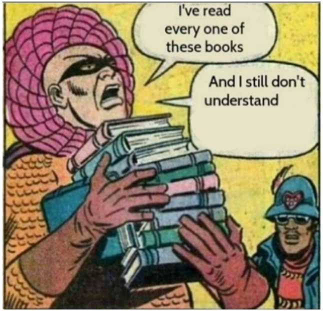
Crypto Engineering 2024
Security definitions & proof methods

Léo Colisson Palais

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

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

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, 3 TDs
- Symmetric cryptography, in particular:
  - Symmetric encryption & block ciphers
  - Authentication (MAC)
  - Hash functions & specificity of password hashing
- Goals:
  - Study **security models**
  - See some **constructions**
  - Analyse and **prove their security**
  - See some bad ideas that you should **NEVER DO**

Symmetric cryptography

With me:

- 5 CMs, 3 TDs
- Symmetric cryptography, in particular:
  - Symmetric encryption & block ciphers
  - Authentication (MAC)
  - Hash functions & specifi
- Goals:
  - Study **security models**
  - See some **constructions**
  - Analyse and **prove their security**
  - See some bad ideas that you should **NEVER DO**

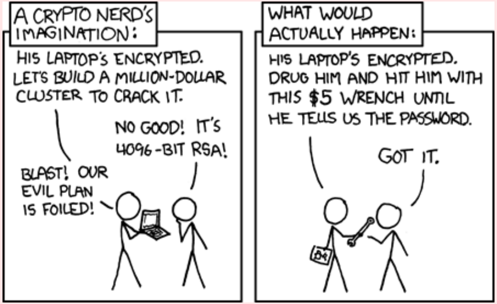
Important to **define them rigorously**, otherwise, easy to declare an insecure protocol secure.
Also important to understand how these definitions influence the security guarantees

Symmetric cryptography

With me:

- 5 CMs, 3 TDs
- Symmetric cryptography, in particul
  - Symmetric encryption & block ciph
  - Authentication (MAC)
  - Hash functions & specificity of pass
- Goals:
  - Study **security models**
  - See some **constructions**
  - Analyse and **prove their security**
  - See some bad ideas that you should **NEVER DO**

Proofs guarantee security in a given attack model, but remember, **a proof is always a model!**



Symmetric cryptography

With me:

- 5 CMs, 3 TDs
- Symmetric cryptography, in particular:
  - Symmetric encryption & block ciphers
  - Authentication (MAC)
  - Hash functions & specificity of password hashing
- Goals:
  - Study **security models**
  - See some **constructions**
  - Analyse and **prove their security**
  - See some bad ideas that you should **NEVER DO**

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. $\forall c \in \mathbb{N}, \lim_{\lambda \to \infty} \lambda^c f(\lambda) = 0$
$\mathsf{poly}(\lambda)$	A function smaller than some polynomials, 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 $f(\lambda) = 2^{-\log \lambda}$

