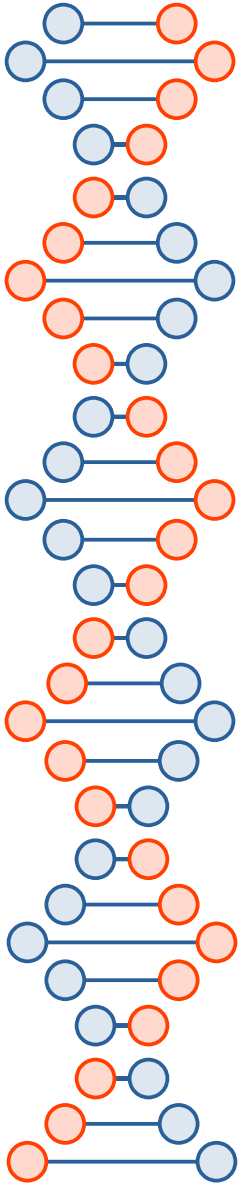
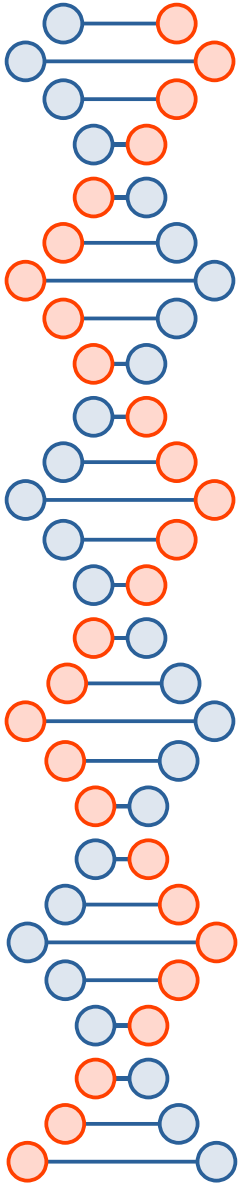


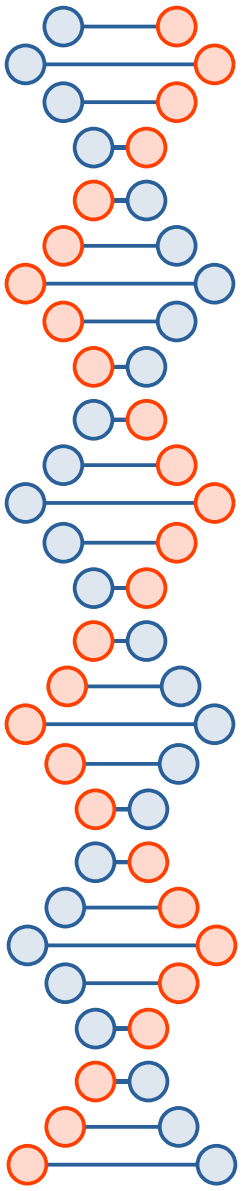
Optimal Asymmetric Encryption Padding (OAEP)

CSE 539
jedimaestro@asu.edu



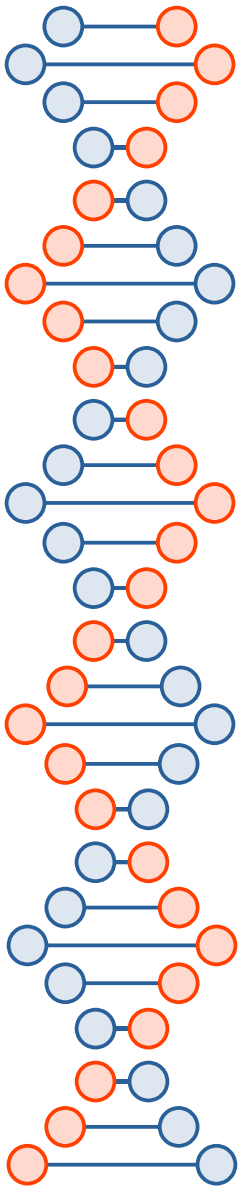
Warning: these slides are very informal, to try to understand the importance of OAEP without digging too deep into the theoretical computer science aspects of it…

(read the paper if you're---unlike me---inclined towards theory)

