Techniques for computing on encrypted data in a practical system[*]

Raluca Ada Popa

September 2014

In this document, we survey various techniques for computing on encrypted data that might be useful when designing a practical security system. These tools or methods provide an interesting spectrum over three coordinates: functionality, security, and performance. Unfortunately, none of these tools alone suffices to enable running the systems we have today over encrypted data. They typically fail to achieve a desired goal with respect to at least one of these three coordinates.

Nevertheless, understanding the tradeoffs these tools provide in functionality, security, and performance, coupled with an understanding of the targeted system, has enabled building systems that are practical for a class of applications and provide a meaningful level of security. Some examples of such systems are [PRZB11, PSV+14, ABE+13, AEK+13, TKMZ13, TLMM13, KGM+14, SSSE14, TSCS13, HHCW12, KKM+13]. Hence, below we describe each tool or method from a functionality, security and performance standpoint, describing the tradeoff it strikes in this space, and the settings in which it might be practical. Since our focus is this tradeoff, we keep the cryptographic presentation of each tool informal, and instead reference outside formal presentations.

We divide these tools and methods in two categories: tools that leak (virtually) nothing about the data and tools that reveal a well-defined function of the data. Many times, the information revealed in the second category helps improve performance drastically.

□ 1 Tools with no leakage

◇ 1.1 Fully homomorphic encryption

Functionality. FHE [Gen09] is a public-key encryption scheme [Gol04]. Using a key generation algorithm, anyone can create a pair of keys: a secret key sk and a public key pk. Anyone can encrypt a value x using the public key pk and obtain a ciphertext $\widehat{x} \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$, where the $\widehat{\ }$ notation indicates encryption with FHE.

An evaluation algorithm Eval can evaluate functions over encrypted data. A function f to be evaluated is typically represented as a boolean circuit of polynomial size in the input size; often, the gates in such a circuit compute either addition or multiplication in a finite field. Using the public key, anyone can evaluate f by computing $\mathsf{Eval}(\mathsf{pk}, \widehat{x_1}, \ldots, \widehat{x_n}, f)$ and obtain $\widehat{f(x_1, \ldots, x_n)}$, the encrypted computation result. This computation result, as well as any other FHE ciphertext, can be decrypted with the secret key sk.

The current FHE schemes enable running virtually any function over encrypted data.

Security. FHE provides strong security guarantees, called *semantic security* [Gol04]. Intuitively, semantic security requires that any adversary holding only the public key and a ciphertext cannot learn any information about the underlying plaintext, other than its length.

[*]This document is based on [Pop14]

Performance and use cases. Following Gentry's scheme [Gen09], there have been a lot of FHE schemes proposed [Gen09, DGHV10, SS10, BV11b, BV11a, Vai11, BGV12, GHS12a, GHS12b, LTV12, Bra12, GSW13], many of which improved the performance of the original scheme drastically. There is also an open-source implementation of FHE available, HElib [Hal13, HS14].

Nevertheless, as of now, FHE remains too slow for running arbitrary functions or for enabling the complex systems we have today. For example, an evaluation (performed in 2012) of the AES circuit reported a performance of 40 minutes per AES block on a machine with very large memory, being more than six orders of magnitude slower than unencrypted AES evaluation.

There are at least three factors that make FHE slow: the cryptographic overhead, the model of computation, and the strong security definition.

The cryptographic overhead consists of the time to perform operations for each gate of the circuit as well as other maintenance operations. Unfortunately, it is hard to characterize simply the cryptographic overhead of FHE because there are a lot of parameters that affect its performance, such as the multiplicative depth of the circuit to evaluate (the largest number of multiplication gates in the circuit on any path from input to output), the security parameter (which indicates the security level desired), the plaintext size, the exact FHE scheme used, the performance of various operations in the finite fields used, etc. Lepoint and Naehrig [LN14] and Halevi [Hal13, GHS12b] provide performance measurements for various settings of these parameters. From our experience with HElib, whenever the multiplicative depth became more than about 3 or whenever we were encrypting non-bit values, performance was becoming too slow (on the order of seconds as opposed to milliseconds) on a commodity machine for the applications we had in mind.

Regarding the model of computation, in order to evaluate a program with FHE, the program must be compiled into a boolean circuit. This could result in a large circuit with a nontrivial multiplication depth.

