

Cryptography

Lecture 4: Pseudorandom Generators

November 6, 2026

Next Dates

LECTURE (FRI 1:15PM)	
Nov 6th	Lecture 4: Pseudorandom Generators
13th	Lecture 5: Multiple Encryptions
20th	Lecture 6: Pseudorandom Functions
27th	Lecture 7: Operation Modes
Dec 4th	Lecture 8: Message Authentication Codes
11th	Lecture 9: Authenticated Encryption
18th	Lecture 10: Hash functions

LECTURE (FRI 1:15PM)	
Dec 25th	
Jan 1st	
8th	Lecture 11: RSA & Diffie-Hellman assumptions
15th	Lecture 12: Elliptic curves
22nd	Lecture 13: Digital signatures
29th	Lecture 14: Zero-Knowledge Proofs
Feb 5th	Lecture 15: Reserve

Contents

- 1 Classical cryptography
(Shift & Vigenère cipher, one-time pad, perfect secrecy)
- 2 **Security definitions & threat models**
(Computational security, CPA & CCA)
- 3 Private-key cryptography
(Message authentication, hash functions, primitives, relevant ciphers)
- 4 Public-key cryptography
(Assumptions, key management, digital signatures, relevant ciphers)

Practical Considerations

Perfect secrecy

- Unbreakable if used correctly
- Expensive due to huge key space & required key exchanges

Practical Considerations

Perfect secrecy

- Unbreakable if used correctly
- Expensive due to huge key space & required key exchanges
- Only feasible for extremely secret information

Practical Considerations

Perfect secrecy

- Unbreakable if used correctly
- Expensive due to huge key space & required key exchanges
- Only feasible for extremely secret information

Practical secrecy

- Allow key reuse
- Weaken requirements
 - ▶ Perfect correctness
 - ▶ Threat model
 - ▶ Perfect secrecy

Practical Considerations

Perfect secrecy

- Unbreakable if used correctly
- Expensive due to huge key space & required key exchanges
- Only feasible for extremely secret information

Practical secrecy

- Allow key reuse
- Weaken requirements
 - ▶ Perfect correctness
 - ▶ Threat model
 - ▶ Perfect secrecy

Practical Considerations

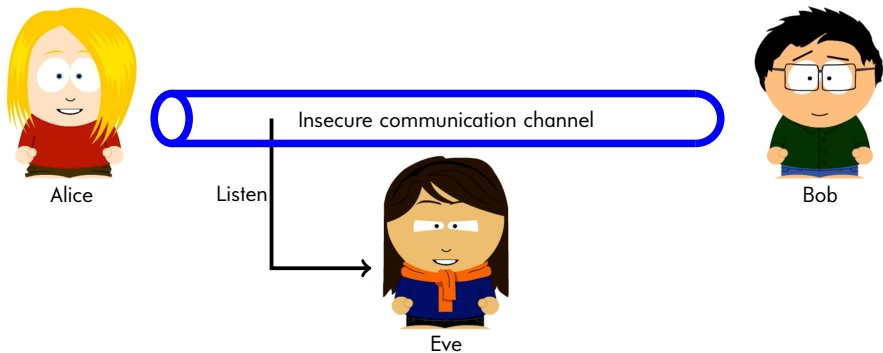
Perfect secrecy

- Unbreakable if used correctly
- Expensive due to huge key space & required key exchanges
- Only feasible for extremely secret information

Practical secrecy

- Allow key reuse
- Weaken requirements
 - ▶ Perfect correctness
 - ▶ Threat model
 - ▶ Perfect secrecy
- Allow control over security

Threat Models



Threat model = What can Eve do?

Threat Models

Goal of attacker

Decrypt current ciphertext

Encryption threat models (in increasing power)

- ❶ **Ciphertext-only attack** = Eve only collects ciphertexts
but has no access to additional information (eavesdropper attack)
(we essentially used this model so far)

Threat Models

Goal of attacker

Decrypt current ciphertext

Encryption threat models (in increasing power)

- 1 **Ciphertext-only attack** = Eve only collects ciphertexts
but has no access to additional information (eavesdropper attack)
(we essentially used this model so far)
- 2 **Known-plaintext attack** = Eve also knows some plaintext-ciphertext
pairs for same key (informed guesses often possible)

Threat Models

Goal of attacker

Decrypt current ciphertext

Encryption threat models (in increasing power)

- 1 **Ciphertext-only attack** = Eve only collects ciphertexts
but has no access to additional information (eavesdropper attack)
(we essentially used this model so far)
- 2 **Known-plaintext attack** = Eve also knows some plaintext-ciphertext
pairs for same key (informed guesses often possible)
- 3 **Chosen-plaintext attack** = Eve knows plaintext-ciphertext pairs for
plaintexts of her choice (Alice encrypts messages of Eve's choice)

Threat Models

Goal of attacker

Decrypt current ciphertext

Encryption threat models (in increasing power)

