Workgroup: Network Working Group
Internet-Draft:
draft-irtf-cfrg-rsa-blind-signatures-06
Published: 21 November 2022
Intended Status: Informational
Expires: 25 May 2023
Authors: F. Denis
Fastly Inc.
F. Jacobs
Apple Inc.
C. A. Wood
Cloudflare

RSA Blind Signatures

Abstract

This document specifies an RSA-based blind signature protocol. 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 25 May 2023.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

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 applications that use Verifiable Oblivious Pseudorandom Functions (VOPRFs) [I-D.irtf-cfrg-voprfs], such as Privacy Pass [I-D.ietf-privacypass-protocol]. Specifically, VOPRFs are not necessarily publicly verifiable, meaning that a verifier needs access to the VOPRF private key to verify that the output of a VOPRF protocol is valid for a given input. This limitation complicates deployments where it is not desirable to distribute private keys to entities performing verification. Additionally, if the private key is kept in a Hardware Security Module, the number of operations on the key is doubled compared to a scheme where only the public key is required for verification.

In contrast, digital 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 signature-message pair, where the signature is computed using from a deterministic blind signature protocol.

This document specifies a protocol for computing the RSA blind signatures using RSA-PSS encoding, and a family of variants for this protocol, denoted RSABSSA. In order to facilitate deployment, it is defined 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:

* bytes_to_int and int_to_bytes: Convert a byte string to and from a non-negative integer. bytes_to_int and int_to_bytes are implemented as OS2IP and I2OSP as described in [RFC8017], respectively. 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 <= R < N.

* inverse_mod(x, n): Compute the multiplicative inverse of x mod n. This function fails if x and n are not co-prime.

*len(s): The length of a byte string, in bytes.

*random(n): Generate n random bytes using a cryptographically-secure pseudorandom number generator.

4. Blind Signature Protocol

The core RSA Blind Signature Protocol is a two-party protocol between a client and server where they 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.

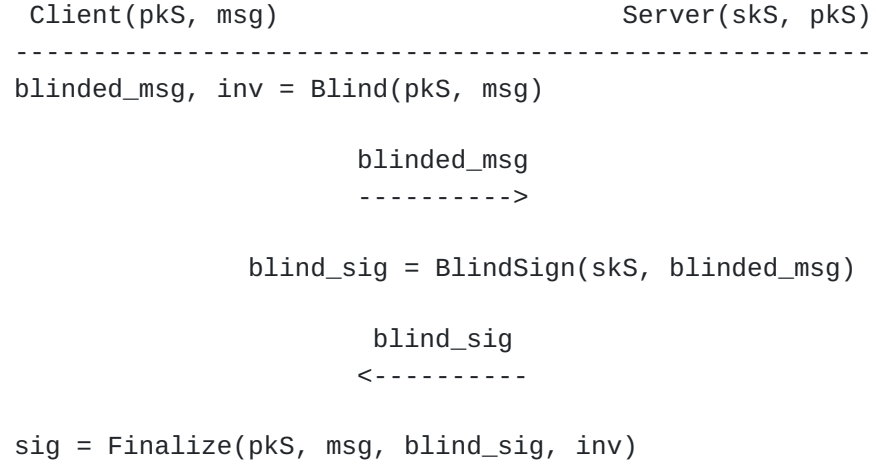
The protocol consists of three functions, Blind, BlindSign, and Finalize, described below.

*Blind(pkS, msg): encodes an input message msg and blinds it with a randomly generated blinding factor using the server's public key pkS, producing a blinded messsage for the server and the inverse of the blind.

*BlindSign(skS, blinded_msg): performs the raw RSA private key operation using private key skS on the client's blinded message blinded_msg, and returns the output.

*Finalize(pkS, msg, blinded_sig, inv): unblinds the server's response blinded_sig using the inverse inv to produce a signature, verifies it for correctness using message msg and public key pkS, and outputs the signature upon success.

Using these three functions, the core protocol runs as follows:



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 Section

8.1.2 of [RFC8017]. The Finalize function performs that check before returning the signature.

In the remainder of this section, we specify Blind, BlindSign, and Finalize.

4.1. Blind

The Blind function encodes an input message and blinds it with the server's public key. It outputs the blinded message to be sent to the server, encoded as a byte string, and the corresponding inverse, an integer. RSAVP1 and EMSA-PSS-ENCODE are as defined in Section 5.2.2 and Section 9.1.1 of [RFC8017], respectively. If this function fails with an "invalid blind" error, implementations SHOULD retry the function again. The probability of multiple such errors in sequence is negligible.

```
Blind(pkS, msg)

Parameters:
- kLen, the length in bytes of the RSA modulus n
- Hash, the hash function used to hash the message
- MGF, the mask generation function
- sLen, the length in bytes of the salt

Inputs:
- pkS, server public key (n, e)
- msg, message to be signed, a byte string

Outputs:
- blinded_msg, a byte string of length kLen
- inv, an integer

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_msg = EMSA-PSS-ENCODE(msg, (kLen * 8) - 1)
   with Hash, MGF, and sLenInBytes as defined in the parameters
2. If EMSA-PSS-ENCODE raises an error, raise the error and stop
3. m = bytes_to_int(encoded_msg)
4. r = random_integer_uniform(1, n)
5. inv = inverse_mod(r, n)
6. If inverse_mod fails, raise an "invalid blind" error
   and stop
7. x = RSAVP1(pkS, r)
8. z = m * x mod n
9. blinded_msg = int_to_bytes(z, kLen)
10. output blinded_msg, inv
```

The blinding factor r MUST be randomly chosen from a uniform distribution. This is typically done via rejection sampling.

4.2. BlindSign

BlindSign performs the RSA private key operation on the client's blinded message input and returns the output encoded as a byte string. RSASP1 is as defined in Section 5.2.1 of [RFC8017].

```
BlindSign(skS, blinded_msg)

Parameters:
- kLen, the length in bytes of the RSA modulus n

Inputs:
- skS, server private key
- blinded_msg, encoded and blinded message to be signed, an
  byte string

Outputs:
- blind_sig, a byte string of length kLen

Errors:
- "unexpected input size": Raised when a byte string input doesn't
  have the expected length.
- "invalid message": Raised when the message representative
  to sign is not an integer between 0 and n - 1.

Steps:
1. If len(blinded_msg) != kLen, raise "unexpected input size"
   and stop
2. m = bytes_to_int(blinded_msg)
3. If m >= n, raise "invalid message" and stop
4. s = RSASP1(skS, m)
5. blind_sig = int_to_bytes(s, kLen)
6. output blind_sig
```

4.3. Finalize

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 Blind.

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

Parameters:
- kLen, the length in bytes of the RSA modulus n
- Hash, the hash function used to hash the message
- MGF, the mask generation function
- sLen, the length in bytes of the salt

Inputs:
- pkS, server public key (n, e)
- msg, message to be signed, a byte string
- blind_sig, signed and blinded element, a byte string of
  length kLen
- inv, inverse of the blind, an integer

Outputs:
- sig, a byte 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 len(blind_sig) != kLen, raise "unexpected input size" and stop
2. z = bytes_to_int(blind_sig)
3. s = z * inv mod n
4. sig = int_to_bytes(s, kLen)
5. result = RSASSA-PSS-VERIFY(pkS, msg, sig) with
   Hash, MGF, and sLenInBytes as defined in the parameters
6. If result = "valid signature", output sig, else
   raise "invalid signature" and stop
```

5. Message Randomization

Message randomization is the process by which the message to be signed and verified is augmented with fresh randomness. As such, message randommization is a procedure invoked before the protocol in Section 4, as it changes the contents of the message being signed. We denote this process by the function Randomize(msg), which takes as input a message msg and produces a randomized message randomMsg. Its implementation is shown below.

```
Randomize(msg)

