KRİPTOLOJİ VE EN YAYGIN KULLANILAN KRİPTOGRAFİK ALGORİTMALAR

Bölüm I: Temel kavramlar, simetrik ve açık anahtarlı algoritmalar, özet fonksiyonları ve performans ölçümleri

Cryptology and the Most Widely Used Cryptographic Algorithms, Part I: Fundamentals, Symmetric and Public-Key Algorithms, Hash Functions and Performance Measurements

Derleme Makale / Review Article

Dr. Burak Olgun
burakolgun@solutionhome.net
Solution Home Bilişim Tekn. ve Dan. Hizm., İstanbul

Kriptoloji • AES • ChaCha20 • RSA • Eliptik Eğri • SHA-2 / SHA-3 • Argon2 • Performans Ölçümü

ÖZET

Kriptoloji, bilgiyi yetkisiz kişilerden korumak için şifre tasarlayan kriptografi ile bu şifreleri çözmeye çalışan kriptanalizin birlikte oluşturduğu bilim dalıdır. Bir web sayfasını açmak, banka uygulamasına girmek ya da e-imzalı bir dilekçeyi UYAP'a göndermek, arka planda aynı küçük algoritma ailesinin çalışmasına dayanır. İki bölümlük serinin bu ilk makalesi, söz konusu ailenin kendisini ele almaktadır. Önce güvenlik hedefleri, Kerckhoffs ilkesi, Shannon'ın karıştırma-yayılma kavramları ve güvenlik biti açıklanmakta, El-Kindî'den AES yarışmasına uzanan tarihsel gelişim özetlenmektedir. Ardından simetrik şifreleme (DES, 3DES, AES, ChaCha20, çalışma kipleri, hafif algoritma Ascon ve Camellia, ARIA, GOST, SM4 gibi bölgesel standartlar), özet fonksiyonları ve mesaj doğrulama (MD5, SHA-1, SHA-2, SHA-3, BLAKE, HMAC, HKDF, Argon2) ile açık anahtarlı kriptografi (Diffie-Hellman, RSA, ECDSA, X25519, Ed25519) teknik ayrıntılarıyla incelenmektedir. Gerçek dünyadaki kırılmaların çoğunun algoritmadan değil uygulama hatalarından kaynaklandığı Heartbleed, ROCA, PS3 ve Debian OpenSSL örnekleriyle gösterilmektedir. Yazar, 21 algoritmayı OpenSSL 3.5.6 ile aynı donanımda ölçmüş; AES-GCM'in donanım desteğiyle saniyede yaklaşık 2 GB işlediğini, RSA imzasının ECDSA'dan 23 kat yavaş olduğunu ve kuantum sonrası ML-KEM-768'in X25519 kadar hızlı çalıştığını belirlemiştir. Makale, yazar tarafından test edilmiş altı kod örneğiyle sona ermektedir. Sertifika altyapısı, protokoller, kuantum sonrası geçiş ve hukuki çerçeve serinin ikinci bölümünde ele alınmaktadır.

Anahtar Kelimeler: *kriptoloji, kriptografi, kriptanaliz, AES, ChaCha20, GCM, Ascon, Camellia, SM4, RSA, Diffie-Hellman, eliptik eğri kriptografisi, ECDSA, Ed25519, SHA-256, SHA-3, HMAC, Argon2, yan kanal saldırıları, performans ölçümü, OpenSSL*

ABSTRACT

Cryptology is the discipline formed by cryptography, which designs ciphers to protect information from unauthorised parties, and cryptanalysis, which attempts to break them. Opening a web page, logging into a banking app or filing an e-signed petition to the Turkish judicial network all rely on the same small family of algorithms. This first article of a two-part series examines that family itself. It first explains security goals, Kerckhoffs's principle, Shannon's confusion and diffusion and the notion of security bits, and summarises the history from al-Kindi to the AES competition. It then analyses in technical detail symmetric encryption (DES, 3DES, AES, ChaCha20, modes of operation, the lightweight cipher Ascon and regional standards such as Camellia, ARIA, GOST and SM4), hash

burakolgun.com

functions and message authentication (MD5, SHA-1, SHA-2, SHA-3, BLAKE, HMAC, HKDF, Argon2) and public-key cryptography (Diffie-Hellman, RSA, ECDSA, X25519, Ed25519). Heartbleed, ROCA, the PS3 signing key and the Debian OpenSSL incident illustrate why most real-world breaks stem from implementation errors rather than from the algorithms. The author benchmarked 21 algorithms with OpenSSL 3.5.6 on the same hardware and found that hardware-accelerated AES-GCM processes about 2 GB per second, RSA signing is 23 times slower than ECDSA, and the post-quantum ML-KEM-768 is as fast as X25519. The article closes with six code examples tested by the author. Certificate infrastructure, protocols, the post-quantum transition and the legal framework are covered in the second part of the series.

Keywords: *cryptology, cryptography, cryptanalysis, AES, ChaCha20, GCM, Ascon, Camellia, SM4, RSA, Diffie-Hellman, elliptic curve cryptography, ECDSA, Ed25519, SHA-256, SHA-3, HMAC, Argon2, side-channel attacks, benchmarking, OpenSSL*

1. Giriş

Bir kullanıcı tarayıcısının adres çubuğuna bir banka adresi yazdığında, sayfa daha ekrana gelmeden birkaç yüz milisaniye içinde en az dört ayrı kriptografik işlem tamamlanır. Tarayıcı ile sunucu, daha önce hiç karşılaşmamış olmalarına rağmen ortak bir gizli anahtar üzerinde anlaşır; sunucu kimliğini bir dijital imzayla kanıtlar; iki taraf da el sıkışmanın yolda değiştirilmediğini bir mesaj doğrulama koduyla denetler; sonrasında bütün trafik simetrik bir şifreyle korunur. Bu zincirin herhangi bir halkası zayıfsa geri kalanların sağlamlığı pek bir şey ifade etmez. Kriptolojinin gündelik hayattaki yeri tam olarak budur: görünmez, ama kopması hâlinde her şeyi açığa çıkaran bir altyapı.

Kriptoloji (cryptology) iki kanattan oluşur. **Kriptografi (cryptography)** bilgiyi korumak için şifre ve protokol tasarlar; **kriptanaliz (cryptanalysis)** ise bu tasarımların zayıf noktalarını arar. İki kanat birbirinden ayrı düşünülemez. Bugün güvendiğimiz her algoritma, yıllarca süren açık kriptanaliz çalışmalarından sağ çıktığı için güvenilirdir; güvenlik bir ispattan çok, başarısız saldırıların birikmiş kaydıdır (Katz ve Lindell, 2020). Bu nedenle bir algoritmanın "kırılmamış" olması ile "incelenmemiş" olması arasında büyük fark vardır ve ciddi sistemlerde yalnızca ilk gruptakiler kullanılır.

Uygulamada kullanılan algoritma sayısı şaşırtıcı derecede azdır. İnternet trafiğinin çok büyük bölümü AES ya da ChaCha20 ile şifrelenir, SHA-2 ailesiyle özetlenir, RSA ya da eliptik eğri imzalarıyla doğrulanır ve X25519 benzeri bir eliptik eğri Diffie-Hellman işlemiyle anahtar üretir. Bu yoğunlaşma bilinçli bir tercihtir. Az sayıda algoritmanın çok sayıda göz tarafından incelenmesi, çok sayıda algoritmanın yüzeysel incelenmesinden daha güvenlidir (Bernstein ve Lange, 2017). Öte yandan aynı yoğunlaşma tek bir kırılmanın etkisini de büyütür; 2017'de SHA-1 için ilk pratik çakışmanın yayımlanması, dünyanın dört bir yanındaki sertifika, sürüm kontrol ve belge imzalama sistemlerini aynı anda etkilemiştir (Stevens vd., 2017).

Bu makale iki bölümlük bir serinin ilkidir. Birinci bölüm algoritmaların kendisine, yani nasıl çalıştıklarına, hangi problemlerin zorluğuna dayandıklarına, nasıl kırıldıklarına ve ne kadar hızlı olduklarına odaklanmaktadır. İkinci bölüm ise bu algoritmaların sertifika altyapısı ve protokoller içinde nasıl bir araya getirildiğini, anahtar yönetimini, kuantum sonrası geçişi, şifreleme politikasını ve Türkiye'deki hukuki çerçeveyi ele almaktadır. Bu bölümün geri

burakoguz.com

kalanı şu şekilde düzenlenmiştir. İkinci kısım temel kavramları, üçüncü kısım tarihsel gelişimi özetlemektedir. Dördüncü kısım simetrik şifrelemeyi (hafif ve bölgesel algoritmalar dahil), beşinci kısım özet fonksiyonlarını ve mesaj doğrulamayı, altıncı kısım açık anahtarlı kriptografiyi incelemektedir. Yedinci kısım kriptanaliz yöntemlerini ve gerçek dünyadaki uygulama hatalarını ele almaktadır. Sekizinci kısımda başlıca algoritmaların hızı yazar tarafından aynı donanımda ölçülerek karşılaştırılmakta, dokuzuncu kısımda test edilmiş kod örnekleri verilmektedir. Onuncu kısım sonuçları içermektedir.

2. Kavramsal Çerçeve

2.1. Güvenlik Hedefleri

Kriptografinin sağladığı hizmetler genellikle dört başlık altında toplanır (Menezes vd., 1996):

- **Gizlilik (confidentiality):** Mesajın içeriğini yalnızca yetkili tarafların okuyabilmesi. Şifreleme algoritmaları bu hedefe hizmet eder.
- **Bütünlük (integrity):** Mesajın yolda değiştirilmediğinin anlaşılabilmesi. Özet fonksiyonları ve mesaj doğrulama kodları (Message Authentication Code, MAC) kullanılır.
- **Kimlik doğrulama (authentication):** Mesajın iddia edilen kaynaktan geldiğinin kanıtlanması. Simetrik tarafta MAC, asimetrik tarafta dijital imza bu işi görür.
- **İnkâr edilemezlik (non-repudiation):** Göndericinin sonradan mesajı kendisinin göndermediğini iddia edememesi. Yalnızca dijital imza bu özelliği verir; çünkü MAC anahtarı iki tarafta da bulunduğundan, MAC'li bir mesajı iki taraftan biri üretmiş olabilir.

Son madde hukuk açısından önemlidir. 5070 sayılı Kanun'un güvenli elektronik imzaya elle atılan imzayla aynı hukuki sonucu tanıması, açık anahtarlı imzanın bu inkâr edilemezlik özelliğine dayanır. Bir şifreli e-posta ya da parola korumalı bir dosya ise, ne kadar güçlü şifrelenmiş olursa olsun, tek başına inkâr edilemezlik sağlamaz.

2.2. Kerckhoffs İlkesi ve Shannon'ın Katkısı

Hollandalı dilbilimci Auguste Kerckhoffs 1883'te askeri şifreler için altı ilke yayımlamıştır. Bunların ikincisi bugün hâlâ alanın temel kuralıdır: bir sistemin güvenliği algoritmanın gizli tutulmasına değil, yalnızca anahtarın gizli tutulmasına dayanmalıdır (Kerckhoffs, 1883). Algoritma düşmanın eline geçse bile sistem güvenli kalmalıdır. AES, RSA ve SHA-256'nın tüm iç yapısı herkese açıktır; güvenlik tamamen anahtara emanet edilmiştir. Tersine, algoritmasını gizleyen ticari ürünlerin (örneğin GSM'in A5/1 akış şifresi ya da DVD koruması CSS) büyük kısmı, tersine mühendislikle açığa çıkarıldıktan kısa süre sonra kırılmıştır.

