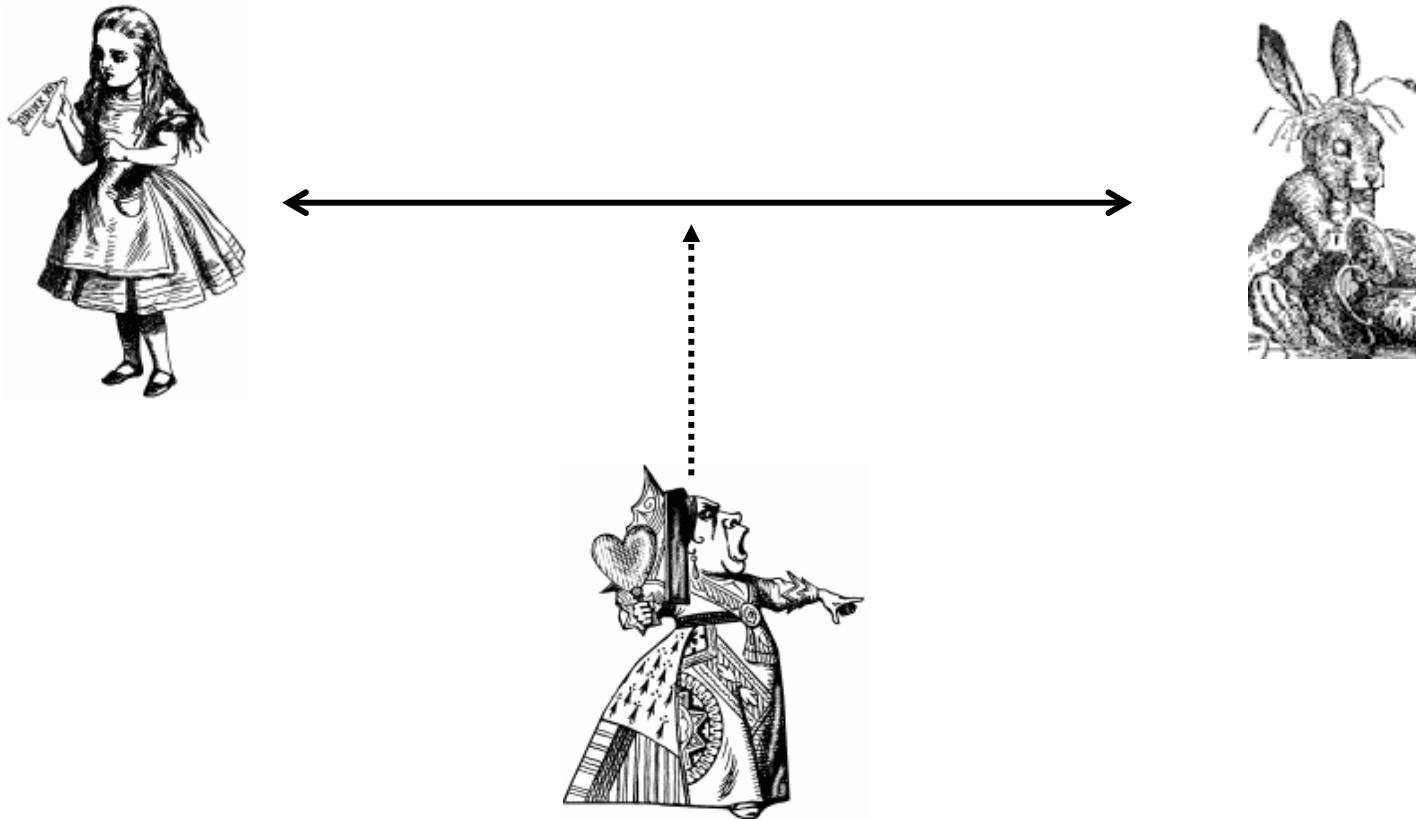# Cryptography and Network Security
CSL 759

Shweta Agrawal

Course Information

- 4-5 homeworks (20% total)
- 2 minors (15% each)
- A major (30%)
- A project (20%)
- Attendance required as per institute policy
- Scribe / Challenge questions (Extra Credit)

When : Tu-Wed-Fri 6 to 7 pm
Where : Bharti, room 201

Course Webpage : http://www.cse.iitd.ac.in/~shweta/teach.html

Administrative stuff

Teaching Assistants:

- Chandrika Bharadwaj chandrika.bhardwaj@gmail.com

- Abhay Gupta abhay3390@gmail.com

- Nikhil Kumar nikhilkumar4@gmail.com

- Utkarsh Ohm utkarshohm@gmail.com

Office Hours : TBA

Policies etc…

- Ask questions!
- Make the class interactive. We're all here to learn.
- Switch of cellphones, laptops, anything distracting.
- Highest ethical standards expected. Any dishonesty/cheating of any kind will result in failing the course.

Course Reading

- Will not follow any one book. But Katz-Kindell's "Introduction to Modern Cryptography" will be handy.

