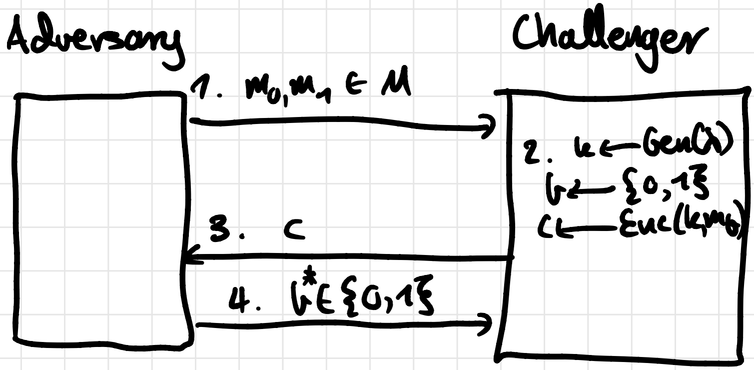
Last week, we discussed game-based security and its limitations. In particular

1) we get no composability guarantees
2) the security guarantee is hard to understand in real-world terms

Today we will see a different perspective on how to define security in a composition-friendly way: simulation.

IND-CPA

Let's review the IND-CPA game from last week; we will focus on the symmetric version.



This is a bit-guessing game, and we require that

$$\left| \Pr[G^{\lambda,s}_{IND\text{-}CPA}(A)] - \frac{1}{2} \right| \in negl(\lambda)$$

for all PPT adversaries A, where $G(A) = 1$ if $b = b^*$.

It turns out this is equivalent to a different formulation in terms of 2 games G_0 and G_1. In G_0, $c = Enc(k, m_0)$, and in G_1, $c = Enc(k, m_1)$. Instead of asking whether $b = b^*$, simply set $G_0(A)$ to be the value A outputs, b^*. Then the security statement is

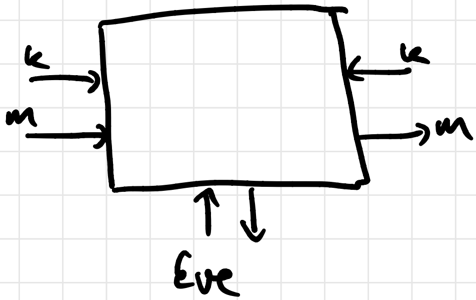
$$\left| \Pr[G_0 = 1] - \Pr[G_1 = 1] \right| \in negl(\lambda)$$

The idea is that the adversary shouldn't be able to tell which world it's in. For example, if it always outputs b in game G_b, then it can definitely tell apart the two ciphertexts! This is called the distinction game.

This multiple worlds idea is the basis of simulation security.

Let's zoom out a bit: we want to construct a secure channel between two parties. Given they share key k, they can send messages.

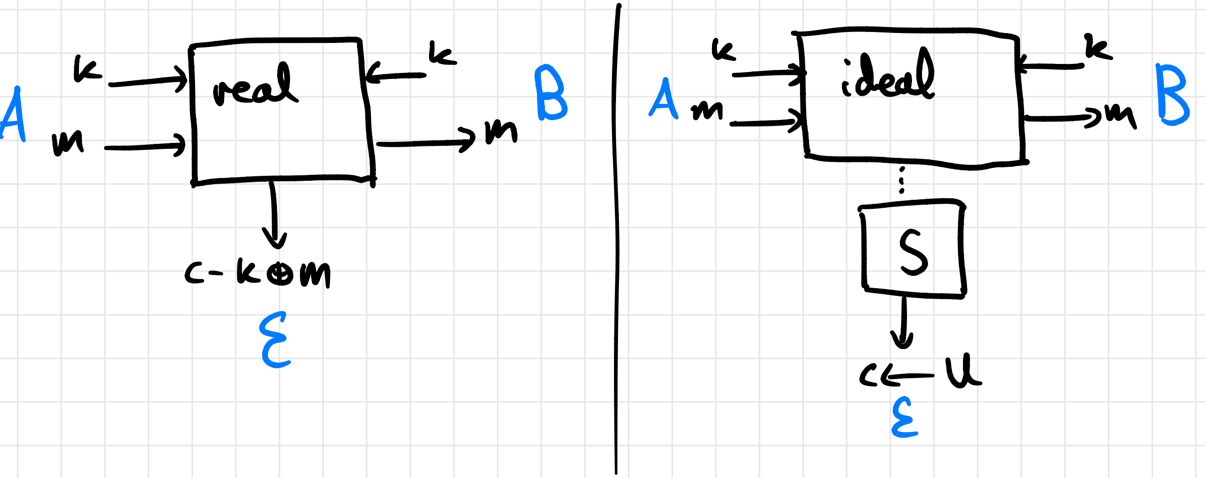
Of course, in the real world an adversary may be listening in.