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

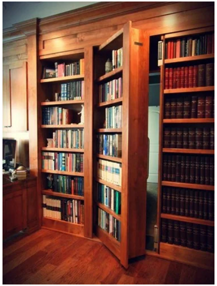
Symmetric encryption

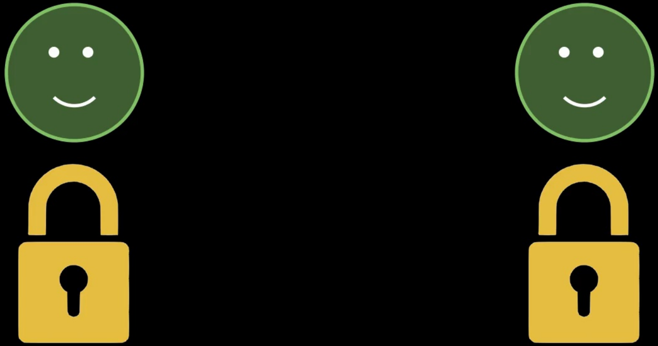
Both parties share the same secret

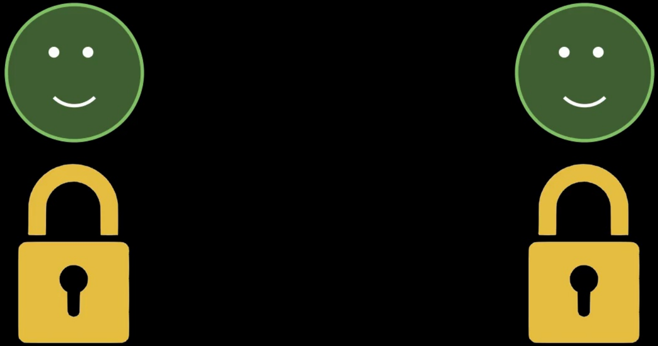
$\neq$

Asymmetric encryption

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



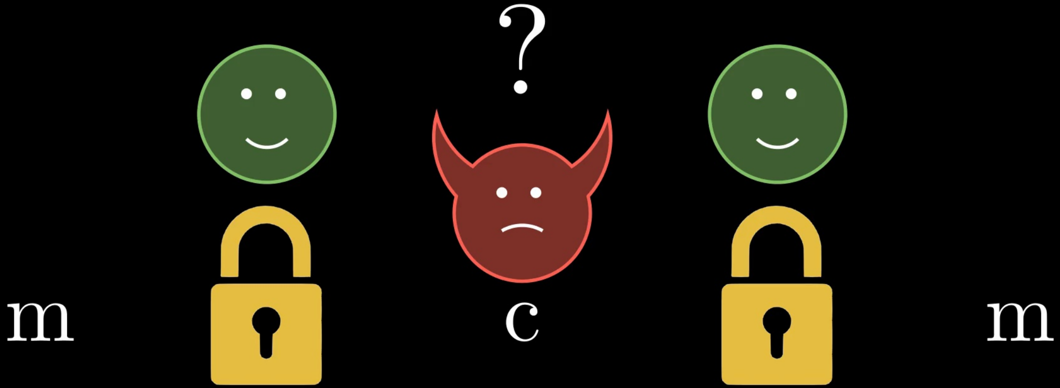





m


m
c


?
m
c


m
?
c
m

Mini-activity

Activity: design your own private-key cryptosystem (2mn) that we will analyse later, i.e.:

- Key-generation $k \leftarrow \mathsf{Gen}(1^\lambda)$
- Encryption $c \leftarrow \mathsf{Enc}_k(m)$
- Decryption $m \leftarrow \mathsf{Dec}_k(c)$

Mini-activity

Activity: design your own private-key cryptosystem (2mn) that we will analyse later, i.e.: Key $k \in \mathcal{K}$

- Key-generation $k \leftarrow \mathsf{Gen}(1^\lambda)$
- Encryption $c \leftarrow \mathsf{Enc}_k(m)$
- Decryption $m \leftarrow \mathsf{Dec}_k(c)$

Mini-activity

Security parameter $\lambda \in \mathbb{N}$ in unary form:
Gen runs in poly time in the size of its input

Activity: design your own private-key cryptosystem (2mn) that we will analyse later, i.e.:

Key $k \in \mathcal{K}$

- Key-generation $k \leftarrow \mathsf{Gen}(1^\lambda)$
- Encryption $c \leftarrow \mathsf{Enc}_k(m)$
- Decryption $m \leftarrow \mathsf{Dec}_k(c)$

Mini-activity

Security parameter $\lambda \in \mathbb{N}$ in unary form:
Gen runs in poly time in the size of its input

Activity: design your own private-key cryptosystem (2mn) that we will analyse later, i.e.:

Key $k \in \mathcal{K}$

- Key-generation $k \leftarrow \mathsf{Gen}(1^\lambda)$
- Encryption $c \leftarrow \mathsf{Enc}_k(m)$ — Message $m \in \mathcal{M}$
- Decryption $m \leftarrow \mathsf{Dec}_k(c)$

Mini-activity

Security parameter $\lambda \in \mathbb{N}$ in unary form:
Gen runs in poly time in the size of its input

Activity: design your own private-key cryptosystem (2mn) that we will analyse later, i.e.:

Key $k \in \mathcal{K}$

- Key-generation $k \leftarrow \text{Gen}(1^\lambda)$
- Encryption $c \leftarrow \text{Enc}_k(m)$ ← Message $m \in \mathcal{M}$
- Decryption $m \leftarrow \text{Dec}_k(c)$

Ciphertext $c \in \mathcal{C}$

Symmetric vs asymmetric cryptography

Asymmetric encryption

😀 No need to share secrets (e.g. internet)

😰 Stronger assumptions factoring, LWE... (functions highly structured)

😰 Less efficient

😰 No statistical security

Symmetric encryption

😰 Need to share secrets

😀 Weaker assumptions (less structure)

😀 More efficient

😀 Statistical security possible (but impractical)

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

Cryptography is not (just) encryption

WARNING

Cryptography is not just about encryption:

- cryptocurrency (bitcoin. . .)
- signature
- commitments
- multi-party computing (MPC)
- quantum money
- position verification
- zero-knowledge (ZK) proofs
- electronic voting
- . . .

