Symmetric-Key Encryption: constructions

Lecture 4

PRG, PRF

Stream and Block Ciphers

Story So Far

- We defined (passive) security of Symmetric Key Encryption (SKE)
  - **SIM-CPA** = **IND-CPA** + (almost perfect) **correctness**
    - Restricts to PPT entities
    - Allows negligible advantage to the adversary
- Today:
  - One-time SKE from Pseudorandomness Generators (PRG)
  - Multi-message SKE from (Weak) Pseudorandom Functions (PRF)
- Later:
  - How to build PRGs and PRFs, in theory and in practice
  - In particular when the output is required to be very long

Constructing SKE schemes

- Basic idea: Use pseudo-random one-time pads (kept compressed in the key)
  - "PRG": an algorithm to stretch keys into long pseudo-random strings
- For multiple message encryption, will also need a mechanism to ensure that the same piece of the one-time pad is not used more than once
  - Implemented using a "PRF"

Pseudorandomness Generator (PRG)

- Expand a short random **seed** to a "random-looking" string
- First, PRG with fixed stretch: G_k: $\{0,1\}^k \to \{0,1\}^{n(k)}$, $n(k) > k$
- How does one define random-looking?
  - Next-Bit Unpredictability: PPT adversary can't predict i^{th} bit of a sample from its first (i-1) bits (for every $i \in \{1,\ldots,n\}$)
  - A "more correct" definition:
    - PPT adversary can't distinguish between a sample from $\{G_k(x)\}_{x \leftarrow \{0,1\}^k}$ and one from $\{0,1\}^{n(k)}$
  - **Turns out they are equivalent!**

$|\ Pr_{y \leftarrow PRG}[A(y)=0] - Pr_{y \leftarrow rand}[A(y)=0]\ |$ is negligible for all PPT A

Computational Indistinguishability

- Two distribution ensembles $\{X_k\}$ and $\{X'_k\}$ are said to be **computationally indistinguishable** if

  $X_k \approx X'_k$

  - $\forall$ (non-uniform) PPT distinguisher D, $\exists$ negligible $\nu(k)$ such that $| Pr_{x \leftarrow X_k}[D(x)=1] - Pr_{x \leftarrow X'_k}[D(x)=1] | \leq \nu(k)$

- cf.: Two distribution ensembles $\{X_k\}$ and $\{X'_k\}$ are said to be **statistically indistinguishable** if $\forall$ functions T, $\exists$ negligible $\nu(k)$ s.t. $| Pr_{x \leftarrow X_k}[T(x)=1] - Pr_{x \leftarrow X'_k}[T(x)=1] | \leq \nu(k)$

  - Equivalently, $\exists$ negligible $\nu(k)$ s.t. $\Delta(X_k, X'_k) \leq \nu(k)$ where $\Delta(X_k, X'_k) := \max_T | Pr_{x \leftarrow X_k}[T(x)=1] - Pr_{x \leftarrow X'_k}[T(x)=1] |$

Pseudorandomness Generator (PRG)

- Takes a short seed and (deterministically) outputs a long string
  - G_k: $\{0,1\}^k \to \{0,1\}^{n(k)}$ where $n(k) > k$
- Security definition: Output distribution induced by random input seed should be "pseudorandom"
  - i.e., Computationally indistinguishable from uniformly random
  - $\{G_k(x)\}_{x \leftarrow \{0,1\}^k} \approx U_{n(k)}$
  - Note: $\{G_k(x)\}_{x \leftarrow \{0,1\}^k}$ cannot be statistically indistinguishable from $U_{n(k)}$ unless $n(k) \leq k$ (Exercise)
    - i.e., no PRG against unbounded adversaries

Equivalent definitions

$|\ Pr_{y \leftarrow PRG}[B(y_1^{i-1}) = y_i\ ] - \frac{1}{2}\ |$ is negligible for all i, all PPT B

$|\ Pr_{y \leftarrow PRG}[A(y)=0] - Pr_{y \leftarrow rand}[A(y)=0]\ |$ is negligible for all PPT A

- Next-Bit Unpredictable ⇔ Pseudorandom
- Pseudorandom ⇒ NBU:
  - Reduction: Given a PPT adversary B (for NBU), will show how to turn it into a PPT adversary A (for Pseudorandomness) with similar advantage. Hence the advantage must be negligible.

