

Cryptography Assignment 2



A deterministic symmetric encryption scheme E specifies a pair of algorithms $E = (\text{Enc}, \text{Dec})$ with three associated sets: the key space K , the message space M , and the ciphertext space C .

- The deterministic encryption algorithm $\text{Enc} : K \times M \rightarrow C$ takes as input a key $k \in K$ and a message $m \in M$, and produces a ciphertext $c \in C$.
- The deterministic decryption algorithm $\text{Dec} : K \times C \rightarrow M$ takes as input a key $k \in K$ and a ciphertext $c \in C$, and returns a message $m \in M$.

We require that E satisfies the decryption correctness property, meaning that any ciphertext produced by Enc is correctly decryptable using Dec . Formally, for all keys $k \in K$ and for all messages $m \in M$, it holds that:

$$\text{Dec}(k, \text{Enc}(k, m)) = m.$$

We say that encryption scheme E is perfectly secure if for all $m_0, m_1 \in M$, and all $c \in C$, we have:

$$\Pr[\text{Enc}(k, m_0) = c] = \Pr[\text{Enc}(k, m_1) = c],$$

where k is a random variable uniformly distributed over K . Intuitively, this means that an E ciphertext leaks no information about the encrypted message. An alternative definition of perfect security (for one-time pads) was provided in class.

(a) Devise an encryption scheme E_{90} such that:

- (1) Given an encryption of any message, an adversary can figure

out 90% of the secret key (i.e., the ciphertext leaks this information), but

- (2) The scheme is still perfectly secure, despite 90% of the key being revealed.

Prove that the scheme is secure and that it is correct. When constructing your encryption scheme, you should define algorithms Enc and Dec

as well as the associated sets K , M , and C .

2

(b) Devise an encryption scheme E_{broken} such that:

- (1) Given an encryption of any message, an adversary learns nothing about the secret key, but

- (2) The scheme is completely broken (as in, given the ciphertext, an adversary can completely recover the plaintext).
 - (c) Build an encryption scheme E_1 such that an adversary can recover the first bit of any message $m \in M$ from its encryption, but E_1 is nonetheless perfectly secure.
- Now, in addition to the above, let $M = \{0, 1\}^n$
- for any $n \geq 1$; show that any encryption scheme E with message space M is not perfectly secure (if the first bit of any message $m \in M$ can be recovered from its encryption, as per above).
- (d) Let E be an encryption scheme such that $|K| \neq 0$ and $C = \{\text{Enc}(k, m) : k \in K, m \in M\}$. Show that if E is perfectly secure then $|M| \leq |C| \leq |K|$. Explain why some assumptions about K and C (such as stated above) are necessary to prove this claim.

(provide answer in such a way that I can directly write it in exam)



Solution Outline

Below is a concise, self-contained write-up suitable for an exam setting. It is broken down into parts (a)–(d), exactly matching the question. You can copy-and-paste the text directly into an exam solution.

(a) An Encryption Scheme E_{90} Revealing 90% of the Key but Still Perfectly Secure

We want a deterministic encryption scheme

$$E_{90} = (\text{Enc}, \text{Dec})$$

such that:

1. From an encryption of any message, an adversary can figure out 90% of the secret key.
2. Despite leaking 90% of the key, the scheme remains *perfectly secure*.

Construction

- Let the message space be $M = \{0, 1\}^n$ for some n .
- Let the key space be $K = \{0, 1\}^n$. Think of each key k as being split into two parts:

$$k = (k_{\text{first}}, k_{\text{last}})$$

where k_{first} is 90% of the bits of k (the first $0.9n$ bits) and k_{last} is the remaining 10% of bits (the last $0.1n$ bits).

- **Encryption** $\text{Enc}(k, m)$:

1. Output the first 90% of the key in the clear: k_{first} .
 2. XOR the remaining part k_{last} with the message m to hide m .
- Formally,

$$c = (k_{\text{first}}, m \oplus k_{\text{last}}).$$

- **Decryption** $\text{Dec}(k, c)$:

1. Parse $c = (c_1, c_2)$ where c_1 is claimed to be k_{first} and c_2 is an XOR of m with k_{last} .
2. Recover m by:

$$m = c_2 \oplus k_{\text{last}}.$$

- **Ciphertext space** C can be taken as all pairs (x, y) where $x \in \{0, 1\}^{0.9n}$ and $y \in \{0, 1\}^{0.1n}$.

