

## 4<sup>η</sup> Εργασία στις σύγχρονες εφαρμογές ασφάλειας δικτύων

Όνομα: Γεώργιος

Επώνυμο: Βέργος

Αριθμός μητρώου: 1072604

Εξάμηνο: 7<sup>ο</sup> (4<sup>ο</sup> έτος)

Τμήμα: ΤΜΗΥΠ(CEID)

Ημερομηνία: 16/12/2022

### Μέρος Α

1)

Με την εντολή: `sudo apt install openssl` εγκαθιστώ το πακέτο `openssl`:

```
vergos@Vergos:~$ sudo apt install openssl
[sudo] password for vergos:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libcurl3-gnutls procmail python3-chardet python3-debian python3-idna python3-pkg-resources
  python3-pycurl python3-six python3-urllib3 sensible-mdm
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  ca-certificates
The following NEW packages will be installed:
  openssl
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 853 kB of archives.
After this operation, 1,501 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bullseye/main amd64 openssl amd64 1.1.1n-0+deb11u3 [853 kB]
Fetched 853 kB in 1s (1,619 kB/s)
Selecting previously unselected package openssl.
(Reading database ... 34860 files and directories currently installed.)
Preparing to unpack .../openssl_1.1.1n-0+deb11u3_amd64.deb ...
Unpacking openssl (1.1.1n-0+deb11u3) ...
Setting up openssl (1.1.1n-0+deb11u3) ...
Processing triggers for man-db (2.9.4-2) ...
vergos@Vergos:~$
```

Με την εντολή : openssl version ελέγχω την εγκατεστημένη έκδοση του openssl.

```
vergos@Vergos:~$ openssl version
OpenSSL 1.1.1n 15 Mar 2022
vergos@Vergos:~$ _
```

2)

Με την εντολή openssl ciphers βλέπω όλους του κώδικες κρυπτογράφησης του openssl:

```
vergos@Vergos:~$ openssl ciphers
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES128-SHA:RSA-PSK-AES256-GCM-SHA384:DHE-PSK-AES256-GCM-SHA384:RSA-PSK-CHACHA20-POLY1305:DHE-PSK-CHACHA20-POLY1305:ECDHE-PSK-CHACHA20-POLY1305:AES256-GCM-SHA384:PSK-AES256-GCM-SHA384:PSK-CHACHA20-POLY1305:RSA-PSK-AES128-GCM-SHA256:DHE-PSK-AES128-GCM-SHA256:AES128-GCM-SHA256:PSK-AES128-GCM-SHA256:AES256-SHA256:AES128-SHA256:ECDHE-PSK-AES256-CBC-SHA384:ECDHE-PSK-AES256-CBC-SHA:SRP-RSA-AES-256-CBC-SHA:SRP-AES-256-CBC-SHA:RSA-PSK-AES256-CBC-SHA384:DHE-PSK-AES256-CBC-SHA384:RSA-PSK-AES256-CBC-SHA:DHE-PSK-AES256-CBC-SHA:AES256-SHA:PSK-AES256-CBC-SHA384:PSK-AES256-CBC-SHA:ECDHE-PSK-AES128-CBC-SHA256:ECDHE-PSK-AES128-CBC-SHA:SRP-RSA-AES-128-CBC-SHA:SRP-AES-128-CBC-SHA:RSA-PSK-AES128-CBC-SHA256:DHE-PSK-AES128-CBC-SHA256:RSA-PSK-AES128-CBC-SHA:DHE-PSK-AES128-CBC-SHA:AES128-SHA:PSK-AES128-CBC-SHA256:PSK-AES128-CBC-SHA
vergos@Vergos:~$ openssl ciphers -s
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:AES256-GCM-SHA384:AES128-GCM-SHA256:AES256-SHA256:AES128-SHA256:AES256-SHA:AES128-SHA
vergos@Vergos:~$
```

Με την εντολή openssl ciphers -s (η δεύτερη στην παραπάνω εικόνα) μόνο αυτούς που υποστηρίζονται.

3)

Με την εντολή openssl ciphers -tls1\_3 βλέπω όλους τους κώδικες που χρησιμοποιούν το πρωτόκολλο tls1.3

```

vergos@Vergos:~$ openssl ciphers -tls1_3
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES128-SHA:RSA-PSK-AES256-GCM-SHA384:DHE-PSK-AES256-GCM-SHA384:RSA-PSK-CHACHA20-POLY1305:DHE-PSK-CHACHA20-POLY1305:ECDHE-PSK-CHACHA20-POLY1305:AES256-GCM-SHA384:PSK-AES256-GCM-SHA384:PSK-CHACHA20-POLY1305:RSA-PSK-AES128-GCM-SHA256:DHE-PSK-AES128-GCM-SHA256:AES128-GCM-SHA256:PSK-AES128-GCM-SHA256:AES256-SHA256:AES128-SHA256:ECDHE-PSK-AES256-CBC-SHA384:ECDHE-PSK-AES256-CBC-SHA:SRP-RSA-AES-256-CBC-SHA:SRP-AES-256-CBC-SHA:RSA-PSK-AES256-CBC-SHA384:DHE-PSK-AES256-CBC-SHA384:RSA-PSK-AES256-CBC-SHA:DHE-PSK-AES256-CBC-SHA:AES256-SHA:PSK-AES256-CBC-SHA384:PSK-AES256-CBC-SHA:ECDHE-PSK-AES128-CBC-SHA256:ECDHE-PSK-AES128-CBC-SHA:SRP-RSA-AES-128-CBC-SHA:SRP-AES-128-CBC-SHA:RSA-PSK-AES128-CBC-SHA256:DHE-PSK-AES128-CBC-SHA256:RSA-PSK-AES128-CBC-SHA:DHE-PSK-AES128-CBC-SHA:AES128-SHA:PSK-AES128-CBC-SHA256:PSK-AES128-CBC-SHA
vergos@Vergos:~$ openssl ciphers -s -tls1_3
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
vergos@Vergos:~$ _

