CSL759: Cryptography and Computer Security

Ragesh Jaiswal
CSE, IIT Delhi

Administrative information

- Lecture Timing/Location:
  - Location: IIA 204
  - Day/Time: Tuesdays, 2-5pm

Introduction

Introduction: Secure communication

- Perfect Secrecy (Information Theoretic Security):
  - Let the message space be $\{0,1\}^n$.
  - For any two message M_0, M_1, and Ciphertext C
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C]$$
    where the probability is over uniformly random K in the Keyspace.
- <u>Fact</u>: If $|M| > |K|$, then no scheme is perfectly secure.
- How do we get around this problem?
  - <u>Relax our notion of security</u>: Instead of saying "it is impossible to break the scheme", we would like to say "it is *computationally infeasible* to break the scheme".

Introduction: Pseudorandom generator

- Suppose there was a *generator* that *stretches* random bits.

001101011 →  → 00101001001010010100101011

G

- Idea:
  - Choose a short key K randomly.
  - Obtain $K' = G(K)$.
  - Use K' as key for the one time pad.
- Issue: ?

Introduction: Pseudorandom generator

- Suppose there was a *generator* that *stretches* random bits.

001101011 →  → 001010010010100101001010 11

G

- Idea:
  - Choose a short key K randomly.
  - Obtain $K' = G(K)$.
  - Use K' as key for the one time pad.
- Issue:
  - Such a generator is not possible!
  - Any such generator produces a longer string but the string is not *random*.

Introduction: Pseudorandom generator

- Suppose there was a *generator* that *stretches* random bits.

001101011 →  → 00101001001010010100101011

G

- Idea:
  - Choose a short key K randomly.
  - Obtain $K' = G(K)$.
  - Use K' as key for the one time pad.
- Issue:
  - Such a generator is not possible!
  - Any such generator produces a longer string but the string is not *random*.
- What if we can argue that the output of the generator is *computationally indistinguishable* from truly random string.

Introduction: Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.
- Formalizing notion of security:
  - When can you say that your protocol has been broken?
    - Adversary is able to figure out the secret key.
      - Question: Can you say that your protocol is secure if no adversary can figure out the secret key?

Introduction: Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.
- Formalizing notion of security:
  - When can you say that your protocol has been broken?
    - Adversary is able to figure out the secret key.
      - Question: Can you say that your protocol is secure if no adversary can figure out the secret key?
        NO! Consider the encryption scheme $C = E_K(M) = M$

Introduction: Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.
- Formalizing notion of security:
  - When can you say that your protocol has been broken?
    - Adversary is able to figure out the secret key.
      - Question: Can you say that your protocol is secure if no adversary can figure out the secret key?
        NO! Consider the encryption scheme $C = E_K(M) = M$
    - Adversary is able to figure out the entire message.
      - Question: Can you say that your protocol is secure if no adversary can figure out the entire message?
        NO! Maybe the first bit of the message is revealed which may be crucial.

Introduction: Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.
- Formalizing notion of security:
  - When can you say that your protocol has been broken?
    - Adversary is able to figure out the secret key.
      - Question: Can you say that your protocol is secure if no adversary can figure out the secret key?
        NO! Consider the encryption scheme $C = E_K(M) = M$
    - Adversary is able to figure out the entire message.
      - Question: Can you say that your protocol is secure if no adversary can figure out the entire message?
        NO! Maybe the first bit of the message is revealed which may be crucial.
    - .
    - .
    - .

Introduction: Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.
- Formalizing notion of security:
  - Think of a strong security property **P** such that if your protocol follows **P**, then your protocol also satisfies all the security properties in the list.

Introduction: Integrity and Authenticity

- Non-tamperable communication: Alice wants to send messages to Bob so that Bob can be sure that the message was not change in transit.
- This kind of communication has completely different security goals and so we will have to come up with an entirely different security notion.
  - Note that Alice is not required to encrypt the message.
  - Why doesn't one time pad scheme work here?

Introduction

- If we have such a nice theoretical framework for constructing secure protocols, why are many protocols we see in practice insecure?
- Not many designers use these ideas (correctly) when designing protocols.
  - Not understanding security properties of basic primitives.
  - Combining secure primitives in an insecure manner.
  - Weakness in implementation.
  - Designing their own basic primitives that has not withstood the test of time.

