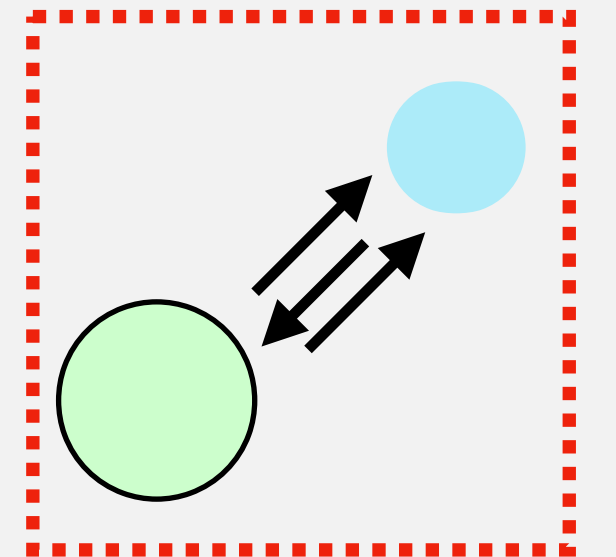


# $\Sigma$ -Protocols and Fiat-Shamir

*Michele Orrù (CNRS), Cathie Yun (Apple)*

`draft-irtf-cfrg-sigma-protocols`  
`draft-irtf-cfrg-fiat-shamir`



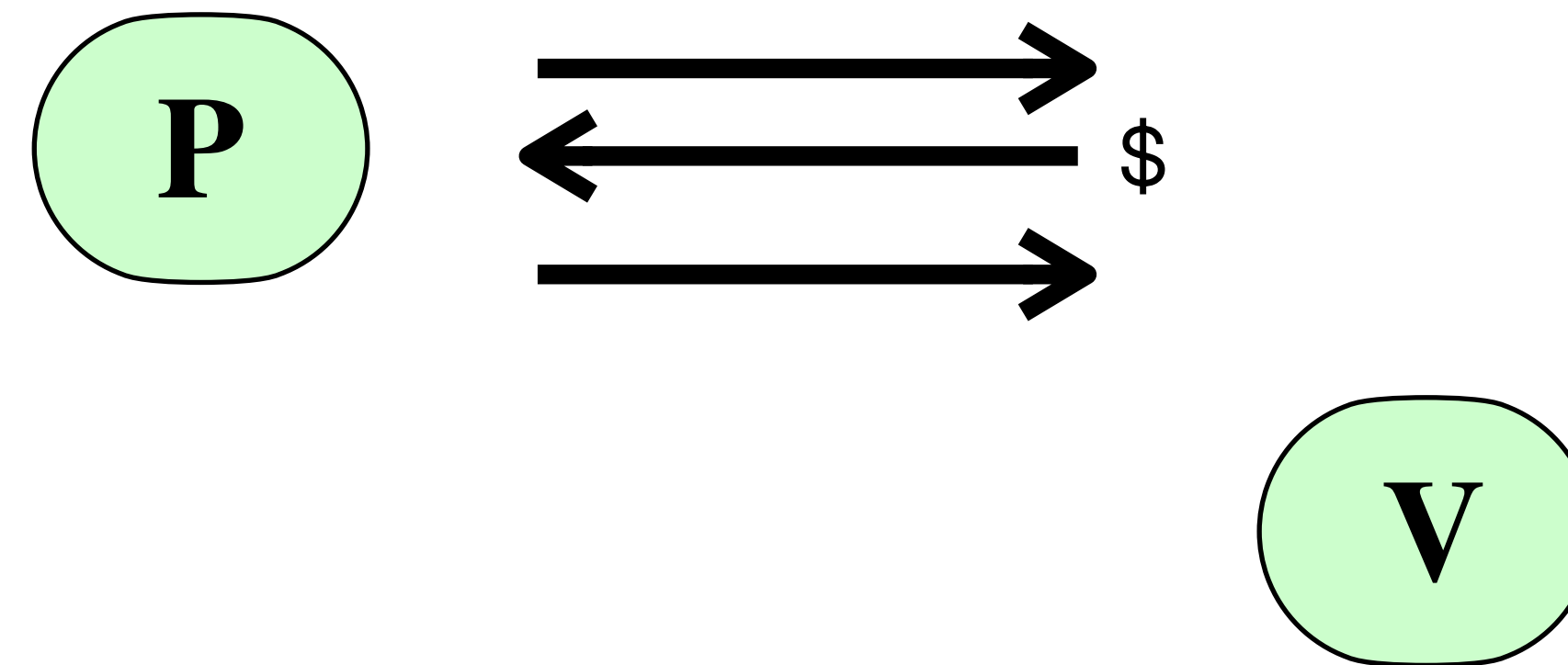
# $\Sigma$ -Protocols and Fiat-Shamir

$\Sigma$ -**protocol** are zero-knowledge interactive proofs for simple relations.

