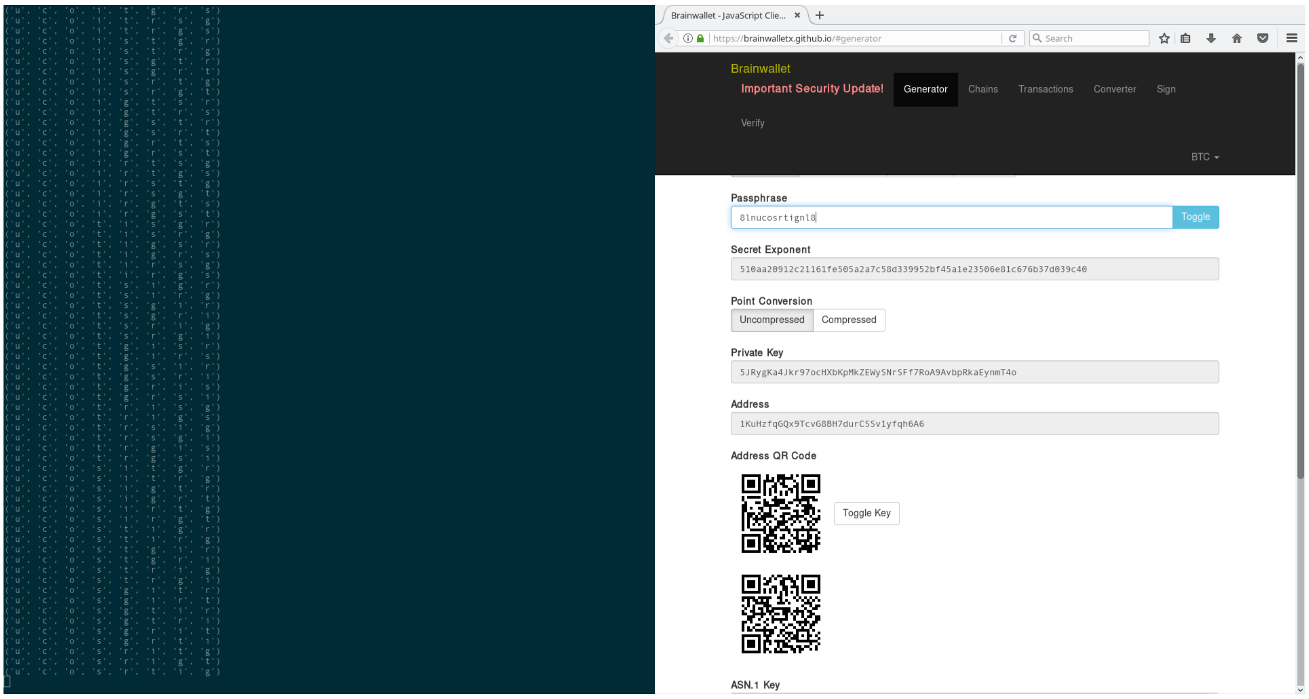


```
('u', 'c', 'o', 'i', 't', 'g', 'r', 's')
('u', 'c', 'o', 'i', 't', 'r', 's', 'g')
('u', 'c', 'o', 'i', 't', 'r', 'g', 's')
('u', 'c', 'o', 'i', 's', 't', 'g', 'r')
('u', 'c', 'o', 'i', 's', 't', 'r', 'g')
('u', 'c', 'o', 'i', 's', 'g', 't', 'r')
('u', 'c', 'o', 'i', 's', 'g', 'r', 't')
('u', 'c', 'o', 'i', 's', 'r', 't', 'g')
('u', 'c', 'o', 'i', 's', 'r', 'g', 't')
('u', 'c', 'o', 'i', 'g', 't', 's', 'r')
('u', 'c', 'o', 'i', 'g', 't', 'r', 's')
('u', 'c', 'o', 'i', 'g', 's', 't', 'r')
('u', 'c', 'o', 'i', 'g', 's', 'r', 't')
('u', 'c', 'o', 'i', 'g', 'r', 't', 's')
('u', 'c', 'o', 'i', 'g', 'r', 's', 't')
('u', 'c', 'o', 'i', 'r', 't', 's', 'g')
('u', 'c', 'o', 'i', 'r', 't', 'g', 's')
('u', 'c', 'o', 'i', 'r', 's', 't', 'g')
('u', 'c', 'o', 'i', 'r', 's', 'g', 't')
('u', 'c', 'o', 'i', 'r', 'g', 't', 's')
('u', 'c', 'o', 'i', 'r', 'g', 's', 't')
('u', 'c', 'o', 't', 'i', 's', 'g', 'r')
('u', 'c', 'o', 't', 'i', 's', 'r', 'g')
('u', 'c', 'o', 't', 'i', 'g', 's', 'r')
('u', 'c', 'o', 't', 'i', 'g', 'r', 's')
('u', 'c', 'o', 't', 'i', 'r', 's', 'g')
('u', 'c', 'o', 't', 'i', 'r', 'g', 's')
('u', 'c', 'o', 't', 's', 'i', 'g', 'r')
('u', 'c', 'o', 't', 's', 'i', 'r', 'g')
('u', 'c', 'o', 't', 's', 'g', 'i', 'r')
('u', 'c', 'o', 't', 's', 'g', 'r', 'i')
('u', 'c', 'o', 't', 's', 'r', 'i', 'g')