Stream Ciphers

Pseudorandom generators

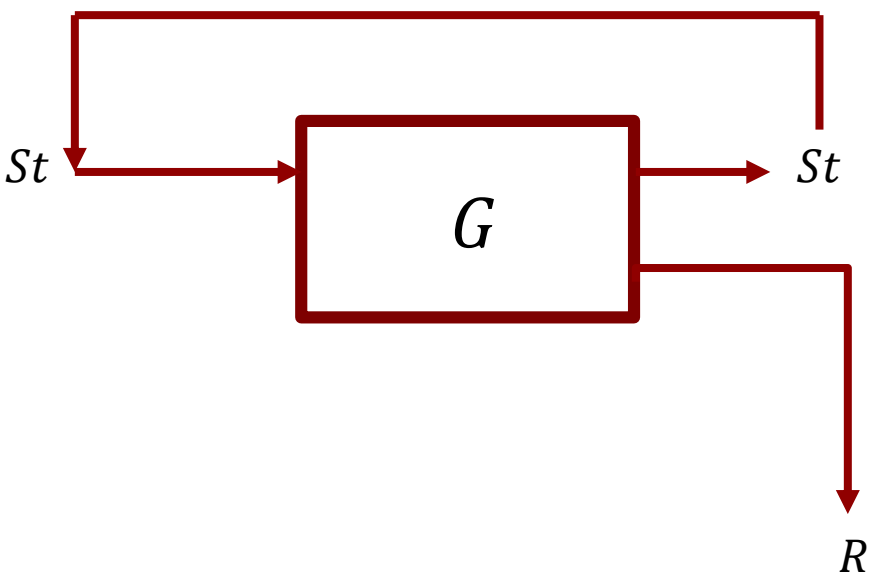
Stream Ciphers: Pseudorandom generators

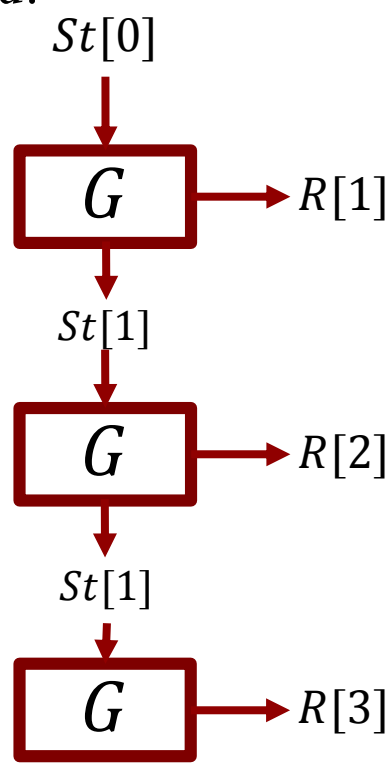
- A pseudorandom generator (PRG) is a function:

$$G: \{0,1\}^s \rightarrow \{0,1\}^n, n \gg s$$

such that $G(x)$ "appears" to be a random n bit string.

- The input to the generator is called the *seed*.





Stream Ciphers: Pseudorandom generators

- A pseudorandom generator (PRG) is a function:

$$G: \{0,1\}^s \rightarrow \{0,1\}^n, n \gg s$$

  such that $G(x)$ "appears" to be a random n bit string.
- Let us see if we can rule out some popular random generators based on this intuitive understanding of PRG:
  - <u>Linear Congruential Generator (LCG)</u>: parameters m, a, c:
    - $R_n = (a \cdot R_{n-1} + c)(mod\ m)$, the seed is R_0 and the output is $R_1 R_2 R_3 \ldots$
    - This has some nice statistical properties but it is "predictable".
    - Never use such "predictable" random number generators for Cryptography.

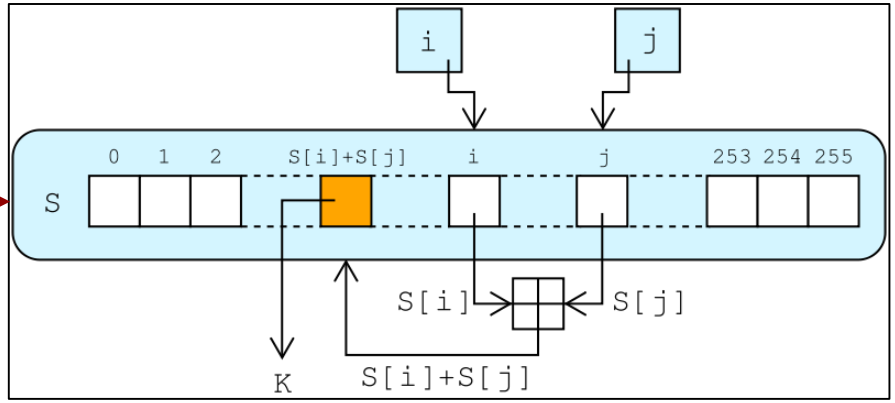
Stream Ciphers: Pseudorandom generators

- Let us see if we can rule out some popular random generators based on this intuitive understanding of PRG:
  - Linear Congruential Generator(LCG):
  - RC4: Used in SSL and WEP

K

(128 bits)

