One-Way Puzzles and the Foundations of Quantum Cryptography

Eli Goldin and Kabir Tomer

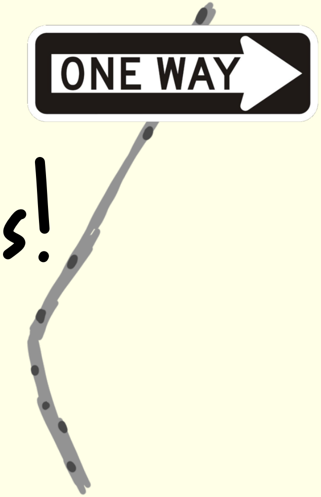
Minimal Cryptographic Primitive?

A question which we all know the answer to

What is the minimal primitive for classical cryptography

Minimal Cryptographic Primitive?

One-way functions!



Growing up we are told that the answer is one-way functions

Minimal Cryptographic Primitive?

One-way functions!

But Why?

But why do we say that one-way functions are minimal?

Minimal Cryptographic Primitive?

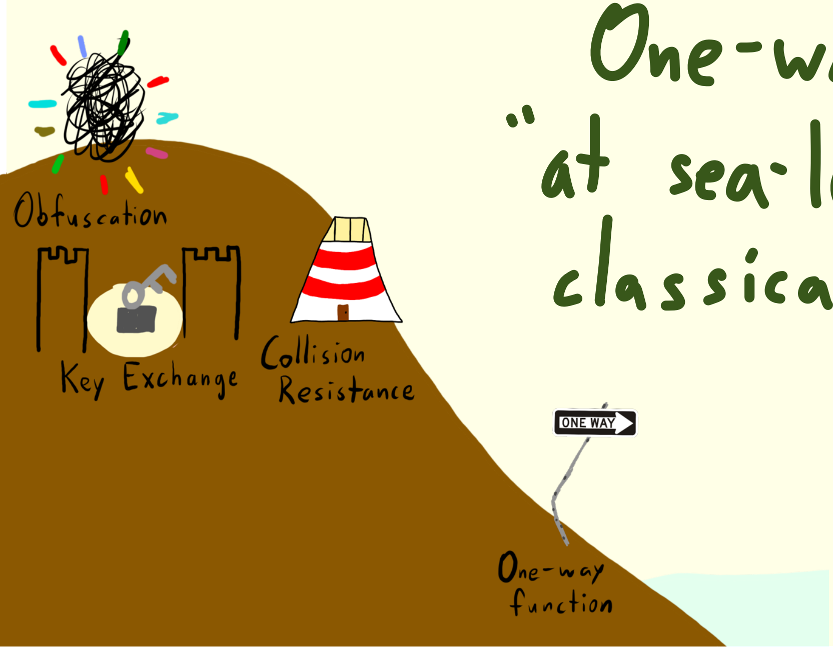
One-way functions!

But Why?

Implied by everything!

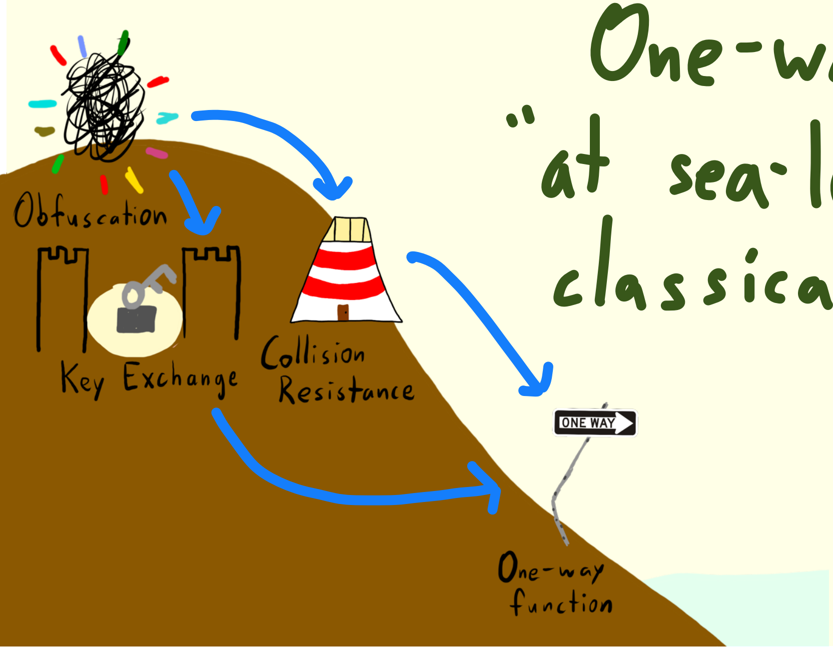
The primary reason is that one-way functions are implied by basically all classical cryptography

One-way functions "at sea-level" for classical crypto



Which is to say that that OWFs are in some sense “at sea level” for classical crypto

One-way functions "at sea-level" for classical crypto



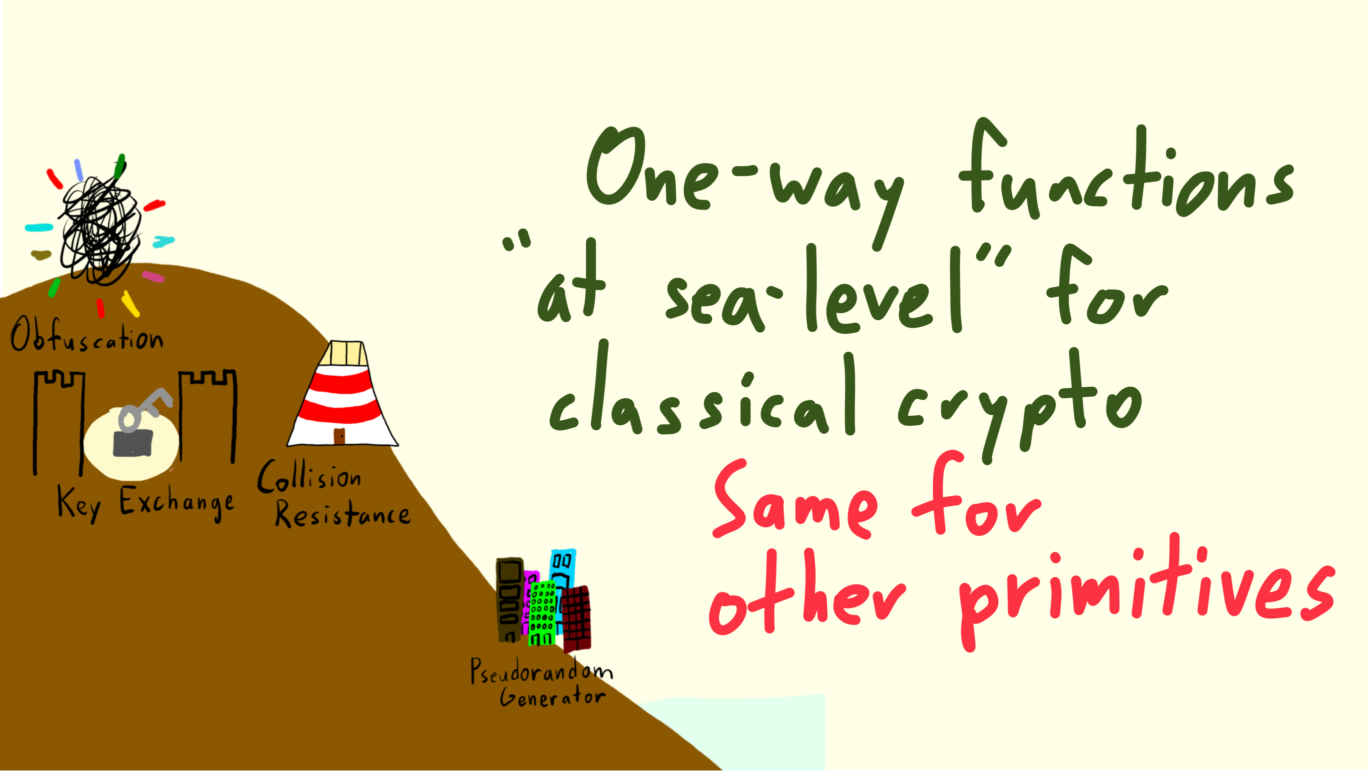
In the sense that they are implied by almost all classical crypto

One-way functions "at sea-level" for classical crypto

Same for other primitives

Obfuscation
Key Exchange
Collision Resistance
Commitment

Now, that is also true for other primitives which are equivalent to OWFs like commitments



One-way functions "at sea-level" for classical crypto

Same for other primitives

Or PRGs

One-way functions "at sea-level" for classical crypto
REALLY EASY
Obfuscation
Key Exchange
Collision Resistance
ONE WAY
One-way function

But what sets one-way functions apart is that they are also very easy to build. They seem to very naturally capture the kind of hardness inherent in cryptographic constructions

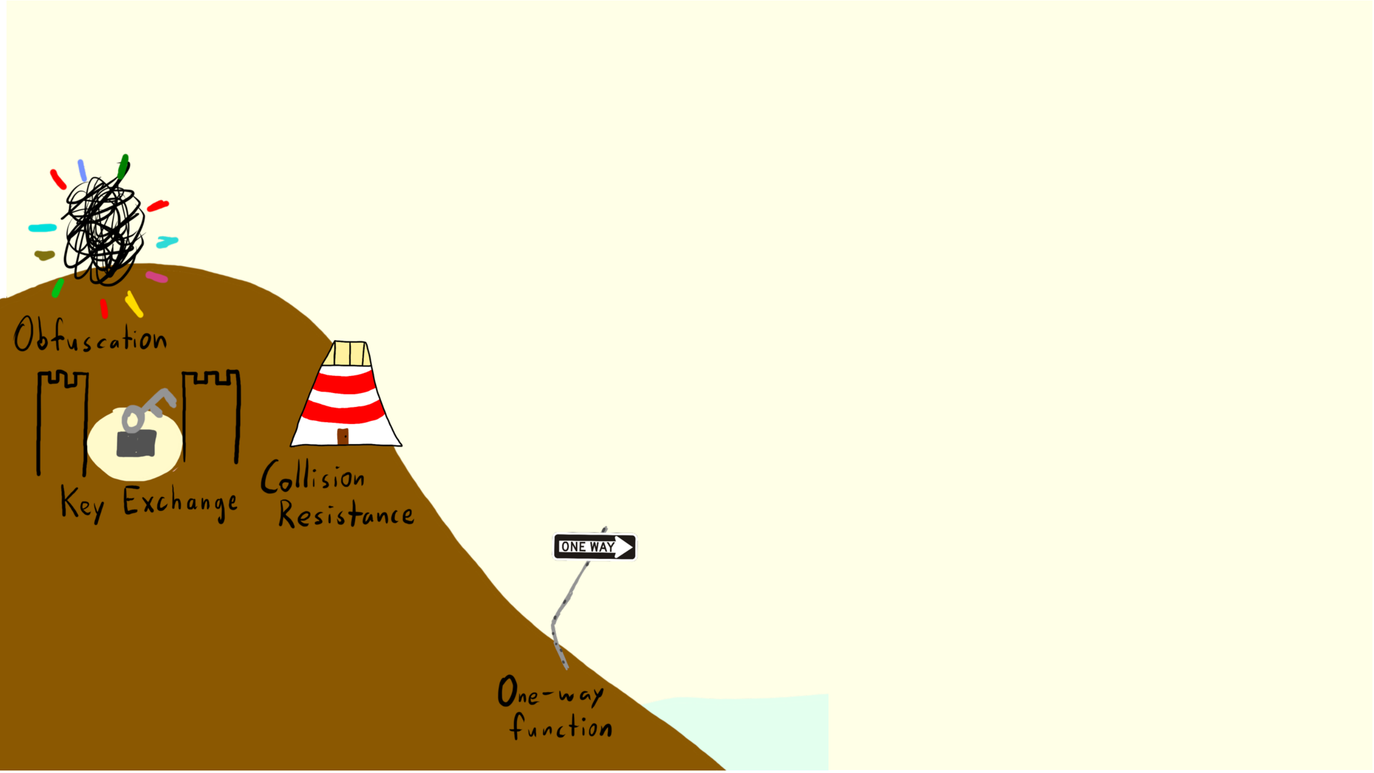
One Way Functions are
the perfect minimal assumption
on which to base cryptography

Which lets us conclude that one way functions are the perfect minimal assumption on which to base cryptography

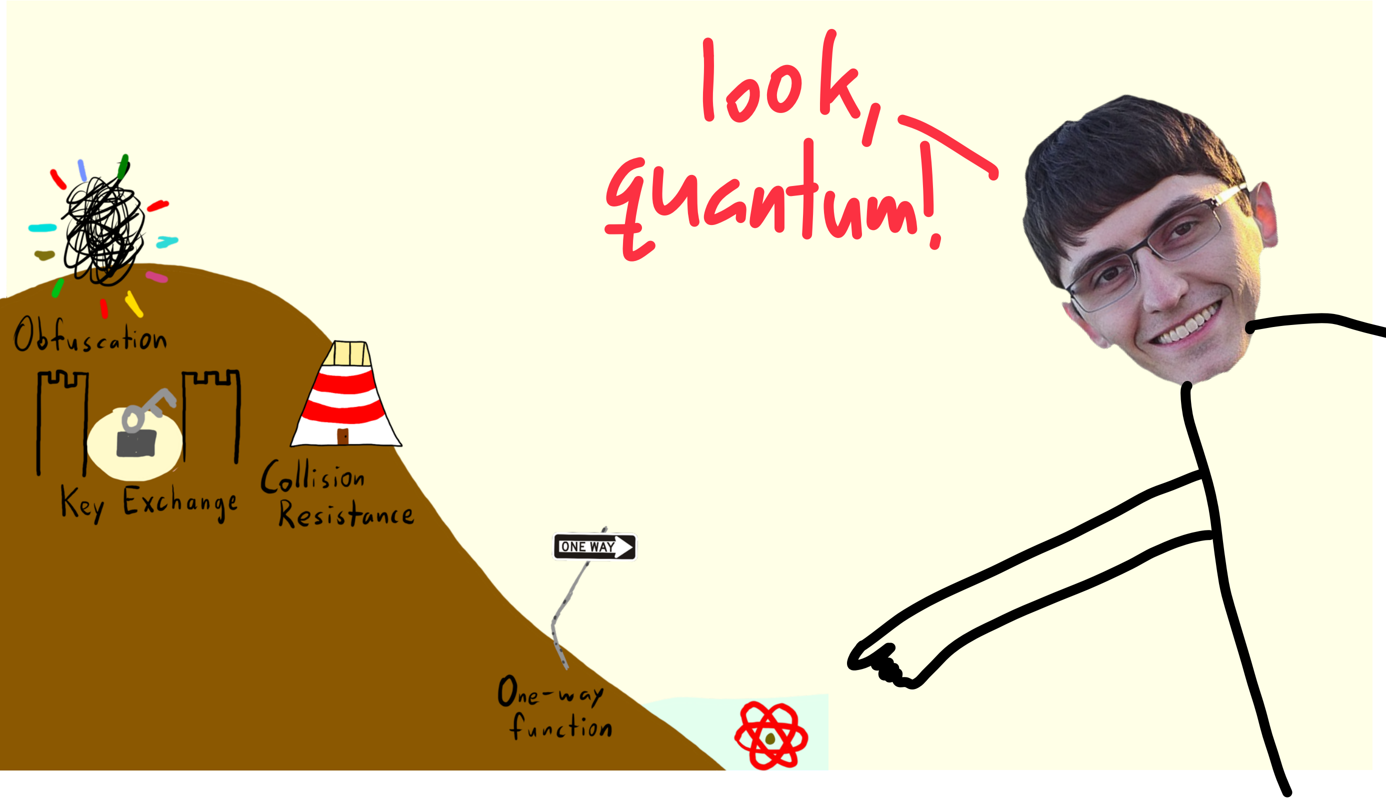
One Way Functions are the perfect minimal assumption which to base cryptography

NO

The problem with this claim though, as William kretschmer showed, this is only true for classical cryptography



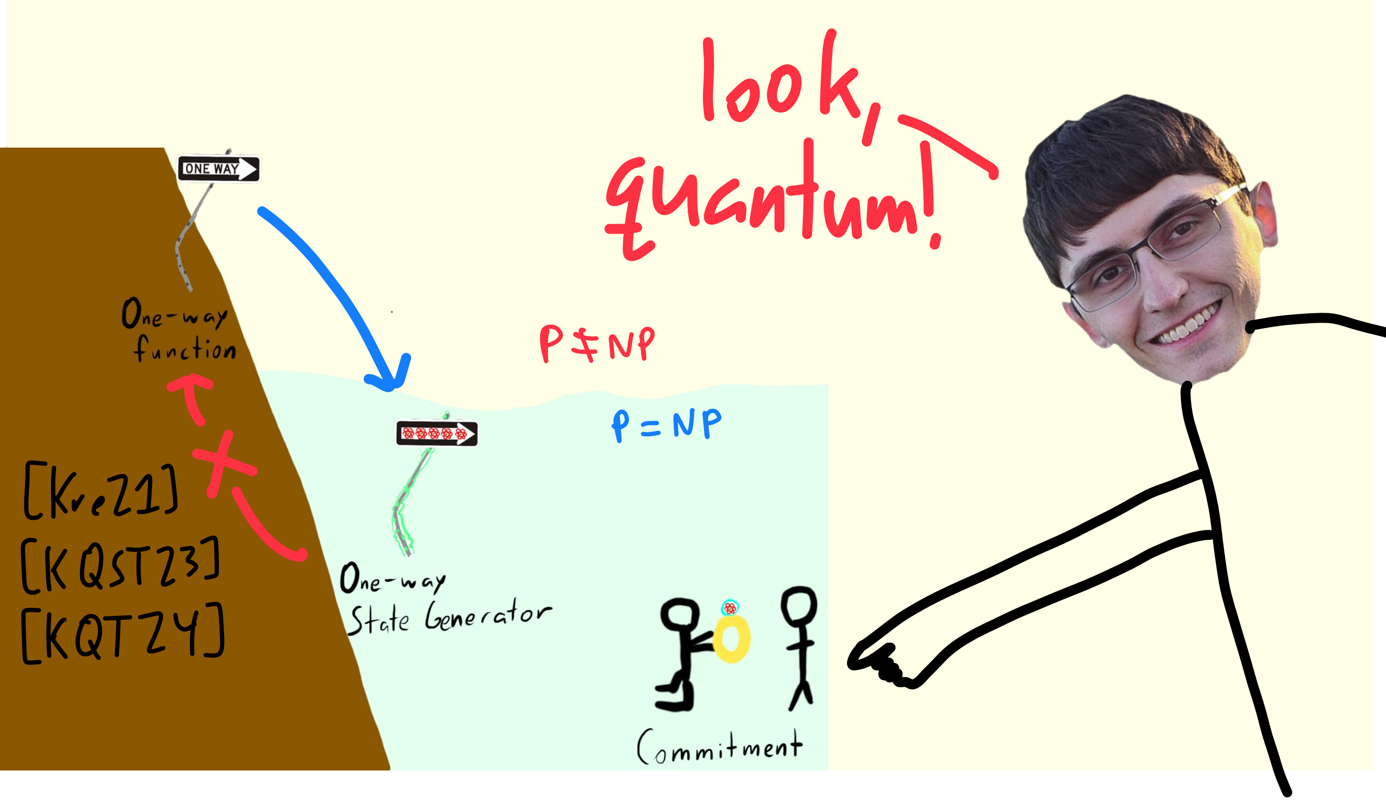
If we think of one way functions as “sea level”, then



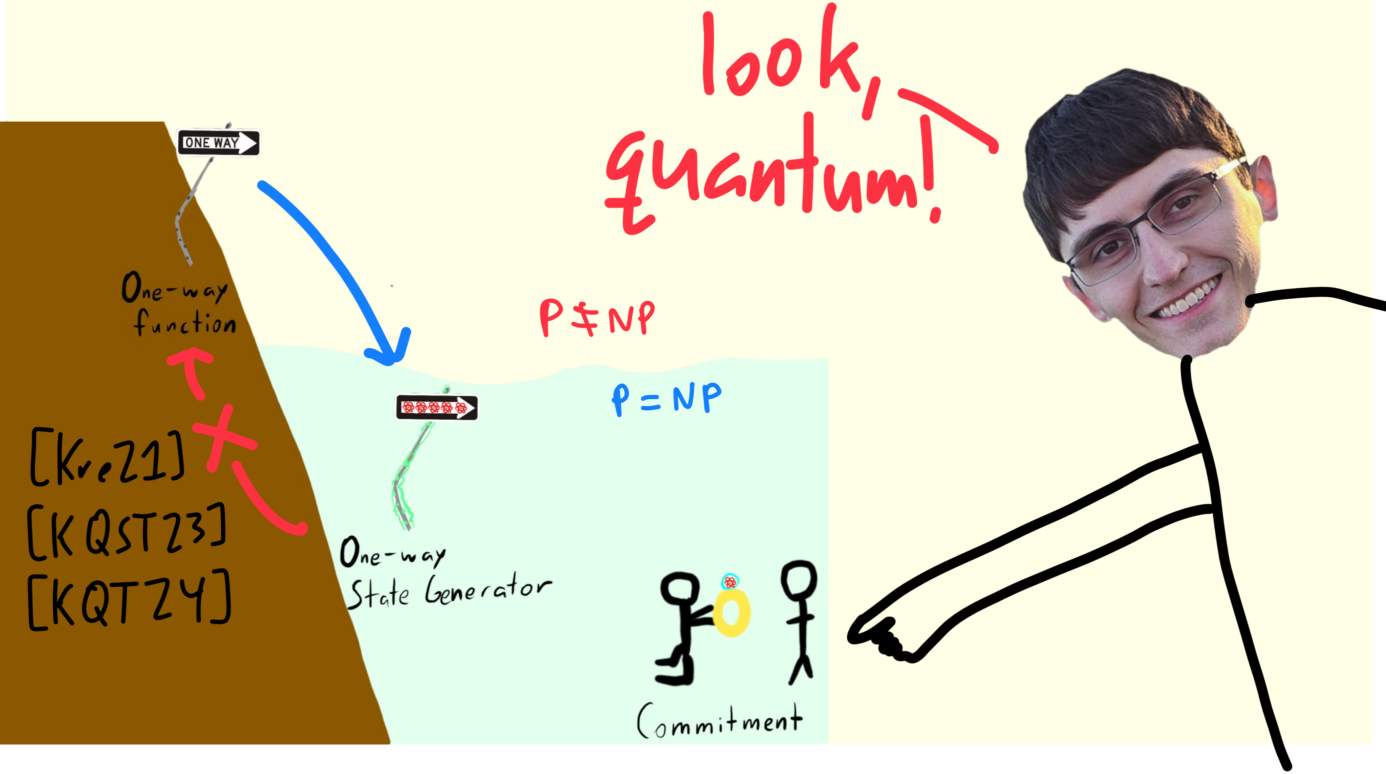
There are quantum cryptographic primitives which live “underwater”

look, quantum!
ONE WAY
One-way function
[Kre21]
One-way State Generator

Specifically, William and collaborators showed that certain kinds of quantum cryptography can be built from one way functions but do not in turn imply one-way functions



In fact, relative to certain oracles, these primitives can exist even if P=NP!



