CS7111: Topics in Cryptography October 9, 2019

Lectures 15: Witness Encryption

Lecturer: *Shweta Agrawal* Scribe: *Simran*

1 Introduction

In this lecture, we study witness encryption (WE) introduced by [GGSW13]. To understand the idea behind WE, let us start with a hypothetical scenario, as described in [GGSW13]. Suppose there is a billionaire, who loves to solve math puzzles and wants to award a huge cash price to the one who solves some hard problem, say $\mathsf{P} \overset{?}{=} \mathsf{NP}$. For this, the billionaire generates an instance of the problem and decides to award whoever solves the problem. Since the problem instance is supposedly hard, the billionaire might not be alive by the time someone ever solves it. To tackle with this, he or she hides the prize amount at some place and encrypts the address of that place in such a way so that only that person who solves the problem can decrypt it. The goal of WE is to help the billionaire with encrypting the address.

We know that for any NP language $L \subset \{0,1\}^*$, $\exists R$ such that $x \in L \iff \exists w : (x, w) \in R$. Thus, solving the instance of this problem implies finding a witness w for the problem. For convenience, let us assume $|w|$ is some a-priori known polynomial in the security parameter. We can then have a reduction construct a circuit C that takes w as input and outputs 1 if w is a valid witness for x and 0 otherwise. Accordingly, the ciphertext in WE is to be decrypted using such a witness w. We now formally define a WE scheme below.

2 Witness Encryption

A witness encryption (WE) scheme for an NP language L consists of the following two PPT algorithms:

- **Enc**(C, m): It takes as input the circuit C and $m \in \{0, 1\}$ and outputs a ciphertext CT.
- **Dec**(C, CT, w): It takes as input the circuit C, the cipertext CT and a witness w and outputs m if $C(w) = 1$, and $\perp$ otherwise.

These algorithms satisfy the following two conditions:

- **Correctness**: If $\exists w$ such that $C(w) = 1$, then $Dec(C, CT, w)$ outputs m.
- **Security**: If $\nexists w$ such that $C(w) = 1$, then the ciphertexts encoding 0 and 1 are computationally indistinguishable, i.e. $Enc(C, 0) \overset{c}{\approx} Enc(C, 1)$.

3 An NP-Complete Language

Here we will design an encryption scheme for an NP-complete language L. In particular, we will consider the NP-complete problem *exact set cover*.

3.1 Exact Set Cover

Given $x = (n, S_1, ..., S_l)$ where $n \in \mathbb{N}$ and $S_i \subseteq [n], \forall i \in [l]$,we have to find a subset of indices $T \subseteq [l]$ such that:

1. $\bigcup_{(i \in T)} S_i = [n]$.
2. $\forall i, j \in T$ such that $i \neq j$, we have $S_i \cap S_j \neq \phi$.

If such a T exists, we say that T is an exact set cover of x.

3.2 Multilinear Maps

For security parameter λ, let $G_1, G_2, ..., G_n$ be groups each of large prime order $p > 2^\lambda$, and let $G_i = \langle g_i \rangle$. A multilinear map e is a set of bilinear maps $\{e_{i,j} : G_i \times G_j \to G_{i+j} | i, j \geq 1; i+j \leq n\}$, such that for any $a, b \in \mathbb{Z}_p$, $e_{i,j}$ satisfies the relation: $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$.

3.3 The n-MDDH Assumption

Let $G_1, G_2, ..., G_n$ be groups each of prime order p and let g_i be the generator of G_i. For $s, c_1, ..., c_n \leftarrow \mathbb{Z}_p$,the assumption states the following:

Given $g_1, g^s, g_1^{c_1}, ..., g_1^{c_n}$, it is hard to distinguish $T = g_n^{s \prod_{i \in [n]} c_i}$, $(T \in G_n)$ from a random element in G_n with non-negligible probability.

3.4 Decision Multilinear No-Exact-Cover Assumption

Let $x = \{S_i : S_i \subseteq [n], i \in [l]\}$ be an instance of Exact Cover that has no solution. Let $params \leftarrow G(1^{\lambda+n}, n)$ be a description of a multilinear group family with order prime $p = p(\lambda)$. Sample uniformly random elements $a_1, ..., a_n, r \in \mathbb{Z}_p$.
For $i \in [l]$, let $c_i = g_{|S_i|}^{\prod_{j \in S_i} a_i}$.

Decision Multilinear No-Exact-Cover Problem : Distinguish between the two distributions:

$$(params, c_1, \ldots, c_l, g_n^{a_1 \cdots a_n}) \text{ and } (params, h_1, ..., h_l, g_r^n).$$

The Decision Multilinear No-Exact-Cover Assumption is that for all adversaries $\mathcal{A}$, there exists a fixed negligible function μ such that for all instances x with no solution, $\mathcal{A}$s distinguishing advantage against the Decision Multilinear No-Exact-Cover Problem for x is at most $\mu(\lambda)$.

3.5 The Encryption Scheme

Encrypt(x, m): It takes as input an Exact Cover instance x and a message $m \in \{0, 1\}$

- Sample $a_1, ..., a_n$ uniformly at random from $\mathbb{Z}_p$
- $\forall i \in [l]$, let $c_i = g_{|S_i|}^{\prod_{j \in S_i} a_i}$.
- Sample uniform $r \leftarrow \mathbb{Z}_p$
- If $m = 1$, let $d = g_n^{\prod_{j \in [n]} a_i}$, else $d = g_n^r$
- Output ciphertext $ct = (d, c_1, c_2, ..., c_l)$

Decyrpt(ct, T): It takes as input a ciphertext ct and a witness of the exact cover T, where $T \subseteq [l]$

- Compute $c = \prod_{i \in T} c_i$.
- If $c = d$ output 1 else output 0.

Correctness: Suppose the exact cover has a witness, say T. Then we have,

$$\prod_{i \in T} \prod_{j \in S_i} a_j = \prod_{i \in [n]} a_i$$

$$c = \prod_{i \in T} c_i = \prod_{i \in T} g_{|S_i|}^{\prod_{j \in S_j} a_j} = g_n^{\prod_{i \in n} a_i}$$

So we have $d = c$ which implies correctness.

Security: The scheme above is a sound witness encryption scheme under the Decision Multilinear No-Exact-Cover Assumption. Suppose that the exact cover problem has no witness, then if we take any subset of indices T' such that $\cup_{i \in T'} S_j = [n]$ we have

$$\sum_{i \in T'} |S_i| > n$$

so we may not even land in the target group g_n. The only way to compute $g_n^{\prod_{j \in [n]} a_i}$ is to find a witness for exact cover.
So it follows from the Decision Multilinear No-Exact-Cover assumption that no ppt adversary can distinguish $g_n^{\prod_{j \in [n]} a_i}$ $(m = 1)$ from g_n^r $(m = 0)$ with non-negligible advantage.

References

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 467–476. ACM, 2013.