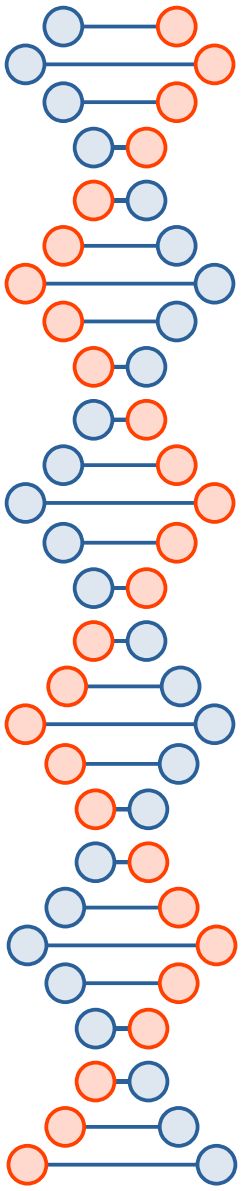


Review: PKCS#7 padding

- AES always encrypts in 128-bit blocks
  - 128 bits == 16 bytes
- If you fill up blocks, that's great
  - The last block might not be full
- Need an “unambiguous” way to pad the last block so the decrypting party knows the padding to throw out
  - *E.g.*, PKCS#7 (PKCS == Public Key Cryptography Standards)

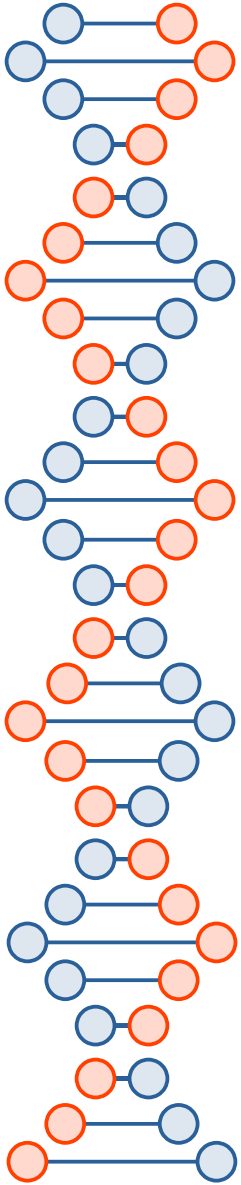


															01
														02	02
													03	03	03
												04	04	04	04
											05	05	05	05	05
										06	06	06	06	06	06
									07	07	07	07	07	07	07
								08	08	08	08	08	08	08	08
							09	09	09	09	09	09	09	09	09
						0A	0A	0A	0A	0A	0A	0A	0A	0A	0A
					0B	0B	0B	0B	0B	0B	0B	0B	0B	0B	0B
				0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C
			0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D
		0E	0E	0E	0E	0E	0E	0E	0E	0E	0E	0E	0E	0E	0E
	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10



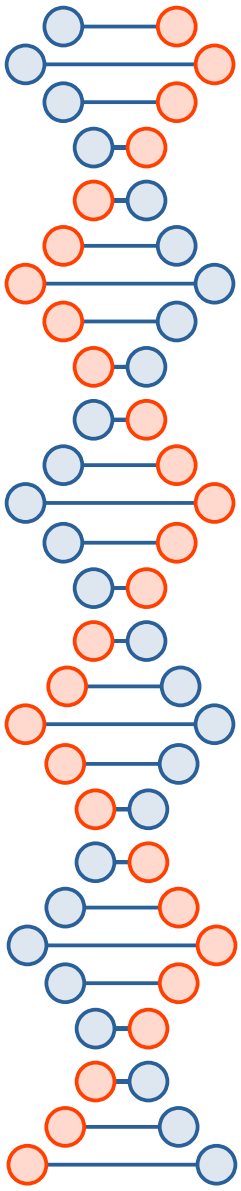
Standard model (cryptography)

- https://en.wikipedia.org/wiki/Standard_model_(cryptography)
- Adversary is only limited by available time and computational power
- Schemes must be proven only using complexity assumptions
  - Difficult to do
  - Hence, cryptographic primitives are replaced by idealized versions



Random oracle model

- Basic idea: replace hash functions with idealized hash functions
- Bellare and Rogaway argued for their practicality in 1994
- Oracle (theoretical black box) “that responds to every unique query with a (truly) random response chosen uniformly from its output domain”
- “If a query is repeated, it responds the same way every time that query is submitted.”
- https://en.wikipedia.org/wiki/Random_oracle



Random Oracles are Practical: A Paradigm for Designing Efficient Protocols

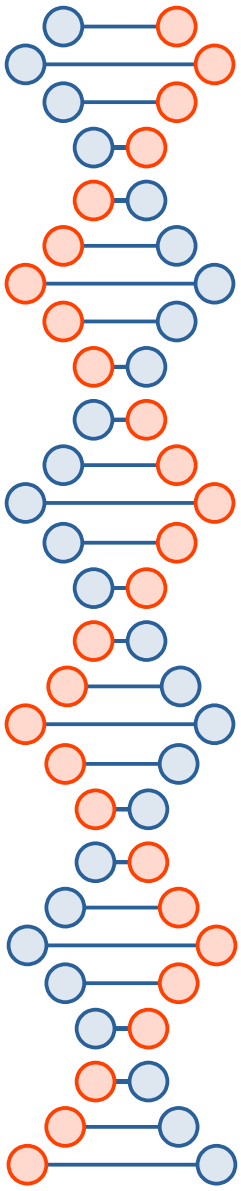
MIHIR BELLARE* PHILLIP ROGAWAY†

Abstract

We argue that the random oracle model —where all parties have access to a public random oracle— provides a bridge between cryptographic theory and cryptographic practice. In the paradigm we suggest, a practical protocol P is produced by first devising and proving correct a protocol P^R for the random oracle model, and then replacing oracle accesses by the computation of an "appropriately chosen" function h. This paradigm yields protocols much more efficient than standard ones while retaining many of the advantages of provable security. We illustrate these gains for problems including encryption, signatures, and zero-knowledge proofs.

