Sandia National Laboratories

Exceptional service in the national interest

BEAVER TRIPLES ON-THE-FLY

Jon Berry, **Carolyn Mayer**, Cindy Phillips, Jared Saia

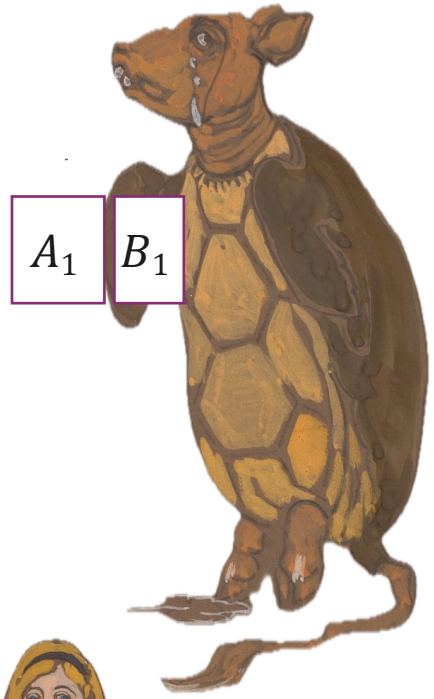
2024 Workshop on Competitive Economics of Cybersecurity

May 18, 2024

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2024-06171PE

MOTIVATION: SECURE MULTIPARTY COMPUTATION



A_1 B_1

A_2 B_2

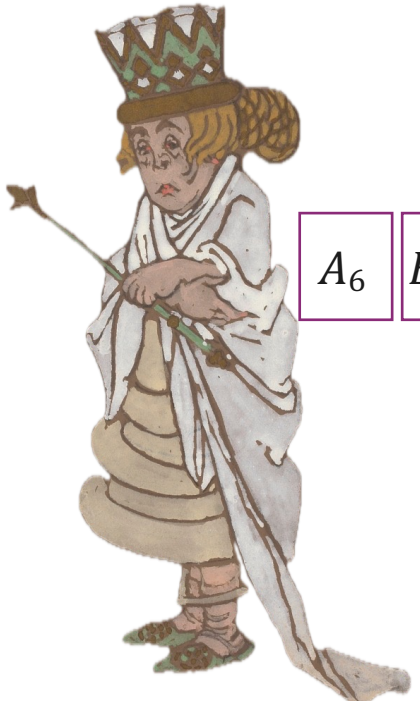
A_3 B_3

A_4 B_4

~~Who stole the tarts?~~

What's AB?

A_5 B_5



A_6 B_6

A_7 B_7

A_8 B_8

SECURE MULTIPARTY COMPUTATION (MPC): ADDITIVE SHARES

We will make **no cryptographic assumptions**.

Computations: **addition** (easy) and **multiplication** (harder)

$\langle s \rangle$ secret s is divided into shares $s_1, s_2, \ldots, s_n$ such that $\sum_{i=1}^{n} = s$ and the individual s_i reveal nothing about s

$\langle s + t \rangle$ add shares locally $(s_1 + t_1) + (s_2 + t_2) + (s_3 + t_3) = s + t$

$\langle s \cdot t \rangle$ ~~multiply shares locally?~~ $(s_1 t_1) + (s_2 t_2) + (s_3 t_3) \neq st$

s_1
t_1

s_2
t_2

s_3
t_3

SECURE MULTIPARTY COMPUTATION (MPC): SHAMIR SHARES

We will make **no cryptographic assumptions**.

Computations: **addition** (easy) and **multiplication** (harder)

$\langle s \rangle$ Define a degree $n-1$ polynomial f with constant term s

$\langle s+t \rangle$ add shares locally $(f+g)(x_i) = f(x_i) + g(x_i)$

$\langle s \cdot t \rangle$ multiply shares locally? ⚠ Constant term is right, but the degree increases

$f(x_1)$

$g(x_1)$

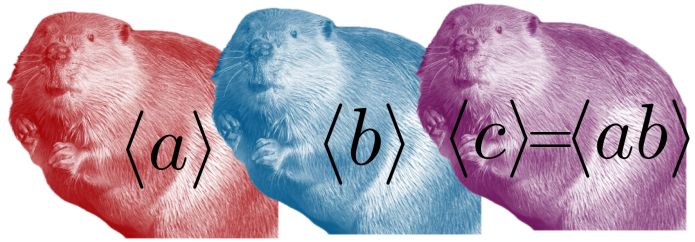
$f(x_2)$

$g(x_2)$

$f(x_3)$

$g(x_3)$

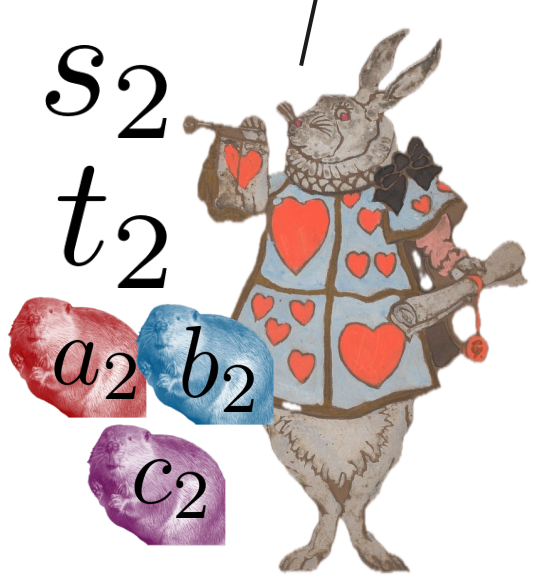
EXAMPLE: MULTIPLICATION WITH BEAVER TRIPLES

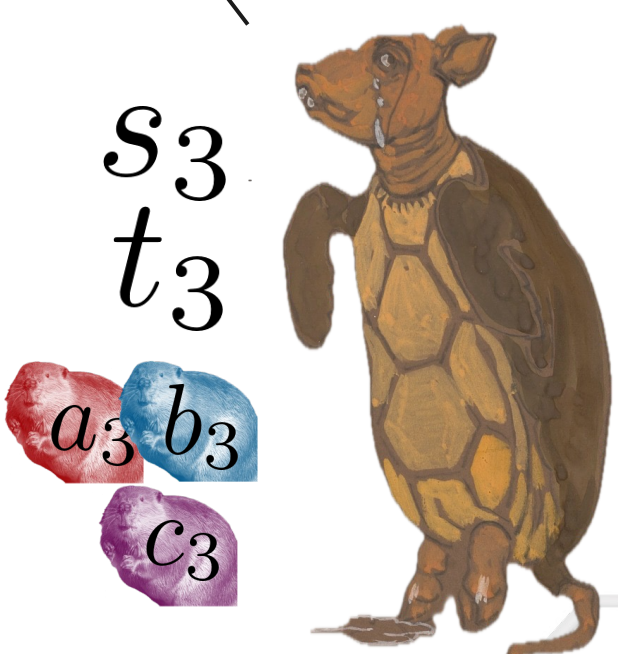


We will make **no cryptographic assumptions**.

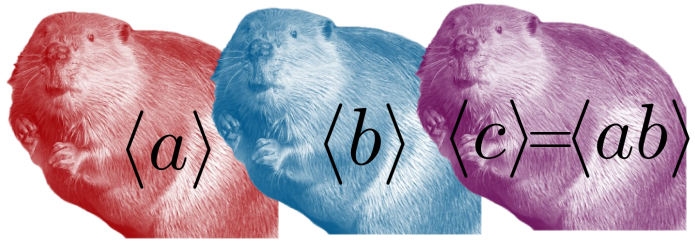
Reveal:

$$\alpha = (s_1 - a_1) + (s_2 - a_2) + (s_3 - a_3)$$





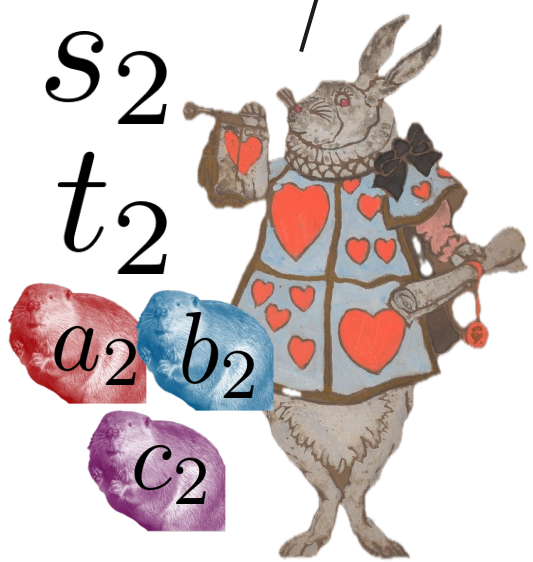
EXAMPLE: MULTIPLICATION WITH BEAVER TRIPLES

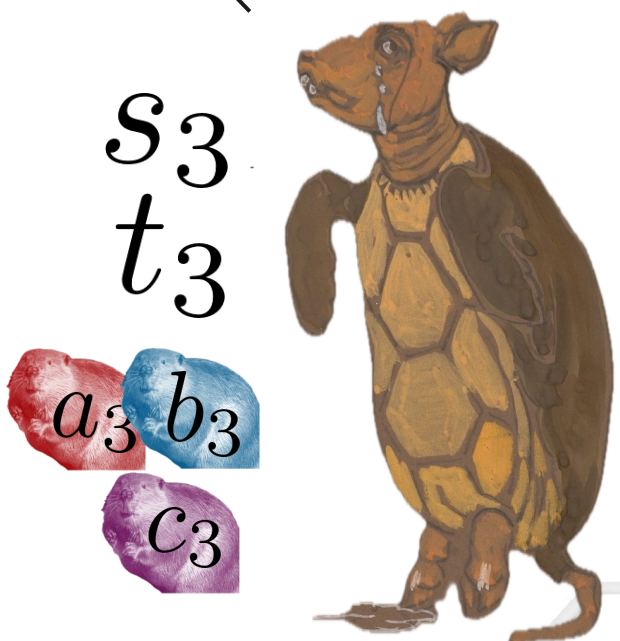


We will make **no cryptographic assumptions**.

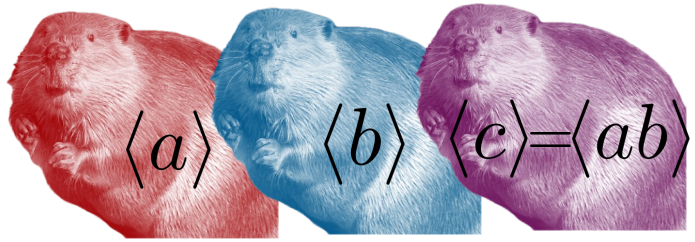
Reveal:

$$\beta = (t_1 - b_1) + (t_2 - b_2) + (t_3 - b_3)$$





EXAMPLE: MULTIPLICATION WITH BEAVER TRIPLES

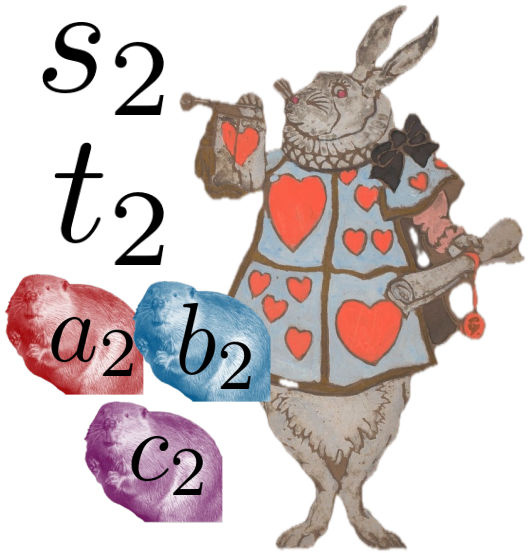


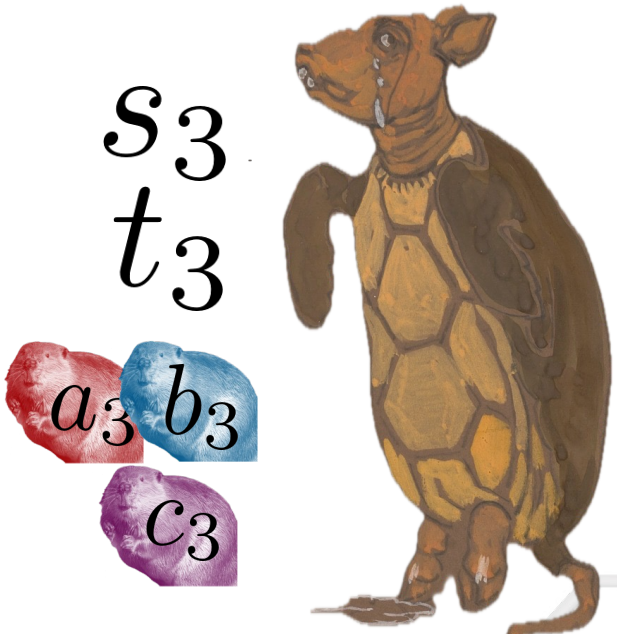
We will make **no cryptographic assumptions**.

