Cryptography with Weights:
MPC, Encryption and Signatures

Sanjam Garg

UC Berkeley & NTT Research

Abhishek Jain

JHU & NTT Research

Pratyay Mukherjee

Supra Research

Rohit Sinha

Meta –> Swirlds Labs

Mingyuan Wang

UC Berkeley

Yinuo Zhang

UC Berkeley

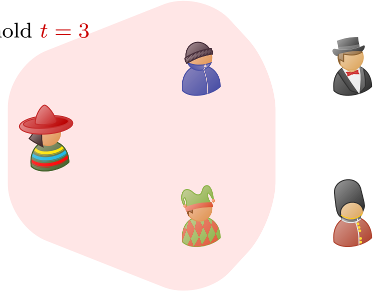
Threshold Cryptography and MPC

- Fundamental primitives that have been extensively studied
- Trust Assumption: All parties are treated equally (i.e., unweighted)

Threshold Cryptography and MPC

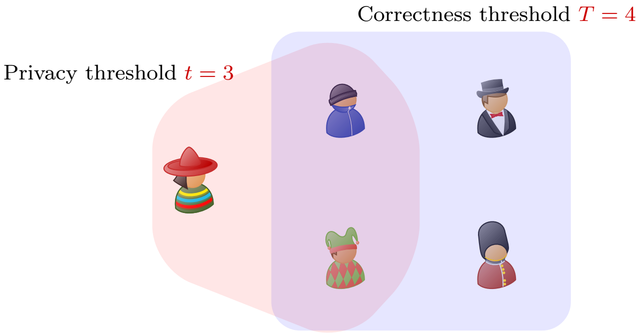
- Fundamental primitives that have been extensively studied
- Trust Assumption: All parties are treated equally (i.e., unweighted)
  - Privacy threshold t: secure if $\leqslant t$ malicious parties.

Privacy threshold $t = 3$



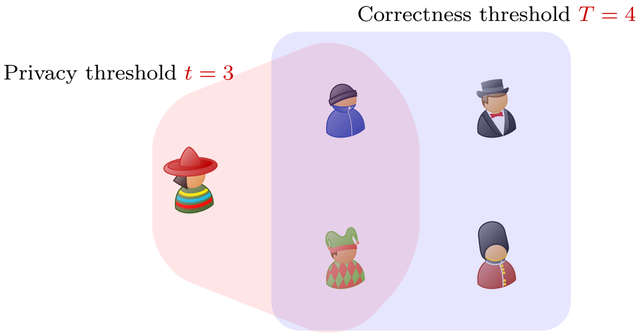
Threshold Cryptography and MPC

- Fundamental primitives that have been extensively studied
- Trust Assumption: All parties are treated equally (i.e., unweighted)
  - Privacy threshold t: secure if $\leqslant t$ malicious parties.
  - Reconstruction threshold T: correct if $\geqslant T$ honest parties.



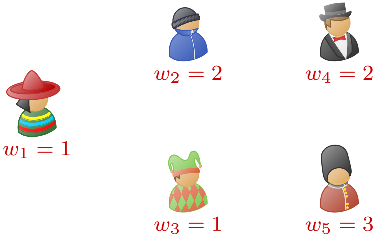
Threshold Cryptography and MPC

- Fundamental primitives that have been extensively studied
- Trust Assumption: All parties are treated equally (i.e., unweighted)
  - Privacy threshold t: secure if $\leqslant t$ malicious parties.
  - Reconstruction threshold T: correct if $\geqslant T$ honest parties.
  - Sharp threshold: $T = t + 1$; Ramp setting: $T > t + 1$



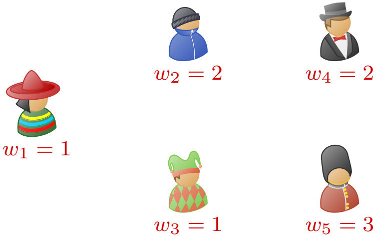
One-party-one-vote may not suffice

- Parties are naturally asymmetric in some emerging applications
  - threshold signature in a stake-based blockchain setting



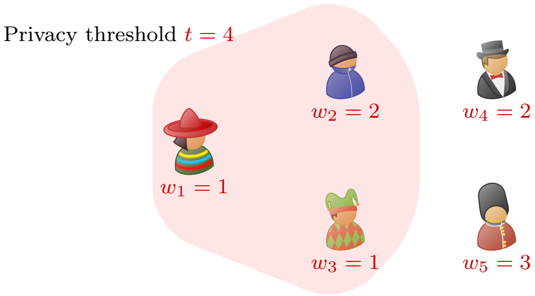
One-party-one-vote may not suffice

- Parties are naturally asymmetric in some emerging applications
  - threshold signature in a stake-based blockchain setting
- Weighted setting:
  - Party are assigned with weights $w_1, w_2, \ldots, w_n \in \mathbb{N}$.



