Lattices and Homomorphic Encryption, Spring 2013	Instructors: Shai Halevi, Tal Malkin

Attribute-Based Encryption for Circuits [GVW13]

April 12, 2013 — Scribe: Kina Winoto

The scheme from [GVW13] works as follows:

(pp, msk) $\leftarrow$ **Setup**($) for ℓ-bit input x'es, depth d circuits: (Note that for this scheme we need a bound on depth of circuit, because at input the error expands as we get to output. Thus to get a bound on the error, we need a bound on the depth.)

We need to generate two matrices for each input wire and a matrix for the output wire. For the input wires we use the lattice-trapdoor-sampling procedure $TGen$ (that returns a nearly matrix $A \in \mathbb{Z}_q^{n\times m}$ together with a trapdoor t for A), for the putput wire we just choose the matrix at random:

- For $i = 1, 2, \ldots, \ell$ and $b \in \{0, 1\}$, set $(A_{i,b}, t_{i,b}) \leftarrow \text{TGen}\,(n, q, m, \text{error distrib})$.
- For the output wire, choose a random matrix, $A_{out,1} \in_R \mathbb{Z}_q^{n\times m}$.

The public parameters are $pp = \{A_{i,b}, A_{\text{out},1}\}_{i\in[\ell],b=0,1}$, and the master secret key is $msk = \{t_{i,b}\}_{i\in[\ell],b=0,1}$.

$\mathbf{CT_x}$ $\leftarrow$**Encrypt**($\vec{M} \in \{0,1\}^m;\ pp, x \in \{0,1\}^\ell$):

- Choose at random $\vec{s} \in \mathbb{Z}_q^n$.
- Choose $\vec{e_1}, \ldots, \vec{e_\ell}, \vec{e}_{\text{out}} \longleftarrow$ error distribution
- Set $\vec{v_i} = \vec{s}A_{i,x_i} + \vec{e_i}$ for $i = 1, \ldots, \ell$ and $\vec{c} = \vec{s}A_{\text{out},1} + \vec{e}_{\text{out}} + \left\lfloor \frac{q}{2} \right\rfloor \vec{M}$
- $CT_x = \big(x, \{v_i\}_{i=1}^{\ell},\ \vec{c}\big)$.

Note that we are only trying to hide $\vec{M}$, not x.

$\mathbf{sk_P}$ $\leftarrow$**KeyGenerator**(P, msk): Let C be a circuit computing the predicate P, with input wires $1, \ldots, \ell$, intermediate wires $\ell + 1, \ldots, N - 1$ and output wire N.

Note that in the delegation scheme in the last lecture we could generate parameters specifically for a given circuit. However, in this construction we don't know anything about the circuit when we generate the parameters, so we have somehow "stitch" the new matrices that we generate for C to the matrices $A_{i,b}$ and $A_{\text{out},1}$ from the public key, using the trapdoors that we have in the master-secret key. We will again use $TGen$ to choose all the matrices that we need, and will use the trapdoors to generates the R's.

- for $i = \ell + 1, \ldots, N - 1,\ b \in \{0, 1\}$, set $(A_{i,b}, t_{i,b}) \longleftarrow \text{TGen}(n, q, m, \text{error distrib.})$
- $A_{N,0} \in_R\ \mathbb{Z}_q^{n\times m}$
- For every gate G with input wires u, v and output wire w, use the trapdoors for $A_{u,*}, A_{v,*}$ to sample the R matrices such that $A_{u,b}R_{bc} + A_{v,c}R'_{bc} = A_{w,G(bc)}$ and R's small. We do this using the same method as the delegation scheme in last lecture:
  - Choose $R[G]'_{bc} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times n},\sigma}$

- Set $\Delta = A_{w,G(bc)} - A_{vc}R[G]'_{bc}$, denote the columns of Δ by $\Delta = \left(\vec{\delta}_1|\dots|\vec{\delta}_m\right)$.
- The i^{th} row of R is drawn from the discrete Gaussian distribution $\vec{r}_i \leftarrow \mathcal{D}_{\mathcal{L}^{\perp}_{\vec{\delta}_i}(A_{u,b}),\sigma}$. Thus $\vec{r}_i$ is Gaussian such that $A_{u,b}\vec{r}_i = \vec{\delta}_i$.
- Set $R[G]_{bc} = (\vec{r}_1|\dots|\vec{r}_m)$.

- The secret key is $sk_P = \{(R[G]_{bc}, R[G]'_{bc}) \mid G \text{ is a gate}; b, c \text{ are bits}\}$.

