COS 533: Advanced Cryptography — Princeton University
Lecture 13 (March 23, 2021)
Lecturer: Mark Zhandry — Scribe: Frederick Qiu

Notes for Lecture 13

1 Introduction to Traitor Tracing

1.1 The Problem

Imagine a scenario where a content distributor broadcasts public key encrypted content, and users can decrypt the content using secret keys provided by the distributor. The distributor would like to deter users from leaking their secret key to others, which would allow anyone with the key to decrypt the content.

Users who leak keys are known as *traitors*. The idea behind *traitor tracing* is to deter piracy by making it so that the act of leaking a secret key reveals the identity of the traitor.

1.2 Difficulties

A naïve approach might be as follows: generate a pair $(\mathsf{pk}, \mathsf{sk})$, and distribute to user i the secret key (sk, i).

The problem, of course, is that only sk is needed to decrypt, not i, so a user could just leak sk and avoid any possibility of tracing. Worse, a user i could frame a user j by leaking the pair (sk, j).

One can think of ways to solve these problems (e.g. use a signature using i instead of using i directly, use identity-based encryption so that the secret key can't be dismantled, etc.), but the naïve approach reveals some deeper issues. Namely, a traitor doesn't necessarily need to leak their secret key directly to cause damage to the content provider.

Instead, traitors can leak a *decoder*, which is a program that decrypts using leaked secret keys (we need to account for the multiple leaked keys setting since traitors may collude). This gives the traitor additional privacy in the sense that they can use cryptographic tools to obfuscate the internal workings of their program, and hence obfuscate any direct use of the leaked secret keys.

2 Formal Setting

2.1 Definitions

Formally, a traitor tracing scheme is defined with the following operations:

$$\begin{aligned}(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) &\leftarrow \mathsf{Gen}(1^\lambda, 1^N)\\ c &\leftarrow \mathsf{Enc}(\mathsf{pk}, m)\\ m &\leftarrow \mathsf{Dec}(\mathsf{sk}_i, c)\\ A &\leftarrow \mathsf{Trace}^D(\mathsf{pk}, \varepsilon, m_0, m_1)\end{aligned}$$

Where N is the number of users (which will be polynomially bounded in our setting), A is the *accused set*, and D is some decoder. We will discuss the exact purpose of ε later.

Before formally defining security, consider some intuitive properties we would like:

- We don't want to accuse honest users, i.e. a user whose secret key has not been leaked should not be included in A.
- For any good decoders, we want to accuse at least one user (not necessarily all dishonest users), i.e. A should be non-empty when given a useful decoder.
- We want a broad definition of "good decoder." That is, even if the decoder fails some or most of the time, or if the decoder doesn't recover the entire message (e.g. a decoder for a streaming service that recovers a lower definition video), the previous property holds.

When we formally introduce the security game in the next section, we use the most general notion of a good decoder: any decoder D that breaks CPA security for some messages m_0 and m_1. Then, we can think of ε as a threshold we set for the distinguishing advantage D needs before we consider it to be a good decoder.

Remark 1 *We can also define secret key traitor tracing schemes where* Trace *takes more key inputs than just* pk *(e.g. it takes some* sk_i *or some special tracing key). However, these are generally less useful.*

One reason is if the additional keys are only known to the distributor, then only the distributor can traitor trace, whereas Trace *above can be run by anyone.*

Further, if Trace *takes secret keys as part of its input, if a distributor wanted to prove the identity of a traitor (in court, for example), the distributor has to either reveal the secret key inputs to* Trace*, or introduce additional complexity to* Trace *so that it also outputs a proof of* A*.*

2.2 The Security Game

1. The adversary Adv sends the challenger Ch the number of users 1^N.
2. Ch runs $\mathsf{Gen}(1^\lambda, 1^N)$ and sends Adv the public key pk.
3. Adv sends Ch queries of $i \in [N]$, and for each query, Ch sends Adv the secret key sk_i. Let S denote the set of all queried i.
4. Ch sends Adv a decoder D and messages m_0, m_1.
5. Ch runs $\mathsf{Trace}^D(\mathsf{pk}, \varepsilon, m_0, m_1)$.

Perhaps it is unrealistic to let the adversary choose the number of users in step 1, but this is to ensure that no scheme has a weakness when the number of users is some special number. The repeated queries in step 3 represent potential collusion between traitors to construct a decoder. One can intuitively think of step 4 as the adversary giving the challenger a decoder, as well as "proof" that the decoder is good, i.e. the pair of messages for which D breaks CPA security.

Define the following events:

- $\mathsf{GoodDec}_\varepsilon$: event that D distinguishes between $\mathsf{Enc}(\mathsf{pk}, m_0)$ and $\mathsf{Enc}(\mathsf{pk}, m_1)$ with probability at least ε.
- $\mathsf{GoodTr}_\varepsilon$: event that $A \neq \emptyset$.
- $\mathsf{BadTr}_\varepsilon$: event that $A \not\subseteq S$.

A traitor tracing scheme is secure if for all PPT adversaries and all non-negligible ε,

$$\Pr[\mathsf{BadTr}_\varepsilon] \quad < \quad \mathsf{negl} \tag{1}$$

$$\Pr[\mathsf{GoodTr}_\varepsilon] \quad \geq \quad \Pr[\mathsf{GoodDec}_\varepsilon] - \mathsf{negl} \tag{2}$$

We can view (1) as the condition that we don't accuse honest users, and (2) as the condition that we accuse at least one user whenever the decoder is good.

