FHE Circuit Privacy Almost For Free

Florian Bourse, Rafaël Del Pino, Michele Minelli,
and Hoeteck Wee

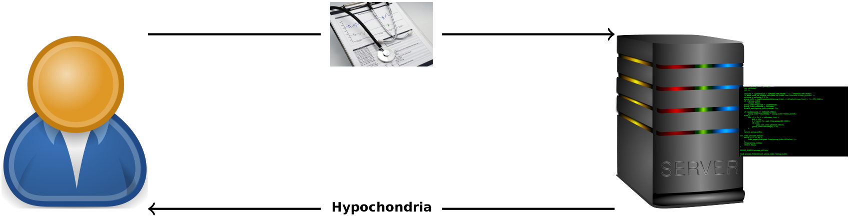
CNRS, École normale supérieure, INRIA, PSL, Paris, France

CRYPTO 2016 — Santa Barbara
Wednesday, August 17

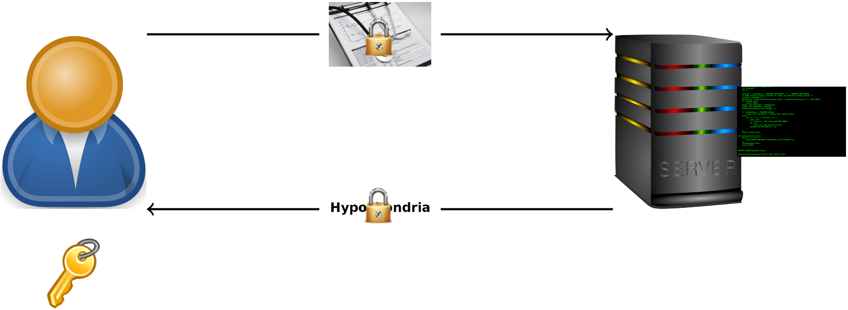
Example: online diagnostic

Example: online diagnostic

Example: online diagnostic



Data privacy: FHE



GSW encryption scheme [GenSahWat13]

$$\mathbf{G} = \mathbf{Id}_n \otimes \mathbf{g}, \quad \mathbf{g} = \left(1, 2, \ldots, 2^k\right)$$

$$\mathbf{C} = \mathrm{Enc}(\mu) = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}\mathbf{A} + \mathbf{e} \end{pmatrix} + \mu\mathbf{G} \in \mathbb{Z}_q^{n \times m}$$

GSW encryption scheme [GenSahWat13]

$$\mathbf{G} = \mathbf{Id}_n \otimes \mathbf{g}, \quad \mathbf{g} = \left(1, 2, \ldots, 2^k\right)$$

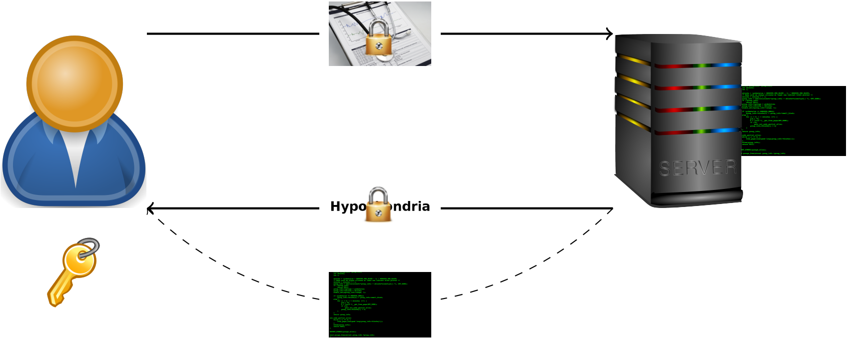
$$\mathbf{C} = \mathrm{Enc}(\mu) = \begin{pmatrix} \mathbf{A} \\ \mathbf{sA} + \mathbf{e} \end{pmatrix} + \mu\mathbf{G} \in \mathbb{Z}_q^{n \times m}$$

Sum $\mathrm{Enc}(\mu_1) + \mathrm{Enc}(\mu_2)$

Product $\mathrm{Enc}(\mu_1) \cdot \mathbf{G}^{-1}(\mathrm{Enc}(\mu_2))$

where $\forall\, \mathbf{v} \in \mathbb{Z}_q^n$, $\mathbf{G}^{-1}(\mathbf{v}) \in \mathbb{Z}_q^m$ is *small* and s.t. $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{v}) = \mathbf{v}$

