Leakage Resilient Cheating Detectable Secret Sharing

Sabyasachi Dutta University of Calgary

Joint work with Rei Safavi-Naini

What is Secret Sharing?

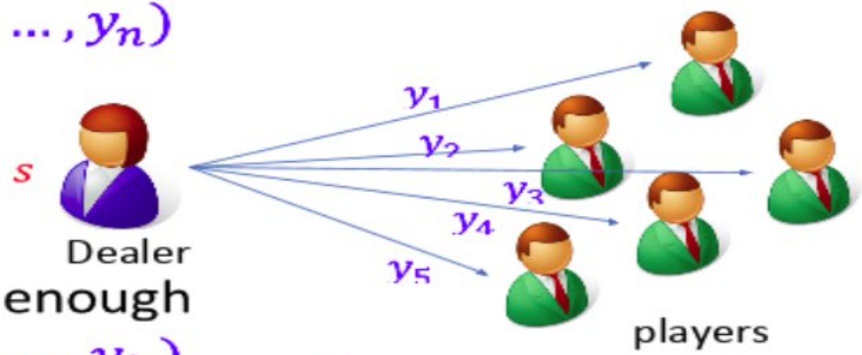
- Encryption is NOT the only way to keep Confidentiality of data
- **Secret Sharing**
    - Dividing secret in randomized way!
    - Share = “Divided, randomized data”
- Moreover :
  secret can be recovered from the shares

Shamir’s (t,n) secret sharing

Adi Shamir

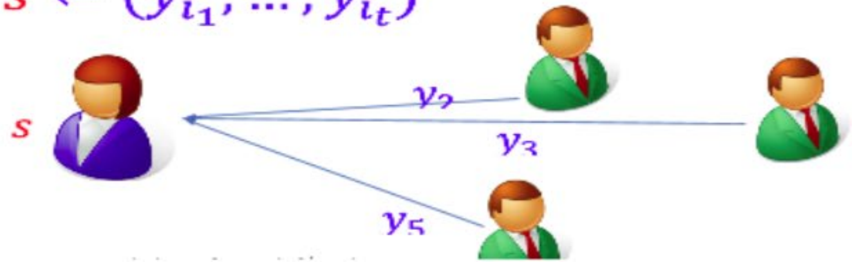
- Sharing secret s with n players

$$s \rightarrow (y_1, \dots, y_n)$$



- To recover s, t shares are enough

$$s \leftarrow (y_{i_1}, \dots, y_{i_t})$$



Sharing Phase (t=3)

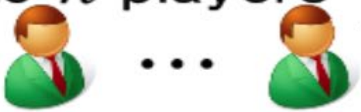
- Dealer chooses a degree $t-1$ polynomial over $\mathbb{Z}/p\mathbb{Z}$
  - ➢ s (secret to be shared) : Constant term
  - ➢ a_1, a_2 : Other coefficients chosen at random from $\mathbb{Z}/p\mathbb{Z}$ (Field)

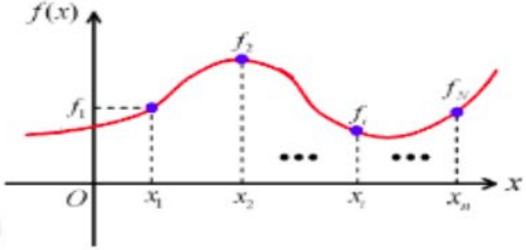
$$f\,(x) = s + a_1 x + a_2 x^2 \bmod p$$

- Dealer computes shares

$$y_i := f(x_i), i = 1, \dots, n$$

- Dealer distributes shares to n players

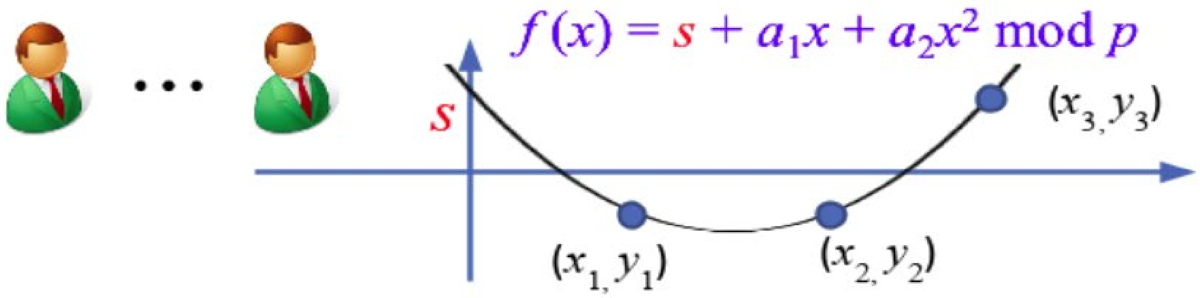




Recovery Phase t = 3

- Idea: From $t = 3$ points, compute the degree $t-1$ curve
  - $t = 3$ players are identified by x–values, $x_1 < x_2 < x_3$
  - $t = 3$ shares are y–values, $y_1,\ y_2,\ y_3$
  - Unknown, degree $t-1$ curve $y = f(x)$ can be determined from $t = 3$ points, $(x_1, y_1), (x_2, y_2), (x_3, y_3)$

Secret s is determined as the constant term!



