CONNOTECH Experts-conseils inc.

A Note About Trust Anchor Key Distribution

Thierry Moreau

Document Number C003444

2005/07/05

Abstract

A new trust anchor key rollover mechanism is proposed. With the initial distribution of a trust anchor configuration, we affix digests of future trust anchor keys. The digest mechanism is based on a secret selection of a hash function within a secure hash function family. The proposed mechanism facilitates long-term integrity protection for trust anchors, and enhanced protection against brute force attacks on public keys. The mechanism effectiveness rests on sound administrative procedures by the central organization behind the trust anchor key.

Table of contents

Introduction

Within the field of public key cryptography, the term "trust anchor key" usually refers to the public signature key of a certification authority that is trusted without any digital certificate by yet another certification authority. Trust anchor keys are deemed to exist because a chain of security certificates can not be of infinite length. This applies to every public key management scheme, the PKI being the foremost example but not the only one. In other cases, a trust anchor may be the public encryption key of a trusted central organization, e.g. in the SAKEM procedure and technology promoted by the author organization ([1]).

The distribution of a trust anchor key refers to its configuration in end-user systems, hopefully with protections against key spoofing threat to the end-user. Once distributed to a given end-user system, a trust anchor key becomes part of the system security configuration, and deserves continued integrity protection.

Trust anchor key distribution is a two-sided issue: initial distribution, and key rollover. Initial distribution is unescapable: a new end-user system needs to be configured, and a new secured application in an existing system can not always rely on pre-existing compatible security configuration elements. The need for trust key rollover stems from two possible causes:

1) Trust anchor private key compromise, i.e. a security incident that casts doubt about the trust anchor private key continued control by the central organization.
2) Trust anchor key cryptoperiod policy, i.e. a central organization decision to periodically renew its trust anchor key public/private key pair.

Note that end-user system security maintenance is not a justification for a trust anchor key rollover recommendation. While such maintenance may involve periodic renewal of local trust in a system configuration, a change in a trust anchor public key is not needed.

The reverse is a different situation: the decision to change a trust anchor key by a central organization forces a configuration operation in end-user systems. This can lead to ambiguities in the effective security provided by a given scheme: the key change may protect the central organization from some threats, but the end-user system integrity protection is dependent on implementation details of the trust anchor key change and might even degrade with some change procedures.

The present proposal is a new trust anchor key rollover mechanism. The suggested acronym for this scheme is TAKREM for Trust Anchor Key REnewal Method.

Central Organization Setup Procedures

Over a period of say 20 years, a central organization is expected to renew trust anchor keys a number of times, say up to 15 times. We use the notation R_i for the public "root" public

key R_i, with the private key counterpart r_i. So the central organization establishes key pairs $<r_0,R_0>$, $<r_1,R_1>$, $<r_2,R_2>$, ..., $<r_n,R_n>$, allocating the pair $<r_0,R_0>$ as the initial private/public trusted key pair, and reserving each key pairs $<r_i,R_i>$ for the cryptoperiod starting with the *i*'th root key renewal, for $1 \le i \le n$. The proposed mechanism applies equally with any public key cryptosystem. The actual use of reserved key pairs may be different from the initial intent, e.g. if security incidents deserve an emergency key renewal or if cryptoperiods are shortened or increased.

Usually, a trust anchor key R_0 is distributed with a self-signed certificate, notation $Cert_0$, with some marginal security benefits. Likewise, self-signed certificate $Cert_i$ might optionally be prepared for reserved trust anchor public key R_i.

Then, the central organization hashes each trust anchor public key R_i with a dedicated hash function, i.e. a separate MASH (Modular Arithmetic Secure Hash) instance H_i is created for each R_i. Two flavors of the MASH are defined in reference [2], and MASH-2 is recommended for highest security. Selecting a dedicated MASH instance requires the central organization to select a large composite modulus number N_i used in the MASH round function and a prime number P_i used in the MASH final reduction function. The parameters N_i and P_i together define the MASH instance, respectively notation N and p in reference [2]. The selection of N_i should take into account the parameter size recommendations for integer factorization cryptosystems. Such recommendations are available from reputable organizations (section 5.6 in [3]) and academia collective work (section 7.2.2.3 in [4]).

In line with the typical use of "salt" or random initialization vectors in data input to cryptographic primitives, the central organization should select a random salt field s_i for each reserved R_i. Thus, the hash computation gives a root key digest D_i with the computation

$$D_i = H_i(s_i|R_i|Cert_i|N_i|P_i) \ .$$

The digest D_i is like an advanced notice of future trust anchor key R_i with certificate $Cert_i$.

