KU LEUVEN

**48th ISC Webinar**
11 March 2025

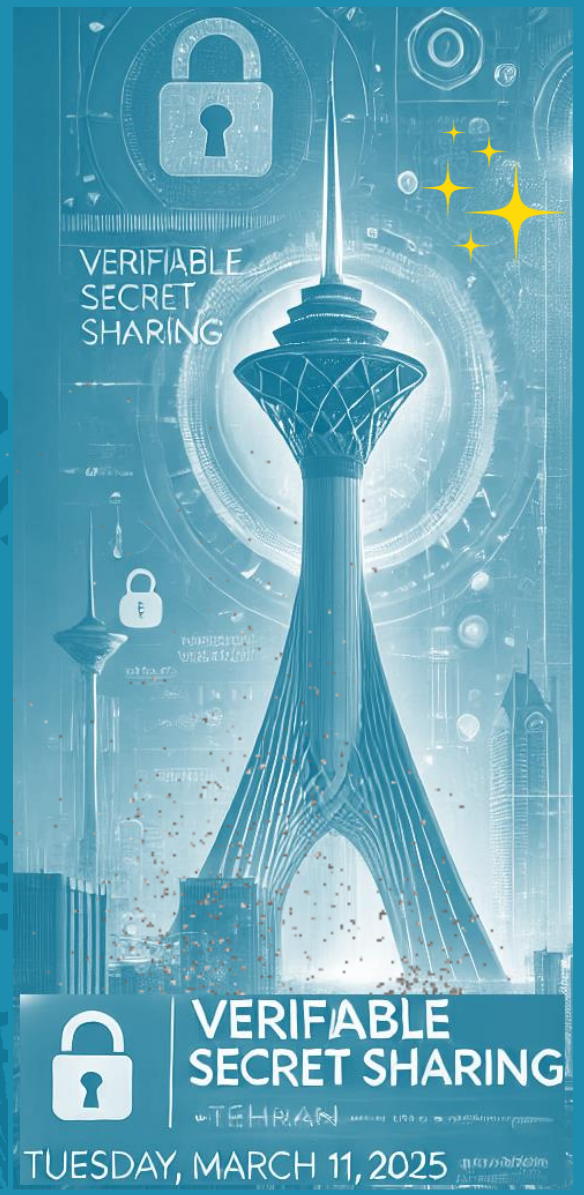
# On Verifiable Secret Sharing


**Karim Baghery**

COSIC

**KU Leuven, Belgium**

**karim.baghery@kuleuven.be**

# Outline:

- Introduction
  - Shamir Secret Sharing and Verifiable Secret Sharing (VSS)
  - Threshold ZK Proofs Over Secret Shared Data
- VSS from Threshold ZK Proofs [ABCP23]
  - The first practical PQ-Secure Computational VSS
  - More Efficient Threshold Protocols Based on DL and Isogenies
- Π: A Unified Framework for Computational VSS [Bag25]
  - $\Pi_F, \Pi_P, \Pi_{LA}$: Alternatives to the Feldman, Pedersen, ABCP23 Schemes
  - Polynomial DL (PDL) Problem and an Efficient PVSS Scheme ($\Pi_S$)
- Traceable Verifiable Secret Sharing and Applications [BEMS25]
  - Traceable Variants of Feldman and Pedersen VSS Schemes
  - An Accoutnable Threshold Protocol Based on ElGamal Encryption



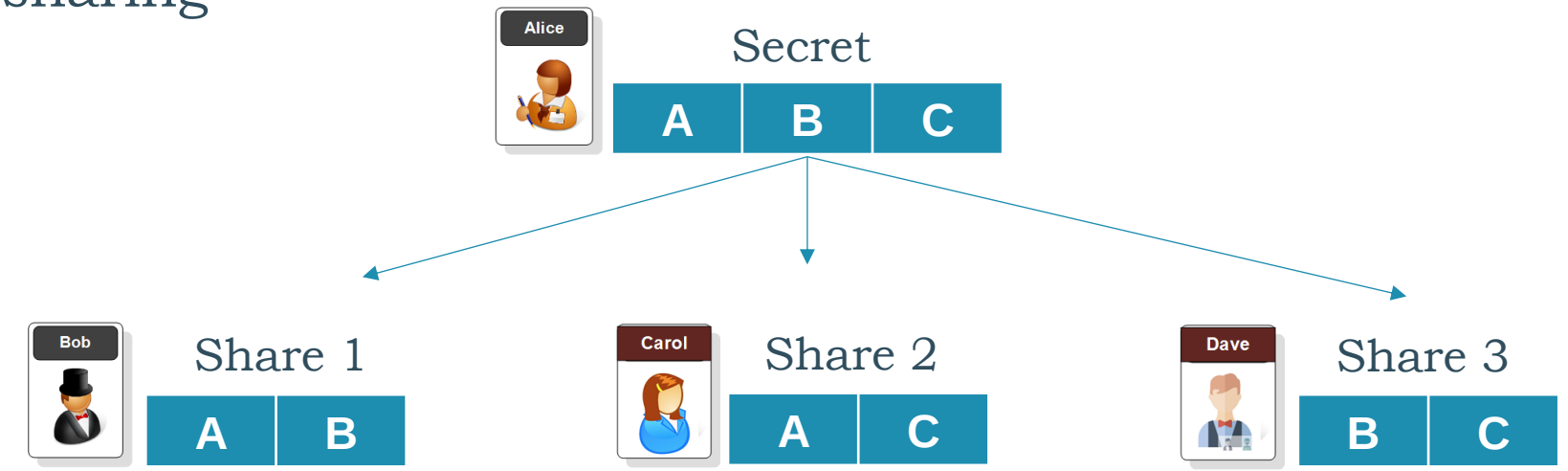
# Secret Sharing: Motivation

- In 2019, two different exchanges in Canada and Korea lost their secret keys and lost $150 million and 2.5 billion Won worth of cryptocurrency forever!

  ✓ **If the exchange's private key had been backed up or recovered, this disaster would have been prevented!**

- Replicated secret sharing

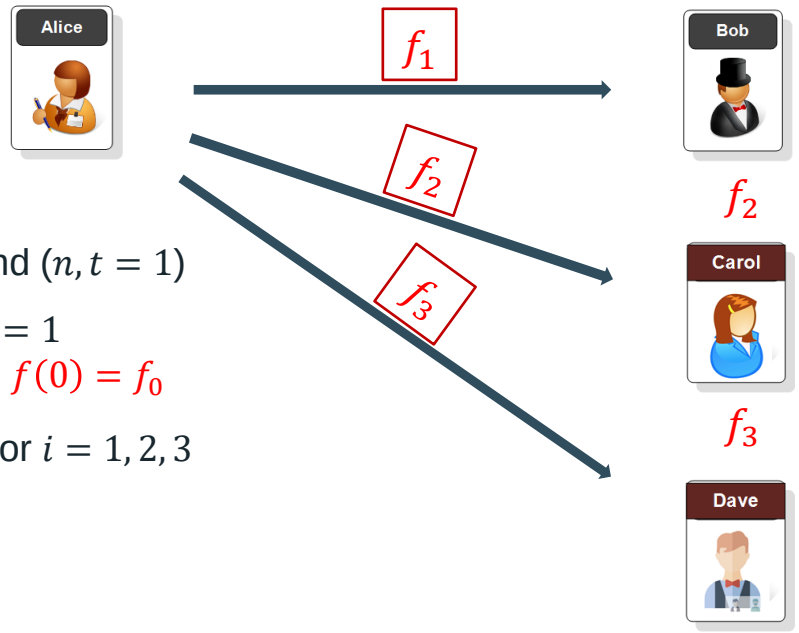
  - Initial
  - Share
  - Reconstruct



# Shamir Secret Sharing: A Toy Example

- Given a secret $f_0$ a **trusted** dealer $P_0$ shares it with $n \geq 2t+1$ parties $\{P_i\}_{i=1}^{n}$, s.t.,
  - $t+1$ shareholders can reconstruct the secret $f_0$ (and **all the other shares**)
  - $t$ shareholders cannot reconstruct the secret $f_0$

Share

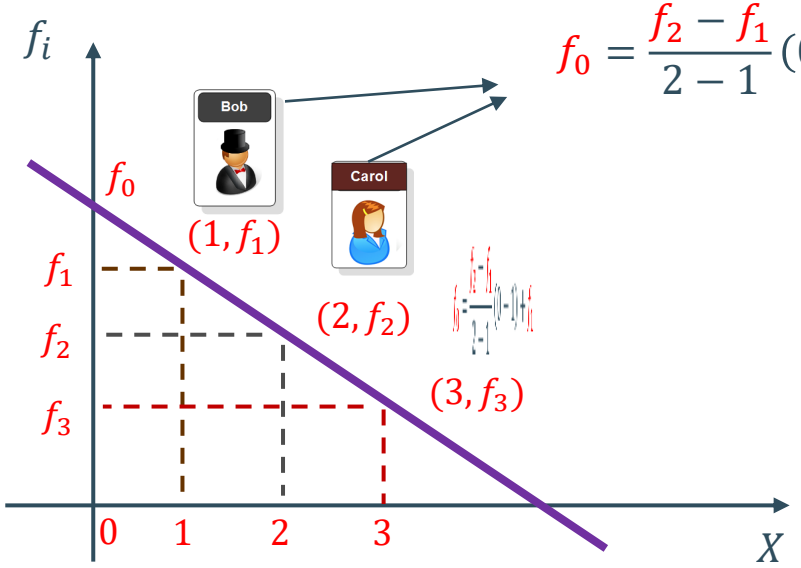



- Given a secret $f_0$ and $(n, t=1)$
- Sample a degree $t=1$ polynomial $f(X)$ s.t. $f(0)=f_0$
- Compute $f_i = f(i)$ for $i = 1, 2, 3$

Reconstruction

$$f_X - f_1 = \frac{f_2 - f_1}{2-1}(X-1)$$

$$f_0 = \frac{f_2 - f_1}{2-1}(0-1) + f_1$$



# VSS from Homomorphic Commitments: Feldman [Fel87]

- Given a secret $f_0$ a ~~**trusted**~~ dealer $P_0$ shares it with $n \geq 2t+1$ parties $\{P_i\}_{i=1}^n$, s.t.,
  - $t+1$ shareholders can reconstruct the secret $f_0$ (and **all the other shares**)
  - $t$ shareholders cannot reconstruct the secret $f_0$

- Initial - Share
- **Verify** - Reconstruct

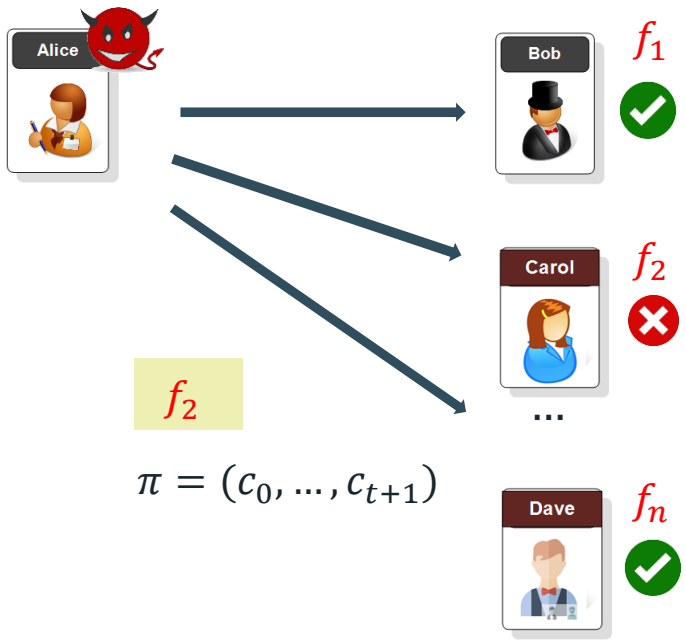
## Share

- Given a secret $f_0$ and $(n, t)$
- Sample a degree $t$ polynomial $f(X) = f_0 + a_1 X + \ldots + a_t X^t$
- Compute $f_i = f(i)$ for $i = 1, \ldots, n$
- Sets $c_0 = g^{f_0},\ c_1 = g^{a_1}, \ldots, c_t = g^{a_t}$
- Publish $\pi = (c_0, \ldots, c_{t+1})$ and send privatly $f_i$ to party $P_i$

