Gazi Üniv. Müh. Mim. Fak. Der.
Cilt 23, No 3, 539-548, 2008

J. Fac. Eng. Arch. Gazi Univ.
Vol 23, No 3, 539-548, 2008

SİMETRİK VE ASİMETRİK ŞİFRELEME YÖNTEMLERİNE METOTLAR: ÇIRPILMIŞ VE BİRLEŞİK AKM-VKM

Kerim YILDIRIM ve H. Engin DEMİRAY
Bilgisayar Mühendisliği, Mühendislik Fakültesi, Kocaeli Üniversitesi, KOCAELİ, kerim_yildirim@yahoo.com
hedemiray@yahoo.com

(Geliş/Received: 15.05.2007; Kabul/Accepted: 28.07.2008)

ÖZET

Klasik Anahtar Kapsülleme Mekanizması-AKM (Key Encapsulation Mechanism-KEM) ve Veri Kapsülleme Mekanizmasına-VKM (Data Encapsulation Mechanism-DEM), eğer bu iki yapıdan her hangi birisinde bir zayıflık varsa, kötü niyetli bir kişinin saldırması mümkündür. Böyle bir zayıflık durumunda ortaya çıkabilecek bir saldırıyı engellemek ve AKM-VKM yapısını daha da güçlendirmek için biz burada yeni bir metot öneriyoruz: Çırpılmış AKM-VKM. Bu metot içinde kullanılan karıştırma algoritması ile AKM ve VKM birleştirilip permütasyona tabi tutulmakta ve sonra da simetrik bir anahtarla şifrelenerek AKM ve VKM tek bir blok haline getirilmektedir. Ayrıca iki düğüm arasında kurulacak uzun süreli bir iletişim için, gelişmiş oturum anahtarı üzerinden mesaj bütünlüğü ve kimlik kontrolü yapan yeni bir metot sunuyoruz: Birleşik AKM-VKM. Bu metotta gönderilen tüm mesajlar ayrı simetrik anahtarlarla şifrelenmekte ve tüm mesaj trafiğinin bütünlük kontrolü yapılmaktadır. Birleşik AKM-VKM metodunda RSA kullanılarak değişik anahtar uzunluklarıyla (256 bit, 512 bit ve 1024 bit) yapılan benzetimlerde AKM'in hesaplanma süresi Tag-AKM/VKM [2] ve Fujisaki-Okamoto AKM-VKM [3] metotlarıyla kıyaslandığında yaklaşık olarak %40 oranında zaman tasarrufu sağlanmıştır.

Anahtar Kelimeler: hibrit şifreleme, AKM-VKM yapısı, veri bütünlüğü, sayısal imza, tüm mesaj trafiğinin kontrolü, geliştirilmiş oturum anahtarı.

METHODS FOR INTEGRATING SYMMETRIC AND ASYMMETRIC ENCRYPTION SCHEMES: SCRAMBLED AND COMBINED KEM-DEM

ABSTRACT

It is possible that an adversary may attack to conventional key encapsulation mechanism (KEM) and data encapsulation mechanism (DEM) structure, if any weakness exists in these algorithms. In order to prevent such kind of attacks and to make the KEM-DEM structure secure in a very strong sense, we propose a scramble algorithm in which KEM and DEM are combined and permuted, then encrypted with symmetric key driving into one block. In addition, for a long term communication between two nodes, we propose a new scheme Combined KEM-DEM which provides message integrity and sender's identity control via an integrated session key. In this method, each message is encrypted with different symmetric key and whole communication traffic is controlled. Simulations made using RSA with different key lengths (256, 512, and 1024 bits) show that by employing the Combined KEM-DEM scheme, computation times of KEM for the encryption and decryption are reduced by 40% as compared to Tag-KEM/DEM [2] and Fujisaki-Okamoto's KEM-DEM [3] schemes.

Keywords: Hybrid encryption, KEM-DEM structure, message integrity, digital signature, whole communication process control, integrated session key.

1. GİRİŞ (INTRODUCTION)

Saldırganlara karşı mesajların gizliliğini korumak için simetrik ve asimetrik şifreleme algoritmaları [1] kullanılmaktadır. Hibrit şifreleme algoritmaları, simetrik ve asimetrik şifreleme algoritmalarının avantajlarını birleştirmektedir. Hibrit şifreleme sisteminde mesajı şifrelemek için simetrik şifreleme algoritması, mesajı şifrelemek için kullanılan simetrik

anahtarı şifrelemek için de asimetrik şifreleme algoritması kullanılmaktadır.

Son zamanlarda hibrit şifreleme [2,3] üzerinde yapılan araştırma ve çalışmalar, Anahtar Kapsülleme Mekanizması-AKM (Key Encapsulation Mechanism-KEM) ve Veri Kapsülleme Mekanizması-VKM (Data Encapsulation Mechanism-DEM) üzerine yoğunlaşmaktadır. Bu modelde, hibrit şifreleme iki ayrı parçadan oluşmaktadır. AKM ve VKM. Tag-AKM/VKM [2] metodu ve Fujisaki-Okamoto AKM-VKM [3] metodu hibrit şifreleme yöntemi için gösterilebilecek bilinen örneklerdir. Tag-AKM/VKM [2] metodunda AKM'in hesaplanması iki fonksiyona paylaştırılmıştır. Birinci fonksiyon rasgele bir anahtar seçerken ikinci fonksiyon verilen bir tag ile bu anahtarı şifrelemektedir. Fujisaki-Okamoto AKM-VKM [3] metodu ise zayıf simetrik ve asimetrik şifreleme algoritmalarını güçlü bir asimetrik şifreleme algoritmasına dönüştüren bir yapıya sahiptir.

Veri bütünlüğü, kimlik kontrolü, anahtar güvenliği ve bunları sağlarken harcanan süre kriptolojide önemli parametrelerdir. Hem Tag-AKM/VKM [2] metodu hem de Fujisaki-Okamoto AKM-VKM [3] metodu bu kıstasları sağlamaktadır. Ancak bizim burada sunduğumuz Birleşik AKM-VKM metodundan daha yavaş çalışmaktadırlar.

Klasik AKM-VKM yapısında kullanılan simetrik ve asimetrik şifreleme algoritmalarında herhangi bir zayıflık varsa, kötü niyetli bir kişi bu zayıflığı kullanarak saldırı düzenleyebilir. Bu da tüm metodun güvenliğinin tehlikeye girmesi demektir. Biz bu çalışmada iki farklı AKM-VKM metodu sunuyoruz: Çırpılmış AKM-VKM ve Birleşik AKM-VKM.

