BYTERIA

26/04/2025

mobile app sec. training

Ahmethan Gültekin – Mehmet Faris Acar

www.byterialab.com

About Us

> Ahmethan Gültekin

> founder / mobile security researcher at Byteria

> consulting and product development

> public / private trainings

byterialab.com
ahmethan@byterialab.com

About Us

> Mehmet Faris Acar

> mobile security researcher at Byteria

> mobile security researching

byterialab.com
mehmet.faris@byterialab.com

Contents

> android internals

> android native internals

> use of reverse engineering tools

> detection mechanisms (frida,xposed e.g)

> some hooks and analysis

> detection bypass techniques

> native side detections

> crypto methods

> ???

All presentation and lab files;

> https://github.com/byterialab/raconf-online-bootcamp-2025

Android Internals

> Android linux çekirdeğini kullanan bir işletim sistemidir

> projeler .apk dosya uzantısında paketlenir.

> her uygulamanın kendine ait klasörü, bu klasörlerin de kendi içinde permissionları vardır.

> uygulama DEX (dalvik executable) bytecode formatında çalışır

> yeni cihazlarda ART (Android Runtime),
eski cihazlarda DVM (Dalvik Virtual Machine) kullanılır

Android Internals

> DVM (dalvik virtual machine):

> eski versiyonlar için kullanılır.

> uygulamalar **DEX** formatında çalıştırılır.

> **JIT** (Just-In-Time) derleme kullanır. bytecode'u çalışma anında **makine koduna** dönüştürür.

> ART (Android Runtime):

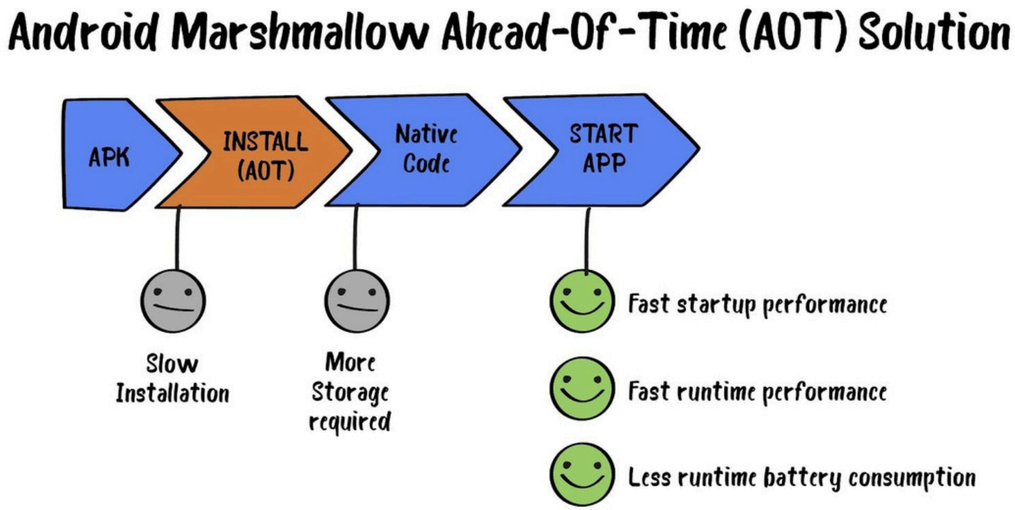
> yeni versiyonlar için kullanılır. **Lollipop** (5.0) verisyonu ile gelmiştir.

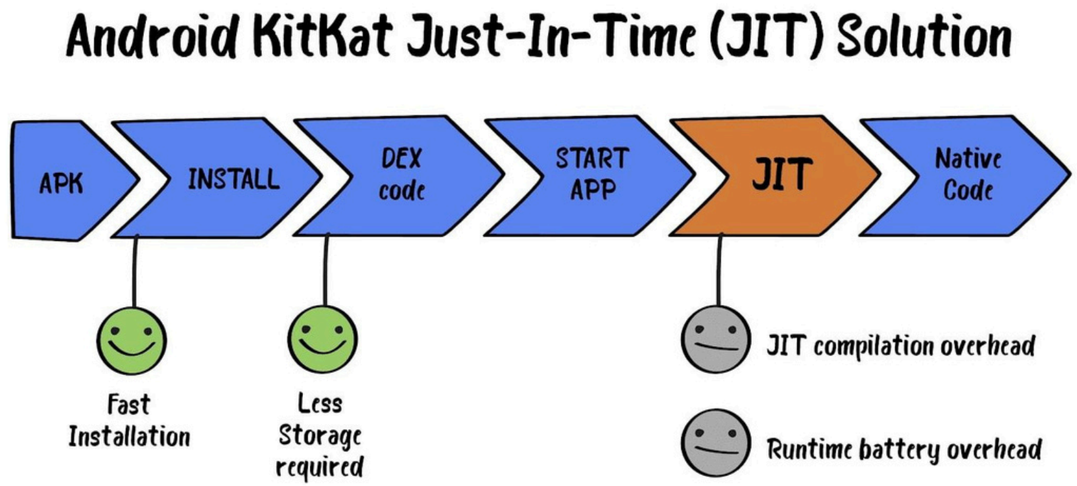
> uygulamalar DEX formatında çalıştırılır.

> **AOT** (Ahead-Of-Time) derleme kullanır. DEX bytecode'u yükleme / kurulum anında **makine koduna** dönüştürür.

> sık kullanılan kod parçaları önbelleklenir ve daha performanslı çalışır.

AOT vs JIT


Android Marshmallow Ahead-Of-Time (AOT) Solution
APK
INSTALL (AOT)
Native Code
START APP
Slow Installation
More Storage required
Fast startup performance
Fast runtime performance
Less runtime battery consumption


Android KitKat Just-In-Time (JIT) Solution
APK
INSTALL
DEX code
START APP
JIT
Native Code
Fast Installation
Less Storage required
JIT compilation overhead
Runtime battery overhead

ART vs Dalvik

> Depolama Alanı:

> ART, **AOT** yaklaşımını kullandığı için **daha fazla** depolama alanına ihtiyaç duyar.

