Computing on Encrypted Data
(Extended Abstract)

Amit Sahai[⋆]

University of California, Los Angeles
sahai@cs.ucla.edu

Abstract. Encryption secures our stored data but seems to make it inert. Can we process encrypted data without having to decrypt it first? Answers to this fundamental question give rise to a wide variety of applications. Here, we explore this question in a number of settings, focusing on how interaction and secure hardware can help us compute on encrypted data, and what can be done if we have neither interaction nor secure hardware at our disposal.

1 Introduction

The increased frequency of cyber-attacks as well as governmental regulations, such as HIPPA and GLBA in the United States, are driving enterprises to encrypt more and more of their data. In the near future, it is expected that encryption will be ubiquitous and the majority of enterprise data will be stored encrypted.

While protecting data with encryption provides clear benefits, it also has some significant drawbacks. Unlike cleartext data which can be edited and searched, encrypted data seems to support none of these operations. It is commonly believed that encrypted data cannot be manipulated without first decrypting it. This often leads to complex key management where multiple entities are given access to the decryption keys. The end result is reduced security and increased cost. To give some examples, consider a database that stores encrypted transactions. Locating all transactions within a certain range of dates is believed to be impossible without giving the database access to decryption keys. More generally, once a database column is encrypted it is typically impossible to issue general queries on that column. The same limitations apply to systems managing encrypted file servers and email. This leads to the following fundamental question:

Can we process encrypted data without having to decrypt it first?

Answers to this fundamental question give rise to a wide variety of applications. Here, we explore this question in a number of settings, focusing on how

⋆ Research supported in part from NSF grants 0627781, 0716389, 0456717, and 0205594, a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, an Alfred P. Sloan Foundation Fellowship, and an Okawa Foundation Research Grant.

interaction and secure hardware can help us compute on encrypted data, and briefly discussing our recent work in this area. We conclude with some musings about what can be done if we have neither interaction nor secure hardware at our disposal.

2 Interaction

Somewhat surprisingly, over two decades ago a powerful positive answer was given to this fundamental question by making use of *interaction*, which led to the flourishing field of secure two-party and multiparty computation [24,11,3,6]. Secure multi-party computation allows users that hold different sets of secret data to collaborate and analyze all their data together, in such a way that no user learns anything about anyone else's secret data except for whatever is revealed by the output of the analysis. Thus, even though the secrecy of other users' data is guaranteed through encryption methods, arbitrarily complex functions of all the data can be jointly computed by the users.

The key to this is interaction. The intuition behind what makes this possible is the fact that the holder of the data in unencrypted form is available to "answer questions" from the other users, who only hold the data in some encrypted form. Of course, the challenge is the following: how can we allow others to process our data when we are only willing to answer questions that don't reveal anything about our data? A number of fascinating and sophisticated techniques have been discovered to meet this challenge.

Secure two-party and multiparty computation remains a vast and exciting field. We will briefly mention here two works regarding secure computation that we were recently involved in. In recent joint work with Ishai and Prabhakaran [17], we show how to achieve a very high level of efficiency for secure two-party and multi-party computation – where the communication overhead of a secure two-party computation protocol for a function F is only a fixed constant factor larger than the circuit size of F. This is possible under standard and minimal computational assumptions. By making somewhat stronger and more non-standard intractability assumptions, in another recent work, joint with Ishai, Kushilevitz, and Ostrovsky [16], we gave the first steps towards achieving secure two-party computation with*constant computational overhead*. By this, we mean that the amount of computation (not just communication) necessary to securely compute some function F is only a fixed constant factor larger than the circuit size of F.

