Cryptology

Sabyasachi Karati

Assistant Professor
Cryptology and Security Research Unit (C.S.R.U)
R. C. Bose Centre for Cryptology and Security
Indian Statistical Institute (ISI)
Kolkata, India

Lecture 05

Pseudo-Random Generator and Stream Cipher

Pseudorandomness

- A pseudorandom string is a string that looks like a uniformly distributed string, as long as the entity that is looking runs in polynomial time.

Pseudorandomness

- A pseudorandom string is a string that looks like a uniformly distributed string, as long as the entity that is looking runs in polynomial time.
- Just as indistinguishability can be viewed as a computational relaxation of perfect secrecy, pseudorandomness is a computational relaxation of true randomness.

Pseudorandomness

- A pseudorandom string is a string that looks like a uniformly distributed string, as long as the entity that is looking runs in polynomial time.
- Just as indistinguishability can be viewed as a computational relaxation of perfect secrecy, pseudorandomness is a computational relaxation of true randomness.
- **Conceptual point:**
  - No fixed string can be said to be pseudorandom.

Pseudorandomness

- A pseudorandom string is a string that looks like a uniformly distributed string, as long as the entity that is looking runs in polynomial time.
- Just as indistinguishability can be viewed as a computational relaxation of perfect secrecy, pseudorandomness is a computational relaxation of true randomness.
- **Conceptual point:**
  - No fixed string can be said to be pseudorandom.
  - Pseudorandomness actually refers to a distribution on strings.

Pseudorandomness

- A pseudorandom string is a string that looks like a uniformly distributed string, as long as the entity that is looking runs in polynomial time.
- Just as indistinguishability can be viewed as a computational relaxation of perfect secrecy, pseudorandomness is a computational relaxation of true randomness.
- **Conceptual point:**
  - No fixed string can be said to be pseudorandom.
  - Pseudorandomness actually refers to a distribution on strings.
  - A distribution $\mathfrak{D}$ over strings of length ℓ is pseudorandom means that $\mathfrak{D}$ is indistinguishable from the uniform distribution over strings of length ℓ.

Pseudorandomness

- A pseudorandom string is a string that looks like a uniformly distributed string, as long as the entity that is looking runs in polynomial time.
- Just as indistinguishability can be viewed as a computational relaxation of perfect secrecy, pseudorandomness is a computational relaxation of true randomness.
- **Conceptual point:**
  - No fixed string can be said to be pseudorandom.
  - Pseudorandomness actually refers to a distribution on strings.
  - A distribution $\mathfrak{D}$ over strings of length ℓ is pseudorandom means that $\mathfrak{D}$ is indistinguishable from the uniform distribution over strings of length ℓ.
  - More precisely, it is infeasible for any polynomial-time algorithm to tell whether it is given a string sampled according to $\mathfrak{D}$ or an ℓ-bit string chosen uniformly at random.

Pseudorandomness

Why do we need Pseudorandomness?

- If a ciphertext looks random, then no adversary can learn any information from it about the plaintext.

Pseudorandomness

Why do we need Pseudorandomness?

- If a ciphertext looks random, then no adversary can learn any information from it about the plaintext.
- This is the exact intuition that lies behind the perfect secrecy of the one-time pad (OTP).

Pseudorandomness

Why do we need Pseudorandomness?

- If a ciphertext looks random, then no adversary can learn any information from it about the plaintext.
- This is the exact intuition that lies behind the perfect secrecy of the one-time pad (OTP).
- In OTP, the ciphertext is uniformly distributed.
- The OTP worked by computing the XOR of a random string (the key) with the plaintext.

Pseudorandomness

Why do we need Pseudorandomness?

- If a ciphertext looks random, then no adversary can learn any information from it about the plaintext.
- This is the exact intuition that lies behind the perfect secrecy of the one-time pad (OTP).
- In OTP, the ciphertext is uniformly distributed.
- The OTP worked by computing the XOR of a random string (the key) with the plaintext.
- If a pseudorandom string were used instead, this should not make any noticeable difference to a polynomial-time observer.
- Thus, security should still hold for polynomial-time adversaries.

Pseudo-Random Generator (PRG)

- PRG: An efficient, deterministic algorithm for transforming a short, uniform string called the *seed* into a longer, "uniform-looking" (or "pseudorandom") output string.

Pseudo-Random Generator (PRG)

- PRG: An efficient, deterministic algorithm for transforming a short, uniform string called the *seed* into a longer, "uniform-looking" (or "pseudorandom") output string.
- PRG uses a small amount of true randomness in order to generate a large amount of pseudorandomness.

Pseudo-Random Generator (PRG)

- PRG: An efficient, deterministic algorithm for transforming a short, uniform string called the *seed* into a longer, "uniform-looking" (or "pseudorandom") output string.
- PRG uses a small amount of true randomness in order to generate a large amount of pseudorandomness.
- Useful whenever a large number of random(-looking) bits are needed, since generating true random bits is difficult and slow.

Pseudo-Random Generator (PRG)

Definition (Pseudo-Random Generator)

Let $\ell(\cdot)$ be a polynomial function with security parameter $n \in \mathbb{N}$, and $\mathcal{S} = \{0,1\}^n$ and $\mathcal{R} = \{0,1\}^{\ell(n)}$. Let $G : \mathcal{S} \to \mathcal{R}$ be a deterministic, polynomial-time algorithm in n such that for any n and any input (*seed*) $s \in \mathcal{S}$, the result of $G(s)$ is a string of length $\ell(n)$ with $\ell(n) \gg n$ for all n. We say that G is a Pseudo-Random Generator (PRG), and is defined over $(\mathcal{S}, \mathcal{R})$. The function $\ell(\cdot)$ is called the expansion factor of G.

Pseudo-Random Generator Advantage

- Similar to Computational Indistinguishability, it is also defined as an attack game played between two parties,
  - a challenger, and
  - an adversary, also called Distinguisher.

Pseudo-Random Generator Advantage

- Similar to Computational Indistinguishability, it is also defined as an attack game played between two parties,
  - a challenger, and
  - an adversary, also called Distinguisher.

PRG Indistinguishability Game

For a given PRG G defined over $(\mathcal{S}, \mathcal{R})$ with security parameter $n \in \mathbb{N}$, and for a given PPT adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes r as follows:
  - if $b = 0 : s \xleftarrow{R} \mathcal{S}, r \longleftarrow G(s)$.

Pseudo-Random Generator Advantage

- Similar to Computational Indistinguishability, it is also defined as an attack game played between two parties,
  - a challenger, and
  - an adversary, also called Distinguisher.

PRG Indistinguishability Game

For a given PRG G defined over $(\mathcal{S}, \mathcal{R})$ with security parameter $n \in \mathbb{N}$, and for a given PPT adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes r as follows:
  - if $b = 0 : s \xleftarrow{R} \mathcal{S}, r \longleftarrow G(s)$.
  - if $b = 1 : r \xleftarrow{R} \mathcal{R}$.

Pseudo-Random Generator Advantage

- Similar to Computational Indistinguishability, it is also defined as an attack game played between two parties,
  - a challenger, and
  - an adversary, also called Distinguisher.

PRG Indistinguishability Game

For a given PRG G defined over $(\mathcal{S}, \mathcal{R})$ with security parameter $n \in \mathbb{N}$, and for a given PPT adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes r as follows:
  - if $b = 0 : s \xleftarrow{R} \mathcal{S}, r \longleftarrow G(s)$.
  - if $b = 1 : r \xleftarrow{R} \mathcal{R}$.
- Challenger sends r to $\mathcal{A}$.
- The adversary outputs a bit $\hat{b} \in \{0, 1\}$.

Pseudo-Random Generator Advantage

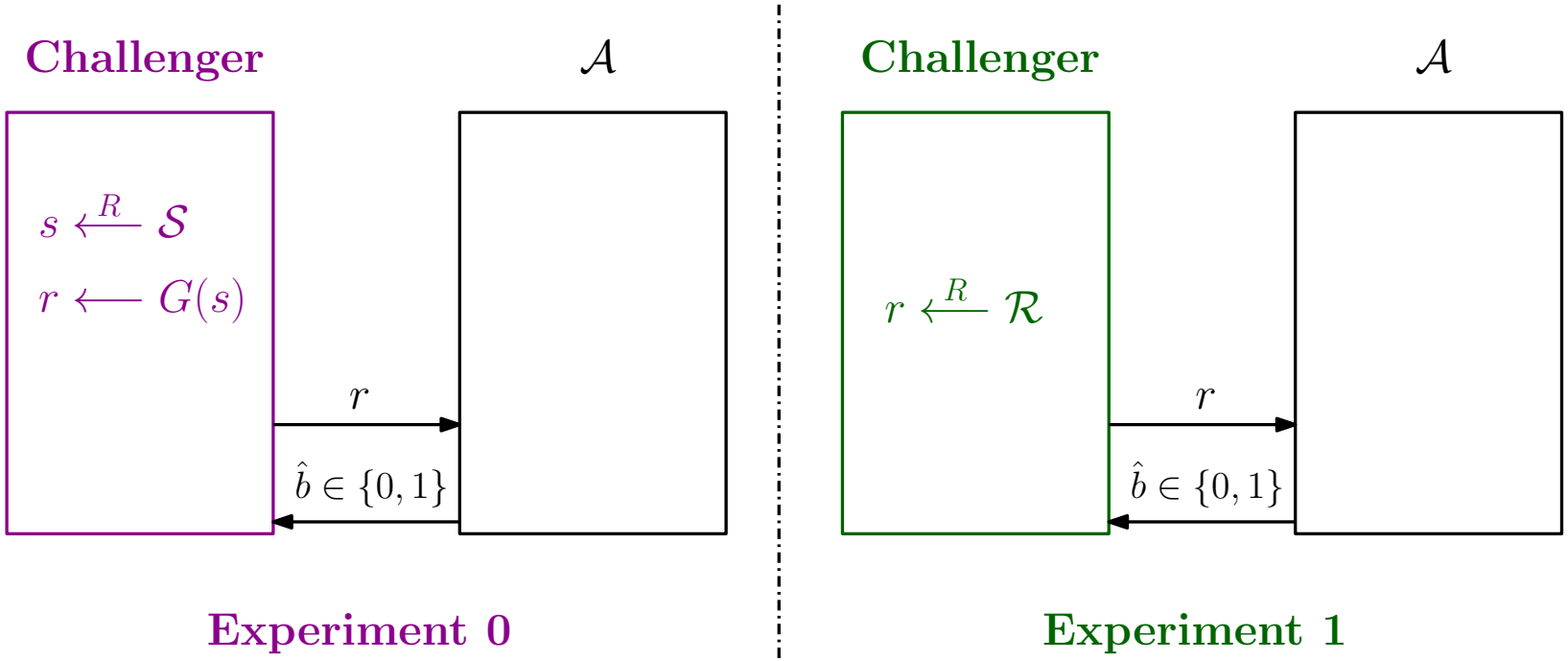
- Similar to Computational Indistinguishability, it is also defined as an attack game played between two parties,
  - a challenger, and
  - an adversary, also called Distinguisher.

PRG Indistinguishability Game

For a given PRG G defined over $(\mathcal{S}, \mathcal{R})$ with security parameter $n \in \mathbb{N}$, and for a given PPT adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes r as follows:
  - if $b = 0 : s \xleftarrow{R} \mathcal{S}, r \longleftarrow G(s)$.
  - if $b = 1 : r \xleftarrow{R} \mathcal{R}$.
- Challenger sends r to $\mathcal{A}$.
- The adversary outputs a bit $\hat{b} \in \{0, 1\}$.
- Assume $\hat{b} = 0 \Rightarrow$ Not random and $\hat{b} = 1 \Rightarrow$ Random.

Pseudo-Random Generator Advantage



Pseudo-Random Generator Advantage

PRG Advantage

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define the advantage of $\mathcal{A}$ in the attack game with respect to G as

$$\mathsf{PRGadv}[\mathcal{A}, G] = |\Pr[W_0] - \Pr[W_1]|.$$

Pseudo-Random Generator Advantage

PRG Advantage

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define the advantage of $\mathcal{A}$ in the attack game with respect to G as

$$\mathsf{PRGadv}[\mathcal{A}, G] = |\Pr[W_0] - \Pr[W_1]|.$$

Source of Randomness

Events W_0 and W_1 are defined with respect to the probability space determined by:

- the random choice of s and r.

Pseudo-Random Generator Advantage

PRG Advantage

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define the advantage of $\mathcal{A}$ in the attack game with respect to G as

$$\mathsf{PRGadv}[\mathcal{A}, G] = |\Pr[W_0] - \Pr[W_1]|.$$

Source of Randomness

Events W_0 and W_1 are defined with respect to the probability space determined by:

- the random choice of s and r.

Secure PRG

A PRG G is secure if for all PPT adversaries (or distinguisher) $\mathcal{A}$ there exists a negligible function ϵ such that

$$\mathsf{PRGadv}[\mathcal{A}, G] \leqslant \epsilon(n).$$

PRG Advantage: Bit Guessing version

PRG Indistinguishability Game

For a given PRG G defined over $(\mathcal{S}, \mathcal{R})$ with security parameter $n \in \mathbb{N}$, and for a given PPT adversary $\mathcal{A}$, we define the Experiment as:

PRG Advantage: Bit Guessing version

PRG Indistinguishability Game

For a given PRG G defined over $(\mathcal{S}, \mathcal{R})$ with security parameter $n \in \mathbb{N}$, and for a given PPT adversary $\mathcal{A}$, we define the Experiment as:

- Challenger first computes $b \xleftarrow{R} \{0,1\}$.

PRG Advantage: Bit Guessing version

PRG Indistinguishability Game

For a given PRG G defined over $(\mathcal{S}, \mathcal{R})$ with security parameter $n \in \mathbb{N}$, and for a given PPT adversary $\mathcal{A}$, we define the Experiment as:

- Challenger first computes $b \xleftarrow{R} \{0,1\}$.
- The challenger computes r as follows:
  - if $b = 0 : s \xleftarrow{R} \mathcal{S}, r \longleftarrow G(s)$.

PRG Advantage: Bit Guessing version

PRG Indistinguishability Game

For a given PRG G defined over $(\mathcal{S}, \mathcal{R})$ with security parameter $n \in \mathbb{N}$, and for a given PPT adversary $\mathcal{A}$, we define the Experiment as:

- Challenger first computes $b \xleftarrow{R} \{0,1\}$.
- The challenger computes r as follows:
  - if $b = 0 : s \xleftarrow{R} \mathcal{S}, r \longleftarrow G(s)$.
  - if $b = 1 : r \xleftarrow{R} \mathcal{R}$.

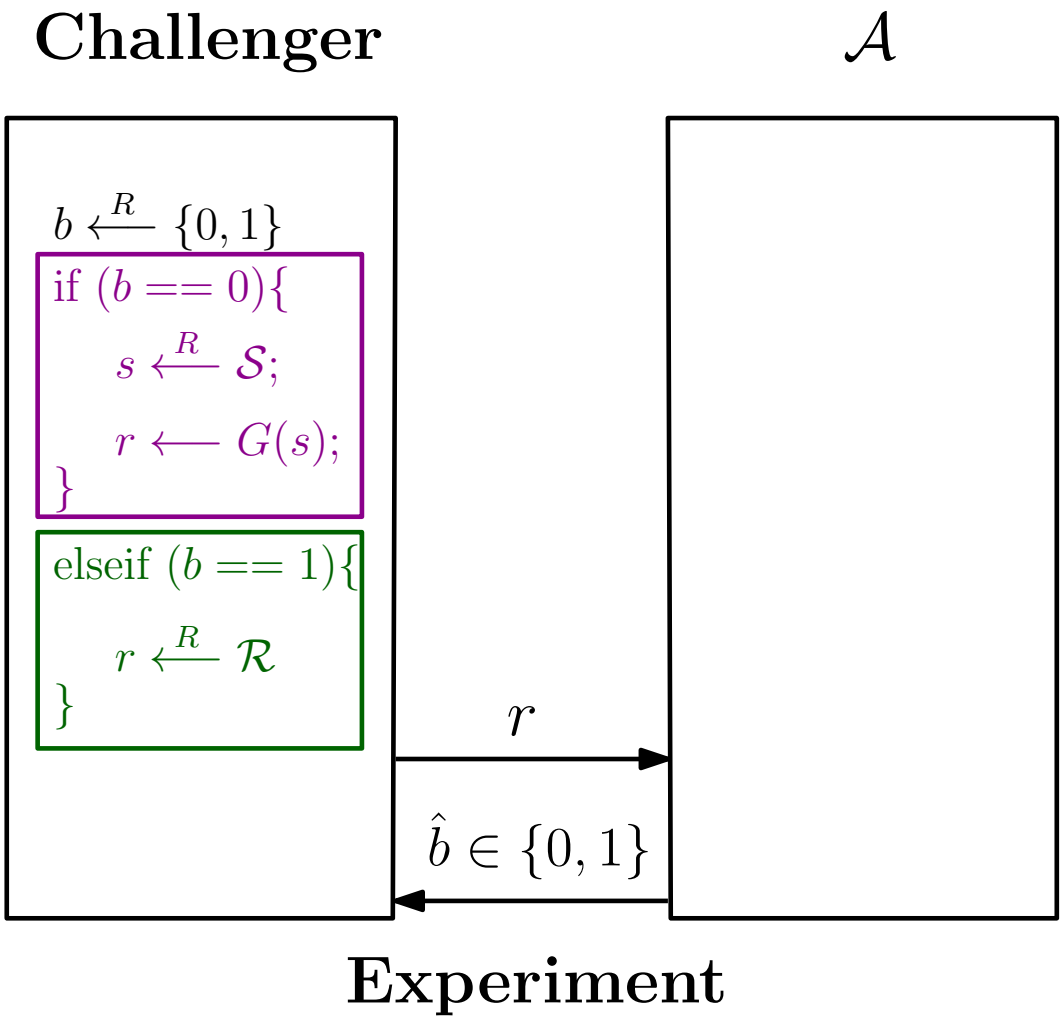
PRG Advantage: Bit Guessing version

PRG Indistinguishability Game

For a given PRG G defined over $(\mathcal{S}, \mathcal{R})$ with security parameter $n \in \mathbb{N}$, and for a given PPT adversary $\mathcal{A}$, we define the Experiment as:

- Challenger first computes $b \xleftarrow{R} \{0,1\}$.
- The challenger computes r as follows:
  - if $b = 0 : s \xleftarrow{R} \mathcal{S}, r \longleftarrow G(s)$.
  - if $b = 1 : r \xleftarrow{R} \mathcal{R}$.
- Challenger sends r to $\mathcal{A}$.
- The adversary outputs a bit $\hat{b} \in \{0,1\}$.

Pseudo-Random Generator Advantage



PRG Advantage: Bit Guessing version

PRG Advantage

Let W be the event that where $\mathcal{A}$ wins if $\mathcal{A}$ outputs $\hat{b} = b$. We define the advantage of $\mathcal{A}$ in the attack game with respect to G as

$$\mathsf{PRGadv}^{*}[\mathcal{A}, G] = \left| \Pr[\hat{b} = b] - \frac{1}{2} \right|.$$

PRG Advantage: Bit Guessing version

PRG Advantage

Let W be the event that where $\mathcal{A}$ wins if $\mathcal{A}$ outputs $\hat{b} = b$. We define the advantage of $\mathcal{A}$ in the attack game with respect to G as

$$\mathsf{PRGadv}^*[\mathcal{A}, G] = \left| \Pr[\hat{b} = b] - \frac{1}{2} \right|.$$

Theorem

For every PRG G and every PPT adversary $\mathcal{A}$, we have

$$\mathsf{PRGadv}[\mathcal{A}, G] = 2 \cdot \mathsf{PRGadv}^*[\mathcal{A}, G].$$

Statistical Test

Statistical Test

An algorithm that is used to distinguish a pseudo-random string $r = G(s)$ from a truly random string r is called a statistical test.

Statistical Test

Statistical Test

An algorithm that is used to distinguish a pseudo-random string $r = G(s)$ from a truly random string r is called a statistical test.

Example of Statistical Test

Let A be a statistical test on a string $\{0,1\}^L$ where $L = \ell(n)$. If x is truely random, then

- k = Expected number of 1's $\approx L/2$.

Statistical Test

Statistical Test

An algorithm that is used to distinguish a pseudo-random string $r = G(s)$ from a truly random string r is called a statistical test.

Example of Statistical Test

Let A be a statistical test on a string $\{0,1\}^L$ where $L = \ell(n)$. If x is truely random, then

- k = Expected number of 1's $\approx L/2$.
  - $A(x) = 1$ if and only if $| k - 0.5L | \leqslant 0.01L$.

Statistical Test

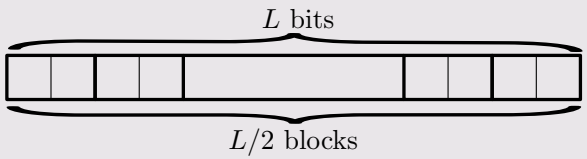
Statistical Test

An algorithm that is used to distinguish a pseudo-random string $r = G(s)$ from a truly random string r is called a statistical test.

Example of Statistical Test

Let A be a statistical test on a string $\{0,1\}^L$ where $L = \ell(n)$. If x is truely random, then

- k = Expected number of 1's $\approx L/2$.
  - $A(x) = 1$ if and only if $\mid k - 0.5L \mid \leqslant 0.01L$.
- Break the string in blocks of length two.



  - k_{ij} = Expected number of blocks containing $ij \approx L/2 \cdot 1/4 = L/8$ where $i, j \in \{0,1\}$.

Statistical Test

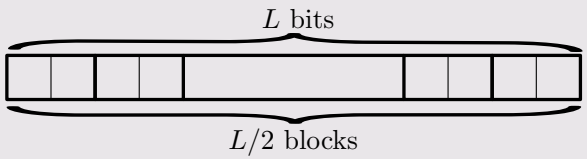
Statistical Test

An algorithm that is used to distinguish a pseudo-random string $r = G(s)$ from a truly random string r is called a statistical test.

Example of Statistical Test

Let A be a statistical test on a string $\{0,1\}^L$ where $L = \ell(n)$. If x is truely random, then

- k = Expected number of 1's $\approx L/2$.
  - $A(x) = 1$ if and only if $\mid k - 0.5L \mid \leqslant 0.01L$.
- Break the string in blocks of length two.

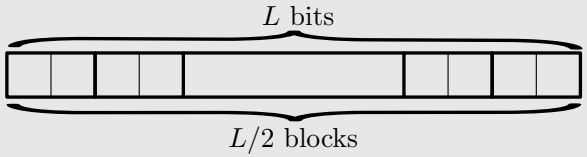


  - k_{ij} = Expected number of blocks containing $ij \approx L/2 \cdot 1/4 = L/8$ where $i, j \in \{0,1\}$.
  - $A(x) = 1$ if and only if $\mid k_{ij} - L/8 \mid \leqslant 0.01L$ for all i, j.

Statistical Test

Example of Statistical Test

- One can also use χ-squared statistical test.
- Break the string in blocks of length two.

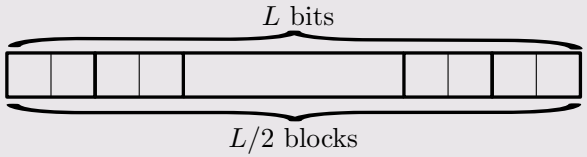


  - $A(x) = 1$ if and only if $\sum_{i,j} \left(k_{ij} - L/8\right)^2 \leqslant 0.005L$.

Statistical Test

Example of Statistical Test

- One can also use χ-squared statistical test.
- Break the string in blocks of length two.



  - $A(x) = 1$ if and only if $\sum_{i,j} \left(k_{ij} - L/8\right)^2 \leqslant 0.005L$.

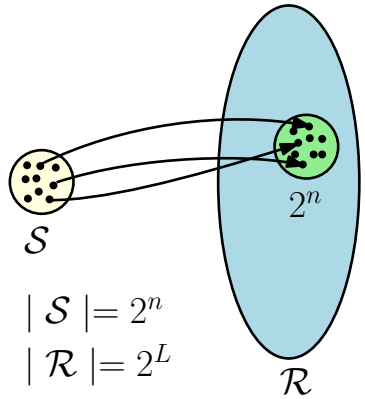
- Conceptually, if PRG G is secure then there must be no efficient statistical test.

Discussion

- PRG is far from random.
- Let PRG G be defined over $\mathcal{S} = \{0,1\}^n$ and $\mathcal{R} = \{0,1\}^L$ where $L = \ell(n)$ and $L >> n$.
- $| \mathcal{S} | = 2^n$ and $| \mathcal{R} | = 2^L$.

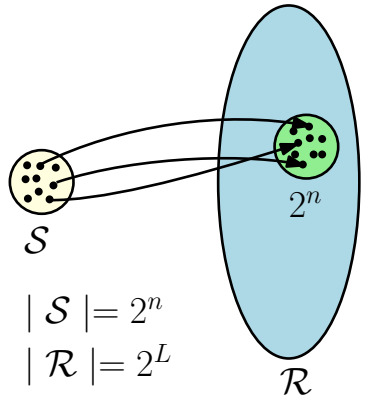
Discussion

- PRG is far from random.
- Let PRG G be defined over $\mathcal{S} = \{0,1\}^n$ and $\mathcal{R} = \{0,1\}^L$ where $L = \ell(n)$ and $L >> n$.
- $|\mathcal{S}| = 2^n$ and $|\mathcal{R}| = 2^L$.
- Therefore, PRG G generates only 2^n elements of $\mathcal{R}$.



Discussion

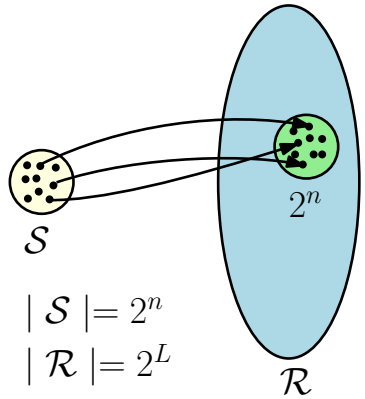
- PRG is far from random.
- Let PRG G be defined over $\mathcal{S} = \{0,1\}^n$ and $\mathcal{R} = \{0,1\}^L$ where $L = \ell(n)$ and $L >> n$.
- $|\mathcal{S}| = 2^n$ and $|\mathcal{R}| = 2^L$.
- Therefore, PRG G generates only 2^n elements of $\mathcal{R}$.



- If we choose an element x uniformly at random from $\mathcal{R}$ with probability 2^{-L}.

Discussion

- PRG is far from random.
- Let PRG G be defined over $\mathcal{S} = \{0,1\}^n$ and $\mathcal{R} = \{0,1\}^L$ where $L = \ell(n)$ and $L >> n$.
- $|\mathcal{S}| = 2^n$ and $|\mathcal{R}| = 2^L$.
- Therefore, PRG G generates only 2^n elements of $\mathcal{R}$.



- If we choose an element x uniformly at random from $\mathcal{R}$ with probability 2^{-L}.
- $\Pr[x \text{ is in range of } G] = 2^{n-L}$.

Discussion

- Let $\mathcal{A}$ be a distinguisher with infinite time and unbounded computational power.

Discussion

- Let $\mathcal{A}$ be a distinguisher with infinite time and unbounded computational power.
- $\mathcal{A}$ knows the details of G and $(\mathcal{S},\mathcal{R})$ as they are public.

Discussion

- Let $\mathcal{A}$ be a distinguisher with infinite time and unbounded computational power.
- $\mathcal{A}$ knows the details of G and $(\mathcal{S},\mathcal{R})$ as they are public.
  - $\mathcal{A}$ takes an input string w of length L.

Discussion

- Let $\mathcal{A}$ be a distinguisher with infinite time and unbounded computational power.
- $\mathcal{A}$ knows the details of G and $(\mathcal{S},\mathcal{R})$ as they are public.
  - $\mathcal{A}$ takes an input string w of length L.
  - The seed s is unknown to $\mathcal{A}$.

Discussion

- Let $\mathcal{A}$ be a distinguisher with infinite time and unbounded computational power.
- $\mathcal{A}$ knows the details of G and $(\mathcal{S}, \mathcal{R})$ as they are public.
  - $\mathcal{A}$ takes an input string w of length L.
  - The seed s is unknown to $\mathcal{A}$.
  - $\mathcal{A}$ computes $G(s)$ for all $s \in \mathcal{S}$ and checks $w \stackrel{?}{=} G(s)$.

Discussion

- Let $\mathcal{A}$ be a distinguisher with infinite time and unbounded computational power.
- $\mathcal{A}$ knows the details of G and $(\mathcal{S}, \mathcal{R})$ as they are public.
  - $\mathcal{A}$ takes an input string w of length L.
  - The seed s is unknown to $\mathcal{A}$.
  - $\mathcal{A}$ computes $G(s)$ for all $s \in \mathcal{S}$ and checks $w \stackrel{?}{=} G(s)$.
  - If it finds a s, then stops and outputs 0.
  - Otherwise outputs 1.

Discussion

- Let $\mathcal{A}$ be a distinguisher with infinite time and unbounded computational power.
- $\mathcal{A}$ knows the details of G and $(\mathcal{S},\mathcal{R})$ as they are public.
  - $\mathcal{A}$ takes an input string w of length L.
  - The seed s is unknown to $\mathcal{A}$.
  - $\mathcal{A}$ computes $G(s)$ for all $s \in \mathcal{S}$ and checks $w \stackrel{?}{=} G(s)$.
  - If it finds a s, then stops and outputs 0.
  - Otherwise outputs 1.
- Then we have

$$\mathsf{PRGadv}[\mathcal{A},G] = \mid \Pr[W_0] - \Pr[W_1] \mid = \mid 0 - (1 - 2^{n-L}) \mid = 1 - 2^{n-L}.$$

Discussion

- Let $\mathcal{A}$ be a distinguisher with infinite time and unbounded computational power.
- $\mathcal{A}$ knows the details of G and $(\mathcal{S},\mathcal{R})$ as they are public.
  - $\mathcal{A}$ takes an input string w of length L.
  - The seed s is unknown to $\mathcal{A}$.
  - $\mathcal{A}$ computes $G(s)$ for all $s \in \mathcal{S}$ and checks $w \stackrel{?}{=} G(s)$.
  - If it finds a s, then stops and outputs 0.
  - Otherwise outputs 1.
- Then we have

$$\mathsf{PRGadv}[\mathcal{A},G] = \mid \Pr[W_0] - \Pr[W_1] \mid = \mid 0 - (1 - 2^{n-L}) \mid = 1 - 2^{n-L}.$$

- This attack is commonly known as Brute Force attack.
- If $\mathcal{A}$ is a poly-time distinguisher, it does not have sufficient time.

Discussion

Seed and Length

- Seed s must be kept secret.

Discussion

Seed and Length

- Seed s must be kept secret.
- For seed space $\mathcal{S} = \{0,1\}^n$, n must be long enough such that brute-forced attack is not possible.

Stream Cipher: encryption with a PRG

Stream Cipher

- Let G be a PRG defined over $(\mathcal{S}, \mathcal{R}) = (\{0,1\}^n, \{0,1\}^L)$.

Stream Cipher: encryption with a PRG

Stream Cipher

- Let G be a PRG defined over $(\mathcal{S}, \mathcal{R}) = (\{0,1\}^n, \{0,1\}^L)$.
- The stream cipher $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ constructed from G is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where
  - $\mathcal{K} = \{0,1\}^n, \mathcal{M} = \{0,1\}^{\leqslant L}$ and $\mathcal{C} = \{0,1\}^{\leqslant L}$.

Stream Cipher: encryption with a PRG

Stream Cipher

- Let G be a PRG defined over $(\mathcal{S}, \mathcal{R}) = (\{0,1\}^n, \{0,1\}^L)$.
- The stream cipher $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ constructed from G is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where
  - $\mathcal{K} = \{0,1\}^n, \mathcal{M} = \{0,1\}^{\leqslant L}$ and $\mathcal{C} = \{0,1\}^{\leqslant L}$.
- $\mathcal{G} : s \xleftarrow{R} \mathcal{S}$.

Stream Cipher: encryption with a PRG

Stream Cipher

- Let G be a PRG defined over $(\mathcal{S}, \mathcal{R}) = (\{0,1\}^n, \{0,1\}^L)$.
- The stream cipher $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ constructed from G is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where
  - $\mathcal{K} = \{0,1\}^n, \mathcal{M} = \{0,1\}^{\leqslant L}$ and $\mathcal{C} = \{0,1\}^{\leqslant L}$.
- $\mathcal{G} : s \xleftarrow{R} \mathcal{S}$.
- $\mathcal{E}(s, m)$: If $| m | = v \leqslant L$, then

$$c := G(s)[0 \ldots (v-1)] \oplus m.$$

Stream Cipher: encryption with a PRG

Stream Cipher

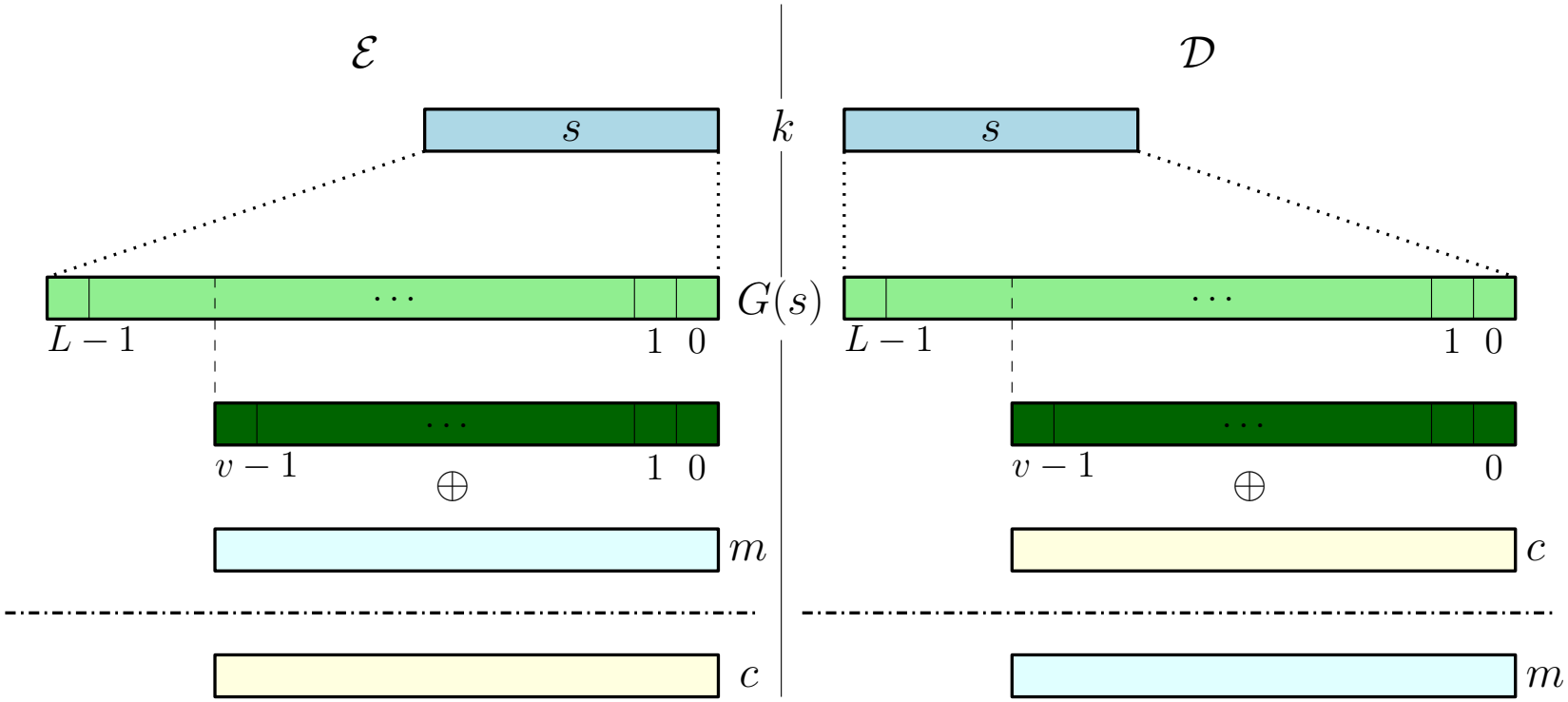
- Let G be a PRG defined over $(\mathcal{S}, \mathcal{R}) = (\{0,1\}^n, \{0,1\}^L)$.
- The stream cipher $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ constructed from G is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where
  - $\mathcal{K} = \{0,1\}^n, \mathcal{M} = \{0,1\}^{\leqslant L}$ and $\mathcal{C} = \{0,1\}^{\leqslant L}$.
- $\mathcal{G} : s \xleftarrow{R} \mathcal{S}$.
- $\mathcal{E}(s, m)$: If $| m | = v \leqslant L$, then

$$c := G(s)[0 \ldots (v-1)] \oplus m.$$

- $\mathcal{D}(s, c)$: If $| c | = v \leqslant L$, then

$$m := G(s)[0 \ldots (v-1)] \oplus c.$$

Stream Cipher



Stream Cipher

Theorem

If G is a secure PRG, then the stream cipher $\mathfrak{E}$ constructed from G is a semantically secure cipher.

