Symmetric Key Encryption

Encryption

- Encryption - transforming plaintext into ciphertext using an algorithm (cipher)
- The plaintext is the information we want to protect (maintain confidentiality.)
- The ciphertext is unreadable unless you pocess a key.

Confidentiality and Encryption

- Encryption algorithms depend on the secrecy of a numerical key used with a well known algorithm.
- Decryption with the key is simple, without the key is difficult.

Symmetric v Asymmetric Encryption

- With symmetric key encryption, the same key is used to encrypt and decrypt the data.
- With asymmetric key encryption, different keys are used to encrypt and decrypt the data.

Symmetric-Key Encryption

- The same key is used to encrypt and decrypt the data.
- Computationally efficient.
- Depends on the secrecy of the key.
- Also known as Secret Key Encryption.

Uses of Symmetric Key Encryption

- Transmission over an insecure channel.
- Secure storage on insecure media.
- Authentication
- Integrity Checks

Transmission over an Insecure Channel.

- Encrypt at one end.
- Decrypt at the other.
- Evesdroppers will only see unintelligible data.

Secure Storage on Insecure Media.

- Store the encrypted data.
- And don’t lose the key.

Authentication

- [Authentication is normally associated with asymmetric key encryption, but can be also done with symmetric key encryption.]
- It depends on being able to prove you know a secret without revealing it.
- Alice sends Bob a random (a challenge.)
- Bob encrypts it and send back the encrypted value which Alice checks.
- This authenticates Bob.

Integrity Checks

- An ordinary checksum (CRC) protects against accidental corruption of a message.
- But not against an attacker.
- A secret key is combined with the data before calculating the hash for the data.
- If an attacker changes data, they will not be able to change the checksum.

Some More Detail

Cipher

- An algorithm for performing encryption or decryption.
- Usually depends on a key.

Symmetric Ciphers: definition

- A cipher defined over (K, M, C) is a pair of "efficient" algorithms (E, D) where
  - K - space of keys
  - M - space of messages
  - C - space of ciphers
  - E - Encryption algorithm
  - D - Decryption algorithm

Symmetric Ciphers: definition

- E: K x M -> C (E: key + message to cipertext)
- D: K x C -> M (D: key + ciphertext to message)
- D(k, E(k, m)) = m
  - When we encrypt and then decrypt, we get the original message back again.

The One Time Pad

- First secure cipher.
- $M = C = \{0,1\}^n$, $K = \{0,1\}^n$
  - The message space and the cipher space is the set of binary values of length n.
  - Keyspace is the same as the message space.
- Key is as long as the message.
- $c = k \oplus m$
- $m = k \oplus c$

The One Time Pad

- $D(k, E(k, m)) = D(k, (k \oplus m)) = k \oplus (k \oplus m)$
  $= (k \oplus k) \oplus m = 0 \oplus m = m$
- [xor ($\oplus$) is associative]
- So the encryption and decryption work properly.
- What is the value of k given m and c?
- $c = k \oplus m$
- $c \oplus m = (k \oplus m) \oplus m = k \oplus (m \oplus m) = k \oplus 0 = k$
- Ans: $k = c \oplus m$
- So if we know a message and the corresponding ciphertext we can easily obtain the key.

The One Time Pad

- Because of the fact that the key (one time pad) is obtainable from a single message and the corresponding ciphertext, a one time pad should only be used once.
- Hence the name.

The One Time Pad

- Very fast encrypting and decrypting.
- But long keys (as long as the message text).
- And keys can only be used once.
- This makes it impractical.

Perfect Secrecy

$$\forall m_0, m_1 \in M \left(len\left(m_0\right) = len\left(m_1\right)\right) \text{and} \forall c \in C$$
$$P\left[E\left(k, m_0\right) = c\right] = P\left[E\left(k, m_1\right) = c\right]$$
$$k \text{ is a random key from } K$$

- This says that if c is found, then it is equally likely to have come from m_0 and m_1.
- The most powerful adversary learns nothing about m from c.
- [It gives a formal way of analysing encryption algorithms for perfect secrecy.]