Burada önerilen iki metotta da güvenilir ve emniyetli bir AKM-VKM yapısının tesis edilmesi hedeflenmiştir. Bunu sağlamak için (8) numaralı eşitlik ile gösterilen karıştırma algoritması (EW) kullanılmıştır. Karıştırma algoritması ile yapılan işlemde, AKM-VKM yapısında ortaya çıkabilecek bir zayıflığın maskelenmesi ve klasik AKM-VKM yapısının güçlendirilmesi amaçlanmıştır.

Çırpılmış AKM-VKM metodunda AKM ve VKM, alıcının açık anahtarı (public key) kullanılarak, karıştırma algoritması (EW) ile simetrik olarak şifrelenmektedir. Karıştırma algoritması AKM ve VKM'i ilk önce permütasyona tabi tutmakta, daha sonra elde edilen metni de simetrik anahtarla şifrelemektedir. Bu metotta amaç, AKM ve VKM'in tek bir blok haline dönüştürülmesi ve karıştırma işleminin sadece doğru anahtara sahip alıcı tarafından çözülmesini sağlamaktır.

Önerdiğimiz ikinci metot Birleşik AKM-VKM, sadece hibrit şifreleme zamanını kısaltmakla kalmamakta, ayrıca tüm mesaj trafiğinin bütünlüğünü de sağlamaktadır. Yapmış olduğumuz benzetim, önermiş olduğumuz metodun Tag-AKM/VKM [2] ve Fujisaki-Okamoto AKM-VKM [3] metotlarından daha hızlı olduğunu göstermiştir. Bu metot sadece şifreleme zamanını kısaltmakla kalmamakta, aynı zamanda kimlik kontrolünü, veri bütünlüğünü ve anahtar güvenliğini de sağlamaktadır. Tüm bunlara ilave olarak, gönderilen her bir mesaj farklı bir simetrik anahtarla şifrelenmektedir. Böylelikle her bir simetrik anahtar yalnızca bir kez kullanılacağı için, saldırganların elinde anahtarı elde edebilmek için, aynı anahtarla şifrelenmiş birden fazla metin olmayacaktır.

2. ÇIRPILMIŞ AKM-VKM METODU (SCRAMBLED KEM-DEM SCHEME)

Veri şifrelemesinde kullanılan simetrik ve asimetrik anahtarlar, şifreleme işleminin daha güvenilir ve emniyetli olmasi için birlikte kullanılmaktadırlar. Buna ilave olarak veri bütünlüğü ve kimlik kontrolünün eklenmesi güçlü bir güvenlik sağlayacaktır.

AKM-VKM yapısında [1-3,8,9] mesaj simetrik anahtarla şifrelenmekte [1-3,10], daha sonra bu simetrik anahtar asimetrik anahtarla şifrelenmekte [4-7] ve şifrelenen mesaja eklenmektedir. Ancak bu yapıda saldırgan aynı anda üzerinde çalışabileceği iki adet veriye sahip olmaktadır. Saldırgan hem AKM üzerinde hem de VKM üzerinde ayrı ayrı çalışarak şifrelenen mesaja veya simetrik anahtara ulaşabilir. Eğer AKM veya VKM yapılarının herhangi birinde bir zayıflık varsa, saldırganın başarıya ulaşma ihtimali artacağı için tüm güvenlik aşamaları başarısız olacaktır.

AKM ve VKM yapısından kaynaklanabilecek bir zayıflıktan dolayı saldırganın elde edeceği bu avantajı ortadan kaldırmak ve AKM-VKM yapısını daha da güçlendirmek için biz burada iki parçalı AKM-VKM yapısını (8) numaralı eşitlik ile gösterilen karıştırma algoritması (EW) ile tek parça haline dönüştüren yeni bir yapı öneriyoruz. Bu yapıda sadece doğru anahtara sahip olan kişi AKM ve VKM'i birbirinden ayırabilir.

Metodun ilk adımında rasgele bir anahtardan (k_{sym}) simetrik anahtar (k_{group}) üretilmekte ve bu simetrik anahtarla mesaj şifrelenmektedir. Elde edilen çıktı VKM olarak adlandırılır. Daha sonra simetrik anahtarın üretilmesinde kullanılan rasgele anahtar alıcının açık anahtarı ile şifrelenmektedir. (6) numaralı eşitlik ile yapılan işlem sayısal imza olarak adlandırılmaktadır. Orijinal mesajın "hash" özeti ve gönderenin açık anahtarı da sayısal imzaya eklenmekte ve tamamı alıcının açık anahtarı ile

şifrelenmektedir. Elde edilen çıktı AKM olarak adlandırılmaktadır.

En son aşamada, bu iki parçayı tek parçaya dönüştürmek için karıştırma algoritması (EW) kullanılmaktadır. Karıştırma algoritması ilk önce AKM ve VKM permütasyona tabi tutulmakta, daha sonra simetrik anahtarla şifrelenmekte ve elde edilen çıktı alıcıya gönderilmektedir. Burada simetrik anahtar olarak alıcının açık anahtarı kullanılmaktadır.

Simetrik anahtarla ilgili en önemli nokta; mesajın şifrelenmesi için kullanılan anahtar ile alıcıya gönderilen şifreli mesajın içine konulan anahtarın aynı olmamasıdır. Alıcıya gönderilen anahtar, orijinal mesajın şifrelenmesinde kullanılacak olan simetrik anahtarın üretilmesi için kullanılacak olan rasgele anahtardır. Yani burada ekstra bir simetrik anahtar üretme algoritması kullanılmaktadır.

Çırpılmış AKM-VKM'e ait metot ve açıklaması aşağıda verilmiştir:

Anahtar üretimi ve şifreleme işleminin detayları Tablo 1'de, algoritmanın blok diyagramı Şekil 1'de verilmiştir.

