Blind RSA Signatures
draft-wood-cfrg-blind-rsa
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

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

**Client**

\[ \text{blind}_\text{msg, inv} = \text{Blind}(pk_S, \text{msg}) \]

\[ \text{sig} = \text{Finalize}(pk_S, \text{msg}, \text{blind}_\text{sig}, \text{inv}) \]

**Server** \((sk_s, pk_s)\)

\[ \text{blind}_\text{sig} = \text{BlindSign}(sk_s, \text{blind}_\text{msg}) \]
Blind RSA

Encoding function

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

Several encoding options exist:

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Secure?</th>
<th>Deterministic signatures?</th>
<th>Randomized signatures?</th>
<th>Widely supported?</th>
<th>Recommended?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FDH</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>?</td>
</tr>
<tr>
<td>PKCS#1 v1.5</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X*</td>
</tr>
</tbody>
</table>

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?