Güvenli E-Posta İçin Anahtar Yönetim Protokolleri

Key Management Protocols for Secure E-Mail

ÖZET

Günümüzde e-posta, kişilerin en çok kullandıkları iletişim araçlarından birisi olmuştur. E-postaların kullanım yoğunluğu güvenlik sorunlarını ve ihtiyaçlarını da birlikte getirmiştir. E-postaların güvenliği gönderenin kimlik doğrulamasını, gönderenin inkar-edemezliğini ve mesajın bütünlüğü ve gizliliğini gerektirir. Bu güvenlik gereksinimleri genellikle Açık Anahtar Kriptografi tabanlı e-posta uygulamalarıyla sağlanmaktadır. Bu tür e-posta uygulamalarının en önemli sorunlarından birisi de açık-gizli anahtar çiftlerinin yönetim zorluklarıdır. Bu bildiri güvenli e-posta uygulamalarının anahtar yönetiminde kullanmaları için tasarlanmış güvenli yönetim protokollerini içermektedir.

ÖZGEÇMİŞ

Mahmut Özcan
Lisans derecesini Bilkent Üniversitesi Bilgisayar ve Enformatik Müh. Bölümünden 2000 yılında alan Mahmut Özcan, yüksek lisans derecesini Sabancı Üniversitesi Elektronik Müh. ve Bilgisayar Bilimleri bölümünden 2003 yılında almıştır. 2000-2003 yılları arasında TUBİTAK-UEKAE 'de araştırmacı olarak da çalışmıştır ve şu an Sabancı Üniversitesi'nde doktora eğitimine araştırma görevlisi olarak devam etmektedir.

E-posta: mahmut@su.sabanciuniv.edu

Albert Levi
Lisans, yüksek lisans ve doktora derecelerini 1991, 1993 ve 1999 yıllarında Boğaziçi Üniversitesi, Bilgisayar Mühendisliği Bölümünden almıştır. 1999 - 2002 yılları arasında Oregon State Üniversitesi, Elektrik ve Bilgisayar Mühendisliği Bölümünde misafir öğretim üyesi ve aynı bölümün Bilgi Güvenliği Laboratuarında doktora üstü araştırma görevlisi olarak bulunmuştur. Halen Sabancı Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi,Bilgisayar Bilimleri ve Mühendisliği Programında öğretim üyesi olarak görev yapmaktadır.

E-posta: levi@sabanciuniv.edu

ABSTRACT

E-mail is the one of the most popular communication mechanism used between people. This intensive use of e-mails necessitates to deal with the security problems and requirements of e-mails. The fundamental security requirements of e-mails are authentication of origin, integrity and confidentiality of exchanged message, and non-repudiation of sender. They can be generally provided by e-mail applications that use public key cryptosystem as the security base. The most challenging issue of that kind of e-mail applications is managing the public-private key pairs. This paper includes the secure key management protocols and algorithms designed for public key cryptography based e-mail applications.

ANAHTAR YÖNETİM PROTOKOLLERİ

Pretty Good Privacy (PGP) [1,2] e-posta uygulaması ve Secure Multipurpose Internet Mail Extensions (S/MIME) [3] temelli e-posta uygulamaları günümüzde en çok kullanılan ve güvenli kabul edilen e-posta uygulamalardır. PGP'nin anahtar dağıtımı için merkezi ve herkesin güvendiği bir Açık Anahtar Sunucusunun (AAS) olmaması ve S/MIME'ın da sayısal sertifikalara bağımlılığı bu mekanizmaların kullanıldığı e-posta uygulamalarının açık anahtar yönetimlerini zorlaştırmaktadır. Bu yüzden açık anahtar dağıtımından sorumlu ve merkezi bir AAS'nin varlığında pratik ve güvenli bir açık anahtar yönetim protokol yığınına (*stack*) ihtiyaç vardır.

Bu bildiride açık anahtar tabanlı e-posta uygulamalarının kullanıcılarının açık anahtarlarını güvenli ve pratik bir şekilde yönetmelerini sağlayan özgün Anahtar Yönetim Protokolleri (AYP) anlatılmaktadır. AYP'ler saldırılara karşı dayanıklıdır ve de protokollerin güvenliğinin kırılması zordur. AYP protokollerinde hem tek anahtarlı geleneksel kriptografi hem de çift anahtarlı açık anahtar tabanlı kriptografi [4] kullanılmıştır. Sayısal imzalama [**5**], mesaj doğrulama kodları (MAC) [6] ve mesaj özet algoritmaları (SHA-1) [6], AYP'lerde kullanılan mekanizmalardandır. AYP protokolleri kullanıcılar ve AAS arasındaki mesaj alışverişlerini içerir. Her AAS'nin bir Açık Anahtar Deposu (AAD) ve bu depo üzerinde işlem yapan sunucu uygulamaları (SU) vardır. AAD'ler, sistem kullanıcılarının bazı gizli bilgilerini (anne kızlık soyadı gibi) ileride bahsedilecek olan ilklendirme protokolünde, imzasız açık anahtar silme ve güncelleme protokolünde kimlik doğrulama amacıyla kullanmak üzere saklamaktadırlar.Toplam 6 tane AYP algoritması önerilmiştir. Bunlar;

