Introduction to Cryptography

David Brumley
dbrumley@cmu.edu
Carnegie Mellon University

Credits:
Many slides from Dan Boneh's June 2012 Coursera crypto class, which is great!

Cryptography is Everywhere

Secure communication:

- web traffic: HTTPS
- wireless traffic: 802.11i WPA2 (and WEP), GSM, Bluetooth

Encrypting files on disk: EFS, TrueCrypt

Content protection:

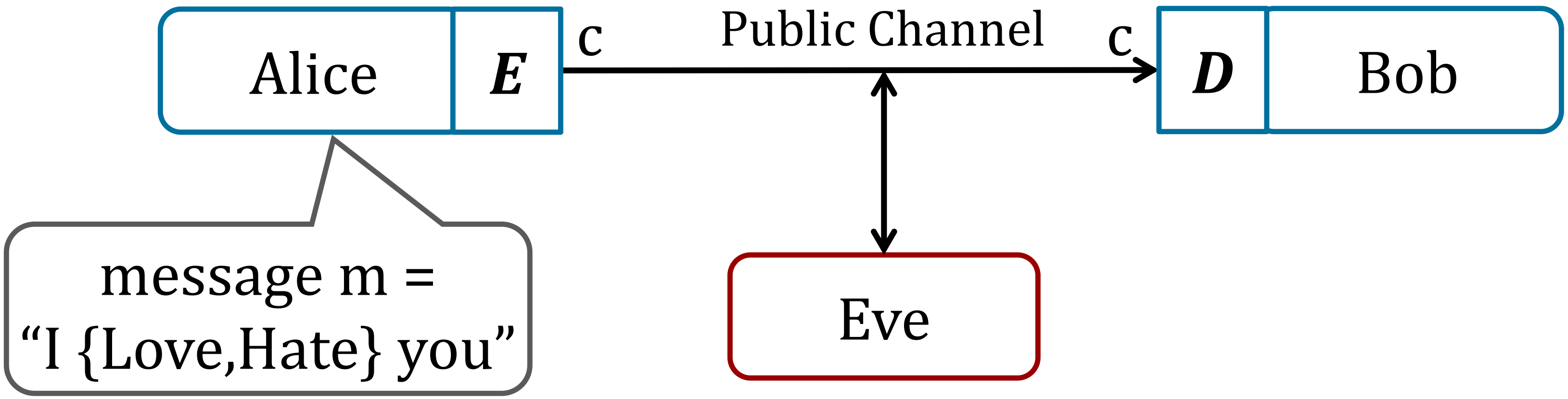
- CSS (DVD), AACS (Blue-Ray)

User authentication

- Kerberos, HTTP Digest

… and much much more

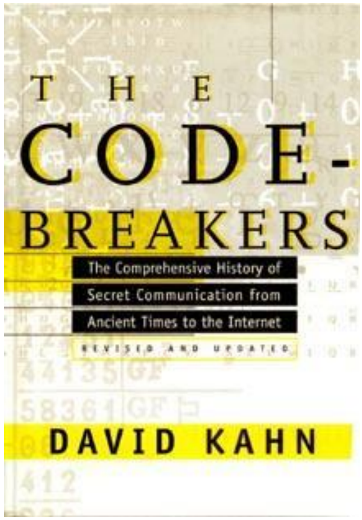
Goal: Protect Alice's Communications with Bob



Eve is a <u>*very*</u>
powerful, smart person
(say any polynomial time alg)

History of Cryptography

David Kahn, "The code breakers" (1996)



The Puzzle Palace

Caesar Cipher: $c = m + 3$ % 26

D E F G

A B C D E F G H I J

K L M N O P Q R

S T U V W X Y Z

m CAT

FDW

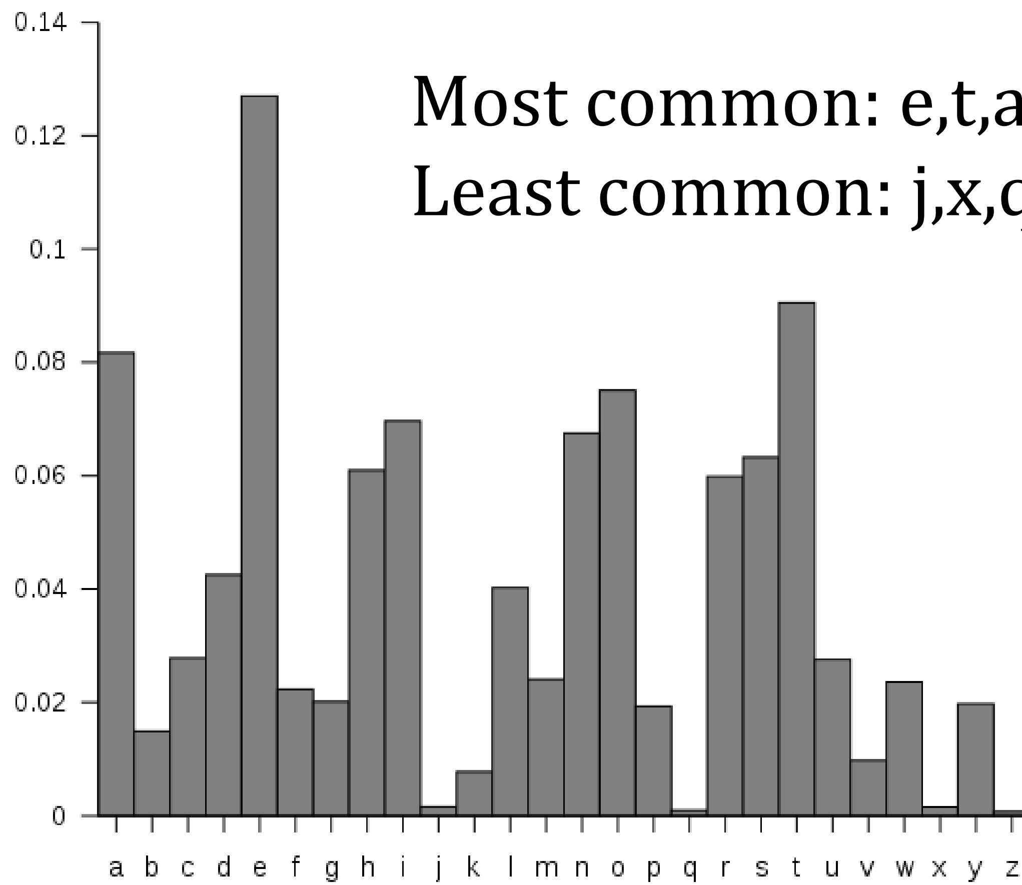
Julius Caesar
100 BC- 44 BC

How would you ***attack*** messages encrypted with a substitution cipher?

Attacking Substitution Ciphers

Trick 1: *Word Frequency*

Trick 2: *Letter Frequency*



Most common: e,t,a,o,i,n

Least common: j,x,q,z

image source: wikipedia

The

Jvl mlwclk yr jvl owmwez twp yusl w zyduo

t t

pjdcluj mqil zydkplmr. Hdj jvlz tykilc vwkc jy

mlwku jvl wkj yr vwsiquo, tvqsv vlmflc mlwc

jvlg jy oklwjulpp. Zyd vwnl jvl fyjlujqwm jy cy

the TOAST

jvl pwgl. Zydk plsklj fwpptykc qp: JYWPJ

J -> T

V -> H

L -> E

http://picoctf.com

Classical Approach: Iterated Design

Scheme 1 → Broken

↓

Scheme 2 → Broken

↓

Scheme 3 → Deploy → Broken

↓

...

Enigma

No way to say anything is secure
(and you may not know when broken)

Iterated design was only one we knew until 1945

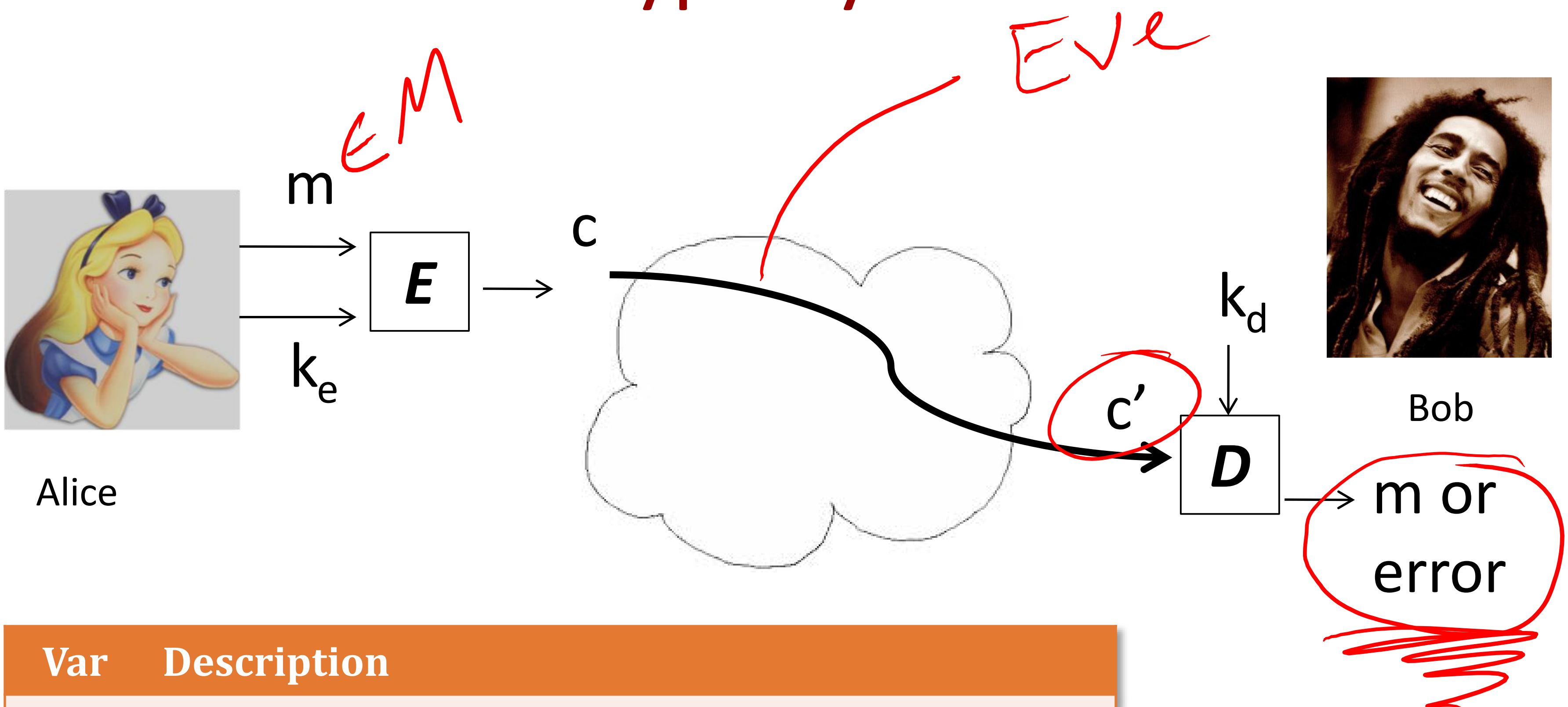
Claude Shannon: 1916 - 2001

Claude Shannon

- Formally define:
  - *security goals*
  - *adversarial models*
  - *security of system wrt goals*

- Beyond iterated design: Proof!

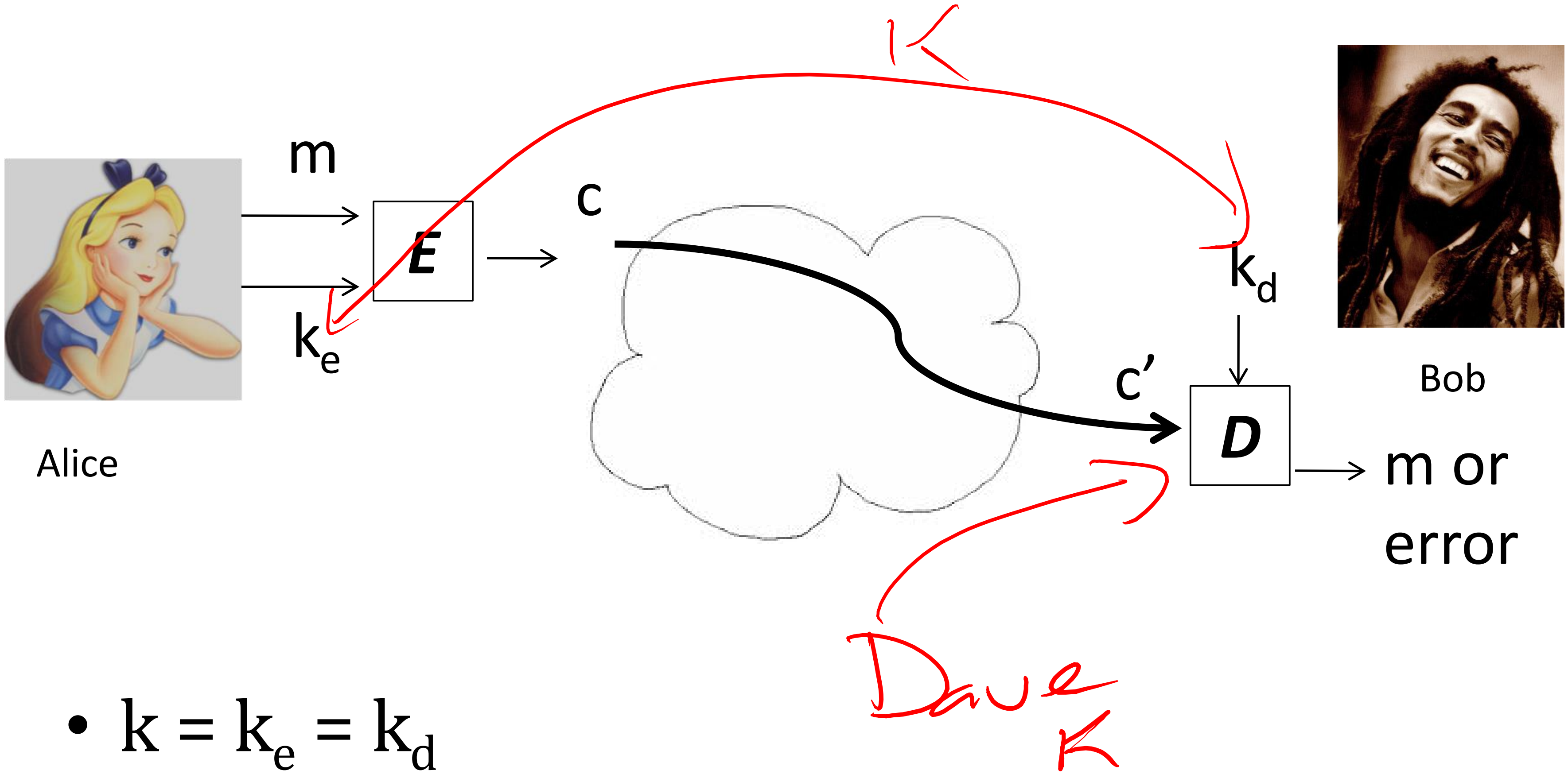
Cryptosystem



$c' \neq c$

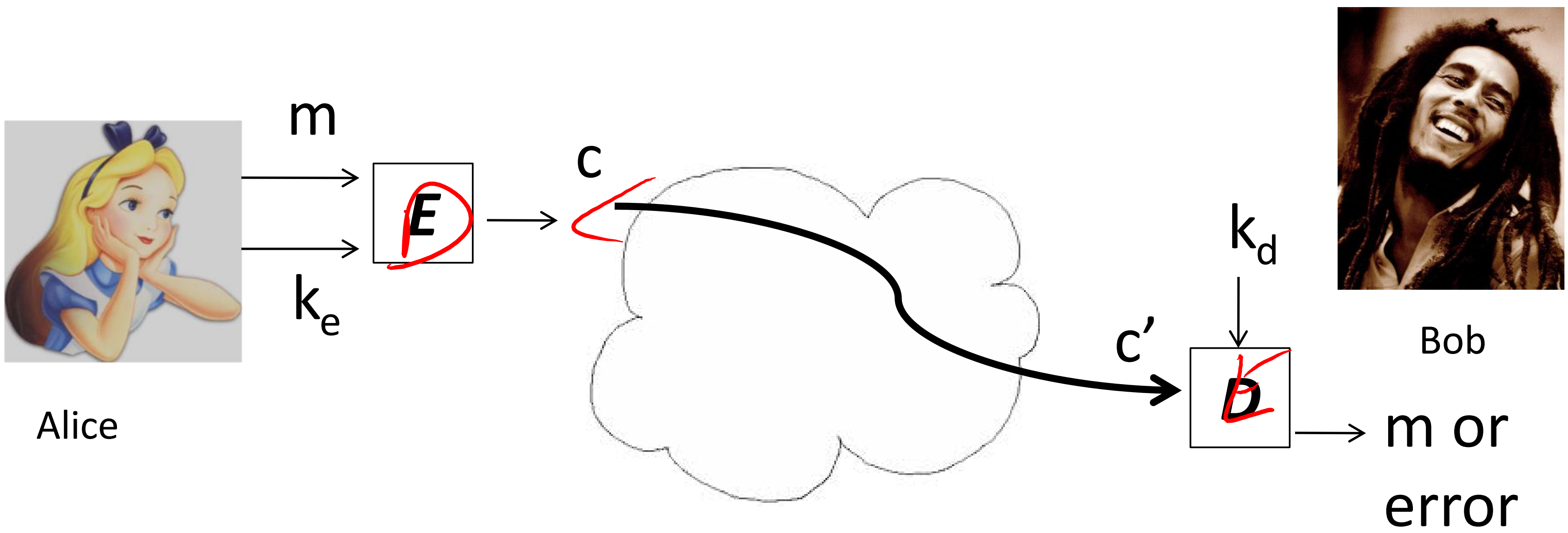
Var	Description
m	Message (aka plaintext). From the <u>*message space*</u> M
c	Ciphertext. From the <u>*ciphertext space*</u> C
E	Encryption Algorithm
D	Decryption Algorithm
$\mathbf{k_e}$	Encryption key. From the <u>*key space*</u> K
$\mathbf{k_d}$	Decryption. Also from the <u>*key space*</u> K

Symmetric Cryptography



- $k = k_e = k_d$
- Everyone who knows k knows the full secret

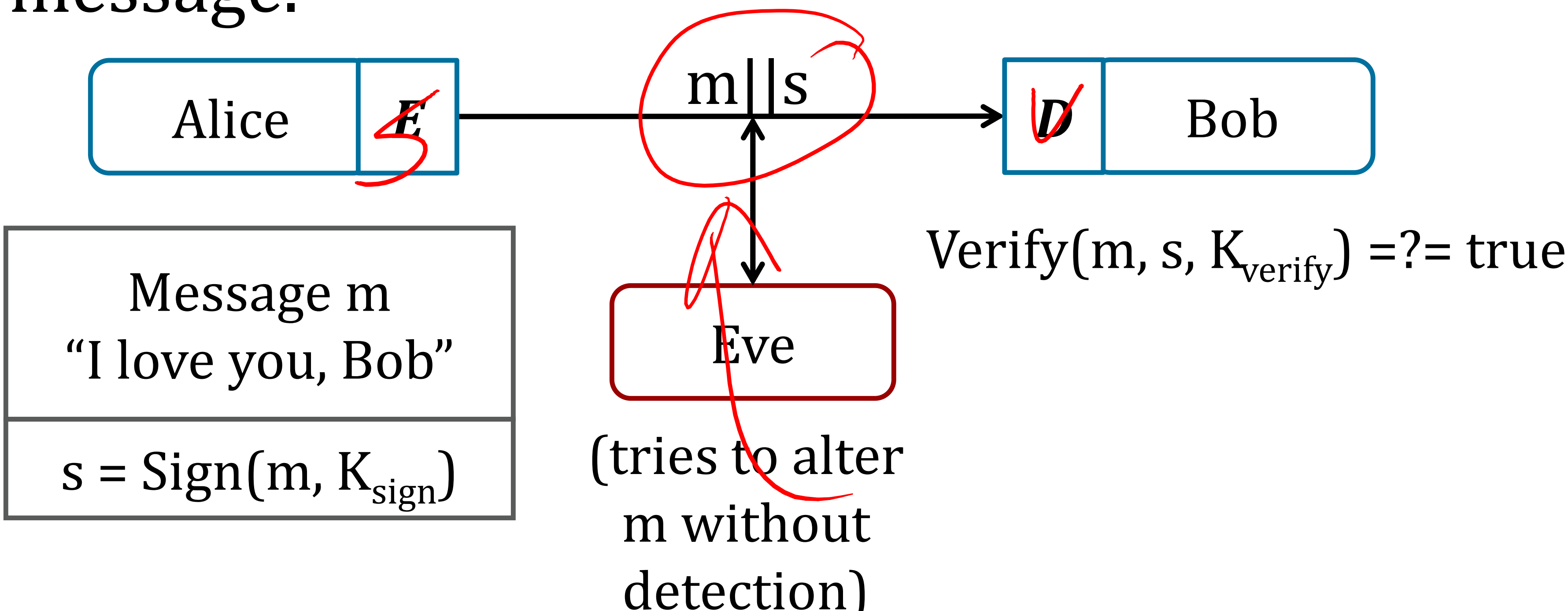
Asymmetric Cryptography



- k_e != k_d
- Encryption Example:
  - Alice generates private (K_d)/public(K_d) keypair. Sends bob public key
  - To encrypt a message to Alice, Bob computes c = E(m,K_e)
  - To decrypt, Alice computes m = D(m, K_d)

But all is not encryption

Message Authentication Code: Only people with the private key k could have sent the message.



