Categories for Cryptographic Composability

Riley Shahar
Advised by Angélica Osorno and Adam Groce

Outline

Outline

- Cryptographic composability

Outline

- Cryptographic composability
- Why categories?

Outline

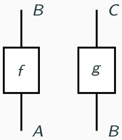
- Cryptographic composability
- Why categories?
- Towards a categorical theory of cryptography

Outline

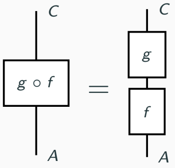
- Cryptographic composability
- Why categories?
- Towards a categorical theory of cryptography
- Open problems

Composition

Composition

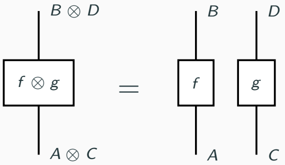


Composition



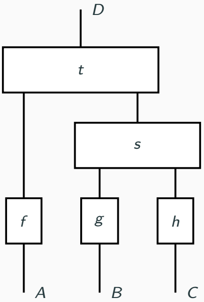
Sequential (Vertical) Composition

Composition



Parallel (Horizontal) Composition

Composition



Cryptography

Cryptography is *the mathematics of secure computation.*

Cryptography

Cryptography is *the mathematics of secure computation.*

Cryptography

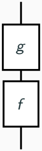
Cryptography is *the mathematics of secure computation.*

Cryptographic Composability

Say f and g are secure.

Cryptographic Composability

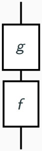
Say f and g are secure.

Is 

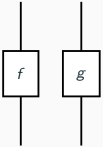
?

Cryptographic Composability

Say f and g are secure.

Is 

?

Is 

?

Commitment

Alice wants to make a binding guess to Bob, but not reveal it yet.

Commitment

Alice wants to make a binding guess to Bob, but not reveal it yet.

She wants a **commitment protocol**.

Commitment

Alice wants to make a binding guess to Bob, but not reveal it yet.
She wants a **commitment protocol**.

We can use this to play rock-paper-scissors:

Commitment

Alice wants to make a binding guess to Bob, but not reveal it yet.
She wants a **commitment protocol**.

We can use this to play rock-paper-scissors:

1. Alice commits $a \in \{R, P, S\}$

Commitment

Alice wants to make a binding guess to Bob, but not reveal it yet. She wants a **commitment protocol**.

We can use this to play rock-paper-scissors:

1. Alice commits $a \in \{R, P, S\}$
2. Bob commits $b \in \{R, P, S\}$

Commitment

Alice wants to make a binding guess to Bob, but not reveal it yet.
She wants a **commitment protocol**.

We can use this to play rock-paper-scissors:

1. Alice commits $a \in \{R, P, S\}$
2. Bob commits $b \in \{R, P, S\}$
3. Alice reveals a

Commitment

Alice wants to make a binding guess to Bob, but not reveal it yet. She wants a **commitment protocol**.

We can use this to play rock-paper-scissors:

1. Alice commits $a \in \{R, P, S\}$
2. Bob commits $b \in \{R, P, S\}$
3. Alice reveals a
4. Bob reveals b

Game-based security

Game-based security

List the properties we want our protocol to have.

Game-based security

List the properties we want our protocol to have.

Secure commitment should be

Game-based security

List the properties we want our protocol to have.

Secure commitment should be *hiding and binding*.

Game-based security

List the properties we want our protocol to have.

Secure commitment should be *hiding and binding*.

The Hiding Game

1. Bob picks m_0, m_1.

Game-based security

List the properties we want our protocol to have.

Secure commitment should be *hiding and binding*.

The Hiding Game

1. Bob picks m_0, m_1.
2. Alice commits m_b at random.

Game-based security

List the properties we want our protocol to have.

Secure commitment should be *hiding and binding*.

