Güvenli Kabuk (SSH) Aktarım Katmanı Protokolü

Özet

Secure Shell (SSH), güvenli olmayan bir ağ üzerinden güvenli uzaktan oturum açma ve diğer güvenli ağ hizmetleri için bir protokoldür.

Bu belge, tipik olarak TCP/IP üzerinde çalışan SSH taşıma katmanı protokolünü açıklar. Protokol, bir dizi güvenli ağ hizmeti için bir temel olarak kullanılabilir. Güçlü şifreleme, sunucu kimlik doğrulaması ve bütünlük koruması sağlar. Ayrıca sıkıştırma sağlayabilir.

Anahtar değişim yöntemi, ortak anahtar algoritması, simetrik şifreleme algoritması, mesaj doğrulama algoritması ve karma algoritmanın tümü üzerinde anlaşmaya varılır.

Bu belge ayrıca Diffie-Hellman anahtar değişim yöntemini ve SSH aktarım katmanı protokolünü uygulamak için gereken minimum algoritma setini de açıklar.

1. Giriş

SSH aktarım katmanı, güvenli, düşük seviyeli bir aktarım protokolüdür.

Güçlü şifreleme, kriptografik ana bilgisayar kimlik doğrulaması ve bütünlük koruması sağlar.

Bu protokol düzeyindeki kimlik doğrulama ana bilgisayar tabanlıdır; bu protokol, kullanıcı kimlik doğrulaması gerçekleştirmez. Bu protokolün üzerine kullanıcı kimlik doğrulaması için daha yüksek seviyeli bir protokol tasarlanabilir.

Protokol, parametre anlaşmasına izin vermek ve gidiş-dönüş sayısını en aza indirmek için basit ve esnek olacak şekilde tasarlanmıştır.

Anahtar değişim yöntemi, ortak anahtar algoritması, simetrik şifreleme algoritması, mesaj doğrulama algoritması ve karma algoritmanın tümü üzerinde anlaşmaya varılır. Çoğu ortamda, tam anahtar değişimi, sunucu kimlik doğrulaması, hizmet isteği ve hizmet isteğinin kabul bildirimi için yalnızca 2 gidiş-dönüş gerekmesi beklenir. En kötü durum 3 gidiş-dönüştür.

2. Katkıda Bulunanlar

Bu belge setinin asıl katkıda bulunanları şunlar olmuştur:

Tatu Ylonen, Tero Kivinen, Timo J. Rinne, Sami Lehtinen (SSH Communications Security Corp'un tamamı) ve Markku-Juhani O. Saarinen (Jyvaskyla Üniversitesi). Darren Moffat, bu belge setinin orijinal editörüydü ve ayrıca çok önemli katkılarda bulundu.

Yıllar boyunca bu belgenin geliştirilmesine birçok kişi katkıda bulunmuştur. Kabul edilmesi gereken kişiler arasında Mats Andersson, Ben Harris, Bill Sommerfeld, Brent McClure, Niels Moller, Damien Miller,

Derek Fawcus, Frank Cusack, Heikki Nousiainen, Jakob Schlyter, Jeff Van Dyke, Jeffrey Altman, Jeffrey Hutzelman, Jon Bright, Joseph Galbraith, Ken Hornstein, Markus Friedl, Martin Forssen, Nicolas Williams, Niels Provos, Perry Metzger, Peter Gutmann, Simon Josefsson, Simon Tatham, Wei Dai, Denis Bider, der Mouse ve

Tadayoshi Kohno. İsimlerinin burada belirtilmesi, bu belgeyi desteklediklerini değil, katkıda bulundukları anlamına gelir.

3. Bu Belgede Kullanılan Kurallar

SSH protokolleri ile ilgili tüm belgelerde "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", ve "OPTIONAL" anahtar kelimeleri kullanılacaktır, ve gereksinimleri açıklamak için "İSTEĞE BAĞLI". Bu anahtar sözcükler [RFC2119]'da açıklandığı gibi yorumlanmalıdır.

Bu belgede "PRIVATE USE", "HIERARCHICAL ALLOCATION", "FIRST COME FIRST SERVED", "EXPERT REVIEW", "SPECIFICATION REQUIRED", "IESG APPROVAL", "IETF CONSENSUS", ve "STANDARDS ACTION" anahtar sözcükleri ad alanı tahsisini tanımlamak için kullanılanlar [RFC2434]'te açıklandığı gibi yorumlanmalıdır.

Protokol alanları ve bunları doldurmak için olası değerler bu belge setinde tanımlanmıştır. Protokol alanları mesaj tanımlarında tanımlanacaktır. Örnek olarak SSH_MSG_CHANNEL_DATA aşağıdaki gibi tanımlanmıştır.

```
byte      SSH_MSG_CHANNEL_DATA
uint32    recipient channel
string    data
```

Bu belgelerde alanlara atıfta bulunulduğunda, tek tırnak içinde görünürler. Bu alanları dolduracak değerlere başvurulduğunda, bunlar çift tırnak içinde görünecektir. Yukarıdaki örneği kullanarak, 'veri' için olası değerler "foo" ve "bar" dır.

4. Bağlantı Kurulumu

SSH, herhangi bir 8 bitlik temiz, ikili şeffaf aktarım üzerinde çalışır. Temel aktarım, SSH bağlantısının sonlandırılmasına neden olduğundan, aktarım hatalarına karşı koruma sağlamalıdır.

İstemci bağlantıyı başlatır.

4.1. TCP/IP üzerinden kullanın

TCP/IP üzerinden kullanıldığında, sunucu normalde 22 numaralı bağlantı noktasındaki bağlantıları dinler. Bu bağlantı noktası numarası IANA'ya kaydedilmiştir ve resmi olarak SSH için atanmıştır.

4.2. Protokol Sürümü Değişimi

Bağlantı kurulduğunda, her iki taraf da bir tanımlama dizisi göndermelidir. Bu tanımlama dizisi olmalıdır

SSH-protoversion-softwareversion SP comments CR LF

Bu belge setinde tanımlanan protokol sürüm 2.0 olduğundan, 'protoversion' "2.0" olmalıdır. 'Yorumlar' dizesi isteğe bağlıdır. 'Yorumlar' dizesi dahil edilmişse, bir 'boşluk' karakteri (yukarıda SP, ASCII 32 olarak belirtilmiştir) 'yazılım sürümü' ve 'yorum' dizelerini ayırmalıdır. Tanımlama, tek bir satır başı (CR) ve tek bir Satır Besleme (LF) karakteri (sırasıyla ASCII 13 ve 10) ile sonlandırılmalıdır. Bu protokolün daha eski, belgelenmemiş sürümleriyle uyumluluğu sürdürmek isteyen uygulayıcılar, bu belgenin 5. Bölümünde açıklanan nedenlerle, satır başı karakterinin varlığını beklemeden tanımlama dizesini işlemek isteyebilir. Boş karakter gönderilmemelidir.

Dizenin maksimum uzunluğu, satır başı ve satır besleme dahil 255 karakterdir.

Tanımlama dizesinin Satır Başı ve Satır Besleme'den önceki kısmı Diffie-Hellman anahtar değişiminde kullanılır (bkz. Bölüm 8).

Sunucu, sürüm dizesini göndermeden önce diğer veri satırlarını gönderebilir. her satır, bir Satır Başı ve Satır Besleme ile sonlandırılmalıdır. Bu tür satırlar "SSH-" ile başlamamalı ve ISO-10646 UTF-8 [RFC3629] (dil belirtilmemiş) ile kodlanmalıdır. Müşteriler bu tür hatları işleyebilmelidir. Bu tür satırlar sessizce yok sayılabilir veya istemci kullanıcıya gösterilebilir. Görüntüleniyorlarsa, [SSH-ARCH] bölümünde tartışıldığı gibi kontrol karakteri filtrelemesi kullanılmalıdır. Bu özelliğin birincil kullanımı, TCP sarmalayıcıların bağlantıyı kesmeden önce bir hata mesajı görüntülemesine izin vermektir.

Hem 'protoversion' hem de 'softwareversion' dizeleri, boşluk karakterleri ve eksi işareti (-) dışında yazdırılabilir US-ASCII karakterlerinden oluşmalıdır. 'Yazılım sürümü' dizesi öncelikle uyumluluk uzantılarını tetiklemek ve bir uygulamanın yeteneklerini belirtmek için kullanılır. 'Yorumlar' dizisi, kullanıcı sorunlarının çözümünde faydalı olabilecek ek bilgiler içermelidir. Bu nedenle, geçerli bir tanımlama dizesi örneği,

SSH-2.0-billsSSH_3.6.3q3<CR><LF>

şeklindedir.

