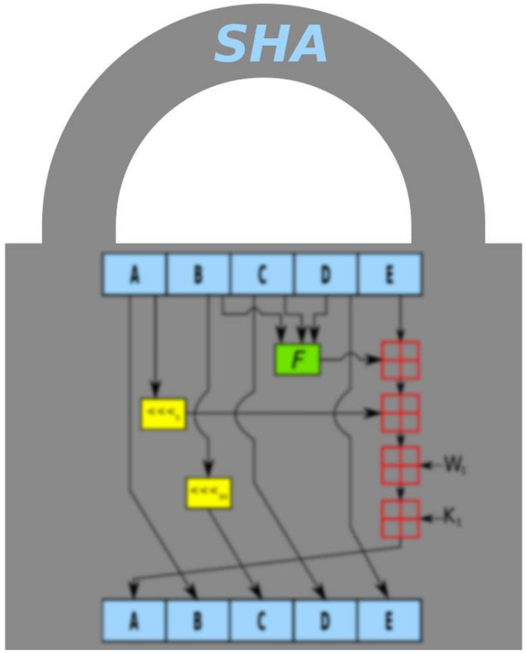
Fixing Cracks in the Concrete: Random Oracles with Auxiliary Input, Revisited

Yevgeniy Dodis
New York University

Joint work with Siyao Guo (Simons Institute, UC Berkeley)
Jonathan Katz (University of Maryland)

Hash Functions Are Ubiquitous



- OWFs
- PRG/PRFs
- MACs
- CRHFs
- KDFs
- …

How to assess the best possible concrete security for each application?

Random Oracle Model Methodology
[BR93]

Random Function O:[N]->[M]

A: T queries

Clean proofs, Precise bounds: e.g., OWFs/MACs $T/\min(N,M)$, PRFs/PRGs T/N, CRHFs T^2/M, etc.

Simple Proof Techniques: programmability, lazy sampling, distinguishing-to-extraction, etc.

<u>Practical heuristics</u>: for "natural" applications,
Security in ROM = Security in "standard model"
(for the best instantiation of O)

- Theoretical counter-examples [CGH04, etc.] "artificial" and don't affect <u>widely used</u> applications

Random Oracle Model Methodology
[BR93]

<u>Practical heuristics</u>: for "natural" applications,

Security in ROM = Security in "standard model"

(for the best instantiation of $\mathcal{O}$)

- Theoretical counter-examples [CGH04, etc.] "artificial" and don't affect <u>widely used</u> applications

Random Oracle Model Methodology
[BR93]

<u>Practical heuristics</u>: for "natural" applications,
Security in ROM = Security in "standard model"
(for the best instantiation of O)

- Theoretical counter-examples [CGH04, etc.] "artificial" and don't affect <u>widely used</u> applications

Cracks in the Concrete $\underbrace{}$ Security

	Random Oracle	Standard Model
CRHF:	T^2/M	1
OWF: [N]->[N]	T/N	1 (rainbow tables; in time $N^{2/3}$ [Hel80])
PRG:	T/N	$1/N^{1/2}$ (constant time [AGHP92])

Non-uniform Cracks in the Concrete $\overbrace{\text{Concrete}}^{\text{Security}}$

	Random Oracle	Standard Model
CRHF:	T^2/M	1
OWF: [N]->[N]	T/N	1 (rainbow tables; in time $N^{2/3}$ [Hel80])
PRG:	T/N	$1/N^{1/2}$ (constant time [AGHP92])

Non-Uniform Adversaries

- Modeled as families of circuits
  - Can "hardwire" arbitrary (bounded) "advice" before attacking the system
  - **<u>Preprocessing</u>**: special case of "computable" advice (corresponds to potentially implementable attack)
- Why/how did this become "standard" model?
  - Uniform model *too weak* (e.g., attacker can focus on a given security parameter in advance)
  - Sometimes *preprocessing realistic* (rainbow tables!)
  - Seems critical for *protocol composition* (i.e., ZK)
  - Wlog, *deterministic* attacker (P/poly=BPP/poly)
  - (Non-uniform) *hardness vs randomness* : non-uniform lower bounds => derandomization

Non-uniform Cracks in the Concrete

Security

Can we "salvage" ROM methodology and be consistent with **non-uniform** attackers?