```

Με την εντολή `openssl ciphers -s -tls1_3` προβάλλω από τους προηγούμενους μόνο αυτούς που υποστηρίζονται.

Για τον κώδικα `TLS_AES_256_GCM_SHA384`:

## Recommended Cipher Suite

### IANA name:

TLS\_AES\_256\_GCM\_SHA384

### OpenSSL name:

TLS\_AES\_256\_GCM\_SHA384

### Hex code:

0x13, 0x02

### TLS Version(s):

TLS1.3

### Protocol:

Transport Layer Security (TLS)

### Key Exchange:

-

### Authentication:

-

### Encryption:

**AEAD** Advanced Encryption Standard with 256bit key in Galois/Counter mode (AES 256 GCM)

### Hash:

Secure Hash Algorithm 384 (SHA384)

Με αλγόριθμο κρυπτογράφησης Advanced Encryption Standard (AES) με μήκος κλειδιού 256 bits με mode λειτουργίας GCM(Galois Counter mode).

Ο αλγόριθμος κατακερματισμού είναι ο SHA-384.

Προφανώς χρησιμοποιείται η έκδοση 1.3 του TLS.

Για τον κώδικα TLS\_CHACHA20\_POLY1305\_SHA256:

### Recommended Cipher Suite

**IANA name:**

TLS\_CHACHA20\_POLY1305\_SHA256

**Hex code:**

0x13, 0x03

**TLS Version(s):**

TLS1.3

---

**Protocol:**

Transport Layer Security (TLS)

**Key Exchange:**

-

**Authentication:**

-

**Encryption:**

**AEAD** ChaCha stream cipher and Poly1305 authenticator (CHACHA20 POLY1305)

**Hash:**

Secure Hash Algorithm 256 (SHA256)

---

**Included in RFC:**

[RFC 8446](#)

**Machine-readable:**

[application/json](#)

Με αλγόριθμο κρυπτογράφησης ChaCha20 POLY1305 με μέγεθος κλειδιού 128 bits.

Ο αλγόριθμος κατακερματισμού είναι ο SHA256.

Προφανώς χρησιμοποιείται η έκδοση 1.3 του TLS.

Για τον κώδικα TLS\_AES\_128\_GCM\_SHA256:

## Recommended Cipher Suite

**IANA name:**

TLS\_AES\_128\_GCM\_SHA256

**OpenSSL name:**

TLS\_AES\_128\_GCM\_SHA256

**Hex code:**

0x13, 0x01

**TLS Version(s):**

TLS1.3

---

**Protocol:**

Transport Layer Security (TLS)

**Key Exchange:**

-

**Authentication:**

-

**Encryption:**

**AES** Advanced Encryption Standard with 128bit key in Galois/Counter mode (AES 128 GCM)

**Hash:**

Secure Hash Algorithm 256 (SHA256)

---

**Included in RFC:**

[RFC 8446](#)

**Machine-readable:**

[application/json](#)

Με αλγόριθμο κρυπτογράφησης Advanced Encryption Standard (AES) με μήκος κλειδιού 128 bits με mode λειτουργίας GCM(Galois Counter mode).

Ο αλγόριθμος κατακερματισμού είναι ο SHA256.

Προφανώς χρησιμοποιείται η έκδοση 1.3 του TLS.

Από αυτούς τους 3 όλοι τους δεν είναι ευπαθείς αλλά ασφαλείς και προτεινόμενοι.

4)

Εφαρμόζοντας τα σωστά φίλτρα βλέπω ποιοι κώδικες είναι

Recommended-Secure του **openssl** που υποστηρίζουν tls1.2 είναι οι εξής:

Secure	DHE-RSA-AES128-CCM
Recommended	ECDHE-ECDSA-CHACHA20-POLY1305
Secure	DHE-RSA-AES128-GCM-SHA256
Recommended	ECDHE-PSK-CHACHA20-POLY1305
Recommended	DHE-PSK-AES128-GCM-SHA256
Recommended	DHE-PSK-CHACHA20-POLY1305
Secure	DHE-RSA-AES256-CCM8
Secure	DHE-RSA-AES256-GCM-SHA384
Secure	ECDHE-ECDSA-AES256-CCM
Secure	DHE-RSA-AES256-CCM
Recommended	ECDHE-ECDSA-AES128-GCM-SHA256
Secure	DHE-PSK-AES128-CCM8
Secure	DHE-PSK-AES256-CCM
Secure	ECDHE-ECDSA-AES128-CCM8
Recommended	DHE-DSS-AES128-GCM-SHA256

Και:

Secure	DHE-RSA-CHACHA20-POLY1305
Secure	ECDHE-RSA-AES256-GCM-SHA384
Recommended	DHE-DSS-AES256-GCM-SHA384
Secure	ECDHE-ECDSA-AES256-CCM8
Recommended	DHE-PSK-AES256-GCM-SHA384
Secure	DHE-PSK-AES256-CCM8
Secure	DHE-RSA-AES128-CCM8
Secure	ECDHE-RSA-CHACHA20-POLY1305
Recommended	ECDHE-ECDSA-AES256-GCM-SHA384
Secure	ECDHE-ECDSA-AES128-CCM
Secure	ECDHE-RSA-AES128-GCM-SHA256
Secure	DHE-PSK-AES128-CCM

Ενώ με library φίλτρο all:

Secure	TLS_DHE_RSA_WITH_AES_128_CCM
Recommended	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
Secure	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
Recommended	TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256
Recommended	TLS_DHE_PSK_WITH_AES_128_GCM_SHA256
Recommended	TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256
Recommended	TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256
Recommended	TLS_DHE_DSS_WITH_ARIA_128_GCM_SHA256
Secure	TLS_DHE_RSA_WITH_AES_256_CCM_8
Secure	TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
Recommended	TLS_DHE_DSS_WITH_ARIA_256_GCM_SHA384
Recommended	TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
Secure	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
Secure	TLS_ECDHE_ECDSA_WITH_AES_256_CCM
Secure	TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384

Και:

Secure	TLS_DHE_RSA_WITH_AES_256_CCM
Recommended	TLS_DHE_PSK_WITH_CAMELLIA_256_GCM_SHA384
Secure	TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256
Recommended	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
Secure	TLS_PSK_DHE_WITH_AES_128_CCM_8
Recommended	TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384
Secure	TLS_DHE_PSK_WITH_AES_256_CCM
Recommended	TLS_DHE_PSK_WITH_ARIA_256_GCM_SHA384
Secure	TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
Secure	TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256
Secure	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
Recommended	TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
Recommended	TLS_DHE_PSK_WITH_CAMELLIA_128_GCM_SHA256
Secure	TLS_ECCPWD_WITH_AES_128_CCM_SHA256
Secure	TLS_ECDHE_PSK_WITH_AES_128_CCM_8_SHA256

Και:

Secure	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
Secure	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
Recommended	TLS_DHE_DSS_WITH_CAMELLIA_128_GCM_SHA256
Recommended	TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256
Secure	TLS_ECCPWD_WITH_AES_256_CCM_SHA384
Recommended	TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
Secure	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
Secure	TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256
Recommended	TLS_DHE_PSK_WITH_AES_256_GCM_SHA384
Secure	TLS_PSK_DHE_WITH_AES_256_CCM_8
Secure	TLS_DHE_RSA_WITH_AES_128_CCM_8
Secure	TLS_ECCPWD_WITH_AES_128_GCM_SHA256
Secure	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
Recommended	TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384
Recommended	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

Και:

Secure	TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
Secure	TLS_ECCPWD_WITH_AES_256_GCM_SHA384
Recommended	TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256
Recommended	TLS_DHE_DSS_WITH_CAMELLIA_256_GCM_SHA384
Secure	TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
Secure	TLS_ECDHE_ECDSA_WITH_AES_128_CCM
Secure	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
Recommended	TLS_DHE_PSK_WITH_ARIA_128_GCM_SHA256
Secure	TLS_DHE_PSK_WITH_AES_128_CCM
Secure	TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384

Ενώ οι Recommended-Secure κώδικες του [openssl](#) που υποστηρίζουν tls1.3 είναι οι εξής:

Secure	TLS_AES_128_CCM_SHA256
Recommended	TLS_AES_128_GCM_SHA256
Recommended	TLS_AES_256_GCM_SHA384

Ενώ οι Recommended-Secure κώδικες με φίλτρο library: all που υποστηρίζουν tls1.3 είναι οι εξής:



**Secure** TLS\_AES\_128\_CCM\_8\_SHA256

**Secure** TLS\_AES\_128\_CCM\_SHA256

**Recommended** TLS\_CHACHA20\_POLY1305\_SHA256

**Recommended** TLS\_AES\_128\_GCM\_SHA256

**Recommended** TLS\_AES\_256\_GCM\_SHA384

Και στις δύο περιπτώσεις οι πιο ισχυροί είναι οι προτεινόμενοι (Recommended) οπότε όλοι οι παραπάνω εκτός αυτών με τη σήμανση Secure.

Δηλαδή:

Κώδικες με φίλτρο library: all

**Recommended** TLS\_DHE\_PSK\_WITH\_AES\_256\_GCM\_SHA384

**Recommended** TLS\_ECDHE\_ECDSA\_WITH\_CHACHA20\_POLY1305\_SHA256

**Recommended** TLS\_ECDHE\_PSK\_WITH\_AES\_256\_GCM\_SHA384

**Recommended** TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384

**Recommended** TLS\_ECDHE\_PSK\_WITH\_CHACHA20\_POLY1305\_SHA256

**Recommended** TLS\_DHE\_DSS\_WITH\_AES\_128\_GCM\_SHA256

**Recommended** TLS\_ECDHE\_ECDSA\_WITH\_ARIA\_128\_GCM\_SHA256

**Recommended** TLS\_DHE\_PSK\_WITH\_CAMELLIA\_128\_GCM\_SHA256

**Recommended** TLS\_DHE\_PSK\_WITH\_CAMELLIA\_256\_GCM\_SHA384

**Recommended** TLS\_DHE\_PSK\_WITH\_AES\_128\_GCM\_SHA256

**Recommended** TLS\_DHE\_DSS\_WITH\_CAMELLIA\_256\_GCM\_SHA384

**Recommended** TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256

**Recommended** TLS\_DHE\_PSK\_WITH\_CHACHA20\_POLY1305\_SHA256

**Recommended** TLS\_ECDHE\_ECDSA\_WITH\_CAMELLIA\_128\_GCM\_SHA256

**Recommended** TLS\_DHE\_DSS\_WITH\_ARIA\_128\_GCM\_SHA256

Και:

**Recommended** TLS\_DHE\_DSS\_WITH\_CAMELLIA\_128\_GCM\_SHA256

**Recommended** TLS\_ECDHE\_PSK\_WITH\_AES\_128\_GCM\_SHA256

**Recommended** TLS\_DHE\_PSK\_WITH\_ARIA\_128\_GCM\_SHA256

**Recommended** TLS\_ECDHE\_ECDSA\_WITH\_ARIA\_256\_GCM\_SHA384

**Recommended** TLS\_DHE\_DSS\_WITH\_ARIA\_256\_GCM\_SHA384

**Recommended** TLS\_ECDHE\_ECDSA\_WITH\_CAMELLIA\_256\_GCM\_SHA384

**Recommended** TLS\_DHE\_DSS\_WITH\_AES\_256\_GCM\_SHA384

**Recommended** TLS\_DHE\_PSK\_WITH\_ARIA\_256\_GCM\_SHA384

## Κώδικες με φίλτρο library: openssl

**Recommended** DHE-PSK-AES256-GCM-SHA384

**Recommended** ECDHE-ECDSA-CHACHA20-POLY1305

**Recommended** ECDHE-ECDSA-AES256-GCM-SHA384

**Recommended** ECDHE-PSK-CHACHA20-POLY1305

**Recommended** DHE-DSS-AES128-GCM-SHA256

**Recommended** DHE-PSK-AES128-GCM-SHA256

**Recommended** ECDHE-ECDSA-AES128-GCM-SHA256

**Recommended** DHE-PSK-CHACHA20-POLY1305

**Recommended** DHE-DSS-AES256-GCM-SHA384

Για tls1.2

Και:

## Κώδικες με φίλτρο library: openssl

**Recommended** TLS\_AES\_128\_GCM\_SHA256

**Recommended** TLS\_AES\_256\_GCM\_SHA384

## Κώδικες με φίλτρο library: all

**Recommended** TLS\_CHACHA20\_POLY1305\_SHA256

**Recommended** TLS\_AES\_128\_GCM\_SHA256

**Recommended** TLS\_AES\_256\_GCM\_SHA384

για tls1.3

5)

Για τον κώδικα με όνομα : ECDHE-ECDSA-AES128-GCM-SHA256 στο openssl έχουμε τις παρακάτω πληροφορίες:

### Recommended Cipher Suite

**IANA name:**

TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256

**OpenSSL name:**

ECDHE-ECDSA-AES128-GCM-SHA256

**GnuTLS name:**

TLS\_ECDHE\_ECDSA\_AES\_128\_GCM\_SHA256

**Hex code:**

0xC0, 0x2B

**TLS Version(s):**

TLS1.2

---

**Protocol:**

Transport Layer Security (TLS)

**Key Exchange:**

Elliptic Curve Diffie-Hellman Ephemeral (ECDHE)

**Authentication:**

Elliptic Curve Digital Signature Algorithm (ECDSA)

**Encryption:**

**AEAD** Advanced Encryption Standard with 128bit key in Galois/Counter mode (AES 128 GCM)

**Hash:**

Secure Hash Algorithm 256 (SHA256)

---

**Included in RFC:**

[RFC 5289](#)

**Machine-readable:**

[application/json](#)

Για τον τρόπο ανταλλαγής κλειδιών χρησιμοποιείται το πρωτόκολλο Elliptic-curve Diffie–Hellman Ephemeral (ECDHE).

Το Elliptic-curve Diffie–Hellman είναι ένα πρωτόκολλο ανταλλαγής κλειδιών που επιτρέπει σε δύο μεριές με καθεμία να διαθέτει ένα

ζεύγος δημοσίου και ιδιωτικού κλειδιού για να εγκαταστήσουν ένα κοινό μυστικό σε ένα μη ασφαλές κανάλι επικοινωνίας. Αυτό το μυστικό μπορεί είτε να χρησιμοποιηθεί απευθείας ως κλειδί είτε ως μέσο για την παραγωγή άλλου κλειδιού. Τα κλειδιά αυτά μπορούν να χρησιμοποιηθούν για να κρυπτογραφήσουν την επικοινωνία στο προαναφερθέν μη ασφαλές κανάλι. Τα δημόσια κλειδιά είναι είτε στατικά είτε εφήμερα (εξ'ού και το ECDHE) με τα εφήμερα κλειδιά να είναι προσωρινά και να μην ταυτοποιούνται.

Για την ταυτοποίηση χρησιμοποιείται ο αλγόριθμος Elliptic Curve Digital Signature (ECDSA).

