CS4501 ~~Cryptographic Protocols~~
Intro to Cryptography Speedrun
Lecture 14: Public Key Encryption, CPA-Security, Oblivious Transfer

https://jackdoerner.net/teaching/#2026/Spring/CS4501

Roadmap of the Last Few Lectures

- Introducing Oblivious Transfer.
- Bounded Adversaries and Computational Security.
- The Cryptographic Toolkit: One-Way Functions.
- The Cryptographic Toolkit: Assumptions on Groups.
- A Simple Application: Key Agreement.
- Ensembles and Computational Indistinguishability.
- Computational Security for Protocols.
- The Cryptographic Toolkit: Public-key Encryption.
- Realizing Oblivious Transfer.

Recap: Computational Indistinguishability

Definition 1 (Ensemble): Suppose that for $\kappa \in \mathbb{N}$, X_κ is a distribution on some domain (these distributions and domains need not be related in any particular way). We say that the set $\mathfrak{X} = \{X_\kappa\}_{\kappa\in\mathbb{N}} = \{X_1, X_2, \dots\}$ is an *ensemble of distributions.*

Def. 2 (Computational Indistinguishability): Let $\mathfrak{X} = \{X_\kappa\}_{\kappa\in\mathbb{N}} = \{X_1, X_2, \dots\}$ and $\mathfrak{Y} = \{Y_\kappa\}_{\kappa\in\mathbb{N}} = \{Y_1, Y_2, \dots\}$ be ensembles. We say that $\mathfrak{X}$ and $\mathfrak{Y}$ *computationally indistinguishable* if, for every PPT distinguisher algorithm $\mathscr{D}$, there exists a negligible function ε such that

$$\left|\Pr\left[\mathscr{D}(1^\kappa, x) = 1 : x \leftarrow X_\kappa\right] - \Pr\left[\mathscr{D}(1^\kappa, y) = 1 : y \leftarrow Y_\kappa\right]\right| \leq \varepsilon(\kappa)$$

Notation 1: we use $\mathfrak{X} \approx_c \mathfrak{Y}$ to indicate that the ensembles $\mathfrak{X}$ and $\mathfrak{Y}$ are computationally indistinguishable.

Notation 2 : by abuse of notation, we use $\mathfrak{X} \equiv \mathfrak{Y}$ to indicate that the ensembles $\mathfrak{X}$ and $\mathfrak{Y}$ comprise distributions such that each corresponding pair are identical.

Def 3: The Decisional Diffie-Hellman Game

- Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q$, $|q| = \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (written multiplicatively). In other words, $\mathcal{G}$ samples a cyclic group given the security parameter.
- The *Decisional Diffie-Hellman Game* $\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},b}(\kappa)$ for $b \in \{0,1\}$ is played between a challenger and distinguisher. The game's output is the output of the distinguisher.

$\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},0}(\kappa)$

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$a, b \leftarrow \mathbb{Z}_q$

$(A, B, C) := (g^a, g^b, g^{a \cdot b})$

$(\mathbb{G}, g, q, A, B, C)$ →

outputs a bit b'

$\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},1}(\kappa)$

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$a, b, c \leftarrow \mathbb{Z}_q$

$(A, B, C) := (g^a, g^b, g^c)$

$(\mathbb{G}, g, q, A, B, C)$ →

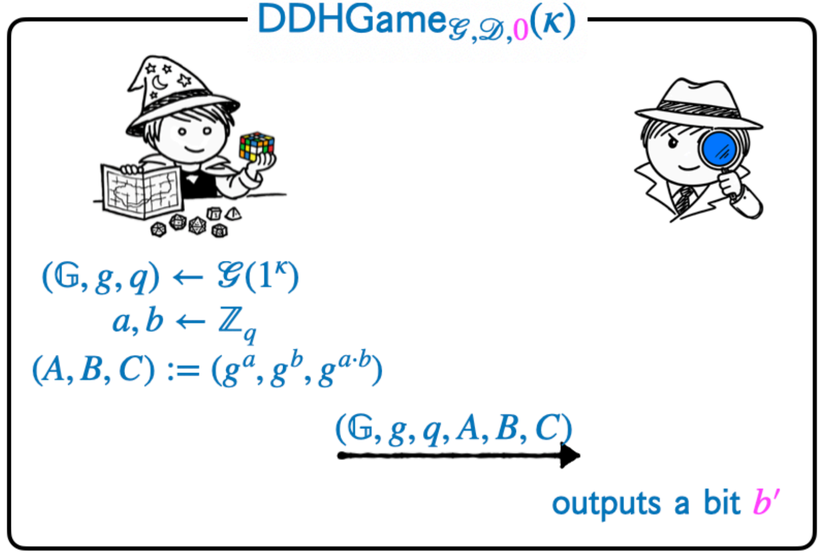
outputs a bit b'

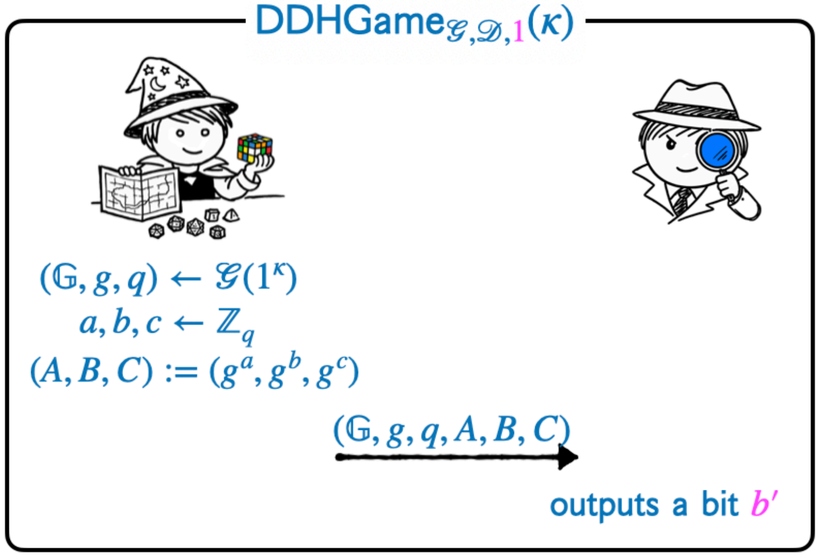
Recap: The Decisional Diffie-Hellman Assumption

Definition 4 (DDH Hardness): The Decisional Diffie-Hellman Problem is *hard* relative to $\mathcal{G}$ if for every PPT distinguisher $\mathcal{D}$, there exists some negligible function ε such that

$$\left| \begin{array}{l} \Pr\left[\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},0}(\kappa) = 1\right] \\ \quad - \Pr\left[\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},1}(\kappa) = 1\right] \end{array} \right| \leq \varepsilon(\kappa)$$

Definition 5 (DDH Assumption): *The Decisional Diffie-Hellman Assumption* asserts that there exists a PPT algorithm $\mathcal{G}$ relative to which the DDH problem is hard.





Recap: Comparing Assumptions on a Group

	Discrete Log	Computational DH	Decisional DH
Type	Search ($\mathcal{A}$ solves a puzzle)	Search ($\mathcal{A}$ solves a puzzle)	Decision ($\mathcal{D}$ distinguishes distributions)
Intuition	Given g^x, computing x is almost as hard as guessing it.	Given g^a and g^b, computing $g^{a \cdot b}$ is almost as hard as computing a and b.	Given g^a, g^b, and g^c, determining whether $c = a \cdot b$ is almost as hard as computing $g^{a \cdot b}$.

DL assumption is false $\Longrightarrow$ CDH assumption is false $\Longrightarrow$ DDH assumption is false

DL assumption is true $\Longleftarrow$ CDH assumption is true $\Longleftarrow$ DDH assumption is true

Weaker $\longleftarrow$ Stronger

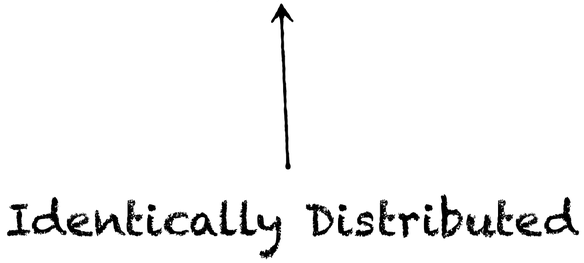
Recall: Simplified Definition for Semi-Honest $\mathcal{A}$

Definition 6. Perfect Semi-Honest SFE (Simplified):

Let $n, t \in \mathbb{N}$ such that $t < n$, let $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ be a (possibly randomized) n-ary function, and let π be a n-party protocol.

