

CSI4108 / SEC5101
(Fundamentals of) Cryptography

Assignment #2

Due: Friday, October 21, 2022 (before 16:00, to be submitted via Brightspace)

1. [1.5 marks] Consider the rotor system shown in the figure below. Assume that the first 20 characters of a plaintext string have already been encrypted (leading to the overall state shown in (b)). Choose the next 10 letters of plaintext and show the output from the 3rd rotor for each of these input letters.

[Note: your plaintext should be chosen independently by you.]

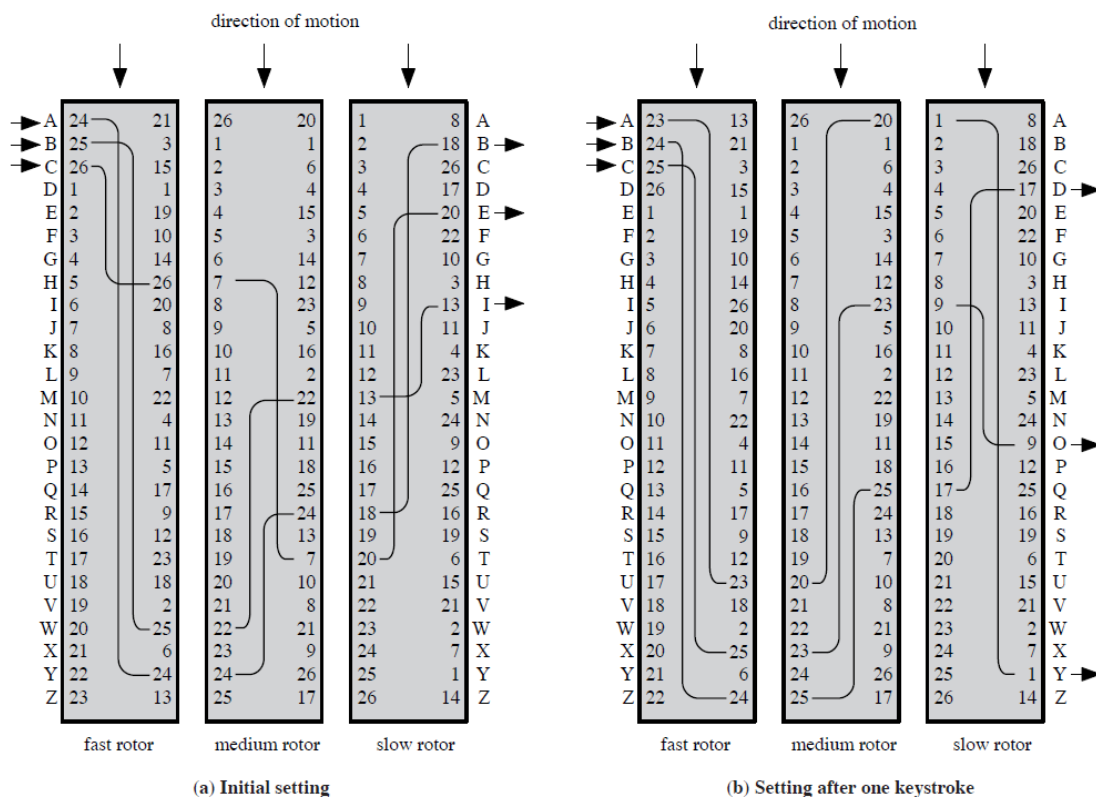


Figure 2.7 Three-Rotor Machine With Wiring Represented by Numbered Contacts

2. **[4.5 marks]** **This question is to be done in teams of 2.** The team members will together randomly generate an s-box S and analyze it to construct its Difference Distribution Table (similar to Table 7 in Heys' paper) and find the best Differential Characteristic (similar to Figure 5 in the paper). Person 1 will randomly generate five 16-bit round keys (the round keys are to be kept secret) and will then encrypt 10,000 known 16-bit plaintext strings, resulting in 10,000 corresponding 16-bit ciphertext strings. Person 2 will follow the process described in Heys' tutorial for differential cryptanalysis to compute a table similar to Table 8 in order to learn one byte of the final round key. Person 1 and Person 2 will then switch roles and repeat the process. Discuss the results of your attacks.

[Note that a student can work individually on this question if s/he is unable to find a partner. Simply pretend to be Person 1, and then pretend to be Person 2.]

