# On the Security of the PKCS#1 v1.5 Signature Scheme

Tibor Jager[1] Saqib A. Kakvi[1] Alexander May[2]

September 10, 2018

[1]Department of Computer Science, Universität Paderborn
{tibor.jager,saqib.kakvi}@upb.de
[2]Hortz Görtz Institute, Ruhr Universität Bochum
alex.may@rub.de

**Abstract**

The RSA PKCS#1 v1.5 signature algorithm is the most widely used digital signature scheme in practice. Its two main strengths are its extreme simplicity, which makes it very easy to implement, and that verification of signatures is significantly faster than for DSA or ECDSA. Despite the huge practical importance of RSA PKCS#1 v1.5 signatures, providing formal evidence for their security based on plausible cryptographic hardness assumptions has turned out to be very difficult. Therefore the most recent version of PKCS#1 (RFC 8017) even recommends a replacement the more complex and less efficient scheme RSA-PSS, as it is provably secure and therefore considered more robust. The main obstacle is that RSA PKCS#1 v1.5 signatures use a deterministic padding scheme, which makes standard proof techniques not applicable.

We introduce a new technique that enables the first security proof for RSA-PKCS#1 v1.5 signatures. We prove full existential unforgeability against adaptive chosen-message attacks (EUF-CMA) under the standard RSA assumption. Furthermore, we give a tight proof under the Phi-Hiding assumption. These proofs are in the random oracle model and the parameters deviate slightly from the standard use, because we require a larger output length of the hash function. However, we also show how RSA-PKCS#1 v1.5 signatures can be instantiated in practice such that our security proofs apply.

In order to draw a more complete picture of the precise security of RSA PKCS#1 v1.5 signatures, we also give security proofs in the standard model, but with respect to weaker attacker models (key-only attacks) and based on known complexity assumptions. The main conclusion of our work is that from a provable security perspective RSA PKCS#1 v1.5 can be safely used, if the output length of the hash function is chosen appropriately.

# 1 Introduction

The RSA PKCS#1 v1.5 signature algorithm was originally specified in version 1.5 of the PKCS#1 standard (RFC 2313), and is still contained in the most recent version (2.2) of PKCS#1 (RFC 8017). It is used in countless important applications, such as X.509 (RFC 4055), S/MIME (RFC 3370), PGP (RFC 4880), IPSec (RFC 4359), all TLS versions up to 1.2 (RFCs 2246, 4346, 5246), JSON Web Signature (RFC 7515), W3C's XML Signature,[1] and many more.

PKCS#1 v1.5 is an attractive choice in practice, because it is RSA-based and extremely simple, therefore it is quick and easy to implement. Its simplicity makes it less prone to implementation errors, which is a highly desirable feature for cryptographic algorithms. Furthermore, it is compatible with legacy hardware and software implementations, which is relevant for applications, such as TLS, where interoperability is important. Another nice feature is that verification is extremely efficient, and significantly faster than that of DSA or ECDSA. Concretely, the OpenSSL benchmark

```
openssl speed rsa dsa ecdsa
```

shows about 1170 signature verifications per second for RSA-2048, while EC-DSA achieves only around 350 verifications/s for nistp256, nistk233, and nistb233 curves, and DSA-2048 achieves only 100 verifications/s.[2] Similar results were obtained by the Crypto++ 6.0.0 benchmarks, even with precomputations to speed-up the verification in DSA.[3]

**Security of RSA PKCS#1 v1.5 signatures.** Even though RSA PKCS#1 v1.5 is still the most important digital signature scheme used in practice, we do not yet have any formal evidence of its security, provided by a rigorous reduction-based security proof under any standard complexity assumption. We do not even know any security proof under a non-standard but plausible interactive assumption, apart from the trivial assumption that the scheme is secure.

Due to the lack of security proofs for PKCS#1 v1.5 signatures, some standards allow to use PKCS#1 v1.5, but recommend RSA-PSS [BR96] instead. This includes TLS 1.3, X.509v3 (RFC 4055), and PKCS#1 itself, since version 2.1 (RFC 3447):

> "Although no attacks are known against RSASSA-PKCS#1 v1.5, in the interest of increased robustness, RSA-PSS is recommended for eventual adoption in new applications." (RFC 3447)

While RSA-PSS has the advantage of being provably secure, it also has several disadvantages. Most importantly, it is much more complex than PKCS#1 v1.5, and thus more difficult to implement and maintain, and it is more prone to implementation errors. Furthermore, it requires randomness, which may be difficult to generate securely on some devices, while PKCS#1 v1.5 is deterministic and has unique signatures. The latter automatically yields security against subversion attacks, as shown in [AMV15], if the RSA public key $(N, e)$ defines a certified trapdoor permutation in the sense of [BY93, KKM12].

**The difficulty of replacing PKCS#1 v1.5 signatures with PSS** It has turned out to be extremely difficult to replace PKCS#1 v1.5 with PSS in practice. Hence, even though several standards prefer PSS, one has to expect that PKCS#1 v1.5 will remain deployed in many applications and retain its huge practical relevance for a long time. For instance, let us consider the recent efforts to remove legacy cryptography from TLS in

[1] See `https://www.w3.org/TR/xmldsig-core1/#sec-PKCS1`.

[2] The experiments were executed on an Apple MacBook Pro running MacOS 10.13.4 on an Intel Core i7 3.3 GHz CPU.

[3] See `https://www.cryptopp.com/benchmarks.html`

TLS 1.3. Although TLS 1.3 has, in principle, moved to RSA-PSS as a preferred algorithm, this decision was preceded by intense discussion[4]. It was pointed out that in many applications it is impossible to remove PKCS#1 v1.5 signatures with a simple software update, because the signing algorithm is implemented in hardware, and it is considered infeasible that this hardware will be replaced at large scale within a foreseeable timeframe. Furthermore, X.509 certificates require PKCS#1 v1.5 signatures anyway, for interoperability reasons. Hence, a TLS implementation now has to implement *both* schemes. The need to support two different RSA signature schemes makes implementations more complex and requires the maintenance of more security-critical code. Furthermore, due to a lack of RSA-certificates that can exclusively be used with RSA-PSS, TLS implementations also accept "legacy" PKCS#1 v1.5 certificates for PSS. Formally, this voids the known security proofs for PSS, since they consider a setting where RSA public keys are used *exclusively* for PSS[5].

In summary, even though the importance of using provably-secure algorithms is increasingly recognized by standardization bodies and practicioners, it turned out to be extremely difficult to replace PKCS#1 v1.5 signatures with provably secure alternatives like PSS at large scale, due to backwards compatibility and interoperability requirements. By providing rigorous security proofs for PKCS#1 v1.5 signatures under standard cryptographic hardness assumptions, we show that PKCS#1 v1.5 signatures can be used securely in practice, if instantiated in a way such that the bounds on the length of the hash function output and the RSA modulus from our security proofs apply.

**The difficulty of proving security of RSA PKCS#1 v1.5 signatures.** The main obstacle in proving security of RSA PKCS#1 v1.5 signatures is that a deterministic padding scheme is used, which makes standard proof techniques not directly applicable. More precisely, a PKCS#1 v1.5-signature $\sigma$ with respect to an RSA public key $(N, e)$ and message $m$ is

$$\sigma = (\mathtt{PAD}||z)^{1/e} \bmod N,$$

where $\mathtt{PAD}$ is a long constant string and $z = \mathsf{H}(m)$ is a relatively short hash of the message. Concretely, for a 2048-bit modulus $N$ and $\mathsf{H} =$ SHA-256, we have $|z| = 256$ and $|\mathtt{PAD}| = 2048 - 256 = 1792$. Hence, the message representatives $\mathtt{PAD}||z$ lie in an extremely small subset of $\mathbb{Z}_N$, and the set of signatures $\sigma$ such that there exists $z$ with $\sigma^e = \mathtt{PAD}||z$ is devoid of any known algebraic structure. The combination of these two factors means that standard proof techniques are not immediately applicable, because in standard security proofs for RSA-based schemes [BR96, Cor00, Cor02, KK12, KK18], one samples a random signature $\sigma$ and computes $\sigma^e$ as the message representative. However, since the set of valid message representatives is very small, the chance of obtaining a valid random signature that satisfies $\sigma^e = \mathtt{PAD}||z$ for some $z$ is very low.

**Our results.** We present the first security proofs for RSA PKCS#1 v1.5. We prove full *existential unforgeability under adaptive chosen message attacks* (UF-CMA) in the random oracle model [BR93]. This is the same security level that other practical signature schemes, such as RSA-PSS or RSA Full-Domain Hash [BR96, Cor00, Cor02, KK12, KK18], provably achieve. We give two different results:

1. The first is based on the standard RSA assumption, and has a linear security loss in the number $q_s$ of signature queries made by the adversary. If the RSA public key $(N, e)$ defines a certified trapdoor permutation in the sense of [KKM12], then this was shown to be optimal in [Cor00, KK12, KK18],

[4] See `https://www.ietf.org/mail-archive/web/tls/current/msg19360.html`, for instance.
[5] As pointed out in `https://nikmav.blogspot.com/2018/05/gnutls-and-tls-13.html`

because in this case RSA PKCS#1 v1.5 has a *unique* signature for each message. Furthermore, in this case our proofs immediately imply *strong* existential unforgeability in the sense of [SPW07], as well as resilience against subversion attacks in the sense of [AMV15].

2. The second is based on the $\varphi$-Hiding assumption [CMS99]. This follows a line of recent results [KOS10, KK18, LOS13, KPS13, Seu14, SZ15], which use lossy trapdoor permutations [PW08], to prove security. Lossy RSA keys are indistinguishable from injective keys under the $\varphi$-Hiding Assumption. In this case, we get a tight security proof that is independent of $q_s$. However, in this case we only get UF-CMA-security, but not *strong* UF-CMA, and no provable security against subversion attacks.

Both proofs are based on a new technique, which introduces an **Encode** algorithm that makes it possible to embed *arbitrary* elements of $\mathbb{Z}_N$ into correctly PKCS#1 v1.5-padded strings modulo $\widehat{N}$, where $\widehat{N} = NR$ for some additional prime factor $R$. The signature scheme is defined modulo $\widehat{N}$, while we consider hardness assumptions modulo $N$. Thus, if $N = PQ$ is the product of two primes, then our proof considers a variant of PKCS#1 v1.5 signatures with a modulus $\widehat{N} = PQR$ that is the product of three primes. We explain below how this results can easily be lifted to the standard two-prime setting.

Finally, in order to complete the picture, we also give security proofs in the standard model, but with respect to very weak security notions. In this case we consider *no message attacks*, which are also know as *key only attacks*. In this scenario the adversary has only the public key and must forge a signature. As with the random oracle case, we prove two different results.

1. The first result is also based on the $\varphi$-Hiding assumption. It is well know that when using lossy keys, the RSA function gives an $e$-regular lossy trapdoor permutation, that is a permutation that is exactly $e$-to-1. This means that the set of $e^{\text{th}}$ residues is much smaller. We argue that the distribution of these residues is such that with high probability there are no message representatives that are $e^{\text{th}}$residues.

2. The second result relies on a variant of the Approximate $e$-th Root assumption (AER), which is a generalization of the RSA assumption. Essentially we give the adversary more freedom, as he does not have to find an $e$-th root of a specific value, but an $e$-th root of any value *sufficiently close to* the given target. We use this to cover the entire set of message representatives and the proof follows from there. More details are discussed in Section 4.2

**Implication for practical instantiations.** Our UF-CMA security proof considers PKCS#1 v1.5 signatures with moduli $\widehat{N} = PQR$ that are a product of three primes. In practice, PKCS#1 v1.5 signatures are usually used with moduli that have only two prime factors. Firstly, we point out that the PKCS#1 standard defines $\widehat{N}$ as a product of *two or more primes*, such that the setting considered in our paper is already standard compliant. In order to avoid an additional assumption, one can simply use a modulus of the form $\widehat{N} = PQR$, which is compatible with PKCS#1. In any case, we can easily lift this result to the standard setting where $\widehat{N}$ is the product of only two primes. To this end, we simply make the additional complexity assumption that for a suitably large modulus $\widehat{N}$ breaking the UF-CMA-security of PKCS#1 v1.5 signatures with $\widehat{N} = PQ$ a product of two primes is at least as hard as breaking PKCS#1 v1.5 signatures with $\widehat{N} = PQR$. We consider this as a very plausible assumption, since more prime factors should only make it *easier* to break the signature scheme, and we prove security even for more prime factors. With this additional assumption, we immediately obtain a security proof for the standard case where $\widehat{N}$ is the product of two primes.