**Fiat-Shamir** transforms the interactive protocol into the non-interactive one with a hash function.

# $\Sigma$ -Protocols and Fiat-Shamir

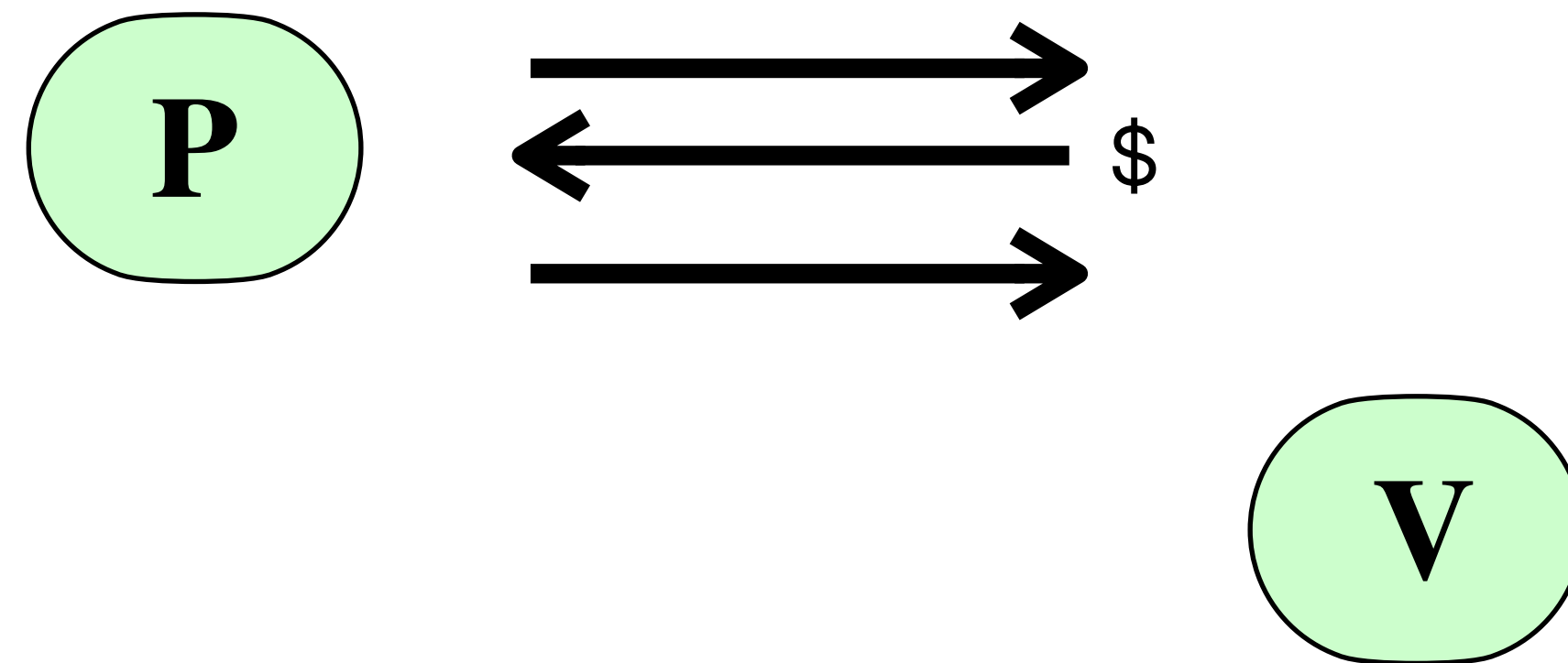
$\Sigma$ -**protocol** are zero-knowledge interactive proofs for simple relations.