- Bellare-Goldwasser's lecture notes
  - http://cseweb.ucsd.edu/~mihir/papers/gb.pdf

- Lecture notes by Yevgeniy Dodis (http://www.cs.nyu.edu/courses/spring12/CSCI-GA.3210-001/index.html) and Luca Trevisan (http://theory.stanford.edu/~trevisan/cs276/)

What is this course about

- Theoretical foundations of cryptography
- Mathematical modeling of real world attack scenarios
- Reductions between crypto primitives and hard number theoretic problems
- Using cryptographic building blocks to build more complex real world protocols

What this course is NOT about

- Implementing secure systems
- Real world attacks / hacking
- Analyzing hardness of underlying number theoretic problems such as factoring etc

You can do your projects on these topics if you like!

Course Outline

- Foundations : Principles of crypto design, number theory, OWF, OWP, TDP, PRGs, PRFs, MACs

- Constructions : symmetric and public key crypto, digital signatures, MPC

- Advanced Topics: Zero Knowledge, Functional encryption, fully homomorphic encryption, broadcast encryption etc
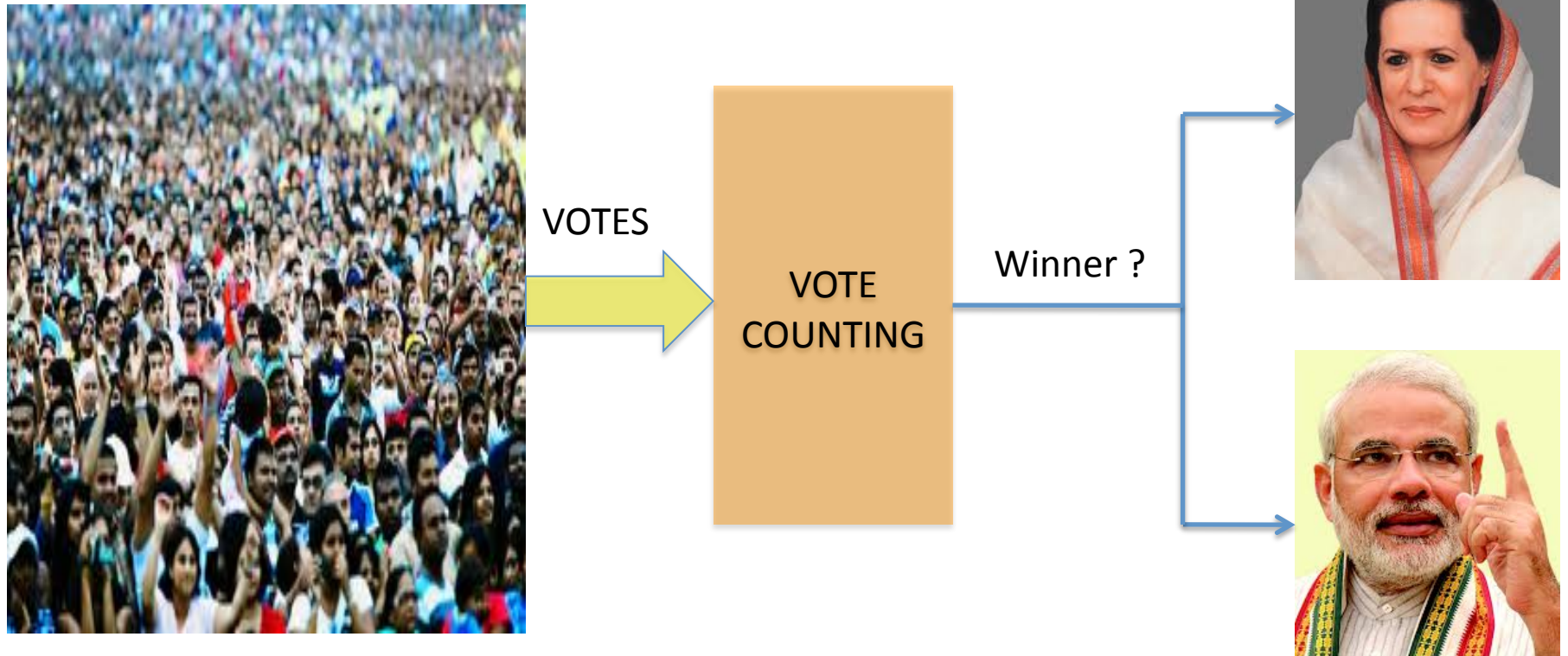
Cryptography

- A mathematical science of controlling access to *information*

- Cryptography deals with methods for protecting the *privacy and integrity* while *preserving functionality* of computer and communication systems.

What would we like to achieve?

Real World Problems

#1 : Secure Elections

Multi-party computation!

VOTES

VOTE COUNTING

Winner ?

CORRECT : Winner determined correctly

SECURITY : individual vote privacy maintained

#2 : Protecting your code

#3 : Activism with safety

How Iran's political battle is fought in cyberspace

International media were banned from reporting on protests in [...]
on Twitter filled the gap.

Iran's Basij Sisters suppressed election protests

Wednesday, 05 August 2009

Probabilistic algorithm

C = Encrypt ("The election was rigged", R)

Under coercion, reveal R' s.t. C =("Really like to cook", R')

R, R' :
Random bits

Where is my VOTE?

Deniable Encryption!

