Computational Security (计算安全)

S. Zhong Y. Zhang

Computer Science and Technology Department
Nanjing University

Outline

Information-theoretically Secure versus Computationally Secure

Computationally security introduces *two relaxations* of perfect security:

- **Information-theoretically Secure** (**Perfectly Secure**): Adversaries with unlimited computation capability do not have enough information to launch a successful attack, thus always fail.
- **Computationally Secure**: Efficient adversaries have the information, and can potentially succeed with some very small probability.
    1. The concrete approach to define Computationally Security.
    2. The asymptotic approach to define Computationally Security.

Information-theoretically Secure versus Computationally Secure

Computationally security introduces *two relaxations* of perfect security:

- **Information-theoretically Secure** (**Perfectly Secure**): Adversaries with unlimited computation capability do not have enough information to launch a successful attack, thus always fail.
- **Computationally Secure**: Efficient adversaries have the information, and can potentially succeed with some very small probability.

2. The asymptotic approach to define Computationally Security. (We only consider this one here.)

Computational security is defined following a *parameterized* manner.

- The integer parameter is called the **security parameter** n. (e.g. usually n is the key length).

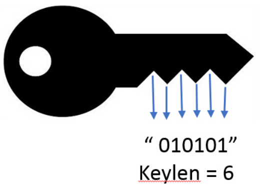


图: An example of “key length”

- A *greater security parameter* GENERALLY implies a *stronger security*.

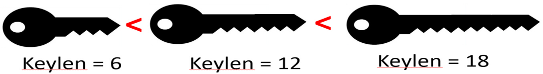


图: Longer key GENERALLY implies stronger security

Computational security: A parameterized security

- Why parameterized?
  Because *flexible*, *easy to measure/understand the security*,...
- But why do we need flexibility? Why not always use the security system with security parameter ∞?
  Because the cost of implementing, using such system also grows with the security parameter.

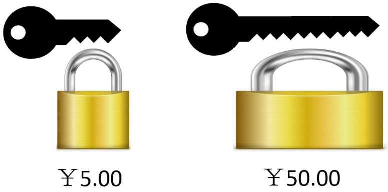


图: Longer key means bigger/more expensive lock

Computational security: A parameterized security

So, a parameterized security allows us to implement security pragmatically:

- "Big lock/safe for great assets!"

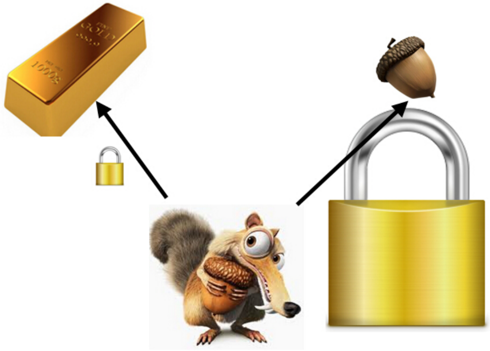


图: Big lock for great assets

Computational security: Against “efficient adversaries”

When defining computational security, we focus on efficient adversaries:

- If a system is secure against efficient adversaries, it should be also secure against non-efficient adversaries.

Non-efficient adversary

Efficient adversary

“God-like” adversary

图: Three kinds of adversaries

What are efficient adversaries and $poly(n)$?

What are efficient adversaries in the digital world?

- Efficient adversaries = Randomized algorithms + Polynomial-time bounded = **Probabilistic Polynomial-Time (bounded)** algorithms = **PPT** algorithms
- *Randomized algorithm*: currently accepted as feasible and powerful computations by practical computers.
- *Polynomial-time bounded*: Given the security parameter n, the algorithm runs no more than $poly(n)$ steps, where $poly(n)$ is a polynomial of n, i.e. can be represented as

$$poly(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0.$$

What are efficient adversaries and $poly(n)$?

What are efficient adversaries in the digital world?

- Efficient adversaries = Randomized algorithms + Polynomial-time bounded = **Probabilistic Polynomial-Time (bounded)** algorithms = **PPT** algorithms
- *Randomized algorithm*: currently accepted as feasible and powerful computations by practical computers.
- *Polynomial-time bounded*: Given the security parameter n, the algorithm runs no more than $poly(n)$ steps, where $poly(n)$ is a polynomial of n, i.e. can be represented as

$$poly(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0.$$

Why polynomial-time bounded?