So, if quantum cryptography can be built from assumptions much weaker than one-way functions, this begs the question

2 years ago in the other room

Is there a quantum primitive which serves the same purpose as OWFs?

What *now* is the minimal assumption required for quantum cryptography? Or in other words, is there… This is the question that Mark talked about in the other room two years ago…

2 years ago in the other room

Is there a quantum primitive which serves the same purpose as OWFs?

What *now* is the minimal assumption required for quantum cryptography? Or in other words, is there… This is the question that Mark talked about in the other room two years ago…

2 years ago in the other room

	OWF	PRS	OWSG	EFI
Minimal	✗			
	✓			
	✓			
	✓			
	✓			

And he gave a list of properties that were desirable for minimality

2 years ago in the other room

	OWF	PRS	OWSG	EFI
Minimal	✗	?	?	?
	✓			
	✓			
	✓			
	✓			

And compared all the leading candidates for minimal assumption

2 years ago in the other room

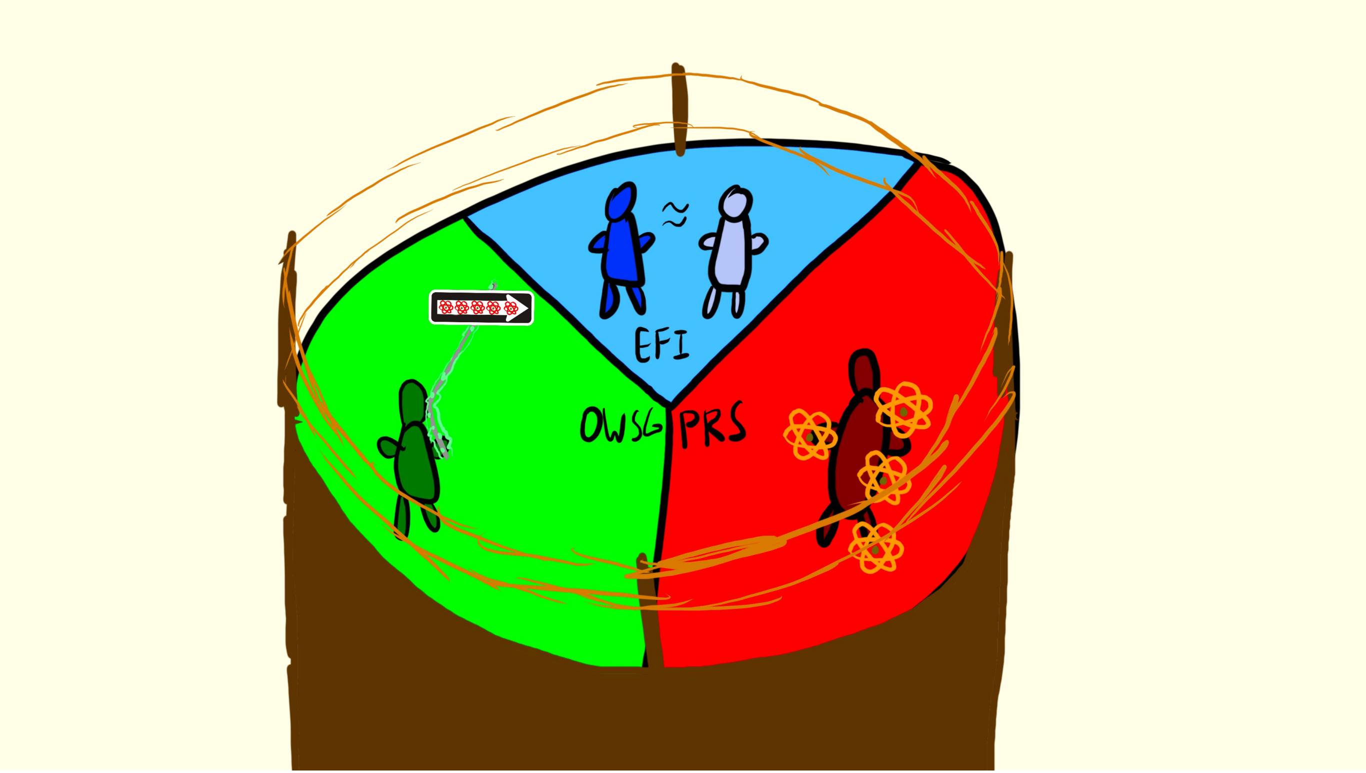
	OWF	PRS	OWSG	EFI
Minimal	✗	?	?	?
	✓	✗	✓	✗
	✓	✗	✗	?
	✓	✓	✗	?
	✓	?	?	✓

Along various axes,

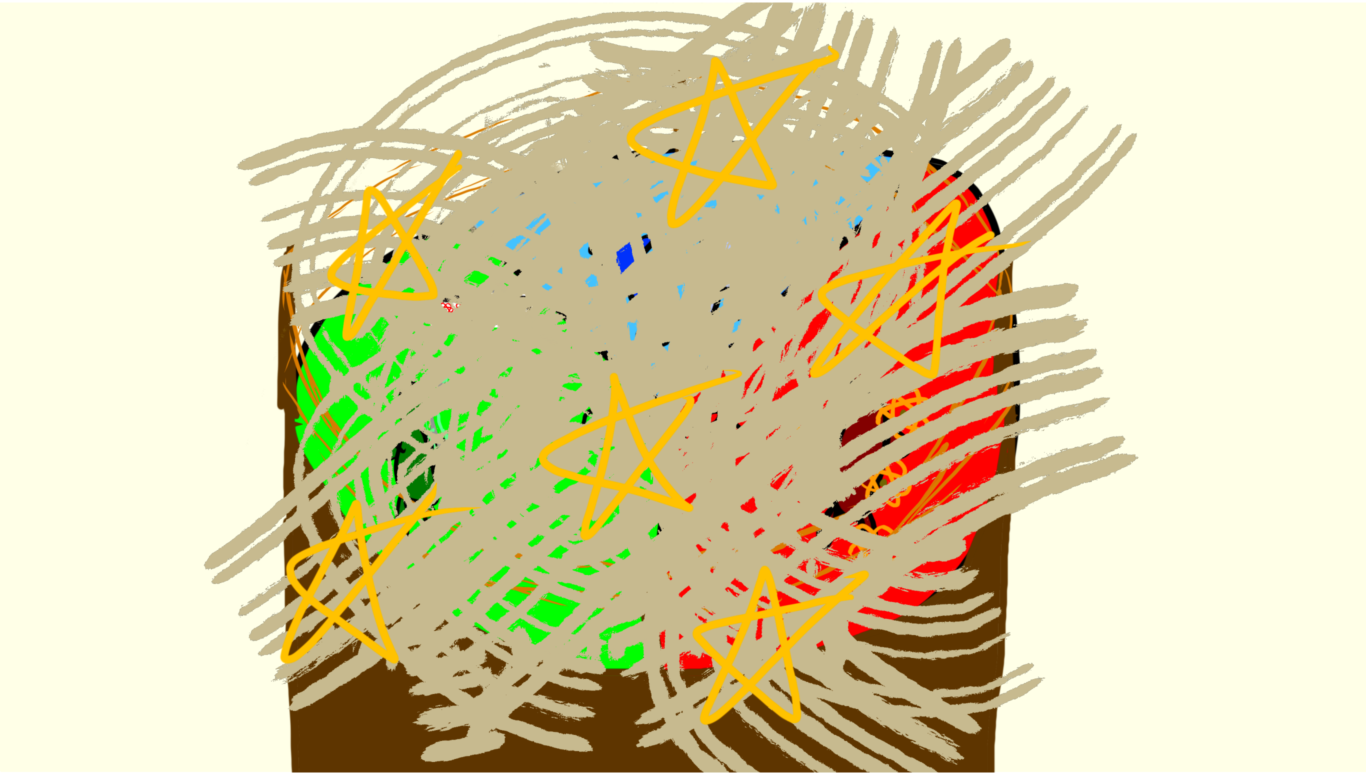
2 years ago in the other room

Possibility: maybe no good minimal quantum assumption?

And basically concluded that none of them really fit the description

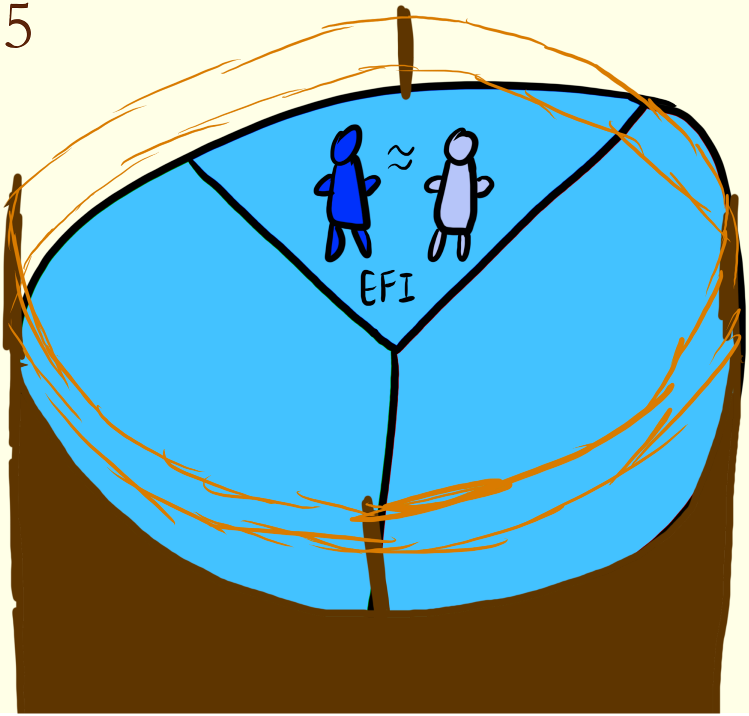


Two years later though, circa 2025, the picture is very different.



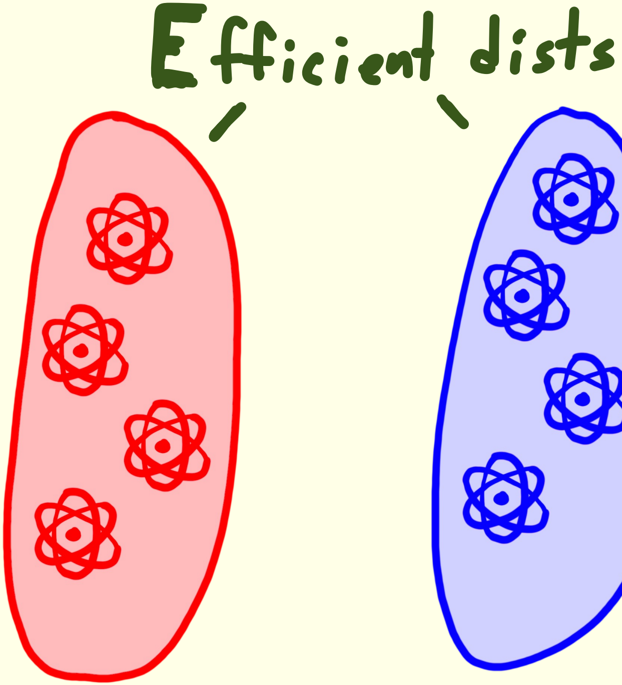
Because we have had these leading candidates for a while, and put them head to head, and after the dust has settled somewhat

Circa 2025



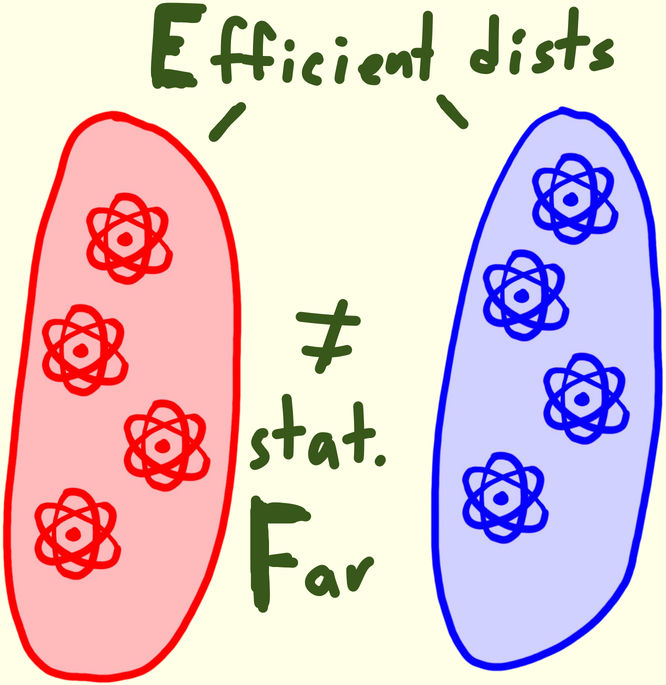
It seems pretty clear that a primitive called an EFI pair is the minimal primitive for quantum cryptography.

EFI

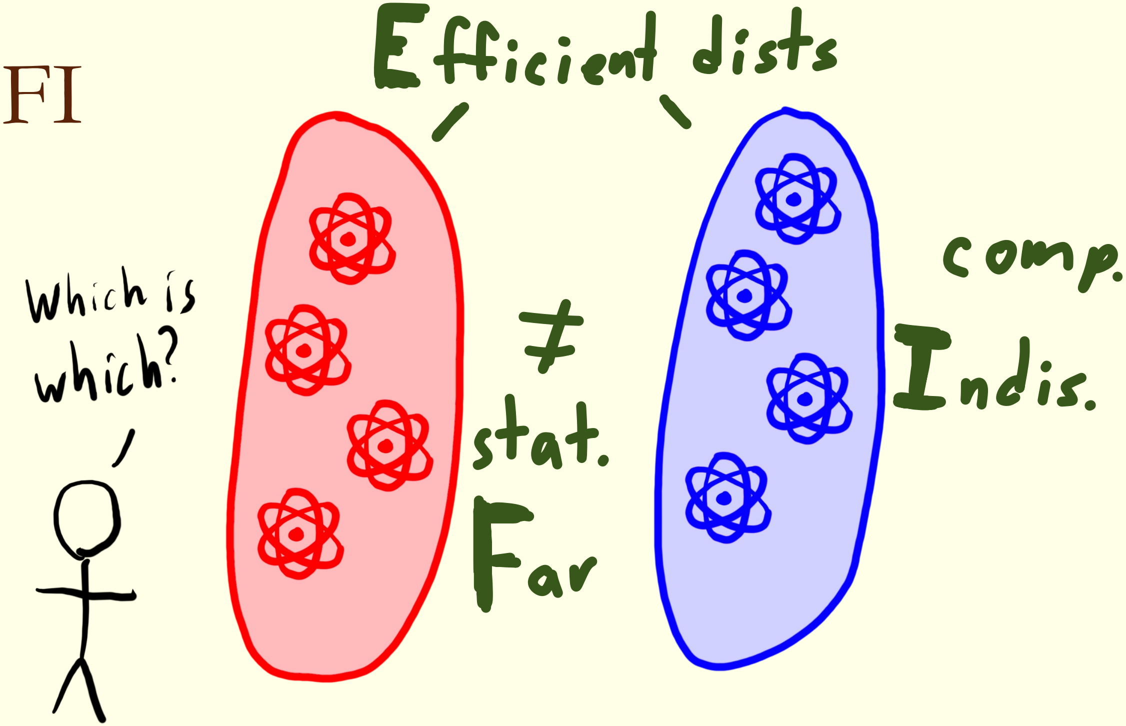


An EFI pair is just a pair of efficient distributions over quantum states

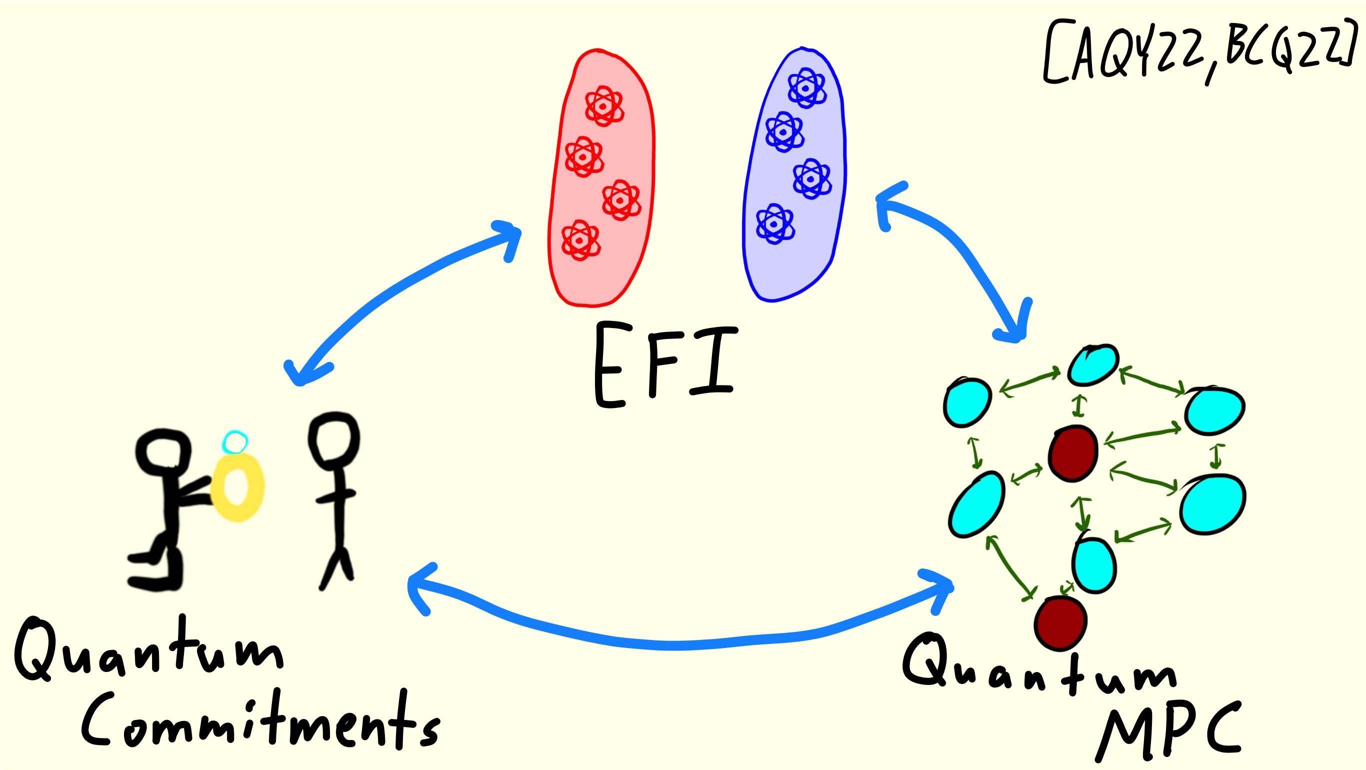
EFI



EFI



That are statistically far, but comp indistinguishable

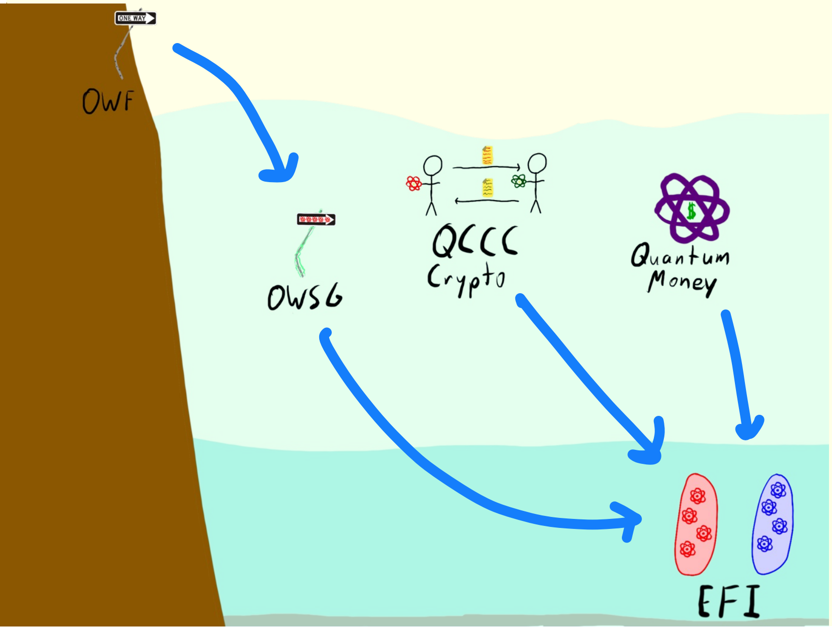


EFI pairs are pretty powerful, in that they can be used to build quantum commitments, which in turn can be used to build quantum MPC

How did we get here?

