Chapter 3 | Private Key Encryption and Pseudorandomness

Wednesday, April 26, 2023 10:16 AM

Story

- We study the notion of pseudorandomness
  idea, things can "look" random (in a precise sense),
  even though they are not

- secure long messages

- bypass inherent limitations of perfect secrecy by instead achieving a weaker (but sufficient) notion of
  computational security

- Examine computational approach to cryptography first
  and then look at private-key encryption

3.1 A computational approach to cryptography

- Recall
  - Ch 1
    - classical crypto
    - historical ciphers—how to break
  - Ch 2
    - Crypto schemes that are mathematically secure relative to some definition
    - adversary can be computationally unbounded
    - such schemes are called, information theoretically secure
      - because adversary doesn't have enough information to break
        i.e. ciphertext contains no information about the plaintext (assume key is hidden)

- Information theoretic security—stark **contrast** to computational security (aim of modern cryptographic construction)
  - Restrict to private-key (although holds generally)
    - modern schemes can be **broken with enough time** and resource
    - Don't satisfy perfect secrecy
    - Nonetheless, under assumptions, amount of computation is beyond many lifetimes
    - for all practical purpose, this level of security is enough
  - Computational security—**relies on assumptions** (unlike information theoretic)

- Why give up on perfect security?
  - Recall: the key length must be as long as the combined length of all messages ever encrypted using this key
  - Similar results hold for other tasks when information theoretic security is needed
  - To get practical schemes, we abandon perfect security
  - Stress: everything is still rigourous—the definitions are weaker

§ 3.1.1 The basic idea of computational security

- Kerckhoffs
  - had stated many principles
  - Recall: cryptographic designs should be made public
  - Another principle:
    - A [cipher] must be practically, if not mathematically, indecipherable
  - In modern language:
    - not necessary to use a perfectly secret encryption scheme but instead
    - use a scheme that cannot be broken in "reasonable time" with
      any "reasonable probability of success"
    - or concretely, the scheme can be broken (in theory) but
      cannot be broken with probability better than 10^{-30} in 200 years
      using the fastest available supercomputer.

- Here's the framework for making formal statements about computationally secure schemes

- We incorporate two relaxations of the notion of perfect security:
  1. Security is only preserved against *efficient* adversaries, and
  2. Adversaries can potentially succeed with some *very small* probability
     (small enough so that we are not concerned that it will ever really happen).

- We need to make the notions above precise.

- There are two approaches
  - Concrete
  - Asymptotic

Concrete Approach

Idea: quantifies the security of a given scheme
by bounding the max. prob. of success of any adversary
running for at most a specified amount of time.

More formally:
- Let t, ϵ be positive constants with $\epsilon \leq 1$
- Roughly,

  A scheme (t, ϵ)-secure if every adversary running for time at most t succeeds in breaking the scheme with probability at most ϵ

- Why rough?
  - e.g. we need to specify what it means to "break" the scheme
  - E.g.
    - running for at most 200 years, using the fastest hardware, cannot succeed with prob. $\geq 10^{-30}$
    - or with CPU cycles as a measuer—with at most 2^{80} cycles, cannot break with prob. more than 2^{-64}
  - It is instructive to get a feel for the values of t, ϵ
    (that are typical of modern schemes)

Example 3.1
Modern private-key encryption schemes are generally assumed to give almost optimal security in the following sense: when the key has length n, an adversary running in time t (measured in, say, computer cycles) can succeed in breaking the scheme with probability $t/2^n$. (We will see later why this is indeed optimal.) Computation on the order of $t = 2^{60}$ is barely in reach today. Running on a 1GHz computer, 2^{60} CPU cycles require $2^{60}/10^9$ seconds, or about 35 years. Using many supercomputers in parallel may bring this down to a few years.

A typical value for the key length, however, might be $n = 128$. The difference between 2^{60} and 2^{128} is a *multiplicative factor* of 2^{68} which is a number containing about 21 decimal digits. To get a feeling for how big this is, note that according to physicists' estimates the number of seconds since the big bang is on the order of 2^{58}.

An event that occurs once every hundred years can be roughly estimated to occur with probability 2^{-30} in any given second. Something that occurs with probability 2^{-60} in any given second is 2^{30} times *less* likely, and might be expected to occur roughly once every 100 billion years. ♢

Limitations
- What hardware was used (to we account for Moore's Law)
- What algorithm is used (do we account for innovations in algorithm design)
- We can at best say (t, ϵ)-secure but never just secure
  - What range of (t, ϵ) should one use?

Written on
« Wednesday, April 27, 2023

The Asymptotic Approach

Idea
- It is rooted in complexity theory
- Views success probability of the adversary as functions
  of some parameter, rather than concrete numbers
- Specifically, it incorporates a security parameter—an integer n
  - Honest parties use this to initialise the scheme (e.g. when they generate the key)
  - value is publicly known

Then we use the following "definitions"
1. "efficient algorithm" = poly time in n
   NB: It could still be that the adversary runs for much longer than the honest party (e.g. honest party could be linear but adversary takes n^3 for instance)
2. "small prob. of success" = smaller than any inverse poly in n
   i.e. for every contsant c,
   probabilty is smaller than n^{-c} for a large enough n = "negligible"

Notation:
PPT: probabilistic poly time

Informal Definition:
"secure scheme" in the asymptotic setting

A scheme is secure if every PPT adversary succeeds
in breaking the scheme with
only negligible probability

Remarks

1. Example 3.2 shows that the guarantees hold for large enough ns only
2. Typically, the security parameter determines the length of the key
3. Larger the security parameter, more secure the scheme
4. Example 3.3: shows the effect of computers getting faster
   (the honest parties use a higher n when this happens; the scheme stays secure)
5. Relation to (extended) Church-Turing Thesis (recall: it essentially says relative speeds of sufficiently *powerful computing devices* are polynomially related):
   The asymptotic approach does not rely on any type of computer
6. Choosing n is not as obvious—but one can convert it into a concrete security guarantee and proceed (Example 3.4)

Example 3.2
Say we have a scheme that is secure. Then it may be the case that an adversary running for n^3 minutes can succeed in "breaking the scheme" with probability $2^{40} \cdot 2^{-n}$ (which is a negligible function of n). When $n \leq 40$ this means that an adversary running for 40^3 minutes (about 6 weeks) can break the scheme with probability 1, so such values of n are not going to be very useful in practice. Even for $n = 50$ an adversary running for 50^3 minutes (about 3 months) can break the scheme with probability roughly $1/1000$, which may not be acceptable. On the other hand, when $n = 500$ an adversary running for more than 200 years breaks the scheme only with probability roughly 2^{-500}. ♢

Example 3.3
Let us see the effect that the availability of faster computers might have on security in practice. Say we have a cryptographic scheme where honest parties are required to run for $10^6 \cdot n^2$ cycles, and for which an adversary running for $10^8 \cdot n^4$ cycles can succeed in "breaking" the scheme with probability $2^{20} \cdot 2^{-n}$. (The numbers in this example are designed to make calculations easier, and are not meant to correspond to any existing cryptographic scheme.)

Example 3.4
A typical proof of security for a cryptographic scheme might show that any adversary running in time $p(n)$ succeeds with probability at most $p(n)/2^n$. This implies that the scheme is (asymptotically) secure, since for *any* polynomial p, the function $p(n)/2^n$ is eventually smaller than any inverse-polynomial in n. Moreover, it immediately gives a concrete security result for any desired value of n; e.g., the scheme with n fixed to 50 is $(50^2, 50^2/2^{50})$-secure (note that for this to be meaningful we need to know the units of time with respect to which p is being measured). ♢

Notation (technical remark).
Recall: The standard convention in algorithms (and complexity theory) is to measure the running time as a function of the length of input

Thus: To be consistent, we provide the adversary/honest party with the security parameter n in unary as input: 1^n
(by 1^n we mean $11 \ldots 1$, i.e. n many 1s) as input

Why couldn't we give a binary encoding?
Well, in binary, we would need only $\log n$ bits and we want the input to be n bits.

Necessity of the Relaxations

Motivation:

- We made two relaxations in our security
  - adversary was **PPT** (instead of unbounded)
  - success probability was allowed to be **negligible** (instead of zero)
- Are these necessary? Yes—we discuss informally

Discusion:

- Consider an encryption scheme with $|\mathcal{K}| \ll |\mathcal{M}|$, i.e. # keys is much smaller than the number of messages
- Here are two attacks

[If the adversary was required to succeed with probability 1, but could take unbounded time]

Attack 1: *Brute-force search*

- Given a ciphertext c, adversary can decrypt c using all keys $k \in \mathcal{K}$
  Yields a set of messages which cannot possibly exhaust $\mathcal{M}$
  Thus, some information is leaked
- One can even learn the exact key when carrying out a known-plaintext attack
  Suppose $c_1 \ldots c_l$ are known to correspond to messages $m_1 \ldots m_l$
  Run the procedure above until the *same* key k decrypts all messages
  This works with high probability
  [either that or there's another attack] I don't know the details here, just asserting what the book

said].
Allows the adversary to succeed with probability essentially 1 in time linear in $|\mathcal{K}|$.

[If the adversary was required to take polynomial time, but we would say the scheme is broken if it succeeded with non-zero probability]

Attack 2: *Guess*
- Again, suppose $c_1 \dots c_l$ corresponds to the message $m_1 \dots m_l$
  Now simply "guess a key" $k \in \mathcal{K}$ and check if it decrypts all ciphertexts
  It succeeds with probability $1/|\mathcal{K}|$ and takes polynomial time to run

- Consequence:
  - The two relaxations (stated at the beginning) are necessary (mostly a restatement).
    - the adversary should be limited to running in a certain time (else brute-force search is possible)
    - allow the adversary to succeed with a small prob (without considering the scheme broken)
  - The key space must therefore depend on n
    - i.e. one must associated a sequence $\{\mathcal{K}_n\}$ s.t.
      - the key is chosen from $\mathcal{K}_n$ when the security parameter is n
    - if want poly adversaries to achive only neglgible success,
      - the **key space must grow super-polynomially in n**
        otherwise, brute-force can be carried out in poly time &
        random guessing would work with non-negligible prob.

3.1.2 Efficient Algorithms and Negligible Success

Revision of algorithms/complexity theory notation about "poly time" etc.

[This section is not written properly]

Efficient Computation

Randomised algorithm:
gets random bits on a special tape.
equivalently, give randomness in the input

Why poly?
closed under composition
Nothing inherantly special—e.g. could take time $n^{O(\log n)}$

Why probabilistc (and not just deterministic)
First, randomness is essential to crypto (the honest party already needs this to generate keys)
Second, randomness may give more power to computation (and we want to model the adversary as powerful as we can)
Whether randomness helps is not known yet (but complexity theory results indicate it helps)

Generating randomness.
How does a computer "toss coins"
(a) Hardware random number generator
(b) Software (use unpredictable events, like key-strokes, movements of the mouse etc.)
These may not be uniform—need processing further (but these are not well understood)

Caveat: random() in C is not random at all (for crypto purposes) so one must be careful.

Negligible Success Probability

- Just as we consider poly time to feasible
  we consider inverse poly probabilities to be *significant*

- i.e. if an adversary succeeds (in breaking a crypto scheme) with $1/p(n)$ for some poly p
  the scheme would not be considered secure
  [basically, one can usually repeat the algorithm poly many times and make the success prob. constant]

- if it is smaller than $1/p$ for every polynomial, then we say the probability is too small—it is negligible

Definition 3.5 A function f is *negligible* if for every poly $p(\cdot)$
there is an N s.t. for all integers $n > N$
it holds that $f(n) < 1/p(n)$

Equivalent condition: require for all constants $c > 0$,
$f(n) < n^{-c}$

Notation: Negligible functions are typically denoted by negl

E.g. 2^{-n}, $2^{-\sqrt{n}}$, $n^{-\log n}$ are all negligible

Remark: for large enough n, it holds that $2^{-\sqrt{n}} < n^{-\log n}$
(but this happens for $n > 65536$; so for lower ns, the other one might be better)

Remark about negligible:
They are closed under composition!

Proposition 3.7 Let negl and negl′ be negligible funtions
1. Their sum is also a negligible function
2. $p \cdot \text{negl}$ is also negligible where p is a positive polynomial.

Remark:
Point 2 above says that if something happens with negligible prob
then even if the process is repeated poly many times,
the prob. of the event occuring stays negligible

Discussion:
Is it ok to ignore negligible probability?
For all practical purposes, yes!
Why?
Insanes events can happen with non-zero probability
E.g. Asteroid hits earth
E.g. the disk storing the secret of honest parties gets erased

Asymptotic Security: A Summary

[repetition] The general template for asymptotic security is:

A scheme is secure if for every *probabilistic poly-time* adversary $\mathcal{A}$
(carrying out an attack), the probability that $\mathcal{A}$ succeeds (in the attack)
is *negligible*.

NB: Guarantees only hold for large enough $n > N$.

3.1.3 Proofs by Reduction

Story:
- Proving a scheme is secure unconditionally
would mean showing lower bounds on time (needed to break the scheme)
this would require breakthroughs in complexity theory (far out of reach today)
[In particular, this would mean $\mathrm{P} \neq \mathrm{NP}$]
- Instead, assume some low-level problem is hard to solve
and then prove the construction is secure given that assumption

- How do these proofs typically work?
  - By "reduction"
  it shows that
  any PPT adversary $\mathcal{A}$ that
  breaks the scheme
  can be converted into
  another PPT adversary $\mathcal{A}'$ that
  solves the problem assumed to be hard

- This scheme is important—look at it in more detail

Typical template for "reductions"

Assumption:
Some problem X cannot be solved (in some precise sense)
by any PPT algorithm
except with negligible probability

Goal: To prove that some crypto construction Π
is secure (in some precise sense)

Here's the template:
1. Fix some PPT adversary $\mathcal{A}$ attacking Π.
Let $\epsilon(n)$ denote its success probability
2. Construct another PPT adversary $\mathcal{A}'$ that
attempts to solve problem X with
$\mathcal{A}$ as a subroutine

NB: $\mathcal{A}'$ need not know how $\mathcal{A}$ works—it is only required that
$\mathcal{A}$ is trying to break Π

Given an input instance x of problem X
$\mathcal{A}'$ will simulate an execution of Π s.t.

(a) As far as $\mathcal{A}$ can tell, it interacts with Π
More formally,
the view of $\mathcal{A}$ when it is run as a sub-routine by $\mathcal{A}'$
should be distributed identically to (or close)
the view of
$\mathcal{A}$ when it interacts with Π itself.

(b) Further, if $\mathcal{A}$ succeeds in "breaking" the excution of Π
(that is being simulated by $\mathcal{A}'$),
this should allow $\mathcal{A}'$ to solve
the instance x it was given
at least with $1/\text{poly}(n)$ probability.

3. Taken together
2(a) and 2(b) imply that
if $\epsilon(n)$ is not negligible, then
$\mathcal{A}'$ solves problem X with
non-negl probability $\epsilon(n)/p(n)$.
Also, since $\mathcal{A}'$ is efficient (assuming calls to $\mathcal{A}$ are free) and
runs $\mathcal{A}$ (also PPT) as a subroutine,
effectively $\mathcal{A}'$ is PPT.

Then, $\mathcal{A}'$ contradicts the assumtpion (that X is hard to solve).

4. Conclusion: Since X is assumed hard (no PPT algorithm $\mathcal{A}$ can succeed with non-negl probability)
therefore, Π is computationally secure.

Story:
- We now look at some examples to clarify how this works in practice

3.2 A definition of Computationally Secure Encryption

Story:
- We are now ready to present a definition
of computational security for
private-key encryption.

- First,
we define the *syntax* of a private-key encryption
essentially the same as the syntax in Ch 2
except—security parameter
- Later,
we define the notion of "security".

Definition 3.8
A ***private-key encryption scheme*** is a tuple of PPT algorithms $(\text{Gen}, \text{Enc}, \text{Dec})$ s.t.

