Hybrid Approach for Secure Mobile Agent Computations

J. Todd McDonald
Florida State University
Department of Computer Science
Tallahassee, FL 32306-4530
mcdonald@cs.fsu.edu

Abstract: Mobile agent applications are particularly vulnerable to malicious parties and thus require more stringent security measures—benefiting greatly from schemes where cryptographic protocols are utilized. We review and analyze methods proposed for securing agent operations in the face of passive and active adversaries by means of secure multi-party computations. We examine the strengths and weaknesses of such techniques and pose hybrid schemes which reduce communication overhead and maintain flexibility in the application of particular protocols.

1 Introduction

Mobile agents offer a unique method for implementing distributed applications. Itinerant agents have the ability to migrate among a preplanned or ad-hoc set of hosts where host inputs are gathered and agent code is executed. The agent carries both its static code and a dynamic data state which embodies all previous results of execution. Security concerns still occupy a large portion of the research effort associated with such mobile programs—both with protecting agents from malicious hosts and protecting hosts from malicious agents.

A multitude of schemes have been developed for mobile agent security and reviews of various mechanisms can be found in [MYT05, BC02, and JK00]. Much work has been done over the last few years to apply the field of theoretical cryptography to the mobile agent security problem [ST98, CC+00, NH+00, AC+01, YS02, TX03a, EM03, ZY03, TX04, and EW04]. By integrating cryptographic protocols based on secure multi-party computations (SMC), software-only protection mechanism can be designed to guarantee the execution integrity and data confidentiality of an agent while it is executed at a remote host.

The use of secure computation involves a trade-off between security, trust, and overhead. SMC protocols can have varying security attributes—whether at the information theoretic or computational level—and varying levels of communicational and computational overhead—normally considered unreasonable for practical applications. In this paper, we review specifically the use of these approaches for mobile agent security and pose hybrid approaches that offer greater efficiency and more flexibility in integrating SMC protocols.

We organize the paper as follows. Section 2 reviews literature related to secure computations while section 3 analyzes various efforts to integrate SMC as an agent protection mechanism. Section 4 poses several hybrid mobile agent approaches that minimize communication overhead and add more flexibility in the application of SMC protocols to security. Section 5 summarizes our contributions.

2 Secure Computations

Cryptographers have for some time sought how to perform a group function when there are a number of mutually or partially distrusting participants to the operation. Yao's blind millionaire problem [Yao86] is often cited as an early formulation for the two-party case where a function $z = f(x,y)$ is computed between Alice and Bob—without leaking any information about Alice's input x or Bob's input y other than what can be deduced from z itself. Goldreich and his colleagues in [GMW87] extend secure computation to $\boldsymbol{n}$ parties—defined in the general case as a publicly available function $\boldsymbol{f}$ that takes $\boldsymbol{n}$ private inputs and returns $\boldsymbol{n}$ private outputs: $f(x_1, x_2, x_3, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$. In some instances, all parties learn the same function output such that $y_1=y_2=\ldots=y_n$, making the output publicly known.

Secure computation is referred to synonymously as secure multi-party computation (SMC), *secure function evaluation* (SFE) or secure circuit evaluation. Various contributions from active research in the field can be found in [BGW88, CCD88, Kil88, AFK89, AF90, BMR90, Bea91, MR92, CF+96, CD+99, NPS99, CC+00, Gol00, HM01, NN01, CL+02, DN03, FG+04]. In terms of practical use, [DA01] summarize privacy-preserving, real-world applications that can be represented as an SMC problem such as database query, scientific

computations, intrusion detection, statistical analysis, geometric computations, and data mining. Malkhi et al. have developed a full programmatic implementation of a two-party secure function evaluator called Fairplay [MN+04] that uses oblivious transfer [Kil88, AFK89, AF90, BM90] and one-pass Boolean circuits [Yao86, GMW87, NPS99, and BMR90].

SMC protocols typically involve several rounds of interaction between parties and assume different types of communication channels including, for example, private channels between every two parties [BGW88, CCD88, RB90], a broadcast channel [RB89, BMR90], and broadcast subsets among player triples [FG+04]. In terms of security, the correctness and privacy of *any* protocol can be reduced to the evaluation of a secure function protocol [BMR90]. In the ideal setting, all parties to an SMC can send their inputs via a secure private channel to a trusted third party that computes the group function and return results fairly.

A primary security result concludes that any function computable with polynomial resources (communication and computation) can be transformed and computed in a *secure* manner using polynomial resources [NN01]. Corruption in multi-party computations deal either with an *honest-but-curious (semi-honest)* adversary that passively reads information from corrupted parties or an *active (malicious)* adversary that exerts full control over parties. Privacy of inputs is at issue in passive attacks while correctness of the outputs is more in view in active attacks. Goldreich concluded in [Gol00] that two parties acting maliciously can be forced to behave in a semi-honest manner or else be caught violating the security of the computation.

For any arbitrary function in the presence of an active adversary, the computation can still be securely accomplished as long as less than 1/2 of the players have not been corrupted [GMW87]. The unconditional security results found by [BGW88, CCD88] state that computations can occur as long as less than 1/3 of the players have been corrupted and secure channels exist in both directions between any two players. When broadcast channels are introduced, unconditional security is possible for the computation as long as less than 1/2 of the players are corrupt. Cachin and colleagues [CC+00] reiterate that computation between two unbounded parties with "full information" is not securely possible for arbitrary functions and only limited to trivial functions $\boldsymbol{g}$ where $g(x,y)$ reveals y. These results are significant when the multi-party computations are applied in the realm of mobile agents.

2.1 Evaluation Techniques and Primitives

Yao first posed the idea that a function $\boldsymbol{f}$ can be modeled and securely executed as a Boolean circuit [Yao82, Yao86] in a protocol known as *secure circuit evaluation*. The circuit can be "scrambled" in a way to secure host inputs and compute the group output. Abadi and Feigenbaum posed a two-player scheme in [AF90] where one player runs a secret program for another player who has a secret input. Other techniques for circuit construction including multi-party cases have been posed in [GMW87, CCD88, BGW88, CDG88, NPS99, and BMR90]. Once the function $\boldsymbol{f}$ is represented as a circuit, parties must run a protocol to evaluate *every* gate in the circuit.

