EE5453 Mid-Term Fall 2022 – James Ryan

1

How does CBC mode solve the security issue with ECB mode?

Answer

There are multiple security issues with ECB mode. The most prominent is a confidientiality issue, where an attacker can infer information about the plaintext from the ciphertext.

Specifically, ECB mode ciphertexts are periodic, with a period equal to the least common multiple of the periodicity of the plaintext and the cipher block size.

CBC mode addresses this security issue by introducing a time-variant dependency in the ciphertext. The dependency is such that each ciphertext block is dependent on the previous ciphertext block. Even if the plaintext is periodic, the ciphertext will not be, because it also depends on the *number of periods* before a given ciphertext block.

2

What is Kerkcoff's principle?

Answer

Kerckoff's principle is a principle for designing cryptosystems. It is the opposite of 'security by obscurity'.

Kerckoff's principle states that a cryptosystem's security should depend only on the secrecy of a 'key'. This is intended to prevent security claims resulting from secrecy of an algorithm or from inaccessability of software.

3

Suppose a key space of 2^{30}. Alice decides to encrypt a plaintext block 6 times with 6 different keys. What is the effective key space for Trudy?

Answer

There are two ways to interpret the 'key space of 2^{30}'.

The way I will interpret it, the 'key space' refers to the key space of the underlying cipher Alice composes to build her cryptosystem. A reasonable assumption for this cipher is that it is a block cipher with a block size of 30 bits, and a key size of 30 bits. Additionally, it is also important to assume that the block cipher composes securely. Under these (reasonable) assumptions, the *effective* key space for Trudy is $2^{30} \times 3$. This is because Trudy can perform a *Meet In The Middle* attack as discussed in class, and so Alice will not have a keyspace of $2^{30} \times 6$.

Another interpretation could be that 'key space of 2^{30}' refers to the key space of the cryptosystem Alice builds. This could occur if the block cipher does not compose securely, i.e., if multiple encryption with two keys was equivalent to single encryption with a different key. This non-secure composeability is present in classical ciphers such as fixed-length XOR ciphers, or in simple substitution ciphers.

4

Consider a computer that can conduct a brute force search with a speed of 2^{40} keys per second. In the worst case, how many seconds does it take to search through a key space of 2^{80}?

Answer

The computer will take

$$\frac{2^{80} \text{ keys}}{2^{40} \text{ keys/second}} = 2^{40} \text{ seconds}$$

to find the correct key (in the worst case).

5

How does a public-key cryptosystem such as RSA address the limitation of symmetric key cryptosystem such as AES?

Answer

There are multiple limitations of a symmetric key cryptosystem, not all of which are addressed by public-key cryptosystems.

The limitation that is addressed is the need in a symmetric key cryptosystem to share secrets. Because symmetric keys need to be kept secret it is difficult to distribute them with out a pre-existing secure channel, which if it existed could be used to communicate securely without the symmetric key cryptosystem!

Public-key cryptosystems address this by constructing two keys, a *public* key which can be publicly shared, and a *private* key which must be kept secret. To communicate securely, Alice and Bob need only each others public keys.

6

DES numerology: 64 bit block length, 56 bit key length and 16 rounds. How many rounds in total are executed to encrypt the ASCII message "ABCDEFGHIJKLMNOPQRSTUVWX" (excluding quotes) in ECB mode? in CBC mode? Explain briefly. Hint: An ASCII character is of size 8 bits.

Answer

The ASCII message consists of 24 8-bit characters, or 3 DES blocks. Both ECB and CBC require full blocks of input, so both ECB and CBC will perform 3×16 rounds to encrypt the plaintext.

7

Consider the one-time pad encryption/decryption scheme where "Attack location XYZ" is the message, K is the key and C is the corresponding ciphertext. Trudy, through her intelligence sources, finds out that Alice was sending the encrypted version of the message "Attack location XYZ". She intercepts the ciphertext C and replaces it with C' such that the recipient decrypts to get the message "Attack location PQR". What is the value of C'? Explain.

Answer

Using the XOR implementation of a one-time pad, C' is given by

$$C' = M' \oplus M \oplus C,$$

where M' is the modified message 'Attack location PQR', M the original message 'Attack location XYZ', and C the ciphertext Alice sent.

