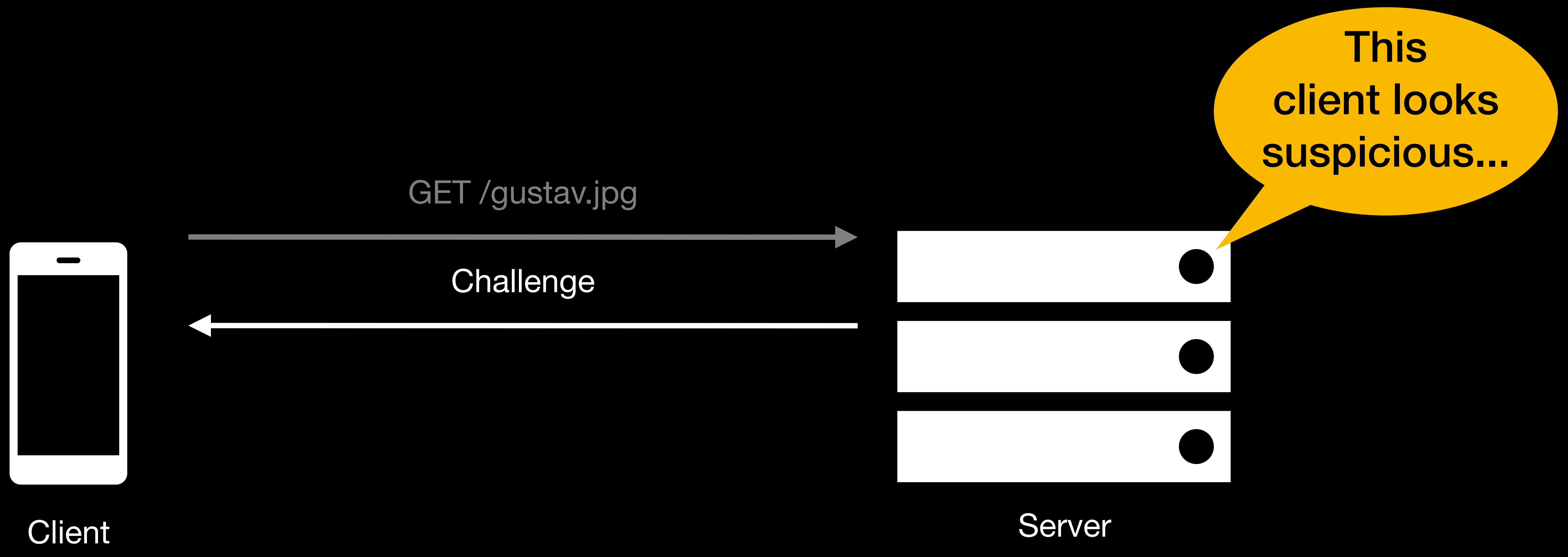
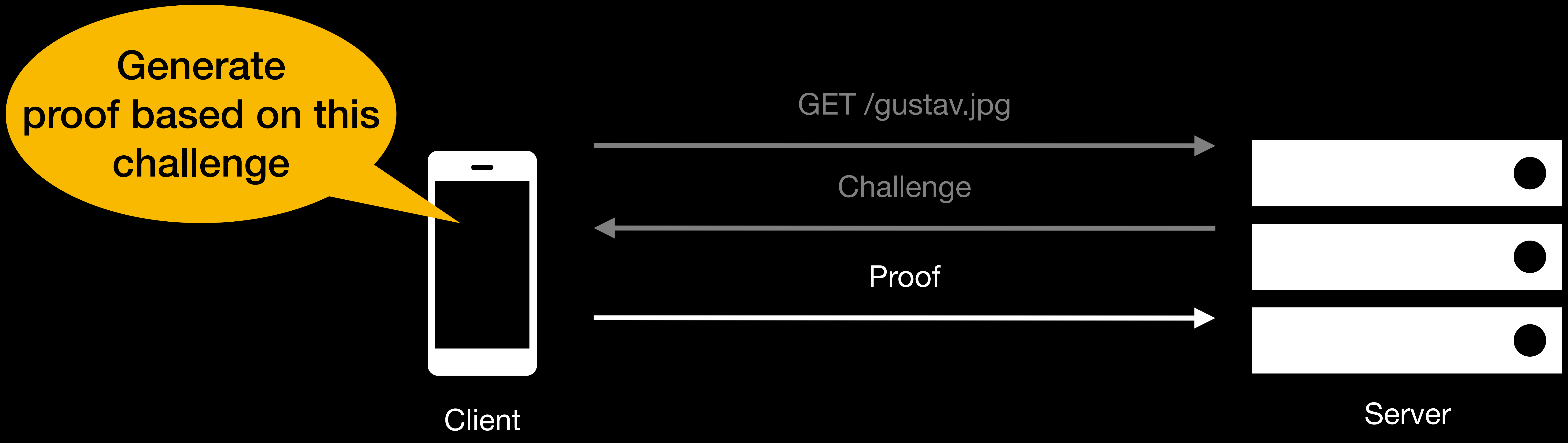
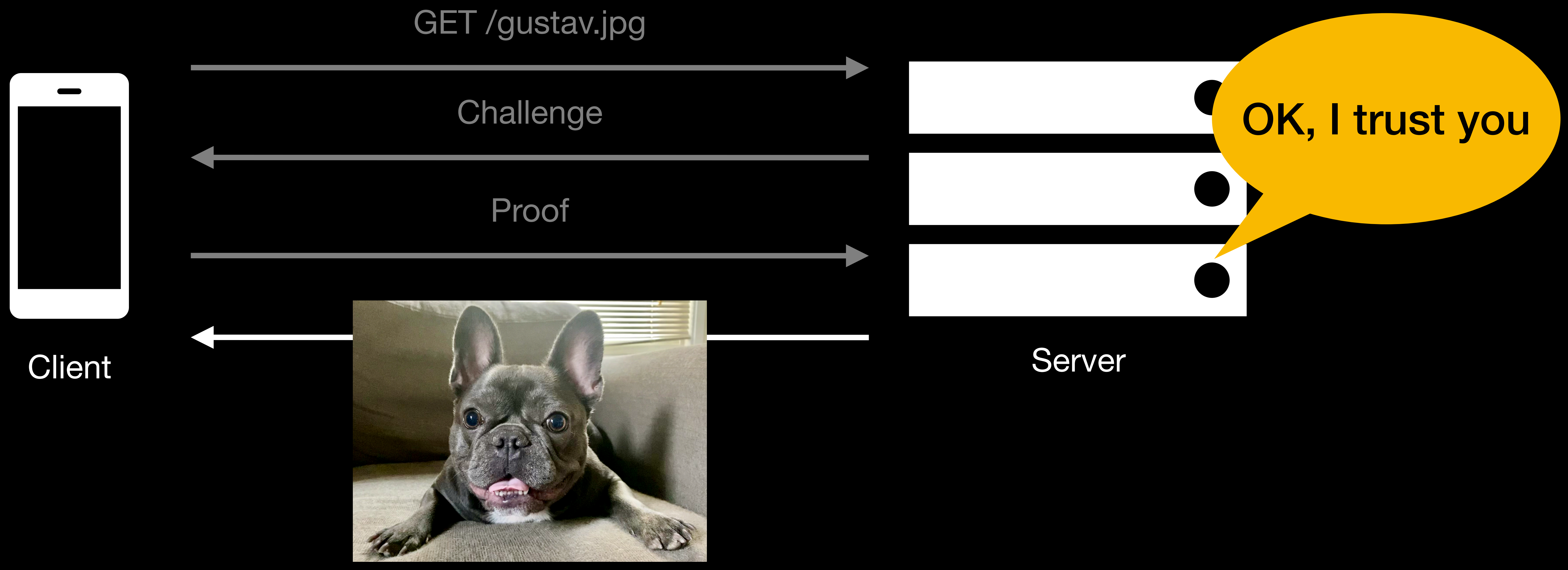
Architecture

