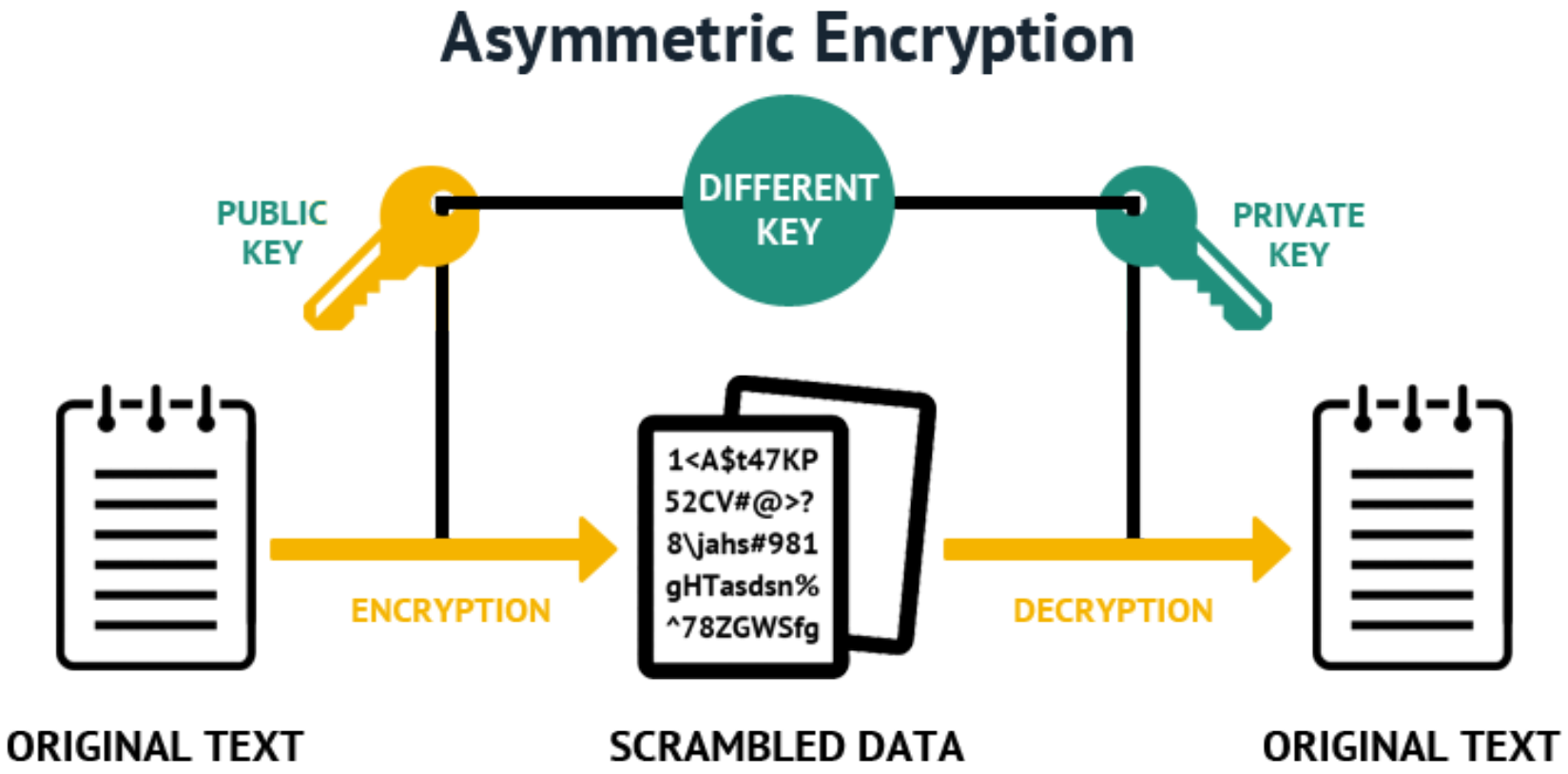
- **DES** – Data Encryption Standard (obsolete, replaced by AES). [Block Cipher]
- **3DES** – Triple DES. Uses DES three times, with 2 or 3 different keys. (multiple encryption) (replaced by AES). [Block Cipher]
- **AES** – Advanced Encryption Standard. The algorithm (Rijndael) won the NIST AES competition in 2000. [Block Cipher]
- **RC4** – Rivest Cipher 4. A very fast stream cipher.
- **Blowfish/Twofish** – Block ciphers. Twofish was an AES finalist.