3. **[1.5 marks]** Using 10-bit primes p and q , an appropriate seed, and a simple calculator, show the first 15 outputs of the Blum, Blum, Shub (BBS) pseudorandom bit generator. What size primes would be required to match the security of DES? What size primes would be required to match the security of AES-128?

1. [1.5 marks] Consider the rotor system shown in the figure below. Assume that the first 20 characters of a plaintext string have already been encrypted (leading to the overall state shown in (b)). Choose the next 10 letters of plaintext and show the output from the 3rd rotor for each of these input letters.

[Note: your plaintext should be chosen independently by you.]

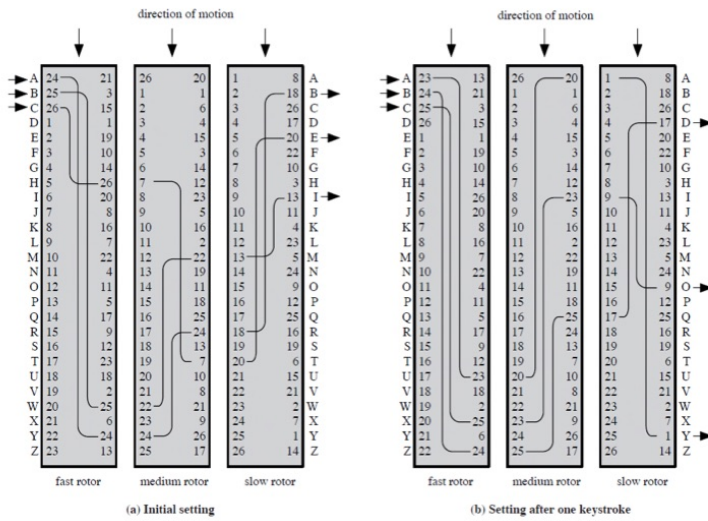
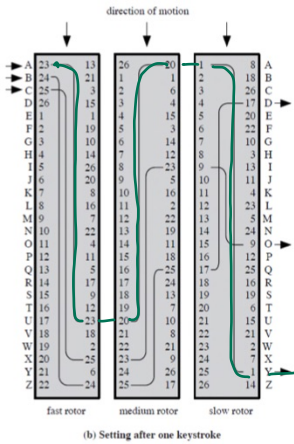


Figure 2.7 Three-Rotor Machine With Wiring Represented by Numbered Contacts

The next 10 letters of plaintext are: AQUAPHOBIA

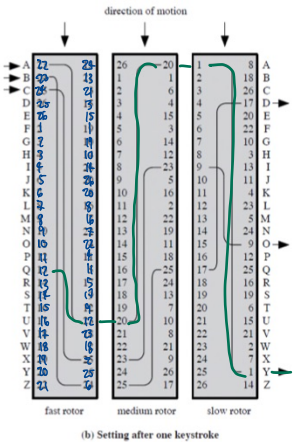
1. Initial state b)



A → U → A → Y

A → Y

2.

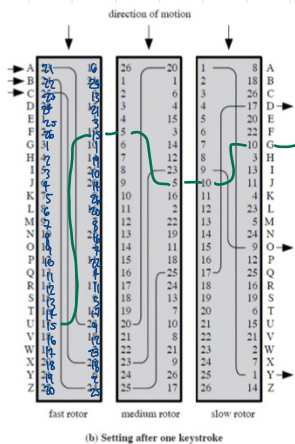


Rotate fast rotor down.

Q → U → A → Y

Q → Y

3.

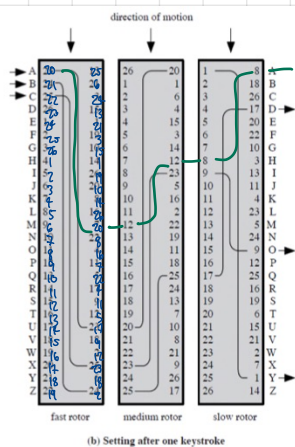


Rotate fast rotor down.

U → F → J → G

U → G

4.

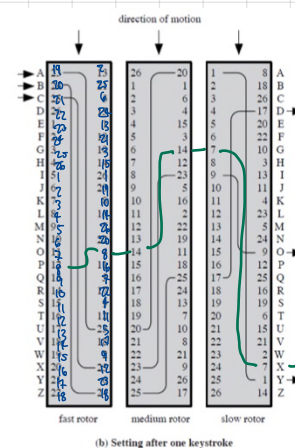


Rotate fast rotor down.

A → M → H → A

A → A

5.