('u', 'c', 'o', 't', 's', 'r', 'g', 'i')
('u', 'c', 'o', 't', 'g', 'i', 's', 'r')
('u', 'c', 'o', 't', 'g', 'i', 'r', 's')
('u', 'c', 'o', 't', 'g', 's', 'i', 'r')
('u', 'c', 'o', 't', 'g', 's', 'r', 'i')
('u', 'c', 'o', 't', 'g', 'r', 'i', 's')
('u', 'c', 'o', 't', 'g', 'r', 's', 'i')
('u', 'c', 'o', 't', 'r', 'i', 's', 'g')
('u', 'c', 'o', 't', 'r', 'i', 'g', 's')
('u', 'c', 'o', 't', 'r', 's', 'i', 'g')
('u', 'c', 'o', 't', 'r', 's', 'g', 'i')
('u', 'c', 'o', 't', 'r', 'g', 'i', 's')
('u', 'c', 'o', 't', 'r', 'g', 's', 'i')
('u', 'c', 'o', 's', 'i', 't', 'g', 'r')
('u', 'c', 'o', 's', 'i', 't', 'r', 'g')
('u', 'c', 'o', 's', 'i', 'g', 't', 'r')
('u', 'c', 'o', 's', 'i', 'g', 'r', 't')
('u', 'c', 'o', 's', 'i', 'r', 't', 'g')
('u', 'c', 'o', 's', 'i', 'r', 'g', 't')
('u', 'c', 'o', 's', 't', 'i', 'g', 'r')
('u', 'c', 'o', 's', 't', 'i', 'r', 'g')
('u', 'c', 'o', 's', 't', 'g', 'i', 'r')
('u', 'c', 'o', 's', 't', 'g', 'r', 'i')
('u', 'c', 'o', 's', 't', 'r', 'i', 'g')
('u', 'c', 'o', 's', 't', 'r', 'g', 'i')
('u', 'c', 'o', 's', 'g', 'i', 't', 'r')
('u', 'c', 'o', 's', 'g', 'i', 'r', 't')
('u', 'c', 'o', 's', 'g', 't', 'i', 'r')
('u', 'c', 'o', 's', 'g', 't', 'r', 'i')
('u', 'c', 'o', 's', 'g', 'r', 'i', 't')
('u', 'c', 'o', 's', 'g', 'r', 't', 'i')
('u', 'c', 'o', 's', 'r', 'i', 't', 'g')
('u', 'c', 'o', 's', 'r', 'i', 'g', 't')
('u', 'c', 'o', 's', 'r', 't', 'i', 'g')
```

