Cryptographic Concepts for Finite Automata

Synopsis.

- One-Way Functions and Hardcores
- Pseudorandom Generators
- Interactive Proof Systems
- Primeimmunity
Course Schedule: 16 Weeks
Subject to Change

- Week 1: Basic Computation Models
- Week 2: NP-Completeness, Probabilistic and Counting Complexity Classes
- Week 3: Space Complexity and the Linear Space Hypothesis
- Week 4: Relativizations and Hierarchies
- Week 5: Structural Properties by Finite Automata
- Week 6: Stype-2 Computability, Multi-Valued Functions, and State Complexity
- Week 7: Cryptographic Concepts for Finite Automata
- Week 8: Constraint Satisfaction Problems
- Week 9: Combinatorial Optimization Problems
- Week 10: Average-Case Complexity
- Week 11: Basics of Quantum Information
- Week 12: BQP, NQP, Quantum NP, and Quantum Finite Automata
- Week 13: Quantum State Complexity and Advice
- Week 14: Quantum Cryptographic Systems
- Week 15: Quantum Interactive Proofs
- Week 16: Final Evaluation Day (no lecture)
YouTube Videos

- This lecture series is based on numerous papers of T. Yamakami. He gave conference talks (in English) and invited talks (in English), some of which were video-recorded and uploaded to YouTube.
- Use the following keywords to find a playlist of those videos.
- **YouTube search keywords:**
  Tomoyuki Yamakami  conference  invited talk playlist

Conference talk video
Main References by T. Yamakami


I. One-Way Functions and Pseudorandom Generators

1. Cryptographic Primitives
2. (Strongly) One-Way Functions
3. Weakly One-Way Functions
4. Natural Candidates for OWFs
5. Pseudorandomness
6. Polynomial-Time Indistinguishability
7. Generating Pseudorandom Bits
8. Pseudorandom Generators
9. PEGs Versus OWFs
Cryptographic Primitives

- If we want to build a complex cryptographic system, it is necessary to break it into small building blocks.
- **Primitives** are such building blocks that support complex cryptographic systems.

- Zero-knowledge proof
- Oblivious transfer
- Bit commitment
- Hardcore predicate
- One-way function
- Pseudorandom generator

...Etc.
What are One-Way Functions?

- Yao (1982) first considered the notion of one-way function.
- Intuitively, a (strongly) one-way function $f(x)$ is
  - **Easy** to compute from its inputs $x$, but
  - **Hard** to invert from its images $y=f(x)$ (i.e., find $x' \in f^{-1}(y)$).

$$\text{Prob}_{x,A}[f(A(f(x),1^n)) = f(x)] < 1/p(n)$$ for any efficient algorithm $A$, any polynomial $p$ and almost all sizes $n$. 

---

Input $x$ → Computes $f(x)$. Easy!

Input $(y,1^n)$ → Computes $x \in \{0,1\}^n$ s.t. $f(x)=y$. Hard!
Probabilistic Poly-Time Algorithms (revisited)

• Recall the model of probabilistic Turing machine from Week 2.

• We informally use the term “probabilistic polynomial-time algorithm” to mean “probabilistic polynomial-time Turing machine.”
Probabilistic Computation of PTMs (revisited)

- A PTM produces accepting/rejecting computation paths.

\[
\Pr_M[M(x) = 1] > \frac{1}{2} \quad \text{M accepts } x
\]

\[
\Pr_M[M(x) = 0] \geq \frac{1}{2} \quad \text{M rejects } x
\]
Consider a function \( f : \{0,1\}^* \rightarrow \{0,1\}^* \).

\( f \) is (strongly) one-way if

1. (easy to compute) there is a deterministic polynomial-time algorithm that computes \( f \), and

2. (hard to invert) for every probabilistic polynomial-time algorithm \( A \), every positive polynomial \( p \), and for all sufficiently large length \( n \),

\[
\Pr_{A,U_n} \left[ A(f(U_n), 1^n) \in f^{-1}(f(U_n)) \right] < \frac{1}{p(n)}
\]

\( U_n \) is a random variable ranging over \( \{0,1\}^n \).
(Strongly) One-Way Functions II

\[
\Pr\left[ A(f(U_n), 1^n) \in f^{-1}(f(U_n)) \right] < \frac{1}{p(n)}
\]

- **This formula** means:
  - the probability that, on input \((y, 1^n)\) with \(y \in \{ f(x) \mid x \in \{0,1\}^n \}\), algorithm A finds \(x'\) satisfying \(f(x') = y\) is polynomially small.
- Note that there are possibly many \(x'\) satisfying \(f(x') = y\).
- So, it suffices to find at least one of them probabilistically.
Weakly One-Way Functions

• There is another notion of one-way function.

