Cryptography: Techniques for Secure Communication

Eric Henely
KØSMD

Picture is German Enigma Machine

Disclaimer

- "No amateur station shall transmit: …messages encoded for the purpose of obscuring their meaning…" 47 CFR 97.113(a)(4)
- In 2013, Don Rolph, AB1PH, filed a petition that sought to amend the FCC rules to permit encryption during emergency operations and related training exercises. HIPAA compliance was one of the main goals of the request. This petition was denied by the FCC.

Health Insurance Portability and Accountability Act

Why discuss Cryptography at an Amateur Radio club meeting?

- One of the agencies we serve is the hospital (HIPAA)
- We are expected to be expert communicators
  - We need to understand appropriate modes of communication. Ham radio is not appropriate for all information.
  - Many of us use the internet for communications. The majority of the traffic on the internet is now encrypted.
  - When someone else doesn't understand how their internet works or needs their computer fixed, they ask us.
- The techniques involved in cryptography relate closely to encoding techniques for amateur radio digital modes (JT65/FT8)

Health Insurance Portability and Accountability Act

Definitions

- Cryptography:
  - The art of writing or solving codes (Oxford)
  - The enciphering and deciphering of messages in secret code or cipher; the computerized encoding and decoding of information (Merriam-Webster)
- Cryptanalysis:
  - The art or process of deciphering coded messages without being told the key (Google)

We don't care much about cryptanalysis, except we want our stuff to not be susceptible to it.

Why do we need Cryptography?

- CIA model:
  - Confidentiality (encryption)
  - Integrity (digital signatures, HMAC, AEAD)
  - Availability
  - Non-repudiation (digital signatures*)

Network traffic travels unencrypted by default
Hard drives can be stolen
Integrity: want to make sure that malicious user did not alter data in transit
Authenticated Encryption with Associated Data
Hash-based Message Authentication Code
Non-repudiation is a legal term. Author of a statement/document will not be able to challenge the authorship; contract.
Electronic digital signature only indicates that the signer had the private key
In my opinion, true non-repudiation requires witnesses

Classic Cryptography

- Caesar Cipher (key = X or 23)

```
Plaintext:  THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
Ciphertext: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD
```

- Vigenère Cipher (key = LEMON)

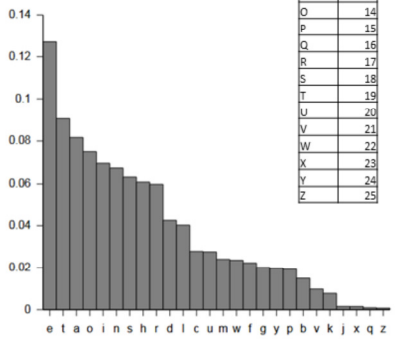
```
Plaintext:  ATTACKATDAWN
Key:        LEMONLEMONLE
Ciphertext: LXFOPVEFRNHR
```

- Letter Frequency
- One-Time Pad

```
Ciphertext: EQNVZ
Key:        XMCKL
Plaintext:  HELLO
```

```
Ciphertext: EQNVZ
Key:        TQURI
Plaintext:  LATER
```

Character	
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8
J	9
K	10
L	11
M	12
N	13
O	14
P	15
Q	16
R	17
S	18
T	19
U	20
V	21
W	22
X	23
Y	24
Z	25