[Unr07] YES: ROM with Auxiliary-Input (ROM-AI)

The ROM methodology is blatantly false for *most* natural and widely deployed applications when:

- Preprocessing is allowed
- The standard model adversary is non-uniform

Strictly worse!

Practical heuristics: for "natural" applications,

Security in ROM = Security in "standard model"

(for the best instantiation of O)

- Theoretical counter-examples [CGH04, etc.] "artificial" and don't affect widely used applications

Fixing Cracks in the Concrete

ROM-AI

Random function
$O:[N]\to[M]$

$A = (A_0, A_1)$

offline A_0: S bits

advice

online A_1 : T queries

- A_0: *computationally* *unbounded*, gets entire RO, and passes S bits of O-dependent advice to A_1
  - Becomes *non-uniform advice* when O instantiated
  - Separating S and T for more accurate *time-space tradeoffs* (e.g., for RAM attackers vs. circuits)
- ROM vs. standard model "separations" disappear! Concrete bounds in **ROM-AI** are meaningful against standard model **non-uniform** attackers!

Fixing Cracks in the Concrete

ROM-AI

Random function
O:[N]->[M]

$A = (A_0, A_1)$

A_0: S bits

advice

A_1 : T queries

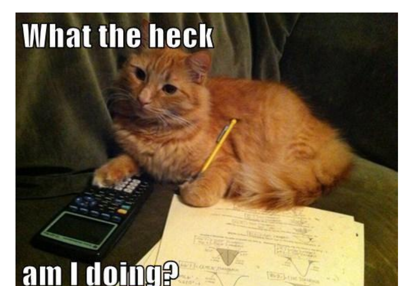
<u>ROM-AI methodology</u>: for "natural" applications,
Security in ROM-<u>**AI**</u> = Security in "standard model"
against <u>**non-uniform**</u> attackers
(for the best instantiation of O)

Security against any <u>**preprocessing**</u> attacks

Handling Auxiliary Input?

Problem: *conditioned* on S-bit "leakage", values of random oracle are not random and independent

	Traditional ROM	ROM-AI
Lazy Sampling	☑	☒
Programmability	☑	☒
Distinguishing-to-Extraction	☑	☒



Handling Auxiliary Input: Pre-sampling [Unruh07]

- **Intuition**: *conditioned* on S-bit leakage, only "few" values of $O(x)$ are "heavily biased"
  - A_0 can pass these "few" values as advice to A_1
- The rest can be re-sampled fresh and independent of the leakage!
  - Lazy sampling, programmability, etc. all come back as long as avoid the "few" *pre-sampled* points

Handling Auxiliary Input: Pre-sampling [Unruh07]

$Pre^O = \{(x_1,y_1)\ldots,(x_P,y_P)\}$

Random Function $O:[N]\to[M]$	$\approx_\varepsilon$	Random Function $R:[N]\to[M] \mid Pre^O$

x

$y=O(x)$

x

$$y=\begin{cases} y_i, \text{ if } x=x_i \\ R(x), \text{ o.w.} \end{cases}$$

S bits about O, then T queries

Handling Auxiliary Input: Pre-sampling [Unruh07]

$\text{Pre}^O = \{(x_1,y_1)\ldots,(x_P,y_P)\}$

Random Function $O:[N]\text{->}[M]$	$\approx_\varepsilon$	Random Function $R:[N]\text{->}[M] \mid \text{Pre}^O$

- Pre^O can depend on S-bit "leakage" z about O
  - P is a free parameter optimized later (see below)
- But R is *random and independent* on z
- How big is ε?

