CS579: Foundations of Cryptography

Spring 2023

Symmetric-Key Encryption

Instructor: **Nikos Triandopoulos**

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Computational security

The big picture

- we formally defined and constructed a perfectly secure cipher
- this encryption has some drawbacks
  - e.g., it employs a very large key
- by Shannon's Theorem, such limitations are unavoidable

Now, what?

Our approach: Relax "perfectness"

Initial model / abstraction

- **perfect secrecy** / security requires that
  - absolutely **no information is leaked** about the plaintext
  - to adversaries that **unlimited computational power**

Refined model / abstraction

- consider a relaxed notion of security, called **computational security**, where an encryption scheme is (for all **practical** purposes) secure even if
  - **a tiny amount of information is leaked** about the plaintext (e.g., w/ prob. 2^{-60})
  - to adversaries with **bounded computational power** (e.g., attacker invests 200ys)

Computational security

- to be contrasted against information-theoretic security
- de facto way in which security is modeled in most cryptographic settings
- an integral part of modern cryptography w/ rigorous mathematical proofs
- entails two relaxations
  - **security is guaranteed against efficient adversaries**
    - if an attacker invests in sufficiently large resources, it may break the scheme's security
    - goal: make required resources larger than those available to any realistic attacker!
  - **adversaries can potentially succeed** (i.e., security guarantees are probabilistic)
    - with some small probability the scheme is breakable
    - goal: make success probability sufficiently small so that it can be practically ignored!

Towards a rigorous definition of computational security

- **Concrete** approach
  - bounds the maximum success probability of any (randomized) adversary running for some specified amount of time or investing a specified amount of resources
  - general security result: *"A scheme is (t,ε)-secure if any adversary $\mathcal{A}$, running for time at most t, succeeds in breaking the scheme with probability at most ε"*
  - need to
    - define what it means for an adversary to "break" a scheme
    - specify precisely the resources (e.g., time in seconds using a particular computer, or CPU cycles in a particular available supercomputer architecture)

Examples

- almost optimal security guarantees
  - key length n, key space size $|K| = 2^n$
  - $\mathcal{A}$ running for time t (e.g., CPU cycles) succeeds w/ prob. at most $ct/2^n$
  - this corresponds to a brute-force type of attack (w/out preprocessing)
- parameter c models advanced computing methods
  - c is typically larger than 1: e.g., parallelism can be used, etc.
- if c = 1, n = 60, security is enough for attackers running a desktop computer
  - 4 GHz (4×10^9 cycles/sec), 2^{60} CPU cycles require about 9 years
  - however, "fastest" available computer runs w/ 2×10^{16} cycles/sec, i.e., in ~1min!
    - choosing n=80 is better: the supercomputer would still need ~2 years

Today's recommendations

- recommended security parameter is n = 128
- large difference between 2^{80} and 2^{128}
  - #seconds since Big Bang is ~2^{58}
- if probability of success (within 1 year of computation) is $1/2^{60}$
  - it is more likely that Alice and Bob are hit by lighting (and thus don't care much about the breached confidentiality)
  - an event happening once in 100 years corresponds to probability 2^{-30} of happening at a particular second
- limitations of the concrete approach
  - harder to achieve, careful interpretation is needed, what if $\mathcal{A}$ runs for 2t or t/2?

An alternative (less quantitative) approach

- **Asymptotic** approach
  - again, a security parameter n is used (e.g., key length)
  - **efficient (or realistic or feasible) adversaries** are equated with probabilistic poly-time (PPT) algorithms that run for time that is a polynomial of n
  - **small probability of success** is equated with success probabilities that are asymptotically smaller than any inverse polynomial in n
  - general security result: *"A scheme is secure if any PPT adversary $\mathcal{A}$ succeeds in breaking the scheme with at most negligible probability"*

Negligible functions (to capture tiny likelihood)

Typically, they measure the probability of success (of an attacker)

- Intuitively: very small probability
    - negligible, can be ignored, it's more likely to be hit by asteroid…
    - approaches 0 faster than the inverse of any polynomial
    - notation: **negl**
- Formally
    - A function $\mu : N \rightarrow R^+$ is negligible if for every positive integer c there exists an integer N such that for all $n > N$, it holds that

$$\mu(n) < 1 / n^c$$

Security parameter

- Asymptotic approach
  - general result: "*A scheme is **secure** if any **PPT adversary** $\mathcal{A}$ succeeds in breaking the scheme with at most **negligible** probability*"
  - the terms "**negligible**" and "**polynomial**" make sense only if any algorithm (and the adversary $\mathcal{A}$) takes an additional input $\mathbf{1^n}$
  - this is called the **security parameter**
  - i.e., we consider an infinite sequence of schemes Π parameterized by n

