

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 9 September 2021

F. Denis
Fastly Inc.
F. Jacobs
Apple Inc.
C.A. Wood
Cloudflare
8 March 2021

RSA Blind Signatures
draft-wood-cfrg-rsa-blind-signatures-00

Abstract

This document specifies the RSA-based blind signature scheme with appendix (RSA-BSSA). RSA blind signatures were first introduced by Chaum for untraceable payments [[Chaum83](#)]. It extends RSA-PSS encoding specified in [[RFC8017](#)] to enable blind signature support.

Discussion Venues

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

Source for this draft and an issue tracker can be found at <https://github.com/chris-wood/draft-wood-cfrg-blind-signatures>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 9 September 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

Internet-Draft

RSA Blind Signatures

March 2021

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Requirements Notation	3
3.	Notation	3
4.	Blind Signature Protocol Overview	4
5.	RSABSSA Signature Instantiation	4
5.1.	Signature Generation	4
5.1.1.	Blind	5
5.1.2.	BlindSign	5
5.1.3.	Finalize	6
5.2.	Encoding Options	7
6.	Public Key Certification	8
7.	Security Considerations	8
7.1.	Timing Side Channels	8
7.2.	Message Robustness	8
7.3.	Salt State	9
7.4.	Key Substitution Attacks	9
7.5.	Alternative RSA Encoding Functions	9
7.6.	Alternative Blind Signature Schemes	9
7.7.	Post-Quantum Readiness	10
8.	IANA Considerations	10
9.	References	10
9.1.	Normative References	11
9.2.	Informative References	11
Appendix A.	Test Vector	14
	Authors' Addresses	17

[1.](#) Introduction

Originally introduced in the context of digital cash systems by Chaum for untraceable payments [[Chaum83](#)], RSA blind signatures turned out to have a wide range of applications ranging from electric voting schemes to authentication mechanisms.

Recently, interest in blind signatures has grown to address operational shortcomings from VOPRFs such as [[I-D.irtf-cfrg-voprf](#)]. Specifically, VOPRF evaluation requires access to the private key, and is therefore required for both issuance and redemption of tokens

in anonymous authentication protocols such as Privacy Pass [[I-D.davidson-pp-protocol](#)]. This limitation complicates deployments where it is not desirable to distribute secret keys entities performing token verification. Additionally, if the private key is kept in a Hardware Security Module, the number of operations on the key are doubled compared to a scheme where the private key is only required for issuance of the tokens.

In contrast, cryptographic signatures provide a primitive that is publicly verifiable and does not require access to the private key for verification. Moreover, [[JKK14](#)] shows that one can realize a VOPRF in the Random Oracle Model by hashing a (deterministic) blind signature-message pair.

This document specifies the RSA Blind Signature Scheme with Appendix (RSABSSA). In order to facilitate deployment, we define it in such a way that the resulting (unblinded) signature can be verified with a standard RSA-PSS library.

[2.](#) Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#) [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

[3.](#) Notation

The following terms are used throughout this document to describe the protocol operations in this document:

- * I2OSP and OS2IP: Convert a byte string to and from a non-negative integer as described in [[RFC8017](#)]. Note that these functions operate on byte strings in big-endian byte order.

- * `random_integer_uniform(M, N)`: Generate a random, uniformly distributed integer R such that $M \leq R < N$.
- * `inverse_mod(x, n)`: Compute the multiplicative inverse of $x \bmod n$.
- * `len(s)`: The length of a byte string, in octets.

[4.](#) Blind Signature Protocol Overview

In this section, we sketch the blind signature protocol wherein a client and server interact to compute `"sig = Sign(skS, msg)"`, where `"msg"` is the private message to be signed, and `"skS"` is the server's private key. In this protocol, the server learns nothing of `"msg"`, whereas the client learns `"sig"` and nothing of `"skS"`.

The core issuance protocol runs as follows:

```

Client(pkS, msg)                                Server(skS, pkS)
-----
blinded_msg, inv = Blind(pkS, msg)

                                blinded_msg
                                ----->

                                blind_sig = BlindSign(skS, blinded_msg)

                                blind_sig
                                <-----

sig = Finalize(pkS, msg, blind_sig, inv)

```

Upon completion, correctness requires that clients can verify signature `"sig"` over private input message `"msg"` using the server public key `"pkS"` by invoking the RSASSA-PSS-VERIFY routine defined in [\[RFC3447\]](#). The finalization function performs that check before returning the signature.

