PRGs to PRPs!

Outline
- Recap
- Game based Diffs
- More Stretch PRGs (BM)
- Hybrid Args
- PRFs (66M) * if time!
- Wrap up / LB
Recap!

 yesterday! Blum-Micali today!

OWFs ➔ PRGs (1-bit stretch) ➔ PRGs (poly (λ) stretch) ➔ GGM

≤

Block ciphers ➔ Luby-Rackoff ➔ PRFs

Def: A PRG \( G : S \rightarrow R \) is a deterministic, poly-time algorithm that, given as input a seed \( s \in S \) (a seed space), outputs an \( r \in R \) (output space).

A PRG \( G \) is secure if for all efficient adversaries \( A \),

\[
\left| \Pr\left[A(r) = 1 \mid r \in G(s)\right] - \Pr\left[A(r) = 1 \mid r \in R\right] \right| \leq \text{negl}(\lambda)
\]

where the probability space is over the random choice of \( r, s \) and randomness of the adversary.

Last Lecture: We saw a construction of a secure PRG \( G : \mathcal{X}_0,13^n \rightarrow \mathcal{X}_0,13^{n+1} \) (\( S = \mathcal{X}_0,13^n \), \( R = \mathcal{X}_0,13^{n+1} \)) with a 1-bit stretch from a OWF using Goldreich-Levin.

Today: We will take a PRG \( G : \mathcal{X}_0,13^n \rightarrow \mathcal{X}_0,13^{2n} \) and construct a PRG \( G' : \mathcal{X}_0,13^n \rightarrow \mathcal{X}_0,13^{2\lambda(n)} \) with arbitrary poly stretch.
Reframing PRG Security as a Game

In the security definition above, the adversary $A$ acts as a distinguisher between two distributions, $\exists r \in \mathcal{R}$ and $\exists G(s \uparrow s)$. We can equivalently reframe the security definition by treating $A$ as an interactive algorithm which interacts with a challenger $C$ and at the end of the interaction outputs a bit $b'$.

We define two experiments,

- In Experiment 0, the challenger samples $s \leftarrow S$, $r \leftarrow G(s)$, then sends $r$ to $A$.
- In Experiment 1, the challenger samples $r \leftarrow \mathcal{R}$, sends $r$ to $A$.

For $b \in \{0, 1\}$, let $W_b$ be the event that $A$ outputs 1 in Exp $b$.

The advantage $A$ has in the PRG Security Game is

$$PRG[A, G] := | \Pr[W_0] - \Pr[W_1]|$$

Where the probability space is over the random choices of challenger and $A$. A PRG $G$ is secure if for all efficient adversaries $A$,

$$PRG[A, G] \leq \text{negl}(\lambda)$$

Identical to the prior definition.
Building a PRG with more stretch

Let $G : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ be a secure PRG. We construct a PRG $G' : \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$, where $\ell$ is a poly, as follows:

**Idea:** Let’s sequentially compose PRG evaluations and output the extra bits!

$S_0 \rightarrow G \rightarrow S_1 \rightarrow G \rightarrow S_2 \rightarrow \ldots \rightarrow S_{\ell(n)-1} \rightarrow G \rightarrow S_{\ell(n)}$

$G'(s \in \{0,1\}^n)$:

1. $S_0 \leftarrow S$
2. For $i \in \{1,2,\ldots, \ell(n)\}$:
   - $(S_i, b_i) \leftarrow G(S_{i-1})$
3. Output $b_1 b_2 \ldots b_{\ell(n)}$

**Theorem:** If $G$ is a secure PRG, then $G'$ is a secure PRG.

**Lemma 1:** $G'$ is polytime.

Since $G$ is a PRG, let $t(n)$ be the poly runtime of $G$. $G'$ runtime is $\ell(n) \cdot t(n) + O(\ell(n))$. Since $\ell(n)$ is poly, $G'$ is polytime.

**Lemma 2:** For every PRG Adv $A$ that plays the PRG Security Game with respect to $G'$, there exists a PRG Adv $B$ that plays the PRG security game with respect to $G$, such that

$$\text{PRG adv}[A, G'] = \ell(n) \cdot \text{PRG adv}[B, G]$$
Hybrid Arguments

Issue: informally, the definition of a PRG tells us that $G(s)$ looks random if $s$ is random. But here we evaluate $S_i \leftarrow G(S_{i-1})$ where $S_{i-1}$ is a PRG eval (not random). How do we leverage that $G$ is secure PRG?

Solution: Apply the def of a PRG one execution at a time!

We define a sequence of "Hybrid Games" for an efficient PRG adversary $A$ for $G'$ such that each game behaves identically to the PRG security game, except that $r := b_1, b_2, \ldots, b_{\ell(n)}$ is sampled differently by the challenger.

