PRIVATE BIDDING FOR MOBILE AGENTS

Bartek Gedrojc, Kathy Cartrysse, Jan C. A. van der Lubbe
Delft University of Technology
Mekelweg 4, 2628 CD, Delft, the Netherlands

Keywords: Untrustworthy Environments, Oblivious Third Party, Φ-Hiding Assumption, ElGamal, Mobile Software Agents, Multi-Party Computation.

Abstract: A major security and privacy threat for Mobile Software Agents are Untrustworthy Environments; which are able to spy on the agents' code and private data. By combining Multi-Party Computation with ElGamal public-key encryption system we are able to create a protocol capable of letting two agents have a private bidding within an Honest-but-Curious environment only with the help of an Oblivious Third Party. The Oblivious party is able to compare two encrypted inputs without being able to retrieve any information about the inputs.

1 INTRODUCTION

In this vast growing digital era we are more often surrounded by intelligent devices which can support us in our daily life. While information flows through these ambient devices, which are not only connected to local networks but also to the outside world, we can expect to encounter security and privacy threats. The most alarming threat is the malicious user who has the ability to create malicious agents, spy on insecure communication channels but also to create untrustworthy environments (Claessens et al., 2003).

Every environment has total control over the information that is executed in his digital "world" and it can therefore manipulate the agent code. One topic that so far received relatively little attention in literature is the execution of mobile software agents within untrustworthy environments. Most research focusses on malicious agents and communications, and assumes the agents are executed in trusted environments.

Mobile software agents may possess private information of their user e.g. credit card numbers, personal preferences, etc. Agents should be able to perform tasks on behalf of their users by using this personal information within untrustworthy environments without compromising security and privacy.

One of the solutions to the untrustworthy environments problem is Execution Privacy, which strives to the correct execution of software agents while keeping the agent code and state private

In section 2 we present our model and we will discuss the relation between previous work. Section 3 describes the building blocks that are necessary to construct the protocol. In section 4 the protocol is presented, while section 5 discusses the security of the protocol. Finally, section 6 ends with concluding remarks.

2 PROBLEM DESCRIPTION

This paper addresses the private bidding problem which was introduced by (Cachin, 1999): Alice wants to buy some goods from Bob if the price is less than a and Bob would like to sell, but only for more than b and neither of them wants to reveal the secret bounds. Our objective is to demonstrate that private bidding is also feasible with Mobile Software Agents while this bidding takes places in an untrustworthy environment. To achieve this goal we are using an Oblivious Third Party to compare the bids of Alice and Bob without learning any information about a or b.

2.1 Model

Our private bidding for mobile agent model is almost similar to the original private bidding model as was described by (Cachin, 1999). Only, in our case the

Gedrojc B., Cartrysse K. and C. A. van der Lubbe J. (2006).
PRIVATE BIDDING FOR MOBILE AGENTS.
In *Proceedings of the International Conference on Security and Cryptography*, pages 277-282
DOI: 10.5220/0002102702770282

communication between the Agents (bidders) is done in an untrustworthy environment i.e. public environment instead of a private environment. Therefore, we must make sure that the communication between the agents and the final decision on who has the highest bid remains secure and private, even if this takes place in an untrustworthy environment, see Figure 1.

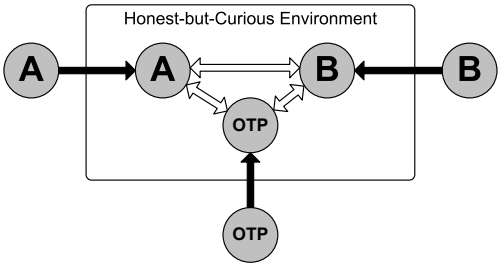

Figure 1: Private Bidding Scenario.

Alice (A), Bob (B) and the Oblivious Third Party (OTP) are the three parties involved in the model. Also, Alice and Bob are mobile software agents who share some random secret with each other. The OTP is a trusted party that will correctly execute the inputs a, b given by Alice or Bob, without being able to learn any information about the inputs. The goal of Alice and Bob is to determine if $a > b$. In an e-commerce setting this means that the maximum price the buyer (Alice) is willing to pay is greater than the minimum price the seller (Bob) is willing to receive. Besides this, they will only reveal their actual bids a, b if it satisfies the $a > b$ condition, otherwise they will keep their secrets private.

Our agent bidding model is situated in an untrustworthy environment. Due to the hardness of the problem we assume that this environment will execute the agents and the code correctly, but it is interested in what it is processing. This implies that the agents are not able to use their private keys or any other secret data. Therefore, we will use the OTP to perform the comparison of the inputs without revealing it to the untrustworthy environment.