The central organization must securely handle the private root key r_i, for $0 \le i \le n$, for to perform the digital signature operation (or the SAKEM incoming message key decapsulation operation). This requires highly secure cryptographic processors ([5]). However, in order to preserve the ultimate security of future trust anchor keys before their respective period fo use, a private key r_i should not even be loaded in any computing device before its period of use. This requires dead storage for r_i, and even for the whole of the data tuple $<r_i,R_i,Cert_i,N_i,P_i,s_i>$.

Specifically, as soon as the key pair $<r_i,R_i>$ and digest D_i are known, the data tuple $<r_i,R_i,Cert_i,N_i,P_i,s_i>$ should be split in two components. For instance, using a one-time-pad cipher, the two components would be the keystream and the data tuple ciphertext. Then, each of the two components are recorded on a digital media which is stored in a tamper-evident packaging. The two packages should be stored in different safe boxes in order to implement effective dual control of key material by the central organization personnel. Since the different key pairs $<r_i,R_i>$ are intended to be put into use at different times, different tamper evident packages should be used. Then the central organization personnel should destroy any copy that might be used in a digital device, for to prevent bypassing the dead storage arrangement.

The digest D_i is not part of the data tuple concealed in the dead storage arrangement. To the contrary, the digests are publicly distributed. In the present proposal, the trust anchor key initial distribution is

$$R_0, Cert_i, D_1, D_2, ..., D_n$$

whereas

$$R_0, Cert_i$$

is a typical prior trust anchor distribution message.

With the storage of data tuple $<r_i,R_i,Cert_i,N_i,P_i,s_i>$ totally concealed until the usage period for key pair $<r_i,R_i>$, an adversary is left with the digest D_i from which it is deemed impossible to mount a brute force attack: the adversary can not even compute the hash function without knowing N_i and P_i. The dead storage arrangement extends this protection to insiders within the central organization.

Trust Anchor Key Configuration Storage in End-User Systems

In end-user systems, the trust anchor key initial distribution message,

$$R_0, Cert_i, D_1, D_2, ..., D_n$$

should be kept indefinitely in secure configuration storage, where this message requires integrity protection.

Trust Anchor Key Rollover Operation

When it is the time for enabling the key pair $<r_i,R_i>$, the central organization personnel retrieves the data tuple components from the dead storage and recovers the data tuple $<r_i,R_i,Cert_i,N_i,P_i,s_i>$. Actually the recovery should be done in the highly secure cryptographic processor where the private key r_i computations are performed. Thus, the cleartext r_i is not available in the clear, even to security personnel.

The data tuple recovery also produces the data needed for a *root key rollover message*, which is specified as

$$i, <R_i, Cert_i, N_i, P_i, s_i>, Cert'_i \ .$$

The presence of a second self-signed certificate $Cert'_i$ in the key rollover message accommodates the possible changes in certificate fields. Formally, $Cert_i$ shall be present if and only if it was used in the initial calculation of digest D_i, and $Cert'_i$ is optional. Any such self-signed security certificate shall be verifiable with the public key R_i. If both $Cert_i$ and $Cert'_i$ are present, the later shall take precedence.

Key Rollover Operation in End-User Systems

Upon receipt of root key rollover message

$$i, <R_i, Cert_i, N_i, P_i, s_i>, Cert'_i \ ,$$

the end-user system becomes in a position to validate the root key digest D_i with the computation

$$H_i(s_i|R_i|Cert_i|N_i|P_i)$$

and test for equality with the D_i value from the trust anchor key initial distribution message. Any self-signed certificates $Cert_i$ and $Cert'_i$ should be verified as well. If the validations are conclusive, the root key R_i should be used, subject to the $Cert'_i$ (or $Cert_i$) contents, if any.

References

[1] See http://www.connotech.com/sakem_index.htm

[2] International standard document ISO/IEC 10118-4:1998, *Information technology - Security techniques - Hash-functions - Part 4: Hash-functions using modular arithmetic*

[3] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid, *Recommendation for Key Management – Part 1: General*, NIST Special Publication 800-57, Draft, April, 2005

[4] *NESSIE security report*, Version 2.0, February 19, 2003

[5] European Committee for Standardization (CEN), *Cryptographic module for CSP signing operations with backup - Protection profile - CMCSOB PP*, CWA 14167-2:2004, May 2004