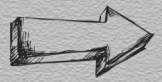
Computational security

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 $|K| = 2^n$
  - 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 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

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

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 | C = c] = Pr[M = m]

2) C is independent of M

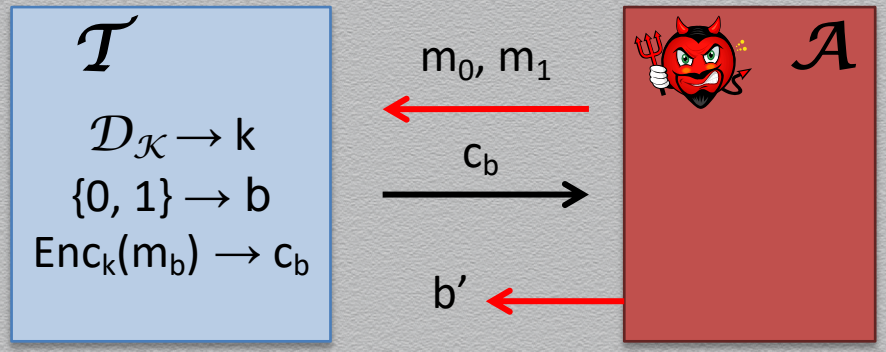
For every m, m' ∈ $\mathcal{M}$ and c ∈ $\mathcal{C}$, it holds that

Pr[Enc_K(m) = c] = Pr[Enc_K(m') = c]

3) indistinguishability

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

Pr[b' = b] = 1/2

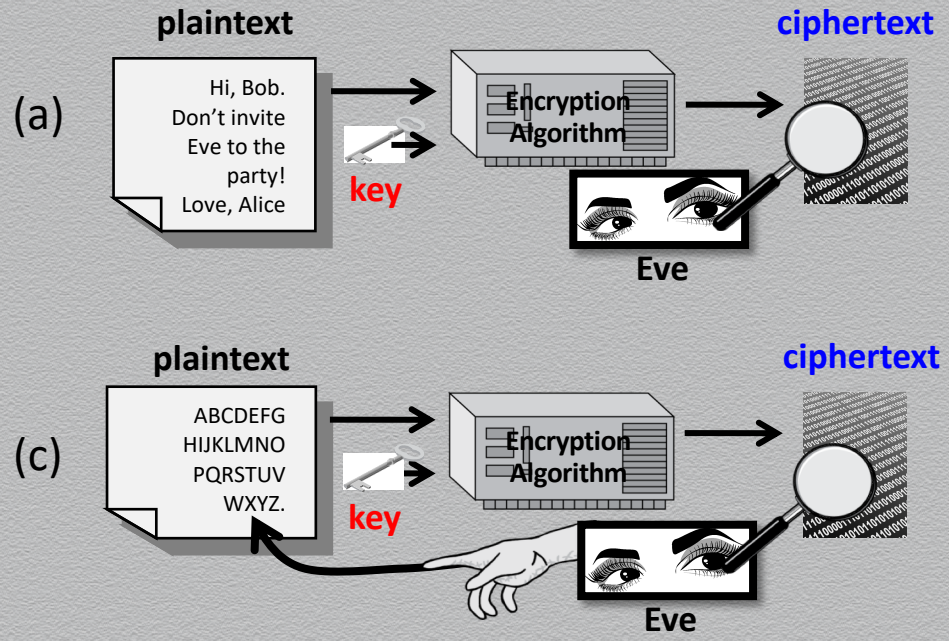


Recall: Possible eavesdropping attacks

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



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**)

Relaxing indistinguishability

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

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

non-negligibly better than guessing

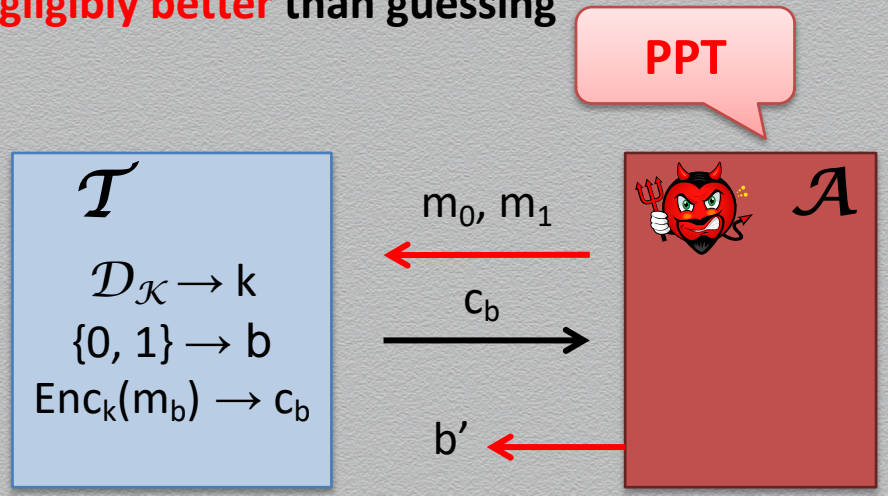
3) indistinguishability

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

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

PPT

PPT



Game-based definition of computational EAV-security

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

Π $\mathcal{T}$

$Gen(1^n) \rightarrow k$

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

$Enc_k(m_b) \rightarrow c_b$

m_0, m_1

c_b

b'