Modular arithmetic

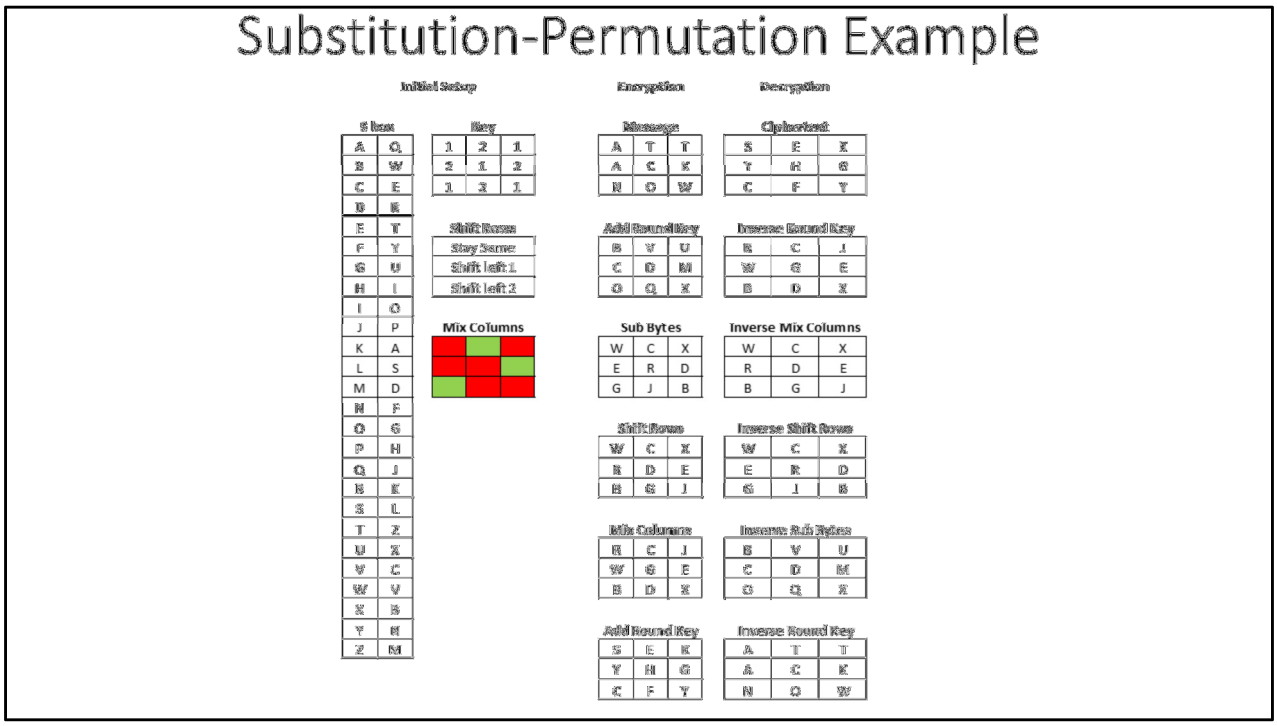
One-Time Pad is provably secure, because each plaintext has equal probability; key is same length as data

Key must be transferred out of band

Symmetric Key Cryptography

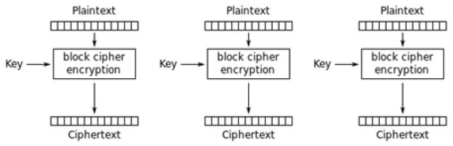
- Same key used to encrypt and decrypt
- Very good performance
- Algorithms are public; the only thing secret is the key
- Key distribution is a problem
- Reasonably resistant to quantum cryptanalysis
- Goals:
  - Confusion: The key does not relate in a simple way to the ciphertext
  - Diffusion: Changing a single bit in the message or key will change half of the bits in the ciphertext on average
- Examples:
  - Feistel Ciphers: DES (56 bit), Triple-DES (112 bit)
  - Substitution-Permutation Ciphers: AES (128, 192, 256 bit)

Box that requires one key to lock or unlock
DES-1978, never broken, 56 bit keys insufficient
Data Encryption Standard
Advanced Encryption Standard