We say that π *perfectly securely computes* f in the presence of a semi-honest adversary that statically corrupts up to t parties if there exists a *simulator algorithm* Sim such that for every $I \subseteq [n]$ of size $|I| \leq t$ and every vector $\vec{x} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ we have

$$\left(\mathsf{Sim}\left(I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x})\right) \equiv (\mathsf{VIEW}_I, \mathsf{OUTPUT}_\pi)$$



Now: Simplified Definition for Bounded SH $\mathscr{A}$

Definition ~~6~~ 7. ~~Perfect~~ Computational Semi-Honest SFE (Simplified):

Let $\kappa, n, t \in \mathbb{N}$ such that $t < n$ and n is upper-bounded by some polynomial in κ, let $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ be a (possibly randomized) n-ary function, and let π be a n-party protocol that runs in PPT relative to κ.

We say that π ~~*perfectly*~~ *securely computes* f in the presence of a bounded semi-honest adversary that statically corrupts up to t parties if there exists a PPT *simulator algorithm* Sim such that for every $I \subseteq [n]$ of size $|I| \leq t$ and every $\vec{x} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ we have

$$\left\{ \left(\mathsf{Sim}\left(1^\kappa, I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x}) \right) \right\}_{\kappa \in \mathbb{N}} \approx_c \left\{ (\mathsf{VIEW}_I, \mathsf{OUTPUT}_\pi) \right\}_{\kappa \in \mathbb{N}}$$

Computationally Indistinguishable

Note: to make things simple, the parties will receive (the correct value of) 1^κ as input!

Recall: Even Simpler Def for Deterministic f

Definition 8. Perfect Semi-Honest Deterministic SFE:

Let $n, t \in \mathbb{N}$ such that $t < n$, let $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ be an n-ary function, **(which does not use any randomness)** and let π be a n-party protocol.

We say that π *perfectly securely computes* f in the presence of a semi-honest adversary that statically corrupts up to t parties if

1. For every input vector $\vec{x} \in \mathcal{X}_1 \times \ldots \mathcal{X}_n$ it holds that $f(\vec{x}) \equiv \mathsf{OUTPUT}_\pi$.
2. There exists a *simulator algorithm* Sim such that for every $I \subseteq [n]$ of size $|I| \leq t$ and every vector $\vec{x} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ it holds that

$$\mathsf{Sim}\left(I, \vec{x}_I, f_I(\vec{x})\right) \equiv \mathsf{VIEW}_I$$

Now: Even Simpler Def for Deterministic f

Definition ~~8~~ 9. ~~Perfect~~ Computational Semi-Honest Deterministic SFE:

Let $\kappa, n, t \in \mathbb{N}$ such that $t < n$ and n is upper-bounded by some polynomial in κ, let $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ be a n-ary function, **(which does not use any randomness)** and let π be a n-party protocol that runs in PPT relative to κ.

We say that π ~~*perfectly*~~ *securely computes* f in the presence of a bounded semi-honest adversary that statically corrupts up to t parties if

1. There exists some negligible function ε such that for every input vector $\vec{x} \in \mathcal{X}_1 \times \ldots \mathcal{X}_n$ it holds that $\Pr[f(\vec{x}) \neq \mathsf{OUTPUT}_\pi] \leq \varepsilon(\kappa)$.
2. There exists a PPT *simulator algorithm* Sim such that for every $I \subseteq [n]$ of size $|I| \leq t$ and every vector $\vec{x} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ it holds that

$$\left\{ \mathsf{Sim}\left(1^\kappa, I, \vec{x}_I, f_I(\vec{x})\right) \right\}_{\kappa \in \mathbb{N}} \approx_c \{\mathsf{VIEW}_I\}_{\kappa \in \mathbb{N}}$$

Remember: This is about Distinguishing!

When we write

$$\left\{\left(\mathsf{Sim}\left(1^\kappa, I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x})\right)\right\}_{\kappa\in\mathbb{N}} \approx_c \left\{(\mathsf{VIEW}_I, \mathsf{OUTPUT}_\pi)\right\}_{\kappa\in\mathbb{N}}$$

we are invoking the definition of computational indistinguishability!

The above statement is equivalent to writing:
For every PPT distinguisher $\mathscr{D}$ there exists a negligible function ε such that

$$\left|\Pr\left[\mathscr{D}\left(1^\kappa, \quad\right) = 1\right] - \Pr\left[\mathscr{D}\left(1^\kappa, \quad\right) = 1\right]\right| \leq \varepsilon(\kappa)$$

Remember: This is about Distinguishing!

When we write

$$\left\{\left(\mathsf{Sim}\left(1^\kappa, I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x})\right)\right\}_{\kappa\in\mathbb{N}} \approx_c \left\{(\mathsf{VIEW}_I, \mathsf{OUTPUT}_\pi)\right\}_{\kappa\in\mathbb{N}}$$

we are invoking the definition of computational indistinguishability!

The above statement is equivalent to writing:
For every PPT distinguisher $\mathscr{D}$ there exists a negligible function ε such that

$$\left|\begin{array}{r}\Pr\left[\mathscr{D}\left(1^\kappa, \left(\mathsf{Sim}\left(1^\kappa, I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x})\right)\right) = 1\right] \\ -\Pr\left[\mathscr{D}\left(1^\kappa, (\mathsf{VIEW}_I, \mathsf{OUTPUT}_\pi)\right) = 1\right]\end{array}\right| \leq \varepsilon(\kappa)$$

Re-Orienting Ourselves

So far we have seen:

- A definition for “realistically” computationally bounded algorithms.
- A definition for sequences of distributions that are indistinguishable to computationally-bounded distinguishers (**Def. 1**).
- A definition for security against computationally bounded adversaries (**Def. 7+9**).
- The Diffie-Hellman key exchange protocol, which is based upon groups.
- The decisional Diffie-Hellman assumption (**Def. 5**), which asserts that the keys that are produced by the DHKE protocol are computationally indistinguishable from uniformly sampled group elements.

What remains:

- A proof that DHKE satisfies our security definition, under the DDH assumption.

What Will a Proof Look Like?

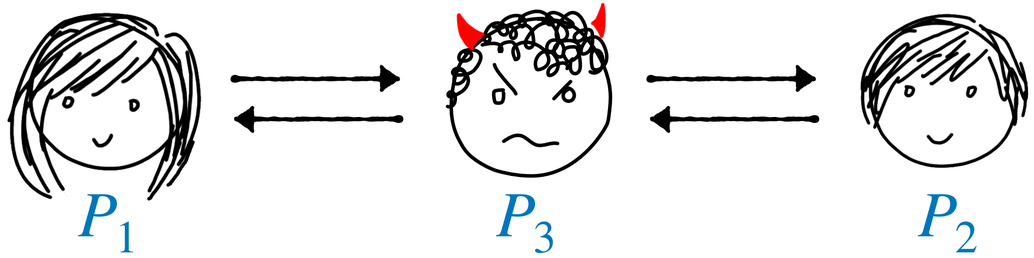
- The hypothesis of our theorem statement must be the truth of the DDH assumption.
- The conclusion will be the security of the protocol (when nobody is corrupt).
- The easiest way to connect them is contrapositively: if the protocol is insecure, then the DDH assumption is false.
- The security of the protocol is invalidated by the existence of a PPT distinguisher $\mathscr{D}$ that has a non-negligible advantage for all simulator algorithms Sim.
- The assumption is falsified by the existence of a PPT distinguisher $\mathscr{D}'$ that has a non-negligible advantage in the DDH game.
- Therefore, we have to show that if we are given any $\mathscr{D}$ satisfying the above criteria, we can build $\mathscr{D}'$ satisfying the above criteria.
- In other words, we are *reducing* the problem of breaking DDH to the problem of breaking security for DHKE. This kind of "indirect cryptanalysis" of a fundamental assumption is called a *security reduction*.

What Will a Proof Look Like?

