Internet Research Task Force
Internet-Draft
Updates: 8017 (if approved)
Intended status: Informational
Expires: 3 September 2026

A. Kario
Red Hat, Inc.

2 March 2026

Implementation Guidance for the PKCS #1 RSA Cryptography Specification
draft-irtf-cfrg-rsa-guidance-07

Abstract

This document specifies additions and amendments to RFC 8017. Specifically, it provides guidance to implementers of the standard to protect against side-channel attacks. It also deprecates the RSAES-PKCS-v1_5 encryption scheme, and provides an alternative depadding algorithm that protects against side-channel attacks raising from users of vulnerable APIs. The purpose of this specification is to increase security of RSA implementations.

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 3 September 2026.

Copyright Notice

Table of Contents

1. Introduction

The PKCS #1 [RFC8017] describes the RSA cryptosystem, providing guidance on implementing encryption schemes and signature schemes.

Unfortunately, straight-forward implementation of the RSA encryption schemes leave it vulnerable to side-channel attacks. Protections against them are not documented in RFC 8017, and attacks are mentioned only in passing.

2. Rationale

Since 1998, when Daniel Bleichenbacher published his attack [Bleichenbacher98], the RSAES-PKCS-v1_5 encryption scheme is known to be problematic. Side-channel attacks against public key algorithms, including RSA, are known to be possible since 1996 thanks to work by Paul Kocher [Kocher96].

Despite those results, side-channel attacks against RSA implementations have proliferated for the next 25 years. Including attacks against simple exponentiation implementations [Dhem98] [Schindler01], implementations that use the Chinese Remainder Theorem optimisation [Schindler00] [Brumley03] [Aciicmez05], and implementations that use either base or exponent blinding exclusively [Aciicmez07] [Aciicmez08] [Schindler14].

Similarly, side-channel-free handling of the errors from the RSAES-PKCS-v1_5 decryption operation is something that implementations struggle with [Boeck18] [Kario23].

We provide general guidance to implementers of the RSA cryptosystem for protections against side-channel attacks, irrespective of the padding scheme used. We also call for complete deprecation of the RSAES-PKCS-v1_5 scheme in any deployments or protocols. For legacy use cases that cannot remove use of the RSAES-PKCS-v1_5 scheme we provide guidance how to implement it in a way that should be secure against at least the simple timing side-channel attacks.

3. Requirements Language

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.

4. Notation

In this document we reuse the notation from [RFC8017], in addition, we define the following:

AL alternative message length, non-negative integer, 0 <= AL <= k - 11

AM alternative encoded message, an octet string

bb base blinding factor, a positive integer

bbInv base un-blinding factor, a positive integer,

bbInv = bb^(-1) mod n

D octet string representation of d

DH an octet string of a SHA-256 [SHA2] hash of D

HMAC (key, msg, hash) an HMAC [RFC2104] calculated using the specified underlying hash, with the provided key, over the provided message msg

KDK an octet string containing a Key Derivation Key for a specific ciphertext C

l length in octets of the message M

b_i an exponent blinding factor for i-th prime, non-negative integer.

g_i a modulus blinding factor for the i-th prime, non-negative integer.

5. Side channel attacks

Cryptographic implementations may provide a lot of indirect signals to an attacker that includes information about the secret processed data. Depending on the type of information, those leaks can be used to decrypt data or retrieve private keys. Those leaks are called "side-channel leaks". Most common side-channels that leak information about secret data are:

1. Different errors returned
2. Different processing times of operations
3. Different patterns of jump instructions and memory accesses
4. Use of hardware instructions that take different amounts of time to execute depending on operands or results

Some of those side-channels may be detectable over the network, while others may require closer access to the attacked system. With closer access, the attacker may be able to measure power usage, electromagnetic emanations, or sounds and correlate them with specific bits of secret information.

Recent research into network-based side-channel detection has shown that even very small side-channels (of just a few clock cycles) can be reliably detected over the network. The detectability depends on the sample size the attacker is able to collect, not on the size of the side-channel.

6. General RSA operations

This section describes considerations for all padding modes that use private key operation: signing, decryption, or decapsulation.

6.1. General recommendations

As a general rule, all operations that process secret information (be it parts of the private key or parts of the encrypted message) MUST be performed with code that doesn’t have secret data dependent branch instructions, secret data dependent memory accesses, or uses non-constant time machine instructions (which ones are those is architecture dependant, but division is commonly non-constant time).

Special care should be placed around the code that handles the conversion of the numerical representation to the octet string representation in RSA decryption operations.

All operations that use private keys SHOULD additionally employ both base blinding and exponent blinding as protections against leaks inside modular exponentiation code.

6.2. Side-channel free modular exponentiation

The underlying modular exponentiation algorithm MUST be constant time with regards to the exponent both for private key decryption and signing using the private key.

For private key decryption the modular exponentiation algorithm MUST be constant time with regards to the output of the exponentiation.

In case the Chinese remainder theorem optimisation is used, the modular exponentiation algorithm must also be constant time with regards to the used moduli.

6.2.1. General recommendations

It’s especially important to make sure that all values that are secret to the attacker are stored in memory buffers that have sizes determined by the public modulus.

For example, the private exponents should be stored in memory buffers that have sizes determined by the public modulus value, not the numerical values of the exponents themselves.

Similarly, the size of the output buffer for multiplication should always be equal to the sum of buffer sizes of multiplicands. The output size of the modular reduction operation should similarly be equal to the size of the modulus and be dependent on the bit size of the output.

6.2.2. Montgomery ladder

For the modular exponentiation algorithm to be side-channel-free, every step of the calculation MUST NOT depend on the bits of the exponent. In particular, the use of simple square-and-multiply algorithm will leak information about bits of the exponent through the lack of multiplication operation in individual exponentiation steps.

The recommended workaround against it, is the use of the Montgomery ladder construction [Montgomery87].

While that approach ensures that both the square and the multiply operations are performed, the fact that the results of them are placed in different memory locations based on bits of the secret

exponent will provide enough information for an attacker to recover the bits of the exponent. To counteract it, the implementation should ensure that both memory locations are accessed and updated on every step.

Alternatively, it's possible to implement it using the square-and-multiply-always algorithm. In such case, the implementation needs to make sure that memory locations of both the result of the squaring operation and of the multiplication are accessed when preparing the variable for the next step of the algorithm, irrespective of the exponent bit.

6.2.3. Montgomery reduction in multiplication

As multiplication operations quickly make the intermediate values in modular exponentiation large, performing a modular reduction after every multiplication or squaring operation is a common optimisation.

To further optimise the modular reduction, the Montgomery modular multiplication is used for performing the combined multiply-and-reduce operation. The last step of that operation is conditional on the value of the output. A side-channel-free implementation should perform the subtraction in all cases and then copy the result or the first operand of the subtraction based on the sign of the result of the subtraction in the side-channel-free manner.

6.3. Base blinding

As a protection against multiple attacks, it's RECOMMENDED to perform all operations involving the private key with the use of blinding [Kocher96].

It should be noted that for decryption operations the unblinding operation MUST be performed using side-channel-free code that does not leak information about the multiplication and reduction modulo operation results.

To implement base blinding, select a number bb uniformly at random such that it is relatively prime to n and smaller than n.

Compute multiplicative inverse of bb modulo n.

```
bbInv = bb^(-1) mod n
```

In the RSADP() operation, after performing step 1, multiply c by bb mod n. Use the result as new c for all the remaining operations.

Before returning the value m in step 3, multiply it by bbInv mod n.

Note: multiplication by bbInv and reduction modulo n MUST be performed using side-channel-free code with respect to value m.

As calculating multiplicative inverse is expensive, implementations MAY calculate new values of bb and bbInv by squaring them:

```
new bb = bb^2 mod n
new bbInv = bbInv^2 mod n
```

A given pair of blinding factors (bb, bbInv) MUST NOT be used for more than one RSADP() operation.

Unless the multiplication (squaring) and reduction modulo operations are verified to be side-channel-free, it's RECOMMENDED to generate completely new blinding parameters every few hundred private key operations.

6.4. Exponent blinding

To further protect against private key leaks, it's RECOMMENDED to perform the blinding of the used exponents [Kocher96].

When performing the RSADP() operation, the blinding depends on the form of the private key.

If the key is in the first form, the pair (n, d), then the exponent d should be modified by adding a multiple of Euler phi(n): m = c^(d + b*phi(n)) mod n. Where b is a 64-bit long uniform random number.

A new value b MUST be selected for every RSADP() operation.

If the key is the second form, the quintuple (p, q, dP, dQ, qInv) with optional sequence of triplets (r_i, d_i, t_i), i = 3, ..., u, then each exponent used MUST be blinded individually.

1. The m_1 = c^(dP + b_1 * phi(p)) mod p

2. The m_2 = c^(dQ + b_2 * phi(q)) mod q

3. If u > 2, then m_i = c^(d_i + b_i * phi(r_i)) mod (r_i)

Where b_1, b_2, ..., b_i are all uniformly selected random numbers at least 64 bits long (or at least 2 machine word sizes, whichever is greater).

As Euler phi(p) for an argument p that's a prime is equal p - 1, it's simple to calculate in this case.

Note: the selection of random b_i values, multiplication of them by the result of phi() function, and addition to the exponent MUST be performed with side-channel-free code.

Use of smaller blinding factor is NOT RECOMMENDED, as values shorter than 64 bits have been shown to still be vulnerable to side-channel attacks [Bauer12] [Schindler11].

The b_1, b_2, ..., b_i factors MUST NOT be reused for multiple RSADP() operations.

6.5. Modulus blinding

To protect against private key leaks, it’s RECOMMENDED to perform blinding of the used modulus values in the CRT implementations.

No blinding is necessary when the key is in the first form (the pair (n, d)), as the modulus used is public.

If the key is in the second form, the quintuple (p, q, dP, dQ, qInv) with the optional sequence f triplets (r_i, d_i, t_i), i = 3, ..., u, then each modulus used MUST be blinded individually.

1. The m_1 = c^dP mod (g_1 * p)

2. The m_2 = c^dQ mod (g_2 * q)

3. If u > 2, then m_i = c^d_i mod (g_3 * r_i)

Where g_1, g_2, ..., g_i are all uniformly selected random numbers at least 64 bits long (or at least 2 machine word sizes, whichever is greater).

Before step 3 of the original algorithm, reduce the returned value m mod n.

7. Depadding

In case of RSA-OAEP, the padding is self-verifying, thus the depadding operation needs to follow the standard algorithm to provide a safe API to users.

The depadding operation MUST ignore the value of the very first octet of padding and process the remaining bytes as if that first octet was equal to zero.

The RSAES-PKCS-v1_5 encryption scheme is considered deprecated, and should be used only to process legacy data. It MUST NOT be used as part of online protocols or API endpoints.

For implementations that can't remove support for this padding mode it's RECOMMENDED to implement an implicit rejection mechanism that completely hides from the calling code whether the padding check failed or not. This makes the code return errors for significantly fewer amount of ciphertexts, only ciphertexts that are publicly invalid (numerical value larger than modulus or wrong number of bytes for the modulus) will return errors from the decryption operation.

It should be noted that the algorithm MUST be implemented as stated, otherwise in case of heterogeneous environments where two implementations use the same key but implement the implicit rejection differently, it may be possible for the attacker to compare behaviour between the implementations to guess if the padding check failed or not.

The basic idea of the implicit rejection is to prepare a random but deterministic message to be returned in case the standard RSAES-PKCS-v1_5 padding checks fail. To do that, use the private key and the provided ciphertext to derive a static, but unknown to the attacker, random value. It's a combination of the method documented in the TLS 1.2 ([RFC5246]) and the deterministic (EC)DSA signatures ([RFC6979]).

7.1. Implicit Rejection Pseudo-Random Function (IRPRF)

For the calculation of the random message for implicit rejection we define a Pseudo-Random Function (PRF) as follows:

IRPRF (KDK, label, length)

Input:

KDK the key derivation key

label a label making the output unique for a given KDK

length requested length of output in octets

Output: derived key, an octet string

Steps:

1. If KDK is not 32 octets long, or if length is larger than 8192 return error and stop.

2. The returned value is created by concatenation of subsequent calls to a SHA-256 HMAC [RFC2104] function with the KDK as the HMAC key and following octet string as the message:

   P_i = I || label || bitLength

3. Where the I is an iterator value encoded as two octet long big endian integer, label is the passed in label, and bitLength is the length times 8 (to represent number of bits of output) encoded as two octet big endian integer. The iterator is initialised to 0 on first call, and then incremented by 1 for every subsequent HMAC call.

4. The HMAC is iterated until the concatenated output is shorter than length

5. The output is the length left-most octets of the concatenated HMAC output

7.2. Implicit rejection

For implementations that cannot remove support for the RSAES-PKCS-v1_5 encryption scheme nor provide a usage-specific API, it's possible to implement an implicit rejection algorithm as a protection measure. It should be noted that implementing it correctly is hard, thus, it's RECOMMENDED instead to disable support for RSAES-PKCS-v1_5 padding instead.

To implement implicit rejection, the RSAES-PKCS1-V1_5-DECRYPT operation from section 7.2.2 of RFC 8017 needs to be implemented as follows:

1. Length checking: If the length of the ciphertext C is not k octets (or if k < 11), output "decryption error" and stop.

2. RSA decryption:

   a. Convert the ciphertext C to an integer ciphertext representative c:

      c = OS2IP (C).

   b. Apply the RSADP decryption primitive to the RSA private key K and the ciphertext representative c to produce an integer message representative m:

      m = RSADP (K, c).

Note: the RSADP MUST be constant-time with respect of message m.

If RSADP outputs "ciphertext representative out of range" (meaning that c >= n), output "decryption error" and stop.

c. Convert the message representative m to an encoded message EM of length k octets:

EM = I2OSP (m, k).

Note: I2OSP MUST be constant-time with respect of m.

3. Derivation of alternative message

1. Derive the Key Derivation Key (KDK)

a. Convert the private exponent d to a string of length k octets:

D = I2OSP (d, k).

Note: when the private key is provided with CRT parameters, the value d SHOULD NOT be recomputed but rather used as-is, as there exist conflicting algorithms that may result in numerically different but equivalent values of d.

b. Hash the private exponent using the SHA-256 algorithm:

DH = SHA256 (D).

Note: This value MAY be cached between the decryption operations, but MUST be considered private-key equivalent.

c. Use the DH as the SHA-256 HMAC key and the provided ciphertext C as the message. If the ciphertext C is not k octets long, it MUST be left padded with octets of value zero.

