On the Construction of One-Way Functions from Average Case Hardness

Noam Livne
Weizmann Institute

Good riddles

- What makes a riddle a good riddle?
- Here’s a riddle:
  - Three turtles are walking in the desert.
  - 1st one says: “Behind me are two turtles.”
  - 2nd one says: “In front of me is one turtle, and behind me is another one.”
  - 3rd one says: “In front of me are two turtles, and behind me is one.”
  - How is this possible?
- A good riddle is a riddle that is hard, but for which **the one telling it knows the solution** (otherwise it’s just an annoying question).

Background

- One-way functions (OWF) are functions that are easy to compute, but hard on average to invert.
- OWF's are necessary for nearly all crypto, and sufficient for a lot.
- Since the existence of OWF implies P≠NP, a line of work studied the possibility of proving the existence of OWF ***based*** on the assumption that P≠NP [*Brassard '79, Feigenbaum&Fortnow '93, Bogdanov&Trevisan '03, AGGM '06*].
- Bottom line: certain types of reductions cannot reduce the security of certain types of OWF's to P≠NP (under some assumptions).

Our starting point

- Such reductions attempt to overcome two challenges at once:

P≠NP) average-case hardness) OWF

worst-case to average-case reduction

(Average-case hardness (ACH): A search problem R and a sampler S where R is hard on average under S.)

- Since the first challenge is hard by itself, it is inviting to study the second:

Can we prove ACH implies OWF?

Can we prove ACH implies OWF?

Suppose we have a relation R that is hard under a sampler S, and we want to prove the existence of OWF. 2 approaches:

1. For some candidate OWF, reduce its security to the hardness of R under S.
2. ***Construct*** a OWF from (R,S).

Can we prove ACH implies OWF ?

Why?

The OWF maps the randomness for S*, to the ***instance only*** (without the solution).

If one could retrieve the randomness given the instance, he could run S* on that randomness and obtain the ***pair***.

• R need not be poly-time verifiable.

• S* need not output only YES-instances.

… is hard under a sampler S, … of OWF. 2 approaches:

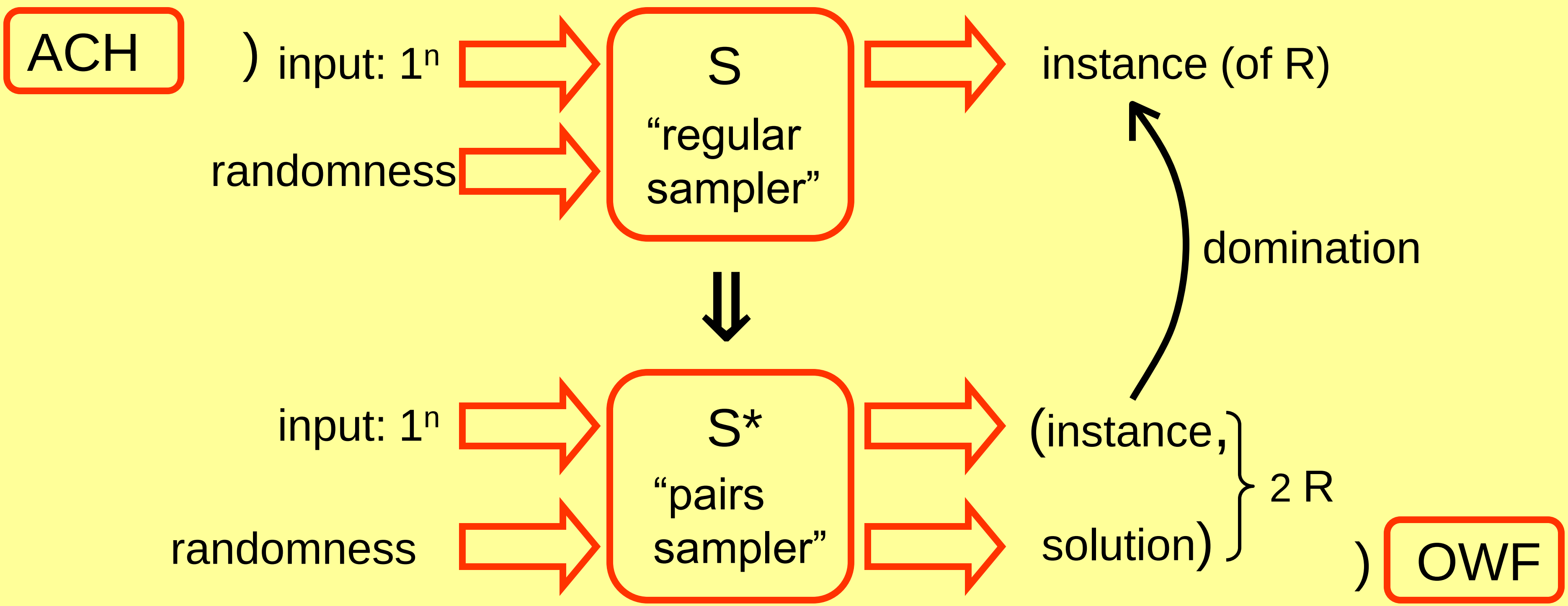
… its security to the

… ke

"If we can sample ha… …llenges for w… we know the answers, then OWF ex…s."

"If there exists a search problem R and a poly-time sampler S that outputs instance-solution pairs of R, where the distribution on the instances is hard on average, then OWF's exist."*

The approach for proving ACH implies OWF



- *Question: When can a regular sampler be "transformed" into a pairs sampler?*

Our result

Under some standard assumption (which is weaker than the existence of OWP):

Roughly: For every polynomial p, there is a pair (R,S) that cannot be transformed into a pairs sampler S* with randomness complexity p.

Our result

Under some standard assumption (which is weaker than the existence of OWP):

There exists a sampler S s.t. for any (arbitrarily large) polynomial p and any (arbitrarily small) super-polynomial function f there exists a search problem R s.t.:

- R is hard under S;
- R is polynomially bounded and is verifiable in time f(n);
- There is no efficient pairs sampler S* for (R,S):
  - If the 1st element output by S* is an R-YES-instance, then the 2nd is a solution;
  - The marginal distribution on the 1st elements dominates S.
  - S* has randomness complexity p.

The idea

- **Our assumption:** There exists (R’,S’) with some properties (in particular R’ is hard under S’). Implied by OWP.
- Based on (R’,S’), we construct (R,S).
- We assume an S* exists for (R,S), and show that R’ is not hard under S’, in contradiction.
- **The crux:** R and S are constructed such that:
  - (R,S) “inherit” the hardness of (R’,S’);
    - R’ is “embedded” in R, and S “imitates” S’;
  - Any S* for (R,S) enables solving R’ in the worst case:

{(☁,x),(w,☁)}2 R

){x,w}2 R’

The idea

Any S* for (R,S) enables solving R’ in the worst case:

- Clearly, S* enables obtaining **random** R-instance-solution pairs (and thus **random** R’-instance-solution pairs);
- The solution in each pair helps obtaining randomness for a **new pair**, where the R’-instance is **a little closer to any desired instance**;
- Thus, given some instance x of R’, using S* we start with a random pair of R (and thus a random embedded instance of R’), and have the embedded R’-instance become closer and closer to x;
- When reaching an R-instance with x embedded in it, the R-solution contains an R’-solution for x.

{(☁,x),(w,☁)}2 R) {x,w}2 R’

Proof by animation

"If S* exists (for R,S) then R' can be solved in the worst-case."

Suppose we want to solve x under R'.

But, we want to diagonalize against all possible S*'s…

S* should dominate S. So we let S output any sampler with noticeable prob.

This "enforces" any potential S* to output any sampler with noticeable prob.

input: 1^n ⇒ **S*** (For R,S) $r^{(0)}$ ⇒ {((S*,☁), $x^{(0)}$) ⊕ e_i),

randomness ⇒ ($w^{(0)}$, $r^{(0)}$)}, $r^{(0)}_i$, $r^{(0)}_n$)} } 2 R

This will help us find $x^{(1)}$ that is Hamming-closer to x.

