Functional Encryption Survey

Gus Liu
Stanford University
Department of Computer Science

Abstract

Functional encryption is defined as a special type of encryption that allows a user to learn a function of encrypted data. More specifically, a secure functional encryption system allows a user to compute $f(x)$ from an encryption of x without enabling the user to learn anything more about x. Functional encryption has very recently emerged as an interesting field of cryptography from its ability to provide fine-grained access control to encrypted data. In this survey paper, we investigate different definitions of security of functional encryption, impossibility proofs that arise from these definitions, and the resulting implications.

1 Introduction

Traditionally, public-key encryption has several shortcomings. First, it only provides coarse-grained decryption of data. More specifically, only a unique secret key can be used to decrypt the data. For use cases such as sharing data with groups of users based on their credentials, this can lack functionality. Secondly, public encryption is all-or-nothing, meaning that one either completely decrypts the data or learns nothing about the corresponding plaintext. For medical applications where a particular field of data needs to be extracted while preserving patient confidentiality, a more selective method of computation is required, which gives rise to functional encryption.

Functional encryption is important for any case where a user Alice wants to communicate a function of x to Bob without revealing x. For example, Alice can reveal to Bob the output of an encrypted program without allowing him to learn anything about the program itself. In practice, a company may have an internal tool that only individuals who satisfy particular conditions should be able to decrypt. It may be essential to give access to a simple function of the plaintext, such as showing a written signature on a document image but no information about the contents of the document. However, ensuring security is a non-trivial task, as we will show.

2 Background

We describe more formally the syntactic definition of functional encryption for a functionality F, which is a description of the functions of the plaintext that can be extracted from the ciphertext.

2.1 Setup

Definition 1. A functional encryption scheme for a functionality F defined over (K,X) is a tuple of four PPT algorithms, (setup, key generation, encryption, decryption) that satisfy the satisfy all of the following conditions for all $k \in K$ and $x \in X$:

$$(pp, mk) \leftarrow setup(1^\lambda)$$
$$sk \leftarrow keygen(mk, k)$$
$$c \leftarrow enc(pp, x)$$
$$y \leftarrow dec(sk, c)$$

We require that $y = F(k,x)$ with probability 1.

The setup step generates a public and master secret key pair. The master key is then used to subsequently generate the secret key for k. Next, the message x is encrypted using the public key pp. Finally, $y = F(k,x)$ is computed from c and sk.

2.2 Game-based Security

Now, we define security of this functional encryption scheme. For $b = 0, 1$, we define experiment b for an

adversary A as follows:
1. Setup: run (pp, mk)$\leftarrow$ setup(1^λ) and give pp to A
2. Query: A submits queries k_i in K for $i = 1, 2, ...$ and receives sk$_i \leftarrow$ keygen(mk, k_i).
3. Challenge: A submits two messages m_0 and m_1. He receives $F(k, m_0)$ and $F(k, m_1)$ for which he has the secret key sk. However, we restrict the choices of m_0 and m_1 such that $F(k, m_0) = F(k, m_1)$ necessarily because A has the secret key. A is given enc(pp, m_b).
4. A issues queries and eventually outputs a bit in $\{0, 1\}$.

The scheme is secure if $|\Pr[W_0] - \Pr[W_1]|$ is negligible.

3 Challenges of Functional Encryption

Boneh et. al note that for some cases, the definition of security is too weak. They provide an example where $F(k, x) = \pi(x)$ where π is a one-way permutation and $k = \varepsilon$ is defined to be the empty key. While the correct way to achieve functional encryption is to output $\pi(x)$ on input x, an incorrect approach is to simply output x. However, this incorrect approach satisfies the security game defined in the previous section. We see this with any two values x and y, where $F(\varepsilon, x) = F(\varepsilon, y)$ if and only if $x = y$, and so the attacker is only able to submit messages where $m_0 = m_1$. It is clear that this approach leaks too much information about x, namely x itself. Therefore, they define a simulation based definition of security.

3.1 Simulation-based Security

Definition 2. A functional encryption scheme is simulation-secure if there exists an oracle PPT algorithm Sim = (Sim$_1$, Sim$_O$, Sim$_2$) such that for any oracle PPT algorithms Message and Adv, the two following distributions are computationally indistinguishable:

Real Distribution:
1. (pp, mk) $\leftarrow$ setup(1^λ)
2. ($\vec{x}$, τ) $\leftarrow$ Message$^{\text{keygen}(mk,\cdot)}$(pp)
3. $\vec{c} \leftarrow$ enc(pp, $\vec{x}$).
4. $\alpha \leftarrow Adv^{\text{keygen}(mk,\cdot)}$(pp, $\vec{c}$, τ).
5. Let $y_1, ..., y_l$ be the queries to keygen made by Message and Adv in previous steps.
6. Output (pp, $\vec{x}$, τ, α, $y_1, ..., y_l$).