Two main properties of any (t,n) SS:

- **Correctness :** Any **t** shares must recover the secret **s**
- **Secrecy :** Any **t-1** shares **must not reveal** any information about the **secret s**

- **Secrecy :** Any **t-1** shares **must not reveal** any information about the **secret** **$s \in Z_p$**

$Sh[i_1]$, $Sh[i_2]$, ... , $Sh[i_{t-1}]$

S = 0	S = 1	....	S = p-1
???	???	....	???

- **Secrecy :** Any **t-1** shares **must not reveal** any information about the **secret s**

$Sh[i_1]$, $Sh[i_2]$, ...

All values are equally probable as secret

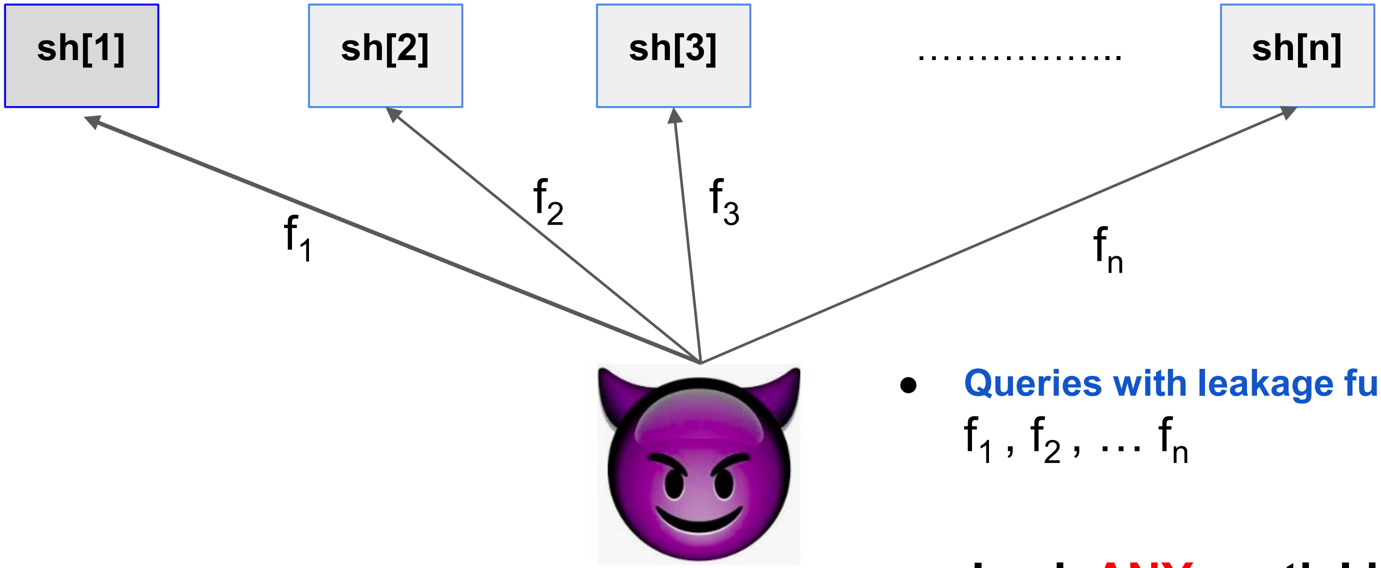
S = 0 ???	S = 1 ???	....	S = p-1 ???

Threshold Secret Sharing

- Numerous Applications
  - ➢ Secure multiparty computation [GMW87, BGW88, CCD88,...]
  - ➢ Threshold cryptographic primitives [DF90,Fra90, ….]

Security of these applications crucially depends on the SECRECY property of secret sharing

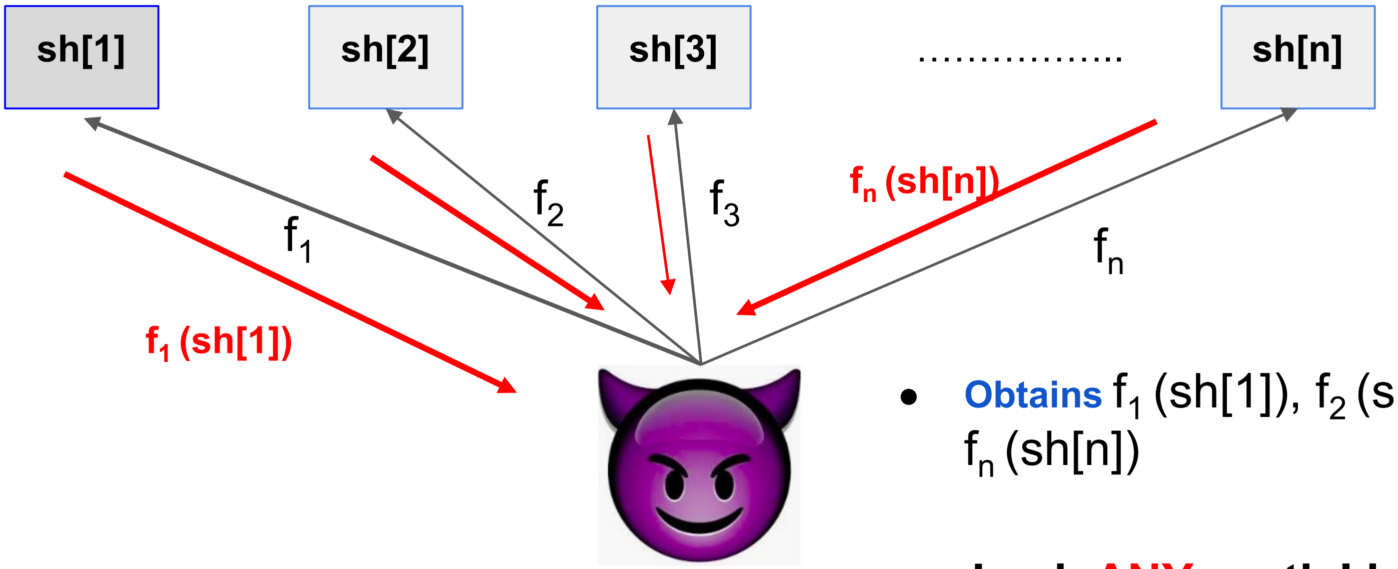
Twist in the story (Introducing leakage)



