Randomly Encoding Functions: a New Cryptographic Paradigm*

Benny Applebaum†

March 25, 2011

Abstract

The notion of *randomized encoding* allows to represent a "complex" function $f(x)$ by a "simpler" randomized mapping $\hat{f}(x;r)$ whose output distribution on an input x encodes the value of $f(x)$. We survey several cryptographic applications of this paradigm.

1 Introduction

To what extent can one simplify the task of computing a function f by settling for computing some (possibly randomized) *encoding* of its output? This question can be formalized as follows: We say that a function $\hat{f}(x;r)$ is a *randomized encoding* (RE) of a function $f(x)$, if its output distribution depends only on the output of f. More precisely, we require the existence of an efficient recovery algorithm Rec and an efficient randomized simulator Sim that satisfy the following conditions:

- (**Correctness**) For every (x,r), given $\hat{f}(x;r)$ the algorithm Rec recovers $f(x)$;
- (**Privacy**) For every x, given $f(x)$ the simulator Sim samples from the distribution of $\hat{f}(x;r)$ induced by a uniform choice of r.

This notion of randomized encoding was introduced by Ishai and Kushilevitz [21] (under the algebraic framework of *randomizing polynomials*) and was implicitly used, in weaker forms, in the context of secure multiparty computation (e.g., [23, 19]). Observe that each of the above requirements alone can be satisfied by a trivial function $\hat{f}$ (e.g., $\hat{f}(x;r) = x$ and $\hat{f}(x;r) = 0$, respectively). However, the combination of the two requirements can be viewed as a non-trivial natural relaxation of the usual notion of computing. This gives rise to the following question: Can we encode "complex" functions f by "simple" functions $\hat{f}$?

It is not hard to show that if one is restricted to *deterministic* encoding the answer is in general negative. For example, let us call a function "simple" if each of its output bits depends on a small constant number of input bits, e.g., 4. In this case, if a boolean function $f : \{0,1\}^n \to \{0,1\}$ can be deterministically encoded by some (possibly non-boolean) simple function $\hat{f}$, then f itself is simple. Indeed if the encoding is deterministic then, by privacy, there is a pair of strings z_0 and z_1

*Invited Survey to International Conference on Information Theoretic Security, 2011.

†School of Electrical Engineering, Tel-Aviv University, benny.applebaum@gmail.com. Supported by Alon and Koshland Fellowships.

such that for every x we have $\hat{f}(x) = z_{f(x)}$. By correctness, z_0 and z_1 should differ in at least a single location i (assuming that f is non-degenerate). Hence, $f(x)$ can be computed by the 4-local function which projects the i-th bit of $\hat{f}(x)$ and, possibly, flips the result. A similar argument holds for any notion of simplicity that is closed under bit-projection and negation.

On the other hand, the use of randomness allows us to encode non-simple functions by simple ones. For example, the sum-function

$$f(x) = x_1 + \ldots + x_n,$$

where x_i is the i-th bit of x and addition is over $\mathbb{F}_2$, can be encoded by the 3-local function

$$\hat{f}(x; (r_1, \ldots, r_{n-1})) = (x_1 - r_1, r_1 + x_2 - r_2, \ldots, r_{n-1} + x_n),$$

which uses $n-1$ random inputs $r = (r_1, \ldots, r_{n-1})$ and outputs n bits. To prove correctness, note that the sum of the output bits of $\hat{f}(x; r)$ equals to $\sum x_i$ as the r_i's cancel out. On the other hand, when r is random, the vector $\hat{f}(x; r)$ is uniformly distributed over all n-bit vectors whose components add to $f(x)$, and so privacy follows.

Perhaps surprisingly, it turns out that REs are powerful enough to encode rich classes of functions. In [21, 22, 4, 3] it is shown that 4-local functions can encode log-space computations, and even poly-time computations if one settles for computational privacy, i.e., the simulator's output is only required to be computationally indistinguishable from $\hat{f}(x; r)$.[1] Similar results hold for other notions of simplicity that will be mentioned later.

The ability to encode complex functions by simple ones is extremely useful. In this short survey we will focus on the applications of REs (and ignore the way REs are constructed). In the next sections we will demonstrate several interesting ways in which this tool can be employed. We consider the archetypal cryptographic setting where Alice and Bob wish to accomplish some computational goal (e.g., a functionality f) at the presence of an adversary. We will see that REs can be beneficial when they are applied to each component of this system: to the functionality, to the honest parties, and even to the adversary.

