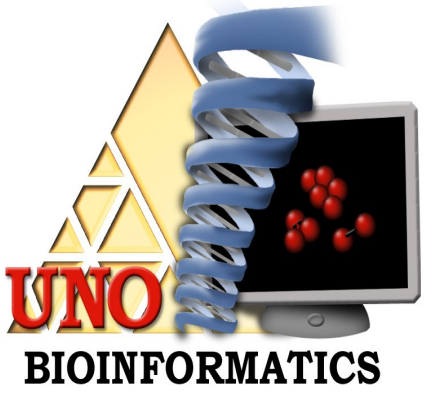


sEncrypt: An Encryption Algorithm Inspired From Biological Processes

Oliver Bonham-Carter1, Abhishek Parakh1,2, Dhundy Bastola1
1School of Interdisciplinary Informatics
2 Nebraska University Center for Information Assurance
University of Nebraska at Omaha Omaha, NE, 68182, USA

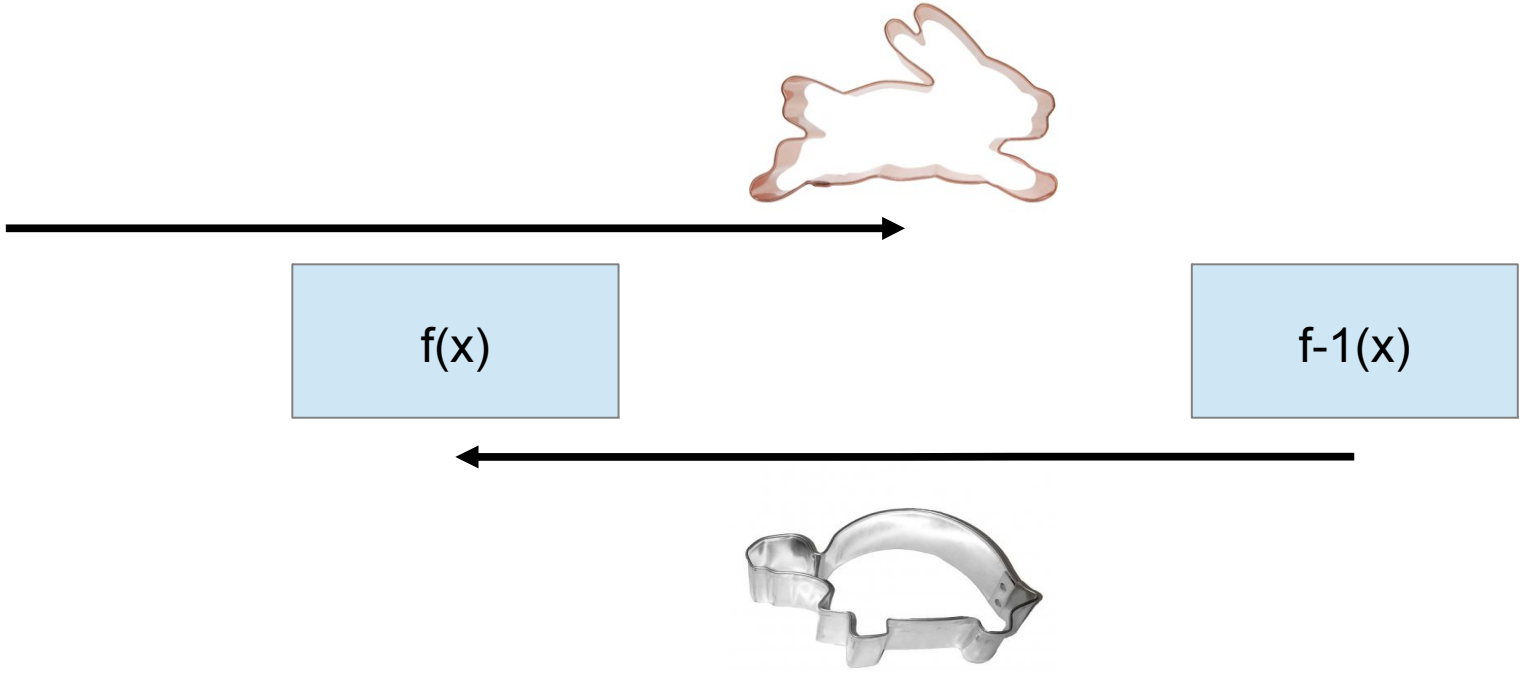
Email: {obonhamcarter, aparakh, dkbastola}@unomaha.edu

Overview of Presentation

- We present an encryption and decryption method:
  - Modeled after biological methods
  - Uses public bioinformatics sequence data for keys
  - Satisfies C. Shannon's *Confusion and Diffusion* properties