- **Queries with leakage functions** $f_1, f_2, \ldots f_n$

- **Leak ANY partial information**
- **Output of each f_i is SMALL**

Twist in the story (Introducing leakage)



- **Obtains** f_1 (sh[1]), f_2 (sh[2]) , … f_n (sh[n])

- **Leak ANY partial information**
- **Output of each f_i is SMALL**

Is this model of (LOCAL) leakage reasonable?

- Physical Separation of servers where the shares are stored

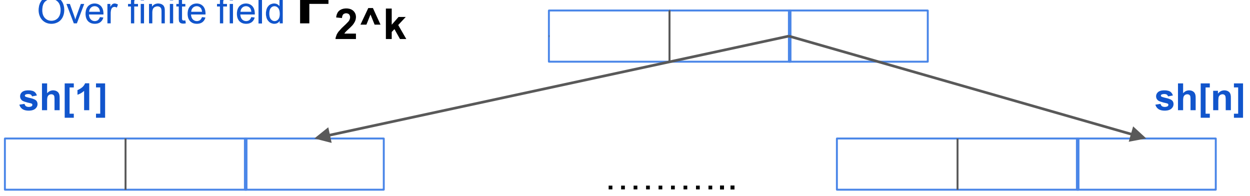
- Shrinked output of leakage

- Adversarial leakage i.e. the adversary gets to choose the leakage functions independent of each other

Shamir scheme not leakage resilient [BDS+18]

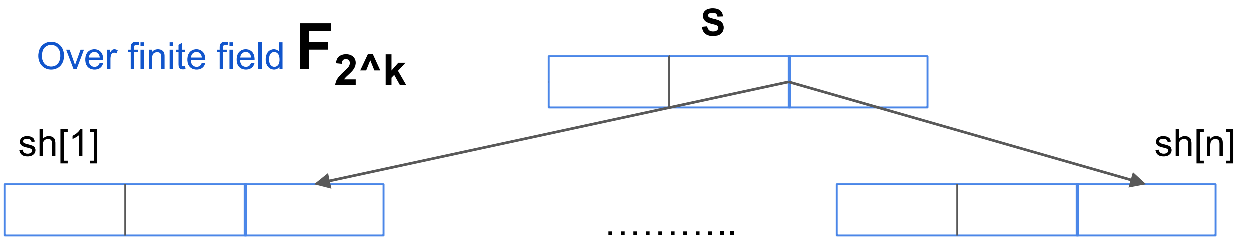
Over finite field $\mathbf{F}_{2^\wedge k}$