In other implementations of the one-time pad, the same technique will work, performing the known (because of Kerckoff's principle!) OTP operation on the ciphertext, the desired modified ciphertext, and the known plaintext.

8

Using a figure, explain how a MAC can be computed in 3DES.

Answer

I will describe the method discussed in class.

We can compute a MAC by operating 3DES in CBC mode. Let the message be P, and break the message into blocks P_i of the same size as a 3DES block (64 bits). Let $E(K, \cdot)$ be 3DES configured with a key K, and $E(K, P_i)$ be the operation of encrypting message block P_i with $E(K, \cdot)$ to produce ciphertext block C_i. Then the MAC T is given by:

$$T = C_n$$
$$T = E(K, P_n \oplus C_{n-1})$$

where

$$C_0 = E(K, P_0 \oplus IV)$$

where IV is a nonce ($n$umber used *once*).

This construction is illustrated in the figure below:

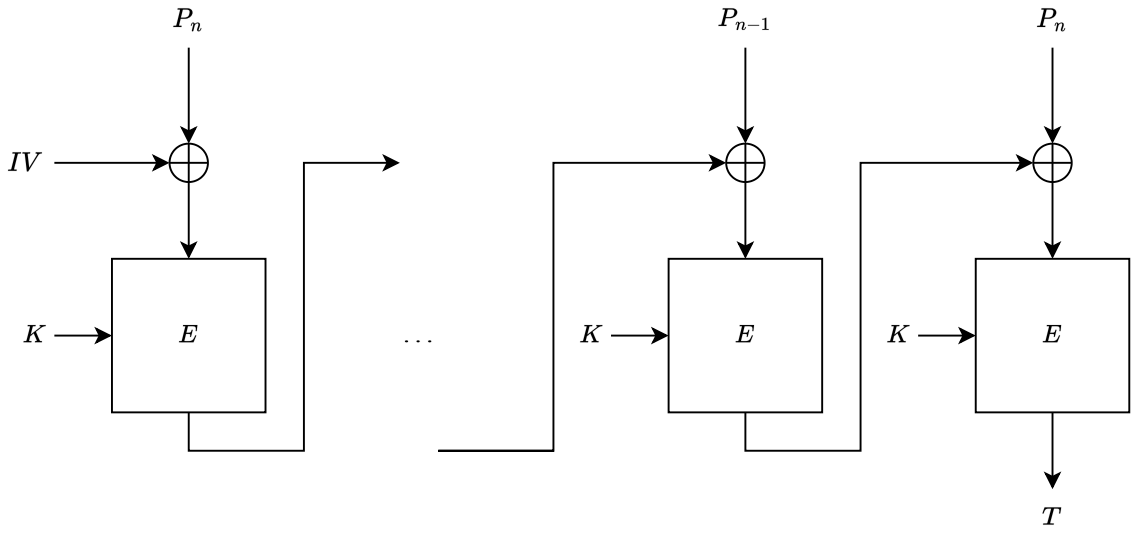


Figure 1: CBC-MAC

9

Suppose Alice and Bob live in a world that only has DES in CBC mode. Assume that Alice and Bob share a key K_{AB}. Using only K_{AB} (i.e., without deriving any other additional key), can Alice send a message to Bob while simultaneously achieving confidentiality and integrity? If yes, explain how. If no, explain why not.

A

Maybe, depending on how 'without deriving any other additional key' is interpreted, and if we play loose with what we mean by 'confidientiality and integrity'.

We can do this using only CBC-MAC, and not more modern modes such as GCM, CCM, EAX, OCB, CCFB, SIV, among others.

To do this, we will split K_{AB} into $K_{AB,e}$ and $K_{AB,m}$ given by:

$$K_{AB,e} = K_{AB}[27:0]$$
$$K_{AB,m} = K_{AB}[55:28]$$

That is, $K_{AB,e}$ is the lower 28 bits of K_{AB}, and $K_{AB,m}$ the upper 28 bits. We can 'expand' these keys by padding with zeros, which may not be considered to be 'deriving other keys'. Then, with two '56-bit' keys, we can use the CBC-MAC construction discussed in class.

