Internet Research Task Force (IRTF)
Request for Comments: 9474
Category: Informational
ISSN: 2070-1721

F. Denis
Fastly Inc.
F. Jacobs
Apple Inc.
C. A. Wood
Cloudflare

October 2023

RSA Blind Signatures

Abstract

This document specifies an RSA-based blind signature protocol. RSA blind signatures were first introduced by Chaum for untraceable payments. A signature that is output from this protocol can be verified as an RSA-PSS signature.

This document is a product of the Crypto Forum Research Group (CFRG) in the IRTF.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Research Task Force (IRTF). The IRTF publishes the results of Internet-related research and development activities. These results might not be suitable for deployment. This RFC represents the consensus of the Crypto Forum Research Group of the Internet Research Task Force (IRTF). Documents approved for publication by the IRSG are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9474.

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 privacy-preserving digital payments to authentication mechanisms [GoogleVPN] [ApplePrivateRelay] [PrettyGoodPhonePrivacy].

Recently, interest in blind signatures has grown to address operational shortcomings from applications that use Verifiable Oblivious Pseudorandom Functions (VOPRFs) [VOPRF], such as Privacy Pass [PRIVACY-PASS]. 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 (message, signature) pair, where the signature is computed using a deterministic blind signature protocol.

This document specifies (1) a protocol for computing RSA blind signatures using RSA-PSS encoding and (2) a family of variants (Section 5) for this protocol, denoted RSABSSA (RSA Blind Signature with Appendix). 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.

This document represents the consensus of the Crypto Forum Research Group (CFRG). It is not an IETF product and is not a standard.

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, which describe different protocol operations, are used throughout 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 between M inclusive and N exclusive, i.e., M <= R < N.

bit_len(n):
: Compute the minimum number of bits needed to represent the positive integer n.

inverse_mod(x, n):
: Compute the multiplicative inverse of x mod n or fail if x and n are not co-prime.

is_coprime(x, n):
: Return true if x and n are co-prime, and false otherwise.

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

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

concat(x0, ..., xN):
: Concatenation of byte strings. For example, concat(0x01, 0x0203, 0x040506) = 0x010203040506.

slice(x, i, j):
   Return bytes in the byte string x starting from offset i and
   ending at offset j, inclusive. For example, slice(0x010203040506,
   1, 5) = 0x0203040506.

4. Blind Signature Protocol

The RSA Blind Signature Protocol is a two-party protocol between a client and server where they interact to compute sig = Sign(sk, input_msg), where input_msg = Prepare(msg) is a prepared version of the private message msg provided by the client, and sk is the private signing key provided by the server. See Section 6.2 for details on how sk is generated and used in this protocol. Upon completion of this protocol, the server learns nothing, whereas the client learns sig. In particular, this means the server learns nothing of msg or input_msg and the client learns nothing of sk.

The protocol consists of four functions -- Prepare, Blind, BlindSign, and Finalize -- and requires one round of interaction between client and server. Let msg be the client's private input message, and let (sk, pk) be the server's private and public key pair.

The protocol begins by the client preparing the message to be signed by computing:

```
input_msg = Prepare(msg)
```

The client then initiates the blind signature protocol by computing:

```
blinded_msg, inv = Blind(pk, input_msg)
```

The client then sends blinded_msg to the server, which then processes the message by computing:

```
blind_sig = BlindSign(sk, blinded_msg)
```

The server then sends blind_sig to the client, which then finalizes the protocol by computing:

```
sig = Finalize(pk, input_msg, blind_sig, inv)
```

The output of the protocol is input_msg and sig. Upon completion, correctness requires that clients can verify signature sig over the prepared message input_msg using the server public key pk by invoking the RSASSA-PSS-VERIFY routine defined in Section 8.1.2 of [RFC8017]. The Finalize function performs this check before returning the signature. See Section 4.5 for more details about verifying signatures produced through this protocol.

Shown graphically, the protocol runs as follows:

```
  Client(pk, msg)                        Server(sk, pk)
 -------------------------------------------------------
 input_msg = Prepare(msg)
 blinded_msg, inv = Blind(pk, input_msg)
```

```
                              blinded_msg
                              ---------->

                     blind_sig = BlindSign(sk, blinded_msg)

                              blind_sig
                              <----------

       sig = Finalize(pk, input_msg, blind_sig, inv)
```

In the remainder of this section, we specify the Prepare, Blind, BlindSign, and Finalize functions that are used in this protocol.

4.1. Prepare

Message preparation, denoted by the Prepare function, is the process by which the message to be signed and verified is prepared for input to the blind signing protocol. There are two types of preparation functions: an identity preparation function and a randomized preparation function. The identity preparation function returns the input message without transformation, i.e., msg = PrepareIdentity(msg).

The randomized preparation function augments the input message with fresh randomness. We denote this process by the function PrepareRandomize(msg), which takes as input a message msg and produces a randomized message input_msg. Its implementation is shown below.

PrepareRandomize(msg)

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

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

Steps:
1. msg_prefix = random(32)
2. input_msg = concat(msg_prefix, msg)
3. output input_msg

4.2. 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 Sections 5.2.2 and 9.1.1 of [RFC8017], respectively.

If this function fails with a "blinding error" error, implementations SHOULD try the function again. The probability of one or more such errors in sequence is negligible. This function can also fail with an "invalid input" error, which indicates that one of the inputs (likely the public key) was invalid. Implementations SHOULD update the public key before calling this function again. See Section 6.1

for more information about dealing with such errors.

Note that this function invokes RSAVP1, which is defined to throw an optional error for invalid inputs. However, this error cannot occur based on how RSAVP1 is invoked, so this error is not included in the list of errors for Blind.

Blind(pk, msg)

Parameters:
- modulus_len, the length in bytes of the RSA modulus n
- Hash, the hash function used to hash the message
- MGF, the mask generation function
- salt_len, the length in bytes of the salt (denoted sLen in RFC 8017)

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

Outputs:
- blinded_msg, a byte string of length modulus_len
- inv, an integer used to unblind the signature in Finalize

Errors:
- "message too long": Raised when the input message is too long (raised by EMSA-PSS-ENCODE)
- "encoding error": Raised when the input message fails encoding (raised by EMSA-PSS-ENCODE)
- "blinding error": Raised when the inverse of r cannot be found
- "invalid input": Raised when the message is not co-prime with n

Steps:
1. encoded_msg = EMSA-PSS-ENCODE(msg, bit_len(n))
   with Hash, MGF, and salt_len as defined in the parameters
2. If EMSA-PSS-ENCODE raises an error, re-raise the error and stop
3. m = bytes_to_int(encoded_msg)
4. c = is_coprime(m, n)
5. If c is false, raise an "invalid input" error and stop
6. r = random_integer_uniform(1, n)
7. inv = inverse_mod(r, n)
8. If inverse_mod fails, raise a "blinding error" error and stop
9. x = RSAVP1(pk, r)
10. z = (m * x) mod n
11. blinded_msg = int_to_bytes(z, modulus_len)
12. output blinded_msg, inv

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

4.3. 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(sk, blinded_msg)

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

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

Outputs:
- blind_sig, a byte string of length modulus_len

Errors:
- "signing failure": Raised when the signing operation fails
- "message representative out of range": Raised when the
  message representative to sign is not an integer between 0
  and n - 1 (raised by RSASP1)

