Crypto Engineering 2025–2026

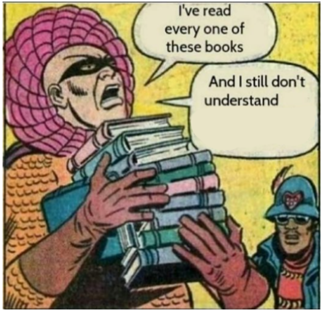
Security definitions & proof methods

Léo COLISSON PALAIS
Master CySec UGA

leo.colisson-palais@univ-grenoble-alpes.fr

`https://leo.colisson.me/teaching/`

Some references



- Framework of this course:
  The Joy of Cryptography, Mike Rosulek
  `https://joyofcryptography.com/`
- *Introduction to Modern Cryptography*, Jonathan Katz & Yehuda Lindell
- *Foundation of Cryptography*, Oded Goldreich

Symmetric cryptography

With me:

- 5 CMs, 4 TDs (3h with computers)
- Symmetric cryptography, in particular:
  - Symmetric encryption & block ciphers
  - Authentication (MAC)
  - Hash functions & specificity of password hashing

Symmetric cryptography

With me:

- 5 CMs, 4 TDs (3h with computers)
- Symmetric cryptography, in particular:
  - Symmetric encryption & block ciphers
  - Authentication (MAC)
  - Hash functions & specificity of password hashing

Goals:

- Open the boxes : **how** are the cryptographic primitive **defined**?

https://www.youtube.com/watch?v=a_HIHG5Nvpk
(slightly improved)

Goals

Goals:

- Open the boxes : **how** are the cryptographic primitive **defined**?
- Precisely specify **what “secure” means**: models, hypothesis, definitions

https://www.youtube.com/watch?v=a_HIHG5Nvpk
(slightly improved)

Goals

Goals:

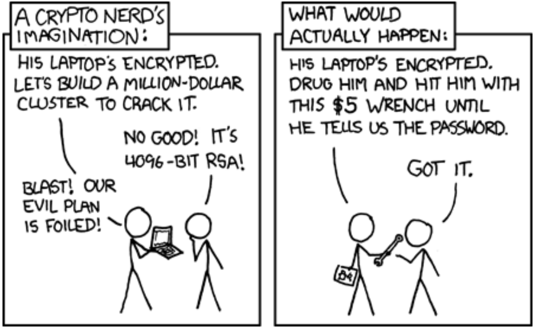
- Open the boxes : **how** are the cryptographic primitive **defined**?
- Precisely specify **what “secure” means**: models, hypothesis, definitions
- How to **formally write security proofs**?

`https://www.youtube.com/watch?v=a_HIHG5Nvpk`
(slightly improved)

Goals

Goals:

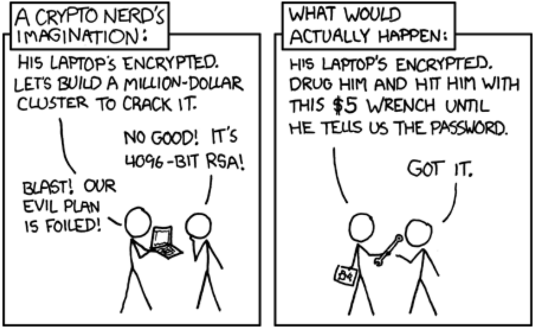
- Open the boxes : **how** are the cryptographic primitive **defined**?
- Precisely specify **what "secure" means**: models, hypothesis, definitions
- How to **formally write security proofs**?
- Understand the **implications** of these models?



Goals

Goals:

- Open the boxes : **how** are the cryptographic primitive **defined**?
- Precisely specify **what "secure" means**: models, hypothesis, definitions
- How to **formally write security proofs**?
- Understand the **implications** of these models?
- Things you should **NEVER** do!!



Associated moodle course

https://moodle.caseine.org/course/view.php?id=1342

Notations

Notation	Meaning
$x \xleftarrow{\$} X$	x is obtained by sampling an element uniformly at random from the set X
$y \leftarrow A(x)$	If A is a (probabilistic) algorithm or a distribution, we run A on input x and store the result in x
$x \stackrel{?}{=} y$	Returns 1 (true) if x equals y, 0 (false) otherwise
$\text{negl}(\lambda)$	An arbitrary function f that is negligible (= smaller than any inverse polynomial), i.e. $\forall c \in \mathbb{N}, \lim_{\lambda \to \infty} \lambda^c f(\lambda) = 0$
$\text{poly}(\lambda)$	Any function f smaller than some polynomial, i.e. $\exists c \in N, N \in N, \forall \lambda > N, f(\lambda) \leq \lambda^c$

?

Which functions are negligible?

- A $f(\lambda) = \frac{1}{2^\lambda}$
- B $f(\lambda) = \frac{1}{\lambda^{1000}}$
- C $f(\lambda) = 2^{-\log \lambda}$

Notations

Notation	Meaning
$x \xleftarrow{\$} X$	x is obtained by sampling an element uniformly at random from the set X
$y \leftarrow A(x)$	If A is a (probabilistic) algorithm or a distribution, we run A on input x and store the result in x
$x \stackrel{?}{=} y$	Returns 1 (true) if x equals y, 0 (false) otherwise
$\mathsf{negl}(\lambda)$	An arbitrary function f that is negligible (= smaller than any inverse polynomial), i.e. $\forall c \in \mathbb{N}, \lim_{\lambda \to \infty} \lambda^c f(\lambda) = 0$
$\mathsf{poly}(\lambda)$	Any function f smaller than some polynomial, i.e. $\exists c \in N, N \in N, \forall \lambda > N, f(\lambda) \leq \lambda^c$

NB: $\mathsf{negl}(\lambda) + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$, $\mathsf{negl}(\lambda) \times \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$, $\mathsf{poly}(\lambda)\mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$

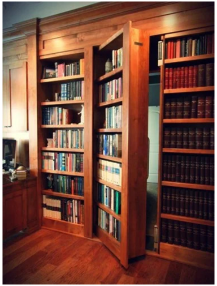
Symmetric vs asymmetric cryptography

Symmetric vs asymmetric cryptography

Symmetric encryption $\neq$ **Asymmetric encryption**

Both parties share the same secret

One party has an extra secret information (**trapdoor** that can be used to invert a function easily)



Symmetric vs asymmetric cryptography

private key

Symmetric encryption

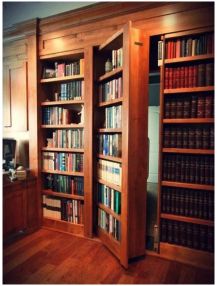
Both parties share the same secret

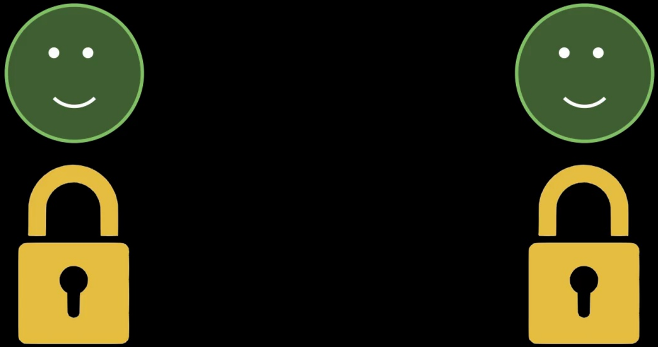
≠

public key

Asymmetric encryption

One party has an extra secret information (**trapdoor** that can be used to invert a function easily)

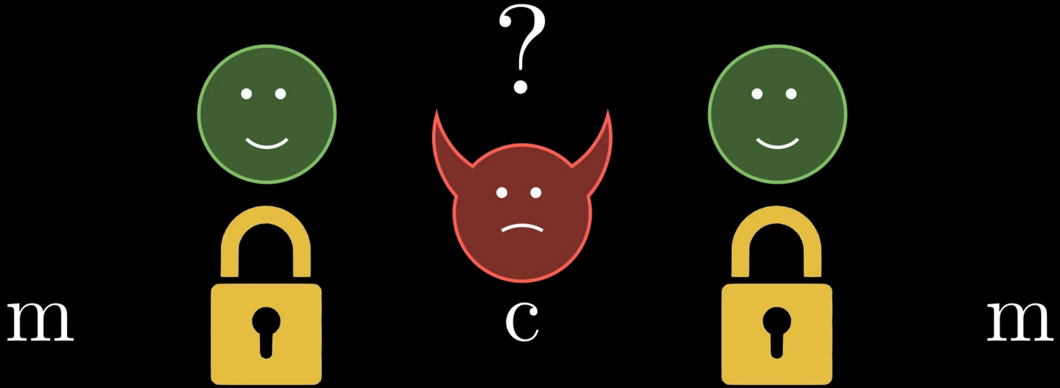



m


m
c


m
?
c


?
m
c
m

Symmetric vs asymmetric cryptography

Asymmetric encryption	**Symmetric encryption**
No need to share secrets (e.g. internet)	Need to share secrets
Stronger assumptions... factoring, LWE... (functions highly structured)	Weaker assumptions (less structure)
Less efficient	More efficient
No statistical security	Statistical security possible (but impractical)

$\Rightarrow$ Hybrid systems: **combine both** = best of both world (efficient + no secret to distribute)

Security models

When designing a crypto system, we want to say:

“The protocol XXX is **secure**

Security models

When designing a crypto system, we want to say:

"The protocol XXX is **secure** assuming YYY is hard.

