Program Obfuscation and Encrypted Computation

Final project: CS 276

Marco Barreno
Computer Science Division
UC Berkeley

May 2004

1 Introduction

For my study project I investigate encrypted computation, or program obfuscation. This is an interesting concept for its own sake, but also because a successful scheme would allow a program running on an untrustworthy system to carry and use secret cryptographic keys securely. A symmetric-key encryption scheme becomes a public-key scheme by obfuscating $E_k(\cdot)$; an obfuscated mobile software agent can carry and use a signing key.

1.1 Papers

The three papers I will be discussing are "On the (Im)possibility of Obfuscating Programs" [1] by Barak, et al., "Towards Mobile Cryptography" [5] by Sander and Tschudin, and "Secure Transactions with Mobile Agents in Hostile Environments" [3] by Kotzanikolaou, Burmester, and Chrissikopoulos.

This collection of papers is interesting because the first proves a negative result about program obfuscation but the other two claim positive results for obfuscating schemes. This apparent contradiction comes from different models of obfuscation as well as different requirements on the scope of protection; some of the differences are not immediately obvious, however.

1.2 Changes from the proposal

I had originally planned to discuss the paper "A Pragmatic Alternative to Undetachable Signatures" [2] by Borselius, Mitchell, and Wilson, but I selected that paper before reading it, and upon reading it I decided it was not interesting enough or well-written enough to include. I replaced it with the Kotzanikolaou et al. paper [3], which is similar to (and cited by) Borselius et al., but seemed better written and more interesting.

I also originally selected the paper "Protecting Mobile Agents Against Malicious Hosts" [4] by Sander and Tschudin but then later discovered that "Towards Mobile Cryptography" [5] (same authors) is essentially the same paper but reworked significantly. The latter is more polished and appeared in the Oakland conference, so I decided to focus on it; the research content of the two papers is more or less the same, though the exposition differs (little or none of the text is actually identical).

Since my original proposal for an ϵ-new contribution concerned the Borselius et al. paper, I have had to change that plan as well. The scheme proposed in Kotzanikolaou et al. is broken, and I will not only explain why it is broken but also show that the same goals can be accomplished with a simple conventional signature.

1.3 Organization

The main part of this report comprises three sections. I first will explore the two paradigms for program secrecy found in the papers: Section 2 examines the work of Barak et al., and Section 3 will focus on the papers of Kotzanikolaou et al. and Sander and Tschudin. In these sections I will summarize the contributions of the papers, presenting the key results with an emphasis on brevity. Then in Section 4 I will compare the two paradigms for obfuscation and analyze their relationship, as well as presenting my analysis of the weaknesses of the Kotzanikolaou et al. scheme.

1.4 A note about this paper

Because this is a survey paper, I naturally include a substantial amount of paraphrasing and some actual text from the papers under consideration. In particular, definitions and theorems from the papers are taken word-for-word or are very close paraphrases. I will not make an explicit citation every time this happens; it should be assumed that the definitions, statements of theorems, and the like in Sections 2 and 3 are not my own original writing, and much of the text in those sections is paraphrased from the papers. Other sections are more entirely my own work, and in them I will cite quotations and concepts normally.

2 Black-box program obfuscation

Barak et al. [1] discuss the paradigm of *black-box obfuscation*. The intuition of this model is that an adversary should gain no more information from an obfuscated program than from oracle access to the original. They define a Turing machine (TM) obfuscator[1] as:

Definition 2.1 (Barak et al.) *A probabilistic algorithm $\mathcal{O}$ is a* TM obfuscator *if the following three conditions hold:*

[1] Although Barak et al. discuss both Turing machine and circuit obfuscators, I will omit the latter for the sake of brevity.

- *(functionality) For every TM M, the string $\mathcal{O}(M)$ describes a TM that computes the same function as M.*
- *(polynomial slowdown) The description length and running time of $\mathcal{O}(M)$ are at most polynomially larger than that of M.*
- *("virtual black box" property) For any PPT A, there is a PTT S and a negligible function α such that for all TMs M*

$$\left|\Pr[A(\mathcal{O}(M)) = 1] - \Pr[S^M(1^{|M|}) = 1]\right| \leq \alpha(|M|).$$

We say that $\mathcal{O}$ is efficient if it runs in polynomial time.