Modern kriptografinin matematiksel temeli Claude Shannon'ın 1949 tarihli "Communication Theory of Secrecy Systems" makalesidir. Shannon bu çalışmada tek kullanımlık şerit (one-time pad) yönteminin mükemmel gizlilik sağladığını ispatlamış, ancak bunun için anahtarın mesaj kadar uzun, tamamen rastgele ve bir kez kullanılır olması gerektiğini göstermiştir (Shannon, 1949). Pratikte bu koşullar sağlanamadığı için Shannon iki tasarım ilkesi önermiştir. **Karıştırma (confusion)**, anahtar ile şifreli metin arasındaki ilişkiyi

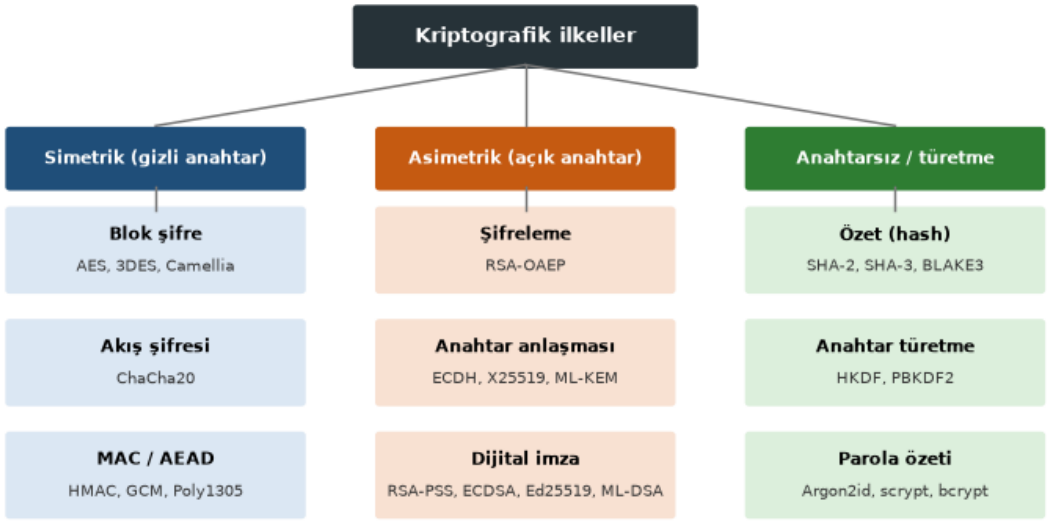
burakolgun.com

olabildiğince karmaşık hâle getirir; **yayılma (diffusion)** ise açık metindeki tek bir bitin değişmesinin şifreli metnin yaklaşık yarısını değiştirmesini sağlar. AES'te S-kutusu karıştırmayı, satır kaydırma ve sütun karıştırma adımları ise yayılmayı sağlar.

2.3. Hesaplamalı Güvenlik ve Güvenlik Biti

Pratikte kullanılan algoritmalar mükemmel değil, **hesaplamalı güvenlik** sağlar: saldırı mümkündür ama gerçekçi kaynaklarla bitirilemeyecek kadar uzun sürer. Bu uzunluk "güvenlik biti" ile ölçülür. 128 bitlik güvenlik, en iyi bilinen saldırının yaklaşık 2^128 işlem gerektirdiği anlamına gelir. Karşılaştırma için: saniyede 10^18 anahtar deneyen hayali bir makine, 2^128 anahtarın yarısını denemek için evrenin yaşının yaklaşık on milyar katı kadar zamana ihtiyaç duyar. NIST bugün 112 bitin altını kabul etmemekte, yeni sistemler için 128 bit ve üzerini önermektedir (Barker, 2020).

Önemli bir ayrıntı, güvenlik bitinin anahtar uzunluğuyla aynı şey olmamasıdır. AES-128'in anahtarı 128 bittir ve güvenliği de yaklaşık 128 bittir. Buna karşılık 3072 bitlik bir RSA anahtarı da yalnızca yaklaşık 128 bitlik güvenlik verir; çünkü RSA'ya karşı kaba kuvvetten çok daha hızlı bir yöntem, sayı cisimleri eleği ile çarpanlara ayırma mevcuttur. Bu ilişki Kısım 6.4'te ayrıntılı olarak ele alınmaktadır. Şekil 1, makalede incelenen kriptografik ilkelleri (primitives) işlevlerine göre sınıflandırmaktadır.



Kaynak: Menezes vd. (1996) ve Boneh ve Shoup (2023) temel alınarak derlenmiştir.

Şekil 1. Kriptografik ilkellerin işlevlerine göre sınıflandırılması

3. Tarihsel Gelişim

Bilinen en eski şifrelerden biri, Suetonius'un aktardığına göre Jül Sezar'ın her harfi alfabede üç harf ileri kaydırarak kullandığı yöntemdir. Bu tür **yerine koyma (substitution)** şifreleri yüzyıllarca kullanıldı. Onları çözmenin sistematik yolunu ilk kez yazıya geçiren kişi, 9. yüzyılda Bağdat'ta yaşamış filozof El-Kindî'dir. El-Kindî'nin şifreli mesajların çözümüne dair risalesi, bir dilde harflerin belirli sıklıklarla geçtiği gözlemine dayanan **frekans analizini** tarif eder (Singh, 1999). Tek alfabeli her yerine koyma şifresi bu yöntemle kırılabilir.

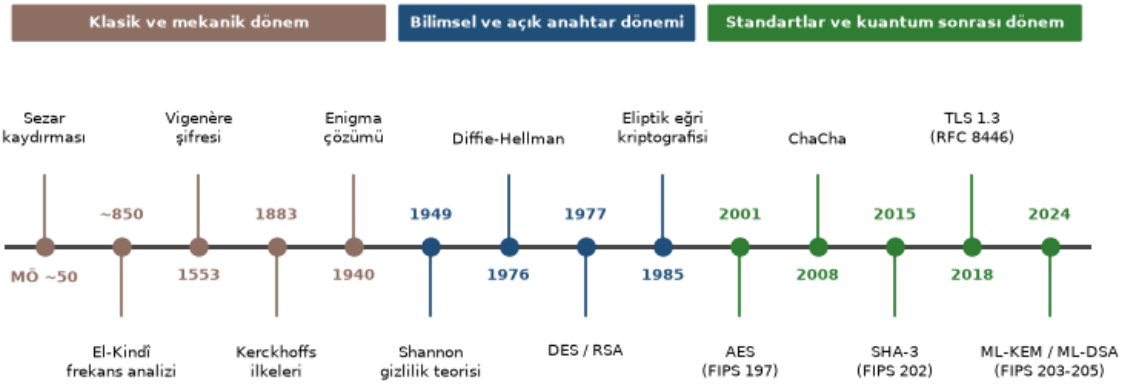
Frekans analizine karşı geliştirilen çok alfabeli şifrelerin en ünlüsü, Giovan Battista Bellaso'nun 1553'te tarif ettiği ve sonradan Blaise de Vigenère'in adıyla anılan yöntemdir. Bir anahtar kelimeye göre her harfte farklı bir kaydırma uygulayan bu şifre "le chiffre indéchiffrable" (çözülemez şifre) olarak ün kazandı; ta ki Friedrich Kasiski 1863'te anahtar uzunluğunu tekrarlanan harf gruplarından bulan yöntemi yayımlayana dek (Kahn, 1996).

Yirminci yüzyılın ilk yarısı mekanik şifre makinelerinin dönemidir. Alman Enigma makinesi, her tuş basışında dönen rotorlar sayesinde her harf için farklı bir yerine koyma uyguluyordu. Polonyalı matematikçi Marian Rejewski 1932'de makinenin iç kablolamasını matematiksel yöntemlerle çıkardı; bu çalışma savaş sırasında Bletchley Park'ta Alan Turing ve ekibinin geliştirdiği elektromekanik "bombe" makinelerine temel oldu (Kahn, 1996). Enigma'nın çözülmesi, bir şifrenin anahtar uzayı ne kadar büyük olursa olsun, işletim hataları ve tahmin edilebilir mesaj kalıpları yüzünden düşebileceğini gösteren ilk büyük örnektir. Bu ders bugün de geçerlidir.

Bilgisayar çağında iki kırılma noktası öne çıkar. İlki 1977'de IBM'in Lucifer tasarımından türetilen **Veri Şifreleme Standardı (Data Encryption Standard, DES)** ile sivil dünyanın ilk kamuya açık şifreleme standardına kavuşmasıdır. İkincisi aynı dönemde yaşanan **açık anahtar devrimidir**. Whitfield Diffie ve Martin Hellman 1976'da, daha önce hiç görüşmemiş iki tarafın güvensiz bir kanal üzerinden ortak bir gizli anahtar üretebileceğini gösterdi (Diffie ve Hellman, 1976). Bir yıl sonra Rivest, Shamir ve Adleman, hem şifreleme hem imza için kullanılabilen RSA algoritmasını önerdi (Rivest vd., 1978). 1997'de kamuoyuna açıklanan belgeler, İngiliz GCHQ'da James Ellis, Clifford Cocks ve Malcolm Williamson'ın benzer fikirlere birkaç yıl önce ulaştığını ama çalışmanın gizli tutulduğunu ortaya koydu.

Sonraki dönemin belirleyici özelliği açık yarışmalardır. NIST 1997'de DES'in yerine geçecek algoritma için uluslararası bir yarışma başlattı; 15 aday arasından Belçikalı Joan Daemen ve Vincent Rijmen'in Rijndael tasarımı 2000'de seçildi ve 2001'de AES olarak standartlaştı (NIST, 2001). Aynı model SHA-3 (2007-2012) ve kuantum sonrası kriptografi (2016-2024) yarışmalarında da uygulandı. Şekil 2 bu gelişimi zaman çizelgesi üzerinde göstermektedir.

burakolgun.com



Kaynak: Kahn (1996), Singh (1999) ve NIST FIPS yayınları temel alınarak derlenmiştir.

Şekil 2. Kriptolojinin tarihsel gelişiminde başlıca kilometre taşları

4. Simetrik Şifreleme

Simetrik şifrelemede şifreleyen ve çözen taraf aynı anahtarı kullanır. Bu algoritmalar hızlıdır; modern bir işlemci çekirdeği AES ile saniyede birkaç gigabayt veri şifreleyebilir. Buna karşılık anahtarın iki tarafa güvenli biçimde ulaştırılması ayrı bir sorundur ve bu sorun açık anahtarlı yöntemlerle çözülür. Gerçek sistemlerde iki yaklaşım birlikte kullanılır: açık anahtarlı kriptografi kısa bir oturum anahtarı üzerinde anlaşmak için, simetrik kriptografi ise asıl veriyi şifrelemek için. Simetrik algoritmalar ikiye ayrılır. **Blok şifreler** sabit uzunluktaki veri bloklarını (AES'te 128 bit) işler; **akış şifreleri** anahtardan uzun bir sözde rastgele bit dizisi üretip veriyle XOR'lar.

4.1. DES ve Üçlü DES

DES, 64 bitlik bloklar üzerinde 16 turlu bir **Feistel ağı** olarak çalışır. Feistel yapısında blok iki yarıya bölünür; her turda bir yarı, anahtara bağlı bir fonksiyondan geçirilip diğer yarıyla XOR'lanır ve yarılar yer değiştirir. Bu yapının güzel yanı, tur fonksiyonunun tersinir olmasına gerek kalmadan tüm şifrenin tersinir olmasıdır (NIST, 1999). DES'in tasarımı sonradan keşfedilen diferansiyel kriptanalize karşı şaşırtıcı derecede dayanıklı çıktı; IBM'in bu saldırıyı 1970'lerde bildiği ve S-kutularını buna göre tasarladığı sonradan Don Coppersmith tarafından açıklandı (Coppersmith, 1994).

DES'in asıl zaafı algoritma değil, 56 bitlik anahtar uzunluğuydu. Electronic Frontier Foundation 1998'de yaklaşık 250 bin dolara mal olan "Deep Crack" makinesiyle bir DES anahtarını 56 saatte buldu; 1999'da distributed.net ile ortak çalışmada süre 22 saat 15 dakikaya indi (EFF, 1998). Geçiş dönemi çözümü olarak DES'i üç kez ve iki ya da üç farklı anahtarla uygulayan **Üçlü DES (3DES)** kullanıldı. 3DES, 64 bitlik blok boyutu yüzünden doğum günü sınırına takıldı: Bhargavan ve Leurent (2016), uzun süreli HTTPS bağlantılarında yaklaşık 785 GB trafik toplanarak çerez çalınabileceğini gösteren **Sweet32** saldırısını yayımladı. NIST 3DES ile şifrelemeye 2023 sonundan itibaren izin vermemektedir (Barker ve Roginsky, 2019).

burakolgun.cc

4.2. AES (Gelişmiş Şifreleme Standardı)

Bugün dünyada en çok kullanılan şifreleme algoritması AES'tir (Advanced Encryption Standard). AES, Feistel yerine **yerine koyma-permütasyon ağı (Substitution-Permutation Network, SPN)** yapısını kullanır. 128 bitlik blok 4x4 baytlık bir durum matrisi olarak düşünülür ve her turda dört işlemden geçer (NIST, 2001; Daemen ve Rijmen, 2002):

- **SubBytes:** Her bayt, GF(2^8) cisminde ters alma işlemine dayanan sabit bir S-kutusundan geçirilir. Algoritmanın tek doğrusal olmayan adımıdır ve karıştırmayı sağlar.
- **ShiftRows:** Matrisin satırları 0, 1, 2 ve 3 bayt sola kaydırılır.
- **MixColumns:** Her sütun sabit bir matrisle çarpılır. ShiftRows ile birlikte iki tur sonunda her çıkış baytı her giriş baytına bağımlı hâle gelir.
- **AddRoundKey:** Durum, anahtar genişletme ile üretilen tur anahtarıyla XOR'lanır.