Computational assumption = what is hard for the attacker
E.g. DDH, factoring, LWE...

Security models

When designing a crypto system, we want to say:

Setup assumption
(e.g. how to model hash function)

"The protocol XXX is **secure** in the plain/CRS/RO model assuming YYY is hard.

Computational assumption = what is hard for the attacker
E.g. DDH, factoring, LWE...

Security models

When designing a crypto system, we want to say:

Setup assumption
(e.g. how to model hash function)

“The protocol XXX is **secure** in the plain/CRS/RO model assuming YYY is hard.

???

Computational assumption = what is hard for the attacker
E.g. DDH, factoring, LWE…

Security models

When designing a crypto system, we want to say:

Setup assumption
(e.g. how to model hash function)

“The protocol XXX is **secure** in the plain/CRS/RO model assuming YYY is hard.

Security model = guarantees (to prove) in term of security
E.g. intuitively “the adversary is unable to find the message”

Computational assumption = what is hard for the attacker
E.g. DDH, factoring, LWE…

Hardness assumptions: Impagliazzo's worlds

Obfustopia	One can make a program code unreadable	Obfuscation
Cryptomania	Public-key cryptography and trapdoor functions exist	MPC, PKC
Minicrypt	One way functions exist, no public key cryptography	Symmetric cryptography, ZK-proofs, Secret sharing, Commitments
Pessiland	NP problems hard in practice, but no one way functions exist	
Heuristica	NP problems hard in the worst case, easy in practice (average)	
Algorithmica	P = NP (no hard problem)	One-Time Pad

No (interesting) cryptography (Pessiland, Heuristica, Algorithmica)

Hardness assumptions: Impagliazzo's worlds

World	Description	Primitives
Obfustopia	One can make a program code unreadable	Obfuscation
Cryptomania	Public-key cryptography and trapdoor functions exist	MPC, PKC
Minicrypt	One way functions exist, no public key cryptography	Symmetric cryptography, ZK-proofs, Secret sharing, Commitments
Pessiland	NP problems hard in practice, but no one way functions exist	
Heuristica	NP problems hard in the worst case, easy in practice (average)	
Algorithmica	P = NP (no hard problem)	One-Time Pad

Pessiland, Heuristica, Algorithmica: **No (interesting) cryptography**

Big question (harder than $P = NP$): in which world are we?

Hardness assumptions: Impagliazzo’s worlds

World	Description	Examples
Obfustopia	One can make a program code unreadable	Obfuscation
Cryptomania	Public-key cryptography and trapdoor functions exist	MPC, PKC
Minicrypt (Our focus)	One way functions exist, no public key cryptography	Symmetric cryptography, ZK-proofs, Secret sharing, Commitments
Pessiland	NP problems hard in practice, but no one way functions exist	
Heuristica	NP problems hard in the worst case, easy in practice (average)	
Algorithmica	P = NP (no hard problem)	One-Time Pad

Pessiland, Heuristica, Algorithmica: **No (interesting) cryptography**

Big question (harder than $P = NP$): in which world are we?

No absolute security

Since we don't know in which world we are = **no unconditional security** (except One-Time Pad) $\Rightarrow$ always rely on some **assumptions**:

"Computational" assumptions

= adversary cannot …

Harness of factoring/elliptic curves

Learning With Errors (LWE)

Code-based Crytography

Existence of one-way functions (functions hard to invert), pseudo-random permutations…

Indistinguishable Obfuscation (iO)…

Setup assumptions

= parties have access to …

Plain model

Common Reference String (CRS)

Random Oracle (RO) model

Replacing RO with hash function = heuristic (no proof that the protocol will still be secure)

Important to **clearly state them** and understand their implications!

No absolute security

Since we don't know in which world we are = **no unconditional security** (except One-Time Pad) $\Rightarrow$ always rely on some **assumptions**:

"Computational" assumptions

= adversary cannot ...

Harness of factoring/elliptic curves

(broken against quantum computers)

Learning With Errors (LWE)

Code-based Crytography

Existence of one-way functions (functions hard to invert), pseudo-random permutations...

Indistinguishable Obfuscation (iO)...

Setup assumptions

= parties have access to ...

Plain model

Common Reference String (CRS)

Random Oracle (RO) model

Replacing RO with hash function = heuristic (no proof that the protocol will still be secure)

Important to **clearly state them** and understand their implications!

Kerckhoff’s principle

The adversaries knows all details of the protocol (but cannot know directly the values sampled while running the protocol)

Security models

When designing a crypto system, we want to say:

Setup assumption
(e.g. how to model hash function)

“The protocol XXX is **secure** in the plain/CRS/RO model assuming YYY is hard.

Security model = guarantees (to prove) in term of security
E.g. intuitively “the adversary is unable to find the message”

Computational assumption = what is hard for the attacker
E.g. DDH, factoring, LWE…

Security models

Easy to intuitively say what we expect, **hard to find a good security model** that captures all possible unwanted behaviors:

E.g. for encryption:

? Attempt 1: “Given an encryption of m, an adversary should not be able to recover m”. Is this a good security definition? (if not, find a scenario where this could go wrong)

- (A) Yes
- (B) No

Security models

Easy to intuitively say what we expect, **hard to find a good security model** that captures all possible unwanted behaviors:

E.g. for encryption:

?

Attempt 1: "Given an encryption of m, an adversary should not be able to recover m". Is this a good security definition? (if not, find a scenario where this could go wrong)

- (A) Yes ✗
- (B) No ✓ Recovering 3/4 of the message is already a big issue! E.g. $m =$ `"?????????????, hence we attack tomorrow"`

Security models

Attempt 2: “Given an encryption of m, an adversary should not be able to recover any bit of m”. Is this a good security definition? (if not, find a scenario where this could go wrong)

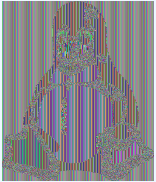
- (A) Yes
- (B) No

Security models

?

Attempt 2: “Given an encryption of m, an adversary should not be able to recover any bit of m”. Is this a good security definition? (if not, find a scenario where this could go wrong)

- (A) Yes ✗
- (B) No ✓ Knowing which groups of bits are different already leaks a lot:



AN ENCRYPTION MUST ALWAYS BE NON-DETERMINISTIC!!!

https://pieropan.ca/portfolio/a-ne-pas-faire-a-la-maison/

NEVER DO THIS

AN ENCRYPTION MUST ALWAYS BE NON-DETERMINISTIC!!!

NEVER USE A HOME-MADE ENCRYPTION, IT <u>WILL</u> BE INSECURE!!!

https://pieropan.ca/portfolio/a-ne-pas-faire-a-la-maison/

Better solution

Instead of asking for the adversary to be unable to learn XXX about m from $\text{Enc}_k(m)$…

Better solution

Instead of asking for the adversary to be unable to learn XXX about m from $\text{Enc}_k(m)\ldots$

$$\implies$$

Better to say that it is **unable to distinguish** between $\text{Enc}_k(m_0)$ and $\text{Enc}_k(m_1)$ where m_0 and m_1 are chosen by the adversary.

Better solution

Instead of asking for the adversary to be unable to learn XXX about m from $\mathsf{Enc}_k(m)\dots$

$$\Longrightarrow$$

Better to say that it is **unable to distinguish** between $\mathsf{Enc}_k(m_0)$ and $\mathsf{Enc}_k(m_1)$ where m_0 and m_1 are chosen by the adversary.

?

Does this definition implies that an adversary can't learn any information XXX about m given $\mathsf{Enc}_k(m)$?

- (A) Yes ✓ Idea: pick m_0 and m_1 with different XXX value, learn XXX from $\mathsf{Enc}_k(m_b)$, deduce b
- (B) No

How to formalize this intuition?