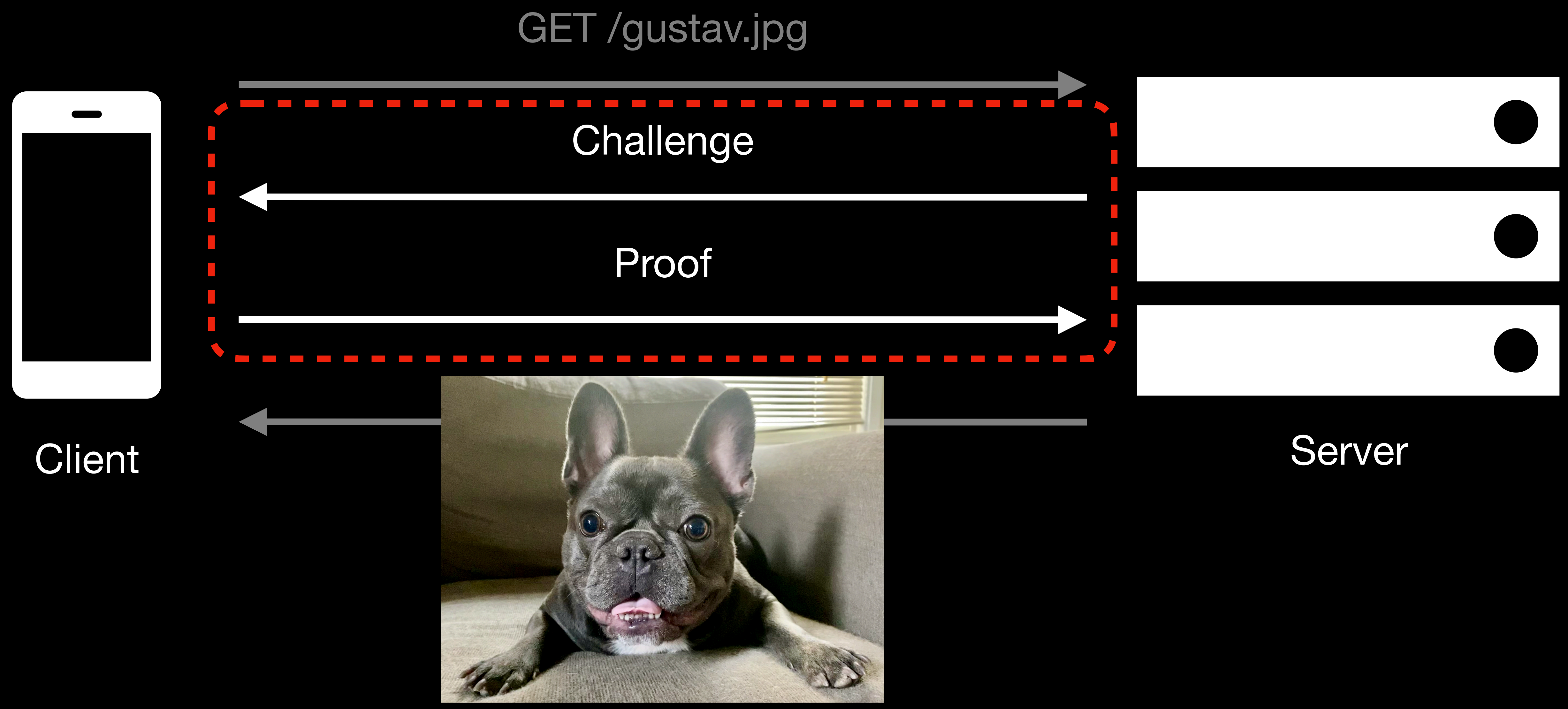
Chris Wood & Jana Iyengar

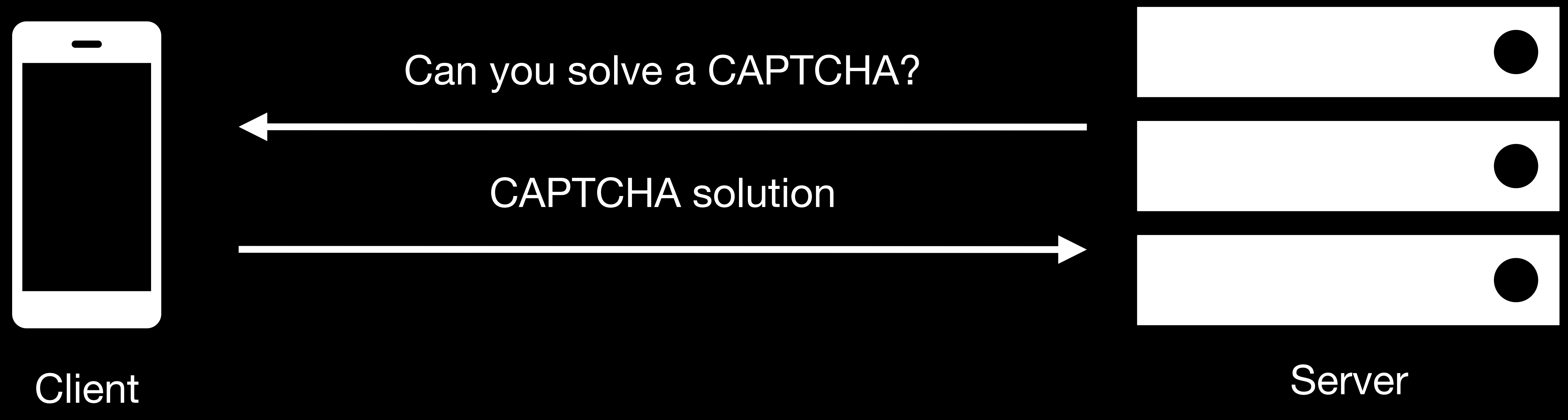

GET /gustav.jpg
Client
Server

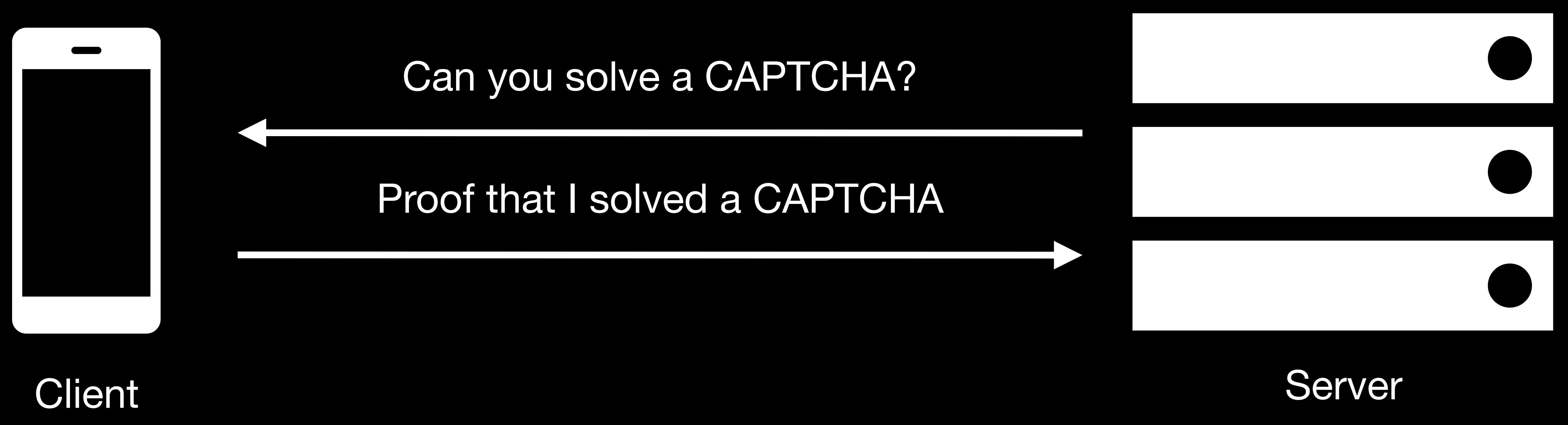

This client looks suspicious...
GET /gustav.jpg
Challenge
Client
Server

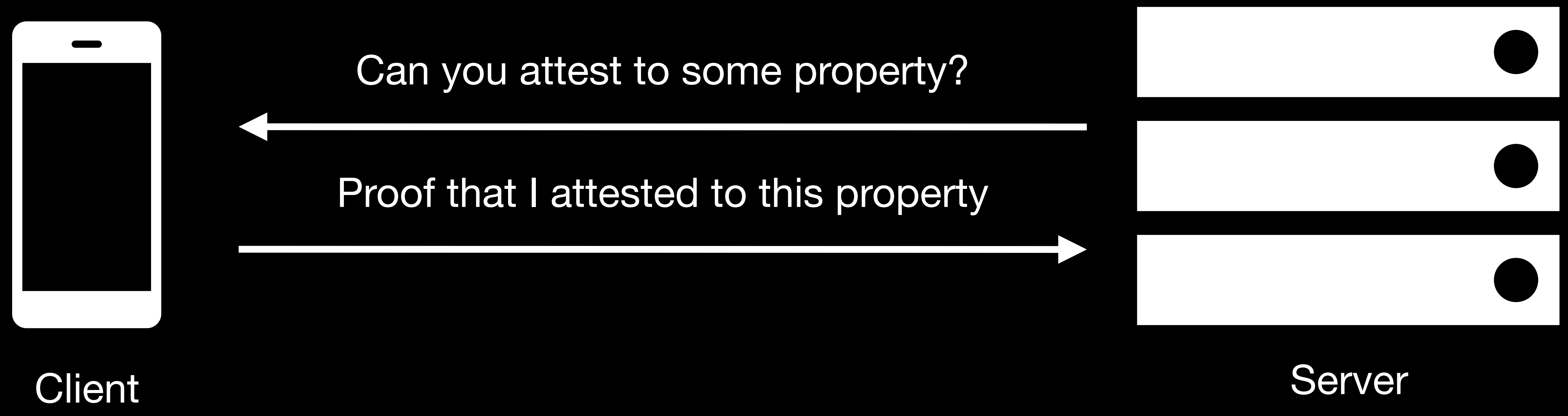

Generate proof based on this challenge
GET /gustav.jpg
Challenge
Proof
Client
Server