- İlklendirme Protokolü (İlklenPro)
- Açık Anahtar İsteme Protokolü (AnahİstePro)
- Açık Anahtar Silme Protokolü (AnahSilPro)
- Açık Anahtar Güncelleme Protokolü (AnahGünPro)
- İmzasız Açık Anahtar Silme Protokolü (İmzasızAnahSilPro)
- İmzasız Açık Anahtar Güncelleme Protokolü (İmzasızAnahGünPro)

Bu 6 protokol aşağıda detaylandırılmıştır.

1. İLKLENDİRME PROTOKOLÜ (İLKLENPRO)

Bu protokol kullanıcı ilk defa açık anahtarını AAD'ye yükleyeceği zaman kullanılır. Kullanıcı AAS'ye kendisini AAD'de kayıtlı bazı özel bilgilerini (doğum tarihi, ID, anne-kızlık soyadı vs.) vererek tanıtır. Bu protokolün detayları Şekil 1'de verilmiştir.

Protokolün detayları:

1. Kullanıcı (istemci) AAS'ye bağlanır. Kullanıcı ID ve e-posta adresini sunucuya gönderir.
2. AAS kullanıcıyı AAD'deki bilgilere göre doğrular. Kullanıcı doğrulanmışsa AAS kullanıcıya AAD'de kayıtlı kullanıcıya özel gizli bir veya birkaç soru sorar. AAS bu soruları imzalı olarak kullanıcıya gönderir.
3. Kullanıcı cevabını rasgele üretilmiş bir simetrik anahtarla şifreleyerek gönderir. Kullanıcı şifrelemede kullandığı simetrik anahtarı AAS'nin açık anahtarıyla şifreleyerek AAS'ye gönderir.
4. AAS kullanıcının simetrik anahtarını kendisine şifrelenmiş mesajı özel anahtarıyla açmak suretiyle alır. Daha sonra bu simetrik anahtarla kullanıcının cevabını açar. Kullanıcının cevabı doğruysa, AAS kullanıcıya kendisinin rasgele ürettiği MAC (Message Authentication Code – Mesaj Doğrulama Kodu) şifresini kullanıcının gönderdiği simetrik anahtarla şifreleyerek bir e-posta içinde gönderir. Aksi halde protokol biter ve soket kapanır.
5. Kullanıcı e-posta'dan şifreli MAC'ı açar. Bu MAC şifresini kullanarak kendi açık anahtarı için bir doğrulama kodu oluşturur sunucuya açık anahtarı ile beraber bu doğrulama kodunu da gönderir.
6. AAS kullanıcıdan gelen ve kullanıcı açık anahtarının gönderildiği mesajın MAC'ını (doğrulama kodunu) kontrol eder. Eğer mesaj doğrulanırsa kullanıcının açık anahtarı, AAS'ye bağlı AAD'ye kaydedilir. AAS sunucuya teyit mesajı gönderir ve protokol biter.