• f is weakly one-way if
  1. (easy to compute) there is a deterministic polynomial-time algorithm that computes f, and
  2. (slightly hard to invert) there exists a polynomial p such that, for every probabilistic polynomial-time algorithm A and all sufficiently large length n,

\[ \Pr_{A,U_n} \left[ A(f(U_n),1^n) \notin f^{-1}(f(U_n)) \right] > \frac{1}{p(n)} \]

• (Claim) A strongly one-way function exists \( \iff \) a weakly one-way function exists. [Yao (1982)]
Natural Candidates for OWFs I

• Unfortunately, we do not know whether or not one-way functions (OWFs) exist.
• However, we have several good candidates for OWFs.
• The RSA function
  – with index set \((N,e)\), where \(N\) is a product of two \((1/2 \cdot \log_2 N)\)-bit primes \(P\) and \(Q\), and \(e\) is an integer smaller than \(N\) and relatively prime to \((P-1)(Q-1)\).
  \[
  RSA_{N,e}(x) = x^e \mod N
  \]
• The Rabin function
  – with a similar condition to the above,
  \[
  Rabin_{N}(x) = x^2 \mod N
  \]
Natural Candidates for OWFs  II

- The DLP (discrete logarithm problem) function
  - with index set (P, G), where P is a \((1/2 \log_2 N)\)-bit prime P and a primitive element G in the multiplicative group modulo P,
  \[
  DLP_{P,G}(x) = G^x \mod P
  \]

- Open Problems
  - Prove or disprove that the aforementioned candidates are truly one-way functions.
  - More generally, prove or disprove the existence of one-way functions.
Pseudorandomness

• **Blum** and **Micali** (1984) considered how to generate a sequence of bits whose next bit is hardly predicted by even powerful adversary.

• In contrast, **Yao** (1982) considered a sequence that no adversary distinguishes from a uniformly random sequence with a small margin of error.

• Let $X = \{ X_n \}_{n \in \mathbb{N}}$ be an ensemble of random variables indexed by $\mathbb{N}$.

• For example, consider an infinite series of fair coins. For each $n \in \mathbb{N}$, we define $X_n$ to be the outcome of the flip of the $(n+1)$th coin.
Polynomial-Time Indistinguishability

• We start with “indistinguishability” of two ensembles of random variables.

• Two ensembles $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$ are indistinguishable in polynomial time (or computationally indistinguishable) if

  for every probabilistic polynomial-time algorithm $M$, every positive polynomial $p$, and all sufficiently large length $n$,

  $$\left| \Pr[M(X_n, 1^n) = 1] - \Pr[M(Y_n, 1^n) = 1] \right| < \frac{1}{p(n)}$$

The probability of distinguishing between $X_n$ and $Y_n$ is polynomially small.
Generating Pseudorandom Bits

short truly random bits

computer algorithm

long pseudorandom bits

truly random bits

“uniform” ensemble

true randomness

hard to distinguish them

??
Pseudorandom Generators

• An ensemble $X = \{ X_n \}_{n \in \mathbb{N}}$ is called pseudorandom if there is a uniform ensemble $U = \{ U_{l(n)} \}_{n \in \mathbb{N}}$ such that $\{ G(U_n) \}_{n \in \mathbb{N}}$ and $U$ are polynomial-time indistinguishable, where $l: \mathbb{N} \rightarrow \mathbb{N}$ is a fixed function.

• A pseudorandom generator $G$ is a deterministic polynomial-time algorithm satisfying the following two conditions:

  1. (expansion) there is a function $l: \mathbb{N} \rightarrow \mathbb{N}$ (called the expansion/stretch factor of $G$) such that $l(n) > n$ for all $n \in \mathbb{N}$ and $|G(s)| = l(|s|)$ for all $s \in \{0,1\}^*$, and

  2. (pseudorandomness) the ensemble $\{ G(U_n) \}_{n \in \mathbb{N}}$ is pseudorandom.

$U_{l(n)}$ is chosen uniformly at random.
PRGs Versus OWFs

• Let \( G : \{0,1\}^* \rightarrow \{0,1\}^* \) be a function with expansion factor \( l(n) = 2n \) (that is, \( |G(x)| = 2|x| \) for all \( x \in \{0,1\}^* \)).

• We define a function \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) by

\[
  f(x, y) = G(x)
\]

• (Claim) If \( G \) is a pseudorandom generator, then \( f \) is a strongly one-way function.

• Moreover, we can prove the following.

• (Claim) If there exists a one-way function, then a pseudorandom generator exists. [Håstad-Impagliazzo-Levin-Luby (1999)]
II. Hardcore Functions

1. What are Hardcore Functions?
2. Hardcore Predicates (or Functions)
3. Examples of Hardcores
4. Why are Hardcores so Useful?
What are Hardcore Functions?

