CSE5014 CRYPTOGRAPHY AND NETWORK SECURITY

Dr. Qi Wang

Department of Computer Science and Engineering
Office: Room413, CoE South Tower
Email: wangqi@sustech.edu.cn

Perfect security

- **Definition 1.6** *Perfect secrecy*. An encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is *perfectly secure* if and only if for every two distinct plaintexts $\{x_0, x_1\} \in \mathcal{M}$, and for every strategy used by Eve, if we choose at random $b \in \{0, 1\}$ and a random key $k \in \{0, 1\}^n$, then the probability that Eve guesses x_b after seeing the ciphertext $c = Enc_k(x_b)$ is at most $1/2$.

Perfect security

- **Definition 1.6** *Perfect secrecy*. An encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is *perfectly secure* if and only if for every two distinct plaintexts $\{x_0, x_1\} \in \mathcal{M}$, and for every strategy used by Eve, if we choose at random $b \in \{0, 1\}$ and a random key $k \in \{0, 1\}^n$, then the probability that Eve guesses x_b after seeing the ciphertext $c = Enc_k(x_b)$ is at most $1/2$.

  Theorem 1.10 (Limitations of perfect secrecy) There is no *perfectly secure* encryption schemes (Gen, Enc, Dec) with n-bit plaintexts and $(n-1)$-bit keys.

  - The key is as long as the message
  - Only secure if each key is used to encrypt a single message
  - Trivially broken by a known-plaintext attack

ϵ-Statistical Security

- **Definition 2.1** Let X and Y be two distributions over $\{0,1\}^n$. The *statistical distance* of X and Y, denoted by $\Delta(X, Y)$ is defined to be
$$\max_{T \subseteq \{0,1\}^n} |\Pr[X \in T] - \Pr[Y \in T]|.$$
If $\Delta(X, Y) \le \epsilon$, we say that $X \equiv_\epsilon Y$.

ϵ-Statistical Security

- **Definition 2.1** Let X and Y be two distributions over $\{0,1\}^n$. The *statistical distance* of X and Y, denoted by $\Delta(X,Y)$ is defined to be
  $$\max_{T\subseteq\{0,1\}^n} |\Pr[X\in T]-\Pr[Y\in T]|.$$
  If $\Delta(X,Y)\le\epsilon$, we say that $X\equiv_\epsilon Y$.

  Definition 2.2 *ϵ-Statistical Security*. An encryption scheme (Gen, Enc, Dec) is *ϵ-statistically secure* if for every pair of plaintexts m, m', we have $Enc_{U_n}(m)\equiv_\epsilon Enc_{U_n}(m')$.

ϵ-Statistical Security

- **Lemma 2.3**

$$\Delta(X, Y) = \frac{1}{2} \sum_{w \in Supp(X) \cup Supp(Y)} |Pr[X = w] - Pr[Y = w]|$$

ϵ-Statistical Security

- **Lemma 2.3**

$$\Delta(X,Y) = \frac{1}{2} \sum_{w \in Supp(X) \cup Supp(Y)} |Pr[X = w] - Pr[Y = w]|$$

Observations:

$0 \leq \Delta(X,Y) \leq 1$

$\Delta(X,Y) = 0$ if $X = Y$

$0 \leq \Delta(X,Y) \leq \Delta(X,Z) + \Delta(Z,Y)$

ϵ-Statistical Security

- **Lemma 2.3**

$$\Delta(X, Y) = \frac{1}{2} \sum_{w \in Supp(X) \cup Supp(Y)} |Pr[X = w] - Pr[Y = w]|$$

Observations:

$$0 \le \Delta(X, Y) \le 1$$
$$\Delta(X, Y) = 0 \text{ if } X = Y$$
$$0 \le \Delta(X, Y) \le \Delta(X, Z) + \Delta(Z, Y)$$

Δ is a *metric*.

ϵ-Statistical Security

- **Lemma 2.4** Eve has at most $1/2 + \epsilon$ success probability if and only if for every pair of m_1, m_2,
  $$\Delta(Enc_{U_n}(m_1), Enc_{U_n}(m_2)) \leq 2\epsilon.$$

ϵ-Statistical Security

- **Lemma 2.4** Eve has at most $1/2 + \epsilon$ success probability if and only if for every pair of m_1, m_2,
  $\Delta(Enc_{U_n}(m_1), Enc_{U_n}(m_2)) \leq 2\epsilon$.

  Theorem 2.5 Let (Gen, Enc, Dec) be a valid encryption with $Enc : \{0,1\}^n \times \{0,1\}^{n+1} \rightarrow \{0,1\}^*$. Then there exist plaintexts m_1, m_2 with $\Delta(Enc_{U_n}(m_1), Enc_{U_n}(m_2)) > 1/2$.

Computational Security

- Statistical security does not allow us to break the impossibility result.

Computational Security

- Statistical security does not allow us to break the impossibility result.
  - In real life, people are using encryption with keys shorter than the message size to encrypt all kinds of sensitive information.
  - If the algorithm you use to break the encryption scheme runs in time 2^n , it seems OK since the message may be expired by then ...

Computational Security

- Statistical security does not allow us to break the impossibility result.
  - In real life, people are using encryption with keys shorter than the message size to encrypt all kinds of sensitive information.
  - If the algorithm you use to break the encryption scheme runs in time 2^n , it seems OK since the message may be expired by then ...
- **Idea**: Would be OK if a scheme leaked information with *tiny probability* to eavesdroppers with *bounded computational resources*

Computational Security

- Statistical security does not allow us to break the impossibility result.
  - In real life, people are using encryption with keys shorter than the message size to encrypt all kinds of sensitive information.
  - If the algorithm you use to break the encryption scheme runs in time 2^n , it seems OK since the message may be expired by then ...
- **Idea**: Would be OK if a scheme leaked information with *tiny probability* to eavesdroppers with *bounded computational resources*
  - Allowing security to "fail" with tiny probability
  - Restricting attention to "efficient" attackers

Tiny probability of failure?

- Say security fails with probability 2^{-60}

Tiny probability of failure?

- Say security fails with probability 2^{-60}
  - Should we be concerned about this?

Tiny probability of failure?

- Say security fails with probability 2^{-60}
  - Should we be concerned about this?
  - With probability $> 2^{-60}$, the sender and receiver will both be struck by lightning in the next year …
  - Something that occurs with probability $2^{-60}/sec$ is expected to occur once every 100 billion years

Bounded attackers?