Hybrid 0 := Exp 0

$b_1 \xleftarrow{\$} \{0,1\}^3 \quad S_0 \xleftarrow{\$} \{0,1\}^3 \rightarrow [G] \rightarrow s_i \rightarrow [G] \rightarrow S_2 \rightarrow \cdots \rightarrow S_{\ell(n)-1} \rightarrow [G] \rightarrow S_{\ell(n)}$

Hybrid 1

$b_1 \xleftarrow{\$} \{0,1\}^3, S_i \xleftarrow{\$} \{0,1\}^3 \rightarrow [G] \rightarrow S_2 \rightarrow \cdots \rightarrow S_{\ell(n)-1} \rightarrow [G] \rightarrow S_{\ell(n)}$

\vdots

Hybrid $j$

$b_1, \ldots, b_j \xleftarrow{\$} \{0,1\}^3, S_j \xleftarrow{\$} \{0,1\}^{3 \cdot j} \rightarrow [G] \rightarrow S_{j+1} \rightarrow \cdots \rightarrow S_{\ell(n)-j} \rightarrow [G] \rightarrow S_{\ell(n)}$

\vdots

Hybrid $\ell(n) :=$ Exp 1

$b_1, \ldots, b_{\ell(n)} \xleftarrow{\$} \{0,1\}^{3 \cdot \ell(n)}$
Define for $i \in \{0, \ldots, \ell(n)\}$, $p_i$ as the probability $A$ outputs 1 in hybrid game $i$. Notice that

$$\text{PRGadv}[A,G'] := |\Pr[W_0] - \Pr[W_1]| = |p_0 - p_{\text{ran}}|$$

**Construction**

We will construct an efficient adv $B$ that plays the PRG Security Game with respect to $G$ that is a wrapper around $A$.

- Receive $r \in \{0,1\}^{n+1}$ from Challenger
- Sample $w \in \{0,1\}^n$
- Sample $b_1, \ldots, b_{w-1} \in \{0,1\}$
- Parse $r$ as $(S_w, b_w)$
- For $i \in \{w+1, \ell(n)\}$
  - $(S_i, b_i) \leftarrow G(S_{i-1})$
- Send $b, b_2, \ldots, b_{\ell(n)}$ to $A$ and output what $A$ outputs.
Analysis:
Conditioned on $w = j$ for $j \in \mathcal{E}_1, \ldots, \ell(n)^3$. In Exp 0 of $B$'s PRG Game, $r \leftarrow G(s \oplus s)$. Thus, $B$ identically simulates the challenger in Hybrid $j-1$ to $A$. In Exp 1 of $B$'s PRG Game, $r \leftarrow \mathbb{E}_0, \mathbb{B}^{n+1}$. Thus, $B$ identically simulates the challenger in Hybrid $j$ to $A$. Therefore, $\Pr[W_0 | w = j] = p_{j-1}$ and $\Pr[W_1 | w = j] = p_j$.

Thus, $\text{PRG}_{\text{adv}}[B, G] = \left| \Pr[W_0] - \Pr[W_1] \right|$

$$= \left| \sum_{j=1}^{\ell(n)} \Pr[W_0 | w = j] \Pr[w = j] - \sum_{j=1}^{\ell(n)} \Pr[W_1 | w = j] \Pr[w = j] \right|$$

$$= \frac{1}{\ell(n)} \left| \sum_{j=1}^{\ell(n)} p_{j-1} - \sum_{j=1}^{\ell(n)} p_j \right| = \frac{1}{\ell(n)} \left| p_0 - p_{\ell(n)} \right| = \text{PRG}_{\text{adv}}[A, G']$$

Since we assumed $G$ is a secure PRG, $\text{PRG}_{\text{adv}}[B, G] \leq \text{neg}(\lambda)$. 

$\Rightarrow \text{PRG}_{\text{adv}}[A, G'] \leq \ell(n) \cdot \text{neg}(\lambda)$

must be $\text{neg}(\lambda)$

Thus, $G'$ is a secure PRG (Lemma 1 + Lemma 2)

Returning to our diagram, we have shown

$$\begin{align*}
\text{PRG with } & \quad \text{1 bit stretch} \quad \rightarrow \quad \text{PRG with } \quad \text{Blum-Micali} \\
& \quad \rightarrow \quad \text{PRG with } \quad \text{poly stretch} \quad \rightarrow \quad \text{PRF}
\end{align*}$$

Now, we will construct PRFs from a PRGs.
PRFs

A PRF is a deterministic algorithm $F: K \times X \rightarrow Y$ that takes as input a key $k \in K$, $x$ in an input space $X$ and outputs $y$ from an output space $Y$.

Intuitively, for a random key $k$, $F(k, \cdot) : X \rightarrow Y$ should “behave like” a random function from the space of functions $\text{Func}[X, Y]$.

