Partial knowledge transfer and almost fair exchange of secrets

Raphael Machado, Davidson Boccardo, Vinícius de Sá, Jayme Szwarcfiter

August 30, 2015

Abstract

Partial knowledge transfer refers to the ability of "proving" to third parties that a set of bits if part of the solution of a problem in such a way that the third party learns nothing more than this set of bits. The concept of partial knowledge transfer was recently defined and applied to the secure exhibition of digital watermarking [5]. In the present work we propose a novel partial knowledge transfer scheme that allows a party to exhibit part of the solution of any NP problem. The method is then applied to de development of fair exchange of secrets protocol. Our contributions are manifold:

1. We improve the partial knowledge transfer scheme so that it can rely only on the validity of the factoring assumption.
2. We show the equivalence between two variants of the factoring assumption.
3. We devise a partial knowledge transfer scheme that allows to securely exhibit part of the solution of any NP problem.
4. We propose the use of the improved partial knowledge transfer scheme in an "almost fair" protocol for the exchange of secrets.

Keywords: security protocols; fair exchange; zero-knowledge proof; factoring assumption.

1 Introduction

Fairness is an important requirement in exchange electronic protocols. It arises when two parties are willing to exchange digital items, but do not trust each other. In order to avoid that some of the parties interrupt the protocol just after receiving the desired digital item, it is important that the "exchange" process is "atomic", i.e., all the items are exchanged at once. Some examples will help to clarify the concept.

Contract signing. Suppose Alice and Bob agreed to sign a contract by which Alice will sell her company to Bob by a price of US$500,000.00. If Alice just sign a term sheet and send it to Bob, then Bob could wait some days (weeks, months,...) and see how the company behaves. If the company goes well — for instance, the profits increase — Bob can sign the contract and buy the company. If, however, the company profits go down, then Bob can simply rip the contract and forget he once tried to make business with Alice. An analogous problem would happen if Bob signs first.

Certified message. Suppose Alice is willing to send some information to Bob, but this information is sensitive, so that is important that Alice receives back from Bob a receipt evidencing that Bob indeed had access to that information. Such requirement is typical from scenarios where Alice needs to share some confidential intellectual property with Bob (in case of disclosure of the intellectual property, Alice can prove that Bob had access to that information). Once again, since Bob is not trusted, Alice can not simply send the secret and "hope" that Bob will send a receipt in turn; Bob could simply take the information and send nothing back. Conversely, Bob cannot sign a receipt of something he did not yet receive.

Selling of secret. Alice is the owner of a logistics company and Bob discovered a new truck routing transportation scheme that would allow Alice to save 5% of the costs. Alice agree to pay a million of dollars for such information, but Bob can not simply send the "solution" and wait that Alice be honest and sign a check of one million of dollars in name of Bob. Analogously, Alice will not give so much money to Bob without being sure she will receive the solution to her company problems.

Exchange of secrets. In a similar way, Alice has a secret that interests to Bob, and Bob has a secret that interests to Alice. They are willing to exchange the secrets, but no one wants to be the first to deliver the information.

In all the above scenarios, the party that takes the first step — signing a document or sending an information — is in clear disadvantage face to the other party. It is desirable that no one has to "make the first step", and that the involved items are exchanged all at once. This is what we mean by a *fair* protocol, i.e., no party can gain an advantage over the other parties by misbehaving, misrepresenting or by prematurely aborting the protocol. The easiest way of assuring that an exchange is fair in this sense is by recurring to a Trusted Third Party (TTP): in the first stage both Alice and Bob deliver their items to the TTP who will exchange the items in a second stage. The scheme is secure, as long as the TTP is honest and unquestionable. However, such a protocol has the several disadvantages of any protocol that requires the involvement of a third party in each and every exchange. Hence, a lot of effort has been done in the development of more practical fair exchange protocols.