2 Encoding the Functionality

Delegating computations. Suppose that Bob is a computationally weak device (client) who wishes to compute a complex function f on an input x. Bob is too weak to compute f on his own and so he delegates the computation to a computationally strong server Alice. Since Bob does not trust Alice, he wishes to guarantee the following: (1) Secrecy: Alice should learn nothing on the input x; and (2) Verifiability: Bob should be able to verify the correctness of the output (i.e., a cheating Alice should be caught whp). Similar problems were extensively studied in various settings, originating from the early works on interactive proofs, program checking and instance-hiding schemes (see references in [6]).

Let us start with secrecy and consider a variant where both parties should learn the output $f(x)$ but x should remain private. In this case, a randomized encoding $\hat{f}$ immediately solves the problem via the following single-round protocol: Bob selects private randomness r, computes $\hat{f}(x; r)$ and

[1] The latter requires to assume the existence of log-space computable one-way function, an assumption which is implied by most standard intractability assumptions used in cryptography.

sends the result to Alice who applies the recovery algorithm and outputs the result. The privacy of the RE guarantees that Alice learns nothing beyond $f(x)$. We refer to this protocol as the *basic RE protocol*. Jumping ahead, we note that the protocol has a non-trivial correctness guarantee: even if the server Alice deviates from the protocol and violates correctness she cannot force an erroneous output which violates privacy; that is, it is possible to simulate erroneous outputs solely based on the correct outputs.

It is not hard to modify the basic protocol and obtain full secrecy: instead of encoding f, encode an encrypted version of f. Namely, define a function $g(x, s) = f(x) \oplus s$, where s plays the role of a one-time pad (OTP), and apply the previous protocol as follows: Bob uniformly chooses the pad s and the randomness r, and sends the encoding $\hat{g}(x, s; r)$ of g to Alice, who recovers the result $y = g(x, s) = f(x) \oplus s$, and sends it back to Bob. Finally, Bob removes the pad s and terminates with $f(x)$. (See [3] for more details.)

Achieving verifiability is slightly more tricky. The idea, due to [6], is to combine an RE with a private-key signature scheme (also known as message authentication code or MAC) and ask the server to sign the output of the computation under the client's private key. Here the privacy property of the RE will be used to hide the secret key. Specifically, given an input x, Bob asks Alice to compute $y = f(x)$ (via the previous protocol) and, in addition, to generate a signature on $f(x)$ under a private key k which is chosen randomly by the client. The latter request is computed via the basic RE protocol that hides the private key from Alice. More precisely, Bob, who holds both x and k, invokes an RE protocol in which both parties learn the function $g(x, k) = \mathtt{MAC}_k(f(x))$. Bob then accepts the answer y if and only if the result of the protocol is a valid signature on y under the key k. (The latter computation is typically cheap). The soundness of the protocol follows by showing that a cheating Alice, which fools Bob to accept an erroneous $y^* \neq f(x)$, can be used to either break the privacy of the RE or to forge a valid signature on a new message. For this argument to hold, we crucially relies on the ability to simulate erroneous outputs based on the correct outputs.

The main advantage of this approach over alternative solutions is the ability to achieve good soundness with low computational overhead. For example, $2^{-\tau}$ soundness error introduce an additive overhead of τ in the communication whereas the overhead in competing approaches is multiplicative in τ. (See [6] a more detailed comparison.) Instantiating these approaches with known constructions of REs lead to protocols with an $\mathbf{NC^0}$ client[2] for either log-space functions or poly-time functions depending on the level of security needed (information-theoretic or computational). In fact, in the computational setting we can even reduce the *sequential*-complexity of the client Bob, assuming that he is allowed to invest a lot of computational resources in a preprocessing phase before seeing the actual input x. We also mention that REs can achieve other related properties such as *correctability* [6]: i.e., Bob is able to *correct* Alice's errors as long as Alice is somewhat correct with respect to a predefined distribution over the inputs. In the latter case REs yield $\mathbf{NC^0}$ correctors for log-space computations strengthening the results of [20].