KDK = HMAC (DH, C, SHA256).

2. Create the candidate lengths and the random message

a. Use the IRPRF with key KDK, "length" as six octet label encoded with UTF-8, to generate 256 octet output. Interpret this output as 128 two octet long big-endian numbers.

CL = IRPRF (KDK, "length", 256).

b. Use the IRPRF with key KDK, "message" as a seven octet label encoded with UTF-8 to generate k octet long output to be used as the alternative message:

AM = IRPRF (KDK, "message", k).

3. Select the alternative length for the alternative message.

Note: this must be performed in side-channel-free way.

a. Iterate over the 128 candidate CL lengths. For each candidate length zero out high-order bits so that they have the same bit length as the maximum valid message size (k - 11).

b. Select the last length that’s not larger than (k - 11), use 0 if none are. Save it as AL.

4. EME-PKCS1-v1_5 decoding: Separate the encoded message EM into a padding string PS consisting of nonzero octets and a message M as

EM = 0x00 || 0x02 || PS || 0x00 || M.

If the first octet of EM does not have hexadecimal value 0x00, if the second octet of EM does not have hexadecimal value 0x02, if there is no octet with hexadecimal value 0x00 to separate PS from M, or if the length of PS is less than 8 octets, the check variable must remember if any of those checks failed. Irrespective of the check variable value, the code should also return length of message M: L. If there is no octet with hexadecimal value 0x00 to separate PS from M, then L should equal 0.

Note: All those checks MUST be performed irrespective if previous checks failed or not. A common technique for that is to have a check variable that is OR-ed with the results of subsequent checks.

5. Decision which message to return: in case the check variable is set, the code should return the last AL octets of AM, in case the check variable is unset the code should return the last L octets of EM.

   Note: The decision which length to use MUST be performed in side-channel-free manner. While the length of the returned message is not considered sensitive, the read memory location is. As such, when returning message M both EM and AM memory locations MUST be read.

7.3. Security analysis

The reasons for this solution being secure and safe to deploy are multi-layered, but they boil down to three things: the sizes of returned messages, the hiding of the padding check result, and the determinism of the decryption.

First, let’s consider an API that requires that the decrypted messages are keys of specific size, then an algorithm that may return various message lengths may look problematic. But, if the calling code cannot handle unexpected lengths, then the attacker can simply exploit it by directly encrypting those sizes, and they don’t need to resort to performing a Bleichenbacher attack. As such, the algorithm doesn’t create any new attack vector.

Secondly, for protection against a potential Bleichenbacher attack we need to hide the information if the decrypted plaintext (the result of applying the RSADP() primitive) starts with specific bytes. The classic oracle for that, is the use of the RSAES-PKCS-v1_5 padding check: accepted ciphertexts definitely start with bytes 0x00 0x02. By making the decryption call always return a message, the attacker can’t easily differentiate plaintexts starting with specific bytes from ones that do not just by presence of an error in decryption.

Thirdly, when performing the actual Bleichenbacher attack (when generating ciphertexts according to the attack), the attacker doesn’t know if any particular ciphertext decrypts to a message or not. But crucially, if the ciphertext, by chance, decrypts to a plaintext that is correctly formatted according to RSAES-PKCS-v1_5 padding, then that particular ciphertext will always decrypt to that message. If an implementation would return random messages in case of a padding check failure, then further processing of them might reveal if those messages are always the same or always different. For example, using them as keys for AES-128-CBC with PKCS #7 padding will cause that padding check to randomly pass. By making the returned message unpredictable but deterministic, we remove the ability for the attacker to tell if the padding check passed or not.

8. Safe API

Any API that returns an error in RSAES-PKCS-v1_5 padding check in a substantially different manner than a successful decryption (e.g., raising an exception, returning a null pointer, returning a different type of structure), is vulnerable to side-channel attacks. Performing all actions in a way that doesn't leak the status of the padding check includes the API provided to 3rd party code. In particular, if the RSA decryption implementation doesn't implement implicit rejection, then all three pieces of information: the padding check, the length of returned message, and the value of the message are sensitive information, useful in mounting an attack.

9. Common Pitfalls

While there are infinite ways to implement those algorithms incorrectly, few ideas to work-around side-channel attacks are often repeated. We list few of them as examples of approaches that don't work and thus MUST NOT be used.

9.1. Random delays

A commonly proposed workaround for timing attacks is to add a random delay to the processing of encrypted data. This is unlikely to be successful, as the statistical methods used to detect the presence of differences inherently filter out noise in measurements. For such mitigation to be effective in practice (raise attacker's work factor to gain meaningful safety margin), it would need to be in the order of at least a few seconds if not tens of seconds. Effective sizes and distributions of delays have not been the subject of extensive study.

It should also be noted that such a delay would need to be applied in addition to the regular mitigation tactic (generate random key, and use it in case RSAES-PKCS-v1_5 padding checks fail). That is, it needs to be implemented in the calling code, not in the depadding code.

At the same time, performing a simple wait masks only the timing side-channel. It doesn't mitigate other side-channels, like ones that depend on power usage or memory access pattern.

9.2. Returning random value from the decryption API

A simpler variant of the proposed implicit rejection algorithm, where the decryption API returns a random message in case the padding check fails or the decrypted message has unexpected length, as described in [RFC5246], isn't a universal mitigation.

In case of a Bleichenbacher like attack, the attacker is trying to differentiate two classes of ciphertexts: (1) ones that decrypt to a message of a specific size, and (2) ones that have invalid padding or decrypt to a message of unexpected size. That translates to two kind of values being returned to the calling code: either the same message always for every decryption, or a different message for every decryption.

If that message is later used in any operation that is not side-channel-free (key derivation, symmetric encryption padding removal, message parsing), then the attacker will be able to observe two kinds of behaviour: consistent one for static messages and an erratic one for randomly generated messages.

Use of such a message as a key with block cipher modes that require padding, like AES-CBC with PKCS #7 padding, is particularly leaky as it has about a 1 in 256 chance of successful decryption with random key.

The simple implicit rejection with random message may be implemented safely in TLS 1.2 because both static messages and randomly generated messages are mixed together with the random nonce generated by the server (the "random" field in the ServerHello message), thus the attacker is unable to differentiate if the randomness originates in the nonce or in the randomisation of the message.

10. Key generation

Multiple attacks against the generation of the parameters used in the RSA cryptosystem exist. Those are considered to be outside the scope of this document.

11. Deprecated Algorithms

Current protocol deployments MUST NOT use encryption with RSAES-PKCS-v1_5 padding. Support for RSAES-PKCS-v1_5 SHOULD be disabled in default configuration of any implementation of RSA cryptosystem. All new protocols MUST NOT specify RSAES-PKCS-v1_5 as a valid encryption padding for RSA keys.

The RSAES-PKCS-v1_5 padding MUST NOT be supported in any part of online (network-accessible) protocols or API endpoints.

12. IANA Considerations

This memo includes no request to IANA.

13. Security Considerations

This whole document specifies security considerations for RSA implementations.

14. References

14.1. Normative References

[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>.

14.2. Informative References

[Aciicmez05] Acimez, O., Schindler, W., and . K. Ko, "Improving Brumley and Boneh timing attack on unprotected SSL implementations", Proceedings of the 12th ACM conference on Computer and communications security CCS ’05, 2005, <https://doi.org/10.1145/1102120.1102140>.

[Aciicmez07] Acimez, O. and W. Schindler, "A Major Vulnerability in RSA Implementations due to MicroArchitectural Analysis Threat", Cryptology ePrint Archive Paper 2007/336, 2007, <https://eprint.iacr.org/2007/336>.

[Aciicmez08] Acimez, O. and W. Schindler, "A Vulnerability in RSA Implementations Due to Instruction Cache Analysis and Its Demonstration on OpenSSL", Lecture Notes in Computer Science vol 4964, 2007, <https://doi.org/10.1007/978-3-540-79263-5_16>.

[Bauer12] Bauer, S., "Attacking Exponent Blinding in RSA without CRT", Lecture Notes in Computer Science vol 7275, 2012, <https://doi.org/10.1007/978-3-642-29912-4_7>.

[Bleichenbacher98] Bleichenbacher, D., "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1", Lecture Notes in Computer Science vol 1462, <https://doi.org/10.1007/BFb0055716>.

[Boeck18] Bck, H., Somorovsky, J., and C. Young, "Return Of Bleichenbacher's Oracle Threat (ROBOT)", 27th USENIX Security Symposium USENIX Security 18, 2018, <https://www.usenix.org/conference/usenixsecurity18/presentation/bock>.

[Brumley03] Brumley, D. and D. Boneh, "Remote timing attacks are practical", Computer Networks Volume 48, Issue 5, 2003, <https://doi.org/10.1016/j.comnet.2005.01.010>.

[Dhem98] Dhem, J., Koeune, F., Leroux, P., Mestr, P., Quisquater, J., and J. Willems, "A Practical Implementation of the Timing Attack", Lecture Notes in Computer Science vol 1820, 1998, <https://doi.org/10.1007/10721064_15>.

[Kario23] Kario, H., "Everlasting ROBOT: the Marvin Attack", 28th European Symposium on Research in Computer Security , 2023, <https://eprint.iacr.org/2023/1442>.

[Kocher96] Kocher, P. C., "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems.", Lecture Notes in Computer Science vol 1109, 1996, <https://doi.org/10.1007/3-540-68697-5_9>.

[Montgomery87] Montgomery, P. L., "Speeding the Pollard and elliptic curve methods of factorization", Mathematics of Computation vol 48, 1987, <https://doi.org/10.2307%2F2007888>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[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>.

[Schindler00] Schindler, W., "A Timing Attack against RSA with the Chinese Remainder Theorem", Lecture Notes in Computer Science vol 1965, 2000, <https://doi.org/10.1007/3-540-44499-8_8>.

[Schindler01] Schindler, W., Koeune, F., and J. Quisquater, "Improving Divide and Conquer Attacks against Cryptosystems by Better Error Detection / Correction Strategies.", Lecture Notes in Computer Science vol 2260, 2001, <https://doi.org/10.1007/3-540-45325-3_22>.

[Schindler11] Schindler, W. and K. Itoh, "Exponent Blinding Does Not Always Lift (Partial) Spa Resistance to Higher-Level Security", Lecture Notes in Computer Science vol 6715, 2011, <https://doi.org/10.1007/978-3-642-21554-4_5>.

[Schindler14] Schindler, W. and A. Wiemers, "Power attacks in the presence of exponent blinding.", Journal of Cryptographic Engineering 4, 2014, <https://doi.org/10.1007/s13389-014-0081-y>.

[SHA2] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

Appendix A. Acknowledgements

We would like to thank Stefan Berger for the discussions that led to the development of the implicit rejection algorithm. We'd like to thank to all the people that provided feedback to the document in the CFRG mailing list.

We also thank Scott Fluhrer, Jon Callas, and Francois Grieu for the detailed review and notes.

Appendix B. Test Vectors

B.1. 2048-bit key

Test vectors with an RSA key with a 2048-bit modulus.

[Note to RFC Editor: please remove this paragraph] For verification of test vectors see: https://github.com/tlsfuzzer/tlslite-ng/blob/master/unit_tests/test_tlslite_utils_rsakey.py#L1694

B.1.1. Private key

Private key to use for decryption of following test data

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDIzIOXFAmNpWyq
I2QPk9yJl8FjcpaPwbDG31ETwclOiyHkitIpfmVBkBG05tj15zsbeLJXQAMh0e9r
YC1OyM6NFByUkF60rTBmOaSSBlNLbn8mB0I+l9/9EzyI1yE5ne+8fpbM3L1/Oq4f
6JJxK/tJKYF9URZmRAofrLeiCPXqFlkQrdij8tSXICNgzLYyAk8NBxacGRjzFveU
sUOu9U7IdSKkwCl4+WiZgL/79knDB+gYGb/4hAljjUi9lL4VK1n/ZJ+gvWKdD/oY
E8Or9LVr08LqVGXf+hRYkpKp2KJK0mvn7gUQdBtjgtQ8g9W/pApGYT0GK+RFUX28
rwy04adpAgMBAAECggEAFFUBDg8tWHZjpmam/xzNu/Dt2BAGRtAqAjkikImSxK05
5VZZKXJu9lCMOnEVjvC2/3UdOdB1gLstLwYyEEQtBgP/UNu9ezX+LJuxmkehr4Wk
wkkB4CyotYt5GbIO3zKqz79RrbS8S2G5t+loyqTVcPcO8Y2AYyKIk+R9Q578p5Ml
m88s0Qij2GiM3weOeseZlp8jOdLB9SK5aWhGKakzuq7CaBYl6rhPTlb0RH6diPua
GZz3ECPg4lexREGzPITTvGfKgDHSYSYYEDp6CkCEQmL3XYiQzWFuUfkDVIj9bgmd
6P9tZaT/EYJUgHyfWNL7uouhUdyMaL40nJd6IE4EwQKBgQD49a1rqCiTG+pFm4o/
bcBB0jSCQJwlcelj8x90hgKiVjcbOIPtRZ7PlwUmRZ7dFuBVIvWkXZR1Gy7C2vJy
x/iBalLADRgIAXFjTaiZ15cyIvUbk3YwVIaWqffYwkpZSXwe/NRVz7l+6G0rbTSX
KzMv2jA/BJmbTra1zAuzPndh3QKBgQDOei47SakLljMKEtxoK9+9+66N1twDthR6
771XV0Pw9tpNhiNQYbca/ZytLTQCXlashrD3dD6zXhrLyiN4lUJEZbcG7SIXXlcY
yMcLZwPqj2tRD5Rb5I5aNrs8PJFzK1id/AXXLYCQMZRFK9ohNIZH7HKUPxGoRuYv
rr6OtTaw/QKBgHb+FfGK4jnN8d9rRFykvGu5aNeIwhkzpPXc0oADPWcSBizAim3y
BMH70L5GMHRD5t1KZFY3VCnU4DjKJW+vHJvekcaxe3b4GZX5HEjLvrx78ONJTAg1
nk6M1qWH17ltYiH9fg+1xVdfCC7ld2l5gHGyu7SjIjgVG0cxS7ZUeQMRAoGBAJmI
SLBVSZoQCcvH0pSzax/98gIObnNkBT6U3hoADck0BYf34nJ29ozfYI11O2M3ewO2
9AhNLAJ8SziWCmIzup7Zc4t28Q6nW+RWB4v3AfZ8xrPz/cGG5kM2x2s3LoCRDsgL
CtzCPQL7muEEhqKCSAdbTqflbd/POILkUVYUcaKRAoGAZDv3RkKffYNmegZTAhNH
77/AXmNR+CGp3rtg4OzN5QBa2ensMeVY9+ksKTKOdFadfO98dMq8KzVe1AGhoJFL
Tjy7BkhOWBlgURae0UyqLvpuoETgVNJhRMwWKcVQEFWKBOEz9Et8JE2sJb+RPFe4
kO5J9UglnNY0BP72hZ3Pl1o=
-----END PRIVATE KEY-----
```

B.1.2. Valid

Hex encoded ciphertext:

```
8bfe264e85d3bdeaa6b8851b8e3b956ee3d226fd3f69063a86880173a273d9f2
83b2eebdd1ed35f7e02d91c571981b6737d5320bd8396b0f3ad5b019daec1b0a
ab3cbbc026395f4fd14f13673f2dfc81f9b660ec26ac381e6db3299b4e460b43
fab9955df2b3cfaa20e900e19c856238fd371899c2bf2ce8c868b76754e5db3b
036533fd603746be13c10d4e3e6022ebc905d20c2a7f32b215a4cd53b3f44ca1
c327d2c2b651145821c08396c89071f665349c25e44d2733cd9305985ceef643
0c3cf57af5fa224089221218fa34737c79c446d28a94c41c96e4e92ac53fbcf3
84dea8419ea089f8784445a492c812eb0d409467f75afd7d4d1078886205a066
```

ASCII encoded decrypted message:

```
lorem ipsum dolor sit amet
```

B.1.3. Valid empty

Hex encoded ciphertext:

```
443ad9c5f00a1f4e9601717d274aacc93a824cb4d99b3c6a42e2b017e52e0184
43bd77d34a80703cd9b6acf523cd3b2cd1fea31940a68fba828836f1c3ed2fef
071e95e0922ff0f47d0e81dacc13ecdeda3db6476f41e5b3f9ccfcfdf9155800
7b68ffbbe5e93bb088f1e4f0bb39bc7d8600b38930ecd00a341d8cc76955837e
fff0f0797c4b46fb1b375bba49bcdc877f39aaadb045c56b836072383eec6627
ae280ad4f9a45d6e5b4cc7cf61d42b194ff0b9c9167c621e5380d8333e3b7f20
c8d564c9ec6c2805f77c0146adea2f688a943b67ce8889f4a6353e6396d551c3
6a6dbf19359a825d14b69ccc4fd747cf14a1ca8578d7f0a67dc14b37f5e17ca3
```

The result of decryption is a message of length 0.

B.1.4. Valid with ciphertext starting with a zero byte

Hex encoded ciphertext:

```
00a2e8f114ea8d05d12dc843e3cc3b2edc8229ff2a028bda29ba9d55e3cd0291
1902fef1f42a075bf05e8016e8567213d6f260fa49e360779dd81aeea3e04c2c
b567e0d72b98bf754014561b7511e083d20e0bfb9cd23f8a0d3c88900c49d2fc
d5843ff0765607b2026f28202a87aa94678aed22a0c20724541394cd8f44e373
eba1d2bae98f516c1e2ba3d86852d064f856b1daf24795e767a2b90396e50743
e3150664afab131fe40ea405dcf572dd1079af1d3f0392ccadcca0a12740dbb2
13b925ca2a06b1bc1383e83a658c82ba2e7427342379084d5f66b544579f0766
4cb26edd4f10fd913fdbc0de05ef887d4d1ec1ac95652397ea7fd4e4759fda8b
```

ASCII encoded message:

```
lorem ipsum
```

B.1.5. Invalid decrypting to empty

Hex encoded ciphertext:

```
20aaa8adbbc593a924ba1c5c7990b5c2242ae4b99d0fe636a19a4cf754edbcee
774e472fe028160ed42634f8864900cb514006da642cae6ae8c7d087caebcfa6
dad1551301e130344989a1d462d4164505f6393933450c67bc6d39d8f5160907
cabc251b737925a1cf21e5c6aa5781b7769f6a2a583d97cce008c0f8b6add5f0
b2bd80bee60237aa39bb20719fe75749f4bc4e42466ef5a861ae3a92395c7d85
8d430bfe38040f445ea93fa2958b503539800ffa5ce5f8cf51fa8171a91f36cb
4f4575e8de6b4d3f096ee140b938fd2f50ee13f0d050222e2a72b0a3069ff3a6
738e82c87090caa5aed4fcbe882c49646aa250b98f12f83c8d528113614a29e7
```

The result of decryption is a message of length 0.

B.1.6. Invalid decrypting to max size

Hex encoded ciphertext:

```
48cceab10f39a4db32f60074feea473cbcdb7accf92e150417f76b44756b190e
843e79ec12aa85083a21f5437e7bad0a60482e601198f9d86923239c8786ee72
8285afd0937f7dde12717f28389843d7375912b07b991f4fdb0190fced8ba665
314367e8c5f9d2981d0f5128feeb46cb50fc237e64438a86df198dd0209364ae
3a842d77532b66b7ef263b83b1541ed671b120dfd660462e2107a4ee7b964e73
4a7bd68d90dda61770658a3c242948532da32648687e0318286473f675b412d6
468f013f14d760a358dfcad3cda2afeec5e268a37d250c37f722f468a70dfd92
d7294c3c1ee1e7f8843b7d16f9f37ef35748c3ae93aa155cdcdfeb4e78567303
```

Hex encoded decrypted message:

```
22d850137b9eebe092b24f602dc5bb7918c16bd89ddbf20467b119d205f9c2e4
bd7d2592cf1e532106e0f33557565923c73a02d4f09c0c22bea89148183e6031
7f7028b3aa1f261f91c979393101d7e15f4067e63979b32751658ef769610fe9
7cf9cef3278b3117d384051c3b1d82c251c2305418c8f6840530e631aad63e70
e20e025bcd8efb54c92ec6d3b106a2f8e64eeff7d38495b0fc50c97138af4b1c
0a67a1c4e27b077b8439332edfa8608dfeae653cd6a628ac550395f7e74390e4
2c11682234870925eeaa1fa71b76cf1f2ee3bda69f6717033ff8b7c95c9799e7
a3bea5e7e4a1c359772fb6b1c6e6c516661dfe30c3
```

B.1.7. Invalid with null padded ciphertext

Ciphertext that decrypts to an invalid plaintext, with the ciphertext starting with a zero byte

Hex encoded ciphertext:

006f89db685c0a132700c6a17f88a37a6635d0ab89de4c45dc09736c891ca5bf
3401ce34c6e5d51e94ed2f518857ddc12d9f9f9e68e01cdc30d86ae5dd83988c
0c46a8e39daa1b328a23def551d67fa1964fb15242c83ddd7dd5b1aec720a391
d0b86cb16cf4d3c466850c3df88a3ed85993900d1287a0c90c4b04d34ba29e59
967661f3f10e0c998f64e14e777e8e81371eca5318b4e0b53414292130c82147
7c51e2bff844836ab10dff293d82e4f40d345968ef268c92ed0bc238f31d50f4
d3f759c23964923e135d15527556410fbd2c451d6a2aa852dc88b01139c6fdd8
26736d8cd3780601b2977b09c080bd8c0fa471606ad59f053ad33d9eeb905f20

Hex encoded decrypted message:

2b5dd72df3cae37f1aef

B.1.8. Invalid with second to last length

The length of the synthetic message comes from second to last value from the PRF output as the last value is too long for this key size.

Hex encoded ciphertext:

1439e08c3f84c1a7fec74ce07614b20e01f6fa4e8c2a6cffdc3520d8889e5d9a
950c6425798f85d4be38d300ea5695f13ecd4cb389d1ff5b82484b494d6280ab
7fa78e645933981cb934cce8bfcd114cc0e6811eefa47aae20af638a1cd163d2
d3366186d0a07df0c81f6c9f3171cf3561472e98a6006bf75ddb457bed036dcc
e199369de7d94ef2c68e8467ee0604eea2b3009479162a7891ba5c40cab17f49
e1c438cb6eaea4f76ce23cce0e483ff0e96fa790ea15be67671814342d0a23f4
a20262b6182e72f3a67cd289711503c85516a9ed225422f98b116f1ab080a80a
bd6f0216df88d8cfd67c139243be8dd78502a7aaf6bc99d7da71bcdf627e7354

Hex encoded decrypted message:

0f9b

B.1.9. Invalid first byte of padding

Otherwise valid plaintext, but with first byte equal to 0x01 instead of 0x00.

Hex encoded ciphertext:

9b2ec9c0c917c98f1ad3d0119aec6be51ae3106e9af1914d48600ab6a2c0c0c8
ae02a2dc3039906ff3aac904af32ec798fd65f3ad1afa2e69400e7c1de81f572
8f3b3291f38263bc7a90a0563e43ce7a0d4ee9c0d8a716621ca5d3d081188769
ce1b131af7d35b13dea99153579c86db31fe07d5a2c14d621b77854e48a8df41
b5798563af489a291e417b6a334c63222627376118c02c53b6e86310f728734f
fc86ef9d7c8bf56c0c841b24b82b59f51aee4526ba1c4268506d301e4ebc498c
6aebb6fd5258c876bf900bac8ca4d309dd522f6a6343599a8bc3760f422c10c7
2d0ad527ce4af1874124ace3d99bb74db8d69d2528db22c3a37644640f95c05f

Hex encoded message:

a1f8c9255c35cfba403ccc

B.1.10. Invalid second byte of padding

Otherwise valid plaintext, but with the second byte equal to 0x01 instead of 0x02.

Hex encoded ciphertext:

782c2b59a21a511243820acedd567c136f6d3090c115232a82a5efb0b178285f
55b5ec2d2bac96bf00d6592ea7cdc3341610c8fb07e527e5e2d20cfaf2c7f23e
375431f45e998929a02f25fd95354c33838090bca838502259e92d86d568bc2c
db132fab2a399593ca60a015dc2bb1afcd64fef8a3834e17e5358d822980dc44
6e845b3ab4702b1ee41fe5db716d92348d5091c15d35a110555a35deb4650a5a
1d2c98025d42d4544f8b32aa6a5e02dc02deaed9a7313b73b49b0d4772a3768b
0ea0db5846ace6569cae677bf67fb0acf3c255dc01ec8400c963b6e49b106772
8b4e563d7e1e1515664347b92ee64db7efb5452357a02fff7fcb7437abc2e579

Hex encoded decrypted message:

e6d700309ca0ed62452254

B.1.11. Invalid with zero length PS

Hex encoded ciphertext:

0096136621faf36d5290b16bd26295de27f895d1faa51c800dafce73d001d607
96cd4e2ac3fa2162131d859cd9da5a0c8a42281d9a63e5f353971b72e36b5722
e4ac444d77f892a5443deb3dca49fa732fe855727196e23c26eeac55eeced826
7a209ebc0f92f4656d64a6c13f7f7ce544ebeb0f668fe3a6c0f189e4bcd5ea12
b73cf63e0c8350ee130dd62f01e5c97a1e13f52fde96a9a1bc9936ce734fdd61
f27b18216f1d6de87f49cf4f2ea821fb8efd1f92cdad529baf7e31aff9bff407
4f2cad2b4243dd15a711adcf7de900851fbd6bcb53dac399d7c880531d06f25f
7002e1aaf1722765865d2c2b902c7736acd27bc6cbd3e38b560e2eecf7d4b576

Hex encoded decrypted message:

ba27b1842e7c21c0e7ef6a

B.1.12. Invalid with null byte at the eighth byte of padding

This creates a plaintext with PS of 7 bytes

Hex encoded ciphertext:

```
a7a340675a82c30e22219a55bc07cdf36d47d01834c1834f917f18b517419ce9
de2a96460e745024436470ed85e94297b283537d52189c406a3f533cb405cc6a
9dba46b482ce98b6e3dd52d8fce2237425617e38c11fbc46b61897ef200d01e4
f25f5f6c4c5b38cd0de38ba11908b86595a8036a08a42a3d05b79600a97ac18b
a368a08d6cf6ccb624f6e8002afc75599fba4de3d4f3ba7d208391ebe8d21f82
82b18e2c10869eb2702e68f9176b42b0ddc9d763f0c86ba0ff92c957aaeab76d
9ab8da52ea297ec11d92d770146faa1b300e0f91ef969b53e7d2907ffc984e9a
9c9d11fb7d6cba91972059b46506b035efec6575c46d7114a6b935864858445f
```

Hex encoded decrypted message:

```
63cb0bf65fc8255dd29e17
```

B.1.13. Invalid with padding separator missing

Hex encoded ciphertext:

```
3d1b97e7aa34eaf1f4fc171ceb11dcfffd9a46a5b6961205b10b302818c1fcc9
f4ec78bf18ea0cee7e9fa5b16fb4c611463b368b3312ac11cf9c06b7cf72b54e
284848a508d3f02328c62c2999d0fb60929f81783c7a256891bc2ff4d91df2af
96a24fc5701a1823af939ce6dbdc510608e3d41eec172ad2d51b9fc61b4217c9
23cadcf5bac321355ef8be5e5f090cdc2bd0c697d9058247db3ad613fdce87d2
955a6d1c948a5160f93da21f731d74137f5d1f53a1923adb513d2e6e1589d44c
c079f4c6ddd471d38ac82d20d8b1d21f8d65f3b6907086809f4123e08d86fb38
729585de026a485d8f0e703fd4772f6668febf67df947b82195fa3867e3a3065
```

Hex encoded decrypted message:

```
6f09a0b62699337c497b0b
```

B.2. 2049-bit key

Test vectors that use an uncommon modulus size: 2049-bits.

B.2.1. Private key

Private key to use for decryption of following test data