So how did we get here, So what has changed in these two years? why do we say that EFI pairs are minimal.

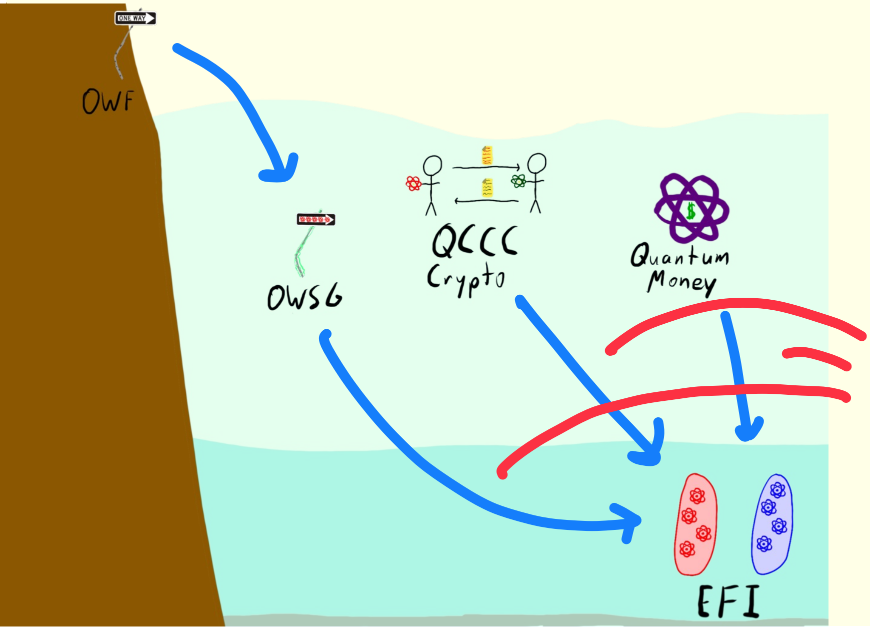
How did we get here?



EFI is really, really minimal!

Well, we found out that EFI pairs are implied by almost everything in quantum cryptography, which we cannot really say about any of the other candidates. And it turns out that really that is the most important requirement for minimality.

How did we get here?

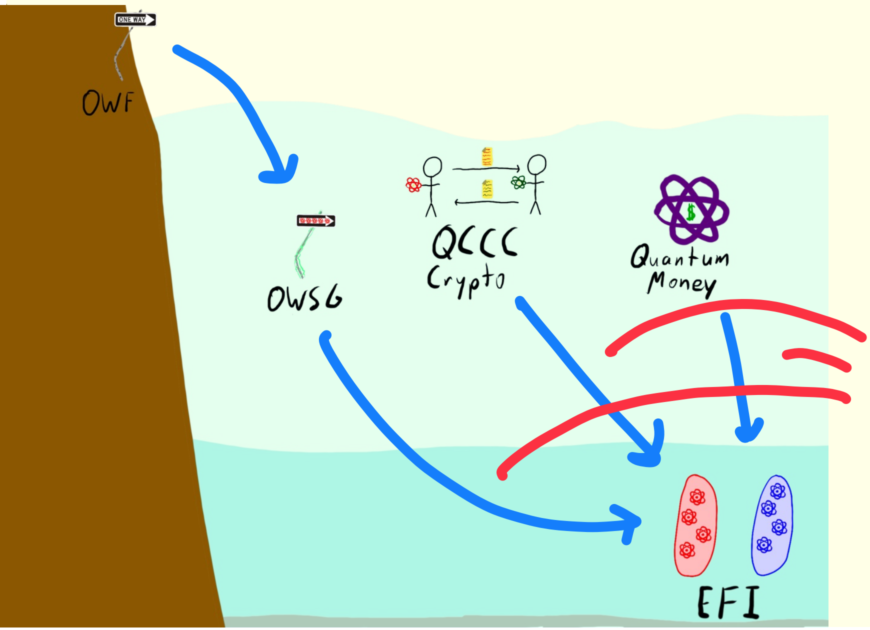


EFI is really, really minimal!

highly non-trivial!

There is a catch of sorts though, which is that it is not really easy to build EFI pairs from most quantum crypto primitives, especially from less structured primitives.

How did we get here?



EFI is really, really minimal!

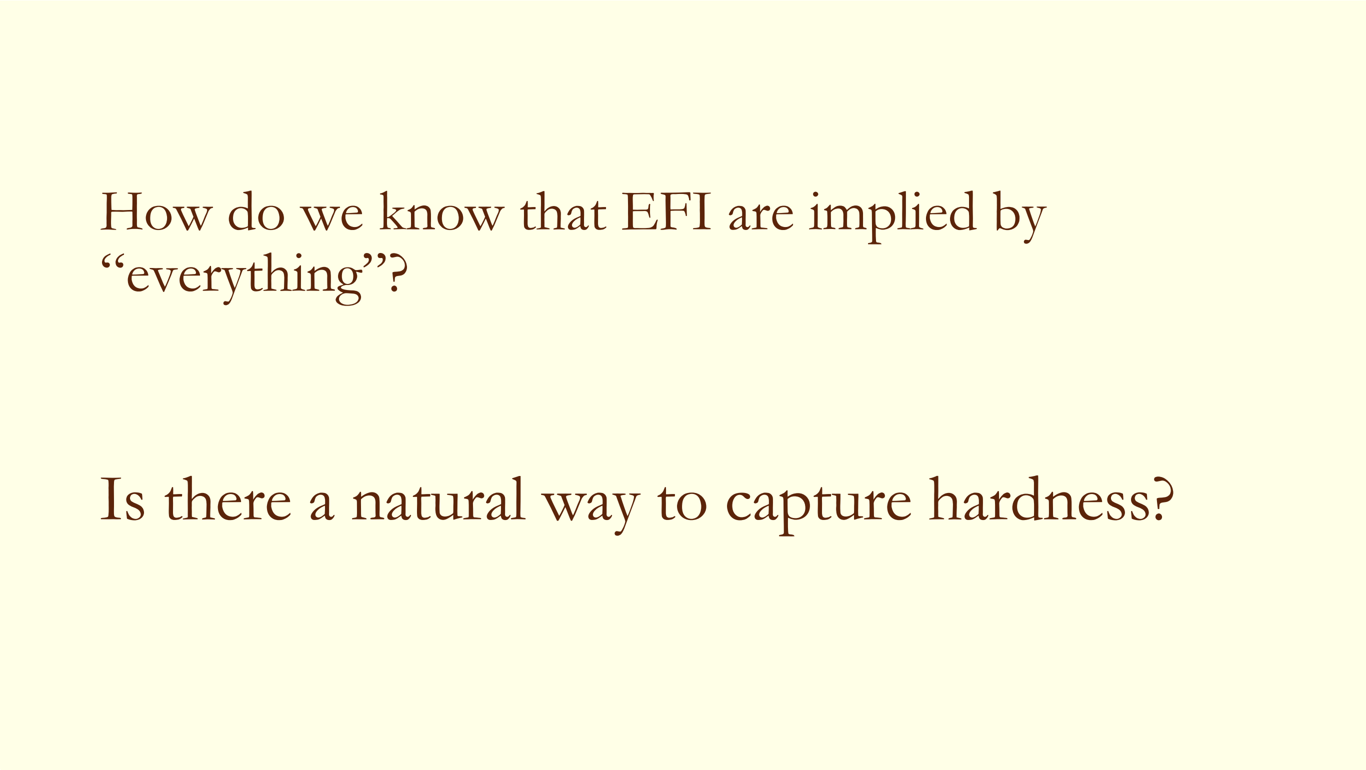
highly non-trivial!

☹

So even though they seem to be minimal, they don’t seem to capture cryptographic hardness as naturally as OWFs do. The main reason for this is that they are a decision primitive, and a lot of the hardness in quantum crypto is search hardness

How do we know that EFI are implied by "everything"?

So this talk aims to answer two questions:
First, how did we get here. How do we know that EFI pairs are implied by everything?
And secondly, is there a more natural way to understand the hardness in quantum cryptography



How do we know that EFI are implied by "everything"?

Is there a natural way to capture hardness?

So this talk aims to answer two questions:
First, how did we get here. How do we know that EFI pairs are implied by everything?
And secondly, is there a more natural way to understand the hardness in quantum cryptography

One-way puzzles

[KT24]

We answer these questions via a different cryptographic primitive called a one-way puzzle.

One-way puzzles

[KT24]

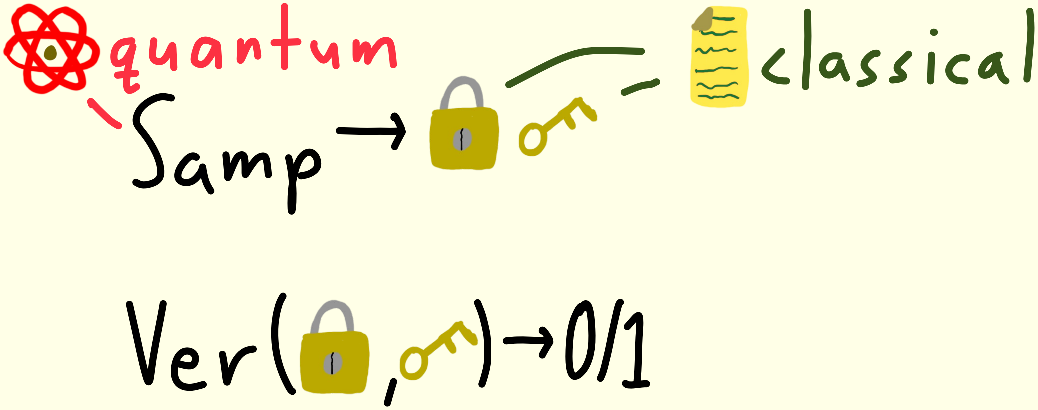
Samp → 🔒🔑

Ver(🔒, 🔑) → 0/1

A one-way puzzle consists of an efficient quantum sampler that outputs a hard problem, which we call the puzzle, along with the key or the solution.

One-way puzzles

[KT24]

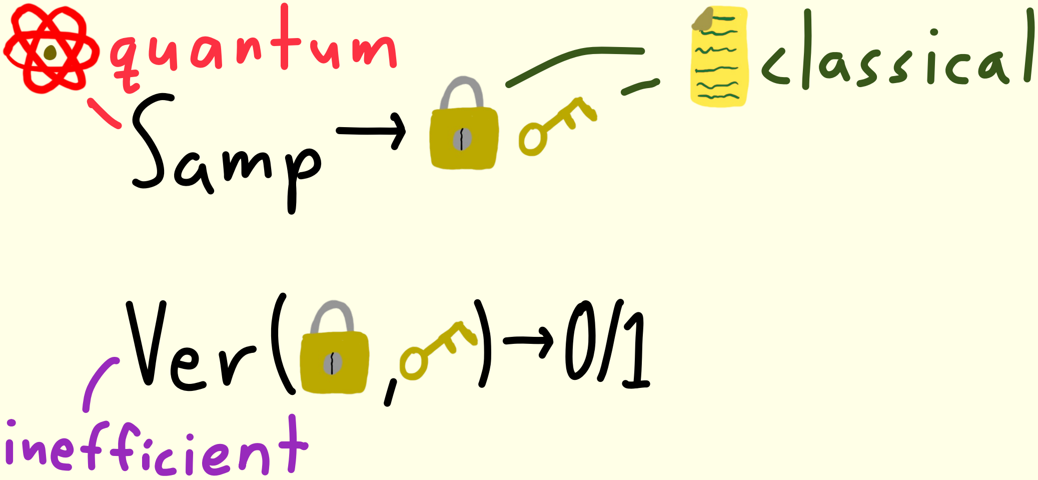


The sampler is a quantum machine, but the puzzle and its solution are both classical strings

Additionally, there is a verification process that checks whether puzzle and solution pairs are valid, where we have the guarantee that honestly generated pairs are always valid

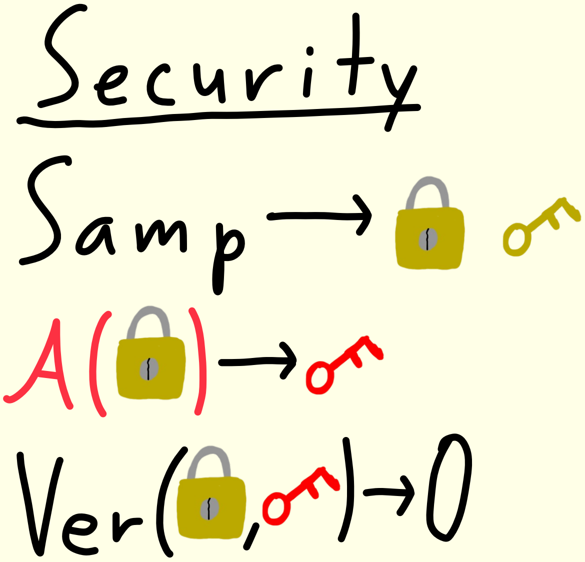
One-way puzzles

[KT24]



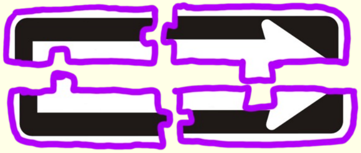
This verifier, however, is crucially allowed to be inefficient!

Security

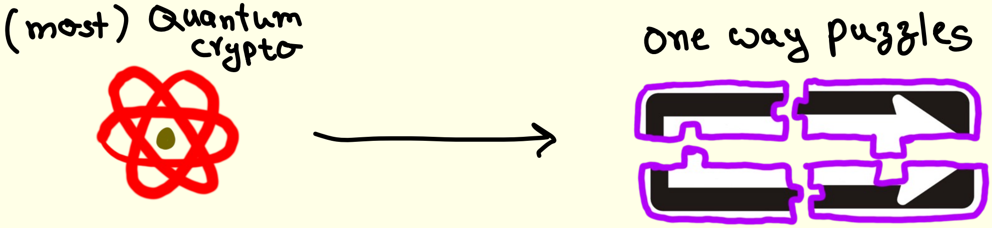


The security we require from one-way puzzles is that for honestly sampled puzzles, it should be computationally hard to find a valid key.

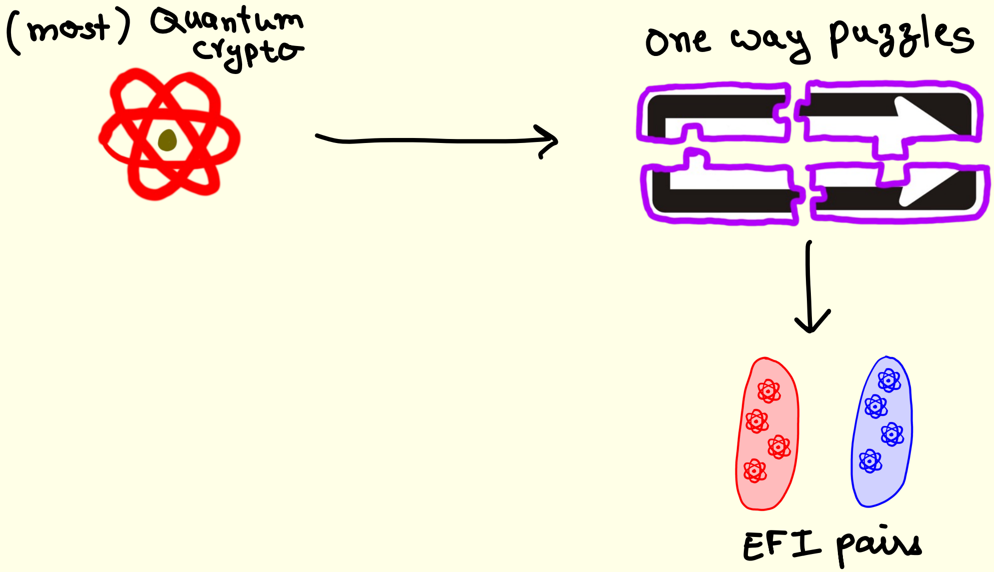
One way puzzles



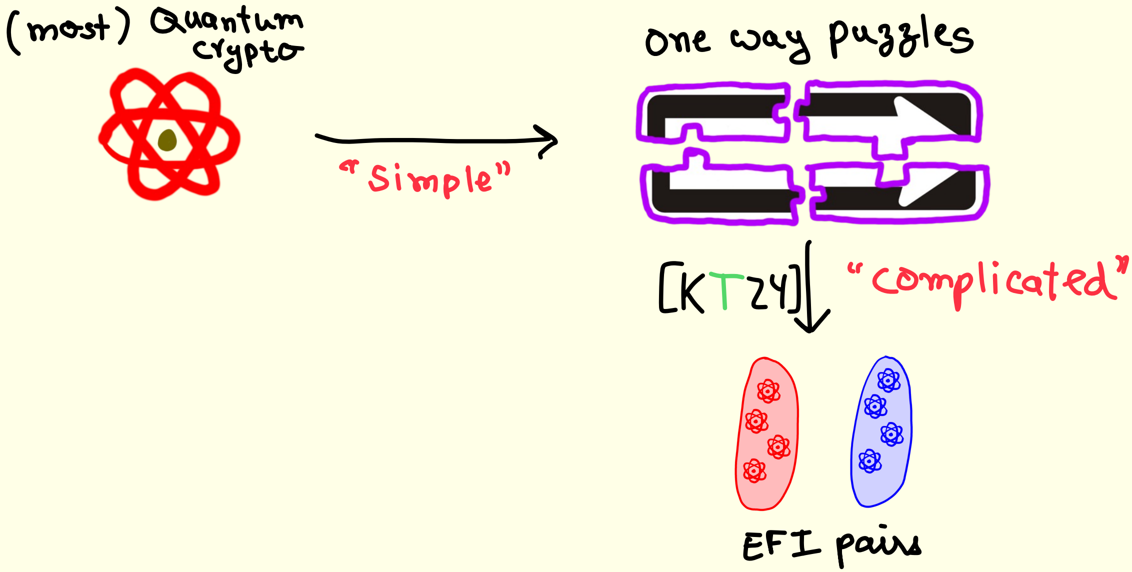
One-way puzzles turn out to be really useful in understanding quantum hardness.



This is because, somewhat surprisingly, one-way puzzles are implied by most quantum cryptography

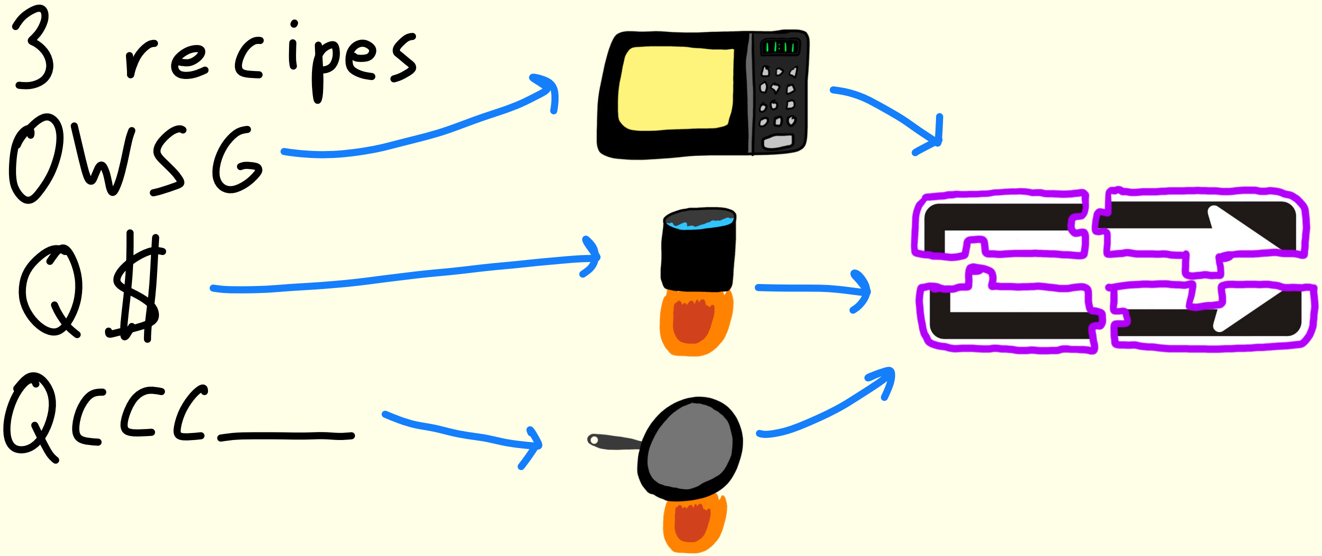


And one-way puzzles, in turn, can be used to build EFI pairs



Furthermore, while the construction of EFI pairs from one-way puzzles is complicated, We have developed techniques that mean constructing one-way puzzles from other cryptographic primitives tends to be very simple.