[Unr07]: $\varepsilon < (ST/P)^{1/2}$

S bits about O, then T queries

Security of OWFs in ROM-AI

S bits, T queries

$|Pre^O| = P$

Random Function $O:[N]\to[N]$	$\approx_\varepsilon$	Random Function $R:[N]\to[N] \mid Pre^O$

$$\text{Advantage} < (ST/P)^{1/2} + P/N + T/(N-P)$$

$$P = (STN^2)^{1/3} \Rightarrow \boxed{\text{Advantage} < (ST/N)^{1/3} + T/N}$$

Does not match best generic attacks ☹ :

$$\boxed{\text{Advantage} > ST/N + T/N} \quad (\text{if } ST^2 < N)$$

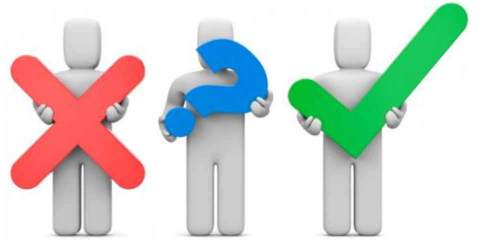
Our Motivating Question

Exact security for basic primitives?
(critical for ROM-AI methodology!)

Our Results

- Unruh's "pre-sampling conjecture" <u>*false*</u>
  - For many apps (OWFs, MACs, etc.), pre-sampling (as defined above) cannot give tight bounds
- <u>**New technique**</u>: Compression
  - Apply to get nearly tight ROM-AI bounds for OWFs, MACs, PRGs, PRFs, (salted) CRHFs ☺
  - Bounds much weaker than traditional ROM ☹ (because there <u>*are*</u> non-uniform attacks!)
- Salting provably defeats preprocessing!
  - Long-enough salt $\Rightarrow$ ROM-AI-sec. $\approx$ ROM-sec.
  - Possible way to reconcile theory and practice!

Our Results



- Salting provably defeats preprocessing!
  - Long-enough salt $\Rightarrow$ ROM-AI-sec. $\approx$ ROM-sec.
  - Possible way to reconcile theory and practice!

Improve Pre-sampling ?

S bits, T queries

$|Pre^O| = P$

Random Function		Random Function
$O:[N]\text{->}[N]$	$\approx_\varepsilon$	$R:[N]\text{->}[N] \mid Pre^O$

- [Unr07]: $\varepsilon < (ST/P)^{1/2}$
  - Can't get negl(n) security with P = poly(n) ☹
  - **Conj**: can get ε = negl(n) for P = poly(n) ☺
- Our result: $\varepsilon > \Omega(ST/P)$ ← tight! [CDGS17]
  - Unruh's conjecture false (in this generality)
- Is it enough to prove tight security?

via "dream pre-sampling"

Security of OWFs in ROM-AI

S bits, T queries

$|Pre^O| = P$

Random Function $O:[N]\to[N]$	$\approx\varepsilon$	Random Function $R:[N]\to[N] \mid Pre^O$

$$\text{Advantage} < (ST/P)^{\cancel{1/2}} + P/N + T/(N-P)$$

$$P = (STN^2)^{\cancel{1/3}\ 1/2} \Rightarrow \boxed{\text{Advantage} < (ST/N)^{\cancel{1/3}\ 1/2} + T/N}$$

Does not match best generic attacks ☹ :

$$\boxed{\text{Advantage} > ST/N + T/N}$$ (if $ST^2 < N$)

Our New Technique

Compression Paradigm [GT00,DTT10]!

High advantage $\Rightarrow$ Compressing RO

RO is impossible to compress

$\Rightarrow$ **Exact** security bounds

Challenge: need to compress by more than S bits!

(salted) CRHFs

$$\Pr[A_1^{O}(A_0(O), a) = (x, x') \text{ s.t. } x \neq x',\ O(a,x) = O(a,x')]$$

Number of salts a

$$= O(S/K + T^2/M)$$

Optimal: can hardwire S collisions inside advice $A_0(O)$!