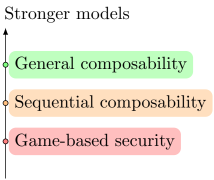
Formalism

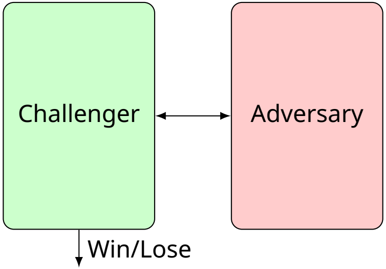
Security models

So how to define a secure protocol/encryption? $\Rightarrow$ There is not one, but **multiple** definitions of security (with different guarantees)

3 **classes** of security models:

1: Game-based security = Fix a **challenger**:





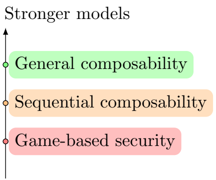
Secure if for any adversary, **the probability of winning is “low”**
(might be $1/2 + \text{negl}(\lambda)$ or $0 + \text{negl}(\lambda)$ depending on the game)

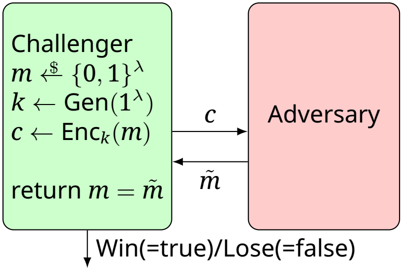
Security models

So how to define a secure protocol/encryption? ⇒ There is not one, but **multiple** definitions of security (with different guarantees)

3 **classes** of security models:

1: Game-based security = Fix a **challenger**:





Secure if for any adversary, **the probability of winning is "low"**
(might be $1/2 + \mathsf{negl}(\lambda)$ or $0 + \mathsf{negl}(\lambda)$ depending on the game)

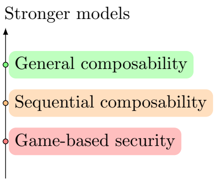
Security models

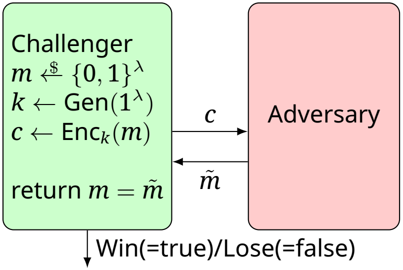
So how to define a secure pro… → There is not one, but **multiple** definitions of security (with different guarar…

Q: Is this challenger corresponding to the "don't learn m" (A) or "learn no bit about m" (B) security notion?

3 **classes** of security models:

1: Game-based security = Fix a **challenger**:



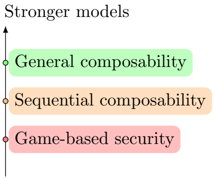


Secure if for any adversary, **the probability of winning is "low"**
(might be $1/2 + \mathsf{negl}(\lambda)$ or $0 + \mathsf{negl}(\lambda)$ depending on the game)

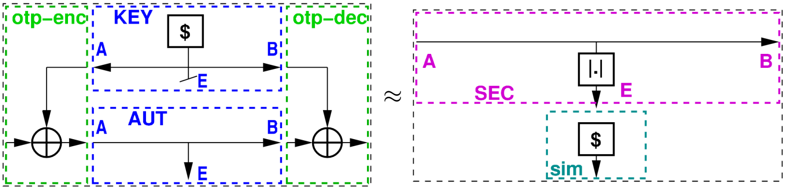
Security models

So how to define a secure protocol/encryption? ⇒ There is not one, but **multiple** definitions of security (with different guarantees)

3 **classes** of security models:



2 & 3: Composable frameworks = security based on a **simulator** that translates attacks on the real protocol to attacks on a **functionality** (trusted party) in an ideal world, supposed to be secure by definition:



Main frameworks: standalone security (sequential), Universal Composability [Can10], Abstract Crytography [MR11,M12] (general)

Security frameworks: comparison

	Game-based security	Composable/simulation-based security
Simple to understand	✓	✗
Simple to see if this is the "good" definition	✗	✓
Stronger guarantees	✗	✓
Notions natural to express	Signatures	MPC
Security guaranteed when protocols are composed	✗	✓
Impossibility results are rare	✓	✗
Example of equivalent definitions	IND-CPA	Semantic-security

IND-CPA → Semantic-security [GM84]

Security frameworks: comparison

Focus of this course → Game-based security

	Game-based security	Composable/simulation-based security
Simple to understand	✓	✗
Simple to see if this is the “good” definition	✗	✓
Stronger guarantees	✗	✓
Notions natural to express	Signatures	MPC
Security guaranteed when protocols are composed	✗	✓
Impossibility results are rare	✓	✗
Example of equivalent definitions	IND-CPA	Semantic-security

IND-CPA → Semantic-security: [GM84]

Game-based security

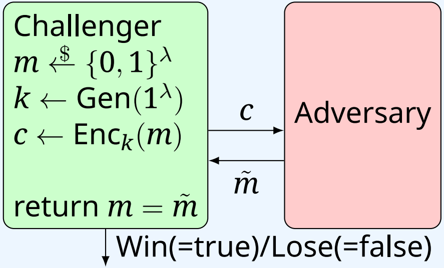
The challenger models what the adversary is allowed to do and what is considered to be "bad" in term of security:

- Which message/function can the adversary read/call?
- Passive (= eavedropper) or active adversary (= man in the middle)?
- Blackbox or with physical access to a device?
  - Side channel attacks (= record electric consumption, noise. . .)
  - Fault attacks (e.g. shooting magnetic waves to disturb a circuit. . .)
- What must be kept secret? (based on the return value of the challenger)

Questions

This models:

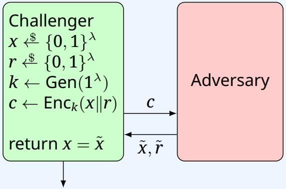
- Ⓐ a passive adversary,
- Ⓑ an active one?



Questions

Consider the following challenger, and assume that for any adversary $\mathcal{A}$, the probability of winning this game is negligible. Let $\mathcal{A}$ be an adversary, then:

- **A** The probability to **completely** recover a **random** message given its cipher is negligible
- **B** The probability to recover the **first half** of a **random** message given its cipher is negligible
- **C** The probability to recover the **first half** of **any** message given its cipher is negligible

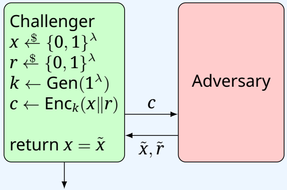


Questions

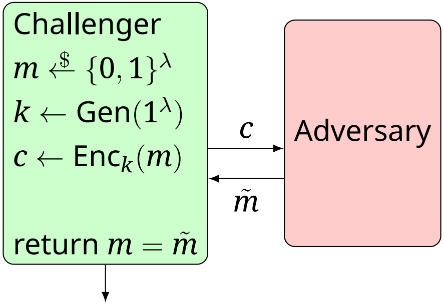
Consider the following challenger, and assume that for any adversary $\mathcal{A}$, the probability of winning this game is negligible. Let $\mathcal{A}$ be an adversary, then:

- (A) The probability to **completely** recover a **random** message given its cipher is negligible
- (B) The probability to recover the **first half** of a **random** message given its cipher is negligible ✓
- (C) The probability to recover the **first half** of **any** message given its cipher is negligible

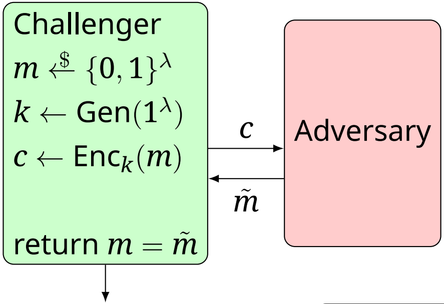
?



