WPA, Asymmetric Cypto Review, and TLS

So far...

- Symmetric crypto review
- Saw problems with CBC mode
- WEP has problems
  - Key too small
  - Reusing key material is always bad for stream ciphers
  - RC4 with weak IVs reveals key
    - Not just keystream, but *key*
  - Key is shared by everybody

Coming up

- WPA introduced TKIP, only some basic improvements on WEP but not fundamentally different
- WPA2 switched to a counter mode based on AES, still a stream cipher
- WPA3 introduced forward secrecy
- Asymmetric crypto review
- TLS for comparison
  - Confidentiality, Integrity, Authentication, Non-repudiation without pre-shared secret

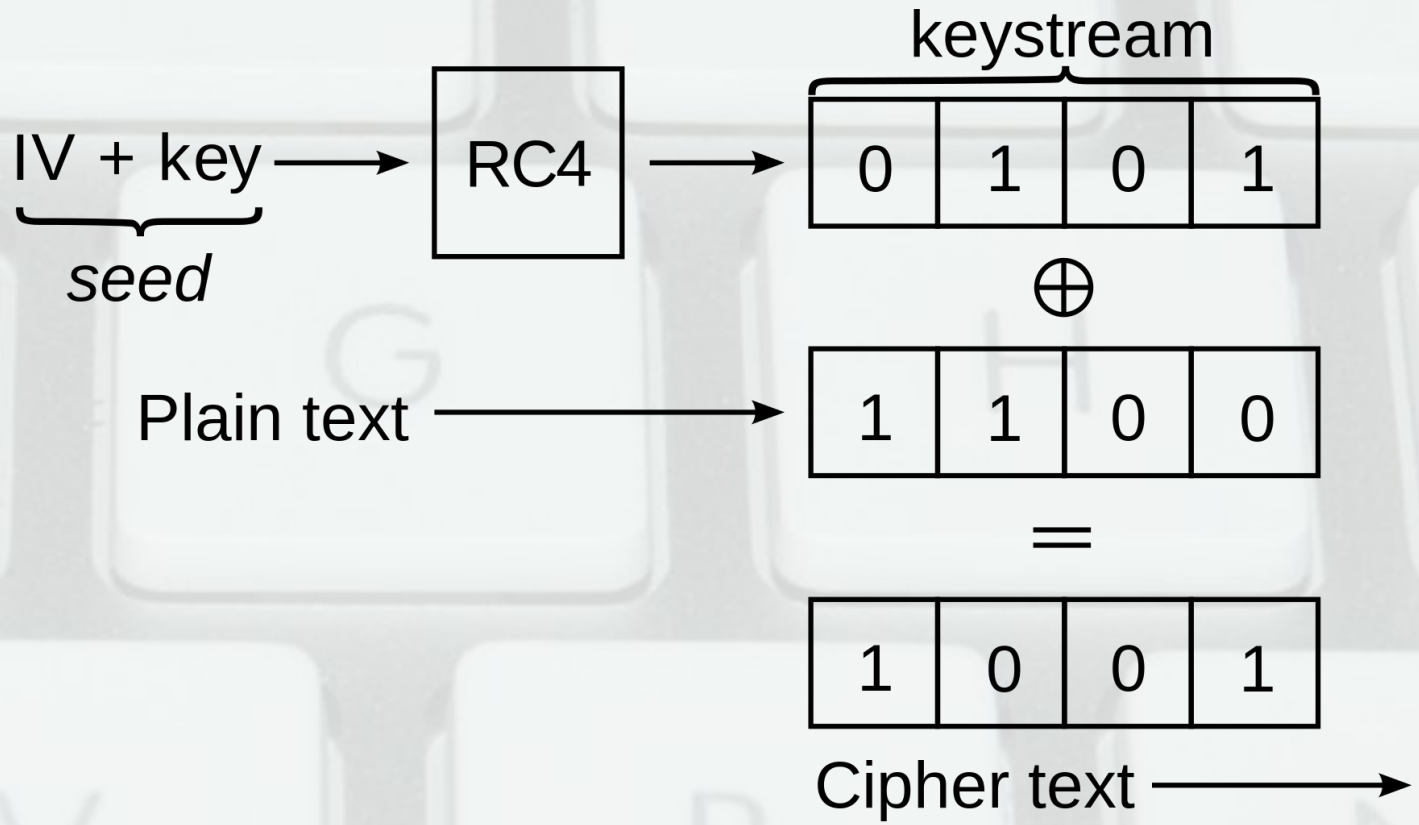
Why these specific topics?

- Network security begins in the physical layer, WiFi encryption is meant to make the physical layer more like wires
  - “Wired Equivalent Privacy”
- You should develop a healthy distrust of any crypto, even if it was developed by a reputable organization
- You should realize that even well-done crypto doesn’t always have certain desirable properties

Types of cryptanalysis...