Hence, if one uses FHE, they should choose a computation that can be easily expressed in additions and multiplications over a finite field, whose multiplicative depth is small, and whose values encrypted are bits or otherwise live in a small plaintext domain.

The final factor, the security guarantees, brings about inherent impracticality. There are settings in which FHE is prohibitively too slow *even if its cryptographic overhead were zero*. The reason is that the security guarantee is so strong that it prevents certain needed optimizations.

Consider the example of a database table containing data about employees in a company. One column represents their salaries and the other column contains further personal information. Now consider a query that requests all the rows where the salary column equals 100. A regular database would execute this query using an index: a fast search tree that enables the database server to quickly locate the items equal to 100 (in a logarithmic number of steps in the number of entries in the database) and return each row in the database having this salary. However, using FHE, the database server has to scan the whole database! The reason is that FHE prohibits the server from learning anything about the data, even learning whether some data is worth returning or not. If the server does not include a salary item in its computation, the server learns that the item did not match the query, which is prohibited by FHE's security. Such linear scans are prohibitively slow in many real databases today. Moreover, with FHE, the database server cannot learn the number of rows that should be in the result. Hence, the server has to return the worst-case number of rows possible. If some rare queries need to return the whole database, the server has to return the whole database *every time*, which can be prohibitively slow.

In Sec. 2.1, we will see how functional encryption addresses exactly this problem: it weakens the security guarantee in a controlled way, which enables the server to know what rows to return and how to find them efficiently (without the server learning the contents of the rows) and avoid these linear costs.

◇ 1.2 Partially homomorphic encryption (PHE)

Functionality. PHE has a similar definition with FHE with the difference that the function f is a specific function. Here are some examples.

- the Goldwasser-Micali [GM82] cryptosystem computes XOR of encrypted bits,
- Paillier [Pai99] computes addition,
- El Gamal [ElG84] computes multiplication,
- the BGN cryptosystem [BGN05] performs any number of additions, one multiplication, followed by any number of additions.

Security. These offer the same strong security as FHE, namely semantic security.

Performance. These schemes are more efficient and closer to practice than FHE, due to their specialized nature. For example, in the experiments in [PRZB11], Paillier takes 0.005 ms to evaluate addition on two encrypted values, and 9.7 ms to encrypt a value.

□ 2 Tools with controlled leakage

The tools in this section do not provide semantic security: they enable the server to learn a function of the data, but nothing else. One can choose the function of the data that the server learns: ideally, this information is small and does not affect privacy, yet it can be exploited to increase performance drastically. We've seen many cases in which using such a scheme instead of fully homomorphic encryption increased performance by orders of magnitude, and brought a significantly slow solution into the practical realm.

◇ 2.1 Functional encryption

Functionality. In *functional encryption* (FE) [SW05, GPSW06, KSW08, LOS$^+$10, OT10, O'N10, BSW11], anyone can encrypt an input x with a master public key mpk and obtain $\mathsf{Enc}(\mathsf{mpk}, x)$. The holder of the master secret key can provide keys for functions, for example sk_f for function f. (Note how each key is tied to a function!) Anyone with access to a key sk_f and the ciphertext $\mathsf{Enc}(\mathsf{mpk}, x)$ can obtain the result of the computation in plaintext form, $f(x)$, by running $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{mpk}, x))$.

As with homomorphic encryption, there are both specialized FE schemes (that support one fixed function) and generic schemes (that support any function).

Regarding specialized FE schemes, the works of Boneh and Waters [BW07], Katz, Sahai and Waters [KSW08], Agrawal, Freeman and Vaikuntanathan [AFV11], and Shen, Shi and Waters [SSW09] show functional encryption schemes for the inner product function.

Regarding general FE schemes, our work [GKP$^+$13a, GKP$^+$13b] constructs an FE scheme for any general function (where the size of the ciphertext is short). Followup work [GGH$^+$13] constructs general FE in a stronger security model based on a relatively new cryptographic assumption.

Security. Intuitively, the security of FE requires that the adversary learns nothing about x, other than the computation result $f(x)$. Some schemes hide the function f as well (but make some changes to the interface of the scheme or to the precise security definition of the scheme).