Π(1), Π(2), …

- e.g., security parameter n equals the key length (i.e., $\{0,1\}^n \rightarrow k$)
  - $\mathcal{A}$ can always guess k with probability 2^{-n} – a negligible function of n
  - $\mathcal{A}$ can also enumerate all possible keys k in time 2^n – an exponential time in n

Asymptotic approach: Pros & cons

- Pros
  - all types of Turing Machines are “equivalent” up to a “polynomial reduction”
  - no need to specify the details of the computational model
  - in analyzing a scheme, the involved formulas get much simpler
- Cons
  - asymptotic results don’t tell us anything about security of the concrete systems
- In practice
  - we prove formally an asymptotic result; and then
  - argue informally that “the constants are reasonable”
    (or calculate them if it is needed)

Negligible functions

Recall: Defining security relaxations

- Concrete approach
  - general result: *"A scheme is **(t,ε)-secure** if any adversary $\mathcal{A}$, running for **time at most t**, succeeds in breaking the scheme with probability **at most ε**"*
- Asymptotic approach
  - general result: *"A scheme is **secure** if any **PPT adversary** $\mathcal{A}$ succeeds in breaking the scheme with at most **negligible** probability"*
  - PPT algorithm: probabilistic algorithm that runs in time $O(n^c)$ for some c
    - notation: **poly**

Negligible functions

Typically, they measure the probability of success (of an attacker)

- Intuitively: very small probability
  - negligible, can be ignored, it's more likely to be hit by asteroid…
  - approaches 0 faster than the inverse of any polynomial
  - notation: **negl**
- Formally
  - A function $\mu : N \rightarrow R^+$ is negligible if for every positive integer c there exists an integer N such that for all $n > N$, it holds that

$$\mu(n) < 1 / n^c$$

Example of negligible functions

f(n) =

- $1 / n^2 \rightarrow$ No
- $2^{-n} \rightarrow$ Yes
  - E.g., for $n > 23$, $f(n) < n^{-5}$
- $2^{-sqrt(n)} \rightarrow$ Yes
  - E.g., for $n > 3500$, $f(n) < n^{-5}$
- $n^{-logn} \rightarrow$ Yes
  - E.g., for $n > 33$, $f(n) < n^{-5}$
- $1 / n^{10000} \rightarrow$ No

Properties of **poly** and **negl** functions

- A sum of two polynomials is a polynomial

  poly + poly = poly

- A product of two polynomials is a polynomial:

  poly * poly = poly

- A sum of two negligible functions is a negligible function:

  negl + negl = negl

Moreover:

- A negligible function multiplied by a polynomial is negligible

  negl * poly = negl

Computational secrecy

Recall: Approach in modern cryptography

Formal treatment

- **fundamental notions** underlying the **design & evaluation** of crypto primitives

Systematic process

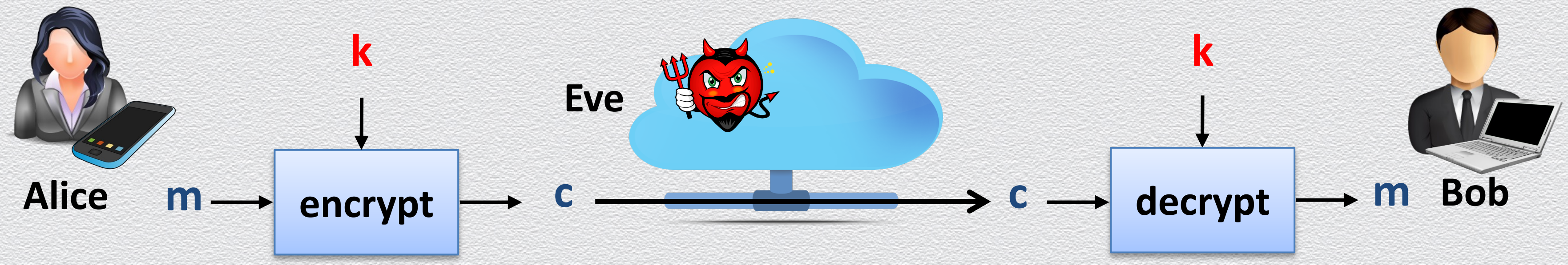
- (A) **formal definitions** (what it means for a crypto primitive to be "secure"?)
- (B) **precise assumptions** (which forms of attacks are allowed – and which aren't?)
- (C) **provable security** (why a candidate instantiation is indeed secure – or not)?

