How to Share a Secret: New Perspectives

Moni Naor

Weizmann Institute of Science

Joint works with Ilan Komargodski and Eylon Yogev

Workshop on Mathematics of Information-Theoretic Cryptography
September 26th 2016

What is Cryptography?

Traditionally: how to maintain secrecy in communication

Alice and Bob talk while Eve tries to listen

Alice

Bob

Eve

History of Cryptography

- Very ancient occupation

  Biblical times, Jeremiah-

Sheshakh has been captured,
the pride of the whole earth seized!
Bavel has become an object of horror
throughout the nations!

א

אֵיךְ נִלְכְּדָה שֵׁשַׁךְ, וַתִּתָּפֵשׂ תְּהִלַּת כָּל-הָאָרֶץ;
אֵיךְ הָיְתָה לְשַׁמָּה בָּבֶל, בַּגּוֹיִם.

Atbash אתבש

Atbash אתבש

- Egyptian Hieroglyphs
  - Unusual ones

...

- Many interesting books and sources,

especially about the Enigma (WW2)

Modern Times

- Up to the mid 70's - mostly classified military work
  - Exceptions: Kerckhoffs, Shannon, Turing
- Since then - explosive growth
  - Commercial applications
  - Scientific work: tight relationship with Computational Complexity Theory
  - Major works: Diffie-Hellman, Rivest, Shamir and Adleman (RSA)
- Recently - more involved models for more diverse tasks.

How to maintain the secrecy, integrity and functionality in computer and communication system.

Emphasis on cooperation: how can parties with only limited trust perform tasks to the benefit of all

Secret Sharing

- Protecting Bitcoin keys
  - Print and destroy?
  - What if the printout is lsot or corrupted (humidity..)
- Be able to retrieve even if some are lost

Idea: split secret into several shares

Want:

- To be able to withdraw money
- But only if enough share owners cooperate!

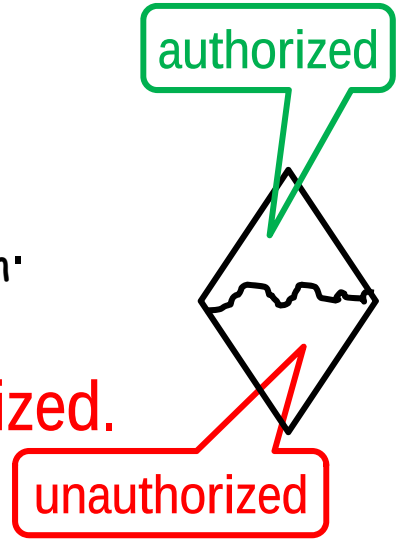
Secret Sharing

The Characters

- Set of users $P_1, P_2, \ldots, P_n$
- Dealer has secret S.

Dealer:

- Gives to users $P_1, P_2, \ldots, P_n$ shares $\Pi_1, \Pi_2, \ldots, \Pi_n$.
  - The shares are a probabilistic function of S.
- A subset of users X is either authorized or unauthorized.

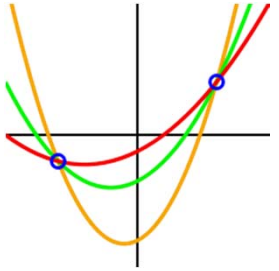


$\Pi(X,S)$

Goal:

- An authorized X can reconstruct S based on their shares.
- An unauthorized X cannot gain *any* knowledge about S.

- Introduced by Blakley and Shamir in the late 1970s.
  - Threshold secret sharing



Source: Wikipedia

"How to share a secret"

Example: Threshold

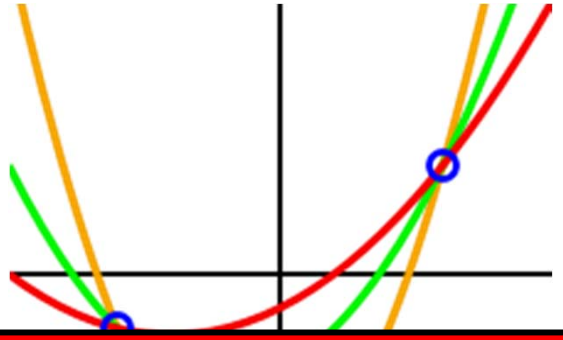
- Shamir's famous example - Threshold Secret Sharing
  - Authorized: any k out of the n parties.
  - Unauthorized: any set of less than k parties.

Solution:

- Fix prime (power) $Q \geq n+1$

Need to know n in advance!

- Choose a random degree $k-1$ polynomial p over $GF[Q]$ s.t.:
  - $p(0) = S$.
  - Let $\Pi_i = p(i)$
- Reconstruction:
  - polynomial interpolation



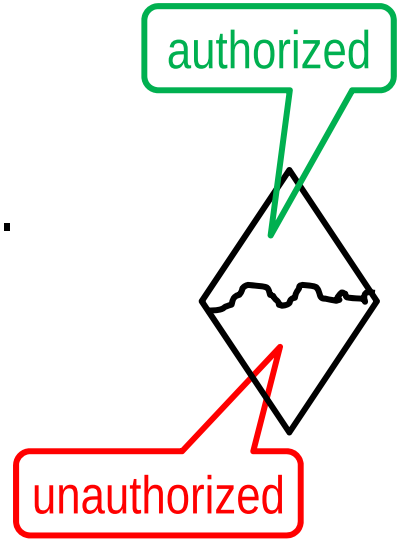
Share size:
$\log Q = \log n$

This is tight: even for single bit secrets
Kilian-Nisan 90,
CascudoCramerXing13

A Few Applications of Secret Sharing

- Protecting the keys of root DNS
- Anonymous petitions:
  - Prove that many members of a group have signed a petition without telling who
- Foundations of Multi-Party Computation
- Electronic Voting

Access Structures



Access Structure M:

- An **indicator function** of the authorized subsets.

