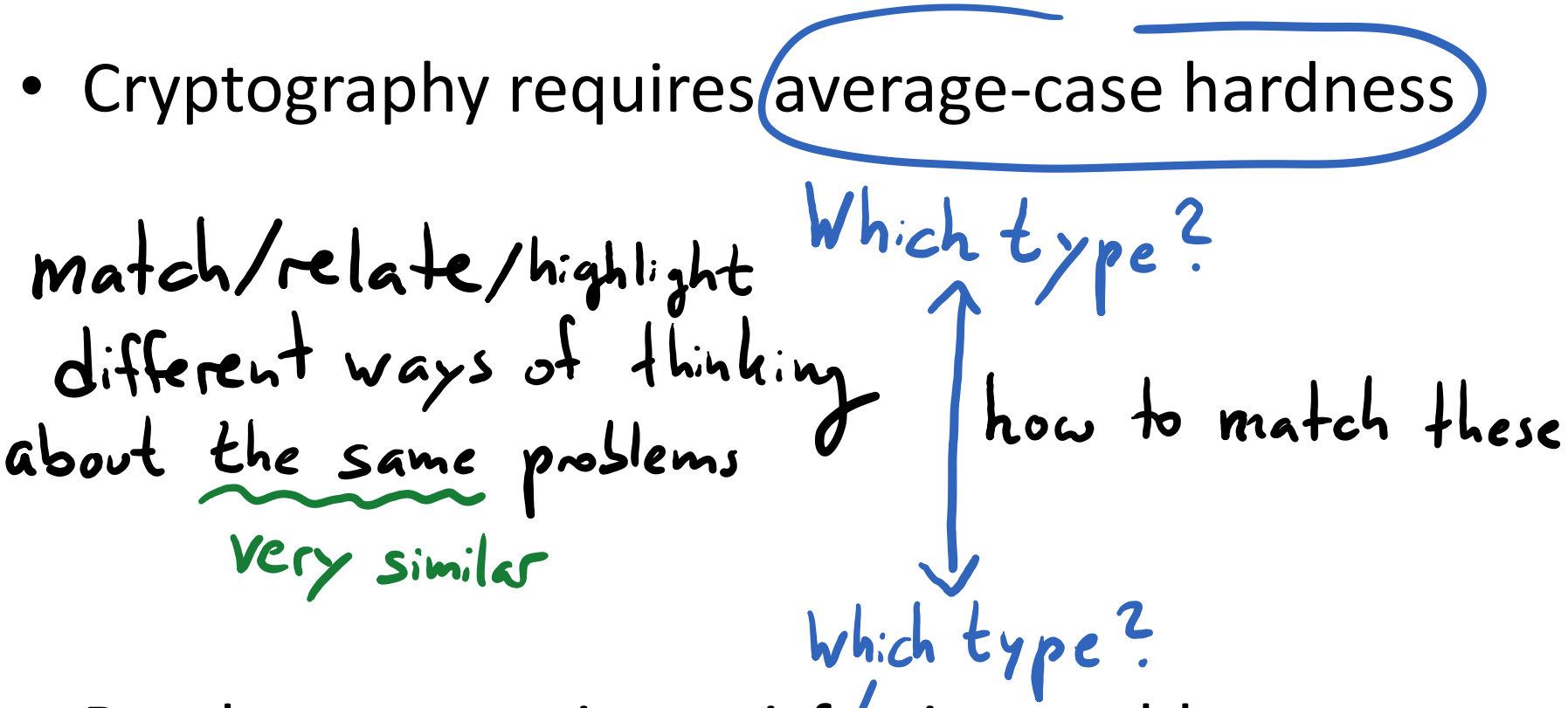
Relations between random constraint satisfaction problems & cryptography

Christopher Brzuska
Aalto University

Goal of this Talk

- Cryptography requires average-case hardness



- Random constraint satisfaction problems provide average-case hardness.

