Network Working Group
Request for Comments: 2726
Category: Standards Track

J. Zsako
BankNet

December 1999

PGP Authentication for RIPE Database Updates

Status of this Memo

Copyright Notice

Abstract

This document presents the proposal for a stronger authentication method of the updates of the RIPE database based on digital signatures. The proposal tries to be as general as possible as far as digital signing methods are concerned, however, it concentrates mainly on PGP, as the first method to be implemented. The proposal is the result of the discussions within the RIPE DBSEC Task Force.

1. Rationale

An increasing need has been identified for a stronger authentication of the database maintainer upon database updates (addition, modification and deletion of objects). The existing authentication methods have serious security problems: the MAIL-FROM has the drawback that a mail header is very easy to forge whereas CRYPT-PW is exposed to message interception, since the password is sent unencrypted in the update mail message.

The goal was to implement a digital signature mechanism based on a widely available and deployed technology. The first choice was PGP, other methods may follow at a later date. PGP is presently quite widely used within the Internet community and is available both in and outside the US.

The current aim is for an improved authentication method and nothing more (in particular, this paper does not try to cover authorization issues other than those related to authentication).

2. Changes to the RIPE database

In order to make the database as much self consistent as possible, the key certificates are stored in the RIPE database. For efficiency reasons a local keyring of public keys will also be maintained, however, the local keyring will only contain a copy of the key certificates present in the database. The synchronization of the database with the local keyring will be made as often as possible. The database objects will be created only via the current e-mail mechanism (auto-dbm@ripe.net), in particular no public key certificate will be retrieved from a key server by the database software.

The presence of the key certificates in the database will allow the users of the database to check the "identity" of the maintainer, in the sense that they can query the database for the certificate of the key the database software uses for authenticating the maintainer. This key certificate can then be checked for existing signatures and can possibly be compared with the key certificate obtained by other means for the same user (e.g. from the owner himself of from a public key server). Although the key certificates can be stored in the RIPE database with any number of signatures, since the RIPE database is not communicating directly with the public key servers, it is a good practice to add the key certificate with the minimum number of signatures possible (preferably with just one signature: the one of itself). See also section 4. for more details.

2.1. The key-cert object

A new object type is defined below for the purpose of storing the key certificates of the maintainers:

key-cert:	[mandatory]	[single]	[primary/look-up key]
method:	[generated]	[single]	[ ]
owner:	[generated]	[multiple]	[ ]
fingerpr:	[generated]	[single]	[ ]
certif:	[mandatory]	[single]	[ ]
remarks:	[optional]	[multiple]	[ ]
notify:	[optional]	[multiple]	[inverse key]
mnt-by:	[mandatory]	[multiple]	[inverse key]
changed:	[mandatory]	[multiple]	[ ]
source:	[mandatory]	[single]	[ ]

The syntax and the semantics of the different attributes are described below.

key-cert: Is of the form PGPKEY-hhhhhhhh, where hhhhhhhh stands for for the hex representation of the four bytes ID of the PGP key. The key certificate detailed in the certif attribute belongs to the PGP key with the id hhhhhhhh. The reason for having PGPKEY- as a prefix is to allow for other types of key certificates at a later date, and at the same time to be able to clearly differentiate at query time between a person query and a key certificate query. At the time of the creation/modification of the key-cert object, the database software checks whether the key certificate in the certif attribute indeed belongs to the PGP id specified here. The creation/modification is authorized only upon the match of these two ids.

method: Line containing the name of the signing method. This is the name of the digital signature method. The present certificate belongs to a key for digitally signing messages using the specified method. The method attribute is generated automatically by the database software upon creation of the key-cert object. Any method attribute present in the object at the time of the submission for creation is ignored. The method has to be consistent with both the prefix of the id in the key-cert attribute and with the certificate contained in the certif attributes. If these latter two (i.e. prefix and certificate) are not consistent, the key-cert object creation is refused. For the PGP method this will be the string "PGP" (without the quotes).

owner: Line containing a description of the owner of the key. For a PGP key, the owners are the user ids associated with the key. For each user id present in the key certificate, an owner attribute is generated automatically by the database software upon creation of the key-cert object. Any owner attribute present in the object at the time of the submission for creation is ignored.

fingerpr: A given number of hex encoded bytes, separated for better readability by spaces. It represents the fingerprint of the key associated with the present certificate. This is also a field generated upon creation of the object instance. Any fingerpr attribute submitted to the robot is ignored. The reason for having this attribute (and the owner attribute) is to allow for an easy check of the key certificate upon a query of the database. The querier gets the owner and fingerprint information without having to add the certificate to his/her own public keyring. Also, since these two attributes are _generated_ by the database software from the certificate, one can trust them (as much as one can trust the database itself).

