Random Functions

We like to consider functions with nice, closed forms. E.g. $f(x) = x^2$.

Random Functions

We like to consider functions with nice, closed forms. E.g. $f(x) = x^2$.

If we have a finite domain, we can represent functions as a *table*

x	$f(x)$
0	0
1	1
2	4
3	9
$\vdots$	$\vdots$
2^n	2^{2n}

Random Functions

We like to consider functions with nice, closed forms. E.g. $f(x) = x^2$.

If we have a finite domain, we can represent functions as a *table*

x	$f(x)$
0	0
1	1
2	4
3	9
$\vdots$	$\vdots$
2^n	2^{2n}

We can sample a *random* function by choosing the values in the right column uniformly and independently:

x	$f(x)$
000	101
001	111
010	100
011	101
100	110
101	010
110	000
111	011

Counting functions

Question

How many functions are there mapping $\{0,1\}^n \to \{0,1\}^n$?

Counting functions

Question

How many functions are there mapping $\{0,1\}^n \to \{0,1\}^n$?

If we change a single output value, we have a new function:

x	$f(x)$
000	101
001	111
010	100
011	101
100	110
101	010
110	000
111	011

x	$f(x)$
000	101
001	111
010	100
011	001
100	110
101	010
110	000
111	011

Counting functions

How many bits does it take to represent one of these functions?

x	$f(x)$
000	101
001	111
010	100
011	001
100	110
101	010
110	000
111	011

Counting functions

How many bits does it take to represent one of these functions?

x	$f(x)$
000	101
001	111
010	100
011	001
100	110
101	010
110	000
111	011

101 111 100 011 110 010 000 011

Counting functions

How many bits does it take to represent one of these functions?

x	$f(x)$
000	101
001	111
010	100
011	001
100	110
101	010
110	000
111	011

101 111 100 011 110 010 000 011

$n2^n$

Counting functions

How many bits does it take to represent one of these functions?

x	$f(x)$
000	101
001	111
010	100
011	001
100	110
101	010
110	000
111	011

101 111 100 011 110 010 000 011

$n2^n$

Each string of length $n2^n$ represents a different function from $\{0,1\}^n \to \{0,1\}^n$

Counting functions

How many bits does it take to represent one of these functions?

x	$f(x)$
000	101
001	111
010	100
011	001
100	110
101	010
110	000
111	011

101 111 100 011 110 010 000 011

$n2^n$

Each string of length $n2^n$ represents a different function from $\{0,1\}^n \to \{0,1\}^n$

How many strings of length x are there? 2^x.

Counting functions

How many bits does it take to represent one of these functions?

x	$f(x)$
000	101
001	111
010	100
011	001
100	110
101	010
110	000
111	011

101 111 100 011 110 010 000 011

$n2^n$

Each string of length $n2^n$ represents a different function from $\{0,1\}^n \to \{0,1\}^n$

How many strings of length x are there? 2^x.
How many strings of length $n2^n$ are there? $2^{n2^n} > 2^{2^n}$

Counting functions

How many bits does it take to represent one of these functions?

x	$f(x)$
000	101
001	111
010	100
011	001
100	110
101	010
110	000
111	011

101 111 100 011 110 010 000 011

$n2^n$

Each string of length $n2^n$ represents a different function from $\{0,1\}^n \to \{0,1\}^n$

How many strings of length x are there? 2^x.
How many strings of length $n2^n$ are there? $2^{n2^n} > 2^{2^n}$ (That's a lot of functions.)

Pseudo-random Functions (PRFs)

We'd like to use randomly chosen functions, but this requires exponential space!

.

Pseudo-random Functions (PRFs)

We'd like to use randomly chosen functions, but this requires exponential space!

Instead, we will use pseudo-random functions: keyed functions that are indistinguishable from random:
$F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$
This is a 2-input function, where 1st input is the key.

Pseudo-random Functions (PRFs)

We'd like to use randomly chosen functions, but this requires exponential space!

Instead, we will use pseudo-random functions: keyed functions that are indistinguishable from random:
$F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$
This is a 2-input function, where 1st input is the key.

The sec. param. determines the key length, the input length, and the output length. Technically, $\ell_{\text{key}}(n), \ell_{\text{in}}(n)$ and $\ell_{\text{out}}(n)$.

Pseudo-random Functions (PRFs)

We'd like to use randomly chosen functions, but this requires exponential space!

Instead, we will use pseudo-random functions: keyed functions that are indistinguishable from random:
$F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$
This is a 2-input function, where 1st input is the key.

The sec. param. determines the key length, the input length, and the output length.
Technically, $\ell_{\text{key}}(n), \ell_{\text{in}}(n)$ and $\ell_{\text{out}}(n)$.
Often, we will assume that F is length preserving:
$\ell_{\text{key}}(n) = \ell_{\text{in}}(n) = \ell_{\text{out}}(n) = n$

.

Pseudo-random Functions (PRFs)

We'd like to use randomly chosen functions, but this requires exponential space!

Instead, we will use pseudo-random functions: keyed functions that are indistinguishable from random:
$F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$
This is a 2-input function, where 1st input is the key.

The sec. param. determines the key length, the input length, and the output length.
Technically, $\ell_{\text{key}}(n), \ell_{\text{in}}(n)$ and $\ell_{\text{out}}(n)$.
Often, we will assume that F is length preserving:
$\ell_{\text{key}}(n) = \ell_{\text{in}}(n) = \ell_{\text{out}}(n) = n$

Often, we will want to fix a single key k and then evaluate F on many different inputs, using the same k. In that case, we might write $F_k : \{0,1\}^* \rightarrow \{0,1\}^*$.
If it is length preserving, and the key is of length n, then $F_k : \{0,1\}^n \rightarrow \{0,1\}^n$.

Security of PRFs

Definition

Let $F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be an efficient, length-preserving, keyed function. F is a *pseudorandom function* if $\forall$ p.p.t. adversaries $\mathcal{A}$, there is a negligible function negl(n) such that $\Pr[\mathsf{PrivK}^{\mathsf{prf}}_{\mathcal{A},\mathsf{F}}(n) = 1] \leq \frac{1}{2} + \mathsf{negl(n)}$.