Steps:
1. m = bytes_to_int(blinded_msg)
2. s = RSASP1(sk, m)
3. m' = RSAVP1(pk, s)
4. If m != m', raise a "signing failure" error and stop
5. blind_sig = int_to_bytes(s, modulus_len)
6. output blind_sig
```

4.4. 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(pk, msg, blind_sig, inv)

Parameters:
- modulus_len, the length in bytes of the RSA modulus n
- Hash, the hash function used to hash the message
- MGF, the mask generation function
- salt_len, the length in bytes of the salt (denoted sLen
  in RFC 8017)

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

Outputs:
- sig, a byte string of length modulus_len

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) != modulus_len, raise an "unexpected input size" error and stop
2. z = bytes_to_int(blind_sig)
3. s = (z * inv) mod n
4. sig = int_to_bytes(s, modulus_len)
5. result = RSASSA-PSS-VERIFY(pk, msg, sig) with Hash, MGF, and salt_len as defined in the parameters
6. If result = "valid signature", output sig, else raise an "invalid signature" error and stop

4.5. Verification

As described in Section 4, the output of the protocol is the prepared message input_msg and the signature sig. The message that applications consume is msg, from which input_msg is derived. Clients verify the msg signature using the server's public key pk by invoking the RSASSA-PSS-VERIFY routine defined in Section 8.1.2 of [RFC8017] with (n, e) as pk, M as input_msg, and S as sig.

Verification and the message that applications consume therefore depend on which preparation function is used. In particular, if the PrepareIdentity function is used, then the application message is input_msg. In contrast, if the PrepareRandomize function is used, then the application message is slice(input_msg, 32, len(input_msg)), i.e., the prepared message with the message randomizer prefix removed.

5. RSABSSA Variants

In this section, we define different named variants of RSABSSA. Each variant specifies EMSA-PSS options Hash, MGF, and sLen as defined in Section 9.1.1 of [RFC8017], as well as the type of message preparation function applied (as described in Section 4.1). Each variant uses the mask generation function 1 (MGF1) defined in Appendix B.2.1. of [RFC8017]. Future specifications can introduce other variants as desired. The named variants are as follows:

RSABSSA-SHA384-PSS-Randomized:
: This named variant uses SHA-384 as the EMSA-PSS Hash option, MGF1 with SHA-384 as the EMSA-PSS MGF option, and 48 as the EMSA-PSS sLen option (48-byte salt length); it also uses the randomized preparation function (PrepareRandomize).

RSABSSA-SHA384-PSSZERO-Randomized:
: This named variant uses SHA-384 as the EMSA-PSS Hash option, MGF1 with SHA-384 as the EMSA-PSS MGF option, and 0 as the EMSA-PSS sLen option (0-byte salt length); it also uses the randomized preparation function (PrepareRandomize).

RSABSSA-SHA384-PSS-Deterministic:
: This named variant uses SHA-384 as the EMSA-PSS Hash option, MGF1 with SHA-384 as the EMSA-PSS MGF option, and 48 as the EMSA-PSS sLen option (48-byte salt length); it also uses the identity

preparation function (PrepareIdentity).

RSABSSA-SHA384-PSSZERO-Deterministic:
This named variant uses SHA-384 as the EMSA-PSS Hash option, MGF1 with SHA-384 as the EMSA-PSS MGF option, and 0 as the EMSA-PSS sLen option (0-byte salt length); it also uses the identity preparation function (PrepareIdentity). This is the only variant that produces deterministic signatures over the client's input message msg.

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 randomized message preparation, as the additional message randomization does not offer security advantages. See [Lys22] and Section 7.3 for more information. For all other applications, the variants that use the randomized preparation function protect clients from malicious signers. A verifier that accepts randomized messages needs to remove the random component from the signed part of messages before processing.

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 7.3. When it is not clear whether an application requires deterministic or randomized signatures, applications SHOULD use one of the variants with randomized message preparation.

6. Implementation and Usage Considerations

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

6.1. 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.

Moreover, implementations can handle errors as needed or desired. Where applicable, this document provides guidance for how to deal with explicit errors that are generated in the protocol. For example, a "blinding error" error is generated in Blind when the client produces a prime factor of the server's public key. Section 4.2 indicates that implementations SHOULD retry the Blind function when this error occurs, but an implementation could also

handle this exceptional event differently, e.g., by informing the server that the key has been factored.

6.2. Signing Key Generation and Usage

The RECOMMENDED method for generating the server signing key pair is as specified in FIPS 186-5 [DSS].

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 id-RSASSA-PSS OID [RFC5756]. It MUST NOT use the rsaEncryption OID [RFC5280].

7. Security Considerations

Lysyanskaya proved one-more-forgery polynomial security of RSABSSA variants in the random oracle model under the one-more-RSA assumption; see [Lys22]. This means the adversary cannot output n+1 valid message and signature tuples, where all messages are distinct, after interacting with the server (signer) as a client only n times, for some n that is polynomial in the protocol's security parameter. Lysyanskaya also proved that the RSABSSA variants, which use the PrepareRandomize function, achieve blindness (see version B of the protocol and related proofs in [Lys22]). Blindness means that the malicious signer learns nothing about the client input and output after the protocol execution. However, additional assumptions on the message inputs are required for blindness to hold for RSABSSA variants that use the PrepareIdentity function; see Section 7.3 for more discussion on those results.

7.1. 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.

Moreover, we assume that the server does not initiate the protocol and therefore has no knowledge of when the Prepare and Blind operations take place. If this were not the case, additional side-channel mitigations might be required to prevent timing side channels through Prepare and Blind.

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(sk, m), verify that m

= RSAVP1(pk, s), as is required by BlindSign. Applying this (or an equivalent) safeguard is necessary to mitigate fault attacks, even for implementations that are not based on the Chinese remainder theorem.

7.2. 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 regarding arbitrary signing oracles. Applications using blind signature protocols should take precautions to ensure that such oracles do not cause cross-protocol attacks. Ensuring that the signing key used for RSABSSA is distinct from other protocols prevents such cross-protocol attacks.

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 proofs 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.

7.3. Message Entropy

As discussed in [Lys22], a malicious signer can construct an invalid public key and use it to learn information about low-entropy 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.

A client that uses this protocol might be vulnerable to attack from a malicious signer unless it is able to ensure that one of the following conditions is satisfied:

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

(2) The input message has a value that the signer is unable to guess. That is, the client has added a high-entropy component that was not available to the signer prior to them choosing their signing key.

The named variants that use the PrepareRandomize function -- 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. In contrast, the named variants that use the

PrepareIdentity function do not inject fresh entropy and therefore could be a problem with low-entropy inputs.

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.

7.4. Randomness Generation

All random values in the protocol, including the salt, message randomizer prefix (msg_prefix; see Appendix A), and random blind value in Blind, MUST be generated from a cryptographically secure random number generator [RFC4086]. If these values are not generated randomly or are otherwise constructed maliciously, it might be possible for them to encode information that is not present in the signed message. For example, the PSS salt might be maliciously constructed to encode the local IP address of the client. As a result, implementations SHOULD NOT allow clients to provide these values directly.

Note that malicious implementations could also encode client information in the message being signed, but since clients can verify the resulting message signature using the public key, this can be detected.

7.5. Key Substitution Attacks

RSA is well known for permitting key substitution attacks, wherein an attacker generates a key pair (skA, pkA) that verifies some known (message, signature) pair produced under a different (sk, pk) 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 that a (message, signature) pair verifies with a valid public key from the expected issuer.

7.6. Alternative RSA Encoding Functions

This 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.509 [RFC4055], and even PKCS #1 itself. According to [RFC8017], "Although no attacks are known against RSASSA-PKCS1-v1_5, in the interest of increased robustness, RSASSA-PSS is REQUIRED in new applications." While RSA-PSS is more complex than RSASSA-PKCS1-v1_5 encoding, ubiquity of RSA-PSS support influenced the design decision in this document, despite PKCS #1 v1.5 having equivalent security properties for digital signatures [JKM18].

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

7.7. 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.

8. IANA Considerations

This document has no IANA actions.

9. References

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>.

[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>.

9.2. Informative References

[ApplePrivateRelay] "iCloud Private Relay Overview", December 2021, <https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf>.

[Chaum83] Chaum, D., "Blind Signatures for Untraceable Payments", Springer-Verlag, 1998, <https://sceweb.sce.uhcl.edu/yang/teaching/csci5234WebSecurityFall2011/Chaum-blind-signatures.PDF>.

[DSS] "Digital Signature Standard (DSS)", National Institute of Standards and Technology report, DOI 10.6028/nist.fips.186-5, February 2023, <https://doi.org/10.6028/NIST.FIPS.186-5>.

[FAULTS] Boneh, D., DeMillo, R. A., and R. J. 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>.

[GoogleVPN] "VPN by Google One, explained",

<https://one.google.com/about/vpn/howitworks>.

[GRSB19] Goldberg, S., Reyzin, L., Sagga, O., and F. Baldimtsi, "Efficient Noninteractive Certification of RSA Moduli and Beyond", October 2019, <https://eprint.iacr.org/2018/057.pdf>.

[JKK14] Jarecki, S., Kiayias, A., and H. Krawczyk, "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. A., 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, pp. 1195-1208, DOI 10.1145/3243734.3243798, September 2018, <https://eprint.iacr.org/2018/855>.

[KK18] Kakvi, S. A. 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>.

[Lys22] Lysyanskaya, A., "Security Analysis of RSA-BSSA", March 2023, <https://eprint.iacr.org/2022/895>.

[PrettyGoodPhonePrivacy] Schmitt, P. and B. Raghavan, "Pretty Good Phone Privacy", Proceedings of the 30th USENIX Security Symposium, August 2021, <https://www.usenix.org/conference/usenixsecurity21/presentation/schmitt>.

[PRIVACY-PASS] Celi, S., Davidson, A., Valdez, S., and C. A. Wood, "Privacy Pass Issuance Protocol", Work in Progress, Internet-Draft, draft-ietf-privacypass-protocol-16, 3 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-protocol-16>.

[RemoteTimingAttacks] Brumley, D. and D. Boneh, "Remote Timing Attacks are Practical", Proceedings of the 12th USENIX Security Symposium, August 2003, <https://www.usenix.org/legacy/events/sec03/tech/brumley/brumley.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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005,

<https://www.rfc-editor.org/info/rfc4086>.

[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] Bellare, M. and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", CCS '93: Proceedings of the 1st ACM conference on Computer and communications security, pp. 62-73, DOI 10.1145/168588.168596, December 1993, <https://dl.acm.org/doi/abs/10.1145/168588.168596>.

[TimingAttacks] Kocher, P. C., "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>.

[VOPRF] Davidson, A., Faz-Hernandez, A., Sullivan, N., and C. A. Wood, "Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups", Work in Progress, Internet-Draft, draft-irtf-cfrg-voprf-21, 21 February 2023, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-voprf-21>.

[WM99] Blake-Wilson, S. and A. Menezes, "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol", International Workshop on Public Key Cryptography, PKC 1999, pp. 154-170, DOI 10.1007/3-540-49162-7_12, October 1999, <https://link.springer.com/chapter/10.1007/3-540-49162-7_12>.

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 (sk and pk) parameters, each encoded as a hexadecimal string.

msg:
Input message 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 the randomization preparation function.

prepared_msg:
The message actually signed. If the variant does not use the randomization preparation function, 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 = e1f4d7a34802e27c7392a3cea32a262a34dc3691bd87f3f310dc756734889305
59c120fd0410194fb8a0da55bd0b81227e843fdca6692ae80e5a5d414116d4803fca
7d8c30eaaae57e44a1816ebb5c5b0606c536246c7f11985d731684150b63c9a3ad9e
41b04c0b5b27cb188a692c84696b742a80d3cd00ab891f2457443dadfeba6d6daf10
8602be26d7071803c67105a5426838e6889d77e8474b29244cefaf418e381b312048
b457d73419213063c60ee7b0d81820165864fef93523c9635c22210956e53a8d9632
2493ffc58d845368e2416e078e5bcb5d2fd68ae6acfa54f9627c42e84a9d3f277401
7e32ebca06308a12ecc290c7cd1156dcccfb2311
q = c601a9caea66dc3835827b539db9df6f6f5ae77244692780cd334a006ab353c8
06426b60718c05245650821d39445d3ab591ed10a7339f15d83fe13f6a3dfb20b945
2c6a9b42eaa62a68c970df3cadb2139f804ad8223d56108dfde30ba7d367e9b0a7a8
0c4fdba2fd9dde6661fc73fc2947569d2029f2870fc02d8325acf28c9afa19ecf962
daa7916e21afad09eb62fe9f1cf91b77dc879b7974b490d3ebd2e95426057f35d0a3
c9f45f79ac727ab81a519a8b9285932d9b2e5ccd347e59f3f32ad9ca359115e7da00
8ab7406707bd0e8e185a5ed8758b5ba266e8828f8d863ae133846304a2936ad7bc7c
9803879d2fc4a28e69291d73dbd799f8bc238385
n = aec4d69addc70b990ea66a5e70603b6fee27aafebd08f2d94cbe1250c556e047
a928d635c3f45ee9b66d1bc628a03bac9b7c3f416fe20dabea8f3d7b4bbf7f963be3
35d2328d67e6c13ee4a8f955e05a3283720d3e1f139c38e43e0338ad058a9495c533
77fc35be64d208f89b4aa721bf7f7d3fef837be2a80e0f8adf0bcd1eec5bb040443a
2b2792fdca522a7472aed74f31a1ebe1eebc1f408660a0543dfe2a850f106a617ec6
685573702eaaa21a5640a5dcaf9b74e397fa3af18a2f1b7c03ba91a6336158de420d
63188ee143866ee415735d155b7c2d854d795b7bc236cffd71542df34234221a0413
e142d8c61355cc44d45bda94204974557ac2704cd8b593f035a5724b1adf442e78c5
42cd4414fce6f1298182fb6d8e53cef1adfd2e90e1e4deec52999bdc6c29144e8d52
a125232c8c6d75c706ea3cc06841c7bda33568c63a6c03817f722b50fcf898237d78
8a4400869e44d90a3020923dc646388abcc914315215fcd1bae11b1c751fd52443aa
c8f601087d8d42737c18a3fa11ecd4131ecae017ae0a14acfc4ef85b83c19fed33cf
d1cd629da2c4c09e222b398e18d822f77bb378dea3cb360b605e5aa58b20edc29d00
```