Equivalent notations/formulations

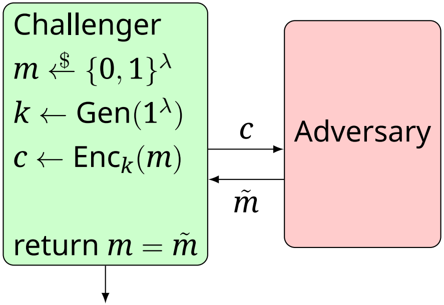


$$= \Pr_{\substack{m \xleftarrow{\$} \{0,1\}^\lambda \\ k \leftarrow \mathsf{Gen}(1^\lambda) \\ c \leftarrow \mathsf{Enc}_k(m) \\ \tilde{m} \leftarrow \mathcal{A}(c)}} [m = \tilde{m}]$$



$$= \Pr_{\substack{m \xleftarrow{\$} \{0,1\}^\lambda \\ k \leftarrow \mathsf{Gen}(1^\lambda) \\ c \leftarrow \mathsf{Enc}_k(m) \\ \tilde{m} \leftarrow \mathcal{A}(c)}} [m = \tilde{m}]$$

If $\mathcal{A}$ has oracle access to $\mathsf{Enc}_k(\cdot)$, we write $\mathcal{A}^{\mathsf{Enc}_k}$



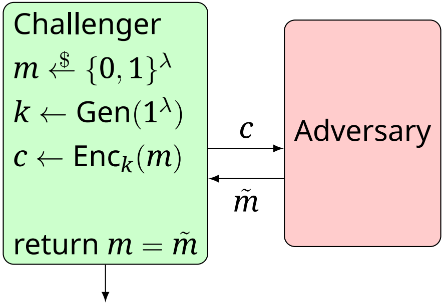
$$= \Pr_{\substack{m \xleftarrow{\$} \{0,1\}^\lambda \\ k \leftarrow \mathsf{Gen}(1^\lambda) \\ c \leftarrow \mathsf{Enc}_k(m) \\ \tilde{m} \leftarrow \mathcal{A}(c)}} [m = \tilde{m}] = \Pr_{\substack{m \xleftarrow{\$} \{0,1\}^\lambda \\ \tilde{m} \leftarrow \mathcal{A} \diamond \mathcal{L}_m}} [m = \tilde{m}]$$

$\mathcal{L}_m$

$k \leftarrow \mathsf{Gen}(1^\lambda)$
$c \leftarrow \mathsf{Enc}_k(m)$
GETC():
  return c

$\mathcal{A} \diamond \mathcal{L}$ means that $\mathcal{A}$ has oracle access to $\mathcal{L}$ (called library), like $\mathcal{A}^{\mathcal{L}}$ but this notation is used in *Joy of cryptography* and is practical when chaining multiple libraries.

Equivalent notations/formulations



$$= \Pr_{\substack{m \xleftarrow{\$} \{0,1\}^\lambda \\ k \leftarrow \mathsf{Gen}(1^\lambda) \\ c \leftarrow \mathsf{Enc}_k(m) \\ \tilde{m} \leftarrow \mathcal{A}(c)}} [m = \tilde{m}] = \Pr_{\substack{m \xleftarrow{\$} \{0,1\}^\lambda \\ \tilde{m} \leftarrow \mathcal{A} \diamond \mathcal{L}_m}} [m = \tilde{m}]$$

$\mathcal{L}_m$
$k \leftarrow \mathsf{Gen}(1^\lambda)$
$c \leftarrow \mathsf{Enc}_k(m)$
GETC():
return c

Verbose, hard to manipulate formally

More standard but often harder to manipulate and check

From *Joy of cryptography*: **easier** to re-use and write/check proofs (explicit dependency, small reductions easy to check)

But **fundamentally the same**, just different presentations!

Exercice library evaluation

We consider the following libraries:

$\mathcal{A}_1$
$r_1 \leftarrow \text{RAND}(6)$
return $r_1 \stackrel{?}{=} 4$

$\mathcal{L}_1$
$\text{RAND}(n)$:
$r \xleftarrow{\$} \mathbb{Z}_{\lceil n/2 \rceil}$
return $2r$

?

What is the value of $\Pr[\mathcal{A}_1 \diamond \mathcal{L}_1 = 1]$[a]?

- (A) 0
- (B) 1/6
- (C) 1/3
- (D) 1

[a] From now on, we define true $= 1$ and false $= 0$.

Exercice library evaluation

We consider the following libraries:

$\mathcal{A}_1$
$r_1 \leftarrow \text{RAND}(6)$
return $r_1 \stackrel{?}{=} 4$

$\mathcal{L}_1$
$\text{RAND}(n)$:
$r \stackrel{\$}{\leftarrow} \mathbb{Z}_{\lceil n/2 \rceil}$
return $2r$

?

What is the value of $\Pr[\mathcal{A}_1 \diamond \mathcal{L}_1 = 1]$[a]?

- (A) 0
- (B) 1/6
- (C) 1/3 ✓
- (D) 1

[a] From now on, we define true $= 1$ and false $= 0$.

Exercice well-defined libraries

?

We consider the following libraries: What is the value of $\Pr[\mathcal{A}_1 \diamond \mathcal{L}_1 = 1]$?

- (A) 0
- (B) 1/2
- (C) 1
- (D) This is not defined for one reason
- (E) This is not defined for two reasons

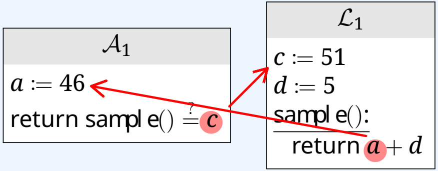
$\mathcal{A}_1$
$a := 46$
return sample() $\stackrel{?}{=} c$

$\mathcal{L}_1$
$c := 51$
$d := 5$
sample():
return $a + d$

Exercice well-defined libraries

We consider the following libraries: What is the value of $\Pr[\mathcal{A}_1 \diamond \mathcal{L}_1 = 1]$?

- (A) 0
- (B) 1/2
- (C) 1
- (D) This is not defined for one reason
- (E) This is not defined for two reasons ✓



Game-based security: power of the adversary

We can also model the power of an adversary (typically modeled as a Turing machine) in the quantification of the adversary:

- "For any **unbounded** $\mathcal{A}$, the probability of winning is low" = statistical/information theoretic security
- "For any **polynomially** bounded adversary $\mathcal{A}$, the probability of winning is low" = computational security

?

If the running time of $\mathcal{A}(n)$ is $\sqrt{n}$, is $\mathcal{A}$ polynomial?

- (A) Yes
- (B) No

Game-based security: power of the adversary

We can also model the power of an adversary (typically modeled as a Turing machine) in the quantification of the adversary:

- "For any **unbounded** $\mathcal{A}$, the probability of winning is low" = statistical/information theoretic security
- "For any **polynomially** bounded adversary $\mathcal{A}$, the probability of winning is low" = computational security

?

If the running time of $\mathcal{A}(n)$ is $\sqrt{n}$, is $\mathcal{A}$ polynomial?

(A) Yes ✗

(B) No ✓ It must run in polynomial time in the **length** ($\log(n)$) of the input (otherwise factoring is efficient!)

Game-based security: power of the adversary

We can also model the power of an adversary (typically modeled as a Turing machine) in the quantification of the adversary:

- “For any **unbounded** $\mathcal{A}$, the probability of winning is low” = statistical/information theoretic security
- “For any **polynomially** bounded adversary $\mathcal{A}$, the probability of winning is low” = computational security

?

If the running time of $\mathcal{A}(1^\lambda)$ is λ^2, is $\mathcal{A}$ polynomial?

(A) Yes

(B) No

Game-based security: power of the adversary

We can also model the power of an adversary (typically modeled as a Turing machine) in the quantification of the adversary:

- "For any **unbounded** $\mathcal{A}$, the probability of winning is low" = statistical/information theoretic security
- "For any **polynomially** bounded adversary $\mathcal{A}$, the probability of winning is low" = computational security

?

If the running time of $\mathcal{A}(1^\lambda)$ is λ^2, is $\mathcal{A}$ polynomial?

Ⓐ Yes ✓ since the argument is specified in unary

Ⓑ No

Game-based security: power of the adversary

We can also model the power of an adversary (typically modeled as a Turing machine) in the quantification of the adversary. What is low?

- "For any **unbounded** $\mathcal{A}$, the probability of winning is low" = statistical/information theoretic security
- "For any **polynomially** bounded adversary $\mathcal{A}$, the probability of winning is low" = computational security

?

If the running time of $\mathcal{A}(1^\lambda)$ is λ^2, is $\mathcal{A}$ polynomial?