s



Shamir scheme not leakage resilient

Over finite field $\mathbf{F}_{2^k}$

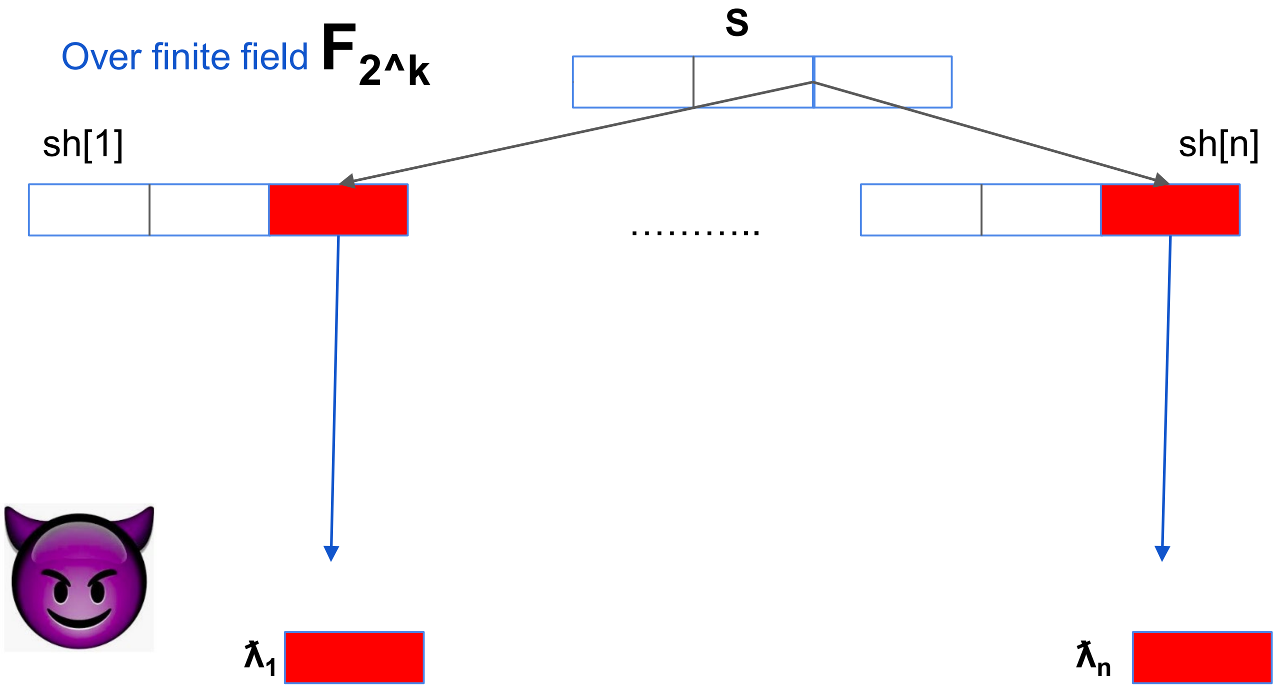


Lagrange interpolation for recovery

$$\mathbf{S = \lambda_1 sh[1] + \ldots.. + \lambda_n sh[n]}$$

Shamir scheme not leakage resilient

Over finite field $\mathbf{F}_{2^\wedge k}$



Lagrange interpolation

$S = \lambda_1 sh[1] + \ldots.. + \lambda_n sh[n]$

Modelling the leakage

- **Local / Independent leakage [**GK 2018, BDS+ 2018, SV 2019**]**
- **Semi-local leakage [**SV 2019**]**
- **Adaptive leakage [**KMS 2019**]**

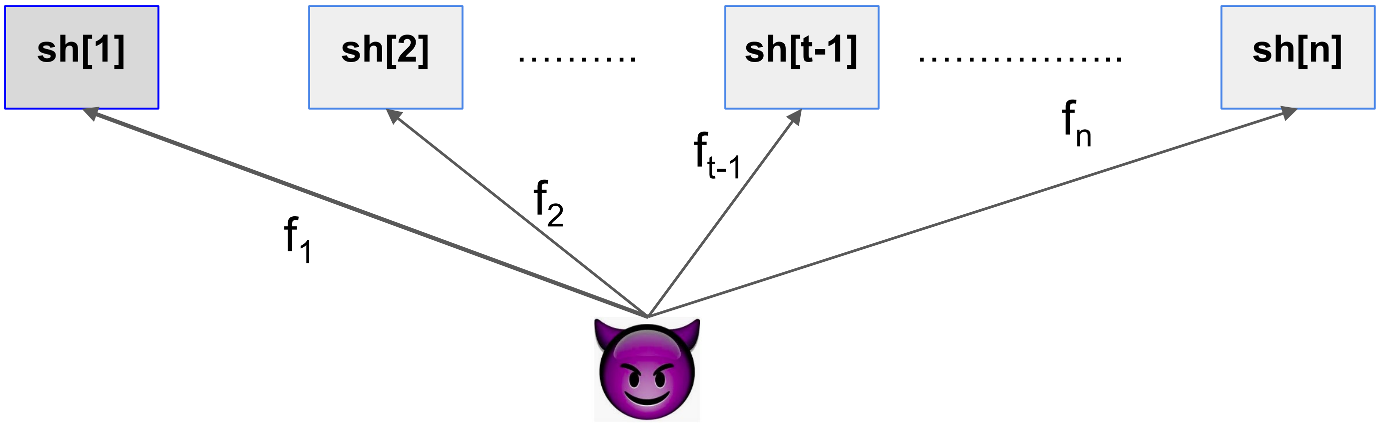
Stronger models of leakage

In this talk

- **Local / Independent leakage [GK 2018, BDS+ 2018, SV 2019] ✓**
- **Semi-local leakage [SV 2019] X**
- **~~Adaptive leakage [KMS 2019]~~ X**

~~Stronger models of leakage~~

Two models of local leakage for (t,n)-SS



- [BDS+18] Weak : each leakage ≠ share (length of each leakage is l bits)
- **[SV'19]** Strong : any t-1 full shares + individual leakage from the rest n-t+1 ✓