Using this definition, their main result is the following simple theorem:

Theorem 1 (Barak et al.) *TM obfuscators do not exist.*

The key to this result is a family of functions $\mathcal{F}$ with a binary predicate $\pi : \mathcal{F} \to \{0,1\}$ such that for any $f \in \mathcal{F}$, $\pi(f)$ is easy to calculate given any program that computes f but cannot be found with much more accuracy than flipping a coin given only oracle access to f. To build this idea into a proof, they introduce an obfuscator that is secure when the adversary has access to more than one obfuscated program (specifically, two). The proof is easier for this obfuscator, and it will readily lead to the proof of Theorem 1.

Definition 2.2 (Barak et al.) *A* 2-TM obfuscator *is defined in the same way as a TM obfuscator, except that the "virtual black box" property is strengthened as follows:*

- *("virtual black box" property) For any PPT A, there is a PPT S and a negligible function α such that for all TMs M, N*

$$\left|\Pr[A(\mathcal{O}(M), \mathcal{O}(N)) = 1] - \Pr[S^{M,N}(1^{|M|+|N|}) = 1]\right| \leq \alpha(\min(|M|, |N|)).$$

Lemma 2 (Barak et al.) *2-TM obfuscators do not exist.*

Proof sketch. Suppose for the sake of contradiction that $\mathcal{O}$ is a 2-TM obfuscator.

Consider two Turing machines $C_{\alpha,\beta}$ and $D_{\alpha,\beta}$ defined for strings $\alpha, \beta \in \{0,1\}^k$ as follows:[2]

$$C_{\alpha,\beta}(x) = \begin{cases} \beta & x = \alpha \\ 0^k & \text{otherwise} \end{cases}$$

$$D_{\alpha,\beta}(C) = \begin{cases} 1 & C(\alpha) = \beta \\ 0 & \text{otherwise} \end{cases}$$

[2] As stated, D is uncomputable; the paper discusses truncating computation as a solution to this problem but I shall just ignore it here. The same is true for the adversary A to be defined shortly.

The TM $D_{\alpha,\beta}$ can distinguish between the case that C computes the same function as $C_{\alpha,\beta}$ from the case that it computes the same function as $C_{\alpha',\beta'}$ for any $(\alpha',\beta') \neq (\alpha,\beta)$. Consider adversary A that simply computes $A(C,D) = D(C)$, and let Z_k be a Turing machine that always outputs 0^k. Consider α and β selected uniformly at random from $\{0,1\}^k$. Then

$$\begin{aligned} \Pr[A(\mathcal{O}(C_{\alpha,\beta}),\mathcal{O}(D_{\alpha,\beta})) = 1] &= 1 \\ \Pr[A(\mathcal{O}(Z_k),\mathcal{O}(D_{\alpha,\beta})) = 1] &= 2^{-k}. \end{aligned} \tag{1}$$

(The paper actually gives the second probability as 0, but I believe this is incorrect because $A(\mathcal{O}(Z_k),\mathcal{O}(D_{\alpha,\beta})) = 1$ when $\beta = 0^k$; the chance that $\beta = 0^k$ is 2^{-k}.)

Now any polynomial-time algorithm S with oracle access to $C_{\alpha,\beta}$ and $D_{\alpha,\beta}$ has exponentially small probability of querying either oracle at a point where its value is nonzero, so

$$\left|\Pr[S^{C_{\alpha,\beta},D_{\alpha,\beta}}(1^k) = 1] - \Pr[S^{Z_k,D_{\alpha,\beta}}(1^k) = 1]\right| \leq 2^{-\Omega(k)}. \tag{2}$$

But the combination of (1) and (2) contradicts the "virtual black box" property in the definition of a 2-TM obfuscator. ■

We need one more tool to prove Theorem 1: for functions or TMs $f_0, f_1 : X \to Y$, define their *combination* $f_0 \# f_1 : \{0,1\} \times X \to Y$ as $(f_0 \# f_1)(b,x) = f_b(x)$. In reverse, we can decompose a TM $C : \{0,1\} \times X \to Y$ into $C_0 \# C_1$ by setting $C_b(x) = C(b,x)$.

