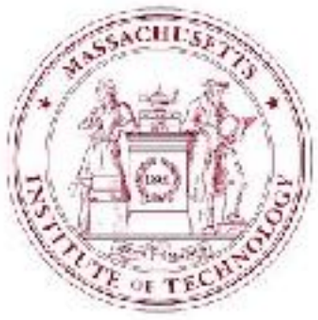
Private Circuits
A Modular Approach

Prabhanjan Ananth

Yuval Ishai

Amit Sahai



Surveillance Devices

Credit card details
SSN number
Passwords
PGP keys
...

...

Surveillance Devices

Credit card details
SSN number
Passwords
PGP keys
...

...

Side Channel Attacks

Adversary can obtain partial information (leakage) about the computation

Leakage-Resilient Cryptography

GOAL

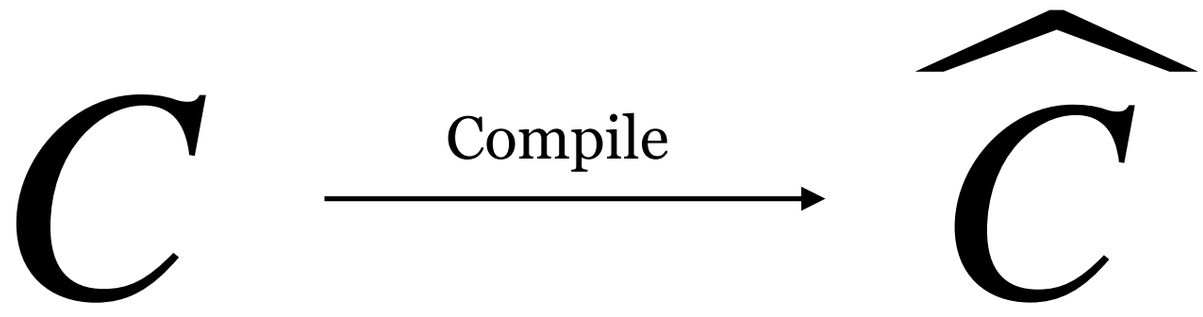
Protecting cryptographic schemes against

side-channel attacks

This Work:

Leakage-Resilient Circuit Compilers [ISW03]