Passphrase: 8lnucosrtignl8

Secret Exponent: 510aa20912c21161fe505a2a7c58d339952bf45a1e23506e81c676b37d039c40

Point Conversion: [✓ Uncompressed] Compressed

Private Key: 5JRygKa4Jkr97ocHXbKpMkZEWySNrSFf7RoA9AvbpRkaEynmT4o

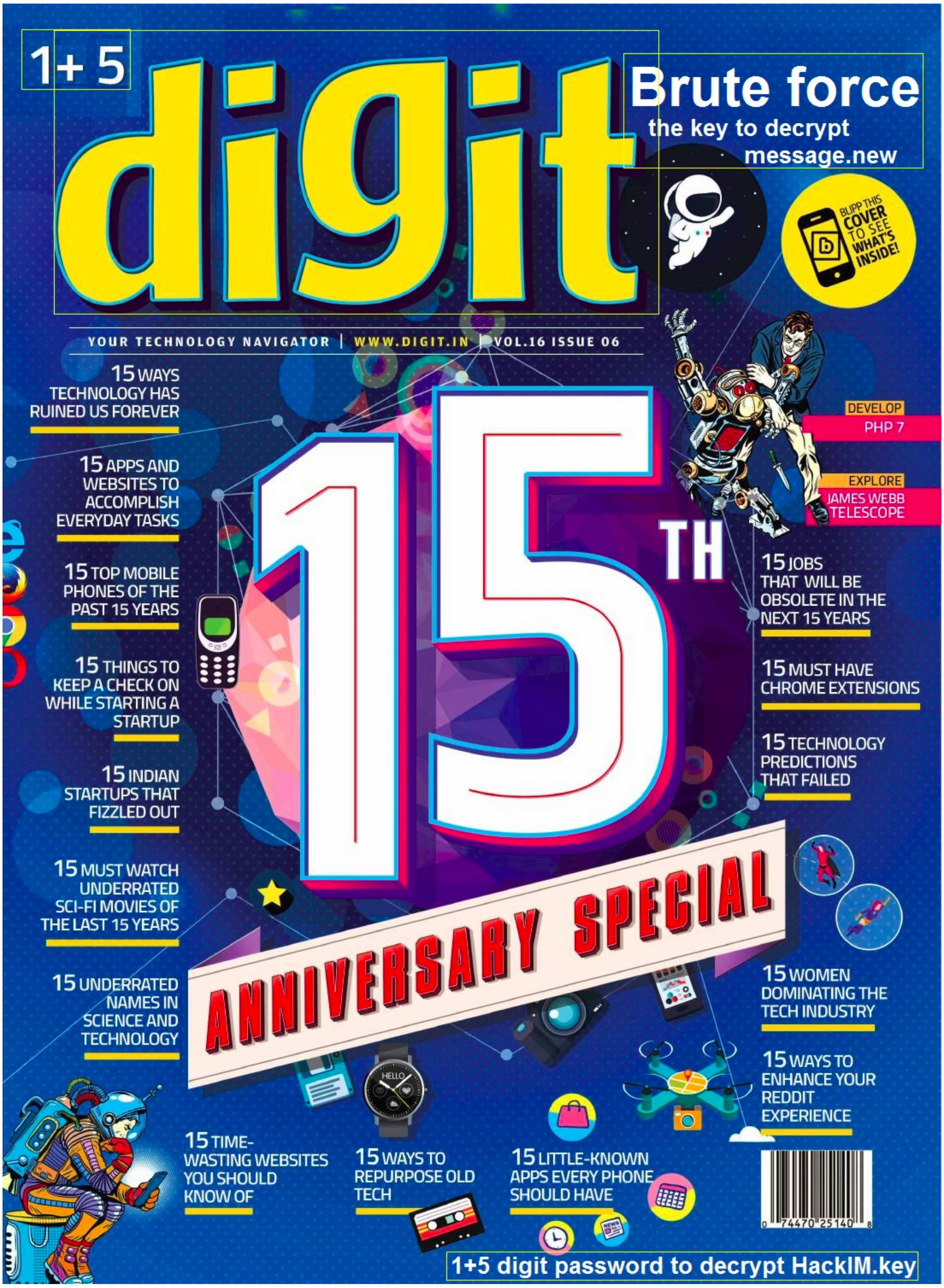
Address: 1KuHzfqGQx9TcvG8BH7durCSSv1yfqh6A6

This is what it looks like, the pass field gets filled up automatically every second, just as if I had typed it.

And when I came back… The passphrase was done! I honestly forgot what the passphrase was, but the resulting private key was `5KjzfnM4afWU8fJeUgGnxKbtG5FHtr6Suc41juGMUmQKC7WYzEG`, and that's the flag!