- The security of the protocol is invalidated by the existence of a PPT distinguisher $\mathscr{D}$ that has a non-negligible advantage for all simulator algorithms Sim.
- The assumption is falsified by the existence of a PPT distinguisher $\mathscr{D}'$ that has a non-negligible advantage in the DDH game.
- Therefore, we have to show that if we are given any $\mathscr{D}$ satisfying the above criteria, we can build $\mathscr{D}'$ satisfying the above criteria.
- In other words, we are *reducing* the problem of breaking DDH to the problem of breaking security for DHKE. This kind of "indirect cryptanalysis" of a fundamental assumption is called a *security reduction*.
- A non-negligible function has an infinite number of points that are (at least) inverse polynomial relative in its input.
- It's sufficient to show that for *any one* value of κ where $\mathscr{D}$ has inverse-polynomial advantage, $\mathscr{D}'$ does too. We call the difference between their advantages the *loss*. If there is no loss, then we say that the reduction is *tight*.

Fudging the Model

- Recall that in the DHKE protocol, we only assume *authenticated* channels, which the adversary can inspect. The adversary therefore learns $(\mathbb{G}, g, q)$, which are transmitted "in the clear," even if nobody is corrupted.
- The simplified model that we have been using assumes *secure* channels that the adversary cannot inspect, and we have not included any method to leak information to an adversary that corrupts nobody.
- Therefore, to fit the DHKE protocol into the model we have specified, we will modify it so that Alice and Bob can only communicate via a corrupted third party, Charlie. Alice and Bob's messages will not be altered relative to what we saw earlier.
- Since the adversary is semi-honest, Charlie will always forward messages between the parties faithfully, just like an authenticated channel would.



Recall: The (Fudged) DHKE Protocol

- Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q$, $|q| = \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (written multiplicatively). In other words, $\mathcal{G}$ samples a cyclic group given the security parameter.

receives input 1^κ

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$a \leftarrow \mathbb{Z}_q,\ A := g^a$

receives input 1^κ

receives input 1^κ

$(\mathbb{G}, g, q, A)$

B

$(\mathbb{G}, g, q, A)$

B

$b \leftarrow \mathbb{Z}_q,\ B := g^b$

outputs $k := B^a$

outputs $(\mathbb{G}, g, q)$

outputs $k := A^b$

Question 1: *why do they all need to receive the security parameter?*

Recall: The (Fudged) DHKE Protocol

- Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q$, $|q| = \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (written multiplicatively). In other words, $\mathcal{G}$ samples a cyclic group given the security parameter.

receives input 1^κ

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$a \leftarrow \mathbb{Z}_q,\ A := g^a$

receives input 1^κ

receives input 1^κ

$(\mathbb{G}, g, q, A)$

B

$(\mathbb{G}, g, q, A)$

B

$b \leftarrow \mathbb{Z}_q,\ B := g^b$

outputs $k := B^a$

outputs $(\mathbb{G}, g, q)$

outputs $k := A^b$

Question 2: *what might go wrong in the proof if* P_1 *or* P_2 *is corrupted instead of* P_3*?*

Security for the (Fudged) DHKE Protocol

- Let $f^{\mathscr{G}}_{\mathsf{FDHKE}} : \{1\}^* \times \{1\}^* \times \{1\}^* \to \{0,1\}^* \times \{0,1\}^* \times \mathrm{image}(\mathscr{G})$ be the function that computes $f^{\mathscr{G}}_{\mathsf{FDHKE}}(1^\kappa, 1^\kappa, 1^\kappa) = (k, k, (\mathbb{G}, g, q)) : (\mathbb{G}, g, q) \leftarrow \mathscr{G}(1^\kappa), k \leftarrow \mathbb{G}$.

Theorem 1: Assuming that the DDH problem is hard relative to $\mathscr{G}$, the Fudged DHKE protocol securely computes $f^{\mathscr{G}}_{\mathsf{FDHKE}}$ among three parties in the presence of a bounded, static, semi-honest adversary who corrupts P_3.

Proof. We will begin by constructing a simulator Sim, which is well defined only for inputs of the form $(1^\kappa, \{3\}, \{1^\kappa\}, \{(\mathbb{G}, g, q)\})$ such that $(\mathbb{G}, g, q) \in \mathrm{image}(\mathscr{G}(1^\kappa))$.

Notice that P_3 behaves deterministically, therefore Sim does not need to simulate the random coins of P_3.

Sim constructs the view of P_3 by sampling $a, b \leftarrow \mathbb{Z}_q$ and computing $A := g^a$ and $B := g^b$. The message from P_1 to P_3 is set to $(\mathbb{G}, g, q, A)$, and the message from P_2 to P_3 is set to B.

Security for the (Fudged) DHKE Protocol

Sim constructs the view of P_3 by sampling $a, b \leftarrow \mathbb{Z}_q$ and computing $A := g^a$ and $B := g^b$. The message from P_1 to P_3 is set to $(\mathbb{G}, g, q, A)$, and the message from P_2 to P_3 is set to B.

Next, assuming a PPT $\mathscr{D}$ that distinguishes the protocol distribution from the simulated distribution, we will construct $\mathscr{D}'$ that distinguishes $(\mathbb{G}, g, q, g^a, g^b, g^{a \cdot b})$ from $(\mathbb{G}, g, q, g^a, g^b, g^c)$ where $(\mathbb{G}, g, q) \leftarrow \mathscr{G}(1^\kappa)$ and $a, b, c \leftarrow \mathbb{Z}_q$.

$\mathscr{D}'$ receives $(1^\kappa, (\mathbb{G}, g, q, A, B, C))$ as input in the DDH game. It then begins to emulate an instance of the Fudged DHKE protocol "in its head" (i.e. as a subroutine), in which P_3 is semi-honestly corrupted.

In this instance of the Fudged DHKE protocol, $\mathscr{D}'$ sets the message from P_1 to be $(\mathbb{G}, g, q, A)$ and the message from P_2 to be B. The outputs of P_1 and P_2 are set to C.

Security for the (Fudged) DHKE Protocol

$\mathscr{D}'$ receives $(1^\kappa, (\mathbb{G}, g, q, A, B, C))$ as input in the DDH game. It then begins to emulate an instance of the Fudged DHKE protocol "in its head" (i.e. as a subroutine), in which P_3 is semi-honestly corrupted.

In this instance of the Fudged DHKE protocol, $\mathscr{D}'$ sets the message from P_1 to be $(\mathbb{G}, g, q, A)$ and the message from P_2 to be B. The outputs of P_1 and P_2 are set to C.

Next, $\mathscr{D}'$ invokes $\mathscr{D}$ as a subroutine on an input tuple comprising 1^κ, the view of P_3, and the outputs of all three parties in the emulated instance Fudged DHKE.

Finally, $\mathscr{D}'$ outputs whatever $\mathscr{D}$ does.

Notice that if $\mathscr{D}'$ is playing the game $\mathsf{DDHGame}_{\mathscr{G},\mathscr{D}',0}(\kappa)$, then $\mathscr{D}$ receives an input that is *identically distributed* to its input in the real-world experiment for the Fudged DHKE protocol. Similarly, if $\mathscr{D}'$ is playing the game $\mathsf{DDHGame}_{\mathscr{G},\mathscr{D}',1}(\kappa)$, then $\mathscr{D}$ receives an input that is *identically distributed* to its input in the ideal-world experiment for the Fudged DHKE protocol.

Security for the (Fudged) DHKE Protocol

Notice that if $\mathscr{D}'$ is playing the game $\mathsf{DDHGame}_{\mathscr{G},\mathscr{D}',0}(\kappa)$, then $\mathscr{D}$ receives an input that is *identically distributed* to its input in the real-world experiment for the Fudged DHKE protocol. Similarly, if $\mathscr{D}'$ is playing the game $\mathsf{DDHGame}_{\mathscr{G},\mathscr{D}',1}(\kappa)$, then $\mathscr{D}$ receives an input that is *identically distributed* to its input in the ideal-world experiment for the Fudged DHKE protocol.

