The Use of Encrypted Functions for Mobile Agent Security*

Hyungjick Lee†
Connectivity Lab., Digital Media R&D Center
Samsung Electronics Co., Ltd
416, Maetan-3Dong, Paldal-Gu,
Suwon City, Gyeonggi-Do, South Korea
hyungjick.lee@samsung.com

Jim Alves-Foss and Scott Harrison
Center for Secure and Dependable Systems
University of Idaho
Moscow, ID 83484
{jimaf, harrison}@cs.uidaho.edu

Abstract

Mobile agent technology is a new paradigm of distributed computing that can replace the conventional client-server model. However, it has not become popular due to some problems such as security. The fact that computers have complete control over all the programs makes it very hard to protect mobile agents from untrusted hosts. In this paper we propose a security approach for mobile agents, which protects mobile agents from malicious hosts. Our new approach prevents privacy attacks and integrity attacks to mobile agents from malicious hosts. This approach is an extension of mobile cryptography, as proposed by Sander and Tschudin, and it removes many problems found in the original idea of mobile cryptography while preserving most of the benefits. Although the original idea of mobile cryptography allowed direct computations without decryptions on encrypted mobile agents, it did not provide any practical ways of implementation due to the fact that no homomorphic encryption schemes are found for their approach. Our approach provides a practical idea for implementing mobile cryptography by suggesting a hybrid method that mixes a function composition technique and a homomorphic encryption scheme that we have found. Like the original mobile cryptography, our approach will encrypt both code and data including state information in a way that enables direct computation on encrypted data without decryption.

1 Introduction

An agent-based computer system is a distributed computing environment in which mobile autonomous processes called mobile agents operate on behalf of users. The autonomous agent concept has been proposed for a variety of applications on large, heterogeneous, distributed systems (e.g., the Internet) [9]. These applications include a specialized search of a large free-text database [7], middleware services such as an active mail system, electronic malls for shopping, and updated networking devices. Mobile agent systems are purported to have many advantages over traditional distributed computing environments. They require less network bandwidth, increase asynchrony among clients and servers, dynamically update server interfaces and introduce concurrency [8].

Mobile agents have existed for some time, but problems with security have limited their popularity. Mobile agents are composed of code, data, and state. Agents migrate from one host to another taking the code, data and state with them. The state information allows the agent to continue execution from the point where it was before it left in the previous host. For example, a mobile agent could be dispatched from the home site with the task of buying an airplane ticket for its owner. The agent would visit all the known hosts of airline companies, one after another, to search for the most reasonably priced ticket, and then purchase one for its owner. Each time the agent hops to the next host, it summarizes the current state, execution pointer on the current state, etc., so that it can start searching for reasonable tickets on the next host. The state of the agent will contain a set of possible tickets to be considered for purchase. When the agent has finished its search, it may return to the host where it found the cheapest or best ticket and purchase it.

While agents roam around the Internet, they are exposed to many threats and may also be a source of threat to others. Sander and Tschudin present two types of security problems that must be solved [16]. The first is host protection against hostile agents. The second is agent protection against hostile hosts. Many techniques have been developed for the first kind of problem, such as access control, password protections, and sand boxes, but the second problem appears to be difficult to solve. It is generally believed that the ex-

*This work was partially supported by DARPA contract MDA972-00-1-0001 and Air Force Research Laboratory contract F30602-02-1-0178.

†This work was conducted while this author was a PhD student at the University of Idaho.

0-7695-2056-1/04 $17.00 (C) 2004 IEEE

ecution environment (host) has full control over executing programs; thus, protecting a mobile agent from malicious hosts is difficult to achieve unless some tamper-proof hardware is used. For example, Yee proposed an approach that uses a secure coprocessor that executes critical computations and stores critical information in secure registers [18]. In this paper, we propose a security approach to protect mobile agents from untrusted hosts.

In this paper, we focus on extending the mobile cryptography approach, proposed by Sander and Tschudin [14, 16, 15], in terms of privacy and integrity, and explore its usefulness and effectiveness in protecting mobile agents (We discuss the mobile cryptography in Section 2). To extend mobile cryptography, we will consider composite functions and additive-multiplicative homomorphism to encrypt mobile agents. As the contribution of this research, the encrypted mobile agent will be able to run on any host without decryption. The encrypted mobile agent will generate encrypted results, which will be decrypted by the agent owner. This will improve the overall security of the mobile agents.

In the remainder of this paper, we expand the idea of mobile cryptography. In Section 2 we provide an overview of some related works. In Section 3 we introduce the idea of homomorphic encryption scheme and function composition. In Section 4 our new approach for mobile agent security is discussed. In Section 5 we discuss the details of mobile cryptography, followed by a detailed discussion of our cryptosystem to implement mobile cryptography in Section 6. In Sections 7 we briefly give conclusion and some future works for our approach.

2 Related Work

Mobile agent protection is difficult due to a host's complete control over executing programs. While many approaches have been proposed to defend mobile agents from untrusted hosts, none adequately addresses every aspect of security. We survey four proposed approaches for the problem of mobile agent protection. The four approaches are chosen because each approach is very uniquely implemented and has strengths that other approaches do not have; we choose social control approach because it mimics our real society where badly behaved merchants are forced to go out of business. Partial result authentication code approach is chosen due to the fact that it can protect results from mobile agents. Environmental key generation approaches are chosen because it uses cryptographic support for privacy attack prevention. Mobile cryptography approaches is chosen because it tries to scramble code and data together.