Outlook

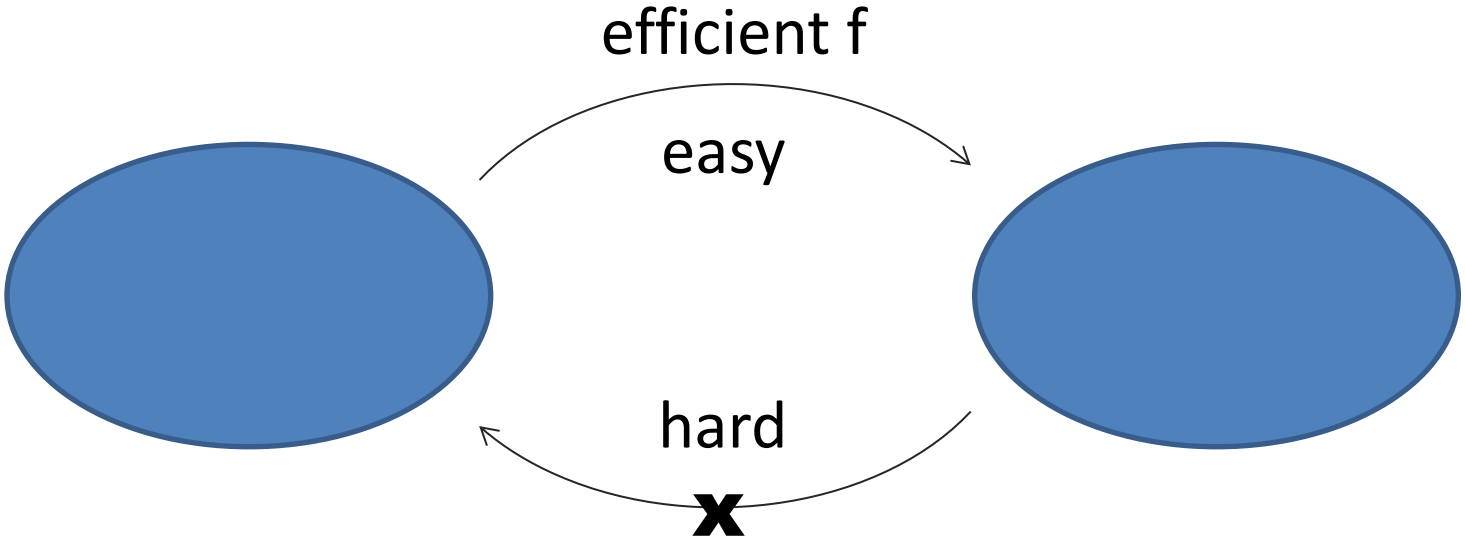
1. Two cryptographic problems
    a) One-way functions (OWF)
    b) Pseudorandom generators (PRG)
2. How to see PlantedCSPs as one-way functions
    a) Ugly way (general)
    b) Nice way (more restricted)
    c) Different views on sampling: Goldreich‘s OWF
    d) Predicates & densities
3. RandomCSP vs. PlantedCSP as PRG
    a) Goldreich‘s PRG
    b) Predicates & densities
4. Discussion/Questions

One-Way Function

uniformly random x

A deterministic, poly-time computable $f: \{0,1\}^* \to \{0,1\}^*$ is one-way if for all probabilistic poly-time adversaries A

$$\Pr_{x \leftarrow^{\$} \{0,1\}^n} [A \; inverts \; f(x)] \to_{n \to \infty} 0$$



One-Way Function

uniformly random x

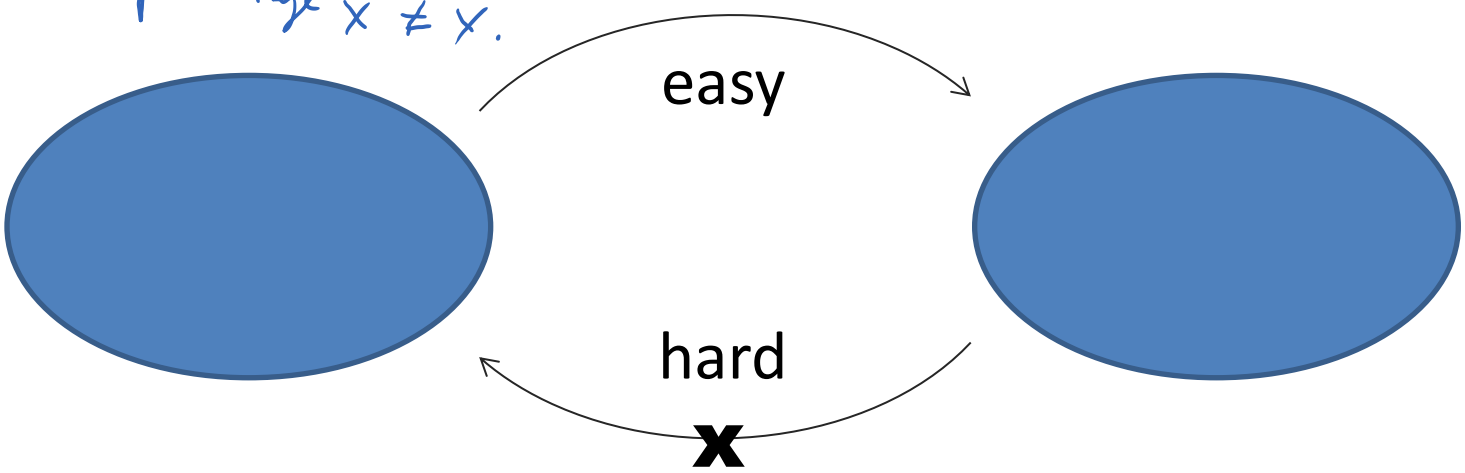
faster than any positive inverse polynomial

A deterministic, poly-time computable $f: \{0,1\}^* \to \{0,1\}^*$ is one-way if for all probabilistic poly-time adversaries A

$$\Pr_{x \leftarrow^{\$} \{0,1\}^n}[A(f(x)) \in f^{-1}(f(x))] \to_{n \to \infty} 0$$

successful inverter even if it returns a pre-image $x' \neq x$.

pre-image is not necessarily unique



Pseudorandom Gen

$\forall x \; |g(x)| = s$

Very different distributions.