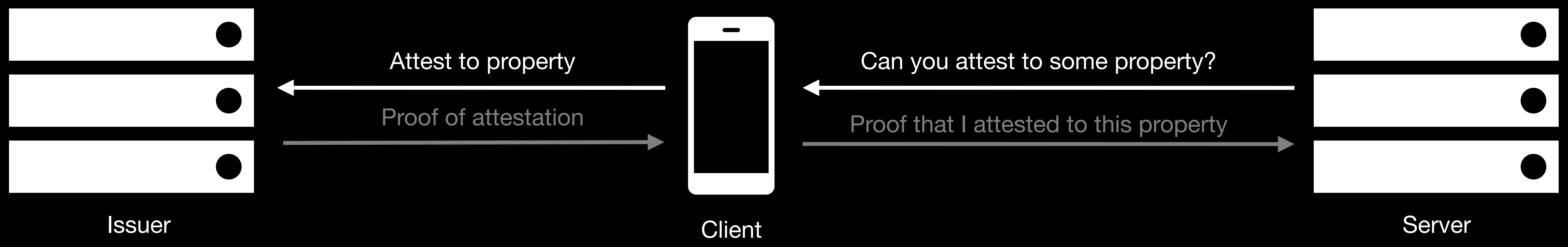

GET /gustav.jpg
Challenge
Proof
OK, I trust you
Client
Server

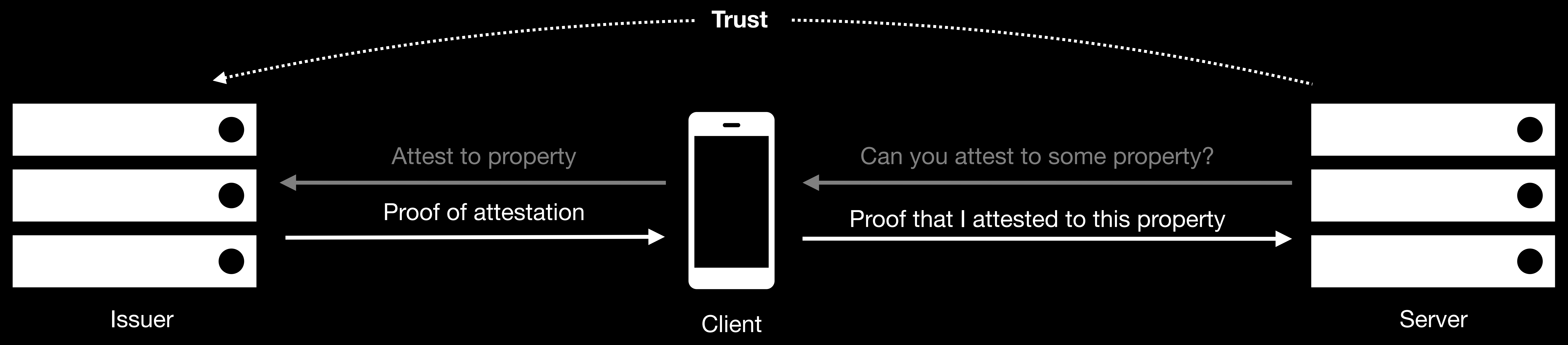

GET /gustav.jpg
Challenge
Proof
Client
Server

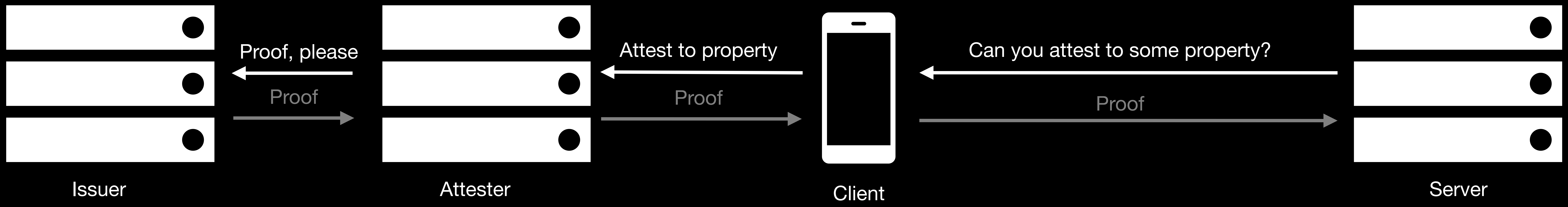

Can you solve a CAPTCHA?
CAPTCHA solution
Client
Server

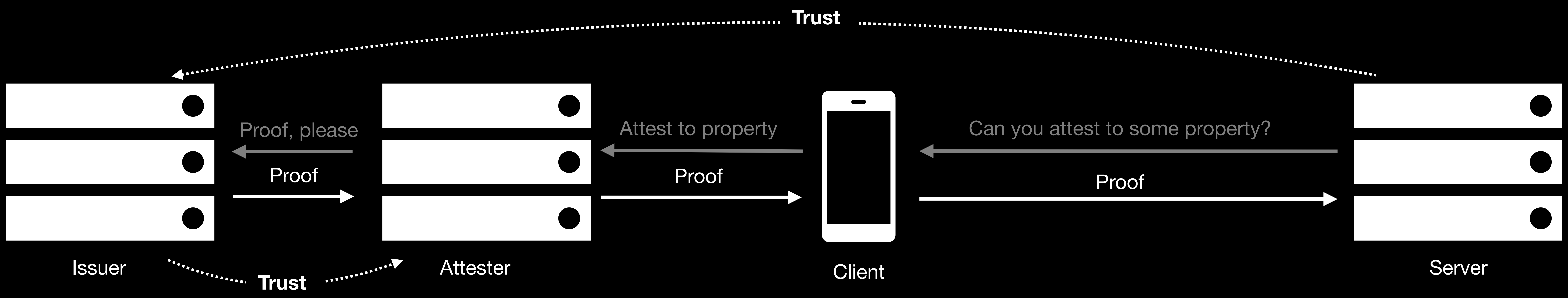

Can you solve a CAPTCHA?
Proof that I solved a CAPTCHA
Client
Server


Can you attest to some property?
Proof that I attested to this property
Client
Server


Attest to property
Proof of attestation
Can you attest to some property?
Proof that I attested to this property
Issuer
Client
Server


Trust
Attest to property
Proof of attestation
Can you attest to some property?
Proof that I attested to this property
Issuer
Client
Server


Proof, please
Proof
Attest to property
Proof
Can you attest to some property?
Proof
Issuer
Attester
Client
Server

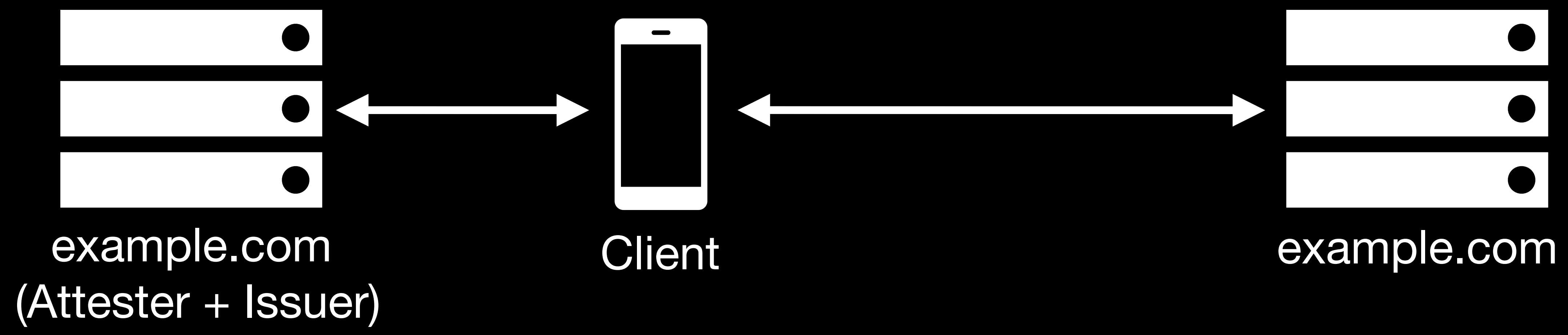

Trust
Proof, please
Proof
Attest to property
Proof
Can you attest to some property?
Proof
Issuer
Trust
Attester
Client
Server