2.1 Partial Result Authentication Codes (PRAC)

Yee proposed an approach, Partial Result Authentication Codes (PRAC), which protects partial result with a Message Authentication Code (MAC) computed on partial results by using a secret key [19]. The agent originator (owner) and mobile agent are given a secret key for each host to be visited. The current secret key used to encrypt the partial result is destroyed before the agent migrates to the next host. Destroying secret keys before agent migration ensures that the previous partial results are secure and intact. Since the agent originator maintains the secret keys, the partial results can be verified on the originator's home site.

PRACs also provide a reasonable protection to mobile agent systems by focusing primarily on integrity issues in the mobile agent system. PRACs have several positive aspects. First, PRACs improve the integrity of partial results, because the secret key(s) used to create PRACs are destroyed before an agent's migration. Second, unless secret keys are compromised, the agent originator can pinpoint which malicious host attempted an attack through comparing the PRAC generated by the malicious host to the PRAC generated by the correct secret key stored in the agent originator. The third advantage of the PRAC approach is that it guarantees forward integrity, which states that even though the current host is malicious, all the previous partial results are safe, because the secret key for each previous host is destroyed before an agent's migration.

There are negative aspects to the PRAC approach also. First, it provides protection to partial results for mobile agents, but not to agent code and other aspects of the agent. Second, if the secret keys are compromised by malicious hosts, then those malicious hosts can read and modify any partial results. Third, although secret keys are destroyed before agent migration, it does not ensure that future results are secure, if a host is ever revisited by an agent.

2.2 Environmental Key Generation

The next approach, Environmental Key Generation proposed by Riordan and Schneier, generates the decryption key for an agent's encrypted code and data by searching through the execution environment [12]. The agent originator sends a cipher-text message (i.e., encrypted data and instructions) and a method for searching the environment for the data that is required to generate the decryption key. If the proper environmental data is found through the given data channel, then the key is generated to decrypt the encrypted mobile agent.

Environmental key generation has many strengths over other approaches. First, environmental key generation improves the integrity and privacy for agent code and data, which are both encrypted by the agent. Second, the decryp-

tion key is kept secure. The programmer can choose any kind of data channel that best suits the application such as a file system, Internet newsgroup, or e-mail. Even though the attacker may know which data channel the agent is searching, he or she must know which data portion of the data channel is required for the key generation.

The environmental key generation can protect the code and data from integrity and privacy attacks, but this approach also has weaknesses. First, the environmental key generation approach is vulnerable to group conspiracy attack. Second, data channel protection is another security issue. Third, although this approach can improve the integrity and the privacy for its code and data, it does not provide any protection for results. Fourth, once the code and data are decrypted, they can be attacked by a malicious host who can insert his or her own decrypting routine and data channel for new hosts.

3 Evaluating Encrypted Functions

Our approach is built on the bases of three-address code, homomorphic encryption scheme (HES), and function composition (FnC) technique. In this section, we describe three-address code, function composition (FnC) and homomorphic encryption scheme (HES) to prepare for our new approach.

3.1 Three-Address code

Many computer languages use compilers to translate source code into executable target code. Compilers go through several phases; after the lexical, syntax and semantic analysis, some compilers, though not all, generate an explicit intermediate representation, before generating target code [1]. The three-address code is one of the forms of intermediate representations.

Three-address code is a sequence of the statements of the form $x := y\ op\ z$, where x, y, and z are names, constants or compiler-generated temporaries; *op* stands for any operator, such as a fixed- or floating-point arithmetic operator, or a logical operator on boolean-valued data. Thus a source language expression like $x + y * z$ might be translated into a sequence:

$$t_1 := y * z$$
$$t_2 := x + t_1$$

where t_1 and t_2 are compiler generated temporary names [1]. In general, three-address code contains three addresses, where there are two addresses for the operands and one for the result.

3.2 Homomorphic Encryption Scheme (HES)

Rivest, Adleman and Dertouzos pointed out that the limitation of an encryption system is that an information system can only store and retrieve encrypted data for users. Further operations on data require decryption, and once the data is decrypted, it is not secure any more. Thus, the researchers proposed a new idea of cryptosystem that enables direct computation on encrypted data without decryption, which they called *privacy homomorphism* [13]. Later, Sander and Tschudin defined additive-multiplicative homomorphism, which is a kind of privacy homomorphism [16, 15]. Additive-Multiplicative homomorphism ensures that the computation result on two encrypted values is exactly the same as the encrypted result of the same computation on two unencrypted values. Sander and Tschudin's mobile cryptography uses HES for its implementation, but there are some drawbacks. First, no single cryptosystem is found to be additively, multiplicatively and mixed multiplicatively homomorphic. Second, only some limited classes of functions (polynomial and rational functions) are proved to be compatible with the HES [16, 15]. Here, we describe the properties of homomorphic encryption scheme that we need for securing mobile agents from the work of Sander and Tschudin [16, 15]:

> Let R and S be rings. We call an (encryption) function $E : R \rightarrow S$
>
> - *additively homomorphic* if there is an efficient algorithm **PLUS** to compute $E(x+y)$ from $E(x)$ and $E(y)$ that does not reveal x and y,
> - *multiplicatively homomorphic* if there is an efficient algorithm **MULT** to compute $E(xy)$ from $E(x)$ and $E(y)$ that does not reveal x and y,
> - *mixed-multiplicatively homomorphic* if there is an efficient algorithm **MIXED-MULT** to compute $E(xy)$ from $E(x)$ and y that does not reveal x.