- A hardcore function $P$ for a function $f$ is
  - Easy to compute from its inputs $x$, but
  - Hard to “predict” $P(x)$ from the images $f(x)$ of the function $f$ without knowing inputs $x$.

\[ |\text{Pr}_{x,A}[A(f(x),1^n) = P(x)] - 1/2^{l(n)}| < 1/p(n) \]

for any polynomial $p$ and almost all sizes $n$.

where $l(n)$ is the size function of $P$
Hardcore Predicates (or Functions)

- Let \( b : \{0,1\}^* \rightarrow \{0,1\} \) be a polynomial-time computable predicate (i.e., functions outputting 0 or 1).
- Let \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) be a function.

- \( b \) is a **hardcore predicate** (or a **hardcore**) of \( f \) if, for every probabilistic polynomial-time algorithm \( A \), every positive polynomial \( p \), and all sufficiently large \( n \),
  \[ \Pr[A(f(U_n) = b(U_n)) < \frac{1}{2} + \frac{1}{p(n)}] \]

- This means that, to predict the value \( b(s) \) from input \( f(s) \) is similar to choosing 0 or 1 at random.

- **Hardcores actually exist** for any strongly one-way function (assuming that one-way functions exist).
Examples of Hardcores

• There are known hardcore predicates for (strongly) one-way functions of a special form (explained below).

• Let $b: \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ be the (bitwise) inner-product-mod-2 function; that is, $b(x,r) = x \odot r \pmod{2}$.

• Example: $b(1011,1101) = 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 \pmod{2}$
  \[= 2 \pmod{2} = 0\]

• (Claim) Let $f$ be any strongly one-way function. Define $g$ as $g(x,r) = f(x)r$ (concatenation), where $|x| = |r|$. The predicate $b$ (defined above) is a hardcore of $g$. [Goldreich-Levin (1989)]
Why are Hardcores so Useful?

- Let \( f \) be any one-way permutation and let \( P \) be any hardcore predicate for \( f \).
- Define \( G(x) = f(x)P(x) \) (string concatenation).
- The definition of a hardcore says that we cannot predict the value \( P(x) \) from the value \( f(x) \) with high confidence.

Well-Known Result: unpredictability = pseudorandomness

Therefore, this function \( G(.) \) is a pseudorandom generator that stretches \( n \) bit seeds to \( n+1 \) bit strings.
III. Basic Cryptosystems

1. Public-Key Cryptosystems
2. Non-Interactive Bit Commitment
Private-Key/Public-Key Encryption Schemes

A schematic of an encryption scheme

Symmetric/asymmetric key

Private/public key

A(n) secure/insecure channel

Encryption key

Decryption key

Encryption

Decryption

An insecure channel

Plaintext

Ciphertext

Alice

Bob

Encryption

Decryption

Ciphertext

Ciphertext
Non-Interactive Bit Commitment

• In a non-interactive bit commitment scheme, a committer (Alice) and a verifier (Bob) communicate with each other and satisfy the following conditions.

  ➢ (hiding) In the commit phase, Alice commits to a single bit $b$ and sends some information $z$ to Bob so that Bob cannot recover $b$ from $z$,

  ➢ (binding) In the opening (or reveal) phase, Alice reveals her bit $b$ and Bob checks if $b$ is the correct committed bit from $z$. We require that Alice cannot cheat Bob by revealing a different bit.
IV. Interactive Proof Systems

1. What is an Interactive Proof?
2. Interactive Proof Systems
3. Constant-Space Interactive Proofs
4. Private Coins vs. Public Coins
5. One-Way Functions for 1-Tape Machines
What is an Interactive Proof?

• An interaction between two (or more) parties has been studied in many cryptographic contexts.

• Goldwasser, Micali, and Rackoff (1989) studied a series of interactions between a prover (who presents a proof) and a verifier (who verifies the proof).

• This gave rise to a notion of interactive proof (IP) systems.

• In an IP system, a prover P sends a proof (either correct or wrong) and a verifier V checks if the proof is indeed correct.
Intuitive Definition

- A language $L$ has an **IP system** $\iff$ there exists a verifier $V$ that satisfies the following two conditions:
  1. For every $x \in L$, there exists a honest prover $P$ such that $V$ accepts a proof from $P$ with probability at least $2/3$; and
  2. For every $x \notin L$, $V$ rejects any proof from any (possibly malicious) prover with probability at least $2/3$.

A proof is a piece of information.

- Let me judge the correctness of your proof.
- Believe me. This is a correct proof.
- Accepts with probability $\geq 2/3$
- Rejects with probability $\geq 2/3$
Underlying Machine Model

- **Dwork-Stockmeyer IP system** is illustrated as follows.