Ο αλγόριθμος κρυπτογράφησης των δεδομένων είναι ο AES(Advanced Encryption Standard) με μέγεθος κλειδιού 128 bit.

Υποστηρίζει την έκδοση του TLS 1.2 και τέλος ο αλγόριθμος κατακερματισμού είναι ο SHA256.

6)

Για την κρυπτογράφηση:

Δημιουργώ μέσω του pwgen με όρισμα -s (από το secure) ένα τυχαίο μυστικό κλειδί (32 χαρακτήρων):

```
vergos@DESKTOP-711LR13:~$ pwgen -s 32 1  
7fQLRSss2xwRIwTM75Np0Wg5YKuQXseW  
vergos@DESKTOP-711LR13:~$
```

Ακόμη δημιουργώ μέσω του pwgen με όρισμα -s (από το secure) τυχαίο initialization vector (16 χαρακτήρων εφόσον το μέγεθος του κλειδιού είναι 256 bit και η επιλεγμένη λειτουργία είναι CBC) . Η κωδικοποίηση είναι base64.

```
vergos@DESKTOP-711LR13:~$ pwgen -s 16 1
5qaEBtP3xkcrOlgP
vergos@DESKTOP-711LR13:~$
```

Όλη η διαδικασία φαίνεται στην παρακάτω εικόνα:

The screenshot shows a web application titled "AES Encryption" with a pink header. It contains several input fields and a "Encrypt" button. The "Enter Plain Text to Encrypt" field contains "1072604". The "Select Mode" dropdown is set to "CBC". The "Key Size in Bits" dropdown is set to "256". The "Enter Initialization Vector" field contains "5qaEBtP3xkcrOlgP". The "Enter Secret Key" field contains "7QLRSs52wRwTM75Np0Wg5YkUQXseW". The "Output Text Format" dropdown is set to "Base64". A green "Encrypt" button is located below the input fields. The output area shows the encrypted result: "wchLAMsKHA9HjF2y6zkoG==".

AES Encryption

Enter Plain Text to Encrypt - 1072604

Select Mode CBC

Key Size in Bits 256

Enter Initialization Vector - 5qaEBtP3xkcrOlgP

Enter Secret Key - 7QLRSs52wRwTM75Np0Wg5YkUQXseW

Output Text Format Base64

Encrypt

The String which is to be encrypted using AES

AES works in 2 modes - CBC and ECB mode.  
**CBC (Cipher Block Chaining)** requires Initialization Vector(IV) to make each message unique. Using IV we randomize the encryption of similar blocks. So any identical plain text blocks will be encrypted into dissimilar cipher text blocks.  
**ECB(Electronic Code Book)** encryption mode does not need the IV for encryption. The input plain text will be divided into blocks and each block will be encrypted with the key provided and hence identical plain text blocks are encrypted into identical cipher text blocks.

The input can be of 128 bit or 192 bit or 256 bit  
So if key size is 128 then "aesEncryptionKey" is a valid secret key because it has 16 characters i.e 16\*8=128 bits

The initialization vector is needed in case of CBC mode  
The initialization vector size should be 128 bit  
So if initialization vector size is 128 then "encryptionInitVec" is a valid initialization vector because it has 16 characters i.e 16\*8=128 bits

As AES is a symmetric algorithm the same secret key can be used for both encryption and decryption. The expected secret key size we have specified in the key size dropdown  
So if key size is 128 then "aesEncryptionKey" is a valid secret key because it has 16 characters i.e 16\*8=128 bits

Specify if output format should be in Base64 encoded format or Hex Encoded format.

wchLAMsKHA9HjF2y6zkoG==

Για την αποκρυπτογράφηση:

Χρησιμοποιώ το ίδιο κλειδί(συμμετρική κρυπτογραφία του AES) και ίδιο initialization vector και στο πρώτο πεδίο βάζω τον κρυπτογραφημένο αριθμό μητρώου μου. Το μέγεθος του κλειδιού προφανώς είναι το ίδιο 256 bit και η λειτουργία CBC . Η κωδικοποίηση όπως και πάνω είναι base64.

AES Decryption

Enter Encrypted Text to Decrypt

wchLAMsKHA9HjF2y6zkocg==

Input Text Format

Base64

Select Mode

CBC

Key Size in Bits

256

Enter Initialization Vector

5qaEBtP3xkcrOlgP

Enter Secret Key

7fQLRSss2xwRlwTM75Np0Wg5YKuQXseW

Decrypt

1072604

The AES Encrypted String which we want to decrypt

Specify if input format is in Base64 encoded format or Hex Encoded format.

AES works in 2 modes - CBC and ECB mode.

CBC (Cipher Block Chaining) requires Initialization Vector(IV) to make each message unique. Using IV we randomize the encryption of similar blocks. So any identical plain text blocks will be encrypted into dissimilar cipher text blocks.

ECB(Electronic Code Book) encryption mode does not need the IV for encryption. The input plain text will be divided into blocks and each block will be encrypted with the key provided and hence identical plain text blocks are encrypted into identical cipher text blocks.