Bu tanımlama dizisi, isteğe bağlı "yorumlar" dizisini içermez ve bu nedenle, "yazılım sürümü" dizisinden hemen sonra bir CR ve LF ile sonlandırılır.

Bu tanımlayıcı gönderildikten hemen sonra anahtar değişimi başlayacaktır.

Tanımlama dizisini izleyen tüm paketler, Bölüm 6'da açıklanan ikili paket protokolünü kullanmalıdır.

5. Eski SSH Sürümleriyle Uyumluluk

Daha önce belirtildiği gibi, bu protokol için belirtilen 'protoversion' "2.0"dır. Bu protokolün önceki sürümleri resmi olarak belgelenmemiştir, ancak "1.x"in "protoversion"unu (örneğin, "1.5" veya "1.3") kullandıkları yaygın olarak bilinmektedir. Bu yazının yazıldığı sırada, SSH'nin birçok uygulaması protokol sürüm 2.0'ı kullanıyor, ancak hala önceki sürümleri kullanan cihazların olduğu biliniyor. Geçiş döneminde, kurulu SSH istemcileri ve protokolün eski sürümünü kullanan sunucularla uyumlu bir şekilde çalışabilmek önemlidir. Bu bölümdeki bilgiler yalnızca uyumluluğu destekleyen uygulamalarla ilgilidir

SSH sürümleri 1.x ile. İlgilenenler için, 1.x protokolünün bilinen tek belgeleri, [ssh-1.2.30] kaynak koduyla birlikte gönderilen BENİOKU dosyalarında bulunur.

5.1. Eski İstemci, Yeni Sunucu

Sunucu uygulamaları yapılandırılabilir bir uyumluluk bayrağını destekleyebilir

bu, eski sürümlerle uyumluluk sağlar. Bu bayrak açıkken, sunucu 'proto sürümünü' "1.99" olarak tanımlamalıdır. Müşteriler protokol 2.0'ı kullanmak, bunu "2.0" ile aynı olarak tanımlayabilmelidir. Bu modda, sunucu kimlik dizesinden sonra satır başı karakterini (ASCII 13) göndermemelidir.

Uyumluluk modunda, sunucu kimlik dizesini gönderdikten sonra istemciden bir kimlik dizesi alana kadar başka veri göndermemelidir. Sunucu daha sonra istemcinin eski bir protokol kullanıp kullanmadığını belirleyebilir ve gerekirse eski protokole geri dönebilir. Uyumluluk modunda, sunucu kimlik dizesinden önce ek veri göndermemelidir.

Eski istemcilerle uyumluluk gerekmediğinde, sunucu kimlik dizesinden hemen sonra ilk anahtar değişimi verilerini gönderebilir.

5.2. Yeni İstemci, Eski Sunucu

Yeni istemci, tanımlama dizesinden sonra (sunucunun tanımlama dizesini almadan önce) hemen ek veri gönderebileceğinden, istemci sunucunun eski olduğunu öğrendiğinde eski protokol zaten bozulmuş olabilir. Bu olduğunda, müşteri

sunucuyla olan bağlantıyı kapatın ve eski protokolü kullanarak yeniden bağlanın.

5.3. Paket Boyutu ve Ek Yükü

Bazı okuyucular, yeni başlıklar, dolgular ve İleti Kimlik Doğrulama Kodu (MAC) nedeniyle paket boyutundaki artıştan endişe duyacaktır. Minimum paket boyutu 28 bayt mertebesindedir (anlaşmalı algoritmalara bağlı olarak). Artış, büyük paketler için ihmal edilebilir, ancak bir baytlık paketler (telnet tipi oturumlar) için çok önemlidir. Bununla birlikte, bunu neredeyse tüm durumlarda sorun olmaktan çıkaran birkaç faktör vardır:

- Bir TCP/IP başlığının minimum boyutu 32 bayttır. Bu nedenle, artış aslında 33'ten 51 bayta (kabaca).
- Bir ethernet paketinin veri alanının minimum boyutu 46bayttır [RFC0894]. Bu nedenle, artış 5 bayttan fazla değildir. Ethernet başlıklarına bakıldığında ise artış yüzde 10'un altında kalıyor.
- İnternetteki telnet tipi verilerin toplam oranı, artan paket boyutlarında bile ihmal edilebilir düzeydedir.

Paket boyutu artışının önemli bir etkiye sahip olabileceği tek ortam, yavaş modem hatları üzerindeki PPP'dir [RFC1661] (PPP, paket boyutundaki artışı vurgulayarak TCP/IP başlıklarını sıkıştırır). Bununla birlikte, modern modemlerde, aktarım için gereken süre, insanların yazabileceğinden çok daha hızlı olan 2 milisaniye mertebesindedir.

Maksimum paket boyutuyla ilgili sorunlar da vardır. Ekran güncellemelerindeki gecikmeleri en aza indirmek için etkileşimli oturumlar için aşırı büyük paketler istenmez. Maksimum paket boyutu, her kanal için ayrı ayrı görüşülür.

6. İkili Paket Protokolü

Her paket aşağıdaki biçimdedir:

```
uint32    packet_length
byte      padding_length
byte[n1]  payload; n1 = packet_length - padding_length - 1
byte[n2]  random padding; n2 = padding_length
byte[m]   mac (Message Authentication Code - MAC); m = mac_length
```

packet_length
'mac' veya 'packet_length' alanının kendisi hariç, bayt cinsinden paketin uzunluğu.

padding_length
'Rastgele doldurma' uzunluğu (bayt)

Payload
Paketin faydalı içeriği. Sıkıştırma konusunda anlaşma sağlandıysa, bu alan sıkıştırılır. Başlangıçta, sıkıştırma "hiç" olmalıdır.

random padding
Toplam uzunluğu (paket_uzunluğu || dolgu_uzunluğu || yük || rastgele doldurma), hangisi daha büyükse, şifre bloğu boyutunun veya 8'in bir katı olacak şekilde keyfi uzunlukta dolgu. En az dört bayt dolgu olmalıdır. Dolgu rastgele baytlardan oluşmalıdır. Maksimum dolgu miktarı 255 bayttır.

mac
Mesaj Doğrulama Kodu. Mesaj kimlik doğrulaması yapıldıysa, bu alan MAC baytlarını içerir. Başlangıçta, MAC algoritması "none" olmalıdır.

'packet_length', 'padding_length', 'payload' ve 'random padding' birleştirme uzunluğunun, hangisi daha büyükse, şifre bloğu boyutunun bir katı veya 8 olması gerektiğini unutmayın. Bu kısıtlama, akış şifreleri kullanılırken bile uygulanmalıdır. 'packet_length' alanının da şifreli olduğunu ve paket gönderirken veya alırken bu alanın işlenmesinin özel dikkat gerektirdiğini unutmayın. Ayrıca, değişken miktarlarda 'rastgele dolgu' eklenmesinin trafik analizini engellemeye yardımcı olabileceğini unutmayın.

Bir paketin minimum boyutu 16 (veya şifre bloğu boyutu, hangisi daha büyükse) bayttır (artı 'mac'). Uygulamalar, bir paketin ilk 8 (veya şifre bloğu boyutu, hangisi daha büyükse) baytını aldıktan sonra uzunluğun şifresini çözmelidir.

6.1. Maksimum Paket Uzunluğu

Tüm uygulamalar, sıkıştırılmamış yük uzunluğu 32768 bayt veya daha az ve toplam paket boyutu 35000 bayt veya daha az olan paketleri işleyebilmelidir ZORUNLU 'packet_length', 'padding_length', 'payload', 'random padding' ve 'mac '). Maksimum 35000 bayt, yukarıda belirtilen sıkıştırılmamış uzunluktan daha büyük, isteğe bağlı olarak seçilen bir değerdir. Uygulama, ihtiyaç duyulabilecekleri yerlerde daha uzun paketleri desteklemelidir. Örneğin, bir uygulama çok sayıda sertifika göndermek istiyorsa, tanımlama dizesi diğer tarafın bunları işleyebildiğini gösteriyorsa daha büyük paketler gönderilebilir. Ancak uygulamalar, uygulamanın hizmet reddi ve/veya arabellek taşması saldırılarından kaçınması için paket uzunluğunun makul olup olmadığını kontrol etmelidir.