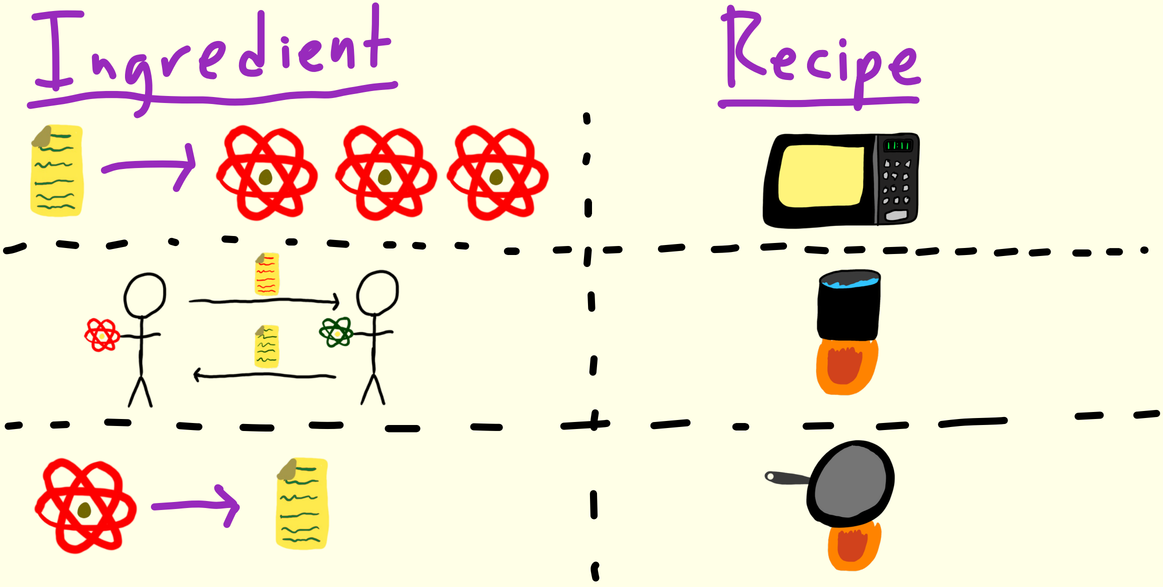
How to build?



The focus of the rest of this talk will be on these different techniques we know can be used to build one-way puzzles, which we will think of as different recipes we can use to capture differing types of quantum hardness, and most primitives that you can think of will fall in one of these categories. Along the way we will see a number of important applications.

Essentially all constructions of one-way puzzles fall into one of three distinct recipes, and Eli is going to give an overview of each of them.

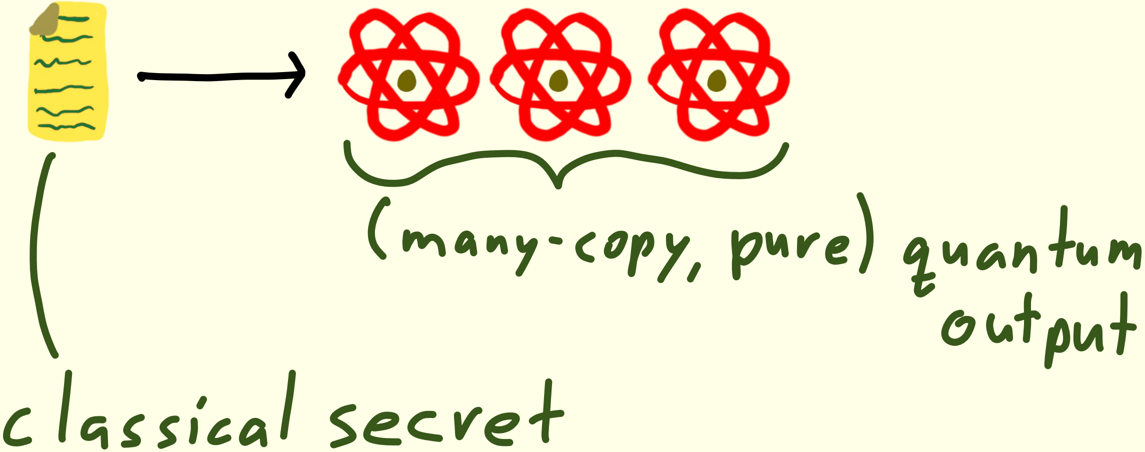
Recipes Overview



As a quick preview, we will give recipes for three different kinds of cryptographic primitives. Classical secret, quantum output. Classical communication. And quantum input, classical output. This covers most cryptographic primitives you could think of, and typically when you encounter an object which falls into one of these categories, it will be easy to plug in the corresponding recipe and build one-way puzzles (and thus non-trivial quantum cryptography)

Recipe #1: Shadow Tomography

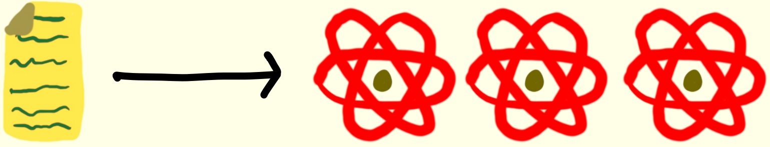
Ingredients



The first recipe follows along a technique known as shadow tomography. This technique applies to any cryptographic primitive with a classical input/secret and quantum output. In particular, the quantum output needs to consist of many copies of the same pure state (if you don't know what that means, don't worry about it).

Recipe #1: Shadow Tomography

Ingredients



Examples:

One-way state generators

Pseudorandom states/unitaries

Quantum encryption of classical messages

There are a number of examples of such primitives which have been previously studied. One-way state generators, a "quantum output" version of a one-way function, fall under this umbrella. As do pseudorandom states and unitaries, as well as encryption schemes where the ciphertext is allowed to be quantum (under a suitable definition).

Recipe #1: Shadow Tomography

Have $k \mapsto |\varphi_k\rangle|\varphi_k\rangle\cdots$

$\xrightarrow[\text{easy}]{}$

$\xleftarrow[\text{hard}]{}$

Want $k \mapsto S_k$ (classical)

$\xrightarrow[\text{easy}]{}$

$\xleftarrow[\text{hard}]{}$

To visualize this, we will begin with some problem where it is easy to go from a classical key k to many copies of the quantum state phi_k, but it is hard to go back. We want to then construct a protocol where it is easy to go from a classical key k to some other _classical_ string S_k, but hard to go back.

Recipe #1: Shadow Tomography

Have $k \mapsto |\varphi_k\rangle|\varphi_k\rangle\cdots$

$\xrightarrow[\text{easy}]{}$

$\xleftarrow[\text{hard}]{}$

assume "injective"!

Want $k \mapsto S_k$ (classical)

$\xrightarrow[\text{easy}]{}$

$\xleftarrow[\text{hard}]{}$

And for now let's just assume that this map is suitably "injective" (or statistically binding), although this isn't actually necessary. That is, phi_k should commit to the string k.

Recipe #1: Shadow Tomography

$$\begin{tikzcd}
\text{Have} & k \arrow[r, mapsto] & |\varphi_k\rangle|\varphi_k\rangle\cdots \arrow[dd, "\text{classical desc.}"] \\
\text{Idea:} & & \\
\text{Want} & k \arrow[r, mapsto] & S_k\ (\text{classical})
\end{tikzcd}$$

One very obvious approach to doing this would be to have S_k be a classical description of phi_k.

Recipe #1: Shadow Tomography

$$\begin{tikzcd}|\psi_k\rangle^{\otimes t} \arrow[d] \\ s_k\end{tikzcd}$$

may have exponential desc.

not always possible

must delete info!

But note that quantum states may in general have an exponential size classical description. In particular, there is no general efficient algorithm that can take many copies of a quantum state and produce an invertible classical description.

Any map from quantum states to classical strings must delete some information.

[HKP20]

Recipe #1: Shadow Tomography

$$\begin{tikzcd} k \arrow[r, mapsto] & {|\varphi_k\rangle^{\otimes t}} \arrow[d] \\ & S_k \end{tikzcd}$$

Shadow Tomography algorithm

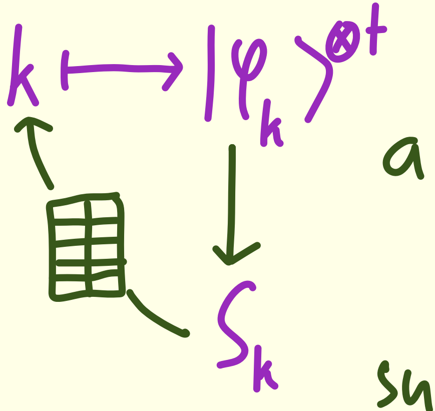
$$|\varphi_k\rangle^{\otimes t} \mapsto S_k$$

(preserves statistical information of choice

And this is where shadow tomography comes in. Classical shadow tomography is a procedure which maps a quantum states to a classical string where some statistical information _of your choice_ is preserved.

[HKP20]

Recipe #1: Shadow Tomography



Shadow Tomography

algorithm

$$|\varphi_k\rangle^{\otimes t} \mapsto S_k$$

such that you can compute

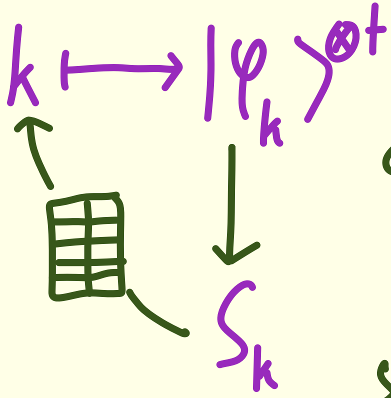
$S_k \to k$ inefficiently!

In particular, from phi_k, we can use classical shadow tomography to construct a classical shadow S_k which ALSO commits information theoretically to k.

That is, this technique (which we will not detail how it works) gives an efficient algorithm producing a classical shadow S_k from phi_k, as well as an inefficient map back to k from S_k.

[HKP20]

Recipe #1: Shadow Tomography



Shadow Tomography

algorithm

$|\varphi_k\rangle^{\otimes t} \longmapsto S_k$ classical shadow

such that you can compute

$S_k \to k$ inefficiently!

This string S_k is called a classical shadow.

[KT24]

Recipe #1: Shadow Tomography

Ingredient

$$\begin{tikzcd} k \mapsto |\varphi_k\rangle|\varphi_k\rangle\cdots \arrow[d, "ST"] \\ S_k \end{tikzcd}$$

$\mathsf{Samp} \rightarrow$ 🔑 : k

🔒 : S_k

So, to build a one-way puzzle from a classical secret, quantum output problem, the recipe goes as follows.
The key will be the classical secret k.
The puzzle will be the classical shadow S_k produced from phi_k.

[KT24]

Recipe #1: Shadow Tomography

Ingredient

$$\begin{tikzcd} k \mapsto |\varphi_k\rangle|\varphi_k\rangle\cdots \arrow[d, "ST"] \\ S_k \end{tikzcd}$$

Samp → 🔑 : k

🔒 : S_k

$\mathrm{Ver}(S_k, k')$: $S_k \longrightarrow k$

$\rightarrow k = k'$?

The inefficient verifier takes in a puzzle S_k and a key k’. It applies the inefficient map sending S_k back to k, and then compares k and k’. This is inefficient, but that is fine for our definition!

[KT24]

Recipe #1: Shadow Tomography

Ingredient

$$\begin{tikzcd} {} & k \mapsto |\varphi_k\rangle|\varphi_k\rangle\cdots \arrow[l, "\text{hard}"'] \arrow[d, "ST"] \\ & S_k \end{tikzcd}$$

Samp → 🔑 : k

🔒 : S_k

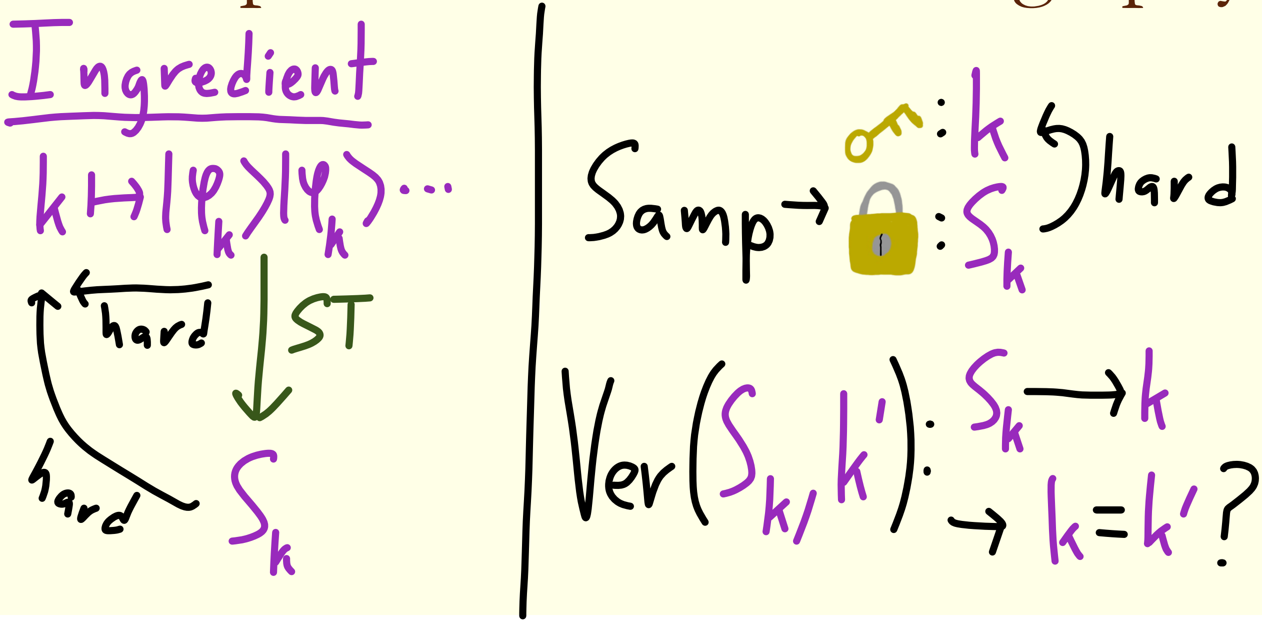
$\mathrm{Ver}(S_k, k')$: $S_k \rightarrow k$

→ $k = k'$?

Since it is hard to go from |phi_k> back to k, and since we can easily build S_k from |phi_k>

[KT24]

Recipe #1: Shadow Tomography



It must also be hard to go from S_k back to k (that is, solve the puzzle!)

And so one-way state generators and friends can all be used to build one-way puzzles

[IL89, Imp Thesis]

Recipe #2: Universal Extrapolation

Classically, one-way function inverter $\Longrightarrow$

① Universal Extrapolation:
For all classically samplable distributions (X,Y)

$$(X,Y) \approx (X, \mathrm{Ext}(X))$$

The second recipe is a technique sometimes known as “universal extrapolation”.

Classically, if one-way functions do not exist, then there exists a “universal extrapolator” Ext which does the following. Given any classically samplable, correlated distributions X and Y. The distribution (X,Y) is statistically close to (X, Ext(X)). That is, the universal extrapolator allows one to sample Y conditioned on the value of X for a randomly drawn x.

[IL89, Imp Thesis]

Recipe #2: Universal Extrapolation

Classically, one-way function inverter ⟹

① Universal Extrapolation:
For all classically samplable distributions (X,Y)

$$(X,Y) \approx (X, \mathrm{Ext}(X))$$

Actually, most common way to build OWF!
Key Exchange, commitments, ... all broken by UE!

From some perspective, this is actually the most common way to build one-way functions. Key exchange, commitments, and pretty much all cryptographic primitives are all easily broken by universal extrapolation, even if it is not immediately obvious how to build a one-way function.

[IL89, Imp Thesis]

Recipe #2: Universal Extrapolation

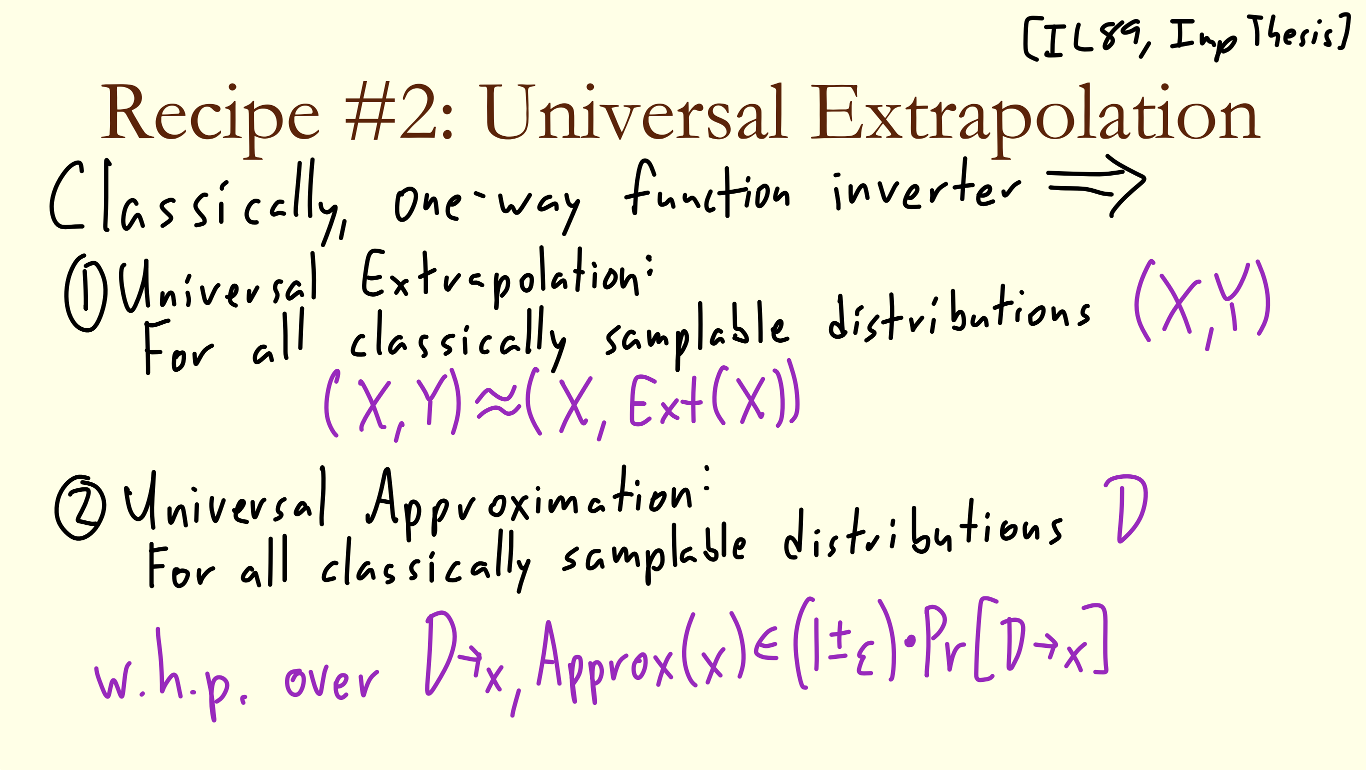
Classically, one-way function inverter ⟹

① Universal Extrapolation:
For all classically samplable distributions (X,Y)

$$(X,Y) \approx (X, Ext(X))$$

Hardness of UE essentially a "distributional OWF"

The hardness of universal extrapolation is essentially something called a "distributional one-way function"



[IL89, Imp Thesis]

Recipe #2: Universal Extrapolation

Classically, one-way function inverter $\Longrightarrow$

① Universal Extrapolation:

For all classically samplable distributions (X,Y)

$$(X,Y)\approx(X,\mathrm{Ext}(X))$$

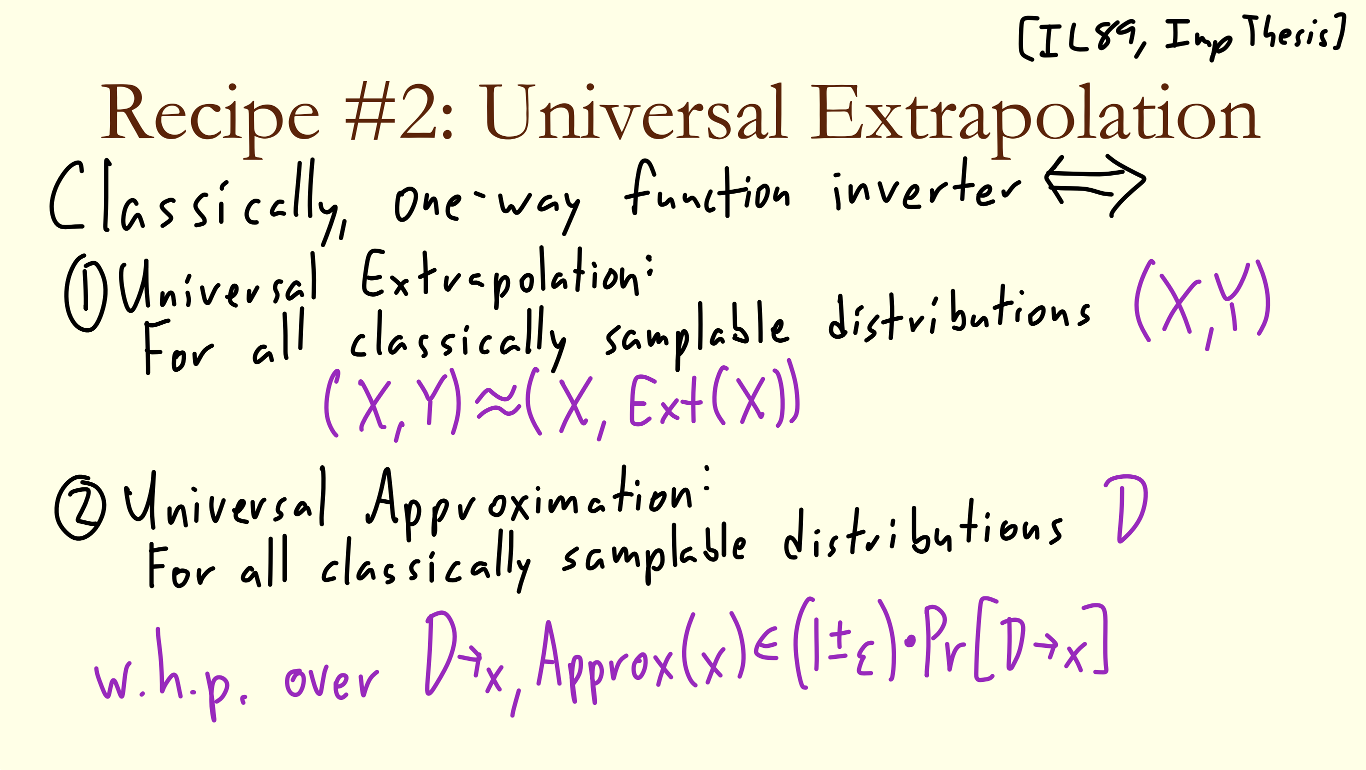
② Universal Approximation:

For all classically samplable distributions D

w.h.p. over $D\to x$, $\mathrm{Approx}(x)\in(1\pm\varepsilon)\cdot\Pr[D\to x]$

A second, very related task is something we call universal approximation (or probability estimation). If one-way functions don't exist, then for all classically samplable distributions D, there exists an approximator Approx. For a randomly chosen x, Approx(x) will give a good multiplicative estimate of the probability that D outputs x.

