

Separation Results on the “One-More” Computational Problems

Emmanuel Bresson (DCSSI, France),

Jean Monnerat (UC San Diego, USA),

Damien Vergnaud (ENS, France)

[RSA Conference 2008](#)

Road Map

One-More Computational Problems

Overview of Our Separation Results

More Technical Details

Cryptographic Applications

Conclusion

Road Map

One-More Computational Problems

Overview of Our Separation Results

More Technical Details

Cryptographic Applications

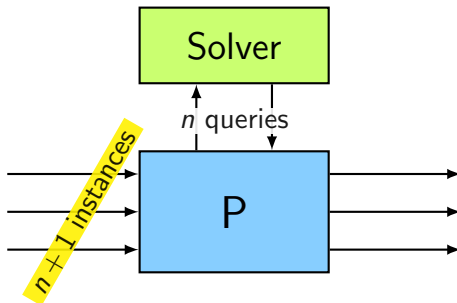
Conclusion

One-More Problems

Let P a computational problem.

One-More Computational Problem

n - P problem: being allowed to see the solutions of n instances, solve $n + 1$ instances



One-More Computational Problems

Introduced in 2001 by Bellare et al.

- Analysis of Chaum's blind signatures

Several subsequent work

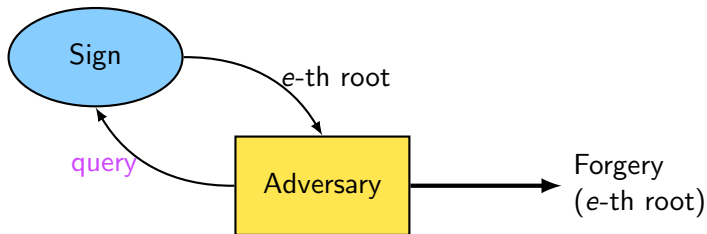
- One-more unforgeability (Pointcheval-Stern 00)
- GQ and Schnorr identifications (Bellare-Palacio 02)
- Blind BLS signatures (Boldyreva 03)
- RSA-based transitive signatures (Bellare-Neven 05)
- Schnorr signature vs Discrete Log (Paillier-Vergnaud 05)

Which confidence should we have?

Example and Application

Chaum's blind signature

- Classical RSA parameters: (N, e) and d with $de = 1 \pmod{\varphi(N)}$
- To sign a message M , ask for the e -th root of $r^e M$
 - Then extract M^d from the answer
- After having asked q signing queries, forge a new signature



Remarks on n -Problems

Chaum's blind signature can be proven under some one-more RSA assumption (the q -RSA)

- can one do better? (weaker assumption)
- can we relate the one-more RSA to RSA?

If $n > m$, the n -P problem is **easier** than m -P

- the $(n + 1)$ -P assumption is **stronger** than the n -P
- computational hierarchy $0\text{-P} \geq 1\text{-P} \geq \dots \geq n\text{-P}$

Our contributions



Evidences that some one-more problems may be easier than standard ones

More precisely:

- $(n + 1)$ -P is strictly easier than n -P, or n -P is easy
- computational hierarchy:

$$0\text{-P} > 1\text{-P} > 2\text{-P} > \dots > n\text{-P} = (n + 1)\text{-P} = \dots$$

Chaum's blind signature **cannot** be proven under the RSA assumption

- unless RSA is easy...
- **Warning !** No concrete weakness on the n -P problem were found ! (non-BB reductions may exist!)

Road Map

One-More Computational Problems

Overview of Our Separation Results

More Technical Details

Cryptographic Applications

Conclusion

Experiment: the P-problem

Problem P is defined via two algorithms

- Parameter generator
- Instance generator

$\text{param} = PG(1^k)$

$\text{ins} = IG(\text{param})$

sol

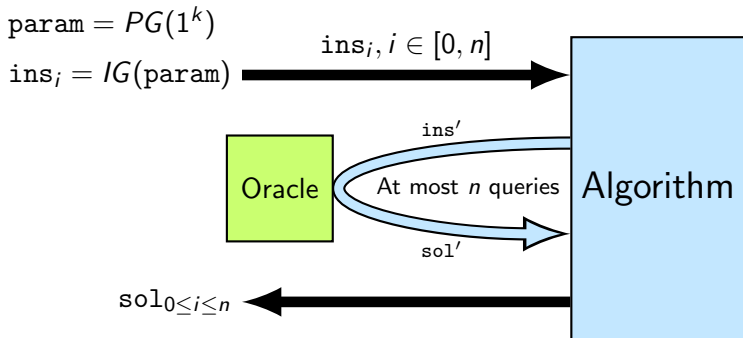
```
graph LR; ins[ins] --> Algorithm[Algorithm]; Algorithm --> sol[sol];
```

Algorithm

$\Pr[\text{Verif}(\text{param}, \text{ins}, \text{sol})] = 1$

Experiment: the One-More n-P problem

Algorithm A is now an oracle machine,
with access to a P-oracle



$$\Pr[\forall i, \text{Verif}(\text{param}, ins_i, sol_i)] = 1$$

Separations : What do we Prove Exactly?

