Computational security & Private key encryption

Emma Arfelt
Stud. BSc. Software Development

Frederik Madsen
Stud. MSc. Software Development

March 2017

Recap

- Perfect Secrecy
- Perfect indistinguishability
- The One-time pad
- Limitations of perfect secrecy

Outline

- Computational Secrecy
  - Concrete relaxation
  - Asymptotic relaxation
- Computational secure encryption
- Constructing computational secure encryption
  - Pseudorandomness
  - EAV-secure schemes
  - Stronger security notions
  - Pseudorandom functions
  - Pseudorandom permutations
  - CPA-secure schemes
  - Defining CCA security

Computational Secrecy

Computational secrecy

Definition 2.3

> Perfect secrecy requires that absolutely no information about an encrypted message is leaked, even to an eavesdropper with unlimited computational power.

This definition seems unnecessary strong and has inherent drawbacks

Could we relax this definition without compromise actual security?

Computational secrecy

Information

We could relax the amount of information that we accept to be leaked.

We are going to allow security to fail with some *tiny probability*, but assume that the adversary has *bounded computational resources*.

Computational secrecy

Tiny?

What do we mean by a *tiny probability*?

For instance 2^{-60} is a tiny probability, but it is secure enough?

We assume that with probability 2^{-60}, we have have bigger problems to worry about than failure of the encryption scheme.

Computational secrecy

Adversaries

An adversary with *bounded computational ressources* is an **efficient attacker**

For an attacker doing a Brute-Force search over the key space with one key pr. clock cycle:

- Desktop computer: 2^{57} keys pr. year
- Super computer: 2^{80} keys pr. year
- Super computer since the Big Bang: 2^{112} keys in total

Most modern key spaces are $> 2^{128}$

We can now narrow our focus to only efficient attackers, since our encryption scheme would still be secure when adversaries have bounded computational resources.

Computational Secrecy

Definition

Computational secrecy allows adversaries to **potentially succeed** with some very small probability and only guarantee security against **efficient adversaries** that run for some feasible amount of time.

Perfect Indistinguishability

Remember from last time that a scheme $\Pi = (Gen, Enc, Dec)$ is perfect indistinguishable, if for every M

Definition 2.5

$$Pr\left[PrivK^{eav}_{A,\Pi} = 1\right] = \frac{1}{2}$$

We claim that Π is perfectly indistinguishable if and only if it's perfectly secret. This means that perfect indistinguishability is just an alternate way of defining perfect secrecy.

Computational Secrecy

Relaxations

We have two ways of relaxing the previous definition of perfect indistinguishability:

- Concrete Approach
- Asymptotic Approach

Computational Secrecy

Concrete approach

Computational security

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

Computational indistinguishability:

$$Pr\left[PrivK^{eav}_{A,\Pi} = 1\right] \leq \frac{1}{2} + \epsilon$$

(t, ϵ)-indistinguishability:

- Security may fail with probability $\leq \epsilon$
- Restrict our attention to attackers running in time $\leq t$

But, what can we assumme about the adversary here?

Computational Secrecy

Asymptotic Approach

Introduce a *security parameter* denoted by n that parameterizes both cryptographic schemes as well as all involved parties. [1]

The running times of all parties and the success probability of the adversary can be viewed as *functions of* n

[1] For now we can view n as the key lenght

Computational Secrecy

Asymptotic Approach

Computational security

> A scheme is secure if any probabilistic polynomial time adversary succeeds in breaking the scheme with at most negligible probability

Computational indistinguishability:

- Security may fail with probability *negligible in n*
- Restrict our attention to attackers running in time *polynomial in n*

Asymptotic Approach

Definitions

Why are *polynomial* and *negligible* good choices to define *efficient* and *tiny*?

Efficient means there is some polynomial p such that the adversary runs for time at most $p(n)$ when the security parameter is n. The notion that this should be *"efficient"* is borrowed from complexity theory.

Tiny means that success probabilities should be smaller than any inverse polynomial in n, which is negligible.