Protecting the algorithm: circuit privacy



Leakage in the error term: toy example

Given $\mathbf{s}$, and 3 encryptions of 0:

$$\mathbf{C_1} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{s}\mathbf{A}_1 + \mathbf{e}_1 \end{pmatrix}, \ \mathbf{C_2} = \begin{pmatrix} \mathbf{A}_2 \\ \mathbf{s}\mathbf{A}_2 + \mathbf{e}_2 \end{pmatrix}, \ \mathbf{C_3} = \begin{pmatrix} \mathbf{A}_3 \\ \mathbf{s}\mathbf{A}_3 + \mathbf{e}_3 \end{pmatrix}.$$

Leakage in the error term: toy example

Given $\mathbf{s}$, and 3 encryptions of 0:

$$\mathbf{C_1} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{s}\mathbf{A}_1 + \mathbf{e}_1 \end{pmatrix},\ \mathbf{C_2} = \begin{pmatrix} \mathbf{A}_2 \\ \mathbf{s}\mathbf{A}_2 + \mathbf{e}_2 \end{pmatrix},\ \mathbf{C_3} = \begin{pmatrix} \mathbf{A}_3 \\ \mathbf{s}\mathbf{A}_3 + \mathbf{e}_3 \end{pmatrix}.$$

$\mathbf{C_i} + \mathbf{C_j}$ leaks i and j:

Leakage in the error term: toy example

Given $\mathbf{s}$, and 3 encryptions of 0:

$$\mathbf{C_1} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{s}\mathbf{A}_1 + \mathbf{e}_1 \end{pmatrix}, \ \mathbf{C_2} = \begin{pmatrix} \mathbf{A}_2 \\ \mathbf{s}\mathbf{A}_2 + \mathbf{e}_2 \end{pmatrix}, \ \mathbf{C_3} = \begin{pmatrix} \mathbf{A}_3 \\ \mathbf{s}\mathbf{A}_3 + \mathbf{e}_3 \end{pmatrix}.$$

$\mathbf{C_i} + \mathbf{C_j}$ leaks i and j:
The error term is $\mathbf{e}_i + \mathbf{e}_j$!

Protecting the algorithm: circuit privacy

$\mathsf{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_\ell)$ should reveal nothing on f but $f(\mu_1, \ldots, \mu_\ell)$.

Noise flooding [Gen09]

$$\mathbf{C}_f = \mathrm{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

Noise flooding [Gen09]

$$\mathbf{C}_f = \text{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

- $\mathbf{C}_f = \mathbf{C}_f + \begin{pmatrix} \mathbf{0} \\ \mathbf{e}' \end{pmatrix}, \quad q \gg \mathbf{e}' \gg \mathbf{e}_f$

Noise flooding [Gen09]

$$\mathbf{C}_f = \text{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

- $\mathbf{C}_f = \mathbf{C}_f + \begin{pmatrix} \mathbf{0} \\ \mathbf{e}' \end{pmatrix}, \quad q \gg \mathbf{e}' \gg \mathbf{e}_f$

Pros Destroys all information contained in the noise

Cons Requires superpolynomial modulus, not multi-hop

Soak-spin-repeat [DucSte16]

$$\mathbf{C}_f = \text{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

Soak-spin-repeat [DucSte16]

$$\mathbf{C}_f = \text{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

- $\mathbf{C}_f = \mathbf{C}_f + \begin{pmatrix} \mathbf{0} \\ \mathbf{e}' \end{pmatrix}, \quad \mathbf{e}' \approx \mathbf{e}_f$

Soak-spin-repeat [DucSte16]

$$\mathbf{C}_f = \mathrm{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

- $\mathbf{C}_f = \mathbf{C}_f + \begin{pmatrix} \mathbf{0} \\ \mathbf{e}' \end{pmatrix}, \quad \mathbf{e}' \approx \mathbf{e}_f$
- $\mathbf{C}_f = \mathrm{Eval}(\mathrm{Dec}(\cdot, \mathbf{C_f}), \mathrm{Enc}(sk))$

Soak-spin-repeat [DucSte16]

$$\mathbf{C}_f = \mathrm{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

- $\mathbf{C}_f = \mathbf{C}_f + \begin{pmatrix} \mathbf{0} \\ \mathbf{e}' \end{pmatrix}, \quad \mathbf{e}' \approx \mathbf{e}_f$
- $\mathbf{C}_f = \mathrm{Eval}(\mathrm{Dec}(\cdot, \mathbf{C_f}), \mathrm{Enc}(sk))$
- Repeat $O(\lambda)$ times