- Consider *brute-force search* of key space; assume one key can be tested per clock cycle

Bounded attackers?

- Consider *brute-force search* of key space; assume one key can be tested per clock cycle
  - Desktop computer $\approx 2^{57}$ keys/year
  - Supercomputer $\approx 2^{80}$ keys/year

Bounded attackers?

- Consider *brute-force search* of key space; assume one key can be tested per clock cycle
  - Desktop computer $\approx 2^{57}$ keys/year
  - Supercomputer $\approx 2^{80}$ keys/year
  - Supercomputer since Big Bang $\approx 2^{112}$ keys
    - Restricting attention to attackers who can try 2^{112} keys is fine!

Bounded attackers?

- Consider *brute-force search* of key space; assume one key can be tested per clock cycle
  - Desktop computer $\approx 2^{57}$ keys/year
  - Supercomputer $\approx 2^{80}$ keys/year
  - Supercomputer since Big Bang $\approx 2^{112}$ keys
    - Restricting attention to attackers who can try 2^{112} keys is fine!

Modern key space: 2^{128} keys or more ...

Computational Security

- Two problems:

Computational Security

- Two problems:

1) While the particular algorithm runs in exponential time, we cannot guarantee that there is no other algorithm is efficient.

2) We need a precise mathematical definition (like *prefect secrecy* definition).

Computational Security

- Two problems:

  1) While the particular algorithm runs in exponential time, we cannot guarantee that there is no other algorithm is efficient.

     e.g., the substitution cipher has a huge key space, but can be broken efficiently.

  2) We need a precise mathematical definition (like *prefect secrecy* definition).

Computational Security

- Two problems:

  1) While the particular algorithm runs in exponential time, we cannot guarantee that there is no other algorithm is efficient.

  e.g., the substitution cipher has a huge key space, but can be broken efficiently.

  2) We need a precise mathematical definition (like *prefect secrecy* definition).

  "Breaking E is very hard"?
  "Problem P cannot be solved in reasonable time"?

Computational Security

- Two problems:

1) While the particular algorithm runs in exponential time, we cannot guarantee that there is no other algorithm is efficient.

e.g., the substitution cipher has a huge key space, but can be broken efficiently.

2) We need a precise mathematical definition (like *prefect secrecy* definition).

"Breaking E is very hard"?
"Problem P cannot be solved in reasonable time"?

$\mathcal{Q}$: How do we model the resources of Eve (the adversary)?

Perfect indistinguishability

- **Definition 1.6** *Perfect secrecy*. An encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is *perfectly secure* if and only if for every two distinct plaintexts $\{m_0, m_1\} \in \mathcal{M}$, and for every strategy used by Eve, if we choose at random $b \in \{0, 1\}$ and a random key $k \in \{0, 1\}^n$, then the probability that Eve guesses m_b after seeing the ciphertext $c = Enc_k(m_b)$ is at most $1/2$.

Perfect indistinguishability

- **Definition 1.6** *Perfect secrecy*. An encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is *perfectly secure* if and only if for every two distinct plaintexts $\{m_0, m_1\} \in \mathcal{M}$, and for every strategy used by Eve, if we choose at random $b \in \{0, 1\}$ and a random key $k \in \{0, 1\}^n$, then the probability that Eve guesses m_b after seeing the ciphertext $c = Enc_k(m_b)$ is at most $1/2$.
- Let $\Pi = (Gen, Enc, Dec)$ be an enryption scheme with message space $\mathcal{M}$, and A an adversary

Perfect indistinguishability

- **Definition 1.6** *Perfect secrecy*. An encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is *perfectly secure* if and only if for every two distinct plaintexts $\{m_0, m_1\} \in \mathcal{M}$, and for every strategy used by Eve, if we choose at random $b \in \{0, 1\}$ and a random key $k \in \{0, 1\}^n$, then the probability that Eve guesses m_b after seeing the ciphertext $c = Enc_k(m_b)$ is at most $1/2$.
- Let $\Pi = (Gen, Enc, Dec)$ be an enryption scheme with message space $\mathcal{M}$, and A an adversary

  Define a randomized experiment $PrivK_{A,\Pi}$:

  1. A outputs $m_0, m_1 \in \mathcal{M}$
  2. $k \leftarrow Gen$, $b \leftarrow \{0, 1\}$, $c \leftarrow Enc_k(m_b)$
  3. $b' \leftarrow A(c)$

Perfect indistinguishability

- **Definition 1.6** *Perfect secrecy*. An encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is *perfectly secure* if and only if for every two distinct plaintexts $\{m_0, m_1\} \in \mathcal{M}$, and for every strategy used by Eve, if we choose at random $b \in \{0, 1\}$ and a random key $k \in \{0, 1\}^n$, then the probability that Eve guesses m_b after seeing the ciphertext $c = Enc_k(m_b)$ is at most $1/2$.
- Let $\Pi = (Gen, Enc, Dec)$ be an enryption scheme with message space $\mathcal{M}$, and A an adversary

  Define a randomized experiment $PrivK_{A,\Pi}$:

  1. A outputs $m_0, m_1 \in \mathcal{M}$
  2. $k \leftarrow Gen$, $b \leftarrow \{0, 1\}$, $c \leftarrow Enc_k(m_b)$
  3. $b' \leftarrow A(c)$

  Adversary A *succeeds* if $b = b'$, and we say the experiment evaluates to 1 in this case.

Perfect indistinguishability

- Let $\Pi = (Gen, Enc, Dec)$ be an enryption scheme with message space $\mathcal{M}$, and A an adversary

  Define a randomized experiment $PrivK_{A,\Pi}$:

  1. A outputs $m_0, m_1 \in \mathcal{M}$
  2. $k \leftarrow Gen$, $b \leftarrow \{0, 1\}$, $c \leftarrow Enc_k(m_b)$
  3. $b' \leftarrow A(c)$

  Adversary A *succeeds* if $b = b'$, and we say the experiment evaluates to 1 in this case.

Perfect indistinguishability

- Let $\Pi = (Gen, Enc, Dec)$ be an enryption scheme with message space $\mathcal{M}$, and A an adversary

  Define a randomized experiment $PrivK_{A,\Pi}$:

  1. A outputs $m_0, m_1 \in \mathcal{M}$
  2. $k \leftarrow Gen$, $b \leftarrow \{0, 1\}$, $c \leftarrow Enc_k(m_b)$
  3. $b' \leftarrow A(c)$

  Adversary A *succeeds* if $b = b'$, and we say the experiment evaluates to 1 in this case.