    Could be seen as showing the contrapositive: ¬NBU ⇒ ¬Pseudorandom
  - For any PPT B and i, consider PPT A which uses it to predict i^{th} bit and then checks if the prediction was correct
  - Formally, A(y) outputs $B(y_1^{i-1}) \oplus y_i$ (i as specified by B). Then:

    $|\ Pr_{y \leftarrow PRG}[A(y)=0] - Pr_{y \leftarrow rand}[A(y)=0]\ | = |\ Pr_{y \leftarrow PRG}[B(y_1^{i-1}) = y_i\ ] - \frac{1}{2}\ |$

Equivalent definitions

$|\ Pr_{y \leftarrow PRG}[B(y_1^{i-1}) = y_i\ ] - \frac{1}{2}\ |$ is negligible for all i, all PPT B

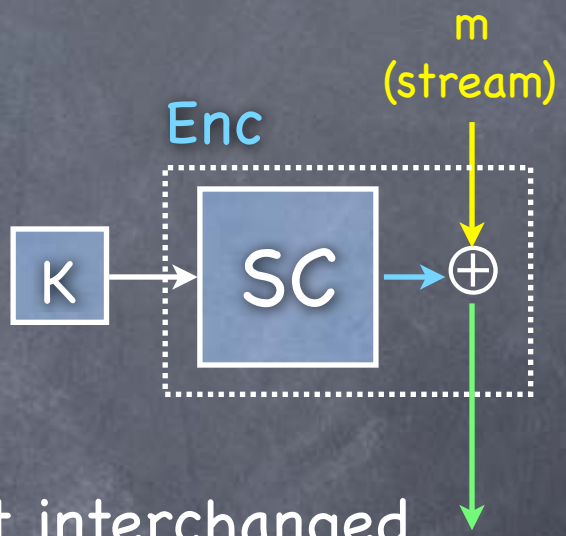
$|\ Pr_{y \leftarrow PRG}[A(y)=0] - Pr_{y \leftarrow rand}[A(y)=0]\ |$ is negligible for all PPT A

- Next-Bit Unpredictable ⇔ Pseudorandom
- NBU ⇒ Pseudorandom: Using a **Hybrid Argument**
  - Define distributions H_i over n-bit strings: $y \leftarrow$ PRG. Output $y_1^i \,||\, r$ where r is n-i independent uniform bits. H_0 = rand, H_n = PRG.
  - PRG is NBU ⇒ $H_i \approx H_{i+1}$: Given a PPT distinguisher A for H_i vs. H_{i+1}, let PPT predictor B be as follows: On input $z \in \{0,1\}^i$, pick $b \leftarrow \{0,1\}$, $r \leftarrow \{0,1\}^{n-i-1}$ and output $A(z \,||\, b \,||\, r) \oplus b$. Then [Exercise] :

    $|Pr_{y \leftarrow PRG}[B(y_1^{i-1}) = y_i\ ] - \frac{1}{2}| \;=\; |Pr_{y \leftarrow H_i}[A(y)=0] - Pr_{y \leftarrow H_{i+1}}[A(y)=0]|$
  - Then [Exercise] : $H_0 \approx H_n$ (for n(k) that is polynomial)

One-time secure SKE with a Stream-Cipher

- One-time Encryption with a **stream-cipher**:
  - Generate a one-time pad from a short seed
  - Can share just the seed as the key
  - Mask message with the pseudorandom pad



- Decryption is symmetric: plaintext & ciphertext interchanged
- SC can spit out bits on demand, so the message can arrive bit by bit, and the length of the message doesn't have to be a priori fixed
- Security: indistinguishability from using a truly random pad (coming up)

Stream Ciphers

- Stream ciphers in practice
  - Naturally useful for onetime (stream) encryption, in protocols where a key is established per session