Example: Precise assumptions (1)

- **adversary**
  - type of attacks – a.k.a. **threat model** ⇨ eavesdropping
  - **capabilities** (e.g., a priori knowledge, access to information, party corruptions)
  - **limitations** (e.g., bounded memory, passive Vs. active)

⇨ Eve may know the a priori distribution of messages sent by Alice

⇨ Eve doesn't know/learn the secret k (shared by Alice and Bob)

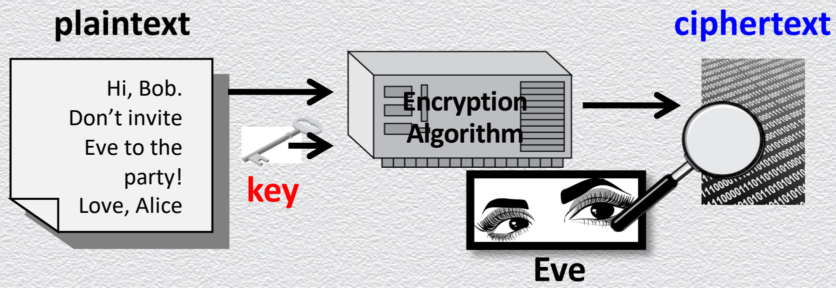


Example: Possible eavesdropping attacks (1.I)

An attacker may possess a

- (a) collection of ciphertexts
  - ciphertext only attack
  - this will be the **default attack type** when we will next define the concept of perfect security

(a)

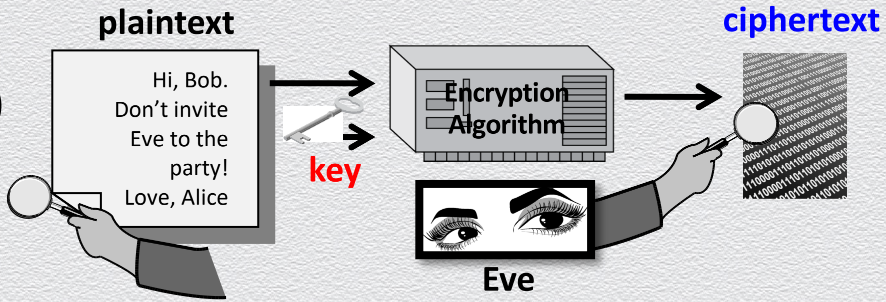


Example: Possible eavesdropping attacks (1.II)

An attacker may possess a

- (a) collection of ciphertexts
  - ciphertext only attack
- (b) collection of plaintext/ciphertext pairs
  - known plaintext attack

(b)

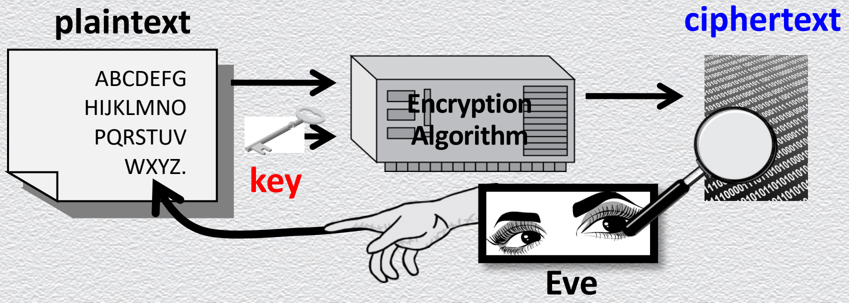


Example: Possible eavesdropping attacks (1.III)

An attacker may possess a

- (a) collection of ciphertexts
  - ciphertext only attack
- (b) collection of plaintext/ciphertext pairs
  - known plaintext attack
- (c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker
  - chosen plaintext attack

(c)

