

# 2550 Intro to cybersecurity

## L4: Crypto, OWF

abhi shelat

A Network is a  
public resource.

Goal: add privacy to  
a public resource.

# Perfect secrecy

$(\underline{\text{Gen}}, \underline{\text{Enc}}, \underline{\text{Dec}}, \underline{\mathcal{M}}, \underline{\mathcal{K}})$

is said to be **PERFECTLY SECRET** if

$$\forall m_1, m_2 \in \underline{\mathcal{M}}, \forall \underline{c}$$

$$\Pr[k \leftarrow \underline{\text{Gen}} : \underline{\text{Enc}}_k(m_1) = \underline{c}] \\ =$$

$$\Pr[\underline{k} \leftarrow \underline{\text{Gen}} : \underline{\text{Enc}}_k(m_2) = c]$$

# One-time pad (Vernam 1917)



$$\frac{\mathcal{M}}{\mathcal{K}} = \frac{\{0,1\}^n}{\{0,1\}^n}$$

$$\text{Gen} = k = k_1 k_2 \dots k_n \leftarrow \{0,1\}^n$$

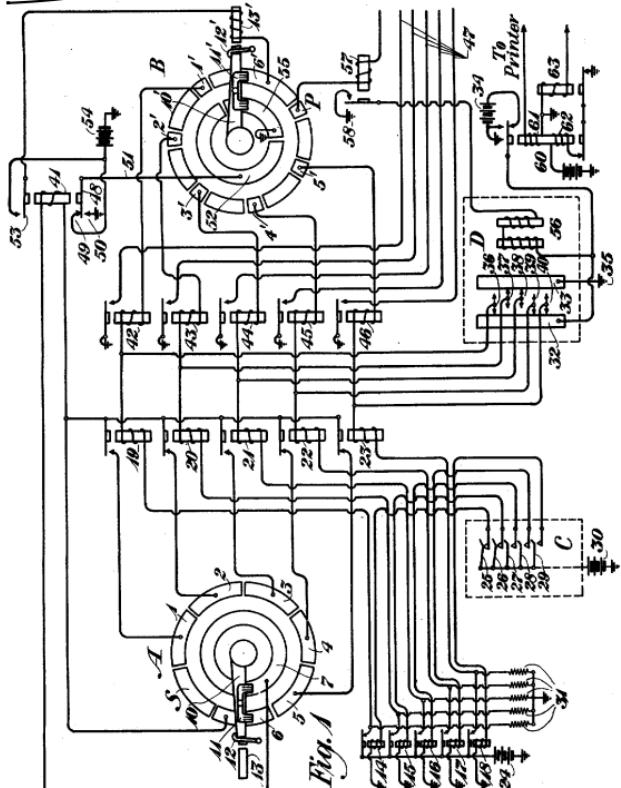
$$Enc_k(m_1 m_2 \dots m_n) = c_1 c_2 \dots c_n \text{ where } c_i = \underline{m_i} \oplus \overrightarrow{k_i}$$

$$Dec_k(c_1 c_2 \dots c_n) = m_1 m_2 \dots m_n \text{ where } m_i = \underline{\overrightarrow{c_i}} \oplus \overline{\underline{k_i}}$$

1,310,719.

G. S. VERNAM.  
~~SECRET SIGNALING SYSTEM.~~  
APPLICATION FILED SEPT. 13, 1918.

Patented July 22, 1919.  
2 SHEETS—SHEET 1.



INVENTOR.  
*G. S. Vernam*  
BY *G. E. Folsom*,  
ATTORNEY.

{ ABCDEFGHIJKLMNOPQRSTUVWXYZ }  
{ 01234567890123456789012345 }

M = IT WAS THE BEST OF TIMES

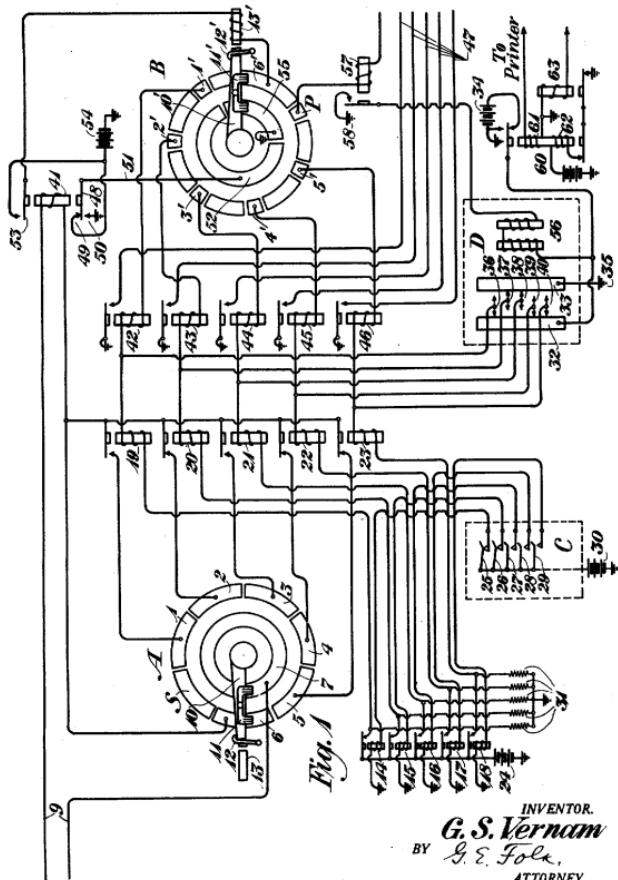
$\text{K} = \text{OSNEKHRJFUTGXMGALEQ}$

WLJ ...