Our separation results must be understood as follows:

- 1 Either there is a strict hierarchy:

$$n\text{-P} > (n+1)\text{-P}$$

this holds if $n\text{-P}$ is hard

- 2 Or there is an equivalence:

$$n\text{-P} = (n+1)\text{-P}$$

this holds if $n\text{-P}$ is easy (then both are easy)

Road Map

One-More Computational Problems

Overview of Our Separation Results

More Technical Details

Cryptographic Applications

Conclusion

Separations vs. Reductions

Reduction: algorithm A can be used to solve problem P

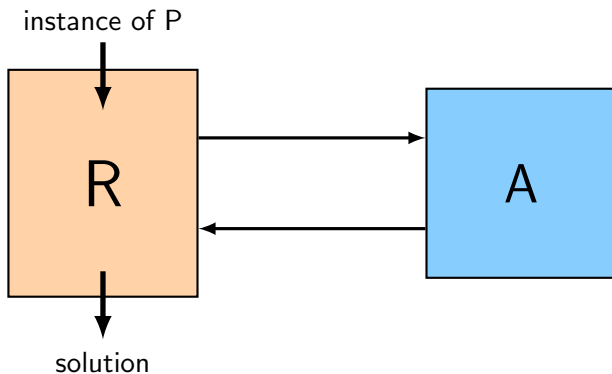
- We exhibit the **reduction** R
- R , using A as subroutine, is a P -solver
- *If P is hard*, then **A cannot exist**: **security**

Separation: reduction R can be used to solve problem Q

- We exhibit a **meta-reduction** M
- M , using R as a subroutine, is a Q -solver
- *If Q is hard*, then **R cannot exist**: **separation**

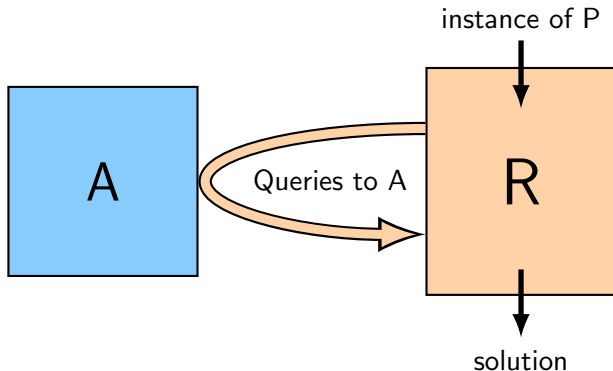
BlackBox Reductions

R provides inputs to A, and uses its output to solve P



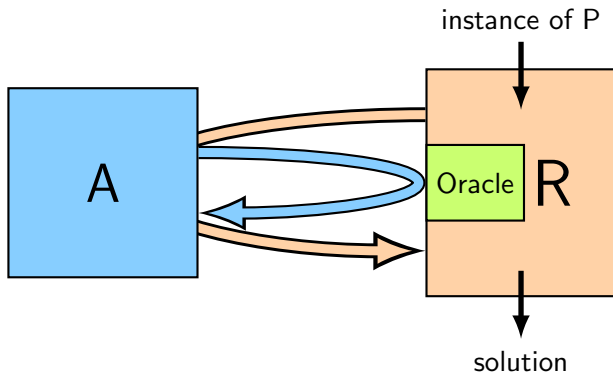
BlackBox Reductions

R can be seen as an oracle Turing machine !
A plays the role of an oracle for R



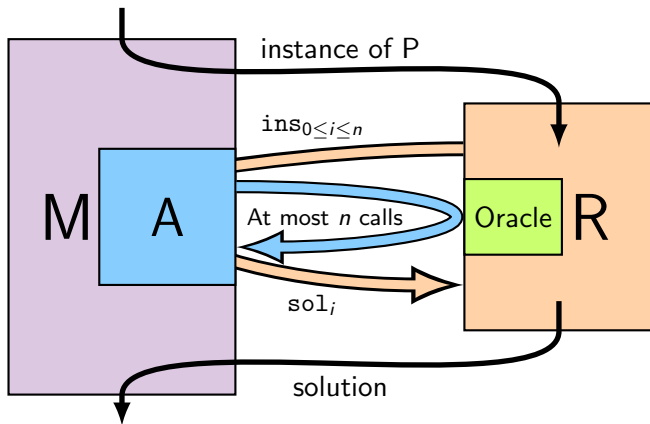
BlackBox Reductions

If A is itself an oracle machine, R must be able to simulate that oracle in order to properly use A