Secure primitives in the circuit evaluation process include tools such as oblivious transfer (OT) [NP01, NP00, NP99] and verifiable secret sharing (VSS). Work by [FG+04] has sought to find minimal complete primitives to accomplish SMC and characterize security and efficiency of such tools beyond the two-party case. To accomplish secure circuit evaluation, the original wire signals for both inputs and outputs of the circuit are encrypted (garbled) and the actual wire signals used by the parties no longer have their same semantic meaning. In order to translate inputs and outputs to their true semantic meaning, data is exchanged between two parties in an oblivious manner—typically 1-of-2 OT [BM90].

While OT deals with privacy in circuit-based SMC, cheating can be addressed by verifiable secret sharing which allows a "dealer" to distribute shares of a piece of data among different parties [Sha79, ZY03]. Normally, parties in the computation must commit to their bits (which become garbled for purposes of evaluation) before they are used. However, no other party could tell whether the scrambled bits actually represent the real semantic meaning of a parties input. By using sharing techniques, parties give shares of their inputs so that any attempt to alter a commitment can be detected. Re-sharing of data to prevent a super adversary with control over some set of parties from gathering enough shares to compromise a system is discussed in [OY91, EM03].

Not all protocols are as secure as their authors envision. For example, a vulnerability is described in [TX03b] in the constant round circuit evaluation of [BMR90] where private information is leaked when gates within a circuit share a common input wire. Efficiency is also a major issue and much work has been done to improve protocols over time [GRR98, BF+90, CDN01, HM01, DN03]. Other more efficient methods than Boolean circuits can be used, for instance, to represent $\boldsymbol{f}$ such as permutation branching

programs, algebraic circuits, low degree and randomizing polynomials, and matrices over large fields [NN01]. Hurt and Meier [HM01] present a protocol that is secure for computing an n-party function with m multiplication gates in the presence of less than 1/3 actively corrupted players with complexity $O(mn^2)$.

Typically, SMC protocols have been adapted for synchronous networks and suffer from computational or communicational complexity too high for use in the real world. Mobile agents operate in asynchronous environments and therefore other factors must be taken into account before SMC techniques can be applied successfully. Work such as [BCG93, BKR94] offer frameworks for realistic network environments and Canetti has characterized the composable nature of security properties for different protocols operating across asynchronous networks in [Can00, Can01]. As [EM03, EW04] suggest, timeouts have to be integrated with distributed computations for asynchronous networks (that model the Internet) and the environment for mobile agent applications.

2.2 Single Round Computations

Mobile agents exhibit three unique properties that make using SMC protocols difficult: autonomy, mobility, and disconnected operations. All of the protocols mentioned thus far have relied on the exchange of information between parties in multiple rounds, including the originator of a function. Agents require non-interactive protocols because the originator of a function may be offline during the actual computation. Autonomy stipulates that the agent does not return home after the first host and can visit some set of known or unknown hosts. Mobility without the help of a trusted third party and minimal communication among parties is a primary goal of agent security schemes. As [RAD78, AF90] discuss, there are two ways to view single round computations between two parties in contrast to traditional secure function evaluation: *computing with encrypted data* and *computing with encrypted functions.*

Computing w/ Encrypted Data (CED)	Alice has input x while Bob holds function f(·). Alice sends an encrypted version of x to Bob who computes and sends the result back to Alice in a single round of interaction. Alice decrypts the result to get f(x) while Bob does not learn x.
Computing w/ Encrypted Functions (CEF)	Alice holds the function f(·) while Bob holds input y. In one-round, Alice sends to Bob an encrypted version of f(·) who provides his input y. Alice receives back and decrypts Bob's result to learn f(y) but does not learn y while Bob does not learn f(·)
Secure Function Evaluation (SFE)	Alice and Bob have private inputs to the function f(x,y). Alice and Bob jointly compute the function f(x,y) in one round of computation. Alice learns only the result (and nothing more) while Bob learns neither the result nor Alice's private input.

CEF represents the mobile agent transaction scheme best and can be extended easily to a multiple host approach. Sander and Tschudin posed one of the first non-interactive CEF approaches for mobile code execution based on homomorphic encryption in [ST98]. Their results were extended to include any function implemented by logarithmic-size circuits in [SYY99]. Cachin *et al*. in [CC+00] developed a non-interactive protocol (which we will refer to as the CCKM scheme) that could be used to evaluate all polynomial time functions via the use of a scrambled circuits and oblivious transfer.

Several important results were derived from [CC+00]: 1) for unbounded passive adversary, any function computable by a polynomial-size circuit can be computed securely; 2) for a bounded active adversary, any function computable by a polynomial-size circuit can be computed securely, given a public-key framework; and 3) any function computable by a polynomial-sized circuit has a one-round secure computation in the model. Non-interactive SMC approaches and results are summarized by [CC+00] as follows:

[BGW88]	Trivial functions where A and B are unbounded
[ST98]	Functions representable as polynomials, B is bounded
[SYY99]	Functions computable by logarithmic-depth circuits, B is bounded
[CC+00]	Functions computable by polynomial-depth circuits, only A is bounded or both A and B are bounded

The CCKM methodology is foundational to several approaches for mobile agent security based on secure multi-party computation, discussed in the next section.

3 Integrating SMC with Agents

Mobile agent applications have brought a practical relevance to development of secure, efficient cryptographic protocol schemes. The goal of SMC has been stated as guaranteeing the correctness of a function and the privacy of results among the parties. In mobile code systems, similar notions exist: malicious hosts can spy on the code, state, or results of mobile agents that they execute. Hosts can gain unfair advantages by altering the normal sequence of execution, replaying agent computations using different inputs, or altering the state information

present in the agent. Software-only approaches to mobile agent security that are secure, efficient, and removing need for trusted relationships have been the holy grail in the research field for quite some time.

There are two primary approaches to integrating SMC protocols with mobile agents: use single agents that implement single-round non-interactive protocols or use multiple agents that execute multi-round SMC protocols in coalition schemes. We discuss approaches and issues with the former next.