Since $\mathscr{D}'$ outputs whatever $\mathscr{D}$ does, it follows that the advantage of $\mathscr{D}'$ is exactly the same as the advantage of $\mathscr{D}$. In other words, if we let

$$\delta(\kappa) = \left| \begin{array}{r} \Pr\left[\mathscr{D}\left(1^\kappa, \left(\mathsf{Sim}\left(1^\kappa, \{3\}, \{1^\kappa\}, \{z_3\}\right), \vec{z}\right)\right) = 1 : z \leftarrow f^{\mathscr{G}}_{\mathsf{FDHKE}}(1^\kappa, 1^\kappa, 1^\kappa)\right] \\ - \Pr\left[\mathscr{D}\left(1^\kappa, \left(\{\mathsf{VIEW}_3\}, \mathsf{OUTPUT}_{\mathsf{FDHKE}}\right)\right) = 1\right] \end{array} \right|$$

then we have $\left|\Pr\left[\mathsf{DDHGame}_{\mathscr{G},\mathscr{D}',0}(\kappa) = 1\right] - \Pr\left[\mathsf{DDHGame}_{\mathscr{G},\mathscr{D}',1}(\kappa) = 1\right]\right| = \delta(\kappa)$.

Security for the (Fudged) DHKE Protocol

Since $\mathscr{D}'$ outputs whatever $\mathscr{D}$ does, it follows that the advantage of $\mathscr{D}'$ is exactly the same as the advantage of $\mathscr{D}$. In other words, if we let

$$\delta(\kappa) = \left| \begin{array}{r} \Pr\left[\mathscr{D}\left(1^\kappa, \left(\mathsf{Sim}\left(1^\kappa, \{3\}, \{1^\kappa\}, \{z_3\}\right), \vec{z}\right)\right) = 1 : z \leftarrow f^{\mathscr{G}}_{\mathsf{FDHKE}}(1^\kappa, 1^\kappa, 1^\kappa)\right] \\ - \Pr\left[\mathscr{D}\left(1^\kappa, \left(\{\mathsf{VIEW}_3\}, \mathsf{OUTPUT}_{\mathsf{FDHKE}}\right)\right) = 1\right] \end{array} \right|$$

then we have $\left|\Pr\left[\mathsf{DDHGame}_{\mathscr{G},\mathscr{D}',0}(\kappa) = 1\right] - \Pr\left[\mathsf{DDHGame}_{\mathscr{G},\mathscr{D}',1}(\kappa) = 1\right]\right| = \delta(\kappa)$.

Next, notice that $\mathscr{D}'$ invokes $\mathscr{D}$ exactly once with security parameter 1^κ, and emulates the Fudged DHKE protocol exactly once with security parameter 1^κ.

Since we know that the protocol is PPT, $\mathscr{D}'$ is PPT if and only if $\mathscr{D}$ is.

Security for the (Fudged) DHKE Protocol

Next, notice that $\mathscr{D}'$ invokes $\mathscr{D}$ exactly once with security parameter 1^κ, and emulates the Fudged DHKE protocol exactly once with security parameter 1^κ.

Since we know that the protocol is PPT, $\mathscr{D}'$ is PPT if and only if $\mathscr{D}$ is.

It remains to observe that if $\mathscr{D}$ breaks the security of the DHKE protocol, then by definition $\delta(\kappa)$ is non-negligible (i.e. there are infinitely many values of κ such that $\delta(\kappa)$ is at least inverse-polynomial). It follows that $\mathscr{D}'$ falsifies the DDH assumption.

Therefore, if the DDH assumption is true, then it must be the case that the Fudged DHKE protocol securely computes $f^{\mathscr{G}}_{\mathsf{FDHKE}}$. ∎

Another Useful Little Lemma

Lemma 1: Let $(\mathbb{G}, \star)$ be a finite group and let $m \in \mathbb{G}$ be an arbitrary element. If $k \leftarrow \mathbb{G}$ is uniformly distributed, then $k' := k \star m$ is also uniformly distributed.

Proof:

- It's sufficient to show that for every $g \in \mathbb{G}$ we have $\Pr[k \star m = g] = \frac{1}{|\mathbb{G}|}$.
- If we fix $g \in \mathbb{G}$ arbitrarily, then $\Pr[k \star m = g] = \Pr[k = g \star m^{-1}]$.
- Since the right hand side is a fixed value and k is uniform, we have $\Pr[k = g \star m^{-1}] = \frac{1}{|\mathbb{G}|}$ ∎.

Note: We were using this lemma when we proved the security of OTP, with group $(\{0,1\}^{\ell}, \oplus)$.

Public-Key Encryption

- As we discussed, using $k \leftarrow \mathbb{G}$ in place of a binary key is a bit tricky, but if $m \in \mathbb{G}$ then it's very easy to convert DHKE into a *public-key encryption* scheme. Suppose we want to encrypt $m \in [0,\ell]$ for some constant ℓ: Alice and Bob can encode this value in $\mathbb{G}$ using $m' := g^m$.

receives input 1^κ

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$, $a \leftarrow \mathbb{Z}_q$, $A := g^a$

receives input $m' \in \mathbb{G}$

$(\mathbb{G}, g, q, A)$

$b \leftarrow \mathbb{Z}_q, B := g^b, C := A^b \cdot m'$

(B, C)

output $m' := C \cdot B^{-a}$

- Lemma 1 tells us that (A, B, C) is computationally indistinguishable from $(A, B, R) : R \leftarrow \mathbb{G}$. We have no formal idea of computational security for encryption, yet, but this is intuitive.
- We'll say that $\mathsf{pk} := (\mathbb{G}, g, q, A)$ is the *public key* that Bob can use to encrypt, and $\mathsf{sk} := a$ is the *secret key* that Alice can use to decrypt. This scheme is called *ElGamal Encryption.*

Multi-Sender, Multi-Message Security?

receives input $m_2' \in \mathbb{G}$

receives input 1^κ

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa), a \leftarrow \mathbb{Z}_q, A := g^a$

receives input $m_1' \in \mathbb{G}$

$(\mathbb{G}, g, q, A)$

$(\mathbb{G}, g, q, A)$

$b_2 \leftarrow \mathbb{Z}_q, B_2 := g^{b_2}, C_2 := A^{b_2} \cdot m_2'$

(B_2, C_2)

(B_1, C_1)

$b_1 \leftarrow \mathbb{Z}_q, B_1 := g^{b_1}, C_1 := A^{b_1} \cdot m_1'$

output $m_1' := C_1 \cdot B_1^{-a}$

output $m_2' := C_2 \cdot B_2^{-a}$

- Suppose that Alice sends her public key to more than one person. If Carol *also* uses pk to send a message to Alice, does this break security for Bob's message?
- In the DHKE protocol the *adversary* learned $(\mathbb{G}, g, q, A) = \mathsf{pk}$, which means it could do *exactly* what Carol is doing here, yet we proved that B^a was indistinguishable uniform. Therefore, it seems intuitive that this is not a problem.

Multi-Sender, Multi-Message Security?

receives input $m_2' \in \mathbb{G}$

receives input 1^κ

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa), a \leftarrow \mathbb{Z}_q, A := g^a$

receives input $m_1' \in \mathbb{G}$

$(\mathbb{G}, g, q, A)$

$(\mathbb{G}, g, q, A)$

$b_2 \leftarrow \mathbb{Z}_q, B_2 := g^{b_2}, C_2 := A^{b_2} \cdot m_2'$

(B_2, C_2)

(B_1, C_1)

$b_1 \leftarrow \mathbb{Z}_q, B_1 := g^{b_1}, C_1 := A^{b_1} \cdot m_1'$

output $m_1' := C_1 \cdot B_1^{-a}$

output $m_2' := C_2 \cdot B_2^{-a}$

- Moreover, it seems intuitive to say that if Carol can safely send a second message to Alice, then Bob can send a second (or third) message too! So it seems the total message length is independent of the key length (apart from the polynomial bound implied by Bob's runtime)!
- When we talked about perfect encryption and one time pad, we showed the a key could *never* be reused without compromising security. Here, on the other hand, it seems that Alice's public key can be reused many times. We need a new notion of security for encryption!

Chosen Plaintext Attacks

- When modeling this kind of multi-message security game, we want to ensure that security holds even when the adversary gets to see encryptions of the *most advantageous* messages. To guarantee this, we'll let the adversary choose which messages are encrypted!
- In the public key setting, giving the public key to the adversary is sufficient: it can sample encryptions itself, as many times as it likes.
- After $\mathscr{A}$ has seen however many messages it wants, it can choose two messages that the game will encrypt. A distinguisher will then work with $\mathscr{A}$ to distinguish between the encryptions of these messages. $\mathscr{A}$ can give a hint to the distinguisher too!

