

به نام خدا



«امنیت داده و شبکه»
دکتر امینی

تمرین دوم

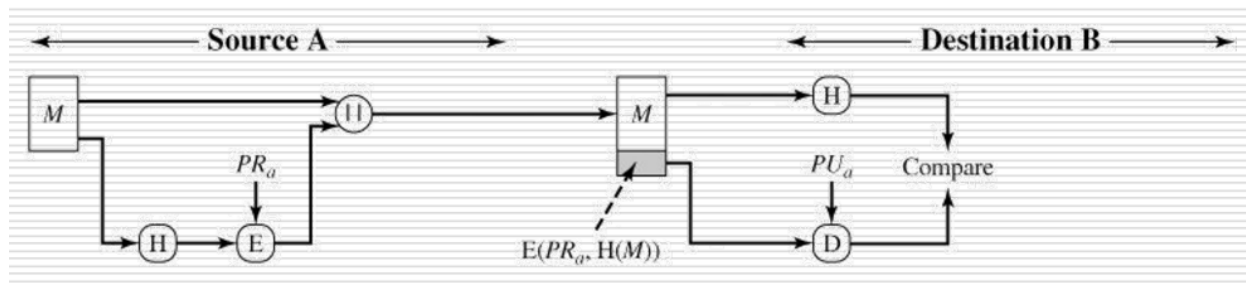
علیرضا دهقانپور فراشاه
۹۸۱۰۱۵۵۵

سوال یک	2
سوال دو	3
سوال سه	4
سوال چهارم	5
سوال پنجم	6
سوال ششم	7
سوال هفتم	8
سوال هشتم	9
سوال نهم	10

سوال یک

الف) کد احراز صحت پیام یا MAC به این علت که با یک کلید عمومی میان دو طرف ایجاد می‌شود خاصیت عدم انکارپذیری را ندارد و دو طرف با همان کلید مشترک قابلیت ایجاد MAC را دارند.

ب) در شکل زیر hash پیام به کمک کلید خصوصی فرستنده رمز می‌شود و سمت گیرنده با رمزگشایی MAC به کمک کلید عمومی فرستنده و مقایسه‌ی hash پیام با مقدار رمزگشایی شده می‌تواند از اصالت فرستنده مطمئن شود.



ج) رمزهای نامتقارن به دو هدف ایجاد شدند. یک ایجاد امضای دیجیتال و دیگری حل مشکل توزیع کلید عمومی بین فرستنده و گیرنده. همچنین برای ایجاد ارتباط میان n نفر به $2/(1-n)$ کلید نیاز است در روش متقارن. مقایسه رمز متقارن و نامتقارن:

- کاربردی: از این نظر رمز نامتقارن بیشتر به عنوان راه حلی برای مسئله‌ی توزیع کلید متقارن استفاده می‌شود. می‌توان با آن امضای دیجیتال داشت و مشکل مدیریت کلیدها را نیز ندارد.
- سرعت: رمز متقارن از عملیات‌های جابجایی و جایگزینی استفاده می‌کند ولی رمز نامتقارن بر اساس عملیات توان‌رسانی است که سنگین‌تر است و بنابراین کندتر از رمزهای متقارن است.
- امنیت: از نظر امنیت هر دو روش امن هستند و امنیت آن‌ها وابسته به طول کلید و توابع مورد استفاده در آن‌ها است.

د) در مورد اول هم محرمانگی را داریم هم صحت پیام زیرا مهاجم پیام را نمی‌تواند ببیند و با تغییر آن نمی‌تواند MAC را تغییر دهد.

در مورد دوم اما محرمانگی پیام حفظ می‌شود ولی صحت آن خیر زیرا مهاجم می‌تواند پیام رمز شده را تغییر دهد و از آن hash بگیرد و کنار پیام خراب شده قرار دهد. دیگر گیرنده نمی‌تواند تشخیص دهد که پیام رمز شده دستکاری شده است یا خیر.