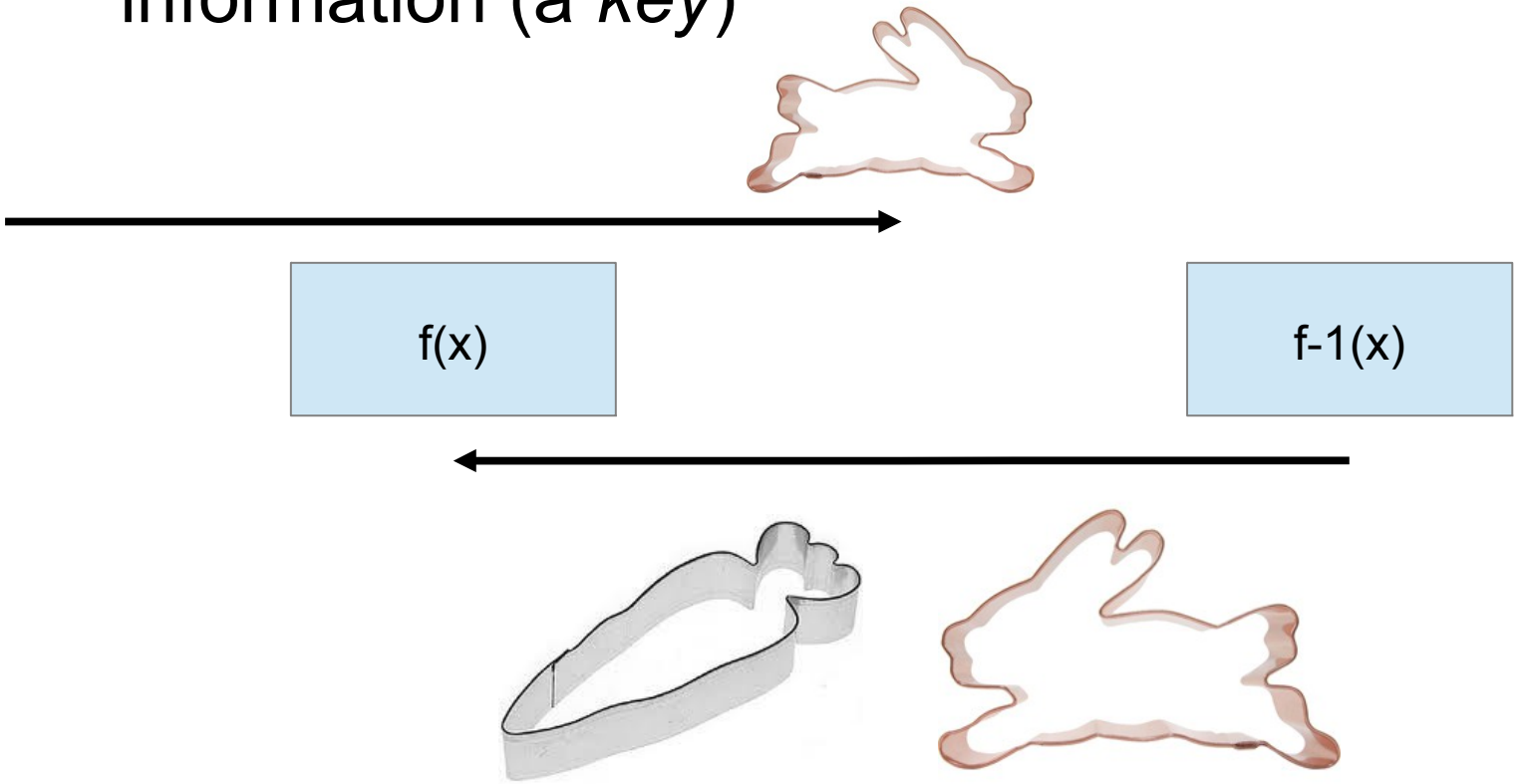
Trap-door Functions:
A Method Behind Encryption

- An “easy” calculation made in one direction
  - can be done in feasible time.
- A “hard” calculation to inverse
  - cannot be done in feasible time.



Trap-door Functions:
A Method Behind Encryption

- The inverse is actually found by the aid of extra information (a *key*)



Example: Traditional Trapdoor Function Using Primes

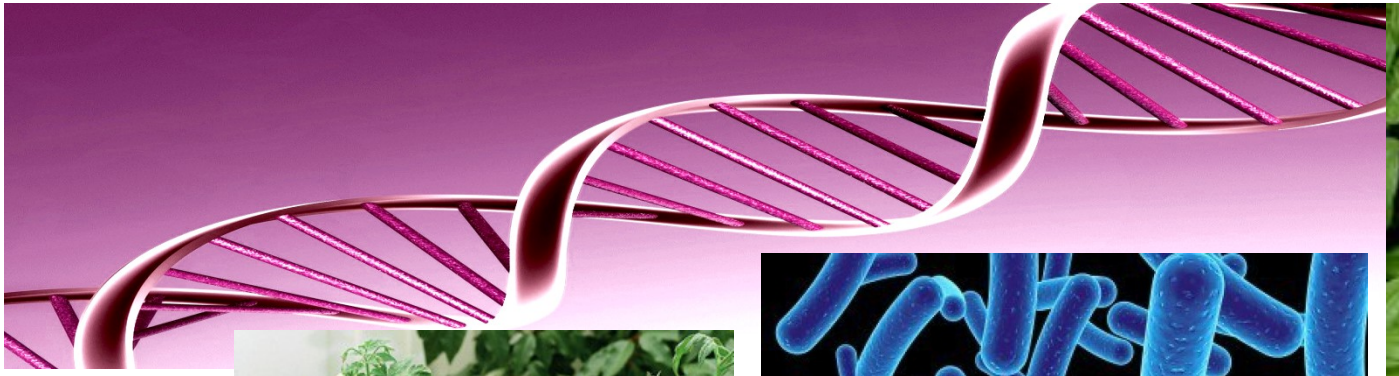
The multiplication of primes is “Easy”

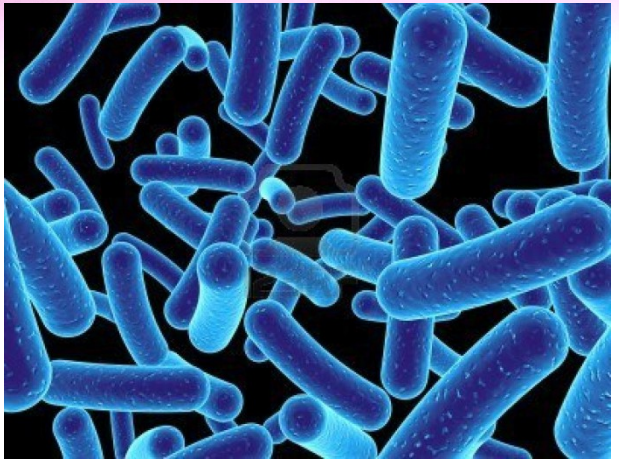
- 7919 * 1709 = 13533571

The factorization of composite numbers is “Hard”

