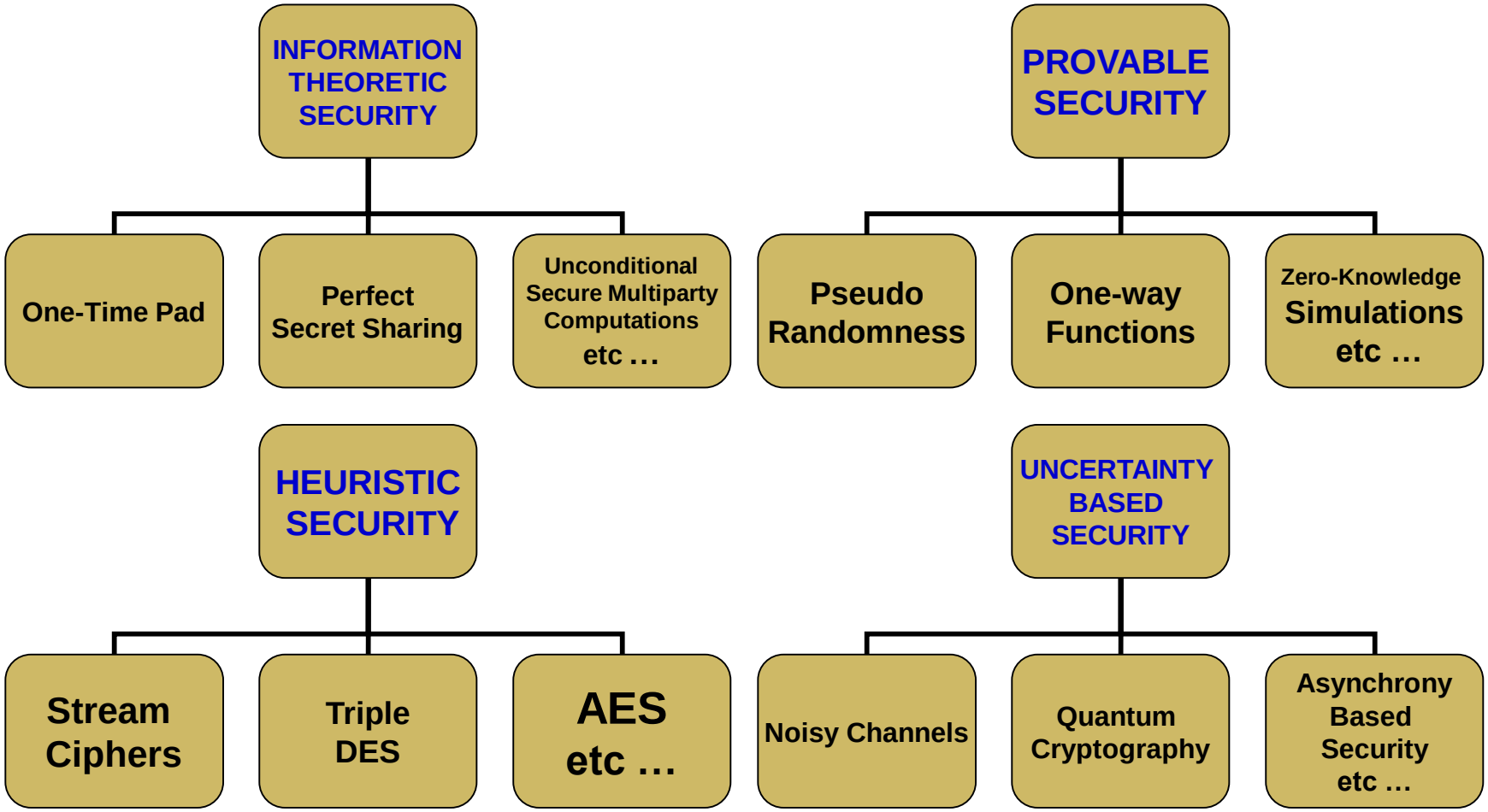
PRINCIPLES OF INFORMATION SECURITY

The Subject Matter, Basic Pre-requisites and Design Challenges

THE SUBJECT MATTER

Today's Multi-verses of Security

The Four Silos of Contemporary Security



BUT WHY DIFFERENT TYPES OF SECURITY?

- ***P = NP*** : A world where it is *impossible* to pose questions (with easily demonstrable solutions) that are hard to answer.
- ***P \neq NP (but only in worst-case)*** : A world where it is *infeasible* to pose questions (with easily demonstrable solutions) that are hard to answer.
- ***Average-case hard problems, but no one-way functions*** : A world where it is *feasible* to pose questions (with easily demonstrable solutions) that are hard to answer (but infeasible for the poser himself/herself to solve it!).
- ***One-way Functions but no PKC*** : A world where it is *easy* to pose questions (with easily demonstrable solutions) that are hard to answer.
- ***Public-Key Cryptography*** : It is *easy* to pose questions (with easily demonstrable solutions) that are hard to answer *except* for a selected group.

WHY IS HEAVY-DUTY MATHEMATICS NEEDED FOR SECURITY?

Kerckhoff's Principle

Advantages of publishing

Kerckhoff's Principle

Dr. Auguste Kerckhoff
1883

"a cryptosystem should be secure even if everything about the system, except the key, is public knowledge"

Why should it be made public?

- Reverse Engineering
- Costly Storage
- Ease of replacement
- Multiple Users

- Ethical Hacking
- Standards

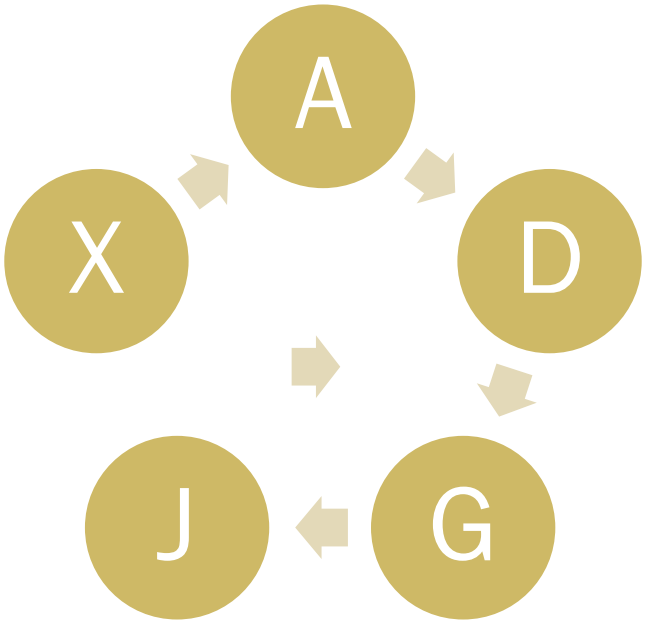
CRYPTANALYSIS OF HISTORICAL CIPHERS

Shift Cipher, Mono-alphabetic Substitution Cipher and Vigenere Cipher

The Shift Cipher and the Sufficient Key Space Principle

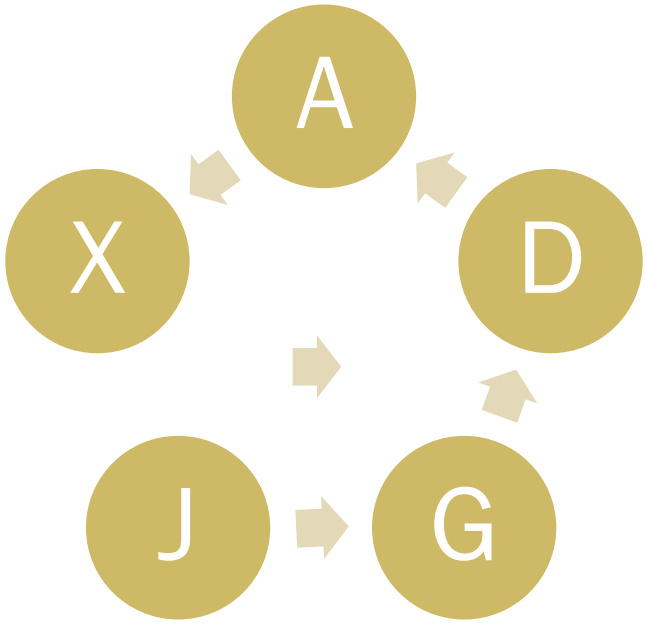
SHIFT CIPHER

- Rotate each letter by the key k
- For example, if k is 3 then:



Encryption

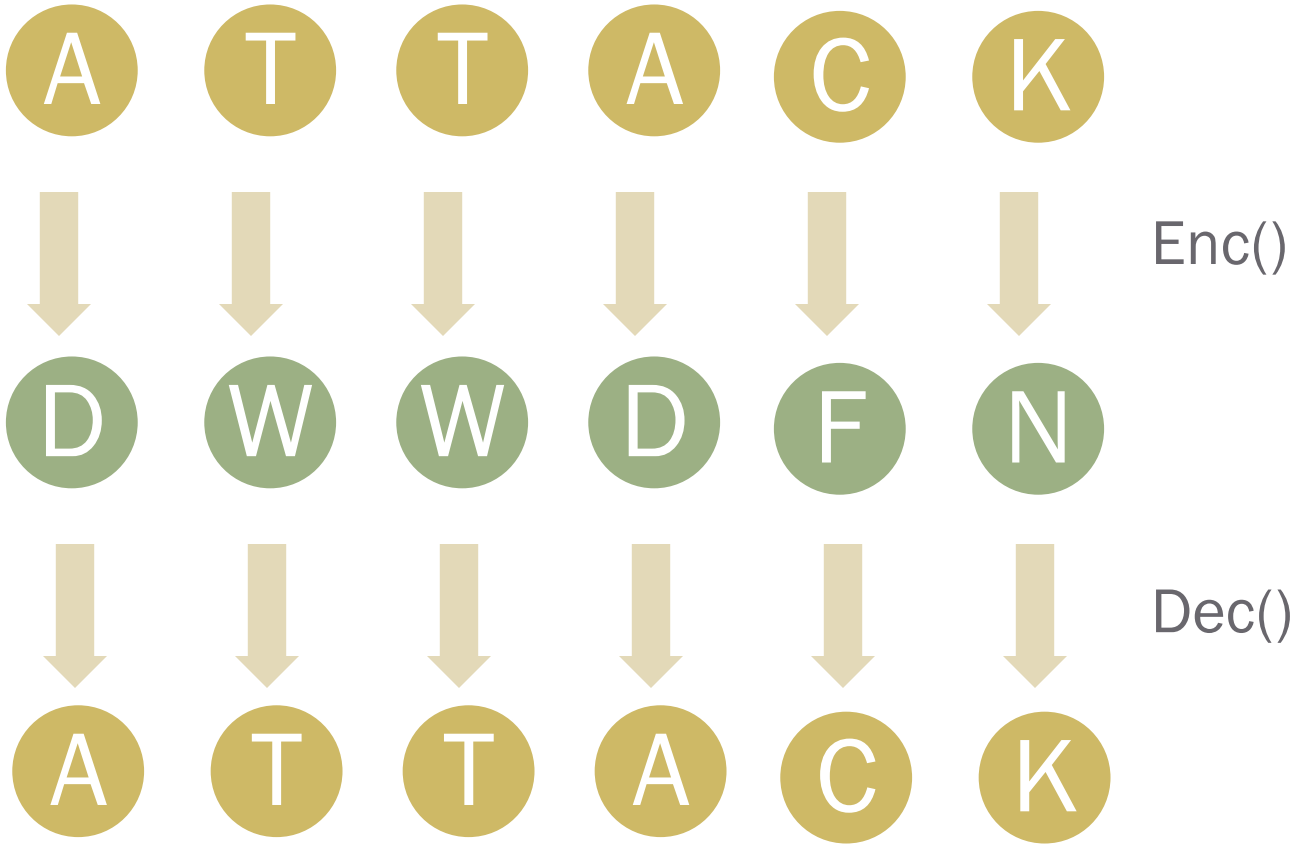
Enc(x) = (x + k) mod 26.



Decryption

Dec(x) = (x - k) mod 26

Example: Key = 3 and Plaintext = “ATTACK”



Problem with Shift ciphers

- Not enough keys!
- If we shift a letter 26 times, we get the same letter back.
  - A shift of 27 is the same as a shift of 1, etc.
  - So we only have 25 keys (1 to 25).
- Therefore, easy to attack via brute force.

Example: Cryptanalysis of shift ciphers

- Cipher text : OVDTHUFWVZZPISLRLFZHYLAOLYL

Key Value	Possible Plain Text
1	NUCSGTEVUYYOHRKQKEYGXKZNKXK
2	MTBRFSDUTXXNGQJPJDXFWJYMJWJ
3	LSAQERCTSWWMFPIOICWEVIXLIVI
4	KRZPDQBSRVVLEOHNHBVDUHWKHUH
5	JQYOCPARQUUKDNGMGAUCTGVJGTG
6	IPXNBOZQPTTJCMFLFZTBSFUIFSF
7	HOWMANYPOSSIBLEKEYSARETHERE
8	GNVLZMXONRRHAKDJDXRZQDSGDQD
9	FMUKYLWNMQQGZJCICWQYPCRFCPC
10	ELTJXKVMLPPFYIBHBVPXOBQEBOB
11	DKSIWJULKOOEXHAGAUOWNAPDANA
12	CJRHVITKJNNDWGZFZTNVMZOCZMZ
13	BIQGUHSJIMMCVFYEYSMULYNBYLY

Mono-Alphabetic Substitution Cipher: Just Large Key Space Isn't Enough!

Mono Alphabetic Substitution Cipher

Each character in the plain text

Mapped to →

Unique character in the Cipher Text

→ A total of 26! Keys

Plain Text : “Tell him about me”

Assume the following mapping:

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
F	A	Y	E	Z	C	S	G	V	H	W	Q	B	D	J	U	I	L	K	M	X	R	T	N	O	P

The plaintext would be encrypted as follows: (ignore the spaces)

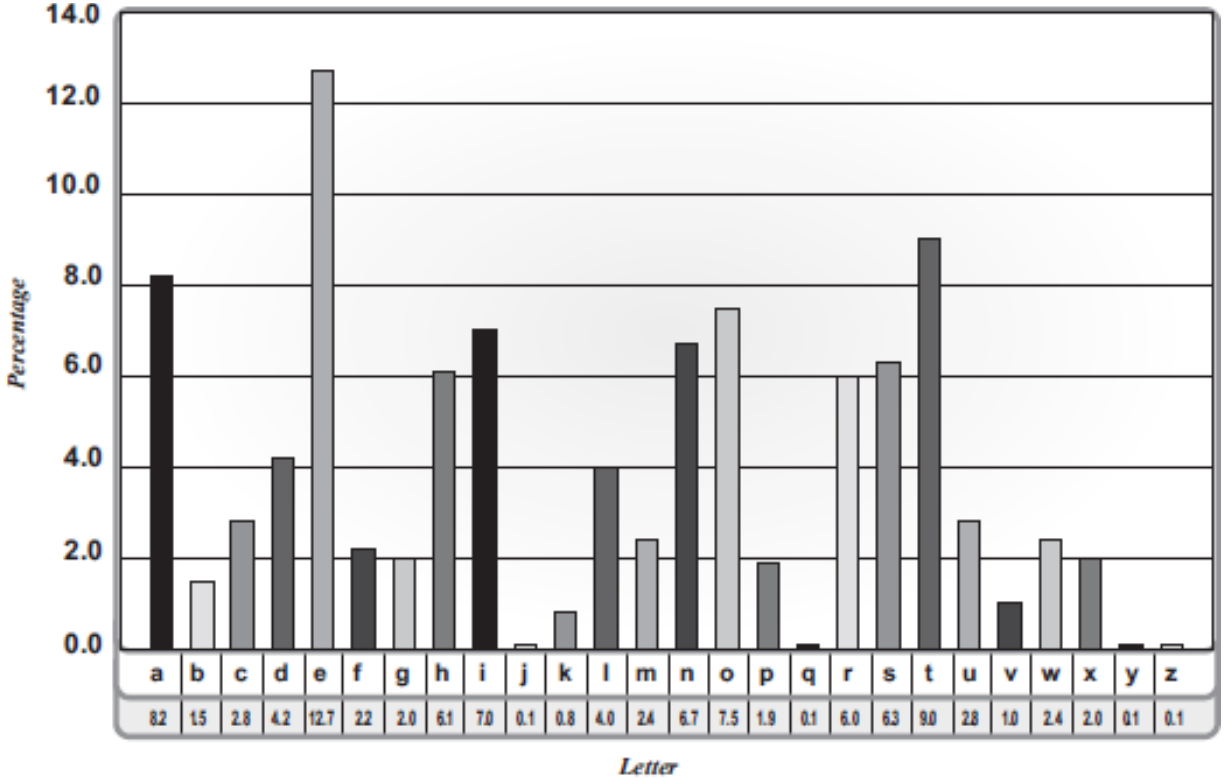
t	e	l	l		h	i	m		a	b	o	u	t		m	e
M	Z	Q	Q		G	V	B		F	A	J	X	M		B	Z

Frequency Analysis

Consider the cipher:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZVUEPHZHMDZSHZOWSF
PAPPDTSVPQUZWYMXUZUHSXEPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

Standard Frequency Distribution of English Alphabets:



Frequency of alphabets in the cipher text:

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				

{P,Z} Є {e,t}

{S,U,O,M,H} Є {a,h,I,n,o,r,s}

{A,B,G,Y,I,J} Є {b,j,k,q,v,x,z}

What is the actual mapping?

The Plaintext (after breaking it)

it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the viet cong in moscow

A Statistical Approach to Break Shift Cipher

p_i : Probability of i^{th} character in plaintext

$$\sum_{i=0}^{25} p_i^2 \approx 0.065$$

q_i : Probability of i^{th} character in ciphertext

$$I_j \stackrel{\text{def}}{=} \sum_{i=0}^{25} p_i \cdot q_{i+j}$$

Find k such that I_k is nearly 0.065

Vigenere Cipher: Code Complexity Does Not Guarantee Security!

Vigenere Cipher

Poly alphabetic substitution

Each character in the plain text → Mapped to → Any character in the Cipher Text

Blaise de Vigenere

Try breaking my cipher !

Encryption

Periodic key: deceptivedeceptivedeceptive

Plaintext: wearediscoveredsaveyourself

Ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

d	e	c	e	p	t	i	v	e	d	e	c	e	p	t	i	v	e	d	e	c	e	p	t	i	v	e
w	e	a	r	e	d	i	s	c	o	v	e	r	e	d	s	a	v	e	y	o	u	r	s	e	l	f
Z	I	C	V	T	W	Q	N	G	R	Z	G	V	T	W	A	V	Z	H	C	Q	Y	G	L	M	G	J

Cryptanalysis of Vigenere Cipher

Brute force → Infeasible !

Frequency Analysis → Information is Obscured !

Unbreakable ?

NO !

Still there is scope for Frequency Analysis !

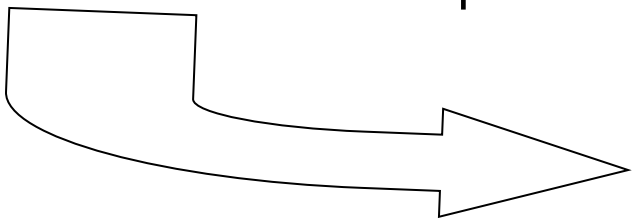
Not all information is lost !

Attacker

Cryptanalysis (Contd.)

1. Find the Key Length

Two Steps:



2. Find the Key

A Statistical Approach to Vigenere Cipher

p_i : Probability of i^{th} character in plaintext

$$\sum_{i=0}^{25} p_i^2 \approx 0.065$$

q_i : Probability of i^{th} character in the cipher sub-sequence

$$c_1, c_{1+\tau}, c_{1+2\tau}, \ldots$$

$$S_\tau \stackrel{\text{def}}{=} \sum_{i=0}^{25} q_i^2$$

Find key length t such that S_t is nearly 0.065

Once key length t is found, Vigenere cipher is nothing but a collection of t shift ciphers!

DESIGNING SECURE SYSTEMS IS HARD

The entry of SHANNON's Genius (next class!)

THANK YOU

Any Questions?

SHANNON’s THEORY

PERFECT SECRECY

ENCRYPTION SCHEME (Generic Definition)

- Three Algorithms
  - Key Generation (***Gen***)
  - Encryption (***Enc***)
  - Decryption (***Dec***)
- Message Space ***M***

Perfect Secrecy

- Adversary must not obtain any *additional* information about the message, due to the ciphertext

For any message m and any valid ciphertext c,

$$\Pr[\text{Message} = m \mid \text{Ciphertext} = c] = \Pr[\text{Message} = m]$$

PERFECTLY SECRET ENCRYPTION (textbook definition)

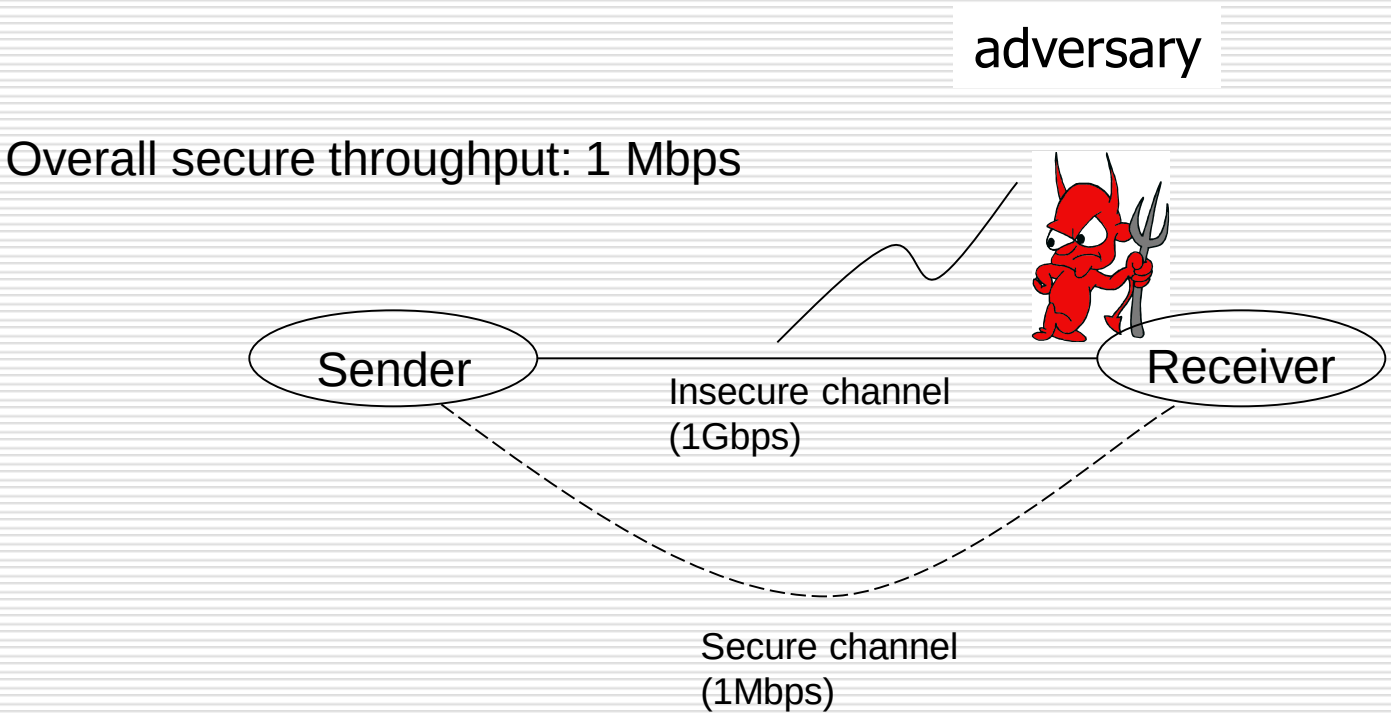
DEFINITION 2.1 *An encryption scheme* (Gen, Enc, Dec) *over a message space* $\mathcal{M}$ *is* perfectly secret *if for every probability distribution over* $\mathcal{M}$*, every message* $m \in \mathcal{M}$*, and every ciphertext* $c \in \mathcal{C}$ *for which* $\Pr[C = c] > 0$*:*

$$\Pr[M = m \mid C = c] = \Pr[M = m].$$

Achieving Perfect Secrecy is Impractical

Shannon proved that for perfect secrecy the size of the (and the entropy of) key space be at least as much as the size of (and the entropy of) the message space

Why is it Impractical?



- The secure communication throughput of the entire system is same as the cumulative bandwidth of truly secure channels in the system

The Issue

- Fast secure channels are required for efficient secure communication

"Chicken-and-Egg" problem!

EQUIVALENT STATEMENTS of PERFECT SECRECY

LEMMA 2.2 *An encryption scheme* (Gen, Enc, Dec) *over a message space* $\mathcal{M}$ *is perfectly secret if and only if for every probability distribution over* $\mathcal{M}$, *every message* $m \in \mathcal{M}$, *and every ciphertext* $c \in \mathcal{C}$:

$$\Pr[C = c \mid M = m] = \Pr[C = c].$$

PROOF

If:

$$\Pr[C = c \mid M = m] = \Pr[C = c]$$

Then, multiplying both sides by: $\Pr[M = m] / \Pr[C = c]$

$$\frac{\Pr[C = c \mid M = m] \cdot \Pr[M = m]}{\Pr[C = c]} = \Pr[M = m]$$

Using Bayes' Theorem, L.H.S. is: $\Pr[M = m \mid C = c]$.