certif: Line containing a line of the ASCII armoured key certificate. The certif attribute lines contain the key certificate. In the case of PGP, they also contain the delimiting lines (BEGIN/END PGP PUBLIC KEY BLOCK). Obviously the order of the lines is essential, therefore the certif attribute lines are presented at query time in the same order as they have been submitted at creation. A database client application could contain a script that strips the certif attribute lines (returned as a result of a query) from the leading "certif:" string and the following white spaces and import the remainder in the local keyring.

mnt-by: The usual syntax the usual semantics this attribute is _mandatory_ for this object. Therefore, the existence of a mntner object is a prerequisite for the creation of a key-cert object. The mntner referenced in the mnt-by attribute may not have the auth attribute set to NONE.

remarks:,
notify:,
changed:,
source: the usual syntax and semantics.

In the case of PGP, when a key-cert object is created, the associated key is also added to a local keyring of public keys. When a key-cert object is deleted, the corresponding public key is deleted from the local keyring as well. Whenever a key-cert object is modified, the key is deleted from the local keyring and the key associated with the new certificate is added to the keyring (obviously this is performed only when the database update is authorized, in particular if the new key certificate does belong to the id specified in the attribute key-cert, see above).

2.2. Changes to the mntner object

The only change is that there is a new possible value for the auth attribute. This value is of the form PGPKEY-<id>, where <id> is the hex representation of the four bytes id of the PGP public key used for authentication.

The semantics of this new value is that the PGP key associated with the key certificate stored in the key-cert object identified by PGPKEY-id is used to check the signature of any creation/modification/deletion message sent to auto-dbm@ripe.net affecting an object maintained by this mntner.

Just as with other values, the auth attribute can be multiple. It does not make much sense to have two auth attributes with different methods (e.g. PGPKEY-<id> and NONE :))), just as it didn't earlier either.

If there are several auth methods with a PGPKEY-<id> value, the semantics is the already known one, namely that _either_ signature is accepted.

3. The PGP signed creation/modification/deletion

The whole message has to be signed. This means, that the database software first checks whether the message is a PGP signed message. If it is, it checks for a valid signature and associates this signature with the objects submitted in the message. A message may contain only one PGP signature.

If an object present in a message has a mnt-by attribute, and the respective mntner has auth attribute(s) with PGPKEY-<id> value, the database software checks whether the object has a signature associated with it (i.e. whether the message being processed had been signed) and whether the type of the signature (PGP in this implementation phase) and the id of the key used for signing the message is the same as the one in (one of) the auth attribute(s). The creation/modification/deletion of the object is performed only if this authentication succeeds.

This approach allows for a simplification of the message parsing process. A different approach would be necessary if one would sign the _objects_, rather then the update messages. In case the objects would be signed, the parser would have to identify which objects were signed, check the signature(s) on each object individually and permit/refuse the update at an object level, depending on (amongst others) whether the signature is valid and whether it belongs to (one of) the maintainer(s). This approach would allow for mixing in the same e-mail message objects signed by different maintainers (which would probably not be typical), and it would also allow for storing the signature in the database (in order to allow for the verification of the signature at query time). This latter (i.e. storing the signatures in the database) is beyond the scope of the first phase of the implementation. It may become a goal at a later date.

It is recommended to check that the mailer program does not make any transformations on the text of the e-mail message (and possibly configure it not to do any). Such common transformations are line-wrapping after a given number of characters, transforming of tabs in spaces, etc. Also check that you only use ASCII characters in the message.

4. Requirements the PGP key certificates must meet

There is no limitation imposed with respect to the version of the PGP software that is/was used for the creation of the key. Key of both version 2.x and 5.0 are supported, although the keys generated with version 5.0 are recommended.

The key certificates submitted for creating a key-cert object must contain a signature of the key itself. Although the certificate may contain other signatures as well, it is recommended to create the key-cert object with as few signatures as possible in the certificate. Anyone concerned about the trustfulness of the key should retrieve a copy of the key certificate from a public key server (or by any other appropriate means and check the signatures present in _that_ certificate. If such a check is performed one should take care to check both the key fingerprint and the key type and length in order to make sure the two certificates (the one retrieved from the RIPE database and the one retrieved from the public key server or collected by other means) belong to the same key.

Although it is highly unlikely, it may happen that a key-cert with the id identical to the id of the key of a maintainer already exists in the RIPE database. In case this latter key had been used for a while and it had been registered at public key servers for some time, the given person should contact the RIPE NCC and report this to ripe-dbm@ripe.net. Anyway, he/she may have to create a new key and register _its_ certificate into the RIPE database. Such a procedure, although highly unlikely to happen, should not create serious problems to the respective maintainer.

5. Short overview of the tasks to be performed in order to use PGP authentication

You must have a mntner object in the RIPE database with auth: other than NONE. The mntner object has to be created in the traditional way.