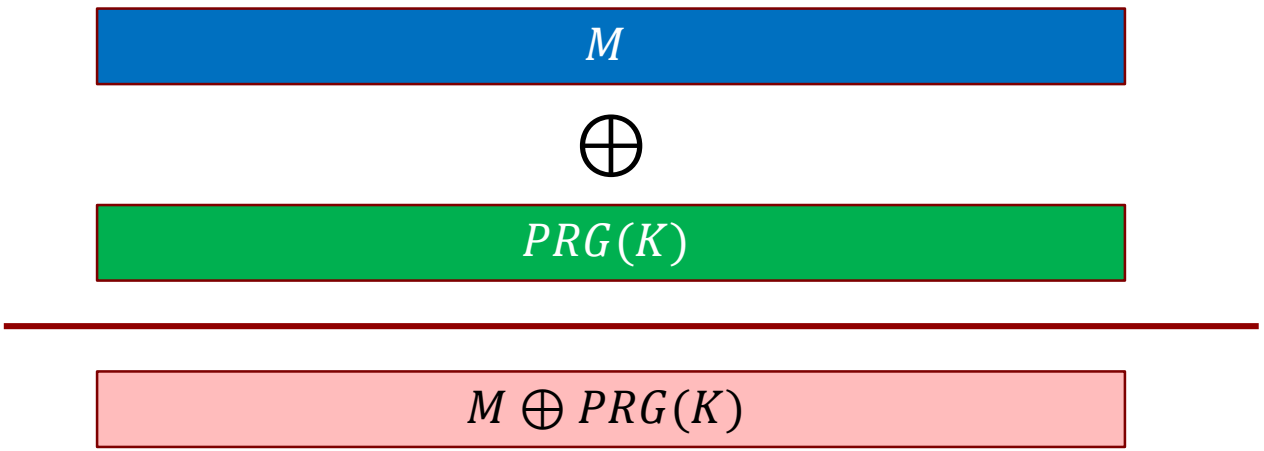
Seed used as secret key



- Observations:
  1. $\Pr[2^{nd}\ byte = 0] = 2/256.$
  2. $\Pr[1^{st}\ byte = 0\ and\ 2^{nd}\ byte = 0] = \frac{1}{256^2} + \frac{1}{256^3}.$
  3. First few bytes of the output are correlated with the key. Let us see an attack based on this idea.

Stream Ciphers: RC4 attack

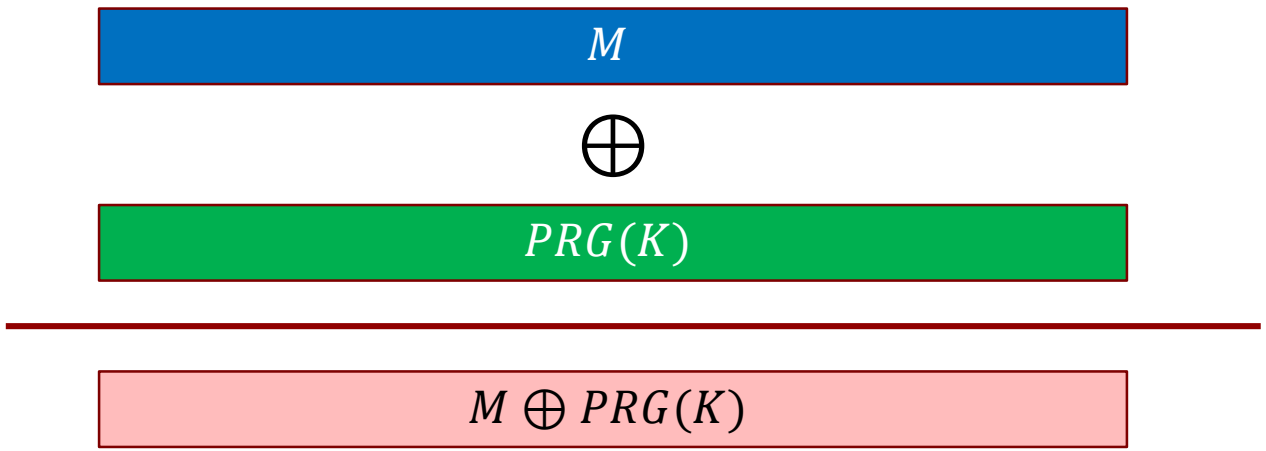
- How do we use a stream cipher?



- What is the issue with this idea?
  - What if there are more than one message that you want to encrypt?

Stream Ciphers: RC4 attack

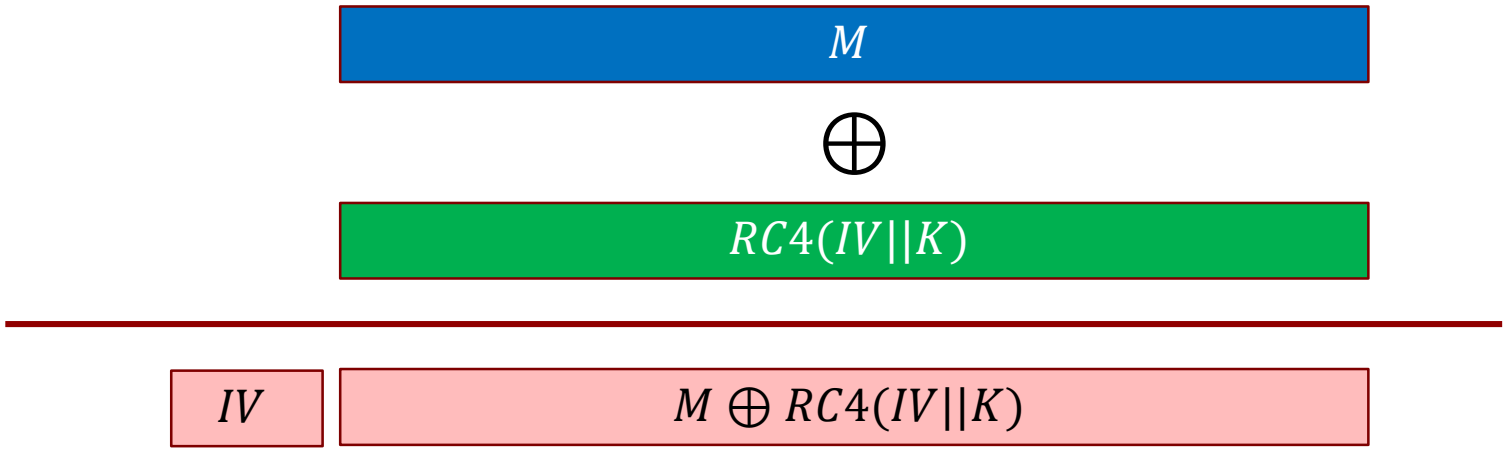
- How do we use a stream cipher?