Stream Cipher

Theorem

If G is a secure PRG, then the stream cipher $\mathfrak{E}$ constructed from G is a semantically secure cipher.
In particular, for every computational indistinguishing adversary $\mathcal{A}$ that attacks $\mathfrak{E}$ as in Symmetric key Encryption Indistinguishability Attack Game, there exists a PRG adversary $\mathcal{B}$ that attacks G as in PRG Indistinguishability Attack Game, such that

$$\mathsf{INDadv}[\mathcal{A}, \mathfrak{E}] = 2 \cdot \mathsf{PRGadv}[\mathcal{B}, G].$$

Proof

- Let G be a secure PRG defined over $(\mathcal{S}, \mathcal{R}) = (\{0,1\}^n, \{0,1\}^L)$.

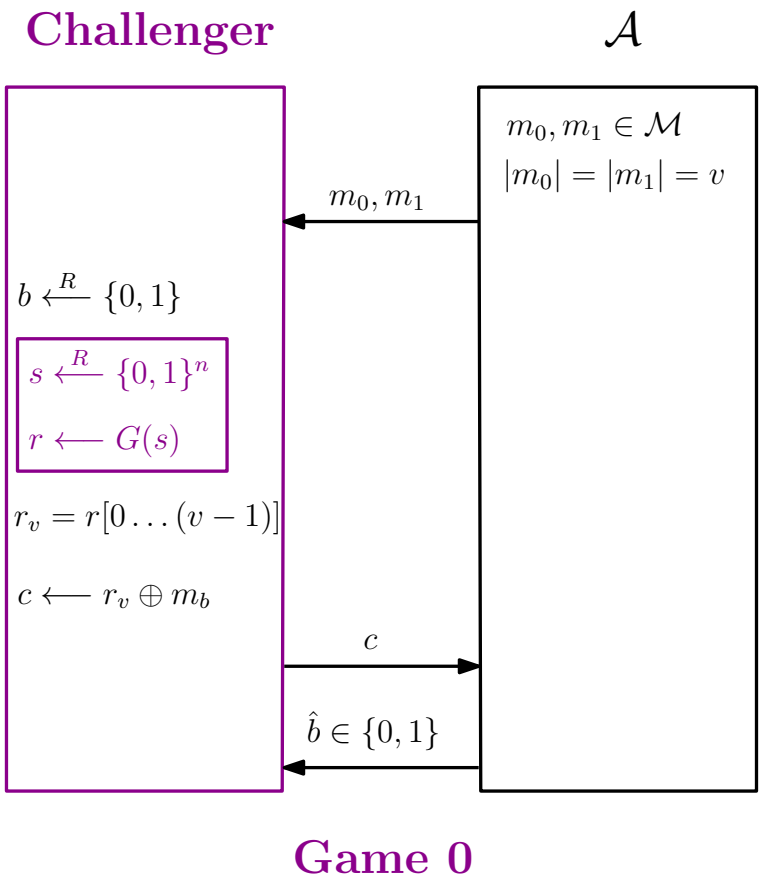
Proof

- Let G be a secure PRG defined over $(\mathcal{S}, \mathcal{R}) = (\{0,1\}^n, \{0,1\}^L)$.
- Therefore, any PPT adversary $\mathfrak{D}$, there exists a negligible function ϵ such that

$$\mathsf{PRGadv}[\mathfrak{D}, G] \leqslant \epsilon(n).$$

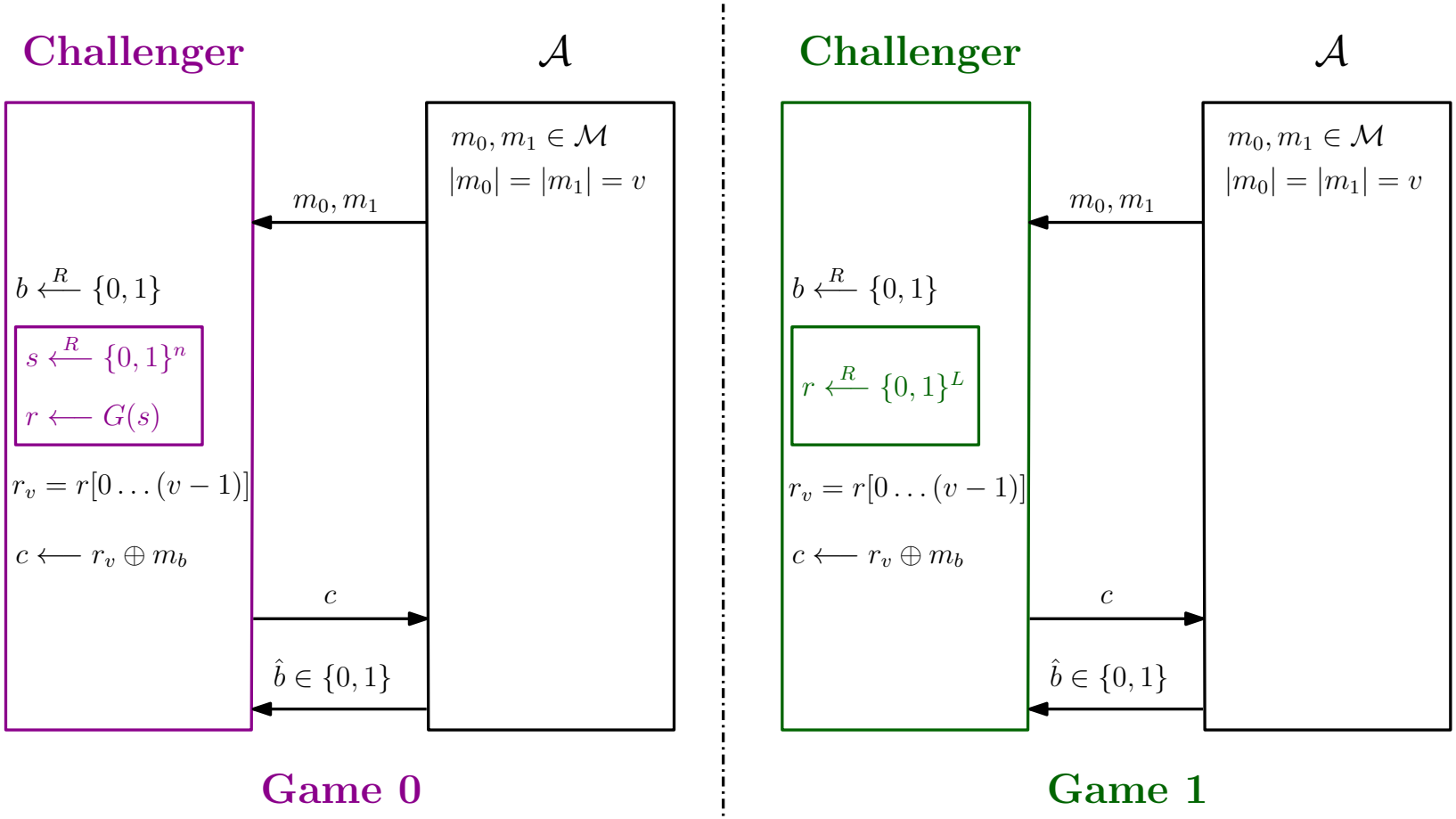
Stream Cipher

Proof



Stream Cipher

Proof



Proof

- Game 0:
  - Let W_0 be the event that $\hat{b} = b$ in Game 0.
  - By definition, we have

$$\mathsf{INDadv}^*[\mathcal{A}, \mathfrak{E}] = \left| \Pr[W_0] - \frac{1}{2} \right|.$$

Proof

- Game 0:
  - Let W_0 be the event that $\hat{b} = b$ in Game 0.
  - By definition, we have

$$\mathsf{INDadv}^*[\mathcal{A}, \mathfrak{E}] = \left| \Pr[W_0] - \frac{1}{2} \right|.$$

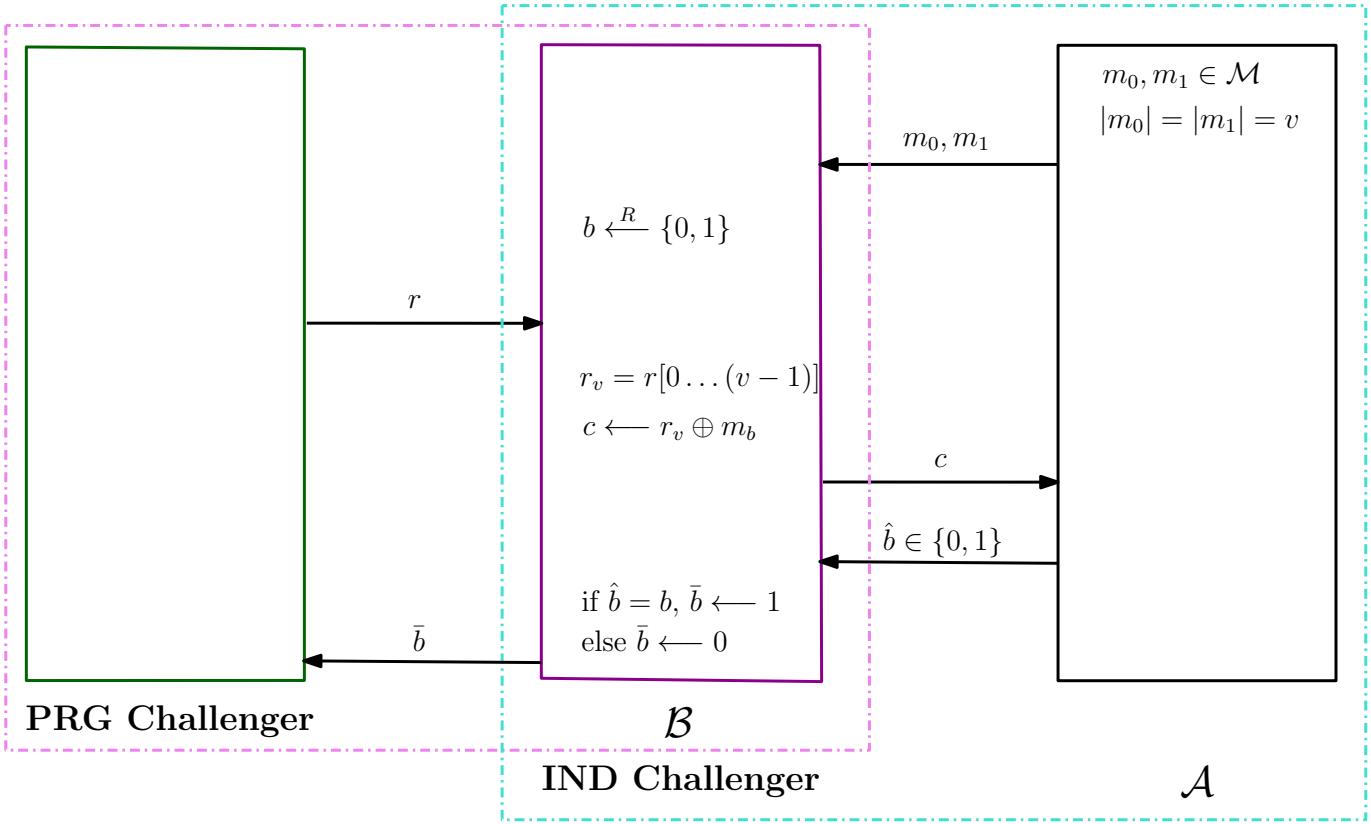
- Game 1:
  - Let W_1 be the event that $\hat{b} = b$ in Game 1.
  - $\mathcal{A}$ is attacking the variable length one-time pad.
  - the challenger's hidden bit b and $\mathcal{A}$'s output $\hat{b}$ are independent.
  - Therefore, we have

$$\Pr[W_1] = \frac{1}{2}.$$

Stream Cipher

Proof

- If $\beta = 0$, we are in Experiment 0 of PRG indistinguishability attack game.
- If $\beta = 1$, we are in Experiment 1 of PRG indistinguishability attack game.



Experiment β

Proof

- Let p_0 be the probability $\mathcal{B}$ outputs 1 in Experiment 0.
- Let p_1 be the probability $\mathcal{B}$ outputs 1 in Experiment 1.

Proof

- Let p_0 be the probability $\mathcal{B}$ outputs 1 in Experiment 0.
- Let p_1 be the probability $\mathcal{B}$ outputs 1 in Experiment 1.
- By definition,

$$\mathsf{PRGadv} = | p_0 - p_1 | .$$

Proof

- Let p_0 be the probability $\mathcal{B}$ outputs 1 in Experiment 0.
- Let p_1 be the probability $\mathcal{B}$ outputs 1 in Experiment 1.
- By definition,

$$\mathsf{PRGadv} = \mid p_0 - p_1 \mid .$$

- Experiment 0 implies $\mathcal{A}$ is playing Game 0.

$$p_0 = \Pr[W_0].$$

Proof

- Let p_0 be the probability $\mathcal{B}$ outputs 1 in Experiment 0.
- Let p_1 be the probability $\mathcal{B}$ outputs 1 in Experiment 1.
- By definition,

$$\mathsf{PRGadv} = \mid p_0 - p_1 \mid .$$

- Experiment 0 implies $\mathcal{A}$ is playing Game 0.

$$p_0 = \Pr[W_0].$$

- Experiment 1 implies $\mathcal{A}$ is playing Game 1.

$$p_1 = \Pr[W_1].$$

Proof

$$
\begin{aligned}
\mathsf{PRGadv}[\mathcal{B}, G] &= |\, p_0 - p_1 \,| \\
&= |\, \Pr[W_0] - \Pr[W_1] \,| \\
&= \left| \Pr[W_0] - \frac{1}{2} \right| \\
&= \mathsf{INDadv}^{*}[\mathcal{A}, \mathfrak{E}]
\end{aligned}
$$

Proof

$$\begin{aligned}
\mathsf{PRGadv}[\mathcal{B}, G] &= |\, p_0 - p_1 \,| \\
&= |\, \Pr[W_0] - \Pr[W_1] \,| \\
&= \left| \Pr[W_0] - \frac{1}{2} \right| \\
&= \mathsf{INDadv}^*[\mathcal{A}, \mathfrak{E}]
\end{aligned}$$

As $\mathsf{INDadv}[\mathcal{A}, \mathfrak{E}] = 2 \cdot \mathsf{INDadv}^*[\mathcal{A}, \mathfrak{E}]$, we have

$$\mathsf{INDadv}[\mathcal{A}, \mathfrak{E}] = 2 \cdot \mathsf{PRGadv}[\mathcal{B}, G] \leqslant 2\epsilon(n).$$

The Two-time pad is insecure

- A stream cipher is secure to encrypt a single message from Alice to Bob.

The Two-time pad is insecure

- A stream cipher is secure to encrypt a single message from Alice to Bob.
- Suppose Alice wants to send two messages m_0 and m_1.

The Two-time pad is insecure

- A stream cipher is secure to encrypt a single message from Alice to Bob.
- Suppose Alice wants to send two messages m_0 and m_1.
- Naive solution:

$$c_0 \longleftarrow m_0 \oplus G(s) \text{ and } c_1 \longleftarrow m_1 \oplus G(s).$$

The Two-time pad is insecure

- A stream cipher is secure to encrypt a single message from Alice to Bob.
- Suppose Alice wants to send two messages m_0 and m_1.
- Naive solution:

$$c_0 \longleftarrow m_0 \oplus G(s) \text{ and } c_1 \longleftarrow m_1 \oplus G(s).$$

  - This construction is insecure.

$$\Delta = c_0 \oplus c_1 = (m_0 \oplus G(s) \oplus (m_1 \oplus G(s)) = m_0 \oplus m_1.$$

The Two-time pad is insecure

- A stream cipher is secure to encrypt a single message from Alice to Bob.
- Suppose Alice wants to send two messages m_0 and m_1.
- Naive solution:

$$c_0 \longleftarrow m_0 \oplus G(s) \text{ and } c_1 \longleftarrow m_1 \oplus G(s).$$

  - This construction is insecure.

$$\Delta = c_0 \oplus c_1 = (m_0 \oplus G(s) \oplus (m_1 \oplus G(s)) = m_0 \oplus m_1.$$

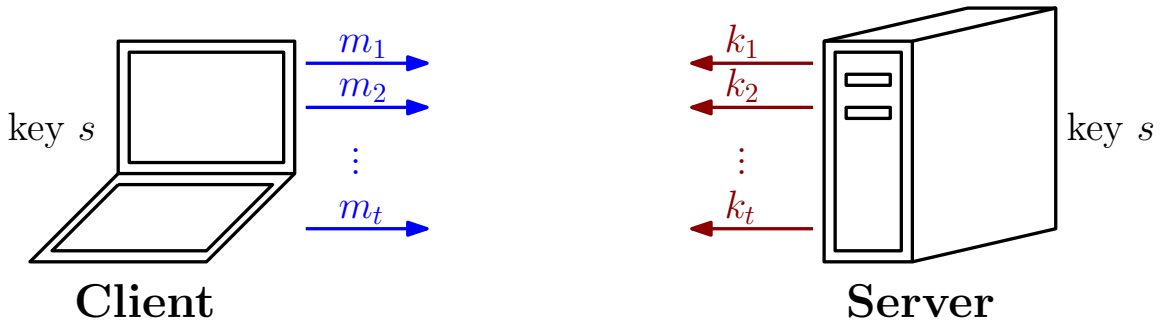
  - English text contains enough redundancy.
  - Given $\Delta = m_0 \oplus m_1$, one can recover both the messages m_0 and m_1.
  - This construction known as Two-time pad.
- A stream cipher key should never be used to encrypt more than one message.

Example of Two-time pad

- Project Venona (1941-1946):
  - During WWII, Russia was using one time pad.
  - Human used to throw dice and the results were used as pad.
  - It was hard for human being to generate all the pads.
  - They started to encrypt two messages using one pad.
  - US intelligence was able, over the next 25 years, to break some 2900 messages just from intercepted ciphers.

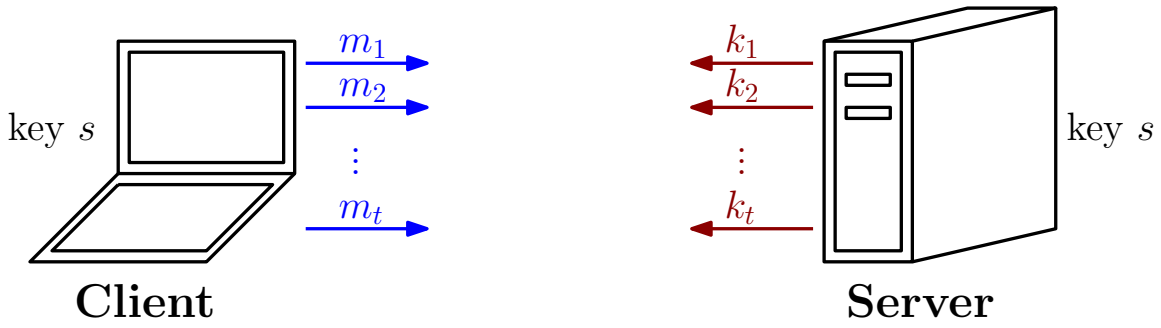
Example of Two-time pad

- MS-PPTP (Windows NT):
  - PPTP stands for Point to Point Transfer Protocol.



Example of Two-time pad

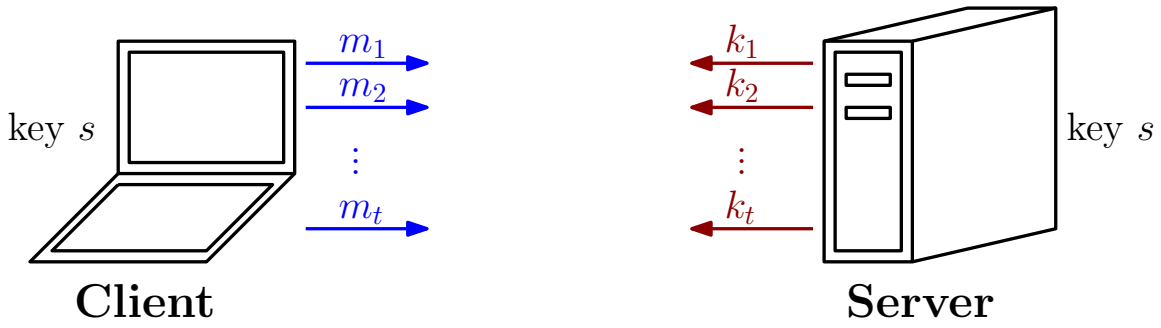
- MS-PPTP (Windows NT):
  - PPTP stands for Point to Point Transfer Protocol.



  - $m := (m_1 \| m_2 \| \cdots \| m_t)$ and $c_1 := m \oplus G(s)$.
  - $k := (k_1 \| k_2 \| \cdots \| k_t)$ and $c_2 := k \oplus G(s)$.

Example of Two-time pad

- MS-PPTP (Windows NT):
  - PPTP stands for Point to Point Transfer Protocol.



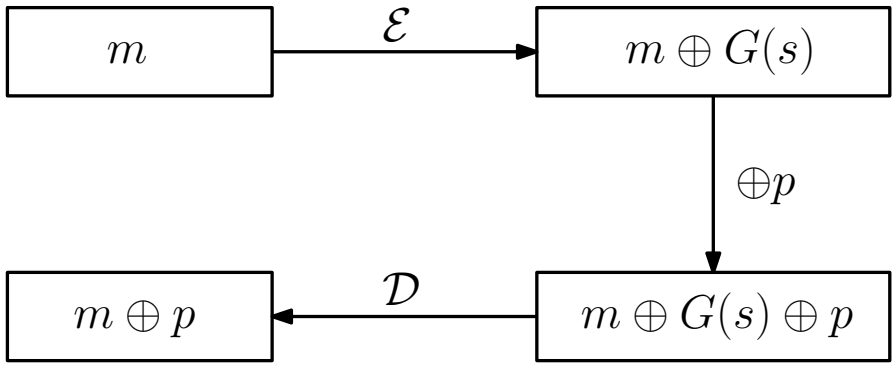
  - $m := (m_1 \| m_2 \| \cdots \| m_t)$ and $c_1 := m \oplus G(s)$.
  - $k := (k_1 \| k_2 \| \cdots \| k_t)$ and $c_2 := k \oplus G(s)$.
  - Solution: $s = (s_{C \to S}, s_{S \to C})$, and both know the key.

The One-Time pad is Malleable

- Till now, we considered that attacker only eavesdrops.
- Let attacker be active one.

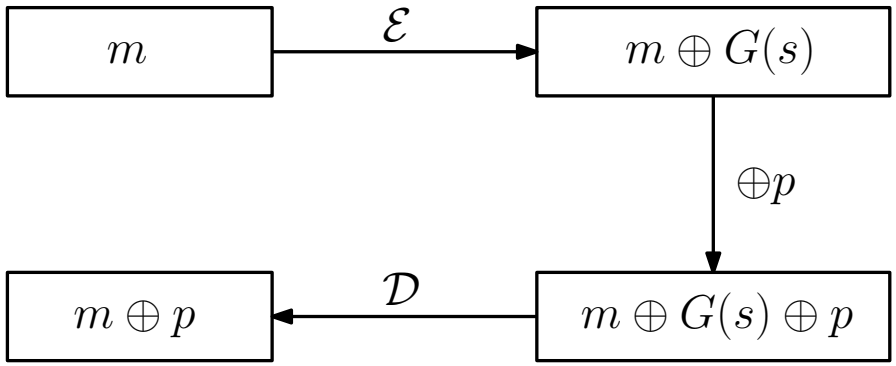
The One-Time pad is Malleable

- Till now, we considered that attacker only eavesdrops.
- Let attacker be active one.



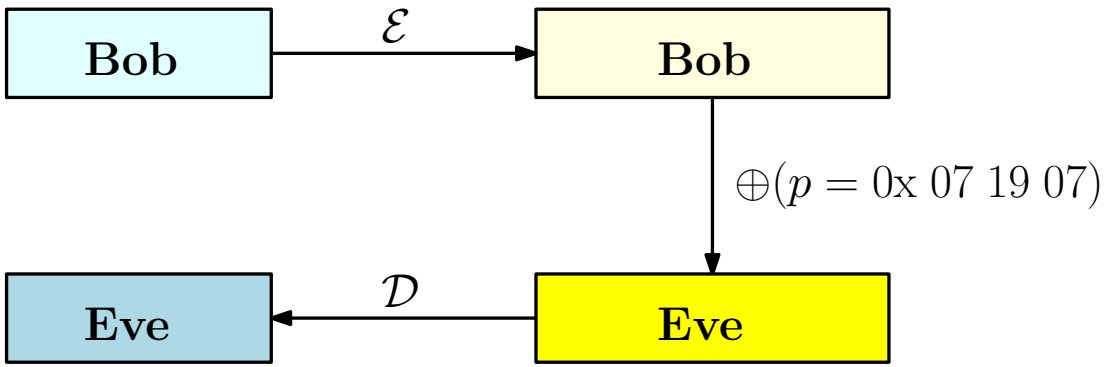
The One-Time pad is Malleable

- Till now, we considered that attacker only eavesdrops.
- Let attacker be active one.

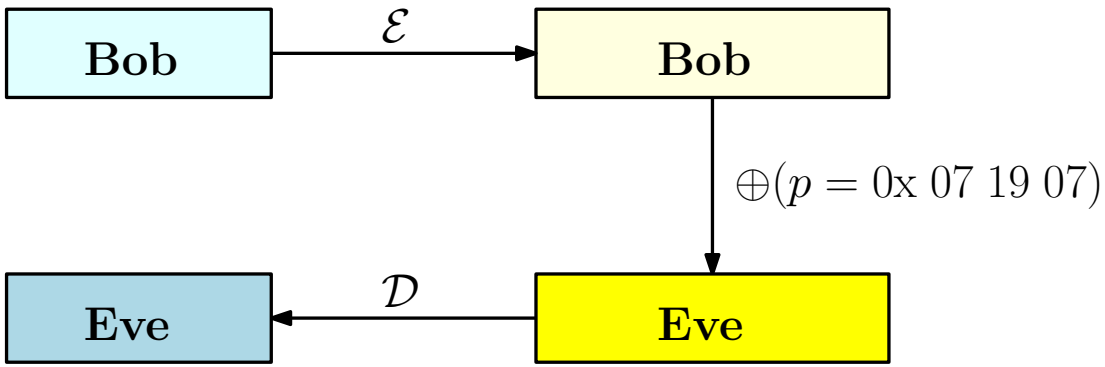


- The modification will never be detected by the decryptor.
- Malleable since an attacker can cause predictable changes to the plaintext.

The One-Time pad is Malleable



The One-Time pad is Malleable



- Bob = 0x 42 6F 62.
- Eve = 0x 45 76 65.
- p =Bob$\oplus$Eve = 0x 07 19 07

Composing PRGs

- To build new PRGs out of old PRGs.
- To increase the size of the output space of the original PRG while preserving its security.

Composing PRGs

- To build new PRGs out of old PRGs.
- To increase the size of the output space of the original PRG while preserving its security.
- The proof technique used here is called hybrid argument.

Composing PRGs

- To build new PRGs out of old PRGs.
- To increase the size of the output space of the original PRG while preserving its security.
- The proof technique used here is called hybrid argument.
- The proof technique is more important than the constructions themselves.

Composing PRGs

- To build new PRGs out of old PRGs.
- To increase the size of the output space of the original PRG while preserving its security.
- The proof technique used here is called hybrid argument.
- The proof technique is more important than the constructions themselves.
- There are two basic constructions:
  - A Parallel Construction,

Composing PRGs

- To build new PRGs out of old PRGs.
- To increase the size of the output space of the original PRG while preserving its security.
- The proof technique used here is called hybrid argument.
- The proof technique is more important than the constructions themselves.
- There are two basic constructions:
  - A Parallel Construction, and
  - A Sequential construction.

A Parallel Construction

Let G be a PRG defined over $(\mathcal{S}, \mathcal{R})$. We construct a new PRG G' that applies G to γ seeds, and is defined over $(\mathcal{S}^\gamma, \mathcal{R}^\gamma)$, and for $s_1, s_2, \ldots, s_\gamma \in \mathcal{S}$,

$$G'(s_1, s_2, \ldots, s_\gamma) := (G(s_1), G(s_2), \ldots, G(s_\gamma)).$$

A Parallel Construction

A Parallel Construction

Let G be a PRG defined over $(\mathcal{S}, \mathcal{R})$. We construct a new PRG G' that applies G to γ seeds, and is defined over $(\mathcal{S}^\gamma, \mathcal{R}^\gamma)$, and for $s_1, s_2, \ldots, s_\gamma \in \mathcal{S}$,

$$G'(s_1, s_2, \ldots, s_\gamma) := (G(s_1), G(s_2), \ldots, G(s_\gamma)).$$

- G' is called the γ-wise parallel composition of G.

A Parallel Construction

A Parallel Construction

Let G be a PRG defined over $(\mathcal{S}, \mathcal{R})$. We construct a new PRG G' that applies G to γ seeds, and is defined over $(\mathcal{S}^\gamma, \mathcal{R}^\gamma)$, and for $s_1, s_2, \ldots, s_\gamma \in \mathcal{S}$,

$$G'(s_1, s_2, \ldots, s_\gamma) := (G(s_1), G(s_2), \ldots, G(s_\gamma)).$$

- G' is called the γ-wise parallel composition of G.
- γ is called a repetition parameter,

A Parallel Construction

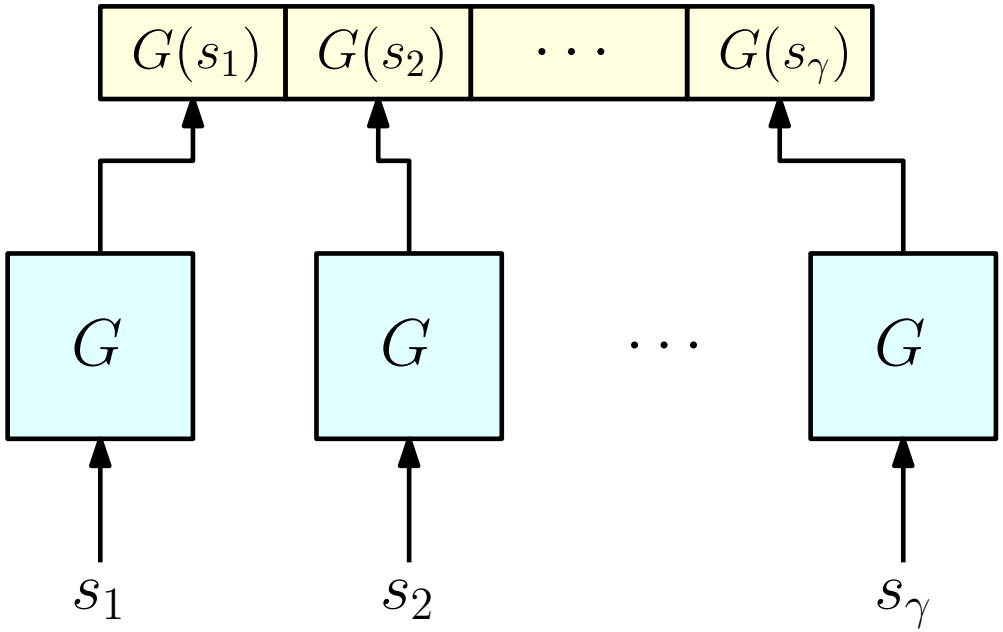
A Parallel Construction

Let G be a PRG defined over $(\mathcal{S}, \mathcal{R})$. We construct a new PRG G' that applies G to γ seeds, and is defined over $(\mathcal{S}^\gamma, \mathcal{R}^\gamma)$, and for $s_1, s_2, \ldots, s_\gamma \in \mathcal{S}$,

$$G'(s_1, s_2, \ldots, s_\gamma) := (G(s_1), G(s_2), \ldots, G(s_\gamma)).$$

- G' is called the γ-wise parallel composition of G.
- γ is called a repetition parameter, and
- γ needs to be a poly-bounded value.

A Parallel Construction



A Parallel Construction

Theorem

If G is a secure PRG, then the γ-wise parallel composition G' of G is also a secure PRG.

A Parallel Construction

Theorem

If G is a secure PRG, then the γ-wise parallel composition G' of G is also a secure PRG.
In particular, for every PRG distinguisher $\mathcal{A}$ that attacks G' as in PRG Indistinguishability Game, there exists a PRG distinguisher $\mathcal{B}$ that attacks G as in PRG Indistinguishability Game, where $\mathcal{B}$ uses $\mathcal{A}$ as sub-routine, such that

$$\mathsf{PRGadv}[\mathcal{A}, G'] = \gamma \cdot \mathsf{PRGadv}[\mathcal{B}, G].$$

A Parallel Construction

Proof

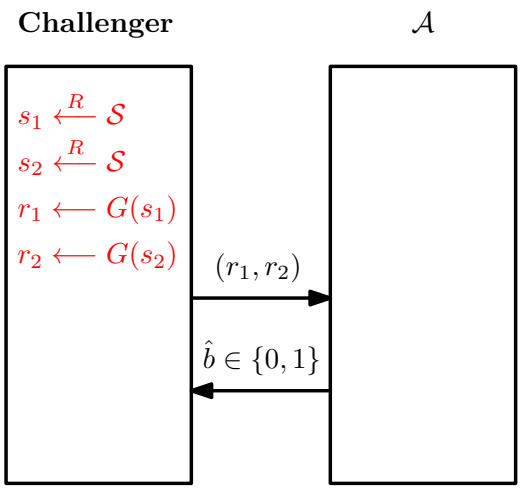
- First we proof it for $\gamma = 2$.

