6.876/18.426: Advanced Cryptography February 10, 2003

Handout 2: Enhanced GMR Notation

1 Introduction

These notations described here are the *fully formal* ones. We have somewhat looser notation in the actual lecture, but the concepts laid out here are crucial for proper understanding.

2 Notation

2.1 Enhanced GMR Notation

For handling more complex probabilistic experiments, we present an **enhancement** to the standard GMR notation [2].

Basic Notations

- *Integers, Sets and Strings.*

  We denote by $\mathcal{N}$ the set of natural numbers. If $n \in \mathcal{N}$, by 1^n we denote the concatenation of n 1's. We identify a binary string σ with the integer x whose binary representation (with possible leading zeroes) is σ.

  By the expression $|x|$ we denote the length of x if x is a string, the length of the binary string representing x if x is an integer, the absolute value of x if x is a real number, or the cardinality of x if x is a set.

  If σ and τ are binary strings, we denote their concatenation by either $\sigma \circ \tau$ or $\sigma\tau$.

  A language is a subset of $\{0,1\}^*$. If L is a language

  and $k > 0$, we set $L_k = \{x \in L : |x| \leq k\}$. For variety of discourse, we may call "theorem" a string belonging to the language at hand. (A "false theorem" is a string string outside L.)

- *Algorithms.*

  An algorithm is a Turing machine. An *efficient* algorithm is a probabilistic Turing machine running in expected polynomial time.

  We emphasize the number of inputs received by an algorithm as follows. If algorithm A receives only one input we write "$A(\cdot)$", if it receives two inputs we write "$A(\cdot,\cdot)$" and so on.

  If $A(\cdot)$ is a probabilistic algorithm, then for any input x, the notation $A(x)$ refers to the probability space that assigns to the string σ the probability that A, on input x, outputs σ.

We assume a standard encoding is adopted and, if A is an algorithm, then we denote by $\langle A \rangle$ the encoding of A.

-

Standard GMR Notation

- *Random assignments.* If S is a probability space, then "$x \stackrel{R}{\leftarrow} S$" denotes the algorithm which assigns to x an element randomly selected according to S. If F is a finite set, then the notation "$x \stackrel{R}{\leftarrow} F$" denotes the algorithm which assigns to x an element selected according to the probability space whose sample space is F and uniform probability distribution on the sample points.

- *Probabilistic experiments.* If $p(\cdot, \cdot, \cdots)$ is a predicate, the notation $\Pr[x \stackrel{R}{\leftarrow} S; y \stackrel{R}{\leftarrow} T; ... : p(x, y, \cdots)]$ denotes the probability that $p(x, y, \cdots)$ will be true after the ordered execution of the algorithms $x \stackrel{R}{\leftarrow} S$, $y \stackrel{R}{\leftarrow} T, ...$

- *Probability spaces.*

  The notation $\{x \stackrel{R}{\leftarrow} S; y \stackrel{R}{\leftarrow} T; \cdots : (x, y, \cdots)\}$ denotes the probability space over $\{(x, y, \cdots)\}$ generated by the ordered execution of the algorithms $x \stackrel{R}{\leftarrow} S$, $y \stackrel{R}{\leftarrow} T, \cdots$.

- *Negligible Functions* We denote by $\nu : \mathcal{N} \rightarrow (0, 1)$ a function that vanishes faster than the inverse of any fixed polynomial, for all sufficiently large arguments.

New GMR Notation

- *History-Preserving Algorithms.* We say that an algorithm (or interactive TM) A is *history-preserving* (HP, for short) if it "never forgets" anything. As soon as it flips a coin or receives an input or a message, A writes it on a separate history tape that is write-only and whose head always moves from left to right. The history tape's content coincides with A's internal configuration before A executes any step.

  If A is an HP algorithm, then if A appears more than once in a piece of GMR notation (e.g, $\Pr[\ldots; a \stackrel{R}{\leftarrow} A(x); \ldots; b \stackrel{R}{\leftarrow} A(y); ... : p(\cdots, a, b, \cdots)]$) then it is understood that the final internal configuration and content of the history tape of one run of A coincide with A's initial internal configuration and content of the history tape of the next run.

  If A is an HP algorithm, then if A appears more than once in a piece of GMR notation (e.g, $\Pr[\ldots; a \stackrel{R}{\leftarrow} A(x); \ldots; b \stackrel{R}{\leftarrow} A(y); ... : p(\cdots, a, b, \cdots)]$) then the history and state of A is preserved from the end of one "use" to the beginning of the next.

  The notation $h \stackrel{H}{\longleftarrow} A$ indicates that h is the content of the current history tape of A.