1. Gen
Input: security parameter 1^n
Output: a key k
Invocation: $k \leftarrow \text{Gen}(1^n)$
Assumption (wlog): $|k| \geq n$ [why without loss of generality? are we just padding with zeros?]
2. Enc
Input: k (a key) and $m \in \{0,1\}^*$ (plaintext; assume m to have length poly in n)
Output: c
Invocation: $c \leftarrow \text{Enc}_k(m)$
3. Dec
Input: k and c
Output: m
Invocation: $m := \text{Dec}_k(c)$ (because Dec is assumed to be deterministic)

Correctness: For every n and k produced by Gen
require that $\text{Dec}_k(\text{Enc}_k(m)) = m$ for all $m \in \{0,1\}^*$

[me: why can't we add the length of the message to the generator?; maybe because it won't work with public key]

Fixed length private-key encryption:
If the scheme is only defined for messages of length $\ell(n)$ for some fixed poly bounded function ℓ

3.2.1 A definition of Security for Encryption

Story:
- There are many ways of defining security for private-key encryption
  - Primarily wrt the power of the adversary
- Begin with the simplest—
security against a **weak form of ciphertext-only attack**
adversary only observes a single ciphertext

Motivating the definition

Recall from Ch 1: Security definition consists of two parts
(a) power of adversary
(b) what does it mean to "break" the scheme?

(a) adversary (weak form of cipher-text only attack)
- Sees only ciphertext (cannot choose which—like its eavesdropping)
- Wishes to crack it
- It is PPT

(Recall: we never assume any restrictions on the strategy; crucial because one cannot predict all possible strategies to give a meaningful notion of security)

Story:
- Recall (from Section 1.4.1), we said we want to ensure "no partial information about the plaintext" can be learnt
  It is formalised as **"Semantic Security"**
  This is involved—not covered here
- Instead, we work with an equivalent definition
  in terms of **indistinguishabilty**

- **Recall:** $\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi}$
  We make two modifications to the requirements on the adversary
  1. adversaries run in PPT
  2. even if the adversary succeeds with $\frac{1}{2}+\mathrm{negl}$, we don't consider the scheme broken
     (must by non-negl more than $1/2$)

  Two modifications to the security game
  1. The game depends on the security parameter n
     As do success probabilities etc.
  2. Require m_0, m_1 are of equal length

NB: The book is using the same name, i.e. PrivK, as the information theoretic version.

Remark: (About point 2 above)—this is necessary because our scheme allows us
to encrypt messages of arbitrary length
Otherwise, from the length of the ciphertexts, the messages would become
trivially distinguishable.

[Me: Why didn't we have this issue in the perfect secrecy case? In the formalism at least, we simply fixed the number of keys and number of messages; so the bits in the keys determined the length of the ciphertext. This is no longer the case; maybe for fixed length private key encryption schemes]

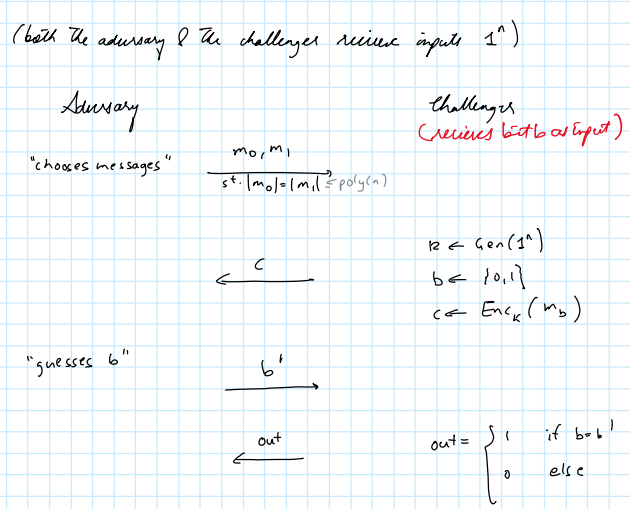
NB: Most schemes don't hide the length of the message (not discussed further)

Notation: $\Pr[\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi}(n)=1]$ denotes the probability that the adversary $\mathcal{A}$ outputs 1 in the game

Indistinguishability in the presence of an eavesdropper

Story: We now give the formal definition.

Adversarial Indistinguishability Game $\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi}(n)$:



NB: If Π is a fixed-length scheme with parameter ℓ then
in the game, require $m_0, m_1 \in \{0,1\}^{\ell(n)}$.

Definition 3.9: A private key scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ has
indistinguishable encryptions in presence of an eavesdropper if
for all PPT adversaries $\mathcal{A}$, there is a negligible function negl s.t.

$$\Pr[\text{PrivK}^{eav}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \text{negl}(n)$$

where the probability is over randomness of $\mathcal{A}$,
and over that used in the game (e.g. by Gen, Enc etc.)

Remarks:

- Note the security covers "all strategies"
- Adversary is allowed to choose m_0, m_1 —
  e.g. it could know the message is "attack today" or "don't attack"
  but the security guarentee guarentees
  it cannot know which among the two was sent
- No restriction on the length of messages (as long as lengths are same and poly in n)

Equivalent formulation

Story:

- There is another way of formalising that
  - "every adversary $\mathcal{A}$ behaves the same way
    when it's given encryptions of m_0 and m_1 (of same length)"
- NB: $\mathcal{A}$ outputs a bit so by "behave" we can imagine,
  the probability of outputting 1
- Therefore, one formalises "same behaviour"
  by requiring the probability of outputting 1
  to be the same in both cases

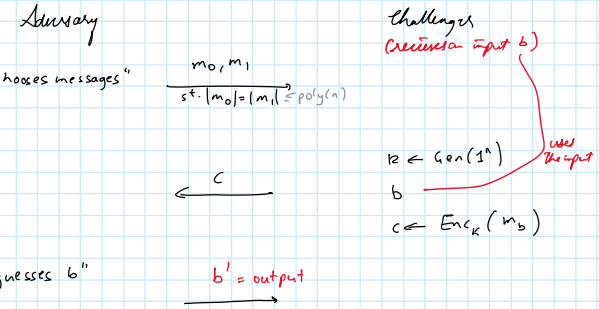
- To formalise this, we consider a close variant of the $\text{PrivK}^{\text{eav}}(n)$ game

Informal Definition: $\text{PrivK}^{eav}_{\mathcal{A},\Pi}(n,b)$

- Same as game $\text{PrivK}^{\text{eav}}(n)$, except that the challenger recieves b as an input (instead of choosing it randomly).

- The output of the game now, is the output of $\mathcal{A}$

(both the adversary & the challenger recieve inputs 1^n)



Definition 3.10 A private-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ has
indistinguishable encryptions in the presence of an eavesdropper if
for all PPT machines $\mathcal{A}$, there's a negligible funtion negl s.t.

$$\left| \Pr\left[\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi}(n,0) = 1\right] - \Pr\left[\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi}(n,1) = 1\right] \right| \leq \mathrm{negl}$$

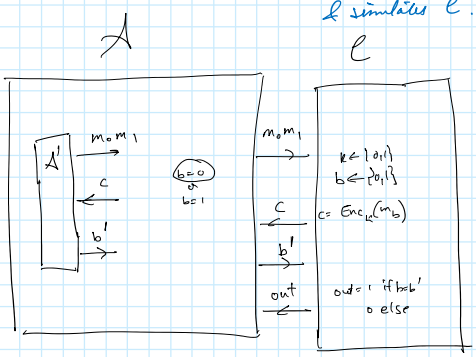
Claim: Definitions 3.9 and 3.10 are equivalent.

(trivial direction)

(1) Suppose 3.10 can be broken by $\mathcal{A}'$ against $\underline{e'}$

challenger for eq^ 3.10.

Show 3.9 can be broken using $\mathcal{A}$ that uses $\mathcal{A}'$ & simulates e'.



$$\underbrace{\Pr[G(\mathcal{A}', e'(0)) = 0]}_{P_{0,0}} \geq \underbrace{\Pr[G(\mathcal{A}', e'(1)) = 0]}_{P_{1,0}} + \underbrace{\text{non-negl.}}_{\eta}$$

$$P_{1,1} = 1 - P_{1,0}$$

$$P_{00} \geq P_{10} + \eta$$

$$-P_{10} \geq -P_{00} + \eta$$

$$\Pr[G(\mathcal{A}, e).\mathrm{out} = 1] = \underbrace{\tfrac{1}{2}}_{b=0} \cdot P_{0,0} + \underbrace{\tfrac{1}{2}}_{b=1} P_{1,1}$$

$$= \tfrac{1}{2}(P_{00} + P_{11})$$

$$= \tfrac{1}{2}(P_{00} + 1 - P_{10})$$

$$\geq \tfrac{1}{2}(1 + \cancel{P_{10}} - \cancel{P_{00}} + \eta)$$

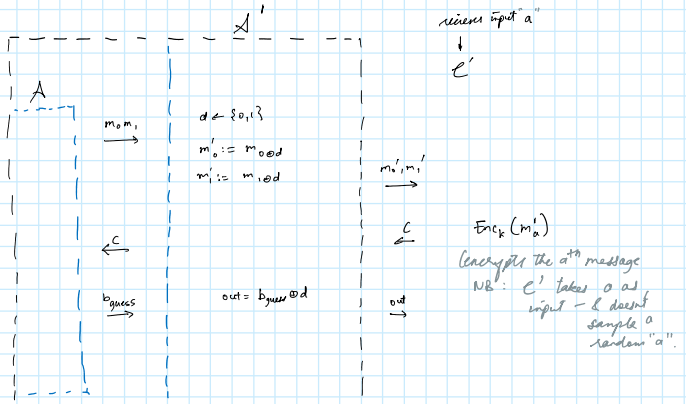
$$\geq \tfrac{1}{2} + \tfrac{1}{2}\eta$$

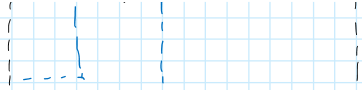
(2) Other direction

Proof Idea:

- Suppose $\mathcal{A}$ wins w.p. $\frac{1}{2} + \underbrace{\eta}_{\text{non-negl}}$ in Def^ 3.9.
- We construct an $\mathcal{A}'$ that interacts with the challenger e' in Def^ 3.10.
- NB
  (a) $\mathcal{A}'$ uses $\mathcal{A}$
  (b) $\mathcal{A}'$ simulates a challenger e for $\mathcal{A}$ (that behaves as $\mathcal{A}$ expects)

Construction for $\mathcal{A}'$





sample a random "a".

NB: $c = \begin{cases} Enc_k(m_0) & \text{w.p } \frac{1}{2} \\ Enc_k(m_1) & \text{w.p } \frac{1}{2} \end{cases}$ ∴ $\mathcal{A}$ sees the "challenge" it expected.

∴ $Pr[\underbrace{\text{output of } \mathcal{A}}_{\text{out}} = a] = \frac{1}{2} + \eta$ i.e. $\mathcal{A}'$ has a way of "knowing" a.

∴ the algorithm's output is correlated to a

$Pr[\mathcal{A}' \to 1 ; c_0'] \quad \frac{1}{2} - \eta$

$Pr[\mathcal{A}' \to 1 ; c_1'] \quad \frac{1}{2} + \eta$

∴ $\mathcal{A}'$ breaks Def 3.10.

where c chooses b at random (message to encrypt)

security condition (3.9)

$\exists$ negl s.t. $\forall$ $\mathcal{A}$ PPT it holds that

$Pr[\mathcal{A} \text{ wins}] \leq \frac{1}{2} + negl.$

(negation)

$\forall$ negl $\exists$ $\mathcal{A}$ PPT s.t.

$Pr[\mathcal{A} \text{ wins}] > \frac{1}{2} + negl$

$\Leftrightarrow$ $\exists$ η non-negl, $\exists$ $\mathcal{A}$ PPT s.t.

$Pr[\mathcal{A} \text{ wins}] = \frac{1}{2} + \text{non-negl}.$

3.2.2 Properties of the Definition

Story:

- We motivated the definition of secure encryption
  by saying it should be *infeasible* to
  learn any partial information about
  the plaintext from the ciphertext
- Our discussion so far, does not seem to have anything to do with it
  - that definition is called *semantic security* (quite involved)
- We now *prove two claims* that show
  weaker versions of semantic security
  (starting from our definition)
- Later, we present the "essence" of the definition of semantic security (details in Ch 5)

Claim 3.11

Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a private-key encryption scheme that
has *indistinguishable encryptions*
in the presence of an eavesdropper.

Then,
for all PPT adversaries $\mathcal{A}$ and all i, there is a negligible function negl s.t.

$$\Pr[\mathcal{A}(1^n, \text{Enc}_k(m)) = m^i] \leq \frac{1}{2} + \text{negl}(n),$$

where m is *chosen uniformly* at random and
probability is taken over
- the random coins of $\mathcal{A}$
- the choice of m and
- the key k
- (and any randomness in the encryption process)

(I am guessing that m^i means the ith bit of m)

Proof idea:

if it is possible to guess the ith bit of m given $\text{Enc}_k(m)$
then it is also possible to distinguish between
encryptions of m_0 and m_1
where the ith bit of m_0 equals 0 and
that of m_1 equals 1.

Formally, we show
if there is an adversary $\mathcal{A}$ that
can guess the ith bit of m given $\text{Enc}_K(m)$ with
prob at least $\frac{1}{2} + \epsilon$ for some function ϵ
then there is an adversary $\mathcal{A}'$ that
succeeds in the security game for Π
with probability $\frac{1}{2} + \epsilon$.

Thus, ϵ must be negligible.

Proof:

Let: $\mathcal{A}$ be a probabilistic poly-time adversary
Defn:

$$\epsilon(n) := \Pr\left[\mathcal{A}\left(1^n, \mathrm{Enc}_k(m)\right) = m^i\right] - \frac{1}{2}$$

where m is chosen uniformyl at random from $\{0,1\}^n$.

(Me: What about i ? Is ϵ a function of i?
It's me again: i is fixed; negation of the statement says there is some i for which there is an adversary that does better than random guessing)

Notation: We drop 1^n for visual clarity

Defn: For $n \geq i$,

I_0^n be the set of all strings of length n whose i^{th} bit is 0
I_1^n be the set of all strings of length n whose i^{th} bit is 1

NB: $\Pr\left[\mathcal{A}\left(\mathrm{Enc}_k(m)\right) = m^i\right] = \frac{1}{2} \cdot \Pr\left[\mathcal{A}(\mathrm{Enc}_k(m_0)) = 0\right] + \frac{1}{2} \cdot \Pr\left[\mathcal{A}(\mathrm{Enc}_k(m_1)) = 1\right]$ (*)

where m_0 is chosen uniformly from I_0^n (and similarly m_1 and I_1^n)

(NB: prob. that a uniformly random string is in I_0^n or I_1^n is the same.

Story: Now consider the following adversary $\mathcal{A}'$ (designed to break Π using $\mathcal{A}$) who eavesdrops on the encryption of a single message:

Defn: Adversary $\mathcal{A}'$

a. On input 1^n,
   choose $m_0 \leftarrow I_0$ and $m_1 \leftarrow I_1$ uniformly at random from the indicated sets
   (Notation: I started dropping the n superscript)
   Output m_0, m_1
b. On ciphertext c,
   invoke $\mathcal{A}$ on input c
   Output $b' = 0$ if $\mathcal{A}$ outputs 0 and
   output $b' = 1$ otherwise.

