Sparse Graph Obfuscation

September 26, 2023

1 Preliminaries

1.1 Bounded Functional Encryption

We will use the notation of static, bounded functional encryption as presented in [AR17].

Security

We will slightly weaken the security notion such that the adversary does not choose which circuits it can learn the functional secret key for. Indeed, this is a weaker notion of functional encryption which fixes the adversary's output circuit. We will assume that we get circuit $C_1, \ldots, C_d$.

See page 8 of [AR17] for now. I'll put in the actual definition later.

Algorithm 1 $\mathrm{Exp}^{\mathrm{real}}_{\mathcal{F},\mathcal{A}}(1^\lambda)$

1: $(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathrm{FE.Setup}(1^\lambda)$
2: $\mathrm{SK}_{C_i} \leftarrow \mathrm{FE.Keygen}(\mathrm{MSK}, C_i)$ for $i \in [d]$
3: $x_i \leftarrow \mathcal{A}(\mathrm{SK}_{C_i})$
4: $\mathrm{CT}_i \leftarrow \mathrm{FE.Enc}(\mathrm{MPK}, x_i)$
5: $\alpha \leftarrow \mathcal{A}(\mathrm{CT}_1, \ldots, \mathrm{CT}_d)$
6: **return** $x_1, \ldots, x_d, \alpha$

Algorithm 2 $\mathrm{Exp}^{\mathrm{ideal}}_{\mathcal{F},\mathrm{Sim}}(1^\lambda)$

1: $(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathrm{FE.Setup}(1^\lambda)$
2: $\mathrm{SK}_{C_i} \leftarrow \mathrm{FE.Keygen}(\mathrm{MSK}, C_i)$ for $i \in [d]$
3: $x_i \leftarrow \mathcal{A}(\mathrm{SK}_{C_i})$
4: $\mathrm{CT}_i \leftarrow \mathrm{Sim}(1^\lambda, 1^{|x_i|}, \mathrm{MPK}, C_i, \mathrm{SK}_{C_i}, C_i(x_i))$
5: $\alpha \leftarrow \mathcal{A}(\mathrm{CT}_1, \ldots, \mathrm{CT}_d)$
6: **return** $x_1, \ldots, x_d, \alpha$

Note that the adversary $\mathcal{A}$ and simulator are stateful but we do not include this in the above notation for simplicity.

2 A sketch for the boys

2.1 Graph Label Randomization in ROM

Say that we have a sparse graph $\mathcal{G} = (V, E)$ such that $|V| = n$ and $\forall v \in V, \deg(v) = d$. (TODO: padding).

Then, we want to create a pseudo-randomized label mapping of the graph, $\phi : \{0,1\}^\lambda \times V \to \{0,1\}^{c\cdot\lambda}$ such that ϕ is deterministic and pseudo-random. In particular, we require that for an adversary that does not know a path from v to $u \in \{v_1, ..., v_p\}$ or u to v where $v \neq u$, then for $K \xleftarrow{\$} \{0,1\}^\lambda$,

$$\mathbf{Pr}\left[\mathcal{A}(v, \phi(K, v_1), \ldots \phi(K, v_p), v_1, \ldots, v_p, C_\Gamma) = \phi(K, v)\right] \leq \mathtt{negl}(\lambda) \tag{1}$$

where C_Γ is the neighbor function for the embedded space: i.e. $C_\Gamma = \phi \circ \Gamma \circ \phi^{-1}$.

2.2 The Construction

Algorithm 3 The circuit for the neighbor function, C_Γ.

---

function $\texttt{INNER}_i(\texttt{Dec}(\phi(v)) = v, K)$
    $u_1, \ldots, u_d = \Gamma(v)$
    $u = u_i$
    $r = H(K, u)$
    return FE.Enc(MPK, (u, K)) where we encrypt with randomness from r.
function $C_\Gamma(\phi(v))$
    for $i \in [d]$ **do**
        $u_i = \text{FE.Dec}(\text{SK}_{\texttt{inner}_i}, \phi(v))$
    return $(u_1, \ldots, u_d)$

Claim 2.1. *eq. (1) holds for a given vertex v for any PPT adversary, $\mathcal{B}$ when C_Γ is implemented as in algorithm 3.*

In order to prove the above, we must show that an adversary which does not know a path from v to $v_1, \ldots v_p$ essentially "learns" nothing of K.