![Diagram](image-url)
Interactive Proof Systems

• Let \((P,V)\) be a pair of prover \(P\) and verifier \(V\).
• Let \(L\) be a language over alphabet \(\{0,1\}\).

\((P,V)\) is an interactive proof system for \(L\) if
  
  ▪ \(V\) is a specified probabilistic machine,
  
  ▪ \((P,V)\) satisfies the following conditions:
    1. (completeness) for every \(x \in L\),
       \[
       \Pr[(P,V)(x) = 1] \geq \frac{2}{3}
       \]
    2. (soundness) for any \(x \notin L\) and any prover \(B\),
       \[
       \Pr[(B,V)(x) = 1] \leq \frac{1}{3}
       \]
Constant-Space Interactive Proofs

- **Dwork and Stockmeyer** (1992) considered interactive proof (IP) systems with 2-way probabilistic finite automata (2pfa’s).

- **Major advantages**: we can prove certain separation results that are impossible (at least at present) to obtain for polynomial-time or logarithmic-space bounded IP systems.

  \[ \text{IP(}\langle\text{restrictions}\rangle\text{)} = \text{the class of all languages that have IP systems satisfying the restrictions given in } \langle\text{restrictions}\rangle. \]

- **For example**:
  - \[ \text{IP(2pfa,poly-time)} = \text{the class of all languages that have IP systems with 2pfa verifiers running in expected polynomial time}. \]
Private Coins vs. Public Coins

• In an IP system, a verifier obtains random bits (by flipping coins) and decides his next actions. The verifier keeps those random bits secretly. A prover has no way knowing those bits of the verifier.

• This situation is described as the verifier playing with “private coins.”

• In contrast, if the verifier reveals his random bits to the prover every time, then this situation is described as the verifier playing with “public coins.”

• If the verifier uses “public coins” instead of “private coins,” then we write $\text{AM}(\langle \text{restriction} \rangle)$ in place of $\text{IP}(\langle \text{restriction} \rangle)$.

“AM” stands for “Arthur-Merlin game.”
Known Results

• **Dwork** and **Stockmeyer** (1992) obtained the following results.

• (**Claim**)  
  1. $2\text{PFA} \subseteq \text{AM}(2\text{pfa}) \subseteq \text{IP}(2\text{pfa},\text{poly-time}) \subseteq \text{IP}(2\text{pfa})$  
  2. Pal = \{ $x \in \{0,1\}^* \mid x = x^R$ \} is in IP(2pfa) but not in AM(2pfa).  
  3. Center = \{ $u1v \mid u,v \in \{0,1\}^*, |u| = |v|$ \} is in AM(2pfa) but not in 2PFA.

• (***) We will return to this topic in Week 13.
Track Notation (revisited)

• To describe the notion of one-way function in the 1-tape linear-time model, we need to introduce a “track” notation

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \begin{bmatrix}
  x_1 \\
  y_1
\end{bmatrix} \begin{bmatrix}
  x_2 \\
  y_2
\end{bmatrix} \ldots \begin{bmatrix}
  x_n \\
  y_n
\end{bmatrix}, \text{ where } x = x_1 x_2 \ldots x_n \text{ and } y = y_1 y_2 \ldots y_n
\]

• Even if \(|x| \neq |y|\), we want to use the same notation to express

\[
\begin{bmatrix}
  x & x^{\#^k} \\
  y^{\#^k} & y
\end{bmatrix}
\]

if \(|x| = |y| + k \text{ and } k \geq 1 \text{ and } |x| + k = |y| \text{ and } k \geq 1\), respectively, where \# is a distinct “blank” symbol.
A total function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ is called one-way if
1. $f \in 1\text{-FLIN}$, and
2. there is no function $g \in 1\text{-FLIN}$ such that
   $$f\left(g\left(\left[ f(x) \right]_1^{\left| x \right|} \right)\right) = f(x)$$
for all inputs $x$.

When $f$ is length-preserving, the above equality can be replaced by $f(g(f(x))) = f(x)$.

**Theorem**: [Tadaki-Yamakami-Lin (2010)]

- There is no one-way function in 1-FLIN.

(*) In the next slide, we will see a proof sketch.
One-Way Functions for 1-Tape Machines II

- Recall 1-DLIN and 1-FLIN from Week 1, and 1-FLIN(partial) and 1-NLINMV from Week 6.

Proof Sketch:
- Assume by contradiction that a one-way function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ exists in 1-FLIN.
- Define $f^{-1}(\[y \ 1^n\]^T) = \{ \ x\#|y|-n \mid |x|=n, \ f(x) = y \} \text{ if } |y|\geq n; \ f^{-1}(\[y \ 1^n\]^T) = \{ \ x \mid |x|=n, \ f(x) = y \} \text{ otherwise.}$
- Clearly, $f^{-1} \in 1$-NLINMV.
- As seen in Week 6, since $1$-NLINMV $\subseteq_{\text{ref}} 1$-FLIN(partial), there is a refinement, say, $g$ of $f^{-1}$ in 1-FLIN(partial).
- We then construct a 1DTM computing $g$ in $O(n)$ time.
- Since $f^{-1} \subseteq_{\text{ref}} g$, $M$ converts $f$, a contradiction against our assumption.

