SECOND PRICE AUCTIONS

A Case Study of Secure Distributed Computing

Bart De Decker[1], Gregory Neven[2], Frank Piessens[3], Erik Van Hoeymissen[1]

[1]*K. U. Leuven, Dept. Computer Science, Celestijnenlaan 200A, B-3001 Leuven, Belgium*

{Bart.DeDecker.Erik.VanHoeymissen}@cs.kuleuven.ac.be

[2]*Research Assistant of the Fundfor Scientific Research, Flanders, Belgium (F.W.O)*

Gregory.Neven@cs.kuleuven.ac.be

Postdoctoral Fellow of the Belgian National Fundfor Scientific Research (F.W.O.)

Frank.Piessens@cs.kuleuven.ac.be

Abstract Secure distributed computing addresses the problem of performing a computation with a number of mutually distrustful participants, in such a way that each of the participants has only limited access to the information needed for doing the computation. Over the past two decades, a number of solutions *requiring no trusted third party* have been developed using cryptographic techniques. The disadvantage of these cryptographic solutions is the excessive communication overhead they incur.

In this paper, we use one of the SDC protocols for one particular application: second price auctions, in which the highest bidder acquires the item for sale at the price of the second highest bidder. The protocol assures that only the name of the highest bidder and the amount of the second highest bid are revealed. All other information is kept secret (the amount of the highest bid, the name of the second highest bidder, ...). Although second price auctions may not seem very important, small variations on this theme are used by many public institutions: e.g., a call for tenders, where contract is given to the lowest offer (or the second lowest).

The case study serves two purposes: we show that SDC protocols can be used for these kind of applications, and secondly, we assess the network overhead and how well these applications scale. To overcome the communication overhead, we use mobile agents and semi-trusted hosts.

Keywords: Secure distributed computing, SDC, mobile agents, second price auction, agents, semi-trusted execution platform

1. INTRODUCTION

Secure distributed computing (SDC) addresses the problem of distributed computing where some of the algorithms and data that are used in the computa-

tion must remain private. Usually, the problem is stated as follows, emphasizing privacy of data. Let f be a publicly known function taking n inputs, and suppose there are n parties (named $P_i, i = 1 \ldots n$), each holding one private input x_i. The n parties want to compute the value $f(x_1, \ldots, x_n)$ without leaking any information about their private inputs (except of course the information about x_i that is implicitly present in the function result) to the other parties. An example is voting: the function f is addition, and the private inputs represent yes ($x_i = 1$) or no ($x_i = 0$) votes. In case an algorithm is to be kept private, instead of just data, one can make f an interpreter for some (simple) programming language, and let one of the x_i be an encoding of a program.

In descriptions of solutions to the secure distributed computing problem, the function f is usually encoded as a boolean circuit, and therefore secure distributed computing is also often referred to as *secure circuit evaluation.*

It is easy to see that an efficient solution to the secure distributed computing problem would be an enabling technology for a large number of interesting distributed applications across the Internet. Some example applications are: auctions ([8]), charging for the use of algorithms on the basis of a usage count ([9, 10]), querying a secret database ([6]), various kinds of weighted voting, protecting mobile code integrity and privacy ([10, 5]), …

Secure distributed computing is trivial in the presence of a globally trusted third party(TTP): all participants send their data and code to the TTP (over a secure channel), the TTP performs the computation and broadcasts the results. The main drawback of this approach is the large amount of trust needed in the TTP.

Solutions without a TTP are also possible. Over the past two decades, a fairly large variety of solutions to the problem has been proposed. An overview is given by Franklin [3] and more recently by Cramer [2] and Neven [7]. These solutions differ from each other in the cryptographic primitives that are used, and in the class of computations that can be performed (some of the solutions only allow for specific kinds of functions to be computed). The main drawback, however, of these solutions is the heavy communication overhead that they incur.

In this paper, we investigate a case study: *second price auctions.* Here, the highest bidder wins but has to pay the second highest bid. The final outcome will only reveal the name of the winner and the amount of the second highest bid. All other bids and even the name of the second highest bidder remain secret. We have chosen this application, because it illustrates the merits of SDC, and is somewhat exemplary for many other useful applications. For instance, the authority and many public institutions request for quotations before awarding the job/purchase to the lowest or second lowest offer. The reader can easily verify that determining the lowest (or second lowest) offer, without revealing the other quotations, is only a small variation on our case study.