سوال دو

الف) خیر مهاجم با داشتن تابع H به راحتی می‌تواند یک بلاک داده جدید به پیام اضافه کند و c را با بلاک جدید به تابع h بدهد و خروجی چکیده ساز را خروجی تابع h قرار دهد. یعنی این روش در مقابل حمله‌ی افزایش طول ضعیف دارد.

$$C_{n+1} = H(k || m_1, m_2, \dots, m_{n+1}) = h(C_n, m_{n+1})$$

ب) خیر امن نخواهد بود زیرا اگر مهاجم بتواند یک تصادم برای m بیابد یعنی یک پیام دیگری مانند m' وجود دارد که hash آن‌ها در در مرحله‌ی آخر یکی می‌شود. سپس چون خروجی بلاک یکی مانده به آخر که مربوط یکی شده است hash آن به همراه کلید نیز یکسان خواهد بود یعنی c برای دو پیام m و m' یکسان خواهد بود. یعنی مهاجم با حدس تصادم می‌تواند پیام m' را به جای m بدهد و خروجی MAC آن‌ها یکسان خواهد بود.

$$H(m) = H(m')$$

$$H(m || k) = h(H(m), k) = h(H(m'), k) = H(m' || k)$$

- محرمانگی: این روش محرمانگی را حفظ می‌کند زیرا پیام با کلید مشترک رمز شده است و همچنین در قسمت دوم متن ارسالی نیز پیام به همراه رمز شده‌اش به همراه کلید به یک تابع چکیده‌ساز داده شده‌است که برگشت ناپذیر است. بنابراین بدون داشتن کلید نمی‌توان به متن پیام دست یافت.
- صحت پیام: این ویژگی را نیز دارا است. زیرا اگر متن رمز شده تغییر کند دیگر خروجی بخش دوم که حاصل اجرای تابع چکیده‌ساز بر روی متن رمز شده به همراه متن اصلی و کلید است دیگر سازگار نخواهد بود. احتمال اینکه بتوان خروجی تابع چکیده‌ساز و متن رمز شده را به گونه‌ای تغییر داد که سازگار باشند بسیار کم است بنابراین در این مورد نیز امن است.
- عدم انکارپذیری: این ویژگی را به دلیل وجود کلید مشترک نخواهد داشت.

ب) اگر فرض کنیم h در برابر تصادم مقاوم نباشد، یعنی می‌توان مقداری را یافت که خروجی آن با خروجی حاصل از پیام و رمز شده‌ی پیام و کلید یکسان است ولی نمی‌توان از این مقدار جدید استفاده کرد زیرا ما در بخش اول پیام فقط بخشی از ورودی تابع چکیده‌ساز که متن رمز شده است را می‌دهیم. یعنی با فرض اینکه تابع چکیده‌ساز در برابر تصادم مقاوم نیست نیز نمی‌توان متن رمزی پیدا کرد که خروجی تابع چکیده‌ساز برای این متن رمز و رمزگشایی شده‌ی آن و کلید یکسان باشد.

سوال چهارم

اگر فرض کنید $1n$ و $2n$ نسبت به هم اول نباشند یعنی یک عامل مشترک دارند که چون هر دو حاصل ضرب دو عدد اول هستند یعنی یکی از عوامل اول آنها مشترک است.

$$n_1 = p_1 \times q_1$$

$$n_2 = p_2 \times q_1$$

$$\gcd(n_1, n_2) \neq 1 \implies p_1 = p_2 \text{ or } q_1 = q_2$$

بدون کاستن از کلیت مسئله فرض می‌کنیم عامل مشترک $1p$ باشد بنابراین می‌توانیم با محاسبه‌ی \gcd میان $1n$ و $2n$ با مرتبه‌ی زمانی \log به مقدار $1p$ برسیم.