G. S. VERNAM.  
SECRET SIGNALING SYSTEM.  
APPLICATION FILED SEPT. 13, 1918.

1,310,719.

Patented July 22, 1919.  
2 SHEETS—SHEET 1.



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

I T W A S T H E B E S T O F T I M E S

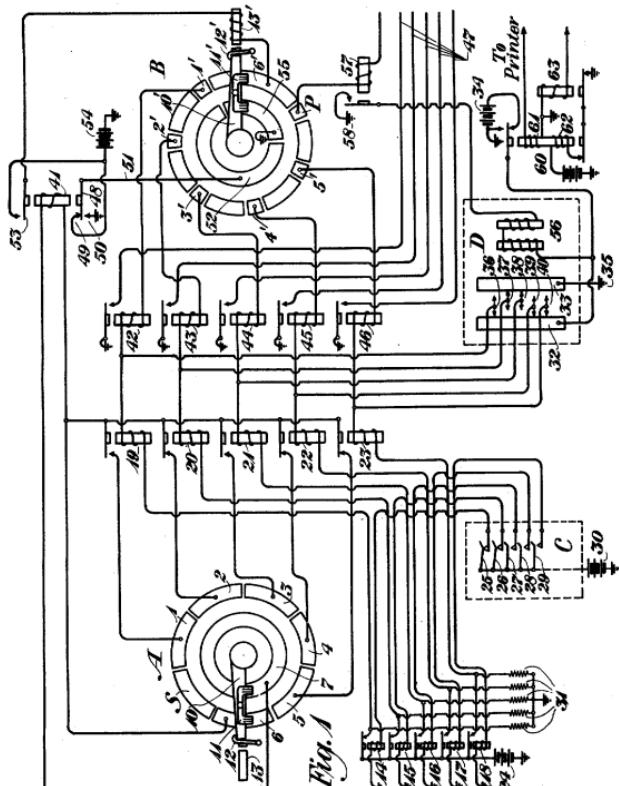
O S N E K H R J F U T G X M G A L E Q

W L J E C A Y N G Y L Z L R Z I X I I

1,310,719.

G. S. VERNAM.  
SECRET SIGNALING SYSTEM.  
APPLICATION FILED SEPT. 13, 1918.

Patented July 22, 1919.  
2 SHEETS—SHEET 1.



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

C = MBGZBWSNDHCVCGFN

K MVCLWOFLMDZBRYMPDV ~~E~~

M = A G E ... - - - - -

INVENTOR.  
BY G. S. VERNAM  
G. E. FOLEY  
ATTORNEY

# One-time pad (Vernam 1917)



- key size has to be equal to msg length
- hard to implement key mgmt.

$$\mathcal{M} = \{0, 1\}^n$$

$$\mathcal{K} = \{0, 1\}^n$$

$$\text{Gen} = k = k_1 k_2 \dots k_n \leftarrow \{0, 1\}^n$$

$$Enc_k(m_1 m_2 \dots m_n) = c_1 c_2 \dots c_n \text{ where } c_i = m_i \oplus k_i$$

$$Deck_k(c_1 c_2 \dots c_n) = m_1 m_2 \dots m_n \text{ where } m_i = c_i \oplus k_i$$

- Shannon shows this is unavoidable for perfect security.

FAILURE CASES : ① key runs out. what to do??

$\Rightarrow$  key reuse, which breaks security

# Uniform distribution on strings of len $n$

$$V = \underline{U_n} = \left\{ t \leftarrow \underbrace{\{0, 1\}^n} \right\}$$

random strings of length  $n$ .



- the keys in the one-time pad are drawn from  $\underline{U_n}$ .

# Goal: 1 key to many keys

1 short key  $\Rightarrow$  produce many long keys.

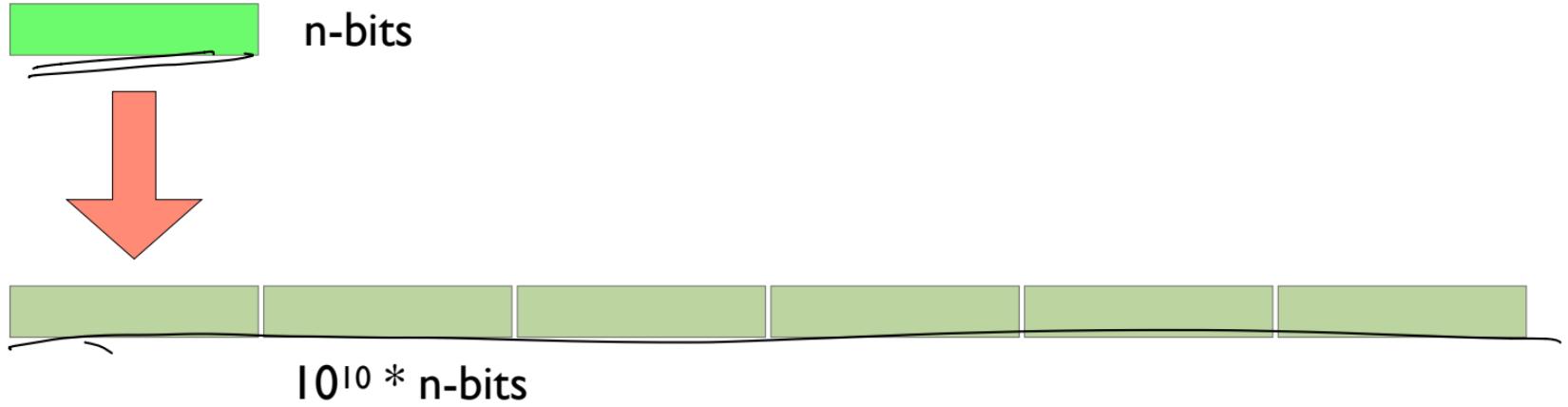
 1 short key

  
Key that we want :

 1 gb of Key : sampled from  $U_{1\text{gb}}$ .