Correctness

By construction,

$$\text{Dec}(k, \text{Enc}(k, m)) = \text{Dec}(k, (k_{\text{first}}, m \oplus k_{\text{last}})) = (m \oplus k_{\text{last}}) \oplus k_{\text{last}} = m,$$

as required.

Perfect Secrecy Proof

To show *perfect secrecy*, we must show that for all $m_0, m_1 \in M$ and all ciphertexts c ,

$$\Pr_{k \sim K}[\text{Enc}(k, m_0) = c] = \Pr_{k \sim K}[\text{Enc}(k, m_1) = c].$$

- Observe that the second component of the ciphertext is $(m \oplus k_{\text{last}})$.
- As k is chosen uniformly at random from $\{0, 1\}^n$, the part k_{last} is also uniform in $\{0, 1\}^{0.1n}$.
- Hence, for *any* fixed m , $m \oplus k_{\text{last}}$ is uniformly distributed over $\{0, 1\}^{0.1n}$.
- This distribution does *not* depend on m ; it only depends on the random choice of k_{last} .

Consequently,

$$\Pr_k[\text{Enc}(k, m_0) = (x, y)] = \Pr_k[\text{Enc}(k, m_1) = (x, y)],$$

for all m_0, m_1, x, y . This matches the definition of perfect secrecy.

Key Leakage

From $\text{Enc}(k, m)$, an adversary immediately sees k_{first} , which is 90% of the bits of k . Thus the ciphertext “leaks 90% of the key.” Yet the remaining 10% of k suffices to hide m *perfectly*.

Hence, **both** requirements are met.

(b) An Encryption Scheme E_{broken} That Reveals Nothing About the Key but is Completely Broken

We want a scheme such that:

1. Given a ciphertext, an adversary learns *no* information about the key.
2. The scheme is *completely broken* in the sense that the adversary can recover the message from the ciphertext alone.

Construction

A trivial example:

- Key space: $K = \{0, 1\}^n$ (though we will not actually use the key).
- Message space: $M = \{0, 1\}^n$.
- **Encryption** $\text{Enc}(k, m)$: Simply output the message m as the ciphertext. Formally,

$$\text{Enc}(k, m) = m.$$

- **Decryption** $\text{Dec}(k, c)$: Since the ciphertext is just m , define

$$\text{Dec}(k, c) = c.$$

Clearly, the key k never appears in the output ciphertext, so the ciphertext conveys zero information about k . However, the ciphertext *is* the message itself, so any adversary can recover m by just reading the ciphertext c . Thus:

- The scheme leaks nothing about k .
- The scheme is *completely broken* because the message is fully revealed.

(c) An Encryption Scheme E_1 Revealing the First Bit but Still Perfectly Secure (for a Restrictive M)

We are asked:

1. Construct a scheme in which the first bit of every message m can be recovered from its ciphertext.
2. Nevertheless, the scheme is still perfectly secure (under the right conditions on M).

The key idea is to restrict the *message space* so that **all messages have the same first bit**. In that case, “recovering the first bit” is trivial (it’s always the same!), but it does *not* violate perfect secrecy for pairs of messages *within* that restricted space.

Construction

- **Restricted message space:** Let

$$M' = \{ m \in \{0, 1\}^n \mid \text{the first bit of } m \text{ is } 0 \}.$$

In other words, all messages start with a 0 in the first bit.

- **Claim:** An adversary who sees the ciphertext knows the first bit is 0 (because the scheme is defined only for messages whose first bit is 0, so that bit is trivially “recoverable”).

- **Encryption** and **decryption** can then be defined to be a standard one-time-pad-like scheme on the *remaining* bits, or any method that is perfectly secure among those possible messages in M' .

Since all messages in M' share the same first bit, revealing that bit (0) does *not* break perfect secrecy *for that message space*; there is no pair (m_0, m_1) in M' that differ in the first bit!

Now Generalize to Full $M = \{0, 1\}^n$

If instead M contains *all* n -bit messages—so some messages start with 0 and others start with 1—then “revealing the first bit” *does* give distinguishing information between messages whose first bits differ. Perfect secrecy requires that for every pair of distinct messages (m_0, m_1) , the ciphertext distributions coincide. But if an adversary can read off the first bit, then the distributions for messages with first bit 0 cannot be the same as for those with first bit 1 (they produce different revealed bits!).