(önbellekleme, geçici derleme dosyaları)

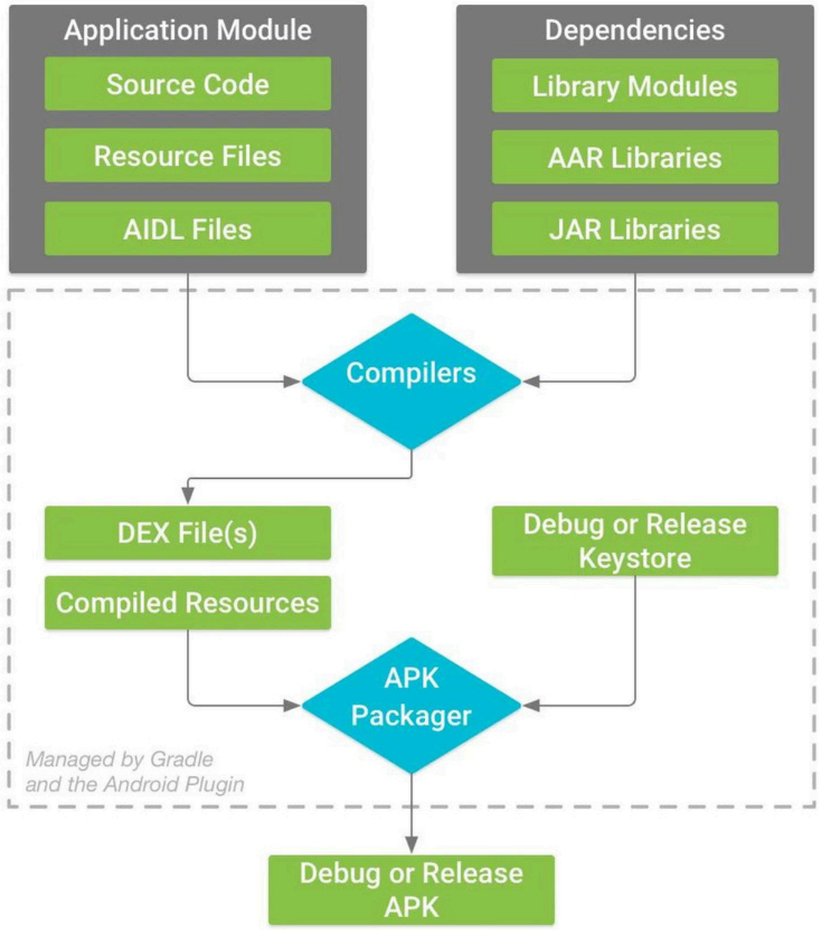
> Dalvik, **JIT** yaklaşımını kullandığı için kod runtimeda derlenir. kod kalıcı olarak saklanmaz, bellekten yürütülür.

> Önyükleme Süresi (booting time):

> **ART**, kurulum esnasında derleme yapar. bundan dolayı boot süresi daha **uzundur**.

> Dalvik, **JIT** yaklaşımını kullandığı için runtimeda derleme yapar. boot süresi daha **kısadır**.

App compilation process



Smali

> android’e özel geliştirilmiş java’ya benzer bir dil

> java kodunun dex bytecode’a dönüşmeden önceki hali

```java
public class Calculation {
    public int add(int a, int b) {
        return a + b;
    }
}
```

java code

```
.class public LCalculation;
.super java/lang/Object

.method public constructor <init>()V
    .registers 1
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
    return
.end method

.method public add(II)I
    .registers 3
    .parameter "a"
    .parameter "b"

    .prologue
    .line 4
    add-int/2addr v0, v1
    return v0
.end method
```

smali code

APK File Structure

lib: projede kullanılan C/C++ librarylerinin olduğu klasör

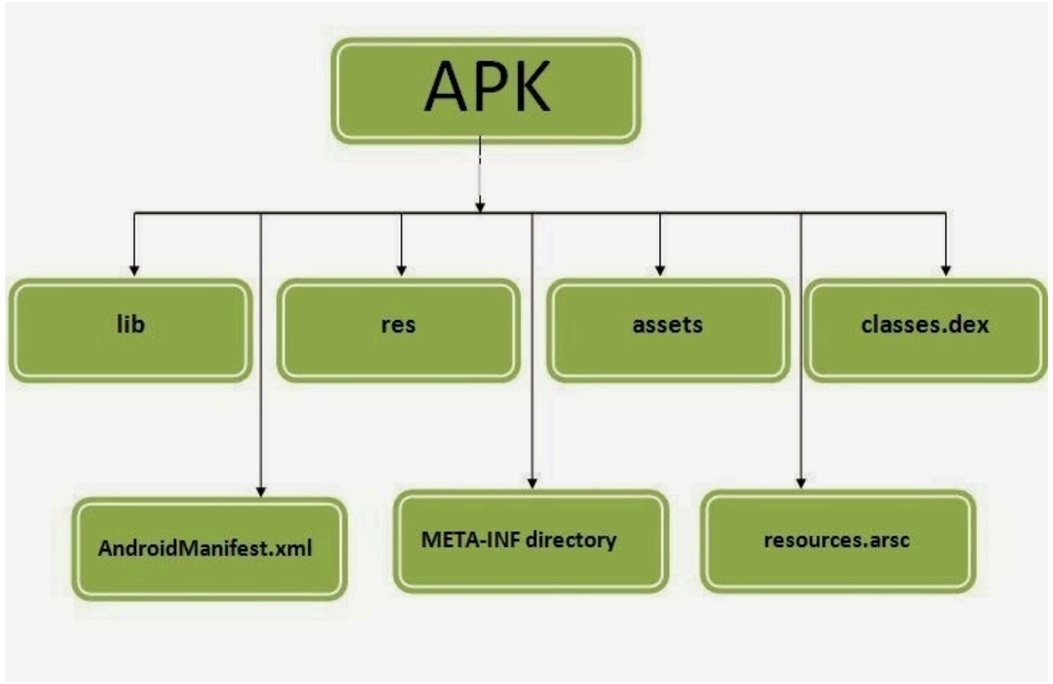
res: projedeki ui,strings, renk tanımlarını içeren klasör .xml formatında veri tutar.

assets: uygulamanın sahip olduğu tüm assetleri içeren klasör

classes.dex: java kodunun derlenip dex bytecode formatına dönüştürülmüş hali

META-INF: apk'nın imza ve sertifika bilgilerini içeren klasör

AndroidManifest.xml: uygulamanın tüm permission ve activity - receiver bilgilerinin bulunduğu dosya



ADB

> android cihaz ile ana makine arasında köprü sağlayan tool

> cihaz ile ilgili bir çok işlemi yapabilir
(install, shell, file push/pull vs.)