Circuit Compilers



Circuit Compilers

$$C \xrightarrow{\text{Compile}} \widehat{C}$$

$$x \xrightarrow[\$\$]{\text{Encode}} \hat{x}$$

$$\widehat{C}\left(\hat{x}\right) \xrightarrow{\text{Decode}} C(x)$$

Remarks

- $C, \widehat{C}$ contain NAND gates

Remarks

- $C, \widehat{C}$ contain NAND gates

  - *other bases for* $\widehat{C}$ *: results can be adapted*

Remarks

- $C, \widehat{C}$ contain NAND gates

  - *other bases for $\widehat{C}$: results can be adapted*

- Circuit compilation is deterministic

Remarks

- $C, \widehat{C}$ contain NAND gates

  - *other bases for $\widehat{C}$: results can be adapted*

- Circuit compilation is deterministic

  - *compiled circuit is reusable; no trapdoors*

Remarks

- $C, \widehat{C}$ contain NAND gates

  - *other bases for $\widehat{C}$: results can be adapted*

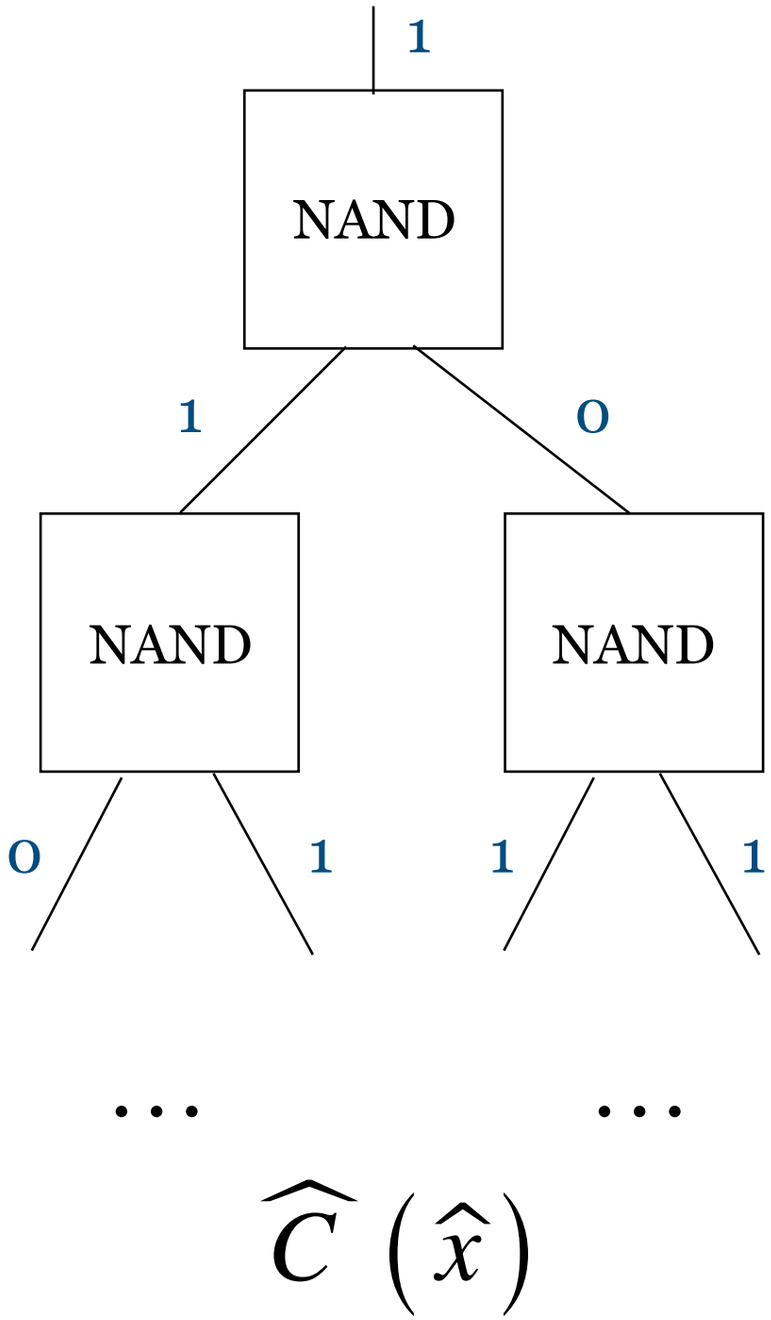
- Circuit compilation is deterministic

  - *compiled circuit is reusable; no trapdoors*

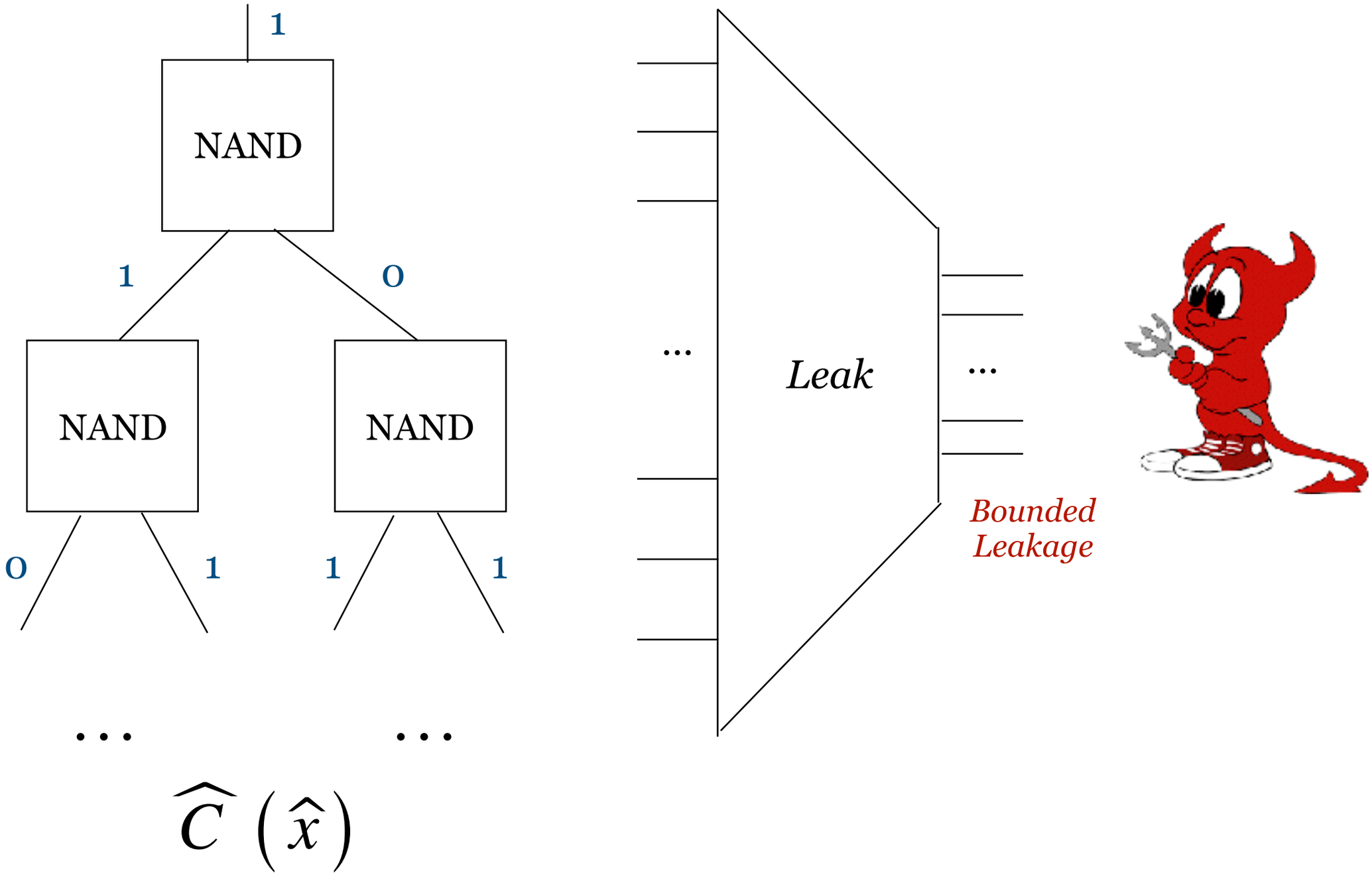
- $\widehat{C}$ can contain random-bit gates

Leakage-Resilient Circuit Compilers

Leakage-Resilient Circuit Compilers



Leakage-Resilient Circuit Compilers



Leakage on computation of $\widehat{C}$ on $\hat{x}$

What is *Leak*?

- **Global leakage:** *Leak* is function of entire computation

- **Local leakage:** adversary has partial view of computation

What is *Leak*?

- **Global leakage:** *Leak* is function of entire computation

  - low-complexity leakage classes [FRRTV11,Rot12]

- **Local leakage:** adversary has partial view of computation

What is *Leak*?

- **Global leakage:** *Leak* is function of entire computation

  - low-complexity leakage classes [FRRTV11,Rot12]

- **Local leakage:** adversary has partial view of computation

  - Wire-probing attacks [ISW03,…]

  - Split-state leakage-resilient compiler [MR03,DP08,GR12,…]

What is *Leak*?

- **Global leakage:** *Leak* is function of entire computation

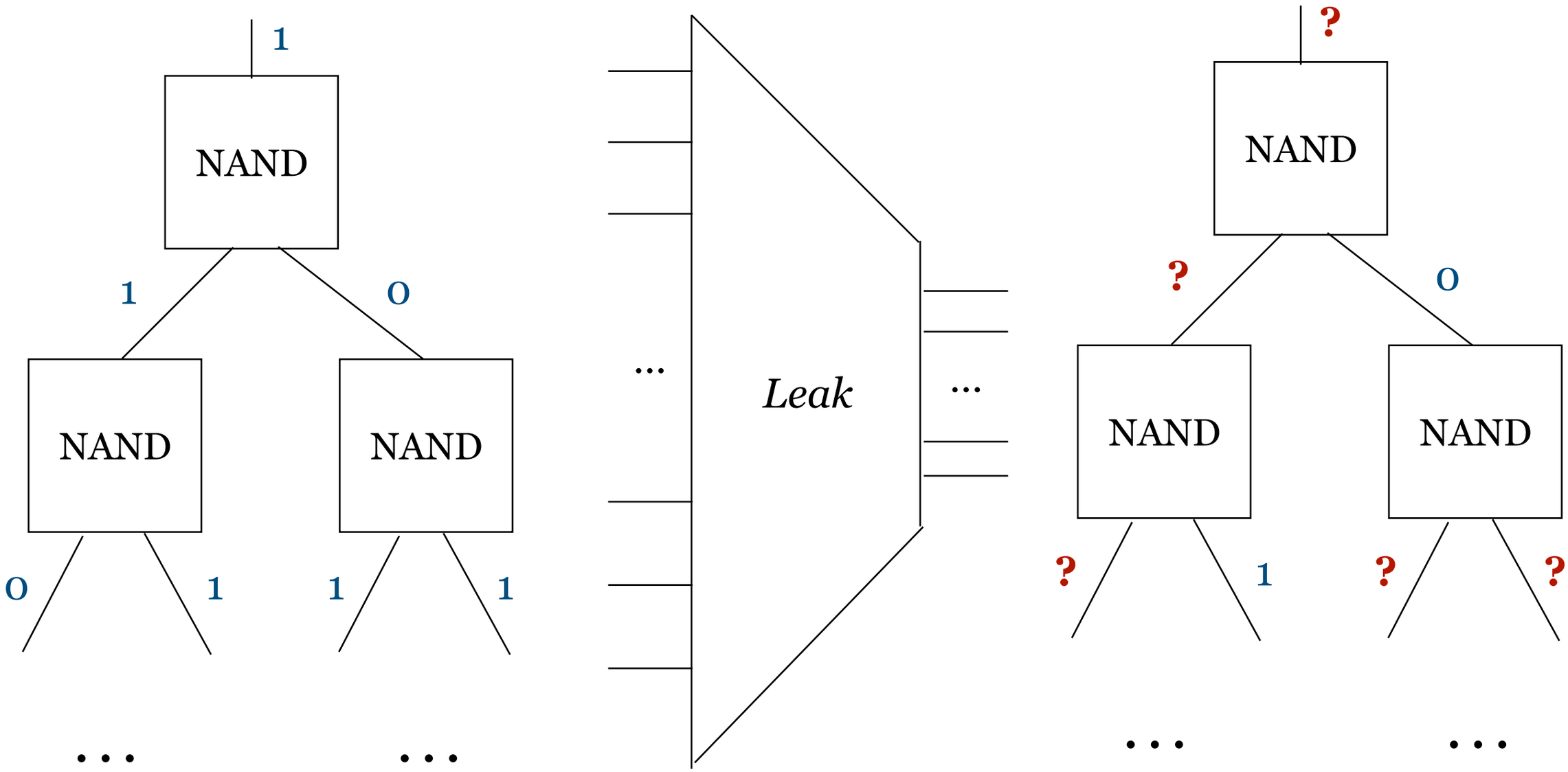
  - low-complexity leakage classes [FRRTV11,Rot12]

- **Local leakage:** adversary has partial view of computation

  - Wire-probing attacks [ISW03,…]

  - Split-state leakage-resilient compiler [MR03,DP08,GR12,…]

Wire-probing attacks [ISW03, …]



Subset of values in the computation leaked

Leakage-Resilience: Wire-probing attacks [ISW03,…]

Worst Case Leakage: threshold t

- Any t wires are leaked

Leakage-Resilience: Wire-probing attacks [ISW03,...]

Worst Case Leakage: threshold t

- Any t wires are leaked

Following [ISW03], several works study this setting...
[RP10,KHL11,GM11,CPR13,CGPQR12,...]