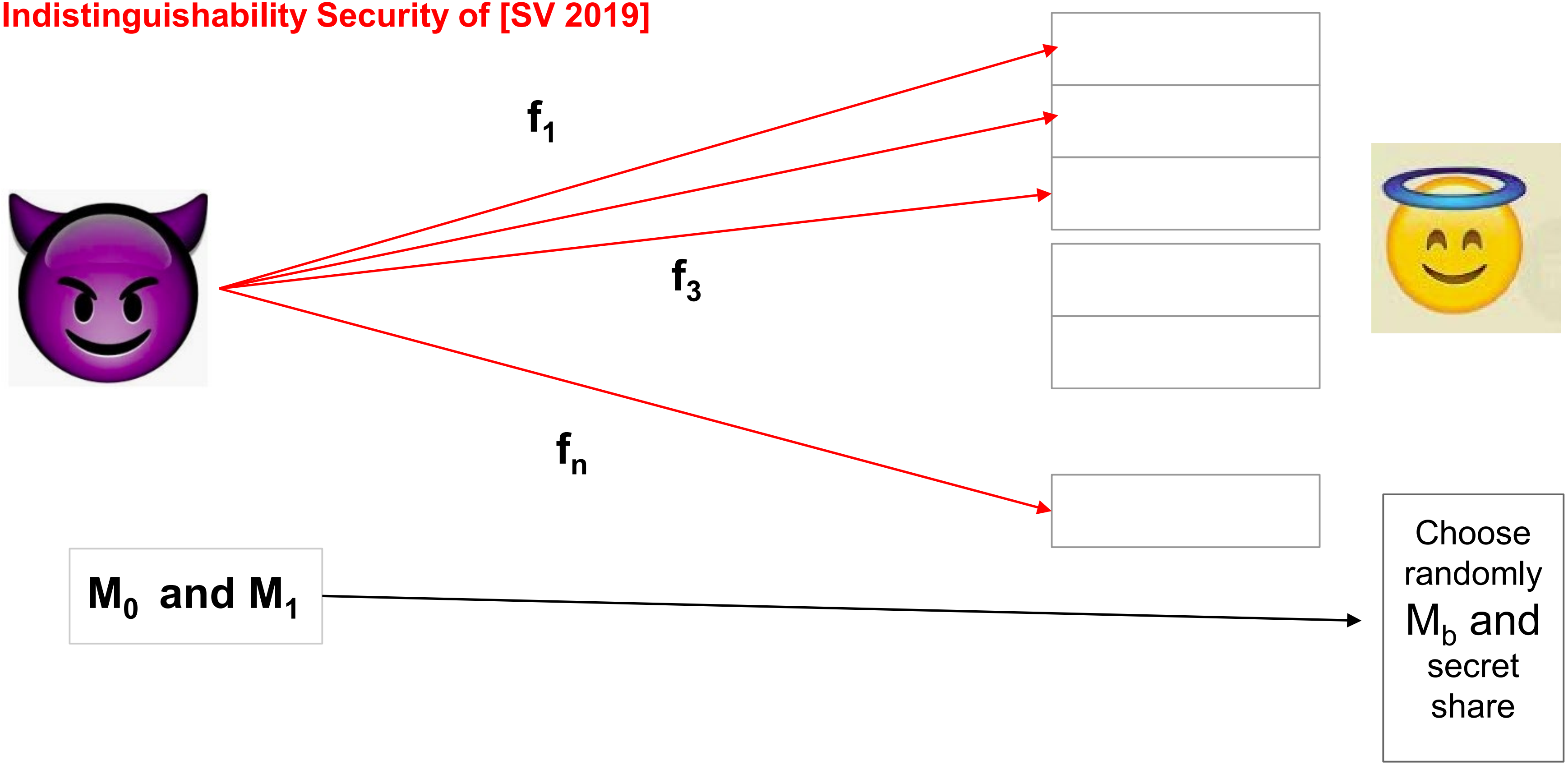
Results with respect to Local Leakage

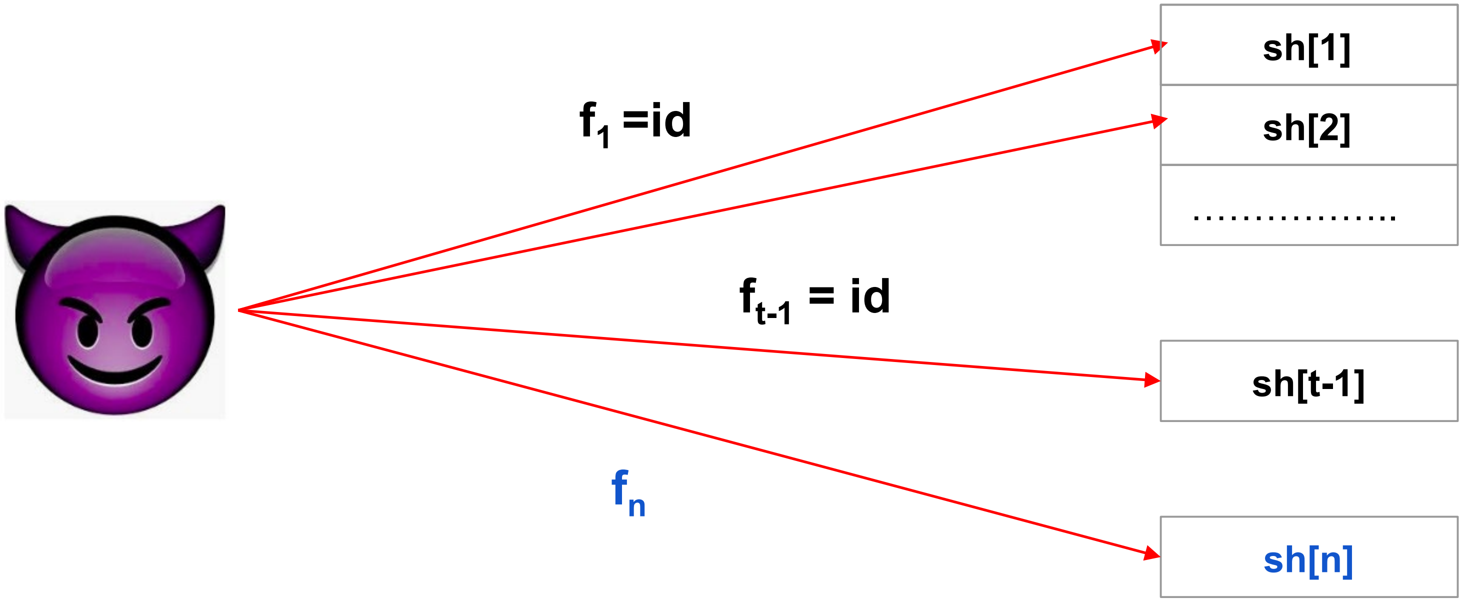
- **Benhamouda et al. 2018** :
  - ➢ Shamir scheme is LR if field is of size large prime p
  - ➢ Threshold is high **n - o(log n)** (>0.85n)
  - ➢ Leakage bound **Ω (log p)** bits

