

## Handout 2: Useful Notation

*Instructor: Silvio Micali**Teaching Assistant: Rafael Pass*

[Handout created by Anna Lysyanskaya]

**Notation****Outputs:** Say  $A$  is an Algorithm.

- $A(\cdot)$  denotes an algorithm with one input.
- $A(\cdot, \cdot)$  denotes an algorithm with two inputs.
- $A(x)$  denotes the probability distribution consisting of the output of algorithm  $A$  on input  $x$ . (Note: This distribution is concentrated on a single element if  $A$  is deterministic).

**Experiments:**

- $x \leftarrow S$ , for a probability distribution  $S$ , denotes the probabilistic experiment which assigns to  $x$  an element selected according to the probability distribution  $S$ .
- $x \leftarrow F$ , for a finite set  $F$ , denotes the experiment which assigns to  $x$  an element selected according to the uniform probability distribution on set  $F$ .
- We now introduce notation for an ordered sequence of experiments:

$$((x, y) \leftarrow A(3); z \leftarrow B(y))$$

denotes the experiment which first assigns to the pair  $(x, y)$  an element selected from the probability distribution of the outputs of algorithm  $A$  on input 3 and then assigns to  $z$  an element selected from the probability distribution of the output of algorithm  $B$  on input  $y$ . (Note: That an experiment in a sequence can be dependent on other experiments earlier in the sequence).

**Output of Experiments**

- If  $p(\cdot, \cdot)$  is a predicate, then the notation  $Pr[x \leftarrow S; y \leftarrow T : p(x, y)]$  denotes the probability that  $p(x, y)$  will be true after the ordered sequence of experiments  $(x \leftarrow S; y \leftarrow T)$ .
- The notation  $\{x \leftarrow S; y \leftarrow T : (x, y)\}$  denotes the Probability distribution over  $\{(x, y)\}$  generated by the ordered sequence of experiments  $(x \leftarrow S; y \leftarrow T)$ .

## Example

Now, using this notation, let us formalize the statement "Factoring an RSA modulus is hard". We would like to transform this statement into a statement about the probability, for any given algorithm  $A$ , that, on input an RSA modulus  $n$ ,  $A$  will output one of  $n$ 's factors.

Let  $PRIMES_k$  be the set of  $k$ -bit prime numbers. The probability that algorithm  $A$  successfully factors an RSA modulus can be written as follows:

$$Pr[p \leftarrow PRIMES_k; q \leftarrow PRIMES_k; x \leftarrow A(pq) : x = p \vee x = q]$$

Now let us see what probability of this undesirable event is satisfactory. We would, of course, like it to be 0. But this is impossible, since we can design an algorithm that guesses a factor by uniformly selecting a  $k$ -bit number at random, and this simple algorithm succeeds with non-zero probability. Therefore, we must be satisfied if the probability is so small that we are so unlikely to observe the effects of the fact that it is non-zero, that for all practical purposes it is 0. Thus we would like it to be so small that no polynomial time algorithm can observe that it is non-zero. That is to say:

$$\forall A \forall c \in \mathbb{N} \exists k_0 \text{ s.t. } \forall k > k_0$$

$$Pr[p \leftarrow PRIMES_k; q \leftarrow PRIMES_k; x \leftarrow A(pq) : x = p \vee x = q] < \frac{1}{k^c}$$