Example: $s(n) = 2n$

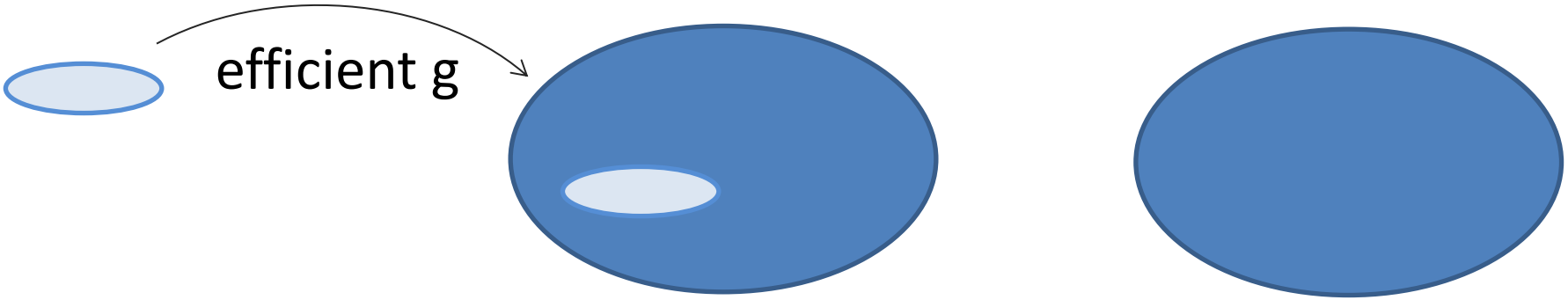
$|\{0,1\}^n| = 2^n \quad |\{0,1\}^{s(n)}| = 2^{2n}$

$2^n / 2^{2n} = 2^{-n}$ exp. small!

A det., (n,s(n))-length-expar

$g: \{0,1\}^* \to \{0,1\}^*$ is a pseudorandom generator if for all prob. poly-time adversaries A

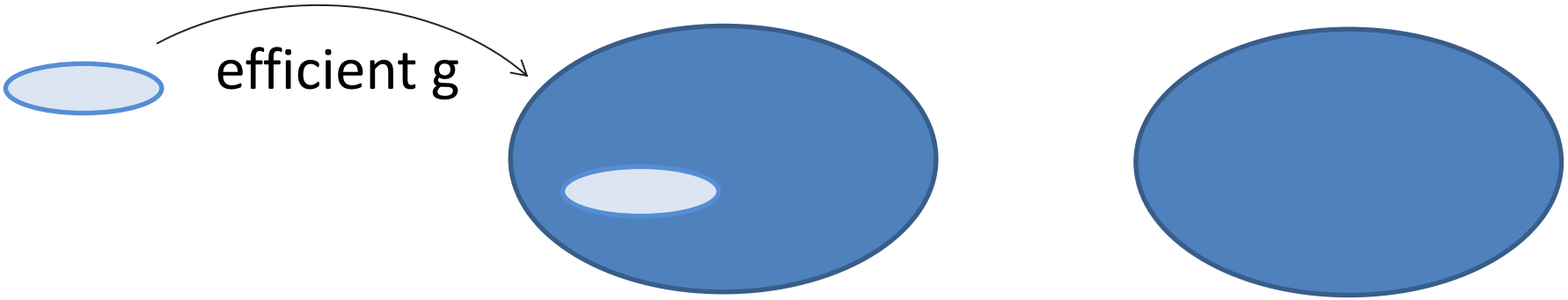
A cannot distinguish the pseudorandom string g(x) from a truly random string y of length |s(|x|)|.



Pseudorandom Generator

A det., (n,s(n))-length-expanding, poly-comput. $g:\{0,1\}^* \to \{0,1\}^*$ is a pseudorandom generator if for all prob. poly-time adversaries A

$$\left| \Pr_{x \leftarrow \$ \{0,1\}^n} [A(g(x)) = 1] - \Pr_{y \leftarrow \$ \{0,1\}^{s(n)}} [A(y) = 1] \right| \to_{n \to \infty} 0$$



[HILL] Hastad, Impagliazzo, Levin, Luby
A PRG from any OWF.

Quiz

Is every PRG a OWF? ✓ Inverting helps distinguish

Is every OWF a PRG? ✗ No. $f(x) \| 0 \ldots 0$

If OWFs exists, are there PRGs? ✓ (not easy)

If PRGs exist, are there OWFs? ✓

Does public-key encryption imply OWFs? ✓

"encryption should be one-way"

Impagliazzo, Luby

"OWFs are essential for complexity-based cryptography"

OWF: one-way

PRG: length-expanding, ind. from random

Outlook

1. Two cryptographic problems
    a) One-way functions (OWF)
    b) Pseudorandom generators (PRG)
2. How to see PlantedCSPs as one-way functions
    a) Ugly way (general)
    b) Nice way (more restricted)
    c) Different views on sampling: Goldreich‘s OWF
    d) Predicates & densities
3. RandomCSP vs. PlantedCSP as PRG
    a) Goldreich‘s PRG
    b) Predicates & densities
4. Discussion/Questions

RandomCSP

Examples:

1. Draw m random 3SAT-constraints $C_1, \ldots, C_m$.

Task: find assignment **a** to the n variables $x_1, \ldots, x_n$ such that **a** satisfies $C_1, \ldots, C_m$.

2. On an n-node graph, include each edge with prob p.

Task: find a clique above a certain size s.

Study ratio m/n where
- problem is likely to have a solution
- the solution space is connected
- the solution space splits into clusters
- certain classes of algorithms fail
- .....

Study (n, p, s) where
- such a clique exists with high probability
- no such clique exists with high probability
- .....

PlantedCSP

A satisfying assignment exists regardless of ratio m/n.