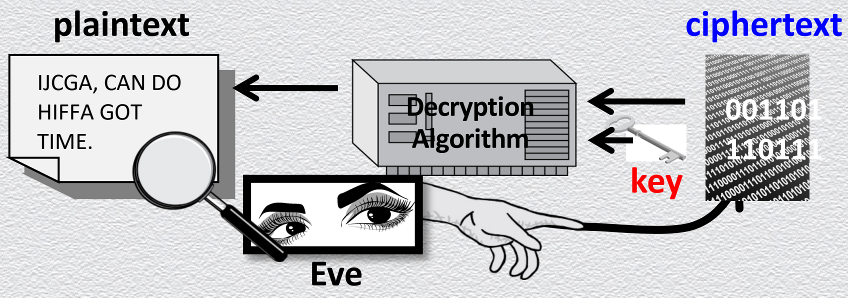


Example: Possible eavesdropping attacks (1.IV)

An attacker may possess a

- (a) collection of ciphertexts
  - ciphertext only attack
- (b) collection of plaintext/ciphertext pairs
  - known plaintext attack
- (c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker
  - chosen plaintext attack
- (d) collection of plaintext/ciphertext pairs for (plaintexts and) ciphertexts selected by the attacker
  - chosen ciphertext attack

(d)

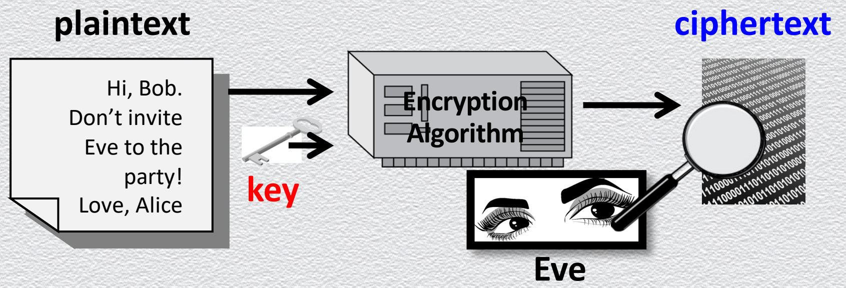


Recall: Possible eavesdropping attacks

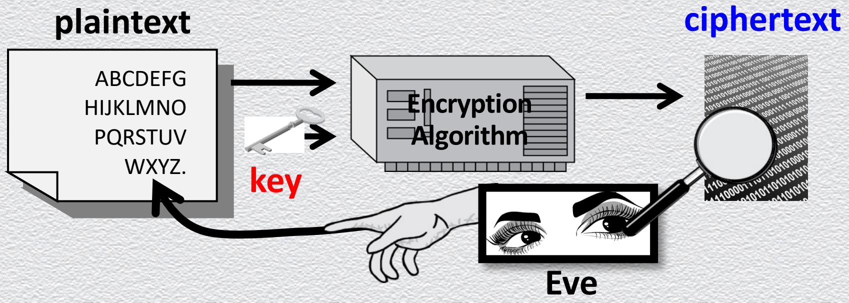
An attacker may possess a

- (a) collection of ciphertexts
  - ciphertext only attack
  - EAV-attack
- (c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker
  - chosen plaintext attack
  - CPA-attack

(a)



(c)



3 equivalent definitions of perfect EAV-security

1) a posteriori = a priori

For every $\mathcal{D}_{\mathcal{M}}$, m ∈ $\mathcal{M}$ and c ∈ $\mathcal{C}$, for which Pr [C = c] > 0, it holds that

$$\Pr[\, M = m \mid C = c \,] = \Pr[\, M = m \,]$$

2) C is independent of M

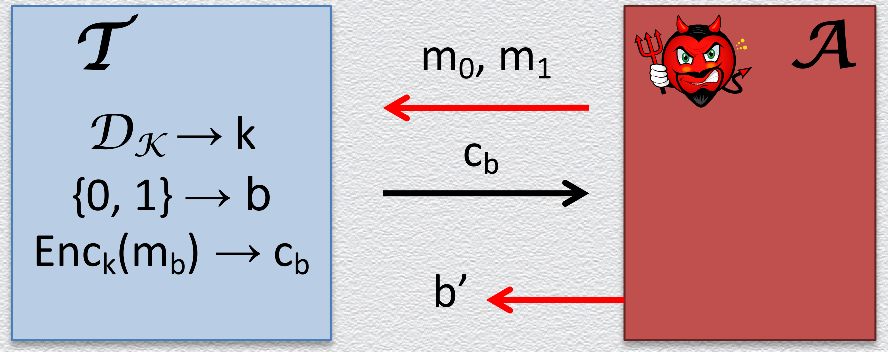
For every m, m′ ∈ $\mathcal{M}$ and c ∈ $\mathcal{C}$, it holds that