Soak-spin-repeat [DucSte16]

$$\mathbf{C}_f = \mathrm{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

- $\mathbf{C}_f = \mathbf{C}_f + \begin{pmatrix} \mathbf{0} \\ \mathbf{e}' \end{pmatrix}, \quad \mathbf{e}' \approx \mathbf{e}_f$
- $\mathbf{C}_f = \mathrm{Eval}(\mathrm{Dec}(\cdot, \mathbf{C_f}), \mathrm{Enc}(sk))$
- Repeat $O(\lambda)$ times

Pros Works with polynomial modulus, multi-hop

Cons Requires circular security (bootstrapping)

Our approach [BDMW16]

$$\mathbf{C}_f = \mathrm{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

Our approach [BDMW16]

$$\mathbf{C}_f = \mathrm{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

- $\mathbf{C}_f = \mathbf{C}_f + \begin{pmatrix} \mathbf{0} \\ \mathbf{e}' \end{pmatrix}, \quad \mathbf{e}' \approx \mathbf{e}_f$

Our approach [BDMW16]

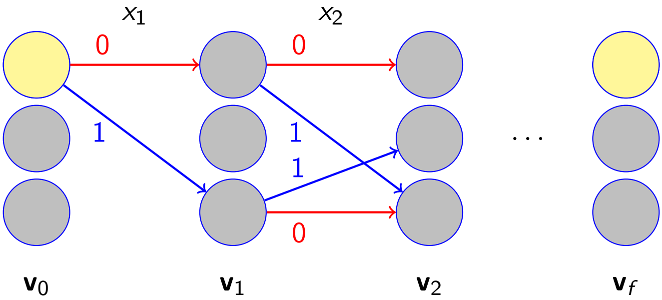
$$\mathbf{C}_f = \mathrm{Eval}(f, \mathbf{C}_1, \ldots, \mathbf{C}_n),$$

- $\mathbf{C}_f = \mathbf{C}_f + \begin{pmatrix} \mathbf{0} \\ \mathbf{e}' \end{pmatrix}, \quad \mathbf{e}' \approx \mathbf{e}_f$

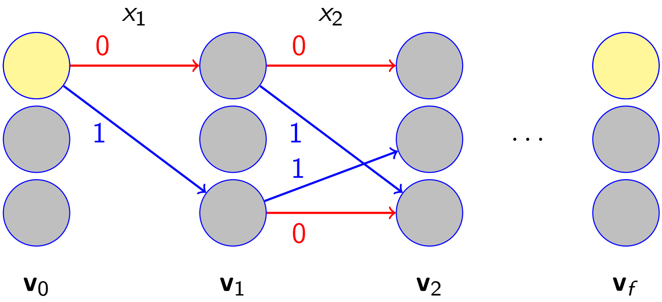
Pros Polynomial modulus, no circular security, multi-hop

Cons Only for NC^1 evaluations on GSW, leaks $|f|$

Branching programs



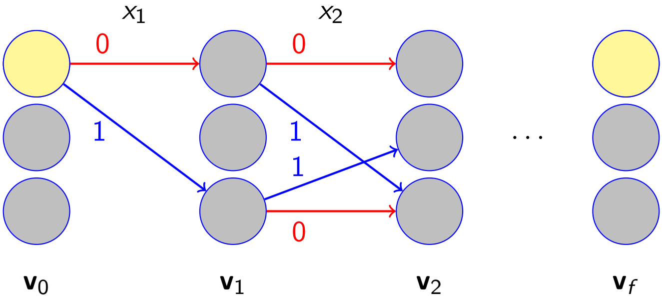
Branching programs



$$\mathbf{v}_t[i] = \begin{cases} \mathbf{v}_{t-1}[j] \text{ if } x_t = 1 \\ \mathbf{v}_{t-1}[k] \text{ if } x_t = 0 \end{cases}$$

