0368.4162 Foundations of Cryptography January 11, 2018

Lecture 12: Functional Encryption

Lecturer: Nir Bitansky

1 Previously on Foundations of Crypto

We've started exploring new cryptographic tools developed in recent years and targeted mostly to protecting various forms of delegated information and computation. Last week, we learned about fully-homomorphic encryption. Today, we'll learn about another powerful tool known as *functional encryption* (FE).

2 Functional Encryption

One limitation of FHE is that encrypted data processed on the cloud will always stay encrypted. In particular, to act upon the result of a computation, the server must send the encrypted $Enc(f(x))$ back to the client for decryption. For instance, consider a spam filter that sits on our email server. We would like the server to only forward non-spam messages, can we do it over encrypted messages? clearly, for the server to avoid sending us spam, it must at least learn that an encrypted message is spam. If it simply runs the spam filter homomorphically, it will only get an encryption of the result, and will not be able to decide.[1] Somehow, we would like that the server would be able to learn if the message was spam, but nothing beyond regarding its contents.

FE is a tool that allows for such fine-grained control of encrypted data. Roughly speaking, it allows generating partial decryption keys sk_f associated with functions f (e.g., a spam filter), so that someone in possession of this key and an encryption of m, can learn $f(m)$ but nothing else regarding m. We will define the public key variant of FE.

Definition 2.1 (Functional Encryption)**.** *A public-key functional encryption scheme consists of PPT algorithms (G, KG, E, D) satisfying:*

- **syntax:**
  - *$G(1^n)$ outputs a master secret key msk and a public key pk.*
  - *$KG(msk, f)$ takes the master secret key msk and circuit $f : \{0,1\}^n \to \{0,1\}^*$ and outputs a function key sk_f.*
  - *$E_{pk}(m)$ given the public key pk and message $m \in \{0,1\}^n$, outputs a ciphertext ct.*
  - *$D_{sk_f}(ct)$ takes a function key sk_f and a ciphertext ct and outputs a value y.*
- **Functionality:** *for any f, $m \in \{0,1\}^n$,*

$$\Pr\left[D_{sk_f}(ct) = f(m) \;\middle|\; \begin{array}{c} msk, pk \leftarrow G(1^n) \\ sk_f \leftarrow KG(msk, f) \end{array}\right] = 1 \ .$$

- **Indistinguishability:** *for any n.u. PPT $A = \{A_n\}$ there is a negligble μ such that it wins the following game with probability at most $1/2 + \mu(n)$:*
  - *A obtains the public key, and can ask for functional keys for f of her choice.*

[1] FHE would still let us save some of the work on the client side, by tagging the email as spam under the encryption, the client would be able to first decrypt only this short tag, and accordingly decide whether to process the rest.

- *A submits two messages $m_0, m_1 \in \{0,1\}^n$ and obtains a ciphertext $ct = E_{pk}(m_b)$ for a random $b \leftarrow \{0,1\}$.*
- *A may ask for more function keys.*
- *A outputs a guess b'.*
- *A wins if $b = b'$ and for all queries f that it made $f(m_0) = f(m_1)$.*

Immediate Application: Verifiable Delegation [PRV12]. So far we've discussed settings where the server was honest-but-curious. Another concern is whether the server computes the prescribed computation at all. Here the client would like to verify that the server computes correctly, but without performing the computation on her own.

Functional encryption gives us an amortized solution for the problem. Say that you have a client that would like to evaluate a circuit $f : \{0,1\}^n \to \{0,1\}$ on many inputs, potentially $\gg |f|$. Then it can do the following:

- Consider the function $g : \{0,1\}^{3n} \to \{0,1\}^n$ that maps

$$(x, r_0, r_1) \longmapsto \begin{cases} r_0 & \text{if } f(x) = 0 \\ r_1 & \text{if } f(x) = 1 \end{cases} .$$

  Generate and send the server a functional key sk_g — takes time $\approx |g| \approx |f|$.[2]
- When the client wishes the server to compute $f(x)$, it chooses random r_0, r_1, and sends the server an encryption ct of (x, r_0, r_1).
- The server cab then use the function key sk_g to return $f(x), r_{f(x)}$, which the verifier can check in time $O(n) \ll |f|$.