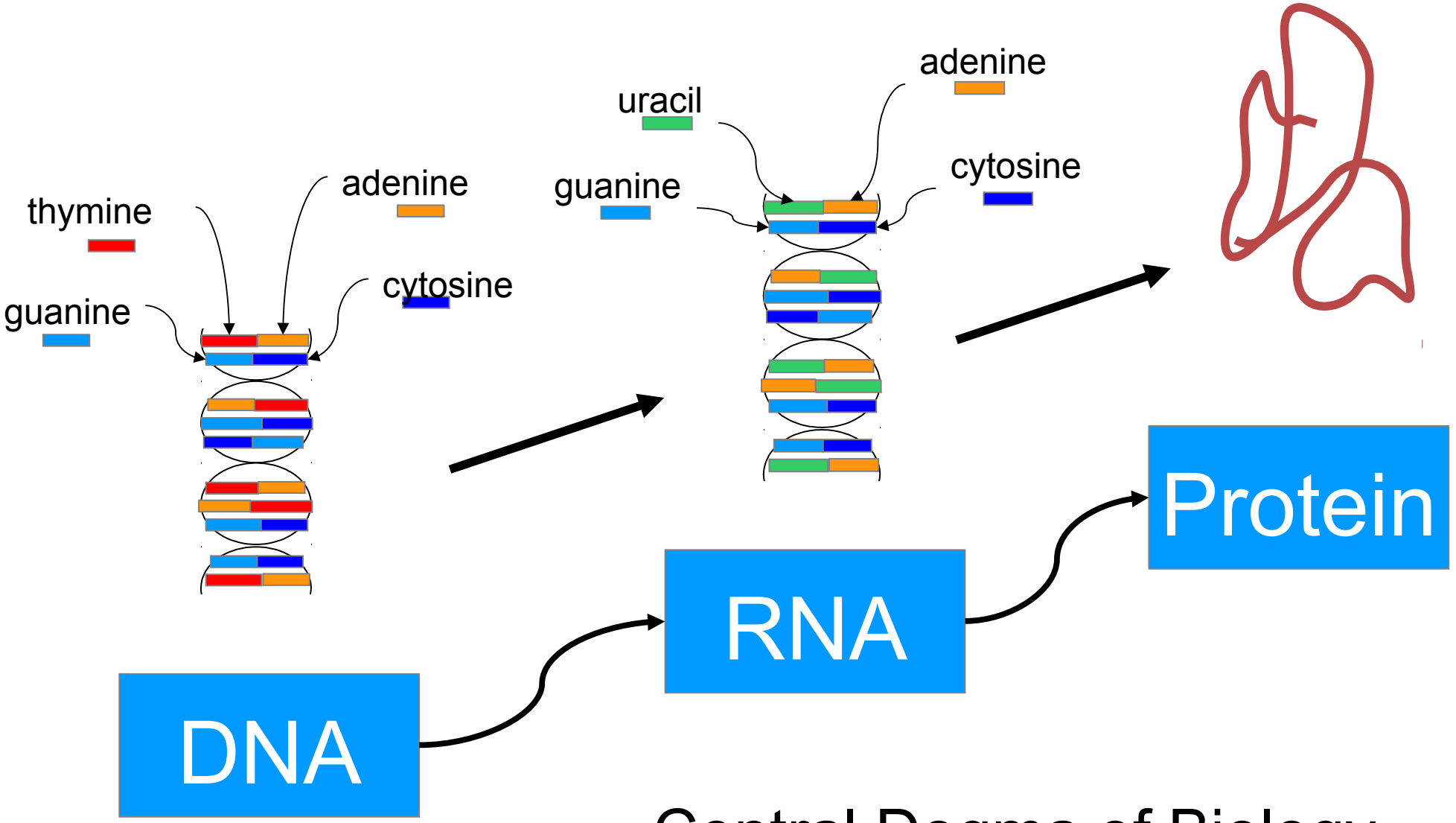
- 13533571 = 7919 * 1709

Trapdoor functions can be modeled from natural mechanisms in biology.