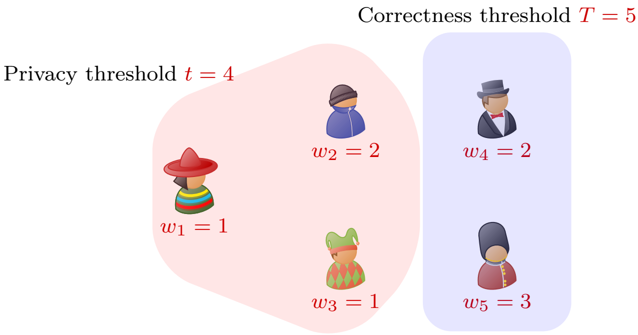
One-party-one-vote may not suffice

- Parties are naturally asymmetric in some emerging applications
  - threshold signature in a stake-based blockchain setting
- Weighted setting:
  - Party are assigned with weights $w_1, w_2, \ldots, w_n \in \mathbb{N}$.
  - Security holds if corrupted parties have cumulative weights $\leqslant t$.



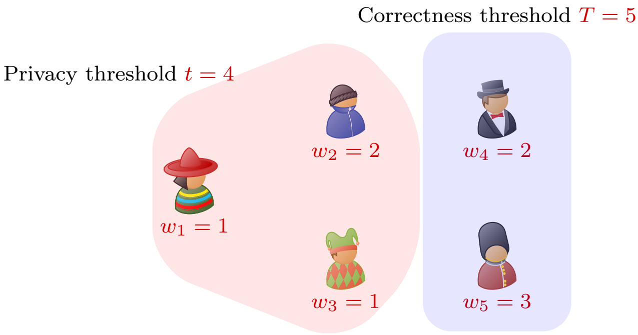
One-party-one-vote may not suffice

- Parties are naturally asymmetric in some emerging applications
  - threshold signature in a stake-based blockchain setting
- Weighted setting:
  - Party are assigned with weights $w_1, w_2, \ldots, w_n \in \mathbb{N}$.
  - Security holds if corrupted parties have cumulative weights $\leqslant t$.
  - Correctness holds if honest parties have cumulative weights $\geqslant T$ participate.



One-party-one-vote may not suffice

- Parties are naturally asymmetric in some emerging applications
  - threshold signature in a stake-based blockchain setting
- Weighted setting:
  - Party are assigned with weights $w_1, w_2, \ldots, w_n \in \mathbb{N}$.
  - Security holds if corrupted parties have cumulative weights $\leqslant t$.
  - Correctness holds if honest parties have cumulative weights $\geqslant T$ participate.
  - Motivated by real-world scenarios, small weight regime $w_i = \mathsf{poly}(\lambda)$.



Existing Solutions: Naïve Virtualization

- Party with weight w_i is treated as w_i virtual parties.
- Reduce to unweighted setting among $W = w_1 + w_2 + \cdots + w_n$ virtual parties.
- A multiplicative overhead w_i for each party (computation, communication).

Existing Solutions: Naïve Virtualization

- Party with weight w_i is treated as w_i virtual parties.
- Reduce to unweighted setting among $W = w_1 + w_2 + \cdots + w_n$ virtual parties.
- A multiplicative overhead w_i for each party (computation, communication).

Objective

Can we realize the weighted setting more efficiently?

Can we make this overhead depend additively *on the weights instead of* multiplicatively.

Existing Solutions: Naïve Virtualization

- Party with weight w_i is treated as w_i virtual parties.
- Reduce to unweighted setting among $W = w_1 + w_2 + \cdots + w_n$ virtual parties.
- A multiplicative overhead w_i for each party (computation, communication).

Objective

Can we realize the weighted setting more efficiently?

Can we make this overhead depend additively *on the weights instead of* multiplicatively.