0a66bd177c682a17e7eb12a63ef7c2e4183e0d898f3d6bf567ba8ae84f84f1d23bf8
b8e261c3729e2fa6d07b832e07cddd1d14f55325c6f924267957121902dc19b3b329
48bdead5
e = 010001
d = 0d43242aefe1fb2c13fbc66e20b678c4336d20b1808c558b6e62ad16a2870771
80b177e1f01b12f9c6cd6c52630257ccef26a45135a990928773f3bd2fc01a313f1d
ac97a51cec71cb1fd7efc7adffdeb05f1fb04812c924ed7f4a8269925dad88bd7dcf
bc4ef01020ebfc60cb3e04c54f981fdbd273e69a8a58b8ceb7c2d83fbcbd6f784d05
2201b88a9848186f2a45c0d2826870733e6fd9aa46983e0a6e82e35ca20a439c5ee7
b502a9062e1066493bdadf8b49eb30d9558ed85abc7afb29b3c9bc644199654a4676
681af4babcea4e6f71fe4565c9c1b85d9985b84ec1abf1a820a9bbebee0df1398aae
2c85ab580a9f13e7743afd3108eb32100b870648fa6bc17e8abac4d3c99246b1f0ea
9f7f93a5dd5458c56d9f3f81ff2216b3c3680a13591673c43194d8e6fc93fc1e37ce
2986bd628ac48088bc723d8fbe293861ca7a9f4a73e9fa63b1b6d0074f5dea2a624c
5249ff3ad811b6255b299d6bc5451ba7477f19c5a0db690c3e6476398b1483d10314
afd38bbaf6e2fbdbcd62c3ca9797a420ca6034ec0a83360a3ee2adf4b9d4ba29731d
131b099a38d6a23cc463db754603211260e99d19affc902c915d7854554aabf608e3
ac52c19b8aa26ae042249b17b2d29669b5c859103ee53ef9bdc73ba3c6b537d5c34b
6d8f034671d7f3a8a6966cc4543df2235653431541 40fd7391c7e7be03e241f4ecfe
b877a051
msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c80023a
a6b59f8cfec5fdbb36331372ebefedae7d
msg_prefix = 8417e699b219d583fb6216ae0c53ca0e9723442d02f1d1a34295527
e7d929e8b
prepared_msg = 8417e699b219d583fb6216ae0c53ca0e9723442d02f1d1a342955
27e7d929e8b8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c8
0023aa6b59f8cfec5fdbb36331372ebefedae7d
salt = 051722b35f458781397c3a671a7d3bd3096503940e4c4f1aaa269d60300ce
449555cd7340100df9d46944c5356825abf
encoded_msg = 2be01c5669eb676cb3f0002eb636427d61568f3f0579da5b998279
a7eb3ab784e5617319376d04809d83e72bef9f0738e7324af3fd1b4f0a35f4f58058
ab329495406bdb5ff31a0274be2d137c735ab0d5a591b3129a6cc46fcecc4b41dbc6
84c965cb30e3eb4864ef18cc8d95b4d6a2002607c821d4d8a7e026ae7bb1f6b4c7c9
3d1b58e9cd87864d6094b0d8f7e2b5f966473703634fb58c774dd4a24376e0eb262a
24b58e3a0b4da4f36ef75651627561ff2ecee9dcbfe1d728cc31a7b46030f7a2815a
e9edf9a2a5c0c6d8dbab1b33b9c3bbda5c083670a3550f7d74c4263aad09f8ed1d43
5fc6295ca4d51fc02c7de9ae28ffd53372c3fa864521b27560daa11ab9daad8d0d74
7661718d2f79c59d0661b09c74863fa32bdcb1c408d3bd24569c57aecae6e06c0c9d
eb7303c5b7b1240960fd2413d61b2e3829af8c09874fdba0fe84ca6aa7e7d533f9b0
ddfe508f562b132ca2d325f1e73f91a8a6b831a2fd9bc0bd5bfa5ea3a1dee16bd9b2
64174b9553a4c0c0d62373353355c05b35824e4bae702f49e5a6bf83eaff65af4990
45bcef1470a0e58ddb21856034af0db96f8636d4a6f1591f34c7224e0c0293e3d3be
2139f2797c5ed8b65473ac2f83c52b87f8cf8754ac2f55f5e41e105df1d079a647fb
1aa591526295667f37db1129752d024eb03bfe506a4366507211842335 1ef9b86633
76f9fc073141e1e7bc
inv = 80682c48982407b489d53d1261b19ec8627d02b8cda5336750b8cee332ae26
0de57b02d72609c1e0e9f28e2040fc65b6f02d56dbd6aa9af8fde656f70495dfb723
ba01173d4707a12fddac628ca29f3e32340bd8f7ddb557cf819f6b01e445ad96f874
ba235584ee71f6581f62d4f43bf03f910f6510deb85e8ef06c7f09d9794a008be7ff
2529f0ebb69decef646387dc767b74939265fec0223aa6d84d2a8a1cc912d5ca25b4
e144ab8f6ba054b54910176d5737a2cff011da431bd5f2a0d2d66b9e70b39f4b050e
45c0d9c16f02deda9ddf2d00f3e4b01037d7029cd49c2d46a8e1fc2c0c17520af1f4
b5e25ba396afc4cd60c494a4c426448b35b49635b337cfb08e7c22a39b256dd032c0
0adddafb51a627f99a0e1704170ac1f1912e49d9db10ec04c19c58f420212973e0cb
329524223a6aa56c7937c5dffdb5d966b6cd4cbc26f3201dd25c80960a1a111b3294
7bb78973d269fac7f5186530930ed19f68507540eed9e1bab8b00f00d8ca09b3f099