Why polynomial-time bounded? Why not 2^n or n^n bounded?
Empirically, polynomial-time computations are considered practical.

- Example 1: "$10^8 \cdot n^4$ $(n = 80)$"
  2GHz computer,$10^8 \cdot n^4$ cycles $\approx$ 3 weeks.
- Example 2: "2^n $(n = 89)$" 2GHz computer, how long are 2^{89} cycles?
  2GHz computer,2^n cycles $= 2^{89} \cdot 2^{-31}$ seconds $= 2^{58}$ seconds.
  "Our universe's age since big bang is on the order of 2^{58} seconds."

Computational security: Allowing negligible success probability

- Why allowing negligible success probability?
  A: The cons of perfect security, e.g. key legth is no shorter than message length.
- What does negligible success probability mean?
  A: Given the security parameter n, the adversary's attack may succeed with a probability that is no greater than $negl(n)$.

What are superpoynomial functions and negligible functions?

- A function $superpoly(\cdot)$ is superpolynomial if.f. for every constant c, it holds that
$$superpoly(n) > n^c$$
when n is sufficiently large.[1]
- Negligible functions are the reciprocal of superpolynomial functions and vice versa.

[1] "sufficiently large" means for each c, there exists a N_c such that the ineqality holds for all $n \geq N_c$.

What is $negl(n)$?

定义 1.1.

A function $negl(\cdot)$ from the natural numbers to the non negative real numbers is **negligible** if for every positive polynomial p there is an N such that for all integers $n > N$ it hods that $negl(n) < \frac{1}{p(n)}$.

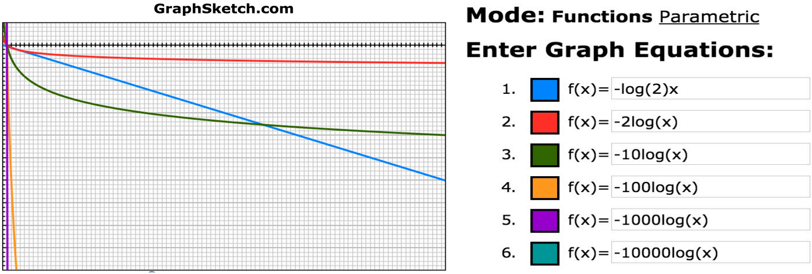
Equivalently,

定义 1.2.

A function $negl(\cdot)$ from the natural numbers to the non negative real numbers is **negligible** if for every positive integer c, there is an N_c such that for all integers $n > N_c$ it hods that $negl(n) < \frac{1}{n^c}$.

A example of negligible probability: $negl(n) = \frac{1}{2^n}$

- $negl(n) = \frac{1}{2^n}$ decreases much dramastically as n grows compared with the inverse of polynomials.
- To verify this, we compare the logarithm of it with the logarithms of $\frac{1}{n^2}, \frac{1}{n^{10}}, \frac{1}{n^{100}}, \frac{1}{n^{1000}}, \frac{1}{n^{10000}}$:

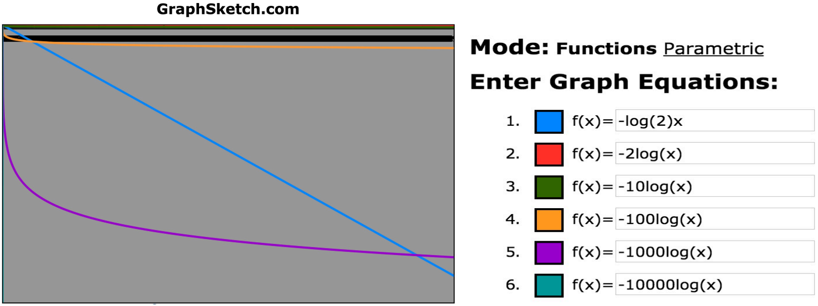


(a) x=1-100 (b) Legend

图: logarithms of functions

A example of negligible probability: $negl(n) = \frac{1}{2^n}$

- $negl(n) = \frac{1}{2^n}$ decreases much more dramastically as n grows compared with the inverse of polynomials.
- To see this, compare the logarithms of $\frac{1}{2^n}, \frac{1}{n^2}, \frac{1}{n^{10}}, \frac{1}{n^{100}}, \frac{1}{n^{1000}}, \frac{1}{n^{10000}}$:



(a) x=1-15000 (b) Legend

图: logarithms of functions

Properties of negligible probability

Proposition 1.