The Hiding Game

1. Bob picks m_0, m_1.
2. Alice commits m_b at random.
3. Bob guesses b'.

Game-based composability

The Hiding Game

1. Bob picks two messages
2. Alice commits one
3. Bob guesses which one

Rock-Paper-Scissors

1. Alice commits $a \in \{R, P, S\}$
2. Bob commits $b \in \{R, P, S\}$
3. They both reveal

Game-based composability

The Hiding Game

1. Bob picks two messages
2. Alice commits one
3. Bob guesses which one

Rock-Paper-Scissors

1. Alice commits $a \in \{R, P, S\}$
2. Bob commits $b \in \{R, P, S\}$
3. They both reveal

Many natural commitment protocols suffer from *malleability.*

Game-based composability

The Hiding Game

1. Bob picks two messages
2. Alice commits one
3. Bob guesses which one

Rock-Paper-Scissors

1. Alice commits $a \in \{R, P, S\}$
2. Bob commits $b \in \{R, P, S\}$
3. They both reveal

Many natural commitment protocols suffer from *malleability.*

It's taken *decades* for a missing property to be noticed [PW91; BHL05]!

Simulation-based security

Simulation-based security

Compare the protocol to an *ideal world* with a trusted third party.

Simulation-based security

Compare the protocol to an *ideal world* with a trusted third party.

Two probability distributions are *computationally indistinguishable* if poly-time algorithms cannot tell between them with nontrivial probability.

Simulation-based security

Compare the protocol to an *ideal world* with a trusted third party.

Two probability distributions are *computationally indistinguishable* if poly-time algorithms cannot tell between them with nontrivial probability.

A protocol is *secure* if it is computationally indistinguishable from the ideal world.

Simulation-based security

Compare the protocol to an *ideal world* with a trusted third party.

Two probability distributions are *computationally indistinguishable* if poly-time algorithms cannot tell between them with nontrivial probability.

A protocol is *secure* if it is computationally indistinguishable from the ideal world.

Simulation-based composability

Simulation-based composability

Theorem [MR92]. *Simulation-secure protocols compose securely in sequences of polynomial length.*

Simulation-based composability

Theorem [MR92]. *Simulation-secure protocols compose securely in sequences of polynomial length.*

However, [GK96] gave a protocol for zero-knowledge proof that's simulation secure, but doesn't compose in parallel.

Universal Composability

Theorem [Can00a]. *UC-secure protocols compose securely in parallel sequences of polynomial length or width.*

Universal Composability

Theorem [Can00a]. *UC-secure protocols compose securely in parallel sequences of polynomial length or width.*

UC was revised in [Can00a; Can00b; Can01; Can05a; Can05b; Can13a; Can13b; Can18; Can20].

Universal Composability

Theorem [Can00a]. *UC-secure protocols compose securely in parallel sequences of polynomial length or width.*

UC was revised in [Can00a; Can00b; Can01; Can05a; Can05b; Can13a; Can13b; Can18; Can20].

The proofs...

- are wildly dependent on small technical details;

Theorem [Can00a]. *UC-secure protocols compose securely in parallel sequences of polynomial length or width.*

UC was revised in [Can00a; Can00b; Can01; Can05a; Can05b; Can13a; Can13b; Can18; Can20].

The proofs...

- are wildly dependent on small technical details;
- leave artifacts in the protocol;

Universal Composability

Theorem [Can00a]. *UC-secure protocols compose securely in parallel sequences of polynomial length or width.*

UC was revised in [Can00a; Can00b; Can01; Can05a; Can05b; Can13a; Can13b; Can18; Can20].

The proofs...

- are wildly dependent on small technical details;
- leave artifacts in the protocol;
- are very hard to trust.

Cryptography is in need of an elegant mathematical theory abstracting composability of computational processes...

Cryptography is in need of an elegant mathematical theory abstracting composability of computational processes. . .

. . . **category theory** is an excellent candidate for such a theory.

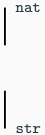
Categories and Programming Languages

In programming language theory:

Categories and Programming Languages

In programming language theory:

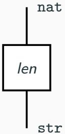
- objects are types;



Categories and Programming Languages

In programming language theory:

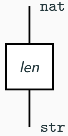
- objects are types;
- morphisms are programs.



Categories and Programming Languages

In programming language theory:

- objects are types;
- morphisms are programs.



(We've already been doing category theory!)

Composing Programs

Consider programs

and


.

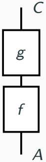
Composing Programs

Consider programs

and

.