3.1 Non-Interactive Approaches

To formulate a single-round secure multi-party computation, the following formal notation from [CC+00, AC+01] is used: an agent originator O embodies a private function to be executed by a set of hosts $H_1,\ldots,H_l$. Two functions—$g_j(\cdot)$ and $h_j(\cdot)$—describe the computation of an agent in terms of a state $x \in X$ and a host input $z \in Z$. Figure 1 illustrates the interaction of an agent which is captured by a multi-party computation. The state update function g_j takes a current state (brought by an agent from the previous host) and the local host input and produces a next state x_j. The host output function h_j takes the current state (brought by the agent from the previous host) and its own local input to produce its own local output.

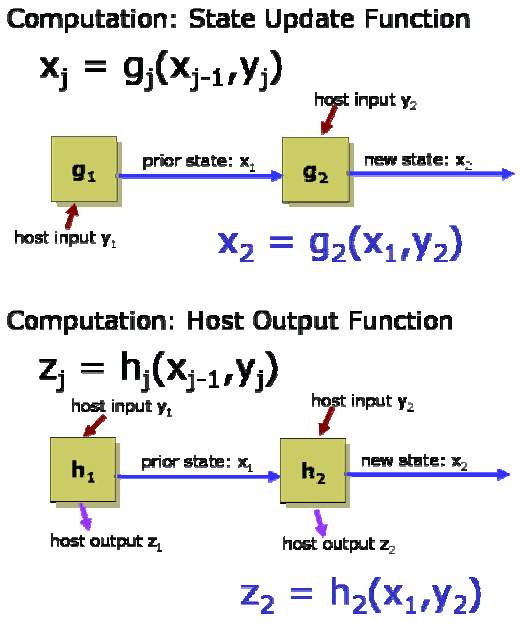

Figure 1: Formalizing the Agent Computation

In the CCKM protocol, once the agent computation is represented as a Boolean circuit and encrypted, translation tables are required to map actual signals to scrambled signals. The circuit encoding is based on Yao's two-party SFE protocol in [Yao86]. In order to know what signals to use for their local input, a host performs oblivious transfer with the originator to get a set of scrambled signals, and the originator does not know which signals are chosen. The following security properties are thus established: 1) the originator has privacy of the function; 2) each host has privacy in respect to their local input. The CCKM approach allows for autonomy in the agent path by creating an encrypted circuit that is a cascade of sub-circuits. Each host in the route of an agent's path would receive an encrypted circuit on which their input is applied. However, the CCKM protocol did not address the ability for each host to use the "unencrypted" local output of the agent because it was still encrypted and could only be evaluated by the originator.

Extending the CCKM approach further, Algesheimer et al. [AC+01] produced a non-interactive protocol (which we refer to as the ACCK protocol) similar to the trusted hardware of [LM99] that would allow for secure decryption of host output when CEF is used. The ACCK scheme, illustrated in figure 2, makes use of a trusted generic computation service which is roughly equivalent to the trust we place in a public key infrastructure. To decrypt the output of the agent at the local host, the mappings for the semantics of the signals are encrypted with the public key of generic service. Each host accomplishes oblivious transfer with the generic service (instead of the originator who may be offline) to decrypt the signals for the output.

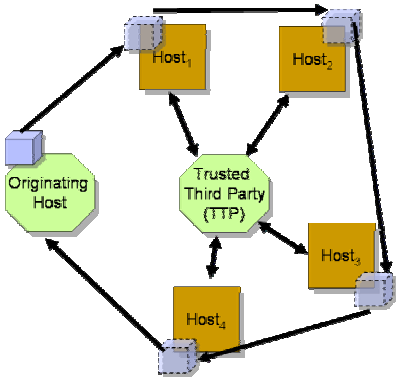

Figure 2: ACCK Protocol w/ Generic Computation Service [AC+01]

By using a secure middleman, the ACCK protocol allows inputs, outputs, and computations of all hosts to be hidden from the originator as well as any other host visited by the agent. The main assumption is that this trusted third party (TTP) does not collude with the originator or with any host, but as proposed would offer a generically secure service for *any* application.

There have been two extensions proposed to the ACCK protocol that target replacement of the TTP in

some form. Zhong and Yang in [ZY03] introduce a cryptographic primitive called verifiable distributed oblivious-transfer (which we refer to as the VDOT protocol) and Tate and Xu in [TX03a] introduce a multi-agent approach utilizing their oblivious threshold decryption (which we refer to as OTD). Figure 3 shows a notional arrangement of parties in the VDOT scheme while figure 4 shows a notional arrangement of parties in the OTD approach.

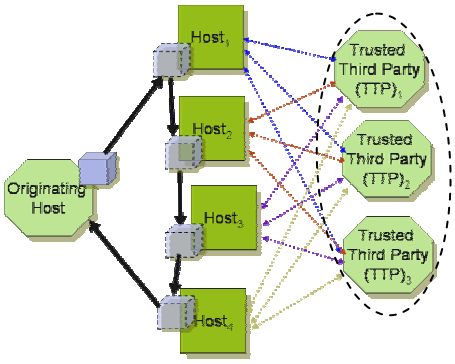

Figure 3: VDOT Protocol [ZY03]

In the VDOT protocol, mobile agent computations are divided into security-sensitive and non-security-sensitive portions. Code that requires integrity or confidentiality is transformed into a garbled Boolean circuit. Instead of interactions with one trusted third party, which has weaknesses involving the corruption of a single server to the detriment of the entire system, several trusted third party servers are used to replicate the functionality of TTP in the VDOT approach. VDOT guarantees with high probability the correctness of receiver's output, enforcement of the code and state privacy, protection from coalitions of malicious hosts and malicious TTPs, and the verification that servers give correct decryption of host signals.

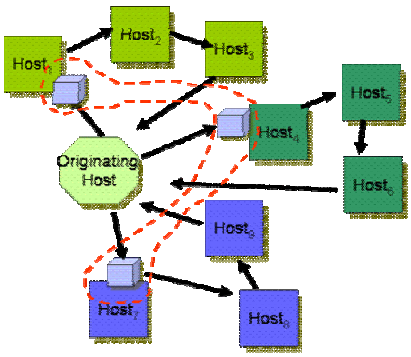

Figure 4: OTD Protocol [TX03a]

Distribution of trust among a group of servers strengthens the original ACCK protocol and forces the table lookup for circuit signals to be performed by a group of servers that hold shares of the decryption. The VDOT protocol is general purpose in the sense that each host need only provide an interpreter for garbled circuits. By using distributed oblivious transfer, trusted third parties act as a proxy for agent owners and provide translation tables for host inputs without being able to discover host inputs themselves. Obvious disadvantages to the approach are increased communication complexity (which the authors contend is negligible in practice) and the complexity of breaking a program into security sensitive portions represented by a Boolean circuit.