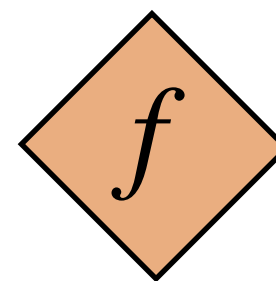
**Fiat-Shamir** transforms the interactive protocol into the non-interactive one with a hash function.

# $\Sigma$ -Protocols and Fiat-Shamir

$\Sigma$ -**protocol** are zero-knowledge interactive proofs for simple relations.

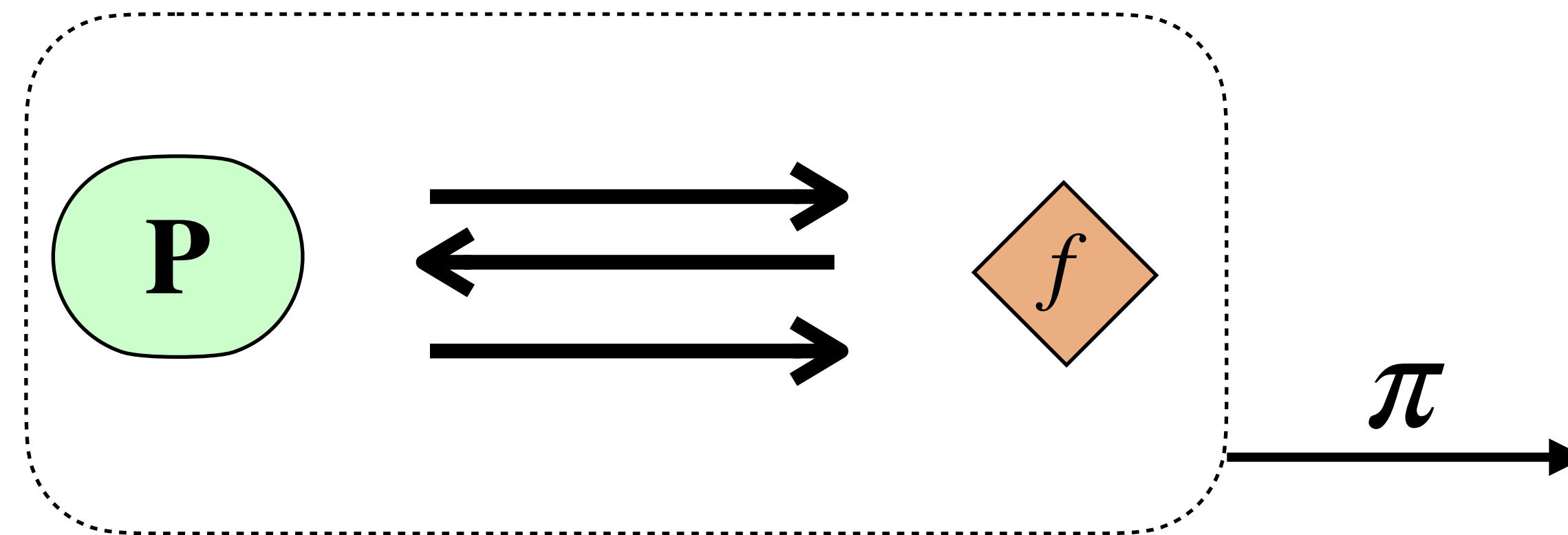


**Fiat-Shamir** transforms the interactive protocol into the non-interactive one with a hash function.



# $\Sigma$ -Protocols and Fiat-Shamir

$\Sigma$ -**protocol** are zero-knowledge interactive proofs for simple relations.



**Fiat-Shamir** transforms the interactive protocol into the non-interactive one with a hash function.

# $\Sigma$ -protocols are specified in: draft-irtf-cfrg-sigma-protocols-02

Crypto Forum  
Internet-Draft  
Intended status: Informational  
Expires: 3 September 2026

M. Orrù  
CNRS  
C. Yun  
Apple, Inc.  
2 March 2026

Interactive Sigma Proofs  
draft-irtf-cfrg-sigma-protocols-02

## Abstract

A Sigma Protocol is an interactive zero-knowledge proof of knowledge that allows a prover to convince a verifier of the validity of a statement. It satisfies the properties of completeness, soundness, and zero-knowledge, as described in Section 3.

