NIST Special Publication 800-89

Recommendation for Obtaining Assurances for Digital Signature Applications

Elaine Barker

Computer Security Division
Information Technology Laboratory

C O M P U T E R S E C U R I T Y

November 2006

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

Technology Administration
Robert Cresanti, Under Secretary of Commerce for Technology

National Institute of Standards and Technology
William Jeffrey, Director

NIST National Institute of Standards and Technology • Technology Administration • U.S. Department of Commerce

Abstract

Entities participating in the generation or verification of digital signatures depend on the authenticity of the process. This Recommendation specifies methods for obtaining the assurances necessary for valid digital signatures: assurance of domain parameter validity, assurance of public key validity, assurance that the key pair owner actually possesses the private key, and assurance of the identity of the key pair owner.

KEY WORDS: assurance, Certification Authority, digital signatures, timestamp token, Trusted Timestamp Authority.

Acknowledgements

The National Institute of Standards and Technology (NIST) gratefully acknowledges and appreciates contributions by Rich Davis from the National Security Agency concerning the many security issues associated with this Recommendation. NIST also thanks the many contributions by the public and private sectors whose thoughtful and constructive comments improved the quality and usefulness of this publication.

Table of Contents

Recommendation for Obtaining Assurances for Digital Signature Applications

1 Introduction

A digital signature is an electronic analogue of a written signature; the digital signature can be used to provide assurance that the claimed signatory signed the information. In addition, a digital signature may be used to detect whether or not the information was modified after it was signed (i.e., to detect the integrity of the signed data). Each signatory has a public and private key and is the owner of that key pair. The private key is used by the owner to generate a digital signature; the public key is used in the signature verification process.

Entities participating in the generation or verification of digital signatures depend on the authenticity of the process. This Recommendation specifies methods for obtaining the assurances necessary for valid digital signatures: assurance of domain parameter validity, assurance of public key validity, assurance that the key pair owner actually possesses the private key, and assurance of the identity of the key pair owner.

Federal Information Processing Standard (FIPS) 186-3 allows three techniques for the generation of digital signatures: the Digital Signature Algorithm (DSA), RSA[1], and the Elliptic Curve Digital Signature Algorithm (ECDSA). For DSA and ECDSA, assurance of the validity of the domain parameters must be obtained (see Section 4). RSA has no domain parameters. Domain parameters may be generated by anyone, and assurance of domain parameter validity **shall** be obtained prior to performing any other process associated with digital signatures, including the generation of digital signature key pairs and the generation and verification of digital signatures.

Digital signature keys may be generated by the intended signatory, cooperatively generated by the intended signatory and a Trusted Third Party (TTP), or generated entirely by a TTP and provided to the intended signatory. For all digital signature algorithms, each party associated with the digital signature process **shall** have assurance of the validity of the public keys (see Section 5) and assurance that the key pair owner actually possesses the private key used for generating the digital signature (see Section 6). In addition, assurance of the claimed signatory's identity is required by all verifiers, including any TTPs involved in the process (see Section 7).

All methods that are used to provide assurance assume the security and reliability of any routines involved in the process. Obtaining assurances normally requires explicit actions by someone. However, once the appropriate assurances are obtained, the explicitly obtained assurance can be leveraged as assurance for subsequent messages. Note that it may be appropriate to renew this assurance periodically.

[1] An algorithm developed by Rivest, Shamir and Adelman.

2 Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines **shall not** apply to national security systems. This recommendation is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This Recommendation has been prepared for use by Federal agencies. It may be used by non-governmental organizations on a voluntary basis and is not subject to copyright (attribution would be appreciated by NIST.)

Nothing in this Recommendation should be taken to contradict standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should this Recommendation be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.

Conformance testing for implementations of this Recommendation will be conducted within the framework of the Cryptographic Module Validation Program (CMVP), a joint effort of NIST and the Communications Security Establishment of the Government of Canada.

3 Definitions and Acronyms

3.1 Definitions

The following definitions are provided.

Approved	FIPS-Approved and/or NIST-recommended. An algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) adopted in a FIPS or NIST Recommendation and specified in an appendix to the FIPS or NIST Recommendation.
assurance_level	The level of assurance (e.g., HIGH, MEDIUM, or LOW) that a claimed signatory possesses the private signature key.
Assurance message	See private-key-possession assurance message.

Assurance of domain parameter validity	Assurance of the arithmetic validity of the domain parameters.
Assurance of possession	Assurance that the owner or claimed signatory actually possesses the private signature key.
Assurance of public key validity	Assurance of the arithmetic validity of the public key.
Assurance-signature	A digital signature on a private-key-possession assurance message.
assurance_time	The time at which assurance of possession is obtained.
Certificate	A set of data that uniquely identifies a key pair owner that is authorized to use the key pair, contains the owner's public key and possibly other information, and is digitally signed by a Certification Authority (i.e., a trusted party), thereby binding the public key to the owner.
Certification Authority (CA)	The entity in a Public Key Infrastructure (PKI) that is responsible for issuing certificates and exacting compliance with a PKI policy.
Claimed signatory	From the verifier's perspective, the claimed signatory is the entity that purportedly generated a digital signature.
Digital signature	The result of a cryptographic transformation of data that, when properly implemented, provides origin authentication, data integrity and signatory non-repudiation.
Domain parameters	Parameters used with a cryptographic algorithm that are usually common to a domain of users.
Entity	An individual (person), organization, device or process. Used interchangeably with "party".
Intended owner	An entity that intends to act as a signatory but has not yet obtained a private key that will be used to generate digital signatures.
Intended signatory	An entity that intends to generate digital signatures in the future.

Key	A parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce the operation, while an entity without knowledge of the key cannot. Examples of the use of a key that are applicable to this Recommendation include: 1. The computation of a digital signature from data, and 2. The verification of a digital signature.
Key pair	A public key and its corresponding private key.
Message	The data that is signed. Also known as "signed data" during the signature verification and validation process.
Non-assurance message	A signed message that does not contain all information required for an assurance message.
Owner	A key pair owner is the entity that is authorized to use the private key of a key pair.
Party	An individual (person), organization, device or process. Used interchangeably with "entity".
Private key	A cryptographic key that is used with an asymmetric (public key) cryptographic algorithm. For digital signatures, the private key is uniquely associated with the owner and is not made public. The private key is used to compute a digital signature that may be verified by the corresponding public key.
Private-key-possession assurance message	A message that is sent by a signatory that will be used to obtain assurance of possession. Also known as an assurance message.
Public key	A cryptographic key that is used with an asymmetric (public key) cryptographic algorithm and is associated with a private key. The public key is associated with an owner and may be made public. In the case of digital signatures, the public key is used to verify a digital signature that was signed by the corresponding private key.
Relying party	A party that depends on the validity of the digital signature process.
Shall	Used to indicate a requirement of this Recommendation.

Should	Used to indicate a strong recommendation, but not a requirement of this Recommendation.
Signatory	The entity that generates a digital signature on data using a private key.
Signature-in-question	The digital signature to be verified and validated.
Signature generation	The process of using a digital signature algorithm and a private key to generate a digital signature on data.
Signature validation	The (mathematical) verification of the digital signature plus obtaining the appropriate assurances (e.g., public key validity, private key possession, etc.).
Signature verification	The process of using a digital signature algorithm and a public key to verify a digital signature on data.
Signed data	The data or message upon which a digital signature has been computed. Also, see Message.
Timestamp	A token of information that is used to provide assurance of timeliness; contains timestamped data, including time, and a signature generated by a Trusted Timestamp Authority (TTA).
timestamp_time	The time provided in a timestamp.
Trusted Third Party (TTP)	An entity other than the owner and verifier that is trusted by the owner, the verifier or both to provide certain services.
Trusted timestamp	A timestamp that has been signed by a Trusted Timestamp Authority.
Trusted Timestamp Authority (TTA)	An entity that is trusted to provide accurate time information.
Verifier	The entity that verifies the authenticity of a digital signature using the public key.

3.2 Acronyms

ANS	American National Standard.