This work: take-home message

The answer is yes if there is a sufficient gap between reconstruction threshold T and privacy threshold t,

$$T - t = \Omega(\lambda).$$

Technical Core

Efficient Weighted Ramp Secret Sharing (WRSS)

Let $(w_1, \ldots, w_n, T, t)$ define a weighted access structure.

$$T - t = \Omega(\lambda).$$

There exists a weighted ramp secret sharing scheme for λ-bit secret such that

- The share size of a party with weight w_i is $O(w_i)$.
  - Comparison to Shamir $w_i \cdot \lambda$ for a λ-bit secret
- Perfectly correct and $\exp(-\lambda)$-statistically secure.
- Build from Chinese Remainder Theorem-based secret sharing [Mignotte'83, Asmuth-Bloom'83]

Technical Core

Efficient Weighted Ramp Secret Sharing (WRSS)

Let $(w_1, \ldots, w_n, T, t)$ define a weighted access structure.

$$T - t = \Omega(\lambda).$$

There exists a weighted ramp secret sharing scheme for λ-bit secret such that

- The share size of a party with weight w_i is $O(w_i)$.
  - Comparison to Shamir $w_i \cdot \lambda$ for a λ-bit secret
- Perfectly correct and $\exp(-\lambda)$-statistically secure.
- Build from Chinese Remainder Theorem-based secret sharing [Mignotte'83, Asmuth-Bloom'83]

Applications

- Applicable to MPC, threshold encryption, and threshold signature.
- The application inherits the efficiency gain of the secret-sharing schemes.
- WRSS is *non-linear*, which presents some technical challenges

Prior Works

[Beimel-Weinreb'05, Beimel-Tassa-Weinreb'05]

- Computational setting (OWF), Sharp threshold
- poly(n) share size, independent of the weights w_i
- Garbling techniques for circuits realizing weighted threshold gate

Prior Works

[Beimel-Weinreb'05, Beimel-Tassa-Weinreb'05]

- Computational setting (OWF), Sharp threshold
- $\mathsf{poly}(n)$ share size, independent of the weights w_i
- Garbling techniques for circuits realizing weighted threshold gate

Concurrent Work

[Benhamouda-Halevi-Stambler ITC'22]

- Information-theoretic and ramp setting, where $T = \beta \cdot W,\ t = \alpha \cdot W$ with constants $\beta > \alpha$.
- share size $\mathsf{poly}(\alpha, \beta, \lambda)$, independent of the weights w_i
- relies on beautiful connections to wiretap channels

Prior Works

[Beimel-Weinreb'05, Beimel-Tassa-Weinreb'05]

- Computational setting (OWF), Sharp threshold
- $\mathsf{poly}(n)$ share size, independent of the weights w_i
- Garbling techniques for circuits realizing weighted threshold gate

Concurrent Work

[Benhamouda-Halevi-Stambler ITC'22]

- Information-theoretic and ramp setting, where $T = \beta \cdot W,\ t = \alpha \cdot W$ with constants $\beta > \alpha$.
- share size $\mathsf{poly}(\alpha, \beta, \lambda)$, independent of the weights w_i
- relies on beautiful connections to wiretap channels

Compare to Our Work

- Our scheme still depends on the weights w_i, trade-off depends on the weights
- Our scheme preserves the *algebraic structure* of the secrets, render it applicable to threshold crypto and MPC

CRT-based Secret Sharing [Mignotte'83, Asmuth-Bloom'83]

- Suppose secret $s \in \mathbb{F}$, where $|\mathbb{F}| = p_0 \approx 2^\lambda$.
- Parties are associated with integers $p_1, \ldots, p_n$.
- p_0 and $p_1, p_2, \ldots, p_n$ are coprime.

CRT-based Secret Sharing [Mignotte'83, Asmuth-Bloom'83]

- Suppose secret $s \in \mathbb{F}$, where $|\mathbb{F}| = p_0 \approx 2^\lambda$.
- Parties are associated with integers $p_1, \ldots, p_n$.
- p_0 and $p_1, p_2, \ldots, p_n$ are coprime.

To share a secret

- Rerandomize s as a "random" integer S, where

$$S \equiv s \mod p_0$$

- i^{th} secret share is defined as $s_i = S \mod p_i$.

CRT-based Secret Sharing [Mignotte'83, Asmuth-Bloom'83]

- Suppose secret $s \in \mathbb{F}$, where $|\mathbb{F}| = p_0 \approx 2^\lambda$.
- Parties are associated with integers $p_1, \ldots, p_n$.
- p_0 and $p_1, p_2, \ldots, p_n$ are coprime.