This document describes Sigma Protocols for proving knowledge of pre-images of linear maps in prime-order elliptic curve groups. Examples include zero-knowledge proofs for discrete logarithm relations, ElGamal encryptions, Pedersen commitments, and range proofs.

## About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://mmaker.github.io/draft-irtf-cfrg-sigma-protocols/draft-irtf-cfrg-sigma-protocols.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-irtf-cfrg-sigma-protocols/>.

Discussion of this document takes place on the Crypto Forum Research Group mailing list (<mailto:cfrg@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/cfrg>. Subscribe at <https://www.ietf.org/mailman/listinfo/cfrg/>.

# $\Sigma$ -protocols are specified in: draft-irtf-cfrg-sigma-protocols-02

Crypto Forum  
Internet-Draft  
Intended status: Informational  
Expires: 3 September 2026

M. Orrù  
CNRS  
C. Yun  
Apple, Inc.  
2 March 2026

Interactive Sigma Proofs  
draft-irtf-cfrg-sigma-protocols-02

## Abstract

A Sigma Protocol is an interactive zero-knowledge proof of knowledge that allows a prover to convince a verifier of the validity of a statement. It satisfies the properties of completeness, soundness, and zero-knowledge, as described in Section 3.

This document describes Sigma Protocols for proving knowledge of pre-images of linear maps in prime-order elliptic curve groups. Examples include zero-knowledge proofs for discrete logarithm relations, ElGamal encryptions, Pedersen commitments, and range proofs.

## About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://mmaker.github.io/draft-irtf-cfrg-sigma-protocols/draft-irtf-cfrg-sigma-protocols.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-irtf-cfrg-sigma-protocols/>.

Discussion of this document takes place on the Crypto Forum Research Group mailing list (<mailto:cfrg@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/cfrg>. Subscribe at <https://www.ietf.org/mailman/listinfo/cfrg/>.

# Fiat-Shamir is specified in: draft-irtf-cfrg-fiat-shamir-02

Crypto Forum  
Internet-Draft  
Intended status: Informational  
Expires: 3 September 2026

M. Orrù  
CNRS  
2 March 2026

Fiat-Shamir Transformation  
draft-irtf-cfrg-fiat-shamir-02

## Abstract

This document describes how to construct a non-interactive proof via the Fiat-Shamir transformation, using a generic procedure that compiles an interactive proof into a non-interactive one by relying on a stateful duplex sponge object.

The duplex sponge interface requires two methods: absorb and squeeze, which respectively read and write elements of a specified base type. The absorb operation incrementally updates the duplex sponge's internal state, while the squeeze operation produces variable-length, unpredictable outputs. This interface can be instantiated with different constructions based on permutation or compression functions.

This specification also defines codecs to securely map prover messages into the duplex sponge domain, from the duplex sponge domain into verifier messages. It also establishes how the non-interactive argument string should be serialized.

## About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://mmaker.github.io/draft-irtf-cfrg-sigma-protocols/draft-irtf-cfrg-fiat-shamir.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-irtf-cfrg-fiat-shamir-02/>.

# What happened since IETF 124: integration

Thanks to Armando Faz, Chris Patton, Chris Wood, Jonathan Katz, Michael Rosenberg, Samuel Schlesinger, Vishruti Ganesh!

# What happened since IETF 124: integration

Thanks to Armando Faz, Chris Patton, Chris Wood, Jonathan Katz, Michael Rosenberg, Samuel Schlesinger, Vishruti Ganesh!

## **Anonymous Rate-Limited Credentials**

draft-yun-privacypass-crypto-arc-00

### Abstract

This document specifies the Anonymous Rate-Limited Credential (ARC) protocol, a specialization of keyed-verification anonymous credentials with support for rate limiting. ARC credentials can be presented from client to server up to some fixed number of times, where each presentation is cryptographically bound to client secrets and application-specific public information, such that each presentation is unlinkable from the others as well as the original credential creation. ARC is useful in applications where a server needs to throttle or rate-limit access from anonymous clients.