5. RSABSSA Signature Instantiation

[Section 8.1 of \[RFC8017\]](#) defines RSASSA-PSS RSAE, which is a signature algorithm using RSASSA-PSS [\[RFC8017\]](#) with mask generation function 1. In this section, we define RSABSSA, a blinded variant of this algorithm.

5.1. Signature Generation

As outlined in [Section 4](#), signature generation involves three subroutines: Blind, BlindSign, and Finalize. The output from Finalize is a signature over the input to Blind. A specification of these subroutines is below.

5.1.1. Blind

rsabssa_blind encodes an input message and blinds it with the server's public key. It outputs the blinded message to be sent to the server and the corresponding inverse, both encoded as octet strings. RSAVP1 and EMSA-PSS-ENCODE are as defined in [\[RFC3447\]](#).

rsabssa_blind(pkS, msg)

Parameters:

- kLen, the length in octets of the RSA modulus n
- kBits, the length in bits of the RSA modulus n

Inputs:

- pkS, server public key (n, e)
- msg, message to be signed, an octet string
- HF, the hash function used to hash the message
- MGF, the mask generation function

Outputs:

- blinded_msg, an octet string of length kLen
- inv, an octet string of length kLen

Errors:

- "message too long": Raised when the input message is too long.
- "encoding error": Raised when the input message fails encoding.
- "invalid blind": Raised when the inverse of r cannot be found.

Steps:

1. `encoded_message = EMSA-PSS-ENCODE(msg, kBits - 1)`
with MGF and HF as defined in the parameters
2. If EMSA-PSS-ENCODE raises an error, raise the error and stop
3. `m = OS2IP(encoded_message)`
4. `r = random_integer_uniform(1, n)`
5. `r_inv = inverse_mod(r, n)`
6. If finding the inverse fails, raise an "invalid blind" error and stop
7. `x = RSAVP1(pkS, r)`
8. `z = m * x mod n`
9. `blinded_msg = I2OSP(z, kLen)`
10. `inv = I2OSP(r_inv, kLen)`
11. output `blinded_msg, inv`

[5.1.2.](#) BlindSign

`rsabssa_blind_sign` performs the RSA private key operation on the client's blinded message input and returns the output encoded as an octet string. `RSASP1` is as defined in [\[RFC3447\]](#).

`rsabssa_blind_sign(skS, blinded_msg)`

Parameters:

- `kLen`, the length in octets of the RSA modulus `n`

Inputs:

- `blinded_msg`, encoded and blinded message to be signed, an octet string

Outputs:

- `blind_sig`, an octet string of length `kLen`

Errors:

- "unexpected input size": Raised when a byte string input doesn't have the expected length.

Steps:

1. If `len(blinded_msg) != kLen`, raise "unexpected input size" and stop
2. `m = OS2IP(blinded_msg)`
3. `s = RSASP1(skS, m)`
4. `blind_sig = I2OSP(s, kLen)`
5. output `blind_sig`

[5.1.3.](#) Finalize

`rsabssa_finalize` validates the server's response, unblinds the message to produce a signature, verifies it for correctness, and outputs the signature upon success. Note that this function will internally hash the input message as is done in `rsabssa_blind`.

`rsabssa_finalize(pkS, msg, blind_sig, inv)`

Parameters:

- `kLen`, the length in octets of the RSA modulus `n`

Inputs:

- `pkS`, server public key
- `msg`, message to be signed, an octet string

- blind_sig, signed and blinded element, an octet string of length kLen
- inv, inverse of the blind, an octet string of length kLen

Outputs:

- sig, an octet string of length kLen

Errors:

- "invalid signature": Raised when the signature is invalid
- "unexpected input size": Raised when a byte string input doesn't have the expected length.

Steps:

1. If $\text{len}(\text{blind_sig}) \neq \text{kLen}$, raise "unexpected input size" and stop
2. If $\text{len}(\text{inv}) \neq \text{kLen}$, raise "unexpected input size" and stop
3. $z = \text{OS2IP}(\text{blind_sig})$
4. $r_inv = \text{OS2IP}(\text{inv})$
5. $s = z * r_inv \bmod n$
6. $\text{sig} = \text{I2OSP}(s, \text{kLen})$
7. $\text{result} = \text{RSASSA-PSS-VERIFY}(\text{pkS}, \text{msg}, \text{sig})$
8. If $\text{result} = \text{"valid signature"}$, output sig, else raise "invalid signature" and stop

[5.2.](#) Encoding Options

The RSASSA-PSS parameters are defined as in [\[RFC8230\]](#).