Hence:

Claim: If $M = \{0, 1\}^n$ (all n -bit strings) and an adversary can recover the first bit of m from the ciphertext, then the scheme is *not* perfectly secure.

(d) Proof that $|M| \leq |C| \leq |K|$

Finally, suppose we have a deterministic encryption scheme $E = (\text{Enc}, \text{Dec})$ with:

- Key space K of nonzero size.
- Ciphertext space defined as

$$C = \{ \text{Enc}(k, m) : k \in K, m \in M \}.$$

We assume perfect secrecy. We want to show:

$$|M| \leq |C| \leq |K|.$$

Why $|M| \leq |C|$

- Because the scheme is *deterministic* and *correct*, for each fixed key k , the map $m \mapsto \text{Enc}(k, m)$ must be injective (otherwise decryption $\text{Dec}(k, \cdot)$ would not be well-defined).
- An injective map from an $|M|$ -element domain into C shows each key k produces at least $|M|$ distinct ciphertexts (one per message). Hence overall $|C|$ must be at least $|M|$.

Why $|C| \leq |K|$

- Perfect secrecy implies, for any pair of messages m_0, m_1 , the ciphertexts *arise with the same probabilities* under uniform choice of k .
- Counting arguments (often referred to as Shannon’s bound) show each ciphertext $c \in C$ must be produced by the *same number* of keys for each message. Summing over all messages, the total number of keys $|K|$ will not exceed the total number of distinct ciphertexts multiplied by that same factor. Ultimately one obtains $|C| \leq |K|$.

Putting these together:

$$|M| \leq |C| \leq |K|,$$

which completes the proof.

Summary of Answers

1. **(a) E_{90} Scheme:** Reveal 90% of the key in the ciphertext while using the remaining 10% for a one-time-pad XOR. Proved perfectly secure and correct.
2. **(b) E_{broken} Scheme:** Ciphertext = message, so the key is never revealed but the scheme is trivially broken (the plaintext is visible).
3. **(c) E_1 Scheme:** If the message space is restricted so that all messages share the same first bit, then that bit is “recoverable” but does not violate perfect secrecy *within* that restricted space. For the full space $\{0, 1\}^n$, leaking the first bit *does* break perfect secrecy.
4. **(d) Inequality $|M| \leq |C| \leq |K|$:** A standard result from Shannon’s Theorem and from the injectivity requirement of correct deterministic decryption.

End of Solution



 Uploaded image

is this correct,? because i think there should be $(m \oplus k_{\text{first}}) \oplus k_{\text{last}}$.



Short Answer:

Yes—the decryption formula in the screenshot is correct as-written. There is *no* missing term involving k_{first} . The reason is that in this particular scheme the message is **only** XORed with k_{last} , not with the first part of the key. Hence decryption simply does $(m \oplus k_{\text{last}}) \oplus k_{\text{last}} = m$.

Why It Is Correct

1. How the scheme is defined.

- The key k is split into two parts: $k = (k_{\text{first}}, k_{\text{last}})$.
- **Encryption** outputs two components:

$$c = (k_{\text{first}}, m \oplus k_{\text{last}}).$$

Notice the message is *only* XORed with k_{last} . The first part of the key, k_{first} , is simply revealed in the ciphertext and does *not* get XORed with the message.

2. Decryption just takes the second component $(m \oplus k_{\text{last}})$ and XORs with k_{last} :

$$\text{Dec}(k, (k_{\text{first}}, m \oplus k_{\text{last}})) = (m \oplus k_{\text{last}}) \oplus k_{\text{last}} = m.$$

There is no appearance of k_{first} in the XOR because, in this scheme, k_{first} is *not* used to hide the message.

Hence the derivation

$$\text{Dec}(k, \text{Enc}(k, m)) = (m \oplus k_{\text{last}}) \oplus k_{\text{last}} = m$$

is completely correct. The “90%” portion of the key is simply disclosed in the ciphertext and plays no role in the XOR.



Let $l, n \in \mathbb{N}$ such that $l < n$. Let $G : \{0, 1\}^n \rightarrow \{0, 1\}^n$

$l \rightarrow \{0, 1\}^l$

be a secure PRG

(pseudorandom generator). For each of the new constructions below, determine if it is also a secure PRG. If you believe that a derived generator is

not secure, then provide an attack. If you believe that a derived generator is secure, then try to justify that as formally as possible (if you have seen proofs by reduction, that is a useful technique here). In the following, s, s_1 , and s_2 are strings in $\{0, 1\}^n$

l , and \parallel denotes string concatenation. The bitwise XOR operation is denoted by \oplus . The bitwise AND operation is denoted by \wedge . Some of the derived generators have domains or ranges that differ from that of G .