Furthermore, we remark that our proofs work only with a hash function with an unusually long output. More precisely, if we use the **Encode** algorithm described in Section 3.1.1, then we can prove security for

PKCS#1 v1.5 signatures instantiated with an $2n$-bit modulus and $\ell$-bit hash function, under the assumption that breaking the RSA (or $\varphi$-Hiding) assumption is hard with respect to an $n$-bit modulus $N$ with $|N| = \ell$. Concretely, for signatures instantiated with an 2048-bit RSA modulus $\widehat{N}$, we can prove security under the 1024-bit RSA assumption, with 1024-bit padding and 1024-bit hash function. Here we note that being able to deal with very long paddings of 1024 bits and more goes significantly beyond what is achievable with prior proof techniques. Furthermore, there are several practical ways to instantiate hash functions with long output of 1024 bits or more, even with solutions that are already standardized:

- The PKCS#1 v2.2 standard (RFC 8017) itself already specifies a way to turn a cryptographic hash function into a function with arbitrary output size. The construction is called `MGF1` (Mask Generation Function 1) in Appendix B of RFC 8017. This function is a building block of the 2-round Feistel network used in PSS signatures and OAEP encryption. It is basically a standard hash function, which is repeatedly applied to the input and increasing counter values, until the total output has the desired length.

- The Keccak construction [BDPA11], which is the basis of the SHA-3 hash function, allows to specify a longer output length. The NIST standard specifying SHA-3 [FIPS 202] also specifies Keccak-based *extendable-output functions* (XOFs), called SHAKE128 and SHAKE256, which have arbitrary length output.

Thus, even though our proofs do not immediately apply to PKCS#1 v1.5 when instantiated with standard hash functions, such as SHA-512, we show that it is still possible to instantiate PKCS#1 v1.5 signatures in a meaningful way, and based on standardized constructions, such as `MGF1` from RFC 8017 or the XOFs SHAKE128 and SHAKE256 standardized by NIST in FIPS 202.

We recommend that future versions of the PKCS#1 standard allow the use of such hash functions with longer output, as a compromise between the simplicity of classical PKCS#1 v1.5 signatures (but without any provable security), and the significantly more complex RSA-PSS scheme (with provable security).

**Idea of our Encode algorithm.** The main novelty of this paper is the idea to use an **Encode** algorithm to embed numbers modulo $N$ into strings with correct PKCS#1 v1.5-padding with respect to some larger integer $\widehat{N} = NR$, where $R$ is a prime number chosen by us. The **Encode** algorithm exploits this knowledge of the part factorization of the modulus $\widehat{N} = NR$. Concretely,

$$(\hat{y}, s, z) \leftarrow_{\$} \textbf{Encode}(N, e, 1, \texttt{PAD}, R)$$

is an efficient algorithm that takes as input an $n$-bit integer $N$, an exponent $e$, $y \in \mathbb{Z}_N^*$, a padding pattern `PAD` and an $r$-bit prime $R$. It outputs $(\hat{y}, s, z) \in \mathbb{Z}_{\widehat{N}} \times \mathbb{Z}_N \times \{0,1\}^\ell$ such that $\hat{y}$ has to correct form for a PKCS#1 signature mod $\widehat{N}$, with $z$ being our message hash. Additionally $s$ will be an $e^{\text{th}}$ root mod $N$. Using this and the knowledge of $R$, we can compute an $e^{\text{th}}$ root modulo $\widehat{N}$. Additionally, we can embed an RSA challenge in our signature if we replace 1 with our random challenge $y$. In this case, we cannot simulate a signature, but given a forgery, we can solve our RSA challenge. With these two uses of the **Encode** algorithm, we are able to prove UF-CMA security of PKCS#1 in the Random Oracle Model.

We remark that the above proof sketch is slightly different from our actual proof. In fact, we use the more sophisticated approach of Coron [Cor00] to embed $y$ in multiple hash values. This achieves a better tightness bound, where the loss is only linear in the number of signature queries $q_s$ and not in the number of hash queries $q_h$. This approach is known to be optimal for unique signatures [Cor02, KK12, BJLS16, KK18]. However, this is only a minor technical difference, and the main proof idea remains identical.

**Instantiating the Encode algorithm.** We instantiate **Encode** based on elementary modular arithmetic. Our algorithm works for both the RSA-based and the $\varphi$-Hiding-based security proof.

**Related work.** At the CRYPTO 2006 rump session, Bleichenbacher presented a low-exponent attack on RSA-PKCS#1 v1.5 signatures. This attack was later described by Finney in a posting to the OpenPGP mailing list.[6] The attack exploits a vulnerable implementation of a flawed signature verification algorithm. In this paper, we consider only forgeries that are accepted by a correct verification algorithm.

The RSA-PSS scheme and RSA Full-Domain Hash signatures have already been mentioned above. There are several further signature schemes that are based on the RSA [HW09, BHJ+13] or the strong-RSA [GHR99, CS99, Fis03, Sch11, HJK11] assumption. None of them is widely used in practice, as they require the generation of relatively large primes, not only for key generation, but also to compute signatures. Hence, they are computationally significantly less efficient that RSA-FDH or RSA-PSS.

The PKCS#1 standards also specify encryption schemes. The RSA-PKCS#1 v1.5 *encryption* scheme is very similar to RSA-PKCS#1 v1.5 signatures, except that a probabilistic padding is used and the RSA *public* key is applied to the padded message, instead of the *secret* key. This scheme is also very widely-used in practice, e.g., in all TLS versions up to 1.2. While there are no known attacks for correctly-implemented RSA-PKCS#1 v1.5 *signatures*, there are many examples of chosen-ciphertext attacks that break PKCS#1 v1.5 *encryption* [Ble98, CFPR96, CJNP00, KPR03, DLP+12, BFK+12, MSW+14, ZJRR14, JSS15a, JSS15b, BSY17]. The RSA OAEP encryption scheme due to Bellare and Rogaway [BR95] uses a similar Feistel-like message encoding as RSA-PSS. Its security has been extensively studied in [BR95, Sho02, KP09, KOS10, FGK+13].

# 2 Preliminaries

## 2.1 Notations and conventions

We denote our security parameter as $n$. For all $n \in \mathbb{N}$, we denote by $0^n$ and $1^n$ the $n$-bit string of all zeroes and all ones, respectively. For any set $S$, we use $x \in_R S$ to indicate that we choose $x$ uniformly random from $S$. All algorithms may be randomized. For any algorithm $A$, we define $x \leftarrow_\$ A(a_1, \ldots, a_n)$ as the execution of $A$ with inputs $a_1, \ldots, a_n$ and fresh randomness and then assigning the output to $x$. For deterministic algorithms, we drop the \$ from the arrow. We denote the set of prime numbers by $\mathbb{P}$ and we denote the set of $\kappa$-bit integers as $\mathbb{Z}[\kappa]$. Similarly, we denote the set of $\kappa$-bit primes as $\mathbb{P}[\kappa]$. We denote by $\mathbb{Z}_N^*$ the multiplicative group modulo $N \in \mathbb{N}$. We will use game based proofs and will denote by $\mathsf{G}^{\mathcal{A}} \Rightarrow 1$ the event that the adversary $\mathcal{A}$ wins game $\mathsf{G}$ i.e. the **Finalize** Procedure outputs 1.

## 2.2 Digital Signatures and their security

A *digital signature scheme* is a triple $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ of algorithms which are defined as follows:

- $\mathsf{KeyGen}$ takes as an input the unary representation of our security parameter ($1^n$) and outputs a private signing key $sk$ and a public verification key $pk$.
- $\mathsf{Sign}$ takes as input a signing key $sk$, message $m$ and outputs a signature $\sigma$.

[6] See `https://www.ietf.org/mail-archive/web/openpgp/current/msg00999.html`.

- Verify is a deterministic algorithm, which on input of a public key and a message-signature pair $(m, \sigma)$ outputs 1 (accept) or 0 (reject).

We say that Sig is correct if for all public key and secret key pairs generated by KeyGen, we have:

$$\Pr[\mathsf{Verify}(pk, m, \mathsf{Sign}(sk, m)) = 1] = 1$$

.

We will consider two different notions of security for digital signatures, namely *Unforegeability under No Message Attack* (UF-NMA) and *Unforgeability under Chosen Message Attack* (UF-CMA). Recall that the distinction between the two security definitions is that in UF-CMA the attacker has access to a signing oracle, whereas in UF-NMA it does not. UF-NMA is sometimes called a "key only attack" in the literature. Additionally, we only consider UF-NMA in the standard model and UF-CMA in the random oracle model, and therefore extend the latter with an additional **Hash**$(m)$ oracle.

Game UF-NMA(StdM)

**Initialize**$(1^n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KeyGen}(1^n)$
return pk

**Finalize**$(m^*, \sigma^*)$
if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$
    return 1
else
    return 0

Figure 1: Game defining NMA security in the Standard Model.

Consider the UF-NMA security experiment in Figure 1, when executed with a signature scheme $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. We say that Sig is $(t, \varepsilon)$-UF-NMA secure if for any forger $\mathcal{F}$ running in time at most $t$, we have:

$$\mathbf{Adv}^{\mathsf{UF\text{-}NMA}}_{\mathcal{F},\mathsf{Sig}} = \Pr\begin{bmatrix} 1 \leftarrow \mathbf{Finalize}(m^*, \sigma^*); \\ (m^*, \sigma^*) \leftarrow \mathcal{F}(\mathsf{pk}) \end{bmatrix} \leqslant \varepsilon$$

Consider the UF-CMA game in Figure 2, executed with scheme $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. We say that Sig is $(t, \varepsilon, q_h, q_s)$-UF-CMA secure if for any forger $\mathcal{F}$ running in time at most $t$, making at most $q_h$ hash queries and making at most $q_s$ signature queries, we have:

$$\mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F},\mathsf{Sig}} = \Pr\begin{bmatrix} 1 \leftarrow \mathbf{Finalize}(m^*, \sigma^*); \\ (m^*, \sigma^*) \leftarrow \mathcal{F}^{\mathbf{Hash}(\cdot),\mathbf{Sign}(\cdot)}(\mathsf{pk}) \end{bmatrix} \leqslant \varepsilon$$

## 2.3 RSA PKCS#1 v1.5 Signatures

We now describe the RSA PKCS#1 v1.5 scheme, see Figure 3. We note that in PKCS#1 version 2.2, the recommended hash functions are SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA512/256. Additionally, MD2, MD5 and SHA-1 are included for backwards compatibility.

Let us point out that our description in Figure 3 deviates from the original scheme in a few minor ways. Strictly speaking, the string `PAD` is not included in the public key, but is publicly known. We included it in

Game UF-CMA(ROM)

**Initialize**
$(pk, sk) \leftarrow_{\$} \mathsf{KeyGen}(1^n)$
return pk

**Hash**$(m)$
if $(m, \cdot) \in \mathcal{H}$
    fetch $(m, y) \in \mathcal{H}$
    return $y$
else
    $y \in_R$ Domain;
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{(m, y)\}$
    return $y$

**Sign**$(m)$
$\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
return $\sigma \leftarrow_{\$} \mathsf{Sign}(sk, m)$

**Finalize**$(m^*, \sigma^*)$
if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) == 1 \wedge m^* \notin \mathcal{M}$
    return 1
else
    return 0

Figure 2: Game defining CMA security in the Random Oracle Model

the public key for self-containment. Also, we define our signing function as taking the $e^{\text{th}}$ root instead of the $d^{\text{th}}$ power. This is done for keeping a consistent notation throughout the paper, since $d$ is not defined for lossy keys.