NB1: (trivial) $\mathcal{A}'$ runs in poly time since $\mathcal{A}$ does

NB2: (using the definition of the security game for Π), it holds that $b' = b \Leftrightarrow$ the game outputs 1

NB3:

$$\Pr\left[\mathrm{PrivK}^{eav}_{\mathcal{A}',\Pi}(n) = 1\right] = \Pr\left[\mathcal{A}(\mathrm{Enc}_k(m_b)) = b\right] \quad \text{(using (*) above)}$$

$$= \frac{1}{2} \cdot \left[\mathcal{A}(\mathrm{Enc}_k(m_0)) = 0\right] + \frac{1}{2} \cdot \left[\mathcal{A}\left(\mathrm{Enc}_k(m_1)\right) = 1\right]$$

$$= \Pr\left[\mathcal{A}\left(\mathrm{Enc}_k(m)\right) = m^i\right]$$

$$= \frac{1}{2} + \epsilon(n).$$

($n > i$)

By the assumption that Π is secure, ϵ must be negligible

(NB: having $n > i$ is not an issue—the guarantee holds asymptotically)

□

Story:

[TODO: Write this better; two generalisations
- Make the distribution over m different from uniform
- Allow arbitrary poly time functions
- Clarify how we need to change the notion slightly

]

- We now want to show no PPT adversary can
  learn any function of the plaintext
  given the cipher text
- This is problematic to define!
  E.g. consider $f(m) = m^i$

Recall: showed up, when m is uniform,
no adversary can succeed at this
wp more than 1/2 + negl
(given the ciphertex)

- How do we generalise?
  - One direction:
    Have the "bit prediction" impossibility for all distributions

    However, if m is not uniform, then it may be possible to predict the ith bit

    E.g. distribution s.t. ith bit of m is fixed; then i is easy to predict

  - What we want is,
    if
    an adversary receiving $c = \mathrm{Enc}_k(m)$
    can compute $f(m)$ for some function f,
    then
    there exists an adversary that
    can compute $f(m)$ (with the same probibility of being correct)
    without given the ciphertext c
    (but only the prior distribution over m)

- In the next claim, we show the above when
  m is chosen uniformly at random from set $S \subseteq \{0,1\}^n$
  (in fact, we consider an infinite set $S \subseteq \{0,1\}^*$
  and the set of interest $S_n := S \cap \{0,1\}^n$, subset of n bit strings from S)

  We assume S_n is
  never empty and
  efficiently samplable (i.e. given 1^n, there's a PPT algorithm sampling from S_n)

Claim 3.12

Let Π be a private-key encryption scheme that has
indistinguishable encrptions (in the presence of an eavesdropper).
Then,
for every PPT adversary $\mathcal{A}$ there is
PPT algorithm $\mathcal{A}'$ s.t.
for every PPT computable function f and every efficiently samplable set S
there is a negligible function negl s.t.

$$\left| \Pr\left[\mathcal{A}(1^n, \mathrm{Enc}_k(m)) = f(m)\right] - \Pr\left[\mathcal{A}'(1^n) = f(m)\right] \right| \leq \mathrm{negl}(n)$$

where m is chosen uniformly at random from $S_n := S \cap \{0,1\}^n$,
and the probabilities are taken over
- the choices of m
- the key k
- and any other random coins used by $\mathcal{A}, \mathcal{A}'$ and
- the encryption process

Proof (sketch).

- Informal sketch
- Assume Π has indistinguishable encryption
  - This means no PPT can distinguish between
    $\mathrm{Enc}_k(m)$ and $\mathrm{Enc}_k(1^n)$ for any $m \in \{0,1\}^n$.
- We claim that the probabilities of the following two events are almost the same
  - $\mathcal{A}$ (correctly) computes $f(m)$ given $\mathrm{Enc}_k(m)$
  - $\mathcal{A}$ (correctly) computes $f(m)$ given $\mathrm{Enc}_k(1^n)$
- Why?
  otherwise $\mathcal{A}$ could be used to distinguish encryptions of m and 1^n
  (here's the distinguisher (algorithm for breaking Π):
  sample m uniformly from S_n and
  sends $m_0 = m$ and $m_1 = 1^n$ to the challenger
  Given c (that is an encryption of either m or 1^n),
  Invoke $\mathcal{A}$ on c
  Output 0 $\Leftrightarrow$ $\mathcal{A}$ outputs $f(m)$

  If $\mathcal{A}$ outputs $f(m)$ wp non-negligibily different from
  whether it got encryption of m or 1^n
  then security of Π is violated
)

- Define $\mathcal{A}'$ is designed to compute $f(m)$ with input only 1^n. [grammar]
  Here's the construction:
  - Sample $k' \leftarrow \mathrm{Gen}(1^n)$ and compute $\mathrm{Enc}_{k'}(1^n)$
  - Invoke $\mathcal{A}$ on input $\mathrm{Enc}_{k'}(1^n)$ and output whatever it outputs

- Claim:
  Probability that $\mathcal{A}'$ computes $f(m)$ correctly
  $\approx$
  probability $\mathcal{A}$ computes $f(m)$ correctly given $\mathrm{Enc}_k(1^n)$
  (here k is generated externally; nothing to do with the k' generated by $\mathcal{A}'$)

  (and by the "claim" above)
  $\approx$
  probability $\mathcal{A}$ computes $f(m)$ correctly given $\mathrm{Enc}_k(m)$

□

Semantic Security

Story:
- Full definition of semantic security
  is considerably more general (than what we proved)
- In particular
  arbitrary distributions over plaintext and
  arbitrary *"external" information* about
  chosen plaintexts
  are also taken into consideration

Notation:
- As above,
  denote by f the function of the plaintext (that needs to be computed)
- Denote by h
  the *"external" knowledge* the adversary may have
  regarding the plaintext

Story:
- In addition to the encryption of m,
  the adversary is also given $\mathrm{h}(\mathrm{m})$

TODO:
- there is an additional function h of messages that is given as an input
- Distributions are now arbirtray but samplable
  - for every n, there's a distribution X_n from which messages are sampled
  - Constraint: Messages have equal length

Definition 3.13

A private key encryption scheme Π is
semantically secure in the presence of an eavesdropper
if for every PPT algorithm $\mathcal{A}$
there is a PPT algorithm $\mathcal{A}'$ s.t.
for every efficiently-samplable distirbution
$X = (X_1 \dots)$ and
all PPT computable functions f and h,
there is a negligible function negl such that

$$\left| \Pr\left[\mathcal{A}\left(1^n, \mathrm{Enc}_k(m), h(m)\right) = f(m)\right] - \Pr\left[\mathcal{A}'\left(1^n, h(m)\right) = f(m)\right] \right| \leq \mathrm{negl}$$

where m is chosen according to X_n, and
the probabilities are taken over
- the choices of m and the key k
- and any randomness used by $\mathcal{A}, \mathcal{A}'$ and
- the encryption process

Theorem 3.14

A private-key encryption scheme has **indistinguishable encryptoins**
in the presence of an eavesdropper
iff
it is **semantically secure** in the presence of an eavesdropper

Remark:

- A similar equivalence is known for all definitions of indistinguishability that we present
- We therefore use indistinguishability as our worknig definition

3.3 Pseudorandomness

Story:

- Instead of going into constructions of secure encryption schemes,
  we introduce the notion of pseudorandomness
- Pseduorandomness
  - plays a fundamental role in cryptography in general, and
  - private-key encryption in particular
- Loosely speaking
  - a pseodorandom string is a string that
    looks like a uniformly distributed string
    as long as the entity that is looking
    runs in poly time
  - Just as indistunguishability is a relaxation of perfect secrecy
    pseudorandomness is a computational relaxation of
    true randomness

- Conceptual point (important)
  - No fixed string can be said to be pseudorandom
    in the same way that no fixed string is random
  - Rather, pseudorandomness refers to
    distributions over strings
  - When we say distribution $\mathcal{D}$ over strings (of length ℓ) is pseudorandom
    we mean $\mathcal{D}$ is indistuingashable from
    the uniform distribution over strings of length ℓ
    (strictly, since we are in the "asymptotic setting", we'll need
    a sequence of distributions,
    one for each value of the security parameter;
    we ignore this for now)
  - More precisely,
    - it is infeasible for any poly-time alg to tell
      whether it is given a string
      sampled from $\mathcal{D}$ or
      from an ℓ bit string, chosen uniformly at random

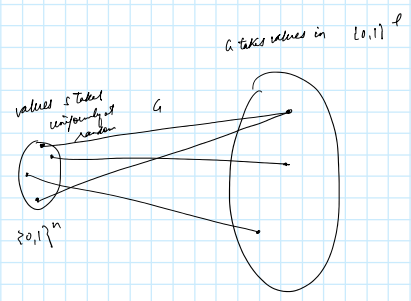
- The specific distributions $\mathcal{D}$ we will be interested in are those
  - that are defined by
    a short random seed $s \leftarrow \{0,1\}^n$ uniformly at random
  - and then outputting $G(s) \in \{0,1\}^\ell$.

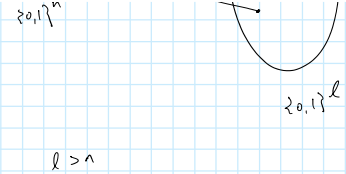
- The distribution $\mathcal{D}$ thus defined,
  outputs the string $y \in \{0,1\}^\ell$ w.p. exactly

$$\frac{|\{s \in \{0,1\}^n \mid G(s) = y\}|}{2^n}$$

which will, in general, not be the uniform distribution.

NB: In fact, we will only be interested in the case
$\ell > n$ in which case,
the distribution will be
very far from uniform





- Even given the above discussion
  - we frequently abuse the notation and
    - call a string sampled from a uniform distribution a "random string" and
    - call a string sampled from a pseudorandom distribution a "pseudo-random string"

- [skipped a remark]

random string". This is only useful shorthand, and it should be noted in particular that if $y = G(s)$ for some s then in fact y can occur as either a random or a pseudorandom string.

- Intuition about
  - why pseudorandomness helps in
    - construction of secure private-key encryptions
    - Simplistic level
      - if a ciphertext looks random,
        - no adversary can learn anything about the plaintext
      - To some extend, this is the intuition behind the
        - perfect secrecy of the one time pad
      - (Recall: ciphertext was uniformly distributed,
        - assuming key is unknown—thus reveals nothing about the plaintext)
      - Caveat: this is only intuitive

      - One time pad worked by XORing
        - the random string (key) with the plaintext

      - If one uses a pseudorandom string instead
        - this should not make any difference to
          - a poly-time adversary
            - thus
              - security should still hold.

- We see below
  - this idea can be implemented.
- But why bother with this?
  - So that one can use a shorter key (seed) and still encrypt long messages

Pseudorandom generators.

- We now formally define pseudorandom generators

Verbose Definition:
A pseudorandom generator is
a deterministic algorithm that
receives a short truly random seed and
stretches it into a long string
that is pesudorandom

Informal: it takes a short amount of randomness and
streches it into a long string that is pseudorandom

Notation:
We set n to be the length of the seed
that is input to the generator and
$\ell(n)$ is the output length

Remark: Clearly, the generator is only interesting if $\ell(n) > n$
(otherwise, it does not generate any new "randomness")

Definition:
Let
- $\ell(\cdot)$ be a polynomial and
- G be a deterministic poly-time alg,

s.t.
for any $s \in \{0,1\}^n$,
$G(s) \in \{0,1\}^{\ell(n)}$.
We say G is a **pseudorandom generator** if
the following two conditions hold:

(1) Expansion:
For every n, $\ell(n) > n$

(2) Pseudorandomness:
For all PPT distinguishers D
there is a neglgible function negl s.t.

$$\left|\Pr[D(x)=1] - \Pr[D(G(s))=1]\right| \leq \text{negl}(n)$$

where
- r is chosen uniformly at random from $\{0,1\}^{\ell(n)}$,
- the seed s is chosen uniformly at random from $\{0,1\}^n$
- probabilities are taken over random coins used by D and the choice of r and s.

The function $\ell(\cdot)$ is called the **expansion factor** of G.

Discussion

- Notice that the output of pseudorandom generator is actually very far from random
  (e.g. for $\ell(n) = 2n$, there are exponentially many strings that never appear, for any fixed G.)

- Thus, it is trivial to distinguish b/w a random string and a pesudorandom string given unlimited time
  Construction for a distinguisher:
  D outputs 1 iff there is some $s \in \{0,1\}^n$ s.t $G(s) = w$

  It is not hard to see that
  $$\left|\Pr[D(r) = 1] - \Pr\left[D\left(G(s)\right) = 1\right]\right| = 1 - 2^{-n}$$

  when $\ell(n) = 2n$.

  This is huge!
  This is called a **brute force attack** because it tries all possible seeds

  But it applies to all generators, irrespective of how they work

- The discussion above shows, G is very far from random
- But the key point is, PPT distinguishers don't have enough time to distinguish them if G is a pesudorandom generator
- This means that pseudorandom strings are just as good as good as truly random ones as long as the seed is kept secret (and we restrict to PPT adversaries).

The seed and its length

Discussion
- The seed for a pseudorandm generator must be chosen uniformly at random and be kept entirely secret from the distinguisher
- Another point
  s must be long enough so that no "efficient" algorithm has time to traverse through all seeds

- Technically, this is not an issue—seeds are taken to be n bit strings so there are 2^n seeds

Existence of pseudorandom generators

- Question: Does any entity satisfying Definition 3.15 even exists?
  - Unfortunately, we don't have an unequivocal proof of existence of pseudorandom generators
  - Neverthless, they are believed to exist
    Why?
    They can be constructed from a rather weak assumption
    one-way functions exist
    These in turn can be constructed from

problems such as integer factoring
(that have been studied for years and
no poly-time algorithm is known)

- Discussed in greater detail in Chapter 6

- In practice, various constructions believed to act as
  pseurandom generators are known

3.4 Constructing Secure Encryption Schemes

3.4.1 A secure fixed-length encryption scheme

We are now ready to construct
fixed-length encryption schemes that
has indistinguishable encryptions
in the presence of an eavesdropper

The encryption scheme is
very similar to the one-time pad encryption scheme (as we alluded to earlier)

The encryption scheme

Let G be a pseudorandom generator
with expansion factor ℓ (i.e. $|G(s)| = \ell(|s|)$)

Recall, an encryption scheme $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ is made of three algorithms

Construction 3.16

Define a private-key encryption scheme for messages of length ℓ as follows:

1. Gen: on input 1^n,
   choose $k \leftarrow \{0,1\}^n$ uniformly at random
   output k
2. Enc: on input $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$,
   output the ciphertext
   $c := G(k) \oplus m$
3. Dec: on input $k \in \{0,1\}^n$ and ciphertext $c \in \{0,1\}^{\ell(n)}$,
   output the plaintext
   $m := G(k) \oplus c$.

We now prove the encryption scheme satisfies
the security of indistinguishable encryption in the presence of an eavesdropper
under the *assumption* that
G is a pseudorandom generator

NB: The claim is not unconditional—we prove it by reduction
We reduce the security of the encryption scheme
to the properties of G as a pseudorandom generator

Story: This is an important technique. Discussed further later.

Theorem 3.17

If G is a pseudorandom generator then
Construction 3.16 is a
private-key encryption that has
indistinguishable encryptions in the presence of an eavesdropper

Proof

Idea

- Let Π denote the scheme in Construction 3.16
- We show
  if there's a PPT adversary $\mathcal{A}$ for which Definition 3.9 breaks then
  we can construct a PPT algorithm that

distinguishes G from a truly random string.

- Intuition:
  - If the encryption in 3.16 used truly random
    instead of pseudorandom
    then the scheme would be just the one-time pad
  - And $\mathcal{A}$ would be unable to guess
    which message was encrypted (wp greater than 1/2)
  - So if Definition 3.9 does not hold
    then $\mathcal{A}$ must implicitly be distinguishing
    output of G from a random string
  - The reduction below, makes it explicit

Reduction:

- Let $\mathcal{A}$ be PPT and define ϵ as

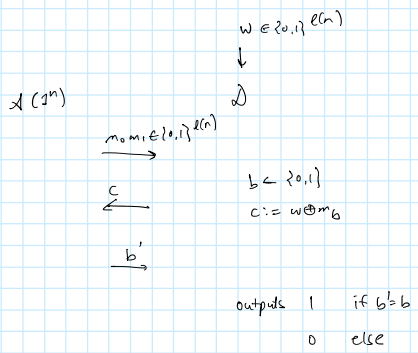
$$\epsilon(n) := P_r\left[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1 \right] - \frac{1}{2}$$

- We use $\mathcal{A}$ to construct a distinguisher D
  for G (the pseudorandom generator)
  that succeeds with prob. $\epsilon(n)$
- What does it mean to succeed?
  - D is given a string w and it has to determine
    whether w is a random string OR
    $w := G(k)$ for k uniformly randomly chosen
  - D simulates the "challenger" for $\mathcal{A}$
    (in the security game for Π)

If $\mathcal{A}$ succeeds, D guesses w is pseudorandom
otherwise, it guesses w is random

Distinguisher D:
Input: $w \in \{0,1\}^{\ell(n)}$

1. Run $\mathcal{A}(1^n)$ to obtain $m_0, m_1 \in \{0,1\}^{\ell(n)}$
2. Choose a random bit $b \leftarrow \{0,1\}$.
   Set $c := w \oplus m_b$
3. Give c to $\mathcal{A}$ and obtain output b'.
   Output 1 if $b' = b$ and
   output 0 otherwise.