It is important to mention that a "perfect" fair exchange — i.e. a deterministic protocol that guarantees that two items are simultaneously exchanged — is impossible to achieve in a two-parties protocol without a TTP. The fact is that the interaction between the parties involved in a protocol is "discrete", in the sense that, at each step, a message is send in one direction. Hence, at each step of a protocol, only one party can gather some information, and is impossible that two parties gain information simultaneously. So, different approaches has been taken in order to achieve some level of fairness in exchange protocols. Such approaches involve the use of distinct types of trusted third parties — either online or offline —, the idea of limiting the advantage each party can have over the other — such as the probability of being bound to a contract —, or gathering evidences about the execution of the protocol, so as to allow an *a posteriori* punishment of the party that do not follow the protocol, as we discuss below.

In the present work, we show that the recently proposed concept [5] of "partial knowledge transfer" can be used to achieve "almost fairness" in exchange protocols. The proposed protocol [5] for partial knowledge transfer allows to prove that a set of binary digits is the prefix of one of the factors of a number. The goal [5] is to use the partial knowledge transfer to prove that a software was watermarked, without needing to exhibit the precise position of the watermark (because this could allow the removal of the watermark). In the present work we construct an analogous scheme that allows to gradually exhibit the solution of any NP problem. The idea is to "encode" the solution of the NP problem as a "sub-solution" of a hypothetically harder problem — in our case, the factoring of a composite number that is the product of two large prime numbers. In practice, we use polynomial-time reductions to convert any NP-complete problem to a logic satisfiability problem whose solution is encoded as a bitstream that is the prefix of a prime number p which in turn is multiplied by another prime number q, obtaining $n = pq$. Then, a partial knowledge transfer scheme is executed to transmit the bits of q one at a time, until n can be factored and p is recovered — hence the solution of the satisfiability problem and therefore of the original NP problem. We say that our protocol achieves "almost fairness" in the sense that, in some moments, one of the parties is "one bit ahead of the other party".

2 Background

2.1 Related Work

Fair exchange protocols have been studied in the context of electronic mail certified delivery, digital signatures exchange, contract signing, and exchange of documents. One key aspect that distinguishes the several proposed fair exchange protocols is whether they guarantee *a priori fairness* — also called *strong* fairness — or they rely on gathering of information that allow an *a posteriori dispute resolution* by an adjudicator, in case one of the participants do not behave properly — also called *weak* fairness. One interesting aspect that makes the problem of exchange of secrets particularly challenging is that other exchange protocols — usually involving digital signatures — assume the existence of a third party (notary, lawyer, judge) that will define if a signature, a document, a check or a contract is valid. In other words, people sign documents because the signature is a proof to be shown to a notary, a lawyer or a judge. In the case of the exchange of secrets, the exchange is eminently two-party: Alice has an information that interests to Bob and Bob has an information that interests to Alice. It can even be the case that the exchanged information is invaluable, and there is nothing to do face to a judge if you already lost such an information.

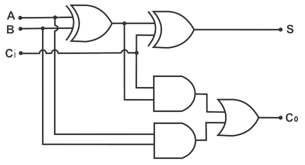
A classic method for the fair exchange of secrets in the one of Blum [1], which show how two parties, each one having the knowledge of a factor of a distinct composite number, can exchange their factors bit by bit. The method of Blum allows that each bit is sent together with a "proof" that it is a bit of a factor of the composite number. The fact that the factors are exchanged bit by bit guarantees that each party has a limited advantage over the other during the execution of the protocol. The fairness is guaranteed as long as the parties have comparable computing power (for otherwise, one of the parties — say, the one that can run 2^{60} operations within a reasonable time — could stop the protocol when there are 60 bits remaining).

The present work is closely related to the method of secret key exchange of Blum [1]. Indeed, the method of Blum [1] allows to parties to exchange prime factors of composite numbers "bit by bit", in such a way that each transmitted bit goes together with a proof that it is indeed a bit of a factor of the composite number. Our method also rely on the "bit by bit transmission" of factors, but the scheme allows to prove not only that the bits are indeed bits of a factor, but also that one of the factors of the composite number somehow "encode" the solution of an NP problem. Our method is therefore much more general, allowing the exchange of any secret that is the solution of an NP problem.