The OTD protocol of [TX03a] is similar in some regards to VDOT but actually eliminates the trusted third-party requirement altogether. As a primary distinction, their approach relies on multiple agents that are dispatched to disjoint sets of the possible host pool. Each of these agents act in a threshold manner (similar to VDOT) to decrypt the encrypted signals for a given host input without relying on the TTP. While the ACCK secure computation service overcame the interaction requirement of Yao's encrypted circuit evaluation—a limiting factor in the mobile code paradigm—OTD replaces this by means of cryptographic operations and multiple agents that cooperate together.

Multiple agents must agree before decryption of the host's input signals can occur and this in turn prevents cheating by keeping a list of hosts that have already decrypted a signal. Agents eventually return back to the originating host where all circuit results are decrypted and combined to produce a final result. The security in this method rests on the security of Yao's secure circuit evaluation, the security of the 1-out-of-2 oblivious transfer, and the strength of threshold cryptography. However, this protocol does not support free-roaming agents and requires knowledge of the set of hosts an agent will visit.

Algesheimer et al. in [AC+01] state the ACCK protocol does not require foreknowledge of the agent's path or the hosts that the agent will visit. Their approach upholds the disconnected and autonomous nature of a mobile agent. However, it is not clear whether the number of host, ℓ, must be specified or known beforehand. The OTD and VDOT extensions both assume a known number of hosts or subsets of hosts in order to design the circuit representation of the group function—thus limiting a true free-roaming dynamic itinerary.

Though single-round non-interactive protocols reduce the communication overhead for SMC, message sizes increase proportionally, regardless of input or output size. Tate and Xu, for example, state that it roughly takes 9k bytes to encrypt 32 bits of secret data [TX03a] under this scheme. Zhong and Yang mitigate overhead by keeping security sensitive

portions separate from normal programmatic requirements. Using multiple round SMC (reviewed in section 2) offers another approach to accomplishing secure transactions with mobile agents, which we analyze now.

3.2 Multiple Round Approaches

Secure multiparty computations have a tradeoff between trust and efficiency. Neven et al. [NH+00] were one of the first to envision the use of agents to implement SMC and reduce the overhead of the communication itself. Figure 5 summarizes four different approaches to integrating agents with hosts to accomplish SMC. Figure 5-a illustrates the ideal world where agents carry host inputs to a trusted third party and a protocol is evaluated without the expense of network broadcasts or bidirectional secure channels. In the context of the TTP, all parties can evaluate the protocol and the TTP is assumed to behave honestly with respect to host inputs.

The most secure but least efficient method is seen if figure 5-b: here hosts simply become the execution environment and setup a multi-party protocol evaluation. In this case, both the computational and communicational complexity inherent in the chosen protocol must be faced and only high speed links (represented by the dotted lines) make such protocols practical. Single-round approaches discussed in the previous section are seen in figure 5-c where an agent embodies the circuit to be securely evaluated and each host provides private input as the agent migrates. In [NH+00], a hybrid solution was posed as depicted in figure 5-d where high speed communication links are present between one or more hosts. Participants in the n-party protocol send agents carrying their private inputs to one of these intermediate TTPs who can then efficiently and securely evaluate the function according to the rules of the protocol.

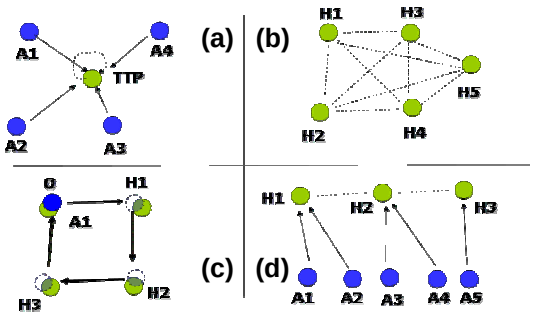

Figure 5: Agent Approaches to SMC

In the realm of mobile agents, as with many real world applications, it is preferable not to rely on a trusted third party and just perform an SMC among the parties of a function. Endsuleit and Mie utilize a group of multiple agents to support such an approach in [EM03]. In their model, multiple agents carrying the same realized circuit are deployed to remote hosts where rounds of the secure protocols are evaluated among parties. Figure 6 illustrates that agents are located on some set of hosts and implement *multi-agent* computations based on some underlying SMC protocol. In [EM03] the authors assume the extensive use of a broadcast channel and suggest the protocol of [BGW88] with an implementation of secret sharing from [Sha97]. In [EW04], follow-on work suggests the use of more efficient protocols such as those of [HM01].

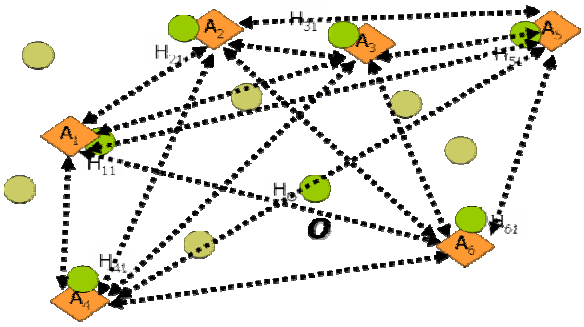

Figure 6: Multi-*Agent* Secure Computation

A nice feature of such multiple agent schemes is that *any* SMC protocol can be used as long as it meets the composable security properties defined by Canetti [Ca01]. In order to adapt the Canetti model, which assume stationary parties, "slices" are defined in [EM03, EW04] as periods where a community of *n* agents is executed by a set of *n* different hosts with no migrations during that period. Resharing of data shares via the Ostravsky and Yung method [OY91] is used to also overcome the adverse affects of migration where malicious hosts can use acquired shares over time to compromise security.

