CHAPTER 8

Key Management

Generating Keys

- The security of an algorithm rest in the key
- If you are using a cryptographically weak process to generate keys, then your whole system is weak.
- In this case the cryptanalyst just cryptanalyze your key generation algorithm rather than your encryption algorithm.

Reduced keyspaces

- DES has a 56-bit key; thus 2^{56} (10^{16}) possible keys.
- However, Norton Discreet for MS-DOS(version 8.0 and earlier) only allows ASCII key, forcing the high order bit of each byte to be zero. The program also converts lowercase letters to uppercase resulting only 2^{40} keys.
- This poor key generation procedures have made its DES ten thousand times easier to break than proper implementation.
- Table 1 shows the number of possible keys with various constraints on the input strings.
- Table 2 shows the time required for an exhaustive search through all of those keys, given a million attempts per second.

Reduced keyspaces Continue…

Table 1

Number of Possible keys of various Keyspaces

	4-Bytes	5-Bytes	6-Bytes	7-Bytes	8-Bytes
Lowercase letters [26]	460,000	$1.2*10^{7}$	$3.1*10^{8}$	$8.0*10^{9}$	$2.1*10^{11}$
Lowercase letters and digits [36]	1,700,000	$6.0*10^{7}$	$2.2*10^{9}$	$7.8*10^{10}$	$2.8*10^{12}$
Alphanumeric characters [62]	$1.5*10^{7}$	$9.2*10^{8}$	$5.7*10^{10}$	$3.5*10^{12}$	$2.2*10^{14}$
Printable characters [95]	$8.1*10^{7}$	$7.7*10^{9}$	$7.4*10^{11}$	$7.0*10^{13}$	$6.6*10^{15}$
ASCII characters [128]	$2.7*10^{8}$	$3.4*10^{10}$	$4.4*10^{12}$	$5.6*10^{14}$	$7.2*10^{16}$
8-bit ASCII characters [256]	$4.3*10^{9}$	$1.1*10^{12}$	$2.8*10^{14}$	$7.2*10^{16}$	$1.8*10^{19}$

Reduced keyspaces Continue…

Table 2

Exhaustive Search of various keyspaces (1 million attempts/s machine)

	4-Bytes	5-Bytes	6-Bytes	7-Bytes	8-Bytes
Lowercase letters [26]	0.5s	12s	5m	2.2h	2.4d
Lowercase letters and digits [36]	1.7s	1m	36m	22h	33d
Alphanumeric characters [62]	15s	15m	16h	41d	6.9y
Printable characters [95]	1.4m	2.1h	8.5d	2.2y	210y
ASCII characters [128]	4.5m	9.5h	51d	18y	2300y
8-bit ASCII characters [256]	1.2h	13d	8.9y	2300y	580,000y

Poor key choice

- When people choose their own keys, they generally choose poor ones.
- They like to choose “Abdullah” rather than “@67dh4&”.
- Because “Abdullah” is easy to remember than “@67dh4&”.

- A smart brute-force attack does not try all possible keys in numerical order; it tries the obvious keys first.
- This is called a dictionary attack, because the attacker uses a dictionary of common keys.

- Daniel Klein was able to crack 40 percent of the passwords on the average computer using this system.

Random keys

- Good keys are random-bit strings generated by some automatic process.
- If the key is 64-bits long, every possible 64-bit key must be equally likely.
- Generate the key bits from either a reliably random source or a cryptographically secure pseudo-random-bit generator.

Pass Phrases

- A better solution is to use an entire phrase instead of a word, and to convert that phrase into a key.
- These phrases are called pass phrases.
- A technique called key crunching converts the easy-to-remember phrases into random keys.
- Use a one-way hash function to transform an arbitrary-length text string into pseudo-random-bit string.
- For example, the easy to remember text string:

  Rajshahi university is the 2nd largest university of Bangladesh, students are over 26000.

  Might crunch into this 64-bit key: 23ht 5t6r yu76 980y

Transferring Keys

- The X9.17 standard specifies two types of keys:
    1. Key encryption keys: which encrypt other keys for distribution.
    2. Data keys: which encrypt message keys.
- This two-tiered key concept is used a lot in the key distribution.

- Another solution to the distribution problem splits the key into several different parts and sends each of those parts over a different channel.
- One part could be sent over the telephone, one by mail, one by overnight delivery service, one by carrier pigeon, and so on.

Transferring Keys Continue…

- Alice sends Bob the key-encryption key securely, either by a face to face meeting or the splitting technique just discussed.

- Once Alice and Bob both have the key encryption key, Alice can send Bob daily data keys over the same communications channel.
- Alice encrypts each data key with the key encryption key.

- Since the amount of traffic being encrypted with the key encryption key is low, it does not have to be changed as often.

Key Distribution in Large Network

- Key encryption keys works well in small networks, but can quickly get cumbersome if the networks become large.
- Since every pair of users must exchanges keys, the total no. of key exchanges required in an n-person network is n(n-1)/2.
- Example: for 6 person required 15 key exchanges. For 1000 person required nearly 500,000 key exchanges.
- In these cases, creating a central server makes the operation much more efficient.

Verifying Keys

- When Bob receives a key, how does he know it came from Alice and not from someone pretending to be Alice?
- If Alice gives it to him when they are face-to-face, its easy.
- If Alice sends it via a trusted courier, then Bob has to trust the courier.

- If the key is encrypted with the key encryption key, then Bob has to trust the fact that only Alice has the key.
- If Alice uses a digital signature protocol to sign the key, Bob has to trust the public key database when he verifies that signature.

- Bob could also verifies Alice’s key over the telephone.
- If it’s a public key, he can safely recite it in public.
- If it’s a private key, he can use a one way hash function to verify the key.

Updating Keys

- If you want to change key daily, an easier way is to generate a new key from the old key; this is sometimes called key updating.
- All it takes is a one-way function.
- Key updating works, but remember that the new key is only as secure as the old was.
- If Eve managed to get her hands on the old key, she can perform the updating function herself.

Storing Keys

- Users can either directly enter the 64-bit key or enter the key as a longer character string. The system then generates a 64-bit key from the character string using a key crunching technique.

- Another solution is to store the key in a magnetic stripe card, plastic key with embedded ROM chip, or smart card.
- User can then enter the key by inserting the physical token into a special reader in his encryption box or attached to the computer terminal.

- Hard to remember key can be stored in encrypted form, using something similar to a key-encryption key. For example, an RSA private key could be encrypted with a DES key and stored on disk. To recover the RSA key, the user has to type in the DES key to a decryption program.

Lifetime of Keys

- No encryption key should be used for an indefinite period. There are several reasons for this.

- The longer a key is used, the greater the chance that it will be compromised.
- The longer a key is used, the greater the loss if the key is compromised.
- The longer a key is used. The greater the temptation for someone to spend the effort necessary to break it.
- It is generally more easier to do cryptanalysis with more ciphertext encrypted with the same key.

Destroying Keys

- Old key must be destroyed.
- If the key is written on paper, the paper should be shredded or burned.
- If the key is in a hardware EEPROM, the key should be overwritten multiple times.
- If the key is in a hardware EPROM or PROM, the chip should be smashed into tiny bits and scattered to the four winds.

- If the key is stored on a computer disk, the actual bits of the storage should be overwritten multiple times or the disk should be shredded.

Public Key Management

- Alice can get Bob’s public key several ways.
  1. She can get it from Bob.
  2. She can get it from centralized database.
  3. She can get from her own private database.