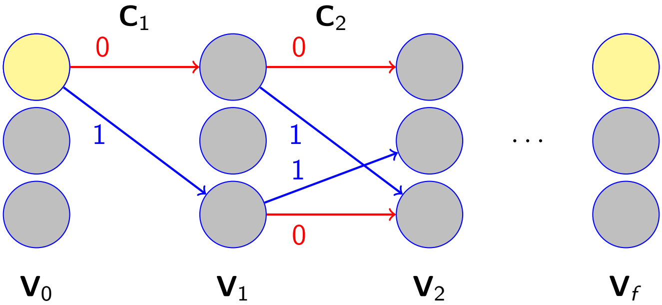
$$= \mathbf{v}_{t-1}[MUX(x_t, j, k)]$$

Branching programs



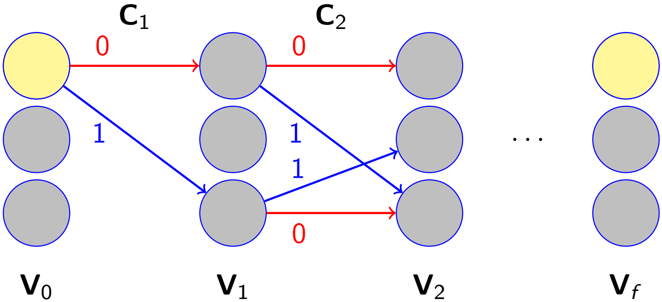
$$\mathbf{v}_t[i] = x_t \mathbf{v}_{t-1}[j] + (1 - x_t)\mathbf{v}_{t-1}[k]$$

Branching programs



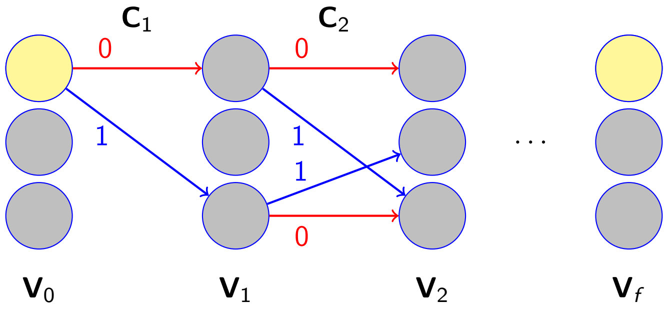
$$\mathbf{v}_t[i] = x_t \mathbf{v}_{t-1}[j] + (1 - x_t)\mathbf{v}_{t-1}[k]$$

Branching programs



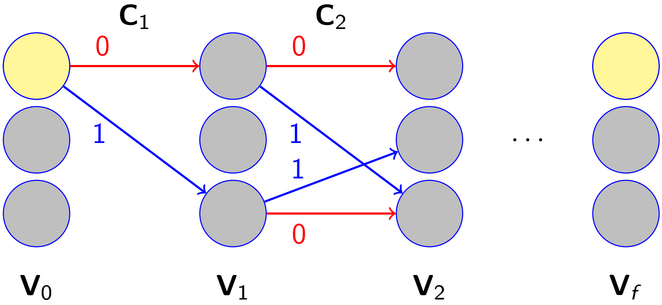
$$\mathbf{V}_t[i] = \mathbf{C}_t\mathbf{G}^{-1}(\mathbf{V}_{t-1}[j]) + (\mathbf{G} - \mathbf{C}_t)\mathbf{G}^{-1}(\mathbf{V}_{t-1}[k])$$

Branching programs



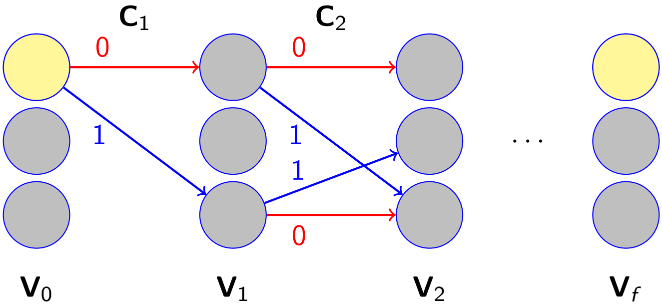
Let $\mathbf{C}_t = \mathbf{B}_t + x_t\mathbf{G}$

$$\mathbf{V}_t[i] = \mathbf{V}_{t-1}[MUX(x_t, j, k)] + \mathbf{B}_t\mathbf{G}^{-1}(\mathbf{V}_{t-1}[j] + \mathbf{V}_{t-1}[k])$$