```
-----BEGIN PRIVATE KEY-----
MIIEvwIBADANBgkqhkiG9w0BAQEFAASCBKkwggSlAgEAAoIBAQFV+IlVahd18cen
eGpQsYvCjJ6Ybt5WZ8qzm4QSTpDrp1wdsIOsPkQ7upTcI1YPdeOoFpOipDvcdCbY
xOr+aMhd4P51f25Ju57UR+YCQwgA27BM6yLn+lehjTOPtmAmzbRn5wzAQOfTZ+9A
PHvx499iRlAJRjHyHq/S+1vJFf8EN5rNERL3MsC0ZgfBeNOKIPUu2lCfL5wEBdUQ
aegMz5QVVNBHBGdQXDz1QeoIl9/J9ADOyymP/HUzctn2kzrxdMxA7ZbUZwMXM7l/
jN3T+SvDoD6oV2xBfyQAe15PdQEQW1RN6frc3/rfmN+0uwW4GZ8/haz9kfepoJS5
o4P1BJCXAgMBAAECggEBARnCs/UKetYVJnnX/1EJWKwtjKbwAoWS8zLVWhZzYXio
5n8X5wXOMA4+h1RyUQBgE/l00KPbSe80TKWiajTARQcE0OQi4M4jppQlwV/vtvJu
EG7v9kzIuddELk2k6MhQCOrrNlhZoilPo5N7wmvlYzLn2B4sFg72NcxSiqe+VeYz
pyPbweFrop5SsprvL55WVP3AZmuw/CVKy+gOY4dPD18CB4Ljydz8JSDQycSntjTk
UD+7ST4ar+6z+IvXoTOYcl2ub+OZ53XNXUzwn8g4NHxMmNqxpIg8zmIFE2Fa+qEK
YzaObXt530FmqxYnOe9RWkQC7h4GAcWlW8cd8OMO34ECgYEBiPaTYPAeGNmi3ilS
U9JSwx5Eds6l/3v4QT33/eNWUjzcl2gF+E/A3ex3DfBs7QZcgRNIdUs0avFpdWh3
/Ts9VoaCyHh9CzFO9qxn1l6BMzmLYqCDwPh2XFrUDVqB+bvc4lJ+1+lQCMsQKctM
q9H56b7fwobJZVIlXafqsZIXjvcCgYEA3sfPEdreg6TEPS+AGX8h/V1G/Ve0MfRP
6Bod439qCR/8BGTtlx3IUIg1rebMX1ZvOWVhOos2eYyS5uI/Uu+QfpVn40G+vFM3
GJYl+76rHzt7P5L/smgebvWnhKjC1498LYmqqiTSztvQZoHP5lw2x7+jxboTUWIi
LvXC6RTJg2ECgYASCT86c8rt2Q9gowTkVAL4casyyMlVsJr0Y6O+Q3Dy1VhKmrur
af2wMepE+YQGXQRh6ECrIYiGYA43FVRsiwuFrSbTjLQwj1LQf5lEfZHwh/Od00A4
2y6TjpetBTpx++1ndeHchxjlTmyvfmVGfZy63cfnZchYniyY39wlyk7KgQKBgQCM
zmE0ec+WCPf3bCRc+Ru0ldYeneZIhJBUtN0bQxbz+YFCDcCVeL95Fv5Gkc+ummTm
NAuGAyNFI/Jdd7ZqZvw+5ZOp8Y3qXfY+1fff650gummlvvVZ/7DslNtyX2/26ruj
1JVHwMp08D4B7B5JDROaoKeUe41mLM5KPA8bXoYXQQKBgQD5lZs0xLyozkiIeBsx
sem12K310NPj7VReg2fT+FRbpkQyuIcwNe+IHCvN4A0YCfksQNh4N7XE+az0iza4
3FOllWGkVlI0AtHo+jrzAOVMkbY+bO4G/m3mZvOSlYKgPh9FTneJ+weBpNb7tSbv
iBYh/R6s0hRm5M3Ziu0Q9OdveQ==
-----END PRIVATE KEY-----
```

B.2.2. Valid

Hex encoded ciphertext:

```
013300edbf0bb3571e59889f7ed76970bf6d57e1c89bbb6d1c3991d9df8e65ed
54b556d928da7d768facb395bbcc81e9f8573b45cf8195dbd85d83a59281cddf
4163aec11b53b4140053e3bd109f787a7c3cec31d535af1f50e0598d85d96d91
ea01913d07097d25af99c67464ebf2bb396fb28a9233e56f31f7e105d71a23e9
ef3b736d1e80e713d1691713df97334779552fc94b40dd733c7251bc522b673d
3ec9354af3dd4ad44fa71c0662213a57ada1d75149697d0eb55c053aaed5ffd0
b815832f454179519d3736fb4faf808416071db0d0f801aca8548311ee708c13
1f4be658b15f6b54256872c2903ac708bd43b017b073b5707bc84c2cd9da70e9
67
```

ASCII encoded decrypted message:

```
lorem ipsum
```

B.2.3. Valid empty

Hex encoded ciphertext:

```
00cc52e83755a4526fea5e6245045063843 0a84a5878fd12c2a571f33c55729c
fab6e35c2e1703c452cff65731249460919aeb1b40084bdef573407851e48b3c
72923e48d5c4f3e80990c462bc291a3e635515636ab9ebeb317ca0d75b04b80c
17e2f4851f8929f72c9bea4ec4a6a1fbc5155837813567062d6b4b2a6b6e40be
545d25da39b08c52f3543e2f2cdfa314832dcbf475fcbb8d3565a64bb09b55f9
22e6ec6cd8bb5203a11e2fa0c1b383674c4f0b63acd78f3690e3a16ad1b71f6c
fe48c56533e2ae42b1393b2d156c2323272490a574ce4f14055249b6a34c3e08
d4a417039450910ec34bd5f08eb06078f51bdd6e50334ee64c9695a5bde52938
e3
```

The result of decryption is a message of length 0.

B.2.4. Valid with ciphertext starting with a zero byte

Hex encoded ciphertext:

```
0002aadf846a329fadc6760980303dbd87bfadfa78c2015ce4d6c5782fd9d3f1
078bd3c0a2c5bfbdd1c024552e5054d98b5bcdc94e476dd280e64d6500893265
42ce7c61d4f1ab40004c2e6a88a883613568556a10f3f9edeab67ae8dddc1e6b
0831c2793d2715de943f7ce34c5c05d1b09f14431fde566d17e76c9feee90d86
a2c158616ec81dda0c642f58c0ba8fa4495843124a7235d46fb4069715a51bf7
10fd024259131ba94da73597ace494856c94e7a3ec261545793b0990279b15fa
91c7fd13dbfb1df2f221dab9fa9f7c1d21e48aa49f6aaecbabf5ee76dc6c2af2
317ffb4e303115386a97f8729afc3d0c89419669235f1a3a69570e0836c79fc1
62
```

ASCII encoded message:

```
lorem ipsum
```

B.2.5. Invalid decrypting to empty

Hex encoded ciphertext:

```
0128a1f7837e53b21ee37f0b4d08c76180305d5d854a1bcf3885471610646795
f1e4c85ce7fce0f71ac3504598afdfc26792dea8ac55c7da10f96d26236ae652
b282459d679ec84847d523f07213e81d1c713fb159eded43112eab68b610e3f8
71d9c0009fde783ad7bcdca5568f7a86a716be6b96219c34b061f68718abad7c
947ed107097dc68341b865d73f2e857a345f5cf05c53bb2899d2895565009125
c7b5fe1c35a73c03bb0f59e7faf381c784988bb71194307ee9a8ac122990fabb
5cc1fd877aaa79039ac163d084c7ee1642aaf05befb9d7ed0e29558f11f0708c
8e83f804f92fd41310a6fd21d91c3ceb88ceee3e424a3fcdda57fe3abb8b7bae
7d
```

The result of decryption is a message of length 0.

B.2.6. Invalid decrypting to max size

Hex encoded ciphertext:

```
013a60aa202dedad2d9e78c0c99077ccc17b7d0533aeaf184dcb8c9a81ca4de5
715ada598d59b926606dcb005935421f6ebfd32e62802f0e2de8df08f1ae00e4
aced6ebf361a38df817c892309bd07c92c4f2f7be89f286f99711372e3dd959c
cb0a150b28578f29040b39ecf989c26eb77a3480c2d4d363b9563a70f0a0789c
c4300af1e600de39dae4a49335d35ac0156f5395ebebf35531c819c9cf498a97
e67ee2299d84564444f7bcd51f9f08d6bac0872439ad57eb9a8134dc665add1f
813d5031484c905a433c115bf889dc46ac672a8898fe235bf463b1b46345299b
6f100b48fa954fc262ce58e83f95955b321c1e86bbfe398b588dd5c75c2c6853
2b
```

Hex encoded decrypted message:

```
5c5345594cec69769af002af6513415a0848a4d45601c44b9e84b6e0a42c1356
cfc20604df8d334de53fafc6de2125f2eae2ec9864b0b0d03ef1539017ecf312
460cbb3c7c49fe124cf1becce8903f227ec85daaae73fe256acdf39f47b46c14
fbd289d7f9b9e22ee0e7c5d77cbbf22a2f9fbf4e26c5999a7b8335f6478912ed
fb77a373a82dabbc8233e585ea5ddfef3d0d872a4a007018005014a887349595
1210a4b90209b8286466030e56b1050ed600c777b4dbdb64f86516ef269a3ba7
7285f9be9afebcaad35226f7b3296932baf8dbfea3d77eb6520da203ffc06722
32761e2a8a05186181b8dd1f77f069f7af8de118aa18
```

B.2.7. Invalid with null padded ciphertext

Ciphertext that decrypts to an invalid plaintext, with the ciphertext starting with a zero byte.

Hex encoded ciphertext:

```
000a56cfd7abf21e2c65a28b8145fee663818b6cd218601e9e62bed7de5ac74c
eb07b435b305c99e3e286152036086b4a13eb26164f6f4bf79ac7d6c3fbcdcd7
4c6dc324c04949a0c88cbf616626cb171a4e333a0e124524b44f78e2d8100c03
e231791523b7d76db2d464e6b5e64244e04ad21128a5feb56092b4ceda35ebfb
84f7fafa8e1c098d9923d66e541b36307574db8912630fcf734af7bcd4118f1f
700804c5001a5950780188fc7862384fb3fc0708eaf3cb2b119177bb6928758f
a75e644d258ac02748dcae7a78ec6c8679598fa9deee4663f1efe27efe8bd155
72ff40676b95469aaa42eab4226d86c1160343255cce4fb7728f7349da27ed09
6b
```

Hex encoded decrypted message:

```
fc874af235e261083c2c
```

B.2.8. Invalid with second to last length

The length of the synthetic message comes from second to last value from the PRF output as the last value is too long for this key size.

Hex encoded ciphertext:

011644d88f2955389b6d215fc3acd733265d82348fa15178d99e38fd2842030d
4e31d1ab15b0f00a80add1a9bfe112f9c42e827c487193c360dc5888c8c42d54
59b1f2c1952dca679ded08f190b6a78bbb6ea00438ebb8b03dc56e45cb36bee4
9a385c71fbf9bbcceda7b130ad6cee2106eff34dd224fba8e9990edf893ae52b
0910cdadf44bf29812a98bb9dc1638815112950bb9e7bad11c610d6406dd2d1c
36cddd639e024205a1a2bd82ec97059ef61b5226e3246ca672962194d1222cc0
32003fe34bf5ca0227f1c86439dcfba5203cf57099884276b0fbbf9b9cec9339
2673d8290a1a3452f3791c3881c9be6a1eef23841a8a5fde255cbbd3c4fc3382
7c

Hex encoded decrypted message:

9b9b496e44544456e70d9be86bd398dae00c7c05d6ea76c45d12f050b1c61c21
9b2fe47749451e44d96d11916f6ec4cf9eea67cb

B.2.9. Invalid first byte of padding

Otherwise valid plaintext, but with first byte equal to 0x01 instead of 0x00.

Hex encoded ciphertext:

002c9ddc36ba4cf0038692b2d3a1c61a4bb3786a97ce2e46a3ba74d03158aeef
456ce0f4db04dda3fe062268a1711250a18c69778a6280d88e133a16254e1f0e
30ce8dac9b57d2e39a2f7d7be3ee4e08aec2fdbe8dadad7fdbf442a29a8fb408
57407bf6be35596b8eefb5c2b3f58b894452c2dc54a6123a1a38d642e2375174
6597e08d71ac92704adc17803b19e131b4d1927881f43b0200e6f95658f559f9
12c889b4cd51862784364896cd6e8618f485a992f82997ad6a0917e32ae5872e
af850092b2d6c782ad35f487b79682333c1750c685d7d32ab3e1538f31dcaa5e
7d5d2825875242c83947308dcf63ba4bfff20334c9c140c837dbdbae7a8dee72
ff

Hex encoded message:

f6d0f5b78082fe61c04674

B.2.10. Invalid second byte of padding

Otherwise valid plaintext, but with the second byte equal to 0x01 instead of 0x02.

Hex encoded ciphertext:

```
00c5d77826c1ab7a34d6390f9d342d5dbe848942e2618287952ba0350d7de672
6112e9cebc391a0fae1839e2bf168229e3e0d71d4161801509f1f28f6e1487ca
52df05c466b6b0a6fbbe57a3268a970610ec0beac39ec0fa67babce1ef2a86bf
77466dc127d7d0d2962c20e66593126f276863cd38dc6351428f884c1384f67c
ad0a0ffdbc2af16711fb68dc559b96b37b4f04cd133ffc7d79c43c42ca4948fa
895b9daeb853150c8a5169849b730cc77d68b0217d6c0e3dbf38d751a1998186
633418367e7576530566c23d6d4e0da9b038d0bb5169ce40133ea076472d0550
01f0135645940fd08ea44269af2604c8b1ba225053d6db9ab43577689401bdc0
f3
```

Hex encoded decrypted message:

```
1ab287fcef3ff17067914d
```

B.2.11. Invalid with zero length PS

Hex encoded ciphertext:

```
011e24da411e182d85e7c350a7dbff898c3e17bee1573fe3d0c0d3be53a384a6
5d597d52247e56d10b01cf1a51533e47c37f38bc62490e449b5f4ea35aec422a
63142c95da26e800e14504ce704edc7e38faaff697b74814c14d19f7f6f4d942
c1db61eca70266748ed83195bffa2eb85f49047a7bc7aae7f927ff445f15136b
28b4cad396312ae09a62ef894688ad50bc53d6977236b76d30c9bb2dacd0d583
00f2b3dc69a7c3212c6dcf6db583c59d94e7550ecc871991139259a9a0014516
35c777c7ac46db113a2460f157f9f830c186f0a0c47ca3012c8d309bea899852
febe76b87874292b8230be9de5c88fb94a0f19b2df71ca1efeb5f772b2d9b380
16
```

Hex encoded decrypted message:

```
5659f49338cfcd423dfcce
```

B.2.12. Invalid with null byte at the eighth byte of padding

This creates a plaintext with PS of 7 bytes

Hex encoded ciphertext:

```
014b47d00714b92e98eacd73b07816ef530f8355fca68de6ba56828afab9d43b
96fcb11bee0e900e966a51705b95e82599f3df202d51d520c96adf32e1c0758a
7e0a8694699af8cfd9ed2850678f70e952f029a4b461ac8a8e049ef9ffc483eb
dc9a525bae3ddf8d69f8f711d30135fd9cce0c3d09d1a75c5b837d9f0d86827a
5f9edccd56af835572ba099a44a69277f31999753cd9887fe46ae7c7eb98d589
4692847179dac0ba305ed584f2887f7a6e88c843e6f0d4e99ffdcf51e2661983
01dab2bcb7b07944207c71a89f1449f3690cf09b9f15a5588a2c16d18ef43787
6285e9d04fd01ff30dea46a3eb17151e63b784548f1c65eeb08bec9901b93534
32
```

Hex encoded decrypted message:

```
c917a56bd800b8d9578657
```

B.2.13. Invalid with padding separator missing

Hex encoded ciphertext:

```
01541f0012b2dfd2d2b91cd22570dd577af0d27c29baa4d7a3cec2ddf8cdb957
257d8d436455d331e35730d1df7058cbb01649beb236154858843e2e14ae54fd
53496883163b8cb30fea540778726453204a846b04535878bebe9e60f8783b77
e5dfd4d346cd688aa2ee9e0ba9ac2bfd16005de994ea4a9c40adbafcf79dac48
59480a8e6e6849077b07f5e3dce8b97f678fc4b1da7e72004cfae36f01cb23f6
9180a46e3b9488f3b595e83510c9c4b298c953971d82610562976271e756f97e
f317b440271be9847178a482098447516ea29bd599592dbf22e35c91f8e2155b
d2f6289bac31c4c608b80bc208608a17cc45f7cff774549782882cf72baf0feb
2f
```

Hex encoded decrypted message:

```
a4e88815652839f2482a8f
```

B.3. 3072-bit key

Test vectors that use a key with a 3072-bit modulus

B.3.1. Private key

Private key to use for decryption of following test data

```
-----BEGIN PRIVATE KEY-----
MIIG/gIBADANBgkqhkiG9w0BAQEFAASCBugwggbkAgEAAoIBgQCv1xyq1en1uMbD
ZwcKR/Gdfmau3hilsnQfs8TTNDRWBpKi2QnvaIjsYC/2uTryWO50MDwwGvzU7b7E
MxHdyN2/AN2744bTO40OIrG0STbcSJhxuFI3s0znR62P2wxOTR2qeq3wc4XFyHMs
y31aSeLlDIg8fXrBDtanTZrJDfkSmQWhfUoIchD8eLbQSx65aUgsEabut5xQ5bFv
PyVPdXFSiy8XFquBbW7KBye96pgFkylzDrjDPOcdYd1Kw5O2JW4HrB0STwIA0cPg
Wkwbx/HtL8g+Vxmc/lkIsQCH4n+9l9LCQhRhnHFHyPvvyjm8JWdipoI1MffiNNaO
rnoNn68Q3RXpUjeAx9WuWAlK1SWpBjtMM/leEAbaLrEtN0NolJXB8gI+QHNTxes+
TKHEjP+BoQkA0Ugg64Aa9PGllsS5zppTH8+KVNn/1yQli27sIBCN9v39dtSuA7p+
pZjcsOSigISVhyhvTX8lbOheXrVnmx2swfkJVkm3Ll+gcq6wA3kCAwEAAQKCAYAl
F+rLOv70v/rmA5jfmVel0qFUqDNo2OFYQrL1nuCfeRl70u8emt2vh4b2tBJ0R0Be
MEKyHy9Qt6pzdxaAw7vLbCJaXV/2tWxHHBiCoKM7CvFlo+1sJJ3HeD5rx1isN+ZX
LTP+MlB47ZUmUPLrlgSQLvmaURoRGdE8T8mkOhddzfv7GhQA/hcJO2nNPNuJX2VD
LqIZX5BRHHM2tYpoXaz/La9MXpLlZbFmWuYOUSuqmWW4CNX/EZzrfNaSy/kgBnr6
zYCfZr5wbaJoEKx5DbFWyUjm/lgbyYSRV/TaST86ZLDG4RngMbEH+ENs4pFgtFi4
+fEJX96xkmM4T/U4dVe8TxC0A03oQXA7rSwedpwjhT62MKNtEGHdRuKmu690Opfw
syw29QwaNyLe86OU2RwuB4vwnXlezeXla4IC+XQCb3X8Vumg3WqI8ufLeO8SmMxs
ZSB8pFvTcYiAe0/NseYN2OW4Vkj7fvqLb91Ejzl0Go2YCf4WOvPeRbrCSlqEHIEC
gcEA5KPYMNc+izHGguJ0/8n9EqwxPS3QUT1QVw2360di/qGT57tUC5SpSl3ddCrP
c/XeucrjG9I6xWC7J5T9aCYfggP1cZKCkE9GHqzuLOfgoAl6p8jbq9M/G/JpkSoH
gnFPqTtJ6sQ26z3nNKfW/9/YwsFDXoQ/xwn5BI5UKhl8SFQr6yuF6tD15kqmPQ7A
FSs/hWEt3Ka/3qvzF11ZfUBWPg4GLZHLAoiACC/p+PCRvb3aMW7rHoWMpE0rAorp
zeOpAoHBAMThzQ7PQphhXh94m6feIv1QlK9K0awpUO6WMDhaIECaKAxlOKL+7QMU
SOJuItZwk6cfncdOvRq8DpzoPWcLAnarHIWsc03Yv5x0zH/svXMtHXX4ie9GCkgZ
ul4bAd4jMlVRgbdvqWVEkxmKYG8Ayv2KkzVuRW8iO3Ub1bXKl64rObp3+3wXTILs
AhhlYNXie/GKJjzCEtnMZrAdHaJnPyl9TBvtRFtO/F2wYTbsqr2Cy1TQ/MQmmdTW
CgInv+ADUQKBwEDzDkHpkznFXQfnPqc/AOYiBibD8e5yBXWFTx7F+6grzDFC9MAJ
bgHTIkqSsrXVPXz31oYbtVhGf0PiPg4s7jxn1Xx6yx4ldtzV8R7Oi+/KYY5yL3zj
GIVe2oBDOTjj6WZAkmHfdV5kCl7Z4uhy9Ud10SZzWQ64lYWmzN/ct4Jwbr1ycqte
ysutnwWvP/+Ddpr0HSwWLmEZ5YdYnEhJU3ZzU2v0g3/nuL8apVNzO2N0IBx0ztOv
ymEODs69GWfEaQKBwQC5iEwUG66XKJJpN9//dm8kpg4njms+QQUagP/Z6tyf5GW/
IJgZygASOchhUQaVbCtIf5vQ2VuNWRCwPo62jwJ4T9GmCpfyEUKoLs0T9EWnxykP
JfLeP/OqdExTKEI/Uo25JwEFmz1XyCKTG/q6QFYKTc9ht5PJIcpEFsHy+YKsx+Ez
3qNoEhCxA7UJxmdVx4OjX9uewAjBpERUzGtDwuYbtA7H9nTHUwyxQWirOKXBxwLT
38mDExk+H6H42/qOILECgcEA4SpCAUB9J1HJrrQssfnmrzTc1EUxqa4qI9tUkvPC
Ip9uM6KKimZA5L8vHGojN4xeVhXg6xK/FOgbuZxM4VG1TmEoIr63yp5BClr92wym
IeWXACudHIGMhWAvmUUpGkdQYuxq9T9PUgea2BrFmjfZ1e9wCHX6d0IdUHBsdM4X
hyicD6D5SynhtlJJafmdTigiLO9JWkbtIZ7QaeB3Ef1Sx2pu/NyKnUQp4dFNmscg
RiYH7HQtpEgHd3Bk2J0rdOQr
-----END PRIVATE KEY-----
```

B.3.2. Valid

Hex encoded ciphertext:

```
6c60845a854b4571f678941ae35a2ac03f67c21e21146f9db1f2306be9f13645
3b86ad55647d4f7b5c9e62197aaff0c0e40a3b54c4cde14e774b1c5959b6c2a2
302896ffae1f73b00b862a20ff4304fe06cea7ff30ecb3773ca9af27a0b54547
350d7c07dfb0a39629c7e71e83fc5af9b2adbaf898e037f1de696a3f328cf45a
f7ec9aff7173854087fb8fbf34be981efbd8493f9438d1b2ba2a86af082662aa
46ae9adfbec51e5f3d9550a4dd1dcb7c8969c9587a6edc82a8cabbc785c40d9f
bd12064559fb769450ac3e47e87bc046148130d7eaa843e4b3ccef3675d06305
00803cb7ffee3882378c1a404e850c3e20707bb745e42b13c18786c4976076ed
9fa8fd0ff15e571bef02cbbe2f90c908ac3734a433b73e778d4d17fcc28f4918
5ebc6e8536a06d293202d94496453bfdf1c2c7833a3f99fa38ca8a81f42eaa52
9d603b890308a319c0ab63a35ff8ebac965f6278f5a7e5d622be5d5fe55f0ca3
ec993d55430d2bf59c5d3e860e90c16d91a04596f6fdf60d89ed95d88c036dde
```

ASCII encoded decrypted message:

```
forty two
```

B.3.3. Valid empty

Hex encoded ciphertext:

```
4a454e0dbba01df544fd2fec0099ffd533301e1a4481597f83b8587ff638c029
ab434c59ce72bc4b13d0d4901945ad8cef4ef913626b09c9cad69336a9e409e8
5f59b0d60df25267443009e8e53b4aeed87c5301649db5f7b38d688850f5a408
b1c6608af1428f5e8b9262a8638cf89c28babf7df6d3d55101ff18ebf1bdf3c9
2e66b5ca279f6f4f759d4163f42e49bd836474ca3d48f4a40a5a2974e64c99d2
ee282e1b089aa6b360c4c815f743dda33ac12dfb0fe7a2653d753ee5cd8a4129
790ca888767f4128d61340bdecc579c3f9f36b1b256b51c1a66c46149f963bc8
fd0771ad2cb4b23054f5cc5fcf5534bc97de55bcc831894baaca7c40d46e056a
836523c52a84de3d7642ff4ce02f508b7ca1f485a028ca34f4c57417023cb724
87ee5fa60fc9caadf29b3fe3ef632abf89baae921fd43ca275b3026e286720f2
3422203eec63a9b737f9d18987736abf241c46b748a60f68a8dddfc9664312b3
6ec52b8a7ce8060b830a4de5f2475e14ed0570c631d608935e75da9b091084db
```

The result of decryption is a message of length 0.

B.3.4. Valid with ciphertext starting with a zero byte

Hex encoded ciphertext:

```
00f4d565a3286784dbb85327db8807ae557ead229f92aba945cecda5225f606a
7d6130edeeb6f26724d1eff1110f9eb18dc3248140ee3837e6688391e78796c5
26791384f045e21b6b853fb6342a11f309eb77962f37ce23925af600847fbd30
e6e07e57de50b606e6b7f288cc777c1a6834f27e6edace508452128916eef778
8c8bb227e3548c6a761cc4e9dd1a3584176dc053ba3500adb1d5e1611291654f
12dfc5722832f635db3002d73f9defc310ace62c63868d341619c7ee15b20243
b3371e05078e11219770c701d9f341af35df1bc729de294825ff2e416aa11526
612852777eb131f9c45151eb144980d70608d2fc4043477368369aa0fe487a48
bd57e66b00c3c58f941549f5ec050fca64449debe7a0c4ac51e55cb71620a703
12aa4bd85fac1410c9c7f9d6ec610b7d11bf8faeffa20255d1a1bead9297d0aa
8765cd2805847d639bc439f4a6c896e2008f746f9590ff4596de5ddde000ed66
6c452c978043ff4298461eb5a26d5e63d821438627f91201924bf7f2aeee1727
```

ASCII encoded message:

```
forty two
```

B.3.5. Invalid decrypting to empty

Hex encoded ciphertext:

```
5e956cd9652f4a2ece902931013e09662b6a9257ad1e987fb75f73a0606df2a4
b04789770820c2e02322c4e826f767bd895734a01e20609c3be4517a7a2a589e
a1cdc137beb73eb38dac781b52e863de9620f79f9b90fd5b953651fcbfef4a9f
1cc07421d511a87dd6942caab6a5a0f4df473e62defb529a7de1509ab99c596e
1dff1320402298d8be73a896cc86c38ae3f2f576e9ea70cc28ad575cb0f854f0
be43186baa9c18e29c47c6ca77135db79c811231b7c1730955887d321fdc0656
8382b86643cf089b10e35ab23e827d2e5aa7b4e99ff2e914f302351819eb4d16
93243b35f8bf1d42d08f8ec4acafa35f747a4a975a28643ec630d8e4fa5be59d
81995660a14bb64c1fea5146d6b11f92da6a3956dd5cb5e0d747cf2ea23f8161
7769185336263d46ef4c144b754de62a6337342d6c85a95f19f015724546ee3f
c4823eca603dbc1dc01c2d5ed50bd72d8e96df2dc048edde0081284068283fc5
e73a6139851abf2f29977d0b3d160c883a42a37efba1be05c1a0b1741d7ddf59
```

The result of decryption is a message of length 0.

B.3.6. Invalid decrypting to max size

Hex encoded ciphertext:

```
1715065322522dff85049800f6a29ab5f98c465020467414b2a44127fe9446da
47fa18047900f99afe67c2df6f50160bb8e90bff296610fde632b3859d4d0d2e
644f23835028c46cca01b84b88231d7e03154edec6627bcba23de76740d83985
1fa12d74c8f92e540c73fe837b91b7d699b311997d5f0f7864c486d499c3a79c
111faaacbe4799597a25066c6200215c3d158f3817c1aa57f18bdaad0be1658d
a9da93f5cc6c3c4dd72788af57adbb6a0c26f42d32d95b8a4f95e8c6feb2f8a5
d53b19a50a0b7cbc25e055ad03e5ace8f3f7db13e57759f67b65d143f08cca15
992c6b2aae643390483de111c2988d4e76b42596266005103c8de6044fb7398e
b3c28a864fa672de5fd8774510ff45e05969a11a4c7d3f343e331190d2dcf24f
b9154ba904dc94af98afc5774a9617d0418fe6d13f8245c7d7626c176138dd69
8a23547c25f27c2b98ea4d8a45c7842b81888e4cc14e5b72e9cf91f56956c93d
bf2e5f44a8282a7813157fc481ff1371a0f66b31797e81ebdb09a673d4db96d6
```

Hex encoded decrypted message:

```
7b036fcd6243900e4236c894e2462c17738acc87e01a76f4d95cb9a328d9acde
81650283b8e8f60a217e3bdee835c7b222ad4c85d0acdb9a309bd2a754609a65
dec50f3aa04c6d5891034566b9563d42668ede1f8992b17753a2132e28970584
e255efc8b45a41c5dbd7567f014acec5fe6fdb6d484790360a913ebb9defcd74
ff377f2a8ba46d2ed85f733c9a3da08eb57ecedfafda806778f03c66b2c5d287
4cec1c291b2d49eb194c7b5d0dd2908ae90f4843268a2c45563092ade08acb6a
b481a08176102fc803fbb2f8ad11b0e1531bd37df543498daf180b12017f4d4d
426ca29b4161075534bfb914968088a9d13785d0adc0e2580d3548494b2a9e91
605f2b27e6cc701c796f0de7c6f471f6ab6cb9272a1ed637ca32a60d117505d8
2af3c1336104afb537d01a8f70b510e1eebf4869cb976c419473795a66c7f5e6
e20a8094b1bb603a74330c537c5c0698c31538bd2e138c1275a1bdf24c5fa8ab
3b7b526324e7918a382d1363b3d463764222150e04
```

B.3.7. Invalid with null padded ciphertext

Ciphertext that decrypts to an invalid plaintext, with the ciphertext starting with a zero byte.

Hex encoded ciphertext:

```
00128e116c0d348217c1d5756611be833caec3bd19f3d2ed383c4523ef3d8a5c
d1130bdb3fcd3ba1c0265322fc98c767b3b971054f5e3067a32b4878bbf7e17d
b80a855427e2e7d2ec26294b79edbb6352c7812270040e3cbaf560de1486171d
a649af786551ae0e5b6ef1fa1e22196c1c3039a50cfe09620da09716e75e9a9b
5ad164953c5a7e48f77ef9e0f59e962cf3985dc572ded966e8241632a9a5a420
3500a50d70491d80846b72019c456bfbdf7f15f740af0c2ef2a46787c54dcd90
a0e91257642f2c10b21052024b1d28ad14d78c0d9702a14b9ab9552f023bfb71
6f9c9a1f691959057f52b197086fb9bc24d45e2b063ffb235b54d3ab7575c7f3
e411398ddb30945c15f3f5d5ebbe302f22f60047d139d402f5b8a959421d1946
cf748c84065c5d0f0302f7ade6335014ab4467698ad827684d2bdeeb4757d276
f131340438506473f271c3fe24ca58e1c60dcc17aff8b0373637d897f627ea57
1da148a81d399ced9d65a4564bb6d23fbf4c14674d7551283a4293de51a1516a
```

Hex encoded decrypted message:

732f025d1adea74649b4

B.3.8. Invalid with second to last length

The length of the synthetic message comes from second-to-last value from the PRF output as the last value is too long for this key size.

Hex encoded ciphertext:

7db0390d75fcf9d4c59cf27b264190d856da9abd11e92334d0e5f71005cfed86
5a711dfa28b791188374b61916dbc11339bf14b06f5f3f68c206c5607380e13d
a3129bfb744157e1527dd6fdf6651248b028a496ae1b97702d44706043cdaa7a
59c0f41367303f21f268968bf3bd2904db3ae5239b55f8b438d93d7db9d1666c
071c0857e2ec37757463769c54e51f052b2a71b04c2869e9e7049a1037b84292
06c99726f07289bac18363e7eb2a5b417f47c37a55090cda676517b3549c873f
2fe95da9681752ec9864b069089a2ed2f340c8b04ee00079055a817a3355b46a
c7dc00d17f4504ccfbcfcadb0c04cb6b22069e179385ae1eafabad5521bac2b8
a8ee1dfff59a22eb3fdacfc87175d10d7894cfd869d056057dd9944b869c1784
fcc27f731bc46171d39570fbffbadf082d33f6352ecf44aca8d9478e53f5a5b7
c852b401e8f5f74da49da91e65bdc97765a9523b7a0885a6f8afe5759d58009f
bfa837472a968e6ae92026a5e0202a395483095302d6c3985b5f5831c521a271

Hex encoded decrypted message:

56a3bea054e01338be9b7d7957539c

B.3.9. Invalid first byte of padding

Otherwise valid plaintext, but with first byte equal to 0x01 instead of 0x00.

Hex encoded ciphertext:

6db80adb5ff0a768caf1378ecc382a694e7d1bde2eff4ba12c48aaf794ded7a9
94a5b2b57acec20dbec4ae385c9dd531945c0f197a5496908725fc99d88601a1
7d3bb0b2d38d2c1c3100f39955a4cb3dbed5a38bf900f23d91e173640e4ec655
c84fdfe71fcdb12a386108fcf718c9b7af37d39703e882436224c877a2235e83
44fba6c951eb7e2a4d1d1de81fb463ac1b880f6cc0e59ade05c8ce35179ecd09
546731fc07b141d3d6b342a97ae747e61a9130f72d37ac5a2c30215b6cbd66c7
db893810df58b4c457b4b54f34428247d584e0fa71062446210db08254fb9ead
1ba1a393c724bd291f0cf1a7143f32df849051dc896d7d176fef3b57ab6dffd6
26d0c3044e9edb2e3d012ace202d2581df01bec7e9aa0727a6650dd373d374f0
bc0f4a611f8139dfe97d63e70c6188f4df5b672e47c51d8aa567097293fbff12
7c75ec690b43407578b73c85451710a0cece58fd497d7f7bd36a8a92783ef7dc
6265dff52aac8b70340b996508d39217f2783ce6fc91a1cc94bb2ac487b84f62

Hex encoded message:

6d8d3a094ff3afff4c

B.3.10. Invalid second byte of padding

Otherwise valid plaintext, but with the second byte equal to 0x01 instead of 0x02.

Hex encoded ciphertext:

417328c034458563079a4024817d0150340c34e25ae16dcad690623f702e5c74
8a6ebb3419ff48f486f83ba9df35c05efbd7f40613f0fc996c53706c30df6bba
6dcd4a40825f96133f3c21638a342bd4663dffbd0073980dac47f8c1dd8e97ce
1412e4f91f2a8adb1ac2b1071066efe8d718bbb88ca4a59bd61500e826f23652
55a409bece0f972df97c3a55e09289ef5fa815a2353ef393fd1aecfc888d611c
16aec532e5148be15ef1bf2834b8f75bb26db08b66d2baad6464f8439d1986b5
33813321dbb180080910f233bcc4dd784fb21871aef41be08b7bfad4ecc3b68f
228cb5317ac6ec1227bc7d0e452037ba918ee1da9fdb8393ae93b1e937a8d469
1a17871d5092d2384b6190a53df888f65b951b05ed4ad57fe4b0c6a47b5b22f3
2a7f23c1a234c9feb5d8713d949686760680da4db454f4acad972470033472b9
864d63e8d23eefc87ebcf464ecf33f67fbcdd48eab38c5292586b36aef5981ed
2fa07b2f9e23fc57d9eb71bfff4111c857e9fff23ceb31e72592e70c874b4936

Hex encoded decrypted message:

c6ae80ffa80bc184b0

B.3.11. Invalid with zero length PS

Hex encoded ciphertext:

8542c626fe533467acffcd4e617692244c9b5a3bf0a215c5d64891ced4bf4f95
91b4b2aedff9843057986d81631b0acb3704ec2180e5696e8bd15b217a0ec36d
2061b0e2182faa3d1c59bd3f9086a10077a3337a3f5da503ec3753535ffd25b8
37a12f2541afefd0cffb0224b8f874e4bed13949e105c075ed44e287c5ae03b1
55e06b90ed247d2c07f1ef3323e3508cce4e4074606c54172ad74d12f8c3a47f
654ad671104bf7681e5b061862747d9afd37e07d8e0e2291e01f14a95a1bb4cb
b47c304ef067595a3947ee2d722067e38a0f046f43ec29cac6a8801c6e3e9a23
31b1d45a7aa2c6af3205be382dd026e389614ee095665a611ab2e8dced2ee1c9
d08ac9de11aef5b3803fc9a9ce8231ec87b5fed386fb92ee3db995a89307bcba
844bd0a691c29ae51216e949dfc813133cb06a07265fd807bcb3377f6adb0a48
1d9b7f4420031158959397736b95371c4febef29edae946fa245e7c50729e2e
558cfaad773d1fd5f67b457a6d9d17a847c6fcbdb103a86f35f228cefc06cea0

Hex encoded decrypted message:

a8a9301daa01bb25c7

B.3.12. Invalid with null byte at the eighth byte of padding

This creates a plaintext with PS of 7 bytes.

Hex encoded ciphertext:

449dfa237a70a99cb0351793ec8677882021c2aa743580bf6a0ea672055cffe8
303ac42855b1d1f3373aae6af09cb9074180fc963e9d1478a4f98b3b4861d3e7
f0aa8560cf603711f139db77667ca14ba3a1acdedfca9ef4603d6d7eb0645bfc
805304f9ad9d77d34762ce5cd84bd3ec9d35c30e3be72a1e8d355d5674a141b5
530659ad64ebb6082e6f73a80832ab6388912538914654d34602f4b3b1c78589
b4a5d964b2efcca1dc7004c41f6cafcb5a7159a7fc7c0398604d0edbd4c8f4f0
4067da6a153a05e7cbeea13b5ee412400ef7d4f3106f4798da707ec37a11286d
f2b7a204856d5ff773613fd1e453a7114b78e347d3e8078e1cb3276b3562486b
a630bf719697e0073a123c3e60ebb5c7a1ccff4279faffa2402bc1109f8d559d
6766e73591943dfcf25ba10c3762f02af85187799b8b4b135c3990793a6fd326
42f1557405ba55cc7cf7336a0e967073c5fa50743f9cc5e3017c172d9898d2af
83345e71b3e0c22ab791eacb6484a32ec60ebc226ec9deaee91b1a0560c2b571

Hex encoded decrypted message:

6c716fe01d44398018

B.3.13. Invalid with padding separator missing

Hex encoded ciphertext:

a7a5c99e50da48769ecb779d9abe86ef9ec8c38c6f43f17c7f2d7af608a4a1bd
6cf695b47e97c191c61fb5a27318d02f495a176b9fae5a55b5d3fabd1d8aae49
57e3879cb0c60f037724e11be5f30f08fc51c033731f14b44b414d11278cd3db
a7e1c8bfe208d2b2bb7ec36366dacb6c88b24cd79ab394adf19dbbc21dfa5788
bacbadc6a62f79cf54fd8cf585c615b5c0eb94c35aa9de25321c8ffefb8916bb
aa2697cb2dd82ee98939df9b6704cee77793edd2b4947d82e00e574966497073
6c59a84197bd72b5c71e36aae29cd39af6ac73a368edbc1ca792e1309f442aaf
cd77c992c88f8e4863149f221695cb7b0236e75b2339a02c4ea114854372c306
b9412d8eedb600a31532002f2cea07b4df963a093185e4607732e46d753b5409
74fb5a5c3f9432df22e85bb17611370966c5522fd23f2ad3484341ba7fd8885f
c8e6d379a611d13a2aca784fba2073208faad2137bf1979a0fa146c1880d4337
db3274269493bab44a1bcd0681f7227ffdf589c2e925ed9d36302509d1109ba4

Hex encoded decrypted message:

aa2de6cde4e2442884

B.4. 4096-bit key

Test vectors that use a key with a 4096-bit modulus.

B.4.1. Private key

Private key to use for decryption of following test data