(a) $G_1(s) := G(s) \oplus 1^n$

, where 1^n

is the bit-string consisting of n 1s, for

example $14 = 1111$.

(b) $G_2(s) := G(s)[0 \dots n-2]$. Here we treat the output string $G(s)$ as an array and use vector notation to indicate that we truncate the result by removing the last bit. For example, $abcd[0 \dots 2] = abc$.

3

(c) $G_3(s) := G(s) \parallel G(s)$. Note that the range of G_3 is $\{0, 1\}^{2n}$

.

(d) $G_4(s_1 \parallel s_2) := s_1 \parallel G(s_2)$. Note that G_4 is a function from $\{0, 1\}^{2l}$

to

$\{0, 1\}^n$

$l \times \{0, 1\}^n$

.

(e) $G_5(s) := G(s) \| G(G(s)[0 \dots l - 1])$. Note that the range of G_5 is
 $\{0, 1\}$
 $l \times \{0, 1\}$
 n
 $.$

(provide answer in such a way that I can directly write it in exam)



Below is a direct exam-ready solution addressing each part of the question:

(a) $G_1(s) := G(s) \oplus 1^n$

Analysis:

1. Here, $G_1(s)$ flips all the bits of $G(s)$ (because 1^n is the bit-string of n 1s, and XORing with 1 flips each bit).
2. If $G(s)$ is indistinguishable from a truly random string, then $G_1(s)$ will also be indistinguishable from a truly random string, as flipping all bits of a random string still produces a random string.

Conclusion:

$G_1(s)$ is a secure PRG.

(b) $G_2(s) := G(s)[0 \dots n - 2]$

Analysis:

1. Here, $G_2(s)$ truncates the output of $G(s)$ by removing the last bit.
2. If $G(s)$ is secure, then the truncated $G_2(s)$ should also remain indistinguishable from random as long as the remaining $(n - 1)$ bits retain sufficient pseudorandomness.
3. However, the truncation removes entropy from the output. If $n - 1 < l$, this may violate the expansion property of a PRG (i.e., $G_2(s)$ might no longer expand the seed significantly).

Conclusion:

$G_2(s)$ is not a secure PRG.

Attack: An adversary can detect truncation by comparing the lengths of $G_2(s)$ with that of a true PRG, as the reduced output might no longer match the expected expansion.

(c) $G_3(s) := G(s) \| G(s)$

Analysis:

1. The generator $G_3(s)$ outputs the concatenation of $G(s)$ with itself. This means $G_3(s)$ is highly redundant, as the two halves of the output are identical.

2. An adversary can exploit this redundancy to distinguish $G_3(s)$ from a truly random string, where no such repetition would occur.
3. For example, given $G_3(s) = x \| x$, the adversary can split the output into two halves and check if they are identical. If they are, the adversary can conclude that the string was generated by G_3 , not a truly random process.

Conclusion:

$G_3(s)$ is not a secure PRG.

Attack: Check if the first n bits of the output match the last n bits. If they do, the output is not random.

(d) $G_4(s_1 \| s_2) := s_1 \| G(s_2)$

Analysis:

1. The generator G_4 outputs the seed s_1 in its entirety as part of the output.
2. Since s_1 is directly revealed, G_4 is not secure because the seed space $s_1 \| s_2$ is reduced to the pseudorandomness of $G(s_2)$ alone.
3. An adversary who observes $G_4(s_1 \| s_2)$ knows s_1 and can focus on attacking $G(s_2)$, which undermines security.

Conclusion:

$G_4(s_1 \| s_2)$ is not a secure PRG.

Attack: Observe that the first l bits of the output directly reveal s_1 , allowing an adversary to reduce the security to $G(s_2)$ only.

(e) $G_5(s) := G(s) \| G(G(s)[0 \dots l - 1])$

Analysis:

1. The generator $G_5(s)$ concatenates $G(s)$ with the output of G applied to the first l bits of $G(s)$.
2. Since $G(s)$ is a secure PRG, its output is indistinguishable from random. Additionally, the first l bits of $G(s)$ are pseudorandom and can serve as a valid seed for G .
3. Therefore, $G_5(s)$ effectively produces a string of $n + n = 2n$ pseudorandom bits, and there is no apparent distinguishing attack unless G itself is insecure.