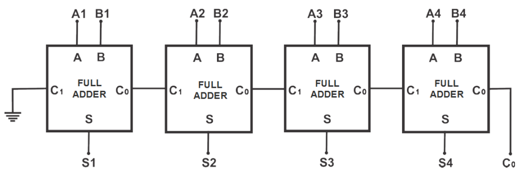
2.2 Partial Knowledge Transfer

In Kilian's doctoral thesis [4] the following problem is described. Bob wants to factor a number n of 500 bits which is known to be the product of five prime numbers of 100 bits. Alice knows one of the factors, denoted q, and is willing to sell 25 bits to Bob. Kilian proposes a method that allows Bob to be sure that Alice indeed knows one of the factors of n, and it still allows that assurance happens upon individual bits of q. The proposed scheme not only allows the disclosure of some bits of q but also uses commitment schemes for individual bits of q in order to ensure that these bits will not be disclosed without the consent of Alice. Finally, it allows the use of *oblivious transfer* in a way Alice is unaware of bit sets actually disclosed.

Machado et al. [5] describes a simple scheme by which one can prove that a set of bits is indeed one of the factors of a composite number. Such scheme preclude the use of commitment schemes or oblivious transfer, being based on the idea that the multiplication of primes can be seen as a logical operation, where each bit of the product is indeed a logical function over the bits of the factors. For the sake of completeness, we briefly describe Machado et al. [5] partial knowledge transfer in what follows.



(a) Full adder.



(b) 4-bits Adder.

Figure 1: Building an adder.

EQUICOMPOSITE problem

We claim that the problem of determining if a number is composite can be reduced to the problem of determining if a logical expression is satisfiable — the well-known SAT Problem [3]. More precisely, we consider the EQUICOMPOSITE variant of the factorization problem. The EQUICOMPOSITE Problem asks whether an integer n is *equicomposite*, i.e., n can be written as a product of two factors, each one with the most $\lceil log_2(n)/2\rceil$ bits.

EQUICOMPOSITE
Input: binary number n, with $\lceil log_2(n)\rceil$ bits.
Output: YES, if n is the product of two number with bit size up to $\lceil log_2(n)/2\rceil$;
NO, otherwise.

To deal with the EQUICOMPOSITE problem using zero-knowledge proofs, we review the implementation of the multiplication operation using combinational circuits, or, equally, using logical expressions involving the bits of the operands.

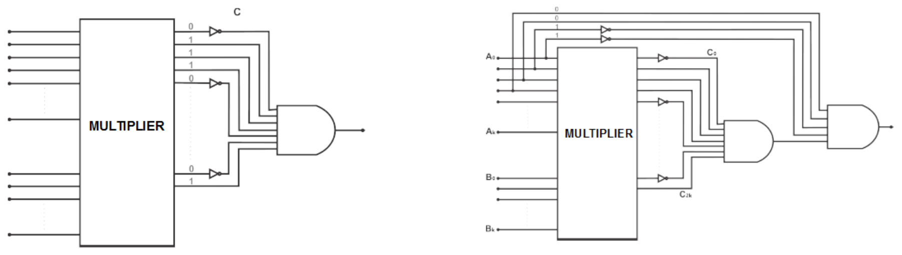
Product of integers as a logical function

It is well-known the product operation of two binary numbers can be described as a combinational circuit, being each digit of the result a logical function over the digits of the operands. Indeed, the multiplication operation over binaries can be obtained by a sequence of additions and multiplications by two.

Adding bits. It is easy to implement a combinational circuit that receives as input two bits A and B (the operands) and a third bit C_i, the carry (generated by a adder in the previous stage), and returns as output the bit S resulting of the sum of the three inputted bits and a new bit of carry C_o (Figure 1(a)). Observe that both bits S and C_o can be described as a logical expression applied over the bits A, B and C_i: $S = (A \oplus B) \oplus C_i$ and $C_o = (A \cdot B) + (C_i \cdot (A \oplus B))$. Naturally, the XOR ("exclusive or", denoted by $\oplus$) may be replaced by operations OR ($+$) and AND ($\cdot$), according to the formula $A \oplus B = \bar{A}B + A\bar{B}$.