Idea: compress $O(a,x')$ using indices $i,j \in [T]$ and $O(a,x)$

~~# of saved bits: |# of a s.t. A succeeds| x $(\log M - 2\log T)$~~
~~$= \varepsilon K \times \log(M/T^2)$~~

# of spent bits: S + description of set $\{a \mid A \text{ succeeds}\}$ =
$= S + \log(K \text{ choose } \varepsilon K)$

$\Rightarrow$ $\boxed{S > \varepsilon K \log(\varepsilon M/eT^2)}$

$\Rightarrow$ $\boxed{\varepsilon = O(S/K + T^2/M)}$

The Order Issue

Consider 2 salts: $O(a_1,x_1) = O(a_1,x_1')$; $O(a_2,x_2) = O(a_2,x_2')$

Ideally, compress both $O(a_1,x_1')$ and $O(a_2,x_2')$

Problem: what if $A(z, a_1)$ would query $O(a_2,x_2')$??

(not so crazy because of advice z…)

Solution: run A on all salts a where he succeeds, and keep track of "salt-specific" indices i_a, j_a for the first collision (which exists!) on all such a's

ROM-AI Bounds for Basic Primitives

Primitive	Our ROM-AI Bound
OWF*	ST/N
PRG	$(ST/N)^{1/2}$
PRF	$(ST/N)^{1/2}$
MAC*	ST/N

* Length preserving for simplicity

Always Better than Pre-sampling ☺

Primitive	Our ROM-AI Bound	Pre-Sampling
OWF*	ST/N	$(ST/N)^{1/3}$
PRG	$(ST/N)^{1/2}$	$(ST/N)^{1/3}$
PRF	$(ST/N)^{1/2}$	$(ST/N)^{1/3}$
MAC*	ST/N	$(ST/N)^{1/3}$

* Length preserving for simplicity

Nearly Tight ☺

Primitive	Our ROM-AI Bound	Best Attack
OWF*	ST/N	$\text{Min}(ST/N, (S^2T/N^2)^{1/3})$
PRG	$(ST/N)^{1/2}$	$(ST/N)^{1/2}$
PRF	$(ST/N)^{1/2}$	$(ST/N)^{1/2}$
MAC*	ST/N	$\text{Min}(ST/N, (S^2T/N^2)^{1/3})$

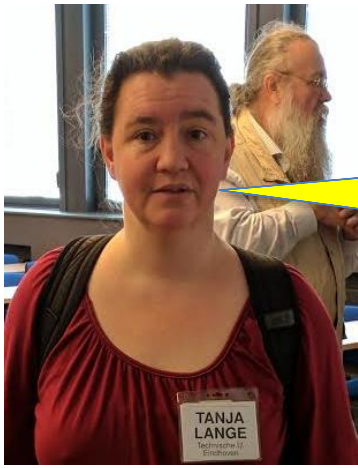
* Length preserving for simplicity

But Much Weaker than ROM ☹

Primitive	Our ROM-AI Bound	Traditional ROM
OWF*	ST/N	T/N
PRG	$(ST/N)^{1/2}$	T/N
PRF	$(ST/N)^{1/2}$	T/N
MAC*	ST/N	T/N

* Length preserving for simplicity

But Much Weaker than ROM ☹



Non-uniform attackers too strong!!!

Maybe we can all live in peace?

How to Defeat Preprocessing?

Extensively used in theory and practice:
Saw the magic for CRHFs already!

Chose random public salt after preprocessing;
Prepend as input to O

Security Bounds for Salting

$O: [K] \times [N] \rightarrow [M]$

Primitive	Salted ROM-AI Bound	Traditional ROM
OWF*	$T/N + ST/KN$	T/N
PRG	$T/N + (ST/KN)^{1/2}$	T/N
PRF	$T/N + (ST/KN)^{1/2}$	T/N
MAC*	$T/N + ST/KN$	T/N
CRHF	$T^2/M + S/K$	T^2/M

* Length preserving for simplicity

Security Bounds for Salting

O: [K] x [N] → [M]

Primitive	Salted ROM-AI Bound	Traditional ROM
OWF*	$T/N +$ [illegible] N	T/N
PRG	$T/N + (ST$ [illegible] $N)^{1/2}$	T/N
PRF	$T/N + (S$ [illegible] $)^{1/2}$	T/N
MAC*	$T/N + S$ [illegible] KN	T/N
CRHF	$T^2/M + S$ [illegible] K	T^2/M

* Length preserving for simplicity

Salting Provably helps!