In this case study, we try to be as specific as possible. We will show how SDC can be used in this application. Moreover, we will look at the performance. In particular, we examine the communication overhead and the scalability of the application in terms of number of participants. Although the communication overhead seems prohibitively high, a reasonable remedy is proposed, using mobile agents and semi-trusted sites. Indeed, mobile agents employing SDC protocols can provide for a trade-off between communication overhead and trust. The communication overhead is alleviated if the communicating parties are brought close enough together. In our approach, every participant sends its representative agent to a trusted execution site. The agent contains a copy of the private data x_i and is capable of running an SDC-protocol. Different participants may send their agents to different sites, as long as these sites are located closely to each other. Of course, a mobile agent needs to trust his execution platform, but we will show that the trust requirements in this case are much lower than for a classical TTP. Also, in contrast with protocols that use unconditionally TTPs, the trusted site is not involved directly. It simply offers a secure execution platform: i.e. it executes the mobile code correctly, does not spy on it and does not leak information to other mobile agents. Moreover, the trusted host does not have to know the protocol used between the agents. In other words, the combination of mobile agent technology and secure distributed computing protocols makes it possible to use a *generic* TTP that, by offering a secure execution platform, can act as TTP for a wide variety of protocols in a uniform way. A detailed discussion of the use of mobile agent technology for advanced cryptographic protocols is given in [7].

The sequel of the paper is organized as follows: in section 2, we review one of the SDC protocols that will be used by the application; a design of the the application, second price auctions, is given in section 3; in this section, we also examine the communication overhead and tackle the scalability issue. In section 4, we introduce a modus operandi for the application. Finally, in section 5, we summarize the main outcomes of this paper.

2. SECURE DISTRIBUTED COMPUTING USING GROUP-ORIENTED CRYPTOGRAPHY

In [4], Franklin and Haber propose a protocol that evaluates **a boolean circuit** on data encrypted with a homomorphic probabilistic encryption scheme for any number of participants. It resembles the protocol for two parties, proposed by Abadi and Feigenbaum ([1]).

To extend the idea of [1] to the multi-party case, an encryption scheme is needed that allows anyone to encrypt, but needs the cooperation of all participants to perform a decryption. In a joint encryption scheme, all participants know the public key K_{pub} while each participant $P_1, \ldots, P_n$ has his own pri-

vate key $K_1, \ldots, K_n$. Using the public key, anyone can create an encryption $E_S(m)$ of some message m, where $S \subseteq \{P_1, \ldots, P_n\}$, such that the private key of each participant in S is needed to decrypt. More formally, if D_i denotes the decryption with P_i's private key, the relation between encryption and decryption is given by

$$D_i(E_S(m)) = E_{S \setminus \{P_i\}}(m)$$

The plaintext m should be easily recoverable from $E_\emptyset(m)$.

In the joint encryption scheme used by Franklin and Haber, a bit b is encrypted as

$$E_S(b) = \left[g^r \mod N,\ (-1)^b \left(\prod_{j \in S} g^{K_j} \right)^r \mod N \right]$$

where $N = pq$, p and q are two primes such that $p \equiv q \mod 4$, and $r \in_R Z_N$. The public key is given by $[N, g, g^{K_1} \mod N, \ldots, g^{K_n} \mod N]$ where each K_i represents the private (secret) key of participant P_i.

This scheme has some additional properties that are used in the protocol:

- *XOR-Homomorphic.* Anyone can compute a joint encryption of the XOR of two jointly encrypted bits. Indeed, if $E_S(b) = [\alpha, \beta]$ and $E_S(b') = [\alpha', \beta']$, then $E_S(b \oplus b') = [\alpha\alpha' \mod N, \beta\beta' \mod N]$.
- *Blindable.* Given an encrypted bit, anyone can create a random ciphertext that decrypts to the same bit. Indeed, if $E_S(b) = [\alpha, \beta]$ and $r \in_R Z_N$, then

  $$\left[\alpha g^r \mod N,\ \beta \left(\prod_{j \in S} g^{K_j} \right)^r \mod N \right]$$

  is a joint encryption of the same bit.
- *Witnessable.* Any participant can withdraw from a joint encryption by providing the other participants with a single value. Indeed, if $E_S(b) = [\alpha, \beta]$, it is easy to compute $D_i(E_S(b))$ from

  $$W_i([\alpha, \beta]) = \alpha^{-K_i} \mod N$$

First of all, the participants must agree on a value for N and g, choose a secret key K_i and broadcast $g^{K_i} \mod N$ to form the public key. To start the actual protocol, each participant broadcasts a joint encryption of his input bits. For an XOR-gate, everyone simply applies the XOR-homomorphism. The encrypted output of a NOT-gate can be found by applying the XOR-homomorphism with a default encryption of a one, e.g. $[1, -1]$.