The One-time Pad has Perfect Secrecy

- It turns out that it is relatively easy to show that a one-time pad has perfect secrecy, which is good.
- However it is impractical as seen above, which is bad.
- What is required is an approximation of the one-time pad which is practical.
- [It turns out that it will not have perfect secrecy but when used properly is good enough]

Stream Cipher

Stream Cipher

- Replace a random key (one-time pad) with a pseudo-random key.
- PRNG – Pseudo Random Number Generator
- Generates a sequence of numbers whose properties approximate the properties of sequences of random numbers.
- Maps a small seed space (s = 128) to sequences of pseudo-random numbers.

Stream Cipher

- Use the PRNG to expand out a key to the size of the message.
- XOR the message with this pseudo-random number.
- As before
  - $c = E(k, m)) = k \oplus m$
  - $m = D(k, c) = k \oplus c$
- A stream cipher does not have perfect secrecy.

Predictable

- A pseudo random number (stream) is predictable if there is some i so if that we know the first i bits of the stream we can find the rest of the stream.
- The cipher would be unsafe.
- For example all SMTP messages start with “From:”
- XOR this with the cipher text to get the first 5 bytes of the key.
- Use this to generate the rest of the key (stream).

Issues with Stream Ciphers

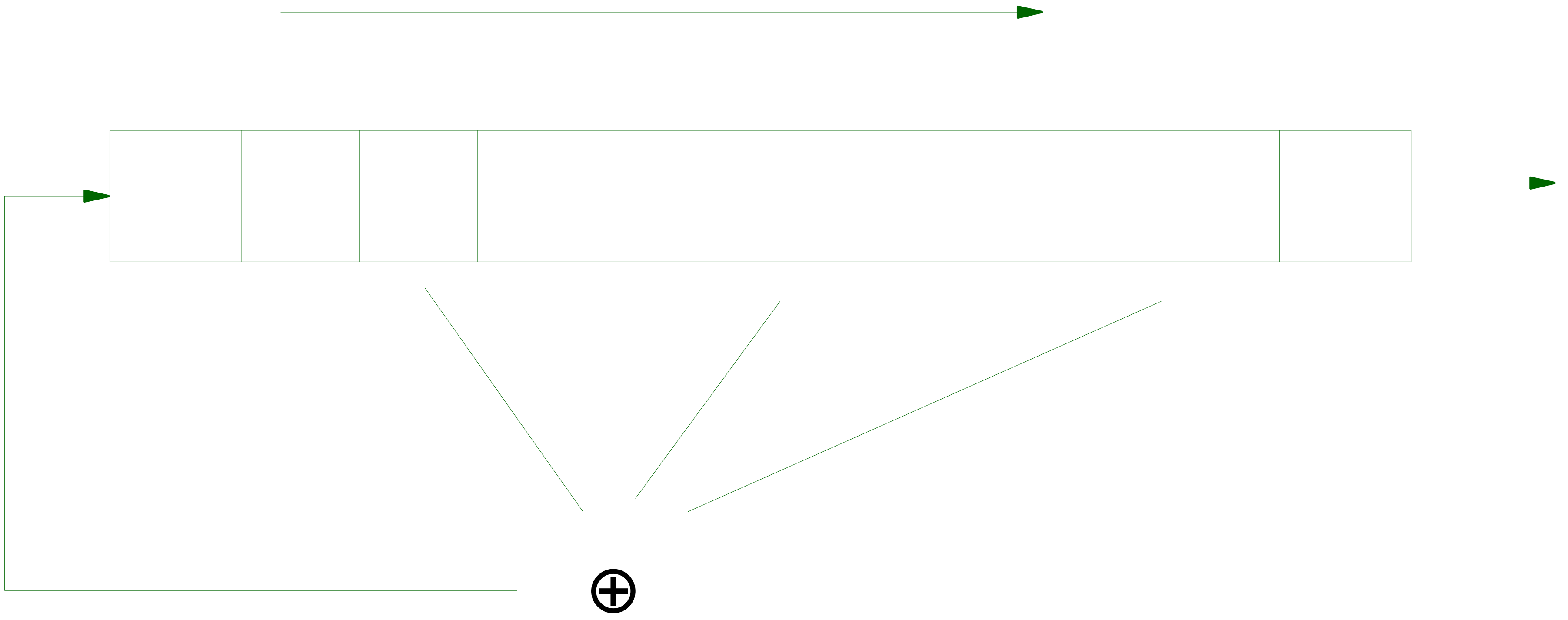
Issues with Stream Ciphers