Let $negl_1$ and $negl_2$ be negligible functions. Then,

1. *If there exists an integer N_c such that $f(n) < negl_1(n)$ holds for all $n \geq N_c$, $f(n)$ is negligible.*
2. *The function $negl_3(n) = negl_1(n) + negl_2(n)$ is also negligible.*
3. *For any positive polynomial p, the function $negl_4$ defined by $negl_4(n) = p(n) \cdot negl_1(n)$ is negligible.*

- "Repeat-to-succeed" can be defeated.

Why is negligible success probability safe?

Two main reasons are:

- Negligible probability is very small, which means very small probability for adversaries to succeed.
- In addition, negligible success probability thwarts "repeat-to-succeed" strategy of PPT adversaries.

- Let p be the success probability of an adversary's single attack. When it attacks n times[2],

$$\begin{aligned}
& \Pr[\texttt{At least one success in } n \texttt{ attacks}] \\
= & \ 1 - \Pr[n \texttt{ straight failures}] \\
= & \ 1 - \Pr[\texttt{Fail the 1st attack} \wedge \ldots \wedge \texttt{Fail the n-th attack}] \\
= & \ 1 - (1-p)^n \\
= & \ 1 - (1 - np + \frac{n(n-1)}{2}p^2 - \frac{n(n-1)(n-2)}{6}p^3 + \ldots) \\
< & \ np
\end{aligned}$$

  Proposition 1.1 and 1.3 tell us the above probability also negligible.
- A PPT adversary can repeat for at most $p(n)$ times ($p(n)$ is an arbitrary polynomial). However, the chance is still negligible.

[2] For simplicity, here we assume attacks are independent and $np < 1$.

The Asymptotic Definition of Computational Security

定义 1.3 (The asymptotic definition of computational security).

A scheme is **secure** if for every PPT adversary $\mathcal{A}$ carrying out an attack of some formally specified type, the probability that $\mathcal{A}$ succeeds in the attack is negligible.

Equivalently,

定义 1.4 (The asymptotic definition of computational security).

A scheme is **secure** if for every PPT adversary $\mathcal{A}$ carrying out an attack of some formally specified type, and for every positive polynomial p, there exists an integer N such that when $n > N$, the probability that $\mathcal{A}$ succeeds in the attack is less than $\frac{1}{p(n)}$.

- Nothing is guaranteed for values $n \leq N$.

1. The Asymptotic Approach of Defining Computational Security

2. Computationally Secure Encryption

3. Constructing Computationally-secure Encryptions

Defining a private-key encryption scheme (with security parameter)

定义 2.1 (Private-key encryption scheme).

A **private-key encryption scheme** is a tuple of probabilistic polynomial-time algorithms (Gen, Enc, Dec) such that:

1. The key-generation algorithm Gen: $k \leftarrow Gen(1^n)$.
2. The encryption algorithm Enc: $c \leftarrow Enc_k(m)$, where the plaintext massage $m \in \{0,1\}^*$.
3. The decryption algorithm Dec: $m := Dec_k(c)$.

- If Enc is only defined for messages $m \in \{0,1\}^{l(n)}$, then we say (Gen, Enc, Dec) is a fixed-length private-key encryption for messages of length $l(n)$.
- Almost always, $Gen(1^n)$: $k \xleftarrow{\$} \{0,1\}^n$.

Motivating the security definition

Formal definition of security requires:

- Threat model: e.g. eavesdropping adversary
- Security goal: ???
  "*The adversary cannot learn any partial information about the plaintext from the ciphertext*"
  $\Rightarrow$ **semantic security** is equivalent to **indistinguishability**.

The adversarial indistinguishability experiment

To define the indistinguishability of a cipher, we first define
The adversarial indistinguishability experiment $PrivK^{eav}_{\mathcal{A},\Pi}(n)$:

1. Given input 1^n, $\mathcal{A}$ outputs a pair of message m_0, m_1 with $|m_0| = |m_1|$.
2. The challenger receives m_0, m_1 from $\mathcal{A}$, generates a random key k by running $Gen(1^n)$, generates a uniform bit b, generates the challenge ciphertext $c \leftarrow Enc_k(m_b)$ and sends c to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a bit b'.
4. $PrivK^{eav}_{\mathcal{A},\Pi}(n)$ equals 1 if $b = b'$, and 0 otherwise. If $PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1$, we say that $\mathcal{A}$ wins.