Implementations MUST support PS384-encoding, using SHA-384 as hash function for the message and mask generation function with a 48-byte salt.

The RSA-PSS encoding functions take the following optional parameters:

- * Hash: hash function (hLen denotes the length in octets of the hash function output)
- * MGF: mask generation function
- * sLen: intended length in octets of the salt

The blinded functions above are orthogonal to the choice of these

options.

6. Public Key Certification

If the server public key is carried in an X.509 certificate, it MUST use the RSASSA-PSS OID [[RFC5756](#)]. It MUST NOT use the rsaEncryption OID [[RFC5280](#)].

7. Security Considerations

Bellare et al. [[BNPS03](#)] proved security of Chaum's original blind signature scheme based on RSA-FDH based on "one-more-RSA-inversion." Note that the design in this document differs only in message encoding, i.e., using PSS instead of FDH.

[[OPEN ISSUE: confirm that results from BNPS03 apply to this construction]]

7.1. Timing Side Channels

rsabssa_blind_sign is functionally a remote procedure call for applying the RSA private key operation. As such, side channel resistance is paramount to protect the private key from exposure [[RemoteTiming](#)]. Implementations MUST include side channel attack mitigations, such as RSA blinding, to avoid leaking information about the private key through timing side channels.

7.2. Message Robustness

An essential property of blind signature schemes is that signer learns nothing of the message being signed. In some circumstances, this may raise concerns of arbitrary signing oracles. Applications using blind signature schemes should take precautions to ensure that such oracles do not cause cross-protocol attacks. This can be done, for example, by keeping blind signature keys distinct from signature keys used for other protocols, such as TLS.

An alternative solution to this problem of message blindness is to give signers proof that the message being signed is well-structured. Depending on the application, zero knowledge proofs could be useful for this purpose. Defining such a proof is out of scope for this document.

[7.3.](#) Salt State

The PSS salt is a randomly generated string chosen when a message is encoded. If the salt is not generated randomly, or is otherwise constructed maliciously, it might be possible for the salt to carry client information to the server. For example, the salt might be maliciously constructed to encode the local IP address of the client. Implementations MUST ensure that the salt is generated correctly to mitigate such issues.

[7.4.](#) Key Substitution Attacks

RSA is well known to permit key substitution attacks, wherein an attacker generates a key pair (skA, pkA) that verify some known (message, signature) pair produced under a different (skS, pkS) key pair [[WM99](#)]. This means it may be possible for an attacker to use a (message, signature) pair from one context in another. Entities that verify signatures must take care to ensure a (message, signature) pair verifies with the expected public key.

[7.5.](#) Alternative RSA Encoding Functions

This document document uses PSS encoding as specified in [[RFC3447](#)] for a number of reasons. First, it is recommended in recent standards, including TLS 1.3 [[RFC8446](#)], X.509v3 [[RFC4055](#)], and even PKCS#1 itself. According to [[RFC3447](#)], "Although no attacks are known against RSASSA-PKCS#1 v1.5, in the interest of increased robustness, RSA-PSS is recommended for eventual adoption in new applications." While RSA-PSS is more complex than RSASSA-PKCS#1 v1.5 encoding, ubiquity of RSA-PSS support influenced the design decision in this draft, despite PKCS#1 v1.5 having equivalent security properties for digital signatures [[JKM18](#)]

Full Domain Hash (FDH) [[RSA-FDH](#)] encoding is also possible, and this variant has equivalent security to PSS [[KK18](#)]. However, FDH is less standard and not used widely in related technologies. Moreover, FDH is deterministic, whereas PSS is probabilistic.

[7.6.](#) Alternative Blind Signature Schemes

There are a number of blind signature protocols beyond RSA. This section summarizes these at a high level, and discusses why an RSA-based variant was chosen for the basis of this specification.

- * Blind Schnorr [[Sch01](#)]: This is a three-message protocol based on the classical Schnorr signature scheme over elliptic curve groups.

Although simple, the hardness problem upon which this is based - Random inhomogeneities in a Overdetermined Solvable system of

linear equations, or ROS - can be broken in polynomial time when a small number of concurrent signing sessions are invoked [[PolytimeROS](#)]. This can lead to signature forgeries in practice. Signers can enforce concurrent sessions, though the limit (approximately 256) for reasonably secure elliptic curve groups is small enough to make large-scale signature generation prohibitive. In contrast, the variant in this specification has no such concurrency limit.