3 Secure Tamper-Proof Hardware

Another resource with a much more obvious application to our question is secure tamper-proof hardware. By "secure tamper-proof hardware," we mean a device that implements some functionality in a "black-box" manner, so that the holder of the device can only query the device with some input and receive some output from the device. As such, it is quite intuitive that such devices could allow for the processing of encrypted data. In particular, the device could have the decryption key built into it, so the device would take as input some encrypted values, decrypt them and process them, and reencrypt the result. Indeed, Goldreich and Ostrovsky [12] showed that using such devices, one can efficiently run "encrypted programs". That is, the entire program to be executed is available only in encrypted form, and therefore nothing about the program is revealed except for roughly how long it runs, and what its outputs are on given inputs. This goal is often called program obfuscation. Even with trusted hardware, this goal is non-trivial to achieve because of the problem of "replay" attacks, where the result of some intermediate computation is maliciously reused at some other point of the program where it should not be used. Goldreich and Ostrovsky [12] solve this problem using secure hardware tokens that maintain state information. Recently, the work of Goyal and Venkatesan [15] achieved the same result using *stateless* hardware tokens, where more sophisticated cryptographic techniques are used to defend against the problem of replay attacks.

Perhaps the most natural question to ask in the context of secure tamper-proof hardware is whether it is reasonable to assume that we could really have secure tamper-proof hardware to begin with? This question has led to two very interesting recent lines of work:

– **Can we implement secure tamper-proof hardware out of insecure components?** In joint works with Ishai, Prabhakaran, and Wagner [19,18], we considered this question, and showed how to make secure hardware devices for implementing any function that can tolerate specific side channel attacks (namely bit probes) and specific tampering attacks (namely tampering with the values on individual wires inside the circuit). Interestingly, the techniques we needed to accomplish these goals are closely related to techniques from secure two-party and multiparty computation.

  In other related work, Gennaro et al. [9] considered the question of whether a general secure hardware device could be separated into a readable but tamper proof part, and a separate unreadable but tamperable part. They gave a number of positive results for various specific functions.
– **What can we accomplish with a very simple secure hardware devices?** The recent work of Goldwasser, Kalai, and Rothblum [13] introduce a very simple hardware device that they call a "one-time memory" device (we call such devices "OT tokens" because they can be thought of as implementing the oblivious transfer [21,8] function). This device has built into it two secret strings s_0 and s_1, takes a single bit t has input, outputs s_t and then self-destructs (i.e. becomes useless). Using a collection of such simple devices, and assuming that one-way functions exist, they show how to achieve a relaxation of program obfuscation, where the encrypted program may only be run *once* – a notion which they call *one-time programs.* (Note that this notion can be trivially generalized to allow the program to run a pre-specified k times.)

  In recent joint work with Goyal, Ishai, and Wadia [14], we extend their results in three ways, two major and one minor. One minor improvement that we offer is that our secure devices can be even simpler – a "bit OT token" where the two strings s_0 and s_1 are replaced with just two bits b_0 and b_1, and the output is just the bit b_t. Our two major improvements are: (1) We strengthen the notion of one-time programs to allow the manufacturer of the program to specify guarantees about the encrypted program that the evaluator of the program can verify at run-time (*i.e.* when the owner of the tokens, Alice, runs the program on her secret input x_A, she could be assured that the program will simply compute $f(x_A, x_B)$, where f is a specific function known to Alice, and x_B is a secret input of the manufacturer). We call this stronger notion one-time programs with security against malicious sender. (2) We achieve this stronger notion of one-time programs *unconditionally*, without needing to make any unproven assumptions, such as the existence of one-way functions.

4 The "Plain" Model

Unfortunately, much less is known about computing on encrypted data in the "plain model," where encrypted data is given to us, and we must non-interactively process it without the assistance of any secure hardware. Indeed, in joint work with Barak, Goldreich, Impaggliazzo, Rudich, Vadhan, and Yang [2], we showed that in this situation the goal of program obfuscation (encrypted programs) is in general impossible to achieve, even when the programs to the obfuscated/encrypted are fairly simple. Nevertheless, this area is filled with a number of fascinating open questions.