- ① **Ciphertext-only attack** = Eve only collects ciphertexts but has no access to additional information (eavesdropper attack)
(we essentially used this model so far)
- ② **Known-plaintext attack** = Eve also knows some plaintext-ciphertext pairs for same key (informed guesses often possible)
- ③ **Chosen-plaintext attack** = Eve knows plaintext-ciphertext pairs for plaintexts of her choice (Alice encrypts messages of Eve's choice)
- ④ **Chosen-ciphertext attack** = Eve also knows plaintext-ciphertext pairs for ciphertexts of her choice (Bob decrypts ciphertexts of Eve's choice)

Threat Models

Computational capabilities of attacker

- **Unrestricted** (used so far) (anything goes)
- **Computable** (any algorithm)
- **Efficient** (i.e. polynomial-time) (any polynomial-time algorithm)

Threat Models

Computational capabilities of attacker

- **Unrestricted** (used so far) (anything goes)
- **Computable** (any algorithm)
- Efficient (i.e. polynomial-time) (any polynomial-time algorithm)

Notes

- Key space usually exponential size in key length (2^n keys of length n)
- Full brute-force attack takes exponential-time in key length
(assuming efficient decryption & recognition of valid messages)

Threat Models

Computational capabilities of attacker

- **Unrestricted** (used so far) (anything goes)
- **Computable** (any algorithm)
- Efficient (i.e. polynomial-time) (any polynomial-time algorithm)

Notes

- Key space usually exponential size in key length (2^n keys of length n)
- Full brute-force attack takes exponential-time in key length
(assuming efficient decryption & recognition of valid messages)
- Partial brute-force attack possible in polynomial-time
(under same assumptions)

→ Weakening of perfect secrecy also necessary

Efficient Encryption Scheme

§3.6 Definition (cf. §2.4; Efficient private-key encryption scheme)

(Efficient) private-key encryption scheme is triple $(\text{gen}, \text{enc}, \text{dec})$ of probabilistic polynomial-time computable functions

- $\text{gen}: R \times \{1\}^* \rightarrow K$ i.e. family $(P_K^n)_{n \in \mathbb{N}}$ of distributions $P_K^n: K \rightarrow [0, 1]$
- $\text{enc}: R \times K \times M \rightarrow C$ and $\text{dec}: K \times C \rightarrow (M \cup \{\perp\})$

Efficient Encryption Scheme

§3.6 Definition (cf. §2.4; Efficient private-key encryption scheme)

(Efficient) private-key encryption scheme is triple $(\text{gen}, \text{enc}, \text{dec})$ of probabilistic polynomial-time computable functions

- $\text{gen}: R \times \{1\}^* \rightarrow K$ i.e. family $(P_K^n)_{n \in \mathbb{N}}$ of distributions $P_K^n: K \rightarrow [0, 1]$
- $\text{enc}: R \times K \times M \rightarrow C$ and $\text{dec}: K \times C \rightarrow (M \cup \{\perp\})$
- $|\text{gen}(r, 1^n)| \geq n$ for all $r \in R$ and $n \in \mathbb{N}$
- $\text{dec}_k(c) = m$ for all $k \in \text{ran}(\text{gen})$, $m \in M$, and $c \in C$ with $P[c \leftarrow \text{enc}_k(m)] \neq 0$ (perfect correctness)

Efficient Encryption Scheme

§3.6 Definition (cf. §2.4; Efficient private-key encryption scheme)

(Efficient) **private-key encryption scheme** is triple $(\text{gen}, \text{enc}, \text{dec})$ of probabilistic polynomial-time computable functions

- $\text{gen}: R \times \{1\}^* \rightarrow K$ i.e. family $(P_K^n)_{n \in \mathbb{N}}$ of distributions $P_K^n: K \rightarrow [0, 1]$
- $\text{enc}: R \times K \times M \rightarrow C$ and $\text{dec}: K \times C \rightarrow (M \cup \{\perp\})$
- $|\text{gen}(r, 1^n)| \geq n$ for all $r \in R$ and $n \in \mathbb{N}$
- $\text{dec}_k(c) = m$ for all $k \in \text{ran}(\text{gen})$, $m \in M$, and $c \in C$ with $P[c \leftarrow \text{enc}_k(m)] \neq 0$ (perfect correctness)

Notes

- **Security parameter** n = additional parameter of gen
- Generated keys have at least length n
- Decrypt failure indicated by $\perp$

Efficient Encryption Scheme

Example: Vigenère cipher

- $P_K^n(k) = 2^{-n}$ for all $k \in \{0,1\}^n$ and $P_K^n(k) = 0$ otherwise
(uniform random selection of key of length n)

Efficient Encryption Scheme

Example: Vigenère cipher

- $P_K^n(k) = 2^{-n}$ for all $k \in \{0, 1\}^n$ and $P_K^n(k) = 0$ otherwise
(uniform random selection of key of length n)
- $\text{enc}_k(m_1, \dots, m_\ell) = (m_1 \oplus k_{0 \bmod n}, \dots, m_\ell \oplus k_{(\ell-1) \bmod n})$
with key $k = (k_0, \dots, k_{n-1})$
- $\text{dec}_k = \text{enc}_k$
- These functions are certainly polynomial-time computable