$\mathbf{M}/\perp \leftarrow \mathbf{Decrypt}(\mathbf{CT_x}, \mathbf{sk_P})$: Evaluate the circtuit $C_P(x)$ and remember the bits on all the wires. If $C_P(x) = 0$ then output $\perp$.

If $C_P(x) = 1$ then go over the circuit C_P in a bottom-up fashion. For every gate with input wires u, v and output wire w, input bits b, c and output bit d, and input vectors $\vec{u}_b, \vec{v}_c$, compute:

$$\vec{w}_d = \vec{u}_b R_{bc} + \vec{v}_c R'_{bc}$$

Denote the output vector by $\vec{w}_{\text{out}}$ and let $\vec{\delta}_{\text{out}} = \vec{c} - \vec{w}_{\text{out}}$ (where $\vec{c}$ is the "output vector" in the ciphertext CT_x). Then output the vector $\vec{M}$ where for all $i = 1, \dots, m$

$$M_i = \begin{cases} 0 \text{ if } |\vec{\delta}_i| < \frac{q}{4} \\ 1 \text{ if } |\vec{\delta}_i| \geq \frac{q}{4} \end{cases}$$

Correctness

If $p(x) = 1$, then $\vec{w}_{\text{out}} = \vec{s}A_{\text{out},1} + \vec{e}$ for some small $\vec{e}_0$. Also $\vec{c}$ is of the same form, except with $\left\lfloor \frac{q}{2} \right\rfloor \vec{M}$ added. Hence $\vec{\delta} = \vec{s}A_{\text{out},1} + \vec{e} + \lfloor q/2 \rfloor \cdot \vec{M}$ for a small $\vec{e}$, and correctness follows.

Security

Recall the interaction between scheme and attacker in our security model:

Scheme		**Attacker**
	$\xleftarrow{x^*}$	
$pp, msk \leftarrow \texttt{Setup}(\$)$	$\xrightarrow{pp}$	$p_i(x^*) = 0$
$sk_{p_i} \leftarrow \texttt{KeyGen}(p_i; msk)$	$\left\{ \xleftarrow{p_i} \; \xrightarrow{sk_{p_i}} \right\}_{i=1}^{q}$	$\forall i\, p_i(x^*) = 0$
$j \in_R \{1,2\}$	$\xleftarrow{m_1, m_2}$	
$ct_{x^*} \leftarrow \texttt{Encrypt}(m_i; pp, x^*)$	$\xrightarrow{ct_{x^*}}$	$\rightarrow j'$
	$j' \stackrel{?}{=} j$	

Will reduce security to the hardness of decision LWE. Namely, we show that if D-LWE is hard for params $(n, m' = m(\ell+1), q, \text{error distrib.})$, then the scheme outlined above is secure. (We note that this proof is slightly different than the one presented in GVW's paper.)

Assume an adversary $\mathcal{A}$ that breaks the scheme with success probability $\frac{1}{2} + \varepsilon$. We build an LWE-distinguisher $\mathcal{B}$ using $\mathcal{A}$. The distinguisher $\mathcal{B}$ gets as input an instance of D-LWE, namely $(A^*, \vec{v}^*)$, which we parse as follows:

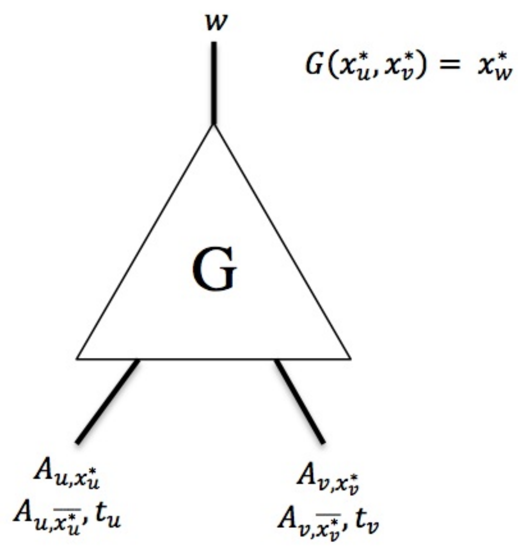

Figure 1: An illustration of one gate in the circuit C

- $A^* = (A_1|A_2|\dots|A_\ell|A_{\text{out}}) \in \mathbb{Z}_q^{n\times m'}$, for $\ell+1$ matrices $A_i, A_{\text{out}} \in \mathbb{Z}_q^{n\times m}$.
- $\vec{v}^* = (\vec{v_1}|\vec{v_2}|...|\vec{v_l}|\vec{v}_{\text{out}})$, for $\ell+1$ vectors $\vec{v}_i, \vec{v}_{\text{out}} \in \mathbb{Z}_q^m$.