Asymptotic Approach

Negligible functions

Why do we wish to use negligible functions?
We need to formalize the *slack* we are allowing in our new definition

Definition 3.4

A function f is negligible *if* for every polynomial p and *all sufficiently large values of* n it holds that $f(n) < \frac{1}{p(n)}$

If the effeciency of our adversary grows the probability that he succeeds should become smaller.

Summary

- We have introduced computational secrecy, which introduces two relaxations of perfect secrecy
- We have identified efficient adversarial strategies as probabilistic polynomial-time algorithms
- We have equated small chances of success with negligible probabilities.

Defining Computationally Secure Encryption

Definition of security

Threat model

- An attacker is only capable of eavesdropping
- An attacker runs in polynomial time

Security goal

- Security when a given key is used to encrypt a *single* message
- Semantic *security**

Computational Security for private-key encryption

Encryption scheme Π:

- $k \leftarrow Gen(1^n)$ where $| k | \geq n$
- $c \leftarrow Enc_k(m)$ where $m \in \{0,1\}^*$
- $m := Dec_k(c)$

We now require that these algorithms run in probabilistic polynomial time, since we want all parties to run efficiently.

Computational indistinguishability

The asymptotic version

Experiment $PrivK^{eav}_{A,\Pi}(n)$

1. The adversary A is given input 1^n; Outputs m_0, m_1 where $|m_0| = |m_1|$

2. $k \leftarrow Gen(1^n)$ generates a key and uniform bit $b \in \{0, 1\}$ is chosen. $c \leftarrow Enc_k(m_b)$ is computed and given to A

3. A outputs bit b'

4. A succeeds if $b' = b$, thus $PrivK^{eav}_{A,\Pi}(n) = 1$

Definition of security

Indistinguishability

A private-key encryption scheme Π is indistinguishable if for all PPT adversaries A there is a negligible function negl such that, for all n

Definition 3.8

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

We allow the adversary to have a *negligible* better probability than just guessing, but requires that the probability must decay faster to zero than any inverse polynomial.

Definition of security

Indistinguishability

The adversary should be *behave the same* whether it sees an encryption of m_0 or m_1. It outputs 1 with the same probability in each case:

Definition 3.9

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

Computational Security with Asymptotic approach

Example

Brute-Force search over the key space

- $Gen(1^n)$ generates a uniform n-bit key
- The adversary A runs in time $t(n)$

From previous experiment:

$$Pr[PrivK^{eav}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + t(n) * \frac{1}{2^n} \quad [2]$$

Why is this secure?

[2] 2^n represents the key space

Computational Security with Asymptotic approach

Example

Remember that for any attacker running a probabilistic polynomial time, the success probability of that attacker is at most $\frac{1}{2} + \text{negl}(n)$

Closure properties:

- $poly * poly = poly$
- $poly * negl = negl$

So for any attacker running in polynomial time $t(n)$ must then be polynomial.

Moreover the function $t(n) * \frac{1}{2^{-n}}$ is negligible, since any polynomial times a negligible function remains negligible.

Computational Security with Asymptotic approach

Example

For any probabilistic polynomial time attacker that runs in time $t(n)$, we do not need to know *what* the attacker is doing, as long as we can be sure that it is polynomial bounded

This allows us to be sure that the success probability of that attacker is going to be $\leq \frac{1}{2} + \text{negl}(n)$

Thus the scheme is **computational secure**

Constructing Secure Encryption Schemes

Pseudorandomness

Informal definition

It turns out, that to create computationally secure encryption schemes, we need `Pseudorandom Generators`:

A pseudorandom generator, G, is an efficient deterministic algorithm that given a uniform seed, s, produces a longer "uniform looking", or `pseudorandom`, output.

Pseudorandomness

Formal Definition

Definition 3.14 (expansion omitted)

- For any PPT algorithm, D, there is a negligible function negl, s.t.:

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

Distinguishers are roughly for pseudorandom generators, what adversaries are for encryption schemes.