Let $\mathbf{C}_t = \mathbf{B}_t + x_t\mathbf{G}$

$$\mathbf{V}_t[i] = \underbrace{\mathbf{V}_{t-1}[MUX(x_t, j, k)]}_{\text{previous step}} + \underbrace{\mathbf{B}_t\mathbf{G}^{-1}(\mathbf{V}_{t-1}[j] + \mathbf{V}_{t-1}[k])}_{\text{additional noise}}$$



Let $\mathbf{C}_t = \mathbf{B}_t + x_t\mathbf{G}$

$$\mathbf{V}_t[i] = \underbrace{\mathbf{V}_{t-1}[MUX(x_t, j, k)]}_{\text{previous step}} + \underbrace{\mathbf{B}_t\mathbf{G}^{-1}(\mathbf{V}_{t-1}[j] + \mathbf{V}_{t-1}[k])}_{\text{additional noise}}$$

The additional noise should not leak information about the branching program

Our core lemma: GSW rerandomization

Let $\mathbf{C}$ be an encryption of 0 with error $\mathbf{e}$. For any matrix $\mathbf{V}$:

$$\mathbf{C} \cdot \mathbf{G}^{-1}(\mathbf{V}) + \begin{pmatrix} \mathbf{O} \\ \mathbf{z} \end{pmatrix} \approx_s \mathbf{C}'$$

- $\mathbf{G}^{-1}$ Gaussian s.t $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{V}) = \mathbf{V}$
- $\mathbf{z}$ Gaussian vector

Our core lemma: GSW rerandomization

Let $\mathbf{C}$ be an encryption of 0 with error $\mathbf{e}$. For any matrix $\mathbf{V}$:

$$\mathbf{C} \cdot \mathbf{G}^{-1}(\mathbf{V}) + \begin{pmatrix} \mathbf{O} \\ \mathbf{z} \end{pmatrix} \approx_s \mathbf{C}'$$

- $\mathbf{G}^{-1}$ Gaussian s.t $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{V}) = \mathbf{V}$
- $\mathbf{z}$ Gaussian vector

With $\mathbf{C}'$ a fresh encryption of 0 with Gaussian error of parameter $\|\mathbf{e}\|$.

Our core lemma: GSW rerandomization

Let $\mathbf{C}$ be an encryption of 0 with error $\mathbf{e}$. For any matrix $\mathbf{V}$:

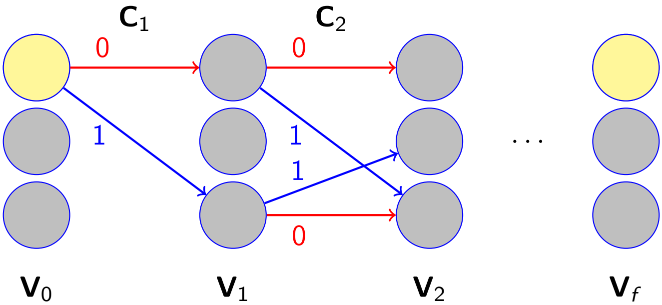
$$\mathbf{C} \cdot \mathbf{G}^{-1}(\mathbf{V}) + \begin{pmatrix} \mathbf{O} \\ \mathbf{z} \end{pmatrix} \approx_s \mathbf{C}'$$

- $\mathbf{G}^{-1}$ Gaussian s.t $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{V}) = \mathbf{V}$
- $\mathbf{z}$ Gaussian vector

With $\mathbf{C}'$ a fresh encryption of 0 with Gaussian error of parameter $\|\mathbf{e}\|$.

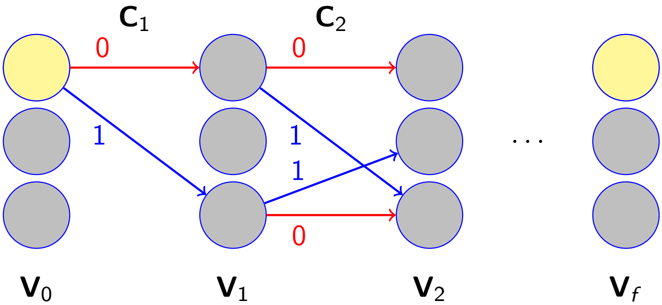
$\Rightarrow$ $\mathbf{C}'$ is independent of $\mathbf{V}$!