aae46180e04e3584bd7ca054df18a1504b89d1d1675d0966c4ae1407be325cdf623c
f13ff13e4a28b594d59e3eadbadf6136eee7a59d6a444c9eb4e2198e8a974f27a39e
b63af2c9af3870488b8adaad444674f512133ad80b9220e09158521614f1faadfe85
05ef57b7df6813048603f0dd04f4280177a11380fbfc861dbcbd7418d62155248dad
5fdec0991f

blinded_msg = aa3ee045138d874669685ffaef962c7694a9450aa9b4fd6465db9b
3b75a522bb921c4c0fdcdfae9667593255099cff51f5d3fd65e8ffb9d3b3036252a6
b51b6edfb3f40382b2bbf34c0055e4cbcc422850e586d84f190cd449af11dc65545f
5fe26fd89796eb87da4bda0c545f397cddfeeb56f06e28135ec74fd477949e7677f6
f36cfae8fd5c1c5898b03b9c244cf6d1a4fb7ad1cb43aff5e80cb462fac541e72f67
f0a50f1843d1759edfaae92d1a916d3f0efaf4d650db416c3bf8abdb5414a78cebc9
7de676723cb119e77aea489f2bbf530c440ebc5a75dccd3ebf5a412a5f346badd61b
ee588e5917bdcce9dc33c882e39826951b0b8276c6203971947072b726e935816056
ff5cb11a71ca2946478584126bb877acdf87255f26e6cca4e0878801307485d3b7bb
89b289551a8b65a7a6b93db010423d1406e149c87731910306e5e410b41d4da32346
24e74f92845183e323cf7eb244f212a695f8856c675fbc3a021ce649e22c6f0d053a
9d238841cf3afdc2739f99672a419ae13c17f1f8a3bc302ec2e7b98e8c353898b715
0ad8877ec841ea6e4b288064c254fefd0d049c3ad196bf7ffa535e74585d0120ce72
8036ed500942fbd5e6332c298f1ffebe9ff60c1e117b274cf0cb9d70c36ee4891528
996ec1ed0b178e9f3c0c0e6120885f39e8ccaadbb20f3196378c07b1ff22d10049d3
039a7a92fe7efdd95d