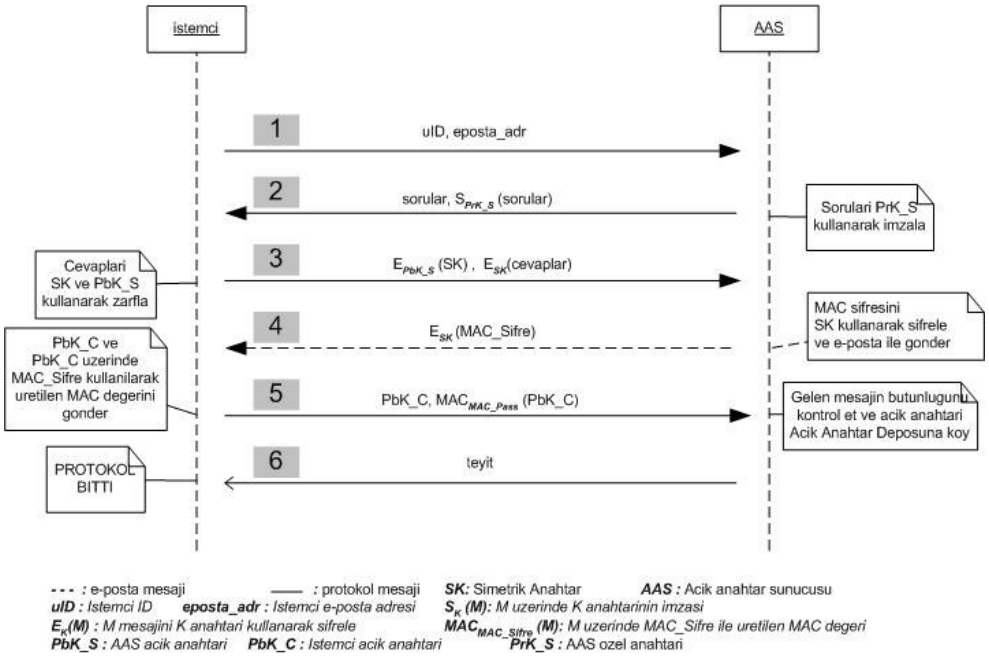
Protokolün Güvenliği

Saldırgan, kullanıcı ve AAS arasındaki hattı gizlice dinlese bile AAD'ye sahte bir açık anahtar yükleyemez. Çünkü, bunu yapmak isteyen saldırganın AAS tarafından kullanıcıya gönderilen MAC şifresini elde etmesi gerekir, bu da kullanıcının kendi inisiyatifiyle tek defalık ürettiği gizli anahtarın da bilinmesini gerektirir. Kullanıcının tek kullanımlık anahtarını elde etmek mümkün olamayacağından AAD'ye sahte açık anahtar yükleme imkansızdır.

Saldırganın başka kullanıcı ID ve e-posta adresi vererek kendisini başka birisi gibi tanıtması durumunda 2 türlü karşı tedbir vardır:

1. Saldırgan, sadece kullanıcı ve AAS arasında paylaşılan gizli bilgileri bilmediği için, saldırı protokolün 2. basamağından itibaren anlaşılacaktır.
2. Saldırganın AAS tarafından kullanıcıya gönderilen MAC şifresini açabilmesi için kullanıcı e-posta hesabına erişebilmesi gerekir. Bunu bilmeyen saldırgan 5. basamaktan itibaren anlaşılacaktır. Çünkü sahte anahtar yükleme yapılırken protokolün 4. basamağındaki AAS tarafından gönderilen e-postanın önce saldırgan tarafından okunması ve gerçek kişi okumadan önce silinmesi gerekir. Saldırganın AAS'de kayıtlı e-posta adresini değiştirmesi mümkün olmadığından, ve ilklendirme protokolü sadece bir defaya mahsus kullanıldığından bu atak mümkün gözükmemektedir.

Taklit etme (impersonation) atağından tamamen kurtulmak AAS ve kullanıcılar arasında önceden paylaşılmış (yarı-gizli bilgi kullanmak yerine) tamamen gizli PIN numarası kullanımıyla mümkün olabilir. Ancak, PIN değerlerinin kullanıcılara çevirim-dışı dağıtımından doğacak sıkıntılar gözden kaçırılmamalıdır. Bu sistemin tasarım kriterlerine ve tercihlerine kalmış bir meseledir.



Şekil 1. İlklendirme Protokolü

2. AÇIK ANAHTAR İSTEME PROTOKOLÜ (ANAHİSTEPRO)

En kolay ve yaygın olarak kullanılacak olan protokoldür. E-posta uygulamasında bir kullanıcı diğer bir kullanıcıya şifreli mesaj atmak istediğinde veya başkasının imzasını doğrulamak istediğinde diğer kullanıcının açık anahtarına ihtiyaç duyar. AAD'den diğer kullanıcının açık anahtarını alabilmek için *AnahİstePro* tasarlanmıştır.Bu protokol kullanıcıların kimliklerini doğrulamaya ihtiyaç duymaz, çünkü herkes bir diğerinin açık anahtarını öğrenebilme hakkına sahiptir. Protokolün detayları Şekil 2'de gösterilmiştir.

Protokolün detayları:

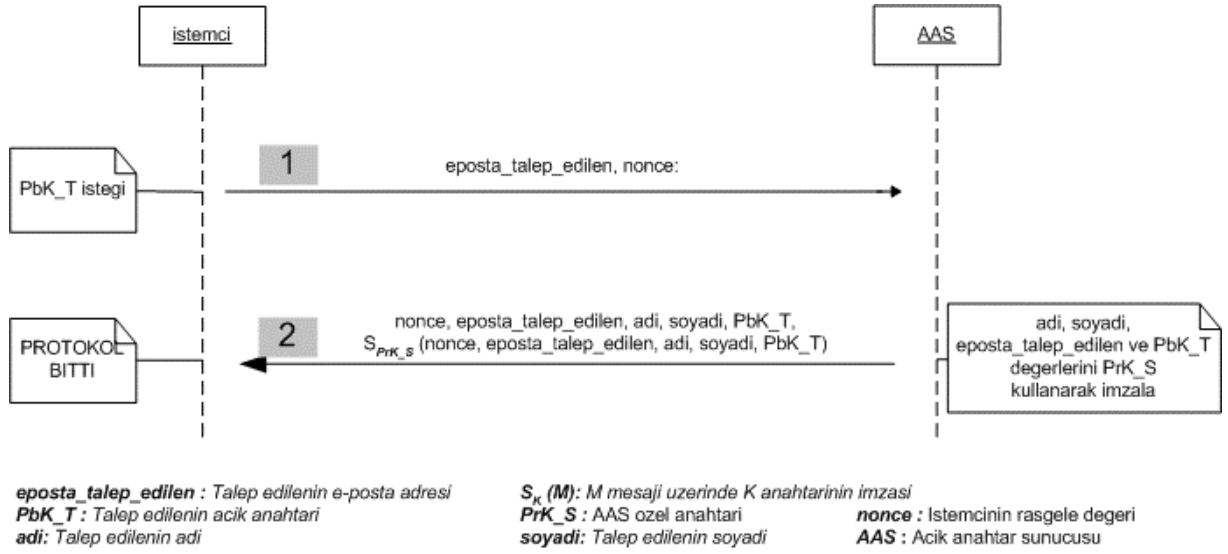
1. Kullanıcı AAS'ye bağlanır ve açık anahtarı talep edilen kişinin e-posta adresini ve rasgele ürettiği *nonce* (tek kullanımlık ve mesajın tekrarlamasını önleyen rasgele sayı) değerini AAS'ye gönderir. AAS, AAD'den talep edilen kişinin açık anahtarını sorgular ve alır. Eğer öyle bir e-posta adresi yoksa protokol biter ve soket kapanır.
2. AAS, talep edilen kişinin açık anahtarını ve talep edenin *nonce* değerini talep edene talep edilenin adı,soyadı ve e-posta adresi bilgileriyle birlikte gönderir. AAS bu bilgileri sayısal olarak imzalayarak talep edene gönderir. Nonce değeri gönderilen mesajların ileride tekrar gönderilmelerini (replay attack) engeller.

Talep edilenin kişisel bilgileri (e-posta adresi, adı ve soyadı), özellikle imzalı e-posta doğrulanırken gönderenin (talep edilen) bilgilerinin doğruluğu konusunda alıcının (talep eden) emin olması için kullanılacaktır.

Protokolün güvenliği

Protokoldeki *nonce* değeri AAS'nin cevabının taze olduğunu temin etmek için kullanılmıştır. Eğer saldırgan AAS'nin gönderdiği önceki bir cevabı kaydetse ve bu cevabın içindeki anahtar silindikten veya güncellendikten sonra kullanıcıların *AnahİstePro* isteklerine cevap olarak eski cevabı gönderse saldırgan geçersiz olan bir açık anahtarı açık anahtar isteyen kullanıcılara yeni bir anahtarmış gibi dağıtmış olur. Nonce kullanımı bu saldırıyı tespit eder ve yanlış *nonce* değeri içeren açık anahtarın kullanıcılarda kullanılmasını engeller.

Açık anahtarlar gizli bilgiler içermediğinden kullanıcıların bu protokolde doğrulanmasına gerek yoktur, ancak AAS'nin kendi kimliğini ve gönderdiği mesajı doğrulaması gerekmektedir, bunu da sayısal imzalama mekanizmasıyla yapar. AAS'nin gizli anahtarı sadece AAS yöneticisinde olduğu için AAS cevaplarında taklidi imza mümkün değildir.



Şekil 2. Açık Anahtar İsteme Protokolü

3. AÇIK ANAHTAR SİLME PROTOKOLÜ (ANAHSİLPRO)

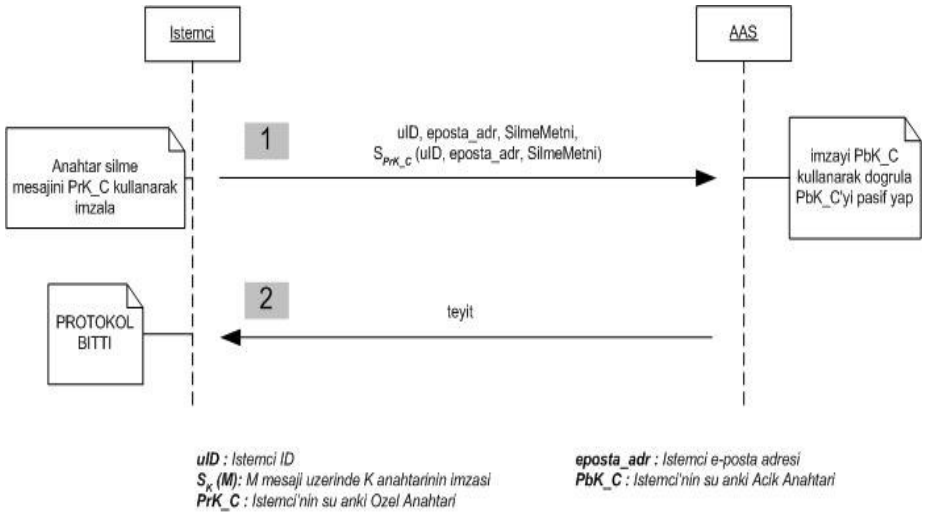
Kullanıcılar açık anahtarlarını herhangi bir nedenden dolayı ya da istek üzerine AAD'den silmek isteyebilir. Kullanıcının aktif açık anahtarını AAD'den silebilmesi için *AnahSilPro* protokolü tasarlanmıştır. Protokolün detayları Şekil 3'de gösterilmiştir.