Similarly, the other direction.

PERFECT INDISTINGUISHABILITY

LEMMA 2.3 *An encryption scheme* (Gen, Enc, Dec) *over a message space* $\mathcal{M}$ *is perfectly secret if and only if for every probability distribution over* $\mathcal{M}$, *every* $m_0, m_1 \in \mathcal{M}$, *and every* $c \in \mathcal{C}$:

$$\Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1].$$

PROOF

One direction follows from previous Lemma.

For the other direction, fix p = $\Pr[C = c \mid M = m_0]$

$$\Pr[C = c] = \sum_{m \in \mathcal{M}} \Pr[C = c \mid M = m] \cdot \Pr[M = m]$$

$$= \sum_{m \in \mathcal{M}} p \cdot \Pr[M = m]$$

$$= p \cdot \sum_{m \in \mathcal{M}} \Pr[M = m] \qquad = p$$

ONE-TIME PAD (VERNAM CIPHER)

- Message Space = Key Space = Cipher space = $\{0,1\}^n$
- ***Gen:*** Choose an n-bit key uniformly at random.
- ***Enc:*** c = m xor key
- ***Dec:*** m = c xor key

Vernam Cipher is perfect

$$\Pr[C = c \mid M = m] = \Pr[M \oplus K = c \mid M = m]$$

$$= \Pr[m \oplus K = c] = \Pr[K = m \oplus c]$$

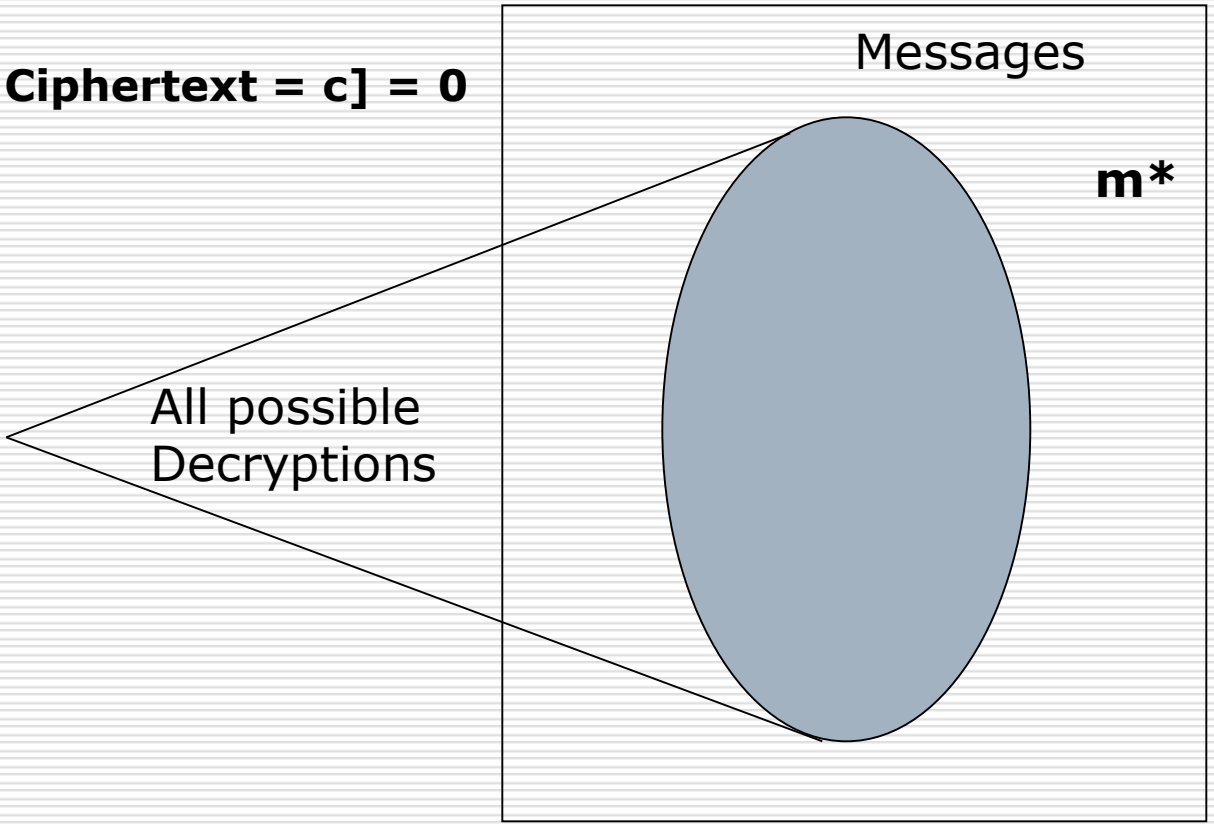
$$= 1/2^n$$

LIMITATIONS OF PERFECT SECRECY

Key space is at least as large as the message space

If More Messages than Keys then it CANNOT be PERFECT!

P[Message = m* | Ciphertext = c] = 0



(Next Class)
Two Relaxations to Perfect Secrecy

- Security is only preserved against efficient adversaries that run in a feasible amount of time

- Adversaries can potentially succeed with some very small probability

These two relaxations are necessary and (if certain interesting mathematical objects exist) are also sufficient

THANK YOU

Any Questions?

PSEUDORANDOMNESS

And One-way Functions

(RECALL)
Two Relaxations to Perfect Secrecy

- Security is only preserved against efficient adversaries that run in a feasible amount of time

- Adversaries can potentially succeed with some very small probability

These two relaxations are necessary and (if certain interesting mathematical objects exist) are also sufficient

DEFINING "EFFICIENT" ADVERSARY

- Borrowing from Tenets of Theoretical Computer Science:
  - Efficiency = **PROBABILISTIC POLYNOMIAL TIME STRATEGIES (PPT)**

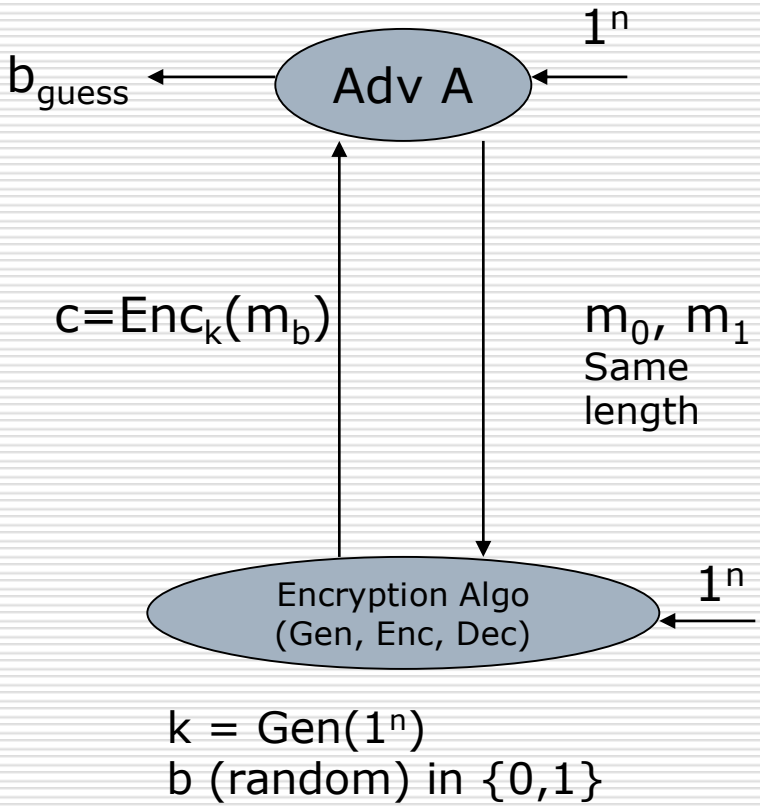
DEFINING "NEGLIGIBLE" PROBABILITY

- **Negligible Functions (asymptotically) grow slower than the inverse of any polynomial.**

Textbook Definition

DEFINITION 3.4 *A function f is* negligible *if for every polynomial $p(\cdot)$ there exists an N such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.*

Defining Computational Security Against Eavesdroppers



Cryptosystem is secure against eavesdroppers if for all PPT adversaries A:

$P[b_{guess} = b] <= ½ + negl(n)$

PSEUDORANDOM GENERATORS

And computational
indistinguishability

Pseudo Random Generator (PRG)

A *deterministic polynomial time algorithm* G, inputs n bits and outputs l(n) bits where:

(a) $l(n) > n$

And

(b) Output of G is computationally indistinguishable from uniform distribution

PRG
(Textbook Definition)

DEFINITION 3.14 *Let $\ell(\cdot)$ be a polynomial and let G be a deterministic polynomial-time algorithm such that for any input $s \in \{0,1\}^n$, algorithm G outputs a string of length $\ell(n)$. We say that G is a* pseudorandom generator *if the following two conditions hold:*

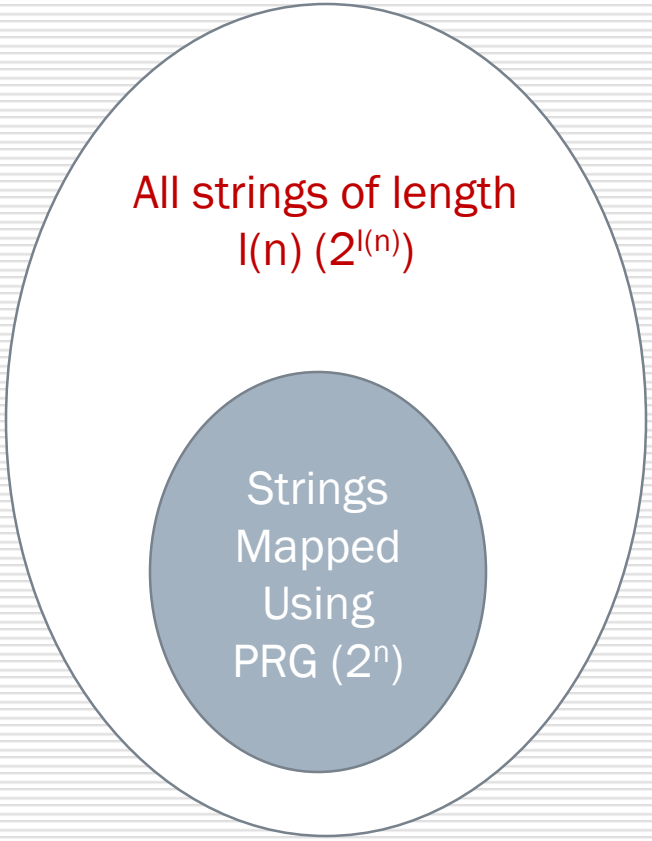
1. (Expansion:) *For every n it holds that $\ell(n) > n$.*
2. (Pseudorandomness:) *For all probabilistic polynomial-time distinguishers D, there exists a negligible function* negl *such that:*

$$\left|\Pr[D(r)=1] - \Pr[D(G(s))=1]\right| \leq \mathsf{negl}(n),$$

where r is chosen uniformly at random from $\{0,1\}^{\ell(n)}$, the seed *s is chosen uniformly at random from $\{0,1\}^n$, and the probabilities are taken over the random coins used by D and the choice of r and s.*

The function $\ell(\cdot)$ is called the expansion factor *of G.*

Discussion



- With *exponential* samples it is easy to distinguish pseudorandomness from randomness!

DESIGNING a SECURE ENCRYPTION SCHEME USING PRGs
(just "pseurorandomize the one-time pad!)

- Gen: on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$, output the ciphertext
$$c := G(k) \oplus m.$$
- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$, output the plaintext message
$$m := G(k) \oplus c.$$

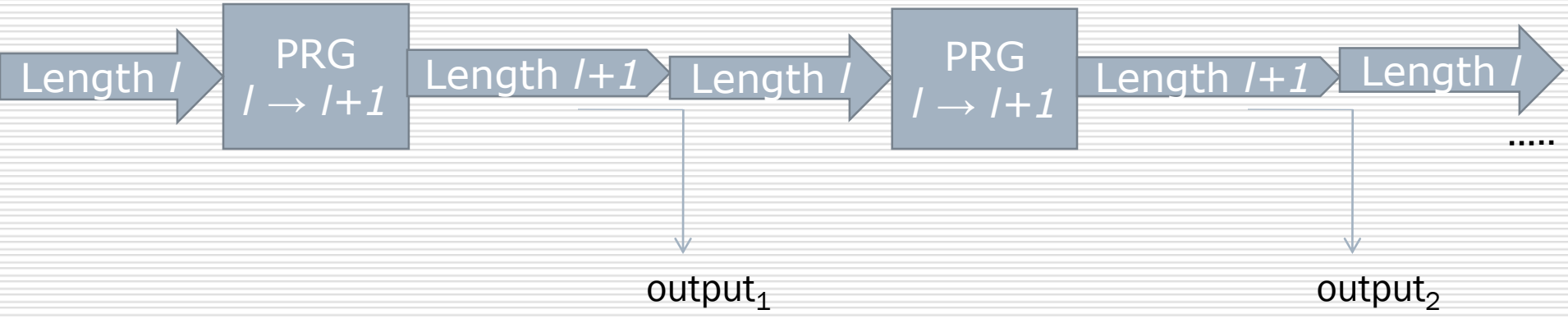
Designing PRGs from Computational Hardness

- Single bit expansion PRG to arbitrary expansion PRG
- From One-way functions to single-bit expansion PRG
- Candidate PRG from Discrete Logarithm

Expanding the Expansion in PRG

THEOREM 6.8 *Assume that there exists a pseudorandom generator with expansion factor $\ell(n) = n + 1$. Then for any polynomial $p(\cdot)$, there exists a pseudorandom generator with expansion factor $\ell(n) = p(n)$.*

THEOREM 6.8 *Assume that there exists a pseudorandom generator with expansion factor $\ell(n) = n + 1$. Then for any polynomial $p(\cdot)$, there exists a pseudorandom generator with expansion factor $\ell(n) = p(n)$.*



1. Take the last bit from *l* + *1* length string for output
2. Apply *l′* times to get output of string *l′*

Designing single-bit expansion PRGs from Computational Hardness

- One-way Functions
  - Easy to compute, Hard to Invert
  - Textbook definition:

DEFINITION 6.1 *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is* one-way *if the following two conditions hold:*

1. (Easy to compute:) *There exists a polynomial-time algorithm* M_f *computing* f*; that is,* $M_f(x) = f(x)$ *for all* x.
2. (Hard to invert:) *For every probabilistic polynomial-time algorithm* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \leq \mathsf{negl}(n).$$

A Candidate One-way Function

- Discrete Logarithm Problem (in Z_p^*)

$$f_{p,g}(x) = g^x \bmod p$$

HARDCORE PREDICATES

- Hardest bit of information about x to obtain from f(x)
- Textbook definition:

DEFINITION 6.5 *A function* hc : $\{0,1\}^* \to \{0,1\}$ *is a* hard-core predicate of a function f *if* **(1)** hc *can be computed in polynomial time, and* **(2)** *for every probabilistic polynomial-time algorithm* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(f(x)) = \mathsf{hc}(x)] \leq \frac{1}{2} + \mathsf{negl}(n),$$

where the probability is taken over the uniform choice of x *in* $\{0,1\}^n$ *and the random coin tosses of* $\mathcal{A}$.

MSB(x) is a Hardcore predicate of Discrete Logarithm Problem

From One-way Functions to PRGs

THEOREM 6.7 *Let f be a one-way permutation and let* hc *be a hard-core predicate of f. Then, $G(s) \stackrel{\text{def}}{=} (f(s), \mathsf{hc}(s))$ constitutes a pseudorandom generator with expansion factor $\ell(n) = n + 1$.*

In case of discrete logarithms:

$$G(s) = (g^s \bmod p, \mathrm{msb}(s))$$

TASK: Implement your own provably secure pseudorandom generator assuming DLP is a one-way function.

Discrete Logarithm based PRG

THANK YOU

Any Questions?

CPA-SECURITY

And

PSEUDORANDOM FUNCTIONS

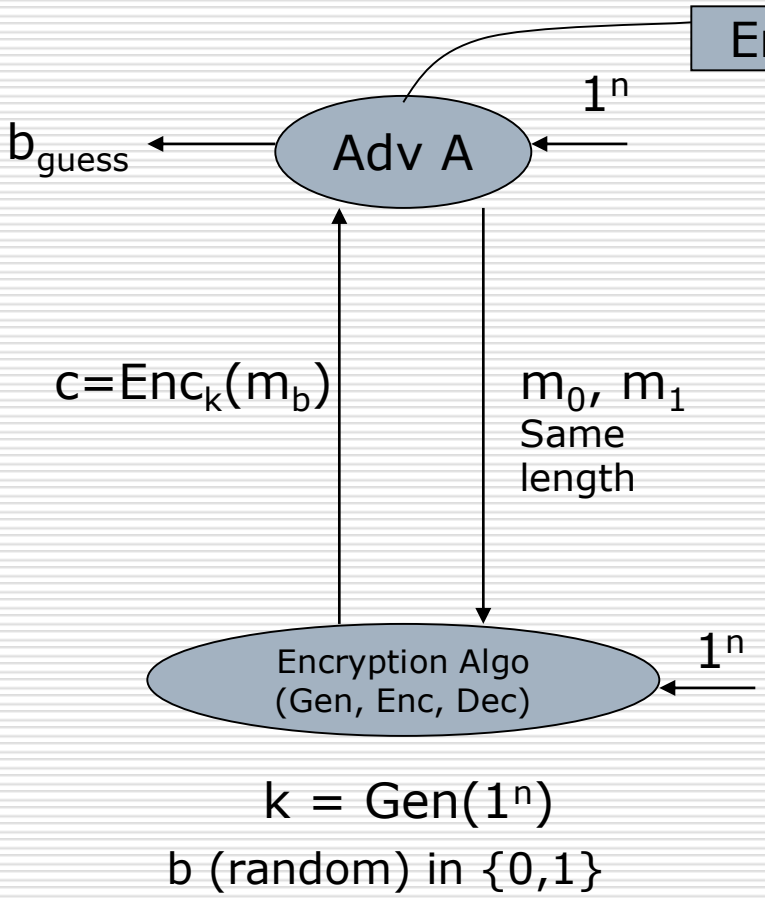
RECALL: IT'S STILL ONE-TIME USAGE!

- Gen: on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$, output the ciphertext
$$c := G(k) \oplus m.$$
- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$, output the plaintext message
$$m := G(k) \oplus c.$$

CHOSEN PLAINTEXT ATTACK (CPA)

Adversary can obtain the ciphertexts of any message of his/her choice!

Defining Computational Security Against CHOSEN PLAINTEXT ATTACK



Cryptosystem is CPA-secure if for all PPT adversaries A:

$P[b_{guess} = b] <= ½ + negl(n)$

THEOREM

- **No deterministic Encryption algortihm can be CPA-secure!**

- Why?

PROBABILISTIC ENCRYPTION

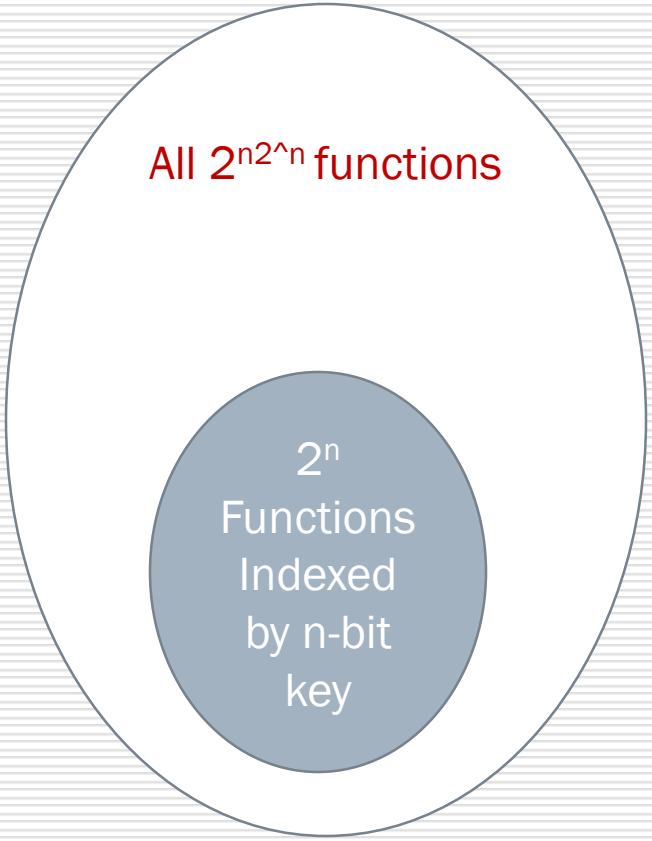
Pseudorandom Functions F_k

Basic idea: $c = (r, F_k(r) + m)$

Pseudorandom Functions (PRF)

- Functions that are easy to compute
- Computationally indistinguishable from a random function, say from domain $\{0,1\}^n$ to co-domain $\{0,1\}^n$
- Recall that there are $2^{n2\hat{}n}$ possible functions

PRFs are even more "pseudo" than PRGs



With *exponential* queries it is easy to distinguish pseudorandom functions from truly random ones!

PRF Definition

DEFINITION 3.23 *Let* $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *be an efficient, length-preserving, keyed function. We say that* F *is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers* D*, there exists a negligible function* negl *such that:*

$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq \mathrm{negl}(n),$$

where $k \leftarrow \{0,1\}^n$ *is chosen uniformly at random and* f *is chosen uniformly at random from the set of functions mapping* n*-bit strings to* n*-bit strings.*

CPA-SECURE ENCRYPTION

CONSTRUCTION 3.24

Let F be a pseudorandom function. Define a private-key encryption scheme for messages of length n as follows:

- Gen: on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose $r \leftarrow \{0,1\}^n$ uniformly at random and output the ciphertext

$$c := \langle r,\ F_k(r) \oplus m \rangle.$$

- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c = \langle r, s \rangle$, output the plaintext message

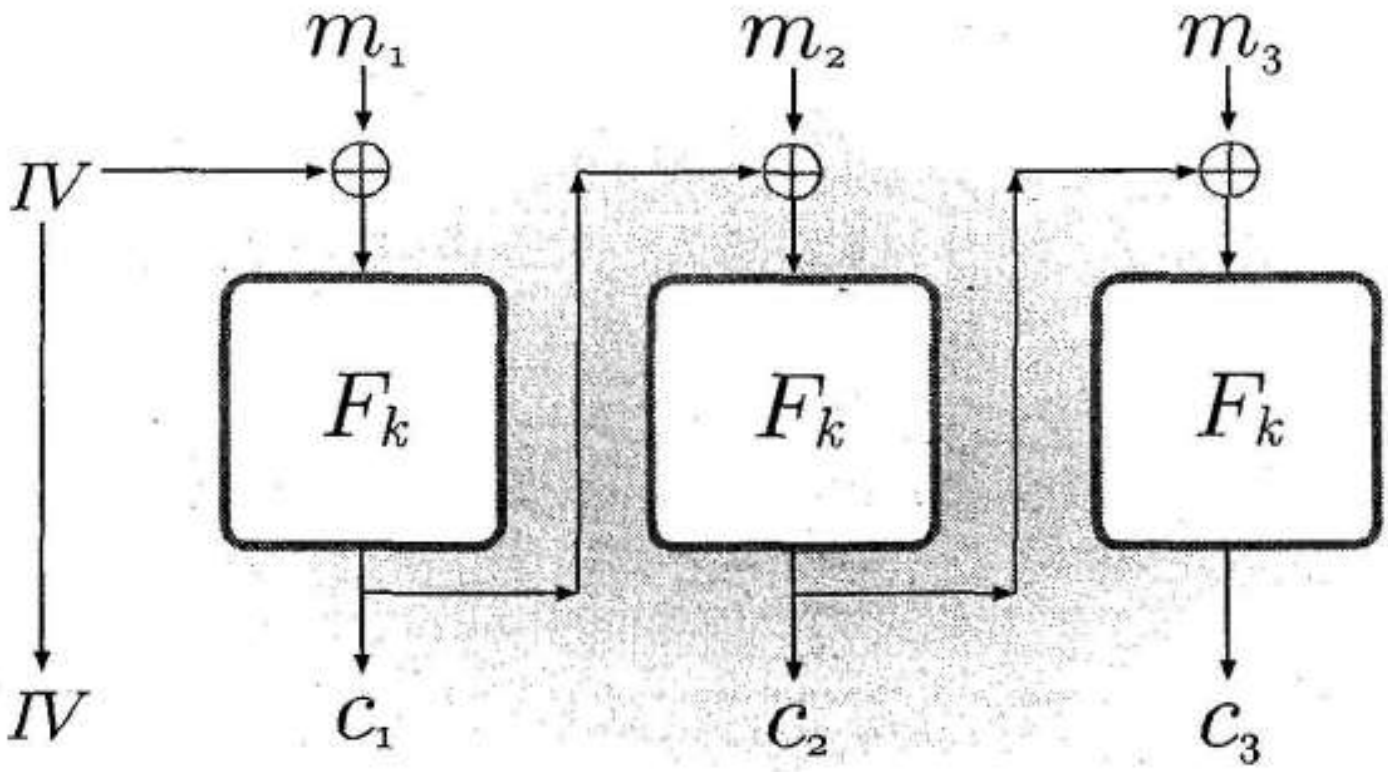
$$m := F_k(r) \oplus s.$$

A CPA-secure encryption scheme from any pseudorandom function.