MPC on Silicon

Applying MPC techniques to design secure hardware

Leakage-Resilience: Wire-probing attacks [ISW03,...]

Worst Case Leakage: threshold t

- Any t wires are leaked

Recent years: focus on randomness complexity

[IKLOPSZ13,BBPPTV16,BBPPTV17]

Randomness Complexity

Randomness Complexity = # of random-bit gates

Randomness Complexity

Randomness Complexity = # of random-bit gates

How many random bit-gates are needed?

Randomness Complexity

Randomness Complexity = # of random-bit gates

How many random bit-gates are needed?

[IKLOPSZ13] $t^{3+\varepsilon}$ random bit-gates sufficient, for any $\varepsilon > 0$

Randomness Complexity

Randomness Complexity = # of random-bit gates

How many random bit-gates are needed?

[IKLOPSZ13] $t^{3+\varepsilon}$ random bit-gates sufficient, for any $\varepsilon > 0$

Q: Is $t^{3+\varepsilon}$ tight?

Randomness Complexity

Randomness Complexity = # of random-bit gates

How many random bit-gates are needed?

[IKLOPSZ13] $t^{3+\varepsilon}$ random bit-gates sufficient, for any $\varepsilon > 0$

Q: Is $t^{3+\varepsilon}$ tight?

NO!

Results: Worst-Case Probing

Leakage resilient compilers for s-sized circuits and threshold t

Results: Worst-Case Probing

Leakage resilient compilers for s-sized circuits and threshold t

- secure against t-wire probing attacks

Results: Worst-Case Probing

Leakage resilient compilers for s-sized circuits and threshold t

- secure against t-wire probing attacks

- compiled circuit has size $s \cdot poly(t)$

Results: Worst-Case Probing

Leakage resilient compilers for s-sized circuits and threshold t

- secure against t-wire probing attacks

- compiled circuit has size $s \cdot poly(t)$

- randomness complexity = $t^{1+\varepsilon}$, for any $\varepsilon > 0$

Leakage-Resilience: Random Wire-probing attacks

[ISW03,Ajtai10,ADF16]

Leakage-Resilience: Random Wire-probing attacks

[ISW03,Ajtai10,ADF16]

Probabilistic Leakage: parameterized by *(p,e)*

Real World	Ideal World
Every wire in $\widehat{C}\left(\hat{x}\right)$ leaked with probability p	Simulate leakage just given C

$$\approx_e$$

Leakage-Resilience: Random Wire-probing attacks

[ISW03,Ajtai10,ADF16]

Probabilistic Leakage: parameterized by *(p,e)*

Real World	Ideal World
Every wire in $\widehat{C}\ (\hat{x})$ leaked with probability p	Simulate leakage just given C

$$\approx_e$$

Related to Noisy Leakage Model: [CJJR99,FRRTV10,DDF15,...]

Prior works:
Random Wire-Probing Attacks

p = constant, e = negligible

Prior works:
Random Wire-Probing Attacks

p = constant, e = negligible

- [Ajtai10]:

  - highly complex

Prior works:
Random Wire-Probing Attacks

p = constant, e = negligible

- [Ajtai10]:

  - highly complex

- [ADF16]:

  - simplifies Ajtai's result

  - still uses heavy machinery (AG codes and expanders)

Results: Random-Wire Probing

Leakage-resilient circuit compiler against *(p,e)*-random probing attacks

- for some $0 < p < 1$

- e negligible in circuit size

Results: Random-Wire Probing

Leakage-resilient circuit compiler against *(p,e)*-random probing attacks

- for some $0 < p < 1$

- *e* negligible in circuit size

p = 0.000065

Results: Random-Wire Probing

Leakage-resilient circuit compiler against *(p,e)*-random probing attacks

- for some $0 < p < 1$

- e negligible in circuit size

- Simple composition-based approach; uses only elementary tools

Results: Random-Wire Probing

Leakage-resilient circuit compiler against *(p,e)*-random probing attacks

- for some $0 < p < 1$

- e negligible in circuit size

Large gates: construction with *p close to 1*

Leakage Tolerance

Leakage Tolerance

$$\widehat{x} = x$$

$$\widehat{C}\left(\widehat{x}\right) = C(x)$$

Input encoding and Output decoding algorithms are identity functions

Leakage Tolerance

$$\widehat{x} = x$$

$$\widehat{C}\left(\widehat{x}\right) = C(x)$$

Input encoding and Output decoding algorithms are identity functions

This implies leakage-resilience!

Security Notions

A fraction of input and output will be leaked

Security Notions

A fraction of input and output will be leaked

- Worst-case: *parameterized by t*

Leakage simulatable given

- *t* bits of input
- *t* bits of output

Security Notions

A fraction of input and output will be leaked

- Probabilistic: *parameterized by (p,**p'**,e)*

Leakage simulatable given
- every bit of input **x** w/ probability *p'*
- every bit of output **C(x)** w/ probability *p'*