An interesting story...

1974

- A student enrolls in the Computer Security course @ Stanford

- Proposes idea for public key crypto. Professor shoots it down

Picture: http://www.merkle.com

1975

- Ralph submits a paper to the Communications of the ACM

- "I am sorry to have to inform you that the paper is not in the main stream of present cryptography thinking and I would not recommend that it be published in the Communications of the ACM. Experience shows that it is extremely dangerous to transmit key information in the clear."

$> B$

Today

Ralph Merkle:
A Father of
Cryptography

Picture: http://www.merkle.com

Covered in this class

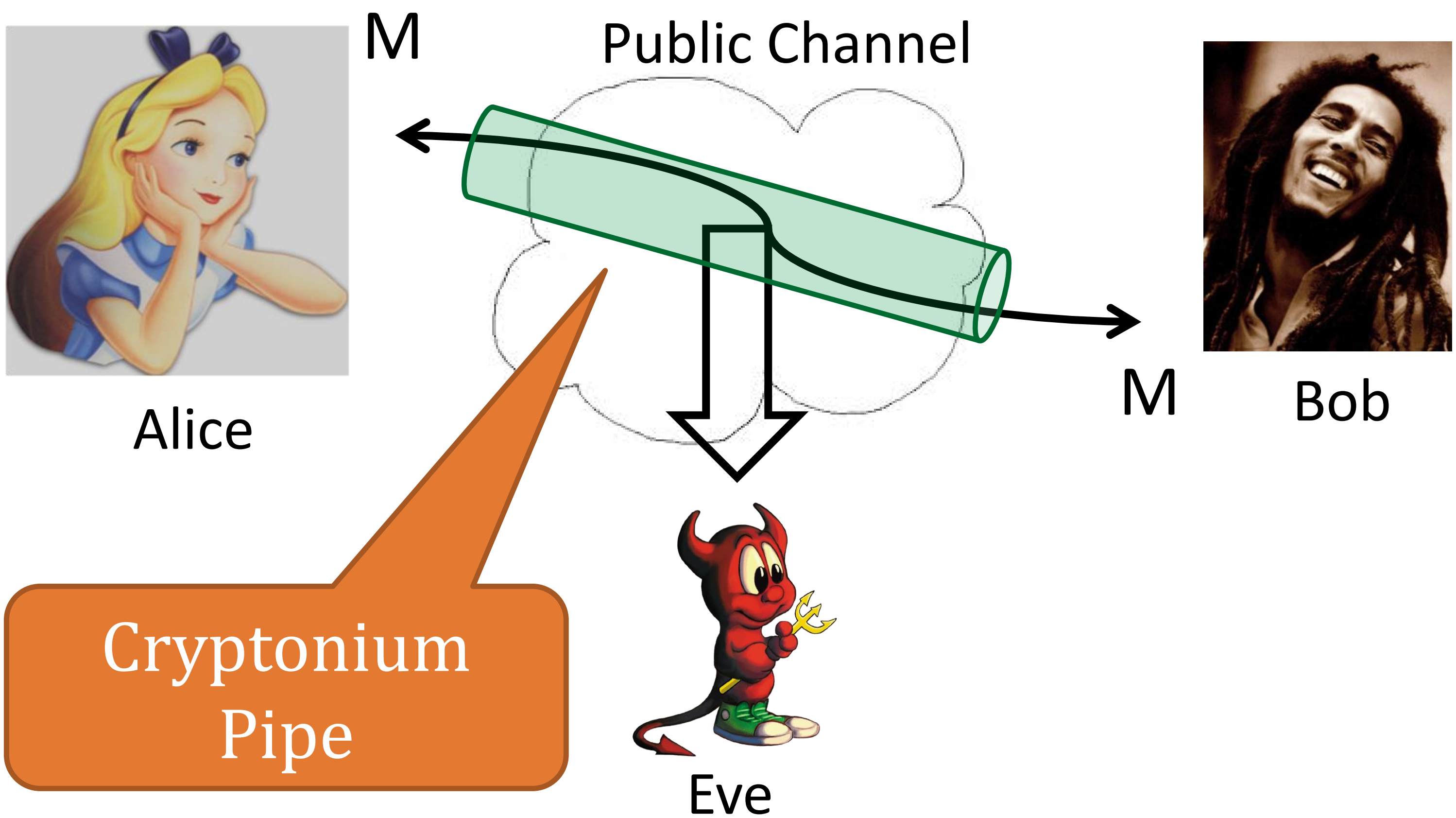
everyone shares *same* secret k

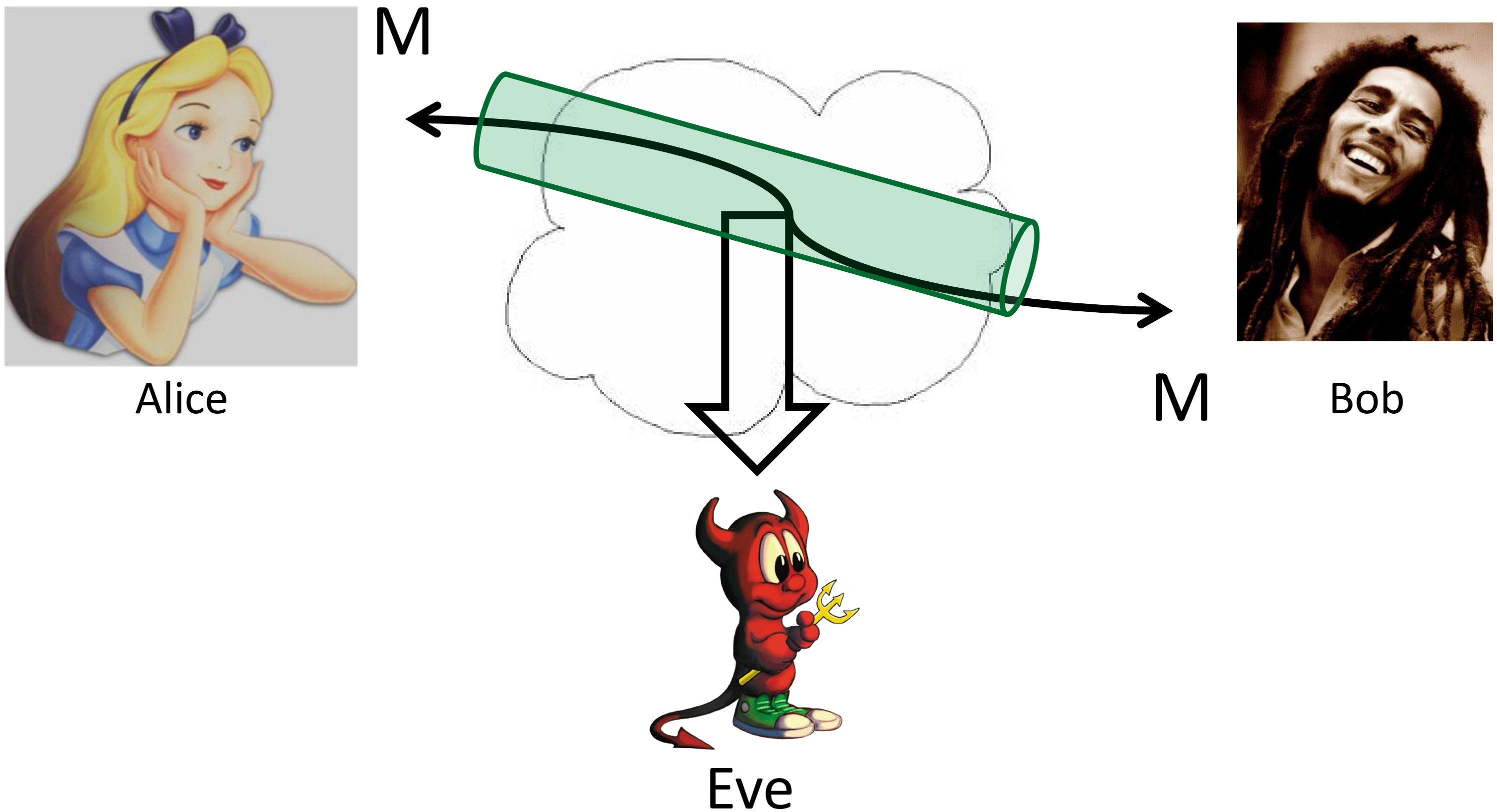
Only 1 party has a secret

	Symmetric Trust Model	Asymmetric Trust Model
Message Privacy	Private key encryption • Stream Ciphers • Block Ciphers	Asymmetric encryption (aka public key)
Message Authenticity and Integrity	Message Authenticity Code (MAC)	Digital Signature Scheme

Principle 1: All algorithms *public*
Principle 2: Security is determined *only* by key size
Principle 3: If you roll your own, it will be *insecure*

Security Goals

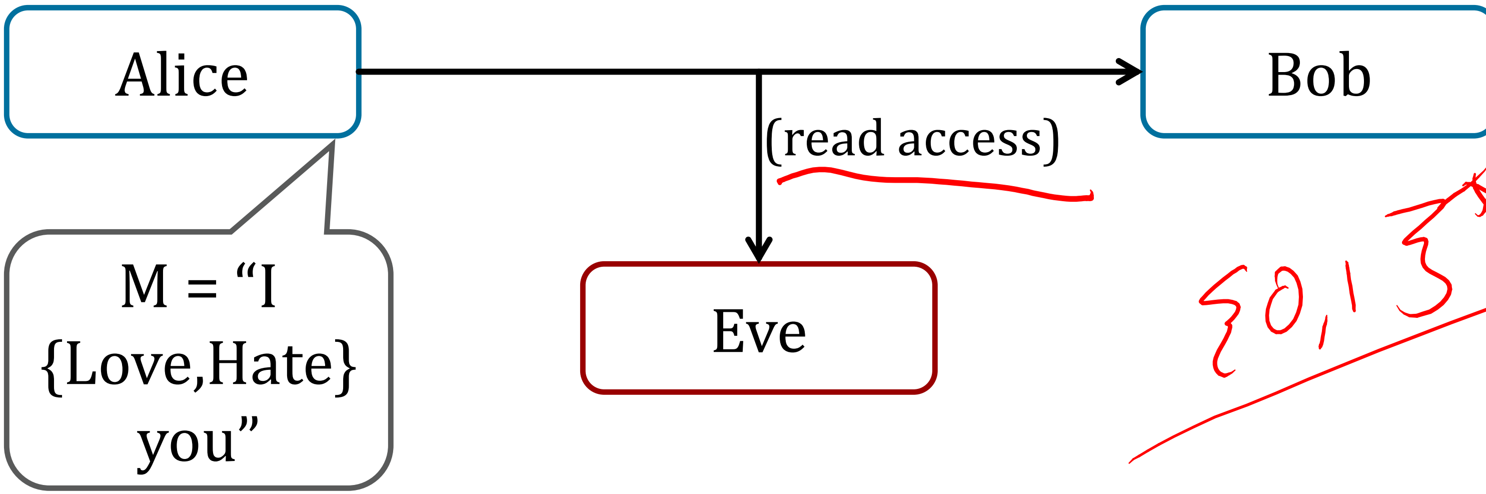




One Goal: *Privacy*

Eve should not be able to read M.

Not even 1 bit...

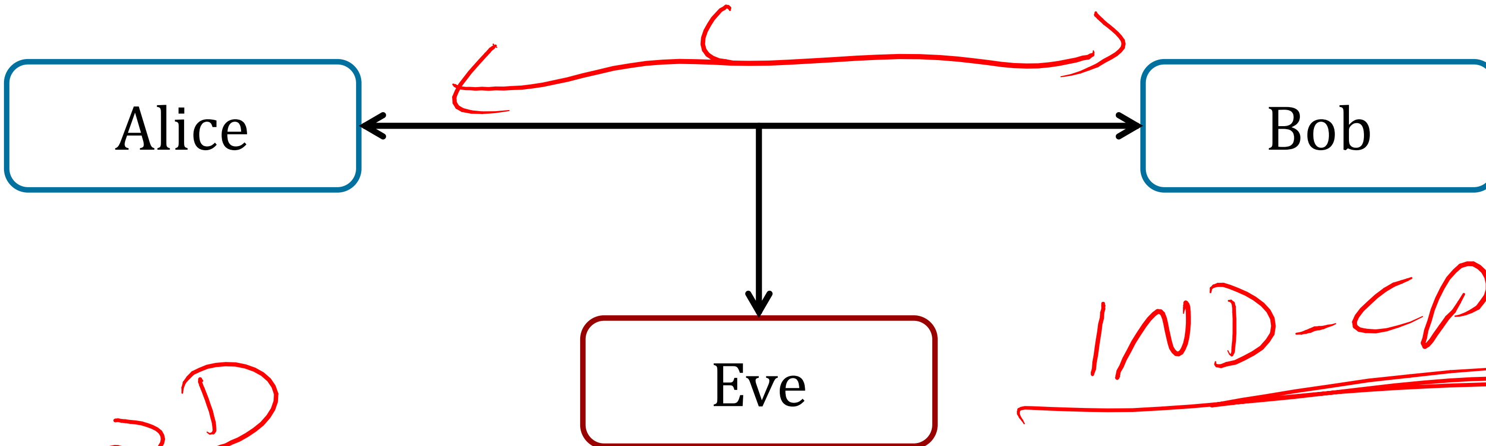


E(0) E(1)

~~$\{0,1\}^*$~~

Security guarantees should hold for all messages, not just a particular kind of message.

which message was sent.



IND-CPA

4 Ways to break crypto:

- Ciphertext Only   c
- Known Plaintext Attack (KPA)   (m, c) ∉ m'
- Chosen Plaintext Attack (CPA)   (m, c) ∈ m'E
- Chosen Ciphertext Attack (CCA)   D

Symmetric Cryptography

Defn: A symmetric key cipher consists of 3 polynomial time algorithms:

1. *KeyGen(l)*: A randomized algorithm that returns a key of length *l*. *l* is called the *security parameter.*
2. *E(k,m):* A potentially randomized alg. that encrypts *m* with *k.* It returns a *c* in *C*
3. *D(k,c):* An always deterministic alg. that decrypts c with key k. It returns an *m* in *M*.

Type Signature

$$\mathsf{KeyGen} : \ell \to K$$

$$\mathsf{E} : K \times M \to C$$

$$\mathsf{D} : K \times C \to M$$

And (correctness condition)

$$\forall m \in M, k \in K : D(k, E(k, m)) = m$$

The One Time Pad

Miller, 1882 and Vernam, 1917

$|m| = n$ $\qquad$ $K \xleftarrow{R} \{0,1\}^n$

$$E(k, m) = k \oplus m = c$$

$$D(k, c) = k \oplus c = m$$

M = C = K = $\{0,1\}^n$

m:	0	1	1	0	1	1	0
$\oplus$ k:	1	1	0	1	0	0	0
c:	1	0	1	1	1	1	0
$\oplus$ k:	1	1	0	1	0	0	0
m:	0	1	1	0	1	1	0

The One Time Pad

Miller, 1882 and Vernam, 1917

$$E(k, m) = k \oplus m = c$$

$$D(k, c) = k \oplus c = m$$

Is it a cipher?

✓ Efficient

✓ Correct

$$\begin{aligned} D(k, E(k, m)) &= D(k, k \oplus m) \\ &= k \oplus (k \oplus m) \\ &= 0 \oplus m \\ &= m \end{aligned}$$

Question

Given *m* and *c* encrypted with an OTP, can you compute the key?

$E(m,k) = k \oplus m$

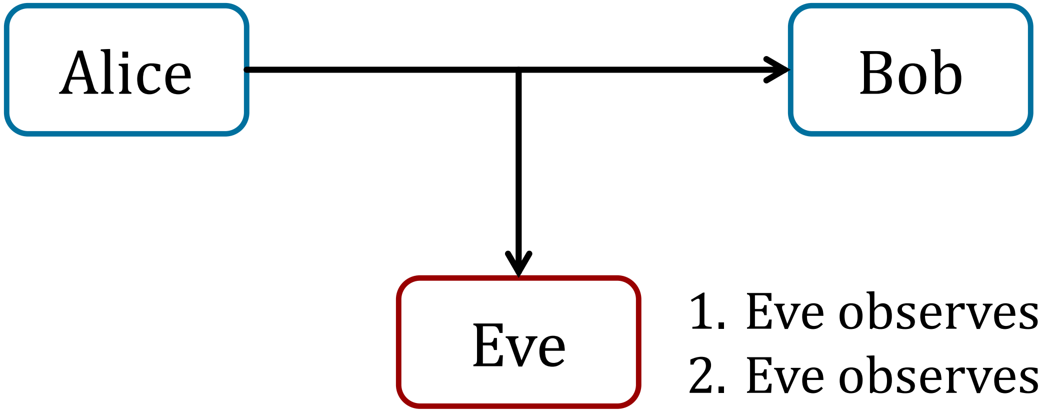
$D(c,k) = c \oplus k$

1. No
2. Yes, the key is $k = m \oplus c$
3. I can only compute half the bits
4. Yes, the key is $k = m \oplus m$

Perfect Secrecy [Shannon1945]
(Information Theoretic Secrecy)

__Defn Perfect Secrecy (informal):__

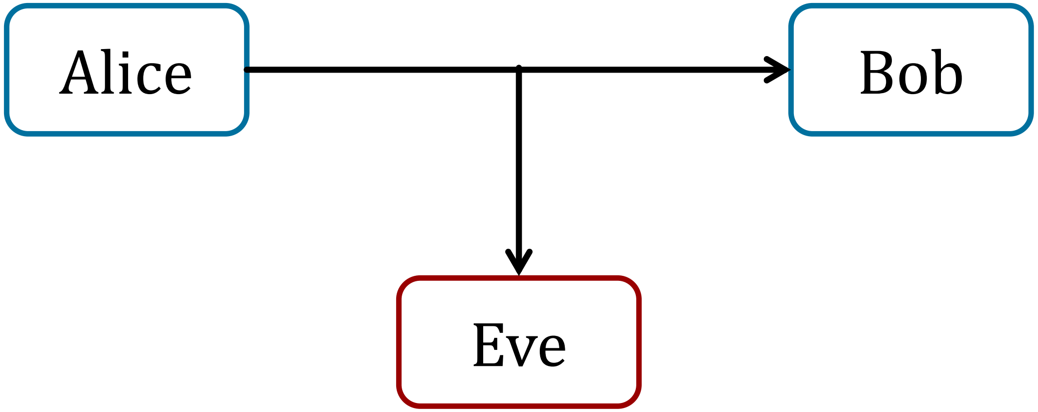
We're no better off determining the plaintext when given the ciphertext.



1. Eve observes everything but the c. Guesses m_1
2. Eve observes c. Guesses m_2

Goal: $\Pr[m = m_1] = \Pr[m = m_2]$

Example



Suppose there are 3 possible messages Alice may send:

- m_1: The attack is at 1pm. The probability of this message is 1/2
- m_2: The attack is at 2pm. The probability of this message is 1/4
- m_3: The attack is at 3pm. The probability of this message is 1/4

M	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$
Pr[M=m]	½	¼	1/4

Perfect Secrecy [Shannon1945]

(Information Theoretic Secrecy)

__Defn Perfect Secrecy (formal):__

$$\forall m_0, m_1 \in M. \text{ where } |m_0| = |m_1|$$

$$\forall c \in C.$$

$$\Pr\left[E(k, m_0) = c\right] = \Pr\left[E(k, m_1) = c\right]$$

Question

$$E(k, m) = k \oplus m = c$$

$$D(k, c) = k \oplus c = m$$

How many OTP keys map *m* to *c*?