Inputs:
- msg, message to be signed, a byte string

Outputs:
- randomMsg, a byte string that is 32 bytes longer than input msg

Steps:
1. msgPrefix = random(32)
2. randomMsg = msgPrefix || msg
3. output randomMsg
```

6. RSABSSA Variants

In this section we define different named variants of RSABSSA. Each variant specifies a hash function, RSASSA-PSS parameters as defined in [RFC8017], Section 9.1.1, and whether or not message randomization (as described in Section 5) is performed on the input message. Future specifications can introduce other variants as desired. The named variants are as follows:

1. RSABSSA-SHA384-PSS-Randomized: This named variant uses SHA-384 as the hash function, MGF1 with SHA-384 as the PSS mask generation function, a 48-byte salt length, and uses message randomization.

2. RSABSSA-SHA384-PSSZERO-Randomized: This named variant uses SHA-384 as the hash function, MGF1 with SHA-384 as the PSS mask generation function, an empty PSS salt, and uses message randomization.

3. RSABSSA-SHA384-PSS-Deterministic: This named variant uses SHA-384 as the hash function, MGF1 with SHA-384 as the PSS mask generation function, 48-byte salt length, and does not use message randomization.

4. RSABSSA-SHA384-PSSZERO-Deterministic: This named variant uses SHA-384 as the hash function, MGF1 with SHA-384 as the PSS mask generation function, an empty PSS salt, and does not use message randomization. This is the only variant that produces deterministic signatures over the input message.

The RECOMMENDED variants are RSABSSA-SHA384-PSS-Randomized or RSABSSA-SHA384-PSSZERO-Randomized.

Not all named variants can be used interchangeably. In particular, applications that provide high-entropy input messages can safely use named variants without message randomization, as the additional

message randomization does not offer security advantages. See [Lys22] and Section 8.3 for more information.

Applications that require deterministic signatures can use the RSABSSA-SHA384-PSSZERO-Deterministic variant, but only if their input messages have high entropy. Applications that use RSABSSA-SHA384-PSSZERO-Deterministic SHOULD carefully analyze the security implications, taking into account the possibility of adversarially generated signer keys as described in Section 8.3. When it is not clear whether an application requires deterministic or randomized signatures, applications SHOULD use one of the recommended variants with message randomization.

7. Implementation and Usage Considerations

This section documents considerations for interfaces to implementations of the protocol in this document. This includes error handling and API considerations.

7.1. Errors

The high-level functions specified in Section 4 are all fallible. The explicit errors generated throughout this specification, along with the conditions that lead to each error, are listed in the definitions for Blind, BlindSign, and Finalize. These errors are meant as a guide for implementors. They are not an exhaustive list of all the errors an implementation might emit. For example, implementations might run out of memory.

7.2. Signing Key Usage and Certification

A server signing key MUST NOT be reused for any other protocol beyond RSABSSA. Moreover, a server signing key MUST NOT be reused for different RSABSSA encoding options. That is, if a server supports two different encoding options, then it MUST have a distinct key pair for each option.

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].

8. Security Considerations

Bellare et al. [BNPS03] proved the following properties of Chaum's original blind signature protocol based on RSA-FDH:

* One-more-forgery polynomial security. This means the adversary, interacting with the server (signer) as a client, cannot output n+1 valid message and signature tuples after only interacting

with the server n times, for some n which is polynomial in the protocol's security parameter.

*Concurrent polynomial security. This means that servers can engage in polynomially many invocations of the protocol without compromising security.

Both results rely upon the RSA Known Target Inversion Problem being hard. However, this analysis is incomplete as it does not account for adversarially-generated keys. This threat model has important implications for appliations using the blind signature protocol described in this document; see Section 8.3 for more details.

Lastly, the design in this document differs from the analysis in [BNPS03] only in message encoding, i.e., using PSS instead of FDH. Note, importantly, that an empty salt effectively reduces PSS to FDH, so the same results apply.

8.1. Timing Side Channels and Fault Attacks

BlindSign 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 [RemoteTimingAttacks]. Implementations SHOULD implement some form of side channel attack mitigation, such as RSA blinding as described in Section 10 of [TimingAttacks]. Failure to apply such mitigations can lead to side channel attacks that leak the private signing key.

Beyond timing side channels, [FAULTS] describes the importance of implementation safeguards that protect against fault attacks that can also leak the private signing key. These safeguards require that implementations check that the result of the private key operation when signing is correct, i.e., given s = RSASP1(skS, m), verify that m = RSAVP1(pkS, s). Implementations SHOULD apply this (or equivalent) safeguard to mitigate fault attacks, even if they are not implementations based on the Chinese remainder theorem.

8.2. Message Robustness

An essential property of blind signature protocols is that the signer learns nothing of the message being signed. In some circumstances, this may raise concerns of arbitrary signing oracles. Applications using blind signature protocols 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.

Verifiers should check that, in addition to signature validity, the signed message is well-structured for the relevant application. For example, if an application of this protocol requires messages to be structures of a particular form, then verifiers should check that messages adhere to this form.

8.3. Message Entropy

As discussed in [Lys22], a malicious signer can construct an invalid public key and use it to learn information about low-entropy with input messages. Note that some invalid public keys may not yield valid signatures when run with the protocol, e.g., because the signature fails to verify. However, if an attacker can coerce the client to use these invalid public keys with low-entropy inputs, they can learn information about the client inputs before the protocol completes.

Based on this fact, using the core protocol functions in Section 4 is possibly unsafe, unless one of the following conditions are met:

1. The client has proof that the signer's public key is honestly generated. [GRSB19] presents some (non-interactive) honest-verifier zero-knoweldge proofs of various statements about the public key.

2. The client input message has high entropy.

The named variants that use message randomization -- RSABSSA-SHA384-PSS-Randomized and RSABSSA-SHA384-PSSZERO-Randomized -- explicitly inject fresh entropy alongside each message to satisfy condition (2). As such, these variants are safe for all application use cases.

Note that these variants effectively mean that the resulting signature is always randomized. As such, this interface is not suitable for applications that require deterministic signatures.

8.4. PSS Salt Choice

For variants that have a non-zero PSS salt, the salt MUST be generated from a cryptographically secure pseudorandom number generator. If the PSS salt is not generated randomly, or is otherwise constructed maliciously, it might be possible for the salt to encode information that is not present in the signed message. For example, the salt might be maliciously constructed to encode the local IP address of the client. As a result, implementations SHOULD NOT allow clients to provide the salt directly.

8.5. 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 a valid public key from the expected issuer.

8.6. Alternative RSA Encoding Functions

This document document uses PSS encoding as specified in [RFC8017] 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 [RFC8017], "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 supports deterministic and probabilistic encodings.

8.7. Alternative Blind Signature Protocols

RSA has some advantages as a signature protocol, particularly around verification efficiency. However, the protocol in this document is not without shortcomings, including:

- RSA key and signature sizes are larger than those of alternative blind signature protocols;
- No evaluation batching support, which means that the cost of the protocol scales linearly with the number of invocations; and
- Extensions for features such as threshold signing are more complex to instantiate compared to other protocols based on, for example, Schnorr signatures.

There are a number of blind signature protocols beyond blind RSA. This section summarizes these at a high level, and discusses why an

RSA-based variant was chosen for the basis of this specification, despite the shortcomings above.

* Blind Schnorr [Sch01]: This is a three-message protocol based on the classical Schnorr signature protocol 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], leading to signature forgeries. Even with small concurrency limits, Wagner's generalized attack [Wagner02] leads to subexponential forgery speedup. For example, a limit of 15 parallel sessions yields an attack runtime of approximately 2^55, which is substantially lower than acceptable security levels. 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 protocol. 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. 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 protocol 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 protocol above, the additional number of round trips requires further analysis and experimentation.

* WFROS-based Schemes [TZ22]: This work contains four proposed schemes, each of which are three-message protocols based on a variant of the blind Schnorr signature protocol. Security of these schemes depend on the Weighted Fractional ROS problem, for which the authors prove an exponential and unconditional lower bound, and therefore have tighter security bounds than the Clause Blind Schnorr schemes. The performance of the scheme is similar to Clause Blind Schnorr, yet signing is more efficient. Similarly to Clause Blind Schnorr schemes, the two round trips required by three-message flows need further analysis and experimentation.

* Blind BLS [BLS-Proposal]: The Boneh-Lynn-Shacham [I-D.irtf-cfrg-bls-signature] protocol 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.

Beyond blind signature protocols, anonymous credential schemes with public verifiability such as U-Prove [UProve] may be used instead of blind signature protocols. Anonymous credentials may even be constructed with blind signature protocols. However, anonymous credentials are higher-level constructions that present a richer feature set.

8.8. Post-Quantum Readiness

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

9. IANA Considerations

This document makes no IANA requests.

10. References

10.1. 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/rfc/rfc2119>.

[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/rfc/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/rfc/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/rfc/rfc8174>.

10.2. Informative References

[Abe01] Abe, M., "A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures", Lecture Notes in Computer

Science pp. 136-151, DOI 10.1007/3-540-44987-6_9, 2001, <https://doi.org/10.1007/3-540-44987-6_9>.

[BLS-Proposal] "[Privacy-pass] External verifiability: a concrete proposal", July 2020, <https://mailarchive.ietf.org/arch/msg/privacy-pass/BDOOhSLwB3uUJcfBiss6nUF5sUA/>.

[BNPS03] Bellare, Namprempre, Pointcheval, and Semanko, "The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme", Journal of Cryptology vol. 16, no. 3, pp. 185-215, DOI 10.1007/s00145-002-0120-1, June 2003, <https://doi.org/10.1007/s00145-002-0120-1>.

[Chaum83] "Blind Signatures for Untraceable Payments", 1983, <http://sceweb.sce.uhcl.edu/yang/teaching/csci5234WebSecurityFall2011/Chaum-blind-signatures.PDF>.

[FAULTS] Boneh, D., DeMillo, R., and R. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults", Advances in Cryptology - EUROCRYPT '97 pp. 37-51, DOI 10.1007/3-540-69053-0_4, 1997, <https://doi.org/10.1007/3-540-69053-0_4>.

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

[GRSB19] "Efficient Noninteractive Certification of RSA Moduli and Beyond", October 2019, <https://eprint.iacr.org/2018/057.pdf>.

[I-D.ietf-privacypass-protocol] Celi, S., Davidson, A., Faz-Hernandez, A., Valdez, S., and C. A. Wood, "Privacy Pass Issuance Protocol", Work in Progress, Internet-Draft, draft-ietf-privacypass-protocol-06, 6 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-protocol-06>.

[I-D.irtf-cfrg-bls-signature] Boneh, D., Gorbunov, S., Wahby, R. S., Wee, H., Wood, C. A., and Z. Zhang, "BLS Signatures", Work in Progress, Internet-Draft, draft-irtf-cfrg-bls-signature-05, 16 June 2022, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-bls-signature-05>.

[I-D.irtf-cfrg-voprfs] "*** BROKEN REFERENCE ***".

[JKK14] "Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only model", August 2014, <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", Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, DOI 10.1145/3243734.3243798, January 2018, <https://doi.org/10.1145/3243734.3243798>.

[KK18] Kakvi, S. and E. Kiltz, "Optimal Security Proofs for Full Domain Hash, Revisited", Journal of Cryptology vol. 31, no. 1, pp. 276-306, DOI 10.1007/s00145-017-9257-9, April 2017, <https://doi.org/10.1007/s00145-017-9257-9>.

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

[Lys22] "Security Analysis of RSA-BSSA", n.d., <https://eprint.iacr.org/2022/895>.

[PolytimeROS] "On the (in)security of ROS", July 2020, <https://eprint.iacr.org/2020/945>.

[RemoteTimingAttacks] "Remote Timing Attacks are Practical", May 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/rfc/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/rfc/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/rfc/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", Information and Communications Security pp. 1-12, DOI 10.1007/3-540-45600-7_1, 2001, <https://doi.org/10.1007/3-540-45600-7_1>.

[TimingAttacks] Kocher, P., "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", Advances in Cryptology - CRYPTO '96 pp. 104-113, DOI 10.1007/3-540-68697-5_9, 1996, <https://doi.org/10.1007/3-540-68697-5_9>.

[TZ22] "Short Pairing-Free Blind Signatures with Exponential Security", January 2022, <https://eprint.iacr.org/2022/047>.

[UProve] "U-Prove", February 2012, <https://www.microsoft.com/en-us/research/project/u-prove/>.

[Wagner02] Wagner, D., "A Generalized Birthday Problem", Advances in Cryptology - CRYPTO 2002 pp. 288-304, DOI 10.1007/3-540-45708-9_19, 2002, <https://doi.org/10.1007/3-540-45708-9_19>.

[WM99] "Unknown key-share attacks on the station-to-station (STS) protocol", October 1999.

Appendix A. Test Vectors

This section includes test vectors for the blind signature protocol defined in Section 4. The following parameters are specified for each test vector:

- *p, q, n, e, d: RSA private and public key parameters, each encoded as a hexadecimal string.
- *msg: Input messsage being signed, encoded as a hexadecimal string. The hash is computed using SHA-384.
- *msg_prefix: Message randomizer prefix, encoded as a hexadecimal string. This is only present for variants that use message randomization.
- *random_msg: The message actually signed. If the variant does not use message randomization, this is equal to msg.
- *salt: Randomly-generated salt used when computing the signature. The length is either 48 or 0 bytes.

*encoded_msg: EMSA-PSS encoded message. The mask generation
 function is MGF1 with SHA-384.

*inv: The message blinding inverse, encoded as a hexadecimal
 string.

*blinded_msg, blind_sig: The protocol values exchanged during the
 computation, encoded as hexadecimal strings.

*sig: The output message signature.

A.1. RSABSSA-SHA384-PSS-Randomized Test Vector