blind_sig = 3f4a79eacd4445fca628a310d41e12fcd813c4d43aa4ef2b81226953
248d6d00adfee6b79cb88bfa1f99270369fd063c023e5ed546719b0b2d143dd1bca4
6b0e0e615fe5c63d95c5a6b873b8b50bc52487354e69c3dfbf416e7aca18d5842c89
b676efdd38087008fa5a810161fcdec26f20ccf2f1e6ab0f9d2bb93e051cb9e86a9b
28c5bb62fd5f5391379f887c0f706a08bcc3b9e7506aaf02485d688198f5e22eefdf
837b2dd919320b17482c5cc54271b4ccb41d267629b3f844fd63750b01f5276c79e3
3718bb561a152acb2eb36d8be75bce05c9d1b94eb609106f38226fb2e0f5cd5c5c39
c59dda166862de498b8d92f6bcb41af433d65a2ac23da87f39764cb64e79e74a8f4c
e4dd567480d967cefac46b6e9c06434c3715635834357edd2ce6f105eea854ac126c
cfa3de2aac5607565a4e5efaac5eed491c335f6fc97e6eb7e9cea3e12de38dfb3152
20c0a3f84536abb2fdd722813e083feda010391ac3d8fd1cd9212b5d94e634e69ebc
c800c4d5c4c1091c64afc37acf563c7fc0a6e4c082bc55544f50a7971f3fb97d5853
d72c3af34ffd5ce123998be5360d1059820c66a81e1ee6d9c1803b5b62af6bc87752
6df255b6d1d835d8c840bebbcd6cc0ee910f17da37caf8488afbc08397a1941fcc79
e76a5888a95b3d5405e13f737bea5c78d716a48eb9dc0aec8de39c4b45c6914ad4a8
185969f70b1adf46

sig = 191e941c57510e22d29afad257de5ca436d2316221fe870c7cb75205a6c071
c2735aed0bc24c37f3d5bd960ab97a829a508f966bbaed7a82645e65eadaf24ab5e6
d9421392c5b15b7f9b640d34fec512846a3100b80f75ef51064602118c1a77d28d93
8f6efc22041d60159a518d3de7c4d840c9c68109672d743d299d8d2577ef60c19ab4
63c716b3fa75fa56f5735349d414a44df12bf0dd44aa3e10822a651ed4cb0eb6f47c
9bd0ef14a034a7ac2451e30434d513eb22e68b7587a8de9b4e63a059d05c8b22c7c5
1e2cfee2d8bef511412e93c859a13726d87c57d1bc4c2e68ab121562f839c3a3d233
e87ed63c69b7e57525367753fbebcc2a9805a2802659f5888b2c69115bf865559f10
d906c09d048a0d71bfee4b33857393ec2b69e451433496d02c9a7910abb954317720
bbde9e69108eafc3e90bad3d5ca4066d7b1e49013fa04e948104a1dd82b12509ecb1
46e948c54bd8bfb5e6d18127cd1f7a93c3cf9f2d869d5a78878c03fe808a0d799e91
0be6f26d18db61c485b303631d3568368fc41986d08a95ea6ac0592240c19d7b2241
6b9c82ae6241e211dd5610d0baaa9823158f9c32b66318f5529491b7eeadcaa71898
a63bac9d95f4aa548d5e97568d744fc429104e32edd9c87519892a198a30d333d427
739ffb9607b092e910ae37771abf2adb9f63bc058bf58062ad456cb934679795bbdf
cdfad5e0f2

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

p = e1f4d7a34802e27c7392a3cea32a262a34dc3691bd87f3f310dc756734889305
59c120fd0410194fb8a0da55bd0b81227e843fdca6692ae80e5a5d414116d4803fca
7d8c30eaaae57e44a1816ebb5c5b0606c536246c7f11985d731684150b63c9a3ad9e
41b04c0b5b27cb188a692c84696b742a80d3cd00ab891f2457443dadfeba6d6daf10
8602be26d7071803c67105a5426838e6889d77e8474b29244cefaf418e381b312048
b457d73419213063c60ee7b0d81820165864fef93523c9635c22210956e53a8d9632
2493ffc58d845368e2416e078e5bcb5d2fd68ae6acfa54f9627c42e84a9d3f277401
7e32ebca06308a12ecc290c7cd1156dcccfb2311
q = c601a9caea66dc3835827b539db9df6f6f5ae77244692780cd334a006ab353c8
06426b60718c05245650821d39445d3ab591ed10a7339f15d83fe13f6a3dfb20b945
2c6a9b42eaa62a68c970df3cadb2139f804ad8223d56108dfde30ba7d367e9b0a7a8
0c4fdba2fd9dde6661fc73fc2947569d2029f2870fc02d8325acf28c9afa19ecf962
daa7916e21afad09eb62fe9f1cf91b77dc879b7974b490d3ebd2e95426057f35d0a3
c9f45f79ac727ab81a519a8b9285932d9b2e5ccd347e59f3f32ad9ca359115e7da00
8ab7406707bd0e8e185a5ed8758b5ba266e8828f8d863ae133846304a2936ad7bc7c
9803879d2fc4a28e69291d73dbd799f8bc238385
n = aec4d69addc70b990ea66a5e70603b6fee27aafebd08f2d94cbe1250c556e047
a928d635c3f45ee9b66d1bc628a03bac9b7c3f416fe20dabea8f3d7b4bbf7f963be3
35d2328d67e6c13ee4a8f955e05a3283720d3e1f139c38e43e0338ad058a9495c533
77fc35be64d208f89b4aa721bf7f7d3fef837be2a80e0f8adf0bcd1eec5bb040443a
2b2792fdca522a7472aed74f31a1ebe1eebc1f408660a0543dfe2a850f106a617ec6
685573702eaaa21a5640a5dcaf9b74e397fa3af18a2f1b7c03ba91a6336158de420d
63188ee143866ee415735d155b7c2d854d795b7bc236cffd71542df34234221a0413
e142d8c61355cc44d45bda94204974557ac2704cd8b593f035a5724b1adf442e78c5
42cd4414fce6f1298182fb6d8e53cef1adfd2e90e1e4deec52999bdc6c29144e8d52
a125232c8c6d75c706ea3cc06841c7bda33568c63a6c03817f722b50fcf898237d78
8a4400869e44d90a3020923dc646388abcc914315215fcd1bae11b1c751fd52443aa
c8f601087d8d42737c18a3fa11ecd4131ecae017ae0a14acfc4ef85b83c19fed33cf
d1cd629da2c4c09e222b398e18d822f77bb378dea3cb360b605e5aa58b20edc29d00
0a66bd177c682a17e7eb12a63ef7c2e4183e0d898f3d6bf567ba8ae84f84f1d23bf8
b8e261c3729e2fa6d07b832e07cddd1d14f55325c6f924267957121902dc19b3b329
48bdead5
e = 010001
d = 0d43242aefe1fb2c13fbc66e20b678c4336d20b1808c558b6e62ad16a2870771
80b177e1f01b12f9c6cd6c52630257ccef26a45135a990928773f3bd2fc01a313f1d
ac97a51cec71cb1fd7efc7adffdeb05f1fb04812c924ed7f4a8269925dad88bd7dcf
bc4ef01020ebfc60cb3e04c54f981fdbd273e69a8a58b8ceb7c2d83fbcbd6f784d05
2201b88a9848186f2a45c0d2826870733e6fd9aa46983e0a6e82e35ca20a439c5ee7
b502a9062e1066493bdadf8b49eb30d9558ed85abc7afb29b3c9bc644199654a4676
681af4babcea4e6f71fe4565c9c1b85d9985b84ec1abf1a820a9bbebee0df1398aae
2c85ab580a9f13e7743afd3108eb32100b870648fa6bc17e8abac4d3c99246b1f0ea
9f7f93a5dd5458c56d9f3f81ff2216b3c3680a13591673c43194d8e6fc93fc1e37ce
2986bd628ac48088bc723d8fbe293861ca7a9f4a73e9fa63b1b6d0074f5dea2a624c
5249ff3ad811b6255b299d6bc5451ba7477f19c5a0db690c3e6476398b1483d10314
afd38bbaf6e2fbdbcd62c3ca9797a420ca6034ec0a83360a3ee2adf4b9d4ba29731d
131b099a38d6a23cc463db754603211260e99d19affc902c915d7854554aabf608e3
ac52c19b8aa26ae042249b17b2d29669b5c859103ee53ef9bdc73ba3c6b537d5c34b
6d8f034671d7f3a8a6966cc4543df2235653431541 40fd7391c7e7be03e241f4ecfe
b877a051
msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c80023a
a6b59f8cfec5fdbb36331372ebefedae7d
msg_prefix = 84ea86c8cf3beedfed73beceabd792027c609d1100bf041fdd60d82
6a718130d
prepared_msg = 84ea86c8cf3beedfed73beceabd792027c609d1100bf041fdd60d
826a718130d8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c8
0023aa6b59f8cfec5fdbb36331372ebefedae7d