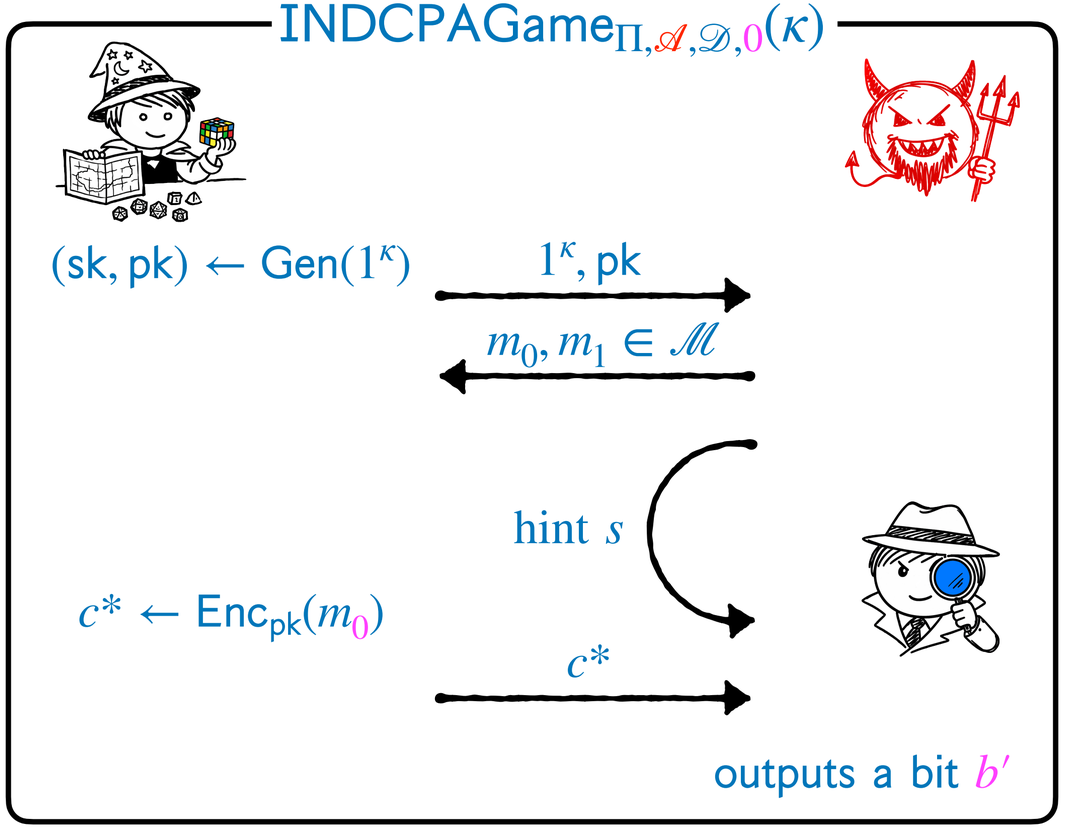
Definition 10: A public-key encryption scheme for message space $\mathscr{M}$ scheme is a trio of PPT algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)$ samples a key pair given a security parameter, $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$ samples a ciphertext c given $m \in \mathscr{M}$, and $m' := \mathsf{Dec}_{\mathsf{sk}}(c)$ decrypts.

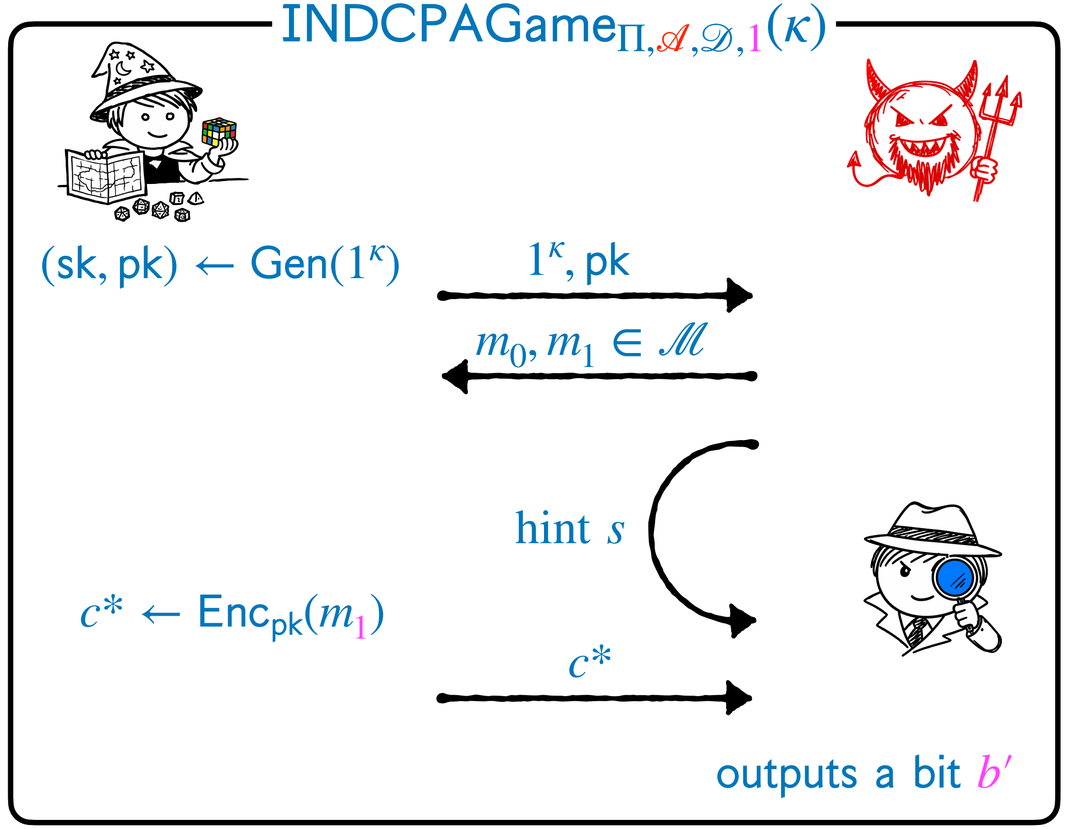
Definition 11: A PKE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *correct* if $\forall m \in \mathscr{M}$, $\forall (\mathsf{sk}, \mathsf{pk}) \in \mathrm{image}(\mathsf{Gen})$, $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m$.

Question: why can PKE *never* achieve security against an unbounded adversary?

Def 12: The IND-CPA Game

- Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme for message space $\mathscr{M}$.
- The *Ciphertext Indistinguishability Chosen Plaintext Attack Game* $\mathsf{INDCPAGame}_{\Pi,\mathscr{A},\mathscr{D},b}(\kappa)$ for $b \in \{0,1\}$ is played between a challenger and an adversary/distinguisher team. The game's output is the output of the distinguisher.