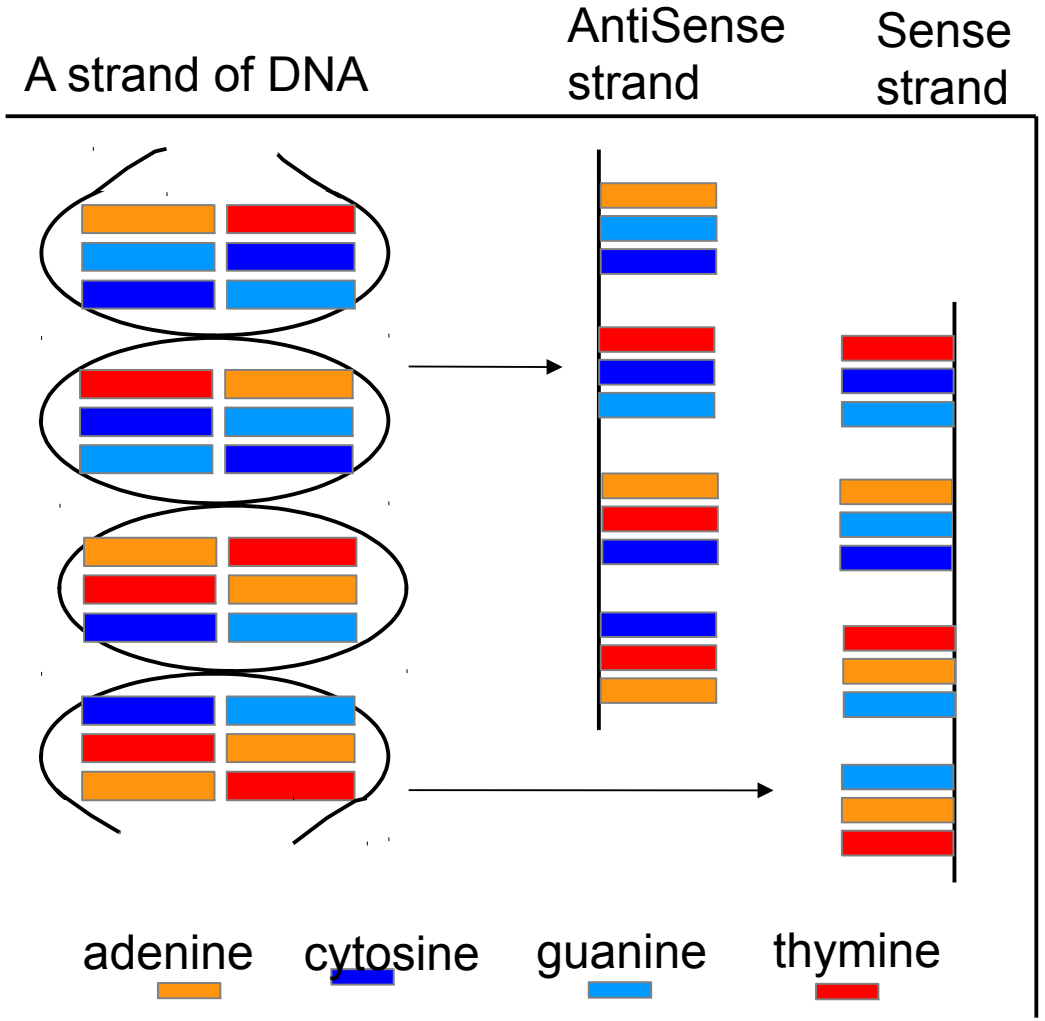


Functions from Biology



Central Dogma of Biology

Protein Translation



AntiSence: TCG AGC TAG GAT

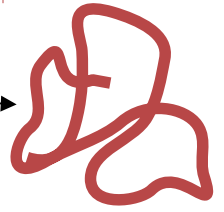
Sense: AGC TCG ATC CTA

Codons (triplets) come from RNA

RNA: AGC UCG AUC CUA

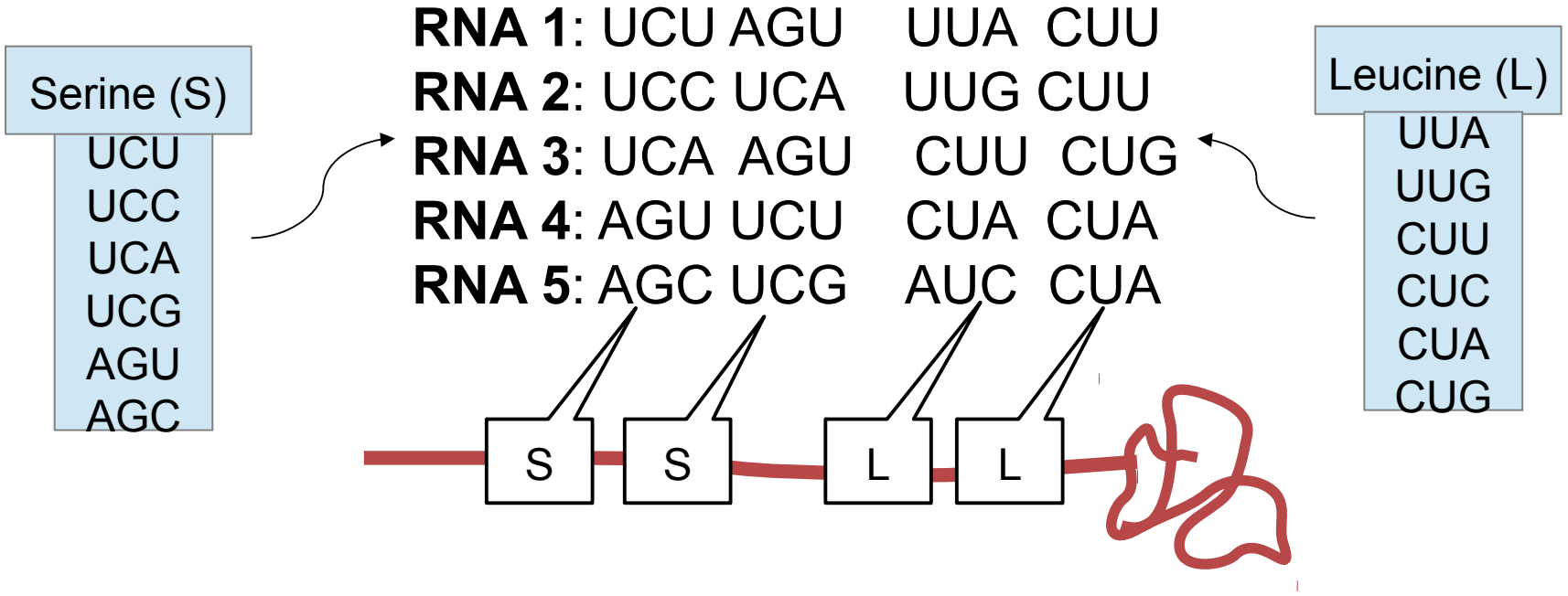
S S L L

Protein is made from **amino acid** building Blocks.



Note: Serine (S) is coincidentally coded by AGC and UCG and Leucine (L) by AUC and CUA.

It is trivial to convert DNA to protein code but hard to convert protein to DNA code.



Note: For a sequence of length *n*, there are 6^*n* possible codon combinations. As the sequence gets longer, the number of combinations diverges.

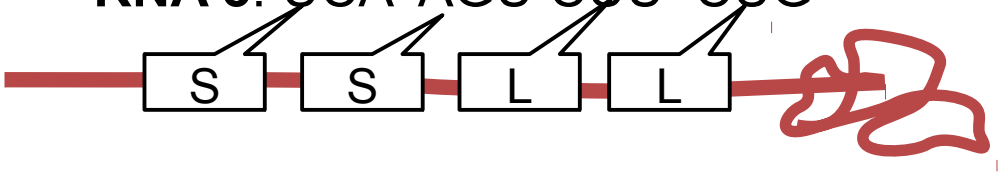
Codon Bias:
The selection of specific triplets for amino acids in protein translation

- Bacterial organisms have unique codon biases.
- Three examples of sequences generating the same protein by means of different RNA (and DNA) sequences.