A Parallel Construction

Proof

- First we proof it for $\gamma = 2$.

$\gamma = 2$ and First Part:



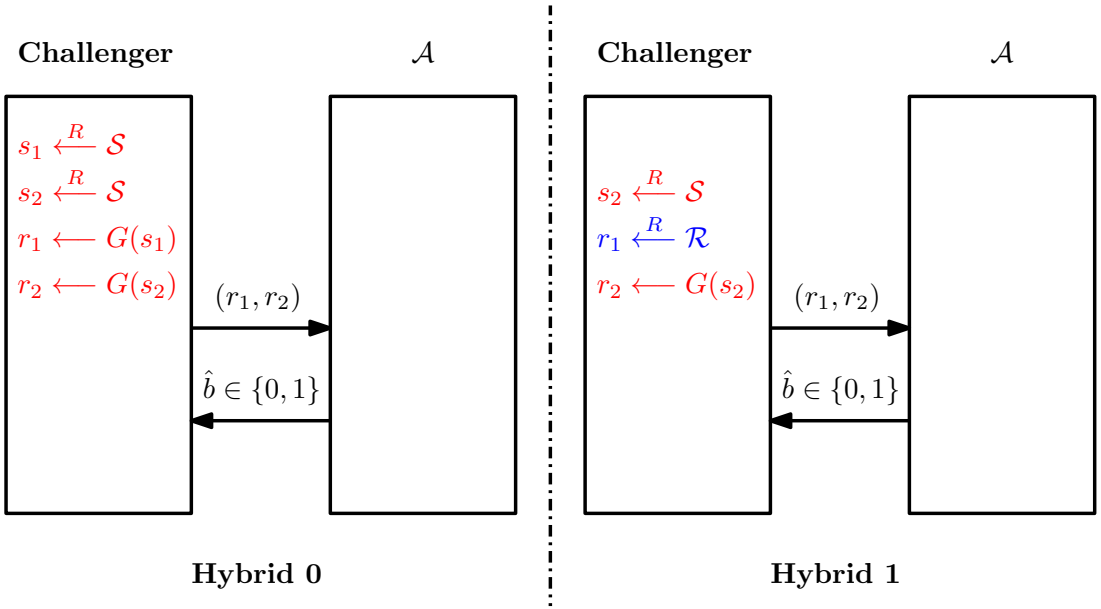
Hybrid 0

A Parallel Construction

Proof

- First we proof it for $\gamma = 2$.

$\gamma = 2$ and First Part:

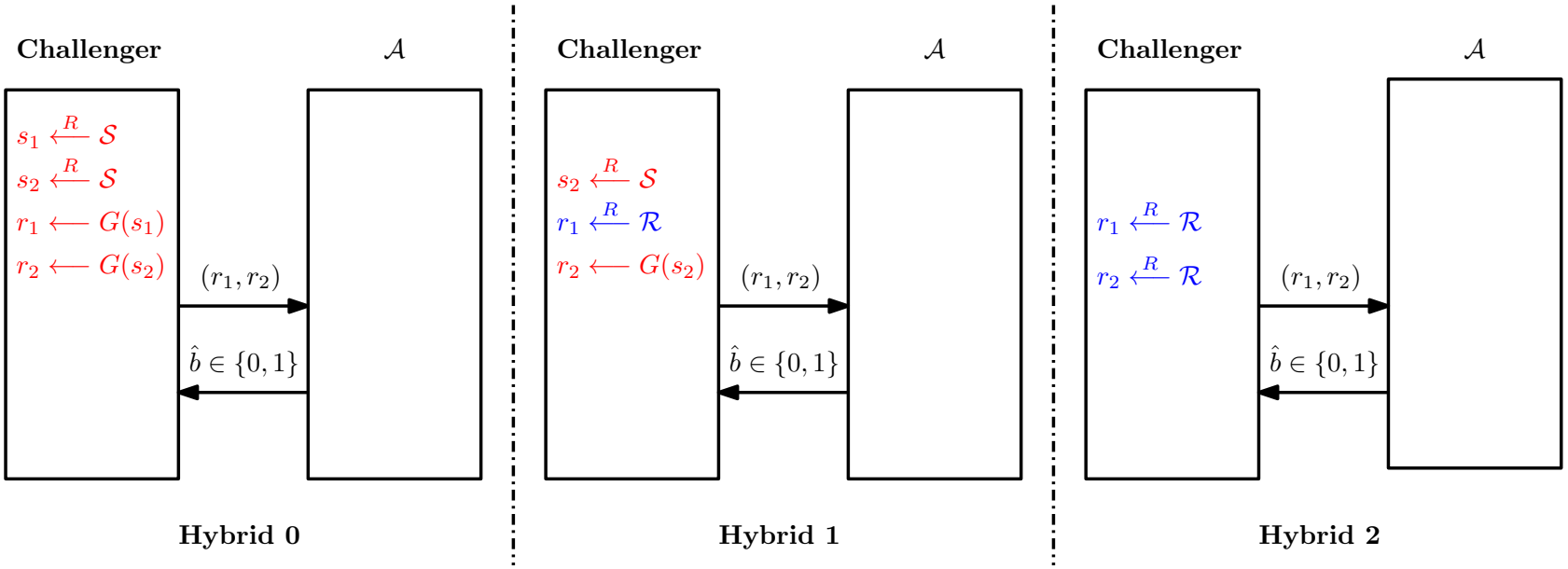


A Parallel Construction

Proof

- First we proof it for $\gamma = 2$.

$\gamma = 2$ and First Part:



A Parallel Construction

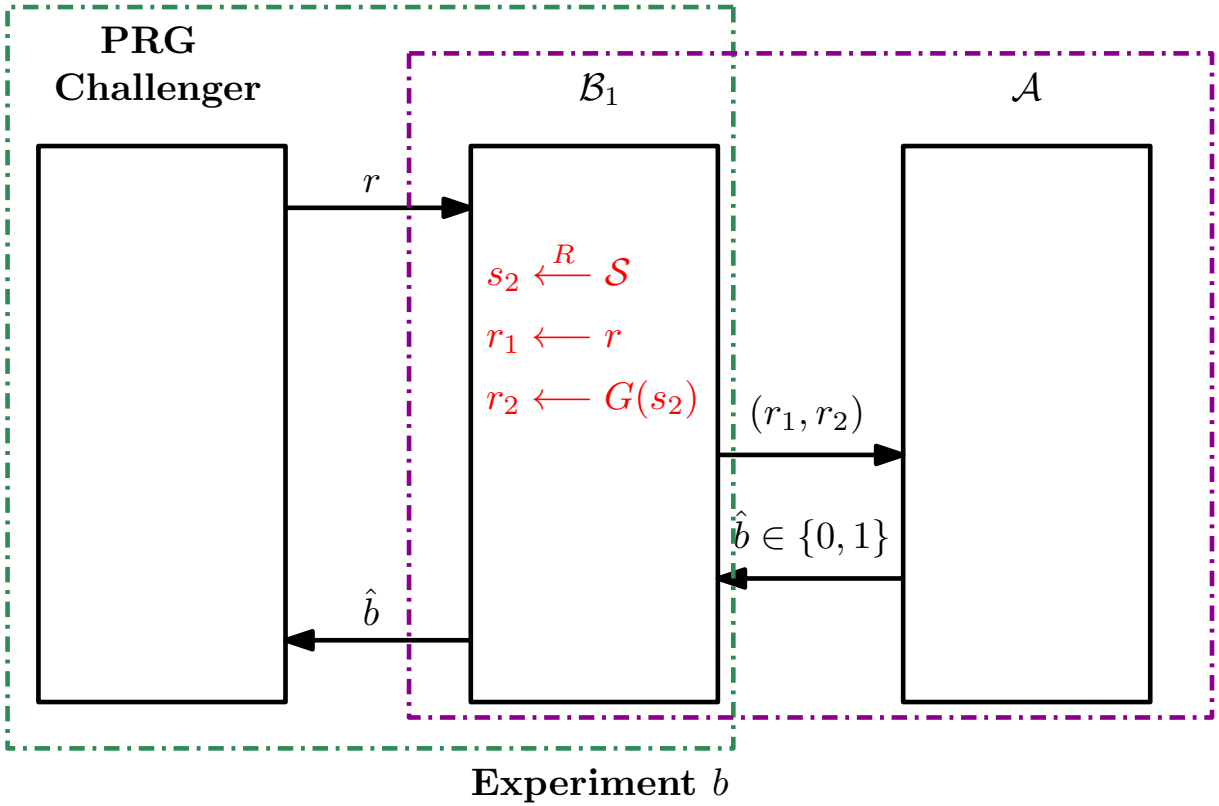
Proof

- Let p_0 = Probability that $\mathcal{A}$ outputs 1 in attack game Hybrid 0.
- Let p_1 = Probability that $\mathcal{A}$ outputs 1 in attack game Hybrid 1.
- Let p_2 = Probability that $\mathcal{A}$ outputs 1 in attack game Hybrid 2.

A Parallel Construction

$\gamma = 2$ and First Part:

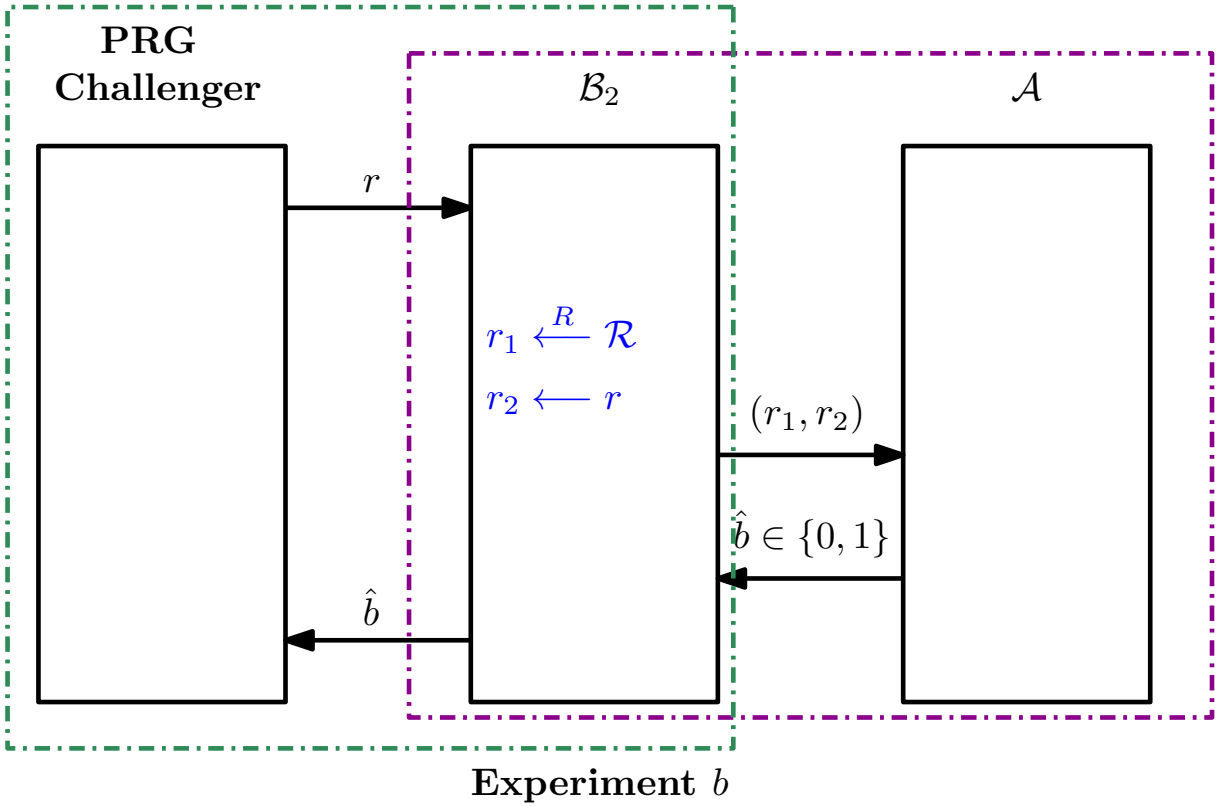
- Construction of Adversary $\mathcal{B}_1$.



A Parallel Construction

$\gamma = 2$ and First Part:

- Construction of Adversary $\mathcal{B}_2$.



A Parallel Construction

$\gamma = 2$ and First Part:

- Adversary $\mathcal{B}_1$:
  - $\Pr[\mathcal{B}_1 \text{ outputs } 1 \text{ in Experiment } 0] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 0] = p_0$.

A Parallel Construction

$\gamma = 2$ and First Part:

- Adversary $\mathcal{B}_1$:
  - $\Pr[\mathcal{B}_1 \text{ outputs } 1 \text{ in Experiment } 0] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 0] = p_0$.
  - $\Pr[\mathcal{B}_1 \text{ outputs } 1 \text{ in Experiment } 1] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 1] = p_1$.

A Parallel Construction

$\gamma = 2$ and First Part:

- Adversary $\mathcal{B}_1$:
  - $\Pr[\mathcal{B}_1 \text{ outputs 1 in Experiment 0}] = \Pr[\mathcal{A} \text{ outputs 1 in Hybrid 0}] = p_0$.
  - $\Pr[\mathcal{B}_1 \text{ outputs 1 in Experiment 1}] = \Pr[\mathcal{A} \text{ outputs 1 in Hybrid 1}] = p_1$.

$$\mathsf{PRGadv}[\mathcal{B}_1, G] = | p_1 - p_0 | \leqslant \epsilon_1(n) \text{ as } G \text{ is secure.}$$

A Parallel Construction

$\gamma = 2$ and First Part:

- Adversary $\mathcal{B}_1$:
  - $\Pr[\mathcal{B}_1 \text{ outputs 1 in Experiment 0}] = \Pr[\mathcal{A} \text{ outputs 1 in Hybrid 0}] = p_0$.
  - $\Pr[\mathcal{B}_1 \text{ outputs 1 in Experiment 1}] = \Pr[\mathcal{A} \text{ outputs 1 in Hybrid 1}] = p_1$.

$$\mathsf{PRGadv}[\mathcal{B}_1, G] = | p_1 - p_0 | \leqslant \epsilon_1(n) \text{ as } G \text{ is secure.}$$

- Adversary $\mathcal{B}_2$:
  - $\Pr[\mathcal{B}_2 \text{ outputs 1 in Experiment 0}] = \Pr[\mathcal{A} \text{ outputs 1 in Hybrid 1}] = p_1$.

A Parallel Construction

$\gamma = 2$ and First Part:

- Adversary $\mathcal{B}_1$:
  - $\Pr[\mathcal{B}_1 \text{ outputs } 1 \text{ in Experiment } 0] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 0] = p_0$.
  - $\Pr[\mathcal{B}_1 \text{ outputs } 1 \text{ in Experiment } 1] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 1] = p_1$.

$$\mathsf{PRGadv}[\mathcal{B}_1, G] = | p_1 - p_0 | \leqslant \epsilon_1(n) \text{ as } G \text{ is secure.}$$

- Adversary $\mathcal{B}_2$:
  - $\Pr[\mathcal{B}_2 \text{ outputs } 1 \text{ in Experiment } 0] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 1] = p_1$.
  - $\Pr[\mathcal{B}_2 \text{ outputs } 1 \text{ in Experiment } 1] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 2] = p_2$.

A Parallel Construction

$\gamma = 2$ and First Part:

- Adversary $\mathcal{B}_1$:
  - $\Pr[\mathcal{B}_1 \text{ outputs } 1 \text{ in Experiment } 0] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 0] = p_0$.
  - $\Pr[\mathcal{B}_1 \text{ outputs } 1 \text{ in Experiment } 1] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 1] = p_1$.

$$\mathsf{PRGadv}[\mathcal{B}_1, G] = | p_1 - p_0 | \leqslant \epsilon_1(n) \text{ as } G \text{ is secure.}$$

- Adversary $\mathcal{B}_2$:
  - $\Pr[\mathcal{B}_2 \text{ outputs } 1 \text{ in Experiment } 0] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 1] = p_1$.
  - $\Pr[\mathcal{B}_2 \text{ outputs } 1 \text{ in Experiment } 1] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in Hybrid } 2] = p_2$.

$$\mathsf{PRGadv}[\mathcal{B}_2, G] = | p_2 - p_1 | \leqslant \epsilon_2(n) \text{ as } G \text{ is secure.}$$

A Parallel Construction

$\gamma = 2$ and First Part:

- Adversary $\mathcal{A}$:

$$\begin{aligned} \mathsf{PRGadv}[\mathcal{A}, G'] &= |\, p_2 - p_0 \,| \\ &= |\, p_2 - p_1 + p_1 - p_0 \,| \end{aligned}$$

A Parallel Construction

$\gamma = 2$ and First Part:

- Adversary $\mathcal{A}$:

$$\begin{aligned} \mathsf{PRGadv}[\mathcal{A}, G'] &= |\, p_2 - p_0 \,| \\ &= |\, p_2 - p_1 + p_1 - p_0 \,| \\ &\leqslant |\, p_2 - p_1 \,| + |\, p_1 - p_0 \,| \end{aligned}$$

A Parallel Construction

$\gamma = 2$ and First Part:

- Adversary $\mathcal{A}$:

$$\begin{aligned} \mathsf{PRGadv}[\mathcal{A}, G'] &= |\, p_2 - p_0 \,| \\ &= |\, p_2 - p_1 + p_1 - p_0 \,| \\ &\leqslant |\, p_2 - p_1 \,| + |\, p_1 - p_0 \,| \\ &\leqslant \epsilon_1(n) + \epsilon_2(n). \end{aligned}$$

A Parallel Construction

$\gamma = 2$ and Second Part:

- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1$ and $\mathcal{B}_2$.

A Parallel Construction

$\gamma = 2$ and Second Part:

- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1$ and $\mathcal{B}_2$.
  - $\mathcal{B}$ computes $\omega \xleftarrow{R} \{1,2\}$.

A Parallel Construction

$\gamma = 2$ and Second Part:

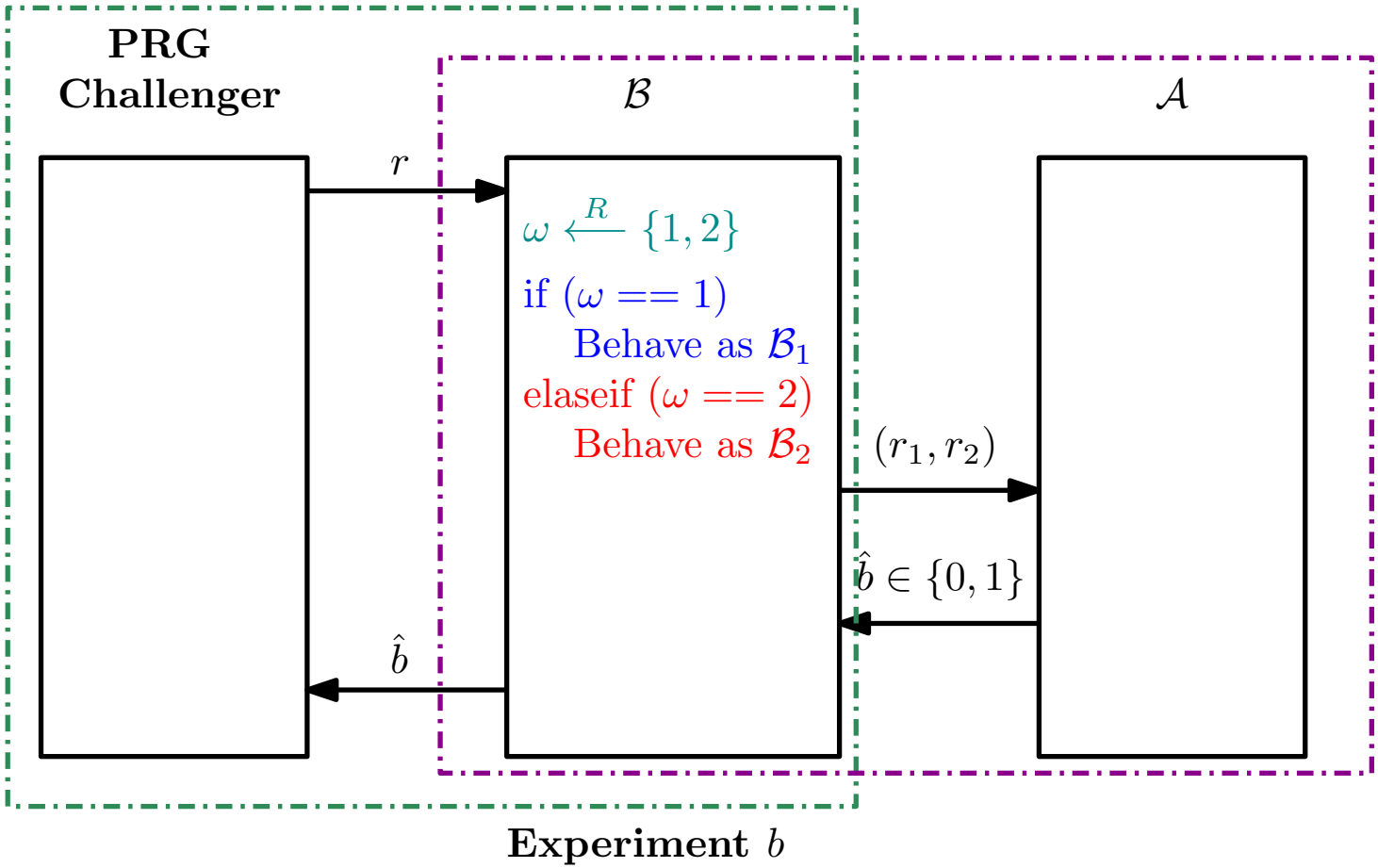
- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1$ and $\mathcal{B}_2$.
  - $\mathcal{B}$ computes $\omega \xleftarrow{R} \{1,2\}$.
  - Follow the behavior of $\mathcal{B}_\omega$.

A Parallel Construction

$\gamma = 2$ and Second Part:

- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1$ and $\mathcal{B}_2$.
  - $\mathcal{B}$ computes $\omega \xleftarrow{R} \{1,2\}$.
  - Follow the behavior of $\mathcal{B}_\omega$.
- Let W_b be the event where $\mathcal{B}$ outputs 1 in Experiment b.

A Parallel Construction



A Parallel Construction

$\gamma = 2$ and Second Part:

$$\begin{aligned}\Pr[W_0] &= \Pr[W_0 \wedge \omega = 1] + \Pr[W_0 \wedge \omega = 2] \\ &= \Pr[W_0 \mid \omega = 1]\Pr[\omega = 1] + \Pr[W_0 \mid \omega = 2]\Pr[\omega = 2]\end{aligned}$$

A Parallel Construction

$\gamma = 2$ and Second Part:

$$\begin{aligned}
\Pr[W_0] &= \Pr[W_0 \wedge \omega = 1] + \Pr[W_0 \wedge \omega = 2] \\
&= \Pr[W_0 \mid \omega = 1]\Pr[\omega = 1] + \Pr[W_0 \mid \omega = 2]\Pr[\omega = 2] \\
&= \frac{1}{2}\left(\Pr[W_0 \mid \omega = 1] + \Pr[W_0 \mid \omega = 2]\right)
\end{aligned}$$

A Parallel Construction

$\gamma = 2$ and Second Part:

$$
\begin{aligned}
\Pr[W_0] &= \Pr[W_0 \wedge \omega = 1] + \Pr[W_0 \wedge \omega = 2] \\
&= \Pr[W_0 \mid \omega = 1]\Pr[\omega = 1] + \Pr[W_0 \mid \omega = 2]\Pr[\omega = 2] \\
&= \frac{1}{2}\left(\Pr[W_0 \mid \omega = 1] + \Pr[W_0 \mid \omega = 2]\right) \\
&= \frac{1}{2}(p_0 + p_1).
\end{aligned}
$$

A Parallel Construction

$\gamma = 2$ and Second Part:

$$
\begin{aligned}
\Pr[W_0] &= \Pr[W_0 \wedge \omega = 1] + \Pr[W_0 \wedge \omega = 2] \\
&= \Pr[W_0 \mid \omega = 1]\Pr[\omega = 1] + \Pr[W_0 \mid \omega = 2]\Pr[\omega = 2] \\
&= \frac{1}{2}\left(\Pr[W_0 \mid \omega = 1] + \Pr[W_0 \mid \omega = 2]\right) \\
&= \frac{1}{2}(p_0 + p_1).
\end{aligned}
$$

$$
\begin{aligned}
\Pr[W_1] &= \Pr[W_1 \wedge \omega = 1] + \Pr[W_1 \wedge \omega = 2] \\
&= \Pr[W_1 \mid \omega = 1]\Pr[\omega = 1] + \Pr[W_1 \mid \omega = 2]\Pr[\omega = 2]
\end{aligned}
$$

A Parallel Construction

$\gamma = 2$ and Second Part:

$$
\begin{aligned}
\Pr[W_0] &= \Pr[W_0 \wedge \omega = 1] + \Pr[W_0 \wedge \omega = 2] \\
&= \Pr[W_0 \mid \omega = 1]\Pr[\omega = 1] + \Pr[W_0 \mid \omega = 2]\Pr[\omega = 2] \\
&= \frac{1}{2}\left(\Pr[W_0 \mid \omega = 1] + \Pr[W_0 \mid \omega = 2]\right) \\
&= \frac{1}{2}(p_0 + p_1).
\end{aligned}
$$

$$
\begin{aligned}
\Pr[W_1] &= \Pr[W_1 \wedge \omega = 1] + \Pr[W_1 \wedge \omega = 2] \\
&= \Pr[W_1 \mid \omega = 1]\Pr[\omega = 1] + \Pr[W_1 \mid \omega = 2]\Pr[\omega = 2] \\
&= \frac{1}{2}\left(\Pr[W_1 \mid \omega = 1] + \Pr[W_1 \mid \omega = 2]\right)
\end{aligned}
$$

A Parallel Construction

$\gamma = 2$ and Second Part:

$$
\begin{aligned}
\Pr[W_0] &= \Pr[W_0 \wedge \omega = 1] + \Pr[W_0 \wedge \omega = 2] \\
&= \Pr[W_0 \mid \omega = 1]\Pr[\omega = 1] + \Pr[W_0 \mid \omega = 2]\Pr[\omega = 2] \\
&= \frac{1}{2}\left(\Pr[W_0 \mid \omega = 1] + \Pr[W_0 \mid \omega = 2]\right) \\
&= \frac{1}{2}(p_0 + p_1).
\end{aligned}
$$

$$
\begin{aligned}
\Pr[W_1] &= \Pr[W_1 \wedge \omega = 1] + \Pr[W_1 \wedge \omega = 2] \\
&= \Pr[W_1 \mid \omega = 1]\Pr[\omega = 1] + \Pr[W_1 \mid \omega = 2]\Pr[\omega = 2] \\
&= \frac{1}{2}\left(\Pr[W_1 \mid \omega = 1] + \Pr[W_1 \mid \omega = 2]\right) \\
&= \frac{1}{2}(p_1 + p_2).
\end{aligned}
$$

A Parallel Construction

$\gamma = 2$ and Second Part:

$$\begin{aligned} \mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_1] - \Pr[W_0]| \\ &= \left| \frac{1}{2}(p_1 + p_2) - \frac{1}{2}(p_0 + p_1) \right| \end{aligned}$$

A Parallel Construction

$\gamma = 2$ and Second Part:

$$
\begin{aligned}
\mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_1] - \Pr[W_0]| \\
&= \left| \frac{1}{2}(p_1 + p_2) - \frac{1}{2}(p_0 + p_1) \right| \\
&= \frac{1}{2} | p_2 - p_0 | \\
&= \frac{1}{2} \cdot \mathsf{PRGadv}[\mathcal{A}, G'].
\end{aligned}
$$

A Parallel Construction

$\gamma = 2$ and Second Part:

$$\begin{aligned}
\mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_1] - \Pr[W_0]| \\
&= \left| \frac{1}{2}(p_1 + p_2) - \frac{1}{2}(p_0 + p_1) \right| \\
&= \frac{1}{2}|\, p_2 - p_0 \,| \\
&= \frac{1}{2} \cdot \mathsf{PRGadv}[\mathcal{A}, G'].
\end{aligned}$$

Therefore,

$$\mathsf{PRGadv}[\mathcal{A}, G'] = 2 \cdot \mathsf{PRGadv}[\mathcal{B}, G].$$

A Parallel Construction

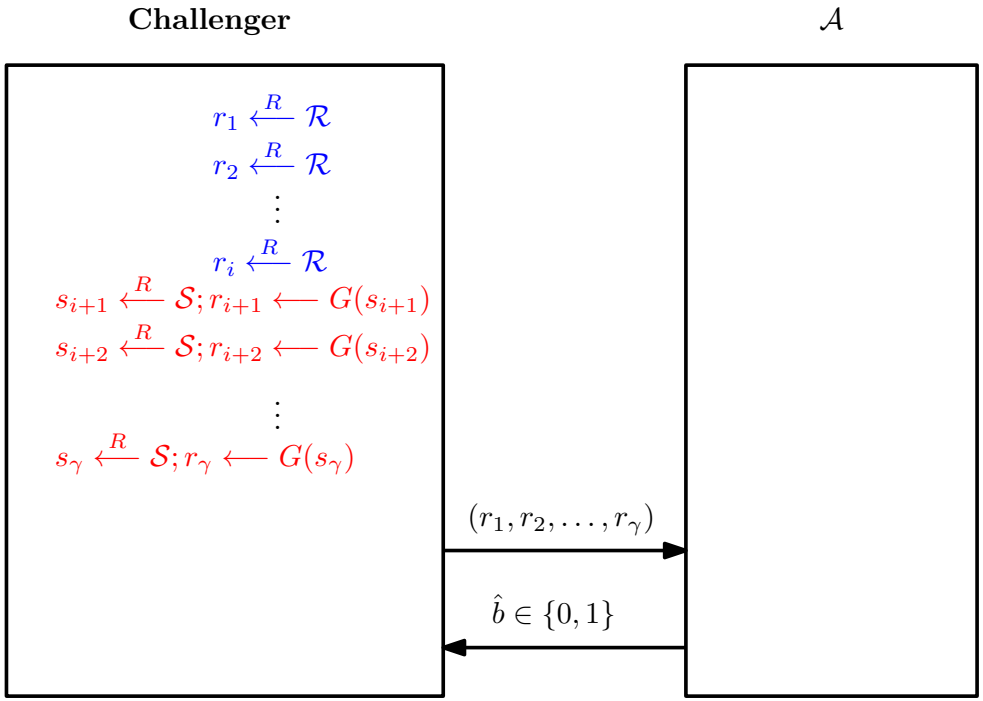
For polynomially bounded γ:

- There will be $\gamma + 1$ hybrid attack games: Hybrid 0, Hybrid 1, …, Hybrid γ.

A Parallel Construction

For polynomially bounded γ:

- There will be $\gamma + 1$ hybrid attack games: Hybrid 0, Hybrid 1, …, Hybrid γ.

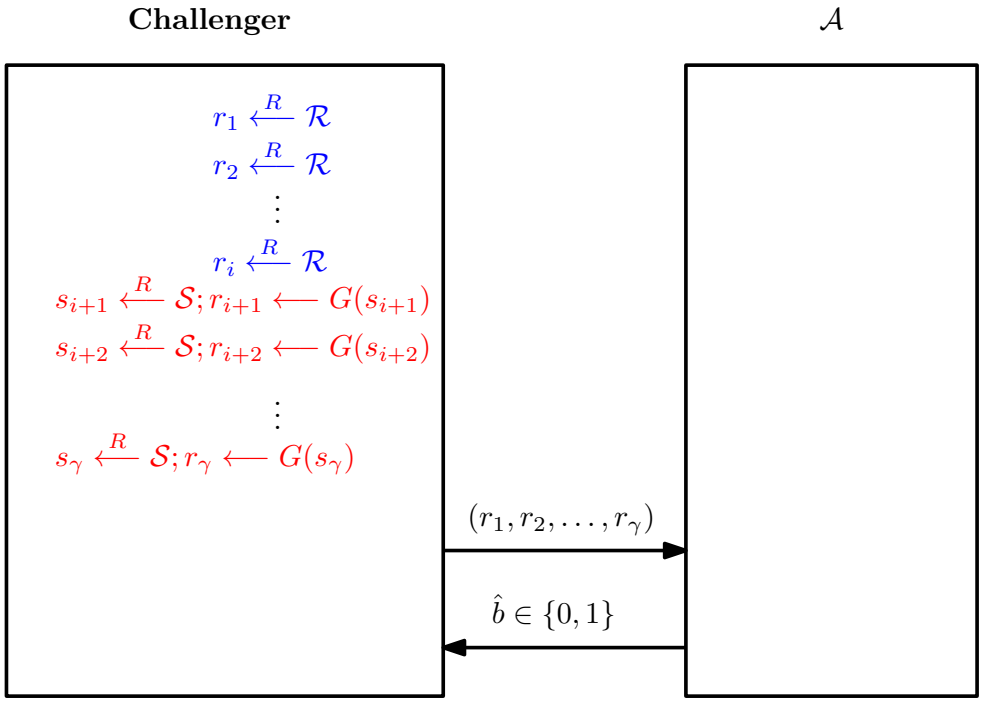


Hybrid i

A Parallel Construction

For polynomially bounded γ:

- There will be $\gamma + 1$ hybrid attack games: Hybrid 0, Hybrid 1, …, Hybrid γ.



Hybrid i

- Let p_i = Probability that $\mathcal{A}$ outputs 1 in attack game Hybrid i.