Privacy Pass Architecture

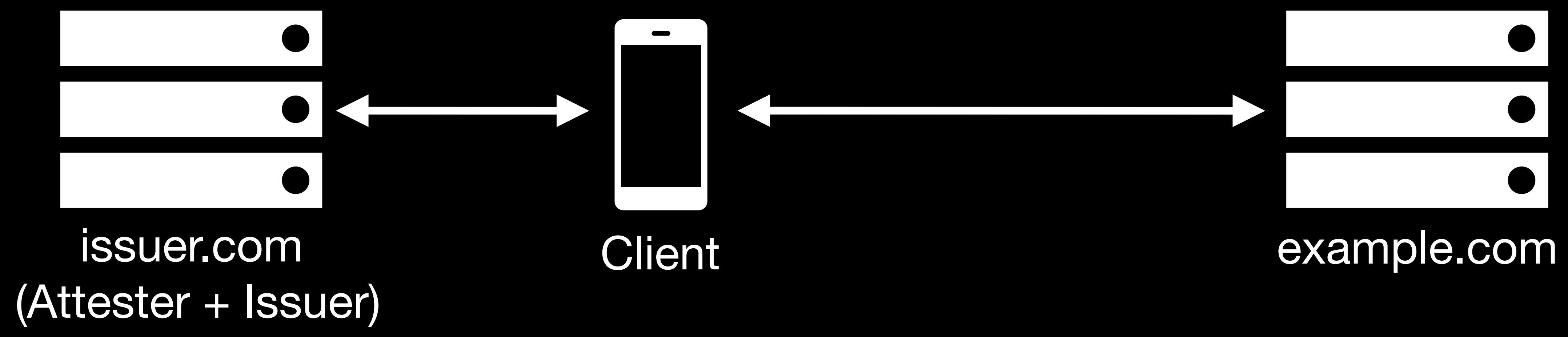
Deployment Variations

The architecture can be instantiated in various ways

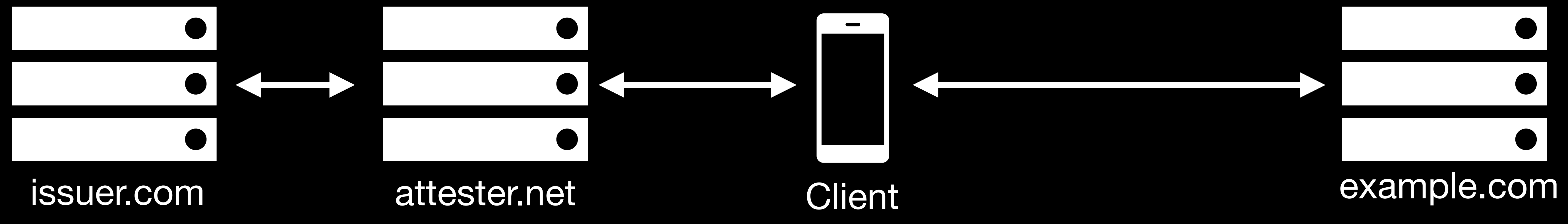
Combined origin/attester/issuer ("single verifier")



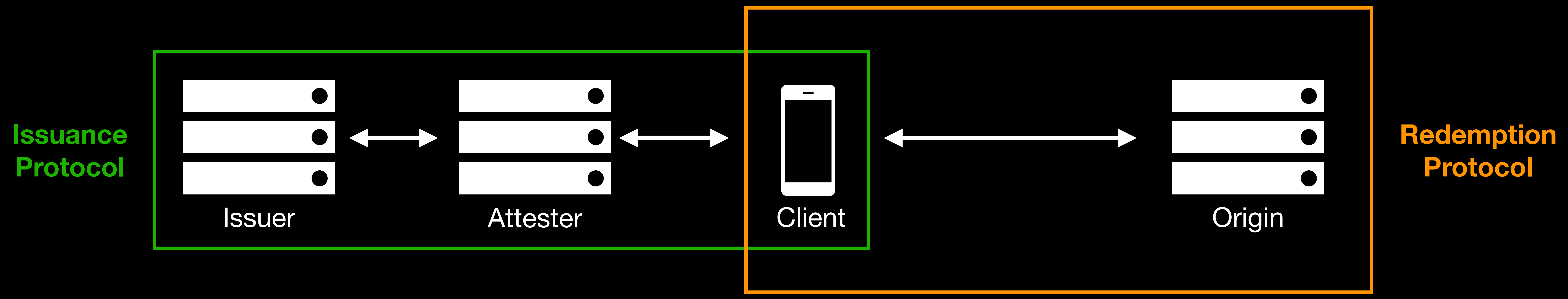
Combined attester/issuer



All separate



Protocol Structure



Architecture describes two parts of the protocol, which are detailed in two separate documents:

Redemption is a consistent/unified API for redeeming tokens, along with the ability to challenge.

Issuance can support multiple types (VOPRF, publicly verifiable, etc). This is the exchange that can be extended or replaced for new deployment models.

Big Picture
Architecture

Some function *attests* to certain state or properties associated with a client

- Has this person solved a CAPTCHA?
- Does this person have a subscriber account?

Issuers that trust attesters produce proof -- tokens -- bound to these properties

Redeemers, or **origins**, consume tokens from trusted issuers

Why rework the architecture?

Current architecture tightly couples *issuance* and *redemption*

- Issuer and redeemer may be the same (as in Privacy Pass) but don't need to be
- Separate roles allow for new deployment models and are more compatible with features like public verifiability

New architecture separates these functions and *shifts extensibility to issuance*

- New extensions or features can be solved by new issuance protocols
- Redemption is unchanged

Makes *attestation* explicit, but deployment specific

Proposal

Define architecture in terms of functional roles (Client, Origin, Attester, Issuer)

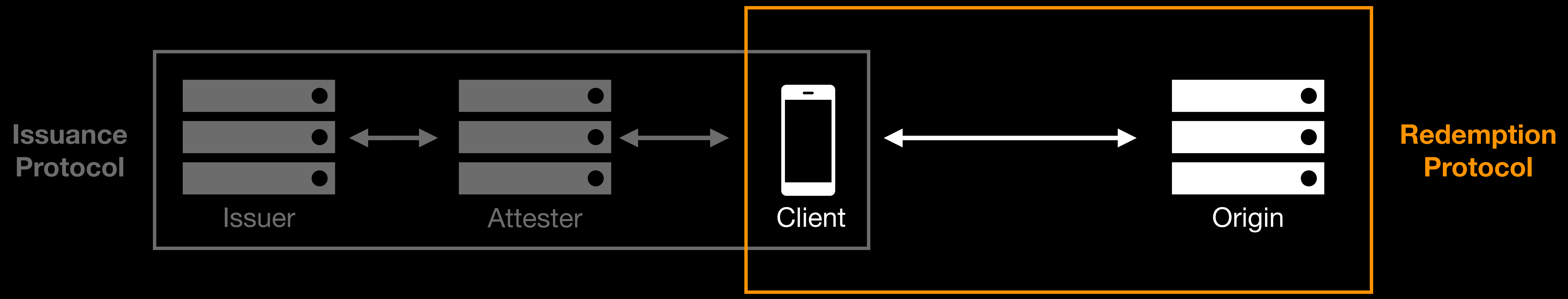
Define protocols in terms of Redemption and Issuance

Merge PR into architecture document

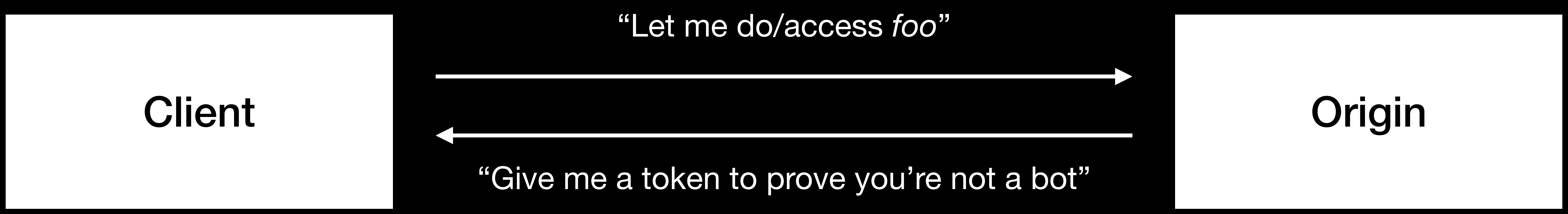
- https://github.com/ietf-wg-privacypass/base-drafts/pull/86