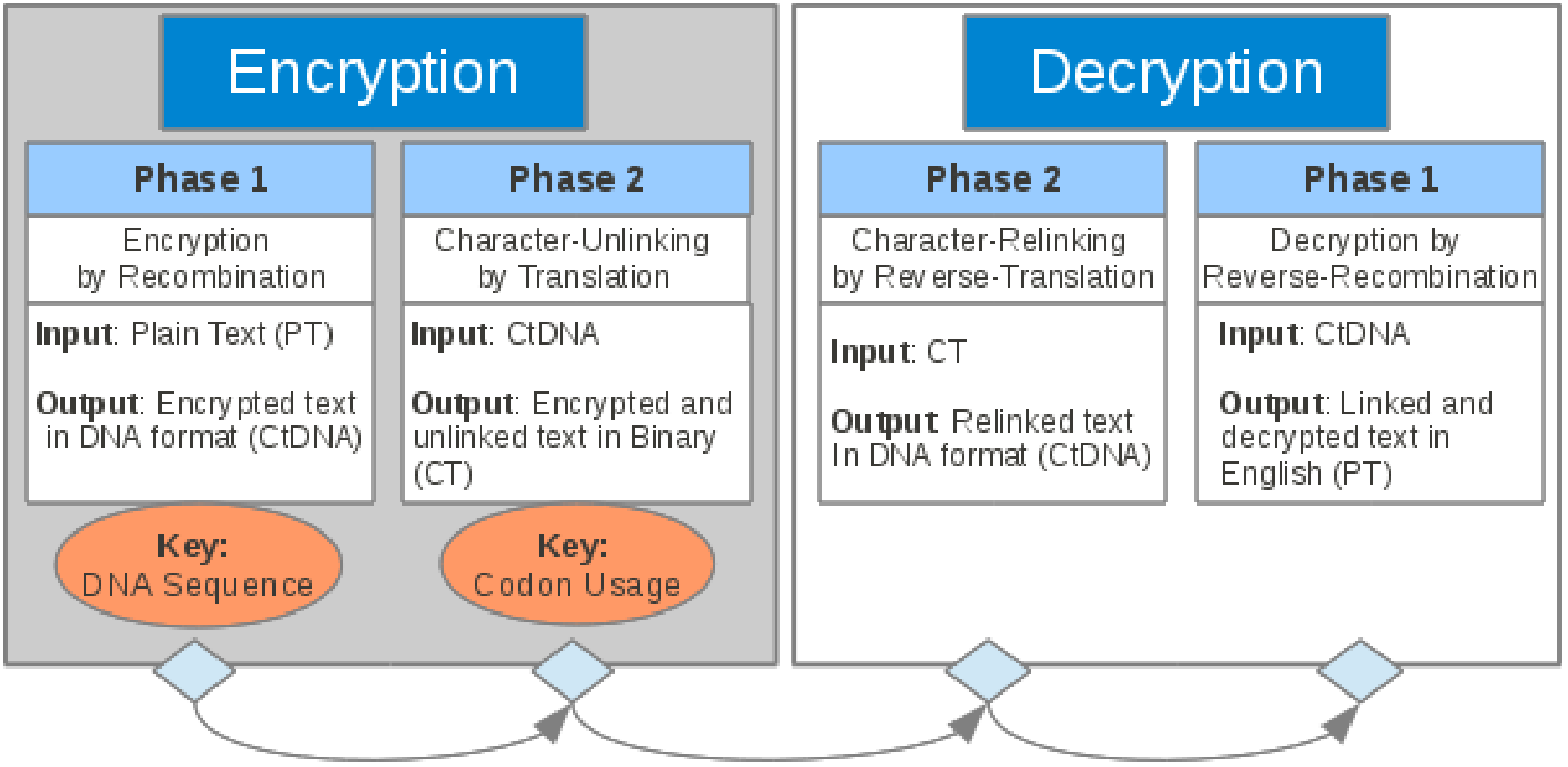
Organism 1 **RNA 1**: UCU AGU UUA CUU

Organism 2 **RNA 2**: UCC UCA UUG CUU

Organism 3 **RNA 3**: UCA AGU CUU CUG



Flowchart:
Encryption and Decryption



Two ke

- The **first key** is a strand of arbitrarily selected DNA.
- The organismal sequence (start and stop positions) must be known.
  - Sequence taken from the National Center for Biotechnology Information (NCBI) public database
  - ref: http://www.ncbi.nlm.nih.gov/

- The **second key** is a codon bias table (64 codon entries) for a biological organism.
- Taken from Codon Usage Database:
  - ref: http://www.kazusa.or.jp/codon/

Nakamura, Yasukazu, Takashi Gojobori, and Toshimichi Ikemura. "Codon usage tabulated from international DNA sequence databases: status for the year 2000." *Nucleic acids research* 28. (2000): 292-292.

First Key: Sequence Data

- Escherichia coli 536, (NC_008253)
- NCBI (http://www.ncbi.nlm.nih.gov/)
- Position x to position y (for example)

Seq:

agcttttcattctgactgcaacgggcaatatgtctctgtgtgg attaaaaaaagagtgtctgatagcagctt . . . aaatat caccaaataaaaaacgccttagtaagtgattttc

For example: we take only the sequence data in red from this organism.

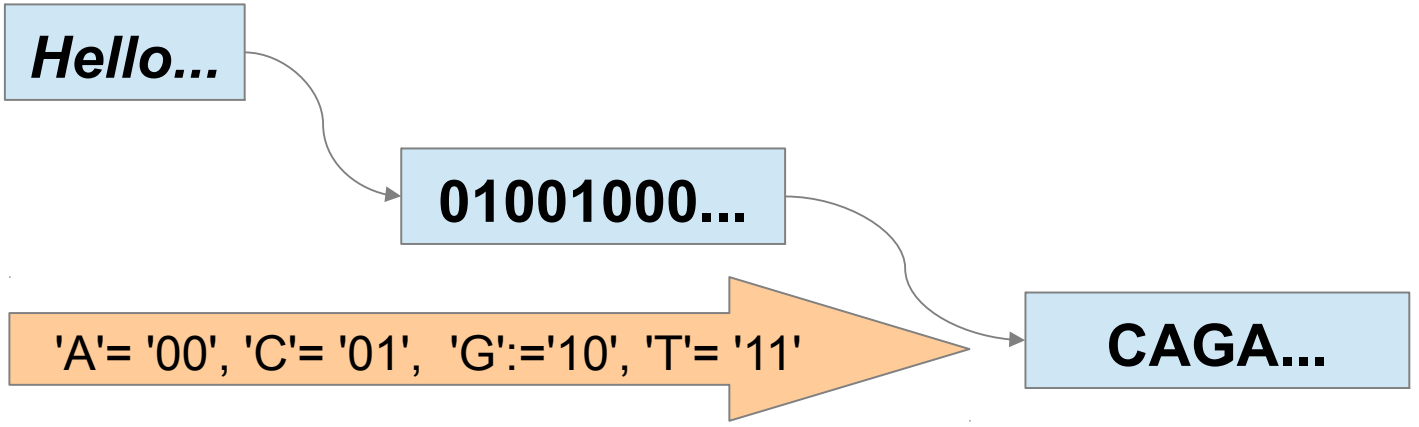
Second Key: Codon Bias Table

- Organism: Escherichia coli 536, (NC_008253)
- Codon Usage Database, http://www.kazusa.or.jp/codon/