Results: Leakage Tolerance

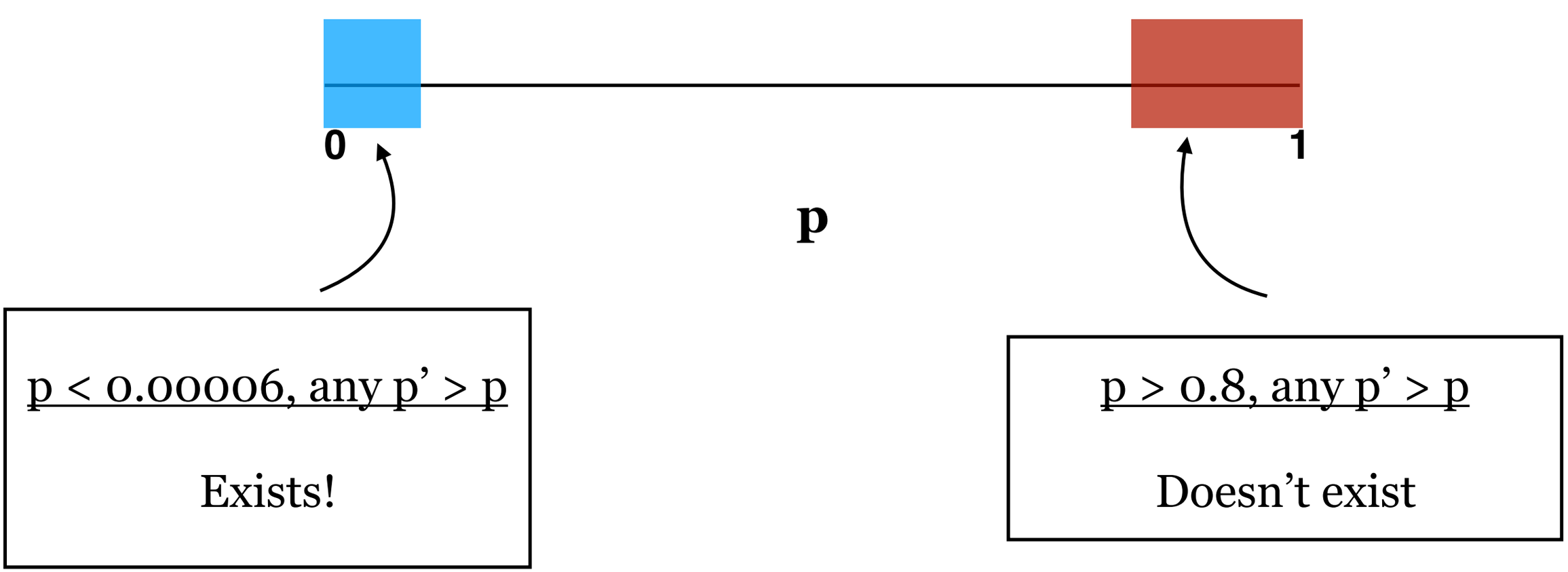
Worst Case: t-wire probing attacks

- *construction: randomness complexity* $t^{1+\varepsilon}$

- *lower bound: require at least* t *random-bit gates*

Results: Leakage Tolerance

Probabilistic Case: (**p**,**p'**,e)-random probing attacks



Techniques

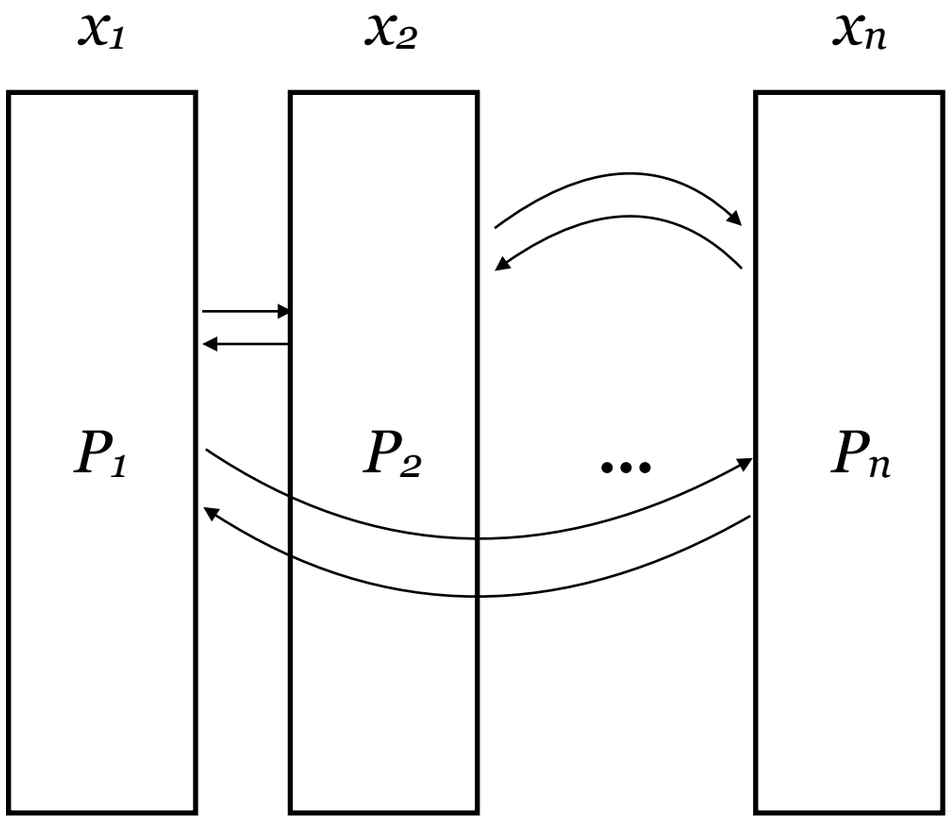
Goal for this talk

- Leakage-resilient circuit compiler

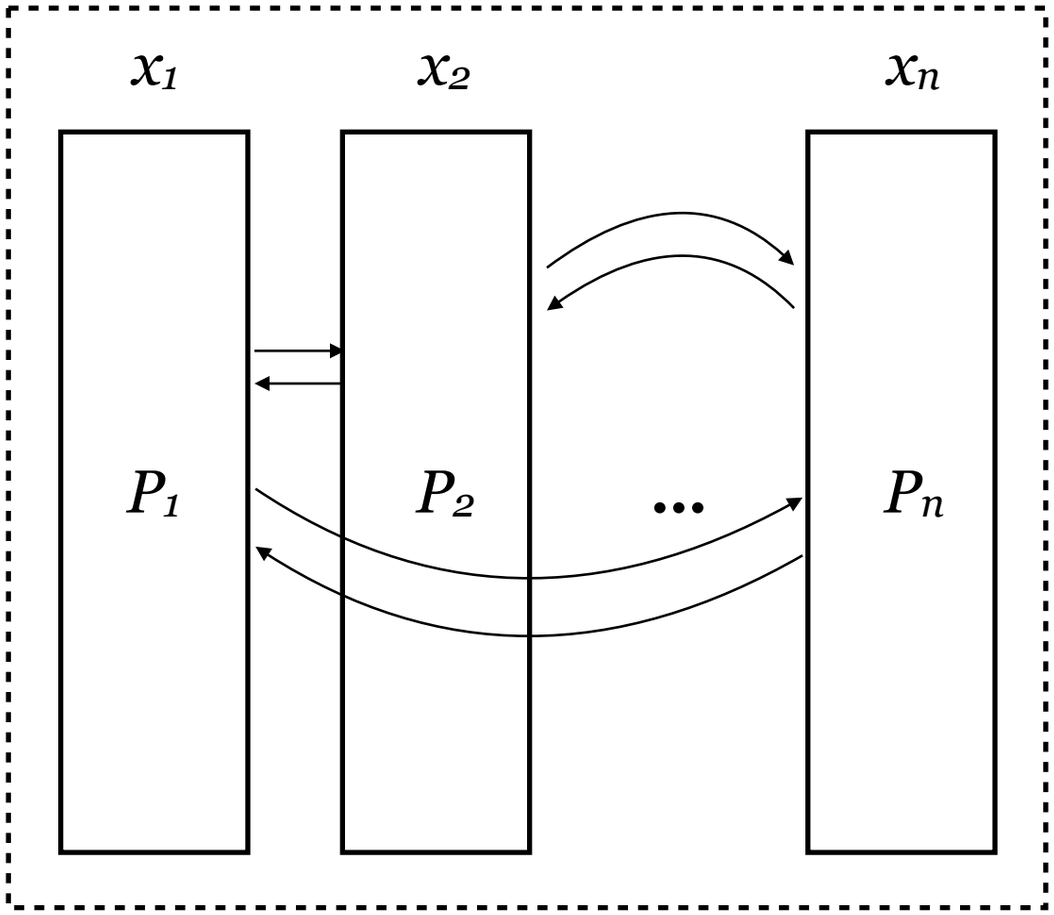
- *(p,e)*-random probing attacks

Starting Point:
t-out-n Secure MPC

$\Pi(C)$

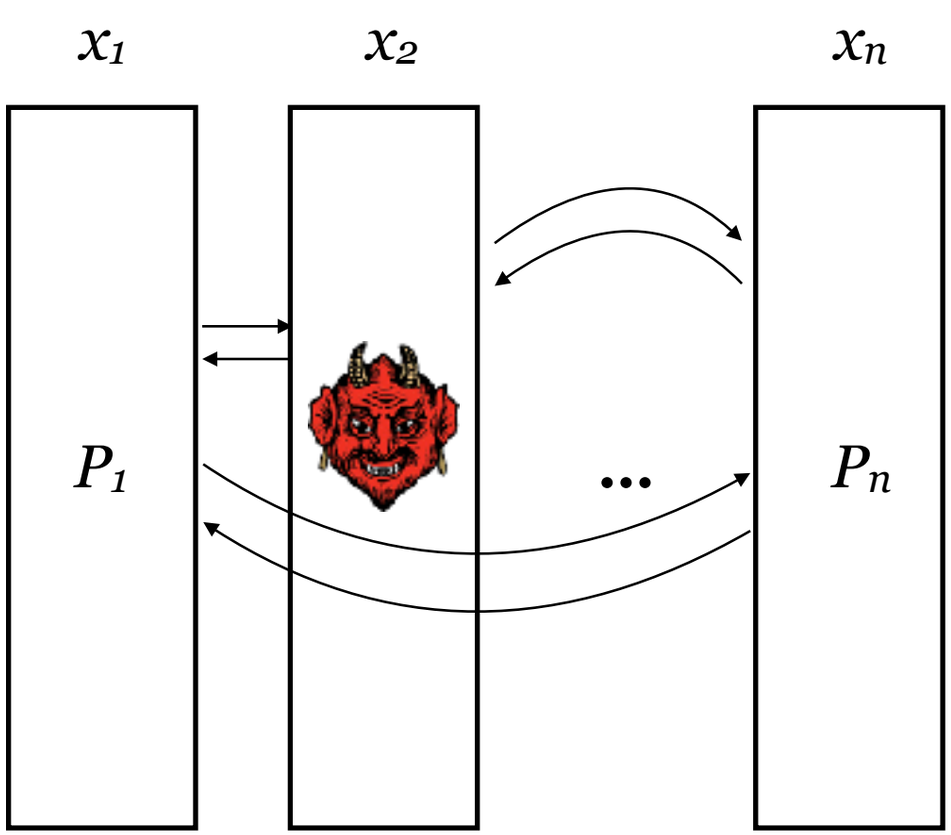


$\widehat{C}$



Starting Point:
t-out-n Secure MPC

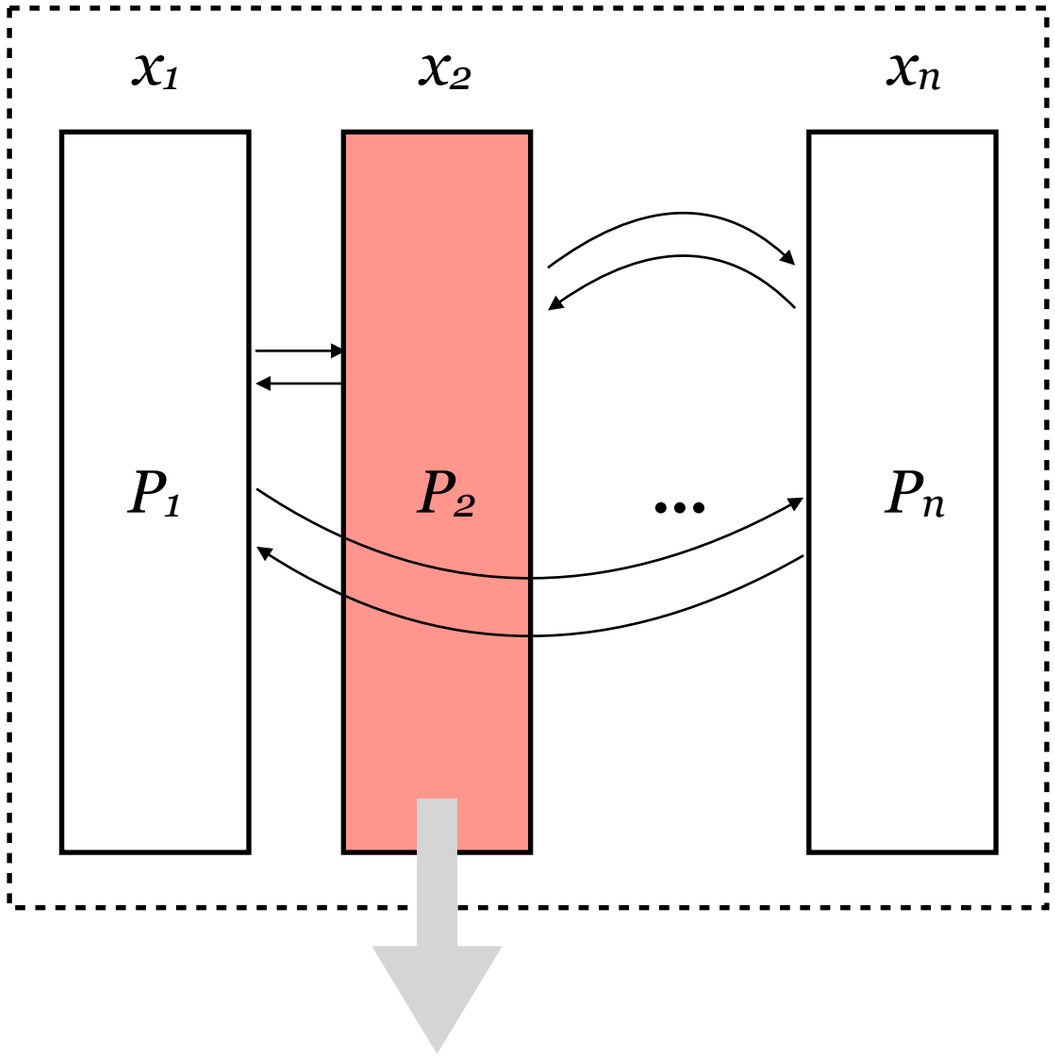
$\Pi(C)$



Passive Corruption of P2

$\equiv$

$\widehat{C}$



Leak State of P2

Leakage-Resilient Circuit Compiler

$$\widehat{C} = \Pi(C')$$

C'

Input: shares of x
Output: shares of C(x)

- *reconstruct x*
- *compute C(x)*
- *share C(x)*

Leakage-Resilient Circuit Compiler

Security?

Leakage-Resilient Circuit Compiler

Security?

If at most t wires leaked then the leakage can be simulated

Leakage-Resilient Circuit Compiler

If at most t wires leaked then the leakage can be simulated

Probability that more than t wires are leaked = Simulation error e

