

# On the usefulness of proof-of-possession

N. Asokan, Valtteri Niemi, Pekka Laitinen  
Nokia Research Center, Finland

{n.asokan, valtteri.niemi, pekka.laitinen}@nokia.com

## Abstract

Public key infrastructure standards assert that proof-of-possession of private key is an essential requirement during the enrollment process. Even though the justifications for this requirement seem to be well-known within the PKI community, they do not appear to be documented anywhere. In this paper, we document and examine potential rationales for proof-of-possession and discuss their merits. We conclude that if protocols and applications are designed “properly”, proof-of-possession does not add any security. However, the world is not perfect. Many existing applications and protocols are in fact not properly designed. Proof-of-possession is a useful safety precaution for the users of such applications and protocols. But there is no simple automated way for a relying party application to check whether proof-of-possession was done during enrollment. Therefore, we argue that designers of public key protocols *must not* assume that CAs require proof-of-possession during enrollment.

## 1 What is proof-of-possession?

In a public key infrastructure (PKI), the process of submitting a certificate request to a certification authority (CA) or a registration authority (RA) is known as *enrollment*. After enrollment, the CA will issue a certificate to the enrolled public key. During enrollment, the end entity that submits the public key may be required to prove that it knows the corresponding private key and that it controls the use of this private key. This is commonly referred to as the *proof of possession* (PoP).

Every PKI standard asserts that PoP is essential. However, none of them explicitly lays out the threats that are intended to be addressed by PoP. The rationales and implications of PoP have been discussed in standards meetings and mailing lists [6, 7]. Yet, there does not appear to be any easily available or commonly known papers or articles that document these issues. It appears to be yet another case of undocumented folklore within the communities involved.

In this paper, we examine the potential rationales for PoP and discuss their merits. Our goal is to clarify the answers to the following questions:

- Should the designer of a new PKI require PoP during enrollment?
- Does the designer of a new public key based application or protocol benefit from having PoP done during enrollment?

Our work was motivated by the on-going work in the 3<sup>rd</sup> generation partnership project (3GPP) for designing support for subscriber certificates [10]. 3GPP security group considered various ways of securing the enrollment messages. One of them was to use the cellular signaling channel which provides mutual authentication and integrity-protection. This channel is severely bandwidth-limited. Thus it was necessary to check that every bit sent through this channel is really essential. This prompted us to start investigating the conditions under which PoP is indeed indispensable.

In the rest of the paper, we use the term “PoP” as it is customarily used, without any additional qualification. The precise characterization is “proof-of-possession of private key during enrollment.” Public key protocols often involve other types of proofs of possession: for example, every time a relying party verifies a signature, it is proof that the signer possessed the signing key; “plaintext-aware” encryption schemes [8] include a proof that an entity claiming to have produced a ciphertext actually knew (hence possessed) the plaintext. Such proofs of possession are not the subject of this paper.

In Section 2 we begin by defining the ways in which a private key of an asymmetric cryptosystem is used. In Section 3 we describe how the public key enrollment process is secured in PKIs. In Section 4 we describe attacks that are not intended to be prevented by PoP, and in Section 5 we describe potential attacks that *can be* prevented by PoP. In Section 6 we consider scenarios where mandating PoP is not advisable. In Section 7 we briefly describe the degree

to which existing PKI specifications require PoP. In Section 8 we consider other possible rationales for requiring PoP. Finally, in Section 9 we summarize our findings.

## 2 Use of a private key

There are three primary ways in which a private key of an asymmetric key pair is used:

- **commitment:** By signing a message with the private key, the purported controller of the private key commits himself to the signed message. Enabling non-repudiation is an example application of this kind.
- **claim:** By signing a message with the private key, or by decrypting a ciphertext (and demonstrating knowledge of the plaintext), the purported controller of the private key can stake a claim for a benefit or ownership. For example, if the signature is on a plaintext message  $m$  and the signature is accompanied by an identity certificate binding an entity  $X$  to the signature verification key, the claimed fact may be “ $X$  is certified to have control of the signing key corresponding to this signature verification key, and therefore message  $m$  signed by this key is known to  $X$  and uttered by  $X$ .” In other words, message authentication is an example application of this kind.
- **encryption:** Of course, a keypair can also be used just for encryption. This has similar properties like the “claim” use case.

In standard X.509v3 certificates, it is possible to use the `keyUsage` or `extKeyUsage` parameters to indicate the types of uses (e.g., “non-repudiation”) to which the public key certificate is limited.