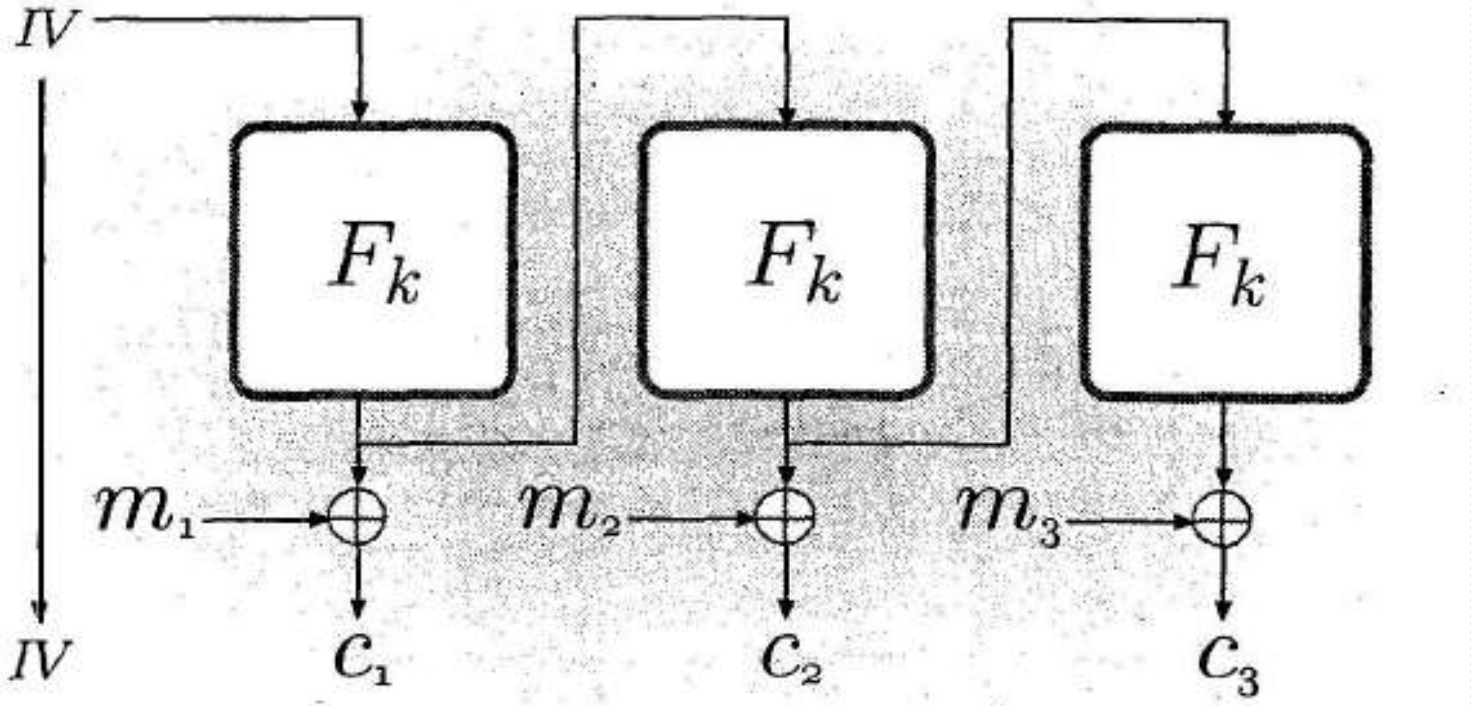
MODES OF OPERATION

Using Blockciphers/PRFs

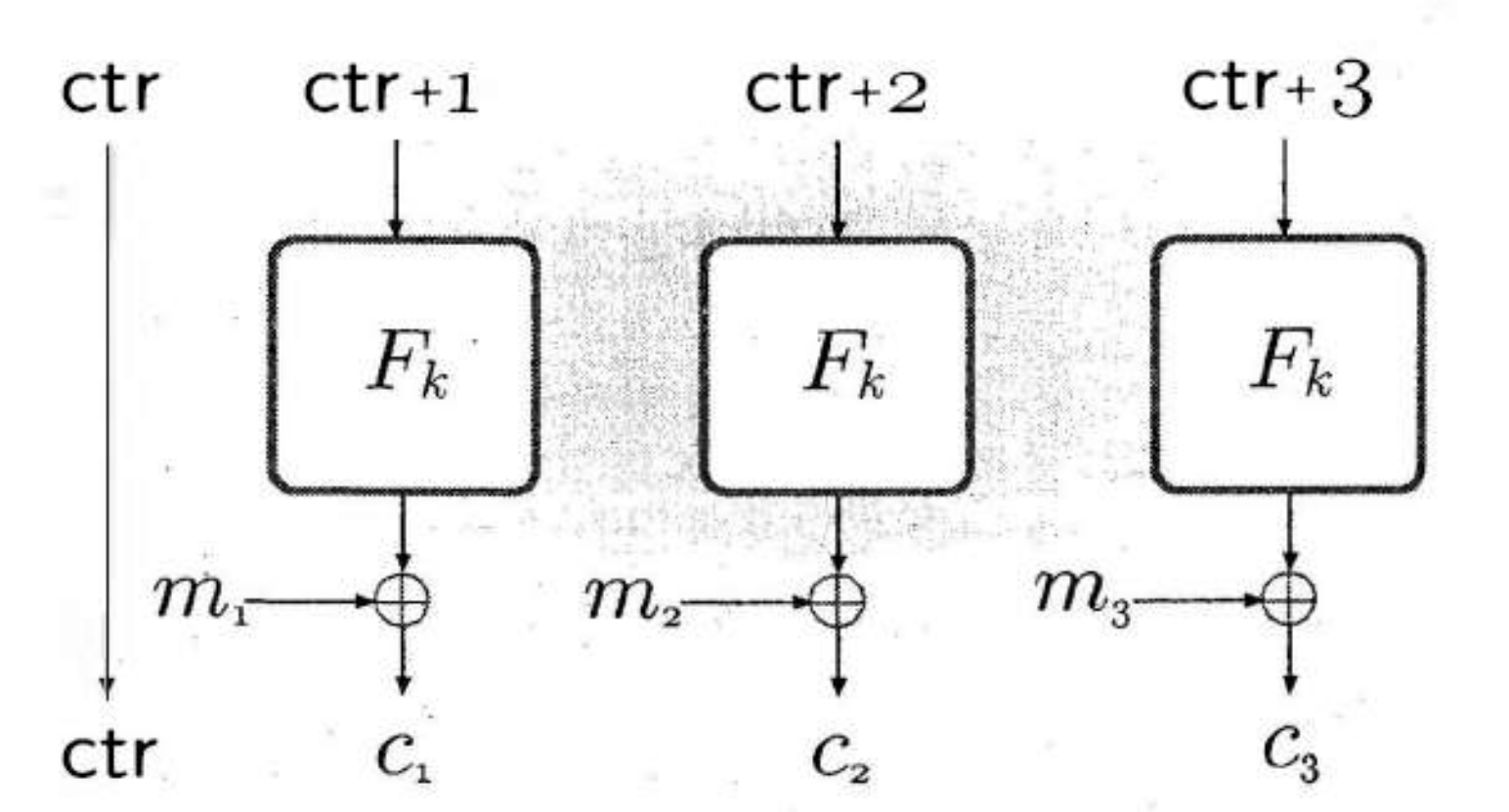
CIPHER BLOCK CHAINING (CBC)



OUTPUT FEEDBACK MODE (OFB)



RANDOMIZED COUNTER MODE



PRF From PRG

CONSTRUCTION 6.24

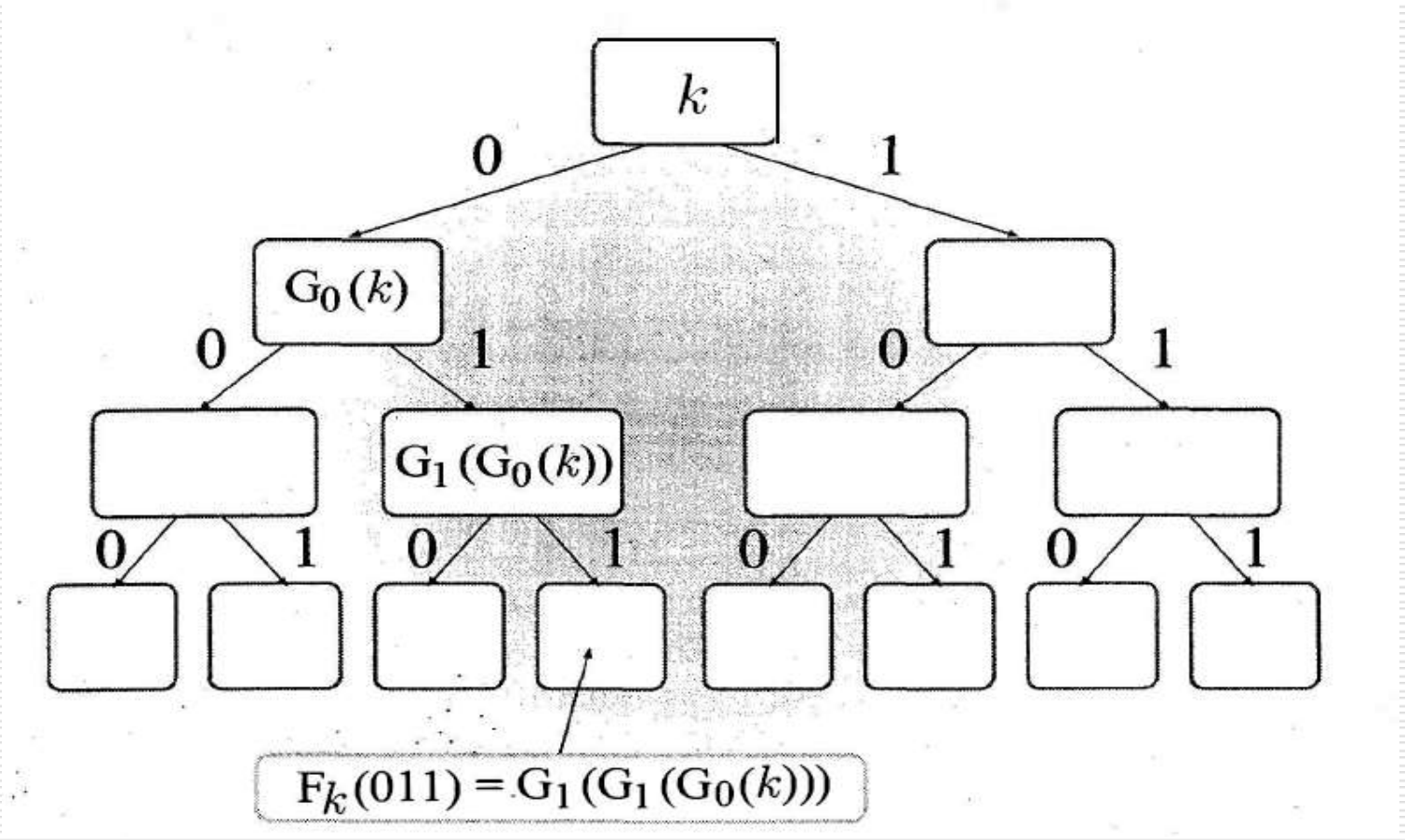
Let G be a pseudorandom generator with expansion factor $\ell(n) = 2n$. Denote by $G_0(k)$ the first half of G's output, and by $G_1(k)$ the second half of G's output. For every $k \in \{0,1\}^n$, define the function $F_k : \{0,1\}^n \to \{0,1\}^n$ as:

$$F_k(x_1 x_2 \cdots x_n) = G_{x_n}\left(\cdots\left(G_{x_2}\left(G_{x_1}(k)\right)\right)\cdots\right).$$

A pseudorandom function from a pseudorandom generator.

THEOREM 6.25 *If G is a pseudorandom generator with expansion factor $\ell(n) = 2n$, then Construction 6.24 is a pseudorandom function.*

PRF from PRG (Contd.)

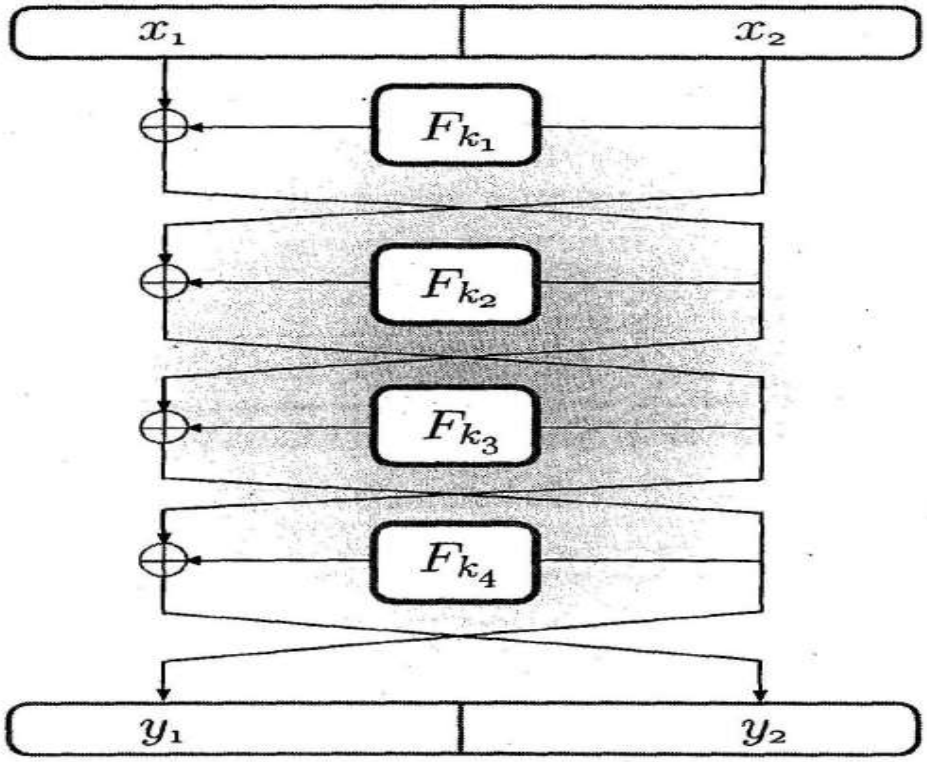


PRFs to Invertible PRFs

- Feistel Networks

$$L_i := R_{i-1} \quad \text{and} \quad R_i := L_{i-1} \oplus f_i(R_{i-1})$$

Multiple Round Feistel Structure



SUMMARY

- CPA attacks is easy in practice today
- CPA-security requires probabilistic encryption
- Pseudorandom Functions (PRF) in the right mode of operation
- Designing PRFs from PRGs

TASK (till date)

- Assuming Discrete Logarithm Problem to be a one-way permutation:
  - Build a provable secure PRG
  - Build a provably secure PRF
  - Use the PRF in some secure mode of operation to obtain a CPA-secure encryption scheme

THANK YOU

Any Questions?

CCA-SECURITY

And

Message Authentication Codes (MAC)

REVIEW

- CPA attacks is easy in practice today
- CPA-security requires probabilistic encryption
- Pseudorandom Functions (PRF) in the right mode of operation
- Designing PRFs from PRGs

PRF From PRG

CONSTRUCTION 6.24

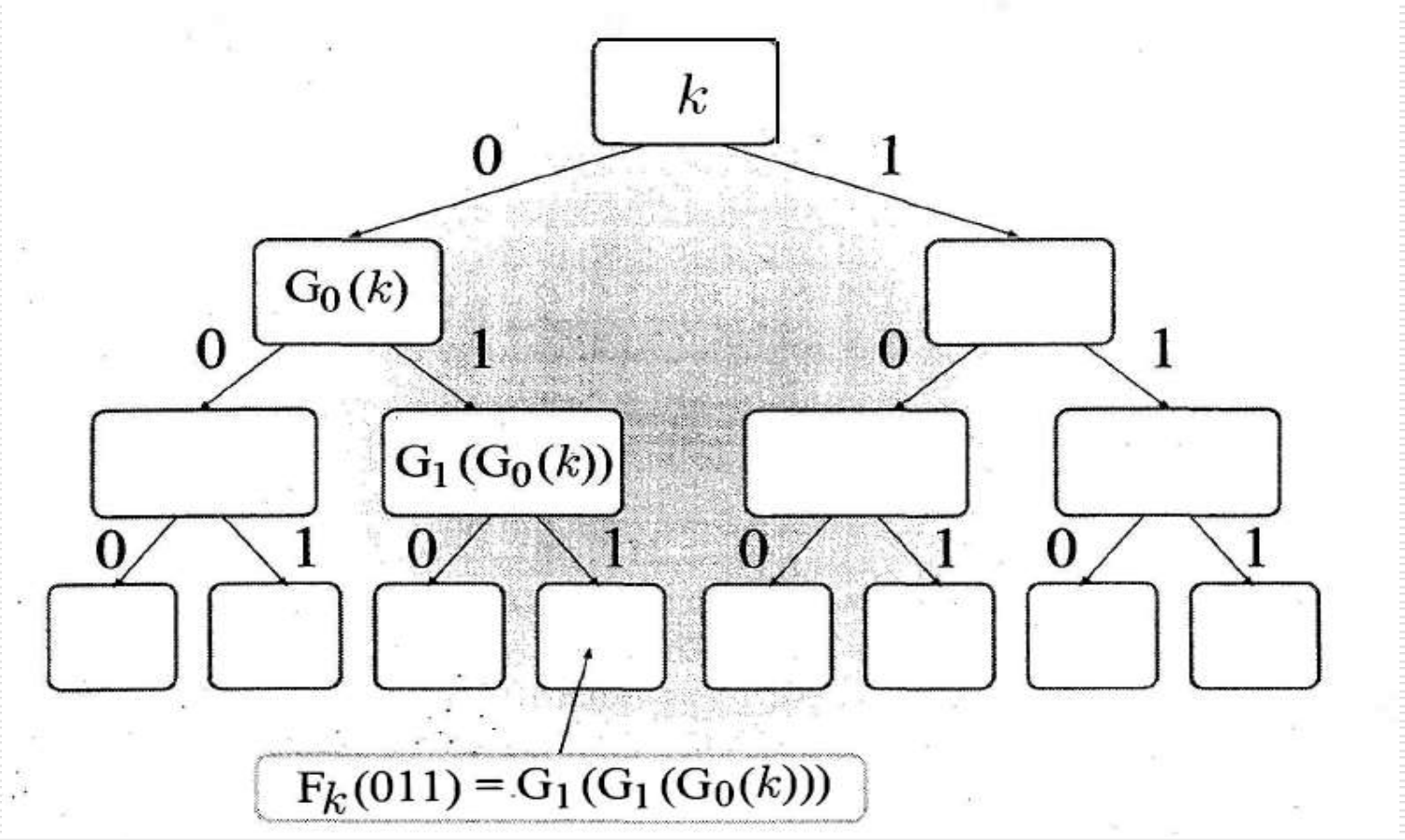
Let G be a pseudorandom generator with expansion factor $\ell(n) = 2n$. Denote by $G_0(k)$ the first half of G's output, and by $G_1(k)$ the second half of G's output. For every $k \in \{0,1\}^n$, define the function $F_k : \{0,1\}^n \to \{0,1\}^n$ as:

$$F_k(x_1 x_2 \cdots x_n) = G_{x_n}\left(\cdots\left(G_{x_2}\left(G_{x_1}(k)\right)\right)\cdots\right).$$

A pseudorandom function from a pseudorandom generator.

THEOREM 6.25 *If G is a pseudorandom generator with expansion factor $\ell(n) = 2n$, then Construction 6.24 is a pseudorandom function.*

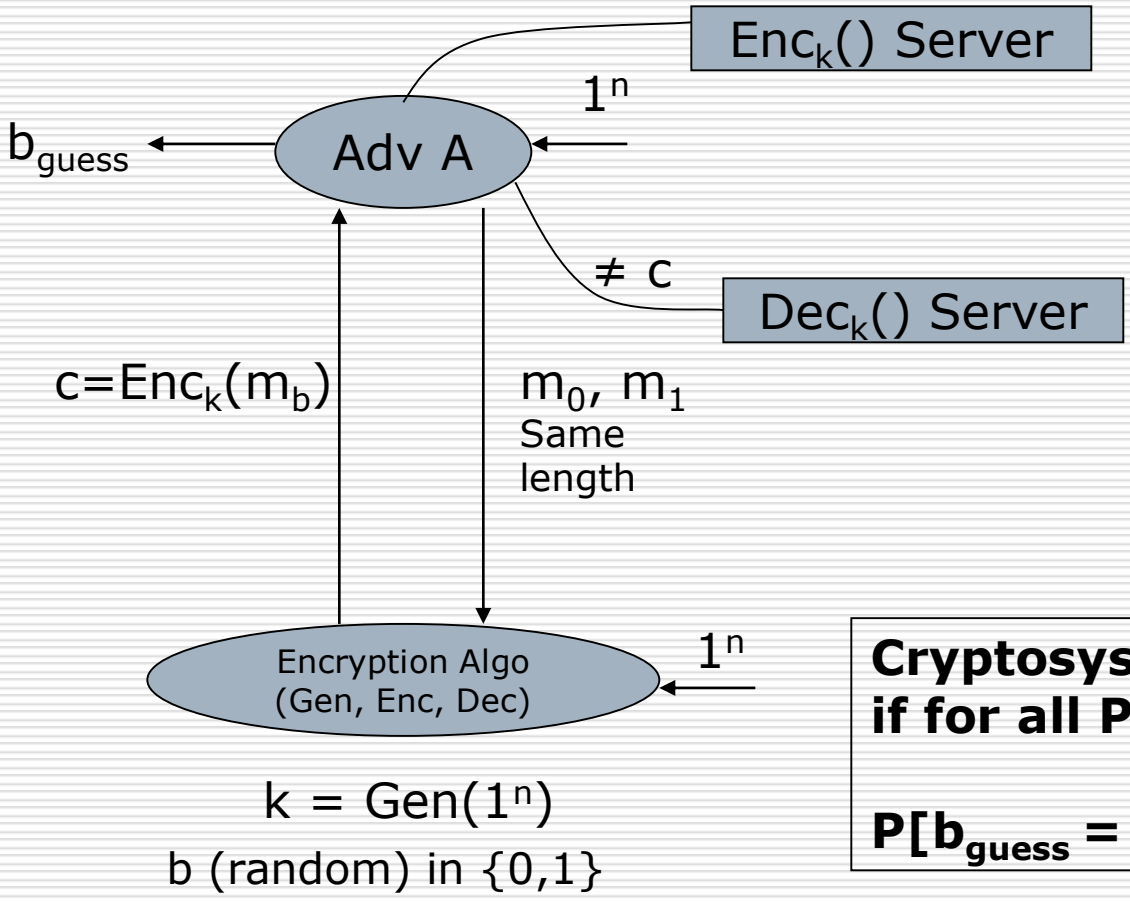
PRF from PRG (Contd.)



CPA-Security Doesn’t Suffice!