Note the difference from homomorphic encryption, in which the server obtains an encrypted computation result $\mathsf{Enc}(f(x))$ and does not know $f(x)$. Hence, this is a weaker security than semantic security; nevertheless, by choosing f in a careful way, one can ensure that little information leaks about x.

Performance and use cases. As with FHE, the practicality of a scheme depends on its cryptographic overhead, model and security. In terms of cryptographic overhead, the general FE schemes are currently prohibitively slow. In fact, so far, we are not even aware of the existence of an implementation of such schemes. On the other hand, the specialized functional encryption schemes could be practical in some settings.

In terms of model and security, FE is closer to practice than FHE. If the cryptographic overhead were engineered to be very small, unlike FHE, the model and security of FE no longer preclude the practical applications discussed in Sec. 1.1 as follows. Recall the database example from Sec. 1.1. With functional encryption, the client (owner of the database and master secret key), gives the server keys for computing certain functions. For example, the client can provide the server with a key for the function $f_{100}(x) = (x \overset{?}{=} 100)$ that compares each salary x to 100. Hence, for each row of the database, the server learns one bit of information: whether the row matches the predicate or not. Nevertheless, the server does not learn the content of those rows or the value 100. This small amount of information drastically improves performance: the server no longer has to return the whole database. (To enable the server to locate the value 100 efficiently, the client can also give keys to the server for functions that help the server navigate the database index.)

◇ 2.2 Garbled circuits

Functionality. With a garbling scheme, someone can compute an "obfuscation" of a function represented as a circuit C, called the garbled circuit, and an encoding of an input x, such that anyone can compute $C(x)$ using only the garbled circuit and the encoded input, without learning anything else about x or C. More concretely, a garbling scheme [Yao82, LP09, BHR12] consists of three algorithms $(\mathsf{Gb.Garble}, \mathsf{Gb.Enc}, \mathsf{Gb.Eval})$ as follows. $\mathsf{Gb.Garble}$ takes as input a circuit C, and outputs the garbled circuit Γ and a secret key sk. In many constructions, the circuit C consists of binary gates such as XOR and AND. $\mathsf{Gb.Enc}(\mathsf{sk}, x)$ takes an input x and outputs an encoding c. Finally, $\mathsf{Gb.Eval}(\Gamma, c)$ takes as input a garbled circuit Γ and an encoding c, and outputs a value y that must be equal to $C(x)$.

Many garbling schemes have the property that the secret key is of the form $\mathsf{sk} = \{L_i^0, L_i^1\}_{i=1}^n$, a set of pairs of labels, one for each bit of the input x, where n is the number of bits of the input. The encoding of an input x is of the form $c = (L_1^{x_1}, \ldots, L_n^{x_n})$ where x_i is the i-th bit of x. Namely, it is a selection of a label from each pair in sk based on the value of each bit in x.

Security. Some garbling schemes can provide both input privacy (the input to the garbled circuit does not leak to the adversary) and circuit privacy (the circuit does not leak to the adversary).

Most garbling schemes are secure *only for a one-time evaluation of the garbled circuit*: namely, the adversary can receive at most one encoding of an input to use with a garbled circuit; obtaining more than one encoding breaks the security guarantee. This means that, to execute a circuit C on a list of encrypted inputs, one has to garble the circuit C every time, for every input. The work to garble C is, in most cases, at least as large as evaluating C.

Our work [GKP+13a] provides garbling schemes that are *reusable*. However, our construction is prohibitively impractical, and thus it represents only a proof of concept.

Performance and use cases. There are a lot of implementations of garbled circuits. The state-of-the-art implementation is JustGarble [BHKR13].

There are at least two factors that affect the efficiency of a garbling scheme: the cost of converting from the desired program to a garbled circuit representation, and the cryptographic cost of garbling and evaluating the garbled circuit.

Recent work [BHKR13] managed to bring the cryptographic overhead of the garbled circuits notably low. For example, JustGarble [BHKR13] evaluates moderate-sized garbled-circuits at an amortized cost of 23.2 cycles per gate, which corresponds to 7.25 nsec on a commodity hardware.

The conversion from a desired program to a circuit of AND and XOR gates is perhaps the most expensive because it tends to create large circuits. There are generic tools that convert a C program to a circuit [BDNP08], as well more efficient methods due to recent work [ZE13]. The most practical solution, though, is to design the circuit for a certain program from scratch, by taking into account some special structure of the program to compute, if that is possible. For example, Kolesnikov et al. [KSS09] shows how to create a simple circuit for comparing two large integers.