Modified evaluation



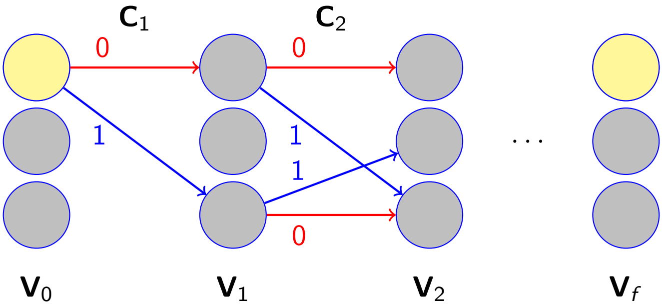
$$\mathbf{V}_t[i] = \mathbf{V}_{t-1}[MUX(x_t, j, k)] + \mathbf{B}_t \mathbf{G}^{-1}(\mathbf{V}_{t-1}[j] + \mathbf{V}_{t-1}[k]))$$

Modified evaluation



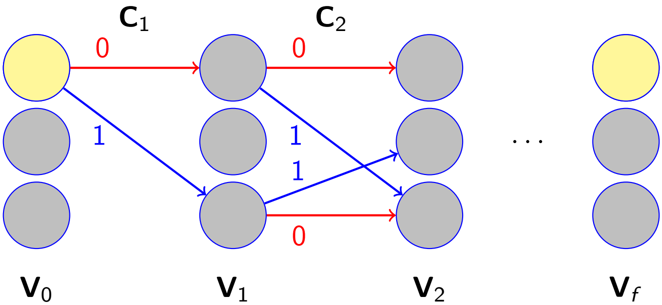
$$\mathbf{V}_t[i] = \mathbf{V}_{t-1}[MUX(x_t, j, k)] + \mathbf{B}_t \mathbf{G}^{-1}(\mathbf{V}_{t-1}[j] + \mathbf{V}_{t-1}[k])$$

Modified evaluation

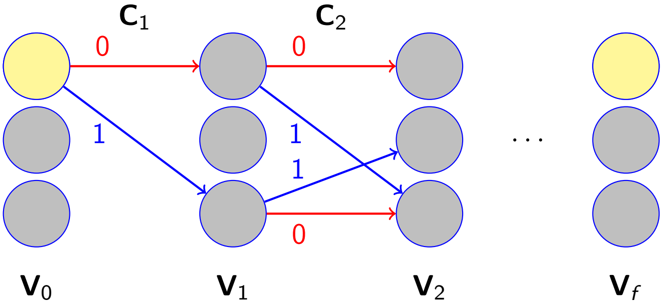


$$\mathbf{V}_t[i] = \mathbf{V}_{t-1}[MUX(x_t, j, k)] + \mathbf{B}_t\mathbf{G}^{-1}(\mathbf{V}_{t-1}[j] + \mathbf{V}_{t-1}[k]) + \begin{pmatrix} \mathbf{0} \\ \mathbf{z} \end{pmatrix}$$

Modified evaluation

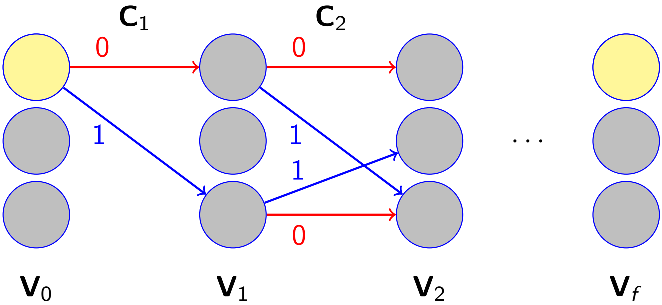


Modified evaluation



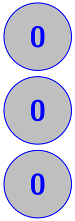
$$\mathbf{V}_t[i] = \mathbf{V}_{t-1}[MUX(x_t, j, k)] + \mathbf{C}$$

Modified evaluation



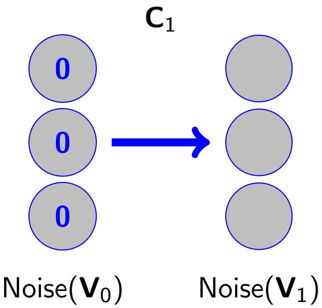
$$\mathbf{V}_t[i] = \mathbf{V}_{t-1}[MUX(x_t, j, k)] + \underbrace{\mathbf{C}}_{\text{depends only on } \mathbf{C}_t}$$