Story:
Before analysing the behaviour of D,
we define a modified encryption scheme Π'
that is exactly the one-time pad encryption scheme
except that
we now incorporate a security parameter
that determines the length of the message
to be encrypted.
I.e. $\text{Gen}'(1^n)$ outputs a
completely random key k of length $\ell(n)$ and
the encryption of a message $m \in \ell(n)$
using the key $k \in \{0,1\}^{\ell(n)}$
is the ciphertext $c := k \oplus m$
(decryption can be performed analogously but not needed here)

NB: By perfect secrecy of the one-time pad
we have that

$$\Pr\left[\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi'}(n)=1\right] = \frac{1}{2} \qquad [3.3]$$

The main observations are as follows:

1. if w is chosen uniformly at random from $\{0,1\}^{\ell(n)}$, then
   the view of $\mathcal{A}$ when run as
   a sub-routine by D is
   distirbuted identically to
   the view of $\mathcal{A}$ in experiment.
   $\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi'}(n)$

   (this is because $\mathcal{A}$ is given a ciphertext $c = w \oplus m_b$
   where w ∈ $\{0,1\}^{\ell(n)}$ is a completely random string

2. If w equals $G(k)$ for $k \leftarrow \{0,1\}^n$ then
   the view of $\mathcal{A}$ when run as
   a sub-routine by D is
   distributed identically to
   the view of $\mathcal{A}$ in experiment,
   $\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi}(n)$

   (this is because $\mathcal{A}$ is given a ciphertext $c = w \oplus m_b$
   where w = G(k) for $k \leftarrow \{0,1\}^n$).

NB: It follows that for $w \leftarrow \{0,1\}^{\ell(n)}$

$$\Pr[D(w)=1] = \Pr\left[\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi'}(n)=1\right] = \frac{1}{2}$$

where the second equaity follows from equation (3.3).

NB2: In constrast, when $w = G(k)$ for $k \leftarrow \{0,1\}^n$, one has

$$\begin{aligned}\Pr[D(w)=1] &= \Pr[D(G(k))=1] \\ &= \Pr\left[\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi}(n)=1\right] \\ &= \frac{1}{2} + \epsilon(n)\end{aligned}$$

(by definition of ϵ)

NB3: Thus, (changing the variables slightly), one has

$$\left|\Pr[D(w)=1] - \Pr\left[D\left(G(s)\right)=1\right]\right| = \epsilon(n)$$

where $w \leftarrow \{0,1\}^{\ell(n)}$ (uniform) and $s \leftarrow \{0,1\}^n$ (again, uniform).

NB4: Since G is a pseudorandom generator (by assumption),
it must be that ϵ is negligible.

NB5: Since ϵ is negligible, Π has
indistinguishable encryptions in the presence of an eavesdropper.

□

Story:

- It is easy to get lost in the details of the proof
- Did we really gain anything over the one time pad?
- Afterall, Construction 3.16 also encrypts by XORing
- The point is that ℓ can be much larger than n
- In particular,
  it is possible to encrypt megabytes long file
  using only a 128 bit key
- Stark contrast to perfect secrecy—there we would need the key to be as long as the message

Reductions—a discussion.

- We do not prove unconditionally that
  Construction 3.16 is secure
- Rather, we prove it is secure
  under the assumption that G is a pseudorandom generator
- This is of great importance because
  - We don't know how to prove existence of such a scheme
    such a proof would have imlpications on complexity/algorithms [already mentioned]
  - Given this, reducing security to a lower-level primitive
    has a bunch of advantages
  - In general,

- it is easier to design lower level primitives (e.g. pseudorandom gen)
  (than the higher level ones, (e.g. private key encr))
- also, easier to check lower level definitions
- NB: doesn't mean lower level schemes are easy to construct
  but likely easier than directly constructing a higher level primitive

3.4.2 Handling Variable length messages

Story:

- In the previous section,
  we only allowed encryptinos of
  fixed-length messages
  (i.e. for any fixed n, only $\ell(n)$ lengthed messages could be encrypted)

- This is easily fixed by
  using a variable length pseudorandom generator (defined next)

Variable output-length pseudorandom generators

- We woud like G to receive two inputs
  - s the seed
  - ℓ the length (ℓ in unary; just as the security parameter)
- It should output a pseudorandom string of length ℓ

Definition 3.18

A deterministic poly-time algorithm G is a
variable output-length pseudorandm generator
if the following hold:

1. Input:
   s be a string and
   $\ell > 0$
   Then $G(s, 1^\ell)$ outputs a string of length ℓ

2. For all s, ℓ, ℓ' with $\ell < \ell'$, the
   string $G(s, 1^\ell)$ is a prefix of $G(s, 1^{\ell'})$

3. Define $G_{\ell''}(s) := G(s, 1^{\ell''(|s|)})$.
   Then, for every polynomial $\ell''(\cdot)$ it holds that
   G_ℓ is a pseudorandom generator with
   expansion factor ℓ.

I'm guessing this means that
if $G(s, 1^\ell) = a_1 \dots a_\ell$
$G(s, 1^{\ell'}) = b_1 \dots b_\ell \dots b_{\ell'}$
then $b_1 \dots b_\ell = a_1 \dots a_\ell$

Remark:

- *Any standard pseudorandom generator*
  (as in Definition 3.15)
  can be converted into a variable output-length one.

- Construction 3.16 can be naturally modified: encrypt using $c := G(k, 1^{|m|}) \oplus m$
  (and similarly decryption)

3.4.3 Stream Ciphers and Multiple Encryptions

Story:

- (variable length) pseudorandom generator is also called a "stream cipher"
  because it can generate a stream of pseudorandom bits
- A "secure stream cipher" means a secure variable length pseudorandom generator

- (there is some confusion in the literature—sometimes the stream cipher
  refers to the full private key encryption scheme)

Stream ciphers in practice

- Many practical constructions are known
- They are typcially, very fast

- E.g. RC4
  Widely considered to be secure (when used appropriately)

- Remark:
  - Security of practical stream ciphers is not well understood
    - particularly in comparison to block ciphers (introduced later in this chapter)
  - Thus, there's no standardised popular stream cipher
    who's security has not come into question
  - E.g. "Plain" RC4 that was considered secure
    now is known to have weakenesses

    - The first few bits are known to be biased
    - WEP encryption can be broken because of this
    - If RC4 is to be used, one must discard the first 1024 bits of the output stream

- Linear Feedback Shift Registers (LSFRs)
  - have historically also been popular as stream ciphers
  - However, they have been shown to be horribly insecure
    (to the extend that the key can be completely recovered,
    given enough bytes of the output)
  - It should never be used today.

- Therefore, the book advocates the use of Block ciphers
  - they can be used to build stream ciphers
  - However, whily they are efficient enough
    they're not as efficient as dedicated stream ciphers

Security for Multiple Encryptions

Story:

- Definition 3.9 (and all our discussion so far)
  has dealt with the case where
  the adversary recieves a single ciphertext

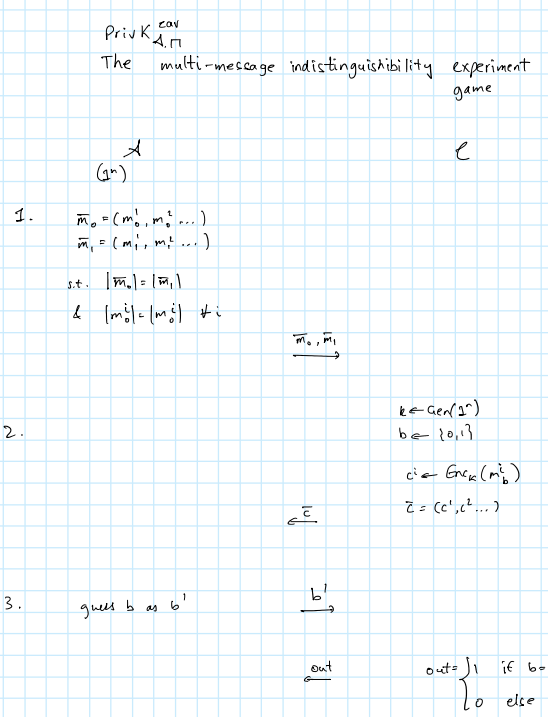
- In reality, many ciphertexts will be exchanged
  and the adversary can see them

- Thus, it's important
  to ensure security can be established
  in this setting as well

- Let's first give a security definition.

$PrivK^{eav}_{A,\Pi}$

The multi-message indistinguishibility experiment
game



Story:

The definition remains unchanged (except that we refer to the new game/experiment)

Definition 3.19

A private key encryption scheme Π has
indistinguishable multiple encryptions in the presence of an eavesdropper
if for all PPT $\mathcal{A}$, there is
a negligible function negl s.t.

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

where the probability is taken over .... [the usua]

Remark

- Security for a *single* encryption (as in Definition 3.9)
  does NOT imply security under *multiple* encryptions.

Claim 3.20

There exist private-key encryption schemes that are secure wrt Def 3.9 but
are not secure wrt Def 3.19.

Proof

- Construction 3.16 is not secure for multiple encryptions
- Same reason why multiple use of one time pad
- Here's how you construct $\mathcal{A}$
  - use $\bar{m}_0 = (0^n, 0^n); \bar{m}_1 = (0^n, 1^n)$
  - if the first two parts of the ciphertext are the same, guess 0
    else, guess 1

□

Remark

- Proof of Claim 3.20 used the fact that
  - encrypting the same message, always yields the same ciphertext
- Thus, any determinstic scheme must be insecure for multiple encryptions!

Theorem 3.21

Let $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be an encryption scheme
for which Enc is a deterministic function of the key and message.
Then, Π does not satisfy Definition 3.19.

Story:

- We thus must construct encryptions such that
  multiple encryptions of the same message
  yield different ciphertexts each time.

- This may seem slightly impossible,
  however, we see how to achieve this later.

Multiple encryptions using a stream cipher (a common error)

Story:

- Incorrect implementations of crypto constructons are frequent
- A common error
  use a stream cipher (in its naïve form as in Construction 3.16)
  to encrypt multiple messages.
- E.g. Microsoft did that in an implemntation of an encryption in Word/Excel
- This is not just a "theoretical artefact"—even if
  the same message is *not* encrypted twice,
  various attacks are possible.

Secure multiple encryptions using a stream cipher

Story:

- There are typcially two ways
  in which a stream cipher (pseudorandom generator)
  can be used to securely encrypt multiple plaintexts

1. Synchronised mode
   - Communicating parties use
     a different part of the stream cihper to encrypt each message
   - "Synchronised" because both parties
     need to know which parts of the stream
     have already been used to
     avoid re-use

   - This mode is useful
     where the parties communicate in a single session
     (i.e. the first party encrypts using the first part of the stream
     the second party encrypts using the second part of the stream etc)

     The important point to notice is
     since each part of the stream is used only once
     one can view the concatenation of all messages
     as a lingle long message
     Security therefore follows from Theorem 3.17
     (I am guessing the one that reduces to pseodorandm
     gen security)

   - This mode
     is not suitable in all applications
     because parties need to "maintain state" between encryptions
     thus it is not even an encryption scheme
     according to Definition 3.8

2. Unsynchroside mode:
   - In this mode
     encryptions are carried out independently of
     one another
     and the parties don't need to "maintain state".

   - To obtain security,
     our notion of pseudorandom generator must be
     significantly strengthened
   - Now, we view a pseudorandom generator as taking
     two inputs
     a seed s and
     an initial vector IV of length n.

     The requirement, (roughly) is that
     (a)
     $G(s, \mathrm{IV})$ is pseudorandom
     even when IV is known
     (but s stays secret)

     (b)
     for two randomly chosen initial vectors
     IV_1 and IV_2
     the streams $G(s, \mathrm{IV}_1)$ and $G(s, \mathrm{IV}_2)$
     are pseudorandom
     even when viewed together
     (NB: the same seed s is used)
     i.e. no poly-time distinguisher D can distinguish
     $(\mathrm{IV}_1, G(s, \mathrm{IV}_1), \mathrm{IV}_2, G(s, \mathrm{IV}_2))$ and
     $(\mathrm{IV}_1, r_1, \mathrm{IV}_2, r_2)$
     where r_1 and r_2 are independently-chosen
     uniformly-distrbuted strings
     of appropriate length

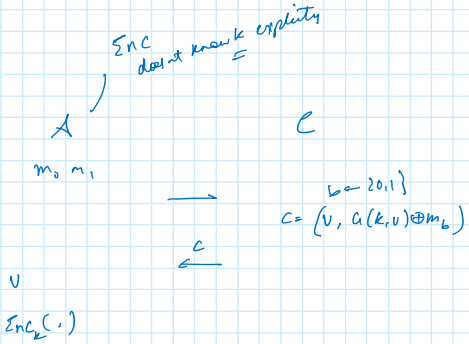
   - Given a generator as above
     one can define the encryption as
     $\mathrm{Enc}_k(m) := (\mathrm{IV}, G(k, IV) \oplus m)$

     where IV is chosen at random
     (by the encryption algorthm;
     for simplicity, restrict to fixed length messages)

     NB1: For each encryption, IV is sampled again
     thus each stream is pseudorandom
     even if the previous streams are known.

     NB2: IV is sent as a part of the plaintext
     (to allow decryption; $m := c \oplus G(s, \mathrm{IV})$)

   - Many stream ciphers in practice
     are assumed to have the "augmented pseudorandmoness" property



Warning: A standard pseudorandom generator
may not have this property—this assumption is strong
(in fact, such a generator, is almost a pseudorandom function—see Section 3.6.1)

3.5 Security under Chosen-Plaintext Attacks (CPA)

- So far, we only considered a relatively weak adversary
  who only passively eavesdrops
  (on the communication b/w the honest parties)

  (NB: our actual definiton of PrivK allows the adversary
  to choose which message to be encrypted—
  but beyond that it is passive)

- Here, we formally introduced a
  more powerful type of adversarial attack
  called "Chosen Plaintext Attack" (CPA)

- Compared to Definition 3.9 (recall, definition of computational security)
  the definition of a break remains the same
  but the adversary's capabalities ar estrengthened.

- Basic idea:
  Adversary $\mathcal{A}$ is allowed to ask for
  encryptions of multiple messages
  that it chooses "on-the-fly"
  in an adaptive manner

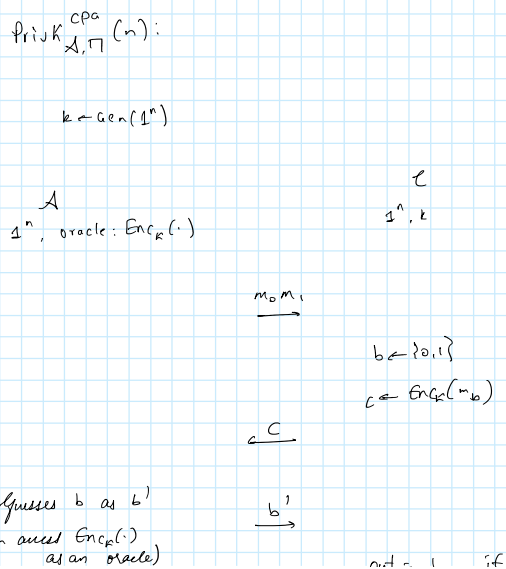
- Formalised as:
  allowing $\mathcal{A}$ to interact freely with
  an encryption oracle
  viewed as a "black-box" that
  encrypt messages of $\mathcal{A}$'s choice
  but the secret key k remains hidden

- Following standard notation in CS,
  denote by $\mathcal{A}^{\mathcal{O}(\cdot)}$
  the computation of $\mathcal{A}$ given access to an oracle $\mathcal{O}$
  and thus in this case
  we denote by $\mathcal{A}_k^{\mathrm{Enc}(\cdot)}$
  the computation of $\mathcal{A}$ with access to an encryption oracle

  NB: When Enc is randomised, each query is returned with a fresh response

- The definition of security requires that
  $\mathcal{A}$ should not be able to distinguish encryptions of two arbitrary messages
  even when $\mathcal{A}$ is given access to an encrypton oracle.