The homomorphic encryption scheme that meets the three properties allow only two types of operators: addition and multiplication. One thing to note is that there is one-to-many relationship, which implies that a single plaintext message, x, can have multiple ciphertext messages of $E(x)$ (i.e., although $E_1(x) \neq E_2(x)$, $D(E_1(x)) = D(E_2(x))$ is true for a plaintext message x). Another point to note is that there should be only a few elements (only one element is desirable) that satisfies the last property (mixed-multiplicativity), otherwise the last property and the second property yield an anomaly, $y = E(y)$. Thus, in integers, only one integer (a multiplicative identity, $x = 1$)

should satisfy the last property, $E(xy) = E(x)y$, to avoid the anomaly.

3.3 Function Composition(FnC)

Sander and Tschudin argue that evaluating encrypted functions (EEF) can be accomplished, not only by an additive and multiplicative homomorphism, but also by mathematical analogues such as composite functions [14]:

> Assume Alice wants to evaluate a linear map A at Bob's input x on Bob's computer. She does not want to reveal A to Bob, so she picks at random an invertible matrix S, computes $B := SA$ and sends B to Bob. Bob computes $y := Bx$ and sends y back to Alice. Alice computes $S^{-1}y$ and obtains the result Ax without having disclosed A to Bob.

We define $f(x)$ as a resultant composite function, if it is derived by taking the output of a function, $h(x)$, and using as the input to another function, $g(x)$. Mathematically, this is represented by $f(x) = g \circ h$ or $f(x) = g(h(x))$, where the function, $h(x)$, is a hidden (original) function. The function (agent) owner must choose an invertible function, $g(x)$, to create a composite function $f(x)$. The function, $f(x)$, is a different function (encrypted functions) from $h(x)$; thus, privacy and integrity requirements are preserved. The result of this composite function, $f(x)$, is also encrypted; malicious hosts do not know the result. The function (agent) owner retrieves the result from the encrypted result by using the inverse function of $g(x)$.

In Figure 1, Alice is the agent (function) owner and has a function, $h(x)$, that she wants to evaluate on Bob's computer with Bob's input x, but she does not want to reveal anything about her function. Alice chooses an invertible function, $g(x)$, creates a composite function, $f(x)$, and sends it to Bob. Bob does the computation with his input, x, and sends the result back to Alice. Bob cannot determine the function, $h(x)$, because what he sees is only the composite function $f(x)$. Only Alice can retrieve the real result of $h(x)$ from the result of $f(x)$ by plugging $f(x)$ into the inverse function of $g(x)$(i.e., $h(x) = g^{-1}(f(x))$).

4 New Approach

Ours is a hybrid approach which combines HES and FnC. A special program called *Mobile Agent Encryption (MAE)* will intercept the three-address code from compilers, and apply HES to encrypt the operands of three-address code and FnC to encrypt codes. In other words, *MAE* will encrypt the sensitive data, such as credit card number and personal information, stored in the operands of three-address code, and scramble the code of the mobile agent to confuse untrusted hosts. Our approach inherits most of the strengths of mobile cryptography; ours encrypt mobile agents, and the encrypted mobile agents are executable without decryption. The partial result is also protected by our approach. Furthermore, our approach removes some critical problems found in the original idea of mobile cryptography by Sander and Tschudin (this is discussed in analysis section). Implementing our approach requires assumptions and there are some limitation stemming from the assumption. Before, we present our approach, we state the assumptions and the goals of our approach.

4.1 Goals of Our Approach

The first goal of our approach is to enhance the privacy so that malicious hosts are not be able to read the contents of important data. The next goal is to enhance the integrity of the agent. Generally, hosts running programs have complete control over the programs; thus malicious hosts can read the mobile agent's code, analyze the flow of control, and modify the mobile agent. Because this will disrupt the normal execution of the mobile agent, the integrity of the result generated by the agent cannot be guaranteed. Another goal is encrypting the mobile agent carefully using an additive-multiplicative homomorphic encryption scheme so that the encrypted mobile agent is executable without decryption. Another goal is to protect the result generated from the encrypted mobile agent. The results also suffer from the same security problems as the mobile agent. Without the protection of the result, malicious hosts can read and modify the result for their own benefits. The last goal is that no one except the agent owner must be able to decrypt the agent and result.