salt =
encoded_msg = 37f4ea66054b3570f2c46f43125a8df8d751a81db1003edcc70e98
88cb3d0fa71bb7634437a779c1bf9e84e88b3479894490ee41cd69fc8e911478326f
e8460d1699f96abedde22ba0ba25a02f78bae77eb039decd41e6cd40fecc28f301c9
4d5644eb3e55b316569e2bec3ccf8e33b06eb6defca5fe672613d33ea60f84daa560
ded4c1c5e65613fb19e090d0fc96a1394e29dfad6a7644362bf30bdc90c7ca0a0651
90f5a099b5c33ae787b872518a724d9aa139229656eb21053bbe86c38f6d03b4c6fa
37a900935d9b8d19e0c394be4af6af028680996e3fd533b6698ce9e2ed6a9f96d4d3
a682027ae5240040e55d75017dc303b7142c1f7e17b79778a94431398d21dc0cc7ae
454cc0d6cf4db4d588d3fd15fd7f71576052fd2a52d688f99790dfb13808ecb24b6b
9e9a43a8c0105670ec3ad8d6318a9c6a9cef9eb99b36d74b8e83dbacf6e8100e135b
609850b34a4b01091b263678d7cd9905af2ffda801a2888d863a25211903b43cb5e5
9f5dba6bc18713ce4f028f1774c593664912f1d181d4544a13a1da354332d8595f59
cf5af260a8aaf21a6bc948b5d5d4a520c1f72c216259dc12a33c2a3bd4d32ff2bf3d
e2ffe76e51f8af030b40fadc5899e740da20be1dd5a50f701292ceaee51fa35d9a04
7f3efc6543dc583fb3f23abeade39c2a5b5b352de26d7a11267435be7bffa8f2292e
139fad923dbaf863bc
inv = 80682c48982407b489d53d1261b19ec8627d02b8cda5336750b8cee332ae26
0de57b02d72609c1e0e9f28e2040fc65b6f02d56dbd6aa9af8fde656f70495dfb723
ba01173d4707a12fddac628ca29f3e32340bd8f7ddb557cf819f6b01e445ad96f874
ba235584ee71f6581f62d4f43bf03f910f6510deb85e8ef06c7f09d9794a008be7ff
2529f0ebb69decef646387dc767b74939265fec0223aa6d84d2a8a1cc912d5ca25b4
e144ab8f6ba054b54910176d5737a2cff011da431bd5f2a0d2d66b9e70b39f4b050e
45c0d9c16f02deda9ddf2d00f3e4b01037d7029cd49c2d46a8e1fc2c0c17520af1f4
b5e25ba396afc4cd60c494a4c426448b35b49635b337cfb08e7c22a39b256dd032c0
0adddafb51a627f99a0e1704170ac1f1912e49d9db10ec04c19c58f420212973e0cb
329524223a6aa56c7937c5dffdb5d966b6cd4cbc26f3201dd25c80960a1a111b3294
7bb78973d269fac7f5186530930ed19f68507540eed9e1bab8b00f00d8ca09b3f099
aae46180e04e3584bd7ca054df18a1504b89d1d1675d0966c4ae1407be325cdf623c
f13ff13e4a28b594d59e3eadbadf6136eee7a59d6a444c9eb4e2198e8a974f27a39e
b63af2c9af3870488b8adaad444674f512133ad80b9220e09158521614f1faadfe85
05ef57b7df6813048603f0dd04f4280177a11380fbfc861dbcbd7418d62155248dad
5fdec0991f
blinded_msg = 4c1b82d9b97b968b2ce0754e326abd49e3d723ed937d84bead34b6
a834483b43d510bf62ca47683ed366d94d3d357b270a85cf2cc2ddd171141b45d754
9d5373cf67d14f6f462c14ebded906793144faba37f129c0f3172854ec0f854e5555
52eec5a30c87788f1039814594f04348709e26a883be82afff207b1886b75c037f4
3f847f45d89bcbf210c22ffcdf8118ce8a526b3723e6209c26319f8f5d2adcf0b637
031c9fdf53470a915c587e30287ba88ed4f1cd5e93cf3d4990acf31fffdbfddec80a
e0b728d5b4c612a396fd81acaa65566a4dc1c24624f44fd10cdba05f3d0bed2e69bb
0d13d41a9f1b4e67aa566520778733ced5e6260f4d1982f63bb835442acffe3cb87f
5f8ec6bb84226e0eab787159d08e57604b13557ceea97f2c4ad0631accf898f302df
86f0b64354ec0b3bdf1b4e2a4deb4d38f655ea8d80de4cc19aa06ffcd56e348faf89
4c8774c53235ddcc152d80cf66b417eee4d182781bab8c979937a3c7502d8f39c57c
4f09884de5a7247f2539910a96e4b15f9a3df88edc21a13030af357467a99dca50db
a4afe4a6185a240ac8f1d8aab2e83443025f94e1af930f56f78661369cc6790701f3
1b83aec40f96a72c7f7ba13b4ebdd8e24e7351f4ffba0a7c072cb28f13aff06cd023
68491044fcc536213b2e3b1cf6ca81cf2097b7b19d2b36bd246f390f53768f1c2e56
113ea91b33c7cfa647
blind_sig = 4894f64d7214c216282d9842cbf7e7cccd9c0dcb1f4294a6bdeccd4c
4c2446160d7cac7892f01b70dfa69f533891d2fbb447f7cf7541d1b504a2d46fc1bb
6de26b345972aada8ebce280b906f3a10a13208f77ef896fbe6bc4504327fd4c5c8f
03211d45ae9672e9f4be0f4900762ba2a7177a58b90d6dd1263faf2b7a5f15d50a7b
00e733742c1b6a1ea4eb5fbfb407abf14496ab26b50cf1a5a56dea616b7a6a559577
7400571a751c682b9fdd6badb3f72292f314f4ba2ba0f394f91676a4bb12e60ea08c
977f7082be6357c1ca82fe3301fe5fb4128609bee2410db0481aea3a5737fb0bce93