Chained full adders. To do the sum of binary numbers with more than one bit, we simply chain full adders, always sending the output carry bit of a stage to the input of the next stage (Figure 1(b)).

Multiplying by power of two. Multiplying a binary number by two



(a) UNI-MULT: fixing the output bits with a final AND port.

(b) PRE-MULT: fixing the leftmost bits of A as "1100".

Figure 2: Converting factorization problems into logical ones.

is basically to perform a "shift left", i.e., to add a bit zero at the right of the number that is being multiplied. We denote the left shift of i bits (multiplication by 2^i) of a binary number B by $B << i$.

Product. Finally, to multiply $A = A_3A_2A_1A_0$ by $B = B_3B_2B_1B_0$, we start with the rightmost bit of one of the operands, say, A. If the bit A_0 is 1, so we add the value of the other operand, B, to the result C (initially zero); if the bit A_0 is 0, no value is added. For each one of the consecutive bits A_i of A, if and only if $A_i = 1$, we do a shift left of size i on B (i.e., we multiply B by 2^i) and we add it to the result. The result, written in a logical expression is equivalent to $C = B \wedge A_0 + (B << 1) \wedge A_1 + (B << 2) \wedge A_2 + (B << 3) \wedge A_3$.

Transforming EQUICOMPOSITE into a logical problem

Once we know how to describe the product of two binary numbers in the form of a combinational circuit, it is easy to adapt it to a modified circuit that has a single output bit whose value is 1 if and only if a certain number n is equicomposite. We need to add NOT ports to each output of the multiplier circuit related to one bit of n that must be 0, and connect all the outputs to a single AND port.

Formally, a circuit UNI-MULT(d, n), where d is an integer and n is a binary number, is built as shown in Figure 2(a), with a multiplier circuit of two binary numbers of d bits, with a NOT port in each output of the multiplier, related tone bit 0 of n, and with a AND port connecting all the $2d$ outputs (inverted or not). The result is given by the following Theorem 2.1, whose proof is omitted due to its simplicity.

Theorem 2.1. *Circuit UNI-MULT(d, n) returns bit 1 if and only if the binary number n can be written as a product of two binary numbers up to d bits each.*

The PREFACTOR problem

We consider the problem of determining if a number can be written as a product of two other numbers of equal bitsize, one of which has part of the bits known.

Consider the following decision problem, which we denote PREFACTOR.

PREFACTOR
Input: binary numbers n and k.
Output: YES, if n is the product of two number with bit size up to $\lceil log_2(n)/2 \rceil$, one of which has k as prefix;
NO, otherwise.

We can reduce the PREFACTOR problem to SAT. Build a circuit similar to the one in Figure 2(a) but "transferring" of some input bits directly to the last stage of the circuit, which receives an additional AND port as illustrated in Figure 2(b).

Formally, a circuit PRE-MULT(d, n, k), where d is integer and n and k are binary numbers, is built as shown in Figure 2(b). Initially, we have a circuit UNI-MULT(d, n). For each input bit of the circuit UNI-MULT(d, n) related with a bit of k, we derive it and connect it to a NOT port if such bit is 0 in k. The derivations are all connected to a AND port, as well the output bit of the circuit UNI-MULT(d, n). The Theorem 2.2 sums up what the circuit PRE-MULT allows to do.

Theorem 2.2. *The circuit PRE-MULT(d, n, k) returns the bit* 1 *if and only if the binary number n may be written as a product of two binary numbers up to d bits, and one of them having k as its prefix.*

Converting to the conjunctive normal form

The reader will observe, again, the output of the circuit PRE-MULT(d, n, k) is a logical function upon the input bits. However, in order to use the framework of complexity theory and its polynomial reductions, it is necessary to have a logical expression in the conjunctive normal form. Fortunately, the transformations of Tseitin [6] allows to build, from any logical expression σ, a new logical expression σ' whose size is linear in the size of σ. Moreover, the transformation is executed in linear time in the size of σ.