- **Srinivasan-Vasudevan 2019**:
  - ➢ Compiler to make (t,n) Shamir scheme leakage resilient where t > 1
  - ➢ Uses average case strong seeded Extractor

Security against passive adversary (who follows protocol)

Indistinguishability Security of [SV 2019]





f1 =id
ft-1 = id
fn
sh[1]
sh[2]
....
sh[t-1]
sh[n]

Srinivasan-Vasudevan 2019

sh[1]
sh[2]
....
sh[t-1]
Leak (sh[t])
Leak (sh[n])

b’

With this view unable to guess !!!

$$\Pr[b' = b] \approx 1/2$$

Srinivasan-Vasudevan 2019

With this view unable to guess !!!

- The secret is (statistically) hidden even when the adversary has leakage information from all shares

- View of Adv. when M_0 is secret shared ≈ View of Adv. when M_1 is secret shared

Overview of SV’19 construction : Secure against passive adversary

sh[1]

sh[2]

m

Shamir Share

sh[n]

Overview of SV'19 construction : Secure against passive adversary

m

sh[1] $\oplus$ Ext (w_1, s) $\oplus$ r

sh[2] $\oplus$ Ext (w_2, s) $\oplus$ r

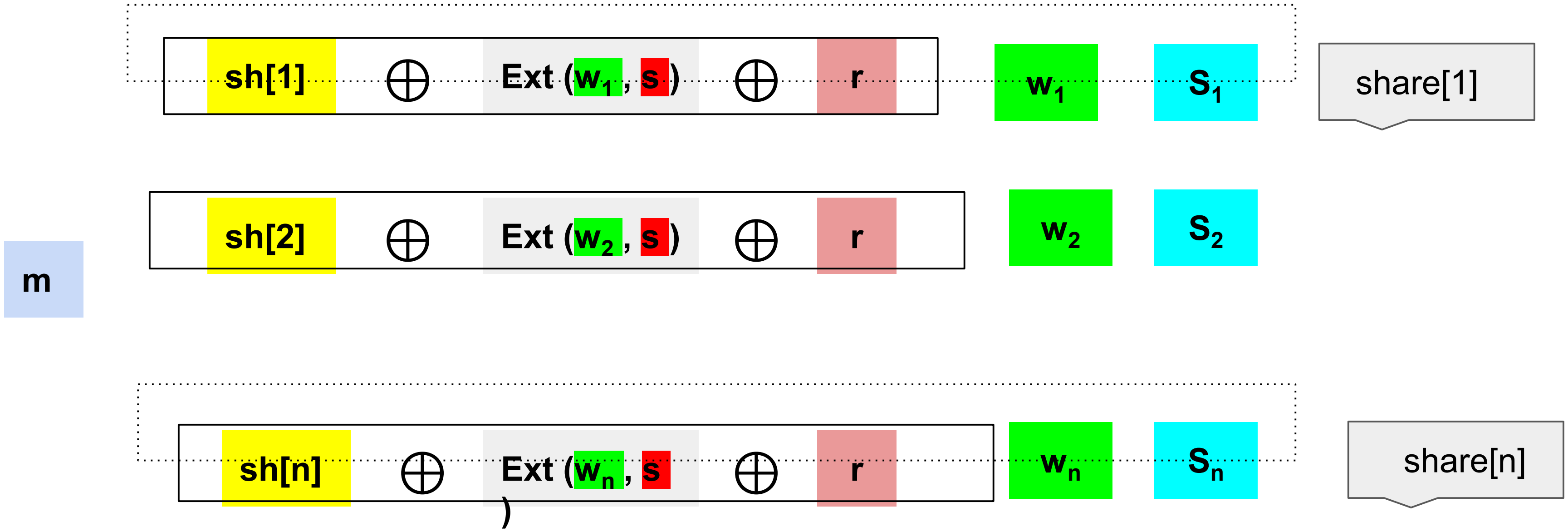
sh[n] $\oplus$ Ext (w_n, s) $\oplus$ r

Run (2,n) Shamir SS :

