Research

A study of security standard encryption and hash algorithms.

⚠ Download Research

Project Roadmap



IDEA

Compare encryption algorithms.



ENCRYPTION

Identify encryption algorithms.



OBJECTIVES

Create encyption tests.



Performs tests for data.



DISCOVERIES

Present findings & comparisons.



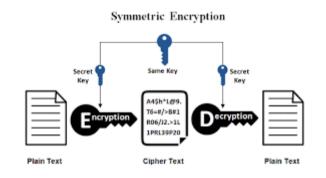
CONCLUSION

Determine the best encryption.

Two Kinds of Encryption

Symmetric, Asymmetric.

- Symmetric Encryption
 - Algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext.
- Asymmetric Encryption
 - A cryptographic system that uses pairs of keys: public keys which may be disseminated widely, and private keys which are known only to the owner.



Symmetrical Key Encryption

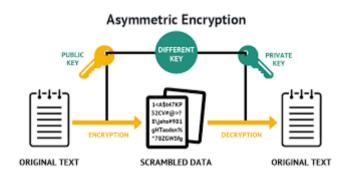
The keys may be identical or there may be a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in

comparison to public-key encryption.

Encryption Stength: Relatively Weak: Single Point of Failure.

Asymmetrical Key Encryption

The generation of shared keys depends on cryptographic algorithms based on mathematical problems to produce one-way functions. Effective security only requires keeping the private key private; the public key can be openly distributed without compromising security.

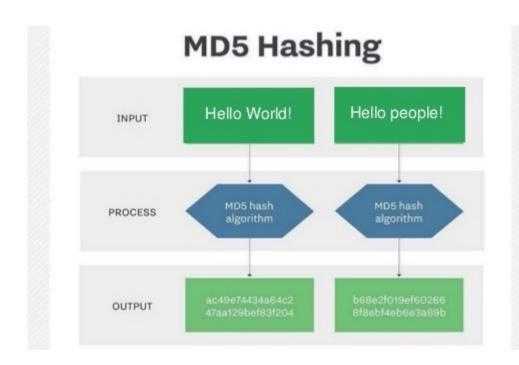


Encryption Stength: Relatively Strong: Still difficult to share private keys in secret, or keep them secret.

Types of Hash Algorithms

Message Digest, Secure Hash.

- Message Digests
 - Cryptographic hash functions containing a string of digits created by a one-way hashing formula. (MD2, MD4, MD5, MD6)
- Secure Hash Algorithms
 - Cryptographic hash functions which takes an input and produces a fixed-bit hash value known as a message digest typically rendered as a hexadecimal number. (SHA-1, SHA-2, SHA-3)

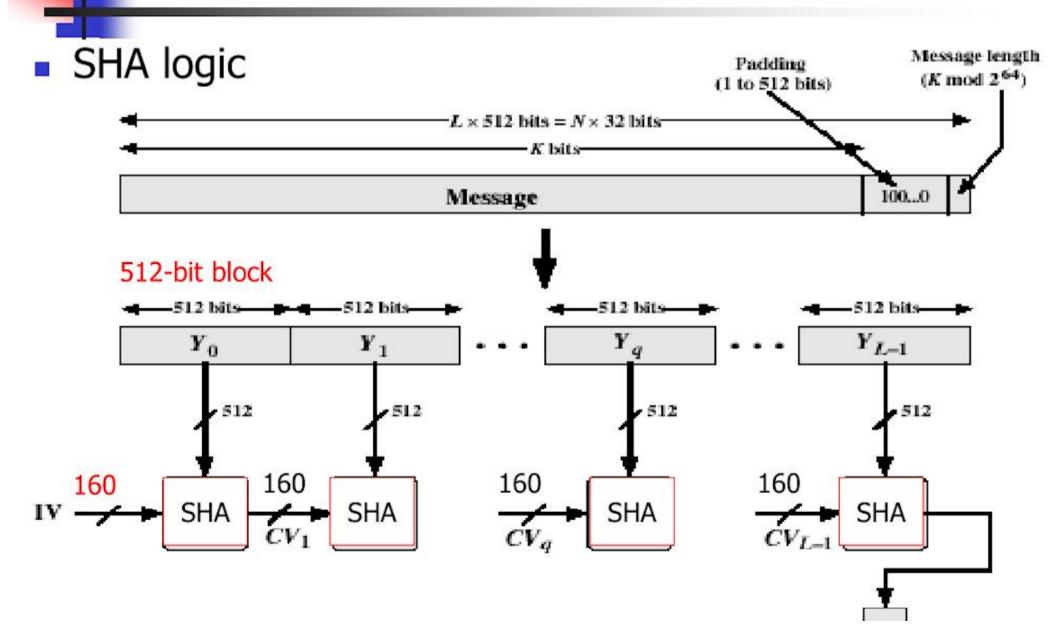


Message Digest

A message digest is a cryptographic hash function containing a string of digits created by a one-way hashing formula. Message digests are designed to protect the integrity of a piece of data or media to detect changes and alterations to any part of a message. They are a type of cryptography utilizing hash values that can warn the copyright owner of any modifications applied to their work.