> Android Studio ile birlikte gelir (platform-tools)

Android Security Model

> permissions:

> kullanıcılar, uygulamaların erişebileceği verilere ve özelliklere izin verir

> izinler, uygulamaların sadece gerekli ve istenilen kaynaklara erişimini sağlar.

> hassas verilere erişim kullanıcı kontrolündedir

> sandboxing:

> her uygulama, kendi izole edilmiş kullanıcı kimliğinde (UID) çalışır ve kendine özel dizinleri vardır

> uygulamalar birbirlerinin verilerine ve sistem kaynaklarına erişemez.

> izolasyon güvenliğini sağlar ve SELinux ile entegreli çalışır.

Android Security Model

> SELinux (Security-Enhanced Linux) :

> zorunlu erişim kontrolü sağlar, sistem seviyesinde güvenlik politikalarını uygular

> uygulamaların ve sistem süreçlerinin kaynaklara erişimini denetler.

> güvenliği çekirdek seviyesinde artırır.

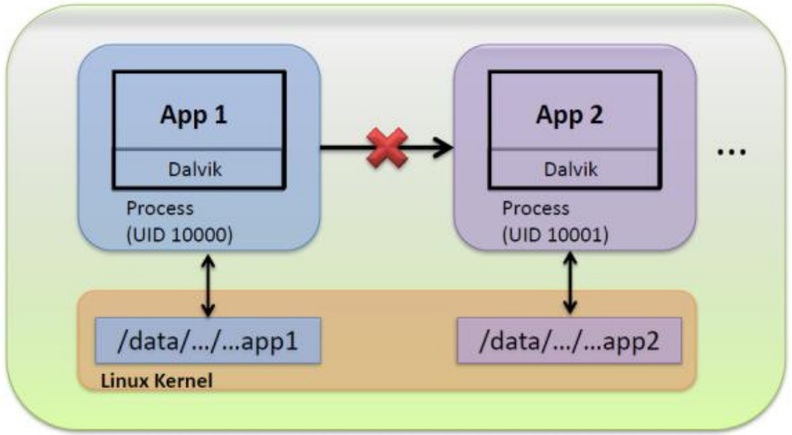
> Enforcing vs Permissive Mode

> Enforcing, SELinux politikalarını katı bir şekilde uygular ve izin verilmeyen işlemleri engeller.

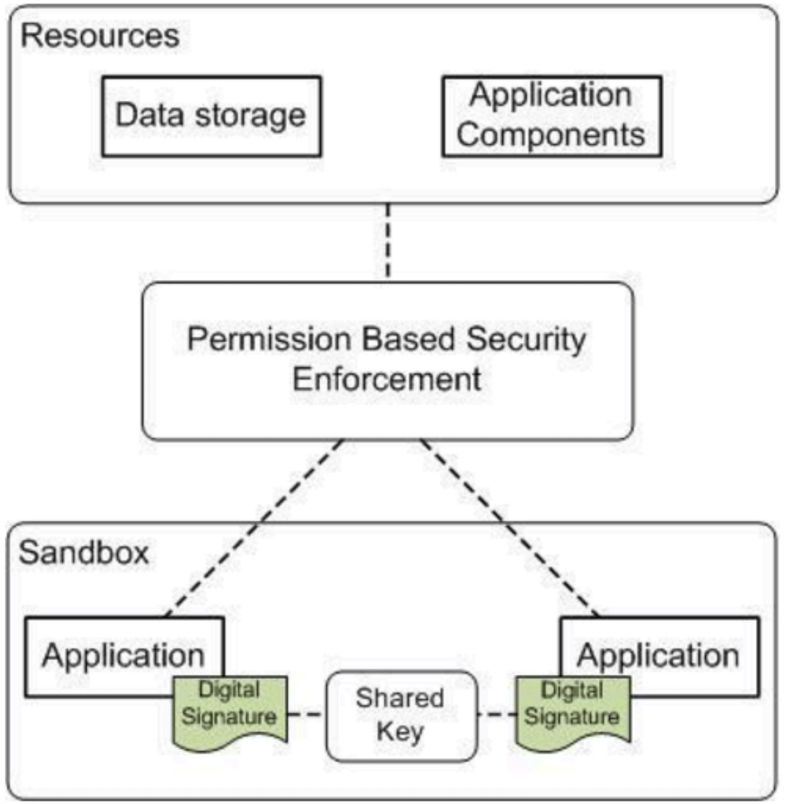
> Permissive: politikaları loglar ancak engellemez, test ve hata ayıklama için kullanılır