Known: $\alpha = s - a, \beta = t - b$

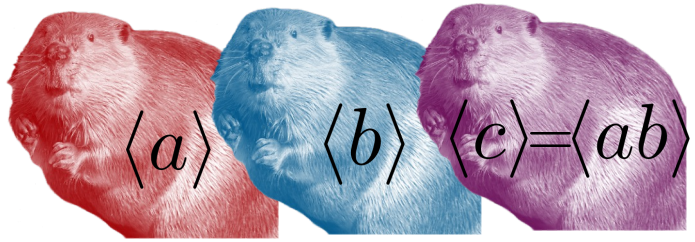
Local computation:

$$u_i = \beta s_i + \alpha b_i + c_i$$





EXAMPLE: MULTIPLICATION WITH BEAVER TRIPLES



We will make **no cryptographic assumptions**.

Known: $\alpha = s - a, \beta = t - b$

Global Impact:

$$\begin{aligned} u_1 + u_2 + u_3 &= (\beta s_1 + \alpha b_1 + c_1) + (\beta s_2 + \alpha b_2 + c_2) + (\beta s_3 + \alpha b_3 + c_3) \\ &= \beta s + \alpha b + c \\ &= (t - b)s + (s - a)b + ab \\ &= st - sb + sb - ab + ab \\ &= st \end{aligned}$$

u_1

u_2

u_3

BEAVER TRIPLES

We will make **no cryptographic assumptions**.

Given $\langle x \rangle$ and $\langle y \rangle$, how can we get $\langle xy \rangle$?

Beaver triple: $\langle a \rangle, \langle b \rangle, \langle c \rangle = \langle ab \rangle$
(D. Beaver `91)

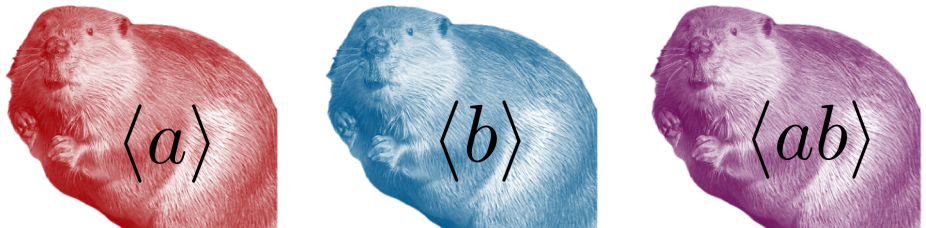
Reveal $\alpha = x - a, \ \beta = y - b$

Set $z_i = \beta x_i + \alpha b_i + c_i$

Then

$$z = (y - b)x + (x - a)b + ab = xy - bx + bx - ab + ab = xy$$

Problem: How do we get the Beaver triple?



FINDING BEAVER TRIPLES

⚠ Beaver triples can be expensive to compute

Precomputation

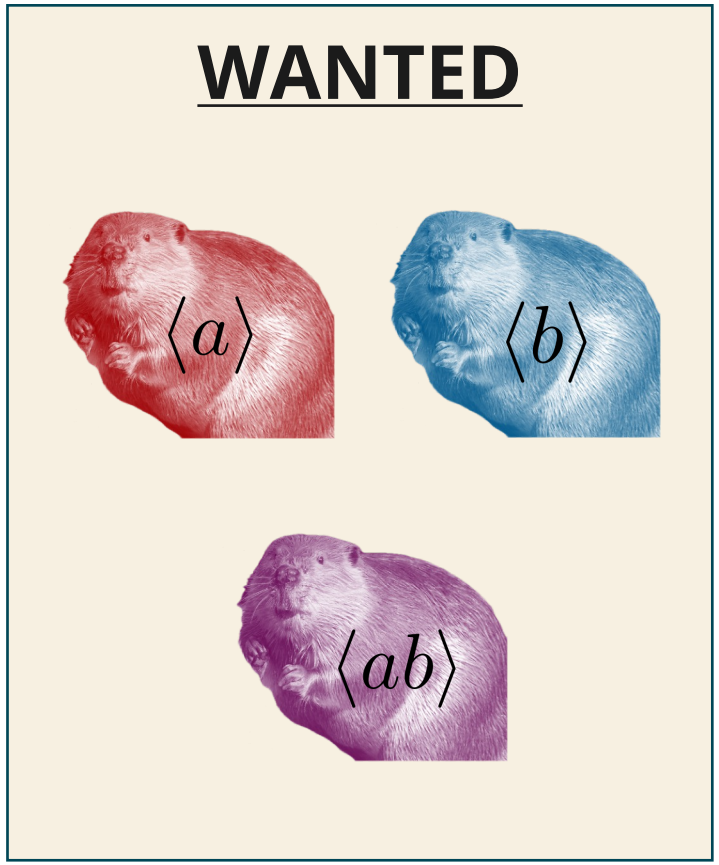
- Oblivious Transfer (e.g., TinyOT)
- Homomorphic encryption (e.g., SPDZ)
- ⋮

⚠ Beaver triples cannot be reused

⚠ Beaver triples cannot be used with dynamically changing groups

Triples as a service (e.g. TaaS, CrypTen)

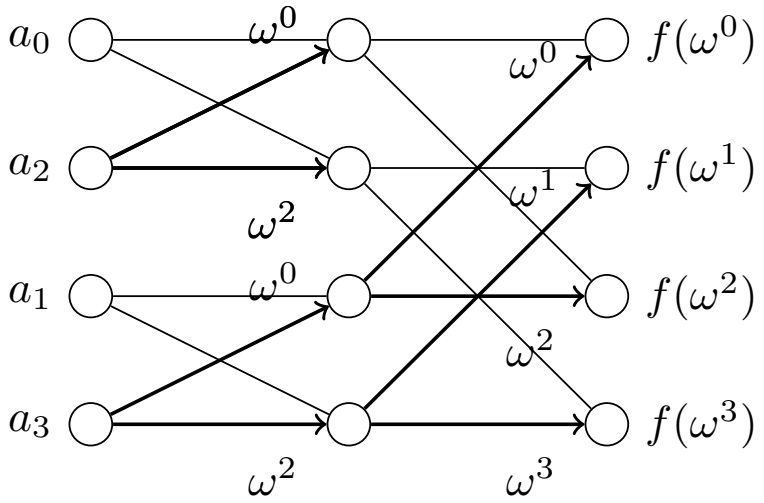
⚠ We don't always want to rely on a third party



Goal: Low-communication, non-cryptographic algorithm

FAST FOURIER TRANSFORM

- **Fast Fourier Transform (FFT)**
  Compute $f(\omega^0), f(\omega^1), \ldots, f(\omega^{n-1})$ from $a_0, a_1, \ldots, a_n - 1$, or vice versa
  with complexity $O(n \log n)$
  where $f(x) = \sum_{k=0}^{n-1} a_k x^k$, and $\omega = e^{\frac{2\pi i}{n}}$ (ω is an n^{th} root of unity)

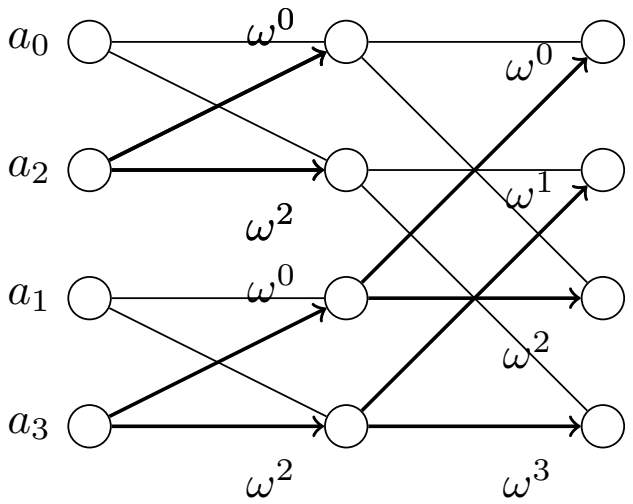


- $f(\omega^0), f(\omega^1), \ldots, f(\omega^{n-1})$ are **Shamir shares** of a_0.
- $f(\omega^0), f(\omega^1), \ldots, f(\omega^{n-1})$ are also **additive shares** of na_0.