CA	Certification Authority.
DSA	Digital Signature Algorithm (specified in FIPS 186-3).
ECDSA	Elliptic Curve Digital Signature Algorithm; allowed in FIPS 186-3 and specified in ANS X9.62.
FIPS	Federal Information Processing Standard.
FISMA	Federal Information Security Management Act.
NIST	National Institute of Standards and Technology.
OMB	Office of Management and Budget.
PKCS	Public Key Cryptography Standard.
PKI	Public Key Infrastructure.
RSA	Algorithm developed by Rivest, Shamir and Adelman (allowed in FIPS 186-3 and specified in ANS X9.31 and PKCS #1).
SHA-1	A hash function specified in FIPS 180-2, the Secure Hash Standard.
TST	Timestamp Token.
TTA	Trusted Timestamp Authority.
TTP	Trusted Third Party.

4 Assurance of Domain Parameter Validity

DSA and ECDSA depend on the arithmetic validity of the domain parameters. All parties associated with the digital signature process **shall** have assurance of their validity prior to using them. Intended signatories **shall** obtain this assurance prior to generating their keys or using keys provided for their use by a TTP. A TTP that generates keys for an intended signatory **shall** obtain assurance of domain parameter validity prior to generating keys. A Certification Authority (CA) that issues certificates for intended signatories **shall** obtain this assurance prior to issuing digital signature certificates. Verifiers **shall** obtain this assurance prior to the verification of a digital signature. Users of the domain parameters must have assurance that the domain parameters are valid for the lifetime of any keys that are associated with the domain parameters.

Each entity **shall** obtain assurance that the domain parameters are valid using at least one of the following methods:

1. The entity itself generates the domain parameters according to the specified requirements (see FIPS 186-3 for DSA, and American National Standard (ANS) X9.62 for ECDSA).

2. The entity performs an explicit domain parameter validation and obtains an indication of validity (see Section 4.1 for DSA, and ANS X9.62 for ECDSA; for ECDSA, a check that the subgroup order *n* and the cofactor *h* are selected appropriately for the security strength **shall** also be included).
3. The entity has received assurance from a TTP (e.g., a CA) that the domain parameters were valid at the time that they were generated because the TTP either generated the domain parameters (see method 1 above and Section 4.2), or has performed an explicit domain parameter validation (see method 2 above).

   Note: Some domain parameters have been generated using SHA-1 (e.g., the NIST Recommended curves in FIPS 186-3). When SHA-1 is no longer approved for the generation of digital signatures (see [2]), those domain parameters that were generated using SHA-1 and were successfully validated while SHA-1 was still Approved will continue to qualify as valid.

The entity, or an agent trusted to act on the entity's behalf, **shall** know which method(s) of assurance were used in order to determine that the provided assurance is sufficient and appropriate to meet an application's requirements.

4.1 Explicit Domain Parameter Validation for DSA

DSA domain parameters are explicitly validated using the following procedure or its equivalent if *p, q, g, domain_parameter_seed*, and *counter* are known:

Input:

1. *L* The length of *p* in bits.
2. *N* The length of *q* in bits.
3. *p, q, g, domain_parameter_seed, counter*

   The domain parameters, where *p* and *q* are primes, *g* is the generator, *domain_parameter_seed* is the domain parameter seed that was used to generate *p* and *q*, and *counter* was obtained during the generation of *p* and *q*.

Output:

1. *status* The status returned from the validation routine, where *status* is either ***g*_partially_validated, *g*_fully_validated** or **INVALID.**

Process:

Comment: Verify the values of the (*L*, *N*) pair specified in FIPS 186-3.

1. Verify that the (*L*, *N*) pair is valid. If (*L*, *N*) is not in the list of Approved pairs (see FIPS 186-3), then return **INVALID.**

2. Verify that *p* and *q* were generated using one of the prime generation algorithms specified in Appendix A.1 of FIPS 186-3[2], and execute the appropriate validation algorithm. If *p* and *q* were not generated as specified in FIPS 186-3, or the validation algorithm returns **INVALID**, then return **INVALID**. Otherwise, go to step 3.
3. If *g* was generated using the unverifiable generation method specified in Appendix A.2.1 of FIPS 186-3, then go to step 4. Otherwise, go to step 5.
4. Perform a partial validation of *g* using the process specified in Annex A.2.2 of FIPS 186-3. If **PARTIALLY VALID** is returned, then return ***g*_partially_validated.** Otherwise, return **INVALID.**
5. If *g* was generated using the verifiable generation method specified in Appendix A.2.3 of FIPS 186-3, perform the validation process specified in Appendix A.2.4 of FIPS 186-3. If this process returns **INVALID**, then return **INVALID**. Otherwise, return ***g*_fully_validated**.

4.2 Assurances When Domain Parameters are Generated by Another Entity

There are two basic scenarios for DSA and ECDSA domain parameter generation: generation by a *trusted party* and generation by a *non-trusted party*.

Domain parameter generation by a trusted party means that this party is trusted by all users in a particular (sub) domain to correctly generate valid sets of domain parameters. This trusted party "guarantees" the validity of all sets of domain parameters that it generates, and generates new sets of parameters as needed. Note that users outside the particular (sub) domain may have little or no trust in the domain parameters generated by this party.

The generation of (candidate) domain parameters by an untrusted party means that the untrusting users require some evidence that the domain parameters are valid. These users may be concerned that the (candidate) domain parameters are generated incorrectly, or that the party generating the domain parameters may be an adversary that is trying to gain an advantage by generating the parameters in a non-standard manner. If these situations can arise, the (candidate) domain parameters **shall** be generated in a manner that will allow any concerned user to obtain assurance that the parameters are indeed valid.

There are different methods of obtaining assurance of domain parameter validity that depend on whether or not the user seeking assurance trusts the party that generated the parameters.

4.2.1 Assurance When Domain Parameters are Generated by a Trusted Party

Domain parameters may be generated by a party (i.e., a TTP) that is trusted by an entity to generate them correctly. Assurance of domain parameter validity is provided to the

[2] It is assumed that the method used will be known or all methods will be tried.

entity because of that trust. However, the TTP **shall** obtain this assurance for itself using one of the following methods:

1. The TTP generates the set of domain parameters.
2. The TTP selects the set of domain parameters from a list published by an entity that the TTP trusts.
3. The TTP performs explicit domain parameter validation and obtains an indication of validity (see Section 4.1 for DSA and ANS X9.62 for ECDSA).

4.2.2 Assurance When Domain Parameters are Generated by an Untrusted Party

Domain parameters may be generated by a party that is not trusted by the entity requiring assurance of domain parameter validity. In this case, the entity **shall** obtain assurance of domain parameter validity using at least one of the following methods:

1. The entity performs explicit domain parameter validation and obtains an indication of validity (see Section 4.1 for DSA and ANS X9.62 for ECDSA).
2. The entity receives assurance from a third party (i.e., a TTP) that is trusted by that entity that the TTP performed an explicit domain parameter validation and obtained an indication of validity (see Section 4.1 for DSA and ANS X9.62 for ECDSA).

5 Assurance of Public Key Validity

The correct functioning of the digital signature algorithms depends on the arithmetic validity of the public/private key pair that is used for digital signature generation and verification. Digital signature key pairs **shall** be generated by the intended signatory, generated in a cooperative process by both the signatory and a TTP, or generated by a TTP (trusted by the intended signatory) and provided to the intended signatory.

Each intended signatory (i.e., the key pair owner) **shall** have assurance of public key validity prior to generating a digital signature (see Section 5.1). Each verifier **shall** have assurance of the validity of the public key prior to verifying a digital signature (see Section 5.2). CAs **shall** have assurance of public key validity prior to issuing a digital signature certificate containing the digital signature public key for the owner.

For DSA and ECDSA, an entity **shall** obtain assurance of domain parameter validity prior to obtaining assurance of public key validity.

5.1 Owner Assurances of Public Key Validity

The owner **shall** obtain assurance of the validity of its own public key using one or more of the following methods. Maximal assurance is provided when method 1 or 2 is combined with method 3 or 4.

1. Intended owner generation. The intended owner generates the key pair.

2. Cooperative generation by the owner and a TTP. The intended owner generates the key pair with the assistance of a TTP using an Approved method.
3. Intended owner validation. The intended owner performs an explicit public key validation (see Section 5.3) and receives an indication of validity.
4. TTP validation. The intended owner receives assurance that a TTP (trusted by the intended owner) has performed a successful public key validation on the intended owner's public key for which assurance is to be obtained (see Section 5.3) and received an indication of validity. The TTP **shall** provide the public key validation result to the intended owner.
5. TTP generation. The intended owner receives assurance when a TTP (trusted by the intended owner) generates the key pair and provides it to the intended owner. This method is not preferred, since the TTP will know the private key and must be trusted not to masquerade as the owner. However, if used, this method **should** include public key validation by the owner or TTP (method 3 or method 4).