- Weak PRNGs
  - Linear congruential generator
  - gclib random()
- Weak Stream Ciphers (RC4 (1987))
  - Was used in HTTPS
  - Used in WEP (wrongly)

Content Scrambling Systems

- For encrypting DVDs.
- Badly broken.
- Hardware stream cipher (designed to be easily implemented in h/w.
- Uses a linear feedback shift register (hardware) to generate random numbers.
- “A LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value.”
- The bit positions that affect the next state are called the taps.

Linear Feedback Shift Register.



LFSR based Stream Ciphers

- DVD encryption (CSS):   2 LFSRs
- GSM encryption (A5/1,2):   3 LFSRs
- Bluetooth (E0):   4 LFSRs
- (All broken)
- All implemented in hardware!!

Modern Stream CIphers

Modern stream ciphers

- eStream project, 2008
  - Salsa20 - designed for both software and hardware implementations.
  - ChaCha (similar)
- Often used where implementation is harware is required.
- And used when input size is unknown (and often less that a block size in Block Ciphers)
- Possibly not as widely used as Block Ciphers.

Block Ciphers

Block Cipher

- A symmetric key cipher operating on fixed-length groups of bits, called blocks.
- n = block size
- 3DES: n= 64 bits, k = 168 bits
- AES: n=128 bits, k = 128, 192, 256 bits
- AES: the larger the key, the more secure it is, but the slower it is.

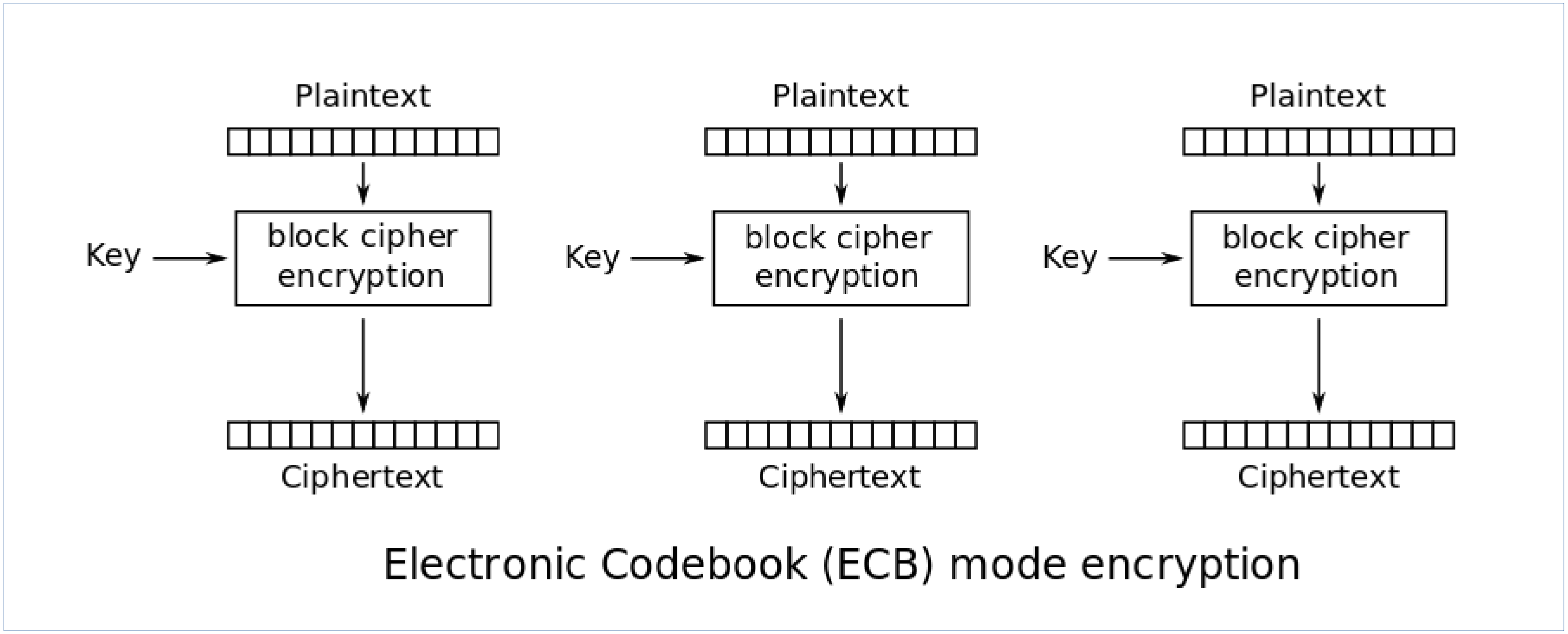
Block Cipher (cont.)