- Hard to find some consistent $\mathbf{a}$?
- Hard to find $\mathbf{a}_{\text{planted}}$?
- Is $\mathbf{a}_{\text{planted}}$ unique?

One-way function type problem

Examples:

1. Draw assignment $\mathbf{a}_{\text{planted}}$ to $x_1, \ldots, x_n$.

Draw m random 3SAT-constraints $C_1, \ldots, C_m$ consistent with $\mathbf{a}_{\text{planted}}$.

Task: find assignment **a** to the n variables $x_1, \ldots, x_n$ such that **a** satisfies $C_1, \ldots, C_m$.

2. In an n-node graph, draw a subset of size s and make it a clique, for the remaining edges, include each edge with prob p.

A clique of size s exists regardless of n and p.

Task: find a clique above a certain size s.

PlantedCSP

1. $f(a_{planted}, r)$ inverting this function requires finding a consistent satisfying assignment.
2. $\tilde{f}(r_1, r_2)$ inverting this function requires finding a clique.

Examples:

1. Draw assignment $a_{planted}$ (uniform) to $x_1, \ldots, x_n$.

   constraints $C_1, \ldots, C_m$ (output; randomized process, can be modeled by random tape r)

   to the n variables $x_1, \ldots, x_n$

   $, \ldots, C_m$. (randomized process with randomness r_1)

2. In an n-node graph, draw a subset of size s and make it a clique, for the remaining edges, include each edge with prob p. (randomized process with randomness r_2)

Task: find a clique above a certain size s.

Ok and can be generalized to other planted CSPs…
…but is a little ugly ☺

A restricted class of CSPs

CSP

We allow y_j to say whether $P(a_{S_j}) = 0$ or $P(a_{S_j}) = 1$.

This would not make sense for a highly biased predicate such as 3SAT. If all litterals are false, one learns 3 variables immediately.

n variables $x_1, \ldots, x_n$

m constraints $C_1, \ldots, C_m$

Balanced predicate $P: \{0,1\}^k \to \{0,1\}$

$$\Pr_{z \leftarrow^{\$} \{0,1\}^k} [P(z) = 1] = 1/2$$

Okay with bal. pred.

$C_j = (S_j, y_j)$

y_j: bit

S_j: ordered set of variable indices

$\boldsymbol{a} \in \{0,1\}^n$ is a solution for $C_1, .., C_m$ if for all j

$$P(\boldsymbol{a}_{S_j}) = y_j$$

CSP

n variables $x_1, \ldots, x_n$

m constraints $C_1, \ldots, C_m$

Balanced predicate $P: \{0,1\}^k \to \{0,1\}$

$$\Pr_{z \leftarrow^{\$} \{0,1\}^k} [P(z) = 1] = 1/2$$

$C_j = (S_j, y_j)$

y_j: bit

S_j: ordered set of variable indices

$\boldsymbol{a} \in \{0,1\}^n$ is a solution for $C_1, .., C_m$ if for all j

$$P(\boldsymbol{a}_{S_j}) = y_j$$

RandomCSP: Fix n, m, (balanced) P

Sample $C_1, \ldots, C_m$

$C_j = (S_j, y_j)$

y_j: bit

S_j: ordered set of variable indices

$\boldsymbol{a} \in \{0,1\}^n$ is a solution for $C_1, .., C_m$ if for all j

$$P(\boldsymbol{a}_{S_j}) = y_j$$

RandomCSP: Fix n, m, (balanced) P

Sample subsets $S_1, \ldots, S_m$ of {1,..,n} of size k.

Sample bits $y_1, \ldots, y_m$. Ok because P is balanced.

$C_j = (S_j, y_j)$

y_j: bit

S_j: ordered set of variable indices

$\boldsymbol{a} \in \{0,1\}^n$ is a solution for $C_1, \ldots, C_m$ if for all j

$$P(\boldsymbol{a}_{S_j}) = y_j$$

RandomCSP: Fix n, m, (balanced) P

Sample subsets $S_1, \ldots, S_m$ of {1,..,n} of size k.

Sample bits $y_1, \ldots, y_m$.

$C_j = (S_j, y_j)$

y_j: bit

S_j: ordered set of variable indices

$\boldsymbol{a} \in \{0,1\}^n$ is a solution for $C_1, .., C_m$ if for all j

$$P(\boldsymbol{a}_{S_j}) = y_j$$

PlantedCSP: Fix n, m, (balanced) P

Sample subsets $S_1, \dots, S_m$ of {1,..,n} of size k.

Do not depend on **a**.

Sample bits **a** = $a_1..a_n$

Compute bits $y_1 := P(\boldsymbol{a}_{S_1}), \dots, y_m := P(\boldsymbol{a}_{S_m})$.

$C_j = (S_j, y_j)$

y_j: bit

S_j: ordered set of variable indices

$\boldsymbol{a} \in \{0,1\}^n$ is a solution for $C_1, .., C_m$ if for all j

$$P(\boldsymbol{a}_{S_j}) = y_j$$

PlantedCSP: Fix n, m, (balanced) P

subsets $S_1, \dots, S_m$ of {1,..,n} of size k.

Sample bits **a** = $a_1..a_n$

Compute bits $y_1 := P(\boldsymbol{a}_{S_1}), \dots, y_m := P(\boldsymbol{a}_{S_m})$.