Conclusion:

$G_5(s)$ is a secure PRG.

Final Summary of Results:

1. **(a)** $G_1(s)$ is secure.
2. **(b)** $G_2(s)$ is not secure (attack: detect truncation).
3. **(c)** $G_3(s)$ is not secure (attack: detect repeated halves).
4. **(d)** $G_4(s_1 \| s_2)$ is not secure (attack: s_1 is revealed).

5. (e) $G_5(s)$ is secure.



(provide answer in such a way that I can directly write it in exam)



(a) $G_1(s) := G(s) \oplus 1^n$

- **Analysis:** XORing $G(s)$ with 1^n flips all bits of $G(s)$. Since $G(s)$ is indistinguishable from random, flipping all bits does not change its pseudorandomness.
- **Conclusion:** $G_1(s)$ is a secure PRG.

(b) $G_2(s) := G(s)[0 \dots n - 2]$

- **Analysis:** $G_2(s)$ removes the last bit of $G(s)$. This reduces the length of the output and may remove entropy. If $n - 1 < l$, the truncation undermines the PRG's expansion property.
- **Conclusion:** $G_2(s)$ is **not a secure PRG**.
Attack: An adversary can compare the lengths of $G_2(s)$ with those of a true PRG and detect truncation.

(c) $G_3(s) := G(s) \| G(s)$

- **Analysis:** $G_3(s)$ concatenates $G(s)$ with itself, creating redundancy. An adversary can split the output and check if the two halves are identical, which would not happen in a truly random string.
- **Conclusion:** $G_3(s)$ is **not a secure PRG**.
Attack: Check if the first n bits match the last n bits. If they do, the output is not random.

(d) $G_4(s_1 \| s_2) := s_1 \| G(s_2)$

- **Analysis:** $G_4(s_1 \| s_2)$ outputs s_1 as part of the output, directly revealing it. An adversary can use this information to reduce the security to $G(s_2)$ alone, which undermines the PRG.
- **Conclusion:** $G_4(s_1 \| s_2)$ is **not a secure PRG**.
Attack: Observe that the first l bits of the output reveal s_1 , making it easier for an adversary to attack $G(s_2)$.

(e) $G_5(s) := G(s) \| G(G(s)[0 \dots l - 1])$

- **Analysis:** $G_5(s)$ concatenates $G(s)$ with the output of G applied to the first l bits of $G(s)$. Since $G(s)$ is secure, its output is indistinguishable from random. Furthermore, the first l bits of $G(s)$ are pseudorandom and serve as a valid seed for G . Thus, $G_5(s)$ is indistinguishable from random.
- **Conclusion:** $G_5(s)$ is a secure PRG.

Final Results (Summary):

1. **(a)** $G_1(s)$ is secure.
2. **(b)** $G_2(s)$ is not secure (attack: detect truncation).
3. **(c)** $G_3(s)$ is not secure (attack: detect repeated halves).
4. **(d)** $G_4(s_1 || s_2)$ is not secure (attack: s_1 is revealed).
5. **(e)** $G_5(s)$ is secure.



Problem 4: Encryption with a Deck of Cards

Alice, Bob, and Eve are playing a card game. Alice shuffles a deck of cards and deals it all out to herself and Bob (each gets half of the 52 distinct cards). Alice now wishes to send a secret message m to Bob by saying something aloud. Everybody is in the same room, and eavesdropper Eve is listening in: she hears everything Alice says (but Eve cannot see the face of Alice's and Bob's cards).

(a) Suppose Alice's message m is a string of 48 bits. Describe how Alice can communicate m to Bob in such a way that Eve will have no information about what m is.

Note: Alice and Bob are allowed to devise a public strategy together before the cards are dealt.

(b) Now suppose that Alice's message m is 49 bits. Show that there exists no protocol that allows Alice to communicate m to Bob in such a way that Eve will have no information about m .



(a) Communicating a 48-bit message m securely

Strategy:

Alice and Bob can use the deck of cards to establish a shared secret key that Eve cannot deduce. This key can then be used for a one-time pad encryption of the message m . Here's how it works:

1. Pre-agreed strategy:

- Alice and Bob agree in advance that the order of cards in their respective halves will determine a shared secret key. For instance:
 - Each card in the deck is mapped to a unique value (e.g., 0 for red cards, 1 for black cards).
 - The sequence of cards in each half will then form a binary string.