6.2. Sıkıştırma

Sıkıştırma üzerinde anlaşmaya varılmışsa, 'payload' alanı (ve yalnızca o) anlaşmalı algoritma kullanılarak sıkıştırılacaktır. 'packet_length' alanı ve 'mac' sıkıştırılmış yükten hesaplanacaktır. Sıkıştırma işleminden sonra şifreleme yapılacaktır.

Sıkıştırma, yönteme bağlı olarak durumsal olabilir. Sıkıştırma her yön için bağımsız olmalıdır ve uygulama her yön için bağımsız algoritma seçimine izin vermelidir. Ancak pratikte sıkıştırma yönteminin her iki yönde de aynı olması önerilir.

Şu anda aşağıdaki sıkıştırma yöntemleri tanımlanmıştır:

none	REQUIRED	no compression
zlib	OPTIONAL	ZLIB (LZ77) compression

"zlib" sıkıştırması [RFC1950] ve [RFC1951]'de açıklanmıştır. Sıkıştırma bağlamı, her anahtar değişiminden sonra başlatılır ve her paketin sonunda yalnızca kısmi bir boşaltma gerçekleştirilerek bir paketten diğerine geçirilir. Kısmi bir temizleme, mevcut sıkıştırılmış bloğun sona erdiği ve tüm verilerin çıktılanacağı anlamına gelir. Mevcut blok depolanmış bir blok değilse, mevcut bloğun blok sonu kodunun başlangıcından sonuna kadar sayılarak en az 8 bit olduğundan emin olmak için mevcut bloktan sonra bir veya daha fazla boş blok eklenir. paket yükünden.

[SSH-ARCH] ve [SSH-NUMBERS]'da belirtildiği gibi ek yöntemler tanımlanabilir.

6.3. Şifreleme

Anahtar değişimi sırasında bir şifreleme algoritması ve bir anahtar üzerinde anlaşmaya varılacaktır. Şifreleme etkin olduğunda, her paketin paket uzunluğu, doldurma uzunluğu, yük ve doldurma alanları verilen algoritma ile şifrelenmelidir.

Bir yönde gönderilen tüm paketlerdeki şifreli veriler, tek bir veri akışı olarak düşünülmelidir. Örneğin, başlatma vektörleri bir paketin sonundan sonraki paketin başına geçirilmelidir. Tüm şifreler, etkin anahtar uzunluğu 128 bit veya daha fazla olan anahtarlar kullanmalıdır.

Her yöndeki şifreler birbirinden bağımsız olarak çalışmalıdır. Yerel ilke tarafından birden çok algoritmaya izin veriliyorsa, uygulamalar her yön için algoritmanın bağımsız olarak seçilmesine izin vermelidir. Ancak pratikte her iki yönde de aynı algoritmanın kullanılması tavsiye edilir.

Şu anda aşağıdaki şifreler tanımlanmıştır:

```
3des-cbc        REQUIRED        three-key 3DES in CBC mode
    blowfish-cbc    OPTIONAL        Blowfish in CBC mode
    twofish256-cbc  OPTIONAL        Twofish in CBC mode,
                                    with a 256-bit key
    twofish-cbc     OPTIONAL        alias for "twofish256-cbc"
                                    (this is being retained
                                    for historical reasons)
    twofish192-cbc  OPTIONAL        Twofish with a 192-bit key
    twofish128-cbc  OPTIONAL        Twofish with a 128-bit key
    aes256-cbc      OPTIONAL        AES in CBC mode,
                                    with a 256-bit key
    aes192-cbc      OPTIONAL        AES with a 192-bit key
    aes128-cbc      RECOMMENDED     AES with a 128-bit key
    serpent256-cbc  OPTIONAL        Serpent in CBC mode, with
                                    a 256-bit key
    serpent192-cbc  OPTIONAL        Serpent with a 192-bit key
    serpent128-cbc  OPTIONAL        Serpent with a 128-bit key
    arcfour         OPTIONAL        the ARCFOUR stream cipher
                                    with a 128-bit key
    idea-cbc        OPTIONAL        IDEA in CBC mode
    cast128-cbc     OPTIONAL        CAST-128 in CBC mode
    none            OPTIONAL        no encryption; NOT RECOMMENDED
```

"3des-cbc" şifresi, anahtarın ilk 8 baytının ilk şifreleme için, sonraki 8 baytın şifre çözme için ve sonraki 8 baytın kullanıldığı üç anahtarlı üçlü DES'tir (şifrele-çöz-çöz-şifrele). son şifreleme için. Bu, 24 bayt anahtar verisi gerektirir (bunların 168 biti gerçekte kullanılır). CBC modunu uygulamak için dış zincirleme kullanılmalıdır (yani, yalnızca bir başlatma vektörü vardır). Bu, 8 baytlık bloklardan oluşan bir blok şifredir. Bu algoritma [FIPS-46-3]'te tanımlanmıştır. Bu algoritmanın etkin anahtar uzunluğu yalnızca 112 bit ([SCHNEIER]) olduğundan, SSH şifreleme algoritmalarının 128 bit veya daha fazla anahtar kullanması gereken özellikleri karşılamadığını unutmayın. Ancak, bu algoritma tarihsel nedenlerle hala gereklidir; esasen, bu yazının yazıldığı sırada bilinen tüm uygulamalar bu algoritmayı destekler ve temel birlikte çalışabilir algoritma olduğu için yaygın olarak kullanılır. Gelecekte, daha güçlü olan başka bir algoritmanın, "3des-cbc" kullanımının başka bir standart eylem tarafından kullanımdan kaldırılacağı kadar yaygın ve yaygın hale gelmesi bekleniyor.

"Blowfish-cbc" şifresi, CBC modunda 128 bit anahtarlı [SCHNEIER] Blowfish'tir. Bu, 8 baytlık bloklardan oluşan bir blok şifredir.

"twofish-cbc" veya "twofish256-cbc" şifresi, CBC modunda Twofish'tir,
[TWOFISH] açıklandığı gibi 256-bit anahtarlarla. Bu, 16 baytlık bloklardan oluşan bir blok şifredir.

"twofish192-cbc" şifresi yukarıdakiyle aynıdır, ancak 192 bitlik bir anahtara sahiptir.

"twofish128-cbc" şifresi yukarıdakiyle aynıdır, ancak 128 bitlik bir anahtara sahiptir.

"aes256-cbc" şifresi, CBC modunda AES (Gelişmiş Şifreleme Standardı) [FIPS-197]'dir. Bu sürüm 256 bit anahtar kullanır.

"aes192-cbc" şifresi yukarıdakiyle aynıdır, ancak 192 bitlik bir anahtara sahiptir.

"aes128-cbc" şifresi yukarıdakiyle aynıdır, ancak 128 bitlik bir anahtara sahiptir.

Serpent AES gönderiminde açıklandığı gibi 256 bitlik bir anahtarla CBC modunda "serpent256-cbc" şifresi.

"serpent192-cbc" şifresi yukarıdakiyle aynıdır, ancak 192-bit
anahtar.

"serpent128-cbc" şifresi yukarıdakiyle aynıdır, ancak 128-bit
anahtar.

"arcfour" şifresi, 128 bitlik anahtarlara sahip Arcfour akış şifresidir. Arcfour şifresinin RC4 şifresi [SCHNEIER] ile uyumlu olduğuna inanılmaktadır. Arcfour (ve RC4) zayıf anahtarlarla ilgili problemlere sahiptir ve dikkatli kullanılmalıdır.

"idea-cbc" şifresi, CBC modunda [SCHNEIER] IDEA şifresidir.

"cast128-cbc" şifresi, CBC modunda 128 bit anahtarlı [RFC2144] CAST-128 şifresidir.

"Hiçbiri" algoritması, şifreleme yapılmayacağını belirtir. Bu yöntemin hiçbir gizlilik koruması sağlamadığını ve önerilmediğini unutmayın. Bu şifre seçilirse bazı işlevler (örneğin, parola doğrulama) güvenlik nedeniyle devre dışı bırakılabilir.

[SSH-ARCH] ve [SSH-NUMBERS]'de belirtildiği gibi ek yöntemler tanımlanabilir.

6.4. Veri bütünlüğü

Veri bütünlüğü, her pakete, paylaşılan bir sırdan hesaplanan bir MAC, paket sıra numarası ve paketin içeriği dahil edilerek korunur.