- Symmetric attack types according to outdated textbooks: Ciphertext-only, known plaintext (*e.g.*, linear cryptanalysis), and chosen plaintext (*e.g.*, differential cryptanalysis)
  - Often forget chosen ciphertext for, *e.g.*, padding oracles
- Asymmetric desired properties: Indistinguishability under Chosen Plaintext (IND-CPA), Chosen Ciphertext (IND-CCA, IND-CCA2)
  - *E.g.*, malleability of RSA (need something like OAEP)
- Machine-in-the-middle attacks, birthday attacks, attacks on hash functions, *etc.* (above list is not exhaustive)

WEP



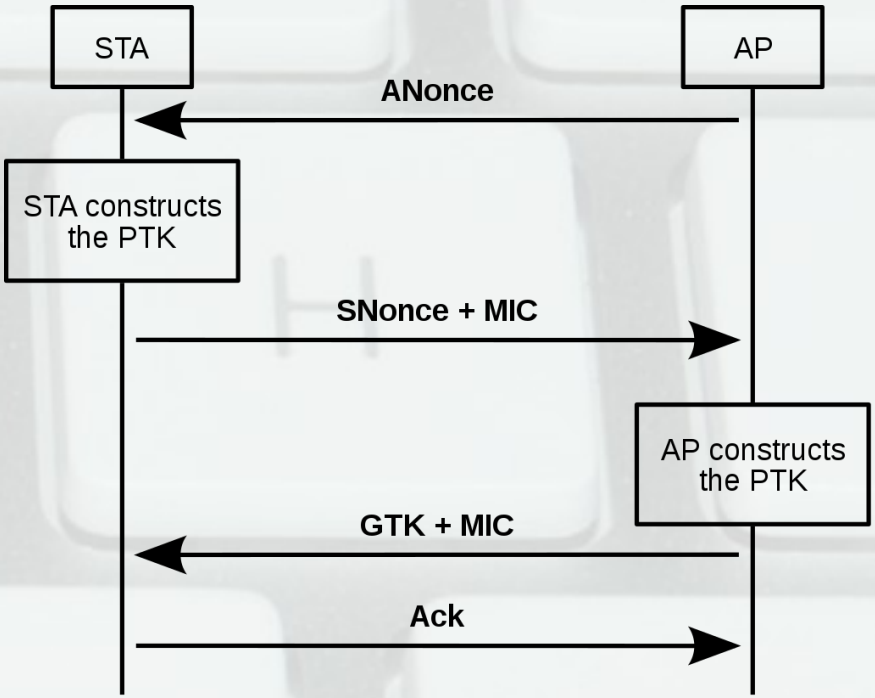
https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy#/media/File:Wep-crypt-alt.svg

Wi-Fi Protected Access (WPA)

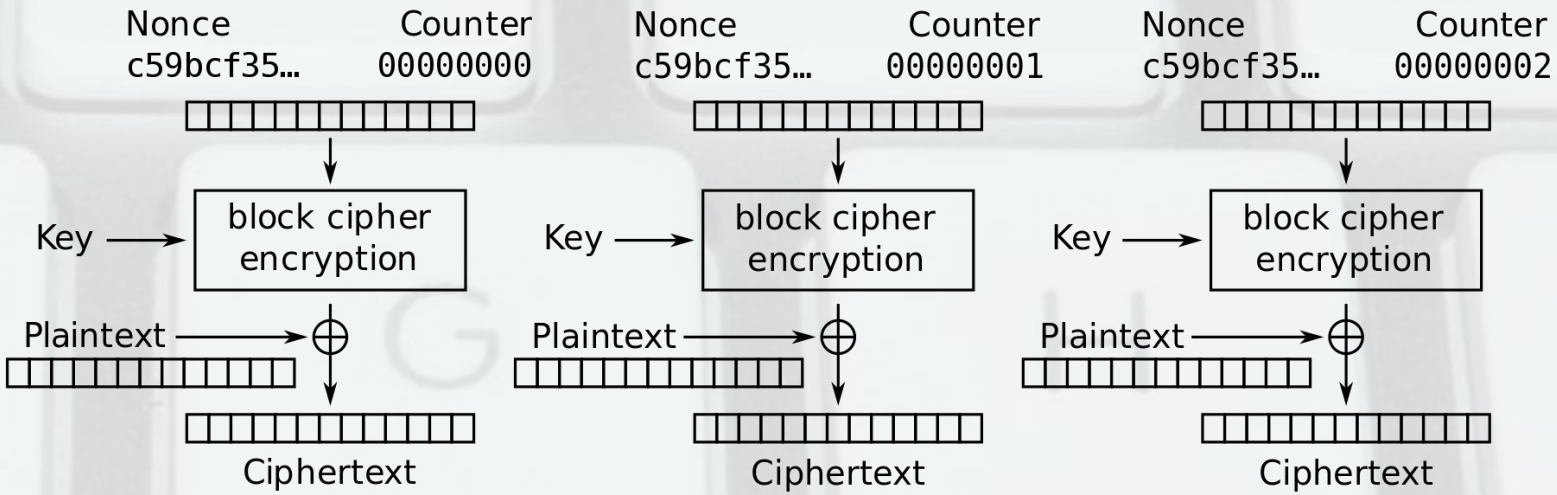
- TKIP: Temporal Key Integrity Protocol
- Same hardware, same RC4
- Key mixing function combines IV and key
- 64-bit Message Integrity Check (MIC)
- Deprecated because of attacks, but none that compromised the key itself