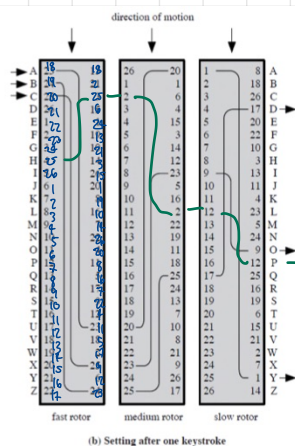


Rotate fast rotor down.

P → O → G → X

P → X

6.

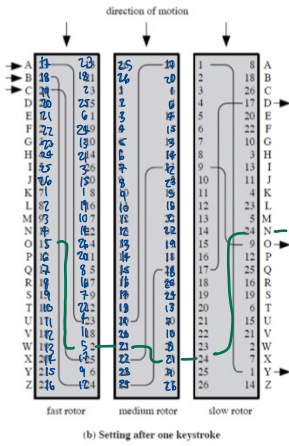


Rotate fast rotor down.

H → C → L → P

H → P

7.



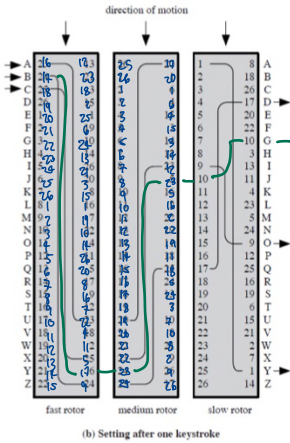
Rotate fast rotor down.

Since the fast rotor rotated 26 times, rotor the medium rotor down.

O → W → X → N

O → N

8.

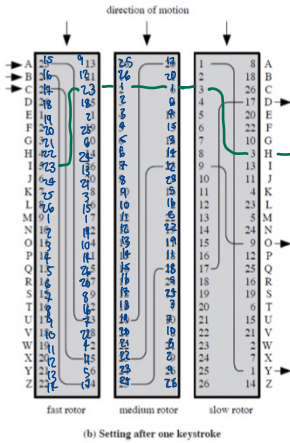


Rotate fast rotor down.

B → Y → J → G

B → G

9.

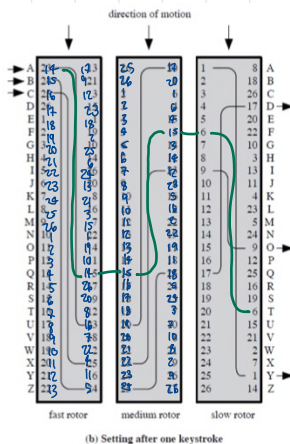


Rotate fast rotor down.

I → C → C → H

I → H

10.



Rotate fast rotor down.

A → Q → F → T

A → T

The next 10 encrypted characters are: YYGAXPNGHT

2. [4.5 marks] **This question is to be done in teams of 2.** The team members will together randomly generate an s-box S and analyze it to construct its Difference Distribution Table (similar to Table 7 in Heys' paper) and find the best Differential Characteristic (similar to Figure 5 in the paper). Person 1 will randomly generate five 16-bit round keys (the round keys are to be kept secret) and will then encrypt 10,000 known 16-bit plaintext strings, resulting in 10,000 corresponding 16-bit ciphertext strings. Person 2 will follow the process described in Heys' tutorial for differential cryptanalysis to compute a table similar to Table 8 in order to learn one byte of the final round key. Person 1 and Person 2 will then switch roles and repeat the process. Discuss the results of your attacks.

[Note that a student can work individually on this question if s/he is unable to find a partner. Simply pretend to be Person 1, and then pretend to be Person 2.]

This question was completed with Samuel Brie (7660408).

All code for this question was included in the submission on Brightspace as A2.zip

There are 8 files in the .zip: A2 - Cryp.py, A2-Decrypt.py, A2-Diff.py, A2-Encrypt.py, Encrypt.txt, Keys.txt, Result.txt, Bob.txt

We both generated different five 16-bit round keys (Keys.txt) and encrypted different plaintexts (Encrypt.txt being the plaintext and Result.txt being the ciphertext). Using the S-box we generated together, we each computed a table similar to Table 8 in order to learn one byte of the other's final round key.

Most of the logic is found in the provided code. The encryption and decryption functionalities of the SPN are found in A2-Encrypt.py and A2-Decrypt.py respectively.