Efficient Encryption Scheme

Example: Vigenère cipher

- $P_K^n(k) = 2^{-n}$ for all $k \in \{0, 1\}^n$ and $P_K^n(k) = 0$ otherwise
(uniform random selection of key of length n)
- $\text{enc}_k(m_1, \dots, m_\ell) = (m_1 \oplus k_{0 \bmod n}, \dots, m_\ell \oplus k_{(\ell-1) \bmod n})$
with key $k = (k_0, \dots, k_{n-1})$
- $\text{dec}_k = \text{enc}_k$
- These functions are certainly polynomial-time computable

Note

- Shift cipher offers no obvious security parameter

Efficient Encryption Scheme

Notes

- Asymptotic approach (polynomial time $\neq$ quick)
(polynomial time
= eventually small complexity increase for small input size increase)
- PPT = probabilistic polynomial-time (computable)
- Also allow eventually small advantage of attacker
→ negligible functions

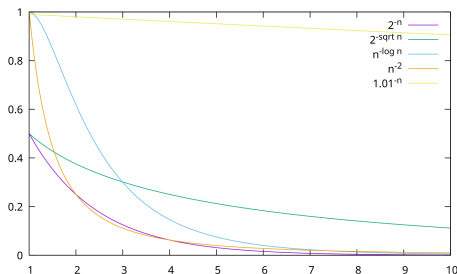
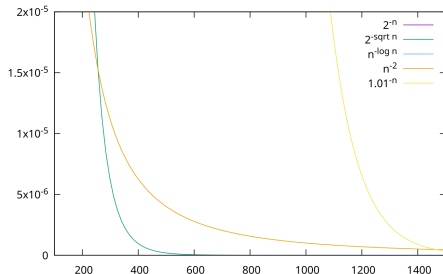
§3.7 Definition (Negligible function)

Function $f: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is **negligible** if for every positive polynomial p there exists $n_0 \in \mathbb{N}$ such that $f(n) < p(n)^{-1}$ for all $n \in \mathbb{N}$ with $n \geq n_0$

Computational Indistinguishability

§3.7 Definition (Negligible function)

Function $f: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is **negligible** if for every positive polynomial p there exists $n_0 \in \mathbb{N}$ such that $f(n) < p(n)^{-1}$ for all $n \in \mathbb{N}$ with $n \geq n_0$



Notation

- We frequently use sequence notation for functions with domain $\mathbb{N}$ i.e. $(2^{-x})_{x \in \mathbb{N}}$ for function $f: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ with $f(x) = 2^{-x}$

Efficient Encryption Scheme

Questions

- Which of those functions are negligible?
 - ▶ $(2^{-n})_{n \in \mathbb{N}}$
 - ▶ $(2^{-\sqrt{n}})_{n \in \mathbb{N}}$
 - ▶ $(n^{-\log n})_{n \in \mathbb{N}}$
 - ▶ $(n^{-2})_{n \in \mathbb{N}}$
 - ▶ $(1.01^{-n})_{n \in \mathbb{N}}$

Efficient Encryption Scheme

Questions

- Which of those functions are negligible?
 - ▶ $(2^{-n})_{n \in \mathbb{N}}$
 - ▶ $(2^{-\sqrt{n}})_{n \in \mathbb{N}}$
 - ▶ $(n^{-\log n})_{n \in \mathbb{N}}$
 - ▶ $(n^{-2})_{n \in \mathbb{N}}$
 - ▶ $(1.01^{-n})_{n \in \mathbb{N}}$
- Is the sum/product of 2 negligible functions negligible?

Efficient Encryption Scheme

Questions

- Which of those functions are negligible?
 - ▶ $(2^{-n})_{n \in \mathbb{N}}$
 - ▶ $(2^{-\sqrt{n}})_{n \in \mathbb{N}}$
 - ▶ $(n^{-\log n})_{n \in \mathbb{N}}$
 - ▶ $(n^{-2})_{n \in \mathbb{N}}$
 - ▶ $(1.01^{-n})_{n \in \mathbb{N}}$
- Is the sum/product of 2 negligible functions negligible?
- Is the sum/product of negligible function with any function negligible?

§4.1 Definition (Almost bound & almost equality)

Function $f: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is **(asymptotically) almost bounded** by $b \in \mathbb{R}_{\geq 0}$, written $f \preceq b$, if there exists negligible function $\tilde{O}: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ such that $f(n) \leq b + \tilde{O}(n)$ for all $n \in \mathbb{N}$.

§4.1 Definition (Almost bound & almost equality)

Function $f: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is (asymptotically) almost bounded by $b \in \mathbb{R}_{\geq 0}$, written $f \preceq b$, if there exists negligible function $\tilde{0}: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ such that $f(n) \leq b + \tilde{0}(n)$ for all $n \in \mathbb{N}$.

Functions $f, g: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ are (asymptotically) almost equal, written $f \simeq g$, if

$$(|f(n) - g(n)|)_{n \in \mathbb{N}} \preceq 0$$

§4.1 Definition (Almost bound & almost equality)

Function $f: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is (asymptotically) almost bounded by $b \in \mathbb{R}_{\geq 0}$, written $f \preceq b$, if there exists negligible function $\tilde{0}: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ such that $f(n) \leq b + \tilde{0}(n)$ for all $n \in \mathbb{N}$.