Proof sketch. (Of Theorem 1) Suppose for the sake of contradiction that $\mathcal{O}$ is a TM obfuscator.

Let $C_{\alpha,\beta}$, $D_{\alpha,\beta}$, and Z_k be as defined in the proof sketch of Lemma 2. Define additional TMs $F_{\alpha,\beta} = C_{\alpha,\beta} \# D_{\alpha,\beta}$ and $G_{\alpha,\beta} = Z_k \# D_{\alpha,\beta}$. Let adversary A decompose the TM F it gets into $F_0 \# F_1$ and then output $F_1(F_0)$. Then, taking S to be the simulator from Definition 2.1,

$$\begin{aligned} \Pr[A(\mathcal{O}(F_{\alpha,\beta})) = 1] &= 1 \\ \Pr[A(\mathcal{O}(G_{\alpha,\beta})) = 1] &= 0 \\ \left|\Pr[S^{F_{\alpha,\beta}}(1^k) = 1] - \Pr[S^{G_{\alpha,\beta}}(1^k) = 1]\right| &\leq 2^{-\Omega(k)} \end{aligned}$$

where the probabilities are over uniform random $\alpha, \beta \in \{0,1\}^k$ and the coin tosses of A, S, and $\mathcal{O}$. This contradicts Definition 2.1. ■

To relate this to the functions and predicates mentioned earlier, the class of functions considered is (loosely) the combined TM $F = C_{\alpha,\beta} \# D_{\alpha',\beta'}$, and the predicate is:

$$\pi(F) = \begin{cases} 1 & (\alpha,\beta) = (\alpha',\beta') \\ 0 & \text{otherwise} \end{cases}$$

3 Encrypted functions and undetachable signatures

The other paradigm for code hiding represented in the papers I read is the mobile code paradigm. The general problem is that Alice wishes to send a software agent to a remote system to interact (semi-)autonomously with Bob, but Alice has no reason to trust Bob or his system. In particular, these papers primarily discuss the case where Alice wants the agent to perform cryptographic signatures on her behalf while running on Bob's computer but does not want Bob's system to have access to her signing key. Alice of course also wants to restrict the set of documents that the agent may sign on her behalf.

3.1 Towards Mobile Cryptography

Sander and Tschudin [5] describe their goal as *computing with encrypted functions* or *evaluation of encrypted functions (EEF)*. They do not offer a formal statement of the problem, but they characterize it as follows:

> Alice (the originating host) has an algorithm to compute a function f. Bob (the remote host) has an input x and is willing to compute $f(x)$ for her, but Alice wants Bob to learn nothing "substantial" about f. Moreover, Bob should not need to interact with Alice during the computation of $f(x)$.

They leave "substantial" unspecified in general, saying that its definition depends on the application.

The paper first describes a proposal for limited EEF using homomorphic encryption.

Definition 3.1 *Let $E : R \to S$ be an encryption function. Call E* additively homomorphic *if there is an efficient algorithm* PLUS *to compute $E(x + y)$ from $E(x)$ and $E(y)$ without revealing x or y. Call E* mixed-multiplicatively homomorphic *if there is an efficient algorithm* MIXED-MULT *to computer $E(xy)$ from $E(x)$ and y without revealing x.*

Given an encryption scheme E that is both additively and mixed-multiplicatively homomorphic, Sander and Tschudin propose a scheme for EEF on the set of polynomials.