The system supports self-repairing code and threshold agreement of computations, as long as up to 1/3 of the community (agents or hosts) has not been compromised. The security results mentioned in section 2 follow because Canetti establishes proof of a secure protocol for n parties computing a joint function in the presence of an active adversary corrupting up to some *k* limited servers. By using such agents to implement a redundantly shared global state of computation and coordinate activity, a wide variety of SMC protocols can be implemented. However, as with any multi-round solution, the communication complexity is extremely high and the originator must know a priori which hosts will be part of the computation.

In [Dad04], another software-only scheme is presented that implements multiple agents acting in a threshold manner similar to [TX03a, EM03, EW04].

However, there approach does not suppose the presence of collusions among hosts or rely necessarily on multiparty protocols. Their approach, which is termed Remote Distribution Scheme or RDS, depends on a set of agents that replicate and share a transaction set. RDS also assumes a publicly known algorithm which does not necessarily correspond to the mobile agent setting where code privacy is required or CEF is being implemented.

4 Hybrid Approaches

SMC offers many advantages for securely accomplishing a group transaction. There are several approaches, some already mentioned, to define how an agent implements a circuit that is part of a multi-party computation. The originator can send a single agent with a cascading circuit whose last migration signals the last computation of the circuit [CC+00, AC+01, ZY03]. Alternatively, the originator can send multiple agents with the same circuit that executes protocols in stepwise multi-round fashion [EM03, EW04]. A single or set of trusted execution sites can also be used to accomplish the SMC interaction [NH+00]. By combining these techniques where full protocols, multiple agents, and semi-trusted hosts are utilized, several advantages can be gained.

Malkhi et al. [MN+04] note a recent trend in SMC research where protocols are focused on specific application contexts—thereby allowing more efficient representations for specific tasks. This will be true in the mobile agent paradigm as well—whereby mobile agents will be used for specific tasks like auctions, trading, or secure voting. Fiegenbaum *et al.* [FP+04] implement a secure computation mechanism utilizing SMC for collecting survey results with sensitive information. Their scheme uses data-splitting techniques and traditional Boolean circuit evaluation Yao-style [Yao86]. Notably, it also uses a secure computation server, which acts in the role of a trusted entity within the system, and is the initiator of the 2-party function evaluation. We use this as an example to point out that in practical applications where true data privacy or true function privacy is needed, the presence of a trusted server is not beyond the realm of possibility. In fact, many agent applications which will be executed "in-house" will indeed benefit from the availability of such trusted entities.

Implementations of SMC in mobile agent systems must seek to reduce message size, number of broadcast or pair-wise channels required, and the size of the circuit. To accommodate agent goals such as disconnected operations, the originator typically remains offline during the protocol evaluation. Agent autonomy requires the task to be accomplished by an agent that decides where and when to migrate. The requirement for full autonomy in the agent path and itinerary lends itself best to a combination of SMC that balances trust with efficiency. While there is a desire to eliminate the need or requirement for any trusted third party or trusted computation service (like PKI), some environments for SMC may be conducive to such assistance.

Non-interactive approaches are limited to a very small number of protocols that derive from [ST98] or [CC+00]. Single-round approaches do not require trusted third parties but come with large message sizes and their own set of limitations which include reliance on a trusted *entity* similar to a PKI. Extensions to the non-interactive approach such [ZY03, TX03a] require foreknowledge of at least the number of hosts to be visited by the agent or the set of hosts themselves. As SMC protocols find better and more efficient means of expression over time (other than Boolean circuits), agent security approaches should be adaptable to integrate them as they improve. However, accomplishing multi-agent fully interactive protocols comes with stiff communicational costs.

We pose several hybrid approaches to SMC integration with mobile agents that can accommodate free-roaming itineraries as well as reduce overall communication cost. These approaches can be used to take advantage of the security properties of multiparty protocols while remaining flexible for integration of other protocols with higher efficiency in the future.

4.1 Invitation and Response

In our first approach, which we term "Invitation and Response", a multi-agent architecture is used with a form of semi-trusted execution sites. We define the protocol informally first and define two *classes* of agents: the *invitation* agent and the *protocol* agent. The originator, O, begins the task by sending an invitation agent which has some initial set of hosts to be visited or at a minimum the first host to be visited. Invitation agents are free-roaming and can make changes in their itinerary based on environmental conditions or information obtained from hosts or other information services.

To guard the invitation agent against data integrity and denial of service attacks, two different schemes can be used. First, a traditional data encapsulation technique can be used with the stipulation that the agent code itself is bound to the dynamic state of each agent instance. Many data encapsulation protocols are reviewed in [MYT05, JK00] and figure 7 depicts only one invitation agent being used. A second approach is to use multiple invitation agents

with overlapping and redundant itineraries that reduce the possibility of malicious corruption. Each invitation agent has a uniquely identifiable code/state (to avoid replay attacks), but the collection of agents represents only a single uniquely identifiable task (such as a specific auction). If a host receives an agent requesting participation in the same unique event, it ignores subsequent requests much like network devices that only forward packets once.

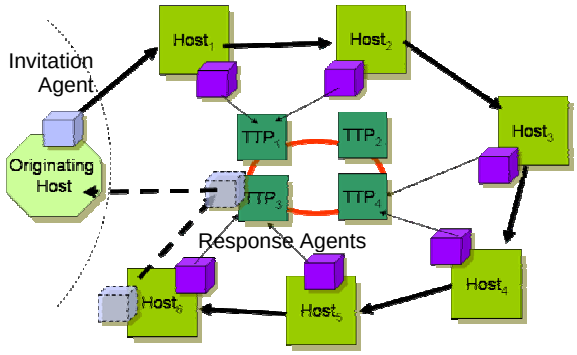

Figure 7: Invitation and Response Protocol

Invitation agents carry with them the specifications for input corresponding to an originator's task. The specification represents the normal query for a host input which is part of a multiparty computation. Hosts will (or will not) respond to this invitation by dispatching a response agent. The response agent is based upon an underlying secure multi-party computation protocol and can be created in different ways.

First, the invitation agent can carry the code for the response agent which each host will use. The host will execute the response agent first on its local input and then send the response agent to a semi-trusted execution location to actually evaluate the circuit. The second approach involve the *dynamic* generation of the code and circuit by the invitation agent when a host responds positively. A third method would involve each host responding to the invitation by sending its input encrypted to the semi-trusted execution site. This method resembles the traditional notion of the ideal SMC environment where parties send their input to a TTP for execution of the protocol.