Functions $f, g: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ are (asymptotically) almost equal, written $f \simeq g$, if

$$(|f(n) - g(n)|)_{n \in \mathbb{N}} \preceq 0$$

Notes

- We generally use $\tilde{0}$ for negligible functions
- $h \preceq 0$ iff h is negligible
- $f \simeq g$ iff $|f - g|$ is negligible (absolute value of pointwise difference)

Computational Indistinguishability

§4.2 Definition (Indistinguishability game)

Let $n \in \mathbb{N}$, $\mathcal{E} = (\text{gen}, \text{enc}, \text{dec})$ efficient private-key encryption scheme and $\mathcal{A}$ stateful algorithm. **Adversarial indistinguishability game** $\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)$ is

- ❶ Security parameter 1^n sent to adversary $\mathcal{A}$
 - $\mathcal{A}$ selects messages $m_0, m_1 \in \mathcal{M}$ with $|m_0| = |m_1|$ ❷
- ❸ $k \leftarrow \text{gen}(1^n)$, uniform random bit $b \in \{0, 1\}$
 $c \leftarrow \text{enc}_k(m_b)$ sent to $\mathcal{A}$ (challenge ciphertext)
 - $\mathcal{A}$ outputs bit $b' \in \{0, 1\}$ ❹
- ❺ Return 1 if $b = b'$ ($\mathcal{A}$ wins)
Return 0 otherwise

Computational Indistinguishability

§4.2 Definition (Indistinguishability game)

Let $n \in \mathbb{N}$, $\mathcal{E} = (\text{gen}, \text{enc}, \text{dec})$ efficient private-key encryption scheme and $\mathcal{A}$ stateful algorithm. **Adversarial indistinguishability game** $\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)$ is

- ❶ Security parameter 1^n sent to adversary $\mathcal{A}$
 $\mathcal{A}$ selects messages $m_0, m_1 \in \mathcal{M}$ with $|m_0| = |m_1|$ ❷
- ❸ $k \leftarrow \text{gen}(1^n)$, uniform random bit $b \in \{0, 1\}$
 $c \leftarrow \text{enc}_k(m_b)$ sent to $\mathcal{A}$ (challenge ciphertext)
 $\mathcal{A}$ outputs bit $b' \in \{0, 1\}$ ❹
- ❺ Return 1 if $b = b'$ ($\mathcal{A}$ wins)
Return 0 otherwise

Changes

- $\mathcal{A}$ receives security parameter n & selects messages of equal length
- Key generation with security parameter

Computational Indistinguishability



Alice



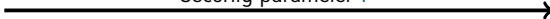
Eve

Computational Indistinguishability



Alice

Security parameter 1^n

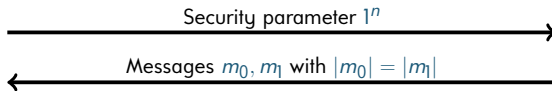


Eve

Computational Indistinguishability



Alice



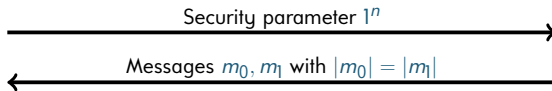
Eve

Computational Indistinguishability



Alice

$k \leftarrow \text{gen}(1^n)$
random b



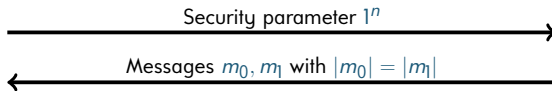
Eve

Computational Indistinguishability



Alice

$k \leftarrow \text{gen}(1^n)$
random b
 $c \leftarrow \text{enc}_k(m_b)$



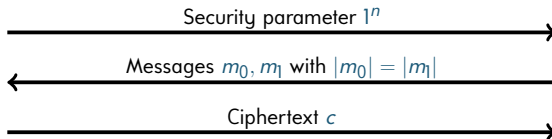
Eve

Computational Indistinguishability



Alice

$k \leftarrow \text{gen}(1^n)$
random b
 $c \leftarrow \text{enc}_k(m_b)$



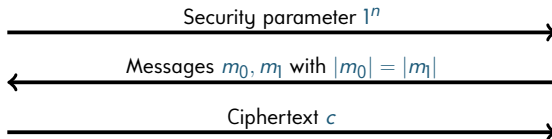
Eve

Computational Indistinguishability



Alice

$k \leftarrow \text{gen}(1^n)$
random b
 $c \leftarrow \text{enc}_k(m_b)$



Eve

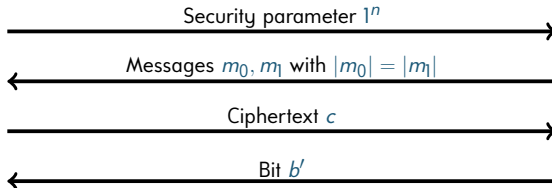
Determines b'

Computational Indistinguishability



Alice

$k \leftarrow \text{gen}(1^n)$
random b
 $c \leftarrow \text{enc}_k(m_b)$



Eve

Determines b'