The string PAD seems somewhat arbitrary and the reason for this is historical. In PKCS v1.5#1 (RFC 2313), all message were encoded in representative "blocks". These representatives had the (hexadecimal) format

$$\texttt{0x00}||\texttt{BT}||\texttt{PS}||\texttt{0x00}||\texttt{D}.$$

BT defined the type of block, which is 0x01 for signatures and 0x02 for encryption. D is the encoding of the message, which for signatures is the hash of the message prefixed with the hash id $\texttt{ID}_\mathsf{H}$. PS is the "padding string", whose purpose is to ensure our message representative is exactly $n$ bits long. PS is fixed to 0xFF...FF for BT = 0x01. Thus, signature "blocks" have the (hexadecimal) format

$$\texttt{0x00}||\texttt{0x01}||\texttt{0xFF}\ldots\texttt{FF}||\texttt{0x00}||\texttt{ID}_\mathsf{H}||\mathsf{H}(m).$$

The overarching concept of blocks has since been dropped, primarily due to the switch to OAEP encryption, but PKCS signatures still use this format for the message representatives. If we convert this format into binary, it is easy to see that the prefix of $\mathsf{H}(m)$ is exactly our prefix PAD. It is worth noting at this juncture that the identifier strings $\texttt{ID}_\mathsf{H}$ do not have a fixed length, thus we need to take into account their bit length $\alpha$.

Scheme RSA-PKCS1-v1.5

**KeyGen**$(1^n, \ell)$
$P, Q \in_R \mathbb{P}[n/2]$
$N = PQ, \varphi(N) = (P-1)(Q-1)$
$e \in_R \mathbb{Z}_N^*, \gcd(e, \varphi(N)) = 1$
pick hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$
Look-up $\alpha$-bit $\mathtt{ID}_\mathsf{H}$ for $\mathsf{H}$
$\nu = n - \ell - \alpha - 23$
$\mathtt{PAD} = 0^{15}||1^\nu||0^8||\mathtt{ID}_\mathsf{H}$
return $(\mathsf{pk} = (N, e, \mathtt{PAD}, \mathsf{H}), \mathsf{sk} = (P, Q))$

**Sign**$(\mathsf{sk}, m)$
$z \leftarrow \mathsf{H}(m)$
$y = \mathtt{PAD}||z$
return $\sigma = y^{1/e} \mod N$

**Verify**$(\mathsf{pk}, m, \sigma)$
$y' = \sigma^e \mod N$
$z \leftarrow \mathsf{H}(m)$
if $(\mathtt{PAD}||z == y')$
    return 1
else
    return 0

Figure 3: RSA PKCS#1 v1.5

# 3 Existential Unforgeability in the Random Oracle Model

In this section, we give two different proofs that RSA PKCS#1 v1.5 signatures are UF-CMA-secure. The first is based on the RSA assumption, the second is based on the $\varphi$-Hiding assumption. Both proofs are in the random oracle model, and both proofs consider the case where the modulus $\widehat{N}$ used in the signature scheme is a product of three primes. Even though RSA PKCS#1 v1.5 signatures are usually used with moduli consisting of only two prime factors, we stress that all versions of the PKCS#1 standards allow to use moduli $\widehat{N}$ consisting of three prime factors, meaning that this choice is standard compliant.

## 3.1 The Encode Algorithm

The main novel ingredient for our RSA PKCS#1 v1.5 signature UF-CMA-security proofs is an **Encode** algorithm, which enable us to efficiently encode an *arbitrary* integer $y$ modulo $N$ as an integer $\hat{y}$ modulo $\widehat{N} = NR$ for some prime $R$, such that $\hat{y}$ has correct PKCS#1 v1.5 padding modulo $\widehat{N}$. Let us define formally the properties that an **Encode** algorithm has to fulfil.

**Definition 1.** Let

$$(\hat{y}, s, z) \cup \perp \xleftarrow{\$} \textbf{Encode}(N, e, y, \ell, \mathtt{PAD}, R)$$

be an algorithm that takes as input an $n$-bit integer $N$, an exponent $e$, $y \in \mathbb{Z}_N^*$, a padding pattern PAD and an $r$-bit prime $R \in \mathbb{P}[r]$, and outputs $(\hat{y}, s, z) \in \mathbb{Z}_{\widehat{N}} \times \mathbb{Z}_N^* \times \{0,1\}^\ell$ or failure $\perp$. We say that **Encode**

*$\ell$-simulates the PKCS#1 v1.5 encoding* modulo $\widehat{N} = NR$, if all the following conditions are satisfied. We say that **Encode** *efficiently $\ell$-simulates the PKCS#1 v1.5 encoding* modulo $\widehat{N} = NR$, if **Encode** additionally runs in time polynomial in $n$.

1. **Encode** outputs $(\hat{y}, s, z)$, except with negligible failure probability (in $n$).
2. $z$ is uniformly distributed over $\{0,1\}^\ell$.
3. $\hat{y} \in \mathbb{Z}_{\widehat{N}}$ is the PKCS#1 v1.5 encoding of the string $z \in \{0,1\}^\ell$ with padding of size $|\mathtt{PAD}| = n+r-\ell$. That is, we have
$$\hat{y} = \mathtt{PAD}||z.$$
4. It holds that $\hat{y} = y \cdot s^e \bmod N$.

Intuitively, we use **Encode** as follows to prove UF-CMA-security of RSA-PKCS#1 v1.5 signatures. **Encode** enables us to embed an RSA-challenge $(N, e, y)$ modulo $N$ into the UF-CMA-security experiment for RSA-PKCS#1 v1.5 signatures defined modulo $\widehat{N}$. Concretely, to achieve this we will run $(\hat{y}, s, z) \xleftarrow{\$}$ **Encode**$(N, e, y, \ell, \mathtt{PAD}, R)$ to obtain $\hat{y} = \mathtt{PAD}||z$. Then we program the random oracle $H$ with range $\{0,1\}^\ell$ such that $H(m^*) = z$, which is possible because $z$ is uniformly distributed. Furthermore, we also show that **Encode** enables us to simulate correct signatures in a similar way.

As we show in the following, our **Encode** algorithm allows us to create correctly-padded strings $\hat{y}$ with freely adjustable padding of size $\approx |r|$ and relatively large hash value size $\ell \approx |n|$.

### 3.1.1 Encode Algorithm

Consider the algorithm **Encode** as defined in Figure 4. It takes as input an $n$-bit modulus $N$, an exponent $e$, an integer $y \in \mathbb{Z}_N$, a hash value length $\ell$, a padding pattern PAD of size $|\mathtt{PAD}| = n + r - \ell$, an $r$-bit prime $R$, and outputs a string $\hat{y} := \mathtt{PAD}||z$ such that $|z| = \ell$.

**Encode** $(N, e, y, \ell, \mathtt{PAD}, R)$
$n = \lceil \log_2 N \rceil$, $r = \lceil \log_2 R \rceil$
$z := 2^\ell$, $k := 0$
**while** $(z \geq 2^\ell)$ and $(k < n \cdot 2^{n-\ell})$:
  $k := k + 1$
  $s \xleftarrow{\$} \mathbb{Z}_N$
  $z := ys^e - 2^\ell \cdot \mathtt{PAD} \bmod N$
$\hat{y} := 2^\ell \cdot \mathtt{PAD} + z$
**if** $z < 2^\ell$ **then**
  return $(\hat{y}, s, z)$
**else**
  return $\perp$.

Figure 4: **Encode** algorithm

**Theorem 1.** *For $\ell < n := |N|$, the **Encode** algorithm described in Figure 4 $\ell$-simulates the PKCS#1 v1.5 encoding modulo $NR$ in time $\mathcal{O}(n^4 \cdot 2^{n-\ell})$.*

PROOF. In order to verify condition (1) of Definition 1, we have to show that **Encode** outputs a tuple $(\hat{y}, s, z)$ expect with negligible failure probability. By the last line in Figure 4, **Encode** fails in producing $(\hat{y}, s, z)$ iff $z \geq 2^\ell$ for all iterations of the while-loop.

Observe that the value $z := ys^e - 2^\ell \cdot \mathtt{PAD} \bmod N$ computed inside each iteration of the while-loop is independently uniformly distributed modulo $N$, because $s$ is uniform over $\mathbb{Z}_N$, and the maps $s \mapsto s^e \bmod N$ and $s \mapsto s \cdot y \bmod N$ are permutations (note here that $y \in \mathbb{Z}_N^*$ is invertible). Hence, for uniformly distributed $z \xleftarrow{\$} \mathbb{Z}_N$, $N < 2^n$, we have per iteration

$$\Pr\left[z \geq 2^\ell\right] = \frac{N - 2^\ell}{N} < 1 - 2^{\ell - n}.$$

Using $1 - x \leq e^{-x}$, we fail after $n \cdot 2^{n-\ell}$ iterations with probability at most

$$(1 - 2^{\ell-n})^{n \cdot 2^{n-\ell}} \leq e^{-n}.$$

The uniform distribution of $z$ over $\{0,1\}^\ell$ (condition (2)) follows from the fact that $z$ is a uniformly distributed integer modulo $N$. Hence for $\ell < n$, we can use the well-known rejection sampling method: Rejecting all values of $z \geq 2^\ell$ yields uniformly distributed values in $\{0,1\}^\ell$.

For checking condition (3), we see that $\hat{y} := 2^\ell \cdot \mathtt{PAD} + z$ by definition in **Encode**. From (2) we have $z < 2^\ell$, which implies $\hat{y} = \mathtt{PAD} || z$ over the integers. However, $\hat{y}$ has to be defined in $\mathbb{Z}_{\widehat{N}}$. Since $|\mathtt{PAD}| = n + r - \ell - 2$, we verify that

$$2^\ell \cdot \mathtt{PAD} + z < 2^\ell \cdot 2^{n+r-\ell-2} = 2^{n-1} \cdot 2^{r-1} < NR = \widehat{N}.$$

To show (4), we compute

$$\begin{aligned} \hat{y} &= 2^\ell \cdot \mathtt{PAD} + z \bmod \widehat{N} \\ &= 2^\ell \cdot \mathtt{PAD} + ys^e - 2^\ell \cdot \mathtt{PAD} \bmod N \\ &= ys^e \bmod N. \end{aligned}$$

Finally, we observe that **Encode**'s running time is dominated by the while-loop with a maximum of $n \cdot 2^{n-\ell}$ iterations each taking time $\mathcal{O}(\log e \log^2 N) = \mathcal{O}(n^3)$. □

**Corollary 1.** *Let $n := |N|$. The **Encode** algorithm described in Figure 4 efficiently $(n - \mathcal{O}(\log n))$-simulates the PKCS#1 v1.5 encoding modulo $NR$.*

By Corollary 1, our **Encode** algorithm is efficient only for hash values as large as $\ell \approx |n|$. We leave it as an important open problem to reduce $\ell$ to output sizes of standard hash functions.

### 3.2 Security Proof under the RSA Assumption

Now we are ready to prove security under the RSA assumption. Normally, the size of the RSA modulus is implicitly assumed to be the same in the assumption as in the scheme, therefore it is never explicitly fixed. However, our proof modifies this, by using a larger modulus for the scheme than in the assumption, thus we need to explicitly state these values. For the sake of notation, we formulate our assumptions with explicit bit-sizes. We also consider the more general case where primes *may*, but need not be of the same size. Once all the primes have been selected, we can pick an exponent $e$ such that $\gcd(e, \varphi(N)) = 1$, where we have for $N = \prod_{i=1}^{k} p_i$ that $\varphi(N) = \prod_{i=1}^{k} (p_i - 1)$.