To convince the client to accept the wrong result $1 - f(x)$, the server must learn $r_{f(x)}$, however the security of FE prevents her from doing so. Indeed, assume w.l.o.g that $f(x) = 0$ (other case is similar, then since $g(x, r_0, r_1) = r_0$ does not depend on r_1, it holds that

$$sk_g, E_{pk}(x, r_0, r_1) \approx_c sk_g, E_{pk}(x, r_0, 0^n) \ .$$

A Weak FE Scheme from Plain PKE [SS10]. We will now show how to construct an FE scheme that is weak in two aspects:

- **Encryption Scales with Functions.** The scheme will only work for circuits f of a-priori bounded size $s(n)$, the time to encrypt may in particular grow with $s(n)$, and not just with the message length n.
- **Selective Security.** The scheme will satisfy a weaker security notion:
  - A obtains the public key.
  - She then submits submits two messages $m_0, m_1 \in \{0,1\}^n$ and **a single** function f.
  - She obtains a ciphertext $ct = E_{pk}(m_b)$ for a random $b \leftarrow \{0,1\}$, and outputs a guess b'.
  - A wins if $b = b'$ and $f(m_0) = f(m_1)$.

We now construct such a scheme (G, KG, E, D) from a PKE (G', E', D') and a circuit garbling scheme $(CEnc, IEnc, Eval)$. Recall that in a garbling scheme we have a circuit encoder $CEnc$ that given f and a secret key sk outputs an encoding $\hat{f}$ and a local input encoder $IEnc$ that given the secret key sk and

[2] Formally some fixed polynomial in $|f|$.

$(i, x_i) \in [n] \times \{0,1\}$ outputs an encoding $\hat{x}_i$ of the ith input bit.[3] We also have an evaluation algorithm $Eval$ that given a circuit and input encodings $\hat{f}, \hat{x}$, decodes $f(x)$. The security guarantee says that the encodings reveal nothing but the output. That is, there is a simulator SIM such that $\hat{f}, \hat{x} \approx_c SIM(f(x))$.[4]

- $G(1^n)$: for $i, b \in [s] \times \{0,1\}$, sample $(sk_{i,b}, pk_{i,b}) \leftarrow G'(1^n)$. Output $pk = \{pk_{i,b}\}$ and $msk = \{sk_{i,b}\}$.
- $KG(msk, f)$: for $f \in \{0,1\}^s$, outputs $sk_f = \{sk_{i,f_i}\}$.
- $E_{pk}(m)$:
  - Let U_m be a universal circuit that given as input a circuit f, outputs $f(m)$.
  - Sample a secret key for a garbled circuit $gsk \leftarrow \{0,1\}^n$.
  - Compute an encoding $\hat{U}_m \leftarrow CEnc_{gsk}(U_m)$.
  - Compute an encoding for each possible input bit $\{\hat{x}_{i,b} \leftarrow IEnc_{gsk}(i,b)\}_{i,b \in [s] \times \{0,1\}}$.
  - Compute encryptions of the input encodings under the corresponding public keys $ct_{i,b} \leftarrow E'_{pk_{i,b}}(\hat{x}_{i,b})$.
  - Output $ct = \hat{U}_m, \{ct_{i,b}\}_{i,b}$.
- $D_{sk_f}(ct)$: use the keys $\{sk_{i,f_i}\}$ to decrypt corresponding inputs encodings $\hat{x} = \left\{\hat{x}_{i,f_i} \leftarrow D'_{sk_{i,f_i}}(ct_{i,f_i})\right\}_{i \in [s]}$ output $Eval(\hat{U}_m, \hat{x})$.

Claim 2.2. *The above scheme is functional and satisfies selective security.*

Proof Sketch. The functionality of the scheme follows directly from the correctness of PKE and garbling — if everything is performed honestly then the decryptor obtains an input encoding $\hat{x} = IEnc_{gsk}(f)$ and

$$Eval(\hat{U}_m, \hat{x}) = U_m(f) = f(m) \ .$$

We'll now argue that the scheme is selectively secure. Consider an adversary A that submits f, m_0, m_1 such that $f(m_0) = f(m_1)$. We'll show that $(sk_f, E_pk(m_b))$ is computationally indistinguishable from a distribution that doesn't depend on b. We will do this by a hybrid argument. First note that

$$\begin{aligned} sk_f, E_{pk}(m_b) \equiv & \{sk_{i,f_i}\}, \left\{E'_{pk_{i,b}}(\hat{x}_{i,b})\right\}_{i,b}, \hat{U}_{m_b} \approx_c \\ & \{sk_{i,f_i}\}, \left\{E'_{pk_{i,1-f_i}}(\perp)\right\}_i, \left\{E'_{pk_{i,f_i}}(\hat{x}_{i,f_i})\right\}_i, \hat{U}_{m_b} \approx_c \\ & \{sk_{i,f_i}\}, \left\{E'_{pk_{i,1-f_i}}(\perp)\right\}_i, \left\{E'_{pk_{i,f_i}}(\hat{z}_i)\right\}_i, \hat{Z} \qquad \text{where} \qquad (\hat{z}, \hat{Z}) \leftarrow SIM(U_{m_b}(f)) \ . \end{aligned}$$