https://dl.acm.org/doi/pdf/10.1145/168588.168596

1st Conf.- Computer & Comm. Security '93-11/93 -VA,USA



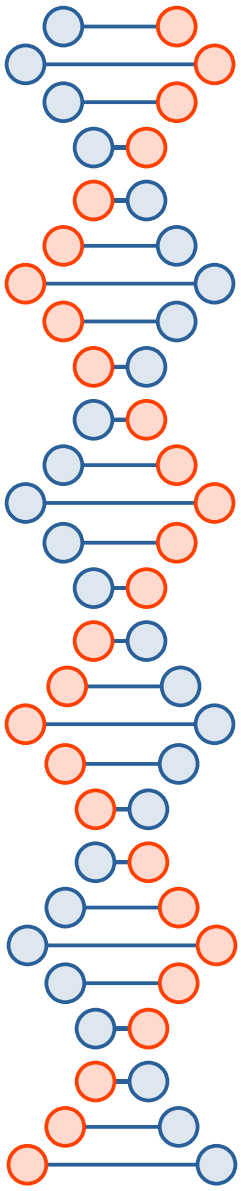
A preliminary version of this paper appeared in *Advances in Cryptology – Eurocrypt* 94 *Proceedings*, Lecture Notes in Computer Science Vol. 950, A. De Santis ed., Springer-Verlag, 1994.

Optimal Asymmetric Encryption — How to Encrypt with RSA

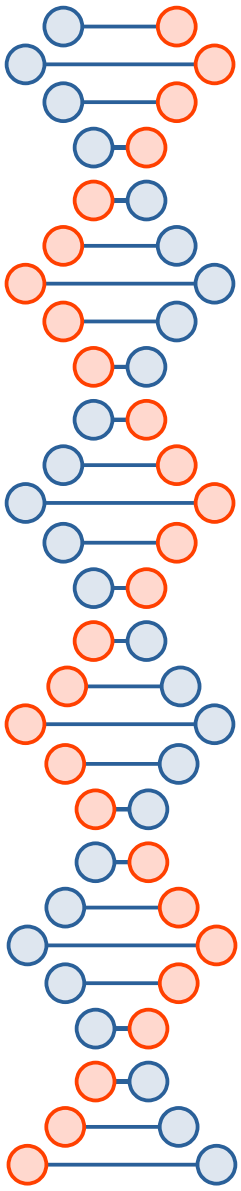
Mihir Bellare* Phillip Rogaway†

November 19, 1995

https://cseweb.ucsd.edu//~mihir/papers/oaep.pdf



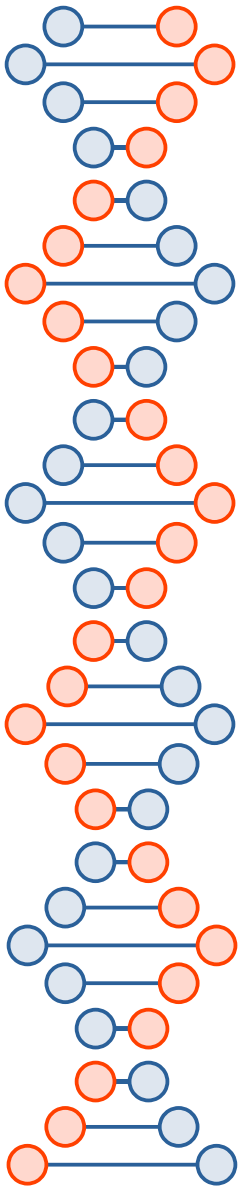

0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
1011000010010110011101111011101100100010111111001110101011110011
0000000011100100101111001001010100100011101101010000101110111011