$$\underset{e}{\text{Simulation Error}} \leq exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot p}\right) \qquad \text{(by Chernoff)}$$

Leakage-Resilient Circuit Compiler

If at most t wires leaked then the leakage can be simulated

Probability that more than t wires are leaked = Simulation error e

$$\underset{e}{\text{Simulation Error}} \leq exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot p}\right) \quad \text{(by Chernoff)}$$

If p, $|C|$, t are constants then e is constant

Leakage-Resilient Circuit Compiler

If at most t wires leaked then the leakage can be simulated

Probability that more than t wires are leaked = Simulation error e

$$\underset{e}{\text{Simulation Error}} \leq exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot p}\right) \quad \text{(by Chernoff)}$$

negligible??

If p, $|C|$, t are constants then e is ~~constant~~

(p,e_o)-Base Gadget G_o

Leakage-resilient circuit compiler

with

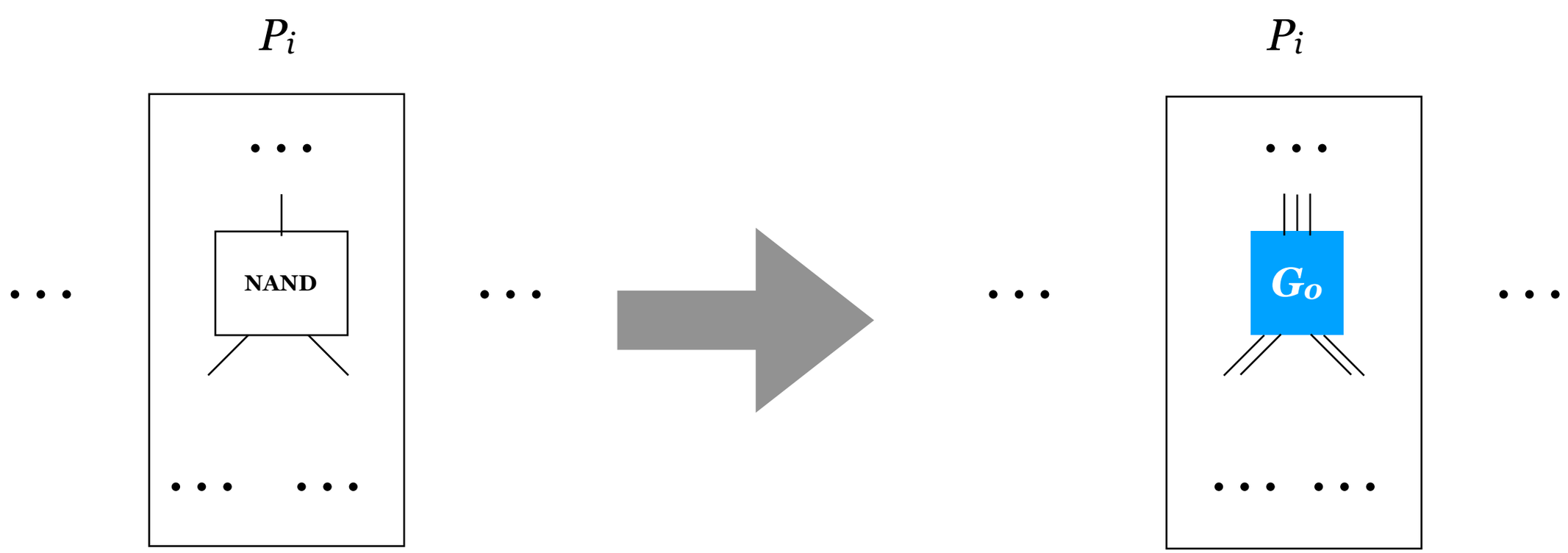
p = constant, $\mathbf{e_o}$ = constant

Reducing the Error

IDEA

- Start with t-out-n secure MPC

- Emulate every gate in t-out-n secure MPC with (p,e_o)-base gadget G_o



Reducing the Error

Security?