- Block Ciphers can be used to encrypt data larger that a block, by splitting the data into blocks (padding the last block) and encrypting each block in turn.
- The repetition of applying the same key to repeated blocks of text would be a point of attack for cryptanalysis.
- This can only be done safely when combined with suitable "modes of operation".

Modes of Operation

- Suitable "Modes of Operation" randomize the procedure of applying the block cipher to each block of data.
- Electronic Code Book (ECB) is a mode of operation that applies the block cipher to each block in turn and is not suitable.
- Cipher-block chaining (CBC) - Each block of plaintext is XOR'ed with the previous block of ciphertext before applying teh cipher.
- A random initialization vector is XOR'ed with the first block.

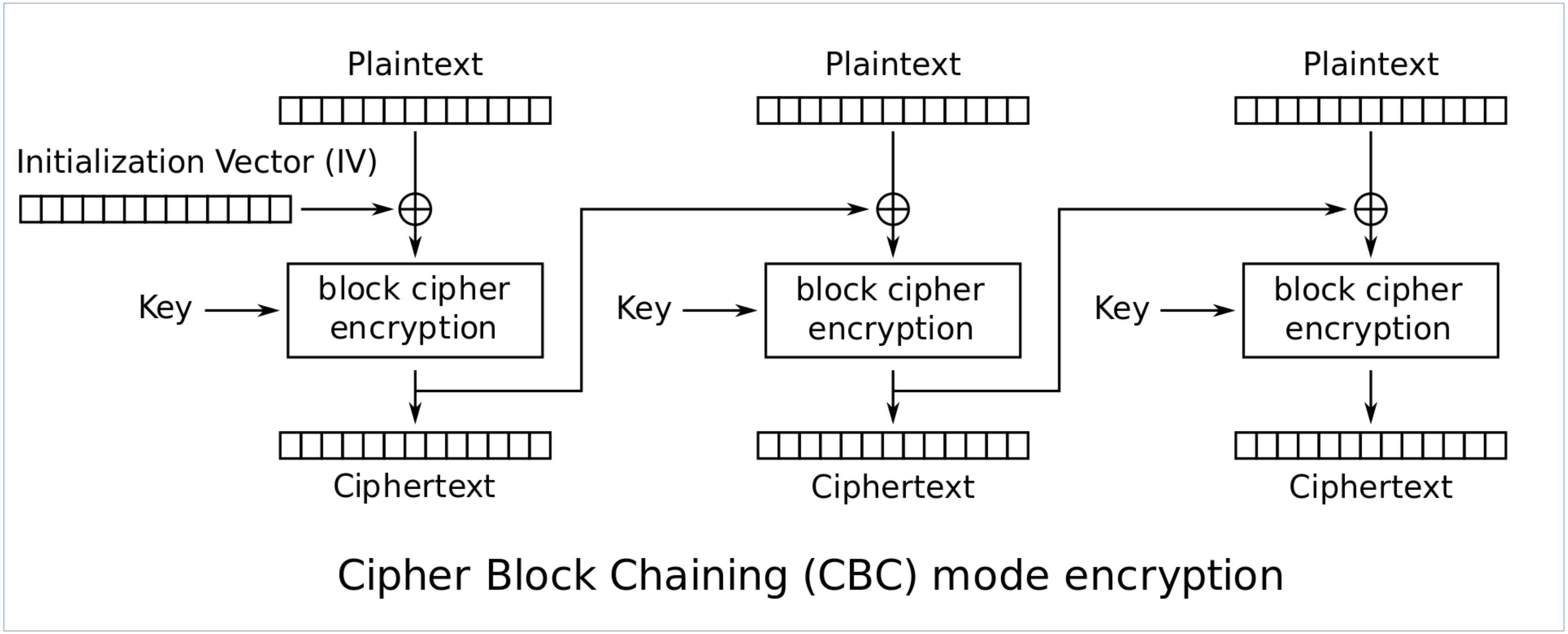
Electronic Code Book (Not Suitable)