- We first give the formal definition
  and then describe what real-world
  adversarial attacks the definition is meant to model



4. $\mathcal{A}$ guesses b as b' (can access $\mathrm{Enc}_k(\cdot)$ as an oracle) → b'

out = 1 if $b' = b$

0 else

Definition 3.22

A private key encryption scheme Π has
indistinguishable encryptions under a chosen-plaintext attack
(or is **CPA-secure**)

if for all PPT $\mathcal{A}$ there is a negligible function negl s.t.

$$\Pr[\mathrm{PrivK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)$$

where the probability is taken over ... (the usual)

NB: Any scheme secure in CPA,
is secure in the presence of "eavesdropping adversary".

Story:

- It may appear that Definition 3.22 is impossible
  - In particular, consider an adversary
    that outputs (m_0, m_1)
    then receives $c \leftarrow \mathrm{Enc}_k(m_b)$
    Since the adversary has oracle access to $\mathrm{Enc}_k(\cdot)$
    it can have this oracle encrypt the messages
    to obtain $c_0 \leftarrow \mathrm{Enc}_k(m_0)$ and
    $c_1 \leftarrow \mathrm{Enc}_k(m_1)$.
    It can then compare c_0 and c_1 to the "challenge" c
    and guess accordingly (b if it matchs c_b)
- Why doesn't this strategy allow $\mathcal{A}$ to determine b wp 1?

NB: As with security under multiple encryptions
no **deterministic encryption** scheme
can be secure under CPA attacks.

[just as we saw, in the synchronised mode for stream ciphers
if the scheme "maintains state" (i.e. remembers prior messages etc)
then randomness may not be necessary
ME: but then in that case,
one has to define CPA accordingly]

Chosen-plaintext attacks in the real world

- Here's a real world example of chosen plain-text attack from World War II
- The Japanese were planning an attack on "Midway Island"
  - The US had intercepted a message saying attack on "AF"
  - But they weren't sure what "AF" corresponded to for the Japanese
  - The US sent a "plaintext" saying Midway is low on water
  - The Japanese sent (the equivalent of) "AF" is low on water

- Therefore, it always advised to use CPA security

- Not only military, e.g.
  - many servers communacite with each other
    using encryption
  - And, these messages are based on internal/external requests
  - Attackers may pretend to be users and make such requests

CPA security for multiple encryptions

- Extension of Definition 3.22 [CPA security]
  to the case of multiple encryptions
  is straightforward
  and is the same as the extension of Def 3.9 [eavesdropper security]
  to Def 3.19 [multiple encryptions, eavesdropper security]

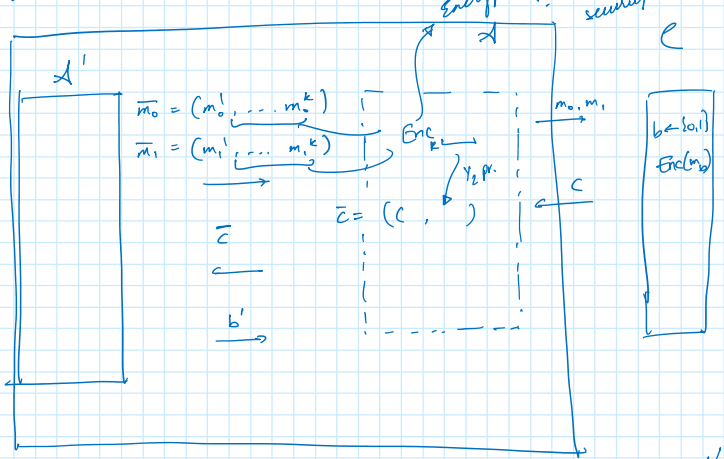
(i.e. define it to be the same as $\mathrm{PrivK}^{\mathrm{cpa}}$ except that

the adversary outputs a pair of vectors of plaintext—
require the adversary cannot distinguish wp greater than 1/2)

- However, here's a major difference:
  CPA security for a single encryption
  automatically implies CPA security for
  multpile encryptions

Trying to prove that claim

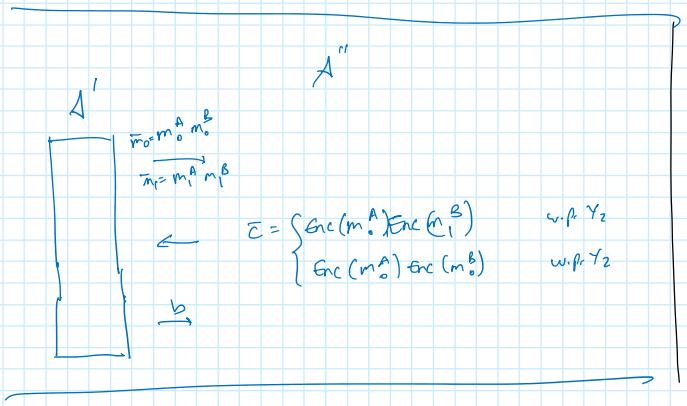
Suppose: $\mathcal{A}'$ breaks the multimessage security
We construct $\mathcal{A}$ that breaks CPA security



$\bar{c}$ well formed w.p. Y_2 → $\mathcal{A}'$ guesses better than half $\left(\frac{1}{2} + \epsilon\right)$ nonnegl.

$\bar{c}$ not well formed w.p. Y_2 → $\mathcal{A}'$ guesses...

$P_{\mathcal{A}}[\mathcal{A} \text{ guess}]$ $\frac{1}{2}\left(\frac{1}{2} + \epsilon\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\epsilon}{2}$



Then use this distinguisher

Restrict to 2-message $\mathcal{A}'$.

to show these are indistinguishable, fix a b, say 1

Suppose $\mathcal{A}'$ guesses correctly when interacting with the following kind of challenger $\mathcal{C}_1$

Perhaps this is equivalent to $\mathcal{C}_2$ s.t.

$\mathcal{A}'$

$\bar{m}_1 = (m_1^A, m_1^B)$

$\mathcal{A}'$

$\bar{m}_i = (m_i^A, m_i^B)$
$i \in \{0,1\}$

$\bar{c} = \left(Enc_k(m_b^A), \underbrace{Enc_k(m_{b\oplus 1}^B)}\right)$

$\bar{c} = \left(Enc_k(m_b^A), \underbrace{Enc_k(1^n)}\right)$

b'

$out = \begin{cases} 1 \\ 0 \end{cases}$ $\quad b' = b$ / else

if $\mathcal{A}'$'s behaviour changes, $\mathcal{A}$ can distinguish encryptions of 0^n from encryptions of m_b

& now if the advantage is $\frac{1}{2} + \epsilon$ in $Win[\mathcal{A}', \ell_2]$ this directly yields an adversary that breaks CPA

Let's construct the distinguisher using $\mathcal{A}'$

$\mathcal{D}$

$\bar{m}_0, \bar{m}_1$ $\quad b \leftarrow \{0,1\}$ $\quad m_0^A, m_1^A$

$\bar{c} = (c, Enc(1^n))$ if $b' = 0$ $\quad \xleftarrow{c}$ $\quad c := Enc(m_b^A)$
$(c, Enc(m_0^B))$

b' $\qquad$ b'

w.p. $\frac{1}{4}$ each $\bar{c} = (Enc(m_0^A), Enc(m_0^B))$
$\bar{c} = (Enc(m_1^A), Enc(m_0^B))$
$\bar{c} = (Enc(m_0^A), Enc(1^n))$
$\bar{c} = (Enc(m_1^A), Enc(1^n))$

case I
II
III
IV

So in case I & III ~ the view of $\mathcal{A}'$ can be seen as the following

$\mathcal{A}'$

$\bar{m}_0 = (m_0^A, m_0^B)$
$\bar{m}_1 = (m_1^A, m_1^B)$

$\mathcal{D}$ $\quad (m_1^B, 1^n)$ $\quad (m_0^B, 1^n)$
w.p. $\frac{1}{2}$ / w.p. $\frac{1}{2}$

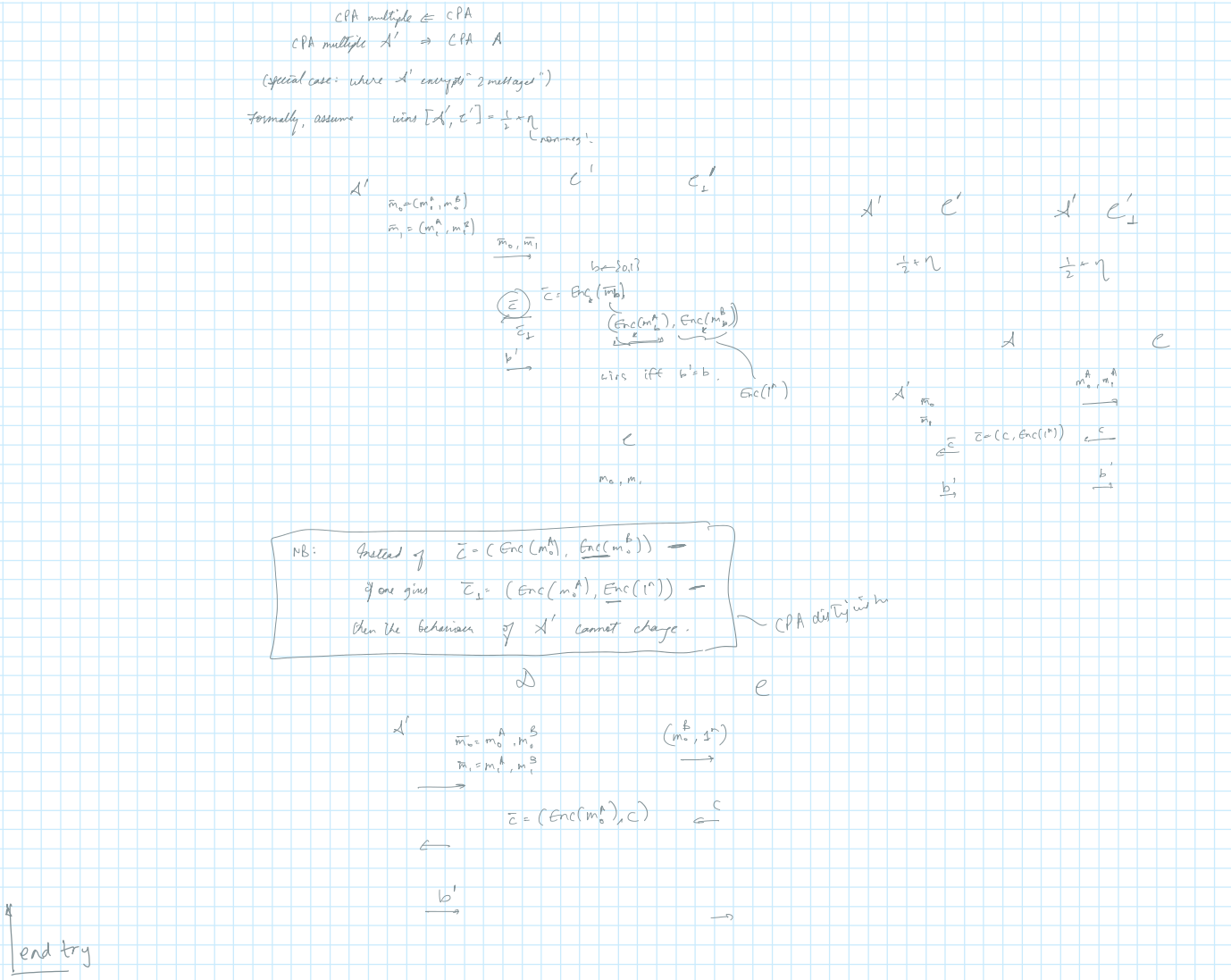
$\xleftarrow{c}$ $\quad c = Enc(m_0^B)$ if $b = 0$
$= Enc(1^n)$ else

$\bar{c} := (Enc(m_0^A), c)$
$\bar{c} \quad \bar{c} := (Enc(m_1^A), c)$

b' $\qquad$ b'

Show that $\mathcal{A}'$ receives $\bar{c} = (Enc(m_b^A), Enc(m_b^A))$ for $b \in \{0,1\}$
behaves the same as
$\bar{c} = (Enc(m_b^A), Enc(1^n))$

Then $\mathcal{A}$ can simply ignore this $Enc(1^n)$ with the $Enc(\cdot)$ oracle.
to break CPA security



Claim 3.23

Any private-key encryption scheme that has
indistinguishable encryptions
under a CPA attack
also has indistinguishable *multiple* encryptions
under a CPA attack.

Story:

- [in the public-key setting—a similar claim is stated and proved in § 10.2.2]
- Thus, it suffices to show CPA security
  and one gets multiple encryption security for free

Fixed-length vs arbitrary-length messages

Story:

- Another advantage with CPA security
  can treat fixed-length encryptions
  without much loss of generality

- Claim:
  Given any CPA-secure fixed length encryption scheme
  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$
  one can construct a variable-length CPA secure encryption scheme
  $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$
  quite easily.

- For simplicity
  suppose Π has length parameter 1
  (so that it only encrypts messages of 1 bit).

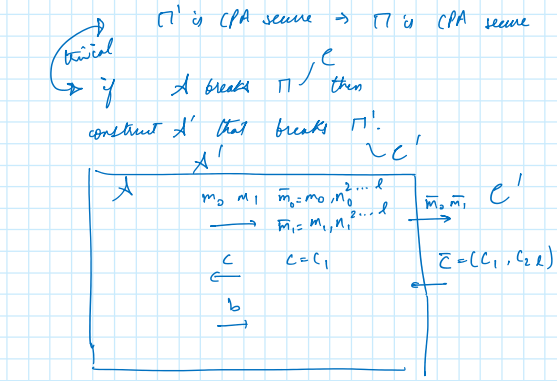
- Construction:
  Let $\text{Gen}' = \text{Gen}$
  Define:

$$\text{Enc}'_k(m) := \text{Enc}_k(m_1), \ldots \text{Enc}_k(m_\ell)$$
where $m = m_1 \ldots m_\ell$ & $m_i \in \{0,1\}^n$ $\forall i$.

  - Decryption: naturally extended

- Claim: Π' is CPA secure $\Leftrightarrow$ Π is CPA secure.

"Our Proof" Π is CPA secure $\Rightarrow$ Π' is CPA secure
using Claim 3.23



□

Story:
- There may, in practice,
  be more efficient ways to encrypt messages
  of arbitrary length
  than by adapting the scheme above
- These other ways are discussed in §3.6.4

3.6 Constructing CPA-Secure Encryption Schemes

Story
- We construct CPA secure encryption schemes
- First, we introduce pseudorandom functions

3.6.1 Pseudorandom Functions

Story:
- Recall,
  pseduorandm generators can be used
  to get security in the presence of eavesdropping adversaries

- Notion of pseudorandomness
  is also crucial for constructing security against CPA attacks

- Instead of working with pseudorandm *strings*
  we consider pseudorandom *functions*.