# Goal: 1 key to many keys





This is the idea behind a **stream cipher**.

# Stream cipher

---

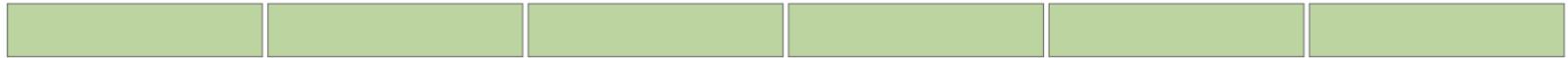
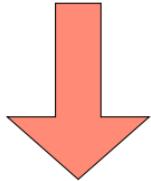
Gen: pick an n-bit binary string k

Enc( $k, m$ ): Output  $G(k) + \underline{m}$

Dec( $k, m$ ): Output  $G(k) + m$



n-bits



$10^{10} * n\text{-bits}$

what security properties are needed for this to work?

- ① the output of our generator should appear uniformly random...

# Vigenere cipher

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	

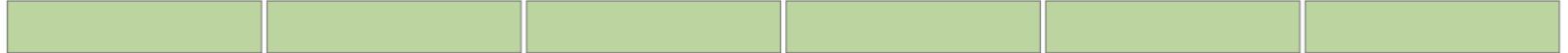
MSG: THE MODERN STUDY OF . . .

(+)

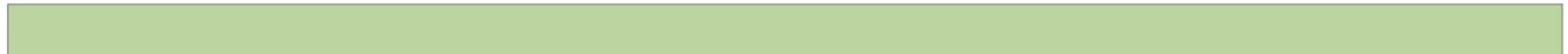
KEY: ABHI ABHI ABHI ABHI ABHI ABHI ABHI

ciphertext: T I L U O E L Z N T A C D Z V N . . .

Insecure b/c Key is not random !



$10^{10} * n\text{-bits}$





Key



Pseudo-random generator.

stream



$10^{10} * n$ -bits

should appear to be the same as a random string  $\{0, 1\}^{10^{10}n}$



$U_{10^{10}n}$

what does it mean  
for a process  $\{X\}$  to be  
pseudo-random?

$$\{X_n\}_{n \in \mathbb{N}}$$

$$x \leftarrow X_n$$

$\{X_n\}_{n \in \mathbb{N}}$

$x \leftarrow X_n$

$\{U_n\}_{n \in \underline{\mathbb{N}}}$

random  
process

roughly same # of 0s and 1s  $\rightarrow 0101010101\dots$

Sum # of 00 and 11, 01, 10  $\rightarrow 00100111|0010$

# of 000, 001, 010 ..., 111

$$\{X_n\}_{n \in \mathbb{N}}$$

$$x \leftarrow X_n$$

roughly same # of 0s and 1s

roughly same # of 00s and 11s

$$\{X_n\}_{n \in \mathbb{N}}$$

$$x \leftarrow X_n$$

roughly same # of 0s and 1s

roughly same # of 00s and 11s

roughly same # of any pattern

"any <sup>efficient</sup> statistical test should pass on both  
 $\{U_n\}$  and  $\{X_n\}$ "

$\{X_n\}_{n \in \mathbb{N}}$  and  $\{\underline{U_n}\}$

$x \leftarrow \underline{X_n}$  ARE

roughly same # of 0s and 1s

INDISTINGUISHABLE

roughly same # of 00s and 11s

TO THE

roughly same # of any pattern

ADVERSARY

given any prefix, hard to guess next bit

{ Computational  
‘Indistinguishability’ }

provides a precise  
way of formulating  
pseudo-randomness

next slide has 2 pics

are they the same  
or different?





same or different?

twice the time.

same or different?





same or different?

twice the time.

same or different?





same or different?

twice the time.

same or different?





# lesson:

Ability to answer correctly...

Depends on the amount of  
computation power that you have

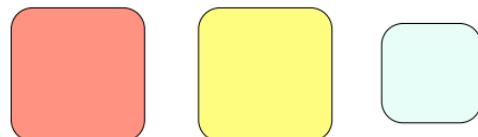
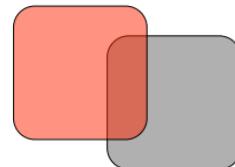


# NEW PROBLEM:

consider all drawings consisting of boxes.