UUU 55.0	UCU 27.5	UAU 18.3	UGU 9.2
UUC 9.2	UCC 0.0	UAC 9.2	UGC 0.0
UUA 36.7	UCA 9.2	UAA9.2	UGA 0.0
UUG 0.0	UCG 9.2	UAG0.0	UGG 9.2
CUU 27.5	CCU 9.2	CAU 18.3	CGU 9.2
CUC 0.0	CCC 0.0	CAC 0.0	CGC 18.3
CUA 18.3	CCA 27.5	CAA 18.3	CGA 0.0
CUG 0.0	CCG 0.0	CAG 0.0	CGG 0.0
AUU 27.5	ACU 9.2	AAU 91.7	AGU 9.2
AUC 0.0	ACC 0.0	AAC 0.0	AGC 0.0
AUA 27.5	ACA 18.3	AAA 64.2	AGA 64.2
AUG 9.2	ACG 0.0	AAG 45.9	AGG 9.2
GUU 73.4	GCU 18.3	GAU 36.7	GGU 36.7
GUC 0.0	GCC 0.0	GAC 0.0	GGC 9.2
GUA 9.2	GCA 0.0	GAA 27.5	GGA 9.2
GUG 9.2	GCG 9.2	GAG 36.7	GGG 0.0

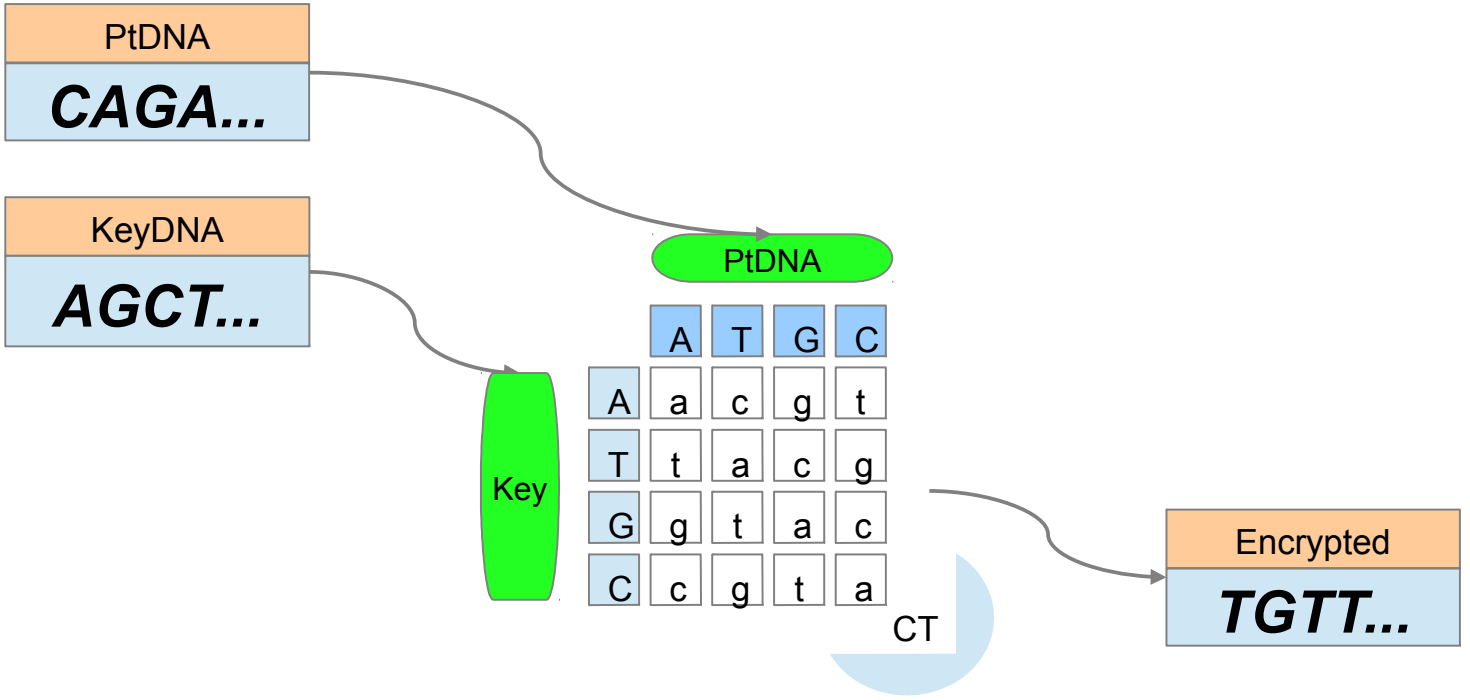
Encryption

- Begin by binary encoding the plain text (PT).
- Then assign DNA Bases to binary double to create encoded plain text DNA (PtDNA).



Apply the Latin Square

- A Latin square is an *n* by *n* matrix filled with *n* unique symbols. No symbol occurs more than once in any row or column.

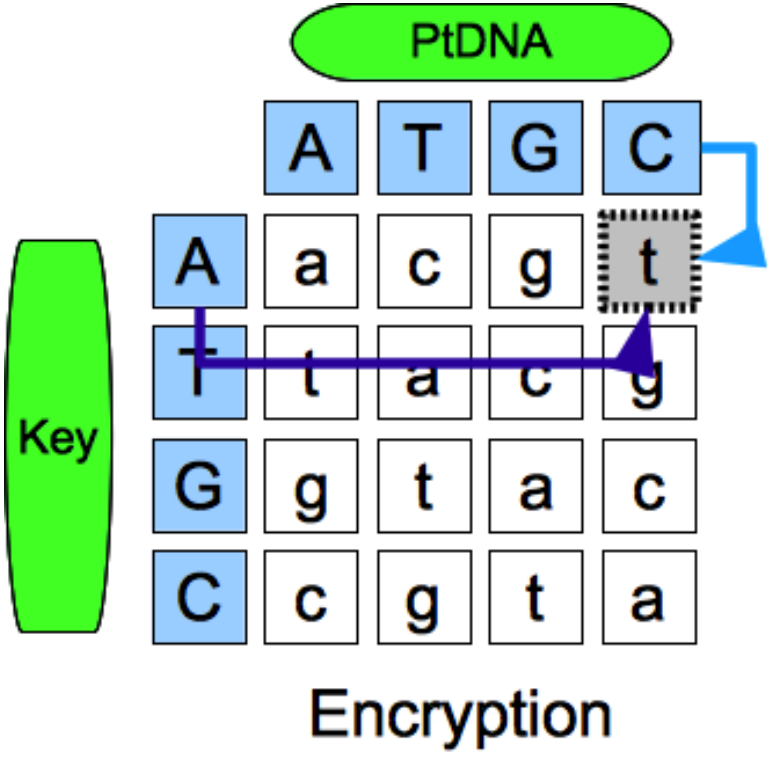


Note: Any Latin square will work as long as it meets above requirement.