Android Security Model

> application UID model



> general security model



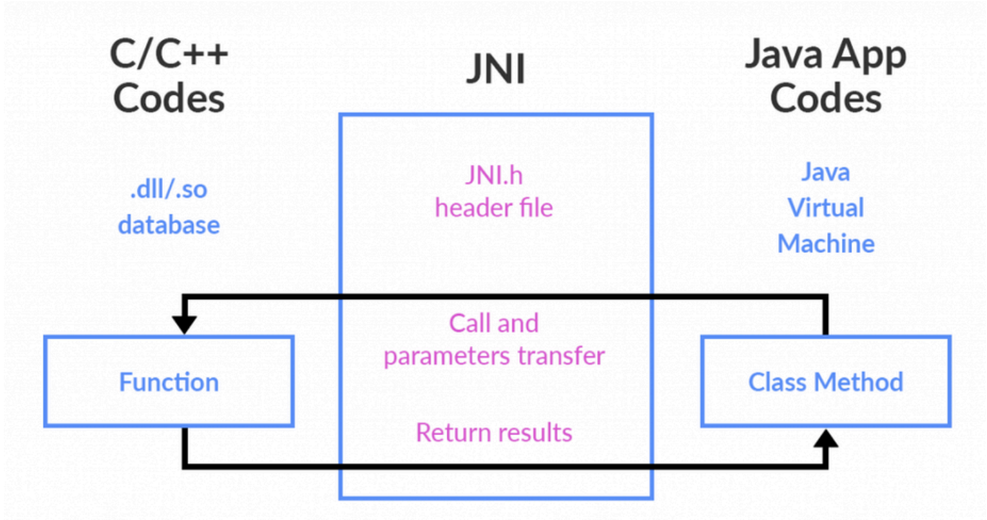
Android NDK

> java / kotlin uygulamalara C/C++ modüllerini entegre etme

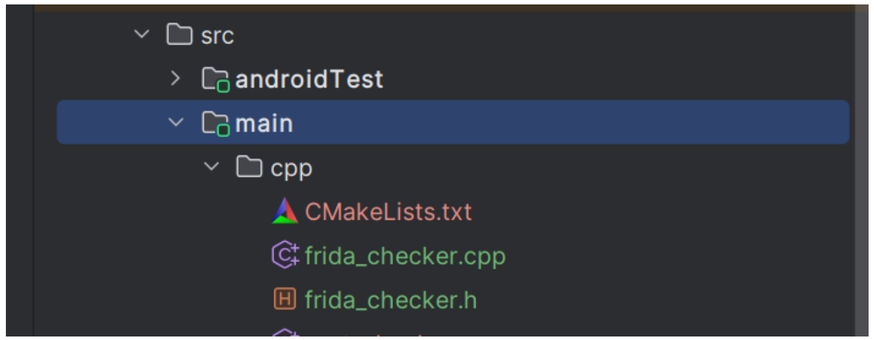
> java ile JNI (Java Native Interface) aracılığıyla iletişim kurar

> c syntax dillerdeki gibi .h başlık dosyası ve c/cpp dosyası kullanılır

> performans gerektiren ses - görüntü gibi işlemlerde ve obfuscate edilmek istenen işlemlerde tercih edilir



Android NDK



> NDK dosyaları default olarak src/main/cpp dizininde bulunur.

> **CMakeLists.txt**: NDK projenizi derlemek için kullanacağınız cmake config dosyası

> **.h dosyaları**: C/C++ dillerinde kullanılan başlık dosyaları

> **.cpp dosyaları**: Modüllerin bulunduğu cpp dosyası

Android NDK- JNI

(Java Native Interface)

> Java tarafından NDK fonksiyonlarına erişebilmek için kullanılır.

> NDK ile Java arasında köprü görevi görür.

> Java ile C/C++ kodları arasında dönüşümleri yapar.

Java primitive type	Native primitive type
void	void
byte	jbyte
int	jint
float	jfloat
double	jdouble
char	jchar
long	jlong
short	jshort
boolean	jboolean

Android NDK- JNI

```
JNIEXPORT jboolean JNICALL Java_com_tsgk_lab_MainActivity_checkPort(JNIEnv *env, jobject, jint port) {
```

JNIEXPORT: native taraftaki C/C++ fonksiyonunu JNI ile dışarı aktarır

jboolean: fonksiyonun dönüş tipi

JNICALL: native fonksiyonun JNI ile Java tarafından çağrılabilmesi için native tarafta nasıl tanımlanacağını belirtir

Java_*: Java tarafında fonksiyonun hangi paket, hangi sınıf ve hangi fonksiyon ismiyle tanımlanacağını belirtir

mobile reversing tools

frida, frida-trace, jnitrace, objection, xposed vs.

Apktool

> en popüler apk analiz tooludur

> decompile edilmiş kaynak kod, uygulamanın tüm assetlerine erişilebilir

> apk decompile edebilir - build alabilir

> tek bir jar dosyası ile tüm sistemlerde çalışabilir

> apktool d base.apk

> apktool b base.apk

Frida

> **vala** dili ile geliştirilmiş bir instrumentation tooludur

> cihazla etkileşime girebilmesi için cihazda frida-server çalışması gerekir

> TCP/IP üzerinden iletişim sağlar

> hedef processe **libfrida-gadget.so** veya **frida-agent.so** isminde library enjekte eder ve injection bu şekilde gerçekleşir

> android için ptrace(), LD_PRELOAD yöntemleri kullanılarak injection işlemi gerçekleşir

> javascript dili ile custom hook scriptleri yazmaya olanak tanır

Frida

> ptrace() ile injection

> ptrace() methodu bir processin başka bir processi kontrol etmesi ve izlemesine olanak sağlar

> methodun birden fazla flagi vardır ve her flagin işlevi farklıdır

> frida, ptrace() kullanarak süreci duraklatır ve dlopen() ile açtığı libfrida-gadget.so kütüphanesini processe inject eder. ardından processi devam ettirir.

> android için ptrace(), LD_PRELOAD yöntemleri kullanılarak injection işlemi gerçekleşir

frida **->** ptrace(*PTRACE_ATTACH*)
-> hedef process **->** ptrace(*PTRACE_POKETEXT*) **->** hedef process
-> ptrace(dlopen("**libfrida-gadget.so**")) **->** hedef process
-> libfrida-gadget.so **->** ptrace(PTRACE_DETACH)

Frida

> ptrace()

> frida, ptrace() ile injection yaparken PTRACE_ATTACH ve PTRACE_CONT flaglerini kullanır.

```c
pid_t target_pid = /* hedef işlemin PID'si */;
ptrace(PTRACE_ATTACH, target_pid, NULL, NULL);

// Hedef işlem durdurulduktan sonra gerekli işlemler yapılır.

// İşlemi devam ettir
ptrace(PTRACE_CONT, target_pid, NULL, NULL);

// İzlemeyi bırakmak için
ptrace(PTRACE_DETACH, target_pid, NULL, NULL);
```

Frida

> LD_PRELOAD ile injection

> LD_PRELOAD ortam değişkeni, bir processin çalıştırılmadan önce belirli bir kitaplığı yüklemesini sağlar

> hedef uygulama başlamadan önce ortam değişkeni olarak LD_PRELOAD değerini set ederek processten önce verilen librarynin yüklenmesini sağlar

> örnek:

```
export LD_PRELOAD=/lib/frida/libfrida-gadget.so
```

```
frida -U -f com.example.app --no-pause -l script.js
```

> burada com.example.app yüklenmeden libfrida-gadget.so kütüphanesi load edilir ve frida bu sayede processe inject olur

Frida

> frida script modules

- **Interceptor.attach()**:
  - Bir işlevin çalışmasını yakalar ve işlevin başında **onEnter()** ve sonunda **onLeave()** özel kod çalıştırır.