حال با تقسیم $1n$ و $2n$ به $1p$ می‌توانیم $2p$ و $2q$ را بدست آوریم و از آنها خواهیم داشت:

$$\phi(n_1) = (p_1 - 1)(q_1 - 1)$$

$$\phi(n_2) = (p_1 - 1)(q_2 - 1)$$

با داشتن این دو مقدار و $1e$ و $2e$ یافتن $1d$ و $2d$ ممکن است.

$$d_1 = e_1^{-1} \mod \phi(n_1)$$

$$d_2 = e_2^{-1} \mod \phi(n_2)$$

سوال پنجم

بله امکان جعل امضا وجود دارد. در روابط زیر سمت چپ نیز $\text{mod } n$ دارد ولی نوشته نشده است یعنی تمامی روابط در $\text{mod } n$ است. داریم:

$$\left. \begin{matrix} s'_1 = m_1^d \mod n \\ s'_2 = m_2^d \mod n \end{matrix} \right\} \implies \left\{ \begin{matrix} s_1'^e = m_1^{de} \mod n \\ s_2'^e = m_2^{de} \mod n \end{matrix} \right. \implies \left\{ \begin{matrix} s_1'^e = m_1^{de} \mod n = m_1 \\ s_2'^e = m_2^{de} \mod n = m_2 \end{matrix} \right.$$

پس بنابراین اگر $2m1m$ را پیام قرار دهیم و $s'1s'2$ را به عنوان امضا می توانیم امضای بانک را جعل کنیم زیرا:

$$\begin{aligned} (s'_1 s'_2)^e \mod n &= \\ &= (m_1 m_2)^{de} \mod n = \\ &= m_1^{de} m_2^{de} \mod n = \\ &= (m_1^{de} \mod n)(m_2^{de} \mod n) = \\ &= m_1 m_2 \end{aligned}$$

بنابراین این پیام توسط گیرنده تایید می شود.

سوال ششم
طبق تعارف صورت سوال داریم:

$$\begin{aligned} \gcd(m', N) &= 1 \\ C' &= m'^e \pmod{N} \\ C'' &= C'c \pmod{N} \end{aligned}$$

حال d را معکوس m' تعریف می کنیم و C'' را می دهیم به ماشین تا رمزگشایی کند. خروجی ماشین را e می نامیم. خواهیم داشت:

$$d = m'^{-1} \pmod{N} \quad (۱)$$

$$C'^d \pmod{N} = m'^{de} \pmod{N} = m' \pmod{N} \quad (۲)$$

$$e = (C'c)^d \pmod{N} = C'^d c^d \pmod{N} \quad (۳)$$

حال اگر ed را به پیمانه N محاسبه کنیم طبق ۲ و ۱ خواهیم داشت:

$$\begin{aligned} ed \pmod{N} &= (c^d \pmod{N})(C'^d \pmod{N})(m'^{-1} \pmod{N}) = \\ &= (c^d \pmod{N})(m' \pmod{N})(m'^{-1} \pmod{N}) = \\ &= (c^d \pmod{N}) \\ &= m \end{aligned}$$

سوال هفتم

ابتدا هر سه عضو بر عدد α و q توافق می کنند. سپس هر کدام برای خود مقادیر X_A, X_B, X_C را به صورت تصادفی مانند روش دیفی-هلمن عادی تولید می کنند.

سپس به صورت چرخشی ابتدا A به B مقدار $\alpha^{X_A} \bmod q$ و B به C مقدار $\alpha^{X_B} \bmod q$ و C به A مقدار $\alpha^{X_C} \bmod q$ را می فرستد.

حال A مقدار $Y_{CA} = \alpha^{X_C \times X_A} \bmod q$ را محاسبه می کند. به همین ترتیب B مقدار $Y_{AB} = \alpha^{X_A \times X_B} \bmod q$ و C مقدار $Y_{BC} = \alpha^{X_B \times X_C} \bmod q$ را محاسبه می کنند.

در این مرحله A به B مقدار Y_{CA} و B به C مقدار Y_{AB} و C به A مقدار Y_{BC} را ارسال می کند.