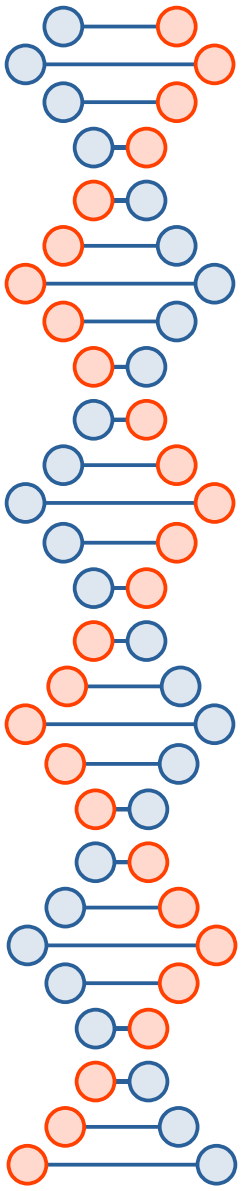
$$C_b \equiv {k_b}^e \pmod{n}$$

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

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

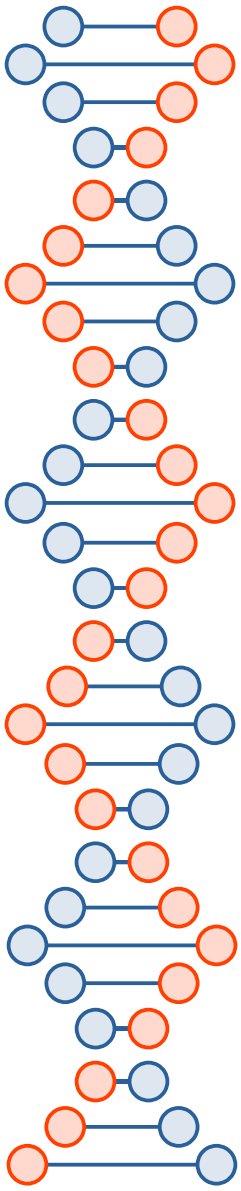


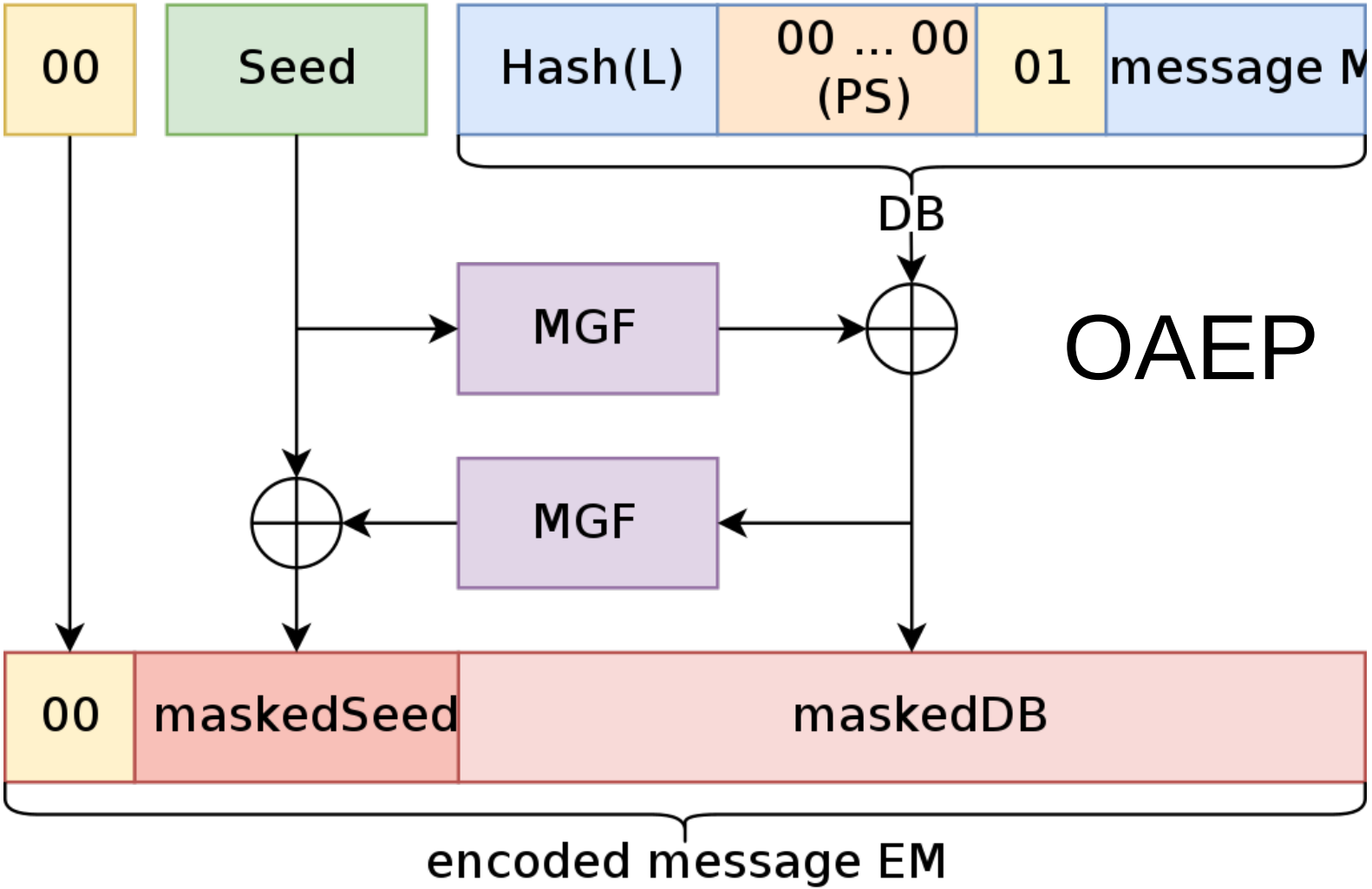

0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0101100001001011001110111101110111001000101111110011101010111001
1000000001110010010111100100101010010001110110101000010111011101
1000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000