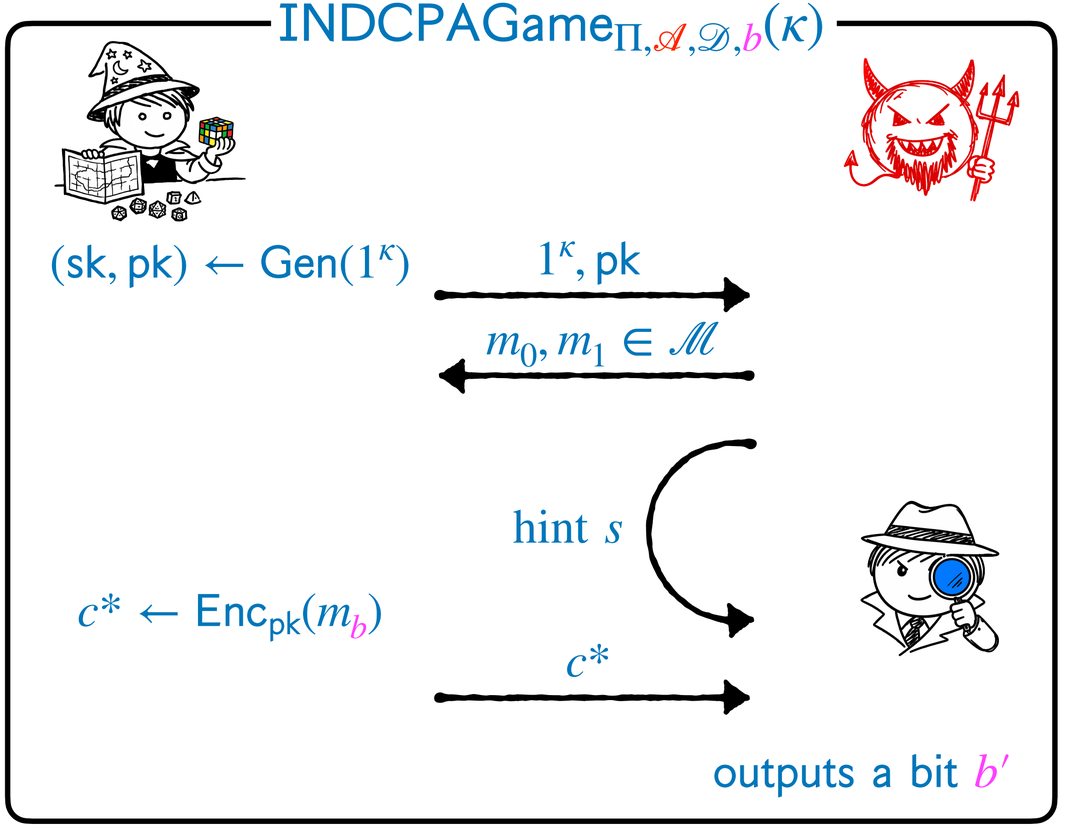


IND-CPA Security Definition

Definition 13 (IND-CPA Security for PKE): We say that PKE scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has *indistinguishable ciphertexts under Chosen Plaintext attacks* if $\forall$ PPT $(\mathcal{A}, \mathcal{D})$ $\exists$ a negligible function ε such that $\forall \kappa \in \mathbb{N}$,

$$\left| \Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},0}(\kappa) = 1\right] - \Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},1}(\kappa) = 1\right] \right| \leq \varepsilon(\kappa)$$

Note: we can define IND-CPA security analogously for *symmetric* encryption, but in that case, there is no public key to give to $\mathcal{A}$. Instead $\mathcal{A}$ can repeatedly request that the challenger sign a message using the secret key, and the challenger will do so.



ElGamal Encryption

Definition 14 (ElGamal Encryption for $\mathcal{M} = [0,\ell]$, for some constant ℓ):
Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q > \ell$, $|q| \geq \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (written multiplicatively). The *ElGamal Encryption Scheme* comprises the following algorithms:

- $\mathsf{Gen}(1^\kappa)$ outputs $(\mathsf{sk}, \mathsf{pk})$ where $(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$, $\mathsf{sk} \leftarrow \mathbb{Z}_q$, $X := g^{\mathsf{sk}}$, $\mathsf{pk} := (\mathbb{G}, g, q, X)$.
- $\mathsf{Enc}_{\mathsf{pk}}(m)$ outputs $c := (R, C)$ where $r \leftarrow \mathbb{Z}_q$, $R := g^r$, $C := X^r \cdot g^m$.
- $\mathsf{Dec}_{\mathsf{sk}}(c = (R, C))$ outputs $m' \in [0,\ell]$ such that $g^{m'} = C \cdot R^{-x}$.

Notice: The decryption algorithm has to break discrete logarithm! This is fine so long as ℓ is a constant, but if ℓ grows with the security parameter, then clearly it won't work.

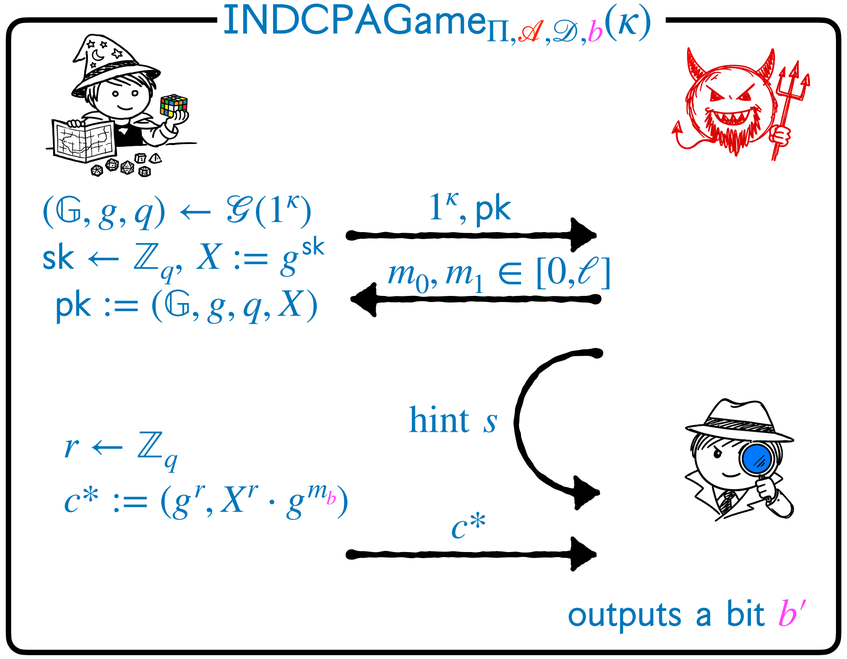
Theorem 2: If the DDH Problem is hard relative to $\mathcal{G}$, then the ElGamal encryption scheme is IND-CPA secure.

ElGamal Encryption, Security Proof

Theorem 2: If the DDH Problem is hard relative to $\mathcal{G}$, then the ElGamal encryption scheme is IND-CPA secure.

Proof Sketch:

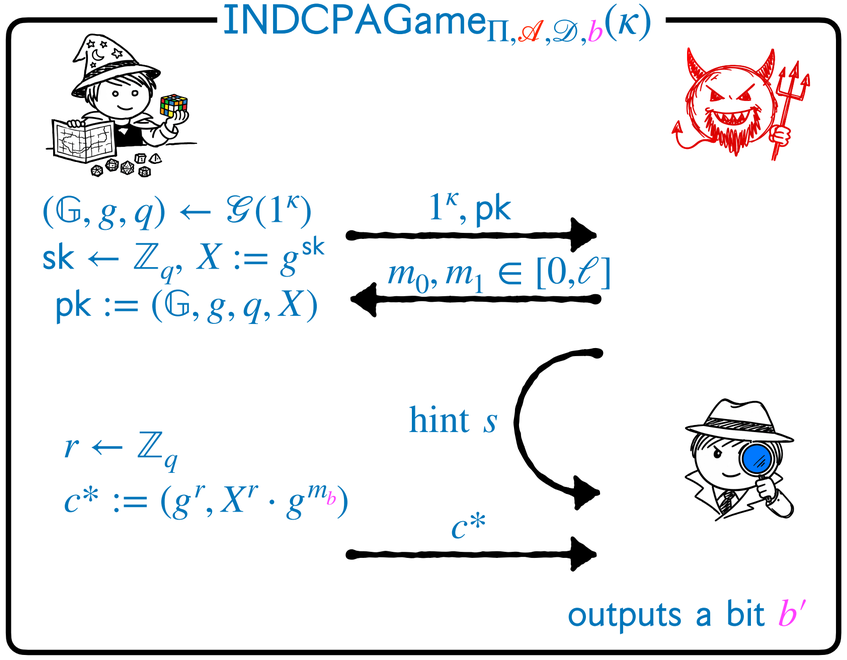
Just like we defined *hybrid experiments* that help us transform one experiment into another in a simulation-based security proof, we can also define *hybrid games* that help us transition between two games in a game-based proof. First, let's have a look at the IND-CPA game for ElGamal:

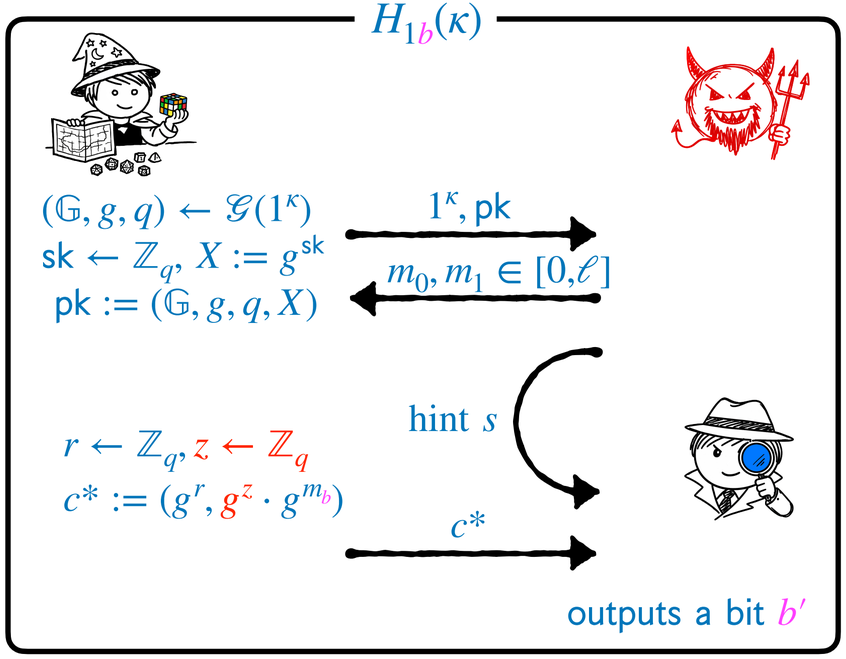


ElGamal Encryption, Security Proof

Proof Sketch:

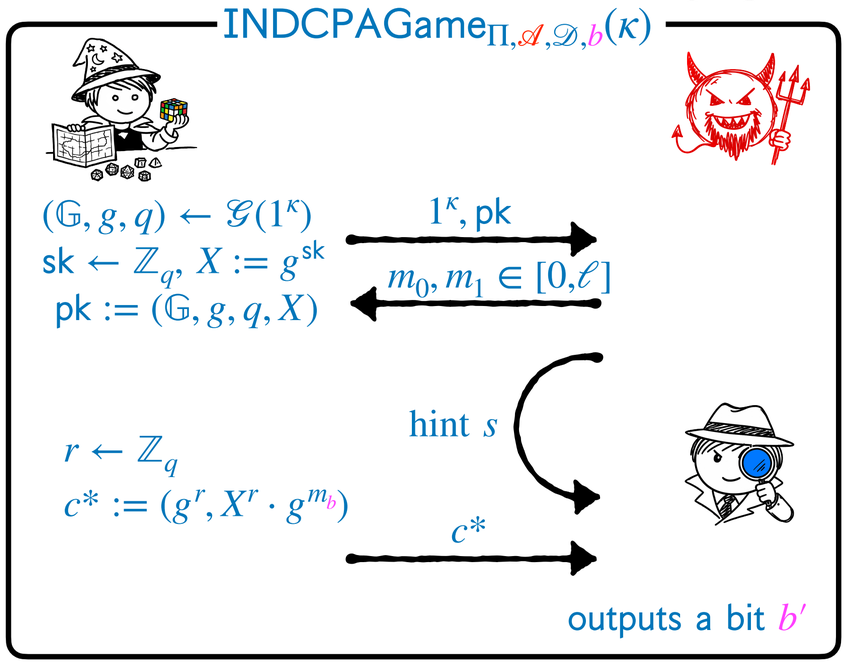
Just like we defined *hybrid experiments* that help us transform one experiment into another in a simulation-based security proof, we can also define *hybrid games* that help us transition between two games in a game-based proof. First, let's have a look at the IND-CPA game for ElGamal:

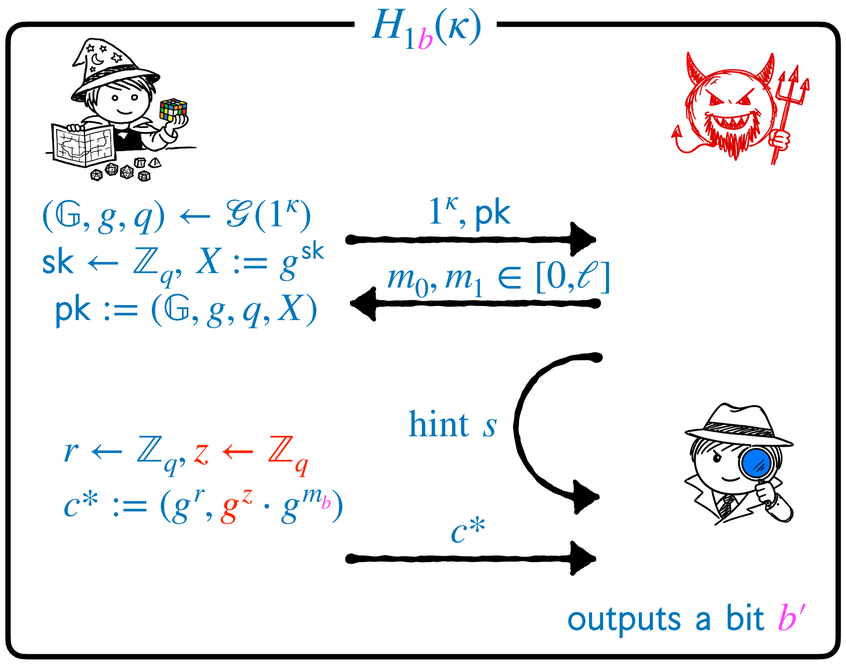




Our first hybrid game, $H_{1b}(\kappa)$, replaces X^r in c^* with g^z for some $z \leftarrow \mathbb{Z}_q$. If the DDH is hard wrt $\mathcal{G}$, then $\exists$ negligible ε such that $\left| \Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},b}(\kappa) = 1\right] - \Pr\left[H_{1b}(\kappa) = 1\right] \right| \leq \varepsilon(\kappa)$.