However, it is the AND-gate that causes some trouble. Suppose the encrypted input bits for the AND-gate are $\hat{u} = E(u)$ and $\hat{v} = E(v)$. To compute a joint encryption $\hat{w} = E(w) = E(u \wedge v)$, they proceed as follows:

1 Each participant P_i chooses random bits b_i and c_i and broadcasts $\hat{b}_i = E(b_i)$ and $\hat{c}_i = E(c_i)$.

2 Each participant repeatedly applies the XOR-homomorphism to calculate $\hat{u}' = E(u') = E(u \oplus b_1 \oplus \ldots \oplus b_n)$ and $\hat{v}' = E(v') = E(v \oplus c_1 \oplus \ldots \oplus c_n)$. Each participant broadcasts decryption witnesses $W_i(\hat{u}')$ and $W_i(\hat{v}')$.

3 Everyone can now decrypt $\hat{u}'$ and $\hat{v}'$. We have the following relation between $w' = u' \wedge v'$ and $w = u \wedge v$:

$$
\begin{aligned}
w' &= u' \wedge v' \\
&= (u \oplus b_1 \oplus \cdots \oplus b_n) \wedge (v \oplus c_1 \oplus \cdots \oplus c_n) \\
&= (u \wedge v) \;\oplus\; (u \wedge c_1) \;\oplus \cdots \oplus\; (u \wedge c_n) \\
&\qquad \oplus\; (b_1 \wedge c_1) \;\oplus \cdots \oplus\; (b_n \wedge c_1) \\
&\qquad \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \\
&\qquad \oplus\; (b_1 \wedge c_n) \;\oplus \cdots \oplus\; (b_n \wedge c_n) \\
&\qquad \oplus\; \underbrace{(b_1 \wedge v)}_{w_1} \;\oplus \cdots \oplus\; \underbrace{(b_n \wedge v)}_{w_n} \\
&= (u \wedge v) \oplus w_1 \oplus \cdots \oplus w_n
\end{aligned}
$$

Each participant is able to compute a joint encryption of w_i: he knows b_i and c_i (he chose them himself) and he received encryptions $\hat{c}_j$ from the other participants, so he can compute $E(b_i \wedge c_j)$ as follows:

- If $b_i = 0$, then $b_i \wedge c_j = 0$, so any default encryption for a zero will do, e.g. [1,1].
- If $b_i = 1$, then $b_i \wedge c_j = c_j$, so $\hat{c}_j$ is a valid substitution for $E(b_i \wedge c_j)$.

$E(u \wedge c_i)$ and $E(v \wedge b_i)$ can be computed in an analogous way. He uses the XOR-homomorphism to combine all these terms, blinds the result and broadcasts this as $\hat{w}_i$.

4 Each participant combines $\hat{w}'$ and $\hat{w}_j$ $(j = 1 \ldots n)$, again using the XOR-homomorphism, to form $\hat{w} = E(w)$.

When all gates in the circuit have been evaluated, every participant has a joint encryption of the output bits. Finally, all participants broadcast decryption witnesses to reveal the output.

3. SECOND PRICE AUCTIONS

In this section we consider second price auctions, where there is one item for sale and there are n bidders. The item will only be sold if the bid of one participant is strictly higher than the other bids. In all other cases there is no

winner. The clearing price is the second highest bid. The requirements for this type of auction are the following:

- if there is no winner, nothing is revealed;
- if there is a winner:
  - the identity of the highest bidder is revealed, but the highest bid remains secret;
  - the 2nd highest bid is revealed, but the identity of the 2nd highest bidder is kept secret;
  - no other information (other bids) are to be revealed.

