TTKS0600, Encryption Techniques and Systems, Lecture assignment 2

Lehosvuo Timo

Bordi Tuukka

1 Review questions

1.1 Execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.

1.2 In diffusion each plaintext digit effects the value of many ciphertext digits which dissipates the statistics of the ciphertext but in confusion discovering the relationships between the statistics and the value of the encryption key is made as complex as possible in the cipher. Main difference is that in diffusion the statistical association between the plaintext and ciphertext is made as complex as possible and in confusion the correlation between the statistics of the ciphertext and the value of the encryption key is made as complex as possible

1.3 Substitution: Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.

Permutation: A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

1.4 Block size
–Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm
Key size
–Larger key size means greater security but may decrease encryption/decryption speeds
Number of rounds
–The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security.
–In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack
Subkey generation algorithm
–Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis
Round function F
–Greater complexity generally means greater resistance to cryptanalysis
Fast software encryption/decryption

–In many cases, encrypting is embedded in applications or utility functions in such a way as to preclude a hardware implementation; accordingly, the speed of execution of the algorithm becomes a concern

Ease of analysis

–If the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength

1.5 1) Security
2) Cost
3) Algorithm and Implementation Characteristics.

1.6 1. General Security
2. Software implementations
3. Restricted-Space environments
4. Hardware Implementations
5. Attacks on Implementations
6. Encryption vs. Decryption
7. Key Agility
8. Other Versatility and Flexibility
9. Potential for Instruction-Level Parallelism

1.7 AES need two separate software or firmware modules for applications that require both encryption and decryption whereas in equivalent inverse cipher you only need one but you also need to change the key schedule

1.8 The use of three stages of encryption with three different keys

1.9 It is an attack used against a double encryption algorithm and requires a known (plaintext, ciphertext) pair.

The attack proceeds as follows:

- First, encrypt for all possible values of K1.
- Store these results in a table and then sort the table by the values of X.
- Next, decrypt C using all 256 possible values of K2.
- As each decryption is produced, check the result against the table for a match.
- If a match occurs, then test the two resulting keys against a new known plaintext–ciphertextpair.
- If the two keys produce the correct ciphertext, accept them as the correct keys.

1.10

The stream cipher is similar to the one-time pad, difference is that a one-time pad uses a genuine random number stream, whereas a stream cipher uses a pseudorandom number stream.

1. The encryption sequence should have a large period. A pseudorandom number generator uses a function that produces a deterministic stream of bits that eventually repeats. The longer the period of repeat the more difficult it will be to do cryptanalysis.

2. The keystream should approximate the properties of a true random number stream as close as possible. For example, there should be an approximately equal number of 1s and 0s. If the keystream is treated as a stream of bytes, then all of the 256 possible byte values should appear approximately equally often. The more random-appearing the keystream is, the more randomized the ciphertextis, making cryptanalysis more difficult.

3. To guard against brute-force attacks, the key needs to be sufficiently long. The same considerations that apply to block ciphers are valid here. Thus, with current technology, a key length of at least 128 bits is desirable.

With a properly designed pseudorandom number generator, a stream cipher can be as secure as a block cipher of comparable key length. A potential advantage of a stream cipher is that stream ciphers that do not use block ciphers as a building block are typically faster and use far less code than do block ciphers. You cannot reuse keys with stream cipher.If two plaintexts are encrypted with the same key using a stream cipher, then cryptanalysis is quite simple. If the two ciphertextstreams are XORedtogether, the result is the XOR of the original plaintexts. If the plaintexts are text strings, credit card numbers, or other byte streams with known properties, then cryptanalysis may be successful

2 Simplified DES

Ciphertext string: 10100010
key: 0111111101

Calculating keys 1 and 2

Key 1
bits used in p10: 0111111101
p10 result: 1111110011

LS-1 inputs: 11111 and 10011
LS-1 results: 11111 and 00111

bits used in p8: 1111100111
p8 result: 01011111 → key 1

Key 2
LS-2 inputs: 11111 and 00111
LS2-2 results: 11111 and 11100
bits used in p8: 1111111100
p8 result: 11111100 → key 2