Mesaj doğrulama algoritması ve anahtar, anahtar değişimi sırasında görüşülür. Başlangıçta hiçbir MAC etkin olmayacaktır ve uzunluğu sıfır olmalıdır. Anahtar değişiminden sonra, seçilen MAC algoritması için 'mac', paket verilerinin birleştirilmesinden şifrelemeden önce hesaplanacaktır:

mac = MAC(key, sequence_number || unencrypted_packet)

burada şifrelenmemiş_paket, 'mac' içermeyen paketin tamamıdır (uzunluk alanları, 'yük' ve 'rastgele doldurma') ve sıra_sayısı, uint32 olarak temsil edilen örtük bir paket sıra numarasıdır. Sıra_numarası ilk paket için sıfır olarak başlatılır ve her paketten sonra artırılır (şifreleme veya MAC kullanımda olup olmadığına bakılmaksızın).

Anahtarlar/algoritmalar daha sonra yeniden görüşülse bile asla sıfırlanmaz. Her 2^32 paketten sonra sıfıra sarılır. Paket sıra numarası, tel üzerinden gönderilen pakete dahil değildir.

Her yön için MAC algoritmaları bağımsız olarak çalışmalı ve uygulamalar, algoritmanın her iki yön için bağımsız olarak seçilmesine izin vermelidir. Ancak pratikte her iki yönde de aynı algoritmanın kullanılması tavsiye edilir.

MAC algoritmasından elde edilen 'mac' değeri, paketin son kısmı olarak şifrelenmeden iletilmelidir. 'mac' bayt sayısı, seçilen algoritmaya bağlıdır.

Şu anda aşağıdaki MAC algoritmaları tanımlanmıştır:

```
hmac-sha1    REQUIRED       HMAC-SHA1 (digest length = key
                            length = 20)
hmac-sha1-96 RECOMMENDED    first 96 bits of HMAC-SHA1 (digest
                            length = 12, key length = 20)
hmac-md5     OPTIONAL       HMAC-MD5 (digest length = key
                            length = 16)
hmac-md5-96  OPTIONAL       first 96 bits of HMAC-MD5 (digest
                            length = 12, key length = 16)
none         OPTIONAL       no MAC; NOT RECOMMENDED
```

"hmac-*" algoritmaları [RFC2104]'te açıklanmıştır. "*-n" MAC'leri, elde edilen değerin yalnızca ilk n bitini kullanır.

SHA-1, [FIPS-180-2]'de anlatılmıştır ve MD5, [RFC1321]'de anlatılmıştır.

[SSH-ARCH] ve [SSH-NUMBERS]'de belirtildiği gibi ek yöntemler tanımlanabilir.

6.5. Anahtar Değişim Yöntemleri

Anahtar değişimi yöntemi, şifreleme ve kimlik doğrulama için tek seferlik oturum anahtarlarının nasıl oluşturulacağını ve sunucu kimlik doğrulamasının nasıl yapıldığını belirtir.

İki gerekli anahtar değişim yöntemi tanımlanmıştır:

```
diffie-hellman-group1-sha1 REQUIRED
diffie-hellman-group14-sha1 REQUIRED
```

Bu yöntemler Bölüm 8'de açıklanmıştır.

[SSH-NUMARALARI]'nda belirtildiği gibi ek yöntemler tanımlanabilir. "diffie-hellman-group1-sha1" adı, [RFC2409]'da tanımlandığı gibi bir Oakley grubu kullanan bir anahtar değişim yöntemi için kullanılır. SSH, Oakley [RFC2412] ve IKE'den mantıksal olarak farklı olan kendi grup tanımlayıcı alanını korur; ancak, ek bir grup için, Çalışma Grubu ikinci tanımlanan grubun adı için diffie-hellman-group14-sha1'i kullanarak [RFC3526] tarafından atanan numarayı kabul etti.

Uygulamalar bu adları opak tanımlayıcılar olarak ele almalı ve
SSH tarafından kullanılan gruplar ile IKE için tanımlanan gruplar arasında herhangi bir ilişki varsaymamalıdır.

6.6. Açık Anahtar Algoritmaları

Bu protokol, hemen hemen tüm ortak anahtar biçimleri, kodlamaları ve algoritmaları (imza ve/veya şifreleme) ile çalışacak şekilde tasarlanmıştır.

Bir ortak anahtar türünü tanımlayan birkaç yön vardır:

- Anahtar biçimi: Anahtarın nasıl kodlandığı ve sertifikaların nasıl temsil edildiği. Bu protokoldeki anahtar blobları, anahtarlara ek olarak sertifikalar içerebilir.

- İmza ve/veya şifreleme algoritmaları. Bazı anahtar türleri hem imzalamayı hem de şifrelemeyi desteklemeyebilir. Anahtar kullanımı, politika ifadeleriyle de kısıtlanabilir (örneğin, sertifikalarda). Bu durumda, farklı politika alternatifleri için farklı anahtar türleri tanımlanmalıdır.

- İmzaların ve/veya şifrelenmiş verilerin kodlanması. Buna dolgu, bayt sırası ve veri biçimleri dahildir ancak bunlarla sınırlı değildir.

Aşağıdaki ortak anahtar ve/veya sertifika biçimleri şu anda tanımlanmıştır:

ssh-dss	REQUIRED	sign	Raw DSS Key
ssh-rsa	RECOMMENDED	sign	Raw RSA Key
pgp-sign-rsa	OPTIONAL	sign	OpenPGP certificates (RSA key)
pgp-sign-dss	OPTIONAL	sign	OpenPGP certificates (DSS key)

[SSH-ARCH] ve [SSH-NUMBERS]'de belirtildiği gibi ek anahtar türleri tanımlanabilir.

Anahtar türü her zaman açıkça bilinmelidir (algoritma anlaşmasından veya başka bir kaynaktan). Normalde anahtar bloğuna dahil edilmez.

Sertifikalar ve genel anahtarlar aşağıdaki gibi kodlanmıştır:

```
string    certificate or public key format identifier
byte[n]   key/certificate data
```

Sertifika bölümü, sıfır uzunlukta bir dize olabilir, ancak bir genel anahtar gereklidir. Bu, kimlik doğrulama için kullanılacak ortak anahtardır. Sertifika dizisi sertifika blobu, yetkilendirme sağlamak için kullanılabilir.

Açıkça bir imza biçimi tanımlayıcısı belirtmeyen ortak anahtar/sertifika biçimleri, imza tanımlayıcısı olarak ortak anahtar/sertifika biçim tanımlayıcısını kullanmalıdır.

İmzalar şu şekilde kodlanmıştır:

string imza biçimi tanımlayıcısı (ortak anahtar/sertifika biçimi

tarafından belirtildiği gibi)

byte[n] biçime özgü kodlamada imza blobu.

"ssh-dss" anahtar biçimi aşağıdaki özel kodlamaya sahiptir:

```
string   "ssh-dss"
mpint    p
mpint    q
mpint    g
mpint    y
```

Burada 'p', 'q', 'g' ve 'y' parametreleri imza anahtarı bloğunu oluşturur.

Bu anahtar biçimini kullanarak imzalama ve doğrulama, SHA-1 karma [FIPS-180-2] kullanılarak Dijital İmza Standardına [FIPS-186-2] göre yapılır.

Ortaya çıkan imza aşağıdaki gibi kodlanmıştır:

```
string   "ssh-dss"
string   dss_signature_blob
```

'dss_signature_blob' değeri, r'yi ve ardından s'yi (uzunluk veya dolgu içermeyen, imzasız ve ağ bayt sırasına göre olan 160 bit tam sayılardır) içeren bir dize olarak kodlanır.

"ssh-rsa" anahtar biçimi aşağıdaki özel kodlamaya sahiptir:

```
string   "ssh-rsa"
mpint    e
mpint    n
```

Burada 'e' ve 'n' parametreleri imza anahtarı bloğunu oluşturur.

Bu anahtar biçimini kullanarak imzalama ve doğrulama, SHA-1 karma kullanılarak [RFC3447] içindeki RSASSA-PKCS1-v1_5 şemasına göre gerçekleştirilir.

Ortaya çıkan imza aşağıdaki gibi kodlanmıştır:

```
string   "ssh-rsa"
string   rsa_signature_blob
```

'rsa_signature_blob' değeri, s (bir tam sayıdır, uzunluk veya dolgu içermeyen, imzasız ve ağ bayt sırasına göre) içeren bir dize olarak kodlanmıştır.

"pgp-sign-rsa" yöntemi, sertifikaların, genel anahtarın ve imzanın OpenPGP uyumlu ikili biçimde ([RFC2440]) olduğunu gösterir. Bu yöntem, anahtarın bir RSA anahtarı olduğunu gösterir.

"pgp-sign-dss" yukarıdaki gibidir, ancak anahtarın bir DSS anahtarı olduğunu gösterir.