Challenge & Redemption

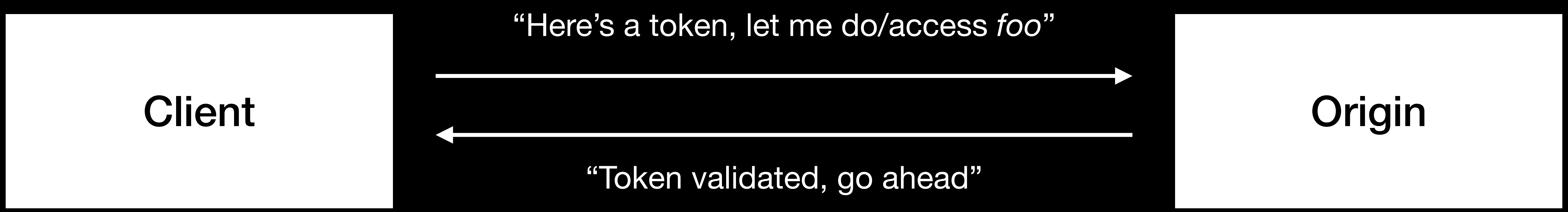
Tommy Pauly


Issuance
Protocol
Issuer
Attester
Client
Origin
Redemption
Protocol

Challenge



Redemption



Challenge & Redemption

All token schemes involve token redemption

- Token redemption is when a client presents a token to gain access, anonymously

Challenges are optional

- Allows a server to indicate that it needs tokens
- Indicates types of tokens and token issuers that are trusted
- Allows for interactive tokens

What was missing?

Previous design required Javascript APIs (W3C) work to functionally drive token interactions

No clear way to support new token types (POPRF vs publicly verifiable, etc)

HTTP authentication method allows a more standard definition

- Explicit support for different types of tokens, defined in their own contexts
- Works both in Javascript (W3C) and non-Javascript contexts

Authors proposing that this work replaces the HTTP API document

Features

Define an IANA registry of token types, indicate in challenge & redemption

Indicate Issuer name(s) (who does the Origin trust to vend tokens?)

Allow for “interactive tokens” with a one-time nonce to prevent farming

Allow for binding tokens to an origin to prevent cross-origin spending

Origin considerations

Make it easy for origins to adopt!

Origins don’t need to do complex crypto, just need to verify

- Publicly verifiable types are simple (RSA signatures)
- Privately verifiable requires Issuer key (or a single HTTP request to the issuer)

Interactive tokens mitigate concerns about farming and double-spending

- Shifts server state from *redeemed tokens* (unbounded) to *number of outstanding challenges* (bounded by active sessions)

Challenge

```
WWW-Authenticate: PrivateToken challenge=abc..., token-
key=123...

struct {
    uint16_t token_type; // Defines Issuance protocol
    opaque issuer_name<1..2^16-1>;
    opaque redemption_nonce<0..32>; // Optional
    opaque origin_name<0..2^16-1>; // Optional
} TokenChallenge;
```

Redemption nonce: If present, token presented must be *fresh* (interactively minted)

Origin name: If present, token is restricted to the origin, else it's *cross-origin*

Redemption

```
Authorization: PrivateToken token=abc...

struct {
    uint16_t token_type; // Matches challenge
    uint8_t nonce[32]; // Client-generated nonce
    uint8_t context[32]; // Hash of TokenChallenge
    uint8_t token_key_id[Nid];
    uint8_t authenticator[Nk]; // From Issuance protocol
} Token;
```

Context: SHA256 hash of the corresponding challenge

Authenticator: Signature, POPRF output, etc

Redemption Properties

Security properties

Redemption unlinkability: Redeemer cannot link two tokens to the same client ✓

Proposal

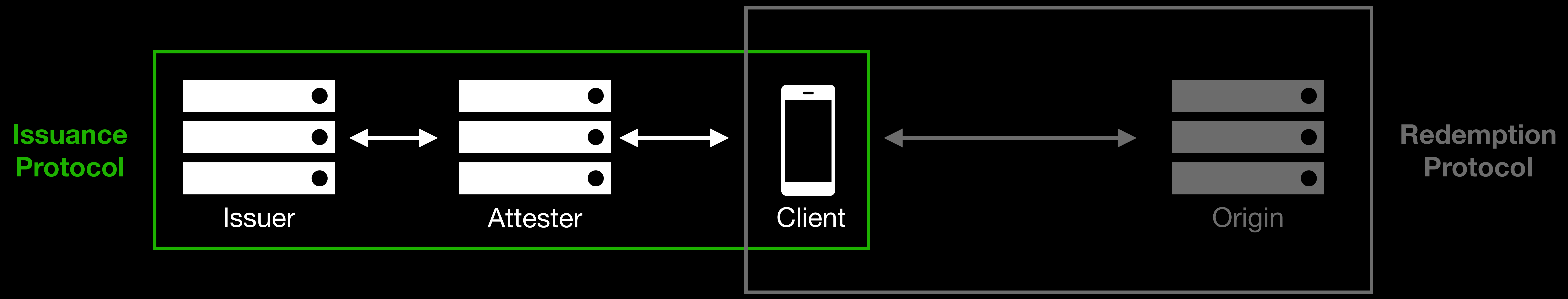
Replace HTTP API document with this HTTP auth scheme