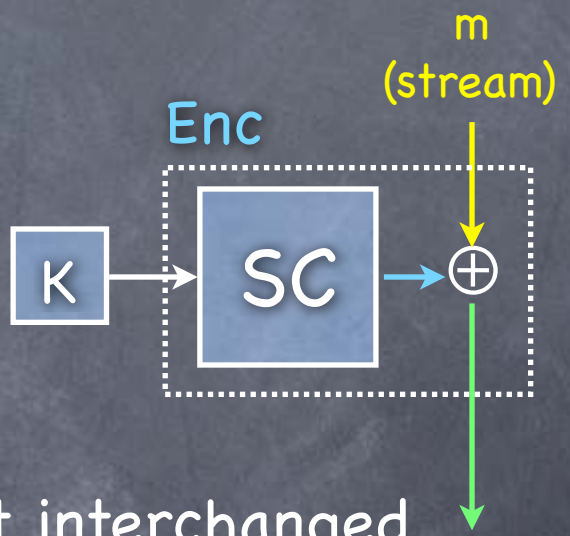
- Many popular candidates:
  - **RC4**: **Obsolete** (but popular). Designed in 1987. Leaked (and broken) in 1994. Still used in BitTorrent, and supported as an option in some protocols.
  - **eSTREAM** portfolio:

Profile 1 (software)	HC-128, Rabbit, Salsa20/12, SOSEMANUK	128 bit keys
Profile 2 (hardware)	Grain, MICKEY, Trivium	80 bit keys

  - **NIST** recommendation: AES in an appropriate mode (later)

One-time secure SKE with a Stream-Cipher

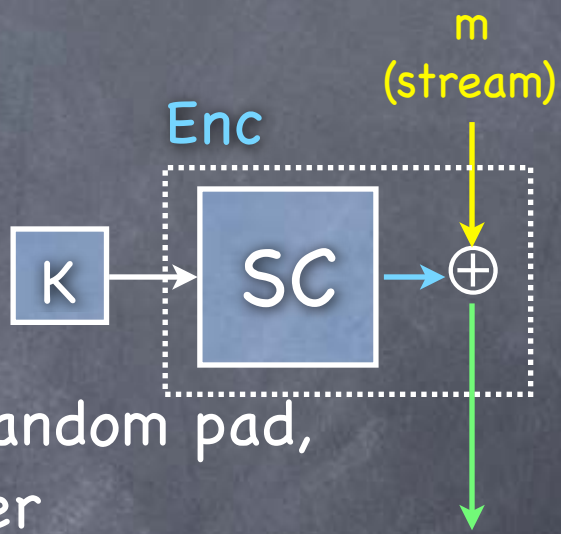
- One-time Encryption with a **stream-cipher**:
  - Generate a one-time pad from a short seed
  - Can share just the seed as the key
  - Mask message with the pseudorandom pad



