Publicly Verifiable Secret Sharing over Class Groups and Applications to DKG and YOSO

Ignacio Cascudo — IMDEA Software Institute, Madrid

Bernardo David — IT University of Copenhagen

EUROCRYPT 24

Zürich, 29 May 2024

Publicly Verifiable Secret Sharing (PVSS)

Publicly Verifiable Secret Sharing (PVSS)

Secret sharing with publicly verifiable proofs of:

Publicly Verifiable Secret Sharing (PVSS)

Secret sharing with publicly verifiable proofs of:

- Sharing correctness (by the *dealer*).

    For Shamir Secret Sharing, “The shares are evaluations of a polynomial of degree $\leq t$”

Publicly Verifiable Secret Sharing (PVSS)

Secret sharing with publicly verifiable proofs of:

- Sharing correctness (by the *dealer*).

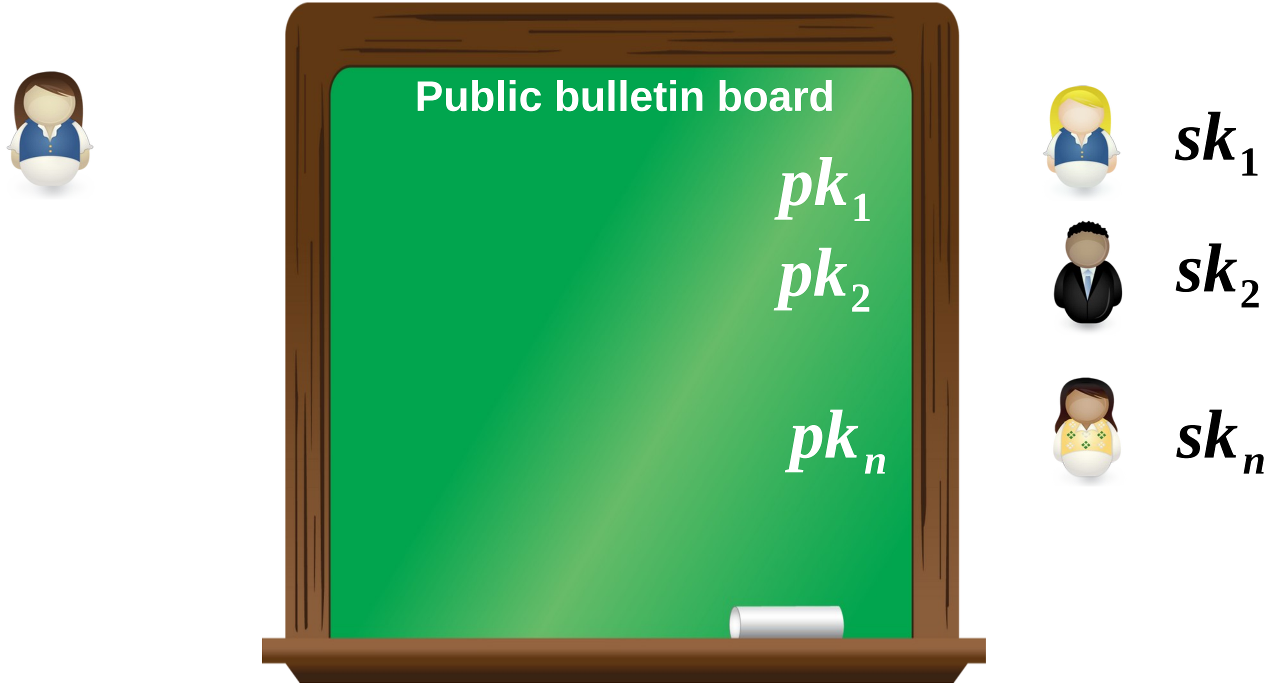
  For Shamir Secret Sharing, “The shares are evaluations of a polynomial of degree ≤t”

- Correct reconstruction of secret (by *reconstructing parties*).

Publicly Verifiable Secret Sharing (PVSS)

Publicly Verifiable Secret Sharing (PVSS)

- Dealer delivers shares via PKE on a public bulletin board.