WPA2

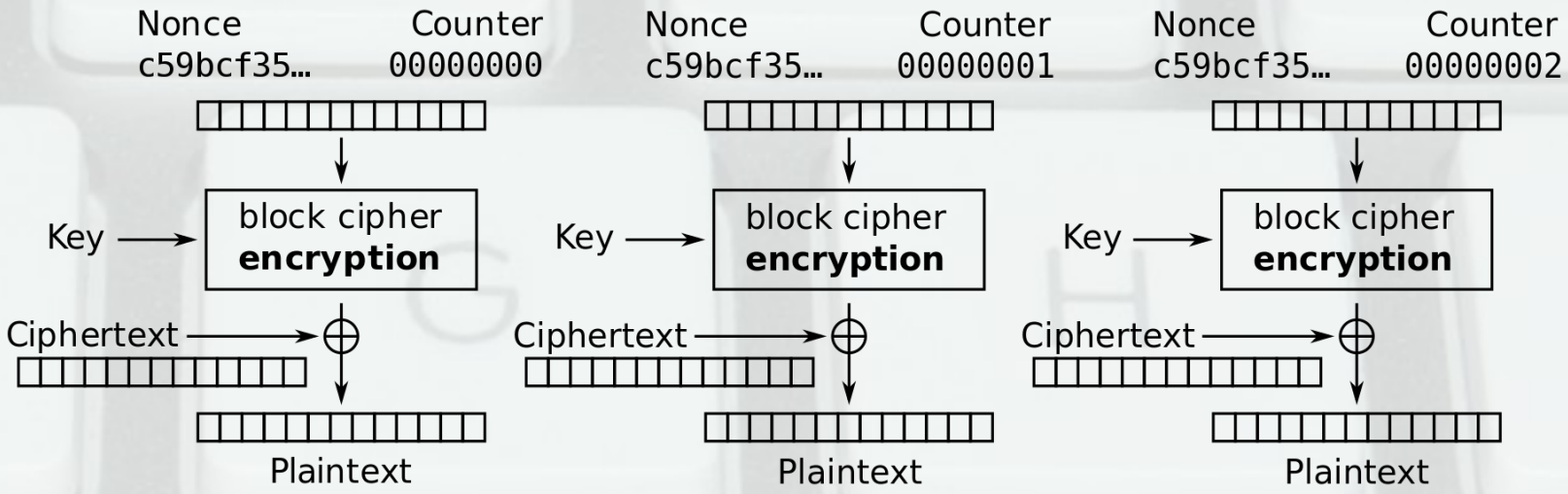
- Main changes: CCMP and 4-way handshake
- CCMP is like CTR counter mode with 128-bit AES, basically



Following images are from:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Counter_.28CTR.29