QED
V. Pseudorandomness for Automata

1. Negligible Functions
2. C-Pseudorandomness
3. Examples of C-Pseudorandom Languages
Negligible Functions

- We apply pseudorandomness to finite automata.

- First, we need a notion of negligible function.

- A real-valued function \( h : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0} \) is negligible \( \iff \)
  - \( \forall p: \) positive polynomial, \( h(n) \leq 1/p(n) \) holds for all but finitely many numbers \( n \in \mathbb{N} \) (super-polynomially small).

- Example: \( h(n) = 1/2^n \), \( h'(n) = 1/n^{\log(n)} \)
Intuition: Pseudorandomness

- $A \triangle L$ denotes the symmetric difference $(A - L) \cup (L - A)$.

- Intuitively, the C-pseudorandomness of $L$ means: for any language $A \in C$ and for almost all $n$’s, 
  $$| (A \triangle L) \cap \Sigma^n |$$
  is “nearly” a half of $| \Sigma^n |$. (Fig.1)

- Equivalently: for any language $A \in C$ and for almost all $n$’s, 
  $$| A \cap (L \cap \Sigma^n) |$$
  is “nearly” equal to $| A \cap (\Sigma^n - L) |$. (Fig.2)
C-Pseudorandomness

- Let \( L \) be any language over \( \Sigma \) with \( |\Sigma| \geq 2 \).
- Let \( C \) be any language family.

- \( L \) is \( C \)-pseudorandom \( \iff \) for all \( A \in C \) over \( \Sigma \),
  \[
  h(n) = \left| \frac{|(A \Delta L) \cap \Sigma^n|}{|\Sigma^n|} - \frac{1}{2} \right| \]
  is negligible.

- (Claim) No language in \( C \) is \( C \)-pseudorandom.
C-Pseudorandomness II

- We may be focused on \( p \)-dense languages.

- A language \( L \) (over \( \Sigma \)) is weakly \( C \)-pseudorandom \( \iff \)
  - for all \( p \)-dense \( A \in C \) (over \( \Sigma \)),
  \[
  h'(n) = \text{def } |\frac{|(A \cap L) \cap \Sigma^n|}{|A \cap \Sigma^n|} - \frac{1}{2}| \text{ is negligible.}
  \]

- A language family \( D \) is (weakly) \( C \)-pseudorandom \( \iff \)
  - \( D \) contains a (weakly) \( C \)-pseudorandom language.

- **NOTE:** Not known whether \( \text{NP} \) is \( \text{P} \)-pseudorandom.

\[
\frac{|(A \cap L) \cap \Sigma^n|}{|A \cap \Sigma^n|} \to \frac{1}{2}
\]
Examples of C-Pseudorandom Languages

- Let $x \otimes y$ denote the (bitwise) binary inner product.
- Consider the following extended language in CFL.
  \[ IP^* = \{ axy \mid a \in \{\lambda, 0, 1\}, x, y \in \{0, 1\}^*, |x| = |y|, x^R \otimes y \equiv 1 \pmod{2} \} \]
- $IP^*$ is REG/n-pseudorandom. Hence, we obtain:
  - **Theorem:** [Yamakami (2011)]
    CFL is REG/n-pseudorandom.
  - The proof of this theorem utilizes the swapping lemma for regular languages, discussed in Week 5. (See the next slide.)
Swapping Lemma for REGs  [Yamakami (2008),(2010)]

• If $L$ is regular, then $\exists m>0$ s.t. $\forall n \in \mathbb{N} \ \forall S \subseteq L \cap \Sigma^n \ (|S| \geq m) \ 
\forall i \in [n] \ \exists xy, uv \in S \ (|x|=|u|=i) \ [ \ xy \neq uv \ 
& \ uy, xv \in L \ ]$.

• See Week 5 for the references.
CFL/n-Pseudorandom Languages

We discuss CFL/n-pseudorandom languages.

Consider the languages

- $\text{IP}^+ = \Sigma^{\leq 8} \cup (\text{IP}_3 \cap \Sigma^{\geq 8}) \Sigma^2$, where
- $\text{IP}_3 = \{ axyz | a \in \{\lambda, 0, 1\}, x, y, z \in \{0, 1\}^*, |x| = |z|, |y| = 2|x|, (xz) \circ y^{R \equiv 1 \pmod{2}} \}$ (extension of IP*)

CFL(2)/n is an advised version of CFL(2), which was discussed in Week 5.

