CSA E0 235: Cryptography 28 January 2016

Scribe for Lecture #5

Instructor: Dr. Arpita Patra *Submitted by: Nidhi Rathi*

1 Pseudo-randomness and PRG's

We saw that computational security introduces two relaxations of perfect secrecy -
1) Security is guaranteed only against efficient adversaries.
2) A small probability of success is allowed.
Both these relaxations are essential for achieving practical encryption schemes, and in particular bypassing the negative results for perfectly secure encryption - key-length and key re-usability.
Generating true random bits is very expensive (difficult and slow). [1] A natural approach for making the scheme more efficient is to use a small amount of true randomness in order to generate a large amount of pseudorandomness.
So, what do we really mean by that?

Definition 1.1: (Pseudorandomness)
Pseudorandomness is a property of **a probability distribution** on a set of binary strings of length l. And, it does not make any sense to say that any fixed string is "pseudorandom" or "random". But, this abuse of terms is anyway used informally.

Let **G** = A Prob. Dist on n-bit strings = Set of probabilities
. A string drawn acc. to G is called *pseudorandom*.
. **U** = Uniform Prob. Dist. on the same set.
. A string drawn acc. to U is called *random*.

G is pseudorandom if a string drawn according to **G** is indistinguishable from a string drawn according to **U** to a PPT distinguisher. This means that *a pseudorandom string is just as good as a uniform string*, as long as we consider only PPT observers.

Remarks:

- Just as indistinguishability is a computational relaxation of perfect secrecy, pseudorandomness is a computational relaxation of true randomness.
- The seed s for a PRG is analogous to the cryptographic key used by the encryption schemes. s must be chosen uniformly and kept secret from adversary.

[1] It suffices to think of entropy as a measure of unpredictability. This pool of high-entropy data is collected from sources like delays b/w network events, hard-disk access times, keystrokes or mouse movements made the user, etc.

A pseudorandom generator G is an efficient deterministic algorithm for transforming a short, uniform string called the *seed* into a longer, "pseudorandom" output string.
(Two requirements : Expansion and pseudo-randomness.)

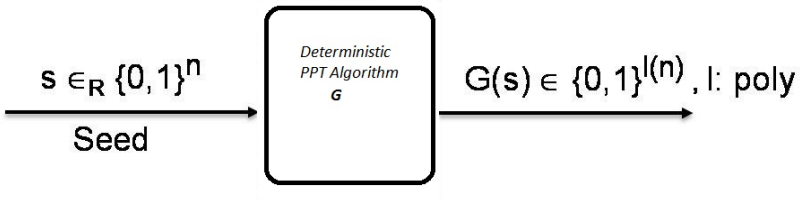

Figure 1: Deterministic PPT Algorithm G

2 PRG security

Let **G** : Probability distribution over $\{G(s) : s \longleftarrow \{0,1\}^n$
. **U** = Uniform Prob. Dist. over $\{0,1\}^{l(n)}\}$

1) The PPT Distinguisher **D** asks for a string of length l(n) from the Challenger C.
2) C then flips a coin to get b=0 or 1.
3) If b=0, he asks the random oracle to provide with a string of length l(n) and gets such a y from him.
If b=1, C sends a random seed s of length n to PRG G, and gets back with $y := G(s)$.
4) C sends the y string (of length l(n) obtained according the value of bit b), across to D, and challenges him to tell how y was selected.

Definition 2.1: (Pseudorandom generator G)
G is a **Pseudorandome generator** if ∀ PPT distinguishers D, ∃a negligible function *negl* s.t.
..
$$|Pr[(D(r) = 1] - Pr[D(G(s)) = 1]| \leq negl$$
where,
the 1^{st} probability is taken over uniform choice of r $\in \{0,1\}^{l(n}$ and the randomness of D
the 2^{nd} probability is taken over uniform choice of $s \in \{0,1\}^n$ and the randomness of D.

Therefore, any PPT distinguisher D must be able to tell apart random strings from pseudo-random strings with only negligible probability.