4.2 Assumptions for New Approach

Our approach offers broader protection to mobile agents than other approaches reviewed in Section 2. Implementing our approach requires some assumptions like the original mobile cryptography. The first assumption we need is that the HES is based on ring theory; thus we assume that the transformation (encryption/decryption) of elements from one set into the other set is additively and multiplicatively homomorphic. The second assumption is that we use only integers, due to the the fact our HES is based on ring theory. The third assumption is that only addition and multiplication are used in the agent code. Again, this is because we are using an additive-multiplicative homomorphic encryption scheme. The fourth assumption is that the control structures of the agent code will not be encrypted by the composite function technique, because the control structures such as `if`-statements have logical expressions with other types of operators such as logical operators, boolean

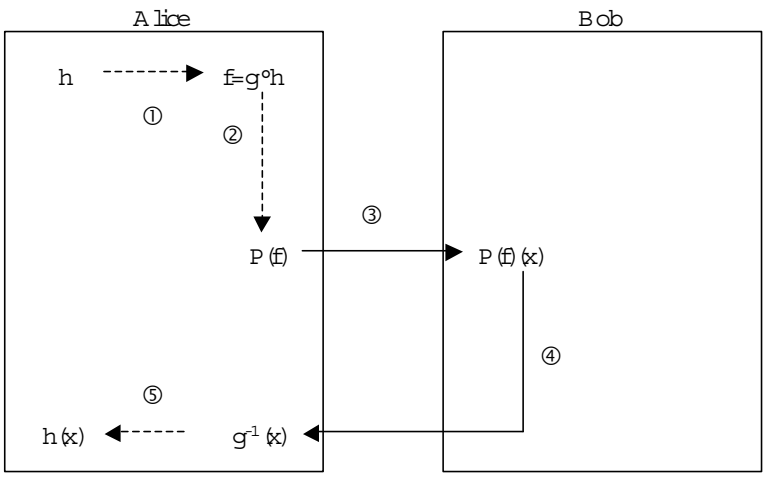

Figure 1. Composite Function

operator and equality operators.

4.3 Encryption

Our new approach extends Sander and Tschudin's idea of mobile cryptography and overcomes the problems of their approach by proposing a practical way of implementing mobile cryptography. The new approach encrypts the data from the agent, then encrypts the three-address code representation of the agent by using the composite function technique. This implies that the mobile agent is doubly encrypted, by first encrypting the data and state information found in the three-address code with the additive-multiplicative homomorphic encryption scheme, and then by encrypting the three-address code (i.e., the code portion of the mobile agent) with the composite function technique. The overall workflow of the approach is a multi-step process as below:

- **Step 1** The operands of the three-address code are encrypted by using an additive-multiplicative homomorphic encryption scheme.
- **Step 2** The operand dependency problem (data encrypted by HES should not be encrypted again) caused by the additive-multiplicative homomorphic encryption scheme is removed.
- **Step 3** Three-address code statement is encrypted by using the function composition technique.
- **Step 4** The three-address code dependency problem caused by the function composition technique is resolved.

During encryption *MAE* grabs the plaintext three-address code from the compiler and analyzes the code for the encryption of sensitive data and state information of the mobile agent by using HES. After the first encryption, *MAE* encrypts the three-address code by using the function composition technique. While performing the first and second encryptions, *MAE* encounters the double encryption problem for the operands and the codes (statements). This will lead to the incorrect encryption of the mobile agents, thus *MAE* carefully looks for all the operands and statements for the double encryption, removing the double encryption, if any. The double encryption problem for each component (HES and FnC) is addressed in [10]. *MAE* will generate the encrypted three-address code, which performs the same task as the plaintext three-address code, and makes it difficult for malicious hosts to read and modify the mobile agent code, data and state information.

4.4 Decryption

The process of decryption is exactly the reverse of the encryption process. The agent owner does not need to decrypt the whole mobile agent; instead only the encrypted result is decrypted. The result of the computation of any encrypted mobile agent is automatically encrypted, and malicious hosts cannot read and understand the encrypted result. The decryption done by the agent owner's *MAE* is a two-pass process, in which the first pass will use the function composition technique to decrypt the result and the second pass will use the additive-multiplicative homomorphic encryption scheme to fully recover the actual result. The first pass will simply use the inverse of the function used to create the composite function during the encryption. The

second pass uses the secret decryption keys of the additive-multiplicative homomorphic encryption scheme to obtain the actual result.

4.5 Overall Idea

Figure 2 depicts the overall process of encryption and decryption. During the encryption, HES is used to encrypt data, and a simple secret function, $g(x) = x^3 + 1$, is used for function composition. The encrypted mobile agent is released on the Internet, and returns with completed task (result) to the agent owner. In the decryption the inverse function, $g^{-1}(x) = \sqrt[3]{x+1}$, is used first, and HES is used to retrieve the actual result.

5 Details Behind Mobile Cryptography

Two decades after the discovery of privacy homomorphisms, only a few privacy homomorphic encryption schemes exist today (see [2, 3, 4, 5, 6, 11]). Rivest, Adleman and Dertouzos originally proposed the idea of privacy homomorphism [13], in which direct computation on encrypted data is possible without any decryption, while normal cryptosystems require the encrypted data be decrypted first. In normal cryptosystems, once the data is decrypted for some computation, its content is vulnerable to misuse by malicious users, software or hosts.