The indistinguishable encryption in the presence of an eavesdropper

定义 2.2.

A private-key encryption scheme $\Pi = (Gen, Enc, Dec)$ has **indistinguishable encryptions in the presence of an eavesdropper** or is **EAV-secure**, if for every PPT adversary $\mathcal{A}$ there is a negligible function *negl* such that for all n,

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

where the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment.

- Adversaries can only do "negligibly" better than randomly guessing.

The indistinguishable encryption in the presence of an eavesdropper

We have an equivalent definition:

定义 2.3.

A private-key encryption scheme $\Pi = (\mathit{Gen}, \mathit{Enc}, \mathit{Dec})$ has **indistinguishable encryptions in the presence of an eavesdropper** or is **EAV-secure**, if for all PPT adversaries $\mathcal{A}$ there is a negligible function *negl* such that for all n,

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

where $PrivK^{eav}_{\mathcal{A},\Pi}(n,b)$ $(b \in \{0,1\})$ denotes that a fixed bit b is used in the indistinguishability experiment, and $out_{\mathcal{A}}(PrivK^{eav}_{\mathcal{A},\Pi}(n,0))$ denotes $\mathcal{A}$'s output.

- Every adversary "behaves almost the same" with only negligible difference whether it sees an encryption of m_0 or of m_1.

Semantic Security

- Read Chapter 3.2.2 of the textbook for formal treatments of the **semantic security** if you are interested.
- It is equivalent to the indistinguishability.

Starting with the OTP

The One-Time Pad

Let $a \oplus b$ denote the bitwise exclusive-or (XOR) of two binary strings a and b, the **One-Time Pad** is as follows:

1. Fix an integer $l > 0$. $\mathcal{M} = \{0,1\}^l$, $\mathcal{K} = \{0,1\}^l$, $\mathcal{C} = \{0,1\}^l$.
2. **Gen**: $K \xleftarrow{\$} \mathcal{K}$, i.e. $Pr[K = k] = 1/2^l$ for every $k \in \mathcal{K}$.
3. $\textbf{Enc}_K(M)$: $C := M \oplus K$.

- The uniformly-random pad K guarantees perfect secrecy or complete indistinguishability, maybe we can use a "negligibley less uniformly-random" pad?
- Remember we also want to avoid "long-key" issue in OTP. So we have to use keys that are shorter than the pads.
- What we need is **pseudo-random generators (PRG)** .

定义 3.1 (Pseudo-random Generator).

Let l be a polynomial and let G be a deterministic polynomial-time algorithm such that for any n and any input $s \in \{0,1\}^n$, the output $G(s)$ is a string of length $l(n)$. We say that G is a **pseudo-random generator** if the following conditions hold:

1. **(Expansion):** For every n, $l(n) > n$.
2. **Pseudo-randomness:** For any PPT algorithm D, there is a negligible function $negl$ such that

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

   where the first probability is taken over uniform choice of $s \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choice of $r \in \{0,1\}^{l(n)}$ and the randomness of D.

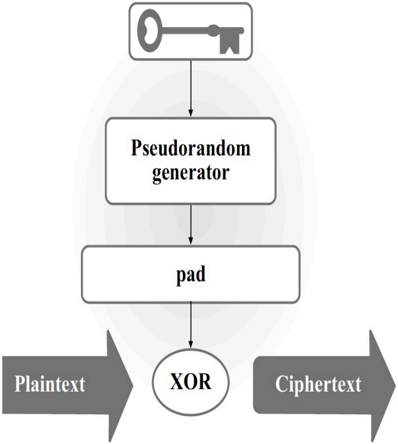
We call l the **expansion factor** of G.

Pseudo-random generators (PRG)

1. PRG is not randomized algorithm, it is deterministic.
2. Apparently, the distribution of a PRG is NOT uniformly random.
3. The brutal-force attack can differentiate PRG with truly randomness, but requires $O(2^n)$ cycles/time.