Leakage simulatable as long as at most t base gadgets fail

Reducing the Error

Security?

Leakage simulatable as long as at most t base gadgets fail

Probability that more than t base gadgets fail = Simulation error e_1

Reducing the Error

Security?

Leakage simulatable as long as at most t base gadgets fail

Probability that more than t base gadgets fail = Simulation error e_1

$$\text{Simulation Error } e_1 \leq exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot e_0}\right)$$

(by Chernoff)

Size?

Size?

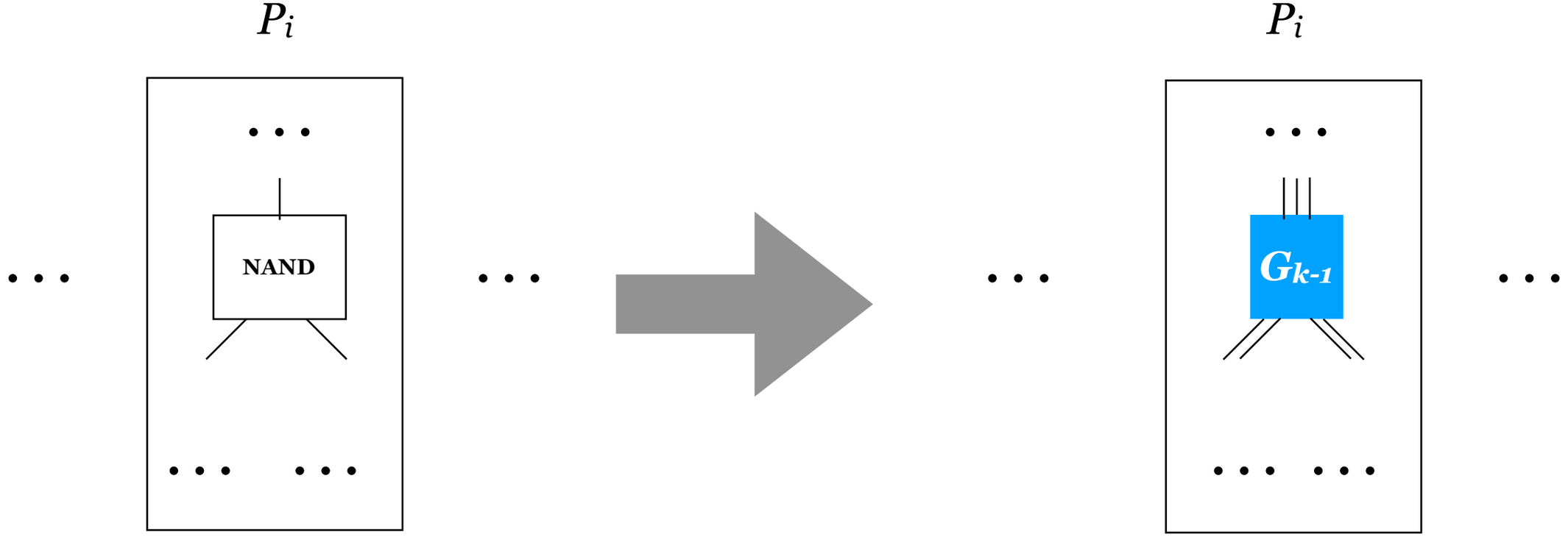
$$|\textbf{Base Gadget}| \times |\Pi(C')|$$

After k steps

IDEA

- Start with t-out-n secure MPC

- Emulate every gate in t-out-n secure MPC with (p,e_{k-1})-gadget G_{k-1}



After k steps: size?

$$\begin{aligned}
\text{Size of } k^{th} \text{ Gadget } G_k &\leq |(k-1)^{th} \textbf{ Gadget}| \times |\Pi(C')| \\
&\leq |(k-2)^{th} \textbf{ Gadget}| \times |\Pi(C')| \times |\Pi(C')| \\
&\cdots \\
&\leq (|\Pi(C')|)^k
\end{aligned}$$

After k steps: size?

$$\text{Size of } k^{th} \text{ Gadget } G_k \leq |(k-1)^{th} \textbf{ Gadget}| \times |\Pi(C')|$$

$$\leq |(k-2)^{th} \textbf{ Gadget}| \times |\Pi(C')| \times |\Pi(C')|$$

$$\cdots$$

$$\leq (|\Pi(C')|)^k$$

$$= exp(O(k))$$ *When |C| is a constant...*

After k steps: error

$$\text{Simulation Error } e_k \leq exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot e_{k-1}}\right)$$

$$\leq exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot e_{k-2}}\right)}\right)$$

$$\leq exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot exp\left(\frac{-(1+t)^2}{12\text{poly}(|C|)\cdot e_{k-3}}\right)}\right)}\right)$$

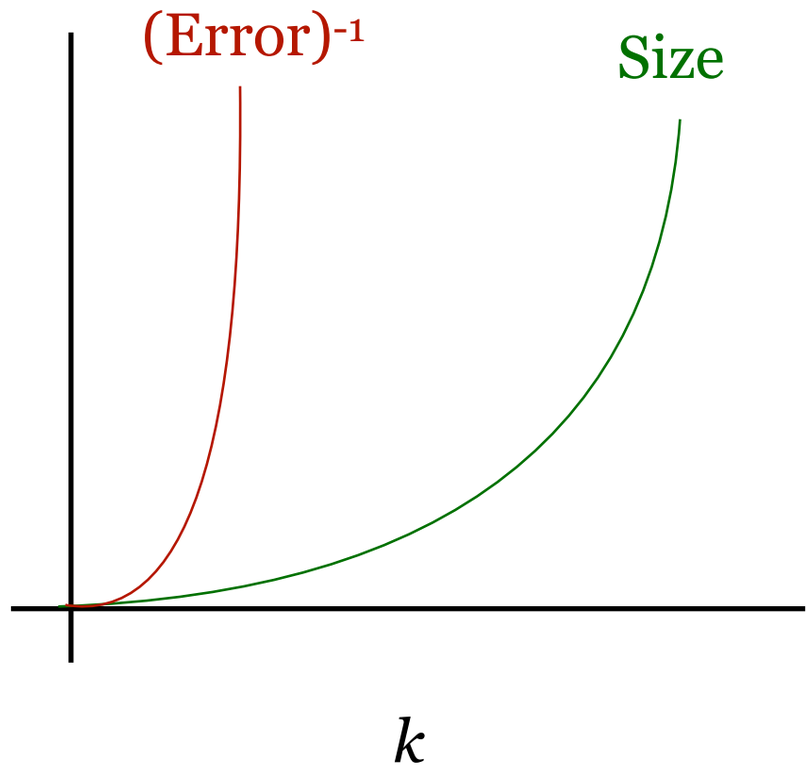
...

$$\leq exp(-2^{O(k)})$$ *When |C| is a constant...*

When |C| is constant,

$$e_k \leq exp(-2^{O(k)})$$

$$\begin{matrix}\text{Size of} \\ k^{th}\text{ Gadget } G_k\end{matrix} \leq exp\left(O(k)\right)$$