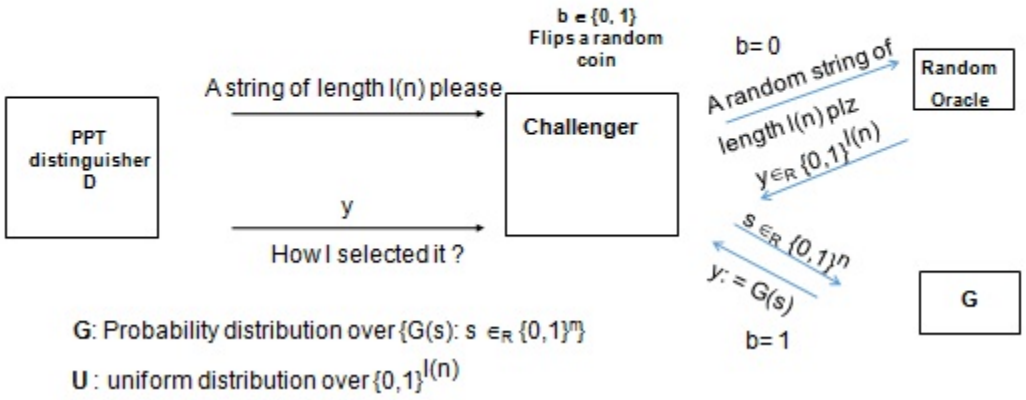

Figure 2: PRG Security

2.1 Let us try to construct a PRG

Let $s \in \{0,1\}^n$.
Let $s' = s_1 \oplus s_2 \oplus .... \oplus sn$ (1 bit string)
Define $G(s) = ss' \Rightarrow$ Expansion factor of G is $l(n) = n + 1$.

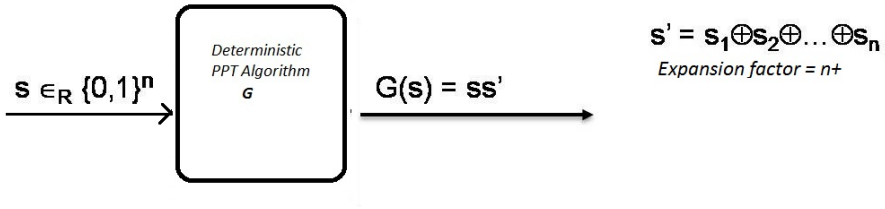

Figure 3: An example of insecure PRG

Is G a PRG?? Do you see a good distinguisher D?
Consider the following -
D, on input of a string w, outputs 1 iff the last bit 0f w = XOR of all the preceding bits of w.
Since it is true for all strings output by G, we have -
.. $Pr[D(G(w)) = 1] = 1$
On the other hand,
if w is uniform, the last bit of w is uniform and so,
.. $Pr[D(w) = 1] = 1/2$

Therefore, D outputs 1 to indicate that y is generated by PRG G, and D outputs 0 to indicate that he thinks y to be random.
$\Rightarrow |Pr[(D(r) = 1] - Pr[D(G(s)) = 1]| = 1/2$, which is not negligible.
$\Rightarrow$ G is not PRG.

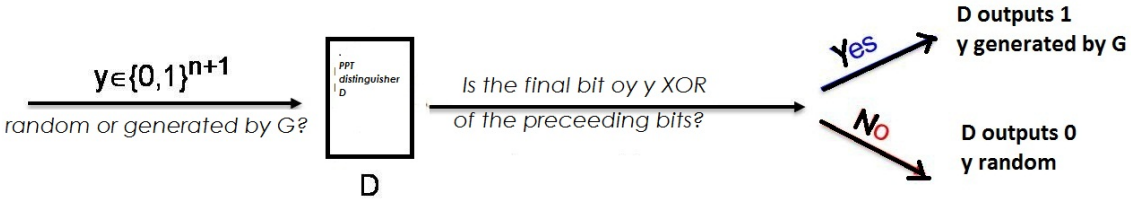

Figure 4: An example of insecure PRG and its PPT distinguisher D

2.2 PRG can be cracked by an unbounded adversary

The distribution on the output of a PRG G is far from uniform. Consider a length-doubling PRG G.
Under the uniform distribution on$\{0,1\}^{2n}$, each of the 2^{2n} possible strings is chosen with probability exactly 2^{-2n}.
In contrast, consider the distribution of the output of G (when G is run on a uniform seed).
. Pr[A random string of 2n-length $\in$ the range of G] $\leq 2^n/2^{2n} = 2^{-n}$

