A Parallel Repetition Theorem for Any Interactive Argument

Or

On the Benefits of Cutting Your Argument Short

Iftach Haitner
Microsoft Research New England
• Motivating examples for the question: 
  **Does parallel repetition improve security?**

• Our result

• Proof’s sketch
Example #1 – CAPTCHAS

CAPTCHAS – Aim to distinguish human beings from a machine.
Used to fight spamming, denial of service, ...

Basic task –

Not hard enough (easy to guess with probability 1/36)

Amplification via “sequential repetition”
Improves security (to any degree)

Problem: impractical, too much time

Amplification via “Parallel repetition”

By how much (if at all) does parallel repetition improve security?
Example #2 – Commitment Schemes

Commit stage

Reveal stage
Example #2 - Commitment Schemes

Reveal stage

Security properties:

**Hiding:** R learns nothing about m during commit stage

**Weakly binding:** S cannot decommit to two different values with ”too high” probability

- More “powerful” than encryption
- Can have statistical hiding
- Extremely useful

By how much (if at all) binding is improved?
Goal – Hardness Amplification

Starting point – A protocol/algorith with “weak security” – security holds with some probability

Goal – Amplify to fully secure protocol/algorith

Examples: one-way functions, PCP’s, CAPTCHAS, identification schemes, interactive arguments, ...

Real challenge – preserve other properties, in particular efficiency

Most natural approach is via parallel repetition

Does parallel repetition improve security?

Answer: (in general) No

Our result: Effectively, Yes
Interactive Arguments

\[ \Pr[V \text{ accepts in } (P^*, V)] \text{ is negligible} \]

- Typically, \((P, V)\) has additional functionality and other useful properties
- Realizes the security of significant types of systems
Amplification of Interactive Arguments

For any efficient $P^*$

$\Pr[V \text{ accepts in } (P^*, V)] < \varepsilon$

For any efficient $P^*$

$\Pr[V' \text{ accepts in } (P^*, V')]$ is negligible

Goal – a generic transformation that preserves other properties of $(P, V)$ (in particular, efficiency), and can be applied to any protocol.
Sequential Repetition

- No overlap between executions
- Verifier accepts if all sub-verifiers do
- Known to reduce the soundness error (to any degree, i.e., $\varepsilon^k$)
  - Since repetitions are independent
- Preserves most properties of the original protocol
- **Blows up** round complexity (# of communication rounds)
Parallel Repetition

- Interactions are done in parallel
- Verifier accepts if all sub-verifiers do
- Preserves round complexity.

Does it improve security?

*Does not work in general!*
The Counterexample of [Bellare et al. ’97]

- Safes are realized as commitment schemes
- Soundness error $\frac{1}{2}$
Both verifiers accept if $b_1 = b_2 \implies$ soundness error $\frac{1}{2}$
Can be extended to any (# of repetitions) $k$

[Pietrzak-Wikstrom ‘07] There exists a single protocol whose soundness error remains $\frac{1}{2}$ for any (poly) $k$
Can we improve security efficiently?

Parallel repetition does improve soundness in few special cases:

- 3-message protocols [Bellare-Impagliazzo-Naor ‘97]
- Public-coin protocols (i.e., verifier sends random coins as its messages) [Håstad-Pass-Pietrzak-Wikström ‘08] and [Chung-Liu ‘09]

- Also in Interactive proofs [Goldreich ‘97] and two-prover Interactive proofs [Raz ‘95]

The above does not apply to many interesting cases

Can we efficiently improve the security of general interactive arguments?
Our Result \[H '09\]

A simple modification of the verifier of any interactive argument, yields a protocol whose security is improved (to any degree) by parallel repetition.

In fact, we are going to “cripple” the original protocol, in a way that, paradoxically, enables repetition to improve security.
The Random Terminating Verifier

\( m \) rounds

\( \ldots \)

halts & accepts w.p. \( \frac{1}{4^m} \)

accept if \( V \) does
The Random Terminating Verifier

- \((P, \tilde{V})\) has, essentially, the same soundness guarantee
- Most properties of original protocol are preserved
- Applicable to many settings

1. \(\tilde{V}\) halts & accepts w.p. \(1/4m\)
2. \(\tilde{V}\) halts & accepts w.p. \(1/4m\)
3. \(\tilde{V}\) halts & accepts w.p. \(1/4m\)
4. Accept if \(V\) does
Why Does Random-Termination Help?