The Untrustworthy Environment is also called an honest-but-curious adversary which was introduced by (Rabin, 1981). Consequently, the environment is a server executing the mobile code i.e. the mobile agent, while storing all information from the agents and all communication between the agents. The server is not a Malicious Environment trying to modify the mobile code. Our goal is to ensure the server does not learn the private inputs a, b before the agents decide privately to reveal their secret inputs e.g. when the bidding satisfies $a > b$.

2.2 Contribution

We present a Private Bidding protocol for Mobile Software agents while these agents are situated in an Honest-but-Curious environment only by using an Oblivious Third Party. Our solution mixes the Private Agent Communication algorithm from (Cartrysse, 2005; Cartrysse and van der Lubbe, 2004) with the Private Bidding protocol from (Cachin, 1999).

With our contribution we demonstrate that it is possible to have a secure, efficient and fair private bidding protocol for mobile software agents. Applications can be found in e-commerce scenarios where the bidding takes place on a public server and where the user has no communication with his mobile software agents.

2.3 Related Work

Private bidding originates from the Millionaires' problem initially described by (Yao, 1982). Further research proved that any multi-party function can be computed securely even without the help from a third party (Goldreich, 2002). But, according to (Algesheimer et al., 2001) secure mobile agents schemes do not exist when some environment is to learn information that depends on the agent's current state. Therefore, in order to merge mobile agents with multi-party computation techniques we needed the help of an Oblivious Third Party.

The concept of combining mobile agents and making decisions in the encrypted domain originates from (Sander and Tschudin, 1998a; Sander and Tschudin, 1998b). They proposed a software only non-interactive computing with encrypted functions which was based on homomorphic encryption schemes. We elaborate on their concept by using (ElGamal, 1984) as the basis of our agent communication algorithm. We also manifest that the ElGamal encryption in our model can be used with the homomorphic in addition property without loosing security.

The Non-Interactive CryptoComputing for NC^1 by (Sander et al., 1999) also discusses computing with encrypted functions w.r.t. mobile agent systems. Their solution is capable of comparing two private inputs but it does not provide fairness because the inputs must be encrypted by one of the agents' public keys. Consequently, the results will only be available to one of agents. Also, the private key to decrypt the result of the comparison cannot be used in the untrustworthy environment and therefore the agent must leave this environment to learn the result of the computation.

3 BUILDING BLOCKS

Before discussing the private bidding protocol in paragraph 4 we will elaborate more on the cryptographic techniques which we have used.

3.1 Φ-Hiding Assumption

Our model is based on the Φ-Hiding Assumption (Φ-HA) published by (Cachin et al., 1999), which states the difficulty of deciding whether a small prime divides $\phi(m)$, where m is a composite integer of unknown factorization. We use the Φ-HA as follows: Choose randomly m that hides a prime p_0 so it is hard to distinguish if p_0 or p_1 is a factor of $\phi(m)$, where p_1 is randomly and independent chosen i.e. it is computationally indistinguishable.

Let us choose $m = p'q'$ where p' is a safe prime (prime p is a safe prime if $\frac{p-1}{2}$ is also a prime) and q' is a quasi-safe prime. We compute $p' = 2q_1 + 1$ and $q' = 2pq_1 + 1$, where q_1 is a prime and p, q_2 are odd primes.

The Euler totient function $\phi(m)$ is defined as the number of positive integers $\leq m$ which are relative prime to m. This results in the Euler totient theorem

$$g^{\phi(m)} \equiv 1 \bmod m \tag{1}$$

where $g \in \mathbb{Z}$. Next, we compute the Euler totient function:

$$\begin{aligned} \phi(m) &= \phi(p'q') = (p'-1)(q'-1) \\ &= (2q_1 + 1 - 1)(2pq_1 + 1 - 1) \\ &= (2q_1)(2pq_2) = 4pq_1q_2 \end{aligned} \tag{2}$$

Combining (1) and (2) we can compute:

$$g^{4pq_1q_2} \equiv 1 \bmod m \tag{3}$$

Given a set of primes $\mathcal{P} = \{p_1, \ldots, p_n\}$ where $n \geq 0$ we can compute $\prod_{i=0}^{n} p_i$. Also, by computing m that hides a prime p we can use the Φ-HA to determine if $\mathcal{P}$ contains a *p-th root* module m using the factorization of m. To perform this check one computes

$$g^{\frac{4pq_1q_2}{p}\prod_{i=0}^{n} p_i} \equiv 1 \bmod m \tag{4}$$

if it is congruent to 1 modulo m then the set $\mathcal{P}$ hides p else is does not. Note that m hides p if and only if $p|\phi(m)$. It is important that the $g \in \mathbb{Z}$ should not be a *p-th root* module m i.e. there exists no μ such that $\mu^p \equiv g \bmod m$. Therefore, we should choose $g \in QR_m$ because p should not be an even prime.