Thus, the vast majority of 2n-bits long strings do not belong to the range of G.
Hence, in particular it is trivial to distinguish b/w a random string and pseudorandom string *given an unlimited amount of time*

Attack:
Let G be as above and consider the exp-time distinguisher D that works as follows -
$D(w)$ outputs 1 iff $\exists$ an $s \in \{0,1\}^n$ such that $G(s) = w$.
This computation is carried out in exponential time by exhaustively computing $G(s) \forall s \in \{0,1\}^n$. Recall that, by Kerchoffs' principle, the specification of G is known D.
Now, If $w \in range(G)$, then D outputs 1 with prob. 1.
In contrast, $w \longleftarrow \{0,1\}^{2n}$, then D outputs 1 with prob at most 2^{-n}.
So,

.. $$|Pr[(D(r) = 1] - Pr[D(G(s)) = 1]| \geq 1 - 2^{-n}$$

which is large.

Above was an example of *brute force attack* by an unbdd. adversary.
Therefore, the seed s of a PRG must be long enough so as to make enumeration of all possible seeds not feasible. This is taken care of by setting the length of the seed equal to the security parameter, so that exhaustive search requires exponential time.

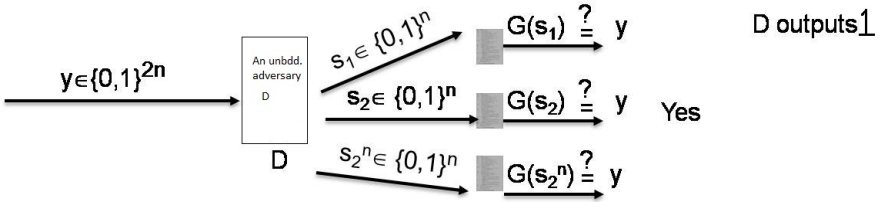

Figure 5: Breaking PRG under unbdd adversary

2.3 Existence of PRG

We do not have any unconditional proof for the existence of such a Pseudorandom generators. They certainly seem difficult to construct, but the crypto-community has strong reasons to believe they exist.
For one, they can be constructed under the rather weak assumption that *one-way functions* exist. And, we have several practical constructions for PRG, called *stream-ciphers*.
We'll study about these primitives later in the course in detail.
But, for now, we've got the very first assumption in this course -

Assumption 1: **PRGs EXIST**

Highly practical Stream-ciphers are used, which would be introduced in the upcoming lectures.

3 Ciphertext only attack

We have seen that a Symmetric Key Encryption scheme is one in which the sender and receiver (Alice and Bob), have a shared secret key. The security definition for an encryption scheme depends on the power the adversary is likely to have and the information that we need to protect.
Let us first consider **Ciphertext-only attack**. This is the most basic attack, and refers to a scenario where the adversary just observes a ciphertext (or multiple ciphertexts) and attempts to determine information about the underlying plaintext(s). This is the threat model we have been implicitly assuming when discussing classical encryption schemes in the previous section. This is a primitive adversary who is only eavesdropping across the transmission channel and has access only to the cipher text that is being sent across.

3.1 COA-secure SKE from PRG

A PRG provides a natural way to construct a secure, fixed-length encryption scheme with a key shorter than the message.
The insight here is rather than encrypting the message by XORing it with a random pad, we can use a pseudorandom pad instead.

3.2 The Encryption Scheme

Assumption: PRG's exist.

- Gen: It takes as input the security parameter n and outputs a random string $k \in \{0,1\}^n$. This is the secret key shared between sender and receiver.
- Enc: It takes as input the secret key k and message $m \in \{0,1\}^{l(n)}$. It outputs $c = G(k) \oplus m$
- Dec: It takes as input the secret key $k \in \{0,1\}^n$ and cipher text $c \in \{0,1\}^{l(n)}$, and outputs $m = G(k) \oplus c$