1. 1
2. 2
3. Depends on m

Good News: OTP is Perfectly Secure

<u>Thm</u>: The One Time Pad is Perfectly Secure

Must show: $\Pr[E(k, m_0) = c] = \Pr[E(k, m_1) = c]$
where $|M| = \{0,1\}^m$

<u>Intuition:</u> Say that M = {00,01,10,11}, and *m* = 11. The adversary receives c = 10. It asks itself whether the plaintext was m_0 or m_1 (e.g., 01 or 10). It reasons:

- if m_0, then $k = m_0 \oplus c = 01 \oplus 10 = 11$.
- if m_1, then $k = m_1 \oplus c = 10 \oplus 10 = 00$.

But all keys are equally likely, so it doesn't know which case it could be.

Good News: OTP is Perfectly Secure

Thm: The One Time Pad is Perfectly Secure

Must show: $\Pr\left[E(k, m_0) = c\right] = \Pr\left[E(k, m_1) = c\right]$

where |M| = $\{0,1\}^m$

Proof:

$$\Pr[E(k, m_0) = c] = \Pr[k \oplus m_0 = c] \quad (1)$$
$$= \frac{|k \in \{0,1\}^m : k \oplus m_0 = c|}{\{0,1\}^m} \quad (2)$$
$$= \frac{1}{2^m} \quad (3)$$
$$\Pr[E(k, m_1) = c] = \Pr[k \oplus m_1 = c] \quad (4)$$
$$= \frac{|k \in \{0,1\}^m : k \oplus m_1 = c|}{\{0,1\}^m} \quad (5)$$
$$= \frac{1}{2^m} \quad (6)$$

Therefore, $\Pr[E(k, m_0) = c] = \Pr[E(k, m_1) = c]$

Two Time Pad is Insecure

Two Time Pad:

$c_1 = m_1 \oplus k$

$c_2 = m_2 \oplus k$

Enough redundancy in ASCII (and english) that $m_1 \oplus m_2$ is enough to know m_1 and m_2

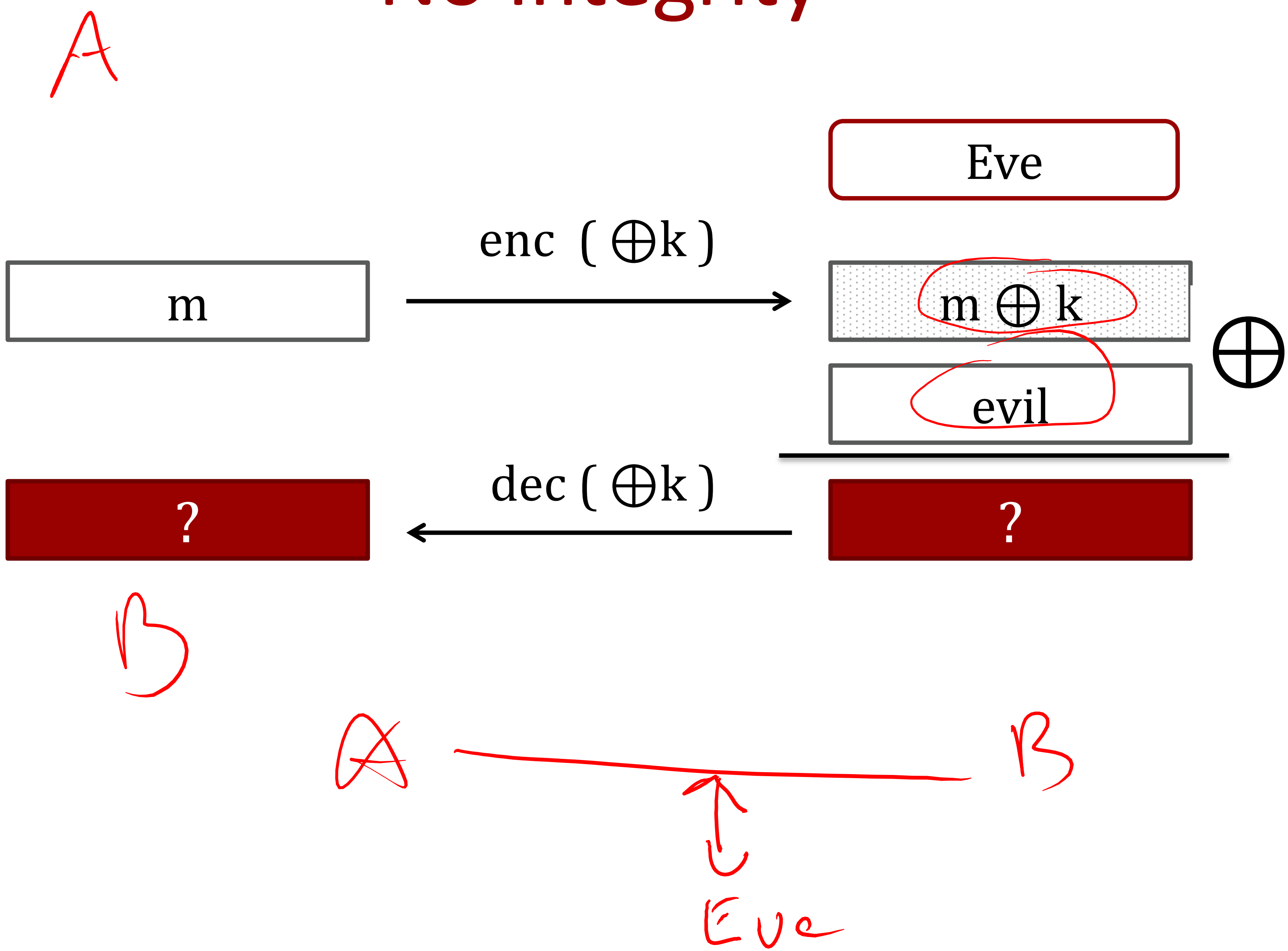
Eavesdropper gets c_1 and c_2.
What is the problem?

$c_1 \oplus c_2 = m_1 \oplus m_2$

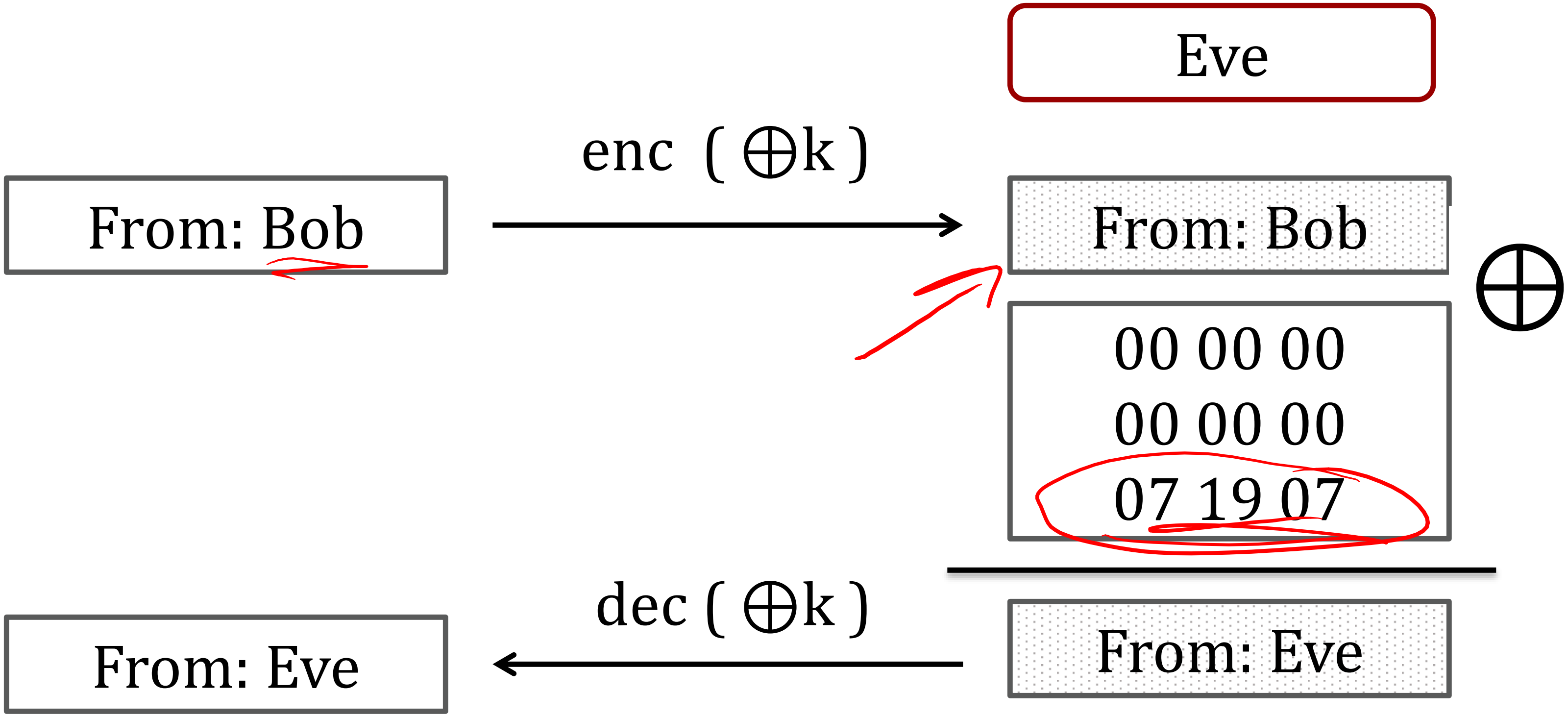
The OTP provides perfect secrecy.

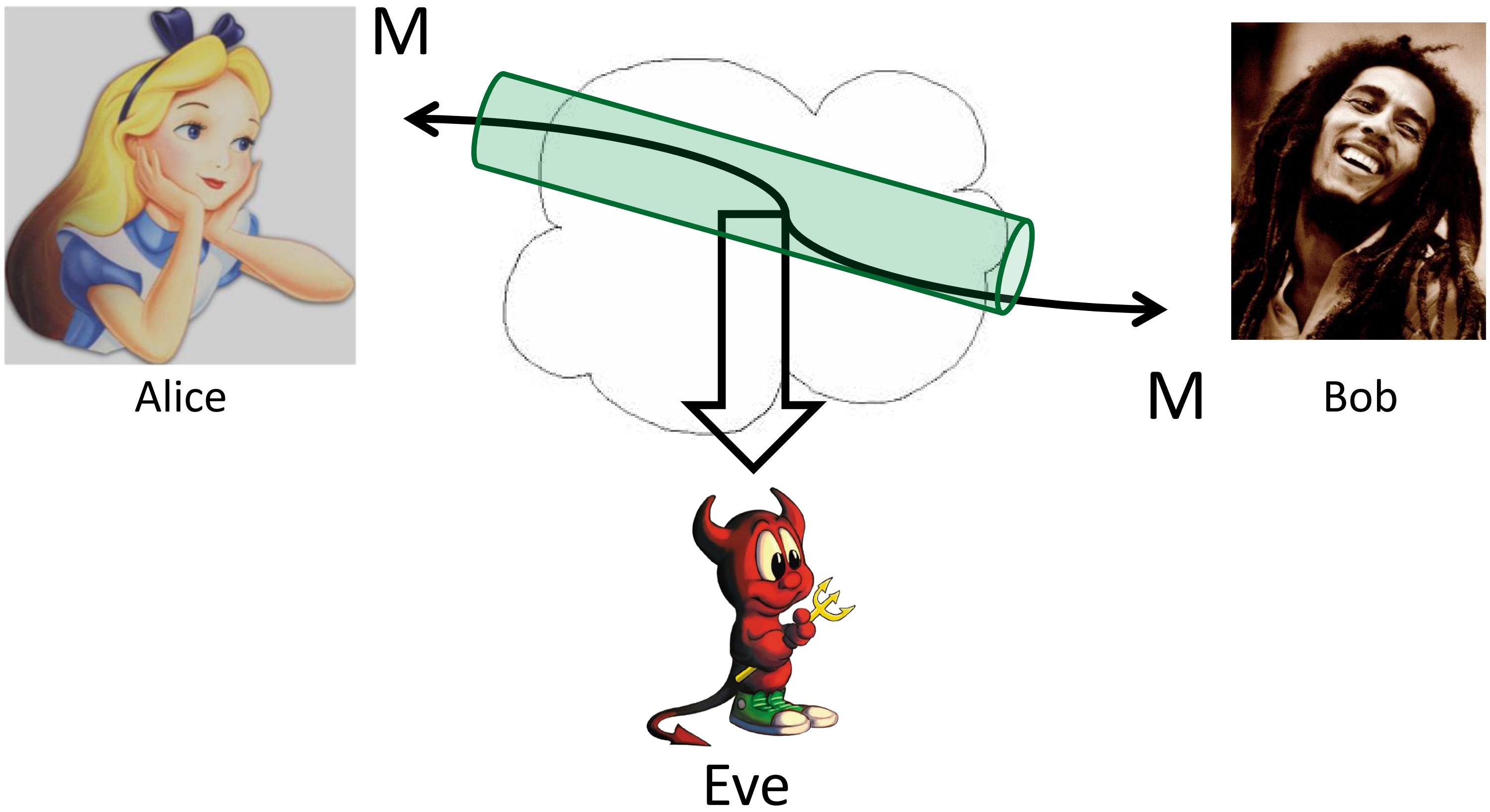
……But is that enough?

No Integrity



No Integrity

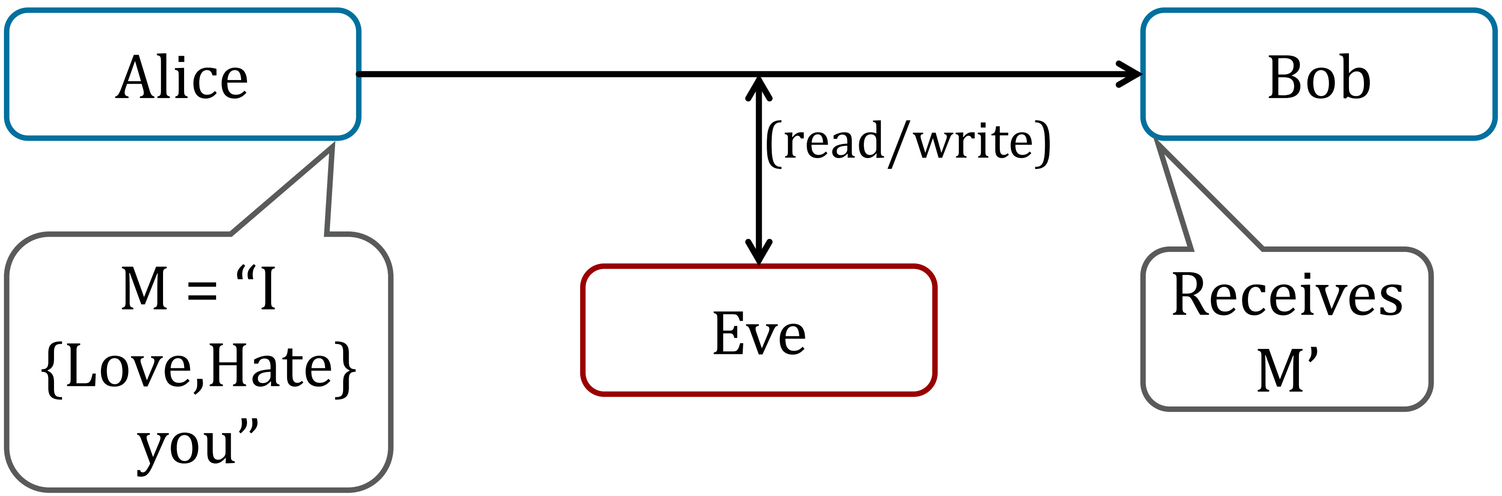




Goal 2: *Integrity*

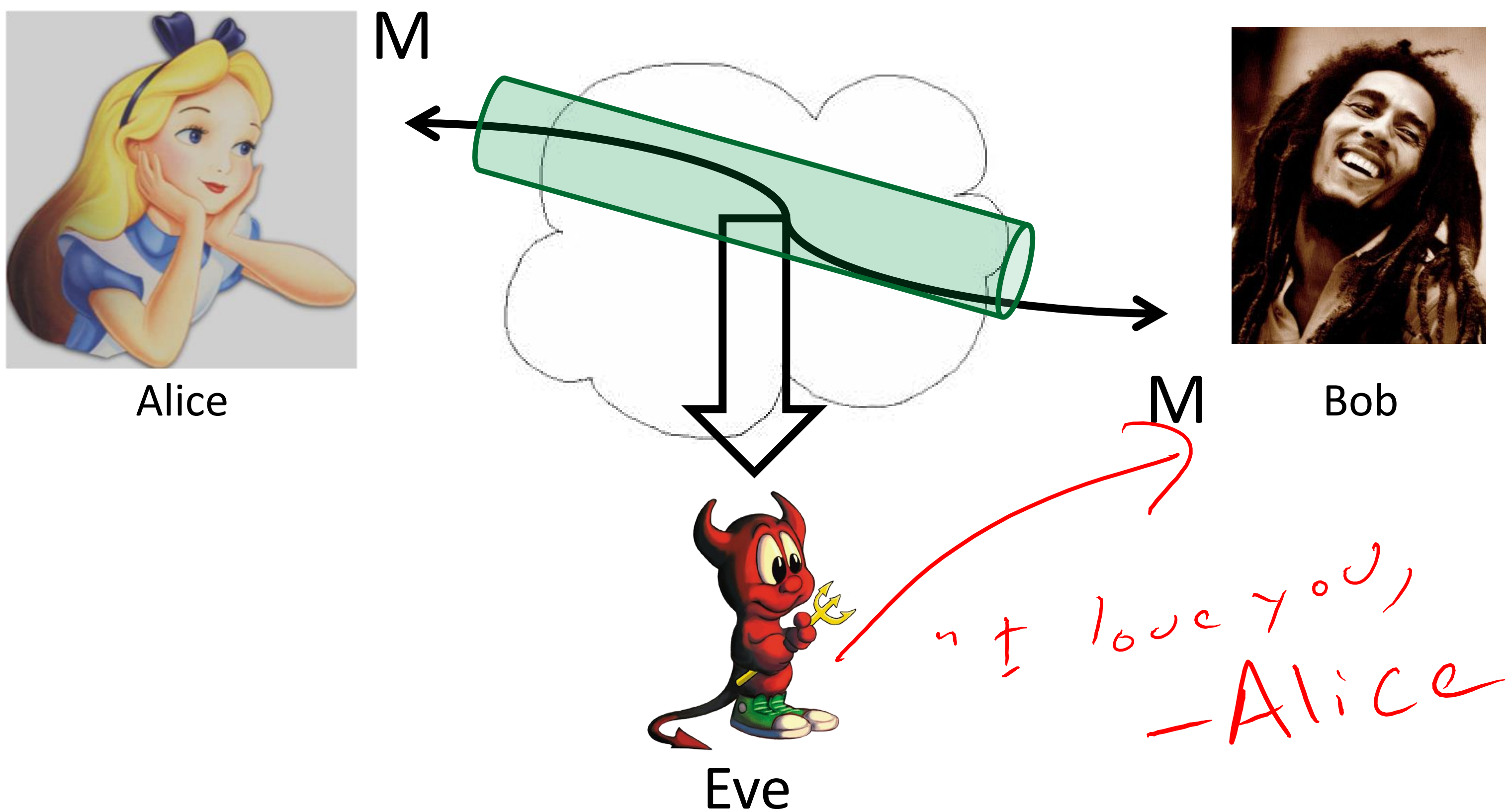
Eve should not be able to alter M.