Pseudorandomness

Stream ciphers

We cannot unconditionally prove that pseudorandom generators exists.
However, we have some candidate pseudorandom generators, where no efficient distinguishers are known, called `Stream Ciphers`:

- Stream ciphers allow for an arbitrary number of pseudorandom bits
- Stream ciphers are essentially state machines, generating pseudorandom data on demand:

ALGORITHM 3.16
Constructing G_ℓ from (Init, GetBits)

Input: Seed s and optional initialization vector IV
Output: $y_1, \ldots, y_\ell$

$\mathsf{st}_0 := \mathsf{Init}(s, IV)$
for $i = 1$ to ℓ:
  $(y_i, \mathsf{st}_i) := \mathsf{GetBits}(\mathsf{st}_{i-1})$
return $y_1, \ldots, y_\ell$

Pseudorandom generator summary

- Deterministic efficient function, that "expands" the randomness of a given seed.
- Distinguishers are the adversaries of pseudorandom generators.
- $|Pr[D(G(s)) = 1] - Pr[D(r) = 1]| \leq \text{negl(n)}$.

EAV-secure, single-message encryption

	Single	Multiple
EAV	X	
CPA		
CCA		

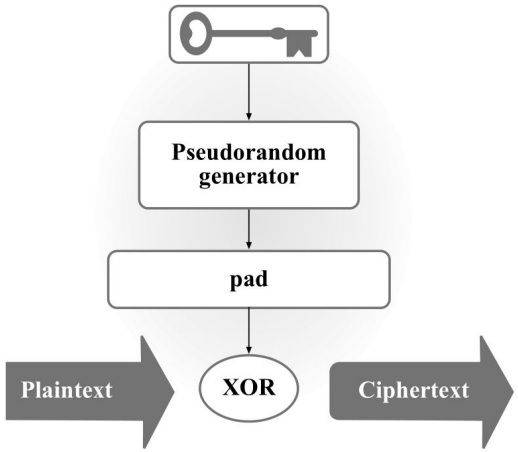

FIGURE 3.2: Encryption with a pseudorandom generator.

CONSTRUCTION 3.17

Let G be a pseudorandom generator with expansion factor ℓ. Define a private-key encryption scheme for messages of length ℓ as follows:

- Gen: on input 1^n, choose uniform $k \in \{0,1\}^n$ and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$, output the ciphertext
$$c := G(k) \oplus m.$$
- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$, output the message
$$m := G(k) \oplus c.$$

- Intuition: one-time pad with pseudorandom data
- For the proof, we denote this algorithm Π

Proof

Computationally secure, single-message, EAV-scheme

	Single	Multiple
EAV	X	
CPA		
CCA		

Since the scheme is exactly one-time pad, except with pseudorandom data as key, the adversary must be able to distinguish between random and pseudorandom data.

Proof by reduction:
Let A be a PPT adversary. We construct D to be a Distinguisher, that takes a string w, and has to guess whether w is the product of true randomness or a pseudorandom generator. It does this by emulating the eavesdropping experiment for A:

1. Run $A(1^n)$ to get m_0 and m_1.
2. Choose a $b \in \{0, 1\}$. Set $c := w \oplus m_b$.
3. Give c to A, which produces output b'.

Proof - cont'd

Computationally secure, single-message, EAV-scheme

	Single	Multiple
EAV	X	
CPA		
CCA		

Let $\widetilde{\Pi} = (\widetilde{Gen}, \widetilde{Enc}, \widetilde{Dec})$ denote the one-time pad.