**Assumption 1** ($(k,n)$-RSA Assumption)**.** The RSA Assumption, denoted by $(k,n)$-RSA, states that given $(N, e, x^e)$ it is hard to compute $x$, where $N$ is the product of $k$ distinct random prime numbers $p_i \in \mathbb{P}[n_i]$, for $i \in [\![1,k]\!]$, $k$ constant, and $\sum_{i=1}^{n} n_i = n$, $e \in \mathbb{Z}^*_{\varphi(N)}$, and $x \in_R \mathbb{Z}_N$. RSA is said to be $(t,\varepsilon)$-hard, if for all adversaries $\mathcal{A}$ running in time at most $t$, we have

$$\mathbf{Adv}_{\mathcal{A}}^{(k,n)\text{-RSA}} = \Pr\left[x = \mathcal{A}(N, e, x^e \bmod N)\right] \leqslant \varepsilon.$$

Additionally, we define RSA-PKCS1-v1.5 $[k, n, \ell]$ to be the RSA PKCS#1 v1.5 scheme with a $n$-bit public modulus $N$ which is the product of $k$ primes and an $\ell$-bit hash function. It should be noted that having a modulus that is the product of more than two primes is allowed by the PKCS standard (RFC 8017).

Our first proof follows the proof technique of Coron [Cor00, Cor01] and has a loss in the order of $q_s$. As was shown by Coron [Cor02, Cor01] and Kakvi and Kiltz [KK12, KK18], this loss is optimal for RSA with "large" exponents. This is due to the fact that RSA with a "large" exponent defines a certified trapdoor permutation [BY93, BY96], as was shown by Kakvi, Kiltz and May [KKM12].

Our second proof follows the proof technique of Kakvi and Kiltz [KK12, KK18] and is tight to the $\varphi$-Hiding assumption. This proof bypasses the optimality result, by using keys that do not define a certified trapdoor permutation. By using "small" keys, which gives us a lossy trapdoor permutation, we can provide a tight security proof to the $\varphi$-Hiding assumption. Furthermore, there is currently no known proof technique which achieves tighter security for small exponents (such as $e = 2^{16} + 1$, as commonly used in real-world instantiations) *and* is based on the RSA assumption.

**Theorem 2.** *Assume the* $(2,n)$*-RSA assumption is* $(t', \varepsilon')$*-hard. Then for any* $(q_h, q_s)$*, the* RSA-PKCS1-v1.5 $[3, 2n-1]$ *scheme is* $(t, \varepsilon, q_h, q_s)$-UF-CMA *secure in the Random Oracle Model, where*

$$\begin{aligned} \varepsilon' &\geqslant \frac{\varepsilon}{q_s} \cdot \left(1 - \frac{1}{q_s + 1}\right)^{q_s+1} \approx \frac{\varepsilon}{q_s} \\ t' &\leqslant t + q_h \cdot \mathcal{O}(n^4). \end{aligned}$$

PROOF. We describe the oracles in Figure 5. The reduction first receives a $(2,n)-$RSA challenge $(N, e, y)$. It then builds a $2n$-bit modulus by selecting $R \in_R \mathbb{P}[n], \gcd(e, R-1) = 1$ and computing $\widehat{N} = NR$. The reduction then computes the corresponding padding by first computing $\nu = n - \alpha - 23$, and then computing $\texttt{PAD} = 0^{15}||1^{\nu}||0^8||\texttt{ID}_\mathsf{H}$. The public key $\mathsf{pk} = (\widehat{N}, e, \texttt{PAD})$ is sent to the forger. We now describe our oracles.

For every hash query $m_i$ the reduction flips a biased coin $\chi_i$ with $\Pr[\chi_i = 0] = p_0, \Pr[\chi_i = 1] = p_1$, the choice of which will be discussed later. The idea behind this is that with probability $p_1$ the reduction embeds the RSA challenge into the oracle response and with probability $p_0$, it will not. For the values where the reduction does not embed the challenge, i.e. $\chi_i = 0$, it can simulate signatures and for ones where it does, i.e. $\chi_i = 1$, it can extract a solution. In the case $\chi_i = 0$, the reduction simply runs **Encode**$(N, e, 1, \ell, \texttt{PAD}, R)$ and gets $(\hat{y}_i, s_i, z_i)$. The reduction now computes $\sigma_i$, using the Chinese Remainder Theorem. It is clear from the construction that $\hat{y}^{1/e} = s_i \bmod N$ and furthermore, the reduction can easily compute $\hat{y}^{1/e} \bmod R$, which gives

$$\sigma_i = s_i \cdot R(R^{-1} \bmod N) + (\hat{y}_i^{1/e} \bmod R) \cdot N(N^{-1} \bmod R) \bmod \hat{N}.$$

The reduction then adds $(m_i, z_i, s_i, \sigma_i)$ to its hash list $\mathcal{H}$.

In the case $\chi_i = 1$, the reduction runs **Encode**$(N, e, y, \ell, \texttt{PAD}, R)$ and gets $(\hat{y}_i, s_i, z_i)$. However, since it cannot simulate a signature, it stores the distinguished abort symbol $\perp$ as the signature, i.e., it adds $(m_i, z_i, s_i, \perp)$ to its hash list $\mathcal{H}$.

**Initialize**$(1^n)$
$(N, e, y) \leftarrow_{\$} \mathsf{RSA}(2, 1^n)$
$R \in_R \mathbb{P}[n], gcd(e, R-1) = 1$
$\widehat{N} = N \cdot R \in \mathbb{Z}[2n]$
pick $\mathtt{ID}_{\mathsf{H}} \in \{0,1\}^{\alpha}$
$\nu = n - \alpha - 23$
$\mathtt{PAD} = 0^{15}||1^{\nu}||0^8||\mathtt{ID}_{\mathsf{H}}$
return $\mathsf{pk} = (\widehat{N}, e, \mathtt{PAD})$

**Hash**$(m)$
if $(m, \cdot) \in \mathcal{H}$
    fetch $(m, z_m) \in \mathcal{H}$
    return $z_m$
else
    $\chi_m \leftarrow_{p_0,p_1} \{0,1\}$
    if $(\chi_i = 0)$
        $(\hat{y}_m, s_m, z_m) \leftarrow_{\$}$ **Encode**$(N, e, 1, \ell, \mathtt{PAD}, R)$;
        $\sigma_m = s_m \cdot R(R^{-1} \bmod N) + (\hat{y}_m^{1/e} \bmod R) \cdot N(N^{-1} \bmod R) \bmod \hat{N}$
    else if $(\chi_m = 1)$
        $(\hat{y}, s_m, z_m) \leftarrow_{\$}$ **Encode**$(N, e, y, \ell, \mathtt{PAD}, R)$;
        $\sigma_m = \perp$
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{(m, z_m, s_m, \sigma_m)\}$
    return $z_m$
**Sign**$(m)$
$\mathcal{M} \leftarrow \mathcal{M} \cup (m)$
fetch $(m, z_m, \sigma_m) \in \mathcal{H}$
return $\sigma_m$

**Finalize**$(m^*, \sigma^*)$
if $(\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1) \wedge (m^* \notin \mathcal{M})$
    if $(\chi_{m^*} = 1)$
        $x = \sigma^* \cdot (s_{m^*})^{-1} \bmod N$
    else
        $x = \perp$
    RSA.**Finalize**$(x)$
    return 1
else
    RSA.**Finalize**$(\perp)$
    return 0

Figure 5: Oracles for the RSA based proof.

Having updated $\mathcal{H}$, the reduction now returns $z_i$ as the response to the random oracle query.

We assume that the forger always makes a hash query on a message before it queries that message to the signing oracle. The reduction can also implicitly do this by querying the random oracle itself. This means that for every signature query $m_i$, there is an entry $(m_i, z_i, \sigma_i) \in \mathcal{H}$. If $\sigma_i = \perp$, i.e. the hash value has a challenge embedded in it, the reduction aborts, as it cannot provide a signature. If this is not the case, it simply returns $\sigma_i$ as response and adds $m_i$ to the list of queried messages $\mathcal{M}$. The probability that the reduction answers any signature query correctly is $p_0$ and the probability that it answers all queries correctly is at least $p_0^{q_s}$.

In the final phase, the forger submits a forgery $(m^*, \sigma^*)$. With probability $p_1$, $m^*$ is one of the messages where we embedded our RSA challenge. First we observe that a valid forgery means that $\sigma^* = \hat{y}^{*1/e} \bmod \widehat{N}$. Our encode algorithm ensures that $\hat{y} = s^{*e} \cdot y \bmod N$, which in turn means that $\sigma^* \bmod N = \hat{y} = s^* \cdot y^{1/e} \bmod N$ . Because the reduction has an entry in $\mathcal{H}$ containing $(m^*, z^*, s^*, \perp)$, it can extract $y^{1/e} = \sigma^* \cdot (s^*)^{-1} \bmod N$, which is the solution to its RSA challenge.

The reduction can extract a solution with probability $p_1 = 1 - p_0$, which means that the reduction succeeds with probability at least $p_0^{q_s} \cdot (1 - p_0) \cdot \varepsilon$. To get the optimal result, we need to maximize the expression $p_0^{q_s} \cdot (1 - p_0) \cdot \varepsilon$. Coron [Cor02] showed an optimal value of $p_0 = \frac{1}{q_s+1}$, giving us a success probability of at least $\frac{\varepsilon}{q_s} \cdot \left(1 - \frac{1}{q_s+1}\right)^{q_s+1}$, as required. The time bound is the time needed to answer all hash queries. □

*Remark* 1. In the proof we select $R$ to be a prime number, however this is not strictly necessary. What we require is that we are able to compute $e^{\text{th}}$ roots mod $R$. Thus it would also suffice if we knew the prime factorization of $R$. It should be noted that the difficulty of factoring $\widehat{N} = NR$ is at least that of factoring $N$ if the smallest prime factor of $R$ is at least $n/2$ bits long. $R$ was chosen to be prime in the proof for convenience, but the proof goes through equally for any suitable composite $R$.

*Remark* 2. While the PKCS#1 standard does not specify how long the padding must be, apart from the limitation that PS must be at least 8 octets long, one must be careful when using smaller paddings. If the padding is less than half of the bits, the message represenatives are suceptible to the attacks of de Jonge and Chaum [dC86] and Girault and Misarsky [GM97]. Although the usage of a hash function complicates the attacks, it is always better to err on the side of caution.

### 3.3 Security Proof under the Phi-Hiding Assumption

We are also able to produce an essentially optimal security proof for "small" exponents $e$, using the techniques of Kakvi and Kiltz [KK12, KK18]. In our case, "small" means our exponent is at most $\frac{1}{4}$ of the bit-length of our modulus. This proof proceeds as a series of games, each of which we will describe separately. We first recall the $\varphi$-Hiding Assumption.

