Creating a Block Cipher Protocol Using Steganographic Methods

Anas, Manan, Michael, Takamitsu, Samyak, Ivan

Agenda

- Project Overview
- Technical Details
- Results
- Demo
- Moving Forward
- Q&A

Project Overview

Initial Inspiration

- Despite modern developments in cryptography and related fields, there are many cases where security can’t be achieved by just encryption:
  - Censorship
  - Eavesdropping using unknown backdoors
  - Government tapping

- Researched ways to circumvent situations where cryptography might not be the best tool of defense

- Realized potential of steganography as a method to transmit sensitive data in disguise of seemingly useless data

Our Project

Goal: create a novel messaging protocol using steganography to disguise ciphertext as unencrypted random texts

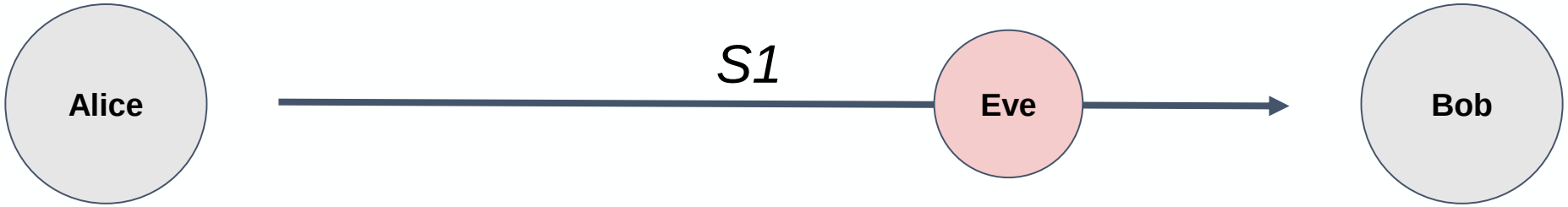
- Python package, easy to run through command line

- Components
  - Client and server set up for sending and receiving IP packets
  - Encryption and decryption functions to encrypt data
  - Masking functions to convert ciphertext to steganotext
  - Masking functions to extract ciphertext from steganotexts

Use Case

Scenario:

- Alice wants to send Bob a government secret
- Alice is worried that Eve may be a middleman in the connection
- Alice is worried that if Eve, who works for the government, can tell that *S* contains encrypted data.



Solution:

- Use our protocol!

Technical Details

Technical Details

- Algorithmic process:

  1. Client1 chooses a byte string *B* to encrypt
  2. Client1 encrypts the byte string and applies our steganographic protocol to it to build a steganostring *S*
  3. Client2 receives *S* and inverts the steganographic protocol and decrypts the string to obtain *B*

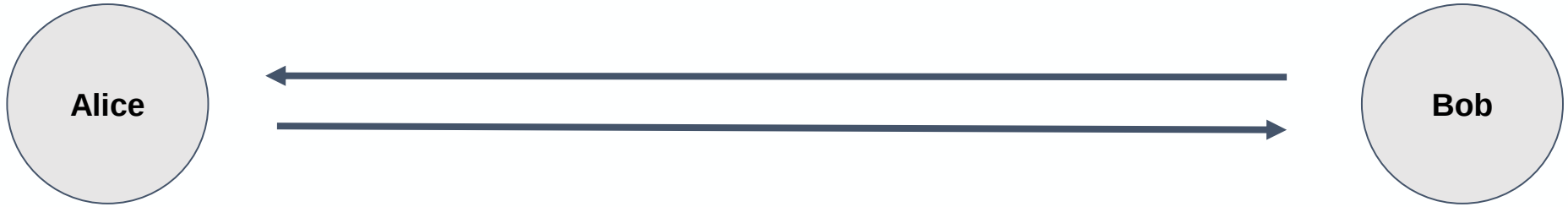
- AES_CTR is used for encryption of the byte string B to ciphertext
  - This gives the scheme forward security as attacker can’t read messages even if they detect transmission of steganographic packets