Detecting Flipped Bits



Bob should be able to determine if M=M'

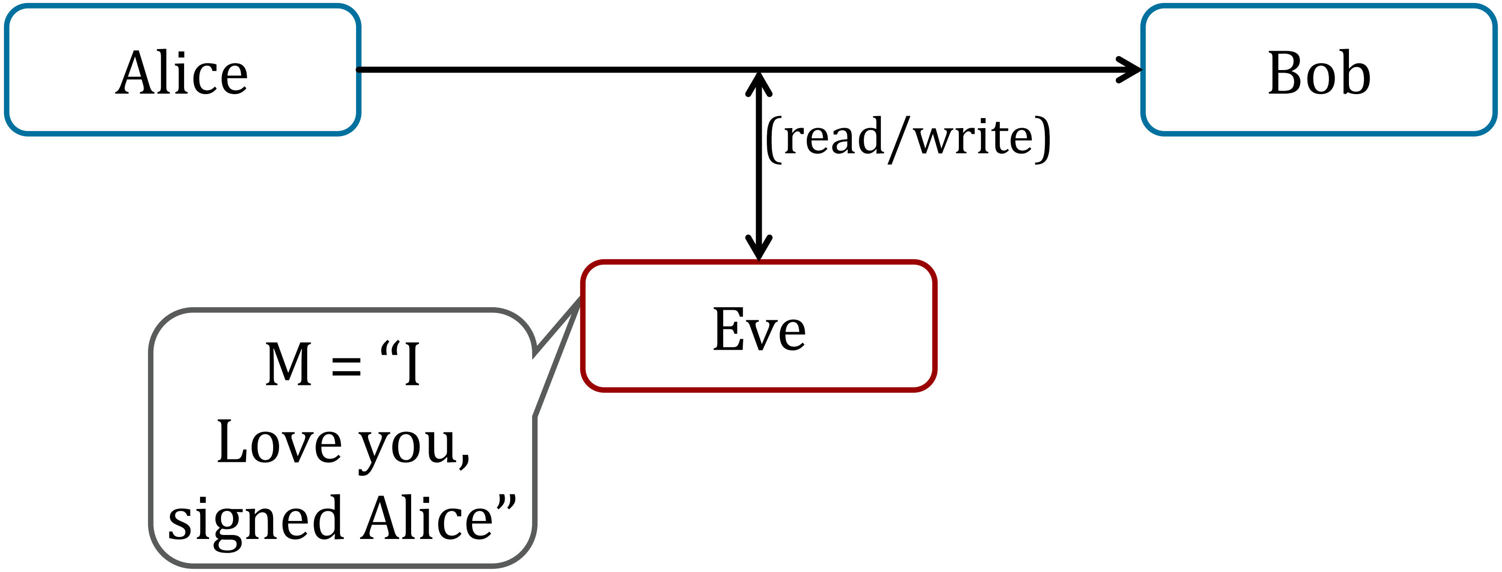
<u>Ex:</u> Eve should not be able to flip Alice's message without detection (even when Eve doesn't know content of M)



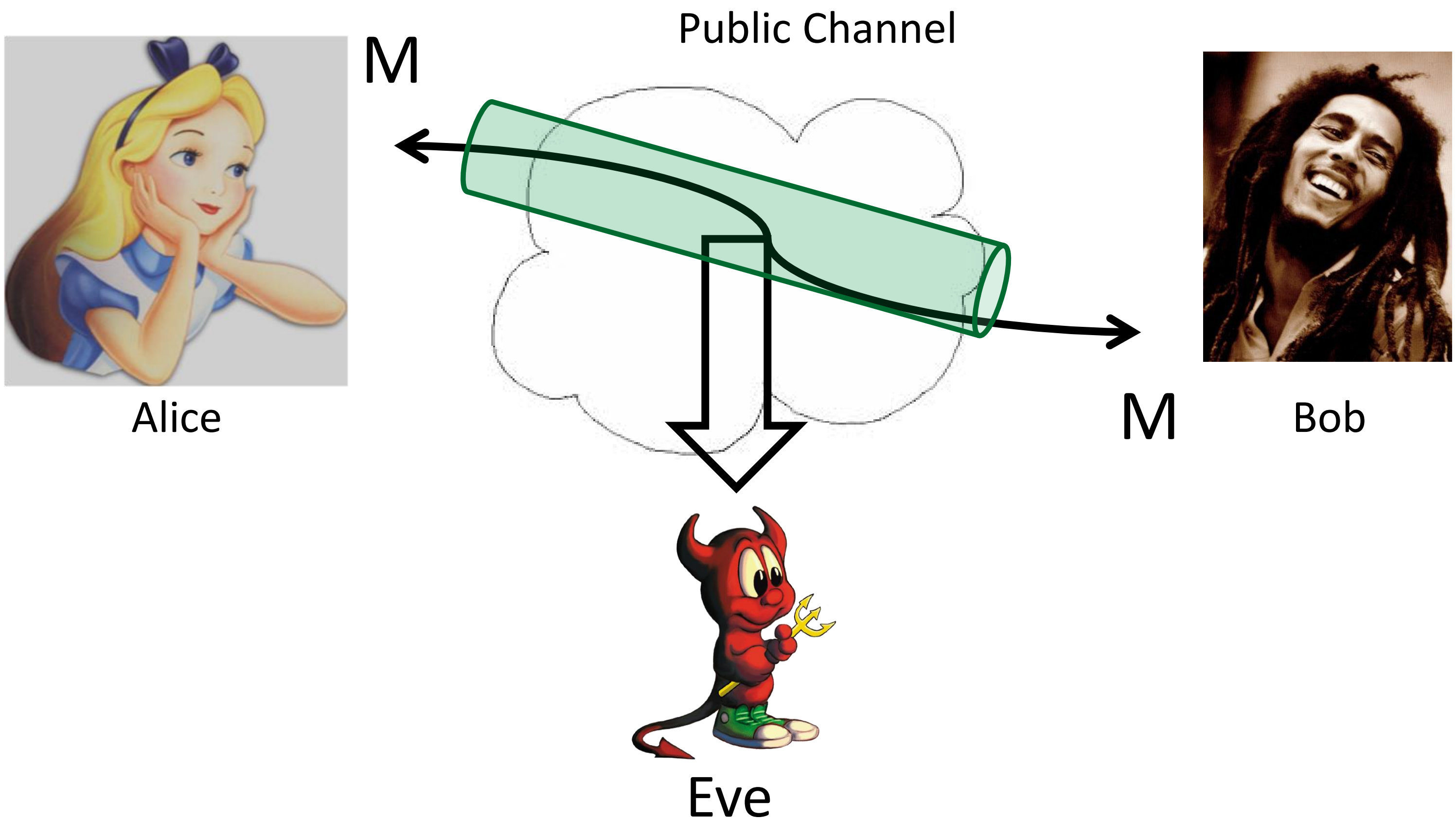
Goal 3: *Authenticity*

Eve should not be able to forge messages as Alice

Detecting Flipped Bits



Bob should be able to determine M wasn't sent from Alice

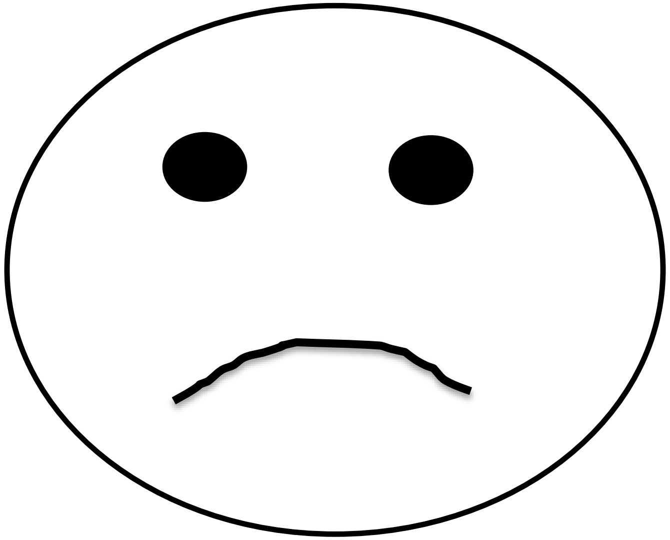


Cryptonium Pipe Goals:
Privacy, Integrity, and Authenticity

Modern Notions: Indistinguishability and Semantic Security

The "Bad News" Theorem

<u>Theorem</u>: Perfect secrecy requires |K| >= |M|



In practice, we usually shoot for *<u>computational security</u>*.

What is a secure cipher?

Attackers goal: recover one plaintext (for now)

Attempt #1: Attacker cannot recover key

Insufficient: $E(k,m) = m$

Attempt #2: Attacker cannot recover all of plaintext

Insufficient: $E(k,m_0 \mid\mid m_1) = m_0 \mid\mid E(k,m_1)$

Recall Shannon's Intuition:
c should reveal no information about *m*

Adversarial Indistinguishability Game

E

Challenger:
I have a
secure cipher E

A

I am *any*
adversary. I can
break your crypto.

Semantic Security Motivation

E ***A***

m_0,m_1 ←

1. ***A*** sends m_0, m_1 s.t. $|m_0|=|m_1|$ to the challenger

2. Challenger encrypts one at random. Sends back c.

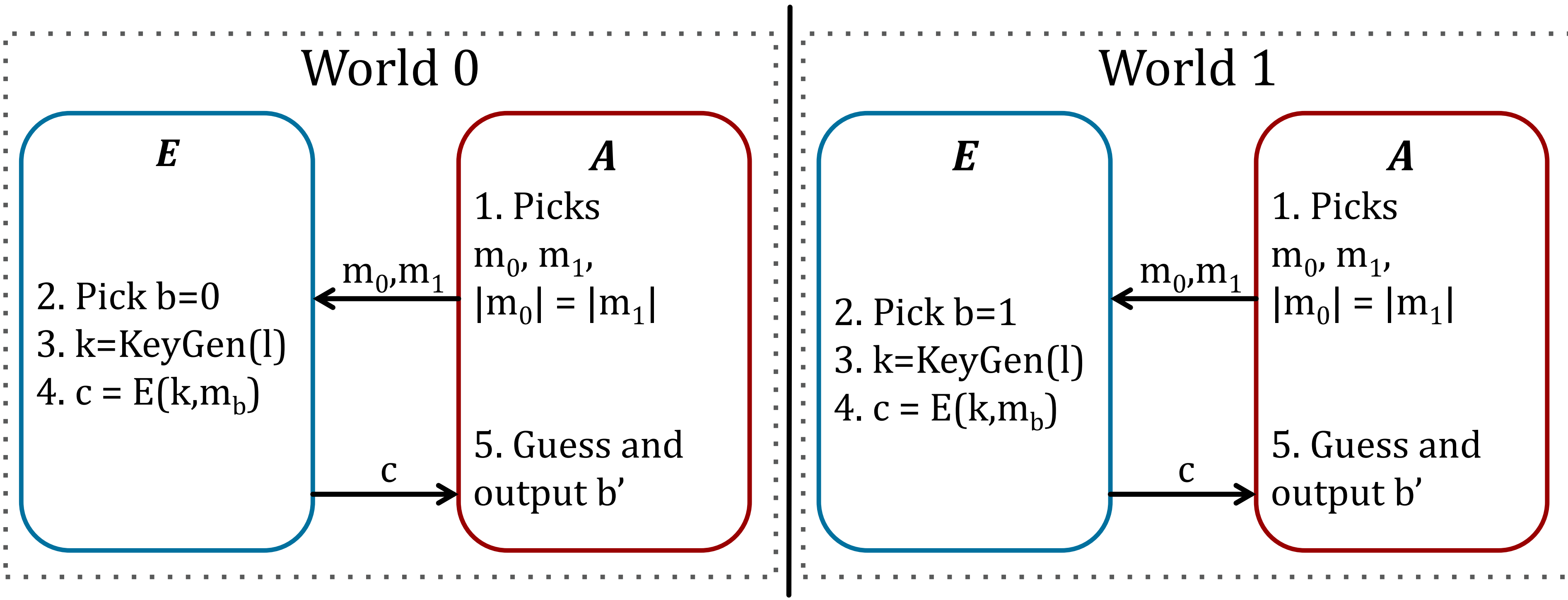
c →

3. ***A*** tries to guess which message was encrypted.

4. Challenger wins of ***A*** is no better than guessing

Semantically secure

Semantic Security Game



A doesn't know which world he is in, but wants to figure it out.

Semantic security is a behavioral model getting at any ***A*** behaving the same in either world when ***E*** is secure.

Semantic Security Game

(A ***behavioral*** model)

World 0

E

2. Pick b=0
3. k=KeyGen(l)
4. c = E(k,m_b)

A

1. Picks m_0, m_1, $|m_0| = |m_1|$

$\xleftarrow{m_0, m_1}$

$\xrightarrow{c}$

5. Guess and output b'

World 1

E

2. Pick b=1
3. k=KeyGen(l)
4. c = E(k,m_b)

A

1. Picks m_0, m_1, $|m_0| = |m_1|$

$\xleftarrow{m_0, m_1}$

$\xrightarrow{c}$

5. Guess and output b'

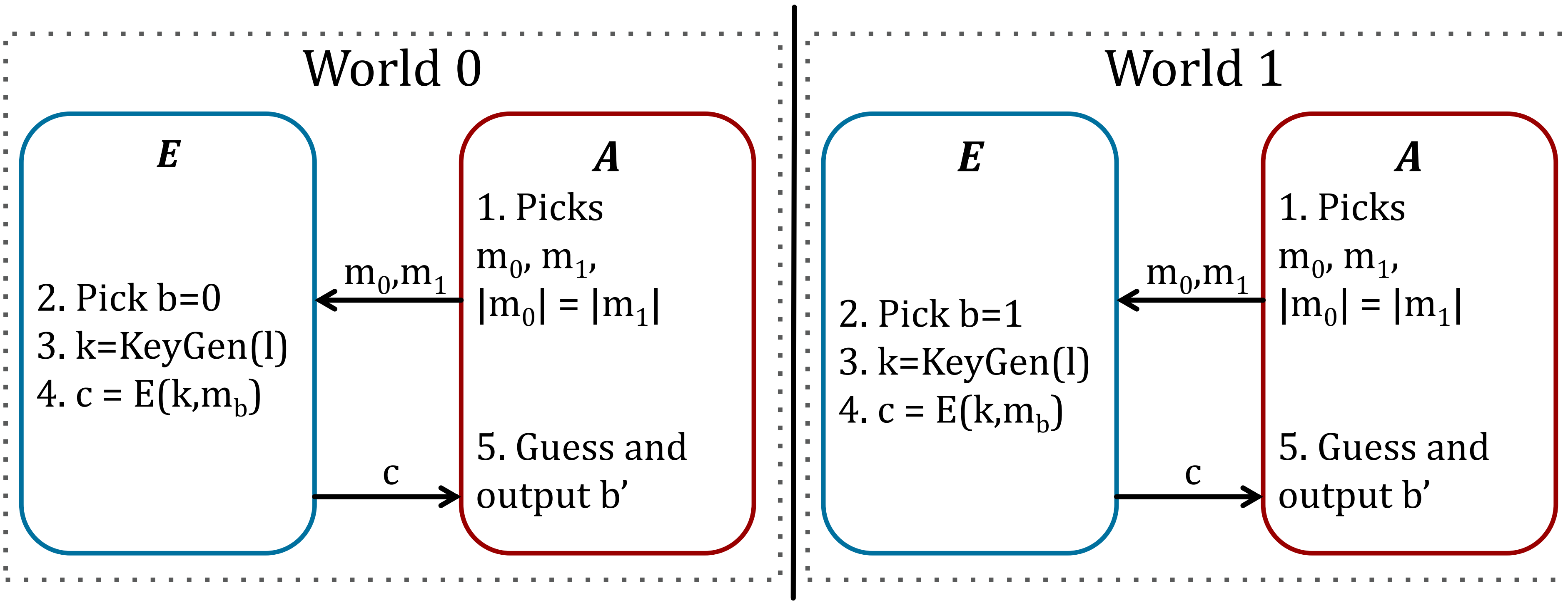
A doesn't know which world he is in, but wants to figure it out.

For b=0,1: W_b := [event that $\boldsymbol{A}(W_b)$ =1]

$\mathrm{Adv}_{SS}[\boldsymbol{A},\boldsymbol{E}]$:= | Pr[W_0] − Pr[W_1] | $\in [0,1]$

Always 1

Example 1: Guessing

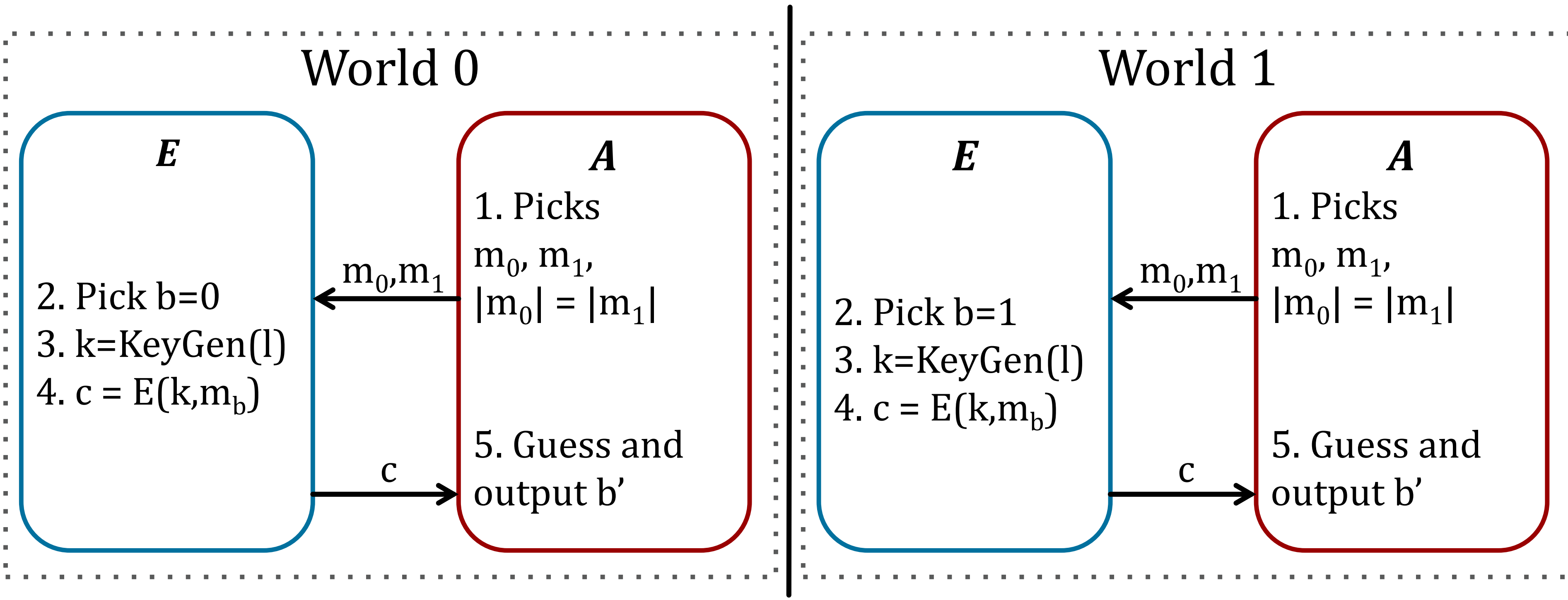


A guesses. W_b := [event that ***A***(W_b) =1]. So

W_0 = .5, and W_1 = .5

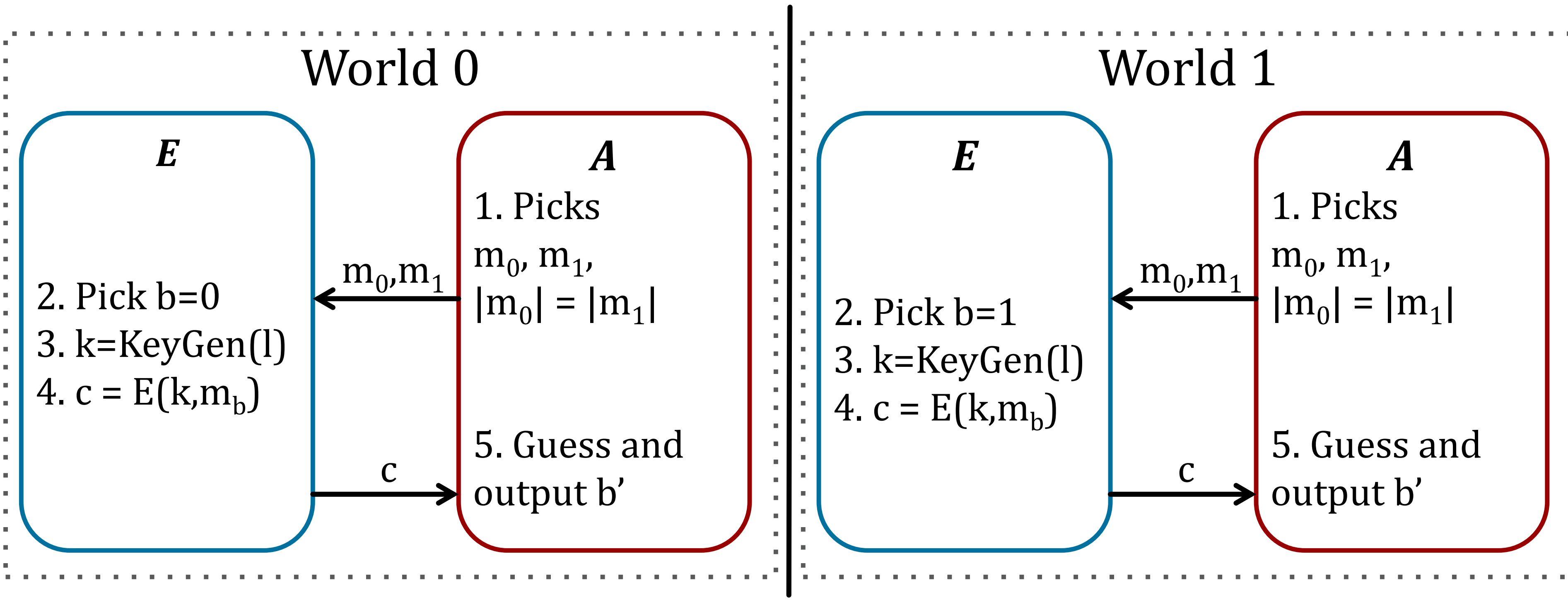
$\text{Adv}_{SS}[\boldsymbol{A},\boldsymbol{E}]$:= | .5 – .5 | = 0

Example 1: ***A*** is right 75% of time



A guesses. W_b := [event that ***A***(W_b) =1]. So
W_0 = .25, and W_1 = .75
$\mathrm{Adv}_{SS}[\boldsymbol{A},\boldsymbol{E}]$:= | .25 – .75 | = .5

Example 1: ***A*** is right 25% of time



A guesses. W_b := [event that ***A***(W_b) =1]. So

W_0 = .75, and W_1 = .25

$\mathrm{Adv}_{SS}[\boldsymbol{A},\boldsymbol{E}]$:= | .75 – .25 | = .5

Note for W_0, ***A*** is wrong more often than right. ***A*** should switch guesses.