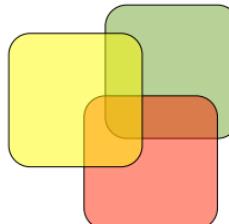
evens

# of boxes that overlap  
another box is even



odds

# of ... is odd



# GAME:

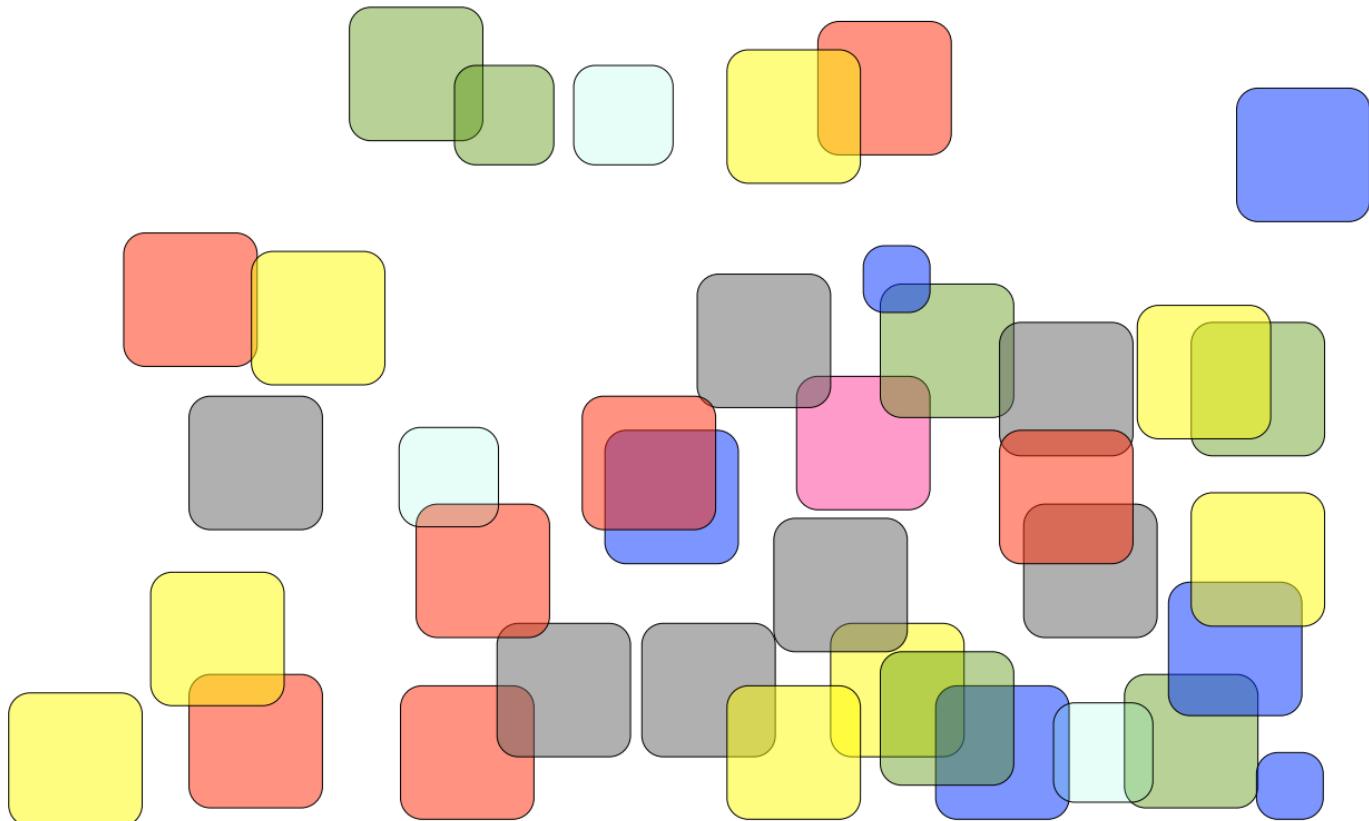
I will pick a sample from either **evens** or **odds**, and you will have to guess which one.

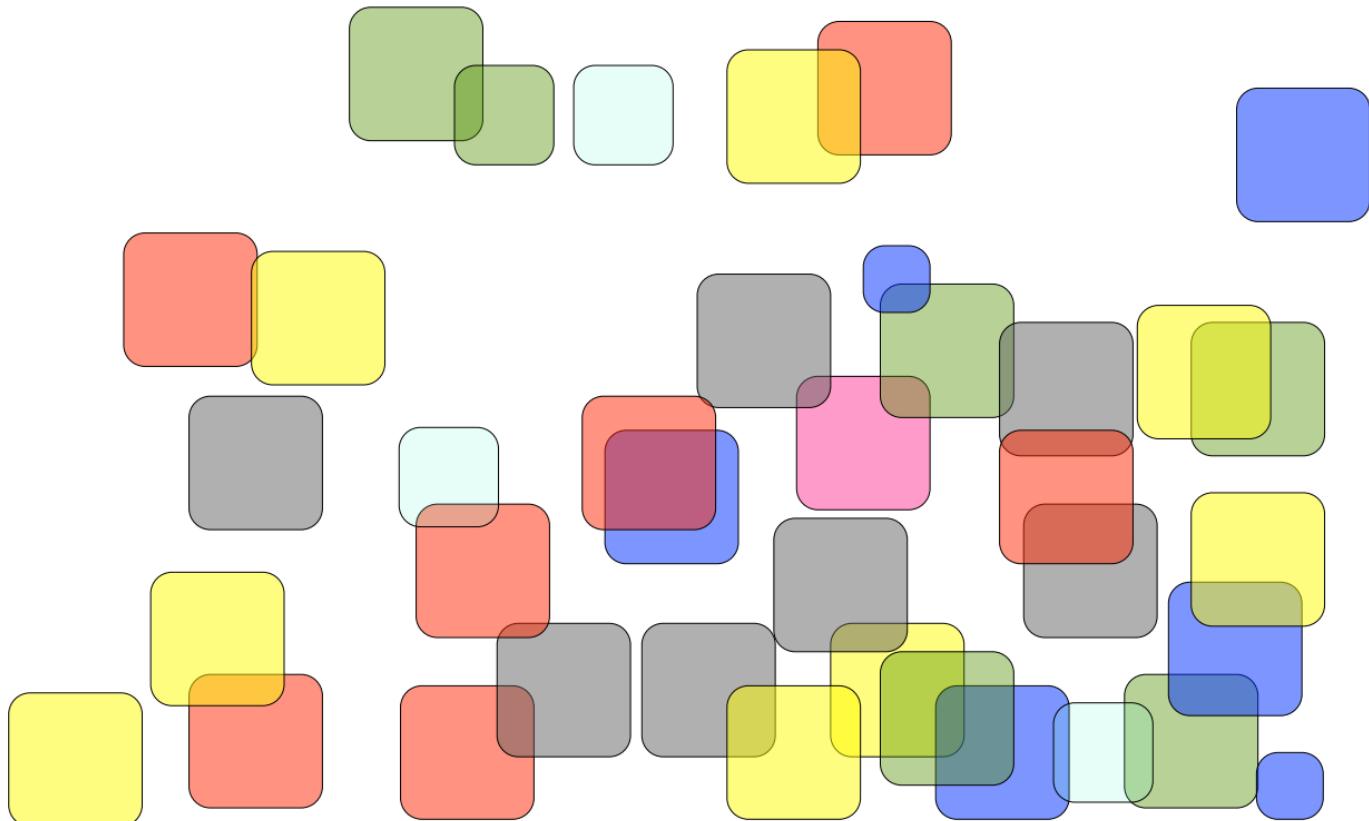
READY?