$$\sum_{j=0}^{n-1} \omega^{jk} = 0, \quad k \in \{1, 2, \ldots, n-1\}$$

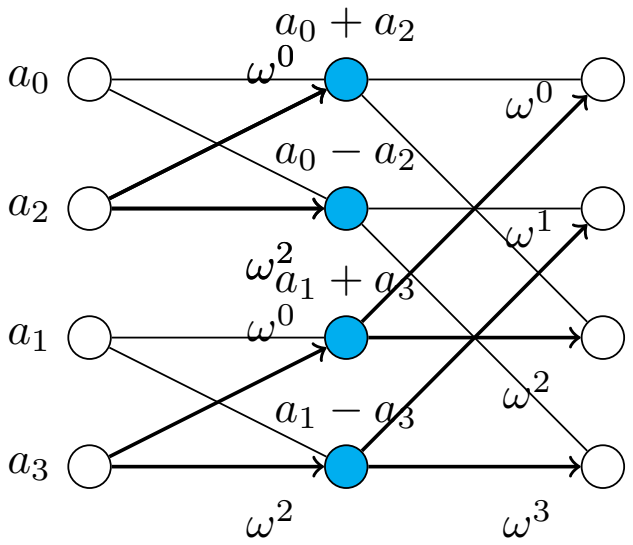
FAST FOURIER TRANSFORM EXAMPLE, $n = 4$

$$\omega^2 = -1,\ \omega^4 = 1$$



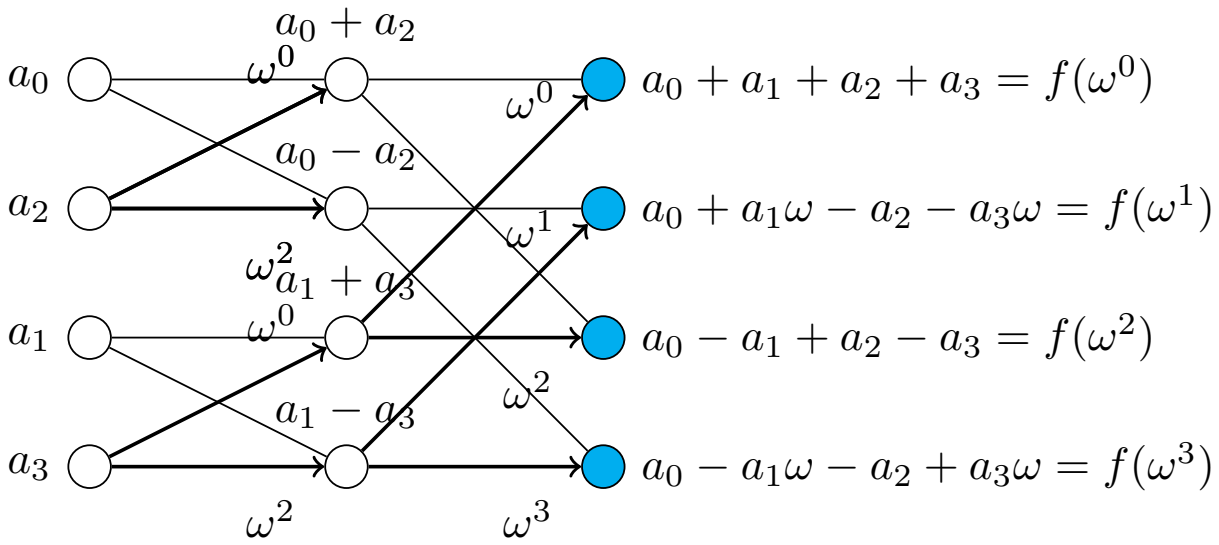
FAST FOURIER TRANSFORM EXAMPLE, $n = 4$

$$\omega^2 = -1,\ \omega^4 = 1$$



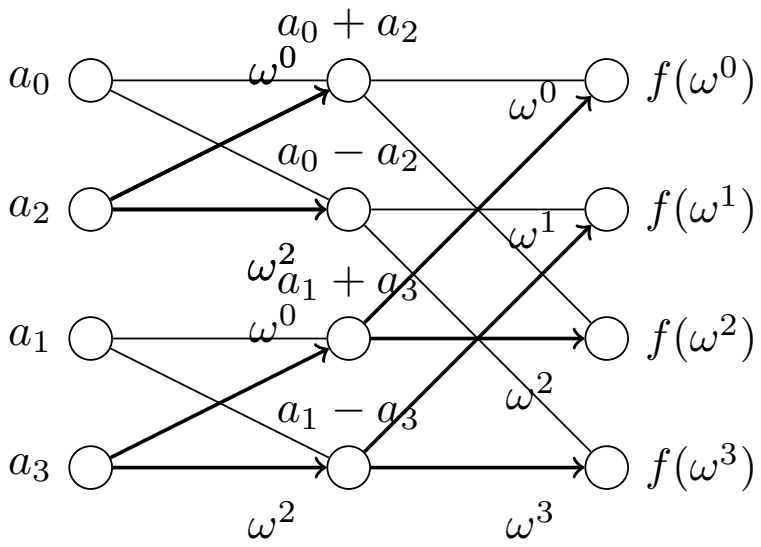
FAST FOURIER TRANSFORM EXAMPLE, $n = 4$

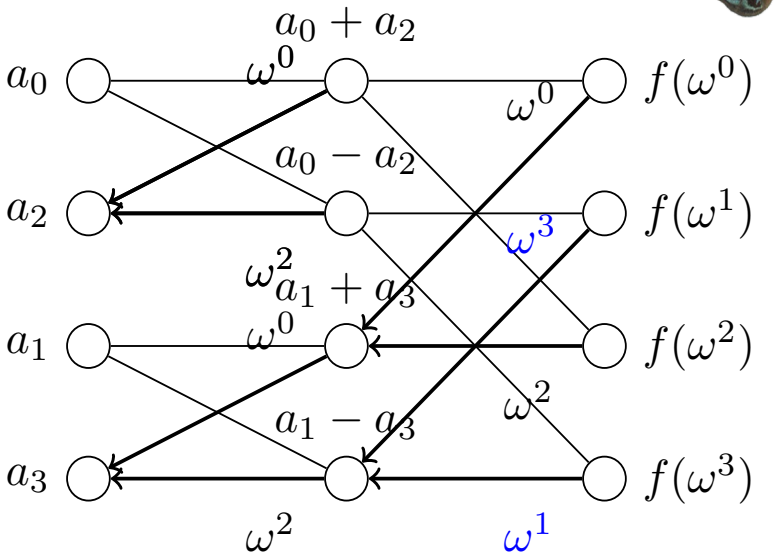
$$\omega^2 = -1,\ \omega^4 = 1$$



FAST FOURIER TRANSFORM EXAMPLE, $n = 4$

$$\omega^2 = -1,\ \omega^4 = 1$$



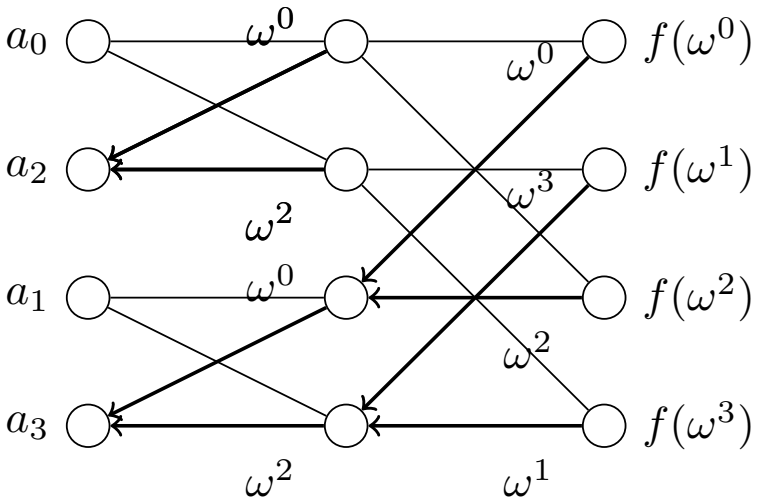


Additive shares:

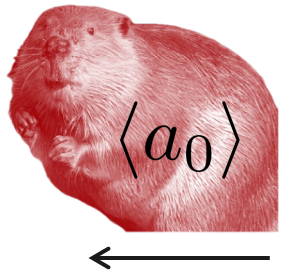
$$f(\omega^0) + f(\omega^1) + f(\omega^2) + f(\omega^3)$$
$$= (a_0 + a_1 + a_2 + a_3) + (a_0 + a_1\omega + a_2\omega^2 + a_3\omega^3) + (a_0 + a_1\omega^2 + a_2\omega^4 + a_3\omega^6) + (a_0 + a_1\omega^3 + a_2\omega^6 + a_3\omega^9)$$
$$= a_0(1+1+1+1) + a_1(1 + \omega + \omega^2 + \omega^3) + a_2(1 + \omega^2 + \omega^4 + \omega^6) + a_3(1 + \omega^3 + \omega^6 + \omega^9)$$
$$= a_0(1+1+1+1) + a_1(1 + \omega - 1 - \omega) + a_2(1 - 1 + 1 - 1) + a_3(1 + -\omega - 1 + \omega)$$
$$= 4a_0$$

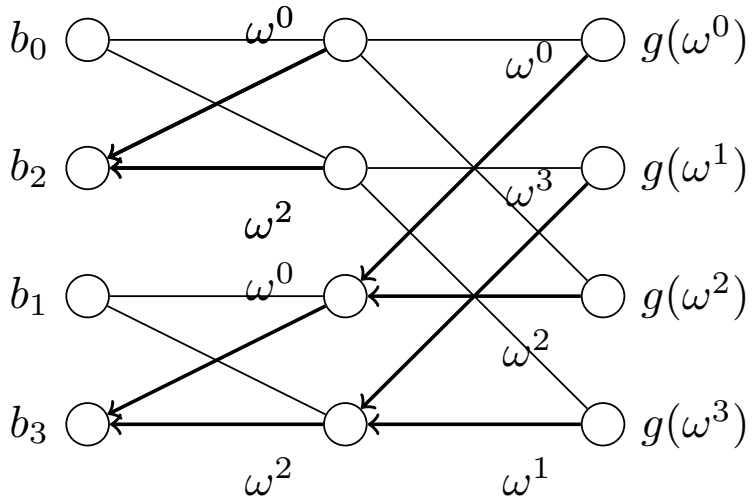
Idea: Use FFTs to get a Beaver Triple

BEAVER TRIPLES FROM THE FAST FOURIER TRANSFORM: A AND B

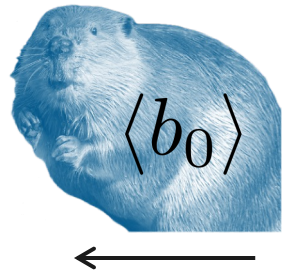


Random





Random



?

$f(\omega^k) \cdot g(\omega^k)$ are Shamir shares of $a_0 b_0$

⚠ ... but has $f \cdot g(x)$ degree $2n - 2 > n - 1$

⚠ $f(\omega^k) \cdot g(\omega^k)$ are not additive shares of $a_0 b_0$

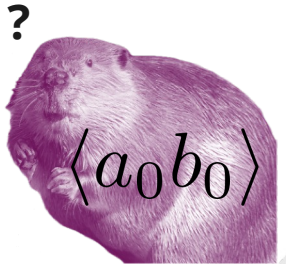
(NON)EXAMPLE BEAVER TRIPLES FROM THE FFT, $n = 4$

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \qquad g(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3$$

$$(f \cdot g)(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 + c_5 x^5 + c_6 x^6$$

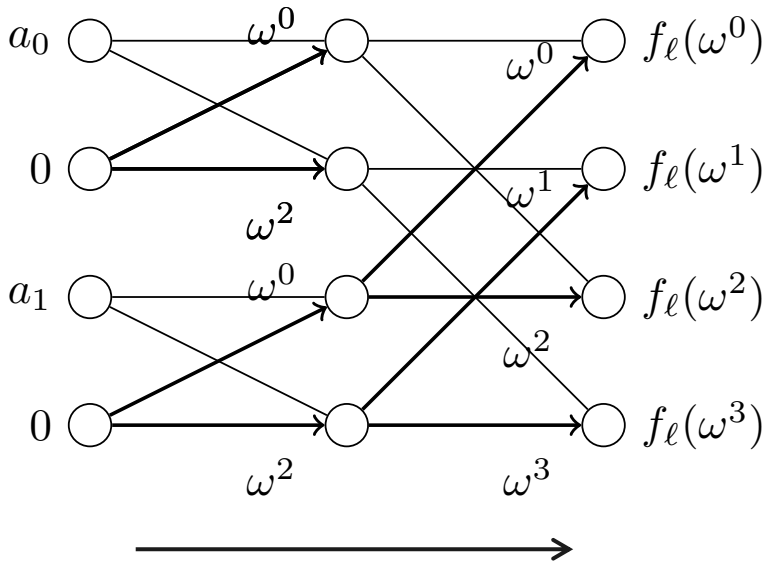
$$\begin{aligned}\sum_{k=0}^{n-1} (f \cdot g)(\omega^k) &= c_0(1+1+1+1) + c_1(1+\omega+\omega^2+\omega^3) + c_2(1+\omega^2+\omega^4+\omega^6) + c_3(1+\omega^3+\omega^6+\omega^9) \\ &\quad + c_4(1+\omega^4+\omega^8+\omega^{12}) + c_5(1+\omega^5+\omega^{10}+\omega^{15}) + c_6(1+\omega^6+\omega^{12}+\omega^{18}) \\ &= 4c_0 + 4c_4 \\ &= 4a_0b_0 + 4(a_1b_3 + a_2b_2 + a_3b_1)\end{aligned}$$