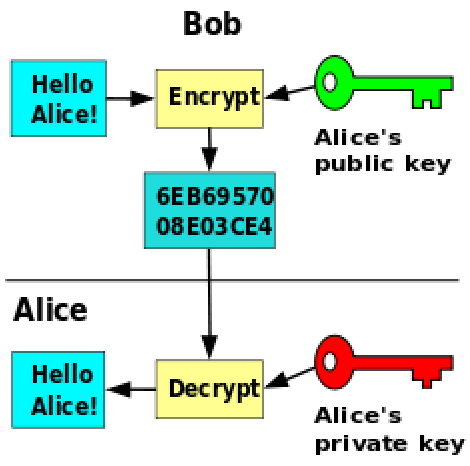
Crypto 3: Next Step

This challenge consists of an encrypted file, `message.new`, and a private key, `HackIM.key`. We were also given a picture: the cover of the Digits magazine for June 2016, with a few modifications. The challenge wwas worth 200 points. I'll go in detail about openssl and private keys, because overall otherwise the challenge was fairly short and easy.


1+ 5
digit
Brute force
the key to decrypt
message.new
BUPP THIS COVER TO SEE WHAT'S INSIDE!
YOUR TECHNOLOGY NAVIGATOR | WWW.DIGIT.IN | VOL.16 ISSUE 06
15 WAYS TECHNOLOGY HAS RUINED US FOREVER
15 APPS AND WEBSITES TO ACCOMPLISH EVERYDAY TASKS
15 TOP MOBILE PHONES OF THE PAST 15 YEARS
15 THINGS TO KEEP A CHECK ON WHILE STARTING A STARTUP
15 INDIAN STARTUPS THAT FIZZLED OUT
15 MUST WATCH UNDERRATED SCI-FI MOVIES OF THE LAST 15 YEARS
15 UNDERRATED NAMES IN SCIENCE AND TECHNOLOGY
15TH
ANNIVERSARY SPECIAL
DEVELOP PHP 7
EXPLORE JAMES WEBB TELESCOPE
15 JOBS THAT WILL BE OBSOLETE IN THE NEXT 15 YEARS
15 MUST HAVE CHROME EXTENSIONS
15 TECHNOLOGY PREDICTIONS THAT FAILED
15 WOMEN DOMINATING THE TECH INDUSTRY
15 WAYS TO ENHANCE YOUR REDDIT EXPERIENCE
15 TIME-WASTING WEBSITES YOU SHOULD KNOW OF
15 WAYS TO REPURPOSE OLD TECH
15 LITTLE-KNOWN APPS EVERY PHONE SHOULD HAVE
1+5 digit password to decrypt HackIM.key
This feels very 90's to me for some reason.

Public/private key pairs, commonly called public key cryptography, are a solution to a simple problem: how do I send a message to someone over a potentially insecure connection, without being able to agree on a passphrase beforehand? If it requires a key of some sort (a ‘key’ being any data that you can use to encrypt or decrypt the message, such as a string of characters), I could just send the key over to you, then encrypt my message with it, but anyone who has intercepted the key could also decrypt the message.