3.2 Homomorphic E-E-D

In (Cartrysse and van der Lubbe, 2004) an algorithm is described that is able to encrypt a message m with the key of Alice and send it to Bob who encrypts the message again with his own key. Resulting in the message to be send back to Alice, who will be able to decrypt the message and be left with the message encrypted by Bob. This algorithm, as visualized in Figure 2 is know as E-E-D (Encryption-Encryption-Decryption).

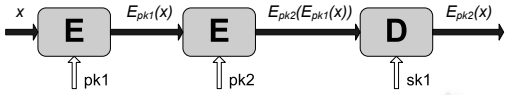

Figure 2: E-E-D model.

Our private bidding model uses this E-E-D property, as well as the homomorphic addition based on ELGamal encryption (ElGamal, 1984). The idea behind the model is to encrypt two messages with different keys and add them together. Next, the messages are decrypted with the first key and the result will be the addition of the two messages which are only encrypted with the second key, see Figure 3.

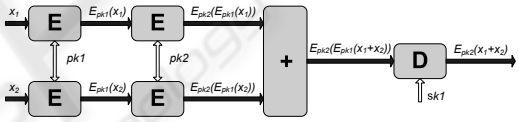

Figure 3: Homomorphic E-E-D in Addition model.

We generate a large prime p and a generator $g \in \mathbb{Z}_p^*$. We choose a random integer $a_1, 1 \leq a_1 \leq p-2$ and compute $\omega = g^{a_1} \bmod p$. We also need to select a random integer $k_1, 1 \leq k_1 \leq p-2$ and compute the following

$$\gamma_1 = g^{k_1} \bmod p$$
$$\delta_{1,x_1} = x_1\omega^{k_1} \bmod p$$

The public key $(pk1)$ is (p, g, ω_1), the message is x_1, the cipher text is $c_{1,x_1} = (\gamma_1, \delta_{1,x_1})$ and the secret key $(sk1)$ is a_1. Next we can encrypt δ_{1,x_1}, using a new public key ω_2 but with the same generator g and prime p. We must also compute $\omega_2 = g^{a_2} \bmod p$ and use this to encrypt the message using the following

$$\gamma_2 = g^{k_2} \bmod p$$
$$\delta_{2,x_1} = \delta_{1,x_1}\omega_2^{k_2} \bmod p = x_1\omega_1^{k_1}\omega_2^{k_2} \bmod p$$

The public key $(pk2)$ is (p, g, ω_2), the cipher text is $c_{2,x_1} = (\gamma_1, \delta_{2,x_1})$ and the secret key $(sk2)$ is a_2. For convenience we can write the above steps as $E_{pk2}(E_{pk1}(x_1))$ which represent the top left two function blocks from Figure 3.

We also want to encrypt a message x_2 with the same keys and the same random integers k_1, k_2 as

message x_1, which can be calculated as above. We can write this result as $E_{pk2}(E_{pk1}(x_2))$ which represents the two bottom left function blocks from Figure 3. Next we can add the two encrypted messages δ_{2,x_1}, δ_{2,x_2} using the homomorphic property of ElGamal as follows

$$\begin{aligned}\delta_{2,x_1} + \delta_{2,x_2} &= \delta_{1,x_1}\omega_2^{k_2} \bmod p + \delta_{1,x_2}\omega_2^{k_2} \bmod p \\ &= x_1\omega_1^{k_1}\omega_2^{k_2} \bmod p + x_2\omega_1^{k_1}\omega_2^{k_2} \bmod p \\ &= (x_1 + x_2)\omega_1^{k_1}\omega_2^{k_2} \bmod p \\ &= \delta_{2,x_1+x_2} = x\end{aligned}$$