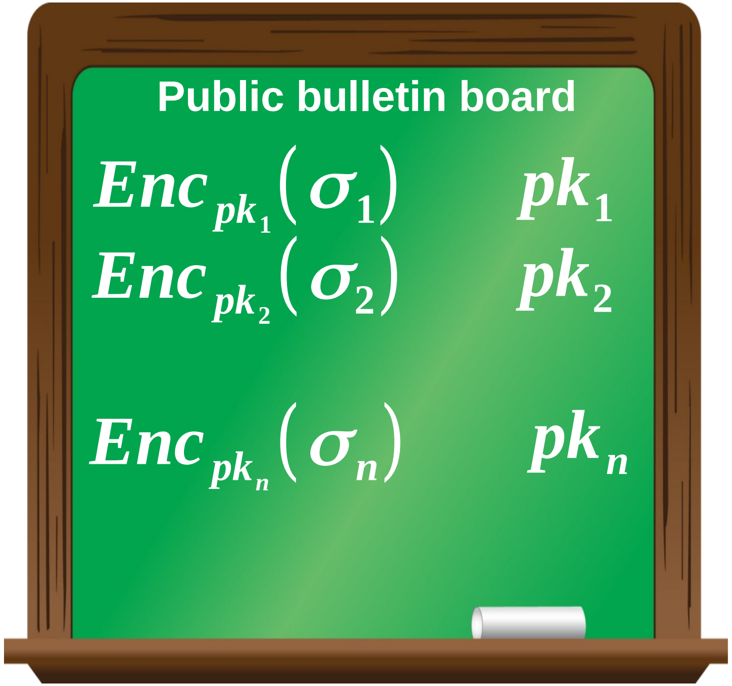


Publicly Verifiable Secret Sharing (PVSS)

- Dealer delivers shares via PKE on a public bulletin board.

Secret: **s**

Shamir shares: **(σ_1,…,σ_n)**



sk_1

sk_2

sk_n

Publicly Verifiable Secret Sharing (PVSS)

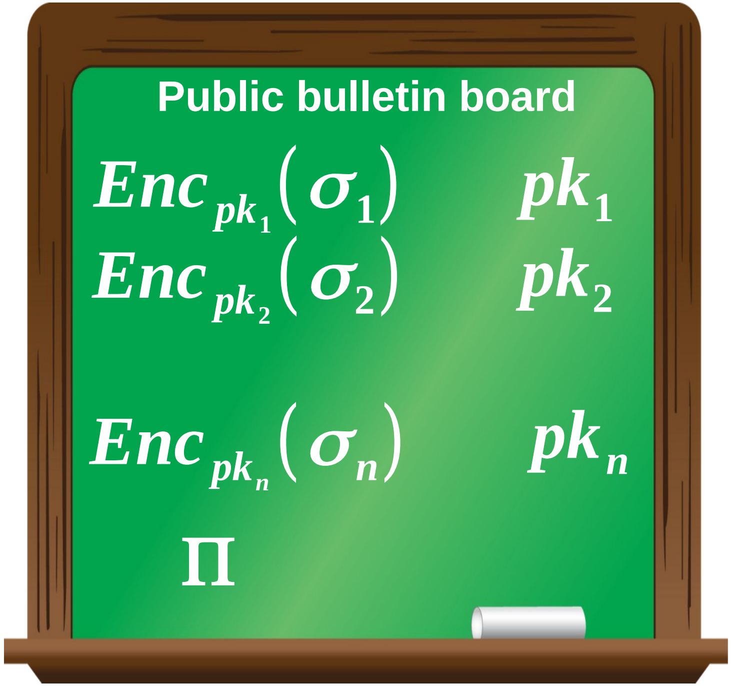
- Dealer delivers shares via PKE on a public bulletin board.
- Dealer publishes NIZK that plaintexts are a correct sharing.