So in short, no one way functions means we can do universal extrapolation and approximation. In other words, average-case conditional sampling and average-case estimation of output probabilities.



[IL89, Imp Thesis]

Recipe #2: Universal Extrapolation

Classically, one-way function inverter $\iff$

① Universal Extrapolation:
For all classically samplable distributions (X,Y)

$$(X,Y) \approx (X, \mathrm{Ext}(X))$$

② Universal Approximation:
For all classically samplable distributions D

w.h.p. over $D \to x$, $\mathrm{Approx}(x) \in (1 \pm \varepsilon) \cdot \Pr[D \to x]$

And in fact the converse also holds.

[KT25, CGG+25, CGG24, HM25, QRZZ25]

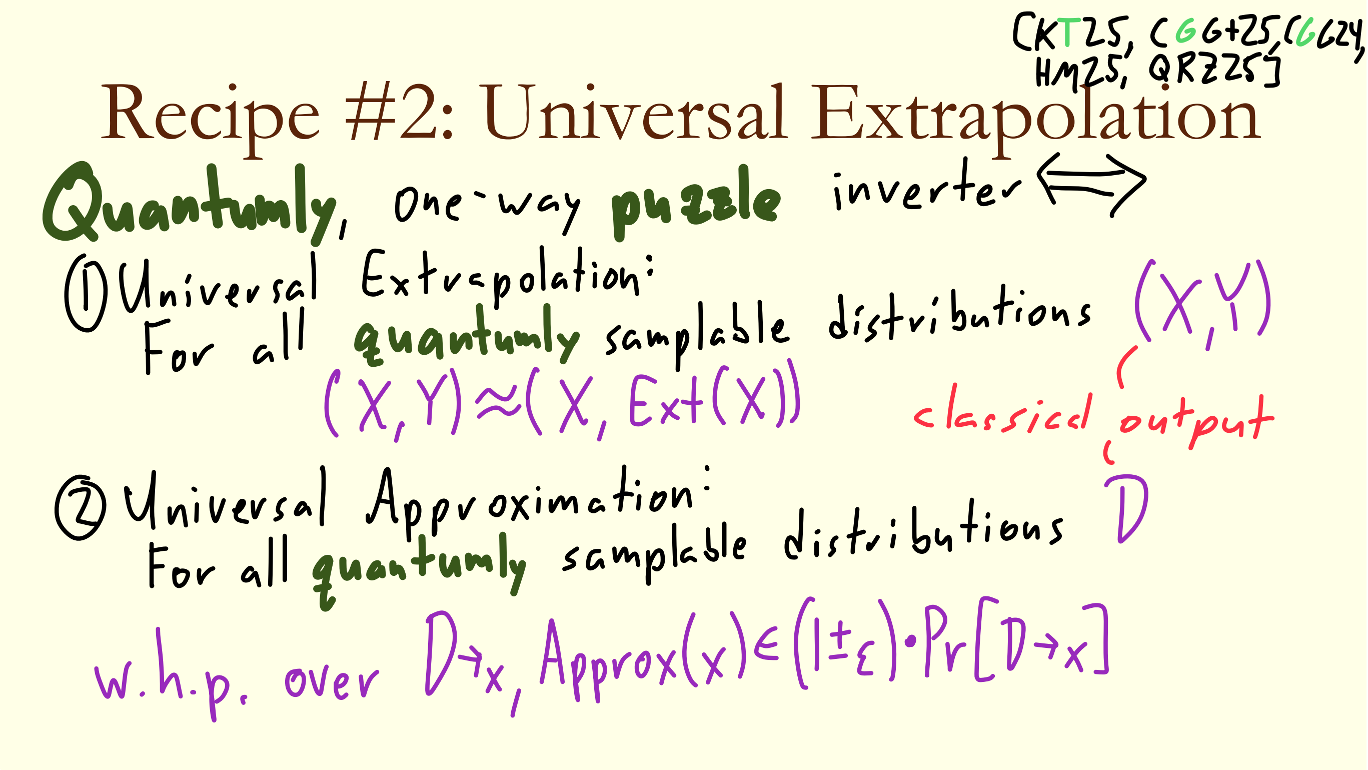
Recipe #2: Universal Extrapolation

Quantumly, One-way **puzzle** inverter $\iff$

1. Universal Extrapolation:
   For all **quantumly** samplable distributions (X,Y)
   $$(X,Y) \approx (X, \mathrm{Ext}(X))$$

2. Universal Approximation:
   For all **quantumly** samplable distributions D
   w.h.p. over $D \to x$, $\mathrm{Approx}(x) \in (1 \pm \varepsilon) \cdot \Pr[D \to x]$

It turns out, that both of these results also hold for one-way puzzles in the quantum setting! The non-existence of one-way puzzles is equivalent to the ability to do universal extrapolation and approximation on _quantumly_ samplable distributions over classical strings. That is, the theorems are exactly the same, but now universal extrapolation works even when the sampler is quantum.



[KT25, CGG+25, CGG24, HM25, QRZ25]

Recipe #2: Universal Extrapolation

Quantumly, one-way **puzzle** inverter ⟺

① Universal Extrapolation:
For all **quantumly** samplable distributions (X,Y)

$$(X,Y)\approx(X,\mathrm{Ext}(X))$$

classical output

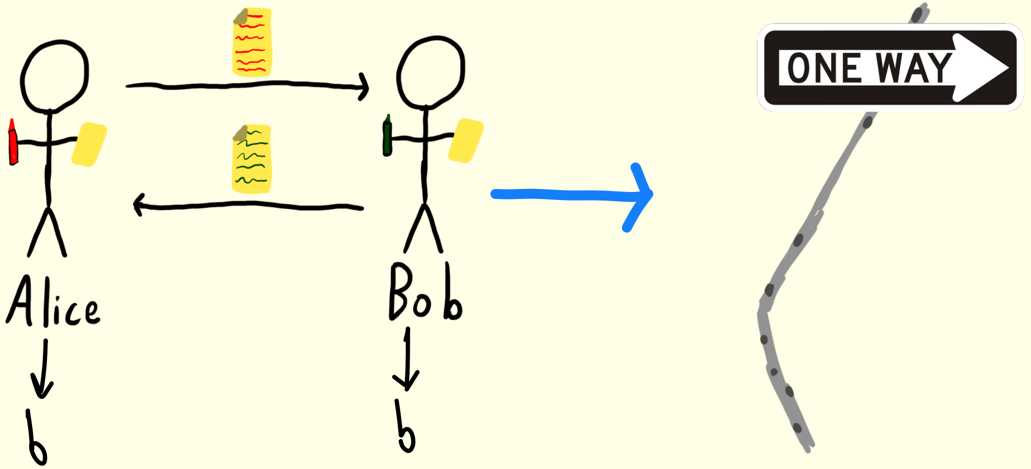
② Universal Approximation:
For all **quantumly** samplable distributions D

w.h.p. over $D\to x$, $\mathrm{Approx}(x)\in(1\pm\varepsilon)\cdot\Pr[D\to x]$

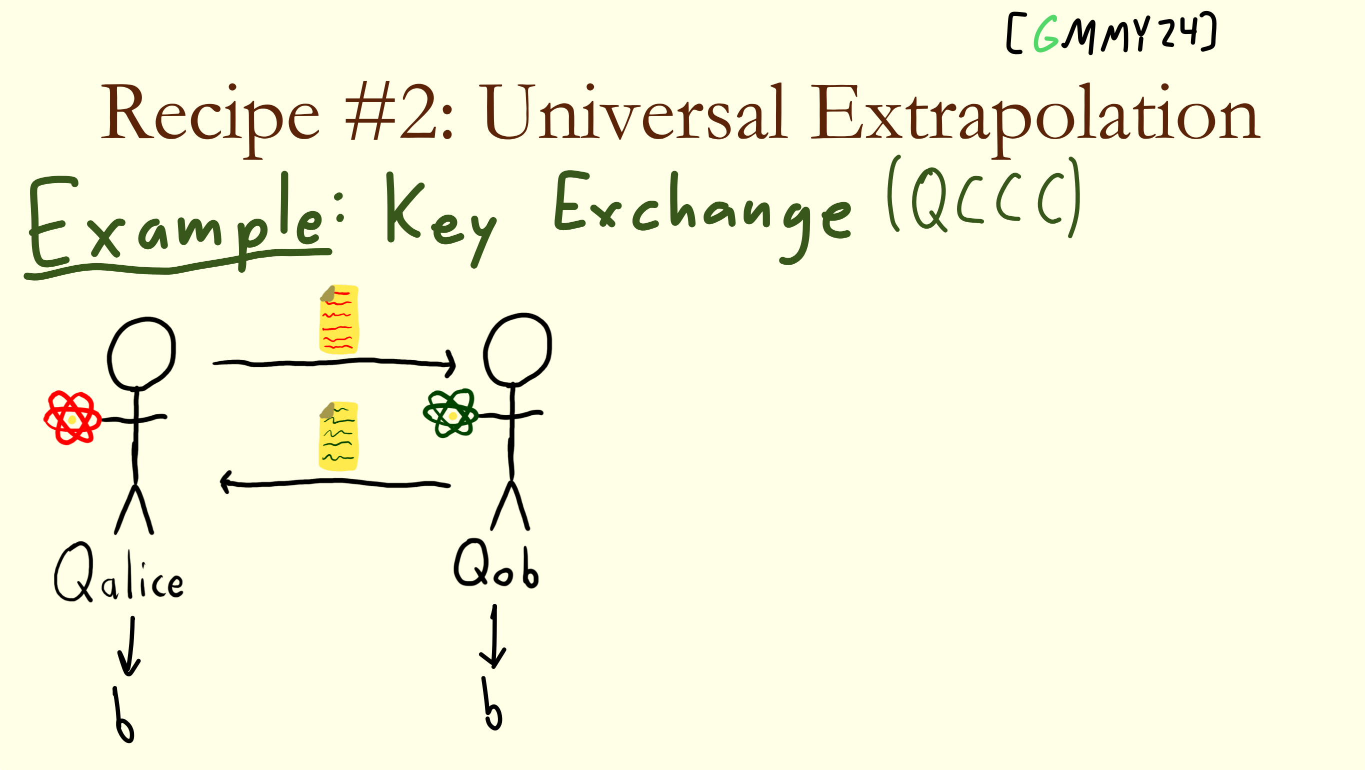
It turns out, that both of these results also hold for one-way puzzles in the quantum setting! The non-existence of one-way puzzles is equivalent to the ability to do universal extrapolation and approximation on _quantumly_ samplable distributions over classical strings. That is, the theorems are exactly the same, but now universal extrapolation works even when the sampler is quantum.

Recipe #2: Universal Extrapolation

Example: Key Exchange



So how can we use this to build one-way puzzles? As an example, I will talk about a primitive you all know and love, key exchange, where classical Alice and classical Bob communicate classically, and then agree on a secret bit b. We know that we can build one-way functions from classical key exchange.



[GMMY24]

Recipe #2: Universal Extrapolation

Example: Key Exchange (QCCC)

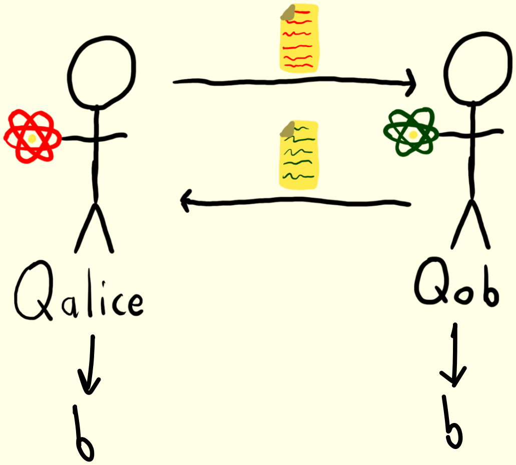
But what happens when we restrict ourselves to the “quantum computation and classical communication setting”? That is, while we will QAlice and Qob to be quantum, but we require them to communicate classically.

[GMMY24]

Recipe #2: Universal Extrapolation

Example: Key Exchange (QCCC)

$(X,Y) = ($📄📄$, b)$

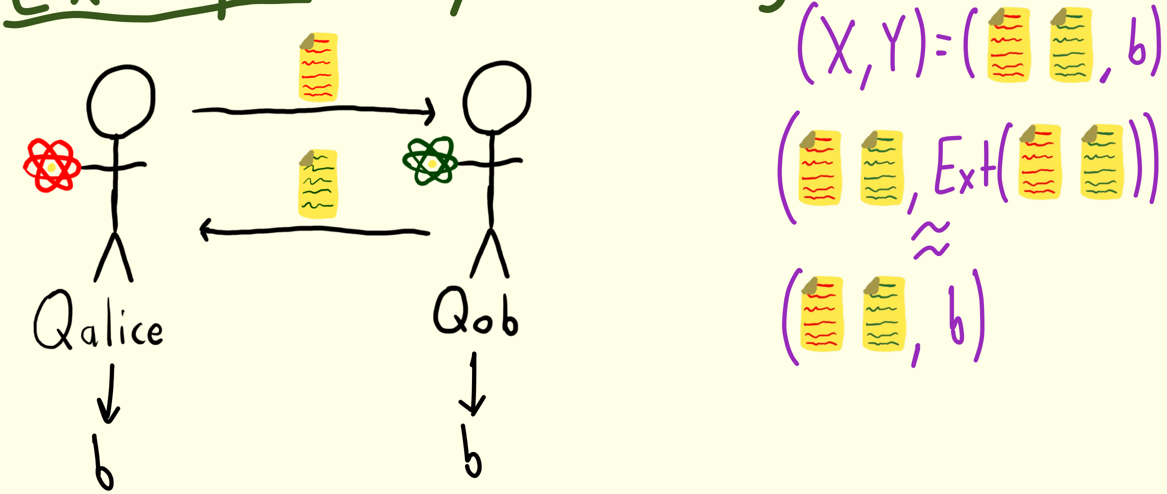


It turns out that so-called QCCC key exchange is very easy to break using universal extrapolation. We will simply let X be the transcript, and Y will be the agreed bit b.

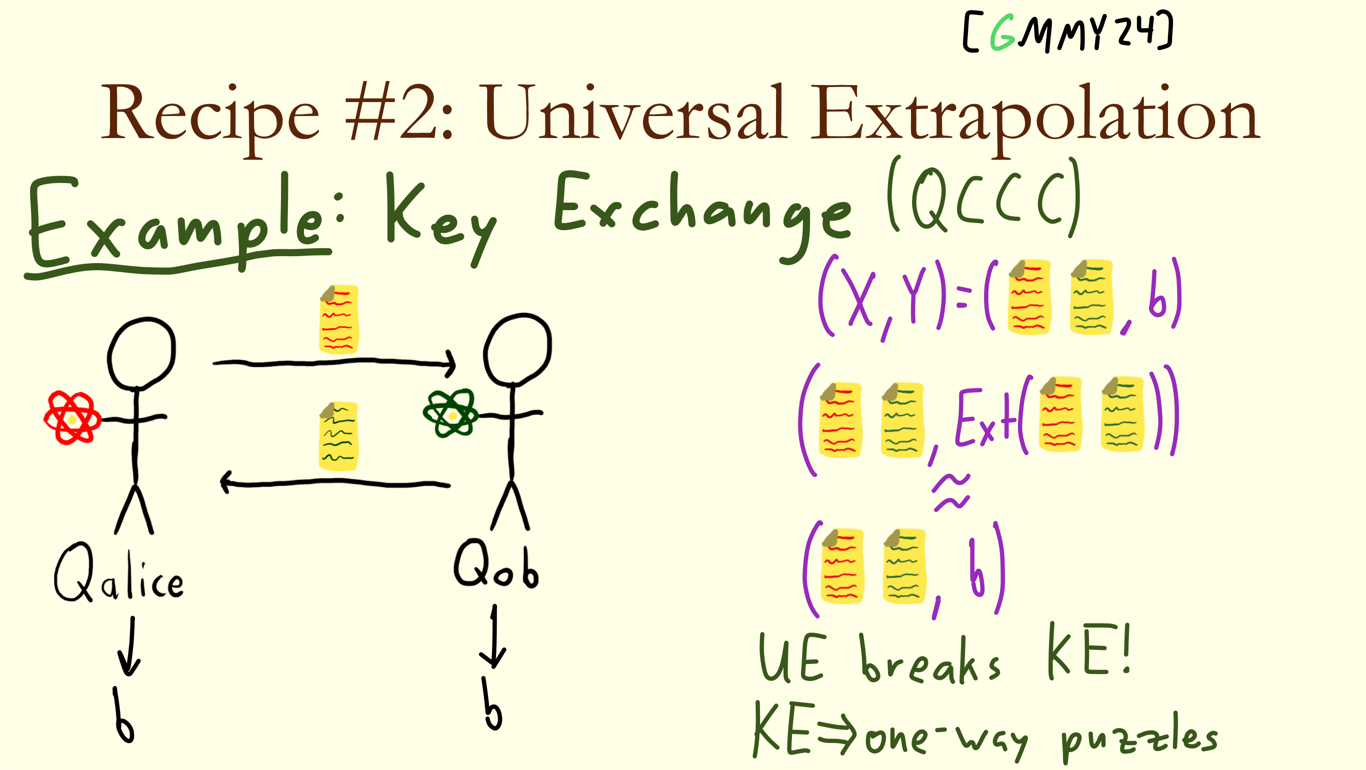
[GMMY24]

Recipe #2: Universal Extrapolation

Example: Key Exchange (QCCC)



Now, universal extrapolation of the transcript will give the exact distribution of the output bit b!



[GMMY24]

Recipe #2: Universal Extrapolation

Example: Key Exchange (QCCC)

$$(X, Y) = (📄📄, b)$$

$$(📄📄, \mathrm{Ext}(📄📄))$$

$$\approx$$

$$(📄📄, b)$$

UE breaks KE!

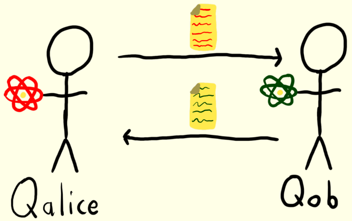
KE ⇒ one-way puzzles

And so the universal extrapolation we get from no one-way puzzles breaks key exchange! And so key exchange can be used to build one-way puzzles.