Secure computation [21]. Let us move to a more general setting where the roles of Alice and Bob are symmetric and none of them is computationally weak. The main observation is that instead of securely computing f it suffices to securely compute the randomized encoding $\hat{f}(x; r)$. Indeed, if Alice and Bob learn a sample from $\hat{f}(x; r)$ then they can locally recover the value of $f(x)$

[2]Functions in $\mathbf{NC^0}$ are computable by constant-depth circuits of bounded fan-in, and so they capture a strong notion of constant parallel-time computation.

and nothing else. In other words, the task of securely computing f *reduces* to the task of securely computing a simpler randomized functionality $\hat{f}(x;r)$. As protocol designers, we get a powerful tool which allows us to construct a complex interactive object (protocol) by arguing about a simpler *non-interactive* object (RE).

This paradigm, which was introduced in [21] (and motivated the original definition of REs), yields several new results in the domain of secure computation. As an example, if the algebraic degree of $\hat{f}$ is constant then it can be computed in constant number of rounds [9, 15]. By instantiating this approach with known RE constructions [22, 16], we derive constant-round protocols for boolean or arithmetic log-space functions with information-theoretic security. In the computational setting, this yields a new constant round protocol for poly-time functions [3] providing an alternative construction to the classical protocol of [8].[3] The RE based approach also simplifies the proofs of classical results such as Yao's garbled-circuit protocol [24] and Kilian's completeness theorem [23].

3 Encoding the Primitive: Parallel Cryptography

Suppose now that we already have an implementation of some cryptographic protocol. A key observation made in [4] is that we can "simplify" some of the computations in the protocol by replacing them with their encodings. Consider, for example, the case of public-key encryption: Alice publishes a public/private key pair $(\mathsf{pk}, \mathsf{sk})$; Bob uses the public-key pk and a sequence of random coins s to "garble" a message m into a ciphertext $c = \mathsf{E}(\mathsf{pk}, m, s)$; Finally, Alice recovers m by applying the decryption algorithm to the ciphertext $\mathsf{D}(\mathsf{sk}, c)$. Suppose that Bob sends an encoding of his ciphertext $\widehat{\mathsf{E}}(\mathsf{pk}, m, s; r)$ instead of sending c. This does not violate semantic-security as all the information available to an adversary in the modified protocol can be emulated by an adversary who attacks the original protocol (thanks to the simulator of the RE). On the other hand, Alice can still decrypt the message: first she recovers the original ciphertext (via the recovery algorithm) and then she applies the original decryption algorithm. As a result, we "pushed" the complexity of the sender (encryption algorithm) to the receiver (decryption algorithm).

By employing REs with some additional properties, it is possible to prove similar results for many other cryptographic protocols (e.g., one-way functions, pseudorandom generators, collision-resistance hash functions, signatures, commitments, zero-knowledge proofs) and even information-theoretic primitives (e.g., ε-biased generators and randomness extractors). In the case of stand-alone primitives (e.g., one-way functions and pseudorandom generators) there is no receiver and so the gain in efficiency comes for "free".

Being security preserving, REs give rise to the following paradigm. In order to construct some cryptographic primitive $\mathcal{P}$ in some low complexity class $\mathcal{WEAK}$, first encode functions from a higher complexity class $\mathcal{STRONG}$ by functions from $\mathcal{WEAK}$; then, show that $\mathcal{P}$ has an implementation f in $\mathcal{STRONG}$, and finally replace f by its encoding $\hat{f} \in \mathcal{WEAK}$ and obtain a low-complexity implementation of $\mathcal{P}$. This approach was used in [4, 3, 5] to obtain cryptographic primitives in $\mathbf{NC^0}$ and even in weaker complexity classes. The fact that REs preserve cryptographic hardness was also used to reduce the complexity of cryptographic *reductions* [4, 3] and to reduce the complexity of complete problems for sub-classes of statistical zero-knowledge [18].

[3] The RE based solution requires slightly stronger assumption – one-way function computable in log-space rather in poly-time – but can also lead to efficiency improvements as shown in [17].

4 Encoding the Adversary: Key-Dependent Security

Key-dependent message (KDM) secure encryption schemes [14, 10] provide secrecy even when the attacker sees encryptions of messages related to the secret-key sk. Namely, we say that an encryption is KDM secure with respect to a function class $\mathcal{F}$ if semantic security holds even when the adversary can ask for an encryption of the message $f(\mathsf{sk})$ where f is an arbitrary function in $\mathcal{F}$. Until recently, it was only known how to achieve KDM security for simple linear (or affine) function families [11, 2, 12]. To improve this situation, we would like to have an *amplification* procedure which starts with $\hat{\mathcal{F}}$-KDM secure encryption scheme and boost it into an $\mathcal{F}$-KDM secure scheme, where the function class $\mathcal{F}$ should be richer than $\hat{\mathcal{F}}$. It was recently shown [13, 7] that a strong form of amplification is possible, provided that the underlying encryption scheme satisfies some special *additional* properties. We show [1] how to use REs in order to achieve a *generic* KDM amplification theorem.