Lemma: [Yamakami (2016)]

$L \in \text{CFL}(2)/n \iff \exists L_1, L_2 \in \text{CFL}/n \text{ s.t. } L = L_1 \cap L_2.$
CFL/n-Pseudorandom Languages II

- **Theorem:** [Yamakami (2016)]
  1. IP$_3$ and IP$^+$ are in $L \cap \text{CFL}(2)/n$.
  2. IP$_3$ and IP$^+$ are CFL/n-pseudorandom.

- For the latter claim of the above theorem, we need the **swapping lemma for context-free languages** discussed in Week 5. (See the next slide.)

- **Corollary:** [Yamakami (2016)]
  1. $L \cap \text{CFL}(2)/n \not\subset \text{CFL}/n$.
  2. $\text{CFL}(2) \not\subset \text{CFL}/n$. 
Swapping Lemma for CFLs [Yamakami, (2008, 2016)]

- If \( L \) is context-free, then \( \exists m > 0 \) s.t. \( \forall n \geq 2 \) \( \forall S \subseteq L \cap \sum^n \forall j_0, k_0 \in [2, n-1] \) \( \forall i \in [0, n] \forall j \in [j_0, k_0] \) \( (i+j \leq n) \) \( \forall u \in \sum^{j_0} \) \( (|S_{i,u}| < |S|/m(k_0-j_0+1)(n-j_0+1)) \) \( \exists x=x_1x_2x_3, y=y_1y_2y_3 \in S \) \( (|x_1|=|y_1|=i)(|x_2|=|y_2|=j)(|x_3|=|y_3|) \) \[ x_2 \neq y_2 & x_1y_2x_3, y_1x_2y_3 \in L. \]

Swapping Lemma for CFLs

- See Week 5 for the references.
Open Problems

• There are many open questions to solve.

1. Is there any CFL/n-pseudorandom language in CFL(2) (instead of CFL(2)/n)?
2. Find natural languages that are C-pseudorandom against D for reasonable language families C and D.
VI. P-Denseness and Primeimmunity

1. P-Denseness
2. P-Dense REG-Immunity
3. C-Primeimmunity
4. Examples of C-Primeimmune Languages
5. C-Bi-Primeimmunity
6. Examples of C-Bi-Primeimmune Languages
7. A Connection to C-Pseudorandomness
C-Immunity (revisited)

- Recall the definition of C-immune languages in Week 5.
- Immunity is concerned with “finiteness.”

- Let $C$ be any nonempty language family.

- A language $L$ is C-immune $\iff$
  1) $L$ is infinite, and
  2) no infinite subset $A$ of $L$ exists in $C$.

- A language family $D$ is C-immune $\iff$
  - $D$ contains a C-immune language.
P-Denseness

• All known context-free REG-immune languages $L$ make the ratio $|L \cap \Sigma^n| / |\Sigma^n|$ exponentially small.
  - E.g., $L_{eq}$ and $\text{Pal}_{\#}$

• A language $L$ is polynomially dense (or $p$-dense) $\iff$
  - There is a non-zero polynomial $p$ s.t. $|L \cap \Sigma^n| / |\Sigma^n| \geq 1/p(n)$ for all but finitely many $n$ (i.e., only polynomially small).

• Polynomial denseness is a key to our further discussion.
P-Dense REG-Immunity

• What language family is p-dense REG-immune?

• **Theorem**: [Yamakami (2011)]
  \( L \cap \text{CFL/n} \) is p-dense REG-immune.

  **Proof Sketch:**
  • Consider the language
    \[ \text{LCenter} = \{ ax0^m10^my \mid a \in \{\lambda,0,1\}, \ 2^m \leq |x|=|y| < 2^{m+1} \}. \]
  • Clearly, \( \text{LCenter} \in L \cap \text{CFL/n} \). Thus, it suffices to prove
    \( \text{LCenter is p-dense REG-immune,} \)
    by the pumping lemma for REGs.

• *(Open Problem)* Is CFL p-dense REG-immune?
C-Primeimmunity

• Let us introduce a variant of C-immunity using “p-dense” sets in place of “finite” sets.

• Let C be any language family.

  A language L is C-primeimmune ⇔
  1) L is p-dense, and
  2) L has no p-dense subset in C.

• A language family D is C-primeimmune ⇔
  ▪ D contains a C-primeimmune language.

• NOTE: p-dense REG-immune ⇒ REG-primeimmune
Examples of C-Primeimmune Languages

- **Equal** = \{ x ∈ \{0,1\}^* | \#_0(x) = \#_1(x) \} is not p-dense.
- Here, we consider its extended language:
  - **Equal**\* = \{ aw | a ∈ \{λ,0,1\}, w ∈ Equal \}

- **(Claim)**
  1. **Equal**\* is p-dense.
  2. **Equal**\* is in CFL.
  3. **Equal**\* is not REG-immune.
  4. **Equal**\* is REG/n-primeimmune.