CCA-Security

Defining Computational Security Against CHOSEN CIPHERTEXT ATTACK (CCA)



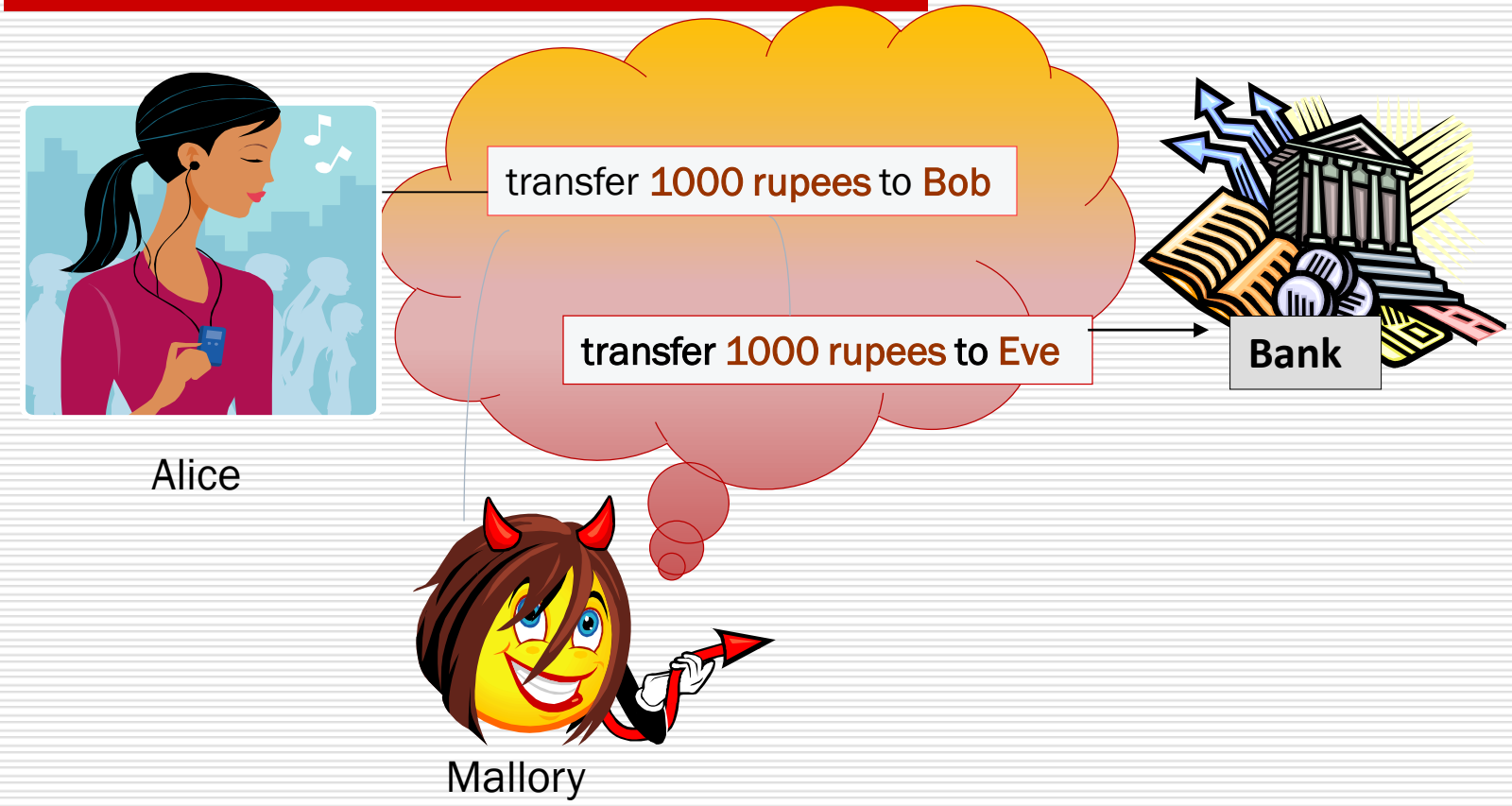
Cryptosystem is CCA-secure if for all PPT adversaries A:

$P[b_{guess} = b] <= ½ + negl(n)$

Message Authentication Codes

Need, Construction and Attacks

Problem of Data Integrity

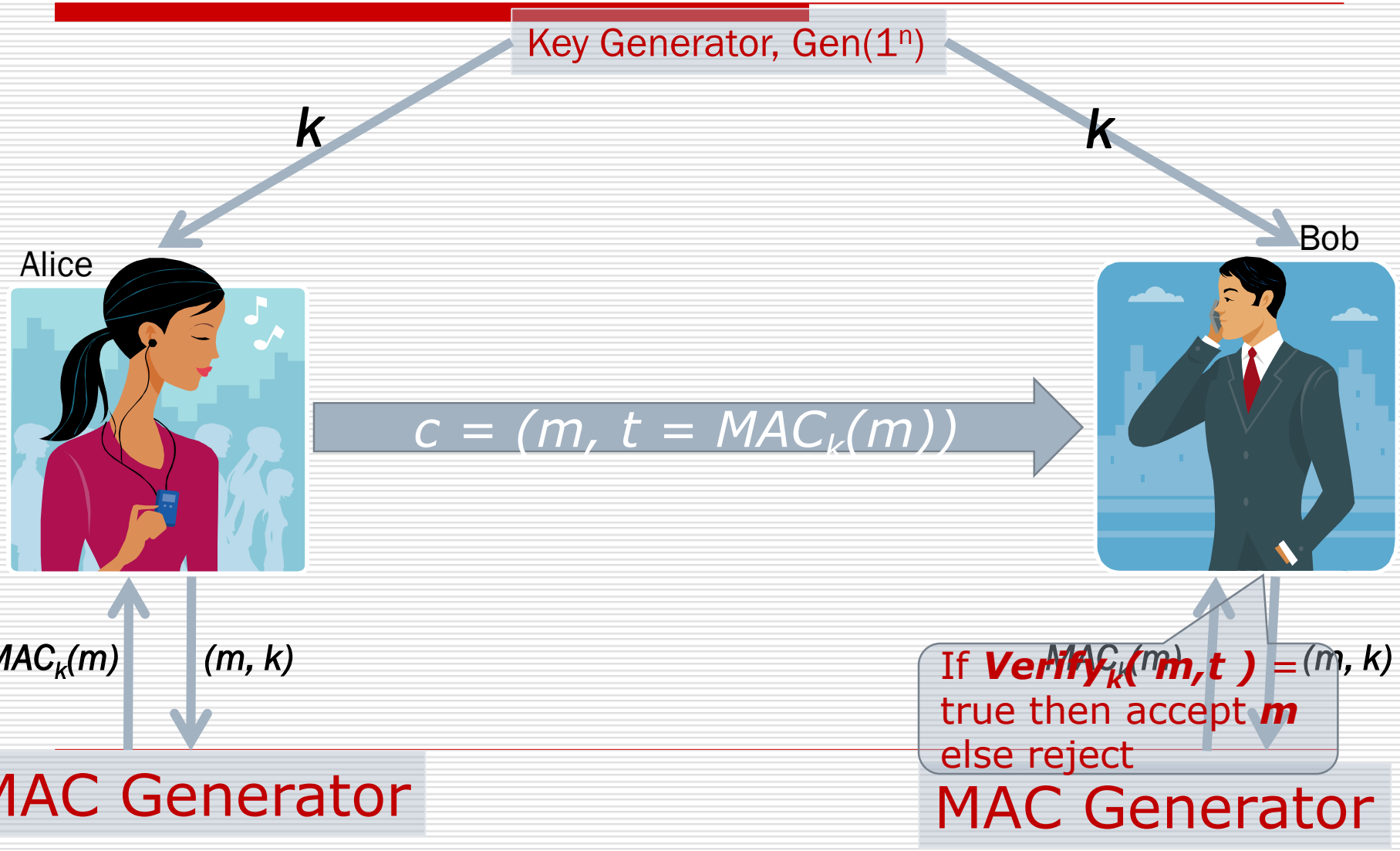


How to prevent such a modification of data?

Does Encryption Guarantee Integrity?

NO!

Data Authentication using a MAC

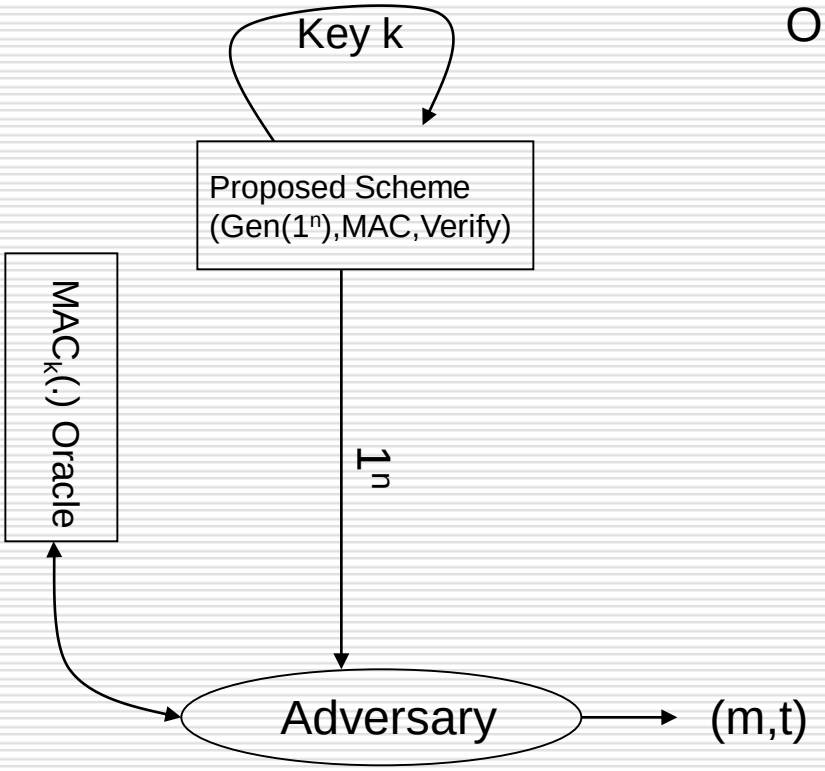


Components of the Authentication Protocol

- A Key Generation Algorithm that returns a secret key ***k***
- A MAC generating algorithm that returns a tag for a given message ***m***. Tag $\boldsymbol{t = MAC_k(m)}$
- A Verification algorithm that returns a bit $\boldsymbol{b = Verify_k(m_1, t_1)}$, given a message $\boldsymbol{m_1}$ and a tag $\boldsymbol{t_1}$
- If the message is not modified then with high probability, the value of ***b*** is true otherwise false

Security of MAC

Mac-Game(n)



Let Q be the set of all queries from Adv to oracle

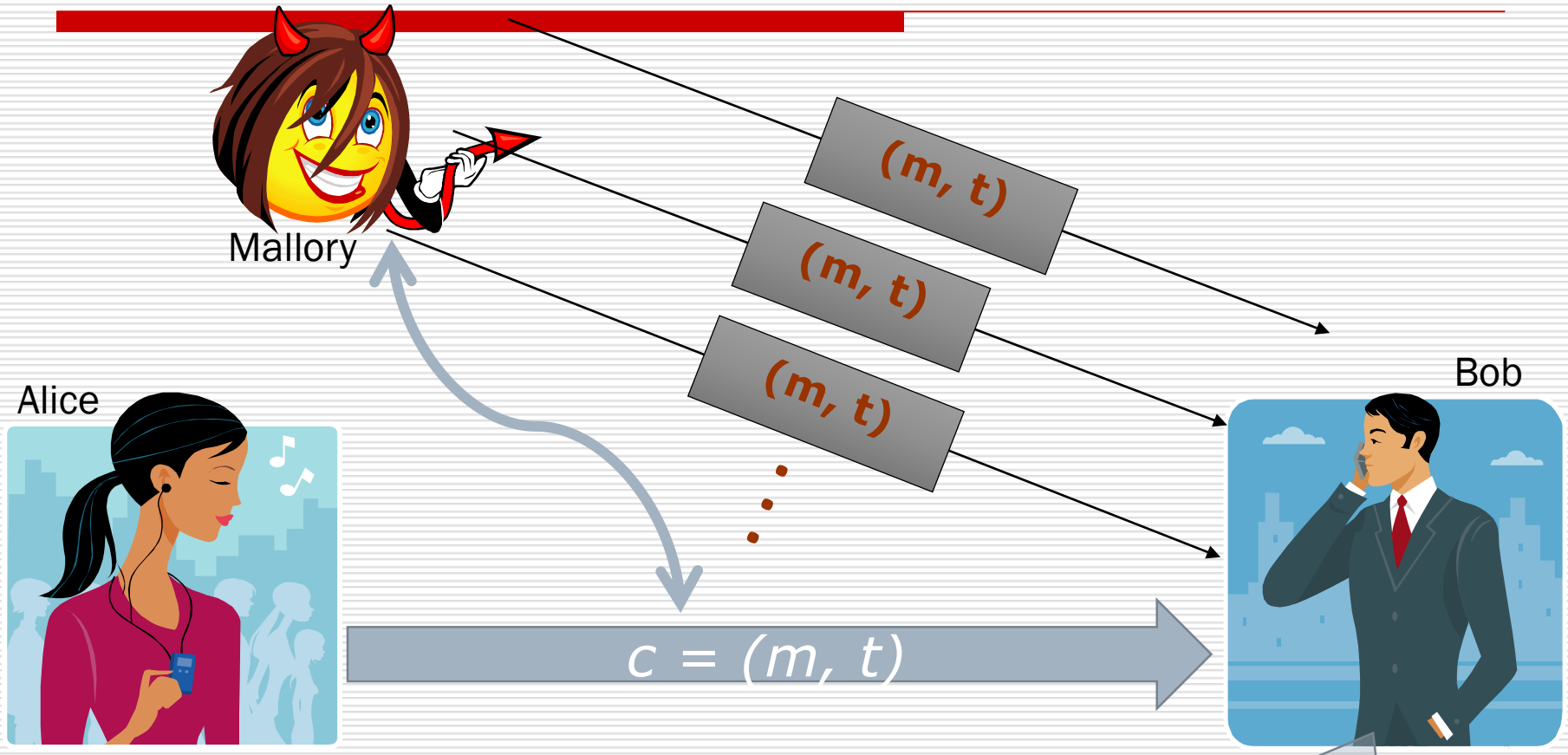
Output of the Game is 1 if and only if:

$Verify_k(m,t) = 1$ and m is not in Q

A message authentication code (Gen,MAC,Verify) is secure if for all probabilistic polynomial-time adversaries ***A***:

$Pr[\text{Mac-Game}(n) = 1] \leq negl(n)$

Replay Attack



Verify has no "*memory*", cannot detect that ***(m, t)*** is not fresh!

How to solve this problem?

Sequence numbers/timestamps can be used

Construction of MAC using a PRF

Gen (1^n) chooses k to be a random n-bit string

$MAC_k(m) = F_k(m) = t$ ***(the tag)***

Verify$_k$ ***(m, t) = Accept***, if and only if $t = F_k(m)$

Theorem: If F is a pseudorandom function, the above scheme is a secure *fixed length* **MAC**

Variable Length MACs (Trial 1)

- Partition the message ***m*** to ***n*** sized blocks $m_1 m_2 \ldots m_q$
- Calculate $MAC_k(m) = MAC_k(m_1 \oplus m_2 \ \ldots \oplus m_q)$
- Is this method Secure?

- NO! We are authenticating the ***xor*** of the message blocks but not the message itself. So we can always choose a message whose ***xor*** value is the same as some other message

Variable Length MACs (Trial 2)

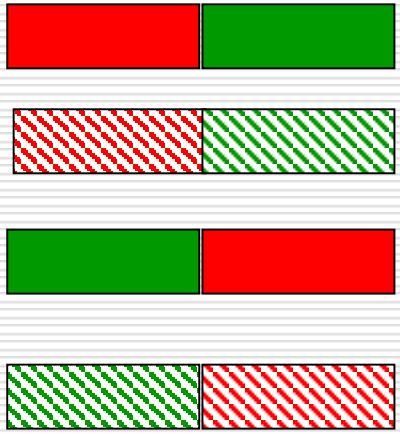
- Concatenate the TAG values of all blocks calculated separately
- But the adversary can rearrange the message blocks and respective tags generating the new message and tag

m:

$t = MAC_k(m)$:

m' = permutation(m):

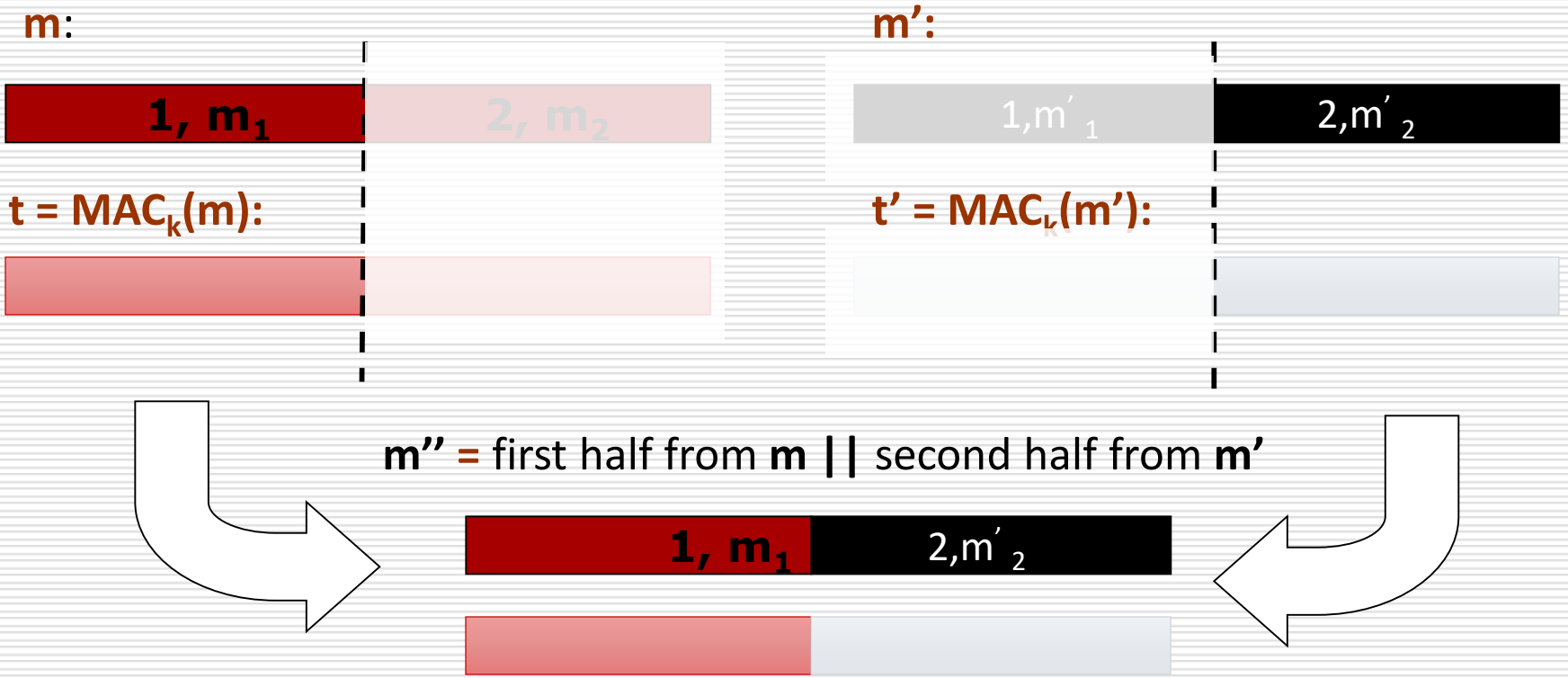
t' = permutation(t):



- Not Secure!

Variable Length MACs (Trial 3)

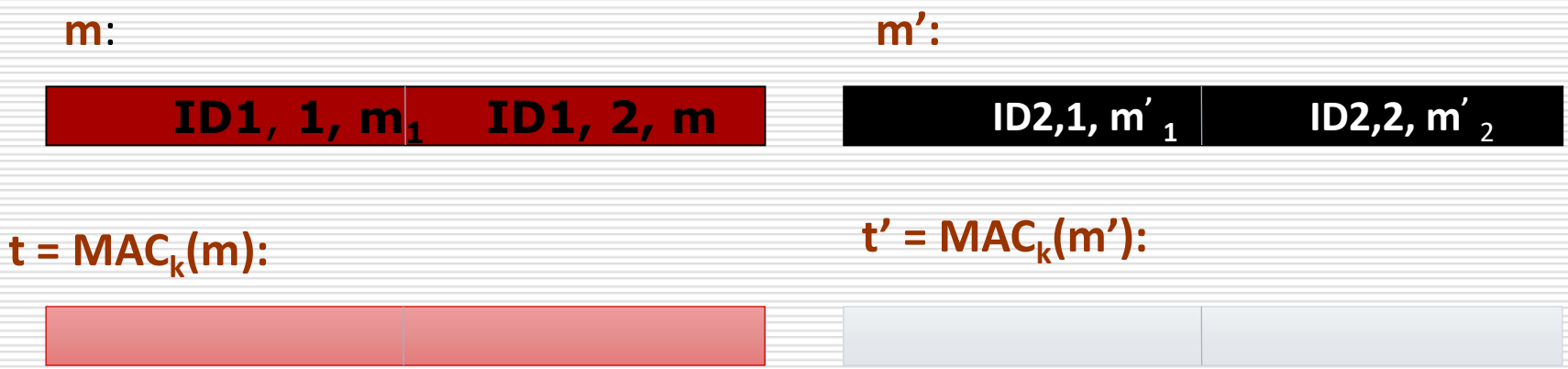
- To prevent the reordering in previous trial, we use sequence numbers. But consider the problem below:



- Then **t''** is a valid tag on **m''**. Not Secure!

Variable Length MACs (Trial 4)

- To prevent the above attack we may use random/unique message ids. So, we may send it as:



- The adversary cannot re-arrange the blocks.

FINAL PROTOCOL (textbook)

CONSTRUCTION 4.5

Let $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ be a fixed-length MAC for messages of length n. Define a MAC as follows:

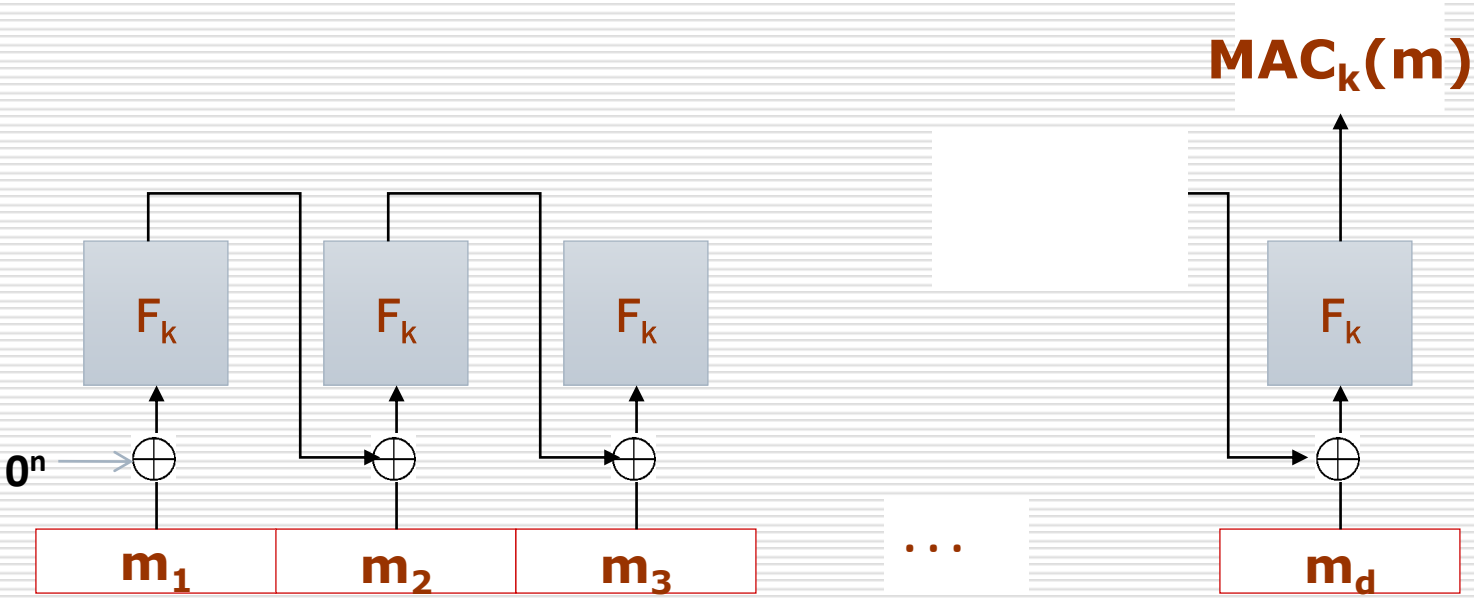
- Gen: this is identical to Gen'.
- Mac: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of length $\ell < 2^{\frac{n}{4}}$, parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Next, choose a random identifier $r \leftarrow \{0,1\}^{n/4}$.

  For $i = 1, \ldots, d$, compute $t_i \leftarrow \mathsf{Mac}'_k(r \| \ell \| i \| m_i)$, where i and ℓ are uniquely encoded as strings of length $n/4$.[4] Finally, output the tag $t := \langle r, t_1, \ldots, t_d \rangle$.
- Vrfy: on input a key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ of length $\ell < 2^{\frac{n}{4}}$, and a tag $t = \langle r, t_1, \ldots, t_{d'} \rangle$, parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Output 1 if and only if $d' = d$ and $\mathsf{Vrfy}'_k(r \| \ell \| i \| m_i, t_i) = 1$ for $1 \leq i \leq d$.

A variable-length MAC from any fixed-length MAC.

Cipher Block Chaining MAC (CBC-MAC)

CBC-MAC Construction



But again, CBC-MAC is secure for fixed length messages but not for variable length messages! Why?