Protokolün detayları:

1. Kullanıcı AAS'ye bağlanır. Kullanıcı kendine ait Kullanıcı ID'si, e-posta adresi ve anahtarını silmek istediğini belirten metni (*SilmeMetni*) özel anahtarıyla imzalayarak AAS'ye gönderir.
2. AAS kullanıcının gönderdiği bilgiler doğrultusunda imzayı ve bilgileri kontrol eder. Eğer AAS kullanıcı bilgilerini ve imzayı doğrularsa kullanıcının açık anahtarı AAD'de silindi olarak işaretlenir ve teyit mesajı kullanıcıya gönderilerek protokol başarılı bir şekilde tamamlanır. Aksi durumlarda protokol başarısız bir şekilde biter.

Protokolün Güvenliği

SilmeMetni kullanıcı tarafından imzalandığı için saldırganın kullanıcının imzasını (özel anahtarını çalmadan) taklit etmesi mümkün değildir. Eğer kullanıcı özel anahtarını çaldırmışsa ileriki bölümlerde bahsedilecek olan *İmzasızAnahSilPro* protokolünü kullanması önerilmektedir.



Şekil 3. Açık Anahtar Silme Protokolü

4. AÇIK ANAHTAR GÜNCELLEME PROTOKOLÜ (ANAHGÜNPRO)

Kullanıcılar açık anahtarlarını herhangi bir nedenden dolayı ya da istek üzerine AAD'de güncellemek isteyebilir. İleri derecede güvenlik isteyen kullanıcıların açık anahtarlarını periyodik olarak güncellemesi de tavsiye edilen bir davranıştır. Kullanıcının aktif açık anahtarını AAD'de güncelleyebilmesi için *AnahGünPro* protokolü tasarlanmıştır. Protokolün detayları Şekil 4'de gösterilmiştir.

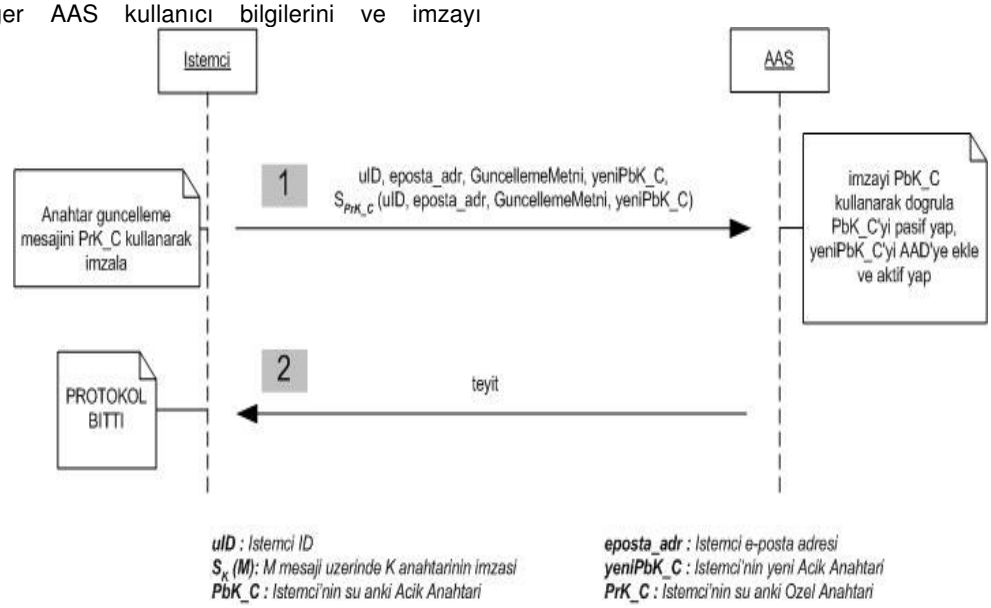
Protokolün detayları:

1. Kullanıcı AAS'ye bağlanır. Kullanıcı kendine ait Kullanıcı ID'si, e-posta adresi, anahtarını güncellemek istediğini belirten metni (*GüncellemeMetni*) ve yeni ürettiği açık anahtarı (*yeniPbK_C*) aktif olan özel anahtarıyla imzalayarak AAS'ye gönderir.
2. AAS kullanıcının gönderdiği bilgiler doğrultusunda imzayı ve bilgileri kontrol eder. Eğer AAS kullanıcı bilgilerini ve imzayı doğrularsa kullanıcının açık anahtarı AAD'de silindi olarak işaretlenir, yeni açık anahtar AAD'ye eklenir ve teyit mesajı kullanıcıya gönderilerek protokol başarılı bir şekilde tamamlanır. Aksi durumlarda protokol başarısız bir şekilde biter.