- **Theorem:** [Yamakami (2011)]
  CFL is REG/n-primeimmune.

- **Proof:** This comes from Claims 2 & 4 above.
C-Bi-Primeimmunity

- Let C be any language family.

- A language L is C-bi-primeimmune $\iff$
  - L and $L^c$ are both C-primeimmune.

- A language family D is C-bi-primeimmune $\iff$
  - D contains a C-bi-primeimmune language.

```
Σ* → C-primeimmune

\[ \begin{array}{c}
\text{C-primeimmune} \\
\hline
L \\
\hline
L^c \\
\text{C-primeimmune}
\end{array} \]
```
Examples of C-Bi-Primeimmune Languages

• Recall that $x \odot y$ is the (bitwise) inner product of $x$ and $y$.
• Consider the following language:
  - $IP^* = \{ axy | a \in \{\lambda, 0, 1\}, x,y \in \{0,1\}^*, |x|=|y|, x^R \odot y \equiv 1 \pmod{2} \}$.  

• **Lemma:** [Yamakami (2011)]  
  $IP^*$ is REG/n-bi-primeimmune.

• Since $IP^* \in CFL$, we conclude the following statement.
• **Theorem:** [Yamakami (2011)]  
  CFL is REG/n-bi-primeimmune.
A Connection to C-Pseudorandomness

• There is a connection to C-pseudorandomness.

• **Lemma: [Yamakami (2011)]**
  If \( L \) is weakly C-pseudorandom, then it is C-bi-primeimmune.

• The converse does not hold, because the language \( \text{Equal}^* (\in \text{CFL}) \) is REG-primeimmune but not weakly REG-pseudorandom.
VII. PRGs by Finite Automata

1. Pseudorandom Generators
2. Existence and Limitation
3. Proof Idea for the Theorem
4. Generators Against CFL/n
Pseudorandom Generators I

- Let $G: \{0,1\}^* \rightarrow \{0,1\}^*$ be any function.

- $G$ has **stretch factor** $s(n) \iff |G(x)| = s(|x|)$ for all $x \in \{0,1\}^*$.

- $G$ fools a language $A$ (over $\{0,1\}^*$) $\iff$
  - $l(n) = \text{def} |\Pr_x[A(G(x)) = 1] - \Pr_y[A(y) = 1]|$ is negligible, where $|x| = n$ and $|y| = s(|x|)$.

- **Intuitively:** $A$ cannot tell the difference between truly random strings $y$ and generated strings $G(x)$.
Pseudorandom Generators II

• Let $G: \{0,1\}^* \rightarrow \{0,1\}^*$ be any function.

• $G$ is a pseudorandom generator against $C \iff$
  - for all $A \in C$ (over $\{0,1\}$), $G$ fools $A$.

• $G$ is a weakly pseudorandom generator against $C \iff$
  - for all p-dense $A \in C$ (over $\{0,1\}$), $G$ fools $A$.

• **NOTE:** pseudorandom generator $\Rightarrow$ weakly pseudorandom generator
There is a close connection between C-pseudorandom generators and C-pseudorandom languages.

First, we introduce a notion of **almost one-to-oneness**.

Let $G: \{0,1\}^* \rightarrow \{0,1\}^*$ have **stretch factor** $n+1$.

- **G is almost 1-1** $\iff$
  - There is a negligible function $t$ such that $|\{ G(x) \mid x \in \{0,1\}^n \}| = |\{0,1\}^n|(1 - t(n))$ holds for all $n$.

- **NOTE**: If $G$ is exactly 1-1, then $t(n)=0$. 
Let $G : \{0,1\}^* \rightarrow \{0,1\}^*$ be any almost 1-1 function with stretch factor $n+1$.

Let $S_G = \{ G(x) | x \in \{0,1\}^* \}$ be the image of $G$.

**Lemma**: [Yamakami (2011)]

$G$ is a (weakly) pseudorandom generator against $C \iff$
- the image $S_G$ of $G$ is (weakly) $C$-pseudorandom.

**Open Problem**

Can we weaken the above conditions of “almost 1-1” and “stretch factor $n+1$”?
Existence 1

- Here, we show the existence of pseudorandom generators against REG/n.
- Recall the function class CFLSV$_t$.

- **Theorem**: [Yamakami (2011)]
  There exists an almost 1-1 pseudorandom generator $G$ in CFLSV$_t$ with stretch factor $n+1$ against REG/n.