*Assumption* 2 (The $\varphi$-Hiding Assumption. [CMS99]). The $\varphi$-Hiding Assumption, denoted by ΦHA, states that it is hard to distinguish between $(N, e)$ and $(N, \hat{e})$, where $N$ is the product of two random $(n/2)$-bit primes and $e, \hat{e} > 3 \in \mathbb{P}$ and and $e, \hat{e} \leq N^{\frac{1}{4}}$, with $gcd(e, \varphi(N)) = 1$ and $gcd(\hat{e}, \varphi(N)) = \hat{e}$, where $\varphi$ is the Euler Totient function. ΦHA is said to be $(t, \varepsilon)$-hard, if for all distinguishers $\mathcal{D}$ running in time at most $t$, we have:

$$\mathbf{Adv}_{\mathcal{D}}^{\Phi\mathsf{H}} = \Pr\left[1 \leftarrow \mathcal{D}(N, e)\right] - \Pr\left[1 \leftarrow \mathcal{D}(N, \hat{e})\right] \leqslant \varepsilon$$

Game $\mathsf{G}_0$

**Initialize**$(1^n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathtt{RSA\text{-}PKCS1\text{-}v1.5}[3, 2n, n]$
return $\mathsf{pk}$

**Hash**$(m)$
if $(m, \cdot) \in \mathcal{H}$
  fetch $(m, z_m) \in \mathcal{H}$
  return $z_m$
else
  $z_m \in_R \{0,1\}^\ell$;
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(m, z_m)\}$
  return $z_m$
**Sign**$(m)$
$\mathcal{M} \leftarrow \mathcal{M} \cup (m)$
fetch $(m, z_m) \in \mathcal{H}$
return $\sigma_m = (\mathtt{PAD}||z_m)^{1/e}$

**Finalize**$(m^*, \sigma^*)$
if $(\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1) \wedge (m^* \notin \mathcal{M})$
  return 1
else
  return 0

(a) Game 0: The Standard UF-CMA Game

Game $\mathsf{G}_1$

**Initialize**$(1^n)$
$(N, e) \leftarrow_{\$} \mathsf{RSAPermGen}(1^n)$ //$\mathsf{G}_1, \mathsf{G}_3$
$(N, e) \leftarrow_{\$} \mathsf{RSALossyGen}(1^n)$ //$\mathsf{G}_2, \mathsf{G}_4$
$R \in_R \mathbb{P}[n], gcd(e, R-1) = 1$
$\widehat{N} = N \cdot R \in \mathbb{Z}[2n]$
pick $\mathtt{ID}_\mathsf{H} \in \{0,1\}^\alpha$
$\nu = n - \alpha - 23$
$\mathtt{PAD} = 0^{15}||1^\nu||0^8||\mathtt{ID}_\mathsf{H}$
return $\mathsf{pk} = (\widehat{N}, e, \mathtt{PAD})$

**Hash**$(m)$
if $(m, \cdot) \in \mathcal{H}$
  fetch $(m, z_m) \in \mathcal{H}$
  return $z_m$
else
  $(\hat{y}, \sigma_m, z_m) \leftarrow_{\$}$ **Encode**$(N, e, 1, \mathtt{PAD}, R)$;
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(m, z_m, \sigma_m)\}$
  return $z_m$
**Sign**$(m)$
$\mathcal{M} \leftarrow \mathcal{M} \cup (m)$
fetch $(m, z_m, \sigma_m) \in \mathcal{H}$
return $\sigma_m$

**Finalize**$(m^*, \sigma^*)$
if $(\sigma^* = \sigma_m)$ //$\mathsf{G}_3, \mathsf{G}_4$
  return 0 //$\mathsf{G}_3, \mathsf{G}_4$
if $(\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1) \wedge (m^* \notin \mathcal{M})$
  return 1
else
  return 0

(b) Games 1-4: The Simulated Reduction Games

The requirement that $e, \hat{e} \leq N^{\frac{1}{4}}$ stems from the fact that simply given a modulus $N$, we are unable to ascertain how many prime factors it consists of and how large they are in relation to one another. We know that $N$ is composite, therefore consists of at least 2 prime factors[7]. The bound then follows directly from the well known attacks due to Coppersmith [Cop97]. We also note that when $\gcd(\hat{e}, \varphi(N)) = \hat{e}$, the function $x^{\hat{e}} \bmod N$ is exactly $\hat{e}$-to-1, i.e. it said to be *$\hat{e}$-regular lossy* as defined by Kakvi and Kiltz [KK12, KK18].

*Theorem* 3. *Assume that* $\Phi\mathsf{HA}$ *is* $(t', \varepsilon')$*-hard and defines a* $\gamma$*-regular lossy function. Then for any* $(q_h, q_s)$*,* $\mathtt{RSA\text{-}PKCS1\text{-}v1.5}$ $[3, 2n]$ *is* $(t, \varepsilon, q_h, q_s)$-UF-CMA *secure in the Random Oracle Model, where*

$$\begin{aligned} \varepsilon &\leqslant \frac{2\gamma - 1}{\gamma - 1} \cdot \varepsilon' \\ t' &\leqslant t + q_h \cdot \mathcal{O}(n^4). \end{aligned}$$

PROOF.

[7] We assume $N$ is not a prime power, which can be checked efficiently [Ber98].

The proof proceeds in a sequence of games, which are given in Figures 6a, 6b. The idea is to move from the UF-CMA game to one the forger *cannot* win. The original UF-CMA game is given in $\mathsf{G}_0$. The first game hop we make is from normal signing to simulating signatures, using our **Encode** algorithm, which does not change anything from the view of the forger. The next step is to move from the "normal" public key where $\gcd(e, \varphi(N)) = 1$, to the lossy key $(N, \hat{e})$ were now we have a 1-to-$\hat{e}$ signing function. The forger does not notice this, unless it breaks the ΦHA. Now, a new abort condition is introduced; the reduction aborts if the forgery is the same signature it picked in the simulation phase. Since the signature is picked from the set of $\hat{e}$ signatures independently from the view of the forger, this happens with probability at most $\frac{1}{\hat{e}}$. Finally the keys are switched back to "normal" keys, which now means any valid forgery leads to an abort from the reduction. If we combine the bounds from all the games, we get exactly the bounds stated in the proof. We prove the Theorem step by step, with Lemmata 1, 2, 3, 4 & 5.

*Lemma* 1. $\Pr[\mathsf{G}_0^{\mathcal{F}} \Rightarrow 1] = \Pr[\mathsf{G}_1^{\mathcal{F}} \Rightarrow 1]$.

PROOF. In $\mathsf{G}_0$, we modelled the hash function as a random oracle. In $\mathsf{G}_1$ we modify the random oracle and the signing queries. On any $m$ the random oracle now "precomputes" a signature for $m$ using the **Encode** algorithm. Note that signing no longer requires the secret key. It can be seen that all our signatures will verify due to the fact that $\sigma_m^e = \mathtt{PAD}||z_m$ for all $m$. Thus our simulation of the signatures is correct. Since **Encode** gives us uniformly distributed values $s, z$, the distribution of our hash queries in $\mathsf{G}_1$ is identical to the distribution in $\mathsf{G}_0$. Thus we have $\Pr[\mathsf{G}_0^{\mathcal{F}} \Rightarrow 1] = \Pr[\mathsf{G}_1^{\mathcal{F}} \Rightarrow 1]$. □

*Lemma* 2. *There exists a distinguisher $\mathcal{D}_1$ against the* ΦHA*, which runs in time $t = t + q_h \cdot \mathcal{O}(n^4)$ and such that* $\left|\Pr[\mathsf{G}_1^{\mathcal{F}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{F}} \Rightarrow 1]\right| = \mathbf{Adv}_{\mathcal{D}_1}^{\Phi\mathsf{HA}}$.

PROOF. From $\mathsf{G}_1$ to $\mathsf{G}_2$, we change the key generation from a normal key $(\widehat{N}, e)$ to a lossy $(\widehat{N}, \hat{e})$, however the oracles are identical in both games. We now build a distinguisher $\mathcal{D}_1$ against the ΦHA, using these games. The distinguisher will run $\mathcal{F}$ and simulates the oracles $\mathsf{Sign}(\cdot), \mathsf{Hash}(\cdot)$ as described in games $\mathsf{G}_1$ and $\mathsf{G}_2$. Note that $\mathcal{D}_1$ does not require the signing key to simulate the oracles. After $\mathcal{F}$ calls **Finalize**, $\mathcal{D}_1$ returns the output of **Finalize**. Thus we can see that $\Pr[1 \leftarrow \mathcal{D}_1(N, e)] = \Pr\left[\mathsf{G}_1^{\mathcal{F}} \Rightarrow 1\right]$ and $\Pr[1 \leftarrow \mathcal{D}_1(N, \hat{e})] = \Pr\left[\mathsf{G}_2^{\mathcal{F}} \Rightarrow 1\right]$. Hence we have

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{F}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathcal{F}} \Rightarrow 1\right]\right| = \left|\Pr[1 \leftarrow \mathcal{D}_1(N, e)] - \Pr[1 \leftarrow \mathcal{D}_1(N, \hat{e})]\right| = \mathbf{Adv}_{\mathcal{D}_1}^{\Phi\mathsf{HA}}.$$

□

*Lemma* 3. $\Pr\left[\mathsf{G}_3^{\mathcal{F}} \Rightarrow 1\right] = \left(\frac{\hat{e}-1}{\hat{e}}\right) \Pr\left[\mathsf{G}_2^{\mathcal{F}} \Rightarrow 1\right]$.

PROOF. In $\mathsf{G}_3$, we introduce a new rule, which aborts the reduction if the forgery $\sigma^*$ provided by $\mathcal{F}$ is the same as the simulated signature $\sigma_{m^*}$ for the target message $m^*$. If this is the case, the forger loses the game, i.e., $\mathsf{G}_3$ outputs 0. $\sigma_{m^*}$ is independent of $\mathcal{F}$'s view and is uniformly distributed in the set of pre-images of $\hat{y}_{m^*}$. Due to the fact that the function $x^{\hat{e}} \bmod \widehat{N}$ is $\hat{e}$-regular lossy, the probability of a collision is $1/\hat{e}$. Thus we see that this abort condition reduces the probability of the forger winning the game by $1/\hat{e}$, hence

$$\Pr\left[\mathsf{G}_3^{\mathcal{F}} \Rightarrow 1\right] = \left(1 - \frac{1}{\hat{e}}\right) \Pr\left[\mathsf{G}_2^{\mathcal{F}} \Rightarrow 1\right] = \left(\frac{\hat{e}-1}{\hat{e}}\right) \Pr\left[\mathsf{G}_2^{\mathcal{F}} \Rightarrow 1\right].$$

□

*Lemma* 4. *There exists a distinguisher $\mathcal{D}_2$ against the* $\Phi$HA*, which runs in time $t = t + q_h \cdot \mathcal{O}(n^4)$ and such that $|\Pr[\mathsf{G}_3^{\mathcal{F}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{F}} \Rightarrow 1]| = \mathbf{Adv}_{\mathcal{D}_2}^{\Phi\mathsf{HA}}$.*

PROOF. From $\mathsf{G}_3$ to $\mathsf{G}_4$, we change the key generation from a lossy key $(\widehat{N}, \hat{e})$ to a normal key $(\widehat{N}, e)$, however the oracles are identical in both games. We now build a distinguisher $\mathcal{D}_2$ against $\Phi$HA, using these games. The distinguisher runs $\mathcal{F}$ and simulates the oracles $\mathsf{Sign}(\cdot), \mathsf{Hash}(\cdot)$ as described in games $\mathsf{G}_3$&$\mathsf{G}_4$. Note that $\mathcal{D}_2$ does not require the signing key to simulate the oracles. After $\mathcal{F}$ calls **Finalize**, $\mathcal{D}_2$ returns the output of **Finalize**. Thus we can see that $\Pr[1 \leftarrow \mathcal{D}_2(N, e)] = \Pr\left[\mathsf{G}_4^{\mathcal{F}} \Rightarrow 1\right]$ and
$\Pr\left[1 \leftarrow \mathcal{D}_2(N, \hat{e})\right] = \Pr\left[\mathsf{G}_3^{\mathcal{F}} \Rightarrow 1\right]$ Hence we have

$$\left|\Pr\left[\mathsf{G}_4^{\mathcal{F}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_3^{\mathcal{F}} \Rightarrow 1\right]\right| = |\Pr[1 \leftarrow \mathcal{D}_2(N, e)] - \Pr[1 \leftarrow \mathcal{D}_2(N, \hat{e})]| = \mathbf{Adv}_{\mathcal{D}_2}^{\Phi\mathsf{HA}}.$$

□

*Lemma* 5. $\Pr[\mathsf{G}_4 \Rightarrow 1] = 0$.