$\mathcal{A}$

chooses m_0, m_1
s.t. $|m_0| = |m_1|$

Alternatively:
"**is semantically secure**"

We say that (Enc,Dec) is **EAV-secure** if a[illegible] ad
probability at most 0.5 + ε(n), where ε is a negligible function

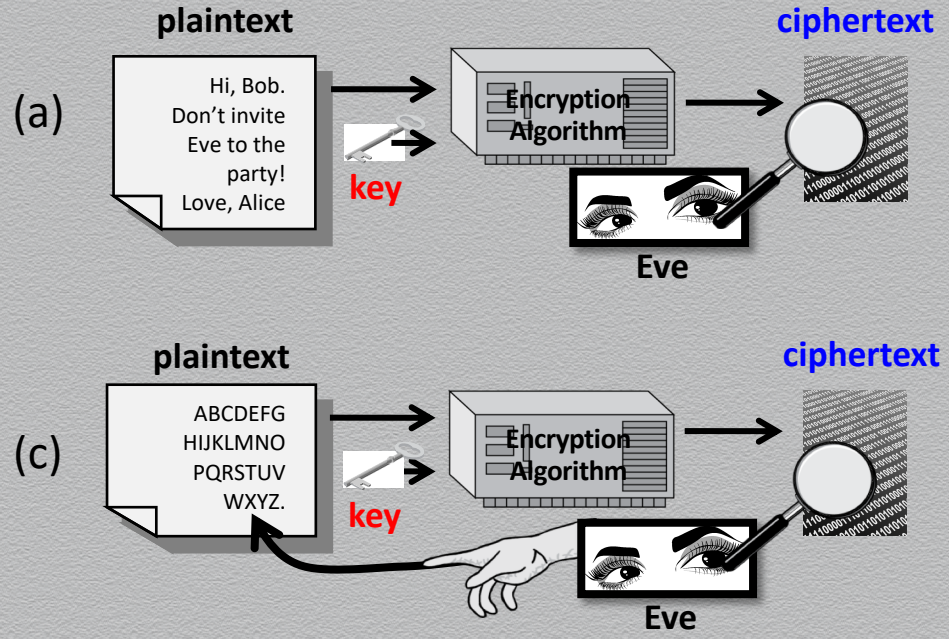
I.e., no PPT $\mathcal{A}$ computes b correctly non-negligibly better than randomly guessing

Recall: Possible eavesdropping attacks

An attacker may possess a

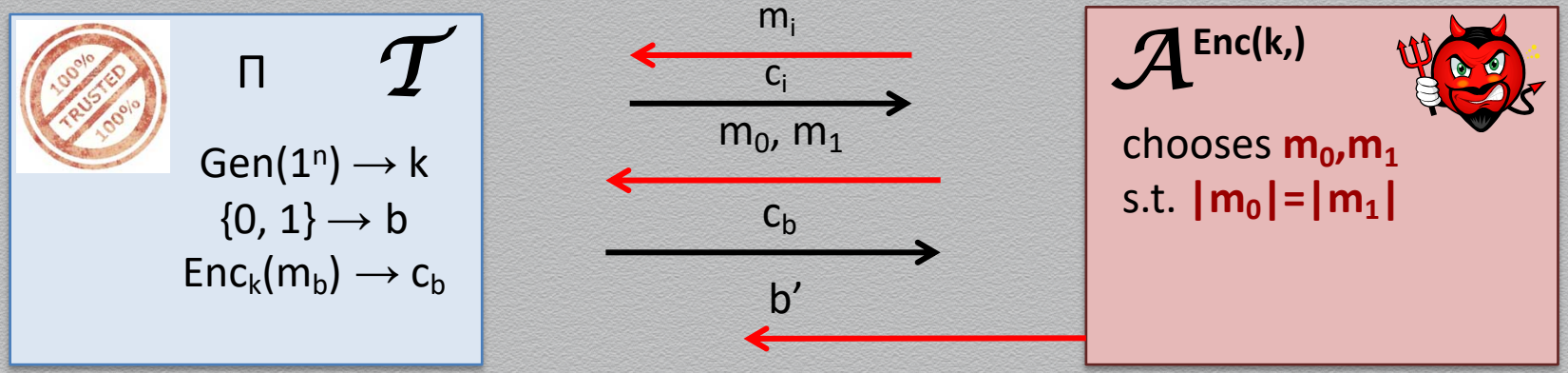
- (a) collection of ciphertexts
  - ciphertext only attack
  - EAV-attack

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



Similarly: CPA-security

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



We say that (Enc,Dec) is **CPA-secure** if any PPT adversary 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, **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