- To make sense: M should be monotone:
  if $X' \subset X$ and $M(X')=1$ then $M(X)=1$

Perfect secret sharing scheme:

- For any two secrets S_0, S_1, subset X s.t. $M(X)=0$:

$$Dist(\Pi(X,S_0)) = Dist(\Pi(X,S_1)).$$

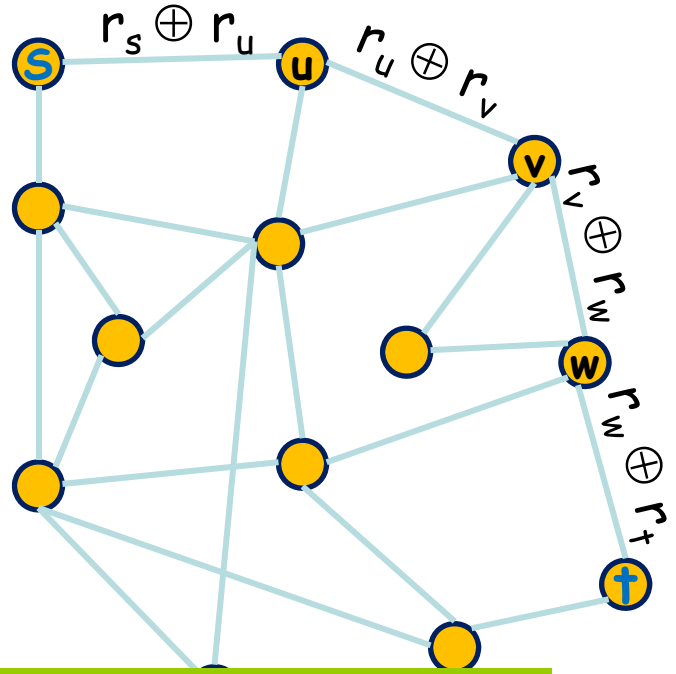
Or equivalently: for any distinguisher A:

$$|Pr[A(\Pi(X,S_0)) = 1] - Pr[A(\Pi(X,S_1)) = 1]|=0$$

The **complexity** of the scheme: the **size** of the largest share.

Example: undirected connectivity

- Parties correspond to edges in a graph G.
- Two special nodes: s and t.
- Authorized sets: those graphs containing a **path from s to t**.
- Solution:
  - Give vertices random values $r_1,\ldots,r_n$.
  - Set $r_t = S \oplus r_s$.
  - For edge $\Pi_{u,v} = r_u \oplus r_v$.
- Reconstruction:
  - XOR all shares.



What about directed connectivity?

Known Results

Theorem [Ito, Saito and Nishizeki 1987] : All access structures.

For every $\mathcal{M}$ there exists a perfect secret sharing scheme

- might have exponential size shares in the number of parties
- proportional to CNF / DNF

Exponential lower bounds!
Cook, Pitassi, Robere and Rossman,2016

Theorem [Benaloh-Leichter 1988] :

If $\mathcal{M}$ is a monotone formula Φ: there is a perfect secret sharing scheme where the size of a share is proportional to $|\Phi|$.

Karchmer-Wigderson generalized this results to monotone span programs [1993]

Major question: can we prove a lower bound on the size of the shares for *some* access structure?

- Even a non constructive result is interesting

This talk

Two new aspects of secret sharing:

- Evolving secret Sharing
- Secret Sharing for NP

Evolving Secret Sharing

- Can we design scalable systems without suffering a great deal of efficiency costs?

This talk: no **fixed upper bound** on the number of participants in the area of **secret sharing**.

Important even if there is an upper bound but

do not want to waste the max if fewer people show up

Can we not assume upper bounds?

Examples

- Prefix codes of integers [Elias75,Dodis-Patrascu-Thorup10]
- Locally Labeling infinite graphs for adjacency [Kannan-Naor-Rudich92]
- Bloom filters of a growing set [Pagh-Segev-Wieder 13]
- Secret sharing [CsirmazTardos12,This Work]

Secret Sharing for Evolving Access Structures

- Parties come **one-by-one**
- Qualified sets are revealed when **all** their members are present.
- Upper bound on # of parties is **unknown**.
- Parties are only **added** and qualified sets **remain qualified**
- Shares are given only to **joining** parties

Cannot refresh shares

Goal:

- Qualified X can reconstruct S based on their shares.
- Unqualified X cannot distinguish S from random.

1. Can this even be done?
2. How large should the shares be?

History

- Christian Cachin. On-line secret sharing, 1995.
- Laszlo Csirmaz and Gabor Tardos. On-line secret sharing. 2012.

Our Results

- Constructions

Evolving access structure	Share size of t^{th} party
General	2^{t-1}
$\mathbf{k}$-threshold	$(\boldsymbol{k}-1)\log t + \boldsymbol{k}^3 \cdot o(\log t)$
$\mathbf{2}$-threshold	$\log t + \text{loglog}\, t + 2\text{logloglog}\, t$
s-t-connectivity	1

- Lower bound
  - Any scheme for $\mathbf{2}$-threshold requires $\log t + \text{loglog}\, t + \text{logloglog}\, t$ bits.

- Equivalence to prefix codes

  Exists a scheme for $\mathbf{2}$-threshold with share size $\boldsymbol{\sigma(t)}$ for single bit secret if and only if there exists a prefix code for the integers where the length of the t-th codeword is $\boldsymbol{\sigma(t)}$ exists.

Talk Plan and Techniques

- Equivalence of schemes for 2-threshold and prefix codes
- A construction for 2-threshold
- Generalization for k-threshold

Domain Reduction
Formula for the future

- Construction for general evolving access structures

Prefix (free) code

Prefix code:

An encoding of $\mathbb{N}$ s.t. no codeword is a prefix of any other

Elias code is a prefix code with t-th codeword length

$$\sigma(t) = \log t + \text{ loglog } t + \text{logloglog } t + \cdots$$

Theorem

- Existence of prefix code $\Sigma: \mathbb{N} \to \{0,1\}^*$
  - codeword length $\boldsymbol{\sigma(t)} = |\Sigma(t)|$

Equivalent to

- Existence of 2-threshold scheme for single bit secrets
  - share of player t of length $\mathbf{m(t)} = \boldsymbol{\sigma(t)}$

Evolving 2-Threshold from Prefix Codes

Single bit secret s

- Evolving 2-threshold from prefix codes:
  - Let $\Sigma: \mathbb{N} \to \{0,1\}^*$ be a prefix code
    - codeword **length** $\sigma(t)$

  Dealer: choose an evolving **random** string $w \in \{0,1\}^*$

  - If the secret is 0: the share of party t is: $w[1:\sigma(t)]$
  - If the secret is 1: the share of party t is $\Sigma(t) \oplus w[1:\sigma(t)]$

  Reconstruction: if one share is a prefix of the other

  - the secret is 0
  - otherwise, the secret is 1

Prefix of length $\sigma(t)$ of w

Share size: $\sigma(t) = \log t + \text{loglog}\, t + \text{logloglog}\, t$ ...