$$\Pr[\, \mathrm{Enc}_K(m) = c \,] = \Pr[\, \mathrm{Enc}_K(m') = c \,]$$

3) indistinguishability

For every $\mathcal{A}$, it holds that

$$\Pr[\, b' = b \,] = 1/2$$



From perfect to computational EAV-security

- **perfect** security: M, $Enc_K(M)$ are independent
  - absolutely **no information is leaked** about the plaintext
  - to adversaries that **unlimited computational power**
- **computational** security: for all **practical** purposes, M, $Enc_K(M)$ are independent
  - **a tiny amount of information is leaked** about the plaintext (e.g., w/ prob. 2^{-60})
  - to adversaries with **bounded computational power** (e.g., attacker invests 200ys)
- attacker's **best strategy** remains **ineffective**
  - **random guess** on secret key; or
  - **exhaustive search** over key space (**brute force attack**)

A formal, mathematic view of symmetric encryption

A symmetric-key encryption scheme is defined by

- a **message space** $\mathcal{M}$, $|\mathcal{M}| > 1$, and a triple (**Gen**, **Enc**, **Dec**)
- **Gen**: probabilistic key-generation algorithm, defines **key space** $\mathcal{K}$
  - $\text{Gen}(1^n) \rightarrow k \in \mathcal{K}$ (security parameter n)
- **Enc**: probabilistic encryption algorithm, defines **ciphertext space** $\mathcal{C}$
  - Enc: $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$, $\text{Enc}(k, m) = \text{Enc}_k(m) \rightarrow c \in \mathcal{C}$
- **Dec**: deterministic encryption algorithm
  - Dec: $\mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$, $\text{Dec}(k, c) = \text{Dec}_k(c) := m \in \mathcal{M}$ or $\perp$

Relaxing indistinguishability

Relax the definition of perfect secrecy – that is based on indistinguishability

- require that **m_0, m_1** are chosen by a **PPT adversary**
- require that no **PPT adversary** can distinguish **$Enc_k(m_0)$** from **$Enc_k(m_1)$**

non-negligibly better than guessing

PPT

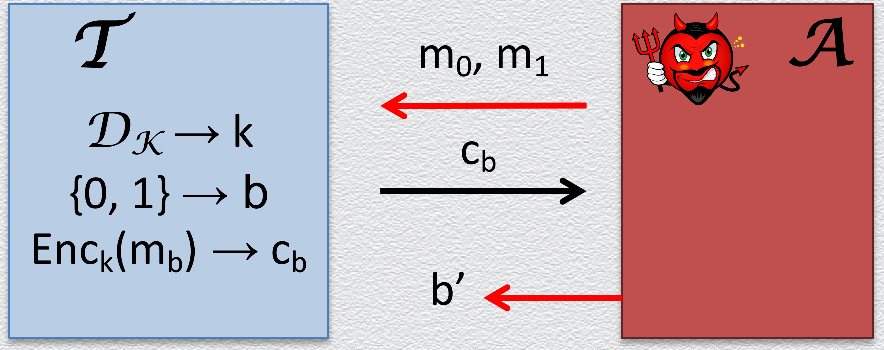
3) indistinguishability

For every $\mathcal{A}$, it holds that

Pr[b' = b] = 1/2 **+ negl**

PPT

negl



Game-based definition of computational EAV-security

encryption scheme Π = {$\mathcal{M}$, (Gen, Enc, Dec)}

Π $\mathcal{T}$

Gen(1^n) ⟶ k

{0, 1} ⟶ b

$Enc_k(m_b)$ ⟶ c_b

m_0, m_1

c_b

b'

$\mathcal{A}$

chooses $\mathbf{m_0, m_1}$

s.t. $|\mathbf{m_0}| = |\mathbf{m_1}|$

Alternatively:
"**is semantically secure**"

We say that (Enc,Dec) is **EAV-secure** if any PPT adversary $\mathcal{A}$ guesses b correctly with probability at most 0.5 + ε(n), where ε is a negligible function

I.e., no PPT $\mathcal{A}$ computes b correctly non-negligibly better than randomly guessing

Similarly: CPA-security

encryption scheme $\Pi = \{\mathcal{M}, (\text{Gen}, \text{Enc}, \text{Dec})\}$

Π $\mathcal{T}$

$\text{Gen}(1^n) \rightarrow k$

$\{0, 1\} \rightarrow b$

$\text{Enc}_k(m_b) \rightarrow c_b$

m_i

c_i

m_0, m_1

c_b

b'