Tur sayısı anahtar uzunluğuna göre 10 (AES-128), 12 (AES-192) veya 14'tür (AES-256). Yaklaşık çeyrek asırlık yoğun kriptanalize rağmen tam tur AES'e karşı pratik bir saldırı bulunamamıştır. Bilinen en iyi genel saldırı olan biclique yöntemi, AES-128'in anahtarını 2^128 yerine 2^126,1 işlemle bulabilmektedir; yani kaba kuvvete göre yalnızca dört kat hızlıdır ve pratik açıdan anlamsızdır (Bogdanov vd., 2011). Intel'in 2010'da işlemcilerine eklediği AES-NI komut seti ve ARM'ın benzer uzantıları, AES'i hem çok hızlı hem de zamanlama saldırılarına karşı daha dayanıklı hâle getirmiştir; çünkü donanım komutları yazılımdaki tablo aramalarının yarattığı önbellek izlerini bırakmaz.

AES-128 ile AES-256 arasındaki seçim sıkça sorulur. Klasik bilgisayarlara karşı 128 bit fazlasıyla yeterlidir. Kuantum bilgisayarlar için Grover algoritması anahtar aramasını karekköküne indirdiğinden AES-256 tercih edilir; ancak serinin ikinci bölümünde açıklanacağı üzere Grover'ın pratik maliyeti o kadar yüksektir ki AES-128'in de uzun süre güvenli kalması beklenmektedir.

4.3. Çalışma Kipleri ve Doğrulamalı Şifreleme

Bir blok şifre tek başına yalnızca 16 baytı şifreleyebilir. Daha uzun veriler için bir **çalışma kipi (mode of operation)** gerekir ve uygulamadaki hataların önemli bir kısmı algoritmadan değil kip seçiminden doğar. En basit kip olan **ECB (Electronic Codebook)**, her bloğu bağımsız şifreler; aynı açık metin blokları aynı şifreli bloklara dönüştüğü için verinin deseni görünür kalır. Linux maskotu Tux'ın ECB ile şifrelenmiş resminde penguenin hatlarının açıkça seçilebilmesi, kriptografi derslerinin klasik örneğidir.

CBC (Cipher Block Chaining) her bloğu şifrelemeden önce bir önceki şifreli blokla XOR'lar ve bu sorunu çözer; ancak dolgu (padding) doğrulaması yanlış yapıldığında Vaudenay'nin (2002) tarif ettiği **dolgu kâhini (padding oracle)** saldırısına açık hâle gelir. Sunucu "dolgu hatalı" ile "MAC hatalı" durumlarını farklı yanıtlarla ya da farklı sürelerde bildiriyorsa, saldırgan anahtarı bilmeden şifreli metni bayt bayt çözebilir. SSL 3.0'ı fiilen sona erdiren POODLE saldırısı (Möller vd., 2014) bu ailenin en bilinen örneğidir.

Bu deneyimler, gizlilik ve bütünlüğü tek işlemde sağlayan **doğrulamalı şifreleme (Authenticated Encryption with Associated Data, AEAD)** kiplerini standart hâline getirdi.

burakolgun.com

Bugün en yaygın AEAD kipi **GCM (Galois/Counter Mode)**'dur. GCM, sayaç kipiyle (CTR) şifreler ve GF(2^128) üzerinde çarpmaya dayanan GHASH fonksiyonuyla 128 bitlik bir doğrulama etiketi üretir (Dworkin, 2007). Hızlıdır, paralel çalışır ve modern işlemcilerde özel komutlarla (PCLMULQDQ) desteklenir.

GCM'in keskin bir kenarı vardır: aynı anahtarla aynı 96 bitlik tek kullanımlık değer (nonce) iki kez kullanılırsa hem gizlilik hem de bütünlük çöker. Saldırgan iki şifreli metni XOR'layarak açık metinlerin XOR'unu elde eder ve doğrulama anahtarını çıkarıp istediği mesaja geçerli etiket üretebilir. Böck ve arkadaşları (2016), İnternet taramasında nonce tekrarlayan 184 HTTPS sunucusu tespit etmiştir. Nonce yönetiminin zor olduğu senaryolar için nonce tekrarına dayanıklı AES-GCM-SIV kipi RFC 8452 ile standartlaşmıştır (Gueron vd., 2019). Disk şifrelemede ise, sektör başına ek alan bulunmadığından, farklı bir kip olan **XTS** kullanılır (Dworkin, 2010).

4.4. ChaCha20-Poly1305

AES'in yazılımda hızlı ve güvenli çalışması donanım desteğine bağlıdır. AES-NI bulunmayan düşük güçlü cihazlarda tablo tabanlı AES hem yavaş hem de önbellek zamanlaması saldırılarına açıktır. Daniel J. Bernstein'ın 2008'de yayımladığı **ChaCha20** akış şifresi bu boşluğu doldurur (Bernstein, 2008). ChaCha20 yalnızca 32 bitlik toplama, sabit miktarda döndürme ve XOR işlemlerini (Add-Rotate-XOR, ARX) kullanır; bu işlemlerin süresi veriden bağımsız olduğu için zamanlama sızıntısı üretmez. 256 bitlik anahtar ve 96 bitlik nonce ile 20 tur çalışır.

ChaCha20, yine Bernstein'ın tasarladığı Poly1305 doğrulayıcısıyla birleştirilerek ChaCha20-Poly1305 AEAD yapısını oluşturur (Nir ve Langley, 2018). Google 2014'te bu yapıyı Android'deki Chrome bağlantılarında kullanmaya başladı; bugün TLS 1.3'ün üç zorunlu ya da önerilen şifre takımından biri, WireGuard VPN protokolünün tek şifreleme seçeneği ve OpenSSH'ın varsayılan şifrelerinden biridir. Tablo 1 başlıca simetrik algoritmaları karşılaştırmaktadır.

Tablo 1. Başlıca simetrik şifreleme algoritmalarının karşılaştırılması

Algoritma	Yapı	Blok / anahtar (bit)	Tur	Güncel durum
DES	Feistel blok şifre	64 / 56	16	Kırık (1998); kullanılmamalı
3DES (TDEA)	Üçlü Feistel	64 / 112-168	48	Sweet32; 2023 sonrası NIST izni yok
AES-128	SPN blok şifre	128 / 128	10	Güvenli; en yaygın
AES-256	SPN blok şifre	128 / 256	14	Güvenli; kuantuma karşı tercih
ChaCha20	ARX akış şifresi	- / 256	20	Güvenli; mobil ve VPN
Camellia	Feistel blok şifre	128 / 128-256	18-24	Güvenli; Japonya/AB listeleri
RC4	Akış şifresi	- / 40-2048	-	Kırık; TLS'te yasak (RFC 7465)

Not. Kaynak: NIST (2001), Barker ve Roginsky (2019), Nir ve Langley (2018) ve Popov (2015) temel alınarak derlenmiştir.

4.5. Hafif Kriptografi: Ascon

Nesnelerin interneti sensörleri, RFID etiketleri ve tıbbi implantlar gibi cihazlarda birkaç kilobaytlık bellek ve mikrovat düzeyinde enerji bütçesi bulunur. AES bu cihazların bir kısmında çalışabilse de her mesaj için ayrı bir özet fonksiyonu ve MAC taşımak çoğu zaman

mümkün değildir. NIST bu ihtiyaç için 2018'de ayrı bir hafif kriptografi yarışması başlattı ve 7 Şubat 2023'te Avusturyalı bir ekibin tasarladığı **Ascon** ailesini seçtiğini açıkladı (NIST, 2023b). Ascon daha önce, 2019'da sonuçlanan CAESAR yarışmasında da hafif kullanım senaryosunun birinci tercihi olarak belirlenmişti (Dobraunig vd., 2021).

Ascon, Keccak'a benzer bir sünger yapısıdır; ancak 1600 bit yerine beş adet 64 bitlik sözcükten oluşan 320 bitlik küçük bir iç durum kullanır. Tek bir permütasyonla hem doğrulamalı şifreleme hem özetleme yapılabildiği için, aynı kod parçası cihazda birden fazla işi görür. Ağustos 2025'te yayımlanan NIST SP 800-232, dört varyantı standartlaştırmıştır: 128 bit anahtar, 128 bit nonce ve 128 bit etiket kullanan Ascon-AEAD128; Ascon-Hash256; değişken uzunlukta çıktı üreten Ascon-XOF128 ve özelleştirilebilir Ascon-CXOF128 (Sönmez Turan vd., 2025). Türkiye'den de bu alana katkı vardır: Karakoç, Demirci ve Harmancı (2013), yazılımda verimli çalışacak biçimde tasarlanmış 80 bitlik **ITUbee** hafif blok şifresini LightSec 2013 çalıştayında yayımlamıştır.

4.6. Bölgesel ve Ulusal Algoritmalar

AES küresel standart olsa da bazı ülkeler kendi algoritmalarını geliştirmiş ve kamu sistemlerinde bunların kullanımını zorunlu tutmaktadır. Bu tercihin arkasında hem teknik bağımsızlık kaygısı hem de dış kaynaklı bir standardın gizli zayıflık içerebileceği endişesi vardır. Japonya'da Mitsubishi ve NTT'nin geliştirdiği **Camellia** (RFC 3713), Güney Kore'de **ARIA** (RFC 5794) ve **SEED** (RFC 4269), Rusya'da GOST R 34.12-2015 standardındaki **Kuznyechik** blok şifresi (RFC 7801) ile **Streebog** özet fonksiyonu (RFC 6986) bu grubun başlıca örnekleridir (Matsui vd., 2004; Lee vd., 2010; Dolmatov, 2016).

En kapsamlı ulusal ekosistem Çin'e aittir. SM2 (eliptik eğri imza ve anahtar değişimi), SM3 (özet) ve SM4 (128 bitlik blok şifre) algoritmaları GB/T ulusal standartları olarak yayımlanmış, SM4 2021'de ISO/IEC 18033-3 standardına bir ek ile dahil edilmiştir (ISO/IEC, 2021). RFC 8998, bu algoritmaların TLS 1.3'te TLS_SM4_GCM_SM3 gibi şifre takımlarıyla kullanımını tanımlar; ancak belge, bu kullanımın IETF tarafından onaylanmadığını açıkça not eder (Yang, 2021). 1 Ocak 2020'de yürürlüğe giren Çin Kriptografi Kanunu, şifrelemeyi "çekirdek", "genel" ve "ticari" olmak üzere üç kategoriye ayırarak devlet denetimini yasal zemine oturtmuştur (NPC, 2019). Türkiye için bu makalenin taramasında kamuya açık ve standartlaşmış bir ulusal şifreleme algoritmasına rastlanmamıştır; e-imza ve kamu sistemlerinde uluslararası algoritmalar kullanılmaktadır. Tablo 2 bölgesel algoritmaları özetlemektedir.

Tablo 2. Başlıca bölgesel ve ulusal kriptografik algoritmalar

Algoritma	Ülke	Tür	Parametreler	Standart
Camellia	Japonya	Blok şifre (Feistel)	128 bit blok; 128/192/256 bit anahtar	RFC 3713; ISO/IEC 18033-3
ARIA	Güney Kore	Blok şifre (SPN)	128 bit blok; 128/192/256 bit anahtar	RFC 5794; KS X 1213
SEED	Güney Kore	Blok şifre (Feistel)	128 bit blok; 128 bit anahtar	RFC 4269
Kuznyechik	Rusya	Blok şifre (SPN)	128 bit blok; 256 bit anahtar	GOST R 34.12-2015; RFC 7801
Streebog	Rusya	Özet fonksiyonu	256 / 512 bit çıktı	GOST R 34.11-2012; RFC 6986

burakgelen.com

SM4	Çin	Blok şifre	128 bit blok; 128 bit anahtar	GB/T 32907-2016; ISO/IEC 18033-3 Amd 1
SM3	Çin	Özet fonksiyonu	256 bit çıktı	GB/T 32905-2016; ISO/IEC 10118-3
SM2	Çin	ECC imza / anahtar değişimi	256 bit eğri	GB/T 32918; ISO/IEC 14888-3
ITUbee	Türkiye (akademik)	Hafif blok şifre	80 bit blok; 80 bit anahtar	Karakoç vd. (2013)

Not. Kaynak: Matsui vd. (2004), Lee vd. (2005, 2010), Dolmatov (2016), Dolmatov ve Degtyarev (2013), Yang (2021), ISO/IEC (2021) ve Karakoç vd. (2013) temel alınarak derlenmiştir.