Add. Homomorphic Commitments:

$com(a_1) \times com(a_2) = com(a_1 + a_2)$

e.g., $g^{a_1} \times g^{a_2} = g^{a_1 + a_2}$




## Verify

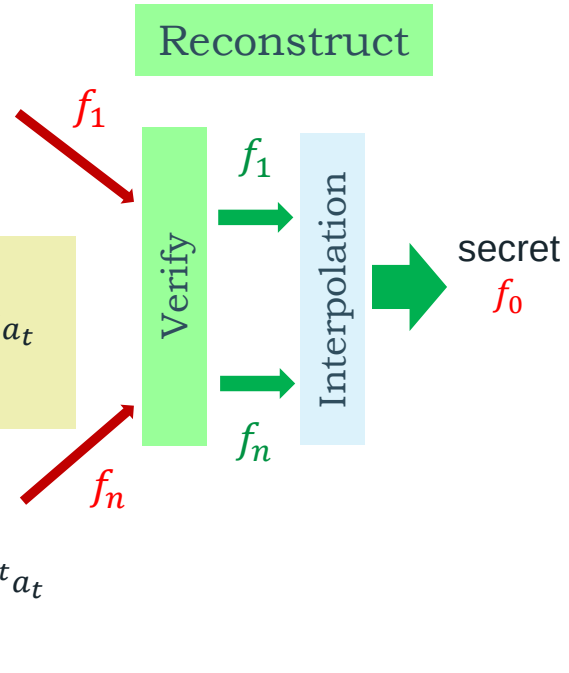
$$g^{f_i} = \prod_{j=0}^{t} c_j^{i^j}$$

$$g^{f_1} = c_0^{1^0} \times c_1^{1^1} \times \cdots \times c_t^{1^t} = g^{f_0} \times g^{a_1} \times \cdots \times g^{a_t} = g^{f(1)}$$

$$g^{f_2} = c_0^{2^0} \times c_1^{2^1} \times \cdots \times c_t^{2^t} = g^{f_0} \times g^{2a_1} \times \cdots \times g^{2^t a_t} = g^{f(2)}$$

$$g^{f_n} = c_0^{n^0} \times c_1^{n^1} \times \cdots \times c_t^{n^t} = g^{f_0} \times g^{na_1} \times \cdots \times g^{n^t a_t} = g^{f(n)}$$

## Reconstruct






[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In 28th Annual Symposium on Foundations of Computer Science, 1987.

# VSS from Homomorphic Commitments: Pedersen [Ped91]

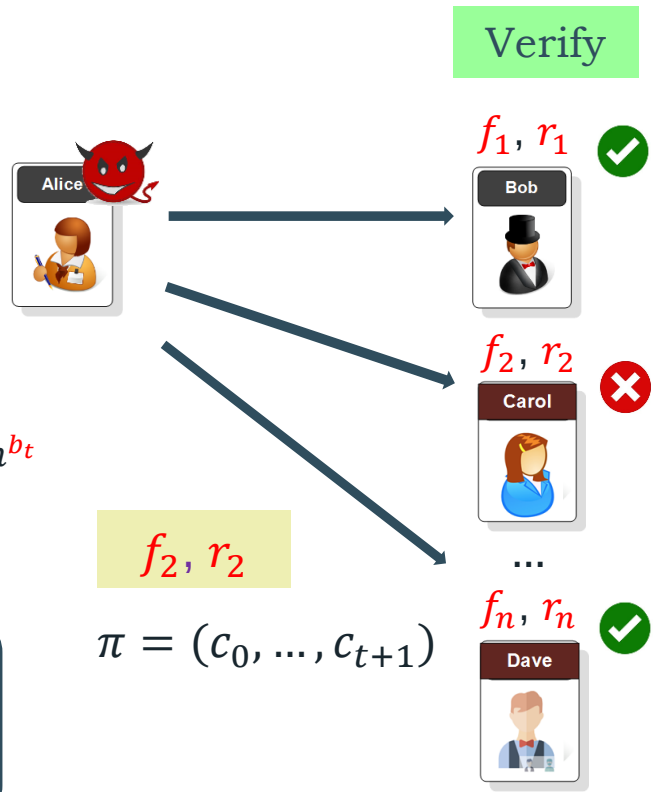
- Pedersen VSS: $t$ of shareholders (information theoretically) learns nothing about $f_0$
- Pedersen Commitment: $com(f,r) = g^f h^r$ (Perfectly Hiding + Computationally Binding)

**Share**

- Given a secret $f_0$ and $(n,t)$
- Sample a degree $t$ polynomial $f(X) = f_0 + a_1 X + \ldots + a_t X^t$
- Sample another degree $t$ polynomial $r(X) = r_0 + b_1 X + \ldots + b_t X^t$
- Set $f_i = f(i)$ and $r_i = r(i)$ for $i = 1, \ldots, n$
- Sets $c_0 = g^{f_0} h^{r_0}, c_1 = g^{a_1} h^{b_1}, \ldots, c_t = g^{a_t} h^{b_t}$
- Publish $\pi = (c_0, \ldots, c_{t+1})$ and send privatly $(f_i, r_i)$ to party $P_i$

Pedersen Commitment:

$com(f,r) \times com(f',r') = g^f h^r \times g^{f'} h^{r'}$
$= g^{f+f'} h^{r+r'} = com(f+f', r+r')$

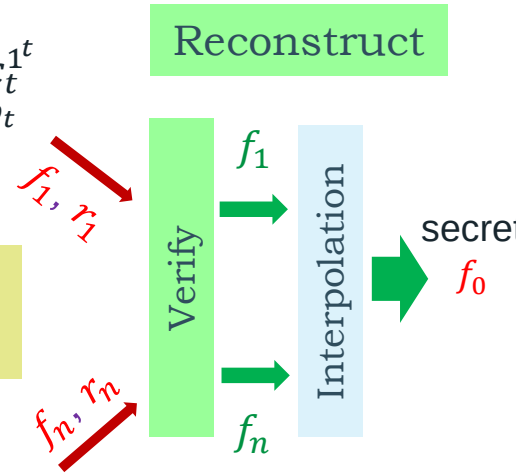



$$g^{f_i} h^{r_i} = \prod_{j=0}^{t} c_j^{i^j}$$

$$g^{f_1} h^{r_1} = c_0^{1^0} \times c_1^{1^1} \times \cdots \times c_t^{1^t}$$
$$= g^{f_0} h^{r_0} \times \cdots \times g^{a_t} h^{b_t}$$
$$= g^{f(1)} h^{r(1)}$$

$$g^{f_2} h^{r_2} = c_0^{2^0} \times c_1^{2^1} \times \cdots \times c_t^{2^t}$$
$$= \cdots = g^{f(2)} h^{r(2)}$$

...

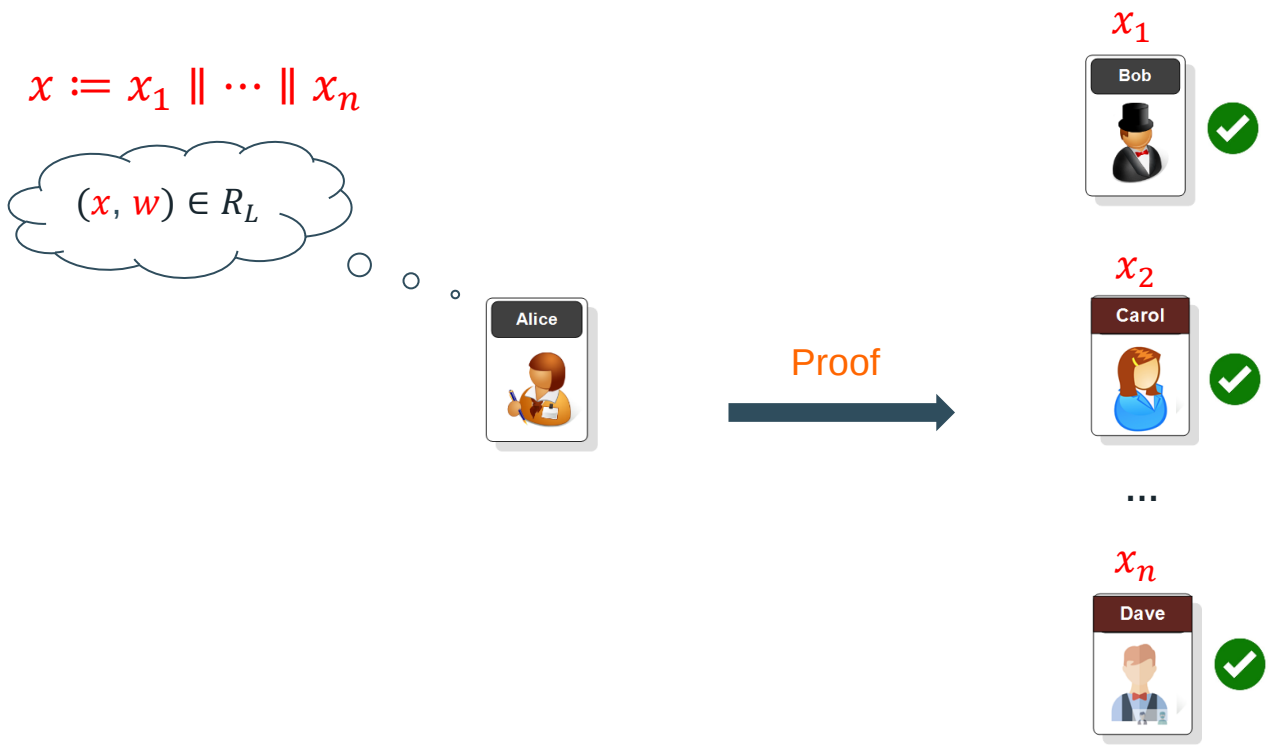





[Ped'91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. Advances in Cryptology – CRYPTO'91.

# ZK Proofs Over Secret Shared Data: [BBC+19]

- Involves a prover and $n \geq 2t + 1$ verifiers, which allows the parties to communicate in synchronous rounds.
- The prover holds an $n$-distributed input $x \coloneqq x_1 \parallel \cdots \parallel x_n$ and a witness $w$, and each $V_i$ holds an input piece $x_j$.




- **Completeness:** If $x \in L$, honest prover will convince honest verifiers.
- **Soundness against prover and $t$ verifiers:** If $x \notin L$, honest verifiers will reject the proof, except for a negligible probability, even if up to $t$ of verifiers collude with the prover.
- **Threshold Zero-Knowledge (TZK):** any subset of the verifiers up to $t$, should learn no additional information about $x$, beyond their own shares and the fact that $x \in L$.

✓ **Positive Result: We can build NI-TZK proof schemes for $n \geq 2t + 1$ [BBC+19].**

**[BBC+19]:** Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, Yuval Ishai. Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs.

# (PQ-Secure) VSS from Distributed ZK Proofs

[ABCP23]

Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. VSS from Distributed ZK Proofs and Applications. Advances in Cryptology – ASIACRYPT 2023

**ia.cr/2023/992**

# [ABCP23]: A TZK Proof for Shamir Relation

- We present a hash-based NI-TZK proof scheme for the following (Shamir) $n$-distributed relation,

$$R_i = \{(f_i, f(X)) \mid f(i) = f_i\} \quad \text{for } i = 1, \dots, n.$$



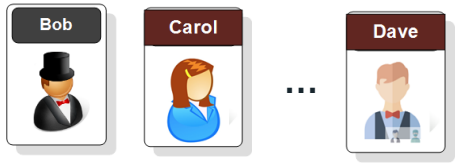

$x := f_1 \| \cdots \| f_n$

Prover




Verifier $V_i$

$(f_i, y_i, y_i')$




1) Sample a degree $t$ polynomial $r(X)$
2) For $i = 1, \dots, n$: sample $y_i$ and $y_i'$ and set $C_i = Com(r(i), y_i)$ & $C_i' = Com(f_i, y_i')$
3) Set $d = Hash(C_1, \dots, C_n, C_1', \dots, C_n)$
4) Set $z(X) = r(X) + df(X)$
5) Publish $\pi = (C_1, \dots, C_n, C_1', \dots, C_n', z(X))$ and send privatly $(f_i, y_i, y_i')$ to verifier $V_i$.