A secure fixed-length encryption scheme constructed with PRG

We construct our first secure encryption scheme with PRG:



Construction 3.1

Let G be a PRG 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 $m \in \{0,1\}^{l(n)}$, output the ciphertext
$$c := G(k) \oplus m.$$
- **Dec**: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{l(n)}$, output the message/plaintext
$$m := G(k) \oplus c.$$

A secure fixed-length encryption scheme constructed with PRG

Construction 3.1

Let G be a PRG 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 $m \in \{0,1\}^{l(n)}$, output the ciphertext
$$c := G(k) \oplus m.$$
- **Dec**: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{l(n)}$, output the message/plaintext
$$m := G(k) \oplus c.$$

Security analysis of Construction 3.1

Regarding the security of Construction 3.1, we have:

定理 3.2.

If G is a PRG, Construction 3.1 is a fixed-length private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.

- To prove the theorem, we use a paradigm called **Reduction**.

The very useful reduction paradigm

Our task: Given problem X is **hard** (i.e. cannot be solved by PPT algorithms except with negl probability), prove a system construction Π is secure.

How we prove it?

A: We prove solving problem X can be reduced to breaking Π:

- In other words, if you can efficiently break Π (with a non-negligible probability), you can also efficiently solve X (with a non-negligible probability).
- The key is to construct an efficient algorithm $\mathcal{A}'$ (we call it a "**reduction**") that solves X with the help of any solver of Π.

The very useful reduction paradigm

A proof by reduction proceeds via the following:

1. Fix some PPT algorithm $\mathcal{A}$ attacking Π. Denote its success prob by $\epsilon(n)$.
2. Construct an efficient algorithm $\mathcal{A}'$ that attempts to solve X using A as as subroutine?
   a. $\mathcal{A}$' *simulates* an instance of Π, and feeds it to $\mathcal{A}$.
   b. If $\mathcal{A}$ succeeds in breaking the instance of Π, this should allow $\mathcal{A}$' to efficiently solve the instance x it was given, at least with inverse polynomial probability $1/\mathrm{p}(\mathrm{n})$.

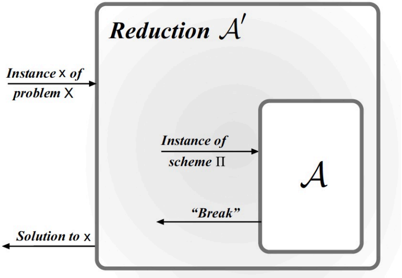


图: A high-level overview of a security proof by reduction

The very useful reduction paradigm

3. Taken together 2(a) and 2(b), $\mathcal{A}$' solves X with probability $\epsilon(n)/p(n)$. Moreover, if $\epsilon(n)$ is not negligible, then neither is $\epsilon(n)/p(n)$. In addition, if $\mathcal{A}$ is efficient, we obtain an efficient algorithm $\mathcal{A}'$ solving X with non-negligible probability, contradicting the initial condition.
4. Therefore, we can conclude no efficient adversary $\mathcal{A}$ can succeed in breaking Π with non-negligible probability.

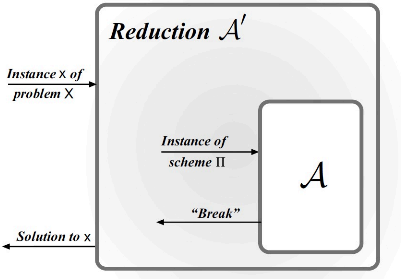


图: A high-level overview of a security proof by reduction

Security proof of Construction 3.1

定理 3.3.

If G is a PRG, Construction 3.1 is a fixed-length private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.

Proof sketch.

Let $\mathcal{A}$ be a PPT algorithm attacking Construction 3.1 Π. Let $1/2 + \epsilon(n)$ equals its success probability in a indistinguishable experiment on Π. We want to construct an adversary $\mathcal{A}'$ that attacks G, the PRG used in Construction 3.1, with the help of subroutine calls on $\mathcal{A}$.

Security proof of Construction 3.1

Proof sketch (Contd.)

- What we need to construct: A PPT pseudo-randomness/randomness distinguisher$\mathcal{A}'$ such that

$$|Pr[\mathcal{A}'(G(s)) = 1] - Pr[\mathcal{A}'(r) = 1]| = p(n) \cdot \epsilon(n). \tag{1}$$

- What we have now: A PPT ciphertext distinguisher/experiment $\mathcal{A}$ such that $|Pr[PriK^{eav}_{\mathcal{A},\Pi} = 1] - 1/2| = \epsilon(n)$, $G(s)$ and r.
- How to construct $\mathcal{A}'$ with $\mathcal{A}$?
  Simulate the distinguishable experiment with $\mathcal{A}$ with $k = G(s)$ and $k = r$ respectively, use this ciphertext distinguisher as our required pseudo-randomness/randomness distinguisher.
  If $\mathcal{A}$ wins, output 1.