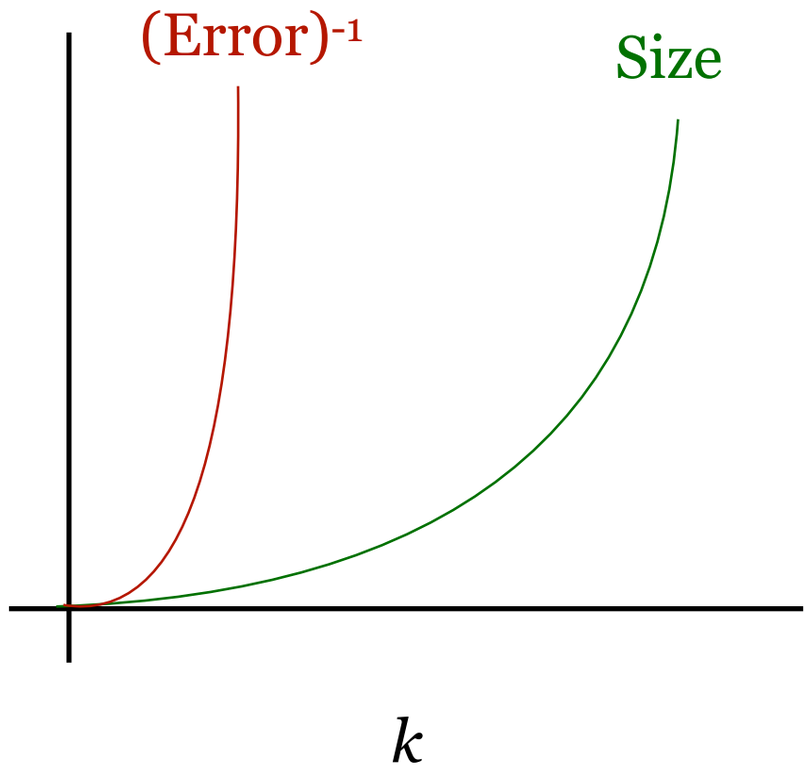


When |C| is constant,

$$\boxed{k = \log(|C|)}$$

$$e_k \leq exp(-2^{O(k)}) = \text{negl}(|C|)$$

$$\text{Size of } k^{th} \text{ Gadget } G_k \leq exp\left(O(k)\right) = \text{poly}(|C|)$$