Let $f(x)$ be a function and let us view the encoding $\hat{f}(x;r)$ as a *collection* of functions $\hat{\mathcal{F}} = \left\{\hat{f}_r(x)\right\}_r$, where each member of the collection corresponds to some possible fixing of the randomness r, i.e., $\hat{f}_r(x) = \hat{f}(x;r)$. Now suppose that our scheme is KDM secure with respect to the family $\hat{\mathcal{F}}$, and we would like to immunize it against the (more complicated) function f. This can be easily achieved by modifying the encryption scheme as follows: to encrypt a message m we first translate it into the $\hat{f}$-encoding by applying the RE simulator $\mathsf{Sim}(m)$, and then encrypt the result under the original encryption scheme. Decryption is done by applying the original decryption algorithm, and then applying the recovery algorithm Rec to translate the result back to its original form. Observe that an encryption of $f(\mathsf{sk})$ in the new scheme is the same as an encryption of $S(f(\mathsf{sk})) = \hat{f}(\mathsf{sk};r)$ under the original scheme. Hence, a KDM query for f in the new scheme is emulated by an old KDM query for a *randomly chosen* function $\hat{f}_r$. It follows that the KDM security of the new scheme with respect to f reduces to the KDM security of the original scheme with respect to $\hat{\mathcal{F}}$.

This idea easily generalizes to the case where instead of a single function f we have a class of functions $\mathcal{F}$ which are all encoded by functions in $\hat{\mathcal{F}}$. Moreover, the simple structure of the reduction (i.e., a single KDM query of the new scheme translates to a single KDM query of the original scheme) allows to obtain a strong amplification theorem which is insensitive to the exact setting of KDM security, including the symmetric-key/public-key setting, the CPA/CCA cases and the case of multiple-keys. Using known constructions of REs, we can amplify KDM security with respect to linear functions (or even bit-projections) into functions computable by circuits of arbitrary fixed polynomial-size (e.g., n^2).

Acknowledgement. I thank to the conference organizers for inviting this survey, and to Yuval Ishai and Eyal Kushilevitz for introducing me to the notion of randomized encoding and for fruitful and enjoyable collaborations.

References

[1] Applebaum, B.: Key-dependent message security: Generic amplification and completeness theorems. In: EUROCRYPT 2011.

[2] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: CRYPTO 2009.

[3] Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Journal of Computional Complexity 15(2), 115–162 (2006)

[4] Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. SIAM Journal on Computing 36(4), 845–888 (2006)

[5] Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography by cellular automata or how fast can complexity emerge in nature? In: ICS 2010.

[6] Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: Efficient verification via secure computation. In: ICALP (1) (2010), draft of full version available at the authors home page.

[7] Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: EUROCRYPT 2010.

[8] Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: STOC 1990.

[9] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC 1988.

[10] Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: SAC 2002.

[11] Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: CRYPTO 2008.

[12] Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: Quadratic residuosity strikes back). In: CRYPTO 2010.

[13] Brakerski, Z., Goldwasser, S., Kalai, Y.: Circular-secure encryption beyond affine functions. In: TCC 2011.

[14] Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: EUROCRYPT 2001.

[15] Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC 1988.

[16] Cramer, R., Fehr, S., Ishai, Y., Kushilevitz, E.: Efficient multi-party computation over rings. In: EUROCRYPT 2003.

[17] Damgård, I. and Ishai, Y.: Scalable Secure Multiparty Computation. In: CRYPTO 2006.

[18] Dvir, Z., Gutfreund, D., Rothblum, G., Vadhan, S.: On Approximating the Entropy of Polynomial Mappings. In: ICS 2011.

[19] Feige, U., Killian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: STOC 1994 .

[20] Goldwasser, S., Gutfreund, D., Healy, A., Kaufman, T., Rothblum, G.N.: A (de)constructive approach to program checking. In: STOC 2008.

[21] Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: FOCS 2000.

[22] Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: ICALP 2002.

[23] Kilian, J.: Founding cryptography on oblivious transfer. In: STOC 1988.

[24] Yao, A.C.C.: Theory and application of trapdoor functions. In: FOCS 1982.