- What is the issue with this idea?
  - What if there are more than one message that you want to encrypt?
  - ***Key reusability should always be avoided when using stream ciphers.***

Stream Ciphers: RC4 attack

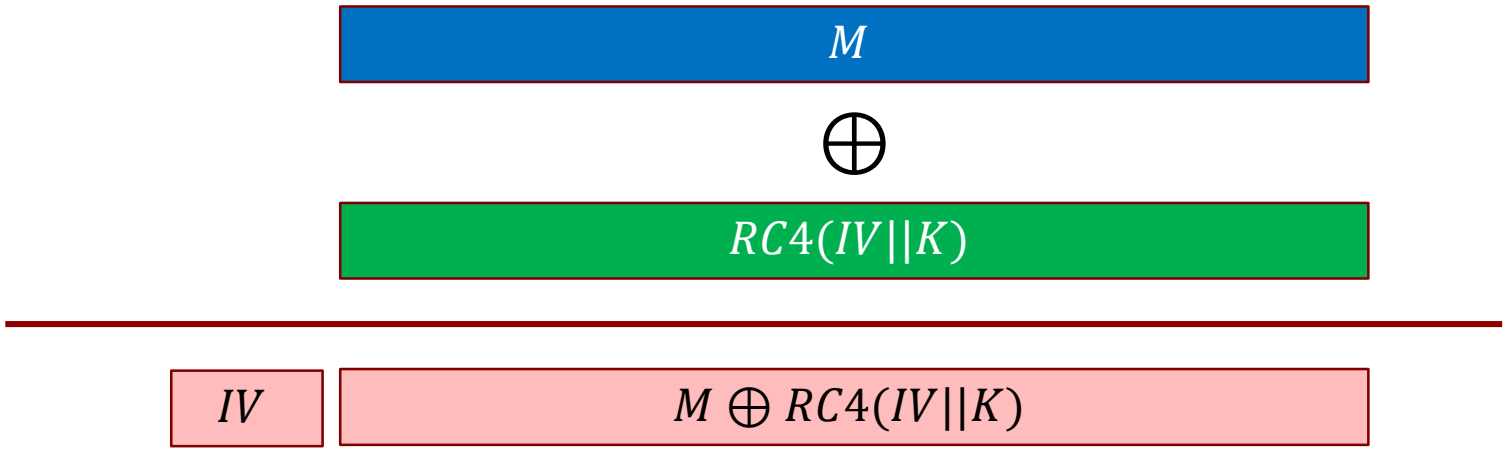
- How do we use a stream cipher?
  - Another idea: This is actually used in 128 bit WEP where $|IV| = 24$ and $|K| = 104$.



- What is the issue with the above protocol?
  - The IV gets repeated after 2^{24} frames.
  - In some 802.11 cards, the IV is set to 0 after every power cycle.

Stream Ciphers: RC4 attack

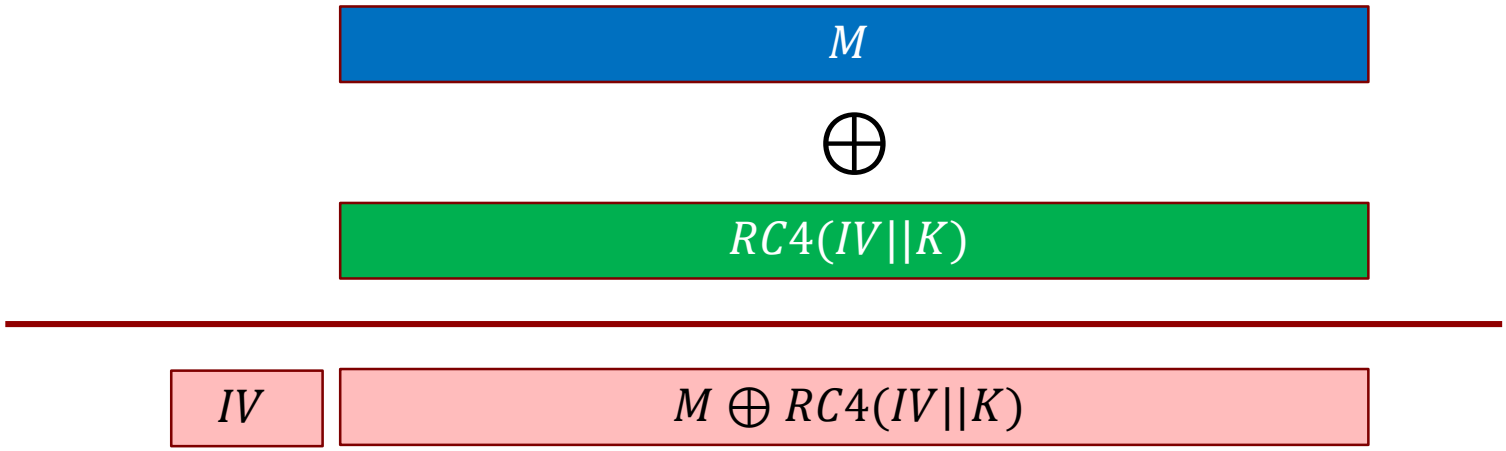
- How do we use a stream cipher?
  - Another idea: This is actually used in 128 bit WEP where $|IV| = 24$ and $|K| = 104$.