Ideal Distribution:
1. (pp, σ) $\leftarrow$ Sim$_1(1^\lambda)$
2. ($\vec{x}$, τ) $\leftarrow$ Message$^{SimO(\cdot)[[\sigma]]}$(pp)
3. $\alpha \leftarrow \text{Sim}_2^{F(\cdot,\vec{x}),Adv^O(pp,\cdot,\tau)}(\sigma, F(\varepsilon, \vec{x}))$ 4. Let $y_1, ..., y_l$ be the queries to F made by Sim in previous steps.
5. Output $(pp, \vec{x}, \tau, \alpha, y_1, ..., y_l)$

Essentially, A is allowed to send adaptive queries to the oracle, where for a query q, $B(q, x)$ is executed and (y, x') is returned. The value y returned is sent to A as the response and x is set as x', which is sent to B the next time a query is sent to the oracle.

3.2 Impossibility Results

Boneh et. al present an impossibility proof for simulation-secure functional encryption.

Proof. Message(1^λ) constructs $\vec{x}$: for $i = 1, ..., len_{sk} + \lambda$, the tuple $(r_i, 0)$ where r_i is a randomly and independently chosen bit and len_{sk} is the maximum bit length produced by the keygen algorithm for the key 0 and security parameter λ. τ is empty. Then, $Adv^{\text{keygen}(mk,\cdot)}$(pp, $\vec{c}$, τ) works by calling the random oracle H on (pp, $\vec{c}$) to obtain a string w of length λ. Then, query for the secret key for (w) and then for 0. Finally, use the key for identity 0 to decrypt the whole ciphertext. Output a full transcript of all computations done.

We consider what *Sim* must do to output a distribution indistinguishable from the real interaction. We have that Adv only makes one key query for (w), so *Sim* must do the same. The distinguisher can check if w is the output of H applied to some string of the form (pp, $\vec{c}$). Therefore, the simulator must query this to H before any other queries to F. At this point, the simulator does not know anything about the plaintexts r_i. As a result, the fixed string $z = (pp, \vec{c})$ has the impossible property that after only receiving len_{sk} bits of information, it can deterministically decode z to be an arbitrary string of length $len_{sk} + \lambda$. □

This impossibility result shows that this level of security can be impossible to achieve even in the case of a non-programmable random oracle model where the simulator only has access to the same random oracle given to the distinguisher. Intuitively, this is the case because any simulation-based definition that allows the adversary to query for secret keys after seeing the challenge ciphertext gives the adversary too much power.

We now present another impossibility result[1] that we can compare with the first. This impossibility result shows that general functional encryption under a one message secure, non-adaptive simulation definition is unusable. The lower bound exploits unbounded collusions. A collusion of users is defined as a group of n users who hold secret keys $sk_1, ..., sk_n$ and an encryption of x but cannot learn anything else about x

besides $C_1(x),...,C_q(x)$ for any polynomial q. Here, C is defined as a circuit, or a function.

Theorem 1. There does not exist a functional encryption scheme for the family:

$$C_d(x) = wPRF(x,d)$$

where the input message x is the PRF seed for $wPRF$, a weak pseudo-random function.

Proof. At a high level, the proof shows that the ciphertext in a secure scheme for C_d must grow with the size of the collusion, which is impossible given the requirement to handle unbounded collusions. The key insight is that if an adversary requests q secret keys, denoted $C_{d_1},...,C_{d_q}$ and then requests for an encryption of a random x, then the simulated ciphertext together with the q simulated secret keys construct a description of the values $wPRF(x,d_1),...,wPRF(x,d_q)$, which is indistinguishable from random by definition of the PRF. Using a standard information-theoretic argument, this implies that the ciphertext along with the secret keys must grow with q. In order to obtain a lower bound on the ciphertext size, we use the fact that the simulator must generate the secret keys before it sees the output of $wPRF(x,\cdot)$. Therefore, the simulator has to generate a small ciphertext that reveals information about all the pseudorandom values, which is impossible using a compressibility argument. Thus, we have shown that a weak pseudo-random family is incompressible and that secure schemes only exist for compressible circuit families. □

4 Implications

The impossibility results demonstrate that with the assumption that the adversary is given the power to make an unbounded number of non-adaptive key queries, it is impossible to achieve simulation-secure functional encryption. The proofs use the fact that the size of the ciphertext depends on the collusion bound q. It does not seem possible to construct a scheme where the ciphertext size is independent of q. So where does this lead us in terms of achieving reasonable security for functional encryption? We explore a construction and a new notion of security as ways to help shape what achieving security means for functional encryption.