7. Anahtar Değişimi

Anahtar değişimi (kex), her iki tarafın da desteklenen algoritmaların ad listelerini göndermesiyle başlar. Her bir taraf, her kategoride tercih edilen bir algoritmaya sahiptir ve herhangi bir zamanda çoğu uygulamanın aynı tercih edilen algoritmayı kullanacağı varsayılır. Her iki taraf, diğer tarafın hangi algoritmayı kullandığını tahmin edebilir ve tercih edilen yöntem için uygunsa, algoritmaya göre bir ilk anahtar değişim paketi gönderebilir.

Tahmin şu durumlarda yanlış kabul edilir:

- kex algoritması ve/veya ana bilgisayar anahtarı algoritmasının yanlış tahmin edilmesi (sunucu ve istemcinin farklı tercih edilen algoritması vardır) veya

- Diğer algoritmalardan herhangi biri üzerinde anlaşma sağlanamıyorsa (prosedür aşağıda Bölüm 7.1'de tanımlanmıştır).

Aksi takdirde, tahmin doğru kabul edilir ve iyimser olarak gönderilen paket, ilk anahtar değişim paketi olarak ele alınmalıdır.

Ancak, tahmin yanlışsa ve bir paket taraflardan biri veya her ikisi tarafından iyimser bir şekilde gönderildiyse, bu tür paketler yok sayılmalıdır (tahmindeki hata ilk paket(ler)in içeriğini etkilemese bile) ve uygun taraf doğru ilk paketi göndermelidir.

Bir anahtar değişim yöntemi, anahtar değişim mesajları bir imza veya sunucunun özgünlüğünün başka bir kanıtını içeriyorsa, açık sunucu kimlik doğrulamasını kullanır. Bir anahtar değişim yöntemi, sunucunun, özgünlüğünü kanıtlamak için, aynı zamanda, istemcinin doğrulayabileceği bir mesaj ve karşılık gelen bir MAC göndererek, paylaşılan sırrı, K'yi bildiğini kanıtlaması gerekiyorsa, örtük sunucu kimlik doğrulamasını kullanır.

Bu belge tarafından tanımlanan anahtar değişim yöntemi, açık sunucu kimlik doğrulamasını kullanır. Ancak, bu protokol ile örtük sunucu kimlik doğrulaması içeren anahtar değişim yöntemleri kullanılabilir. Örtük sunucu kimlik doğrulaması ile bir anahtar değişiminden sonra, istemci daha fazla veri göndermeden önce hizmet isteği mesajına bir yanıt beklemelidir.

7.1. Algoritma Pazarlığı

Anahtar değişimi, her iki tarafın da aşağıdaki paketi göndermesiyle başlar:

```
byte         SSH_MSG_KEXINIT
byte[16]     cookie (random bytes)
name-list    kex_algorithms
name-list    server_host_key_algorithms
name-list    encryption_algorithms_client_to_server
name-list    encryption_algorithms_server_to_client
name-list    mac_algorithms_client_to_server
name-list    mac_algorithms_server_to_client
name-list    compression_algorithms_client_to_server
name-list    compression_algorithms_server_to_client
name-list    languages_client_to_server
name-list    languages_server_to_client
boolean      first_kex_packet_follows
uint32       0 (reserved for future extension)
```

Algoritma ad listelerinin her biri, virgülle ayrılmış bir algoritma adları listesi olmalıdır (bkz. Desteklenen (izin verilen) her algoritma, en çoktan en aza doğru tercih sırasına göre listelenmelidir.

Her isim listesindeki ilk algoritma, tercih edilen (tahmin edilen) algoritma olmalıdır. Her isim listesi en az bir algoritma ismi içermelidir.

Cookie

' cookie', gönderen tarafından oluşturulan rastgele bir değer olmalıdır. Amacı, her iki tarafın da anahtarları ve oturum tanımlayıcısını tam olarak belirlemesini imkansız kılmaktır.

kex_algorithms

Anahtar değişim algoritmaları yukarıda tanımlanmıştır. İlk algoritma tercih edilen (ve tahmin edilen) algoritma olmalıdır. Her iki taraf da aynı tahmini yapıyorsa, o algoritma kullanılmalıdır. Aksi takdirde, bir anahtar değişim yöntemi seçmek için aşağıdaki algoritma kullanılmalıdır: İstemcinin kex algoritmalarını teker teker yineleyin. Aşağıdaki koşulları karşılayan ilk algoritmayı seçin:

+ sunucu da algoritmayı destekler,

+ algoritma şifreleme özellikli bir ana bilgisayar anahtarı gerektiriyorsa, sunucunun server_host_key_algoritmaları üzerinde istemci tarafından da desteklenen şifreleme özellikli bir algoritma vardır ve

+ Algoritma imza özellikli bir ana bilgisayar anahtarı gerektiriyorsa, sunucunun server_host_key_algoritmaları üzerinde istemci tarafından da desteklenen imza özellikli bir algoritma vardır.

Tüm bu koşulları sağlayan bir algoritma bulunamazsa, bağlantı başarısız olur ve her iki tarafın da bağlantısını kesmesi gerekir.

server_host_key_algorithms

Sunucu ana bilgisayar anahtarı için desteklenen algoritmaların ad listesi. Sunucu, ana bilgisayar anahtarlarına sahip olduğu algoritmaları listeler; müşteri kabul etmeye istekli olduğu algoritmaları listeler. Bir ana bilgisayar için, muhtemelen farklı algoritmalara sahip birden çok ana bilgisayar anahtarı olabilir.

Bazı ana bilgisayar anahtarları hem imzaları hem de şifrelemeyi desteklemeyebilir (bu, algoritmadan belirlenebilir) ve bu nedenle tüm ana bilgisayar anahtarları, tüm anahtar değişim yöntemleri için geçerli değildir.

Algoritma seçimi, seçilen anahtar değişim algoritmasının bir imza mı yoksa şifreleme özellikli bir ana bilgisayar mı gerektirdiğine bağlıdır.
anahtar. Bunu açık anahtar algoritma adından belirlemek mümkün olmalıdır. İstemcinin isim listesindeki gereksinimleri karşılayan ve sunucu tarafından da desteklenen ilk algoritma seçilmelidir. Böyle bir algoritma yoksa, her ikisi de tarafların bağlantısı kesilmelidir.

encryption_algorithms

Tercih sırasına göre kabul edilebilir simetrik şifreleme algoritmalarının (şifreler olarak da bilinir) isim listesi. Her yöne seçilen şifreleme algoritması, aynı zamanda sunucunun isim listesinde de bulunan istemcinin isim listesindeki ilk algoritma olmalıdır. Böyle bir algoritma yoksa, her iki tarafın da bağlantısını kesmesi gerekir.

Kabul edilebilir olması için "hiçbirinin" açıkça listelenmesi gerektiğini unutmayın. Tanımlanan algoritma adları Bölüm 6.3'te listelenmiştir.

mac_algorithms

Tercih sırasına göre kabul edilebilir MAC algoritmalarının bir isim listesi. Seçilen MAC algoritması, aynı zamanda sunucunun isim listesinde de bulunan istemcinin isim listesindeki ilk algoritma olmalıdır. Böyle bir algoritma yoksa, her iki tarafın da bağlantısını kesmesi gerekir.

Kabul edilebilir olması için "hiçbirinin" açıkça listelenmesi gerektiğini unutmayın. MAC algoritma adları Bölüm 6.4'te listelenmiştir.

compression_algorithms

Tercih sırasına göre kabul edilebilir sıkıştırma algoritmalarının bir isim listesi. Seçilen sıkıştırma algoritması, aynı zamanda sunucunun isim listesinde de bulunan istemcinin isim listesindeki ilk algoritma olmalıdır. Böyle bir algoritma yoksa, her iki tarafın da bağlantısını kesmesi gerekir.

Kabul edilebilir olması için "hiçbirinin" açıkça listelenmesi gerektiğini unutmayın. Sıkıştırma algoritması adları Bölüm 6.2'de listelenmiştir.

languages

Bu, tercih sırasına göre dil etiketlerinin bir ad listesidir [RFC3066]. Her iki taraf da bu isim listesini görmezden gelebilir. Herhangi bir dil tercihi yoksa, bu isim listesi [SSH-ARCH] Bölüm 5'te tanımlandığı gibi boş olmalıdır. Dil etiketleri, gönderen tarafça ihtiyaç duyulduğu bilinmedikçe mevcut olmamalıdır.

first_kex_packet_follows

Tahmin edilen bir anahtar değişim paketinin takip edip etmediğini gösterir. Tahmini bir paket gönderilecekse, bu doğru olmalıdır. Tahmini paket gönderilmeyecekse, bu yanlış olmalıdır.