$(s, r) \longrightarrow$

$(S_1, S_2, \ldots\ldots\ldots, S_n)$

Overview of SV'19 construction : Secure against passive adversary



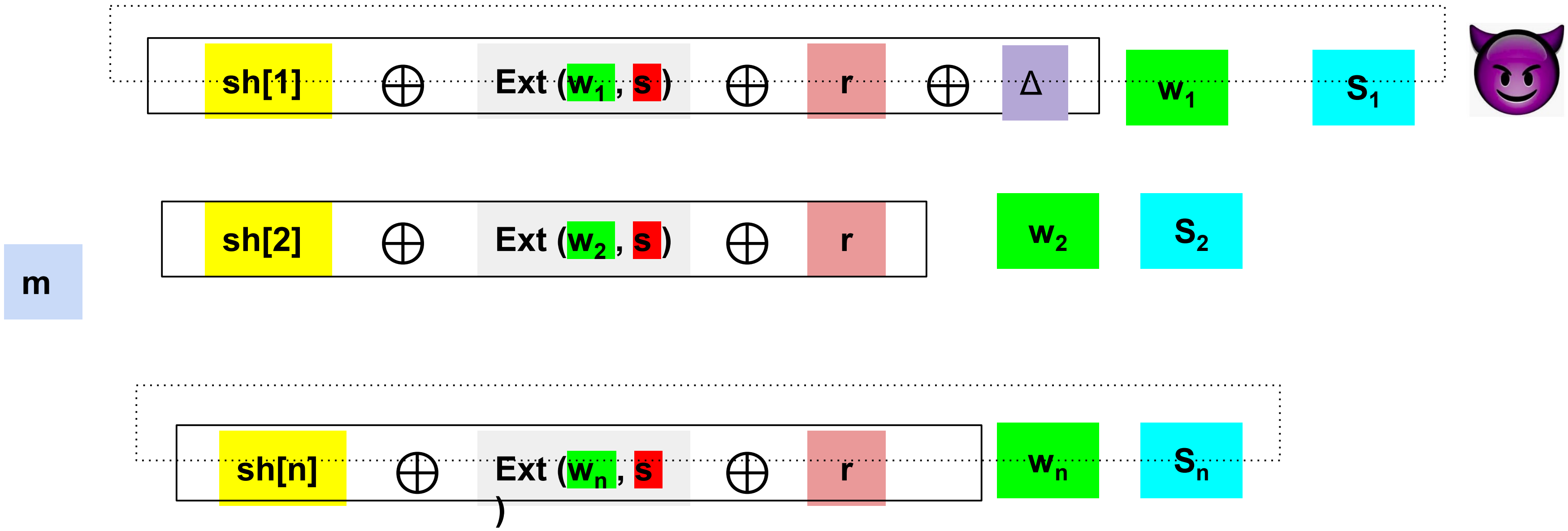
Reconstruction

- Rec s and r from S_i's

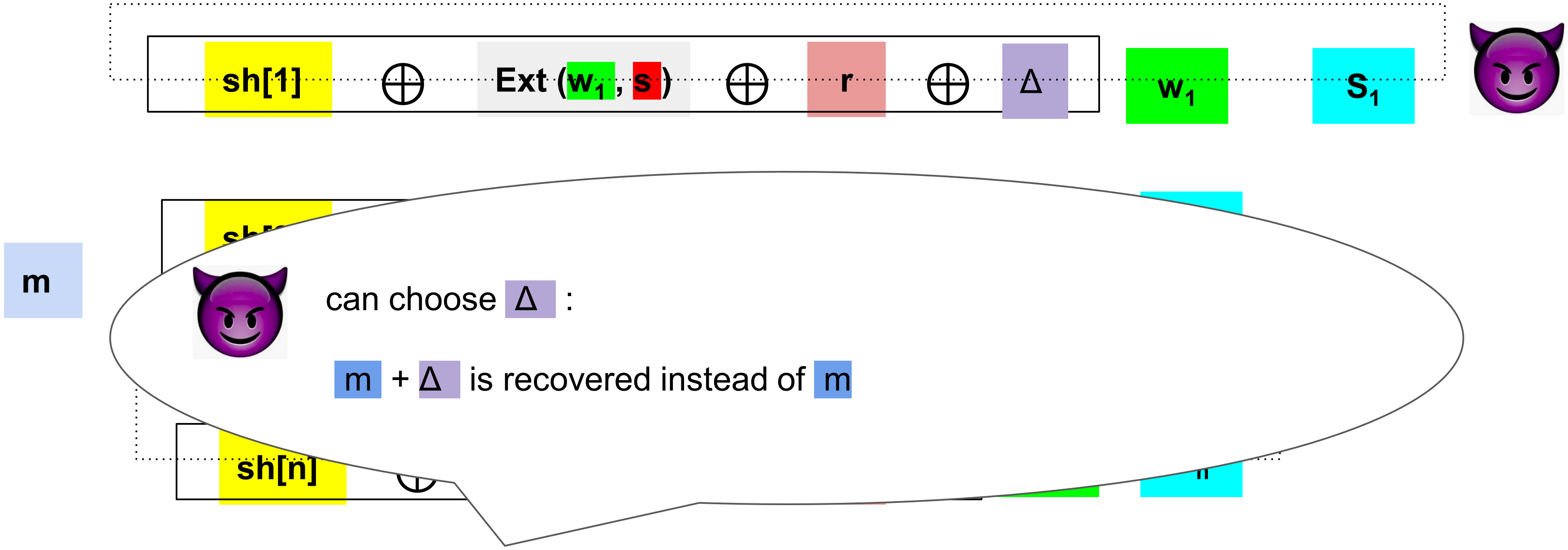
- Remove masking to obtain Shamir shares sh[i_1] sh[i_2] sh[i_t]

Reconstruct **m** = **λ_1 sh[i_1]** + + **λ_t sh[i_t]**

[SV'19] construction : Active adversary attacks !!