  Π is *perfectly indistinguishable* if for all attackers (algorithms) A, it holds that $\Pr[PrivK_{A,\Pi} = 1] \leq 1/2$

Perfect indistinguishability

- Let $\Pi = (Gen, Enc, Dec)$ be an enryption scheme with message space $\mathcal{M}$, and A an adversary

  Define a randomized experiment $PrivK_{A,\Pi}$:

  1. A outputs $m_0, m_1 \in \mathcal{M}$
  2. $k \leftarrow Gen$, $b \leftarrow \{0, 1\}$, $c \leftarrow Enc_k(m_b)$
  3. $b' \leftarrow A(c)$

  Adversary A *succeeds* if $b = b'$, and we say the experiment evaluates to 1 in this case.

  Π is *perfectly indistinguishable* if for all attackers (algorithms) A, it holds that $\Pr[PrivK_{A,\Pi} = 1] \leq 1/2$

  Claim: Π is *perfectly indistinguishable* $\Leftrightarrow$ Π is *perfectly secure*

Computational security?

- **Idea**: relax *perfect indistinguishability*

Computational security?

- **Idea**: relax *perfect indistinguishability*

  Two approaches

  – *Concrete* security

  – *Asympototic* security

Computational indistinguishability

- (t, ϵ)-*indistinguishability* (concrete)
  - Security may fail with probability $\leq \epsilon$
  - Restrict attention to attackers running in time $\leq t$

Computational indistinguishability

- (t, ϵ)-*indistinguishability* (concrete)
  - Security may fail with probability $\leq \epsilon$
  - Restrict attention to attackers running in time $\leq t$

- Π is (t, ϵ)-indistinguishable if for all attackers A running in time at most t, it holds that

$$\Pr[PrivK_{A,\Pi} = 1] \leq 1/2 + \epsilon$$

Computational indistinguishability

- (t, ϵ)-*indistinguishability* (concrete)
  - Security may fail with probability $\leq \epsilon$
  - Restrict attention to attackers running in time $\leq t$

- Π is (t, ϵ)-indistinguishable if for all attackers A running in time at most t, it holds that
$$\Pr[PrivK_{A,\Pi} = 1] \leq 1/2 + \epsilon$$

Does not lead to a clean theory ...
  - Sensitive to exact computational model
  - Π can be (t, ϵ)-secure for many choices of t, ϵ

Computational indistinguishability

- Introduce *security parameter* n (asymptotic)
  - For now, can view n as the key length
  - Fixed by honest parties at initialization
  - Known by adversary

Computational indistinguishability

- Introduce *security parameter* n (asymptotic)
  - For now, can view n as the key length
  - Fixed by honest parties at initialization
  - Known by adversary

Measure running time of all parties, and the success probability of the adversary, as functions of n

Computational indistinguishability

- Introduce *security parameter* n (asymptotic)
  - For now, can view n as the key length
  - Fixed by honest parties at initialization
  - Known by adversary

Measure running time of all parties, and the success probability of the adversary, as functions of n

Computational indistinguishability:

- Security may fail with probability *negligible* in n
- Restrict attention to attackers running in time (at most) polynomial in n

Definitions

- A function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ is (at most) *polynomial* if there exists c s.t. $f(n) < n^c$ for large enough n.

  A function $f : \mathbb{Z}^+ \to [0, 1]$ is *negligible* if every polynomial p it holds that $f(n) < 1/p(n)$ for large enough n.

  – Typical example: $f(n) = poly(n) \cdot 2^{-cn}$

Definitions

- A function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ is (at most) *polynomial* if there exists c s.t. $f(n) < n^c$ for large enough n.

  A function $f : \mathbb{Z}^+ \to [0, 1]$ is *negligible* if every polynomial p it holds that $f(n) < 1/p(n)$ for large enough n.

  – Typical example: $f(n) = poly(n) \cdot 2^{-cn}$
- "Efficient" $=$ "(probabilistic) polynomial-time (**PPT**)" borrowed from *complexity theory*

Definitions

- A function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ is (at most) *polynomial* if there exists c s.t. $f(n) < n^c$ for large enough n.

  A function $f : \mathbb{Z}^+ \to [0, 1]$ is *negligible* if every polynomial p it holds that $f(n) < 1/p(n)$ for large enough n.

  – Typical example: $f(n) = poly(n) \cdot 2^{-cn}$
- "Efficient" $=$ "(probabilistic) polynomial-time (**PPT**)" borrowed from *complexity theory*
- Convenient closure properties
  – $poly * poly = poly$
    – Poly-many calls to PPT subroutine (with poly-size input) is still PPT
  – $poly * negl = negl$
    – Poly-many calls to subroutine that fails with negligible probability fails with negligible probability overall

(Re)defining encryption

- A *private-key encryption scheme* is defined by three PPT algorithms (Gen, Enc, Dec):
  - Gen: takes as input 1^n; outputs k
  - Enc: takes as input a key k and message $m \in \{0,1\}^*$; outputs ciphertext c: $\quad c \leftarrow Enc_k(m)$
  - Dec: takes key k and ciphertext c as input; outputs a message m or "error" ($\perp$)

Computational indistinguishability (asymptotic)

- Fix Π, A
  Define a randomized experiment $PrivK_{A,\Pi}(n)$:
  1. $A(1^n)$ outputs $m_0, m_1 \in \{0,1\}^*$ of equal length
  2. $k \leftarrow Gen(1^n)$, $b \leftarrow \{0,1\}$, $c \leftarrow Enc_k(m_b)$
  3. $b' \leftarrow A(c)$

  Adversary A *succeeds* if $b = b'$, and we say the experiment evaluates to 1 in this case.

Computational indistinguishability (asymptotic)

- Fix Π, A
  Define a randomized experiment $PrivK_{A,\Pi}(n)$:
  1. $A(1^n)$ outputs $m_0, m_1 \in \{0,1\}^*$ of equal length
  2. $k \leftarrow Gen(1^n)$, $b \leftarrow \{0,1\}$, $c \leftarrow Enc_k(m_b)$
  3. $b' \leftarrow A(c)$

  Adversary A *succeeds* if $b = b'$, and we say the experiment evaluates to 1 in this case.