(A) Yes ✓ since the argument is specified in unary

(B) No

Search vs decision

Definition of “low” = depends on the challenger, but typically we have 2 cases:

- **Search problem**: adversary needs to find a **bit-string** (e.g. “decrypt this message”): low $= \text{negl}(\lambda)$
- **Decision problem**: adversary needs to find a **single bit** b (e.g. “is this an encryption of m_0 or m_1?”): low $= 1/2 + \text{negl}(\lambda)$

Search vs decision

Definition of “low” = depends on the challenger, but typically we have 2 cases:

- **Search problem**: adversary needs to find a **bit-string** (e.g. “decrypt this message”): low $= \mathsf{negl}(\lambda)$

Focus

- **Decision problem**: adversary needs to find a **single bit** b (e.g. “is this an encryption of m_0 or m_1?”): low $= 1/2 + \mathsf{negl}(\lambda)$

Search vs decision

Definition of “low” = depends on the challenger, but typically we have 2 cases:

- **Search problem**: adversary needs to find a **bit-string** (e.g. “decrypt this message”): low $= \text{negl}(\lambda)$
- **Decision problem**: adversary needs to find a **single bit** b (e.g. “is this an encryption of m_0 or m_1?”): low $= 1/2 + \text{negl}(\lambda)$

Focus

Definition (interchangeability)

Two libraries $\mathcal{L}_0$ and $\mathcal{L}_1$ are *interchangeable* (or *equal*), written $\mathcal{L}_0 \equiv \mathcal{L}_1$, if for any adversary $\mathcal{A}$,

$$\Pr\left[\mathcal{A} \diamond \mathcal{L}_0 = 1\right] = \Pr\left[\mathcal{A} \diamond \mathcal{L}_1 = 1\right]$$

Practice time

Caseine: faire le quiz “Distinguer des librairies”

Goal

Sometimes we need a relaxed version when adversaries are computationally bounded:

Definition (advantage and indistinguishability)

We say that two libraries $\mathcal{L}_0$ and $\mathcal{L}_1$ are **indistinguishable** (denoted $\mathcal{L}_0 \approx \mathcal{L}_1$) if for any computationally bounded adversary (polynomial time) $\mathcal{A}$, **the advantage** $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ is negligible, with:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) := \left|\Pr\left[\mathcal{A}(1^\lambda) \diamond \mathcal{L}_0 = 1\right] - \Pr\left[\mathcal{A}(1^\lambda) \diamond \mathcal{L}_1 = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

Goal

Sometimes we need a relaxed version when adversaries are computationally bounded:

Definition (advantage and indistinguishability)

We say that two libraries $\mathcal{L}_0$ and $\mathcal{L}_1$ are **indistinguishable** (denoted $\mathcal{L}_0 \approx \mathcal{L}_1$) if for any computationally bounded adversary (polynomial time) $\mathcal{A}$, **the advantage** $\text{Adv}_{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ is negligible, with:

$$\text{Adv}_{\mathcal{A}}(\lambda) := \left|\Pr\left[\mathcal{A}(1^\lambda) \diamond \mathcal{L}_0 = 1\right] - \Pr\left[\mathcal{A}(1^\lambda) \diamond \mathcal{L}_1 = 1\right]\right| \leq \text{negl}(\lambda)$$

Asymptotic notion!

We finally have all the tools to define security of encryption!

Antoine Daniel will finally be able to define security of an encryption scheme

IND-CPA

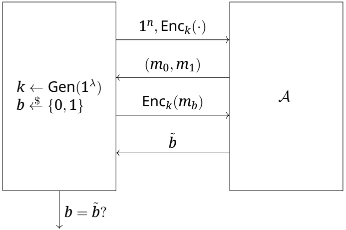
Definition (IND-CPA)

An encryption scheme $\Sigma = (\text{Gen}, \text{Enc}, \text{Dec})$ has indistinguishable security against *chosen-plaintext attacks* (IND-CPA security) if:

$$\begin{array}{|l|}\hline \mathcal{L}^{\Sigma}_{\text{cpa-L}} \\ \hline k \leftarrow \text{Gen}(1^\lambda) \\ \underline{\text{EAVESDROP}(m_L, m_R \in \mathcal{M}):} \\ \quad \text{return } \text{Enc}_k(m_L) \\ \hline \end{array} \approx \begin{array}{|l|}\hline \mathcal{L}^{\Sigma}_{\text{cpa-R}} \\ \hline k \leftarrow \text{Gen}(1^\lambda) \\ \underline{\text{EAVESDROP}(m_L, m_R \in \mathcal{M}):} \\ \quad \text{return } \text{Enc}_k(m_R) \\ \hline \end{array}$$

Various definitions of IND-CPA

You might see this other **equivalent** definition of IND-CPA:



vs

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$
$k \leftarrow \mathsf{Gen}(1^\lambda)$
EAVESDROP$(m_L, m_R \in \mathcal{M})$:
return $\mathsf{Enc}_k(m_L)$

$\approx$

$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$
$k \leftarrow \mathsf{Gen}(1^\lambda)$
EAVESDROP$(m_L, m_R \in \mathcal{M})$:
return $\mathsf{Enc}_k(m_R)$

- Instead of b, when $b = 0$ we play $\mathcal{L}^{\Sigma}_{\text{cpa-L}}$ otherwise $\mathcal{L}^{\Sigma}_{\text{cpa-R}}$.
- In our definition, no access to oracle $\mathsf{Enc}_k(\cdot)$, but we can **simulate it** by calling EAVESDROP(m, m) (same message twice).
- In our definition, no restriction on the number of allowed calls to EAVESDROP (= stronger notion, while in the other we have a single message $Enc_k(m_b)$). But equivalent (advantage is multiplied by the maximum number of queries done by $\mathcal{A}$, but still negligible): proof via a sequence of **hybrids on the number of queries**.

How to prove INsecurity?

How to prove INsecurity

To prove **in**security for a decision game between $\mathcal{L}_0$ and $\mathcal{L}_1$:

1. exhibits a given attacker $\mathcal{A}$
2. compute $\varepsilon = |\Pr[\mathcal{A} \diamond \mathcal{L}_0 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_1 = 1]|$
3. show that $\exists c \in \mathbb{N}$ s.t. ε is greater than $\frac{1}{\lambda^c}$

How to prove INsecurity

?

We consider the encryption scheme $\mathsf{Gen}(1^\lambda) := \textbf{return } 0$ and $\mathsf{Enc}_k(m) := m \oplus 1 \dots 1$. Is this scheme IND-CPA secure, and if not, which attacker can distinguish these two libraries, and with which advantage ?

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$
$k \leftarrow \mathsf{Gen}(1^\lambda)$
$\text{EAVESDROP}(m_L, m_R \in \mathcal{M})$:
return $\mathsf{Enc}_k(m_L)$

$\overset{?}{\approx}$

$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$
$k \leftarrow \mathsf{Gen}(1^\lambda)$
$\text{EAVESDROP}(m_L, m_R \in \mathcal{M})$:
return $\mathsf{Enc}_k(m_R)$

(1)

How to prove INsecurity

?

We consider the encryption scheme $\text{Gen}(1^\lambda) := \textbf{return } 0$ and $\text{Enc}_k(m) := m \oplus 1\ldots1$. Is this scheme IND-CPA secure, and if not, which attacker can distinguish these two libraries, and with which advantage ?

$$\begin{array}{|l|}\hline \mathcal{L}^{\Sigma}_{\text{cpa-L}} \\ \hline k \leftarrow \text{Gen}(1^\lambda) \\ \text{EAVESDROP}(m_L, m_R \in \mathcal{M}): \\ \quad \text{return Enc}_k(m_L) \\ \hline \end{array} \overset{?}{\approx} \begin{array}{|l|}\hline \mathcal{L}^{\Sigma}_{\text{cpa-R}} \\ \hline k \leftarrow \text{Gen}(1^\lambda) \\ \text{EAVESDROP}(m_L, m_R \in \mathcal{M}): \\ \quad \text{return Enc}_k(m_R) \\ \hline \end{array} \tag{1}$$

1. $\begin{array}{|l|}\hline \mathcal{A} \\ \hline c := \text{EAVESDROP}(0^\lambda) \\ \text{return } c \oplus 1\ldots1 \overset{?}{=} 0^\lambda \\ \hline \end{array}$, advantage 0 (A), $1/2$ (B), $1/2 - \frac{1}{2^\lambda}$ (C) or 1 (D)

2. $\begin{array}{|l|}\hline \mathcal{A} \\ \hline c := \text{EAVESDROP}(0^\lambda) \\ \text{return } c \oplus c \overset{?}{=} 0^\lambda \\ \hline \end{array}$, advantage 0 (E), $1/2$ (F), $1/2 - \frac{1}{2^\lambda}$ (G) or $1 - \frac{1}{2^\lambda}$ (H)

How to prove INsecurity

We consider the encryption scheme $\text{Gen}(1^\lambda) := \textbf{return } 0$ and $\text{Enc}_k(m) := m \oplus 1 \ldots 1$. Is this scheme IND-CPA secure, and if not, which attacker can distinguish these two libraries, and with which advantage ?

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$
$k \leftarrow \text{Gen}(1^\lambda)$
$\text{EAVESDROP}(m_L, m_R \in \mathcal{M})$:
return $\text{Enc}_k(m_L)$

$\overset{?}{\approx}$ (1)

$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$
$k \leftarrow \text{Gen}(1^\lambda)$
$\text{EAVESDROP}(m_L, m_R \in \mathcal{M})$:
return $\text{Enc}_k(m_R)$

?

1.

$\mathcal{A}$
$c := \text{EAVESDROP}(0^\lambda)$
return $c \oplus 1 \ldots 1 \overset{?}{=} 0^\lambda$

, advantage 0 (A), $1/2$ (B), $1/2 - \frac{1}{2^\lambda}$ (C) or 1 (D ✓)

2.