```
p = 0xe1f4d7a34802e27c7392a3cea32a262a34dc3691bd87f3f310dc7567348893
0559c120fd0410194fb8a0da55bd0b81227e843fdca6692ae80e5a5d414116d4803f
ca7d8c30eaaae57e44a1816ebb5c5b0606c536246c7f11985d731684150b63c9a3ad
9e41b04c0b5b27cb188a692c84696b742a80d3cd00ab891f2457443dadfeba6d6daf
108602be26d7071803c67105a5426838e6889d77e8474b29244cefaf418e381b3120
48b457d73419213063c60ee7b0d81820165864fef93523c9635c22210956e53a8d96
322493ffc58d845368e2416e078e5bcb5d2fd68ae6acfa54f9627c42e84a9d3f2774
017e32ebca06308a12ecc290c7cd1156dcccfb2311
q = 0xc601a9caea66dc3835827b539db9df6f6f5ae77244692780cd334a006ab353
c806426b60718c05245650821d39445d3ab591ed10a7339f15d83fe13f6a3dfb20b9
452c6a9b42eaa62a68c970df3cadb2139f804ad8223d56108dfde30ba7d367e9b0a7
a80c4fdba2fd9dde6661fc73fc2947569d2029f2870fc02d8325acf28c9afa19ecf9
62daa7916e21afad09eb62fe9f1cf91b77dc879b7974b490d3ebd2e95426057f35d0
a3c9f45f79ac727ab81a519a8b9285932d9b2e5ccd347e59f3f32ad9ca359115e7da
008ab7406707bd0e8e185a5ed8758b5ba266e8828f8d863ae133846304a2936ad7bc
7c9803879d2fc4a28e69291d73dbd799f8bc238385
n = 0xaec4d69addc70b990ea66a5e70603b6fee27aafebd08f2d94cbe1250c556e0
47a928d635c3f45ee9b66d1bc628a03bac9b7c3f416fe20dabea8f3d7b4bbf7f963b
e335d2328d67e6c13ee4a8f955e05a3283720d3e1f139c38e43e0338ad058a9495c5
3377fc35be64d208f89b4aa721bf7f7d3fef837be2a80e0f8adf0bcd1eec5bb04044
3a2b2792fdca522a7472aed74f31a1ebe1eebc1f408660a0543dfe2a850f106a617e
c6685573702eaaa21a5640a5dcaf9b74e397fa3af18a2f1b7c03ba91a6336158de42
0d63188ee143866ee415735d155b7c2d854d795b7bc236cffd71542df34234221a04
13e142d8c61355cc44d45bda94204974557ac2704cd8b593f035a5724b1adf442e78
c542cd4414fce6f1298182fb6d8e53cef1adfd2e90e1e4deec52999bdc6c29144e8d
52a125232c8c6d75c706ea3cc06841c7bda33568c63a6c03817f722b50fcf898237d
788a4400869e44d90a3020923dc646388abcc914315215fcd1bae11b1c751fd52443
aac8f601087d8d42737c18a3fa11ecd4131ecae017ae0a14acfc4ef85b83c19fed33
cfd1cd629da2c4c09e222b398e18d822f77bb378dea3cb360b605e5aa58b20edc29d
000a66bd177c682a17e7eb12a63ef7c2e4183e0d898f3d6bf567ba8ae84f84f1d23b
f8b8e261c3729e2fa6d07b832e07cddd1d14f55325c6f924267957121902dc19b3b3
2948bdead5
e = 0x10001
d = 0xd43242aefe1fb2c13fbc66e20b678c4336d20b1808c558b6e62ad16a287077
180b177e1f01b12f9c6cd6c52630257ccef26a45135a990928773f3bd2fc01a313f1
dac97a51cec71cb1fd7efc7adffdeb05f1fb04812c924ed7f4a8269925dad88bd7dc
fbc4ef01020ebfc60cb3e04c54f981fdbd273e69a8a58b8ceb7c2d83fbcbd6f784d0
52201b88a9848186f2a45c0d2826870733e6fd9aa46983e0a6e82e35ca20a439c5ee
7b502a9062e1066493bdadf8b49eb30d9558ed85abc7afb29b3c9bc644199654a467
6681af4babcea4e6f71fe4565c9c1b85d9985b84ec1abf1a820a9bbebee0df1398aa
e2c85ab580a9f13e7743afd3108eb32100b870648fa6bc17e8abac4d3c99246b1f0e
a9f7f93a5dd5458c56d9f3f81ff2216b3c3680a13591673c43194d8e6fc93fc1e37c
e2986bd628ac48088bc723d8fbe293861ca7a9f4a73e9fa63b1b6d0074f5dea2a624
c5249ff3ad811b6255b299d6bc5451ba7477f19c5a0db690c3e6476398b1483d1031
4afd38bbaf6e2fbdbcd62c3ca9797a420ca6034ec0a83360a3ee2adf4b9d4ba29731
d131b099a38d6a23cc463db754603211260e99d19affc902c915d7854554aabf608e
3ac52c19b8aa26ae042249b17b2d29669b5c859103ee53ef9bdc73ba3c6b537d5c34
b6d8f034671d7f3a8a6966cc4543df2235653431541 40fd7391c7e7be03e241f4ecf
eb877a051
```