- HTTP interactions with Issuers go to the Issuance Protocol document

Update W3C APIs to drive this HTTP API

Issuance

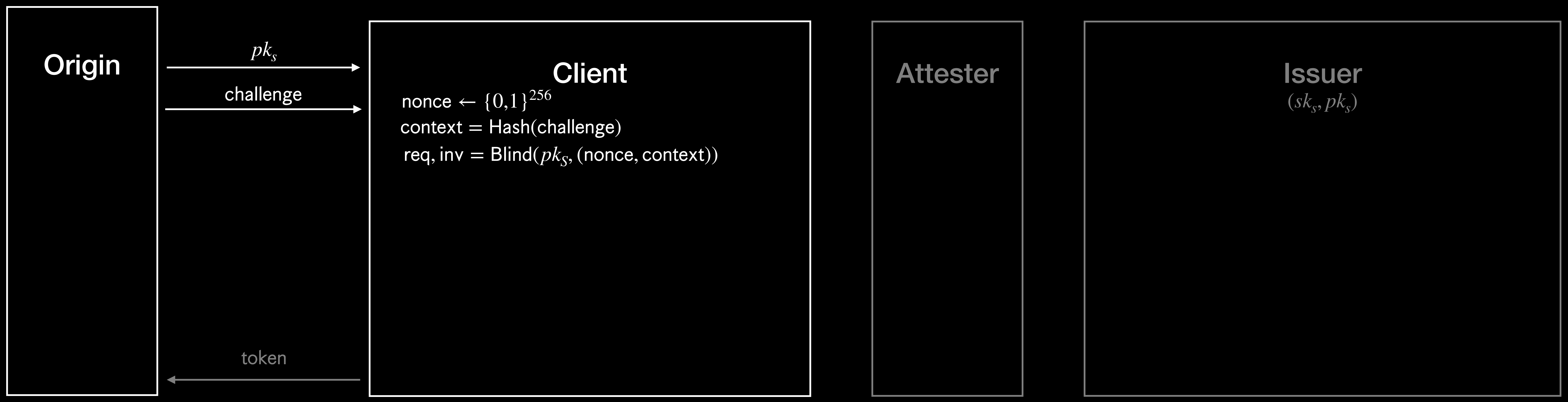
Chris Wood


Issuance
Protocol
Issuer
Attester
Client
Origin
Redemption
Protocol

Basic Tokens
Issuance protocols

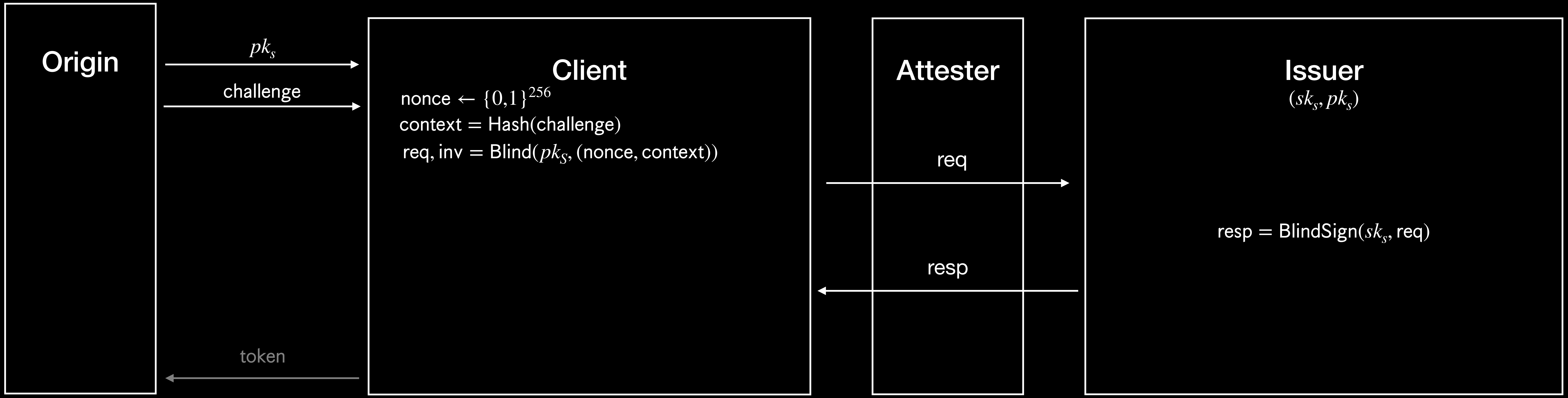
Basic Tokens

Issuance protocols



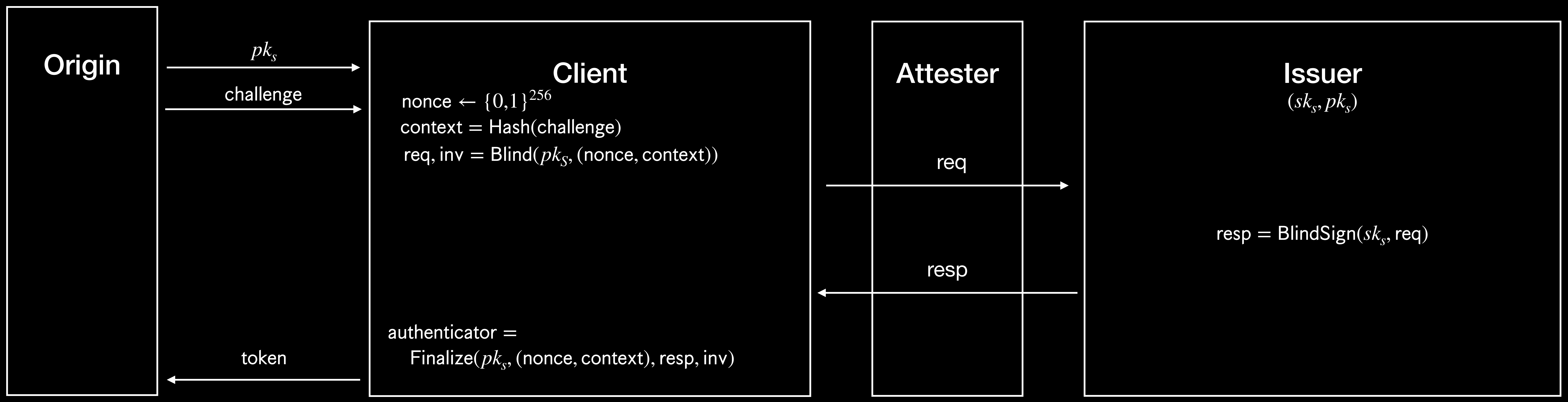
Basic Tokens

Issuance protocols



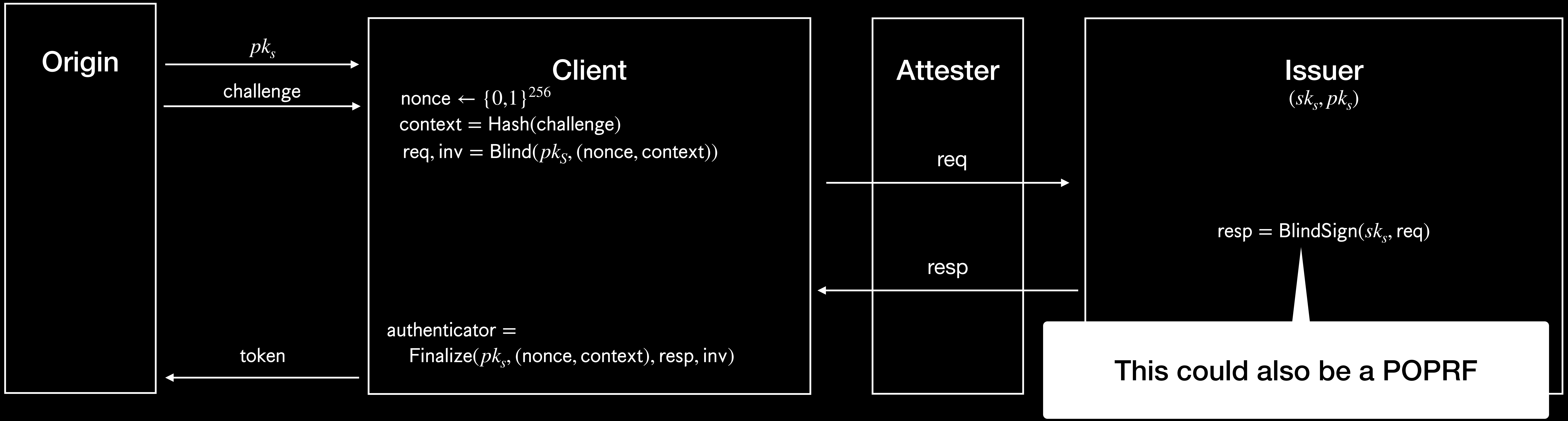
Basic Tokens

Issuance protocols



Basic Tokens

Issuance protocols



Issuance Registry

Issuance protocols

Value	Name	Publicly Verifiable	Public Metadata	Private Metadata	Authenticator Size
0x0001	POPRF(P-384, SHA-384)	N	Y	N	48
0x0002	Blind RSA (4096)	Y	N	N	512
...	...	...	...	...	...

Extensions for consideration:

Anonymous tokens with private metadata bit (https://eprint.iacr.org/2020/072)

Publicly verifiable anonymous tokens with private metadata bit (https://eprint.iacr.org/2022/004)

Blind BLS (https://datatracker.ietf.org/doc/draft-irtf-cfrg-bls-signature/)

Issuance Considerations

Issuance protocols

Issuance protocol is assumed to be stateless on the *Issuer*

- Blind signature protocols that require multiple rounds and state are possible, but not specified

Compatible with deployment specific key consistency mechanisms

- Issuer keys are discoverable such that applications can build consistency systems on top

Issuance Properties

Security properties

One-more unforgeability: Clients cannot forge tokens ✓

Issuance secrecy: Issuing parties cannot link per-client and per-origin state ✓

Proposal

Replace existing protocol document with new issuance protocol details

- Integrates with HTTP-based redemption protocol
- Satisfies private and public verifiability (per the charter)
- Makes issuance flow in the protocol document explicit and interoperable

Questions for the WG

Wrapping up

1. Are the document proposals clear?
2. Is there consensus in this new direction, which includes:
    1. Updates to *draft-ietf-privacypass-architecture* and *draft-ietf-privacypass-protocol*
    2. Adoption of *draft-pauly-privacypass-auth-scheme*

Rate-Limited Issuance

Chris Wood

Rate-Limited Tokens
Issuance protocols

Rate-limited tokens *extend* the basic issuance protocol with new properties:

1. Issuers learn origin associated with a token challenge

2. Attesters learn *stable mapping* between per-client secret and per-origin secret, and no per-origin information

3. Token requests may fail if the per-origin rate limit is exceeded

Challenge: How to reveal only the origin to issuer, and only the mapping to attester?

Rate-Limited Tokens

Issuance protocols

Rate-limited tokens *extend* the basic issuance protocol with new properties:

1. Issuers learn origin associated with a token challenge
2. Attesters learn ***stable mapping*** between per-client secret and per-origin secret, and no per-origin information
3. Token requests may fail if the per-origin rate limit is exceeded

Challenge: How to reveal only the origin to issuer, and only the mapping to attester?