We can *always* make a program



.

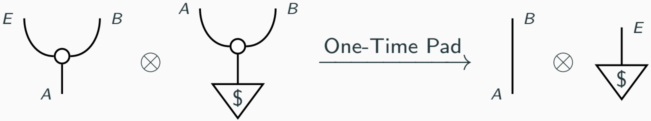
Composition "works" in PL theory.

Categorical Cryptography: The Idea

Make a relation $\approx$ between morphisms, roughly like indistinguishability.

Categorical Cryptography: The Idea

Make a relation $\approx$ between morphisms, roughly like indistinguishability.

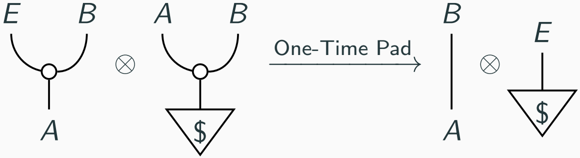


Categorical Composable Cryptography

As far as I know, there is only one published paper using category theory for composability in cryptography [BK22].

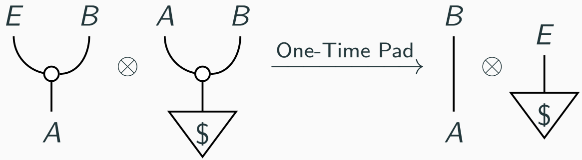
Categorical Composable Cryptography

As far as I know, there is only one published paper using category theory for composability in cryptography [BK22].



Categorical Composable Cryptography

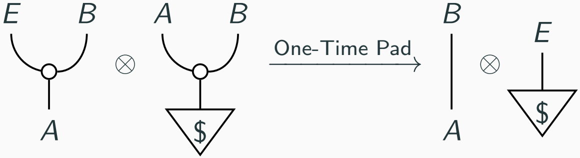
As far as I know, there is only one published paper using category theory for composability in cryptography [BK22].



The moral:

Categorical Composable Cryptography

As far as I know, there is only one published paper using category theory for composability in cryptography [BK22].

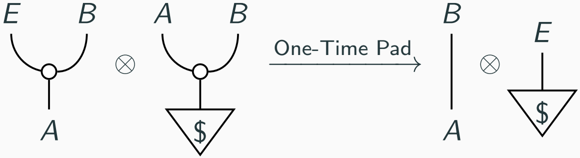


The moral:

- objects are resources, like channels or keys;

Categorical Composable Cryptography

As far as I know, there is only one published paper using category theory for composability in cryptography [BK22].

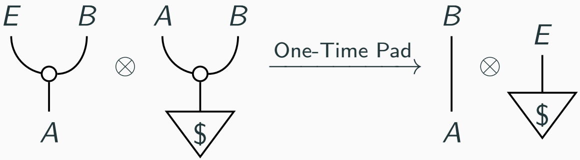


The moral:

- objects are resources, like channels or keys;
- morphisms are "protocols with holes";

Categorical Composable Cryptography

As far as I know, there is only one published paper using category theory for composability in cryptography [BK22].

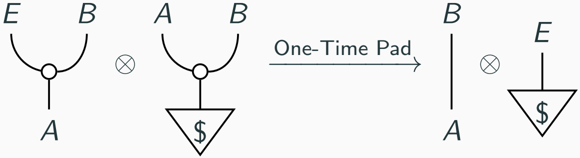


The moral:

- objects are resources, like channels or keys;
- morphisms are "protocols with holes";
- composition "plugs in the holes".

Categorical Composable Cryptography

As far as I know, there is only one published paper using category theory for composability in cryptography [BK22].



The moral:

- objects are resources, like channels or keys;
- morphisms are "protocols with holes";
- composition "plugs in the holes".

This is the standard paradigm in programming language theory, transported to cryptographic language.

Attack Models

In CCC, adversaries are constrained by *attack models*.

Attack Models

In CCC, adversaries are constrained by *attack models*.

An attack model $\mathcal{A}$ assigns, to each morphism f, a collection of morphisms $\mathcal{A}(f)$ satisfying some axioms. If the adversary is "supposed" to do f, then they can instead do anything in $\mathcal{A}(f)$.