Correctness : $Dec_k(Enc_k m) = m$

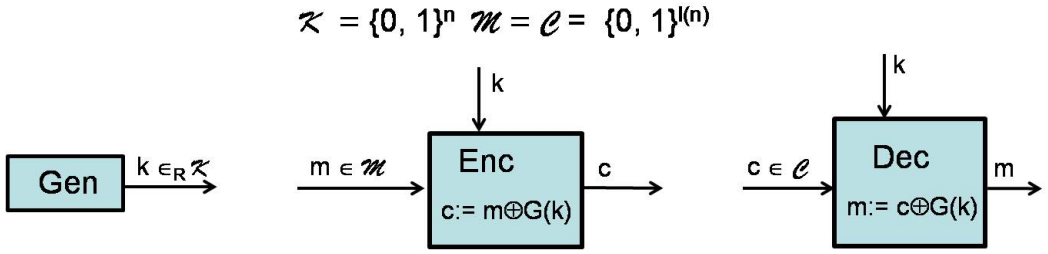

Figure 6: COA-secure SKE

3.3 COA-seurity Proof by Reduction

Recall that we have unconditional/ absolute proofs for perfectly secure encryption schemes (e.g. OTP). But, in the computational world, our proof for security of some construction relies on some unproven assumptions or problems believed to be hard to solve. We intend to reduce the security of a scheme to hardness of some difficult mathematical problem or to security of some low-level cryptographic primitive.
The **art** lies in how to transform any efficient adversary $\boldsymbol{A}$ that succeeds in "breaking" our construction into an efficient algorithm $\boldsymbol{A}$'that solves the problem that was assumed to be hard.

The four cases that may arise are -

- **Case1:** If π is secure, then π' is secure.
- **Case1:** If A holds, then π is secure.
- **Case1:** If A1 holds, then A2 holds.
- **Case1:** If π is secure, then A holds.

Outline : Proof by contradiction/reduction (For case2) :

Assumption: Problem X cannot be solved by any PPT algorithm.
Want to show: Some cryptographic construction π is secure.

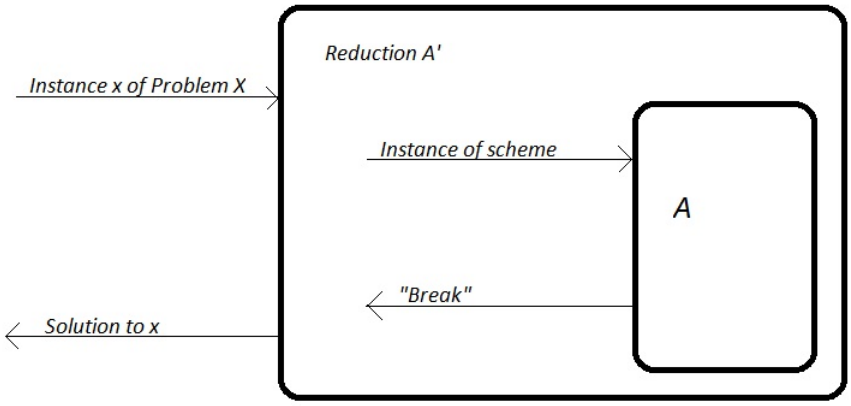

Figure 7: A high;level overview of a security proof by reduction

1. Let $\exists$ some efficient PPT adv. $\boldsymbol{A}$ attacking π, with non-neg success prob. $= f(n)$.
2. Construct a PPT algorithm $\boldsymbol{A'}$, called the "reduction", that attempts to solve problem X using the adversary $\boldsymbol{A}$ as a sub-routine (a black-box).
3. The only thing $\boldsymbol{A'}$ knows is that $\boldsymbol{A}$ is expecting to attack π. So, given some input instance x of problem X, our algorithm $\boldsymbol{A'}$ will simulate for $\boldsymbol{A}$ an instance of π s.t.
   - As far $\boldsymbol{A}$ can tell, it is interacting with π.
   - The simulation is somehow related to his own problem instance x. And, if $\boldsymbol{A}$ succeeds in "breaking" this instance, this should allow $\boldsymbol{A'}$ to solve x with prob $\geq 1/p(n)$.
4. Taken together, the probability that $\boldsymbol{A'}$ solves X is at least f(n)/P(n) – non-negligible. $\Rightarrow$ Problem X can be solved efficiently, which is a contradiction.

Below is the graphical outline for proof by reduction for the case 1, which will be used to prove coa-security of the above-mentioned SKE.

