Cost and Scalability of Hardware Encryption Techniques

Adam Waksman
Department of Computer Science
Columbia University
New York, USA
waksman@cs.columbia.edu

Simha Sethumadhavan
Department of Computer Science
Columbia University
New York, USA
simha@cs.columbia.edu

I. Data Obfuscation and Fully Homomorphic Encryption

We discuss practical details and basic scalability for two recent ideas for hardware encryption for trojan prevention. The broad idea is to encrypt the data used as inputs to hardware circuits to make it more difficult for malicious attackers to exploit hardware trojans. The two methods we discuss are data obfuscation and fully homomorphic encryption (FHE).

Data obfuscation [5] is a technique wherein specific data inputs are encrypted so that they can be operated on within a hardware module without exposing the data itself to the hardware.

FHE is a technique recently discovered to be theoretically possible [2]. With FHE, not only the data but also the operations and the entire circuit are encrypted. FHE primarily exists as a theoretical construct currently. It has been shown that it can theoretically be applied to any program or circuit [2]. It has also been applied in a limited respect to some software [4]. Some initial algorithms for hardware applications have been proposed [1].

We find that data obfuscation is efficient enough to be immediately practical, while FHE is not yet in the practical realm. There are also scalability concerns regarding current algorithms for FHE.

A. Data Obfuscation

Data obfuscation encrypts or re-maps the inputs to hardware modules. We can implement two different versions of this, one for modules that transport data and one for modules that operate on data.

The data transport version is for protecting data while it is in memory, registers, buses or any of the other myriad components that transport and store data. For this case, we have a dynamically chosen on-chip random value called the *key*. The key is applied via bitwise XOR with all data, addresses and indices for memory, registers and other similar components. While data is stored in memory, it has a random value and is at a random location. Reads and writes are applied with the same key, so finding data is never a problem.

For arithmetic circuits, we need a structure preserving mapping so that the work of the arithmetic circuits is not lost. Our idea for this is to use displacement constants that have a small number of ones in their binary representations. As noticed in prior work [3], such numbers allow for fast arithmetic in hardware and also achieve good randomness. To take a multiplier as a motivating example, the goal of a multiplier is to compute:

$$P = X * Y$$

without receiving X or Y as inputs. To do this, we choose displacement constants a and b. These constant should be odd, positive integers; they need not be prime. We then compute $a+X$, $b+Y$, bX, and aY, all of which can be done with fast partial adders, and we compute the constant ab offline (does not need to be re-computed each cycle, the result can be stored on-chip). Our multiplier then takes the inputs $a+X$ and $b+Y$ to produce :

$$XY + aY + bX + ab$$

We then subtract out the three values we've pre-computed to get:

$$\overbrace{XY + aY + bX + ab} - \underbrace{ab - bX - aY} = XY = P$$

The original multiplier is the only circuit that has done any multiplication. Since addition is structure preserving over arithmetic operations, this technique can be applied with minor modifications to most common arithmetic circuits, with varying cost. Due in part to the good scalability of fast adders, the cost of this technique scales well to large bit widths.

The takeaway is that data obfuscation scales well.

B. Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) has excellent security properties but in its current form appears to be impractical. In FHE, input data has to be encrypted before entering a unit and then periodically re-encrypted depending on the logical depth. Both of these operations are too expensive.

Logical depth refers to the length of a path of logic in a combinational circuit. In FHE, each logical gate roughly doubles the size of the data, so that data has to be periodically re-encrypted to reset the process.

The complexity of the encryption relies on large prime numbers and has a quadratic cost in terms of the bit width of the prime numbers. The cost essentially boils down to a multiply. Each FHE gate requires at least one full multiply operation, and an n-bit multiply scales as n^2 in terms of area and power. For a prime number size p and a logical depth d, the area and power costs of a single FHE gate scale as $c(2^d p)^2$ where c is the baseline cost.

The factor of 2^d comes that from the fact that each multiply doubles the bit width. For example, the product of two 32-bit numbers is a 64-bit number. If the circuit has depth d, since each gate requires at least one multiply operation, the total bit width has to be at least 2^d larger than that of the initial prime number size. Thus the size of the operands is at least $2^d p$, so the cost in terms of area and power scales as:

$$(2^d p)^2$$

or equivalently:

$$(2^{2d})p^2$$

The conclusion is that current implementations of FHE are too expensive to be immediately practical. The issue does not yet come down to low-level circuit optimizations. The overall algorithm is expensive. The exponential growth in cost with respect to logical depth is the biggest problem. Prime number size is also an issue, as large multipliers are expensive.

For a conservative lower-bound estimate on cost, we synthesized some small FHE circuits. Consider for example the relatively small prime number size of 64 bits and a small depth of only two. Using 90nm technology libraries and a large modern server die with 300 mm^2 of area, we could fit at most about a hundred logical gates. This is too few gates to build anything interesting. A design that small could be much more easily secured with formal verification and/or code review. Results are similar with the absolute minimum logical depth of one.

These results show that FHE is orders of magnitude away from where it would need to be for practical applications. Moving into smaller technology nodes and applying circuit optimization techniques will not be enough to overcome that large of a gap.

The takeaway is that current implementations of FHE scale poorly and are not yet practical.

REFERENCES

[1] M. Brenner, J. Wiebelitz, G. von Voigt, and M. Smith. Secret program execution in the cloud applying homomorphic encryption. In *Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on*, pages 114 –119, 31 2011-june 3 2011.

[2] C. Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, 2010.

[3] M. Kharbutli, Y. Solihin, and J. Lee. Eliminating conflict misses using prime number-based cache indexing. *IEEE Transactions on Computers*, 54(5):573–586, 2005.

[4] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovitch. SCiFI - A System for Secure Computation of Face Identification. In *Proceedings of the 31st IEEE Symposium on Security and Privacy*, 2010.

[5] A. Waksman and S. Sethumadhavan. Silencing hardware backdoors. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, pages 49–63, Washington, DC, USA, 2011. IEEE Computer Society.