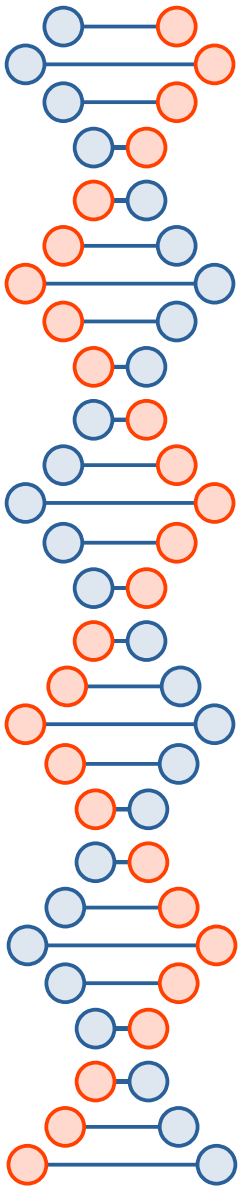
https://en.wikipedia.org/wiki/Daniel_Bleichenbacher

- Bleichenbacher-style attack published in 1998
  - Chosen ciphertext attack
  - Padding oracle attack
- 0x00 0x02 [non-zero bytes] 0x00 [M]
  - 2^{-17} to 2^{-15} probability a random ciphertext has this format when decrypted with RSA
  - https://crypto.stackexchange.com/questions/12688/can-you-explain-bleichenbachers-cca-attack-on-pkcs1-v1-5
  - Takes a few million connections



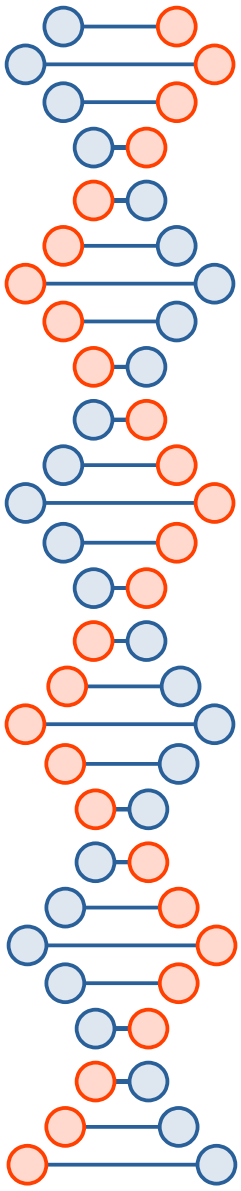


https://en.wikipedia.org/wiki/File:OAEP_encoding_schema.svg



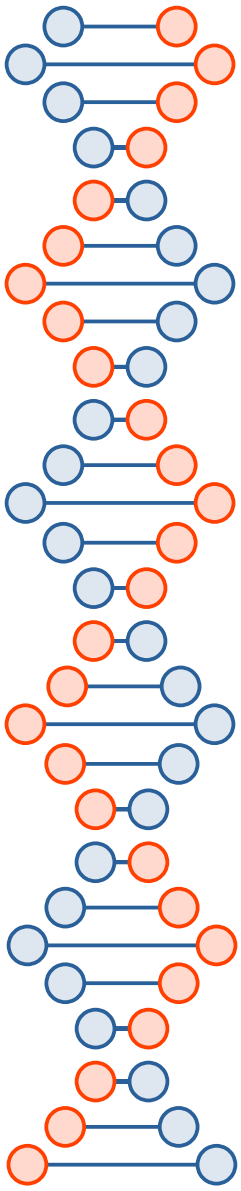
Good enough?

