Preliminaries

Lovesh Harchandani

5th January 2018, version 0.1

1 Terminology

Some preliminary concepts need to be defined; a basic understanding of set theory and number theory is required, set theory [1] and group theory [2] is required.

Throughout, $\mathbb{G}$ will always denote a finite multiplicative group with τ-bit prime order q and a fixed generator $g \in \mathbb{G}$ and $\mathbb{G}^* = \mathbb{G} \setminus \{1\}$ will refer to its subset of non-identity elements. Likewise, $\mathbb{Z}_q$ will denote the field of integers modulo q, and $\mathbb{Z}_q^* = \mathbb{Z}_q \setminus \{0\}$ will refer to its multiplicative group of units.

If Λ is a finite set, then the set of all length-n sequences of elements from Λ is denoted by Λ^n and the set of all finite sequences (of any finite, non-negative length) of elements from Λ is denoted by $\Lambda^* = \bigcup_{n \in \mathbb{N}} \Lambda^n$. The sequences in Λ^* are called finite strings over the alphabet Λ and subsets of Λ^* are called languages of strings over Λ. Let S and W be arbitrary languages over Λ. A collection of ordered pairs $R \subseteq S \times W$ is a (binary) relation on strings from Λ^*. We call the language of strings $L_R = \{ \ s \in S \mid \exists$w $\in$ *W*, *(s, w)* $\in R$ } the language induced by *R*. We treat *R(s , w)* as a function evaluating to 1 if *(s, w)* $\in R$ and to 0 otherwise. If *R(s, w)* = 1, then the string w is called a witness that s $\in L_R$. Given a string s $\in$ S, the set W_R*(s)* = { *s* w $\in$ W |

R(s, w) = 1 } is called the witness set for s $\in L_R$. Of course, W_R*(s)* = {} if s ($\notin$) L_R.
An algorithm whether it be deterministic or probabilistic is efficient if its expected running time is polynomial in the length of its inputs.
Throughout, we assume that there are efficient algorithms to test membership of arbitrary strings in S and W so that identifying instances for R is easy.
We make no assumptions about the ease or difficulty of testing membership of a string s in L_R when no appropriate witness w $\in$ W_R*(s)* has been provided.

NP-relations and NP-languages. A language L_R is called an NP-language if it belongs to the complexity class NP; that is, if (i) there exists an efficient algorithm to evaluate *R(s, w)* on any instance *(s,w)* $\in$ *S* ($\times$) *W*, and (ii) there exists a polynomial time function *p(n)* such that every s $\in L_R$ has at least one witness w $\in$ W_R*(s)* satisfying $|w| \leq p(|s|)$. Any witness w $\in$ W_R*(s)* that satisfies the latter bound is called an NP-witness that s $\in L_R$. If L_R is an NP-language, then we call the relation R an NP-relation. Viewing W_R*(s)* as the set of proofs that s $\in L_R$, we can interpret NP as the class of languages whose strings each have one or more "short" proofs of

[1] https://en.wikipedia.org/wiki/Set-builder_notation, https://www.tutorialspoint.com/discrete_mathematics/discrete_mathematics_relations.htm, https://www.tutorialspoint.com/discrete_mathematics/discrete_mathematics_functions.htm

[2] https://www.tutorialspoint.com/discrete_mathematics/discrete_mathematics_group_theory.htm

membership that can be checked in polynomial time.

Interactive protocols. A protocol is a system of rules describing the sequence, syntax, and semantics of message exchange between two or more interactive algorithms where each exchange is called a move and 2 consecutive moves constitute a round. Protocols comprising just one move are called non-interactive protocols and those comprising two or more moves are called interactive protocols. We will be dealing with two-party protocols called interactive proof systems, which involve a pair of interactive algorithms that play two distinct roles: one algorithm is the prover and the other algorithm is the verifier. The "honest" prover and verifier algorithms are denoted by P and V respectively, potential "dishonest" impostors, P^* and V^* denote arbitrary algorithms taking on the prover and verifier roles in the protocol. If P is an interactive algorithm, then P(w) denotes P given the string w $\in \Lambda^*$ as its private auxiliary input. If V is a second interactive algorithm, then (P, V) denotes the protocol arising when P interacts with V, and $\langle$P, V$\rangle$(s) denotes the random variable describing the output of V in such an interaction when the common input string is s $\in$ S. P always makes the final move in an interactive proof, after which V checks one or more verification equations to decide whether it should accept or reject the interaction. $Pr[1 \leftarrow \langle P, V \rangle (s)]$ denotes the probability that "V accepts" and $Pr[0 \leftarrow \langle P, V \rangle (s)]$ denotes the probability that "V rejects"
Subsequent definitions assume an alphabet Λ, S and W as infinite subsets of Λ^*, an infinite relation $R \subseteq S \times W$ and an NP language L_R

Interactive proof.