$\mathcal{A}^{\text{Enc}(k,)}$

chooses $\mathbf{m_0,m_1}$

s.t. $\mathbf{|m_0|=|m_1|}$

We say that (Enc,Dec) is **CPA-secure** if any PPT adversary $\mathcal{A}$ guesses b correctly with probability at most $0.5 + \varepsilon(n)$, where ε is a negligible function

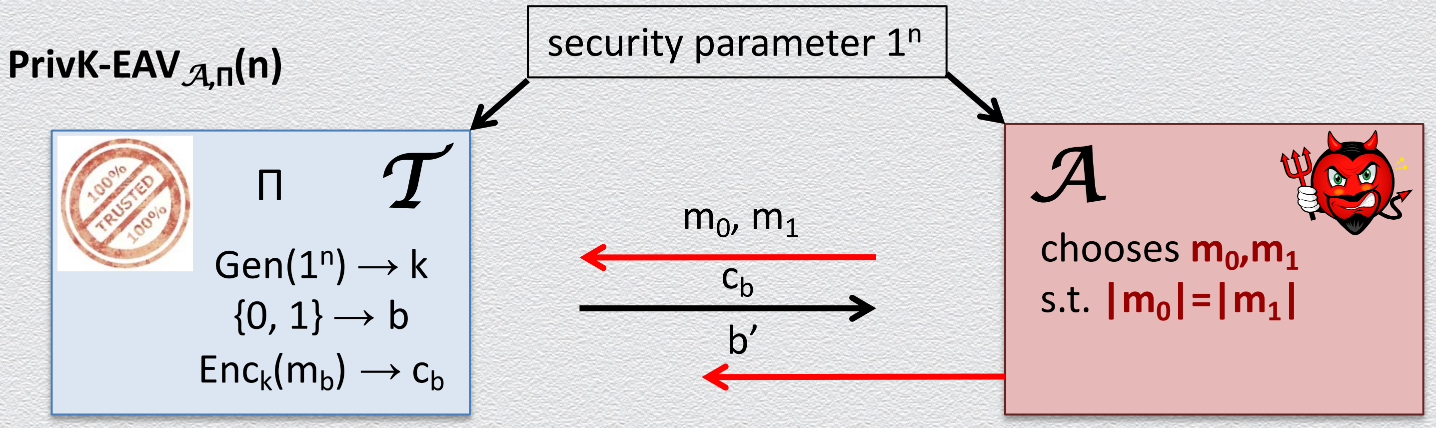
I.e., no PPT $\mathcal{A}$ computes b correctly non-negligibly better than randomly guessing, **even when it learns the encryptions of messages of its choice**

On CPA security

Facts

- Any encryption scheme that is CPA-secure is also CPA-secure for multiple encryptions
- **CPA security implies probabilistic encryption – can you see why?**
- EAV-security for multiple messages implies probabilistic encryption