- * Clause Blind Schnorr [[FPS20](#)]: This is a three-message protocol based on a variant of the blind Schnorr signature scheme. This variant of the protocol is not known to be vulnerable to the attack in [[PolytimeROS](#)], though the protocol is still new and under consideration. In the future, this may be a candidate for future blind signatures based on blind signatures. However, the three-message flow necessarily requires two round trips between the client and server, which may be prohibitive for large-scale signature generation. Further analysis and experimentation with this scheme is needed.
- * BSA [[Abe01](#)]: This is a three-message protocol based on elliptic curve groups similar to blind Schnorr. It is also not known to be vulnerable to the ROS attack in [[PolytimeROS](#)]. Kastner et al. [[KLRX20](#)] proved concurrent security with a polynomial number of sessions. For similar reasons to the clause blind Schnorr scheme above, the additional number of round trips requires further analysis and experimentation.
- * Blind BLS [[BLS-Proposal](#)]: The Boneh-Lynn-Shacham [[I-D.irtf-cfrg-bls-signature](#)] scheme can incorporate message blinding when properly instantiated with Type III pairing group. This is a two-message protocol similar to the RSA variant, though it requires pairing support, which is not common in widely deployed cryptographic libraries backing protocols such as TLS. In contrast, the specification in this document relies upon widely deployed cryptographic primitives.

[7.7](#). Post-Quantum Readiness

The blind signature scheme specified in this document is not post-quantum ready since it is based on RSA. (Shor's polynomial-time factorization algorithm readily applies.)

8. IANA Considerations

This document makes no IANA requests.

9. References

Denis, et al.

Expires 9 September 2021

[Page 10]

Internet-Draft

RSA Blind Signatures

March 2021

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", [RFC 3447](#), DOI 10.17487/RFC3447, February 2003, <<https://www.rfc-editor.org/info/rfc3447>>.
- [RFC5756] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Updates for RSAES-OAEP and RSASSA-PSS Algorithm Parameters", [RFC 5756](#), DOI 10.17487/RFC5756, January 2010, <<https://www.rfc-editor.org/info/rfc5756>>.
- [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", [RFC 8017](#), DOI 10.17487/RFC8017, November 2016, <<https://www.rfc-editor.org/info/rfc8017>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8230] Jones, M., "Using RSA Algorithms with CBOR Object Signing and Encryption (COSE) Messages", [RFC 8230](#), DOI 10.17487/RFC8230, September 2017, <<https://www.rfc-editor.org/info/rfc8230>>.

9.2. Informative References

[Abe01] Abe, M., "A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures", DOI 10.1007/3-540-44987-6_9, Lecture Notes in Computer Science pp. 136-151, 2001, <https://doi.org/10.1007/3-540-44987-6_9>.

[BLS-Proposal] "[Privacy-pass] External verifiability: a concrete proposal", n.d., <<https://mailarchive.ietf.org/arch/msg/privacy-pass/BD00hSLwB3uUJcfBiss6nUF5sUA/>>.

Denis, et al.

Expires 9 September 2021

[Page 11]

Internet-Draft

RSA Blind Signatures

March 2021

[BNPS03] Bellare, ., Namprempre, ., Pointcheval, ., and . Semanko, "The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme", DOI 10.1007/s00145-002-0120-1, Journal of Cryptology Vol. 16, pp. 185-215, June 2003, <<https://doi.org/10.1007/s00145-002-0120-1>>.

[Chaum83] "Blind Signatures for Untraceable Payments", <<http://sceweb.sce.uhcl.edu/yang/teaching/csci5234WebSecurityFall2011/Chaum-blind-signatures.PDF>>.

[FPS20] Fuchsbauer, G., Plouviez, A., and Y. Seurin, "Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model", DOI 10.1007/978-3-030-45724-2_3, Advances in Cryptology - EUROCRYPT 2020 pp. 63-95, 2020, <https://doi.org/10.1007/978-3-030-45724-2_3>.

[I-D.davidson-pp-protocol] Davidson, A., "Privacy Pass: The Protocol", Work in Progress, Internet-Draft, [draft-davidson-pp-protocol-01](https://www.ietf.org/internet-drafts/draft-davidson-pp-protocol-01), 13 July 2020, <<http://www.ietf.org/internet-drafts/draft-davidson-pp-protocol-01.txt>>.