```
msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c80023a
a6b59f8cfec5fdbb36331372ebefedae7d
msg_prefix = 7283fe9dc3a4b6622e98fa8ebab99ff8b4717b8d5195dc54502c034
b1fde79ed
random_msg = 7283fe9dc3a4b6622e98fa8ebab99ff8b4717b8d5195dc54502c034
b1fde79ed8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c800
23aa6b59f8cfec5fdbb36331372ebefedae7d
salt = 051722b35f458781397c3a671a7d3bd3096503940e4c4f1aaa269d60300ce
449555cd7340100df9d46944c5356825abf
encoded_msg = 6e0c464d9c2f9fbc147b43570fc4f238e0d0b38870b3addcf7a421
7df912ccef17a7f629aa850f63a063925f312d61d6437be954b45025e8282f9c0b11
31bc8ff19a8a928d859b37113db1064f92a27f64761c181c1e1f9b251ae5a2f8a404
7573b67a270584e089beadcb13e7c82337797119712e9b849ff56e04385d144d3ca9
d8d92bf78adb20b5bbeb3685f17038ec6afade3ef354429c51c687b45a7018ee3a69
66b3af15c9ba8f40e6461ba0a17ef5a799672ad882bab02b518f9da7c1a962945c2e
9b0f02f29b31b9cdf3e633f9d9d2a22e96e1de28e25241ca7dd04147112f57897340
3e0f4fd80865965475d22294f065e17a1c4a201de93bd14223e6b1b999fd548f2f75
9f52db71964528b6f15b9c2d7811f2a0a35d534b8216301c47f4f04f412cae142b48
c4cdff78bc54df690fd43142d750c671dd8e2e938e6a440b2f825b6dbb3e19f1d7a3
c0150428a47948037c322365b7fe6fe57ac88d8f80889e9ff38177bad8c8d8d98db4
2908b389cb59692a58ce275aa15acb032ca951b3e0a3404b7f33f655b7c7d83a2f8d
1b6bbff49d5fcedf2e030e80881aa436db27a5c0dea13f32e7d460dbf01240c2320c
2bb5b3225b17145c72d61d47c8f84d1e19417ebd8ce3638a82d395cc6f7050b6209d
9283dc7b93fecc04f3f9e7f566829ac41568ef799480c733c09759aa9734e2013d76
40dc6151018ea902bc
inv = 80682c48982407b489d53d1261b19ec8627d02b8cda5336750b8cee332ae26
0de57b02d72609c1e0e9f28e2040fc65b6f02d56dbd6aa9af8fde656f70495dfb723
ba01173d4707a12fddac628ca29f3e32340bd8f7ddb557cf819f6b01e445ad96f874
ba235584ee71f6581f62d4f43bf03f910f6510deb85e8ef06c7f09d9794a008be7ff
2529f0ebb69decef646387dc767b74939265fec0223aa6d84d2a8a1cc912d5ca25b4
e144ab8f6ba054b54910176d5737a2cff011da431bd5f2a0d2d66b9e70b39f4b050e
45c0d9c16f02deda9ddf2d00f3e4b01037d7029cd49c2d46a8e1fc2c0c17520af1f4
b5e25ba396afc4cd60c494a4c426448b35b49635b337cfb08e7c22a39b256dd032c0
0adddafb51a627f99a0e1704170ac1f1912e49d9db10ec04c19c58f420212973e0cb
329524223a6aa56c7937c5dffdb5d966b6cd4cbc26f3201dd25c80960a1a111b3294
7bb78973d269fac7f5186530930ed19f68507540eed9e1bab8b00f00d8ca09b3f099
aae46180e04e3584bd7ca054df18a1504b89d1d1675d0966c4ae1407be325cdf623c
f13ff13e4a28b594d59e3eadbadf6136eee7a59d6a444c9eb4e2198e8a974f27a39e
b63af2c9af3870488b8adaad444674f512133ad80b9220e09158521614f1faadfe85
05ef57b7df6813048603f0dd04f4280177a11380fbfc861dbcbd7418d62155248dad
5fdec0991f
blinded_msg = 10c166c6a711e81c46f45b18e5873cc4f494f003180dd7f115585d
871a28930259654fe28a54dab319cc5011204c8373b50a57b0fdc7a678bd74c52325
9dfe4fd5ea9f52f170e19dfa332930ad1609fc8a00902d725cfe50685c95e5b2968c
9a2828a21207fcf393d15f849769e2af34ac4259d91dfd98c3a707c509e1af55647e
faa31290ddf48e0133b798562af5eabd327270ac2fb6c594734ce339a14ea4fe1b9a
2f81c0bc230ca523bda17ff42a377266bc2778a274c0ae5ec5a8cbbe364fcf0d2403
f7ee178d77ff28b67a20c7ceec009182dbcaa9bc99b51ebbf13b7d542be337172c64
74f2cd3561219fe0dfa3fb207cff89632091ab841cf38d8aa88af6891539f263adb8
```

eac6402c41b6ebd72984e43666e537f5f5fe27b2b5aa114957e9a580730308a5f5a9
c63a1eb599f093ab401d0c6003a451931b6d124180305705845060ebba6b0036154f
cef3e5e9f9e4b87e8f084542fd1dd67e7782a5585150181c01eb6d90cb9588383738
4a5b91dbb606f266059ecc51b5acbaa280e45cfd2eec8cc1cdb1b7211c8e14805ba6
83f9b78824b2eb005bc8a7d7179a36c152cb87c8219e5569bba911bb32a1b923ca83
de0e03fb10fba75d85c55907dda5a2606bf918b056c3808ba496a4d95532212040a5
f44f37e1097f26dc27b98a51837daa78f23e532156296b64352669c94a8a855acf30
533d8e0594ace7c442
blind_sig = 364f6a40dbfbc3bbb257943337eeff791a0f290898a6791283bba581
d9eac90a6376a837241f5f73a78a5c6746e1306ba3adab6067c32ff69115734ce014
d354e2f259d4cbfb890244fd451a497fe6ecf9aa90d19a2d441162f7eaa7ce3fc4e8
9fd4e76b7ae585be2a2c0fd6fb246b8ac8d58bcb585634e30c9168a434786fe5e0b7
4bfe8187b47ac091aa571ffea0a864cb906d0e28c77a00e8cd8f6aba4317a8cc7bf3
2ce566bd1ef80c64de041728abe087bee6cadd0b7062bde5ceef308a23bd1ccc154f
d0c3a26110df6193464fc0d24ee189aea8979d722170ba945fdcce9b1b4b63349980
f3a92dc2e5418c54d38a862916926b3f9ca270a8cf40dfb9772bfbdd9a3e0e089236
9c18249211ba857f35963d0e05d8da98f1aa0c6bba58f47487b8f663e395091275f8
2941830b050b260e4767ce2fa903e75ff8970c98bfb3a08d6db91ab1746c86420ee2
e909bf681cac173697135983c3594b2def673736220452fde4ddec867d40ff42dd3d
a36c84e3e52508b891a00f50b4f62d112edb3b6b6cc3dbd546ba10f36b03f06c0d82
aeec3b25e127af545fac28e1613a0517a6095ad18a98ab79f68801e05c175e15bae2
1f821e80c80ab4fdec6fb34ca315e194502b8f3dcf7892b511aee45060e3994cd15e
003861bc7220a2babd7b40eda03382548a34a7110f9b1779bf3ef6011361611e6bc5
c0dc851e1509de1a
sig = 6fef8bf9bc182cd8cf7ce45c7dcf0e6f3e518ae48f06f3c670c649ac737a8b
8119a34d51641785be151a697ed7825fdfece82865123445eab03eb4bb91cecf4d69
51738495f8481151b62de869658573df4e50a95c17c31b52e154ae26a04067d5ecdc
1592c287550bb982a5bb9c30fd53a768cee6baabb3d483e9f1e2da954c7f4cf492fe
3944d2fe456c1ecaf0840369e33fb4010e6b44bb1d721840513524d8e9a3519f40d1
b81ae34fb7a31ee6b7ed641cb16c2ac999004c2191de0201457523f5a4700dd64926
7d9286f5c1d193f1454c9f868a57816bf5ff76c838a2eeb616a3fc9976f65d4371de
ecfbab29362caebdff69c635fe5a2113da4d4d8c24f0b16a0584fa05e80e607c5d9a
2f765f1f069f8d4da21f27c2a3b5c984b4ab24899bef46c6d9323df4862fe51ce300
fca40fb539c3bb7fe2dcc9409e425f2d3b95e70e9c49c5feb6ecc9d43442c33d5000
3ee936845892fb8be475647da9a080f5bc7f8a716590b3745c2209fe05b17992830c
e15f32c7b22cde755c8a2fe50bd814a0434130b807dc1b7218d4e85342d70695a5d7
f29306f25623ad1e8aa08ef71b54b8ee447b5f64e73d09bdd6c3b7ca224058d7c67c
c7551e9241688ada12d859cb7646fbd3ed8b34312f3b49d69802f0eaa11bc4211c2f
7a29cd5c01ed01a39001c5856fab36228f5ee2f2e1110811872fe7c865c42ed59029
c706195d52

A.2. RSABSSA-SHA384-PSSZERO-Randomized Test Vector