$C_j = (S_j, y_j)$

y_j: bit

S_j: ordered set of variable indices

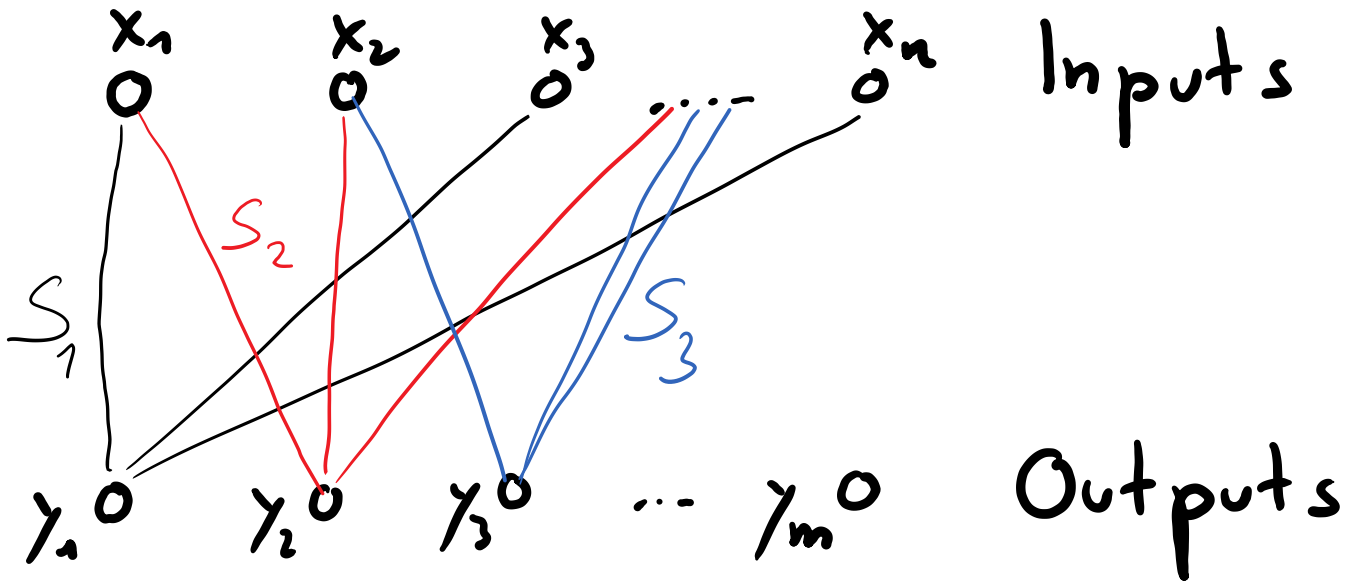
$\boldsymbol{a} \in \{0,1\}^n$ is a solution for $C_1, .., C_m$ if for all j
$P(\boldsymbol{a}_{S_j}) = y_j$

PlantedCSP: Fix n, m, (balanced) P

subsets $S_1, \ldots, S_m$ of {1,..,n} of size k.

Sample bits **a** = $a_1..a_n$

Compute bits $y_1 := P(\boldsymbol{a}_{S_1}), \ldots, y_m := P(\boldsymbol{a}_{S_m})$.



Goldreich's
One-Way Function

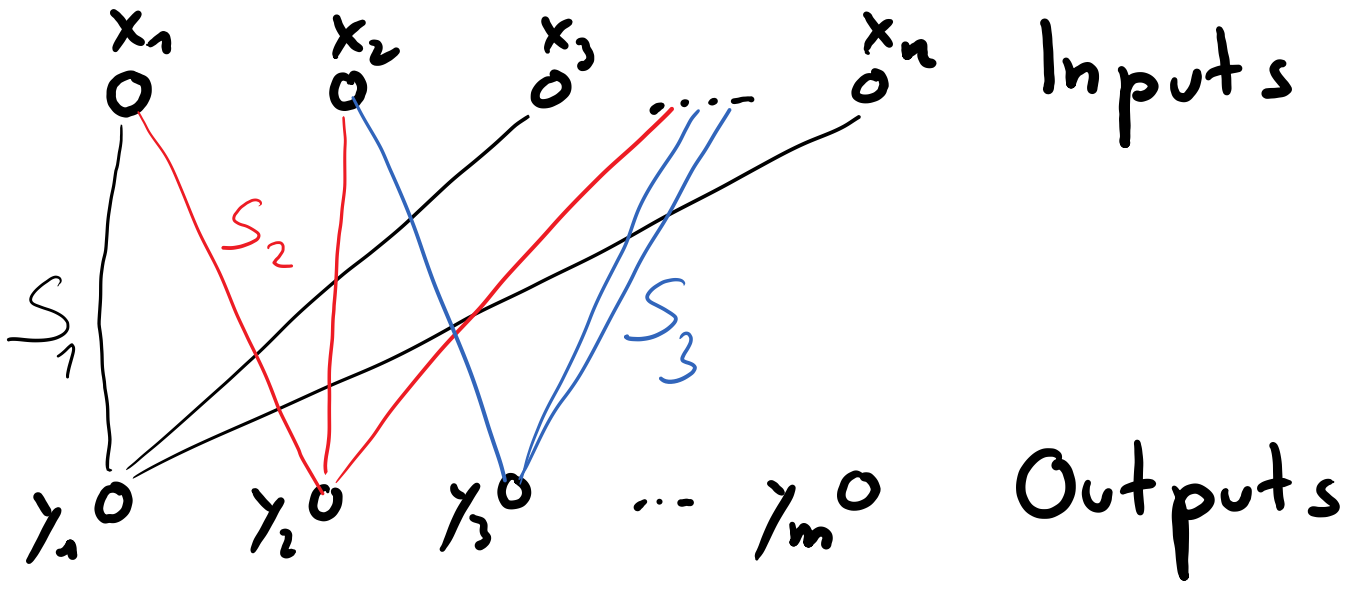
Fix n, m, (balanced) P
subsets $S_1, \ldots, S_m$ of {1,..,n} of size k.