4.1 Modified Brute Force Construction

We define a random oracle $H : \{0,1\}^* \to \{0,1\}$. We define $s = |K| - 1$ and $K = \{\varepsilon, k_1, ..., k_s\}$. The following brute force functional encryption scheme uses a semantically secure public-key encryption scheme (G,E,D) and has the following steps:

1. Setup(1^λ): for $i = 1,...,s$ run $(pp_i, mk_i) \leftarrow G(1^\lambda)$.
2. Keygen(mk, k_i): output $sk_i := mk_i$.
3. Enc(pp, x): choose random values $r_1,...,r_s \in_R \{0,1\}^\lambda$. Output $\vec{c} := (F(\varepsilon,x), E(pp_1,r_1), H(r_1) \oplus F(k_1,x), ..., E(pp_s,r_s), H(r_s) \oplus F(k_s,x))$.
4. Dec($sk_i, \vec{c}$): output c_0 if $sk_i = \varepsilon$, and output $H(D(sk_i, c_{2i-1})) \oplus c_{2i}$ otherwise.

We want to prove the following theorem.

Theorem 2. Let F be a functionality that reveals functional bit lengths. If (G,E,D) is a semantically secure public-key encryption scheme then the modified brute force system above is simulation-secure in the random oracle model.

Proof. Here, we provide a proof sketch for Theorem 2. We do this by constructing simulators $Sim_1, SimO$, and Sim_2.

1. $Sim_1(1^\lambda)$ executes setup(1^λ) to obtain pp and mk. Output pp and $\sigma = (mk, O, \kappa)$ where O and κ are empty lists. O is used to track the simulated random oracle, and κ to track key queries.

2. $SIM)(\cdot)[[\sigma]]$ responds to random oracle queries and keygen queries that the adversary *Message* makes.

   a. In response to random oracle queries q, the simulator checks if a pair (q,y) exists in O. If it does, it provides y as the response. Otherwise, the simulator chooses a random string y, adds (q,y) to O, and responds with y. O is updated in σ.

   b. In response to a adversary query for key k_i, the simulator sends the secret key mk_i to the adversary. It then adds k_i to κ. κ is updated in σ.

3. $Sim_2^{F(\vec{x},\cdot),Adv(pp,\cdot,\tau)}(\sigma, F(\vec{x},\varepsilon))$ works as follows:

   a. Let m be the number of elements in $\vec{x}$. For $i = 1,...,m$, choose random strings $r_{i,1},...,r_{i,s}$ and $R_{i,1},...,R_{i,s}$, and for $j = 1,...,s$, create the ciphertext components $c_{i,2j-1} = E(pp_j, r_{i,j})$ and $c_{i,2j} = R_{i,j}$. If O already contains any queries to the random $r_{i,j}$'s, abort.

   b. For all keys k_i in κ, the simulator invokes the F oracle and obtains $F(k_i,\vec{x}) = (z_1,...,z_m)$. Add the pairs $(r_{i,1}, R_{i,1} \oplus z_1), ..., (r_{i,m}, R_{i,m} \oplus z_m)$ to O. If any $r_{i,j}$ were in O, abort.

   c. Invoke $Adv(pp,\vec{c},\tau)$ using the "fake" ciphertext created above.

   d. Responds to queries as follows:

      i. Random oracle queries: On query q, the simulator checks if (q,y) is in O. If it does, it provides y as the response. If it doesn't, it checks

if q is equal to any of the random $r_{i,j}$ chosen above. If so, abort. Otherwise, the simulator chooses a random string y, adds (q,y) to O, and responds with y.

ii. Key queries: In response to an adversary query for k_i, the simulator invokes the F oracle and obtains $F(k_i,\vec{x}) = (z_1,..,z_m)$. It then adds the pairs $(r_{i,1}, R_{i,1} \oplus z_1),...,(r_{i,m}, R_{i,m} \oplus z_m)$ to O. Finally, it sends the secret key mk_i to the adversary.

e. When the adversary terminates and outputs α, the simulator outputs α as well.

The rest of the proof is structured as follows. Note that if the above simulation only aborts with negligible probability, then the ideal distribution is statistically close to the real distribution, since (except for the abort condition) the simulation above behaves exactly as the real execution. Thus, we prove that the simulation only aborts with negligible probability by contradiction. We assume that it doesn't. This means that with noticeable probability δ, the adversary queries the random oracle for some $r_{i,j}$ value before asking for the key k_i. We use this to break the one-way security of the public-key encryption scheme used.