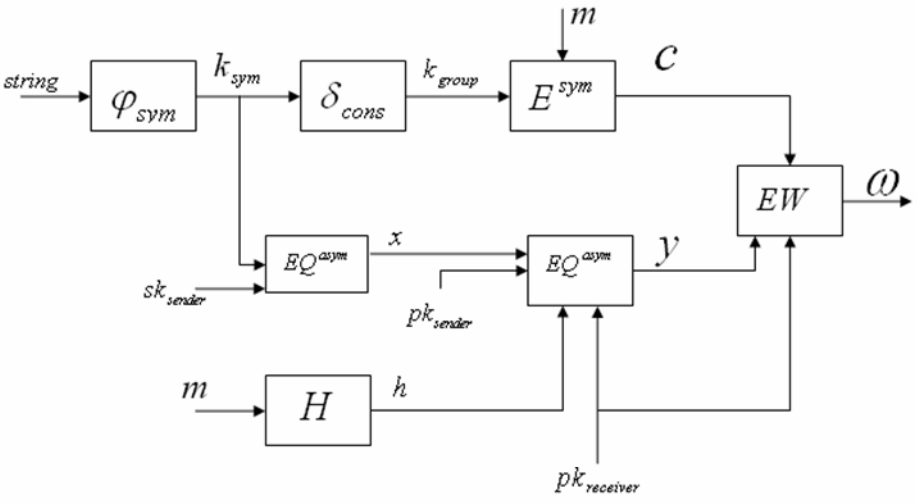
Simetrik anahtar üretimi: Şekil 1'de görüleceği gibi simetrik anahtar iki basamakta üretilmektedir. İlk basamakta simetrik anahtarın (k_{group}) üretiminde kullanılacak olan rasgele anahtar (k_{sym}) üretilmektedir (2). φ_{sym} algoritması her seferinde farklı bir anahtar üretecek yapıya sahiptir. k_{sym} ayrıca şifreli mesajın içinde alıcıya gönderilecek olan anahtardır. Simetrik anahtar üretme algoritması δ_{cons}, φ_{sym}'den farklıdır. δ_{cons} algoritması aynı girdi değerleri için aynı çıktıları üretir. δ_{cons} algoritması k_{sym} değerinden k_{group} değerini elde etmede kullanılmaktadır. k_{group} mesajın şifrelenmesinde kullanılacak olan simetrik anahtardır.

Çırpılmış AKM-VKM'e ait şifreleme algoritması:

$$\omega \leftarrow EW_{pk_{receiver}}(EQ^{asym}_{pk_{receiver}}(EQ^{asym}_{sk_{sender}}(k_{sym}), H(m), pk_{sender}), E^{sym}_{k_{group}}(m)) \quad (1)$$

Mesajın şifrelenmesi: Mesaj, simetrik şifreleme algoritması (4) ile şifrelenmekte ve simetrik anahtar olarak k_{group} kullanılmaktadır. Elde edilen sonuç (c) Veri Kapsülleme Mekanizması-VKM (Data Encapsulation Mechanism-DEM) [1-3] olarak adlandırılmaktadır. Veri bütünlüğünü (message integrity) kontrol edebilmek için (5) numaralı eşitlik ile gösterilen algoritma ile mesajın "hash" özeti elde edilmektedir. Burada ayrıca mesajı gönderen kişinin kimliğinin tespit edebilmek için (6) numaralı eşitlik ile sayısal imza eklenmiştir. Mesajı gönderen kişinin kimliğini tespit edebilmek için rasgele anahtar (k_{sym}) gönderenin gizli anahtarı ile şifrelenir. Gizli anahtar sadece tek bir kişi tarafından bilindiği için kimlik kontrolü için kullanılmaktadır. Alıcı mesajı gönderenin açık anahtarını kullanarak gelen mesajdaki k_{sym} değerini elde eder. Eğer başlangıçta mesaj doğru kişi tarafından şifrelenmişse alıcı doğru k_{sym} değerini elde edebilir. Aksi takdirde elde edeceği sonuç işine yaramayacaktır.

(7) numaralı eşitlik Anahtar Kapsülleme Mekanizması-AKM (Key Encapsulation Mechanism-KEM) olarak adlandırılmaktadır. AKM, veri bütünlüğü, k_{sym} değeri üzerinde gönderilen sayısal imza ve gönderenin açık anahtar bilgilerini içermektedir. Burada, gönderilen mesaj üzerinden gönderici kimliğinin tespit edilmesinin mümkün olmadığı kabul edildiği için pk_{sender} değeri y değerine eklenmiştir. pk_{sender} değerine ancak AKM'in şifresinin çözülmesinden sonra ulaşılabilir. Veri



Anahta

$k_{sym} \leftarrow$

$k_{group} \leftarrow$

$\omega \leftarrow EW_{pk_{receiver}}(c, y)$ (x)

Şekil 1. Çırpılmış AKM-VKM Metoduna ait blok diyagram (Block Diagram of Scrambled KEM-DEM Scheme)

bütünlüğünü kontrol etmek için mesajın "hash" özeti de eklenmekte ve hepsi birlikte alıcının açık anahtarı ile şifrelenmektedir. Bunda amaç; şifrelenmiş mesajın sadece doğru alıcı tarafından açılmasını sağlamaktır.

Şifrelenen mesaj c (VKM) ve anahtar kapsülü y (AKM) karıştırma algoritması ile karıştırılmakta ve simetrik olarak şifrelenmektedir (8). Simetrik anahtar olarak alıcının açık anahtarı kullanılmaktadır. Burada unutulmaması gereken nokta; mesaj üzerinden alıcının kimliğinin tespit edilmesinin mümkün olmadığının kabul edilmiş olmasıdır. Dolayısıyla sadece doğru alıcı şifreli mesajı çözebilecektir ya da saldırganın sadece karıştırma algoritması ile yapılan AKM-VKM birleştirmesini çözebilmesi için sistemdeki tüm açık anahtarları denemesi gerekecektir

3. BİRLEŞİK AKM-VKM METODU (COMBINED KEM-DEM SCHEME)

Kısa süreli iletişimlerde Çırpılmış AKM-VKM metodu rahatlıkla kullanılabilir. Ancak iki düğüm arasında kurulacak olan uzun süreli bir iletişimde, mesajın şifrelenmesi aşamasında kullanılan asimetrik şifreleme algoritması nedeniyle, bu metodu kullanmak çok zaman alacağı için doğru bir tercih olmayacaktır. Çırpılmış AKM-VKM, Tag-AKM/VKM [2] ve Fujisaki-Okamoto AKM-VKM [3] metotlarında AKM'in hesaplanması ve elde edilmesi VKM'e göre daha fazla işlem gerektirmekte ve daha fazla zaman almaktadır.