- **.overload()**:
  - Aynı isimli fakat farklı imzalara sahip işlevleri ayırt etmek için kullanılır.

- **.implementation()**:
  - Bir işlevin davranışını yeniden tanımlar ve yeni bir implementasyon sağlar.

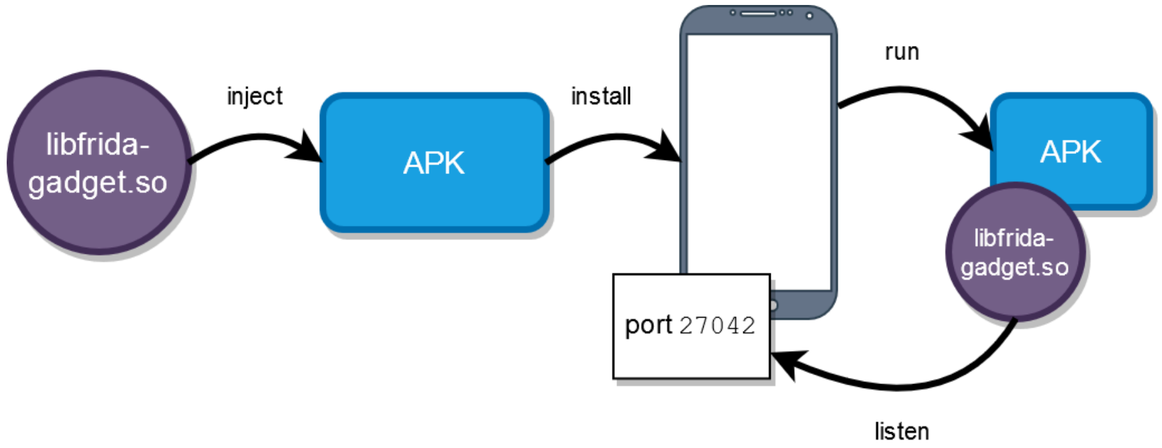
Frida

> Gadget

> fridanın processe inject olmasını sağlayan kütüphane

> android / ios / windows / linux gibi tüm OS'larda çalışır

> root & jailbreak gerektirmez



Frida

> frida-trace

> uygulamanın fonksiyon çağrılarını izler

> hem java hem native taraftaki fonksiyonları trace edebilir

> regex desteği vardır. esnek trace inputu alabilir

> example:

frida-trace -U **-j** "com.tsgk.lab.MainActivity.traceTest" com.tsgk.lab

(java function trace)

frida-trace -U **-i** "nativeFunction" com.tsgk.lab

(ndk function trace)

Frida

> jnitrace

> frida-trace’nin alternatifidir, daha detaylı versiyonudur

> frida-trace’den daha detaylı çıktı verir. (parametreler, backtrace gibi)

> sadece JNI fonksiyonlarını trace eder.

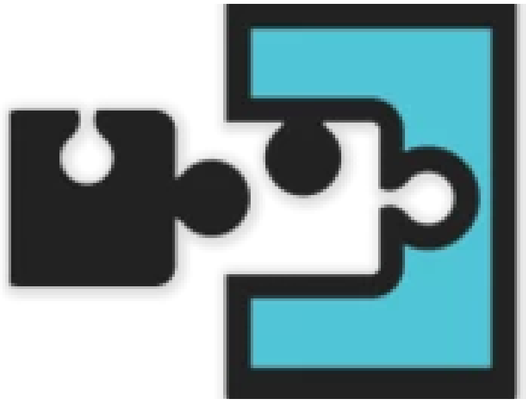
Xposed Framework

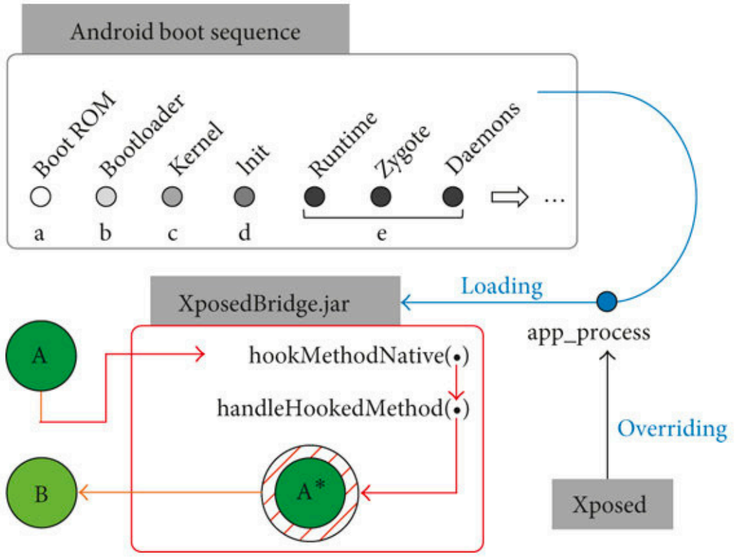
> xposed-bridge

> application ve system levelinde değişiklik yapmayı sağlar

> hem java hem native taraftaki fonksiyonları trace edebilir

> 3. parti bir çok modüle sahiptir (ssl pinning bypass, root detection bypass)





Objection

> objection

> frida tabanlı çalışır

> runtimeda bir çok işlemi pratikçe yapmayı sağlar

> kendi içinde hazır modülleri vardır.
ek olarak 3. parti modül de eklenebilir

some cryptography

AES,RSA,SHA256 ...