From Reductions to MetaReductions

A (meta) reduction can use R, provided the oracle for R (i.e. A) is simulated



The interactions

R has to interact with A : reminds that A plays the role of an oracle to R

We identify several types of queries

- **Launch** : starts a new execution of A with inputs and a fresh random tape (unknown to R)
- **Rewind** : starts an execution of A with a **previously used random tape** but **a new input**
- **Relaunch** : starts an execution with a **previously used random tape** and **the corresponding input**
- **Stop** : definitely stops the interaction

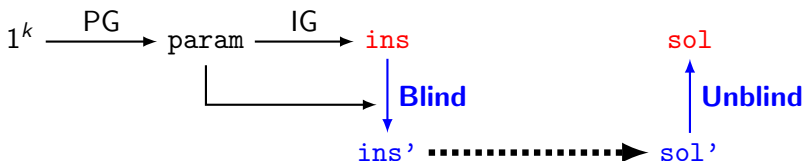
Preliminaries : the setting we are considering

The basic problem P must be **verifiable**

- exhaustive search is possible (in long time...)
- **RSA, DLog** fall in this category

The problem P must be **random self-reducible**

- **blinding algorithm** applied to instances
- **unblinding algorithm** applied to solutions

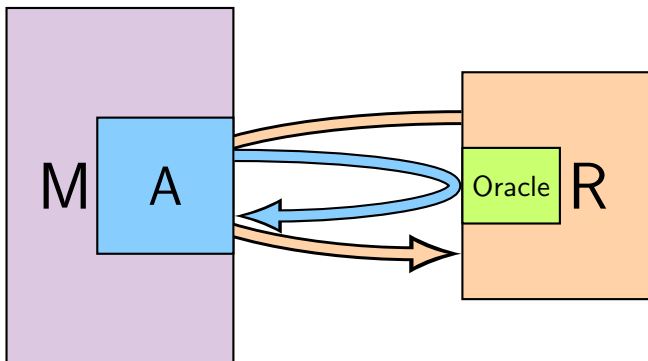


Experiment: the P-problem

Our Main Theorem

Theorem — Main Result

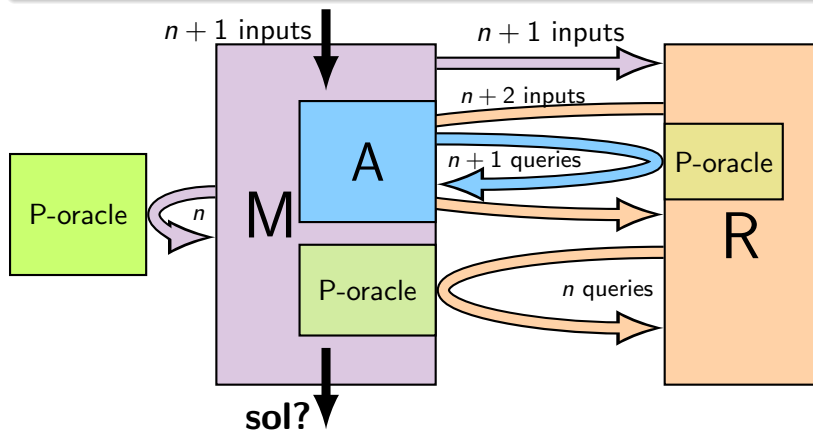
If R is a *parameter-invariant* reduction from n - P to $(n + 1)$ - P , then M is a n - P solver