The two encrypted messages δ_{2,x_1}, δ_{2,x_2} can only be added to one another when they are both encrypted with the same public keys and with the same k_1, k_2. The result is $E_{pk2}(E_{pk1}(x_1 + x_2))$ and can be decrypted using the first private key a_1 and the following formula

$$\begin{aligned}x' = D_{sk1}(x) &= \frac{\delta_{2,x_1+x_2}}{\gamma_1^{a_1}} \bmod p \\ &= \frac{(x_1 + x_2)\omega_1^{k_1}\omega_2^{k_2}}{g^{k_1 a_1}} \bmod p \\ &= (x_1 + x_2)\omega_2^{k_2} \bmod p\end{aligned}$$

We now have the first and the second message added to one another and encrypted with the second public key $pk2$, which can be written as

$$D_{sk1}(E_{pk2}(E_{pk1}(x_1 + x_2))) = E_{pk2}(x_1 + x_2) = x'$$

Consequently, decrypting x' produces the following

$$\begin{aligned}D_{sk1}(x') &= \frac{x'}{\gamma_2^{a_2}} \bmod p \\ &= \frac{(x_1 + x_2)\omega_2^{k_2}}{g^{k_2 a_2}} \bmod p \\ &= x_1 + x_2\end{aligned}$$

which is the addition of the two messages without encryption (plaintext).

It has to be noticed that according to the Diffie-Hellman problem it is impossible to compute $g^{ak} \bmod p$ from $g^a \bmod p$ and $g^k \bmod p$ (Diffie and Hellman, 1976). This means that the key a and the random integer k must remain secure, and preferably be changed frequently. We have chosen to make a double encryption by one user using an equal k, because otherwise we cannot use the homomorphic property in addition of the ElGamal encryption. A security evaluation will follow in paragraph 5 concerning the use of this identical k.

4 PROTOCOL

The protocol starts by letting the Oblivious Third Party (OTP) generate an ElGamal key-pair with the public-key E_{pkt} and the secret-key D_{skt}.

The buyer Alice does not want to spend more than her maximum amount a, which can be represented as a binary string. She also chooses random and independent values $x_l, x_{l-1}, \ldots, x_0 \in \mathbb{Z}$, $r_{l-1}, \ldots, r_0 \in \mathbb{Z}$ and $s_{l-1}, \ldots, s_0 \in \mathbb{Z}$, where l is the amount of bits needed for a and b, and where b is the minimum price of Bob the seller.

Alice needs to define an n-bit prime $p_a = \lambda(t_a)$ where $t_a \in \{0,1\}^n$ is randomly chosen. Using $\lambda(x)$ she can map a string x to an n-bit prime p e.g. by finding the smallest prime greater than x. Then Alice generates a random m_A using the Φ-Hiding Assumption, as described above, which hides her prime p_a. She also computes $G_a = \frac{\phi(m_a)}{p_a}$ which is kept secret from the OTP.

Next, a key κ for the hash function H_κ must be generated and published. Then Alice needs to calculate $\varphi_{a,j}$ using the hash function $H_\kappa(x_j + s_j)$ and compute the following for $j = l-1, \ldots, 0$

$$\varphi_{a,j} = H_\kappa(x_j + s_j) \oplus t_a$$

Alice also needs to encrypt the input string a_j using her public-key $E_{pka,j}$ and random integers $k_{a,j}$ for $j = l-1, \ldots, 0$. This has to be *"sealed"* using the public-key of the OTP and the random integers $k_{t,j}$

$$y_{a,j} = \begin{cases} E_{pkt,j}(E_{pka,j}(x_j - x_{j+1} + r_j)) & \text{if } a_j = 0 \\ E_{pkt,j}(E_{pka,j}(x_j - x_{j+1} + s_j + r_j)) & \text{if } a_j = 1 \end{cases}$$

The bidding is prepared for Bob by Alice using the same cryptosystem with random integers $k_{t,j}$, $k_{a,j}$ and public-keys $E_{pkt,j}$

$$y_{b,j} = \begin{cases} y_{b0,j} = E_{pkt,j}(E_{pka,j}(-r_j)) & \text{if } b_j = 0 \\ y_{b1,j} = E_{pkt,j}(E_{pka,j}(-s_j - r_j)) & \text{if } b_j = 1 \end{cases}$$

She also computes

$$\Psi_{b,j} = H_\kappa(x_j - s_j)$$

and sends it to Bob.

Bob is able to acquire the public $y_{b0,j}$, $y_{b1,j}$ and compute according his own input string b_j, $y_{b,j}$. Because $y_{a,j}$ is also publicly available he can compute $y_{ab,j}$ using

$$y_{ab,j} = y_{a,j} + y_{b,j} \tag{5}$$

To be sure that the Untrustworthy Environment will not be able to construct b from y_{ab} Bob has to encrypt (5) again, by computing

$$w_{ab,j} = E_{pkt,j}(y_{ab,j})$$

The only requirement is that Bob uses the same cryptosystem as Alice is using, in other words he has to use the same generator g and prime p but he can use a different k.