- What is the issue with the above protocol?
  - The IV gets repeated after 2^{24} frames.
  - In some 802.11 cards, the IV is set to 0 after every power cycle.
  - <u>Related key attack</u>: IV is incremented by 1 for each frame. So, the key though different, are very similar and one may use the correlation property to attack.

Stream Ciphers: RC4 attack

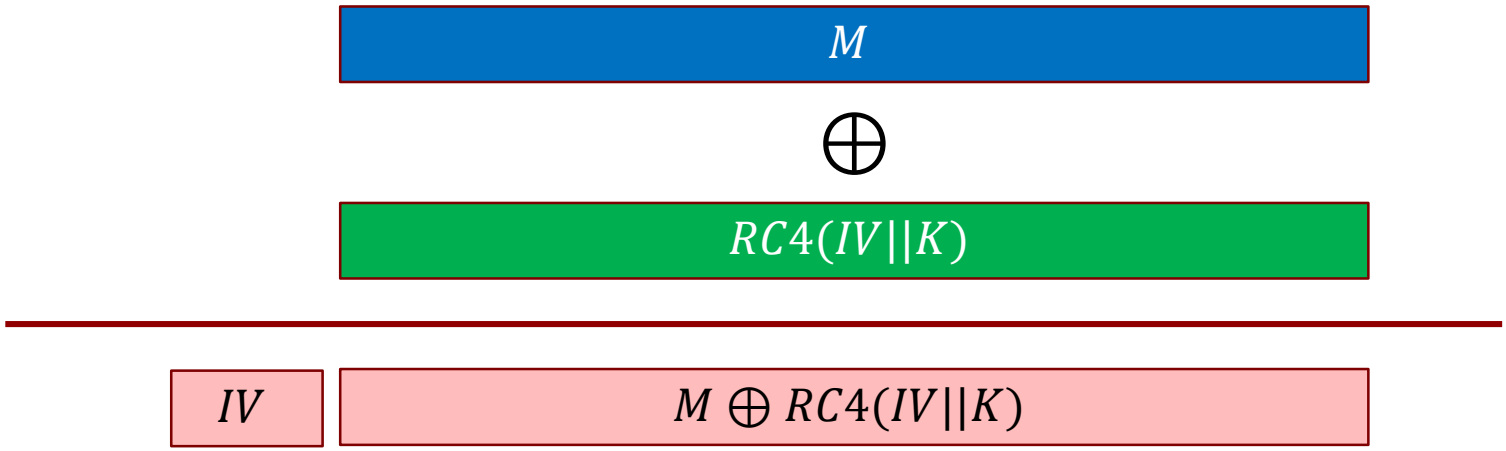
- How do we use a stream cipher?
  - Another idea: This is actually used in 128 bit WEP where $|IV| = 24$ and $|K| = 104$.



128 bit WEP is insecure. DO NOT USE!
There are attacks that will figure out your secret key in less than a minute. Check out *aircrack-ptw*.

Stream Ciphers: RC4 attack

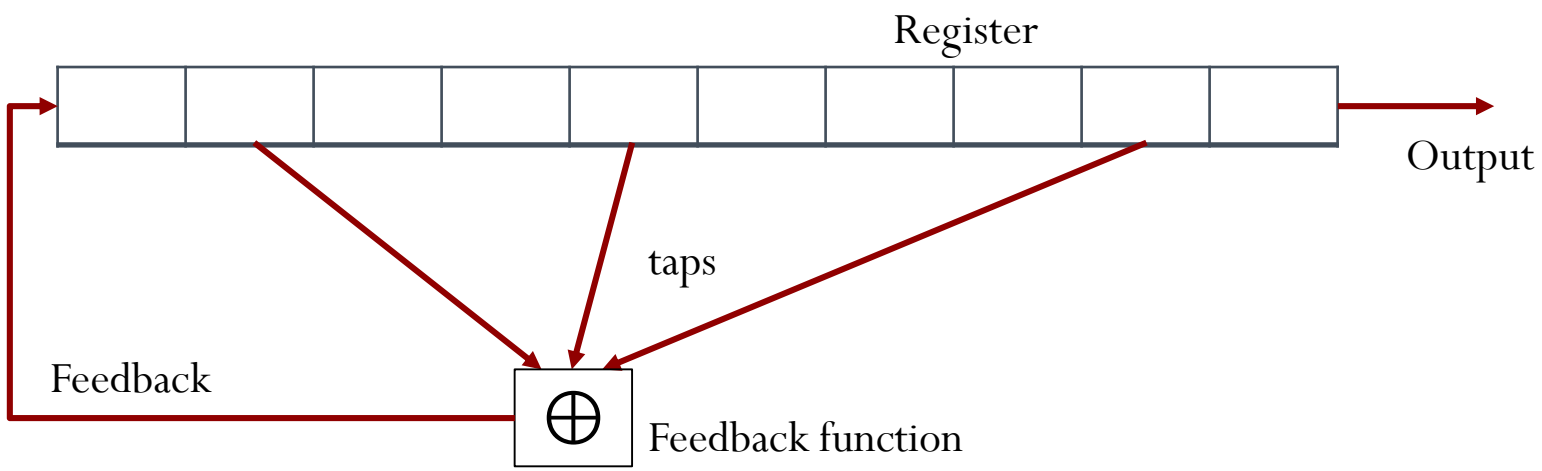
- How do we use a stream cipher?
  - Another idea: This is actually used in 128 bit WEP where $|IV| = 24$ and $|K| = 104$.