Overview of SV’19 construction : Fails against Active adversary



LRSS Schemes secure against active

- Existing LR SS constructions provide security against passive adversary
- We consider

➔ Can LRSS provide security against active attacks?

➔ Honest parties can detect that recovered secret is not correct

➔ This is the minimum requirement of security against active attacks

➔ Known as Cheating Detection

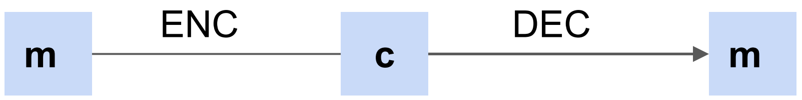
Stronger requirements : cheater identification, robustness etc.

Building Blocks

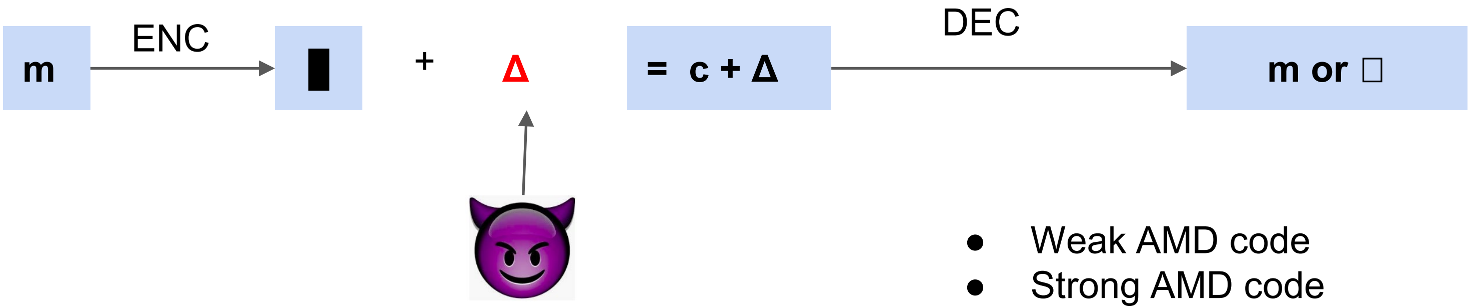
- Leakage-resilient Algebraic manipulation detection (AMD) codes
- LRSS of [SV'19]

AMD codes [CDF+2008]

AMD code = (ENC, DEC)



Security:



- Weak AMD code
- Strong AMD code

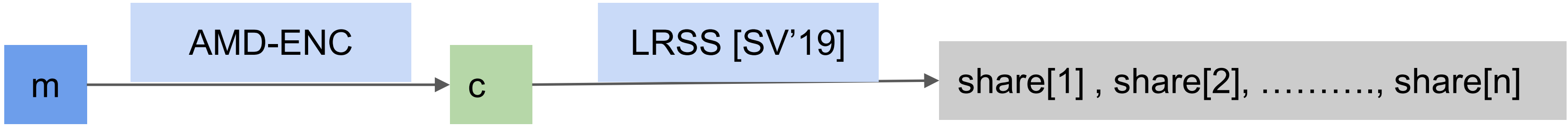
Initial idea:

We want :

1. Our scheme should be Leakage resilient

2. Any active attack should be detected i.e. either recover **m** or recover □

- How about?



- LRSS guarantees leakage resilience
- AMD-DEC detects any additive tampering

- Rec of [SV'19] is a linear sum λ_1 **sh[1]** + λ_2 **sh[2]** + λ_t **sh[t]**

of Shamir shares $\Rightarrow$ either c is obtained or c + Δ is obtained

- AMD-DEC can now output either m or □
- Just a small glitch :

AMD provides security if 😈 does not see c

However, LRSS reveals some leakage information on c

Requirement : Leakage resilient AMD code

Good news : [Ahmadi, Safavi-Naini'13], [Lin,S-N,Wang'16], [Aggarwal, Kazana, Obremski'18] studied LR-AMD codes

- The leakage from AMD codes is measured through leakage **rate** ρ = ratio of AMD codeword symbols (bits) that are leaked to the adversary

- LR-AMD codes guarantee security when c is partially leaked to the adversary but the entropy conditioned on the leakage information remain high

Main Challenge

- How to relate :
- ➔ leakage rate ρ of LR-AMD codes and
- ➔ privacy error / leakage on secret message ε of LR-Secret Sharing

We use average guessing probability

GP(C | **Leak from LRSS**) = 2^ {-- H_∞ (**C** | **Leak from LRSS**) }

to bound the leakage - rate ρ of AMD code given **Leak from LRSS**

Our results

- Compiler for cheating detectable LRSS in local leakage model

    - (**OKS** model of cheating) : LR-weak AMD Code + [SV’19] compiler
    - (**CDV** model of cheating) : LR-strong AMD Code + [SV’19] compiler

❑ Leakage-resilience rate is 1 (same as [SV’19] compiler)
❑ Information rate is 2 times the rate of [SV’19]

➔ Extension to semi-local leakage model : (**OKS** & **CDV** models of cheating)

धन्यवाद

多謝

ขอบคุณ

Спасибо

Gracias

شكراً

Obrigado

Thank You

Diolch

Grazie

Danke

Merci

நன்றி

多谢

감사합니다

ありがとうございました