$\pi = (C_1, \dots, C_n, C_1', \dots, C_n', z(X))$



- ✓ Completeness
- ✓ Soundness against prover and $t$ verifiers
- ✓ Threshold ZK

1) Party $V_i$ → accept/reject:
   - If $deg(z(X)) \neq d$ or $C_i' \neq Com(f_i, y_i')$ reject
   - Set $d = Hash(C_1, \dots, C_n, C_1', \dots, C_n')$
   - If $C_i == Com(z(i) - df_i, y_i)$ outputs accept, otherwise, reject.
2) Return accept, *if all parties* return accept.



# [ABCP23]: A VSS Scheme from New TZK Proof

- Given the new NI-TZK proof scheme we construct a PQ secure (hash-based) NI-VSS

Dealer

$f_0$




Share

1) Given $(n, t, f_0)$: sample a degree-$t$ $f(X)$ s.t., $f(0) = f_0$ & set $f_i = f(i)$ for $i = 1, \dots, n$
2) Using $f_1 \| \cdots \| f_n$ and $f(X)$ run the new NI-TZK scheme & obtains $\pi = (C_i, C_i',\ z(X))$ and $(f_i, y_i, y_i')$ for $i = 1, \dots, n$
3) Publish $\pi = (C_i, C_i',\ z(X))$ and send privatly $(f_i, y_i, y_i')$ to participant $V_i$

$\pi = (C_1, \dots, C_n, C_1', \dots, C_n', z(X))$

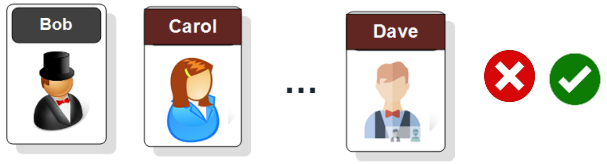
- ✓ Completeness
- ✓ Verifiable (soundness)
- ✓ Comp. Sim. (TZK)

❑ In practice we can omit $y_i$ (*and* $y_i'$ if $f_0$ has enough entropy)

Shareholder $V_i$

Verify

$(f_i, y_i, y_i')$

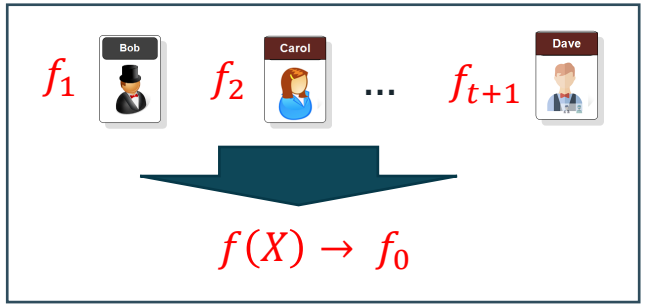



1) Party $V_i$ → outputs accept/reject:
   - If $\deg(z(X)) \neq d$ or $C_i' \neq Com(f_i, y_i')$ reject
   - Set $d = Hash(C_1, \dots, C_n, C_1', \dots, C_n')$
   - If $C_i == Com(z(i) - df_i, y_i)$ outputs accept, otherwise, reject.
2) If $V_j$ complains, dealer publish $(f_j, y_j, y_j')$, s.t., other parties can check them.
3) At the end, the honest parties either reject, or get sure that they have received a valid share.

# [ABCP23]: A New Secret Reconstruction Approach

Reconstruct

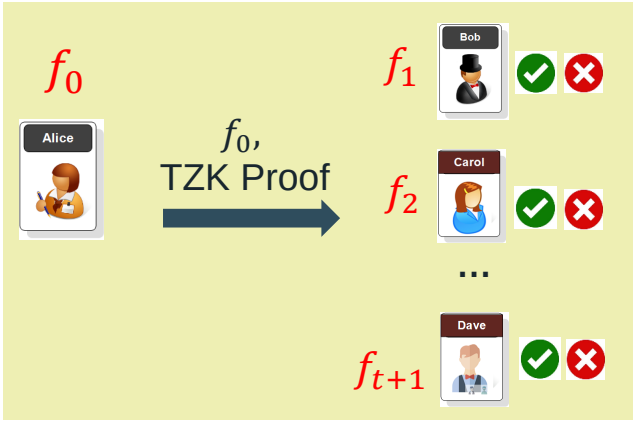
- Given $t+1$ valid shares, parties can reconstruct the secret:
  - ✓ Parties broadcast their shares, i.e., $f_1, \dots, f_n$ → verify the shares and collect at least $t+1$ valid ones. Return false if could not collect $t+1$ valid shares.
  - ✓ Interpolate $f(X)$ of degree $t$ and output $f(0)$.




- New Approach:
  - ✓ The dealer publishes $f_0 = f(0)$ along with a NI-TZK proof for the distributed relation,

$$R_i = \{(f_0, f_i, f(X)) \mid f(0) = f_0 \wedge f(i) = f_i\} \text{ for } i = 1, \dots, n$$

- the proof is TZK, but can be verified by any set of $t+1$ valid shares.




✓ Using the new approach, one can perform both sharing and reconstruction at the same time, and construct low-round (and more efficient) protocols.

# Performance of [ABCP23] VSS: Asymptotic Costs

- [ABCP23] VSS Scheme vs. Feldman [Fel87] and Pedersen [Ped91]
  - DL: Discrete Logarithm, BC: Broadcast, PV: Private, $n$: Number of parties, $E_G$: Exponentiation in $G$, $PE$: Degree-$d$ polynomial evaluations, $H$: Hashing, $|X|$: $X$ element size.

| VSS Scheme | High-Entropy Secret? | Assumption | Sharing | Dealer's Communication | Verification |
|---|---|---|---|---|---|
| Feldman [Fel87] | Yes | DL (Plain) | $0.5n \approx t\ E_G$<br>$n\ PE$ | PV: $n\ \|Z_q\|$<br>BC: t $\approx 0.5n\ \|G\|$ | $t \approx 0.5n\ E_G$ |
| Pedersen [Ped91] | No | DL (Plain) | $1n \approx 2t\ E_G$<br>$2n\ PE$ | PV: $2n\ \|Z_q\|$<br>BC: t $\approx 0.5n\ \|G\|$ | $t \approx 0.5n\ E_G$ |
| **[ABCP23]** | **Yes (or No)** | **Hash function (QROM)** | $\boldsymbol{2n\ H}$<br>$\boldsymbol{2n\ PE}$ | **PV: $1n\ \|Z_q\|$ (or $2n\ \|Z_q\|$)**<br>**BC: $2n\ \|H\| + 0.5n\ \|Z_q\|$** | $\boldsymbol{1\ PE + 3\ H}$ |

- ☺ Considerably faster verification
- ☺ PQ secure
- 😐 Not IT-unpredictable (vs. [Ped91])
- ☺ Faster sharing (lightweight operations)
- 😐 Slightly higher communication (~5x)

# Performance of [ABCP23] VSS: Empirical Performance

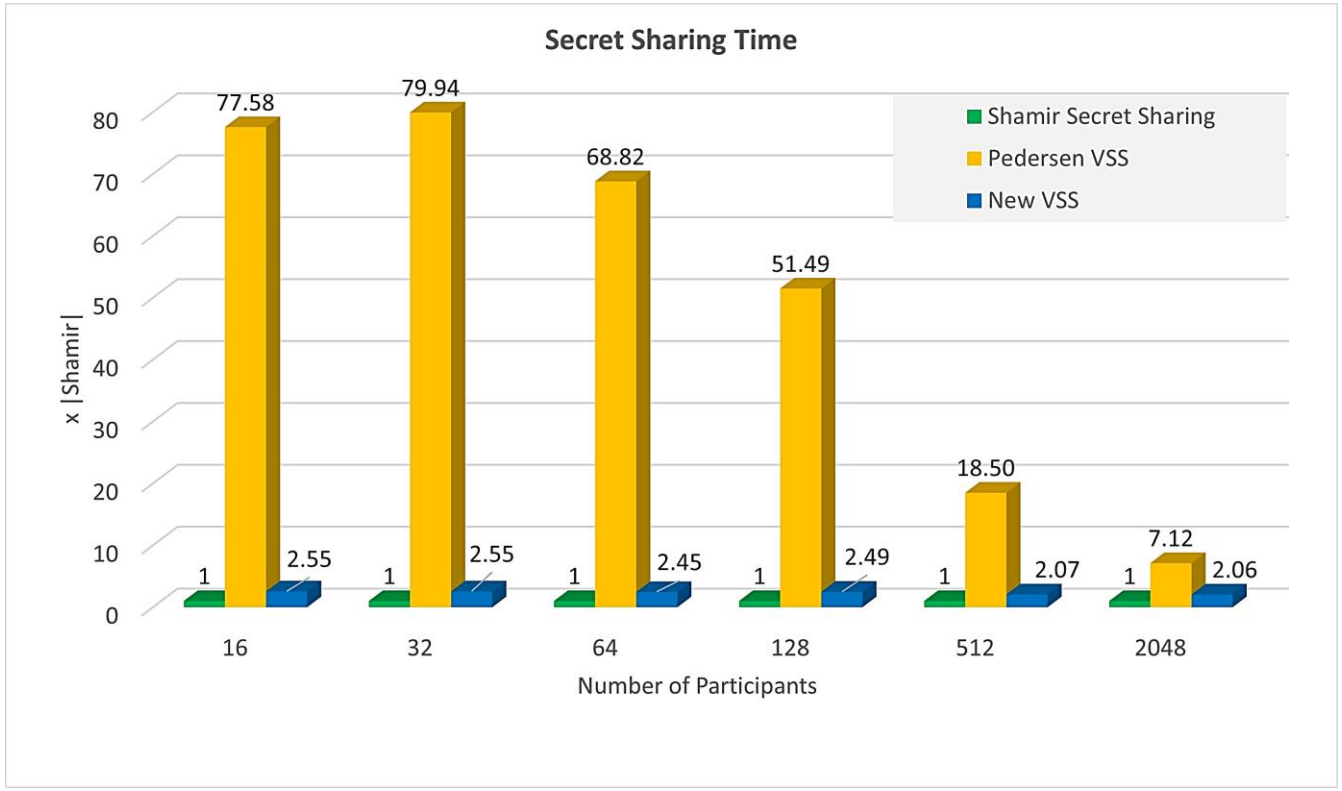
- For both commitment and random oracle, we used the SHA256 hash function.
- For the implementation of Pedersen VSS, we have used Curve 25519.
- Laptop: Ubuntu 22.04 LTS, a 11$^{th}$ Gen Intel(R) Core i9-11950H at base frequency 2.60GHz, & 64GB of memory.
- Our codes are available on [github.com/Baghery/VSS-ABCP23](github.com/Baghery/VSS-ABCP23)
- Can be run by scanning the QR-code

**Table 3.** Empirical performance of NI-VSS schemes Pedersen [28] and our proposed scheme for various numbers of parties and threshold values $(n, t)$. n: Number of parties, t: Threshold value, BC: Broadcast, $|\mathbb{Z}_q| = |\mathbb{Z}_N| = |\mathbb{G}| = |\mathcal{H}| = 256$ bits.