Counter (CTR) mode encryption

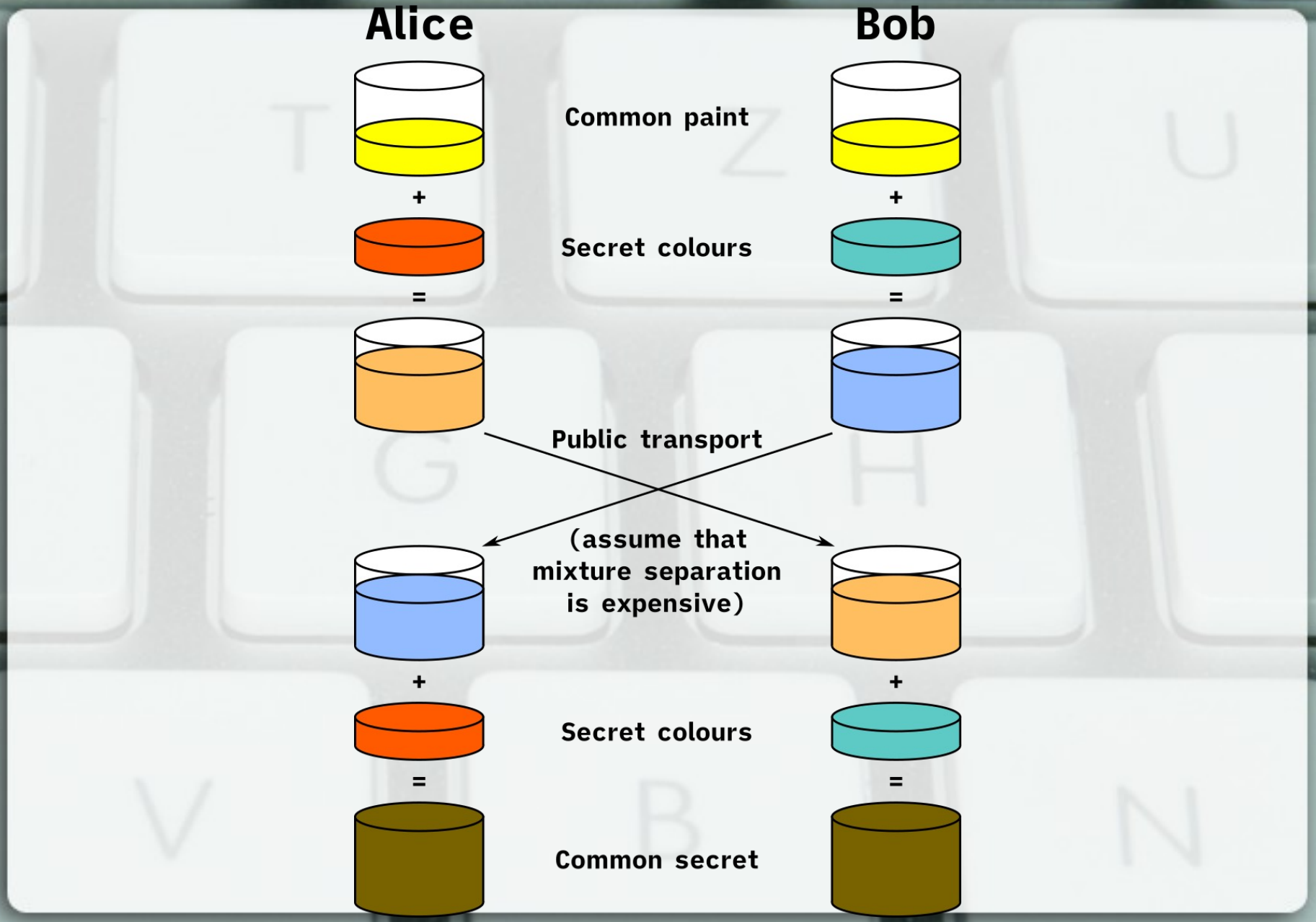


Counter (CTR) mode decryption

WPA3

- AES with 128/256-bit key in GCM mode (also like CTR mode)
- SHA-384 as HMAC
- Simultaneous Authentication of Equals, basically a Diffie-Hellman Key Exchange
  - Forward secrecy
  - Stronger authentication is possible...

Asymmetric crypto review...



https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange#/media/File:Diffie-Hellman_Key_Exchange.svg

Diffie-Hellman

Alice		Bob		Eve	
Known	**Unknown**	**Known**	**Unknown**	**Known**	**Unknown**
$p = 23$		$p = 23$		$p = 23$	
$g = 5$		$g = 5$		$g = 5$	
$\mathbf{a = 6}$	$\mathbf{b}$	$\mathbf{b = 15}$	$\mathbf{a}$		$\mathbf{a}, \mathbf{b}$
$A = 5^{a} \bmod 23$		$B = 5^{b} \bmod 23$			
$A = 5^{6} \bmod 23 = 8$		$B = 5^{15} \bmod 23 = 19$			
$\mathbf{B = 19}$		$\mathbf{A = 8}$		$A = 8, B = 19$	
$\mathbf{s} = B^{a} \bmod 23$		$\mathbf{s} = A^{b} \bmod 23$			
$\mathbf{s} = 19^{6} \bmod 23 = \mathbf{2}$		$\mathbf{s} = 8^{15} \bmod 23 = \mathbf{2}$			$\mathbf{s}$

In the food coloring or paint demos, it is assumed that mixing colors is cheap, but *un-mixing* them is prohibitively expensive.

Modular arithmetic

$$5 + 7 = 2 \pmod{10}$$

$$7^2 = 9 \pmod{10}$$

$$8 + 8 = 6 \pmod{10}$$

Modular arithmetic

$$8 + 9 = ? \pmod{10}$$

$$4^3 = ? \pmod{10}$$

$$1 + 1 = ? \pmod{10}$$

Modular arithmetic

$$8 + 9 = 7 \pmod{10}$$

$$4^3 = 4 \pmod{10}$$

$$1 + 1 = 2 \pmod{10}$$

RSA

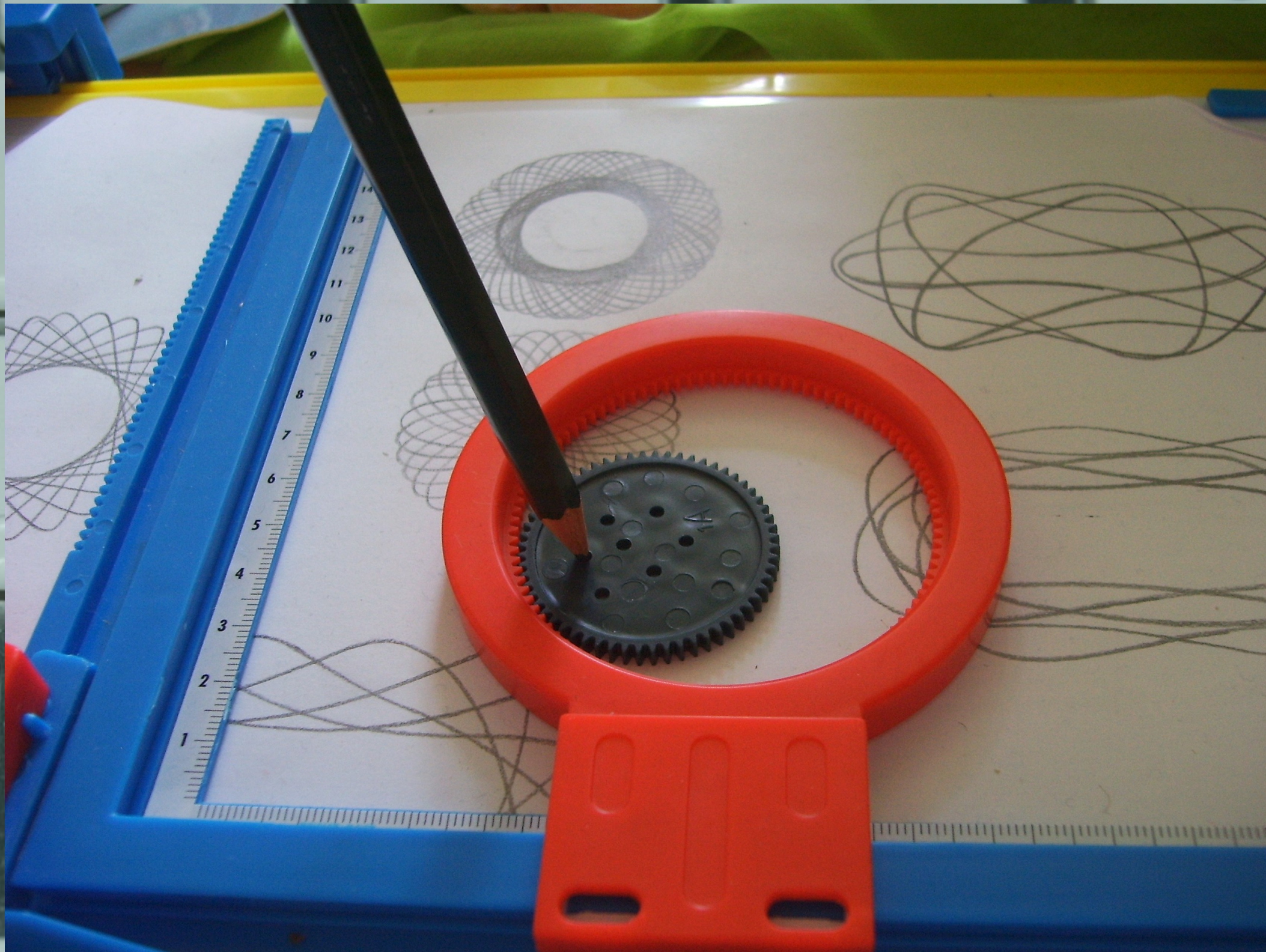
Encryption:

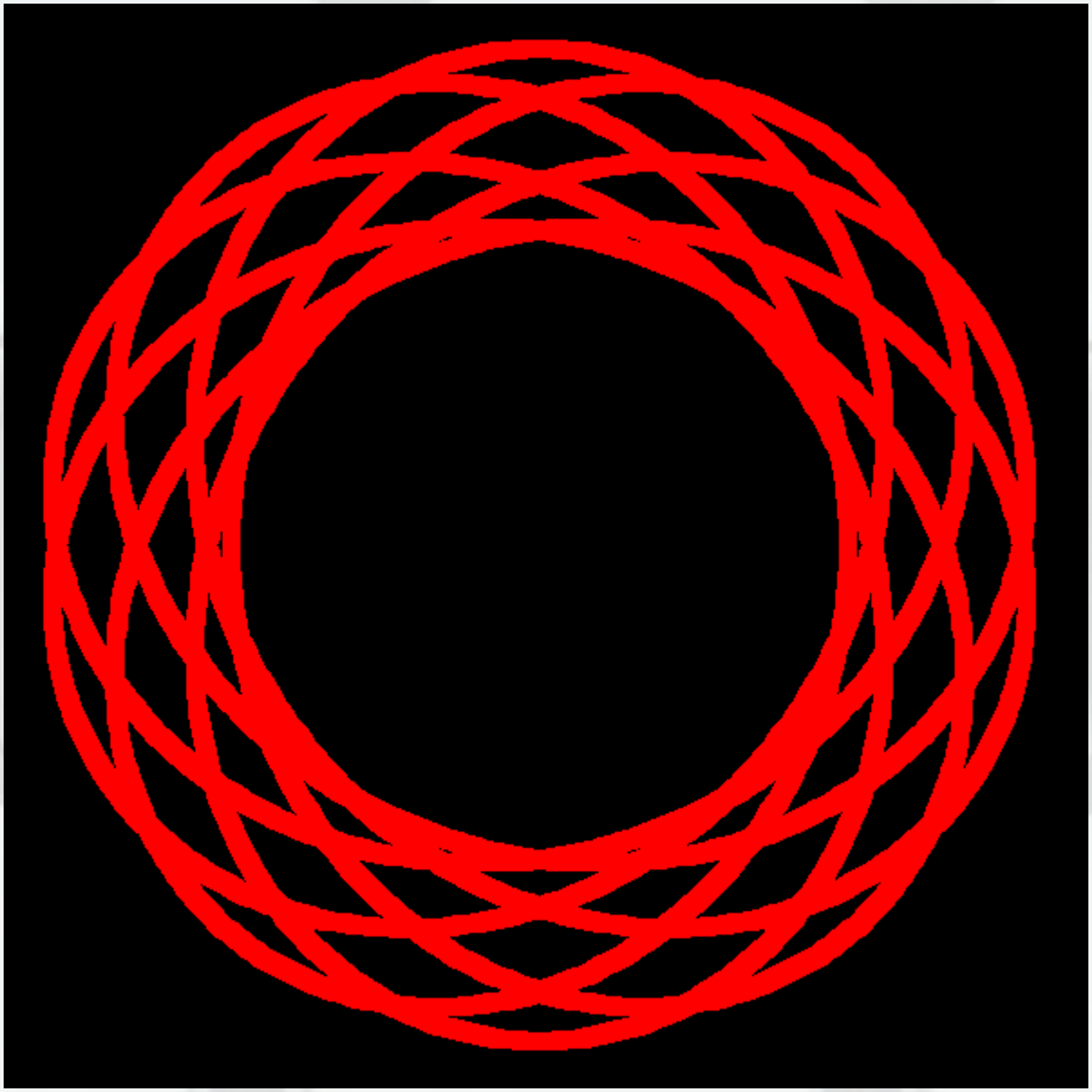
$$c \equiv m^e \bmod n$$

Decryption:

$$c^d \equiv (m^e)^d \bmod n$$

RSA provides encryption, authentication, and non-repudiation





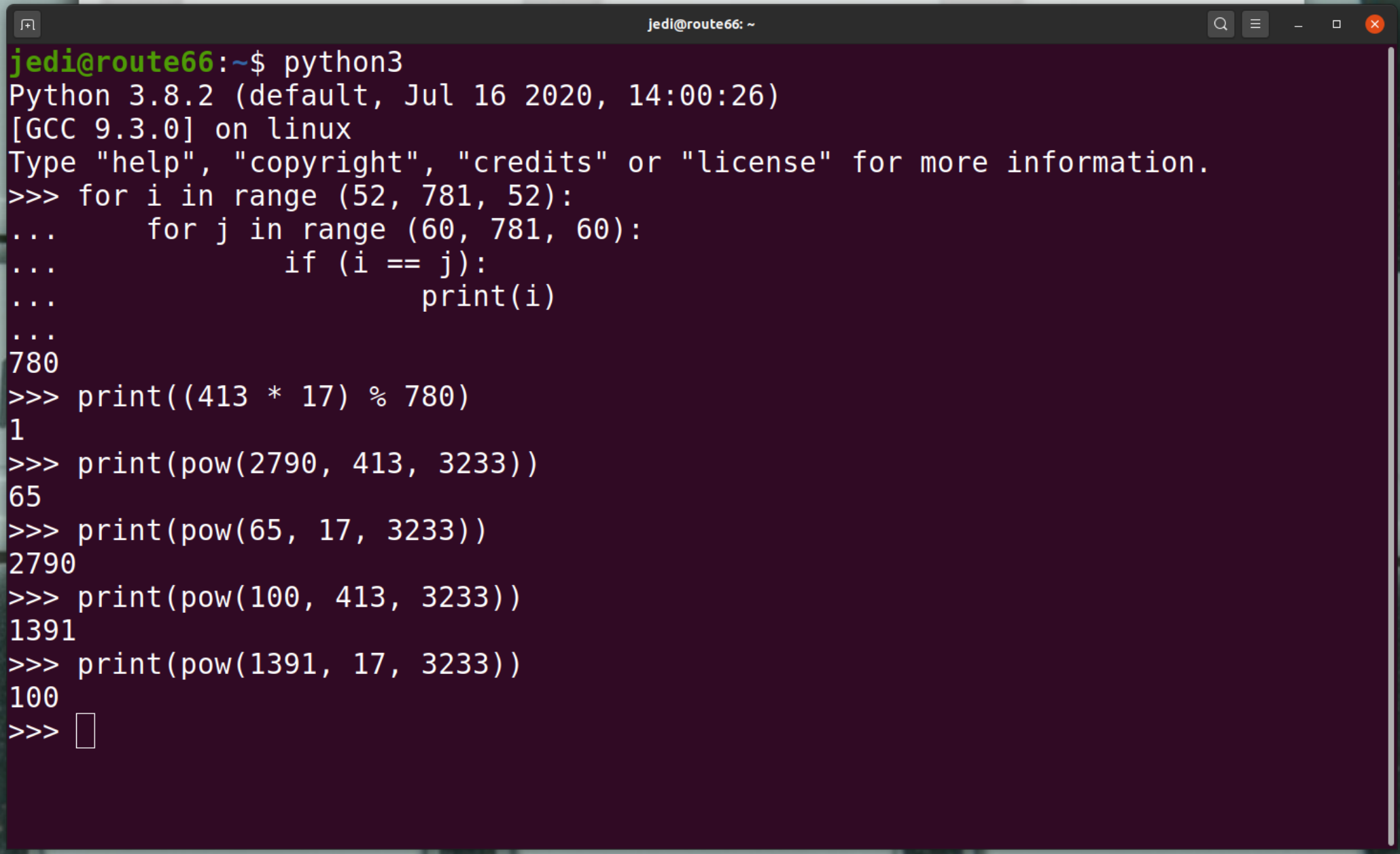
RSA

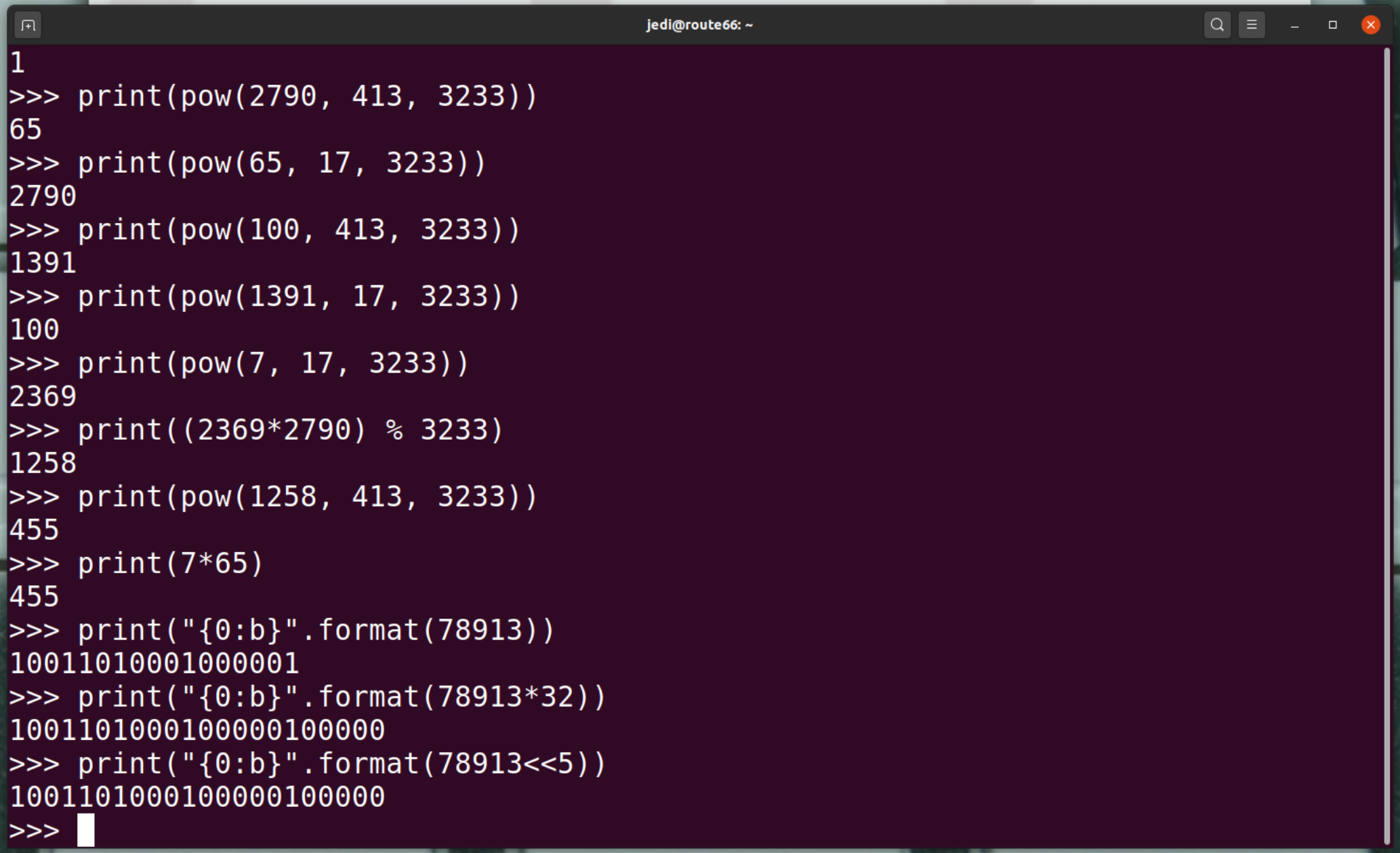
- Security is based on the hardness of integer factorization

n = pq

- p and q are primes, suppose p = 61, q = 53
- n = 3233
- Euler's totient counts the positive integers up to n that are relatively prime to n
- totient(n) = lcm(p – 1, q – 1) = 780
  - 52,104,156,208,260,312,364,416,468,520,572,624,676,728,780
  - 60,120,180,240,300,360,420,480,540,600,660,720,780
- Choose 1 < e < 780 coprime to 780, e.g., e = 17
- d is the modular multiplicative inverse of e, d = 413
- 413 * 17 mod 780 = 1

- Public key is (n = 3233, e = 17)
- Private key is (n = 3233, d = 413)
- Encryption: $c(m = 65) = 65^{17} \bmod 3233 = 2790$
- Decryption: $m = 2790^{413} \bmod 3233 = 65$
- Could also do...
  - Signature: $s = 100^{413} \bmod 3233 = 1391$
  - Verification: $100 = 1391^{17} \bmod 3233$
- Fast modular exponentiation is the trick
- Using RSA for key exchange or encryption is often a red flag, more commonly used for signatures


jedi@route66: ~
jedi@route66:~$ python3
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> for i in range (52, 781, 52):
...   for j in range (60, 781, 60):
...   if (i == j):
...   print(i)
...
780
>>> print((413 * 17) % 780)
