Crypto

Information Theoretic (Perfect) Security

Definition 1. *a private-key encryption scheme is (G, E, D) perfectly secure if for all $m_0, m_1 \in \mathcal{M}$, $c \in \mathcal{C}$, and uniformly random variable k on $\mathcal{K}$ if*

$$k \leftarrow (1^n) \ \Pr[E_k(m_0) = c] = \Pr[E_k(m_1) = c]$$

Limitations

- implies security against any adversary, even with unbounded computation
- statement about the properties of the encryption scheme itself
- k is the only random variable, an attacker gets no information about the message
- doesn't model an adversary, assumes nothing is leaked at all!
- What if cA knew some information before? like the language of the ct, or if its a yes or no answer?

Semantic Security

Definition 2. *A private-key encryption scheme (G, E, D) is semantically secure (in the private key model) if for every non-uniform probabilistic polynomial time algorithm $\mathcal{A}$ there exists a non-uniform probabilistic-polynomial time algorithm $\mathcal{A}'$ such that for every probability ensemble $\{X_n\}_{n\in\mathbb{N}}$ with $|X_n| \leq poly(n)$, every pair of polynomially-bounded functions $f, h : \{0,1\}^* \to \{0,1\}^*$ every positive polynomial $p(\cdot)$ and all sufficiently large n*

$$k \leftarrow (1^n) \ \Pr\left[\mathcal{A}(1^n, E_k(X_n), 1^{|X_n|}), h(1^n, X_n) = f(1^n, X_n)\right]$$

$$< \Pr\left[\mathcal{A}(1^n, \cancel{E_k(X_n)}, 1^{|X_n|}), h(1^n, X_n) = f(1^n, X_n)\right] + \frac{1}{p(n)}$$

- any information $\mathcal{A}$ can compute about the plaintext X_n, cA' can compute almost as well without the ciphertext. Except for a negligible advantage.
- Here, $\mathcal{A}$ is in the real world with access to the ciphertext and the simulator $\mathcal{A}'$ doesn't.
- Semantic Security holds if the 2 scenarios are computationally indistinguishable
- If computational indistinguishability didn't hold, then $\mathcal{A}$ would have a higher probability to guess

Limitations

- assumes unlimited computational power
- leads to impossibility results

Moving to Semantic Security

- assumes $\mathcal{A}$ is bounded by probabilistic polynomial time algorithms
- easily accounts for auxillary information

Preliminaries and notation

- $S \subseteq \{0,1\}^*$ defines set S as a finite subset of $\{0,1\}$ all finite-length strings.
- $x \in_R S$, R indicates x is chosen randomly / uniformly from S
- U_n x is chosen from the set of all n-bit strings
- $\mu(\cdot)$ means the negligible function can take any input. Negligible functions decrease faster than inverse polynomial as n increases
- must use positive polynomial, if not, maybe it won't be negligible. We want $\mu(n) < \frac{1}{p(n)}$
- λ is an empty string

Polynomial time, Security parameter

- Polynomial time refers to the computational complexity of an algorithm with respect to the security parameter
- it means the protocol is efficient and practical because algorithms are feasible in polynomial time
- But also secure because breaking the algorithm is infeasible
- security parameter determines level of security e.g. length of keys in bits
- an algorithm running in polynomial time in the security parameter can be expressed as a polynomial function of the security parameter. That is, there exists $p(\lambda)$ such that $\mathcal{O}(p(\lambda))$
- E.g. if $\lambda = 128$ bits runs in polynomial time, running time will be a function of λ like $\mathcal{O}\lambda^2$
- In contrast to exponential time which are $\mathcal{O}2^\lambda$

Theory of Computation

- a turing machine is a theoretical device that "manipulates symbols on a strip of tape according to a table of rules" simulating algorithm logic
- Includes an infinitely long tape divided into blocks, a head can read and write symbols on the tape, a state register storing the machine state, a finite table of instructions
- We use unary 1^n, a string of 1's on the security paramter tape for reasons:
  - Unary: 1^n provides unary representation of n meaning input length corresponds with n, the longest possible representation (Worst case and Lower bound)
  - in binary, n is represented in $log_2(n)$ bits. e.g. 1000 is 11111101000 = 10 bits long. In unary, 1^{1000} is a string of 1000 ones.
  - using binary, an algorithm taking time proportional to input length would run in $\mathcal{O}(log(n))$ which runs in $\mathcal{O}(n)$
- Security parameter tape is used to model how a system scales with security parameter, 1^λ is written on it e.g. string of 1's
- This means, the function is bounded by the length of the input on the security parameter tape
- Advice Tape: is used in non-uniform computation, it's additional information given to an algorithm

Uniform, Non-Uniform Non uniform algorithms aren't non-uniform randomness! Uniform algorithms don't change procedure based on the input size. Non-uniform algorithms can have logic based around the input size. Uniform algorithms have a fixed strategy, non-uniform can adapt. The non-uniformity is not about randomness but the potential for the algorithm to have different strategies for input lengths. Allowing a distinguisher algorithm D to be non-uniform means it's a powerful attacker that has different strategies for each input length (key length or message size) and can more easily distinguish.