Using zero-knowledge proofs

Knowing how to reduce the problem PREFACTOR to SAT, we can simply recur to zero-knowledge proofs with polynomial reductions. We can, for example, to reduce a SAT instance to a 3-COLORING instance in polynomial time [3], so then to use a classical scheme of zero-knowledge proof for this last problem [2].

3 Almost Fair Exchange of Secrets

3.1 Preliminary results

We start this section by stating the assumptions over which we construct our method, as well as deriving the basic consequences of such assumptions which will be relevant to us.

A modified factoring assumption

We propose a modified, formally stronger, version of the factoring assumption. The result is formally stated in Theorem 3.1 and its intuition is that, if a problem is "hard", then it remains hard even if "part" of the solution is revealed. We understand that such result, though being prove with a straightforward method, has interest by its own, as long as it gives intuition about the "hardness" of computational problems. We define, in the following, the two versions of the "Factoring Assumption" which we later prove to be equivalent.

Factoring Assumption. For any positive polynomial $r(\cdot)$ and any probabilistic polynomial time algorithm A the following holds for k sufficiently large

$$Pr[A(n) = (p, q)] \leq \frac{1}{r(k)},$$

where $n = pq$ and p, q are random k-bit primes.

Factoring Assumption With t Revealed Bits. For any positive polynomial $r(\cdot)$ and any probabilistic polynomial time algorithm A' the following holds for k sufficiently large

$$Pr[A'(n, p_0, p_1, ..., p_{t-1}) = (p, q)] \leq \frac{1}{r(k)},$$

where $n = pq$, p, q are random k-bit primes, and $p_0, p_1, ..., p_{t-1}$ are the less significant bits of p.

Theorem 3.1. *For any fixed t, the Factoring Assumption imply the Factoring Assumption With t Revealed Bits.*

Proof. Suppose that the Factoring Assumption With t Revealed Bits does not hold. Hence there exists a probabilistic polynomial time algorithm A' such that $Pr[A'(n, p_0, p_1, ..., p_{t-1}) = (p, q)] \geq \frac{1}{r(k)}$.

Consider the algorithm described in Algorithm 3.2:

Algorithm 3.2. Factorization
Input: binary number $i = i_0 i_1 ... i_{t-1}$
Output: Factors (p', q') of i or the information that "i is composite"

```
    for $i = 0, ..., 2^k - 1$
        let $(p', q') = A(n, i_0, i_1, ..., i_{t-1})$
        if $n = p' \cdot q'$ then return $(p', q')$
    else return "$i$ is composite"
```

Note that k is fixed, so that the **for** loop is iterated a constant number of times, and the above algorithm is probabilistic polynomial time.

Observe that for precisely one choice of i, the bits of i are the less significant bits of p, and by hypothesis the algorithm $A(n, i_0, i_1, ..., i_{t-1})$ finds the factors of n with non-negligible probability. Hence, with non-negligible probability, the probabilistic polynomial time algorithm A returns the factors of n, and the Factoring Assumption does not hold (a contradiction). $\square$

Building blocks

We define three basic blocks that will be used to build the logic circuit that allow the transference of information that is related to a "secret", i.e., the solution of an NP problem. The framework formalizes and extend the idea of Partial Knowledge Transfer.

An *AND function* $A_k : \{0,1\}^k \to \{0,1\}$ is such that $A_k(x_0, x_1, ..., x_{k-1}) = 1$ if and only if $x_0 = x_1 = ... = x_{k-1} = 1$.

A *Product function* $P_k : \{0,1\}^{2k} \to \{0,1\}^{2k}$ is a function that receives as input the binary representation of two k-bits number and outputs the binary representation of their product.

A *Mixing function* $M_B : \{0,1\}^k \to \{0,1\}^k$, where $B = \Sigma_{i=0}^{k-1} B_i$ is a binary number on k bits $B_0, ..., B_{k-1}$, is a boolean function such that $M_B(x_0, x_1, ..., x_{k-1}) = (\overline{x_0 \oplus B_0}, \overline{x_1 \oplus B_1}, ..., \overline{x_{k-1} \oplus B_{k-1}})$. Note that M_B is a bijective function that outputs a stream of 1's precisely in case the input is B.