- So what is the fix? How do we use PRGs like RC4?
  - Throw away initial few bytes of RC4 output.
  - Use unrelated keys.

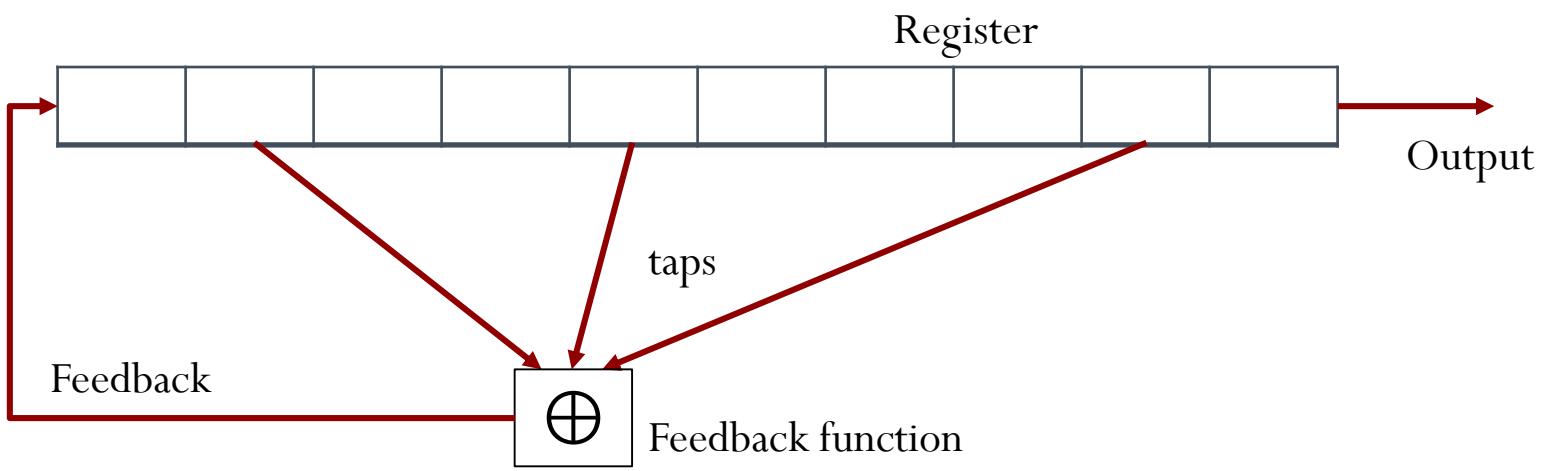
Stream Ciphers: Pseudorandom generators

- Linear Feedback Shift Registers (LFSR):
  - Fast hardware implementation.
  - <u>Examples</u>: DVD encryption (CSS), GSM encryption (A5/1,2).
  - Is this generator predictable?



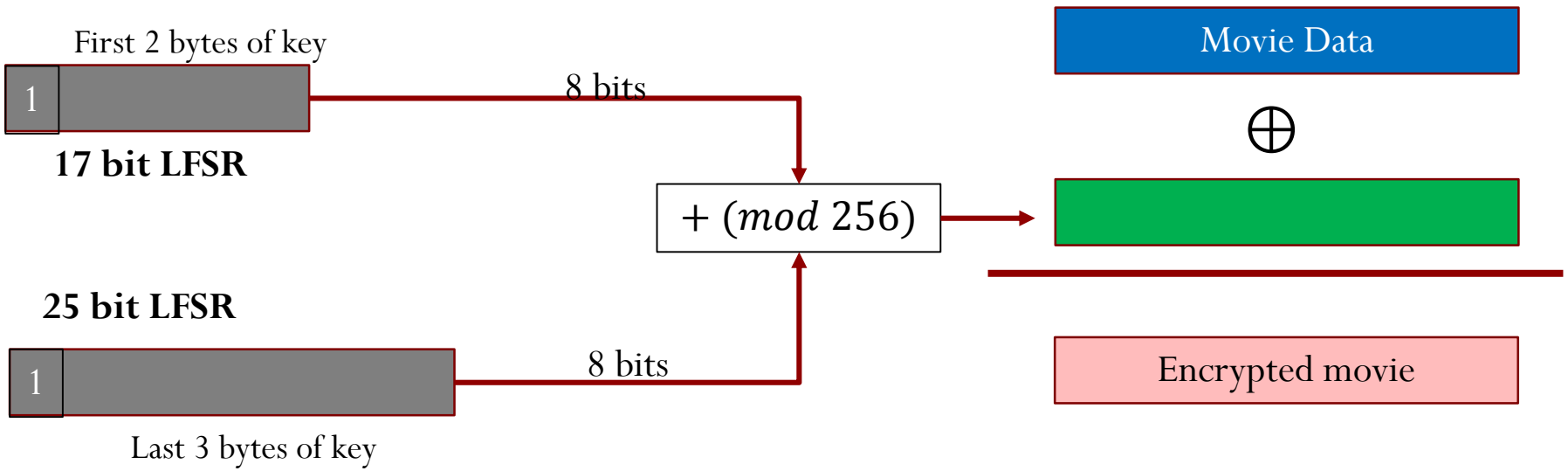
Stream Ciphers: Pseudorandom generators

- Linear Feedback Shift Registers (LFSR):
  - Fast hardware implementation.
  - Examples: DVD encryption (CSS), GSM encryption (A5/1,2).
  - Is this generator predictable?
    - Yes.
    - One solution that is used in practice is to use a combination of multiple LFSRs.