At most n bits of salt yield ≈ **<u>same</u>** security
in ROM with auxiliary input
as without auxiliary input

↓

n-bit salt provably defeats preprocessing

Summary

$T \longrightarrow S, T$

Summary

- ROM-AI is the new “cool kid” in town !
  - very clean: just S and T !
  - addresses both theory (non-uniformity) and practice (preprocessing)
  - non-trivial, yet elegant proofs
  - *1000's of ROM papers need re-evaluation* !
- Obfuscation without the mess !

Thanks!

Your proposal is written with clarity and conviction.
Send it up to legal for obfuscation.

Follow-Up Work [CDGS17]

- Optimal Pre-Sampling Error ST/P
  - Improves $(ST/P)^{1/2}$ [Unruh07]
  - Gives tight bounds for indistinguishability apps
- New pre-sampling for unpredictability apps
  - Matches compression for all current apps
- Salting generically defeats preprocessing
- Random Permutation and Ideal Cipher

  with Auxiliary Input

Limitation of Pre-sampling

Random Function O:[N]-> {0,1}	$\approx_\varepsilon$	Random Function R:[N]-> {0,1} | Pre^O

$\$A=(A_0,A_1)$ " $|\mathrm{Pre}|=P$

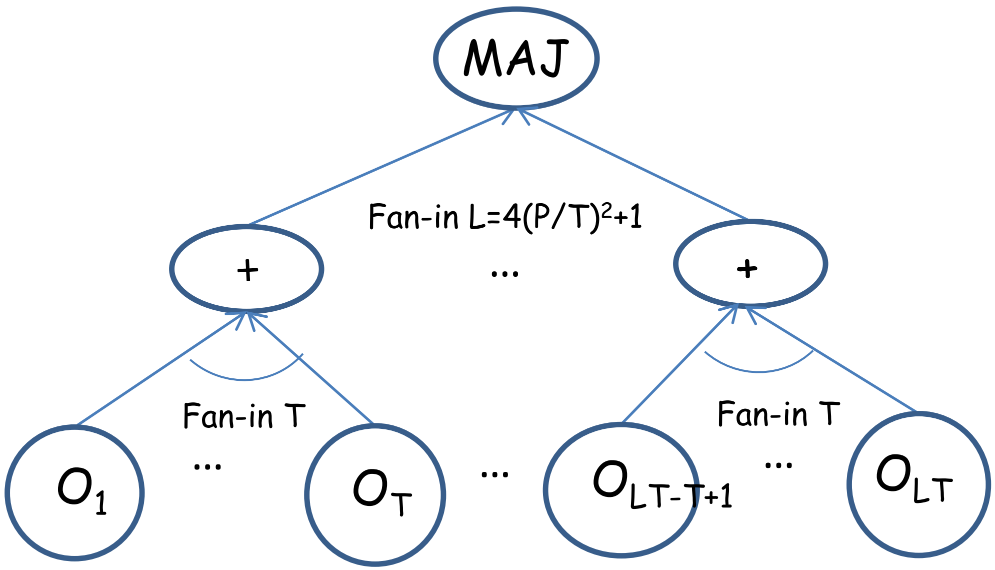
$\Pr[A_1^O(A_0(O))=1] - \Pr[A_1^{R|\mathrm{Pre}}(A_0(O))=1] > 1/24P$

$> 1/2 + 1/3L^{1/2}$ $\quad$ $<= 1/2 + P/2L$ $\quad$ $L= 4P^2+1$

$A_0(O)=\mathrm{Marj}(O_1,\ldots,O_L)$ $\quad$ $A_1=1$ if $A_0(O)= O_i$ where $i\sim[L]$

Extending to large S,T

Extending to large T: xor first



Extending to large S: repeat on disjoint inputs