Detour: Stable Mappings

Issuance protocols

A stable mapping is a deterministic function between per-client and per-origin information, e.g., F(client secret, origin secret)

The mapping is used to enforce per-origin limits

Attester uses mapping as index into data structure tracking per-client state

Mapping	Count
….	**…**
12311235123	N
….	…

Detour: Stable Mappings

Issuance protocols

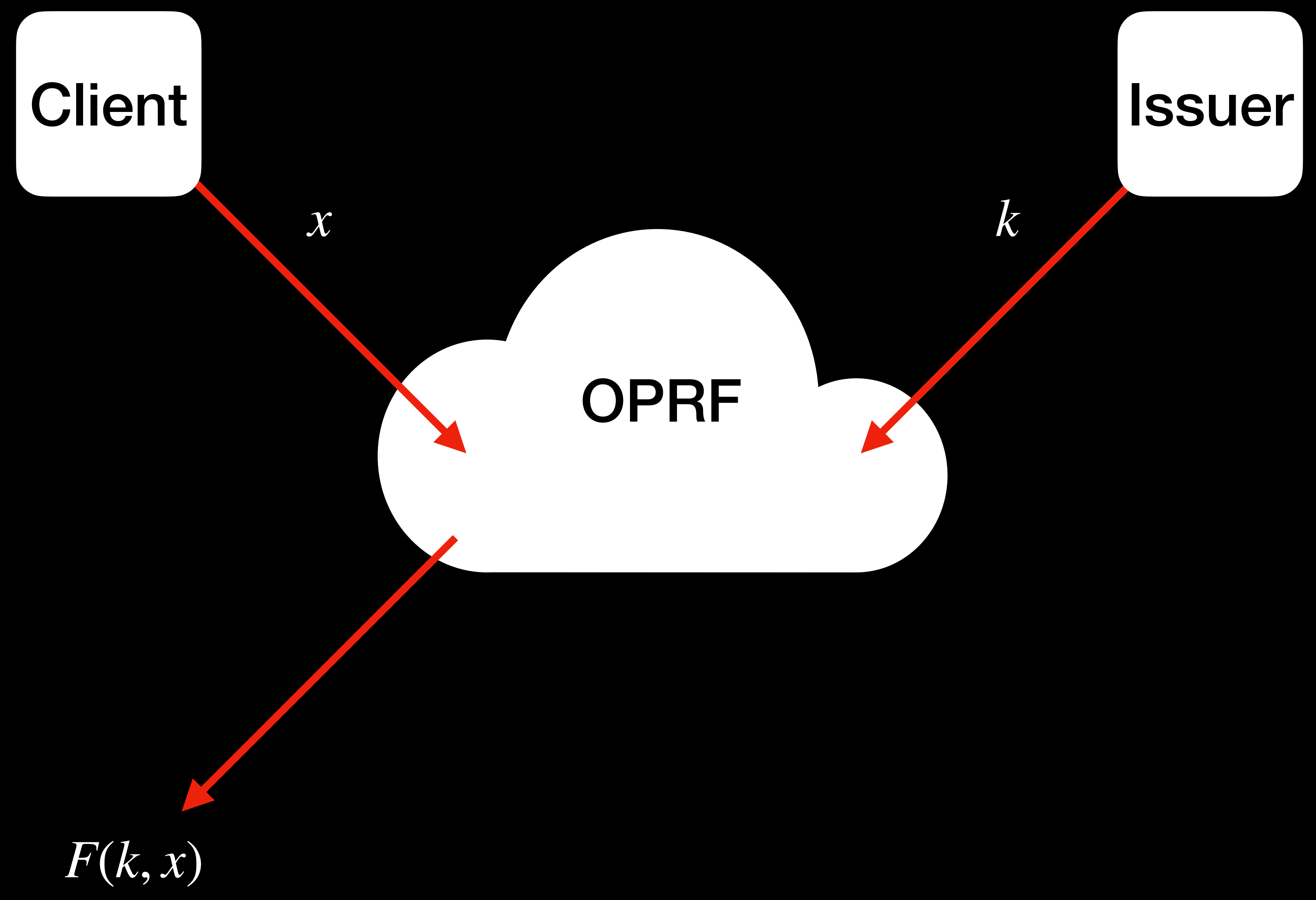
Detour: Stable Mappings

Issuance protocols

Detour: An OPRF Sketch

Issuance protocols

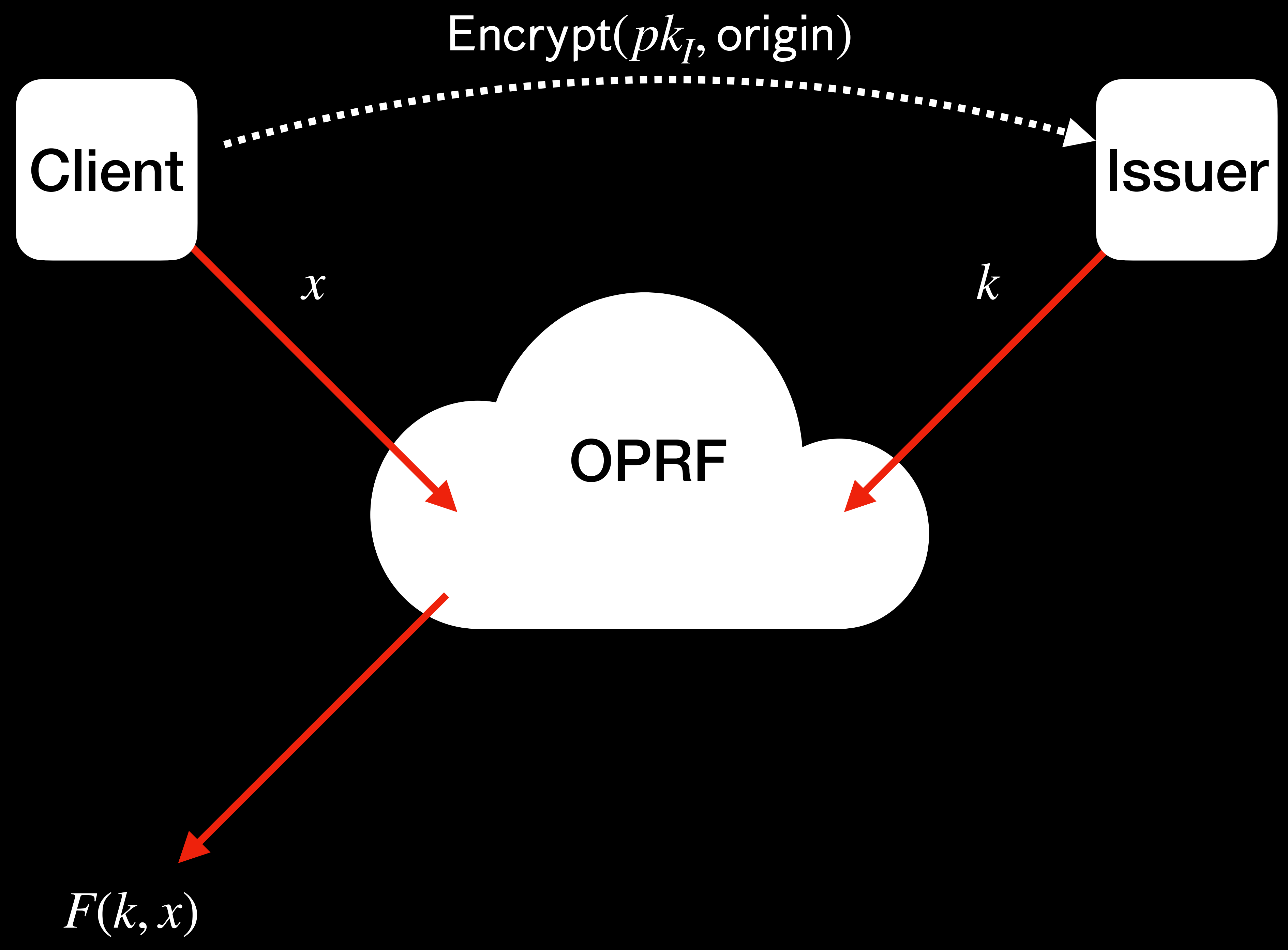
An OPRF protocol computes $F(k, x)$ for per-origin k and per-client x