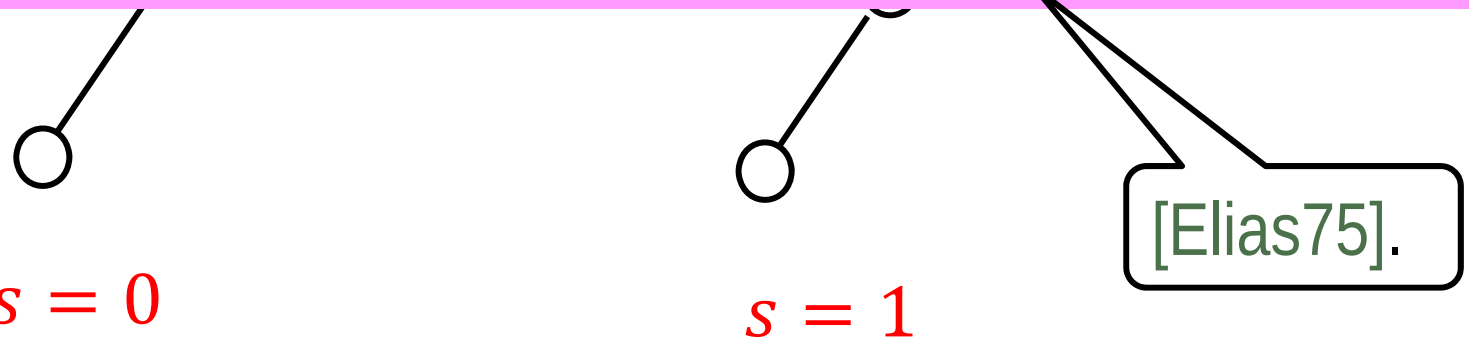
Evolving 2-Threshold from Prefix Codes

Correctness: shift by string w of prefix code yields a prefix code

Security: each party t gets a random string of length $\sigma(t)$

Cannot deduce anything on its own

Share size: $\sigma(t) = \log t + \text{loglog}\, t + \text{logloglog}\, t \cdots$



Lower bound for Evolving 2-Threshold

Claim: ([KN90] and [CCX13, Appendix A]).

For any $n \in \mathbb{N}$ it holds that

Kraft's Ineq.

$$\sum_{t=1}^{n} 1/2^{m_t} \leq 1$$

m_t length of share t

Let $s_{t,b}$ be R.V. distributed as the share of party t when the secret is b
By **secrecy**: for single players for every t: $s_{t,0}$ and $s_{t,1}$ identically dist.

correctness

$$\Pr[s_{t,0} = s_{t,1}] \geq \frac{1}{2^{m_t}}$$

Every pair of parties determines the secret: **equality events are disjoint**

$$1 \geq \Pr[\exists i: s_{i,0} = s_{i,1}] = \sum_{t=1}^{n} \Pr[s_{t,0} = s_{t,1}] \geq \sum_{t=1}^{n} \frac{1}{2^{m_t}}$$

2-Threshold Evolving Access Structures

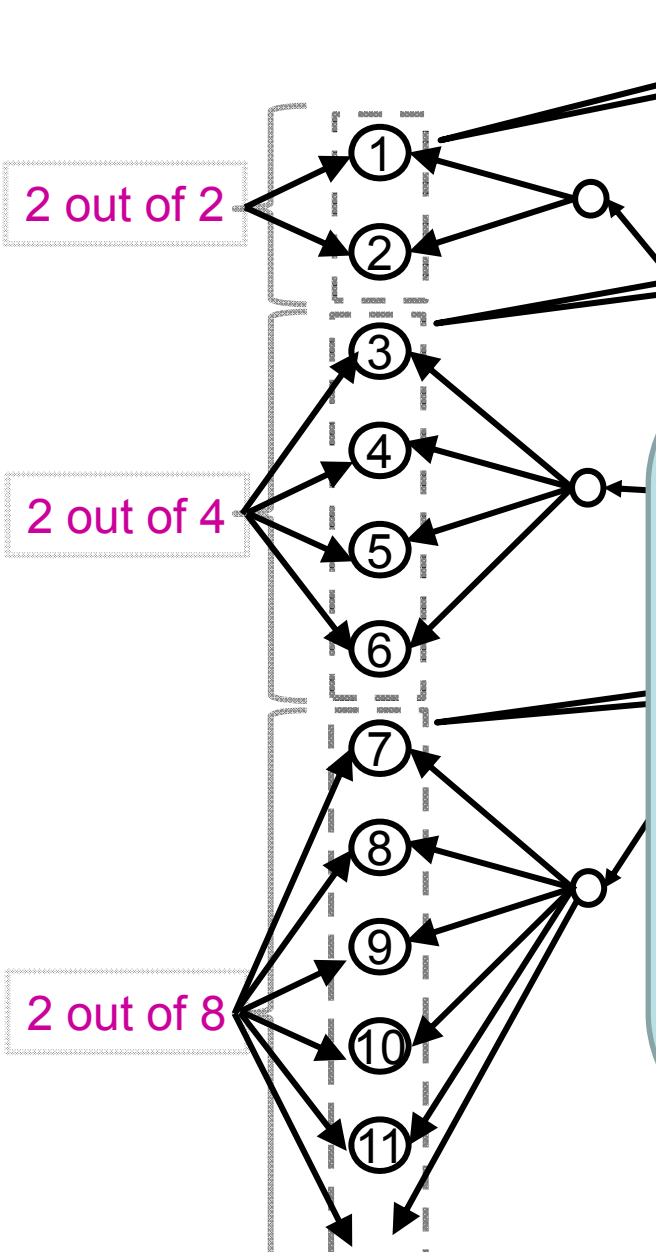
- Basic scheme Π^0 [CsirmazTardos12]
  - Party 1 is given a random bit b_1
  - Party 2 is given a random bit b_2 and the bit $b_1' = b_1 \oplus s$
  - Party 3 is given a random bit b_3 and the bits b_1' and $b_2' = b_2 \oplus s$
  - …
  - Party t is given a random bit b_t and bits $b_1', \ldots, b_{t-1}'$ where $b_i' = b_i \oplus s$
- Every pair $i < j$ can compute s

$$b_i \oplus b_i' = b_i \oplus (b_i \oplus s) = s$$

From player i From player j

- Each single player has no information about s
- Total share size: $\sigma^0(t) = t$ bits.