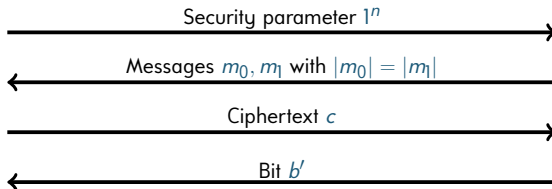
Computational Indistinguishability



Alice

$k \leftarrow \text{gen}(1^n)$
random b
 $c \leftarrow \text{enc}_k(m_b)$

Return $b \stackrel{?}{=} b'$



Eve

Determines b'

§4.3 Definition (EAV-secure)

Private-key encryption scheme $\mathcal{E} = (\text{gen}, \text{enc}, \text{dec})$ is **EAV-secure** if for every PPT algorithm $\mathcal{A}$ (in security parameter)

$$\left(\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \right)_{n \in \mathbb{N}} \preceq \frac{1}{2}$$

§4.3 Definition (EAV-secure)

Private-key encryption scheme $\mathcal{E} = (\text{gen}, \text{enc}, \text{dec})$ is **EAV-secure** if for every PPT algorithm $\mathcal{A}$ (in security parameter)

$$\left(\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \right)_{n \in \mathbb{N}} \preceq \frac{1}{2}$$

Notes

- $\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)]$ = probability that $\mathcal{A}$ wins for security parameter n

§4.3 Definition (EAV-secure)

Private-key encryption scheme $\mathcal{E} = (\text{gen}, \text{enc}, \text{dec})$ is **EAV-secure** if for every PPT algorithm $\mathcal{A}$ (in security parameter)

$$\left(\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \right)_{n \in \mathbb{N}} \preceq \frac{1}{2}$$

Notes

- $\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)]$ = probability that $\mathcal{A}$ wins for security parameter n
- EAV-secure = no PPT algorithm $\mathcal{A}$ can do **much** better asymptotically
- Can replace $\left(\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \right)_{n \in \mathbb{N}} \preceq \frac{1}{2}$ by $\left(\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \right)_{n \in \mathbb{N}} \simeq \frac{1}{2}$?

§4.3 Definition (EAV-secure)

Private-key encryption scheme $\mathcal{E} = (\text{gen}, \text{enc}, \text{dec})$ is **EAV-secure** if for every PPT algorithm $\mathcal{A}$ (in security parameter)

$$\left(\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \right)_{n \in \mathbb{N}} \preceq \frac{1}{2}$$

Notes

- $\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)]$ = probability that $\mathcal{A}$ wins for security parameter n
- EAV-secure = no PPT algorithm $\mathcal{A}$ can do **much** better asymptotically
- Can replace $\left(\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \right)_{n \in \mathbb{N}} \preceq \frac{1}{2}$ by $\left(\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \right)_{n \in \mathbb{N}} \simeq \frac{1}{2}$?
- **Careful:** no concrete guarantees

Computational Indistinguishability

Example: Vigenère cipher $\mathcal{E}$

- Adversary $\mathcal{A}$ receives 1^n and selects messages $m_0 = a^n a^n$ and $m_1 = a^n b^n$ of equal length
- Adversary $\mathcal{A}$ receives ciphertext $c = c_1 c_2$ with $|c_1| = n = |c_2|$
 - ▶ Returns 0 if $c_1 = c_2$
 - ▶ Returns 1 otherwise
- Adversary $\mathcal{A}$ obviously PPT

Computational Indistinguishability

Example: Vigenère cipher $\mathcal{E}$

- Adversary $\mathcal{A}$ receives 1^n and selects messages $m_0 = a^n a^n$ and $m_1 = a^n b^n$ of equal length
- Adversary $\mathcal{A}$ receives ciphertext $c = c_1 c_2$ with $|c_1| = n = |c_2|$
 - ▶ Returns 0 if $c_1 = c_2$
 - ▶ Returns 1 otherwise
- Adversary $\mathcal{A}$ obviously PPT

$$\begin{aligned} & \mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \\ &= \frac{1}{2} \mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n) \mid b = 0] + \frac{1}{2} \mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n) \mid b = 1] \end{aligned}$$

Computational Indistinguishability

Example: Vigenère cipher $\mathcal{E}$

- Adversary $\mathcal{A}$ receives 1^n and selects messages $m_0 = a^n a^n$ and $m_1 = a^n b^n$ of equal length
- Adversary $\mathcal{A}$ receives ciphertext $c = c_1 c_2$ with $|c_1| = n = |c_2|$
 - ▶ Returns 0 if $c_1 = c_2$
 - ▶ Returns 1 otherwise
- Adversary $\mathcal{A}$ obviously PPT

$$\begin{aligned} & \mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \\ &= \frac{1}{2} \mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n) \mid b = 0] + \frac{1}{2} \mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n) \mid b = 1] \\ &= \frac{1}{2} \mathbb{P}[c_1 = c_2 \mid b = 0] + \frac{1}{2} \mathbb{P}[c_1 \neq c_2 \mid b = 1] \\ &= \frac{1}{2} + \frac{1}{2} = 1 \not\approx \frac{1}{2} \end{aligned}$$