However $U_{m_b}(f) = f(m_b) = f(m_0)$, and thus the latter distribution doesn't depend on b. □

Note that the constructed encryption scheme is indeed weak:

- The encryption indeed grows with the size s of circuits f (it contains a garbled circuit that has to evaluate f.)
- It is insecure if the adversary can ask for multiple keys. (In fact, even two — if it can ask for a key for f and a circuit g where each bit g_i in its description is $1 - f_i$, then it can decrypt the entire garbled circuit and learn everything.)

[3] We defined the locality more generally, but saw that Yao's garbled circuit has input locality one.

[4] We actually asked that garbling only hides the input x (meaning that the simulator also got f). A function-hiding garbling can be easily obtained from an input hiding one by considering a universal circuit (make sure you understand why). In our construction, it will be slightly more convenient in terms of notation to consider function-hiding garbling.

Beyond Weak Functional Encryption. Can we overcome the two weaknesses described above? it turns out that overcoming any one of them would give a tremendous jump in the power of functional encryption and would make it essentially "complete" for cryptography.

Theorem 2.3 (Informal (and Inaccurate))**.** *Assume selectively-secure FE where either:*

- *the complexity of encryption is independent of $s = |f|$ (depends only on $n = |m|$).*
- *the adversary can ask for any polynomial number of functional keys.*

Then there exists indistinguishability obfuscation.

Indistinguishability obfuscation is an extremely powerful tool that turns out to suffice for basically any crypto task that we know. At this point, we do not know how to achieve it, however, under standard (or even just "nice") assumptions. Indeed, this is also the case for full-fledged FE.[5] We will discuss this further next lecture.

3 FE for Restricted Functions: The Case of IBE

While full-fledged FE has yet to be constructed from standard assumptions, we can construct FE for interesting restrictions of this notion. A well-known special case, introduced more than 20 years before FE [Sha84], is that of *identity-based encryption* (IBE). Here the idea is that instead of remembering the public-keys of all of your contacts, their public-key can be as simple as their identity, or something that determines it, such as their e-mail address.

Such a scheme allows generating identity decryption keys sk_{id}, which can only decrypt encryptions under identity id, and reveal nothing about encryptions for other identities. Indeed, it is a special case of FE for the class of functions

$$f_{id}(id', m) = \begin{cases} m & \text{if } id' = id \\ \perp & \text{if } id' \neq id \end{cases} .$$

The functionality and security definitions accordingly follow from those for general FE.

A property that is crucial for IBE is that the public key is *compact* that is $|pk| \ll 2^{|id|}$, otherwise IBE follows directly from plain PKE (think why).

Bilinear Maps in Cryptography. For more than a decade and a half, it was not known how to construct IBE from the commonly used hard problems in crypto. This naturally led to searching for new hard problems with additional structure.[6] An avenue that turned out to be very successful, and in particular led to IBE, is that of Bilinear maps, which are a generalization of discrete log groups.

In the classical setting of discrete-log groups, we have a group $\mathbb{G}$, say of prime order q (for instance quadratic residues mod a prime $p = 2q + 1$). The group allows to encode elements $x \in \mathbb{Z}_q$ in the exponent as g^x, and conjecture that x is hard to recover as long as it is chosen at random. This hard problem already exhibits some structure; specifically, we can homomorphically compute in the exponent any affine function $L(x) = ax + b$ (for known constants $a, b \in \mathbb{Z}_q$). In particular, if we think about the map $x \mapsto g^x$ as an encryption of x then it already allows to learn some function of x, in fact for any $g^{x_1} \dots g^{x_n}$ and affine $L(x_1, \dots, x_n)$ we can test if $L(x_1, \dots, x_n) = 0$ (simply by comparing to g).

What about more complex functions, say quadratic functions $Q(x)$. This is generally not possible in discrete log groups and in fact under the common Diffie-Hellman assumption, which says that given g^x, g_y for random x, y it is hard to compute g^{xy}. Bilinear groups essentially extend the concept of discrete-log

[5] There are settings of FE that are somewhat in between and can be achieved under standard assumptions. For instance, under LWE it is known how to construct single-key FE where the complexity of encryption only scales with the output size, and not necessarily the circuit size [GKP+13].