## 3 Security of enrollment

To enroll in a PKI an end entity will send a certificate request to the CA/RA. A certificate request contains a claimed public key, and any additional information, such as a claimed identity and attributes, and additional certificates in support of the request. Attribute certificates do not contain a public key: instead they bind a name to a set of attributes. When requesting an attribute certificate, the request may not contain public key. We do not consider this case further because use of such certificates should be accompanied by an identity certificate that binds a public key to the said name. Therefore PoP does not appear to be relevant when requesting attribute certificates.

The enrollment process must be secured so that the CA/RA can associate the submitted public key with the correct authorizations allowed for the submitter. An example of such an authorization is the right to be bound to the identity of a specific end entity. Typically this is done by distributing a shared one-time key or password ahead of time, for example, by mailing scratch cards containing initial PIN codes to potential users of PKI. Such a key or password is known as an initial authentication key. The initial authentication key will be used to authenticate and integrity-protect messages in the enrollment process. There are also other ways of securing enrollment, for example by using any existing or derivable security association between the end entity and the CA/RA, e.g., as done in the PIC protocol [11]. This approach is applicable when we need to bootstrap a PKI from an existing infrastructure [10]. The purpose of this authentication is to allow the CA/RA to determine whether and how to approve the certificate request.

## 4 Examples of attacks not prevented by PoP

The certificate request is authenticated as mentioned in Section 3. If the cryptographic transforms used are inappropriate, it will allow an attacker to compromise the security of the enrollment process: an attacker on the network can change the contents of the request or fake a new request without being detected. We can call this the “weak password attack.” PoP does not help against this attack. PoP is not intended to be a replacement for having to design appropriate mechanisms for the security of enrollment.

If PoP is required as part of certificate request, it assures the CA/RA that the requestor had access to the private key corresponding to the public key on which the certificate is requested.

Suppose the private key is used for commitments only, such as for non-repudiation. If PoP is not done during the certificate request process, a legitimate end entity Alice can indeed obtain a certificate binding her name to the public key of another end entity Bob. But this only leaves Alice, the supposed attacker, liable for commitments made by Bob, who controls the private key and can make signatures. In other words, this is not a protocol attack. (However, as described in Section 5.2.2, there are protocol attacks related to claim type uses.) In fact, Alice may use a slightly different variation, as we discuss in Section 6, to delegate authority to Bob.

Note that Alice still cannot send an arbitrary message  $m$  and claim that it belongs to and/or was sent by Bob. The security of the enrollment process (Section 3) is intended to prevent Alice from being able to send arbitrary signed message and fool the attacker into thinking that they came from Bob. See Section 5.1 for more discussion.

## 5 Attacks prevented by PoP

### 5.1 Replacing public key in enrollment request sent by victim

#### 5.1.1 Attack description and assumptions

In a certificate request sent by Bob, the attacker replaces Bob's public key by Alice's public key. The effectiveness of this attack is contingent on the following assumptions being valid:

1. *Security of enrollment is weak*: Normally, this attack is prevented by the mechanism for securing enrollment. One aspect of securing enrollment is the protection of the certificate request message itself. As discussed in Section 4, PoP is not relevant from this aspect. A second aspect of securing enrollment is the access control on the devices used by the victim Bob. The access control mechanisms on Bob's device may not be robust enough to prevent the attacker from (i) changing the contents of the local public key repository without being detected, or (ii) accessing the secret keys used to ensure the security of enrollment. We can call this the "Trojan attack."
2. *Although the attacker is able to insert a public key into Bob's device, it is unable to insert a private key or intercept the communication path between the signing algorithm and the calling application*: if this were not true, the attacker could insert a whole key pair into Bob's device. Thereafter PoP is of no use. Note that Alice need not suffer any harm by inserting a private key: it can be the private key of a keypair that the attacker generated solely for this attack. Alternately, even if the attacker cannot insert the private key in the standard place where private keys are stored on Bob's device (e.g., a smart-card), it is enough if the attacker intercepts private key operation requests or replaces the responses.

#### 5.1.2 Type of uses

In commitment type uses, if the public key is a signature verification key, then Alice can make commitments (signatures), have them supported by the certificate, and leave Bob liable for them.

In claim type uses, if the public key is an encryption key, then Alice can send it to a peer Carol and trigger Carol into encrypting, with this key, some confidential data intended for Bob.