Bob has already chosen a $t_b \in \{0,1\}^n$, defined an n-bit prime $p_b = \lambda(t_b)$ and generated a random

m_b. He also computes $G_b = \frac{\phi(m_b)}{p_b}$ and keeps this secret from the OTP. Bob receives from Alice $\Psi_{a,j}$ and computes before entering the environment for $j = l-1, \ldots, 0$

$$\varphi_{b,j} = \Psi_{a,j} \oplus t_b$$

Alice and Bob have prepared their bidding using the public information and are now able to move into the Untrustworthy Environment. Then Bob sends $\varphi_{b,j}$, m_b and $w_{ab,j}$ to Alice.

Alice is able to decrypt $w_{ab,j}$ because she is in possession of the secret-key $D_{ska,j}$

$$\begin{aligned} z_j &= D_{ska,j}(w_{ab,j}) \\ &= D_{ska,j}(E_{pkt,j}(E_{pkt,j}(E_{pka,j}(y_{ab,j})))) \\ &= E_{pkt,j}(E_{pkt,j}(y_{ab,j})) \end{aligned}$$

The bidding is prepared and Alice sends

$$\kappa, x_l, z_{l-1}, \ldots, z_0, \varphi_{a,l-1}, \ldots, \varphi_{a,0} \\ \varphi_{b,l-1}, \ldots, \varphi_{b,0}, m_a, m_b$$

to the OTP.

The OTP makes $c_l = x_l$ and chooses $g_{a,l} \in QR_{m_a}$, $g_{b,l} \in QR_{m_b}$ by selecting a random element of $\mathbb{Z}_{m_a}$, respectively $\mathbb{Z}_{m_b}$, and squaring it. For $j = l-1, \ldots, 0$ the following five steps are repeated

$$\begin{aligned} &1. c_j = c_{j+1} + D_{skt,j}(D_{skt,j}(z_j)) \\ &2. q_{a,j} = \lambda(H_\kappa(c_j) \oplus \varphi_{a,j}) \\ &3. g_{a,j} = (g_{a,j+1})^{q_{a,j}} \bmod m_a \\ &4. q_{b,j} = \lambda(H_\kappa(c_j) \oplus \varphi_{b,j}) \\ &5. g_{b,j} = (g_{b,j+1})^{q_{b,j}} \bmod m_b \end{aligned}$$

After this iteration the OTP chooses $r_a \in \mathbb{Z}_{m_a}$, $r_b \in \mathbb{Z}_{m_b}$ randomly and independently, computing

$$\begin{aligned} h_a &= (g_{a,0})^{r_a} \\ h_b &= (g_{b,0})^{r_b} \end{aligned}$$

where h_a is send to Alice and h_b is send to Bob.

Both agents can test the result given by the OTP by using the factorization of m_a and m_b. Alice can check if $a > b$

$$h_a^{G_a} = h_a^{\frac{\phi(m_a)}{p_a}} \equiv 1 \bmod m_a$$

Similarly, Bob can check if $a < b$ by computing

$$h_b^{G_b} = h_b^{\frac{\phi(m_b)}{p_b}} \equiv 1 \bmod m_b$$

5 SECURITY

The original private bidding protocol by (Cachin, 1999) did not use ElGamal for the communication between the bidders. Our choice for ElGamal was defined by the use of the E-E-D algorithm, which also implies that we could not encrypt 0 because it did not lie in the group. Therefore, we had to add an extra parameter r_j to solve this problem.

Secure: Our protocol is secure if the bidding takes place in a honest-but-curious environment that does not conspire with the OTP e.g. the environment should not give the private-key of Alice to the OTP which is used to decrypt $w_{ab,j}$.

Private-key: The environment is able to spy on the agents' code and communication. Therefore it possesses the private decryption key of Alice $D_{ska,j}$. This private-key is only known by the environment and Alice, and should not be know by the OTP. This is not a threat, because one of the constraints of the model is that the OTP and the environment do not conspire. Even, if the environment decrypts the inputs, it is still not able to recover the plaintext because it needs the private-key of the OTP.