The owner, or an agent trusted to act on the owner's behalf, **shall** know which method(s) of assurance were used in order to determine that the provided assurance is sufficient and appropriate to meet the owner's requirements.

5.2 Verifier Assurances of Public Key Validity

A verifier **shall** obtain assurance of the validity of a claimed signatory's public key using one or more of the following methods; the first two methods are preferred.

1. Verifier validation: The verifier performs an explicit public key validation as described in Section 5.3 and receives an indication of validity.
2. TTP assurance of validity: The verifier receives assurance from a TTP (i.e., trusted by the verifier) that the TTP has performed an explicit public key validation as described in Section 5.3 and received an indication of validity.
3. TTP assurance of public key generation or regeneration: The verifier receives assurance when a TTP (trusted by the verifier) has generated or regenerated the public key using trusted routines and has checked the consistency of the key pair. This method may impact the non-repudiation property that could be desired for some signatures, as the TTP will know the private key and must be trusted not to masquerade as the owner.

The verifier, or an agent trusted by the verifier, **shall** know which method(s) of assurance were used in order for the verifier to determine that the provided assurance is sufficient and appropriate to meet the verifier's requirements.

5.3 (Explicit) Public Key Validation

Public key validation is the process of checking the arithmetic characteristics of the public key. Public key validation does not require knowledge of the associated private key; therefore, it may be performed by anyone at any time. Section 5.3.1 specifies the method for full public key validation for DSA. Section 5.3.2 specifies the method for full public key validation for ECDSA. At present, there is no method defined for full public

key validation for RSA; however, a method for partial public key validation is specified in Section 5.3.3 that is to be used until an Approved method for full validation is available.

5.3.1 (Explicit) Full Public Key Validation for DSA

The following process or its equivalent **shall** be used to validate a DSA signature verification public key when public key validation is performed.

Input:

1. *p*, *q* The primes.
2. *g* The generator.
3. *y* The public key to be validated with respect to *p*, *q* and *g*.

Output:

1. *status* The status returned from the validation routine, where *status* is either **VALID** or **INVALID.**

Process:

1. Verify that $2 \leq y \leq p\text{-}2$. Comment: Ensure that the key has the unique correct representation and range in the field.
2. Verify that $y^q \equiv 1 \bmod p$. Comment: Ensure that the key has the correct order in the subgroup.
3. If either of the above checks fail, then return **INVALID.** Otherwise, return **VALID**.

5.3.2 (Explicit) Full Public Key Validation for ECDSA

The method for full public key validation is specified in ANS X9.62.

5.3.3 (Explicit) Partial Public Key Validation for RSA

Partial public key validation for RSA consists of conducting plausibility tests. These tests determine whether the public modulus and public exponent are plausible, not necessarily whether they are completely valid, i.e., that they conform to all RSA key generation requirements as specified in FIPS 186-3. Plausibility tests can detect unintentional errors with a reasonable probability. Note that full RSA public key validation is not specified in this Recommendation, as it is an area of research. Therefore, if an application's requirements require assurance of full public key validation, then either DSA or ECDSA **shall** be used.

Plausibility testing **shall** include the following tests, at a minimum:

1. The length of the modulus is one of the specified values in FIPS 186-3.
2. The value of the public exponent is in the valid range, as specified in FIPS 186-3.
3. The modulus and the public exponent are odd numbers.

4. The modulus is composite, but not a power of a prime
5. The modulus has no factors smaller than 752. Testing for additional factors is allowed.

It is possible for an adversary to maliciously construct an invalid candidate RSA public key that passes all the above specified plausibility tests. However, a system error that results in an invalid candidate RSA public key is likely to be detected. The plausibility tests detect some invalid candidate RSA public keys that are easily broken (e.g., where the public exponent is one, or where the modulus is very easily factored).

The following is an example of a method that includes all the specified tests:

a) *Length of the modulus:* Check that the length of the modulus *n* (*nlen*) is a length specified in FIPS 186-3. If the desired security strength is known, then the length **shall** be appropriate for that security strength.

b) *Size of the public exponent e*: Check that the value of the public key *e* lies in the range specified in FIPS 186-3. If the desired security strength is known, then the range **shall** be appropriate for that security strength.

c) *Oddness:* Check that *n* and *e* are odd.

d) *Compositeness:* Check that *n* fails an Approved primality test. Approved primality tests are provided in FIPS 186-3. If *n* is valid, the test will indicate that *n* is a composite number, as the Approved probable prime tests prove compositeness when they fail to indicate that the number is a probable prime.

e) *Not a power of a prime:* Check that *n* is not a power of a prime, i.e., there are no integers $b \geq 2$, $c \geq 2$ such that $n = b^c$. This can be done by obtaining an output of **COMPOSITE AND NOT A POWER OF A PRIME** using the enhanced Miller-Rabin primality test in FIPS 186-3.

f) *No very small factors:* Check that **GCD** (*n*, *r*) = 1 where *r* is (in decimal).

   145188775577763990151158743208307020242261438098488931355057091965
   931517706595657435907891265414916764399268423699130577757433083166
   651158914570105971074227669275788291575622090199821297575654322355
   049043101306108213104080801056529374892690144291505781966373045481
   835947239164288532817130229924555666307371985 5.

   Note that the value of *r* is the product of the 132 smallest odd primes, from 3 to 751. This value is used to determine that the factors of *n* (i.e., *p* and *q*) are not less than 751. More potential small factors greater than 751 may also be tested in a similar fashion, if desired.

6 Assurance of Private Key Possession

The owner of a digital signature key pair is the entity authorized to use the private key of that key pair when acting as a signatory, generating digital signatures that can be verified with the public key. Authorization to use the private key to generate signatures does not

imply that the owner actually knows the (correct) key; therefore, assurance **shall** be obtained that a key pair owner is actually in possession of the correct private key.

A digital signature key pair may be:

1. Generated and maintained in such a way that the private key is known only by its owner,
2. Generated by an owner with the assistance of a TTP in such a way that the private key is known only by its owner,
3. Generated by a TTP and provided to the owner,
4. Generated using method 1, and provided to a TTP, who may act as a key server.
5. Generated using method 2, and provided to a (possibly different) TTP, who may act as a key server.

Methods 4 and 5 are distinct from the case of a TTP (e.g., a CA) that is provided only with the public key.

In the latter three methods, the trusted party knows the private key and must be relied upon not to use that key to generate digital signatures. This trust must be shared by the owner of the key pair, potential signature verifiers, and any other parties relying on the validity of the owner's signatures. The relying parties must, in effect, be willing to ignore the fact that the trusted party knows the private key, so that any entity that can demonstrate knowledge of the private key is assumed to be its owner. Therefore, methods 3, 4 and 5 are less desirable than methods 1 and 2, where, by design, only the owner knows the private key.

Assurance of the owner's possession of the (correct) private key is required by:

- The key pair owner. Assurance **shall** be obtained prior to or concurrently with the generation of a digital signature.
- A verifier. Assurance **shall** be obtained prior to accepting a verified digital signature as valid. The verifier **shall** obtain assurance of the identity of the party claiming to be the signatory prior to or concurrently with obtaining assurance that the claimed signatory possesses the private key.
- A TTP who will be required to provide assurance to others of the owner's possession of the private key. The TTP **shall** obtain assurance of the owner's identity prior to or concurrently with obtaining assurance that the owner possesses the private key. In the case of a certification authority (CA), the CA **shall** have both assurance of the (intended) owner's identity and assurance of the (intended) owner's possession of the private key prior to issuing a digital signature certificate containing the public key corresponding to that private key.
- A TTP that provides a key pair to the owner. In this case, the TTP **shall** obtain assurance of the owner's identity prior to providing the key pair to the owner and **shall** subsequently obtain assurance that the owner actually possesses the correct private key.