Ayrıca AKM'e sayısal imza ve veri bütünlüğü değerlerinin eklenmesi de AKM'in hesaplanma süresinin uzamasına neden olmaktadır. AKM'in hesaplanmasındaki bu zaman artışının kısa süreli iletişimler için bir zararı olmayacaktır. Ancak uzun süreli bir iletişim için durum tam tersi olacak ve gereksiz zaman kaybı meydana geleceği için AKM'in hesaplanma süresi bu metotların darboğazı olacaktır.

Yukarıda bahsedilen darboğazı ortadan kaldırmak için burada yeni bir metot öneriyoruz: Birleşik AKM-VKM. Bu metot sadece hesaplama zamanını kısaltmakla kalmamakta ayrıca tüm mesajlaşma trafiğini de kontrol etmektedir. Ayrıca metotta gelişmiş bir oturum anahtarı kullanılarak her bir mesaj ayrı bir simetrik anahtarla şifrelenmekte olup kimlik kontrolü, veri bütünlüğü ve anahtar güvenliği hususlarını da sağlamaktadır.

Kısaca bu metotta (y) ve mesajın (m) "hash" özeti hesaplanmakta, elde edilen "hash" özeti alıcının açık anahtarı ile şifrelenmekte, şifrelenen anahtar VKM'e eklenmektedir.

3.1. Şifreleme (Encryption)

İki düğüm arasında kurulacak olan iletişim üç adımda sağlanmaktadır: Bağlantı talebi, anahtar dağıtımı ve veri transferi. İlk iki adım, karşılıklı olarak kimlik kontrolü amacıyla kullanılmakta olup basitçe el sıkışma mekanizması olarak da ifade edilebilir. Gerçek veri transferine üçüncü adımda başlanmaktadır. Burada bağlantı talebinde bulunan düğüm istemci, bağlantı talebini alan düğüm de sunucu olarak adlandırılmıştır.

İlk adımda, istemci bağlantı talebini sunucuya ω_1 ile bildirir. ω_1 aşağıdaki şekilde hesaplanmaktadır. İstemci tarafından yapılan, (9) numaralı eşitlik ile verilen bağlantı talebinin ayrıntıları Tablo 2'de gösterilen (10-16) numaralı eşitlikler ile verilmiştir.

İstemciden gönderilen mesajın (m_1) içeriğinin bir önemi yoktur. m_1 sadece istemcinin kimliğinin kontrol edilmesinde kullanılacaktır. Birinci ve ikinci adımlarda kullanılan metot, Çırpılmış AKM-VKM metodu ile aynıdır. Sadece bu metoda mesaj numarasını gösteren basit bir sayıcı (Φ) eklenmiştir.

$$\omega_1 \leftarrow EW_{pk_{receiver}}(EQ^{asym}_{pk_{receiver}}(EQ^{asym}_{sk_{sender}}(k_{sym}), H(m_1), pk_{sender}, \Phi), E^{sym}_{k_{group}}(m_1)) \quad (9)$$

İkinci adım olan anahtar dağıtımında, sunucu oturum başlangıç anahtarını (m_2) istemciye gönderir. Bu adımda kullanılan eşitlikler Tablo 3'te verilmiştir.

Tablo 2. Birleşik-AKM-VKM Metodu, 1. Adım: Bağlantı Talebi (Combined-KEM-DEM Scheme, Step 1: Communication Request)

Anahtar Üretimi		**Şifreleme**	
$k_{sym} \leftarrow \varphi_{sym}(string)$	(10)	$c \leftarrow E^{sym}_{k_{group}}(m_1)$	(12)
$k_{group} \leftarrow \delta_{cons}(k_{sym})$	(11)	$h \leftarrow H(m_1)$	(13)
		$x \leftarrow EQ^{asym}_{sk_{sender}}(k_{sym})$	(14)
		$y_1 \leftarrow EQ^{asym}_{pk_{receiver}}(x, h, pk_{sender}, \Phi)$	(15)
		$\omega_1 \leftarrow EW_{pk_{receiver}}(c, y_1)$	(16)

İlk iki adımda sunucu ve istemci karşılıklı olarak kimlik kontrolü yapmakta ve oturum başlangıç anahtarı üzerinde anlaşmaktadır.

Üçüncü adım olan veri transferinde kullanılacak olan metot ilk iki adımdan farklı fakat onlardan bağımsız değildir. Bu adım ve sonrasındaki mesajlar hep birbirleriyle bağlantılı olarak çalışmakta ve her mesaj şifrelenirken bir önceki mesajdan veriler almaktadır.

Bu bağlantı sayesinde hem AKM'in hesaplanma süresi oldukça kısaltılmış hem de tüm mesaj trafiğinin kontrol edilmesine olanak sağlanmıştır. Oluşan fark, Tablo 3 ile Tablo 4 karşılaştırıldığında da rahatlıkla görülmektedir. Bir ve ikinci adımda iki kez asimetrik şifreleme algoritması kullanılırken, üçüncü adım ve sonrasında bir kez asimetrik şifreleme yapılmaktadır. Ayrıca üçüncü adımda asimetrik şifreleme yapılacak olan verinin boyutu da ("byte" olarak) kısaltılmış ve tek bir veri girdi olarak kullanılmıştır.

AKM'in hesaplanma süresi ve mesajın iletim zamanını kısaltmak için yeni bir değer (Ω) hesaplanmaktadır. Ω değeri simetrik anahtarın üretilmesinde kullanılacak değer olup gönderilecek olan mesaj (m) ile en son gönderilmiş olan mesajdaki AKM değerinden (y) hesaplanmaktadır.

Üçüncü adımda ve sonrasında şifreleme için kullanılacak metot (24) numaralı eşitlik ile, metodun blok diyagramı Şekil 3'te, ayrıntıları da Tablo 4'te verilmiştir.

$$\omega_n \leftarrow (EQ^{asym}_{pk_{receiver}}(H(m_n, y_{n-2}) \parallel \Phi_n) \quad (24)$$

$$\parallel EW_{\Omega_n}(E^{sym}_{\delta_{cons}(\Omega_{n-1})}(m_n)))$$

Ω değerinin hesaplanmasına üçüncü adımda başlanmakta olup başlangıç değeri olarak (25) numaralı eşitlik ile gösterilen değeri alır. (25) numaralı eşitlik sadece üçüncü veri transferinde yani n=3 için geçerlidir. Daha sonrasında bu eşitlik kullanılmayacaktır. Üçüncü veri transferinden sonra (n>3 için) k_{sym_n} değerini Ω_{n-1}'den alır. Ω_{n-1} değeri, bir sonraki mesajı gönderecek olan düğüm tarafından, simetrik anahtarın üretiminde kullanılır. Bu da bize gönderilecek olan her mesajın farklı bir anahtarla şifrelenmesini sağlar.