Lemma 2.2 (Does not learn K)**.** *For any PPT adversary $\mathcal{A}$*

$$\mathbf{Pr}[\mathcal{A}(\phi(K, v_1), \ldots, \phi(K, v_p), v_1, \ldots, v_p, C_\Gamma) = K] \leq \mathtt{negl}(\lambda).$$

Proof. We proceed via a series of hybrids to build two computationally indistinguishable distributions. We then show that assuming $\mathcal{A}$ can output K with non-negligible probability, we can distinguish the two distributions. First, let $K' \xleftarrow{\$} \{0,1\}^\lambda$.

Then, we have the following hybrids

- $\texttt{Hyb}_0$: The adversary plays the game outlined in algorithm 1 where the circuits are $\texttt{inner}_1, \ldots, \texttt{inner}_d$ and encryptions of (K, v_ℓ) are done with true randomness

- $\texttt{Hyb}_1$: As the above except that we replace the real protocol with the simulated one, algorithm 2

- $\texttt{Hyb}_2$: As the above except that we invoke the ROM to replace r with fixed randomness for each u

- $\texttt{Hyb}_3$: As the above except that we replace $\texttt{inner}_i$ with $\texttt{inner}'_i$ where $\texttt{inner}'_i$ invokes the FE simulator, Sim when generating its output. Note that Sim must know the access pattern of $\mathcal{A}$ to $\texttt{inner}_i$ in order to simulate the output of $\texttt{inner}_i$. So, we replace $\texttt{inner}'_i$ working backwards from the last call to the simulator to the first. As the simulator is stateful, we can see that the simulator knows the access pattern.

- $\texttt{Hyb}_4$: As the above except that we replace K with K'. We can see that this is valid as both inputs to $\texttt{inner}'_i$ and outputs of $\texttt{inner}'_i$ are simulated and thus independent of K.

Now, we show that we can replace the true randomness used with randomness derived from a random oracle

- $\texttt{Hyb}^a_5$: As $\texttt{Hyb}_4$ except that we fix any randomness used by the simulator to be $H(K', \cdot)$ generated randomness.

- $\texttt{Hyb}^a_6$: As the above except that we replace $\texttt{inner}'_i$ with $\texttt{inner}_i$.

- $\texttt{Hyb}^a_7$: As the above except that we replace the simulated version with the real protocol, except that this time we are working over randomness generated by K'

We now do something similar to the above except we swap K and K'

- $\texttt{Hyb}^b_5$: As $\texttt{Hyb}_0$

- $\texttt{Hyb}^b_6$: As the above except that we replace the true randomness with $H(K, \cdot)$ generated randomness.

We now have that $\texttt{Hyb}^a_8 \approx \texttt{Hyb}^b_6$. Assume that $\mathcal{A}$ can output K with non-negligible probability. Then, we can build $\mathcal{A}'$ to distinguish $\texttt{Hyb}^a_8$ and $\texttt{Hyb}^b_6$. $\mathcal{A}'$ proceeds as follows:

1. Call $\mathcal{A}(\cdot)$ to get K

2. Check whether $C_\Gamma(\phi(K, v_\ell)) = C_\Gamma(\text{FE.Enc}(K, \Gamma(v_\ell)))$ If the above equality holds, output 1, otherwise, output 0.

Assume that with probability $\alpha > \texttt{negl}(\lambda)$, $\mathcal{A}$ gives the correct K. Then, because K' is random, we have that with high probability $C_\Gamma(\phi(K, u_i)) \neq C_\Gamma(\phi(K', u_i))$. Thus, $\mathcal{A}$ can distinguish the two hybrids. □

Now we can prove Claim 2.1,

Proof of Claim 2.1. First note that the adversary cannot learn $\phi(K, v)$ via calling FE.Enc(K, v) as that would imply the ability to construct an adversary that can output K, breaking lemma 2.2.