5. Özet Fonksiyonları ve Mesaj Doğrulama

5.1. Kriptografik Özet Fonksiyonlarının Özellikleri

Kriptografik özet (hash) fonksiyonu, istenen uzunluktaki bir girdiyi sabit uzunlukta bir çıktıya dönüştürür. SHA-256 bir kelimeyi de, 50 GB'lık bir disk imajını da 256 bitlik bir değere indirger. Bir özet fonksiyonunun kriptografik sayılabilmesi için üç koşulu sağlaması gerekir (Menezes vd., 1996):

- **Ön görüntü direnci:** Bir özet değeri verildiğinde, bu değeri üreten girdiyi bulmak pratikte imkânsız olmalıdır.
- **İkinci ön görüntü direnci:** Bir girdi verildiğinde, aynı özeti üreten başka bir girdi bulunamamalıdır.
- **Çakışma direnci:** Aynı özeti üreten herhangi iki farklı girdi bulunamamalıdır.

Çakışma direnci en zayıf halkadır. Doğum günü paradoksu gereği n bitlik bir özette çakışma bulmak için 2^n değil, yaklaşık 2^(n/2) deneme yeterlidir. Bu yüzden 128 bitlik çıktı veren MD5 en fazla 64 bitlik çakışma güvenliği sunabilir ve bugünün donanımıyla bu bile fazla iyimser bir tahmindir. Adli bilişimde özet değerlerinin delil bütünlüğünü kanıtlamak için kullanılması tam olarak bu özelliklere dayanır: imaj alındığı anda hesaplanan SHA-256 değeri ile mahkemeye sunulan kopyanın değeri eşleşiyorsa, iki kopyanın aynı olduğu kabul edilir.

5.2. MD5 ve SHA-1'in Çöküşü

Ron Rivest'in 1991'de tasarladığı MD5 ile NSA kaynaklı SHA-1 (1995), aynı Merkle-Damgård tasarım anlayışını paylaşır. Xiaoyun Wang ve Hongbo Yu 2005'te MD5 için dakikalar içinde çakışma bulan bir yöntem yayımladı (Wang ve Yu, 2005). Stevens ve arkadaşları (2009), seçilmiş ön ekli MD5 çakışmasıyla tarayıcıların güvendiği sahte bir sertifika otoritesi sertifikası üretti. 2012'de ortaya çıkan Flame casus yazılımı aynı tekniği kullanarak Microsoft adına sahte bir kod imzalama sertifikası oluşturdu ve Windows Update üzerinden yayıldı.

SHA-1 için teorik zayıflıklar 2005'ten beri biliniyordu, ancak ilk gerçek çakışma 2017'de Google ve CWI Amsterdam ekibinin **SHAttered** çalışmasıyla geldi. Ekip, aynı SHA-1 özetine sahip iki farklı PDF dosyası üretmek için yaklaşık 2^63,1 SHA-1 hesaplaması yaptı (Stevens vd., 2017). 2020'de Leurent ve Peyrin, çok daha tehlikeli olan seçilmiş ön ekli çakışmayı yaklaşık 45 bin dolarlık bulut maliyetiyle gerçekleştirdi ve bununla PGP anahtar imzalarının taklit edilebileceğini gösterdi (Leurent ve Peyrin, 2020). NIST, SHA-1'in tüm kullanımlarının 31 Aralık 2030'a kadar sonlandırılmasını istemektedir (NIST, 2022).

burakolgun.com

Adli bilişim açısından bu gelişmelerin anlamı dikkatle okunmalıdır. MD5 ve SHA-1 çakışmaları, saldırganın **iki dosyayı birlikte ve önceden** hazırlamasını gerektirir. Mevcut bir disk imajıyla aynı MD5 özetine sahip ikinci bir imaj üretmek (ikinci ön görüntü saldırısı) bugün hâlâ pratik değildir. Yine de mahkemeye sunulacak raporlarda MD5 yerine ya da onunla birlikte SHA-256 kullanılması, karşı tarafın itirazlarını baştan kesmek için doğru yaklaşımdır.

5.3. SHA-2, SHA-3 ve BLAKE Ailesi

Bugün en yaygın özet fonksiyonu **SHA-256**'dır. SHA-2 ailesi (SHA-224, SHA-256, SHA-384, SHA-512 ve kesik türevleri) 2001'de yayımlandı ve FIPS 180-4 ile güncellendi (NIST, 2015a). SHA-2'ye karşı pratik bir çakışma saldırısı yoktur. Ancak Merkle-Damgård yapısından kaynaklanan bir özelliği vardır: **uzunluk uzatma (length extension)**. H(gizli || mesaj) değerini bilen saldırgan, gizli değeri bilmeden H(gizli || mesaj || ek) değerini hesaplayabilir. Duong ve Rizzo (2009), aynı Merkle-Damgård yapısını paylaşan MD5 ile oluşturulan Flickr API imzalarında bu özelliği kullanarak yetkisiz istek üretebildiklerini gösterdi; SHA-256 de aynı saldırıya yapısal olarak açıktır. Doğru çözüm, özet fonksiyonunu MAC olarak doğrudan kullanmamak ve HMAC yapısına başvurmaktır.

SHA-3, NIST'in 2007-2012 yarışmasını kazanan Keccak algoritmasına dayanır ve 2015'te FIPS 202 olarak yayımlanmıştır (NIST, 2015b; Bertoni vd., 2011). SHA-3, SHA-2'nin yerine geçmek için değil, onunla tamamen farklı bir yapı sunarak yedek oluşturmak için seçilmiştir. Keccak, 1600 bitlik bir iç durumu kullanan **sünger (sponge)** yapısıdır: veri durumun bir kısmına "emdirilir", çıktı ise yine aynı kısımdan "sıkılarak" alınır. Bu yapı uzunluk uzatma saldırısına doğal olarak dayanıklıdır ve SHAKE128/256 ile istenen uzunlukta çıktı üretmeye izin verir. Kuantum sonrası ML-KEM ve ML-DSA standartları iç işlemlerinde SHAKE kullandığından, SHA-3 önümüzdeki yıllarda çok daha sık karşımıza çıkacaktır.

Standart dışı tarafta **BLAKE2** (Saarinen ve Aumasson, 2015) ve **BLAKE3** (O'Connor vd., 2020) öne çıkar. BLAKE2, yazılımda SHA-256'dan belirgin biçimde hızlıdır ve Argon2, WireGuard ile birçok dosya bütünlük aracının içinde kullanılır. BLAKE3 ağaç yapısı sayesinde çok çekirdekli işlemcilerde paralel çalışır. Tablo 3 başlıca özet fonksiyonlarını özetlemektedir.

Tablo 3. Başlıca kriptografik özet fonksiyonları ve güvenlik durumları

Fonksiyon	Yıl	Çıktı (bit)	Yapı	Durum / bilinen saldırı
MD5	1992	128	Merkle-Damgård	Kırık: çakışma saniyeler içinde (Wang ve Yu, 2005)
SHA-1	1995	160	Merkle-Damgård	Kırık: SHAttered (2017), seçilmiş ön ek (2020)
SHA-256	2001	256	Merkle-Damgård	Güvenli; en yaygın; uzunluk uzatmaya açık
SHA-512	2001	512	Merkle-Damgård	Güvenli; 64 bit işlemcide hızlı
SHA3-256	2015	256	Sünger (Keccak)	Güvenli; uzunluk uzatmaya dayanıklı
BLAKE2b	2012	1-512	HAIFA	Güvenli; yazılımda çok hızlı
BLAKE3	2020	Değişken	Merkle ağacı	Güvenli; paralel çalışır

Not. Kaynak: NIST (2015a, 2015b), Stevens vd. (2017), Leurent ve Peyrin (2020) ve Aumasson vd. (2013) temel alınarak derlenmiştir.

kriptoloji.com

5.4. HMAC ve Anahtar Türetme

HMAC (Hash-based Message Authentication Code), bir özet fonksiyonunu gizli anahtarla iki katmanlı biçimde kullanarak MAC üretir: HMAC(K, m) = H((K ⊕ opad) || H((K ⊕ ipad) || m)). Bellare, Canetti ve Krawczyk (1996) tarafından önerilen bu yapı, alttaki özet fonksiyonunun çakışma direnci zayıflasa bile güvenli kalır; HMAC-MD5'in bugün hâlâ doğrudan kırılmamış olmasının nedeni budur (yine de yeni sistemlerde kullanılmamalıdır). HMAC, TLS, IPsec, JSON Web Token (HS256) ve bankacılık API'lerinde imza aracı olarak kullanılır.

Anahtar değişiminden çıkan ham sır doğrudan şifreleme anahtarı olarak kullanılmaz; önce bir **anahtar türetme fonksiyonundan (Key Derivation Function, KDF)** geçirilir. HMAC tabanlı **HKDF** (Krawczyk, 2010; Krawczyk ve Eronen, 2010) iki adımlıdır: "çıkarma" adımı düzensiz dağılımlı girdiyi düzgün bir ana anahtara dönüştürür, "genişletme" adımı bu anahtardan istenen sayıda bağımsız alt anahtar üretir. TLS 1.3'ün tüm anahtar çizelgesi HKDF üzerine kuruludur.

5.5. Parolaların Saklanması

Parolalar şifrelenmez, özetlenir; ama SHA-256 gibi hızlı bir fonksiyonla değil. Hızlı özet fonksiyonları, grafik işlemcilerle saniyede milyarlarca tahmin yapılmasına izin verir. Parola saklamak için bilerek yavaş ve bellek tüketen fonksiyonlar kullanılır. Bunların başlıcaları PBKDF2, bcrypt (1999), scrypt (2009) ve 2015 Parola Özetleme Yarışması'nı kazanan **Argon2**'dir (Biryukov vd., 2016, 2021). Argon2id, her tahmin için onlarca megabayt bellek ayırmayı zorunlu kıldığından GPU ve ASIC ile paralel saldırının maliyetini ciddi ölçüde artırır. OWASP bugün Argon2id için en az 19 MiB bellek, 2 iterasyon ve 1 paralellik; PBKDF2-HMAC-SHA256 için ise en az 600.000 iterasyon önermektedir (OWASP, 2025). Her kullanıcı için rastgele bir **tuz (salt)** eklenmesi, önceden hesaplanmış gökkuşağı tablolarını işe yaramaz kılar.

6. Açık Anahtarlı (Asimetrik) Kriptografi

Açık anahtarlı kriptografide her kullanıcının matematiksel olarak bağlantılı iki anahtarı vardır. Açık anahtar herkesle paylaşılır, özel anahtar yalnızca sahibinde kalır. Açık anahtardan özel anahtarı hesaplamak, altta yatan matematik problemin zorluğu yüzünden pratikte imkânsızdır. Bugün kullanılan klasik algoritmalar iki problem ailesine dayanır: büyük sayıların **çarpanlara ayrılması** (RSA) ve **ayrık logaritma** (Diffie-Hellman, DSA ve eliptik eğri türevleri). Bu iki problemin kuantum bilgisayarla verimli çözülebilmesi, serinin ikinci bölümünde ele alınan geçiş sürecinin nedenidir.

6.1. Diffie-Hellman Anahtar Değişimi

Diffie-Hellman (DH) protokolünde taraflar büyük bir asal p ve bir üreteç g üzerinde anlaşır. Ayşe gizli bir a seçip g^a mod p değerini, Bora gizli bir b seçip g^b mod p değerini gönderir. Her iki taraf da ortak sır olan g^(ab) mod p değerini hesaplayabilir; dinleyen üçüncü kişi ise yalnızca g^a ve g^b'yi gördüğünden, ayrık logaritma problemini çözmeden bu değere ulaşamaz (Diffie ve Hellman, 1976). DH tek başına kimlik doğrulaması sağlamaz; araya giren

burakoksuz.com

saldırgan her iki tarafla ayrı ayrı anahtar değişimi yapabilir. Bu yüzden gerçek protokollerde DH, imza ya da sertifikayla birlikte kullanılır.

Oturum başına yeni (geçici, ephemeral) DH anahtarı üretilmesi **ileri gizlilik (forward secrecy)** özelliğini sağlar: sunucunun uzun vadeli özel anahtarı ileride çalınsa bile, geçmişte kaydedilmiş oturumlar çözülemez. Logjam çalışması (Adrian vd., 2015), birçok sunucunun aynı 1024 bitlik asal sayıları paylaştığını ve devlet düzeyinde kaynaklara sahip bir saldırganın bu birkaç asal için tek seferlik ön hesaplama yaparak çok sayıda VPN ve HTTPS bağlantısını çözebileceğini gösterdi. Aynı çalışmada, ihracat sınıfı 512 bitlik DH'ye düşürme saldırısına en popüler bir milyon alan adının %8,4'ünün açık olduğu tespit edildi.