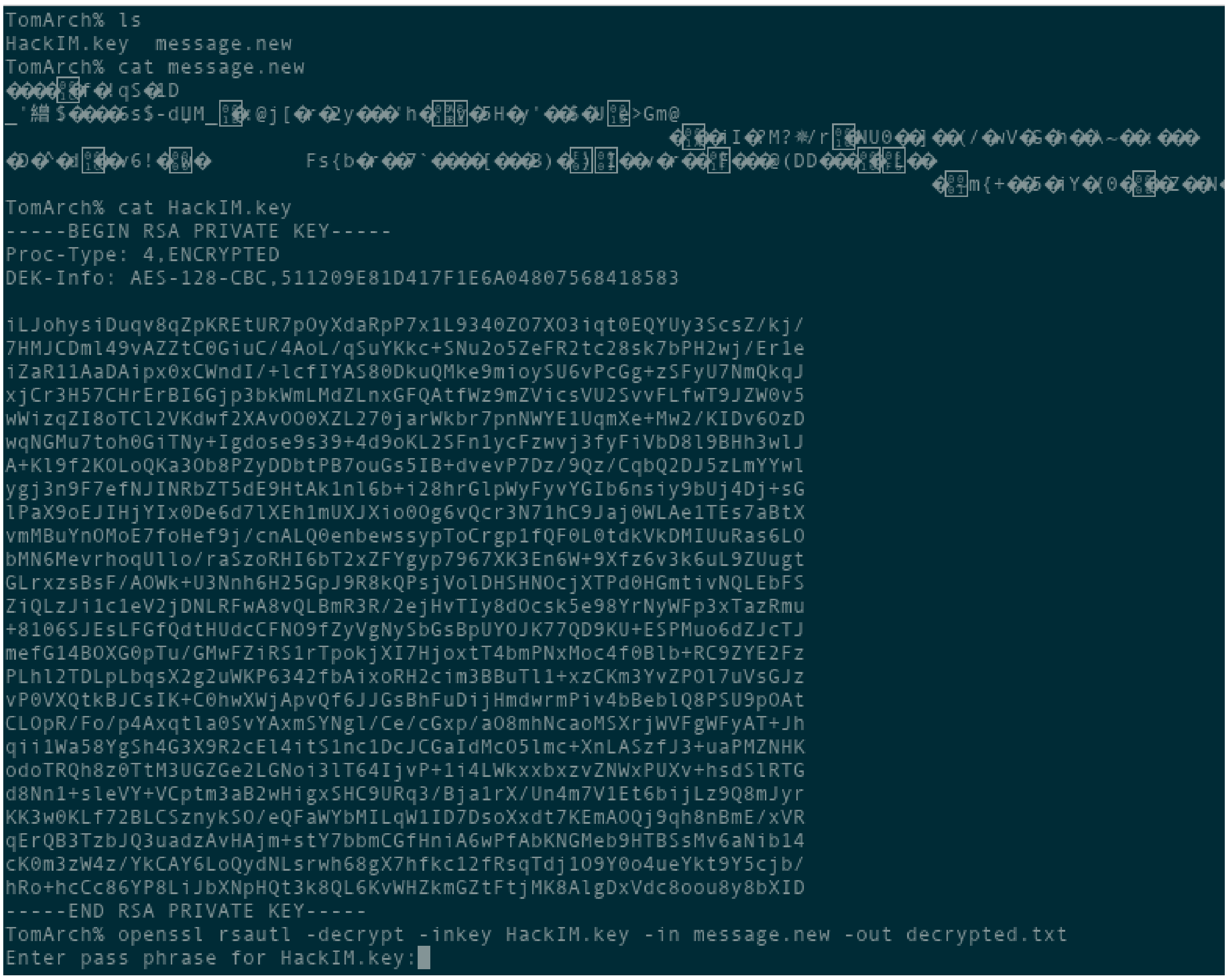
Public key cryptography solves this problem with assymetric encryption. Let’s say Alice wants people to be able to send messages to her securely. She generates two keys: A private and public key. The public key is used to encode the data, and then **only** the private key can decode it. Alice puts her public key out in the open for all to see, and keeps her private key secret. Now anyone can encrypt a message meant for alice with her public key and send it over an insecure connection. Even if it is intercepted, the attacker won’t have access to the private key, they therefore cannot decrypt the data!



Bob and Alice are very secretive about saying hello.

Key pairs are an important part of the SSL protocol, which is dedicated to providing secure connections between peers. To add another layer of security, private keys can be protected by a password, so that even if an user’s private key fell in the wrong hands (like the private key we received for this challenge), no one could use it without the password. An usual way of manipulating key pairs is using `openssl`, an open source toolkit that implements SSL.