Technical Details *(continued)*

- Key exchange for the encryption is symmetric and is done on establishment of connection in chat client
- The model to generate random English words is also shared at startup so that sender and receiver generate the same words.
- We use a neural network model to generate randomized words to use as the steganographic blocks
  - With this, ciphertext bytes are transformed into steganographic blocks (randomized English words) to mask the ciphertext

How Our Protocol Works

Step 1
Key Exchange

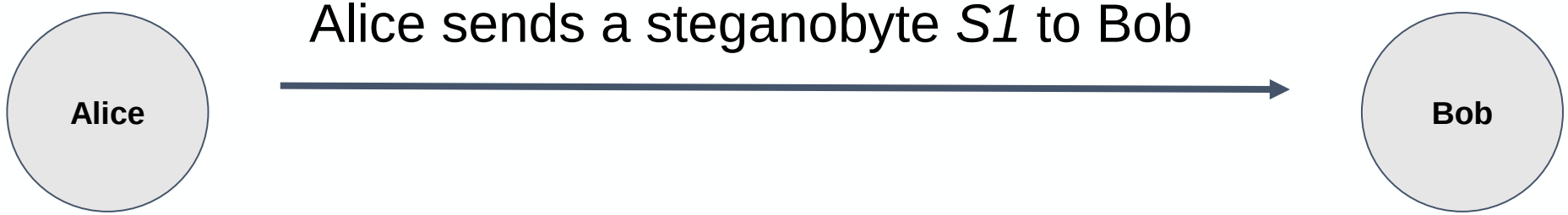


exchange(secret key *K* || text transformation array *T*)

How Our Protocol Works

Step 2

Alice wants to send Bob a byte *B1*



Alice generates *S1* from *B1* as such:

1. *B1* is enciphered by OTP, using a PRF F with key *K* to obtain *C1*

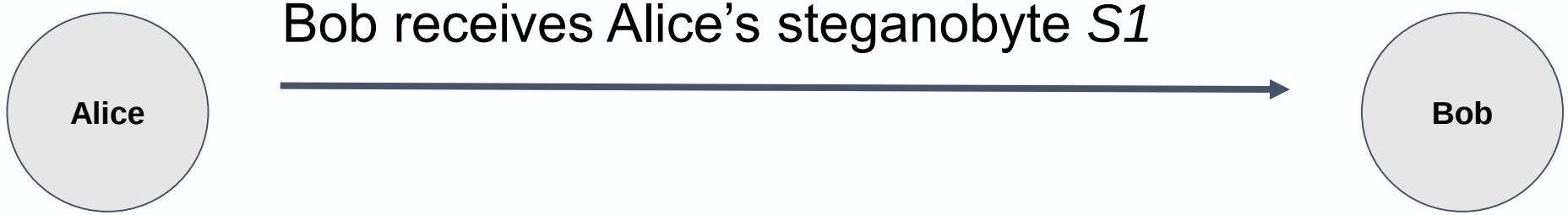
In other words, $C1 = B1 \oplus F(K, 1)$

1. *C1* is then transformed into a word in *T* at index *C1* to obtain steganobyte *S1*

In other words, $S1 = T(1)[C1]$

How Our Protocol Works

Step 3
Bob receives *S1* and decrypts it



Bob obtains *B1* from *S1* as such:

1. *S1* is receive and Bob computes T_inverse(1)[*S1*] to obtain *C1*
2. Bob then computes *C1* $\oplus$ F(*K*, 1) to obtain *B1*

As such, Eve can only see harmless english words being sent back and forth between Alice and Bob.