Of course, by doing the above we reduce the *effective* keysize of our construction to 28 bits. This small keysize is well within the computational power of a motivated 5th grader, and so we cannot achieve the same security level for either confidentiality or integrity as if we had an effective keysize of 56 bits.

If we allow neither a construction like above nor modern block cipher modes of operation and we are restricted only to DES in CBC mode, then we cannot guarantee both integrity and authenticity.

10

Consider the following DEE variation of 3DES in which you decrypt with K_1, encrypt with K_2 and encrypt with K_1. What is the effective key space assuming a meet-in-the-middle attack? Does it offer the same two major benefits of the original 3DES? Explain.

A

A meet-in-the-middle attack focuses on a point in the multiple encryption cycle that can be reached without using every multiple encryption key. The proposed scheme does not have such a point, and so it's effective keyspace under a MiTM attack is 2^{112}, because it uses a 112-bit key.

The major benefits of the original 3DES are: - 112 bits keyspace - Backwards compatible, since $E(K, D(K, (E(K, \cdot)))) = E(K, \cdot)$.

The DEE scheme has a 112 bit keyspace, and so has the first benefit of classic 3DES. It also has the backwards compatibility benifit, since

$$E(K, E(K, D(K, \cdot))) = E(K, \cdot)$$

11

Give 2 advantages of symmetric key crypto such as AES over public key crypto such as DH. Give 2 advantages of public key crypto over symmetric key crypto.

A

Symmetric & public key cryptosystems are used (in practice) for entirely different things, and have different properties from each other. Therefore, disscussing 'advantages' of one over the other is difficult, as symmetric key cryptosystems cannot do things public key cryptosystems can do and vice versa. Instead, I will discuss some properties one cryptosystem has that the other does not.

Symmetric key cryptosystems are typically much more efficient than public key cryptosystems. They are typically more efficient in speed. For example, according to some publicly available benchmarks, RSA with a modulus of 2048 bits takes $\approx$ 2600000 clock cycles to encrypt 59 bytes on a Zen3 processor. On the same processor, AES128 in CTR mode (a mode of similar speed to ECB) takes $\approx$ 2 cycles per byte for 64 bytes, or around 118 clock cycles for the same 59 bytes. Symmetric key cryptosystems are also typically more efficient in key size. Consider that the NSA considers AES with a key size of 256 bits to be sufficiently secure to protect information up to TOP SECRET. Meanwhile the NSA specify that RSA moduli must have a length of at least 3072 bits to be sufficient to protect information up to TOP SECRET.

Public key cryptosystems, by nature of having public and private keys, can solve problems symmetric key cryptosystems cannot. For example, there is no known construction of a publicly verifiable digital signature algorithm using only symmetric key cryptography. All publicly verifiable digital signature systems use public key cryptography. Additionally, symmetric key crypto has a particular key establishment problem, so that it is infeasible to establish a symmetric key without an existing secure channel. Public key cryptosystems, on the other hand, find their other primary use in key establishment / key exchange mechanisms.

12

Diffie-Hellman works for two parties. Explain clearly using a figure how you can extend Diffie-Hellman to establish a symmetric key between three parties. Number the message exchanges in the figure and explain the contents of each message clearly. Next, extend it such that it works for n parties.

A

The three-party Diffie-Hellman (3PDH) construction is a special case of the n-party Diffie-Hellman (NPDH) construction. 3PDH is described first, and then its generalization for n parties is described.

No security proof is given, but the construction doesn't seem *obviously* insecure. The construction is not efficient, and requires $O(n^2)$ exchanges of and storage of $O(n^2)$ intermediate values (round keys). A more efficient construction exists where two parties establish a shared secret, and then any participating party could incrementally include new parties in the shared secret. This construction would require $O(n)$ exchanges, and $O(1)$ storage for each party. However, that requires a method of determining which two parties to start with. The construction given here does not require a method of determining which parties to start with, but is inferior in other aspects.

3PDH Let the three parties involved be Alice, Bob, and Carol. Suppose also that there is an agreed upon DH field $\mathbb{F}_p^*$ with p a large prime, and an agreed upon generator $g \in \mathbb{F}_p^*$.

Then the 3PDH construction consists of multiple rounds. All communication is assumed to happen over an insecure channel, as no symmetric key is available to all parties.