expander

function description

Input
Sample bits $\mathbf{a} = a_1..a_n$

Compute bits $y_1 := P(\boldsymbol{a}_{S_1}), \ldots, y_m := P(\boldsymbol{a}_{S_m})$.



Goldreich's
One-Way Function

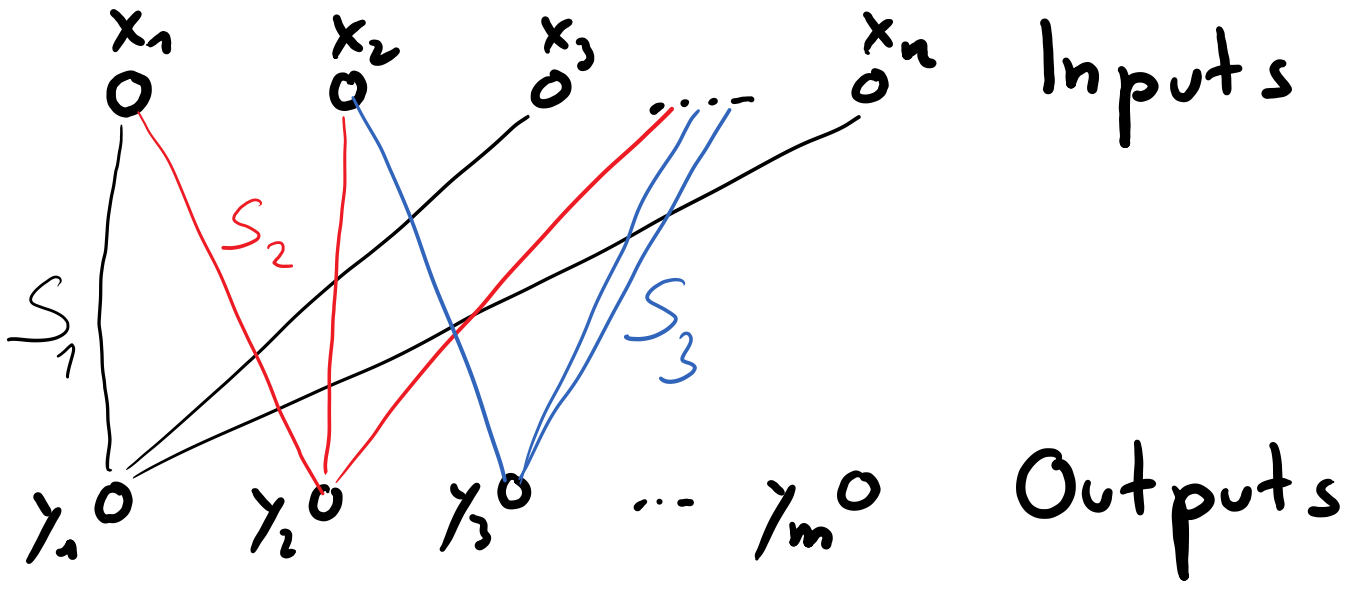
Fix n, m, (balanced) P
subsets $S_1, \ldots, S_m$ of {1,..,n} of size k.

expander

function description

Input: $\mathbf{a} = a_1..a_n$

Output: $y_1 := P(\boldsymbol{a}_{S_1}), \ldots, y_m := P(\boldsymbol{a}_{S_m})$.