  Definition 3.1 Π is *computationally indistinguishable* (aka *EAV-secure*) if for all PPT attackers (algorithms) A, there is a *negligible* function ϵ such that

$$\Pr[PrivK_{A,\Pi}(n) = 1] \leq 1/2 + \epsilon(n)$$

Example

- Consider a scheme where the best attack is *brute-force search* over the key space, and $Gen(1^n)$ generates a uniform n-bit key
  - So if A runs in time $t(n)$, then
    $\Pr[PrivK_{A,\Pi}(n) = 1] < 1/2 + O(t(n)/2^n)$

Example

- Consider a scheme where the best attack is *brute-force search* over the key space, and $Gen(1^n)$ generates a uniform n-bit key
  - So if A runs in time $t(n)$, then
    $\Pr[PrivK_{A,\Pi}(n) = 1] < 1/2 + O(t(n)/2^n)$
  - The scheme is EAV-secure: for any polynomial t, the function $t(n)/2^n$ is negligible.

Example

- Consider a scheme and a particular attacker A that runs for n^3 minutes and breaks the scheme with probability $2^{40}2^{-n}$
  - This does not contradict asymptotic security

Example

- Consider a scheme and a particular attacker A that runs for n^3 minutes and breaks the scheme with probability $2^{40}2^{-n}$
  - This does not contradict asymptotic security
  - What about real-world security (against this attacker)?
    - $n = 40$: A breaks with prob. 1 in 6 weeks
    - $n = 50$: A breaks with prob. $1/1000$ in 3 months
    - $n = 500$: A breaks with prob. 2^{-500} in 200 years

Example

- What happens when computers get faster?
  - – Consider a scheme that takes time n^2 to run but time 2^n to break with prob. 1

Example

- What happens when computers get faster?

  – Consider a scheme that takes time n^2 to run but time 2^n to break with prob. 1

  What if computers get $4\times$ faster?

  – Users double n; maintain same running time

  – Attacker's work is (roughly) squared!

Encryption and plaintext length

- In practice, we want encryption schemes that can encrypt arbitrary-length messages.

Encryption and plaintext length

- In practice, we want encryption schemes that can encrypt arbitrary-length messages.

- In general, encryption does not hide the plaintext length
  - The definition takes this into account by requiring m_0, m_1 to have the same length.

Encryption and plaintext length

- In practice, we want encryption schemes that can encrypt arbitrary-length messages.

- In general, encryption does not hide the plaintext length
  - The definition takes this into account by requiring m_0, m_1 to have the same length.

- But leaking plaintext length can often lead to problems in the real world!
  - Databases searches
  - Encrypting compressed data

Micali & Goldwasser

Silvio Micali

Shafi Goldwasser

1984: semantic security, indistinguishability
(Turing Award 2012)

Micali & Blum

Silvio Micali

Manuel Blum

1984: defined notion of pseudo-random generator
(Turing Award 1995)

Pseudorandomness

- Important building block for computationally secure encryption

Pseudorandomness

- Important building block for computationally secure encryption
- What does "random" mean?

Pseudorandomness

- Important building block for computationally secure encryption
- What does "random" mean?
- Which of the following is a uniform string?
  - 0101010101010101
  - 0010111011100110
  - 0000000000000000

Pseudorandomness

- Important building block for computationally secure encryption
- What does "random" mean?
- Which of the following is a uniform string?
  - 0101010101010101
  - 0010111011100110
  - 0000000000000000
- If we generate a uniform 16-bit string, each of the above occurs with probability 2^{-16}

What does “pseudorandom” mean?

- Informal: Cannot be distinguished from uniform (“random”)

What does "pseudorandom" mean?

- Informal: Cannot be distinguished from uniform ("random")
- Which of the following is pseudorandom?
  - 0101010101010101
  - 0010111011100110
  - 0000000000000000

What does "pseudorandom" mean?

- Informal: Cannot be distinguished from uniform ("random")
- Which of the following is pseudorandom?
  - 0101010101010101
  - 0010111011100110
  - 0000000000000000
- *Pseudorandomness* is a property of a *distribution*, not a string.

Pseudorandomness

- Fix some distribution D on n-bit strings
  - $x \leftarrow D$ means "sample x according to D"

Pseudorandomness

- Fix some distribution D on n-bit strings
  - $x \leftarrow D$ means "sample x according to D"

- Historically, D was considered *pseudorandom* if it "*passed a bunch of statistical tests*"
  - $\Pr_{x \leftarrow D}[1^{st} \text{ bit of } x \text{ is } 1] \approx 1/2$
  - $\Pr_{x \leftarrow D}[\text{parity of } x \text{ is } 1] \approx 1/2$
  - $\Pr_{x \leftarrow D}[A_i(x) = 1] \approx \Pr_{x \leftarrow U_n}[A_i(x) = 1]$
    for $i = 1, \ldots, 20$

Pseudorandomness

- Fix some distribution D on n-bit strings
  - $x \leftarrow D$ means "sample x according to D"

- Historically, D was considered *pseudorandom* if it "*passed a bunch of statistical tests*"
  - $\Pr_{x \leftarrow D}[1^{st} \text{ bit of } x \text{ is } 1] \approx 1/2$
  - $\Pr_{x \leftarrow D}[\text{parity of } x \text{ is } 1] \approx 1/2$
  - $\Pr_{x \leftarrow D}[A_i(x) = 1] \approx \Pr_{x \leftarrow U_n}[A_i(x) = 1]$ for $i = 1, \ldots, 20$

  This is not sufficient, since it is not possible to know what statistical test an attacker will use.

Pseudorandomness

- Cryptographic definition of *pseudorandomness*
  - D is *pseudorandom* if it passes all *efficient* statistical tests

Pseudorandomness

- Cryptographic definition of *pseudorandomness*
  - D is *pseudorandom* if it passes all *efficient* statistical tests

(Concrete) Let D be a distribution on p-bit strings. D is (t, ϵ)-*pseudorandom* if for all A running in time at most t,

$$|\Pr_{x \leftarrow D}[A(x) = 1] - \Pr_{x \leftarrow U_p}[A(x) = 1]| \leq \epsilon$$

Pseudorandomness

- Cryptographic definition of *pseudorandomness*
  - D is *pseudorandom* if it passes all *efficient* statistical tests

(Concrete) Let D be a distribution on p-bit strings. D is (t, ϵ)*-pseudorandom* if for all A running in time at most t,

$$|\Pr_{x \leftarrow D}[A(x) = 1] - \Pr_{x \leftarrow U_p}[A(x) = 1]| \leq \epsilon$$