A Parallel Construction

Hybrid 0	$G(s_1)$	$G(s_2)$	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$

A Parallel Construction

Hybrid 0	$G(s_1)$	$G(s_2)$	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$
Hybrid 1	r_1	$G(s_2)$	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$

A Parallel Construction

Hybrid 0	$G(s_1)$	$G(s_2)$	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$
Hybrid 1	r_1	$G(s_2)$	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$
Hybrid 2	r_1	r_2	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$

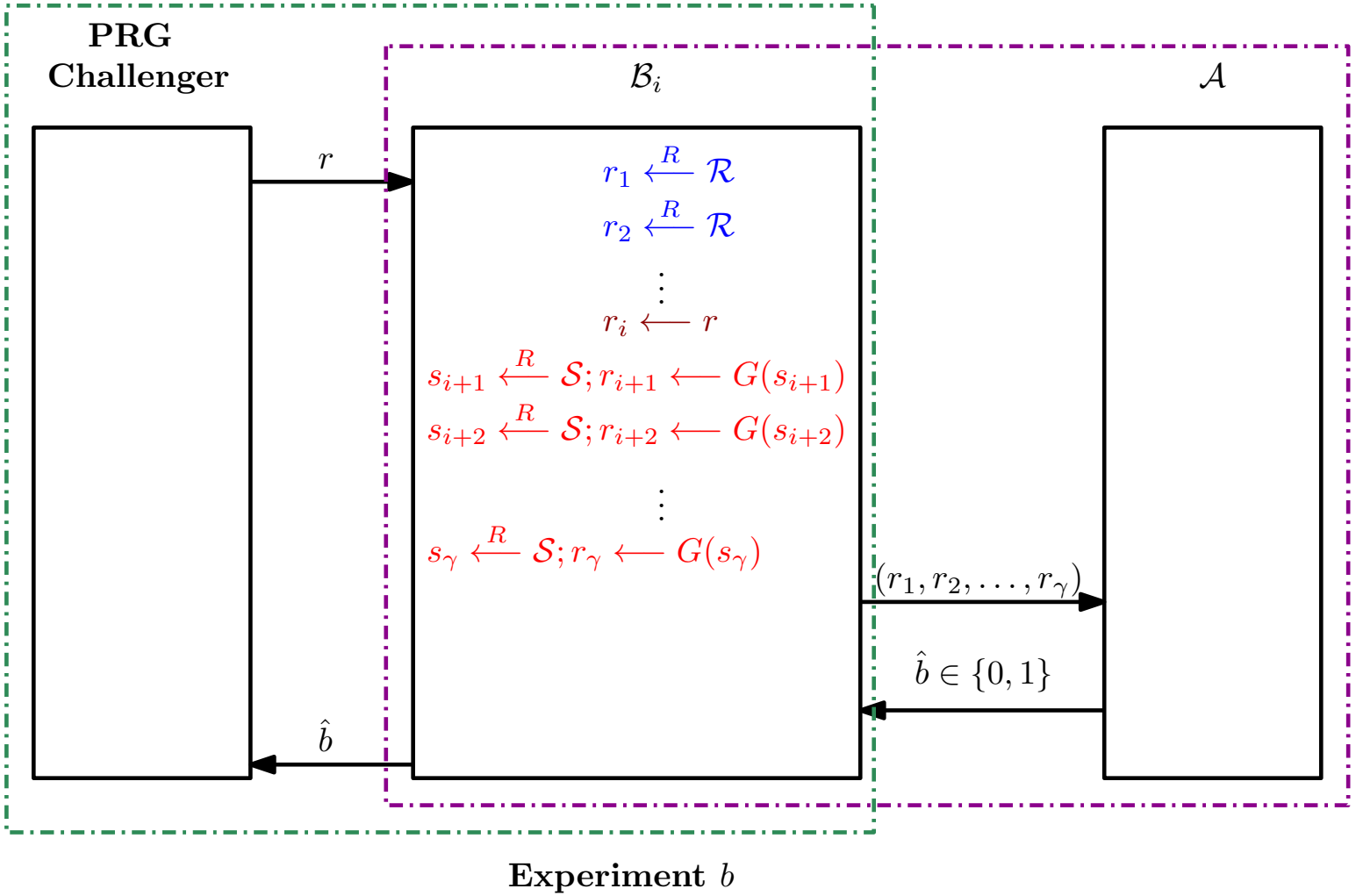
A Parallel Construction

Hybrid 0	$G(s_1)$	$G(s_2)$	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$
Hybrid 1	r_1	$G(s_2)$	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$
Hybrid 2	r_1	r_2	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Hybrid $\gamma-1$	r_1	r_2	r_3	$\cdots$	$r_{\gamma-1}$	$G(s_\gamma)$

A Parallel Construction

Hybrid 0	$G(s_1)$	$G(s_2)$	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$
Hybrid 1	r_1	$G(s_2)$	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$
Hybrid 2	r_1	r_2	$G(s_3)$	$\cdots$	$G(s_{\gamma-1})$	$G(s_\gamma)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Hybrid $\gamma-1$	r_1	r_2	r_3	$\cdots$	$r_{\gamma-1}$	$G(s_\gamma)$
Hybrid γ	r_1	r_2	r_3	$\cdots$	$r_{\gamma-1}$	r_γ

A Parallel Construction



Experiment b

A Parallel Construction

For polynomially bounded γ:

- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\gamma$.

A Parallel Construction

For polynomially bounded γ:

- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\gamma$.
  - $\mathcal{B}$ computes $\omega \xleftarrow{R} \{1, 2, \ldots, \gamma\}$.

A Parallel Construction

For polynomially bounded γ:

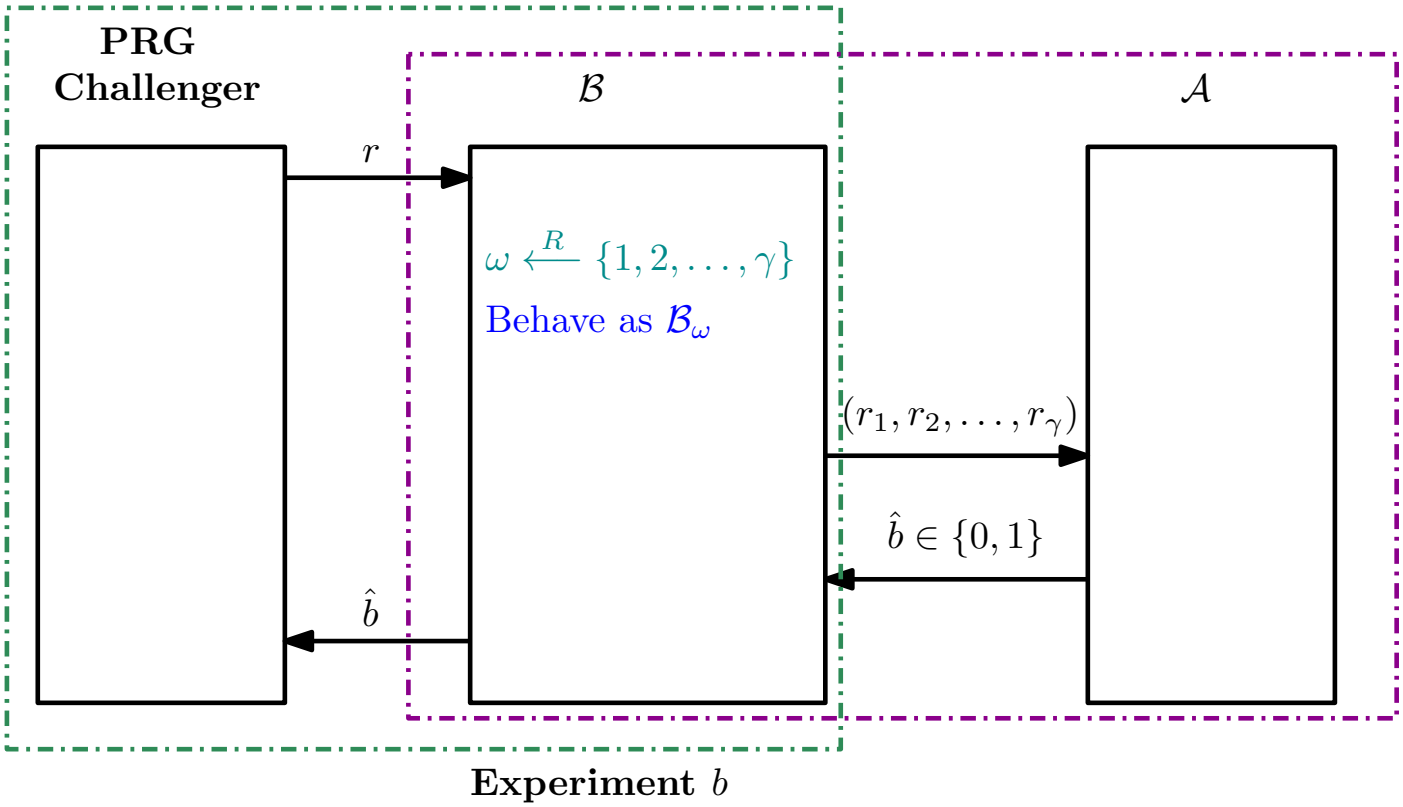
- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\gamma$.
  - $\mathcal{B}$ computes $\omega \xleftarrow{R} \{1, 2, \ldots, \gamma\}$.
  - Follow the behavior of $\mathcal{B}_\omega$.

A Parallel Construction

For polynomially bounded γ:

- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\gamma$.
  - $\mathcal{B}$ computes $\omega \xleftarrow{R} \{1, 2, \ldots, \gamma\}$.
  - Follow the behavior of $\mathcal{B}_\omega$.
- Let W_b be the event where $\mathcal{B}$ outputs 1 in Experiment b.

A Parallel Construction



A Parallel Construction

For polynomially bounded γ:

$$
\begin{aligned}
\Pr[W_0] &= \sum_{i=1}^{\gamma} \Pr[W_0 \wedge \omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_0 \mid \omega = i] \Pr[\omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_0 \mid \omega = i] \frac{1}{\gamma} \\
&= \frac{1}{\gamma} \sum_{i=1}^{\gamma} p_{i-1}
\end{aligned}
$$

A Parallel Construction

For polynomially bounded γ:

$$
\begin{aligned}
\Pr[W_0] &= \sum_{i=1}^{\gamma} \Pr[W_0 \wedge \omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_0 \mid \omega = i] \Pr[\omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_0 \mid \omega = i] \frac{1}{\gamma} \\
&= \frac{1}{\gamma} \sum_{i=1}^{\gamma} p_{i-1}
\end{aligned}
\qquad
\begin{aligned}
\Pr[W_1] &= \sum_{i=1}^{\gamma} \Pr[W_1 \wedge \omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_1 \mid \omega = i] \Pr[\omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_1 \mid \omega = i] \frac{1}{\gamma} \\
&= \frac{1}{\gamma} \sum_{i=1}^{\gamma} p_i
\end{aligned}
$$

A Parallel Construction

For polynomially bounded γ:

$$\mathsf{PRGadv}[\mathcal{B}, G] \quad = \quad |\Pr[W_1] - \Pr[W_0]|$$

A Parallel Construction

For polynomially bounded γ:

$$\begin{aligned} \mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_1] - \Pr[W_0]| \\ &= \left| \left(\frac{1}{\gamma}\sum_{i=1}^{\gamma} p_i\right) - \left(\frac{1}{\gamma}\sum_{i=1}^{\gamma} p_{i-1}\right) \right| \end{aligned}$$

A Parallel Construction

For polynomially bounded γ:

$$
\begin{aligned}
\mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_1] - \Pr[W_0]| \\
&= \left| \left(\frac{1}{\gamma} \sum_{i=1}^{\gamma} p_i \right) - \left(\frac{1}{\gamma} \sum_{i=1}^{\gamma} p_{i-1} \right) \right| \\
&= \frac{1}{\gamma} | \, p_\gamma - p_0 \, | \\
&= \frac{1}{\gamma} \cdot \mathsf{PRGadv}[\mathcal{A}, G'].
\end{aligned}
$$

A Parallel Construction

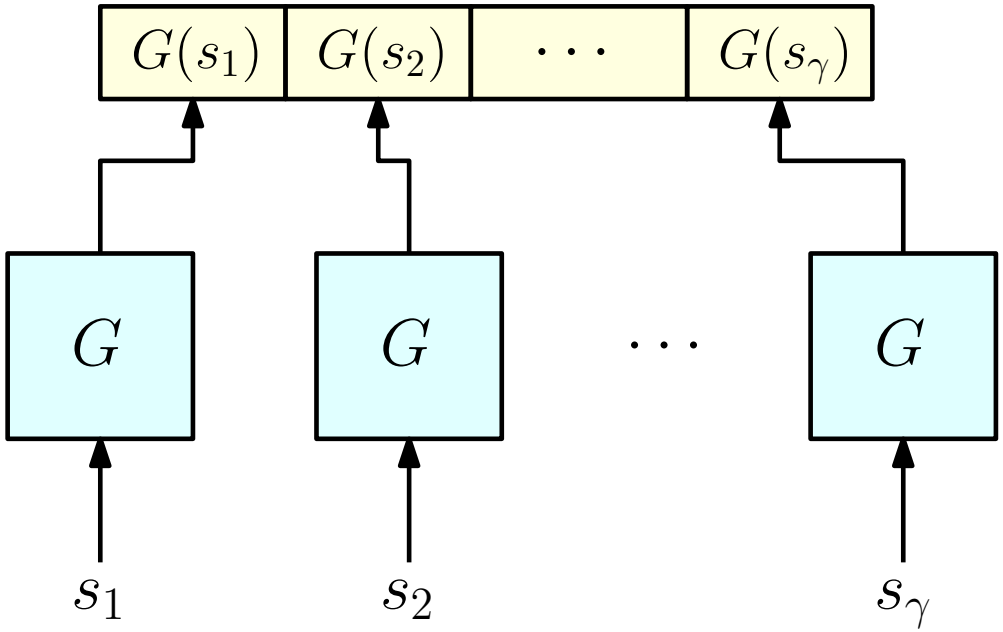
For polynomially bounded γ:

$$\begin{aligned}
\mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_1] - \Pr[W_0]| \\
&= \left| \left(\frac{1}{\gamma} \sum_{i=1}^{\gamma} p_i \right) - \left(\frac{1}{\gamma} \sum_{i=1}^{\gamma} p_{i-1} \right) \right| \\
&= \frac{1}{\gamma} | \, p_\gamma - p_0 \, | \\
&= \frac{1}{\gamma} \cdot \mathsf{PRGadv}[\mathcal{A}, G'].
\end{aligned}$$

Therefore,

$$\mathsf{PRGadv}[\mathcal{A}, G'] = \gamma \cdot \mathsf{PRGadv}[\mathcal{B}, G].$$

A Parallel Construction



A Sequential construction

Blum-Micali method

Let G be a PRG defined over $(\mathcal{S}, \mathcal{R} \times \mathcal{S})$. We construct a new PRG G' that is defined over $(\mathcal{S}, \mathcal{R}^\gamma \times \mathcal{S})$, and for $s \in \mathcal{S}$,

A Sequential construction

Blum-Micali method

Let G be a PRG defined over $(\mathcal{S}, \mathcal{R} \times \mathcal{S})$. We construct a new PRG G' that is defined over $(\mathcal{S}, \mathcal{R}^\gamma \times \mathcal{S})$, and for $s \in \mathcal{S}$,

$$
\begin{aligned}
G'(s) \quad := \quad & \{s_0 \longleftarrow s \\
& \text{For } i = 1, 2, \ldots, \gamma \text{ do} \\
& \quad (r_i, s_i) \longleftarrow G(s_{i-1}) \\
& \text{Output } (r_1, r_2, \ldots, r_\gamma, s_\gamma)\}
\end{aligned}
$$

A Sequential construction

Blum-Micali method

Let G be a PRG defined over $(\mathcal{S}, \mathcal{R} \times \mathcal{S})$. We construct a new PRG G' that is defined over $(\mathcal{S}, \mathcal{R}^\gamma \times \mathcal{S})$, and for $s \in \mathcal{S}$,

$$
\begin{aligned}
G'(s) \quad := \quad & \{s_0 \longleftarrow s \\
& \text{For } i = 1, 2, \ldots, \gamma \text{ do} \\
& \quad (r_i, s_i) \longleftarrow G(s_{i-1}) \\
& \text{Output } (r_1, r_2, \ldots, r_\gamma, s_\gamma)\}
\end{aligned}
$$

- G' is called the γ-wise sequential composition of G.

A Sequential construction

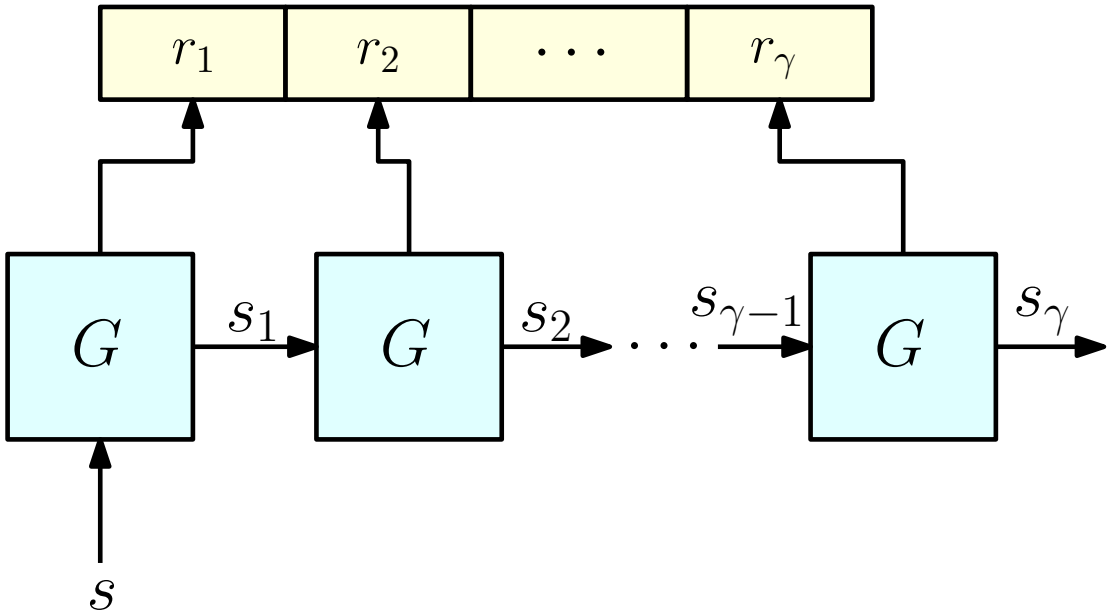
Blum-Micali method

Let G be a PRG defined over $(\mathcal{S}, \mathcal{R} \times \mathcal{S})$. We construct a new PRG G' that is defined over $(\mathcal{S}, \mathcal{R}^\gamma \times \mathcal{S})$, and for $s \in \mathcal{S}$,

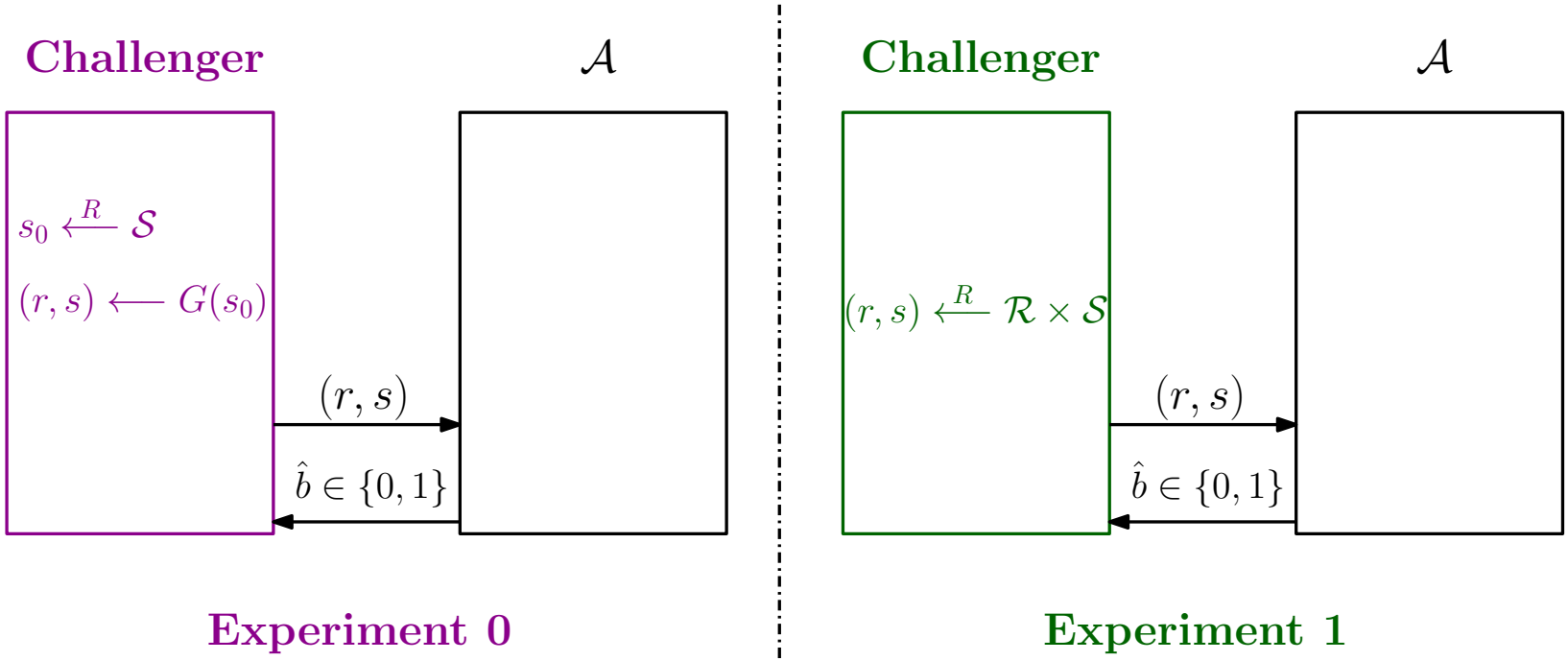
$$
\begin{aligned}
G'(s) \quad := \quad & \{s_0 \longleftarrow s \\
& \text{For } i = 1, 2, \ldots, \gamma \text{ do} \\
& \quad (r_i, s_i) \longleftarrow G(s_{i-1}) \\
& \text{Output } (r_1, r_2, \ldots, r_\gamma, s_\gamma)\}
\end{aligned}
$$

- G' is called the γ-wise sequential composition of G.
- γ needs to be a poly-bounded value.

A Sequential Construction



Pseudo-Random Generator Advantage



A Sequential Construction

Theorem

If G is a secure PRG, then the γ-wise sequential composition G' of G is also a secure PRG.

A Sequential Construction

Theorem

If G is a secure PRG, then the γ-wise sequential composition G' of G is also a secure PRG.
In particular, for every PRG distinguisher $\mathcal{A}$ that attacks G' as in PRG Indistinguishability Game, there exists a PRG distinguisher $\mathcal{B}$ that attacks G as in PRG Indistinguishability Game, where $\mathcal{B}$ uses $\mathcal{A}$ as sub-routine, such that

$$\mathsf{PRGadv}[\mathcal{A}, G'] = \gamma \cdot \mathsf{PRGadv}[\mathcal{B}, G].$$

A Sequential Construction

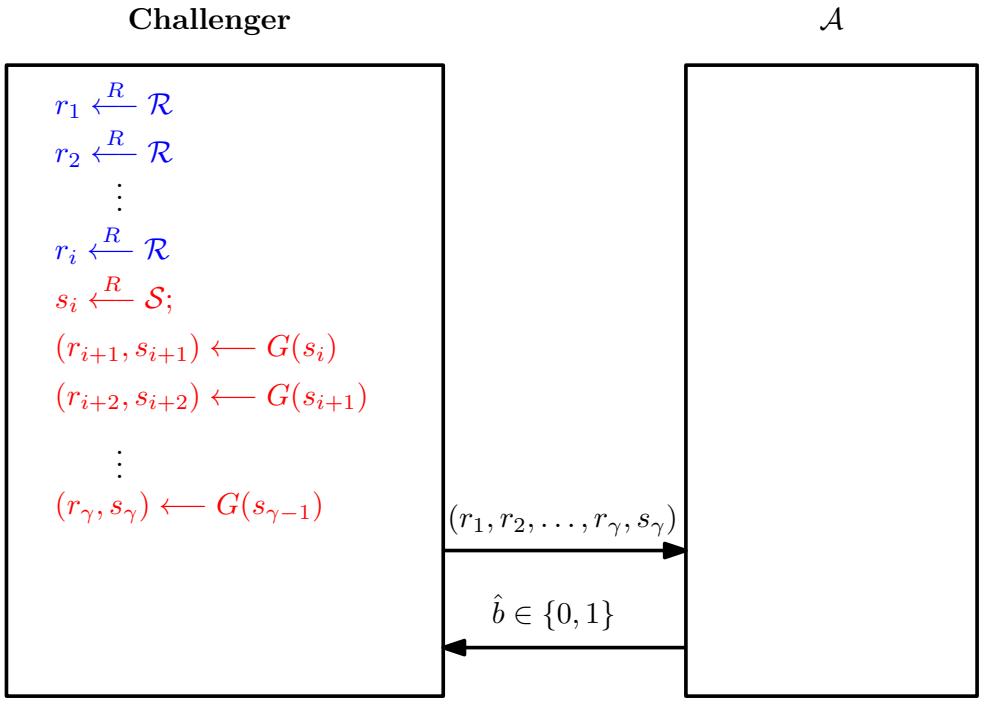
For polynomially bounded γ:

- There will be $\gamma + 1$ hybrid attack games: Hybrid 0, Hybrid 1, …, Hybrid γ.

A Sequential Construction

For polynomially bounded γ:

- There will be $\gamma + 1$ hybrid attack games: Hybrid 0, Hybrid 1, …, Hybrid γ.

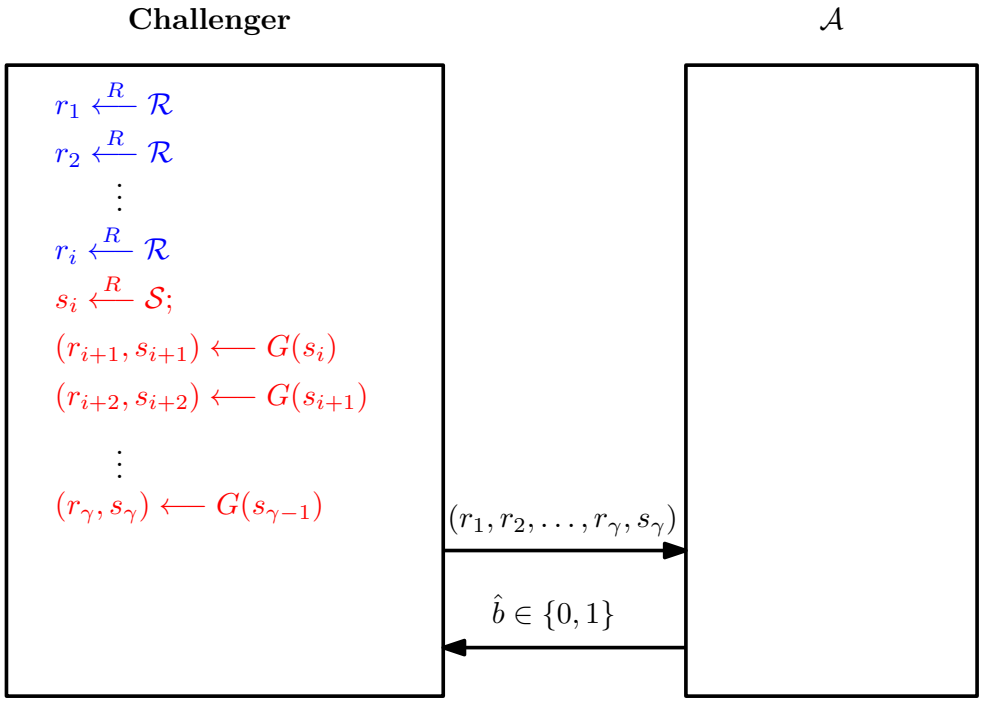


Hybrid i

A Sequential Construction

For polynomially bounded γ:

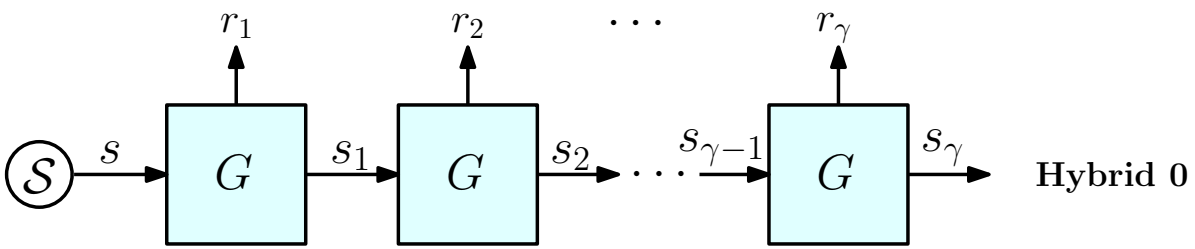
- There will be $\gamma + 1$ hybrid attack games: Hybrid 0, Hybrid 1, …, Hybrid γ.



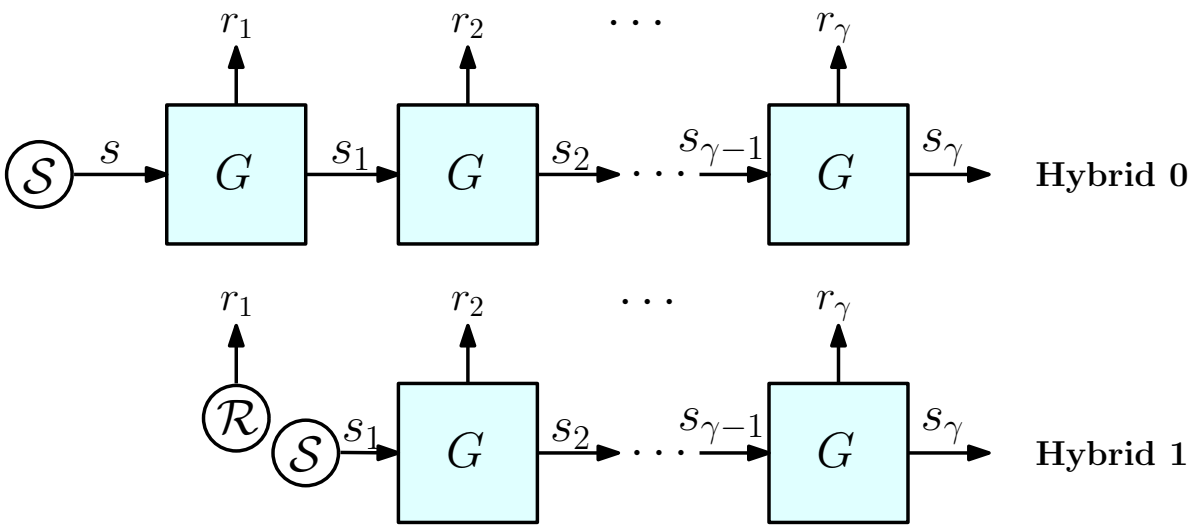
Hybrid i

- Let p_i = Probability that $\mathcal{A}$ outputs 1 in attack game Hybrid i.