Domain Reduction for 2-Threshold



secret s

Parties are assigned to generations

t-th share size analysis:

Internal scheme: $\log t$ bits

External scheme: $\sigma^0(\log t)$ bits

Total: $\sigma^1(t) = \log t + \sigma^0(\log t)$

$= \log \mathrm{t} + \log \mathrm{t} = 2\log \mathrm{t}$

Apply recursively to get σ^2 using σ^1

Apply recursively to get σ^3 using σ^2

$\sigma^3(t) = \log t + \mathrm{loglog}\, t + 2\mathrm{logloglog}\, t$

Comparison with Prefix Codes Based Scheme

- Both schemes result with same share size
- The scheme based on prefix codes is non-linear while the direct scheme is.
- Direct scheme is more efficient w.r.t longer secrets.
- The direct scheme is used as a basis for the generalization for larger thresholds

k-Threshold Domain Reduction

- Start with a basic solution

Which one? next slide

- Partition parties into generations, where the generations are **geometrically increasing** in size
- Within a generation use a **Shamir** k-threshold scheme
  - Handles case where **all** k parties come from the same generation

As before

As before

- Generate $k-1$ shares of the basic scheme. **Split** them between the parties such that any $i < k$ parties can learn i of these shares.
  - Altogether k players will hold k shares of the basic scheme
  - Done by sharing share π_i using a Shamir i-threshold scheme

$\pi_1, \ldots, \pi_{k-1}$

Share Size of t-th Party

- Setting parameters gives total share size roughly
$$\sigma^i(t) = (k-1) \cdot \log t + k \cdot \sigma^{i-1}(\log t + k)$$

- What is σ^0?
  - Using a naïve scheme (with share size 2^t or t^k) results with an **exponential** dependence on k
  - We construct a more efficient scheme with **poly** depdendency

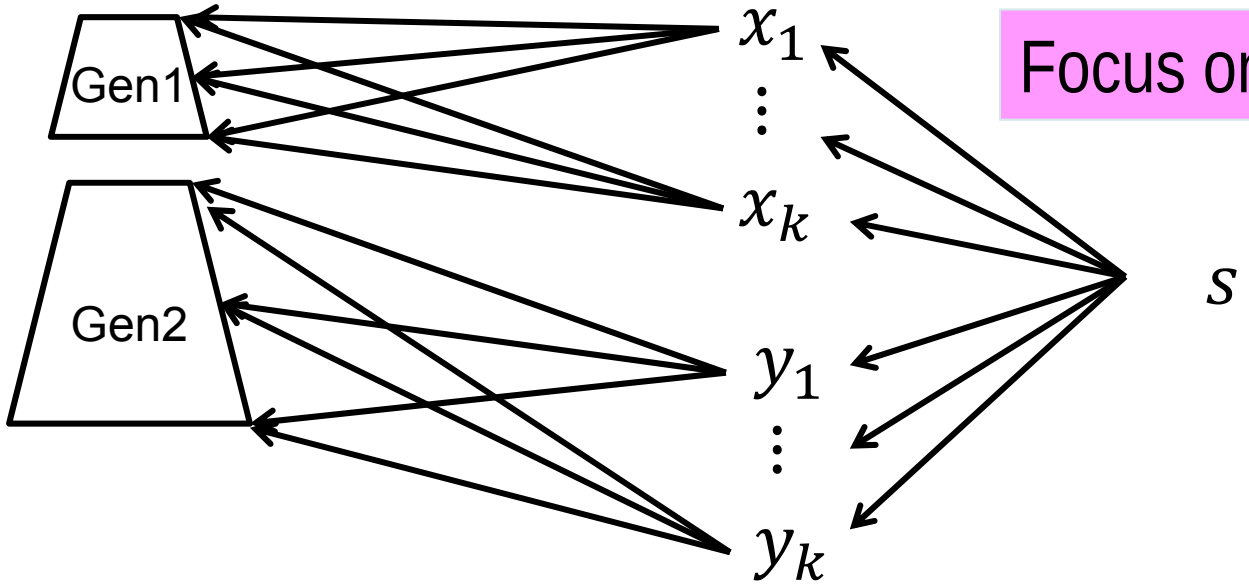
The Basic Scheme for k-Threshold

– At any point: we know the **past** but don't know the **future**.
  • Should prepare for any possible future
– How many different futures can there be?
  • Naively: any set of players potentially defined a different future.

Key idea: we do not care *who* comes from the future but only *how many*.

– Only k relevant options!
– Say ℓ parties came so far. This tells us that if $k - \ell$ parties will come in the future, they should learn the secret
– Do so for every party and for every $\ell \in [k]$.
– Share size is still exponential in k!

– Group parties into generations of geometrically increasing size.
– Only care about **quantity** of parties within a generation