#4: Computing on encrypted data

Cloud Computing

Having secure access to all your applications and data from any network device

❖ Users access data and infrastructure on-the-go

❖ Cloud stores data about you, me and many more

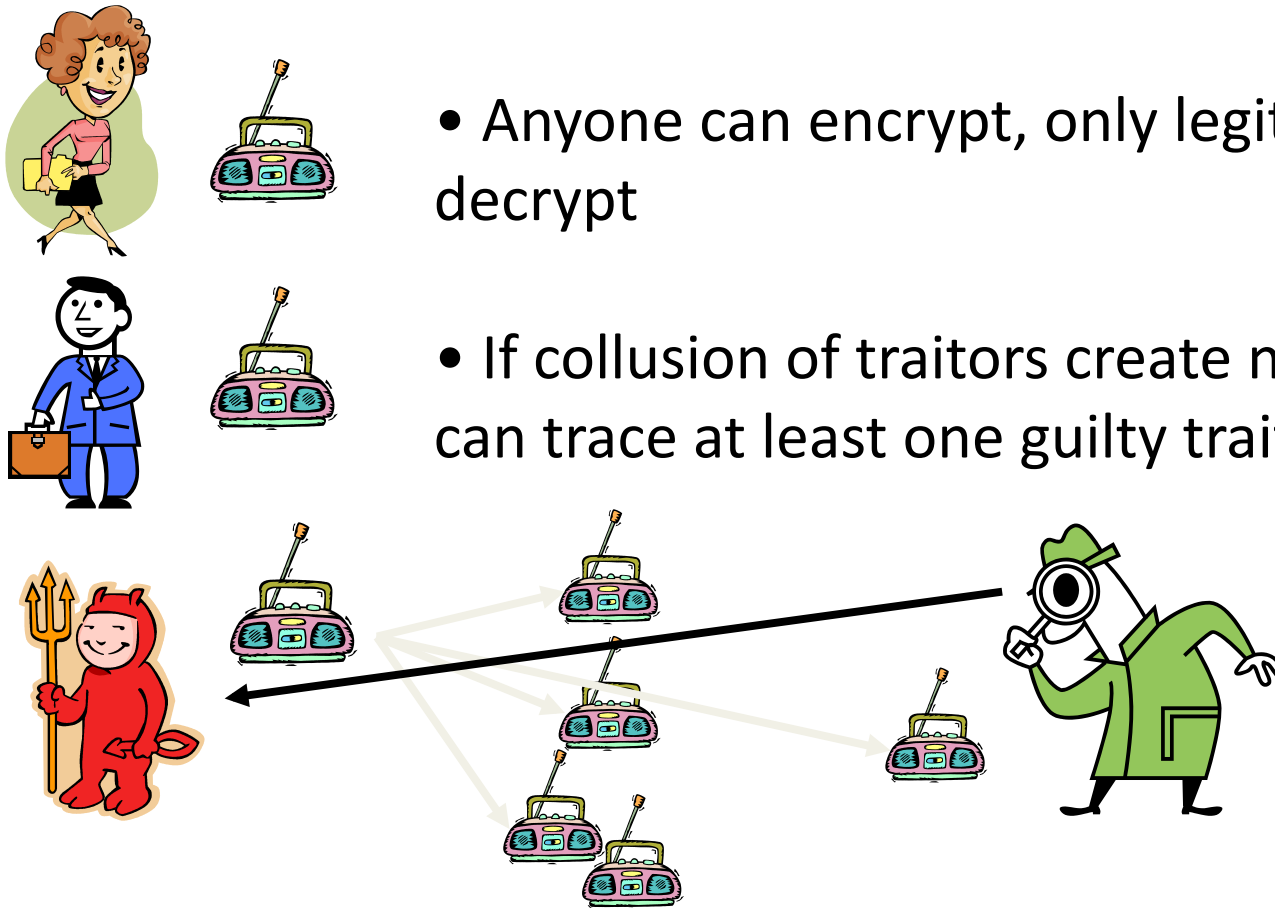❖ I should learn information about myself but no information about you

#5: Traitor Tracing

#5: Traitor Tracing

- N users in system, One PK, N SKs

- Anyone can encrypt, only legitimate user should decrypt

- If collusion of traitors create new secret key SK^*, can trace at least one guilty traitor.

This course ....

1. How can we build these things from math ?

2. What guarantees can we have ?

3. How do we move from messy real world scenarios to clean mathematical definitions?

4. How do theorems in math say anything about real world attacks?

Building Blocks

St. Pancreas International Station - 18 months, 150,000 LEGO bricks
Warren Elsemore

What he started with

Building cryptography

- Same idea!

One way functions, trapdoor permutations, Pseudo random generators, PRFs
Symmetric key crypto, public key crypto, Digital signatures ......

Multiparty computation, homomorphic encryption, functional encryption, deniable signatures, obfuscation, traitor tracing .....