Cipher-block chaining (CBC)

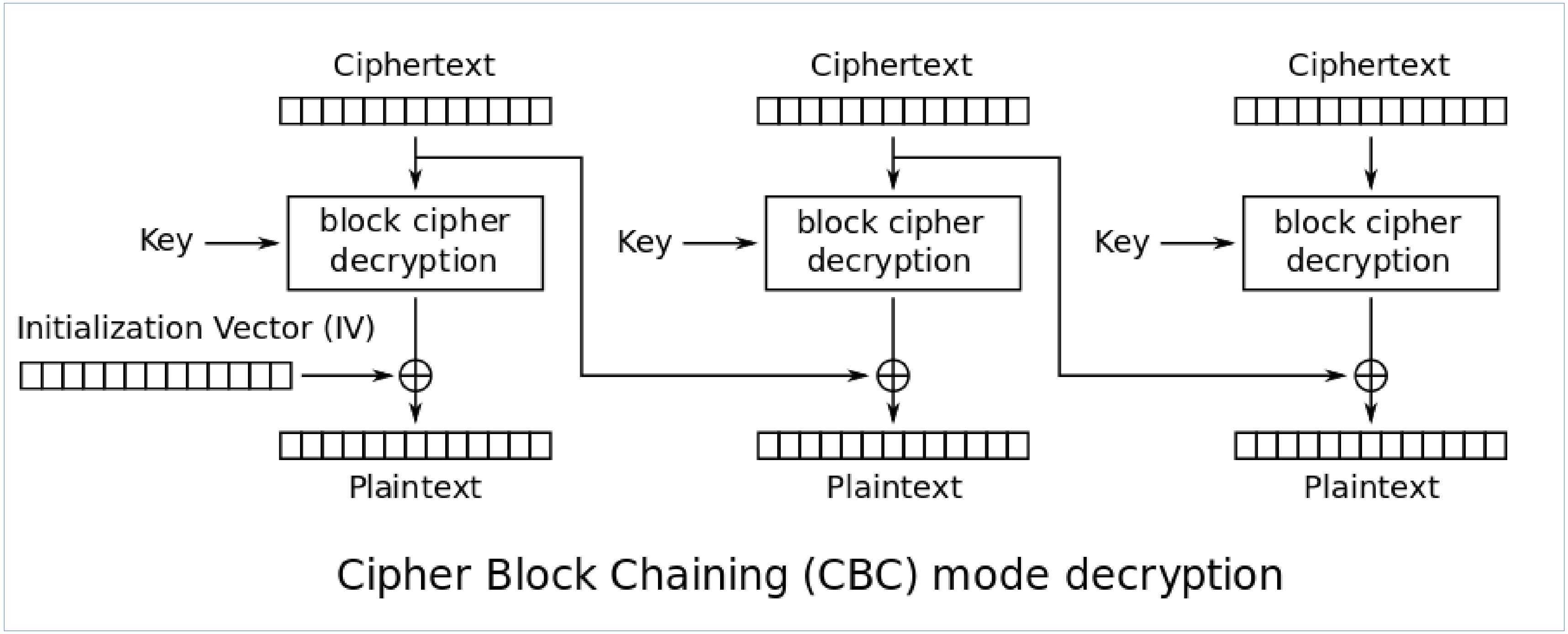
- Each block of plaintext is XOR'ed with the previous block of ciphertext.
- An initialization vector is XOR'ed with the first block.
- [The repetition of applying the same key to repeated blocks of text would be a point of attack for cryptanalysis.]