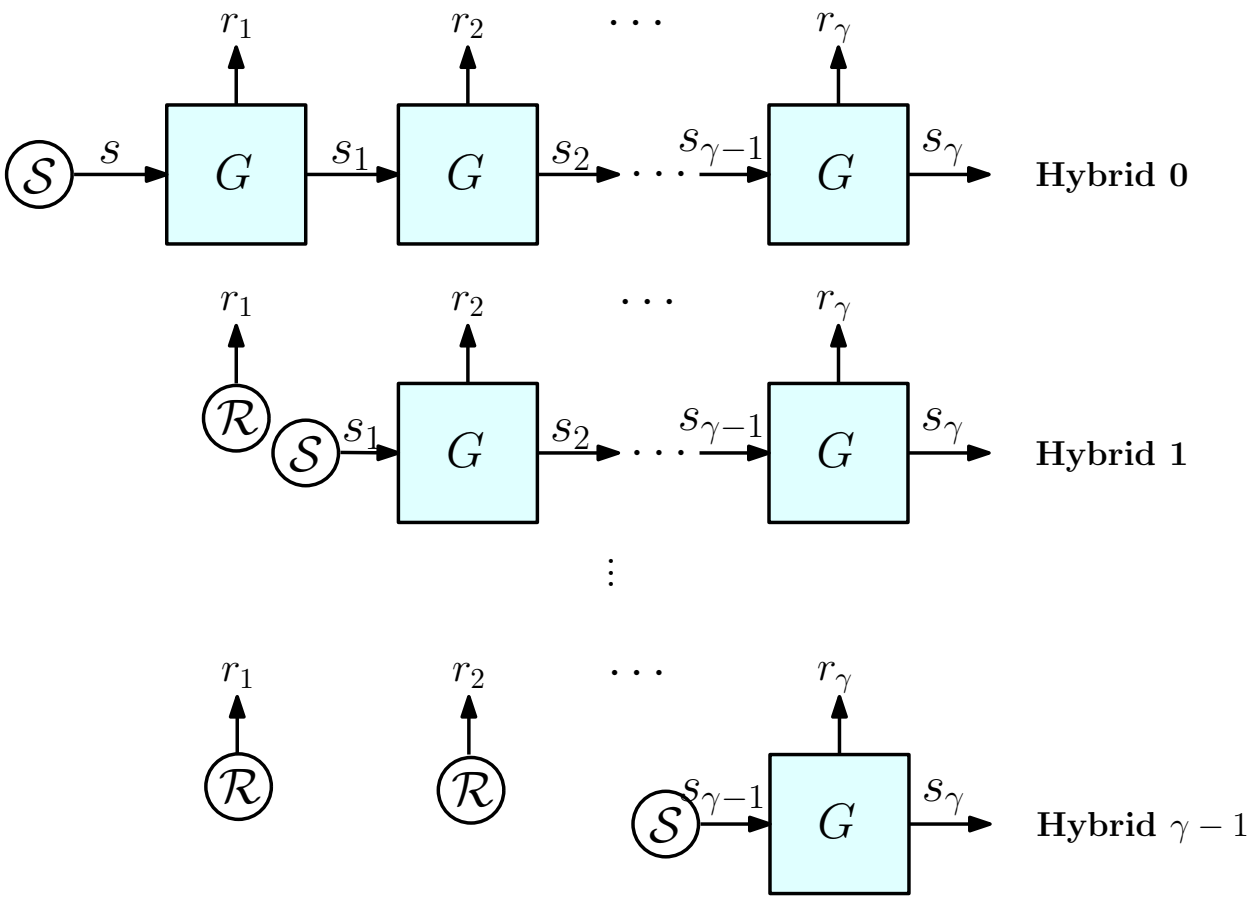
A Sequential Construction



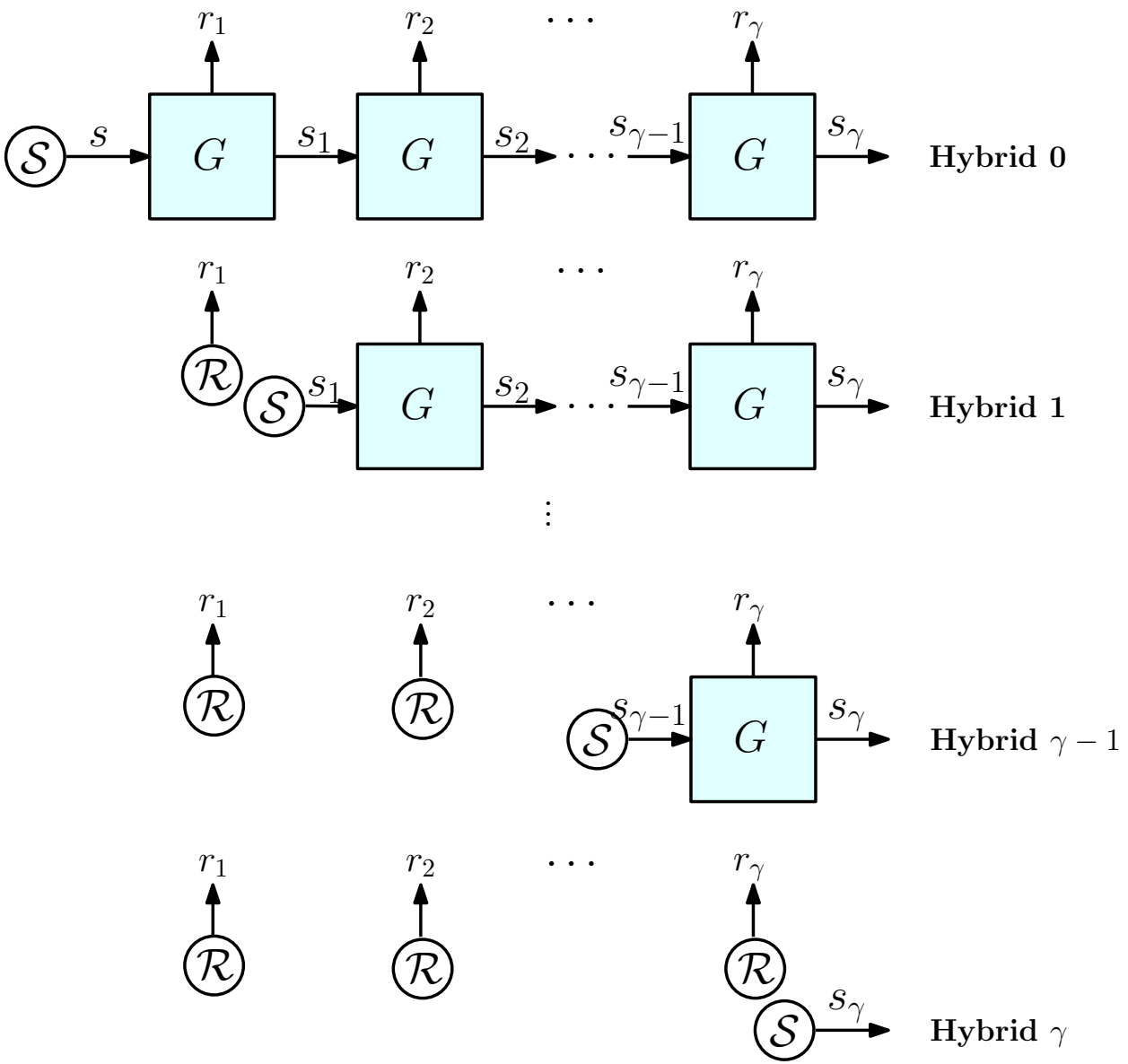
A Sequential Construction



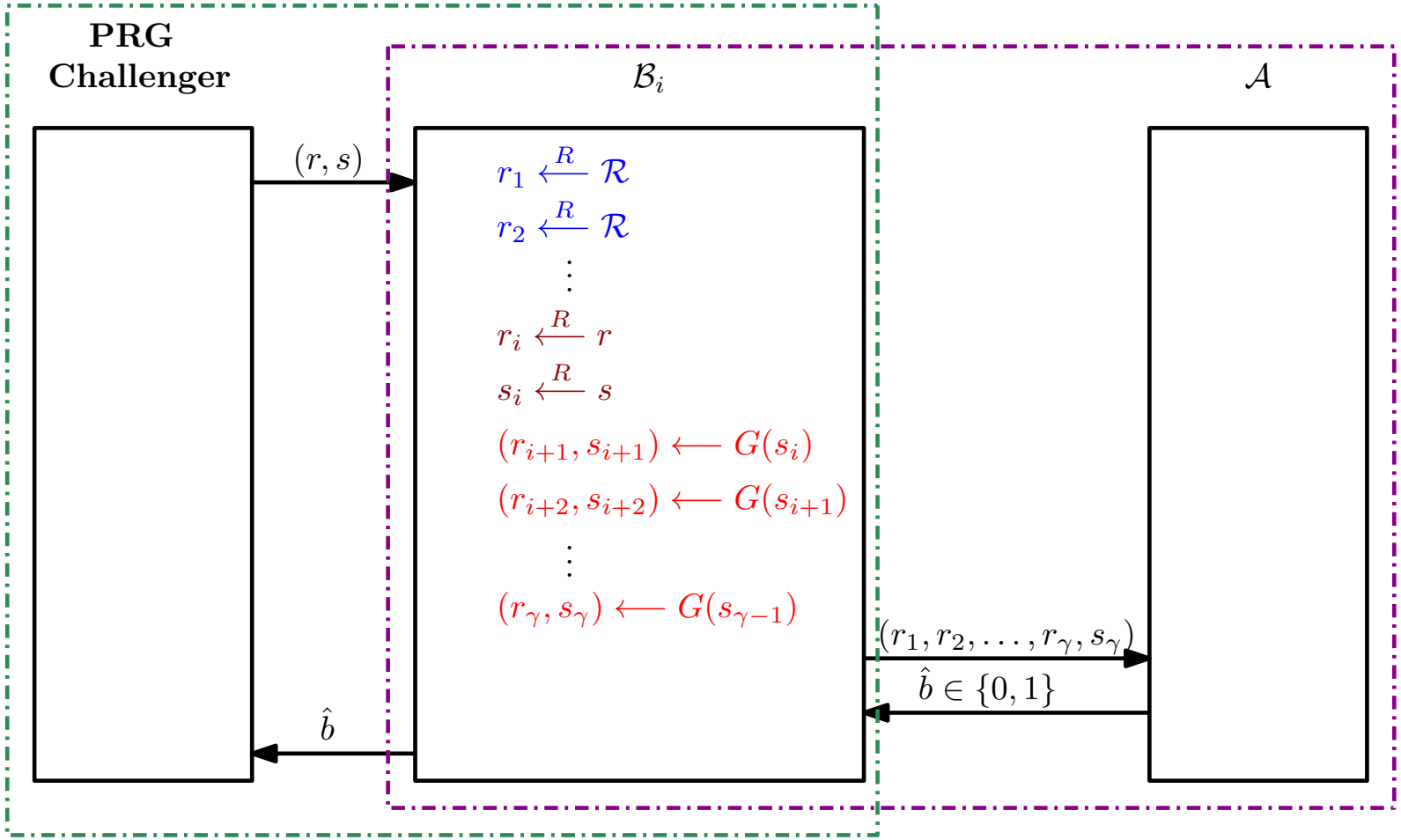
A Sequential Construction



A Sequential Construction



A Sequential Construction



Experiment b

A Sequential Construction

For polynomially bounded γ:

- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\gamma$.

A Sequential Construction

For polynomially bounded γ:

- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\gamma$.
  - $\mathcal{B}$ computes $\omega \xleftarrow{R} \{1, 2, \ldots, \gamma\}$.

A Sequential Construction

For polynomially bounded γ:

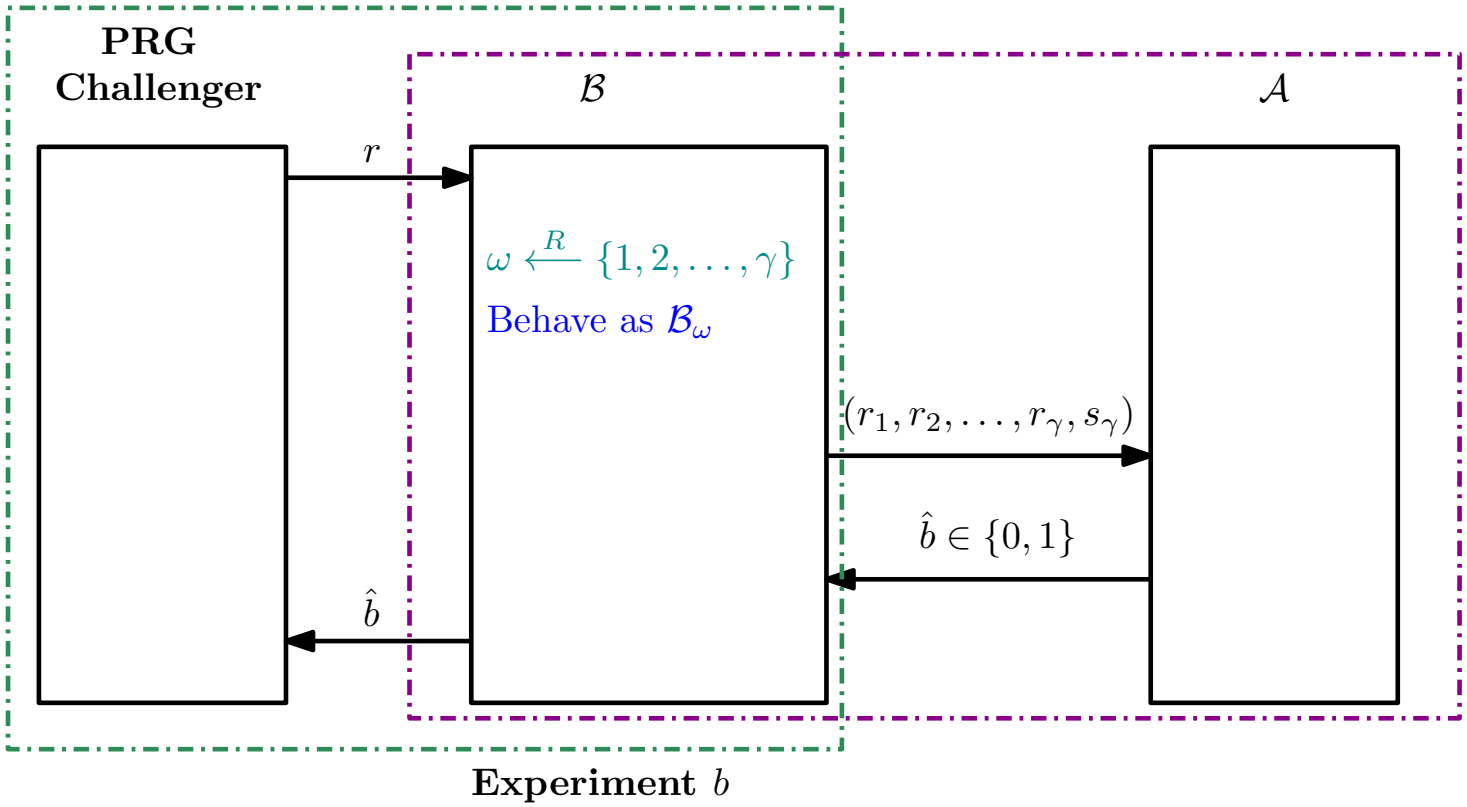
- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\gamma$.
  - $\mathcal{B}$ computes $\omega \xleftarrow{R} \{1, 2, \ldots, \gamma\}$.
  - Follow the behavior of $\mathcal{B}_\omega$.

A Sequential Construction

For polynomially bounded γ:

- Construction of Adversary $\mathcal{B}$:
  - Combine $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\gamma$.
  - $\mathcal{B}$ computes $\omega \xleftarrow{R} \{1, 2, \ldots, \gamma\}$.
  - Follow the behavior of $\mathcal{B}_\omega$.
- Let W_b be the event where $\mathcal{B}$ outputs 1 in Experiment b.

A Sequential Construction



A Sequential Construction

For polynomially bounded γ:

$$
\begin{aligned}
\Pr[W_0] &= \sum_{i=1}^{\gamma} \Pr[W_0 \wedge \omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_0 \mid \omega = i] \Pr[\omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_0 \mid \omega = i] \frac{1}{\gamma} \\
&= \frac{1}{\gamma} \sum_{i=1}^{\gamma} p_{i-1}
\end{aligned}
$$

A Sequential Construction

For polynomially bounded γ:

$$
\begin{aligned}
\Pr[W_0] &= \sum_{i=1}^{\gamma} \Pr[W_0 \wedge \omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_0 \mid \omega = i] \Pr[\omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_0 \mid \omega = i] \frac{1}{\gamma} \\
&= \frac{1}{\gamma} \sum_{i=1}^{\gamma} p_{i-1}
\end{aligned}
\qquad
\begin{aligned}
\Pr[W_1] &= \sum_{i=1}^{\gamma} \Pr[W_1 \wedge \omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_1 \mid \omega = i] \Pr[\omega = i] \\
&= \sum_{i=1}^{\gamma} \Pr[W_1 \mid \omega = i] \frac{1}{\gamma} \\
&= \frac{1}{\gamma} \sum_{i=1}^{\gamma} p_i
\end{aligned}
$$

A Sequential Construction

For polynomially bounded γ:

$$\mathsf{PRGadv}[\mathcal{B}, G] \quad = \quad |\Pr[W_1] - \Pr[W_0]|$$

A Sequential Construction

For polynomially bounded γ:

$$\begin{aligned} \mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_1] - \Pr[W_0]| \\ &= \left| \left(\frac{1}{\gamma}\sum_{i=1}^{\gamma} p_i\right) - \left(\frac{1}{\gamma}\sum_{i=1}^{\gamma} p_{i-1}\right) \right| \end{aligned}$$

A Sequential Construction

For polynomially bounded γ:

$$
\begin{aligned}
\mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_1] - \Pr[W_0]| \\
&= \left| \left(\frac{1}{\gamma} \sum_{i=1}^{\gamma} p_i \right) - \left(\frac{1}{\gamma} \sum_{i=1}^{\gamma} p_{i-1} \right) \right| \\
&= \frac{1}{\gamma} | \, p_\gamma - p_0 \, | \\
&= \frac{1}{\gamma} \cdot \mathsf{PRGadv}[\mathcal{A}, G'].
\end{aligned}
$$

A Sequential Construction

For polynomially bounded γ:

$$\begin{aligned}
\mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_1] - \Pr[W_0]| \\
&= \left| \left(\frac{1}{\gamma}\sum_{i=1}^{\gamma} p_i\right) - \left(\frac{1}{\gamma}\sum_{i=1}^{\gamma} p_{i-1}\right) \right| \\
&= \frac{1}{\gamma} | \, p_\gamma - p_0 \, | \\
&= \frac{1}{\gamma} \cdot \mathsf{PRGadv}[\mathcal{A}, G'].
\end{aligned}$$

Therefore,

$$\mathsf{PRGadv}[\mathcal{A}, G'] = \gamma \cdot \mathsf{PRGadv}[\mathcal{B}, G].$$

Expansion Rate

- A PRG that stretches an n-bit seed to an L-bit output has expansion rate of $\frac{L}{n}$.

Expansion Rate

- A PRG that stretches an n-bit seed to an L-bit output has expansion rate of $\frac{L}{n}$.
- If G is defined over $(\mathcal{S}, \mathcal{R})$, then expansion rate is $\frac{\log |\mathcal{R}|}{\log |\mathcal{S}|}$.

Expansion Rate

- A PRG that stretches an n-bit seed to an L-bit output has expansion rate of $\frac{L}{n}$.
- If G is defined over $(\mathcal{S}, \mathcal{R})$, then expansion rate is $\frac{\log |\mathcal{R}|}{\log |\mathcal{S}|}$.
- The parallel construction is better for parallelization compared to a sequential composition.

Expansion Rate

- A PRG that stretches an n-bit seed to an L-bit output has expansion rate of $\frac{L}{n}$.
- If G is defined over $(\mathcal{S}, \mathcal{R})$, then expansion rate is $\frac{\log |\mathcal{R}|}{\log |\mathcal{S}|}$.
- The parallel construction is better for parallelization compared to a sequential composition.
- The sequential construction has better expansion rate compared to a parallel composition.

Unpredictability

- Let G be a PRG defined over $(\{0,1\}^n, \{0,1\}^L)$.
- Let there be an adversary $\mathcal{A}$ that can predict the the last bit of the output of G given first $L-1$ bits.
- We can easily construct an adversary $\mathcal{B}$ that can win PRG indistinguishability game with sufficient advantage.

Unpredictability

- Let G be a PRG defined over $(\{0,1\}^n, \{0,1\}^L)$.
- Let there be an adversary $\mathcal{A}$ that can predict the the last bit of the output of G given first $L-1$ bits.
- We can easily construct an adversary $\mathcal{B}$ that can win PRG indistinguishability game with sufficient advantage.
- Therefore, there must not be any adversary that can predict the i-th bit of the output given first $i-1$ bits.

Unpredictability

- Let G be a PRG defined over $(\{0,1\}^n, \{0,1\}^L)$.
- Let there be an adversary $\mathcal{A}$ that can predict the the last bit of the output of G given first $L-1$ bits.
- We can easily construct an adversary $\mathcal{B}$ that can win PRG indistinguishability game with sufficient advantage.
- Therefore, there must not be any adversary that can predict the i-th bit of the output given first $i-1$ bits.
- This property of PRG is called Unpredictability.

Unpredictability

- Let G be a PRG defined over $(\{0,1\}^n, \{0,1\}^L)$.
- Let there be an adversary $\mathcal{A}$ that can predict the the last bit of the output of G given first $L-1$ bits.
- We can easily construct an adversary $\mathcal{B}$ that can win PRG indistinguishability game with sufficient advantage.
- Therefore, there must not be any adversary that can predict the i-th bit of the output given first $i-1$ bits.
- This property of PRG is called Unpredictability.
- Unpredictability and PRG security in terms of indistinguishability are equivalent.

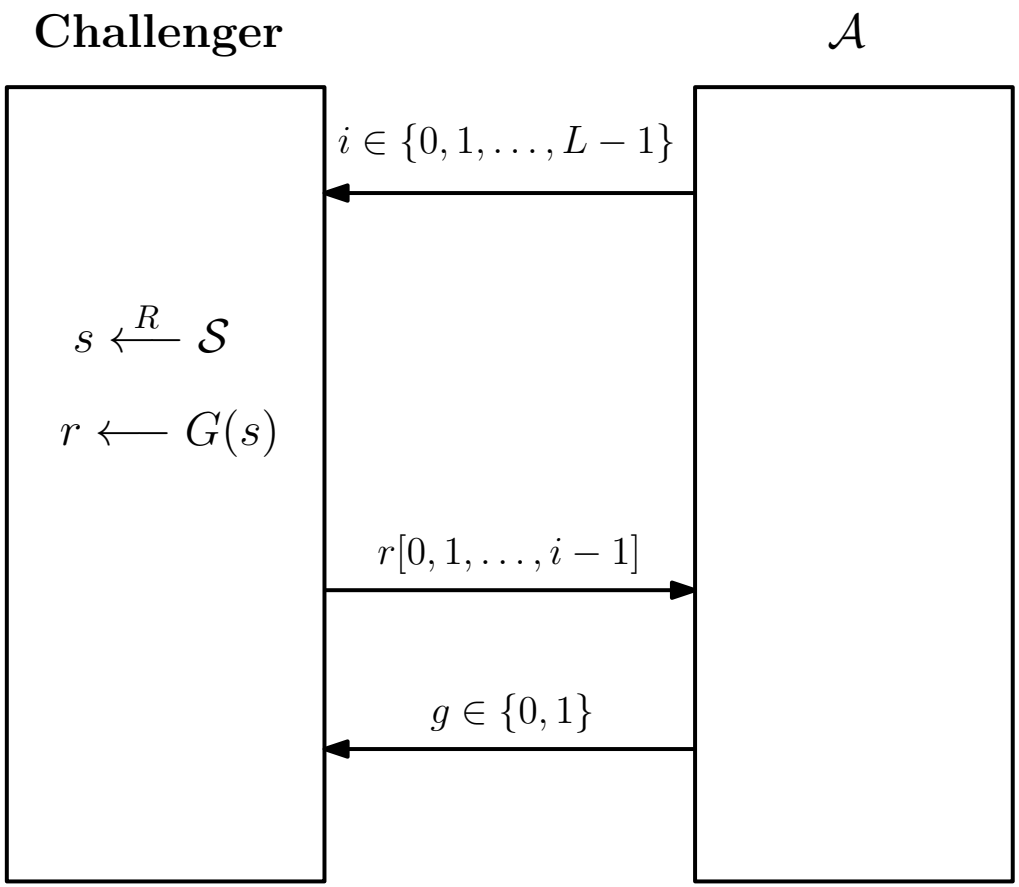
Unpredictability

- Let G be a PRG defined over $(\{0,1\}^n, \{0,1\}^L)$.
- Let there be an adversary $\mathcal{A}$ that can predict the the last bit of the output of G given first $L-1$ bits.
- We can easily construct an adversary $\mathcal{B}$ that can win PRG indistinguishability game with sufficient advantage.
- Therefore, there must not be any adversary that can predict the i-th bit of the output given first $i-1$ bits.
- This property of PRG is called Unpredictability.
- Unpredictability and PRG security in terms of indistinguishability are equivalent.

Unpredictability

Let G be a PRG defined over $(\{0,1\}^n, \{0,1\}^L)$. We say G is unpredictable if given the first i bits of G's output, it is hard to predict the next bit (that is, the $(i+1)$-st bit) with probability significantly better that 1/2 (here, i is an adversarially chosen index).

Unpredictable Attack Game



Unpredictable Attack Game

Unpredictability Advantage

Let W be be the event $\mathcal{A}$ wins if $r[i] = g$. We define the advantage of $\mathcal{A}$ in the attack game with respect to G as

$$\mathsf{PREDadv}[\mathcal{A}, G] = \left| \Pr[W] - \frac{1}{2} \right|.$$

Unpredictable Attack Game

Unpredictability Advantage

Let W be be the event $\mathcal{A}$ wins if $r[i] = g$. We define the advantage of $\mathcal{A}$ in the attack game with respect to G as

$$\mathsf{PREDadv}[\mathcal{A}, G] = \left| \Pr[W] - \frac{1}{2} \right|.$$

Unpredictable PRG

A PRG G is unpredictable if for all PPT adversaries (or distinguisher) $\mathcal{A}$ there exists a negligible function ϵ such that

$$\mathsf{PREDadv}[\mathcal{A}, G] \leqslant \epsilon(n).$$

Unpredictability

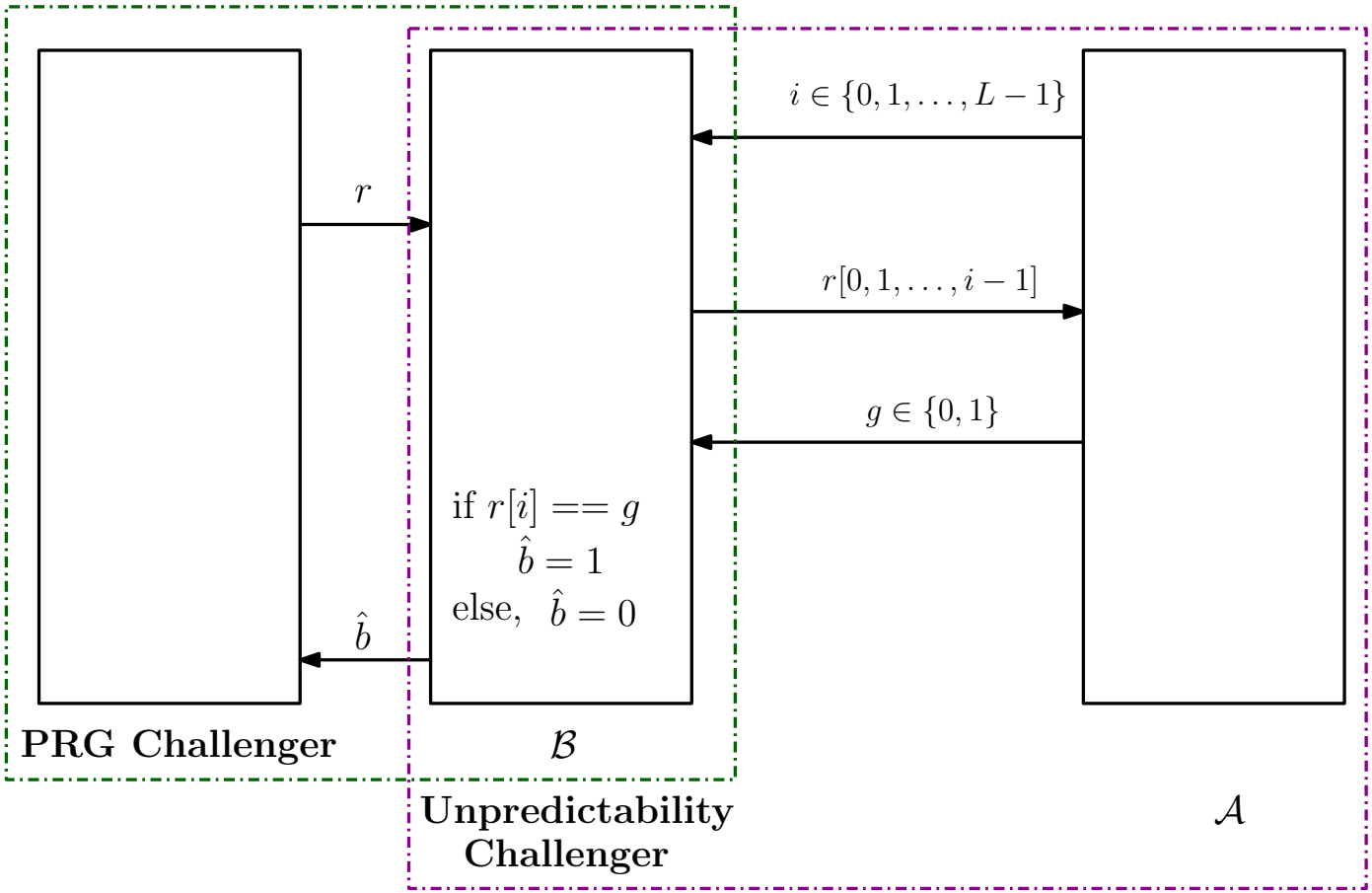
Theorem

Let G be a PRG, defined over $(\mathcal{S}, \mathcal{R})$ where $\mathcal{S} = \{0,1\}^n, \mathcal{R} = \{0,1\}^L$. If G is secure, then G is unpredictable.
In particular, for every adversary $\mathcal{A}$ breaking the unpredictability of G, as in Unpredictable Attack Game, there exists an adversary $\mathcal{B}$ breaking the security of G, as in PRG Indistinguishability Attack Game, where $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine, such that

$$\mathsf{PREDadv}[\mathcal{A}, G] = \mathsf{PRGadv}[\mathcal{B}, G].$$

Unpredictability

Proof



Experiment b

Unpredictability

Proof

- For $b = 0, 1$, let W_b be the event $\mathcal{B}$ outputs 1 in Experiment b. Then advantage of $\mathcal{B}$ against G in PRG indistinguishability game is

$$\mathsf{PRGadv}[\mathcal{B}, G] = |\Pr[W_0] - \Pr[W_1]| .$$

Unpredictability

Proof

- For $b = 0, 1$, let W_b be the event $\mathcal{B}$ outputs 1 in Experiment b. Then advantage of $\mathcal{B}$ against G in PRG indistinguishability game is

$$\mathsf{PRGadv}[\mathcal{B}, G] = |\Pr[W_0] - \Pr[W_1]| .$$

- Let $\mathcal{A}$ wins in the unpredictability game with probability p. Then advantage of $\mathcal{A}$ against G in PRG unpredictability game is

$$\mathsf{PREDadv}[\mathcal{A}, G] = \left| p - \frac{1}{2} \right| .$$

Unpredictability

Proof

- For $b = 0, 1$, let W_b be the event $\mathcal{B}$ outputs 1 in Experiment b. Then advantage of $\mathcal{B}$ against G in PRG indistinguishability game is

$$\mathsf{PRGadv}[\mathcal{B}, G] = |\Pr[W_0] - \Pr[W_1]| .$$

- Let $\mathcal{A}$ wins in the unpredictability game with probability p. Then advantage of $\mathcal{A}$ against G in PRG unpredictability game is

$$\mathsf{PREDadv}[\mathcal{A}, G] = \left| p - \frac{1}{2} \right| .$$

- W_0 occurs if and only if $r[i] = g$ in Experiment 0.

Unpredictability

Proof

- For $b = 0, 1$, let W_b be the event $\mathcal{B}$ outputs 1 in Experiment b. Then advantage of $\mathcal{B}$ against G in PRG indistinguishability game is

$$\mathsf{PRGadv}[\mathcal{B}, G] = |\Pr[W_0] - \Pr[W_1]| .$$

- Let $\mathcal{A}$ wins in the unpredictability game with probability p. Then advantage of $\mathcal{A}$ against G in PRG unpredictability game is

$$\mathsf{PREDadv}[\mathcal{A}, G] = \left| p - \frac{1}{2} \right| .$$

- W_0 occurs if and only if $r[i] = g$ in Experiment 0.

$$\begin{aligned} \Pr[W_0] &= \Pr[\mathcal{B} \text{ outputs 1 in Experiment 0}] \\ &= \Pr[\mathcal{A} \text{ wins in unpredictability game}] \\ &= p. \end{aligned}$$

Unpredictability

Proof

- In Experiment 1, r is a truly random bit string,
- W_1 occurs if and only if $r[i] = g$ in Experiment 1.

Unpredictability

Proof

- In Experiment 1, r is a truly random bit string,
- W_1 occurs if and only if $r[i] = g$ in Experiment 1.
- As random variables, the values of $\mathbf{r}[\mathbf{i}]$ and $\mathbf{g}$ are independent.

Unpredictability

Proof

- In Experiment 1, r is a truly random bit string,
- W_1 occurs if and only if $r[i] = g$ in Experiment 1.
- As random variables, the values of $\mathbf{r}[\mathbf{i}]$ and $\mathbf{g}$ are independent.

$$\Pr[W_1] = \Pr[\mathcal{B} \text{ outputs 1 in Experiment 1}] = \frac{1}{2}.$$

Unpredictability

Proof

- In Experiment 1, r is a truly random bit string,
- W_1 occurs if and only if $r[i] = g$ in Experiment 1.
- As random variables, the values of $\mathbf{r}[\mathbf{i}]$ and $\mathbf{g}$ are independent.

$$\Pr[W_1] = \Pr[\mathcal{B} \text{ outputs 1 in Experiment 1}] = \frac{1}{2}.$$

$$\mathsf{PRGadv}[\mathcal{B}, G] \quad = \quad |\Pr[W_0] - \Pr[W_1]|$$

Unpredictability

Proof

- In Experiment 1, r is a truly random bit string,
- W_1 occurs if and only if $r[i] = g$ in Experiment 1.
- As random variables, the values of $\mathbf{r}[\mathbf{i}]$ and $\mathbf{g}$ are independent.

$$\Pr[W_1] = \Pr[\mathcal{B} \text{ outputs 1 in Experiment 1}] = \frac{1}{2}.$$

$$\begin{aligned} \mathsf{PRGadv}[\mathcal{B}, G] &= |\Pr[W_0] - \Pr[W_1]| \\ &= \left| p - \frac{1}{2} \right| \\ &= \mathsf{PREDadv}[\mathcal{A}, G] \end{aligned}$$

Unpredictability

Theorem

Let G be a PRG, defined over $(\mathcal{S}, \mathcal{R})$ where $\mathcal{S} = \{0,1\}^n, \mathcal{R} = \{0,1\}^L$. If G is unpredictable, then G is secure.
In particular, for every adversary $\mathcal{A}$ breaking the security of G as in PRG Indistinguishability Attack Game, there exists an adversary $\mathcal{B}$, breaking the unpredictability of G as in Unpredictable Attack Game, where $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine, such that

$$\mathsf{PRGadv}[\mathcal{A}, G] = L \cdot \mathsf{PREDadv}[\mathcal{B}, G].$$

Unpredictability

Proof Sketch

- We prove using a hybrid argument.
- Let $\mathcal{A}$ be an efficient adversary that can effectively distinguish a pseudo-random L-bit string from a random L-bit string.

Unpredictability

Proof Sketch

- We prove using a hybrid argument.
- Let $\mathcal{A}$ be an efficient adversary that can effectively distinguish a pseudo-random L-bit string from a random L-bit string.
- We construct an efficient adversary $\mathcal{B}$ that can effectively distinguish

$$x_0 \cdots x_{i-1} x_i$$

from

$$x_0 \cdots x_{i-1} r$$

Unpredictability

Proof Sketch

- We prove using a hybrid argument.
- Let $\mathcal{A}$ be an efficient adversary that can effectively distinguish a pseudo-random L-bit string from a random L-bit string.
- We construct an efficient adversary $\mathcal{B}$ that can effectively distinguish

$$x_0 \cdots x_{i-1} x_i$$

from

$$x_0 \cdots x_{i-1} r$$

  - i is a randomly chosen index,

Unpredictability

Proof Sketch

- We prove using a hybrid argument.
- Let $\mathcal{A}$ be an efficient adversary that can effectively distinguish a pseudo-random L-bit string from a random L-bit string.
- We construct an efficient adversary $\mathcal{B}$ that can effectively distinguish

$$x_0 \cdots x_{i-1} x_i$$

from

$$x_0 \cdots x_{i-1} r$$

  - i is a randomly chosen index,
  - $x_0, x_1, \ldots, x_{L-1}$ is the pseudo-random output, and

Unpredictability

Proof Sketch

- We prove using a hybrid argument.
- Let $\mathcal{A}$ be an efficient adversary that can effectively distinguish a pseudo-random L-bit string from a random L-bit string.
- We construct an efficient adversary $\mathcal{B}$ that can effectively distinguish

$$x_0 \cdots x_{i-1} x_i$$

from

$$x_0 \cdots x_{i-1} r$$

  - i is a randomly chosen index,
  - $x_0, x_1, \ldots, x_{L-1}$ is the pseudo-random output, and
  - r is a random bit.

Unpredictability

Proof Sketch

- We prove using a hybrid argument.
- Let $\mathcal{A}$ be an efficient adversary that can effectively distinguish a pseudo-random L-bit string from a random L-bit string.
- We construct an efficient adversary $\mathcal{B}$ that can effectively distinguish

$$x_0 \cdots x_{i-1} x_i$$

from

$$x_0 \cdots x_{i-1} r$$

  - i is a randomly chosen index,
  - $x_0, x_1, \ldots, x_{L-1}$ is the pseudo-random output, and
  - r is a random bit.
- $\mathcal{B}$ can distinguish the pseudo-random bit x_i from the random bit r, given the side information $x_0 \cdots x_{i-1}$.

Unpredictability

Proof Sketch

- Turn $\mathcal{B}$'s distinguishing advantage into a predicting advantage.

Unpredictability

Proof Sketch

- Turn $\mathcal{B}$'s distinguishing advantage into a predicting advantage.
- Given $x_0 \cdots x_{i-1}$:

Unpredictability

Proof Sketch

- Turn $\mathcal{B}$'s distinguishing advantage into a predicting advantage.
- Given $x_0 \cdots x_{i-1}$:
  - Feed $\mathcal{B}$ the string $x_0 \cdots x_{i-1} r$.

Unpredictability

Proof Sketch

- Turn $\mathcal{B}$'s distinguishing advantage into a predicting advantage.
- Given $x_0 \cdots x_{i-1}$:
  - Feed $\mathcal{B}$ the string $x_0 \cdots x_{i-1} r$.
  - If $\mathcal{B}$ outputs 1, predictor outputs r,

Unpredictability

Proof Sketch

- Turn $\mathcal{B}$'s distinguishing advantage into a predicting advantage.
- Given $x_0 \cdots x_{i-1}$:
  - Feed $\mathcal{B}$ the string $x_0 \cdots x_{i-1} r$.
  - If $\mathcal{B}$ outputs 1, predictor outputs r,
  - Otherwise, predictor outputs $\bar{r}$.