(Asymptotic) *Security parameter n*, polynomial p

Definiton 3.2 Let D_n be a distribution over $p(n)$-bit strings. $\{D_n\}$ is *pseudorandom* if for all probabilistic, polynomial-time (PPT) distinguishers A, there is a negligible function ϵ such that

$$|\Pr_{x \leftarrow D_n}[A(x) = 1] - \Pr_{x \leftarrow U_{p(n)}}[A(x) = 1]| \leq \epsilon(n)$$

Pseudorandom generators (PRGs)

- A *PRG* is an efficient, deterministic algorithm that expands a *short*, *uniform seed* into a *longer*, *pseudorandom* output

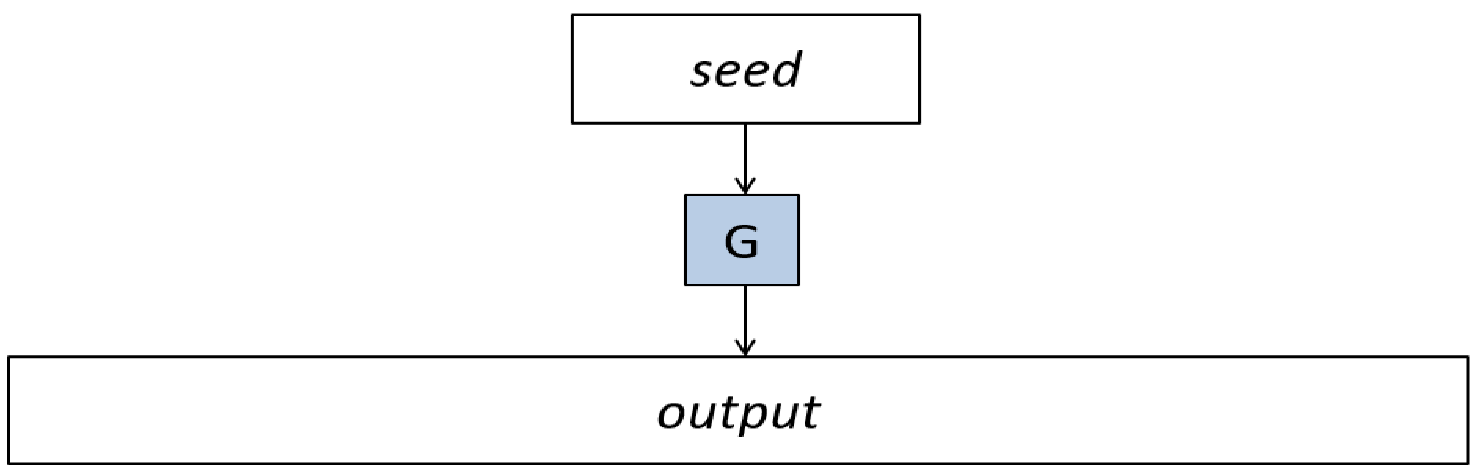
Pseudorandom generators (PRGs)

- A *PRG* is an efficient, deterministic algorithm that expands a *short*, *uniform seed* into a *longer*, *pseudorandom* output
  - Useful whenever you have a "small" number of true random bits, and want lots of "random-looking" bits

Pseudorandom generators (PRGs)

- A *PRG* is an efficient, deterministic algorithm that expands a *short*, *uniform seed* into a *longer*, *pseudorandom* output
  - Useful whenever you have a “small” number of true random bits, and want lots of “random-looking” bits

Let G be a deterministic, poly-time algorithm that is *expanding*, i.e., $|G(x)| = p(|x|) > |x|$.



Pseudorandom generators (PRGs)

- A *PRG* is an efficient, deterministic algorithm that expands a *short*, *uniform seed* into a *longer*, *pseudorandom* output
  - Useful whenever you have a "small" number of true random bits, and want lots of "random-looking" bits

Let G be a deterministic, poly-time algorithm that is *expanding*, i.e., $|G(x)| = p(|x|) > |x|$.

G defines a sequence of distributions.

- D_n = the distribution on $p(n)$-bit strings defined by choosing $x \leftarrow U_n$ and outputting $G(x)$
- $\Pr_{D_n}[y] = \Pr_{U_n}[G(x) = y] = \sum_{x:\ G(x)=y} \Pr_{U_n}[x]$

$$= \sum_{x:\ G(x)=y} 2^{-n}$$

$$= |\{x :\ G(x) = y\}|/2^n$$

PRGs

- For all efficient distinguishers A, there is a negligible function ϵ such that
  $$|\Pr_{x \leftarrow U_n}[A(G(x)) = 1] - \Pr_{y \leftarrow U_{p(n)}}[A(y) = 1]| \leq \epsilon(n)$$

PRGs

- For all efficient distinguishers A, there is a negligible function ϵ such that
  $$|\Pr_{x \leftarrow U_n}[A(G(x)) = 1] - \Pr_{y \leftarrow U_{p(n)}}[A(y) = 1]| \leq \epsilon(n)$$

  No efficient A can distinguish whether it is given $G(x)$ (for uniform x) or a uniform string y!

PRGs

- For all efficient distinguishers A, there is a negligible function ϵ such that
  $$|\Pr_{x \leftarrow U_n}[A(G(x)) = 1] - \Pr_{y \leftarrow U_{p(n)}}[A(y) = 1]| \le \epsilon(n)$$
  No efficient A can distinguish whether it is given $G(x)$ (for uniform x) or a uniform string y!
- PRGs are limited
  - They have fixed-length output
  - They produce the entire output in "one shot"
  - In practice, PRGs are based on *stream ciphers*
  - Can be viewed as producing an "unbounded" stream of pseudorandom bits, on demand
  - Will be revisited later

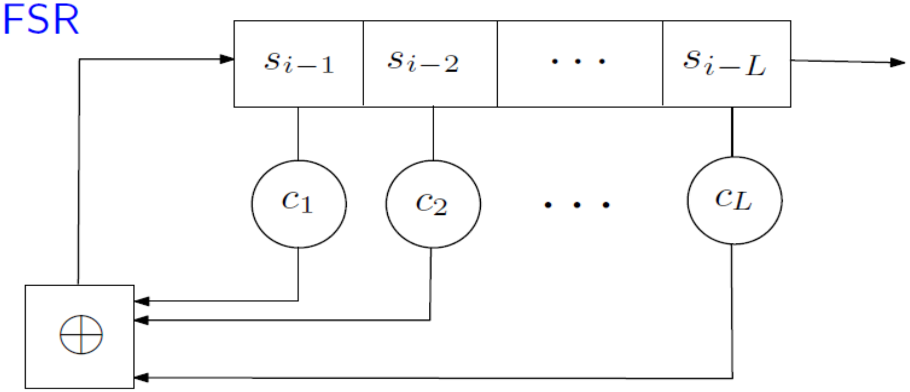
Do PRGs/stream ciphers exist?