In either case any protection provided by PoP is subject to both the assumptions listed in Section 5.1.1 being true. The assumptions are not very realistic because they require the access control on the victim's device to be faulty in a very specific manner. If the victim's device does not have adequate access control, PoP does not help because the attacker can make sure that the certificate request has the correct signature as described above in assumption 2. If the victim's device is indeed secure, then the Trojan attack should not succeed, and assumption 2 would not hold.

### 5.2 Using victim's public key in own enrollment request

#### 5.2.1 Attack description and assumptions

In a certificate request sent by Alice, she can use the public key of another legitimate end entity Bob. As explained in the next section, the basic attack does not appear to rely on any strong assumption other than badly designed applications or application protocols.

#### 5.2.2 Type of uses

Without PoP, CA may issue Alice a certificate containing Bob's public key. Depending on the type of use, this threat can be turned into concrete attacks as follows.

In claim type uses

1. if the public key is a signature verification key, then Alice could falsely claim ownership of messages that were signed by Bob. Note that if the application protocol would bind some identification of the sender within the signed message, and the verifying party's application would compare this identification with what is in the certificate used to verify the signature, then this attack would not work. We can call this the "sloppy application protocol attack."
2. if the public key is a signature verification key, then Alice can mislead a peer Carol into revealing to her some private data that Carol intended for Bob. This works as follows [12]:

Alice obtains a certificate for Bob's public signature verification key. Then Alice waits until Bob sends a signed, encrypted message and Bob's certificate to Carol. Alice intercepts these. Alice does not know what the contents of the message were. She then forwards this intercepted message to Carol along with the certificate she obtained earlier. Carol concludes that the message actually came from Alice because all cryptographic checks

succeed. This might cause Carol to send some private information to Alice in the clear (e.g., something about the contents of the message she just received).

Again, this is a variation of the “sloppy application protocol attack:” if the messaging protocol required that the sender’s identity must be included in the signed text, Carol’s software would notice that the certificate and the signed data do not match.

In commitment type uses, there does not appear to be any effective protocol attack as a result of this flaw. However, there may be a software program like an e-mail client that does use public keys in both ways (commitment and claim). If the application is not properly written, it may either not pay attention to the limited use of a certificate (such as `keyUsage` set to “non-repudiation”), or if user is careless, she might not notice that the signature does not authenticate the sender. In either case she may incorrectly conclude the sender is authenticated and act based on this conclusion. For example, in example 2), suppose the certificate has `keyUsage` set to “non-repudiation”, but the e-mail client of Carol does not indicate this unambiguously. So Carol may be misled into assuming that the signature and certificate authenticate the sender, and make the same conclusions as though the certificate is also intended for “claim” type of uses. We can call this the “sloppy application attack.”

The attack against claim type use assumes that application software and application protocol designers may have made some basic mistakes. The security of the users of such applications can benefit from mandatory PoP. The attack against commitment type use can also be avoided if

1. a keypair has only one type of use, e.g., either authentication or non-repudiation<sup>1</sup>, and
2. the software application of the relying party handles `keyUsage/extendedKeyUsage` restrictions correctly.

If it is difficult to ensure either of the above (e.g., it may be difficult to mandate the former, and unrealistic to expect the latter) then PoP can help limit the damage. However, attempting to provide protection against sloppy application designers is ultimately a doomed exercise.

## 6 Does PoP do any harm?

We saw that PoP could potentially offer some protection for users of badly designed applications and protocols. If PoP has no harmful consequences, requiring PoP is a prudent safety precaution. So, the logical next question is whether PoP does any harm. We consider the following factors.

- **Bandwidth:** PoP means that the requestor has to perform a private key operation. This in turn implies that a large message (e.g., signature or encryption) needs to be sent between the requestor and the CA/RA. If the certificate request channel is bandwidth constrained, size of messages becomes a factor. As we mentioned in Section 1, this was the context in which we started to investigate whether PoP is indeed indispensable.
- **Latency:** To prevent against replay attacks the PoP protocol must ensure freshness. As usual, this can be done using timestamps, which requires synchronized clocks. A better alternative is where the CA/RA sends a challenge nonce. But this means that the certificate request procedure will typically contain an extra request/response pair.
- **Novel applications:** It appears that the need for PoP arose from the “traditional” PKI scenarios where the certificates issued are identity certificates. In such cases, it is of course quite logical to try to prevent two persons from attempting to get certificates for the same public key. However, as limited scope PKIs are becoming more realistic and more prevalent, there may be new uses that are prevented or made harder by mandating PoP. For example, suppose certificates are used for authorizing payments from a bank account. A use case may be for Alice to obtain a certificate for Bob’s public key as a surprise gift voucher (so that Bob is allowed to spend a certain amount of money which will be paid from Alice’s account). If PoP is required, then obtaining such a certificate will require Bob’s involvement, which will eliminate the surprise factor, and hence the point of this use case! Note that the certificates used in this case would not bind Alice’s identity to a public key. Instead, they would bind some authorizations to a public key. In other words, they would be authorization certificates [4], rather than identity certificates.