Principles of Crypto Design [Katz-Lindell]

1. Formulate a rigorous and precise definition of security for cryptosystem – security model.

2. Precisely formulate the mathematical assumption (e.g. factoring) on which the security of the cryptosystem relies.

3. Construct cryptosystem (algorithms) and provide proof (reduction) that cryptosystem satisfying security model in (1) is as hard to break as mathematical assumption in (2).
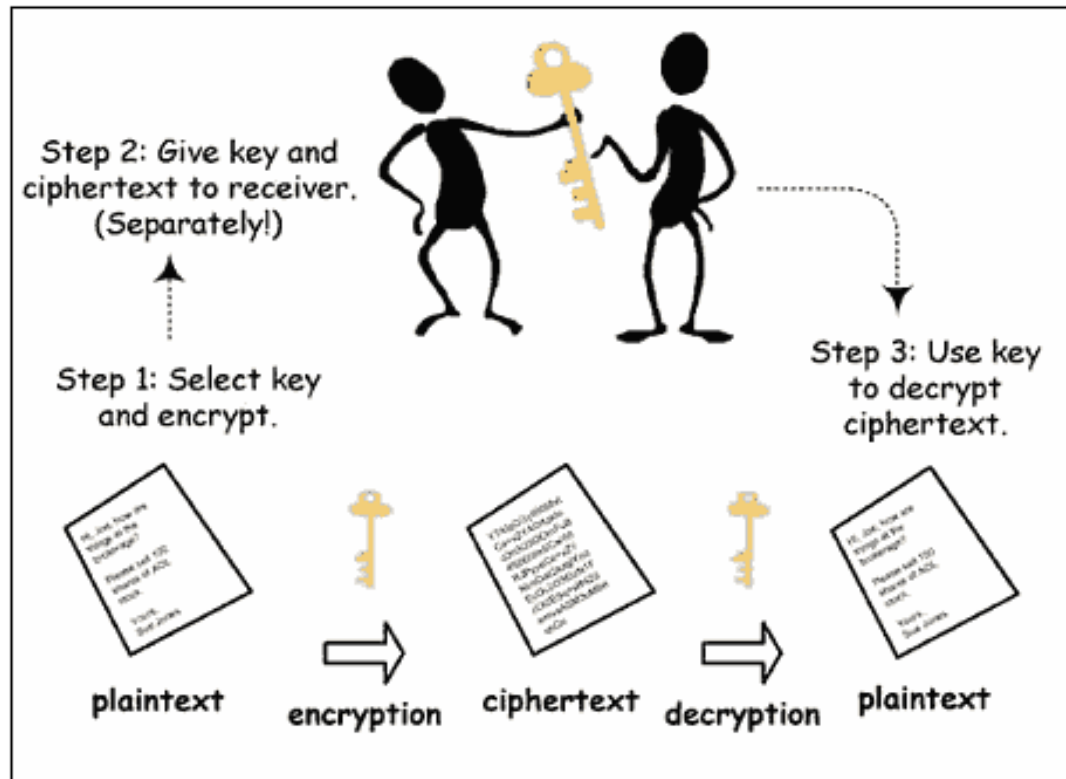
1: Security Model

Real world attacks ⟷ Crypto Proofs

Security Model : Mathematical definition that scheme has to satisfy

Scheme achieves security in given model = Scheme secure against attacks captured by that model

Case Study : Secure encryption

- Every pair of users must share a unique secret key
- Need key to encrypt and decrypt. Intuitively, only holder of secret key should be able to decrypt

Case Study : Secure encryption

Syntax

We must construct the following algorithms:

1. Keygen : Algorithm that generates secret key K

2. Encrypt(K,m) : Algorithm used by Alice to garble message m into "ciphertext" CT

3. Decrypt(K, CT) : Algorithm used by Bob to recover message m from ciphertext CT.

Case Study : Secure encryption

How should security of encryption be defined?

Answer 1 : Upon seeing ciphertext, Eve should not be able to find the secret key.

But our goal is to protect the message!

Consider encrypt algorithm that ignores the secret key and just outputs the message. An attacker cannot learn the key from the ciphertext but learns the entire message!

Case Study : Secure encryption

Answer 2 : Upon seeing ciphertext, Eve should not be able to find the message.

Is it secure intuitively to find 99% of the mesg?

Answer 3 : Upon seeing ciphertext, Eve should not be able to find a single character of the message.

Is it ok to leak some property of the mesg, such as whether m> k?

Case Study : Secure encryption

Answer 4 : Any function that Eve can compute given the ciphertext, she can compute without the ciphertext.
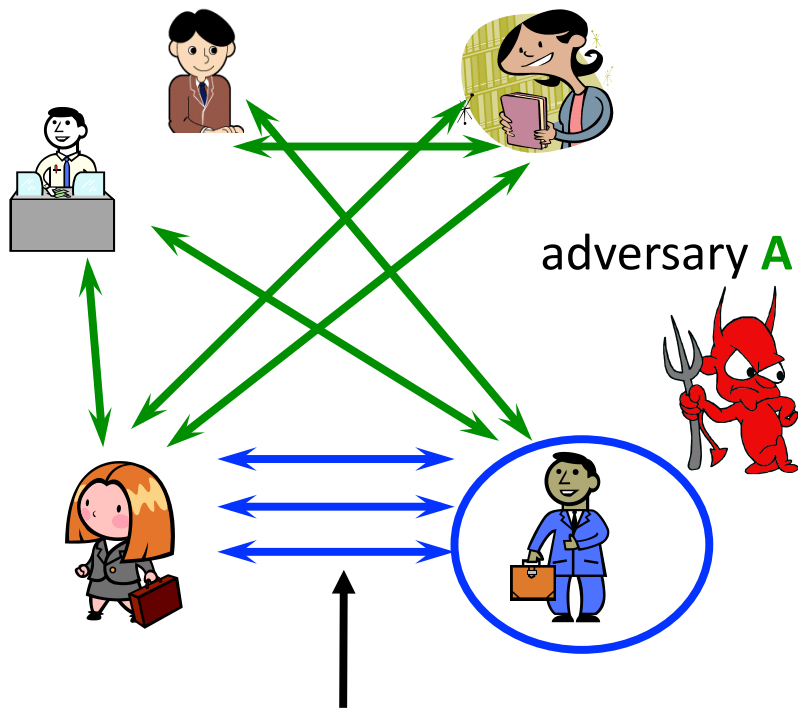
Still need to specify :

- Can Eve see ciphertexts of messages of her choice?