Fix n, m, (balanced) P

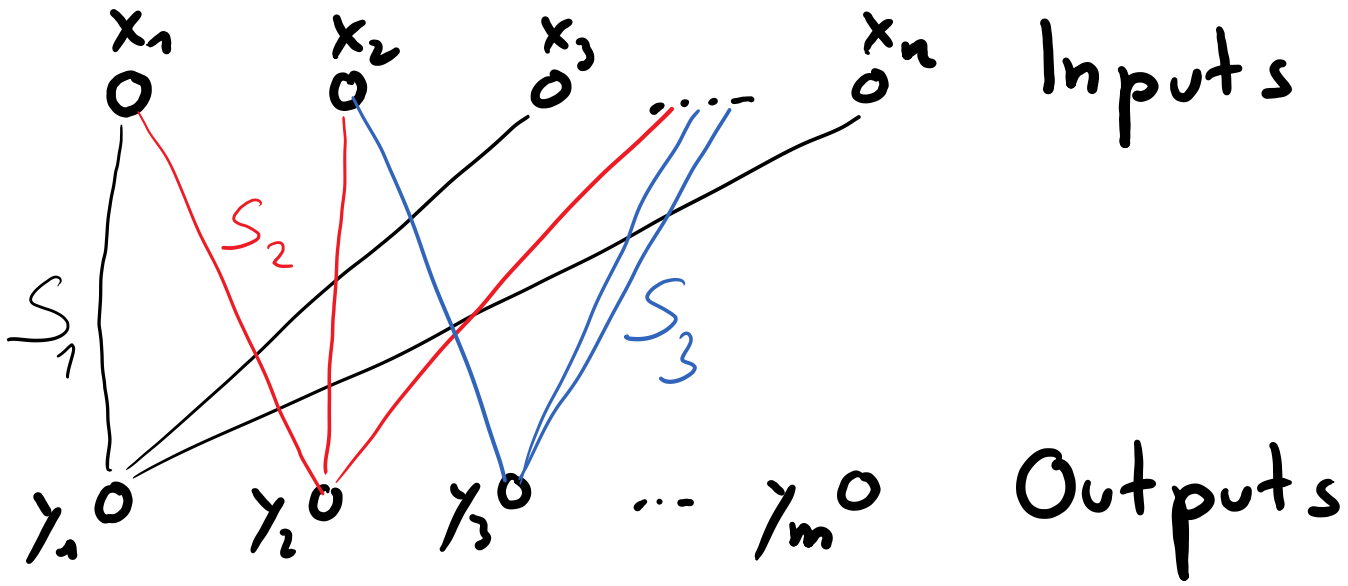
Planted CSP: subsets $S_1, \ldots, S_m$ of {1,..,n} of size k.

Goldreich's OWF:

A random graph is a good expander with high probability

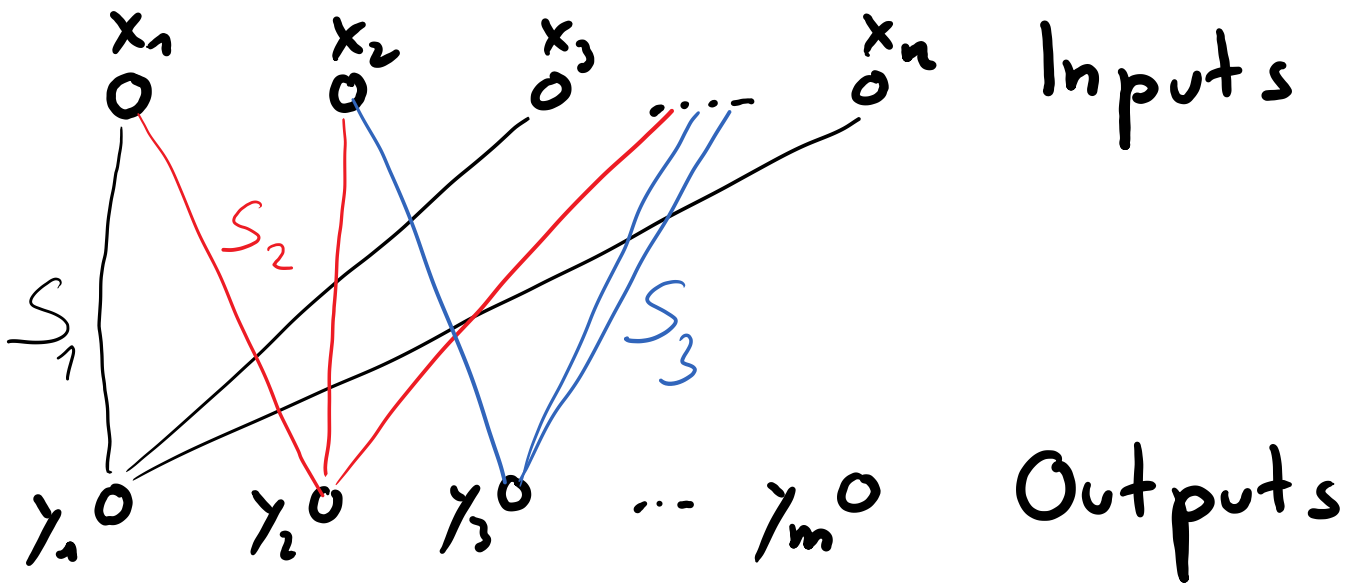
Input: $\mathbf{a} = a_1..a_n$

Output: $y_1 := P(\boldsymbol{a}_{S_1}), \ldots, y_m := P(\boldsymbol{a}_{S_m})$.



Goldreich's OWF: Predicates & density

- Balanced predicates (better than random pred.)
- Should not correlate with 1 variable
- Else, can be inverted for m=c·n for some constant c.



Bogdanov, Qiao, On the Security of Goldreich's One-Way Function

Outlook

1. Two cryptographic problems
   a) One-way functions (OWF)
   b) Pseudorandom generators (PRG)
2. How to see PlantedCSPs as one-way functions
   a) Ugly way (general)
   b) Nice way (more restricted)
   c) Different views on sampling: Goldreich's OWF
   d) Predicates & densities
3. RandomCSP vs. PlantedCSP as PRG
   a) Goldreich's PRG
   b) Predicates & densities
4. Discussion/Questions

same function as the function we just saw, just with the additional hope that it is not only one-way but also pseudorandom

PlantedCSP vs. RandomCSP

Fix n, m, (balanced) P

≤ 1 statistic security

< 1 computational sec.

indistinguishable for efficient alg.?

PlantedCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits **a** = $a_1..a_n$

Compute bits

$y_1 := P(\boldsymbol{a}_{S_1}), \dots, y_m := P(\boldsymbol{a}_{S_m})$.