- *Proven* to be IND-CCA2 secure (with some assumptions)
  - Reduction proof
- What this means in practice:
  - If I'm using RSA-OAEP and you perform an adaptive chosen ciphertext attack against my scheme, give me the source code for your attack and I'll use it to factor large integers



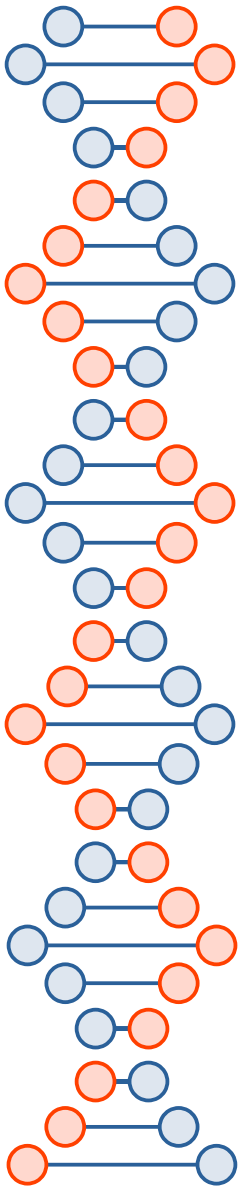
What is IND-CCA2?

- In the olden days, in the beforetime, in the long long ago…
  - Ciphertext only (Viginere cipher cracking), known plaintext (Enigma), chosen plaintext (differential cryptanalysis)
- Now threat models are very complicated, but in a nutshell:
  - IND-CPA – Indistinguishability under chosen plaintext attack
  - IND-CCA – Indistinguishability under chosen ciphertext attack
  - IND-CCA2 – Indistinguishability under chose ciphertext attack (adaptive)



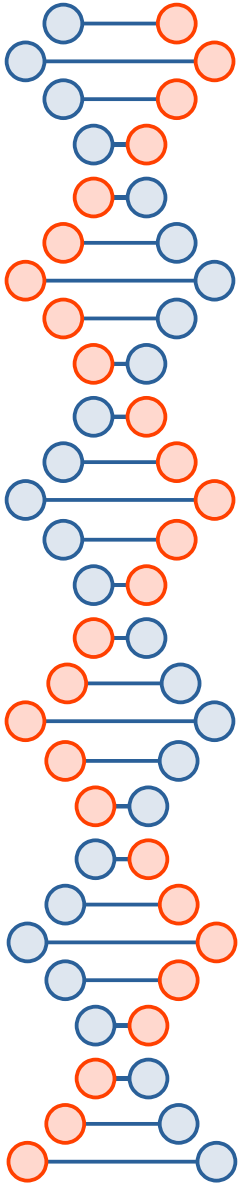
IND-CCA2 in a nutshell

- I'll encrypt or decrypt as many plaintexts or ciphertexts as you like
  - plaintext/ciphertext pairs
- You give me two plaintexts, I'll flip a coin (heads or tails) and encrypt one of them (you don't know which) to give you C
- In polynomial time, you can do more encryption and decryption, just not for C
- You guess my coin flip (heads or tails)



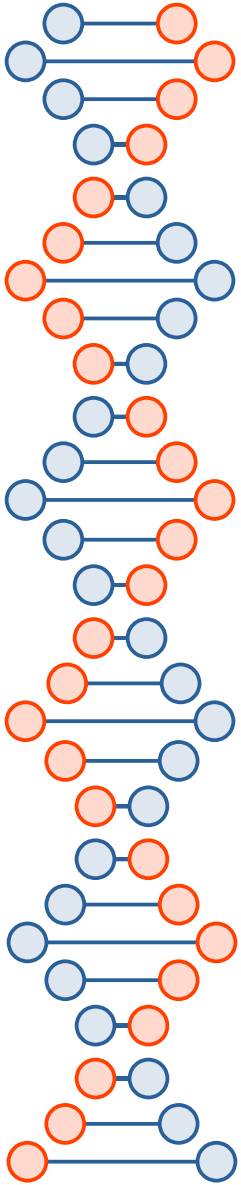
If you can’t win with >50% probability

- You can’t break my scheme (*e.g.*, OAEP) with an adaptive chosen ciphertext attack



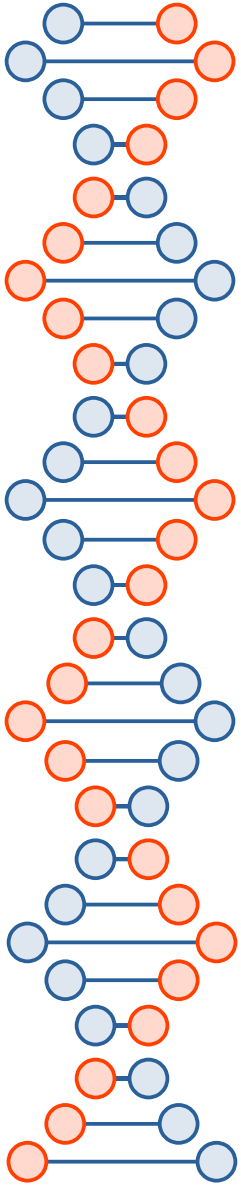
If you **can** win with >50% probability

- You've potentially broken my scheme with an adaptive chosen ciphertext attack
- Let's win the Turing award together, by publishing a paper showing how to factor large integers with a classical computer in polynomial time
  - Or, build a cybercrime business together?