Because the one-time pad is perfectly secret, it must hold that:

$$Pr_{w \leftarrow \{0,1\}^{l(n)}}[D(w) = 1] = Pr[PrivK^{eav}_{A,\widetilde{\Pi}} = 1] = \frac{1}{2}$$

Because of the way we constructed D, it must also hold that:

$$Pr_{k \leftarrow \{0,1\}^{n}}[D(G(k)) = 1] = Pr[PrivK^{eav}_{A,\Pi} = 1]$$

Proof - cont'd

Computationally secure, single-message, EAV-scheme

	Single	Multiple
EAV	X	
CPA		
CCA		

Because the G is a pseudorandom generator, it must hold that:

$$|Pr_{w \leftarrow \{0,1\}^{l(n)}}[D(w) = 1] - Pr_{k \leftarrow \{0,1\}^{l(n)}}[D(G(k)) = 1]| \leq \mathsf{negl}(n)$$

Thus, using substitution, we get that:

$$|\frac{1}{2} - Pr[PrivK^{eav}_{A,\Pi} = 1]| \leq \mathsf{negl}(n)$$

EAV-secure scheme summary

- Uses key as seed in pseudorandom generator, and pad message with $G(k)$.
- Proof by reduction: generate D by using A and simulate indistinguishability experiment.

Stronger security notions

Encryption schemes with multiple messages

	Single	Multiple
EAV		X
CPA		
CCA		

However, this encryption scheme does not hold, if several messages were to be sent. To show this, we have to define a new eavesdropping experiment, with multiple messages, $PrivK^{mult}_{A,\Pi}(n)$:

1. A is given input 1^n, and produces two lists of equal length, $\overrightarrow{M_0} = (m_{0,0}, \ldots, m_{0,t})$ and $\overrightarrow{M_1} = (m_{1,0}, \ldots, m_{1,t})$.
2. k is generated by $Gen(1^n)$, and a uniform $b \in \{0, 1\}$ is chosen. Each message in $\overrightarrow{M_b}$ is encrypted and given to A.
3. A outputs b'.
4. The output of the experiment is 1 if $b' = b$, else 0.

NB! The original eavesdropping experiment is only a special case of this.

Stronger security notions

Argument for probabilistic encryption schemes

	Single	Multiple
EAV		X
CPA		
CCA		

It turns out that all of the encryption-schemes we have seen so far, are not secure under this security notion.

(Informal) proof:
Run the $PrivK^{mult}$ experiment on the one-time pad and let A choose $\overrightarrow{M_0} = (0^l, 0^l)$ and $\overrightarrow{M_1} = (0^l, 1^l)$.

This fails because the one-time pad is deterministic and stateless.
This leads the book to define the following theorem:

Theorem 3.21

> If Π is a (stateless) encryption scheme in which Enc is a deterministic function of the key and message, then Π cannot have indistinguishable multiple encryptions in the presence of an eavesdropper

Thus, we must design schemes using `randomized` encryption

Stronger security notions

Chosen-plaintext, single message, attack

	Single	Multiple
EAV		
CPA	X	
CCA		

We now extend the security notion to chosen-plaintext, single message, attacks. As before, we must define the cpa indistinguishable experiment $PrivK^{cpa}_{A,\Pi}$:

1. k is generated by running $Gen(1^n)$
2. A is given input 1^n and oracle access to $Enc_k(\cdot)$, and produces m_0 and m_1 of the same length.
3. The uniform bit $b \in \{0, 1\}$ is chosen, $c \leftarrow Enc(m_b)$ is computed and given to A
4. A still has access to the oracle, and outputs b'
5. The output of the experiment is 1 if $b' = b$, else 0.

The only real difference between this and the eavesdropping experiment, is the oracle access.

Stronger security notions

Chosen-plaintext, multiple messages, attack

	Single	Multiple
EAV		
CPA		X
CCA		

The book has a similar experiment, $PrivK_{A,\Pi}^{LR-cpa}$, for multiple messages, where it uses an $LR_{k,b}$-oracle:

- Generalizes the message-list from $PrivK_{A,\Pi}^{mult}$
- Models adversaries that adaptively chooses plaintexts to be encrypted

NB! The experiment could just as well have been extended with message lists.

However, it turns out that this experiment doesn't really matter.