XOR (Exclusive OR)

> iki bitin karşılaştırılmasıyla yapılan bir mantıksal işlemdir

> İki bit farklıysa sonuç 1, aynıysa 0 olur

- 0 XOR 0 = 0
- 0 XOR 1 = 1
- 1 XOR 0 = 1
- 1 XOR 1 = 0

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

> veriyi sabit uzunluklara böler ve blok şeklinde şifreleme yapar

- Bit Dizisi A: 1101
- Bit Dizisi B: 1010
- XOR Sonucu: 0111

> iki bit farklıysa 1, aynıysa 0 verir

AES (Advanced Encryption Standard)

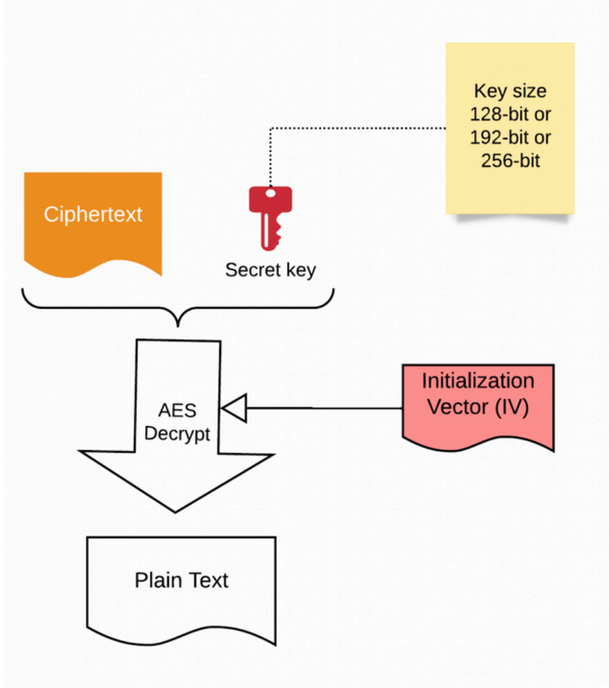
> simetrik şifreleme algoritmasıdır

> 128-bit, 192-bit ve 256-bit anahtar uzunluğu

> birden fazla şifreleme modu (her mod farklı şekilde çalışır)

> her sütuna belirli matematiksel işlemler uygulanır

> matematiksel işlemler, XOR ve satır kaydırma



Symmetric and Asymmetric Encryption

simetrik şifreleme

> aynı şifre hem şifreleme, hem deşifreleme için kullanılır

> anahtar gizlidir

> daha hızlıdır ve yönetimi asimetriğe göre daha basittir

asimetrik şifreleme

> iki farklı anahtar vardır (public key, private key)

> açık anahtar (public key) şifreleme için, gizli anahtar (private key) deşifreleme için kullanılır

> açık anahtar paylaşılabilir, gizli anahtar saklanır

> hız olarak simetrik şifrelemeden daha yavaştır

> anahtar güvenliği simetrik şifrelemeye göre daha güvenlidir

AES (Advanced Encryption Standard)

IV (Initialization Vector)

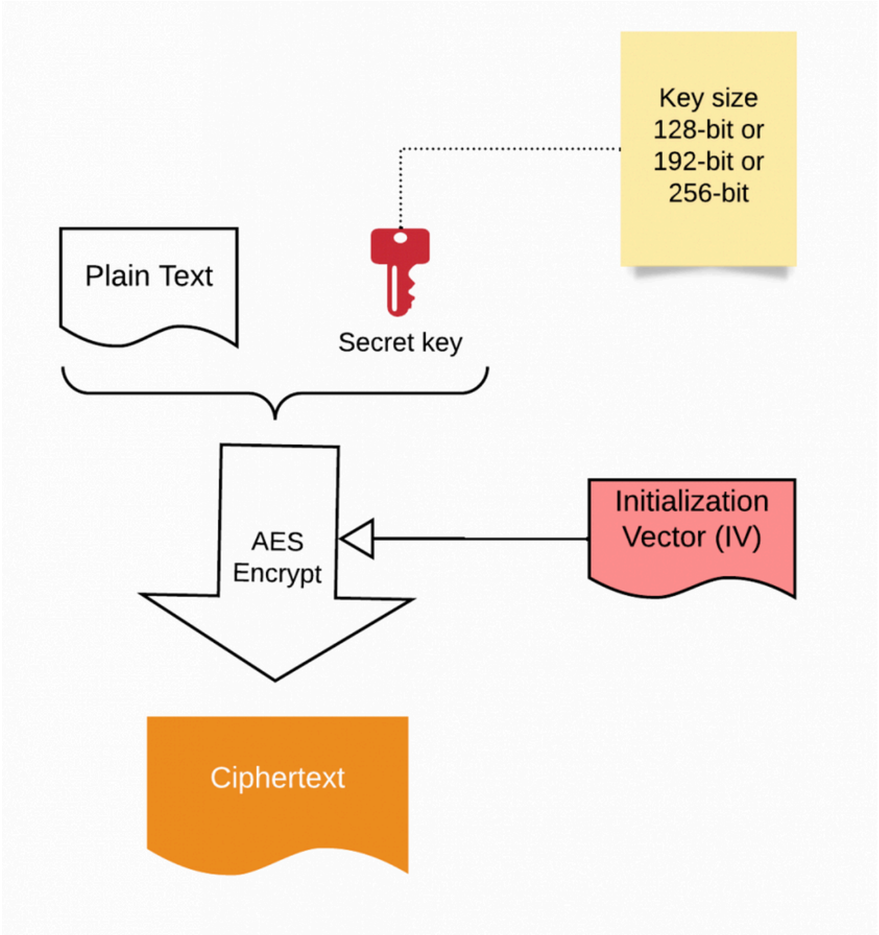
> her işlem için rastgele ve benzersiz bir değerdir

> rastgele olması sayesinde datalar arası benzersizliği sağlar

> şifreleme işlemi başlamadan başlangıç noktası olarak kullanılır

> CBC ve GCM şifreleme modları IV kullanır

> ECB methodu IV kullanmaz, bu yüzden daha az güvenlidir



AES (Advanced Encryption Standard)

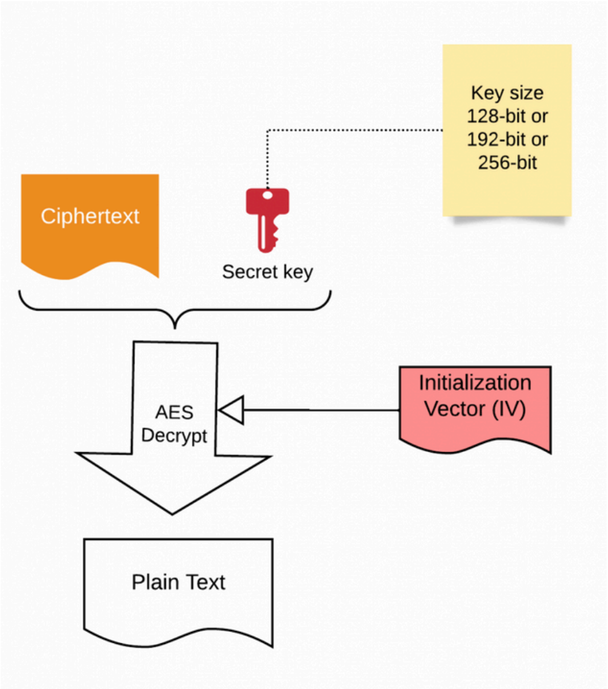
AES ECB (Electronic Codebook)