Semantic Security

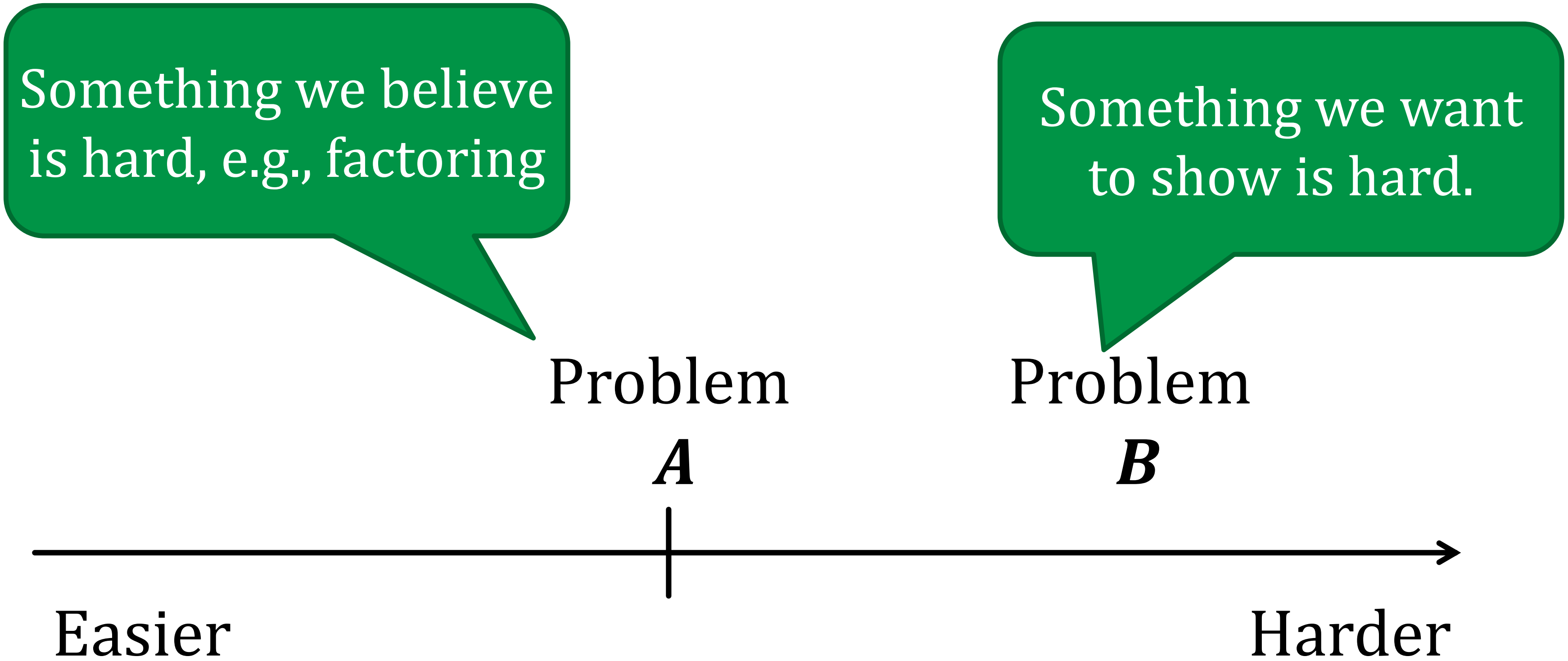
Given:
For b=0,1: W_b := [event that $\boldsymbol{A}(W_b)$ =1]
$\text{Adv}_{SS}[\boldsymbol{A},\boldsymbol{E}]$:= | Pr[W_0] − Pr[W_1] | $\in [0,1]$

Defn:
E is *semantically secure* if for all efficient ***A***:

$\text{Adv}_{SS}[\boldsymbol{A}, \boldsymbol{E}]$ is negligible.

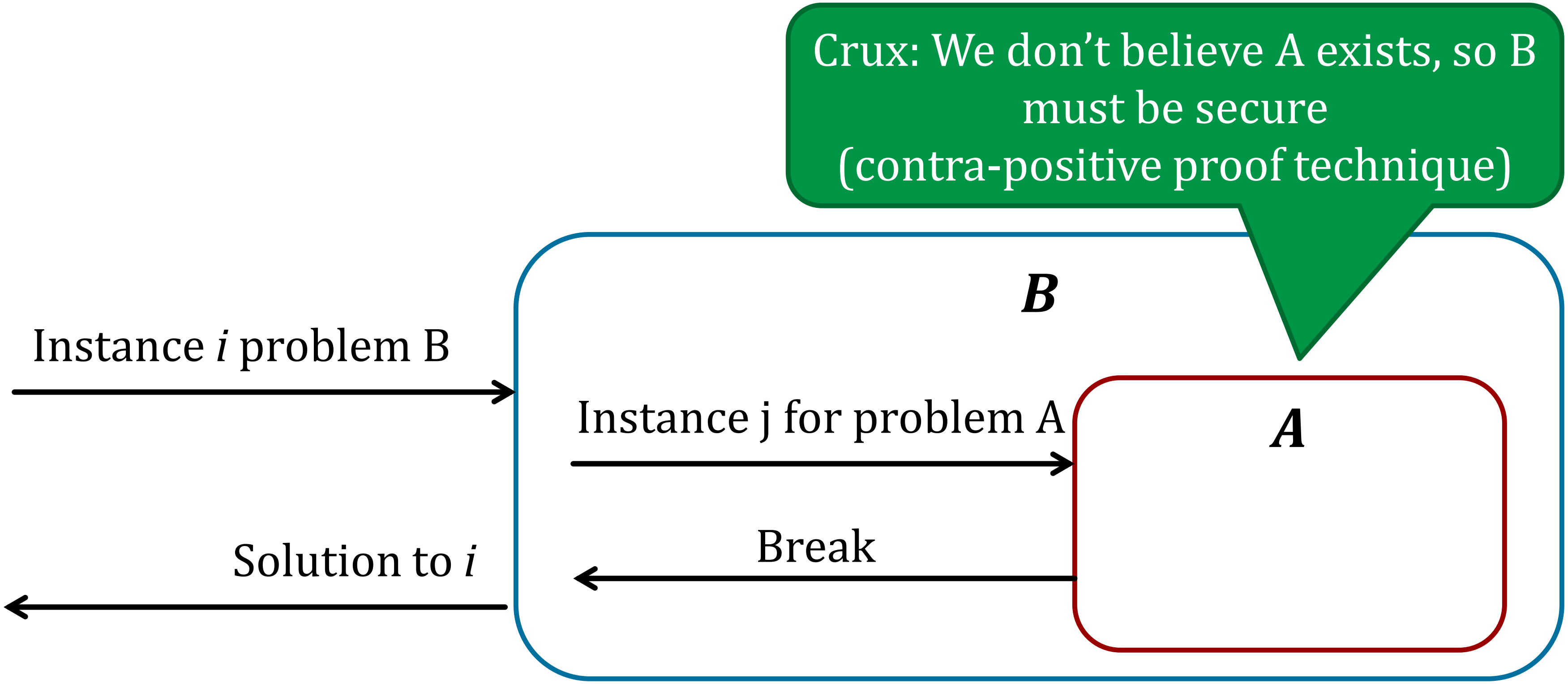
⇒ for all explicit m_0 , $m_1 \in M$:
$\{ E(k,m_0) \} \approx_p \{ E(k,m_1) \}$

This is what it means to be secure against eavesdroppers. No partial information is leaked


Something we believe is hard, e.g., factoring
Something we want to show is hard.
Problem A
Problem B
Easier
Harder

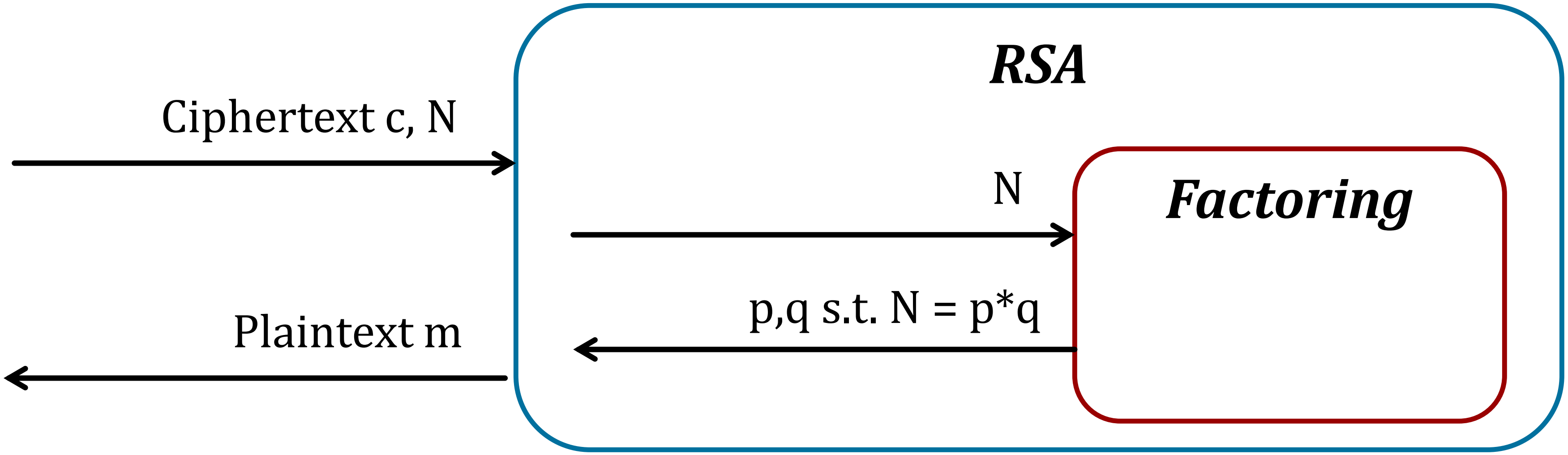
Security Reductions

<u>*Reduction*</u>: Problem $\boldsymbol{A}$ is at least as hard as $\boldsymbol{B}$ if an algorithm for solving $\boldsymbol{A}$ efficiently (if it existed) could also be used as a subroutine to solve problem $\boldsymbol{B}$ efficiently.



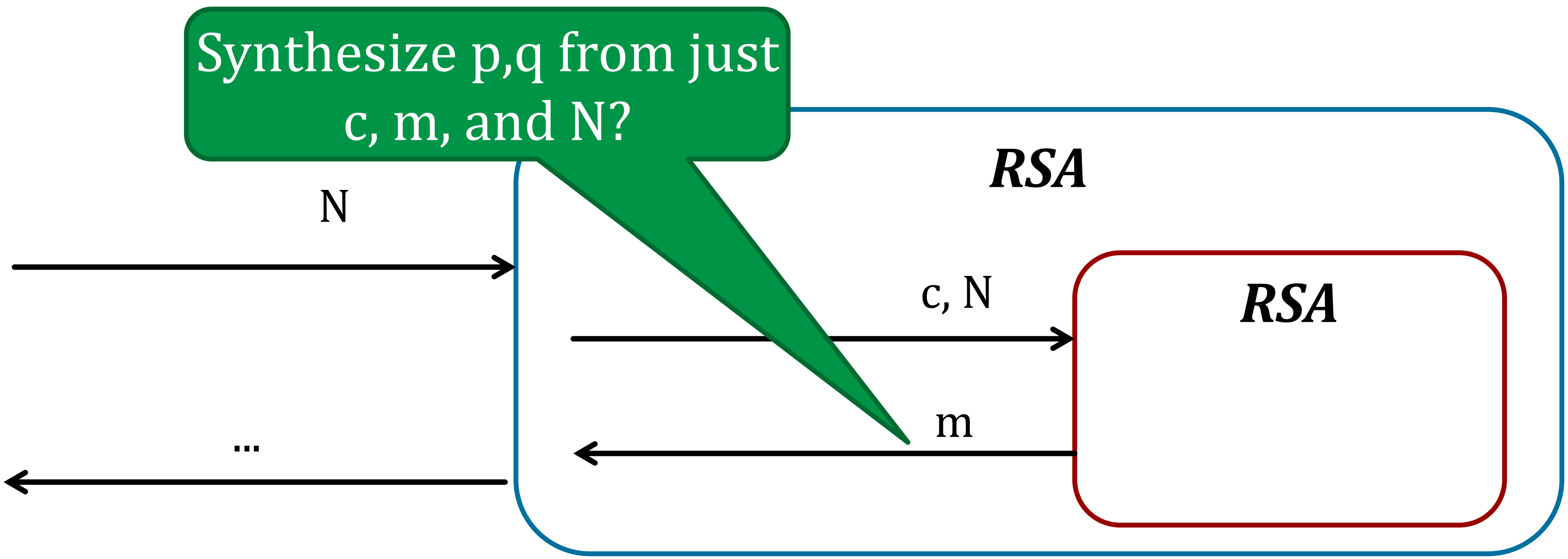
Example

<u>*Reduction*</u>: Problem ***Factoring (A)*** is at least as hard as ***RSA (B)*** if an algorithm for solving ***Factoring (A)*** efficiently (if it existed) could also be used as a subroutine to solve problem ***RSA (B)*** efficiently.

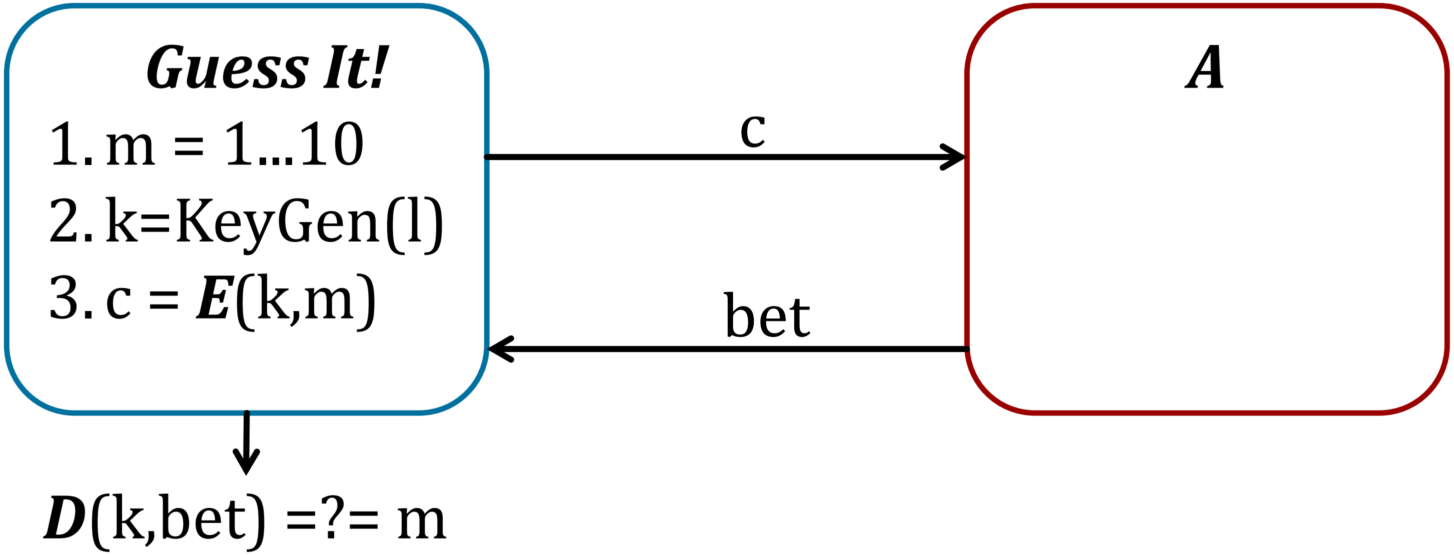


What's *unknown*...

Reduction: Problem ***RSA (A)*** is at least as hard as ***Factoring (B)*** if an algorithm for solving ***RSA (A)*** efficiently (if it existed) could also be used as a subroutine to solve problem ***Factoring (B)*** efficiently.



Games and Reductions

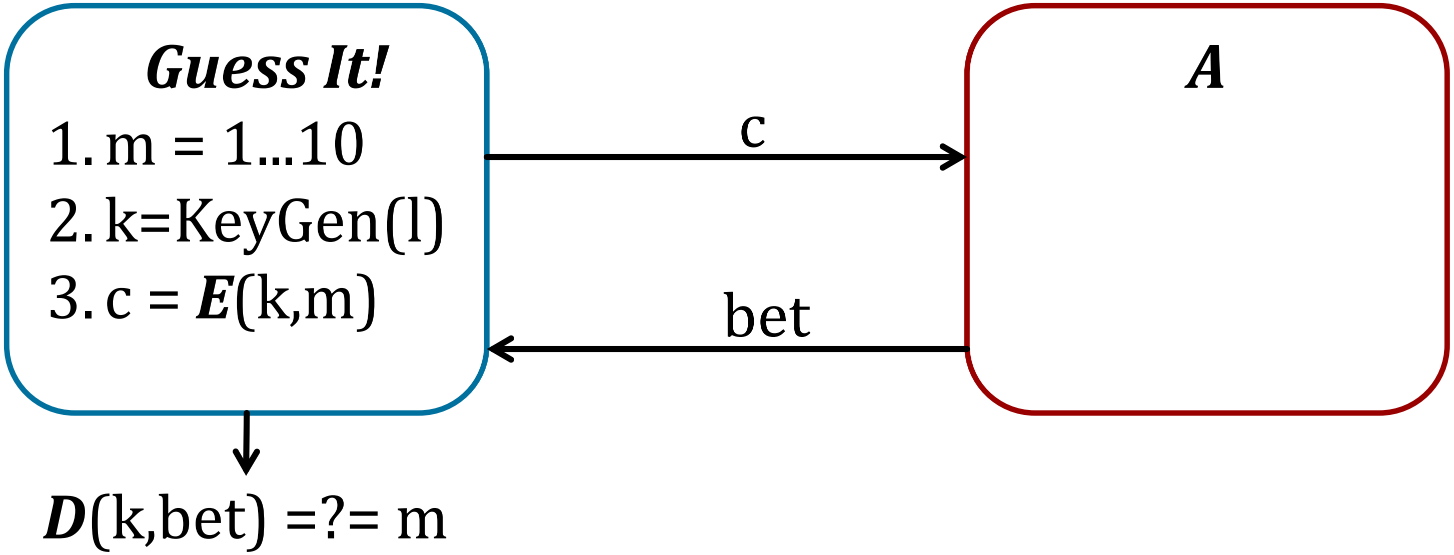


Suppose ***A*** is in a guessing game. Guess It! uses ***E*** to encrypt. How can we prove, in this setting, that ***E*** is secure?

Reduction: If ***A*** does better than 1/10, we break ***E*** in the semantic security game. Showing security of ***E*** reduces to showing if ***A*** exists, it could break the semantic security game.

Note: The "type" of A is A: c -> bet, not that of the game.