Stronger security notions

$CPA_{single} \iff CPA_{mult}$ argument

	Single	Multiple
EAV		
CPA	X	X
CCA		

It should be somewhat evident, that an encryption-scheme that is secure in the $PrivK^{LR-cpa}_{A,\Pi}$ experiment, is also secure in $PrivK^{cpa}_{A,\Pi}$. Surprisingly, the opposite also holds.

Theorem 3.24

> Any private-key encryption scheme that is CPA-secure is also CPA-secure for multiple encryptions.

Not proven in the book, but it has a similar proof in section 11.2.2. Using this fact, we can also construct an arbitrary-length CPA-secure encryption-scheme, by concatenating encryptions of a fixed length.

Pseudorandom functions

Before we can start designing CPA-secure encryption schemes, we have to understand the notion of pseudorandom functions.

Definition 3.25

Let F: $\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be an efficient, length-preserving, keyed function. F is a pseudorandom function if for all probabilistic polynomial-time distinguishers, D, there is a negligible function negl such that:

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

NB! D does not know k!

Notice that $n = l_{in} = l_{out}$

Pseudorandom permutations

Definition 3.28

Let F: $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving, keyed function. F is a strong pseudorandom permutation if for all probabilistic polynomial-time distinguishers, D, there is a negligible function negl such that:

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

Intuition: A permutation is a bijection between two sets. I.e., we now have an inverse function.

Example: Block-ciphers.

Designing CPA secure functions

Deterministic encryption

	Single	Multiple
EAV		
CPA	X	X
CCA		

Recall that we needed to make randomized (or stateful) encryption-schemes, to be able to handle multiple messages.

As such, the intuitive, and naive, approach will not work:

$$Enc_k(m) = F_k(m)$$

Designing CPA secure functions

Randomized encryption

	Single	Multiple
EAV		
CPA	X	X
CCA		

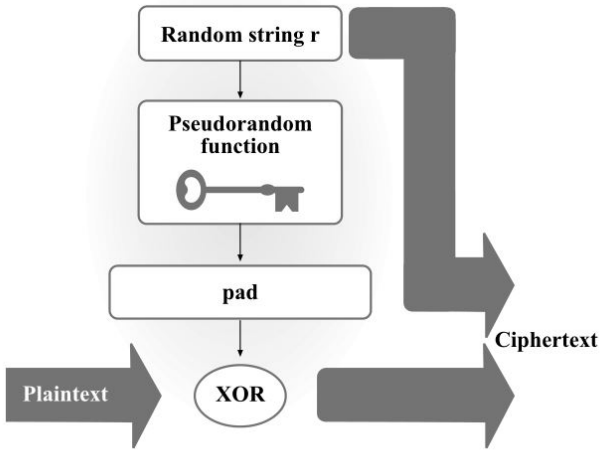

FIGURE 3.3: Encryption with a pseudorandom function.

CONSTRUCTION 3.30

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 uniform $k \in \{0,1\}^n$ and output it.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose uniform $r \in \{0,1\}^n$ 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

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

- Randomization here comes from the randomized string.
- Notice that this is not randomized if r is reused. This happens only with negligible probability.

Designing CPA secure functions

Two step proof

	Single	Multiple
EAV		
CPA	X	X
CCA		

- Step one: Show that if *Enc* used a truly random function, f, the adversary is not significantly effected
- Step two: Analyze the truly random encryption-scheme

Proving CPA secure functions

Step one

	Single	Multiple
EAV		
CPA	X	X
CCA		

We want to prove:

$$|Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] - Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n) = 1]| \leq \text{negl}(n)$$

We use the same procedure as in the *EAV*-single proof:

- Construct distinguisher, D, which "wraps around" A. This time A has access to an oracle, which D simulates
- Show that the two cases of D (Π and $\widetilde{\Pi}$) are equivalent $PrivK^{cpa}_{A,\Pi}(n)$ and $PrivK^{cpa}_{A,\widetilde{\Pi}}(n)$
- Since we know that $|Pr[D(w) = 1] - Pr[D(G(k)) = 1]| \leq \text{negl}(n)$, due to the definition of pseudorandomness. Using substitution, the top equation must hold.