In the 0th round, each of Alice, Bob, and Carol will *secretly* and *securely* select an appropriate secret $s \in \mathbb{F}_p^*$. Call Alice's secret a, Bob's secret b, and Carol's secret c. They will also compute $A_0 \equiv g^a \bmod p$, $B_0 \equiv g^b$, and $C_0 \equiv g^c \bmod p$ respectively. Call these *round keys*.

This is the end of the 0th round.

In the 1st round, Alice will send her round key (A_0) to Bob and Carol, and receive from Bob and Carol their round keys B_0 and C_0 respectively. Bob will do similarly, and so will Carol. Each of them will then compute two new round keys.

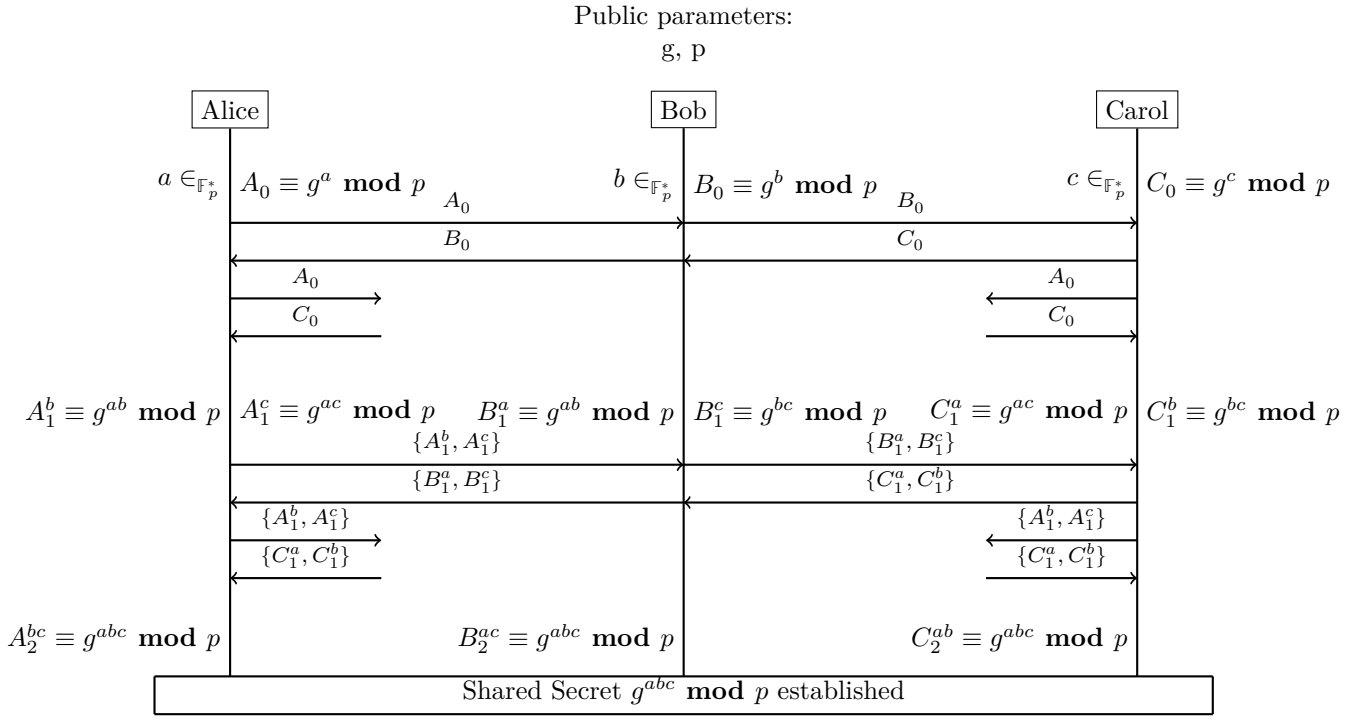
Alice will compute her round keys as $A_1^b = (B_0)^a \equiv g^{ab} \bmod p$ and $A_1^c = (C_0)^a \equiv g^{ac} \bmod p$. Bob and Carol will again do likewise with their respective round keys.

This is the end of the 1st round.