(☁, $x^{(0)}$) is an R-YES-instance

){(☁, $x^{(0)}$), ($w^{(0)}$, ☁)}2 R

){$x^{(0)}$, $w^{(0)}$}2 R'

S*($r^{(0)}_i$)={(S*, $x^{(0)} \oplus e_i$), ☁}

Definition of R and S

We're given R',S'.

Definition of R:

R : $\{ (\langle M \rangle, x) , (w, r_1, \ldots, r_{|x|}) \}$ is in R iff:

- w is a solution for x under R';
- For all i, on input r_i the machine M outputs a pair where the 1st element is $(\langle M \rangle, x \oplus e_i)$ (in at most $f(|x|)$ steps);
- For all i, $|r_i| \leq p(|x|)$.

Definition of R and S

We're given R',S'.

Definition of S:

On input 1^n:

1. Choose i uniformly from [0,n].
2. Choose a potential sampler $\langle M \rangle$ uniformly from $\{0,1\}^i$.
3. Choose x of length n-i according to the distribution of S'.
4. Output $(\langle M \rangle, x)$.

Note: $\Pr[S(1^n) = (\langle M \rangle, x) \text{ for some } x] = (n+1)^{-1}\, 2^{-|\langle M \rangle|}$.

"Every machine is output by S with noticeable probability."

Elaborating the proof (and the assumption)

We assume S* exists for (R,S) and solve R' in the worst case:

On input x:

1. Run $S^*(1^{|\langle S^*\rangle|+|x|})$ repeatedly until it outputs a pair of the form $\{(\langle S^*\rangle, x^{(0)})\ ,\ y\}$.

   S outputs ⟨S⟩ with noticeable probability, S* dominates S*

) S outputs ⟨S*⟩ with noticeable probability*

) Step 1 takes expected poly time

Elaborating the proof (and the assumption)

We assume S^* exists for (R,S) and solve R' in the worst case:

On input x:

1. Run $S^*(1^{|\langle S^*\rangle|+|x|})$ repeatedly until it outputs a pair of the form $\{(\langle S^*\rangle, x^{(0)}) , y\}$.

   We want $(\langle S^*\rangle, x^{(0)})$ to have an R-solution.

   a) *We assume some properties on R' and S'.*

   b) *These yield some properties of R,S.*

   c) *These yield that all involved instances are YES-instances.*

   The existence of (R',S') with these properties is implied by the existence of onto OWF (and OWP).

Elaborating the proof (and the assumption)

We assume S* exists for (R,S) and solve R' in the worst case:

On input x:

1. Run $S^*(1^{|\langle S^*\rangle|+|x|})$ repeatedly until it outputs a pair of the form $\{(\langle S^*\rangle, x^{(0)}) , y\}$.
2. Parse y to $(w^{(0)}, r^{(0)}_1, \ldots, r^{(0)}_n)$.
3. Let $i_1,\ldots,i_h$ be the bits that are different between $x^{(0)}$ and x.

   For j=1 to h:

   Use $r^{(j-1)}_{i_j}$ as randomness for S* to obtain output

   $\{(\langle S^*\rangle, x^{(j)}), (w^{(j)}, r^{(j)}_1, \ldots, r^{(j)}_n)\}$.
4. Output $w^{(h)}$.

Interpretation of our result

- **We asked:** When can a regular sampler be "transformed" into a pairs sampler?
- **We saw:** (Under some assumption)**:** There is a (universal) sampler that for any polynomial randomness bound cannot be transformed into a pairs-sampler with that randomness WRT some R.
- "Transformed samplers (S*) can require arbitrarily large polynomial randomness."
- **A "generic" transformation:** Given any S that is hard for some (reasonable) R, transform it to a pairs sampler that uses randomness that depends ***only*** on S.
  - **A generic transformation does not exist.**

THANKS!