As mentioned previously, Sander and Tschudin proposed the idea of mobile cryptography (EEF-Evaluating Encrypted Functions) [16]. They argued that some classes of functions can be encrypted via an additive-multiplicative homomorphic encryption scheme. The encrypted function can then be transmitted to a remote host where the function can be computed without prior decryption [16]. Considering the fact that an additive-multiplicative homomorphism allows addition and multiplication on encrypted data without decryption, it is a form of privacy homomorphism. Sander and Tschudin also proposed the idea of mixed-multiplicative homomorphism [16], in which there exists an efficient algorithm MIXED-MULT such that we can computer $E(xy)$ from $E(x)$ and y without revealing x, where E is the encryption algorithm. This implies that if $x = 1$, then we can efficiently calculate $E(xy) = E(y)$; from $E(1)$ and y. Therefore a remote host can encrypt the value y without seeing the encryption key, only the efficient function MIXED-MULT. Of course such a simple use of this type of encryption is vulnerable to chosen plaintext attacks, since an eavesdropper can intercept the original $E(1)$ and try a large number of y values to find a chosen $E(y)$. Expanding the form of the encrypted field to $y * 2^k + r$, where r is a large k-bit random number will reduce the effectiveness of such an attack; but may affect the homomorphic properties of the crypto system.

5.1 Privacy Homomorphism and Additive-Multiplicative Homomorphism

An additive-multiplicative homomorphism is a subset of privacy homomorphisms. Encryption using an additive-multiplicative homomorphism preserves addition and multiplication operators, such that $E(x + y)$ and $E(xy)$ can be efficiently computed from $E(x)$ and $E(y)$; while encryption with a privacy homomorphism can preserve additional operators.

The concept behind privacy homomorphism is to improve security by allowing direct computation on encrypted data without decryption. Rivest, Adleman and Dertouzos define privacy homomorphism as follows [13].

> We define U as denoting the plaintext objects with data types S, operators f_i, predicates p_i and distinguished constants s_i and similarly for C denoting the ciphertext objects where f'_i is the *encrypted* version of f_i. The decryption function is $\phi : S' \rightarrow S$, while the encryption function is $\phi^{-1} : S \rightarrow S'$.
>
> **U:** $< S; f_1, \cdots, f_k; p_1, \cdots, p_l; s_1, \cdots, s_m >$
> **C:** $< S'; f'_1, \cdots, f'_k; p'_1, \cdots, p'_l; s'_1, \cdots, s'_m >$

According to Rivest, Adleman and Dertouzos, the operation of privacy homomorphism is as follows [13]:

> To make the direct computation on the encrypted data, the decrypting function ϕ should be homomorphic from C to U, which means that
>
> $$\forall i(a, b, c, \ldots).\ [f'_i(a, b, \cdots) = c \Rightarrow f_i(\phi(a), \phi(b), \cdots) = \phi(c))],$$
> $$\forall i(a, b, c, \ldots).\quad p'(a, b, \cdots) \equiv p(\phi(a), \phi(b), \cdots),$$
> and
> $$\phi(s'_i) = s_i$$
>
> If the user wants to know the value of $f_1(d_1, d_2)$, the user asks the computer to compute $f'_1(\phi^{-1}(d_1), \phi^{-1}(d_2))$. Since ϕ is a homomorphism,
>
> $$\phi(f'_1(\phi^{-1}(d_1), \phi^{-1}(d_2))) = f_1(d_1, d_2)$$
>
> so that the computer can generate the encrypted form of the answer [13].

5.1.1 Additive-Multiplicative Homomorphism

Additive-multiplicative homomorphisms are a subset of the privacy homomorphism, and they are defined formally by Sander and Tschudin as follows [16]:

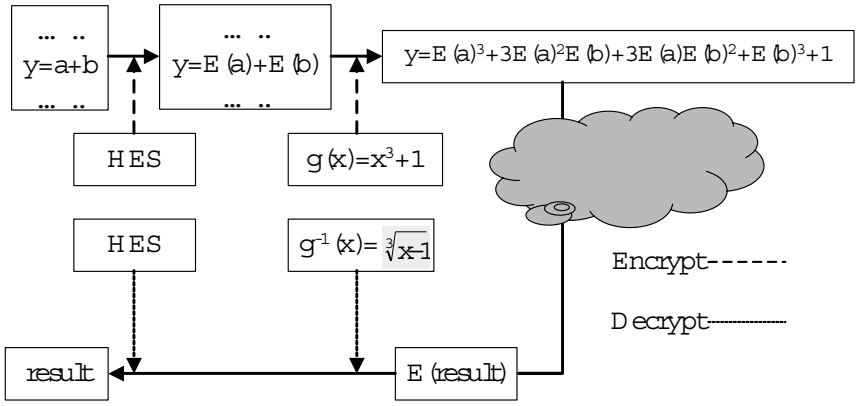

Figure 2. Overall Scheme

Let R and S be rings. We call an (encryption) function $E : R \rightarrow S$

- additively homomorphic if there is an efficient algorithm PLUS to compute $E(x + y)$ from $E(x)$ and $E(y)$ that does not reveal x and y,
- multiplicatively homomorphic if there is an efficient algorithm MULT to compute $E(xy)$ from $E(x)$ and $E(y)$ that does not reveal x and y.

The additive homomorphism and multiplicative homomorphism preserve the addition and multiplication, respectively. Both, privacy homomorphism and additive-multiplicative homomorphism, allow secure computing on encrypted data without decryption.