- We don't know ...
  - Would imply $P \neq NP$
  - We will assume certain algorithms are PRGs
  - Can construct PRGs from weaker assumptions (later)

Do PRGs/stream ciphers exist?

- We don't know ...
  - Would imply $P \neq NP$
  - We will assume certain algorithms are PRGs
  - Can construct PRGs from weaker assumptions (later)

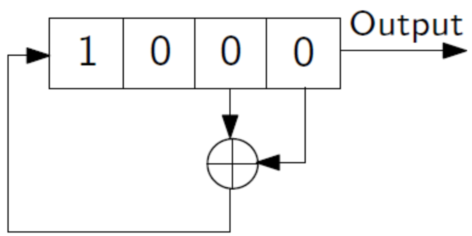
LFSR



Linear Feedback Shift Register (LFSR)

Example

$\mathbf{s} = (000100110101111)^{15}$



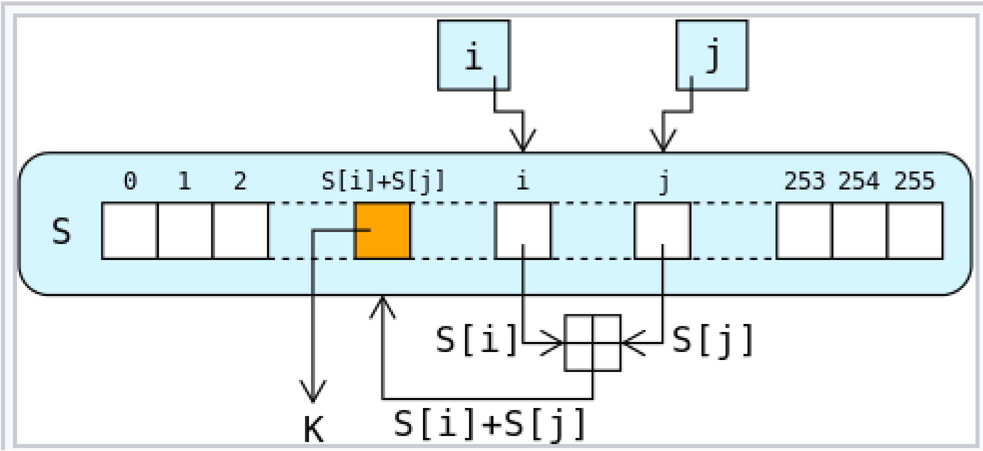
$\mathrm{LC}(\mathbf{s}) = 4$

$P_{\mathbf{s}}(x) = x^4 + x + 1$

Practical "PRGs"

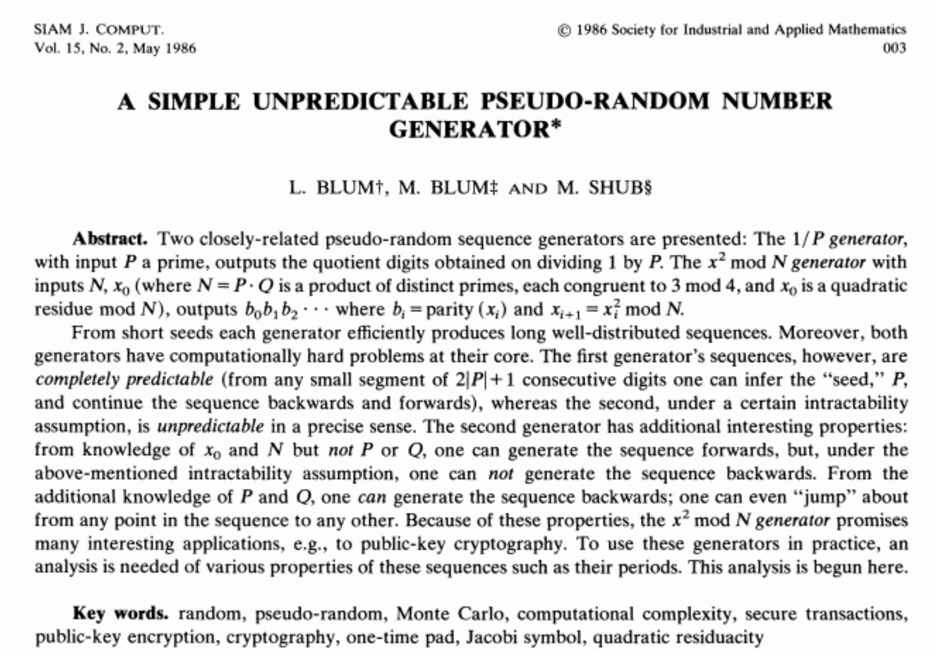
- RC4

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
endwhile
```



Blum-Blum-Shub

