T-79.511 Special Course on Cryptology / Zero Knowledge: IP=PSPACE

Lauri Tarkkala

October 1, 2001

1 Introduction

The complexity of decision problems (usually formally modeled as language recognition) when solved by a classical deterministic, non-deterministic or probabilistic Turing Machine have been categorized into a relatively well-understood and well-known complexity hierarchy, including classes of problems such as **P**, **NP**, **coNP**, **PH** and **PSPACE**. [1].

The class of decision problems, for which a prover P can convince a verifier V of only correct solutions using an interactive protocol and with a defined margin of error, is called **IP**.

The verifier V (or Victor) is limited to probabilistic polynomial time computation and the prover P (or Peggy) can be considered to be computationally unbounded, but can be limited to computations bounded by polynomial space, without restricting the class **IP** further. [1] [4]

This survey considers the proof culminating in the result **IP** = **PSPACE**.

2 The class PSPACE

A rigorous definition of the class **PSPACE** requires a thorough consideration using Turing machines, and becomes quite lengthy. The definition of a Turing machine used here is the one shown in [1] which contains a comprehensive treatment on the subject.

The class **PSPACE** is the class of all languages, for which there exists a deterministic Turing machine, which can decide the language using an amount of space on it's working tape which is bounded by a polynomial with respect to the length of the input.

3 Interactive Proof Systems

An interactive proof system is a formalism used to define the computation and interaction performed by Peggy the Prover and Victor the Verifier. The interactive proof systems allows Peggy (P) to convince Victor (V) that $x \in L$ for some language L. [5]

Definition 3.1 Let Σ denote an alphabet, which does not contain the symbol $\#$ and let $\Sigma' = Sigma \cup \{\#\}$. Let $L \subseteq \Sigma^*$ be a language over the alphabet Σ.

Let $V : \Sigma^* \times \Sigma^* \times \Sigma'^* \to \Sigma^* \cup \{\text{accept}, \text{reject}\}$ and $P : \Sigma^* \times \Sigma'^* \to \Sigma^*$ be functions representing the computations of Victor and Peggy.

Let $r \subseteq \Sigma^*$ be a random string, which is accessible only by V.

Let $s_i = \#y_1\#z_1\# \cdots \#y_i\#z_i$ such that $y_1 = V(x, r, \varepsilon)$, $z_1 = P(x, \#y_1)$, $y_{i+1} = V(x, r, s_i)$ and $z_{i+1} = P(x, s_{i+1})$.

The string s_i represents the interaction between P and V during the protocol, y_i is a message from V to P and z_i is the reply from P to V.

A pair (V, P) is an interactive proof system with error probability ε, if the following conditions hold:

(i) *Well-formedness* For all x and for all P and a polynomially bounded i, using only a polynomially bounded random string r and such that y_j and z_j for all $j \leq i$ are polynomially bounded, V either accepts or rejects.

(ii) *Completeness* For every $x \in L$

$$\Pr((V, P) \text{ accepts } x) \geq 1 - \epsilon.$$

(iii) *Soundness* For all $P' : \Sigma^* \times \Sigma'^* \to \Sigma^*$ such that (V, P') is well-formed and every $x \notin L$,

$$\Pr((V, P') \text{ accepts } x) \leq \epsilon.$$

Definition 3.2 Let **IP** denote the class of languages that have interactive proof systems with error probability 1/3.

4 IP $\subseteq$ PSPACE

There exist several proofs for **IP** $\subseteq$ **PSPACE** in the literature. This section informally presents a simple sketch of a proof based on the material presented in [4]. An induction was shown to support the same theorem in the previous survey. [5]

Theorem 4.1 IP $\subseteq$ PSPACE

Proof sketch. Let (V, P) be an interactive proof system. Remember that V must always terminate within a polynomial number of steps, given any prover and any random value f. Remember also that after fixing the random value r the behaviour of the protocol is completely deterministic, because the prover can limited to polynomial space computations. [5].

Let $L \subseteq \Sigma^*$ be the language and $x \in L$ the problem instance to be proven.

For a verifier V and the input x, the maximal length of the random number is polynomially bounded. Let R be the set of all valid random numbers for a certain input x. This set can be defined based on the description of the verifier V.