Secret: **s**

Shamir shares: $(\sigma_1,\ldots,\sigma_n)$

Proof :

Π=NIZK($\exists$ f , deg f $\leq$ t,
f(α_i)=σ_i , $\forall$i $\in$ [n])



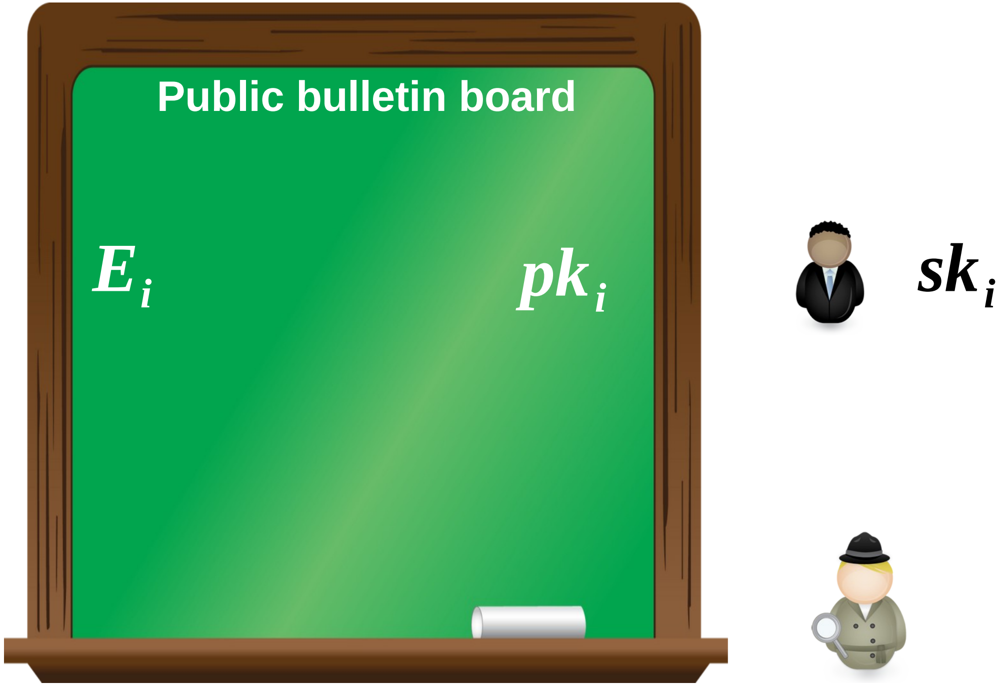
sk_1

sk_2

sk_n

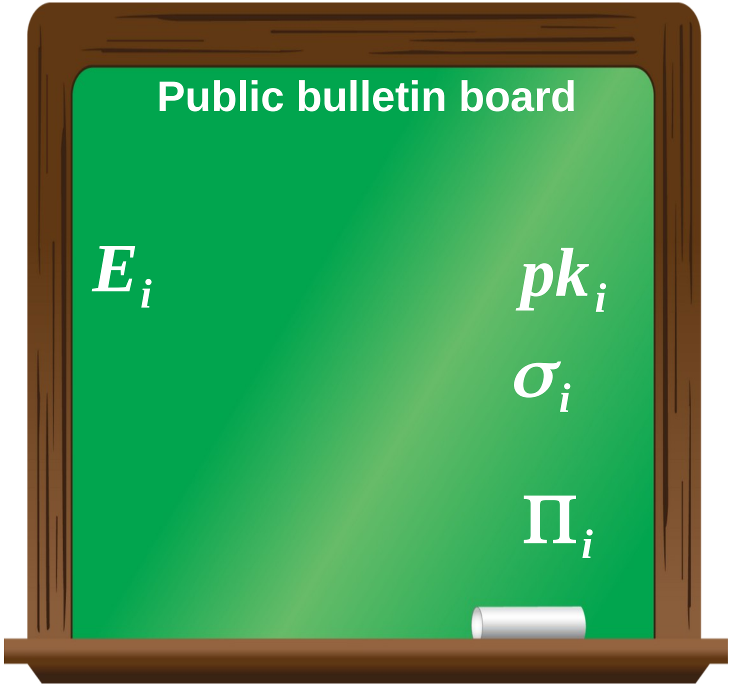
Publicly Verifiable Secret Sharing (PVSS)

- Parties proves correct opening of σ_i, given encryption and pk_i



Publicly Verifiable Secret Sharing (PVSS)

- Parties proves correct opening of σ_i, given encryption and pk_i



sk_i

Proof :

$\Pi_i = \mathrm{NIZK}(\sigma_i = \mathrm{Dec}_{sk}(E_i))$

DL-based PVSS

DL-based PVSS

Several DL-based PVSS exist, e.g.:

DL-based PVSS

Several DL-based PVSS exist, e.g.:
Schoenmakers [Sch99], SCRAPE [CD17], ALBATROSS [CD20], YOLO YOSO [CDGK22]

DL-based PVSS

Several DL-based PVSS exist, e.g.:
Schoenmakers [Sch99], SCRAPE [CD17], ALBATROSS [CD20], YOLO YOSO [CDGK22]

Common features:

- Cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q with hard DL
- Parties get (after decryption) only g^{σ_i} (***not the Shamir shares*** σ_i)
- Hence secret (they can reconstruct) is actually g^s

DL-based PVSS

Several DL-based PVSS exist, e.g.:
Schoenmakers [Sch99], SCRAPE [CD17], ALBATROSS [CD20], YOLO YOSO [CDGK22]

Common features:

- Cyclic group $\mathbb{G}$ **= <g>** of prime order **q** with hard DL
- Parties get (after decryption) only g^{σ_i} (***not the Shamir shares*** σ_i)
- Hence secret (they can reconstruct) is actually g^s

Some applications:

- MPC linear functions with **small output**, e.g. elections [Sch99]
- Randomness beacons [SCRAPE, Albatross]
- **Non-linear** PVSS of **r** in $\mathbb{Z}_q$ [YOLO YOSO]:

Dealer PVSSs random g^s and broadcasts $r - H(g^s)$, (for H: $\mathbb{G} \rightarrow \mathbb{Z}_q$ random oracle)

DL-based PVSS

Several DL-based PVSS exist, e.g.:
Schoenmakers [Sch99], SCRAPE [CD17], ALBATROSS [CD20], YOLO YOSO [CDGK22]

Common features:

- Cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q with hard DL
- Parties get (after decryption) only g^{σ_i} (***not the Shamir shares*** σ_i)
- Hence secret (they can reconstruct) is actually g^s

DL-based PVSS

Several DL-based PVSS exist, e.g.:
Schoenmakers [Sch99], SCRAPE [CD17], ALBATROSS [CD20], YOLO YOSO [CDGK22]

Common features:

- Cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q with hard DL
- Parties get (after decryption) only g^{σ_i} (***not the Shamir shares*** σ_i)
- Hence secret (they can reconstruct) is actually g^s

Drawbacks:
Parties do not learn σ_i

- Bad for Distributed Key Generation (DKG).
- Bad for MPC.

DKG (for DL Key Pairs) via PVSS

Goal: Jointly compute **tpk** = $\mathbf{g}^{\mathbf{tsk}}$. Party i obtains Shamir share $\mathbf{tsk_i}$ of **tsk**

DKG (for DL Key Pairs) via PVSS

Goal: Jointly compute **tpk** = $\mathbf{g}^{\mathbf{tsk}}$. Party i obtains Shamir share $\mathbf{tsk_i}$ of **tsk**

Suppose PVSS where parties can recover Shamir shares σ_i of secret **s**.

DKG (for DL Key Pairs) via PVSS

Goal: Jointly compute **tpk** = g^{tsk}. Party i obtains Shamir share tsk_i of **tsk**
Suppose PVSS where parties can recover Shamir shares σ_i of secret **s**.

- Each party j PVSS $r^{(j)}$, shares $(\sigma^{(j)})_i$

DKG (for DL Key Pairs) via PVSS

Goal: Jointly compute **tpk** = g^{tsk}. Party i obtains Shamir share tsk_i of **tsk**

Suppose PVSS where parties can recover Shamir shares σ_i of secret **s**.

- Each party j PVSS $r^{(j)}$, shares $(\sigma^{(j)})_i$

DKG (for DL Key Pairs) via PVSS

Goal: Jointly compute **tpk** = $\mathbf{g}^{\mathbf{tsk}}$. Party i obtains Shamir share $\mathbf{tsk_i}$ of **tsk**

Suppose PVSS where parties can recover Shamir shares $\boldsymbol{\sigma_i}$ of secret **s**.

- Each party j PVSS $\mathbf{r^{(j)}}$, shares $(\boldsymbol{\sigma^{(j)}})_i$
- Parties determine **set Q of *correctly shared*** $\mathbf{r^{(j)}}$
- Aggregate correct shares:

DKG (for DL Key Pairs) via PVSS

Goal: Jointly compute **tpk** = $\mathbf{g}^{tsk}$. Party i obtains Shamir share $\mathbf{tsk_i}$ of **tsk**
Suppose PVSS where parties can recover Shamir shares $\boldsymbol{\sigma_i}$ of secret **s**.