Stream Ciphers: Attack on CSS

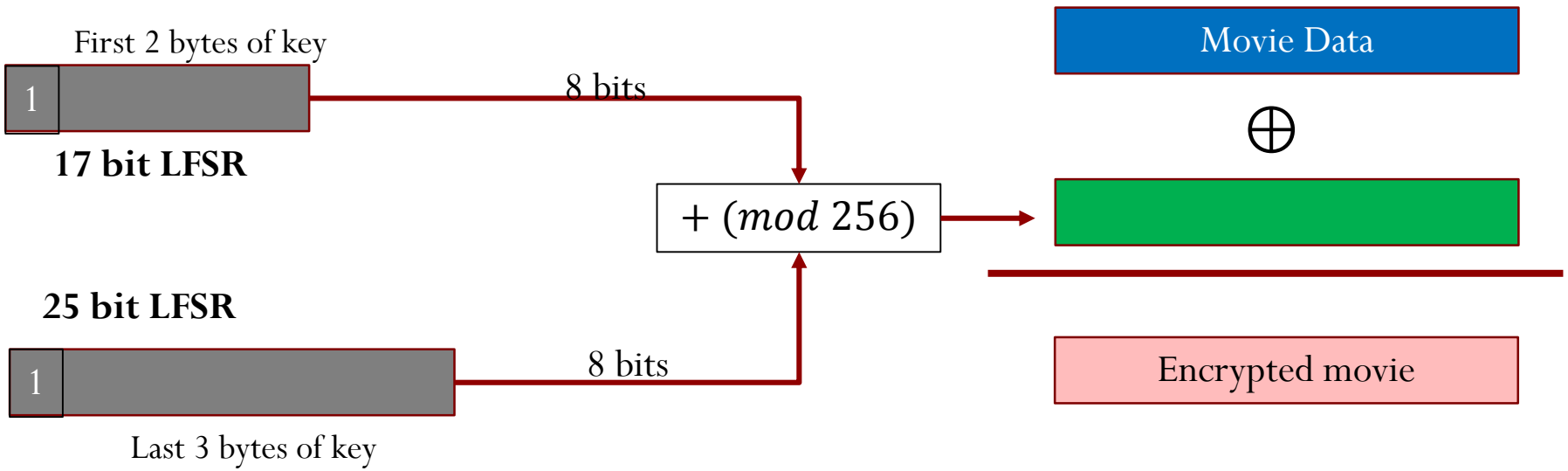
- CSS: Content Scrambling System is an encryption system for encrypting DVDs.
  - It uses 40 bit encryption key and 2 LFSRs in the manner shown below:



- How do you attack this protocol?

Stream Ciphers: Attack on CSS

- CSS: Content Scrambling System is an encryption system for encrypting DVDs.
  - It uses 40 bit encryption key and 2 LFSRs in the manner shown below:



- How do you attack this protocol?
  - Try all possibilities for the seed of the first LFSR.

Stream Ciphers: Pseudorandom generator

- What are the desirable properties of a pseudorandom generator $G: \{0,1\}^k \rightarrow \{0,1\}^n$ for Cryptographic purposes:
  - <u>Stretch</u>: $n > k$
  - <u>Efficient</u>: G should be efficient
  - <u>Indistinguishability</u>: No bounded resource algorithm should be able to distinguish the output of the generator $G(x)$ (for random x) from a random n bit string.
  - <u>Unpredictability</u>: The output of the generator should not be predictable.
- Let us develop these security notions and then study the relationship between them.

Stream Ciphers: Indistinguishability

- Indistinguishability: The output of the generator should appear to be random:
  - Question: To whom?
    - Answer: To all efficient algorithms.
  - What does "appear" mean?
    - Any efficient statistical testing algorithm should behave similarly when given random n bit inputs and when given output of the generator for random seed.
    - Definition (Advantage): The PRG advantage of an algorithm A with respect to a generator G is denoted by $Adv_{PRG}(A, G)$ is defined as

$$Adv_{PRG}(A, G) = \left| \Pr_{K \leftarrow \{0,1\}^k}\left[A\big(G(K)\big) = 1\right] - \Pr_{R \leftarrow \{0,1\}^n}[A(R)] = 1 \right|$$

    - Observations:
      - $0 \leq Adv_{PRG}(A, G) \leq 1$
      - $Adv_{PRG}(A, G)$ close to 1 means A can distinguish G's output from random.
      - $Adv_{PRG}(A, G)$ close to 0 means A cannot distinguish G's output from random.