Encryption

Close-up of Latin Square

- The intersection of A (Key) and C (PlainText) gives 't'.
- Repeat for all PtDNA.
- Note: the Key sequence should be the same length as PtDNA



Encryption

Encryption

Apply Huffman encoding to CtDNA

- Codon table: frequencies are used for Huffman encoding of all corresponding codons by amino acid (protein building blocks).
- It is necessary to specify the amino acid to know which particular codon is being addressed.
- Codon codes between different amino acids are not unique.

Proline (P)
Code: 00011

Codon	Code
CCA	**0**
CCC	11
CCG	110
CCT	10

Alanine (A)
Code: 10101

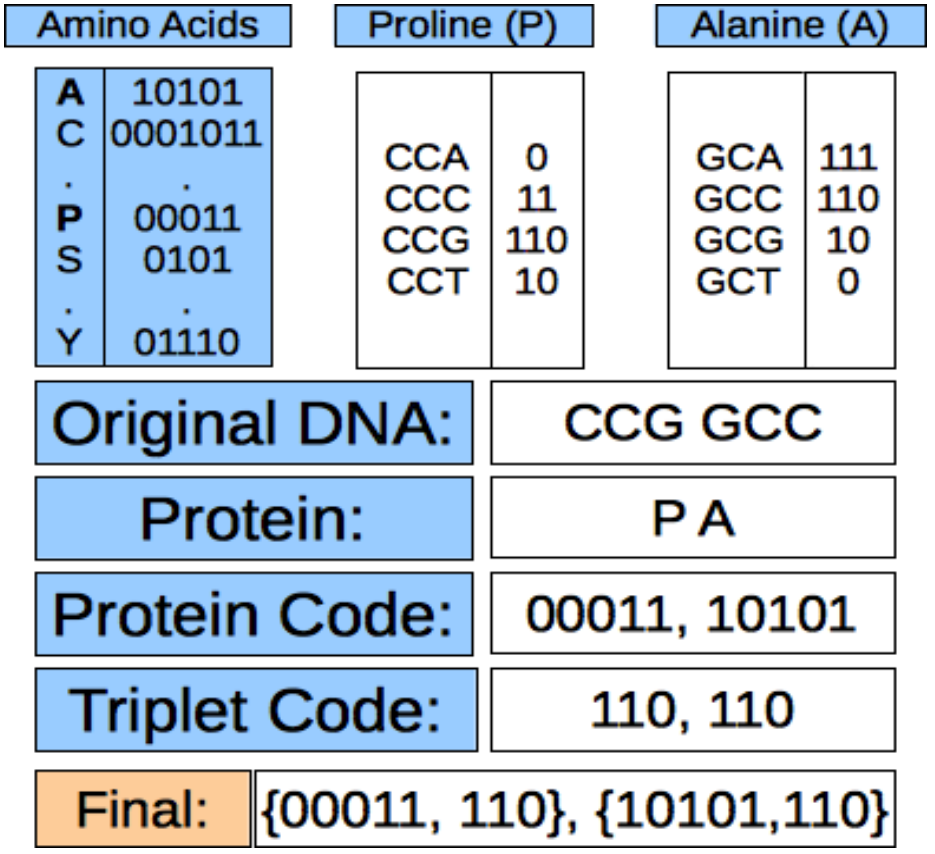
Codon	Code
GCA	111
GCC	110
GCG	10
GCT	**0**

Note: Codons, CCA for Proline (P), and GCT for Alanine (A), have the same coding. Codes must be accessed by their amino acids which are Huffman encoded from the summed frequencies of their codon sets.

Product of Encryption (CT)

- The output is two Huffman coding sequences for the cipher text (CT): one sequence for the amino acid and the other for its associated codon.
- Without knowing the amino acid, the codon code cannot be known (inter-amino acid redundancy).
- One line contains amino acid information and the other for codons:

Transmit: "*0001110101*\n, *110110*"



Decryption:
Begin by Re-reading Ciphertext (CT)

- The CT is read as follows:
  - Read first Huffman code of protein sequence.
  - Once a protein code is recognized, read the codon code to recover the exact codon.
  - Record codon triplet and repeat until finished.

Decryption: Example

CT: 0001110101, 110110

Start **00011**10101, **110**110

Amino Acid: Proline → Proline (P) Code: **00011**

Codon	*Code*
CCA	0
CCC	11
CCT	10
CCG	110

Cont. 00011**10101**, 110**110**

Amino Acid: Alanine → Alanine (A) Code: **10101**

Codon	*Code*
GCA	111
GCT	0
GCG	10
GCC	110

Record cipher Text for Latin square (CtDNA):

1 **CCG**

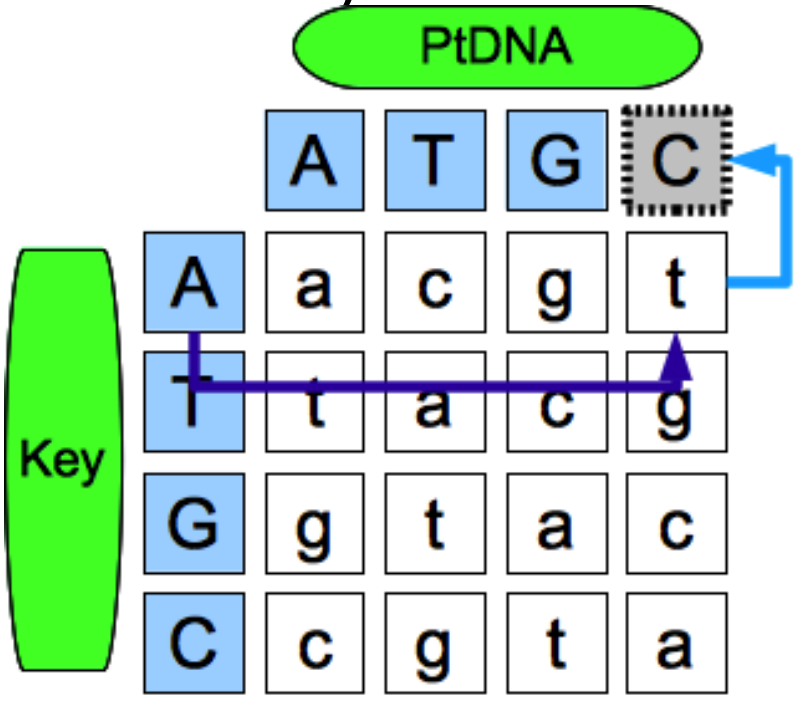
2 CCG **GCC**

Decryption

Latin Square: Decryption

- Reapply the same KeyDNA as in the encryption process to create PtDNA.