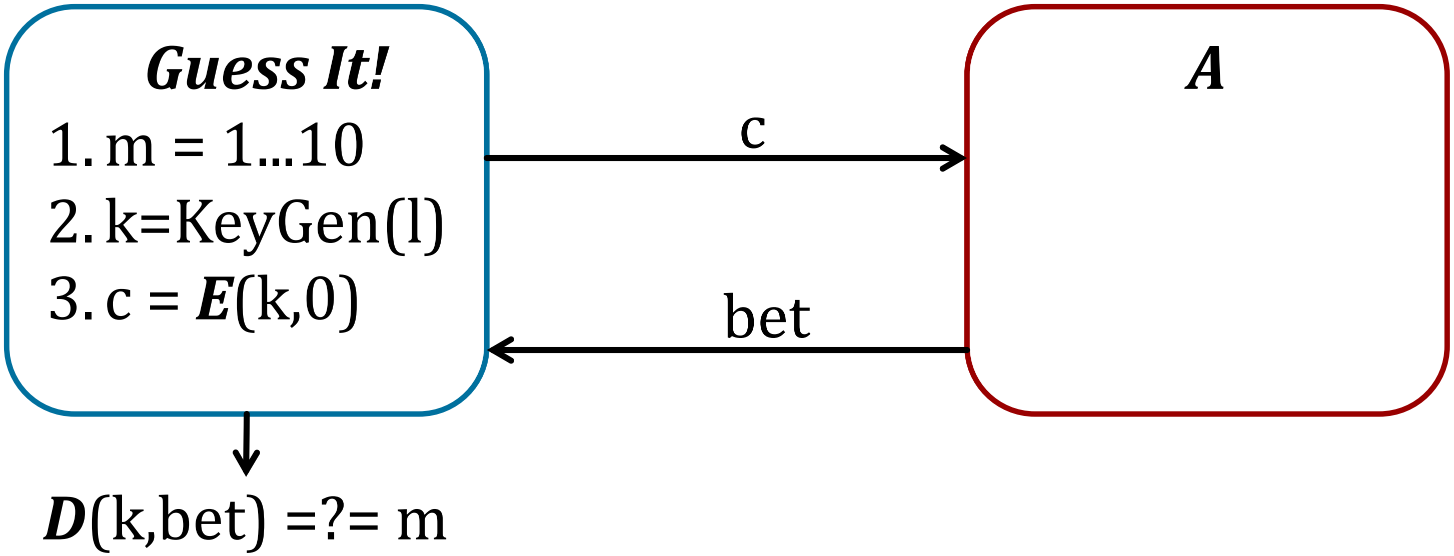
The Real Version



In the *<u>real</u>* version, ***A*** always gets an encryption of the real message.

– Pr[A wins in *<u>real</u>* version] = p_0

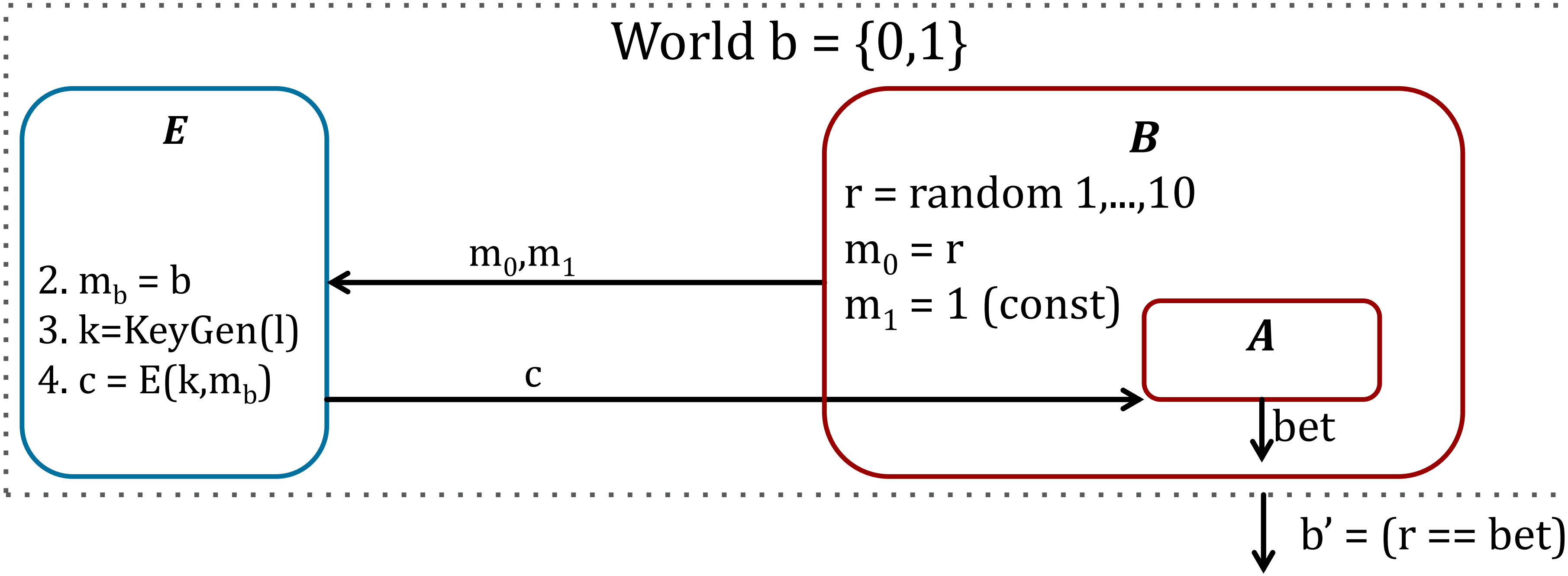
Idealized Version



In the ideal version, ***A*** always gets an encryption of a constant, say 1. (A still only wins if it gets *m* correct.)

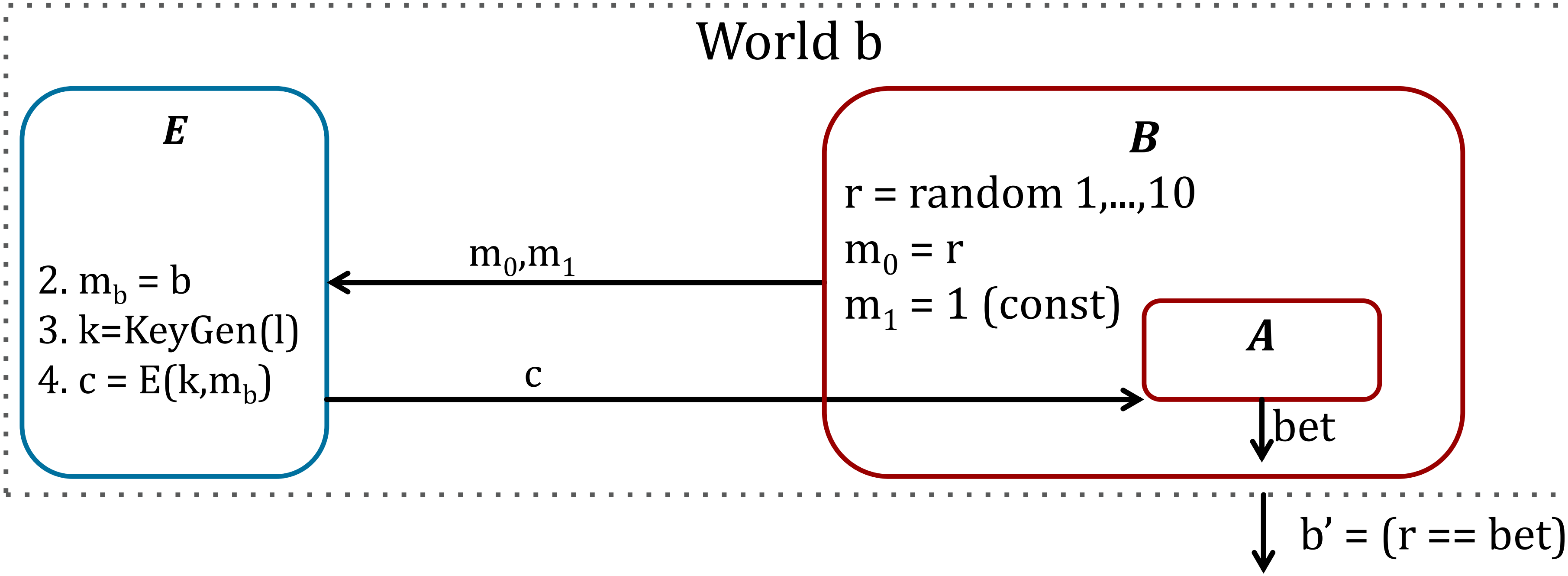
– Pr[A wins in Idealized Version] = p_1 = 1/10

Reduction



- If B is in world 0, then Pr[b' = 1] = p_0
  - B can guess r==bet with prob. p_0.
- If B is in world 1, then Pr[b' = 1] = $p_1 = 1/10$
- For b=0,1: W_b := [event that $\boldsymbol{B}(W_b) = 1$]
  $Adv_{SS}[\boldsymbol{A},\boldsymbol{E}] = |\Pr[W_0] - \Pr[W_1]|$
  $= |p_0 - p_1|$

Reduction



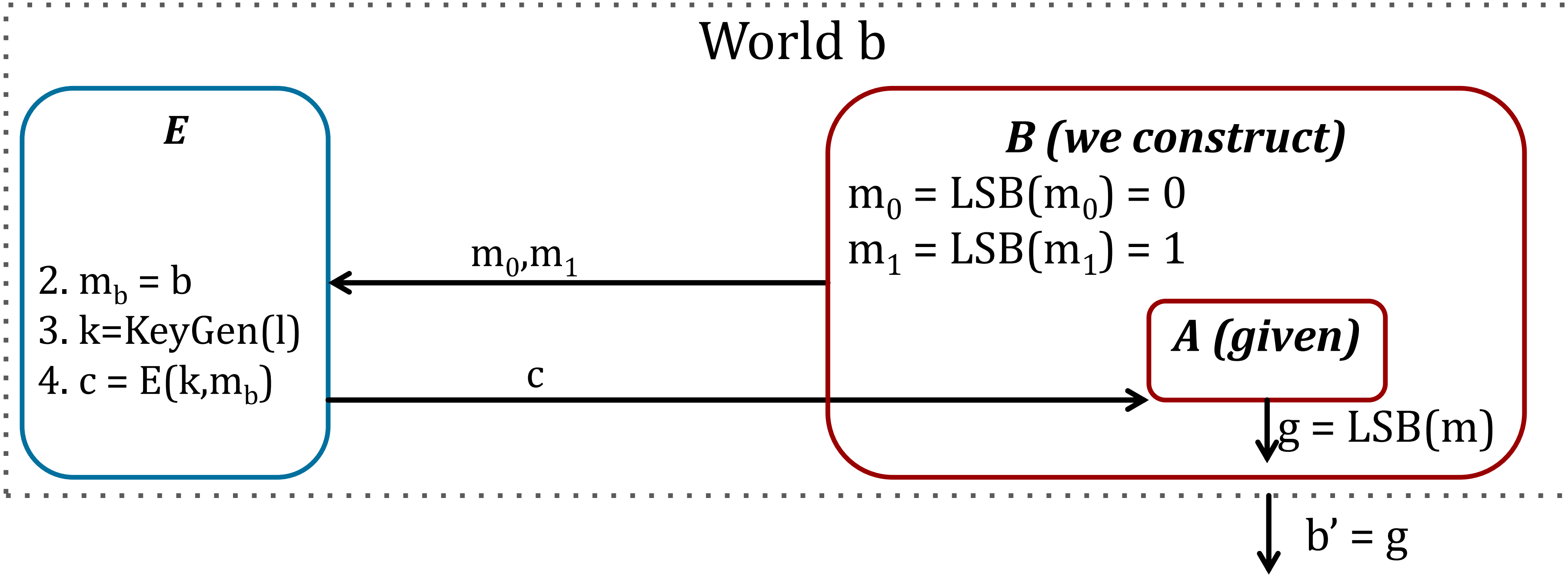
- If B is in world 0, then Pr[b' = 1] = p_0
  - B can guess r==bet with prob. p_0.
- If B is in world 1, then Pr[b' = 1] = p_1 = 1/10
- For b=0,1: W_b := [event that ***B***(W_b) =1]
  $\text{Adv}_{SS}[\boldsymbol{A},\boldsymbol{E}]$ = | Pr[W_0] – Pr[W_1] |
  = $|p_0 - p_1|$

Suppose 33% correct

33%-%10 = 23% Advantage

Reduction Example 2

Suppose efficient A can always deduce LSB of PT from CT.
Then $\mathbb{E} = (E,D)$ is not semantically secure.



$$\mathrm{Adv}_{SS}[\boldsymbol{A},\boldsymbol{E}] = |\ \Pr[\ W_0\ ] - \Pr[\ W_1\ ]\ | = |0 - 1| = 1$$

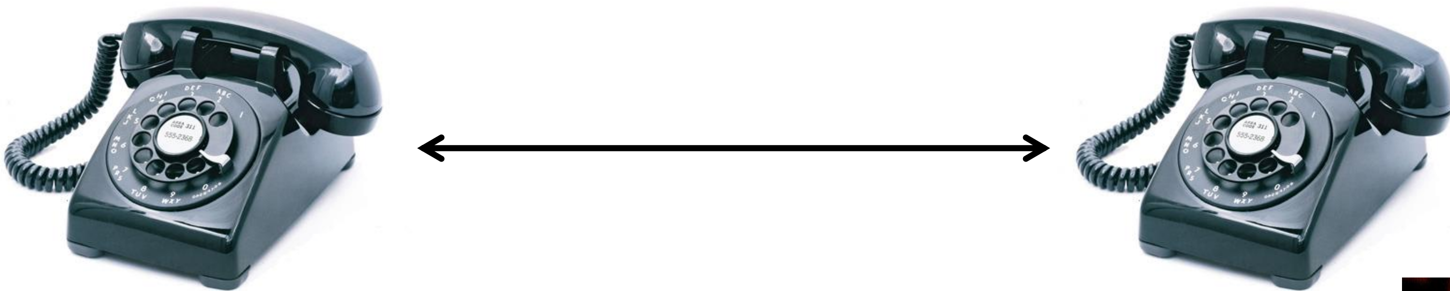
Summary

- Cryptography is a awesome tool
  - But not a complete solution to security
  - Authenticity, Integrity, Secrecy
- Perfect secrecy and OTP
  - Good news and Bad News
- Semantic Security
- Reductions

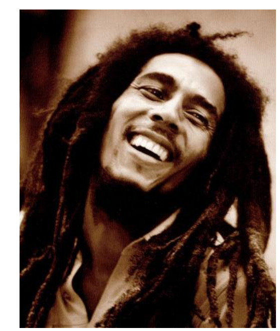

Questions?