Let a state S of the verifier V be the tuple $S =< r, i, s_i >$ where r is the random number and s_i is the message history defined earlier and i is the number of the round being processed.

For all states s there exist the same finite set M_p of messages that the prover P could send to the verifier V. These messages can easily be defined into a total order (based on e.g. a total order over the alphabet Σ and the message lengths). The set is finite, because the size of all messages bust be polynomially-bounded. This bound depends on the verifier V.

We now construct the Turing machine M for simulating the interactive proof system (V, P).

The machine M keeps two counters, c_{accept} is the number of random variables which lead to V accepting and c_{reject} is the the number of random variables which lead to V rejecting.

The machine M operates iterates over all possible states of V for each random variable $r \in R$. For each state which accepts, the counter c_{accept} is incremented and for each state which rejects, the counter c_{reject} is incremented. If a state $< r, i, s_i >$ either accepts or rejects, then the states S such $S =< r, k > i, s_i\#... >$ are not considered. This limits the amount of states for a random variable to a polynomially bounded amount, due to (V, P) being well-founded.

The machine M after having iterated over the combination of all random variables and execution paths which lead to either acceptance or rejection can compute the ratio of the counters c_{accept} and c_{reject} and based on these either accept or reject the input x.

The machine obviously terminates if (V, P) is well-founded (which it is) and uses only a polynomial amount of space if (V, P) is well-founded (which it is).

■

5 PSPACE $\subseteq$ IP

The proof that **PSPACE** $\subseteq$ **IP** was originally shown by Shamir using arithmetization of simple quantified Boolean formula (QBF) and providing an interactive proof system for convincing a verifier of the value of this arithmetized QBF. [2]

Shen improved and simplified the proof by modifying the arithmetization of QBF used by Shamir and using degree reduction instead of simple QBF's. [3] This section concentrates on the simplified proof by Shen. The arithmethization technique used by Shamir is shown for completeness in Appendix A.

5.1 QSAT

Let φ be a Boolean expression in conjunctive normal form containing the Boolean variables x_1,x_2,...,x_n.

QSAT is the problem of deciding if the closed quantified Boolean formula $\exists x_1 \forall x_2 \exists x_3 ... Q_n x_n \varphi$ is true.

QSAT is **PSPACE**-complete. Therefore any language in **PSPACE** can be expressed as a quantified Boolean formula. [1]

An interactive proof system for QSAT will therefore prove that **PSPACE** $\subseteq$ **IP**.

5.2 Shen's arithmetization

Shen used the following technique to arithmetize a QBF. [3]

Let φ be a Boolean formula in CNF containing the Boolean variables x_1,...,x_n. Create a function $b(x_1, ..., x_n)$ from φ by the following procedure.

(i) Replace each occurence of $\neg\alpha$ with $(1-\alpha)$.

(ii) Replace each occurence of $\alpha \wedge \beta$ with $\alpha \times \beta$.

(iii) Replace each occurence of $\alpha \vee \beta$ with $\alpha + \beta - \alpha \times \beta$.

The value of b equals the value of φ for a truth-assignment, when assuming that "true" = 1 and "false" = 0.

The construction contains one multiplication for every $\wedge$ and $\vee$ operator. The degree of any variable in b is bounded above by the amount of such multiplications + 1, and hence the length of φ.

As an example let $\varphi = (x_1 \vee x_2) \wedge (x_3 \vee \neg x_2)$. Then the arithmetized function b is $b(x_1, x_2, x_3) = x_1 - x_1 \times x_2 + x_1 \times x_2 \times x_3 + x_2 - x_2^2 + x_2^2 \times x_3 - x_1 \times x_2 + x_1 \times x_2^2 - x_1 \times x_3 \times x_2^2$.

Shen's proof does not as such arithmethize quantified Boolean formula, but instead transforms the polynomial functions depending on how they are quantified.

The following transforms are used to transform a polynomial P in accordance with how φ was quantified.

- $(AxP)(x, y_1, \cdots, y_n) = P(0, y_1, \cdots, y_n) \times P(1, y_1, \cdots, y_n)$.
- $(ExP)(x, y_1, \cdots, y_n) = P(0, y_1, \cdots, y_n) + P(1, y_1, \cdots, y_n) - (AxP)(x, y_1, \cdots, y_n)$.
- $(Rx_kP)(x_1, \cdots, x_n) = P \pmod{x_k^2 - x_k}$, $1 \leq k \leq n$.