- Specificially, we consider
  pseudorandom functions from $\{0,1\}^n \to \{0,1\}^n$

- Recall, it does not make much sense to talk about a fixed function
  being pseudorandom
  (just as one doesn't talk about a fixed string being random)

- Thus, we must refer to the pseudorandomness of a
  distribution on functions.

- An easy way to do this,
  is to consider *keyed functions*.

- Keyed function F
  is a two-input function $F: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$
  where the first input is called the key k
  and the second input is just called the input (haha).
- In general,
  the key k will be chosen and
  then *fixed*
  and we will then be interested in the
  (single-input) function
  $F_k: \{0,1\}^* \to \{0,1\}^*$
  defined by $F_k(x) := F(k,x)$.

- For simplicity,
  we assume that F is *length preserving*
  so that the key input and output lengths of F
  are all the same
  i.e. F is only defined when $|F_k(x)| = |x| = |k|$.

- Therefore,
  by fixing a key $k \in \{0,1\}^n$
  we obtain a function $F_k(\cdot): \{0,1\}^n \to \{0,1\}^n$

- We will only be interested in function F
  that are efficient—i.e. there's a
  deterministic poly-time algo
  that computes $\mathrm{F}(\mathrm{k}, \mathrm{x})$
  given k and x as input.

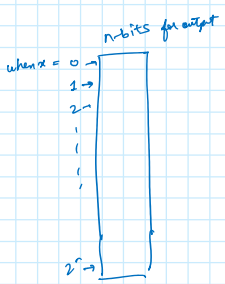
- A keyed function F
  induces a natural distribution on functions
  given by choosing a random key $k \leftarrow \{0,1\}^n$
  and then considering the resulting
  single-input function F_k.

- Intuitively,
  we call F pseudorandm if
  F_k (for randomly-chosen key k)
  is indistinguishable from
  a function chosen uniformly at random
  (from the set of all functions having the same domain and range)

  i.e. no PPT adversary can distinguish which function it is interacting with
  —F_k (for a randomly chosen k) or
  f (where f is chosen at random).

- Let's understand the space of these functions better
- From a mathematical point of view,
  we consider the set Func_n
  of all functions mapping $\{0,1\}^n \to \{0,1\}^n$
  and so randomly selecting a function
  mapping n-bit strings to n-bit strings
  corresponds exactly to choosing
  an element uniformly at random
  from this set.

- How large is the set Func_n?
  - A function f is exactly specified
    by its value on each point in its domain
  - One can view f as a large look-up table.
    The look-up table can be specified using $T = n \cdot 2^n$ bits
    Thus, there can be 2^T possible different tables, i.e.
    Func_n has size $2^T = 2^{n \cdot 2^n}$



bits to specify a table = $n \cdot 2^n = T$ bits

possible assignments to the values in the table: $2^T = 2^{n \cdot 2^n}$

- Viewing a function as a look-up table

provides another way of thinking about
selecting a function $f_n \in \text{Func}_n$
uniformly at random.

This is how: picking f_n uniformly at random is equivalent to
picking each row of the look-up table of f_n
uniformly at random.

I.e. the values of $f_n(x)$ and $f_n(y)$ (for $x \neq y$)
are completely independent and uniformly distributed.

- Returning to pseudorandm functions
  recall: we want to construct a
  keyed function F s.t.
  F_k (for $k \leftarrow \{0,1\}^n$)
  is indistinguishable from
  f_n (for $f_n \leftarrow \text{Func}_n$).

  - NB: the former is chosen from a distribution of 2^n distinct functions
    while the latter is chosen from all $2^{n \cdot 2^n}$ functions

- Despite this,
  the "behaviour" of these functions
  must look the same to any PPT distinguisher.

- A first attempt at formalising the notion of a pseudorandm function
  would be as we did in Definition 3.15
  — require that every PPT distinguisher D
  that recieves a description of
  the pseudorandom function F_k outputs 1 with
  almost the same probability as when
  it recieves a description of a random function f_n
- Issue:
  the description of a random function has "exponential length"
  (i.e. given by its look-up table, that has length $n \cdot 2^n$)
  while D is limited to poly-time so it cannot even
  view the look-up table.

- The actual definition, therefore
  gives D (the distinguisher) **oracle access**
  to the function in question (either f_n or F_k)

  - D is allowed to query the oracle
    at any point x in response to which
    the oracle returns the value of the function
    evaluated at x

  - We treat this oracle as a black-box
    (in the same way, as we did for the encryption oracle in the CPA definition)

  - NB: Here the oracle is deterministic
    (i.e. once k is fixed)

Here's the definition

Definition 3.24

Let $F: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving, keyed function.
We say F is a **pseudorandom function** if
for all PPT distinguishers D, there is a neglgible function negl s.t.

$$\left| \Pr\left[D^{F_k(\cdot)}(1^n) = 1\right] - \Pr\left[D^{f_n(\cdot)}(1^n) = 1\right] \right| \leq \text{negl}(n)$$

where $k \leftarrow \{0,1\}^n$ is chosen uniformly at random
and f_n is chosen uniformly at random
from Func_n (the set of functions mapping n-bit strings to n-bit strings).

Remarks:

- Notice that D interacts freely with its oracle
  - It can therefore ask queries adaptively
    i.e. it can choose its next input based on the previous outputs etc.
  - However,

since D runs in poly time
it can only ask a
poly number of queries

- A pseudorandom function must inherit
  any efficiently checkable property of a random function
  - E.g. even if x, x' differ in only a single bit
    the outputs $F_k(x)$ and $F_k(x')$ must
    (with overwhelming probability over k)
    look completely uncorrelated.
  - This tells us why
    pseudorandom functions are useful for constructing
    secure encryption schemes
- Note:
  the distinguisher D is not given the key k
  - It is meaningless to require that F_k be pseudorandm
    if k is known
    Because f_n is easily distiguished from F_k in that case
    E.g. query the oracle at 0^n, and compute $F_k(0^n)$ and see if they match.
    An oracle for F_k will always match
    while a random oracle will match with prob 2^{-n}
  - In practice,
    e.g. it could be that given k,
    it is easy to find the pre-image of 0^n
    but for random functions, this is provably hard (given only oracle access)

On the existence of pseudorandom functions

Story:

- It is natural to ask: do such entities exist?
- For now, we note that
  there exist efficient primitives called
  "block ciphers" that are
  believed to act as *pseudorandom functions*.
- From a theoretical point of view
  pseuodarondm functions exist
  iff
  pseudorandm generators exist
  - Thus, PRFs can be generated from any
    hard problem that allow for the construction of
    PRGs
- These are discussed later (in Ch 5 and 6)

Emotion:

- Existence of PRFs is very surprising
  - the fact that these can be constructed
    based on hard problems (of certain types)
    is one of the truly amazing contributions of modern cryptography.

Using pseudorandom functions in cryptography

Story:

- PRFs turn out to be useful for various crypto constructions
- Below, we use them to obtain CPA-secure encryption
  - and later in Ch 4 to obtain MACs (message authentication codes)
- Why are they so useful?
  - allow for a clean and elegant analysis of the constructions
    that use them
  - i.e. given a scheme that is based on PRFs
    a general way of analysing the scheme
    is to first prove under the assumption that
    a truly random function is used instead
    This step relies on probabilistic arguments (nothing to do with computational)
    Next, the security is derived
    by proving that if an adversary can break the scheme
    when a PRF is used then

it must implicitly be distinguishing the function from random

§ 3.6.2 CPA-Secure Encryption Schemes from Pseudorandm Functions

We focus here on
constructing a fixed-length encryption scheme.

Recall:
CPA secure for fixed-length also yields CPA secure for arbitrary length (as we saw in Section 3.5)
In §3.6.4, we consider more efficient constructions.

A naïve attempt (for cosntructing encryption from PRFs): $\text{Enc}_k(m) = F_k(m)$

- This cannot be CPA secure because
  it is a determistic encryption scheme
  (Me: go back and revise if this is not clear)

Our actual construction is probabilistic:
$c := (r, F_k(r) \oplus m); r \leftarrow \{0,1\}^n$

Remark:
This can be viewed as an instance of XORing
a pseudorandm "pad" with a plaintext
major differenc being
that we are choosing an "indpendent" pseudorandom string
each time

Construction 3.25

Let F be a pseudorandom function. For length n, define it as follows:

- Gen: on 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output as key
- Enc: on $k, m \in \{0,1\}^n$, choose $r \leftarrow \{0,1\}^n$ informly at random and ...

CONSTRUCTION 3.25

Let F be a pseudorandom function. Define a private-key encryption scheme for messages of length n as follows:

- **Gen:** on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.
- **Enc:** on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose $r \leftarrow \{0,1\}^n$ uniformly at random and output the ciphertext

$$c := \langle r, F_k(r) \oplus m \rangle.$$

- **Dec:** on input a key $k \in \{0,1\}^n$ and a ciphertext $c = \langle r, s \rangle$, output the plaintext message

$$m := F_k(r) \oplus s.$$

Story:

- Intuitively, security holds because
  $F_k(r)$ looks completely random
  to an adversary who observes a ciphertext (r, s)
  (and thus the encryption scheme is similar to the one-time pad
  granted r was not used in a previous encryption)
  (specificially, it wasn't used by the encryption oracle at any of the adversary's query)
  Moreover, this "bad event" (i.e. repeating value of r)
  occurs with only neglgibile probability.

Theorem 3.26

If F is a pseuodarondm function, then
Construction 3.25 is CPA secure, i.e.
fixed-length private key encrytpion scheme with
length parametere $\ell(n) = n$
that has indistinguishable encrytpions under a
chosen plaintext attack

Proof

Story:
- Follows a generalise paradigm
  - Analyse using an idealised world where f_k (truly random function) is used
    instead of F_k

- Then we claim if the scheme were insecure when F_k was used
  it would mean F_k can be distinguished from a truly random function

- Let $\widetilde{\Pi}$ be an encryption scheme that is
  exactly the same as Π
  except that f_n instead of F_k

  (i.e. $\widetilde{\mathrm{Gen}}(1^n)$ chooses a random function $f_n \leftarrow \mathrm{Func}_n$

  & $\widetilde{\mathrm{Enc}}$ encrypts just like Enc
  except using f_n (instead of F_k).

(NB: this is not a "legal encryption"
∴ it's not efficient — still well defined
& enough for our purpose)

- Claim: $\forall$ $\mathcal{A}$ making at most $q(n)$ queries
  (to the Enc oracle),

  one has that

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1\right] \le \frac{1}{2} + \frac{q(n)}{2^n}. \qquad (3.4)$$

(NB: We don't assume anything about the
computational power of $\mathcal{A}$)

Proof

- Every time m is encrypted
  (either by the Enc oracle of
  the challenger)
  a random $r \in \{0,1\}^n$ is chosen &
  the ciphertext equals
  $(r, f_n(r) \oplus m)$.

- Let r_c denote a random string
  used by the challenger,
  i.e. $c = (r_c, \underbrace{f_n(r_c) \oplus m_b}_{:= s_c})$

- there are two subcases.

(a) the values r_c is used by the encryption oracle
(at least once, over the many times $\mathcal{A}$ invokes
the encryption oracle).

In this case, $\mathcal{A}$ can trivially tell
which message c corresponds to.

(Suppose $\mathcal{A}$ had queried $\mathrm{Enc}(m)$
& it got $(r_c, f_n(r_c) \oplus m)$;
so it can learn
$f_n(r_c)$.

Now it easily computes $m_b = c_c \oplus f_n(r_c)$)

NB: Prob. of learning the value of $f_n(r_c)$
with $q(n)$ queries is at most
$q(n)/2^n$.

(b) the value r_c is never used by the $\mathrm{Enc}(\cdot)$ oracle

$q(n)/2^n$.

(b) The value r_c is never used by the $\mathrm{Enc}(\cdot)$ oracle for answering $\mathcal{A}$'s queries.
(In this case $\mathcal{A}$ learns nothing about the value of $f_n(r_c)$ from the interaction.
$\therefore$ $\mathcal{A}$'s view is that of a OTP & so $b' = b$ w.p. $\frac{1}{2}$)

Denote by rep the event that (a) happens.

$\Pr[\mathrm{PrivK}^{cpa}_{\mathcal{A},\tilde{\Pi}}(n)=1] = \Pr[\mathrm{PrivK}^{cpa}_{\mathcal{A},\tilde{\Pi}}(n)=1 \mid \mathrm{rep}]\Pr[\mathrm{rep}]$ ($\Pr[\mathrm{rep}] \le q(n)/2^n$, $\Pr[\mathrm{PrivK}^{cpa}_{\mathcal{A},\tilde{\Pi}}(n)=1 \mid \mathrm{rep}] \le 1$)
$+ \Pr[\mathrm{PrivK}^{cpa}_{\mathcal{A},\tilde{\Pi}}(n)=1 \mid \neg\mathrm{rep}]\Pr[\neg\mathrm{rep}]$ (γ_2, $1 - q(n)/2^n$)
$\le \frac{1}{2} + q(n)/2^n$

Part II of the proof:

- Now fix some PPT adversary $\mathcal{A}$ & define the f^n ϵ by

$$\epsilon(n) := \Pr[\mathrm{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)=1] - \frac{1}{2}. \quad (3.5)$$

NB: # calls $\mathcal{A}$ makes is upper bounded by its running time & thus by some poly $q(\cdot)$.

NB2: (3.4) applies to $\mathcal{A}$.

Summary: At this point we have

$\Pr[\mathrm{PrivK}^{cpa}_{\mathcal{A},\tilde{\Pi}}(n)=1] \le \frac{1}{2} + \frac{q(n)}{2^n}$ ~ we already showed.

$\Pr[\mathrm{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)=1] = \frac{1}{2} + \epsilon(n)$ ~ (by def^n)

NB: If ϵ is not negl, then their diff. is not negl either.

Intuition: Such a gap should allow us to distinguish random from pseudorandom. Here the red^n.

Informal description of the Distinguisher D.

- We use $\mathcal{A}$ to construct a distinguisher D for the pseudorandm function F.
- D is given access to some function and it has to say whether the function is pseudorandm (i.e. equal of F_k for some $k \in \{0,1\}^n$) or random (equal to f_n for randomly chosen $f_n \leftarrow \mathrm{Func}_n$).
- To do this, D emulates the CPA indistinguishableitiy experiment for $\mathcal{A}$
  If $\mathcal{A}$ succeeds, then D guesses the oracle is pseudorandom
  otherwise it guesses the oracle is random

Construction: Distinguisher D

Given 1^n as input, and access to an oracle $\mathcal{O}$

1. Run $\mathcal{A}(1^n)$.
   Whenever $\mathcal{A}$ queries its encryption oracle on a message m,
   do the following:

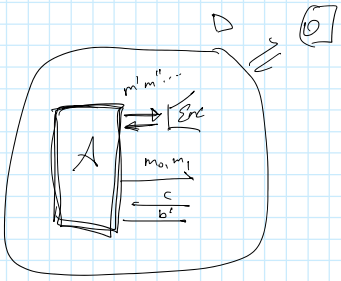
   return $Enc(m)$ where

   $Enc(m) :=$

   (a) Choose $r \leftarrow \{0,1\}^n$ uniformly at random
   (b) Query to obtain $s = \mathcal{O}(r)$
   (c) Return $(r, s' \oplus m)$ to $\mathcal{A}$

2. When $\mathcal{A}$ outputs messages $m_0, m_1 \in \{0,1\}^n$,
   choose a random bit $b \leftarrow \{0,1\}$ and then

   return $Enc(m_b)$.



3. Continue answering any encryption oracle queries
   of $\mathcal{A}$ as before
   Eventually, $\mathcal{A}$ outputs a bit b'.

   Output 1 if $b' = b$; otherwise
   output 0

Observataions:

1. If D's oracle is a pseudorandm function
   then the view of $\mathcal{A}$ (when run as a subroutine of D) is
   distributed identically to the view of $\mathcal{A}$
   in experiment $\text{PrivK}^{cpa}_{\mathcal{A},\Pi}$

   This is because
   a key k is chosen at random and then every encryption
   is carried out by choosing a random r
   computing $s' = F_k(r)$
   and setting the ciphertext equal to
   $(r, s' \oplus m)$.