[I-D.irtf-cfrg-bls-signature] Boneh, D., Gorbunov, S., Wahby, R., Wee, H., and Z. Zhang,

"BLS Signatures", Work in Progress, Internet-Draft, [draft-irtf-cfrg-bls-signature-04](http://www.ietf.org/internet-drafts/draft-irtf-cfrg-bls-signature-04), 10 September 2020, <<http://www.ietf.org/internet-drafts/draft-irtf-cfrg-bls-signature-04.txt>>.

[I-D.irtf-cfrg-voprf]

Davidson, A., Faz-Hernandez, A., Sullivan, N., and C. Wood, "Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups", Work in Progress, Internet-Draft, [draft-irtf-cfrg-voprf-05](http://www.ietf.org/internet-drafts/draft-irtf-cfrg-voprf-05), 2 November 2020, <<http://www.ietf.org/internet-drafts/draft-irtf-cfrg-voprf-05.txt>>.

[JKK14] "Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only model", <<https://eprint.iacr.org/2014/650>>.

[JKM18] Jager, T., Kakvi, S., and A. May, "On the Security of the PKCS#1 v1.5 Signature Scheme", DOI 10.1145/3243734.3243798, Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, January 2018, <<https://doi.org/10.1145/3243734.3243798>>.

Denis, et al.

Expires 9 September 2021

[Page 12]

Internet-Draft

RSA Blind Signatures

March 2021

[KK18] Kakvi, S. and E. Kiltz, "Optimal Security Proofs for Full Domain Hash, Revisited", DOI 10.1007/s00145-017-9257-9, Journal of Cryptology Vol. 31, pp. 276-306, April 2017, <<https://doi.org/10.1007/s00145-017-9257-9>>.

[KLRX20] "On Pairing-Free Blind Signature Schemes in the Algebraic Group Model", n.d., <<https://eprint.iacr.org/2020/1071>>.

[PolytimeROS]

"On the (in)security of ROS", n.d., <<https://eprint.iacr.org/2020/945.pdf>>.

[RemoteTiming]

"Remote Timing Attacks are Practical", 2003, <<https://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf>>.

[RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in

the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 4055](#), DOI 10.17487/RFC4055, June 2005, <<https://www.rfc-editor.org/info/rfc4055>>.

- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 5280](#), DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", [RFC 8446](#), DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RSA-FDH] "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols", October 1995, <<https://cseweb.ucsd.edu/~mihir/papers/ro.pdf>>.
- [Sch01] Schnorr, C., "Security of Blind Discrete Log Signatures against Interactive Attacks", DOI 10.1007/3-540-45600-7_1, Information and Communications Security pp. 1-12, 2001, <https://doi.org/10.1007/3-540-45600-7_1>.
- [WM99] "Unknown key-share attacks on the station-to-station (STS) protocol", n.d..

[Appendix A](#). Test Vector

This section includes a test vector for the blind signature scheme defined in this document. The following parameters are specified:

- * p, q, n, e, d: RSA private and public key parameters, each encoded as a hexadecimal string.
- * msg: Message being signed, encoded as a hexadecimal string. Its hash is computed using the hash function identified by the 'hash' test vector parameter.

- * salt: Randomly-generated salt used when computing the signature.
- * inv: The message blinding inverse, encoded as a hexadecimal string.
- * blinded_message, blind_sig: The protocol values exchanged during the computation, encoded as hexadecimal strings.
- * sig: The message signature.

p = e1f4d7a34802e27c7392a3cea32a262a34dc3691bd87f3f310dc756734889305
 59c120fd0410194fb8a0da55bd0b81227e843fdca6692ae80e5a5d414116d4803fca
 7d8c30eaaae57e44a1816ebb5c5b0606c536246c7f11985d731684150b63c9a3ad9e
 41b04c0b5b27cb188a692c84696b742a80d3cd00ab891f2457443dadfeba6d6daf10
 8602be26d7071803c67105a5426838e6889d77e8474b29244cefaf418e381b312048
 b457d73419213063c60ee7b0d81820165864fef93523c9635c22210956e53a8d9632
 2493ffc58d845368e2416e078e5bcb5d2fd68ae6acfa54f9627c42e84a9d3f277401
 7e32ebca06308a12ecc290c7cd1156dcccfb2311