The S-box mappings we used in our code:

Before	After
0000	0110
0001	0101
0010	1100
0011	1101
0100	0001
0101	1111
0110	1000
0111	1001
1000	1011
1001	1110

1010	0100
1011	0111
1100	0010
1101	0000
1110	0011
1111	1010

A2-Diff.py generates a Differential Distribution Table for the previously described S-box mappings. Each element in the table represents the number of occurrences of an output difference value for a given input difference. Using the probabilities in this table, A2-Diff.py identifies the “paths” with the highest probabilities (as a printed array).

		Output Difference															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Input Difference	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	4	2	4	0	2	0	0	0	2	0	0	0	0	2	0
	2	0	2	0	0	0	0	2	0	2	4	4	0	0	0	0	2
	3	0	0	0	2	0	0	0	2	4	2	2	2	2	0	0	0
	4	0	0	0	0	4	0	0	4	0	2	2	0	0	2	2	0
	5	0	0	0	0	4	4	0	0	0	2	0	2	2	0	2	0
	6	0	0	2	0	2	0	2	2	2	0	0	0	2	2	2	0
	7	0	2	0	2	2	2	0	0	0	0	0	0	2	4	0	2
	8	0	0	0	4	0	0	0	0	2	0	2	4	0	2	0	2
	9	0	2	2	0	0	0	0	0	2	2	2	2	0	2	2	0
	A	0	0	6	2	0	2	0	2	0	2	2	0	0	0	0	0
	B	0	4	2	0	0	0	2	0	2	0	0	4	2	0	0	0
	C	0	2	0	0	2	2	0	2	0	0	2	0	2	0	2	2
	D	0	0	0	0	2	0	4	2	0	0	0	0	0	2	2	4
	E	0	0	0	2	0	4	0	2	2	0	0	0	0	2	2	2
	F	0	0	2	0	0	0	6	0	0	0	0	2	4	0	0	2

The code determined the following path as the “best” path with the highest probability (0.0234375): ['Delta X0: 0000101000000000], ['Delta Y0: 0000001000000000], ['Perm/Delta X1: 000000000010000000], ['Delta Y1: 000000000010000000], ['Perm/Delta X2: 0000001000000000], ['Delta Y2: 0000100100000000], ['Perm/Delta X3: 01000000000000100],

Using A2-Cryp.py, I launched an attack to determine one byte of the final round key, taking into consideration a deltaP of 0000101000000000 and a targetP of 01000000000000100.

$P \oplus P' = \text{deltaP}$ is generated using the plaintext, which produces a C and a C'. To find one byte of the final round key, a partial decryption is performed for each partial subkey in the span of [K1...K4, K12...K16] (this is in accordance with Delta X3 being 01000000000000100).

After attempting all possible combinations, the highest bias occurs for a partial subkey value of 1 6 (00010110), which has a bias of 0.0227, a value close to the expected value of 0.0234375. 00010110 is exactly one byte of the final round key in Keys.txt.

Partial Subkey [K1...K4, K12...K16]	Bias (Probability)	Partial Subkey [K1...K4, K12...K16]	Bias (Probability)
0 E	0.0005	1 A	0.0015
0 F	0.0019	1 B	0.0019
1 0	0.0034	1 C	0.0036
1 1	0.0075	1 D	0.0033
1 2	0.0095	1 E	0.0008
1 3	0.0155	1 F	0.0041
1 4	0.0068	2 0	0.0017
1 5	0.0033	2 1	0.0014
1 6	0.0227	2 2	0.0021
1 7	0.0108	2 3	0.0021
1 8	0.003	2 4	0.0019
1 9	0.0005	2 5	0.0015

3. [1.5 marks] Using 10-bit primes p and q , an appropriate seed, and a simple calculator, show the first 15 outputs of the Blum, Blum, Shub (BBS) pseudorandom bit generator. What size primes would be required to match the security of DES? What size primes would be required to match the security of AES-128?

Step 1: Choose primes p & q such that $p \equiv q \equiv 3 \pmod{4}$

$$p = 523 = 1000001011 \text{ (in binary)} \quad 523 \equiv 3 \pmod{4}$$

$$q = 547 = 1000100011 \text{ (in binary)} \quad 547 \equiv 3 \pmod{4}$$

Step 2: Compute $n = p \cdot q$

$$n = p \cdot q = 286081 = 1000101110110000001 \text{ (in binary)}$$

Step 3: Choose a random s that is relatively prime to n

$$s = 30 = 11110 \text{ (in binary)}$$

Step 4: Compute $X_0 = s^2 \pmod{n}$ (X_0 is the seed value)