   Thus,

   $$Pr[D^{F_k(\cdot)}(1^n) = 1] = Pr[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1]$$

   where $k \leftarrow \{0,1\}^n$ is chosen uniformly at random.

2. If D's oracle is a random function
   then the view of $\mathcal{A}$ (when run as a subroutine of D) is
   distributed identically to the view of $\mathcal{A}$
   in experiment (privK with $\widetilde{\Pi}$).

   This can be seen exactly as above,
   the only differenc is that f_n (a random function) is used
   instead of F_k.

   Thus,

   $$Pr[D^{f_k(\cdot)}(1^n) = 1] = Pr[PrivK^{cpa}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$$

   where $f_n \leftarrow Func_n$ is chosen uniformly at random.

Since F is a pseudorandom function and
D runs in probabilistic poly time
there exists a negligible function negl
s.t.

$$\left| Pr[D^{F_k(\cdot)}(1^n) = 1] - Pr[D^{f_n(\cdot)}(1^n) = 1] \right| \leq negl(n)$$

Combining with the above observation & Eq (3.4) (3.5)

combining with the above observation &
Eq (3.4) (3.5)
(bound with $\tilde{\Pi}$) def of $\epsilon(n)$.

$$\begin{aligned}
\text{negl}(n) &\geq \left| \Pr[D^{f_k(\cdot)}(1^n)=1] - \Pr[D^{f_n(\cdot)}(1^n)=1] \right| \\
&= \left| \Pr[\text{PrivK}^{cpa}_{A,\Pi}(n)=1] - \Pr[\text{PrivK}^{cpa}_{A,\tilde{\Pi}}(n)=1] \right| \\
&\geq \Pr[\text{PrivK}^{cpa}_{A,\Pi}(n)=1] - \Pr[\text{PrivK}^{cpa}_{A,\tilde{\Pi}}(n)=1] \\
&\geq \frac{1}{2} + \epsilon(n) - \frac{1}{2} - \frac{q(n)}{2^n} \\
&= \epsilon(n) - \frac{q(n)}{2^n}
\end{aligned}$$

& from this, it's clear that $\epsilon(n)$ ∠ negligible
∵ q is poly.

□

Story

- Recall: CPA secure fixed length implies CPA secure arbitrary length
- Applying the discussion in Section 3.5, we obtain teh following
  scheme for variable length encryption

$$\langle r_1, F_k(r_1) \oplus m_1, r_2, F_k(r_2) \oplus m_2, \ldots, r_\ell, F_k(r_\ell) \oplus m_\ell \rangle.$$

- The scheme can handle messages whose length is not
  an exact multiple of n

Corollary 3.27

If F is a pseudorandm function, the scheme above
is a private-key encrytpion scheme
for arbitrary-length messages that
has indistinguishable encryptions under
a chosen-plaintext attack.

Efficiency of Construction 3.25

- CPA secure encryption above
  - and its extension
    has the drawback that
    length of ciphertext is
    at least double that of the plaintext
- In § 3.6.4 we will show how
  longer plaintexts can be encedode more efficiently.

3.6.3 Pseudorandom Permutations and Block Ciphers

Inf definition:

Let $F: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be
an efficient, length-preserving, keyed function.

We call F a **keyed permutation** if for every k,
the function $F_k(\cdot)$ is one-to-one
(and therefore, since F is length-preserving, a bijection).

We call F a **keyed permutation** if for every k,
the function $F_k(\cdot)$ is one-to-one
(and therefore, since F is length-preserving, a bijection).

We say a keyed permutation is efficient
if there is a poly-time algorithm computing
$F_k(x)$ given k and x
as well as
poly-time algorithm computing $F_k^{-1}(x)$ given k and x.

Story:

- We define what it means for a keyed permutation F
to be pseudorandom in a manner analogous to Def 3.24
but with two differences.

$U|x\rangle|y\rangle$

$U_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle$

$U_f|x\rangle|y\rangle|0\rangle = |x\rangle|y\rangle|f(x)\rangle$

$U_f := U_{g_\ell} \cdots U_{g_2} U_{g_1}$

U_f^{-1} $U_{g_1}^{-1} \cdots U_{g_\ell}^{-1}$

$U_{f^{-1}}|0\rangle|0\rangle|z\rangle = |f_x^{-1}(z)\rangle|f_y^{-1}(z)\rangle|z\rangle$

$U_f^{-1} \neq U_{f^{-1}}$

f

$x \mapsto y$

$|x\rangle|y\rangle$

I

$|x\rangle \mapsto |y\rangle$ $f(x)$

$O|x\rangle = |y\rangle$

$g_1 \cdots g_\ell$

$O^{-1} = g_\ell^{-1} \cdots g_1^{-1}$

II

$f(x)$

$O|x\rangle|0\rangle = |x\rangle|y\rangle$

First,
we require that F_k (for a randomly chosen k)
be indistinguishable from a random permutation
rather than a random function.

Second,
more significant and is motavited by the fact that
cryptographic schemes
using a keyed permutation may utilise
the inverse F_k^{-1} in addition to F_k

Thus, we require F_k to be indistinguishable from a random permutation
even if th edistinguisher is given oracle access to the
inverse of the permutation.

[in other works, what is termed a pseudorandom permutation,
is referred to as a strong or super pseudorandom (because we are adding conditions about its inverse)]

Definition 3.28

Let $F: \{0,1\}^* \to \{0,1\}^* \to \{0,1\}^*$ be an efficient, keyed permutation.

We say F is a **pseudorandom permutation** if

$\forall$ PPT distinguishers D,

$\exists$ negl f^n negl s.t.

$$\left| Pr\left[D^{F_k(\cdot), F_k^{-1}(\cdot)}(1^n) = 1\right] - Pr\left[D^{f_n(\cdot), f_n^{-1}(\cdot)}(1^n) = 1\right] \right| \leq negl(n)$$

where $k \leftarrow \{0,1\}^n$ is chosen uniformly at random &

f_n is chosen uniformly at random from the set of all permutations

Story:

- A pseudorandom permutation can be used
  in place of a pesudorandom function in
  any cryptographic construction.

- This is due to the fact that to any PPT observer
  a pseudorandom permutation cannot be distuished from
  a pseudorandom function.

- Intuitively,
  this is due to the fact
  that a random function f_n looks
  identical to a random permutation
  unless a distinct pair of values
  x and y are found for
  each $f_n(x) = f_n(y)$
  (since in such a case,
  the function cannot be a permutation)

  The probability of finding such points x, y
  using a poly number of queries, is negligible.
  [proof is an exercise]

Proposition 3.29
If F is a **pseudorandom permutation** then
it is also a **pesudorandom function**.

Story

- We noted earlier
  that a **stream cipher** can be modelled as a **pseudorandm generator**

- The analogue for the case of **pseudorandm permutations** is
  a **block cipher**.

- We stress that,
  as with stream ciphers,
  block ciphers themselves are not encryption schemes
  Rather, they are buliding blocks
  that can be used to construct secure encryption schemes.

- E.g. Construction 3.25 yields a CPA secure private key encryption scheme
  In contrast, an encryption scheme that
  works by just computing
  $c = F_k(m)$
  where $\mathrm{F_k}$ is a pseudorandom permutation (block cipher)
  yields a scheme that is not CPA

- Story: constructions of block ciphers in Ch 5
  Remark:
  conventions that lengths of key, input and output are not alwasy the same
  the keys can be smaller
  input/output must have the same length, of course—permutation
  typically called the "block size".

3.6.4 Modes of Operation

Informal defn:
Mode of operation
is essentially a way of encrypting arbitrary-length messages
using a block cipher (i.e. pseudorandom permutation).

Story:

- In Corollary 3.27
  we have already seen one example of a mode of encryption
  albeit one that is not
  very efficient in terms of
  the length of the ciphertext.

- In this section
  we will see a number of modes
  of encryption having
  improved ciphertext expansion
  (defined to be the difference b/w length of
  ciphertext and message)

- Note
  arbitrary-length messages
  can be unambiguously padded
  to a total length that is
  a multiple of any desired block size
  by appending a 1 following by sufficiently many 0

  (and adding a block in case the length of the message is already a
  multpile of the block size).

Assumption:
For most consturctions
we therefore just assume
that the length of the plaintext
is exactly a multiple of the block size.

Story;

- Through out this section,
  we refer to a pseudorandom permutation/block cipher F
  with block length n (i.e. input output size)
  and we consider
  encryptions of messages consisitng of ℓ blocks
  each of length n.

- We present four modes of operation and discuss their security.

Mode 1 — Electronic Code Book (ECB) mode [INSECURE! DON'T USE]

Story: This is the most naïve mode of operation
Encryption:

- Given $m = m_1 \dots m_\ell$
  $c = \langle F_k(m_1) \dots F_k(m_\ell) \rangle$
  where F is a pseudorandom permutation
  To decrypt, use F_k^{-1} is used
  (and that is also efficient to compute by
  definition of pseudorandom permutations)

NB:

- Encryption is deterministic
  Therefore, this scheme cannot be CPA-secure
- Even worse, it does not have IND (indistinguishable encryptions)
  in the presence of eavesdropper
  even if it is only used once.
- E.g. have $m_1 = m_2$ one can already distinguish it from a
  message where $m_1 \neq m_2$.

Mode 2 — Cipher Block Chaining (CBC) mode.

Encryption:

- $v \leftarrow \{0,1\}^n$
- $c = \big(v, \underbrace{F_k(v \oplus m_1)}_{c_1}, \underbrace{F_k(c_1 \oplus m_2)}_{c_2}, \dots \underbrace{F_k(c_{\ell-1} \oplus m_\ell)}_{c_\ell}\big)$

NB: v is sent with the encryption—otherwise decryption
by the honest party is also unclear

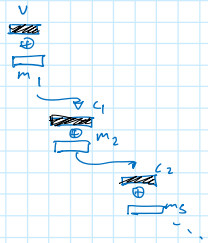
NB2: Encryption in CBC is probabilistic

Claim: If F is a pseudorandom permutation,
then CBC-mode encryption is CPA secure

Me: In case you're an idiot like me, the point of this scheme is that
the length of the ciphertext here is only $\ell(n+1)$
while the scheme based on
pseudorandom functions needed
$2\ell n$ lengthed ciphertexts.

Remark:

- Main drawback of this mode:
  Encryption must be carried out in a sequence.



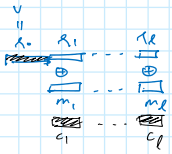
Moded 3—Output Feedback (OFB) mode.

Story:

- Using a block-cipher
  - to generate a pseudorandm stream
    - and this is then XORed with the message

Encryption:

- $v \leftarrow \{0,1\}^n$
- generate a stream v as follows
  - $r_0 := v$
  - $r_i := F_k(r_{i-1})$
- $c_i := m_i \oplus r_i$



NB: We don't need F to be invertible to decrypt

Claim: This scheme is CPA secure if F is a pseudorandom function

NB: Encryption and decryption must still be computed sequentially
  However, the sequential computation can be done
    is independently of the encrypted message.

  I.e. one can do pre-processing and then
    the encryption can be done in parallel (therefore "really fast").

Mode 4—Counter (CTR) mode.

Story:

- This mode of operation is less common than CBC
  but has many advantages
- There are different variants of counter modes
  We focus on the "randomised counter mode".

- As with OFB
  this mode can alsobe viewed as a
    way to "genaret pseudorandm stream
      from a black cipher"

Encryption:

- $\text{ctr} \leftarrow \{0,1\}^n$
- $r_i := F_k(\text{ctr} + i)$ (addition is modulo 2^n)
- $c_i := r_i \oplus m_i$

NB:

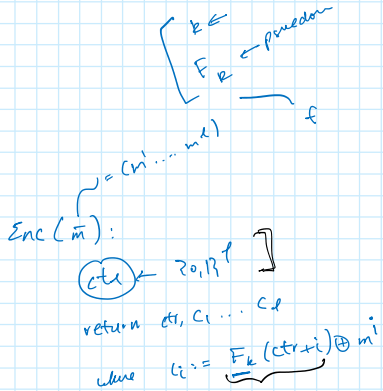
- The encryption scheme (above) is CPA secure
  (we prove this below)
- Both encryption and decryption
  can be fully parallesied
- Also admits pre/post processing independent of the message
- Encryption can be **randomly accessed**—
  Can decrypt the ith block
    without decrypting anything else.

Theorem 3.30
If F is a pseudorandm function
  then randomised counter mode (as defined above)
    has IND (indistinguishable) encryptions under CPA attacks.

Proof.

Strategy:

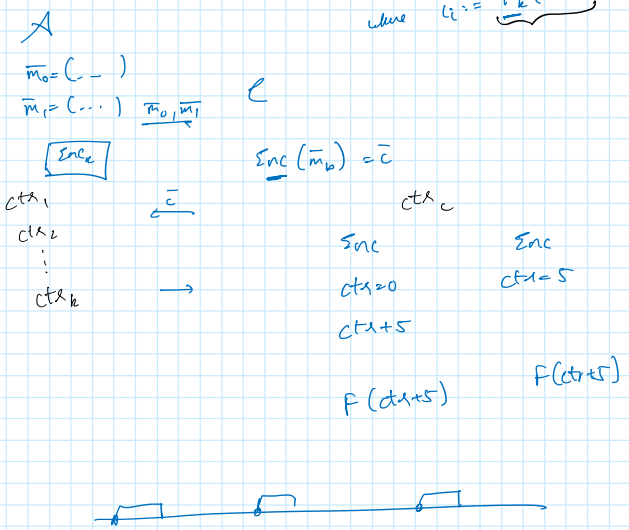
- As in the proof of Thorem 3.26
  - we prove the present theorem
    by first showing that the randomised counter mode
    is CPA secure
    when a truly random function is used
    we then prove that replacing the random function
    by a pseudorandom function
    cannot make the scheme insecure.



Notation:

Let ctr_c denote
the initial value ctr used when
the challenge ciphertext is encrypted.

(basically the counter used by the challenger)



Intuition:

When a random function f_n is used
in randomised counter mode
security is achieved
s along as each block c_i
of the challenge ciphertext
is encrypted using $\text{ctr}_c + i$
that was never used by the encryption oracle
for answering any previous queries

Why is it secure?
Because if $\text{ctr}_c + i$ was never used to answer
a previous encryption query, then
the value of $f_n(\text{ctr}_c + i)$
is completely random
and so XORing it with a block of the plaintext
has the same effect as encrypting
with a one-time pad.

Proving that a randomised counter mode
is CPA secure boils down to bounding
the probability that $\text{ctr}_c + i$ was previously used.

Notation (cont.):

Let $\Pi = (Gen, Enc, Dec)$ denote the randomised counter mode encryption scheme.

$\widetilde{\Pi} = (\widetilde{Gen}, \widetilde{Enc}, \widetilde{Dec})$ denote the same as above, except a truly random f_n (instead of pseudorandom F_k) is used.

[i.e. $\widetilde{Gen}(1^n)$ chooses $f_n \leftarrow Func_n$ at random
$\widetilde{Enc}$ encrypts as Enc except f_n is used instead of F_k.
$\widetilde{Dec}$ similarly

NB: Neither $\widetilde{Gen}$ nor $\widetilde{Enc}$ nor $\widetilde{Dec}$ are efficient
This doesn't matter for the purposes of defining an experiment with $\widetilde{\Pi}$.

Goal: $\exists$ negl negligible f^n s.t.
$P_r[PrivK^{cpa}_{A,\widetilde{\Pi}}(n) = 1] \leq \frac{1}{2} + \text{negl}$ [3.6]

NB: Don't require A to be computationally bounded
but only require

- it makes poly many queries
  (to the encryption oracle)
  (each query is on a poly sized message)
- outputs
  m_0, m_1 of poly length.

Notation:

Let: q be a poly upper-bound on the # queries made by $\mathcal{A}$ to its oracle (as well as max length of m_0, m_1)

Fix: n (security parameter).