Garbling schemes are often used along with oblivious transfer schemes; in such a scheme, party A having an input x, can obtain the labels corresponding to x from party B without B knowing what x is. An efficient state-of-the-art oblivious transfer scheme is due to Asharov et al. [ALSZ13].

◇ 2.3 Secure multi-party computation (MPC)

Functionality. With MPC [Yao82, GMW87], n parties each having a private input (x_i being the private input of party i) can compute jointly a function f such that each party learns the result of the computation $f(x_1, \dots, x_n)$, but no one else learns more about the inputs to the other parties.

There exist secure multi-party computation protocols for generic functions [Yao82, GMW87, LP07, IPS08, LP09], as well as implementations for generic functions [HKS+10], [MNPS04], [BDNP08], [BHKR13]. MPC protocols are often constructed using garbled circuits at their basis.

Besides these generic constructions, there are tens of specialized MPC algorithms, supporting all kinds of functions.

Security. Intuitively, each party learns $f(x_1, \dots, x_n)$ and nothing else about any other party's input. There are a number of variations on the security definition. For example, the security can be defined with respect to honest-but-curious adversaries (adversaries that follow the protocol but try to learn private data), or malicious adversaries (adversaries that can cheat in a variety of ways).

Performance and uses cases. The state-of-the-art tools for generic computation [HKS+10], [MNPS04], [BDNP08], [BHKR13] are too inefficient to run general-purpose programs. Hence, they can be practical when they run simple programs.

To run slightly more complex computation, the hope is again specialization. The trick here is to observe some structural properties specific to the computation of interest; then, hopefully, one can exploit these properties to design an efficient MPC protocol. For example, the VPriv [PBB09] protocol we designed uses MPC to compute certain aggregation functions on a driver's path, such as a toll cost. The common operations to these aggregation functions were summation and set intersection. We designed a specialized protocol which was three orders of magnitude faster than the then state-of-the-art generic tool. Besides VPriv, the literature contains tens of other specialized and efficient MPC protocols [KSS13] for all kinds of computations. For guidelines on designing efficient MPC protocols, Kolesnikov et al. [KSS13] provide a systematic guide.

One caveat to remember is that these protocols are interactive: they require communication between the various parties involved and many times require that all parties be online at the same time.

◇ 2.4 Specialized tools

There are a lot of schemes that can compute all kinds of specialized functions or fit in various setups, while revealing only a well-defined amount of information to the server. Many of these are very efficient because of their specialized nature. When designing a practical system, these provide a gold mine of resources because of the efficiency they provide. For example, in CryptDB, 4 out of the 5 encryptions that can compute fall in this category.

Here are two notable examples:

- *Searchable encryption*. There are all kinds of searchable encryption schemes such as [SWP00, Goh, BCOP04, CM05, BBO07, CGKO06, BW07, BDDY08, ZNS11, YLW11, Raj12, KPR12, CJJ$^+$13, CGKO06, YLW11, ZNS11, Raj12]; Bosch et al. [BHJP14] provide a recent survey. The common functionality of these schemes is to enable a server to find whether some (encrypted) keyword appears in an encrypted document. Some of the variations come from whether the server can search for conjunctions or disjunctions of keywords, whether the server can use a search data structure to improve search time, whether the server can search by some limited regular expressions, variations in security (whether the server learns the number of the repetitions of a word in a document), and others.
- *Order-preserving encryption (OPE)* [AKSX04, PLZ13] maintains the order of the plaintexts in the cirphertexts. Namely, if $x \leq y$, then $\mathsf{Enc}(x) \leq \mathsf{Enc}(y)$. OPE has the advantage of allowing existing servers (such as database servers) to compute range, order, or sort queries on encrypted data in *the same way* as on unencrypted data. Further benefits are that existing software does not need to change to compute on the encrypted data and fast search data structures still work. At the same time, OPE should be used carefully because of its order leakage. One typical example of OPE usage are timestamps, a combined numberic value specifying the tuple (year, month, day, hour, minute, second, millisecond, microsend). Hence, timestamps come from a sparse field, are unique and have high entropy, so their order does not reveal much about their exact value.
- *Deterministic encryption* [BFO08] enables equality checks because the same value will yield the same ciphertext, considering a fixed encryption key. Deterministic encryption is a great tool to use when the relevant values are known to be unique: in this case, such encryption scheme leaks virtually nothing about the data.