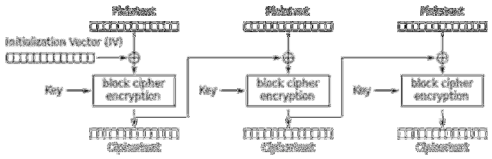
Made this algorithm up to work similar to AES
AES works on binary data
Need to pad message to fill the last block
S box selection is very important, NSA modified DES sbox in 1978.
Mix columns in AES uses an invertible linear transformation based on polynomial multiplication over a Finite/Galois Field, but the calculation does not change so this can be represented by a lookup table in a practical implementation
In AES, these steps are done 10 times for AES128, 12 times for AES192 and 14 times for AES256
AES also uses a “Key Expansion” function to mix up the keys for each round
26^9 possible keys = 5 trillion possible keys, or about 42 bits of security
Single CPU could brute force through all possible keys in about 20 minutes, assuming one attempt per clock cycle

Block Cipher Modes



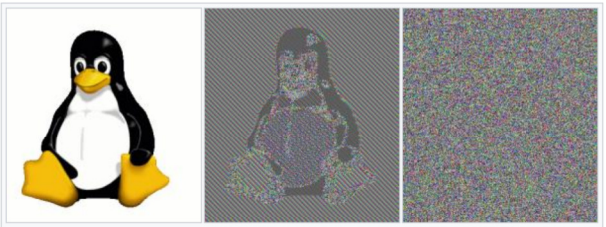
Electronic Codebook (ECB) mode encryption

By WhiteTimberwolf (SVG version) - PNG version, Public Domain, https://commons.wikimedia.org/w/index.php?curid=26434116



Cipher Block Chaining (CBC) mode encryption

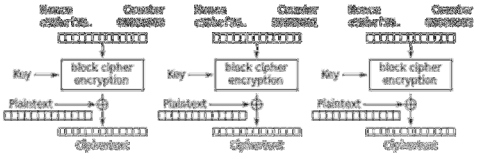
By WhiteTimberwolf (SVG version) - PNG version, Public Domain, https://commons.wikimedia.org/w/index.php?curid=26434096



Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

The image on the right is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the image on the right does not ensure that the image has been securely encrypted; many kinds of insecure encryption have been developed which would produce output just as "random-looking".

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation



Counter (CTR) mode encryption

By WhiteTimberwolf (SVG version) - Gwenda (PNG version), PNG version, Public Domain, https://commons.wikimedia.org/w/index.php?curid=26434109

With block ciphers such as AES, patterns in data longer than 1 block will be easy to spot

Public Key Cryptography

- Much slower than symmetric key cryptography
  - Generally used to exchange symmetric keys or for digital signatures
- Usually based on computationally hard problems that are easy to verify
- Algorithms are public; the only thing secret is the key
- Keys are generally much larger than symmetric key algorithms for similar level of security
- Uses one key to encrypt and a second related key to decrypt
- Partially solves key distribution problems of symmetric key cryptography (Certificate Authorities)
- Existing algorithms are highly vulnerable to quantum computing
- Examples:
  - Diffie-Hellman (DHE)
  - Rivest-Shamir-Adleman (RSA)
  - Elliptic Curve Cryptography (ECC)

Box that required one key to lock, but a separate key to unlock
Protecting private key is very important

Rivest-Shamir-Adleman (RSA)

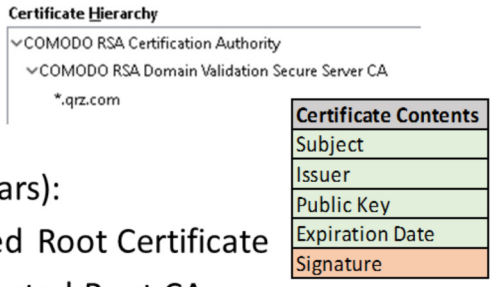
- Security is based on difficulty of factoring the product of two large prime numbers
- Current recommended key size is 2048 bits or greater (~617 digits)
- Encryption: the public key (e) of the recipient is used by the sender to encrypt and the recipient uses their private key (d) to decrypt
- Digital Signatures: the private key (d) of the signer is used to sign and the public key of the signer (e) is used to verify the signature
- 768 bit key has been broken