So with all that in mind, let’s try and decrypt the message we got with openssl!



```
TomArch% ls
HackIM.key  message.new
TomArch% cat message.new
����[illegible]�f�qS�D
_'[illegible]$����s$-dUM_[illegible]@j[�r�2y���'h�[illegible]�5H�y'��6�U[illegible]>Gm@
                                                         �[illegible]�iI�?M?[illegible]/r[illegible]NU0��]��(/�wV�6�n��~��[illegible]���
�D�^�d[illegible]�v6!�[illegible]�        Fs{b�r��7`����[���B)�[illegible]��v�r��[illegible]����(DD���[illegible]��
                                                                                 �[illegible]m{+��5�iY�(0�[illegible]�Z��N
TomArch% cat HackIM.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,511209E81D417F1E6A04807568418583

iLJohysiDuqv8qZpKREtUR7pOyXdaRpP7x1L9340ZO7XO3iqt0EQYUy3ScsZ/kj/
7HMJCDml49vAZZtC0GiuC/4AoL/qSuYKkc+SNu2o5ZeFR2tc28sk7bPH2wj/Er1e
iZaR11AaDAipx0xCWndI/+lcfIYAS80DkuQMke9mioySU6vPcGg+zSFyU7NmQkqJ
xjCr3H57CHrErBI6Gjp3bkWmLMdZLnxGFQAtfWz9mZVicsVU2SvvFLfwT9JZW0v5
wWizqZI8oTCl2VKdwf2XAvOO0XZL270jarWkbr7pnNWYE1UqmXe+Mw2/KIDv6OzD
wqNGMu7toh0GiTNy+Igdose9s39+4d9oKL2SFn1ycFzwvj3fyFiVbD8l9BHh3wlJ
A+Kl9f2KOLoQKa3Ob8PZyDDbtPB7ouGs5IB+dvevP7Dz/9Qz/CqbQ2DJ5zLmYYwl
ygj3n9F7efNJINRbZT5dE9HtAk1nl6b+i28hrGlpWyFyvYGIb6nsiy9bUj4Dj+sG
lPaX9oEJIHjYIx0De6d7lXEh1mUXJXio0Og6vQcr3N71hC9Jaj0WLAe1TEs7aBtX
vmMBuYnOMoE7foHef9j/cnALQ0enbewssypToCrgp1fQF0L0tdkVkDMIUuRas6LO
bMN6MevrhoqUllo/raSzoRHI6bT2xZFYgyp7967XK3En6W+9Xfz6v3k6uL9ZUugt
GLrxzsBsF/AOWk+U3Nnh6H25GpJ9R8kQPsjVolDHSHNOcjXTPd0HGmtivNQLEbFS
ZiQLzJi1c1eV2jDNLRFwA8vQLBmR3R/2ejHvTIy8dOcsk5e98YrNyWFp3xTazRmu
+8106SJEsLFGfQdtHUdcCFNO9fZyVgNySbGsBpUYOJK77QD9KU+ESPMuo6dZJcTJ
mefG14BOXG0pTu/GMwFZiRS1rTpokjXI7HjoxtT4bmPNxMoc4f0Blb+RC9ZYE2Fz
PLhl2TDLpLbqsX2g2uWKP6342fbAixoRH2cim3BBuTl1+xzCKm3YvZPOl7uVsGJz
vP0VXQtkBJCsIK+C0hwXWjApvQf6JJGsBhFuDijHmdwrmPiv4bBeblQ8PSU9pOAt
CLOpR/Fo/p4Axqtla0SvYAxmSYNgl/Ce/cGxp/aO8mhNcaoMSXrjWVFgWFyAT+Jh
qii1Wa58YgSh4G3X9R2cEl4itS1nc1DcJCGaIdMcO5lmc+XnLASzfJ3+uaPMZNHK
odoTRQh8z0TtM3UGZGe2LGNoi3lT64IjvP+1i4LWkxxbxzvZNWxPUXv+hsdSlRTG
d8Nn1+sleVY+VCptm3aB2wHigxSHC9URq3/Bja1rX/Un4m7V1Et6bijLz9Q8mJyr
KK3w0KLf72BLCSznykSO/eQFaWYbMILqW1ID7DsoXxdt7KEmAOQj9qh8nBmE/xVR
qErQB3TzbJQ3uadzAvHAjm+stY7bbmCGfHniA6wPfAbKNGMeb9HTBSsMv6aNib14
cK0m3zW4z/YkCAY6LoQydNLsrwh68gX7hfkc12fRsqTdj1O9Y0o4ueYkt9Y5cjb/
hRo+hcCc86YP8LiJbXNpHQt3k8QL6KvWHZkmGZtFtjMK8AlgDxVdc8oou8y8bXID
-----END RSA PRIVATE KEY-----
TomArch% openssl rsautl -decrypt -inkey HackIM.key -in message.new -out decrypted.txt
Enter pass phrase for HackIM.key:
```

I really need a green-on-black theme to be a true hacker.