q = c601a9caea66dc3835827b539db9df6f6f5ae77244692780cd334a006ab353c8
 06426b60718c05245650821d39445d3ab591ed10a7339f15d83fe13f6a3dfb20b945
 2c6a9b42eaa62a68c970df3caddb2139f804ad8223d56108dfde30ba7d367e9b0a7a8
 0c4fdb2fd9dde6661fc73fc2947569d2029f2870fc02d8325acf28c9afa19ecf962
 daa7916e21afad09eb62fe9f1cf91b77dc879b7974b490d3ebd2e95426057f35d0a3
 c9f45f79ac727ab81a519a8b9285932d9b2e5ccd347e59f3f32ad9ca359115e7da00
 8ab7406707bd0e8e185a5ed8758b5ba266e8828f8d863ae133846304a2936ad7bc7c
 9803879d2fc4a28e69291d73dbd799f8bc238385

n = aec4d69addc70b990ea66a5e70603b6fee27aafebd08f2d94cbe1250c556e047
 a928d635c3f45ee9b66d1bc628a03bac9b7c3f416fe20dabea8f3d7b4bbf7f963be3
 35d2328d67e6c13ee4a8f955e05a3283720d3e1f139c38e43e0338ad058a9495c533
 77fc35be64d208f89b4aa721bf7f7d3fef837be2a80e0f8adf0bcd1eec5bb040443a
 2b2792fdca522a7472aed74f31a1ebe1eebc1f408660a0543dfe2a850f106a617ec6
 685573702eaaa21a5640a5dcaf9b74e397fa3af18a2f1b7c03ba91a6336158de420d
 63188ee143866ee415735d155b7c2d854d795b7bc236cfd71542df34234221a0413
 e142d8c61355cc44d45bda94204974557ac2704cd8b593f035a5724b1adf442e78c5
 42cd4414fce6f1298182fb6d8e53cef1adfd2e90e1e4deec52999bdc6c29144e8d52
 a125232c8c6d75c706ea3cc06841c7bda33568c63a6c03817f722b50fcf898237d78

8a4400869e44d90a3020923dc646388abcc914315215fcd1bae11b1c751fd52443aa
 c8f601087d8d42737c18a3fa11ecd4131ecae017ae0a14acfc4ef85b83c19fed33cf
 d1cd629da2c4c09e222b398e18d822f77bb378dea3cb360b605e5aa58b20edc29d00
 0a66bd177c682a17e7eb12a63ef7c2e4183e0d898f3d6bf567ba8ae84f84f1d23bf8