Problem with Variable Length CBC-MAC

Mallory chooses these two messages that Alice has sent

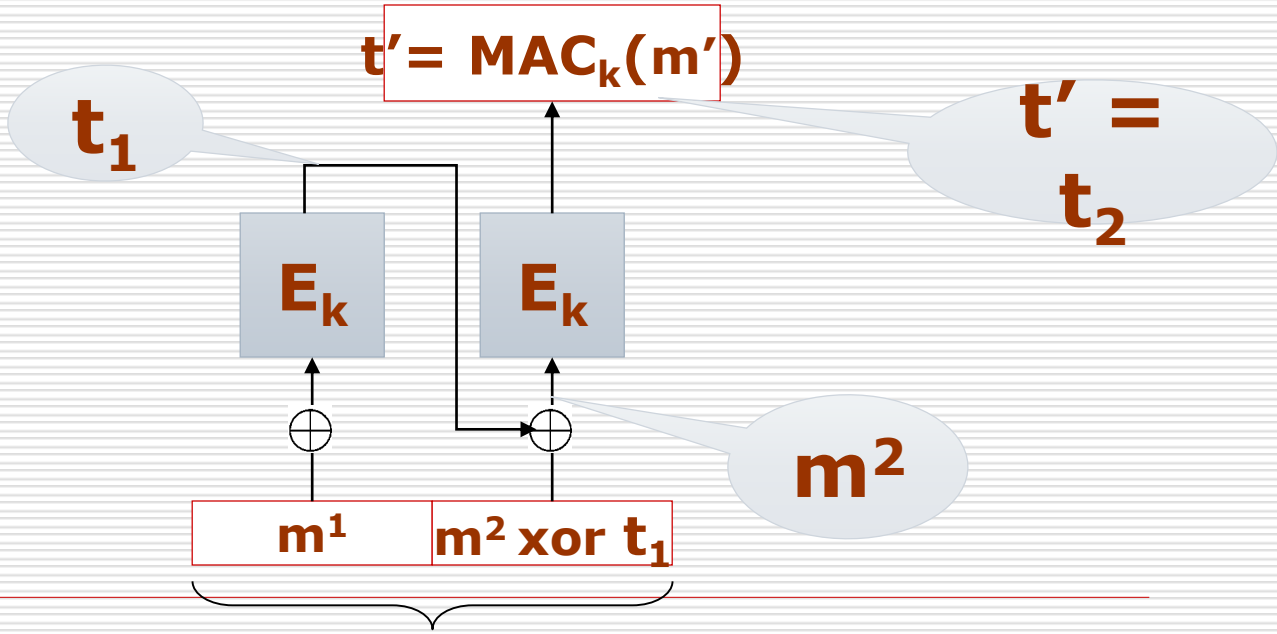
m^1 $t_1 = MAC_k(m^1)$

m^2 $t_2 = MAC_k(m^2)$

Problem with Variable Length CBC-MAC

Mallory has two message pairs as shown above. She now can construct a new message shown below

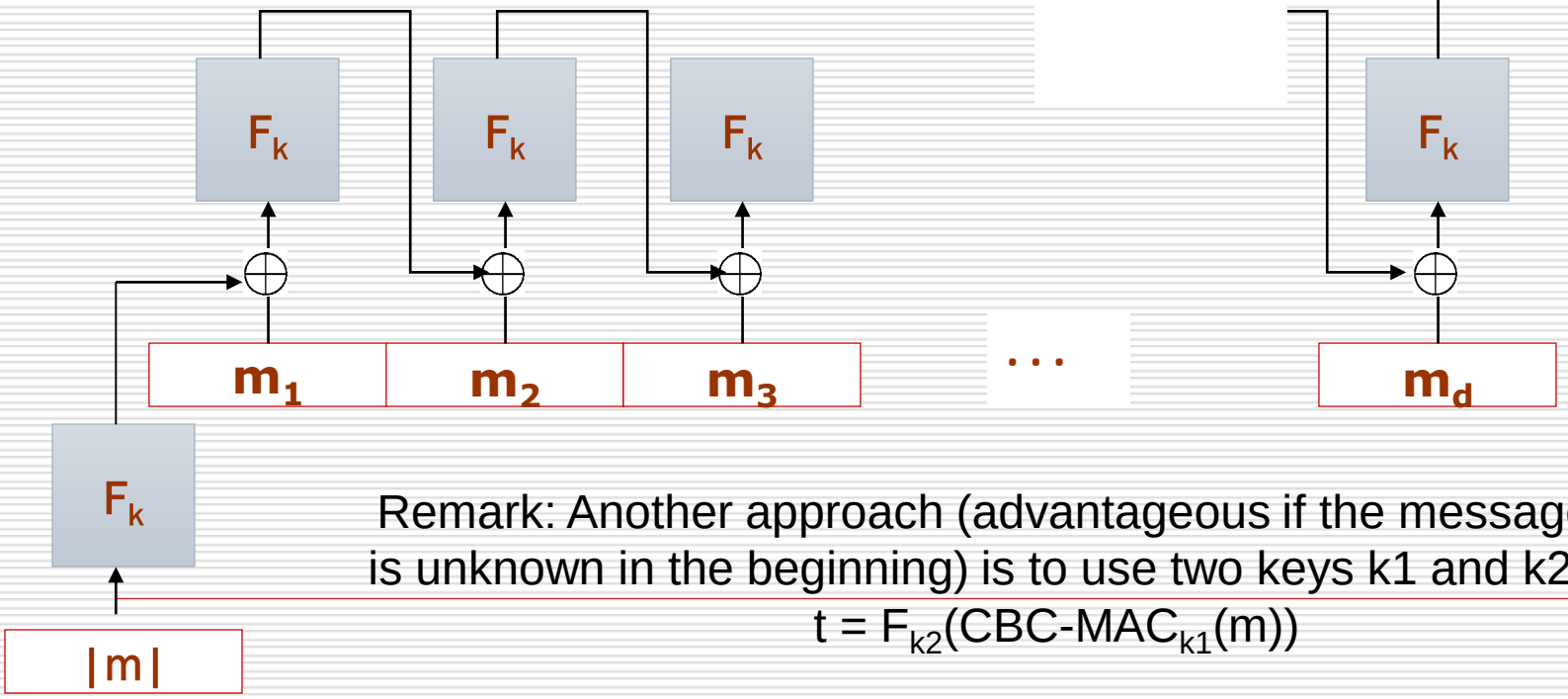
Mallory can now send this new valid pair ***(m′, t′)*** to Bob



CBC-MAC Construction

A secure CBC-MAC for variable length messages

Prepend length of the message |m| (encoded as an n-bit string) to m and then compute the tag (appending the length to the end is not secure!)



Remark: Another approach (advantageous if the message length is unknown in the beginning) is to use two keys k1 and k2 and set

$t = F_{k2}(\text{CBC-MAC}_{k1}(m))$

CCA-SECURITY

From CPA-security
and
Secure MAC

ENCRYPT-THEN-AUTHENTICATE

$$c = (r, F_{k1}(r)+m), MAC_{k2}(r, Fk1(r)+m)$$

CCA-Secure Encryption (textbook)

CONSTRUCTION 4.19

Let $\Pi_E = (\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme and let $\Pi_M = (\mathsf{Gen}_M, \mathsf{Mac}, \mathsf{Vrfy})$ be a message authentication code. Define an encryption scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- Gen': on input 1^n, run $\mathsf{Gen}_E(1^n)$ and $\mathsf{Gen}_M(1^n)$ to obtain keys k_1, k_2, respectively.
- Enc': on input a key (k_1, k_2) and a plaintext message m, compute $c \leftarrow \mathsf{Enc}_{k_1}(m)$ and $t \leftarrow \mathsf{Mac}_{k_2}(c)$ and output the ciphertext $\langle c, t \rangle$
- Dec': on input a key (k_1, k_2) and a ciphertext $\langle c, t \rangle$, first check whether $\mathsf{Vrfy}_{k_2}(c, t) \stackrel{?}{=} 1$. If yes, then output $\mathsf{Dec}_{k_1}(c)$; if no, then output $\perp$.

A CCA-secure private-key encryption scheme.

TASKS (till date)

Assuming Discrete Logarithm Problem to be a one-way

- **Build a provable secure PRG**
- **Build a provably secure PRF**
- **Use the PRF in some secure mode of operation to obtain a CPA-secure encryption scheme**
- **Use the PRF to build a secure MAC**
- **Use CPA-security and secure MAC to design a state-of-the-art provably CCA-secure encrytpion scheme**

THANK YOU

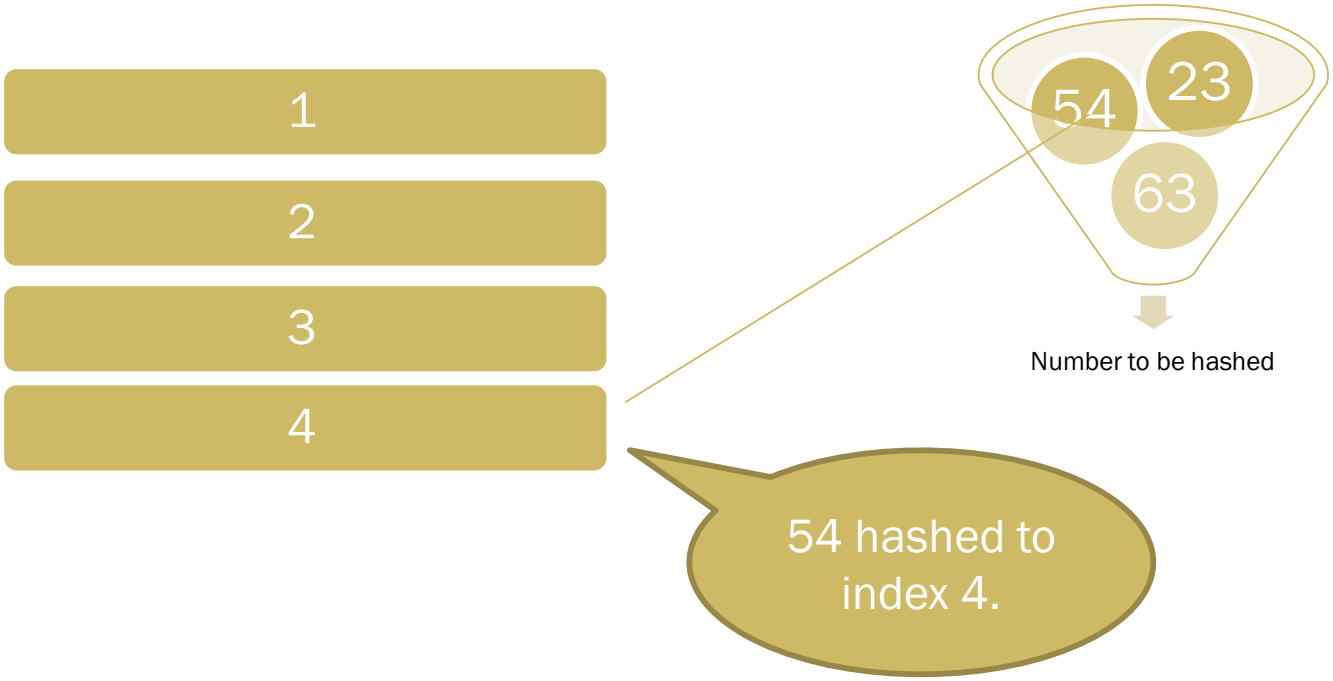
Any Questions?

Collision Resistant Hash Functions

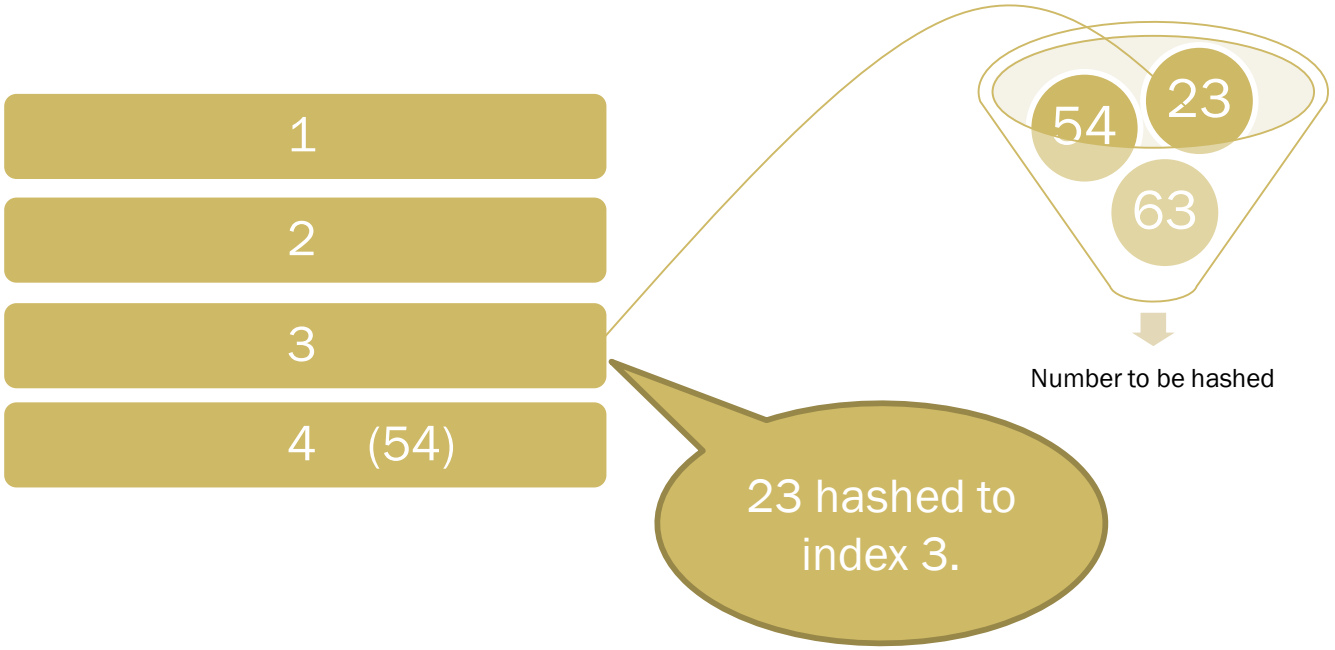
Hash Functions

- Traditionally, hash functions take arbitrary length strings and compress them into shorter strings
- Classically used in data structures for improved look-up times in storage/retrieval
- Collisions for the hash function H are distinct inputs x and y such that H(x) = H(y)

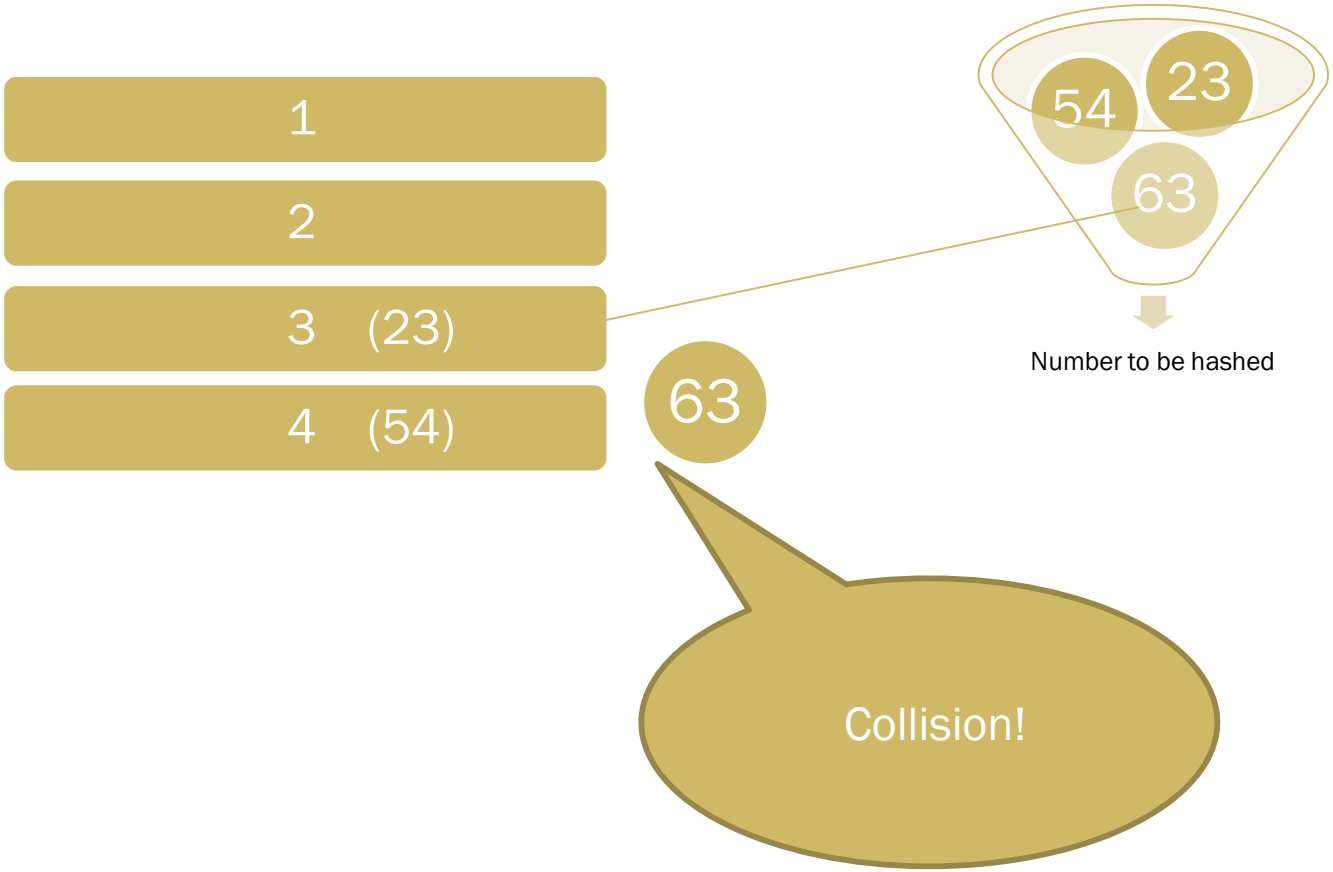
An Example



An Example



An Example

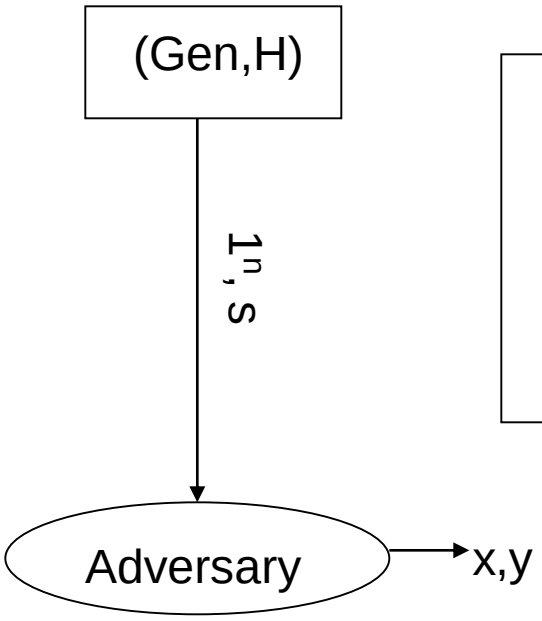


Collision Resistance

- A function H is *collision resistant* if it is infeasible for any probabilistic polynomial-time algorithm to find a collision in H
- We deal with a family of functions indexed by s, $H^s(x) = H(s,x)$
- A **hash function** is a pair of algorithms (Gen, H) where $Gen(1^n)$ outputs the index s (for choosing H^s)
- If H^s is defined only for inputs x of a certain length, we say it is a **fixed length** hash function

Defining Collision Resistant Hash Function

Hash-game



The output of Hash-game is 1 if and only if $x \neq y$ and $H^s(x) = H^s(y)$

A hash function (Gen,H) is collision resistant if for all probabilistic polynomial time adversaries A:

Pr[Output of Hash-game = 1] ≤ negl(n)

Birthday Attack

We can find a collision on a hash function by hashing random numbers until these hashed values have a collision

Relies on 'Birthday Paradox'

Theorem: If we pick independent random numbers in [1,2,...,N] with uniform distribution $\theta\sqrt{N}$ times, we get at least one number twice with probability tending to

$$(1-e^{-(\theta/2)*\theta})$$

For N = 365 and θ = 1.2 we get prob ~ 50%

Birthday Paradox: If there are 23 people,with probability at least ½ at least two of them have the same birthday

And for θ = 2.1 we get prob ~ 90% !

TEXTBOOK VERSION

LEMMA A.10 *Fix a positive integer N, and say $q \leq \sqrt{2N}$ elements $y_1, \ldots, y_q$ are chosen uniformly and independently at random from a set of size N. Then the probability that there exist distinct i, j with $y_i = y_j$ is at least $\frac{q(q-1)}{4N}$. That is,*

$$\mathsf{coll}(q, N) \geq \frac{q(q-1)}{4N}.$$

PROOF

- NoColl_i : Denotes the event of NO COLLISION upto i

$$\Pr[\text{NoColl}_q] = \Pr[\text{NoColl}_1] \cdot \Pr[\text{NoColl}_2 \mid \text{NoColl}_1] \cdots \Pr[\text{NoColl}_q \mid \text{NoColl}_{q-1}].$$

$$\Pr[\text{NoColl}_1] = 1 \qquad \Pr[\text{NoColl}_{i+1} \mid \text{NoColl}_i] = 1 - \frac{i}{N}$$

$$\Pr[\text{NoColl}_q] = \prod_{i=1}^{q-1} \left(1 - \frac{i}{N}\right)$$

PROPOSITION A.4 *For all x with $0 \le x \le 1$ it holds that*

$$e^{-x} \le 1 - \left(1 - \frac{1}{e}\right) \cdot x \le 1 - \frac{x}{2}.$$

$$\Pr[\text{NoColl}_q] \le \prod_{i=1}^{q-1} e^{-i/N} = e^{-\sum_{i=1}^{q-1}(i/N)} = e^{-q(q-1)/2N}$$

$$\Pr[\text{Coll}] = 1 - \Pr[\text{NoColl}_q] \ge 1 - e^{-q(q-1)/2N} \ge \frac{q(q-1)}{4N}$$

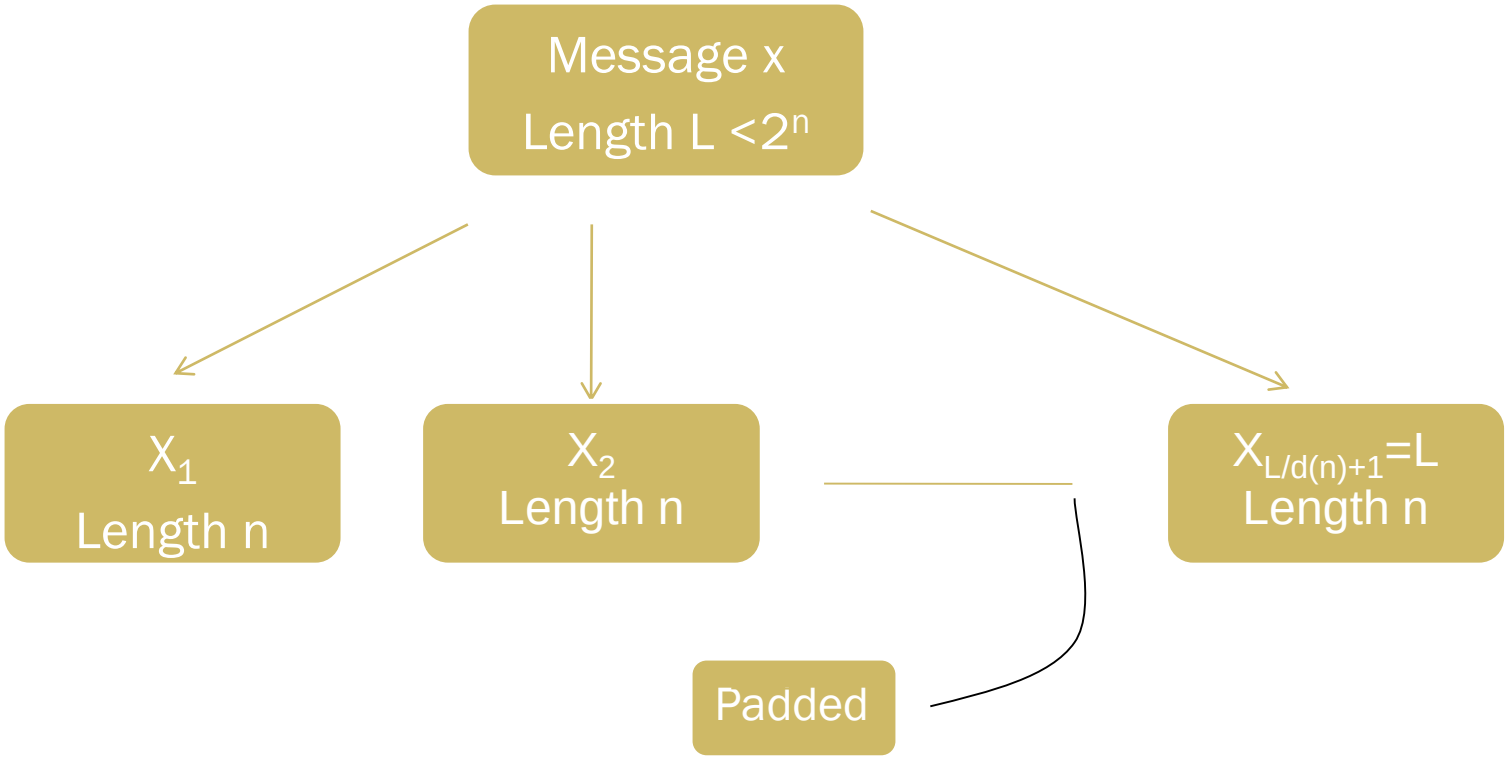
CONSTRUCTING COLLISION RESISTANT HASH FUNCTIONS