One of the central open problems in cryptography today, called *doubly-homomorphic encryption*, was first posed by Rivest et al. [22] almost 30 years ago. The problem can be stated succinctly as follows. Let E be a (semantically) secure encryption system where plaintexts are integers in $\{0, \ldots, n\}$ for some n. The system is said to be doubly homomorphic if given the encryption of two plaintexts $E_k(x)$ and $E_k(y)$ anyone can construct a new independent encryption of $E_k(x+y)$ and $E_k(x \cdot y)$, without knowledge of x or y. This property enables arbitrary computations on encrypted data, where the output and all intermediate computations remain in encrypted form. More precisely, E is doubly homomorphic if there are two efficient algorithms A_+ and A_* such that

$$A_+\big(\, E_k(x_1),\ E_k(x_2)\,\big) \ =_p\ \big(\, \mathbf{E_k(x_1+x_2)},\ E_k(x_1),\ E_k(x_2)\,\big)\,, \quad \text{and}$$

$$A_*\big(\, E_k(x_1),\ E_k(x_2)\,\big) \ =_p\ \big(\, \mathbf{E_k(x_1\cdot x_2)},\ E_k(x_1),\ E_k(x_2)\,\big)$$

where $=_p$ denotes indistinguishability of distributions.

Virtually nothing is known about the existence of such an E, other than some very weak impossibility results [5]. The **major open problem** is to build a semantically secure public-key encryption that is doubly homomorphic where, furthermore, algorithms A_+ and A_* are efficient and practical. Such a system will enable a host of magical applications, such as:

- **Searching.** In principle, E will enable a very rich set of search queries on encrypted data, as discussed in the previous sections. It will also enable very general searches with encrypted queries.
- **Minimally interactive distributed data-mining and secure computation**. E will enable simple minimally-interactive data-mining of databases distributed across multiple competing entities (e.g., multiple airlines, hospitals, or governments). The goal is to ensure that nothing other than the data-mining results is revealed. Currently, this requires highly interactive protocols based on secure computation techniques.

Several existing public-key systems, such as ElGamal [7] and Pallier [20], are singly-homomorphic — they support only one homomorphic operation. Previous attempts [23] at building doubly-homomorphic systems only applied to boolean operations and doubled the ciphertext size at every step. As a result, one could only perform few boolean operations before the ciphertext size became unmanageable. Recent work of Boneh, Goh, and Nissim [4] use techniques from elliptic curves to construct a system that allows for an arbitrary number of additions, and one multiplication. Surprisingly, this small additional homomorphic property already leads to a number of exciting new constructions. Thus, even seemingly minor progress on this fundamental open question can lead to significant payoffs.

5 Conclusions

The question of computing on encrypted data has spawned countless fascinating techniques as well as deep open problems. It remains a driving force behind much of the most exciting research in cryptography today.

Acknowledgements

We thank Dan Boneh, Vipul Goyal, Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, Ramarathnam Venkatesan, Akshay Wadia, David Wagner, and Brent Waters for several insightful conversations on these topics over the course of several collaborations.

References

1. Proc. 20th STOC. ACM, New York (1988)
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139. Springer, Heidelberg (2001)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proc. 20th STOC [1], pp. 1–10

4. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–342. Springer, Heidelberg (2005)
5. Boneh, D., Lipton, D.: Black box fields. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109. Springer, Heidelberg (1996)
6. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: Proc. 20th STOC [1], pp. 11–19
7. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
8. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM 28(6), 637–647 (1985)
9. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
10. Goldreich, O.: Foundations of Cryptography: Basic Applications. Cambridge University Press, Cambridge (2004)
11. Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: ACM (ed.) Proc. 19th STOC, pp. 218–229. ACM, New York (1987); See [10, Chap. 7] for more details
12. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM 43(3), 431–473 (1996)
13. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
14. Goyal, V., Ishai, Y., Sahai, A., Wadia, A.: Cryptography from tamper-proof hardware, revisited (manuscript, 2008)
15. Goyal, V., Venkatesan, R.: On obfuscation complete oracles (manuscript, 2008)
16. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: STOC, pp. 433–442. ACM, New York (2008)
17. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
18. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
19. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
20. Pallier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
21. Rabin, M.: How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory (1981)
22. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. Foundations of Secure Computation (1978)
23. Sander, T., Young, A., Yung, M.: Non-interactive CryptoComputing for NC^1. In: Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS), New York, NY, USA, October 1999, pp. 554–567. IEEE Computer Society Press, Los Alamitos (1999)
24. Yao, A.C.: How to generate and exchange secrets. In: Proc. 27th FOCS, pp. 162–167. IEEE, Los Alamitos (1986)