```
p = 0xe1f4d7a34802e27c7392a3cea32a262a34dc3691bd87f3f310dc7567348893
0559c120fd0410194fb8a0da55bd0b81227e843fdca6692ae80e5a5d414116d4803f
ca7d8c30eaaae57e44a1816ebb5c5b0606c536246c7f11985d731684150b63c9a3ad
9e41b04c0b5b27cb188a692c84696b742a80d3cd00ab891f2457443dadfeba6d6daf
108602be26d7071803c67105a5426838e6889d77e8474b29244cefaf418e381b3120
48b457d73419213063c60ee7b0d81820165864fef93523c9635c22210956e53a8d96
322493ffc58d845368e2416e078e5bcb5d2fd68ae6acfa54f9627c42e84a9d3f2774
017e32ebca06308a12ecc290c7cd1156dcccfb2311
q = 0xc601a9caea66dc3835827b539db9df6f6f5ae77244692780cd334a006ab353
c806426b60718c05245650821d39445d3ab591ed10a7339f15d83fe13f6a3dfb20b9
452c6a9b42eaa62a68c970df3cadb2139f804ad8223d56108dfde30ba7d367e9b0a7
a80c4fdba2fd9dde6661fc73fc2947569d2029f2870fc02d8325acf28c9afa19ecf9
62daa7916e21afad09eb62fe9f1cf91b77dc879b7974b490d3ebd2e95426057f35d0
a3c9f45f79ac727ab81a519a8b9285932d9b2e5ccd347e59f3f32ad9ca359115e7da
008ab7406707bd0e8e185a5ed8758b5ba266e8828f8d863ae133846304a2936ad7bc
7c9803879d2fc4a28e69291d73dbd799f8bc238385
n = 0xaec4d69addc70b990ea66a5e70603b6fee27aafebd08f2d94cbe1250c556e0
47a928d635c3f45ee9b66d1bc628a03bac9b7c3f416fe20dabea8f3d7b4bbf7f963b
e335d2328d67e6c13ee4a8f955e05a3283720d3e1f139c38e43e0338ad058a9495c5
3377fc35be64d208f89b4aa721bf7f7d3fef837be2a80e0f8adf0bcd1eec5bb04044
3a2b2792fdca522a7472aed74f31a1ebe1eebc1f408660a0543dfe2a850f106a617e
c6685573702eaaa21a5640a5dcaf9b74e397fa3af18a2f1b7c03ba91a6336158de42
0d63188ee143866ee415735d155b7c2d854d795b7bc236cffd71542df34234221a04
13e142d8c61355cc44d45bda94204974557ac2704cd8b593f035a5724b1adf442e78
c542cd4414fce6f1298182fb6d8e53cef1adfd2e90e1e4deec52999bdc6c29144e8d
52a125232c8c6d75c706ea3cc06841c7bda33568c63a6c03817f722b50fcf898237d
788a4400869e44d90a3020923dc646388abcc914315215fcd1bae11b1c751fd52443
aac8f601087d8d42737c18a3fa11ecd4131ecae017ae0a14acfc4ef85b83c19fed33
cfd1cd629da2c4c09e222b398e18d822f77bb378dea3cb360b605e5aa58b20edc29d
000a66bd177c682a17e7eb12a63ef7c2e4183e0d898f3d6bf567ba8ae84f84f1d23b
f8b8e261c3729e2fa6d07b832e07cddd1d14f55325c6f924267957121902dc19b3b3
2948bdead5
e = 0x10001
d = 0xd43242aefe1fb2c13fbc66e20b678c4336d20b1808c558b6e62ad16a287077
180b177e1f01b12f9c6cd6c52630257ccef26a45135a990928773f3bd2fc01a313f1
dac97a51cec71cb1fd7efc7adffdeb05f1fb04812c924ed7f4a8269925dad88bd7dc
fbc4ef01020ebfc60cb3e04c54f981fdbd273e69a8a58b8ceb7c2d83fbcbd6f784d0
52201b88a9848186f2a45c0d2826870733e6fd9aa46983e0a6e82e35ca20a439c5ee
7b502a9062e1066493bdadf8b49eb30d9558ed85abc7afb29b3c9bc644199654a467
6681af4babcea4e6f71fe4565c9c1b85d9985b84ec1abf1a820a9bbebee0df1398aa
e2c85ab580a9f13e7743afd3108eb32100b870648fa6bc17e8abac4d3c99246b1f0e
a9f7f93a5dd5458c56d9f3f81ff2216b3c3680a13591673c43194d8e6fc93fc1e37c
e2986bd628ac48088bc723d8fbe293861ca7a9f4a73e9fa63b1b6d0074f5dea2a624
c5249ff3ad811b6255b299d6bc5451ba7477f19c5a0db690c3e6476398b1483d1031
4afd38bbaf6e2fbdbcd62c3ca9797a420ca6034ec0a83360a3ee2adf4b9d4ba29731
d131b099a38d6a23cc463db754603211260e99d19affc902c915d7854554aabf608e
3ac52c19b8aa26ae042249b17b2d29669b5c859103ee53ef9bdc73ba3c6b537d5c34
b6d8f034671d7f3a8a6966cc4543df2235653431541 40fd7391c7e7be03e241f4ecf
eb877a051
```

msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c80023a
a6b59f8cfec5fdbb36331372ebefedae7d
msg_prefix = 574ee33ed775752cc7cebcdd39368aba6acff4f6d5b1c8f997edc39
795d34472
random_msg = 574ee33ed775752cc7cebcdd39368aba6acff4f6d5b1c8f997edc39
795d344728f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c800
23aa6b59f8cfec5fdbb36331372ebefedae7d
salt = 051722b35f458781397c3a671a7d3bd3096503940e4c4f1aaa269d60300ce
449555cd7340100df9d46944c5356825abf
encoded_msg = 6e0c464d9c2f9fbc147b43570fc4f238e0d0b38870b3addcf7a421
7df912ccef17a7f629aa850f63a063925f312d61d6437be954b45025e8282f9c0b11
31bc8ff19a8a928d859b37113db1064f92a27f64761c181c1e1f9b251ae5a2f8a404
7573b67a270584e089beadcb13e7c82337797119712e9b849ff56e04385d144d3ca9
d8d92bf78adb20b5bbeb3685f17038ec6afade3ef354429c51c687b45a7018ee3a69
66b3af15c9ba8f40e6461ba0a17ef5a799672ad882bab02b518f9da7c1a962945c2e
9b0f02f29b31b9cdf3e633f9d9d2a22e96e1de28e25241ca7dd04147112f57897340
3e0f4fd80865965475d22294f065e17a1c4a201de93bd14223e6b1b999fd548f2f75
9f52db71964528b6f15b9c2d7811f2a0a35d534b8216301c47f4f04f412cae142b48
c4cdff78bc54df690fd43142d750c671dd8e2e938e6a440b2f825b6dbb3e19f1d7a3
c0150428a47948037c322365b7fe6fe57ac88d8f80889e9ff38177bad8c8d8d98db4
2908b389cb59692a58ce275aa15acb032ca951b3e0a3404b7f33f655b7c7d83a2f8d
1b6bbff49d5fcedf2e030e80881aa436db27a5c0dea13f32e7d460dbf01240c2320c
2bb5b3225b17145c72d61d47c8f84d1e19417ebd8ce3638a82d395cc6f7050b6209d
9283dc7b93fecc04f3f9e7f566829ac41568ef799480c733c09759aa9734e2013d76
40dc6151018ea902bc
inv = 80682c48982407b489d53d1261b19ec8627d02b8cda5336750b8cee332ae26
0de57b02d72609c1e0e9f28e2040fc65b6f02d56dbd6aa9af8fde656f70495dfb723
ba01173d4707a12fddac628ca29f3e32340bd8f7ddb557cf819f6b01e445ad96f874
ba235584ee71f6581f62d4f43bf03f910f6510deb85e8ef06c7f09d9794a008be7ff
2529f0ebb69decef646387dc767b74939265fec0223aa6d84d2a8a1cc912d5ca25b4
e144ab8f6ba054b54910176d5737a2cff011da431bd5f2a0d2d66b9e70b39f4b050e
45c0d9c16f02deda9ddf2d00f3e4b01037d7029cd49c2d46a8e1fc2c0c17520af1f4
b5e25ba396afc4cd60c494a4c426448b35b49635b337cfb08e7c22a39b256dd032c0
0adddafb51a627f99a0e1704170ac1f1912e49d9db10ec04c19c58f420212973e0cb
329524223a6aa56c7937c5dffdb5d966b6cd4cbc26f3201dd25c80960a1a111b3294
7bb78973d269fac7f5186530930ed19f68507540eed9e1bab8b00f00d8ca09b3f099
aae46180e04e3584bd7ca054df18a1504b89d1d1675d0966c4ae1407be325cdf623c
f13ff13e4a28b594d59e3eadbadf6136eee7a59d6a444c9eb4e2198e8a974f27a39e
b63af2c9af3870488b8adaad444674f512133ad80b9220e09158521614f1faadfe85
05ef57b7df6813048603f0dd04f4280177a11380fbfc861dbcbd7418d62155248dad
5fdec0991f
blinded_msg = 10c166c6a711e81c46f45b18e5873cc4f494f003180dd7f115585d
871a28930259654fe28a54dab319cc5011204c8373b50a57b0fdc7a678bd74c52325
9dfe4fd5ea9f52f170e19dfa332930ad1609fc8a00902d725cfe50685c95e5b2968c
9a2828a21207fcf393d15f849769e2af34ac4259d91dfd98c3a707c509e1af55647e
faa31290ddf48e0133b798562af5eabd327270ac2fb6c594734ce339a14ea4fe1b9a
2f81c0bc230ca523bda17ff42a377266bc2778a274c0ae5ec5a8cbbe364fcf0d2403
f7ee178d77ff28b67a20c7ceec009182dbcaa9bc99b51ebbf13b7d542be337172c64
74f2cd3561219fe0dfa3fb207cff89632091ab841cf38d8aa88af6891539f263adb8

eac6402c41b6ebd72984e43666e537f5f5fe27b2b5aa114957e9a580730308a5f5a9
c63a1eb599f093ab401d0c6003a451931b6d124180305705845060ebba6b0036154f
cef3e5e9f9e4b87e8f084542fd1dd67e7782a5585150181c01eb6d90cb9588383738
4a5b91dbb606f266059ecc51b5acbaa280e45cfd2eec8cc1cdb1b7211c8e14805ba6
83f9b78824b2eb005bc8a7d7179a36c152cb87c8219e5569bba911bb32a1b923ca83
de0e03fb10fba75d85c55907dda5a2606bf918b056c3808ba496a4d95532212040a5
f44f37e1097f26dc27b98a51837daa78f23e532156296b64352669c94a8a855acf30
533d8e0594ace7c442
blind_sig = 364f6a40dbfbc3bbb257943337eeff791a0f290898a6791283bba581
d9eac90a6376a837241f5f73a78a5c6746e1306ba3adab6067c32ff69115734ce014
d354e2f259d4cbfb890244fd451a497fe6ecf9aa90d19a2d441162f7eaa7ce3fc4e8
9fd4e76b7ae585be2a2c0fd6fb246b8ac8d58bcb585634e30c9168a434786fe5e0b7
4bfe8187b47ac091aa571ffea0a864cb906d0e28c77a00e8cd8f6aba4317a8cc7bf3
2ce566bd1ef80c64de041728abe087bee6cadd0b7062bde5ceef308a23bd1ccc154f
d0c3a26110df6193464fc0d24ee189aea8979d722170ba945fdcce9b1b4b63349980
f3a92dc2e5418c54d38a862916926b3f9ca270a8cf40dfb9772bfbdd9a3e0e089236
9c18249211ba857f35963d0e05d8da98f1aa0c6bba58f47487b8f663e395091275f8
2941830b050b260e4767ce2fa903e75ff8970c98bfb3a08d6db91ab1746c86420ee2
e909bf681cac173697135983c3594b2def673736220452fde4ddec867d40ff42dd3d
a36c84e3e52508b891a00f50b4f62d112edb3b6b6cc3dbd546ba10f36b03f06c0d82
aeec3b25e127af545fac28e1613a0517a6095ad18a98ab79f68801e05c175e15bae2
1f821e80c80ab4fdec6fb34ca315e194502b8f3dcf7892b511aee45060e3994cd15e
003861bc7220a2babd7b40eda03382548a34a7110f9b1779bf3ef6011361611e6bc5
c0dc851e1509de1a
sig = 6fef8bf9bc182cd8cf7ce45c7dcf0e6f3e518ae48f06f3c670c649ac737a8b
8119a34d51641785be151a697ed7825fdfece82865123445eab03eb4bb91cecf4d69
51738495f8481151b62de869658573df4e50a95c17c31b52e154ae26a04067d5ecdc
1592c287550bb982a5bb9c30fd53a768cee6baabb3d483e9f1e2da954c7f4cf492fe
3944d2fe456c1ecaf0840369e33fb4010e6b44bb1d721840513524d8e9a3519f40d1
b81ae34fb7a31ee6b7ed641cb16c2ac999004c2191de0201457523f5a4700dd64926
7d9286f5c1d193f1454c9f868a57816bf5ff76c838a2eeb616a3fc9976f65d4371de
ecfbab29362caebdff69c635fe5a2113da4d4d8c24f0b16a0584fa05e80e607c5d9a
2f765f1f069f8d4da21f27c2a3b5c984b4ab24899bef46c6d9323df4862fe51ce300
fca40fb539c3bb7fe2dcc9409e425f2d3b95e70e9c49c5feb6ecc9d43442c33d5000
3ee936845892fb8be475647da9a080f5bc7f8a716590b3745c2209fe05b17992830c
e15f32c7b22cde755c8a2fe50bd814a0434130b807dc1b7218d4e85342d70695a5d7
f29306f25623ad1e8aa08ef71b54b8ee447b5f64e73d09bdd6c3b7ca224058d7c67c
c7551e9241688ada12d859cb7646fbd3ed8b34312f3b49d69802f0eaa11bc4211c2f
7a29cd5c01ed01a39001c5856fab36228f5ee2f2e1110811872fe7c865c42ed59029
c706195d52

A.3. RSABSSA-SHA384-PSS-Deterministic Test Vector