For three participants X, Y and Z, the boolean circuit is shown in Figure 1. The inputs to the circuit are 32-bit bids[1]. The output is the identity of the winner, represented by the bits $R1$ and $R0$ ($R1R0 = 00$ no winner, 01 winner is X, 10 winner is Y, 11 winner is Z), and the clearing price. If there is no winner, the clearing price is set to zero. To determine the winner, the circuit uses three comparators and a number of AND and OR gates. To determine the clearing price, four multiplexers are used. Consider the situation where X makes the highest bid. In this case $G1 \wedge G2 = 1$, $L1 \wedge G3 = 0$, $L2 \wedge L3 = 0$ and $R1R0 = 10$, so the second input to the final multiplexer will be chosen. The input on this line is determined by the bids made by Y and Z. If $Y > Z$ then $G3 = 1$ and Y will be selected as the clearing price. In the other cases ($Y < Z$ or $Y = Z$) Z will be the clearing price.

Our goal is to estimate the communication overhead of an implementation of secure distributed second price auctions with the protocol proposed by Franklin and Haber. The auction is designed as a boolean circuit and the communication overhead for secure circuit evaluation is estimated. The communication overhead is determined by the following steps in the protocol:

- broadcast of the encrypted input bits of each participant;
- evaluation of an AND gate:
  - broadcast of the encrypted bits $E(b_i)$, $E(c_i)$;
  - broadcast of the decryption witnesses $W_i(\hat{u}')$, $W_i(\hat{v}')$;
  - broadcast of the blinded $\hat{w}_i$;
- broadcast of the output decryption witnesses.

The associated communication overhead is:

[1] In reality, fewer bits (e.g. 8 or 16) would suffice.

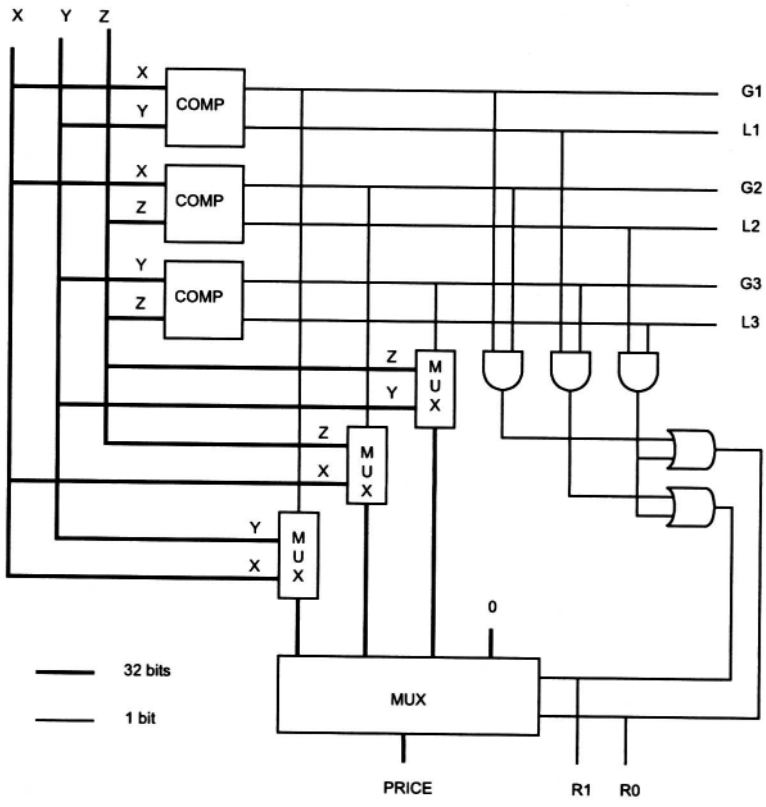

Figure 1. Boolean circuit implementation of second price auctions.