- Can Eve see decryptions of some ciphertexts?

- How much power does she have?

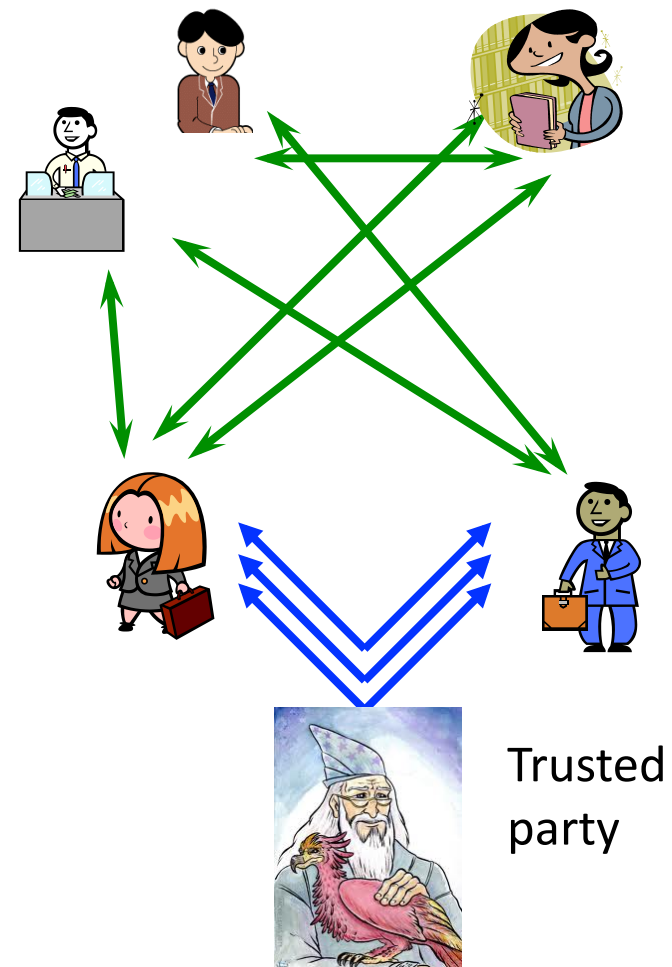# What about security of real world functionalities?
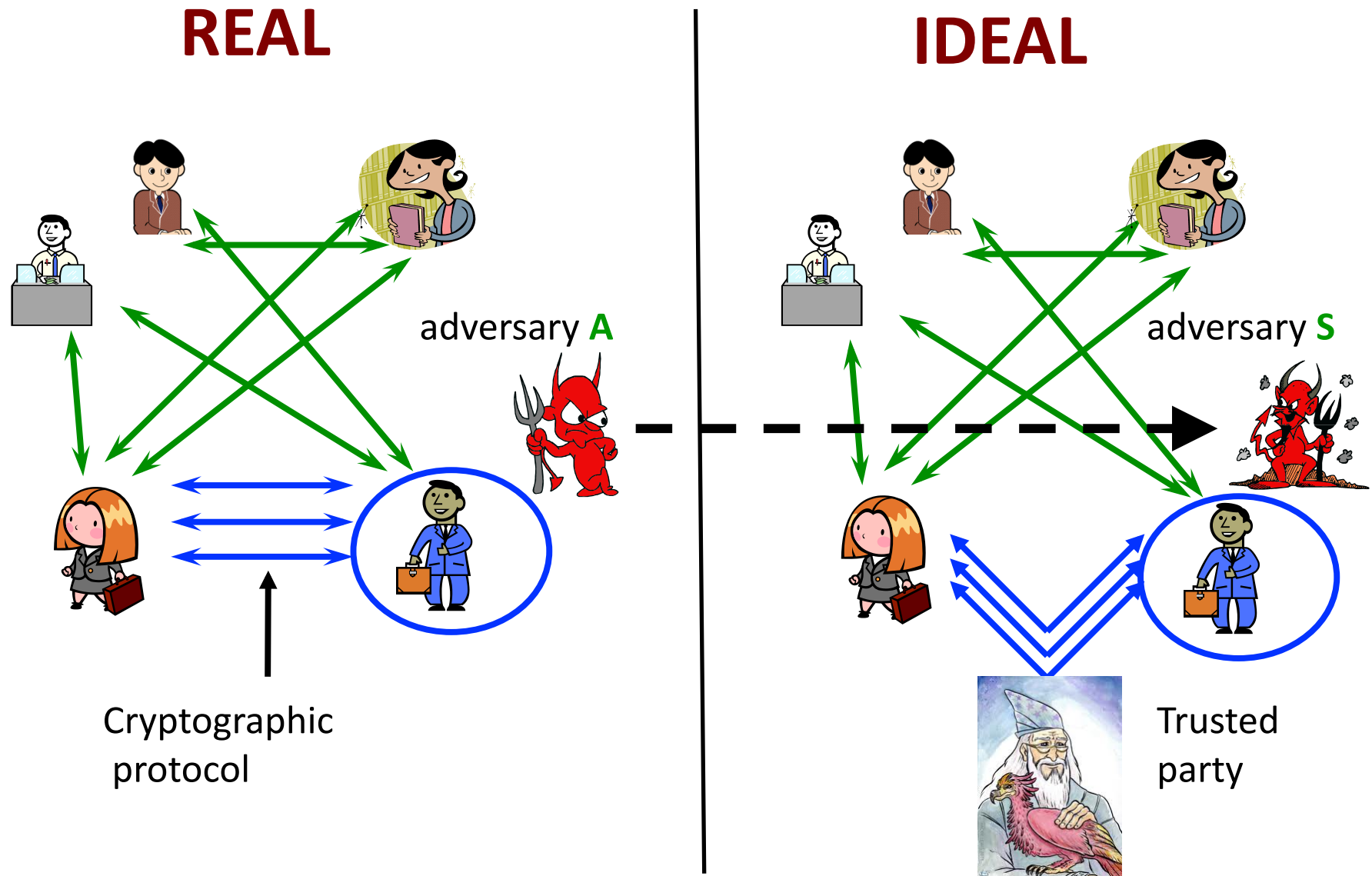
Ideal Security definition