$\mathcal{B}$ runs $\mathcal{A}$ to get the "challenge pattern" $x^* \in \{0,1\}^\ell$, then proceeds as follows:

- For $i = 1, 2, \dots, \ell$, let $A_{i,x_i^*} := A_i$, and also set $A_{\text{out},1} := A_{\text{out}}$.
- Also choose the matrices $A_{i,\bar{x_i^*}}$ together with trapdoors, $(A_{i,\bar{x_i^*}}, t_{i,\bar{x_i^*}}) \leftarrow \text{TDGen}(q, m, n, ..)$

The public params that we give to $\mathcal{A}$ are $A_{\text{out},1}$ and all the $\{A_{i,b}\}_{i=1,\dots,\ell,\ b=0,1}$. When the attacker asks for a secret key sk_P, with C being the circuit for P, then $\mathcal{B}$ does the following:

- For every wire $i = 1, 2, ...N$, denote by x_i^* the bit on the i'th wire when evaluating the circuit $C(x^*)$. (Hence the input wires are labeled just as before, and for the internal wires we now have the "active bit" on that wire x_i^* and the "inactive bit" $\bar{x_i^*}$.)
- $\mathcal{B}$ chooses the A and R matrices for the sk_P so that on every wire i, we know a trapdoor for $A_{i,\bar{x_i^*}}$ but not for A_{i,x_i^*}. (And also we don't know either of the trapdoors for the output wire.) Specifically, fora gate G with input wires u, v and output wire w $\mathcal{B}$ does the following (see illustration in Figure 1):
  - For the bits x_u^*, x_v^*, choose random small matrices from the discrete Gaussian distribution over the integers, $R_{x_u^*,x_v^*}, R'_{x_u^*,x_v^*} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\sigma}$.
  - Then $\mathcal{B}$ sets $A_{w,x_w^*} = A_{u,x_u^*} R_{x_u^* x_v^*} + A_{v,x_v^*} R'_{x_u^* x_v^*}$. That is, $\mathcal{B}$ computes the matrix A_{w,x_w^*} in the "forward direction" (first compute the R's then A), and it does not know a trapdoor for it.
  - For each of the other three pairs $(b, c) \neq (x_u^*, x_v^*)$, $\mathcal{B}$ uses the trapdoor that it knows for b or c. First it chooses $A_{w,\bar{x_w^*}}$ with a trapdoor,$(A_{w,\bar{x}_w^*}, t_{w,\bar{x}_w^*}) \leftarrow \text{TDGen}(...)$. Then it uses the same procedure as in the scheme itself to compute the relevant R's.

When $\mathcal{A}$ sends the challenge messages $(\vec{M}_1, \vec{M}_2)$, $\mathcal{B}$ does the following:

- Use $\vec{v}_i^*$ from the input of B as the i^{th} input vector, corresponding to input wire i.
- Use $\vec{c} = \vec{v}_{\text{out}} + \left\lfloor \frac{q}{2} \right\rfloor \vec{M}_j$ for a random $j \in \{1, 2\}$.

When $\mathcal{A}$ guesses j', then B output "LWE" if $j' = j$ and "random" otherwise.

Analysis of the distinguisher $\mathcal{B}$. Observe that if the input to B is LWE instance then:

- All the vectors in the ciphertext CT_{x^*} that $\mathcal{B}$ generates have the correct distribution asin the actual scheme.
- The matrices in all the secret keys sk_P have nearly the right distribution. This is because setting $R, R' \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$ and $A_w := A_u R + A_v R'$ (as $\mathcal{B}$ does) yields nearly the same distribution as choosing at random $A_w \leftarrow \mathbb{Z}_q^{n \times m}$ and $R' \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$ and using the trapdoor to sample $R \leftarrow \mathcal{D}_{\mathcal{L}_\delta^\perp(A),\sigma}$ (as done in the scheme).

Therefore in the case that the input to $\mathcal{B}$ was indeed an LWE instance, $\mathcal{A}$ will guess j with probability $\geq \frac{1}{2} + \varepsilon - negl$.

On the other hand, if the input to $\mathcal{B}$ is random then in particular $\vec{v}_{\text{out}}$ is random, so $\vec{c}$ is random, independent of $\vec{M}_1, \vec{M}_2$, so $\mathcal{A}$ guesses j with probability $\leq \frac{1}{2}$.

References

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee, *Predicate encryption for circuits*, STOC, 2013.