Conspire: The OTP and the environment should not conspire otherwise the OTP is able to learn the private inputs a, b of Alice and Bob. Using the private-key of Alice, the OTP is able to learn the plaintext content of $y_{a,j}$ and $y_{b,j}$ which are the transformed private bids of the agents. To do so, he also needs to know if $y_{b,j}$ corresponds with $b_j = 0$ or $b_j = 1$.

When the OTP possesses G_a or G_b, given by the environment, then it is capable of checking after every iteration in the final stage of the protocol when it contains a *p-th root* modulo m_a or m_b. This means the OTP will learn in which bit-position j the inputs a and b are different for the first time. It will also know that the bits before j are equal. Therefore, G_a and G_b should not be known to the OTP.

Fair: The protocol is fair because both Alice and Bob are able to check individually what the result is of the comparison computed by the OTP.

Efficient: While only having two messages between Alice and Bob and two messages between the bidders and the OTP, we can say the protocol is efficient. The efficiency is gained by using an OTP with standard cryptographic techniques instead of using circuits.

Random k: ElGamal is not secure if we us the homomorphic in addition property, because we have to keep the random k identical for every encryption. In our case we also keep the k the same but we claim this is secure. Alice is the only one who encrypts the messages $y_{a,j}$, $y_{b0,j}$ and $y_{b1,j}$ therefore she is the only one who knows the content of the messages; which is also not known to Bob. Suppose the same k is used to encrypt two messages x_1 and x_2 and the result is the ciphertext pairs (γ_1, δ_1) and (γ_2, δ_2). Then

$$\frac{\delta_1}{\delta_2} = \frac{x_1}{x_2} \tag{6}$$

is easily computed if x_1 or x_2 is known (Menezes et al., 1996). In our case, the messages are only

known to Alice and therefore no other entity is capable of computing (6). We therefore assume that the use of the identical k does not make our model insecure.

Cheating bidder: Alice initiates the private bidding, therefore she can cheat the protocol. This could be prevented by adding commitments and letting the OTP be more actively involved in the bidding process.

If Bob would collude with the environment and get access tot the private-key of Alice, he could use the OTP as an oracle by querying it and gaining access to her private data.

6 CONCLUSION

We have presented our private bidding protocol that keeps the bids of the bidders private if the communication is executed in an honest-but-curious environment and where the involved parties do not collude with one another. We have also seen that the system crumbles if one of the parties is malicious, and manipulates the data or communication. We have demonstrated that private bidding is possible for mobile software agents, but future work is needed to make the system more resilient.

REFERENCES

Algesheimer, J., Cachin, C., Camenisch, J., and Karjoth, G. (2001). Cryptographic security for mobile code. In *Proceedings of the IEEE Symposium on Security and Privacy*, page 2. IEEE Computer Society.

Cachin, C. (1999). Efficient private bidding and auctions with an oblivious third party. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*, pages 120–127, New York, NY, USA. ACM Press.

Cachin, C., Micali, S., and Stadler, M. (1999). Computationally private information retrieval with polylogarithmic communication. *Lecture Notes in Computer Science*, 1592:402–414.

Cartrysse, K. (2005). *Private Computing and Mobile Code Systems*. PhD thesis, Delft University of Technology.

Cartrysse, K. and van der Lubbe, J. (August 2004). Privacy in mobile agents. *First IEEE Symposium on Multi-Agent Security and Survivability*.

Claessens, J., Preneel, B., and Vandewalle, J. (2003). (how) can mobile agents do secure electronic transactions on untrusted hosts? a survey of the security issues and the current solutions. *ACM Trans. Inter. Tech.*, 3(1):28–48.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654.

ElGamal, T. (1984). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18. Springer-Verlag New York, Inc.

Goldreich, O. (2002). Secure multi-party computation. Final (incomplete) Draft, Version 1.4.

Menezes, A., Vanstone, S., and Oorschot, P. v. (1996). *Handbook of Applied Cryptography*. CRC Press, Inc.

Rabin, M. O. (1981). How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187. http://eprint.iacr.org/.

Sander, T. and Tschudin, C. F. (1998a). Protecting mobile agents against malicious hosts. *Lecture Notes in Computer Science*, 1419:44–60.

Sander, T. and Tschudin, C. F. (1998b). Towards mobile cryptography. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA. IEEE Computer Society Press.

Sander, T., Young, A., and Yung, M. (1999). Non-interactive cryptocomputing for NC 1. In *IEEE Symposium on Foundations of Computer Science*, pages 554–567.

Yao, A. (1982). Protocols for secure computations. In Carberry, M. S., editor, *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science, Chicago IL*, pages 160–164. IEEE Computer Society Press.