# What happened since IETF 124: integration

Thanks to Armando Faz, Chris Patton, Chris Wood, Jonathan Katz, Michael Rosenberg, Samuel Schlesinger, Vishruti Ganesh!

## Anonymous Rate-Limited Credentials

draft-yun-privacypass-crypto-arc-00

### Abstract

This document specifies the Anonymous Rate-Limited Credential (ARC) protocol, a specialization of keyed-verification anonymous credentials with support for rate limiting. ARC credentials can be presented from client to server up to some fixed number of times, where each presentation is cryptographically bound to client secrets and application-specific public information, such that each presentation is unlinkable from the others as well as the original credential creation. ARC is useful in applications where the server needs to throttle or rate-limit access from anonymous clients.

## Anonymous Credit Tokens

draft-schlesinger-cfrg-act-00

### Abstract

This document specifies Anonymous Credit Tokens (ACT), a privacy-preserving authentication protocol that enables numerical credit systems without tracking individual clients. Based on keyed-verification anonymous credentials and privately verifiable BBS-style signatures, the protocol allows issuers to grant tokens containing credits that clients can later spend anonymously with that issuer.

# What happened since IETF 124: stabilization

# What happened since IETF 124: stabilization

- Test vectors as part of the spec

# What happened since IETF 124: stabilization

- Test vectors as part of the spec
- Kicking off the formal verification effort for the Fiat-Shamir transformation

# **spec-compatible Rust stack**

Thanks to Lindsey Tulloch, Victor Graf, Ian Goldberg!

# spec-compatible Rust stack

Thanks to Lindsey Tulloch, Victor Graf, Ian Goldberg!

```
// define credential
CMZ! { ArcCredential<G>: m1}

// issuance protocol
muCMZProtocol! { issue,,
    ArcCredential { m1: H },
}
```

# spec-compatible Rust stack

Thanks to Lindsey Tulloch, Victor Graf, Ian Goldberg!

```
// define credential
CMZ! { ArcCredential<G>: m1}

// issuance protocol
muCMZProtocol! { issue,,
    ArcCredential { m1: H },
}
```

```
// define the credential
CMZ! { NymCredential<G>: nym_secret }

// issue a credential
muCMZProtocol! { issue,,
    NymCredential { nym_secret: J },
}

// present a credential
muCMZProtocol! { present<@DOMAIN, @NYM>,
    Cred: NymCredential { nym_secret: H }, ,
    NYM = Cred.nym_secret * DOMAIN
}
```

# spec-compatible Rust stack

Thanks to Lindsey Tulloch, Victor Graf, Ian Goldberg!

```
// define credential
CMZ! { ArcCredential<G>: m1}

// issuance protocol
muCMZProtocol! { issue,,
    ArcCredential { m1: H },
}

// presentation protocol
muCMZProtocol! {
    present<nonce, @TAG, @DOMAIN>,
    Cred: ArcCredential { m1: H },,
    DOMAIN = (Cred.m1 + counter) * TAG
}
```

```
// define the credential
CMZ! { NymCredential<G>: nym_secret }

// issue a credential
muCMZProtocol! { issue,,
    NymCredential { nym_secret: J },
}

// present a credential
muCMZProtocol! { present<@DOMAIN, @NYM>,
    Cred: NymCredential { nym_secret: H }, ,
    NYM = Cred.nym_secret * DOMAIN
}
```

# **spec-compatible Rust stack**

Thanks to Lindsey Tulloch, Victor Graf, Ian Goldberg!