$\mathcal{A}$
$c := \text{EAVESDROP}(0^\lambda)$
return $c \oplus c \overset{?}{=} 0^\lambda$

, advantage 0 (E), $1/2$ (F), $1/2 - \frac{1}{2^\lambda}$ (G) or $1 - \frac{1}{2^\lambda}$ (H)

How to prove INsecurity

Which attacker can distinguish these two libraries, and with which advantage?

$\mathcal{L}^{\Sigma}_{\text{ots\$-real}}$

$\textsc{ctxt}(m \in \{0,1\}^\lambda)$:

$k \leftarrow \{0,1\}^\lambda$ // Σ.KeyGen

$c := k \,\&\, m$ // Σ.Enc

return c

$\mathcal{L}^{\Sigma}_{\text{ots\$-rand}}$

$\textsc{ctxt}(m \in \{0,1\}^\lambda)$:

$c \leftarrow \{0,1\}^\lambda$ // $\Sigma.\mathcal{C}$

return c

?

How to prove INsecurity

?

Which attacker can distinguish these two libraries, and with which advantage?

$\mathcal{L}^{\Sigma}_{\text{ots\$-real}}$

$\text{CTXT}(m \in \{0,1\}^\lambda)$:
- $k \leftarrow \{0,1\}^\lambda$ // Σ.KeyGen
- $c := k \,\&\, m$ // Σ.Enc
- return c

$\mathcal{L}^{\Sigma}_{\text{ots\$-rand}}$

$\text{CTXT}(m \in \{0,1\}^\lambda)$:
- $c \leftarrow \{0,1\}^\lambda$ // $\Sigma.C$
- return c

1. $\mathcal{A}$: $c := \text{CTXT}(0^\lambda)$; return $c = 0^\lambda$ — advantage $1/4$ (A), $1/2$ (B), $1/2 - \frac{1}{2^\lambda}$ (C) or $1 - \frac{1}{2^\lambda}$ (D)
2. $\mathcal{A}$: $c := \text{CTXT}(1^\lambda)$; return $c = 0^\lambda$ — advantage $1/4$ (E), $1/2$ (F), $1/2 - \frac{1}{2^\lambda}$ (G) or $1 - \frac{1}{2^\lambda}$ (H)

How to prove INsecurity

Which attacker can distinguish these two libraries, and with which advantage?

$\mathcal{L}^{\Sigma}_{\text{ots\$-real}}$

$\textsc{ctxt}(m \in \{\texttt{0},\texttt{1}\}^\lambda)$:
- $k \leftarrow \{\texttt{0},\texttt{1}\}^\lambda$ // Σ.KeyGen
- $c := k \,\&\, m$ // Σ.Enc
- return c

$\mathcal{L}^{\Sigma}_{\text{ots\$-rand}}$

$\textsc{ctxt}(m \in \{\texttt{0},\texttt{1}\}^\lambda)$:
- $c \leftarrow \{\texttt{0},\texttt{1}\}^\lambda$ // $\Sigma.C$
- return c

?

1. $\mathcal{A}$: $c := \textsc{ctxt}(\texttt{0}^\lambda)$; return $c = \texttt{0}^\lambda$, advantage $1/4$ (A), $1/2$ (B), $1/2 - \frac{1}{2^\lambda}$ (C) or $1 - \frac{1}{2^\lambda}$ (D ✓)

2. $\mathcal{A}$: $c := \textsc{ctxt}(\texttt{1}^\lambda)$; return $c = \texttt{0}^\lambda$, advantage $1/4$ (E), $1/2$ (F), $1/2 - \frac{1}{2^\lambda}$ (G) or $1 - \frac{1}{2^\lambda}$ (H)

Practice

See previous exercise in Caseine for more examples

Concrete vs asymptotic cryptography

Asymptotic vs actual security

In theoretical analysis, security is asymptotic. In practice: **How to choose λ ?**
Typically:

- (A) Study the best known attacks, **count the number of operations T** and the advantage ε (trade-off time/precision), consider that the actual number of operations is roughly[1] T/ε.
  $\Rightarrow$ this protocol has $\log(T/\varepsilon)$-bits of security.
- (B) Realize that:
  - 2^{40} operations is really easy to do (small raspberry pi cluster)
  - 2^{60} operations doable with large CPU/GPU cluster
  - 2^{80} operations doable with an ASIC cluster (bitcoin mining)
  - 2^{128} operations = **very hard** (next slide)

[1] More details in [Watanabe, Yasunaga 2021] and [Micciancio, Walter 2018].

How big is 2^{128}?

Say that:

- problem is parallelizable
- you can access all 500 best super-computers $=$ 10 000 000 000 GFLOPS (FLOPS = floating point operations per second)

Then, you need in total:

$$\frac{2^{128}}{10 \times 10^9 \times 10^9 \times 3600 \times 24 \times 365} \approx \boxed{1\ 000\ 000\ 000\ 000 \text{ years}}$$

(roughly $4\times$ age of earth)

How to write security proofs

Basic properties

Properties (also hold when replacing $\approx$ with $\equiv$)

- **Transitivity**: $(\mathcal{L}_0 \approx \mathcal{L}_1) \wedge (\mathcal{L}_1 \approx \mathcal{L}_2) \Rightarrow \mathcal{L}_0 \approx \mathcal{L}_2$
- **Chaining**: $(\mathcal{L}_0 \approx \mathcal{L}_1) \Rightarrow ((\mathcal{L} \diamond \mathcal{L}_0) \approx (\mathcal{L} \diamond \mathcal{L}_1))$

Preuves: exercice

Basic properties

Properties (also hold when replacing $\approx$ with $\equiv$)

- **Transitivity**: $(\mathcal{L}_0 \approx \mathcal{L}_1) \land (\mathcal{L}_1 \approx \mathcal{L}_2) \Rightarrow \mathcal{L}_0 \approx \mathcal{L}_2$
- **Chaining**: $(\mathcal{L}_0 \approx \mathcal{L}_1) \Rightarrow ((\mathcal{L} \diamond \mathcal{L}_0) \approx (\mathcal{L} \diamond \mathcal{L}_1))$

Proof transitivity (basically triangle inequality): We assume $\mathcal{L}_0 \approx \mathcal{L}_1 \land \mathcal{L}_1 \approx \mathcal{L}_2$. Let $\mathcal{A}$ run in polynomial time. Then by definition:

$$|\Pr[\mathcal{A} \diamond \mathcal{L}_0 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_1 = 1]| \leq \mathsf{negl}(\lambda) \land |\Pr[\mathcal{A} \diamond \mathcal{L}_1 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_2 = 1]| \leq \mathsf{negl}(\lambda)$$

But

$$\begin{aligned}
&|\Pr[\mathcal{A} \diamond \mathcal{L}_0 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_2 = 1]| \\
&= |\Pr[\mathcal{A} \diamond \mathcal{L}_0 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_1 = 1] + \Pr[\mathcal{A} \diamond \mathcal{L}_1 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_2 = 1]| \\
&\leq |\Pr[\mathcal{A} \diamond \mathcal{L}_0 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_1 = 1]| + |\Pr[\mathcal{A} \diamond \mathcal{L}_1 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_2 = 1]| \\
&\leq \mathsf{negl}(\lambda) + \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda)
\end{aligned}$$

Basic properties

Properties (also hold when replacing $\approx$ with $\equiv$)

- **Transitivity**: $(\mathcal{L}_0 \approx \mathcal{L}_1) \wedge (\mathcal{L}_1 \approx \mathcal{L}_2) \Rightarrow \mathcal{L}_0 \approx \mathcal{L}_2$
- **Chaining**: $(\mathcal{L}_0 \approx \mathcal{L}_1) \Rightarrow ((\mathcal{L} \diamond \mathcal{L}_0) \approx (\mathcal{L} \diamond \mathcal{L}_1))$

Proof chaining: We assume that $\mathcal{L}_0 \approx \mathcal{L}_1$. Let $\mathcal{A}$ run in poly time. We want to show $(\mathcal{L} \diamond \mathcal{L}_0) \approx (\mathcal{L} \diamond \mathcal{L}_1)$:

$$\begin{aligned} & |\Pr[\mathcal{A} \diamond (\mathcal{L} \diamond \mathcal{L}_0) = 1] - \Pr[\mathcal{A} \diamond (\mathcal{L} \diamond \mathcal{L}_2) = 1]| \\ &= |\Pr[(\mathcal{A} \diamond \mathcal{L}) \diamond \mathcal{L}_0 = 1] - \Pr[(\mathcal{A} \diamond \mathcal{L}) \diamond \mathcal{L}_1 = 1]| \\ &\overset{\mathcal{A}' := \mathcal{A} \diamond \mathcal{L}}{=} |\Pr[\mathcal{A}' \diamond \mathcal{L}_0 = 1] - \Pr[\mathcal{A}' \diamond \mathcal{L}_1 = 1]| \end{aligned}$$

since $\mathcal{A}$ runs in poly time, so does $\mathcal{A}'$. Hence using $\mathcal{L}_0 \approx \mathcal{L}_1$ the above is $\mathsf{negl}(\lambda)$. $\square$

