Project Verona — wikipedia

14 Jan 2025

Information theoretic security

Shannon Security

$\forall m, c$: $\Pr(\mathcal{P} = m \mid \mathcal{C} = c) = \Pr(\mathcal{P} = m)$

Equivalent

Perfect Indistinguishibility

- m_1, m_2 picked by Adv.
- one of them randomly encrypted by the challenger
- Adv's advantage in distinguishing which of the two messages produced c $= 0$

$\forall m_1, m_2, \forall c$

$\Pr(\mathcal{P} = m_1 \mid \mathcal{C} = c) = \Pr(\mathcal{P} = m_2 \mid \mathcal{C} = c)$

(I) Shannon security $\Rightarrow$ Perfect indistinguishability

$$\Pr(\mathcal{P} = m_1 \mid \mathcal{C} = c) = \Pr(\mathcal{P} = m_1) \quad \text{for any } m_1, c$$

defn of conditional prob

$$\frac{\Pr(\mathcal{P} = m_1 \cap \mathrm{Enc}(k, m_1) = c)}{\Pr(\mathcal{C} = c)} = \Pr(\mathcal{P} = m_1)$$

$$\frac{\cancel{\Pr(\mathcal{P} = m_1)} \cdot \Pr(\mathrm{Enc}(k, m_1) = c)}{\Pr(\mathcal{C} = c)} = \cancel{\Pr(\mathcal{P} = m_1)}$$

$$\Pr_{k,c}(\mathrm{Enc}(k, m_1) = c) = \Pr_{c}(\mathcal{C} = c)$$

$$= \Pr(\mathrm{Enc}(k, m_2) = c)$$

$\Rightarrow \quad P_r(\mathrm{Enc}(k, m_1) = c) = P_r(\mathrm{Enc}(k, m_2) = c)$

(II) Perfect indist $\Rightarrow$ Shannon security

This is the strongest notion of encryption security — even against a exponential time attacker, this security is enough

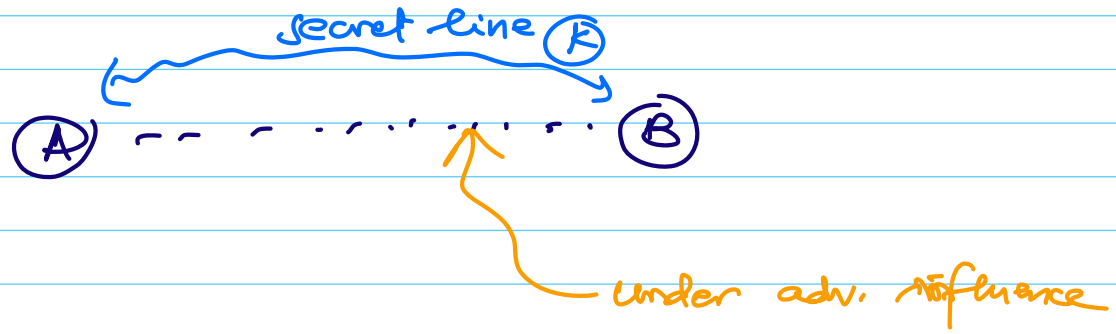
<u>Practical instantiation</u>. OTP

$$\mathrm{Enc}(k, m) = k \oplus m$$
$$\mathrm{Dec}(k, c) = k \oplus c$$

$k, m, c \in \{0,1\}^n$, k was chosen uniformly at random

<u>Practical problem</u> –

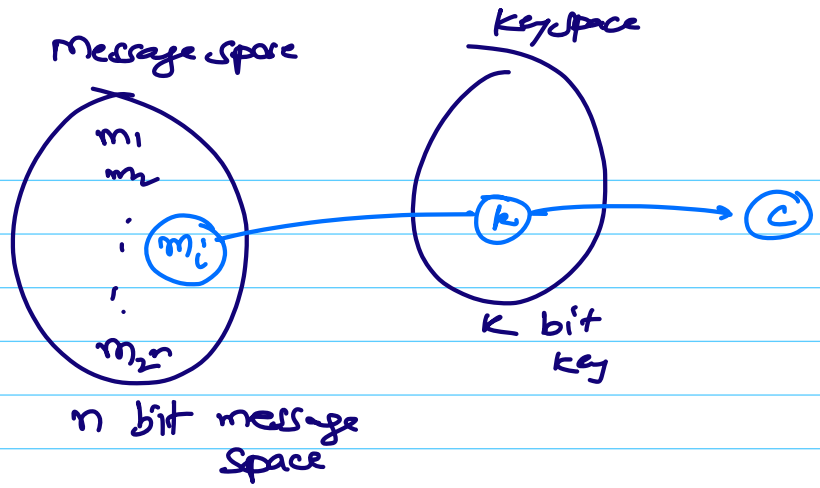
to encrypt a message of length n bits, we need a secret key of length n bits