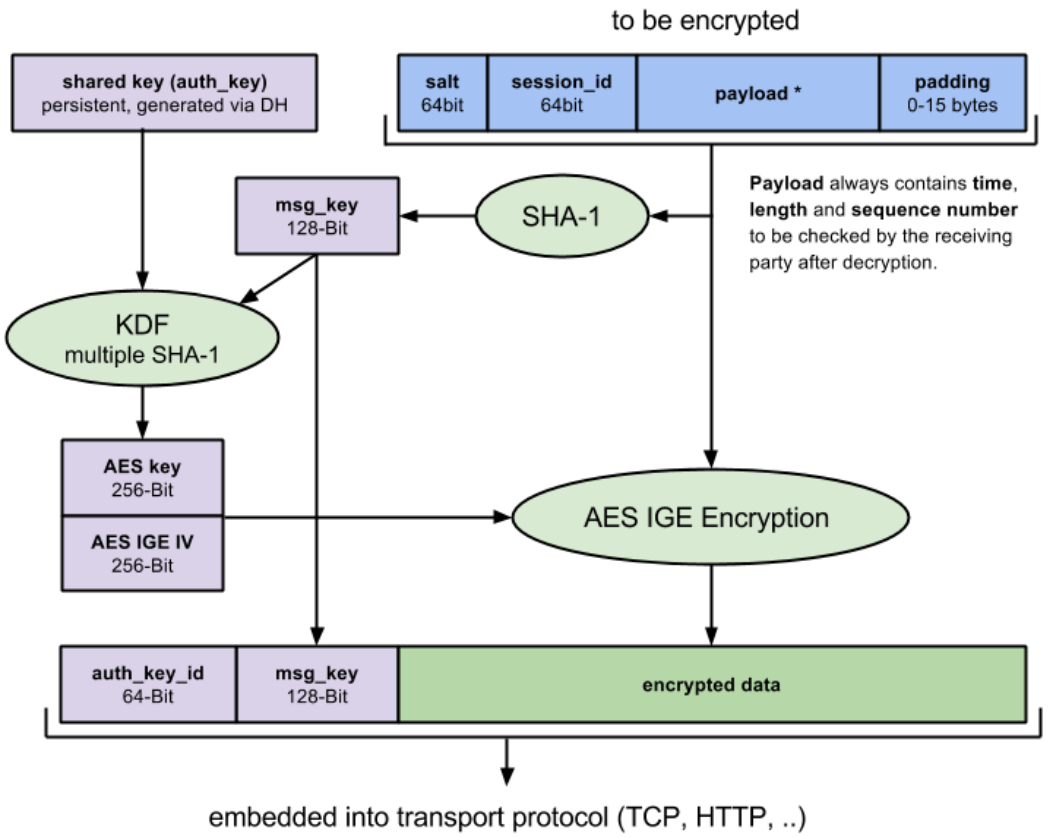


Telegram

a new era of messaging

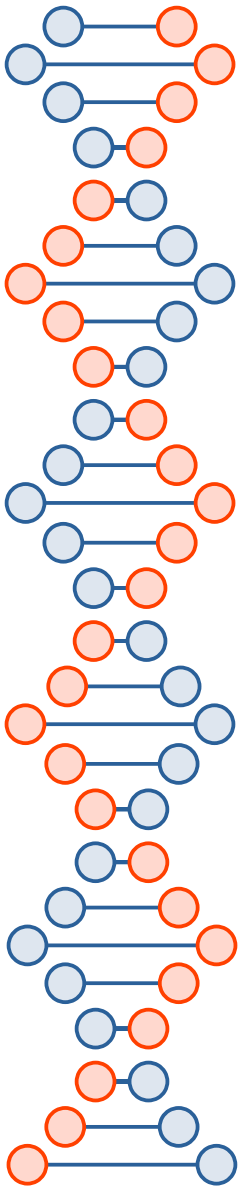


MTProto encryption



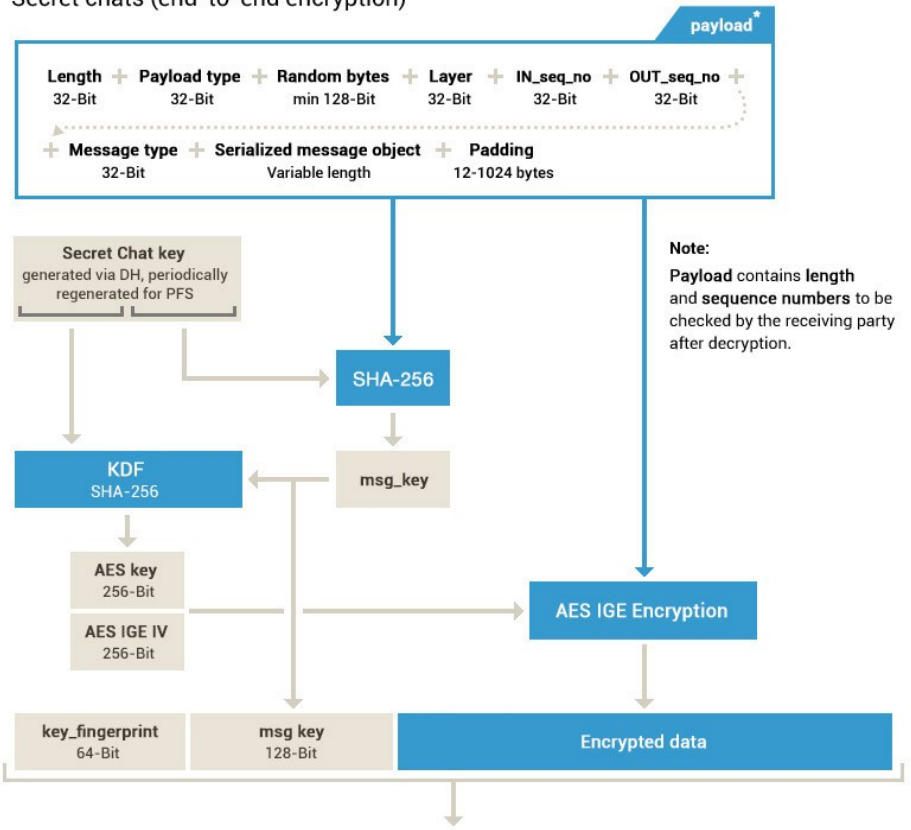
NB: After decryption, **msg_key** MUST be equal to SHA-1 of data thus obtained.

https://core.telegram.org/img/mtproto_encryption.png



MTProto 2.0, part II

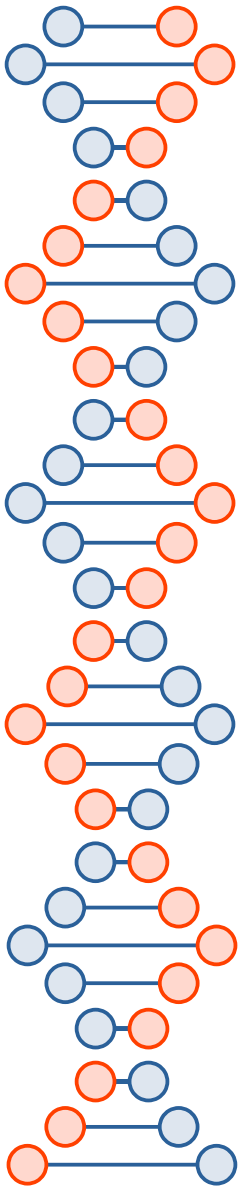
Secret chats (end-to-end encryption)



embedded into an outer layer of client-server (cloud) MTProto encryption, then into the transport protocol (TCP, HTTP, ..)

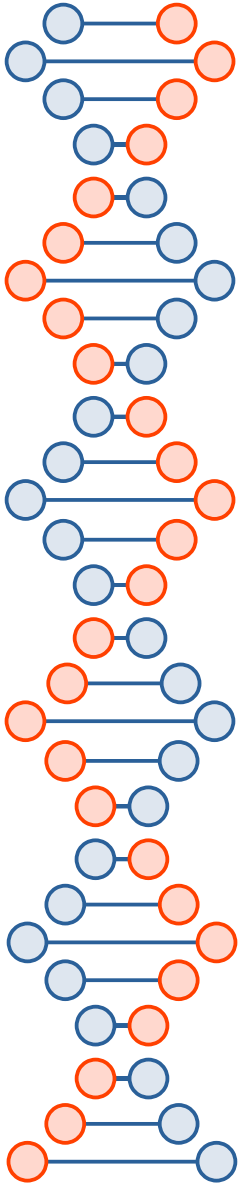
Important: After decryption, the receiver **must** check that msg_key = SHA-256(fragment of the secret chat key + decrypted data)

https://core.telegram.org/api/end-to-end



Some “attacks” and criticisms...

- https://caislab.kaist.ac.kr/publication/paper_files/2017/SCIS17_JU.pdf
- https://unhandledexpression.com/crypto/general/security/2013/12/17/telegram-stand-back-we-know-maths.html
- https://enos.itcollege.ee/~edmund/materials/Telegram/A-practical-cryptanalysis-of-the-Telegram-messaging-protocol_master-thesis.pdf
- https://caislab.kaist.ac.kr/publication/paper_files/2017/SCIS17_JU.pdf
- https://mtpsym.github.io/
- https://iacr.org/submit/files/slides/2022/rwc/rwc2022/60/slides.pdf



Takeaways

- Padding is important
- You don’t always have to settle for “we tried to break it really hard for a long time and couldn’t”