3.2 The construction

Say Alice knows two k-bit factors p and q of a binary number n. She wants to prove to Bob that a set of bits $x_0, ..., x_{t-1}$ correspond to the t most significant bits of q. The Partial Knowledge Transfer concept provide a method to do so. She proceeds as follows.

1. Construct the boolean function $\psi(p_0, ..., p_{k-1}, q_0, ..., q_{k-1}) :=$ $A_{2k+t}(M_n(P(p_0, ..., p_{k-1}, q_0, ..., q_{k-1})), M_{x_0,...,x_{t-1}}(q_0, ..., q_{t-1}))$, on the boolean variables $p_0, ..., p_{k-1}, q_0, ..., q_{k-1}$.

2. Convert $\psi(p_0, ..., p_{k-1}, q_0, ..., q_{k-1})$ to a boolean function $\psi'(p_0, ..., p_{k-1}, q_0, ..., q_{k-1})$ in conjunctive normal form using Tseitin transformation [6].

3. Prove that ψ' is satisfiable using a zero-knowledge proof scheme.

Note that the above procedure just formalize the method depicted in Figure 2(b). Now we formally prove that the method allows Alice to prove Bob that $x_0, ..., x_{t-1}$ correspond to the less significant bits of a k-bit factor of n.

Theorem 3.3. *The scheme described in the present section provides a proof that $x_0, ..., x_{t-1}$ correspond to the less significant bits of a k-bit factor of n.*

Proof. The zero-knowledge proof performed in Step 3 allow Bob to be confident that ψ is satisfiable, and because ψ and ψ' are equivalent (by Tseitin transformation), ψ is satisfiable as well. By the construction of ψ, we know that the output bits of $M_n(P(p_0, ..., p_{k-1}, q_0, ..., q_{k-1}))$ are all 1 (because these bits are input of an AND Function), hence the output bits of $P(p_0, ..., p_{k-1}, q_0, ..., q_{k-1})$ are the binary representation of n. Therefore, the $p_0, ..., p_{k-1}$ and $q_0, ..., q_{k-1}$ are the binary representations of two numbers whose multiplication results in n, i.e., they represent factors of n. By the construction of ψ, we also know that the output bits of $M_{x_0,...,x_{t-1}}(q_0, ..., q_{t-1})$ are all 1 (once again, because these bits are input of an AND Function), hence $q_0 = x_0$, $q_1 = x_1$,..., $q_{t-1} = x_{t-1}$, i.e., the less significant bits of factor q of n are precisely $x_0, ..., x_{t-1}$. $\square$

Extending to SAT problem

Now we extend the above scheme to allow the encoding of the solution of a SAT problem as a set of bits in a factor of a number. The idea is that, in order to reveal the solution of the SAT problem, one has to factor a large number (or to solve the SAT problem itself). The goal is that the parties use the scheme described in the previous section to gradually reveal the bits of one of the factors of a large number, and when this factor is completely revealed, then the other factor will reveal the solution of the SAT problem.

The idea is simple: just extend the scheme described in the previous section by creating an extra function that is the SAT formula, and make the output of the SAT formula ϕ to be input of the final "AND gate" of the formula (so that the SAT formula must be satisfiable). In order to rely on the Factoring Assumption, the solution of the SAT problem — which is not random — cannot appear directly as part of one of the factors. This is solved by passing the bits of the solution of the SAT problem into a random mixing function.

The precise construction is the following.

1. Chose random bits $r_0, ..., r_{s-1}$ and reveal them

2. Construct the boolean function $\psi(x_0, ...x_{s-1}, p_s, ..., p_{k-1}, q_0, ..., q_{k-1}) :=$ $A_{2k+t+1}(\phi(p_0, ..., p_{s-1}), M_n(P(M_{r_0,...,r_{s-1}}(x_0, ..., x_{s-1}), p_s, ..., p_{k-1}, q_0, ..., q_{k-1})),$ $M_{y_0,...,y_{t-1}}(q_0, ..., q_{t-1}))$ on the boolean variables $x_0, ...x_{s-1}, p_s, ..., p_{k-1}, q_0, ..., q_{k-1}$.