Although mandatory PoP will prevent a solution to this use case using standard certificates, it is of course possible to design solutions using other types of constructs, e.g., by defining some form of delegation tokens.

Enrollment is a relatively rare occurrence. In typical scenarios, it does not have any real-time requirements. Therefore, we conclude that in general, bandwidth and latency are not critical factors. While mandatory PoP may in fact preclude some class of applications using certificates, there are other ways of designing these applications. Thus, we conclude that PoP does no harm.

---

<sup>1</sup>Achieving non-repudiation requires a lot more than digital signatures; but discussion on the usefulness of “non-repudiation” as a `keyUsage` is beyond the scope of this paper.

## 7 The place of PoP in current PKI specifications

The PKCS #10 specification [9], designed by RSA Laboratories, states the following:

“The signature on the certification request prevents an entity from requesting a certificate with another party’s public key. Such an attack would give the entity the minor ability to pretend to be the originator of any message signed by the other party. This attack is significant only if the entity does not know the message being signed and the signed part of the message does not identify the signer. The entity would still not be able to decrypt messages intended for the other party, of course.” (Section 3, Note 2 of [9]).

The threat described here is the sloppy application protocol attack we discussed in Section 5.2.2. Surprisingly, the wording of the above text, quoted from PKCS #10, suggests that the impact of this attack is minor. Nevertheless PKCS #10 mandates the use of PoP.

The Wireless PKI specification [13], designed by the precursor to the Open Mobile Alliance, states that PoP is necessary “in order to avoid certain substitution attacks” (Section 4.1) but it does not describe the attacks themselves.

The IETF CMP specification [2] states that PoP is necessary “in order to prevent certain attacks and to allow a CA/RA to properly check the validity of the binding between an end entity and a key pair” (Section 2.3 of [2]). It does not describe what these attacks may be or whether they are limited to the case of identity certificates only.

The e-mail archives of the IETF PKIX working group [6, 7] contain records of extensive discussions on whether PoP should be mandatory. The participants recognized both the assumptions under which PoP is useful (e.g., sloppy application protocols) and the limitations that PoP may impose (e.g., precluding novel applications).

As a result of these discussions, the working group appears to have chosen to require PoP, but *not to mandate PoP to be part of the certificate request protocol itself*. The current CMP specification contains the following explanation:

“... it is REQUIRED that CAs/RAs MUST enforce POP by some means because there are currently many non-PKIX operational protocols in use (various electronic mail protocols are one example) that do not explicitly check the binding between the end entity and the private key. Until operational protocols that do verify the binding (for signature, encryption, and key agreement key pairs) exist, and are ubiquitous, this binding can only be assumed to have been verified by the CA/RA.”[2]

## 8 Rationales for justifying PoP

One of the primary reasons for requiring PoP seems to be to minimize potential damage due to

- badly designed application-level protocols, or
- badly designed end entity application software, or
- carelessness of an end entity user.

The attacks described in Section 5.2.2 may become possible due to one of the factors listed above. PoP could potentially reduce the likelihood of the resulting attacks. CA operators therefore may view PoP as a way of protecting them from liability arising from damage due to these factors.

However, currently there is no easy *automated* way for a relying party application to check if PoP was done during enrollment. This is because there is no standard place in a certificate for the CA to indicate this. In order to benefit from PoP, relying parties must make sure that they never use certificates issued by CAs that do not require PoP. For example, users may have to examine that certification practice statements (CPSs) of CAs before accepting certificates issued by them. Needless to say, this is not a pragmatic solution.