Regardless of the method chosen, response agents migrate and move to a set of semi-trusted host environments in order to evaluate the protocol. The semi-trusted hosts can be specifically designed to serve multi-party computations (predefined based on some underlying protocol) or can simply provide basic agent execution environments with communication facilities. The key characteristic of these hosts are that they are connected by a high bandwidth network so communication costs are negligible. This corresponds to the SMC approach seen in figure 5-d where a tradeoff is made with overhead by bringing agents closer together through the availability of a high speed communication link among the servers. Environments are semi-trusted because group and threshold operations can be accomplished to eliminate the full trust in any one

In terms of security, "invitation and response" has the following properties. Hosts can only send one agent to the computation which removes the possibility the circuit can be evaluated on multiple host inputs. As long as multiple host submissions (and therefore cheating) are detectable, the originator's privacy is preserved. The local host input is kept private under two scenarios: 1) when the execution sites are fully trusted, as depicted in figure 8, no extra security is required and each execution site is expected to maintain privacy of host inputs; 2) when the execution sites are semi-trusted, as depicted in figure 9, a threshold mechanism can be used to distribute the trust among the set of hosts for decryption operations of circuit operations.

The advantages of this hybrid approach include the ability to accommodate true free-roaming agent scenarios and to use any type of secure multi-party protocol for the evaluation of the secure function. Protocols which have high communication and low computational complexity can thus be favored because agents are sent to a semi-trusted environment that has an assumed high-speed link among execution sites. Depending on the trust level of the application environment, fully trusted hosts may be a possibility and simpler protocols can be utilized that do not involve threshold decryption of signals.

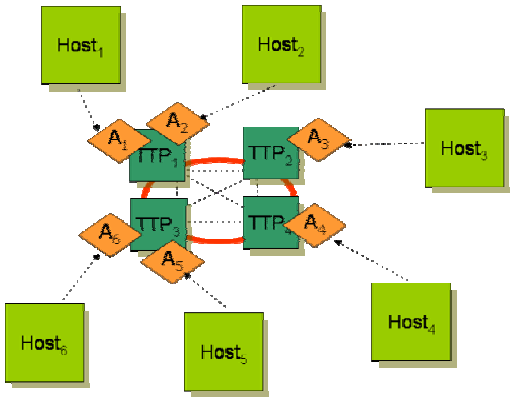

Figure 8: Fully Trusted Evaluation

The selection of execution environments becomes one of the issues with the invitation and response protocol. The two primary factors are the presence of a high speed communications link between servers and a common trust level among all parties of the protocol with the trusted servers. Migration of agents

also becomes more structured as the only free-roaming portion of the task is to find interested parties to the computation itself. Response agents only make two subsequent migrations: to the trusted server environment and then back to the originator, who can decrypt the final agent state and obtain the result.

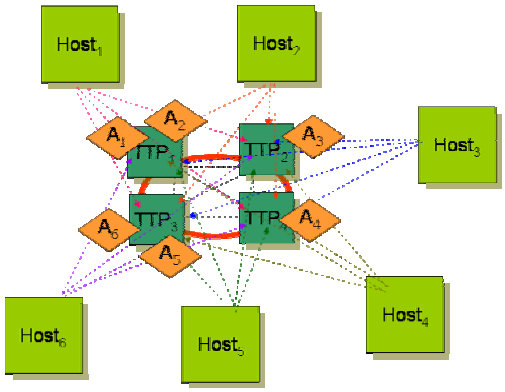

Figure 9: Semi-Trusted Evaluation

One of the issues well discussed in [AC+00] with SMC and agents is how can an individual host get its local output (function *h(x,y)* seen in figure 1) as the mobile agent migrates. In invitation and response, local host output can be handled in one of two ways. First, since the host output is not private in terms of the originator, O can be responsible for providing the output to each host after the evaluation of the secure function on the execution environment and after response agents migrate back to the originator. Second, the set of TTPs can each send their share of the output or the single TTP can send the output corresponding to a host back to it, through message passing or another class of agent.

4.2 Multi-Agent Trusted Execution

When the itinerary of an agent is known beforehand, simpler agent architecture can be used to facilitate trusted execution. Several configurations are possible for host environment in terms of a secure computation. First the host can be the computation environment for a cascaded circuit that requires only one round of execution. Next, the host can communicate with a semi-trusted party to evaluate an encrypted circuit or can communicate with a threshold of semi-trusted parties that provide signal decryption services in an oblivious manner. The host can also be the computation environment for a multi-round circuit and can be visited by more than one agent.

As figure 10 and 11 illustrate, multiple agents can be used to initiate a multi-party protocol among a predefined set of hosts. Similar to the multi-agent approaches of [EM03, EW04], multi-agent trusted execution would allow agents to migrate to hosts where input is first gathered. When one trusted execution environment is fully trusted by all parties, agents can then migrate there to accomplish a multi-round protocol, as suggested in [NH+00].

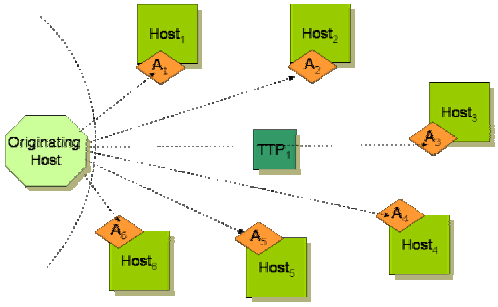

Figure 10: Fully Trusted Middle-man

If a less trusted set of execution environments are in view, figure 11 represents that trusted parties need to be linked by a high bandwidth communications network. In either case, agents are dispatched to hosts in the manner of a traditional multi-party function as the first step of the task. Once agents obtain host input they then migrate to a centralized trusted execution site where the multi-round protocol is evaluated. In performing such an operation, the goal again is to minimized the communication overhead of the network while maximizing the benefit of any given SMC protocol that is chosen.