81272c2202644f662e99f64bf1190d66e230cc0371ec33fe32fe725dfd872041914d
39462a909414a780c9aab394af443199eba56c83986d22d57d4421b41ff8e5bec537
d271223adb34d26c64989048a88d8f352a06a7cc153e216a6bed9548bb38d2a1600b
2f3403289df6df74aec525ef9e413b7140a7c1a914dedd74a336f1beed39a8e5e2ce
f76cac094df0dbb3fa55d4b7ee781c74bed3bd8bc7aa6ef3f1dbfa4674945720ec93
dafa6d0650229ab75e3fae687327fac081cf4bb376e02a2b73314c54c12f88572c28
980f13aba5731bc5a3a60575ea116c8ea2fe5009168deb1255026c9310783ff7f644
255d3e1691e194db1babd7780b9a5dc0cb3de2b700d12f49cbe4db51ca2f3c8a58b0
9e854cc71e8070ab
sig = 195363ba25e4bf763f6538c86865785f93f4ea6092da3ad200d41b99eb0eb0
869fa792df619fd8fa5923d5d03d5882faae6d25054118deef5e4a6a252dd5afb0da
c262b74c391090b1575fbafd959d26bc294f47fb45a2c1c209932c4f94b24394eded
91fbdd015e1a85dde63c9e77a0283f812cad1192d86432c51331e46fd4f3771bbafb
929f847a19cb05e5f79b6b519d67e8f005951e53656be97cb612d2f506618b366403
b34648451d6fbc7318c2f3f583cc6fa17bf2108398f9284e0602187904406a9322f1
e7b8016ca9ad11b835756df862c465c420535e25faa48bf341f7ee8192be47fa8757
91f32f56d5e631d237060688f052426dee5b0b2b74ca5f830e82a453379eedb541fa
4fcdaa19dae6509401e3cdd4c40f5c9243db3f6d7115c4e8cd6db8290723ab01d9d0
d7e355a97a01547800e43f11736668c3f8908848d759c33a67a2f506abc3f6871cbe
625b1bc71eb06d785a59501396712c581a60d6ccc450d2f4eb4cf08ae0dbfa45c286
0425be90cc4cd4c989495bbd2963e19c59ae5d90d1ca884e80d654b5f2cd6a80c358
8b514ee91c802736f594c340397b316a97e9c70b0609955b6c3ee06f4760d9377f07
97a0411a244db395bb8b711ef79fbcb5589226174029be79a72dcd6f4ca566b7b1b9
a27e43b5c02a9a579d60bdda183398d66d76e0e8eceb1af2f27633589d043bcdc041
683b31f7f1

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

p = e1f4d7a34802e27c7392a3cea32a262a34dc3691bd87f3f310dc756734889305
59c120fd0410194fb8a0da55bd0b81227e843fdca6692ae80e5a5d414116d4803fca
7d8c30eaaae57e44a1816ebb5c5b0606c536246c7f11985d731684150b63c9a3ad9e
41b04c0b5b27cb188a692c84696b742a80d3cd00ab891f2457443dadfeba6d6daf10
8602be26d7071803c67105a5426838e6889d77e8474b29244cefaf418e381b312048
b457d73419213063c60ee7b0d81820165864fef93523c9635c22210956e53a8d9632
2493ffc58d845368e2416e078e5bcb5d2fd68ae6acfa54f9627c42e84a9d3f277401
7e32ebca06308a12ecc290c7cd1156dcccfb2311
q = c601a9caea66dc3835827b539db9df6f6f5ae77244692780cd334a006ab353c8
06426b60718c05245650821d39445d3ab591ed10a7339f15d83fe13f6a3dfb20b945
2c6a9b42eaa62a68c970df3cadb2139f804ad8223d56108dfde30ba7d367e9b0a7a8
0c4fdba2fd9dde6661fc73fc2947569d2029f2870fc02d8325acf28c9afa19ecf962
daa7916e21afad09eb62fe9f1cf91b77dc879b7974b490d3ebd2e95426057f35d0a3
c9f45f79ac727ab81a519a8b9285932d9b2e5ccd347e59f3f32ad9ca359115e7da00
8ab7406707bd0e8e185a5ed8758b5ba266e8828f8d863ae133846304a2936ad7bc7c
9803879d2fc4a28e69291d73dbd799f8bc238385
n = aec4d69addc70b990ea66a5e70603b6fee27aafebd08f2d94cbe1250c556e047
a928d635c3f45ee9b66d1bc628a03bac9b7c3f416fe20dabea8f3d7b4bbf7f963be3
35d2328d67e6c13ee4a8f955e05a3283720d3e1f139c38e43e0338ad058a9495c533
77fc35be64d208f89b4aa721bf7f7d3fef837be2a80e0f8adf0bcd1eec5bb040443a
2b2792fdca522a7472aed74f31a1ebe1eebc1f408660a0543dfe2a850f106a617ec6
685573702eaaa21a5640a5dcaf9b74e397fa3af18a2f1b7c03ba91a6336158de420d
63188ee143866ee415735d155b7c2d854d795b7bc236cffd71542df34234221a0413
e142d8c61355cc44d45bda94204974557ac2704cd8b593f035a5724b1adf442e78c5
42cd4414fce6f1298182fb6d8e53cef1adfd2e90e1e4deec52999bdc6c29144e8d52
a125232c8c6d75c706ea3cc06841c7bda33568c63a6c03817f722b50fcf898237d78
8a4400869e44d90a3020923dc646388abcc914315215fcd1bae11b1c751fd52443aa
c8f601087d8d42737c18a3fa11ecd4131ecae017ae0a14acfc4ef85b83c19fed33cf