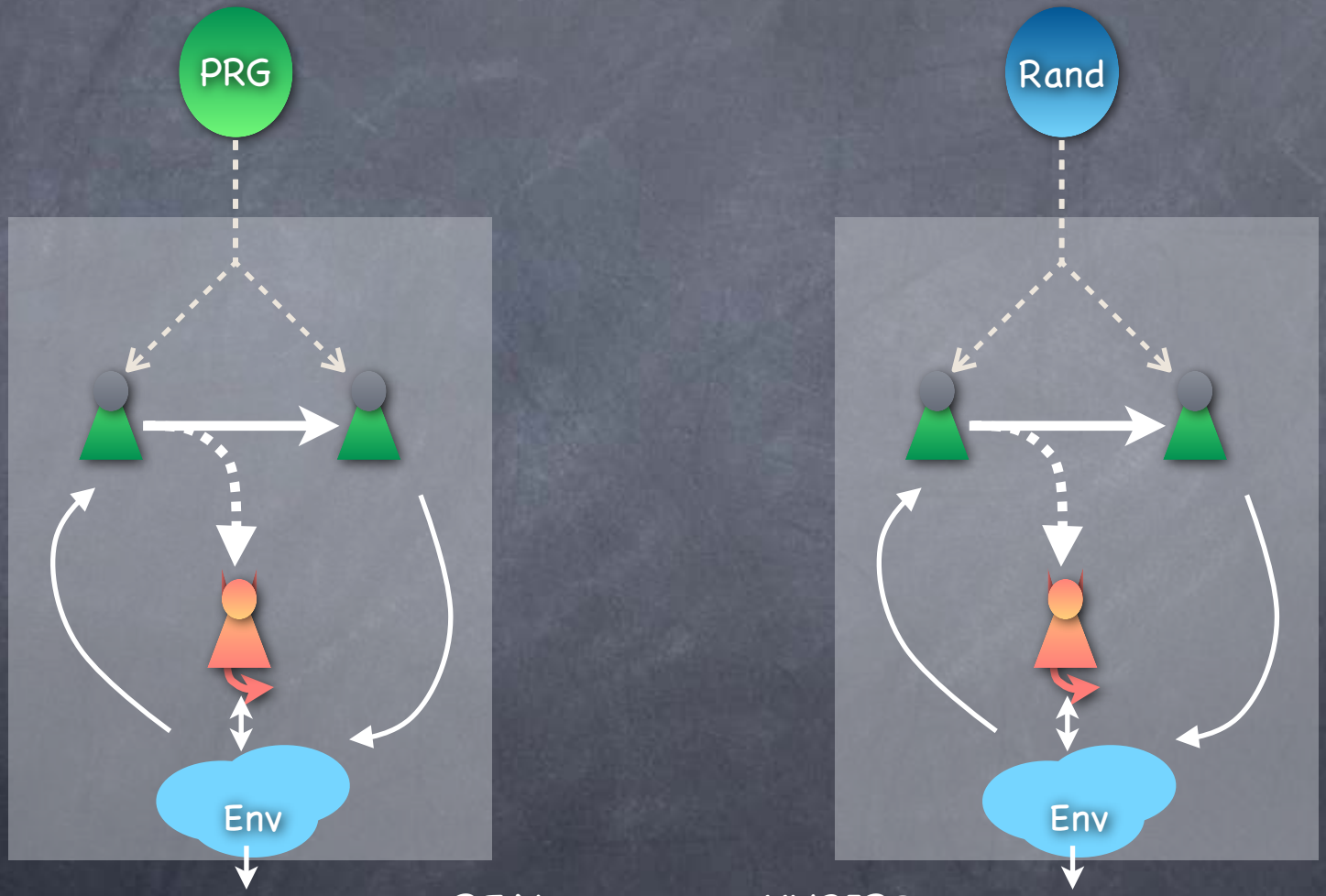
- Decryption is symmetric: plaintext & ciphertext interchanged
- SC can spit out bits on demand, so the message can arrive bit by bit, and the length of the message doesn't have to be a priori fixed
- SIM-CPA security due to indistinguishability from using a truly random pad

One-time secure SKE with a Stream-Cipher

- In IDEAL experiment, consider simulator that uses a truly random string as the ciphertext
- To show REAL ≈ IDEAL
- Consider an intermediate world, HYBRID:
  - Like REAL, but Enc/Dec use a (long) truly random pad, instead of the output from the stream-cipher
  - HYBRID = IDEAL (recall perfect security of one-time pad)
  - Claim: REAL ≈ HYBRID
    - Consider the experiments as a system that accepts the pad from outside (R′ = SC(K) for a random K, or truly random R) and outputs the environment's output. This system is PPT, and so can't distinguish pseudorandom from random.



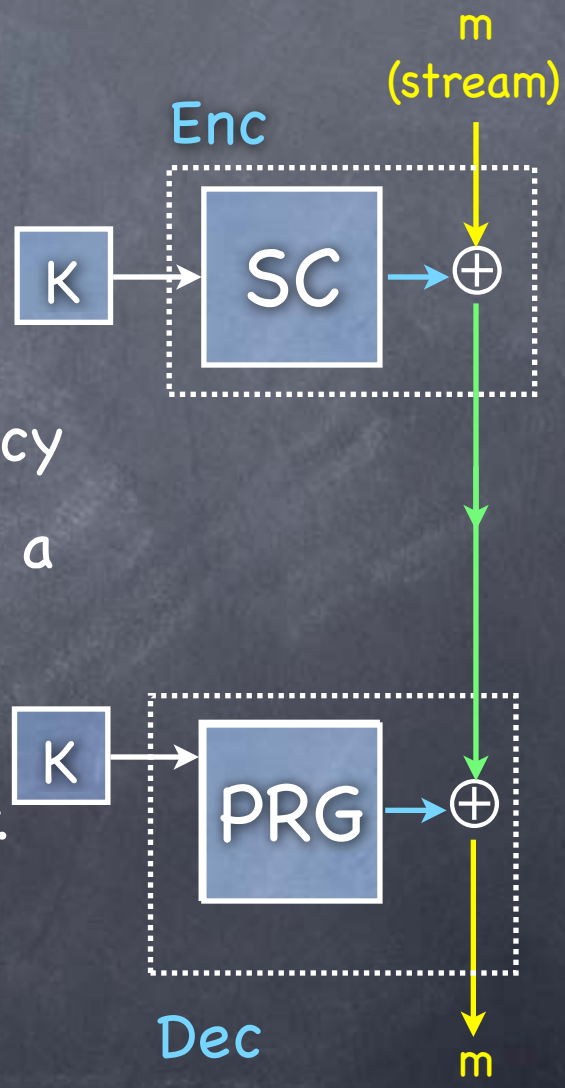
One-time secure SKE with a Stream-Cipher