Recipe #2: Universal Extrapolation

Ingredients:

Any cryptography with classical communication & quantum parties

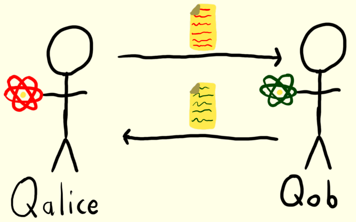


So what kinds of primitives will this approach work for? Really it works for practically _any_ cryptography with classical communication and quantum parties.

Recipe #2: Universal Extrapolation

Ingredients:

Any cryptography with classical communication & quantum parties



REALLY POWERFUL

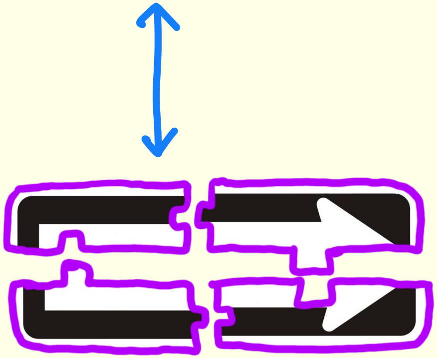
This is in fact much more powerful than just “QCCC versions of normal cryptography”, and it has a bunch of relatively surprising applications.

[CGG+25, HM25]

Recipe #2: Universal Extrapolation

Examples:

① Hardness of estimating metacomplexity on quantumly samplable dists



One direct example is that this can be used to show that the existence of one-way puzzles is equivalent to the hardness of some metacomplexity problem on _quantumly samplable_ distributions.

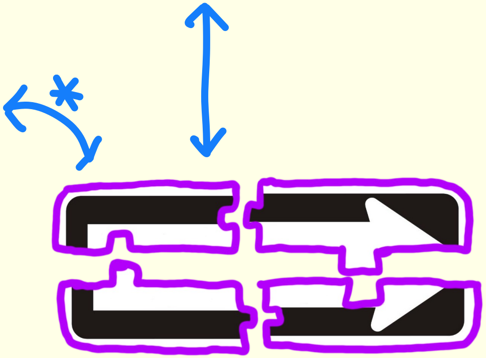
[MYY25]

Recipe #2: Universal Extrapolation

Examples:

① Hardness of estimating metacomplexity on quantumly samplable dists

② inefficient proofs of quantumness



A very surprising other example of this recipe is that it is used to show that one-way puzzles are equivalent to some notion of quantum advantage! That is, one-way puzzles (with classical security) exist if and only if there exists proofs of quantumness with inefficient verification!

[KT25]

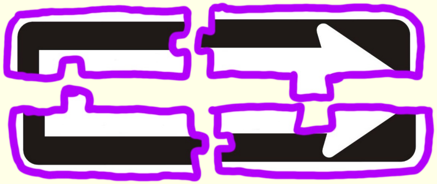
Recipe #2: Universal Extrapolation

Examples:

① Hardness of estimating metacomplexity on quantumly samplable dists

② inefficient proofs of quantumness *

③ Quantum advantage assumptions



This approach also lets us give concrete candidate constructions of one-way puzzles (and thus EFI and quantum crypto more generally) from quantum advantage style assumptions.

[KT24, GMMY25]

Recipe #2: Universal Extrapolation

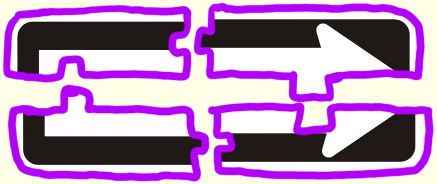
Examples:

① Hardness of estimating metacomplexity on quantumly samplable dists

② inefficient proofs of quantumness *

③ Quantum advantage assumptions

④ QCCC crypto



Finally, and maybe the most obvious application, we get that essentially all cryptography with classical communication can be used to construct one-way puzzles.

Recipe #2: Universal Extrapolation

Security

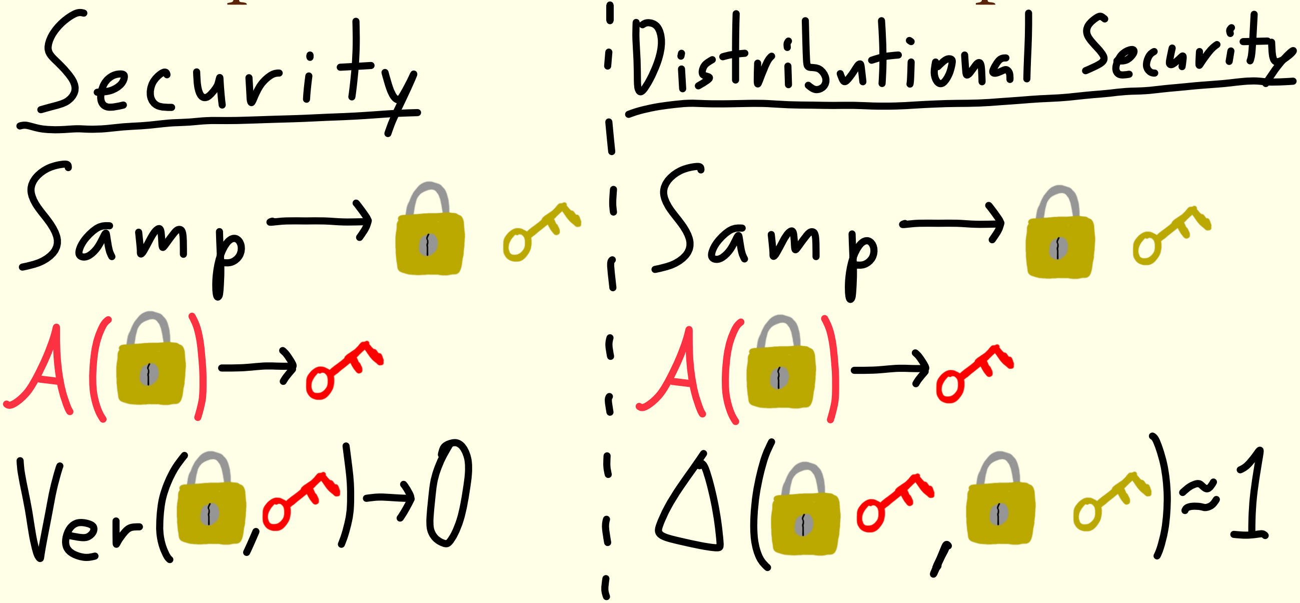
Samp → 🔒 🔑

A(🔒) → 🔑

Ver(🔒, 🔑) → 0

One final advantage of this equivalence is that it allows us to weaken the definition of one-way puzzles to a “distributional” notion. Recall that security of a one-way puzzle originally said that an adversary cannot find a key that verifies.

Recipe #2: Universal Extrapolation



But we could consider a distributional notion, which says that an adversary cannot sample from the conditional distribution on keys for a given puzzle. That is, the distribution of puzzles followed by adversary's keys should be statistically far from the honest puzzle, key distribution.

[CGG24]

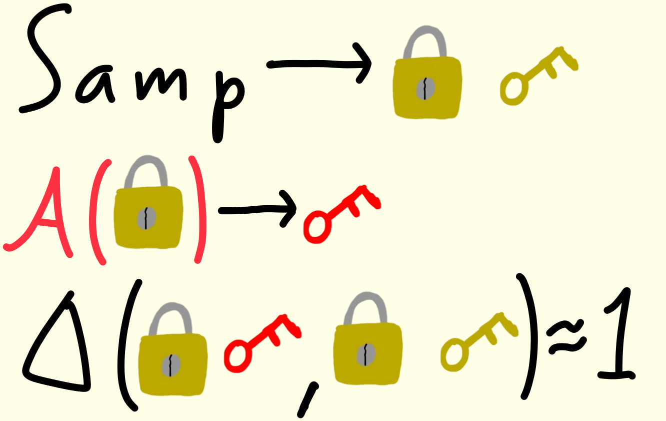
Recipe #2: Universal Extrapolation

"Distributional One-way puzzles"

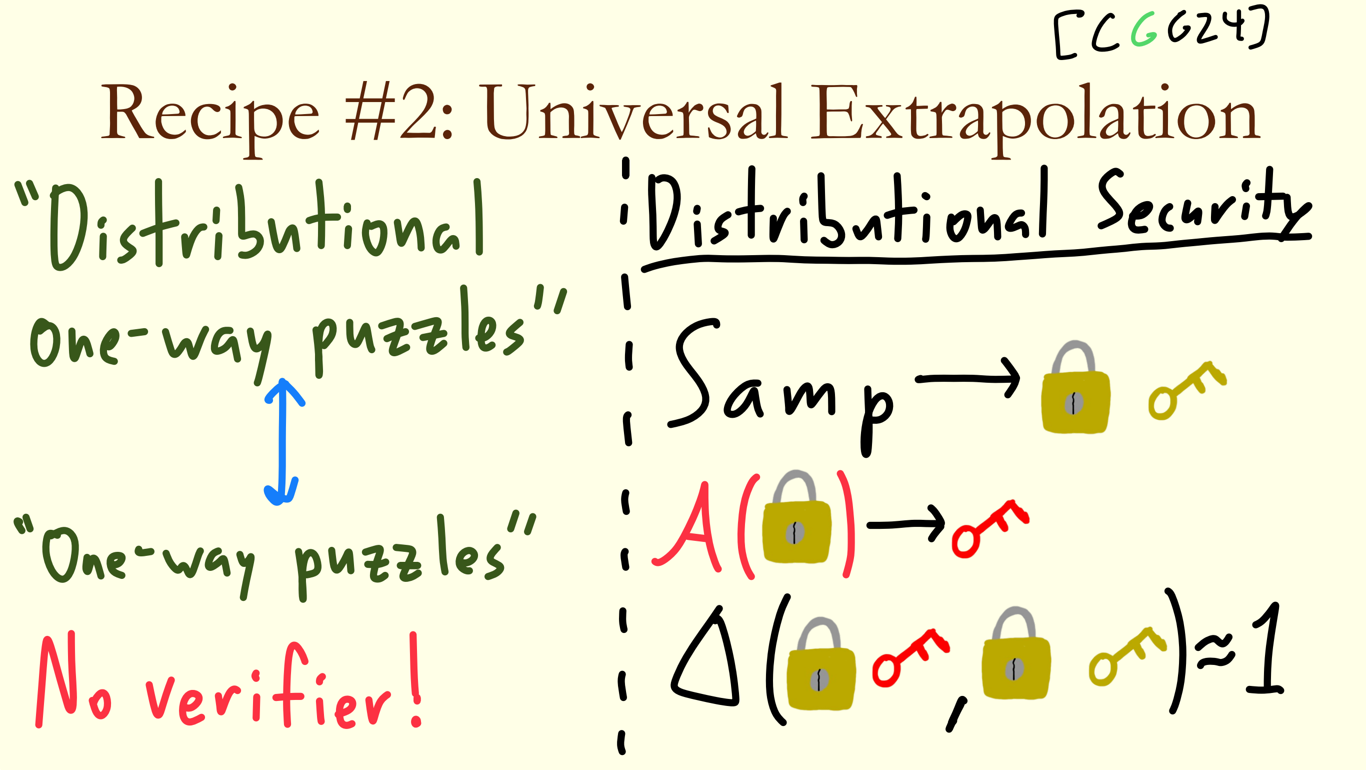
$\updownarrow$

"One-way puzzles"

Distributional Security



We call such a primitive a "distributional one-way puzzles" and it is essentially a hard instance of universal extrapolation. And so we have that dist owpz exist if and only if owpz exist.



[CGG24]

Recipe #2: Universal Extrapolation

"Distributional One-way puzzles"

↕

"One-way puzzles"

No verifier!

Distributional Security

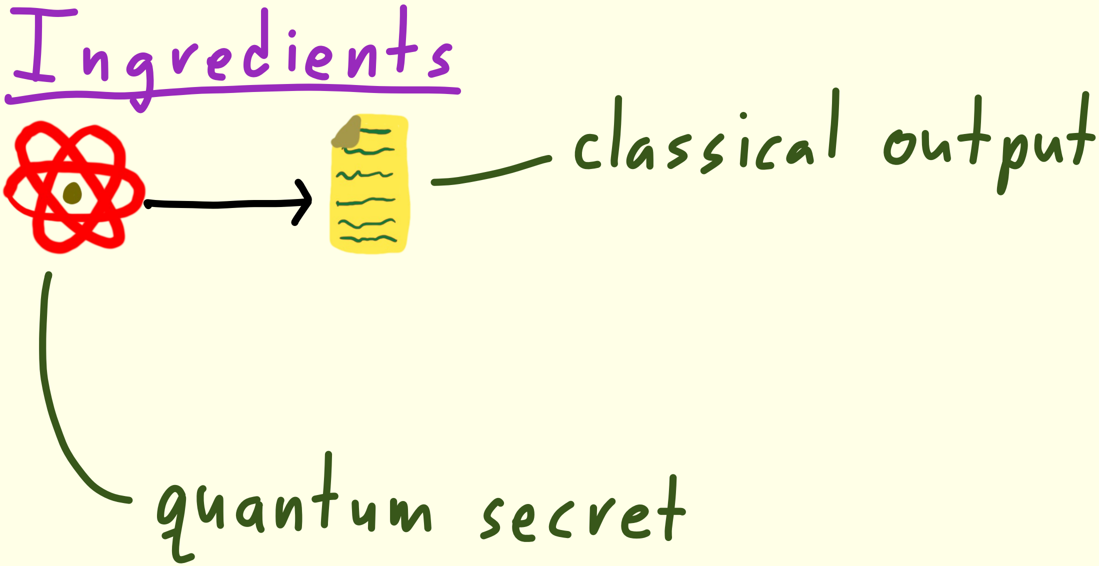
Samp → 🔒 🔑

A(🔒) → 🔑

Δ(🔒 🔑, 🔒 🔑) ≈ 1

The nice thing about this alternative definition of one-way puzzles is that there we don’t need to deal about the weirdness of having an inefficient verifier, since there is no verifier in the first place!

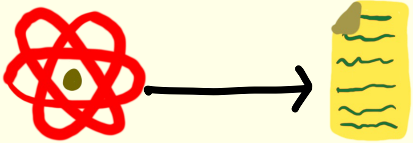
Recipe #3: State Puzzles



Our final recipe for constructing one-way puzzles will go through something called "state puzzles"
It will work for any cryptographic primitive with a classical output, even if the secret is quantum.

Recipe #3: State Puzzles

Ingredients



Examples:

- Quantum money
- Classical public key + quantum secret key
- 2 round quantum key distribution

This is probably the most general of the techniques, but a few examples of these are quantum money, encryption with a classical public key and quantum secret key, as well as 2 round key exchange with unauthenticated quantum communication.

Recipe #3: State Puzzles

Have $|\varphi_k\rangle \mapsto k$

$$\xrightarrow[\text{easy}]{}$$

$$\xleftarrow[\text{hard}]{}$$

This technique will build upon the previous one. In particular, we will begin with a primitive that maps a quantum state to a classical key such that it is easy to find the key from the state (or sample both together), but it is hard to recover the state from the key.

Recipe #3: State Puzzles

Have $|\varphi_k\rangle \mapsto k$

$$\xrightarrow[\text{easy}]{}$$

$$\xleftarrow[\text{hard}]{}$$

$$\left.\begin{array}{l}\text{Want distribution over classical strings } D\\ \quad \text{easy: sampling } D \to x\\ \quad \text{hard: approximating } \Pr[D \to x]\end{array}\right\}\begin{array}{l}\text{hardness of}\\ \text{universal}\\ \text{approximation}\end{array}$$

We will then want to build a hard instance of universal approximation. That is, some distribution D such that given a sample x, it is hard to approximate the probability that D outputs x.

Recipe #3: State Puzzles

Have $|\varphi_k\rangle \mapsto k$

$$\xrightarrow[\text{easy}]{}$$

$$\xleftarrow[\text{hard}]{}$$

Solution: State Synthesis

Want distribution over classical strings D

easy: sampling $D \to x$

hard: approximating $\Pr[D \to x]$

$\Bigg\}$ hardness of universal approximation

In the worst-case setting, it is known how to do this using state synthesis techniques.

And in fact these techniques also translate to the average-case setting (with some work).

Recipe #3: State Puzzles

State puzzle:

$$\mathsf{Samp} \to |\varphi_s\rangle, s$$

quantum

classical

For ease of use, we abstract out this recipe into a primitive dubbed by Kabir as a state puzzle. A state puzzle is essentially a one way puzzle with a _quantum_ key.

Recipe #3: State Puzzles

[KT25, QRZ25]

State puzzle:

$$\mathsf{Samp} \to |\varphi_s\rangle, s$$

$$\left.\begin{array}{l} A(s) \to |\psi\rangle \\ |\varphi_s\rangle \neq |\psi\rangle \end{array}\right\} \text{security}$$

Furthermore, we relax security to be the simplest possible thing. We simply require that no adversary on input a puzzle s can recover the _actual_ key state |phi_s>

Recipe #3: State Puzzles

[KT25, QRZ25]

State puzzle:

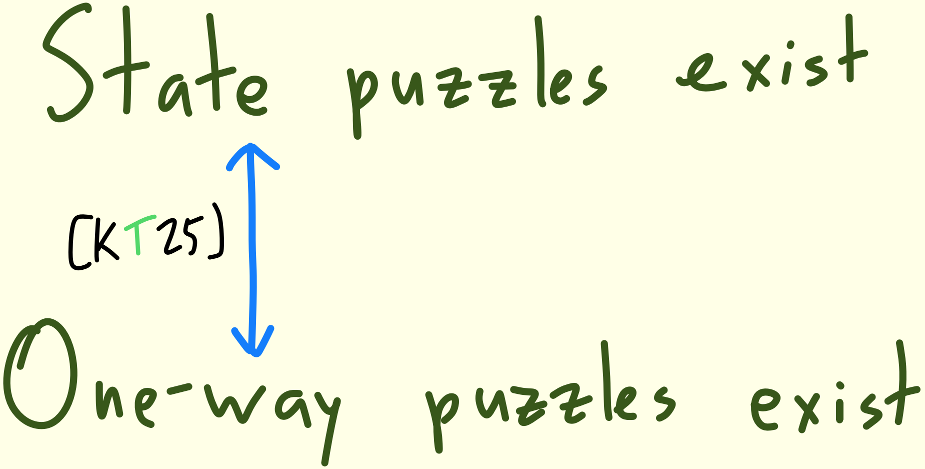
Falsifiable!

$$\mathsf{Samp} \rightarrow |\varphi_s\rangle, s$$

$$\left.\begin{array}{l} A(s) \rightarrow |\psi\rangle \\ |\varphi_s\rangle \neq |\psi\rangle \end{array}\right\} \text{security}$$

Note that this is actually a falsifiable definition! Given an adversary, you can test whether it works yourself by comparing the adversary output to your stored key.

Recipe #3: State Puzzles



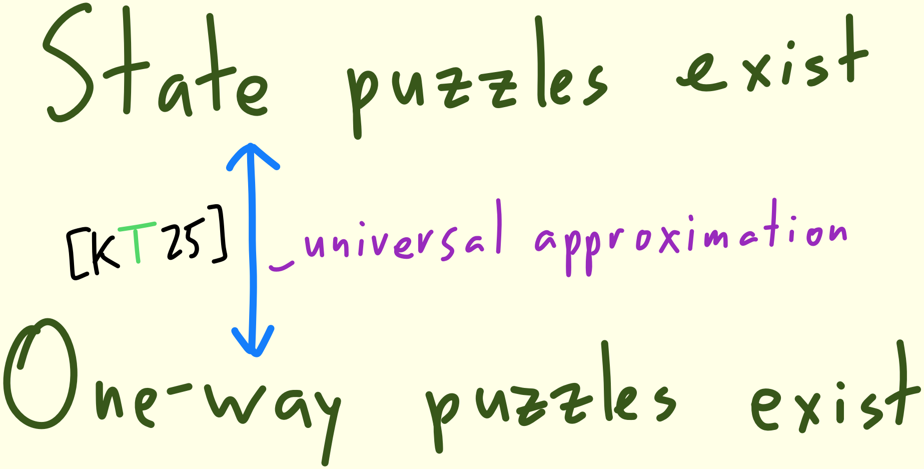
State puzzles exist

[KT25]

One-way puzzles exist

In a recent work, KT25 showed that state puzzles exist if and only if one-way puzzles exist, exactly by going through universal approximation

Recipe #3: State Puzzles



State puzzles exist

[KT25]

universal approximation

One-way puzzles exist

In a recent work, KT25 showed that state puzzles exist if and only if one-way puzzles exist, exactly by going through universal approximation

Recipe #3: State Puzzles

Example: Building one-way puzzles from a single random state

I will show a kind of cool, surprising, and very easy application. Note that if you have just a random string, this is cryptographically useless. There is no way to information theoretically build one-way functions. But, if you have a single random state, then it is possible to build one-way puzzles!

Recipe #3: State Puzzles

Example: Building one-way puzzles from a single random state

random $|\varphi\rangle_{AB}$

In particular, let |phi> be a random state on two registers A and B.