```
p = 0xe1f4d7a34802e27c7392a3cea32a262a34dc3691bd87f3f310dc7567348893
0559c120fd0410194fb8a0da55bd0b81227e843fdca6692ae80e5a5d414116d4803f
ca7d8c30eaaae57e44a1816ebb5c5b0606c536246c7f11985d731684150b63c9a3ad
9e41b04c0b5b27cb188a692c84696b742a80d3cd00ab891f2457443dadfeba6d6daf
108602be26d7071803c67105a5426838e6889d77e8474b29244cefaf418e381b3120
48b457d73419213063c60ee7b0d81820165864fef93523c9635c22210956e53a8d96
322493ffc58d845368e2416e078e5bcb5d2fd68ae6acfa54f9627c42e84a9d3f2774
017e32ebca06308a12ecc290c7cd1156dcccfb2311
q = 0xc601a9caea66dc3835827b539db9df6f6f5ae77244692780cd334a006ab353
c806426b60718c05245650821d39445d3ab591ed10a7339f15d83fe13f6a3dfb20b9
452c6a9b42eaa62a68c970df3cadb2139f804ad8223d56108dfde30ba7d367e9b0a7
a80c4fdba2fd9dde6661fc73fc2947569d2029f2870fc02d8325acf28c9afa19ecf9
62daa7916e21afad09eb62fe9f1cf91b77dc879b7974b490d3ebd2e95426057f35d0
a3c9f45f79ac727ab81a519a8b9285932d9b2e5ccd347e59f3f32ad9ca359115e7da
008ab7406707bd0e8e185a5ed8758b5ba266e8828f8d863ae133846304a2936ad7bc
7c9803879d2fc4a28e69291d73dbd799f8bc238385
n = 0xaec4d69addc70b990ea66a5e70603b6fee27aafebd08f2d94cbe1250c556e0
47a928d635c3f45ee9b66d1bc628a03bac9b7c3f416fe20dabea8f3d7b4bbf7f963b
e335d2328d67e6c13ee4a8f955e05a3283720d3e1f139c38e43e0338ad058a9495c5
3377fc35be64d208f89b4aa721bf7f7d3fef837be2a80e0f8adf0bcd1eec5bb04044
3a2b2792fdca522a7472aed74f31a1ebe1eebc1f408660a0543dfe2a850f106a617e
c6685573702eaaa21a5640a5dcaf9b74e397fa3af18a2f1b7c03ba91a6336158de42
0d63188ee143866ee415735d155b7c2d854d795b7bc236cffd71542df34234221a04
13e142d8c61355cc44d45bda94204974557ac2704cd8b593f035a5724b1adf442e78
c542cd4414fce6f1298182fb6d8e53cef1adfd2e90e1e4deec52999bdc6c29144e8d
52a125232c8c6d75c706ea3cc06841c7bda33568c63a6c03817f722b50fcf898237d
788a4400869e44d90a3020923dc646388abcc914315215fcd1bae11b1c751fd52443
aac8f601087d8d42737c18a3fa11ecd4131ecae017ae0a14acfc4ef85b83c19fed33
cfd1cd629da2c4c09e222b398e18d822f77bb378dea3cb360b605e5aa58b20edc29d
000a66bd177c682a17e7eb12a63ef7c2e4183e0d898f3d6bf567ba8ae84f84f1d23b
f8b8e261c3729e2fa6d07b832e07cddd1d14f55325c6f924267957121902dc19b3b3
2948bdead5
e = 0x10001
d = 0xd43242aefe1fb2c13fbc66e20b678c4336d20b1808c558b6e62ad16a287077
180b177e1f01b12f9c6cd6c52630257ccef26a45135a990928773f3bd2fc01a313f1
dac97a51cec71cb1fd7efc7adffdeb05f1fb04812c924ed7f4a8269925dad88bd7dc
fbc4ef01020ebfc60cb3e04c54f981fdbd273e69a8a58b8ceb7c2d83fbcbd6f784d0
52201b88a9848186f2a45c0d2826870733e6fd9aa46983e0a6e82e35ca20a439c5ee
7b502a9062e1066493bdadf8b49eb30d9558ed85abc7afb29b3c9bc644199654a467
6681af4babcea4e6f71fe4565c9c1b85d9985b84ec1abf1a820a9bbebee0df1398aa
e2c85ab580a9f13e7743afd3108eb32100b870648fa6bc17e8abac4d3c99246b1f0e
a9f7f93a5dd5458c56d9f3f81ff2216b3c3680a13591673c43194d8e6fc93fc1e37c
e2986bd628ac48088bc723d8fbe293861ca7a9f4a73e9fa63b1b6d0074f5dea2a624
c5249ff3ad811b6255b299d6bc5451ba7477f19c5a0db690c3e6476398b1483d1031
4afd38bbaf6e2fbdbcd62c3ca9797a420ca6034ec0a83360a3ee2adf4b9d4ba29731
d131b099a38d6a23cc463db754603211260e99d19affc902c915d7854554aabf608e
3ac52c19b8aa26ae042249b17b2d29669b5c859103ee53ef9bdc73ba3c6b537d5c34
b6d8f034671d7f3a8a6966cc4543df2235653431541 40fd7391c7e7be03e241f4ecf
eb877a051
```