The input can be of 128 bit or 192 bit or 256 bit

So if key size is 128 then "aesEncryptionKey" is a valid secret key because it has 16 characters i.e 16\*8=128 bits

The initialization vector is needed in case of CBC mode

The initialization vector size should be 128 bit

So if initialization vector size is 128 then "encryptionIntVec" is a valid initialization vector because it has 16 characters i.e 16\*8=128 bits

As AES is a symmetric algorithm the same secret key can be used for both encryption and decryption. The expected secret key size we have specified in the key size dropdown

So if key size is 128 then "aesEncryptionKey" is a valid secret key because it has 16 characters i.e 16\*8=128 bits

Επαληθεύοντας και από την άλλη ιστοσελίδα έχουμε:

AES Online Encryption

Enter text to be Encrypted

1072604

Select Cipher Mode of Encryption

CBC

Key Size in Bits

256

Enter IV (Optional)

5qaEBtP3xkcrOlgP

Enter Secret Key

7fQLRSss2xwRlwTM75Np0Wg5YKuQXseW

Output Text Format:

Base64

Encrypt

AES Encrypted Output:

wchLAMsKHA9HjF2y6zkocg==

AES Online Decryption

Enter text to be Decrypted

wchLAMsKHA9HjF2y6zkocg==

Input Text Format:

Base64

Select Cipher Mode of Decryption

CBC

Enter IV Used During Encryption(Optional)

5qaEBtP3xkcrOlgP

Key Size in Bits

256

Enter Secret Key used for Encryption

7fQLRSss2xwRlwTM75Np0Wg5YKuQXseW

Decrypt

AES Decrypted Output (Base64):

MTA3MjYwNA==

Decode to Plain Text

1072604

Online DES Encrypt

Decrypt

Δεύτερη έκδοση: Υλοποίηση με τη χρήση του rand του openssl

Δημιουργώ το secret key 32 χαρακτήρων και το initialization vector των 16 χαρακτήρων:

```
bonis@DESKTOP-711LR13:~$ openssl rand -base64 24
jEyZ8QGcPfPnKAfhsP4yM1jbYpyfYBYxq
bonis@DESKTOP-711LR13:~$ openssl rand -base64 12
aQDowRr3Y/Pgoghs
bonis@DESKTOP-711LR13:~$
```

Για την κρυπτογράφηση έχω:

The screenshot shows a web application titled "AES Encryption". It has several input fields and a "Encrypt" button. The "Enter Plain Text to Encrypt" field contains "1072604". The "Select Mode" dropdown is set to "CBC". The "Key Size in Bits" dropdown is set to "256". The "Enter Initialization Vector" field contains "aQDowRr3Y/Pgoghs". The "Enter Secret Key" field contains "jEyZ8QGcPfPnKAfhsP4yM1jbYpyfYBYxq". The "Output Text Format" dropdown is set to "Base64". The "Encrypt" button is green. Below the button, the output is displayed in a text area: "qnlcLoeaKouQahSMqLpXQ==".

**AES Encryption**

Enter Plain Text to Encrypt - 1072604

Select Mode - CBC

Key Size in Bits - 256

Enter Initialization Vector - aQDowRr3Y/Pgoghs

Enter Secret Key - jEyZ8QGcPfPnKAfhsP4yM1jbYpyfYBYxq

Output Text Format - Base64

**Encrypt**

qnlcLoeaKouQahSMqLpXQ==

The String which is to be encrypted using AES

AES works in 2 modes - CBC and ECB mode.  
**CBC (Cipher Block Chaining)** requires Initialization Vector(IV) to make each message unique. Using IV we randomize the encryption of similar blocks. So any identical plain text blocks will be encrypted into dissimilar cipher text blocks.  
**ECB(Electronic Code Book)** encryption mode does not need the IV for encryption. The input plain text will be divided into blocks and each block will be encrypted with the key provided and hence identical plain text blocks are encrypted into identical cipher text blocks.

**The input can be of 128 bit or 192 bit or 256 bit**  
So if key size is 128 then "aesEncryptionKey" is a valid secret key because it has 16 characters i.e. 16\*8=128 bits

**The initialization vector is needed in case of CBC mode**  
The initialization vector size should be 128 bit  
So if initialization vector size is 128 then "encryptionIntVec" is a valid initialization vector because it has 16 characters i.e. 16\*8=128 bits

**As AES is a symmetric algorithm the same secret key can be used for both encryption and decryption.** The expected secret key size we have specified in the key size dropdown  
So if key size is 128 then "aesEncryptionKey" is a valid secret key because it has 16 characters i.e. 16\*8=128 bits

Specify if output format should be in Base64 encoded format or Hex Encoded format.

Για την αποκρυπτογράφηση:

Χρησιμοποιώ το ίδιο κλειδί(συμμετρική κρυπτογραφία του AES) και ίδιο initialization vector και στο πρώτο πεδίο βάζω τον κρυπτογραφημένο αριθμό μητρώου μου. Το μέγεθος του κλειδιού προφανώς είναι το ίδιο 256 bit και η λειτουργία CBC . Η κωδικοποίηση όπως και πάνω είναι base64.

AES Decryption

Enter Encrypted Text to Decrypt -

qnlcioeaK0HuQahSMqLpXQ==

The AES Encrypted String which we want to decrypt

Input Text Format

Base64

Specify if input format is in Base64 encoded format or Hex Encoded format.

Select Mode

CBC

AES works in 2 modes - CBC and ECB mode

CBC (Cipher Block Chaining) requires Initialization Vector(IV) to make each message unique Using IV we randomize the encryption of similar blocks. So any identical plain text blocks will be encrypted into dissimilar cipher text blocks

ECB(Electronic Code Book) encryption mode does not need the IV for encryption. The input plain text will be divided into blocks and each block will be encrypted with the key provided and hence identical plain text blocks are encrypted into identical cipher text blocks.