Providing/obtaining assurance of possession normally requires explicit actions. However, once the necessary actions have been taken, the assurance obtained may be leveraged as assurance of private key possession by the owner during some reasonable period containing the time at which those actions were performed. It may be appropriate to periodically repeat the process of explicitly providing/obtaining assurance of possession, due to the possible lessening of assurance over time (see Section 6.1).

In the case of DSA, the private key is denoted as *x*, coupled with the (public) domain parameters; for ECDSA, the private key is *d*, coupled with the (public) domain parameters. No explicit assurance of possession is required for the DSA and ECDSA per-message secret number *k* (which may be considered as a key). For RSA, the private key is denoted as the private value *d*, coupled with the (public) modulus, or an equivalent representation.

For DSA and ECDSA, assurance of domain parameter validity **shall** be obtained prior to obtaining assurance of private key possession.

For all digital signature algorithms, assurance of the validity of the public key **shall** be obtained prior to or concurrently with obtaining assurance of the possession of the associated private key.

6.1 The Effect of Time on Assurance of Private Key Possession

Assurance of private key possession is explicitly provided at a specific point in time, herein referred to as the *assurance_time*. The passage of time may adversely affect the level of confidence that a relying party has in the owner's continued possession of that key. For example, the possible occurrence of one or more of the following events would negatively impact assurance of possession:

1. Owner operations. Activities by the owner may corrupt the private key or break the association between private and public keys. For example, the owner may lose a pointer into a database.
2. Faults due to random errors. Hardware failures may lead to a non-repeatable corruption of the stored private key or the links from the associated public key.
3. Implementation errors. Undetected implementation errors may affect the integrity of either the public or private key, or the association between them. For example, a program could unintentionally overwrite memory. Validation by a NIST Approved testing laboratory provides assurance that implementation errors do not exist.
4. Deliberate attack. Adversarial actions can compromise the assurance of key pair integrity, private key possession and/or private key confidentiality. If a sufficiently long interval has passed since assurance of possession was last explicitly obtained, there may have been adequate time and opportunity for an adversary to mount an attack.

As time progresses, there is ordinarily a decrease in the level of assurance because of the increased probability that one or more of the above events has occured.

On the other hand, it may be reasonable to infer that the owner was in possession of the correct private key for some period of time prior to the *assurance_time*; a relying party's organization must decide whether or not such an inference is acceptable.

For the purposes of this discussion, three levels of assurance are defined relative to the *assurance_time*: HIGH, MEDIUM and LOW. An organization may choose to implement additional or fewer assurance levels.

Figure 1 depicts a simple model for changes in the level of assurance of possession over an extended period of time. In this figure, t_A denotes the actual time when assurance of private key possession is explicitly provided to the relying party. (In practice, one may have to accommodate the use of an approximation.) The quantities *a*, *b* and *c* are values selected by the relying party and/or the relying party's organization:

- *a* and *b* are lengths of time before and after the point at which assurance of private key possession was explicitly provided to the relying party. In this model, the relying party has a HIGH level of assurance that the owner of the key pair is in possession of the private key between times $t_A - a$ and $t_A + b$.
- *c* is a length of time after $t_A + b$. Assurance that the owner possesses the private key degrades between times $t_A + b$ and $t_A + b + c$ to a LOW level of assurance.
- The relying party has a LOW level of assurance that the owner possesses the private key after time $t_A + b + c$.

After degrading in level, the process of explicitly obtaining/providing assurance of private key possession **shall** be repeated if a higher level of assurance is required.

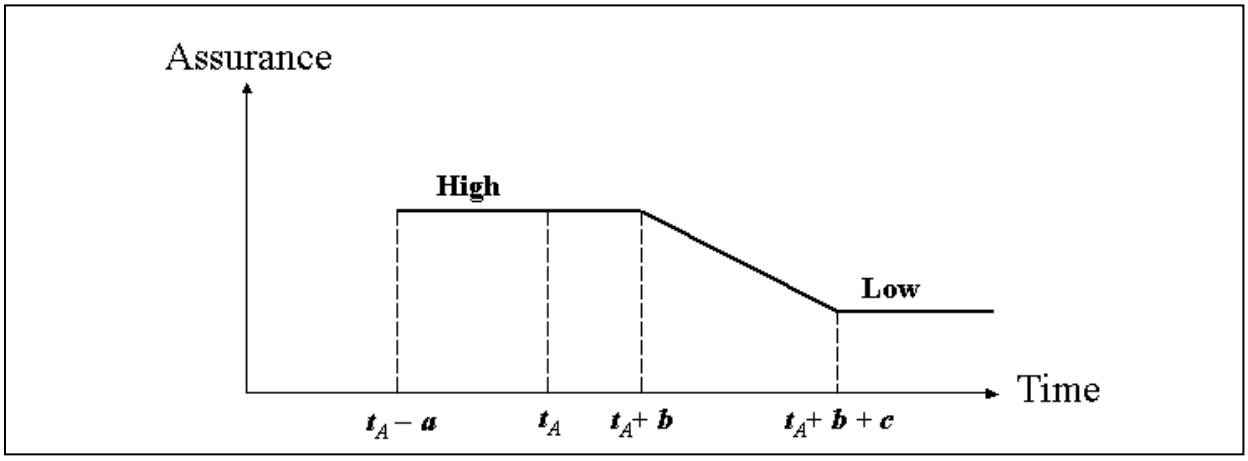

Figure 1: General Assurance of Possession Model

6.2 Assurance of Possession Model with an Estimated *assurance_time*

Ideally, when assurance of private key possession is provided, the *assurance_time* would be a precise time. However, in practice, an estimate may be used. To ensure that the assigned *assurance_time* is a useful estimate, a pair of times must be obtained that will "bracket" the actual time at which assurance was provided, restricting the possibilities to

an interval that is compatible with the general model for assurance of possession described in Section 6.1.

Figure 2 depicts this situation. This figure is based on Figure 1 and will be used to discuss methods for assigning a useful value to *assurance_time*. Note that this figure depicts the initial level of assurance as either HIGH or MEDIUM; this is for illustrative purposes only and is not always achieved in practice (see Sections 6.3.1.3 and 6.3.2.2). In Figure 2:

- *a*, *b* and *c* are as defined for Figure 1.
- t_G represents the precise time that the assurance_time is explicitly provided.
- t_1 is a time that is trusted by the relying parties to precede (or equal) t_G.
- t_2 is a time that is trusted by the relying parties to follow (or equal) t_G.
- *d* is the maximum time lapse allowed between t_1 and t_2.
- t_A is the assigned *assurance_time*. It must satisfy $t_1 \leq t_A \leq t_2$. In this Recommendation, for the purpose of simplicity, t_A will be set equal to either t_1 or t_2 (see Sections 6.3.1.2 and 6.3.2.1).

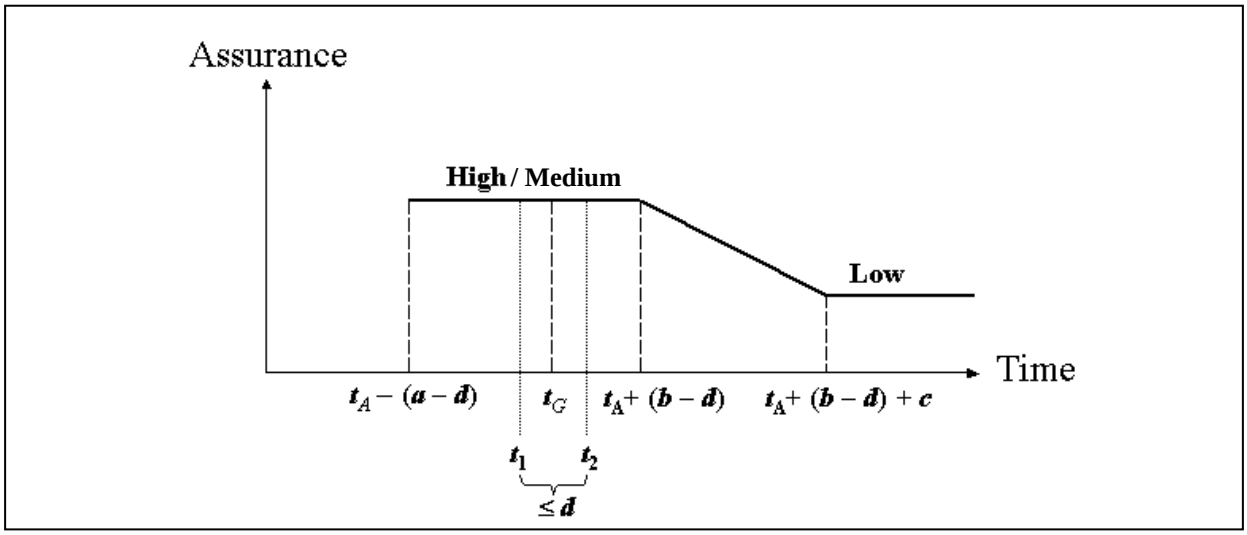

Figure 2: Assurance Model based on an Estimated *assurance_time*

Values for *a, b, c* and *d* are selected by relying parties or their organizations with regard to the following:

1. *a, b* and *c* **should** be selected with regard to the judgments to be made about the validity of the digital signatures processed. Consideration **should** also be given to the ease (or difficulty) of (repeatedly) using a given method to explicitly obtain assurance of possession.
2. A value of *d* **shall** be selected such that *d* is less than one half the minimum of *a* and *b* (i.e., $d < ½$ **min**(*a, b*)). An interval of length *d* must allow sufficient time for the acquisition of time t_1, the actions required to provide/obtain assurance, and the

acquisition of time t_2. In particular, expected transmission times between various parties who must exchange information should be considered when selecting *d*.

Depending on the trustworthiness of the estimated t_A, a relying party may initially have a HIGH, MEDIUM, or LOW level of assurance that the owner of the key pair is in possession of the private key between times $t_A - (a–d)$ and $t_A + (b–d)$ (see Sections 6.3.1.3 and 6.3.2.2) If the assurance level is initially determined to be either HIGH or MEDIUM, assurance that the owner possesses the private key degrades between times $t_A + (b–d)$ and $t_A + (b–d) + c$ to a LOW level. The relying party has only a LOW level of assurance that the owner possesses the private key after time $t_A + (b–d) + c$.

After degrading in level, the process of explicitly providing/obtaining assurance of private key possession **shall** be repeated if a higher level of assurance is required.

6.3 Explicitly Providing/Obtaining Assurance of Private Key Possession

Assurance of an owner's possession of the private key that corresponds to the public key may be explicitly provided/obtained in one or more of the following ways:

1. Generation of a (fresh) digital signature by an entity that purports to be the owner (i.e., the claimed signatory), followed by a verification of the resulting signature using the owner's public key;
2. Regeneration of the entire key pair, followed by a successful comparison of the newly-generated key pair with the key pair currently held by the owner;
3. Regeneration of one key of the key pair from the currently-held value of the other key of that pair, followed by a successful comparison of the newly-generated key value with the value currently held by the owner.

Note that the regeneration methods are only applicable to the key pair owner or an entity that is trusted by the owner with knowledge of the private key. Regeneration might, for example, be used to provide assurance that a key pair generated by a TTP is correct, or might be performed much later than the original generation to obtain assurance that the private key is still correct.

An owner **shall** obtain assurance of possession either through independent action or by interacting with a trusted party (see Section 6.5.1). An entity who will act as a TTP **shall** obtain such assurance by interacting with the owner (see Section 6.5.2). Other relying parties **shall** obtain assurance of the owner's possession of the correct private key either from a TTP or by interacting with the owner (see Section 6.5.3).

The time at which assurance of private key possession was explicitly provided **shall** be recorded (as accurately as possible). This (approximate) *assurance_time* may be used to assess the degree of confidence a relying party may have that the owner (or claimed signatory of a message) was/is in possession of the private key at some other point in time.

The time may be obtained from any number of sources, some more trustworthy than others. For example, the following time sources are listed in decreasing order of trustworthiness: 1) a (signed) timestamp obtained from an authority that is trusted by the

entity obtaining the assurance, the entity's organization, and any other relying parties; 2) a clock whose accuracy is trusted by the entity recording the time; 3) a clock of unknown accuracy. Parties relying on the value assigned to *assurance_time* **shall** be made aware of the time source used, in order to assess its trustworthiness.

When assurance of private key possession is explicitly provided using one of the methods described above, the *assurance_time* **shall** be recorded.

6.3.1 Obtaining Assurance of Possession Using a Digital Signature

Assurance that an entity possesses the private key corresponding to a given public key may be obtained by verifying the signature on an appropriately formed message. The party who obtains assurance using this method may be the same entity that is authorized to use the private key of the key pair (i.e., the intended owner of the key pair), a TTP, or some other entity.

6.3.1.1 The Private-Key-Possession Assurance Message

When a digital signature is used to provide assurance of private key possession, the verifying entity (who must be trusted by all relying parties) is sent a signed *private-key-possession assurance message* (simply called the *assurance message* herein); the digital signature on the assurance message is called the *assurance signature*.

The assurance message **shall** include the following:

1. The signatory's identity;
2. The intended verifier's identity;
3. A timestamp token (TST) created by a Trusted Timestamp Authority (TTA) that is trusted by all relying parties. This TST could be obtained from the TTA by the signatory or by the intended verifier (who then provides it to the signatory). Relying parties must be aware of the security strength provided by the TTA's digital signature. The security strength provided by the TTA's digital signature **shall** meet or exceed the security requirements of the requesting entity and relying party (see SP 800-57);

   AND/OR

   A *nonce*[3] supplied by the intended verifier. If this option is used, the nonce **shall** contain a random component with entropy[4] equal to or greater than the security strength associated with the private key for which assurance of possession is sought. If the assurance message will not include a TST, and the relying parties require that an *assurance_time* be recorded upon successful verification of the assurance signature, then the nonce **should** also contain a verifier-supplied timestamp component that unambiguously represents the time that the nonce was made available for inclusion in the assurance message.

[3] A time-varying value that has at most a negligible chance of repeating.

[4] A measure of the disorder, randomness or variability in a closed system. See SP 800-90.

Unless the "commitment" condition discussed below is met, the assurance message **shall** also include:

4. The public key corresponding to the private key for which assurance of possession is sought**.**

   The public key may be omitted if the (claimed) signatory's "commitment" to the key pair has been unambiguously demonstrated before the exact form of the assurance message could have been known by the signatory.

   Such commitment must precede the time provided in the trusted timestamp token (i.e., the *timestamp_time*) mentioned above, or – in the absence of a TST – must precede the time provided in a timestamp component included in a verifier-supplied nonce. If neither the public key of the claimed signatory nor a time trusted by the relying parties will be included in the assurance message, then the relying parties must obtain trustworthy evidence that the (claimed) signatory committed to the key pair in question before the verifier-supplied nonce was made available for use in the computation of the assurance signature.

   Commitment to a key pair can be demonstrated, for example, by the presence of a trusted timestamp on a certificate (signed by a CA whose signature is trusted by the verifier and other relying parties) that contains the public key of the key pair and identifies the claimed signatory as the owner. Relying parties must be aware of the security strength provided by the CA's signature. The security strength provided by the CA's digital signature **shall** meet or exceed the security requirements of the requesting entity and relying party (see SP 800-57);

   Commitment could also be demonstrated by the existence of a signature that was previously generated by the same claimed signatory that can be successfully verified using the same public key; such a signature **shall** be known to have been generated at a time when the claimed signatory was the owner, and at a time before the exact form of the current assurance message (in particular, the values of included timestamps and/or nonces) could have been known by the signatory.

   A (claimed) signatory's commitment to a key pair could also be demonstrated by making the value of the public key available to the verifier before the verifier supplies a nonce to the signatory (as described in item 3 above).

Additional information may be included in the signed data of the assurance message, such as:

- An explicit indication that the message may be used to provide assurance of private key possession,
- A nonce supplied by the signatory that contains a random component with entropy equal to or greater than the security strength associated with the private key, or
- Any other data to be provided to the intended verifier by the signatory.

Note that any message containing the required information may be used as an assurance message; the message may be intended only to provide assurance of possession or may be

intended for other purposes as well. The message **shall** be signed using the private key for which the assurance is to be provided, thus producing the assurance signature. The signatory **should** generate and provide the assurance signature immediately after the trusted timestamp, nonce, and/or other data included in the assurance message are made available. See Appendix A for an example of an assurance message that is instantiated using a certificate request message.