- Each party j PVSS $\mathbf{r^{(j)}}$, shares $(\boldsymbol{\sigma}^{(j)})_i$
- Parties determine **set Q of *correctly shared*** $\mathbf{r^{(j)}}$
- Aggregate correct shares:
  - Party i defines $\mathbf{tsk_i} = \Sigma_{j\in Q}(\boldsymbol{\sigma}^{(j)})_i$
  - Party i publishes $\mathbf{tpk_i} = \mathbf{g}^{tsk_i}$ and proof of correctness
  - Parties reconstruct **tpk** from correct $\mathbf{tpk_i}$

Contributions

Contributions

- Using **class groups,** we construct a PVSS that allows parties to retrieve the field shares σ_i

Contributions

- Using **class groups,** we construct a PVSS that allows parties to retrieve the field shares σ_i

- Same asymptotical costs as [CDGK22], although over class groups
  - Sharing requires to broadcast n+1 class group elements
  - Sharing proof constant size (3 integers of group size)

Contributions

- Using **class groups,** we construct a PVSS that allows parties to retrieve the field shares σ_i
- Same asymptotical costs as [CDGK22], although over class groups
  - Sharing requires to broadcast n+1 class group elements
  - Sharing proof constant size (3 integers of group size)
- DKG:
  - **2-round DKG** with **unbiasable PK** (round optimal, [Katz23]) with roughly a 4.5-7x gain in communication wrt to Paillier [Katz23].
  - Also, **1-round DKG** with **biasable PK.**

Contributions

- Using **class groups,** we construct a PVSS that allows parties to retrieve the field shares σ_i
- Same asymptotical costs as [CDGK22], although over class groups
  - Sharing requires to broadcast n+1 class group elements
  - Sharing proof constant size (3 integers of group size)
- DKG:
  - **2-round DKG** with **unbiasable PK** (round optimal, [Katz23]) with roughly a 4.5-7x gain in communication wrt to Paillier [Katz23].
  - Also, **1-round DKG** with **biasable PK.**
- Efficient YOSO MPC with transparent setup based on class groups.

Castagnos-Laguillaumie Framework

Castagnos-Laguillaume Framework

Based on class groups.

$\mathbb{G} = G_q \times F$, where:

$F = \langle f \rangle$ of order q, with **easy DL**.

$G_q = \langle g_q \rangle$ cyclic of unknown order.

Castagnos-Laguillaume Framework

Based on class groups.

$\mathbb{G} = G_q \times F$, where:

$F = \langle f \rangle$ of order q, with **easy DL**.

$G_q = \langle g_q \rangle$ cyclic of unknown order.

El Gamal like encryption:

$pk = g_q^{sk}$

$m \to (g_q^{r}, pk^{r} \cdot f^{m})$, with randomness r

Castagnos-Laguillaume Framework

Based on class groups.

$\mathbb{G} = G_q \times F$, where:

$F = \langle f \rangle$ of order q, with **easy DL**.

$G_q = \langle g_q \rangle$ cyclic of unknown order.

El Gamal like encryption:

$pk = g_q^{sk}$

$m \rightarrow (g_q^r, pk^r \cdot f^m)$, with randomness r

Decryptor recovers f^m as in El Gamal, **solves DL in** F, gets m.

PVSS based on Class Groups

PVSS based on Class Groups

- We revisit scheme DHPVSS from YOLO-YOSO [CDGK22] and observe that share encryption can be seen as El-Gamal "multi-encryption":

  Dealer posts common g^r, and $(pk_i)^r \cdot g^{\sigma_i}$ for all i.

PVSS based on Class Groups

- We revisit scheme DHPVSS from YOLO-YOSO [CDGK22] and observe that share encryption can be seen as El-Gamal “multi-encryption”:

  Dealer posts common g^r, and $(pk_i)^r \cdot g^{\sigma_i}$ for all i.

- Natural idea: replace El Gamal by CL:

  Dealer posts common g^r, and encrypted shares $(pk_i)^r \cdot f^{\sigma_i}$

  Now parties can retrieve σ_i !