PROOF. In $\mathsf{G}_4$ we are using the normal keys, i.e. $x^e \bmod \widehat{N}$ is a 1-to-1 function. Since our signing function is now injective, any forgery implies a collision. Therefore whenever the forger is able to produce a valid forgery, the game outputs 0 due to it failing the check, as $\forall m^*, \sigma_m = \sigma^*$. Whenever the forger is unable to make a forgery, the game outputs 0. Thus we can see that in all cases, the game will output 0, hence $\Pr[\mathsf{G}_4 \Rightarrow 1] = 0$. □

We combine Lemmata 1 to 5 to get

$$\Pr[1 \leftarrow \mathsf{G}_0] = \mathbf{Adv}_{\mathcal{D}_1}^{\Phi\mathsf{HA}} + \left(\frac{\hat{e}}{\hat{e} - 1}\right) \mathbf{Adv}_{\mathcal{D}_2}^{\Phi\mathsf{HA}}.$$

Because $\mathcal{D}_1$ and $\mathcal{D}_2$ run in time at most $t + q_h \cdot \mathcal{O}(n^3)$, by our assumption, both distinguishers have an advantage of at most $\varepsilon'$, giving us:

$$\varepsilon \leq \frac{2\hat{e} - 1}{\hat{e} - 1} \cdot \varepsilon'.$$

As we have an $\hat{e}$-regular lossy permutation, we see that for $\gamma = \hat{e}$, we get exactly the bounds we need. □

### 3.4 The Case of Two Prime Factors

We can easily lift our results for three prime factors presented in the previous section to the standard case of two prime factors, by additionally assuming that breaking the UF-CMA-security of RSA PKCS#1 v1.5 signatures with 2 prime factors is not significantly easier than with 3 prime factors. To make this more precise, we define the 2 vs. 3 Primes Assumption, denoted by 2v3PA, which states that

$$\mathbf{Adv}_{\mathcal{F},\texttt{RSA-PKCS1-v1.5}[2,n,\ell]}^{\mathsf{UF\text{-}CMA}} \leq \mathbf{Adv}_{\mathcal{F},\texttt{RSA-PKCS1-v1.5}[3,n,\ell]}^{\mathsf{UF\text{-}CMA}} + \varepsilon$$

for small $\varepsilon$ and all efficient UF-CMA-forgers $\mathcal{F}$.

*Assumption* 3 (Two vs. Three Primes Assumption). We say that 2v3PA is $(t, \varepsilon)$-hard, if for all $\mathcal{F}$ running in time at most $t$, we have:

$$\mathbf{Adv}_{\mathcal{F},\texttt{RSA-PKCS1-v1.5}[2,n,\ell]}^{\mathsf{UF\text{-}CMA}} \leq \mathbf{Adv}_{\mathcal{F},\texttt{RSA-PKCS1-v1.5}[3,n,\ell]}^{\mathsf{UF\text{-}CMA}} + \varepsilon$$

Note that breaking RSA PKCS#1 v1.5 signatures should only become *easier* if the modulus $\hat{N}$ is a product of more prime factors. For instance, let $\hat{N}_2 = PQ$ be a product of two primes of approximately equal size, and $\hat{N}_3 = PQR$ be a product of three primes of approximately equal size, such that $\hat{N}_2$ and $\hat{N}_3$ have the same size. With known factoring algorithms it should be easier to compute the factorisation of $\hat{N}_3$ than that of $\hat{N}_2$. Hence, even though 2v3PA is a very specific and non-standard assumption, we consider it as very plausible for sufficiently large moduli. With this additional assumption we get the following result.

*Theorem* 4. *Assume the* $(2, n)$-*RSA assumption is* $(t', \varepsilon')$*-hard, and* 2v3PA *is* $(t'', \varepsilon'')$*-hard. Then for any* $(q_h, q_s)$*, the* RSA-PKCS1-v1.5 $[2, 2n, n]$ *scheme is* $(t, \varepsilon, q_h, q_s)$-UF-CMA *secure in the Random Oracle Model, where*

$$\begin{aligned} \varepsilon' + \varepsilon'' &\geqslant \frac{\varepsilon}{q_s} \cdot \left(1 - \frac{1}{q_s + 1}\right)^{q_s+1} \approx \frac{\varepsilon}{q_s} \\ t' &\leqslant t + t'' + q_h \cdot \mathcal{O}(n^4). \end{aligned}$$

PROOF SKETCH. The proof works nearly identically to the proof of Theorem 2, except that we use the 2v3PA assumption in order to replace a two-prime modulus $\hat{N}$ with a three-prime modulus. More precisely, we begin with considering the RSA-PKCS1-v1.5 $[2, 2n]$ defined over a two-prime modulus. Then we replace this scheme with RSA-PKCS1-v1.5 $[3, 2n, n]$. Now we can apply the proof from Theorem 2. □

The following theorem can be proven similarly, by extending the proof of Theorem 3 with an additional step that uses the 2v3PA assumption.

*Theorem* 5. *Assume that* ΦHA *is* $(t', \varepsilon')$*-hard and defines a* $\gamma$*-regular lossy function and* 2v3PA *is* $(t'', \varepsilon'')$*-hard. Then, for any* $(q_h, q_s)$*,* RSA-PKCS1-v1.5 $[2, n + \ell]$ *is* $(t, \varepsilon, q_h, q_s)$-UF-CMA *secure in the Random Oracle Model, where*

$$\begin{aligned} \varepsilon &\leq \frac{2\gamma - 1}{\gamma - 1} \cdot \varepsilon' + \varepsilon'' \\ t' &\leq t + t'' + q_h \cdot \mathcal{O}(n^4). \end{aligned}$$

# 4 Proofs in the Standard Model

In order to give a more complete picture of what security properties of RSA PKCS#1 v1.5 signatures can be proved, we now present different approaches to provide security proofs without random oracles.

## 4.1 Proof based on The Phi-Hiding Assumption

We now present our proof based on the lossiness of RSA and the ΦHA. In this proof we rely on the fact that when $e|\varphi(N)$ the $e^{\text{th}}$roots modulo $N$ are close to uniformly distributed. Okamoto and Stern [OS03] show that in any interval of length $L \geq N^{1/2}$ the number of $e^{\text{th}}$ powers is close to its expectation. However, we need the same statement for intervals of size $2^\ell$, which is usually smaller than $N^{\frac{1}{2}}$. So [OS03] gives us an indication that our assumption in Theorem 6 is plausible, but we cannot direct apply [OS03].

The main idea of the proof of Theorem 6 is that for lossy RSA keys with $e|\varphi(N)$ with high probability there does not even exist a valid RSA PKCS#1 v1.5 signature.

*Theorem* 6. *Assume* $\Phi\mathsf{HA}$ *is* $(t', \varepsilon')$*-hard. Further assume that* $e$*-th powers* $S = \{\sigma^e \mid \sigma \in \mathbb{Z}_N^*\}$ *are uniformly distributed in* $\mathbb{Z}_N^*$ *when* $e|\varphi(N)$*. Then* `RSA-PKCS1-v1.5` *with an* $\ell$*-bit hash function* $\mathsf{H}$ *is* $(t, \varepsilon)$*-UF-NMA secure in the Standard Model, where*

$$\begin{aligned} \varepsilon &\leqslant \left(1 - \frac{2^\ell}{e}\right)\varepsilon', \\ t &\approx t'. \end{aligned}$$

PROOF. We prove this theorem via game hops, which we show in Figures 7a,7b & 7c. $\bar{\mathsf{G}}_0$ is the UF-NMA game. In $\bar{\mathsf{G}}_1$ we switch to "simulated" real keys. There is no change, since we consider no message attacks (NMA) and therefore never need a secret key. Secondly, in $\bar{\mathsf{G}}_2$ we switch to "simulated" lossy keys. The difference here is $\varepsilon'$, unless the forger breaks the $\Phi\mathsf{HA}$. Finally in this game, we show that the adversary cannot win, except with a very small probability.

Game $\bar{\mathsf{G}}_0$

**Initialize**$(1^n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathtt{RSA\text{-}PKCS1\text{-}v1.5}[2, n]$
return $\mathsf{pk}$

**Finalize**$(m^*, \sigma^*)$
if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$
    return 1
else
    return 0

(a) Standard UF-NMA Game

Game $\bar{\mathsf{G}}_1$

**Initialize**$(1^n)$
$(N, e, \hat{e}) \leftarrow_\$ \Phi\mathsf{HA}$
Look-up $\alpha$-bit $\mathtt{ID}_\mathsf{H}$ for $\mathsf{H}$
$\nu = n - \ell - \alpha - 23$
$\mathtt{PAD} = 0^{15}||1^\nu||0^8||\mathtt{ID}_\mathsf{H}$
return $(\mathsf{pk} = (N, e, \mathtt{PAD}, \mathsf{H})$

**Finalize**$(m^*, \sigma^*)$
if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$
    return 1
else
    return 0

(b) UF-NMA Game with simulated public key

Game $\bar{\mathsf{G}}_2$

**Initialize**$(1^n)$
$(N, e, \hat{e}) \leftarrow_\$ \Phi\mathsf{HA}$
Look-up $\alpha$-bit $\mathtt{ID}_\mathsf{H}$ for $\mathsf{H}$
$\nu = n - \ell - \alpha - 23$
$\mathtt{PAD} = 0^{15}||1^\nu||0^8||\mathtt{ID}_\mathsf{H}$
return $(\mathsf{pk} = (N, \hat{e}, \mathtt{PAD}, \mathsf{H})$

**Finalize**$(m^*, \sigma^*)$
if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$
    return 1
else
    return 0

(c) UF-NMA Game with lossy simulated public key

*Lemma* 6. $\Pr[\bar{\mathsf{G}}_0^{\mathcal{F}} \Rightarrow 1] = \Pr[\bar{\mathsf{G}}_1^{\mathcal{F}} \Rightarrow 1]$.

PROOF. From $\bar{\mathsf{G}}_0$ to $\bar{\mathsf{G}}_1$, we changed from the `RSA-PKCS1-v1.5` key generation to the $\Phi\mathsf{HA}$ instance generation. It is clear to see that the distributions are the same, thus there is no change from the view of the forger. □

*Lemma* 7. *There exists a distinguisher* $\mathcal{D}$ *against the* $\Phi\mathsf{HA}$*, which runs in time* $t' \approx t$ *and such that* $|\Pr[\bar{\mathsf{G}}_1^{\mathcal{F}} \Rightarrow 1] - \Pr[\bar{\mathsf{G}}_2^{\mathcal{F}} \Rightarrow 1]| = \mathbf{Adv}_{\mathcal{D}}^{\Phi\mathsf{HA}}$.

PROOF. From $\bar{\mathsf{G}}_1$ to $\bar{\mathsf{G}}_2$, we change the key generation from a normal key $(N, e)$ to a lossy $(N, \hat{e})$. We now build a distinguisher $\mathcal{D}$ against the $\Phi\mathsf{HA}$, using these games. The distinguisher will run $\mathcal{F}$ and after $\mathcal{F}$ calls **Finalize**, $\mathcal{D}$ returns the output of **Finalize**. Thus we can see that $\Pr[1 \leftarrow \mathcal{D}_1(N, e)] = \Pr\left[\bar{\mathsf{G}}_1^{\mathcal{F}} \Rightarrow 1\right]$ and $\Pr[1 \leftarrow \mathcal{D}_1(N, \hat{e})] = \Pr\left[\bar{\mathsf{G}}_2^{\mathcal{F}} \Rightarrow 1\right]$ Hence we have $|\Pr\left[\bar{\mathsf{G}}_1^{\mathcal{F}} \Rightarrow 1\right] - \Pr\left[\bar{\mathsf{G}}_2^{\mathcal{F}} \Rightarrow 1\right]| = |\Pr[1 \leftarrow \mathcal{D}_1(N, e)] - \Pr[1 \leftarrow \mathcal{D}_1(N, \hat{e})]| = \mathbf{Adv}_{\mathcal{D}}^{\Phi\mathsf{HA}}$. □

*Lemma* 8. $\Pr[\bar{\mathsf{G}}_2^{\mathcal{F}} \Rightarrow 1] \leqslant \left(\frac{2^\ell}{e}\right) \Pr[\bar{\mathsf{G}}_1^{\mathcal{F}} \Rightarrow 1]]$.