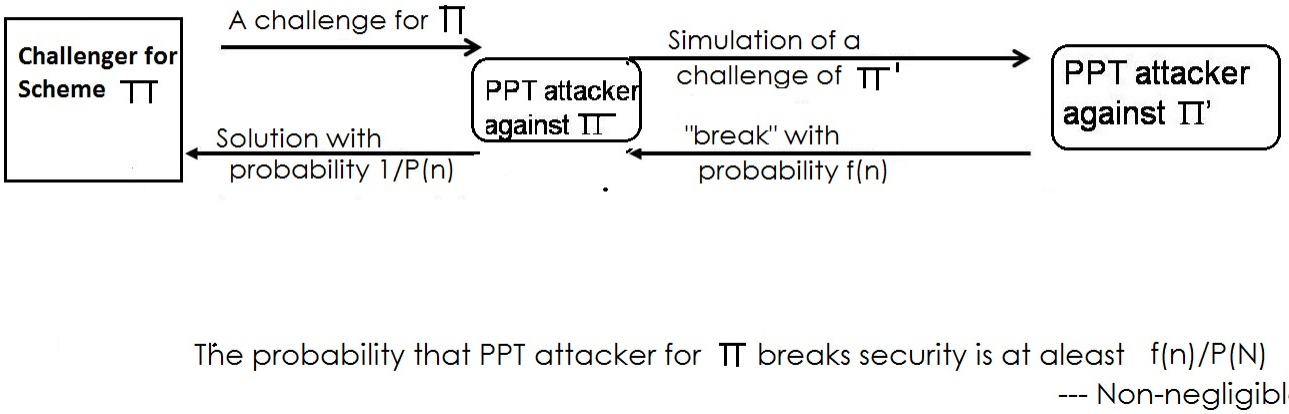

Figure 8: Proof by Reduction Case 1

4 Indistinguishability Based Definition - COA

Indistinguishability experiment: $PrivK^{coa}_{A,\pi}(n)$ $\pi = (Gen, Enc, Dec), M$

This game is played between a challenger and an adversary, both PPT machines.
The adversary picks two strings m_0 and m_1 and sends them to the challenger.
The challenger then chooses a random bit $b \longleftarrow \{0,1\}$ and sends the (corresponding) encrypted cipher text of m_b.
The adversary then guesses b'. If $b = b'$, the adversary wins, otherwise he looses.

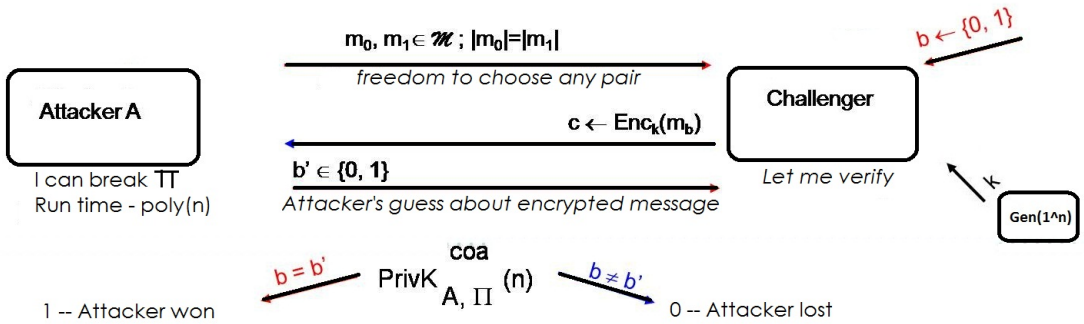

Figure 9: Indistinguishability based COA

Definition 4.1: (COA-security of a scheme π)

Π is coa-secure if for every PPT attacker A, there is a negligible function negl(.) such that

$$\Pr\left[\mathrm{PrivK}^{\mathrm{coa}}_{A,\Pi}(n) = 1\right] \leq \tfrac{1}{2} + \mathrm{negl}(n)$$

Probability is taken over the randomness used by A and the attacker

Now, we finally prove the security of the above game.

5 Security of the PRG-based SKE

Theorem 1: If G is a PRG, then π is a fixed-length coa-secure SKE.