Focus on 2 generations

x_i (**present**) is shared s.t. any i parties from gen 1 can recover it

y_i (**future**) is shared s.t. any i parties from gen 2 can recover it

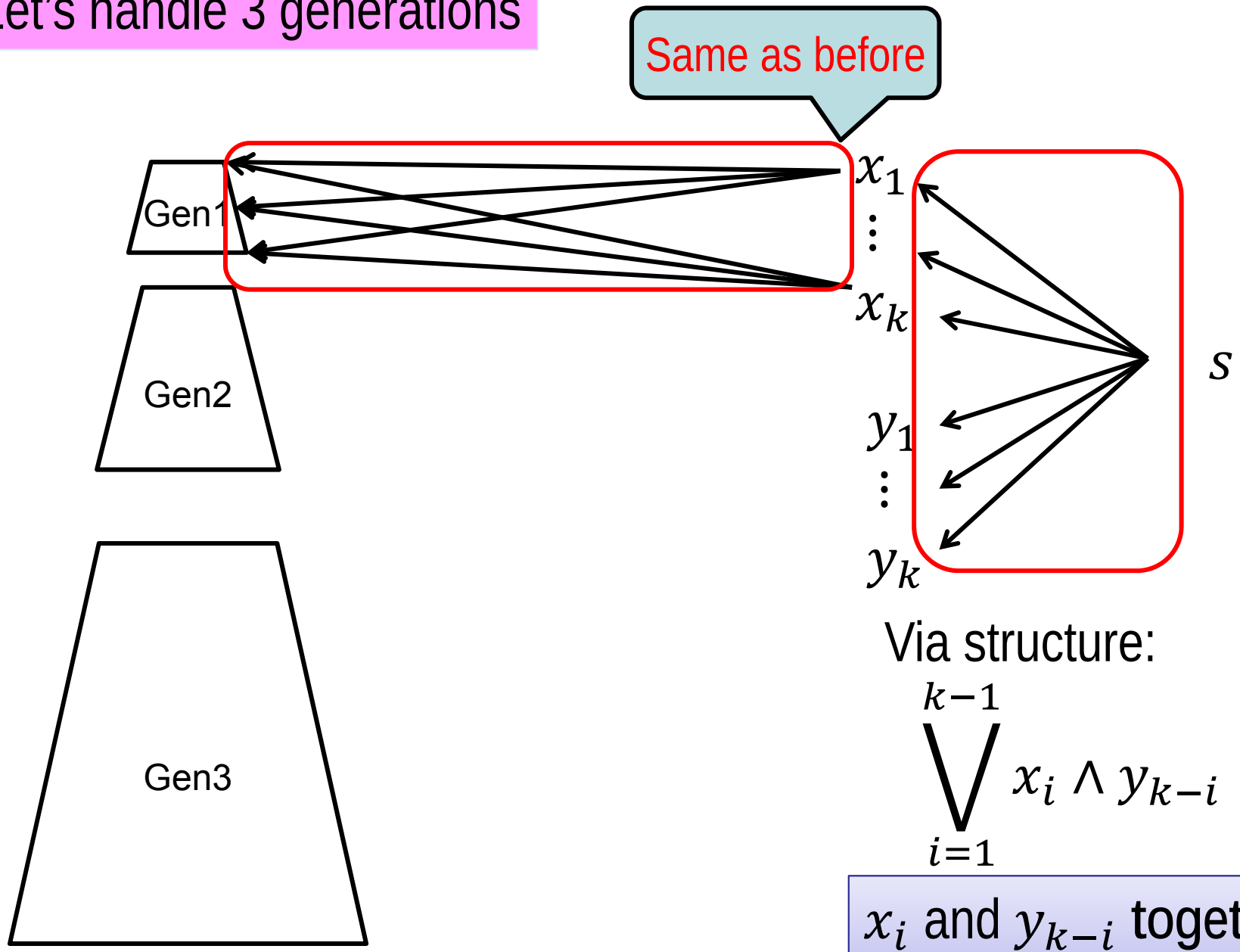
using Shamir's scheme

Via structure:

$$\bigvee_{i=1}^{k-1} x_i \wedge y_{k-i}$$

x_i and y_{k-i} together can recover s

Let's handle 3 generations

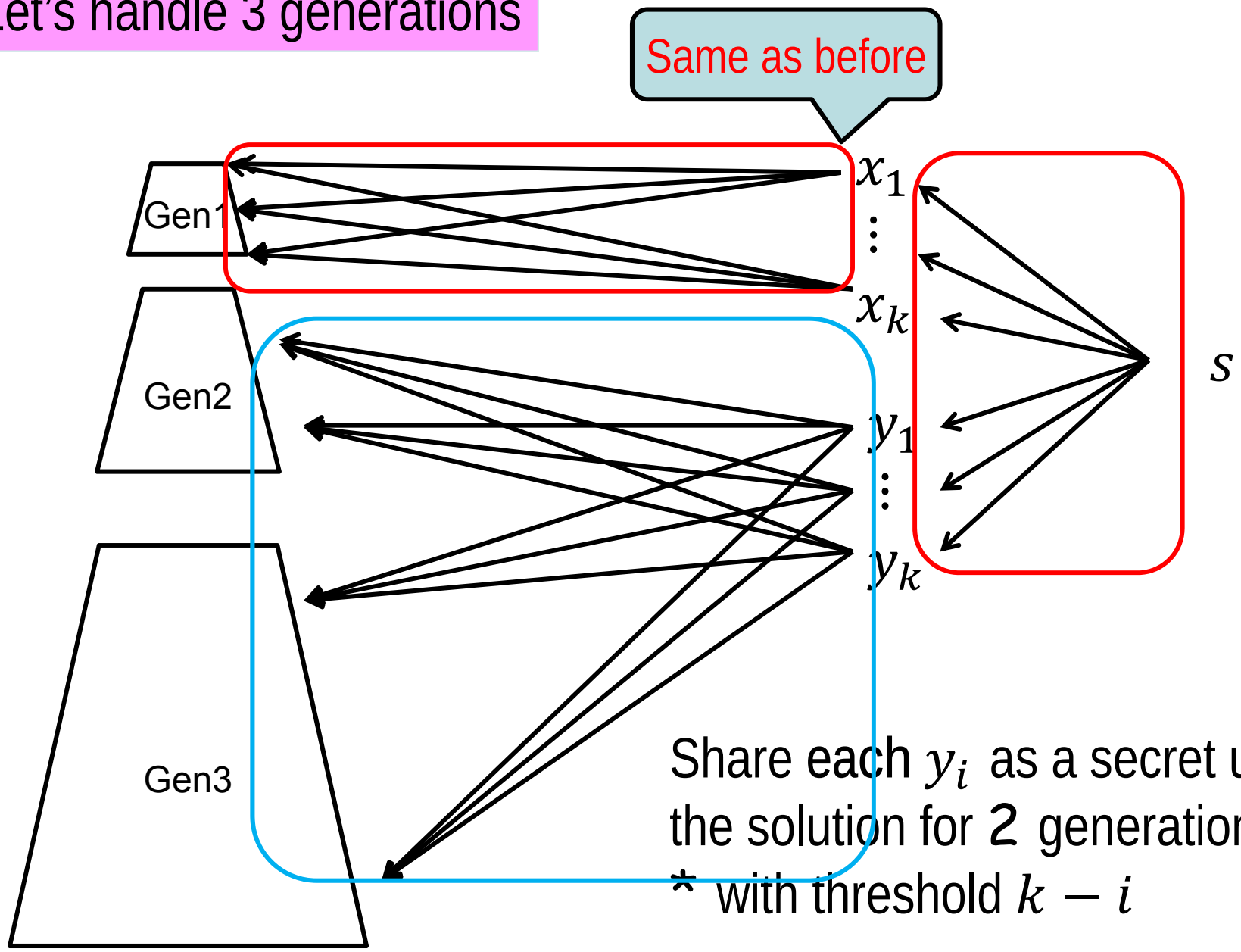


Via structure:

$$\bigvee_{i=1}^{k-1} x_i \wedge y_{k-i}$$

x_i and y_{k-i} together can recover s

Let's handle 3 generations



Share each y_i as a secret using the solution for 2 generations!