⚠ $f(\omega^k) \cdot g(\omega^k)$ are not additive shares of $a_0 b_0$

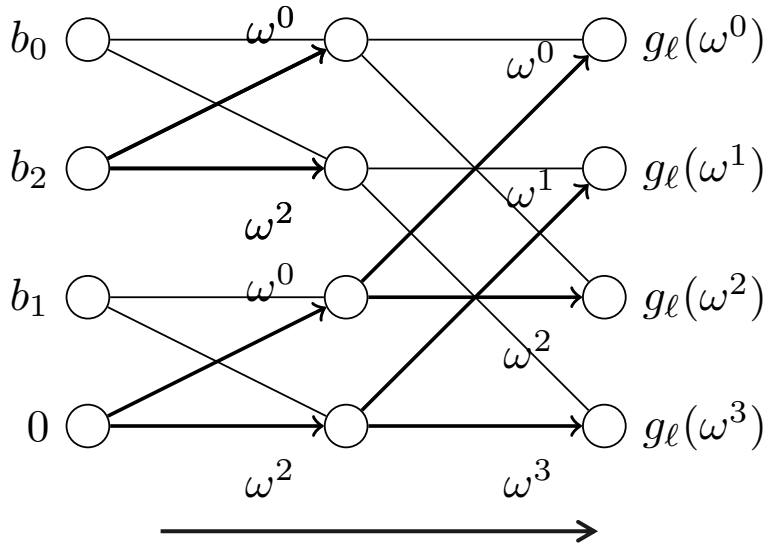


BEAVER TRIPLES FROM THE FAST FOURIER TRANSFORM: AB

$f_\ell(x)$: lower $\frac{n}{2}$ terms of f



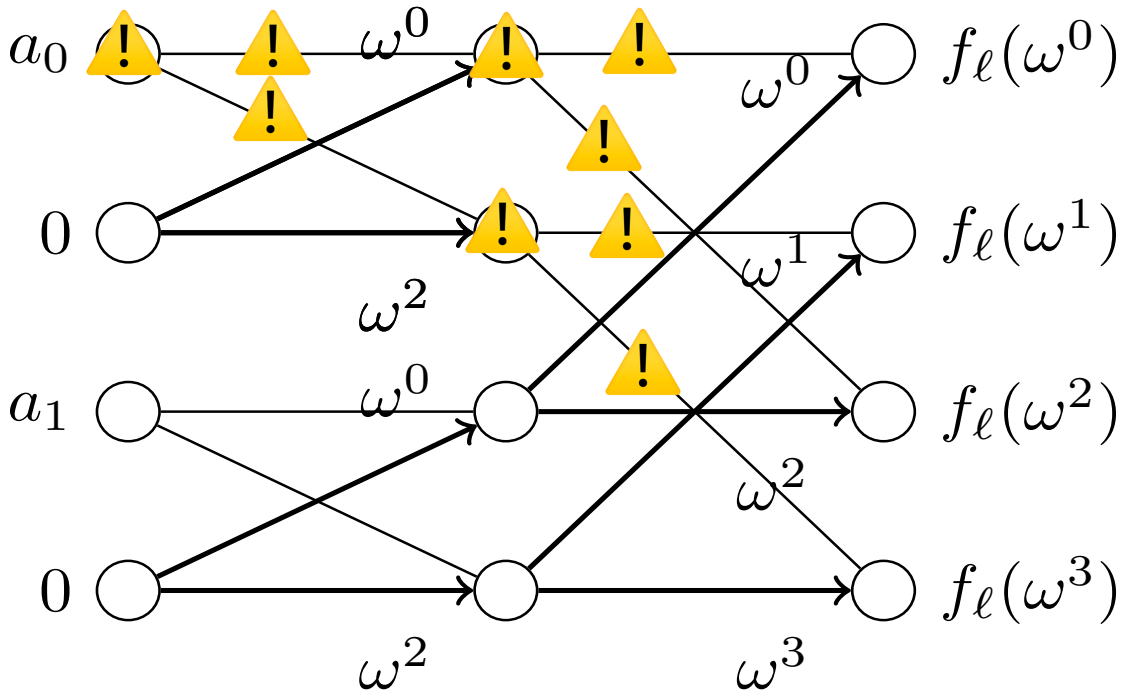
$g_\ell(x)$: lower $\frac{n}{2}+1$ terms of g



$f_\ell(\omega^k) \cdot g_\ell(\omega^k)$ are Shamir shares of $a_0 b_0$

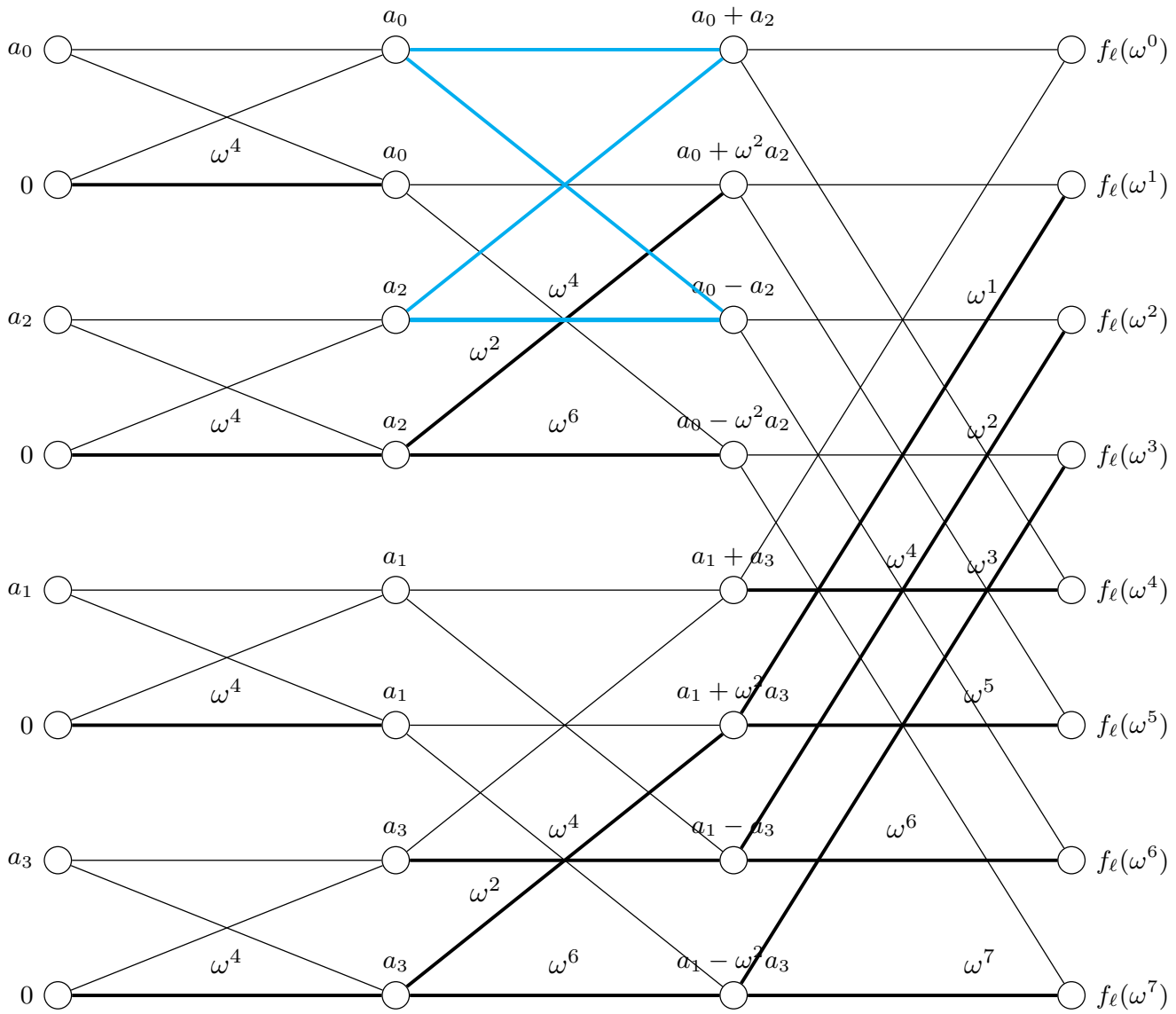
- ✓ $f_\ell \cdot g_\ell(x)$ has degree $n-1$
- ✓ $f(\omega^k) \cdot g(\omega^k)$ add to $n a_0 b_0$