To share a secret

- Rerandomize s as a "random" integer S, where

$$S \equiv s \mod p_0$$

- i^{th} secret share is defined as $s_i = S \mod p_i$.

To reconstruct a secret

- Given the secret shares $\{s_i\}_{i \in A}$ from an authorized set A
- Invoke Chinese remaindering theorem to find the integer S such that

$$\forall i \in A, \qquad S \mod p_i = s_i.$$

- Reconstruct the secret s as $s = S \mod p_0$.

Why is a good candidate for weighted secret sharing

- Party i receives $\log(p_i)$-bit information!

Why is a good candidate for weighted secret sharing

- Party i receives $\log(p_i)$-bit information!
- It gives a *fine-grained* way to control how much information each party receives.

Why is a good candidate for weighted secret sharing

- Party i receives $\log(p_i)$-bit information!
- It gives a *fine-grained* way to control how much information each party receives.
- A party with a high weight should receive more information!

Why is a good candidate for weighted secret sharing

- Party i receives $\log(p_i)$-bit information!
- It gives a *fine-grained* way to control how much information each party receives.
- A party with a high weight should receive more information!
  - Set $\log(p_i)$ to be proportional to w_i, e.g., $p_i \approx 2^{w_i}$.

Why is a good candidate for weighted secret sharing

- Party i receives $\log(p_i)$-bit information!
- It gives a *fine-grained* way to control how much information each party receives.
- A party with a high weight should receive more information!
  - Set $\log(p_i)$ to be proportional to w_i, e.g., $p_i \approx 2^{w_i}$.
- Authorized set A satisfies $\sum_i w_i > T$. Enough information to construct!

Why is a good candidate for weighted secret sharing

- Party i receives $\log(p_i)$-bit information!
- It gives a *fine-grained* way to control how much information each party receives.
- A party with a high weight should receive more information!
  - Set $\log(p_i)$ to be proportional to w_i, e.g., $p_i \approx 2^{w_i}$.
- Authorized set A satisfies $\sum_i w_i > T$. Enough information to construct!
- Unauthorized set B satisfies $\sum_i w_i < t$. Small enough such that no information of the secret is leaked.

Weighted MPC

- information-theoretic, Honest majority

$$t < \frac{\sum_{i=1}^{n} w_i}{2} = \frac{W}{2}$$

Weighted MPC

- information-theoretic, Honest majority

$$t < \frac{\sum_{i=1}^{n} w_i}{2} = \frac{W}{2}$$

- Our secret sharing + BGW framework? [Ben-Or-Goldwasser-Wigderson'88]

Weighted MPC

- information-theoretic, Honest majority

$$t < \frac{\sum_{i=1}^{n} w_i}{2} = \frac{W}{2}$$

- Our secret sharing + BGW framework? [Ben-Or-Goldwasser-Wigderson'88]
- Works similarly, but with some *differences*

Weighted MPC

- information-theoretic, Honest majority

$$t < \frac{\sum_{i=1}^{n} w_i}{2} = \frac{W}{2}$$

- Our secret sharing + BGW framework? [Ben-Or-Goldwasser-Wigderson'88]
- Works similarly, but with some *differences*

Weighted MPC

- information-theoretic, Honest majority

$$t < \frac{\sum_{i=1}^{n} w_i}{2} = \frac{W}{2}$$

- Our secret sharing + BGW framework? [Ben-Or-Goldwasser-Wigderson'88]
- Works similarly, but with some *differences*

Local Homomorphism

- Suppose we have secrets x and y shared as integers X and Y such that

$$X \equiv x \mod p_0 \qquad Y \equiv y \mod p_0$$

Weighted MPC

- information-theoretic, Honest majority

$$t < \frac{\sum_{i=1}^{n} w_i}{2} = \frac{W}{2}$$

- Our secret sharing + BGW framework? [Ben-Or-Goldwasser-Wigderson'88]
- Works similarly, but with some *differences*

Local Homomorphism

- Suppose we have secrets x and y shared as integers X and Y such that

$$X \equiv x \mod p_0 \qquad Y \equiv y \mod p_0$$

- Party i gets the secret share $x_i = X \mod p_i$ and $y_i = Y \mod p_i$.