Alice creates a program $P(x)$ to implement the polynomial

$$p(x) = \sum_{i_1, i_2, \ldots, i_s} a_{i_1 \ldots i_s} \prod_{j=1}^{s} x_j^{i_j}$$

on input $x = (x_1, x_2, \ldots, x_s)$ in the following way. She precomputes $E(a_i)$ and forms $P(x)$:

1. Compute the monomials $\prod_{j=1}^{s} x_j^{i_j}$

2. Use MIXED-MULT to multiply each encrypted coefficient by the corresponding unencrypted monomial

3. Use PLUS to sum the terms of step 2

Alice can now send P to Bob, who can compute the function on his input x and send the (encrypted) result back to Alice.

The paper describes information leakage related to the set of nonzero coefficients, which they call the *skeleton* of the polynomial, but the explanation is very unclear and I cannot determine precisely what the authors mean. I believe the problem comes from the fact that the authors assume a deterministic cryptosystem (though they never state whether the cryptosystem is deterministic or probabilistic), so the adversary could compute $E(0)$ and compare it with the encrypted coefficients. In any case, the authors give no proof of security for the scheme, and their analysis does not dispell concerns that Bob may be able to exploit the structure of the polynomial to learn more about the coefficients or the computation.

The idea of composing functions, and the related proposal for "undetachable signatures," is more interesting. The concept is simple: take two rational functions s and f (a rational function can be expressed as the quotient of two polynomials) and compose them as $s \circ f$, or $s(f(x))$. The authors claim no known polytime algorithm for decomposing multivariate rational functions, though they give no additional evidence of its hardness (and the source they cite is from 1991). Alice forms an undetachable signature by composing a signing function s with the function f that it should be tied to, and passing the composition $f_s = s \circ f$ to Bob, who then computes $m = f(x)$ and $z = f_s(x)$.

The paper describes four attacks against composed functions, including an interesting interpolation attack; I will forgo further discussion of this paper, however, in order to move directly to a proposed implementation of undetachable signatures.

3.2 Secure Transactions with Mobile Agents in Hostile Environments

Kotzanikolaou, Burmester, and Chrissikopoulos [3] offer a scheme implementing Sander and Tschudin's concept of undetachable signatures using RSA. They describe this proposal in an electronic commerce setting, where Alice wishes to purchase goods that Bob (along with others) will bid to sell her.

Alice creates an RSA signing key in the usual way, with modulus $n = pq$ and signing exponent d, and let hash() be a cryptographic hash function. Let A be an identifier for Alice, and let R_A be a string describing her requirements, such as the name and maximum price for the product she wishes to purchase, as well as a timestamp. Bob likewise has an identifier B and requirements R_B, which include his bid.

Alice equips her agent with the function pair

$$\begin{array}{rcll} f(\cdot) & = & h^{(\cdot)} & \mod n \quad \text{and} \\ f_s(\cdot) & = & k^{(\cdot)} & \mod n \end{array} \tag{3}$$

where $h = \text{hash}(A, R_A)$ and $k = h^d \mod n$, which is her (naive) RSA signature of h. Here $f_s(\cdot) = s \circ f(\cdot)$ because exponentiations are commutative. Alice sends her agent to Bob's server, including f, f_s, A, and R_A. Bob then gives the agent his input $x = \text{hash}(B, A, R_B)$ and obtains the signature $(m, z) = (f(x), f_s(x))$. Kotzanikolaou et al. assert that although the server can give the agent input x such that R_B is not compatible with R_A, the resulting signature will be invalid; presumably this is because anyone can inspect the two values and see the incompatibility.

4 Analysis

I have described three papers that discuss the problem of hiding information about a program when an adversary controls the platform on which the program runs. Barak et al. take a theoretical approach and prove that information about a program cannot in general be hidden in the black-box model; that is, there exist functions for which oracle access to the function provides provably less information than access to the code of any program that computes the function. On the other hand, Kotzanikolaou et al. and Sander and Tschudin take a more practical approach and try to develop classes of functions that can be obfuscated in some meaningful way.