6.2. RSA

RSA'da iki büyük asal p ve q seçilir, n = p·q modülü hesaplanır. Açık anahtar (n, e), özel anahtar d'dir ve e·d ≡ 1 mod φ(n) koşulunu sağlar. Şifreleme c = m^e mod n, çözme m = c^d mod n işlemiyle yapılır. İmzada roller yer değiştirir. d'yi bulmak için n'yi çarpanlarına ayırmak yeterlidir ve bilinen en hızlı klasik yöntem olan **genel sayı cisimleri eleği (General Number Field Sieve, GNFS)** alt-üstel bir süre gerektirir (Rivest vd., 1978). Açık olarak çarpanlarına ayrılmış en büyük RSA meydan okuma sayısı, 2020'de yaklaşık 2.700 çekirdek-yıllık hesaplamayla çözülen 829 bitlik RSA-250'dir (Boudot vd., 2020). 2048 bitlik bir modülün klasik yöntemlerle çarpanlara ayrılması bu ölçeğin çok ötesindedir.

"Ders kitabı RSA" olarak adlandırılan çıplak formül güvenli değildir; deterministiktir ve çarpımsal yapısı nedeniyle manipüle edilebilir. Bu yüzden şifrelemede **OAEP**, imzada **PSS** dolgu şemaları kullanılır (Moriarty vd., 2016; RFC 8017). Eski PKCS #1 v1.5 dolgusu ise Bleichenbacher'ın (1998) uyarlamalı seçilmiş şifreli metin saldırısına açıktır. Bu saldırı 20 yıl sonra ROBOT adıyla geri döndü: Böck, Somorovsky ve Young (2018), Alexa listesinin ilk 100 sitesinin yaklaşık üçte birinin, aralarında Facebook ve PayPal'ın da bulunduğu, bu 1998 saldırısının türevlerine açık olduğunu gösterdi. TLS 1.3'ün RSA ile anahtar taşımayı tamamen kaldırmasının bir nedeni budur.

6.3. Eliptik Eğri Kriptografisi

Neal Koblitz (1987) ve Victor Miller (1985) birbirinden bağımsız olarak, ayrık logaritma probleminin sonlu bir cisim üzerindeki eliptik eğri noktalarının oluşturduğu grupta kurulabileceğini önerdi. Eliptik eğri kriptografisinin (Elliptic Curve Cryptography, ECC) avantajı, bu grupta alt-üstel saldırıların bilinmemesidir. Bu sayede 256 bitlik bir ECC anahtarı, 3072 bitlik bir RSA anahtarıyla aynı güvenliği verir; anahtarlar, imzalar ve hesaplama süresi buna göre küçülür.

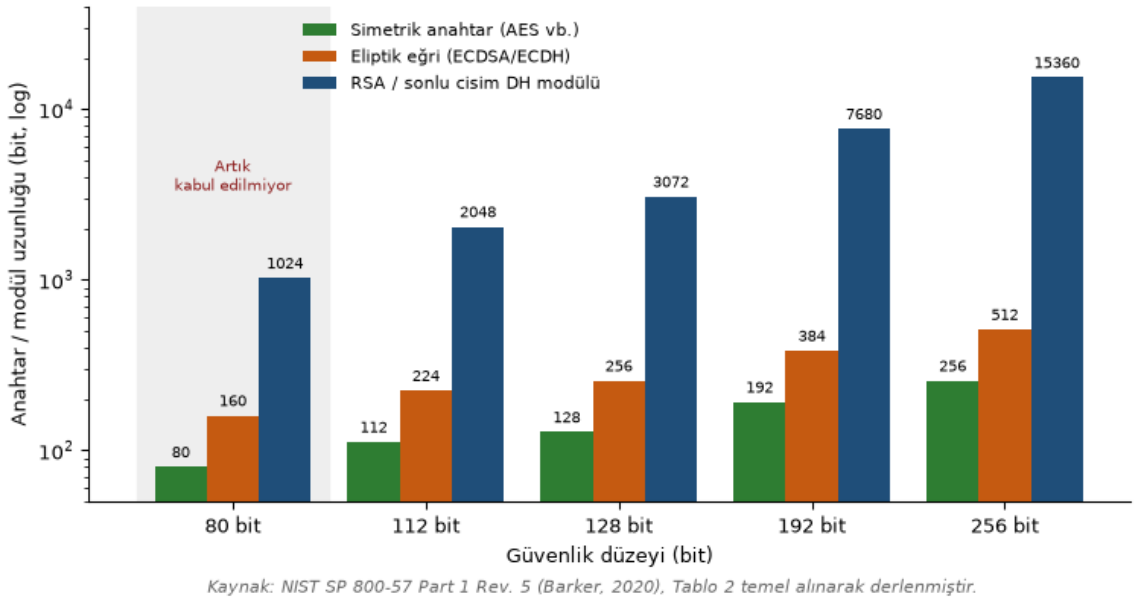
Yaygın kullanılan eğriler üç gruba ayrılır. NIST'in P-256 ve P-384 eğrileri, sertifika altyapısında ve devlet sistemlerinde standarttır (NIST, 2023a). Bitcoin ve Ethereum **secp256k1** eğrisini kullanır. Daniel Bernstein'ın tasarladığı **Curve25519** ise uygulama hatalarına karşı dayanıklı olacak biçimde seçilmiş parametreleriyle en hızlı yayılan eğri olmuştur. Bu eğri üzerindeki anahtar anlaşması **X25519** (Langley vd., 2016), aynı eğrinin eşdeğer Edwards biçimi üzerindeki imza şeması ise **Ed25519**'dur (Bernstein vd., 2012; Josefsson ve Liusvaara, 2017). X25519 bugün TLS 1.3, SSH, Signal ve WireGuard'da varsayılan anahtar anlaşmasıdır; Ed25519 ise 2023'te FIPS 186-5 ile NIST onayı almıştır.

burakgun.com

ECDSA'nın tehlikeli bir özelliği vardır: her imza için rastgele seçilen k değeri iki farklı mesajda tekrar kullanılırsa özel anahtar basit bir cebirle hesaplanabilir. fail0verflow grubu 2010'da Sony'nin PlayStation 3 yazılım imzalama anahtarını tam olarak bu yolla, Sony'nin k değerini sabit tutması sayesinde elde etti. 2013'te Android'deki hatalı rastgele sayı üreteci nedeniyle bazı Bitcoin cüzdanlarından para çalınması da aynı hatadan kaynaklandı. Ed25519 ve RFC 6979 (Pornin, 2013), k değerini mesaj ve özel anahtardan deterministik türeterek bu riski ortadan kaldırır.

6.4. Eşdeğer Güvenlik Düzeyleri

Farklı algoritma ailelerini karşılaştırmanın doğru yolu güvenlik bitini esas almaktır. NIST'in anahtar yönetimi rehberi SP 800-57, simetrik, eliptik eğri ve RSA/DH anahtar uzunlukları arasındaki eşdeğerliği tablo hâlinde verir (Barker, 2020). Şekil 3'te görüldüğü gibi güvenlik düzeyi arttıkça RSA'nın modül uzunluğu doğrusal değil, çok daha hızlı büyür: 128 bitten 256 bite çıkmak simetrik anahtarı iki katına, ECC anahtarını iki katına, RSA modülünü ise beş katına çıkarır. Uzun vadeli sistemlerde RSA yerine ECC tercih edilmesinin temel gerekçesi budur. Tablo 4, başlıca asimetrik algoritmaları özetlemektedir.



Kaynak: NIST SP 800-57 Part 1 Rev. 5 (Barker, 2020), Tablo 2 temel alınarak derlenmiştir.

Şekil 3. Aynı güvenlik düzeyi için gereken simetrik, ECC ve RSA anahtar uzunlukları

Tablo 4. Başlıca açık anahtarlı algoritmalar

Algoritma	Dayandığı problem	Tipik boyut	Başlıca kullanım	Kuantuma karşı
RSA (OAEP/PSS)	Çarpanlara ayırma	2048-4096 bit	Sertifika, e-imza, kod imzalama	Kırılır (Shor)
DH (FFDHE)	Sonlu cisimde ayrık log.	2048-4096 bit	IPsec, eski TLS	Kırılır (Shor)
ECDH / X25519	ECC ayrık logaritma	256 bit	TLS 1.3, SSH, Signal, WireGuard	Kırılır (Shor)
ECDSA (P-256)	ECC ayrık logaritma	256 bit	Web PKI, kimlik kartları	Kırılır (Shor)
ECDSA (secp256k1)	ECC ayrık logaritma	256 bit	Bitcoin, Ethereum	Kırılır (Shor)
Ed25519	ECC (Edwards) ayrık log.	256 bit	SSH anahtarları, yazılım imzası	Kırılır (Shor)
ML-KEM-768	Modül kafes (MLWE)	1.184 B açık	Hibrit TLS/SSH anahtar	Dayanıklı

		ank.	değişimi	
ML-DSA-65	Modül kafes (MLWE/MSIS)	1.952 B açık ank.	Kuantum sonrası imza	Dayanıklı

Not. Kaynak: Barker (2020), NIST (2023a, 2024a, 2024b), Langley vd. (2016) ve Bernstein vd. (2012) temel alınarak derlenmiştir.

7. Kriptanaliz ve Uygulama Hataları

7.1. Saldırı Modelleri

Kriptanalizde saldırganın elindeki bilgiye göre farklı modeller tanımlanır. En zayıf saldırgan yalnızca şifreli metinleri görür (ciphertext-only). Daha güçlü modellerde saldırgan bazı açık metin-şifreli metin çiftlerini bilir (known-plaintext), kendi seçtiği metinleri şifreletebilir (chosen-plaintext) veya kendi seçtiği şifreli metinleri çözdürebilir (chosen-ciphertext). Modern algoritmalardan en güçlü modelde bile güvenli kalmaları beklenir; AEAD kiplerinin ve RSA-OAEP'in varlık nedeni budur (Katz ve Lindell, 2020). Diferansiyel ve doğrusal kriptanaliz gibi matematiksel yöntemler, algoritma tasarımında hâlâ ilk sınav olarak kullanılır.

7.2. Yan Kanal Saldırıları

Matematiksel olarak kusursuz bir algoritma, çalıştığı donanım aracılığıyla bilgi sızdırabilir. Paul Kocher 1996'da RSA ve DH işlemlerinin süresinin özel anahtarın bitlerine bağlı olarak değiştiğini ve bu farkın ölçülerek anahtarın çıkarılabileceğini gösterdi (Kocher, 1996). Üç yıl sonra aynı ekip, akıllı kartların anlık güç tüketiminden anahtar çıkaran **diferansiyel güç analizini** yayımladı. Önbellek zamanlaması, elektromanyetik yayılım, akustik ve hata enjeksiyonu saldırıları da aynı aileye girer. Karşı önlem, "sabit zamanlı" (constant-time) kod yazmak, yani işlem süresinin ve bellek erişim deseninin gizli veriye bağlı olmamasını sağlamaktır.

7.3. Gerçek Dünyadan Örnekler

Son yirmi yılın büyük kriptografik olaylarına bakıldığında, AES, SHA-256 veya doğru kullanılan RSA'nın matematiksel olarak kırıldığı tek bir vaka yoktur. Kırılan hemen her zaman uygulama, rastgelelik ya da anahtar yönetimidir:

- **Debian OpenSSL (2008):** Bir paket bakımcısı, bellek denetim aracının uyarısını susturmak için entropi toplayan satırı kaldırdı. İki yıl boyunca Debian ve Ubuntu sistemlerinde üretilen SSH ve SSL anahtarlarının entropisi süreç numarasına indi; olası anahtar sayısı yalnızca 32.767 idi (CVE-2008-0166).
- **Heartbleed (2014):** OpenSSL'in TLS "heartbeat" uzantısındaki sınır denetimi hatası, saldırganın sunucu belleğinden 64 KB'a kadar veri okumasına ve bu yolla özel anahtarları, parolaları ve oturum çerezlerini elde etmesine izin verdi (CVE-2014-0160).
- **Dual_EC_DRBG (2013):** NIST SP 800-90A'daki bu rastgele sayı üretecinin, sabit parametreleri bilen tarafın çıktıyı tahmin edebileceği bir arka kapı içerdiği Snowden belgeleriyle gündeme geldi; NIST algoritmayı 2014'te standarttan çıkardı. 2015'te Juniper güvenlik duvarlarında aynı mekanizmanın kötüye kullanıldığı ortaya çıktı (Checkoway vd., 2016).

burakolgun.com

- **ROCA (2017):** Infineon'un RSA anahtar üretim kütüphanesi, asal sayıları özel bir yapıda seçtiği için açık anahtardan özel anahtarın hesaplanmasına izin veriyordu. Estonya, yaklaşık 750 bin elektronik kimlik kartının sertifikasını askıya almak zorunda kaldı (Nemec vd., 2017).

