Blind RSA Signatures

`draft-wood-cfrg-blind-rsa`

Denis, Jacobs, Wood - IETF 110 - CFRG

Motivation

Background

A *verifiable oblivious pseudorandom function* (VOPRF) is a multi-party protocol that computes

$$y = F(k, x)$$

with server secret key k and client input x such that:

1. Server learns nothing of x

2. Client learns only output y

Motivation

Applications

A growing number of applications require VOPRF-like constructions

- Privacy Pass
- Tor DoS defenses
- Ad-click fraud prevention

… but VOPRFs raise operational challenges

- Widely shared secrets
- Key server (HSM) load

Blind Signatures

Overview

Blind signatures are multi-party protocols similar to VOPRFs, with one important distinction: signatures are *publicly verifiable*

Many constructions exist

- Blind Schnorr Signatures
- Blind BLS
- Abe Blind Signatures
- Chaum Blind RSA Signatures
- … others

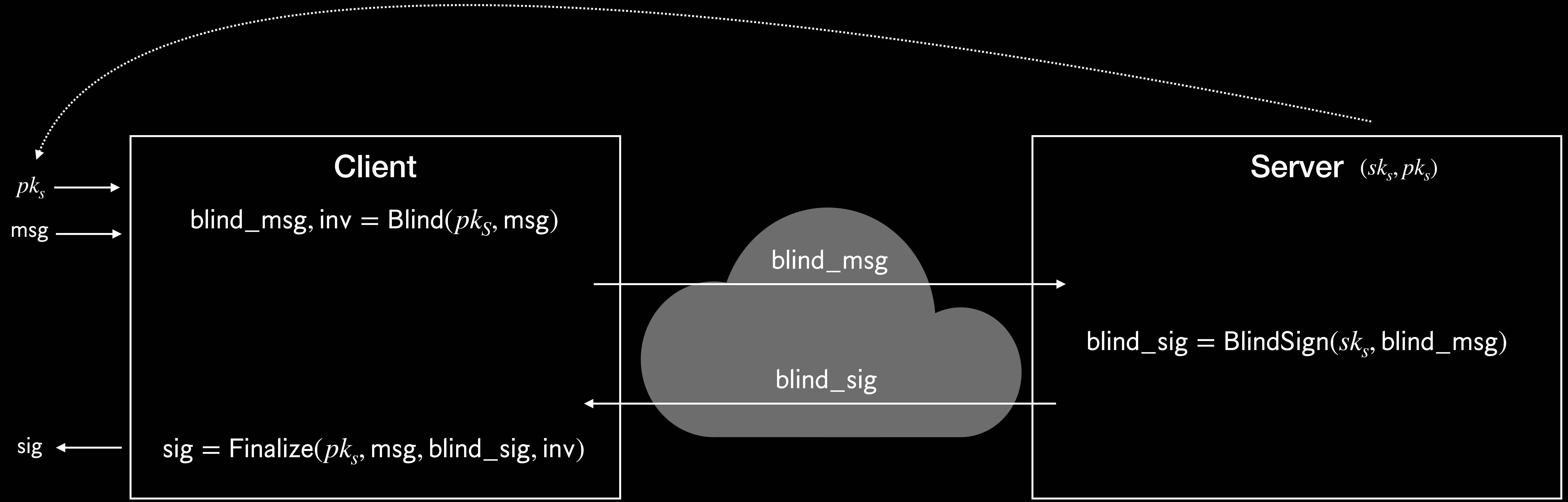
Blind Signatures

Tradeoffs and considerations

Scheme	Pros	Cons
Blind Schnorr Signatures	Lightweight Threshold-friendly (c.f. FROST)	Three messages (state or computation overhead) Polynomial-time ROS attack (2020/945), but FPS20 seems plausible
Blind BLS	Lightweight	Expensive signing and verification Pairing support is not (yet!) widely supported in common libraries (BoringSSL, ring, etc)
Abe	Polynomial concurrent security Seems unaffected by ROS attack (2020/945)	Three messages (state or computation overhead) Large signature sizes (several group elements)
Chaum Blind RSA Signatures	One round issuance (stateless issuance server) Verification widely supported in libraries*	Large signature sizes (256-512B) Difficult to support threshold operations “Legacy”

Blind RSA

Protocol



pk_s
msg
sig
Client
blind_msg, inv = Blind(pk_S, msg)
sig = Finalize(pk_s, msg, blind_sig, inv)
blind_msg
blind_sig
Server (sk_s, pk_s)
blind_sig = BlindSign(sk_s, blind_msg)

Blind RSA

Encoding function

Client “Blind” routine hashes and encodes the message before blinding it

Several encoding options exist:

Scheme	Secure?	Determinstic signatures?	Randomized signatures?	Widely supported?	Recommended?
PSS	✓	✓	✓	✓	✓
FDH	✓	✓	✗	✗	?
PKCS#1 v1.5	✓	✗	✓	✓	✗*

This draft chose **PSS** to maximize code reuse, align with current recommended algorithms, and support deterministic and randomized signatures… but this can change!

Current Status

Running code and wider use

Current status:

- Several interoperable implementations with test vectors available
- Solves Privacy Pass charter item to support public verifiability

> "… The Working Group will specify a preliminary set of extensions, including Issuer-supplied metadata and cryptographic instantiations that additionally support ***public verifiability*** of Issued tokens, …"

1) Interest in working on blind signatures?

2) Interest in adopting this document?