Diğer taraftan SSH_MSG_KEXINIT paketini aldıktan sonra, her iki taraf da tahminlerinin doğru olup olmadığını anlayacaktır. Diğer tarafın tahmini yanlışsa ve bu alan doğruysa, sonraki paket sessizce yok sayılmalı ve ardından her iki taraf da anlaşmalı anahtar değişim yöntemiyle belirlendiği şekilde hareket etmelidir. Tahmin doğruysa, anahtar değişimi tahmin edilen paketi kullanmaya devam etmelidir.

SSH_MSG_KEXINIT mesaj alışverişinden sonra anahtar değişim algoritması çalıştırılır. Anahtar değişim yöntemi tarafından belirtildiği gibi, birkaç paket alışverişini içerebilir.

Bir taraf, anahtar değişimi veya yeniden değişim için bir SSH_MSG_KEXINIT mesajı gönderdiğinde, bir SSH_MSG_NEWKEYS mesajı gönderene kadar (Bölüm 7.3), aşağıdakiler dışında herhangi bir mesaj göndermemelidir:

- Taşıma katmanı genel mesajları (1 ila 19) (ancak SSH_MSG_SERVICE_REQUEST ve SSH_MSG_SERVICE_ACCEPT gönderilmemelidir);
- Algoritma anlaşma mesajları (20 ila 29) (ancak daha fazla SSH_MSG_KEXINIT mesajı gönderilmemelidir);
- Özel anahtar değişim yöntemi mesajları (30 ila 49).

Bölüm 11'in hükümleri, tanınmayan mesajlar için geçerlidir.

Bununla birlikte, bir anahtarın yeniden değiş tokuşu sırasında, bir SSH_MSG_KEXINIT mesajı gönderdikten sonra, tarafların her birinin diğer taraftan bir SSH_MSG_KEXINIT mesajı almadan önce uçuş halinde olabilecek rastgele sayıda mesajı işlemeye hazır olması gerektiğini unutmayın.

7.2. Anahtar Değişiminden Çıktı

Anahtar değişimi iki değer üretir: paylaşılan bir gizli K ve bir değişim karma H. Şifreleme ve kimlik doğrulama anahtarları bunlardan türetilir. Birinci anahtar değişiminden gelen değişim karması H ayrıca bu bağlantı için benzersiz bir tanımlayıcı olan oturum tanımlayıcısı olarak kullanılır. Özel bir anahtara sahip olduğunun kanıtı olarak imzalanan verilerin bir parçası olarak kimlik doğrulama yöntemleri tarafından kullanılır. Hesaplandıktan sonra, anahtarlar daha sonra yeniden değiş tokuş edilse bile oturum tanımlayıcısı değişmez.

Her anahtar değişim yöntemi, anahtar değişiminde kullanılan bir karma işlevi belirtir. Anahtar türetmede aynı karma algoritma kullanılmalıdır. Burada HASH diyeceğiz.

Şifreleme anahtarları, bilinen bir değere sahip HASH ve K olarak aşağıdaki gibi hesaplanmalıdır:

- İlk IV istemciden sunucuya: HASH(K || H || "A" || session_id)

(Burada K, mpint olarak ve "A" bayt ve session_id olarak ham veri olarak kodlanmıştır. "A", tek A karakteri, ASCII 65 anlamına gelir).

- İstemciye ilk IV sunucusu: HASH(K || H || "B" || session_id)
- İstemciden sunucuya şifreleme anahtarı: HASH(K || H || "C" || session_id)
- İstemciye şifreleme anahtarı sunucusu: HASH(K || H || "D" || session_id)
- Sunucuya bütünlük anahtarı istemcisi: HASH(K || H || "E" || session_id)
- İstemciye bütünlük anahtarı sunucusu: HASH(K || H || "F" || session_id)

Anahtar veriler, hash çıktısının başlangıcından alınmalıdır. Hash değerinin başlangıcından gerektiği kadar bayt alınır.

Gereken anahtar uzunluğu HASH'ın çıktısından daha uzunsa, anahtar, K ve H'nin ve o ana kadarki tüm anahtarın HASH'ının hesaplanması ve elde edilen baytların (HASH'ın ürettiği kadar) anahtara eklenmesiyle genişletilir. . Bu işlem, yeterli anahtar materyal bulunana kadar tekrarlanır; anahtar bu değerin başından alınır. Başka bir deyişle:

```
K1 = HASH(K || H || X || session_id)   (X is e.g., "A")
K2 = HASH(K || H || K1)
K3 = HASH(K || H || K1 || K2)
...
key = K1 || K2 || K3 || ...
```

K'deki entropi miktarı HASH'ın iç durum boyutundan daha büyükse, bu süreç entropiyi kaybeder.

7.3. Anahtarları Kullanıma Alma

Anahtar değişimi, her iki tarafın da bir SSH_MSG_NEWKEYS mesajı göndermesiyle sona erer. Bu mesaj eski anahtarlar ve algoritmalar ile gönderilir. Bu mesajdan sonra gönderilen tüm mesajlar yeni anahtarları ve algoritmaları kullanmalıdır.

Bu mesaj alındığında, almak için yeni anahtarlar ve algoritmalar kullanılmalıdır.

Bu mesajın amacı, bir tarafın, anahtar değişiminde bir şeyler ters giderse, diğer tarafın anlayabileceği bir SSH_MSG_DISCONNECT mesajı ile yanıt verebilmesini sağlamaktır.

byte SSH_MSG_NEWKEYS

8. Diffie-Hellman Anahtar Değişimi

Diffie-Hellman (DH) anahtar değişimi, her iki tarafça da tek başına belirlenemeyen paylaşılan bir sır sağlar. Anahtar değişimi, ana bilgisayar kimlik doğrulamasını sağlamak için ana bilgisayar anahtarıyla bir imza ile birleştirilir. Bu anahtar değişim yöntemi, Bölüm 7'de tanımlandığı gibi açık sunucu kimlik doğrulaması sağlar.

Bir anahtarı değiştirmek için aşağıdaki adımlar kullanılır. Bunda C, istemcidir; S sunucudur; p büyük bir güvenli asaldır; g, GF(p)'nin bir alt grubu için bir üreticidir; q, alt grubun sırasıdır; V_S, S'nin tanımlama dizisidir; V_C, C'nin tanımlama dizisidir; K_S, S'nin genel ana bilgisayar anahtarıdır; I_C, C'nin SSH_MSG_KEXINIT mesajıdır ve I_S, S'nin mesajıdır.
Bu bölüm başlamadan önce değiştirilen SSH_MSG_KEXINIT mesajı.

1. C rastgele bir x (1 < x < q) sayısı üretir ve e = g^x mod p'yi hesaplar. C, e'yi S'ye gönderir.

2. S rastgele bir y (0 < y < q) sayısı üretir ve f = g^y mod p'yi hesaplar. S e alır. K = e^y mod p, H = hash(V_C || V_S || I_C || I_S || K_S || e || f || K) hesaplar (bu elemanlar türlerine göre kodlanmıştır; aşağıya bakınız) ve özel ana bilgisayar anahtarıyla H üzerinde imza s. S, C'ye (K_S || f || s) gönderir. İmzalama işlemi, ikinci bir karma işlemi içerebilir.

3. C, K_S'nin gerçekten S için ana bilgisayar anahtarı olduğunu doğrular (örneğin, sertifikalar veya yerel bir veritabanı kullanarak). C'nin anahtarı doğrulamadan kabul etmesine de izin verilir; bununla birlikte, bunu yapmak protokolü aktif saldırılara karşı güvensiz hale getirecektir (ancak birçok ortamda kısa vadede pratik nedenlerle istenebilir). C daha sonra K = f^x mod p, H = hash(V_C || V_S || I_C || I_S || K_S || e || f || K) hesaplar ve H üzerindeki s imzasını doğrular.

[1, p-1] aralığında olmayan 'e' veya 'f' değerleri her iki tarafça da gönderilmemeli veya kabul edilmemelidir. Bu koşul ihlal edilirse, anahtar değişimi başarısız olur.

Bu, aşağıdaki mesajlarla uygulanır. Değişim karmasını hesaplamak için karma algoritma, yöntem adıyla tanımlanır ve HASH olarak adlandırılır. İmzalama için ortak anahtar algoritması şu kişilerle görüşülür: SSH_MSG_KEXINIT mesajları.