Bu örneklerin ortak dersi şudur: kriptografik kodu kurumun kendisinin yazması neredeyse hiçbir zaman doğru tercih değildir. OpenSSL, BoringSSL, libsodium veya işletim sisteminin sunduğu kriptografi arayüzleri gibi yaygın incelenmiş kütüphaneler kullanılmalı, güncellemeleri zamanında uygulanmalı ve rastgele sayılar yalnızca işletim sisteminin kriptografik üretecinden (/dev/urandom, getrandom, BCryptGenRandom) alınmalıdır.

8. Performans Karşılaştırması: Yazarın Ölçümleri

8.1. Yöntem

Literatürdeki hız karşılaştırmaları farklı donanım, derleyici ve kütüphane sürümleriyle yapıldığı için birbiriyle doğrudan karşılaştırılamaz. Bu nedenle bu çalışmada başlıca algoritmalar aynı makinede ve aynı kütüphaneyle yazar tarafından ölçülmüştür. Ölçümler 28 Eylül 2026'da, Intel Core i7-8550U işlemcili (dört çekirdek, AES-NI ve AVX2 destekli, SHA donanım uzantısı yok) bir dizüstü bilgisayarda, Windows 11 üzerinde OpenSSL 3.5.6 sürümünün yerleşik "speed" aracıyla yapılmıştır. 8 Nisan 2025'te yayımlanan ve Nisan 2030'a kadar desteklenecek olan OpenSSL 3.5, üç NIST kuantum sonrası standardını (ML-KEM, ML-DSA, SLH-DSA) doğrudan destekleyen ve TLS'te X25519MLKEM768'i varsayılan olarak sunan ilk uzun süreli destek sürümüdür (OpenSSL Project, 2025). Bu sayede klasik ve kuantum sonrası algoritmalar aynı ortamda karşılaştırılabilmiştir.

Simetrik algoritmalar 16 KB'lık bloklar üzerinde, iki saniyelik yedi tekrar hâlinde ölçülmüş ve medyan değer raporlanmıştır. Asimetrik algoritmalar tek çekirdekte, iki saniyelik üç tekrar hâlinde ölçülmüştür. Dizüstü bilgisayarlarda işlemci frekansı sıcaklık ve güç yönetimine göre değiştiğinden tekrarlar arasında belirgin sapma gözlenmiştir; Şekil 4'teki hata çubukları bu aralığı göstermektedir. Mutlak değerler bu donanıma özgüdür; asıl anlamlı olan algoritmalar arasındaki oranlardır.

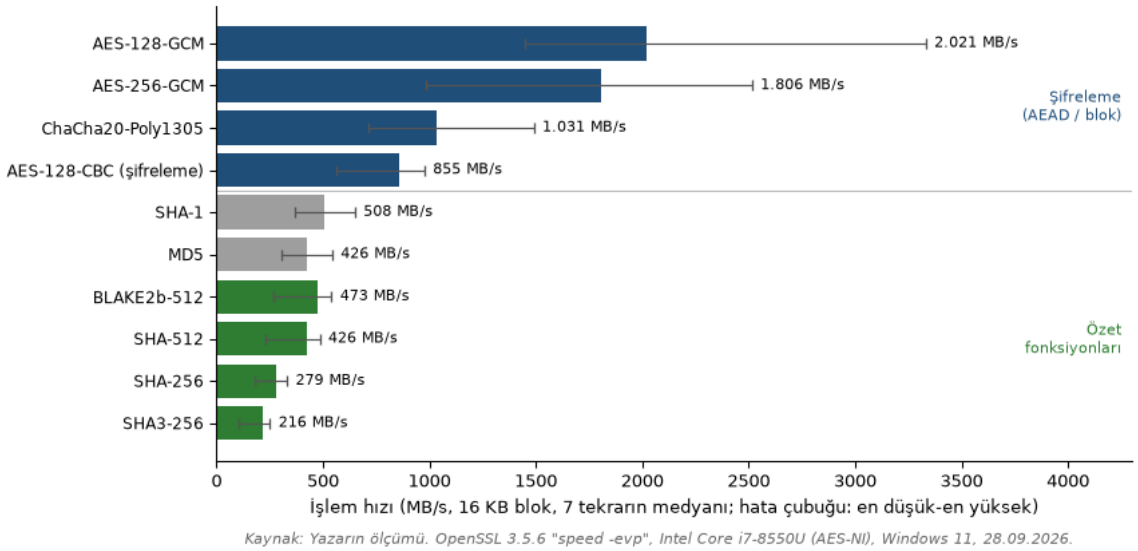
8.2. Simetrik Algoritmalar ve Özet Fonksiyonları

Şekil 4'te görüldüğü gibi AES-128-GCM saniyede yaklaşık 2 GB ile en hızlı algoritmadır. AES-256-GCM, dört fazladan tur nedeniyle biraz daha yavaştır (1,8 GB/s). Donanım desteği olmayan ChaCha20-Poly1305 bu işlemcide AES-GCM'in yaklaşık yarısı hızında (1,0 GB/s) çalışmaktadır; ancak bu değer bile tek bir çekirdeğin 8 Gbit/s'lik bir bağlantıyı doldurmasına yeter. AES-128-CBC şifrelemenin (855 MB/s) GCM'den belirgin biçimde yavaş olması, CBC'nin her bloğun bir öncekinin sonucunu beklemesi yüzünden paralel çalışamamasından kaynaklanır. Bu sonuçlar, TLS 1.3'ün CBC'yi kaldırıp AEAD kiplerine geçmesinin performans açısından da doğru bir karar olduğunu göstermektedir.

Özet fonksiyonlarında dikkat çekici bir sonuç, 64 bitlik işlemcide SHA-512'nin (426 MB/s) SHA-256'dan (279 MB/s) daha hızlı olmasıdır. SHA-512 her turda 64 bitlik sözcüklerle çalışıp tur başına iki kat veri işlediği için, SHA donanım uzantısı bulunmayan işlemcilerde bu sonuç beklenen bir durumdur. SHA3-256 (216 MB/s) bu donanımda en yavaş

burakolgun.com

seçenektir. Kırık olan MD5 ve SHA-1'in daha hızlı olması, adli bilişim yazılımlarında bu algoritmaların hâlâ tercih edilmesinin bir nedenidir; ancak aradaki fark, delil bütünlüğü gibi güvenlik açısından kritik bir işlemde zayıf algoritmayı haklı çıkaracak büyüklükte değildir. Terabaytlık bir disk imajında sınırlayıcı etken genellikle özet fonksiyonu değil, diskin okuma hızıdır.



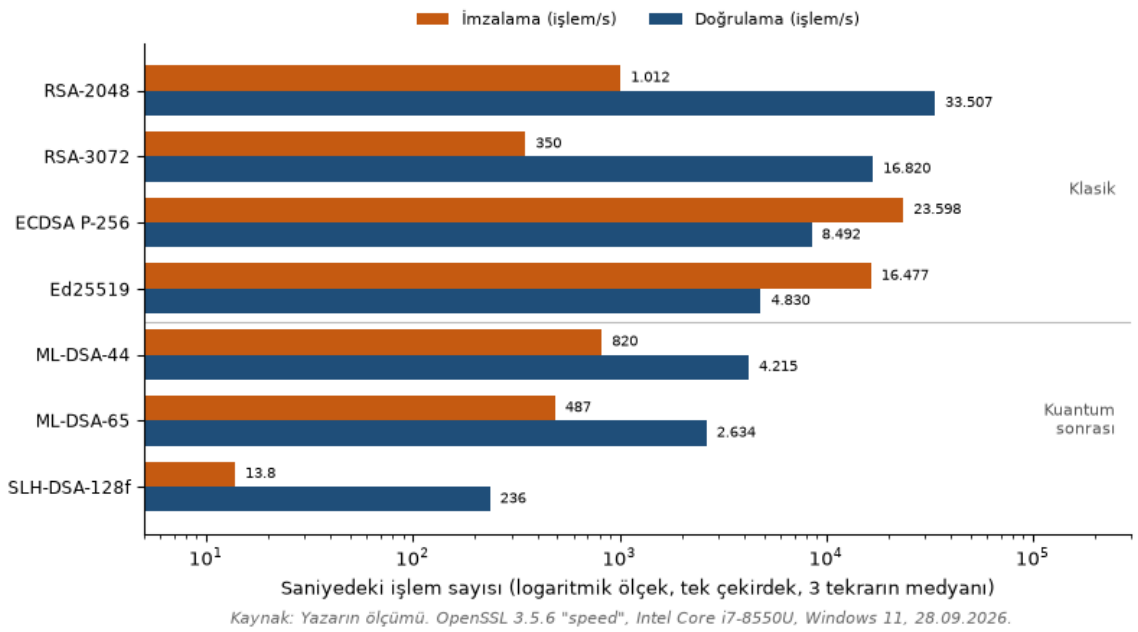
Şekil 4. Simetrik şifreleme ve özet fonksiyonlarının ölçülen işlem hızları

8.3. Açık Anahtarlı ve Kuantum Sonrası Algoritmalar

Şekil 5 ve Tablo 5, asimetrik algoritmalarda çok daha büyük farklar olduğunu göstermektedir. RSA-2048 ile imzalama saniyede yaklaşık 1.000 işlemle ECDSA P-256'dan (23.598) yaklaşık 23 kat yavaştır; aynı 128 bitlik güvenliği veren RSA-3072'ye geçildiğinde fark 67 kata çıkar. Buna karşılık RSA doğrulama çok hızlıdır (33.507 işlem/s), çünkü küçük bir açık üs (e = 65537) kullanır. Sertifika zincirlerinde imza bir kez atılıp milyonlarca kez doğrulandığı için RSA'nın web PKI'daki uzun ömrü kısmen bu asimetriyle açıklanabilir.

Kuantum sonrası algoritmalar için sonuçlar, yaygın kanının aksine hesaplama maliyetinin sorun olmadığını göstermektedir. ML-KEM-768 ile kapsülleme saniyede 20.626, kapsül açma 10.406 işlemle X25519 anahtar anlaşmasıyla (16.677) aynı düzeydedir ve RSA-2048 ile özel anahtar işlemine (538) göre yaklaşık yirmi kat hızlıdır. Tarayıcıların kullandığı hibrit X25519MLKEM768 grubu, iki işlemi birlikte yaptığı hâlde saniyede 5.000'in üzerinde el sıkışmaya karşılık gelen hıza ulaşmaktadır. Kuantum sonrası geçişin asıl maliyeti, serinin ikinci bölümünde açıklandığı gibi işlemci süresinde değil, bant genişliğindedir. İmzalarda tablo farklıdır: ML-DSA-65 ile imzalama bu ölçümde ECDSA'dan yaklaşık 50 kat yavaş, SLH-DSA-128f ise saniyede yalnızca 14 imza üretebilmektedir. Bu sonuç OpenSSL 3.5'in ML-DSA uygulamasına özgüdür ve farklı kütüphanelerde değişebilir; yine de SLH-DSA'nın yalnızca yazılım güncellemesi imzalama gibi seyrek ve uzun ömürlü imzalar için uygun olduğu açıktır.

Bulaklığı



Şekil 5. Klasik ve kuantum sonrası imza algoritmalarının ölçülen işlem hızları

Tablo 5. Anahtar anlaşması ve imza algoritmalarının ölçülen hızları (işlem/saniye, tek çekirdek)

Algoritma	Anahtar üretimi	İmza / kapsülleme	Doğrulama / kapsül açma	Not
RSA-2048	8,1	1.012	33.507	Özel anahtarla çözme: 538
RSA-3072	2,5	350	16.820	128 bit güvenlik
ECDSA P-256	-	23.598	8.492	Web PKI standardı
Ed25519	-	16.477	4.830	SSH, yazılım imzası
ECDH P-256	-	11.569 (anlaşma)	-	Klasik anahtar değişimi
X25519	-	16.677 (anlaşma)	-	TLS 1.3 varsayılanı
ML-KEM-768	12.709	20.626	10.406	FIPS 203
X25519MLKEM768	6.284	5.472	7.090	Tarayıcı hibrit grubu
ML-DSA-44	4.286	820	4.215	FIPS 204, kategori 2
ML-DSA-65	2.629	487	2.634	FIPS 204, kategori 3
SLH-DSA-SHA2-128f	282	13,8	236	FIPS 205, hızlı varyant

Not. Kaynak: Yazarın ölçümü (OpenSSL 3.5.6, Intel Core i7-8550U, Windows 11, 28.09.2026; üç tekrarın medyanı). Değerler donanıma ve kütüphane sürümüne göre değişir.