PVSS based on Class Groups

- We revisit scheme DHPVSS from YOLO-YOSO [CDGK22] and observe that share encryption can be seen as El-Gamal "multi-encryption":

  Dealer posts common g^r, and $(pk_i)^r \cdot g^{\sigma_i}$ for all i.

- Natural idea: replace El Gamal by CL:

  Dealer posts common g^r, and encrypted shares $(pk_i)^r \cdot f^{\sigma_i}$

  Now parties can retrieve σ_i !

- Obstacle: We need to change our proof of sharing.

Sharing Correctness Proof in CDGK22

Sharing Correctness Proof in CDGK22

Sharing proof from [CDGK22] uses “SCRAPE trick” [CD17]:

Linear check $(\sigma_1,\ldots,\sigma_n)$ is a Shamir sharing:

Sample $(w_1,\ldots,w_n)$ uniformly in corresponding **dual code**

Check $w_1\sigma_1 + \ldots + w_n\sigma_n = 0 \bmod q$

Sharing Correctness Proof in CDGK22

Sharing proof from [CDGK22] uses "SCRAPE trick" [CD17]:

Linear check $(\sigma_1,\ldots,\sigma_n)$ is a Shamir sharing:

Sample $(w_1,\ldots,w_n)$ uniformly in corresponding **dual code**

Check $w_1\sigma_1 + \ldots + w_n\sigma_n = 0 \bmod q$

In [CDGK22], dealer publishes $R = g^r$ and $B_i = (pk_i)^r \cdot g^{\sigma_i}$

Sharing proof uses SCRAPE to reduce to DL equality proof:

Sample random $(w_1,\ldots,w_n)$, prove $\Pi\, B_i^{w_i} = \Pi\, (pk_i^{w_i})^r$ for same r s.t. $g^r = R$

Modifications in our Work

Modifications in our Work

In this work, dealer publishes $R = g_q^r$ and $B_i = (pk_i)^r \cdot f^{\sigma_i}$

Modifications in our Work

In this work, dealer publishes $\mathbf{R = g_q^r}$ and $\mathbf{B_i = (pk_i)^r \cdot f^{\sigma_i}}$

Some technical problems arise to use exact same strategies as CDGK22 because:

Modifications in our Work

In this work, dealer publishes $\mathbf{R = g_q^r}$ and $\mathbf{B_i = (pk_i)^r \cdot f^{\sigma_i}}$

Some technical problems arise to use exact same strategies as CDGK22 because:

- **<f>** is of order $\mathbf{q}$, but $\mathbb{G}$ is not. → We need to rerandomize the $\mathbf{w_i}$ to $\mathbf{w_i} + \mathbf{c_i}\,\mathbf{q}$ (for random "small" integers $\mathbf{c_i}$)

Modifications in our Work

In this work, dealer publishes $R = g_q^r$ and $B_i = (pk_i)^r \cdot f^{\sigma_i}$

Some technical problems arise to use exact same strategies as CDGK22 because:

- $\langle f \rangle$ is of order q, but $\mathbb{G}$ is not. → We need to rerandomize the w_i to $w_i + c_i q$ (for random "small" integers c_i)

- DL-EQ PoKs are more expensive.

  Alternatively we show we **can** use **sound** proofs for sharing and reconstruction correctness.

  [BDO23] provides more efficient sound DL-EQ proofs

Comparison with [KMM+23]

Comparison with [KMM+23]

- Kate et al. [KMM+23] presented a PVSS with:
  - **same sharing encryption** as ours,
  - **different sharing correctness proof**
  - they also propose a 1-round DKG