(29) numaralı eşitlik üçüncü adımdan sonra (n>3 için) kullanılacak olup y değeri de Ω'nin hesaplanmasında kullanılacak olan değerlerden birisidir.

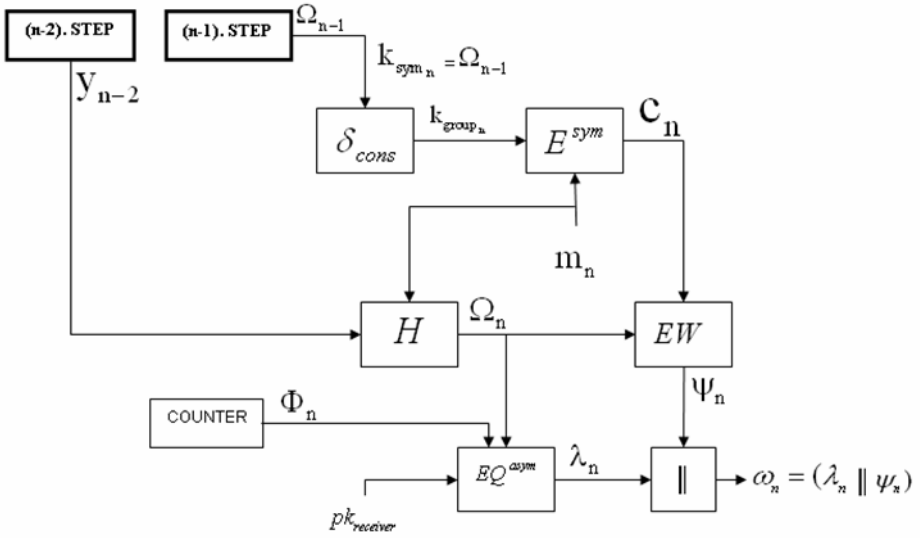
Üçüncü adımda ve sonrasında yapılan işlemleri daha iyi anlamak için; iki düğüm arasında kurulmuş olan iletişimin Aydın (istemci) ve Burak (sunucu) arasında geçtiğini varsayalım. İlk iki adımda Aydın ve Burak karşılıklı olarak birbirlerinin kimliklerini kontrol ettiler ve Burak Aydın'a oturum başlangıç anahtarını gönderdi. Aydın Burak'a göndereceği mesajı bu anahtarla şifreleyecek, ayrıca Aydın, tüm mesaj trafiğinin kontrolünü ve tüm mesajlaşmanın bütünlüğünü de sağlamak için Burak'a göndermiş olduğu en son mesajdan da veri alacak, bunları birleştirerek hazırladığı şifreli mesajı Burak'a gönderecektir.

Tablo 3. Birleşik-AKM-VKM Metodu, 2. Adım: Anahtar Dağıtımı (Combined-KEM-DEM Scheme, Step 2: Key Distribution)

Anahtar Üretimi		**Şifreleme**	
$k_{sym} \leftarrow \varphi_{sym}(string)$	(17)	$c \leftarrow E^{sym}_{k_{group}}(m_2)$	(19)
$k_{group} \leftarrow \delta_{cons}(k_{sym})$	(18)	$h \leftarrow H(m_2)$	(20)
		$x \leftarrow EQ^{asym}_{sk_{sender}}(k_{sym})$	(21)
		$y_2 \leftarrow EQ^{asym}_{pk_{receiver}}(x, h, pk_{sender}, \Phi)$	(22)
		$\omega_2 \leftarrow EW_{pk_{receiver}}(c, y_2)$	(23)

Tablo 4. Birleşik-AKM-VKM Şifreleme Metodu, n. Adım: Veri Transferi (Combined-KEM-DEM Encryption Scheme, Step n: Data Transfer)

Anahtar Üretimi		**Şifreleme**	
$\Omega_2 = m_2$	(25)	$c_n \leftarrow E^{sym}_{k_{group_n}}(m_n)$	(28)
$k_{sym_n} = \Omega_{n-1}$	(26)	$\Omega_n \leftarrow H(m_n, y_{n-2})$	(29)
$k_{group_n} \leftarrow \delta_{cons}(k_{sym_n})$	(27)	$\psi_n \leftarrow EW_{\Omega_n}(c_n)$	(30)
		$y_n \leftarrow (\Omega_n \parallel \Phi_n)$	(31)
		$\lambda_n \leftarrow EQ^{asym}_{pk_{receiver}}(y_n)$	(32)
		$\omega_n \leftarrow (\lambda_n \parallel \psi_n)$	(33)



Şekil 3. Birleşik AKM-VKM Metodunun Blok Diyagramı (n'nci Adım) (Block Diagram of Combined KEM-DEM Scheme (Step n))

n'nci adımda Aydın Burak'a mesaj göndermek istiyorsa:

VKM değerini hesaplamak için:

- Burak'ın en son göndermiş olduğu mesajdaki Ω değerini (yani Ω_{n-1} değerini) alacak
- Ω_{n-1} değerini kullanarak kendi mesajını şifrelemek için kullanacağı simetrik anahtarı (k_{group_n}) elde edecek
- k_{group_n} ile mesajını şifreleyecek
- Elde ettiği çıktıyı karıştırma algoritması (EW) ile karıştırıp simetrik olarak tekrar şifreleyecektir. Simetrik anahtar olarak da Ω_n değeri kullanılmaktadır. Ω_n değeri (29) numaralı eşitlikte de gösterildiği gibi, Aydın'ın Burak'a bu adımda göndereceği mesaj (m_n) ile, Burak'a en son göndermiş olduğu şifreli mesajdaki y değerinden (y_{n-2}'den) elde edilmektedir.

AKM değerini (y_n) *hesaplamak için:*

- AKM değerini (y_n) hesaplamak için Ω_n değeri ile mesaj numarasını gösteren Φ_n değeri birleştirilir (Φ_n değeri Ω_n değerinin arkasına eklenir)
- Daha sonra y_n değeri Burak'ın açık anahtarı ile şifrelenir.
- AKM ve VKM değerlerinin hesaplanmasından sonra bu iki değer birbirine eklenerek Burak'a gönderilir.