Negligible Function A function $\mu(n)$ is negligible if it decreases faster than the inverse of any polynomial.

Definition 3. *given a function $\mu : \mathbb{N} \to [0, 1]$, we say μ is negligible if for all polynomials p, there exists $n_0 \in \mathbb{N}$ such that*

$$\forall\, n \geq n_0,\ \ \mu(n) \leq \frac{1}{p(n)}$$

Tests: are the following negligible?

1. $\mu(n) = \frac{1}{n^2}$
2. $\mu(n) = \frac{1}{2^n}$
3. $\mu(n) = \frac{1}{n!}$
4. $\mu(n) = \frac{1}{2^{-n}}$
5. $\mu(n) = \frac{1}{2^{\log(n)}}$
6. $\mu(n) = \frac{1}{n^{\log(n)}}$

Answer and Discussion

1. $\frac{1}{n^k}$ is a inverse power function, aka polynomial time decreasing function. e.g. inverse cubic $1/n^3$, inverse quartic $1/n^4$, they approach 0 as n approaches ∞. They are efficiently computable and tractable, though non-negligible.

2. $\mu(n) = \frac{1}{2^n}$ is an inverse exponential function and will always satsify the inequality because an exponential function will decrease faster than the inverse polynomial. It's non-negligible and used in crypto

3. $\mu(n) = \frac{1}{n!}$ is an inverse factorial, defined only for non-negative, it's super exponential decreasing faster than exponential and isn't seen in computer science but maybe in poisson distribution. It's non-negligible but not used

4. $\mu(n) = \frac{1}{2^{-n}}$ without calculation, this is 2^n which is exponential growth rather than decay, definately not negligible.

5. $\frac{1}{2^{\log(n)}}$ is negligible but is sub-exponential, decreasing slower than $\frac{1}{2^n}$. For any $p(n)$ we show that a large enough n satisifes our inequality. $\frac{1}{2^{\log(n)}} = \frac{1}{n^{\log(2)}}$, $n^{\log(2)} = n^{0.693}$ therefore for sufficiently large n, this satifies the inequality.

6. $\frac{1}{n^{\log(n)}}$ decreases faster than the above

log/exp rule: $x^{\log_a^{(y)}} = y^{\log_a^{(x)}}$

Inequalities Regularly, security of a crypto scheme is defined by something in the form of

$$\Pr[\text{Algo}() = 1] \qquad \text{inequality / comparator} \qquad \Pr[\text{Algo} = 1] \qquad \text{some inequality } f$$

I'll try to identify the differences in comparators and inequalities.
First, the definition of computational indistinguishability:

- First: Computational Indistinguishability

$$|\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]| \leq \mu(n)$$

- We look for the absolute value. We minus the first probability from the second and bound that to be at-most negligible.

- let's analyse the impact of changing bounds:

$$a. \leq \mu(n) \qquad b. < \frac{1}{p(n)} \qquad c. \geq \frac{1}{p(n)}$$

- a and b are comparable since $\mu(n)$ is defined by the bound $\mu(n) < 1/p(n)$ if for positive polynomial $p(\cdot)$ and sufficiently large n and that's why it uses $<$ rather than $\leq$
-

Distinguishing Advantage Quantifies D's ability to distinguish between X and Y when given a sample from each $\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]$. The definition states this distinguishing advantage must be $\leq$ some negligible function $\mu(n)$ for suffiently large n.

Algorithm bounds The definition of computational indistinguishability states the distinguishing advantage $\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]$ is bound by a negligible function which by definition "Given unbounded computational power" "Brute force atttack"

Computational Indistinguishability

Two probability ensembles, X, Y are computationally indistinguishabile: $X \stackrel{c}{\equiv} Y$ if

$$|\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]| \leq \mu(n)$$

- n is input length, the security parameter
- a a binary string of any length, could be public parameters
- ensembles X, Y are computationally indistinguishable if no polynomial time algorithm D can tell them apart with greater than negligible advantage.
- D is a PPT algorithm trying to distinguish between samples from X and Y, not guessing the bit
- $D's$ output is binary $\{0,1\}$. Output 1 means a successful distinguish
- We look at the absolute value difference in probability of D outputting 1 for X vs Y