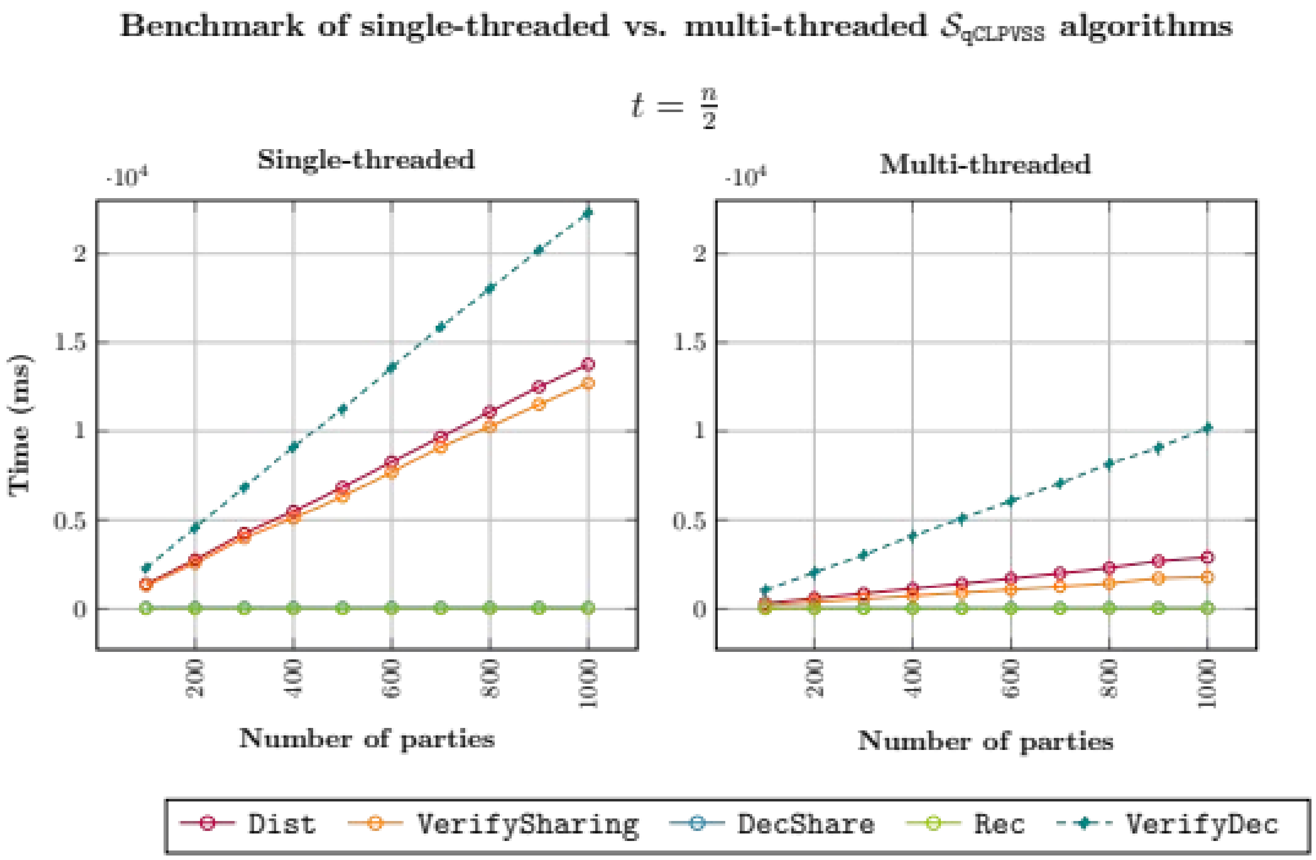
Comparison with [KMM+23]

- Kate et al. [KMM+23] presented a PVSS with:
  - **same sharing encryption** as ours,
  - **different sharing correctness proof**
  - they also propose a 1-round DKG

Comparison:

- Our PVSS sharing is more communication-efficient and we achieve a stronger notion of security (they leak g^s).
- For 1-round DKG: Their scheme is more efficient than ours in communication and computation.

Implementation



Implementation by
Rasmus Føgh Sørensen

Main bottleneck:

Verification of share
opening proofs.

Conclusions

- We present an **efficient PVSS over class groups**, counterpart to CDGK22
- We present **2-round DKG** (unbiasable key) and **1-round DKG**
- We also instantiate **MPC in the YOSO model** based on our PVSS
- Implementation is fast, main bottleneck verification of (many) DLEQ proofs.

Thank you!

https://eprint.iacr.org/2023/1651

Funded by projects:

- SecuRing (grant no. PID2019-110873RJ-I00, MCIN/AEI)
- PRODIGY (grant no. TED2021-132464B-I00, MCIN/AEI and European Union NextGenerationEU/PRTR)
- CONFIDENTIAL-6G (GA 101096435, EU).
- Grant 0165-00079B (Independent Research Fund Denmark)