Recall: ctr_c denote the counter used by the challenger (corresponding to the ciphertext).

Let: ctr_i denote the counter for the i^{th} query to the encryption oracle.

NB: When the ciphertext is encrypted, the function f_n is applied to the value

$$ctr_c, ctr_c+1, \ldots ctr_c + l_c$$

where $l_c \leq q(n)$ is the length (in "blocks") of $\overline{m}_0$ & $\overline{m}_1$

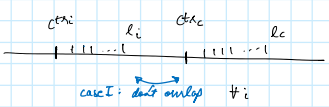
NB2: When the i^{th} (encrypting) oracle is queried, f_n is evaluated at

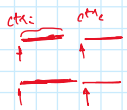
$$ctr_c + 1, \ldots, ctr_i + l_i$$

where $l_i \leq q(n)$ is the length (in blocks) of the message whose encryption was requested

Case I: There do not exist any $i, j, j' \geq 1$ (with $j \leq l_i$ & $j' \leq l_c$) for which

$$ctr_i + j = ctr_c + j'.$$





case I: don't overlap $\forall i$

NB: This case is exactly like the OTP so the $\mathcal{A}$ outputs $b' = b$ is exactly $1/2$.

case II: There exist $i, j, j' \geq 1$ (with $j \leq l_i$, $j' \leq l_c$) for which

$$ctr_i + j = ctr_c + j'$$

(i.e. the value of f used for encrypting the i^{th} block of the message by C j^{th} block of the i^{th} message by $\mathcal{A}$ match)

Thus, the adversary knows $f_n(ctr_i + j) = f_n(ctr_c + j)$

$\mathcal{A}$ can therefore decrypt the j^{th} block of $\overline{c}$ & distinguish whether $\overline{m}_0$ was encrypted or $\overline{m}_1$ was encrypted. (sometimes we drop the bar)

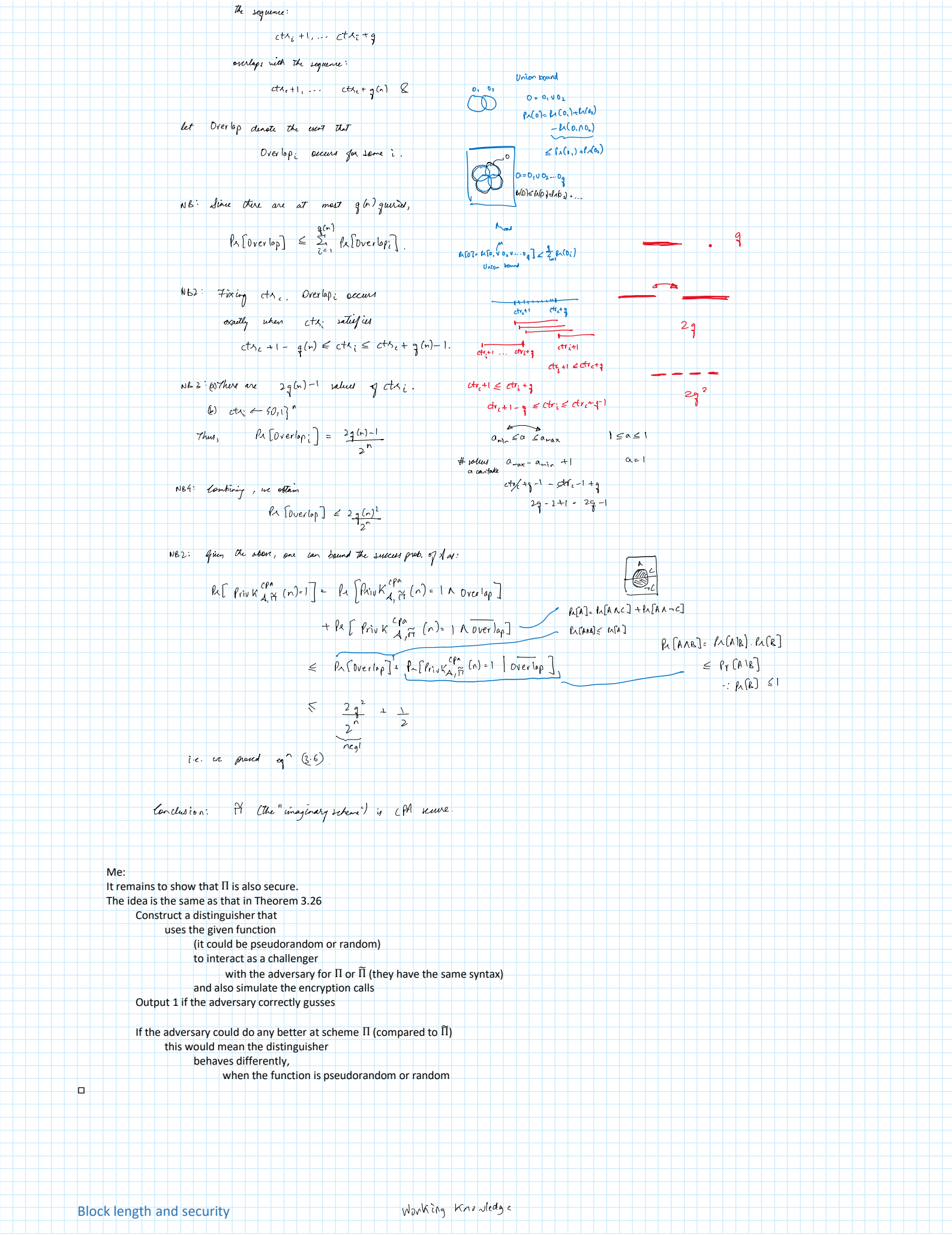
$\therefore$ $\mathcal{A}$ guesses b' correctly is almost 1.

Goal: Evaluate the prob. that case II happens.

NB: The probability is maximised when l_c & l_i are each as large as possible.

$\therefore$ we assume that $l_c = l_i = q(n)$ $\forall i$.

Let: $Overlap_i$ denote the event that



Me:
It remains to show that Π is also secure.
The idea is the same as that in Theorem 3.26
Construct a distinguisher that
uses the given function
(it could be pseudorandom or random)
to interact as a challenger
with the adversary for Π or $\widetilde{\Pi}$ (they have the same syntax)
and also simulate the encryption calls
Output 1 if the adversary correctly gusses

If the adversary could do any better at scheme Π (compared to $\widetilde{\Pi}$)
this would mean the distinguisher
behaves differently,
when the function is pseudorandom or random

□

Working knowledge

Block length and security

Working knowledge

stream — generator
block — function / permutation
ciphers

Story

- All of the above modes
  (with the exception of ECB that is anyway insecure)
  use a random initial vector (IV)
- The IV has the effect of randomising the encryption process
  and ensures (whp)
  the block cipher is always evaluated
  on a new input
  that was never used before.

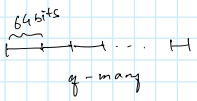
- This is important because
  in the proofs of Thm 3.26
  (simplest scheme that was CPA secure,
  using pseudorandm functions) and
  Thm 3.30
  (CTR is CPA secure)
  if the input to the block cipher is used more than once
  then security can be violated

  (e.g. in the CTR mode, the same pseudorandm key will get
  XORed wth two different plaintext blocks).

  - This shows that
    the security in these cases
    depends on both the key length and the block length
    (e.g. if we use a block-cipher with 64 bit block length,
    we showed in the proof above
    in CTR mode,
    even if a completely random function
    with this block length is used
    the success prob for
    an adversary is at about $\frac{1}{2} + \frac{q^2}{2^{64}}$
    (when it makes q queries to the encryption oracle
    (of q length blocks))

    While this is asymptotically negligible
    (i.e. when the block length grows with n)

    However, for this specific setting,
    there's no security when the number of blocks is $q \approx 2^{30}$



$\frac{1}{2} + \frac{q^2}{2^{64}} \approx 1$ when $q = 2^{30}$.

Other modes of operation

Story:

- In recent years
  many different modes of operations have been introduced
  offering certain advantages for certain settings

- In general, CBC, OFB, and CTR modes
  suffice for most applications where
  CPA security is needed

NB: none of these modes are secure against CCA (chosen ciphertext) attack.

Modes of encryption and message tampering.

Story:

- In many texts on cryptography
  modes of operation are also compared based on
  how well they protect against adversarial modifications of
  ciphertexts
- We do *not* include such a comparsion here
  Because, issue of message integritiy/authentication
  should be dealt with separately from
  encryption
  This is done in the next chapter.

- In fact,

none of the modes above, achieve
full message integrity
in the sense we define there.

Stream ciphers vs Block ciphers.

Remarks:

- It is possible to work in
  "stream cipher mode" using a "block-cipher"
  (i.e. generating a stream of pseudorandom bits and XORing it with the plaintext)

- A block cipher can be used
  to generate multiple (independent)
  pseudorandom streams
  while a stream cipher is limited to
  generating a single such stream.

Question: Which is preferable? A block cipher or a stream cipher?

Answer:

- The only advantage of stream ciphers is relative efficiency
  (e.g. on resource-constrained devices)
  (RC4 (stream cipher) only twice as fast as AES (block cipher))

- Stream ciphers are much less well understood, compared to block ciphers
  Block ciphers are better understood—many excellent block ciphers are known
  efficient and believed to be secure
  (details in Chapter 5)

- Stream ciphers get broken more often,
  thus our confidence in their security is lower

- Furthermore, it is more likely that
  stream ciphers will be misused in such a way that
  the same pesudorandom stream will be used twice

Bottom line: Use of block ciphers is recommended (unless its not possible for some reason).

§ 3.7 Security Against Chosen-Ciphertext Attacks (CCA)

Story:

- So far we have considered two types of adversaries
  (a) passive adversary that only eavesdrops
  (b) active adversary that carries out a chosen-plaintext attack
- We now consider a third type of attack
  called a chosen-ciphertext attack
  is even more powerful than these two

Intuitive definition: Chosen Ciphertext attack:
We provide the adversary with the following abilities:
(a) encrypt any messages of its choice
as in a chosen-plaintext attack &
(b) decrypt any ciphertext of its choice
(except the ones it must distungiuish—details later)

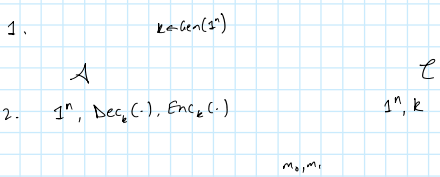
Story:

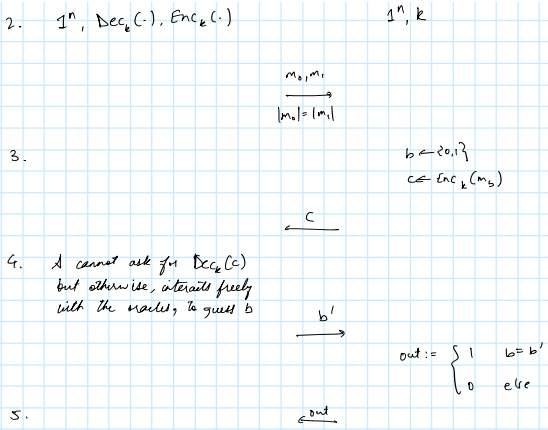
- Formally, the adversary is given access to a
  decryption oracle
  in addition to the encryption oracle.

Formal Definition

Let
$\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a private-key encryption scheme
$\mathcal{A}$ be an adversary
n for the security parameter

The CCA indistinguishability game/experiment $\mathbf{PrivK}^{cca}_{\mathcal{A},\Pi}(n)$





Definition 3.31

A private-key encryption scheem Π has
indistinguishable encryptions under a chosen-ciphertext attack **(CCA secure)**
if for all probabilistic poly-time adversaries $\mathcal{A}$ there exists a negligible function negl
such that

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

where the probability is taken over all random coins used in the experiment.

Remarks:

- The restriction on access to the decryption oracle is necessary
  otherwise no scheme can be secure

Question: Are CCA attacks realistic?
Response:

- As in the CPA attack,
  we don't expect honest parties to decrypt arbitrary ciphertexts (chosen by the adversary)
- Yet, there may be scenarios where
  an adversary may be able to influence what gets decrypted
  and learn partial information about the result

E.g.

1. In the case of Midway (as in Section 3.5)
   it is conceivable that the US cryptanalysts might also have
   tried sending encrypted messages to the Japanese
   and then tried monitering their behaviour.
   Such behaviour (e.g. movement of forces, etc) could have
   provided important information about
   the underlying plaintext.

2. Suppose a user is communicating with their bank
   where all communication is encrypted.

   If this communication is not authenticated
   then an adversary may be able to send
   certain ciphertexts
   on behalf of the user

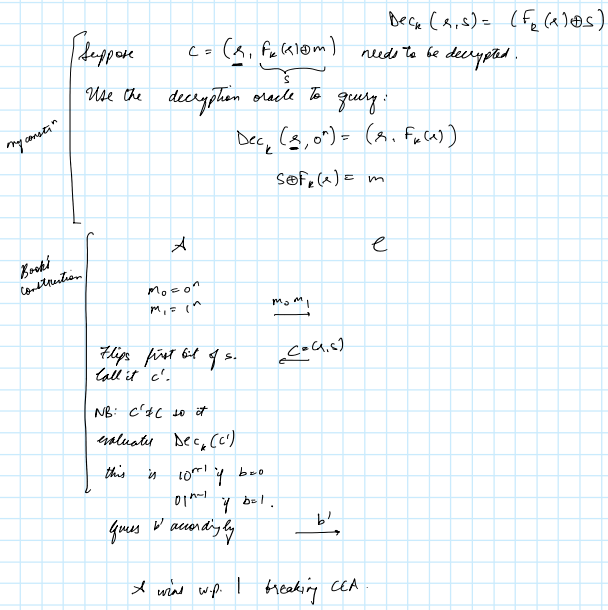
   The bank will decrypt these ciphertexts
   and the adversary may learn something about the result
   E.g. if a ciphertext corresponds to an ill-formed plaintext
   the adversary may be able to deduce this from
   the pattern of the subsequent communication.

3. Encryption is often used in higher-level protocols
   e.g. an encryption scheme might be used
   as part of an authentication protocol
   where one party sends a ciphertext to the other
   who decrypts it and returns the result
   (NB: we don't recommend such a protocol—but such protocols are sometimes suggested)
   In this case, one of the honest parties may exactly act like a decryption oracle.

Insecurity of the schemes we have studied.

Story:

- None of the encryption schemes we have seen is CCA secure
- Recall (in particular), in Construction 3.25,
  the encryption is carried out as $\mathrm{Enc}_k(m) = (r, F_k(r) \oplus m)$



Remark:
This example demonstrates why CCA security is so stringent

- Any encryption that allows ciphertexts to be
  manipulated in a "logical way" cannot be CCA secure

- Thus, CCA security implies a very important property
  non-malleability.

Informal Definition: Non-malleability:
If the adversary tries to modify a given ciphertext
the result is either an illegal ciphertext or
one that encrypts a plaintext having
no relation to the original one

Exercise: Show that none of the modes of encryption we saw are CCA secure.

Construction of a CCA-secure encryption scheme.

- Deferred to Section 4.8 (because we need tools from Chapter 4)

References and Additional Reading

- **Goldwasser Micali**—computational approach to cryptography
  notion of semantic security
  showed this goal could be achieved in the public key setting
- Formal definitions of security against CPA
  Luby and Bellare et al
  CCA (in public key) were formally defined by **Noar-Yung** and **Rackoff-Simon**

- Other notions of cryptography

- Pseudorandomness was first introduced by **Yao**
  Pseudorandom generators were defined and
  cosntructed by **Blum and Micali**
  who also pointed out their connection
  to encryption via stream ciphers
  (use of stream ciphers for encryption pre-dated the formal notion of
  pseduorandom generators)
- Pseudorandom functions
  were defined and constructed by **Goldreich** et al
  and their application to encryption was demonstrated
  in subsequent work by the same authors
  Pseudorandom permutations were studied by **Luby** and **Rackoff**