حال A با داشتن Y_{BC} می تواند مقدار $Y_{BC}^{X_A} \bmod q = \alpha^{X_A \times X_B \times X_C} \bmod q$ را می کند و این مقدار می شود کلید مشترک سه نفر.

سپس B نیز به همین ترتیب با داشتن Y_{CA} می تواند مقدار $Y_{CA}^{X_B} \bmod q = \alpha^{X_A \times X_B \times X_C} \bmod q$ را می کند.

در نهایت C نیز مقدار $Y_{AB}^{X_C} \bmod q = \alpha^{X_A \times X_B \times X_C} \bmod q$ را محاسبه می کند.

در نهایت $\alpha^{X_A \times X_B \times X_C} \bmod q$ کلید مشترک هر سه نفر است.

سوال هشتم

(الف)

رمز الجمل بر پایه‌ی سختی مسئله‌ی لگاریتم گسسته است ولی RSA بر پایه‌ی سختی تجزیه اعداد بزرگ است. از RSA برای رمزگذاری و رمزگشایی، امضای رقمی و توزیع کلید استفاده می‌شود ولی از الجمل برای رمزگذاری و رمزگشایی و توزیع کلید استفاده می‌شود. مهم‌ترین برتری الجمل نسبت به RSA وجود عامل تصادفی در رمزنگاری است که سبب می‌شود یک پیام ثابت در هر بار رمزنگاری به یک عبارت متفاوت رمز شود. به طور کلی رمز RSA سریع‌تر از الجمل است زیرا محاسبات در RSA حجم کمتری دارد. همچنین برای داشتن یک سطح برابر امنیتی معمولاً طول کلید RSA باید بیشتر از الجمل باشد. معمولاً برای زمانی که حجم داده ارسالی کم باشد مانند توزیع کلید از الجمل استفاده می‌شود و زمانی که داده بزرگ باشد از رمز RSA استفاده می‌شود.

(ب)

این روش در مقابل حمله‌ی Chosen Cipher ضعف دارد. فرض کنید (C_1, C_2) رمز شده‌ی پیام m باشد و بخواهیم آن را رمزگشایی کنیم. پارامترهای عمومی α, q هستند و کلید خصوصی X_A و کلید عمومی α^{X_A} است. داریم:

$$(C_1, C_2) = (\alpha^r \mod q, m \times \alpha^{X_A r} \mod q)$$

حال r' و m' را تصادفی انتخاب می‌کنیم و پیام $(C_1', C_2') = (C_1 * \alpha^{r'} \mod q, C_2 * m' * \alpha^{X_A r'})$ را به سیستم رمزگشایی می‌دهیم. داریم:

$$\begin{aligned}(C_1', C_2') &= \\ &= (\alpha^r \alpha^{r'} \mod q, m m' \times \alpha^{X_A r} \alpha^{X_A r'} \mod q) = \\ &= (\alpha^{r+r'} \mod q, m m' \times \alpha^{X_A (r+r')} \mod q)\end{aligned}$$

بنابراین خروجی سیستم رمزگشایی برابر $(m * m') \mod q$ خواهد شد و اگر خروجی را در $m^{-1} \mod q$ ضرب کنیم به $m \mod q$ که پیام است خواهیم رسید. بنابراین این رمزگذاری در برابر Chosen Cipher آسیب‌پذیر است.

در ابتدا از نصب شدن openssl مطمئن می‌شویم و سپس محتویات فایل secret.txt را مشاهده می‌کنیم.

```
Win64 OpenSSL Command Pr x + v
C:\Users\alireza\Desktop\dns-q9-98101555>openssl version
OpenSSL 3.1.0 14 Mar 2023 (Library: OpenSSL 3.1.0 14 Mar 2023)

C:\Users\alireza\Desktop\dns-q9-98101555>type secret.txt
Alireza Dehghanpour Farashah

C:\Users\alireza\Desktop\dns-q9-98101555>
```