Unpredictability

Proof Sketch

- A random variable $\mathbf{x}$ for side information $x_0 \cdots x_{i-1}$ and any random coins used by the adversary $\mathcal{B}$ (if any).

Unpredictability

Proof Sketch

- A random variable **x** for side information $x_0 \cdots x_{i-1}$ and any random coins used by the adversary $\mathcal{B}$ (if any).
- A 0/1-valued random variable **b**, which corresponds to x_i, and which may be correlated with **x**.

Unpredictability

Proof Sketch

- A random variable $\mathbf{x}$ for side information $x_0 \cdots x_{i-1}$ and any random coins used by the adversary $\mathcal{B}$ (if any).
- A 0/1-valued random variable $\mathbf{b}$, which corresponds to x_i, and which may be correlated with $\mathbf{x}$.
- A 0/1-valued random variable $\mathbf{r}$, which corresponds to r.
  - Independent of $(\mathbf{x}, \mathbf{b})$.

Unpredictability

Proof Sketch

- A random variable $\mathbf{x}$ for side information $x_0 \cdots x_{i-1}$ and any random coins used by the adversary $\mathcal{B}$ (if any).
- A 0/1-valued random variable $\mathbf{b}$, which corresponds to x_i, and which may be correlated with $\mathbf{x}$.
- A 0/1-valued random variable $\mathbf{r}$, which corresponds to r.
  - Independent of $(\mathbf{x}, \mathbf{b})$.
- A function d, which corresponds to $\mathcal{B}$'s strategy.

Unpredictability

Proof Sketch

- A random variable $\mathbf{x}$ for side information $x_0 \cdots x_{i-1}$ and any random coins used by the adversary $\mathcal{B}$ (if any).
- A 0/1-valued random variable $\mathbf{b}$, which corresponds to x_i, and which may be correlated with $\mathbf{x}$.
- A 0/1-valued random variable $\mathbf{r}$, which corresponds to r.
  - Independent of $(\mathbf{x}, \mathbf{b})$.
- A function d, which corresponds to $\mathcal{B}$'s strategy.
- $\mathcal{B}$'s distinguishing advantage be $|\ \epsilon\ |$, where

$$\epsilon = \Pr[d(\mathbf{x}, \mathbf{b}) = 1] - \Pr[d(\mathbf{x}, \mathbf{r}) = 1].$$

Unpredictability

Proof Sketch

- A random variable $\mathbf{x}$ for side information $x_0 \cdots x_{i-1}$ and any random coins used by the adversary $\mathcal{B}$ (if any).
- A 0/1-valued random variable $\mathbf{b}$, which corresponds to x_i, and which may be correlated with $\mathbf{x}$.
- A 0/1-valued random variable $\mathbf{r}$, which corresponds to r.
  - Independent of $(\mathbf{x}, \mathbf{b})$.
- A function d, which corresponds to $\mathcal{B}$'s strategy.
- $\mathcal{B}$'s distinguishing advantage be $|\ \epsilon\ |$, where

$$\epsilon = \Pr[d(\mathbf{x}, \mathbf{b}) = 1] - \Pr[d(\mathbf{x}, \mathbf{r}) = 1].$$

- A random variable $\mathbf{b}'$ that corresponds to prediction of the mentioned strategy.

Unpredictability

Proof Sketch

- A random variable $\mathbf{x}$ for side information $x_0 \cdots x_{i-1}$ and any random coins used by the adversary $\mathcal{B}$ (if any).
- A 0/1-valued random variable $\mathbf{b}$, which corresponds to x_i, and which may be correlated with $\mathbf{x}$.
- A 0/1-valued random variable $\mathbf{r}$, which corresponds to r.
  - Independent of $(\mathbf{x}, \mathbf{b})$.
- A function d, which corresponds to $\mathcal{B}$'s strategy.
- $\mathcal{B}$'s distinguishing advantage be $|\ \epsilon\ |$, where

$$\epsilon = \Pr[d(\mathbf{x}, \mathbf{b}) = 1] - \Pr[d(\mathbf{x}, \mathbf{r}) = 1].$$

- A random variable $\mathbf{b}'$ that corresponds to prediction of the mentioned strategy.
  - $\mathbf{b}' = \mathbf{r}$, if $d(\mathbf{x}, \mathbf{r}) = 1$,
  - $\mathbf{b}' = \bar{\mathbf{r}}$, otherwise.

Unpredictability

Proof Sketch

- A random variable $\mathbf{x}$ for side information $x_0 \cdots x_{i-1}$ and any random coins used by the adversary $\mathcal{B}$ (if any).
- A 0/1-valued random variable $\mathbf{b}$, which corresponds to x_i, and which may be correlated with $\mathbf{x}$.
- A 0/1-valued random variable $\mathbf{r}$, which corresponds to r.
  - Independent of $(\mathbf{x}, \mathbf{b})$.
- A function d, which corresponds to $\mathcal{B}$'s strategy.
- $\mathcal{B}$'s distinguishing advantage be $|\ \epsilon\ |$, where

$$\epsilon = \Pr[d(\mathbf{x}, \mathbf{b}) = 1] - \Pr[d(\mathbf{x}, \mathbf{r}) = 1].$$

- A random variable $\mathbf{b}'$ that corresponds to prediction of the mentioned strategy.
  - $\mathbf{b}' = \mathbf{r}$, if $d(\mathbf{x}, \mathbf{r}) = 1$,
  - $\mathbf{b}' = \bar{\mathbf{r}}$, otherwise.
- $\Pr[\mathbf{b}' = \mathbf{b}] = \frac{1}{2} + \epsilon$.

Unpredictability

Distinguisher/predictor Lemma

Let $\mathbf{x}$ be a random variable taking values in some set S, and let $\mathbf{b}$ and $\mathbf{r}$ be a 0/1-valued random variables, where $\mathbf{r}$ is uniformly distributed over $\{0,1\}$ and is independent of $(\mathbf{x},\mathbf{b})$. Let $d : S \times \{0,1\} \longrightarrow \{0,1\}$ be an arbitrary function, and let

$$\epsilon := \Pr[d(\mathbf{x},\mathbf{b}) = 1] - \Pr[d(\mathbf{x},\mathbf{r}) = 1].$$

Define the random variable $\mathbf{b}'$ as follows:

$$\mathbf{b}' := \begin{cases} \mathbf{r}, & \text{if } d(\mathbf{x},\mathbf{r}) = 1 \\ \bar{\mathbf{r}}, & \text{otherwise.} \end{cases}$$

Then

$$\Pr[\mathbf{b}' = \mathbf{b}] = \frac{1}{2} + \epsilon.$$

Unpredictability

Proof of Distinguisher/predictor Lemma

$$\Pr[\mathbf{b}' = \mathbf{b}] \quad = \quad \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \mathbf{r}]\Pr[\mathbf{b} = \mathbf{r}] + \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \bar{\mathbf{r}}]\Pr[\mathbf{b} = \bar{\mathbf{r}}]$$

Proof of Distinguisher/predictor Lemma

$$\begin{aligned}
\Pr[\mathbf{b}' = \mathbf{b}] &= \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \mathbf{r}]\Pr[\mathbf{b} = \mathbf{r}] + \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \bar{\mathbf{r}}]\Pr[\mathbf{b} = \bar{\mathbf{r}}] \\
&= \Pr[d(\mathbf{x}, \mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}]\frac{1}{2} + \Pr[d(\mathbf{x}, \mathbf{r}) = 0 \mid \mathbf{b} = \bar{\mathbf{r}}]\frac{1}{2}
\end{aligned}$$

Proof of Distinguisher/predictor Lemma

$$
\begin{aligned}
\Pr[\mathbf{b}' = \mathbf{b}] &= \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \mathbf{r}]\Pr[\mathbf{b} = \mathbf{r}] + \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \bar{\mathbf{r}}]\Pr[\mathbf{b} = \bar{\mathbf{r}}] \\
&= \Pr[d(\mathbf{x}, \mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}]\frac{1}{2} + \Pr[d(\mathbf{x}, \mathbf{r}) = 0 \mid \mathbf{b} = \bar{\mathbf{r}}]\frac{1}{2} \\
&= \frac{1}{2}\left(\Pr[d(\mathbf{x}, \mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}] + (1 - \Pr[d(\mathbf{x}, \mathbf{r}) = 1 \mid \mathbf{b} = \bar{\mathbf{r}}])\right)
\end{aligned}
$$

Proof of Distinguisher/predictor Lemma

$$
\begin{aligned}
\Pr[\mathbf{b}' = \mathbf{b}] &= \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \mathbf{r}]\Pr[\mathbf{b} = \mathbf{r}] + \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \bar{\mathbf{r}}]\Pr[\mathbf{b} = \bar{\mathbf{r}}] \\
&= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}]\frac{1}{2} + \Pr[d(\mathbf{x},\mathbf{r}) = 0 \mid \mathbf{b} = \bar{\mathbf{r}}]\frac{1}{2} \\
&= \frac{1}{2}\left(\Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}] + (1 - \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \bar{\mathbf{r}}])\right) \\
&= \frac{1}{2} + \frac{1}{2}(\alpha - \beta), \text{ where}
\end{aligned}
$$

$$
\begin{aligned}
\alpha &= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}], \\
\beta &= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \bar{\mathbf{r}}].
\end{aligned}
$$

Proof of Distinguisher/predictor Lemma

$$\begin{aligned}
\Pr[\mathbf{b}' = \mathbf{b}] &= \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \mathbf{r}]\Pr[\mathbf{b} = \mathbf{r}] + \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \bar{\mathbf{r}}]\Pr[\mathbf{b} = \bar{\mathbf{r}}] \\
&= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}]\frac{1}{2} + \Pr[d(\mathbf{x},\mathbf{r}) = 0 \mid \mathbf{b} = \bar{\mathbf{r}}]\frac{1}{2} \\
&= \frac{1}{2}\left(\Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}] + (1 - \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \bar{\mathbf{r}}])\right) \\
&= \frac{1}{2} + \frac{1}{2}(\alpha - \beta), \text{ where}
\end{aligned}$$

$$\begin{aligned}
\alpha &= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}], \\
\beta &= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \bar{\mathbf{r}}].
\end{aligned}$$

By independence,

$$\alpha = \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}]$$

Proof of Distinguisher/predictor Lemma

$$\begin{aligned}
\Pr[\mathbf{b}' = \mathbf{b}] &= \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \mathbf{r}]\Pr[\mathbf{b} = \mathbf{r}] + \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \bar{\mathbf{r}}]\Pr[\mathbf{b} = \bar{\mathbf{r}}] \\
&= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}]\frac{1}{2} + \Pr[d(\mathbf{x},\mathbf{r}) = 0 \mid \mathbf{b} = \bar{\mathbf{r}}]\frac{1}{2} \\
&= \frac{1}{2}\left(\Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}] + (1 - \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \bar{\mathbf{r}}])\right) \\
&= \frac{1}{2} + \frac{1}{2}(\alpha - \beta), \text{ where}
\end{aligned}$$

$$\begin{aligned}
\alpha &= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}], \\
\beta &= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \bar{\mathbf{r}}].
\end{aligned}$$

By independence,

$$\alpha = \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}] = \Pr[d(\mathbf{x},\mathbf{b}) = 1 \mid \mathbf{b} = \mathbf{r}]$$

Unpredictability

Proof of Distinguisher/predictor Lemma

$$
\begin{aligned}
\Pr[\mathbf{b}' = \mathbf{b}] &= \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \mathbf{r}]\Pr[\mathbf{b} = \mathbf{r}] + \Pr[\mathbf{b}' = \mathbf{b} \mid \mathbf{b} = \bar{\mathbf{r}}]\Pr[\mathbf{b} = \bar{\mathbf{r}}] \\
&= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}]\frac{1}{2} + \Pr[d(\mathbf{x},\mathbf{r}) = 0 \mid \mathbf{b} = \bar{\mathbf{r}}]\frac{1}{2} \\
&= \frac{1}{2}\left(\Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}] + (1 - \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \bar{\mathbf{r}}])\right) \\
&= \frac{1}{2} + \frac{1}{2}(\alpha - \beta), \text{ where}
\end{aligned}
$$

$$
\begin{aligned}
\alpha &= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}], \\
\beta &= \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \bar{\mathbf{r}}].
\end{aligned}
$$

By independence,

$$\alpha = \Pr[d(\mathbf{x},\mathbf{r}) = 1 \mid \mathbf{b} = \mathbf{r}] = \Pr[d(\mathbf{x},\mathbf{b}) = 1 \mid \mathbf{b} = \mathbf{r}] = \Pr[d(\mathbf{x},\mathbf{b}) = 1].$$

Unpredictability

Proof of Distinguisher/predictor Lemma

$$\epsilon \quad = \quad \Pr[d(\mathbf{x}, \mathbf{b}) = 1] - \Pr[d(\mathbf{x}, \mathbf{r}) = 1]$$

Unpredictability

Proof of Distinguisher/predictor Lemma

$$\begin{aligned}\epsilon &= \Pr[d(\mathbf{x},\mathbf{b})=1]-\Pr[d(\mathbf{x},\mathbf{r})=1] \\ &= \alpha-(\Pr[d(\mathbf{x},\mathbf{r})=1 \mid \mathbf{b}=\mathbf{r}]\Pr[\mathbf{b}=\mathbf{r}]+\Pr[d(\mathbf{x},\mathbf{r})=1 \mid \mathbf{b}=\bar{\mathbf{r}}]\Pr[\mathbf{b}=\bar{\mathbf{r}}])\end{aligned}$$

Unpredictability

Proof of Distinguisher/predictor Lemma

$$
\begin{aligned}
\epsilon &= \Pr[d(\mathbf{x},\mathbf{b})=1] - \Pr[d(\mathbf{x},\mathbf{r})=1] \\
&= \alpha - (\Pr[d(\mathbf{x},\mathbf{r})=1 \mid \mathbf{b}=\mathbf{r}]\Pr[\mathbf{b}=\mathbf{r}] + \Pr[d(\mathbf{x},\mathbf{r})=1 \mid \mathbf{b}=\bar{\mathbf{r}}]\Pr[\mathbf{b}=\bar{\mathbf{r}}]) \\
&= \alpha - \frac{1}{2}(\alpha+\beta)
\end{aligned}
$$

Unpredictability

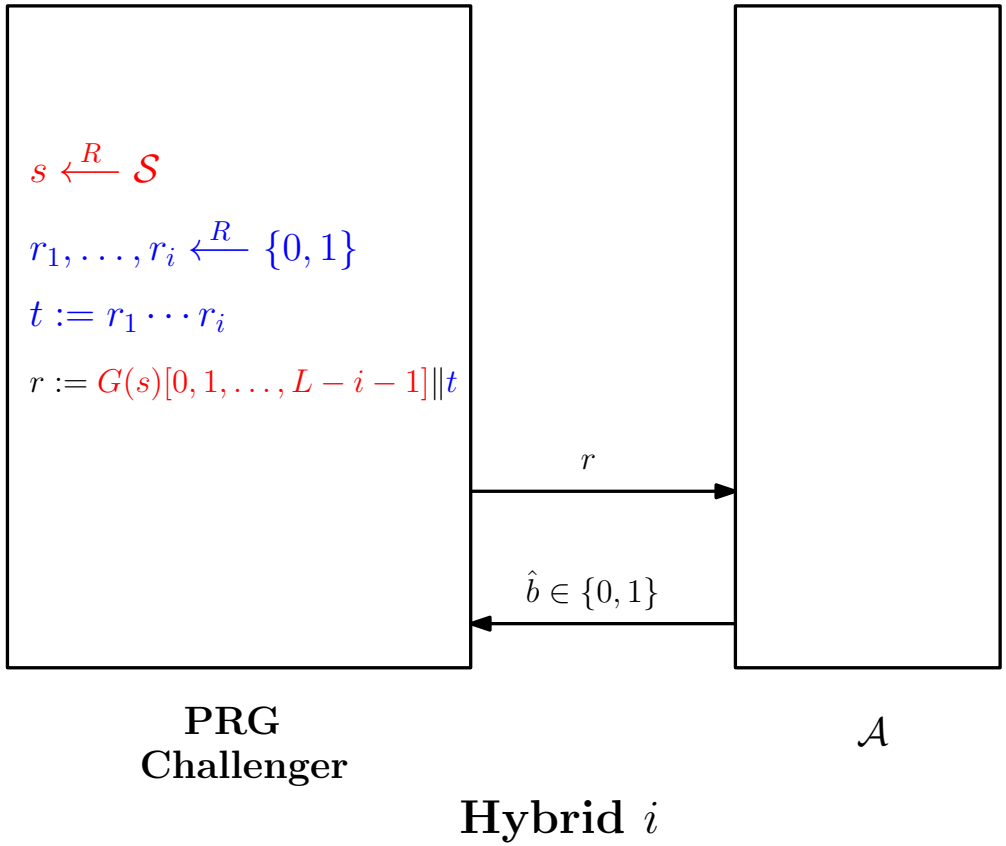
Proof of Distinguisher/predictor Lemma

$$
\begin{aligned}
\epsilon &= \Pr[d(\mathbf{x},\mathbf{b})=1] - \Pr[d(\mathbf{x},\mathbf{r})=1] \\
&= \alpha - (\Pr[d(\mathbf{x},\mathbf{r})=1 \mid \mathbf{b}=\mathbf{r}]\Pr[\mathbf{b}=\mathbf{r}] + \Pr[d(\mathbf{x},\mathbf{r})=1 \mid \mathbf{b}=\bar{\mathbf{r}}]\Pr[\mathbf{b}=\bar{\mathbf{r}}]) \\
&= \alpha - \frac{1}{2}(\alpha+\beta) \\
&= \frac{1}{2}(\alpha-\beta).
\end{aligned}
$$

Hence proved.

Unpredictability

Proof



Unpredictability

Proof

- In the attack game Hybrid i:

Unpredictability

Proof

- In the attack game Hybrid i:
  - Let the probability $\mathcal{A}$ outputs 1 be p_i.

Unpredictability

Proof

- In the attack game Hybrid i:
  - Let the probability $\mathcal{A}$ outputs 1 be p_i.
- Hybrid 0 represents the Experiment 0 of PRG indistinguishability attack game.

Unpredictability

Proof

- In the attack game Hybrid i:
  - Let the probability $\mathcal{A}$ outputs 1 be p_i.
- Hybrid 0 represents the Experiment 0 of PRG indistinguishability attack game.
- Hybrid L represents the Experiment 1 of PRG indistinguishability attack game.

Unpredictability

Proof

- In the attack game Hybrid i:
  - Let the probability $\mathcal{A}$ outputs 1 be p_i.
- Hybrid 0 represents the Experiment 0 of PRG indistinguishability attack game.
- Hybrid L represents the Experiment 1 of PRG indistinguishability attack game.
- Then the advantage of $\mathcal{A}$ against G is

$$\mathsf{PRGadv}[\mathcal{A}, G] = | p_0 - p_L | .$$

Unpredictability

Proof

- Construction of $\mathcal{B}$:
  - Choose $\omega \in \{1, \ldots, L\}$ at random.

Unpredictability

Proof

- Construction of $\mathcal{B}$:
  - Choose $\omega \in \{1, \ldots, L\}$ at random.
  - Send $L - \omega$ to the challenger, obtaining a string $x \in \{0,1\}^{L-\omega}$.

Unpredictability

Proof

- Construction of $\mathcal{B}$:
  - Choose $\omega \in \{1, \ldots, L\}$ at random.
  - Send $L - \omega$ to the challenger, obtaining a string $x \in \{0,1\}^{L-\omega}$.
  - Generate ω random bits $r_1, \ldots, r_\omega$, and give the L-bit string $x \| r_1 \cdots r_\omega$ to $\mathcal{A}$.

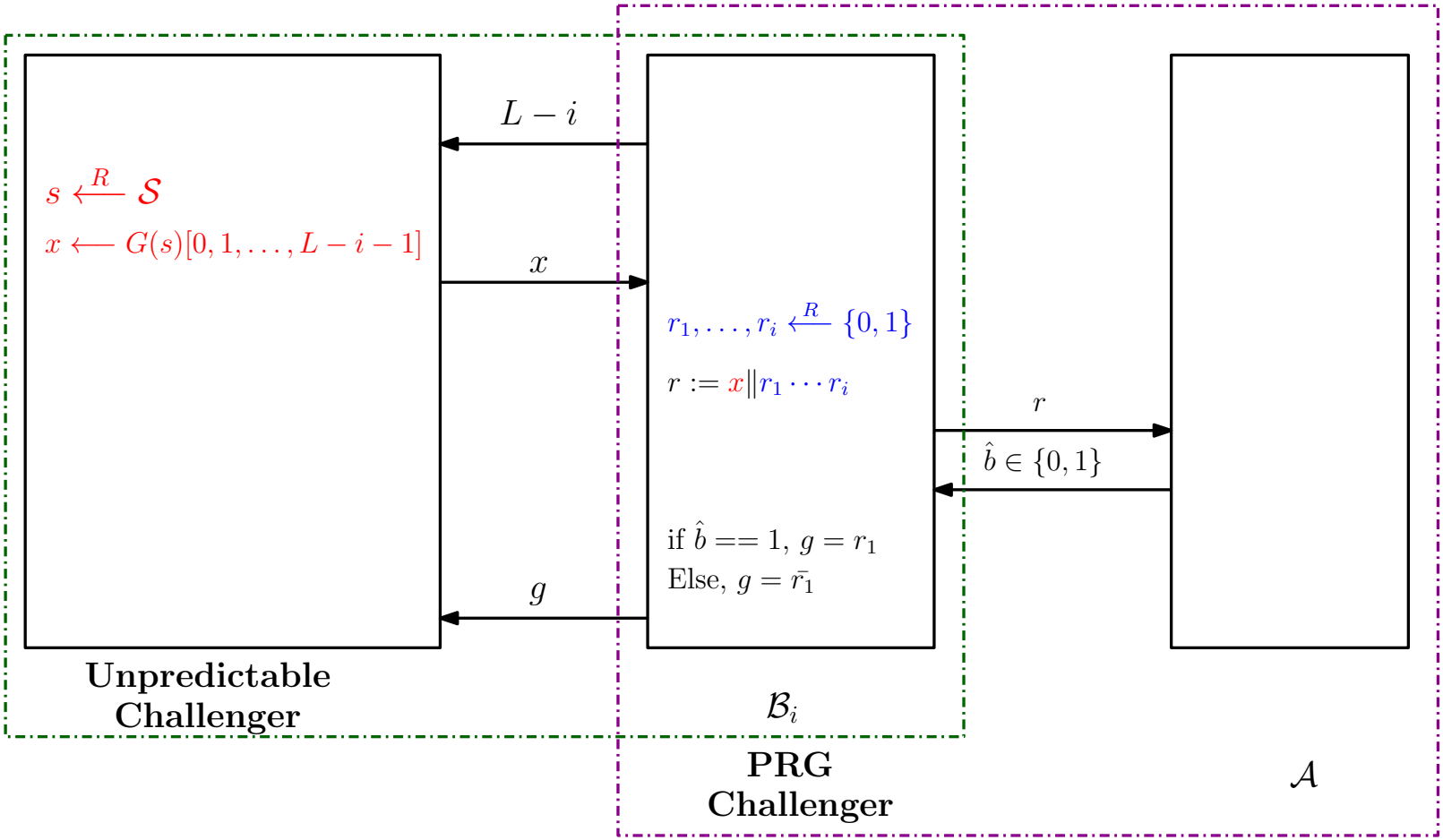
Unpredictability

Proof

- Construction of $\mathcal{B}$:
  - Choose $\omega \in \{1, \ldots, L\}$ at random.
  - Send $L - \omega$ to the challenger, obtaining a string $x \in \{0,1\}^{L-\omega}$.
  - Generate ω random bits $r_1, \ldots, r_\omega$, and give the L-bit string $x \| r_1 \cdots r_\omega$ to $\mathcal{A}$.
  - If $\mathcal{A}$ outputs 1, then $\mathcal{B}$ output r_1; otherwise, $\mathcal{B}$ output $\bar{r_1}$.

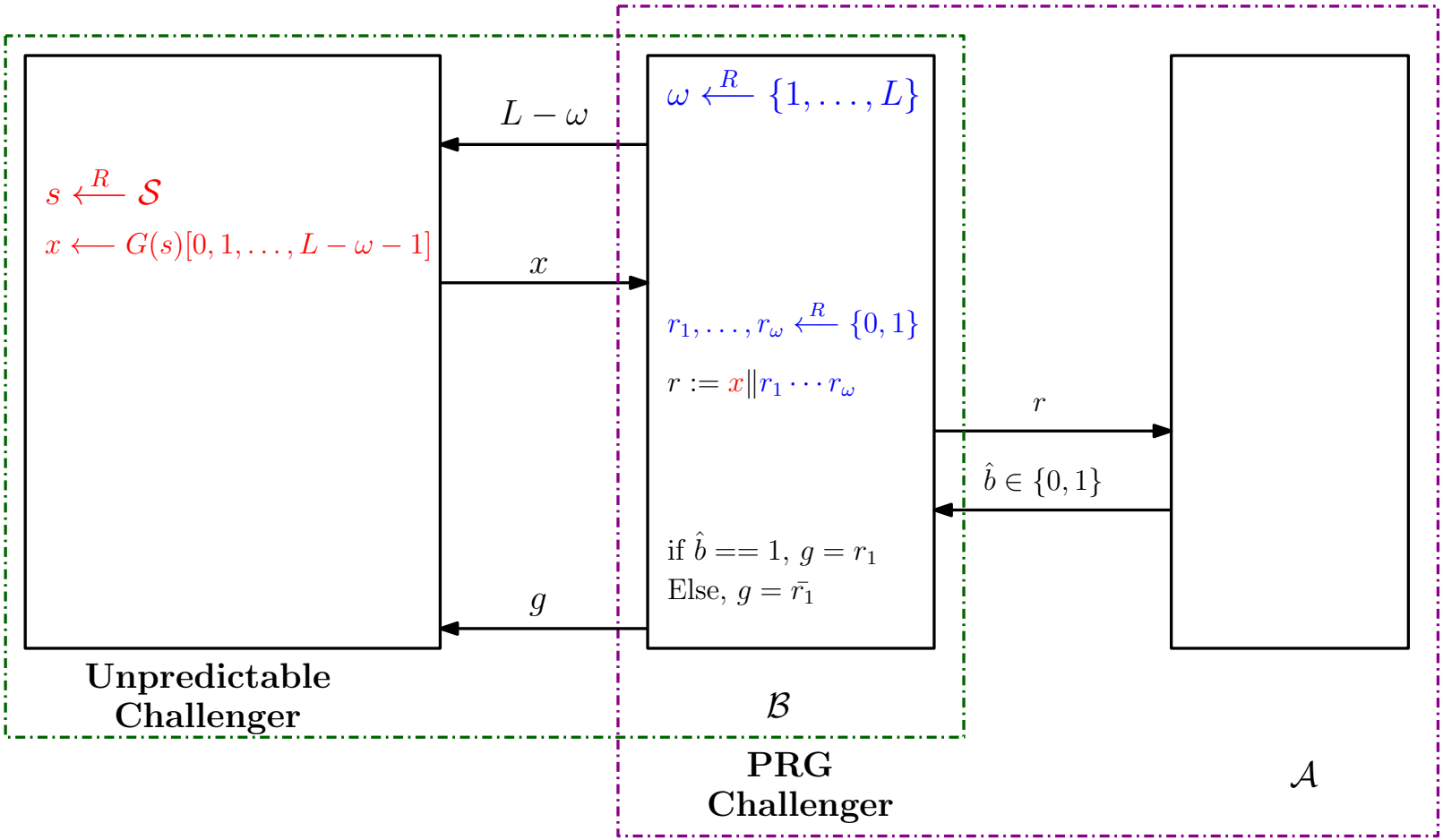
Unpredictability

Proof



Unpredictability

Proof



Unpredictability

Proof

- Let W be the event where $\mathcal{B}$ wins that is $g = G(s)[L - \omega]$.

$$\Pr[W] = \sum_{i=1}^{L} \Pr[W \wedge \omega = i]$$

Unpredictability

Proof

- Let W be the event where $\mathcal{B}$ wins that is $g = G(s)[L - \omega]$.

$$\begin{aligned} \Pr[W] &= \sum_{i=1}^{L} \Pr[W \wedge \omega = i] \\ &= \sum_{i=1}^{L} \Pr[W \mid \omega = i] \Pr[\omega = i] \end{aligned}$$

Unpredictability

Proof

- Let W be the event where $\mathcal{B}$ wins that is $g = G(s)[L - \omega]$.

$$\begin{aligned}
\Pr[W] &= \sum_{i=1}^{L} \Pr[W \wedge \omega = i] \\
&= \sum_{i=1}^{L} \Pr[W \mid \omega = i] \Pr[\omega = i] \\
&= \frac{1}{L} \sum_{i=1}^{L} \Pr[W \mid \omega = i]
\end{aligned}$$

Unpredictability

Proof

- **By Distinguisher/predictor Lemma:** $\Pr[W \mid \omega = i] = \left(\frac{1}{2} + p_{i-1} - p_i\right)$.

$$\Pr[W] = \frac{1}{L}\sum_{i=1}^{L} \Pr[W \mid \omega = i]$$

Proof

- **By Distinguisher/predictor Lemma:** $\Pr[W \mid \omega = i] = \left(\frac{1}{2} + p_{i-1} - p_i\right)$.

$$\begin{aligned} \Pr[W] &= \frac{1}{L}\sum_{i=1}^{L} \Pr[W \mid \omega = i] \\ &= \frac{1}{L}\sum_{i=1}^{L} \left(\frac{1}{2} + p_{i-1} - p_i\right) \end{aligned}$$

Proof

- **By Distinguisher/predictor Lemma:** $\Pr[W \mid \omega = i] = \left(\frac{1}{2} + p_{i-1} - p_i\right)$.

$$\begin{aligned}
\Pr[W] &= \frac{1}{L}\sum_{i=1}^{L} \Pr[W \mid \omega = i] \\
&= \frac{1}{L}\sum_{i=1}^{L} \left(\frac{1}{2} + p_{i-1} - p_i\right) \\
&= \frac{1}{2} + \frac{1}{L}(p_0 - p_L)
\end{aligned}$$

Unpredictability

Proof

$$\mathsf{PREDadv}[\mathcal{B}, G] = \left| \Pr[W] - \frac{1}{2} \right|$$

Unpredictability

Proof

$$\begin{aligned} \mathsf{PREDadv}[\mathcal{B}, G] &= \left| \Pr[W] - \frac{1}{2} \right| \\ &= \left| \frac{1}{L} \cdot (p_0 - p_L) \right| \end{aligned}$$

Unpredictability

Proof

$$\begin{aligned}
\mathsf{PREDadv}[\mathcal{B}, G] &= \left| \Pr[W] - \frac{1}{2} \right| \\
&= \left| \frac{1}{L} \cdot (p_0 - p_L) \right| \\
&= \frac{1}{L} \cdot | \ (p_0 - p_L) \ |
\end{aligned}$$

Unpredictability

Proof

$$
\begin{aligned}
\mathsf{PREDadv}[\mathcal{B}, G] &= \left| \Pr[W] - \frac{1}{2} \right| \\
&= \left| \frac{1}{L} \cdot (p_0 - p_L) \right| \\
&= \frac{1}{L} \cdot | \ (p_0 - p_L) \ | \\
&= \frac{1}{L} \cdot \mathsf{PRGadv}[\mathcal{A}, G].
\end{aligned}
$$

Unpredictability

Proof

$$\begin{aligned}
\mathsf{PREDadv}[\mathcal{B}, G] &= \left| \Pr[W] - \frac{1}{2} \right| \\
&= \left| \frac{1}{L} \cdot (p_0 - p_L) \right| \\
&= \frac{1}{L} \cdot | \ (p_0 - p_L) \ | \\
&= \frac{1}{L} \cdot \mathsf{PRGadv}[\mathcal{A}, G].
\end{aligned}$$

Therefore,

$$\mathsf{PRGadv}[\mathcal{A}, G] = L \cdot \mathsf{PREDadv}[\mathcal{B}, G] \leqslant L \cdot \epsilon(n).$$

Modes of Operation

Two Modes

- Synchronized mode, and
- Unsynchronized mode.

Synchronized mode

Synchronized mode

- Stateful, and the sender and receiver must be synchronized in the sense that they know how much plaintext has been encrypted (resp., decrypted) so far.
- They can use the same key to encrypt multiple messages.
- Sender and Receiver has the same (IV, key) pair.

Synchronized mode

Synchronized mode

- Stateful, and the sender and receiver must be synchronized in the sense that they know how much plaintext has been encrypted (resp., decrypted) so far.
- They can use the same key to encrypt multiple messages.
- Sender and Receiver has the same (IV, key) pair.
- Treat multiple messages $m_1, m_2, m_3, \ldots$ as a single, long message.
- Let the length of m_i be l_i.

Synchronized mode

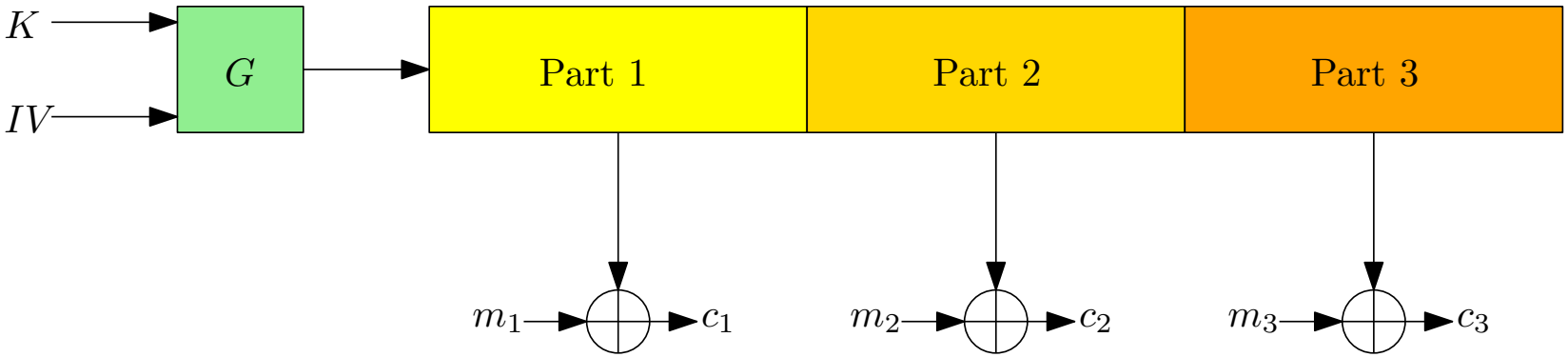
- Stateful, and the sender and receiver must be synchronized in the sense that they know how much plaintext has been encrypted (resp., decrypted) so far.
- They can use the same key to encrypt multiple messages.
- Sender and Receiver has the same (IV, key) pair.
- Treat multiple messages $m_1, m_2, m_3, \ldots$ as a single, long message.
- Let the length of m_i be l_i.
- Divide the output stream into multiple blocks $pad_1, pad_2, pad_3, \ldots$ such that length of pad_i is l_i.
- $c_i \longleftarrow m_i \oplus pad_i$.