- The intersection of ‘A’ (Key) and 't' (in column), gives 'C' (at top of PtDNA column).

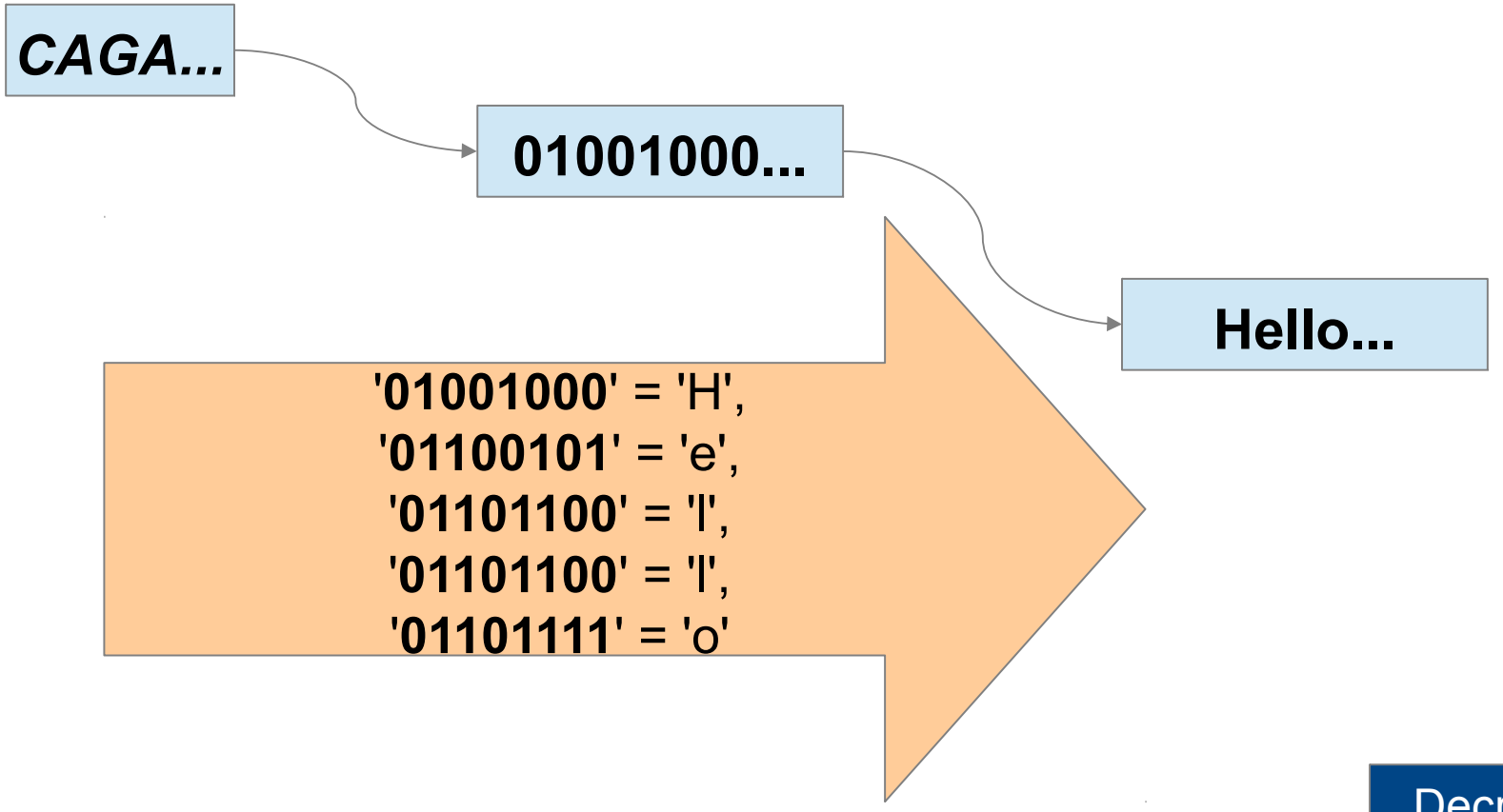


Decryption

Decryption

Convert the CtDNA back to PT.

CtDNA to binary to PtDNA



Decryption

Summary of Steps: Encryption and Decryption

Phase	Stage	Description	Sequence
One	PT	Plain text written in a language (here, English)	A, B, C
	PtDNA	Binary PT encoded into DNA	CAA CCA AGC AAT
	keyDNA	The sequence of DNA which is used by the Latin square to encrypt PtDNA	AGC TTT TCA TTC
	CtDNA	Cipher text in DNA form having completed the Latin square of phase one	TGC GGT TTT TTG
Two	CtProtein	Amino acids, A translated version of CtDNA	['C', 'G', 'F', 'L']
	CtFinal	Text version of the encoded amino acids and triplets	[('0001011','1'), ...,]
	CT	CtFinal in binary format containing the protein encoding followed by the corresponding triplet codes.	[00010110010..., 011011110101...]

The details of the analysis (entropy and size reduction for transmission) are found in our paper, "**sEncrypt: An Encryption Algorithm Inspired From Biological Processes**."

Encryption

Decryption

Conclusions

- We introduce a novel encryption and decryption method which is based on the central dogma of biology.
- The keys of this method are generated by public bioinformatics sequence data of which there is a seemingly infinite amount available.
- On encryption, plain text is applied to binary coding, a Latin square and Huffman encoding system to remove letter associations.

Conclusions

- On decryption, the Huffman encoding, Latin square and binary coding work is undone to return to plain text.

- Future works:
  - We intend to analyze the robustness of the method, determine algorithmic complexities and to demonstrate its applications to insecure channel communication.

Acknowledgments

- We would like to thank:
  - The support staff in the UNO-Bioinformatics Core Facility (Omaha, NE, USA), funded by the grants from the National Center for Research Resources (5P20RR016469) and the National Institute for General Medical Science (NIGMS) (8P20GM103427).

Please address any questions to:

Oliver Bonham-Carter

Email: obonhamcarter@unomaha.edu

Thank you for your attention.