ElGamal Encryption, Security Proof



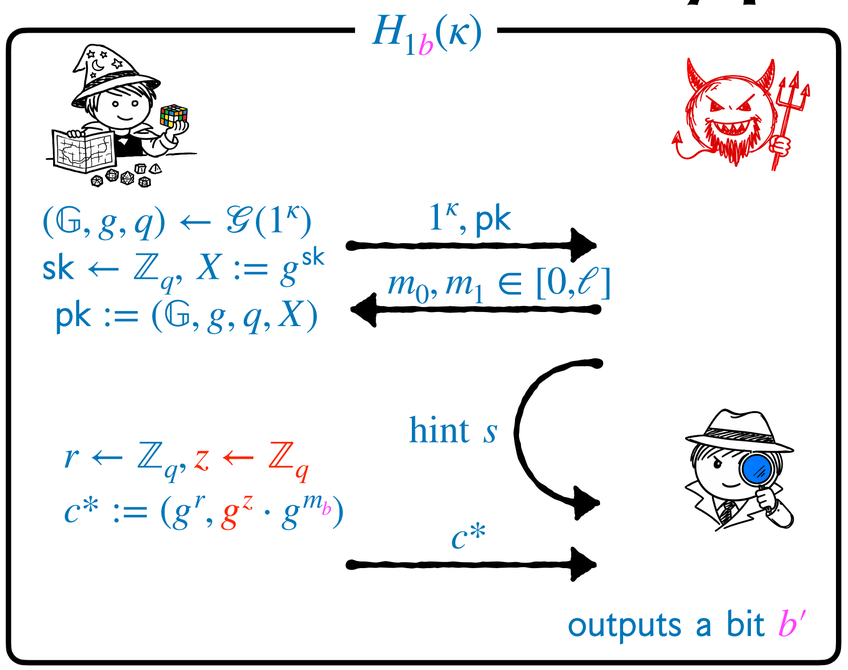


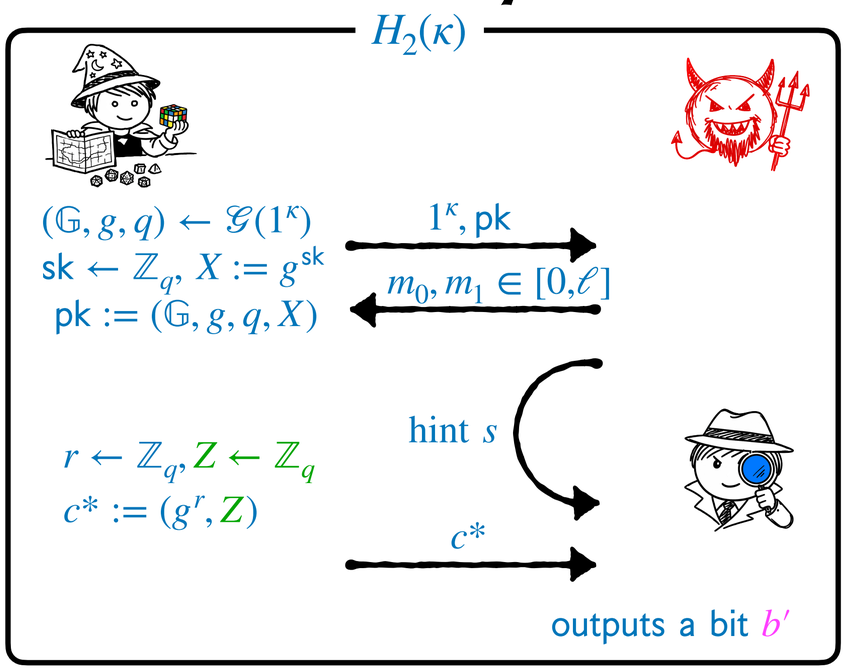
Our first hybrid game, $H_{1b}(\kappa)$, replaces X^r in c^* with g^z for some $z \leftarrow \mathbb{Z}_q$. If the DDH is hard wrt $\mathcal{G}$, then $\exists$ negligible ε such that $\left|\Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},b}(\kappa) = 1\right] - \Pr\left[H_{1b}(\kappa) = 1\right]\right| \leq \varepsilon(\kappa)$.

We can prove this by constructing a distinguisher $\mathcal{D}'$ that plays the DDH games. $\mathcal{D}'$ emulates either $\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},b}(\kappa)$ or $H_{1b}(\kappa)$ in its head using the tuple $(\mathbb{G}, g, q, X, R, Z)$ that it receives.

If $\mathcal{D}'$ plays $\mathsf{DDHGame}_{\mathcal{G},\mathcal{D}',0}(\kappa)$, then the view of $(\mathcal{A}, \mathcal{D})$ will be identical to $\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},b}(\kappa)$.
If $\mathcal{D}'$ plays $\mathsf{DDHGame}_{\mathcal{G},\mathcal{D}',1}(\kappa)$, then the view of $(\mathcal{A}, \mathcal{D})$ will be identical to $H_{1b}(\kappa)$.

ElGamal Encryption, Security Proof





Our next hybrid game, $H_2(\kappa)$, replaces $g^z \cdot g^{m_b}$ in c^* with $Z \leftarrow \mathbb{G}$.

By Lemma 1, which we proved earlier, the distributions of the view of $(\mathcal{A}, \mathcal{D})$ are identical in $H_{1b}(\kappa)$ and $H_2(\kappa)$. Therefore, it must hold that $\left| \Pr\left[H_{1b}(\kappa) = 1\right] - \Pr\left[H_2(\kappa) = 1\right] \right| = 0$.

ElGamal Encryption, Security Proof

Putting it together, we have

$$\left|\Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},0}(\kappa)=1\right]-\Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},1}(\kappa)=1\right]\right|$$

$$=\left|\begin{array}{l}\Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},0}(\kappa)=1\right]-\Pr\left[H_{10}(\kappa)=1\right]\\ +\Pr\left[H_{10}(\kappa)=1\right]-\Pr\left[H_{2}(\kappa)=1\right]+\Pr\left[H_{2}(\kappa)=1\right]-\Pr\left[H_{11}(\kappa)=1\right]\\ +\Pr\left[H_{11}(\kappa)=1\right]-\Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},1}(\kappa)=1\right]\end{array}\right|$$

Adding 0s.

$$\leq\left|\Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},0}(\kappa)=1\right]-\Pr\left[H_{10}(\kappa)=1\right]\right|$$
$$+\left|\Pr\left[H_{10}(\kappa)=1\right]-\Pr\left[H_{2}(\kappa)=1\right]\right|+\left|\Pr\left[H_{2}(\kappa)=1\right]-\Pr\left[H_{11}(\kappa)=1\right]\right|$$
$$+\left|\Pr\left[H_{11}(\kappa)=1\right]-\Pr\left[\mathsf{INDCPAGame}_{\Pi,\mathcal{A},\mathcal{D},1}(\kappa)=1\right]\right|$$

By the Triangle Inequality

$\leq \varepsilon(\kappa)+0+0+\varepsilon(\kappa)=2\varepsilon(\kappa)$ which is negligible if $\varepsilon(\kappa)$ is. ∎

Are you ready to see the last piece?

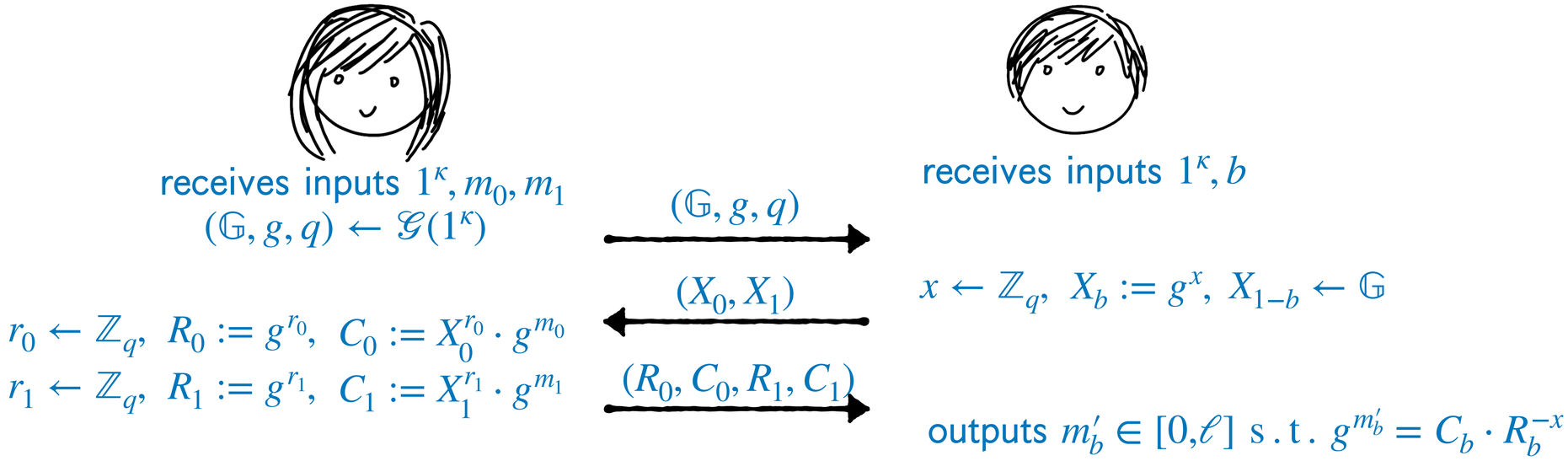
- Introducing Oblivious Transfer.
- Bounded Adversaries and Computational Security.
- The Cryptographic Toolkit: One-Way Functions.
- The Cryptographic Toolkit: Assumptions on Groups.
- A Simple Application: Key Agreement.
- Ensembles and Computational Indistinguishability.
- Computational Security for Protocols.
- The Cryptographic Toolkit: Public-key Encryption.
- Realizing Oblivious Transfer.

Final Recap: Introducing *Oblivious Transfer.*

- Let $\mathscr{M}$ be some message space, and $m_0, m_1 \in \mathscr{M}$ be two messages known to P_1.
- The *Oblivious Transfer* functionality allows P_2 to receive *one* message of its choice.
- P_1 isn't allowed to learn which of the two messages P_2 received.
- P_2 isn't allowed to learn anything about the message that it didn't choose.

Realizing Oblivious Transfer

- We will consider oblivious transfer for some message space $\mathscr{M} = [0,\ell]$ where ℓ is a constant. In particular, we will design a protocol that securely computes $f_{\mathsf{OT}}((m_0, m_1), b) = (\lambda, m_b)$.
- Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q > \ell$, $|q| \geq \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (denoted $\cdot$).



Theorem 3: if DDH is hard relative to $\mathcal{G}$, then the above protocol securely computes f_{OT} in the presence of a bounded semi-honest adversary that statically corrupts one party.

Proof: Homework!

CS4501 ~~Cryptographic Protocols~~
Intro to Cryptography Speedrun
Lecture 14: Public Key Encryption, CPA-Security, Oblivious Transfer

https://jackdoerner.net/teaching/#2026/Spring/CS4501