- $2 \cdot |N| \cdot in_i \cdot n$ for the broadcast of the input bits;
- $8 \cdot |N| \cdot n$ for the evaluation of an AND gate;
- $|N| \cdot out \cdot n$ for the decryption broadcast.

where $|N|$ is the length of N in bits, which is the same as the number of bits needed to represent an element of Z_N^*, in_i is the number of input bits of participant i, n is the number of participants and *out* is the number of output bits of the circuit. In order to estimate the communication overhead, we need to be able to determine the number of AND gates in the boolean circuit (note that each OR gate can be implemented with AND and NOT gates). Each comparator can be built with 374 AND-gates[2]

[2]The booleanfunction $A > B$, can be expressed as $A_0.\overline{B_0} + (A_0.B_0 + \overline{A_0}.\overline{B_0}).(A_1.\overline{B_1} + (A_1.B_1 + \overline{A_1}.\overline{B_1}).(A_2.\overline{B_2}...))$. Hence if A and B are k-bit numbers, $1 + 6(k-1)$ AND gates are needed. Both functions, $A > B$ and $B > A$, are needed for each comparator.

For $n > 3$ participants, the circuit changes as follows. The number of comparators needed is now $\binom{n}{2} = n \cdot (n-1)/2$. The final multiplexer will need to distinguish between $n+1$ different cases, i.e. n possible winners or no winner at all. The other n multiplexers are there to select the clearing price out of $n-1$ bids when there is a winner. The number of AND gates needed for each multiplexer as a function of the number of inputs m is shown in Figure 2. Besides the comparators and the multiplexers, some additional AND and OR gates are needed. However, the number of these gates is negligible compared to the number of gates needed for the comparators and multiplexers. In summary, the circuit has a total gate complexity of $O(n^2)$.

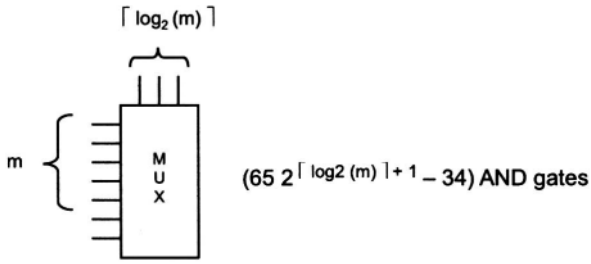

Figure 2. Number of AND gates needed in a mulitplexer

The results of estimating the communication overhead for this circuit as a function of the number of participants n are summarized in Table 1[3]. Franklin and Haber's protocol is linear in the number of broadcasts, so the total message complexity is $O(n^3)$. However, it must be noted that this only holds on a network with broadcast or multicast functionality, such that the communication overhead of sending a message to all participants is the same as that of sending a message to a single participant. In absence of such infrastructure, the total message complexity is $O(n^4)$.

Table 1. Network overhead of secure second price auctions

n	4	16	32
Overhead (MB)	15	1000	8000

[3]We choose $|N|$ to be 1024 bits.

4. MODUS OPERANDI

From the previous section, it should be clear that the design of the application has pros and cons:

- A major advantage is that our solution does not require a globally trusted third party that plays the role of the arbiter.
- The worst drawback is the immense communication overhead and the fact that the solution does not scale very well.

There exists a trade-off between 'trust' and 'communication overhead' in both options, the first one using a TTP and the solution that uses SDC. In this section, we investigate this trade-off and present a nice remedy for the communication overhead.

If a globally trusted third party is used, every participant, P_i, has to send its private bid x_i to that TTP who will select the highest bidder, determine the second highest bid, and disseminate its decision to the participants $P_i, i = 1..n$ (see Figure 3). Of course, before sending its private data to the TTP, every P_i

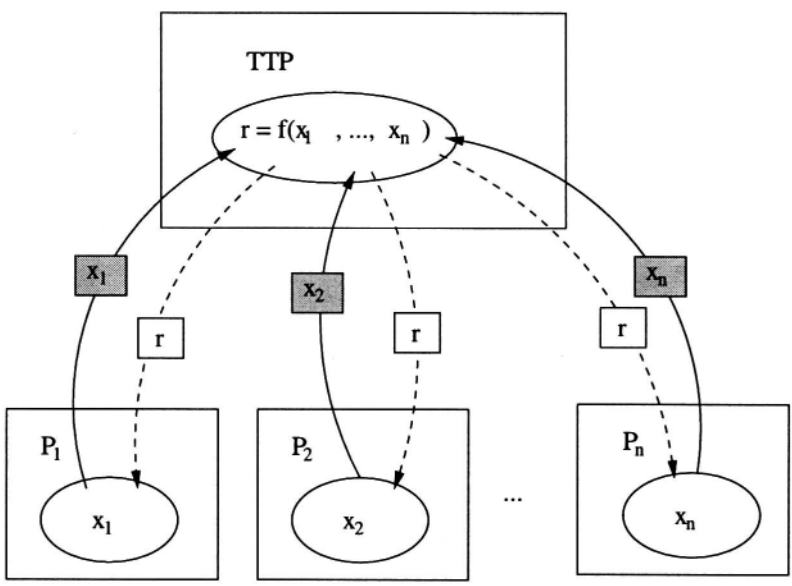

Figure 3. 2nd Price Auction Using a TTP.