Protokolün Güvenliği

GüncellemeMetni kullanıcı tarafından imzalandığı için saldırganın kullanıcının imzasını (özel anahtarını çalmadan) taklit etmesi mümkün değildir. Eğer kullanıcı özel anahtarını çaldırmışsa anahtarını güncellemek için ileriki bölümlerde bahsedilecek olan *İmzasızAnahGünPro* protokolünü kullanması önerilmektedir.

Bu protokolde saldırgan 1. basamaktaki mesajı tekrarlama (replay attack) saldırısında bulunabilir, ancak bu herhangi bir soruna yol açmaz, çünkü istek metni kullanıcı tarafından imzalanmaktadır ve tekrarlanmış isteğin imzası anahtar önceden güncellendiği için doğrulanamayacaktır. Bu yüzden *nonce* tabanlı bir protokole burada ihtiyaç yoktur.



Şekil 4. Açık Anahtar Güncelleme Protokolü

5. İMZASIZ AÇIK ANAHTAR SİLME PROTOKOLÜ (İMZASIZANAHSİLPRO)

AnahSilPro'daki nedenlerden dolayı açık anahtarını silmek isteyen kullanıcılar eğer açık anahtarlarının eşleniği olan özel anahtarlarının güvenliğinden şüpheleniyorsa veya bu gizli anahtar kaybedilmişse, bu durumda aktif açık anahtarı silmek için ilklendirme protokolünde olduğu gibi MAC tabanlı bir doğrulamanın yapıldığı *İmzasızAnahSilPro* protokolünü kullanırlar. Protokolün detayları Şekil 5'de gösterilmiştir.

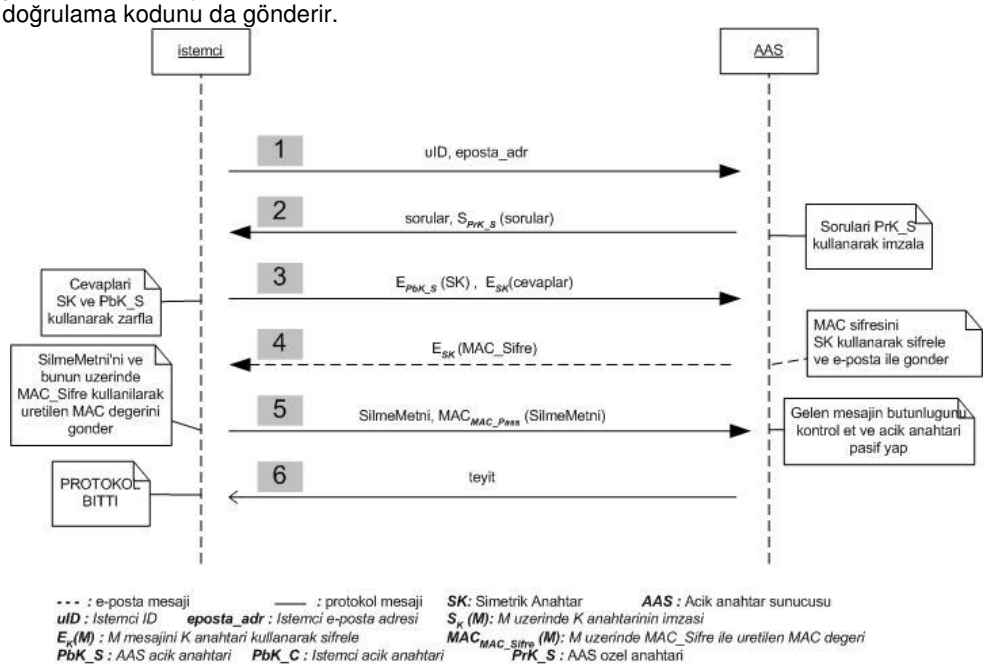
Protokolün detayları:

1. Kullanıcı (istemci) AAS'ye bağlanır. Kullanıcı ID ve e-posta adresini sunucuya gönderir.
2. AAS kullanıcıyı AAD'deki bilgilere göre doğrular. Kullanıcı doğrulanmışsa AAS kullanıcıya AAD'de kayıtlı kullanıcıya özel gizli bir veya birkaç soru sorar. AAS bu soruları imzalı olarak kullanıcıya gönderir.
3. Kullanıcı cevabını rasgele üretilmiş bir simetrik anahtarla şifreleyerek gönderir. Kullanıcı şifrelemede kullandığı simetrik anahtarı AAS'nin açık anahtarıyla şifreleyerek AAS'ye gönderir.