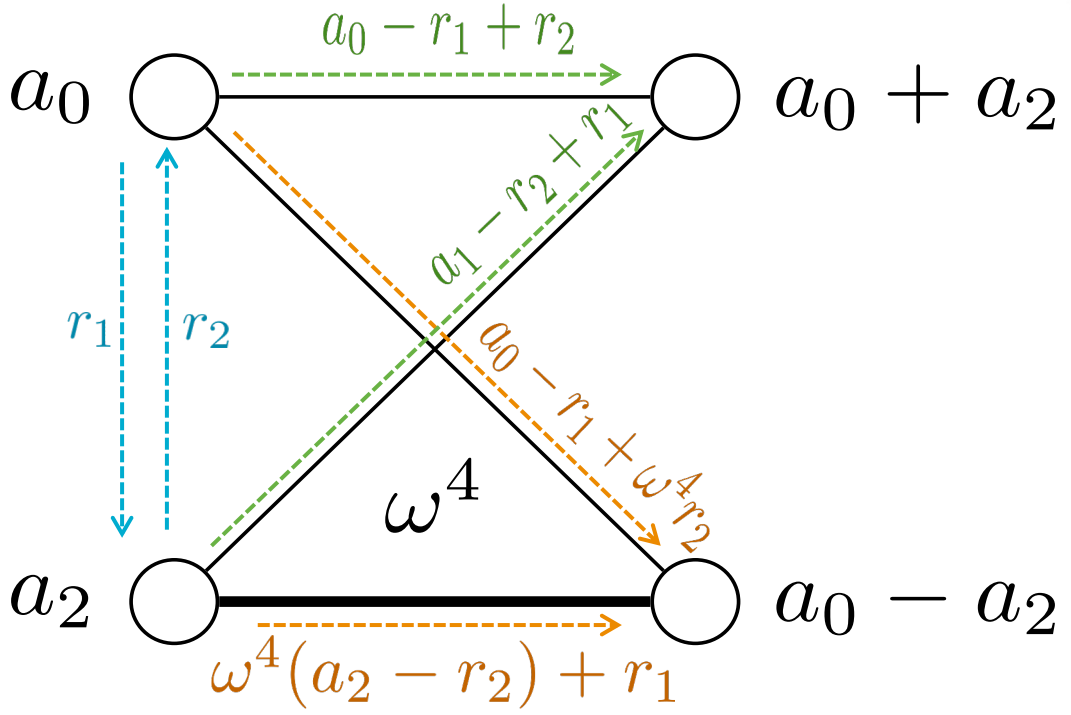
KEEPING a_0 SECRET: PROBLEM



We need to make sure that no individual learns a_0

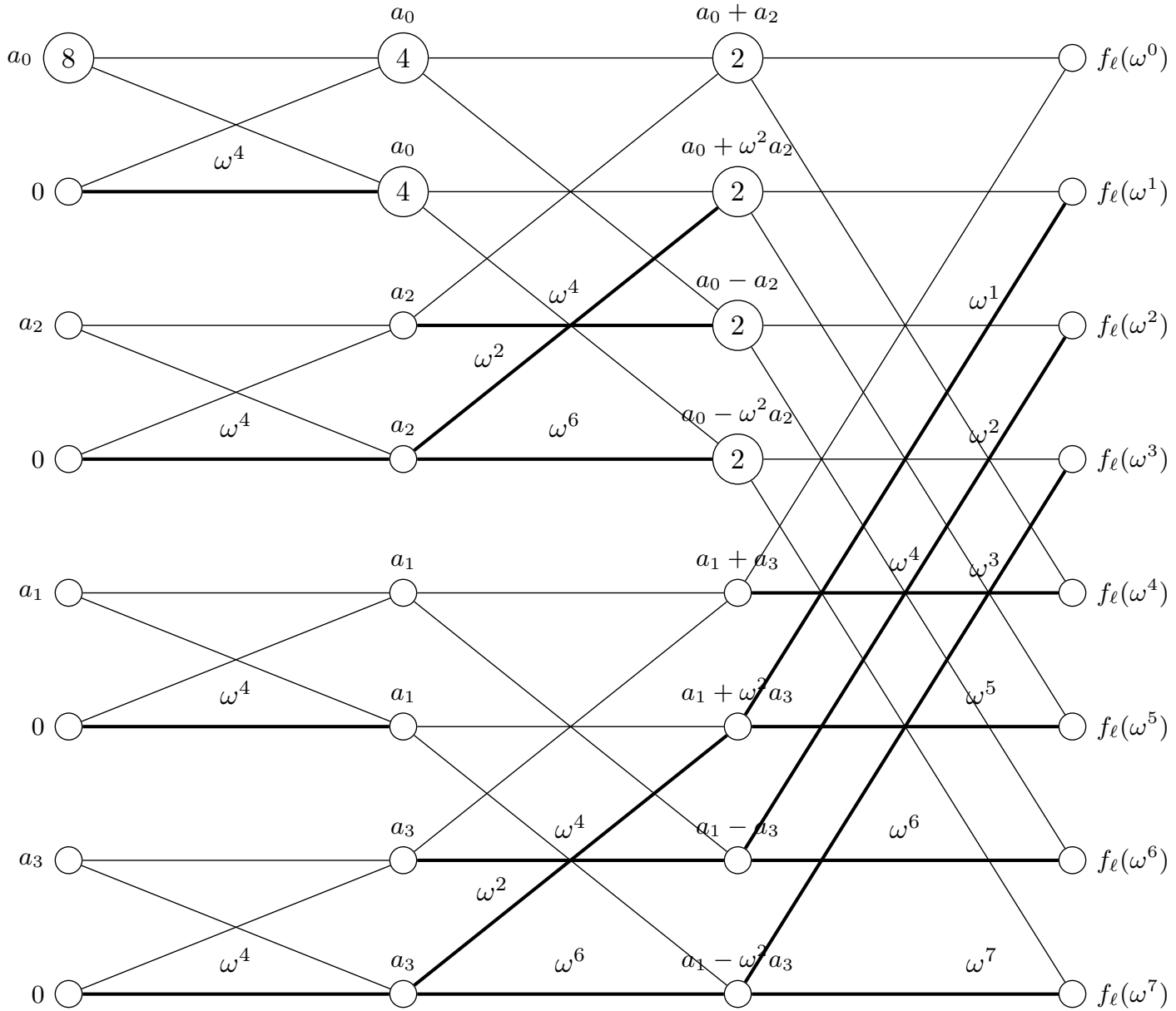
KEEPING a_0 SECRET: MESSAGES SENT





a_0 is never sent as a message

KEEPING a_0 SECRET: ADDING NODE WEIGHTS

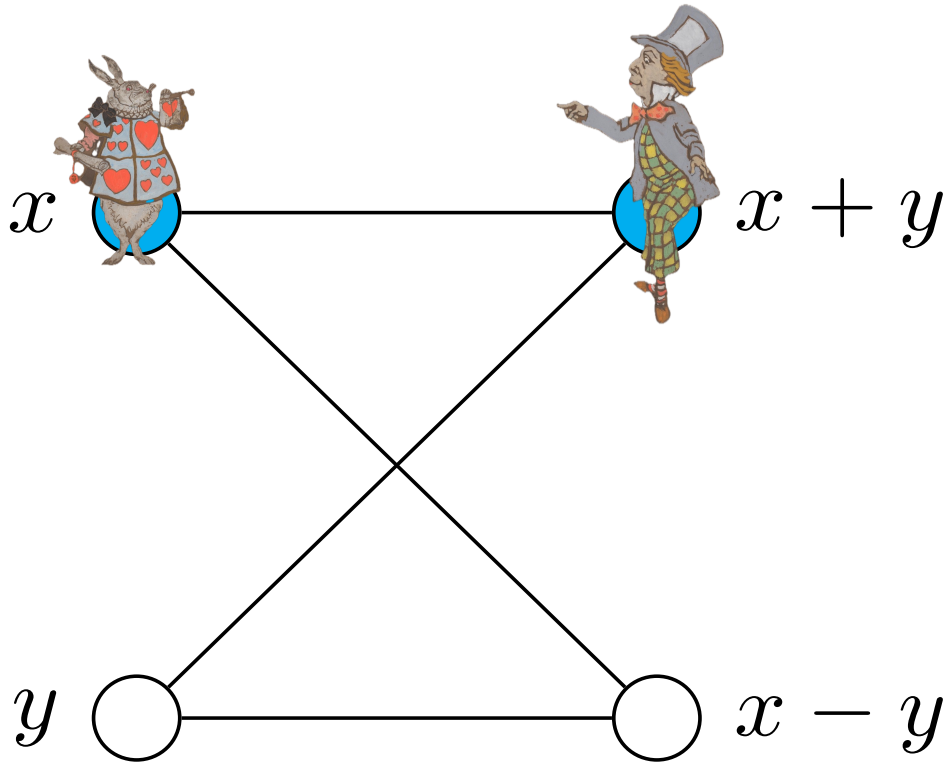


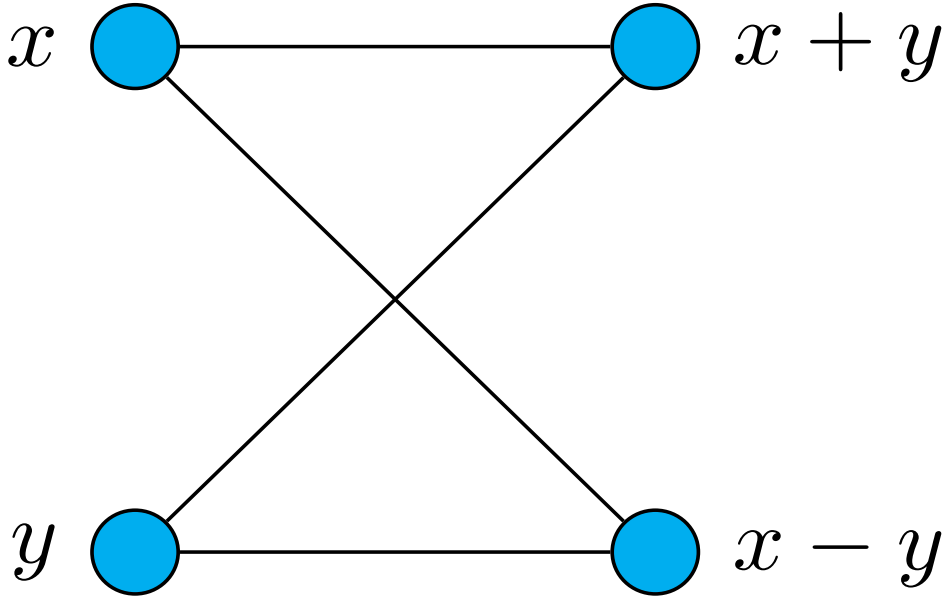
No weight one node sees a_0

ONGOING WORK: COALITION RESISTANCE

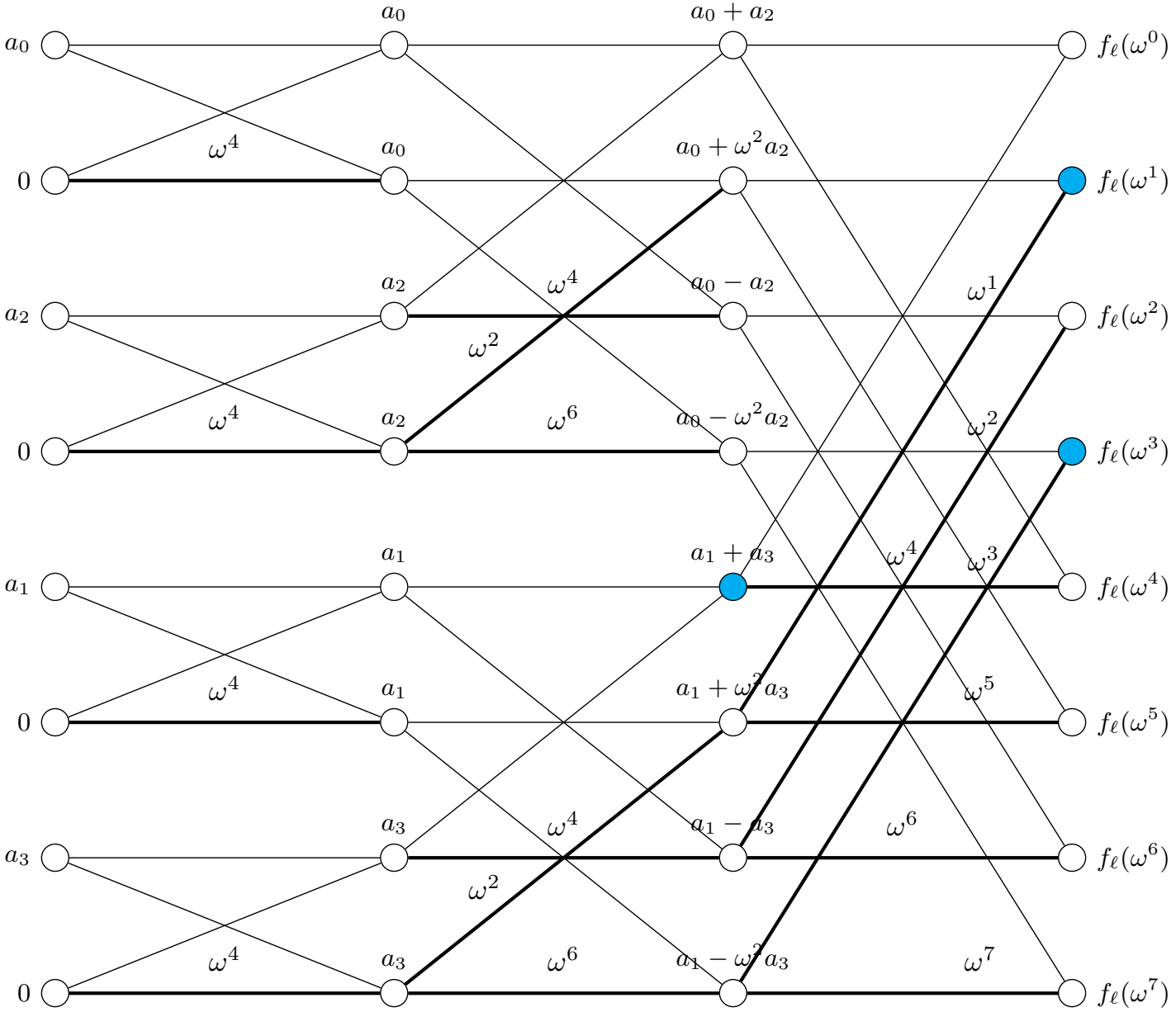
HONEST-BUT-CURIOUS COALITIONS

- Participants follow the rules, but learn from any information they see
- Some collections of participants pool information





HONEST-BUT-CURIOUS COALITIONS: NON BUTTERFLY EXAMPLE, $n = 8$



Known:

$$a_0 + a_1\omega + a_2\omega^2 + a_3\omega^3$$

$$a_0 + a_1\omega^3 + a_2\omega^6 + a_3\omega^9 = a_0 + a_1\omega^3 - a_2\omega^2 + a_3\omega$$

$$a_1 + a_3$$

$$\downarrow$$

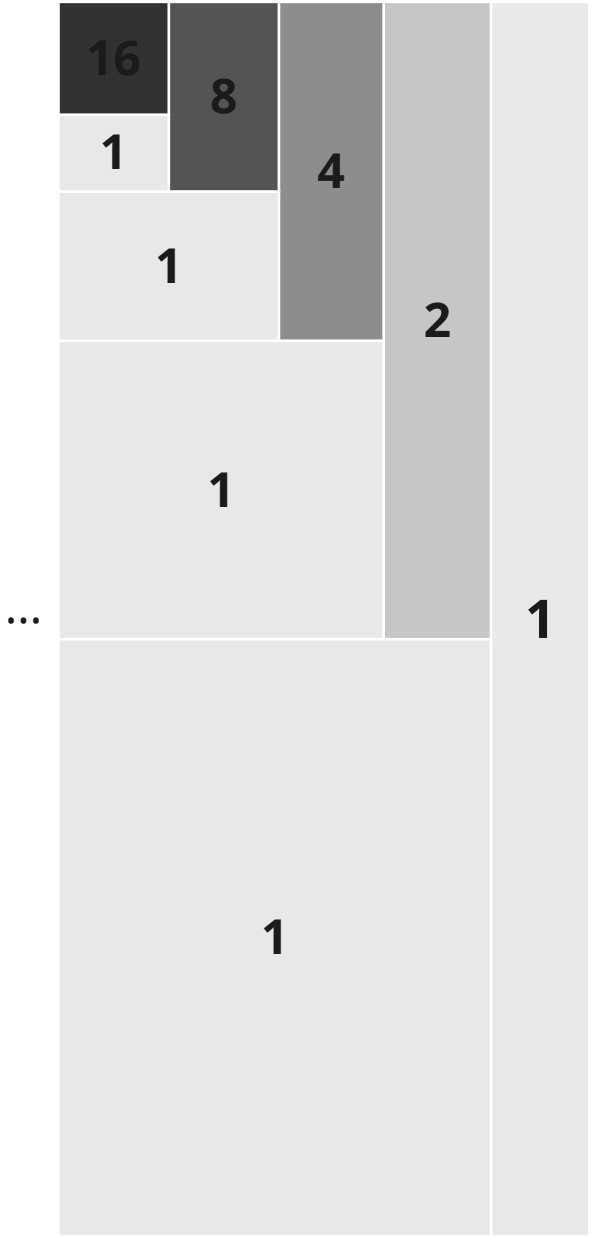
$$2a_0 + (\omega + \omega^3)(a_1 + a_3)$$