Proof: Let us suppose π is not coa-secure SKE. Then, let $\boldsymbol{A}$ is PPT Turing machine such that for some polynomial $q(n)$ and messages m_0 and m_1 chosen by $\boldsymbol{A}$,

$|Pr[\boldsymbol{A}(c) = 1|c = Enc(m_0)]Pr[\boldsymbol{A}(c) = 1|c = Enc(m_1)]| > 1/q(n)$
for infinitely many n.

Consider a game $PrivK^{coa}_{A,\pi'}(n)$,that is played in the same way as $PrivK^{coa}_{A,\pi}(n)$ but the encryption scheme of π' uses a truly random string instead of the pseudorandom string to encrypt the message. This encryption scheme is now identical to the one time pad. Hence
.
$$Pr[A \text{ wins } PrivK^{coa}_{A,\pi'}(n)] = 1/2$$

Now we construct a distinguisher D for the pseudorandom generator G.

- D receives a string s from the challenger. This string may be truly random or pseudorandom, unknown to D.
- D plays a game with the adversary $\boldsymbol{A}$ as described below.

- $\boldsymbol{A}$ sends two strings m_0 and m_1 to D.
- D chooses a random bit b, computes $s \oplus m_b$ and sends it to $\boldsymbol{A}$.
- $\boldsymbol{A}$ outputs b'.
- If $b = b'$, D outputs 1 else it outputs 0.

Suppose the challenger sent a truly random string s. Then the game between D and $\boldsymbol{A}$ is is $PrivK^{coa}_{A,\pi'}(n)$. So $Pr[b = b'|s \text{ is truly random}] = 1/2$.
Suppose the challenger sent the output of the pseudorandom generator G, then the game between D and $\boldsymbol{A}$ is $PrivK^{coa}_{A,\pi}(n)$. So $Pr[b = b'||s \text{ is pseudo-random}] > 1/2 + 1/q(n)$ for infinitely many n.

Hence, $|Pr[D(G(s)) = 1]Pr[D(r) = 1]| > 1/2 + 1/q(n) - 1/2 > 1/q(n)$ for infinitely many n. Hence the probability of distinguishing between a truly random generator and PRG G is not negligible. This implies that G is not a pseudorandom generator.

Thus π is not COA-Secure $\Rightarrow$ G is not a PRG.
Since G is a PRG, π is COA-secure.

Pictorial representation :

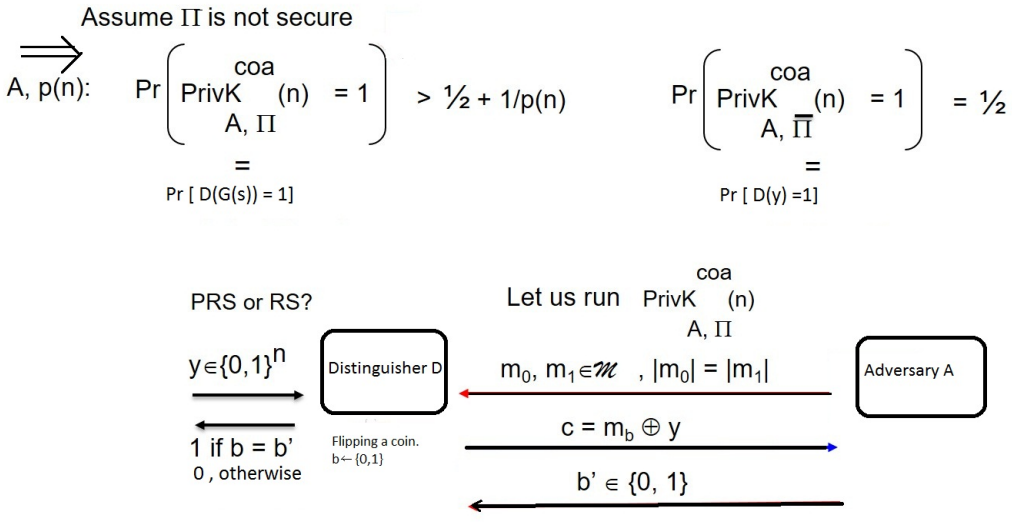

Figure 10: COA-security based on PRG-based scheme

6 Multi-message COA security

Until now, we have just considered passive adversary attacking on a single ciphertext transmitted b/w the honest parties. Recall that a security definition specifies a security goal and an attack model. We will modify our security goal to "*Security of Multiple Encryptions (using the same key)*" and strengthen the attack model to "*Chosen-Plaintext attacks (CPA)*". The former will be discussed now, while the latter in the next lecture.