حال کلید خصوصی را تولید می‌کنیم.

```
Win64 OpenSSL Command Pr x + v
C:\Users\alireza\Desktop\dns-q9-98101555>type secret.txt
Alireza Dehghanpour Farashah

C:\Users\alireza\Desktop\dns-q9-98101555>openssl genrsa -out key.pem 1024

C:\Users\alireza\Desktop\dns-q9-98101555>type key.pem
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBALTCduIPWHpx6S+w
x9rJnfs0LI76kvh86mvTny+yjU3SmDJqIG2hrUyLxL312e8PYg6yDwQDm2JuFCx
NLcWsX9Q6TYEfsShkHBREzuCsosQUrvVxvwFLBuFDGGkR8GMhDBH0vNi/szosK4j
xxVR//9r8zQoVldWrCOiK8L0feRfAgMBAECgYEAJCGFIU6oGHf/09jLYlRNhe9
16RIJnV9pEII1QvZU/ECn/zCfcJh9F2y3B2FRfqqhlh9vVL/L98bz+zYoW4UeDqa
R4LPHwBqrvtSxkGPBfLdrPOSNYacgS34j3Y5/e+A39t2sCJp8GTXXo8YrQriyYB
lbPLZD2d7HXy9eOCK0kCQDQ5/wyKM3ahPT8zKy05KnQhu3FMTsvV0JQ1BfsPjx7
QVsiAPkSSTYtkDCJqJjq4dbcAnRDXJB+uTgGd/VYjrB9AkEAXP3Cj275iiJIh3Ge
pE8xSPz90Wp2ImOPW+XGAmZ4LiHpzUCaPhRaLMf/LZ68iiHRU3aJBjU/3hoQNimC
rQW7CwJAKW5nPdWSkTMYHqpe37zg8qWJtFj8aNyKfquB2KoVcSVRLgsr4vzEsH6Z
AP6akfrpMX7BUMw3tBIO/oMT492u5QJAIofY8wug8nXQA04JwQOgwAqY2NewnZaW
KqW8b+9GYc/QDAqm/vrSU8c7Jh8NLal/6AF/Q3dSGry22sesAGhwQJAGUYVWnC8
r+Ejto1yFv5MrDLbXRI81FXqEgOFE4okgt9hc01Gopb/xjHowXYtKWlLnitQzZNX
Hmw3emX17cHuaw==
-----END PRIVATE KEY-----

C:\Users\alireza\Desktop\dns-q9-98101555>
```

سپس کلید عمومی را تولید می‌کنیم.

```
Win64 OpenSSL Command Pr x + v

C:\Users\alireza\Desktop\dns-q9-98101555>openssl rsa -in key.pem -pubout -out pub-key.pem
writing RSA key

C:\Users\alireza\Desktop\dns-q9-98101555>type pub-key.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCsGqSIb3DQEBAQUAA4GNADCBiQKBgQC0wnbiD1h6cekvsMQPayZ37DpS
O+pL4f0pr058vso1N0pgyaiBtoa1Mi8S99dnvD2IOsg8EA5tibhQsTS3FrF/UOk2
BH7EoZBwURM7grKLEFK71cb8BSwbhQxhpEfBjIQwRzrzYv7M6LCuI8cVUf//a/M0
KFZXVqwjoivC9H3kXwIDAQAB
-----END PUBLIC KEY-----

C:\Users\alireza\Desktop\dns-q9-98101555>
```

حال باید از فایل secret.txt با الگوریتم SHA-256 چکیده بگیریم.

```
Win64 OpenSSL Command Pr x + v

C:\Users\alireza\Desktop\dns-q9-98101555>openssl dgst -sha256 -binary -out secret.sha256 secret.txt

C:\Users\alireza\Desktop\dns-q9-98101555>type secret.sha256
| a?iè·||#L#EfiZ%Yû-r{||,SIñ||ç
C:\Users\alireza\Desktop\dns-q9-98101555>
```