9. Uygulamalı Örnekler

Bu bölümdeki örnekler, önceki bölümlerde anlatılan kavramların gerçek kodda nasıl göründüğünü göstermek amacıyla verilmiştir. Python örnekleri yaygın kullanılan cryptography kütüphanesinin 49.0.0 sürümüyle, komut satırı örnekleri OpenSSL 3.5.6 ile yazar tarafından çalıştırılarak test edilmiştir. Örneklerde algoritmaların yalnızca bu kütüphanelerin üst düzey arayüzleri üzerinden kullanıldığına dikkat edilmelidir; Kısım 7.3'te vurgulandığı gibi kriptografik ilkelleri sıfırdan kodlamak güvenli bir yaklaşım değildir.

Örnek 1, AES-256-GCM ile doğrulamalı şifrelemeyi göstermektedir. Şifrelenmeyen ama doğrulanan ek veri (associated data) olarak bir dosya numarası kullanılmıştır; bu değer değiştirildiğinde çözme işlemi InvalidTag hatasıyla reddedilir. Şifreli çıktı, 17 baytlık açık metin ile 16 baytlık doğrulama etiketinin toplamı olan 33 bayttır.

Örnek 1: AES-256-GCM ile doğrulamalı şifreleme (Python)

```
import os
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
anahtar = AESGCM.generate_key(bit_length=256)
aes = AESGCM(anahtar)
nonce = os.urandom(12)                # 96 bit; her mesajda YENİ olmalı
ek_veri = b"dosya-no:2026/125"        # şifrelenmez ama doğrulanır
sifreli = aes.encrypt(nonce, b"Gizli rapor metni", ek_veri)
print(len(sifreli))                   # 33 = 17 bayt metin + 16 bayt etiket
print(aes.decrypt(nonce, sifreli, ek_veri))   # b"Gizli rapor metni"
# Ek veri değiştirilirse çözme reddedilir:
aes.decrypt(nonce, sifreli, b"dosya-no:2026/126")   # -> InvalidTag
```

Örnek 2, iki tarafın X25519 ile ortak sır üretmesini ve bu sırrın Kısım 5.4'te anlatılan HKDF ile 256 bitlik bir oturum anahtarına dönüştürülmesini göstermektedir. TLS 1.3 el sıkışmasının çekirdeği bu iki adımdan oluşur.

Örnek 2: X25519 anahtar anlaşması ve HKDF ile anahtar türetme (Python)

```
from cryptography.hazmat.primitives.asymmetric.x25519 import X25519PrivateKey
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
from cryptography.hazmat.primitives import hashes
ayse = X25519PrivateKey.generate()
bora = X25519PrivateKey.generate()
# Taraflar yalnızca açık anahtarlarını (32 bayt) değiş tokuş eder
sir_a = ayse.exchange(bora.public_key())
sir_b = bora.exchange(ayse.public_key())
assert sir_a == sir_b                 # aynı ortak sır
oturum = HKDF(algorithm=hashes.SHA256(), length=32, salt=None,
              info=b"ornek oturum anahtari").derive(sir_a)
```

Örnek 3, Ed25519 ile bir belgenin imzalanmasını ve doğrulanmasını göstermektedir. Belgede tek bir karakterin değiştirilmesi imzanın geçersiz sayılması için yeterlidir. Bilirkişi raporlarının ve dijital delillerin bütünlüğünün kanıtlanmasında kullanılan mantık budur.

Örnek 3: Ed25519 ile dijital imza (Python)

```
from cryptography.hazmat.primitives.asymmetric.ed25519 import Ed25519PrivateKey
from cryptography.exceptions import InvalidSignature
ozel = Ed25519PrivateKey.generate()
acik = ozel.public_key()
belge = b"Bilirkisi raporu, sayfa 1-24"
imza = ozel.sign(belge)               # 64 bayt
acik.verify(imza, belge)              # geçerli: istisna yok
try:
    acik.verify(imza, belge.replace(b"24", b"25"))
except InvalidSignature:
    print("değiştirilmiş belge reddedildi")
```

Örnek 4, Kısım 4.3'te anlatılan nonce tekrarı hatasının sonucunu somut olarak göstermektedir. Aynı anahtar ve aynı nonce ile şifrelenen iki mesajın şifreli metinleri XOR'landığında, açık metinlerin XOR'u elde edilir. Saldırgan mesajlardan birini biliyor ya da tahmin edebiliyorsa (burada ilk mesaj), diğerini anahtarı hiç öğrenmeden okuyabilir. Örnekte program ikinci ödeme talimatını eksiksiz olarak geri elde etmiştir.

Örnek 4: GCM'de nonce tekrarının gizliliği nasıl bozduğu (Python)

```
p1 = b"Odeme: 1000 TL, alici A"      # saldırganın bildiği mesaj
p2 = b"Odeme: 9000 TL, alici B"      # gizli mesaj
sabit_nonce = os.urandom(12)          # HATA: iki mesajda aynı nonce
```

```
c1 = aes.encrypt(sabit_nonce, p1, None)[:-16]     # etiketi at
c2 = aes.encrypt(sabit_nonce, p2, None)[:-16]
x = bytes(a ^ b for a, b in zip(c1, c2))          # = p1 XOR p2
print(bytes(a ^ b for a, b in zip(x, p1)))
# Çıktı: b"Odeme: 9000 TL, alici B"   -> anahtar olmadan çözüldü
```

Örnek 5, parolanın OWASP'ın önerdiği asgari Argon2id parametreleriyle (19 MiB bellek, 2 iterasyon, 1 paralellik) özetlenmesini göstermektedir. Her kullanıcı için rastgele üretilen tuz, özetle birlikte veritabanında saklanır.

Örnek 5: Argon2id ile parola özetleme (Python)

```
from cryptography.hazmat.primitives.kdf.argon2 import Argon2id
tuz = os.urandom(16)
def kdf(): return Argon2id(salt=tuz, length=32, iterations=2,
                           lanes=1, memory_cost=19 * 1024)   # KiB
ozet = kdf().derive(b"parola123!")     # veritabanına: tuz + ozet
kdf().verify(b"parola123!", ozet)      # doğru parola: istisna yok
```

Örnek 6, OpenSSL 3.5 komut satırıyla ML-KEM-768 anahtar çifti üretilmesini, açık anahtarla bir sırrın kapsüllenmesini ve özel anahtarla açılmasını göstermektedir. Testte kapsül dosyası FIPS 203'te belirtildiği gibi tam 1.088 bayt, paylaşılan sır 32 bayt çıkmış ve iki tarafın sırları birebir eşleşmiştir. Son satırda aynı dosyanın SHA-256 özetinin Windows'un yerleşik certutil aracıyla hesaplanması gösterilmiştir; adli incelemelerde delil dosyalarının bütünlük değeri bu tür bağımsız araçlarla çapraz doğrulanabilir.

Örnek 6: OpenSSL 3.5 ile ML-KEM-768 ve SHA-256 bütünlük denetimi (komut satırı)

```
# Alıcı: anahtar çifti üretir ve açık anahtarı yayımlar
openssl genpkey -algorithm ML-KEM-768 -out mlkem_ozel.pem
openssl pkey -in mlkem_ozel.pem -pubout -out mlkem_acik.pem
# Gönderen: açık anahtarla 32 baytlık sır üretir ve kapsüller (1.088 bayt)
openssl pkeyutl -encap -pubin -inkey mlkem_acik.pem \
        -out kapsul.bin -secret sir_gonderen.bin
# Alıcı: kapsülü özel anahtarla açar
openssl pkeyutl -decap -inkey mlkem_ozel.pem -in kapsul.bin -secret sir_alici.bin
cmp sir_gonderen.bin sir_alici.bin && echo "sırlar aynı"
# Bütünlük değeri: iki bağımsız araç aynı sonucu vermelidir
openssl dgst -sha256 kapsul.bin
certutil -hashfile kapsul.bin SHA256
```

10. Sonuç

Kriptoloji, Sezar'ın harf kaydırmasından kafes tabanlı kuantum sonrası algoritmalara uzanan uzun bir yol kat etmiştir. Bu yolun her aşamasında aynı döngü yaşanmıştır: bir tasarım ortaya çıkar, bir süre güvenli kabul edilir, kriptanaliz onu zayıflatır ve yerini yenisi alır. DES anahtar uzunluğu yüzünden, MD5 ve SHA-1 çakışmalar yüzünden, RC4 istatistiksel sapmalar yüzünden, 3DES ise 64 bitlik blok boyutu yüzünden terk edilmiştir. Bugün kullanılan AES, ChaCha20, SHA-2, SHA-3, HMAC ve eliptik eğri algoritmalarına karşı ise klasik bilgisayarlarla pratik bir saldırı bilinmemektedir.

Bu bölümdeki ölçümler, algoritma seçiminin artık büyük ölçüde performans kaygısıyla sınırlanmadığını göstermiştir. Donanım desteğiyle AES-GCM tek çekirdekte saniyede yaklaşık 2 GB veri işlemekte, donanım desteği olmayan ChaCha20-Poly1305 bile 8 Gbit/s'lik

bir bağlantıyı doldurmaktadır. Açık anahtarlı tarafta eliptik eğriler RSA'ya göre hem daha kısa anahtar hem de çok daha hızlı imza sunmakta, kuantum sonrası ML-KEM ise X25519 ile aynı hızda çalışmaktadır. Dolayısıyla eski algoritmaları "hızlı olduğu için" korumanın teknik bir gerekçesi kalmamıştır.

Belki de bu bölümden çıkan en önemli sonuç, gerçek dünyadaki kırılmaların neredeyse tamamının algoritmalardan değil, onların nasıl kullanıldığından kaynaklanmasıdır. Tekrar kullanılan bir nonce, silinmiş bir entropi satırı, denetlenmemiş bir sınır ya da hatalı üretilmiş bir asal sayı, en güçlü algoritmayı da işe yaramaz kılabilir. Örnek 4'te gösterildiği gibi, tek bir nonce tekrarı AES-256 ile şifrelenmiş bir ödeme talimatını anahtar olmadan okunur hâle getirmektedir. Algoritmaların bu şekilde bir araya getirildiği sertifika altyapısı, protokoller, anahtar yönetimi ve kuantum sonrası geçiş, serinin ikinci bölümünde ele alınmaktadır.

Kaynakça

Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J. A., Heninger, N., Springall, D., Thomé, E., Valenta, L., VanderSloot, B., Wustrow, E., Zanella-Béguelin, S., & Zimmermann, P. (2015). Imperfect forward secrecy: How Diffie-Hellman fails in practice. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 5-17). https://doi.org/10.1145/2810103.2813707

Aumasson, J.-P., Neves, S., Wilcox-O'Hearn, Z., & Winnerlein, C. (2013). BLAKE2: Simpler, smaller, fast as MD5. In Applied Cryptography and Network Security (ACNS 2013), LNCS 7954 (pp. 119-135). Springer. https://doi.org/10.1007/978-3-642-38980-1_8

Barker, E. (2020). Recommendation for key management: Part 1 - General (NIST Special Publication 800-57 Part 1 Rev. 5). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-57pt1r5

Barker, E., & Roginsky, A. (2019). Transitioning the use of cryptographic algorithms and key lengths (NIST Special Publication 800-131A Rev. 2). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-131Ar2

Bellare, M., Canetti, R., & Krawczyk, H. (1996). Keying hash functions for message authentication. In Advances in Cryptology - CRYPTO '96, LNCS 1109 (pp. 1-15). Springer. https://doi.org/10.1007/3-540-68697-5_1

Bernstein, D. J. (2008). ChaCha, a variant of Salsa20. In Workshop Record of SASC 2008: The State of the Art of Stream Ciphers. https://cr.yp.to/chacha/chacha-20080128.pdf

Bernstein, D. J., & Lange, T. (2017). Post-quantum cryptography. Nature, 549(7671), 188-194. https://doi.org/10.1038/nature23461

Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., & Yang, B.-Y. (2012). High-speed high-security signatures. Journal of Cryptographic Engineering, 2(2), 77-89. https://doi.org/10.1007/s13389-012-0027-1

Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. (2011). The Keccak reference (Version 3.0). https://keccak.team/files/Keccak-reference-3.0.pdf

Bhargavan, K., & Leurent, G. (2016). On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 456-467). https://doi.org/10.1145/2976749.2978423

buraolgun.com

Biryukov, A., Dinu, D., & Khovratovich, D. (2016). Argon2: New generation of memory-hard functions for password hashing and other applications. In 2016 IEEE European Symposium on Security and Privacy (pp. 292-302). https://doi.org/10.1109/EuroSP.2016.31

Biryukov, A., Dinu, D., Khovratovich, D., & Josefsson, S. (2021). Argon2 memory-hard function for password hashing and proof-of-work applications (RFC 9106). Internet Engineering Task Force. https://doi.org/10.17487/RFC9106

Bleichenbacher, D. (1998). Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Advances in Cryptology - CRYPTO '98, LNCS 1462 (pp. 1-12). Springer. https://doi.org/10.1007/BFb0055716

Böck, H., Somorovsky, J., & Young, C. (2018). Return of Bleichenbacher's oracle threat (ROBOT). In 27th USENIX Security Symposium (pp. 817-849). USENIX Association.