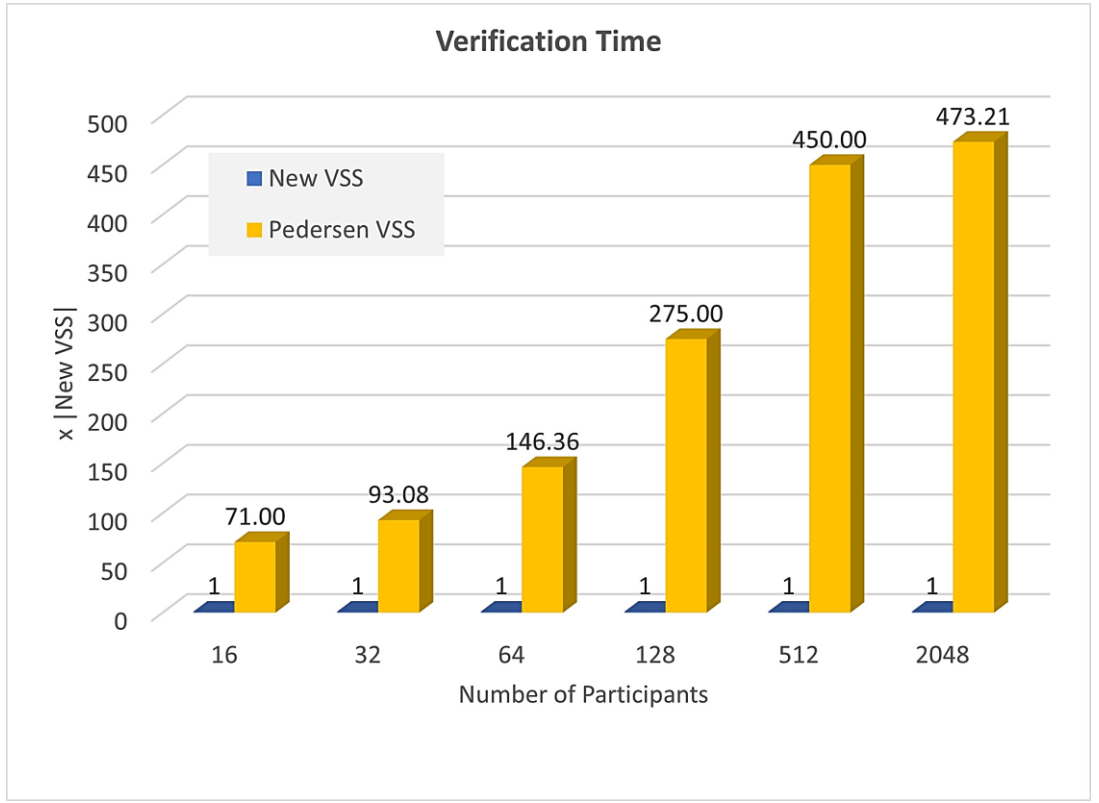
| (n, t) | VSS Scheme | Sharing | Dealer's Communication | Verification |
|---|---|---|---|---|
| (32, 15) | Pedersen [28] | 78.1 msec | Private: 2.0 KB + BC: 0.5 KB | 10.2 msec |
| | This Work | 2.40 msec | Private: 1.0 KB + BC: 2.5 KB | 0.12 msec |
| (128, 63) | Pedersen [28] | 310 msec | Private: 8.0 KB + BC: 2.0 KB | 97.6 msec |
| | This Work | 13.7 msec | Private: 4.0 KB + BC: 10.0 KB | 0.36 msec |
| (512, 255) | Pedersen [28] | 1.310 sec | Private: 32.0 KB + BC: 8.0 KB | 525 msec |
| | This Work | 0.140 sec | Private: 16.0 KB + BC: 40.0 KB | 1.20 msec |
| (2048, 1023) | Pedersen [28] | 6.53 sec | Private: 128 KB + BC: 32 KB | 2.35 sec |
| | This Work | 2.00 sec | Private: 64 KB + BC: 160 KB | 4.90 msec |
| (8192, 4095) | Pedersen [28] | 48.1 sec | Private: 512 KB + BC: 128 KB | 9.47 sec |
| | This Work | 30.6 sec | Private: 256 KB + BC: 640 KB | 0.019 sec |
| (16384, 8191) | Pedersen [28] | 153.6 sec | Private: 1024 KB + BC: 256 KB | 19.2 sec |
| | This Work | 121.7 sec | Private: 512 KB + BC: 1280 KB | 0.039 sec |

# The Overhead to Achieve Verifiability: Pedersen, New VSS

**Comparison of sharing time in standard Shamir secret sharing and VSS schemes.**




**Comparison of verification time for new VSS and Pedersen scheme**



# Π: A Unified Framework for Computational Verifiable Secret Sharing

## (& An Efficient Publicly Verifiable Secret Sharing (PVSS))

[Bag25]

Karim Baghery. Π: A Unified Framework for Computational Verifiable Secret Sharing.
Public-Key Cryptography - PKC 2025

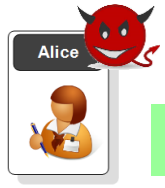
**ia.cr/2023/1669**

# Π: A Unified Framework for Computational VSS

- Our construction from [ABCP23] can be optimized and generalized
- Let $Com(.,.)$ be a commitment scheme, and $(Q)ROM$ be a (Quantum) ROM

Dealer

$f_0$




Share

1) Given $(f_0, n, t)$: sample a degree-$t$ $f(X)$ s.t., $f(0) = f_0$ & set $f_i = f(i)$ for $i = 1, \dots, n$
2) Sample a degree-$t$ polynomial $r(X)$ & set $r_i = r(i)$ for $i = 1, \dots, n$
3) For $i = 1, \dots, n$: sample $\gamma_i$, and set $C_i = Com((f_i, r_i), \gamma_i)$
4) Set $d = (Q)ROM(C_1, \dots, C_n)$
5) Set $z(X) = r(X) + df(X)$ and $\pi = (z(X), C_1, \dots, C_n)$.
6) Publish $\pi = (C_1, \dots, C_n,\ z(X))$ and send privatly $(f_i, \gamma_i)$ to participant $V_i$

$(f_i, \gamma_i)$

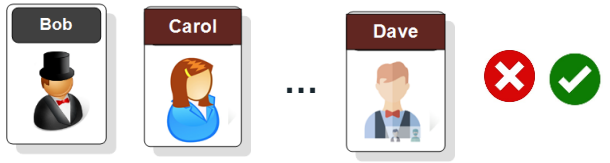
$\pi = (C_1, \dots, C_n, z(X))$



- ✓ Complete
- ✓ Verifiable
- ✓ Sim. (or Unpredictable)

❑ In practice we can omit $\gamma_i$ if $f_0$ has enough entropy

Shareholder $V_i$ $(f_i, \gamma_i)$ Verify




1) Party $V_i$ → outputs accept/reject:
   - If $\deg(z(X)) > d$ reject
   - Set $d = (Q)ROM(C_1, \dots, C_n)$
   - If $C_i == Com((f_i, z(i) - df_i), \gamma_i)$ outputs accept, otherwise, reject.
2) If $V_j$ complains, dealer publish $(f_j, \gamma_j)$, s.t., other parties can check them.
3) At the end, the honest parties either reject, or get sure that they have received a valid share.



**Note:** In Turkish, 'Pay' (pronounced [paɪ]) is a noun for Share, and 'Payla' means Share it.

# Performance of New VSS Schemes: Asymptotic Costs

- Using $\Pi$ we construct alternative VSS schemes to [Fel87, Ped91, ABCP23]
  - DL: Discrete Logarithm, (Q)ROM: (Quantum) Random Oracle Model, BC: Broadcast, PV: Private, $n$: Number of parties, $E_G$: Exponentiation in $G$, $PE$: Degree-$d$ polynomial evaluations, $H$: Hashing, $|X|$: $X$ element size.

| VSS Scheme | High-Entropy Secret? | Assumption | Sharing | Dealer's Communication | Verification | Reconstruction |
|---|---|---|---|---|---|---|
| Feldman [Fel87] | Yes | DL (Plain) | $0.5n \approx t$ $E_G$<br>$n$ $PE$ | PV: $n\|Z_q\|$<br>BC: t $\approx 0.5n\|G\|$ | $t$ $E_G + t$ $M_G$ | $t^2$ $E_G + t^2$ $M_G$ |
| $\boldsymbol{\Pi_F}$<br>$(C_i = g_1^{f_i} g_2^{r_i})$ | **Yes** | **DL (ROM)** | $\boldsymbol{2n}$ $\boldsymbol{E_G}$<br>$\boldsymbol{2n}$ $\boldsymbol{PE}$ | **PV:** $\boldsymbol{n\|Z_q\|}$<br>**BC:** $\boldsymbol{n\|G\| + 0.5n\|Z_q\|}$ | $\boldsymbol{2}$ $\boldsymbol{E_G + 1}$ $\boldsymbol{PE + 1}$ $\boldsymbol{H}$ | $\boldsymbol{2t}$ $\boldsymbol{E_G + t}$ $\boldsymbol{PE + t}$ $\boldsymbol{H}$ |
| Pedersen [Ped91] | No | DL (Plain) | $1n \approx 2t$ $E_G$<br>$2n$ $PE$ | PV: $2n\|Z_q\|$<br>BC: t $\approx 0.5n\|G\|$ | $t$ $E_G + t$ $M_G$ | $t^2$ $E_G + t^2$ $M_G$ |
| $\boldsymbol{\Pi_P}$<br>$(C_i = g_1^{f_i} g_2^{r_i} g_3^{\gamma_i})$ | **No** | **DL (ROM)** | $\boldsymbol{3n}$ $\boldsymbol{E_G}$<br>$\boldsymbol{2n}$ $\boldsymbol{PE}$ | **PV:** $\boldsymbol{2n\|Z_q\|}$<br>**BC:** $\boldsymbol{n\|G\| + 0.5n\|Z_q\|}$ | $\boldsymbol{3}$ $\boldsymbol{E_G + 1}$ $\boldsymbol{PE + 1}$ $\boldsymbol{H}$ | $\boldsymbol{3t}$ $\boldsymbol{E_G + t}$ $\boldsymbol{PE + t}$ $\boldsymbol{H}$ |
| [ABCP23] | Yes (or No) | Hash function (QROM) | $2n$ $H$<br>$2n$ $PE$ | PV: $1n\|Z_q\|$ (or $2n\|Z_q\|$)<br>BC: $2n\|H\| + 0.5n\|Z_q\|$ | $1$ $PE + 3$ $H$ | $t$ $PE + 3t$ $H$ |
| $\boldsymbol{\Pi_{LA}}$<br>$(C_i = Hash(f_i, r_i, \sout{\gamma_i}))$ | **Yes (~~or No~~)** | **Hash function (QROM)** | $\boldsymbol{n}$ $\boldsymbol{H}$<br>$\boldsymbol{2n}$ $\boldsymbol{PE}$ | **PV:** $\boldsymbol{1n\|Z_q\|}$ **(or** $\boldsymbol{2n\|Z_q\|}$**)**<br>**BC:** $\boldsymbol{n\|H\| + 0.5n\|Z_q\|}$ | $\boldsymbol{1}$ $\boldsymbol{PE + 2}$ $\boldsymbol{H}$ | $t$ $\boldsymbol{PE + 2t}$ $\boldsymbol{H}$ |



**Note:** In Turkish, 'Pay' (pronounced [paɪ]) is a noun for Share, and 'Payla' means Share it.

# Empirical Performance: [Ped91, ABCP23] vs. $\Pi_P$ & $\Pi_{LA}$

- For both commitment and (quantum) random oracle, we used the SHA256 hash function.
- For the implementation of Pedersen and $\Pi_P$ VSS schemes, we have used Curve 25519.
- Laptop: Ubuntu 22.04 LTS,
  a 11th Gen Intel(R) Core i9-11950H at base frequency 2.60GHz, & 128GB of memory.
- Our codes are available on github.com/Baghery/VSS-Pi as well as github.com/KULeuven-COSIC/Pi-VSS
- Can be run by scanning the QR-code

**Table 2.** Implementation results of VSS schemes Pedersen [23], $\Pi_P$, ABCP [1], and $\Pi_{LA}$. n: Number of parties, t: Threshold value, $|\mathbb{Z}_q| = |\mathbb{Z}_N| = |\mathbb{G}| = |\mathcal{H}| = 256$ bits.