İlk olarak, müşteri aşağıdakileri gönderir:

byte SSH_MSG_KEXDH_INIT
mpint e

Sunucu daha sonra aşağıdakilerle yanıt verir:

byte SSH_MSG_KEXDH_REPLY
string server public host key and certificates (K_S)
mpint f
string signature of H

Hash H, aşağıdakilerin birleştirilmesinin HASH hash'i olarak hesaplanır:

string V_C, müşterinin tanımlama dizisi (CR ve LF hariçtir)

string V_S, sunucunun tanımlama dizisi (CR ve LF hariçtir)

string I_C, istemcinin SSH_MSG_KEXINIT'inin yükü

string I_S, sunucunun SSH_MSG_KEXINIT yükü

string K_S, host anahtarı
mpint   e, müşteri tarafından gönderilen değişim değeri
mpint   f, sunucu tarafından gönderilen değişim değeri
mpint   K, paylaşılan sır

Bu değer, değişim karması olarak adlandırılır ve anahtar değişimini doğrulamak için kullanılır. Değişim karması gizli tutulmalıdır.

İmza algoritması, orijinal veriye değil, H'ye uygulanmalıdır. Çoğu imza algoritması, karma ve ek dolgu içerir (örneğin, "ssh-dss", SHA-1 karma oluşturmayı belirtir). Bu durumda, H'yi hesaplamak için veriler önce HASH ile hashlenir ve ardından H, imzalama işleminin bir parçası olarak SHA-1 ile hashlenir.

8.1. diffie-hellman-group1-sha1

"diffie-hellman-group1-sha1" yöntemi, HASH ve Oakley Group 2 [RFC2409] (1024-bit MODP Group) olarak SHA-1 ile Diffie-Hellman anahtar değişimini belirtir. Bilinen tüm uygulamalar şu anda onu desteklediğinden, bu yöntem birlikte çalışabilirlik için desteklenmelidir. Oakley Group 2'nin kullanımını belirtmesine rağmen, bu yöntemin "grup1" ifadesi kullanılarak adlandırıldığını unutmayın.

8.2. diffie-hellman-group14-sha1

"diffie-hellman-group14-sha1" yöntemi, HASH ve Oakley Group 14 [RFC3526] (2048-bit MODP Group) olarak SHA-1 ile bir Diffie-Hellman anahtar değişimini belirtir ve ayrıca desteklenmesi gerekir.

9. Anahtar Değişimi

Anahtar değişimi, halihazırda bir anahtar değişimi yapılmadığında bir SSH_MSG_KEXINIT paketi gönderilerek başlatılır (Bölüm 7.1'de açıklandığı gibi). Bu mesaj alındığında, alınan SSH_MSG_KEXINIT'in zaten bir yanıt olduğu durumlar dışında, taraf kendi SSH_MSG_KEXINIT mesajıyla yanıt vermelidir. Taraflardan herhangi biri yeniden değişimi başlatabilir, bu roller değiştirilmemelidir (yani, sunucu sunucu olarak kalır ve istemci istemci olarak kalır).

Anahtar değişimi, değişim başlatıldığında yürürlükte olan şifreleme kullanılarak gerçekleştirilir. Anahtar değişiminden sonra (ilk anahtar değişiminde olduğu gibi) yeni bir SSH_MSG_NEWKEYS gönderilmeden önce şifreleme, sıkıştırma ve MAC yöntemleri değiştirilmez. Yeniden değişim, değişmeden kalacak oturum tanımlayıcısı dışında, ilk anahtar değişimiyle aynı şekilde işlenir. Yeniden değişim sırasında algoritmaların bir kısmının veya tamamının değiştirilmesine izin verilir. Ana bilgisayar anahtarları da değişebilir. Tüm anahtarlar ve başlatma vektörleri, değiş tokuştan sonra yeniden hesaplanır. Sıkıştırma ve şifreleme bağlamları sıfırlanır.

Aktarılan her gigabayt veriden sonra veya her saat bağlantı süresinden sonra, hangisi daha önce gerçekleşirse, anahtarların değiştirilmesi önerilir. Ancak, yeniden değişim bir ortak anahtar işlemi olduğundan, makul miktarda işlem gücü gerektirir ve çok sık yapılmamalıdır.

SSH_MSG_NEWKEYS paketi gönderildikten sonra daha fazla uygulama verisi gönderilebilir; anahtar değişimi, SSH taşıma katmanının üzerinde bulunan protokolleri etkilemez.

10. Servis Talebi

Anahtar değişiminden sonra müşteri bir hizmet talep eder. Hizmet bir adla tanımlanır. Adların formatı ve yeni adları tanımlama prosedürleri [SSH-ARCH] ve [SSH-NUMBERS] içinde tanımlanmıştır.

Şu anda, aşağıdaki isimler rezerve edilmiştir:

ssh-userauth
ssh-connection

Algoritma adlarına uygulandığı gibi, hizmet adlarına da benzer yerel adlandırma ilkesi uygulanır. Yerel bir hizmet, "hizmetadı@etkialanı"nın özel kullanım sözdizimini kullanmalıdır.

byte SSH_MSG_SERVICE_REQUEST
string service name

Sunucu hizmet talebini reddederse, uygun bir SSH_MSG_DISCONNECT mesajı göndermesi GEREKİR ve bağlantıyı kesmelidir.

Hizmet başladığında, anahtar değişimi sırasında oluşturulan oturum tanımlayıcısına erişimi olabilir.

Sunucu hizmeti destekliyorsa (ve istemcinin kullanmasına izin veriyorsa), aşağıdakilerle yanıt vermelidir:

byte SSH_MSG_SERVICE_ACCEPT
string service name

Servisler tarafından kullanılan mesaj numaraları kendilerine ayrılan alanda olmalıdır (bkz. [SSH-ARCH] ve [SSH-NUMARALARI]). Taşıma düzeyi kendi mesajlarını işlemeye devam edecektir.

Örtük sunucu kimlik doğrulaması ile bir anahtar değişiminden sonra, istemcinin daha fazla veri göndermeden önce hizmet istek mesajına yanıt beklemesi gerektiğini unutmayın.

11. Ek Mesajlar

Taraflardan herhangi biri herhangi bir zamanda aşağıdaki mesajlardan herhangi birini gönderebilir.

11.1. Bağlantı Kesilme Mesajı

bayt SSH_MSG_DISCONNECT
uint32 neden kodu
ISO-10646 UTF-8 kodlamasında dize açıklaması [RFC3629]
dize dil etiketi [RFC3066]

Bu mesaj, bağlantının derhal kesilmesine neden olur. Tüm uygulamalar bu mesajı işleyebilmelidir; bu mesajı gönderebilmeleri gerekir.

Gönderici bu mesajdan sonra herhangi bir veri göndermemeli veya almamalıdır ve alıcı bu mesajı aldıktan sonra herhangi bir veri kabul etmemelidir.

Bağlantı Kesilme Mesajı 'açıklama' dizesi, insan tarafından okunabilir bir biçimde daha spesifik bir açıklama sağlar. Bağlantı Kesilme Mesajı 'neden kodu' nedeni daha makine tarafından okunabilir bir biçimde (yerelleştirmeye uygun) verir ve aşağıdaki tabloda gösterilen değerlere sahip olabilir. Okunabilirlik için bu tabloda ondalık gösterimin görüntülendiğini, ancak değerlerin aslında uint32 değerleri olduğunu unutmayın.

Symbolic name	reason code
SSH_DISCONNECT_HOST_NOT_ALLOWED_TO_CONNECT	1
SSH_DISCONNECT_PROTOCOL_ERROR	2
SSH_DISCONNECT_KEY_EXCHANGE_FAILED	3
SSH_DISCONNECT_RESERVED	4
SSH_DISCONNECT_MAC_ERROR	5
SSH_DISCONNECT_COMPRESSION_ERROR	6
SSH_DISCONNECT_SERVICE_NOT_AVAILABLE	7
SSH_DISCONNECT_PROTOCOL_VERSION_NOT_SUPPORTED	8
SSH_DISCONNECT_HOST_KEY_NOT_VERIFIABLE	9
SSH_DISCONNECT_CONNECTION_LOST	10
SSH_DISCONNECT_BY_APPLICATION	11
SSH_DISCONNECT_TOO_MANY_CONNECTIONS	12
SSH_DISCONNECT_AUTH_CANCELLED_BY_USER	13
SSH_DISCONNECT_NO_MORE_AUTH_METHODS_AVAILABLE	14
SSH_DISCONNECT_ILLEGAL_USER_NAME	15

'açıklama' dizesi görüntülenirse, terminal kontrol karakterleri göndererek saldırıları önlemek için [SSH-ARCH]'de tartışılan kontrol karakteri filtrelemesi kullanılmalıdır.