```
-----BEGIN PRIVATE KEY-----
MIIJQQIBADANBgkqhkiG9w0BAQEFAASCCSswggknAgEAAoICAQCP3jlT5VxITxxC
Sq7JA99vOdMNsWBSloAaGC0650aOGSezhv550foWGyiU+Gixo3XzGcMGs7vaX8tF
oROlVw6yRzGAz+ERn6B8JeRojKJV4hWmrFa8EQfuJDPM6EXLb1X2u4wwDpwDs0IV
c4jUD6ekXTsNEFW0mjw7CkRLcnGXFUFDXj/J8pVveF5cxtFZi5+JwcE79MX+8TEH
8OAW9W11O9svcPFuzHUI0gv1BixcZIvVu0pzVcQpNeKv41ZZW20PNOpynI7JjOjC
f4ebGZx03y4i7jA2LpLznbjcmiT1Ge1a/NHS6BriA//9gCjcAvgdO/Sws7rEJLEH
JQ0gmMBuSt3yaQpOdrejmXNX2Cvz6ewQ16i4hiq8AramcyiRIVHjsfQJWMeirYX7
ZCsrCVRt0WGE9PPUxo/mDIZ3CpQ0IRwNxCwuJO35AqLtTUaFXhRW5l/W/hbVbaTj
yAmd2W57LCFjPh7S41zECRl6zuf7keGbad094Tkm8dr+/N47mAtP8C+Q2YBIEsxO
qYobVlAP/Rznsq6ScUyKagMdKby38EewybvOcN/g4fDbBs7a+Jdp1Y4cFT7ugTet
ig+yHJlJgjbQQYA8sjwAPeZRArmnMsOBvEG+F8tLIRmT9iIG7AV8cVdQ1c0xq2e+
bIZV6kq1pstAEcKJs2wS3hr3oI7dDQIDAQABAoICABUFM5wZ9XdNM8RN1CbjrXg7
WyZzha7bdYEaijVCiHEHpODatwDGmMM1GaNWfjmCOCAaYf/vL+HSwEj+pXexdiC/
PTDkbq6cA59QDATok2l3/JDbOlsFJAL8ACOOxm/YZxeDRJLa95mkzeGDfXj6hpSB
2LhqbBNUu7smn17krEsWXHBC65AOpR3h2Ou2Uu1B9L/0cs7XTzyWMgNGgv9nwaPw
6HAQkP4O2oS85+lXixC+0h2a3WBnph8dU+8+CKZKsO8ZXK39SPGS6IlOh9o2DkGX
QE7zPVEVUfxNTa7RyNeU/7SdHu0xY4glbHg2CwJGXskqsvBEemPhNopkM/xQxUDA
AoJ3HsNbpXsp+V4HjijdV47ncQBPr2O3Ln/V3nVqgJJ88MWqC3kfDMwtE+b8yLE3
RJ8Tqb+Rih0Zr/4AfRNlSZzzmUKbN39f9swLzv5BJMuBhbq2W/019xb+b+Jq/o5p
NgUKVImnvJ+Pm0tFaMHWKE3LHOib3o4D19bVHfkEXXeFwy2Dul/imDwOQqEu4Gei
3mkzIEEA2f4Tl17ta62kdNn73lgpdtEqE0DNq9yQpK7itxRjTlwGQShLzYJaD9bz
TIgx5H+0FIGO4yzO5BJelPSFNg646BpiZsKGPy9PhYhA2UKPvKwbYhdEub3pO4pu
aSn9WE6hFrtphmzrs0+DAoIBAQDJmfE5qhyZdUlhG4a6gCzirXlYRG0LESf5pqJT
IT7nfCV25j7ir3hXqVA3M4zXNDP0jtDUFUlmvs2Iwte4cvybmbnlCYYCaMvienq2
soT9S+wRWT1ituC+xg9w9JAdkrdm55A/r+ioxeAID1OKdlmOtGxyb9vusNvLM/xq
yjciOvx/hYzRTnVme2qVuCfxo3WgFneLmT3KgZqeKI4bklP5ZCiqmtWrVE5JV7FA
5yeuvqt5hpKwnYLCyWpyrX6ZiWW4NoZQEgT7nvyAxlJXsxibhXK0VLIJbFyFOyRF
XzHJblcyFKt2FEyqKY8Uo9vhFpm4YSU2/h4/BRpEbxZ60pPLAoIBAQC2sDpoWubg
FYJ76iZ0rLce+UnRw7fiySz73Pcirky7ohduawM83VuG6MlxYTgTrvNFKQxRt/kg
b9SZmqVj4TCdPG+rlbfh31nCs5o7tIXVwpXJYF6Zw+3XFcVQfs6yOKGED7kvcxlv
c0IogRY7VwL+2uUPY8P2pSubRh46ojCk2rBPnUvAOiywuVp/8ce4vWD4OE9qMGxL
tQCtWcNQc7IloopxD/6JxDvbi3AkLmhnZTwVQ1pvsTLxSftBeDMtBihwyYkzKp2q
LQ77z+ROQjxWNueGlJiyLiISy5OpfzftrEI7HbbRijFfIW1wpBLktjJc+V3TQpvw
81DfPWz7HSeHAoIBABbkhkH4ETtT2DtQtuHiwDYVYNwnGQLStelhhLlP4ybCMDbq
OmXfaxj5PwSls+6UHPo+ZXRP3n/uVsSa64gkdo7sy/E4PaF0aQKUye6JIxToVCw8
J/5Xqfm/3+0BuIPDL1ReI0E4JoFTxvbjVqc4fT75Of9cuGrIZo0SoC04PT6D2Dxa
2/zAnhA/H4FU4iRMXYUKSBTwF9YxFW+0Rz7WRPbi6sNExdKoxiMJ9DTciuMK/8nT
AEFSAZFA8feHwAKLWaOrfWEGSs6P5ThyTNNsoPS4GKhlnbn4NiFi7+F4x9yJzRkA
w9+qunHwOjEr0kIjIN2RGnq+LI0XdT4kJzdXRjUCggEAUs9hOmpyGwNFCXxjACJG
Q6EGiDIqiX9dh7FqyOqvV6Iq8t9JMW65jVa58U37SFjWZvQrZGN7ZuuqOBJ0g+jQ
y4VYrTOJjbZbjrkw4X176SByGz02xIaeqg9xSIKimQC4ng8uh0aqMe3SAGA7Ppy7
e+CnUimL66KFOLY4/6UDXcbPgtcvPixgk58BWdu0B5a4fOuxe9YFUO/0JZ/5u98l
0o9yJ7vzSmmMJIF5TL1IkA3AhXbpaJWvHNbHMK8Wq4MOE8oXCf14SGpyT0y2FY9K
```

oF305vk2mhX1VgW3om5LYM3jm75lX2g/5vpVDGkjM08vVxumHxwjab4xW3ARlWZq
wwKCAQBbs6Gt0843wVJcNkrQiOTLL8HewV8N9LNf3miAO4/2VgSYyO+Y8xjE/xyt
eQRE9ZKrqKb1L192w7vEgd7+a9vwoLlxeLbV7oUwG+gp+aVt5GNPW05mAgUynsEn
zNrdC+Ng9GkXzygYfky2b/ErNicJfQtoubfCSvcHL2R9Fk6dY3B9FdjkJvfjv/SL
se+aEl3AWJnAnpM3KtjY5qje+R/m6f5NkHvECiV/rCu0Bz8+gGpo4MlFKtsz27by
3NyzfWoQ0h6Fcz+AbrxqdMw3JRLoIRhhlC7XMnaCP2hphOd9exTRJYBvRSXTSyDY
q4gl5hM8H5F0iWcKq25nZYHamz7f
-----END PRIVATE KEY-----

B.4.2. Valid

Hex encoded ciphertext:

041f9bbe8372454ffadce0befc99ad4bd01cdbe016dffb5d0d84ee879e7ff64a
25dced4d504f48a14b700a34a1c148e4b4028eecc9158cf5e3469daad1dddb21
57b176dcf5716627aebd6072b965cfb67b42b149e88c0a8dc54703ec244637b3
039b80c06d1d4968aeb838f3afcd3dc675fc90736cf0f0adaba182e27a19a729
4cef500ed1fcca324c3ffa6ff02bd749a4a66f18da138d53ff549c4e6b3fff0b
9ee2029ea8293fcd72a03e1dee4445629979be7fd65e5dbe5e6e77ec2aa87879
a01c3e2502af0cc4bc04b637d17b175d92b4dcd70cfe83b6c1d4f91c61e911b8
1cb6d60b99146f17937d127054521b132acb9ca95cbc6f80cd926d709bd7219d
48ce7378e4416328a49be6c773dbdeb00ce23dc91f25f4807526cf4de8dd3fbb
13ae1011eefe3a2aad9982f6c9268883b4633057d119c97c1178aae671859af0
a488ab7b841c6583572d1261137a4292ba1c1caa4baf808be42075940d7e612e
fa56ac9a7ffff3e7dbf10b6193ea14c4841fc5c43e031a69ddbd79118ccd92b1
6a6ef66c7404c001ebc6402301a90248eb6562b7a2f549f52c058109dd5f2617
877719ad13e81d19425a4cd5cf9e56e8538f5cb09b6f7ab646910b8c2ff6bf8c
f17abdb4758bfba80df9f643950a6d4a8f60c6872700e5919d36503486667e43
28a6b74b774eb483a9922706baa7f456644781cc0d78add9024e28da2d5e9d81

ASCII encoded decrypted message:

lorem ipsum

B.4.3. Valid empty

Hex encoded ciphertext:

```
31529676f990b750e8b742babe933346bd77610e7ea74a6b815bb06ee3c91a37
6848a4d8b148c2882d65ca0213e68600354b68b7790110ed744e34786fa5f7b7
03144a12f7f7a4d5402ef048f248fec83dede7f931bcb872054fd25cfe427984
0352f2c495bcb511cf20269d8177baba474e790a2d16b655c4e07b28ad6a61e8
1ce5db3845cb5395aa4affa413e3f1bdecf3fa0aa3073b40d23e9bb0aa7cf359
5bd73d1d38a2661d70a8e4ef525fd446f496ddbe5413657c06d51464977421e0
57387a92a5b9428d02d1a8fb0d50a1623e1b1d0685c371db24b63dd6a4aebf0e
31c637997ae26dbd3441b9f7dce164d948a156aee8c9dc4049286244c85178f7
b4f49d3ba0870f7c71f8f546a902a559fd860ed61e550c4143907d118a15f317
e9ccd84ce3893db372fff1d9a9e5182258d9f9d840a6b75b1618ce4999734d14
7f5334763c0b87ca1c8e57c6f923cff9c7107eea45cf5d4fae0ec0b94e892a6f
6790b07f9e24bff041620b1fb47bcefb956ba61949fea02db6ec785b4070e84f
390234d83d17dbf89819d5c6b52a488f36cd9f8d81e1811bbadbd6756b9ec5fb
dcc1d32728efc8ef4318463996c0f829f9064436580f9502af97b1e40c854b0e
f0a20da4368e5b94327b62c27a8a1cf6103bc780e06e5ee5232624fe5bee59fe
79a3956a7d782380a3398eab6d11f618fc2c698eb28a6029f846c378ecf2d102
```

The result of decryption is a message of length 0.

B.4.4. Valid with ciphertext starting with a zero byte

Hex encoded ciphertext:

```
00c40ab6440e544ced2c0bbd3f6db05fd0068eb8e9ed83099cf5843ea3d2cfd6
adf7ede1c61f1974a5696f503205346b51d7b00eca20432f0082abf2a3cd6743
3c5f860b32f1f6fe9985dcec65e7f19949999e142bc76ce5b2dcb80615d379e9
715ac24ef77fbbe3d969131c0f39666b3ad641fac669d53542ae9389d86f6f28
fe63ba272b1f6bd015b4187b6e2014fb74fc32bf4a2e48abfbfc0470956c7379
1c1d81db6c024f4d1cc81ba01be114b41ddb95572a832086c33fcaf90b32358b
5b13ef0bea74a6dd01d5c351a231d4e7d15d121cffff023e0c2bdac11c982fb2
419955a495e8dac3d84cbd8d71ac380072a02bb026ed61151f0d202b3133e603
0a2db7560ca926de3927f1ed578522edac441071498b4565fb0c8071886b79fc
9e8d8c14bec1d7a6106441c16e9b2bd3090dad1fe82d0e43f40b036f00506cea
36af61d5f10aac0d0591c12107ceb8999787e34943974025b9c47d16cb523a58
c5828da975fa74e431ace2dcf934e21613f877f65c67b729ca79ee25f27ea07a
74ff38c3b25bcaf22dfe2e9e5d0adecaa01d98d55b5cd0b20f80e672d1bedf4a
cda32184db0f5ce89dd64f538f038cbefb625347ef77a16ab9d48dbee9549f1c
8d55b5470a5c6693bec146f0190bbca5b93a66667150a661851add729b24dca7
8a1f051093ecd40a0ac43a593101f579597b7638065bdff4191bdc0cae12c052
```

ASCII encoded message:

```
lorem ipsum
```

B.4.5. Invalid decrypting to empty

Hex encoded ciphertext:

```
577d6457be536bf1fac80993f5f76e797607227a42e325deb246bf8bf475e1d1
819c5dfb6d288eb131ead32238b7a8796b76517e78f979b34f5c2272ae4d40e6
0b265ad3c897ecc4d26587c8ac32db431ed8d2996d64edadf7719baa4b4292b3
4e042f26693c90f04addffe8ba0e3c3f1cbabafbfd2e3a6bcd9203e9a1cebb1c
968caa73430556ea5743ecfa49edab656bfeadf7c114105f3e222fe60983d55b
c48cb738c5a307353281c573be6ffc69630185ae6de695c35fbf8cbc24b5590f
5e511adb68a38a37bc6dc74a5052e5063f20c30d2f331dafb07797f9e577cb3d
18280f318fe2a6116cef6846b7a8384663a5aaac32693b9b159f506d8812f76a
1ee405abb1e5e439a0aee4d1b51f435ea2d043fe4f4ef1a6ccf069201ceb7978
139eed579b01bcb5b4e525cce6b179f72fcb6aabbd916538d580ce3e1ed57a4d
3433efd826fffd75d8fdc43de69afab66dc32a4f4f81b14c0650a097709265e2
6b57f233008013b275e7b58817b02f4473a99bb48ec7d9562bc78d1f032340d3
083b762f19fb204be7f26d7ceef6c7ff712479a6cdf18123586b87c2751493f1
bb2585626b1aa486a2ec50a197728cf24d3968e6d9b9bdfedfda91db292abdef
fda2334c85fd359e65e7e662193b6feb5df6d70c7727880150c785d809ba3ffc
e3b2701aad313da60cd4affe85e85d32c4fe271b0b107e529a22cffbb01871c8
```

The result of decryption is a message of length 0.

B.4.6. Invalid decrypting to max size

Hex encoded ciphertext:

```
09be60b83b63aa1f2398c6ccd7ba602917dfffbc1a2ee01094221ef7140ba8b3
64b4979b7d068be084d34b70309bc48103d9e7dc76c042d1063ce6d7239d3542
ad511da821c6ad53dcaefdd9e950de889d1d462a5a8bcdda4e2ac579c0dc12ab
684a16a3c4075ec5062db79f95e5b436a8927fe7e3a795152cb6407faedf088a
e4531cca8482348a3c44267b1ea46fa3bfcb4754be735e434c4ef17b84f6fd6c
184e8c4adf91fbf00be6a6ff86351e6fd76c8929ef7fae14113370cbe6ba0181
ef6970855a2cfde5b366a44bf0575e7e5d5354676b5429a6916bb7065c234174
42150f8fa23a1ae284a27008980691eb886b693adb1bf4d38faae5037007900e
47124a695c67b51fe9e1e66153cb32fae0e7370151024fe49b2781e50ac5a31d
1c7de58923c0360ae61f4a3bcfd7839a104c23d95ddeb76bcaf2b1956f81c355
aeb11a46b48762baf072318727e56d4e682a3b71898d3725b00a334c94cb1adc
cea81525b28c2a1c2b82950b93786ea5b893ffe6dc0932a95b064c838d6f04ea
a8334f92b0dec8c7bbc1a5900d7e7680c24e66867ec2b837e30be18083c6fb2b
fc67ace569989171f05d312ce97307b477379837351e9199528e88671a93e558
68a577fb9eae76a2cbf3cc62c6aaabbee88cf6638012554ba92772e923fbe531
fe8aa2036ee7230954ffa24e802a399f531c8cad0a78262442c92089f06b06c7
```

Hex encoded decrypted message:

```
7a57bf4c557abe6bdd45ed471260ec2749e66710b707ff4e4761738dbe2cdc19
267aec5070d2472f53ba0c86e8b607566c871b6c3de28772aa197e369fad293f
8218fd32178ecebe60cc7926e093bbdd629ed6a6a29b4a905eefe443f3621d89
582a41bac7d4f6b77f9c935681e892d17b2261151a4b4244506cfc49ba578b97
5840d88f637321d20c25950640d2b43aa660611cc07d016534324b84bada2244
85488af08a8d54af1698babccd76b87218b074987273dd3746eacd2dee8068d0
c4f5e8f219689d55deea3cccb86e52599cebf3777cab3bfc2da5ae31c9019973
9ba5d5e01816f8f8a21c7f6b6acdada8a2b2ab4f32502d6296a365267a768378
78d204a1b8cc5299708e9eb1b44d9663c75c9b8c17dd8f1a9f04109087e22329
769fcc5fa65506a0f6294d8bf4a362559c497a36b2a5c55ee12213bd9c42a2dc
0f6f35176e194641dcfbb334b13924e6883b4f68c88d016ecdc67bafdeff4b2e
4122777726e06b0e37642d42090be622288ce11a8af4261b5d186092002e7d71
cb43fc2182e4d341c610e6860904cbc526e74292aa1df81dd08d87b51a953c0b
e511346d604aa3a8e87f943db3e951a69cc4dcd1b996d3d8ab595aa9b7ca591a
e2e9b49798a9b98aa1c1c233f938937c88ad675e6297f36ba0fb161a8b42e9ee
9a98d8b781165ed75d30cc55d88e6df5c688c32390
```

B.4.7. Invalid with null padded ciphertext

Ciphertext that decrypts to an invalid plaintext, with the ciphertext starting with a zero byte.

Hex encoded ciphertext:

```
00cd3bf408c742514e30ebfd001708e01f42de439a09d0d4b10044628becd8bc
690c45b967a6a900d6259c7d7d20ad49810c270d2fe1b75ffaa84f50071f64d1
696bce76e96dbcf2af054d77ded54c742aa38fadba52eebf7a5c8b6dbbf01f28
b3ebf4e9b3ca453c92ca8b65771c23671b9da85c51e7a2987395de45b125353d
430820b4c90b0ab9fc29e1c576a0ec35828c99c9880e208ecddf80bdaa9c3474
561e4c5bf67bf5d144b173f4f28edd1064a9dd8cf5f40b7c35e7e4dbeb370cca
bf06efb54e243645b902871a804c27904a620f31ef321af4d1589ca359de8895
24dfb43ddad51ab50617ab79a50a2d20e2325a05c9963602f2a1d7feaffc80d2
b8b88cbd3673531c399342f0bf1c04521d9a441c06593d9e95771129da942d83
bb81c627a50c8baafc2a731d97f2e1f638217f9f7b18f2644578fde5073a54a8
988fda357c64398a95965a7dd35ffdd7eb7495d4d1c9a950a0cd208a68371829
420633cb25e9da9e22b215e50f6ec2714f94ce6abb9a94fbbf6cd18f2fd5bfb7
8076034b88186957428fe0ea5e1dec7be31d753f155e2a8f7f4854a726612f43
ef91f88124011cd32f6841a347dc0411d708db65492ec277b81a335b76a86230
2acebf7db7cac724144c975ed4d91412eeec4cac3b2409696e3678cffcb7756b
de795c55f72da44f325b71dd6ac4e3c1ef512ee8ff24ab316507c6ac60031569
```

Hex encoded decrypted message:

```
59ccd9c2cfe740e9287b
```

B.4.8. Invalid with second to last length

The length of the synthetic message comes from the second-to-last value from the PRF output as the last value is too long for this key size.

Hex encoded ciphertext:

```
09f010936fd77967dd2aa090be72ecdd8cb2b2a9fb954e644f08a1c1fd545000
7ece0b4e8e992831534309419c034b1c4b369269b808a890eac31d5bbfe73737
fd716e117e19638064292539e84dae20ec1d98ecd104e6c88d39d3544d5c43c9
fd4e9f4361c67116516fc2569f96e5df50a63e949d92009a7c865c7dd70ea89c
0002dd697dcd0e0304f8fa0738a3681322e672a9748f9d9eb1b7b8c1646a7bf2
a1967b2aa9793ae0ecdbaa2d601350fd931ea164bc02a33a9c22cee47f2d536d
92e465180b44e310bf044eec49cc54972d22e735123fbee76496a4c768acef89
66ae9e0ed46b753ec47e20a89af0ccf78c266aeb63aeb99d80e4c34bc4608e16
4aeffd93e886accd426e9e1cfee4f912f235940a9b8d4554953bee8753b8fe05
606fe5f46da70394b10f7e71dba3c10becf85a1f4ee619d421ef69fca837d8b1
b42598dea0232668b9c42223d77bae7fbe79a9dbf057eeaa51fceac114661a3b
572130776c36cfff9f134b08907ab2a6f927145806a779bc55bfb4215cb8c85d
ecaec5b15cd0a4be691b13e068a2381c33fb5adbdb564fd8be0938fdd710590c
08f95aa4ea90cc2cea195ef7a344e14715beff2186ad7d898e8f904861ab9133
817a13d8a6af23f83a89e5e1e7f8e163d6190e91a09c11939e3a7d48990b8e3b
5c0e01b773ba683f7df73b2f88746593c010cd9cab2ea3d9af5ebf30fba1ef0c
```

Hex encoded decrypted message:

```
0cfdf702efa4dc6bc686
```

B.4.9. Invalid first byte of padding

Otherwise valid plaintext, but with first byte equal to 0x01 instead of 0x00.

Hex encoded ciphertext:

```
0eb5e0c87c4ffffb760c8fac2a7f5b06f46301ab5b8374a59cc7006aa16e7f38
f27d957a4b475b41975246ffb5f2387ffdb62565411733331a4522a07a70bd40
ffe23f28f457be55a6cd1b6aab8c7127ec4b0c9d653c3d979fbd371854a80727
c0546d9852ccd6220b32a9081f6687fd2262dc806e55964ac799cfba56c0be1a
9fb3d4f46161cc5f251ddf3579a87c48c086da786d953829e513a525a87d8896
61606593600109a98159a91b606f138b9da1b2d427418d50647476fbdd17b521
11ca1a2ff9896167277d158d82ade196ad52ea6d381a63748e0068b160331c9c
b27c20afbeb696c1db16ea145e96a6e54a825c87c10f85b0d92fc299e254aa7f
74c73f17bc704407c31dba9fbda37e31f1231d06744beacb82a0130a9e7d004b
ed7e7036e33f1c89bd8ed0833def8e946efb53a9ea4abda91f1e1ed44e884d8e
45ea692e8f7e2b0f698024c8ed7ade62b251dcb91b249e937a85f823e6978e01
e8f7e9ed9c53f2f16d99afe58d3b77b3818aba64e139b3fcb0174542f348f9d2
c47dc9d78902a28e1a6613397a0f5938b860a3f6cd44e3b74a37ccdeda248cde
ee4a39497d76722e6860127eefdd80f447e69279bf177c1cd6c27ddd3f2f8992
4f21775df4824ece7f2a6f16266772a13669bfc04aa3b998ad03de17bbc27e62
9a495052e3e7f0b7b1c06dd4b429585a663172ba20527ad5186447bb74b5f368
```

Hex encoded message:

```
8bf3b682d8b950055940
```

B.4.10. Invalid second byte of padding

Otherwise valid plaintext, but with the second byte equal to 0x01 instead of 0x02.

Hex encoded ciphertext:

```
714102bbe701855ec564853befd91262dc4cfbb3c417113c0c650b49c6878b10
1a76ba4822bde7ca538a2726c6eb9272a3dbda84119ec107d86d2b3a49d82de9
d176824f1c2d9cd9b432064c45dbfc60f1e71ec2772aef2669e756cae67ed757
b528cccc4ac6f1437a2d4bbedefece1bb5c21381eb4aebc1670c5bd65d408a19
c1329b9d9d236939da58a1285357e910edc83d428d1e5315c81cb070aecc24be
7fde807ce5d4f50dbe14478334c26be91ed4cb7335c63561b1a8c8c67e40844b
465fcf7df6e0df031572682427c62d3cd0c650ec5ef3875fd420516c5cb8089a
34757c81360dda37f7fbbd5ea9c8a54ff29f702741a0d496e268a8934b32cd16
bea2aa2628397097df0ba08545b9b23bc103a08077745239de34eec09f63fdcb
f3aece33a796adbd8dba0705ee3a1092d51f18f195e896b9c35b1a185752c627
55c4d9bf2069db141fddc6755c6927cae8d2811aef492f8324ca555b51a4eaf7
3001ddf88918798df67138a5475fd881b79a158dbe7dd61f241680039a7ae312
8a7b925ec7577cddbe116940e2f50ffa3ed36aee7e46ed6dc5b26e7c5f1d16cd
c199140742d3fe6bb7d2d4b74d0be675a3388c6fd6d09112dfdea93a701486e3
8f28add60fb674ac141b389eb4e09153167596a96d2e6618d98593278d22f560
5041882743ebafbeb18cdae093609ae6852498bedd8fb6b0a18ce358bfc9d6f2
```

Hex encoded decrypted message:

```
630fbcef34c1b72f790d
```

B.4.11. Invalid with zero length PS

Hex encoded ciphertext:

```
1dd961276ba110ba4fdaf4f177780cbcb6373d2ae6769417a32c9b02eb00a48b
427e7e6edefac562fc42c5e2216c885af0f76bcfaf3da4db54ce9db0e22c498c
71e146561c1bbf7ff6246ffc6b0bfcd107830790c07ed9aeda70f2ead9e95799
2e3ed781e054f336e2ab08110f14ca3be11b92d77b0048c334d97d61c8bc4d82
db9c7236973d9ba4da066643440333b5a9e905e799f966c9164907866d9e6af2
d7f83466fe8409f24d5c9b3a06614af620087e838039355e65bde8f3ab7a8e06
943613e00fd143e21ec2684ac07e1dfdef85da32188a97e7585a667f89694111
6250f30e31bc0e2b20e536366e225759b12cdd578df18799dfee20b529189ad4
9789f60ce3649431889d740641df90d1dc372a62b4d77f9f5f9677b4f96770b0
7107ef37afe4e4dd6af4838ae18b61ce953eedcfc95d081951cbe0b097c6c334
489cf46fbba26009d7ebc8ecaa0b155ac60e1f40cc381cd9c85ac7fb25f8458d
964bd6e1fc85d6e18bc0fa5491c6995fa7225dcfe43d6a12bde24343d16ce421
46cf26a5ec1fd7fae5d829e1f274819a10445ac106f5a517ab89e62455c02469
271812be18e972af7d5ba3079dc427f1b7ff7eda2d0cba55f28edef280f5d924
3466d1c6f9c4b671777c75c9464e571e7115d97d0d86712781e6346a4472c98d
a976032ff1073850e7304b6bb4b60b019a7bd870c8d3fb26b212ddfd889b9fae
```

Hex encoded decrypted message:

```
778f208b90bef0f260a8
```

B.4.12. Invalid with null byte at the eighth byte of padding

This creates a plaintext with PS of 7 bytes.

Hex encoded ciphertext:

```
3d4477615f8b49c9453d31a5d5a228610d6d3e476737c163725106fa386440ff
23a9139f57977c09a1e885ddf2f180ddf0f0b0502ef60f0ff53d2ba444f03228
20f11acdd70e48543bd8ddd40d00ad9214d03487b265910cf423dea905af9088
36bf56ef872493b686aa15731714d1f0280e5cddad24ada53374c8aad063184d
62bbdc00efd4839b8f06de6c258d26149480b2fad6fcb2fc97cd78fb60305aa2
54cf1186fa134741a2340dc1d5243423c82c442f3afd915241f317607a2e2236
64601932e7b967d6793a7fde2819d475e2b8ab0117e3cc4854063a0c1ff1f5cd
9c9dc3e6c993d8861c11ee7155dd0ba2d4f47ff0ffc9c7fc8a891284789e5988
806fe7b5f5ec5783861fef756ef7339380215de11337ec8f2379b293c3cacf2f
81691c1ad75c9223a6c8edfb0451373d0b759d9d701f547b3c46d3fe1f3d24d3
3447a52bb55b9dd7050c41d11f3108ddccd5738a072905eb48067350e76a65f0
d274b0f4bde004cb673b715d4ab01dded4b6b7f69e133135ffcad4b1776e3610
830ae55a98d23610256865e305153ad7319ff905c16453297f5ebe64b8857bff
69c750d338368f6a55d73ae363a516fa4bf719cd01d46b609c134e3508d9616f
495ff2c869db7ad146376b102529c26407d8ffdedbfb005b7b220e0dcb089da3
682c9af7c278472cc19b6523b09661fe6f165bd7b7765ca2524eefa3526397b0
```

Hex encoded decrypted message:

364cfecf7e70a0829f28

B.4.13. Invalid with padding separator missing

Hex encoded ciphertext:

44c5b648e960aed2ad38497b6af118577a7978db288c0019cb6f8818578021ca
0d782c4e87bd6c3a73ef89e379311f0d449410c336bffdc9d970f995e7b59789
10c230b1ef11c06cc5de6ee79ca2f85f6f14e5c42cbb8269d40c032b91783755
fbfa7b87f16790bfea91933c67d2499a3d815cb70142285449757e606a438752
b803b0928c28dab4fd21125e5b79af04fe912fb444d32039e9e0e10210fbe017
4f43e2833ab862e4370b007025a919cf7b9c11241cf95ab10a9baa44a7ce7fa6
e802c5b8e5c466dba52704fc2325317526f36d25842e130fcdbbc3c631a1e4c6
7d23ffcb2218065863c178526616e8429916dff9101baf71857901bdaf305d26
9c944994f9cf0e02a5499432324b90a62c3c78bd7a7821420a11c43d0a80ee39
68ab8a363d6e6476f5424ae98bd59352aa9842e8f42cb0a34da68eac9dd16cf7
04573007efc4b3fc97161c342c836ff781c331c306f61052d3877ab190e307f4
f8d63c3bc8f6ab9a6920f9d1b9be482d3096b0b02447f53b0f974693e2a49b73
3684d8e33d7dfbb60d3b1aae02c222c395209b1e2647e7fcbf3c44cdee9c7332
9ce9ed255e011847d6d2e119d252d57c72572b2a309472f059cdae1f24e9cf1f
fddebc70b8b252229c7c0adfa763bda2243840bc9a553b6a4ea7737c0a002261
b36e198fca4314f659ca1071aceb2668079d663e4ed40e15e10d764aa8cb0c68

Hex encoded decrypted message:

750d91268328712552cd

Author’s Address

Alicja Kario
Red Hat, Inc.
Purkynova 115
61200 Brno
Czech Republic
Email: hkario@redhat.com