```
msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c80023a
a6b59f8cfec5fdbb36331372ebefedae7d
random_msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698
c80023aa6b59f8cfec5fdbb36331372ebefedae7d
salt =
encoded_msg = 159499b90471b496c2639ec482e99feaba525c0420c565d17dc60c
1bb1f47703f04436cceaa8f69811e1bf8546fa971226c9e71421b32b571ed5ea0e03
2269d4219b4404316eb17a58f277634aeed394b7f3888153b5bb163e40807e605daf
dd1789dd473b0846bdcb6524417bc3a35366fab4261708c0e4b4beba07a1a64bbccb
4b1ac215d1350a50a501e8e96612028b535ad731abf1f117ee07d07a4de9cef3d70f
5845ba84c29d5d92c6e66a1f9489a5f527b846825360fd6e90f40ed041c682e489f3
acde984a3ea580181418c1d15017af2657bc4b70485cdc0f1ebc3693e0d70a5d01f3
7ff640993fa071274fb9ee44e0c24dcb58ffa21a9a6540d87f24379beaafcc3b4bd4
2c45ec6820e03738ce98bea11c71685f31db63429fab8658bdb816f1ecccb1888f24
02de0bd2f0f9646decdcad4c11b41428eec1ed25f2a86d43bb04f95726bfbd98ea34
ca091b7adbabd0e28f17fa0345b89542d23c3530554987508a23641bd4f9e52962b0
bee3ac9ffe005322d26a39941c5847774300411c69635f96903e8d593530908bd92a
4fa6a2d52f88073a647a4b3894b7e4ebb80699e60227397bfa93f41b1c97e107b632
f68e70409372ead2f072c11cf99be4486fcbf763dde28ee156db26cd358a69fcb796
44f1f2fcc166f41a4c80f5851ee08be051f14b601418d6e56e61733b9b210c6bef17
edac121a754d19b9bc
inv = 80682c48982407b489d53d1261b19ec8627d02b8cda5336750b8cee332ae26
0de57b02d72609c1e0e9f28e2040fc65b6f02d56dbd6aa9af8fde656f70495dfb723
ba01173d4707a12fddac628ca29f3e32340bd8f7ddb557cf819f6b01e445ad96f874
ba235584ee71f6581f62d4f43bf03f910f6510deb85e8ef06c7f09d9794a008be7ff
2529f0ebb69decef646387dc767b74939265fec0223aa6d84d2a8a1cc912d5ca25b4
e144ab8f6ba054b54910176d5737a2cff011da431bd5f2a0d2d66b9e70b39f4b050e
45c0d9c16f02deda9ddf2d00f3e4b01037d7029cd49c2d46a8e1fc2c0c17520af1f4
b5e25ba396afc4cd60c494a4c426448b35b49635b337cfb08e7c22a39b256dd032c0
0adddafb51a627f99a0e1704170ac1f1912e49d9db10ec04c19c58f420212973e0cb
329524223a6aa56c7937c5dffdb5d966b6cd4cbc26f3201dd25c80960a1a111b3294
7bb78973d269fac7f5186530930ed19f68507540eed9e1bab8b00f00d8ca09b3f099
aae46180e04e3584bd7ca054df18a1504b89d1d1675d0966c4ae1407be325cdf623c
f13ff13e4a28b594d59e3eadbadf6136eee7a59d6a444c9eb4e2198e8a974f27a39e
b63af2c9af3870488b8adaad444674f512133ad80b9220e09158521614f1faadfe85
05ef57b7df6813048603f0dd04f4280177a11380fbfc861dbcbd7418d62155248dad
5fdec0991f
blinded_msg = 982790826556aabe6004467671a864397eea3b95740e9a11c8b80b
99ee0cf4dbc50af860bda81b601a2eceaa6943ef104f13325ad0be2e37f42030b312
0e87cfee8cfe59cde1acfb25485a43275ebe777292e2518181ae531e596f988ff16f
458daa5a42408939cbe60e7271391a21657276427d195bee6a20054101d4ceb892ec
dea402ea1a866acf0e451a3336f07e7589330d96c3883fd5bc1a829a715b618b74a8
6b2a898764246ad081d4c9f1edb8ab5077e315fde2417ec2dd33cad93e120340b49b
e89c18a63e62c6bb289037283d3bf18608be11ee4c823c710b0c6b89235fed3f03a7
b96ddd25a8f54f20dac37ce8905093ad8e066810f354fb1773236e3d3788ba755de2
c9bce8d340078bb1831ddc7314a5018673427ced65cb356281aae08b5e6636f3eb24
17e09d6ae476a9abcc410bc8c90813d0740e39ae75efae4c02eed49dbb7aa51258bb
71197445d17a6029bf566ba6b36282173af2c42e9b9631366f22eb6a19ef1d92bd3c
e0631d3a7fb3288195b0ba380a3828d5411cefd5eba83e52198c001ac9946a333a33
```

d89d4d235fc833239d59837f04eaf065e9563659b00c7624a6263b727d8f2c07959b
a2bb592e7ff251b8f09c85995fd2e4474e743586576b518230986b6076b762ae7708
8a37e4bffd2ef41ae68d6d4e79205290b4f76c42ef039638c41cdc6fe8af9b429c0d
ee45b2942e3861da2a
blind_sig = 362ef369f9b8c1487e285514702a7cd6fe03e4a2fb854881f3d3f986
b7742a0c9bfab6562a6cd5ed71c574af67d7e77e71b33420c08ebb0ff37886b85829
7f9562fc366066c6d8e77bad1918b04756ba03f5c385d44f06759daf1b7a38b2a642
48dee95d0e3886c8afa1f74afd8ac3c56520d0f3fd206df8e0d257312756803b09a7
9d0cc38112592c3aec32de5a9bc3284c5a0a2d0808b102deafa5cc60f04e3d71c028
4cba04f17f88aa8e07d5544fe0265807d515877f79d30ed26d522b9d9c56597647b0
dbca5a69d6418f8d1b51481723f272c2a3d48f6f4fd6beeac3576c3edb00e8779964
548aeab8e004c7c4f8ef9cb6e680e2d2d49792004bb3e6974fa48f241a361ca449c0
2bd4c0ad4e66252c55e656f16049908efe59acbafa1171895dfac64d909808e54204
69d622c7253ec1de7522b41634d383bf8786bf881cbf1561627f1e62b2d93300ec30
ec0f5f0ab32036fce068bc76b0b0c6452079537f8d7f8dcee4b42bbf2d9ad7499d38
35cd93cfc7e8ebea3554ab5241e181e5d73241b7bebf0a281b63594a35f4993e2b41
6d60db966b58b648cfcba2c4bee4c2830aae4a70ff55012480298f549c13b1b26842
77bca12f592471b8a99285174f1c0ebb38fc80e74a10b3f02ec3e6682ba873f7ff0e
1e79718b470927c74ed754d4f7c3d9a55e22246e829cdb5a1c6fb2a0a6c896df3030
63c918bcf5eb0017
sig = 4454b6983ff01cb28545329f394936efa42ed231e15efbc025fdaca00277ac
f0c8e00e3d8b0ecebd35b057b8ebfc14e1a7097368a4abd20b555894ccef3d1b9528
c6bcbda6b95376bef230d0f1feff0c1064c62c60a7ae7431d1fdfa43a81eed9235e3
63e1ffa0b2797aba6aad6082fcd285e14fc8b71de6b9c87cb4059c7dc1e96ae1e637
95a1e9af86b9073d1d848aef3eca8a03421bcd116572456b53bcfd4dabb0a9691f1f
abda3ed0ce357aee2cfee5b1a0eb226f69716d4e011d96eede5e38a9acb531a64336
a0d5b0bae3ab085b658692579a376740ff6ce69e89b06f360520b864e33d82d029c8
08248a19e18e31f0ecd16fac5cd4870f8d3ebc1c32c718124152dc905672ab0b7af4
8bf7d1ac1ff7b9c742549c91275ab105458ae37621757add83482bbcf779e777bbd6
1126e93686635d4766aedf5103cf7978f3856ccac9e28d21a850dbb03c811128616d
315d717be1c2b6254f8509acae862042c034530329ce15ca2e2f6b1f5fd59272746e
3918c748c0eb810bf76884fa10fcf749326bbfaa5ba285a0186a22e4f628dbf178d3
bb5dc7e165ca73f6a55ecc14c4f5a26c4693ce5da032264cbec319b12ddb9787d0ef
a4fcf1e5ccee35ad85ecd453182df9ed735893f830b570faae8be0f6fe2e571a4e0d
927cba4debd368d3b4fca33ec6251897a137cf75474a32ac8256df5e5ffa518b88b4
3fb6f63a24

A.4. RSABSSA-SHA384-PSSZERO-Deterministic Test Vector

```
p = 0xe1f4d7a34802e27c7392a3cea32a262a34dc3691bd87f3f310dc7567348893
0559c120fd0410194fb8a0da55bd0b81227e843fdca6692ae80e5a5d414116d4803f
ca7d8c30eaaae57e44a1816ebb5c5b0606c536246c7f11985d731684150b63c9a3ad
9e41b04c0b5b27cb188a692c84696b742a80d3cd00ab891f2457443dadfeba6d6daf
108602be26d7071803c67105a5426838e6889d77e8474b29244cefaf418e381b3120
48b457d73419213063c60ee7b0d81820165864fef93523c9635c22210956e53a8d96
322493ffc58d845368e2416e078e5bcb5d2fd68ae6acfa54f9627c42e84a9d3f2774
017e32ebca06308a12ecc290c7cd1156dcccfb2311
q = 0xc601a9caea66dc3835827b539db9df6f6f5ae77244692780cd334a006ab353
c806426b60718c05245650821d39445d3ab591ed10a7339f15d83fe13f6a3dfb20b9
452c6a9b42eaa62a68c970df3cadb2139f804ad8223d56108dfde30ba7d367e9b0a7
a80c4fdba2fd9dde6661fc73fc2947569d2029f2870fc02d8325acf28c9afa19ecf9
62daa7916e21afad09eb62fe9f1cf91b77dc879b7974b490d3ebd2e95426057f35d0
a3c9f45f79ac727ab81a519a8b9285932d9b2e5ccd347e59f3f32ad9ca359115e7da
008ab7406707bd0e8e185a5ed8758b5ba266e8828f8d863ae133846304a2936ad7bc
7c9803879d2fc4a28e69291d73dbd799f8bc238385
n = 0xaec4d69addc70b990ea66a5e70603b6fee27aafebd08f2d94cbe1250c556e0
47a928d635c3f45ee9b66d1bc628a03bac9b7c3f416fe20dabea8f3d7b4bbf7f963b
e335d2328d67e6c13ee4a8f955e05a3283720d3e1f139c38e43e0338ad058a9495c5
3377fc35be64d208f89b4aa721bf7f7d3fef837be2a80e0f8adf0bcd1eec5bb04044
3a2b2792fdca522a7472aed74f31a1ebe1eebc1f408660a0543dfe2a850f106a617e
c6685573702eaaa21a5640a5dcaf9b74e397fa3af18a2f1b7c03ba91a6336158de42
0d63188ee143866ee415735d155b7c2d854d795b7bc236cffd71542df34234221a04
13e142d8c61355cc44d45bda94204974557ac2704cd8b593f035a5724b1adf442e78
c542cd4414fce6f1298182fb6d8e53cef1adfd2e90e1e4deec52999bdc6c29144e8d
52a125232c8c6d75c706ea3cc06841c7bda33568c63a6c03817f722b50fcf898237d
788a4400869e44d90a3020923dc646388abcc914315215fcd1bae11b1c751fd52443
aac8f601087d8d42737c18a3fa11ecd4131ecae017ae0a14acfc4ef85b83c19fed33
cfd1cd629da2c4c09e222b398e18d822f77bb378dea3cb360b605e5aa58b20edc29d
000a66bd177c682a17e7eb12a63ef7c2e4183e0d898f3d6bf567ba8ae84f84f1d23b
f8b8e261c3729e2fa6d07b832e07cddd1d14f55325c6f924267957121902dc19b3b3
2948bdead5
e = 0x10001
d = 0xd43242aefe1fb2c13fbc66e20b678c4336d20b1808c558b6e62ad16a287077
180b177e1f01b12f9c6cd6c52630257ccef26a45135a990928773f3bd2fc01a313f1
dac97a51cec71cb1fd7efc7adffdeb05f1fb04812c924ed7f4a8269925dad88bd7dc
fbc4ef01020ebfc60cb3e04c54f981fdbd273e69a8a58b8ceb7c2d83fbcbd6f784d0
52201b88a9848186f2a45c0d2826870733e6fd9aa46983e0a6e82e35ca20a439c5ee
7b502a9062e1066493bdadf8b49eb30d9558ed85abc7afb29b3c9bc644199654a467
6681af4babcea4e6f71fe4565c9c1b85d9985b84ec1abf1a820a9bbebee0df1398aa
e2c85ab580a9f13e7743afd3108eb32100b870648fa6bc17e8abac4d3c99246b1f0e
a9f7f93a5dd5458c56d9f3f81ff2216b3c3680a13591673c43194d8e6fc93fc1e37c
e2986bd628ac48088bc723d8fbe293861ca7a9f4a73e9fa63b1b6d0074f5dea2a624
c5249ff3ad811b6255b299d6bc5451ba7477f19c5a0db690c3e6476398b1483d1031
4afd38bbaf6e2fbdbcd62c3ca9797a420ca6034ec0a83360a3ee2adf4b9d4ba29731
d131b099a38d6a23cc463db754603211260e99d19affc902c915d7854554aabf608e
3ac52c19b8aa26ae042249b17b2d29669b5c859103ee53ef9bdc73ba3c6b537d5c34
b6d8f034671d7f3a8a6966cc4543df2235653431541 40fd7391c7e7be03e241f4ecf
eb877a051
```