b8e261c3729e2fa6d07b832e07cddd1d14f55325c6f924267957121902dc19b3b329
48bdead5
e = 010001
d = 0d43242aefe1fb2c13fbc66e20b678c4336d20b1808c558b6e62ad16a2870771
80b177e1f01b12f9c6cd6c52630257ccef26a45135a990928773f3bd2fc01a313f1d
ac97a51cec71cb1fd7efc7adffdeb05f1fb04812c924ed7f4a8269925dad88bd7dcf
bc4ef01020ebfc60cb3e04c54f981fdbd273e69a8a58b8ceb7c2d83fbc6d6f784d05
2201b88a9848186f2a45c0d2826870733e6fd9aa46983e0a6e82e35ca20a439c5ee7
b502a9062e1066493bdadf8b49eb30d9558ed85abc7afb29b3c9bc644199654a4676
681af4babcea4e6f71fe4565c9c1b85d9985b84ec1abf1a820a9bbebee0df1398aae
2c85ab580a9f13e7743afd3108eb32100b870648fa6bc17e8abac4d3c99246b1f0ea
9f7f93a5dd5458c56d9f3f81ff2216b3c3680a13591673c43194d8e6fc93fc1e37ce
2986bd628ac48088bc723d8fbe293861ca7a9f4a73e9fa63b1b6d0074f5dea2a624c
5249ff3ad811b6255b299d6bc5451ba7477f19c5a0db690c3e6476398b1483d10314
afd38bbaf6e2fbdbcd62c3ca9797a420ca6034ec0a83360a3ee2adf4b9d4ba29731d
131b099a38d6a23cc463db754603211260e99d19affc902c915d7854554aabf608e3
ac52c19b8aa26ae042249b17b2d29669b5c859103ee53ef9bdc73ba3c6b537d5c34b
6d8f034671d7f3a8a6966cc4543df223565343154140fd7391c7e7be03e241f4ecfe
b877a051
msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c80023a
a6b59f8cfe5fdbb36331372ebefedae7d
salt = 051722b35f458781397c3a671a7d3bd3096503940e4c4f1aaa269d60300ce
449555cd7340100df9d46944c5356825abf
inv = 80682c48982407b489d53d1261b19ec8627d02b8cda5336750b8cee332ae26
0de57b02d72609c1e0e9f28e2040fc65b6f02d56bdb6aa9af8fde656f70495dfb723
ba01173d4707a12fddac628ca29f3e32340bd8f7ddb557cf819f6b01e445ad96f874
ba235584ee71f6581f62d4f43bf03f910f6510deb85e8ef06c7f09d9794a008be7ff
2529f0ebb69decef646387dc767b74939265fec0223aa6d84d2a8a1cc912d5ca25b4
e144ab8f6ba054b54910176d5737a2cff011da431bd5f2a0d2d66b9e70b39f4b050e
45c0d9c16f02deda9ddf2d00f3e4b01037d7029cd49c2d46a8e1fc2c0c17520af1f4
b5e25ba396afc4cd60c494a4c426448b35b49635b337cfb08e7c22a39b256dd032c0
0adddafb51a627f99a0e1704170ac1f1912e49d9db10ec04c19c58f420212973e0cb
329524223a6aa56c7937c5dffdb5d966b6cd4cbc26f3201dd25c80960a1a111b3294
7bb78973d269fac7f5186530930ed19f68507540eed9e1bab8b00f00d8ca09b3f099
aae46180e04e3584bd7ca054df18a1504b89d1d1675d0966c4ae1407be325cdf623c
f13fff13e4a28b594d59e3eadbadf6136eee7a59d6a444c9eb4e2198e8a974f27a39e
b63af2c9af3870488b8adaad444674f512133ad80b9220e09158521614f1faadfe85
05ef57b7df6813048603f0dd04f4280177a11380fbfc861dbcbd7418d62155248dad
5fdec0991f
encoded_message = 6e0c464d9c2f9fbc147b43570fc4f238e0d0b38870b3addcf7
a4217df912ccef17a7f629aa850f63a063925f312d61d6437be954b45025e8282f9c
0b1131bc8ff19a8a928d859b37113db1064f92a27f64761c181c1e1f9b251ae5a2f8
a4047573b67a270584e089beadcb13e7c82337797119712e9b849ff56e04385d144d
3ca9d8d92bf78adb20b5bbeb3685f17038ec6afade3ef354429c51c687b45a7018ee

3a6966b3af15c9ba8f40e6461ba0a17ef5a799672ad882bab02b518f9da7c1a96294
5c2e9b0f02f29b31b9cdf3e633f9d9d2a22e96e1de28e25241ca7dd04147112f5789
73403e0f4fd80865965475d22294f065e17a1c4a201de93bd14223e6b1b999fd548f
2f759f52db71964528b6f15b9c2d7811f2a0a35d534b8216301c47f4f04f412cae14
2b48c4cdf78bc54df690fd43142d750c671dd8e2e938e6a440b2f825b6dbb3e19f1
d7a3c0150428a47948037c322365b7fe6fe57ac88d8f80889e9ff38177bad8c8d8d9
8db42908b389cb59692a58ce275aa15acb032ca951b3e0a3404b7f33f655b7c7d83a
2f8d1b6bbff49d5fcedf2e030e80881aa436db27a5c0dea13f32e7d460dbf01240c2
320c2bb5b3225b17145c72d61d47c8f84d1e19417ebd8ce3638a82d395cc6f7050b6
209d9283dc7b93fecc04f3f9e7f566829ac41568ef799480c733c09759aa9734e201
3d7640dc6151018ea902bc

blinded_msg = 10c166c6a711e81c46f45b18e5873cc4f494f003180dd7f115
585d871a28930259654fe28a54dab319cc5011204c8373b50a57b0fdc7a678bd74c5
23259dfe4fd5ea9f52f170e19dfa332930ad1609fc8a00902d725cfe50685c95e5b2
968c9a2828a21207fcf393d15f849769e2af34ac4259d91dfd98c3a707c509e1af55
647efaa31290ddf48e0133b798562af5eabd327270ac2fb6c594734ce339a14ea4fe
1b9a2f81c0bc230ca523bda17ff42a377266bc2778a274c0ae5ec5a8cbbe364fcf0d
2403f7ee178d77ff28b67a20c7ceec009182dbcaa9bc99b51ebbf13b7d542be33717
2c6474f2cd3561219fe0dfa3fb207cfff89632091ab841cf38d8aa88af6891539f263
adb8eac6402c41b6ebd72984e43666e537f5f5fe27b2b5aa114957e9a580730308a5
f5a9c63a1eb599f093ab401d0c6003a451931b6d124180305705845060ebba6b0036
154fcef3e5e9f9e4b87e8f084542fd1dd67e7782a5585150181c01eb6d90cb958838
37384a5b91dbb606f266059ecc51b5acbaa280e45cfd2eec8cc1cdb1b7211c8e1480
5ba683f9b78824b2eb005bc8a7d7179a36c152cb87c8219e5569bba911bb32a1b923
ca83de0e03fb10fba75d85c55907dda5a2606bf918b056c3808ba496a4d955322120
40a5f44f37e1097f26dc27b98a51837daa78f23e532156296b64352669c94a8a855a
cf30533d8e0594ace7c442