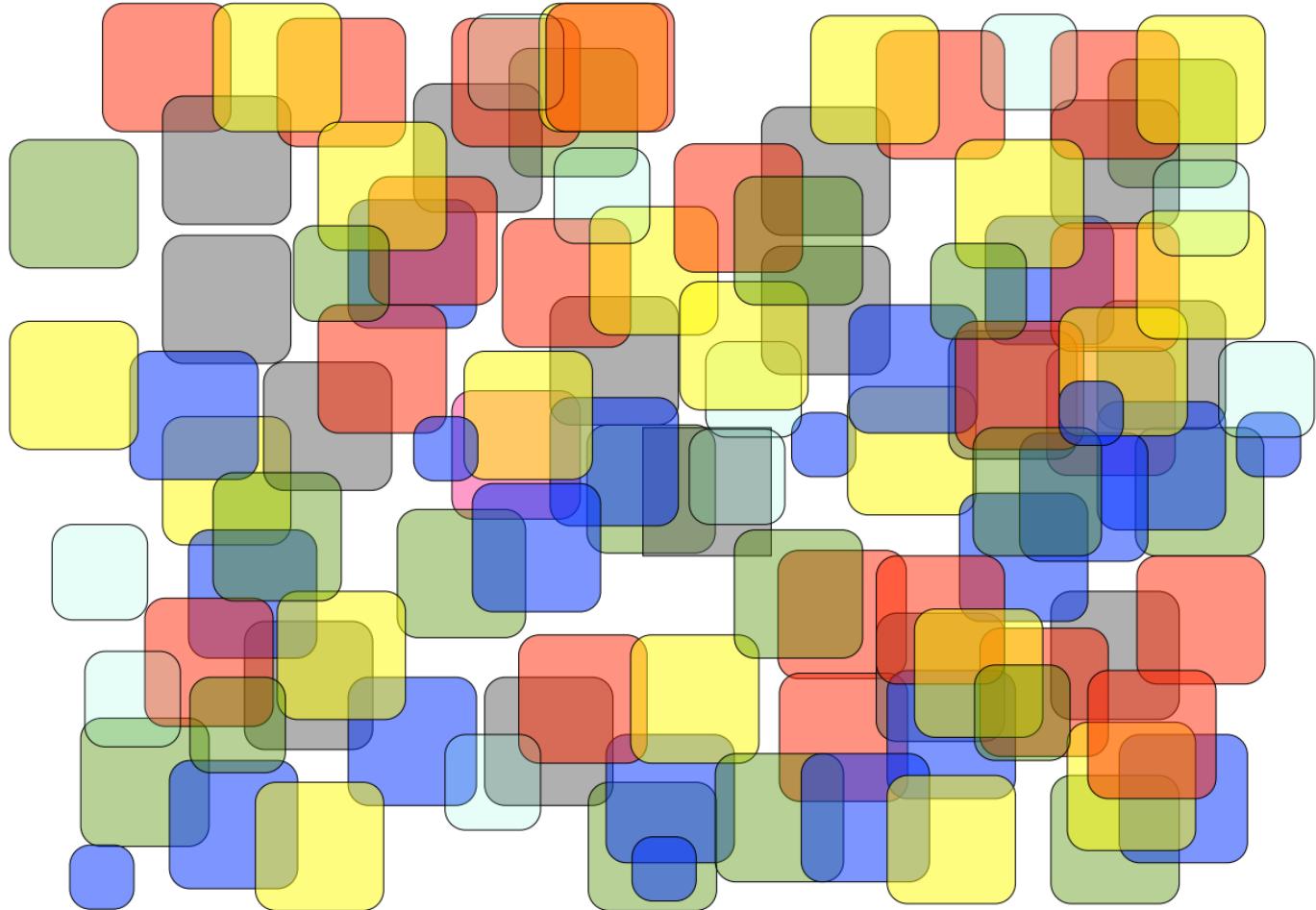
READY?