* with threshold $k - i$

Share Size Analysis

The share of party t from generation g composed of

- k^{g+1} shares generated via standard threshold schemes over $\text{size}(g)$ parties.

The share size of party t is bounded by

$$k^{g+1} \cdot \log(\text{size}(g)) \ .$$

Set $g = \log_k t$ and $\text{size}(g) = k^{g+1}$.

Therefore, the share size is bounded by

$$kt \cdot log(kt)$$

General Evolving Access Structures

- Scheme for DNFs in the **standard** setting:
  - For a vector **(1,0,1,0,0,1)** representing a qualified set:

  Dealer gives party **1** a random bit r_1, party 3 a random bit r_3, and party 6 the bit $r_1 \oplus r_3 \oplus s$

- In the **evolving** setting:
  - **If** there is an upper bound on the number of qualified sets a party is a member of, we can give the party this number of random bits (one per clause) [CsirmazTardos12]
  - However, we don't have an upper bound…

General Evolving Access Structures

- Party t holds 2^{t-1} bits $w(b_1, \dots, b_{t-1}, 1)$.

- If party t completes a **qualified** set $(b_1, \dots, b_{t-1}, 1)$:

$$w(b_1, \dots, b_{t-1}, 1) = w(b_1, \dots, b_{t-1}) \oplus \cdots \oplus w(b_1) \oplus s$$

$$- w(b_1, \dots, b_i, 0) = 0 \text{ for all } i$$

- Otherwise, it gets a random bit

$$w(b_1, \dots, b_{t-1}, 1) \leftarrow \{0,1\}$$

- **Correctness**: immediate
- **Security**: at least one of the XORed components with the secret must be missi
- Total share size: $\sigma(t) =$

No need to know
the access structure before

General Evolving Access Structures

- If party t completes a **qualified** set $(b_1, \ldots, b_{t-1}, 1)$:

$$w(b_1, \ldots, b_{t-1}, 1) = w(b_1, \ldots, b_{t-1}) \oplus \cdots \oplus w(b_1) \oplus s$$

$$- w(b_1, \ldots, b_i, 0) = 0 \text{ for all } i$$

Good when there are a few unqualified sets

- Otherwise, it gets a random bit

$$w(b_1, \ldots, b_{t-1}, 1) \leftarrow \{0,1\}$$

- Some optimization (if access structure is known)

  Share size: what matters is how many times party t appears in

  – **Unqualified** subsets that can be expanded to qualified

  – **Qualified** subsets where t is the last one in the subset

When applied to **k**-threshold: $\sum_{i=0}^{k-1} \binom{t-1}{i}$

Open Problems

- Lower bound for general access structure
  - Easier than the standard setting
- Can we make the share size **independent** of k?
  - Say $O(\log k + \log t)$
- An efficient scheme for dynamic majority
  - Qualified sets are those that form a majority at *some* point in time
- Applications for MPC?
  - Our schemes are linear which is a critical property for constructions of MPC
- Can we gain from allowing statistical error? computational security?
- Verifiable/robust/visual evolving secret sharing…

For k-threshold

Computational Secret Sharing

- **Perfect** secret sharing scheme:

  Any unauthorized subset X gains absolutely *no* information:

  – For any A, secrets S_0, S_1, subset X s.t. $M(X)=0$:

  $$|Pr[A(\Pi(X,S_0)) = 1]-Pr[A(\Pi(X,S_1)) = 1]|=0.$$

- **Computational** secret sharing scheme:

  Any unauthorized subset X gains no useful information:

  $$\Pi(X,S_0) \approx_c \Pi(X,S_1)$$

  In the **indistinguishability** of encryption style:

  For any PPT A, two secrets S_0, S_1, subset X s.t. $M(X)=0$:

  $$|Pr[A(\Pi(X,S_0)) = 1] - Pr[A(\Pi(X,S_1)) = 1]| < neg$$

This is a non-uniform definition

Computational Secret Sharing

Theorem [Yao~89]:
If $\mathcal{M}$ can be computed by a **monotone** poly-size circuit $\mathcal{C}$ then:

There is a computational secret sharing scheme for $\mathcal{M}$.

- Size of a share is proportional to $|\mathcal{C}|$.
- Assuming one-way functions.

Construction similar to Yao's garbled circuit

- What about monotone access structure that have small non-monotone circuits?
  - Matching:
    - Parties correspond to edges in the complete graph.
    - Authorized sets: the subgraphs containing a **perfect matching**.

Open problem: do all **monotone functions** in $\mathcal{P}$ have computational secret sharing schemes?

Secret Sharing for NP

Rudich circa 1990

What about going beyond P?

- Efficient **verification** when the authorized set **proves that it is indeed** authorized
  - Provides a witness

Example:

- Parties correspond to edges in the **complete graph**.
- Authorized sets: subgraphs containing a **Hamiltonian Cycle**.
- The **reconstruction** algorithm should be provided with the **witness**: a cycle.

Secret Sharing and Oblivious Transfer

Theorem:

If one-way functions exist and a computationally secret sharing scheme for the Hamiltonian problem exists then:

Oblivious Transfer Protocols exist.

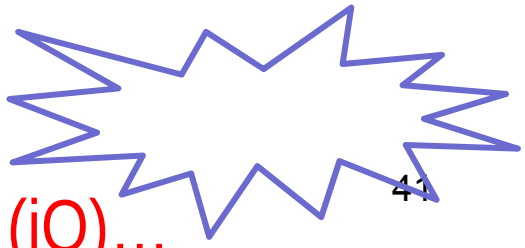
- In particular Minicrypt = Cryptomania
- Construction is non-blackbox

- No hope *under standard assumptions* for perfect or statistical scheme for Hamiltonicity

Witness Encryption [Garg, Gentry, Sahai, Waters 2013]

Includes y

- A witness encryption (Enc_L, Dec_L) for a language $L \in NP$:
  - Encrypt message m relative to string y: $ct = Enc_L(y,m)$
  - For any $y \in L$: let $ct = Enc_L(y,m)$ and let w be any witness for x. Then $Dec_L(ct,w) = m$.
  - For any $y \notin L$: $ct = Enc_L(y,m)$ computationally hides the message m.
    Gave a candidate construction for witness encryption.
- Byproduct: a candidate construction for secret sharing for a specific language in NP (Exact Cover).