The transformation makes the verifier **less predictable**
Prevents cheating prover from using one verifier against the other
Beats the Counterexample

\[ \Pr[\tilde{V} \text{ accepts in } (P^*, \tilde{V})] = \frac{9}{32} < \frac{1}{2} \]
Proof’s Overview

Assume for any efficient $P^*$
(1) $\Pr[V\text{ accepts in }(P^*, V)] < \epsilon$

Prove for any efficient $P^{(k)*}$
(2) $\Pr[V\text{ accepts in }(P^*, V)] < \epsilon^{(k)} \subseteq \epsilon^k$

Proof by reduction –
Assuming $P^{(k)*}$ contradicts (2)
build $P^*$ that contradicts (1)

$\epsilon$ is much larger than $\epsilon^k$, thus an averaging argument would not be enough

The proof “almost” works for any interactive argument

$V$ accepts in $(P^*, V) \leftrightarrow P^*$ “wins”
Defining $P^*$

$p(k)^*$

$V_{-i}$

$V_1$

$V_i$

$V_k$

$i$ chosen at random
If succeeded, do the same for the second round
Does such \( q_{-i} \) always exist?

W.h.p, over \( q_i \), a noticeable fraction of the \( q_{-i} \) are “good”

\[
\Pr[q_i \text{ wins} | q_{-i} ] \geq (1 - 1 / 2m) \cdot \varepsilon^{(k)}
\]

**Proposition:** Let \( W \) be an event over \( X \) many candidates, and for large enough \( a \)
\( q \), w.h.p. over \( i \leftarrow [k] \) and \( x \leftarrow X_i \)

**Given** \( q_i \), **find** \( q_{-i} \) such that

\[
\Pr[P^{(k)*} \text{ wins} ] \geq (1 - 1 / 2m) \cdot \varepsilon^{(k)}
\]
Estimating $\alpha$

Estimate $\alpha$ \((= \Pr[P^{(k)\ast}\text{ wins} | q_i, q_{-i}])\) as the fraction of successful, random, continuations (i.e., \(P^{(k)\ast}\text{ wins} – \text{ all sub-verifiers accept}\))

If \(V\) is public coin, sampling random continuations is easy

Sampling might be infeasible for arbitrary \(V\) – As hard as finding a random preimage of an arbitrary (efficient) function. \textbf{This is why parallel repetition fails}
The Random Terminating Case

We sample random continuations, conditioned that \( \tilde{V}_i \) halts after first

Still hard to sample
$\alpha'$ Approximates $\alpha$ Well

$\alpha' = \Pr[P(k)^* \text{ wins} \mid (q_i, q_{-i}) \text{ & } \tilde{V}_i \text{ halts after first round}]$

Since many of the $\tilde{V}_j$'s are expected to halt after the first round

$\Rightarrow \alpha' \preceq \alpha$ for a random $i$

**Proposition:** Let $W$ be an event over $X = (X_1, \ldots, X_k)$, then

$\Pr[W \mid X_i = x] \preceq \Pr[W]$ w.h.p. over $i \leftarrow [k]$ and $x \leftarrow X_i$
More Details
Estimate $\alpha = \Pr[P^{(k)}^* \text{ wins } | q_i, q_{j_i}]$

- Sample random continuations of all verifiers.
- Random coins of the $j$-th verifying verifier.

For the emulated verifiers, as hard as finding a random second pre-image of a function!

- Feasible for (arbitrary) emulated verifiers.
- Impossible for (arbitrary) real verifier (even for unbounded sampler).
- Feasible for the real random termination verifier.

Find $q_{j_i}$ such that

$$\Pr[P^{(k)}^* \text{ wins } | q_i, q_{j_i}] \geq (1 + \frac{1}{2m}) \epsilon$$
Problem: threshold sensitivity

Solution: follows “Smooth sampling” approach of Håstad et al.: $P^*$ Samples many $(r_{-i}, r^2, ..., r^m)$ (all protocol’s random coins), and chooses $r_{-i}$ as the prefix of first successful execution ($P^*$ wins).

- Proof w.r.t. $\alpha$ still goes through
- The probability that $r_{-i}$ is picked, is proportional to $\alpha(r_{-i})$
- Hence, proof still go through w.r.t. $\alpha'$

- The original proof can be fixed, using soft thresholds

$P^*$ picks the first $r_{-i}$ s.t.

$\alpha'(r_{-i}) = \Pr[P^{(k)^*} \text{ wins} | (r_i, r_{-i}) \& V_i \text{ halts after first round}] > (1 - 1/2m)\epsilon^{(k)}$, where $\alpha'(r_{-i})$ is estimation for $\alpha(r_{-i}) = \Pr[P^{(k)^*} \text{ wins} | r_i, r_{-i}]$
Summary

• Parallel repetition may not improve security
• Does improve security of a slight variant of any protocol
• Main reason, the modified verifier is unpredictable
• Useful for many settings

Main open question:
• Can this proof technique be applied to other settings