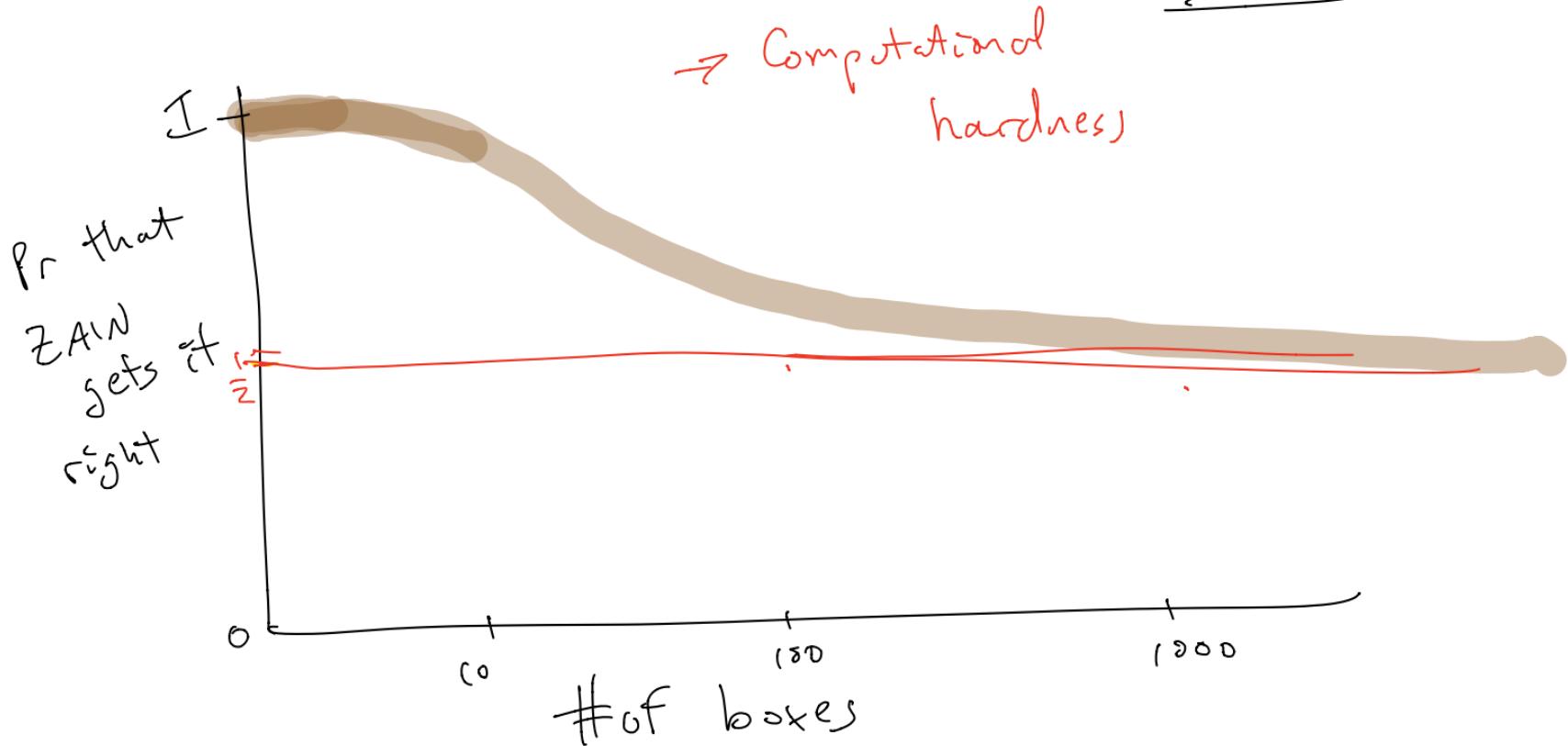








This game is parameterized by its size: i.e, # of boxes.



This game is parameterized by its size: i.e, # of boxes.

evens<sub>n</sub>

odds<sub>n</sub>

As the game size increases, it becomes  
intractable to  
distinguish b/w  
evens and odd

# parameterized experiment

$$\underbrace{\{X_n\}_{n \in \mathbb{N}}}_{\text{C}}$$

a sequence of probability distributions  
where  $X_n$  is a distribution over strings of length  
 $\ell(n)$

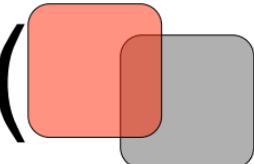
ensembles

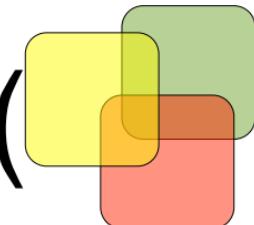
# Computational Indistinguishability

“let there be two parameterized experiments,  $X$  and  $Y$ .  
as the experiment **size** increases, no p.p.t. algorithm  $D$   
succeeds in distinguishing  $X$  from  $Y$ .”

probabilistic  
polynomial  
time } “efficient”

what does it mean  
for an algorithm  $D$  to  
distinguish a sample?