Synchronized mode

Synchronized mode

- Stateful, and the sender and receiver must be synchronized in the sense that they know how much plaintext has been encrypted (resp., decrypted) so far.
- They can use the same key to encrypt multiple messages.
- Sender and Receiver has the same (IV, key) pair.
- Treat multiple messages $m_1, m_2, m_3, \ldots$ as a single, long message.
- Let the length of m_i be l_i.
- Divide the output stream into multiple blocks $pad_1, pad_2, pad_3, \ldots$ such that length of pad_i is l_i.
- $c_i \longleftarrow m_i \oplus pad_i$.
- Upon receiving the cipher texts, receiver also computes the output stream from the same (IV, key).
- Divide the output stream into multiple blocks $pad_1, pad_2, pad_3, \ldots$ such that length of pad_i is l_i.
- $m_i \longleftarrow c_i \oplus pad_i$.

Synchronized mode



Synchronized mode

Synchronized mode

- Appropriate when two parties are communicating within a single session.
- Does not work well for sporadic communication or when a party might, over time, communicate from different devices.
  - easy to keep copies of a fixed key in different locations;
  - harder to maintain synchronized state across multiple locations
- If the parties ever get out of sync decryption will return an incorrect result.
- Resynchronization is possible, but adds additional overhead.

Unsynchronized mode

- Stateless.
- They can use the same key to encrypt multiple messages.
- Each message has different IV.

Unsynchronized mode

Unsynchronized mode

- Stateless.
- They can use the same key to encrypt multiple messages.
- Each message has different IV.
- Let i-th message be m_i.
- Let the length of m_i be l_i.

Unsynchronized mode

Unsynchronized mode

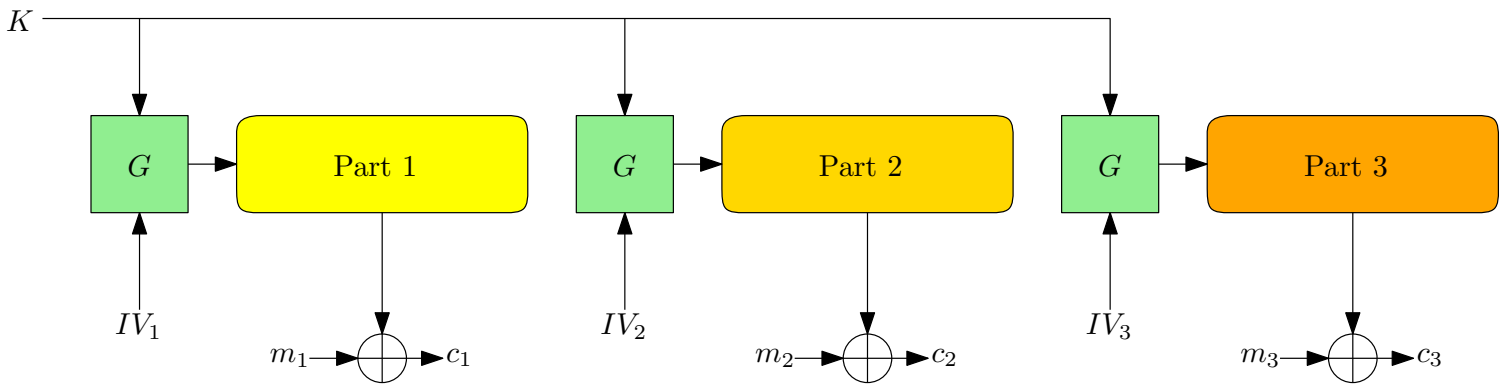
- Stateless.
- They can use the same key to encrypt multiple messages.
- Each message has different IV.
- Let i-th message be m_i.
- Let the length of m_i be l_i.
- Generate a new IV, say IV_i.
- Generate key stream of (IV_i, key) say pad_i.
- $c_i \longleftarrow m_i \oplus pad_i[0, 1, \ldots, l_i]$.
- Send (IV_i, c_i).

Unsynchronized mode

Unsynchronized mode

- Stateless.
- They can use the same key to encrypt multiple messages.
- Each message has different IV.
- Let i-th message be m_i.
- Let the length of m_i be l_i.
- Generate a new IV, say IV_i.
- Generate key stream of (IV_i, key) say pad_i.
- $c_i \longleftarrow m_i \oplus pad_i[0, 1, \ldots, l_i]$.
- Send (IV_i, c_i).
- Upon receiving the cipher text, receiver also computes pad_i from the same (IV_i, key).
- $m_i \longleftarrow c_i \oplus pad_i[0, 1, \ldots, l_i]$.

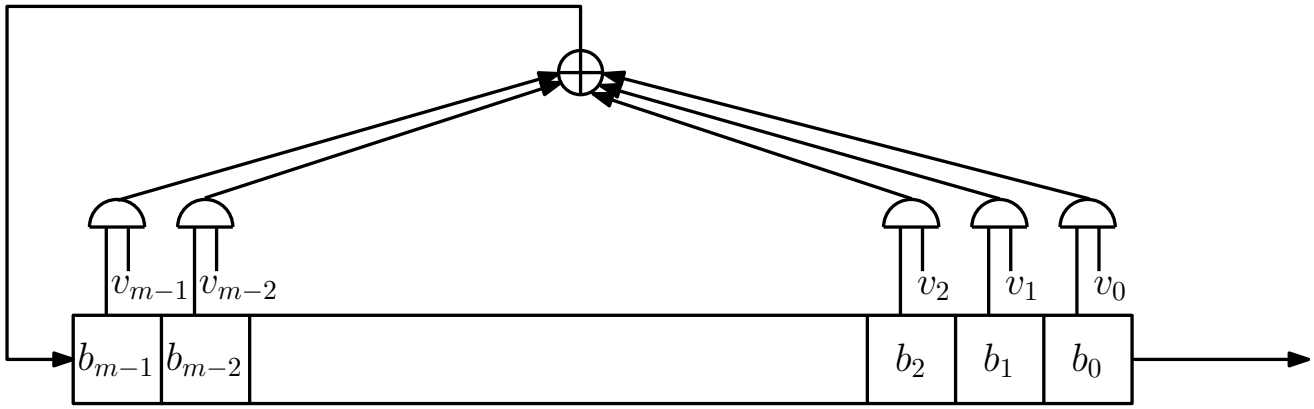
Unsynchronized mode



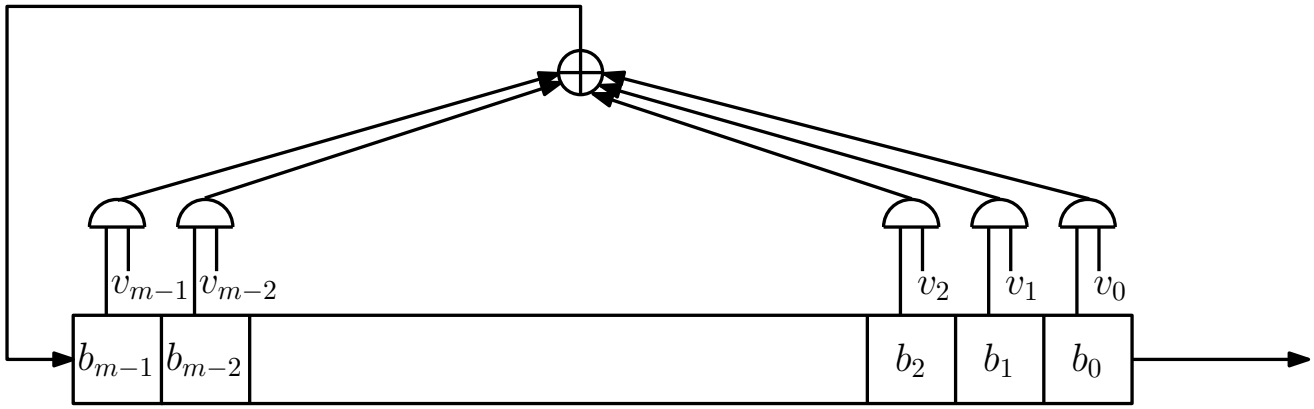
Case Study

- LFSR based PRG
  - CSS
  - Trivium
- RC4
- Salsa and ChaCha

Linear Feedback Shift Register (LFSR)



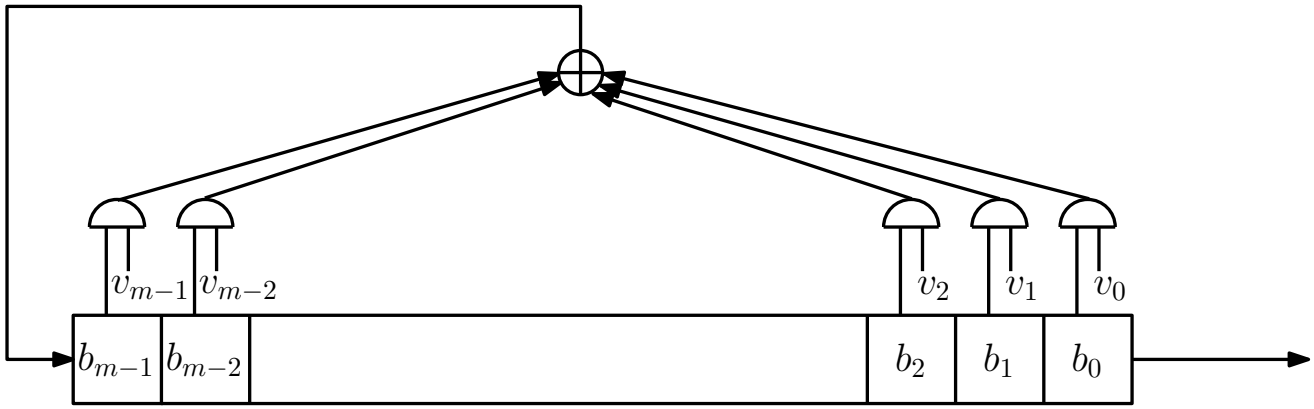
Linear Feedback Shift Register (LFSR)



LFSR

- Let the register be *m*-bit long.

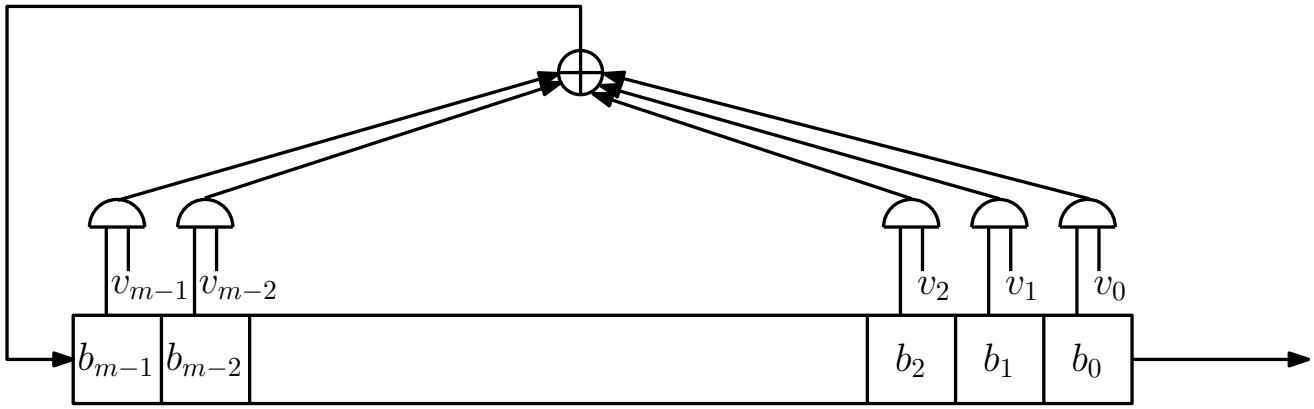
Linear Feedback Shift Register (LFSR)



LFSR

- Let the register be m-bit long.
- Tap Position $V = \{v_i \mid v_i = 1\}$.

Linear Feedback Shift Register (LFSR)



LFSR

- Let the register be m-bit long.
- Tap Position $V = \{v_i \mid v_i = 1\}$.

$$b_m = b_0 v_0 + b_1 v_1 + \cdots + b_{m-1} v_{m-1} \pmod 2.$$

Linear Feedback Shift Register (LFSR)

LFSR

- Let the register be m-bit long.

Linear Feedback Shift Register (LFSR)

LFSR

- Let the register be m-bit long.
- Tap Position $V = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$.

Linear Feedback Shift Register (LFSR)

LFSR

- Let the register be m-bit long.
- Tap Position $V = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$.

Input: $s = (b_{m-1}, \ldots, b_0) \in \{0,1\}^m$ and $s \neq 0^m$
Output: $y \in \{0,1\}^\ell$ where $\ell > m$

Linear Feedback Shift Register (LFSR)

LFSR

- Let the register be m-bit long.
- Tap Position $V = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$.

Input: $s = (b_{m-1}, \ldots, b_0) \in \{0,1\}^m$ and $s \neq 0^m$
Output: $y \in \{0,1\}^\ell$ where $\ell > m$

for $i \longleftarrow 1, \ldots, \ell$ do

Linear Feedback Shift Register (LFSR)

LFSR

- Let the register be m-bit long.
- Tap Position $V = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$.

Input: $s = (b_{m-1}, \ldots, b_0) \in \{0,1\}^m$ and $s \neq 0^m$
Output: $y \in \{0,1\}^\ell$ where $\ell > m$

for $i \longleftarrow 1, \ldots, \ell$ do
    output b_0 //output one bit

Linear Feedback Shift Register (LFSR)

LFSR

- Let the register be m-bit long.
- Tap Position $V = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$.

Input: $s = (b_{m-1}, \ldots, b_0) \in \{0,1\}^m$ and $s \neq 0^m$
Output: $y \in \{0,1\}^\ell$ where $\ell > m$

for $i \longleftarrow 1, \ldots, \ell$ do
    output b_0 //output one bit
    $b \longleftarrow b_{v_{i_1}} \oplus \cdots \oplus b_{v_{i_d}}$ //compute feedback bit

Linear Feedback Shift Register (LFSR)

LFSR

- Let the register be m-bit long.
- Tap Position $V = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$.

Input: $s = (b_{m-1}, \ldots, b_0) \in \{0,1\}^m$ and $s \neq 0^m$
Output: $y \in \{0,1\}^\ell$ where $\ell > m$

for $i \longleftarrow 1, \ldots, \ell$ do
    output b_0 //output one bit
    $b \longleftarrow b_{v_{i_1}} \oplus \cdots \oplus b_{v_{i_d}}$ //compute feedback bit
    $s \longleftarrow (b, b_{m-1}, \ldots, b_1)$ //shift register bits to the right

Linear Feedback Shift Register (LFSR)

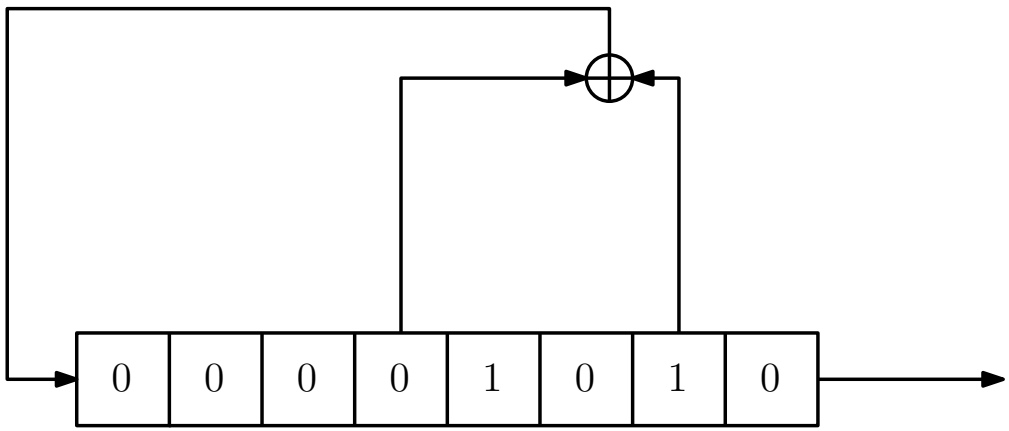


Figure: LFSR Example

Linear Feedback Shift Register (LFSR)

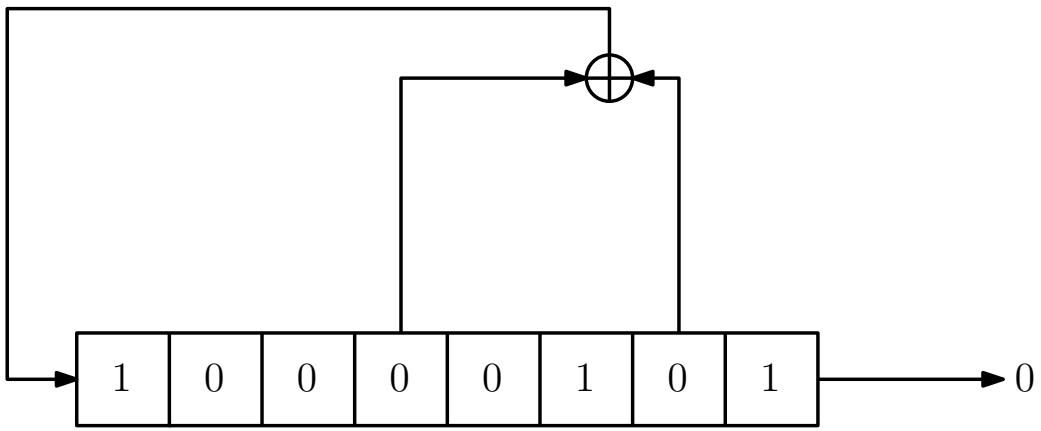


Figure: LFSR Example

Linear Feedback Shift Register (LFSR)

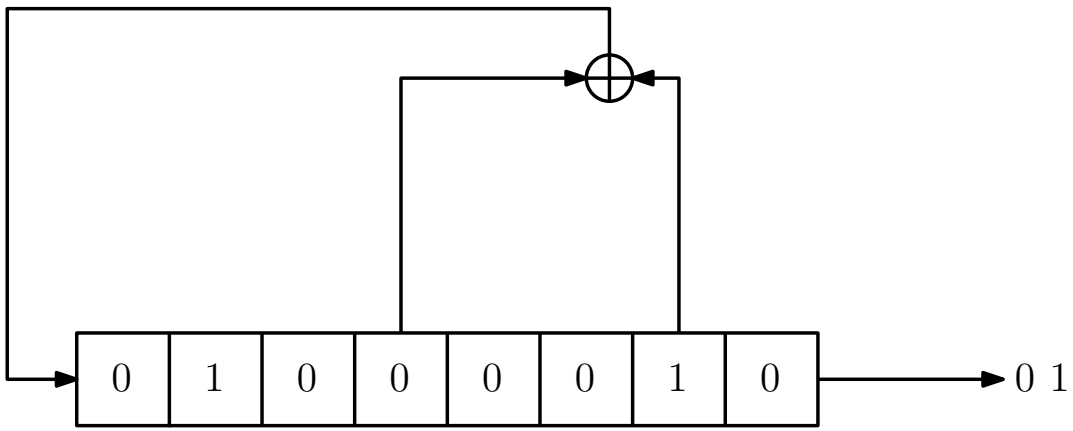


Figure: LFSR Example

Linear Feedback Shift Register (LFSR)

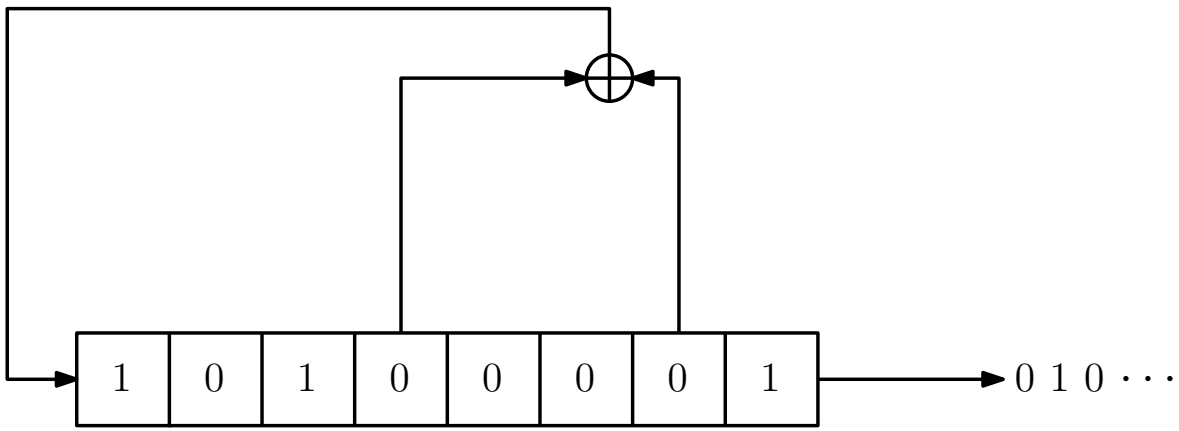


Figure: LFSR Example

Stream ciphers from LSFRs

- A single LFSR is completely insecure as a PRG
  - Given m consecutive bits of its output it is trivial to compute all subsequent bits.

Stream ciphers from LSFRs

- A single LFSR is completely insecure as a PRG
  - Given m consecutive bits of its output it is trivial to compute all subsequent bits.
- Combine several LFSRs using a non-linear component.
  - Get some (weak) security as a PRG.

Stream ciphers from LSFRs

- A single LFSR is completely insecure as a PRG
  - Given m consecutive bits of its output it is trivial to compute all subsequent bits.
- Combine several LFSRs using a non-linear component.
  - Get some (weak) security as a PRG.
- Approach 1:
  - Run several LFSRs in parallel, and
  - Combine their output using a non-linear operation.

Stream ciphers from LSFRs

- A single LFSR is completely insecure as a PRG
  - Given m consecutive bits of its output it is trivial to compute all subsequent bits.
- Combine several LFSRs using a non-linear component.
  - Get some (weak) security as a PRG.
- Approach 1:
  - Run several LFSRs in parallel, and
  - Combine their output using a non-linear operation.
    - CSS (2 LFSR)
    - Trivium (3 LFSR)
    - A5/1 stream cipher used to encrypt GSM cell phone traffic (3 LFSR)
    - Bluetooth E0 stream cipher (4 LFSR)

Stream ciphers from LSFRs

- A single LFSR is completely insecure as a PRG
  - Given m consecutive bits of its output it is trivial to compute all subsequent bits.
- Combine several LFSRs using a non-linear component.
  - Get some (weak) security as a PRG.
- Approach 1:
  - Run several LFSRs in parallel, and
  - Combine their output using a non-linear operation.
    - CSS (2 LFSR)
    - Trivium (3 LFSR)
    - A5/1 stream cipher used to encrypt GSM cell phone traffic (3 LFSR)
    - Bluetooth E0 stream cipher (4 LFSR)
- Approach 2:
  - Run a single LFSR, and
  - Generate the output from a non-linear operation on its internal state.

Stream ciphers from LSFRs

- A single LFSR is completely insecure as a PRG
  - Given m consecutive bits of its output it is trivial to compute all subsequent bits.
- Combine several LFSRs using a non-linear component.
  - Get some (weak) security as a PRG.
- Approach 1:
  - Run several LFSRs in parallel, and
  - Combine their output using a non-linear operation.
    - CSS (2 LFSR)
    - Trivium (3 LFSR)
    - A5/1 stream cipher used to encrypt GSM cell phone traffic (3 LFSR)
    - Bluetooth E0 stream cipher (4 LFSR)
- Approach 2:
  - Run a single LFSR, and
  - Generate the output from a non-linear operation on its internal state.
    - `snow 3G` cipher used to encrypt 3GPP cell phone traffic

Content Scrambling System (CSS)

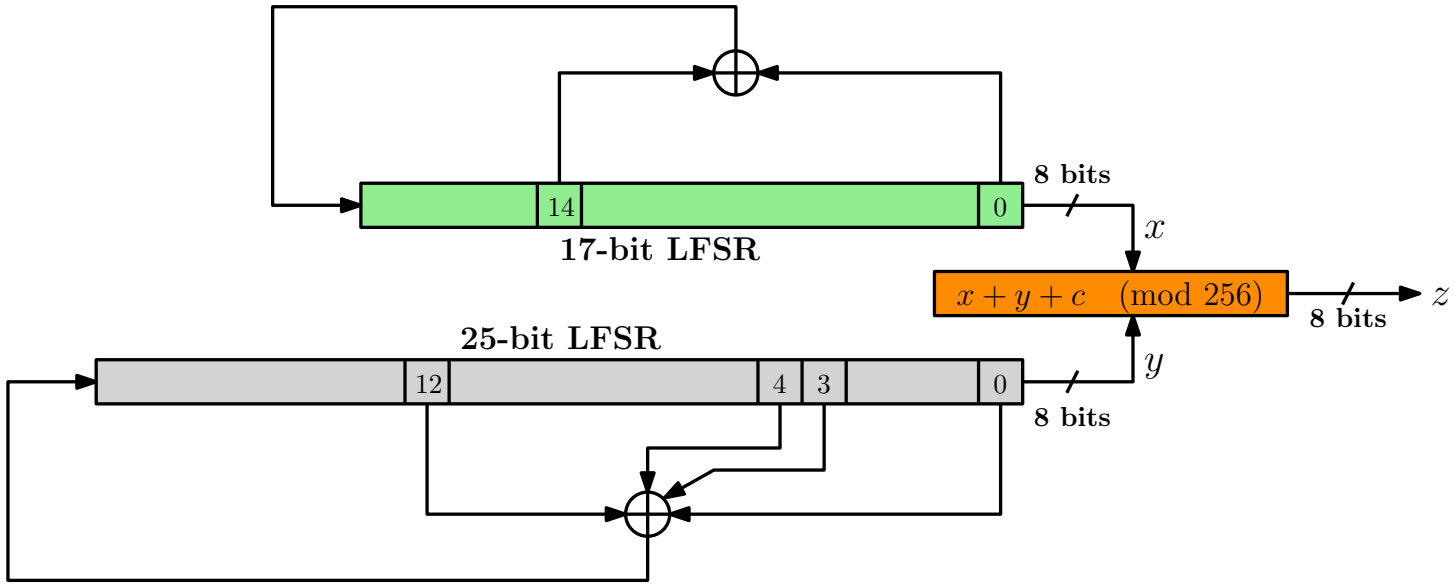
CSS

- Used for protecting movies on DVD disk.
- Designed in the 1980.
- Key size was restricted to 40-bit keys.

Content Scrambling System (CSS)

CSS

- Used for protecting movies on DVD disk.
- Designed in the 1980.
- Key size was restricted to 40-bit keys.



Content Scrambling System (CSS)

CSS

Input: seed $s \in \{0,1\}^{40}$

Output: ℓ bytes

Content Scrambling System (CSS)

CSS

Input: seed $s \in \{0,1\}^{40}$
Output: ℓ bytes

/*Initialization Stage*/
write $s = s_1 \| s_2$ where $s_1 \in \{0,1\}^{16}$ and $s_1 \in \{0,1\}^{24}$
load $1 \| s_1$ into a 17-bit LFSR
load $1 \| s_2$ into a 25-bit LFSR
$c \longleftarrow 0$ // carry bit

Content Scrambling System (CSS)

CSS

Input: seed $s \in \{0,1\}^{40}$
Output: ℓ bytes

/*Initialization Stage*/
write $s = s_1 \| s_2$ where $s_1 \in \{0,1\}^{16}$ and $s_1 \in \{0,1\}^{24}$
load $1\|s_1$ into a 17-bit LFSR
load $1\|s_2$ into a 25-bit LFSR
$c \longleftarrow 0$ // carry bit

/*Computational Stage*/
for $i \longleftarrow 1, \ldots, \ell$ do
    run both LFSRs for eight cycles to obtain $x_i, y_i \in \{0,1\}^8$
    treat x_i and y_i as integers in $0, \ldots, 255$
    output $z_i := x_i + y_i + c \pmod{256}$
    if $x_i + y_i > 255$ then $c \longleftarrow 1$ else $c \longleftarrow 0$ //carry bit

Cryptanalysis of CSS

Cryptanalysis

- Given the PRG output, recover the secret seed in time 2^{40} by exhaustive search.

Cryptanalysis of CSS

Cryptanalysis

- Given the PRG output, recover the secret seed in time 2^{40} by exhaustive search.
- Faster attack that takes only 2^{16} guesses.

Cryptanalysis of CSS

Cryptanalysis

- Given the PRG output, recover the secret seed in time 2^{40} by exhaustive search.
- Faster attack that takes only 2^{16} guesses.
  - Given the first 100 bytes $\bar{z} := (z_1, z_2, \ldots)$ outputs.
  - Observation:
    - Let (x_1, x_2, x_3) be the first three bytes output by the 17-bit LFSR.
    - Let (y_1, y_2, y_3) be the first three bytes output by the 25-bit LFSR.

$$(x_1 + x_2 2^8 + x_3 2^{16}) + (y_1 + y_2 2^8 + y_3 2^{16}) \equiv (z_1 + z_2 2^8 + z_3 2^{16}) \pmod{2^{24}}.$$

- Given (x_1, x_2, x_3) and (z_1, z_2, z_3), uniquely compute (y_1, y_2, y_3).

Cryptanalysis of CSS

Cryptanalysis

- Given the PRG output, recover the secret seed in time 2^{40} by exhaustive search.
- Faster attack that takes only 2^{16} guesses.
  - Given the first 100 bytes $\bar{z} := (z_1, z_2, \ldots)$ outputs.
  - Observation:
    - Let (x_1, x_2, x_3) be the first three bytes output by the 17-bit LFSR.
    - Let (y_1, y_2, y_3) be the first three bytes output by the 25-bit LFSR.

$$(x_1 + x_2 2^8 + x_3 2^{16}) + (y_1 + y_2 2^8 + y_3 2^{16}) \equiv (z_1 + z_2 2^8 + z_3 2^{16}) \pmod{2^{24}}.$$

  - Given (x_1, x_2, x_3) and (z_1, z_2, z_3), uniquely compute (y_1, y_2, y_3).

1. Make a guess for s_1, and compute (x_1, x_2, x_3).

Cryptanalysis of CSS

Cryptanalysis

- Given the PRG output, recover the secret seed in time 2^{40} by exhaustive search.
- Faster attack that takes only 2^{16} guesses.
  - Given the first 100 bytes $\bar{z} := (z_1, z_2, \ldots)$ outputs.
  - Observation:
    - Let (x_1, x_2, x_3) be the first three bytes output by the 17-bit LFSR.
    - Let (y_1, y_2, y_3) be the first three bytes output by the 25-bit LFSR.

$$(x_1 + x_2 2^8 + x_3 2^{16}) + (y_1 + y_2 2^8 + y_3 2^{16}) \equiv (z_1 + z_2 2^8 + z_3 2^{16}) \pmod{2^{24}}.$$

  - Given (x_1, x_2, x_3) and (z_1, z_2, z_3), uniquely compute (y_1, y_2, y_3).

1. Make a guess for s_1, and compute (x_1, x_2, x_3).
2. Compute (y_1, y_2, y_3) from (x_1, x_2, x_3) and (z_1, z_2, z_3).

Cryptanalysis of CSS

Cryptanalysis

- Given the PRG output, recover the secret seed in time 2^{40} by exhaustive search.
- Faster attack that takes only 2^{16} guesses.
  - Given the first 100 bytes $\bar{z} := (z_1, z_2, \ldots)$ outputs.
  - Observation:
    - Let (x_1, x_2, x_3) be the first three bytes output by the 17-bit LFSR.
    - Let (y_1, y_2, y_3) be the first three bytes output by the 25-bit LFSR.

$$(x_1 + x_2 2^8 + x_3 2^{16}) + (y_1 + y_2 2^8 + y_3 2^{16}) \equiv (z_1 + z_2 2^8 + z_3 2^{16}) \pmod{2^{24}}.$$

  - Given (x_1, x_2, x_3) and (z_1, z_2, z_3), uniquely compute (y_1, y_2, y_3).

1. Make a guess for s_1, and compute (x_1, x_2, x_3).
2. Compute (y_1, y_2, y_3) from (x_1, x_2, x_3) and (z_1, z_2, z_3).
3. Compute s_2 from (y_1, y_2, y_3).

Cryptanalysis of CSS

Cryptanalysis

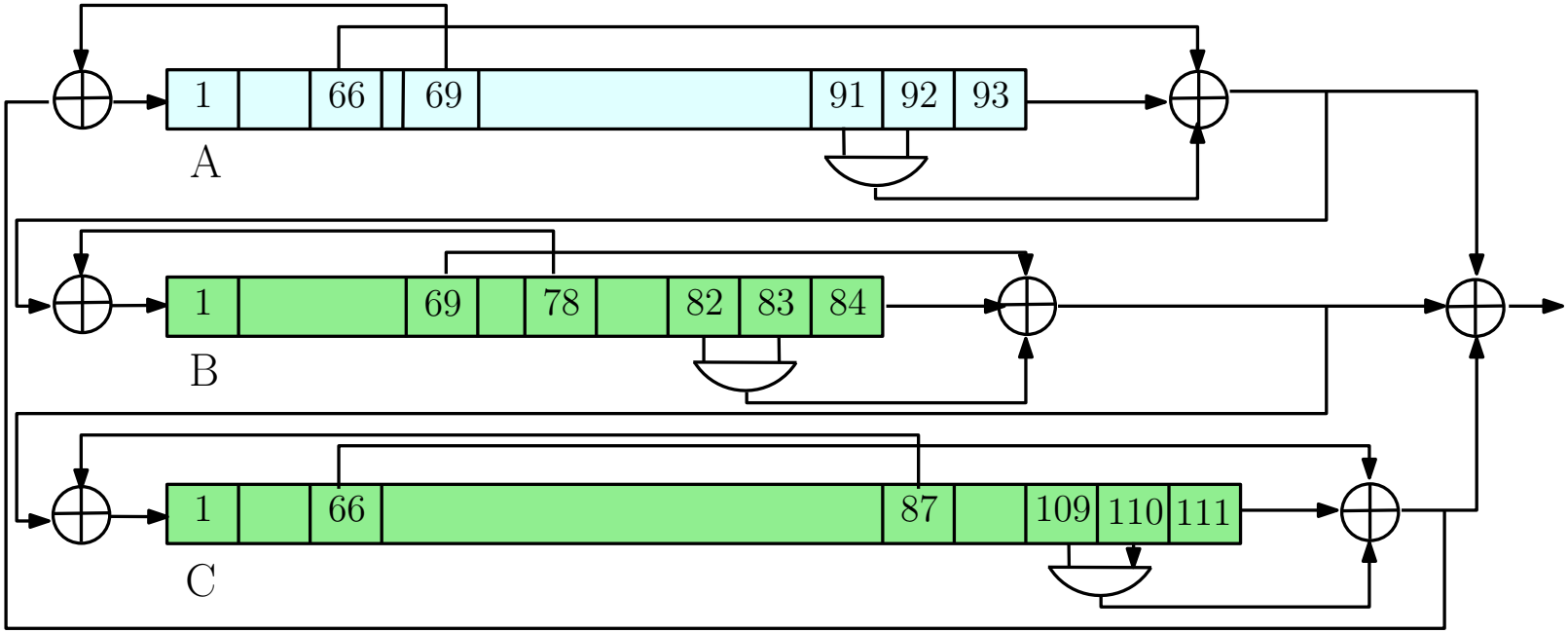
- Given the PRG output, recover the secret seed in time 2^{40} by exhaustive search.
- Faster attack that takes only 2^{16} guesses.
  - Given the first 100 bytes $\bar{z} := (z_1, z_2, \ldots)$ outputs.
  - Observation:
    - Let (x_1, x_2, x_3) be the first three bytes output by the 17-bit LFSR.
    - Let (y_1, y_2, y_3) be the first three bytes output by the 25-bit LFSR.

$$(x_1 + x_2 2^8 + x_3 2^{16}) + (y_1 + y_2 2^8 + y_3 2^{16}) \equiv (z_1 + z_2 2^8 + z_3 2^{16}) \pmod{2^{24}}.$$

- Given (x_1, x_2, x_3) and (z_1, z_2, z_3), uniquely compute (y_1, y_2, y_3).