# spec-compatible Rust stack

Thanks to Lindsey Tulloch, Victor Graf, Ian Goldberg!

disjunction invariant

transformations

composition: and, or, threshold

Maurer proofs

compressed  $\Sigma$  protocols

Fiat-Shamir transformation

**sigma-compiler**

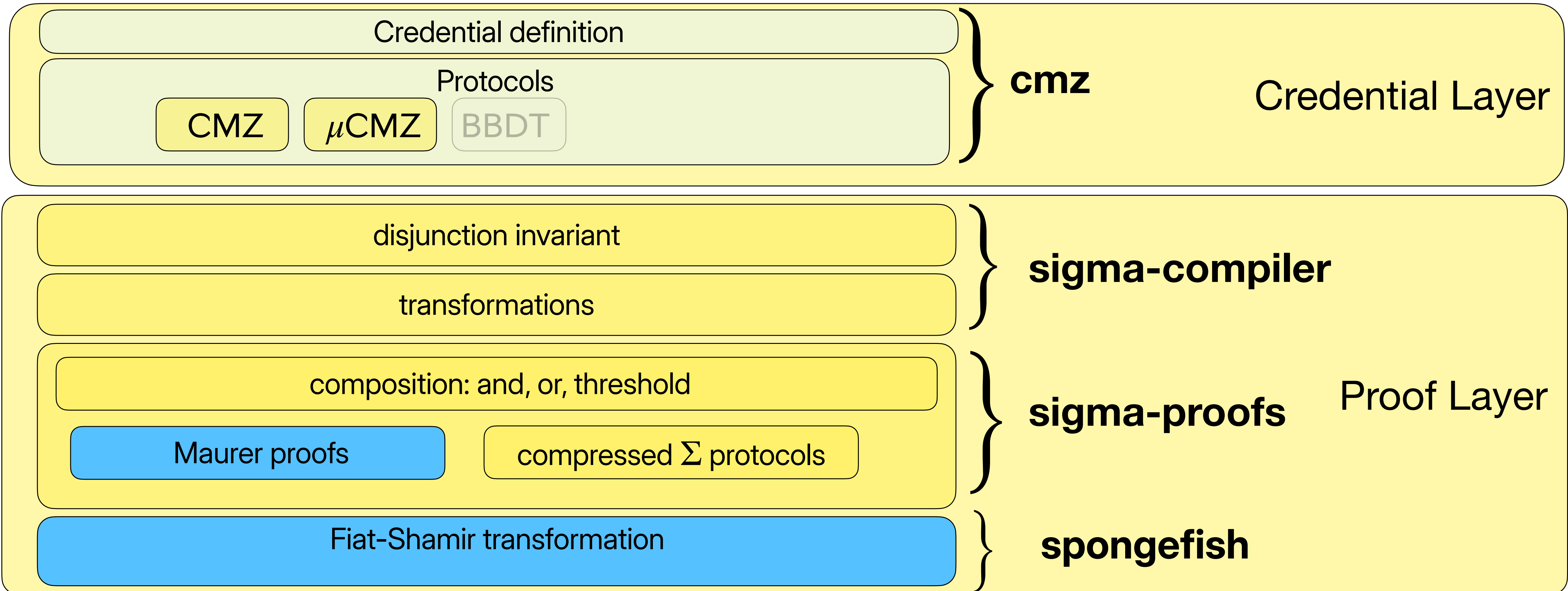
**sigma-proofs**

Proof Layer

**spongefish**

# spec-compatible Rust stack

Thanks to Lindsey Tulloch, Victor Graf, Ian Goldberg!



# spec-compatible Rust stack

Thanks to Lindsey Tulloch, Victor Graf, Ian Goldberg!

your cool application could be here

Application Layer

Credential definition

Protocols

CMZ

$\mu$ CMZ

BBDT

cmz

Credential Layer

disjunction invariant

transformations

sigma-compiler

composition: and, or, threshold

Maurer proofs

compressed  $\Sigma$  protocols

sigma-proofs

Proof Layer

Fiat-Shamir transformation

spongefish

# spec-compatible Rust stack

Thanks to Lindsey Tulloch, Victor Graf, Ian Goldberg!

your cool application could be here

Application Layer

Credential definition

Protocols

CMZ

$\mu$ CMZ

BBDT

cmz

Credential Layer

disjunction invariant

transformations

sigma-compiler

composition: and, or, threshold

Maurer proofs

compressed  $\Sigma$  protocols

sigma-proofs

Proof Layer

Fiat-Shamir transformation

spongefish

what about other specs?

# $\Sigma$ -Protocols and Fiat-Shamir

*Michele Orrù (CNRS), Cathie Yun (Apple)*

`draft-irtf-cfrg-sigma-protocols`  
`draft-irtf-cfrg-fiat-shamir`