Cipher Block Chaining - Encryption



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

Cipher Block Chaining - Decryption



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

CBC

- p = plaintext, c = ciphertext
- Consider first block
- $c = E(p \oplus IV, key)$
- $D(c, key) \oplus IV = D(E(p \oplus IV, key)) \oplus IV$
  $= (p \oplus IV) \oplus IV = p$

Examples of Block Ciphers

- AES (Advanced Encryption Standard)
- Blowfish,
- DES, Triple DES, Serpent, Twofish

DES

- Data Encryption Standard - 1977 (IBM)
- Block size – 64 bits
- Symmetric-key algorithm that uses a 56-bit key (now too short).
- Suspicions initially about the existence of an NSA backdoor.
- Broken in 1999 in 22 hours 15 minutes
- Susceptible to brute-force attack.

Triple DES

- Block size – 64 bits
- Key size - 168, 112 or 56
- In fact uses three DES keys, K1, K2 and K3, each of 56 bits

AES

- Advanced Encryption Standard (1998)
- Originally called Rijndael
- Successor to DES
- Selected in AES contest in 2001 [National Institute of Standards and Technology(NIST)]
- Block size - 128 bits
- Key sizes - 128, 192 or 256 bits

Blowfish

- 1993, alternative to DES
- Block size - 64
- Key length - varies from 1 bit up to 448 bits
- No cryptanalysis has been found to date.
- Not as widely used as AES.

Serpent

- Serpent
  - Second in the Advanced Encryption Standard (AES) contest.
- Twofish
  - Based on Blowfish.
  - Another finalist in AES contest.

Symmetric Key Encryption

Java

Encryption – DES Example

```
String ALGORITHM = "AES" ;
KeyGenerator keygen = KeyGenerator.getInstance(ALGORITHM);
SecretKey key = keygen.generateKey();
Cipher eCipher = Cipher.getInstance(ALGORITHM);

// Initialize the cipher for encryption
eCipher.init(Cipher.ENCRYPT_MODE, key);

String s = "This is just an example";
System.out.println("Clear text: " + s);
byte[] cleartext = s.getBytes();

// Encrypt the cleartext
byte[] ciphertext = eCipher.doFinal(cleartext);
System.out.println("Cipher text: " + HexUtil.toString(ciphertext));
```

Encryption – DES Example

```
/////////////////////////////////////////////////
// Decrypt

Cipher dCipher = Cipher.getInstance(ALGORITHM);
dCipher.init(Cipher.DECRYPT_MODE, key);

// Decrypt the ciphertext
byte[] clearText1 = dCipher.doFinal(ciphertext);

String text = new String(clearText1);
System.out.println("Decrypted text: " + text);
```

Encryption – AES, BlowFish

- Similar to above.
- Use Strings AES and Blowfish.

java.crypto.SealedObject

- SealedObject(Serializable object, Cipher c)
  - Constructs a SealedObject from any Serializable object.
  - Cipher should be initialized for encryption
- Object getObject(Cipher c)
  - Retrieves the original (encapsulated) object.
  - Cipher should be initialized for decryption