Thus, $\mathcal{B}$ can only learn $\phi(K, v)$ via calling C_Γ or manipulating given cipher texts. We now proceed to show that this is computationally infeasible via a hybrid algorithm.

- $\texttt{Hyb}_0$: The adversary plays the game outlined in algorithm 1 where the circuits are $\texttt{inner}_1, \ldots, \texttt{inner}_d$ and $\mathcal{A}$ is given $\phi(K, v_1) = \text{FE.Enc}(v_1, K), \ldots, \phi(K, v_p) = \text{FE.Enc}(v_p, K)$ where encryption randomness is derived from $\texttt{PRF}(K, v_1), \ldots, \texttt{PRF}(K, v_p)$.

- $\texttt{Hyb}_1$: As the above except that we invoke the random oracle to replace the encryption randomness with true random strings fixed for each v_ℓ where $\ell \in [p]$.
- $\texttt{Hyb}_2$: As the above except that we replace the real protocol with the simulated one, algorithm 2
- $\texttt{Hyb}_3$: As the above except that we replace $\texttt{inner}_i(\phi(K,u))$ with $\perp$ if $\texttt{inner}_i(\phi(K,u)) = \phi(K,v)$. Note that $\mathcal{A}$ cannot distinguish between this and the above hybrid because $\mathcal{A}$ does not know the path from v_ℓ to v and can thus not find $\phi(K,u)$ from repeated queries of the embedded neighbor function, C_Γ nor can $\mathcal{A}$ generate $\phi(K,u)$ as K is hard to learn.

Indeed the simulated labels in $\texttt{Hyb}_3$, which we will denote $\phi'(K,v_\ell)$, are simulated independently of $\phi(K,v)$. So,

$$\mathbf{Pr}[\mathcal{B}\big(v,\phi'(K,v_1),\dots,\phi'(K,v_p),v_1,\dots,v_p,C_\Gamma\big)=\phi(K,v)] - \mathbf{Pr}[\mathcal{B}(v,C_\Gamma)=\phi(K,v)] \le \texttt{negl}(\lambda).$$

We can then assert that

$$\mathbf{Pr}[\mathcal{B}\big(v,\phi(K,v_1),\dots,\phi(K,v_p),v_1,\dots,v_p,C_\Gamma\big)=\phi(K,v)] - \mathbf{Pr}[\mathcal{B}(v,C_\Gamma)=\phi(K,v)] \le \texttt{negl}(\lambda).$$

And, by lemma 2.2, we have that

$$\mathbf{Pr}[\mathcal{B}(v,C_\Gamma)=\phi(K,v)] \le \texttt{negl}(\lambda)$$

thus concluding the proof. □

3 Welded Tree Graph Construction

Algorithm 4 The circuit for the neighbor function, C_Γ.

function $\texttt{INNER}_i(\texttt{Dec}(\phi(v)) = v, K, K')$
  $K_\alpha, K_\beta, K_\gamma = PRG(K')$
  $u_1,\dots,u_3 = \Gamma(v)$
  $a_1,a_2,a_3 = PRP(K_\gamma,1), PRP(K_\gamma,2), PRP(K_\gamma,3)$ where we are permuting over $\{1,2,3\}$
  $u = u_{a_i}$
  $r = H(K,u)$
  return FE.Enc(MPK, (u,K,K')) where we encrypt with randomness from r.
function $C_\Gamma(\phi(v))$
  for $i \in [d]$ **do**
    $u_i = \text{FE.Dec}(\text{SK}_{\texttt{inner}_i}, \phi(v))$
  return $(u_1,\dots,u_d)$

September 26, 2023

Abstract

References

[AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In *Theory of Cryptography Conference*, pages 173–205. Springer, 2017. 1.1, 1.1