RSA Example

Initial Setup			Encryption	Decryption
p	Prime Number	3	Message (m) = 18	Ciphertext (c) = 24
q	Prime Number	11	Ciphertext (c) = (m^e) mod n	Message (m) = (c^e) mod n
n	Product of two large primes	33		
phi(n)	(p-1)(q-1)	20	18^3 mod 33	24^7 mod 33
e	Public exponent. Chosen such that 1 < e < phi(n), e and phi(n) are coprime	3	18^3 = 5832	24^7 = 4,586,471,424
d	Private exponent. Calculated such that e*d = 1 mod phi(n)	7	5832 mod 33 = 24	4,586,471,424 mod 33 = 18
(e,n)	Public Key	(3,33)	c = 24	m = 18
(d,n)	Private Key	(7,33)		

Primes are not completely tested, probable primes are determined using Miller-Rabin primality test
Math behind proving this works and is secure is somewhat complicated
Safe prime is (2p+1) where p is also prime
A prime number *q* is a *strong* prime if $q + 1$ and $q - 1$ both have some large prime factors
e is usually chosen to be 65537 (2^16 +1)
Can be computed efficiently using exponentiation by squaring
Small messages are bad: 0,1
For signatures, messages is hashed before being signed
Message must be padded (OAEP = Optimal Assymetric Encryption Padding)
For digital signatures, e and d are flipped
Attack using general number field sieve

Protocol Example: HTTPS



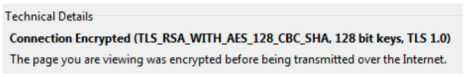
These steps are done once (every couple years):

1. Certificate Authority generates self-signed Root Certificate
2. CA petitions browsers for inclusion as trusted Root CA
3. CA generates Intermediate Certificate and signs it with Private Key of Root Certificate using RSA
4. QRZ generates public key (e), private key (d) and n (product of d*e) for RSA
5. QRZ sends (e,n,*.qrz.com) to CA for signature
6. CA signs QRZ's (e,n, *.qrz.com) and returns certificate to QRZ

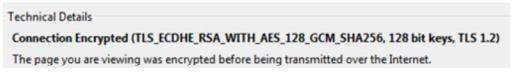
Private key must not be shared with anyone, if so, attacker could create a counterfeit website and/or decrypt traffic

Protocol Example: HTTPS

RHEL5



RHEL7



These steps are done for each TLS Session:

1. Client requests https://www.qrz.com
2. Server sends certificate to client
3. Client validates the certificate chain and verifies that the Root CA is trusted
4. Client generates 128 bit AES Key using good random number generator
5. Client encrypts AES Key using RSA and the server's public RSA key and sends it to the server
6. Server decrypts AES Key sent by client with the server's private RSA key
7. Server encrypts requested page with AES Key and sends result to client
8. Client decrypts page using AES Key
9. Client and Server continue communicating with AES Key until TLS session expires

Third party interaction not required for every communication
Browser should check periodically to see if any certificates in the chain have been revoked
Must revoke certificates where private key is compromised

Key Size

- Number of atoms in the universe is estimated to be between 10^78 and 10^82 (2^259 and 2^272)
- Cryptographers and Physicists believe there is not enough matter and energy in the universe to count through 2^{256} keys; this doesn't even consider testing each key against an encrypted message

AES	
Key Size	Strength
128	126
192	189.9
256	254.3

RSA	
Key Size	Strength
1024	80
2048	112
3072	128
7680	192
15360	256

ECC	
Key Size	Strength
160	80
224	112
256	128
384	192
521	256

Key Size vs Brute Force Time*	
Key Size	Time in Years
64	0.01827945
80	1197.962071
112	5.14521E+12
128	3.37136E+17
192	6.22017E+36
256	1.14742E+56