The game $PrivK_{A,\pi}^{coa-mult}(n)$ is played between a challenger and an adversary, both PPT machines. The adversary picks two sets of strings $M_0 =< m_{0,1}, m_{0,2}, ..., m_{0,t} >$ and $M_1 =< m_{1,1}, m_{1,2}, ..., m_{1,t} >$ and sends them to the challenger. The challenger then chooses a random bit b and sends the encrypted cipher text $C = Enc(M_b) =< Enc(m_{b,1}), Enc(m_{b,2}), ..., Enc(m_{b,t}) >$. The adversary then guesses b'. If $b = b'$, the adversary wins and otherwise he loses.

Definition 6.1: (COA-Multiple Security)
A *symmetric key encryption scheme* (Gen, Enc, Dec) is said to be ciphertext-only-attack multiple message secure if for all PPT algorithms $\boldsymbol{A}$, there exists a negligible function $negl(.)$ such that
$Pr[\boldsymbol{A}$ wins in $PrivK_{A,\pi}^{coa-mult}(n)$] is atmost negligibly better than 1/2.

$$\text{i.e. } \Pr\left[\text{PrivK}^{\text{coa-mult}}_{A,\Pi}(n) = 1\right] \leq \tfrac{1}{2} + \text{negl}(n)$$

A natural question to ask is if the two notions of security above are actually equivalent or one is stronger than the other. It is easy to observe that the single message security is a special case of multiple message security with t = 1. The converse is not true though. By allowing multiple messages, an adversary still may not be able to get any information about the individual messages. However he can still learn somethings about the combination of the multiple messages. A break for the encryption scheme above is sketched below.

Consider an adversary A that chooses $M_0 =< x, x >$ and $M_1 =< x, y >$ for some two strings x and y such that Enc(x) $\neq$ Enc(y). When the challenger sends $C =< c1, c2 >$ A outputs 0 if $< c1 = c2 >$ and 1 otherwise. The adversarys output is always right by the choice of M_0 andM_1.

The above break works for any scheme where the encryption of the message $< x, x >$ is of the form $< c1, c1 >$. In particular, any deterministic encryption algorithm fails the multiple messages security definition. This calls for randomized encryption algorithms.
As we have already mentioned COA-security is the first step towards understanding various threat and break models. We'll further learn about CPA-security, which is much stronger notion.
We have come across two assumptions today :
1) PRG's exists
2) COA-secure SKE exist
Do they imply existence of something even more fundamental?
Yes!

7 One-way functions

Functions that are easy to compute but "difficult" to invert (almost-always).
A OWF $f : \{0,1\}^n \to \{0,1\}^n$ is computable in polynomial time for every input, but to invert efficiently.
Mathematically,

- Easy to compute: $\forall x \longleftarrow \{0,1\}^n, f(x)$ can be computed in poly(n) times.
- Hard to invert: $\forall$ PPT algorithms A, $\exists$ negligible function $negl(.)$ such that -

  . $Pr[Invert_{A,f}(n) = 1] \leq negl(.)$
  $\approx Pr[A(f(x), 1^n \in f^{-1}(f(x))] \leq negl(n)$ over randomly chosen x from the domain.

We'll discuus OWF's in detail later in the course.

We have our first two questions amongst the giant web of implications between the crypto primitves (creatures)-
(To be covered in Chalk and talk sessions 5,6)

1) Existence of PRG $\Rightarrow$ Existence of OWF's
2) Existence of COA SKE's $\Rightarrow$ Existence of OWF's.
The other way round too are true, which will be done later in the course.

References:

[1] Jonathan Katz and Yehuda Lindell *Introduction to Modern Cryptography, second edition.* CRC Press, 2014.

[2] Arpita Patra. *http://drona.csa.iisc.ernet.in/ arpita/Cryptography16.html* Course Materials.