3. Convert $\psi(p_0, ..., p_{k-1}, q_0, ..., q_{k-1})$ to a boolean function $\psi'(p_0, ..., p_{k-1}, q_0, ..., q_{k-1})$ in conjunctive normal form using Tseitin transformation [6].

4. Prove that ψ' is satisfiable using a zero-knowledge proof scheme.

Note that factoring n allows to recover the solution of the SAT problem, namely $M_n(P(M_{r_0,...,r_{s-1}}(p_0, ..., p_{s-1})$.

Formally we denote by $Deliver(\phi, n, r_0, ..., r_{s-1}, q_0, ..., q_t)$ the execution of the above protocol[1].

Theorem 3.4. *The scheme described in the present section, $Deliver(\phi, n, r_0, ..., r_{s-1}, q_0, ..., q_t)$, provides a proof that $y_0, ..., y_{t-1}$ correspond to the most significant bits of a k-bit factor q of n and that the most significant bits $p_0, ..., p_{s-1}$ of $p = n/q$ are such that $\phi(M_{r_0,...,r_{s-1}}(p_0, ..., p_{s-1})) = 1$.*

[1]Note that Step 4 does not specify the acceptable soundness error, i.e., the agreed maximum value on the probability of a proof of a false statement. This is not needed, as long as this can be defined in an *a priori* agreement between the parties.

Proof. The zero-knowledge proof performed in Step 3 allow Bob to be confident that ψ is satisfiable, and because ψ and ψ' are equivalent (by Tseitin transformation), ψ is satisfiable as well. By the construction of ψ, we know that the output bits of $M_n(P(p_0, ..., p_{k-1}, q_0, ..., q_{k-1}))$ are all 1, hence the output bits of $P(p_0, ..., p_{k-1}, q_0, ..., q_{k-1})$ are the binary representation of n. Therefore, the $p_0, ..., p_{k-1}$ and $q_0, ..., q_{k-1}$ are the binary representations of two numbers whose multiplication results in n, i.e., they represent factors of n. By the construction of ψ, we also know that the output bits of $M_{y_0,...,y_{t-1}}(q_0, ..., q_{t-1})$ are all 1, hence $q_0 = y_0$, $q_1 = y_1$,..., $q_{t-1} = y_{t-1}$, i.e., the most significant bits of factor q of n are precisely $y_0, ..., y_{t-1}$. By the construction of ψ, we also know that the output bit of $\phi(p_0, ..., p_{s-1})$ is 1, hence *phi* has a satisfying solution whose truth values $x_0, ..., x_{s-1}$ are such that $x_0 \oplus r_0$, $x_1 \oplus r_1$,..., $x_{s-1} \oplus r_{s-1}$ are the most significant bits of factor p of n. □

3.3 The complete protocol

The protocol described in the previous section allows one to deliver a set of the bits of one of the factors q of a number n at the same time (s)he proves that $p = n/q$ contains the solution of a SAT problem. We propose to use such scheme as a subprotocol that allows to gradually exchange secrets between two parties.

Say Alice knows the solution $x_0^A, ..., x_{s-1}^A$ of a SAT formula ϕ_A on s variables, while Bob knows the solution $x_0^B, ..., x_{s-1}^B$ of a SAT formula ϕ_B also on s variables, and they want to exchange their solutions by gradually and alternately revealing the bits of the solutions.

Initialization:

1. Alice and Bob agree on the size $2k$ of the number n to be factored
2. Alice select s random bits $r_0^A, ..., r_{s-1}^A$ that will "mask" her solution of the SAT formula (so that her prime factors are truly random) and informs these bits to Bob.
3. Alice defines $p_i^A := r_0^A \oplus x_i^A$ for $i = 0, ..., s-1$
4. Alice choses $2k - s$ random bits $p_s^A, ..., p_{k-1}^A, q_0^A, ..., q_{k-1}^A$ in such a way that $p^A := \Sigma_{i=0}^{k-1} p_i^A$ and $q^A := \Sigma_{i=0}^{k-1} q_i^A$ are prime numbers and keep these bits secret.
5. Alice computes $n^A := p^A q^A$, informing n^A to Bob.
6. Bob select s random bits $r_0^B, ..., r_{s-1}^B$ that will "mask" his solution of the SAT formula (so that his prime factors are truly random) and informs these bits to Alice.
7. Bob defines $p_i^B := r_0^B \oplus x_i^B$ for $i = 0, ..., s-1$