Step 1 (*Merkle-Damgard*): From Fixed Length Hash to Arbitrary Hash

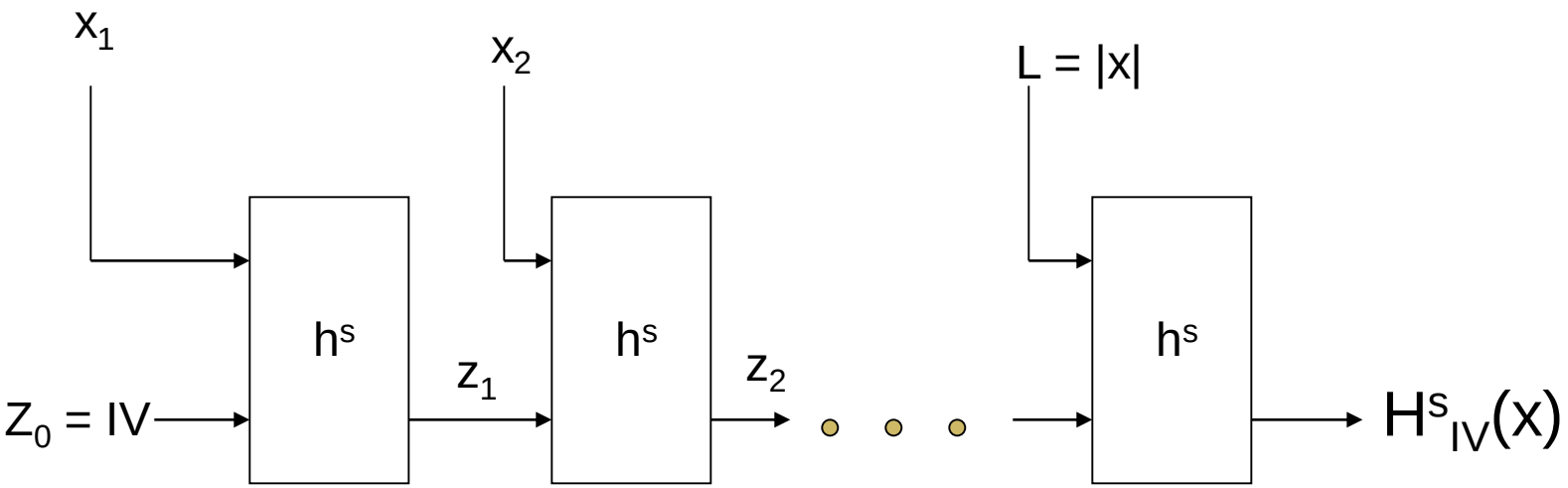
Step 2: From DLP to fixed length Hashing

Merkle Damgard Transform

Constructing hash functions $H^s(x)$ from fixed length hash functions (h^s) with inputs of length 2n and output length n



Merkle Damgard Transform



Theorem: If (Gen,h) is a fixed length collision resistant hash function, then (Gen, H) is a collision resistant hash function

TEXTBOOK VERSION

CONSTRUCTION 4.13

Let (Gen, h) be a fixed-length collision-resistant hash function for inputs of length $2\ell(n)$ and with output length $\ell(n)$. Construct a variable-length hash function (Gen, H) as follows:

- Gen: remains unchanged.
- H: on input a key s and a string $x \in \{0,1\}^*$ of length $L < 2^{\ell(n)}$, do the following (set $\ell = \ell(n)$ in what follows):

  1. Set $B := \lceil \frac{L}{\ell} \rceil$ (i.e., the number of blocks in x). Pad x with zeroes so its length is a multiple of ℓ. Parse the padded result as the sequence of ℓ-bit blocks $x_1, \ldots, x_B$. Set $x_{B+1} := L$, where L is encoded using exactly ℓ bits.
  2. Set $z_0 := 0^\ell$.
  3. For $i = 1, \ldots, B+1$, compute $z_i := h^s(z_{i-1} \| x_i)$.
  4. Output z_{B+1}.

The Merkle-Damgård transform.

SECURITY PROOF

1. *Case 1:* $L \neq L'$. In this case, the last step of the computation of $H^s(x)$ is $z_{B+1} := h^s(z_B \| L)$ and the last step of the computation of $H^s(x')$ is $z'_{B'+1} := h^s(z'_{B'} \| L')$. Since $H^s(x) = H^s(x')$ it follows that $h^s(z_B \| L) = h^s(z'_{B'} \| L')$. However, $L \neq L'$ and so $z_B \| L$ and $z'_{B'} \| L'$ are two different strings that collide for h^s.

2. *Case 2:* $L = L'$. Note this means that $B = B'$ and $x_{B+1} = x'_{B+1}$. Let z_i and z'_i be the intermediate hash values of x and x' during the computation of $H^s(x)$ and $H^s(x')$, respectively. Since $x \neq x'$ but $|x| = |x'|$, there must exist at least one index i (with $1 \leq i \leq B$) such that $x_i \neq x'_i$. Let $i^* \leq B + 1$ be the *highest* index for which it holds that $z_{i^*-1} \| x_{i^*} \neq z'_{i^*-1} \| x'_{i^*}$. If $i^* = B + 1$ then $z_B \| x_{B+1}$ and $z'_B \| x'_{B+1}$ are two different strings that collide for h^s because

$$h^s(z_B \| x_{B+1}) = z_{B+1} = H^s(x) = H^s(x') = z'_{B+1} = h^s(z'_B \| x'_{B+1}).$$

If $i^* \leq B$, then maximality of i^* implies $z_{i^*} = z'_{i^*}$. Thus, once again, $z_{i^*-1} \| x_{i^*}$ and $z'_{i^*-1} \| x'_{i^*}$ are two different strings that collide for h^s.

BULIDING MACs USING HASHING

Keyed Hashing

HMAC: A Message Authentication Code

HMAC is the current industry standard as CBC-MAC is deemed to be slow

(Gen,h): A fixed length hash function

(Gen,H): Hash function after applying MD transform to (Gen,h)

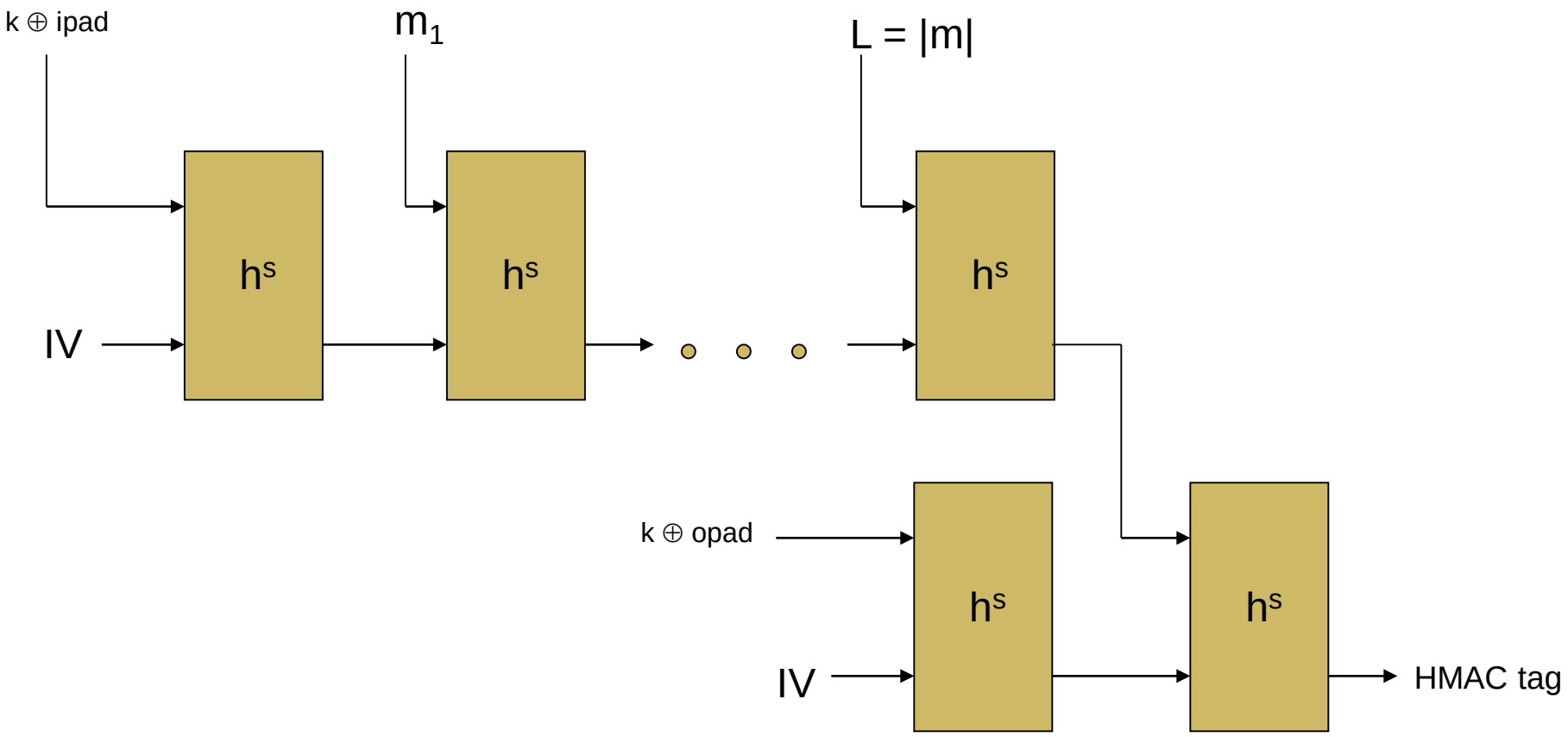
Fixed constants: *IV*, *opad* and *ipad*

HMAC tag for m = $H^s_{IV}((k \oplus opad) \,||\, H^s_{IV}((k \oplus ipad) \,||\, m))$

opad: 0x36 repeated as many times as needed

ipad: 0x5C repeated as many times as needed

HMAC Construction



Collision Resistant Hash Functions in Practice

- Popular practical constructions
  - MD5 (broken in 2004, should no longer be used)
  - SHA-1, SHA-2 etc. (uses Merkle-Damgard transform) and others
- Theoretical Constructions
  - Based on hardness of the discrete logarithm problem

A Fixed Length Hash Function

Let P be a polynomial time algorithm that on input 1^n outputs a cyclic group G of order q (length of q is n) and generator g

Gen: Run $P(1^n)$ obtain (G,q,g); select uniformly at random an element h from G; Output **s = (G,q,g,h)**

H: On input x_1 and x_2 (both in the range 0 to q-1), output

$$H^s(x_1,x_2) = g^{x_1} h^{x_2}$$

Theorem: If discrete logarithm problem is hard then the above is a fixed length collision resistant hash function

TEXTBOOK VERSION

CONSTRUCTION 7.72

Let $\mathcal{G}$ be as described in the text. Define a fixed-length hash function (Gen, H) as follows:

- Gen: on input 1^n, run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$ and then select $h \leftarrow \mathbb{G}$. Output $s := \langle \mathbb{G}, q, g, h \rangle$ as the key.
- H: given a key $s = \langle \mathbb{G}, q, g, h \rangle$ and input $(x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$, output $H^s(x_1, x_2) := g^{x_1} h^{x_2}$.

A fixed-length hash function.

SECURITY PROOF

THEOREM 7.73 *If the discrete logarithm problem is hard relative to $\mathcal{G}$, then Construction 7.72 is a fixed-length collision-resistant hash function*

$$H^s(x_1, x_2) = H^s(x_1', x_2') \Rightarrow g^{x_1} h^{x_2} = g^{x_1'} h^{x_2'}$$
$$\Rightarrow g^{x_1 - x_1'} = h^{x_2' - x_2}.$$

$$\Delta \stackrel{\text{def}}{=} x_2' - x_2$$

$$g^{(x_1 - x_1') \cdot \Delta^{-1}} = \left(h^{x_2' - x_2}\right)^{[\Delta^{-1} \bmod q]} = h^{[\Delta \cdot \Delta^{-1} \bmod q]} = h^1 = h.$$

TASKS (till date)

Assuming Discrete Logarithm Problem to be a one-way

- Build a provably secure PRG
- Build a provably secure PRF from PRG
- Use the PRF in some secure mode of operation to obtain a CPA-secure encryption scheme
- Use the PRF to build a secure MAC
- Use CPA-security and secure MAC to design a state-of-the-art provably CCA-secure encrytpion scheme

- Using DLP to build fixed-length Collision Resistant Hash function
- Using MD transform to obtain a (provably secure) Collision Resistant Hash Function
- Using a collision resistant hash function to build H-MACs

THANK YOU

Any Questions?

PUBLIC-KEY REVOLUTION

And
Key Distribution

Issues With Private-Key Cryptography

- Private-key cryptography requires a shared secret key
- Initial key sharing requires a secure channel (**Key Distribution Problem**)
- Not scalable (**Key Management Problem**)
- What about **Open Systems**?

First-Cut Solution

- Use a Key "Controller"
- Issues:
  - Not "fully" secure
  - Large secure-storage required
  - Still focused on *Closed Systems*!

Partial Solution

- Key Distribution Centers (KDC)
  - Key distribution is simplified
  - Solves the key storage problem using "just-in-time" *session keys*
  - New issue: *single-point-of-failure* versus *multiple-points-of-trust*
  - Open systems are still *open!*

PUBLIC-KEY CRYPTOGRAPHY

If the encryption *key* is public,
how (the hell!) can it be *secure?*

In a nutshell:

Full *information*, but no *knowledge*!

Ease of Computation Depends on the Representation

It also depends on the operation!

Ease/Speed of Operation Depends on The Representation

- viii * xvi = cxxviii
- 8 * 16 = 128
- $2^3 * 2^4 = 2^7$

- viii + xvi = xxiv
- 8 + 16 = 24
- $2^3 + 2^4 = 2^3.3$

- viii < ix is true
- 8 < 9 is true
- $2^3 < 3^2$ is true

Is There a Representation Where all Common Operations are FAST?

Not Easy!

Top Three Most Frequent Operations

- Comparison (<)
- Addition (+)
- Multiplication (*)

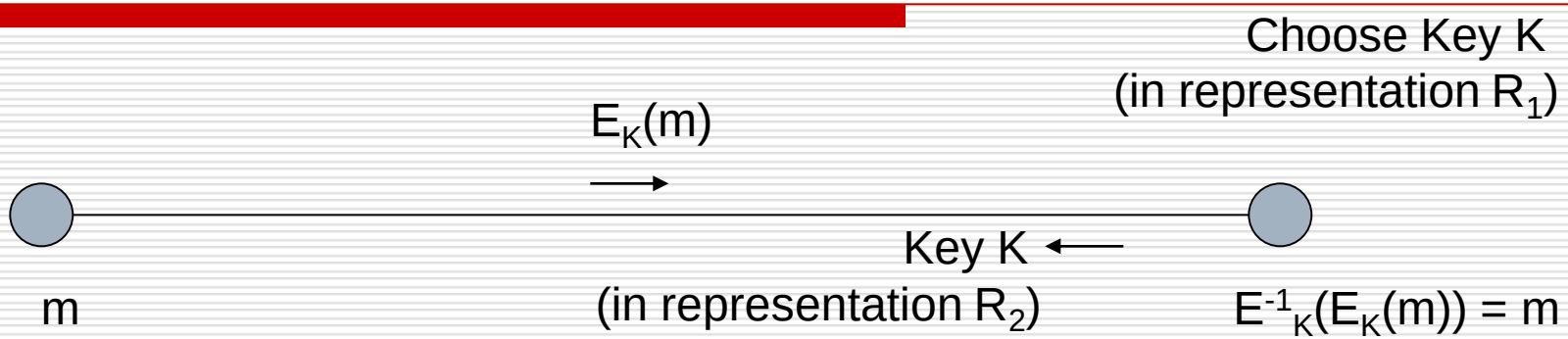
Why is the Decimal System Popular?

	Addition	Multiplication	Comparison
ROMAN	SLOW	SLOW	SLOW
DECIMAL	FAST	MEDIUM	FAST
PRIME PRODUCT	SLOW	FAST	SLOW
RESIDUE SYSTEM	FAST	FAST	MEDIUM

Slowness is ADVANTAGEOUS too!

Public Key Cryptography

Public-Key Revolution



In Representation R_2

- **Operation E_K is FAST**
- **Operation E_K^{-1} is VERY SLOW**

In Representation R_1

- **Operation E_K^{-1} is FAST**

Diffie-Hellman Key Exchange

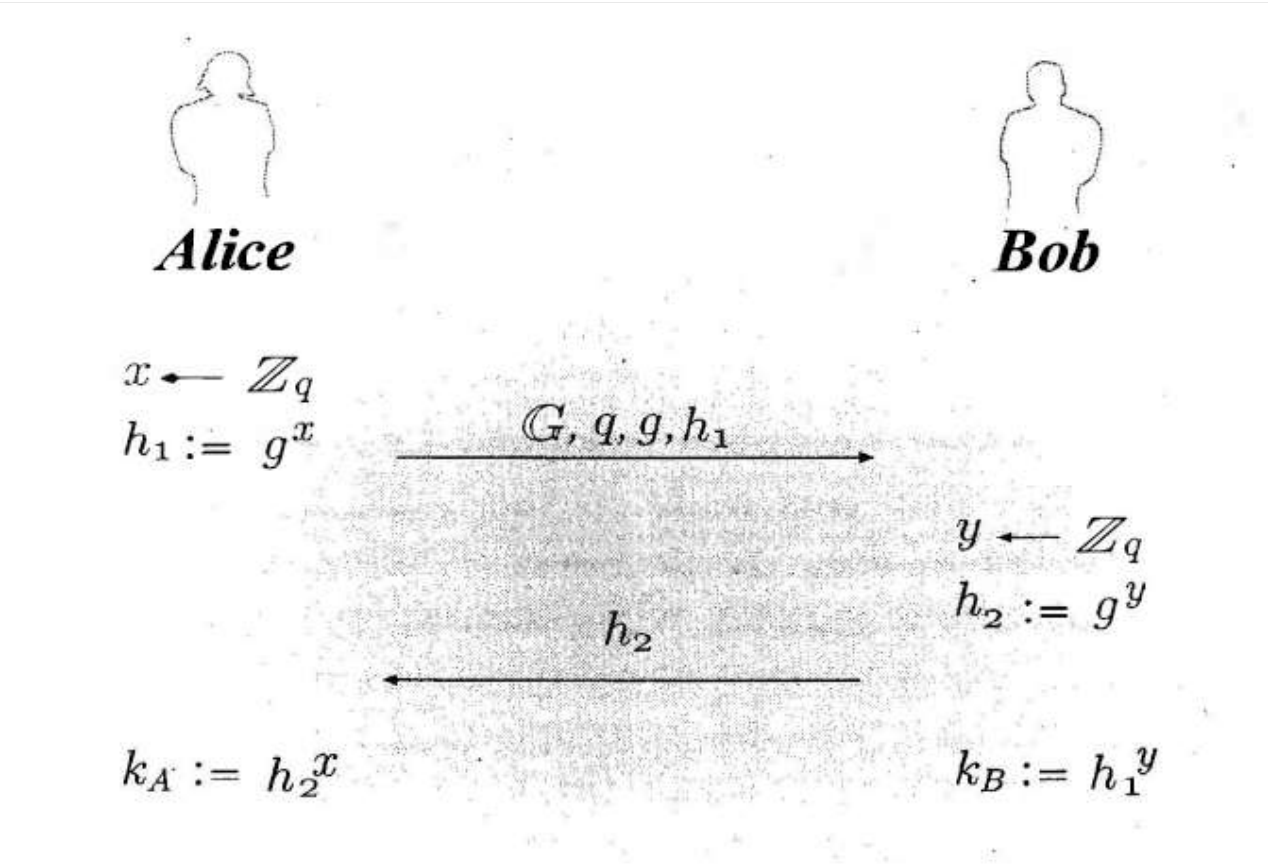
- Defining the key-exchange problem:

The key-exchange experiment $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$**:**

1. *Two parties holding* 1^n *execute protocol* Π. *This execution of the protocol results in a transcript* trans *containing all the messages sent by the parties, and a key* k *that is output by each of the parties.*
2. *A random bit* $b \leftarrow \{0,1\}$ *is chosen. If* $b = 0$ *then choose* $\hat{k} \leftarrow \{0,1\}^n$ *uniformly at random, and if* $b = 1$ *set* $\hat{k} := k$.
3. $\mathcal{A}$ *is given* trans *and* $\hat{k}$, *and outputs a bit* b'.
4. *The output of the experiment is defined to be 1 if* $b' = b$, *and 0 otherwise. (In case* $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1$, *we say that* $\mathcal{A}$ succeeds.)

For all PPT adversaries A: $\Pr\left[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n)$

The PROTOCOL



The DDH Assumption

DEFINITION 7.60 *We say that* the DDH problem is hard relative to $\mathcal{G}$ *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]\right| \leq \mathsf{negl}(n),$$

where in each case the probabilities are taken over the experiment in which $\mathcal{G}(1^n)$ *outputs* $(\mathbb{G}, q, g)$*, and then random* $x, y, z \in \mathbb{Z}_q$ *are chosen.*

RSA Public/Secret Key Generation

And

RSA Assumption!

RSA Key Generation Algorithm

ALGORITHM 7.47

GenRSA — **high-level outline**

Input: Security parameter 1^n

Output: N, e, d as described in the text

$(N, p, q) \leftarrow \mathsf{GenModulus}(1^n)$

$\phi(N) := (p-1)(q-1)$

find e such that $\gcd(e, \phi(N)) = 1$

compute $d := [e^{-1} \bmod \phi(N)]$

return N, e, d

RSA ASSUMPTION

The RSA experiment $\mathsf{RSA\text{-}inv}_{\mathcal{A},\mathsf{GenRSA}}(n)$**:**

1. *Run* $\mathsf{GenRSA}(1^n)$ *to obtain* (N, e, d).
2. *Choose* $y \leftarrow \mathbb{Z}_N^*$.
3. $\mathcal{A}$ *is given* N, e, y, *and outputs* $x \in \mathbb{Z}_N^*$.
4. *The output of the experiment is defined to be 1 if* $x^e = y \bmod N$, *and 0 otherwise.*

DEFINITION 7.46 *We say that* the RSA problem is hard relative to GenRSA *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\Pr[\mathsf{RSA\text{-}inv}_{\mathcal{A},\mathsf{GenRSA}}(n) = 1] \leq \mathsf{negl}(n).$$

THANK YOU

Any Questions?

PUBLIC-KEY ENCRYPTION

Defining Secure PKC

The CPA indistinguishability experiment $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$**:**

1. $\mathsf{Gen}(1^n)$ *is run to obtain keys* (pk, sk).
2. *Adversary* $\mathcal{A}$ *is given* pk *as well as oracle access to* $\mathsf{Enc}_{pk}(\cdot)$. *The adversary outputs a pair of messages* m_0, m_1 *of the same length. (These messages must be in the plaintext space associated with* pk*.)*
3. *A random bit* $b \leftarrow \{0,1\}$ *is chosen, and then a ciphertext* $c \leftarrow \mathsf{Enc}_{pk}(m_b)$ *is computed and given to* $\mathcal{A}$. *We call* c *the* challenge ciphertext.
4. $\mathcal{A}$ *continues to have access to* $\mathsf{Enc}_{pk}(\cdot)$, *and outputs a bit* b'.
5. *The output of the experiment is defined to be 1 if* $b' = b$, *and 0 otherwise.*

Definition (Contd.)

DEFINITION 10.4 *A public-key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under a chosen-plaintext attack *(or is* CPA secure*) if for all probabilistic polynomial-time adversaries* A *there exists a negligible function* negl *such that:*

- $\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$.

Eavesdropping vs. CPA

PROPOSITION 10.5 *If a public-key encryption scheme Π has indistinguishable encryptions in the presence of an eavesdropper, then Π also has indistinguishable encryptions under a chosen-plaintext attack.*

IMPOSSIBILITY OF PERFECT PKC

- Perfectly-secret public-key encryption is impossible

- Regardless of how long the keys are and how short the message is.