You must get a certificate of your own key and prepare a key-cert object from it. The object has to reference in mnt-by the mntner mentioned above.

Create the key-cert object in the RIPE database, by sending the object prepared above to auto-dbm@ripe.net. Obviously you must pass the authentication checks required by the mntner object (i.e. mail from a predefined address or send the correct password).

Change the mntner object to have the auth: attribute value of PGPKEY-<id>, where <id> is the hex id of your PGP key.

Check all objects maintained by the given mntner (preferably with the command This is the only way to benefit from the stronger authentication method in order to assign more trustfulness to the database. Remember that you are the only person who can check for and correct possible inconsistencies.

From now on always sign the (whole) update messages that refer to objects maintained by you, with the key you submitted to the RIPE database.

6. Example of objects using the new feature

```
mntner:      AS3244-MNT
descr:       BankNet, Budapest HU
descr:       Eastern European Internet Provider via own VSAT network
admin-c:     JZ38
tech-c:      JZ38
tech-c:      IR2-RIPE
upd-to:      ncc@banknet.net
mnt-nfy:     ncc@banknet.net
auth:        PGPKEY-23F5CE35
remarks:     This is the maintainer of all BankNet related objects
notify:      ncc@banknet.net
mnt-by:      AS3244-MNT
changed:     zsako@banknet.net 19980525
source:      RIPE

key-cert: PGPKEY-23F5CE35
method:   PGP
owner:    Janos Zsako <zsako@banknet.net>
fingerpr: B5 D0 96 D0 D0 D3 2B B2  B8 C2 5D 22 D4 F5 78 92
certif: -----BEGIN PGP PUBLIC KEY BLOCK-----
 Version: 2.6.2i
+
 mQCNAzCqKdIAAAEEAPMSQtBNFFuTS0duoUiqnPHm05dxrI76rrOGwx+OU5tzGavx
 cm2iCInNtikeKjlIMD7FiCH1J8PWdZivpwhzuGeeMimT8ZmNn4z3bb6ELRyiZOvs
 4nfxVlh+kKKD9JjBfy8DnuMs5sT0jw4FEt/PYogJinFdndzywXHzGHEj9c41AAUR
 tB9KYW5vcyBac2FrbyA8enNha29AYmFua25ldC5uZXQ+iQCVAwUQMjkx2XHzGHEj
 9c41AQEuagP/dCIBJP+R16Y70yH75kraRzXY5rnsHmT0Jknrc/ihEEviRYdMV7X1
 osP4pmDU8tNGf0OfGrok7KDTCmygIh7/me+PKrDIj0YkAVUhBX3gBtpSkhEmkLqf
 xbhYwDn4DV3zF7f5AMsbD0UCBDyf+vpkMzgd1Pbr439iXdgwgwta50qJAHUDBRAy
 OSsrO413La462EEBAdIuAv4+Cao1wqBG7+gIm1czIb1M2cAM7Ussx6y+oL1d+HqN
 PRhx4upLVg8Eqm1w4BYpOxdZKkxumIrIvrSxUYv4NBnbwQaa0/NmBou44jqeN+y2
 xwxAEVd9BCUtT+YJ9iMzZlE=
 =w8xL
 -----END PGP PUBLIC KEY BLOCK-----
remarks: This is an example of PGP key certificate
mnt-by:  AS3244-MNT
changed: zsako@banknet.net 19980525
source:  RIPE
```

7. Security Considerations

This document addresses authentication of transactions for making additions, deletions, and updates to the routing policy information through strong cryptographic means. The authorization of these transactions are addressed in [1].

8. Acknowledgements

The present proposal is the result of the discussions within the RIPE DBSEC Task Force, which was set up at RIPE 27 in Dublin at the initiative of Joachim Schmitz and Wilfried Woeber. The list of participants who have contributed to the discussions at different ocasions (TF meetings and via e-mail) is (in alphabetical order): Cengiz Allaettinoglu, Joao Luis Silva Damas, Havard Eidnes, Chris Fletcher, Daniel Karrenberg, David Kessens, Jake Khuon, Craig Labovitz, Carl Malamud, Dave Meyer, Maldwyn Morris, Sandy Murphy, Mike Norris, Carol Orange, Joachim Schmitz, Tom Spindler, Don Stikvoort, Curtis Villamizar, Gerald Winters, Wilfried Woeber, Janos Zsako.

9. References

[1] Meyer, D., Villamizar, C., Alaettinoglu, C. and S. Murphy, "Routing Policy System Security", RFC 2725, December 1999.

10. Author’s Address

Janos Zsako
BankNet
1121 Budapest
Konkoly-Thege ut 29-33.
Hungary

Phone: +36 1 395 90 28
Fax: +36 1 395 90 32
EMail: zsako@banknet.net

11. Notices

PGP is a commercial software.

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

12. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.