Key Size in Bits

256

The input can be of 128 bit or 192 bit or 256 bit

So if key size is 128 then "aesEncryptionKey" is a valid secret key because it has 16 characters i.e 16\*8=128 bits

Enter Initialization Vector -

aqDowRr3Y/Pgoghs

The initialization vector is needed in case of CBC mode

The initialization vector size should be 128 bit

So if initialization vector size is 128 then "encryptionIVVec" is a valid initialization vector because it has 16 characters i.e 16\*8=128 bits

Enter Secret Key -

jEyZ8Q3cfPnKAfhsP4yMjlbYpyfYBYxq

As AES is a symmetric algorithm the same secret key can be used for both encryption and decryption. The expected secret key size we have specified in the key size dropdown

So if key size is 128 then "aesEncryptionKey" is a valid secret key because it has 16 characters i.e 16\*8=128 bits

Decrypt

1072604

Επαληθεύοντας από την άλλη ιστοσελίδα έχω:

AES Online Encryption

Enter text to be Encrypted

1072604

Select Cipher Mode of Encryption

CBC

Key Size in Bits

256

Enter IV (Optional)

aqDowRr3Y/Pgoghs

Enter Secret Key

jEyZ8Q3cfPnKAfhsP4yMjlbYpyfYBYxq

Output Text Format: Base64 Hex

Encrypt

AES Encrypted Output:

qnlcioeaK0HuQahSMqLpXQ==

AES Online Decryption

Enter text to be Decrypted

qnlcioeaK0HuQahSMqLpXQ==

Input Text Format: Base64 Hex

Select Cipher Mode of Decryption

CBC

Enter IV Used During Encryption(Optional)

aqDowRr3Y/Pgoghs

Key Size in Bits

256

Enter Secret Key used for Encryption

jEyZ8Q3cfPnKAfhsP4yMjlbYpyfYBYxq

Decrypt

AES Decrypted Output (Base64):

MTA3MjYwNA==

Decode to Plain Text

1072604

Το base64 είναι ένα σύνολο από κωδικοποιήσεις ψηφιακών δεδομένων σε κείμενο. Αντιστοιχίζονται τα byte ενός αρχείου σε ένα υποσύνολο του ASCII. Κάθε ψηφίο ενός κειμένου σε base64 αντιστοιχεί σε 6 bits οπότε για να κωδικοποιήσω ένα byte πληροφορίας θέλω



τουλάχιστον 2 τέτοια ψηφία. Πάντα κωδικοποιούνται 3 συνεχόμενα bytes πληροφορίας οπότε αφού  $3 \cdot 8 = 24 \text{ bits}$   $\rightarrow 24/6 = 4$  ψηφία base64.

Χρησιμοποιείται ο ακόλουθος πίνακας για τις αντιστοιχίσεις ψηφιακών δεδομένων σε χαρακτήρες base64:

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

7)

Οι κρυπτογραφικές συναρτήσεις κατακερματισμού είναι αλγόριθμοι κατακερματισμού που χρησιμοποιούνται στην κρυπτογραφία.

Αντιστοιχίζεται ένα μήνυμα αυθαίρετου μεγέθους σε πεπερασμένη ακολουθία από bits. Μερικά χαρακτηριστικά μίας τέτοιας συνάρτησης είναι : 1) Να είναι ανθεκτική σε (ισχυρές ) συγκρούσεις δηλαδή να είναι πρακτικά αδύνατο να βρεθούν δύο διαφορετικά μηνύματα που παράγουν την ίδια τιμή κατακερματισμού. 2) Μονόδρομη έτσι ώστε να μην μπορεί να ανακτηθεί το αρχικό μήνυμα.

Χρησιμοποιείται στην ασφάλεια πληροφοριών , σε ψηφιακές υπογραφές, ταυτοποιήσεις(όπως κωδικών πρόσβασης) , MACs καθώς και επιβεβαίωση της αυθεντικότητας δεδομένων που στέλνονται.

Παραδείγματα τέτοιων συναρτήσεων είναι οι MD5, SHA-1,SHA-2(SHA-256, SHA-512) κλπ.