must first authenticate the TTP, and then send x_i through a safe channel. This can be accomplished via conventional cryptographic techniques. It is clear that this approach has a very low communication overhead: the data is only sent once to the TTP; later, every participant receives the result of the computation. However, every participant should uncondionally trust the TTP. It is not clear

whether n distrustful participants will easily agree on one single trustworthy site. If this site is compromised, all secrets, x_i may be compromised! Also, the site needs the appropriate software for this particular application. Hence, for every new 'application' new software needs to be installed. Therefore, the participants not only need to trust the (security of) the site, but also the software for this application.

In our approach (see Figure 4), the trust requirements are really minimal: every participant P_i trusts its own execution site E_i, and expects that the other participants provide correct values for their own inputs. (Note that in this protocol, a participant cannot cheat, because of the use of witnesses.) Although our approach is very attractive, it suffers extensive communication overhead and does not scale well.

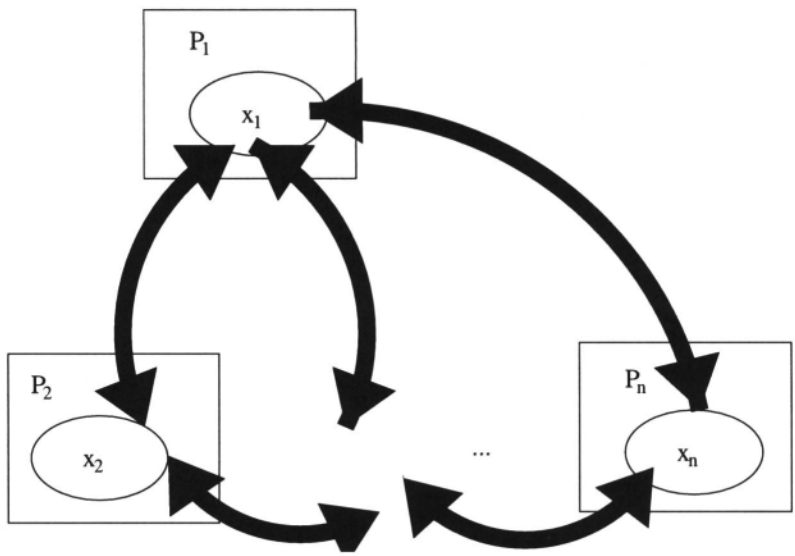

Figure 4. 2nd Price Auction Using SDC.

The communication overhead of SDC-techniques can be remedied by introducing semi-trusted execution sites and mobile agents (see Figure 5). Every participant P_i sends its representative, agent A_i, to a trusted execution site E_j. The agent contains a copy of the private data x_i and is capable of running a SDC-protocol. It is allowed that different participants send their agents to different sites. The only restriction being that the sites should be located closely to each other, i.e. should have high bandwidth communication between them. Of course, every execution site needs a mechanism to safely download an agent. However, that can be easily accomplished through convential cryptographic techniques. The amount of large distance communication is moderate: every participant sends its agent to a remote site, and receives the result from its

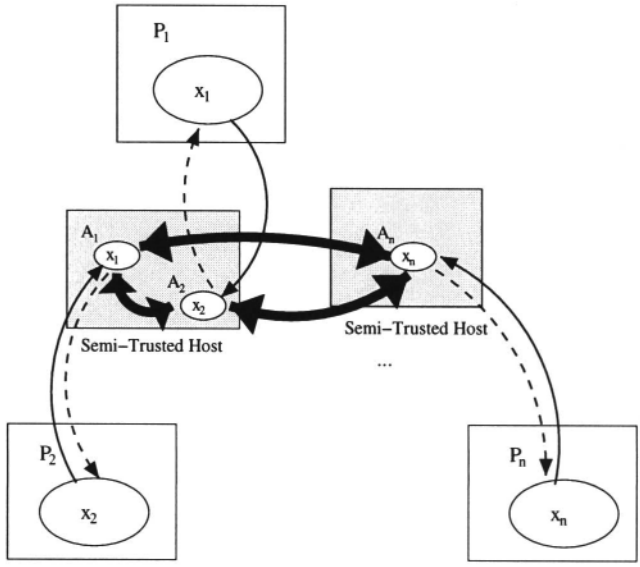

Figure 5. 2nd Price Auctions Using Agents (SDC) and Semi-Trusted Sites.