Weighted MPC

- information-theoretic, Honest majority

$$t < \frac{\sum_{i=1}^{n} w_i}{2} = \frac{W}{2}$$

- Our secret sharing + BGW framework? [Ben-Or-Goldwasser-Wigderson'88]
- Works similarly, but with some *differences*

Local Homomorphism

- Suppose we have secrets x and y shared as integers X and Y such that

$$X \equiv x \mod p_0 \qquad Y \equiv y \mod p_0$$

- Party i gets the secret share $x_i = X \mod p_i$ and $y_i = Y \mod p_i$.
- The local sum of secret shares $x_i + y_i$ secret shares the integer $X + Y$ (hence, the secret $x + y$).

$$X + Y \equiv x_i + y_i \mod p_i$$

Weighted MPC

- information-theoretic, Honest majority

$$t < \frac{\sum_{i=1}^{n} w_i}{2} = \frac{W}{2}$$

- Our secret sharing + BGW framework? [Ben-Or-Goldwasser-Wigderson'88]
- Works similarly, but with some *differences*

Local Homomorphism

- Suppose we have secrets x and y shared as integers X and Y such that

$$X \equiv x \mod p_0 \qquad Y \equiv y \mod p_0$$

- Party i gets the secret share $x_i = X \mod p_i$ and $y_i = Y \mod p_i$.
- The local sum of secret shares $x_i + y_i$ secret shares the integer $X + Y$ (hence, the secret $x + y$).

$$X + Y \equiv x_i + y_i \mod p_i$$

- The local products of secret shares $x_i \cdot y_i$ secret shares the integer $X \cdot Y$ (hence, the secret $x \cdot y$).

$$X \cdot Y \equiv x_i \cdot y_i \mod p_i$$

Weighted MPC

Integer growing issue

- If the integer becomes too large $S > p_1 \cdot p_2 \cdots p_n \approx 2^W$, one cannot ensure correctness!

Weighted MPC

Integer growing issue

- If the integer becomes too large $S > p_1 \cdot p_2 \cdots p_n \approx 2^W$, one cannot ensure correctness!
- Integer grows *slowly* for $+$. For a polynomial-size circuit, not an issue.

Weighted MPC

Integer growing issue

- If the integer becomes too large $S > p_1 \cdot p_2 \cdots p_n \approx 2^W$, one cannot ensure correctness!
- Integer grows *slowly* for $+$. For a polynomial-size circuit, not an issue.
- Integer grows *quickly* for $\times$. Every multiplication doubles the length of the integer.

Weighted MPC

Integer growing issue

- If the integer becomes too large $S > p_1 \cdot p_2 \cdots p_n \approx 2^W$, one cannot ensure correctness!
- Integer grows *slowly* for $+$. For a polynomial-size circuit, not an issue.
- Integer grows *quickly* for $\times$. Every multiplication doubles the length of the integer.
- "degree-reduction" protocol after each multiplication!

Applications to Threshold Crypto

Given a sharing $[\![s]\!] = (s_1, \ldots, s_n)$, how do parties reconstruct g^s for some group generator g.

Applications to Threshold Crypto

Given a sharing $[\![s]\!] = (s_1, \ldots, s_n)$, how do parties reconstruct g^s for some group generator g.

Challenges with non-linear secret sharing

To reconstruct a secret s,

$$s = \Bigg(\underbrace{(s_1 \cdot \lambda_1 + s_2 \cdot \lambda_2 + \cdots + s_n \cdot \lambda_n) \overbrace{\bmod P}^{\text{non-linear}}}_{S} \Bigg) \bmod p_0.$$

λ_i is the "Lagrange" coefficient, i.e., $\lambda_i \bmod p_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$.

Applications to Threshold Crypto

Given a sharing $[\![s]\!] = (s_1, \ldots, s_n)$, how do parties reconstruct g^s for some group generator g.

Challenges with non-linear secret sharing

To reconstruct a secret s,

$$s = \Big(\underbrace{\left(s_1 \cdot \lambda_1 + s_2 \cdot \lambda_2 + \cdots + s_n \cdot \lambda_n\right) \overbrace{\mod P}^{\text{non-linear}}}_{S}\Big) \mod p_0.$$

λ_i is the "Lagrange" coefficient, i.e., $\lambda_i \mod p_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$.