Computational Indistinguishability

Example: Vigenère cipher $\mathcal{E}$

- Adversary $\mathcal{A}$ receives 1^n and selects messages $m_0 = a^n a^n$ and $m_1 = a^n b^n$ of equal length
- Adversary $\mathcal{A}$ receives ciphertext $c = c_1 c_2$ with $|c_1| = n = |c_2|$
 - ▶ Returns 0 if $c_1 = c_2$
 - ▶ Returns 1 otherwise
- Adversary $\mathcal{A}$ obviously PPT

$$\begin{aligned} & \mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \\ &= \frac{1}{2} \mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n) \mid b = 0] + \frac{1}{2} \mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n) \mid b = 1] \\ &= \frac{1}{2} \mathbb{P}[c_1 = c_2 \mid b = 0] + \frac{1}{2} \mathbb{P}[c_1 \neq c_2 \mid b = 1] \\ &= \frac{1}{2} + \frac{1}{2} = 1 \not\approx \frac{1}{2} \end{aligned}$$

→ Vigenère cipher not EAV-secure

Question: Which is uniformly random 32 bit string?

- 0101 0101 0101 0101 0101 0101 0101 0101
- 1010 0101 1010 1001 1110 1011 0110 1101
- 0000 0000 0000 0000 0000 0000 0000 0000

Question: Which is uniformly random 32 bit string?

- 0101 0101 0101 0101 0101 0101 0101 0101
- 1010 0101 1010 1001 1110 1011 0110 1101
- 0000 0000 0000 0000 0000 0000 0000 0000

Answer

- Each probability 2^{-32} if generated uniformly
- Uniform randomness is property of distribution (not drawn strings)
- **Uniform distribution** U_n with $U_n(b) = 2^{-n}$ for all $b \in \{0,1\}^n$

Remarks

- We accept casual terminology
uniformly random $b = b$ drawn from uniformly random distribution
- Given distribution D we write $d \leftarrow D$ for d sampled from D

Remarks

- We accept casual terminology
uniformly random $b = b$ drawn from uniformly random distribution
- Given distribution D we write $d \leftarrow D$ for d sampled from D
- Instead of “uniform random bit $b \in \{0, 1\}$ ” we write $b \leftarrow \mathcal{U}_1$
- We agree on **universal security parameter** $n \in \mathbb{N}$
and omit $(\cdot)_{n \in \mathbb{N}}$ whenever context demands sequence

$$\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \preceq \frac{1}{2} \quad \text{instead of} \quad \left(\mathbb{P}[\text{PrivK}_{\mathcal{E}, \mathcal{A}}^{\text{one}}(n)] \right)_{n \in \mathbb{N}} \preceq \frac{1}{2}$$

Pseudorandom Generator

§4.4 Definition (Pseudorandom generator)

Let p be polynomial and G deterministic polynomial-time algorithm such that $G(s) \in \{0,1\}^{p(n)}$ for all $s \in \{0,1\}^n$ and $n \in \mathbb{N}$.

G is **pseudorandom generator** if

- $p(n) > n$ for all $n \geq 1$ (expansive)
- For any PPT algorithm $D: \{0,1\}^* \rightarrow \{0,1\}$

$$\sum_{s \in \{0,1\}^n} \frac{P(D(G(s)))}{2^n} \simeq \sum_{r \in \{0,1\}^{p(n)}} \frac{P(D(r))}{2^{p(n)}}$$

Pseudorandom Generator

§4.4 Definition (Pseudorandom generator)

Let p be polynomial and G deterministic polynomial-time algorithm such that $G(s) \in \{0,1\}^{p(n)}$ for all $s \in \{0,1\}^n$ and $n \in \mathbb{N}$.

G is **pseudorandom generator** if

- $p(n) > n$ for all $n \geq 1$ (expansive)
- For any PPT algorithm $D: \{0,1\}^* \rightarrow \{0,1\}$

$$\sum_{s \in \{0,1\}^n} \frac{P(D(G(s)))}{2^n} \simeq \sum_{r \in \{0,1\}^{p(n)}} \frac{P(D(r))}{2^{p(n)}}$$

Notes

- G extends short random string s into long pseudorandom string $G(s)$
- Long pseudorandom string $G(s)$ should be computationally indistinguishable from long, truly random string r

Pseudorandom Generator

$$\sum_{s \in \{0,1\}^n} \frac{P(D(G(s)))}{2^n}$$
$$= \mathbb{E} \left[P(D(G(s))) \right]_{s \leftarrow U_n}$$

$$\sum_{r \in \{0,1\}^{p(n)}} \frac{P(D(r))}{2^{p(n)}}$$
$$= \mathbb{E} \left[P(D(r)) \right]_{r \leftarrow U_{p(n)}}$$

We compare success probabilities of these 2 algorithms

① $s \leftarrow U_n$

② $r' = G(s)$

③ $D(r')$

① $r \leftarrow U_{p(n)}$

② $D(r)$

Example

- $G(b) = bb^R$ with polynomial $p(x) = 2x$
- Expansive & deterministically computable in polynomial time

Pseudorandom Generator

Example

- $G(b) = bb^R$ with polynomial $p(x) = 2x$
- Expansive & deterministically computable in polynomial time
- Stupid distinguisher D with input $d_1 \cdots d_{2n}$ for $d_1, \dots, d_{2n} \in \{0, 1\}$
 - 1 $i \leftarrow (n^{-1})_{a \in \{1, \dots, n\}}$ (draw i from uniform distribution on $\{1, \dots, n\}$)
 - 2 Return 1 if $d_i = d_{n+i}$; return 0 otherwise