1. Make a guess for s_1, and compute (x_1, x_2, x_3).
2. Compute (y_1, y_2, y_3) from (x_1, x_2, x_3) and (z_1, z_2, z_3).
3. Compute s_2 from (y_1, y_2, y_3).
4. Test the correctness, and continue until the correct guess.

Trivium



Trivium

- Input:
  - Key k and Initialization Vector IV.
  - IV serves as randomizer, once per encryption session.
  - IV is also referred as number used once or nonce.

Trivium

- Input:
  - Key k and Initialization Vector IV.
  - IV serves as randomizer, once per encryption session.
  - IV is also referred as number used once or nonce.
- Initialization:
  - An 80-bit IV is loaded into the 80 leftmost locations of register A.

Trivium

- Input:
  - Key k and Initialization Vector IV.
  - IV serves as randomizer, once per encryption session.
  - IV is also referred as number used once or nonce.
- Initialization:
  - An 80-bit IV is loaded into the 80 leftmost locations of register A.
  - An 80-bit key is loaded in the 80 leftmost locations of register B.

Trivium

- Input:
  - Key k and Initialization Vector IV.
  - IV serves as randomizer, once per encryption session.
  - IV is also referred as number used once or nonce.
- Initialization:
  - An 80-bit IV is loaded into the 80 leftmost locations of register A.
  - An 80-bit key is loaded in the 80 leftmost locations of register B.
  - All other register bits are set to zero.

Trivium

- Input:
  - Key k and Initialization Vector IV.
  - IV serves as randomizer, once per encryption session.
  - IV is also referred as number used once or nonce.
- Initialization:
  - An 80-bit IV is loaded into the 80 leftmost locations of register A.
  - An 80-bit key is loaded in the 80 leftmost locations of register B.
  - All other register bits are set to zero.
  - Exception: The three rightmost bits of register C are set to 1.

Trivium

- Input:
  - Key k and Initialization Vector IV.
  - IV serves as randomizer, once per encryption session.
  - IV is also referred as number used once or nonce.
- Initialization:
  - An 80-bit IV is loaded into the 80 leftmost locations of register A.
  - An 80-bit key is loaded in the 80 leftmost locations of register B.
  - All other register bits are set to zero.
  - Exception: The three rightmost bits of register C are set to 1.
- Warm-up Phase:
  - Clocked $4 \times 288 = 1152$ times.

Trivium

- Input:
  - Key k and Initialization Vector IV.
  - IV serves as randomizer, once per encryption session.
  - IV is also referred as number used once or nonce.
- Initialization:
  - An 80-bit IV is loaded into the 80 leftmost locations of register A.
  - An 80-bit key is loaded in the 80 leftmost locations of register B.
  - All other register bits are set to zero.
  - Exception: The three rightmost bits of register C are set to 1.
- Warm-up Phase:
  - Clocked $4 \times 288 = 1152$ times.
  - Start with the output bit at clock 1153.

RC4

- The RC4 (Rivest Cipher 4) stream cipher was designed by Ron Rivest in 1987.

RC4

- The RC4 (Rivest Cipher 4) stream cipher was designed by Ron Rivest in 1987.
- Historically used:
  - in the SSL/TLS protocol to secure Web traffic,
  - in the 802.11b WEP protocol to secure wireless traffic.

RC4

- The RC4 (Rivest Cipher 4) stream cipher was designed by Ron Rivest in 1987.
- Historically used:
  - in the SSL/TLS protocol to secure Web traffic,
  - in the 802.11b WEP protocol to secure wireless traffic.
- Designed to operate on 8-bit processors with little internal memory.

RC4

- The RC4 (Rivest Cipher 4) stream cipher was designed by Ron Rivest in 1987.
- Historically used:
  - in the SSL/TLS protocol to secure Web traffic,
  - in the 802.11b WEP protocol to secure wireless traffic.
- Designed to operate on 8-bit processors with little internal memory.
- CSS and Trivium works good for Hardware implementation, but poorly in software implementation.

RC4

- The RC4 (Rivest Cipher 4) stream cipher was designed by Ron Rivest in 1987.
- Historically used:
  - in the SSL/TLS protocol to secure Web traffic,
  - in the 802.11b WEP protocol to secure wireless traffic.
- Designed to operate on 8-bit processors with little internal memory.
- CSS and Trivium works good for Hardware implementation, but poorly in software implementation.
- RC4 works good when implemented in software.

RC4

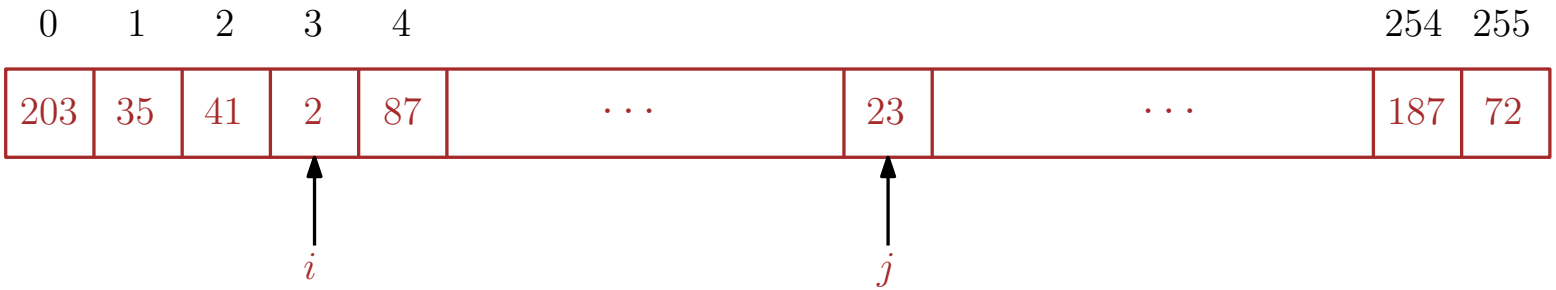
RC4

- Input: Variable-length key(s) of 1 to 256 bytes (8 to 2048 bits).
  - typically between 5 and 16 bytes.

RC4

RC4

- Input: Variable-length key(s) of 1 to 256 bytes (8 to 2048 bits).
  - typically between 5 and 16 bytes.
- Internal State: Consists of an array S of 256 bytes and two additional bytes i, j.
  - The array S contains all the numbers $0, \ldots, 255$.



RC4

S Initialization

Input: seed s

Output: Initialized S

S Initialization

Input: seed s
Output: Initialized S

for $i \longleftarrow 0, \ldots, 255$ do
    $S[i] \longleftarrow i$

$j \longleftarrow 0$
for $i \longleftarrow 0, \ldots, 255$ do
    $k \longleftarrow s[i \pmod{|s|}]$//extract one byte from seed
    $j \longleftarrow (j + S[i] + k) \pmod{256}$
    swap$(S[i], S[j])$

RC4 Pseudo-Random Stream generation

Input: Initialized S

Output: RC4 Pseudo-Random Stream of 1 byte per cycle

RC4

RC4 Pseudo-Random Stream generation

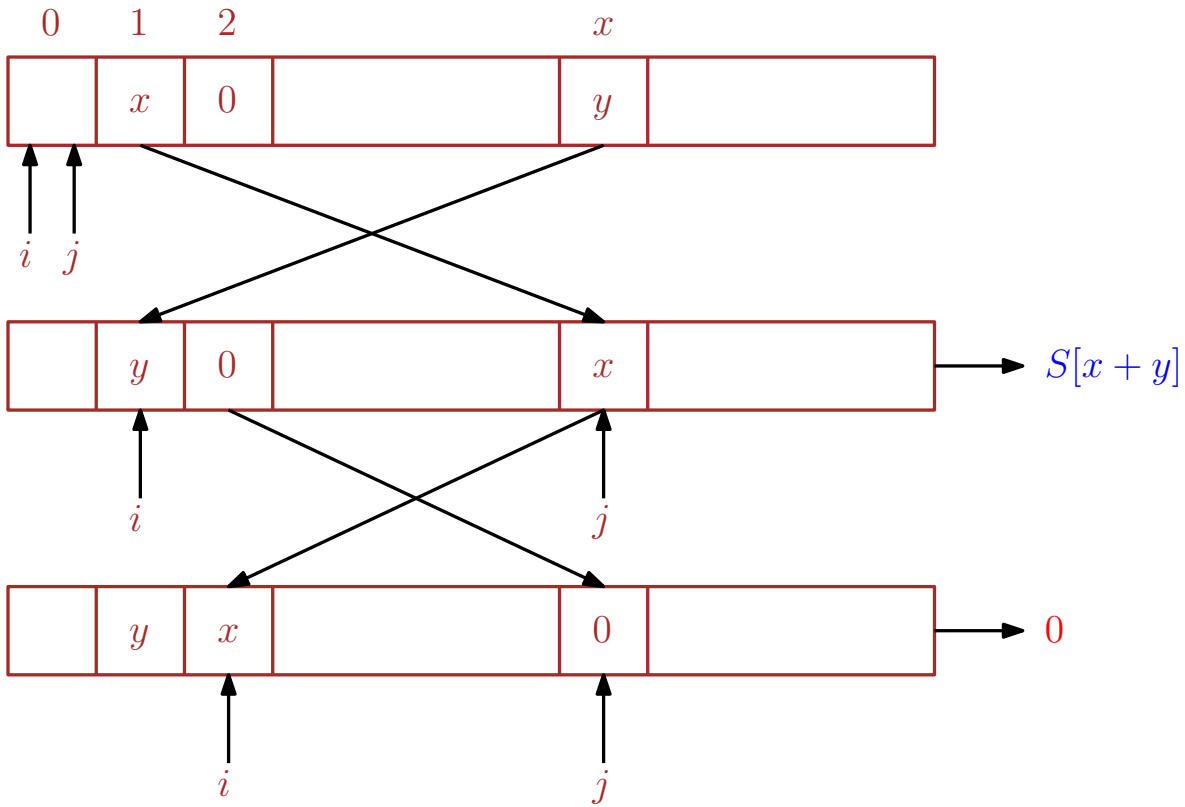
Input: Initialized S
Output: RC4 Pseudo-Random Stream of 1 byte per cycle

$i \longleftarrow 0$; $j \longleftarrow 0$;
repeat
  $i \longleftarrow (i+1) \pmod{256}$
  $j \longleftarrow (j+S[i]) \pmod{256}$
  swap$(S[i], S[j])$
  output $S[S[i]+S[j] \pmod{256}]$
forever

RC4 Weaknesses

Lemma (Mantin-Shamir)

Suppose the array S is set to random permutation of $0, \ldots, n-1$ and that i, j are set to 0. Then the probability that the second byte of the output of RC4 is equal to 0 is $\frac{2}{n}$.



RC4 Weaknesses

Proof

- Let z_2 be the second byte output by RC4.
- Let P be the event that $S[2] = 0$ and $S[1] \neq 2$.

RC4 Weaknesses

Proof

- Let $\mathbf{z}_2$ be the second byte output by RC4.
- Let P be the event that $S[2] = 0$ and $S[1] \neq 2$.
- Observations:
  - When event P happens then $\mathbf{z}_2 = 0$ with probability 1.

RC4 Weaknesses

Proof

- Let $\mathbf{z}_2$ be the second byte output by RC4.
- Let P be the event that $S[2] = 0$ and $S[1] \neq 2$.
- Observations:
  - When event P happens then $\mathbf{z}_2 = 0$ with probability 1.
  - When P does not happen then $\mathbf{z}_2$ is uniformly distributed in $0, \ldots, n-1$.

Proof

- Let $\mathbf{z}_2$ be the second byte output by RC4.
- Let P be the event that $S[2] = 0$ and $S[1] \neq 2$.
- Observations:
  - When event P happens then $\mathbf{z}_2 = 0$ with probability 1.
  - When P does not happen then $\mathbf{z}_2$ is uniformly distributed in $0, \ldots, n-1$.

$$\Pr[\mathbf{z}_2 = 0] \quad = \quad \Pr[\mathbf{z}_2 = 0 \mid P]\Pr[P] + \Pr[\mathbf{z}_2 = 0 \mid \neg P]\Pr[\neg P]$$

RC4 Weaknesses

Proof

- Let $\mathbf{z}_2$ be the second byte output by RC4.
- Let P be the event that $S[2] = 0$ and $S[1] \neq 2$.
- Observations:
  - When event P happens then $\mathbf{z}_2 = 0$ with probability 1.
  - When P does not happen then $\mathbf{z}_2$ is uniformly distributed in $0, \ldots, n-1$.

$$\begin{aligned} \Pr[\mathbf{z}_2 = 0] &= \Pr[\mathbf{z}_2 = 0 \mid P]\Pr[P] + \Pr[\mathbf{z}_2 = 0 \mid \neg P]\Pr[\neg P] \\ &\approx 1 \cdot \frac{1}{n} + \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right) \\ &\approx \frac{2}{n} \end{aligned}$$

RC4 Weaknesses

Proof

- Let $\mathbf{z}_2$ be the second byte output by RC4.
- Let P be the event that $S[2] = 0$ and $S[1] \neq 2$.
- Observations:
  - When event P happens then $\mathbf{z}_2 = 0$ with probability 1.
  - When P does not happen then $\mathbf{z}_2$ is uniformly distributed in $0, \ldots, n-1$.

$$\begin{aligned} \Pr[\mathbf{z}_2 = 0] &= \Pr[\mathbf{z}_2 = 0 \mid P]\Pr[P] + \Pr[\mathbf{z}_2 = 0 \mid \neg P]\Pr[\neg P] \\ &\approx 1 \cdot \frac{1}{n} + \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right) \\ &\approx \frac{2}{n} \end{aligned}$$

- In fact, first and third bytes are also biased.

RC4 Weaknesses

Proof

- Let $\mathbf{z}_2$ be the second byte output by RC4.
- Let P be the event that $S[2] = 0$ and $S[1] \neq 2$.
- Observations:
  - When event P happens then $\mathbf{z}_2 = 0$ with probability 1.
  - When P does not happen then $\mathbf{z}_2$ is uniformly distributed in $0, \ldots, n-1$.

$$\begin{aligned} \Pr[\mathbf{z}_2 = 0] &= \Pr[\mathbf{z}_2 = 0 \mid P]\Pr[P] + \Pr[\mathbf{z}_2 = 0 \mid \neg P]\Pr[\neg P] \\ &\approx 1 \cdot \frac{1}{n} + \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right) \\ &\approx \frac{2}{n} \end{aligned}$$

- In fact, first and third bytes are also biased.
- Recommendation: Discard first 256 bytes.

RC4 Weaknesses

Lemma ((Fluhrer-McGrew)

Suppose RC4 is initialized with a random state T in ST_{RC4}, where ST_{RC4} be the set of all possible internal states of RC4. Let $(\mathbf{z}_1, \mathbf{z}_2)$ be the first two bytes output by RC4 when started in state T. Then

$$i \neq n-1 \quad \Rightarrow \quad \Pr[(\mathbf{z}_1, \mathbf{z}_2) = (0,0)] \geqslant \frac{1}{n^2} \cdot \left(1 + \frac{1}{n}\right)$$

$$i \neq 0, 1 \quad \Rightarrow \quad \Pr[(\mathbf{z}_1, \mathbf{z}_2) = (0,1)] \geqslant \frac{1}{n^2} \cdot \left(1 + \frac{1}{n}\right)$$

- A pair of consecutive outputs $(\mathbf{z}_1, \mathbf{z}_2)$ is called a digraph.

RC4 Weaknesses

Lemma ((Fluhrer-McGrew)

Suppose RC4 is initialized with a random state T in ST_{RC4}, where ST_{RC4} be the set of all possible internal states of RC4. Let $(\mathbf{z}_1, \mathbf{z}_2)$ be the first two bytes output by RC4 when started in state T. Then

$$i \neq n-1 \quad \Rightarrow \quad \Pr[(\mathbf{z}_1, \mathbf{z}_2) = (0,0)] \geqslant \frac{1}{n^2} \cdot \left(1 + \frac{1}{n}\right)$$

$$i \neq 0, 1 \quad \Rightarrow \quad \Pr[(\mathbf{z}_1, \mathbf{z}_2) = (0,1)] \geqslant \frac{1}{n^2} \cdot \left(1 + \frac{1}{n}\right)$$

- A pair of consecutive outputs $(\mathbf{z}_1, \mathbf{z}_2)$ is called a digraph.
- In a truly random string, the probability of all digraphs (x, y) is exactly $\frac{1}{n^2}$.

RC4 Weaknesses

Lemma ((Fluhrer-McGrew)

Suppose RC4 is initialized with a random state T in ST_{RC4}, where ST_{RC4} be the set of all possible internal states of RC4. Let $(\mathbf{z}_1, \mathbf{z}_2)$ be the first two bytes output by RC4 when started in state T. Then

$$i \neq n-1 \quad \Rightarrow \quad \Pr[(\mathbf{z}_1, \mathbf{z}_2) = (0,0)] \geqslant \frac{1}{n^2} \cdot \left(1 + \frac{1}{n}\right)$$

$$i \neq 0, 1 \quad \Rightarrow \quad \Pr[(\mathbf{z}_1, \mathbf{z}_2) = (0,1)] \geqslant \frac{1}{n^2} \cdot \left(1 + \frac{1}{n}\right)$$

- A pair of consecutive outputs $(\mathbf{z}_1, \mathbf{z}_2)$ is called a digraph.
- In a truly random string, the probability of all digraphs (x, y) is exactly $\frac{1}{n^2}$.
- For RC4, the probability of $(0,0)$ is greater by $\frac{1}{n^3}$ from what it should be.
- The same holds for the digraph $(0,1)$.

RC4 Weaknesses

Lemma ((Fluhrer-McGrew)

Suppose RC4 is initialized with a random state T in ST_{RC4}, where ST_{RC4} be the set of all possible internal states of RC4. Let $(\mathbf{z}_1, \mathbf{z}_2)$ be the first two bytes output by RC4 when started in state T. Then

$$i \neq n-1 \quad \Rightarrow \quad \Pr[(\mathbf{z}_1, \mathbf{z}_2) = (0,0)] \geqslant \frac{1}{n^2} \cdot \left(1 + \frac{1}{n}\right)$$

$$i \neq 0, 1 \quad \Rightarrow \quad \Pr[(\mathbf{z}_1, \mathbf{z}_2) = (0,1)] \geqslant \frac{1}{n^2} \cdot \left(1 + \frac{1}{n}\right)$$

- A pair of consecutive outputs $(\mathbf{z}_1, \mathbf{z}_2)$ is called a digraph.
- In a truly random string, the probability of all digraphs (x, y) is exactly $\frac{1}{n^2}$.
- For RC4, the probability of $(0,0)$ is greater by $\frac{1}{n^3}$ from what it should be.
- The same holds for the digraph $(0,1)$.

- Another weakness of RC4 is related key attack.

Salasa and ChaCha

- Outcome of a project called eStream that was concluded in 2008.
- Designed by Dan Bernstein in 2005.

Salasa and ChaCha

- Outcome of a project called eStream that was concluded in 2008.
- Designed by Dan Bernstein in 2005.
- Defined as:

$$G : \mathcal{S} \times N \longrightarrow \mathcal{R}, \text{ where}$$

  - $\mathcal{S} = \{0,1\}^{128 \text{ or } 256}$

Salasa and ChaCha

- Outcome of a project called eStream that was concluded in 2008.
- Designed by Dan Bernstein in 2005.
- Defined as:

$$G : \mathcal{S} \times N \longrightarrow \mathcal{R}, \text{ where}$$

  - $\mathcal{S} = \{0,1\}^{128 \text{ or } 256}$
  - $\mathcal{R} = \{0,1\}^{512 \cdot L}$ where $L < 2^{64}$,

Salasa and ChaCha

- Outcome of a project called eStream that was concluded in 2008.
- Designed by Dan Bernstein in 2005.
- Defined as:

$$G : \mathcal{S} \times N \longrightarrow \mathcal{R}, \text{ where}$$

  - $\mathcal{S} = \{0,1\}^{128 \text{ or } 256}$
  - $\mathcal{R} = \{0,1\}^{512 \cdot L}$ where $L < 2^{64}$, and
  - $N = \{0,1\}^{64}$ Nonce.

Salasa and ChaCha

- Outcome of a project called eStream that was concluded in 2008.
- Designed by Dan Bernstein in 2005.
- Defined as:

$$G : \mathcal{S} \times N \longrightarrow \mathcal{R}, \text{ where}$$

  - $\mathcal{S} = \{0,1\}^{128 \text{ or } 256}$
  - $\mathcal{R} = \{0,1\}^{512 \cdot L}$ where $L < 2^{64}$, and
  - $N = \{0,1\}^{64}$ Nonce.

- The key-nonce pair (k, n) must not be repeated.

Salasa and ChaCha

- Outcome of a project called eStream that was concluded in 2008.
- Designed by Dan Bernstein in 2005.
- Defined as:

$$G : \mathcal{S} \times N \longrightarrow \mathcal{R}, \text{ where}$$

  - $\mathcal{S} = \{0,1\}^{128 \text{ or } 256}$
  - $\mathcal{R} = \{0,1\}^{512 \cdot L}$ where $L < 2^{64}$, and
  - $N = \{0,1\}^{64}$ Nonce.
- The key-nonce pair (k, n) must not be repeated.
- The key can be repeated with different nonce.

Salasa20

$$G(k,n) = \Pi(k,(0,n))||\Pi(k,(1,n))||\Pi(k,(2,n))||\cdots$$

Salasa20

$$G(k,n) = \Pi(k,(0,n))||\Pi(k,(1,n))||\Pi(k,(2,n))||\cdots$$

- Key $k \in \{0,1\}^{256}$,
  - $k_0, k_1, \ldots, k_7 \in \{0,1\}^{32}$

Salasa20

$$G(k,n) = \Pi(k,(0,n))||\Pi(k,(1,n))||\Pi(k,(2,n))||\cdots$$

- Key $k \in \{0,1\}^{256}$,
  - $k_0, k_1, \ldots, k_7 \in \{0,1\}^{32}$
- Nonce $n \in \{0,1\}^{64}$,
  - $n_0, n_1 \in \{0,1\}^{32}$, and

Salasa20

$$G(k,n) = \Pi(k,(0,n))||\Pi(k,(1,n))||\Pi(k,(2,n))||\cdots$$

- Key $k \in \{0,1\}^{256}$,
  - $k_0, k_1, \ldots, k_7 \in \{0,1\}^{32}$
- Nonce $n \in \{0,1\}^{64}$,
  - $n_0, n_1 \in \{0,1\}^{32}$, and
- Counter $i \in \{0,1\}^{64}$
  - $i_0, i_1 \in \{0,1\}^{32}$.

padding(k, i, n)

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ i_0 & i_1 & n_0 & n_1 \end{pmatrix}$$

padding(k, i, n)

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ i_0 & i_1 & n_0 & n_1 \end{pmatrix} \rightarrow \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & n_0 & n_1 \\ i_0 & i_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}$$

padding(k, i, n)

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ i_0 & i_1 & n_0 & n_1 \end{pmatrix} \rightarrow \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & n_0 & n_1 \\ i_0 & i_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix} \rightarrow \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \Phi$$

padding(k, i, n)

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ i_0 & i_1 & n_0 & n_1 \end{pmatrix} \rightarrow \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & n_0 & n_1 \\ i_0 & i_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix} \rightarrow \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \Phi$$

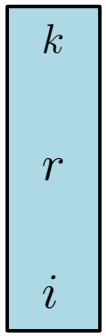
- $c_0, c_1, c_2, c_4 \in \{0,1\}^{32}$ are some specific constant.

padding(k, i, n)

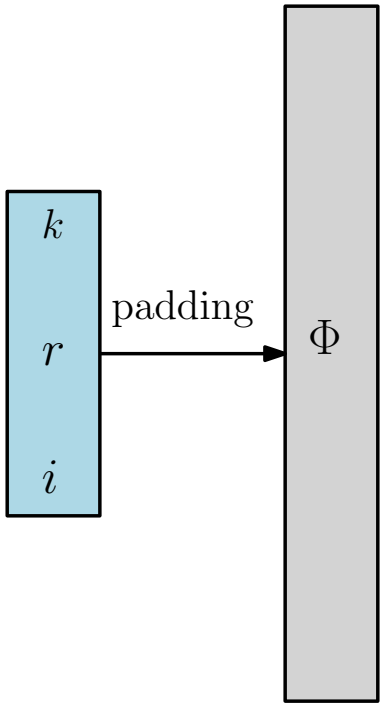
$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ i_0 & i_1 & n_0 & n_1 \end{pmatrix} \rightarrow \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & n_0 & n_1 \\ i_0 & i_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix} \rightarrow \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \Phi$$

- $c_0, c_1, c_2, c_4 \in \{0,1\}^{32}$ are some specific constant.
- Let $\pi : \{0,1\}^{512} \longrightarrow \{0,1\}^{512}$ an invertible function.

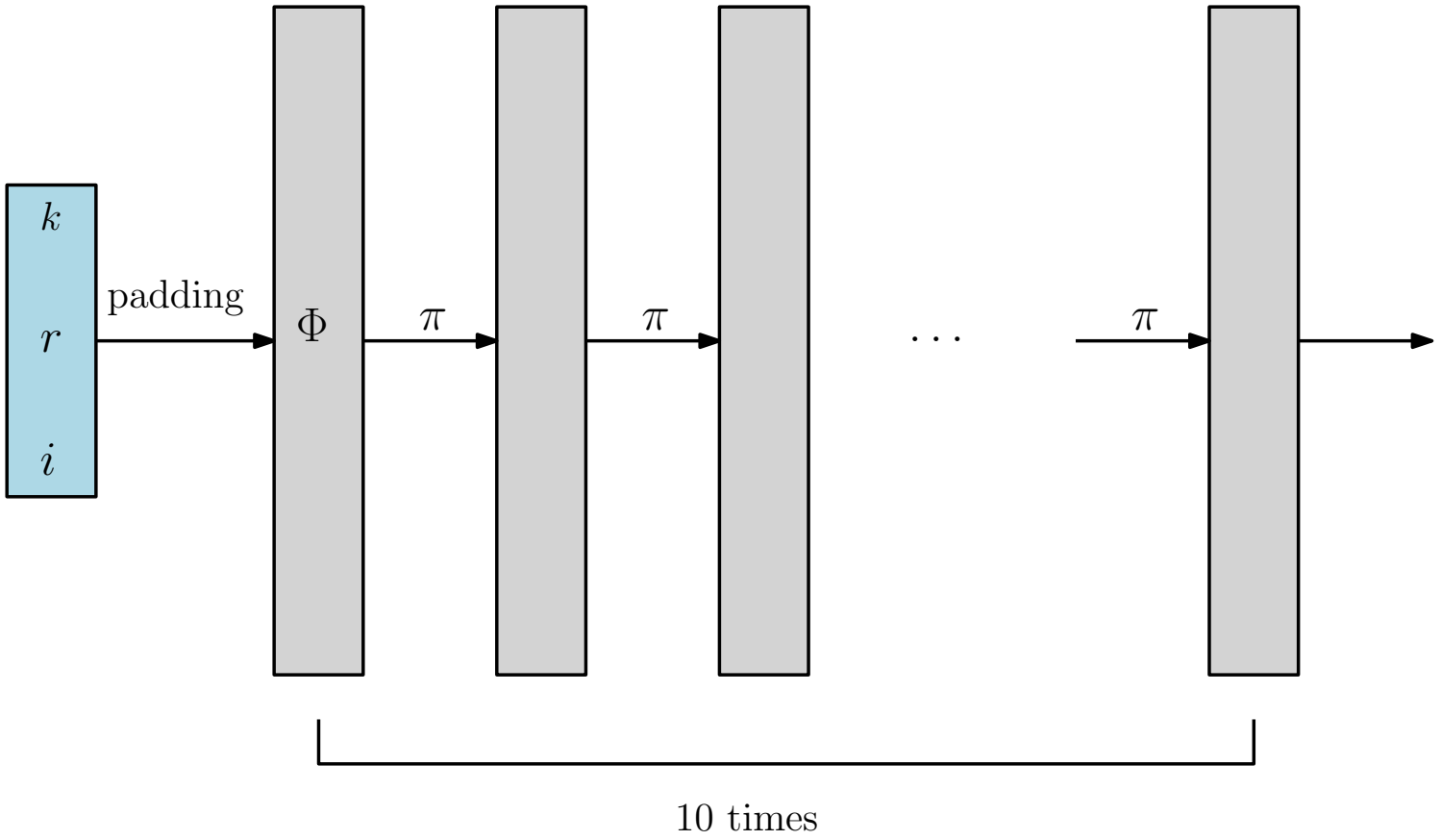
Salasa20



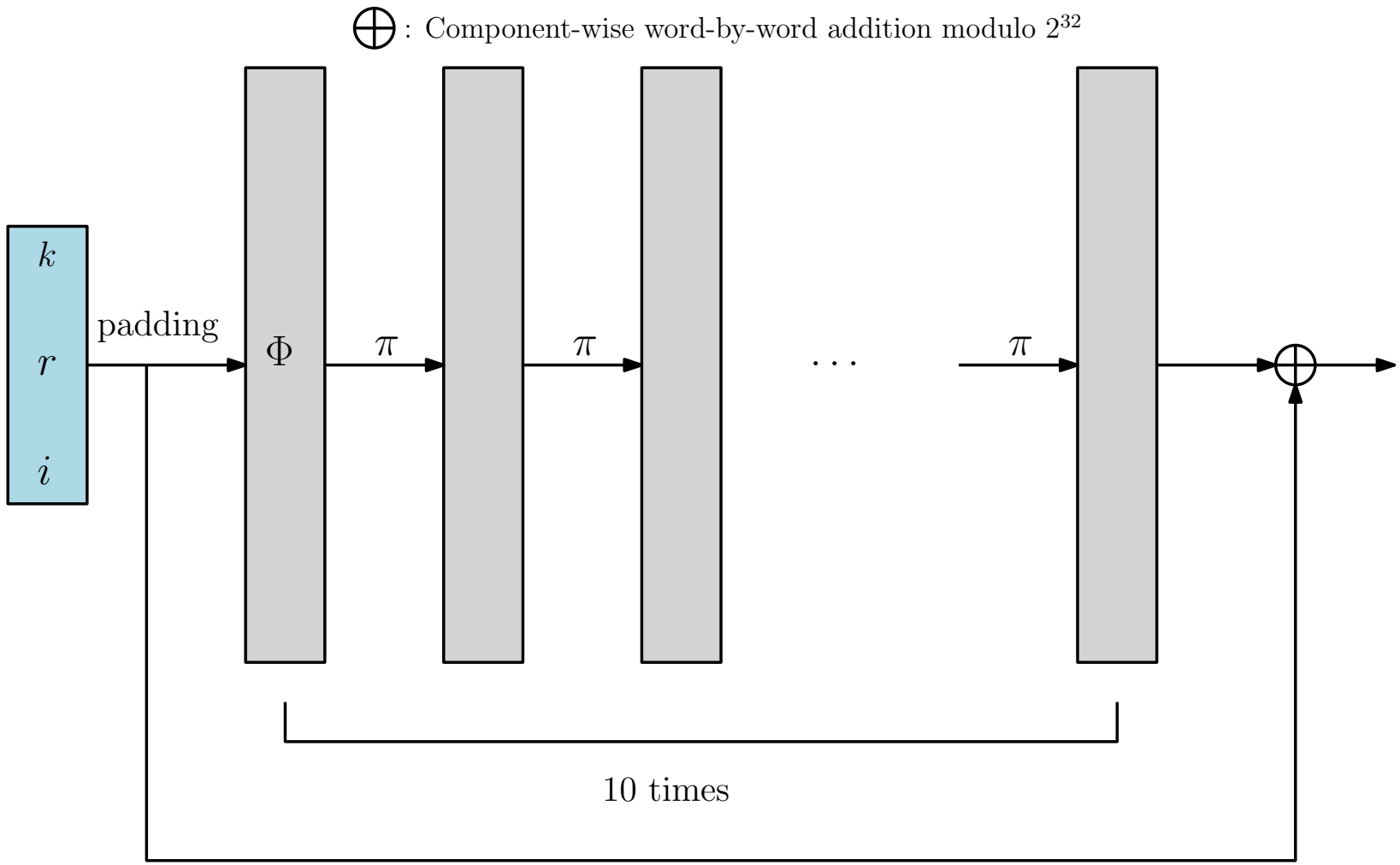
Salasa20



Salasa20



Salasa20


⊕ : Component-wise word-by-word addition modulo 2^{32}
k
r
i
padding
Φ
π
π
$\cdots$
π
10 times

π

1. QuarterRound(x_0, x_4, x_8, x_{12})
2. QuarterRound(x_1, x_5, x_9, x_{13})
3. QuarterRound$(x_2, x_6, x_{10}, x_{14})$
4. QuarterRound$(x_3, x_7, x_{11}, x_{15})$
5. QuarterRound$(x_0, x_5, x_{10}, x_{15})$
6. QuarterRound$(x_1, x_6, x_{11}, x_{12})$
7. QuarterRound(x_2, x_7, x_8, x_{13})
8. QuarterRound(x_3, x_4, x_9, x_{14})

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix}$$

QuarterRound(a, b, c, d)

#define ROTL(a, b) ($((a) \ll (b))$ | $((a) \gg (32 - (b)))$)

QuarterRound(a, b, c, d)

#define ROTL(a, b) ($((a) \ll (b))$ | $((a) \gg (32 - (b)))$)

$$
\begin{array}{lll}
a+ = b; & d\hat{} = a; & \text{ROTL}(d, 16); \\
c+ = d; & b\hat{} = c; & \text{ROTL}(b, 12); \\
a+ = b; & d\hat{} = a; & \text{ROTL}(d, 8); \\
c+ = d; & b\hat{} = c; & \text{ROTL}(b, 7);
\end{array}
$$

QuarterRound(a, b, c, d)

#define ROTL(a, b) ($((a) \ll (b))$ | $((a) \gg (32 - (b)))$)

$$
\begin{array}{lll}
a+ = b; & d\hat{} = a; & \text{ROTL}(d, 16); \\
c+ = d; & b\hat{} = c; & \text{ROTL}(b, 12); \\
a+ = b; & d\hat{} = a; & \text{ROTL}(d, 8); \\
c+ = d; & b\hat{} = c; & \text{ROTL}(b, 7);
\end{array}
$$

Advantage

- Highly parallelizable.
- Performs good in hardware or software implementation.

End