حال باید با کلید خصوصی آن را امضا کنیم. برای امضا دو دستور rsautl و pkeyutl وجود دارد که در داکيومنت openssl آورده شده است که دستور اول در حال منسوخ شدن است. از دستور دوم برای اینکار استفاده می‌کنیم.

```
Win64 OpenSSL Command Pr x + v

C:\Users\alireza\Desktop\dns-q9-98101555>openssl pkeyutl -sign -in secret.sha256 -inkey key.pem
-keyform PEM -out secret.sha256.sign

C:\Users\alireza\Desktop\dns-q9-98101555>type secret.sha256.sign
|±_aμ^||6U`m{âûxqum|ÿF-^Üà*e▲_uY7B@ ♦[!-L£-X"≈L3]%;!üÿ -+æuS
}Lg||{Ω≈π ||g||π£É! ?ωæ||Üj_0δSPYâ0||·L 8[á|'Lñ0LlÖ0L
C:\Users\alireza\Desktop\dns-q9-98101555>
```

حال با الگوریتم DES3 فایل را رمز می‌کنیم.

```
Win64 OpenSSL Command Pr x + v - □ x
C:\Users\alireza\Desktop\dns-q9-98101555>openssl des3 -in secret.txt -out secret.3des -pass
pass:IHateAmniat -pbkdf2

C:\Users\alireza\Desktop\dns-q9-98101555>type secret.3des
Salted__ε||çç+η-£].!sfr!đô«
|'JφΓúΠεφ-°J3÷°f
C:\Users\alireza\Desktop\dns-q9-98101555>|
```

حال فایل رمز شده را رمزگشایی می کنیم.

```
Win64 OpenSSL Command Pr x + v - □ x
C:\Users\alireza\Desktop\dns-q9-98101555>openssl des3 -d -in secret.3des -out decrypted.txt
-pass pass:IHateAmniat -pbkdf2

C:\Users\alireza\Desktop\dns-q9-98101555>type decrypted.txt
Alireza Dehghanpour Farashah

C:\Users\alireza\Desktop\dns-q9-98101555>|
```

حال باید با کلید عمومی امضا را verify کنیم.

```
Win64 OpenSSL Command Pr x + v - □ x
C:\Users\alireza\Desktop\dns-q9-98101555>openssl pkeyutl -verifyrecover -in secret.sha256.sign
-out secret_verified.sha256 -pubin -inkey pub-key.pem

C:\Users\alireza\Desktop\dns-q9-98101555>type secret_verified.sha256
|f a?îê·||L#Le£îz%Yû~rf||,SIñ|ç
C:\Users\alireza\Desktop\dns-q9-98101555>|
```

حال از فایل رمزگشایی شده چکیده می گیریم.

```
Win64 OpenSSL Command Pr x + v - □ x
C:\Users\alireza\Desktop\dns-q9-98101555>openssl dgst -sha256 -binary -out decrypted.sha256 decrypted.txt
C:\Users\alireza\Desktop\dns-q9-98101555>type decrypted.sha256
| a7iê·||L#Lefiz%Yû~rf||,SIñ|t
C:\Users\alireza\Desktop\dns-q9-98101555>
```

حال در مود باینری دو فایل 256secret_verified.sha و 256decrypted.sha را مقایسه می کنیم و متوجه می شویم که یکسان هستند.

```
Win64 OpenSSL Command Pr x + v - □ x
C:\Users\alireza\Desktop\dns-q9-98101555>fc /b decrypted.sha256 secret_verified.sha256
Comparing files decrypted.sha256 and SECRET_VERIFIED.SHA256
FC: no differences encountered

C:\Users\alireza\Desktop\dns-q9-98101555>|
```

منابع

<https://pagefault.blog/2019/04/22/how-to-sign-and-verify-using-openssl/>

<https://opensource.com/article/19/6/cryptography-basics-openssl-part-2>

<https://www.openssl.org/docs/man3.0/man1/openssl-rsautl.html>

<https://www.openssl.org/docs/man3.0/man1/openssl-pkeyutl.html>