<u>Que:</u> Can we reduce randomness requirement?

i.e message length $= n$ bits
but key length $= k$ bits where $k < n$

<u>Claim:</u> In this setting, perfect security is impossible



<u>Attacker's idea:</u>

- Pick the cipher text c
- Decrypt by all possible keys

$$\mathcal{P}' = \{ Dec(k_1, c), Dec(k_2, c), Dec(k_3, c), \ldots \}$$

$$|\mathcal{P}'| \leq |\mathcal{K}|$$

But

$$|\mathcal{K}| < |\mathcal{P}|$$

$\Rightarrow \exists\ m^* \in \mathcal{P}$ which is not in $\mathcal{P}'$

$$Pr(\mathcal{P} = m^* \mid C = c) = 0$$

But for some message $m' \in \mathcal{P}'$

$$Pr(\mathcal{P} = m' \mid C = c) \neq 0$$

$\Rightarrow$ Practical limitation in using OTP in real-life

Computational Security

(this is weaker than perfect security)

(I) Now we will limit the power of the adversary.

- Adv can not run for exponential time
- allowed only poly. time operations
- Security parameter = n

(II) Attacker will have some non-zero prob. of winning the distinguishability game

$$Pr(\ldots) - Pr(\ldots) \neq 0$$

Now

$$Pr(\ldots) - Pr(\ldots) = \epsilon$$

$\downarrow$

negligible in n

$\simeq 1/2^n$

Negligible function:

Pr of winning the game should be "negligible"

$$\text{poly}(n) \times (\text{Prob of win}) = \text{negl}(n)$$

$\Rightarrow$

$$\text{negl}(n) = \frac{\text{poly}(n)}{\exp(n)}$$ or something of that sort

$n \to \infty$, $\text{negl}(n) \to 0$

How to create a computationally secure encryption scheme ?

Why did OTP kill any statistics in the input ?