Decryption

Ciphertext string: 10100010
Key 1: 01011111
Key 2: 11111100

IP
ip input (ciphertext string): 10100010
ip result: 00110001

L: 0011
R: 0001
SK (key 2): 11111100

E/P input (R): 0001
E/P result: 10000010
XOR input (E/P and SK): 10000010 and 11111100
XOR result: 01111110

s0 input (first 4-bits from XOR): 0111
s0 array

row/column numbers	**0**	**1**	**2**	**3**
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

result for s0 row : decimal (1)
result for s0 column: decimal (3)
s0 binary result (row 1 column 3): 00

s1 input (last 4-bits from XOR): 1110
s1 array

row/column numbers	**0**	**1**	**2**	**3**
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

result for s1 row: decima l(2)
result for s1 column: decimal (3)
s1 binary result (row 2 column 3): 00

s0 + s1 result (s0 and s1 binary): 0000
p4 input (s0 + s1): 0000
p4 result: 0000

XOR input (L, p4 result): 0011 and 0000
XOR result: 0011

Fk_1
fk_1 input (XOR result, R): 0011 and 0001
fk_1 result: 00110001
fk_1 result separated: 0011 and 0001

SW
SW (fk_1 result): 00010011

Fk_2

L: 0001
R: 0011
SK (key 1): 01011111

E/P input (R): 0011
E/P result: 10010110
XOR input (EP and SK): 10010110 and 01011111
XOR result: 11001001

s0 input (first 4-bits from XOR): 1100
s0 array

row/column numbers	**0**	**1**	**2**	**3**
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

result for s0 row: decimal (2)
result for s0 column: decimal (2)
s0 binary result (row 2 column 2): 01

s1 input (last 4-bits from XOR): 1001
s1 array

row/column numbers	**0**	**1**	**2**	**3**
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

result for s1 row: decimal (3)
result for s1 column: decimal (0)
s1 binary result (row 3 column 0): 10

s0 + s1 result (s0 and s1 binary): 0110
p4 input (s0 + s1): 0110
p4 result: 1010

XOR (L, p4 result): 0001 and 1010
XOR result 1011
fk_2(XOR result, R): 1011 and 0011
fk_2 result: 10110011

IP^{-1}

ip^{-1} input (fk_2 result): 10110011
ip^{-1} result: 11101010

first 4 bits: 1110 → translates to the letter O
second 4 bits: 1010 → translates to the letter K

decrypted plaintext: OK

3 Modes of Operation

Encrypting images with ECB is no secure since you can clearly see text and the outline of the penguin

Figure 1: ECB encryption

Figure 2: ECB encryption

whereas using CBC the image is encrypted properly

Figure 3: CBC encryption (penguin)

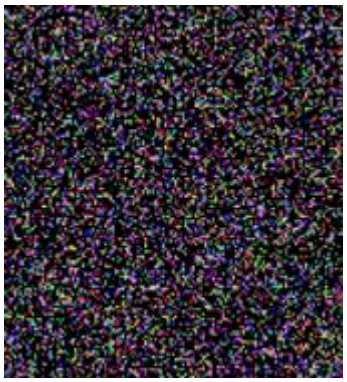

Figure 4: CBC encryption (TOP SECRET)

4 Random bit generator

Python random bit generator

- Python has random() function which I would not use for cryptographic purposes since it uses Mersenne Twister as the core generator. Mersenne Twister is too deterministic which means that if the attacker observes sufficient number of iterations the attacker can predict future iterations

Java random bit generator

- java.util.Random is a linear congruential pseudorandom number generator. Linear congruential pseudorandom number generators are considered too deterministic to fulfill any security-critical purposes just like python random() function.