Despite all the tools we presented above, many times in practice, we need an encryption scheme that does not exist (as was the case with the order-preserving scheme for CryptDB [PRZB11] and the multi-key search for Mylar [PSV$^+$14]). In that case, one needs to design a new tool that fits the setting of interest.

□ 3 Final lessons

Here are a few simple lessons to take from the above discussion:

- There are a number of ways to compute on encrypted data and they provide different tradeoffs in the space of functionality, security, and efficiency.
- The key to good performance is specialization: use or design a specialized encryption scheme, or find a specialized model of the system setting of interest to plug into an existing generic scheme.
- When fully general functionality and ideal security are not possible, focus on a meaningful class of applications with commonalities and design a system that provides a meaningful level of security for this class.

References

[ABE$^+$13] Arvind Arasu, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, and Ramaratnam Venkatesan. Orthogonal security with Cipherbase. In *Proceedings of the 6th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

[AEK$^+$13] Arvind Arasu, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, and Ramaratnam Venkatesan. A secure coprocessor for database applications. In *Proceedings of the 23rd International Conference on Field Programmable Logic and Applications (FPL)*, 2013.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Proceedings of the 17th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2011.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004.

[ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, 2013.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *Proceedings of the 27th International Cryptology Conference (CRYPTO)*, 2007.

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Proceedings of the 23rd Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2004.

[BDDY08] Feng Bao, Robert H. Deng, Xuhua Ding, and Yanjiang Yang. Private query on encrypted data in multi-user settings. In *Proceedings of the 4th International Conference on Information Security Practice and Experience*, 2008.

[BDNP08] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: A system for secure multi-party computation. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, 2008.

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Proceedings of the 28th International Cryptology Conference (CRYPTO)*, 2008.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proceedings of the 2nd Theory of Cryptography Conference (TCC)*, 2005.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS)*, 2012.

[BHJP14] Cristopher Bosch, Pieter Hartel, Willem Jonker, and Andreas Peter. A survey of provably secure searchable encryption. In *ACM computing surveys, 47 (2)*, 2014.

[BHKR13] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (IEEE S&P)*, 2013.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Garbling schemes. *Cryptology ePrint Archive, Report 2012/265*, 2012.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Proceedings of the 32nd International Cryptology Conference (CRYPTO)*, 2012.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Proceedings of the 8th Theory of Cryptography Conference (TCC)*, 2011.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Proceedings of the 31st International Cryptology Conference (CRYPTO)*, 2011.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Proceedings of the 4th Theory of Cryptography Conference (TCC)*, 2007.

[CGKO06] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13rd ACM Conference on Computer and Communications Security (CCS)*, 2006.

[CJJ$^+$13] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel Rosu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. Cryptology ePrint Archive, Report 2013/169, 2013. http://eprint.iacr.org/.

[CM05] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Proceedings of the 3rd international conference on Applied Cryptography and Network Security (ACNS)*, 2005.

[DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2010.

[ElG84] Taher ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of the 4th International Cryptology Conference (CRYPTO)*, 1984.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC)*, Bethesda, MD, May–June 2009.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science (FOCS)*, 2013.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *Proceedings of the 32nd International Cryptology Conference (CRYPTO)*, 2012.

[GKP$^+$13a] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *ACM Symposium on Theory of Computing (STOC)*, 2013.

[GKP$^+$13b] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *Proceedings of the 33rd International Cryptology Conference (CRYPTO)*, 2013.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC)*, 1982.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the 19th ACM Symposium on Theory of Computing (STOC)*, 1987.

[Goh] Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume II Basic Applications*. Cambridge University Press, 2004.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13rd ACM Conference on Computer and Communications Security (CCS)*, 2006.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Proceedings of the 33rd International Cryptology Conference (CRYPTO)*, 2013.

[Hal13] Shai Halevi. HElib - an implementation of homomorphic encryption. https://github.com/shaih/HElib, 2013. URL: https://github.com/shaih/HElib.

[HHCW12] Rene Hummen, Martin Henze, Daniel Catrein, and Klaus Wehrle. A cloud design for user-controlled storage and processing of sensor data. In *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2012.