0x00000010 ila 0xFDFFFFFF aralığındaki yeni Bağlantı Kesme Mesajı 'neden kodu' değerlerinin (ve ilgili 'açıklama' metninin) atama istekleri, [RFC2434]'te açıklandığı gibi IETF CONSENSUS yöntemiyle YAPILMALIDIR. 0xFE000000 ila 0xFFFFFFFF aralığındaki Bağlantı Kesme Mesajı 'sebep kodu' değerleri özel kullanım için ayrılmıştır. Belirtildiği gibi, IANA'ya yönelik asıl talimatlar [SSH-NUMARALARI] içindedir.

11.2. Yok Sayılan Veri Mesajı

```
bayt SSH_MSG_IGNORE
string data
```

Tüm uygulamaların bu mesajı herhangi bir zamanda (tanımlama dizesini aldıktan sonra) anlaması (ve yoksayması) gerekir. Bunları göndermek için herhangi bir uygulamaya gerek yoktur. Bu mesaj, gelişmiş trafik analiz tekniklerine karşı ek bir koruma önlemi olarak kullanılabilir.

11.3. Hata Ayıklama Mesajı

```
byte     SSH_MSG_DEBUG
boolean  always_display
string   message in ISO-10646 UTF-8 encoding [RFC3629]
string   language tag [RFC3066]
```

Tüm uygulamalar bu mesajı anlamalıdır, ancak görmezden gelmelerine izin verilir. Bu mesaj, hata ayıklamaya yardımcı olabilecek bilgileri iletmek için kullanılır. 'always_display' doğruysa, mesaj görüntülenmelidir. Aksi takdirde, hata ayıklama bilgisi kullanıcı tarafından açıkça talep edilmedikçe görüntülenmemelidir.

' message ' yeni bir satır içermesi gerekmez. Ancak, CRLF (Satır Başı - Satır Besleme) çiftleriyle ayrılmış birden çok satırdan oluşmasına izin verilir.

' message ' dizesi görüntülenirse, terminal kontrol karakterleri göndererek saldırıları önlemek için [SSH-ARCH]'de tartışılan terminal kontrol karakter filtrelemesi kullanılmalıdır.

11.4. Ayrılmış Mesajlar

Bir uygulama, tanınmayan tüm mesajlara, mesajların alındığı sırayla bir SSH_MSG_UNIMPLEMENTED mesajı ile yanıt vermelidir. Bu tür mesajlar aksi takdirde göz ardı edilmelidir. Daha sonraki protokol sürümleri, bu mesaj türleri için başka anlamlar tanımlayabilir.

```
bayt SSH_MSG_UNIMPLEMENTED
uint32 paket reddedilen mesajın sıra numarası
```

12. Mesaj Numaralarının Özeti

Aşağıda, mesajların ve bunlarla ilişkili mesaj numaralarının bir özeti yer almaktadır.

```
SSH_MSG_DISCONNECT          1
SSH_MSG_IGNORE              2
SSH_MSG_UNIMPLEMENTED       3
SSH_MSG_DEBUG               4
SSH_MSG_SERVICE_REQUEST     5
SSH_MSG_SERVICE_ACCEPT      6
SSH_MSG_KEXINIT            20
SSH_MSG_NEWKEYS            21
```

Kex paketleri için 30-49 numaralarının kullanıldığını unutmayın. Farklı kex yöntemleri, bu aralıktaki mesaj numaralarını yeniden kullanabilir.

13. IANA Hususları

Bu belge bir kümenin parçasıdır. [SSH-ARCH], [SSH-USERAUTH], [SSH-CONNECT] ve bu belgede tanımlandığı gibi SSH protokolü için IANA değerlendirmeleri [SSH-NUMARALARI]'nda ayrıntılı olarak açıklanmıştır.

14. Güvenlik Hususları

Bu protokol, güvenli olmayan bir ağ üzerinden güvenli bir şifreli kanal sağlar. Sunucu ana bilgisayar kimlik doğrulaması, anahtar değişimi, şifreleme ve bütünlük koruması gerçekleştirir. Ayrıca, daha yüksek seviyeli protokoller tarafından kullanılabilecek benzersiz bir oturum kimliği türetir.

Bu protokol için tam güvenlik hususları [SSH-ARCH] içinde verilmektedir.

15 Referanslar

15.1. Normatif Referanslar

[SSH-ARCH] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[SSH-USERAUTH] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.

[SSH-CONNECT] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, January 2006.

[SSH-NUMBERS] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, January 2006.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm ", RFC 1321, April 1992.

[RFC1950] Deutsch, P. and J-L. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.

[RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2144] Adams, C., "The CAST-128 Encryption Algorithm", RFC 2144, May 1997.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2440] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

[RFC3066] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[FIPS-180-2] US National Institute of Standards and Technology, "Secure Hash Standard (SHS)", Federal Information Processing Standards Publication 180-2, August 2002.

[FIPS-186-2] US National Institute of Standards and Technology, "Digital Signature Standard (DSS)", Federal Information Processing Standards Publication 186-2, January 2000.

[FIPS-197] US National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197, November 2001.

[FIPS-46-3] US National Institute of Standards and Technology, "Data Encryption Standard (DES)", Federal Information Processing Standards Publication 46-3, October 1999.

[SCHNEIER] Schneier, B., "Applied Cryptography Second Edition: protocols algorithms and source in code in C", John Wiley and Sons, New York, NY, 1996.

[TWOFISH] Schneier, B., "The Twofish Encryptions Algorithm: A 128-Bit Block Cipher, 1st Edition", March 1999.

15.2. Referans Bilgileri

[RFC0894] Hornig, C., "Standard for the transmission of IP datagrams over Ethernet networks", STD 41, RFC 894, April 1984.

[RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC2412] Orman, H., "The OAKLEY Key Determination Protocol", RFC 2412, November 1998.

[ssh-1.2.30] Ylonen, T., "ssh-1.2.30/RFC", File within compressed tarball ftp://ftp.funet.fi/pub/unix/security/login/ssh/ssh-1.2.30.tar.gz, November 1995.

Authors' Addresses

Tatu Ylonen
SSH Communications Security Corp

Valimotie 17
00380 Helsinki
Finland

EMail: ylo@ssh.com

Chris Lonvick (editor)
Cisco Systems, Inc.
12515 Research Blvd.
Austin 78759
USA

EMail: clonvick@cisco.com

Ticari Marka Bildirimi

"ssh", Amerika Birleşik Devletleri ve/veya diğer ülkelerde tescilli bir ticari markadır.

Fikri mülkiyet

IETF, bu belgede açıklanan teknolojinin uygulanması veya kullanımı ile ilgili olduğu iddia edilebilecek herhangi bir Fikri Mülkiyet Haklarının veya diğer hakların geçerliliği veya kapsamı veya bu haklar kapsamındaki herhangi bir lisansın ne ölçüde uygulanıp uygulanmayacağı konusunda herhangi bir pozisyon almamaktadır. kullanılabilir; ne de bu tür hakları belirlemek için bağımsız bir çaba sarf ettiğini göstermez. RFC belgelerindeki haklara ilişkin prosedürlere ilişkin bilgiler BCP 78 ve BCP 79'da bulunabilir.

IETF Sekreterliği'ne yapılan fikri mülkiyet hakları açıklamalarının kopyaları ve kullanıma sunulacak her türlü lisans güvencesi veya bu spesifikasyonun uygulayıcıları veya kullanıcıları tarafından bu tür mülkiyet haklarının kullanımı için genel bir lisans veya izin alma girişiminin sonucu elde edilebilir. adresindeki IETF çevrimiçi fikri mülkiyet hakları deposundan
http://www.ietf.org/ipr.

IETF, ilgili herhangi bir tarafı, bu standardı uygulamak için gerekli olabilecek teknolojiyi kapsayabilecek telif haklarını, patentleri veya patent başvurularını veya diğer mülkiyet haklarını dikkatine sunmaya davet eder. Lütfen bilgileri şu adresten IETF'ye iletin:

ietf-ipr@ietf.org.

teşekkürler

RFC Düzenleyici işlevi için finansman, IETF İdari Destek Etkinliği (IASA) tarafından sağlanır.