D() = “evens”

D() = “odds”

Two ensembles are comp. indistinguishable

Two ensembles are comp. indistinguishable

$$\{X_n\}_{n \in N} \approx \{Y_n\}_{n \in N}$$

# Two ensembles are comp. indistinguishable

$$\underbrace{\{X_n\}_{n \in N}}_{\text{---}} \approx \underbrace{\{Y_n\}_{n \in N}}_{\text{---}}$$

if for all non-uniform p.p.t. alg D,  
there exists a negligible function  $\epsilon(n)$   
such that for all n

# Two ensembles are comp. indistinguishable

*formally*

$$\{X_n\}_{n \in N} \approx \{Y_n\}_{n \in N}$$

if for all non-uniform p.p.t. alg  $D$ ,  
there exists a negligible function  $\epsilon(n)$   
such that for all  $n$

$$|\Pr [t \leftarrow X_n, D(t) = 1] - \Pr [t \leftarrow Y_n, D(t) = 1]| \leq \epsilon(n).$$

what does it mean  
for a process  $\{X\}$  to be  
pseudo-random?

$$\{X_n\}_{n \in \mathbb{N}}$$

$$x \leftarrow X_n$$

$$\{X_n\}_{n \in \mathbb{N}}$$

$$x \leftarrow X_n$$

roughly same # of 0s and 1s

$$\{X_n\}_{n \in \mathbb{N}}$$

$$x \leftarrow X_n$$

roughly same # of 0s and 1s

roughly same # of 00s and 11s

$$\{X_n\}_{n \in \mathbb{N}}$$

$$x \leftarrow X_n$$

roughly same # of 0s and 1s

roughly same # of 00s and 11s

roughly same # of any pattern

$$\{X_n\}_{n \in \mathbb{N}}$$

$$x \leftarrow X_n$$

roughly same # of 0s and 1s

roughly same # of 00s and 11s

roughly same # of any pattern

given any prefix, hard to guess next bit

indistinguishability  
provides a precise  
way of formulating  
pseudo-randomness

pseudo-random

An ensemble  $\{X\}$  is said to be

pseudo-random

pseudo-random

if

$$\{X\}_{n \in \mathbb{N}} \approx \{U_n\}_{n \in \mathbb{N}}$$

---

An ensemble  $\{X\}$  is said to be

pseudo-random

if

$$\{X\}_{n \in \mathbb{N}} \approx \{U_n\}_{n \in \mathbb{N}}$$

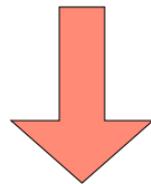
# Original goal

small key

\



n-bits



large one-time pad.



$10^{10} * n\text{-bits}$

indistinguishable from a random key.

# Pseudo-random generator

A function  $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$

is a **pseudo-random generator** if

$G$  can be computed in p.p.t.  $\leftarrow$

$|G(x)| > \ell(|x|)$  for some  $\ell(y) > y$   $\leftarrow$

$\{x \leftarrow U_n : G(x)\}_{n \in \mathbb{N}}$  is pseudo-random

How can we build  
pseudo-random  
generators?

7

$$7 * 7 =$$

49

$7 \cdot 7 \cdots 7$

$x$  times

$7x$

7<sup>1000</sup>

7<sup>10000</sup>



7<sup>10000000000000000000</sup>



Examples Random

Input:

7<sup>100 000 000 000 000 000</sup>

Power of 10 representation:

$10^{10^{1.228577610529823}}$

Number length:

84 509 804 001 425 684 decimal digits

$\approx 8.45098 \times 10^{16}$  digits

Last few digits:

...00000000001

Computed by **Wolfram Mathematica**

Download page

Inverse problem is EASY:

**Given:**

1231928

**Find**

X such that  $7^X = 1231928$

Logarithm, base 7 of 1231928

7.2069571077666163863237014656943296629255654569808791967051



2010



2010



$$f(x) = 7^x$$

Easy

X

Y

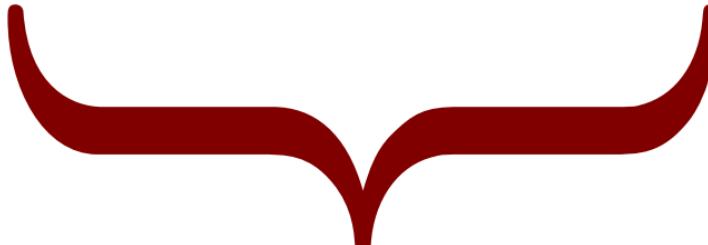
Also easy

$$f^{-1}(Y)$$

log base 7

$$7^x$$

$7 \cdot 7 \cdots 7$



$x$  times

$$f(x) = 7^x \pmod{p}$$

$$\underbrace{7^2}_{49} = \underbrace{7 \cdot 7}_{49} \mod \underline{131}$$

$$49 \mod 131 = 49$$

$$7^2 = 7 \cdot 7 \mod 131$$

$$= 49$$

$$7^3 = \underline{49} \cdot 7 \pmod{131}$$

$$\begin{array}{r} 2 \\ 131 \overline{)343}, \text{ r. } 81 \\ \underline{\underline{3}} \\ 262 \\ \underline{\underline{-262}} \\ 81 \end{array} \pmod{131}$$

$$7^3 = 49 \cdot 7 \pmod{131}$$

$$\begin{array}{r} 2 \\ \hline 131 \overline{)343} \end{array} \quad \text{mod } 131$$

262

81

$$7^3 = \cancel{49} \cdot 7 \pmod{131}$$

$$\begin{array}{r} 2 \\ \hline 131 \sqrt{343} \\ \hline 262 \\ \hline 81 \end{array} \pmod{131}$$

$$7^4 = \underline{\underline{81}} \cdot 7$$

$$= 81 \pmod{\underline{\underline{131}}}$$

Pick a large prime number,  $\sim 200$  digits long

$$p = \begin{array}{l} 5206171726881102258298508534096289004305794879961025454 \\ 3088975844032721944045806199348262783479007502882378061 \\ 8996828085538214474401045588718257206979528176365692374 \\ 6706192367848073555079554994166440456995548499 \end{array}$$

$$(7, \underline{x}, p) \rightarrow \underline{7^x \bmod p}$$

Remarkably easy

Assailing U.S. and Kiev, Putin Keeps Open Option of Force, By STEVEN LEE MYERS, ELLEN BARRY and ALAN COWELL 56 minutes ago, President Vladimir V. Putin of Russia on Tuesday described events in Ukraine as an unconstitutional coup, expressed contempt toward the United States and said that any potential use of military force would be a last resort.