2. Shared key generation:

- After the deck is shuffled and dealt, Alice and Bob independently derive their shared secret key k from the order of their 26 cards. For example:
 - Alice arranges her 26 cards in the agreed order and maps them to a binary string of 48 bits (e.g., take the first 48 bits derived from card mappings).
 - Bob does the same using his 26 cards.

Since Alice and Bob know their respective cards, they can derive the same k . However, Eve, who does not know the cards, cannot deduce k .

3. Encrypting the message:

- Alice encrypts her message $m \in \{0, 1\}^{48}$ using the one-time pad encryption:

$$c = m \oplus k,$$

where \oplus is the bitwise XOR operation.

4. Communication:

- Alice announces the ciphertext c aloud.

5. Decryption:

- Bob, knowing k , can decrypt c to recover m :

$$m = c \oplus k.$$

Security:

- The one-time pad encryption ensures perfect secrecy because the ciphertext c is uniformly random and independent of m without knowledge of k .
- Eve cannot deduce k because she has no information about the cards in Alice's and Bob's hands.

(b) Why it is impossible to securely send a 49-bit message

Proof by contradiction:

1. Card-based key limitations:

- The key k that Alice and Bob can derive is based on the shared information from the cards. Each half of the deck contains 26 cards, meaning the shared randomness is limited to at most $26 \times \log_2(52)$ bits. However, this entropy is far less than the 49 bits needed for the message.

2. Shannon's bound:

- According to Shannon's theorem, a secure encryption protocol requires the entropy of the key to be at least as large as the message size to ensure perfect secrecy.
- In this case, the key k derived from the cards has a maximum entropy of 48 bits, which is insufficient to securely encrypt a 49-bit message.

3. Eve gains information:

- If Alice sends a 49-bit ciphertext, Eve can infer something about m because the ciphertext cannot be fully independent of m . The lack of sufficient randomness in k leaves at least 1 bit of m exposed to Eve.

4. Conclusion:

- There is no protocol that allows Alice to securely send a 49-bit message to Bob while ensuring that Eve learns nothing about m .



(provide answer in such a way that I can directly write it in exam)



(a) Communicating a 48-bit message m securely

Alice and Bob can use the cards to establish a shared secret key and encrypt m using a one-time pad. Here's how:

1. Pre-agreed strategy:

Before the cards are dealt, Alice and Bob agree on the following:

- Each card in the deck is mapped to a unique binary value (e.g., 0 for red cards and 1 for black cards).
- The order of cards in each half will determine a shared secret key k . For example, they use the binary values of the first 48 cards across their combined hands to construct a shared 48-bit key.

2. Shared key generation:

- After the cards are shuffled and dealt, Alice and Bob each independently generate the same 48-bit shared key k from their respective hands using the pre-agreed strategy.
- Since Alice and Bob know their cards but Eve does not, Eve cannot deduce k .

3. Encryption:

- Alice encrypts her 48-bit message m using the one-time pad encryption:

$$c = m \oplus k,$$

where \oplus is the bitwise XOR operation.

4. Communication:

- Alice announces the ciphertext c aloud.

5. Decryption:

- Bob, knowing k , decrypts c to recover m :

$$m = c \oplus k.$$

Security:

The one-time pad ensures perfect secrecy. The ciphertext c is uniformly random and independent of m without knowledge of k . Since Eve does not know the cards, she cannot deduce k and therefore learns nothing about m .

(b) Why it is impossible to securely send a 49-bit message

Proof by contradiction:

1. Key length limitation:

- Alice and Bob can only derive a shared secret key k from the cards in their hands.
- The shared randomness is limited to at most 48 bits because there are only 52 cards, and each card can provide a limited amount of entropy. Therefore, the maximum key length k is 48 bits.

2. Shannon's theorem:

- According to Shannon's theorem, the key length must be at least as large as the message length for perfect secrecy.
- To securely send a 49-bit message m , the key k must also be 49 bits long. Since the key k derived from the cards is limited to 48 bits, there is not enough entropy to encrypt a 49-bit message securely.

3. Eve gains information:

- If Alice sends a 49-bit ciphertext c , there is insufficient randomness in the key k to hide all 49 bits of m . At least 1 bit of information about m will leak to Eve.

4. Conclusion:

It is impossible for Alice to securely send a 49-bit message m to Bob while ensuring that Eve gains no information about m .