Aydın mesajı gönderdikten sonra Burak'tan alacağı mesajın Ω_n değerinden üretilecek olan simetrik anahtarla şifrelenmiş olmasını bekleyecektir. Aksi takdirde gelen mesajın Burak'tan olmadığını anlayacaktır.

Burak da Aydın'a mesaj gönderirken yukarıdaki işlem basamaklarını takip edecektir.

Aydın'ın göndermiş olduğu mesaj bir saldırganın eline geçse bile Ω değerine ulaşamayacaktır. Bu değere ulaşabilmesi için Burak'ın gizli anahtarını bilmesi gerekecektir (Ω değeri (32) numaralı eşitlik ile de gösterildiği gibi Burak'ın açık anahtarı ile şifrelenmişti) ki gizli anahtar sadece anahtarın gerçek sahibinde bulunabilir.

Metodun diğer bir avantajı da mesajın şifrelenmesi için kullanılan anahtar ile mesajın içinde gönderilen anahtarın aynı olmamasıdır. Hatırlanacağı üzere Aydın göndereceği mesajını şifrelemek için kullanacağı simetrik şifreyi Ω_{n-1} değerinden elde ediyordu. Ω_{n-1} değerini de Burak göndermişti. Aydın da Burak'a göndereceği mesajını şifrelemesi için Ω_n değerini göndermişti. Yani Aydın mesajını şifrelemek için Ω_{n-1} değerini kullanmış, Burak'a da Ω_n değerini göndermiştir. Dolayısıyla mesajın şifrelenmesi için kullanılan Ω değeri ile karşı tarafa gönderilen Ω değeri aynı olmayacaktır.

Buradaki diğer bir nokta da Ω değerinin önceki mesajlarla bağlantılı olması ve sürekli değişmesidir. Bu da bize hem tüm mesajlaşma trafiğini kontrol etmemizi hem de her mesaj için farklı bir simetrik anahtar oluşturmamızı sağlamaktadır.

3.2. Şifreyi Çözme (Decryption)

Birleşik AKM-VKM'e ait üçüncü adım ve sonrasında ($n \geq 3$ için) kullanılacak olan algoritmalar (34-40) numaralı eşitlikler ile Tablo 5'de verilmiştir.

Şifreleme kısmında anlatıldığı gibi, üçüncü mesajın Aydın tarafından Burak'a gönderildiğini düşünelim (Burada anlatılan şifreyi çözme işlemi n'nci adım için de aynen geçerlidir). Burak Aydın'dan gelen bu mesajı açmak için:

- Mesajı aldıktan sonraki ilk adım AKM (λ_n) ve VKM (ψ_n) değerlerini tekrar elde etmektir. Bu en kolay adımdır, çünkü AKM ve VKM sadece birbirlerine eklenmişti (34).

AKM'in şifresinin çözülmesi:

- Bir sonraki adım AKM (λ_n)'den Ω_n ve Φ_n değerlerinin elde edilmesidir. Şifreleme işleminde bu iki değer birleştirilip alıcının açık anahtarıyla şifrelenmişti. Yani Aydın bu iki değeri $\hat{\Omega}_n$ ve $\hat{\Phi}_n$ tekrar elde eder (35). $\hat{\Phi}_n$ değeri gönderilen mesajın mesaj numarasını göstermekte olup hem şifreyi çözmek için kullanılacak olan algoritmanın belirlenmesini hem de n değerinin belirlenmesini sağlamaktadır. Burak elde ettiği $\hat{\Omega}_n$ değerini (eğer tüm şifreyi çözme işlemi başarı ile sonuçlanırsa) Aydın'a göndereceği mesajı şifrelemek için kullanacağı simetrik anahtarı elde etmek için kullanacaktır. Ayrıca bu değeri, gönderilen mesajın yolda değişip değişmediğini kontrol etmek ve mesajı gönderen kişi ile bir önceki mesajı gönderen kişinin aynı kişi olup olmadığını kontrol etmek (dolaylı kimlik kontrolü) amacıyla kullanılacaktır.

VKM'in şifresinin çözülmesi:

- AKM'in şifresinin çözülmesinden sonra elde edilen $\hat{\Omega}_n$ değeri ile karıştırma algoritması (30) ile yapılmış olan şifreleme çözülür (36).
- Elde edilen $\hat{c}_n$ değeri orijinal mesajın simetrik anahtarla şifrelenmiş halidir. Bu şifreyi çözmek için simetrik anahtara ihtiyaç vardır. Hatırlanacağı üzere Aydın'a bu simetrik anahtarı üretmesi için gerekli olan değeri Burak kendisi göndermişti ve Aydın bu değeri kullanarak simetrik anahtarı üretmişti (26-27). Dolayısıyla Burak simetrik anahtarın üretilmesi için gerekli olan değere Ω_{n-1} zaten sahiptir. Kendisinde olan bu değeri kullanarak simetrik anahtarı (k_{group_n}) elde eder (37-38).
- Elde ettiği bu simetrik anahtarla $\hat{c}_n$'i çözer ve Aydın'ın göndermiş olduğu mesajı elde eder.

Veri Bütünlüğü ve Kimlik Kontrolü:

Burak tüm bu işlemlerden sonra elde ettiği verilerin doğru olduğunu, yolda değişip değişmediğini ve mesajı gönderenin gerçekten Aydın olduğunu anlamak için Ω_n değerini tekrar hesaplar.