Impagliazzo's worlds

Obfustopia	One can make a program code unreadable
Cryptomania	Public-key cryptography and trapdoor functions exist
Minicrypt	One way functions exist, no public key cryptography
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)

Obfuscation

MPC

PKC

Asymmetric cryptography

ZK-proofs Secret sharing

Commitments

One-Time Pad

No (interesting) cryptography

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	Asymmetric cryptography ZK-proofs Secret sharing Commitments
Pessiland	NP problems hard in practice, but no one way functions exist	**No (interesting) cryptography**
Heuristica	NP problems hard in the worst case, easy in practice (average)	
Algorithmica	P = NP (no hard problem)	One-Time Pad

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

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	Asymmetric 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

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 absolute 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

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!

Security models

When designing a crypto system, we want to say:

Setup assumption

Computational assumption

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

Security models

When designing a crypto system, we want to say:

Setup assumption

Computational assumption

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

???

Security models

When designing a crypto system, we want to say:

Setup assumption

Computational assumption

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

security model

$\Rightarrow$ We also need to define a ***security model*** (a.k.a *attack model*)
$=$ expectations in term of security (e.g. the adversary should not learn the message)

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)

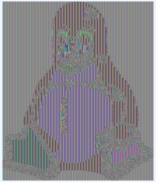
- (A) Yes
- (B) No

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)

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



NEVER DO THIS

AN ENCRYPTION MUST ALWAYS BE NON-DETERMINISTIC!!!

NEVER DO THIS

Was it the case of your encryption algorithm?

AN ENCRYPTION MUST ALWAYS BE NON-DETERMINISTIC!!!

NEVER DO THIS

Was it the case of your encryption algorithm?

AN ENCRYPTION MUST ALWAYS BE NON-DETERMINISTIC!!!

NEVER USE A HOME-MADE ENCRYPTION, IT WILL BE INSECURE!!!

?

Attempt 3: "Given 2 random messages m_0 and m_1 (known to the adversary), an adversary should not be able to tell if the message m_0 or m_1 was encrypted.". Is this a good security definition? (if not, find a scenario where this could go wrong)

- (A) Yes
- (B) No

Security models

Attempt 3: "Given 2 random messages m_0 and m_1 (known to the adversary), an adversary should not be able to tell if the message m_0 or m_1 was encrypted.". Is this a good security definition? (if not, find a scenario where this could go wrong)

?

- (A) Yes ✗
- (B) No ✓ Good enough if we encrypt random messages... But in practice we encrypt precise messages, say "Yes" and "No", and it could be a very bad encryption for these precise two messages while still being good on all others.

Security models

Attempt 4: "**For all** messages m_0 and m_1 (known to the adversary), an adversary should not be able to tell if the message m_0 or m_1 was encrypted.". Is this a good security definition? (if not, find a scenario where this could go wrong)

- (A) Yes
- (B) No

Security models

?

Attempt 4: "**For all** messages m_0 and m_1 (known to the adversary), an adversary should not be able to tell if the message m_0 or m_1 was encrypted.". Is this a good security definition? (if not, find a scenario where this could go wrong)

- Ⓐ Yes ✗
- Ⓑ No ✓ This is actually **too strong**: when $m_0 = k$ and $m_1 = 0$, the adversary can just use m_0 (i.e. k) to decrypt. And if we also require k to be sampled *after* m_0 (so that m_0 and k are independent), this is **too weak**: in practice, the message may depend on k (e.g. after seeing a previous encryption).

Security models

Attempt 4: "**For all** messages m_0 and m_1 (known to the adversary), an adversary should not be able to tell if the message m_0 or m_1 was encrypted.". Is this a good security definition? (if not, find a scenario where this could go wrong)

?