Insecurity of Deterministic Public-Key Encryption

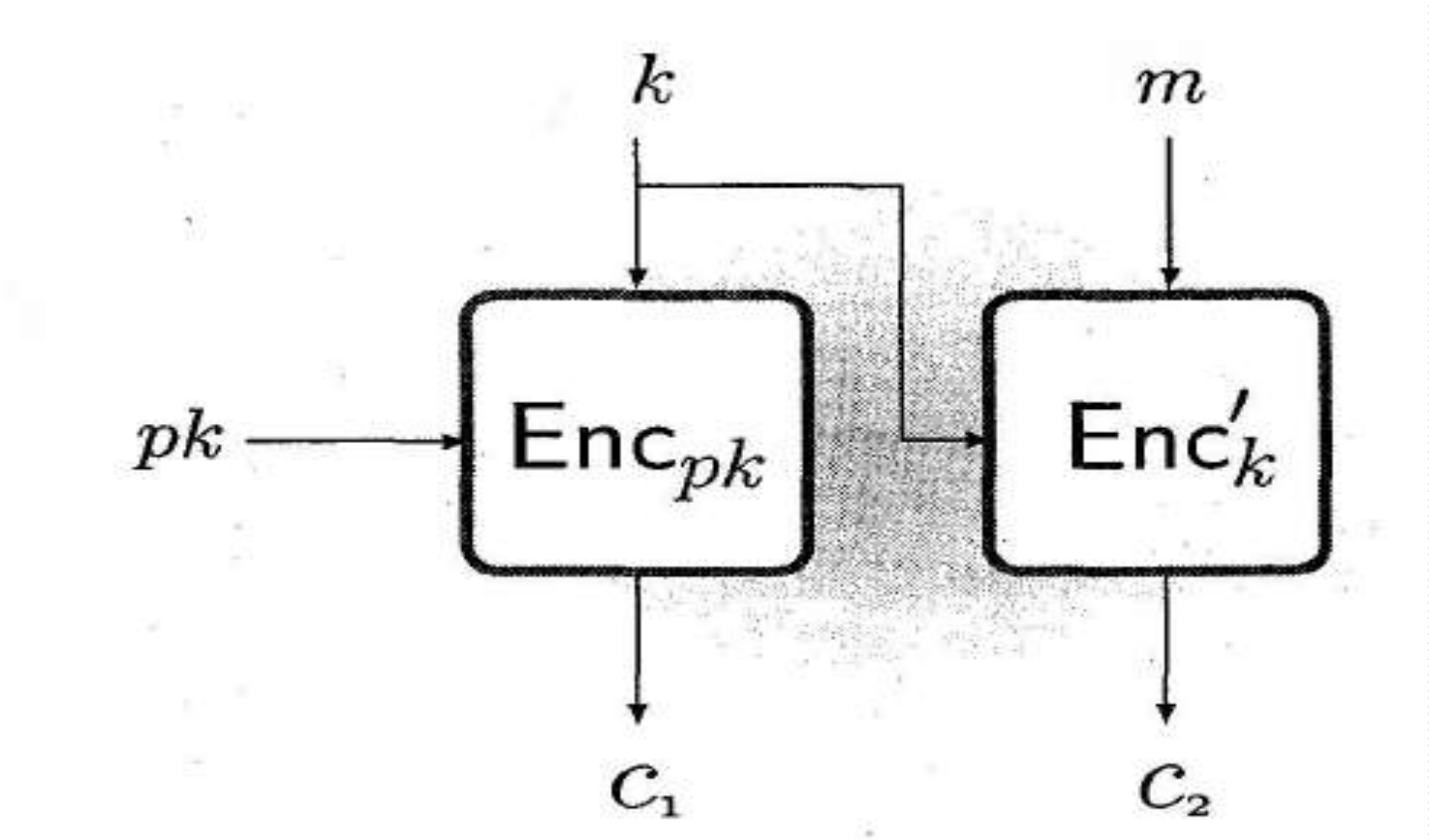
THEOREM 10.6 *No deterministic public-key encryption scheme has indistinguishable encryptions in the presence of an eavesdropper.*

Encrypting Arbitrary-Length Messages

Simply encrypt each block!

$$\mathsf{Enc}'_{pk}(m) = \mathsf{Enc}_{pk}(m_1), \ldots, \mathsf{Enc}_{pk}(m_t)$$

HYBRID ENCRYPTION



Hybrid Encryption (Contd.)

CONSTRUCTION 10.12

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme, and let $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a private-key encryption scheme. Construct a public-key encryption scheme $\Pi^{\mathsf{hy}} = (\mathsf{Gen}^{\mathsf{hy}}, \mathsf{Enc}^{\mathsf{hy}}, \mathsf{Dec}^{\mathsf{hy}})$ as follows:

- $\mathsf{Gen}^{\mathsf{hy}}$: on input 1^n run $\mathsf{Gen}(1^n)$ and use the public and private keys (pk, sk) that are output.
- $\mathsf{Enc}^{\mathsf{hy}}$: on input a public key pk and a message $m \in \{0,1\}^*$, proceed as follows:
  1. Choose a random $k \leftarrow \{0,1\}^n$ (recall that, by convention, n can be determined from pk).
  2. Compute $c_1 \leftarrow \mathsf{Enc}_{pk}(k)$ and $c_2 \leftarrow \mathsf{Enc}'_k(m)$.
  3. Output the ciphertext $\langle c_1, c_2 \rangle$.
- $\mathsf{Dec}^{\mathsf{hy}}$: on input a private key sk and a ciphertext $\langle c_1, c_2 \rangle$ proceed as follows:
  1. Compute $k := \mathsf{Dec}_{sk}(c_1)$.
  2. Output the message $m := \mathsf{Dec}'_k(c_2)$.

Hybrid encryption.

RSA ENCRYPTION

KEY GENERATION (Textbook)

ALGORITHM 10.14

RSA key generation GenRSA

Input: Security parameter 1^n

Output: N, e, d as described in the text

$(N, p, q) \leftarrow \mathsf{GenModulus}(1^n)$

$\phi(N) := (p-1)(q-1)$

choose e such that $\gcd(e, \phi(N)) = 1$

compute $d := [e^{-1} \bmod \phi(N)]$

return N, e, d

RSA "Textbook" Encryption

CONSTRUCTION 10.15

Let GenRSA be as in the text. Define a public-key encryption scheme as follows:

- Gen: on input 1^n run GenRSA(1^n) to obtain N, e, and d. The public key is $\langle N, e \rangle$ and the private key is $\langle N, d \rangle$.
- Enc: on input a public key $pk = \langle N, e \rangle$ and a message $m \in \mathbb{Z}_N^*$, compute the ciphertext

$$c := [m^e \bmod N].$$

- Dec: on input a private key $sk = \langle N, d \rangle$ and a ciphertext $c \in \mathbb{Z}_N^*$, compute the message

$$m := [c^d \bmod N].$$

The "textbook RSA" encryption scheme.

ATTACKS ON TEXTBOOK RSA

- Encrypting short messages using small e (say 3).

$$m := c^{1/3} \text{ over the integers.}$$

- A general attack when small e is used (say e =3).

$$c_1 = [m^3 \bmod N_1] \quad \text{and} \quad c_2 = [m^3 \bmod N_2] \quad \text{and} \quad c_3 = [m^3 \bmod N_3]$$

  - Use Chinese Remainder Theorem to solve for m^3

- Common Modulus Attack

$$c_1 = m^{e_1} \bmod N \quad \text{and} \quad c_2 = m^{e_2} \bmod N.$$

$$c_1^X \cdot c_2^Y = m^{Xe_1} m^{Ye_2} = m^{Xe_1 + Ye_2} = m^1 = m \bmod N$$

PADDED RSA

Probabilistic RSA Encryption

CONSTRUCTION 10.18

Let GenRSA be as before, and let ℓ be a function with $\ell(n) \leq 2n - 2$ for all n. Define a public-key encryption scheme as follows:

- Gen: on input 1^n, run GenRSA(1^n) to obtain (N, e, d). Output the public key $pk = \langle N, e \rangle$, and the private key $sk = \langle N, d \rangle$.
- Enc: on input a public key $pk = \langle N, e \rangle$ and a message $m \in \{0,1\}^{\ell(n)}$, choose a random string $r \leftarrow \{0,1\}^{\|N\| - \ell(n) - 1}$ and interpret $r \| m$ as an element of $\mathbb{Z}_N$ in the natural way. Output the ciphertext

$$c := [(r\|m)^e \bmod N].$$

- Dec: on input a private key $sk = \langle N, d \rangle$ and a ciphertext $c \in \mathbb{Z}_N^*$, compute

$$\hat{m} := [c^d \bmod N],$$

and output the $\ell(n)$ low-order bits of $\hat{m}$.

The padded RSA encryption scheme.

El Gamal Encryption

LEMMA 10.20 *Let $\mathbb{G}$ be a finite group, and let $m \in \mathbb{G}$ be an arbitrary element. Then choosing random $g \leftarrow \mathbb{G}$ and setting $g' := m \cdot g$ gives the same distribution for g' as choosing random $g' \leftarrow \mathbb{G}$. I.e., for any $\hat{g} \in \mathbb{G}$*

$$\Pr[m \cdot g = \hat{g}] = 1/|\mathbb{G}|,$$

where the probability is taken over random choice of g.

The Scheme

CONSTRUCTION 10.21

Let $\mathcal{G}$ be as in the text. Define a public-key encryption scheme as follows:

- Gen: on input 1^n run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$. Then choose a random $x \leftarrow \mathbb{Z}_q$ and compute $h := g^x$. The public key is $\langle \mathbb{G}, q, g, h \rangle$ and the private key is $\langle \mathbb{G}, q, g, x \rangle$.
- Enc: on input a public key $pk = \langle \mathbb{G}, q, g, h \rangle$ and a message $m \in \mathbb{G}$, choose a random $y \leftarrow \mathbb{Z}_q$ and output the ciphertext

$$\langle g^y, h^y \cdot m \rangle.$$

- Dec: on input a private key $sk = \langle \mathbb{G}, q, g, x \rangle$ and a ciphertext $\langle c_1, c_2 \rangle$, output

$$m := c_2 / c_1^x.$$

The El Gamal encryption scheme.

THEOREM 10.22 *If the DDH problem is hard relative to $\mathcal{G}$, then the El Gamal encryption scheme has indistinguishable encryptions under a chosen-plaintext attack.*

CHOSEN CIPHERTEXT ATTACK on EL GAMAL CRYPTOSYSTEM

$$c_1 = g^y \quad \text{and} \quad c_2 = h^y \cdot m$$

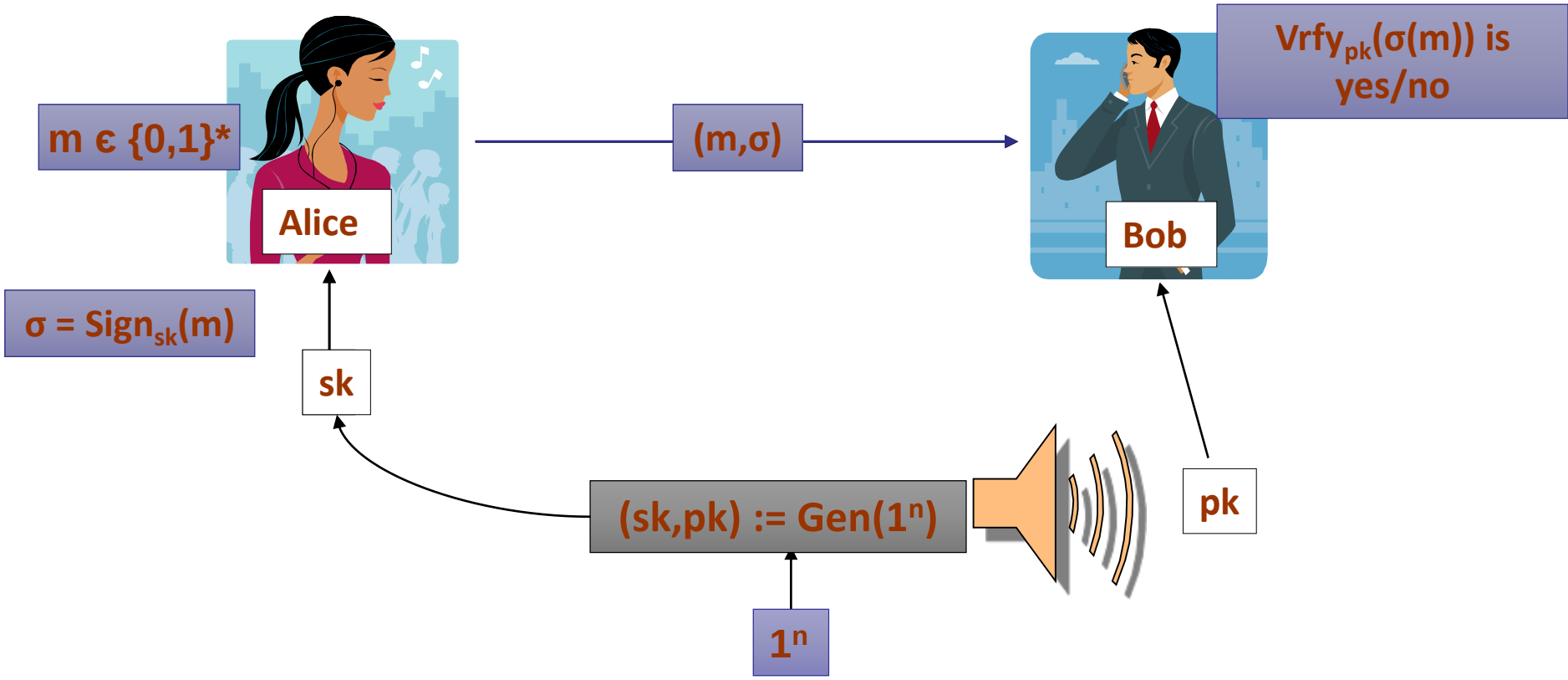
$$c_1'' = g^y \cdot g^{y''} = g^{y+y''} \quad \text{and} \quad c_2'' = h^y m \cdot h^{y''} m' = h^{y+y''} mm'$$

THANK YOU

Any Questions?

DIGITAL SIGNATURES

Digital Signature Scheme



Digital Signature Schemes

A **digital signature scheme** is a tuple **(Gen,Sign,Vrfy)** of poly-time algorithms, such that:

- the **key-generation** algorithm **Gen** takes as input a security parameter 1^n and outputs a pair **(sk,pk)**,
- the **signing** algorithm **Sign** takes as input a key **sk** and a message $m \in \{0,1\}^*$ and outputs a signature **σ**,
- the **verification** algorithm **Vrfy** takes as input a key **pk**, a message **m** and a signature **σ**, and outputs a bit **b ϵ {yes, no}.**

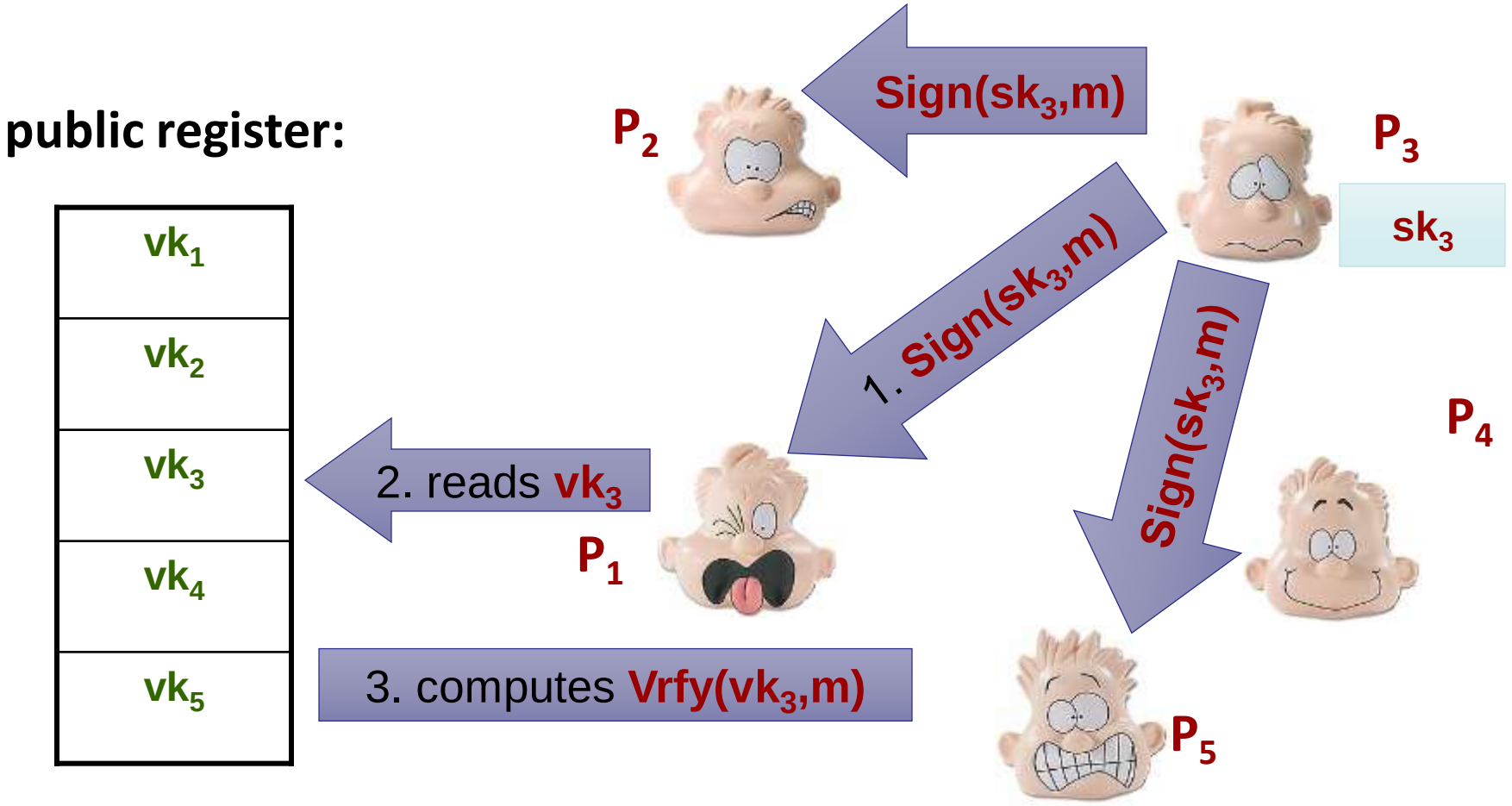
If $\text{Vrfy}_{vk}(m,\sigma) = \text{yes}$ then we say that **σ** is a **valid signature on the message m.**

Advantages of the signature schemes

Digital signatures:

1. **Simplify Key Management**
2. **Are publicly verifiable**
3. **Are transferable**
4. **Provide non-repudiation**

Easy Key Management



Signatures are publicly-verifiable!

sk_A

$m \in \{0,1\}^*$

$(m, \sigma = Sign_{sk}(m))$

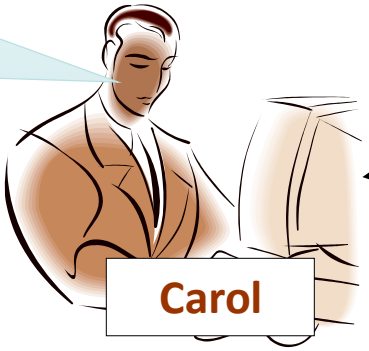
pk_A

Look, I got (m,σ) from Alice

I can calculate

$Vrfy_{pkA}(m,\sigma)$

and check.



So, the signatures are transferable

Alice

sk_A

$\sigma = Sign(sk_3, m)$

"Alice signed m"

I believe it!

"Alice signed m"

I believe it!

"Alice signed m"

I believe it!

pk_A P_1

(m,σ)

pk_A P_2

(m,σ)

pk_A P_3

(m,σ)

pk_A P_4

Non-repudiation

sk_A

$m \in \{0,1\}^*$

Alice

$(m, \sigma = Sign_{sk}(m))$

pk_A

Bob

"I've got (m,σ) from Alice"

It's not true!
I never signed **m**!

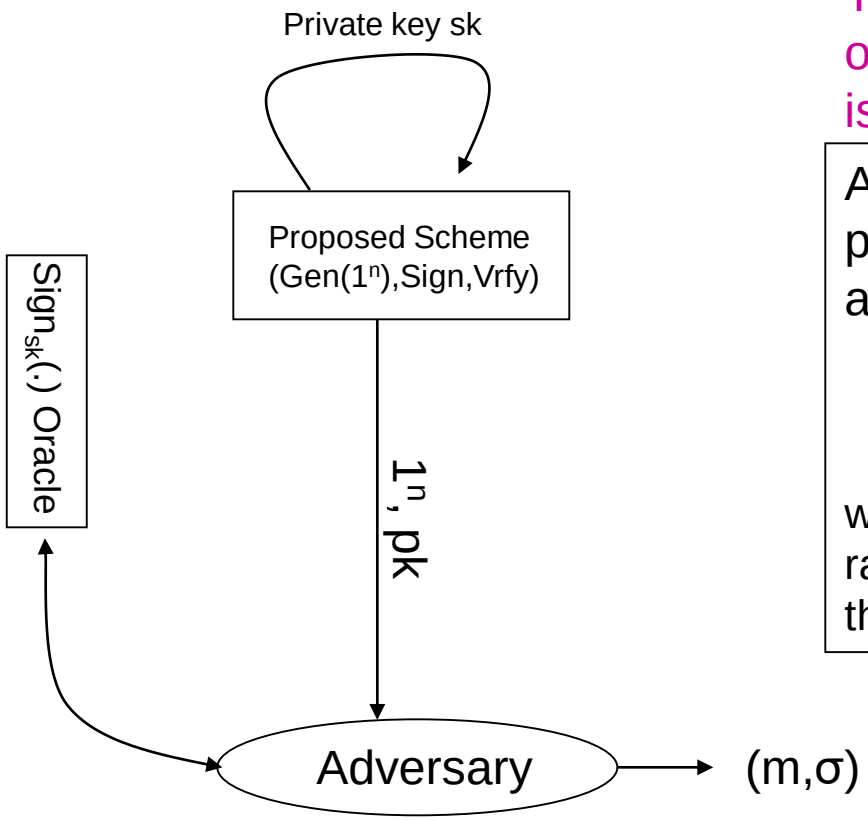
Judge

$Vrfy_{pk}(m,\sigma) = yes$

so you cannot **repudiate** signing **m**...

Defining Secure Digital Signatures

The Signature Adversarial Game



Let Q be the set of messages whose signatures were requested by the adversary from the oracle. The output of the game is 1 if and only if (1) $Vrfy_{pk}(m,\sigma) = 1$ and (2) m is not in Q, else 0

A signature scheme is secure if for all probabilistic polynomial-time adversaries:

Pr[Output of the game = 1] ≤ negl(n)

where the probability is taken over the random coins used by adversary as well as the random coins used in the game

RSA Signatures

“RSA signatures”

N = pq - **RSA** modulus.

$\phi(N) = (p-1)(q-1)$ –Euler Totient Function

e is such that $\gcd(e, \phi(N)) = 1$,

d is such that $ed \equiv 1 \pmod{\phi(N)}$

$$\sigma(m) = m^d \bmod N$$

And

$$Vrfy_{(e,N)}(m, \sigma) = \text{yes} \text{ iff } m = \sigma^e \bmod N$$

Correctness:

$$\sigma^e \equiv (m^d)^e \equiv m^{de} \equiv m^1 \equiv m$$

Attacks on RSA

Key Only Attack

$s = Sign_{sk}(m) = m^d \bmod N$

1. Intercepts the message signature pair (**m,s**)

Computes:
m'= **$s^e \bmod N$**

$Sign_{sk}(m') = m^d \bmod N$

this is a valid signature on **m', thus bob believe s is a valid signature on m**

Known message attack

Alice

Adversary

$\text{Sign}_{sk}(m_2) = m_2^d \bmod N$

1. Itercepts two message signature pair (m_1, s_1) and (m_2, s_2)

computes (**mod N**):

$m_1^d \cdot m_2^d$

$= (m_1 \cdot m_2)^d$

$= m^d$

$\text{Sign}_{sk}(m) = m^d \bmod N$

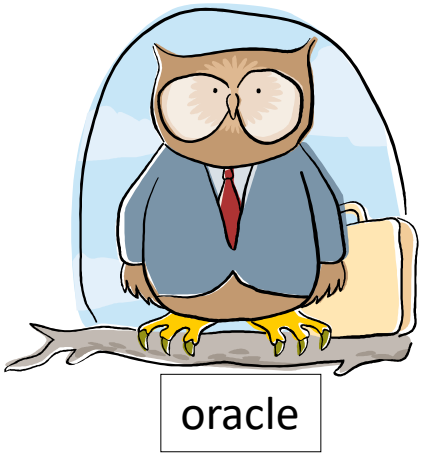
Bob

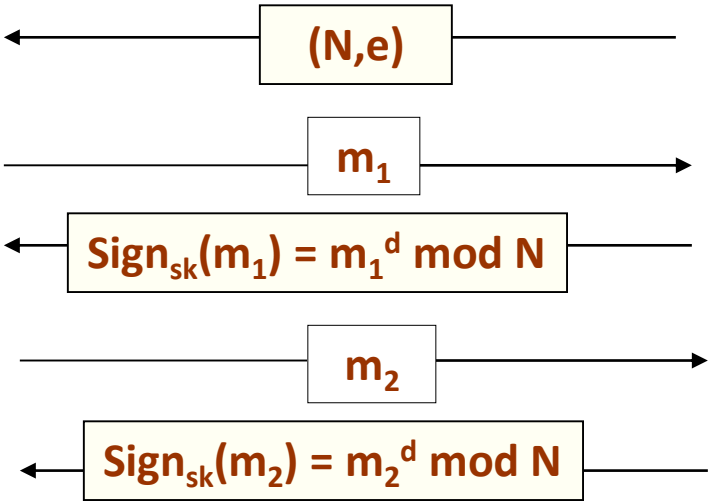
this is a valid signature on **m, thus bob believe s is a valid signature on m**

Chosen message attack

How to forge a signature on an arbitrary message **m**?