$Pr(b=0) = 0.9$, $Pr(b=1) = 0.1$

$Pr(k=0) = Pr(k=1) = 0.5$ uniformly random key bit

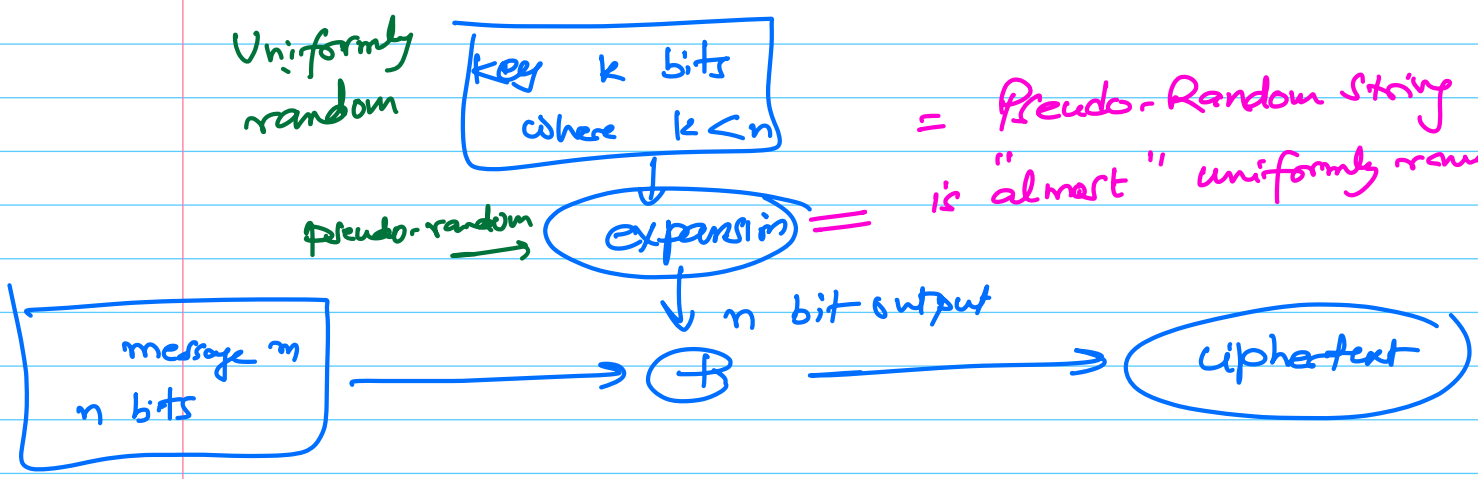
Que

$$Pr(b \oplus k = 0) = ?$$

$$= Pr(b = k = 0) + Pr(b = k = 1)$$

$$= \tfrac{1}{2} \times 0.9 + \tfrac{1}{2} \times 0.1$$

$$= \tfrac{1}{2}$$



Example: rand( ) function in C, Java, ...

What is a "random" string ?

1. is 000.... 0 random ?
2. is 110011... random ?
3. is 011001.... random ?

Source is random (Not the strings)

Yao – given a string 

(Next bit test)

Statistical test of randomness

NIST Test suite

(National Institute of Std. & Tech.)

Dept of Commerce US govt.

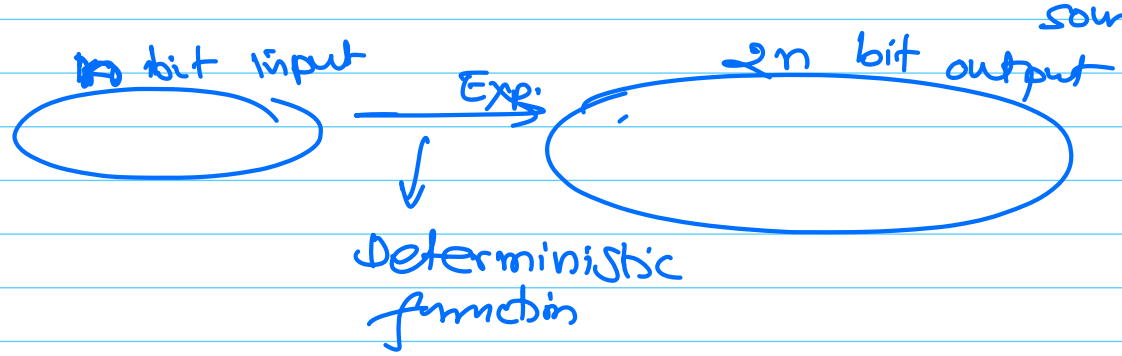
- Java randomness → entropy pool
- Lava rand – wall of entropy

PRG (Pseudo-Random Generator)

Idea: input: small amount of randomness

output: large " of pseudo-randomness

Something which can't be identified by "someone" as coming from random vs. pseudo random source



How many possible outputs for this function?

$= 2^n$

Size of output set $= 2^{2n}$

if we pick any random string in the output set, it is not likely to have been produced by this algo

$$\Pr\left(\begin{array}{c}\text{a random string} \\ \text{from output is} \\ \text{actually produced} \\ \text{from this algo.}\end{array}\right) = 1/2^n$$

Brute force attack will win with prob $\simeq 1$

— to prevent this, we will <u>not</u> allow attacker this exponential power

---

16-Jan-25

PRG - Pseudo-random generator (G)

<u>Idea:</u> a PRG takes a short seed and produces a long output which "<u>looks like</u>" random

& PRG is a deterministic function.

$G : \{0,1\}^s \longrightarrow \{0,1\}^\ell$ where <u>expansion</u> $\ell > s$

should be "looking like" random

Adv A : PPT : <u>Probabilistic</u> Polynomial Time algorithm

(i) internal coin tosses

(ii) also gets a random string as input

<u>Real world</u>	<u>Ideal world</u>
$s \xleftarrow{\$} \{0,1\}^n$	$y \xleftarrow{\$} \{0,1\}^\ell$
$y = G(s)$	