> her veri bloğunu bağımsız olarak şifreler

> aynı veri her zaman aynı şifreyi üretir

> diğer encryption modlarına göre güvenliği daha düşük

> IV kullanmaz



AES (Advanced Encryption Standard)

AES CBC (Cipher Block Chaining)

> her veri bloğunu bir önceki bloğun şifreli verisi ile XOR'lar

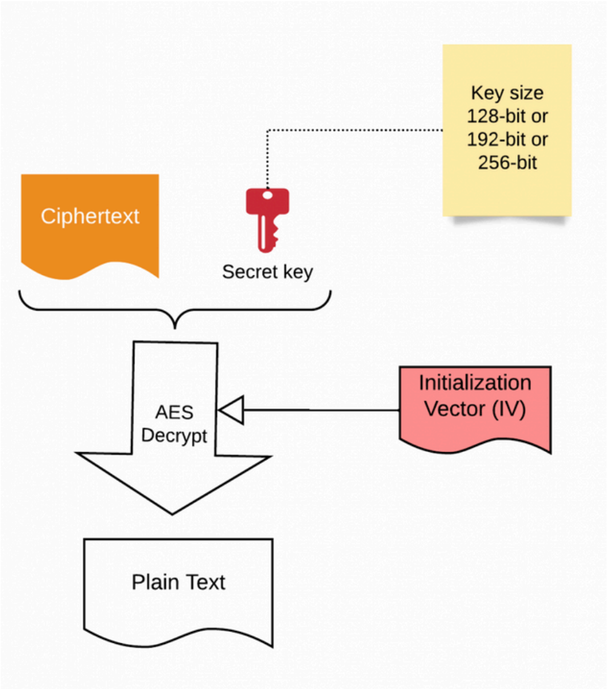
> her blok birbirinden bağımsız

> diğer encryption modlarına göre güvenliği daha düşük

> IV (Initialization Vector) kullanılır

> aynı veriler farklı cipherlar üretir

> paralel işlem zordur



AES (Advanced Encryption Standard)

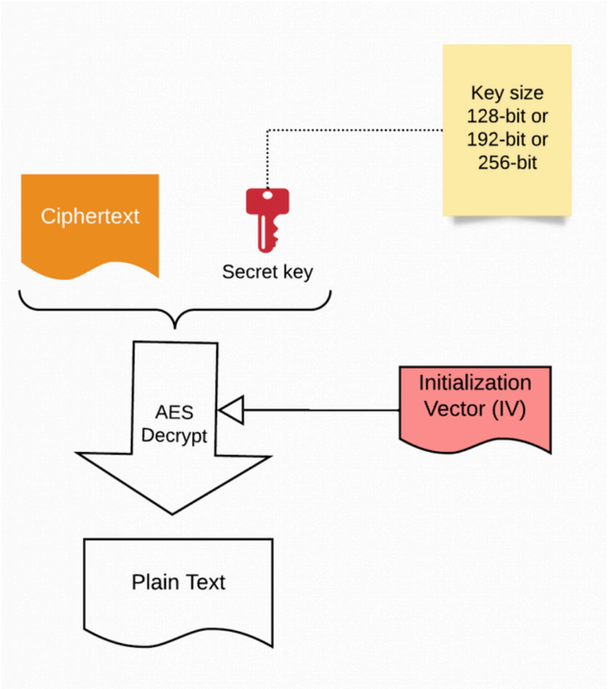
AES GCM(Galois/Counter Mode)

> şifreleme ve doğrulama işlemleri paralel yürütülür

> hem şifreleme hem veri bütünlüğü

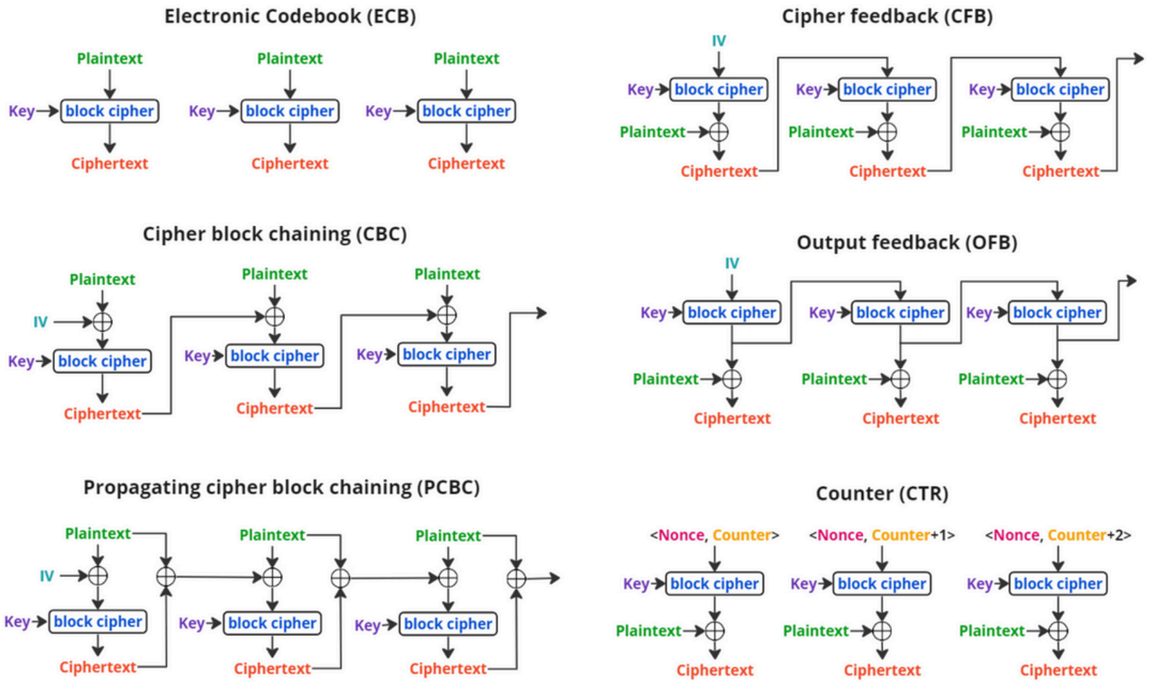
> IV (Initialization Vector) kullanılır

> aynı veriler farklı cipherlar üretir



AES (Advanced Encryption Standard)