1
>>> print(pow(2790, 413, 3233))
65
>>> print(pow(65, 17, 3233))
2790
>>> print(pow(100, 413, 3233))
1391
>>> print(pow(1391, 17, 3233))
100
>>>


jedi@route66: ~
1
>>> print(pow(2790, 413, 3233))
65
>>> print(pow(65, 17, 3233))
2790
>>> print(pow(100, 413, 3233))
1391
>>> print(pow(1391, 17, 3233))
100
>>> print(pow(7, 17, 3233))
2369
>>> print((2369*2790) % 3233)
1258
>>> print(pow(1258, 413, 3233))
455
>>> print(7*65)
455
>>> print("{0:b}".format(78913))
1001101000100001
>>> print("{0:b}".format(78913*32))
10011010001000001000000
>>> print("{0:b}".format(78913<<5))
10011010001000001000000
>>>

Let C be the RSA encryption of 128-bit AES key k with RSA public key (n, e). Thus, we have

$$C \equiv k^e \pmod{n}$$

Now let C_b be the RSA encryption of the AES key

$$k_b = 2^b k$$

i.e., k bitshifted to the left by b bits. Thus, we have

$$C_b \equiv {k_b}^e \pmod{n}$$

We can compute C_b from only C and the public key, as

$$\begin{aligned} C_b &\equiv C(2^{be} \bmod n) \pmod{n} \\ &\equiv (k^e \bmod n)(2^{be} \bmod n) \pmod{n} \\ &\equiv k^e 2^{be} \pmod{n} \\ &\equiv (2^b k)^e \pmod{n} \\ &\equiv {k_b}^e \pmod{n} \end{aligned}$$

Server chops off all but the lowest 128 bits

1. Record a session
2. Connect to the server with key shifted left 127 bits
3. Can you encrypt/decrypt with 1000000... or 0000000...?