[HKS$^+$10] Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Tasty: Tool for automating secure two-party computations. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, 2010.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In *Proceedings of the 34th International Cryptology Conference (CRYPTO)*, 2014.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In *Proceedings of the 28th International Cryptology Conference (CRYPTO)*, 2008.

[KGM$^+$14] Jeremy Kepner, Vijay Gadepally, Peter Michaleas, Nabil Schear, Mayank Varia, Arkady Yerukhimovich, and Robert K. Cunningham. Computing on masked data: a high performance method for improving big data veracity. *CoRR*, 2014.

[KKM$^+$13] George C. Kagadis, Christos Kloukinas, Kevin Moore, Jim Philbin, Panagiotis Papadimitroulas, Christos Alexakos, Paul G. Nagy, Dimitris Visvikis, and William R. Hendee. Cloud computing in medical imaging. In *Cloud computing in medical imaging*, 2013.

[KPR12] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS)*, 2012.

[KSS09] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *In Cryptology and Network Security (CANS)*, 2009.

[KSS13] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. A systematic approach to practically efficient general two-party secure function evaluation protocols and their modular design. In *Journal of Computer Security*, 2013.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of the 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2008.

[LN14] Tancrede Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes fv and yashe. Technical Report MSR-TR-2014-22, Microsoft Research, January 2014.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2010.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Proceedings of the 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2007.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *J. Cryptol.*, 22:161–188, April 2009. URL: http://dl.acm.org/citation.cfm?id=1530703.1530708, doi:10.1007/s00145-008-9036-8.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*, 2012.

[MNPS04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, pages 287–302, 2004.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Proceedings of the 30th International Cryptology Conference (CRYPTO)*, 2010.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 18th Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 1999.

[PBB09] Raluca Ada Popa, Hari Balakrishnan, and Andrew J. Blumberg. VPriv: Protecting privacy in location-based vehicular services. In *USENIX Security Symposium*, 2009.

[PLZ13] Raluca Ada Popa, Frank H. Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. In *IEEE Symposium on Security and Privacy (IEEE S&P)*, 2013.

[Pop14] Raluca Ada Popa. Building practical systems that compute on encrypted data, 2014.

[PRZB11] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*, 2011.

[PSV^{+}14] Raluca Ada Popa, Emily Stark, Steven Valdez, Jonas Helfer, Nickolai Zeldovich, M. Frans Kaashoek, and Hari Balakrishnan. Building web applications on top of encrypted data using Mylar. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2014.

[Raj12] Remya Rajan. Efficient and privacy preserving multi user keyword search for cloud storage services. In *IJATER*, pages 48–51, 2012.

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2010.

[SSSE14] Julian James Stephen, Savvas Savvides, Russell Seidel, and Patrick Eugster. Practical confidentiality preserving big data analysis. In *HotCloud*, 2014.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *Proceedings of the 6th Theory of Cryptography Conference (TCC)*, 2009.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2005.

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium of Security and Privacy*, 2000.

[TKMZ13] Stephen Tu, M. Frans Kaashoek, Samuel Madden, and Nickolai Zeldovich. Processing analytical queries over encrypted data. In *International Conference on Very Large Data Bases (VLDB)*, 2013.

[TLMM13] Sai Deep Tetali, Mohsen Lesani, Rupak Majumdar, and Todd Millstein. Mrcrypt: static analysis for secure cloud computations. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications (OOPSLA)*, 2013.

[TSCS13] Shruti Tople, Shweta Shinde, Zhaofeng Chen, and Prateek Saxena. AUTOCRYPT: enabling homomorphic computation on servers to protect sensitive web content. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, 2013.

[Vai11] Vinod Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2011.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *FOCS*, pages 160–164, 1982.

[YLW11] Yanjiang Yang, Haibing Lu, and Jian Weng. Multi-user private keyword search for cloud computing. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2011.

[ZE13] Samee Zahur and David Evans. Circuit structures for improving efficiency of security and privacy tools. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (IEEE S&P)*, 2013.

[ZNS11] Fangming Zhao, Takashi Nishide, and Kouichi Sakurai. Multi-user keyword search scheme for secure data sharing with fine-grained access control. In *Proceedings of the 14th international conference on Information Security and Cryptology (ICISC)*, 2011.