| $(n, t)$ | Metrics | Pedersen [23] | $\Pi_P$ | ABCP [1] | $\Pi_{LA}$ |
|---|---|---|---|---|---|
| (32, 15) | Share | 74.8 msec | 222.1 msec | 2.2 msec | 1.7 msec |
| | Verify | 10.7 msec | 7.1 msec | 0.13 msec | 0.10 msec |
| | Dealer's Broadcast | 0.5 KB | 1.5 KB | 2.5 KB | 1.5 KB |
| | Dealer's Private Com. | 2.0 KB | 2.0 KB | 1.0 KB | 1.0 KB |
| (128, 63) | Share | 303 msec | 852 msec | 13.2 msec | 11.4 msec |
| | Verify | 99.2 msec | 8.3 msec | 0.37 msec | 0.33 msec |
| | Dealer's Broadcast | 2.0 KB | 6.0 KB | 10.0 KB | 6.0 KB |
| | Dealer's Private Com. | 8.0 KB | 8.0 KB | 4.0 KB | 4.0 KB |
| (512, 255) | Share | 1.29 sec | 3.63 sec | 0.13 sec | 0.12 sec |
| | Verify | 552 msec | 12.5 msec | 1.2 msec | 1.1 msec |
| | Dealer's Broadcast | 8.0 KB | 24.0 KB | 40.0 KB | 24.0 KB |
| | Dealer's Private Com. | 32.0 KB | 32.0 KB | 16.0 KB | 16.0 KB |
| (2048, 1023) | Share | 6.45 sec | 16.07 sec | 1.81 sec | 1.78 sec |
| | Verify | 2.32 sec | 28.3 msec | 4.9 msec | 4.8 msec |
| | Dealer's Broadcast | 32.0 KB | 96.0 KB | 160.0 KB | 96.0 KB |
| | Dealer's Private Com. | 128.0 KB | 128.0 KB | 64.0 KB | 64.0 KB |
| (8192, 4095) | Share | 47.1 sec | 84.6 sec | 28.6 sec | 28.5 sec |
| | Verify | 9.38 sec | 0.092 sec | 0.020 sec | 0.018 sec |
| | Dealer's Broadcast | 128 KB | 384 KB | 640 KB | 384 KB |
| | Dealer's Private Com. | 512 KB | 512 KB | 256 KB | 256 KB |
| (16384, 8191) | Share | 149.0 sec | 221.69 sec | 112.0 sec | 111.5 sec |
| | Verify | 18.7 sec | 0.178 sec | 0.070 sec | 0.050 sec |
| | Dealer's Broadcast | 256 KB | 768 KB | 1280 KB | 768 KB |
| | Dealer's Private Com. | 1024 KB | 1024 KB | 512 KB | 512 KB |

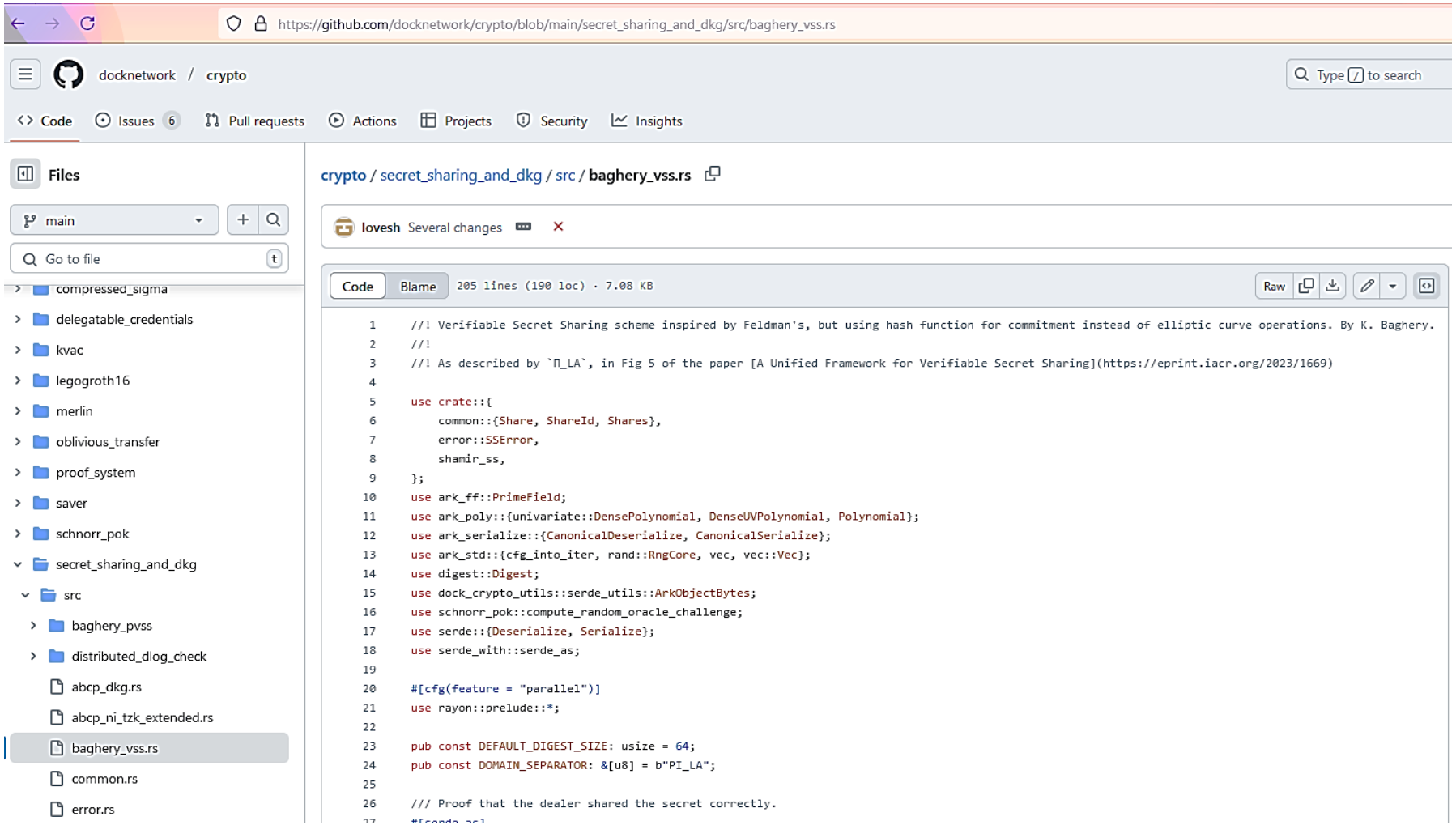
# Implementation of $\Pi_{LA}$ by Dock Network:

- ✓ **The new VSS scheme $\Pi_{LA}$ was recently implemented in Rust by the developers of Dock Network, www.dock.io, to be used in real world applications.**

- ✓ Their open-source implementations are publicly available on:

https://github.com/docknetwork/crypto/blob/main/secret_sharing_and_dkg/src/baghery_vss.rs

QR-code:



# $\Pi_S$: an Efficient Publicly Verifiable Secret Sharing (PVSS) Based on $\Pi_F$ VSS Scheme

[Bag25]

Karim Baghery. $\Pi$: A Unified Framework for Computational Verifiable Secret Sharing.
Public-Key Cryptography - PKC 2025

**ia.cr/2023/1669**

# Publicly Verifiable Secret Sharing (PVSS):

- Roughly speaking, in a Shamir-based VSS scheme a Prover (so called dealer) who knows a secret degree-$t$ polynomial $f(X)$ generates a threshold ZK proof for the following $n$-distributed relation,

$$R_i = \{(f_i, f(X)) \mid f(i) = f_i\} \quad \text{for } i = 1, \dots, n.$$

The threshold ZK proof is designated and can only be verified by at least $t + 1$ distinct values of "$f_i$"s.

- Common VSS schemes → have happy and unhappy paths → 1 (happy) or 3 (unhappy) rounds
- [Fel87, Ped91, ABCP23] and all the schemes built by **Π** → 1 (happy) or 3 (unhappy) rounds

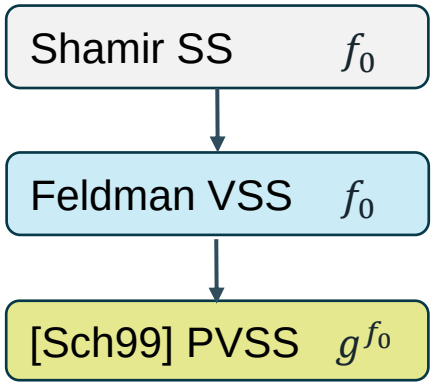
- In a Shamir-based PVSS scheme, given the public keys $\{pk_i\}_{i=1}^n$, the Prover encrypt the Shamir shares and proves that the sharing and encryption are done correctly. E.g., it generates a NIZK proof for

$$R_{PVSS} = \{(C_i, pk_i, f(X)) \mid C_i = Enc(f(i), pk_i)\} \text{ for } i = 1, \dots, n.$$

- PVSS schemes do not have happy/unhappy paths and given the public keys they require a single round.
- In PVSS schemes, the parties first register their public keys $\{pk_i\}_{i=1}^n$ and store their secret keys.

# Schoenmakers' PVSS Scheme:

- In some PVSS schemes the main secret is $g^{f_0}$ [Sch99] and in some others it is $f_0 \in Z_q$ [Gro21].
- In CRYPTO'99, Schoenmakers [Sch99]:
  - Proposed an efficient PVSS protocol based on [Fel87] VSS scheme.
  - In his PVSS scheme, the secret is $g^{f_0}$ and parties obtain $g^{f_i}$ as a share of $g^{f_0}$.
  - In his scheme, given the public key $\{pk_i = g^{s_i}\}_{i=1}^{n}$ and the secret value $f_0$, the dealer runs Feldman VSS and obtains the shares $\{f_i\}_{i=1}^{n}$ and the proof $\pi_{Feldman}$. Then, the prover encrypts the shares as $C_i = pk_i^{f_i}$ and generates another NIZK proof that the encrypted values are valid Shamir shares.




- He also introduced several practical applications for his PVSS scheme, e.g., a universally verifiable e-voting protocol, threshold PKI, …

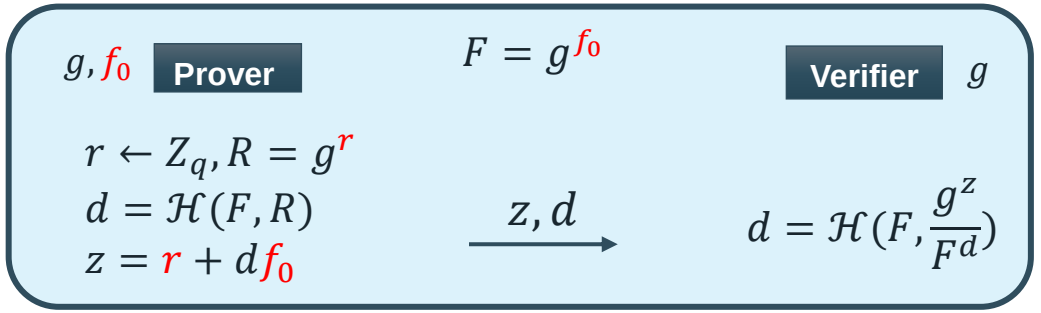
| PVSS Scheme | Assumption | Sharing | Dealer's Communication | Verification | Reconstruction |
|---|---|---|---|---|---|
| [Sch99] | DDH & RO | $4.5n\ E_G + n\ PE$ | PV: –<br>BC: $1.5n\ \|G\| + n\ \|Z_q\|$ | $nt + 4n\ E_G + 2.5n\ M_G$ | $5t\ E_G + t\ M_G$ |



[Sch99] Berry Schoenmakers. A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting. CRYPTO'99.

# Generalizing DL Assumption Over Polynomials:

- **Discrete Logarithm (DL) Problem:** Given $(g, F = g^{f_0})$ its computationally hard to recover $f_0$.
- **Schnorr's $\sum$-Protocol:** Allows a prover to prove knowledge of $f_0$ for $R_{DL} = \{((g, F), f_0) \mid F = g^{f_0}\}$.




**Polynomial Discrete Logarithm (PDL) Problem:** Let $(n, t)$ be integer values s.t., $n > t$, and $x_1, \ldots, x_n$ be public distinct elements from $Z_q$. Given $(g,\ x_i,\ F_i = g^{f(x_i)})$ for $i = 1, \ldots, n$, its computationally hard to recover the secret polynomial $f(X) \in Z_q[X]_t$ of degree (up to) $t$.