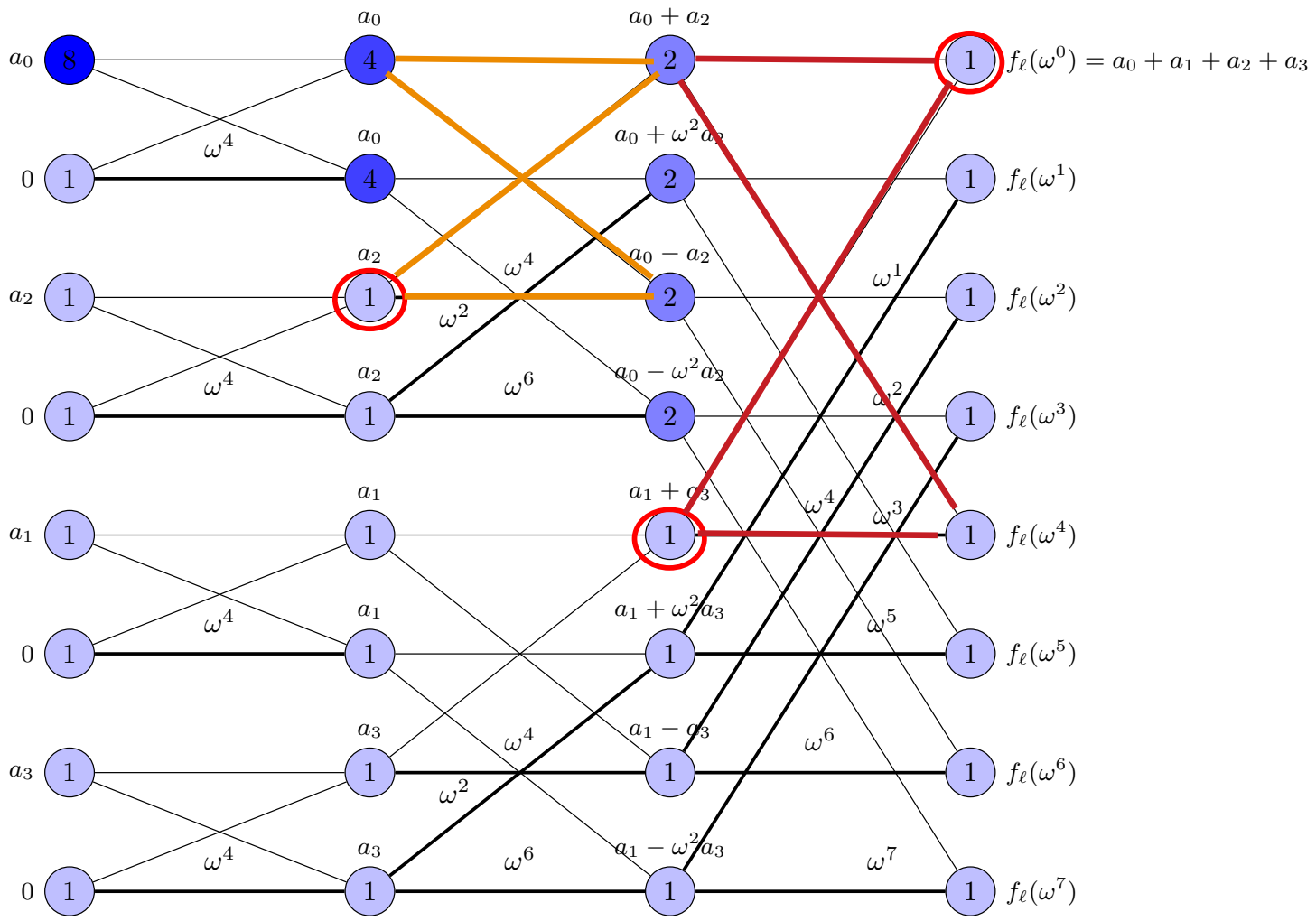
$$a_1 + a_3$$

$$\downarrow$$

$$a_0$$

EXAMPLE: LOW WEIGHT

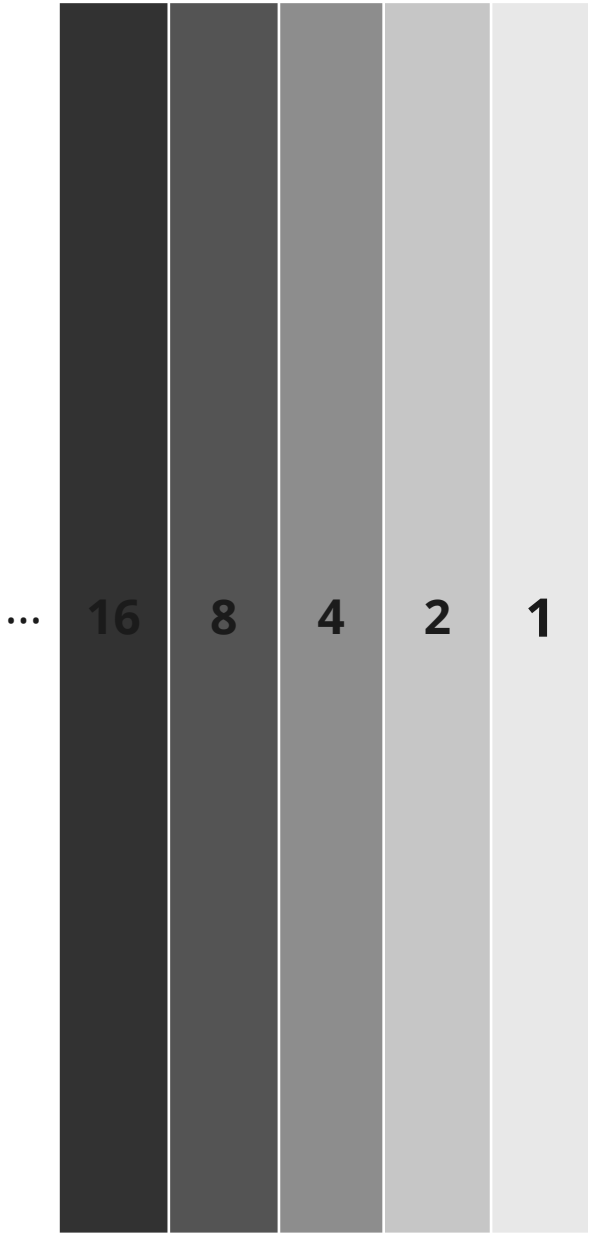


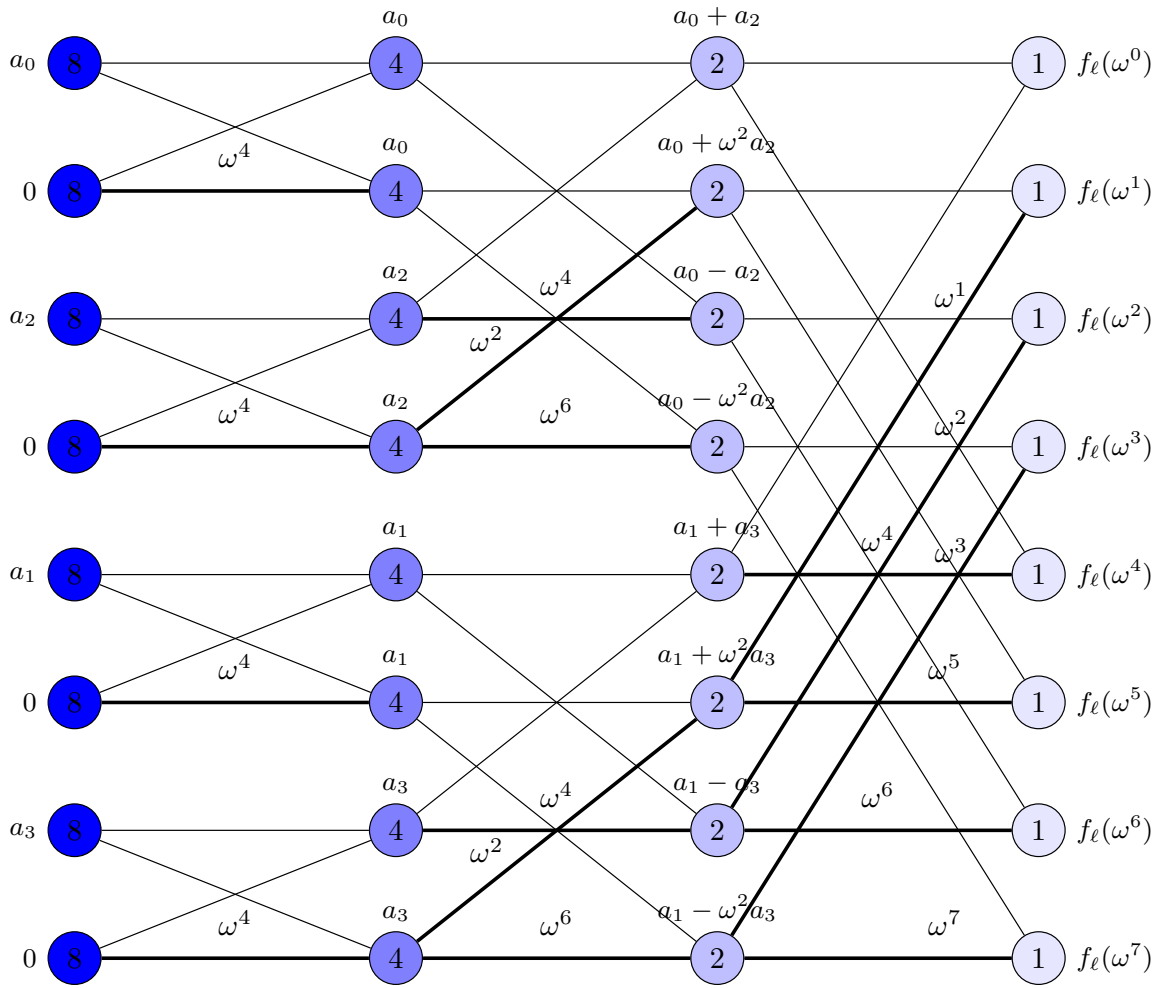


Total weight in nodes: $\Theta(n \log n)$ ☺

Coalition resistance: $O(\log n)$ ☹

EXAMPLE: HIGH COALITION RESISTANCE

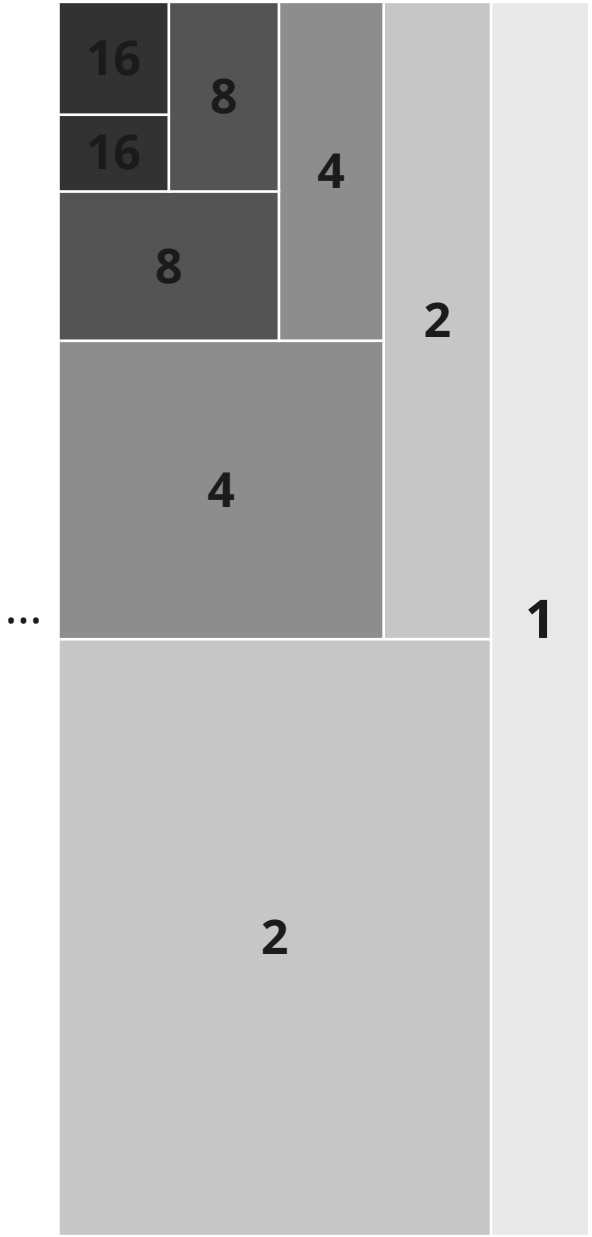


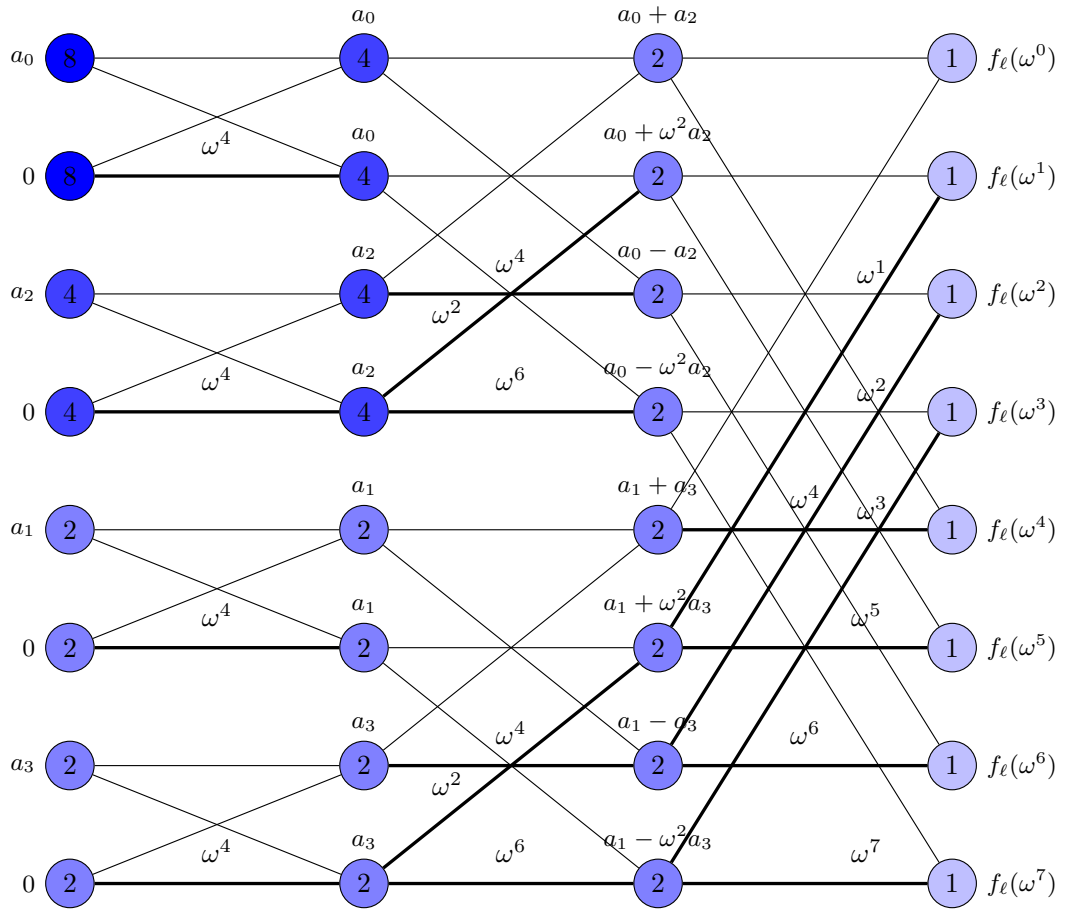


Total weight in nodes: $\Theta(n^2)$ ☹

Coalition resistance: $\Theta(n)$ ☺