PROOF. We show that with overwhelming probability there are no valid signatures in the desired PKCS interval, i.e. $\nexists \sigma \in \mathbb{Z}_N^*$ s.t. $\sigma^e \in [\![\mathtt{PAD}||0^\ell, \mathtt{PAD}||1^\ell]\!]$. For lossy keys $e|\varphi(N)$, the mapping

$$\mathbb{Z}_N^* \to \mathbb{Z}_N^*,\ x \mapsto x^e \bmod N$$

is $e$-to-1 for all $x \in \mathbb{Z}_N^*$. Hence $m := |S| = \frac{\varphi(N)}{e}$. Let $S = \{s_1, \ldots, s_m\}$ and define a $0,1$-random variable $X_i$ that takes value 1 iff $s_i \in [\![\mathtt{PAD}||0^\ell, \mathtt{PAD}||1^\ell]\!]$ for $i = 1, \ldots, m$. By assumption, we have

$$\Pr[X_i = 1] = \frac{2^\ell}{\varphi(N)} \text{ for all } i.$$

Let $X := X_1 + \ldots + X_m$ denote the number of signatures in the desired interval $[\![\mathtt{PAD}||0^\ell, \mathtt{PAD}||1^\ell]\!]$. Then we have expectation $\mathbb{E}[X] = \frac{\varphi(N)}{e} \cdot \frac{2^\ell}{\varphi(N)} = \frac{2^\ell}{e}$. Since $X$ is a non-negative integer-valued random variable, it follows that

$$\Pr[X \geq 1] \leq \mathbb{E}[X] = \frac{2^\ell}{e}.$$

Thus, there is a valid signature in the desired interval with probability at most $\frac{2^\ell}{e}$. □

If we combine Lemmata 6 to 8, we get

$$\varepsilon \leqslant \left(1 - \frac{2^\ell}{e}\right) \varepsilon'$$

as required. This completes the proof. □

### 4.2 Proof based on a Variant of the Approximate e-th Root Assumption

The Approximate $e$-th Root (AER) assumption was introduced in 1985 by Okomoto and Shibaishi [OS85] to prove the security of the ESIGN signature scheme (see also [OS03]). In the AER problem, one requires the adversary not to find the root of a specific target value, but one also accept a root of any value "close" to the target. By close we mean that the adversary can submit a root for any value that it is most $\Delta$ away from our target value, where $\Delta$ is part of the challenge. The AER assumption has been thoroughly studied in cryptanalysis [BD86, BO88, GTV90].

*Assumption* 4 (Approximate $e$-th Root (AER).). The Approximate $e$-th Root assumption, denoted by $\mathsf{AER}$, states that given $(N, e, y, \Delta)$, where $N$ is the product of two distinct random $n/2$-bit prime numbers $P$ and $Q$, $e \in \mathbb{Z}_{\varphi(N)}^*$, $y \in_R \mathbb{Z}_N^*$ and $\Delta \leq N^{\frac{e-1}{e}}$, it is hard to compute $x$ such that $|y - x^e| \leqslant \Delta$. $\mathsf{AER}$ is said to be $(t, \varepsilon)$-hard, if for all adversaries $\mathcal{A}$ running in time at most $t$, we have:

$$\mathbf{Adv}^{\mathsf{AER}} = \Pr\left[x = \mathcal{A}(N, e, y, \Delta) \wedge |y - x^e \bmod N| \leqslant \Delta\right] \leqslant \varepsilon$$

The first thing that we can see is that we select $\Delta = 0$, we get exactly the $\mathsf{RSA}$ Assumption. This is because if $\Delta = 0$, we require that the adversary finds $x$ such that $|y - x^e \bmod N| \leqslant 0$, which means $y - x^e = 0 \bmod N$, i.e. $y = x^e \bmod N$. Beyond this, we must note that the the Approximate $e$-th root problem becomes "easier" as the value of $\Delta$ grows. In fact we can see that for $\Delta \geq (N-1)/2$ $\mathsf{AER}$ is trivially solvable, as all $x \in \mathbb{Z}_N$ are valid, that is to say $\forall x, y \in \mathbb{Z}_N, |y - x^e| \leqslant \Delta$.

The problem is also easily solvable if there exists an $e$-th root over the integers nearby $y$. Let us look at the following algorithm. We define as our AER solution $\tilde{x} = \lfloor y^{1/e} \rfloor \in \mathbb{N}$, the rounded $e$-th root of $y$ over the integers. How close is then $\tilde{x}^e$ to $y$? Let us define $x = y^{1/e} \in \mathbb{R}$ with $x \geq \tilde{x}$ and $x - \tilde{x} < 1$. By the Mean Value Theorem for every continuous function $g(x)$ on the interval $[x_0, x_1]$ there exists some $x' \in [x_0, x_1]$ such that

$$g'(x') = \frac{g(x_1) - g(x_0)}{x_1 - x_0},$$

where $g'(x)$ is the first derivative of $g(x)$. For $g(x) = x^e$ this implies

$$0 \leq y - \tilde{x}^e = x^e - \tilde{x}^e \leq g'(x) \cdot (x - \tilde{x}) < ex^{e-1} < eN^{\frac{e-1}{e}}.$$

Hence, we can efficiently solve AER for $\Delta \geq eN^{\frac{e-1}{e}}$. To the best of our knowledge, this is the best known bound for solving AER. Thankfully, the value of $\Delta$ we need for PKCS#1 signatures is much smaller than this, which means that we are not susceptible to this attack.

Let us look more closely as to how we apply the AER assumption to the securoty of PKCS#1 v1.5 signatures. The first thing we need to note is that if we fix our modulus length $n$ and our hash function H, the prefix PAD is the same regardless of the choice of $(N, e)$. Let us call the set of message representatives $\mathbf{Y}_{n,\mathsf{H}}$. We can compute the boundaries of $\mathbf{Y}_{n,\mathsf{H}}$ by simply setting the hash to all zeros and all ones, respectively. This gives us a lower bound of $\mathtt{PAD}||0^\ell$ and an upper bound of $\mathtt{PAD}||1^\ell$, thus $\mathbf{Y}_{n,\mathsf{H}} = [\![\mathtt{PAD}||0^\ell, \mathtt{PAD}||1^\ell]\!]$.

To take this into account, we consider a specific case of the AER Assumption, which we will called *Fixed-Range* Approximate $e$-th Root Assumption. Essentially, we fix our values $y, \Delta$ so that we have $\mathtt{PAD}||0^\ell = y - \Delta$ and $\mathtt{PAD}||1^\ell = y + \Delta$. For notional simplicity, we write $\mathbf{Y}_{n,\mathsf{H}}$ in place of $(y, \Delta)$ and assume it to be publically known.

*Assumption* 5 (Fixed-Range Approximate $e$-th Root Assumption.). The Fixed-Range Approximate $e$-th Root Assumption, denoted by $\mathbf{Y}_{n,\mathsf{H}}$-FAER, states that given $(N, e)$, where $N$ is the product of two distinct random $n/2$-bit prime numbers $P$ and $Q$, $e \in \mathbb{Z}^*_{\varphi(N)}$, it is hard to compute $x$ such that $x^e \bmod N \in \mathbf{Y}_{n,\mathsf{H}}$. $\mathbf{Y}_{n,\mathsf{H}}$-FAER is said to be $(t, \varepsilon)$-hard, if for all adversaries $\mathcal{A}$ running in time at most $t$, we have:

$$\mathbf{Adv}^{\mathbf{Y}_{n,\mathsf{H}}\text{-FAER}}_{\mathcal{A}} = \Pr\left[x = \mathcal{A}(N, e) \wedge x^e \bmod N \in \mathbf{Y}_{n,\mathsf{H}}\right] \leqslant \varepsilon$$

*Theorem* 7. *Assume the* $\mathbf{Y}_{n,\mathsf{H}}$-*FAER assumption is* $(t', \varepsilon')$*-hard, then* RSA-PKCS1-v1.5 *is* $(t, \varepsilon)$-UF-NMA *secure in the Standard Model, where:*

$$\begin{aligned} \varepsilon &= \varepsilon' \\ t &\approx t' \end{aligned}$$

PROOF. Given $(N, e)$ from the challenger and knowing $\mathbf{Y}_{n,\mathsf{H}}$, we can compute our public key $\mathsf{pk} = (N, e, \mathtt{PAD}, \mathsf{H})$. We simply take $(N, e)$ from the challenge and knowing $\mathbf{Y}_{n,\mathsf{H}} = [\![\mathtt{PAD}||0^\ell, \mathtt{PAD}||1^\ell]\!]$ we can derive the value PAD and the hash function, as the identifier $\mathtt{ID}_\mathsf{H}$ is contained in PAD. We now send this public key to the forger. If the forger produces a valid forgery $\sigma^*$ for message $m^*$, we know that $\sigma^{*e} \bmod N = \mathtt{PAD}||\mathsf{H}(m^*) \in \mathbf{Y}_{n,\mathsf{H}}$. Thus we can submit $\sigma^*$ as our approximate root. We see that any valid forgery by the adversary is a valid solution to our $\mathbf{Y}_{n,\mathsf{H}}$-FAER challenge. Thus, we are successful whenever the forger is successful, which gives us $\varepsilon = \varepsilon'$, as required. □

# 5 Conclusions

This paper presents the first security proofs for PKCS#1 v1.5 signatures under plausible cryptographic hardness assumptions. In particular, we prove full UF-CMA-security in the random oracle model for a three prime modulus variant of PKCS#1 v1.5 signatures, based on the standard RSA assumption, and give a tight security proof under the $\varphi$-Hiding assumption. This matches the known security proofs of other practical RSA-based signature schemes, such as RSA-PSS and RSA-FDH. Using an additional assumption, we are able to lift our results from our variant to the more standard two prime modulus instantiation of PKCS#1 v1.5 signatures. Hence, PKCS#1 v1.5 signatures can be used securely in practice, when instantiated such that the bounds on the length of the hash function output and the modulus from our proofs apply.

An interesting research direction for future work is to improve the provided bounds on the size of the padding and the hash function, ideally such that standard hash functions, such as SHA-512, are covered. Furthermore, an interesting question is whether one can directly prove the security of two prime modulus PKCS#1 v1.5 signatures, without any additional assumptions.

# Acknowledgements

We would like to thank the CCS 2018 anonymous reviewer for their insightful comments. We would also like to thank Jan Bobolz for pointing us to the SHAKE XOFs.

# References

[AMV15] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15: 22nd Conference on Computer and Communications Security*, pages 364–375, Denver, CO, USA, October 12–16, 2015. ACM Press.

[BD86] Ernest F. Brickell and John M. DeLaurentis. An attack on a signature scheme proposed by Okamoto and Shiraishi. In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, pages 28–32, Santa Barbara, CA, USA, August 18–22, 1986. Springer, Heidelberg, Germany.

[BDPA11] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The Keccak SHA-3 submission. Submission to NIST (Round 3), 2011. Available at `https://keccak.team/files/Keccak-reference-3.0.pdf`

[Ber98] Daniel J. Bernstein. Detecting perfect powers in essentially linear time. *Math. Comput.*, 67(223):1253–1283, July 1998.

[BFK+12] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. Efficient padding oracle attacks on cryptographic hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 608–625, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[BHJ+13] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, Jae Hong Seo, and Christoph Striecks. Practical signatures from standard assumptions. In Thomas Johansson and Phong Q.

Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 461–485, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 273–304, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

[BO88] Ernest F Brickell and Andrew M Odlyzko. Cryptanalysis: A survey of recent results. *Proceedings of the IEEE*, 76(5):578–593, 1988.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Perugia, Italy, May 9–12, 1995. Springer, Heidelberg, Germany.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany.

[BSY17] Hanno Böck, Juraj Somorovsky, and Craig Young. Return of bleichenbacher's oracle threat (ROBOT). Cryptology ePrint Archive, Report 2017/1189, 2017. `https://eprint.iacr.org/2017/1189`.

[BY93] Mihir Bellare and Moti Yung. Certifying cryptographic tools: The case of trapdoor permutations. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 442–460, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany.

[BY96] Mihir Bellare and Moti Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, 1996.

[CFPR96] Don Coppersmith, Matthew K. Franklin, Jacques Patarin, and Michael K. Reiter. Low-exponent RSA with related messages. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 1–9, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany.

[CJNP00] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. New attacks on PKCS#1 v1.5 encryption. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 369–381, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

[Cop97] Don Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *Journal of Cryptology*, 10(1):233–260, November 1997.

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany.

[Cor01] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. Cryptology ePrint Archive, Report 2001/062, 2001. `http://eprint.iacr.org/2001/062`.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.

[CS99] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In *ACM CCS 99: 6th Conference on Computer and Communications Security*, pages 46–51, Kent Ridge Digital Labs, Singapore, November 1–4, 1999. ACM Press.

[dC86] Wiebren de Jonge and David Chaum. Attacks on some RSA signatures. In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, pages 18–27, Santa Barbara, CA, USA, August 18–22, 1986. Springer, Heidelberg, Germany.

[DLP+12] Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, Nigel P. Smart, and Mario Strefler. On the joint security of encryption and signature in EMV. In Orr Dunkelman, editor, *Topics in Cryptology – CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 116–135, San Francisco, CA, USA, February 27 – March 2, 2012. Springer, Heidelberg, Germany.

[FGK+13] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. *Journal of Cryptology*, 26(1):39–74, January 2013.

[Fis03] Marc Fischlin. The Cramer-Shoup strong-RSA signature scheme revisited. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 116–129, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany.

[FIPS 202] Information Technology Laboratory and National Institute of Standards and Technology. SHA-3 standard: Permutation-based hash and extendable-output functions. Federal Information Proccessing Standards Publication 202, August 2015.

[GM97] Marc Girault and Jean-François Misarsky. Selective forgery of RSA signatures using redundancy. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 495–507, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

[GTV90] Marc Girault, Philippe Toffin, and Brigitte Vallée. Computation of approximate l-th roots modulo n and application to cryptography. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 100–117, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Heidelberg, Germany.

[HJK11] Dennis Hofheinz, Tibor Jager, and Eike Kiltz. Short signatures from weaker assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 647–666, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

[HW09] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 333–350, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany.

[JSS15a] Tibor Jager, Jörg Schwenk, and Juraj Somorovsky. On the security of TLS 1.3 and QUIC against weaknesses in PKCS#1 v1.5 encryption. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15: 22nd Conference on Computer and Communications Security*, pages 1185–1196, Denver, CO, USA, October 12–16, 2015. ACM Press.

[JSS15b] Tibor Jager, Jörg Schwenk, and Juraj Somorovsky. Practical invalid curve attacks on TLS-ECDH. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015: 20th European Symposium on Research in Computer Security, Part I*, volume 9326 of *Lecture Notes in Computer Science*, pages 407–425, Vienna, Austria, September 21–25, 2015. Springer, Heidelberg, Germany.

[KK12] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 537–553, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[KK18] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. *Journal of Cryptology*, 31(1):276–306, January 2018.

[KKM12] Saqib A. Kakvi, Eike Kiltz, and Alexander May. Certifying RSA. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 404–414, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[KOS10] Eike Kiltz, Adam O'Neill, and Adam Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 295–313, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[KP09] Eike Kiltz and Krzysztof Pietrzak. On the security of padding-based encryption schemes - or - why we cannot prove OAEP secure in the standard model. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 389–406, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany.

[KPR03] Vlastimil Klíma, Ondrej Pokorný, and Tomás Rosa. Attacking RSA-based sessions in SSL/TLS. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 426–440, Cologne, Germany, September 8–10, 2003. Springer, Heidelberg, Germany.

[KPS13] Eike Kiltz, Krzysztof Pietrzak, and Mario Szegedy. Digital signatures with minimal overhead from indifferentiable random invertible functions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 571–588, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[LOS13] Mark Lewko, Adam O'Neill, and Adam Smith. Regularity of lossy RSA on subdomains and its applications. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 55–75, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[MSW+14] Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. Revisiting SSL/TLS implementations: New bleichenbacher side channels and attacks. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 733–748. USENIX Association, 2014.

[OS85] Tatsuski Okamoto and Akira Shibaishi. A fast signature scheme based on quadratic inequalities. In *Security and Privacy, 1985 IEEE Symposium on*, pages 123–123. IEEE, 1985.

[OS03] Tatsuaki Okamoto and Jacques Stern. Almost uniform density of power residues and the provable security of ESIGN. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 287–301, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Heidelberg, Germany.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[RSA78] Ronald L. Rivest, A. Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978.

[Sch11] Sven Schäge. Tight proofs for signature schemes without random oracles. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 189–206, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[Seu14] Yannick Seurin. On the lossiness of the Rabin trapdoor function. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 380–398, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[Sho02] Victor Shoup. OAEP reconsidered. *Journal of Cryptology*, 15(4):223–249, 2002.

[SPW07] Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In Masayuki Abe, editor, *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 357–371, San Francisco, CA, USA, February 5–9, 2007. Springer, Heidelberg, Germany.

[SZ15] Adam Smith and Ye Zhang. On the regularity of lossy RSA - improved bounds and applications to padding-based encryption. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 609–628, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[ZJRR14] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Cross-tenant side-channel attacks in PaaS clouds. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 990–1003, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press.

## Primary RFC References

[2246] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Obsoleted by RFCs [4346], updated by RFC [3546, 5746, 6176, 7465, 7507, 7919]. `https://www.rfc-editor.org/rfc/rfc2246.txt`

[2313] B. Kaliski. *PKCS #1: RSA Encryption Version 1.5*. RFC Editor, Fremont, CA, USA, March 1998. Obsoleted by RFC [2437]. `https://www.rfc-editor.org/rfc/rfc2313.txt`

[2437] B. Kaliski and J. Staddon. PKCS #1: RSA Cryptography Specifications Version 2.0. RFC 2437 (Informational), October 1998. Obsoleted by RFC [3447]. `https://www.rfc-editor.org/rfc/rfc2437.txt`

[3370] R. Housley. Cryptographic Message Syntax (CMS) Algorithms. RFC 3370 (Proposed Standard), August 2002. Updated by RFC [5754]. `https://www.rfc-editor.org/rfc/rfc3370.txt`

[3447] J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational), February 2003. Obsoleted by RFC [8017]. https://www.rfc-editor.org/rfc/rfc3447.txt

[4055] J. Schaad, B. Kaliski, 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 (Proposed Standard), June 2005. Updated by RFC [5756]. https://www.rfc-editor.org/rfc/rfc4055.txt

[4346] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Obsoleted by RFC [5246], updated by RFCs [4366, 4680, 4681, 5746, 6176, 7465, 7507, 7919]. https://www.rfc-editor.org/rfc/rfc4346.txt

[4359] B. Weis. The Use of RSA/SHA-1 Signatures within Encapsulating Security Payload (ESP) and Authentication Header (AH). RFC 4359 (Proposed Standard), January 2006. https://www.rfc-editor.org/rfc/rfc4359.txt

[4880] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP Message Format. RFC 4880 (Proposed Standard), November 2007. Updated by RFC [5581]. https://www.rfc-editor.org/rfc/rfc4880.txt

[5246] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFC [5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905, 7919]. https://www.rfc-editor.org/rfc/rfc5246.txt

[7515] M. Jones, J. Bradley, and N. Sakimura. JSON Web Signature (JWS). RFC 7515 (Proposed Standard), May 2015. https://www.rfc-editor.org/rfc/rfc7515.txt

[8017] K. Moriarty (Ed.), B. Kaliski, J. Jonsson, and A. Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017 (Informational), November 2016. https://www.rfc-editor.org/rfc/rfc8017.txt

## Secondary RFC References

[3546] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 3546 (Proposed Standard), June 2003. Obsoleted by RFC [4366]. https://www.rfc-editor.org/rfc/rfc3456.txt

[4366] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 4366 (Proposed Standard), April 2006. Obsoleted by RFCs [5246, 6066], updated by RFC [5746]. https://www.rfc-editor.org/rfc/rfc4366.txt

[4680] S. Santesson. TLS Handshake Message for Supplemental Data. RFC 4680 (Proposed Standard), October 2006. https://www.rfc-editor.org/rfc/rfc4680.txt

[4681] S. Santesson, A. Medvinsky, and J. Ball. TLS User Mapping Extension. RFC 4681 (Proposed Standard), October 2006. https://www.rfc-editor.org/rfc/rfc4681.txt

[5581] D. Shaw. The Camellia Cipher in OpenPGP. RFC 5581 (Informational), June 2009. https://www.rfc-editor.org/rfc/rfc4880.txt

[5746] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov. Transport Layer Security (TLS) Renegotiation Indication Extension. RFC 5746 (Proposed Standard), February 2010. https://www.rfc-editor.org/rfc/rfc5746.txt

[5754] S. Turner. Using SHA2 Algorithms with Cryptographic Message Syntax. RFC 5754 (Proposed Standard), January 2010. https://www.rfc-editor.org/rfc/rfc5754.txt

[5756] S. Turner, D. Brown, K. Yiu, R. Housley, and T. Polk. Updates for RSAES-OAEP and RSASSA-PSS Algorithm Parameters. RFC 5756 (Proposed Standard), January 2010. https://www.rfc-editor.org/rfc/rfc5756.txt

[5878] M. Brown and R. Housley. Transport Layer Security (TLS) Authorization Extensions. RFC 5878 (Experimental), May 2010. https://www.rfc-editor.org/rfc/rfc5878.txt

[6066] D. Eastlake 3rd. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066 (Proposed Standard), January 2011. https://www.rfc-editor.org/rfc/rfc6066.txt

[6176] S. Turner and T. Polk. Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176 (Proposed Standard), March 2011. https://www.rfc-editor.org/rfc/rfc6176.txt

[7465] A. Popov. Prohibiting RC4 Cipher Suites. RFC 7465 (Proposed Standard), February 2015. https://www.rfc-editor.org/rfc/rfc7465.txt

[7507] B. Moeller and A. Langley. TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks. RFC 7507 (Proposed Standard), April 2015. https://www.rfc-editor.org/rfc/rfc7505.txt

[7568] R. Barnes, M. Thomson, A. Pironti, and A. Langley. Deprecating Secure Sockets Layer Version 3.0. RFC 7568 (Proposed Standard), June 2015. https://www.rfc-editor.org/rfc/rfc7568.txt

[7627] K. Bhargavan (Ed.), A. Delignat-Lavaud, A. Pironti, A. Langley, and M. Ray. Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension. RFC 7627 (Proposed Standard), September 2015. https://www.rfc-editor.org/rfc/rfc7627.txt

[7685] A. Langley. A Transport Layer Security (TLS) ClientHello Padding Extension. RFC 7685 (Proposed Standard), October 2015. https://www.rfc-editor.org/rfc/rfc7685.txt

[7905] A. Langley, W. Chang, N. Mavrogiannopoulos, J. Strombergson, and S. Josefsson. ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS). RFC 7905 (Proposed Standard), June 2016. https://www.rfc-editor.org/rfc/rfc7905.txt

[7919] D. Gillmor. Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS). RFC 7919 (Proposed Standard), August 2016. https://www.rfc-editor.org/rfc/rfc7919.txt