Proof sketch (Contd.)

Still need to verify (1):

- When feeding $k = G(s)$,

$$Pr[\mathcal{A}'(G(s)) = 1] = Pr[\mathcal{A}\mathrm{wins} PrivK^{eav}_{\mathcal{A},\Pi}] = Pr[PrivK^{eav}_{\mathcal{A},\Pi} = 1].$$

We have already known:

$$|Pr[PriK^{eav}_{\mathcal{A},\Pi} = 1] - 1/2| = \epsilon(n). \tag{2}$$

Security proof of Construction 3.1

Proof sketch (Contd.)

- When feeding r, denote the corresponding encryption scheme by Π'

$$Pr[\mathcal{A}'(r) = 1] = Pr[\mathcal{A} wins PrivK^{eav}_{\mathcal{A},\Pi'}] = Pr[PrivK^{eav}_{\mathcal{A},\Pi'} = 1].$$

Since Π' is actually OTP (perfect secret), we know

$$Pr[PrivK^{eav}_{\mathcal{A},\Pi'} = 1] = 1/2. \tag{3}$$

Security proof of Construction 3.1

Proof sketch (Contd.)

- Based on (2) and (3), we know $|Pr[\mathcal{A}'(G(s)) = 1] - 1/2| = \epsilon(n)$ and $Pr[\mathcal{A}'(r) = 1] = 1/2$, thus

$$|Pr[\mathcal{A}'(G(s)) = 1] - Pr[\mathcal{A}'(r)]| = \epsilon(n). \quad (4)$$

If $\epsilon(n)$ is non-negligible, we find an (efficient???) algorithm $\mathcal{A}$' that distinguishes a PRG with a truely random number with non-negligible probability.
However this is impossible according to our definition about PRG, therefore we cannot find any $\mathcal{A}$ who breaks Construction 3.1 with non-negligible probability.
We still need to verify $\mathcal{A}$' is efficient, to do this we look into the details of the reduction.

Details of the reduction

Given $K = G(s)$ or r, $\mathcal{A}$' uses $PrivK^{eav}_{\mathcal{A},\Pi}/PrivK^{eav}_{\mathcal{A},\Pi'}$ as its routine, simulates $\mathcal{A}$'s challenger:

1. Given input 1^n, $\mathcal{A}$ outputs a pair of message m_0, m_1 with $|m_0| = |m_1|$, sends them to the ~~challenger $\mathcal{C}$.~~ $\mathcal{A}$'
2. ~~After receiving m_0, m_1, $\mathcal{C}$ computes $b \xleftarrow{\$} \{0,1\}$, the challenge ciphertext $c := K \oplus m_b$, and sends c to $\mathcal{A}$.~~
3. $\mathcal{A}$ outputs a bit b'.
4. ~~$PrivK^{eav}_{\mathcal{A},\Pi}(n)$ equals 1 if $b = b'$, and 0 otherwise.~~

1. ~~Given input 1^n, $\mathcal{A}'$ outputs a pair of message m_0, m_1 with $|m_0| = |m_1|$, sends them to the challenger $\mathcal{C}$.~~
2. After receiving m_0, m_1, $\mathcal{A}'$ computes $b \xleftarrow{\$} \{0,1\}$, the challenge ciphertext $c := K \oplus m_b$, and sends c to $\mathcal{A}$.
3. ~~$\mathcal{A}$ outputs a bit b'.~~
4. $\mathcal{A}$' outputs 1 if $b = b'$, and 0 otherwise.

Security proof of Construction 3.1

Efficiency of the reduction

- According to our assumption "$\mathcal{A}$ is a PPT algorithm attacking Construction 3.1", so step (1)+step(3) is efficient, i.e. polynomial-time bounded.
- step (2) requires to compute a $l(n)$-bit where l is a polynomial, so is efficient, i.e. polynomial-time bounded.
- step (4) requires to compare two bits, so is efficient.

Therefore, we know $\mathcal{A}$' is efficient.

References I

Katz, J. and Lindell, Y..
Chapter 3.1-3.3 of "Introduction to modern crytography" (2nd ed).
Chapman & Hall/CRC, 2014

The logarithm functions' graphs are generated using
https://graphsketch.com/