<u>Question</u>: what does Eve see? In the real world, Eve sees $c = \mathrm{Enc}(k, m)$. We want to say that Eve cannot tell the real world apart from an ideal world where she gets no information.

If Eve sees <u>nothing</u> in the ideal world, she can easily tell them apart. We need to simulate the output Eve sees in the real world in such a way that Eve still gets no information about the message.

I draw 3 systems: real world, ideal world, and simulator.



The use of a simulator is no minor sleight-of-hand: it leads to some surprising impossibility results among other things (to be discussed in the future).

Our security statement is now:

$\forall$ PPT adversaries $\mathcal{A}$ $\exists$ a PPT simulator S such that the transcript of $\mathcal{A}$ interacting with the real world is indistinguishable from the transcript of S interacting with the ideal world.

This is a bit vague so let's try to be more explicit.

In the real world, the adversary $\mathcal{A}$ sees $c = k \oplus m$ where $k \leftarrow \{0,1\}^*$ is chosen uniformly at random. This is $\mathcal{A}$'s transcript.

In the ideal world, the adversary S sees c where $c \leftarrow \{0,1\}^*$ is chosen uniformly at random. This is $\mathcal{U}$'s transcript.

The security claim is: for all PPT algorithms Δ,

$$\left| \Pr[\Delta(k \oplus m) = 1] - \Pr[\Delta(c) = 1] \right| \in \mathrm{negl}(\lambda)$$

and we recover the statement from the distinction game. (In this case, the distributions are actually equal, so the LHS is 0.)

We say that π is a protocol representing the real world (Alice computes $c = k \oplus m$ and sends it to Bob, who computes $m = c \oplus k$), that the ideal world is a function f, and that π securely realises f.

(We still have not tried to make this precise for reasons that will become apparent.)

The "picture" is: the simulator S translates attacks on the real world to attacks on the ideal world. Since the ideal world is perfect by construction, if any real-world attack has a corresponding ideal-world attack*, the real world is secure.

* Except with negligible probability

This (finally!) gives us sequential composability: if π securely realises f, and φ uses f to securely realise g, then we have a chain

$$\{\} \xrightarrow{\pi} f \xrightarrow{\varphi} g$$

Unfortunately it does <u>not</u> give parallel composability: security may fall apart if the same protocol is executed several times concurrently. I give an intuitive explanation of an example to suggest a way to capture this requirement.

Without going into detail, a zero-knowledge proof involves a relation $R(x, w)$ where w is called a witness for x. For example, let $G = \langle g \rangle$ be a cyclic group, and

$$R = \{ (g^w, w) \mid w \in \mathbb{Z} \}$$

where w is a witness for discrete logarithms. The idea is a prover P can convince a verifier V that she knows w s.t. $R(x, w)$ holds, without revealing w to V.

We imagine a "puzzle system": P and V can generate puzzles p such that

1) P can solve all p
2) V cannot solve any p (nor verify a solution)

Let π be a ZKP protocol for some relation R. Consider π':

	$P: (x, w)$		$V: (x)$
1.		$\xleftrightarrow{\ \pi\ }$	
2.		$\xrightarrow{\ p\ }$	
3.		$\xleftarrow{\ s,\, p'\ }$	
4.	If s solves p:	$\xrightarrow{\ w\ }$	
	Else:	$\xrightarrow{\ s'\ }$	

π' is zero-knowledge alone because V can never find a solution, so P never reveals w.

However: imagine two instances of π' run concurrently. V waits to receive p_1, p_2, then sends (s, p_2) to the first instance. P responds with s_2, and then V can send (s_2, p_1) to P in order to learn w!

We need a stronger security definition that can guarantee secure constructions in a universal context.

The solution is UC, in which the global environment is explicitly modelled, including the ability for information to flow between protocol sessions.