5.2 Mixed-Multiplicative Homomorphisms

Both privacy homomorphisms and additive-multiplicative homomorphisms are very useful due to their unique property that enable direct computation on the encrypted data. Another very useful homomorphism is a mixed-multiplicative homomorphism, which was first proposed by Sander and Tschudin [15, 16, 17]:

Let R and S be rings and E be a function (encryption) between them: $E : R \rightarrow S$. E is called *mixed-multiplicative homomorphic* if:

- there is an efficient algorithm MIXED-MULT to compute $E(xy)$ from $E(x)$ and y that does not reveal x.

The definition implies that we can encrypt a plaintext, y, without any knowledge of the cryptosystem by simply using the efficient function MIXED-MULT on y and $E(1)$.

5.2.1 Meaning and Applications

As briefly stated, a mixed-multiplicative homomorphism allows encryption of a plaintext message (value) without any knowledge of the cryptosystem including the keys and encryption algorithm. To encrypt a plaintext message, y, users need only a single encrypted message, $E(1)$; this is a large benefit of this approach. An advantage of this approach is that the encryption can be done in real-time, because the encryption of plaintext data, y, requires only a single invocation of MIXED-MULT. An advantage of this approach is that it improves security by avoiding the decryption of encrypted data to generate a new encrypted value from new data and a previously encrypted value. This permits us to transmit the original data, in encrypted form, to remote hosts that can perform the necessary computation, while maintaining the privacy of not only the encrypted data, as in privacy homomorphisms, but the privacy of the keys as well.

There are many possible applications for a mixed-multiplicative homomorphic encryption scheme. One possible application is multi-party computation, where each participant does not want to reveal its data to the other participants. A mixed-multiplicative homomorphic encryption scheme will allow each participant to encrypt inputs to a program, and perform the direct computation on the encrypted data.

Another application can be found in mobile cryptography. Sander and Tschudin first proposed the idea of mobile cryptography, in which a function is encrypted by using an additive, multiplicative and mixed-multiplicative homomorphic encryption scheme to create an executable encrypted

function [17]. The result generated from the encrypted function is already encrypted, and requires decryption to retrieve the actual result. Sander and Tschudin argued that there is no known homomorphic encryption scheme that is additively, multiplicatively and mixed-multiplicatively homomorphic to encrypt functions [15, 16, 17]. The closest encryption scheme to implement Sander and Tschudin's mobile cryptography was Domingo-Ferrer's cryptosystem, which was additive and multiplicative, but not mixed-multiplicative [5, 16].

The last example application for a mixed-multiplicative homomorphism is an electronic voting scheme, where the vote is electronically cast, encrypted and tallied. Each voter receives ballots electronically signed by some official, and verifies if the ballot is the right one. If the ballot is valid, the voter can use mixed-multiplicative homomorphism to encrypt the decision 1, if the corresponding choice on the ballot would have been punched; otherwise the decision 0 will be encrypted. In practice the voter will submit a random even number, r, to which the vote will be added. This prevents eavesdroppers from using a chosen plaintext attack to determine the encrypted value. Although the voter does not need to know the encryption algorithm and keys, the voter can multiply the transmitted ballot by $r + v$, where v represents the choice. The voter then digitally signs the ballot, to protect against fraud and transmits it to the electronic ballot box. The balloting agency verifies the signature and then discards it before decrypting the vote.

6 MMH Cryptosystem

In this section, we present our simple mixed-multiplicatively homomorphic cryptosystem, MMH. We believe this is the first such published algorithm, and although it has not undergone extensive cryptoanalysis, we believe it to be strong against ciphertext only attacks.

Our system is a modified version of the cryptosystem originally proposed by Domingo-Ferrer and Herrera-Joancomartí, who argue that their cryptosystem is only additively and multiplicatively homomorphic [6]. As noted earlier, Sander and Tschudin proposed the idea of mixed-multiplicative homomorphism based on ring theory, which is essential to fully implement mobile cryptography, and insisted that there is no published cryptosystem that is additively, multiplicatively and mixed-multiplicatively homomorphic based on ring theory. In this section, we discuss the simple modified version of Domingo-Ferrer and Herrera-Joancomartí's cryptosystem; with the biggest possible application, mobile cryptography, in mind.

6.1 Our New Cryptosystem (MMH)

The modified cryptosystem is similar to the preceding cryptosystem, except that it is easier to implement and just as secure. The modified version uses a large number, n, such that $n = p \times q$, where p and q are large prime numbers. Let $Z_p = \{x \mid x \leq p\}$ be the set of original plaintext messages, $Z_n = \{x | x < n\}$ be the set of ciphertext message and $Q_p = \{a \mid a \notin Z_p\}$ be a set of encryption *clues*. The types of operations defined are addition and multiplication on Z_p. The encryption and decryption algorithms are as follows:

Encryption Given $x \in Z_p$, pick a random number a in Q_p such that $x = a \bmod p$. Compute the encrypted value $y = E_p(x) = a \bmod n$. (This can be accomplished by picking a random r and creating $a = x + rp$.)

Decryption Given $y = E_p(x) \in Z_n$, use the key p to recover $x = D_p(y) = y \bmod p$.