Symmetric Algorithms in PHP

- sodium_crypto_*()
- mcrypt_encrypt() / mcrypt_encrypt() (**deprecated**)

Asymmetric Algorithms Example Uses

- You want to share data with someone without giving away your secret or getting theirs.
- You want to store things so that only certain people can read them.
- You need to share sensitive data over an insecure channel.


Asymmetric Encryption
PUBLIC KEY
DIFFERENT KEY
PRIVATE KEY
ENCRYPTION
DECRYPTION
1<A$t47KP
52CV#@>?
8\jahs#981
gHTasdsn%
^78ZGWSfg
ORIGINAL TEXT
SCRAMBLED DATA
ORIGINAL TEXT

Asymmetric Algorithms

- Also called **public key cryptography**
- Two keys are used for each party: a **public key** and a **private key**
- The **public key** can be shared
- The **private key** must never be shared
- Often used to both *encrypt/decrypt* (provide confidentiality) and *sign* (provide authentication and non-repudiation)
- **Non-repudiation** – The signer of the message can prove that they signed it.

Asymmetric Encryption (Sending sensitive data to Jim)

passwords.txt

Username: khardy
Password: IL0v3Cryp40

Encrypt
Cipher: PGP
Encryption Key: Jim's *Public* Key
Signing Key: My *Private* Key

passwords.txt.gpg (shown in a hex editor)

```
0000000085 02 0C 03 A7 43 67 DC 1F C7 53 5F 01 0F FF 6A.....Cg...S_...j
00000010B8 0E CE F5 13 65 D6 0A 88 9F 80 F5 7A 4A 63 30.....e......zJc0
000000201D CD 49 AE 29 45 F0 B6 0E 62 BD 2C 97 1F 16 0E..I.)E...b.,....
0000003004 CD 3B C6 80 D2 BC 76 13 F2 25 B9 AD 97 63 42..;....v..%...cB
0000004094 CD 06 83 08 D7 CB C3 B0 36 6F F7 F2 8A 3E EE.........6o...>.
00000050AE D8 AF AB 34 21 B4 DF 4B B6 00 14 C6 71 A2 B0....4!..K....q..
0000006071 42 D3 D6 BD 74 B3 1F 83 F0 94 22 6D 61 8D 3AqB...t....."ma.:
0000007007 B8 6F 83 B9 E9 EA A5 EF 62 7F 84 BF 3F F7 D6..o......b...?..
00000080EC 01 46 68 A1 8B D9 34 50 23 54 89 8C DA 47 7D..Fh...4P#T...G}
000000901A 40 DD 60 32 61 3F A3 83 73 C3 23 A2 66 EF 6E.@.`2a?..s.#.f.n
```


```
0000000085 02 0C 03 A7 43 67 DC 1F C7 53 5F 01 0F FF 6A.....Cg...S_...j
00000010B8 0E CE F5 13 65 D6 0A 88 9F 80 F5 7A 4A 63 30.....e......zJc0
000000201D CD 49 AE 29 45 F0 B6 0E 62 BD 2C 97 1F 16 0E..I.)E...b.,....
0000003004 CD 3B C6 80 D2 BC 76 13 F2 25 B9 AD 97 63 42..;....v..%...cB
0000004094 CD 06 83 08 D7 CB C3 B0 36 6F F7 F2 8A 3E EE.........6o...>.
00000050AE D8 AF AB 34 21 B4 DF 4B B6 00 14 C6 71 A2 B0....4!..K....q..
0000006071 42 D3 D6 BD 74 B3 1F 83 F0 94 22 6D 61 8D 3AqB...t....."ma.:
0000007007 B8 6F 83 B9 E9 EA A5 EF 62 7F 84 BF 3F F7 D6..o......b...?..
00000080EC 01 46 68 A1 8B D9 34 50 23 54 89 8C DA 47 7D..Fh...4P#T...G}
000000901A 40 DD 60 32 61 3F A3 83 73 C3 23 A2 66 EF 6E.@.`2a?..s.#.f.n
```