**We show that the hardness of PDL problem can be reduced to that of DL problem.**

**DL ⇔ PDL**

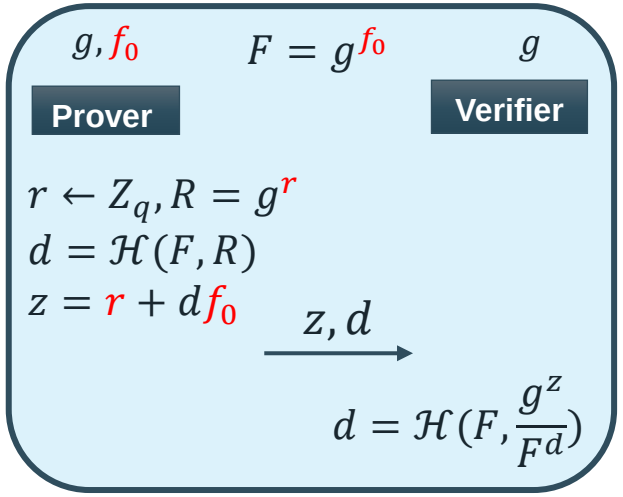
**Generalizing Schnorr's $\sum$-Protocol:** Allows a prover to prove knowledge of a (up to) degree-$t$ secret polynomial $f(X)$ for $R_{PDL}$ defined as,

$$R_{PDL} = \{((g, x_i, F_i), f(X)) \mid F_i = g^{f(x_i)}\} \text{ for } i = 1, \ldots, n \text{ and } n > t.$$

# An Efficient NIZK Proof of Knowledge for $R_{PDL}$:

**Generalization of Schnorr's Protocol:**

It allows a prover to prove knowledge of a witness polynomial $f(X)$ for the $R_{PDL}$ relation:




$$R_{PDL} = \{((g, x_i, F_i), f(X)) \mid F_i = g^{f(x_i)}\} \text{ for } i = 1, \ldots, n \text{ and } n > t.$$

**Prover:** Given the statement $(g, x_1, \ldots, x_n, F_1, \ldots, F_n)$ and the witness polynomial $f(X)$, proceed as follows and output a proof $\pi$.

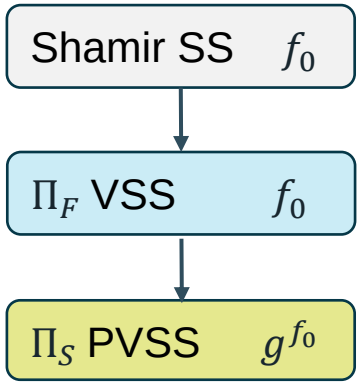
1. Sample a degree-$t$ polynomial $r(X) \in \mathbb{Z}_q[X]_t$; Set $\{\Gamma_i = g^{r(x_i)}\}_{i=1}^n$.
2. Set $d \leftarrow \mathcal{H}(F_1, \ldots, F_n, \Gamma_1, \ldots, \Gamma_n)$, where $\mathcal{H}$ is a random oracle.
3. Set $z(X) = r(X) + d \cdot f(X) \pmod q$;
4. Return $\pi := (\Gamma_1, \ldots, \Gamma_n, z(X))$

**Verifier:** Given statement $(g, \{x_i, F_i\}_{i=1}^n)$ and $\pi := (\Gamma_1, \cdots, \Gamma_n, z(X))$, the verifier first checks if $z(X)$ is a degree-$t$ polynomial. If so, then sets $d \leftarrow \mathcal{H}(F_1, \ldots, F_n, \Gamma_1, \ldots, \Gamma_n)$ and checks if: $g^{z(x_i)} = \Gamma_i (F_i)^d$ for $i = 1, \ldots, n$, and outputs `true` or `false`. Note that to make the communication shorter, as in Schnorr's ID protocol, alternatively, the prover could publish $\pi := (d, z(X))$, and then the verifier would need to check if $z(X)$ is a degree-$t$ polynomial and $d = \mathcal{H}(F_1, \ldots, F_n, \frac{g^{z(x_1)}}{F_1^d}, \ldots, \frac{g^{z(x_n)}}{F_n^d})$.

We prove that the generalized protocol satisfies **Completeness**, **Special Sound**, **HVZK** under PDL problem, and can be made non-interactive in the RO model.



# $\Pi_S$: An Efficient PVSS Scheme Based on $\Pi_F$ VSS




- **Dealer**: In $\Pi_S$, given $(n, t, pk_1 = g^{s_1}, \dots, pk_n = g^{s_n}, f_0)$, the dealer acts as follows:
  - Does Shamir secret sharing using $f(X)$ and obtains the shares $\{f_i = f(i)\}_{i=1}^{n}$.
  - Given $\{pk_i\}_{i=1}^{n}$, it sets $C_i = pk_i^{f_i}$ for $i = 1, \dots, n$. Then, it uses a variant of the generalized Schnorr protocol, and generates a NIZK proof $\pi_{share}$.
    1) Samples $r(X)$ and Sets $\Gamma_i = pk_i^{r(i)}$ for $i = 1, \dots, n$.
    2) Computes $d = RO(C_1, \dots, C_n, \Gamma_1, \dots, \Gamma_n)$ and $z(X) = r(X) + df(X)$.
    3) Broadcasts $\pi_{share} := (z(X), d)$ as the proof and $C_i = pk_i^{f_i}$ for $i = 1, \dots, n$ as the encrypted shares.

- **Verifier**: Given $(n, t, pk_1, \dots, pk_n, C_1, \dots, C_n, z(X), d)$, a public verifier checks if the followings hold,

$$\deg\big(z(X)\big) \le t \quad \text{and} \quad d = RO\left(C_1, \dots, C_n, \frac{pk_1^{z(1)}}{C_1^d}, \dots, \frac{pk_1^{z(1)}}{C_1^d}\right)$$

- **Reconstruction**: As in [Sch99], a qualified set Q of parties decrypt their encrypted shares and obtain their shares $\left\{D_k := g^{f_k} = C_k^{1/s_k}\right\}_{k \in Q}$.
  They, also attach a proof for the correctness of decryption. Then, given any $t+1$ of valid shares $\{D_k = g^{f_k}\}_{k \in Q}$, one can obtain the secret $g^{f_0}$ by Lagrange interpolation.

We show that the new PVSS scheme satisfies **Completeness**, **Verifiability** and **Secrecy** under PDL and DDH assumptions in the RO model.

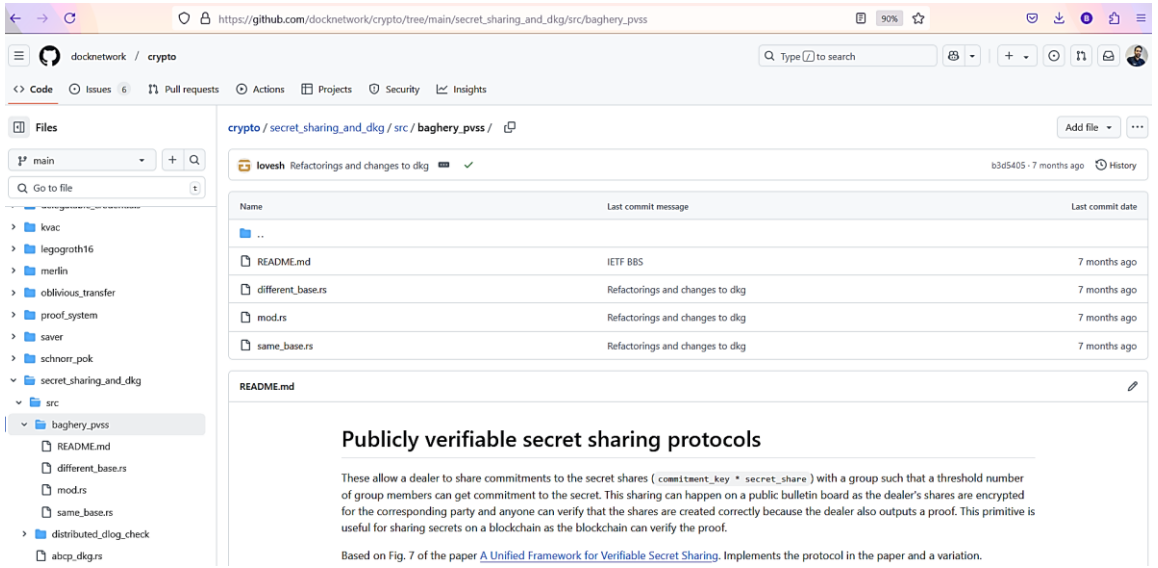


# Performance of $\Pi_S$ PVSS Scheme: Asymptotic Costs

| PPVSS | Assumption | Sharing | Dealer's Com. | Verification | Reconstruction |
|---|---|---|---|---|---|
| [Sch99] | DDH, RO | $4.5n\ E_G + n\ PE$ | $1.5n\ \|G\| + n\ \|Z_q\|$ | $nt + 4n\ E_G + 2.5n\ M_G$ | $5t\ E_G + t\ M_G$ |
| [CD17] | DBS, Plain | $2n\ E_G + n\ PE$ | $2n\ \|G\|$ | $2n\ P_G + n\ E_G + n\ M_G$ | $2t\ P_G + t\ E_G + t\ M_G$ |
| [CD17] | DDH, RO | $4n\ E_G + n\ PE$ | $2n\ \|G\| + n\ \|Z_q\|$ | $5n\ E_G + 3n\ M_G$ | $5t\ E_G + t\ M_G$ |
| $\Pi_S$ , [CD22] | DDH, PDL, RO | $2n\ E_G + 2n\ PE$ | $n\ \|G\| + 0.5n\ \|Z_q\|$ | $2n\ E_G + n\ PE + n\ M_G$ | $5t\ E_G + t\ M_G$ |

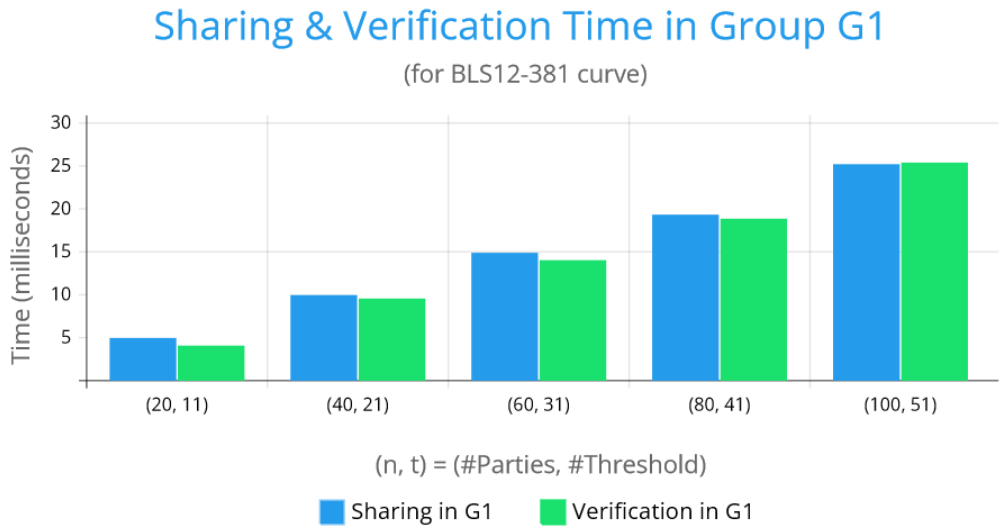
- ✓ **The PVSS scheme $\Pi_S$ is implemented in Rust by the developers of Dock Network, www.dock.io, to be used in blockchains.**
- ✓ Their open-source implementations are publicly available on:

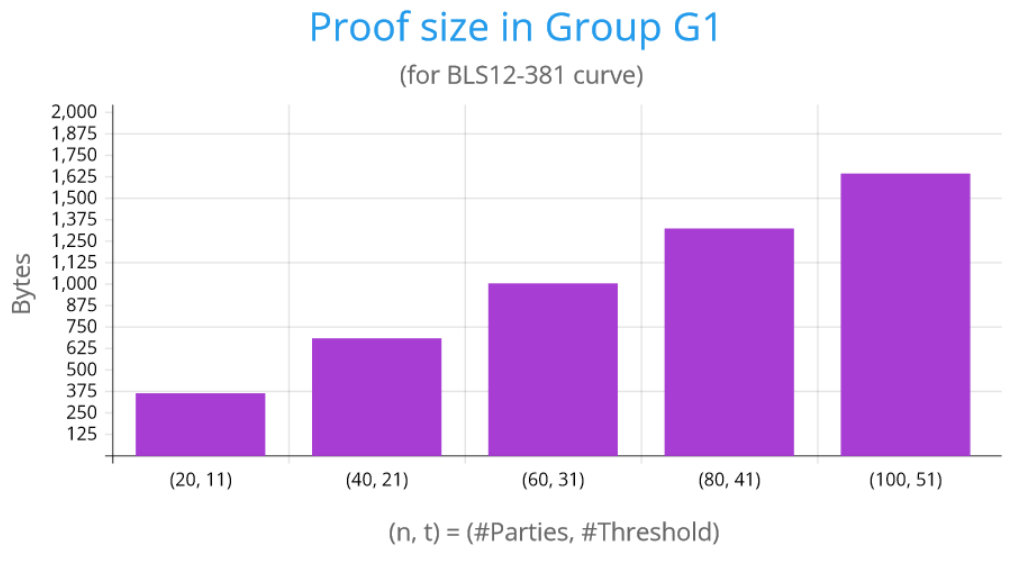
  https://github.com/docknetwork/crypto/tree/main/secret_sharing_and_dkg/src/baghery_pvss



# Performance of $\Pi_S$: Run by the Developers of Dock Network

- ✓ **Performance of $\Pi_S$ over BLS12-381 curve (by Dock Network, www.dock.io)**
  https://github.com/docknetwork/crypto/tree/main/secret_sharing_and_dkg/src/baghery_pvss




Made with Livegap Charts

**"This primitive is useful for sharing secrets on a blockchain as the blockchain can verify the proof."**

— Git Page of DOCK Network

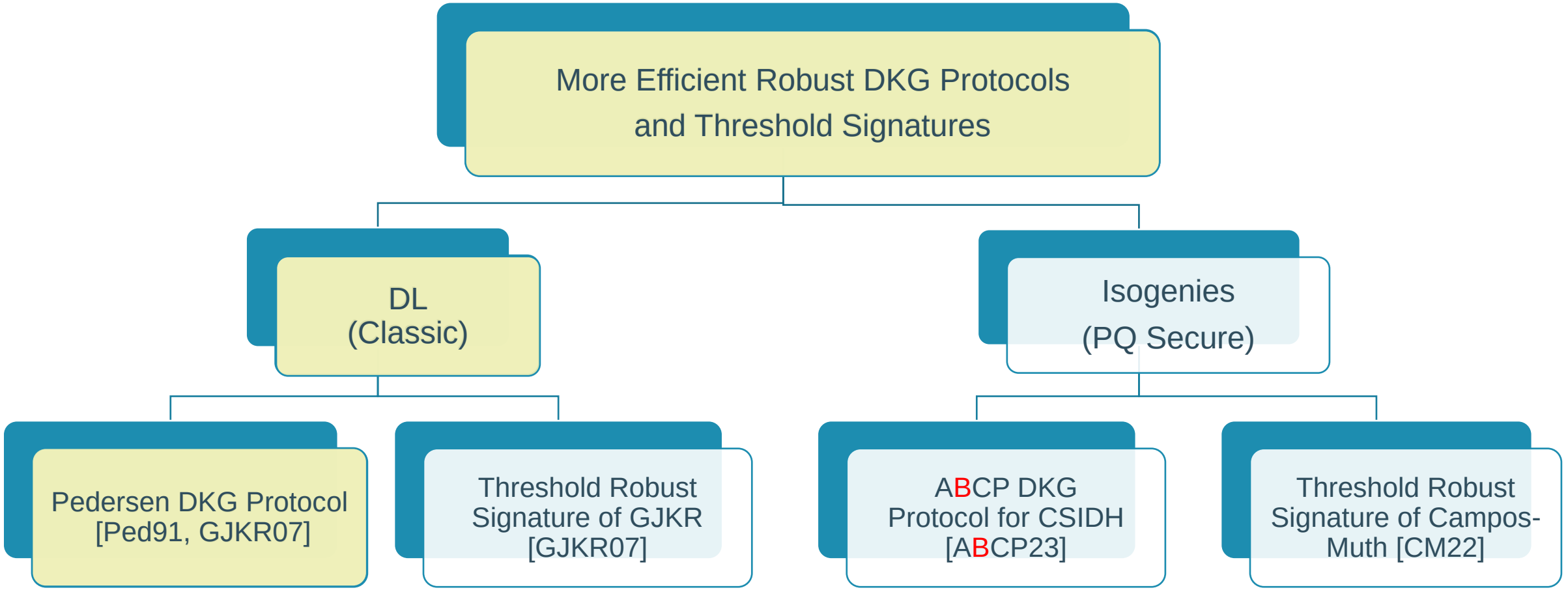
# More Efficient Threshold Protocols Based on New VSS Schemes

[ABCP23]

Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. VSS from Distributed ZK Proofs and Applications. Advances in Cryptology – ASIACRYPT 2023

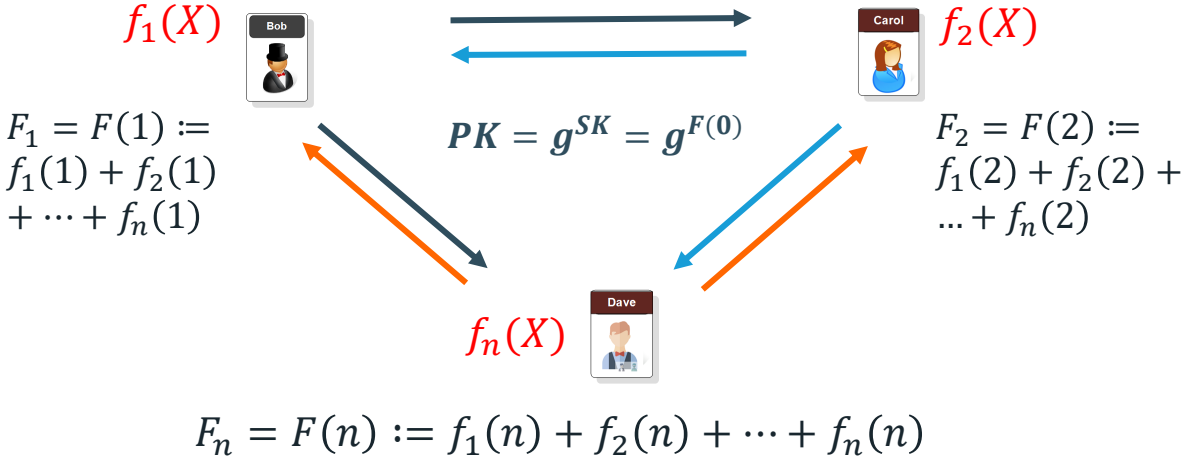
**ia.cr/2023/992**

# More Efficient Robust Threshold Protocols:



# A More Efficient Robust DKG Protocol for DL:

- Pedersen DKG [Ped91] allows a set of parties to sample $PK = g^{SK}$ in a distributed manner.




- TZK proof scheme for DL → More efficient DKG protocol for DL (2-round in the happy path)

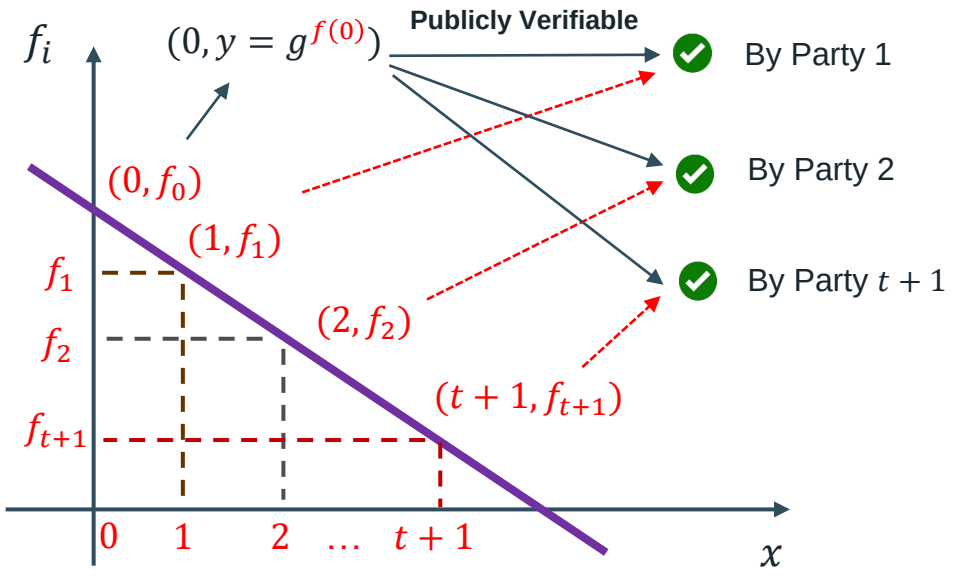
✓ NI-TZK for DL can be of an independent interest

✓ In a follow-up work it is called Group-Oriented VSS [GroSho'24]

- **Key building block**: a NI-TZK proof scheme for

$$R_i = \{(y, f_i, f(X)) \mid y = g^{f(0)} \wedge f(i) = f_i\} \; \forall \, i = 1, \ldots, n.$$

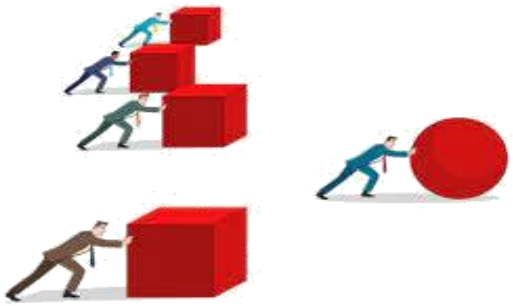
✓ Allows a party to prove that he/she has generated (partial public key) $y$ using the shared secret $f(0)$.



# Efficiency of New DKG Protocol: Asymptotic Costs

- Pedersen DKG protocol [Ped91, GJKR07] vs. our proposed variant
  - DL: Discrete Logarithm, $n$: number of parties, t $\approx n/2$: threshold parameter, $E_G$: Exponentiation in $G$, $PE$: Degree-$d$ polynomial evaluation, $H$: Hashing, $|X|$: $X$ element size.

| DKG Scheme | Assumption | Parties' Computation | Communication |
|---|---|---|---|
| Pedersen [Ped91, GJKR07] | DL | $(2nt + n)\ E_G + 2n\ PE + 1\ H$ | Private: $2n\ \|Z_q\|$<br>Broadcast: $2\text{t} \approx n\ \|G\|$ |
| **This Work** | DL & Hash | $2n\ E_G + 3n\ PE + 5\text{n}\ H$ | Private: $1n\ (or\ 2n)\ \|Z_q\|$<br>Broadcast: $2n\ \|H\| + t\ \|Z_N\|$ |



In new DKG protocol each party needs to compute $O(n)$ exponentions instead of $O(n^2)$.