The modified cryptosystem is additively, multiplicatively, and mixed-multiplicatively homomorphic. Also, our cryptosystem encrypts one plaintext message, x, into many ciphertext messages. Thus, even though $E_1(x) \neq E_2(x)$, $D(E_1(x)) = D(E_2(x))$. It is easy to prove that the modified cryptosystem works correctly in an approach similar to Domingo-Ferrer and Herrera-Joancomartí's original proof [6].

Theorem 1 (Correctness). *For all $x \in Z_p$, $D_p(E_p(x)) = x$ holds true.*

Proof. Let $y = E_p(x)$ and a be the random number used to encrypt the message. Then it is true that

$$a \bmod n = y \tag{1}$$

Since p divides n, equation 1 implies that

$$y \bmod p = (a \bmod n) \bmod p = x \tag{2}$$

□

The MMH cryptosystem can be used as in the following example:

Example (Multiplication) Let $p = 11$, $q = 7$, $n = 77 = p \times q$ and the values, $x1 = 5$ where $E(5) = 38$. and $x2 = 2$ where $E(2) = 13$.

$$(38 \times 13) \bmod 77 = 32$$

Decrypting 32 yields,

$$10 = 32 \bmod 11$$

The above example shows how to perform multiplication on encrypted values. Addition in accomplished in a similar manner. As in other privacy homomorphic encryptions with modulo n, our cryptosystem performs arithmetic modulo n, which is not shown in our examples to simplify the presentation.

The modified version is mixed-multiplicatively homomorphic as shown in the following:

Theorem 2 (Mixed-Multiplicativity). *For all s and t in Z_p, $D(E(s)t) = D(E(st))$.*

Proof. We will first evaluate the terms $E(s)t$ and $E(st)$

- $E(s)t$: To encrypt s we first choose an a_1 such that $s = a_1 \bmod p$, in otherwords we know that $a_1 = k_1 p + s$

  Encrypting, we get $y_1 = a_1 \bmod n$ which tells us that $a_1 = k_2 n + y_1$, therefore $k_1 p + s = k_2 n + y_1$. Solving for y_1 we get

  $$y_1 = k_1 p - k_2 n + s = (k_1 - k_2 q)p + s \qquad (3)$$

  Since $E(s) = y_1$, and $E(s)t = y_1 t$ we have:

  $$y_1 t = (tk_1 - tk_2 q)p + ts.$$

  Decryption will then give us:

  $$D(E(s)t) = D(y_1 t) = y_1 t \bmod p = st. \qquad (4)$$

- $E(st)$: To encrypt st we first choose an a_2 such that $st = a_2 \bmod p$ where $a_2 = k_3 p + st$. Using this we encrypt st as $y_2 = a_2 \bmod n$ which implies that $a_2 = k_4 n + y_2$. Solving for y_2 we get $y_2 = (k_3 - k_4 q)p + st$.

  Decrypting will then give us:

  $$D(E(st)) = D(y_2) y_2 \bmod p = st. \qquad (5)$$

- $D(E(s)t) = D(E(t))$: From equations 4 and 5,

  $$y_1 t \bmod p = st = y_2 \bmod p$$

  This implies that $D(E(s)t) = D(E(t))$ using the modulus p.

□

The properties of additivity and multiplicativity can be proven in a similar manner.

6.2 Automatic Encryption of Remote Inputs

By the definition of the mixed-multiplicative homomorphism, the computing partner's input, t, will be automatically encrypted by multiplying t by $E(1)$ (i.e., $E(1) \times t$). The following example demonstrates this property of mixed-multiplicative homomorphism of the modified cryptosystem:

Example Assume $p = 101$, $q = 71$, and $n = pq = 7171$. Also assume the agent owner provides $E(1) = 203$. The malicious host wishes to encrypt the input, 8. Then, the malicious host multiplies $E(1)$ by 8, which yields the ciphertext, $E(8) = 1624$. To verify this, choose $A = 15966$, $15966 \bmod 7171 = 1624$ (Remember $A \in Q_n = \{A |\ (A \notin Z_n) \cap (A \geq n)\}$). Again $1624 \bmod 101 = 8$.
□

6.3 Security of MH

Domingo-Ferrer and Herrera-Joancomartí discussed the security of their cryptosystem in [6]. Recall that encryption of a value x in their system involved finding an a and b such that $x = ab^{-1} \bmod p$ and then creating the encrypted value $y = ab^{-1} mod n$. For decryption of y, you need to choose values A and B such that $y = AB^{-1} \bmod n$ and then calculate $AB^{-1} \bmod p$. However, an adversary does not have to work this hard. Note that if $y = AB^{-1} \bmod n$ then $AB^{-1} = mn + y = mpq + y$, and therefore $AB^{-1} \bmod p = y \bmod p$. As with our approach, we can decrypt by calculating $x = y \bmod p$. As for encryption,

Since our modified version is a simplified version and still shares all the properties with the original version, we can use their arguments on the security for our modified cryptosystem with small modifications.

- *Ciphertext-Only Attack.* The cryptanalyst does not need p to find a number $A \in Q_n$ corresponding to a ciphertext $y \in Z_q$. However, p is needed to compute $a \bmod p = x$. But, if the cryptanalyst sees only ciphertext, then finding the secret p from the public n is as difficult as factoring n [6]. Therefore, with only ciphertext, finding the original value is difficult.