Stream Ciphers: Indistinguishability

- <u>Definition (Advantage)</u>: The PRG advantage of an algorithm A with respect to a generator G is denoted by $Adv_{PRG}(A, G)$ is defined as

$$Adv_{PRG}(A, G) = \left| \Pr_{K \leftarrow \{0,1\}^k}[A(G(K)) = 1] - \Pr_{R \leftarrow \{0,1\}^n}[A(R)] = 1 \right|$$

  - <u>Observations</u>:
    - $0 \leq Adv_{PRG}(A, G) \leq 1$
    - $Adv_{PRG}(A, G)$ close to 1 means A can distinguish G's output from random.
    - $Adv_{PRG}(A, G)$ close to 0 means A cannot distinguish G's output from random.

- Let $S = \{G(K) | K \in \{0,1\}^k\}$. Let A be an algorithm that outputs 1 iff the input belongs to S.
  - <u>Question</u>: Is A a good statistical test for G?
  - <u>Question</u>: What is the PRG advantage of A?

Stream Ciphers: Indistinguishability

- <u>Definition (Advantage)</u>: The PRG advantage of an algorithm A with respect to a generator G is denoted by $Adv_{PRG}(A, G)$ is defined as

$$Adv_{PRG}(A, G) = \left| \Pr_{K \leftarrow \{0,1\}^k}[A(G(K)) = 1] - \Pr_{R \leftarrow \{0,1\}^n}[A(R)] = 1 \right|$$

  - <u>Observations</u>:
    - $0 \leq Adv_{PRG}(A, G) \leq 1$
    - $Adv_{PRG}(A, G)$ close to 1 means A can distinguish G's output from random.
    - $Adv_{PRG}(A, G)$ close to 0 means A cannot distinguish G's output from random.

- Let G be a generator such that the 5th bit of the output is 1 in 3/4 of the input seeds. Let A be an algorithm that outputs 1 iff the fifth bit of its input string is 1.
  - <u>Question</u>: Is G a good PRG in the sense of indistinguishability?
  - <u>Question</u>: What is the PRG advantage of A?

Stream Ciphers: Indistinguishability

- <u>Definition ((t, ϵ)-indistinguishable PRG)</u>: A function $G: \{0,1\}^k \rightarrow \{0,1\}^n$ is said to be (t, ϵ)-indistinguishable pseudorandom generator if for all algorithms that run in time at most t, we have:

$$Adv_{PRG}(A, G) \leq \epsilon$$

Stream Ciphers: Unpredictability

- <u>Definition ((t, ϵ)-unpredictable PRG)</u>: A function $G: \{0,1\}^k \to \{0,1\}^n$ is called (t, ϵ)-unpredictable pseudorandom generator of for all algorithms A that run in time at most t and for all $i \in \{1, \dots, n-1\}$, we have: $\Pr\left[A(G(K)[1 \dots i]) = G(K)[i+1]\right] \leq \frac{1}{2} + \epsilon$.

Stream Ciphers: Indistinguishability Vs Unpredictability

- Definition ((t,ϵ)-indistinguishable PRG): A function $G:\{0,1\}^k \rightarrow \{0,1\}^n$ is said to be (t,ϵ)-secure Pseudorandom Generator if for all algorithms that run in time at most t, we have:

$$Adv_{PRG}(A,G) \leq \epsilon$$

- Definition ((t,ϵ)-unpredictable PRG): A function $G:\{0,1\}^k \rightarrow \{0,1\}^n$ is called (t,ϵ)-unpredictable pseudorandom generator of for all algorithms A that run in time at most t and for all $i \in \{1,\dots,n-1\}$, we have:
$\Pr\left[A(G(K)[1\dots i]) = G(K)[i+1]\right] \leq \frac{1}{2} + \epsilon.$

- Theorem(indistinguishability implies unpredictability): Let $G:\{0,1\}^k \rightarrow \{0,1\}^n$. If G is a $(t+1,\epsilon)$-indistinguishable PRG, then G is also a (t,ϵ)-unpredictable PRG.
- Proof: We show the contrapositive.
  - Suppose A is an algorithm that runs in time at most t and i be the index such that
  $\Pr\left[A(G(K)[1\dots i]) = G(K)[i+1]\right] > \frac{1}{2} + \epsilon.$
  - Consider algorithm $B(x)$: If $(A(x[1\dots i]) = x[i+1])$, then output 1 else 0.

Stream Ciphers: Indistinguishability Vs Unpredictability

- <u>Theorem(indistinguishability implies unpredictability)</u>: Let $G: \{0,1\}^k \to \{0,1\}^n$. If G is a (t, ϵ)-indistinguishable PRG, then G is also a (t, ϵ)-unpredictable PRG.
- <u>Proof</u>: We show the contrapositive.
  - Suppose A is an algorithm that runs in time at most t and i be the index such that

    $$\Pr\left[A(G(K)[1 \dots i]) = G(K)[i+1]\right] > \frac{1}{2} + \epsilon.$$
  - Consider algorithm $B(x)$: If $(A(x[1 \dots i]) = x[i+1])$, then output 1 else 0.
  - <u>Claim 1</u>: B runs in time $t + 1$.
  - <u>Claim 2</u>: $Adv_{PRG}(B, G) > \epsilon$.
    - <u>Claim 2.1</u>: $\Pr_{K \leftarrow \{0,1\}^k}\left[B\big(G(K)\big) = 1\right] > \frac{1}{2} + \epsilon$.
    - <u>Claim 2.2</u>: $\Pr_{R \leftarrow \{0,1\}^n}\left[B(R) = 1\right] = \frac{1}{2}$.

End