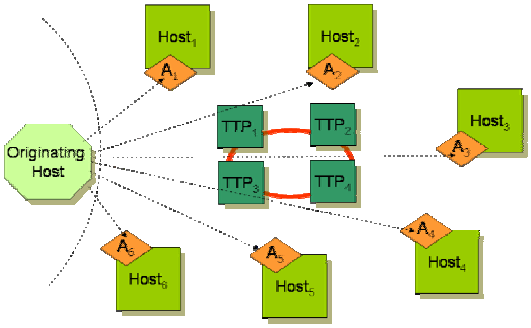

Figure 11: Semi-Trusted Middle-men

5 Conclusions

There is a distinct trade-off when using secure multiparty computations with mobile agent applications. The overhead of both computation and communication are barriers which must be overcome before protocols can be used in a practical manner. We have reviewed the state of the art in such integration approaches and posed variations of hybrid approaches that utilize fully trusted or semi-trusted execution environments for secure multi-agent computations. These schemes offer an alternative to

other architectures posed which combine the best of non-interactive approaches and multi-round SMC approaches. Future work will involve the formal description of such protocols and an analysis of their overhead when specific SMC protocols are in view.

6 References

[AF90] ABADI, M. and FEIGENBAUM, J. Secure circuit evaluation: A protocol based on hiding information from an oracle. *Jour. of Cryptology*. 2 (1990), 1-12.

[AFK89] ABADI, M., FEIGENBAUM J., and J. KILIAN. On hiding information from an oracle. *Jour. of Computer and System Sciences*. 39 (1989), 21-50.

[AC+01] ALGESHEIMER, J., CACHIN, C., CAMENISCH, J., and KARJOTH, G. Cryptographic security for mobile code. *Proc. IEEE Symposium on Security and Privacy* (May 2001), 2-11.

[Bea91] BEAVER, D. Foundations of secure interactive computing. In *Proc. of the 11th Annual International Cryptology Conference on Advances in Cryptology*, *LNCS 576*, pp. 377-391, Springer-Verlag, 1991.

[BC02] BIERMAN, E. and CLOETE, E. Classification of malicious host threats in mobile agent computing. In *Proc. of the 2002 Annual Research Conf. of the South Africa IoCS and IToETT*, Port Elizabeth, South Africa, 2002, pp. 141 – 148. ISBN:1-58113-596-3.

[BCG93] BEN-OR, M., CANETTI, R., and GOLDREICH, O. Asynchronous secure communications. In *Proc. of 25th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 52-61, ACM 1993.

[BF+90] BEAVER, D. FEIGENBAUM, J., KILIAN, J., and ROGAWAY, P. Security with low communication overhead. In *Advances in Cryptology—CRYPTO '90, LNCS 37*, Springer Verlag, 1990.

[BGW88] BEN-OR, M., GOLDWASSER, S., and WIGDERSON, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of Annual ACM Symposium on Theory of Computing'88*, pp. 1-10, 1988.

[BKR94] BEN-OR, M., KELMER, B., and RABIN, T. Asynchronous secure computations with optimal resilience. In *13th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 183-192, 1994.

[BM90] BELLARE, M. and MICALI, S. Non-interactive oblivious transfer and applications. *Advances in Cryptology—CRYPTO '89*, pp. 547-559, Springer-Verlag, 1990.

[BMR90] BELLARE, M., MICALI, S., and P. ROGAWAY. The round complexity of secure protocols. In *Proc. of 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 503-513, 1990.

[Can01] CANETTI, R. Universally composable security: a new paradigm for cryptographic protocols. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science*, p.136, October 14-17, 2001.

[Can00] CANETTI, R. Security and composition of multiparty cryptographic protocols. *Jour. of Cryptology*, 13:1 (2000), 143-202.

[CC+00] CACHIN, C., CAMENISCH, J., KILIAN, J., and MÜLLER, J. One-round secure computation and secure autonomous mobile agents. In MONTANARI, U., ROLIM, J.P., WELZL, E., editors, *Proc. 27th International Colloquium on Automata Languages and Programming (ICALP), LNCS 1853*, pp 512-523, Springer-Verlag, 2000.

[CCD88] CHAUM, D., CRÉPEAU, C., DAMGARD, I. Multiparty unconditionally secure protocols (extended abstract). In *Proc. of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, Chicago, Illinois, pp.11-19, May 2-4, 1988.

[CDG88] CHAUM, D., DAMGARD, I., and VAN DE GRAAF, J. Multiparty computations ensuring privacy of each party's input and correctness of the result. In, POMERANCE, C., editor, *Proc. CRYPTO '87*, *LNCS 293*, 1988.

[CDN01] CRAMER, R., DAMGARD, I., NIELSEN, J.B. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT '01, LNCS 2045*, pp. 280-300, 2001.

[CD+99] CRAMER, R., DAMGARD, I., DZIEMBOWSKI, S., HIRT, M., and RABIN, T. Efficient multiparty computations with dishonest minority. In STERN, J., editor, *Proc. of EUROCRYPT 99*, *LNCS 1592*. IACR, Springer-Verlag, 1999.

[CF+96] CANETTI, R., FEIGE, U., GOLDREICH, O., and NAOR, M. Adaptively secure multi-party computation. In *Proc. of the 34th Annual ACM Symposium on Theory of Computing (STOC),* pp. 639-648, 1996.

[CL+02] CANETTI, R., LINDELL, Y., OSTROVSKI, R., and SAHAI, A. Universally composable two-party and multi-party secure computation. In *Proc. of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 494–503, 2002.

[Dad04] DADON-ELICHAI, A. RDS: Remote distributed scheme for protecting mobile agents. In *Proc. of AAMAS'04, July 19-23, New York, New York*. ACM, 2004.

[DA01] DU, W. And ATALLAH, M. Secure multi-party computation problems and their applications: a review and open problems. In *Proc. of New Security Paradigms Workshop*, Cloudcroft, NM, USA, pp. 11-20, Sept 2001.

[DN03] DAMGARD, I. and NIELSEN, J. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Proc. Advances in Cryptology - Crypto 2003*, *LNCS 2729,* pp. 247-264, 2003.

[EM03] ENDSULEIT, R. and MIE, T. Secure multi-agent computations. In *Proc. of Int. Conf. on Security and Management*, vol. 1, pp. 149-155. CSREA, 2003.

[EW04] ENDSULEIT, R. and WAGNER, A. Possible attacks on and countermeasures for secure multi-agent computation. In *Proc. of Int. Conf. on Security and Management* (SAM), pp. 221-227, 2004.