Decrypt
Cipher: PGP
Verification Key: My *Public* Key
Decryption Key: Jim's *Private* Key

passwords.txt

Username: khardy
Password: IL0v3Cryp40

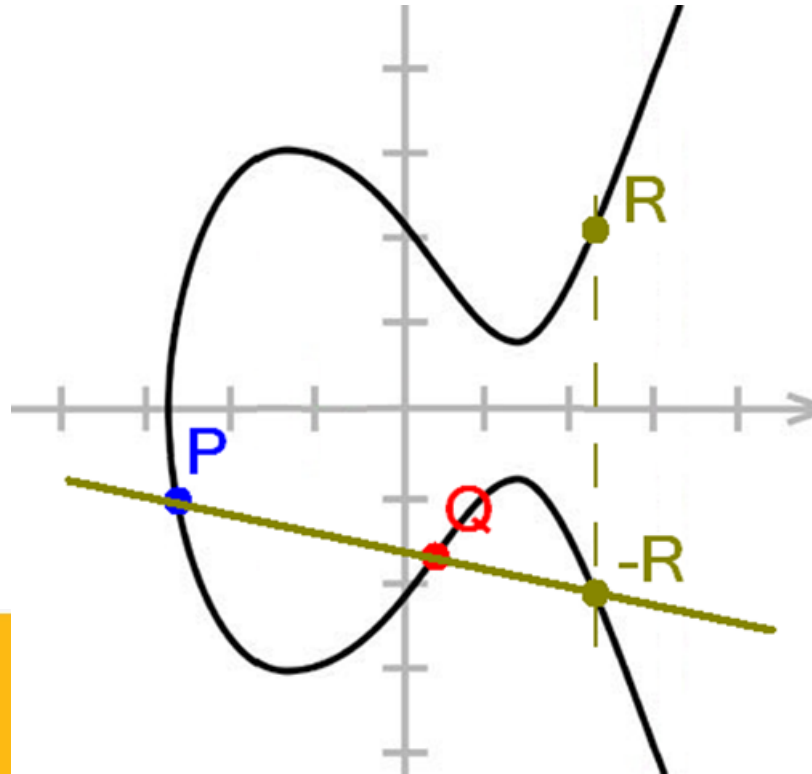
Asymmetric Encryption Algorithms

- Rely on mathematically hard *trapdoor functions*
  - Functions that are easy in one direction, but hard in the other direction
- Example: **Prime Factorization** - Multiplying two prime numbers together is easy. Factoring this large number into its two primes is difficult.
  - What is 5683 x 8803? (it's 5002749)
  - What are the two factors of 14273419? (How would you solve this?)