$$X = \{X(a,n)\}_{a \in \{0,1\}^*; n \in \mathbb{N}}$$

- Set $X = \{X(a,n)\}$ defines the set of random variables X
- Subscript $a \in \{0,1\}^*; n \in \mathbb{N}$ defines the indexing, that is, exactly what values of a, n can take for infinitely any element in set X. e.g. $X('01', 3)$ is the index of a random variable X where a is a binary string of any finite length "0, 1, 01, 000", n is a natural number 1,2
- $\{0,1\}^*$ is the Kleene star operation, means all finite strings, an infinite set because there's no limit to the length
- $n \in \mathbb{N}$ means n can be any natural number, also an infinite set
- indexing gives us an address or a way to talk about 1 specific random variable rather than the collection
- "probability ensemble" is a term in cryptography / probability theory referring to a collection or family of probability distributions or random variables. Used to describe systems where behaviours depend on on input length, security parameter, etc
- "ensemble" means we're dealing with a collection of probabilistic objects rather than a single fixed distribution

Non-uniformity

- D is defined above as non-uniform which increases its power to distinguish
- basically says the concept of computational indistinguishability is non-uniform. Even if we start with uniform ensembles of X, Y, the distinguisher that potentially breaks indistinguishability might be non-uniform
- Non-uniformity comes from the fact that for different input length n, there may be different aux input a that allows a distinguisher to win
- The value a is a public parameter that needs to be "written on the advice tape of the reduction algorithm
- This means securtity proofs and analysis need to consider non-uniform adversaries

Order of quantifiers for computational indistinguishability

- the main distinction this section is making is allowing μ the negligible function to depend on a, as in μ_a or $\mu(a,n)$ is very different to $\mu(\cdot)$ the prior definition of a neligible function.
- a negligible function that's parameterized by a leads to a weaker security definition for computational complexity.
- The negligible probability of distinguishing doesn't vary on the specific problem instance a, only on the security parameter n.
- a quantifier specifies the number of elements in a domain satisfy a given predicate. E.g. for all $\forall$, there exists $\exists$. This section identifies the fact that ordering changes the meaning of a statement.
- "For all a, there exists a negligible function" is different to saying "there exists a negligible function for all a". The former says that every a uses the same negligible function, the latter says that all a have a negligible function but it could use different.

- $X \stackrel{c}{\equiv} Y$ if for every non-uniform ppt algo D, there exists a negl. funcion $\mu(\cdot)$ for every $a \in \{0,1\}^*$ and every $n \in \mathbb{N}$ such that

$$\{X(a,n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}} \stackrel{c}{\equiv} \{Y(a,n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$$

  Is not the same as: for every $a \in \{0,1\}^*$ it holds that

$$\{X(a,n)\}_{n\in\mathbb{N}} \stackrel{c}{\equiv} \{Y(a,n)\}_{n\in\mathbb{N}}$$

- For the piecewise function below, $|a|$ denotes the length of a bit string a.

$$\mu_a(n) = \begin{cases} 1, & \text{if } n < 2^{|a|} \\ 2^{-n}, & \text{if } n \geq 2^{|a|} \end{cases}$$

  If the security parameter n is less than the length of the bit string a then the negligible function is not negligible.
  If the security parameter n is more than the length of the bit string a, the negligible function is exponential in n and thus negligible.
- a can be the parameters of a crypto scheme, a public key, other instance specific information. Security should not rely on an instance of a problem, rather the security parameter.

Semantic Security

- polynomial length plain texts = plaintext length is bounded by a polynomial function of the security parameter. Notice the maximum length of the plaintext depends on a polynomial function of the security parameter secparam^3 rather than secparam^n

```
fn A(plaintext, secparam):
    max_len = secparam^3
    if len(plaintext) > max_len:
        return plaintext[:max_len] //or return error
    else:
        return plaintext
```

- Arbitrary distributions of plaintext: refers to any distribution of plaintexts, e.g. uniformly random such as encrypting output of a hash function or uuid, non-uniform such as encrypting names, ages, or email addresses where distribution is clustered around ranges or common names, or fixed distribution such as encryption of "YES" or "NO" type responses.
- Aim of the adversary is to learn some function f of the plaintext:
  We model this scenario

```
fn adversary_guess(ciphertext, auxillary_info, f):

    % advesary strategy to guess f(plaintext)
    guess = secret_strategy(ciphertext, auxillary_info)
    return guess

fn evaluate_security(plaintext, ciphertext, auxillary_info, f):
    actual_value = f(plain_text)
    guess = adversary_guess(ciphertext, auxillary_info, f)
    return guess == actual_value
```

  The adversaries wins if their function f learns anything about the ciphertext, such as the first bit, if a number is even or odd, the length of a string.
- Auxillary Information: denoted as h is additional information available to the adversary, like partial information of the ciphertext e.g. the language of the plaintext, or side-channel information.