Therefore, any good application developer has to assume that PoP was not done at the time of enrollment. In particular, an application developer must

- explicitly include all necessary identification and context information in the parts of application protocol messages that are cryptographically protected, (for example, a PKI-enabled e-mail client could include the name of the sender in the signed text; signature verification should fail if this address does not match the name in the certificate used to verify the signature.)
- require the use of different keys for different purposes, and
- consistently and correctly identify the purpose of a given key (e.g., by precisely defining the semantics of the `keyUsage` attributes) so that it is not used for a different purpose.

Such rules are part of the general guidelines for well-designed cryptographic protocols discussed elsewhere [1, 3] and are applicable in this context. The first rule was also repeatedly pointed out in the IETF PKIX mailing list discussions [6, 7].

## 9 Conclusions

Several standards allude to unspecified attacks in justifying why PoP is needed. We discussed potential threats and discussed how PoP can help reduce their impact. A well designed application protocol does not need PoP. However, many existing protocols and applications are not well designed in this sense. PoP is useful as a safeguard for users of such applications and protocols.

Mandating PoP has some drawbacks. It will preclude the use of standard certificates to achieve one class of use cases where Alice is allowed to delegate authority to Bob by obtaining a certificate for Bob's public key without Bob's involvement. Also, if the communication channel used for enrollment is resource constrained, it is necessary to check if PoP is really needed for the application under consideration. But none of these drawbacks is substantial.

It is becoming increasingly clear that the successful uses of PKI tend to be for specific applications [5]. Designers of application-specific PKIs can and should check if PoP is really needed for the applications of interest to them.

Thus we conclude that by and large, requiring PoP during enrollment is a useful safety precaution because of the shortcomings in applications that are already widely deployed. Designers of new PKIs should require it, especially if there is any likelihood that their PKI will be used with legacy applications.

However, as there is no simple automated way for a relying party application to check whether PoP was done during enrollment, we argue that designers of new security protocols and applications *must not* assume that CAs require PoP during enrollment. They must follow the well known rules of secure protocol design referred to in Section 8.

## 10 Acknowledgments

We thank the anonymous referees, Antti Vähä-Sipilä, Jukka Virtanen, Kaisa Nyberg, Michael Waidner, Pasi Eronen, Philip Ginzboorg, Olli Immonen, and the members of the 3GPP SA3 working group for their valuable feedback on previous versions of this paper.

## References

- [1] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.
- [2] C. Adams and S. Farrell. Internet X.509 public key infrastructure: Certificate management protocols. Internet Engineering Task Force, RFC 2510, March 1999.
- [3] Ross Anderson and Roger Needham. Robustness principles for public key protocols. In Don Coppersmith, editor, *Advances in Cryptology; CRYPTO '95*, number 963 in Lecture Notes in Computer Science, pages 236–247. Springer-Verlag, 1995.
- [4] C. Ellison et al. SPKI Certificate Theory. Internet Engineering Task Force, RFC 2693, September 1999.
- [5] Peter Gutmann. PKI: It's Not Dead, Just Resting. *IEEE Computer*, 35(8):41–49, August 2002.
- [6] IETF PKIX Mailing list discussions. IETF PKIX mailing list archive, February 1997. <http://www.imc.org/ietf-pkix/old-archive-97/thrd3.html#00081>.
- [7] IETF PKIX Mailing list discussions. IETF PKIX mailing list archive, October 1997. <http://www.imc.org/ietf-pkix/old-archive-97/threads.html#01062>.
- [8] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Available from <http://www.cacr.math.uwaterloo.ca/hac>.
- [9] RSA Laboratories. PKCS #10 v1.7: Certification Request Syntax Standard. RSA Laboratories, May 2000. Also published as IETF RFC 2986.
- [10] 3GPP security group. Support for subscriber certificates. Third generation partnership project (3GPP), Security working group work item description, February 2002. Available from [http://www.3gpp.org/ftp/tsg\\_sa/WG3\\_Security/TSGS3\\_22\\_Bristol/Docs/PDF/S3-020162.pdf](http://www.3gpp.org/ftp/tsg_sa/WG3_Security/TSGS3_22_Bristol/Docs/PDF/S3-020162.pdf).
- [11] Y. Sheffer, H. Krawczyk, and Bernard Aboba. PIC, A Pre-IKE Credential Provisioning Protocol, October 2002. IETF *ipsra* working group draft `draft-ietf-ipsra-pic-06.txt`.
- [12] Michael Waidner. Personal communication, August 2002.
- [13] WAP Forum. Wireless application protocol, public key infrastructure definition. WAP forum, WAP-217-WPKI, April 2001.