8. Bob choses $2k-s$ random bits $p_s^B, ..., p_{k-1}^B, q_0^B, ..., q_{k-1}^B$ in such a way that $p^B := \Sigma_{i=0}^{k-1} p_i^B$ and $q^B := \Sigma_{i=0}^{k-1} q_i^B$ are prime numbers and keep these bits secret.

9. Bob computes $n^B := p^B q^B$, informing n^B to Alice.

10. Bob and Alice agree about the number of rounds (or acceptable soundness error) in the execution of the zero-knowledge proof.

We can now describe the execution of the main protocol.

Protocol Fair Exchange of Secrets

```
for $i = 1, ..., k$
    Execute $Deliver(\phi_A, n, r_0^A, ..., r_{s-1}^A, q_0^A, ..., q_i^A)$ so that Alice deliver to
    Bob the $i$ more significant bits of her prime factor $q^A$.
    Execute $Deliver(\phi_B, n, r_0^B, ..., r_{s-1}^B, q_0^B, ..., q_i^B)$ so that Bob deliver to
    Alice the $i$ more significant bits of his prime factor $q^B$.
```

4 Final Considerations

In the present work, we addressed the problem of developing fair protocols to the exchange of secrets. Recall that, where only to parties are involved in such a protocol, then there is no hope to have a fair protocol, because simultaneity cannot be achieved in such a configuration. Hence, we focus on the approach of "partial delivery" of secrets, i.e., the parties gradually show their secrets — i.e., bit-by-bit, always with a proof that the delivered bit is indeed part os the secret.

Our approach can be applied to any secret that is the solution of an NP-problem, and its security rely on the Factoring Assumption. The key observation that supports our scheme is the fact that Factoring is hard even if some bits of a factor is revealed.

Other important aspect of our scheme, that can not be overemphasized, is the fact that the bits of the "real secret" are never revealed. Recall that $n = pq$ is such that the secret is "encoded" in p, but only bits of q are transferred. The real secret is only revealed when n is factored and p is obtained. This approach is fundamental because is allows us to rely on the Factoring Assumption With T Revealed Bits — note that an analogous assumption does not hold for any NP problem.

We recall, also, that the bit-by-bit exchange of a secret presents a problem that was already registered in the literature. It is the fact that the parties can stop the protocol at any moment and try to factor n by themselves. This can put the party with less computational power in disadvantage (the more powerful party could stop the protocol in a moment where this powerful party can factor n in feasible time but the less powerful party can not). Anyway, in the current times of cloud computing where computational power has a price, it is not unreasonable to assume that both parties have virtually the same computational power.

Future work. In future work, we will investigate whether such Partial Knowledge Transfer schemes can be used in adjudicated protocols, in order to achieve fairness with a third party that only interferes in case of misbehavior of the participants.

Acknowledgements

Partially supported by CNPq and Faperj.

References

[1] M. Blum. *How to exchange (secret) keys.*. ACM Transactions on Computer Systems **2** (1983) 175–193.

[2] S. Goldwasser, S. Micali and C. Rackoff. *Knowledge Complexity of Interactive Proofs*. Proc. 17th STOC (1985) 439–448.

[3] R. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum Press (1972) pp. 85103.

[4] J. Kilian. *Uses of randomness in algorithms and protocols*. MIT Press 1990, ISBN 978-0-262-11153-9, pp. 1-235

[5] R. Machado, D. Boccardo, V. de Sá, J. Szwarcfiter. Fair fingerprinting protocol for attesting software misuses. Proc. 10th International Conference on Availability, Reliability and Security (ARES 2015).

[6] G. S. Tseitin. *On the complexity of derivation in propositional calculus*. Leningrad Seminar on Mathematical Logic (1966).