C random bit generator

- C has a rand() function for creating random numbers, but it is neither random or secure, as it creates the same "random" numbers in the same sequence every time it is ran. There's a more secure version around - srand() - which takes a random seed to determine a starting point, but since the numbers are still generated in the same sequence every time, it's hardly an improvement

5 File encryption

	VeraCrypt	TrueCrypt	CyberSafe Top Secret	Dm-crypt / LUKS	Ccrypt
Ciphers	AES, Camellia, Kuznyechik, Serpent, Twofish, AES-Twofish, AES-Twofish-Serpent, Camellia-Kuznyechik, Camellia-Serpent, Kuznyechik-AES, Kuznyechik-Serpent-Camellia, Kuznyechik-Twofish-Serpent-AES, Serpent-Twofish-AES, Twofish-Serpent	AES, Serpent Twofish, AES-Twofish, AES-Twofish-Serpent, Serpent-AES, Serpent-Twofish-AES and Twofish-Serpent.	AES, 3DES, RSA, ГОСТ, BlowFish. Also uses libraries OpenSSL, OpenPGP and Crypto-Pro	AES, Twofish, cast5, cast6 and their various combinations	Rijndael with 256-bit blocks
Supported OS	Windows XP or higher, Windows SERVER 2003 to 2016 (except 2012 R2),	Windows XP or higher, Linux, OS X	Windows NT, Android	Windows NT, Windows Mobile, Linux, DragonFly BSD	Windows, Linux, Mac OS X, FreeBSD, OpenBSD, Sun Solaris,

	Mac OS X 10.7 Lion or higher, Linux x86, FreeBSD x86 and Raspberry Pi OS				Minix (i386), Android 4+, AIX, Linux for Alpha, Sun Solaris (i386), NetBSD, HP-UX
Usability	Easy to use, very clear instructions, user friendly. Graphical User interface	Old interface bit difficult, new one is simpler. Graphical User Interface	Simple to use. Maybe bit oldish. Graphical User interface. could not find any updates after 2013, so I am not sure if this is supported anymore	Easy to use if you use the default options. It also provides more specific options for people who know what to do. Command-line tool. Has Graphical User Interfaces as addons	very simple and easy to use command-line interface.
Features	Hidden containers, Pre-boot authentication (windows only), Multiple keys, Passphrase strengthening, Hardware acceleration, Filesystems, Two-factor authentication	Hidden containers (one per partition), Pre-boot authentication (Windows only), Custom authentication, Multiple keys, Passphrase strengthening, Hardware acceleration, partitions, Two-factor authentication	Hidden containers, Multiple keys, Passphrase strengthening, Hardware acceleration, Filesystems, Two-factor authentication	Pre-boot Authentication, Custom authentication, Multiple keys, Passphrase strengthening, Hardware Acceleration, TPM (partial support), Two-factor authentication	Only file or stream encryption
Open source	Yes	Yes	No	Yes	Yes
Hash	RIPEMD-160, SHA-256, SHA-512, Whirlpool, Streebog	RIPEMD-160, SHA-512, Whirlpool	Unknown	SHA-1, SHA-256, SHA-512, RIPEMD160,	Only internal hashing of

				WHIRLPOOL (as an add-on)	keys (256 bits)
Modes of operation	XTS	XTS	XTS	ECB, CBC-PLAIN64, CBC-ESSIV:hash, XTS-PLAIN6	CFB
Filesystem	Windows on both MBR and UEFI GPT drives; dynamic drives discouraged	Windows (MBR only)	Only Windows MBR volumes; no UEFI GPT drives, and dynamic drives discouraged	Any OS, any filesystem	Any supported OS

General comments about assignment

- First assignment was simple eventhou copying answers felt weird (you said we can copy the answers in the last lecture 😊)
- Second assignment was hard because the material was hard to understand.
- Third assignment was fun but very difficult (maybe even too hard) if you don't know how to code
- Forth assignment was easy
- Fifth was hard compared to the points it gives since it was hard to decide what to compare and what was enough. Also, it was hard to find information for every little thing. Maybe we over thought this one and simpler answer would be enough
- We did all the assignments together so the work share was 50/50