**PRF Security Game**

Experiment $b \in \{0, 1\}$:

- Challenger samples $f \in \text{Func}[X, Y]$ as follows
  - If $b = 0$: $k \leftarrow K$, $f \leftarrow F(k, \cdot)$
  - If $b = 1$: $f \leftarrow \text{Func}[X, Y]$
- For $i \in \{1, \ldots, Q\}$, a polynomial number of queries
  - Adversary sends a query $x_i \in X$ to the chal
  - Chal responds with $y_i := f(x_i)$
- Adversary outputs a bit $b' \in \{0, 1\}$
Similarly, define \( W_0 \) be the event that \( A \) outputs 1 in \( \text{Exp}_b \). We define the advantage of an efficient adversary \( A \) with respect to \( F \) as

\[
\text{PRF}_{\text{adv}}[A, F] := \left| \Pr[W_0] - \Pr[W_1] \right|
\]

An adversary is a \( Q \)-query PRF adversary if \( A \) makes at most \( Q \) queries.

A PRF \( F \) is secure if for all efficient adv \( A \),

\[
\text{PRF}_{\text{adv}}[A, F] \leq \text{negl}(\lambda)
\]

---

**PRGs \( \rightarrow \) PRFs**

Here we give a construction of a PRF and provide intuition about its security proof (see 4.6 of GCAC for more details). We are given a PRG \( G : S \to S^2 \). For example, we can use the previous construction to obtain \( G' : \Sigma_0,1^\lambda \to \Sigma_0,1^{2\lambda} \) from a PRG \( G : \Sigma_0,1^\lambda \to \Sigma_0,1^{\lambda+1} \).

Looks a node in a binary tree.
We can construct a PRF $F : S \times \mathbb{Z}_p^2 \rightarrow S$ as follows:

Construct an evaluation tree by selectively composing the PRG evals. For an input $x := b_1 b_2 \ldots b_e$, evaluate the path selected by the bits.

More formally,

$$F(S, b_1 \ldots b_e):$$

$$t \leftarrow s$$

$$\text{for } i \in \{1, \ldots, e\}:$$

$$t \leftarrow G_{b_i}(t)$$

output $t$

Efficiency: $e$ evals of PRG $G$

Security: If $G$ is a secure PRG, then $F$ constructed above is a secure PRF.

- for every PRF adv $A$, we can construct a $Q$-query adv $B$ s.t.

$$\text{PRF}_{\text{adv}}[A, F] = eQ \cdot \text{PRG}_{\text{adv}}[B, G]$$
Sketch of Argument

- Given an adversary $A$ that plays the PRF game, we want to construct an adversary $B$ that plays the PRG game with respect to the parallel PRG.
- We will proceed with a Hybrid Argument:
  - In the hybrid games, we can progressively replace each level of the tree with random seeds.

![Diagram of hybrid games]

- The PRG adversary $B$ will need to simulate the challenger in the PRF Game when interacting with the PRF adversary $A$.
- However, each level of the tree is exponentially sized. So, how can $B$ remain efficient?
  - Since $A$ is $Q$-query bounded, $B$ only needs to simulate at most $Q$ (a polynomial) number of paths of the tree!
  - For more details, read the section in the book!
**Symmetric Lecture Conclusion**

**OWF** \(\rightsquigarrow\) **PRG** \(_{+1}\) \(\rightsquigarrow\) **PRG** \(_{\text{poly}}\) \(\rightsquigarrow\) **PRF**

- GL \(\leftarrow\) By definition
- BM \(\leftrightarrow\) Truncate
- GGM \(\leftarrow\) CTR Mode
- \text{Luby Rackoff/Fiestal Networks} \(\leftarrow\) Switching Lemma

- \text{Just a PRF but a permutation}
- \text{Just a bonus if time}

**PRF \rightarrow PRP/BC (LR/Fiestal Networks)**

1. Let \(F : k \times x \rightarrow x\) be a secure PRF where \(x = \{0, 1\}^n\) (hence \(x, \oplus x\) is a valid op)
2. Define a round fn \(\Phi_k : x^2 \rightarrow x^2\), \((a, b) \mapsto (b, a \oplus F(k, b))\)
3. \text{flip} : \(x^2 \rightarrow x^2\), \((a, b) \mapsto (b, a)\)

Note \(\Phi_k\) is a permutation, \(\Phi_k^{-1} = \text{flip} \circ \Phi_k \circ \text{flip}\)

We can construct a PRP \(P : k^3 \times x^2 \rightarrow x^2\) as follows:

\[P((k_1, k_2, k_3), \cdot) := \Phi_{k_3} \circ \Phi_{k_2} \circ \Phi_k,\]

The inverse is also efficiently computable given the key:

\[P^{-1}((k_1, k_2, k_3), \cdot) := \Phi_k^{-1} \circ \Phi_{k_2}^{-1} \circ \Phi_{k_3}^{-1} = \text{flip} \circ \Phi_k \circ \Phi_{k_2} \circ \Phi_{k_3} \circ \text{flip}\]