In the 2nd round, Alice will send *both* of her round keys to *both* Bob and Carol. Bob already has $A_1^b \equiv B_1^a \equiv g^{ab} \bmod p$ because he computed it in the 1st round, and he now has $A_1^c \equiv C_1^a \equiv g^{ac} \bmod p$ also. Carol likewise already has $A_1^c \equiv C_1^a \equiv g^{ac} \bmod p$, and she now also has $A_1^c \equiv B_1^a \equiv g^{ab} \bmod p$. Bob and Carol will mimic Alice again, sending *both* of their round keys to the other two participants. Each of Alice, Bob, and Carol can now compute the action of their secret on the *new* round key they acquired this round. So Alice will compute $A_2^{bc} \equiv (g^{bc})^a \equiv g^{abc} \bmod p$, Bob will compute $B_2^{ac} \equiv (g^{ac})^b \equiv g^{abc} \bmod p$, and Carol will compute $C_2^{ab} \equiv (g^{ab})^c \equiv g^{abc} \bmod p$.

This is the end of 3PDH, and Alice, Bob, and Carol all share the secret g^{abc}.

The sequence of messages is captured in the following diagram:



NPDH The construction of NPDH is simply a generalization of the construction of 3PDH. We can call 3PDH NPDH-3, that is, NPDH with 3 parties.

For brevity, NPDH is described only from the perspective of Alice. In NPDH, all other parties do the same things as Alice, only using their own secrets and round keys. As 3PDH had 3 rounds, NPDH will have N rounds.

NPDH will use a similar choice of public DH group $\mathbb{F}_p^*$ with generator g.

In the 0th round, Alice will generate her secret $a \in \mathbb{F}_p^*$ and compute her round key $A_0 \equiv g^a \bmod p$. She will then send her round key to every other participant, and receive all of their round keys as well. This is the end of the 0th round.

In the 1st round, Alice will compute all of her round keys by computing the action of her secret on all round keys she received the previous round. She will then send all her round keys to every other participant, and receive all of their round keys as well. This is the end of the 1st round.

In the ith round, Alice will compute all of her round keys by computing the action of her secret on the round keys she received in the prior round. She will then send all of her round keys to every other participant, and receive all of their round keys as well.

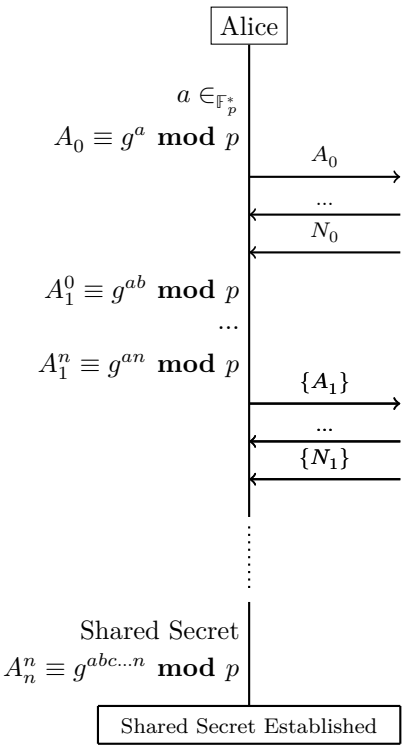
The final round is the $n-1$th round, where Alice will compute all of her round keys by the action of her secret on all previously received round keys. Exactly one of these received round keys will *not* have been acted upon by Alice, so she

computes the action of her secret upon it. This key is now the shared secret, as it is the action of every party's secret upon the public DH generator g.

This is seen (from Alice's perspective) in the diagram below:

Public parameters:
g, p

$\{A_i\}$ the set of Alice's round i keys



13

Suppose currently files[*sic*] storage services such as Dropbox and Google Drive do not offer confidentiality and integrity of data that goes over the Internet. You launch a startup called RowdyBox. RowdyBox distinguishes itself from Dropbox by addressing confidentiality and integrity of their customers' files going over the public Internet. Explain using illustrations how you would design a security architecture for RowdyBox. Identify the various concerns and show how your security architecture addresses those security concerns. Remember, you start from scratch. We do not have any pre-agreed keys, etc. Draw a figure with various entities involved, label the messages and explain. You should use the notations used in this course. Do not use any technique we have not covered so far in this course.