AES Modes



RSA (Rivest-Shamir-Adleman)

> asimetrik şifreleme yöntemidir

> public key kullanılarak şifreleme, private key ile de çözme işlemi

> güvenliği matematiksel formüllere dayanır

> 1024, 2048 veya 4096 bit uzunluk

> ssl/tls işlemleri, signature işlemleri

1. **Anahtar Üretimi:**
   - p = 61, q = 53
   - n = 61 * 53 = 3233
   - φ(n) = (61-1)(53-1) = 3120
   - e = 17 (genellikle küçük bir asal sayı seçilir)
   - d = 2753 (çünkü $17 \times 2753 \equiv 1 \pmod{3120}$)
2. **Şifreleme:**
   - Mesaj (m) = 123
   - $c = 123^{17} \bmod 3233 = 855$
3. **Deşifreleme:**
   - $m = 855^{2753} \bmod 3233 = 123$

Base64

> binary'i ASCII karakterlere çevirir

> veriyi üç adet 8-bitlik bayt olarak gruplar
ve bu baytları dört adet 6-bitlik parçalara böler

> 6 bitlik parçaları base64 setindeki karşılıklarıyla değiştirir

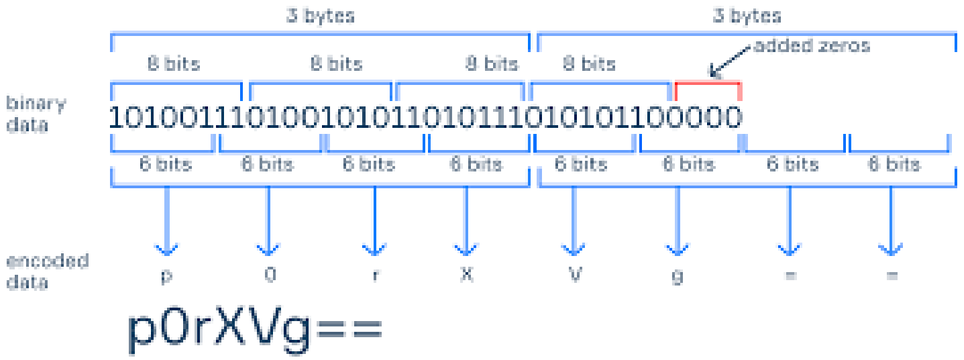


Table 1: The Base 64 Alphabet

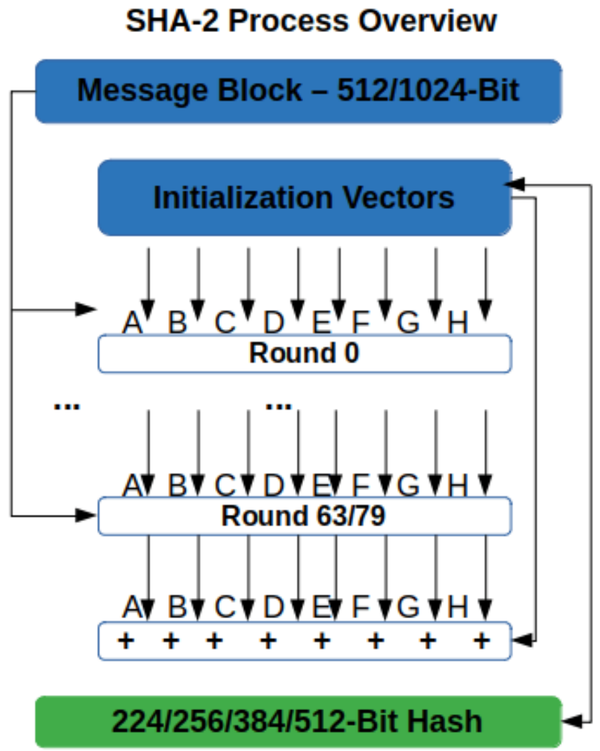
Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

SHA (Secure Hash Algorithm)

> veri belirli bloklara bölünür

> eğer veri blok sayısını tam karşılamıyorsa padding (doldurma) ile tamamlanır

> her blok çeşitli matematiksel işlemler ve bit kaydırma işlemlerine tabi tutulur.

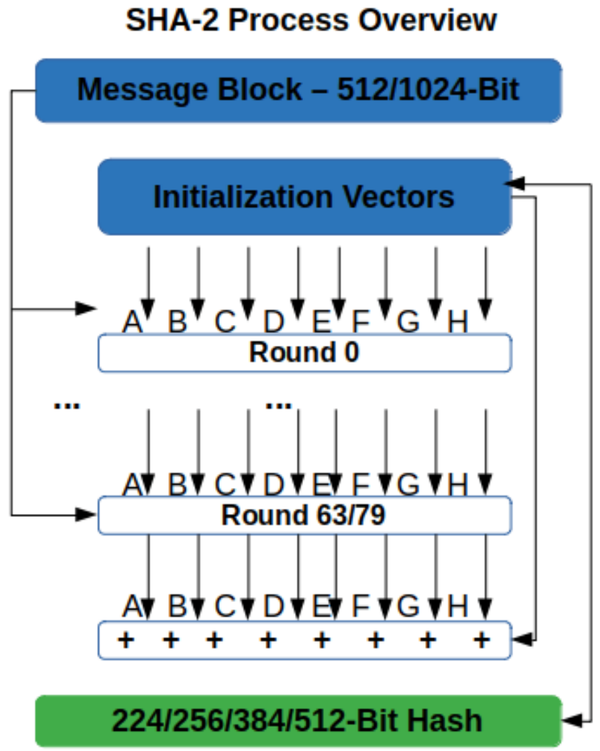


SHA (Secure Hash Algorithm)

SHA256

> sabit uzunlukta hash üretir. (256 bitlik (32 byte))

> 512 bitlik bloklara bölünür ve bloklar
matematiksel işlemler, bit kaydırma işlemlerine tabi tutulur



practice time

let's hook some places