6.3.1.2 Assigning the *assurance_time* for an Assurance Signature

Following the successful verification of an assurance signature, the assignment of a value to *assurance_time* is affected by the form of the assurance message and the trustworthiness of the sources for the times t_1 and t_2 that bracket the actual time (t_G) at which the assurance signature was generated (see Section 6.2 and Figure 2.)

As defined in Section 6.2, t_1 is a time that either precedes or is equal to the time that the assurance signature was generated. There may be several possibilities for t_1, including one or more of the following:

- If the assurance message includes a timestamp from a TTA (trusted by the relying parties), the *timestamp_time* in that timestamp is a candidate for t_1.
- If the assurance message includes a verifier-supplied time (for example, as a timestamp component of a verifier-supplied nonce), that time is a candidate for t_1.
- If the assurance message includes a verifier-supplied nonce, and the verifier has recorded the time that the nonce was first made available to the claimed signatory, the recorded time is a candidate for t_1.

The candidate time whose source is considered most trustworthy by the relying parties **should** be used as t_1. To choose between equally trustworthy candidates, preference **should** be given to the *latest* time that is included in the assurance message.

As defined in Section 6.2, t_2 is a time that either follows or is equal to the time that the assurance signature was generated. There may be multiple possibilities for t_2, including one or more of the following:

- If the assurance signature is included as part of the *timestamped_data* in a TST obtained from a TTA (trusted by the relying parties), the *timestamp_time* in that timestamp is a candidate for t_2.
- If the verifier has recorded the time that the assurance signature was received, the recorded time of receipt is a candidate for t_2.
- If the verifier has recorded the time that the assurance signature was verified, the recorded verification time is a candidate for t_2.

The candidate whose source is considered most trustworthy by the relying parties **should** be used as t_2. To choose between equally trustworthy candidates, preference **should** be given to the *earliest* time.

Assuming that the assurance signature has been successfully verified, and that an acceptable degree of accuracy (d) has been selected by the relying parties, the estimated *assurance_time* (t_A) **shall** be determined as follows:

a. If $t_2 - t_1 \leq d$ units, and the source of t_1 is at least as trustworthy as the source of t_2, then time t_1 **shall** be used as the *assurance_time*.

b. If $t_2 - t_1 \leq d$ units, and the source of t_1 is less trustworthy than t_2, then time t_2 **shall** be used as the *assurance_time*.

c. If $t_2 - t_1 > d$ units, then assurance of possession has not been provided, and there is no value assigned to *assurance_time*.

If the assurance signature cannot be verified, then assurance of private key possession has not been provided, and there is no value assigned to *assurance_time*.

6.3.1.3 Assigning the Initial Assurance Level for an Assurance Signature

Following the successful verification of an assurance signature and the assignment of the *assurance_time* (see Section 6.3.1.2), the initial level of assurance (LOW, MEDIUM or HIGH) **should** be assigned as follows:

a. If the time used to determine the *assurance_time* was obtained from a timestamp token provided by a TTA trusted by the relying parties, then the initial assurance level is HIGH.

b. If the time used to determine the *assurance_time* was not obtained from a TTA, but was obtained by the verifier from a source whose accuracy is trusted by the relying parties, then the initial assurance level is MEDIUM.

c. If the time used to determine the *assurance_time* was obtained by the verifier from a source of unknown accuracy, then the initial assurance level is LOW.

6.3.2 Obtaining Assurance of Possession via the Regeneration of Keys

Regeneration may be performed by the key pair owner or by a TTP who is provided with the key pair currently held by the owner. When assurance of possession is first explicitly provided/obtained by regeneration, one or both keys of the key pair **shall** be regenerated using different routines and/or a different processor than that used during the original key generation process. On subsequent occasions requiring provision of assurance of possession, the regeneration may be performed using the same routines, and/or the same processor that was used previously. Assurance of private key possession **shall** only be successfully obtained by the relying parties if the regenerated key(s) are the same as the key(s) currently held by the owner.

The regeneration method is different for each digital signature algorithm in FIPS 186-3. For example, for DSA and ECDSA, the public key can be regenerated from the private key. For RSA, the private exponent and the modulus may be regenerated from the public exponent and the prime factors of the modulus (if known).

6.3.2.1 Assigning the *assurance_time* for the Regeneration of Keys

Section 6.2 and Figure 2 may also be used to describe the use of regeneration as a method for obtaining assurance of private key possession, where:

1. t_1 is a time that either precedes or is equal to the time at which the owner's currently held keys – along with any other required data – were made available to the (trusted) party performing the regeneration[5];
2. t_G is the time that the regenerated key(s) were actually compared to those currently held by the owner; and
3. t_2 is a time that either follows or is equal to the time at which the regenerated key(s) were actually compared to those currently held by the owner.

The entity that regenerates the key(s) is encouraged to do so as soon after t_1 as possible, and t_2 **should** be acquired as soon after regeneration and comparison as possible.

Assuming that the regenerated key(s) have been successfully compared to the keys currently held by the owner, and that an acceptable degree of accuracy (d) has been selected by the relying parties, the estimated *assurance_time* (t_A) **shall** be determined as follows:

a. If $t_2 - t_1 \leq d$ units, and the source of t_1 is at least as trustworthy as the source of t_2, then time t_1 **shall** be used as the *assurance_time*.
b. If $t_2 - t_1 \leq d$ units, and the source of t_1 is less trustworthy than t_2, then time t_2 **shall** be used as the *assurance_time*.
c. If $t_2 - t_1 > d$ units, then assurance of possession has not been provided, and there is no value assigned to *assurance_time*.

If the regenerated key(s) do not match those currently held by the owner, then assurance of private key possession has not been provided, and there is no value assigned to *assurance_time*.

6.3.2.2 Assigning the Initial Assurance Level for the Regeneration of Keys

If the regenerated key(s) have been successfully compared to the keys currently held by the owner, and the *assurance_time* (t_A) is assigned as in Section 6.3.2.1, then the initial assurance level **should** be assigned as follows:

a. If the time used to determine the *assurance_time* was obtained from a timestamp token provided by a TTA that is trusted by the relying parties, then the initial assurance level is HIGH.
b. If the time used to determine the *assurance_time* was not obtained from a TTA, but was obtained from a source whose accuracy is trusted by the relying parties, then the initial assurance level is MEDIUM.
c. If the time used to determine the *assurance_time* was obtained from a source of unknown accuracy, then the initial assurance level is LOW.

[5] For example, t_1 is the time that the owner provides the keys to its own regeneration process, or the time that a TTP that will be performing the regeneration is provided with the keys by the owner.

6.4 Assurance of Possession for Signatures of Generic Messages

During normal operations, parties may need to determine a level of assurance of private key possession by the (claimed) signatory at the (apparent) time of generation of the signature of a *generic* message, i.e., a message that does not fulfill the requirements of an assurance message (see Section 6.3.1.1).

In the following, the signature of a particular generic message will be referred to as the *signature-in-question*. The *signature-in-question* may have been generated either before, concurrently with, or after some *assurance_time* (t_A) that was established when assurance of possession by the (claimed) signatory was explicitly provided for the private key corresponding to the public key used for verification of the *signature-in-question*. The *assurance_time* and the initial assurance level associated with that provision of assurance may be used to assess a level of assurance that the (claimed) signatory also possessed the private key at the time that the *signature-in-question* was generated.

It is assumed that t_A and the initial level of assurance have been established as specified in either Section 6.3.1 or Section 6.3.2. Let the time that the *signature-in-question* was generated be denoted as t_s. This t_s may have a known value[6], may only be known to belong to a certain interval of time[7], or it may be completely unknown.

When t_s is known with sufficient accuracy, the level of assurance of private key possession by the (claimed) signatory at the time of generation of the *signature-in-question* may be assigned as follows (in accordance with the assurance of possession model described in Section 6.2, using values for a, b, c and d that are known and have been agreed upon by the relying parties):

- If $(t_A - (a - d)) \leq t_s \leq (t_A + (b - d))$, then the level of assurance associated with the *signature-in-question* is equal to the initial assurance level.
- The level of assurance gradually degrades from the initial level to LOW for $(t_A + (b - d)) \leq t_s \leq (t_A + (b - d) + c)$, and
- If $t_s > (t_A + (b - d) + c)$, then the level of assurance associated with the *signature-in-question* is LOW.