agent. The agents use a SDC-protocol, which unfortunately involves a high communication overhead. However, since the agents are executing on sites that are near each other, the overhead of the SDC-protocol is acceptable. No high bandwidth communication between the participants is necessary, and there is no longer a need for one single trusted execution site. The agents that participate in the secure computation are protected against malicious behaviour of other (non-trusted) execution sites by the SDC-protocols. That is sufficient to make this approach work. Moreover, in contrast with the approach where one uses an unconditionally trusted third party, the trusted sites are not involved directly. They simply offer a *secure execution platform*: the trusted hosts do *not* have to know the protocol used between the agents. In other words, the combination of mobile agent technology and secure distributed computing protocols makes it possible to use *generic* trusted third parties that, by offering a secure execution platform, can act as trusted third party for a wide variety of protocols in a uniform way. Finally, the question remains whether it is realistic to assume that participants can find execution sites that are close enough to each other. Given the fact however that these execution sites can be *generic*, we believe that providing such execution sites could be a commercial occupation. Various deployment strategies are possible. Several service providers, each administering a set of geographically dispersed "secure hosts", can propose their subscribers an appropriate site for the secure computation. The site is chosen to be in the neighborhood of a secure site of the other service providers involved. Another

approach is to have execution parks, offering high bandwidth communication facilities, were companies can install their proprietary "secure site". The park itself could be managed by a commercial or government agency.

5. CONCLUSIONS

This paper demonstrates that second price auctions, and many other relevant applications, can be implemented by using SDC protocols. That way, the participants can make sure that all confidential information is kept secret. The major disadvantage, the overwhelming communication overhead, can be remedied through the use of mobile agents and semi-trusted sites. There is no need for one generally trusted site, nor does the program code have to be endorsed by all participants. The trusted execution sites are generic and can be small (which might allow to draft a formal security for these sites). The communication overhead of secure distributed computing protocols is no longer prohibitive for their use since the execution sites are located closely to each other.

References

[1] M. Abadi and J. Feigenbaum, "Secure circuit evaluation, a protocol based on hiding information from an oracle," Journal of Cryptology, 2(1), p. 1–12, 1990

[2] R. Cramer. "An introduction to secure computation", in LNCS 1561, pp 16–62, 1999.

[3] M. Franklin, "Complexity and security of distributed protocols," Ph. D. thesis, Computer Science Department of Columbia University, New York, 1993

[4] M. Franklin and S. Haber, "Joint encryption and message-efficient secure computation," Journal of Cryptology, 9(4), p. 217–232, Autumn 1996

[5] S. Loureiro and R. Molva, "Privacy for Mobile Code", Proceedings of the workshop on *Distributed Object Security*, OOPSLA '99, p. 37–42.

[6] G. Neven, F. Piessens, B. De Decker, "On the Practical Feasibility of Secure Distributed Computing: a Case Study", Information Security for Global Information Infrastructures (S. Qing, J. Eloff, ed.), Kluwer Academic Publishers, 2000, pp. 361-370.

[7] G. Neven, E. Van Hoeymissen, B. De Decker, F. Piessens, "Enabling Secure Distributed Computations : Semi-trusted Hosts and Mobile Agents", to appear in Networking and Information Systems Journal 3 (2001).

[8] N. Nisan, "Algorithms for selfish agents", *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, Trier, Germany, March 1999, p. 1–15.

[9] T. Sander and C. Tschudin, "On software protection via function hiding", *Proceedings of the second workshop on Information Hiding*, Portland, Oregon, USA, April 1998.

[10] T. Sander and C. Tschudin, 'Towards mobile cryptography", *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, Oakland, California, May 1998.

[11] T. Sander, A. Young, M. Yung, "Non-Interactive CryptoComputing for NC^1", preprint.