blind_sig = 364f6a40dbfbc3bbb257943337eeff791a0f290898a67912
83bba581d9eac90a6376a837241f5f73a78a5c6746e1306ba3adab6067c32ff69115
734ce014d354e2f259d4cbfb890244fd451a497fe6ecf9aa90d19a2d441162f7eaa7
ce3fc4e89fd4e76b7ae585be2a2c0fd6fb246b8ac8d58bcb585634e30c9168a43478
6fe5e0b74bfe8187b47ac091aa571ffea0a864cb906d0e28c77a00e8cd8f6aba4317
a8cc7bf32ce566bd1ef80c64de041728abe087bee6cadd0b7062bde5ceef308a23bd
1ccc154fd0c3a26110df6193464fc0d24ee189aea8979d722170ba945fdcce9b1b4b
63349980f3a92dc2e5418c54d38a862916926b3f9ca270a8cf40dfb9772bfbdd9a3e
0e0892369c18249211ba857f35963d0e05d8da98f1aa0c6bba58f47487b8f663e395
091275f82941830b050b260e4767ce2fa903e75ff8970c98bfb3a08d6db91ab1746c
86420ee2e909bf681cac173697135983c3594b2def673736220452fde4ddec867d40
ff42dd3da36c84e3e52508b891a00f50b4f62d112edb3b6b6cc3dbd546ba10f36b03
f06c0d82aeec3b25e127af545fac28e1613a0517a6095ad18a98ab79f68801e05c17
5e15bae21f821e80c80ab4fdec6fb34ca315e194502b8f3dcf7892b511aee45060e3
994cd15e003861bc7220a2babd7b40eda03382548a34a7110f9b1779bf3ef6011361
611e6bc5c0dc851e1509de1a

sig = 6fef8bf9bc182cd8cf7ce45c7dcf0e6f3e518ae48f06f3c670c649ac737a8b
8119a34d51641785be151a697ed7825fdfece82865123445eab03eb4bb91cecf4d69
51738495f8481151b62de869658573df4e50a95c17c31b52e154ae26a04067d5ecdc
1592c287550bb982a5bb9c30fd53a768cee6baabb3d483e9f1e2da954c7f4cf492fe

Internet-Draft

RSA Blind Signatures

March 2021

b81ae34fb7a31ee6b7ed641cb16c2ac999004c2191de0201457523f5a4700dd64926
7d9286f5c1d193f1454c9f868a57816bf5ff76c838a2eeb616a3fc9976f65d4371de
ecfbab29362caebdff69c635fe5a2113da4d4d8c24f0b16a0584fa05e80e607c5d9a
2f765f1f069f8d4da21f27c2a3b5c984b4ab24899bef46c6d9323df4862fe51ce300
fca40fb539c3bb7fe2dcc9409e425f2d3b95e70e9c49c5feb6ecc9d43442c33d5000
3ee936845892fb8be475647da9a080f5bc7f8a716590b3745c2209fe05b17992830c
e15f32c7b22cde755c8a2fe50bd814a0434130b807dc1b7218d4e85342d70695a5d7
f29306f25623ad1e8aa08ef71b54b8ee447b5f64e73d09bdd6c3b7ca224058d7c67c
c7551e9241688ada12d859cb7646fbd3ed8b34312f3b49d69802f0eaa1bc4211c2f
7a29cd5c01ed01a39001c5856fab36228f5ee2f2e1110811872fe7c865c42ed59029
c706195d52

Authors' Addresses

Frank Denis
Fastly Inc.

Email: fd@00f.net

Frederic Jacobs
Apple Inc.

Email: frederic.jacobs@apple.com

Christopher A. Wood
Cloudflare

Email: caw@heapingbits.net