Böck, H., Zauner, A., Devlin, S., Somorovsky, J., & Jovanovic, P. (2016). Nonce-disrespecting adversaries: Practical forgery attacks on GCM in TLS. In 10th USENIX Workshop on Offensive Technologies (WOOT 16). USENIX Association.

Bogdanov, A., Khovratovich, D., & Rechberger, C. (2011). Biclique cryptanalysis of the full AES. In Advances in Cryptology - ASIACRYPT 2011, LNCS 7073 (pp. 344-371). Springer. https://doi.org/10.1007/978-3-642-25385-0_19

Boneh, D., & Shoup, V. (2023). A graduate course in applied cryptography (Version 0.6). https://toc.cryptobook.us/

Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., & Zimmermann, P. (2020). Comparing the difficulty of factorization and discrete logarithm: A 240-digit experiment. In Advances in Cryptology - CRYPTO 2020, LNCS 12171 (pp. 62-91). Springer. https://doi.org/10.1007/978-3-030-56880-1_3

Checkoway, S., Maskiewicz, J., Garman, C., Fried, J., Cohney, S., Green, M., Heninger, N., Weinmann, R.-P., Rescorla, E., & Shacham, H. (2016). A systematic analysis of the Juniper Dual EC incident. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 468-479). https://doi.org/10.1145/2976749.2978395

Coppersmith, D. (1994). The Data Encryption Standard (DES) and its strength against attacks. IBM Journal of Research and Development, 38(3), 243-250. https://doi.org/10.1147/rd.383.0243

Daemen, J., & Rijmen, V. (2002). The design of Rijndael: AES - The Advanced Encryption Standard. Springer. https://doi.org/10.1007/978-3-662-04722-4

Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. IEEE Transactions on Information Theory, 22(6), 644-654. https://doi.org/10.1109/TIT.1976.1055638

Dobraunig, C., Eichlseder, M., Mendel, F., & Schläffer, M. (2021). Ascon v1.2: Lightweight authenticated encryption and hashing. Journal of Cryptology, 34(3), Article 33. https://doi.org/10.1007/s00145-021-09398-9

Dolmatov, V. (Ed.). (2016). GOST R 34.12-2015: Block cipher "Kuznyechik" (RFC 7801). Internet Engineering Task Force. https://doi.org/10.17487/RFC7801

Dolmatov, V., & Degtyarev, A. (2013). GOST R 34.11-2012: Hash function (RFC 6986). Internet Engineering Task Force. https://doi.org/10.17487/RFC6986

Duong, T., & Rizzo, J. (2009). Flickr's API signature forgery vulnerability (Advisory MOCB-01). https://www.ioactive.com/wp-content/uploads/2012/08/flickr_api_signature_forgery.pdf

buraforgun.com

Dworkin, M. (2007). Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC (NIST Special Publication 800-38D). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-38D

Dworkin, M. (2010). Recommendation for block cipher modes of operation: The XTS-AES mode for confidentiality on storage devices (NIST Special Publication 800-38E). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-38E

EFF. (1998). Cracking DES: Secrets of encryption research, wiretap politics and chip design. Electronic Frontier Foundation / O'Reilly.

Gueron, S., Langley, A., & Lindell, Y. (2019). AES-GCM-SIV: Nonce misuse-resistant authenticated encryption (RFC 8452). Internet Engineering Task Force. https://doi.org/10.17487/RFC8452

ISO/IEC. (2021). ISO/IEC 18033-3:2010/Amd 1:2021 Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers - Amendment 1: SM4. International Organization for Standardization.

Josefsson, S., & Liusvaara, I. (2017). Edwards-curve digital signature algorithm (EdDSA) (RFC 8032). Internet Engineering Task Force. https://doi.org/10.17487/RFC8032

Kahn, D. (1996). The codebreakers: The comprehensive history of secret communication from ancient times to the Internet (Rev. ed.). Scribner.

Karakoç, F., Demirci, H., & Harmancı, A. E. (2013). ITUbee: A software oriented lightweight block cipher. In Lightweight Cryptography for Security and Privacy (LightSec 2013), LNCS 8162 (pp. 16-27). Springer. https://doi.org/10.1007/978-3-642-40392-7_2

Katz, J., & Lindell, Y. (2020). Introduction to modern cryptography (3rd ed.). CRC Press. https://doi.org/10.1201/9781351133036

Kerckhoffs, A. (1883). La cryptographie militaire. Journal des sciences militaires, 9, 5-38, 161-191.

Koblitz, N. (1987). Elliptic curve cryptosystems. Mathematics of Computation, 48(177), 203-209. https://doi.org/10.1090/S0025-5718-1987-0866109-5

Kocher, P. C. (1996). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Advances in Cryptology - CRYPTO '96, LNCS 1109 (pp. 104-113). Springer. https://doi.org/10.1007/3-540-68697-5_9

Krawczyk, H. (2010). Cryptographic extraction and key derivation: The HKDF scheme. In Advances in Cryptology - CRYPTO 2010, LNCS 6223 (pp. 631-648). Springer. https://doi.org/10.1007/978-3-642-14623-7_34

Krawczyk, H., & Eronen, P. (2010). HMAC-based extract-and-expand key derivation function (HKDF) (RFC 5869). Internet Engineering Task Force. https://doi.org/10.17487/RFC5869

Langley, A., Hamburg, M., & Turner, S. (2016). Elliptic curves for security (RFC 7748). Internet Engineering Task Force. https://doi.org/10.17487/RFC7748

Lee, H. J., Lee, S. J., Yoon, J. H., Cheon, D. H., & Lee, J. I. (2005). The SEED encryption algorithm (RFC 4269). Internet Engineering Task Force. https://doi.org/10.17487/RFC4269

Lee, J., Lee, J., Kim, J., Kwon, D., & Kim, C. (2010). A description of the ARIA encryption algorithm (RFC 5794). Internet Engineering Task Force. https://doi.org/10.17487/RFC5794

Leurent, G., & Peyrin, T. (2020). SHA-1 is a shambles: First chosen-prefix collision on SHA-1 and application to the PGP web of trust. In 29th USENIX Security Symposium (pp. 1839-1856). USENIX Association.

burakolgun.com

Matsui, M., Nakajima, J., & Moriai, S. (2004). A description of the Camellia encryption algorithm (RFC 3713). Internet Engineering Task Force. https://doi.org/10.17487/RFC3713

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of applied cryptography. CRC Press.

Miller, V. S. (1985). Use of elliptic curves in cryptography. In Advances in Cryptology - CRYPTO '85, LNCS 218 (pp. 417-426). Springer. https://doi.org/10.1007/3-540-39799-X_31

Möller, B., Duong, T., & Kotowicz, K. (2014). This POODLE bites: Exploiting the SSL 3.0 fallback [Security advisory]. Google. https://www.openssl.org/~bodo/ssl-poodle.pdf

Moriarty, K., Kaliski, B., Jonsson, J., & Rusch, A. (2016). PKCS #1: RSA cryptography specifications version 2.2 (RFC 8017). Internet Engineering Task Force. https://doi.org/10.17487/RFC8017

Nemec, M., Sýs, M., Švenda, P., Klinec, D., & Matyáš, V. (2017). The return of Coppersmith's attack: Practical factorization of widely used RSA moduli. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1631-1648). https://doi.org/10.1145/3133956.3133969

Nir, Y., & Langley, A. (2018). ChaCha20 and Poly1305 for IETF protocols (RFC 8439). Internet Engineering Task Force. https://doi.org/10.17487/RFC8439

NIST. (1999). Data Encryption Standard (DES) (FIPS PUB 46-3). National Institute of Standards and Technology.

NIST. (2001). Advanced Encryption Standard (AES) (FIPS PUB 197). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.197

NIST. (2015a). Secure Hash Standard (SHS) (FIPS PUB 180-4). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.180-4

NIST. (2015b). SHA-3 standard: Permutation-based hash and extendable-output functions (FIPS PUB 202). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.202

NIST. (2022, 15 Aralık). NIST retires SHA-1 cryptographic algorithm [Basın duyurusu]. https://www.nist.gov/news-events/news/2022/12/nist-retires-sha-1-cryptographic-algorithm

NIST. (2023a). Digital Signature Standard (DSS) (FIPS PUB 186-5). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.186-5

NIST. (2023b, 7 Şubat). NIST selects 'lightweight cryptography' algorithms to protect small devices [Basın duyurusu]. https://www.nist.gov/news-events/news/2023/02/nist-selects-lightweight-cryptography-algorithms-protect-small-devices

NIST. (2024a). Module-lattice-based key-encapsulation mechanism standard (FIPS PUB 203). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.203

NIST. (2024b). Module-lattice-based digital signature standard (FIPS PUB 204). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.204

NPC. (2019). Cryptography Law of the People's Republic of China. National People's Congress. http://www.npc.gov.cn/englishnpc/c2759/c23934/202009/t20200929_384279.html

O'Connor, J., Aumasson, J.-P., Neves, S., & Wilcox-O'Hearn, Z. (2020). BLAKE3: One function, fast everywhere [Teknik belirtim]. https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf

OpenSSL Project. (2025, 8 Nisan). OpenSSL 3.5 final release. https://openssl-library.org/post/2025-04-08-openssl-35-final-release/

burakolgun.com

OWASP. (2025). Password storage cheat sheet. OWASP Cheat Sheet Series. https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

Popov, A. (2015). Prohibiting RC4 cipher suites (RFC 7465). Internet Engineering Task Force. https://doi.org/10.17487/RFC7465

Pornin, T. (2013). Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA) (RFC 6979). Internet Engineering Task Force. https://doi.org/10.17487/RFC6979

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126. https://doi.org/10.1145/359340.359342

Saarinen, M.-J., & Aumasson, J.-P. (2015). The BLAKE2 cryptographic hash and message authentication code (MAC) (RFC 7693). Internet Engineering Task Force. https://doi.org/10.17487/RFC7693

Shannon, C. E. (1949). Communication theory of secrecy systems. Bell System Technical Journal, 28(4), 656-715. https://doi.org/10.1002/j.1538-7305.1949.tb00928.x

Singh, S. (1999). The code book: The science of secrecy from ancient Egypt to quantum cryptography. Fourth Estate.

Sönmez Turan, M., McKay, K. A., Chang, D., Kang, J., & Kelsey, J. (2025). Ascon-based lightweight cryptography standards for constrained devices: Authenticated encryption, hash, and extendable output functions (NIST Special Publication 800-232). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-232

Stevens, M., Bursztein, E., Karpman, P., Albertini, A., & Markov, Y. (2017). The first collision for full SHA-1. In Advances in Cryptology - CRYPTO 2017, LNCS 10401 (pp. 570-596). Springer. https://doi.org/10.1007/978-3-319-63688-7_19

Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D. A., & de Weger, B. (2009). Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In Advances in Cryptology - CRYPTO 2009, LNCS 5677 (pp. 55-69). Springer. https://doi.org/10.1007/978-3-642-03356-8_4

Vaudenay, S. (2002). Security flaws induced by CBC padding: Applications to SSL, IPSEC, WTLS... In Advances in Cryptology - EUROCRYPT 2002, LNCS 2332 (pp. 534-545). Springer. https://doi.org/10.1007/3-540-46035-7_35

Wang, X., & Yu, H. (2005). How to break MD5 and other hash functions. In Advances in Cryptology - EUROCRYPT 2005, LNCS 3494 (pp. 19-35). Springer. https://doi.org/10.1007/11426639_2

Yang, P. (2021). ShangMi (SM) cipher suites for TLS 1.3 (RFC 8998). Internet Engineering Task Force. https://doi.org/10.17487/RFC8998

burakdsn.com