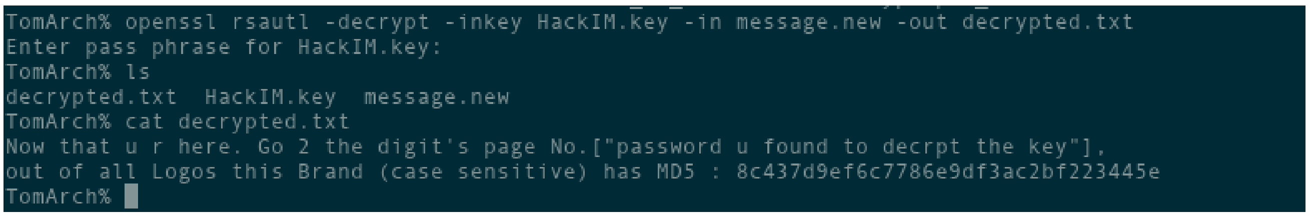
Oh no, it's password protected! :c I should mention in a regular attack this could be a bit of a headache, even bruteforcing the password would take me a while. Luckily, the Digit cover that I showed above says that the password to decrypt `HackIM.key` is "1 + 5 digits". That can be understood as "1, then 5 digits", so 1xxxxx or as just 6 digits, and since I don't really trust whoever wrote it, I'll be safe and go for 6 digits.

Ok so we know the password is 6 digits. Looking up a way to test RSA private key passwords in code yields this Sackoverflow thread which has the exact code I need - my script is actually pretty much a copy of the top reply. Let's code our bruteforcer! We just need to try every possible 6 digit number.

```
import paramiko
import itertools
from paramiko import rsakey

kf = open("HackIM.key", "r")

```

```
dlist = itertools.product(['1', '2', '3', '4','5', '6',
                           '7', '8', '9', '0'], repeat=6)

for d in dlist:
    s = ''.join(d)

    kf.seek(0)
    try:
        nk = rsakey.RSAKey.from_private_key(kf, password=s)
        print("success: " + s)
        break
    except paramiko.ssh_exception.SSHException:
        print("fail: " + s)
```

We run it and it cracks the key in a few seconds - and it turns out the passphrase did have a '1' as its first digit! It's `141525`. We can now decrypt the message:



```
TomArch% openssl rsautl -decrypt -inkey HackIM.key -in message.new -out decrypted.txt
Enter pass phrase for HackIM.key:
TomArch% ls
decrypted.txt  HackIM.key  message.new
TomArch% cat decrypted.txt
Now that u r here. Go 2 the digit's page No.["password u found to decrpt the key"],
out of all Logos this Brand (case sensitive) has MD5 : 8c437d9ef6c7786e9df3ac2bf223445e
TomArch%
```

Well, this isn't what I expected.

Alright, so we have to go to page `141525` of the magazine (I actually tried it with 41525 at first because remember, the cover says it's 1 + a 5 digit password, so the 1 isn't part of the password, but it didn't work), I found the digit magazine here… But the pdf only has 124 pages. That's fine, we can just do a modulus, and it'll be as if we kept counting pages until we reached 141525:

```
141525 % 124 = 41
```

We go to page 41 and manually try the md5 hash of all of these brands… ans clearTax turns out to be it!

```
TomArch% echo -ne "clearTax" | md5sum
8c437d9ef6c7786e9df3ac2bf223445e  -
```

And that’s it, `clearTax` was the flag! Overall I’d say, CTF makers, please, skip stuff like the magazine search. It actually was a bit of a pain, between the two interpretations of what the password is, the different possible ways to interpret what “page 141525” is, and having to manually enter brand names for hashes, the flag might as well just have been in that text file. But it was still a fun little bruteforcing I suppose.

Thanks for reading, I hope you enjoyed my two little writeups!

Edit: I have since read some other writeup and it turns out that when you open the magazine in a certain website, the correct logo is on pages “14-15/25”, which spells out 141525. If this is how it was meant to be solved, it’s very convoluted and I was pretty lucky I got this flag with my modulus…

Contact me!