Suppose we are given the public key pk for an encryption scheme and a ciphertext $C = E(pk,r)$ for a random value $r \in \{0,1\}^\lambda$. We will construct an attacker that outputs r with probability at least $\frac{\delta}{sM^2}$, where M is some polynomial upper bound on m and the number of random oracle queries that the adversary can make. The attacker runs the simulation described above, but it chooses $i \in [1,s]$ and $j \in [1,M]$ at random ahead of time, and it replaces pp_i with pk and $r_{i,j}$ with r and $c_{i,2j-1}$ with C. If $j > m$, then the attacker aborts. At this point, it randomly picks one of the queries q made by the adversary so far, and outputs this query as its guess for r. It succeeds with probability at least $\frac{\delta}{sM^2}$ by construction, and so we are done. □

4.2 Unbounded Simulation

Agrawal, et al. take a different approach and instead try to re-define what meaningful security is for functional encryption. More specifically, they consider what they call USIM security, where the simulator has unbounded computational power. Recall that a polynomial-time simulation-based security for functional encryption guarantees that against a computationally bounded adversary holding a secret key sk, an encryption of x leaks no more information about x than what an efficient adversary can deduce given $F(x)$. With USIM, security means that an encryption of x leaks no more information about x than what a computationally unbounded adversary can deduce given $F(x)$. The authors argue that this is a very natural approach to security with many parallels to multiparty computation and zero knowledge.

5 Relationship Between Notions of Security

It is important to understand the relationship between different modes of security. Recall that so far, we have defined three types of security. The first is the indistinguishability or game-based definition, which we will refer to as IND. The second is a simulation-based security definition, which we call SIM. Lastly, we have the unbounded simulation-based scheme that we call USIM. We present a theorem that relates these types of security.

Theorem 3. Let FE be a functional encryption scheme. If FE is SIM secure, then it is also USIM secure. In addition, if FE is USIM secure, then it is also IND secure.

Proof. First we prove that SIM $\implies$ USIM. This is trivially obtained because the unbounded simulator can run the polynomial-time simulator.

Now, we prove USIM $\implies$ IND by contrapositive. Let $A = (A_1,A_2)$ be the adversary that breaks IND security of the FE scheme. We construct an adversary $B = (B_1,B_2)$ from A. Here, A is an IND adversary and B is an USIM adversary.

1. Keygen: Run the keygen of A. Answer A_1's key queries using the keygen oracle. Then, A_1 outputs two messages x_0,x_1 and a state st_a. Now, choose a random bit b and output $(x,st) = (x_b,[st_a,x_0,x_1])$.
2. SimO invokes that of A_2 and outputs whatever bit b' it outputs. The output of the real experiment is $(x_b, st = [st_a,x_0,x_1], \alpha = b', C_1,...,C_q)$. Hence, with probability significantly greater than a half, $b = b'$ and the distinguisher can verify this. Now, we claim that there is no unbounded simulator Sim_u for the adversary (B_1,B_2). That is, we argue that Sim_u cannot guess bit b' with probability better than a half. We argue this because the IND adversary $A = (A_1,A_2)$ is admissible, meaning for all queries $\{C'_i\}_{i\in[q']}$ that A makes to B and hence B makes to Sim_u, we have that $C'_i(x_0) = C'_i(x_1)$. The view of Sim_u is statistically independent of the challenge bit b.
Secondly, since the queries $C_1,...,C_q$ are part of the output of the experiment, Sim_u is restricted to only make admissible queries to the function oracle. Otherwise, a distinguisher can easily distinguish between the real and ideal worlds. Thus, it must be that $C_i(x_0) = C_i(x_1)$, $\forall i \in [q]$. Thus, the view of Sim_u is statistically independent of the challenge bit b.

From these two points, we see that the probability the simulator guesses b correctly is at most $\frac{1}{2}$. Therefore, it cannot produce the output indistinguishable from the real experiment. □

6 Conclusion

It is evident that there is still much exploration to be done in either defining new methods of achieving an existing security scheme for functional encryption or defining a new notion of security that is practical. More specifically, one of the largest open problems in functional encryption is to construct a secure scheme for all polynomial-time functionalities. As of now, it is a more practical goal to accomplish security for all polynomial-time predicates. As another possible avenue of research, Agrawal et al. showed that it is worth taking time to develop new lower bounds that can lead to a more practical and realizable security notion, which USIM is a great step towards.

7 References

[1] Agrawal, S., Gorbunov, S., Vaikuntanathan, V., & Wee, H. Functional Encryption: New Perspectives and Lower Bounds.

[2] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: definitions and challenges. In *TCC*, 2011.

[3] Mihir Bellare and Adam O'Neill. Semantically-Secure Functional Encryption: Possibility Results, Impossibility Results and the Quest for a General Definition. Cryptology ePrint Archive, Report 2012/515.