Basic case with One “Launch” Execution

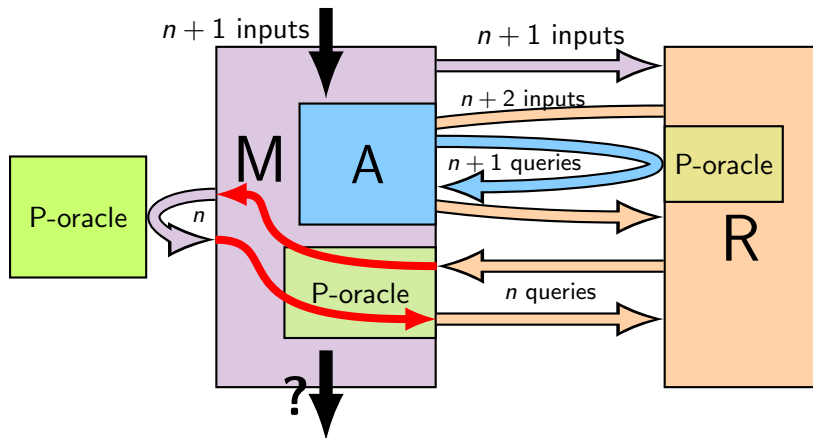
Theorem — Main Result

If R is a *parameter-invariant* reduction from n - P to $(n+1)$ - P , then M is a n - P solver



Basic case with One “Launch” Execution

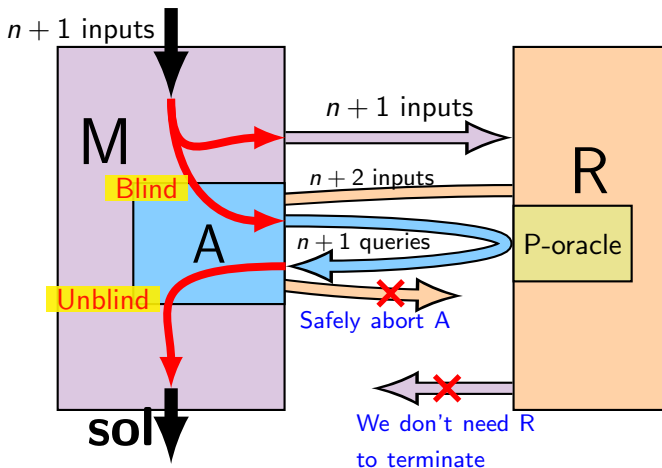
M starts R and must answer two types of queries
 First deal with P-oracle queries



Basic case with One “Launch” Execution

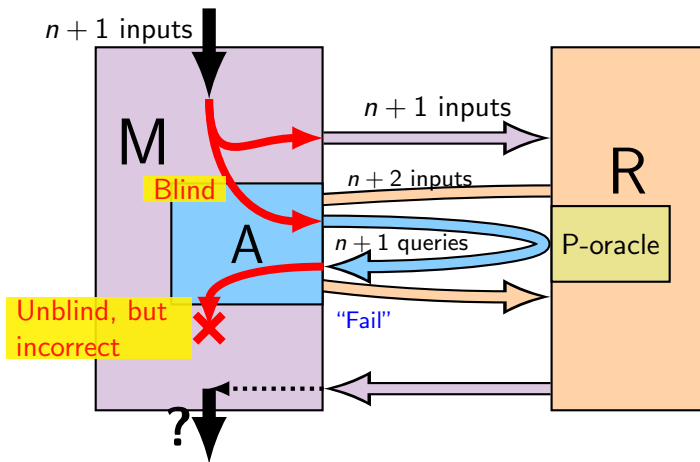
How to deal with “Launch” queries?

We will **exploit** the P-oracle simulated *by* R ...



Basic case with One “Launch” Execution

What happens if R cheats (in the simulation of P-oracle)?

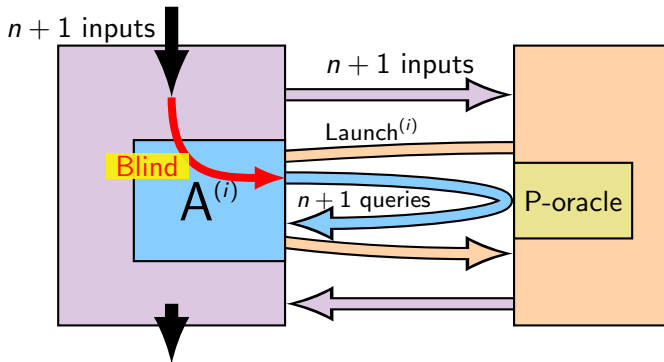


M makes A to fail and hopes for an “made in R ” solution

Extended to Multiple Queries

R can ask many “Launch” queries of A

How M must take care of this (how to blind its $n + 1$ inputs)?