```
msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c80023a
a6b59f8cfec5fdbb36331372ebefedae7d
random_msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698
c80023aa6b59f8cfec5fdbb36331372ebefedae7d
salt =
encoded_msg = 159499b90471b496c2639ec482e99feaba525c0420c565d17dc60c
1bb1f47703f04436cceaa8f69811e1bf8546fa971226c9e71421b32b571ed5ea0e03
2269d4219b4404316eb17a58f277634aeed394b7f3888153b5bb163e40807e605daf
dd1789dd473b0846bdcb6524417bc3a35366fab4261708c0e4b4beba07a1a64bbccb
4b1ac215d1350a50a501e8e96612028b535ad731abf1f117ee07d07a4de9cef3d70f
5845ba84c29d5d92c6e66a1f9489a5f527b846825360fd6e90f40ed041c682e489f3
acde984a3ea580181418c1d15017af2657bc4b70485cdc0f1ebc3693e0d70a5d01f3
7ff640993fa071274fb9ee44e0c24dcb58ffa21a9a6540d87f24379beaafcc3b4bd4
2c45ec6820e03738ce98bea11c71685f31db63429fab8658bdb816f1ecccb1888f24
02de0bd2f0f9646decdcad4c11b41428eec1ed25f2a86d43bb04f95726bfbd98ea34
ca091b7adbabd0e28f17fa0345b89542d23c3530554987508a23641bd4f9e52962b0
bee3ac9ffe005322d26a39941c5847774300411c69635f96903e8d593530908bd92a
4fa6a2d52f88073a647a4b3894b7e4ebb80699e60227397bfa93f41b1c97e107b632
f68e70409372ead2f072c11cf99be4486fcbf763dde28ee156db26cd358a69fcb796
44f1f2fcc166f41a4c80f5851ee08be051f14b601418d6e56e61733b9b210c6bef17
edac121a754d19b9bc
inv = 80682c48982407b489d53d1261b19ec8627d02b8cda5336750b8cee332ae26
0de57b02d72609c1e0e9f28e2040fc65b6f02d56dbd6aa9af8fde656f70495dfb723
ba01173d4707a12fddac628ca29f3e32340bd8f7ddb557cf819f6b01e445ad96f874
ba235584ee71f6581f62d4f43bf03f910f6510deb85e8ef06c7f09d9794a008be7ff
2529f0ebb69decef646387dc767b74939265fec0223aa6d84d2a8a1cc912d5ca25b4
e144ab8f6ba054b54910176d5737a2cff011da431bd5f2a0d2d66b9e70b39f4b050e
45c0d9c16f02deda9ddf2d00f3e4b01037d7029cd49c2d46a8e1fc2c0c17520af1f4
b5e25ba396afc4cd60c494a4c426448b35b49635b337cfb08e7c22a39b256dd032c0
0adddafb51a627f99a0e1704170ac1f1912e49d9db10ec04c19c58f420212973e0cb
329524223a6aa56c7937c5dffdb5d966b6cd4cbc26f3201dd25c80960a1a111b3294
7bb78973d269fac7f5186530930ed19f68507540eed9e1bab8b00f00d8ca09b3f099
aae46180e04e3584bd7ca054df18a1504b89d1d1675d0966c4ae1407be325cdf623c
f13ff13e4a28b594d59e3eadbadf6136eee7a59d6a444c9eb4e2198e8a974f27a39e
b63af2c9af3870488b8adaad444674f512133ad80b9220e09158521614f1faadfe85
05ef57b7df6813048603f0dd04f4280177a11380fbfc861dbcbd7418d62155248dad
5fdec0991f
blinded_msg = 982790826556aabe6004467671a864397eea3b95740e9a11c8b80b
99ee0cf4dbc50af860bda81b601a2eceaa6943ef104f13325ad0be2e37f42030b312
0e87cfee8cfe59cde1acfb25485a43275ebe777292e2518181ae531e596f988ff16f
458daa5a42408939cbe60e7271391a21657276427d195bee6a20054101d4ceb892ec
dea402ea1a866acf0e451a3336f07e7589330d96c3883fd5bc1a829a715b618b74a8
6b2a898764246ad081d4c9f1edb8ab5077e315fde2417ec2dd33cad93e120340b49b
e89c18a63e62c6bb289037283d3bf18608be11ee4c823c710b0c6b89235fed3f03a7
b96ddd25a8f54f20dac37ce8905093ad8e066810f354fb1773236e3d3788ba755de2
c9bce8d340078bb1831ddc7314a5018673427ced65cb356281aae08b5e6636f3eb24
17e09d6ae476a9abcc410bc8c90813d0740e39ae75efae4c02eed49dbb7aa51258bb
71197445d17a6029bf566ba6b36282173af2c42e9b9631366f22eb6a19ef1d92bd3c
e0631d3a7fb3288195b0ba380a3828d5411cefd5eba83e52198c001ac9946a333a33
```

d89d4d235fc833239d59837f04eaf065e9563659b00c7624a6263b727d8f2c07959b
a2bb592e7ff251b8f09c85995fd2e4474e743586576b518230986b6076b762ae7708
8a37e4bffd2ef41ae68d6d4e79205290b4f76c42ef039638c41cdc6fe8af9b429c0d
ee45b2942e3861da2a
blind_sig = 362ef369f9b8c1487e285514702a7cd6fe03e4a2fb854881f3d3f986
b7742a0c9bfab6562a6cd5ed71c574af67d7e77e71b33420c08ebb0ff37886b85829
7f9562fc366066c6d8e77bad1918b04756ba03f5c385d44f06759daf1b7a38b2a642
48dee95d0e3886c8afa1f74afd8ac3c56520d0f3fd206df8e0d257312756803b09a7
9d0cc38112592c3aec32de5a9bc3284c5a0a2d0808b102deafa5cc60f04e3d71c028
4cba04f17f88aa8e07d5544fe0265807d515877f79d30ed26d522b9d9c56597647b0
dbca5a69d6418f8d1b51481723f272c2a3d48f6f4fd6beeac3576c3edb00e8779964
548aeab8e004c7c4f8ef9cb6e680e2d2d49792004bb3e6974fa48f241a361ca449c0
2bd4c0ad4e66252c55e656f16049908efe59acbafa1171895dfac64d909808e54204
69d622c7253ec1de7522b41634d383bf8786bf881cbf1561627f1e62b2d93300ec30
ec0f5f0ab32036fce068bc76b0b0c6452079537f8d7f8dcee4b42bbf2d9ad7499d38
35cd93cfc7e8ebea3554ab5241e181e5d73241b7bebf0a281b63594a35f4993e2b41
6d60db966b58b648cfcba2c4bee4c2830aae4a70ff55012480298f549c13b1b26842
77bca12f592471b8a99285174f1c0ebb38fc80e74a10b3f02ec3e6682ba873f7ff0e
1e79718b470927c74ed754d4f7c3d9a55e22246e829cdb5a1c6fb2a0a6c896df3030
63c918bcf5eb0017
sig = 4454b6983ff01cb28545329f394936efa42ed231e15efbc025fdaca00277ac
f0c8e00e3d8b0ecebd35b057b8ebfc14e1a7097368a4abd20b555894ccef3d1b9528
c6bcbda6b95376bef230d0f1feff0c1064c62c60a7ae7431d1fdfa43a81eed9235e3
63e1ffa0b2797aba6aad6082fcd285e14fc8b71de6b9c87cb4059c7dc1e96ae1e637
95a1e9af86b9073d1d848aef3eca8a03421bcd116572456b53bcfd4dabb0a9691f1f
abda3ed0ce357aee2cfee5b1a0eb226f69716d4e011d96eede5e38a9acb531a64336
a0d5b0bae3ab085b658692579a376740ff6ce69e89b06f360520b864e33d82d029c8
08248a19e18e31f0ecd16fac5cd4870f8d3ebc1c32c718124152dc905672ab0b7af4
8bf7d1ac1ff7b9c742549c91275ab105458ae37621757add83482bbcf779e777bbd6
1126e93686635d4766aedf5103cf7978f3856ccac9e28d21a850dbb03c811128616d
315d717be1c2b6254f8509acae862042c034530329ce15ca2e2f6b1f5fd59272746e
3918c748c0eb810bf76884fa10fcf749326bbfaa5ba285a0186a22e4f628dbf178d3
bb5dc7e165ca73f6a55ecc14c4f5a26c4693ce5da032264cbec319b12ddb9787d0ef
a4fcf1e5ccee35ad85ecd453182df9ed735893f830b570faae8be0f6fe2e571a4e0d
927cba4debd368d3b4fca33ec6251897a137cf75474a32ac8256df5e5ffa518b88b4
3fb6f63a24

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