REAL

IDEAL

adversary **A**

Cryptographic
protocol

Trusted
party

Ideal Security definition

REAL

adversary **A**

Cryptographic
protocol

IDEAL

adversary **S**

Trusted
party

Ideal Security definition

REAL

IDEAL

adversary **A**

adversary **S**

≈

Cryptographic
protocol

Trusted
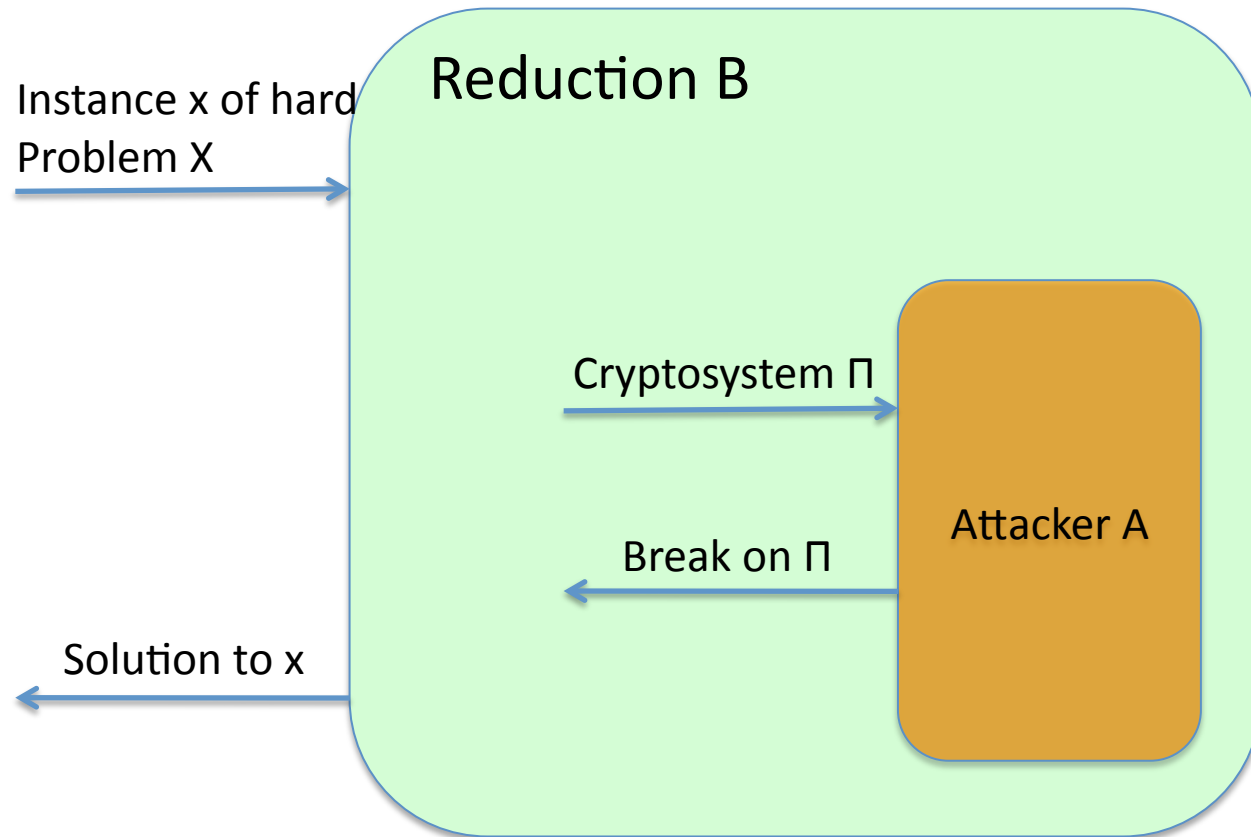party
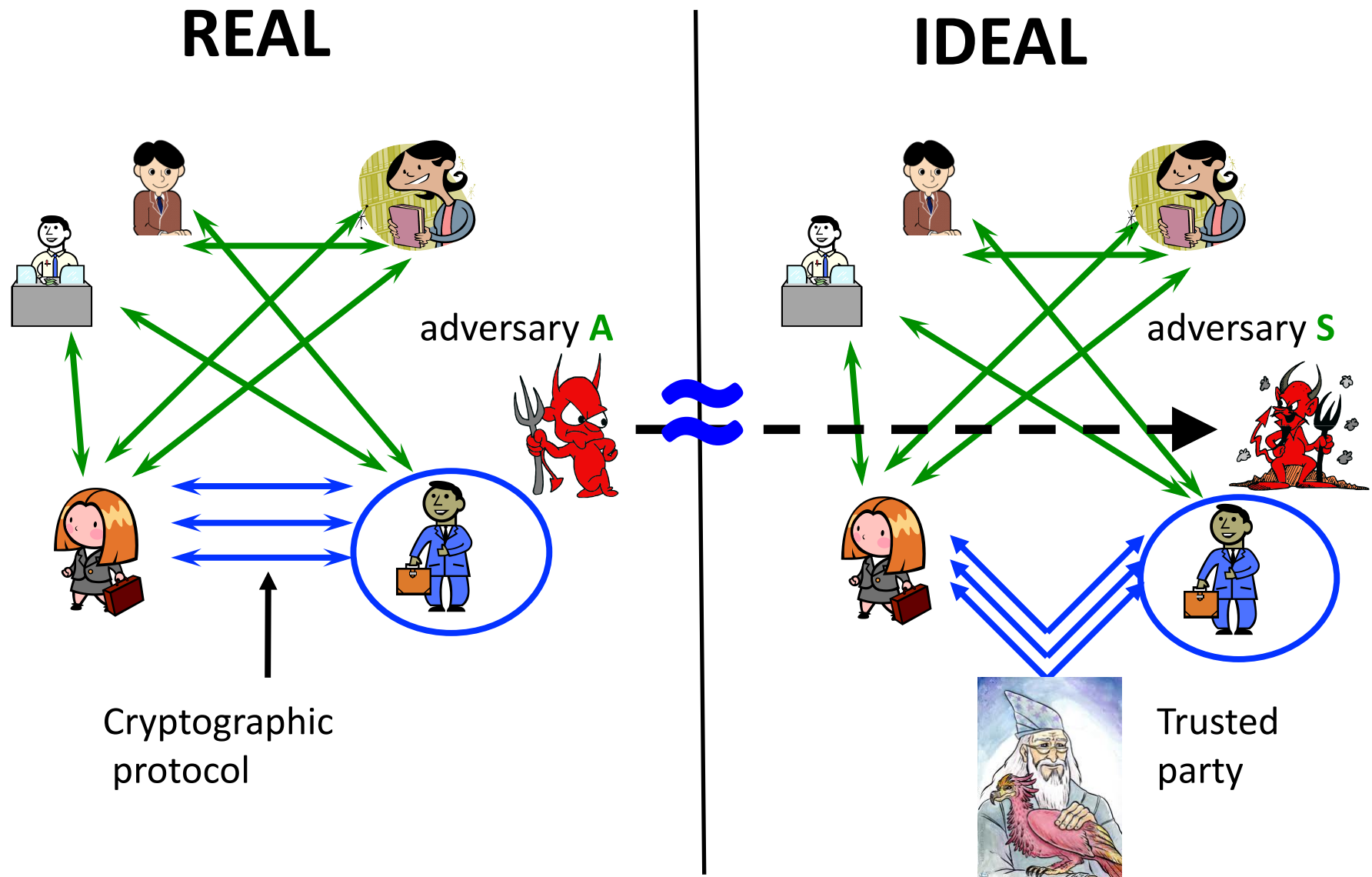
2: Mathematical Assumption

- Trivial assumption : my scheme is secure
- Use minimal assumptions
  - Existence of one way functions
- Use well studied assumptions
  - Examples: factoring, discrete log, shortest vector problem etc…

3: Reduction

Reduction B

Instance x of hard Problem X

Cryptosystem Π

Attacker A

Break on Π

Solution to x

3: Reduction

Show how to use an adversary for breaking primitive 1 in order to break primitive 2

Important :