In cases where the generation time for the *signature-in-question* is not known relative to the *assurance_time*, the level of assurance is LOW (at best).

[6] For example, the time t_s could be provided within the generic message as a transmission time, as is commonly done in email messages.

[7] For example, suppose that the generic message M_R corresponding to the *signature-in-question* is a response to a previous message M_P, and that message M_R contains information provided in message M_P that could not have been determined prior to receiving M_P. It follows that t_s occurred between the time M_P was sent and the time M_R was received (along with the *signature-in-question*).

6.5 Obtaining Assurance of Private Key Possession: A Summary

6.5.1 Owner's Assurance of Private Key Possession

A party that intends to act as a signatory (i.e., a key pair owner) **shall** have assurance of possession of the (correct) private key either prior to the generation of a digital signature, or as a result of the successful verification of a digital signature that was expressly generated for the purpose of obtaining assurance of possession of the private key. Key pairs **shall** be generated as specified in Section 5.1 using an Approved method for the digital signature algorithm. For Federal government applications, sufficient assurance of possession is not provided by generation alone.

An owner **shall** obtain an acceptable level of assurance of private key possession using one or more of the following methods.

1. Owner self-assurance using an assurance signature.

   The owner **shall**:

   a. Determine the appropriate value of *d*. For example, *d* may be selected in accordance with the assurance of possession model described in Section 6.2, along with appropriate values of *a*, *b* and *c*.

   b. Determine a trusted value for t_1 (see Sections 6.2 and 6.3.1.2).

   c. Generate a new private-key-possession assurance message (see Section 6.3.1.1).

   d. Sign the assurance message using the private key for which assurance is to be obtained, thus generating an assurance signature.

   e. Verify the assurance signature using the public key that corresponds to the private key.

   f. If verification is successful:

      - Determine a trusted value for t_2 (see Sections 6.2 and 6.3.1.2).
      - Provided that $t_1 \leq t_2 \leq t_1+d$, assign and record the *assurance_time* and the initial assurance level (see Sections 6.3.1.2 and 6.3.1.3).

2. Owner obtains assurance from a TTP using an assurance signature:

   a. The owner **shall** determine the appropriate value of *d*. For example, *d* may be selected in accordance with the assurance of possession model described in Section 6.2, along with appropriate values of *a*, *b* and *c*.

      If the TTP will be responsible for assigning the *assurance_time*, then the value of *d* **shall** be made known to the TTP.

   b. The owner and/or the TTP **shall** determine a value for t_1 that is trusted by the owner (see Sections 6.2 and 6.3.1.2).

   c. The owner **shall** generate a new private-key-possession assurance message

(see Section 6.3.1.1).

d. The owner **shall** sign the assurance message using the private key for which assurance is to be obtained, thus generating an assurance signature.

e. The owner **shall** provide the assurance message, the assurance signature and any other necessary data to the TTP.

f. The TTP **shall** verify the assurance signature using the public key that corresponds to the owner's private key.

g. If verification is successful:

- The owner **shall** be notified of the successful verification.
- The owner and/or the TTP **shall** determine a value for t_2 that is trusted by the owner (see Sections 6.2 and 6.3.1.2).
- Provided that $t_1 \leq t_2 \leq t_1+d$, either the owner or the TTP **shall** assign the *assurance_time* and the initial assurance level (see Sections 6.3.1.2 and 6.3.1.3).

  The party who assigns the *assurance_time* must know the value of d, as well as the values of t_1 and t_2; the party who assigns the initial assurance level must also be aware of the method(s) used to determine t_1 and t_2.
- The owner **shall** record the *assurance_time* and the initial level of assurance.

3. Owner self-assurance via the regeneration of key(s).

   The owner **shall**:

   a. Determine the appropriate value of d. For example, d may be selected in accordance with the assurance of possession model described in Section 6.2, along with appropriate values of a, b and c.

   b. Determine a trusted value for t_1 (see Sections 6.2 and 6.3.2.1).

   c. Perform one of the following:

   - Regenerate the entire key pair corresponding to the private key in question, or
   - Regenerate one key of the key pair corresponding to the private key in question from the currently-held value of the other key of that pair.

   d. Compare the value(s) of the regenerated key(s) with those currently held.

   e. If the regenerated and currently-held keys match:

   - Determine a trusted value for t_2 (see Sections 6.2 and 6.3.2.1).
   - Provided that $t_1 \leq t_2 \leq t_1+d$, assign and record the *assurance_time* and the initial assurance level (see Sections 6.3.2.1 and 6.3.2.2).

4. Owner obtains assurance from a TTP via the regeneration of key(s):

    a. Determine the appropriate value of *d*. For example, *d* may be selected in accordance with the assurance of possession model described in Section 6.2, along with appropriate values of *a*, *b* and *c*.

    If the TTP will be responsible for assigning the *assurance_time*, then the value of *d* **shall** be made known to the TTP.

    b. The owner and/or the TTP **shall** determine a value for t_1 that is trusted by the owner (see Sections 6.2 and 6.3.2.1).

    c. The owner **shall** provide the owner's currently-held keys to the trusted party, along with any other necessary data.

    d. The TTP **shall**:

    - Regenerate the owner's entire key pair, or
    - Regenerate one key of the owner's key pair from the currently-held value of the other key of that pair.

    e. The TTP **shall** compare the value(s) of the regenerated key(s) with the key(s) currently held by the owner.

    f. If the regenerated and currently-held keys match:

    - The owner **shall** be notified of the successful comparison.
    - The owner and/or the TTP **shall** determine a value for t_2 that is trusted by the owner (see Sections 6.2 and 6.3.2.1).
    - Provided that $t_1 \leq t_2 \leq t_1+d$, either the owner or the TTP **shall** assign the *assurance_time* and the initial assurance level (see Sections 6.3.2.1 and 6.3.2.2).

      The party who assigns the *assurance_time* must know the value of *d*, as well as the values of t_1 and t_2; the party who assigns the initial assurance level must also be aware of the method(s) used to determine t_1 and t_2.

    - The owner **shall** record the *assurance_time* and the initial level of assurance.

6.5.2 Assurance Obtained by a Trusted Third Party from the Owner

A TTP – who will be required to provide assurance to others of the owner's possession of the private key – **shall** first explicitly obtain that assurance from the owner either by verifying an assurance signature or by regeneration of the owner's key(s). In the case of regeneration, the TTP must be relied upon not to use knowledge of the owner's key pair to generate digital signatures. This trust must be shared by the owner of the key pair, potential signature-verifiers, and any other parties relying on the validity of the owner's signatures. The use of assurance signatures is preferred.

The TTP **shall** obtain assurance of a (claimed or intended) owner's possession of the private key using one or more of the following methods:

1. TTP obtains assurance from the owner by verifying an assurance signature.
    a. Determine the appropriate value of *d*. For example, *d* may be selected in accordance with the assurance of possession model described in Section 6.2, along with appropriate values of *a*, *b* and *c*.
    b. The owner **shall** generate a new private-key-possession assurance message (see Section 6.3.1.1).
    c. The owner **shall** sign the assurance message using the private key for which assurance is to be obtained, thus obtaining an assurance signature.
    d. The owner **shall** provide the assurance message, the assurance signature and any other necessary data to the TTP.
    e. The TTP **shall** determine a trusted value for t_1 (see Sections 6.2 and 6.3.1.2).
    f. The TTP **shall** verify the assurance signature using the public key that corresponds to the owner's private key.
    g. If verification is successful:
        - The TTP **shall** determine a trusted value for t_2 (see Sections 6.2 and 6.3.1.2).
        - Provided that $t_1 \leq t_2 \leq t_1+d$, the TTP **shall** assign the *assurance_time* and the initial assurance level (see Sections 6.3.1.2 and 6.3.1.3).
        - The TTP **shall** record the *assurance_time*, the initial assurance level, and the value of *d*. These values **should** also be provided to the owner.