Remark 2 *One might notice that (2) is perhaps not precisely what we want; it might be more accurate to stipulate that* $\Pr[\neg\mathsf{GoodTr}_\varepsilon \wedge \mathsf{GoodDec}_\varepsilon] < \mathsf{negl}$. *As written above, a traitor tracing scheme would count as secure even if the outcomes where we have a good trace and a good decoder don't overlap, but the probabilities of each occurring are roughly equal. However, (2) is what usually appears in the literature, so we use it here as well.*

3 Schemes

3.1 A Simple Scheme

$\mathsf{Gen}(1^\lambda, 1^N)$: $(\mathsf{pk}'_i, \mathsf{sk}'_i) \leftarrow \mathsf{Gen}_{\mathrm{PKE}}(1^\lambda)$
$\mathsf{pk} = (\mathsf{pk}'_1, \ldots, \mathsf{pk}'_N)$
$\mathsf{sk}_i = \mathsf{sk}'_i$

$\mathsf{Enc}(\mathsf{pk}, m)$: $(c_1, \ldots, c_N), \quad c_i = \mathsf{Enc}(\mathsf{pk}_i, m)$

$\mathsf{Dec}(\mathsf{sk}_i, c)$: $\mathsf{Dec}(\mathsf{sk}_i, c_i)$

$\mathsf{Trace}(\mathsf{pk}, \varepsilon, m_0, m_1)$: Construct hybrids D_i that return $(c'_1, \ldots, c'_N)$, where
$c'_j = \mathsf{Enc}(\mathsf{pk}_j, m_0)$ if $j \leq i$
$c'_j = \mathsf{Enc}(\mathsf{pk}_j, m_1)$ if $j > i$
Define $p_i = \left| \Pr[1 \leftarrow D(D_i)] - \Pr[1 \leftarrow D(D_{i-1})] \right|$
Approximate p_i and return $A = \{i : p_i \geq \varepsilon/(2N)\}$

3.2 Security Proof

First, note that by security of public key encryption and the fact that all $(\mathsf{pk}'_i, \mathsf{sk}'_i)$ are independently generated, the scheme is a secure with regards to encryption and decryption.

Next, observe that we only accuse user i when our approximation p_i of the distinguishing advantage of D between D_i and D_{i-1} is non-negligible. There are two ways this can occur: either D's distinguishing advantage is actually non-negligible, or our approximation is wildly inaccurate. We take for granted that we can approximate to arbitrary precision such that the error is negligible.

Suppose that Adv did not query i. Then the actual distinguishing advantage of D between D_i and D_{i-1} is non-negligible only with negligible probability, or else D would violate the security of public key encryption.

Since the probability of the approximation p_i being wildly inaccurate is negligible, and the probability of D distinguishing (with non-negligible advantage) between D_i and D_{i-1} is negligible (when user i is honest), security property (1) holds.

Now, suppose that D is a good decoder. This means that it distinguishes between D_0 and D_N with advantage ε, which implies it distinguishes between some D_i and D_{i-1} with advantage at least ε/N. It can be shown that the probability that we

approximate p_i to be less than $\varepsilon/(2N)$ is negligible. Thus, whenever D is a good decoder, Trace accuses some user i except when the approximation of p_i is wildly inaccurate, which occurs with negligible probability, so security property (2) holds.

3.3 Bounds for Other Known Schemes

An obvious problem with the simple scheme above is the size of the ciphertexts (and also the public key); every time we want to encrypt something, we need to spend $O(N)$ time, and transmit a message of $O(N)$ length.

The main goal of traitor tracing literature is to reduce the parameter overhead, that is, the length of the ciphertext, keys, or both.

3.3.1 Combinatorial Methods

Using an information theoretic object known as *fingerprinting codes*, along with public key encryption, we can get the size of ciphertexts down to $O(1)$ (the constants here are good as well, only twice the length of a regular public key encryption scheme). However, the size of the secret keys and public key increase to $O(N^2)$ (the constants here are also quite large, making the scheme impractical). The general idea is to give each user some subset of N^2 secret keys (where the fingerprinting codes describe how to do so), and then encryption only requires encrypting using two public keys.

There is also middle-ground between the simple scheme and the fingerprinting codes scheme, where by interpolating the two schemes, we can get public keys of size $O(N^{2-A})$, secret keys of size $O(N^{2-2A})$, and ciphertexts of size $O(N^A)$. Letting $A = 0$ gives us the fingerprinting codes scheme, and letting $A = 1$ gives us the simple scheme.

Also note that we can do better in the *bounded collusion* case, where we know that at most k secret keys will be involved in the making of any decoder. If $k = 1$, this is similar to the case of identity-based encryption, and extending that allows us to get ciphertexts of size $O(k)$. Schemes for the bounded collusion case can generally be used in practice.

3.3.2 Algebraic Methods

Using pairings we can get public keys and ciphertexts of size $O(N^{1/2})$ and secret keys of size $O(1)$. The intuition is that the pairing allows us to compress the information in the public key and take combinations of pairs to support traitor tracing on all N users. Bounds can also be "traded off" similarly to in the fingerprinting codes scheme, giving us public keys, ciphertexts, and secret keys of size $O(N^{1/3})$. This $O(N^{1/3})$ bound is only obtainable when using secret key tracing.

Using LWE, we can get public keys, ciphertexts, and secret keys of size $O(1)$. This bound is also only obtainable when using secret key tracing.

The constants for the $O(N^{1/3})$ pairing scheme and $O(1)$ LWE scheme are large enough to render the schemes impractical, though the constants for the $O(N^{1/2})$ pairing scheme are not so bad.