Suppose parties want to reconstruct g^s by broadcasting g^{s_i}. Note that

$$(g^{s_1})^{\lambda_1} \cdots (g^{s_n})^{\lambda_n} \neq g^s$$

as

$$\left(s_1 \cdot \lambda_1 + s_2 \cdot \lambda_2 + \cdots + s_n \cdot \lambda_n\right) \mod p_0 \neq \Big(\left(s_1 \cdot \lambda_1 + s_2 \cdot \lambda_2 + \cdots + s_n \cdot \lambda_n\right) \mod P\Big) \mod p_0$$

Our Solution

We change the reconstruction to be

$$s = \left(\Big((s_1 \cdot \lambda_1) \mod P + (s_2 \cdot \lambda_2) \mod P + \cdots + (s_n \cdot \lambda_n) \mod P\Big) \mod P\right) \mod p_0.$$

Our Solution

We change the reconstruction to be

$$s = \left(\left((s_1 \cdot \lambda_1) \mod P + (s_2 \cdot \lambda_2) \mod P + \cdots + (s_n \cdot \lambda_n) \mod P\right) \mod P\right) \mod p_0.$$

Let r_i be $(s_i \cdot \lambda_i) \mod P$. Note that

$$s \equiv r_1 + r_2 + \cdots + r_n - \alpha \cdot P \mod p_0$$

for some $\alpha \in \{0, 1, \ldots, n-1\}$.

Our Solution

We change the reconstruction to be

$$s = \left(\left((s_1 \cdot \lambda_1) \mod P + (s_2 \cdot \lambda_2) \mod P + \cdots + (s_n \cdot \lambda_n) \mod P \right) \mod P \right) \mod p_0.$$

Let r_i be $(s_i \cdot \lambda_i) \mod P$. Note that

$$s \equiv r_1 + r_2 + \cdots + r_n - \alpha \cdot P \mod p_0$$

for some $\alpha \in \{0, 1, \ldots, n-1\}$.

- Suppose parties broadcast g^{r_i}.
- Now, parties know

  $$g^s = g^{r_1} \cdots g^{r_n} \cdot g^{-\alpha \cdot P}$$

  for some $\alpha \in \{0, 1, \ldots, n-1\}$.

Our Solution

We change the reconstruction to be

$$s = \Big(\big((s_1 \cdot \lambda_1) \mod P + (s_2 \cdot \lambda_2) \mod P + \cdots + (s_n \cdot \lambda_n) \mod P\big) \mod P\Big) \mod p_0.$$

Let r_i be $(s_i \cdot \lambda_i) \mod P$. Note that

$$s \equiv r_1 + r_2 + \cdots + r_n - \alpha \cdot P \qquad \mod p_0$$

for some $\alpha \in \{0, 1, \ldots, n-1\}$.

- Suppose parties broadcast g^{r_i}.
- Now, parties know

  $$g^s = g^{r_1} \cdots g^{r_n} \cdot g^{-\alpha \cdot P}$$

  for some $\alpha \in \{0, 1, \ldots, n-1\}$.

Weighted Threshold Encryption/Signature

- Threshold ElGamal: The encryptor will send additional information to help parties recover α.
- We also constructed weighted threshold ECDSA. Refer to the paper for details.

Follow-up Works

- Weighted (sharp-)Threshold Signature
  - [Garg-Jain-Mukherjee-Sinha-Wang-Zhang S&P'24] **ia.cr/2023/567**
  - [Das-Camacho-Xiang-Nieto-Bunz-Ren CCS'23] **ia.cr/2023/598**
  - Efficiency *fully* independent of the weights
  - building on ideas from SNARK literature

Follow-up Works

- Weighted (sharp-)Threshold Signature
  - [Garg-Jain-Mukherjee-Sinha-Wang-Zhang S&P'24] **ia.cr/2023/567**
  - [Das-Camacho-Xiang-Nieto-Bunz-Ren CCS'23] **ia.cr/2023/598**
  - Efficiency *fully* independent of the weights
  - building on ideas from SNARK literature
- Weighted (sharp-)Threshold Encryption
  - Ongoing work: [Garg-Kolonelos-Policharla-Wang 2023]
  - Efficiency *partially* independent of the weights
  - A more efficient computational weighted secret sharing (from pairing)

Follow-up Works

- Weighted (sharp-)Threshold Signature
  - [Garg-Jain-Mukherjee-Sinha-Wang-Zhang S&P'24] **ia.cr/2023/567**
  - [Das-Camacho-Xiang-Nieto-Bunz-Ren CCS'23] **ia.cr/2023/598**
  - Efficiency *fully* independent of the weights
  - building on ideas from SNARK literature
- Weighted (sharp-)Threshold Encryption
  - Ongoing work: [Garg-Kolonelos-Policharla-Wang 2023]
  - Efficiency *partially* independent of the weights
  - A more efficient computational weighted secret sharing (from pairing)

Thanks! Questions?