Attack Models

In CCC, adversaries are constrained by *attack models*.

An attack model $\mathcal{A}$ assigns, to each morphism f, a collection of morphisms $\mathcal{A}(f)$ satisfying some axioms. If the adversary is "supposed" to do f, then they can instead do anything in $\mathcal{A}(f)$.

Open Question 1: *Can the axioms be formulated as functoriality plus some conditions? If $\mathcal{A}$ is a functor, what should its codomain be?*

Attack Models

In CCC, adversaries are constrained by *attack models*.

An attack model $\mathcal{A}$ assigns, to each morphism f, a collection of morphisms $\mathcal{A}(f)$ satisfying some axioms. If the adversary is "supposed" to do f, then they can instead do anything in $\mathcal{A}(f)$.

Open Question 1: *Can the axioms be formulated as functoriality plus some conditions? If $\mathcal{A}$ is a functor, what should its codomain be?*

Open Question 2: *How broad is the definition of an attack model? Does it capture enough of modern cryptography?*

Computational Indistinguishability

Composition should only work polynomially many times.

Computational Indistinguishability

Composition should only work polynomially many times. In fact, computational indistinguishability is not even an equivalence relation.

Computational Indistinguishability

Composition should only work polynomially many times. In fact, computational indistinguishability is not even an equivalence relation.

B&K work around this by artificially limiting the universe size.

Computational Indistinguishability

Composition should only work polynomially many times. In fact, computational indistinguishability is not even an equivalence relation.

B&K work around this by artificially limiting the universe size.

Open Question 3: *Is there a natural model of computational indistinguishability in nice symmetric monoidal categories?*

Computational Indistinguishability

Composition should only work polynomially many times. In fact, computational indistinguishability is not even an equivalence relation.

B&K work around this by artificially limiting the universe size.

Open Question 3: *Is there a natural model of computational indistinguishability in nice symmetric monoidal categories?*

B&K propose attaching an extended metric to the category, interpreted as the "computational distance" between two morphisms.

Computational Indistinguishability

Composition should only work polynomially many times. In fact, computational indistinguishability is not even an equivalence relation.

B&K work around this by artificially limiting the universe size.

Open Question 3: *Is there a natural model of computational indistinguishability in nice symmetric monoidal categories?*

B&K propose attaching an extended metric to the category, interpreted as the "computational distance" between two morphisms.

We need some way to deal with asymptotic behavior. Our current idea is to value the metric in $\mathbb{R}^{\mathbb{N}}$.

Enrichment

Many of the constructions of B&K enhance the category with extra structure.

Enrichment

Many of the constructions of B&K enhance the category with extra structure.

Open Question 4: *Do their composition theorems extend to the context of enriched category theory?*

Many of the constructions of B&K enhance the category with extra structure.

Open Question 4: *Do their composition theorems extend to the context of enriched category theory? If so, can the extra structure they need be framed as some kind of enrichment?*

Conclusion

CCC is very important progress, but there's a lot to be done.

Conclusion

CCC is very important progress, but there's a lot to be done. CCC largely staples cryptographic notions on top of a category, rather than working with the machinery. We'd like to try to improve on this.

Conclusion

CCC is very important progress, but there's a lot to be done. CCC largely staples cryptographic notions on top of a category, rather than working with the machinery. We'd like to try to improve on this.

Once we have a more natural model, there will be lots of cool applications:

Conclusion

CCC is very important progress, but there's a lot to be done. CCC largely staples cryptographic notions on top of a category, rather than working with the machinery. We'd like to try to improve on this.

Once we have a more natural model, there will be lots of cool applications:

- Can we incorporate categorical notions from game theory, programming languages, etc. into cryptography?

Conclusion

CCC is very important progress, but there's a lot to be done. CCC largely staples cryptographic notions on top of a category, rather than working with the machinery. We'd like to try to improve on this.

Once we have a more natural model, there will be lots of cool applications:

- Can we incorporate categorical notions from game theory, programming languages, etc. into cryptography?
- What does the presence of various categorical structure ((co)limits, monads, etc.) say cryptographically?