Με την εντολή : openssl list -digest-algorithms:

```
bonis@DESKTOP-711LR13:~$ openssl list -digest-algorithms
RSA-MD4 => MD4
RSA-MD5 => MD5
RSA-RIPEMD160 => RIPEMD160
RSA-SHA1 => SHA1
RSA-SHA1-2 => RSA-SHA1
RSA-SHA224 => SHA224
RSA-SHA256 => SHA256
RSA-SHA3-224 => SHA3-224
RSA-SHA3-256 => SHA3-256
RSA-SHA3-384 => SHA3-384
RSA-SHA3-512 => SHA3-512
RSA-SHA384 => SHA384
RSA-SHA512 => SHA512
RSA-SHA512/224 => SHA512-224
RSA-SHA512/256 => SHA512-256
RSA-SM3 => SM3
BLAKE2b512
BLAKE2s256
Id-rsassa-pkcs1-v1_5-with-sha3-224 => SHA3-224
Id-rsassa-pkcs1-v1_5-with-sha3-256 => SHA3-256
Id-rsassa-pkcs1-v1_5-with-sha3-384 => SHA3-384
Id-rsassa-pkcs1-v1_5-with-sha3-512 => SHA3-512
MD4
md4WithRSAEncryption => MD4
MD5
MD5-SHA1
md5WithRSAEncryption => MD5
ripemd => RIPEMD160
RIPEMD160
ripemd160WithRSA => RIPEMD160
rmd160 => RIPEMD160
SHA1
sha1WithRSAEncryption => SHA1
SHA224
sha224WithRSAEncryption => SHA224
SHA256
sha256WithRSAEncryption => SHA256
SHA3-224
SHA3-256
SHA3-384
SHA3-512
SHA384
sha384WithRSAEncryption => SHA384
SHA512
SHA512-224
sha512-224WithRSAEncryption => SHA512-224
SHA512-256
sha512-256WithRSAEncryption => SHA512-256
sha512WithRSAEncryption => SHA512
SHAKE128
SHAKE256
SM3
sm3WithRSAEncryption => SM3
ssl3-md5 => MD5
ssl3-sha1 => SHA1
whirlpool
bonis@DESKTOP-711LR13:~$
```

Δημιουργώ το SHA512 του αριθμού μητρώου μου (1072604):

## SHA512

SHA512 online hash function

1072604

Input type Text

Hash ☒ Auto Update

6a384a8fc1cbc87706466e91116953820af8e8966f673bf494e6ad5bf3f812bf92c7f760b162a5f80587bf17cb471037cb678363575234315be21eaae420aef0

## Μέρος Β

Δημιουργώ το ιδιωτικό μου κλειδί:

```
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl genrsa -aes256 -out 1072604_private.pem 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....
.....+++++
.....+++++
e is 65537 (0x010001)
Enter pass phrase for 1072604_private.pem:
Verifying - Enter pass phrase for 1072604_private.pem:
vergman@vergman-hppavilion15notebookpc:~/Desktop$
```

Δημιουργώ το δημόσιο μου κλειδί:

```
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl rsa -in 1072604_private.pem -outform PEM -pubout -out 1072604_public.pem
Enter pass phrase for 1072604_private.pem:
writing RSA key
vergman@vergman-hppavilion15notebookpc:~/Desktop$
```

Αποθηκεύω το μήνυμα: 'Hello world: 1072604' στο αρχείο 1072604\_msg.txt

```
vergman@vergman-hppavilion15notebookpc:~/Desktop$ echo 'Hello world: 1072604' > 1072604_msg.txt
vergman@vergman-hppavilion15notebookpc:~/Desktop$
```

Δημιουργώ το κλειδί για τη συμμετρική κρυπτογράφηση του μηνύματος : 1072604\_randomkey.bin

```
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl rand 64 > 1072604_randomkey.bin
vergman@vergman-hppavilion15notebookpc:~/Desktop$
```

Κρυπτογραφώ το μήνυμα με το παραπάνω κλειδί:

```
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl enc -aes-256-cbc -salt -in 1072604_msg.txt -out 1072604_msg.bin -pass file:./1072604_randomkey.bin
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
vergman@vergman-hppavilion15notebookpc:~/Desktop$
```

Κωδικοποιώ το κρυπτογραφημένο μήνυμα σε base64:

```
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl base64 -in 1072604_msg.bin -out 1072604_msg.b64
vergman@vergman-hppavilion15notebookpc:~/Desktop$
```

Κρυπτογραφώ το κλειδί συμμετρικής κρυπτογραφίας (το 1072604\_randomkey.bin) και προκύπτει το 1072604\_randomkey.enc.bin. Έπειτα κωδικοποιώ το τελευταίο αρχείο σε base64.

```
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl rsautl -encrypt -inkey kvlachos_public.pem -pubin -in 1072604_randomkey.bin -out 1072604_randomkey.enc.bin
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl base64 -in 1072604_randomkey.enc.bin -out 1072604_randomkey.enc.b64
vergman@vergman-hppavilion15notebookpc:~/Desktop$
```

Δημιουργώ το hash digest του 1072604\_msg.txt ώστε ο παραλήπτης να ελέγξει ότι δεν υπήρξαν λάθη κατά τη μεταφορά του μηνύματος ούτε ότι "πειράχθηκε".

```
vergman@vergman-hppavilion15notebookpc:~/Desktop$ cat 1072604_msg.txt | openssl dgst -sha256 -binary | xxd -p > 1072604_msg.digest.txt
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl base64 -in 1072604_msg.digest.txt -out 1072604_msg.digest.b64
vergman@vergman-hppavilion15notebookpc:~/Desktop$ █
```

Με τη χρήση του ιδιωτικού μου κλειδιού και του hash digest του μηνύματος που στέλνω κατασκευάζω την υπογραφή μου ώστε ο παραλήπτης να βεβαιωθεί πως το μήνυμα ήρθε όντως από εμένα.

```
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl dgst -sha256 -sign 1072604_private.pem -out 1072604_msg.signature.bin 1072604_msg.digest.txt
Enter pass phrase for 1072604_private.pem:
vergman@vergman-hppavilion15notebookpc:~/Desktop$ openssl base64 -in 1072604_msg.signature.bin -out 1072604_msg.signature.b64
vergman@vergman-hppavilion15notebookpc:~/Desktop$ █
```