$$X_0 = 30^2 \pmod{286081} = 900 = 1110000100 \text{ (in binary)}$$

$$X_1 = (X_0)^2 \pmod{n}$$

$$= (900)^2 \pmod{286081}$$

$$= 237838 = 111010000100001110 \text{ (in binary)}$$

$$b_1 = 237838 \pmod{2}$$

$$= 0$$

$$X_2 = (X_1)^2 \pmod{n}$$

$$= (237838)^2 \pmod{286081}$$

$$= 118114 = 11100110101100010 \text{ (in binary)}$$

$$b_2 = 118114 \pmod{2}$$

$$= 0$$

$$X_3 = (X_2)^2 \pmod{n}$$

$$= (118114)^2 \pmod{286081}$$

$$= 177031 = 101011001110000111 \text{ (in binary)}$$

$$b_3 = 177031 \pmod{2}$$

$$= 1$$

$$X_4 = (X_3)^2 \pmod{n}$$

$$= (177031)^2 \pmod{286081}$$

$$= 87492 = 10101010111000100 \text{ (in binary)}$$

$$b_4 = 87492 \pmod{2}$$

$$= 0$$



$$\begin{aligned}
 X_5 &= (X_4)^2 \bmod n \\
 &= (87492)^2 \bmod 286081 \\
 &= 180747 = 101100001000001011 \text{ (in binary)}
 \end{aligned}$$

$$\begin{aligned}
 b_5 &= 180747 \bmod 2 \\
 &= 1
 \end{aligned}$$

$$\begin{aligned}
 X_6 &= (X_5)^2 \bmod n \\
 &= (180747)^2 \bmod 286081 \\
 &= 172133 = 101010000001100101 \text{ (in binary)}
 \end{aligned}$$

$$\begin{aligned}
 b_6 &= 172133 \bmod 2 \\
 &= 1
 \end{aligned}$$

$$\begin{aligned}
 X_7 &= (X_6)^2 \bmod n \\
 &= (172133)^2 \bmod 286081 \\
 &= 74438 = 10010001011000110 \text{ (in binary)}
 \end{aligned}$$

$$\begin{aligned}
 b_7 &= 74438 \bmod 2 \\
 &= 0
 \end{aligned}$$

$$\begin{aligned}
 X_8 &= (X_7)^2 \bmod n \\
 &= (74438)^2 \bmod 286081 \\
 &= 199036 = 11000010010111100 \text{ (in binary)}
 \end{aligned}$$

$$\begin{aligned}
 b_8 &= 199036 \bmod 2 \\
 &= 0
 \end{aligned}$$

$$\begin{aligned}
 X_9 &= (X_8)^2 \bmod n \\
 &= (199036)^2 \bmod 286081 \\
 &= 262821 = 1000000001010100101 \text{ (in binary)}
 \end{aligned}$$

$$\begin{aligned}
 b_9 &= 262821 \bmod 2 \\
 &= 1
 \end{aligned}$$

$$\begin{aligned}
 X_{10} &= (X_9)^2 \bmod n \\
 &= (262821)^2 \bmod 286081 \\
 &= 48429 = 1011110100101101 \text{ (in binary)}
 \end{aligned}$$

$$\begin{aligned}
 b_{10} &= 48429 \bmod 2 \\
 &= 1
 \end{aligned}$$

$$\begin{aligned}
 X_{11} &= (X_{10})^2 \bmod n \\
 &= (48429)^2 \bmod 286081 \\
 &= 76003 = 10010100011100011 \text{ (in binary)}
 \end{aligned}$$

$$\begin{aligned}
 b_{11} &= 76003 \bmod 2 \\
 &= 1
 \end{aligned}$$

$$\begin{aligned}
 X_{12} &= (X_{11})^2 \bmod n \\
 &= (76003)^2 \bmod 286081 \\
 &= 194538 = 10111101111101010 \text{ (in binary)}
 \end{aligned}$$

$$\begin{aligned}
 b_{12} &= 194538 \bmod 2 \\
 &= 0
 \end{aligned}$$

$$X_{13} = (X_{12})^2 \bmod n \\ = (194538)^2 \bmod 286081 \\ = 236197 = 111001101010100101 \text{ (in binary)}$$

$$b_{13} = 236197 \bmod 2 \\ = 1$$

$$X_{14} = (X_{13})^2 \bmod n \\ = (236197)^2 \bmod 286081 \\ = 80918 = 10011110000010110 \text{ (in binary)}$$

$$b_{14} = 80918 \bmod 2 \\ = 0$$

$$X_{15} = (X_{14})^2 \bmod n \\ = (80918)^2 \bmod 286081 \\ = 186877 = 10110110011111101 \text{ (in binary)}$$

$$b_{15} = 186877 \bmod 2 \\ = 1$$

$$\therefore b_1 = 0, b_2 = 0, b_3 = 1, b_4 = 0, b_5 = 1, b_6 = 1, b_7 = 0, b_8 = 0, b_9 = 1, b_{10} = 1, b_{11} = 1, \\ b_{12} = 0, b_{13} = 1, b_{14} = 0, b_{15} = 1$$