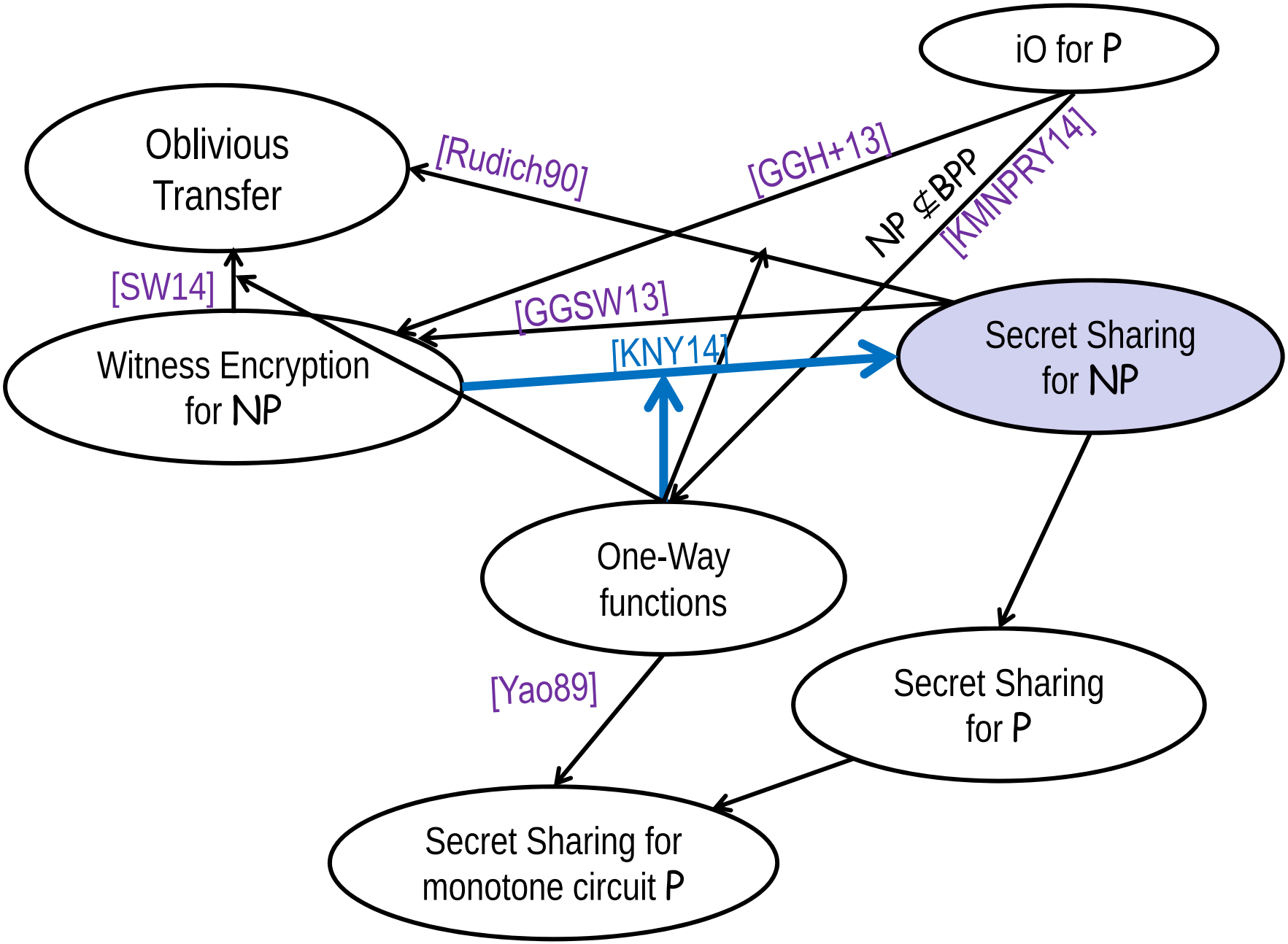


Multilinear Maps, Indistinguishability Obfuscation (iO)…

Our Results

If one-way functions exist then:

- Secret Sharing for NP and **Witness Encryption** for NP are (existentially) equivalent.
- If there is a **secret sharing** scheme for one NP-complete language, then there is one for all languages in NP.


iO for P
Oblivious Transfer
[Rudich90]
[GGH+13]
NP ⊄ BPP
[KMNPRY14]
[SW14]
[GGSW13]
Witness Encryption for NP
[KNY14]
Secret Sharing for NP
One-Way functions
[Yao89]
Secret Sharing for P
Secret Sharing for monotone circuit P

Definition of secret sharing for NP

Let M be a monotone access structure in NP.

- Completeness:
  For any X s.t. $M(X)=1$, any witness w (for X), and any secret S:

$$\text{recon}(\Pi(X,S),w) = S.$$

  – All operations polytime

Definition of secret sharing for NP: Security

- Let M be a monotone access structure in NP.

Security:

For any adversary $A=(A_{samp},A_{dist})$ such that A_{samp} chooses two secrets S_0,S_1 and a subset X it holds that:

$$|Pr[M(X)=0 \wedge A_{dist}(\Pi(S_0,X)) = 1] - Pr[M(X)=0 \wedge A_{dist}(\Pi(S_1,X)) = 1]|$$

$< neg.$

This is a static and uniform definition

The Construction

For access structure $M \in NP$.

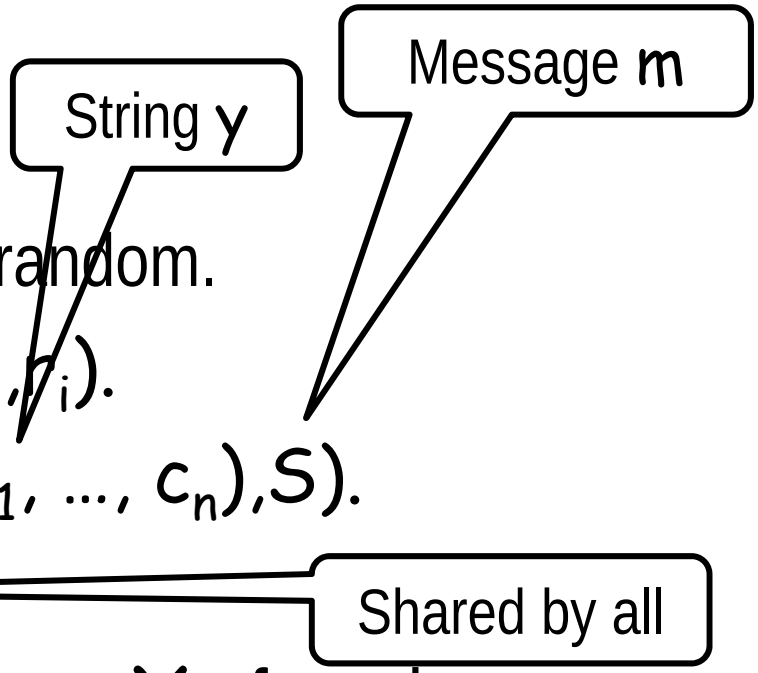
- Define a new language $M' \in NP$:
  - Let $c_1, \ldots, c_n$ be n strings.
  - Then $M'(c_1,\ldots,c_n) = 1$ iff $M(X) = 1$ where:

$$X_i = \begin{cases} 1 \text{ if exist } r_i \text{ s.t. } c_i = com(i, r_i) \\ 0 \text{ otherwise} \end{cases}$$

Computationally hiding: $com(x_1) \approx com(x_2)$
Perfect Binding: $com(x_1)$ and $com(x_2)$ have disjoint support.