*Assuming 8000 CPU cores operating at 4 GHz testing one key per clock cycle

Cryptographic Hashes

- Hash Function: any function that can be used to map data of arbitrary size to data of a fixed size
- Cryptographic Hash Function: a special class of hash function that makes it suitable for use in cryptography
- Uses: digital signatures, message authentication, password storage, verifying file integrity, file identifier (svn)
- Ideal Properties:
  - Deterministic: the same message always results in the same hash
  - Computing the hash of any given message is fast
  - A small change in the message changes the hash value so extensively that the new hash value appears uncorrelated to the old hash value
- Security is based on:
  - Pre-image resistance: given hash value h, it should be difficult to find any message m such that h = hash(m) (one-way function)
  - Second pre-image resistance: given an input m, it should be difficult to find an input n such that hash(m) = hash(n)
  - Collision resistance: it should be difficult to find two different messages m and n such that hash(m) = hash(n) (birthday problem)

Internally, many hash functions work similar to block ciphers
A hash function ideally can operate on arbitrary message length, so there are an infinite number of possible messages
Birthday problem: with just 23 people in a room, the probability that two of them share the same birthday is 50%, 70 people = 99.9%
Birthday problem is approximated by the square root function
Sqrt(2^128) = 2^64

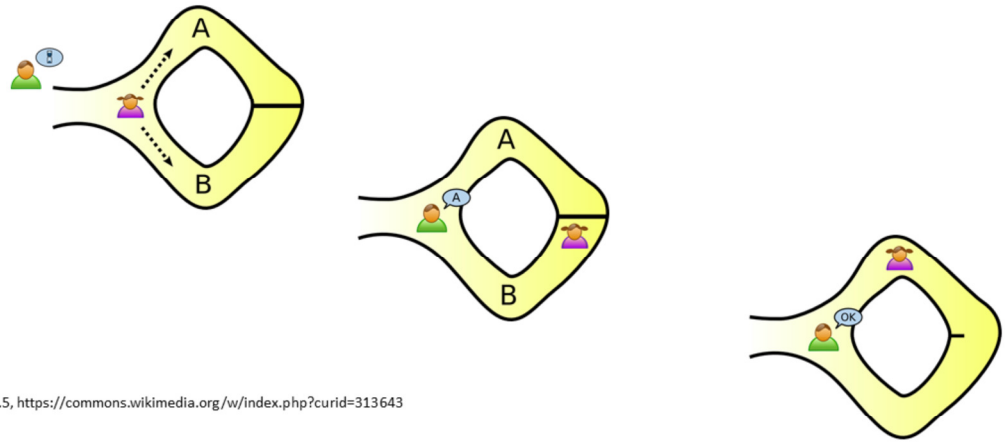
Cryptographic Hash Examples

Function	Output length	Theoretical Strength	Year	Function Status
MD5	128 bits	64 bits	1992	Collisions found
SHA1	160 bits	80 bits	1995	Collisions considered imminent
SHA2-256	256 bits	128 bits	2002	Safe
SHA2-512	512 bits	256 bits	2002	Safe
SHA3-256	256 bits	128 bits	2008	Safe
SHA3-512	512 bits	256 bits	2008	Safe

Message	Hash Function	Hash Value (represented in hexadecimal)
Alice owes Mallory $100	MD5	e1f2b91211360211e027882d26307b9c
Alice owes Mallory $1000	MD5	760dd119b0e1233e1fb9b26eb9a87d20
Alice owes Mallory $100	SHA1	57da6cc4fa9e9719aab4db5e2b761f098778fe30
Alice owes Mallory $1000	SHA1	6b6b13a3ae90e1ef65a355e43f9767c7d9a9c171
Alice owes Mallory $100	SHA2-256	3d345207504592566ecfa6529a711d8fb5c90c1b64e7301d1aa8c7ba59415a0f
Alice owes Mallory $1000	SHA2-256	a45474595457eed34e1f0614bca2ffcb8d47b916e8127620c0699e3a3f617403

A message is like a document or a file

Zero Knowledge Proofs



CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=313643

CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=313645

CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=313648

Computationally, this is done with problems that are NP-Complete
Hamitonian Path through a large graph

