Partially Blind RSA Signatures

draft-amjad-cfrg-partially-blind-rsa

Ghous Amjad*, Scott Hendrickson, Christopher Wood, Kevin Yeo
IETF 116 - CFRG
Outline

• Motivation

• Background: Blind RSA Signatures

• Partially Blind RSA Signatures

• Benchmarks

• Current Status
Motivation: Blind Signatures

- Privacy Pass
- Web Browsing, e.g.,
  - VPN by Google One
  - iCloud Private Relay
- Avoiding Repeated CAPTCHA Solving
- Private Click Measurement
- Tor DOS Defenses
Motivation: Partially Blind Signatures

- ‘draft-irtf-cfrg-voprf’ offers partially oblivious variant
- Signatures that can only be used for
  - specific settings
  - specific geographic location etc.
- Avoiding one key per metadata approach
  - May require fixed public metadata choices ahead of time
  - Key management scalability concerns
Motivation: Blind RSA Signatures

- IETF document adopted for Blind RSA Signatures
  - Simple (one-round scheme, stateless server issuance)
  - Widely supported public verification
  - ‘draft-irtf-cfrg-rsa-blind-signatures’

- Natural to think of Public Metadata support for this standard
Outline

• Motivation
• Background: Blind RSA Signatures
• Partially Blind RSA Signatures
• Benchmarks
• Current Status
Background: Blind RSA Signatures

Client

msg = “new_attendee”

Pk

Server

Key = Pk, Sk
Background: Blind RSA Signatures

Blind_sig = \text{BlindSign}(Sk, \text{Blinded}_msg)

\text{Blinded}_msg, inv = \text{Blind}(Pk, msg)

Client \rightarrow \text{Server}

msg, Pk \rightarrow \text{Blinded}_msg, inv \rightarrow \text{Blind}(Pk, msg) \rightarrow \text{BlindSig}(Sk, \text{Blinded}_msg) \rightarrow \text{Blind_sig}
Background: Blind RSA Signatures

\[
\text{Signature} = \text{Finalize} (P_k, \text{msg}, \text{inv}, \text{Blind\_sig})
\]
Background: Blind RSA Signatures

• Signature is verified as a sub-routine in Finalize.
• Signature is publicly verifiable.

• Input message is encoded before being blinded.
  • PSS Encoding
Outline

- Motivation
- Background: Blind RSA Signatures
- Partially Blind RSA Signatures
- Benchmarks
- Current Status
Partially Blind RSA Signatures

• Same public metadata (md) needed in all stages of the protocol
  • Blinding
  • Signing
  • Finalizing
  • Verifying
Partially Blind RSA Signatures

- Augmented Input Message
  - Unique encoding of message and “md” passed to PSS encoding
- Augmented Public Key
  - $P_k \times H(md)$
  - using HKDF as $H$ for implementation ease
  - $H(md)$ needs to be co-prime to $\phi(N)$ where $N$ is the RSA modulus, to generate the correct private key
- Generating special RSA modulus
  - $N$ should be a product of two safe primes
Security Considerations

• One-more-unforgeability
• Unlinkability under same public metadata
• Domain separation
  • Different RSA moduli will ensure different augmented public keys for same public metadata
  • Hash functions in input message augmentation and public key augmentation are domain separated
• Denial of Service attacks due to larger public keys
Outline

- Motivation
- Background: Blind RSA Signatures
- Partially Blind RSA Signatures
- Benchmarks
- Current Status
## Benchmarks

<table>
<thead>
<tr>
<th></th>
<th>Blind RSA Signatures</th>
<th>Partially Blind RSA Signatures*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blind</td>
<td>459,169 ns</td>
<td>1,695,262 ns</td>
</tr>
<tr>
<td>BlindSign</td>
<td>1,298,156 ns</td>
<td>5,368,773 ns</td>
</tr>
<tr>
<td>Finalize</td>
<td>37,821 ns</td>
<td>1,262,426 ns</td>
</tr>
</tbody>
</table>

* Timing should improve with more optimized code (e.g. once CRT is used)  
* [https://github.com/google/anonymous-tokens](https://github.com/google/anonymous-tokens)  
Outline

• Motivation
• Background: Blind RSA Signatures
• Partially Blind RSA Signatures
• Benchmarks
• Current Status
Current Status

- Two implementations (C++, Go)
- Solves needs in Privacy Pass and related real world applications
- draft-amjad-cfrg-partially-blind-rsa
- Academic paper with security proofs to be put out soon.
- Interest in adopting this document?
Thank you!