2. TTP obtains assurance from the owner by regenerating the owner's key(s)
    a. Determine the appropriate value of *d*. For example, *d* may be selected in accordance with the assurance of possession model described in Section 6.2, along with appropriate values of *a*, *b* and *c*.
    b. The owner **shall** provide the owner's currently-held keys to the trusted party, along with any other necessary data.
    c. The TTP **shall** determine a trusted value for t_1 (see Sections 6.2 and 6.3.1.2).
    d. The TTP **shall**:
        - Regenerate the owner's entire key pair, or
        - Regenerate one key of the owner's key pair from the currently-held value of the other key of that pair.

e. The TTP **shall** compare the value(s) of the regenerated key(s) with the key(s) currently held by the owner.

f. If the regenerated and currently-held keys match:

- The TTP **shall** determine a trusted value for t_2 (see Sections 6.2 and 6.3.1.2).
- Provided that $t_1 \leq t_2 \leq t_1+d$, the TTP **shall** assign the *assurance_time* and the initial assurance level (see Sections 6.3.2.1 and 6.3.2.2).

  The TTP must know t_1 and t_2; the TTP must also be aware of the method(s) used to determine their values.
- The TTP **shall** record the *assurance_time*, the initial assurance level, and the value of *d*. These values **should** also be provided to the owner.

The TTP **shall** provide both the recorded *assurance_time* and the initial assurance level associated with the assurance of private key possession provided by the owner in response to a request by any relying party. In addition, the TTP may provide an assessment of its level of assurance that the owner is/was in possession of the private key at the time of the request for information (or any other time; see Sections 6.2 and 6.4).

Note that the values of *a*, *b*, *c* and *d* that are selected by the TTP may be different than the values deemed acceptable by other parties seeking assurance from the TTP. If this is the case, the third party **should** be prepared to (at least) provide an upper bound on the value of $t_2 - t_1$ (e.g., *d*) to potential relying parties – as well as a description of the time sources used – so that those parties can decide whether or not a TTP-provided *assurance_time* has sufficient accuracy (and trustworthiness) to meet their needs.

6.5.3 Assurance Obtained by a Verifier

Assurance that the (claimed) signatory was in possession of the private key at the time the signature was generated **shall** be obtained by the verifier prior to accepting a verified digital signature as valid.

The initial assurance of a (claimed) signatory's possession of the private key **shall** be obtained by the verifier through interaction with either the (claimed) signatory or a TTP. However, once the necessary actions have been taken, the *assurance_time* and the initial assurance level obtained may be used as the basis for obtaining assurance of private key possession by the same (claimed) signatory at other times (see Sections 6.2 and 6.4).

A verifier **shall** obtain assurance that a (claimed) signatory possesses the correct private key using one or more of the following methods:

1. Assurance obtained by verifying an assurance signature obtained by interaction with the (claimed) signatory.

   a. Determine the appropriate value of *d*. For example, *d* may be selected in accordance with the assurance of possession model described in Section 6.2, along with appropriate values of *a*, *b* and *c*.

b. The (claimed) signatory **shall** provide a new private-key-possession assurance message to the verifier (see Section 6.3.1), along with a corresponding assurance signature, and any other necessary data.

e. The verifier **shall** determine a trusted value for t_1 (see Sections 6.2 and 6.3.1.2).

f. The verifier **shall** verify the assurance signature using the public key of the key pair owned by the (claimed) signatory.

g. If verification is successful:

- The verifier **shall** determine a trusted value for t_2 (see Sections 6.2 and 6.3.1.2).
- Provided that $t_1 \leq t_2 \leq t_1+d$, the verifier **shall** assign the *assurance_time* and the initial assurance level (see Sections 6.3.1.2 and 6.3.1.3).
- The verifier **shall** record the *assurance_time* and the initial assurance level.

2. Assurance obtained via interaction with a TTP (trusted by the verifier):

a. Determine the appropriate value of *d*. For example, *d* may be selected in accordance with the assurance of possession model described in Section 6.2, along with appropriate values of *a*, *b* and *c*.

b. The verifier **shall** request assurance of the (claimed) signatory's possession of its private signature key from a TTP.

If the TTP has obtained this assurance:

c. The TTP **shall** provide the method of obtaining the assurance, the recorded *assurance_time*, and an initial level of assurance of private key possession.

d. If requested by the verifier – the TTP **should** provide an upper bound on the value of $t_2 - t_1$, a description of the method(s) used to obtain the times t_1 and t_2 (see Sections 6.3.1.2 and 6.3.2.1), and/or the TTP's assessment of its level of assurance that the (claimed) signatory is/was in possession of the private key at the time of this request.

e. The verifier **shall**:

- (If provided by the TTP) reject the TTP-provided assurance if the upper bound on $t_2 - t_1 > d$; otherwise,
- Record the *assurance_time* and the initial assurance level provided by the TTP, and/or
- (If provided by the TTP) record the TTP's assessment of its level of assurance that the (claimed) signatory is/was in possession of the private key at the time of this request, and/or

- (If provided by the TTP) use the description of the method(s) used to obtain the times t_1 and t_2 to determine whether or not to adjust the TTP-provided assurance level(s) (see Sections 6.2, 6.3.1.3 and 6.3.2.1).

7 Assurance of Identity

A digital signature verifier may require various levels of assurance of the claimed signatory's identity. The verifier may require assurance of the actual identity of the signatory (e.g., the signatory is the John Smith who works in division 301 of the XYZ agency), or that the signatory is authorized by agency XYZ to purchase 15 computers, or that the signatory is device X on network Y; a verifier may allow cases where the signatory is anonymous.

Requirements for the assurance associated with a signatory's identity **shall** be addressed in an organization's security policy. A verifier's requirements for signatory identity **shall** be consistent with the requirements for establishing a signatory's identity.

Appendix A: The Assurance Message Instantiated via an RFC 4211 Certificate Request Message

From Section 6.3.1.1 about the The Private-Key-Possession Assurance Message:

When a digital signature is used to provide assurance of private key possession, the verifying entity (who must be trusted by all relying parties) is sent a signed *private-key-possession assurance message* (simply called the *assurance message* herein); the digital signature on the assurance message is called the *assurance signature*.

The assurance message **shall** include the following:

1. The signatory's identity;

   Note: This should either be in the **certReq** item, e.g., in the **subject** field of the **certTemplate,** or might be handled by the **authInfo** field (e.g., in the form of the password and identity string used in the **publicKeyMAC** computation) of **poposkInput** in the **popo** item.

2. The intended verifier's identity;

   Note: This should either be in the **certReq** item, e.g., in the **issuer** field of the **certTemplate**, or might be handled by the **authInfo f**ield (e.g., in the form of the password and identity string used in the **publicKeyMAC** computation) of **poposkInput** in the **popo item**.

3. A timestamp token (TST) created by a Trusted Timestamp Authority (TTA) that is trusted by all relying parties. This TST could be obtained from the TTA by the signatory or by the intended verifier (who then provides it to the signatory). Relying parties must be aware of the security strength provided by the TTA's digital signature. The security strength provided by the TTA's digital signature **shall** meet or exceed the security requirements of the requesting entity and relying party (see SP 800-57);

   Note: This could be included in the **certReq** item, as one of the **controls.**

   AND/OR

   A *nonce* supplied by the intended verifier. If this option is used, the nonce **shall** contain a random component with entropy equal to or greater than the security strength associated with the private key for which assurance of possession is sought. If the assurance message will not include a TST, and the relying parties require that an *assurance_time* be recorded upon successful verification of the assurance signature, then the nonce **should** also contain a verifier-supplied timestamp component that unambiguously represents the time that the nonce was made available for inclusion in the assurance message.

Note: This could be included in the **certReq** item, as one of the **controls**, e.g., as a **regToken control** or an **authenticator control**; it might also take the form of a salt value[8] provided for use in the **publicKeyMAC** computation included in the **authInfo** field of **poposkInput** in the **popo** item.

4. The public key corresponding to the private key for which assurance of possession is sought**.**

   Note: This should be in the **certReq** item, in the **publicKey f**ield of the **certTemplate**, and/or in the **publicKey** field of **poposkInput** in the **popo** item.

[8] A random string of bits.

Appendix B: References

[1] Federal Information Processing Standard (FIPS) 186-3, The Digital Signature Standard, TBD.

[2] NIST Special Publication (SP) 800-57, Recommendation for Key Management – Part 1: General, Revised June 12, 2006.

[3] ANS X9.95-2005, Trusted Time Stamp Management and Security.

[4] ISO/IEC 18014, Information Technology – Security techniques – Time-stamping services, 2005