Recall: EAV-security for secrecy



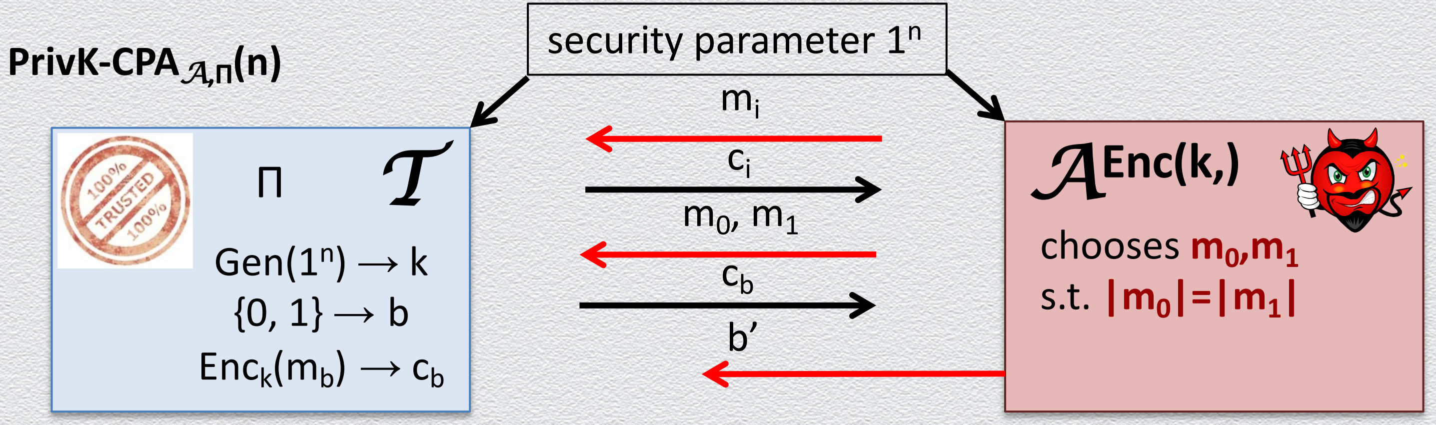
Security definition

We say that Π = (Enc,Dec) is **EAV-secure**, if ∀PPT adversaries $\mathcal{A}$, ∃a negligible function ε(n) such that it holds that

$$\Pr[\text{PrivK-EAV}_{\mathcal{A},\Pi}(n)=1] \le \tfrac{1}{2} + \varepsilon(n)$$

I.e., no PPT $\mathcal{A}$ computes b correctly non-negligibly better than randomly guessing

Recall: CPA-security for secrecy



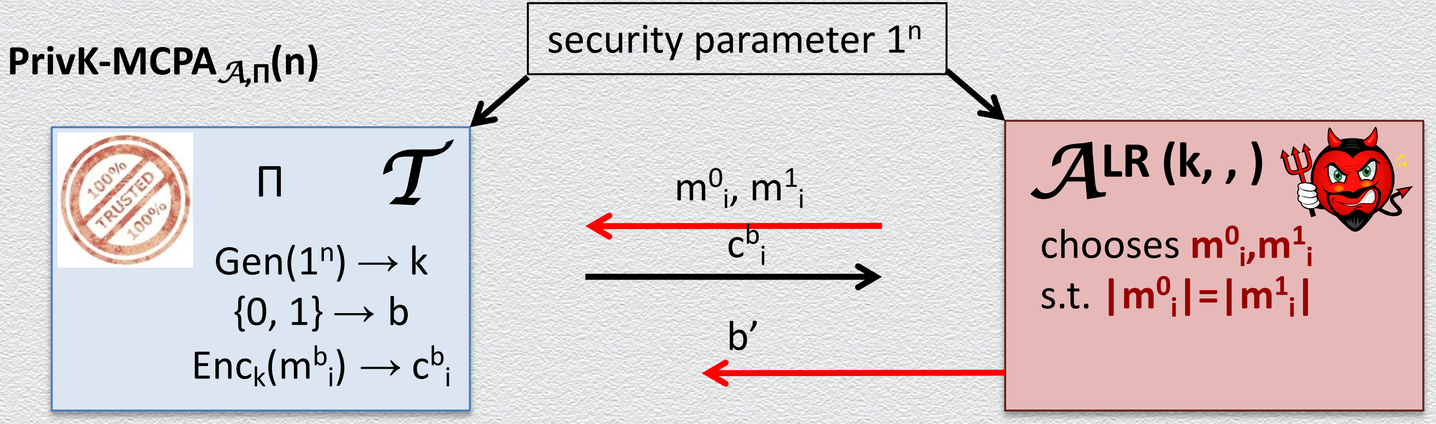
Security definition

We say that Π = (Enc,Dec) is **CPA-secure**, if ∀PPT adversaries $\mathcal{A}$, ∃a negligible function ε(n) such that it holds that

$$\Pr[\text{PrivK-CPA}_{\mathcal{A},\Pi}(n)=1] \leq \tfrac{1}{2} + \varepsilon(n)$$

I.e., no PPT $\mathcal{A}$ computes b correctly non-negligibly better than randomly guessing, **even when it learns the encryptions of messages of its choice**

CPA-security for secrecy for multiple messages



Security definition

We say that Π = (Enc,Dec) is **CPA-secure for multiple encryptions**, if ∀PPT adversaries $\mathcal{A}$, ∃a negligible function ε(n) such that it holds that

$$\Pr[\text{PrivK-MCPA}_{\mathcal{A},\Pi}(n)=1] \leq ½ + \varepsilon(n)$$

I.e., no PPT $\mathcal{A}$ computes **b for many challenge ciphertexts** correctly non-negligibly better than randomly guessing, even when it learns the encryptions of messages of its choice