(Just learned one bit of the key, repeat for left shift of 126 bits, 125 bits, etc. until you learn the key of the recorded session and can decrypt it)

This is a chosen ciphertext attack, and a padding oracle attack, but involves RSA padding rather than AES-CBC padding

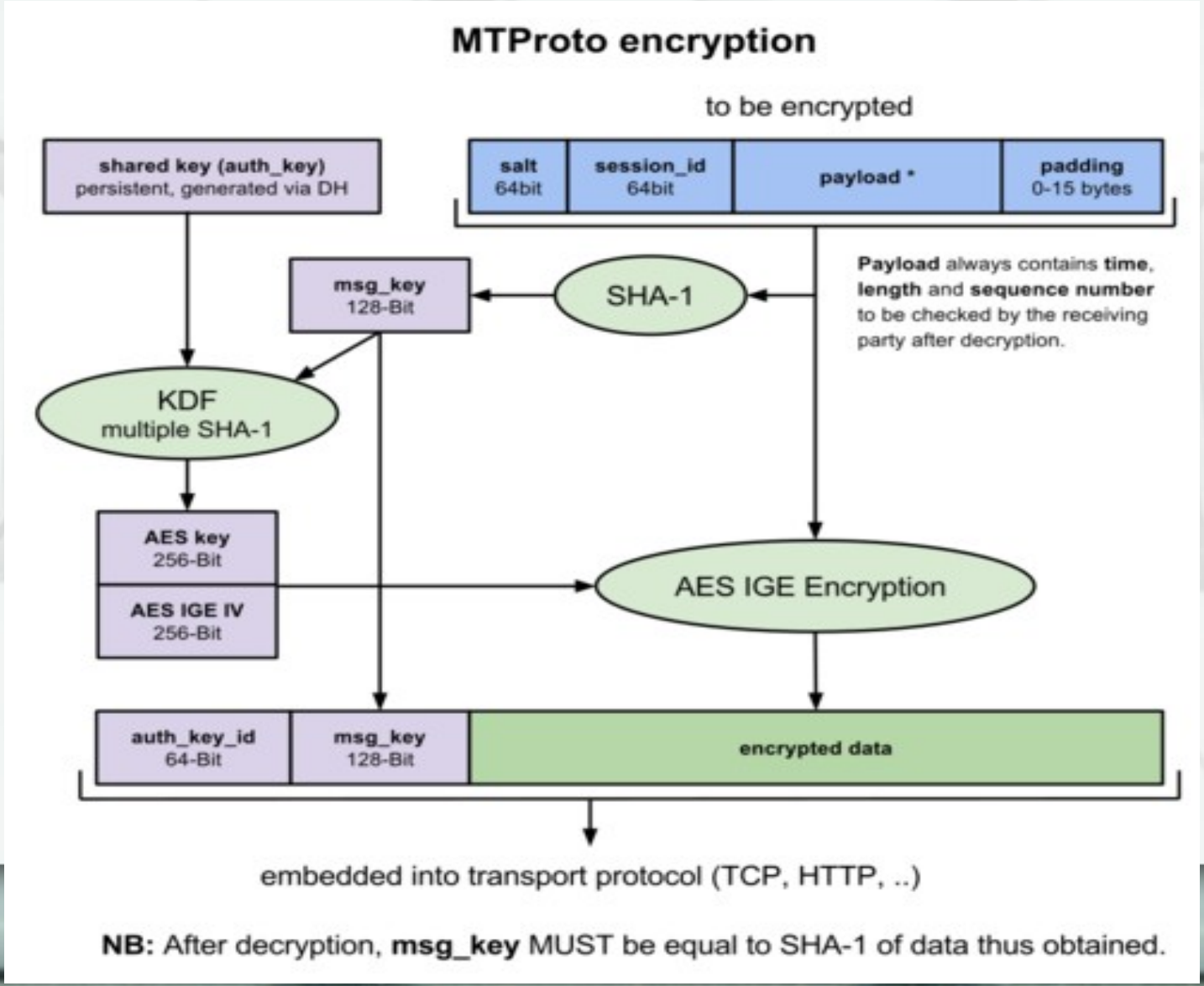
Also called a Bleichenbacher-style attack.

Semantic security (*e.g.*, OAEP)

- Basic problem: we don't know the format of the plaintext
- Desirable properties
  - Indistinguishability under Chosen Plaintext Attack (IND-CPA)
  - Indistinguishability under Chosen Ciphertext Attack (IND-CCA)
  - Indistinguishability under Adaptive Chosen Ciphertext Attack (IND-CCA2)

Telegram

(http://www.cryptofails.com/post/70546720222/telegrams-cryptanalysis-contest)



TLS

- Transport Layer Security, used to be called SSL (Secure Socket Layer)
- TLS happens in user space, somewhere between transport layer and application layer
- WEP and WPA{2,3}, were in link layer, below IP layer
  - Complement each other
- Non-repudiation *via* certificates
- Let’s look at an example in Wireshark…
- Wikipedia article should give you some sense of the long history of TLS being vetted

References

- [Cryptography Engineering] *Cryptography Engineering: Design Principles and Applications,* by Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. Wiley Publishing, 2010.
- [Cryptovirology] *Malicious Cryptography: Exposing Cryptovirology*, by Adam Young and Moti Yung. Wiley Publishing, 2004.
- Lots of images and info plagiarized from Wikipedia