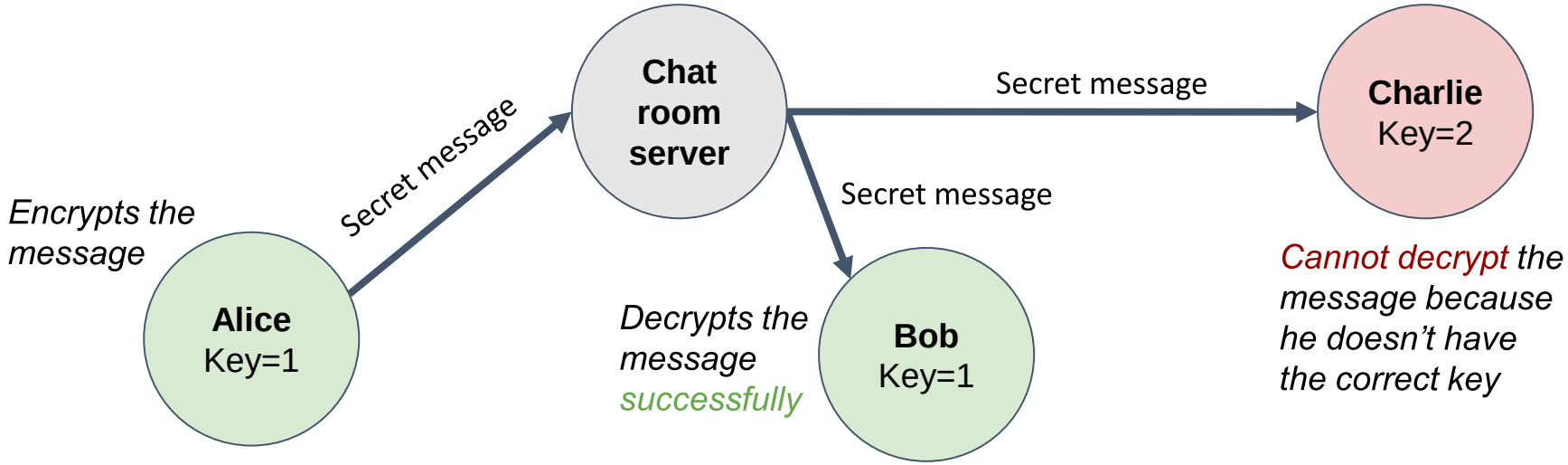
Results

Proof of Concept - “Chat Room”

Our protocol is shown in action through a chat room application.

Process:

1. Chat room server started
2. Clients join the chat room with predetermined secret keys
3. Clients can send and receive messages securely
4. Chat room server broadcasts messages to all clients



Results

- Functional protocol to send messages from a client to a receiver
  - Encrypted and steganographically secure
  - Easy-to-use Python package
  - Modeled using a chat-room
- Interactive Python scripts for chat room client and server
- Confirmed to work on LAN connection
  - Uses IP to transfer messages
  - Easily repurposable to any form of media transfer

GitHub repository: https://github.com/tkShir/EC521-Group6-Novel-Steganographic-Scheme

Demo

Video *(in case the demo doesn't work)*

Moving Forward

Moving Forward

- Include message authentication functionality
- Better models for more coherent text
- Portability
- More automation for text generation
- Easy to-use GUI
- Cryptanalysis and Steganalysis done by people seasoned in the respective fields
- Add feature to use images as cover files

Questions?

Thank you!

Team Member Responsibilities

- **Anas**: invented the protocol, outlined its components and the process of which the components are used; wrote the code for hiding ciphertext in human readable text, and the recovery of such ciphertext from the human readable text; demo and testing
- **Ivan**: [fill in]
- **Manan**: helped develop the idea for steganography and cryptographic hybridization, weighed in on AES_CTR for encryption and worked on slides and the final report documentation
- **Michael**: helped make chat room sender/receiver messaging protocols, worked on PowerPoint design and content, worked on debugging and testing IP messaging, helped with server
- **Takamitsu**: worked on creating general outline of the final slideshow, cryptographic mechanism, integration of messaging protocol and cryptographic mechanism, debugging tools for demo purpose, and groundwork for code management on GitHub
- **Samyak**: Helped with developing the technique using cryptography-steganography combination, mechanism for applying steganography on ciphertext and worked on the final report documentation and slides