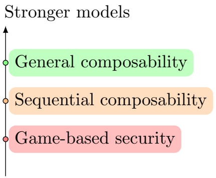
- (A) Yes ✗
- (B) No ✓ This is actually **too strong**: when $m_0 = k$ and $m_1 = 0$, the adversary can just use m_0 (i.e. k) to decrypt. And if we also require k to be sampled *after* m_0 (so that m_0 and k are independent), this is **too weak**: in practice, the message may depend on k (e.g. after seeing a previous encryption).

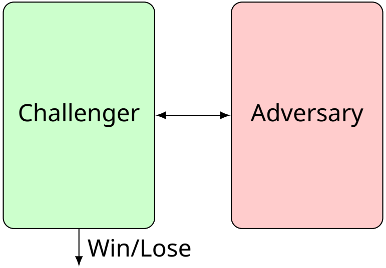
The adversary should choose m_0 and m_1, but when? What can the adversary use before choosing them? How to formalize this?

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** (defines the security goals):





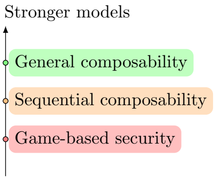
Secure if for any adversary, **the probability of winning is “low”**
(might be $1/2 + \text{negl}(\lambda)$ or $0 + \text{negl}(\lambda)$ depending on the game)

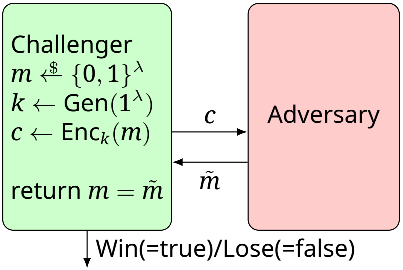
Security models

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

3 **classes** of security models:

1: Game-based security = Fix a **challenger** (defines the security goals):





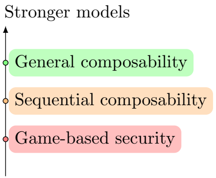
Secure if for any adversary, **the probability of winning is "low"**
(might be $1/2 + \mathsf{negl}(\lambda)$ or $0 + \mathsf{negl}(\lambda)$ depending on the game)

Security models

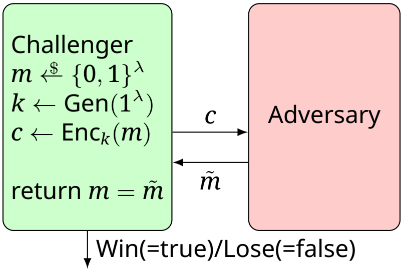
So how to define a secure protocol/encryption? $\Rightarrow$ There is not one, but **multiple** definitions of s

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** (defines the security goals):

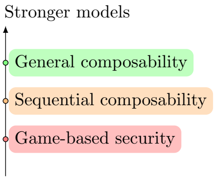


Secure if for any adversary, **the probability of winning is "low"**
(might be $1/2 + \mathsf{negl}(\lambda)$ or $0 + \mathsf{negl}(\lambda)$ depending on the game)

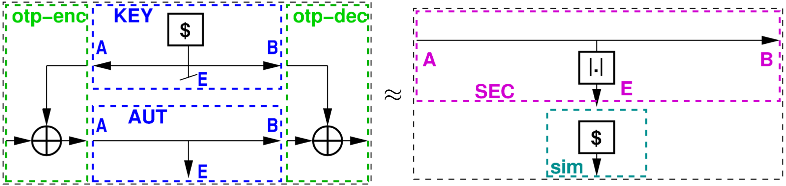
Security models

So how to define a secure protocol/encryption? $\Rightarrow$ 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

[GM84]

Security frameworks: comparison

Focus of this course

	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

[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)

Kerckhoff's principle

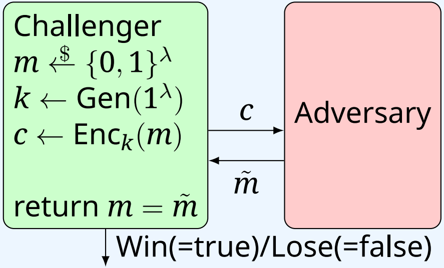
Kerckhoff's principle

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

Questions

Consider the following challenger: is it modeling:

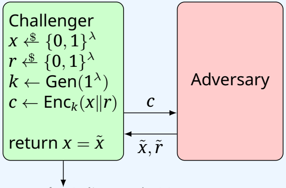
- (A) a passive adversary,
- (B) an active one?