EXAMPLE: BOX WEIGHTING SCHEME





$$n \log_2 n + n + \sum_{k=0}^{\log_2 n - 1} 2^k \cdot \frac{n}{2^k} \cdot (\log_2 n - k)$$

$$= n \log_2 n + n + n \log_2^2 n - n \sum_{k=0}^{\log_2 n - 1} k$$

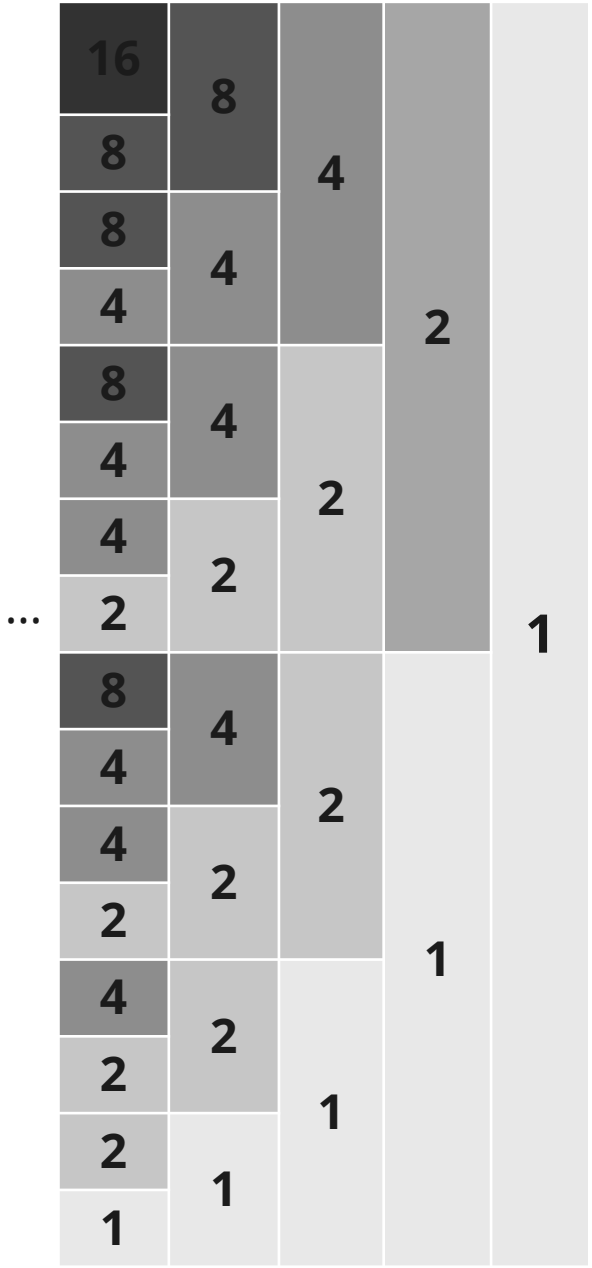
$$= \frac{3}{2} n \log_2 n + n + \frac{1}{2} n \log_2^2 n$$

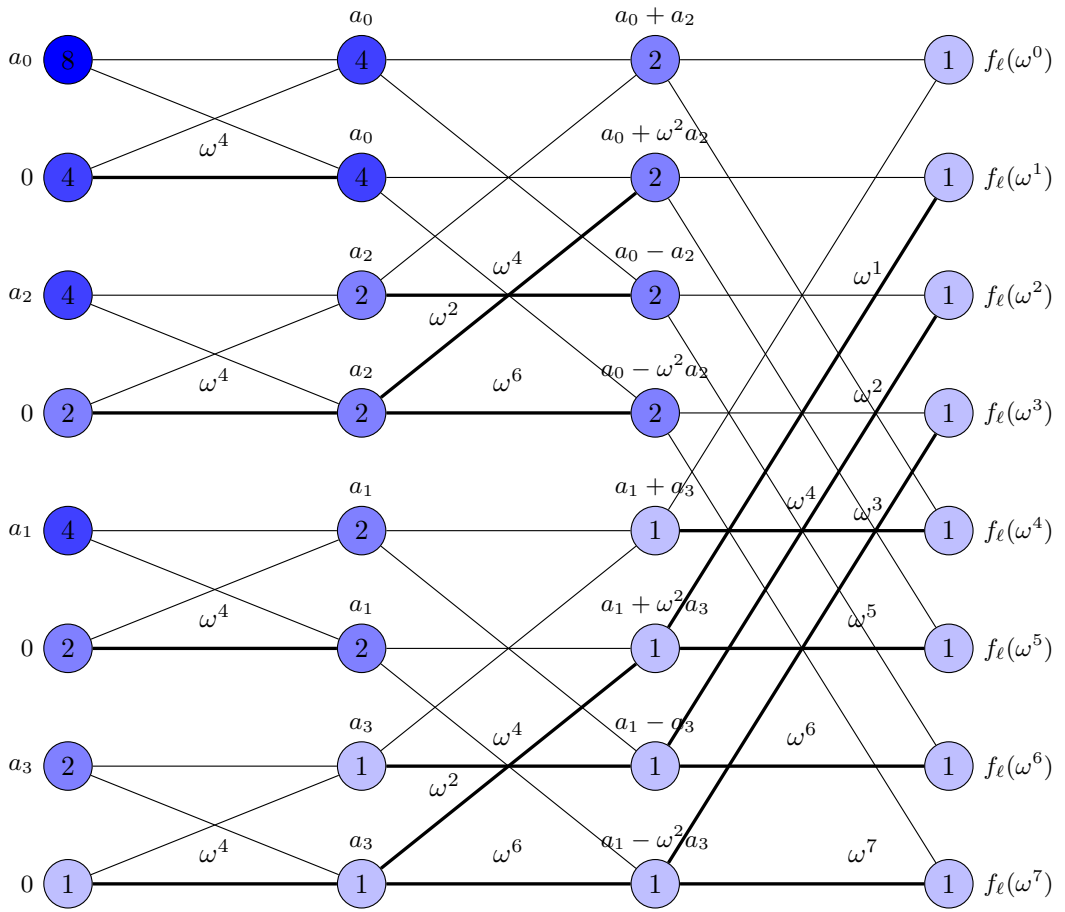
$$= \Theta(n \log^2 n)$$

Total weight in nodes: $\Theta(n \log^2 n)$

Coalition resistance: ?