END

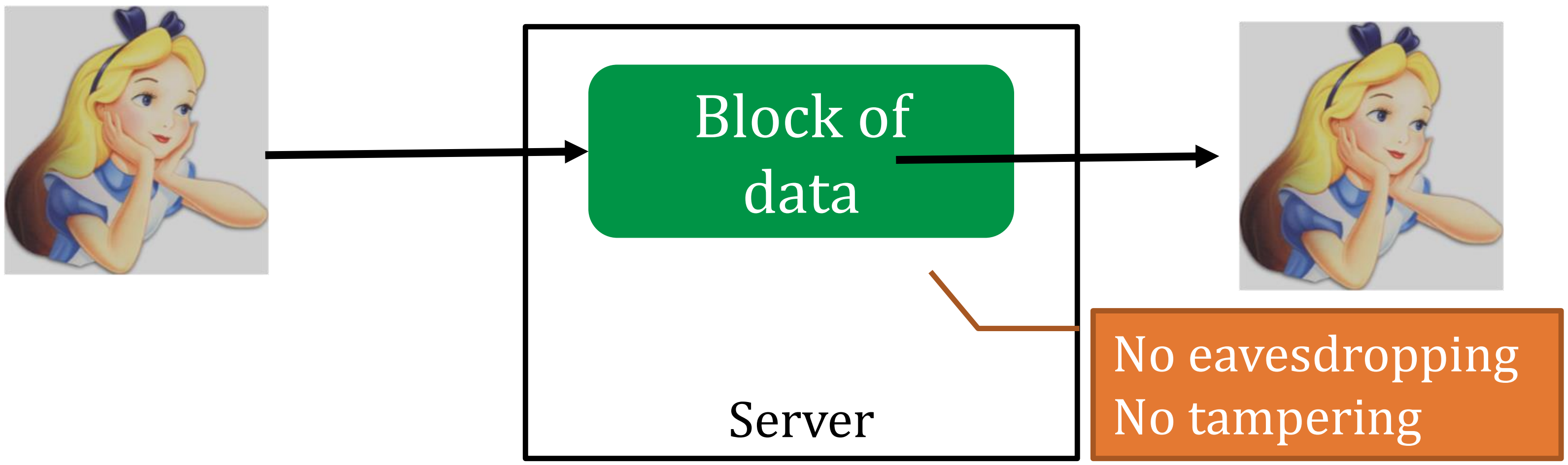
Stream Ciphers



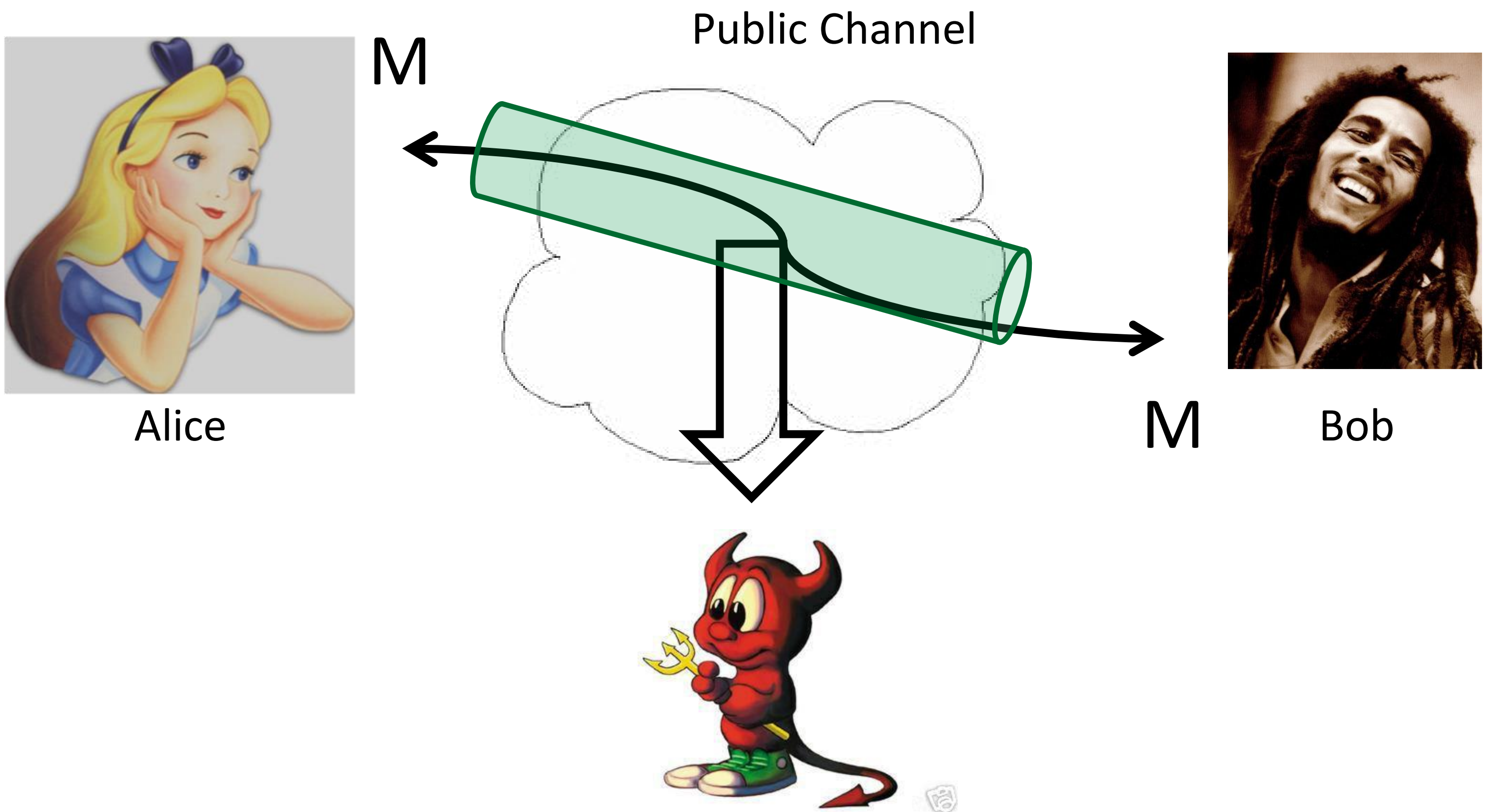
Continuous stream of data



Block Ciphers



Analogous to secure communication:
Alice today sends a message to Alice tomorrow



Cryptonium Pipe Goals:
Privacy, Integrity, and Authenticity

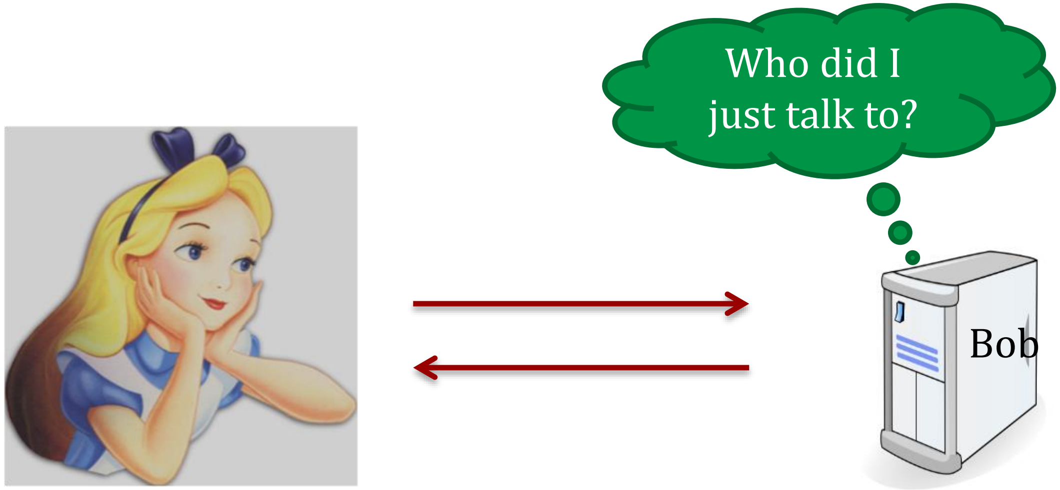
But crypto can do much more

- Digital signatures

But crypto can do much more

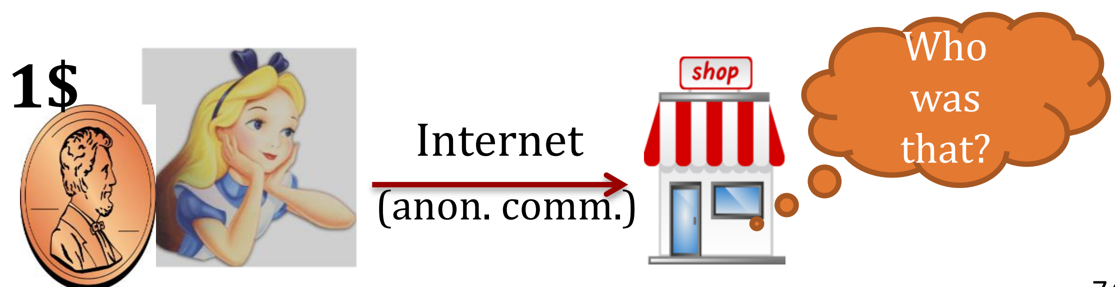
- Digital signatures

- Anonymous communication

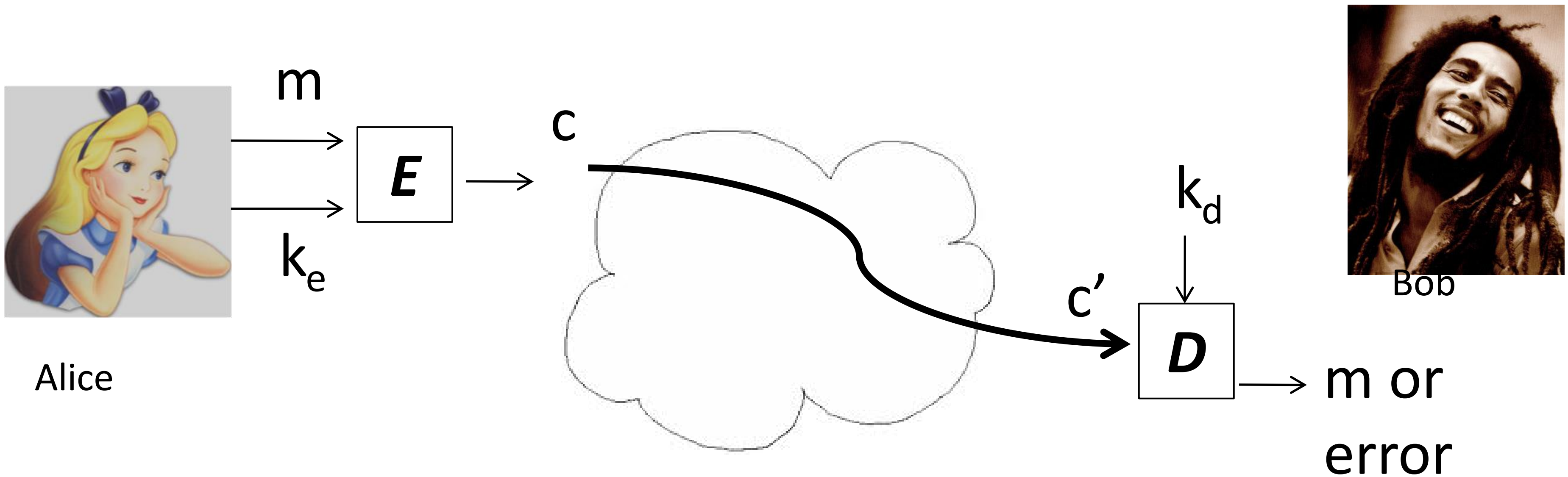


But crypto can do much more

- Digital signatures
- Anonymous communication
- Anonymous **digital** cash
  - Can I spend a "digital coin" without anyone knowing who I am?
  - How to prevent double spending?



Cryptosystem



E: Encryption Algorithm

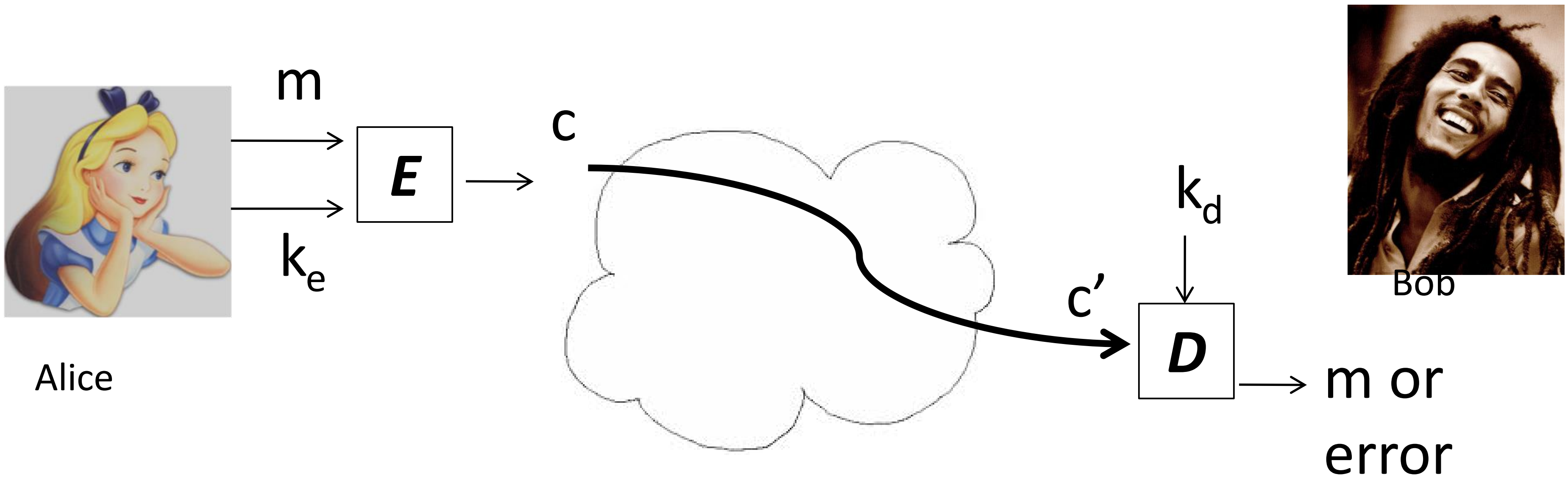
D: Decryption Algorithm

$\boldsymbol{k_e}$: Encryption Key

$\boldsymbol{k_d}$: Decryption Key

Algorithms: Standardized and Public

Cryptosystem



E: Encryption Algorithm

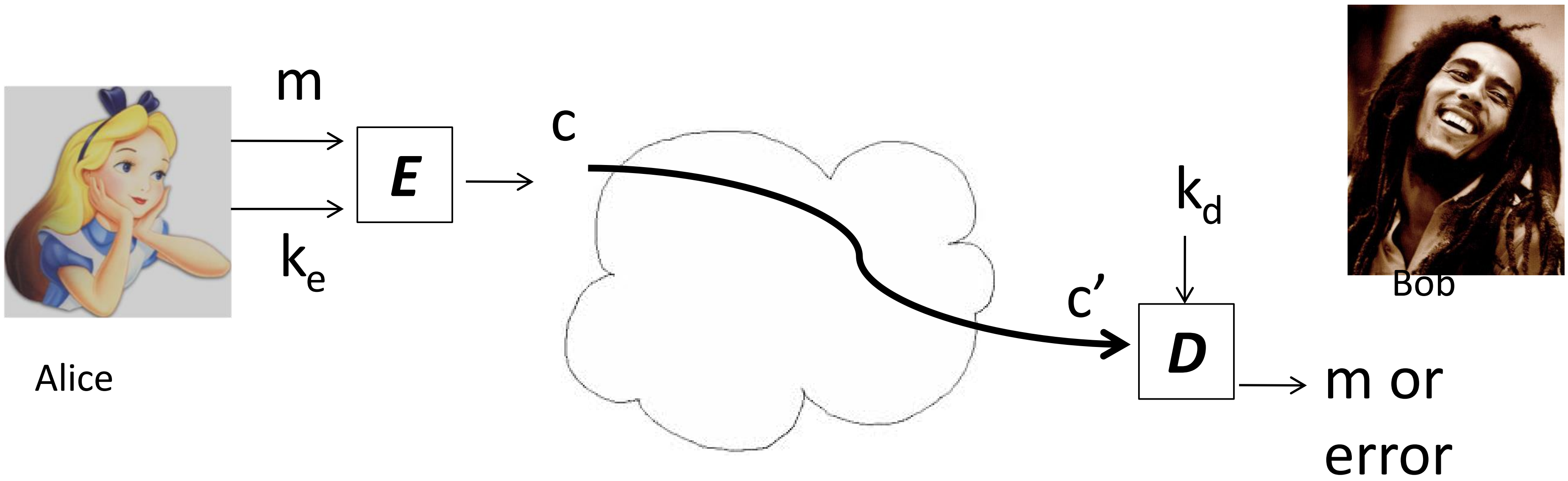
D: Decryption Algorithm

$\boldsymbol{k_e}$: Encryption Key

$\boldsymbol{k_d}$: Decryption Key

Private. Length of key determines security

Symmetric and Asymmetric Cryptosystem



E: Encryption Algorithm $\quad$ $\boldsymbol{k_e}$: Encryption Key

D: Decryption Algorithm $\quad$ $\boldsymbol{k_d}$: Decryption Key

Symmetric (shared key) : $k_e = k_d$
Asymmetric (public key) : k_e public, k_d private

Quiz

- What were the three properties crypto tries to achieve?

1. Privacy
2. Integrity
3. Authenticity

A rigorous science

The three steps in cryptography:

1. Precisely specify threat model
2. Propose a construction
3. Prove that breaking construction under threat mode will solve an underlying hard problem

Mathematical properties in terms of security parameter

A rigorous science

The three steps in cryptography:

1. Precisely specify threat

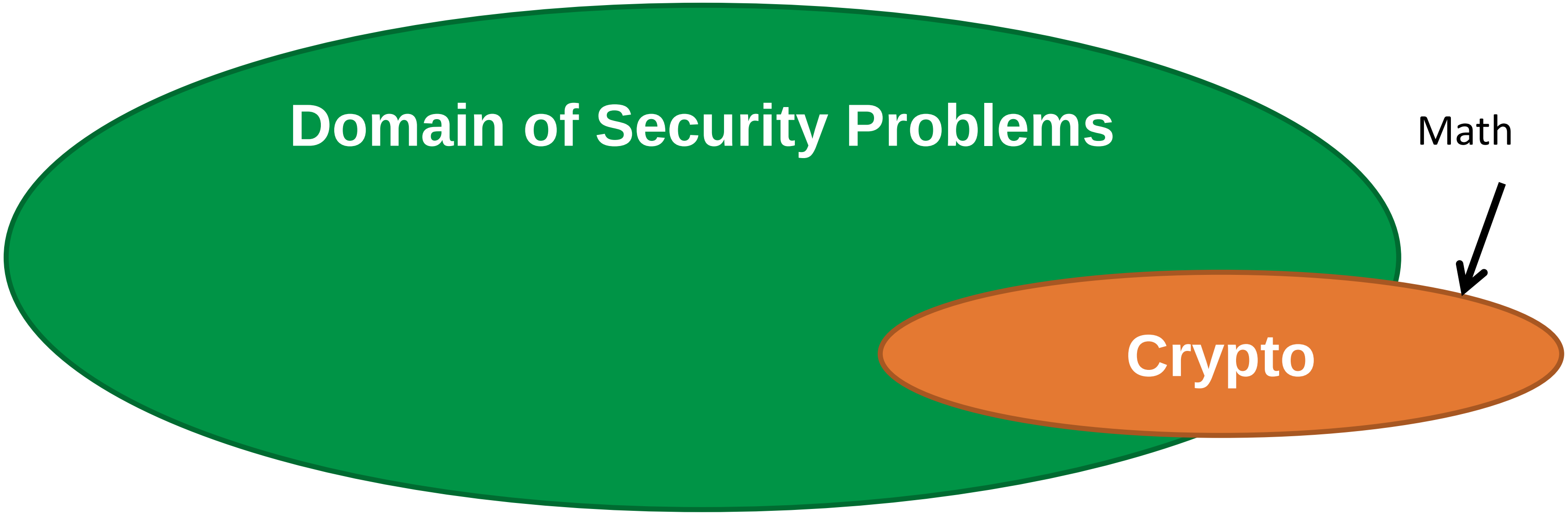
The #1 Rule

Never role your own crypto.

(including inventing your own protocol)

underlying hard problem

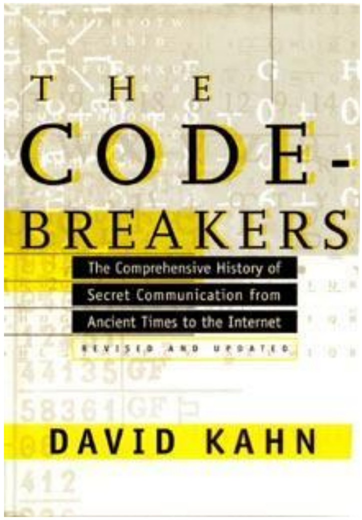
Computer Security



- How do write software that can protect private information like K_e, K_D?
- How do we know implementation of ***E*** and ***D*** are correct?
- How do we build networks that are secure, reliable, and available?
- How do we ensure only Alice can access her keys?

History of Cryptography

David Kahn, "The code breakers" (1996)



Early History: Substitution Cipher

- $K_e = K_d = \pi: \Sigma \rightarrow \Sigma$
- e.g., Σ = {a,b,c,...} or {1,2,3,..} etc.
- π is a permutation

σ	A	B	C	D
$\pi(\sigma)$	E	A	Z	U

E_π(CAB)
= π(C) π(A) π(B)
= Z E A

D_π(ZEA)
= π^{-1}(Z) π^{-1}(E) π^{-1}(A)
= C A B

Complete Insecure!

Attacking Substitution Ciphers

- How would you break a message encrypted with the substitution cipher?

- Analyze the ciphertext (CT attack)!

- Frequency of letters
  - “e” 12.7%, “t” 9.1%, “a” 8.1%, ...

- Pairs of letters: “he”, “an”, “in”, “th”, ...

An Example

UKBYBIPOUZBCUFEEBORUKBYBHOBBRFESPVKBWFOFERVNBCVBZPRU
BOFERVNBCVBPCYYFVUFOFEIKNWFRFIKJNUPWRFIPOUNVNIPUBRNCU
KBEFWWFDNCHXCYBOHOPYXPUBNCUBOYNRVNIWNCPOJIOFHOPZRVFZ
IXUBORJRUBZRBCHNCBBONCHRJZSFWNVRJRUBZRPCYZPUKBZPUNVPW
PCYVFZIXUPUNFCPWRVNBCVBRPYYNUNFCPWWJUKBYBIPOUZBCUIPOU
NVNIPUBRNCHOPYXPUBNCUBOYNRVNIWNCPOJIOFHOPZRNCRVNBCUN
ENVVFZIXUNCHPCYVFZIXUPUNFCPWZPUKBZPUNVR

B	36	→ E
N	34	
U	33	→ T
P	32	→ A
C	26	

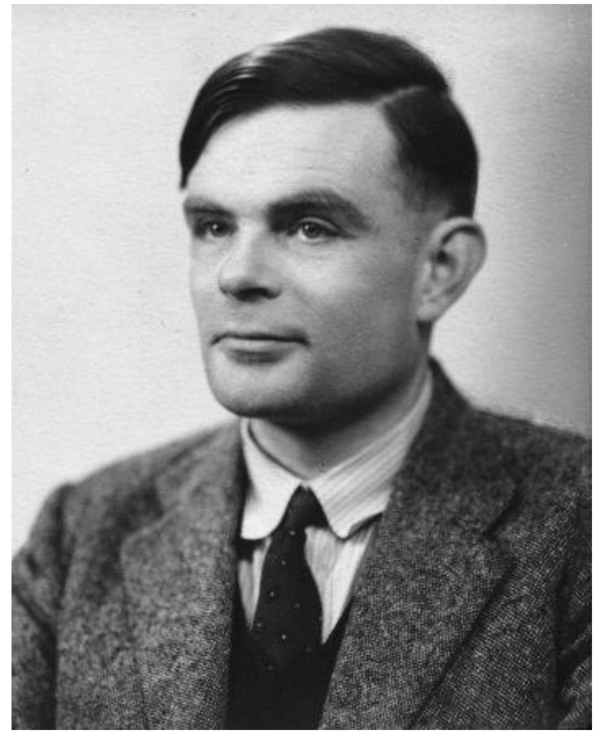
NC	11	→ IN
PU	10	→ AT
UB	10	
UN	9	

digrams

UKB	6	→ THE
RVN	6	
FZI	4	

trigrams

WWII: Enigma



Broken by an effort led by our friend
Alan Turing

Classical Approach: Iterated Design

Scheme 1 → Broken

↓

Scheme 2 → Broken

↓

Scheme 3 → Deploy → Broken

↓

…

No way to say anything is secure
(and you may not know when broken)

Iterated design was only one known until 1945

Claude Shannon: 1916 - 2001

- Modern Cryptography: 1945 with Shannon
- Formally define *security goals*, *adversarial models*, *and security of system*
- Beyond iterated design: Proof by reduction that cryptosystem achieves goals

Proving Information Theoretic Secrecy

Given: $\forall m_0, m_1 \in M.\text{where}|m_0| = |m_1|$

$\forall c \in C.$

$\Pr\left[E(k, m_0) = c\right] = \Pr\left[E(k, m_1) = c\right]$

Fact: $\Pr\left[E(k, m) = c\right] = \dfrac{k \in K \text{ s.t. } E(k, m) = c}{|K|}$

So, if

$$\forall m, c : \#\{k \in K : E(k, m) = c\} = \text{constant}$$

Then perfectly secure.