A

In considering confidentiality & integrity, I will assume RowdyBox does not provide any method for users to share their data with other users. This feature is not completely orthogonal to confidentiality and integrity, particularly if the file sharing happens on the server side.

Some security concerns include: * Confidentiality of files - only the user should be able to access their files. * Integrity of files - the user should be able to detect if their files have been modified.

RowdyBox's security architecture provides confidentiality and integrity to users who wish to store files. We uses end-to-end encryption to ensure said confidentiality and integrity of user files, and public-key

To that end, users with a RowdyBox account can protect their files by encrypting them using a strong encryption method in the RowdyBox client: AES256-CBC-CBCMAC (hereafter AES-CBCMAC). The RowdyBox client (trust us on this) generates secure AES-CBCMAC keys, and encrypts all files with them. These encrypted files are now ready to be transmitted over the internet to our servers and securely stored. Because we use end-to-end encryption, it doesn't matter

how the files are transmitted (currently the RowdyBox client uses raw IP datagrams as specified by RFC6214), only the user is able to read the plaintext of the file. See figure 2 for a diagram of the RowdyBox *File Upload Protocol*.

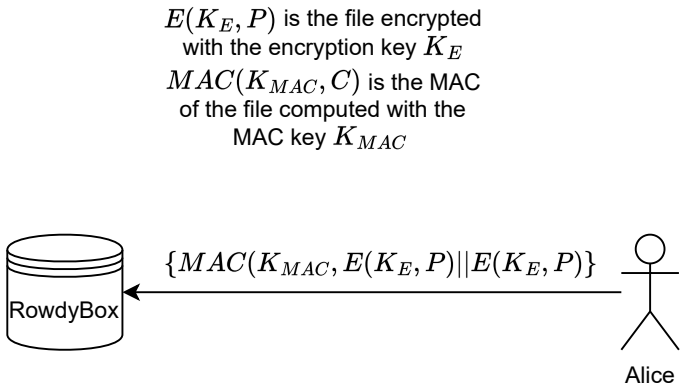


Figure 2: RowdyBox File Upload Protocol

It is important to know how users can request files stored with RowdyBox, and how RowdyBox knows which files belong to which users. That's actually quite simple - we don't! We don't keep any logs of which users have stored what files on our server! When a user wants to access a file, they can request it by sending the MAC of the encrypted file to us. Because the files are end-to-end encrypted with a key known only to the user, This MAC can be public information, and will be unique for any encrypted file. The MAC is much smaller than the message (for AES-CBCMAC the MAC is 256 bits), so users can store a near-unlimited number of MACs on their devices. (If a user runs out of space locally, they could encrypt the MAC database, store it on RowdyBox, and only store the MAC of the MAC database!) See figure 3 for a diagram of the RowdyBox *File Request Protocol*.

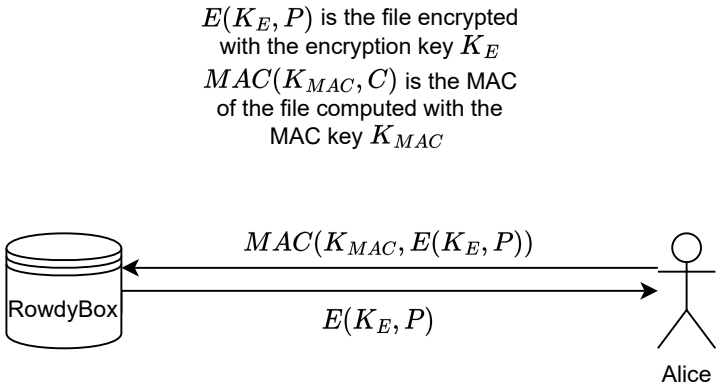


Figure 3: RowdyBox File Request Protocol

RowdyBox ensures confidentiality of users files with end-to-end encryption. The RowdyBox client generates a local encryption key and uses it to encrypt files before uploading them to RowdyBox's servers. Only the user can read their files, because only the user has the encryption key. RowdyBox ensures integrity of users files again with end-to-end encryption. The RowdyBox client generates a local MAC key and uses it to compute a MAC of the encrypted file. Only the user can compute the MAC on the encrypted file, because only the user has the MAC key.