Pseudorandom Generator

Example

- $G(b) = bb^R$ with polynomial $p(x) = 2x$
- Expansive & deterministically computable in polynomial time
- Stupid distinguisher D with input $d_1 \cdots d_{2n}$ for $d_1, \dots, d_{2n} \in \{0, 1\}$
 - 1 $i \leftarrow (n^{-1})_{a \in \{1, \dots, n\}}$ (draw i from uniform distribution on $\{1, \dots, n\}$)
 - 2 Return 1 if $d_i = d_{n+i}$; return 0 otherwise
- $r \leftarrow U_{2n}$; $D(r)$ succeeds with probability $\frac{1}{2}$ (compares two random bits)

Pseudorandom Generator

Example

- $G(b) = bb^R$ with polynomial $p(x) = 2x$
- Expansive & deterministically computable in polynomial time
- Stupid distinguisher D with input $d_1 \cdots d_{2n}$ for $d_1, \dots, d_{2n} \in \{0, 1\}$
 - 1 $i \leftarrow (n^{-1})_{a \in \{1, \dots, n\}}$ (draw i from uniform distribution on $\{1, \dots, n\}$)
 - 2 Return 1 if $d_i = d_{n+i}$; return 0 otherwise
- $r \leftarrow U_{2n}$; $D(r)$ succeeds with probability $\frac{1}{2}$ (compares two random bits)
- $s \leftarrow U_n$; $r' = G(s)$; $D(r')$ succeeds
 - ▶ with probability $\frac{1}{2}$ if n is even (compares two random bits)

Pseudorandom Generator

Example

- $G(b) = bb^R$ with polynomial $p(x) = 2x$
- Expansive & deterministically computable in polynomial time
- Stupid distinguisher D with input $d_1 \cdots d_{2n}$ for $d_1, \dots, d_{2n} \in \{0, 1\}$
 - 1 $i \leftarrow (n^{-1})_{a \in \{1, \dots, n\}}$ (draw i from uniform distribution on $\{1, \dots, n\}$)
 - 2 Return 1 if $d_i = d_{n+i}$; return 0 otherwise
- $r \leftarrow U_{2n}$; $D(r)$ succeeds with probability $\frac{1}{2}$ (compares two random bits)
- $s \leftarrow U_n$; $r' = G(s)$; $D(r')$ succeeds
 - ▶ with probability $\frac{1}{2}$ if n is even (compares two random bits)
 - ▶ with probability $\frac{1}{n} + \frac{(n-1)}{n} \cdot \frac{1}{2} = \frac{2+n-1}{2n} = \frac{n+1}{2n}$ if n is odd

Pseudorandom Generator

Example

- $G(b) = bb^R$ with polynomial $p(x) = 2x$
- Expansive & deterministically computable in polynomial time
- Stupid distinguisher D with input $d_1 \cdots d_{2n}$ for $d_1, \dots, d_{2n} \in \{0, 1\}$
 - 1 $i \leftarrow (n^{-1})_{a \in \{1, \dots, n\}}$ (draw i from uniform distribution on $\{1, \dots, n\}$)
 - 2 Return 1 if $d_i = d_{n+i}$; return 0 otherwise
- $r \leftarrow U_{2n}$; $D(r)$ succeeds with probability $\frac{1}{2}$ (compares two random bits)
- $s \leftarrow U_n$; $r' = G(s)$; $D(r')$ succeeds
 - ▶ with probability $\frac{1}{2}$ if n is even (compares two random bits)
 - ▶ with probability $\frac{1}{n} + \frac{(n-1)}{n} \cdot \frac{1}{2} = \frac{2+n-1}{2n} = \frac{n+1}{2n}$ if n is odd
- Hence $\frac{n+1}{2n} - \frac{1}{2} = \frac{n+1-n}{2n} = \frac{1}{2}n^{-1}$, which is **not** negligible

Pseudorandom Generator

Example

- $G(b) = bb^R$ with polynomial $p(x) = 2x$
- Expansive & deterministically computable in polynomial time
- Stupid distinguisher D with input $d_1 \cdots d_{2n}$ for $d_1, \dots, d_{2n} \in \{0, 1\}$
 - 1 $i \leftarrow (n^{-1})_{a \in \{1, \dots, n\}}$ (draw i from uniform distribution on $\{1, \dots, n\}$)
 - 2 Return 1 if $d_i = d_{n+i}$; return 0 otherwise
- $r \leftarrow U_{2n}$; $D(r)$ succeeds with probability $\frac{1}{2}$ (compares two random bits)
- $s \leftarrow U_n$; $r' = G(s)$; $D(r')$ succeeds
 - ▶ with probability $\frac{1}{2}$ if n is even (compares two random bits)
 - ▶ with probability $\frac{1}{n} + \frac{(n-1)}{n} \cdot \frac{1}{2} = \frac{2+n-1}{2n} = \frac{n+1}{2n}$ if n is odd
- Hence $\frac{n+1}{2n} - \frac{1}{2} = \frac{n+1-n}{2n} = \frac{1}{2}n^{-1}$, which is **not** negligible