```
num_outputted = 0;
while num_outputted < m:
    X := X*X mod N
    num_outputted := num_outputted + 1
    output least-significant-bit(X)
```



SIAM J. COMPUT.
Vol. 15, No. 2, May 1986

A SIMPLE UNPREDICTABLE PSEUDO-RANDOM NUMBER GENERATOR*

L. BLUM†, M. BLUM‡ AND M. SHUB§

Abstract. Two closely-related pseudo-random sequence generators are presented: The $1/P$ *generator*, with input P a prime, outputs the quotient digits obtained on dividing 1 by P. The $x^2 \bmod N$ *generator* with inputs N, x_0 (where $N = P \cdot Q$ is a product of distinct primes, each congruent to 3 mod 4, and x_0 is a quadratic residue mod N), outputs $b_0 b_1 b_2 \cdots$ where $b_i = \text{parity}(x_i)$ and $x_{i+1} = x_i^2 \bmod N$.

From short seeds each generator efficiently produces long well-distributed sequences. Moreover, both generators have computationally hard problems at their core. The first generator's sequences, however, are *completely predictable* (from any small segment of $2|P|+1$ consecutive digits one can infer the "seed," P, and continue the sequence backwards and forwards), whereas the second, under a certain intractability assumption, is *unpredictable* in a precise sense. The second generator has additional interesting properties: from knowledge of x_0 and N but *not* P or Q, one can generate the sequence forwards, but, under the above-mentioned intractability assumption, one can *not* generate the sequence backwards. From the additional knowledge of P and Q, one *can* generate the sequence backwards; one can even "jump" about from any point in the sequence to any other. Because of these properties, the $x^2 \bmod N$ *generator* promises many interesting applications, e.g., to public-key cryptography. To use these generators in practice, an analysis is needed of various properties of these sequences such as their periods. This analysis is begun here.

Key words. random, pseudo-random, Monte Carlo, computational complexity, secure transactions, public-key encryption, cryptography, one-time pad, Jacobi symbol, quadratic residuacity

Where things stand

- We saw that there are some inherent limitations if we want *perfect security*
  - In particular, key must be as long as the message

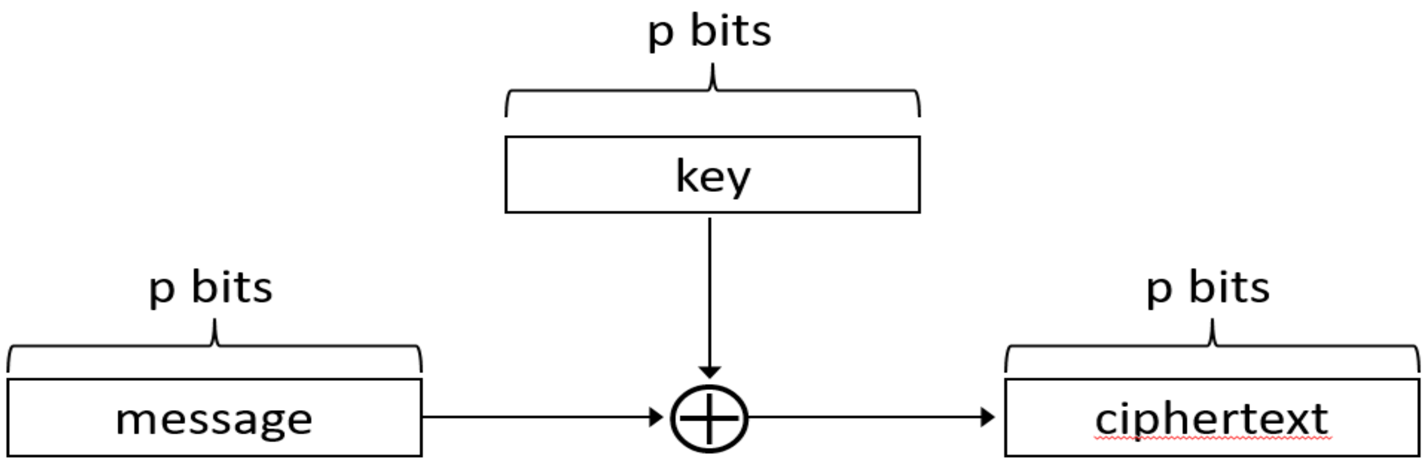
Where things stand

- We saw that there are some inherent limitations if we want *perfect security*