Recipe #3: State Puzzles

Example: Building one-way puzzles from a single random state

random $|\psi\rangle_{AB}$

Samp: measure $B \rightarrow |\varphi_s\rangle_A, s$

The state puzzle will act as follows. It will measure register B in the standard basis producing a string s as well as a residual state |phi_s> on register A.

Recipe #3: State Puzzles

Example: Building one-way puzzles from a single random state

random $|\psi\rangle_{AB}$

Samp: measure $B \rightarrow |\varphi_s\rangle_A, s$

Then, it will set the puzzle to be s, and the key to be phi_s.

Recipe #3: State Puzzles

Example: Building one-way puzzles from a single random state

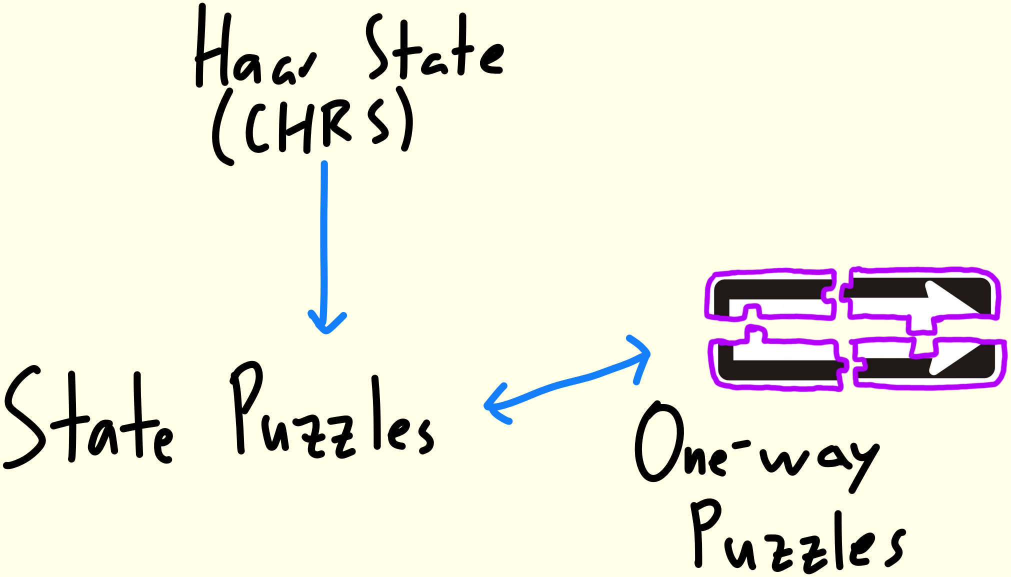
random $|\psi\rangle_{AB}$

Samp: measure $B \rightarrow |\psi_s\rangle_A, s$

State puzzle!

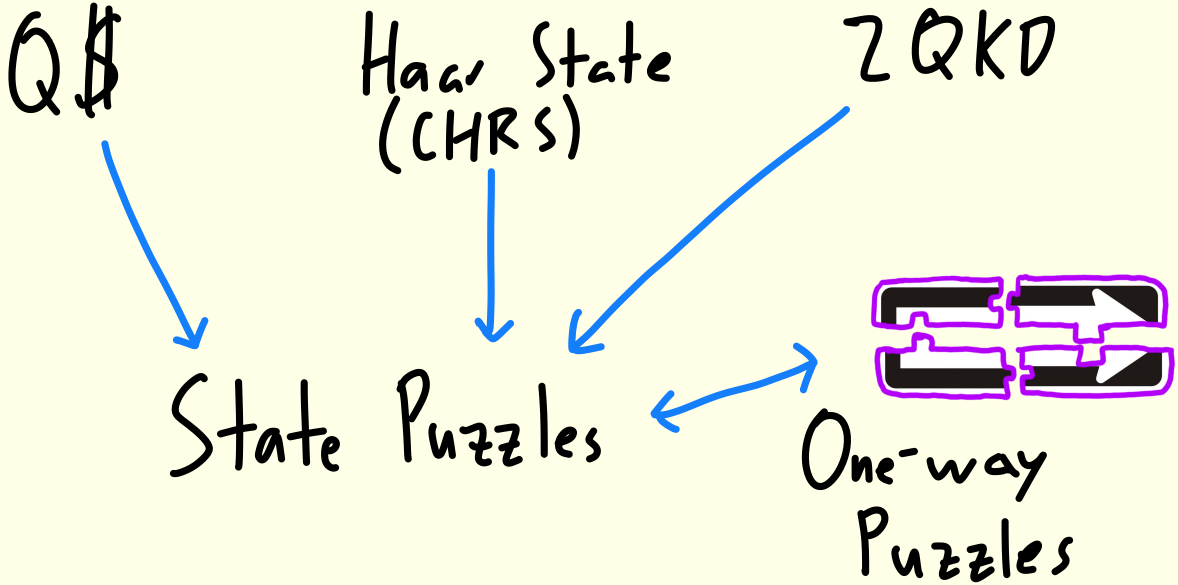
It is information-theoretically impossible to construct phi_s from s and phi in polynomial time, and so this is a state puzzle!

Recipe #3: State Puzzles



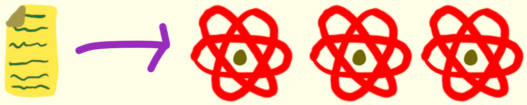
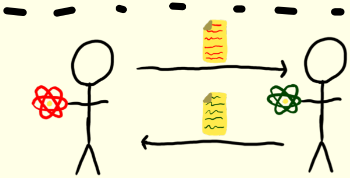
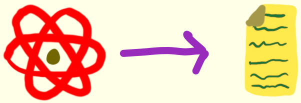
And so this gives a construction of state-puzzles (and thus one-way puzzles) from just a single random state floating in the air.

Recipe #3: State Puzzles



And this technique also works for a bunch of important cryptographic primitives (like quantum money or 2-round key distribution)

Recipes Recap

Ingredient	Recipe
	shadow tomography
	universal extrapolation/approx.
	state puzzles

To recap, we gave three recipes for three different types of protocols.
The first recipe is shadow tomography, which works when there are classical secrets and (many) quantum outputs.

The second is universal extrapolation/approximation, which works for anything with classical communication.

And the last is state puzzles, which works when there are quantum secrets and classical outputs.

Recipes Recap

all doable worst-case in #P!

<u>Recipe</u>

shadow tomography

universal extrapolation/approx.

state puzzles

Note that all three of these techniques are solvable in the worst-case by an algorithm in the counting complexity class #P (a complete problem is counting the number of satisfying assignments of a SAT formulat)

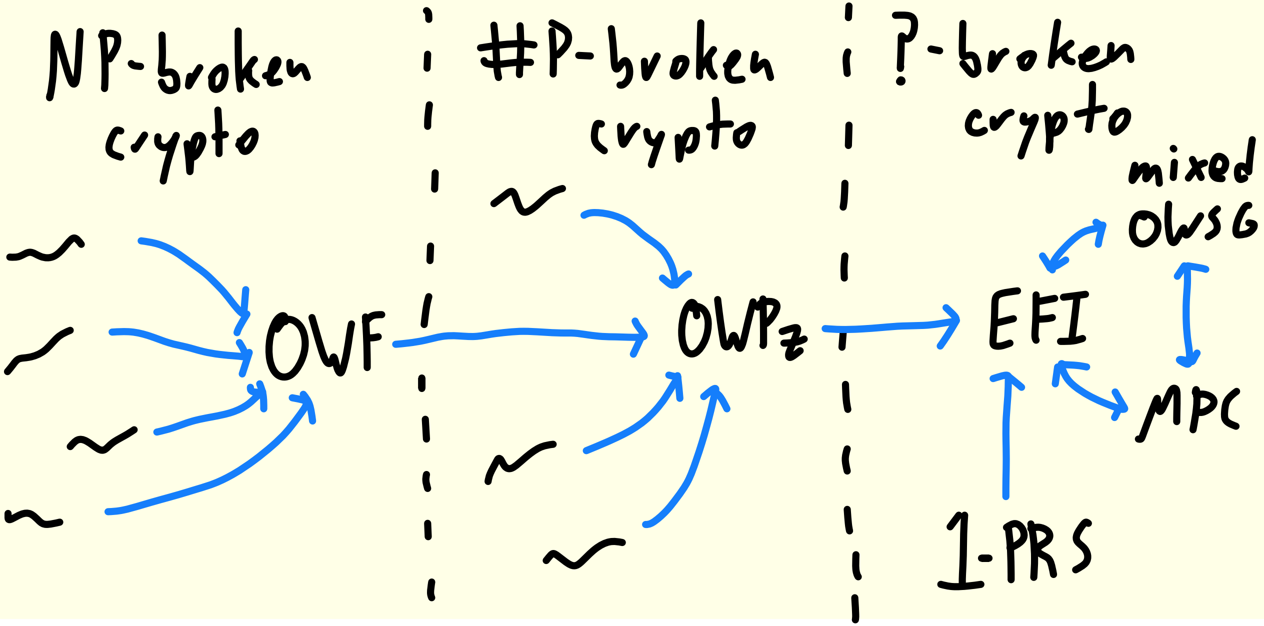
Recipes Recap

Counting Heuristic

If a (cryptographic) object can be broken in $P^{\#P}$, then it implies one-way puzzles.

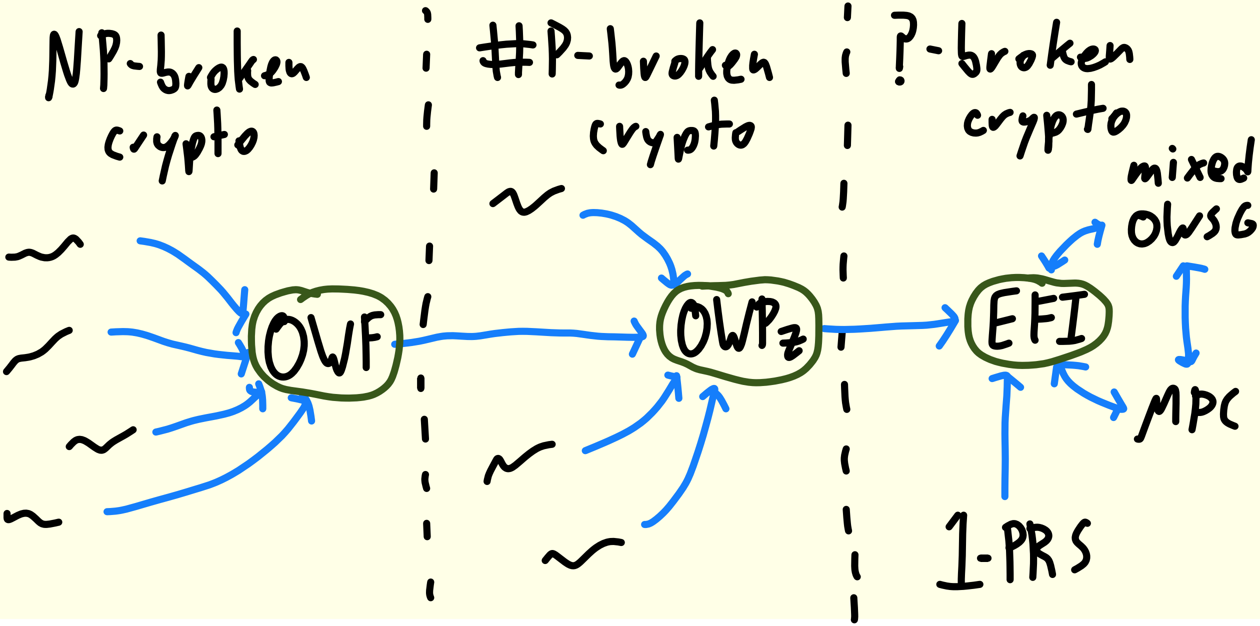
This leads to the following heuristic, for which I know no counterexamples. If a cryptographic object can be broken in P^#P, then it implies one-way puzzles.

Central Primitives



As a final view, we can divide crypto up into three categories, crypto broken in NP, broken in #P, and crypto we just don't know how to break.

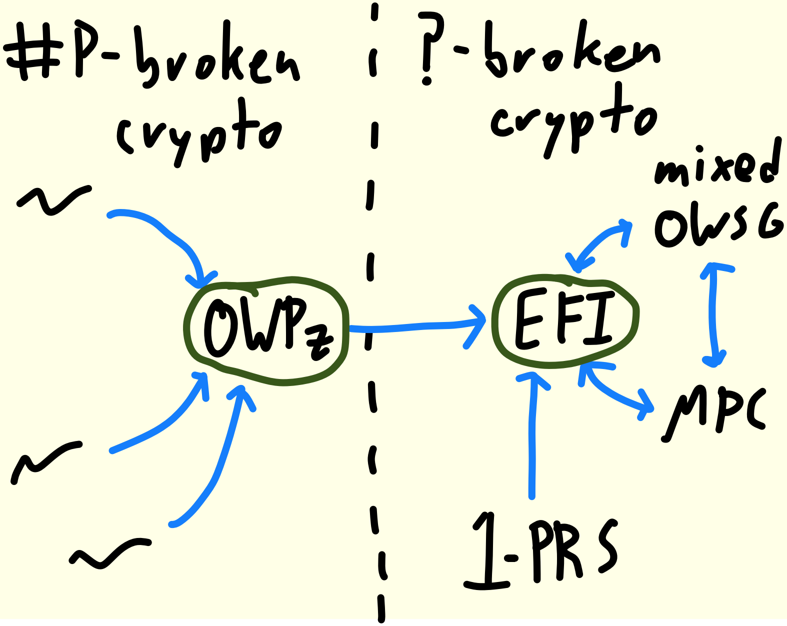
Central Primitives



Each of these worlds has a minimal primitive, OWF for NP, OWPz for #P, and EFI for the rest

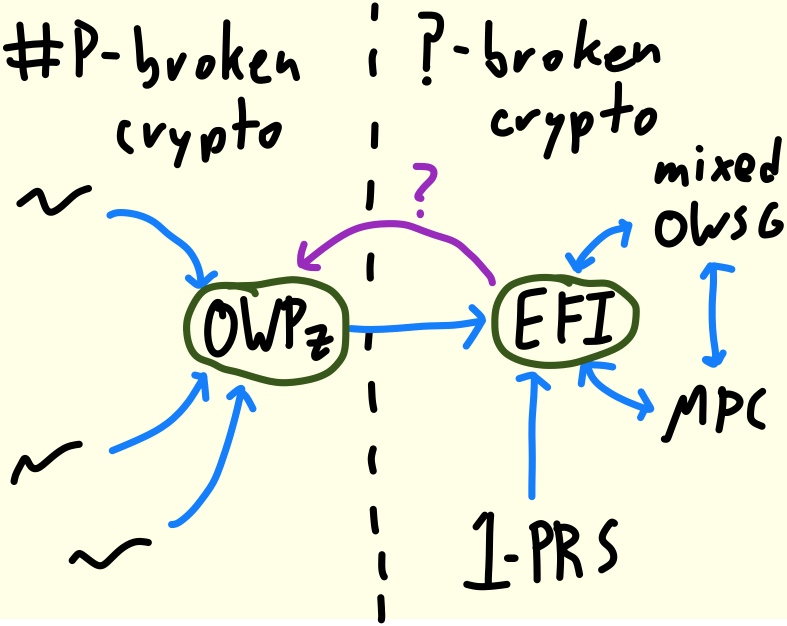
Open Questions

Connections to EFI



The first, glaring open problem is how OWPuzz relate to EFI. We know that one-way puzzles are broken by #P oracles, and we don't know of any classical oracle that can break EFI pairs

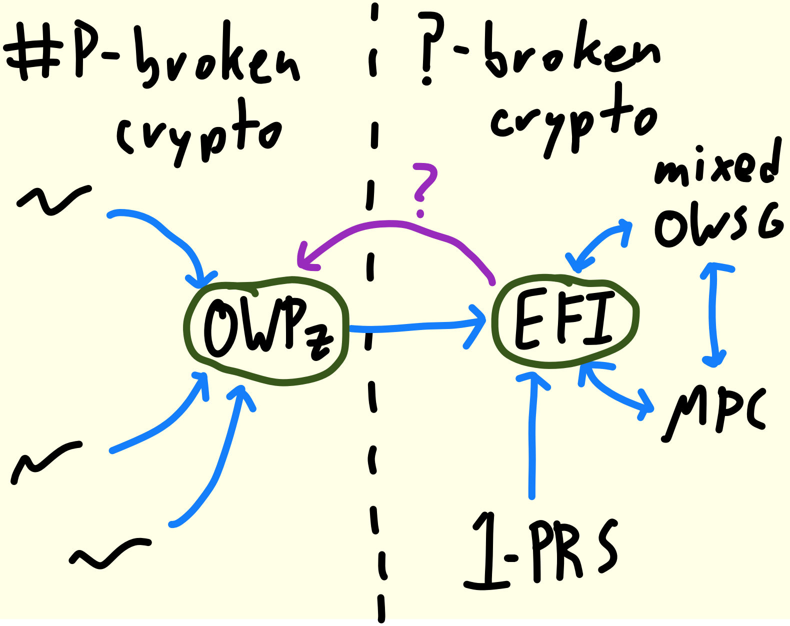
Connections to EFI



Connections to EFI

Only weak (1-query) barriers!

[LMW24]



And yet, we have no separations between EFI pairs and one-way puzzles. The best we have is a separation where we are restricted to making a single query to the oracle, which doesn't actually give a black box separation!

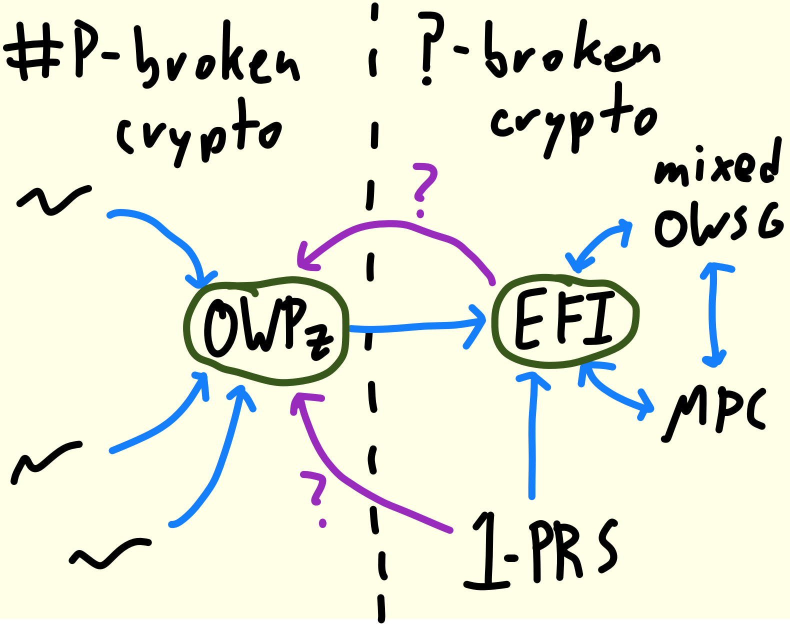
EFI pairs are the main candidate for showing that the unitary synthesis problem is hard relative to classical oracles, and a prerequisite for that is to separate from one way puzzles.

The other possibility is that we can actually construct one-way puzzles from EFI, which would really simplify the picture of quantum minimality, and actually such a construction wouldn't even contradict the unitary synthesis conjecture.

Connections to EFI

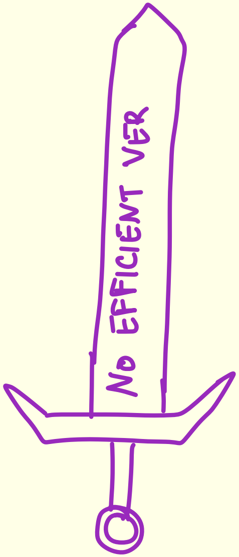
Only weak (1-query) barriers!

[LMW24]



An easier thing to try is building one-way puzzles from single copy PRS, which seem stronger than EFI, but we don't know how to build anything useful from them, other than EFI

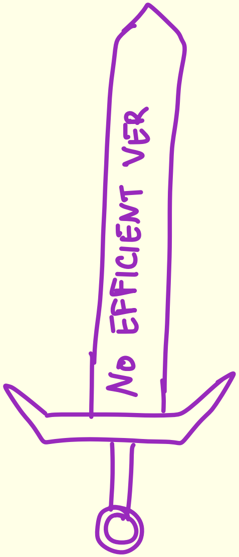
Using one-way puzzles



The second question is about using one-way puzzles. OWPuzz are defined with the double edged sword of NO EFF VER.

Using one-way puzzles

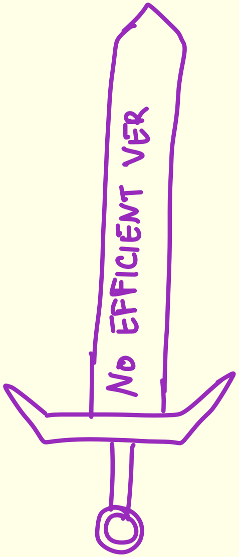
Requires very weak assumptions (maybe just $BQP \neq \#P$)



This is good because it lets us build them from very very weak assumptions, maybe even as weak as the worst case quantum hardness of #P.