→ G is **not** pseudorandom generator

Pseudorandom Generator

Example

- $G(b) = bb^R$ with polynomial $p(x) = 2x$
- Better distinguisher D' with input $d_1 \cdots d_{2n}$ for $d_1, \dots, d_{2n} \in \{0, 1\}$
 - 1 Return 1 if $d_1 = d_{2n}$
 - 2 Return 0 otherwise

Pseudorandom Generator

Example

- $G(b) = bb^R$ with polynomial $p(x) = 2x$
- Better distinguisher D' with input $d_1 \cdots d_{2n}$ for $d_1, \dots, d_{2n} \in \{0, 1\}$
 - 1 Return 1 if $d_1 = d_{2n}$
 - 2 Return 0 otherwise
- $r \leftarrow U_{2n}$; $D'(r)$ succeeds with probability $\frac{1}{2}$
- $s \leftarrow U_n$; $r' = G(s)$; $D'(r')$ always succeeds

→ G is **not** pseudorandom generator

Pseudorandom Generator

Why “pseudorandom”

- Expansion is $p(x) \geq x + 1$ for pseudorandom generator G
- Each “random” bit sequence of length $p(n)$ has probability $2^{-p(n)}$

Pseudorandom Generator

Why “pseudorandom”

- Expansion is $p(x) \geq x + 1$ for pseudorandom generator G
- Each “random” bit sequence of length $p(n)$ has probability $2^{-p(n)}$
- $|\{G(s) \mid s \in \{0,1\}^n\}| \leq 2^n$,
so at least $2^{p(n)} - 2^n \geq 2^n$ sequences impossible

Pseudorandom Generator

Why “pseudorandom”

- Expansion is $p(x) \geq x + 1$ for pseudorandom generator G
- Each “random” bit sequence of length $p(n)$ has probability $2^{-p(n)}$
- $|\{G(s) \mid s \in \{0,1\}^n\}| \leq 2^n$,
so at least $2^{p(n)} - 2^n \geq 2^n$ sequences impossible
- Universal distinguisher D with input $d \in \{0,1\}^{p(n)}$
 - 1 Return 1 if there exists $s \in \{0,1\}^n$ such that $G(s) = d$
 - 2 Return 0 otherwise

Pseudorandom Generator

Why “pseudorandom”

- Expansion is $p(x) \geq x + 1$ for pseudorandom generator G
- Each “random” bit sequence of length $p(n)$ has probability $2^{-p(n)}$
- $|\{G(s) \mid s \in \{0, 1\}^n\}| \leq 2^n$,
so at least $2^{p(n)} - 2^n \geq 2^n$ sequences impossible
- Universal distinguisher D with input $d \in \{0, 1\}^{p(n)}$
 - 1 Return 1 if there exists $s \in \{0, 1\}^n$ such that $G(s) = d$
 - 2 Return 0 otherwise
- $r \leftarrow U_{p(n)}$; $D(r)$ succeeds with probability at most $\frac{2^n}{2^{p(n)}} \leq \frac{1}{2}$

Pseudorandom Generator

Why “pseudorandom”

- Expansion is $p(x) \geq x + 1$ for pseudorandom generator G
- Each “random” bit sequence of length $p(n)$ has probability $2^{-p(n)}$
- $|\{G(s) \mid s \in \{0, 1\}^n\}| \leq 2^n$,
so at least $2^{p(n)} - 2^n \geq 2^n$ sequences impossible
- Universal distinguisher D with input $d \in \{0, 1\}^{p(n)}$
 - 1 Return 1 if there exists $s \in \{0, 1\}^n$ such that $G(s) = d$
 - 2 Return 0 otherwise
- $r \leftarrow U_{p(n)} ; D(r)$ succeeds with probability at most $\frac{2^n}{2^{p(n)}} \leq \frac{1}{2}$
- $s \leftarrow U_n ; r' = G(s) ; D(r')$ always succeeds

Pseudorandom Generator

Why “pseudorandom”

- Expansion is $p(x) \geq x + 1$ for pseudorandom generator G
- Each “random” bit sequence of length $p(n)$ has probability $2^{-p(n)}$
- $|\{G(s) \mid s \in \{0,1\}^n\}| \leq 2^n$,
so at least $2^{p(n)} - 2^n \geq 2^n$ sequences impossible
- Universal distinguisher D with input $d \in \{0,1\}^{p(n)}$
 - 1 Return 1 if there exists $s \in \{0,1\}^n$ such that $G(s) = d$
 - 2 Return 0 otherwise
- $r \leftarrow U_{p(n)}$; $D(r)$ succeeds with probability at most $\frac{2^n}{2^{p(n)}} \leq \frac{1}{2}$
- $s \leftarrow U_n$; $r' = G(s)$; $D(r')$ always succeeds
- Difference is **not** negligible, but D is **not** known to be polynomial time (brute-force attack against pseudorandom generator)

- Threat models
- Negligible functions
- Computational indistinguishability & EAV-security
- Pseudorandom generators