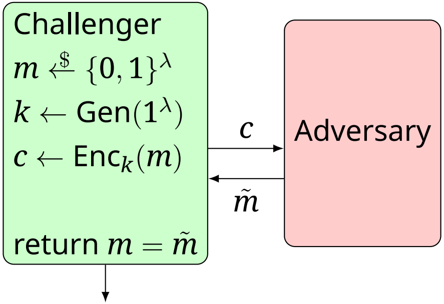


Questions

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 for $\mathcal{A}$ to learn x is 0
- (B) $\mathcal{A}$ has negligible chance to learn the first half of x
- (C) $\mathcal{A}$ has negligible chance to learn all bits of x
- (D) $\mathcal{A}$ has negligible chance to learn all bits of r
- (E) If in practice an adversary can observe arbitrary pairs of messages and their encryption, they are still unable to recover x





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

Equivalent notations/formulations

Challenger

$m \xleftarrow{\$} \{0,1\}^\lambda$

$k \leftarrow \mathsf{Gen}(1^\lambda)$

$c \leftarrow \mathsf{Enc}_k(m)$

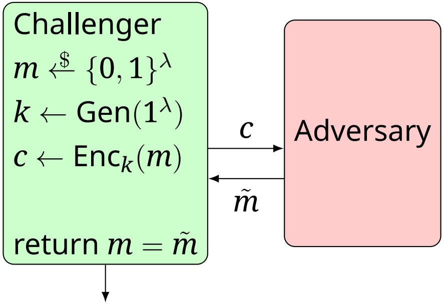
return $m = \tilde{m}$

$\xrightarrow{c}$ **Adversary** $\xleftarrow{\tilde{m}}$

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

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

Equivalent notations/formulations

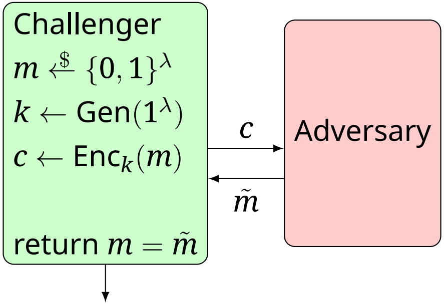


$$= \Pr_{\substack{m \xleftarrow{\$} \{0,1\}^\lambda \\ k \leftarrow \mathsf{Gen}(1^\lambda) \\ c \leftarrow \mathsf{Enc}_k(m) \\ \tilde{m} \leftarrow \mathcal{A}(c)}} [m = \tilde{m}] = \Pr_{\substack{m \xleftarrow{\$} \{0,1\}^\lambda \\ \tilde{m} \leftarrow \mathcal{A} \diamond \mathcal{L}_m}} [m =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

$\mathcal{L}_m$
$k \leftarrow \mathsf{Gen}(1^\lambda)$ $c \leftarrow \mathsf{Enc}_k(m)$ <u>GETC():</u>   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!

Game-based security

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

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?

Ⓐ Yes ✗

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

Game-based security

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 $\mathcal{A}(n)$ is $\sqrt{n}$, is $\mathcal{A}$ polynomial?

(A) Yes

(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 $= 1/2 + \text{negl}(\lambda)$
  $\Rightarrow$ We define the **advantage**:

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

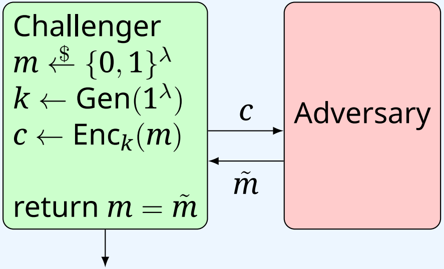
NB: theoretically, security is an **asymptotic** notion!

Search vs decision

?

Consider the following challenger, is it modeling:

- (A) a search problem
- (B) a decision problem

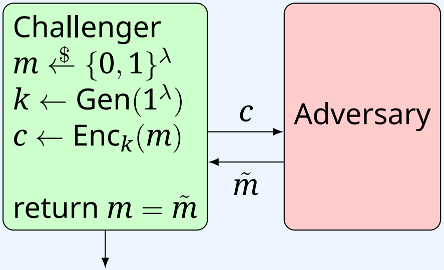


Search vs decision

?