AxP and ExP remove a Boolean argument from the polynomial, emulating the universal and existential quantifiers, respectively. RxP reduces the degree of all instances of the Boolean variable x_k in the polynomial P to at most one.

The transforms preserve the truth-value. AxP and ExP clearly preserve the truth-value of a quantified boolean formula, when simulating the universal and existential quantifiers respectively. The degree reduction performed by Rx_kP does not affect the value of the expression when the instances of the formal parameter are substituted for by 0 or 1.

5.3 Interactive Sub-protocol β

Let $S(x_1, \cdots, x_n)$ be a polynomial over a finite field F.

Let there be an interactive protocol α for proving that $S(x_1, \cdots, x_n) = v$ for any inputs $x_1, \cdots, x_n, v \in F$, with probability 1 if the assertion is correct and with error probability ε if the assertion is false.

P can convince V that $U(y_1, \cdots, y_l) = e$ using the following protocol β, which relies on protocol α. $U(y_1, \cdots, y_l)$ is constructed from $S(\cdots)$ using the procedures Ex, Ax or Rx shown in the previous section.

(1) Let $\deg S < d$. Let d be known to both P and V

(2) P sends V the polynomial $s(x) = S(x, c_1, \cdots, c_l)$

(3) If $\deg s > d$ then V rejects.

(4) The fourth step depends on how the polynomial U relates to S.

Case A. $U(y_1, \cdots, y_l) = (AxS)(x, y_1, \cdots, y_l)$
If $s(0) \times s(1) \neq e$ then V rejects.

Case E. $U(y_1, \cdots, y_l) = (ExS)(x, y_1, \cdots, y_l)$
If $s(0) + s(1) - s(0) \times s(1) \neq e$ then V rejects.

Case R. $U(x, y_1, \cdots, y_l) = (RxS)(x, y_1, \cdots, y_l)$
If $s(0) + (s(1) - s(0)) \times x \neq e$ then V rejects.

(5) V chooses a random element $r \in F$ and sends it to P.

(6) P must now convince V using protocol α that $S(r, c_1, \cdots, c_l) = s(r)$.

If another parameter instead of the first is chosen as the indeterminate in the polynomial $s(x)$ and as the parameter for the (RxS) operation, the protocol remains essentially the same. The polynomial $s(x)$ is now constructed in an obvious fashion leaving another parameter as the indeterminate and step 4R considers this indeterminate parameter instead of the first parameter.

If $U(x)$ has been constructed from $S(x)$ using one of the three transforms, then obviously step 4A, 4E or 4R will not reject and algorithm α will succeed with probability 1 as per it's definition. The protocol is therefore complete.

If steps 4A, 4E or 4R do not reject, then either $s(x)$ is another polynomial with behaviour similar to that of $S(x)$ or it is $S(x, c_1, \cdots, c_l)$, therefore it remains to check that $S(x, c_1, \cdots, c_l) = s(x)$ to confirm the assertion.

To perform this last step, V chooses a random number r and then P must convince V that $S(r, c_1, \cdots, c_l) = s(r)$ to complete the proof.

If P has chosen the incorrect polynomial $s(x)$ in the protocol, then there are by the interpolation theorem at most d points in the field F for which $s(x) = S(x, c_1, \cdots, c_l)$.

The error probability for protocol β in the case $U(c_1, \cdots, c_l) \neq e$ is therefore $\varepsilon + d/|F|$. The protocol is therefore sound, provided that the field F is sufficiently large.

5.4 Interactive proof system for QSAT

The tools to define an interactive proof system for QSAT have now been described.

Let $\varphi = Q_1 x_1 \cdots Q_n x_n B(x_1, \cdots, x_n)$ be a QBF. Let $b(x_1, \cdots, x_n)$ be the function resulting from arithmethizing B.

P reduces the QBF φ to a constant $e \in \{0, 1\}$ (depending on the truthfulness of the QBF φ), using the following algorithm.