$Return\ y_1, \dots, y_m$

contiguous up to $m/n <$ condensation threshold, known for classes of predicates*

RandomCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits

$y_1, \dots, y_m$.

$Return\ y_1, \dots, y_m$

*Coja-Oghlan, Kapetanopoulos, Müller
The replica symmetric phase of random constraint satisfaction problems.

PlantedCSP vs. RandomCSP
Fix n, m, (balanced) P

PlantedCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits **a** = $a_1..a_n$

Compute bits

$y_1 := P(\boldsymbol{a}_{S_1}), \dots, y_m := P(\boldsymbol{a}_{S_m})$.

Return $y_1, \dots, y_m$

RandomCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits

$y_1, \dots, y_m$.

Return $y_1, \dots, y_m$

PlantedCSP vs. RandomCSP

Fix n, m, (balanced) P

Fixed size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

PlantedCSP:

Sample bits **a** = $a_1..a_n$

Compute bits

$y_1 := P(\boldsymbol{a}_{S_1}), \dots, y_m := P(\boldsymbol{a}_{S_m})$.

$Return\ y_1, \dots, y_m$

RandomCSP:

Sample bits

$y_1, \dots, y_m$.

$Return\ y_1, \dots, y_m$

PlantedCSP vs. RandomCSP

Fix n, m, (balanced) P

Fixed size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

PlantedCSP:

Sample bits **a** = $a_1..a_n$

Compute bits

$y_1 := P(\boldsymbol{a}_{S_1}), \dots, y_m := P(\boldsymbol{a}_{S_m})$.

$Return\ y_1, \dots, y_m$

Random String:

Sample bits

$y_1, \dots, y_m$.

$Return\ y_1, \dots, y_m$

PlantedCSP vs. RandomCSP

Fix n, m, (balanced) P

Fixed size-k subsets $S_1, \ldots, S_m$ of {1,..,n}.

Output of Goldreich PRG on uniformly random input:

Sample bits **a** = $a_1..a_n$

Compute bits

$y_1 := P(\boldsymbol{a}_{S_1}), \ldots, y_m := P(\boldsymbol{a}_{S_m})$.

Return $y_1, \ldots, y_m$

Random String:

Sample bits

$y_1, \ldots, y_m$.

Return $y_1, \ldots, y_m$

PlantedCSP vs. RandomCSP
Fix n, m, (balanced) P

PlantedCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits **a** = $a_1..a_n$

Compute bits

$y_1 := P(\boldsymbol{a}_{S_1}), \dots, y_m := P(\boldsymbol{a}_{S_m})$.

Return $y_1, \dots, y_m$

RandomCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits

$y_1, \dots, y_m$.

Return $y_1, \dots, y_m$

The higher m/n, the easier the problem. For very high m/n, we have efficient refutation algorithms* (proving unsatisfiability) for large classes of predicates which allows us to distinguish…

PlantedCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits **a** = $a_1..a_n$

Compute bits

$y_1 := P(\boldsymbol{a}_{S_1}), \dots, y_m := P(\boldsymbol{a}_{S_m})$.

Return $y_1, \dots, y_m$

RandomCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits

$y_1, \dots, y_m$.

Return $y_1, \dots, y_m$

*Allen, O'Donnell, Witmer, *How to refute a random CSP.*

The higher m/n, the easier the problem. For very high m/n, we have efficient refutation algorithms* (proving unsatisfiability) for large classes of predicates which allows us to distinguish…
…less is known about inverting.

PlantedCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits **a** = $a_1..a_n$

Compute bits

$y_1 := P(\boldsymbol{a}_{S_1}), \dots, y_m := P(\boldsymbol{a}_{S_m})$.

$Return\ y_1, \dots, y_m$

RandomCSP:

Sample size-k subsets $S_1, \dots, S_m$ of {1,..,n}.

Sample bits

$y_1, \dots, y_m$.

$Return\ y_1, \dots, y_m$

*Allen, O'Donnell, Witmer, *How to refute a random CSP.*

The higher m/n, the easier the problem. For very high m/n, we have efficient refutation algorithms* (proving unsatisfiability) for large classes of predicates which allows us to distinguish…
…less is known about inverting.

Discussion/Questions

PlantedCSP:

Sample size-k subsets $S_1, \ldots, S_m$ of {1,..,n}.

Sample bits **a** = $a_1..a_n$

Compute bits

$y_1 := P(\boldsymbol{a}_{S_1}), \ldots, y_m := P(\boldsymbol{a}_{S_m})$.

$Return\ y_1, \ldots, y_m$

RandomCSP:

Sample size-k subsets $S_1, \ldots, S_m$ of {1,..,n}.

Sample bits

$y_1, \ldots, y_m$.

$Return\ y_1, \ldots, y_m$

*Allen, O'Donnell, Witmer, *How to refute a random CSP.*

Thanks! ☺

1. Two cryptographic problems
    a) One-way functions (OWF)
    b) Pseudorandom generators (PRG)
2. How to see PlantedCSPs as one-way functions
    a) Ugly way (general)
    b) Nice way (more restricted)
    c) Different views on sampling: Goldreich‘s OWF
    d) Predicates & densities
3. RandomCSP vs. PlantedCSP as PRG
    a) Goldreich‘s PRG
    b) Predicates & densities
4. Discussion/Questions