- *Known-Plaintext Attack.* If the cryptanalyst knows a plaintext-ciphertext pair (x, y), then the cryptanalyst can generate a set of t numbers, $A_i \in Q_n$ for $i = 1, \cdots, t$ such that $A_i = y \bmod n$. Then, the cryptanalyst knows that $A_i = x \bmod p$ for each i, so that $p \mid (A_i - x)$. With high probability $p = \gcd_{i=1}^{t}(A_i - x)$.

- *Integrity Attack.* Since all of the decryption is performed modulo p, any unencrypted number $x < p$ will b e deciphered as itself. Therefore an adversary can replace any encrypted value with a chose value and claim it is encrypted.

7 Conclusion

Mixed-multiplicative homomorphism provides a novel way of encrypting data without using any keys and encryption algorithms. We emphasized the importance of

this property in this paper. It can be used in many useful applications including multi-party computation, electronic voting, and mobile cryptography [16]. Sander and Tschudin argue that they have found no published additive, multiplicative and mixed-multiplicative homomorphic encryption schemes to implement their mobile mobile cryptography (i.e., evaluating encrypted functions). It is true that there is no sophisticated cryptosystem that is additively, multiplicatively and mixed-multiplicatively homomorphic. However, we have found a simple cryptosystem that is additively, multiplicatively and mixed-multiplicatively homomorphic and modified it so that it can be employed in mobile cryptography. The modified cryptosystem example is very simple and suggestive, although it may not be very practical. More research is required to find more examples or usable cryptosystems based on the example given in this paper.

In this paper we proposed a hybrid approach, which is a combination of HES and FnC, and argued that ours can provide broader range of protection to mobile agents. However, there are some limitations and assumptions in our approach, which restrict the application of our approach and require further study. We discuss some of the future works that must be given considerations for the improvement of our approach as follows:

- Our modified encryption scheme is additive, multiplicative and mixed-multiplicative homomorphic encryption scheme. It is a simple cryptosystem, and requires extra work to develop more sophisticated encryption schemes with complete security analysis.
- Due to the assumption of ring theory, the possible operators are restricted to addition and multiplication only. If we can move up to field, we can add two more operators such as subtraction and division.
- The number sets that we are dealing with in our approach is integers, because of the assumption of ring theory. The number sets should be extended to some other types of numbering systems such as real numbers.
- The types of function calls within an encrypted function or mobile agent is limited to some primitive ones such as basic input and output. More study is required to find a way of calling user-defined and system functions within an encrypted mobile agent.

References

[1] A. Aho, R. Sethi, and J. Ullman. *Compilers, Principles, Techniques, and Tools*, pages 462–512. Addison-Wesley, 1988.

[2] J. Benaloh. Dense probabilistic encryption. In *Proc. 26th ACM Symposium on Theory of Computing*, pages 120–128, May 1994.

[3] E. Brickell and Y. Yacobi. On privacy homomorphisms. In *Advances in Cryptology—EUROCRYPT '87*, pages 117–126, 1987.

[4] D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen. Paillier's cryptosystem revisited. In *Proc. 8th ACM conference on Computer and Communications Security*, pages 206–214. ACM Press, 2001.

[5] J. Domingo-Ferrer. A new privacy homomorphism and applications. *Information Processing Letters*, 60:277–282, 1996.

[6] J. Domingo-Ferrer and J. Herrera-Joancomatí. A privacy homomorphism allowing field operations on encrypted data. *Jornades de Matematica Discreta i Algorsmica*, 1998.

[7] J. Guttman and V. Swarup. Authentication for mobile agents. In *LNCS*, pages 114–136. Springer, 1998.

[8] N. Karnik. *Security in Mobile Agent Systems*. PhD thesis, Department of Computer Science and Engineering. University of Minnesota, 1998.

[9] C. Krintz. Security in agent-based computing environments using existing tools. Technical report, University of California, San Diego, 1998.

[10] H. Lee. *Mobile Agent: Evaluating Encrypted Functions*. PhD thesis, Department of Computer Science, University of Idaho, August 2002.

[11] D. Naccache and J. Stern. A new public-key cryptosystem. In *Theory and Application of Cryptographic Techniques*, pages 27–36, 1997.

[12] J. Riordan and B. Schneier. Environmental key generation towards clueless agents. In *LNCS*, pages 15–24. Springer, 1998.

[13] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–178. Academic Press, 1978.

[14] T. Sander and C. Tschudin. Towards mobile cryptography. Technical report, International Computer Science Institute, Berkeley, 1997.

[15] T. Sander and C. Tschudin. On software protection via function hiding. In *Information Hiding*, pages 111–123, 1998.

[16] T. Sander and C. Tschudin. Protecting Mobile Agents Against Malicious Hosts. In G. Vigna, editor, *Mobile Agent Security*, pages 44–60. Springer-Verlag: Heidelberg, Germany, 1998.

[17] T. Sander and C. Tschudin. Towards mobile cryptography. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1998. IEEE Computer Society Press.

[18] B. Yee. *Using Secure Coprocessors*. PhD thesis, Carnegie Mellon University, 1994.

[19] B. Yee. A sanctuary for mobile agents. DARPA Workshop on Foundations for Secure Mobile Code Workshop, March 1997.