Consider the following challenger, is it modeling:

- (A) a search problem ✔
- (B) a decision problem ✘



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:

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

(roughly $4\times$ age of earth)

How to write security proofs

Goal

Focus: decision problems. Goal: bound $|\Pr[\mathcal{A} \diamond \mathcal{L}_0 = 1] - \Pr[\mathcal{A} \diamond \mathcal{L}_1 = 1]|$.

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 $\mathcal{A}$,

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

Goal

Definition (Indistinguishability)

Two libraries $\mathcal{L}_0$ and $\mathcal{L}_1$ are *indistinguishable*, written $\mathcal{L}_0 \approx \mathcal{L}_1$, if for any adversary $\mathcal{A}(1^\lambda)$ running in polynomial time and outputting a single bit:

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

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

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

But

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

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 $\mathsf{negl}(\lambda)$. $\square$

Reduction

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, they $\mathcal{A}$ 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

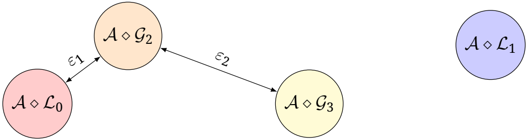
Proof = sequence of **hybrid** games:

Hybrid games

Proof = sequence of **hybrid** games:

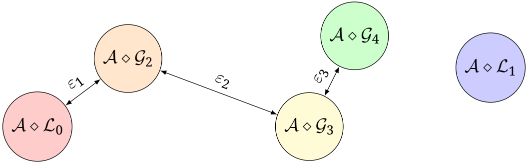
Hybrid games

Proof = sequence of **hybrid** games:



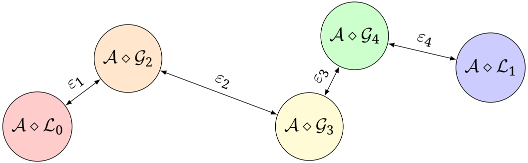
Hybrid games

Proof = sequence of **hybrid** games:



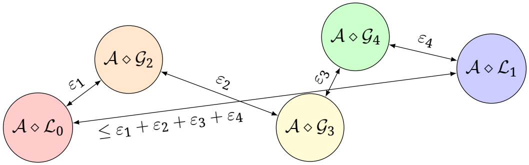
Hybrid games

Proof = sequence of **hybrid** games:

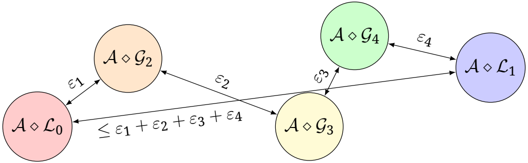


Hybrid games

Proof = sequence of **hybrid** games:



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?

?

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

- (A) Yes
- (B) No

Equality

Are these two libraries equal?

?

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

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

Equality

?

Are these two libraries equal?

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

(A) Yes

(B) No

Equality

Are these two libraries equal?

?

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

- (A) Yes ✗
- (B) No ✓ k is not defined in $\mathcal{L}_2$

Equality

Are these two libraries equal?

$\mathcal{L}_0$
$k \leftarrow \text{Gen}(1^\lambda)$
<u>GET():</u>
return 42

$\equiv$

$\mathcal{L}_1$
<u>GET():</u>
return 42

Ⓐ Yes

Ⓑ No

Equality

?

Are these two libraries equal?

$$\boxed{\begin{array}{l} \mathcal{L}_0 \\ \hline k \leftarrow \mathsf{Gen}(1^\lambda) \\ \underline{\mathsf{GET}():} \\ \quad \text{return } 42 \end{array}} \equiv \boxed{\begin{array}{l} \mathcal{L}_1 \\ \hline \underline{\mathsf{GET}():} \\ \quad \text{return } 42 \end{array}}$$

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

Method: compute probabilities

Theorem (One-time-pad uniform ciphertext)

$$\begin{array}{|l|}\hline \mathcal{L}_{\text{otp-real}} \\ \hline \underline{\text{OTENC}(m \in \{0,1\}^\lambda):} \\ \quad k \xleftarrow{\$} \{0,1\}^\lambda \\ \quad \text{return } k \oplus m \\ \hline \end{array} \equiv \begin{array}{|l|}\hline \mathcal{L}_{\text{otp-rand}} \\ \hline \underline{\text{OTENC}(m \in \{0,1\}^\lambda):} \\ \quad c \xleftarrow{\$} \{0,1\}^\lambda \\ \quad \text{return } c \\ \hline \end{array}$$

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 $\mathcal{L}_{\text{otp-real}}$:

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

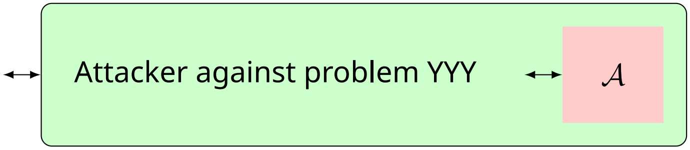
$$= \Pr\left[C = k \mid k \xleftarrow{\$} \{0,1\}^\lambda\right] = \frac{1}{2^\lambda} = \Pr[\text{OTENC}(m) = \tilde{c}]$$

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

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

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:

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

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 $\cdot$.

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

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

$\square$

Application bad event lemma

?

We want to show that

$\mathcal{L}_{\text{left}}$
$\underline{\text{PREDICT}(x):}$
$s \xleftarrow{\$} \{\mathtt{0},\mathtt{1}\}^\lambda$
return $x \stackrel{?}{=} s$

$\approx$

$\mathcal{L}_{\text{right}}$
$\underline{\text{PREDICT}(x):}$
return `false`

. A student already wrote these two hybrid games:

$\mathcal{G}_1$
bad $:= 0$
$\underline{\text{PREDICT}(x):}$
$s \xleftarrow{\$} \{\mathtt{0},\mathtt{1}\}^\lambda$
if $x \stackrel{?}{=} s$:
bad $:= 1$
return `false`

and

$\mathcal{G}_2$
bad $:= 0$
$\underline{\text{PREDICT}(x):}$
$s \xleftarrow{\$} \{\mathtt{0},\mathtt{1}\}^\lambda$
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):
  $s \xleftarrow{\$} \{\texttt{0},\texttt{1}\}^\lambda$
  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):
  $s \xleftarrow{\$} \{\texttt{0},\texttt{1}\}^\lambda$
  if $x \stackrel{?}{=} s$:
    bad $:= 1$
  return `false`

and

$\mathcal{G}_2$

bad $:= 0$
PREDICT(x):
  $s \xleftarrow{\$} \{\texttt{0},\texttt{1}\}^\lambda$
  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) $\mathcal{L}_{\text{left}} = \mathcal{G}_2 \approx \mathcal{G}_1 = \mathcal{L}_{\text{right}}$ ✔ 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) $\mathcal{L}_{\text{left}} \approx \mathcal{G}_2 = \mathcal{G}_1 \approx \mathcal{L}_{\text{right}}$

How to prove INsecurity?

How to prove INsecurity

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

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

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?

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

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

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

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

?

1. $\mathcal{A}$: $c := \textsc{ctxt}(0^\lambda)$; 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. $\mathcal{A}$: $c := \textsc{ctxt}(1^\lambda)$; 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?

$\mathcal{L}^{\Sigma}_{\text{ots\$-real}}$
$\textsc{ctxt}(m \in \{0,1\}^\lambda)$:
$k \leftarrow \{0,1\}^\lambda$ // Σ.KeyGen
$c := k \,\&\, m$ // Σ.Enc
return c

$\mathcal{L}^{\Sigma}_{\text{ots\$-rand}}$
$\textsc{ctxt}(m \in \{0,1\}^\lambda)$:
$c \leftarrow \{0,1\}^\lambda$ // $\Sigma.C$
return c

?

1

$\mathcal{A}$
$c := \textsc{ctxt}(0^\lambda)$
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

$\mathcal{A}$
$c := \textsc{ctxt}(1^\lambda)$
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)

Uniform vs non-uniform

Small subtleties: we always consider infinite sequences of adversaries, based on security parameter λ. How do we define these algorithms?

- **Uniform algorithm**: same Turing machine for all instance size
- **Non-uniform algorithm**: sequence $\{C_\lambda\}_{\lambda \in \mathbb{N}}$ of circuits, or, equivalently, a fixed Turing machine with an auxiliary "advice" input, identical for all instances of same size

Non-uniform adversaries = slightly stronger (P$/$poly vs P) + somewhat unrealistic, but appear naturally e.g. in simulation-based security (see [Lindel 17] for examples)

Uniform vs non-uniform

In practice, **not a big deal**:

- Mostly changes assumptions: “YYY is hard to solve in polynomial time” $\Rightarrow$ “YYY is hard against non-uniform adversaries”
- But all common assumptions are believed to hold in both cases anyway