REAL ≈ HYBIRD

Recap

- One-time pad provides perfect secrecy
  - Key = a random string as long as the message
  - Major limitation: Cannot communicate indefinitely using a given key
- Stream cipher provides (only) computational secrecy
  - Key = seed of a PRG, used to generate as long a pseudorandom one-time pad as needed
  - Major limitation: Only one output stream. And receiver needs to stay in sync with the sender.
    - Fix: use a Psuedorandom Function (PRF) instead of a PRG

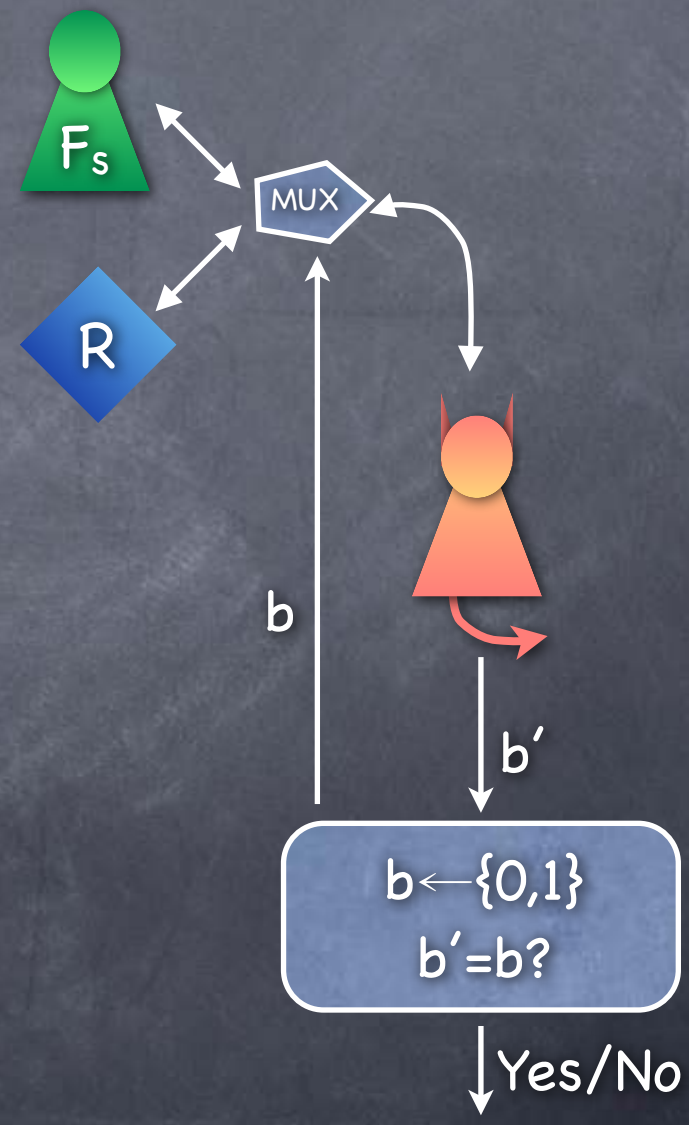


Pseudorandom Function (PRF)

- A compact representation of an exponentially long (pseudorandom) string
  - Allows "random-access" (instead of just sequential access)
    - A function F(s;i) outputs the i^{th} block of the pseudorandom string corresponding to seed s
    - Exponentially many blocks (i.e., large domain for i)
- Pseudorandom Function
  - Need to define pseudorandomness for a function (not a string)