- (*) In the next slide, we will give a sketch of the proof of the above theorem.
Existence II

Proof Sketch:

• First, we define an almost 1-1 function $G: \{0,1\}^* \rightarrow \{0,1\}^*$ with stretch factor $n+1$ such that $G \in \text{CFLSV}_t$ and $S_G = \text{IP}^*$, where $S_G$ is the image $\{ G(x) | x \in \{0,1\}^* \}$ of $G$.

• We already know that $\text{IP}^*$ is REG/n-pseudorandom.

• Since $S_G = \text{IP}^*$, $S_G$ is REG/n-pseudorandom.

• As seen before, this implies that $G$ is a pseudorandom generator against REG/n.

QED
Next, we show a limitation of pseudorandom generators against REG/n.

- **Theorem: [Yamakami (2011)]**

  There is no almost 1-1 weakly pseudorandom generator in 1-FLIN with stretch factor $n+1$ against REG.

- (*) In the next slide, we will give a sketch of the proof.
Proof Sketch:

• Assume that such a generator $G$ exists.
• Define $H(xb) = G(x)$ for any $b \in \{0,1\}$.
• Since $H \in 1$-FLIN, it follows that $H^{-1} \in 1$-NLINMV.
• Take a refinement $f$ of $H^{-1}$ in $1$-FLIN(partial) by Week 6.
• Consider the image $S_G$ of $G$. Note that $y \in S_G \iff f(y) \downarrow$.
• Since $f \in 1$-FLIN(partial), we obtain $S_G \in 1$-DLIN = REG.
• It follows that $S_G$ is REG-pseudorandom.
• Since REG cannot be weakly REG-pseudorandom, a contradiction follows.

QED
Function Class CFLMV(2)/n

- Before moving to the next subject, we discuss an advised function class, called CFLMV(2)/n.
- Recall CFLMV(2) (= CFLMV \land CFLMV) from Week 6.
- Here, we consider its advised version, denoted by CFLMV(2)/n.

- **Lemma:** [Yamakami (2016)]
  For any multi-valued partial function f, \( f \in \text{CFLMV}(2)/n \iff \) there exist two multi-valued partial functions g,h \( \in \text{CFLMV}/n \) such that \( f(x) = g(x) \cap h(x) \) for any x.

- **In other words,** CFLMV(2)/n = CFLMV/n \land CFLMV/n.
Generators Against CFL/n

• Next, we consider pseudorandom generators against CFL/n.

• Theorem: [Yamakami (2016)]
  There exists an almost 1-1 pseudorandom generator $G$ in $FL \cap CFLMV(2)/n$ against CFL/n.

• Note that a famous design-theoretic method of Nisan and Wigderson (1994) does not provide a generator in $FL \cap CFLMV(2)/n$.

• (*) In the next slide, we will show how to define such a $G$. 
Definition of the Desired Generator

Proof Idea:

- We define the desired generator $G$ as follows.
- Let us set the value $G(w)$ with $w = axy$ and $|x| = |y| + 1$ for $a \in \{ \lambda, 0, 1 \}$ and $x, y \in \{0, 1\}^*$.
- If $a \neq \lambda$, set $G(aw) = aG(xy)$.
- Assume $a = \lambda$. Let $x = bz$ for $b \in \{0, 1\}$ and $k = (|w|-1)/2$.
  1. If $w = bzy \wedge z^R \odot y \equiv 1 \pmod{2}$, set $G(w) = bzyb^c$.
  2. If $w = 1zy \wedge z^R \odot y \equiv 0 \pmod{2}$, set $G(w) = 1zy1$.
  3. If $w = 0zy \wedge z^R \odot y \equiv 0 \pmod{2}$, there are two cases.
    a. If $\exists i \ [ z_{(k-i-1)} = 1$, set $G(w) = 0zy*0$, where $y*$ is obtained from $y$ by flipping only the $i$-th bit.
    b. Otherwise, $G(w) = 1zy1$.

QED
Here, we present an impossibility result.

**Theorem:** [Yamakami (2016)]
There is no almost 1-1 weakly pseudorandom generator in CFLMV with stretch factor $n+1$ against CFL.

The proof can be done by contradiction.
Open Problems

- There are many open questions to solve.
  1. Does a 1-1 PRG against CFL/n exist in CFLMV(2)/n?
  2. What happens if we use randomized advice instead of deterministic advice for pseudorandom generators?
  3. Is CFL p-sense REG-immune?
  4. We can define CFL-primesimple languages. Find CFL-primesimple languages.
  5. Is DCFL weakly REG/n-pseudorandom?
  6. Construct efficient pseudorandom generators against $\Sigma_k^{CFL}$. (See Week 4 for $\Sigma_k^{CFL}$.)
  7. Find a natural 1-1 pseudorandom generator against REG/n.
Thank you for listening
I’m happy to take your question!