0 Let $i = n$.

1 For all variables x_k such that $1 \leq k \leq i$ in b perform the following operation $b(\cdots) = (Rx_k b)(\cdots)$.

2 If $Q_i = \forall$ then let $b(\cdots) = (Ax_i b)(\cdots)$.

3 If $Q_i = \exists$ then let $b(\cdots) = (Ex_i b)(\cdots)$.

4 $i = i - 1$.

5 If $i > 0$ then goto 1.

The algorithm reduces the degree of all variables in $b(\cdots)$ to 1 in step 1 and the steps 3 and 4 raise the degree of a variable in $b(\cdots)$ to at most 2, before the degrees are reduced to 1 in the next iteration.

The number of operations performed by the algorithm is exactly $n + \Sigma_{i=1}^{n} i = (n^2 + 3n)/2$.

A prover P must now convince a verifier V that the constant e output by the previous algorithm is reduced from the original QBF φ.

P and V both know the original QBF φ and the arithmethized form of $B(\cdots)$. Formally P is trying to convince V that φ is in the language of all true QBF's.

The first step for P is to convince V that $U = e$ when S is a function of x_1 using protocol β. Protocol β then uses itself as the "protocol α", and backtracks through the transforms performed in the algorithm earlier untill it can no longer do so.

At this point P has to convince V that $b(u_1, \cdots, u_n) = v$ for some $u_1, \cdots, u_n, v$ which are determined by the protocol while backtracking. The verifier V can do this as it knows both $B(\cdots)$ and $b(\cdots)$.

The error probability is determined by the number of operations performed in the algorithm above and the size of the field F the transforms are performed in.

The number of operations is linear with respect to n^2 and the maximal degree is at most n. Therefore it is sufficient to use a field size which is $\mathcal{O}(n^4)$, to let the error probability shrink as n grows.

The prime p for which $|F| = p$ can therefore be logarithmic in length with respect to n.

Theorem 5.1 PSPACE $\in$ IP

Proof. The protocol described above is clearly a well-formed, complete and sound interactive proof system with an error-probability which can be made as low as is desired. The protocol solves a **PSPACE**-complete problem (QSAT) using an interactive proof system, and hence **PSPACE** $\in$ **IP**. ∎

Corollary 5.2 IP = PSPACE

Proof. Together with the **IP** $\in$ **PSPACE** theorem this implies the result. ∎

References

[1] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

[2] A. Shamir. IP = PSPACE. *J. ACM* **39** (1992) no. 4, 869–877.

[3] A. Shen. IP = PSPACE: Simplified proof. *J. ACM* **39** (1992) no. 4, 878–880.

[4] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudo-randomness.* Springer Verlag, Berlin, 1999.

[5] P. Kaski. Tik-79.511 Special Cource on Cryptology / Zero Knowledge: Rudiments of Interactive Proof Systems. XXX: HTTP HREF: HERE

6 Appendix A: Shamir's arithmethization

Shamir used the following technique to arithmetize a QBF. [2]

(i) Replace each Boolean variable x_i by a variable z_i which ranges over $\mathbb{Z}$.

(ii) Repace each occurence of $\neg z_i$ by $(1 - z_i)$.

(iii) Replace each occurence of $\wedge$ by integer multiplication $\times$.

(iv) Replace each occurence of $\vee$ by integer addition $+$.

(v) Replace each occurence of an universal quantifier $\forall x_i$ by the integer product $\Pi_{z_i \in \{0,1\}}$.

(vi) Replace each occurence of an existential quantifier $\exists x_i$ by the integer sum $\Sigma_{z_i \in \{0,1\}}$.

As an example, the QBF $\exists x_1 \forall x_2 \exists x_3 (x_1 \vee x_2) \wedge (x_3 \vee \neg x_2)$ would become the integer value $y = \Sigma_{z_1 \in \{0,1\}} \Pi_{z_2 \in \{0,1\}} \Sigma_{z_3 \in \{0,1\}} (z_1 \times z_3 + z_1 - z_1 \times z_2 + z_2 \times z_3 + z_2 - z_2^2)$.

If the value of the original QBF is true, then the value y of the arithmethized QBF is non-zero. If the value of the original QBF is false, then then value of the arithmethized QBF is zero.