4.1 Disillusionment

Let KBC denote the RSA detachable signature scheme presented by Kotzanikolaou et al. KBC seems at first to be an interesting result that gives a mobile agent some novel signing power. It turns out, however, to be nothing but smoke: their detachable signature agent functions are no more powerful than one conventional signature on an appropriate document.

Alice's goal is to commit to a transaction that falls within her specified parameters. Her security requirement is that Bob may not cause her to sign any transaction incompatible with her specifications (this subsumes compromise of her signing key).

I now claim that that the KBC scheme is critically flawed, and furthermore, it has no advantages over Alice's merely signing the requirements and transmitting that document and signature to Bob.

The first flaw in the KBC system is that they propose signing by exponentiation of $\text{hash}(R_A)$, for which they suggest using MD5. However, MD5 has a 128-bit range, which is far short of a reasonable domain size for RSA; as we saw in class, this setup is not provably secure against forgery. The security could be ensured with a probabilistic full-domain hash, but then the commutative property of the exponentiation is lost because h (respectively, k) is hashed with

a random value before the exponentiation. It is not clear that this flaw can be fixed, as the commutativity is the key to KBC.

I will also now prove that the goals of KBC can be accomplished merely by Alice's signing the document (A, R_A) specifying what transactions she will agree to.

Theorem 3 *Given:*

1. $D_A = (A, R_A)$ *is a document or string specifying Alice's identity and her requirements on a transaction with Bob*
2. (d, e) *is Alice's RSA signing key with signing exponent* d *and verifying exponent* e
3. $KBC(D_A)$ *is the undetachable signature scheme specified in Eq. 3, with* $h = MD5(D_A)$ *and* $k = h^d$
4. $S_A = (r, (FDH(r, D_A))^d \mod n)$ *is Alice's full-domain hash RSA signature of* D_A

Then for all transactions $M \in \{0,1\}^*$*: if Bob can cause Alice to commit to* M *given only* S_A *and* D_A*, then he can cause Alice to commit to* M *given only* $KBC(D_A)$*.*

Proof. The notion of causing Alice to commit to M must include agreement between R_A and R_B because a KBC computation is considered invalid if they do not agree.

So consider what it means to commit to M given $KBC(D_A)$ (putting aside the issue of forgery introduced by the MD5). Bob can present a digital signature by Alice of (D_A, B, R_B) for any (B, R_B), and it is considered valid if R_B agrees with R_A; hence Bob causes Alice to commit to M if he produces a valid signature on (D_A, B, R_B) and R_B agrees with R_A.

In the case that Bob has only S_A and D_A, he causes Alice to commit to M if he presents a valid signature on (D_A) and a (B, R_B) such that R_B agrees with R_A. The signature does not cover (B, R_B) in this case, but that does not matter because the exact same set of parameter documents will be accepted. ■

5 Conclusion

In this paper, I analyzed three papers and discussed the two paradigms of obfuscation presented. I provided an ϵ-new result by demonstrating weaknesses in the scheme of Kotzanikolaou et al. And I had fun doing it! I greatly enjoyed exploring the subject of program obfuscation, and I may return to it at some point.

References

[1] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs (extended abstract). In *Advances in Cryptology — CRYPTO 2001*, pages 1–18. Springer-Verlag, 2001.

[2] Niklas Borselius, Chris J. Mitchell, and Aaron Wilson. A pragmatic alternative to undetachable signatures. *ACM SIGOPS Operating Systems Review*, 36(2):6–11, April 2002.

[3] Panayiotis Kotzanikolaou, Mike Burmester, and Vassilios Chrissikopoulos. Secure transactions with mobile agents in hostile environments. In *Australasian Conference on Information Security and Privacy*, pages 289–297, 2000.

[4] Tomas Sander and Christian F. Tschudin. Protecting mobile agents against malicious hosts. In *Mobile Agents and Security*, volume 1419 of *LNCS*, pages 44–60. Springer-Verlag, Berlin, Germany, 1998.

[5] Tomas Sander and Christian F. Tschudin. Towards mobile cryptography. In *IEEE Symposium on Security and Privacy*, pages 215–24, Oakland, CA, 1998.