Circuit privacy by induction



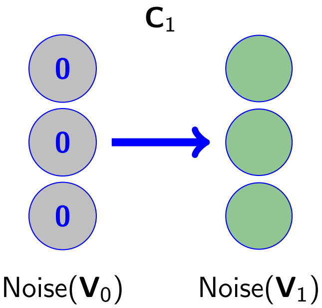
Noise($\mathbf{V}_0$)

Circuit privacy by induction



$$\text{Noise}(\mathbf{V}_1[i]) = \text{Noise}(\mathbf{V}_0[MUX(x_1, j, k)]) + \mathbf{B}_t \mathbf{G}^{-1}(\mathbf{V}_0[j] + \mathbf{V}_0[k]) + \begin{pmatrix} \mathbf{O} \\ \mathbf{z} \end{pmatrix}$$

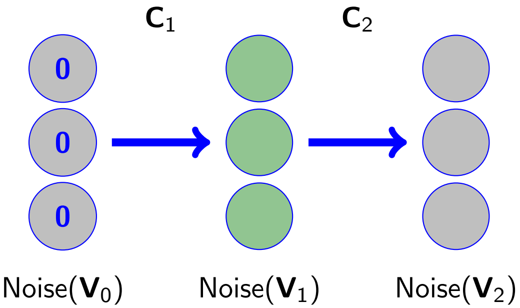
Circuit privacy by induction



$$\text{Noise}(\mathbf{V}_1[i]) \approx_s \mathbf{O} + \mathbf{C}_1'$$

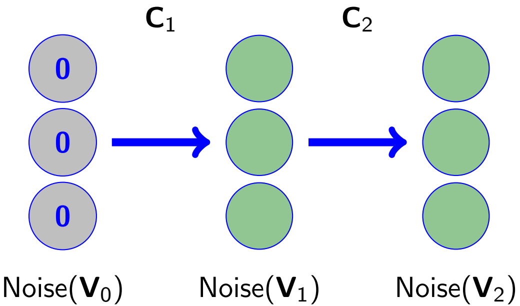
Where $\mathbf{C}_1'$ depends only on $\mathbf{C}_1$

Circuit privacy by induction



$$\begin{aligned}\mathrm{Noise}(\mathbf{V}_2[i]) = &\mathrm{Noise}(\mathbf{V}_1[MUX(x_2, j, k)]) \\ &+ \mathbf{B}_t \mathbf{G}^{-1}(\mathbf{V}_1[j] + \mathbf{V}_1[k]) + \begin{pmatrix} \mathbf{0} \\ \mathbf{z} \end{pmatrix}\end{aligned}$$

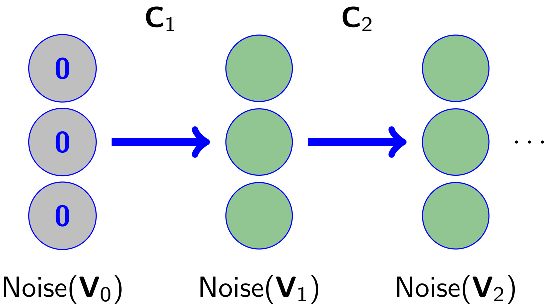
Circuit privacy by induction



$$\text{Noise}(\mathbf{V}_1[i]) \approx_s \mathbf{C}'_1 + \mathbf{C}'_2$$

Where $\mathbf{C}'_2$ depends only on $\mathbf{C}_1$

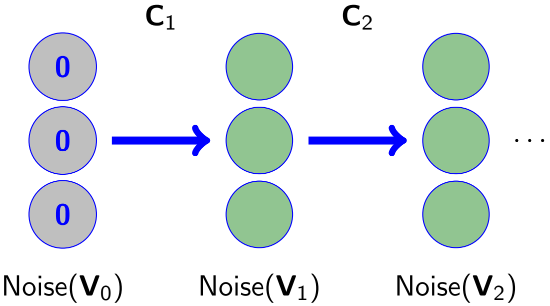
Circuit privacy by induction



$$\text{Noise}(\mathbf{V}_f[i]) \approx_s \sum_1^f \mathbf{C}'_k$$

The final noise depends only on the number of time each choice bit has been used

Circuit privacy by induction



$$\text{Noise}(\mathbf{V}_f[i]) \approx_s \sum_{1}^{f} \mathbf{C}'_k$$

Padding $\Rightarrow \text{Noise}(\mathbf{V}_f[i]) = \text{function of } |BP|$

Thank you!

Questions?