  - In particular, key must be as long as the message

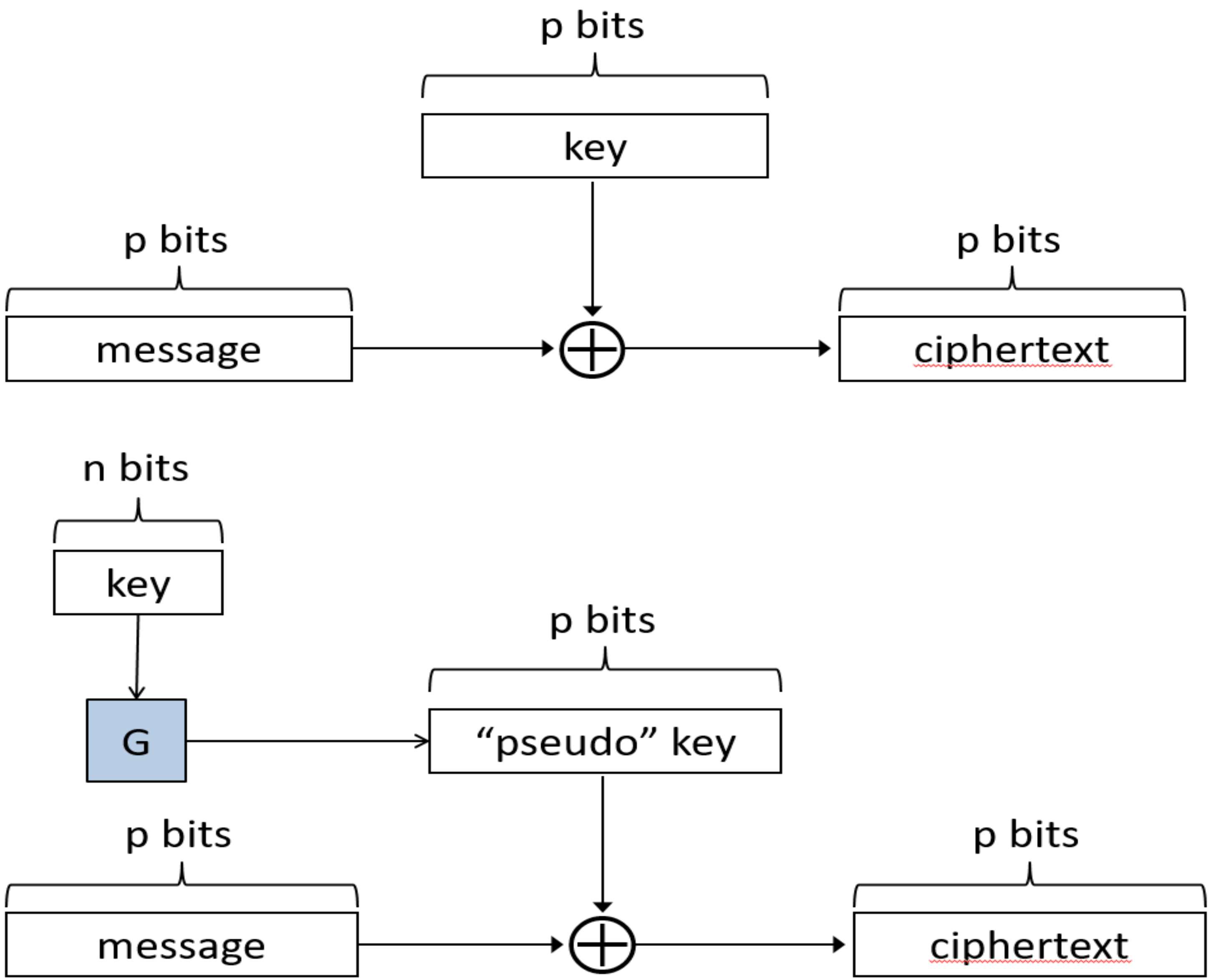
  We defined *computational security*, a relaxed notion of security

  $\mathcal{Q}$: Can we overcome prior limitations?

Recall: one-time pad



Recall: one-time pad



Pseudo one-time pad

- Let G be a deterministic, with $|G(k)| = p(|k|)$

  $Gen(1^n)$: output uniform n-bit key k

  – Security parameter $n \Rightarrow$ message space $\{0,1\}^{p(n)}$

  $Enc_k(m)$: output $G(k) \oplus m$

  $Dec_k(m)$: output $G(k) \oplus c$

Pseudo one-time pad

- Let G be a deterministic, with $|G(k)| = p(|k|)$

  $Gen(1^n)$: output uniform n-bit key k

  – Security parameter $n \Rightarrow$ message space $\{0,1\}^{p(n)}$

  $Enc_k(m)$: output $G(k) \oplus m$

  $Dec_k(m)$: output $G(k) \oplus c$

- Would like to be able to prove *computational security*
  – Based on the assumption that G is a PRG

Proof by reduction

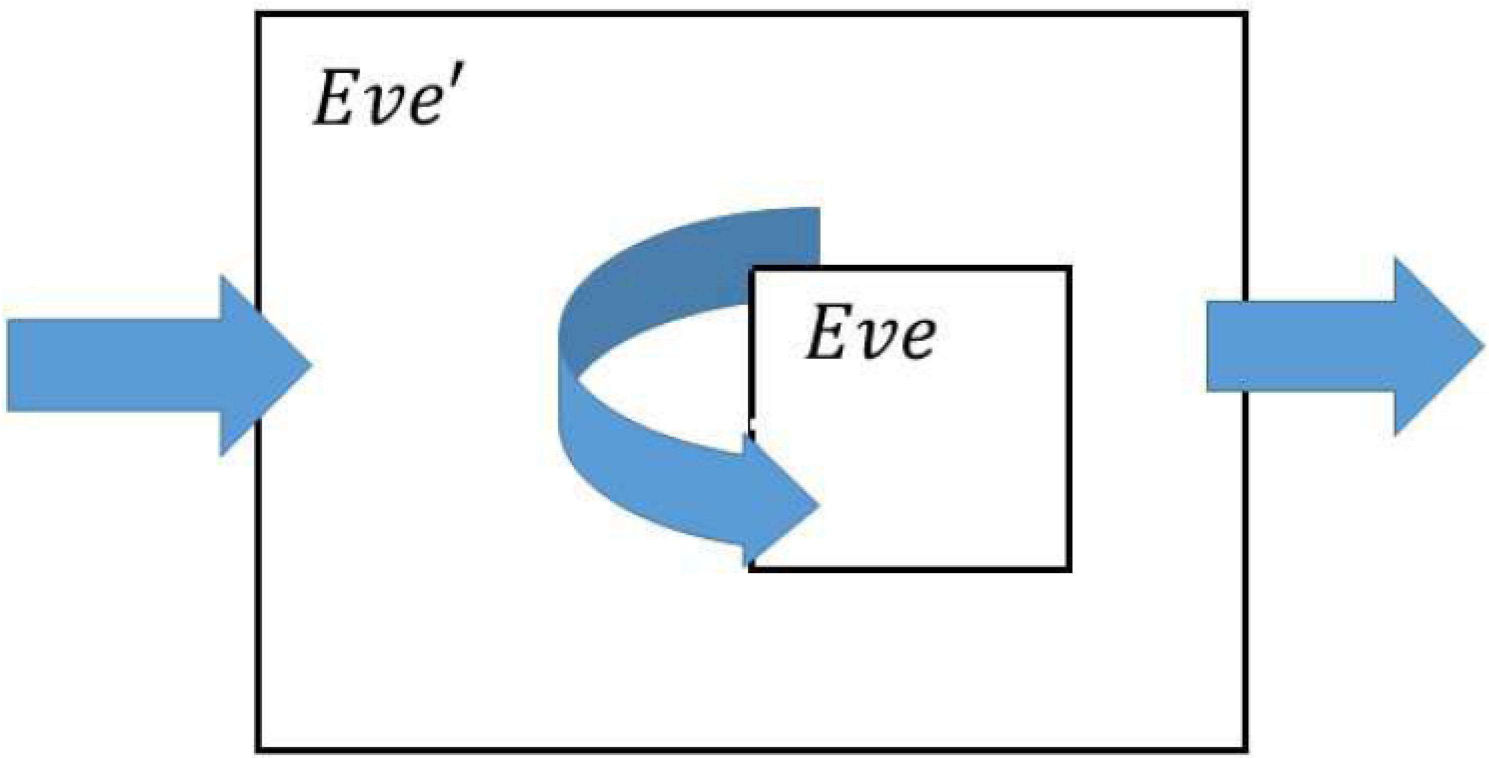


Figure 2.1: We show that the security of S' implies the security of S by transforming an adversary *Eve* breaking S into an adversary *Eve*′ breaking S'

Eve breaks $S \rightarrow$ Eve' breaks S'
S' is secure $\rightarrow S$ is secure

Proof by reduction

- 1. Assume that G is a *PRG*

  2. Assume toward a contradiction that there is an efficient attacker A who "breaks" the pseudo-OTP scheme

  3. Use A as a subroutine to build an efficient D that "breaks" *pseudorandomness* of G

Proof by reduction

- 1. Assume that G is a *PRG*

  2. Assume toward a contradiction that there is an efficient attacker A who "breaks" the pseudo-OTP scheme

  3. Use A as a subroutine to build an efficient D that "breaks" *pseudorandomness* of G

  – By assumption, no such D exists!

  $\Rightarrow$ No such A can exist

Proof by reduction

- 1. Assume that G is a *PRG*

  2. Assume toward a contradiction that there is an efficient attacker A who "breaks" the pseudo-OTP scheme

  3. Use A as a subroutine to build an efficient D that "breaks" *pseudorandomness* of G

  – By assumption, no such D exists!

  $\Rightarrow$ No such A can exist

  Theorem 3.3 If G is a pseudorandom generator (PRG), then the pseudo one-time pad (pseudo-OTP) Π is *EAV-secure* (i.e., *computationally secure*)

Next Lecture

- Pseudorandom functions, block ciphers ...