# Traceable Verifiable Secret Sharing and Applications

Karim Baghery, Ehsan Ebrahimi, Omid Mirzamohammadi, Mahdi Sedaghat

ia.cr/2025/318

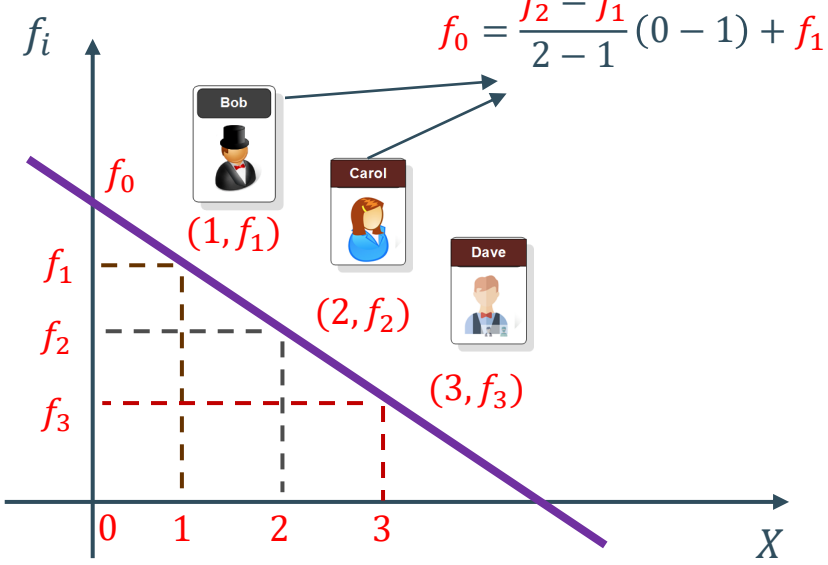
# Limitations of Shamir Secret Sharing and TVSS:

- Shamir secret sharing does not protect against malicious dealer and malicious shareholders.
  - ✓ **Verifiable Secret Sharing [Fel87, Ped91, ABCP23, CCG24, Bag25, ...]**
- In Shamir secret sharing any set of $t+1$ parties can reconstruct the secret and all the shares! In other words, secret reconstruction is not accountable!
  - ✓ **Traceable Secret Sharing [GSS21, BPR24]**
    - ✓ **Non-Imputable:** (No adversary can falsely accuse a particular party, even if it has access to the shares of all other parties).
    - ✓ **Traceable** (Secret Reconstruction is Accountable)
    - × **Not Verifiable** ([BPR24] left it as a research question)

| Scheme | Verifiability | Traceability | Non-Imputability | Share Size |
|---|---|---|---|---|
| SS-Shmair [27] | ✗ | ✗ | ✗ | $1\|s\|$ |
| VSS-Feldman [15] | ✓ | ✗ | ✗ | $1\|s\|$ |
| VSS-Pedersen [25] | ✓ | ✗ | ✗ | $2\|s\|$ |
| VSS-ABCP [2] | ✓ | ✗ | ✗ | $3\|s\|$ |
| TSS-GSS [19] | ✗ | ✓* | ✓ | $\mathcal{O}(\|s\|^2)$ |
| TSS-BPR [6] | ✗ | ✓* | ✓ | $2\|s\|$ |

$$f_X - f_1 = \frac{f_2 - f_1}{2-1}(X-1)$$

$$f_0 = \frac{f_2 - f_1}{2-1}(0-1) + f_1$$



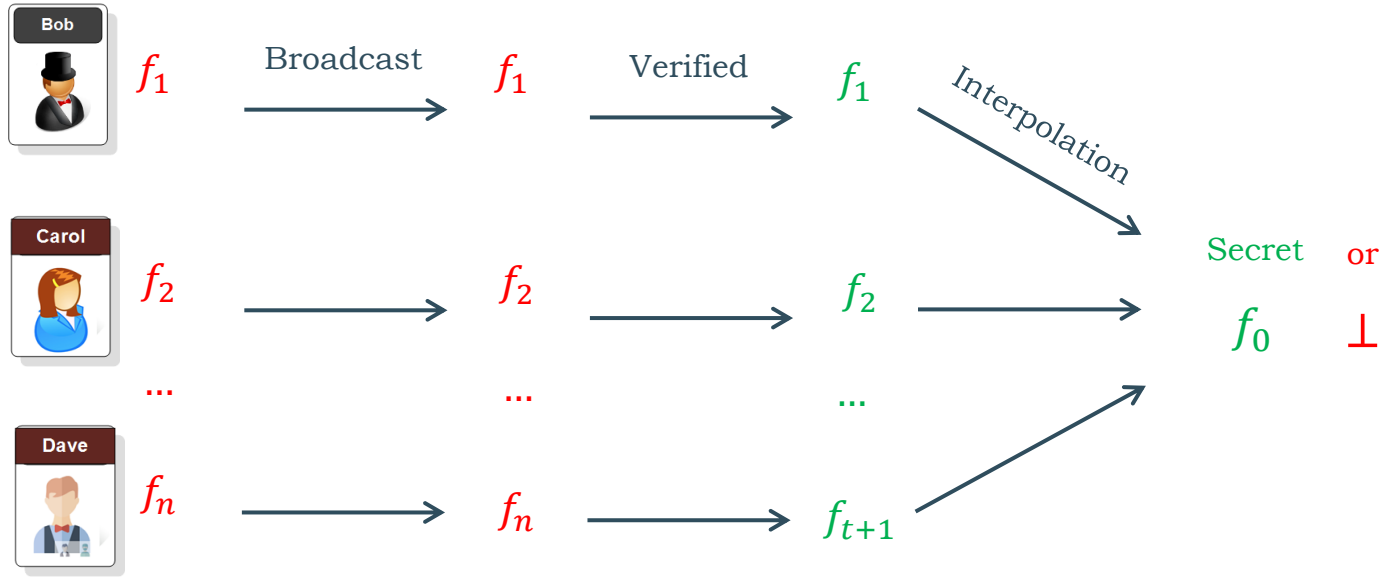
# Traceable VSS (TVSS) and Constructions:

- **In this paper, we define TVSS and present the syntax**
  - ✓ **Non-Imputable**
  - ✓ **Traceable** (Secret Reconstruction is Accountable)
  - ✓ **Verifiable** (and Completeness & Unpredictable/Secrecy)

- **Initial:** Generate the parameters
- **Share:** Share the secret & prove
- **Verify:** Verify the validity of shares
- **Reconstruct:** Reconstruct a function of the shares or the main secret
- **VerifTrace:** Verify the output of Rec. and Trace the parties involved in Rec.
- **Judge:** Verify the traced parties

In common VSS schemes: the Reconstruct algorithm typically assumes that parties broadcast the plain values of their shares. After verifying these shares using the scheme's Verify algorithm, the Reconstruct algorithm returns either

- the secret $f_0$
- or a $\perp$ symbol if it fails to collect at least t+1 valid shares.



# Our Key Ideas: Revisiting Reconstruction & Sharing



- **In TVSS schemes, we extend the Reconstruct algorithm.**
  - ✓ We allow parties to broadcast a function of their shares instead of the plain values.
    - To maintain verifiability, when parties reveal a function of their shares, they must also provide NIZK proof that these values are derived from the original shares, and they know (or hold) the shares.
  - ✓ The Reconstruct algorithm is modified to ensure it produces proof of correct execution and valid outputs.
  - ✓ The Reconstruct algorithm is modified to return the valid collected shares along with individual proofs, even if they are fewer than the threshold value t + 1 and insufficient to reconstruct the secret.

- **Part of individual shares are sampled by the parties themselves (which are unknown for the dealer and other parties).**
  - ✓ This is crucial to achieve non-imputability



Sampled by the party

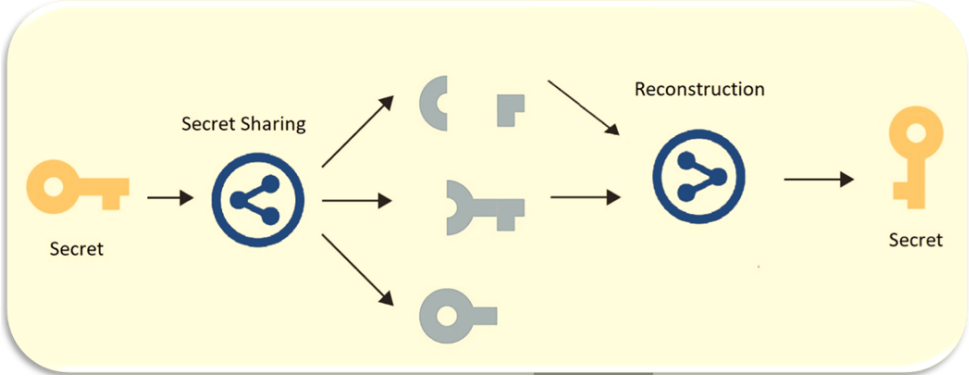


Sampled by the dealer

# Our Key Ideas: Shamir & Threshold-Sum Secret Gen.



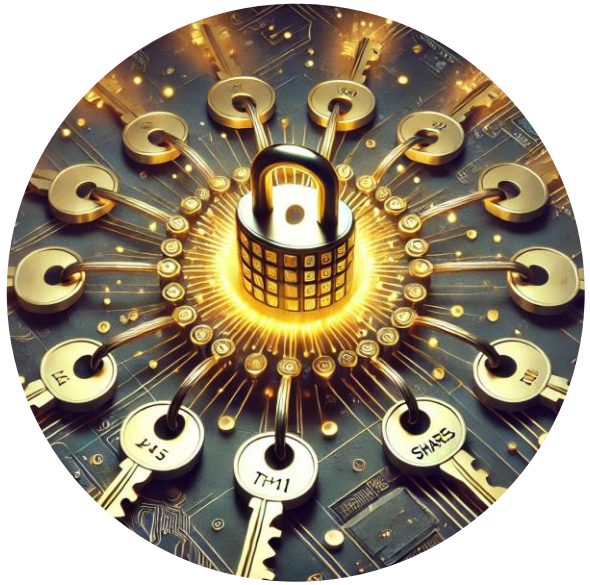
- **In Threshold Secret Sharing Schemes, like Shamir, there is a unique secret and any qualified set of parties can reconstruct a unique secret.**
  - ✓ **This make the protocol efficient, as Shamir-based protocols can have a single public kry!** ☺
  - ✓ **Having a unique secret makes it more useful!** ☺
  - ✓ But it cannot be accountable, as any $t+1$ shares are sufficient to generate all the $n$ shares! ☹




$Share_i :=$ Individual share $i$

$(f_i^{Shamir}, \gamma_i^{TSSG})$

- **Threshold Secret Generation (TSG): In a special case, which I call Threshold-Sum Secret Generation (TSSG), parties sample their own shares non-interactively and just publish a commit to it. Then, as an instance, the secret can be called "sum of at least t+1 of those individual random shares".**
  - ✓ This make the protocol less efficient, as TSG-based protocols can have a larger public key! 😐
  - ✓ Non-unique secret limits the applications of protocol. 😐
  - ✓ **But it is accountable, and parties involved in the reconstruction phase can be uniquely traced.** ☺



# Building TVSS and Accountable Threshold Cryptosystem:

- We propose a general strategy to turn A Shamir-based VSS → A Shamir-based TVSS
- In CRYPTO'24, Boneh, Partap, and Rotem studied the concept of accountability in threshold decryption protocols.
- **We leverage the new Feldman-based TVSS scheme and constructed a practical Accountable Threshold Cryptosystem based on the ElGamal cryptosystem.**

**Table 1.** Comparisons of key features and share size in various relevant constructions. SS: Secret Sharing, VSS: Verifiable Secret Sharing, TSS: Traceable Secret Sharing, TVSS: Traceable Verifiable Secret Sharing, $|s|$ indicates the size of main secret. ✓: Satisfied. ✗: Not satisfied.

| Scheme | Verifiability | Traceability | Non-Imputability | Share Size |
|---|---|---|---|---|
| SS-Shmair [27] | ✗ | ✗ | ✗ | $1\|s\|$ |
| VSS-Feldman [15] | ✓ | ✗ | ✗ | $1\|s\|$ |
| VSS-Pedersen [25] | ✓ | ✗ | ✗ | $2\|s\|$ |
| VSS-ABCP [2] | ✓ | ✗ | ✗ | $3\|s\|$ |
| TSS-GSS [19] | ✗ | ✓* | ✓ | $\mathcal{O}(\|s\|^2)$ |
| TSS-BPR [6] | ✗ | ✓* | ✓ | $2\|s\|$ |
| TVSS-Feldman (Sec. 4.1) | ✓ | ✓ | ✓ | $2\|s\|$ |
| TVSS-Pedersen (Sec. 4.2) | ✓ | ✓ | ✓ | $3\|s\|$ |

* In these schemes, the tracer must query the reconstruction box twice, and in our case only once.

# Future Research:

- New VSS schemes can be used in various classic/PQ-secure (threshold) protocols
  - In many cases, $\Pi_F$ and $\Pi_{LA}$ can be used instead of Feldman VSS.
  - Similarly, $\Pi_P$ can be used instead of Pedersen VSS.
  - $\Pi_{LA}$ is PQ-secure and assomptetically only 1x slower than standard Shamir!
- New PVSS scheme $\Pi_S$ can be used in various (blockchian) applicaitons.
- New TVSS schemes can be used to build various accountable threshold protocols.
  - We presented an accountable threshold decryption scheme based on ElGamal cryptosystem, but the idea behind it is general enougth to be applied to other cryptosystems and primitives.
- New results are coming out soon!



KU LEUVEN

# Thank You!



karim.baghery@kuleuven.be