d1cd629da2c4c09e222b398e18d822f77bb378dea3cb360b605e5aa58b20edc29d00
0a66bd177c682a17e7eb12a63ef7c2e4183e0d898f3d6bf567ba8ae84f84f1d23bf8
b8e261c3729e2fa6d07b832e07cddd1d14f55325c6f924267957121902dc19b3b329
48bdead5
e = 010001
d = 0d43242aefe1fb2c13fbc66e20b678c4336d20b1808c558b6e62ad16a2870771
80b177e1f01b12f9c6cd6c52630257ccef26a45135a990928773f3bd2fc01a313f1d
ac97a51cec71cb1fd7efc7adffdeb05f1fb04812c924ed7f4a8269925dad88bd7dcf
bc4ef01020ebfc60cb3e04c54f981fdbd273e69a8a58b8ceb7c2d83fbcbd6f784d05
2201b88a9848186f2a45c0d2826870733e6fd9aa46983e0a6e82e35ca20a439c5ee7
b502a9062e1066493bdadf8b49eb30d9558ed85abc7afb29b3c9bc644199654a4676
681af4babcea4e6f71fe4565c9c1b85d9985b84ec1abf1a820a9bbebee0df1398aae
2c85ab580a9f13e7743afd3108eb32100b870648fa6bc17e8abac4d3c99246b1f0ea
9f7f93a5dd5458c56d9f3f81ff2216b3c3680a13591673c43194d8e6fc93fc1e37ce
2986bd628ac48088bc723d8fbe293861ca7a9f4a73e9fa63b1b6d0074f5dea2a624c
5249ff3ad811b6255b299d6bc5451ba7477f19c5a0db690c3e6476398b1483d10314
afd38bbaf6e2fbdbcd62c3ca9797a420ca6034ec0a83360a3ee2adf4b9d4ba29731d
131b099a38d6a23cc463db754603211260e99d19affc902c915d7854554aabf608e3
ac52c19b8aa26ae042249b17b2d29669b5c859103ee53ef9bdc73ba3c6b537d5c34b
6d8f034671d7f3a8a6966cc4543df2235653431541 40fd7391c7e7be03e241f4ecfe
b877a051
msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c80023a
a6b59f8cfec5fdbb36331372ebefedae7d
msg_prefix =
prepared_msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a6
98c80023aa6b59f8cfec5fdbb36331372ebefedae7d
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
c63a1eb599f093ab401d0c6003a451931b6d12418030570584506 0ebba6b0036154f
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.4. RSABSSA-SHA384-PSSZERO-Deterministic Test Vector

p = e1f4d7a34802e27c7392a3cea32a262a34dc3691bd87f3f310dc756734889305

59c120fd0410194fb8a0da55bd0b81227e843fdca6692ae80e5a5d414116d4803fca
7d8c30eaaae57e44a1816ebb5c5b0606c536246c7f11985d731684150b63c9a3ad9e
41b04c0b5b27cb188a692c84696b742a80d3cd00ab891f2457443dadfeba6d6daf10
8602be26d7071803c67105a5426838e6889d77e8474b29244cefaf418e381b312048
b457d73419213063c60ee7b0d81820165864fef93523c9635c22210956e53a8d9632
2493ffc58d845368e2416e078e5bcb5d2fd68ae6acfa54f9627c42e84a9d3f277401
7e32ebca06308a12ecc290c7cd1156dcccfb2311
q = c601a9caea66dc3835827b539db9df6f6f5ae77244692780cd334a006ab353c8
06426b60718c05245650821d39445d3ab591ed10a7339f15d83fe13f6a3dfb20b945
2c6a9b42eaa62a68c970df3cadb2139f804ad8223d56108dfde30ba7d367e9b0a7a8
0c4fdba2fd9dde6661fc73fc2947569d2029f2870fc02d8325acf28c9afa19ecf962
daa7916e21afad09eb62fe9f1cf91b77dc879b7974b490d3ebd2e95426057f35d0a3
c9f45f79ac727ab81a519a8b9285932d9b2e5ccd347e59f3f32ad9ca359115e7da00
8ab7406707bd0e8e185a5ed8758b5ba266e8828f8d863ae133846304a2936ad7bc7c
9803879d2fc4a28e69291d73dbd799f8bc238385
n = aec4d69addc70b990ea66a5e70603b6fee27aafebd08f2d94cbe1250c556e047
a928d635c3f45ee9b66d1bc628a03bac9b7c3f416fe20dabea8f3d7b4bbf7f963be3
35d2328d67e6c13ee4a8f955e05a3283720d3e1f139c38e43e0338ad058a9495c533
77fc35be64d208f89b4aa721bf7f7d3fef837be2a80e0f8adf0bcd1eec5bb040443a
2b2792fdca522a7472aed74f31a1ebe1eebc1f408660a0543dfe2a850f106a617ec6
685573702eaaa21a5640a5dcaf9b74e397fa3af18a2f1b7c03ba91a6336158de420d
63188ee143866ee415735d155b7c2d854d795b7bc236cffd71542df34234221a0413
e142d8c61355cc44d45bda94204974557ac2704cd8b593f035a5724b1adf442e78c5
42cd4414fce6f1298182fb6d8e53cef1adfd2e90e1e4deec52999bdc6c29144e8d52
a125232c8c6d75c706ea3cc06841c7bda33568c63a6c03817f722b50fcf898237d78
8a4400869e44d90a3020923dc646388abcc914315215fcd1bae11b1c751fd52443aa
c8f601087d8d42737c18a3fa11ecd4131ecae017ae0a14acfc4ef85b83c19fed33cf
d1cd629da2c4c09e222b398e18d822f77bb378dea3cb360b605e5aa58b20edc29d00
0a66bd177c682a17e7eb12a63ef7c2e4183e0d898f3d6bf567ba8ae84f84f1d23bf8
b8e261c3729e2fa6d07b832e07cddd1d14f55325c6f924267957121902dc19b3b329
48bdead5
e = 010001
d = 0d43242aefe1fb2c13fbc66e20b678c4336d20b1808c558b6e62ad16a2870771
80b177e1f01b12f9c6cd6c52630257ccef26a45135a990928773f3bd2fc01a313f1d
ac97a51cec71cb1fd7efc7adffdeb05f1fb04812c924ed7f4a8269925dad88bd7dcf
bc4ef01020ebfc60cb3e04c54f981fdbd273e69a8a58b8ceb7c2d83fbcbd6f784d05
2201b88a9848186f2a45c0d2826870733e6fd9aa46983e0a6e82e35ca20a439c5ee7
b502a9062e1066493bdadf8b49eb30d9558ed85abc7afb29b3c9bc644199654a4676
681af4babcea4e6f71fe4565c9c1b85d9985b84ec1abf1a820a9bbebee0df1398aae
2c85ab580a9f13e7743afd3108eb32100b870648fa6bc17e8abac4d3c99246b1f0ea
9f7f93a5dd5458c56d9f3f81ff2216b3c3680a13591673c43194d8e6fc93fc1e37ce
2986bd628ac48088bc723d8fbe293861ca7a9f4a73e9fa63b1b6d0074f5dea2a624c
5249ff3ad811b6255b299d6bc5451ba7477f19c5a0db690c3e6476398b1483d10314
afd38bbaf6e2fbdbcd62c3ca9797a420ca6034ec0a83360a3ee2adf4b9d4ba29731d
131b099a38d6a23cc463db754603211260e99d19affc902c915d7854554aabf608e3
ac52c19b8aa26ae042249b17b2d29669b5c859103ee53ef9bdc73ba3c6b537d5c34b
6d8f034671d7f3a8a6966cc4543df2235653431541 40fd7391c7e7be03e241f4ecfe
b877a051
msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a698c80023a
a6b59f8cfec5fdbb36331372ebefedae7d
msg_prefix =
prepared_msg = 8f3dc6fb8c4a02f4d6352edf0907822c1210a9b32f9bdda4c45a6
98c80023aa6b59f8cfec5fdbb36331372ebefedae7d
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

Acknowledgments

We would like to thank Bjoern Tackmann, who provided an editorial and security review of this document.

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