[FG+04] FITZI, M., GARAY, J., MAURER, U., and OSTRAVSKY, R. Minimal complete primitives for secure multi-party computation. *Jour. of Cryptography*, 18 (2005), 37-61.

[FP+04] FEIGENBAUM, J., PINKAS, B., RYGER, R., and SAINT JEAN, F. Secure computation of surveys. *EU Workshop on Secure Multiparty Protocols,* 2004.

[GMW87] GOLDREICH, O., MICALI, S., and WIGDERSON, A. How to play any mental game. In *Proc. of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 218-229, 1987.

[Gol00] GOLDREICH, O. Secure multi-party computation. Working draft, version 1.2, March 2000.

[GRR98] GENNARO, R., RABIN, M.O., RABIN, T. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proc. 17th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 101-111, 1998.

[HM01] HIRT, M. and MAURER, U. Robustness for free in unconditional multi-party computation. In *Proc. of Cypto'01, LNCS 2139*, pp. 101-118, Springer-Verlag, 2001.

[JK00] JANSEN, W. and KARYGIANNIS, T. NIST Special Publication 800-19 - Mobile Agent Security. National Institute of Standards and Technology, 2000.

[Kil88] KILIAN, J. Founding cryptography on oblivious transfer. In *Proc. of 20th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 20-31, 1988.

[LM99] LOUREIRO, S. and MOLVA, R. Function hiding based on error correcting codes. In BLUM, M. and LEE, C. H., editors, *Cryptographic Techniques and E-Commerce*, *Proc. of the 1999 Int'l Wrkshp on Cryptographic Techniques and E-Commerce (CrypTEC '99*), City University of Hong Kong Press, 1999.

[MN+04] MALKHI, D., NISAN, D., PINKAS, B., and SELLA, Y. Fairplay—A secure two-party computation system. In *Proc. Usenix Security Symposium 2004*, pp. 287-302, August 2004.

[MR92] MICALI, S. and ROGAWAY, P. Secure computation. In *Advances in Cryptology—CRYPTO '91*, *LNCS 576*, pp, 392-404. Springer-Verlag, 1992.

[MYT05] MCDONALD, J.T., YASINSAC, A., and THOMPSON, W. A survey on mobile agent security. Technical report, TR-050329, Dept. of Computer Science, Florida State University. Available, http://www.cs.fsu.edu/ research/reports/TR-050329.pdf.

[NN01] NAOR, M. and NISIM, K. Communication complexity and secure function evaluation. In *Electronic Colloquium on Computational Complextiy (ECCC)*, 8(62), 2001.

[NP01] NAOR, M. and PINKAS, B. Efficient oblivious transfer protocols. In *Proc. of SODA 2001 (SIAM Symposium on Discrete Algorithms)*, Jan. 7-9 2001, Washington DC.

[NP00] NAOR, M. and PINKAS, B. Distributed oblivious transfer. *Proc. Advances in Cryptology -- Asiacrypt '00*, *LNCS 1976,* Springer-Verlag, pp.200-219, December 2000.

[NP99] NAOR, M. and PINKAS, B. Oblivious transfer and polynomial evaluation. In *Proc. of the 31st Annual ACM Symposium on Theory of Computer Science (STOC),* Atlanta, GA, pp. 245-254, May 1-4, 1999.

[NPS99] NAOR, M., PINKAS, B., and SUMNER, R. Privacy preserving auctions and mechanism design. In *1st ACM Conference on Electronic Commerce,* pp. 129-139, 1999.

[NH+00] NEVEN, G., VAN HOEYMISSEN, E., DE DECKER, B., and PIESSENS, F. Enabling secure distributed computations: semi-trusted hosts and mobile agents. *Networking and Information Systems Journal*, 3:1-18, 2000.

[OY91] OSTRAVSKY, R. and YUNG, M. How to withstand mobile virus attacks. In *Proc. of the 10th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 51-59, 1991.

[RB89] RABIN, T. and BEN-OR, M. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. of the 21st Annual ACM Symposium on Theory of Computing,* Seattle, Washington, pp.73-85, May 14-17, 1989.

[RAD78] RIVEST, R.L., ADLEMAN, L, and DERTOUZOS, M.L. On data banks and privacy homomorphisms. In DEMILLO, R.A., DOBKIN, D., JONES, A., and LIPTON, R., editors, *Foundations of Secure Computation,* pp. 169-177, Academic Press, 1978,

[Sha79] SHAMIR, A. How to share a secret. *Communications of the ACM*, 22:11 (1979), 612-613.

[ST98] SANDER, T. and TSCHUDIN, C. Protecting mobile agents against malicious hosts. In VIGNA, G., editor, *Mobile Agents and Security, LNCS 1419*, pp. 44-61, Springer-Verlag, 1998.

[SYY99] SANDER, T., YOUNG, A., and YUNG, M. Non-interactive cryptocomputing for NC^1. In *Proc. of the 40th IEEE Symposium on Foundations of Computer Science*, pp. 17-19, 1999.

[TX03A] TATE, S.R. and XU, K. Mobile agent security through multi-agent cryptographic protocols. In *Proc. of the 4th International Conference on Internet Computing (IC 2003)*, pp. 462-468, 2003.

[TX03b] TATE, S.R. and XU, K. On garbled circuits and constant round secure function evaluation. CoPS Lab Technical Report 2003-02. Available http://cops.csci.unt.edu/publications/2003-02/2003-02.pdf, 2003.

[TX04] TATE, S.R. and XU, K. Universally composable mobile agent computation. In *Proc. of the 7th Information Security Conference (ISC'04)*, Sept. 2004.

[Yao82] YAO, A.C. Protocols for secure computation. In *Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science,* 1982.

[Yao86] YAO, A.C. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symposium on Foundations of Computer Science,* pp. 162-167, 1986.

[YS02] YOKOO, M. and SUZUKI, K. Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions. In *Proc. of the 1st International Joint Conference on Autonomous agents and Multiagent Systems*, ICAA'02, Bologna, Italy, pp. 112 – 119, 2002.

[ZY03] ZHONG, S. and YANG, Y.R. Verifiable distributed oblivious transfer and mobile agent security. In *Proc. of the 2003 Joint Workshop on Foundations of Mobile Computing*, September 2003.