  If A is a HP algorithm, and h the final history of an execution of A, we denote by $A\{h\}$ the algorithm having the same tapes and finite state control of A and initial configuration equal to the last configuration of h.

- *Adversaries.* An *adversary* is an efficient history-preserving algorithm (interactive TM).

- *Non-Determinism.* If S is a set, the notation $x \stackrel{ND}{\longleftarrow} S$ indicates that x has been non-deterministically chosen from S.

  Whenever non-deterministic choices appear within an experiment, they are regarded as constants (and not random variables), and all probabilistic statements made refers to each possible individual choice. For instance, the expression $Pr(x \stackrel{ND}{\longleftarrow} S : A(x) = 1) = 1/2$ means that, for every $x \in S$, A on x outputs 1 with exactly probability 1/2.

2.2 Protocols

Following [1], we consider a two-party protocol as a pair, (A, B), of interactive Turing machines. (ITMs for short). Briefly, on input (x, y), where x is a private input for A and y a private input for B, and random input (r_A, r_B), where r_A is a private random tape for A and r_B a private random tape for B, protocol (A, B) computes in a sequence of rounds, alternating between A-rounds and B-rounds. In an A-round (B-round) only A (only B) is active and sends a message (i.e., a string) that will become an available input to B (to A) in the next B-round (A-round). A computation of (A, B) ends in a B-round in which B sends the empty message and computes a private output.[1]

Executions, Transcripts, and Outputs

If (A, B) is a protocol and (x, y) an input for (A, B), we let $EXE^{A,B}(x|y)$ denote the experiment of randomly executing (A, B) on input (x, y). In our definitions, we think of this experiment as affecting the computation history of the participants rather than having an output.

Letting E be an execution of protocol (A, B) on input (x, y) and random input (r_A, r_B), we make the following definitions:

- The *transcript* of E consists of the sequence of messages exchanged by A and B, and is denoted by $TRANS^{A,B}(x, r_A|y, r_B)$;
- The *view of* A consists of the triplet (x, r_A, t), where t is E's transcript, and is denoted by $VIEW_A^{A,B}(x, r_A|y, r_B)$;
- The *view of* B consists of the triplet (y, r_B, t), where t is E's transcript, and is denoted by $VIEW_B^{A,B}(x, r_A|y, r_B)$;
- The *output of* B *in* E consists of the string z output by B in the last round of E, and is denoted by $OUT_B^{A,B}(x, r_A|y, r_B)$, though it only depends on B's inputs, coin tosses, and E's transcript.

  When we are only concerned with the output, and when both parties have the same input x, we sometimes write $(A, B)[x]$ as shorthand for $OUT^{A,B}(x, \cdot|y, \cdot)$.

We also define the following random distributions

- $TRANS^{A,B}(x, \cdot|y, r_B)$, $TRANS^{A,B}(x, r_A|y, \cdot)$, and $TRANS^{A,B}(x, \cdot|y, \cdot)$

[1] Due to the one-sidedness of secure computation, only machine B produces an output.

as the distribution respectively obtained by randomly selecting r_A, r_B, or both, and then drawing from $TRANS^{A,B}(x, r_A|y, r_B)$. We also consider the similarly defined random variables

- $VIEW_A(x, \cdot|y, r_B)$, $VIEW_A(x, r_A|y, \cdot)$, $VIEW_A(x, \cdot|y, \cdot)$, $VIEW_B(x, \cdot|y, r_B)$, $VIEW_B(x, r_A|y, \cdot)$, and $VIEW_B(x, \cdot|y, \cdot)$; and
- $OUT_B^{A,B}(x, \cdot|y, r_B)$, $OUT_B^{A,B}(x, r_A|y, \cdot)$, and $OUT_B^{A,B}(x, \cdot|y, \cdot)$;

In all above quantities the superscript (A, B) will sometimes be omitted when clear from the context. When we do not wish to explicitly consider the random coins of the participants, we will omit them.

Polynomial-Time Protocols

A protocol (A, B) is called *polynomial time* if there is a fixed polynomial P such that, for all $k \in \mathcal{N}$, in every execution in which the length of both private inputs is $\leq k$, the number of steps taken by both A and B in that execution is $\leq P(k)$.

Security parameters

If k is a positive integer, we denote by 1^k the unary representation of k (i.e., the string consisting of k 1-symbols). We say that an execution of a protocol (A, B) has *security parameter* k if the private input of A is of the form $(1^k, x)$ and the private input of B is of the form $(1^k, y)$. (Thus, *de facto* 1^k is a "common input" while x and y are the "real private inputs.")

References

[1] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowedge complexity of interactive proof systems. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 291–304, 1985. Superseded by journal version.

[2] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital-signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, April 1988.