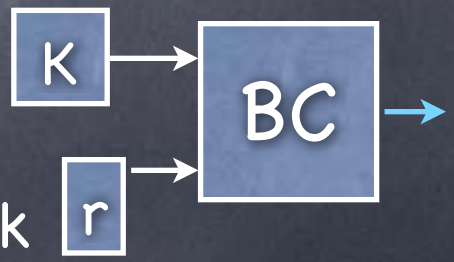
Pseudorandom Function (PRF)

- $F: \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ is a PRF if all PPT adversaries have negligible advantage in the PRF experiment
  - Adversary given oracle access to either F with a random seed, or a random function $R: \{0,1\}^m \rightarrow \{0,1\}^n$. Needs to guess which.
  - Note: Only 2^k seeds for F
    - But $2^{(n2^m)}$ functions R
  - PRF stretches k bits to $n2^m$ bits



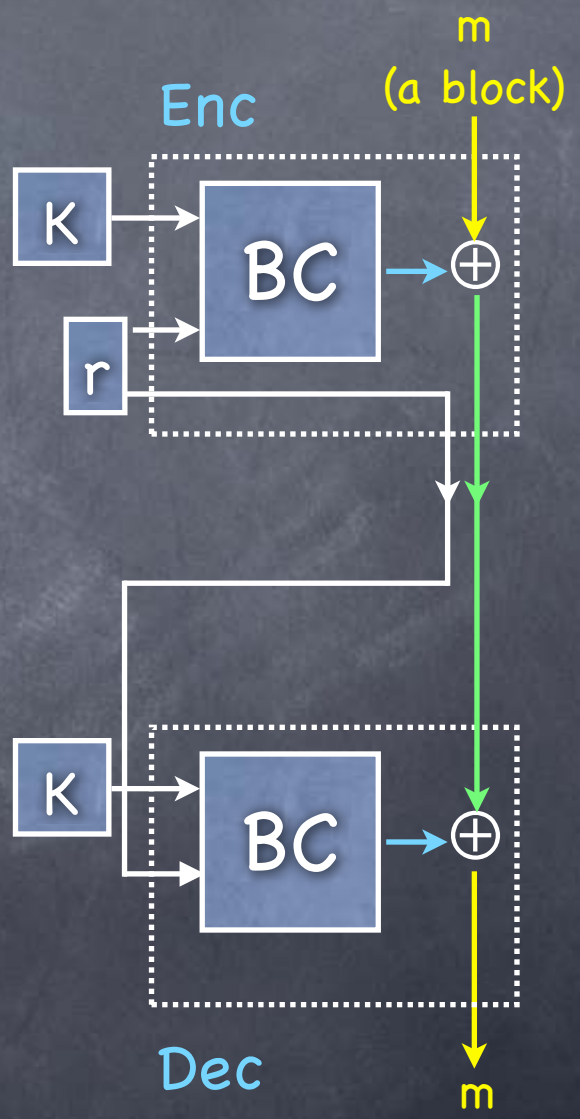
Pseudorandom Function (PRF)

- A PRF can be constructed from any PRG
  - Not blazing fast: needs |r| evaluations of a PRG
  - Faster constructions based on specific number-theoretic computational complexity assumptions
  - Fast heuristic constructions
- PRF in practice: Block Cipher
  - Extra features/requirements:
    - Permutation: input block (r) to output block
    - Key can be used as an inversion trapdoor
    - Pseudorandomness even with access to inversion



SKE with a PRF (or Block Cipher)

- Suppose Alice and Bob have shared a key (seed) for a block-cipher (or PRF) BC
- For each encryption, Alice will pick a fresh pseudorandom pad, by picking a new value r and setting pad=$BC_K(r)$
- Bob needs to be able to generate the same pad, so Alice sends r (in the clear, as part of the ciphertext) to Bob
- Even if Eve sees r, PRF security guarantees that $BC_K(r)$ is pseudorandom
- How to pick a new r?
  - Pick at random!



Recap

- **One-time pad** provides perfect secrecy
  - Major limitation: Cannot communicate indefinitely using a given key
- **Stream cipher (PRG)** gives computational secrecy
  - Major limitation: Receiver needs to stay in sync with the sender.
- Can use a **block cipher** (**PRF** suffices)
  - Limitation: To encrypt one block of message, two blocks of ciphertext
- Next up: More efficient ways (modes) of using a block cipher for encryption
- Later: Constructions for PRG and PRF