4. AAS kullanıcının simetrik anahtarını kendisine şifrelenmiş mesajı özel anahtarıyla açmak suretiyle alır. Daha sonra bu simetrik anahtarla kullanıcının cevabını açar. Kullanıcının cevabı doğruysa, AAS kullanıcıya kendisinin rasgele ürettiği MAC (Message Authentication Code – Mesaj Doğrulama Kodu) şifresini kullanıcının gönderdiği simetrik anahtarla şifreleyerek bir e-posta içinde gönderir. Aksi halde protokol biter ve soket kapanır.
5. Kullanıcı e-posta'dan şifreli MAC şifresini açar. Bu MAC şifresini kullanarak anahtar silme istek metni (*SilmeMetni*) için bir doğrulama kodu oluşturur ve sunucuya *SilmeMetni* ile beraber bu doğrulama kodunu da gönderir.
6. AAS, kullanıcıdan gelen MAC ile *SilmeMetni*'nin gönderildiği mesajın MAC'ının (doğrulama kodunun) eşit olup olmadığını kontrol eder. Eğer mesaj doğrulanırsa kullanıcının açık anahtarı, AAS'ye bağlı AAD'de silindi olarak işaretlenir. AAS sunucuya teyit mesajı gönderir ve protokol biter.

Protokolün Güvenliği

SilmeMetni kullanıcının MAC şifresiyle AAS tarafından doğrulandığı için, saldırgan (MAC şifresini çalmadan) *SilmeMetni*'nin bütünlüğünü ve doğruluğunu AAS'nin de anlayacağı şekilde sağlayamaz.



Şekil 5. İmzasız Açık Anahtar Silme Protokolü

6. İMZASIZ AÇIK ANAHTAR GÜNCELLEME PROTOKOLÜ (İMZASIZANAHGÜNPRO)

AnahGünPro'daki nedenlerden dolayı açık anahtarını güncellemek isteyen kullanıcılar eğer açık anahtarlarının eşleniği olan özel anahtarlarının güvenliğinden şüpheleniyorsa veya bu gizli anahtar kaybedilmişse, bu durumda anahtarı güncellemek için ilklendirme protokolünde olduğu gibi MAC tabanlı bir doğrulamanın yapıldığı *İmzasızAnahGünPro* protokolünü kullanırlar. Protokolün detayları Şekil 6'de gösterilmiştir.

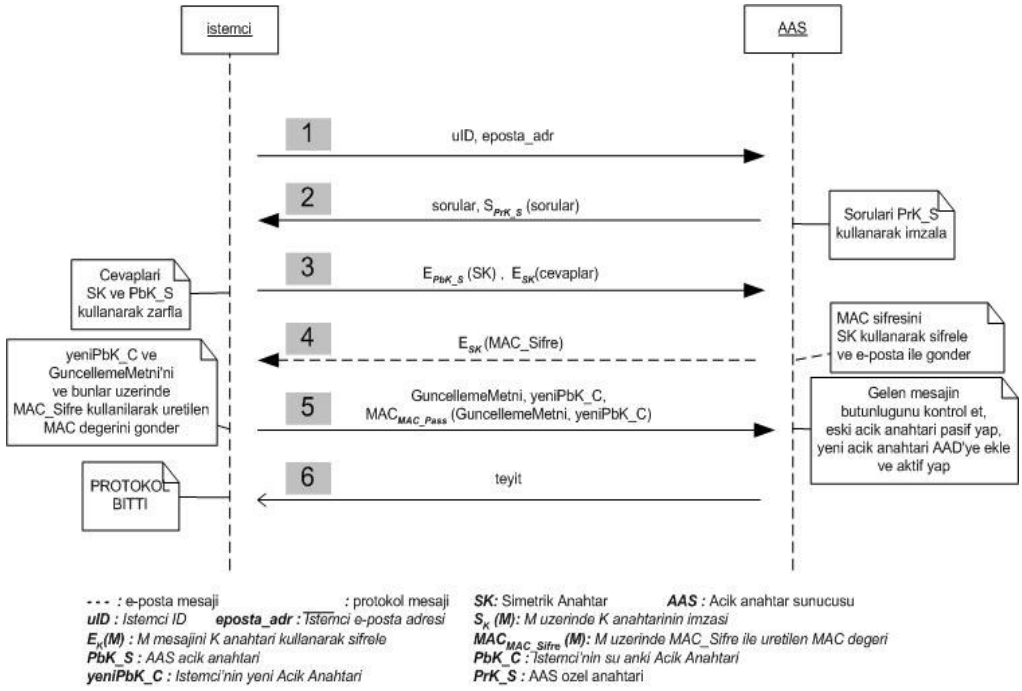
Protokolün detayları:

1. Kullanıcı (istemci) AAS'ye bağlanır. Kullanıcı ID ve e-posta adresini sunucuya gönderir.
2. AAS kullanıcıyı AAD'deki bilgilere göre doğrular. Kullanıcı doğrulanmışsa AAS kullanıcıya AAD'de kayıtlı kullanıcıya özel gizli bir veya birkaç soru sorar. AAS bu soruları imzalı olarak kullanıcıya gönderir.
3. Kullanıcı cevabını rasgele üretilmiş bir simetrik anahtarla şifreleyerek gönderir. Kullanıcı şifrelemede kullandığı simetrik anahtarı AAS'nin açık anahtarıyla şifreleyerek AAS'ye gönderir.
4. AAS kullanıcının simetrik anahtarını kendisine şifrelenmiş mesajı özel anahtarıyla açmak suretiyle alır. Daha sonra bu simetrik anahtarla kullanıcının cevabını açar. Kullanıcının cevabı doğruysa, AAS kullanıcıya kendisinin rasgele ürettiği MAC (Message Authentication Code – Mesaj Doğrulama Kodu) şifresini kullanıcının gönderdiği simetrik anahtarla şifreleyerek bir e-posta içinde gönderir. Aksi halde protokol biter ve soket kapanır.
5. Kullanıcı e-posta'dan şifreli MAC şifresini açar. Bu MAC şifresini kullanarak anahtar güncelleme istek metni (*GüncellemeMetni*) ve yeni ürettiği açık anahtarı (*yeniPbK_C*) için bir doğrulama kodu oluşturur. Sunucuya *GüncellemeMetni* ve *yeniPbK_C* ile beraber bu doğrulama kodunu da gönderir.
6. AAS, kullanıcıdan gelen MAC ile *yeniPbK_C* ve *GüncellemeMetni*'nin gönderildiği mesajın MAC'ının (doğrulama kodunun) eşit olup olmadığını kontrol eder. Eğer mesaj doğrulanırsa kullanıcının aktif açık anahtarı, AAS'ye bağlı AAD'de silindi olarak işaretlenir ve

yeni açık anahtar AAD'ye eklenir. AAS sunucuya teyit mesajı gönderir ve protokol biter.

Protokolün Güvenliği

GüncellemeMetni ve *yeniPbK_C*'nin bütünlüğü ve doğruluğu kullanıcının MAC şifresiyle AAS'ye ispatlandığı için, saldırgan (MAC şifresini çalmadan) *GüncellemeMetni* ve *yeniPbK_C*'nin bütünlüğünü ve doğruluğunu AAS'ye ispat edemez.



Şekil 6. İmzasız Açık Anahtar Güncelleme Protokolü

SONUÇ

Bu bildirideki AYP'leri gerçekleyen e-posta uygulamaları pratik ve güvenli bir anahtar yönetim hizmetini kullanıcılarına sunmuş olurlar. AYP'ler merkezi AAS temelli bir yığın olduğu için, PGP'de olduğu gibi güvenilir bir AAS eksikliğinden kaynaklanan problemlerden de arınmıştır. AYP'leri gerçekleyen e-posta uygulamaları sertifika temelli S/MIME protokolünü kullanan e-posta uygulamalarına göre daha avantajlıdır. Çünkü sertifika temelli protokoller (S/MIME vb.) sertifikaların dağıtık doğasından kaynaklanan yönetim zorluklarından (sertifikaların geçerlik kontrolleri) acı çekmektedir. AYP temelli e-posta uygulamaları ise merkezi bir sunucu üzerinden anahtar yönetimini gerçekleştirdiği için daha pratik ve düzgün çalışmaktadır. AYP'nin *AnahİstePro* protokolünün cevap olarak kullanıcının açık anahtarıyla birlikte ekstra bilgileri (AAS'de kayıtlı kullanıcı adı, soyadı ve e-posta adresi) de göndermesi S/MIME ve PGP'de yapılamayan ekstra güvenlik kontrollerinin yapılmasına imkan tanır. AYP'lerin gerçeklendiği en güzel örnek PractiSES [7,8] sistemidir. PractiSES, AYP anahtar yönetim mekanizmasını kullanan bir açık anahtar kriptografi tabanlı e-posta uygulamasıdır.

KAYNAKÇA

1. *Network Associates, "PGP Freeware for Windows 95, Windows 98,Windows NT, Windows 2000 & Windows Millennium User's Guide Version 7.0", available from http://www.pgpi.org/doc/guide/7.0/en/win/, 2001.*

2. *Elkins, M., "MIME Security with Pretty Good Privacy (PGP)", RFC 2015, 1996.*

3. *S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka, "S/MIME Version 2 Message Specification", RFC 2311, March 1998.*

4. *Diffie, W., and M. E. Hellman,* "New Directions in Cryptography*", IEEE Transactions on Information Theory, vol. IT-22, no. 6, pp. 644-654, November 1976.*

5. *Rivest, R., A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, vol. 21, no. 2, pp. 120-126, February 1978.*

6. *Stallings, W., "Cryptography and Network Security, 3/E", Chapter 11, Prentice Hall, New Jersey, 2003.*

7. *Özcan, M., "Design and Development of Practical and Secure E-mail System", M.Sc. Thesis, Sabanci University, February 2003.*

8. *Levi A., and M.Özcan, "Açık Anahtar Altyapısı Neden Zordur?" Bilişim 2002 – TBD 19. Bilişim Kurultayı, İstanbul, September 2002.*