Asymmetric Encryption Algorithms

Answer: 14273419 = 3109 x 4591

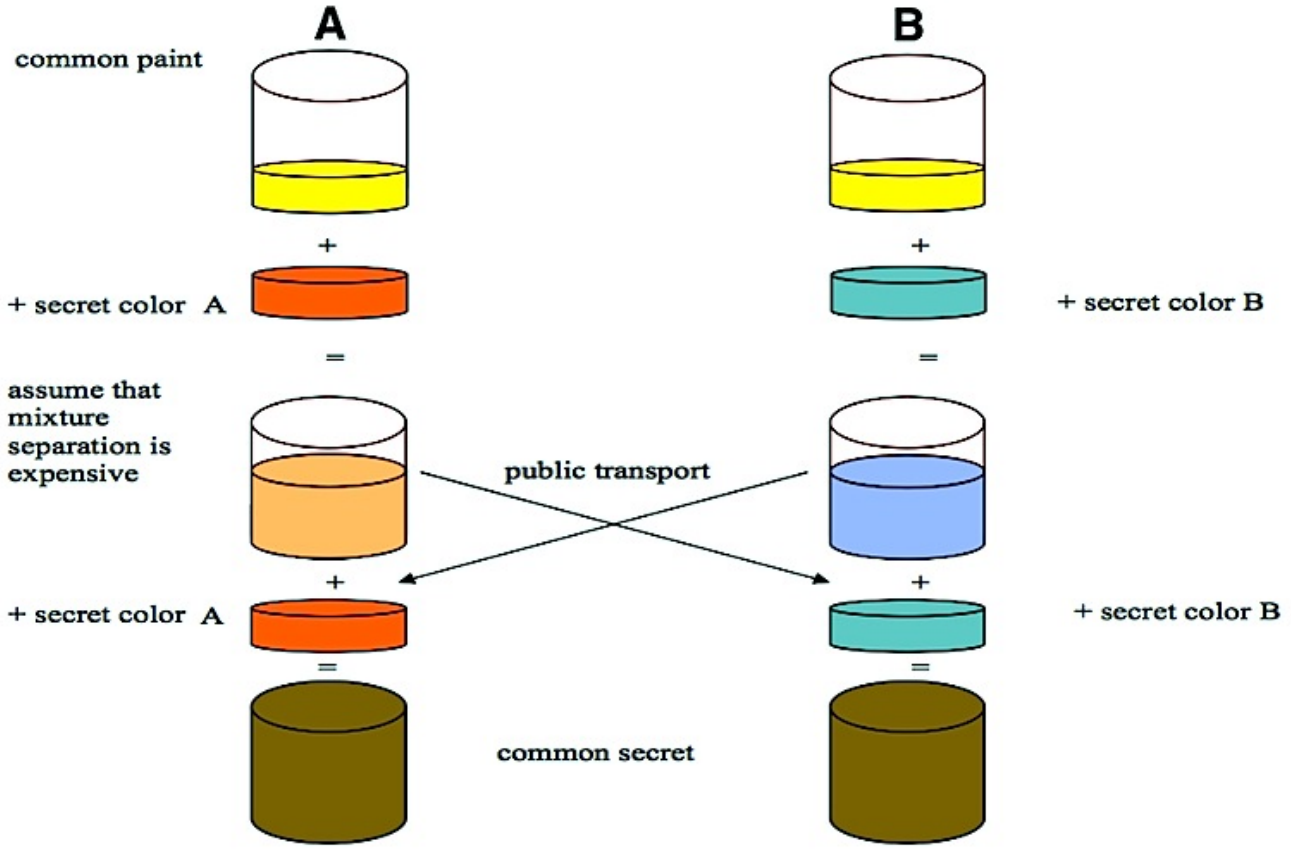
- Another Example: **Elliptic Curve Cryptography (ECC)** – Using a 3rd order polynomial curve, following a line between points is easy. If you only know the end result, reversing the process is hard.
  - P + Q = R



Asymmetric Cryptographic Algorithms

- **RSA** – One of the first public key cryptosystems. Usually used to encrypt a symmetric key, which is used to encrypt data. (*electronic key exchange*)
- **DSA** – Digital Signature Algorithm. Used to sign and authenticate data.
- **Diffie-Hellman** – The gold standard for key exchange. Allows the sharing of a secret key between two people who have not previously contacted each other.

Diffie-Hellman Key Exchange



Symmetric vs. Asymmetric

Symmetric Cipher	Asymmetric Cipher
Faster Good for bulk data All keys must be protected	Slower Good for smaller datasets Easier key management

How about using both Symmetric & Asymmetric ciphers together?

SSL does this using a **Session Key**.

- The data is encrypted with a symmetric cipher using a random symmetric key.
- The asymmetric cipher is used to encrypt a symmetric key.

Asymmetric Algorithms in PHP

- openssl_public_encrypt()
- openssl_public_decrypt()
- openssl_private_encrypt()
- openssl_private_decrypt()
- sodium_crypto_box() / sodium_crypto_box_open()

Weak Algorithms

- Errors in ciphers may create vulnerabilities.
- Ciphers rely on difficult mathematical problems, which may be easier to solve as computers get more powerful.
- As ciphers become weak, disable them in your systems.
- Examples: DES, 3DES, MD5, SHA-1