- Ω_n değerini tekrar hesaplamak için şifresini çözdüğü mesajı $\hat{m}_n$ ve en son Aydın'dan gelen mesajdan elde etmiş olduğu y_{n-2} değerini kullanır. Eğer elde edeceği Ω_n değeri (40) ile Aydın'ın gönderdiği mesajdan elde etmiş olduğu $\hat{\Omega}_n$ değeri (35) aynı ise mesaj yolda değişmemiştir ve mesajı gönderen kişi gerçekten Aydın'dır sonucuna ulaşılır. Şöyleki:

Kimlik Kontrolü:

- Ω_n değerini hesaplarken kullanılan y_{n-2} değeri, Burak'a Aydın tarafından gönderilmişti ve bu değer gelen mesajın içinde olmamasına rağmen Burak'ta bulunmaktaydı (y_{n-2} değeri Aydın'ın gönderdiği bir önceki mesajdan elde edilmektedir). Araya girip Aydın'mış gibi davranan ve Burak'ı yanıltmaya çalışan bir saldırgan bu değere sahip olamayacağı için, saldırganın göndereceği mesajdan elde edilen $\hat{\Omega}_n$ değeri ile Ω_n değerinin aynı olması mümkün olmayacaktır. Mesajların birbirleriyle böyle irtibatlandırılması sayesinde bu mesajı gönderen kişiyle bir önceki mesajı gönderen kişi aynı kişidir

Tablo 5. Birleşik-AKM-VKM Şifreyi Çözme Metodu: n'nci Adım (Combined-KEM-DEM Decryption Scheme, Step n)

$(\hat{\lambda}_n, \hat{\psi}_n) \leftarrow \omega_n$	(34)
$(\hat{\Omega}_n, \hat{\Phi}_n) \leftarrow DQ^{asym}_{sk_{receiver}}(\hat{\lambda}_n)$	(35)
$\hat{c}_n \leftarrow DW_{\hat{\Omega}_n}(\hat{\psi}_n)$	(36)
$k_{sym_n} = \Omega_{n-1}$	(37)
$k_{group_n} \leftarrow \delta_{cons}(k_{sym_n})$	(38)
$\hat{m}_n \leftarrow D^{sym}_{k_{group_n}}(\hat{c}_n)$	(39)
$\Omega_n \leftarrow H(\hat{m}_n, y_{n-2})$	(40)
Eğer $\Omega_n = \hat{\Omega}_n$ ise $m_n = \hat{m}_n$ ve sonuç = $\hat{m}_n$ değilse sonuç = $\perp$	

sonucuna ulaşılacak ve iletişimin ilk iki adımında da karşılıklı olarak kimlik kontrolü yapıldığı için mesajın gerçekten aynı kişiden geldiği sonucuna varılacaktır. Aynı zamanda mesajların birbirleriyle böyle irtibatlandırılması sayesinde iletişimin başlangıcından bitişine kadar da tüm mesajlaşmanın bütünlük kontrolü de yapılmış olacaktır.

Veri Bütünlüğü:

- Ω_n değerini tekrar hesaplanırken $\hat{m}_n$ değeri kullanılmaktadır (40). Eğer herhangi bir şekilde mesaj yolda değişirse (29) numaralı eşitlik ile elde edilen Ω_n değeri ile (40) numaralı eşitlikten elde edilen değerler aynı olamayacak ve sonuç olarak $m_n = \hat{m}_n$ anlamına geleceği için sonuç başarısız ($\perp$) olacaktır.

4. PERFORMANS ÖLÇÜMÜ (PERFORMANCE EVALUATION)

Birleşik AKM-VKM'e ait AKM'in hesaplanma süresi ile Tag-AKM/VKM [2] ve Fujisaki-Okamoto AKM-VKM [3] metotlarına ait AKM hesaplanma sürelerini karşılaştırmak için RSA kullandık. Benzetim, Intel Celeron 2.4 GHz CPU, 512 MB RAM, Microsoft XP işletim sistemi üzerinde Dave Shapiro'nun JavaScript ile hazırlamış olduğu RSA uyarlaması [11] ile yapılmıştır. Kullanılan anahtar uzunlukları: 256, 512 ve 1024 bit. AKM'in hesaplanmasında kullanılan asimetrik şifreleme algoritmalarında girdi olarak kullanılan simetrik anahtar ve "hash" özetinin boyutları aynı alınmıştır. Kullanılan anahtar uzunlukları: 128, 256 ve 512 bit. Tablo 6'da gösterilen değerler AKM'in hesaplanma süreleridir.

Birleşik AKM-VKM metodunun Tag-AKM/VKM [2] ve Fujisaki-Okamoto AKM-VKM [3] metotlarına göre en büyük avantajı asimetrik şifreleme algoritmasına giren veri büyüklüğünün yaklaşık olarak %50 oranında azaltılmış olmasıdır. Giren veri büyüklüğünde elde edilen bu küçültme sayesinde AKM'in hesaplanma süresi kısaltılmış ve önerdiğimiz metodun etkinliğini arttırmıştır.

Benzetim sonuçları da açıkça göstermektedir ki, özellikle küçük boyutlu anahtarlar kullanıldığında Birleşik AKM-VKM diğer iki metottan oldukça hızlı çalışmaktadır. Anahtar uzunlukları arttıkça performanslar birbirine yaklaşmaktadır.

6. SONUÇ (CONCLUSION)

Bu çalışmada iki farklı AKM-VKM yapısı (Çırpılmış AKM-VKM ve Birleşik AKM-VKM) sunulmuştur.

Çırpılmış AKM-VKM metodu ile klasik AKM-VKM yapısında ortaya çıkabilecek bir zayıflığı ortadan kaldırmak ve AKM-VKM yapısını daha güçlü hale getirmek için, iki parça olan AKM-VKM yapısı, karıştırma algoritması (EW) ile tek parça haline dönüştürülmektedir.

Uzun süreli iletişim için yeni bir metot önerilmiştir: Birleşik AKM-VKM. Her ne kadar uzun süreli iletişimler için yaygın olarak kullanılan metotlar olsa da bizim önerdiğimiz yöntemde:

- Her bir mesaj farklı bir anahtarla şifrelenmekte
- Her düğüm karşı tarafın göndereceği mesajı şifrelerken kullanacağı anahtarı karşı tarafa kendisi söylemekte
- Karşı tarafa gönderilen şifreli mesajın içinde orijinal mesajın şifrelenmesi için kullanılan simetrik anahtar bulunmamakta
- Tüm mesajlaşma trafiğinin bütünlüğü kontrol edilmekte
- Ayrıca bunları yaparken Tag-AKM/VKM [2] ve Fujisaki-Okamoto AKM-VKM [3] metotlarıyla karşılaştırıldığında Tablo 6'da da görüleceği gibi %40 oranında zaman tasarrufu sağlamakta

Tablo 6. Birleşik AKM-VKM, Tag-AKM/VKM ve Fujisaki-Okamoto AKM-VKM metotlarına ait AKM'lerin hesaplanma süreleri (Calculation Times of KEM using Combined KEM-DEM, Tag-KEM/DEM and Fujisaki-Okamoto's KEM-DEM schemes)