Proving CPA secure functions

Step two

	Single	Multiple
EAV		
CPA	X	X
CCA		

We want to prove:

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

Let $r*$ denote the original random string, used to encrypt m_b (recall that $c := < r*, f(r*) \oplus m_b >$). Two cases, when A asks the oracle:

- The oracle never reuses $r*$, which is equivalent to $f(r*)$ being uniformly distributed and independent, and thus this is the same case as one-time pad.
- The oracle reuses $r*$, and A is easily able to determine whether m_0 or m_1 was encrypted.

The probability of the oracle reusing $r*$ is $\frac{q(n)}{2^n}$

Proving CPA secure functions

Putting the steps together

	Single	Multiple
EAV		
CPA	X	X
CCA		

Since $\frac{q(n)}{2^n}$ is negligible, we get:

$$Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n) = 1] \leq \frac{1}{2} + \frac{q(n)}{2^n} \iff$$
$$Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + \frac{q(n)}{2^n} + \mathsf{negl}(n) \iff$$
$$Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}'(n)$$

Modes of operation

Stream ciphers

Stream cipher can work as the underlying pseudorandom generator/function of synchronized encryption algorithms:

- Synchronized Mode.
  - Using the number and length of messages sent, to achieve a stateful pseudorandom generator.
  - Only *EAV*-secure, but with multiple messages.
- Unsynchronized mode.
  - Using the initialization vector to act as the random string, r, and such that it becomes a pseudorandom function.
  - CPA-secure.

CCA-security

The indistinguishability experiment

	Single	Multiple
EAV		
CPA		
CCA	X	

Recall that in a chosen ciphertext attack, the adversary gets to decrypt ciphertexts of its choice, *except* the "challenge ciphertext" itself.

This is the appropriate indistinguishability experiment:

1. k is generated by running $Gen(1^n)$
2. A is given input 1^n and oracle access to $Enc_k(\cdot)$ and $Dec_k(\cdot)$, and produces m_0 and m_1 of the same length.
3. The uniform bit $b \in \{0, 1\}$ is chosen, $c \leftarrow Enc(m_b)$ is computed and given to A
4. A still has access to the oracles, and outputs b'
5. The output of the experiment is 1 if $b' = b$, else 0.

The only difference is the $Dec_k(\cdot)$ oracle.

CCA-security

Multiple encryptions

	Single	Multiple
EAV		
CPA		
CCA	X	X

- Theorem 3.24 also holds for CCA-secure encryption schemes.
  - That is, if a scheme has indistinguishable encryptions under CCA, then it has indistinguishable `multiple` encryptions under CCA.

CCA-security

Non-malleability

	Single	Multiple
EAV		
CPA		
CCA	X	X

It turns out that all the encryption schemes we have seen so far are insecure under CCA.

Example:
Recall the 3.30-scheme where $Enc_k(m) =< r, F_k(r) \oplus m >$.
A can simply choose $m_0 = 0^n$ and $m_1 = 1^n$.
When it receives the ciphertext, it flips the first bit, gets the decryption, and check which of m_0 and m_1 is equal to the decryption, minus the first bit.

This is because the 3.30-scheme (as well as all the other schemes seen so far), is malleable.
I.e. because a change in the ciphertext can be translated directly onto a change in the plaintext.

For CCA-security, we will look for schemes that are non-malleable.

Summary

- Computational Secrecy is a relaxation of perfect secrecy
  - We used the asymptotic approach to prove indistinguishability, and we will continue to do so
- We defined computational secure encryption as encryption with at most negligible probability of failure
- We defined a EAV-secure scheme, by using a key as a seed in a pseudorandom generator and padding
- We extended the security notion with CPA-security and security under multiple encryptions
- We showed a CPA-secure scheme, and proved it correct
- We defined CCA security, which we will continue to apply next time