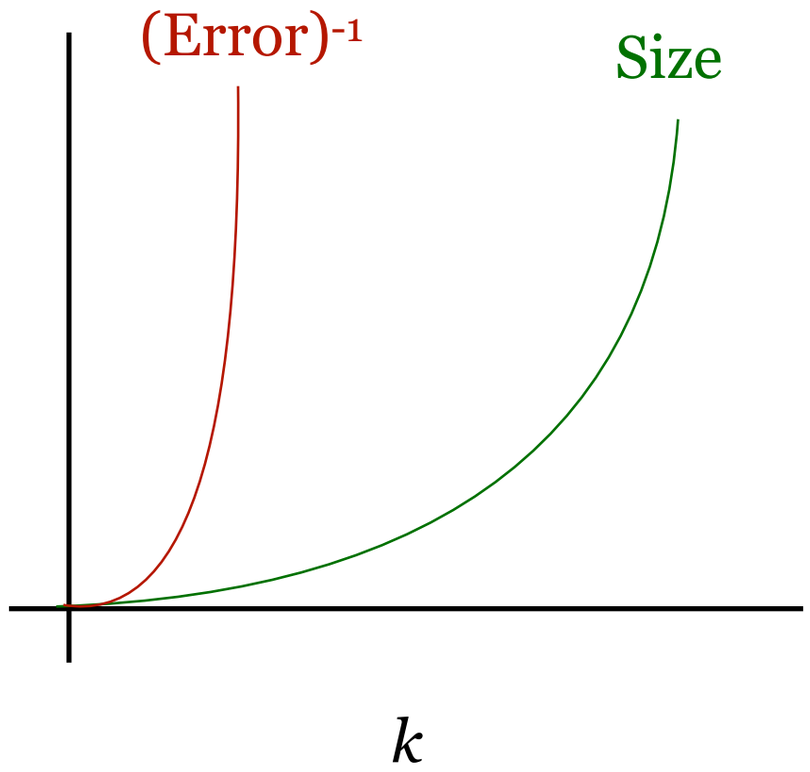


C = NAND

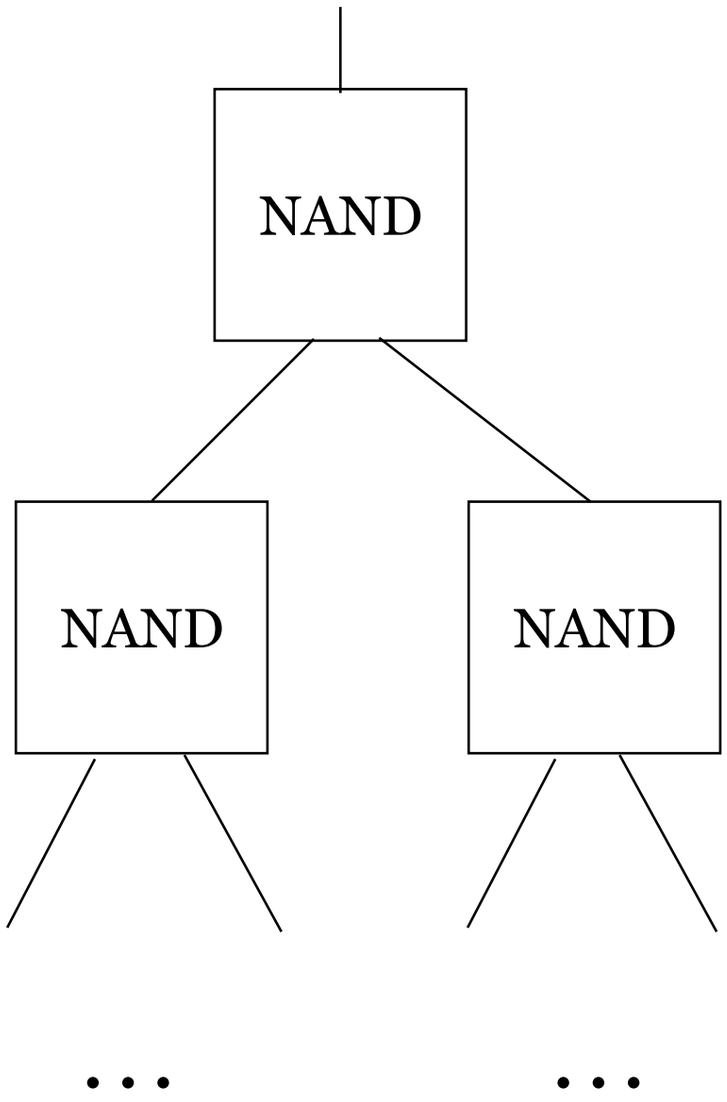
When |C| is constant,

$$e_k \leq exp(-2^{O(k)}) = \text{negl}(|C|)$$

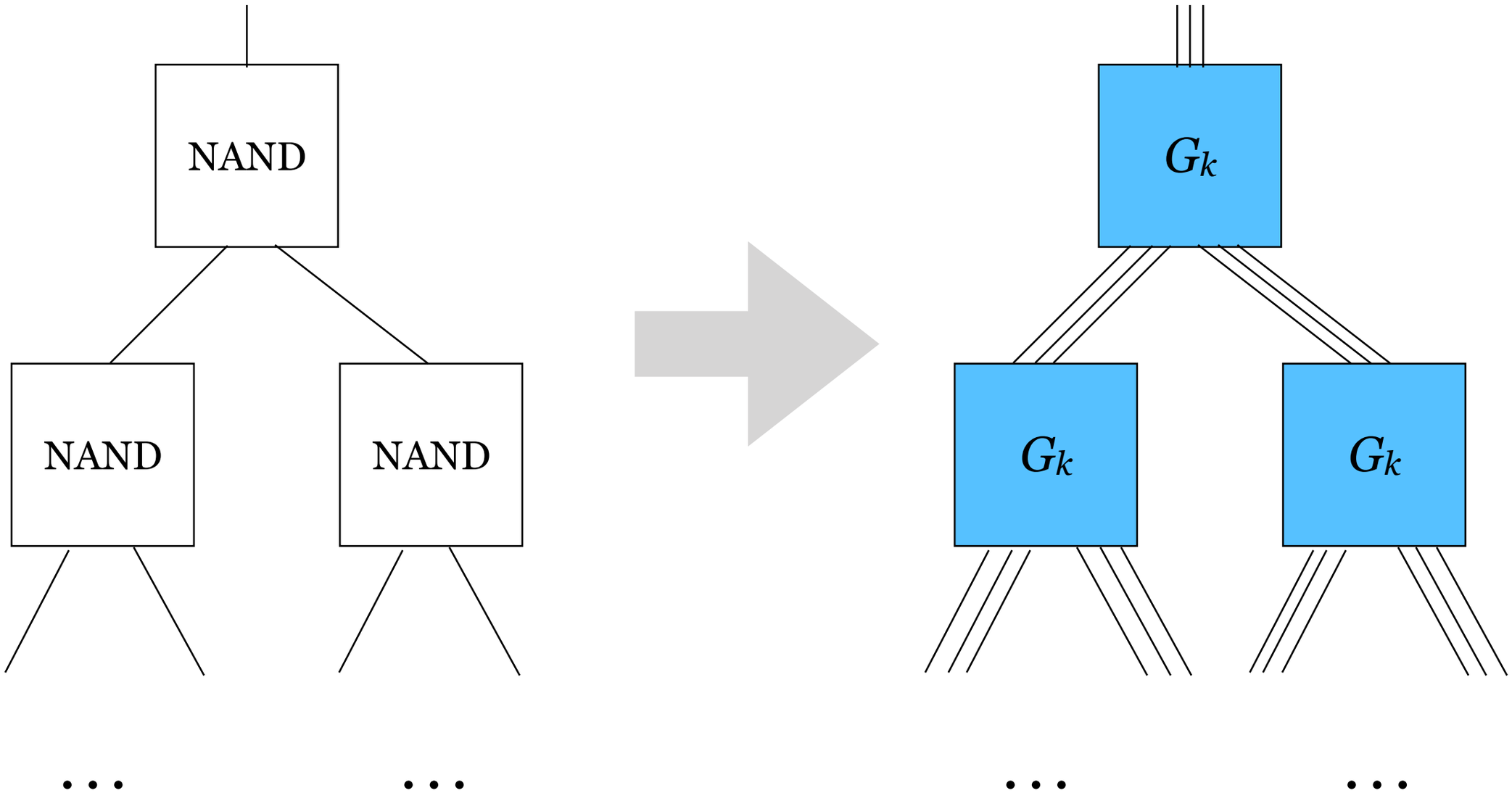
$$\begin{matrix}\text{Size of} \\ k^{th}\text{ Gadget } G_k\end{matrix} \leq exp\left(O(k)\right) = \text{poly}(|C|)$$



To compile a large circuit C,

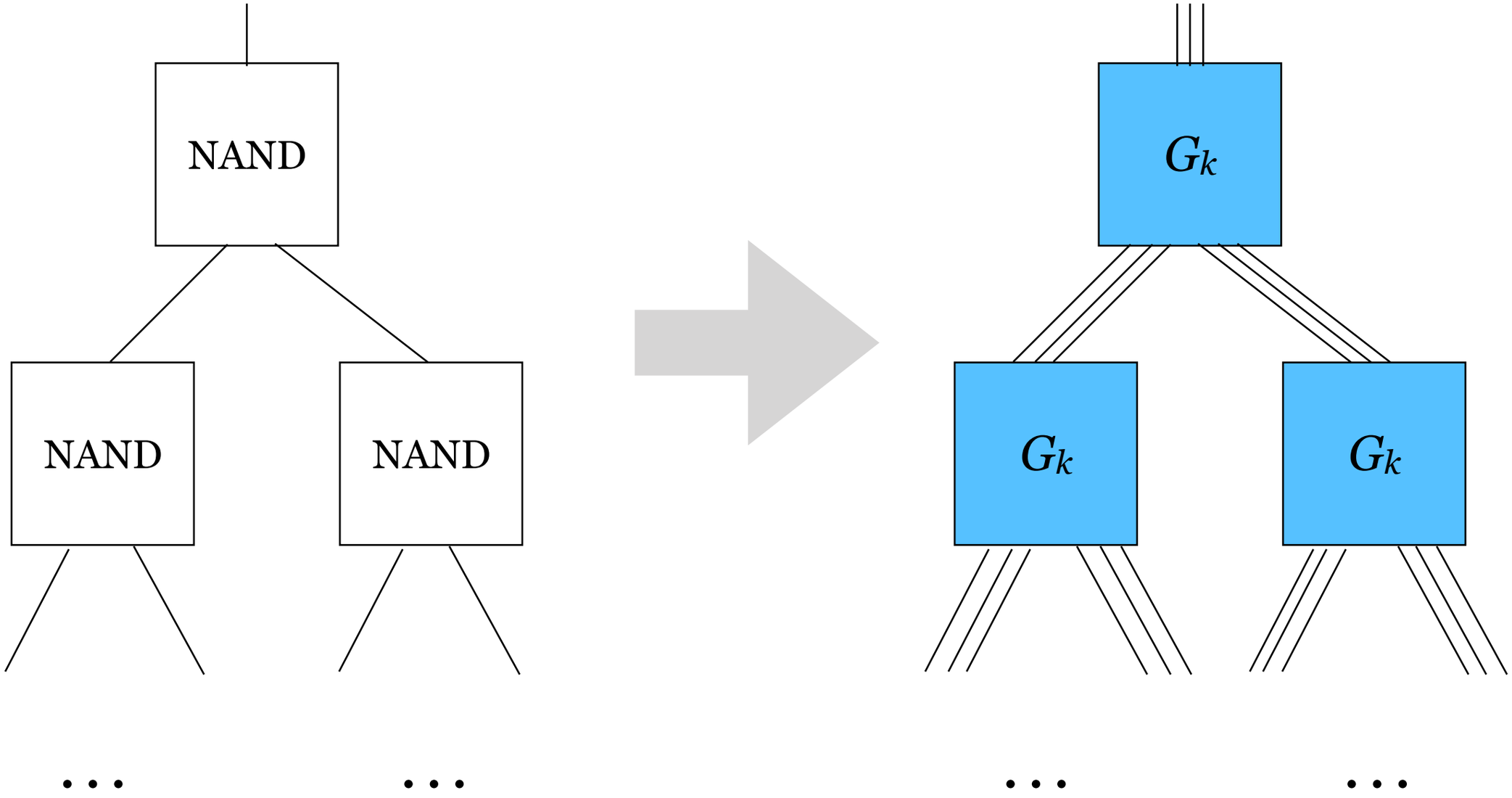


To compile a large circuit C,



Stitch together the gadgets for every gate in the circuit

To compile a large circuit C,



Compositional issues?

Worst-Case Leakage: t-wire probing

- Similar approach: analysis much simpler

Worst-Case Leakage: t-wire probing

- Similar approach: analysis much simpler
- Randomness complexity:

Worst-Case Leakage: t-wire probing

- Similar approach: analysis much simpler
- Randomness complexity:
  - G_O has constant randomness locality

Worst-Case Leakage: t-wire probing

- Similar approach: analysis much simpler
- Randomness complexity:
  - G_0 has constant randomness locality
  - G_k has randomness locality $O(k)$

Worst-Case Leakage: t-wire probing

- Similar approach: analysis much simpler
- Randomness complexity:
  - G_0 has constant randomness locality
  - G_k has randomness locality $O(k)$
  - k=O(log(t))

Worst-Case Leakage: t-wire probing

- Similar approach: analysis much simpler
- Randomness complexity:
  - G_0 has constant randomness locality
  - G_k has randomness locality $O(k)$
  - k=O(log(t))
  - [IKLOPSZ13] “small” randomness locality implies “small” randomness complexity

Conclusion

- Worst-case wire-probing attacks:

  - *Randomness complexity* $t^{1+\varepsilon}$ *(optimal)*

  - *Prior to our work: randomness complexity* $t^{3+\varepsilon}$

- Random wire-probing attacks:

  - *Simpler construction using elementary tools*

Thanks!