Stream Ciphers

PRNG's and amplifying secrets

Amplifying Randomness

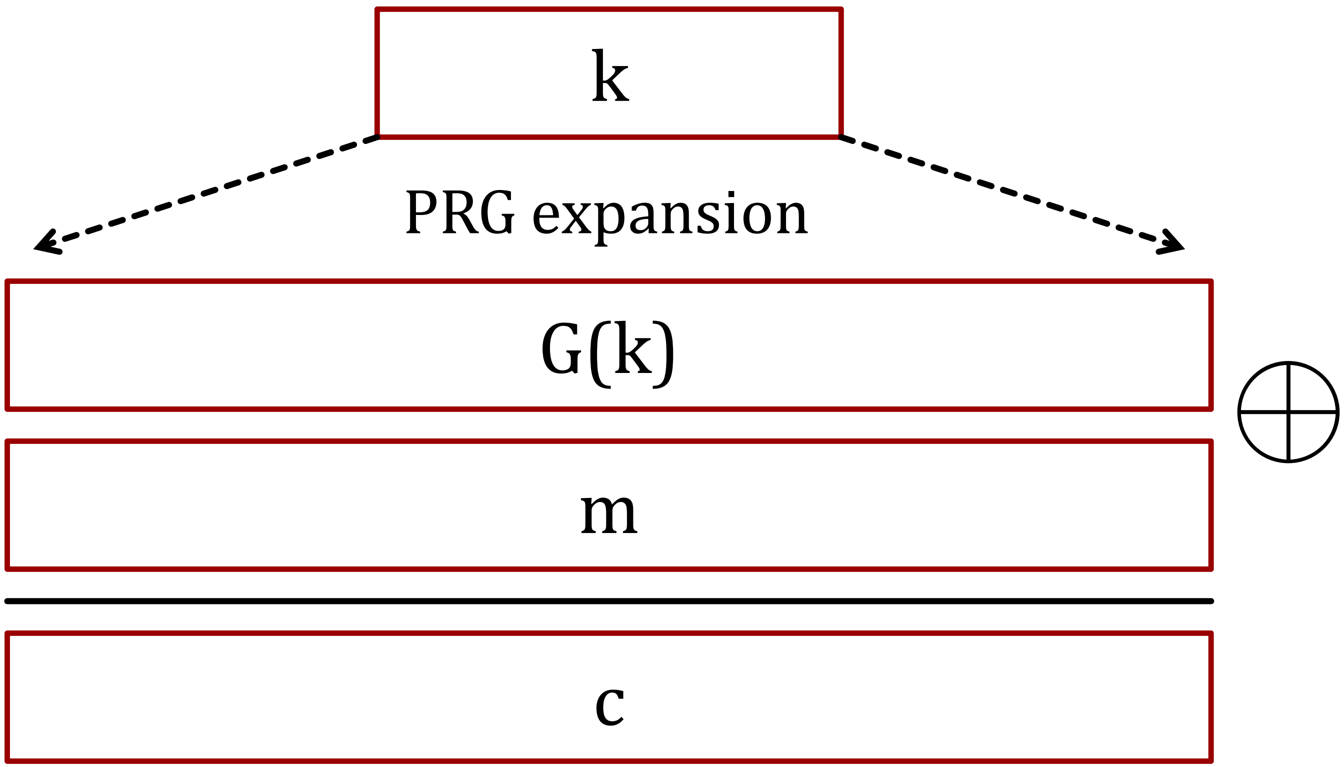
Problem: Perfect cipher requires |K| >= |M|

To make practical: replace "random" key with "pseudo-random" key generated by a *pseudo-random (number) generator* (PRG)

$$\text{Let } S : \{0,1\}^s \text{ and } K : \{0,1\}^k$$

$$G : S \to K \text{ where } k \gg s$$

Stream Ciphers: A Practical OTP



$$c = E(k, m) = m \oplus G(k)$$

$$D(k, c) = c \oplus G(k)$$

Question

Can a stream cipher have perfect secrecy?

- Yes, if the PRG is secure
- No, there are no ciphers with perfect secrecy
- No, the key size is shorter than the message

PRG Security

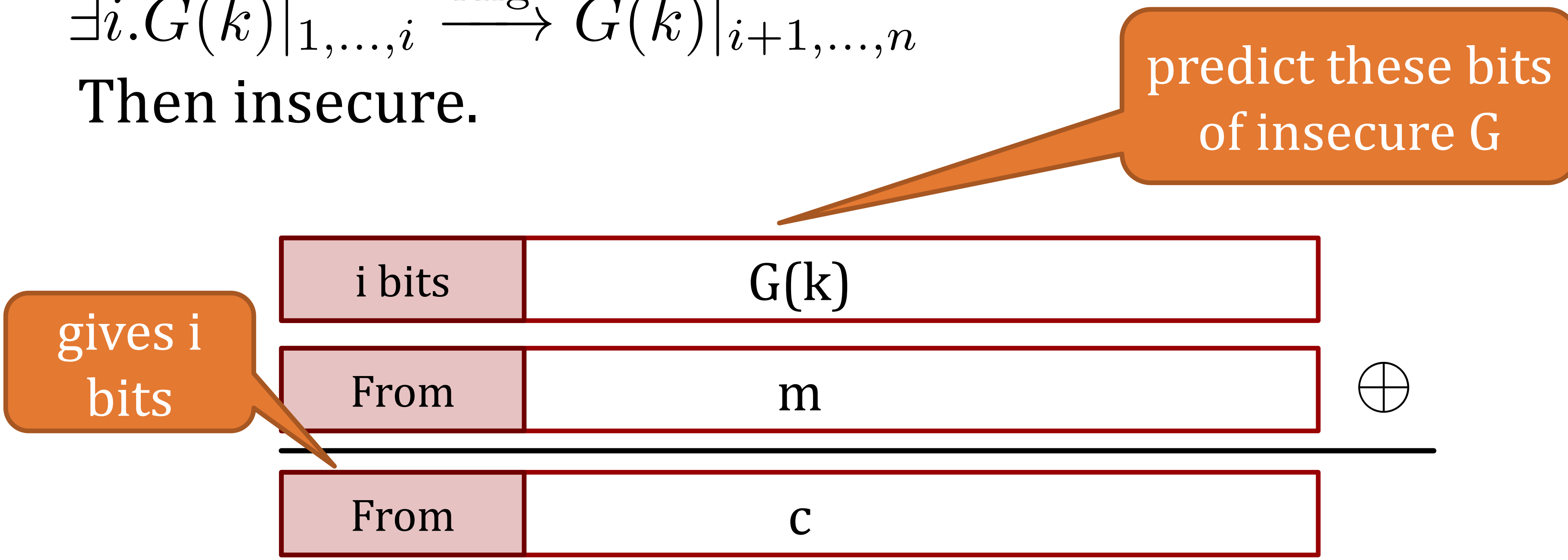
One requirement: Output of PRG is unpredictable (mimics a perfect source of randomness)

Even predicting 1 bit is insecure

Suppose PRG is predictable:

$$\exists i. G(k)|_{1,\ldots,i} \xrightarrow{\text{Alg}} G(k)|_{i+1,\ldots,n}$$

Then insecure.



PRG Security

Goal: Output of PRG is unpredictable (mimics a perfect source of randomness)

<u>Predictable:</u>

PRG G is predictable if there is an efficient alg **Adv**

$$\Pr\left[\mathbf{Adv}(G(k)|_{1,\ldots,i}) = G(k)|_{i+1}\right] > \frac{1}{2} + \epsilon$$

for non-negligible ε (for now, ε> $1/2^{30}$)

<u>Unpredictable:</u>

PRG is unpredictable if not predictable for all i

Negligible Functions

Practical:

Something is negligible if it is very small *constant*.

- Non-negligible: 2^{30} (one GB of data)
- Negligible: 2^{80} (age of universe in seconds: 2^{60})

Formally:

A *function* ε: $Z^{\geq 0} \rightarrow R^{\geq 0}$ is negligible if it approaches 0 faster than the reciprocal of any polynomial.

$$\forall n \in \mathbb{N}.\exists n_c : \epsilon(n) \leq n^{-c} \qquad \forall n \geq n_c$$

Weak PRGs

- Linear congruence generators
  - Look random (see Art of Programming)
  - But are predictable

- GNU libc random()
  - Kerberos v4 did and was broken

Two Time Pad is Insecure

Two Time Pad:

$c_1 = m_1 \oplus k$

$c_2 = m_2 \oplus k$

Enough redundancy in ASCII (and english) that $m_1 \oplus m_2$ is enough to know m_1 and m_2

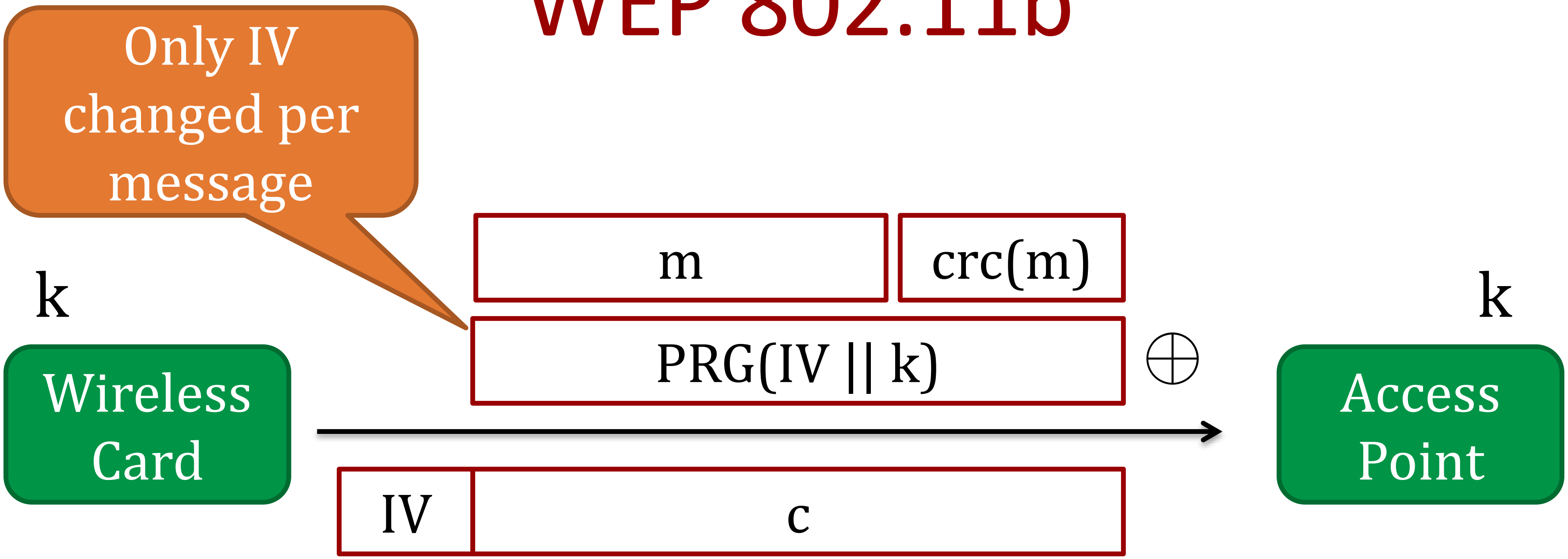
Eavesdropper gets c_1 and c_2.
What is the problem?

$$c_1 \oplus c_2 = m_1 \oplus m_2$$

Real World Examples

- Project Venona (~1942-1945)
  - Russians used same OTP twice → break by American and British cryptographers
- WEP 802.11b
- Disk Encryption
- MS-PPTP (Windows NT)

WEP 802.11b



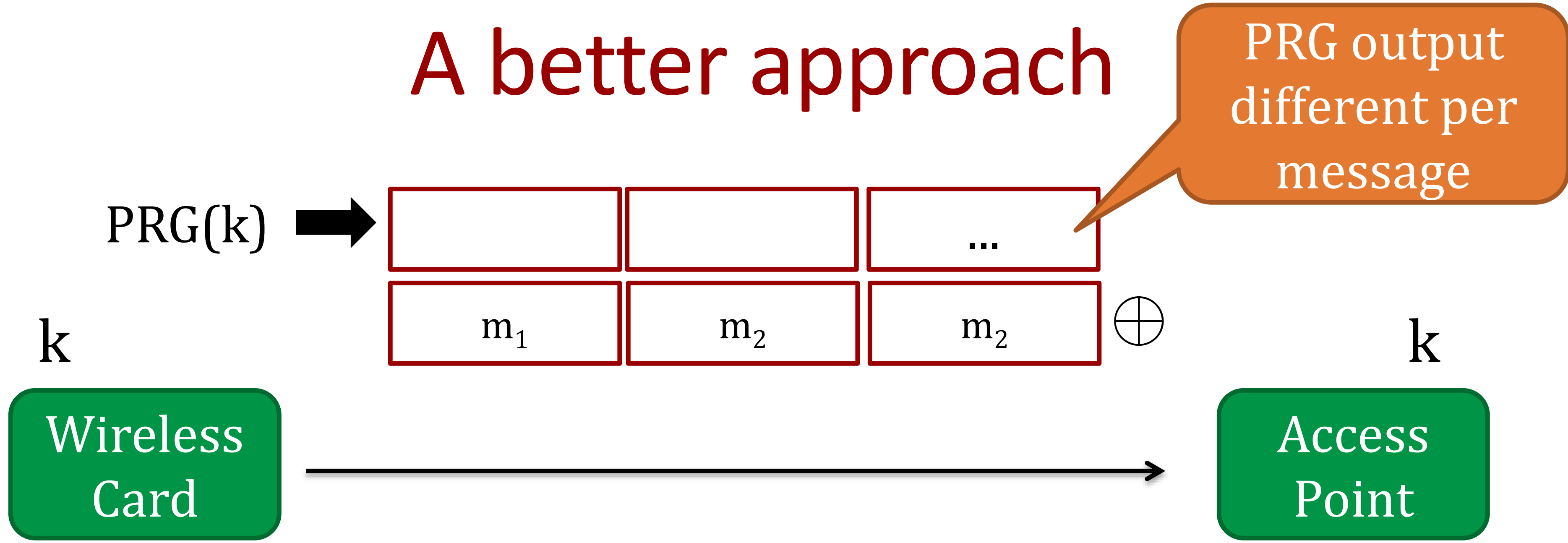
Length of IV: 24 bits

– Repeat after $2^{24} \approx$ 16M frames

– Some cards reset to 0 after power cycle

– Best attacks reduce to 10^6

A better approach



Each message has a unique key

Best method: use WPA2

Disk Encryption

m_1

Dear Alice:
You are my
sunshine.

m_2

Dear Grace:
You are my
sunshine.

$m_1 \oplus k$

Dear Alice: You are my sunshine.

$m_2 \oplus k$

Dear Grace: You are my sunshine.

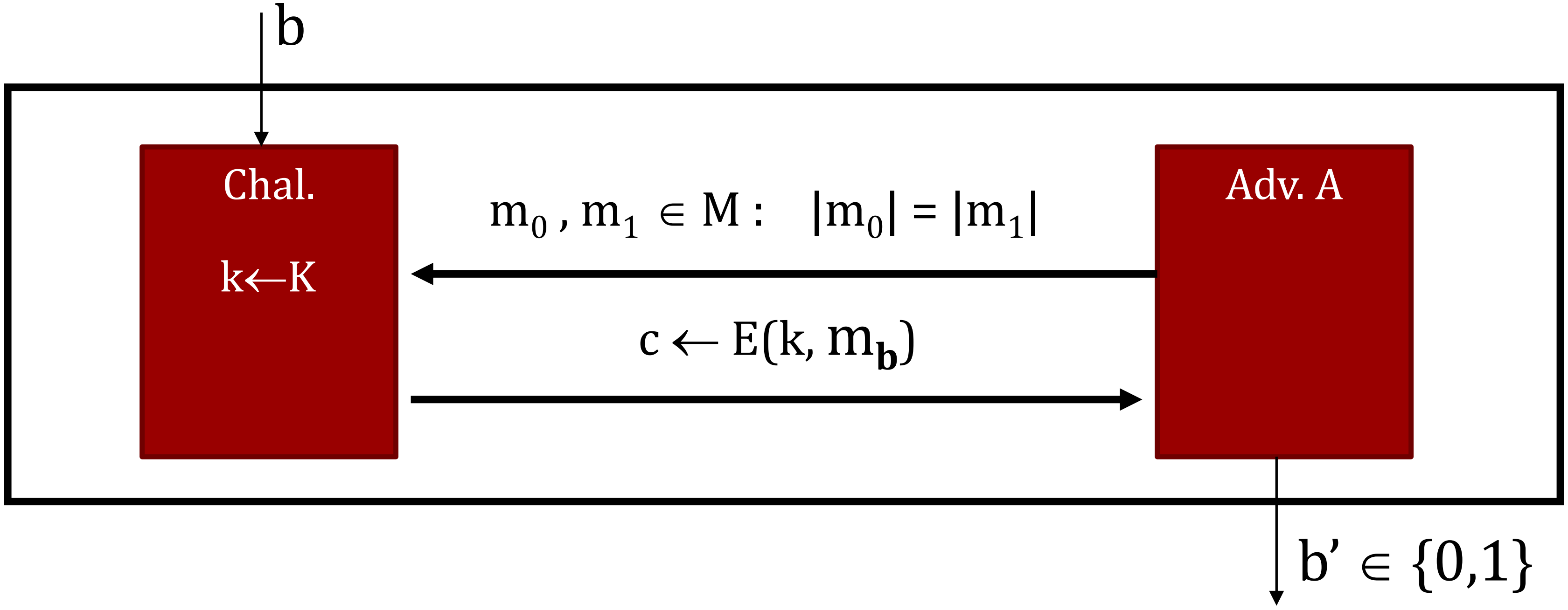
Attacker knows where messages are same, and where different!

Two Time Pad

Never use the same stream cipher key twice!

– Network traffic: Pick a new key each time, and a separate key for client and server

– Disk encryption: don't use stream cipher

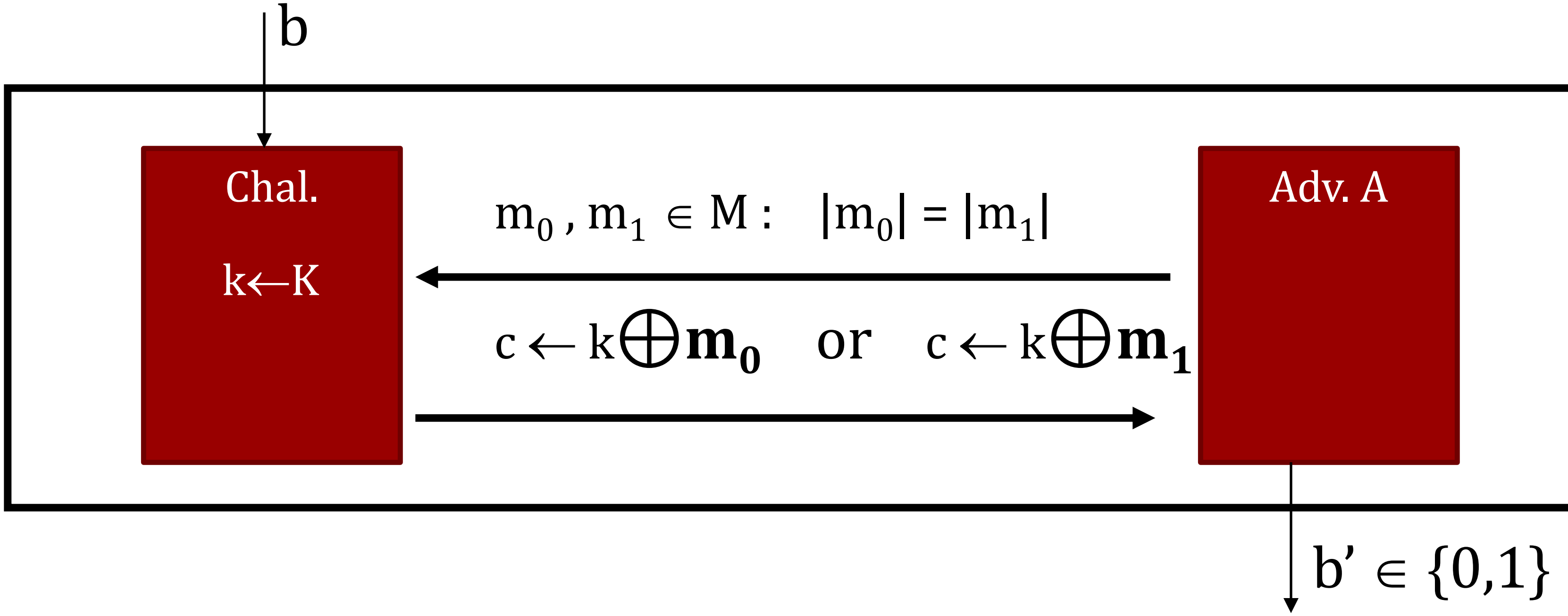


for b=0,1: W_b := [event that EXP(b)=1]

$\mathrm{Adv}_{SS}[A,E] := \left| \Pr[W_0] - \Pr[W_1] \right| \in [0,1]$

Security for:
- Chosen Plaintext Attack (CPA)
- IND-CPA Game

OTP is semantically secure



For **all** A:

$\mathrm{Adv}_{SS}[A,OTP] = |\ \Pr[\ \mathbf{A(k \oplus m_0)}=1\ ] - \Pr[\ \mathbf{A(k \oplus m_1)}=1\ ]\ |$
$= 0$