Can be constructed from one-way functions in the CRS model with high probability.

The Construction…



Dealer(S):

- Choose $r_1, \ldots, r_n$ uniformly at random.
- For $i \in [n]$, compute $c_i = com(i, r_i)$.
- Compute $ct = WE.Enc_{M'}((c_1, \ldots, c_n), S)$.
- Set $\Pi_i = (r_i, ct)$.

Reconstruction: authorized subset X of parties: $M(X)=1$

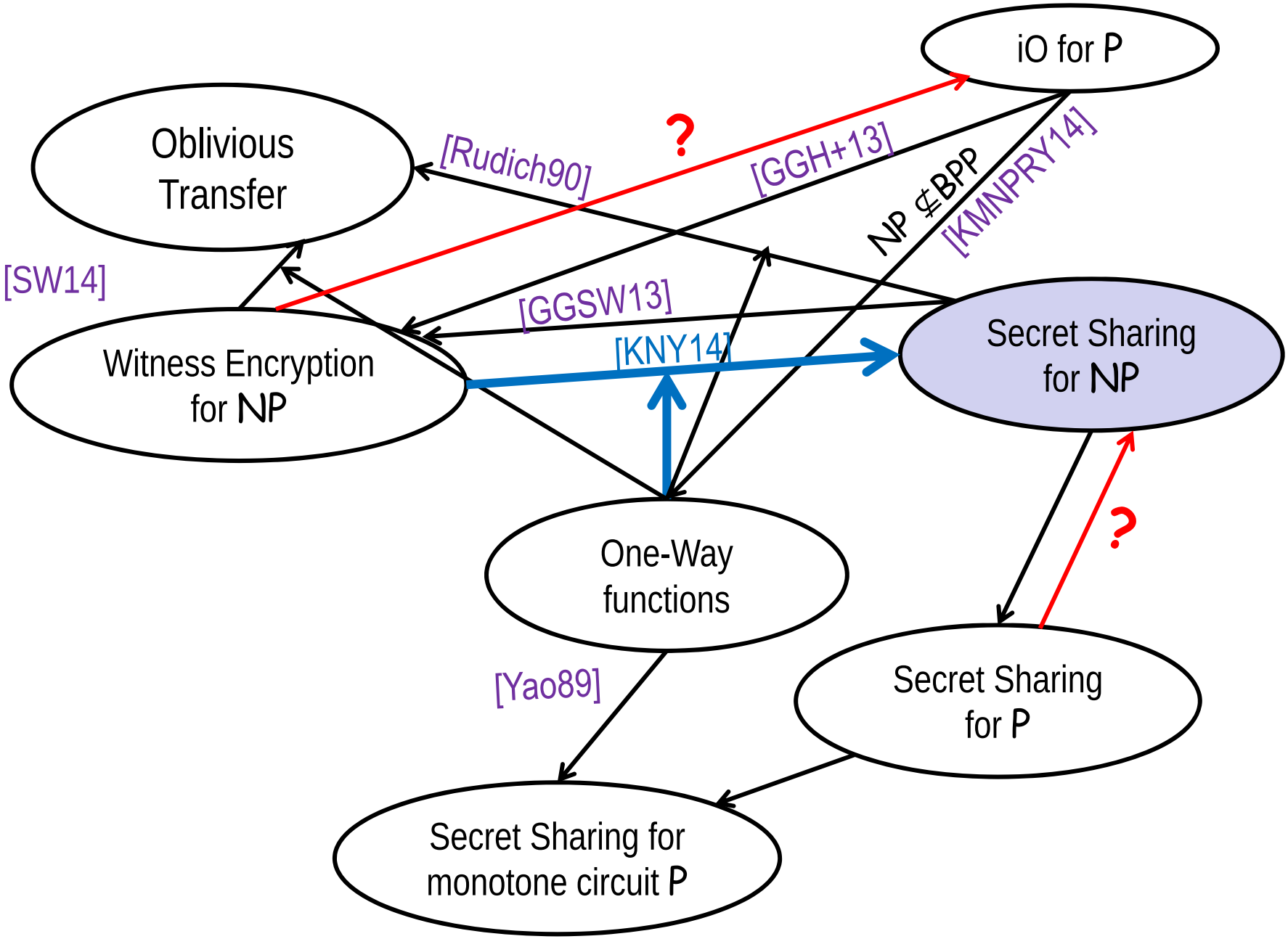
and witness w witness for X.

- Witness for M' consists of openings r_i such that $X_i=1$.
- Set $w'=(r'_1, \ldots, r'_n, w)$.

Security

Suppose an adversary $A=(A_{samp}, A_{dist})$ breaks the system.

- Construct an algorithm D that breaks the commitment scheme:
  - For a list of commitments $c_1, \ldots, c_n$ distinguish between two cases:
    - They are commitments of $1, \ldots, n$.
    - They are commitments of $n+1, \ldots, 2n$.


iO for P
Oblivious Transfer
[Rudich90]
?
[GGH+13]
NP ⊄ BPP
[KMNPRY14]
[SW14]
[GGSW13]
Witness Encryption for NP
[KNY14]
Secret Sharing for NP
One-Way functions
?
[Yao89]
Secret Sharing for P
Secret Sharing for monotone circuit P

Open Problems

- **Adaptive** choice of the set X.

Brakerski: diO

- Perfect Secret-Sharing Scheme for directed connectivity.
  - How to cope with the fan-out
- Computational Secret Sharing Scheme for Matching.
  - How to cope with negation?
- A secret sharing scheme for P based on less heavy cryptographic machinery.