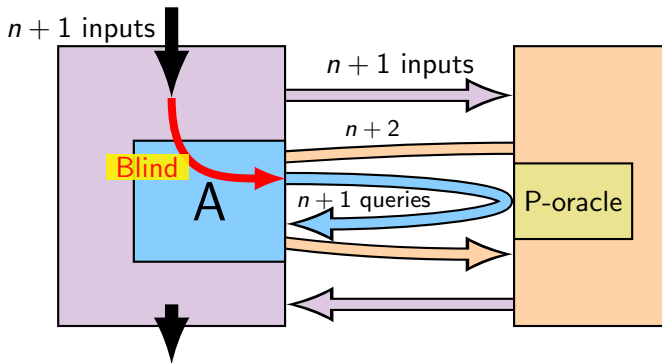


Not a big deal:

each **new** execution of A allows $n + 1$ **new P-queries** to R!

Extended to Multiple Queries

What about the **Rewind**? (rerun A with same random tape)
 How M must simulate A?



Not a big deal:

Queries asked by A depends only on **M's inputs!**

Results for the Discrete Logarithm

The Discrete Logarithm is:

- Verifiable
- Self-Reducible

The “parameters” of an instance consist in:

- The group
- The **generator**: (basis) relative to which discrete log should be computed, and relative to which a DL-oracle is available

We assumed R was “parameter-invariant”...

Extension to Algebraic Reductions

We extend **our main result** (applied to one-more Discrete Log) to **group invariant** reductions:

- R receive inputs in a group \mathbb{G} , relative to a basis $g \in \mathbb{G}$
- R can **launch A** with inputs in \mathbb{G} , **but** relative to $g' \neq g$
- R will **answer A's** DLog queries in **that basis g'**

The **price to pay** for that extension:

- R is an **algebraic reduction**: for *any* $Y \in \mathbb{G}$ output by R, we can **extract** integers a_i such that:

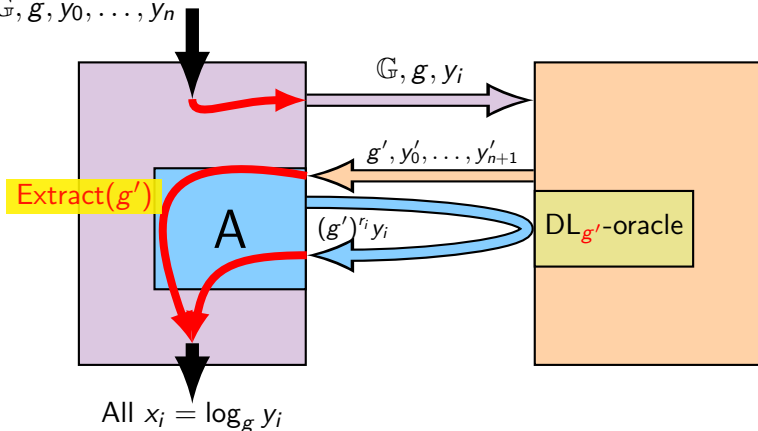
$$Y = g_1^{a_1} g_2^{a_2} \cdots g_k^{a_k}$$

where the g_i are the **inputs** of R

Extended to Algebraic Reductions

If R is **algebraic**, we use the **extract** feature to retrieve information relative to the second base

$\mathbb{G}, g, y_0, \dots, y_n$



Road Map

One-More Computational Problems

Overview of Our Separation Results

More Technical Details

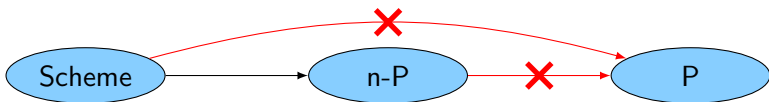
Cryptographic Applications

Conclusion

Cryptographic Applications

The one-more unforgeability of Chaum's signature can be reduced to the one-more RSA [BNPS03]

- The signing oracle is a RSA-inversion oracle
- Here: RSA cannot be reduced to one-more RSA
- Thus: **RSA cannot be reduced to Chaum's unforgeability**



The one-more unforgeability of blind BLS is equivalent to the (verifiable) one-more CDH problem [Bol03]

- Here: the CDH does not reduce to one-more CDH
- Thus: **the CDH cannot be reduced to blind BLS security**

Road Map

One-More Computational Problems

Overview of Our Separation Results

More Technical Details

Cryptographic Applications

Conclusion

Summary

It **seems difficult** to prove one-more problems as hard as their classical counterparts

This would require **non black-box** techniques

Relying security on one-more problems does not offer the **same security guarantees**