Using one-way puzzles

Requires very weak assumptions (maybe just BQP≠#P)

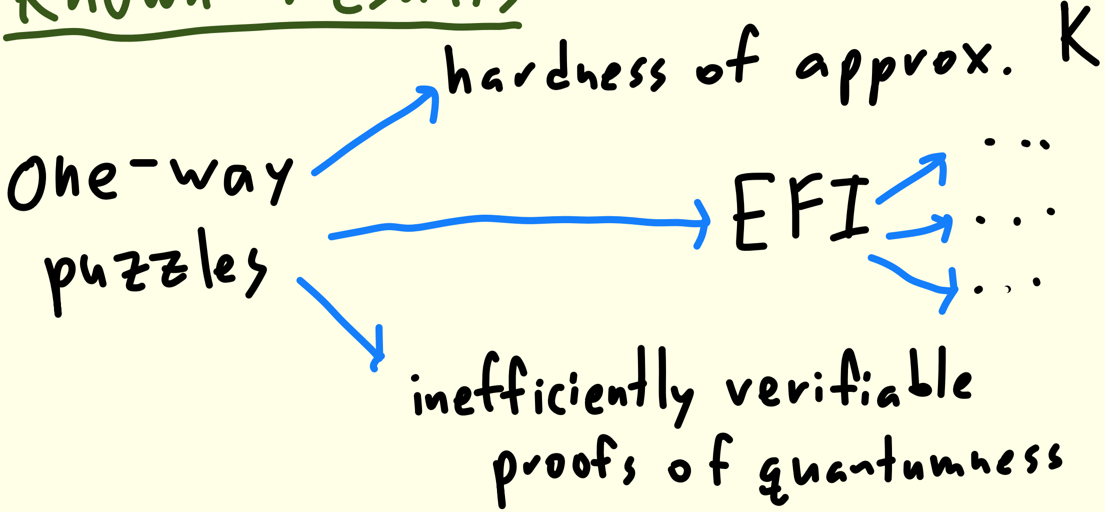


Hard to build other primitives from it

The downside is this makes it harder to build other primitives from it

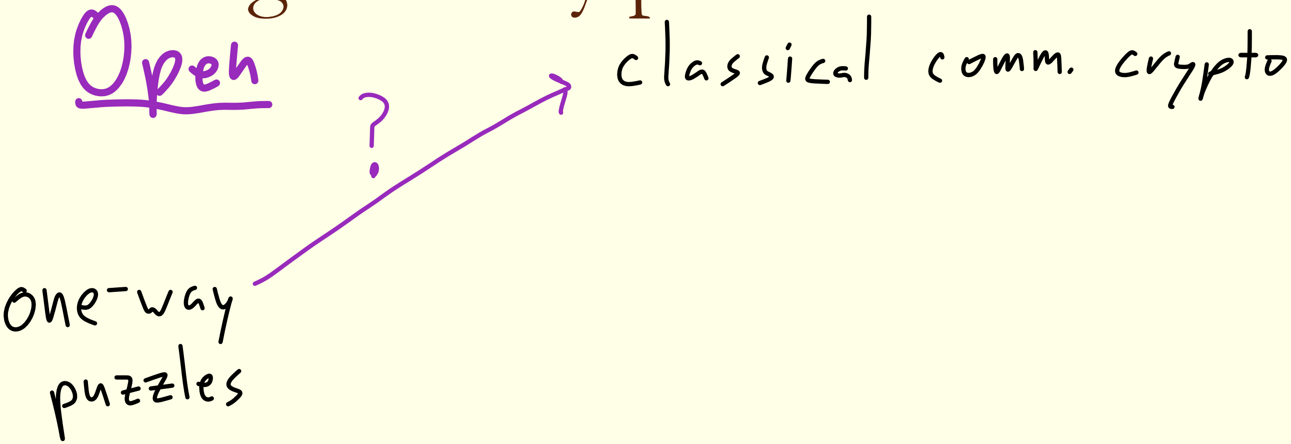
Using one-way puzzles

Known results



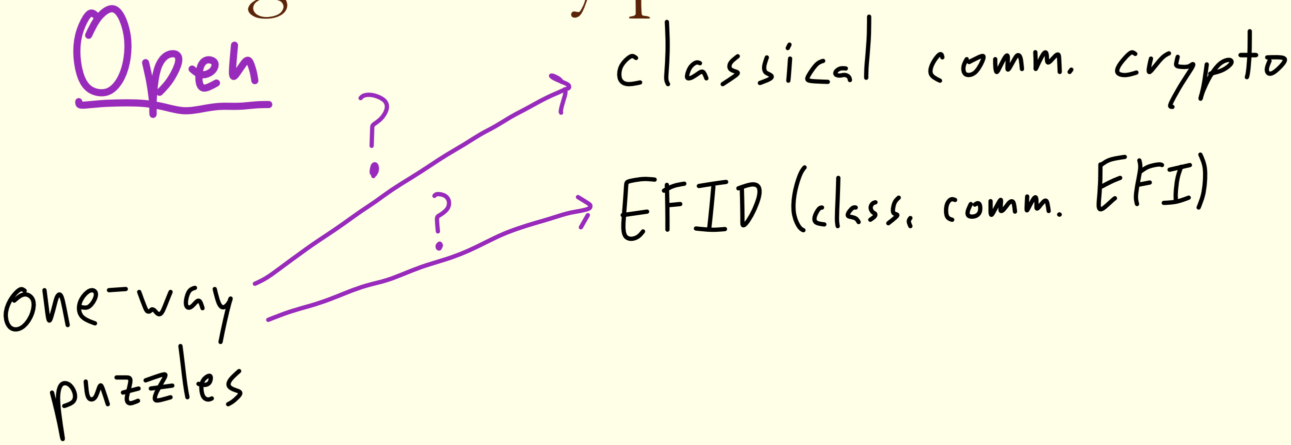
The known results are that one-way puzzles imply the hardness of approximating K complexity, they imply IV-PoQ, they imply EFI, and they imply everything else implied by EFI

Using one-way puzzles



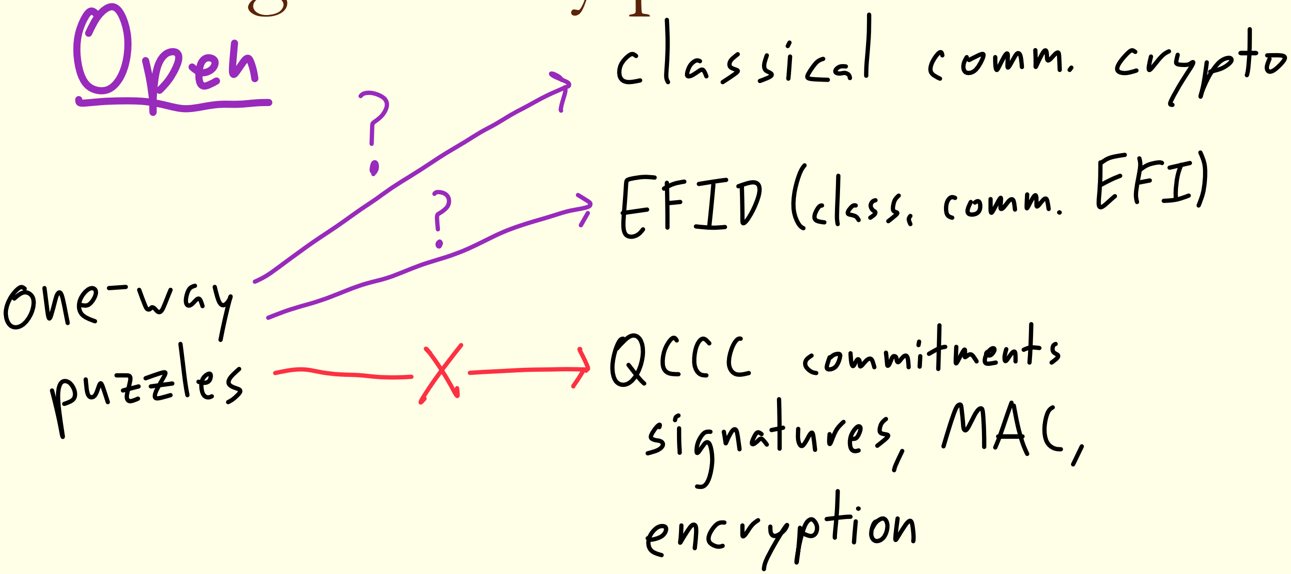
A natural thing to try to build from OWP is a classical communication cryptography

Using one-way puzzles



Specifically, we can consider a classical version of EFI called an EFID where both distributions are just classical. These we can almost build from one-way puzzles, except that the construction has a small amount of non-uniformity, and we don't really know how to get rid of it.

Using one-way puzzles



Open

one-way puzzles

?

classical comm. crypto

?

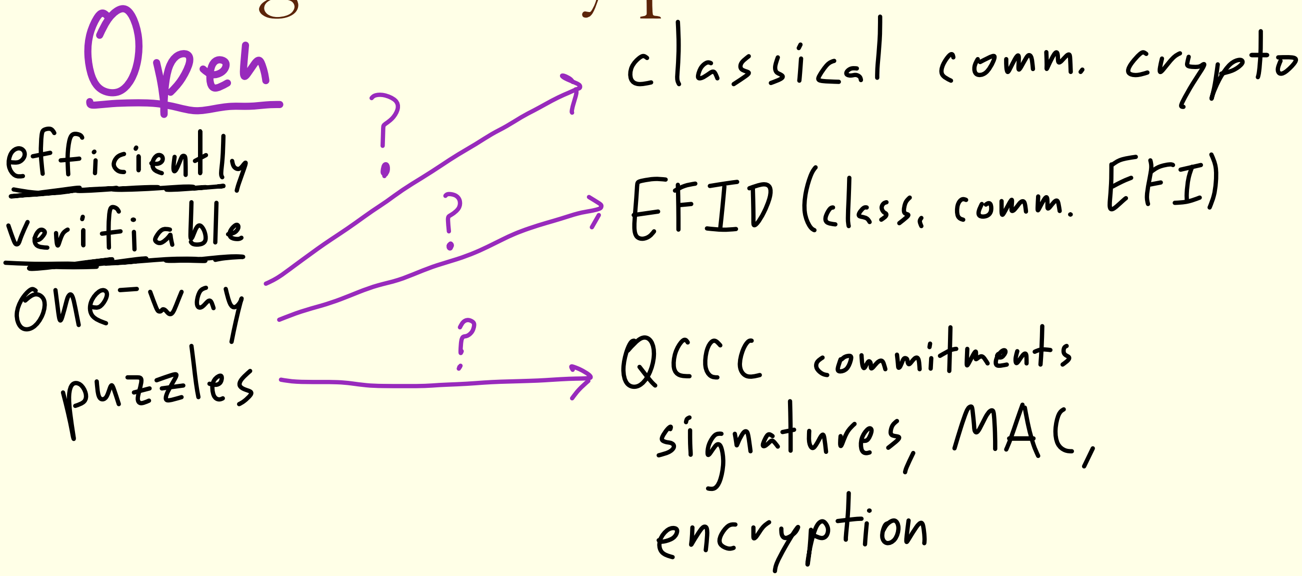
EFID (class. comm. EFI)

X

QCCC commitments signatures, MAC, encryption

But for other primitives, there are barriers towards constructing them from one-way puzzles, and this is again because there is no efficient verification algorithm.

Using one-way puzzles



Open

efficiently verifiable one-way puzzles

?

classical comm. crypto

?

EFID (class. comm. EFI)

?

QCCC commitments signatures, MAC, encryption

So…what if we added efficient verification? We still get a primitive weaker than one-way functions, it is still implied by all QCCC cryptography….can we build more cryptographic primitives? Specifically, I think there should be a way to get MACs or even signatures from these, and the barriers we encounter are very classical in nature, you don't really need a to know anything about quantum computing to understand them.

Quantum Advantage

Concrete quantum advantage assumptions

+ $\mathrm{BQP} \neq \mathrm{PP}$

$\downarrow$

OWPuzz

Another intriguing observation: there seem to be deep connections between quantum advantage and one-way puzzles. One connection is that the same conjectures that are used to show quantum advantage in leading experiments also imply one-way puzzles

Quantum Advantage

Concrete quantum advantage assumptions

+ BQP $\neq$ PP

↓ [KT25]

OWPuzz

(Inefficiently verifiable) quantum advantage

↕ [MYY25]

(classically secure) OWPuzz

Another intriguing observation: there seem to be deep connections between quantum advantage and one-way puzzles. One connection is that the same conjectures that are used to show quantum advantage in leading experiments also imply one-way puzzles

Quantum Advantage

Concrete quantum advantage assumptions

+ $BQP \neq PP$

↓ [KT25]

OWPuzz

(Inefficiently verifiable) quantun advantage

↕ [MYY25]

(classically secure) OWPuzz

Deeper Connections?

Another intriguing observation: there seem to be deep connections between quantum advantage and one-way puzzles. One connection is that the same conjectures that are used to show quantum advantage in leading experiments also imply one-way puzzles

Metacomplexity

Hardness of approximating $K(\cdot)$

↕ [CGG+25, HM25]

OWPuzz

Metacomplexity

Hardness of approximating $K(\cdot)$

↕ [CGG+25, HM25]

OWPuzz

No other connections!

Metacomplexity

Hardness of approximating $K(\cdot)$

↕ [CGG+25, HM25]

OWPuzz

No other connections!

Most metacomplexity problems in NP

Metacomplexity

Hardness of approximating $K(\cdot)$

↕ [CGG+25, HM25]

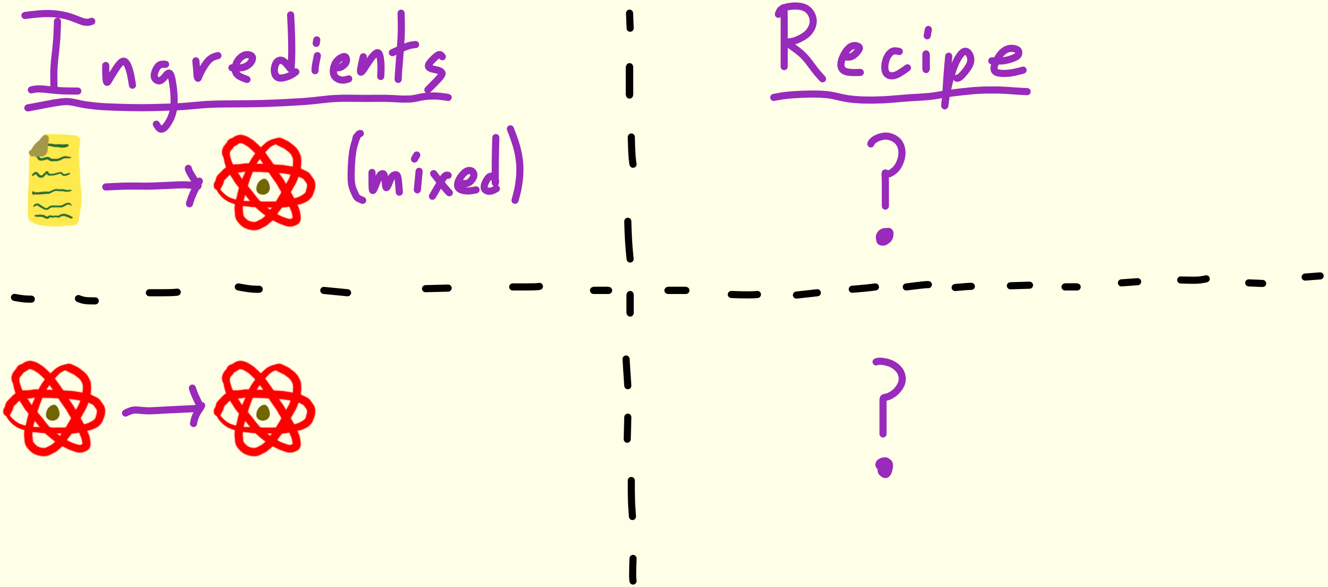
OWPuzz

No other connections!

Most metacomplexity problems in NP

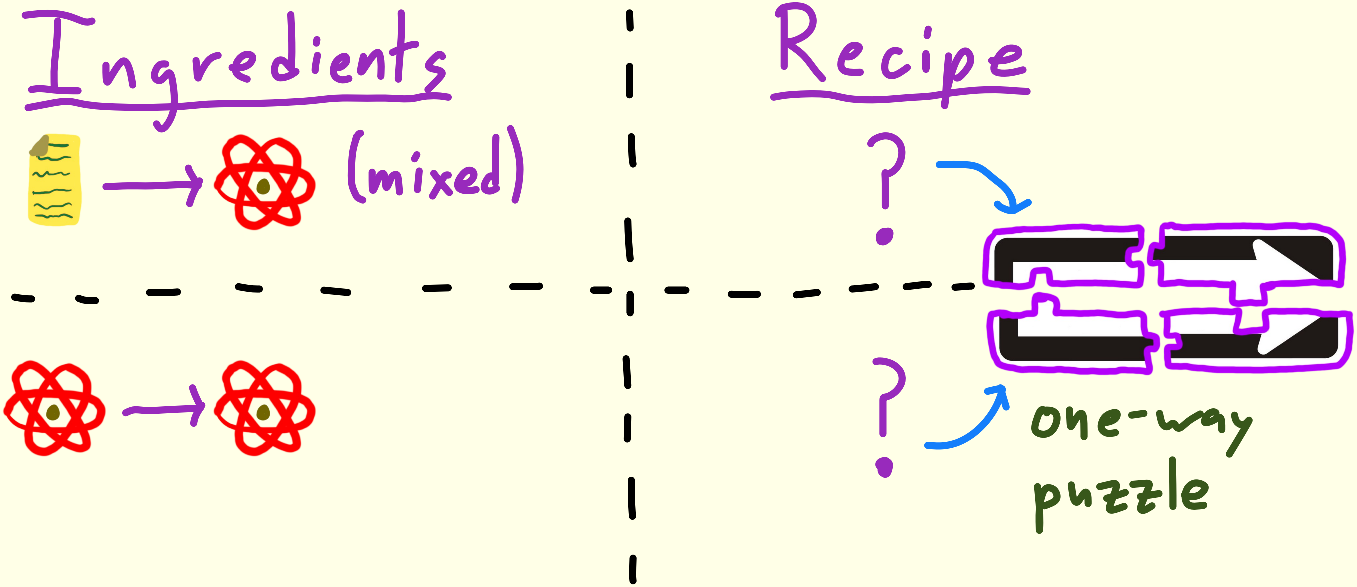
Need new quantum metacomplexity notions

Missing Recipes



Another, more abstract question is whether we can expand our repertoire of recipes.

Missing Recipes

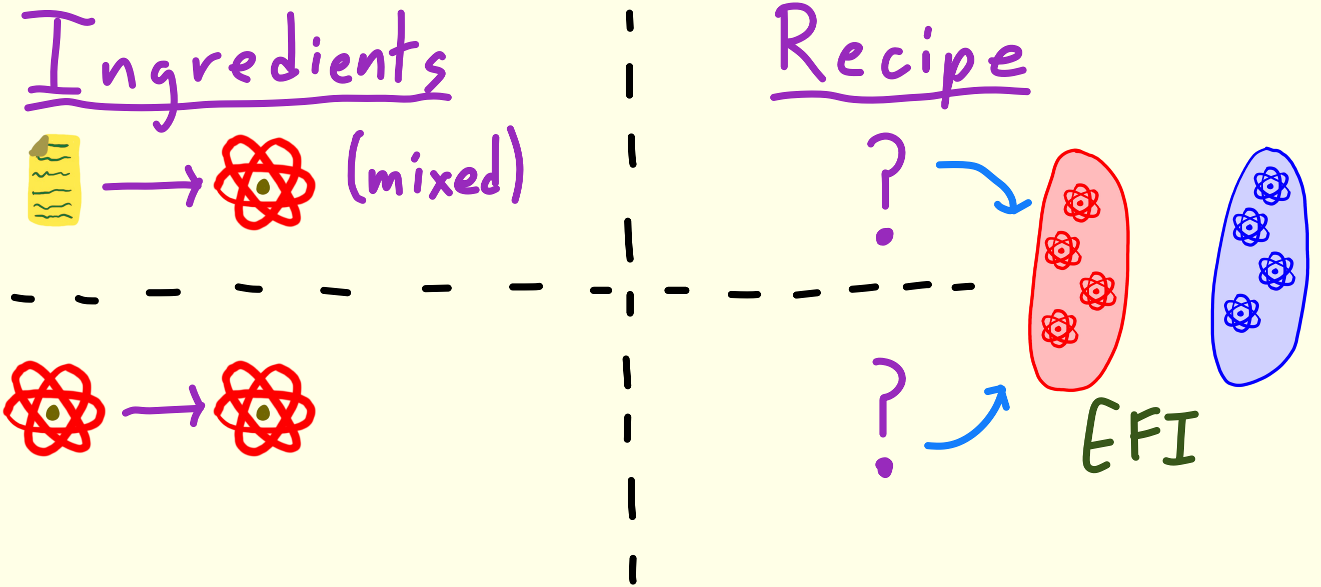


There are two obvious types of hardness that we can imagine but don’t know how to construct puzzles from

…

The main problem here is we don’t really have many primitives which fall into these categories, so its challenging to think of the right definitions for these

Missing Recipes



In both cases, when we can't construct puzzles we usually don't know how to construct EFI either