[6] By now, it is already known how to construct IBE from a variety of assumptions, like LWE, LPN, CDH, and Factoring.

groups to also support quadratic functions. These are groups $\mathbb{G}$ associated with another group $\mathbb{H}$ (known as the target group) and a (bilinear) map

$$(g^x, g^y) \stackrel{e}{\longmapsto} h^{xy} \ ,$$

where g and h are some fixed generators of $\mathbb{G}$ and $\mathbb{H}$.

This in, particular allows encoding elements $g^{x_1}, \ldots g^{x_n} \in \mathbb{G}$ and quadratic $Q(x_1, \ldots, x_n)$ to test whether it is zero, indeed any such Q could be written as a linear function L in the variables $y_{ij} = x_i x_j$ and we can use the mapping to compute each $h^{y_{ij}} = e(g^{x_i}, g^{x_j})$ and then test if $L((y_{ij} : i, j \in [n])) = 0$ in $\mathbb{H}$. We can then consider corresponding hard problems such as the bilinear Computational Diffie-Hellman assumption (BCDH) that says that given g^x, g^y, g^z for random x, y, z it is hard to find h^{xyz}, or even that h^{xyz} is pseudorandom which is the decision version of the problem (known as BDDH).

Finding candidate groups that have such a mapping and where problems like BCDH may hold is not an easy task, and there are no easy to describe examples such as in the case of discrete-log groups. Known candidates are based on Weil/Tate pairings in certain Eliptic curve groups, and describing them is outside the scope of this course.

Constructing IBE from Bilinear groups. Here we'll see how to construct IBE from such groups with BDDH under a weak security definition where the adversary obtains the identity key sk_{id} for polynomially many identities **chosen at random** and then tries to distinguish encryptions under another random identity id^*.

- $G(1^n)$: sample $x \leftarrow \mathbb{Z}_q$ and output $pk = g^x$ and $msk = x$.
- $KG(msk, id)$: interpret $id \in \mathbb{G}$ and output id^x.
- $E_{pk}(id, m)$: sample z, and output $ct = (g^z, e(id^z, g^x) \cdot m)$, where we interpret $m \in \mathbb{H}$.
- $D_{sk_{id}}(ct)$: parse sk_{id} as $A \in \mathbb{G}$ and ct as $B, C \in \mathbb{G} \times \mathbb{H}$ and , output $C/e(A, B)$.

To see correctness, let $id = g^y$, note that when trying to decrypt $E_{pk}(id, m)$ with a matching sk_{id}

$$C/e(A, B) = e(g^{yz}, g^x) \cdot m/e(g^{yx}, g^z) = h^{xyz} m/h^{xyz} = m \ .$$

To prove security, consider an adversary A that

- is given random (w.l.o.g distinct) identities $id_1, \ldots, id_t, id^*$ and the keys $id_1^x, \ldots, id_t^x$
- it chooses $m_0, m_1 \in \mathbb{H}$, and obtains $g^z, e(g^x z, id^*) m_b$ for a random b
- guesses b with probability $1/2 + \varepsilon$.

Then we can break BDDH with advantage ε. Our reduction given g, g^x, g^y, g^z, T emulates A as follows:

- It chooses $y_1, \ldots, y_t \leftarrow \mathbb{Z}_q$ and gives the identities $g^{y_1}, \ldots, g^{y_t}, g^y$ and the keys $g^{xy_1}, \ldots, g^{xy_t}$ (which it can compute using the y_i's) to A.
- A produces m_0, m_1, and the reduction gives it back $g^z, T m_b$ for a random $b \leftarrow \{0, 1\}$.
- The reduction returns 1 if and only if A guesses b correctly.

Note that if $T = g^{xyz}$ then A's emulated view is identical to her view in the real scheme, implying that she will guess b with probability $1/2 + \varepsilon$, whereas if T is random the A can guess b with probability $1/2$.

From Random Identities to Arbitrary Ones via Random Oracles. A common way to remove the assumption on random identities is to consider a model where all parties have access to a random oracle R. In this case, we can adapt the construction so that the instead of exponentiating id directly we exponentiate $R(id)$. Of course that the real world does not have random oracles, and the random oracle is heuristically replaced with some specific hash functions, such as SHA256. While such instantiation won't have a reduction to any standard assumption, in reality it is still not known how to break it. Although by now we also have schemes that do not rely on random oracle, in reality the above scheme is still more common because it is simple and relatively efficient.

References

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564, 2013.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 422–439, 2012.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 47–53, 1984.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 463–472, 2010.