Detour: An OPRF Sketch

Issuance protocols

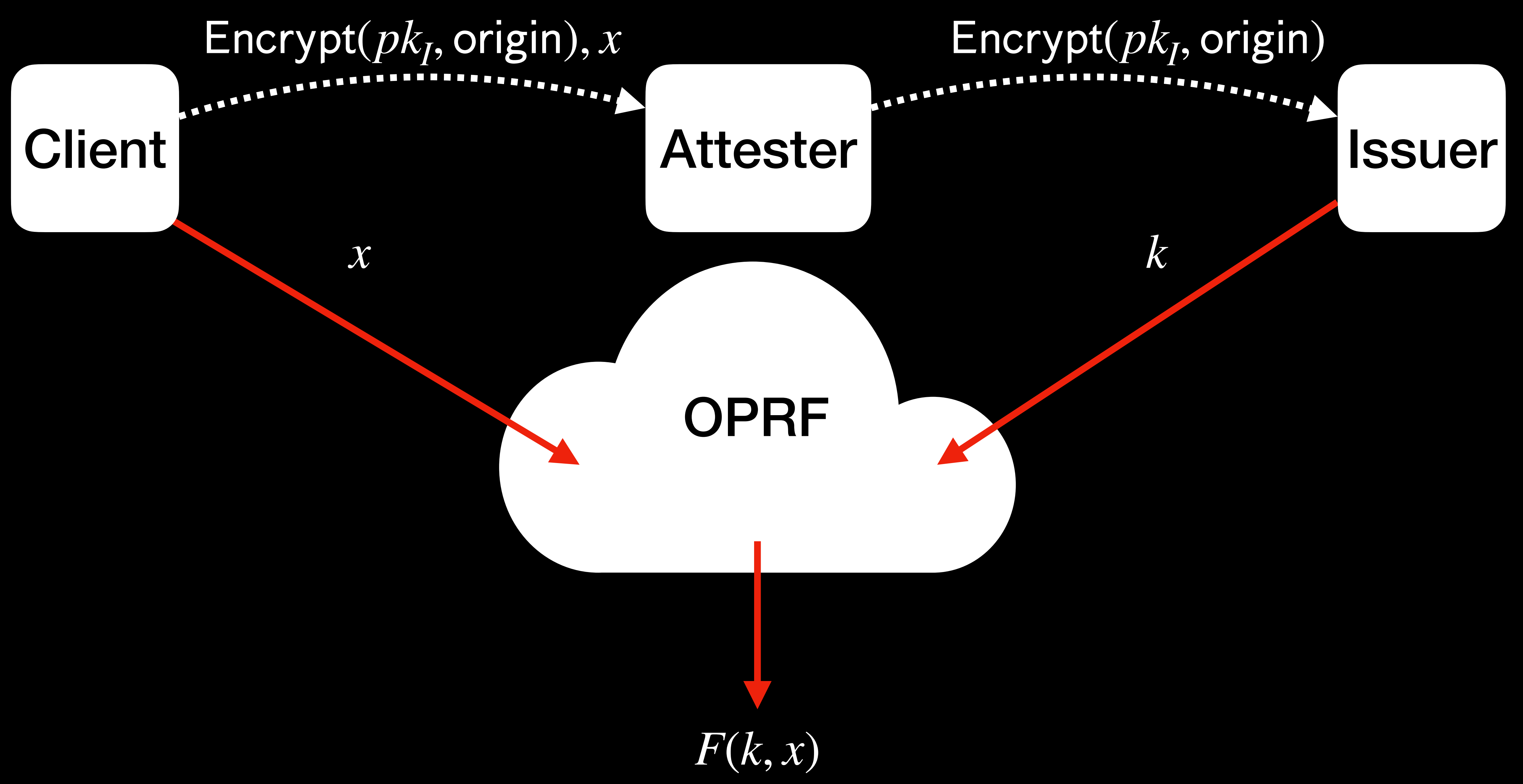
Clients can encrypt the origin identifier under the Issuer's public key



Detour: An OPRF Sketch

Issuance protocols

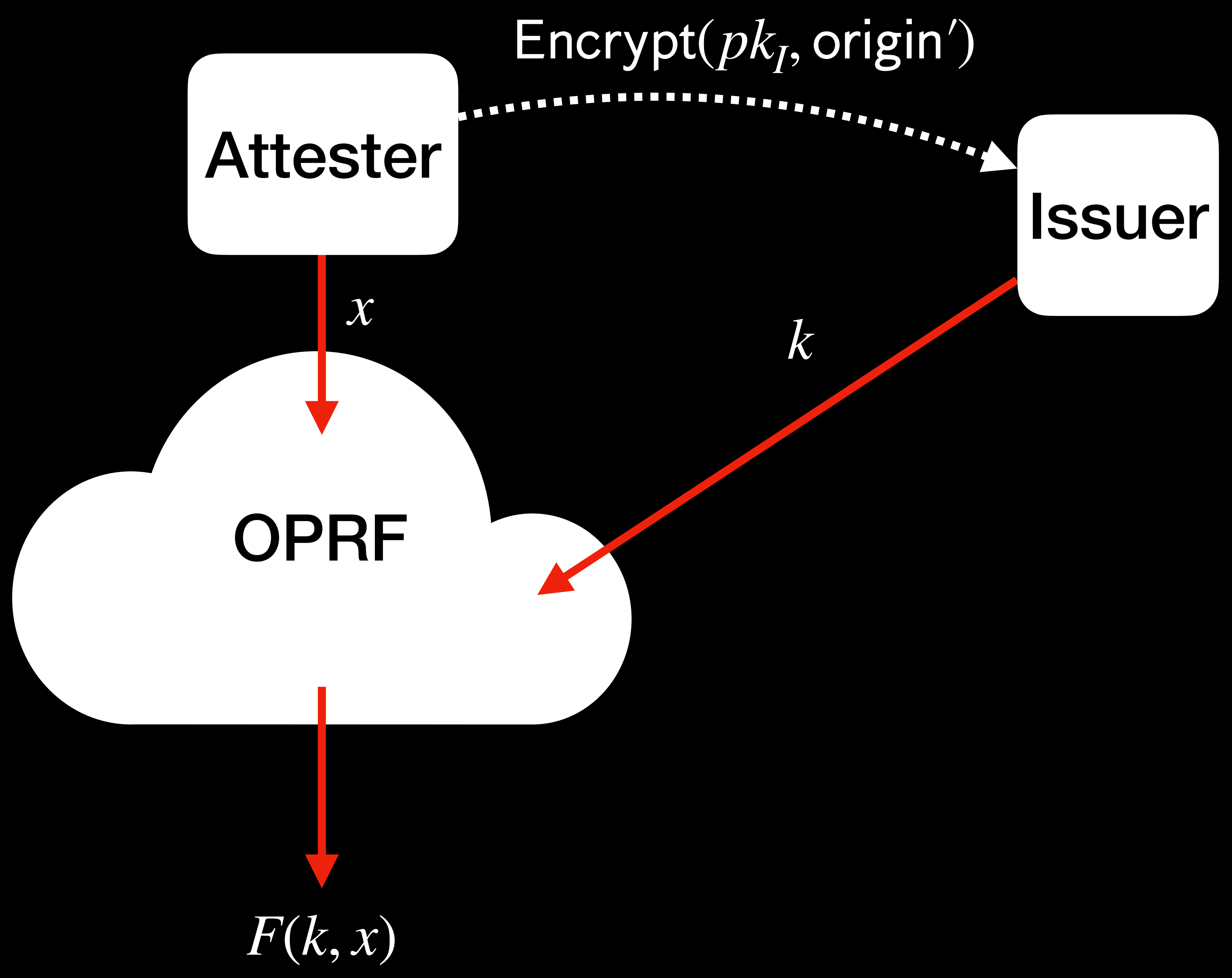
An Attester can relay the encrypted origin name and complete the OPRF



Detour: An OPRF Sketch

Issuance protocols

… Attester can perform a dictionary attack to learn $F(k, x)$



Rate-Limited Tokens

Issuance protocols

Rate-limited tokens *extend* the basic issuance protocol with new properties:

1. Issuers learn origin associated with a token challenge
2. Attesters learn *stable mapping* between per-client secret and per-origin secret, and no per-origin information
3. Token requests may fail if the per-origin rate limit is exceeded

Challenge 1: How to reveal only the origin to issuer, and only the mapping to attester?

Challenge 2: How to ensure the attester cannot dictionary attack or replay client requests to learn per-origin information?

Proposed solution to both uses the same mechanism!

Rate-Limited Tokens

Issuance protocols

Cryptographic primitives:

- Blind RSA: Token request
- HPKE: Encrypting origin names from Client to Issuer
- EdDSA with key blinding: Signing Client requests *and* computing stable mappings

This is the interesting piece!

Detour: EdDSA with Key Blinding

Issuance protocols

Extend RFC8032 EdDSA with two functionalities

BlindPublicKey and UnblindPublicKey: Given public key and secret blind, produce *blinded public key*

UnblindPublicKey(BlindPublicKey(pkS, skB), skB) = pkS

BlindKeySign: Sign message with secret key and secret blind

Verify(BlindPublicKey(pkS), msg, BlindKeySign(skS, skB, msg)) = true

Draft specification: https://chris-wood.github.io/draft-wood-cfrg-eddsa-blinding/draft-wood-cfrg-eddsa-blinding.html

Rate-Limited Tokens

Issuance protocols