Simetrik Anahtar ve "Hash" Özeti Uzunlukları	Hibrit Metot	RSA Anahtar Uzunlukları					
		256 bit		512 bit		1024 bit	
		Şifreleme (Enc) (ms)	Çözme (Dec) (ms)	Şifreleme (Enc) (ms)	Çözme (Dec) (ms)	Şifreleme (Enc) (ms)	Çözme (Dec) (ms)
128 bit	Fujisaki-Okamoto	250	6531	500	26102	969	92422
	Tag-AKM/VKM	250	6531	500	26102	969	92422
	Birleşik AKM-VKM	172	4078	282	13619	890	91560
256 bit	Fujisaki-Okamoto	453	10102	781	38975	1063	95123
	Tag-AKM/VKM	453	10102	781	38975	1063	95123
	Birleşik AKM-VKM	265	6540	515	26644	984	94875
512 bit	Fujisaki-Okamoto	843	19097	1421	60250	2093	189734
	Tag-AKM/VKM	843	19097	1421	60250	2093	189734
	Birleşik AKM-VKM	469	9856	859	37718	1781	189016

- Bunların hepsini de geliştirilmiş bir oturum anahtarı (Ω_n) ile yapmaktadır.

Birleşik AKM-VKM metodu diğer AKM-VKM metotları ile de uyumludur ve birlikte kullanılabilir. Birleşik AKM-VKM metodunun ilk iki adımında (karşılıklı kimlik kontrolü ve anahtar değişimi) Çırpılmış AKM-VKM metodundan faydalanılmıştır. Burada kullanılan Çırpılmış AKM-VKM yerine başka metotlar da (Tag-AKM/VKM [2] ve Fujisaki-Okamoto AKM-VKM [3] gibi) kolaylıkla kullanılabilir. Yani ilk iki adımda yapılan kimlik kontrolü ve anahtar değişimini sağlayabilecek herhangi bir metot Çırpılmış AKM-VKM yerine adapte edilebilir. Birleşik AKM-VKM metodunun asıl özellikleri ve avantajları üçüncü adımdan sonra devreye girmekte olup başka AKM-VKM metotlarıyla birlikte kullanılmasında sakınca yoktur. Tam tersine onların da etkinliğini arttıracak özelliklere sahiptir.

SEMBOLLER (Nomenclature)

φ_{sym} : Rasgele anahtar olan k_{sym}'yi üreten algoritma. Rasgele anahtarı üretmek için algoritma "string" bir değer alır. Ancak "string" değer aynı olsa bile algoritma her seferinde farklı bir anahtar üretmektedir.

k_{sym} : Alıcıya gönderilecek olan anahtar. Bu anahtar ayrıca mesajın şifrelenmesinde kullanılacak olan simetrik anahtarın (k_{group}) üretilmesinde de kullanılmaktadır.

δ_{cons} : Simetrik anahtarı (k_{group}) üreten algoritma. Simetrik anahtarı üretmek için algoritma rasgele anahtarı (k_{sym}) kullanır. Bu algoritma φ_{sym}'den farklı olarak aynı girdi için aynı çıktı üreten bir yapıya sahiptir.

k_{group} : Simetrik anahtar ya da simetrik anahtarlar grubu. Biz bu anahtarı grup olarak adlandırdık, çünkü simetrik şifrelemede bir yerine birden fazla anahtar kullanılması daha fazla güvenlik sağlayacaktır, 3DES'te olduğu gibi. Ancak bu anahtarı tek bir anahtar gibi düşünmek de mümkündür. Her iki durum da bizim yapımıza uygundur.

m : Orijinal mesaj

c : Simetrik anahtarla şifrelenmiş olan orijinal mesaj. Bu değer aynı zamanda VKM'i ifade etmektedir.

E^{sym} : Simetrik şifreleme algoritması [10].

H : "Hash" fonksiyonu.

sk_A : Gönderenin/Alıcının gizli anahtarı.

pk_A : Gönderenin/Alıcının açık anahtarı.

EQ^{asym} : Asimetrik şifreleme algoritması.

x : Sayısal imza. Mesajı gönderenin kimliğini kontrol ederken kullanılacak olan değer.

y : AKM. Bu değerin içinde k_{sym}, h ve pk_{sender} değerleri vardır.

EW : AKM ve VKM'i karıştırmak ve şifrelemek için kullanılacak olan karıştırma algoritması.

ω : Alıcıya gönderilecek olan şifreli mesaj.

KAYNAKLAR (REFERENCES)

1. Dent, A.W., "Hybrid Cryptology", http://eprint.iacr.org/2004/210
2. Abe, M., Gennaro, R., Kurusawa, K., "Tag-KEM/DEM: A New Framework for Hybrid Encryption", In R.Cramer, editor, Advance in Cryptology-Eurocrypt 2005, volume 3494 of Lecture Notes in Computer Science, Springer-Verlag, 2003
3. Fujisaka, E., Okamoto, T., "Secure Integration of Asymmetric and Symmetric Encryption Schemes", In W.Wiener, editor, Advances in Cryptology-CRYPTO'99, volume 1666 of Lecture Notes in Computer Science, Springer-Verlag, 2001
4. Halevi, S., Krawczyk, H., "Public-Key Cryptography and Password protocols", In 5th ACM Conference on Computer and Communication Security. Nov 2000
5. Bellare, M. Rogaway, P., "Optimal Asymmetric Encryption-How to encrypt with RSA", Advances in Cryptology-CRYPTO'94
6. Bellare, M., Desai, A., Poitcheval, D., Rogaway, P., "Relations of Security for Public-Key Encryption Schemes", Advances in Cryptology-CRYPTO'98
7. Yuliang, Z., "Authenticated Public Key Encryption Schemes using Universal Hashing", http://grouper.ieee.org/groups/1363/P1363a/contributions/aes-uhf3.pdf
8. Galindo, D., Martin, S., Morillo, P., Villar ,J.L., "Fujisaki-Okamoto IND-CCA Hybrid Encryption Revisited"
9. Bellare, M., Boldyreva, A., Palacio, A., "An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. Advances in Cryptology", EUROCRYPT'04, Lecture Notes in Computer Science Vol. Springer-Verlag, 2004.
10. Daemen, J., Rijmen, V., "The Design of Rijndael AES-The Advanced Encryption Standard Series: Information Security and Cryptography", Springer Verlag 2002..
11. http://www.ohdave.com/rsa/