The security of BBS is generally based on the complexity of factoring, which is:

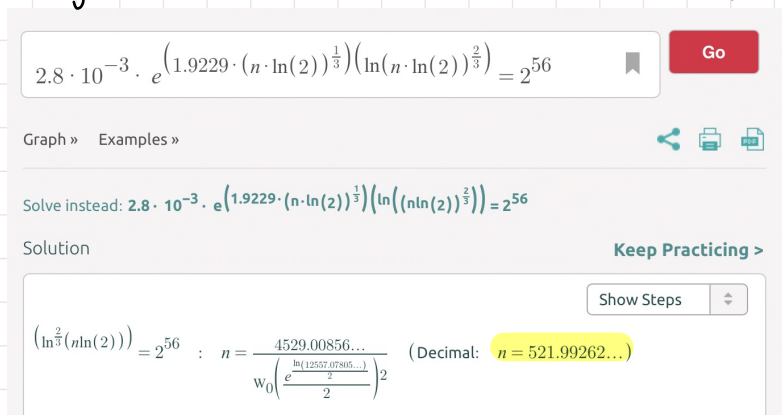
Let $L(n)$ be the number of clock cycles needed to factor an n -bit integer. We assume that $L(n) \approx \gamma \exp(1.9229(n \ln 2)^{1/3} (\ln(n \ln 2))^{2/3})$. Experience from available data points suggests that $L(512) \approx 3 \cdot 10^{17}$ clock cycles, therefore $\gamma \approx 2.8 \cdot 10^{-3}$ and

$$\text{Formula (A): } L(n) \approx 2.8 \cdot 10^{-3} \cdot \exp(1.9229(n \ln 2)^{1/3} (\ln(n \ln 2))^{2/3}). \quad (8)$$

← (From https://link.springer.com/chapter/10.1007/11586821_24)
(I sent this formula and link to the professor over email and he approved it).

In DES, the key length is 56 bits, so there are 2^{56} possible keys.

Using an online calculator to determine for which value of n does Formula (A) = 2^{56} :



Graph » Examples »

Solve instead: $2.8 \cdot 10^{-3} \cdot e^{(1.9229 \cdot (n \cdot \ln(2))^{1/3} (\ln(n \cdot \ln(2)))^{2/3})} = 2^{56}$

Solution

Keep Practicing >

Show Steps

$(\ln^{\frac{2}{3}}(n \ln(2))) = 2^{56} : n = \frac{4529.00856...}{w_0\left(\frac{e^{\frac{\ln(12557.07605...)}{2}}}{2}\right)^2}$ (Decimal: $n = 521.99262...$)

← (The professor, over email, approved the use of an online calculator).

$\therefore n$ equals 521.99262... So to match the security of DES, the primes should be at least of size 522 bits.

Note that NIST previously indicated, with consideration of standard block sizes, that AES-128 (with a 128-bit security level) is considered equivalent to RSA with a 3072-bit key (Table 10.3 of 6th edition textbook).



In AES-128, the key length is 128 bits, so there are 2^{128} possible keys.
 Using an online calculator to determine for which value of n does Formula ① = 2^{128} :

$2.8 \cdot 10^{-3} \cdot e^{\left(1.9229((n)\ln(2))^{\frac{1}{3}}(\ln((n) \cdot \ln(2)))^{\frac{2}{3}}\right)} = 2^{128}$

Graph » Examples »

Solution [Keep Practicing >](#)

$\ln(2))^{\frac{2}{3}} = 2^{128} : n = \frac{42946.84712...}{w_0\left(\frac{e^{\frac{\ln(119673.94400...)}{2}}}{2}\right)^2}$ (Decimal: $n = 2954.91159...$)

Show Steps

$\therefore n$ equals 2954.91159... So to match the security of AES-128, the primes should be at least of size 2955 bits.