Methods

Six main methods:

1. **Hybrid games** : Decompose into a sequence of hybrid games (to make methods 2 – 6 easier)
2. **Probabilities** : Explicitly compute the probability, and show equality or bound the statistical distance (statistical security only)
3. **Equality** : Show that the two games are trivially doing exactly the same thing (variant of 2)
   (e.g. code simply externalized to a sub-library, code that is simply inlined. . .)
4. **Reduction** : show that if we can distinguish them, then $\mathcal{A}$ can be used to break a hard problem (factor numbers. . .)
5. **Theorem/assumption** : use a theorem already seen in the course or an assumption
6. **Chaining** : prove $\mathcal{L}_1 \approx \mathcal{L}_2$, then $\mathcal{A} \diamond \mathcal{L}_1 \approx \mathcal{A} \diamond \mathcal{L}_2$

We detail methods 1,2,3,4 now (5 & 6 trivial).

Hybrid games

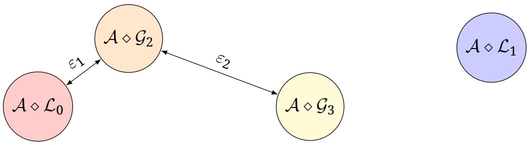
Proof = sequence of **hybrid** games:

Hybrid games

Proof = sequence of **hybrid** games:

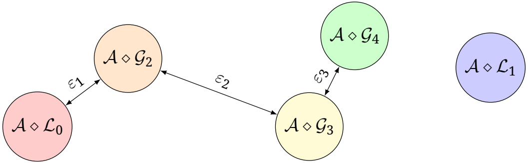
Hybrid games

Proof = sequence of **hybrid** games:



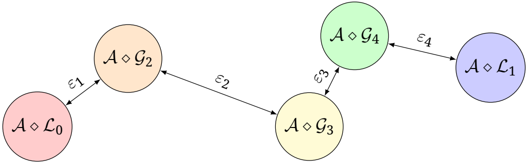
Hybrid games

Proof = sequence of **hybrid** games:



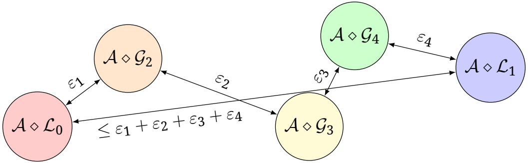
Hybrid games

Proof = sequence of **hybrid** games:



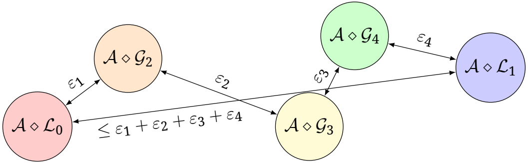
Hybrid games

Proof = sequence of **hybrid** games:



Hybrid games

Proof = sequence of **hybrid** games:



By transitivity, if $\mathcal{L}_0 \approx \mathcal{G}_2 \approx \mathcal{G}_3 \approx \mathcal{G}_4 \approx \mathcal{L}_1$, then $\mathcal{L}_0 \approx \mathcal{L}_1$.

Equality

Just realize two libraries are trivially **doing the exact same thing** (e.g. move a call in a sub-library or inline a sub-library in a code)
WARNING: Make sure variables are always well defined, with no naming collision and well **scoped** (a sub-library cannot refer to a variable of a parent library)

Equality

?

Are these two libraries equal?

$$\boxed{\begin{array}{l}\underline{\textsc{ctxt}(m):}\\ \quad k_1 \leftarrow \{0,1\}^\lambda \\ \quad c_1 := k_1 \oplus m \\ \quad c_2 := \textsc{ctxt}'(c_1) \\ \quad \text{return } c_2\end{array}} \diamond \boxed{\begin{array}{l}\mathcal{L}_{\text{otp-rand}} \\ \hline \underline{\textsc{ctxt}'(m'):}\\ \quad c \leftarrow \{0,1\}^\lambda \\ \quad \text{return } c\end{array}} \equiv \boxed{\begin{array}{l}\underline{\textsc{ctxt}(m):}\\ \quad k_1 \leftarrow \{0,1\}^\lambda \\ \quad c_1 := k_1 \oplus m \\ \quad c_2 \leftarrow \{0,1\}^\lambda \\ \quad \text{return } c_2\end{array}}$$

Ⓐ Yes

Ⓑ No

?

Are these two libraries equal?

$$
\begin{array}{|l|}\hline
\underline{\textsc{ctxt}(m):} \\
\quad k_1 \leftarrow \{\texttt{0},\texttt{1}\}^\lambda \\
\quad c_1 := k_1 \oplus m \\
\quad c_2 := \textsc{ctxt}'(c_1) \\
\quad \text{return } c_2 \\ \hline
\end{array}
\diamond
\begin{array}{|l|}\hline
\mathcal{L}_{\text{otp-rand}} \\ \hline
\underline{\textsc{ctxt}'(m'):} \\
\quad c \leftarrow \{\texttt{0},\texttt{1}\}^\lambda \\
\quad \text{return } c \\ \hline
\end{array}
\equiv
\begin{array}{|l|}\hline
\underline{\textsc{ctxt}(m):} \\
\quad k_1 \leftarrow \{\texttt{0},\texttt{1}\}^\lambda \\
\quad c_1 := k_1 \oplus m \\
\quad c_2 \leftarrow \{\texttt{0},\texttt{1}\}^\lambda \\
\quad \text{return } c_2 \\ \hline
\end{array}
$$

- (A) Yes ✓ Variable are well scoped, inlined a sub-library
- (B) No ✗

Equality

Are these two libraries equal?

?

$$\begin{array}{|l|}\hline \mathcal{L}_0 \\ \hline k \leftarrow \textsf{Gen}(1^\lambda) \\ c \leftarrow \textsf{Enc}_k(m) \\ \underline{\textsf{GET}():} \\ \quad \text{return } c \\ \hline \end{array} = \begin{array}{|l|}\hline \mathcal{L}_1 \\ \hline k \leftarrow \textsf{Gen}(1^\lambda) \\ \underline{\textsf{GET}():} \\ \quad \text{return } \textsf{MYGET}() \\ \hline \end{array} \diamond \begin{array}{|l|}\hline \mathcal{L}_2 \\ \hline c \leftarrow \textsf{Enc}_k(m) \\ \underline{\textsf{MYGET}():} \\ \quad \text{return } c \\ \hline \end{array}$$

- (A) Yes
- (B) No

Equality

Are these two libraries equal?

$$\begin{array}{|l|}\hline \mathcal{L}_0 \\ \hline k \leftarrow \text{Gen}(1^\lambda) \\ c \leftarrow \text{Enc}_k(m) \\ \underline{\text{GET():}} \\ \quad \text{return } c \\ \hline \end{array} = \begin{array}{|l|}\hline \mathcal{L}_1 \\ \hline k \leftarrow \text{Gen}(1^\lambda) \\ \underline{\text{GET():}} \\ \quad \text{return MYGET()} \\ \hline \end{array} \diamond \begin{array}{|l|}\hline \mathcal{L}_2 \\ \hline c \leftarrow \text{Enc}_k(m) \\ \underline{\text{MYGET():}} \\ \quad \text{return } c \\ \hline \end{array}$$

A. Yes ✗

B. No ✓ k is not defined in $\mathcal{L}_2$

Equality

Are these two libraries equal?