Encryption Stength: Relatively Weak: Collision Attacks can be carried out in seconds for MD5 and lower.

Secure hash algorithm (SHA)



Secure Hash

A secure hash algorithm is a set of algorithms developed by the NIST along with other government and private parties. These secure encryption or "file check" functions are used to meet the top cybersecurity challenges of the 21st century. A number of public service groups work with federal government agencies to provide better online security standards for organizations and the public.

Encryption Stength: Relatively Strong: Google broke SHA-2 encryption using collision attacks in 2017.

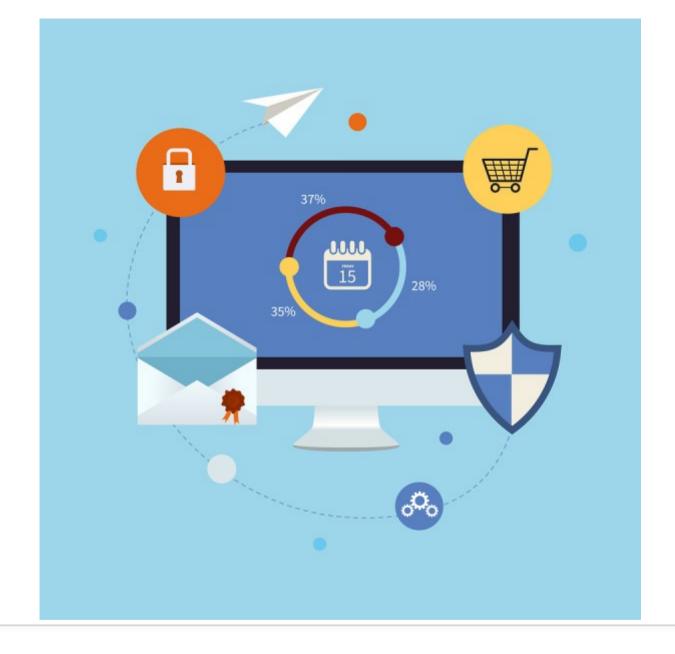
Files on Github.



```
r00t@r00t-KitPloit:~/KitPloit$ uncompyle6 -h
 uncompyle6 [OPTIONS]... [ FILE | DIR]...
uncompyle6 [--help | -h | --V | --version]
  uncompyle6
                       foo.pyc bar.pyc
                                                   # decompile foo.pyc, bar.pyc to stdout
  uncompyle6 -o . foo.pyc bar.pyc # decompile to ./foo.pyc_dis and ./bar.pyc_dis uncompyle6 -o /tmp /usr/lib/python1.5 # decompile whole library
Options:
                    output decompiled files to this path: if multiple input files are decompiled, the common prefix
  -o <path>
                    is stripped from these names and the remainder appended to
                      uncompyle6 -o /tmp bla/fasel.pyc bla/foo.pyc

-> /tmp/fasel.pyc_dis, /tmp/foo.pyc_dis

uncompyle6 -o /tmp bla/fasel.pyc bar/foo.pyc
                      -> /tmp/bla/fasel.pyc_dis, /tmp/bar/foo.pyc_dis
uncompyle6 -o /tmp /usr/lib/python1.5
-> /tmp/smtplib.pyc_dis ... /tmp/lib-tk/FixTk.pyc_dis
   --compile | -c <python-file>
                    attempts a decompilation after compiling <python-file>
                    print timestamps
  -p <integer> use <integer> number of processes
                    recurse directories looking for .pyc and .pyo files
                   use fragments deparser
  --fragments
                    compare generated source with input byte-code
  --verify-run compile generated source, run it and check exit code
  --syntax-verify compile generated source
--linemaps generated line number correspondencies between byte-code
                    and generated source output
  --encoding <encoding> use <encoding> in generated source according to pep-0263
  --help
                    show this message
 ebugging Options:
  --asm
                 -a include byte-code
                                                      (disables --verify)
  --grammar
                      show matching grammar
  --tree
                      include syntax tree (disables --verify add template rules to --tree when possible
                                                     (disables --verify)
  --tree++
Extensions of generated files:
   .pyc_dis' '.pyo_dis'
+ '_unverified'
                                successfully decompiled (and verified if --verify) successfully decompile but --verify failed
     + ' failed'
                                 decompile failed (contact author for enhancement)
 r00t@r00t-KitPloit:~/KitPloit$ ∏
    KitPloit : bash
IMPLEMENTATION
```



CONCLUSION

Closing Remarks.

There are only two kinds of companies:

- 1. Those that have been hacked.
- 2. Those that will be.
- ~ Robert Mueller

Best,

Brandon Rowe, Kyle Batson, and Josh Howard.