Password Storage Goals

1. Verifying a password should be fast to minimize load on the verifying server.
2. Passwords should not be stored in plaintext since plaintext passwords are easy to exploit if the password file is stolen. This is usually accomplished by storing the cryptographic hash of the password.
3. If two users choose the same password, the stored values for each user should be different. This makes it so that an attacker must attack the hash of each password individually. This is accomplished by appending a "Salt" to the password before hashing it.
4. Attacking the salted hash of a password should be computationally expensive to deter brute force attacks. This is done by iterative hashing. (conflicts with 1)
5. Ideally, the password should never be sent to a server in plaintext. This can be accomplished by using a Zero Knowledge Password Proof.

Online attack is easy to mitigate with lockout function or by throttling login attempts
Biggest concern is an offline attack, which happens if the attacker gets direct access to the password database
Salts should be considered public.
The salt should be large enough so that no two users share the same salt.
Secure Remote Password Protocol

Password Storage Example

Username	Password	Salt	Function	Stored in PW DB
user1	password1	salt1	SHA1(salt1 + password1)	salt1, c184a598f71e00f343b3a451f593259854bbcb6fa
user2	password1	salt2	SHA1(salt2 + password1)	salt2, bda17c8a186806c1007e14a9f3530ef26922a230
user3	password2	salt3	PBKDF2(HMAC-SHA1, password2, salt3, 1000, 32)	salt3, e32bf16cd43f7b175148d7c1adf382c37d8fd466c6c4d3ec32931432752e363d
user4	password2	salt4	PBKDF2(HMAC-SHA1, password2, salt4, 1000, 32)	salt4, ac807d792f52f576ab08a032438895f9ba5f308b6ab664a2dd47739917492ad8

In this context, the + symbol indicates concatenation

PBKDF2(Hash Function, Password, Salt, Number of Iterations, Number of Bytes in Output)

Notice that 1 digit change in the salt completely changes the resultant hash
Bcrypt/Scrypt
Recommended salt size is at least 64 bits

Post-Quantum Cryptography

- Shor's algorithm reduces the complexity to break RSA to polynomial time
- A quantum computer theoretically reduces the strength of AES to half the number of bits in the key
- NIST/NSA want new algorithms that are resistant to attacks by quantum computers

Final Remarks

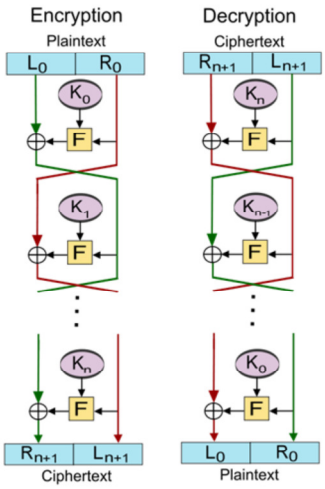
- Do not encrypt your amateur radio communications
- Do not write your own cryptographic algorithms
- Implementations are often a bigger source of security issues than the algorithms themselves
- It is difficult or impossible to prove the security of most ciphers (except one-time pad)
- For sensitive applications, I recommend using NIST approved algorithms and implementations (NSA Suite B, FIPS 140-2) since these have been rigorously tested by the cryptographic community

Cryptography is applied statistics, we are trying to make the likelihood of our data being decrypted meet chosen statistical criteria with the least amount of computational work
Passing a statistical randomness test does not necessarily mean that the data is cryptographically random
Saying that something is encrypted is not enough
What algorithm was used?
What implementation was used?
Who has the key?
How is it stored?
Snowden Leaks, ECC weak curve

Questions?

Data Encryption Standard (DES)

- Developed by IBM in the 70s
- Feistel Cipher
- 56 bit keys
- Triple DES



By Feistel_cipher_diagram.svg: Amirkiderivative work: Amirki (talk) - Feistel_cipher_diagram.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=16419258