$\mathcal{L}_0$
$k \leftarrow \mathsf{Gen}(1^\lambda)$
GET():
return 42

$\equiv$

$\mathcal{L}_1$
GET():
return 42

- (A) Yes<2> ✓ k is never used, safe to remove it
- (B) No<2> ✗

Method: compute probabilities

Theorem (One-time-pad uniform ciphertext)

$\mathcal{L}_{\text{otp-real}}$		$\mathcal{L}_{\text{otp-rand}}$
$\text{OTENC}(m \in \{0,1\}^\lambda)$: $k \xleftarrow{\$} \{0,1\}^\lambda$ return $k \oplus m$	$\equiv$	$\text{OTENC}(m \in \{0,1\}^\lambda)$: $c \xleftarrow{\$} \{0,1\}^\lambda$ return c

Proof Let $m, \tilde{c} \in \{0,1\}^\lambda$. In $\mathcal{L}_{\text{otp-rand}}$, $\Pr[\text{OTENC}(m) = \tilde{c}] = \frac{1}{2^\lambda}$ (uniform sampling). In $\mathcal{L}_{\text{otp-real}}$:

$$\Pr[\text{OTENC}(m) = \tilde{c}] = \Pr\left[k \oplus m = \tilde{c} \mid k \xleftarrow{\$} \{0,1\}^\lambda\right] = \Pr\left[\tilde{c} \oplus m = k \mid k \xleftarrow{\$} \{0,1\}^\lambda\right]$$

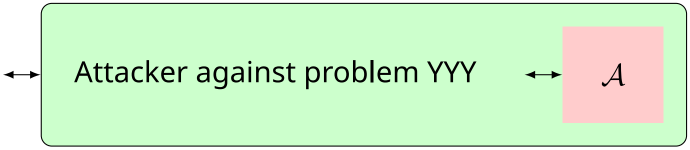
$$= \Pr\left[C = k \mid k \xleftarrow{\$} \{0,1\}^\lambda\right] = \frac{1}{2^\lambda} = \Pr[\text{OTENC}(m) = \tilde{c}]$$

where $C := \tilde{c} \oplus m$. Henc $\mathcal{L}_{\text{otp-real}} = \mathcal{L}_{\text{otp-rand}}$ □

Method: reduction

All the above methods = interchangeability (statistical indistinguishability). What about **computational** indistinguishability? Either directly an assumption that the two libraries are hard to distinguish (possibly need an hybrid sequence first), otherwise:

Reduction!



Idea: to prove $\mathcal{L}_0 \approx \mathcal{L}_1$, assume $\mathcal{L}_0 \not\approx \mathcal{L}_1$, i.e. $\exists$ polynomial adversary $\mathcal{A}$ s.t. $|\Pr[\mathcal{A} \diamond \mathcal{L}_0 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_1 = 1]|$. **Use $\mathcal{A}$ as a subroutine to break a hard problem (compute explicitly the success probability)** $\Rightarrow$ contradiction!

Method: reduction

Option 1: single huge reduction: ✗ hard to write and read

Option 2: hybrids + small reduction ✓ Easier to read and verify

Often not even needed if the assumptions are already expressed as indistinguishable libraries

Method: reduction

Option 1: single huge reduction: ✗ hard to write and read

Option 2: hybrids + small reduction ✓ Easier to read and verify

Often not even needed if the assumptions are already expressed as indistinguishable libraries

Some useful theorems

Bad event lemma

Bad event lemma

Let $\mathcal{L}_{\text{left}}$ and $\mathcal{L}_{\text{right}}$ be two libraries that define a variable named bad, that is initialized to 0. If $\mathcal{L}_{\text{left}}$ and $\mathcal{L}_{\text{right}}$ have identical code except for code blocks reachable only when bad $= 1$ (e.g. guarded with an "if bad $= 1$" statement), then:

$$|\Pr[\mathcal{A} \diamond \mathcal{L}_{\text{left}} = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_{\text{right}} = 1]| \leq \Pr[\mathcal{A} \diamond \mathcal{L}_{\text{left}} \text{ sets bad } = 1] \tag{2}$$

Proof: Define A_{left} the event "$\mathcal{A} \diamond \mathcal{L}_{\text{left}} = 1$", A_{right} the event "$\mathcal{A} \diamond \mathcal{L}_{\text{right}} = 1$", B_{left} the event $\mathcal{A} \diamond \mathcal{L}_{left}$ sets bad $= 1$, and B_{right} the event $\mathcal{A} \diamond \mathcal{L}_{\text{left}}$ sets bad $= 1$, and $\bar{\cdot}$ is the negation of event $\cdot$.

$$\begin{aligned}
&|\Pr[A_{\text{left}}] - \Pr[A_{\text{right}}]| = |\Pr[B_{\text{left}}]\Pr[A_{\text{left}} \mid B_{\text{left}}] + \Pr[\bar{B}_{\text{left}}]\Pr[A_{\text{left}} \mid \bar{B}_{\text{left}}] \\
&- \Pr[B_{\text{right}}]\Pr[A_{\text{right}} \mid B_{\text{right}}] - \Pr[\bar{B}_{\text{right}}]\Pr[A_{\text{right}} \mid \bar{B}_{\text{right}}]| \\
&\leq \Pr[\bar{B}_{\text{left}}]\underbrace{|\Pr[A_{\text{left}} \mid \bar{B}_{\text{left}}] - \Pr[A_{\text{right}} \mid \bar{B}_{\text{right}}]|}_{=0 \text{ (same code when bad is 0)}} + \Pr[B_{\text{left}}]\underbrace{|\Pr[A_{\text{left}} \mid B_{\text{left}}] - \Pr[A_{\text{right}} \mid B_{\text{right}}]|}_{\leq 1} \\
&\leq \Pr[B_{\text{left}}]
\end{aligned}$$

Triangle ineq. & $\Pr[B_{\text{left}}] = \Pr[B_{\text{right}}]$ (identical code before setting bad)

□

Application bad event lemma

?

We want to show that

$\mathcal{L}_{\text{left}}$

PREDICT(x):
  $s \xleftarrow{\$} \{0,1\}^\lambda$
  return $x \stackrel{?}{=} s$

$\approx$

$\mathcal{L}_{\text{right}}$

PREDICT(x):
  return false

. A student already wrote these two hybrid games:

$\mathcal{G}_1$

bad $:= 0$
PREDICT(x):
  $s \xleftarrow{\$} \{0,1\}^\lambda$
  if $x \stackrel{?}{=} s$:
    bad $:= 1$
  return false

and

$\mathcal{G}_2$

bad $:= 0$
PREDICT(x):
  $s \xleftarrow{\$} \{0,1\}^\lambda$
  if $x \stackrel{?}{=} s$:
    bad $:= 1$
    return true
  return false

. How can you finish the proof?

(A) $\mathcal{L}_{\text{left}} = \mathcal{G}_1 \approx \mathcal{G}_2 = \mathcal{L}_{\text{right}}$

(B) $\mathcal{L}_{\text{left}} \approx \mathcal{G}_1 = \mathcal{G}_2 \approx \mathcal{L}_{\text{right}}$

(C) $\mathcal{L}_{\text{left}} = \mathcal{G}_2 \approx \mathcal{G}_1 = \mathcal{L}_{\text{right}}$

(D) $\mathcal{L}_{\text{left}} \approx \mathcal{G}_2 = \mathcal{G}_1 \approx \mathcal{L}_{\text{right}}$

Application bad event lemma

We want to show that

$\mathcal{L}_{\text{left}}$

```
PREDICT(x):
  s <-$ {0,1}^λ
  return x =? s
```

$\approx$

$\mathcal{L}_{\text{right}}$

```
PREDICT(x):
  return false
```

. A student already wrote these two hybrid games:

$\mathcal{G}_1$

```
bad := 0
PREDICT(x):
  s <-$ {0,1}^λ
  if x =? s:
    bad := 1

  return false
```

and

$\mathcal{G}_2$

```
bad := 0
PREDICT(x):
  s <-$ {0,1}^λ
  if x =? s:
    bad := 1
    return true
  return false
```

. How can you finish the proof?

- (A) $\mathcal{L}_{\text{left}} = \mathcal{G}_1 \approx \mathcal{G}_2 = \mathcal{L}_{\text{right}}$
- (B) $\mathcal{L}_{\text{left}} \approx \mathcal{G}_1 = \mathcal{G}_2 \approx \mathcal{L}_{\text{right}}$
- (C) $\mathcal{L}_{\text{left}} = \mathcal{G}_2 \approx \mathcal{G}_1 = \mathcal{L}_{\text{right}}$ ✔ We use the bad event lemma to show $\mathcal{G}_2 \approx \mathcal{G}_1$ $(\Pr[\text{bad} = 1] = \frac{1}{2^\lambda} = \text{negl}(\lambda))$
- (D) $\mathcal{L}_{\text{left}} \approx \mathcal{G}_2 = \mathcal{G}_1 \approx \mathcal{L}_{\text{right}}$

Conclusion

Conclusion

- Can't dissociate cryptography from the **security models** and **proofs**
- **Lot's of parameters** to consider ((un)bounded), computational assumptions, setup assumptions, asymptotic/concrete, security model…
- For us: prove security of protocol = **show that two libraries are indistinguishable**
- One example is the **IND-CPA** security property
- We saw a list of **methods to write security proofs**
- Conversely, **to prove the insecurity** of a protocol we must exhibit an efficient (=polynomial) distinguisher that can distinguish the libraries with a non-negligible advantage