- Run time: how does T_1 relate to T_2
- Probability of success: how does $Succ_1$ relate to $Succ_2$
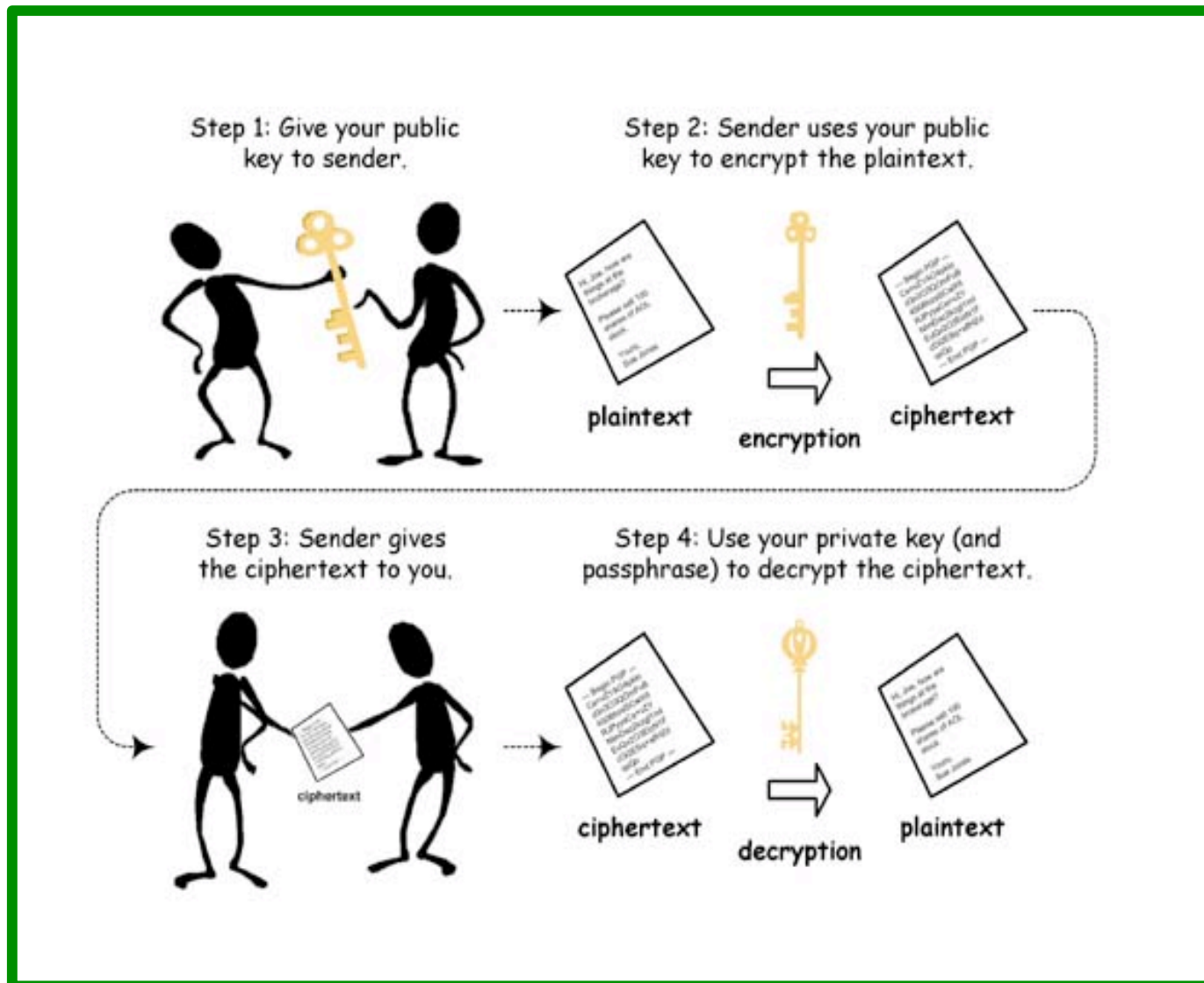- Access to the system 1 vs. 2

Secret Key Encryption Construction

- Keygen : Pick a random string r . Set K = r. Give to both Alice and Bob

- Encrypt (m, K) : CT = $m \oplus r$

- Decrypt (CT, K) : $m \oplus r \oplus r = m$

Only works for single use of r!

How to generate shared key?

Public Key Cryptography

Step 1: Give your public key to sender.

Step 2: Sender uses your public key to encrypt the plaintext.

plaintext → encryption → ciphertext

Step 3: Sender gives the ciphertext to you.

ciphertext

Step 4: Use your private key (and passphrase) to decrypt the ciphertext.

ciphertext → decryption → plaintext

What we need…

1. **Invertible**: It must be possible for Alice to decrypt encrypted messages.

2. **Efficient to compute**: It must be reasonable for people to encrypt messages for Alice.

3. **Difficult to invert**: Eve should not be able to compute m from the "encryption" f(m).

4. **Easy to invert given some auxiliary information**: Alice should restore m using SK.
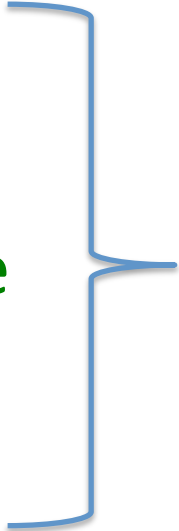
What we need…

1. Invertible

2. Efficient to compute

3. Difficult to invert

One way functions!

4. Easy to invert given some auxiliary information

What we need…

1. Invertible

2. Efficient to compute

3. Difficult to invert

} One way permutations!

4. Easy to invert given some auxiliary information

What we need…

1. Invertible

2. Efficient to compute

3. Difficult to invert

4. Easy to invert given some auxiliary information

Trapdoor permutations!

Up Next …

- Discuss some number theory

- Introduce conjectured hard problems such as factoring, discrete log.

- Build candidate one way functions, one way permutations and trapdoor permutations

- Construct proofs of security.