EXAMPLE: TOP LEFT WEIGHTING SCHEME





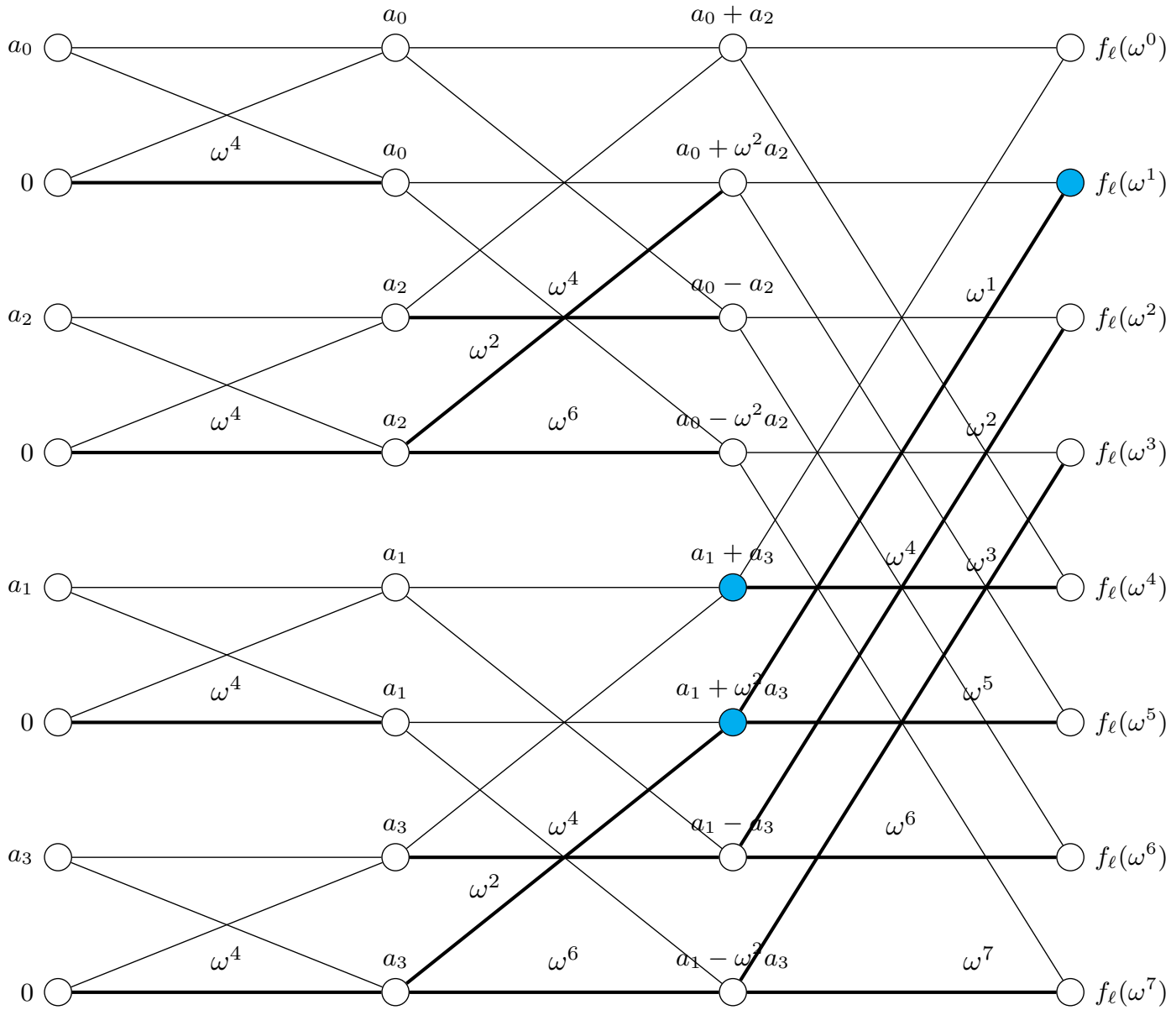
$$n \sum_{k=0}^{\log_2 n} \left(\frac{3}{2}\right)^k$$
$$= n \cdot \left(\frac{1-(3/2)^{\log_2 n+1}}{1-(3/2)}\right)$$
$$= 3^{\log_2 n+1} - 2n$$
$$= \Theta(n^{1/\log_3 2})$$

< 1.59

Total weight in nodes: $\Theta(n^{1/\log_3 2})$

Coalition resistance: ?

AN APPROACH TO FINDING COALITION RESISTANCE



$$M = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ 1 & \omega & \omega^2 & \omega^3 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & \omega^2 \end{pmatrix}$$

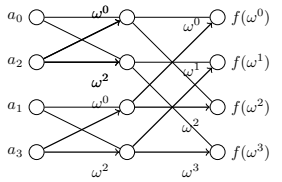
equations at nodes held by coalition

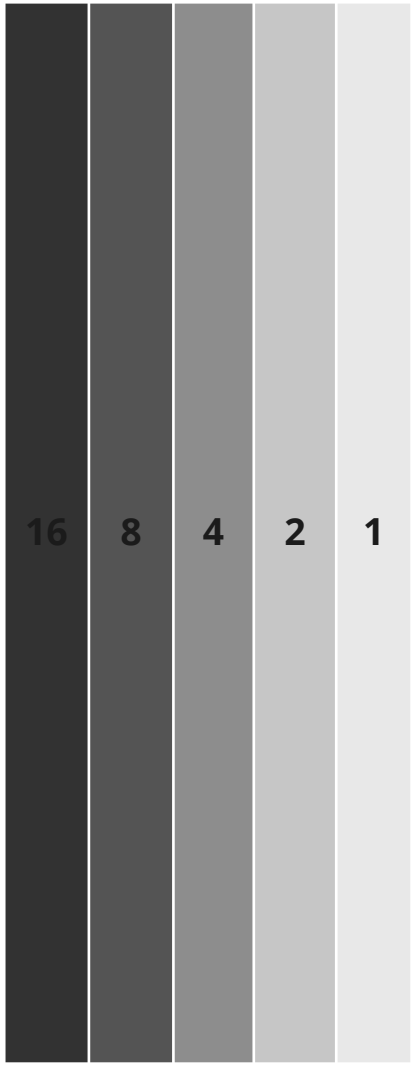
Want to show:

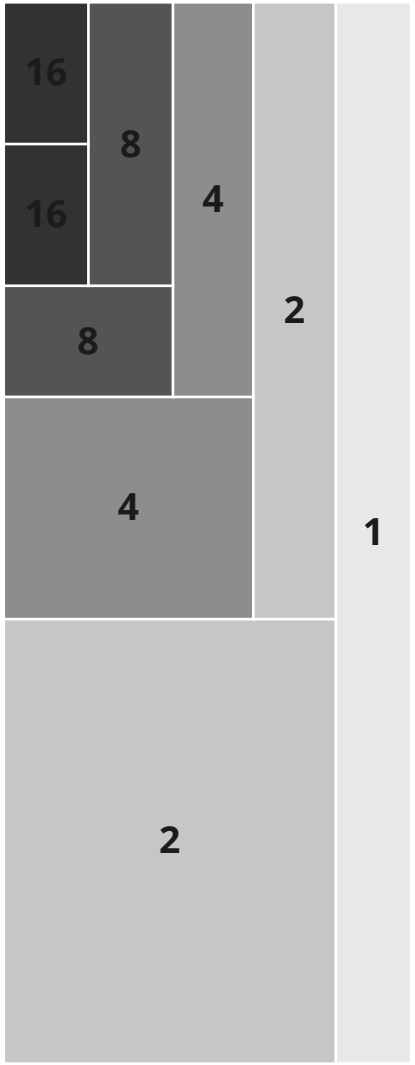
a_0 is not in the row space of M for low-cost sets

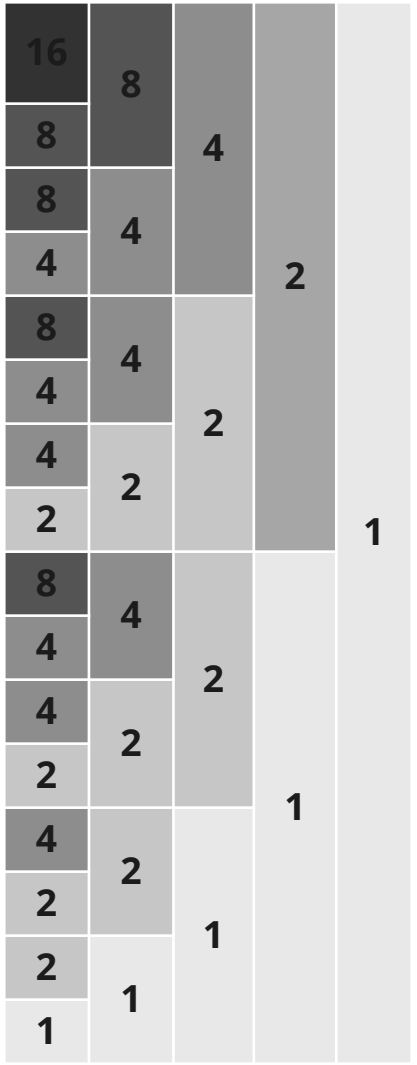
Note: $x \times x$ Submatrices of Vandermonde matrix are full rank

AN APPROACH TO FINDING COALITION RESISTANCE: EXAMPLES









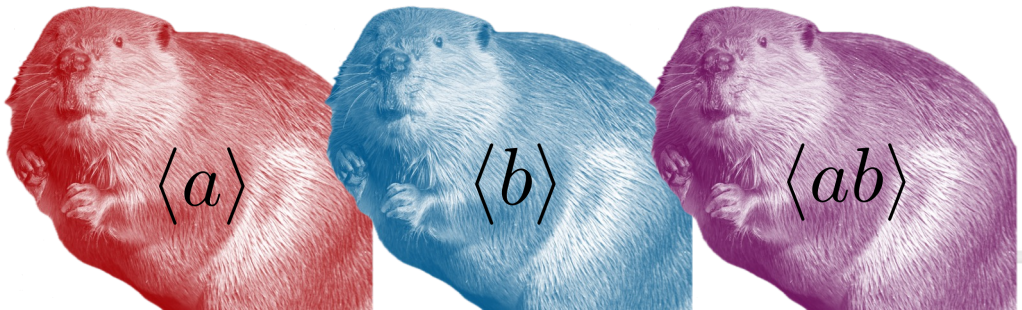
SUMMARY

- Given a Beaver triple, multiplication of additively-shared values is inexpensive

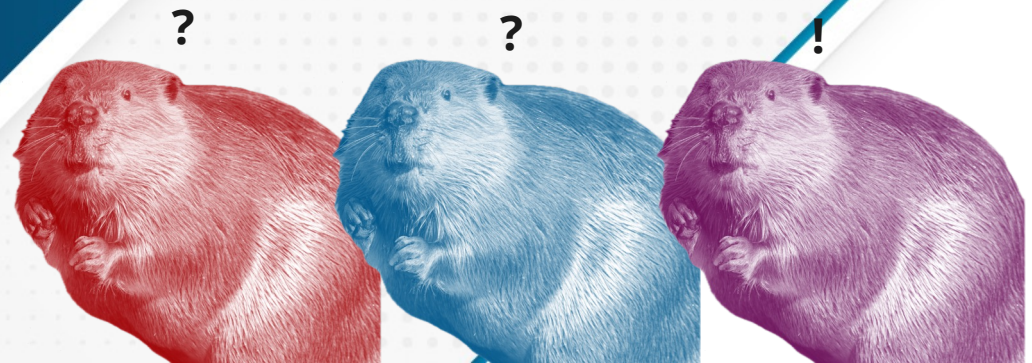
$$z_i = \beta x_i + \alpha b_i + c_i$$

- We introduce an FFT-based method to allow for on-the-fly generation of Beaver triples

- Work in progress: node weights to tolerate coalitions
  - What balance between total weight & coalition resistance can we expect?



QUESTIONS OR COMMENTS?



REFERENCES

- **[Beaver]** D. Beaver. “Efficient Multiparty Protocols Using Circuit Randomization.” *J. Advances in Cryptology – CRYPTO `91*. Lecture Notes in Computer Science, vol 576. (1991).
- **[SPDZ]** I. Damgard, et. al. ”Multiparty Computation from Somewhat Homomorphic Encryption.” *Advances in Cryptology – CRYPTO 2012*. Lecture Notes in Computer Science, vol 7417. (2012).
- **[CrypTen]** B. Knott, S. Venkataraman, and A. Hannun. ”CrypTen: Secure Multi-Party Computation Meets Machine Learning.” *35th Conference on Neural Information Processing Systems (NeurIPS 2021).* (2021).
- **[TinyOT]** J. Nielsen, et. al. ”A New Approach to Practical Active-Secure Two-Party Computation.” *Advances in Cryptology – CRYPTO 2012*. Lecture Notes in Computer Science, vol 7417. (2012).
- **[TaaS]** N. Smart and T. Tanguy. “TaaS: Commodity MPC via Triples-as-a-Service.” *Proc.. Of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop.* (2019)