sage:

$$7^{x \text{ mod } p}$$

```
power_mod(7, 66161598060906110332854683463298110032760  
60218443281181706103276020213153280130210328013170612  
10331202327112150001443267213283846986697832766969327  
78969828344326976766978326665828289329811003265766578  
32677987697676325354331006111817021532980411443281150  
21606010211163287089801061006143286463281181706103312  
02328318161605973312103285180216009821330102160015059  
90200330219021117153306103286081498061101329815329810  
33181100121116170617181706121098083300121812443302211  
31502161602003300121117021013163317121998150033170501  
32861106170200328416981702153298110033159806003317049  
81632981121331312170211170598083318160133120233100609  
06169815213303121500013320121809003299013297330898161  
63315021612151646, p)
```

49792971168858421752786333230062932909865498326942335  
77295952890262039411267933871586617511805097012352566  
33429962635553676002083869590769727522267027725835398  
0892233568459367693160250707721036268518993463932429

$$Y = \underline{\underline{12345}}$$

$$(7, p, Y) \rightarrow x$$

such that  $7^x \bmod p = \underline{\underline{Y}}$

discrete logarithm

Incredibly hard!

# World record in discrete logarithms in GF(p)

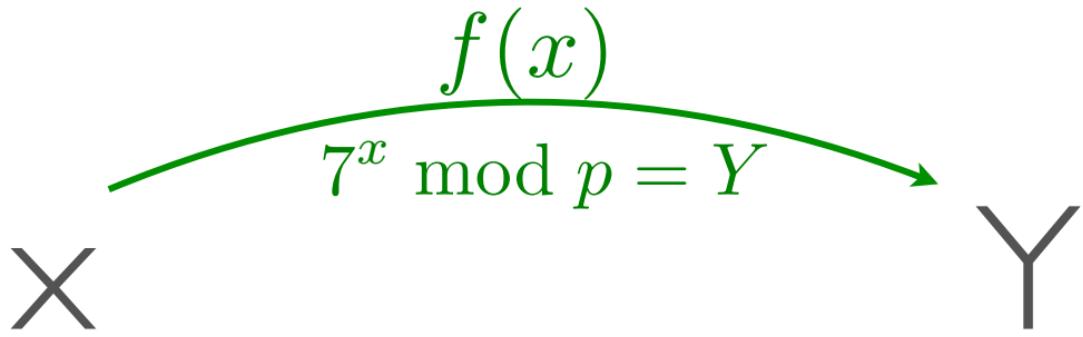
232 digits (768 bits)

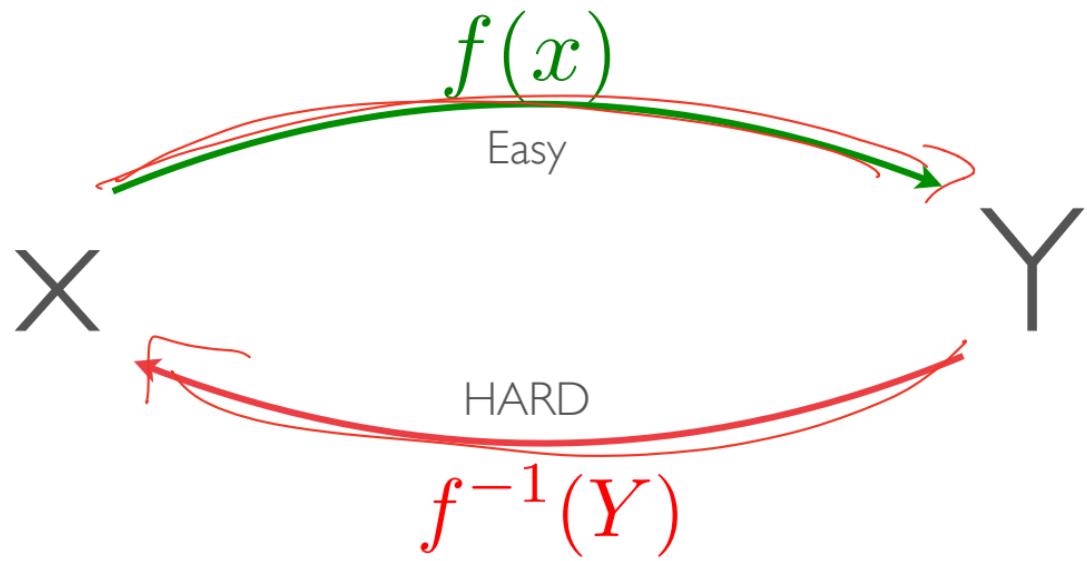
6600yrs of CPU time

Intel Xeon E5-2660 at 2.2 GHz

2016

Thorsten Kleinjung, Claus Diem, [Arjen K. Lenstra](#), Christine Priplata, and Colin Stahlke





compute an  $x$  such that  $7^x = Y \text{ mod } p$



<http://www.fitzmuseum.cam.ac.uk/gallery/chinesevases/sortingfragments.html>



$f^x \text{ mod } f$ .