Use the homomorphic properties of RSA.





chooses:

1. random m_1
2. $m_2 := m / m_1 \bmod N$

computes (**mod N**):

$$m_1^d \cdot m_2^d$$
$$= (m_1 \cdot m_2)^d$$
$$= m^d$$

this is a valid signature on **m**

Remedial Measure: Hashed RSA

Before computing the RSA function – apply some function **H**.

N = pq, such that **p** and **q** are large random primes
e is such that **gcd(e, φ(N)) = 1**
d is such that **ed = 1 (mod φ(N))**

$Sign_d$: $Z_N^* \rightarrow Z_N^*$ is defined as:

$$Sign(m) = H(m)^d \bmod N.$$

$Vrfy_e$ is defined as:

$$Vrfy_e(m,\sigma) = yes \text{ iff } \sigma^e = H(m) \pmod N$$

How to choose such **H**?

A minimal requirement:

it should be collision-resistant.

(because if the adversary can find two messages **m,m'** such that

H(m) = H(m')

then he can forge a signature on **m'** by asking the oracle for a signature on **m**)

A typical choice of **H**

Usually **H** is one of the popular **hash functions**.

Additional advantage:

We can sign very long messages keeping the modulus **N** small (it's much more efficient!).

It is called the

hash-and-sign paradigm.

Hash & Sign paradigm

Hash-and-Sign

Hash and sign is a generic construction that takes as input:

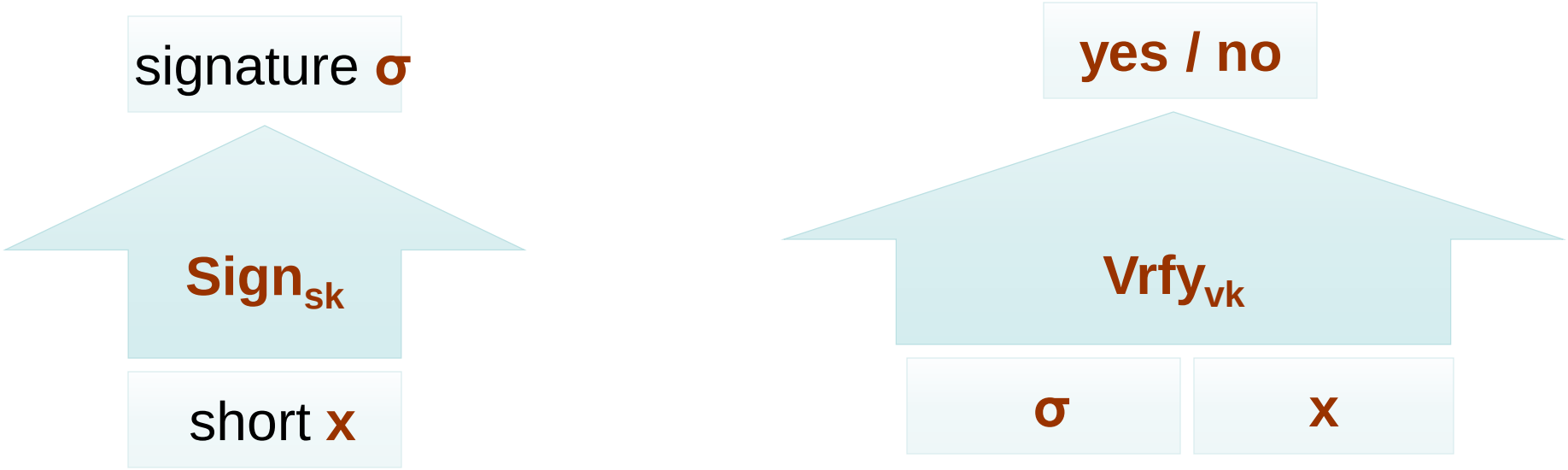
- a signature scheme that works on "**short messages**", and
- a **hash function**,

and transforms it into a

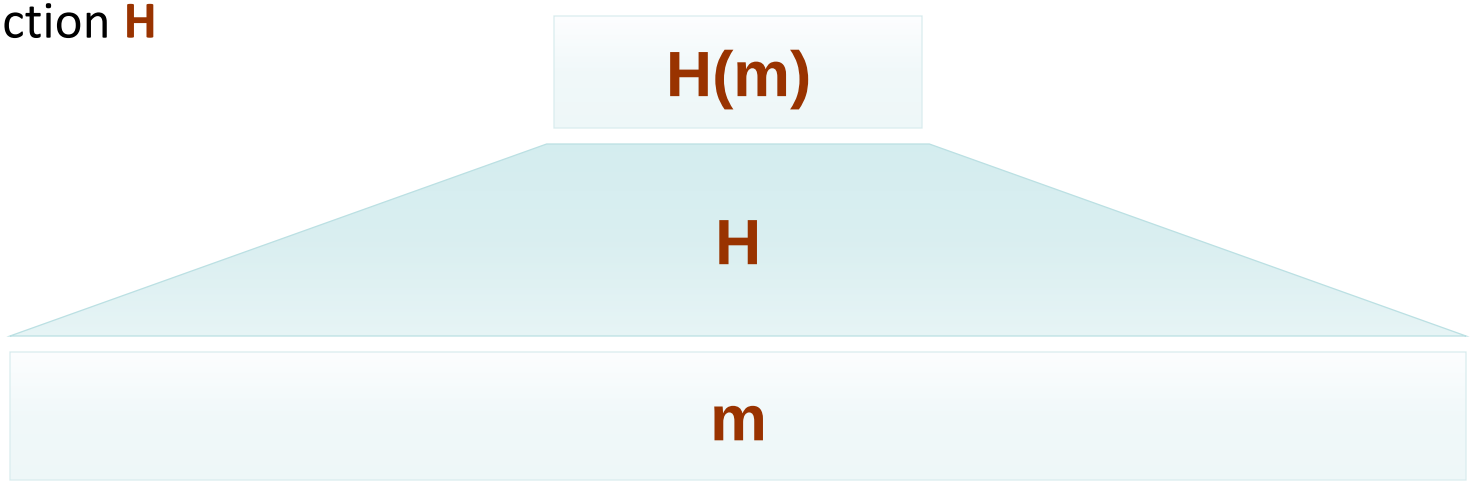
- a signature scheme that works on "**long messages**".

Hash-and-Sign

1. **(Gen,Sign,Vrfy)** – a signature scheme “for short messages”

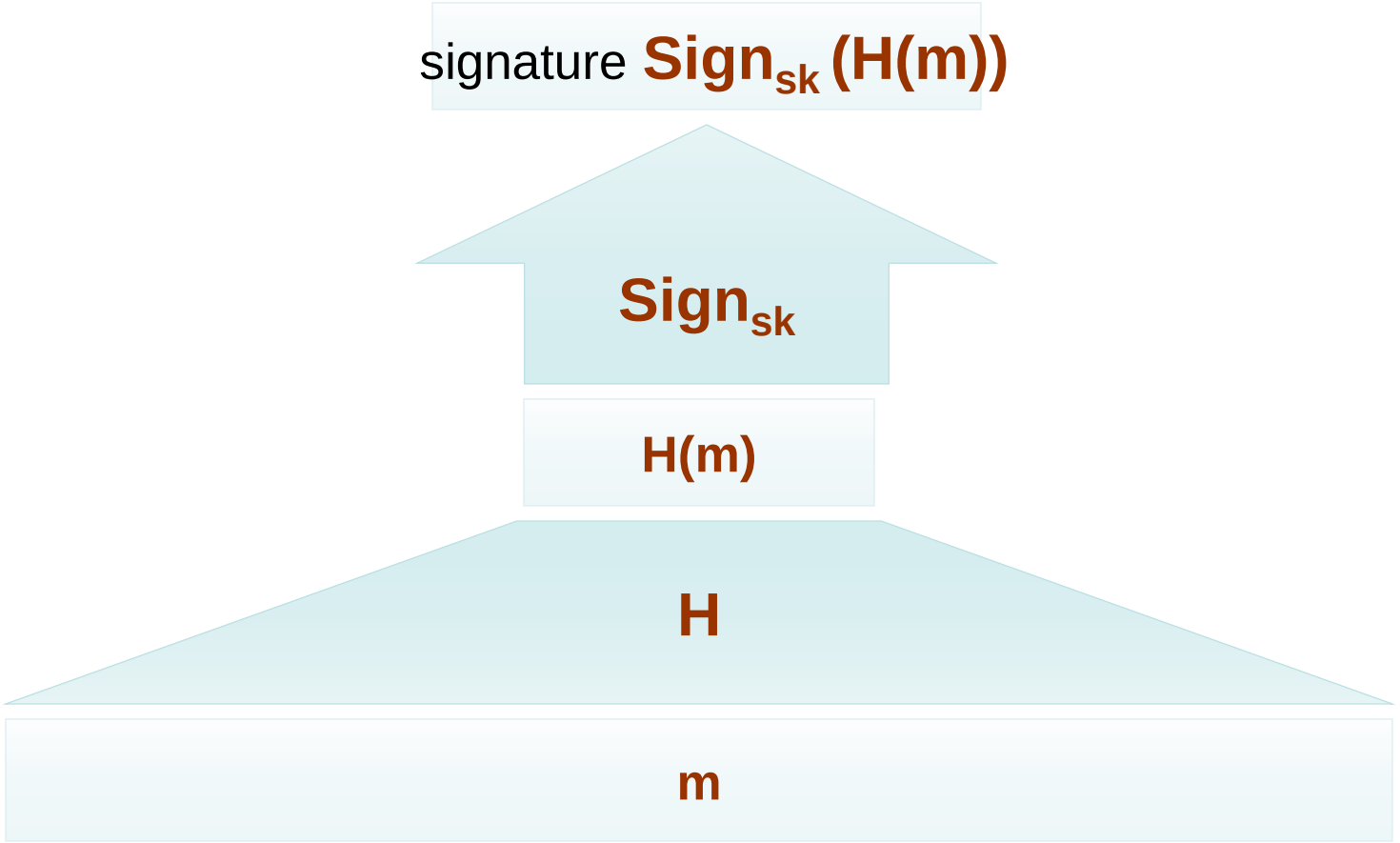


2. a hash function **H**



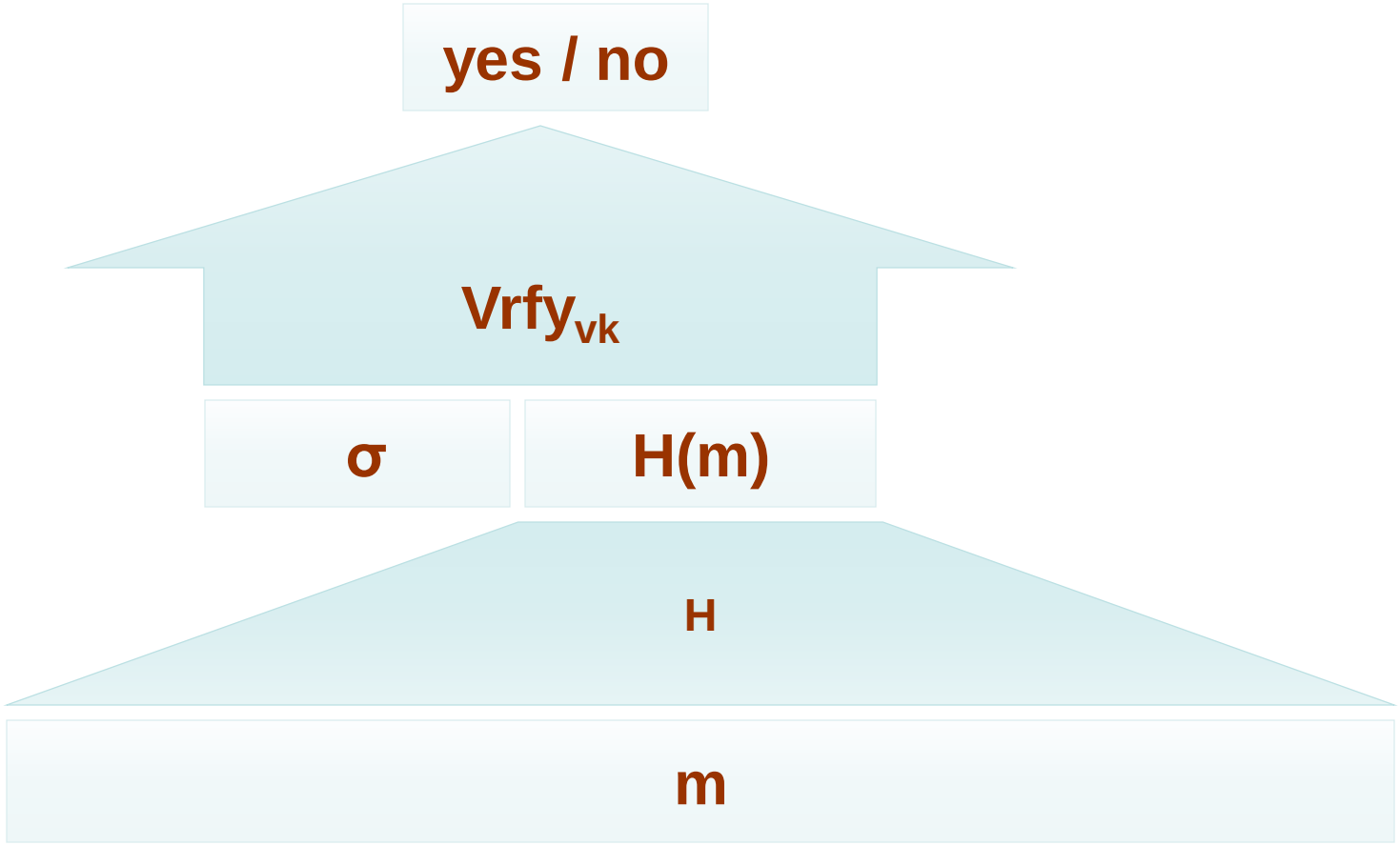
Hash-and-Sign

How to sign a message **m**?



Hash-and-Sign

How to verify?



Hash-and-Sign

It can be proven that this construction is secure.

For this we need to assume that **H** is taken from a family of collision-resilient hash functions.

$$\{H^s\}_{s \in keys}$$

Then **s** becomes a part of the public key and the private key.

What can be proven

Suppose

1. $\{H^s\}_{s \in keys}$ is a family of collision-resistant hash functions,
2. **(Gen,Sign,Vrfy)** is a secure signature scheme.

Theorem: The signature scheme constructed in the prequel is secure (for signing arbitrary length messages)

Can anything be proven about the "hashed RSA" scheme?

In the plain model - not really.

But at least the attacks described before "look infeasible"

1. For the "Key only attack": one would need to invert **H**.

2. The Chosen message attack:
   Looks impossible because the adversary would need to find messages $\mathbf{m, m_1, m_2}$ such that

$$\mathbf{H(m) = H(m_1) \cdot H(m_2)}$$

THANK YOU

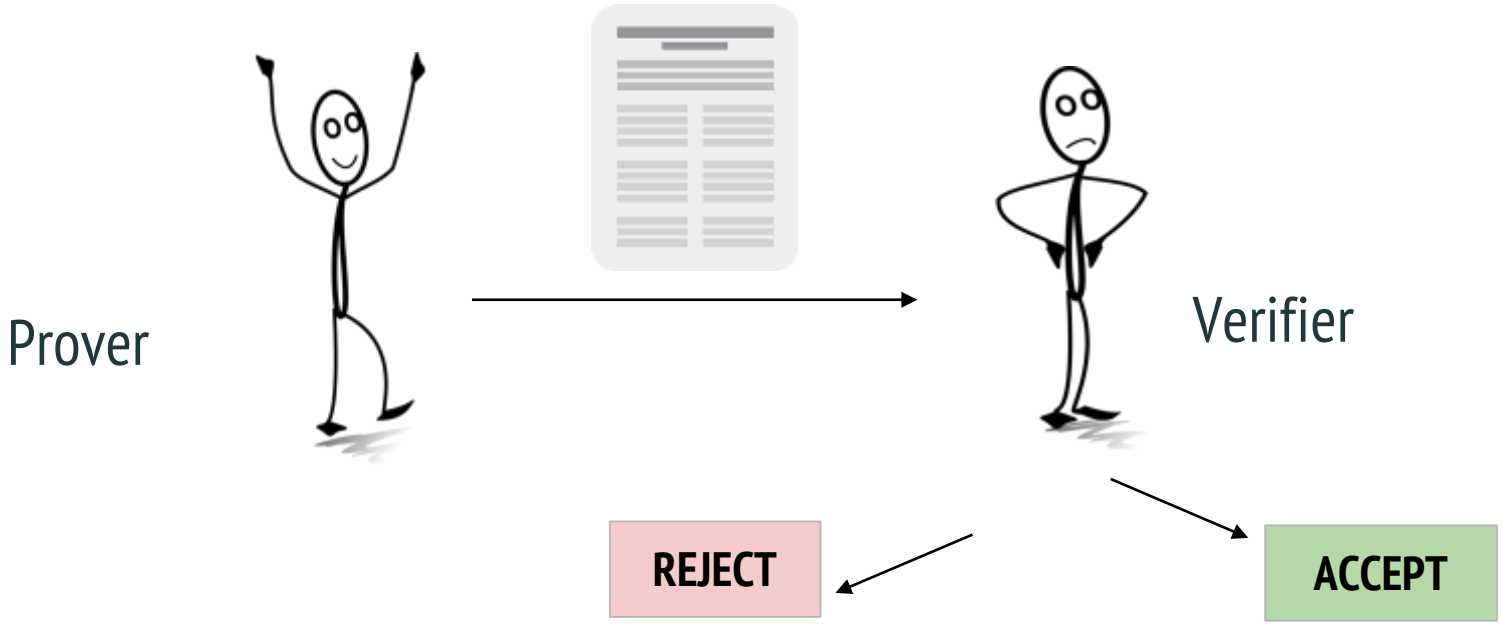
Any Questions?

ZERO KNOWLEDGE PROOFS

Contents

- Interactive Proofs
  - How to convince others (that you know something)?
- Zero-Knowledge Proofs (ZKP)
  - Proofs that reveal nothing (about what you know)
- Non-interactive ZKP
  - Using the Fiat-Shamir Heuristic
- Succinctness
  - Can the proof that you know something be shorter than (or even *independent* of) the length of what you know?
- ZK-SNARKs
  - Using Quadratic Arithmetic Programs & Bilinear Pairings

Standard Proof



Prover sends the actual proof to the Verifier

Efficient Verifiability

A ***verifier*** for a language A is an algorithm V, where

$$A = \{w | \ V \text{ accepts } \langle w, c \rangle \text{ for some string } c\}.$$

We measure the time of a verifier only in terms of the length of w, so a ***polynomial time verifier*** runs in polynomial time in the length of w. A language A is ***polynomially verifiable*** if it has a polynomial time verifier.

CLASS NP

NP is the class of languages that have polynomial time verifiers.

EXAMPLES of Languages in NP

$$CLIQUE = \{\langle G, k\rangle \mid G \text{ is an undirected graph with a } k\text{-clique}\}.$$

PROOF The following is a verifier V for $CLIQUE$.

V = "On input $\langle\langle G, k\rangle, c\rangle$:

1. Test whether c is a set of k nodes in G
2. Test whether G contains all edges connecting nodes in c.
3. If both pass, *accept*; otherwise, *reject*."

$$SUBSET\text{-}SUM = \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\} \text{ and for some } \{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\}, \text{ we have } \Sigma y_i = t\}.$$

PROOF The following is a verifier V for *SUBSET-SUM*.

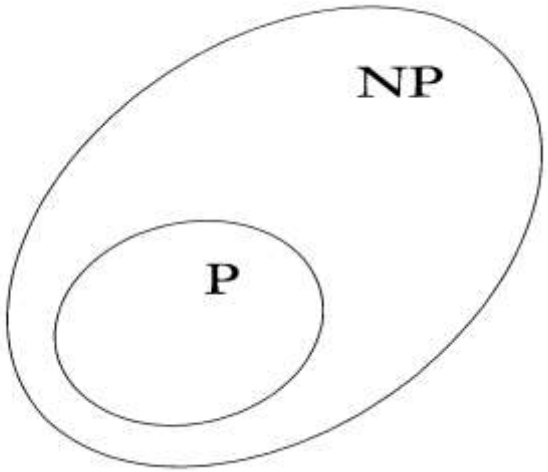
V = "On input $\langle\langle S, t\rangle, c\rangle$:

1. Test whether c is a collection of numbers that sum to t.
2. Test whether S contains all the numbers in c.
3. If both pass, *accept*; otherwise, *reject*."

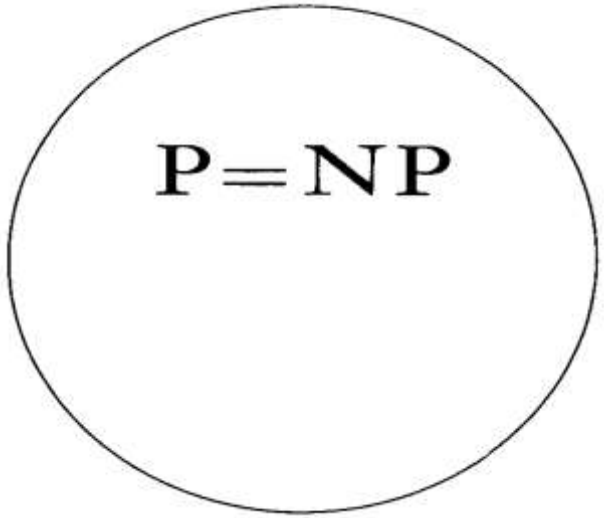
THE P VERSUS NP QUESTION

P = the class of languages for which membership can be *decided* quickly.

NP = the class of languages for which membership can be *verified* quickly.



OR



Idea of an interactive proof

- Verifier has **randomness.**
- **Probabilistic** conclusion.
- **Interactive** in nature.

How does it help?

- **Efficiently verifiable proofs beyond NP?**
- **Zero-Knowledge proofs?**
- **Succinct proofs for NP?**
- **What "extra" assumptions can bring back non-interactive proofs for NP (without trading off succinctness and zero-knowledge)?**

How to assure Prover?

Completeness property

If Prover doesn't lie, Verifier will accept

(with probability 1)

How to assure Verifier?

Soundness property

If Prover lies, Verifier will reject

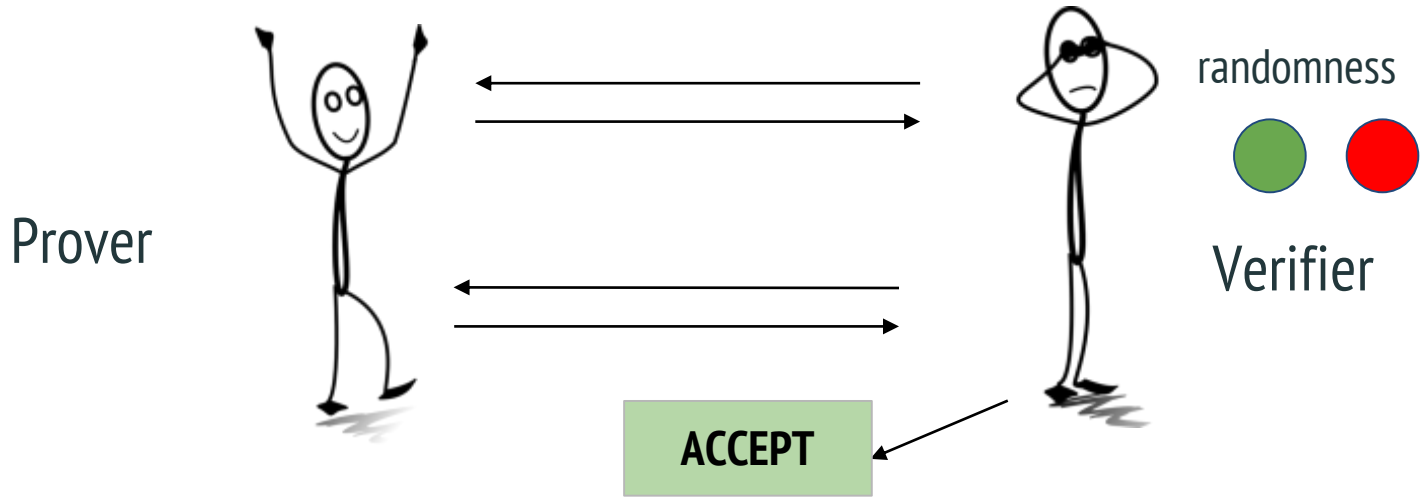
(with high probability)

Example of an Interactive Proof

Prove to a colorblind friend that two balls are differently colored

Simple Interactive Proof

Completeness: If they are of different colors



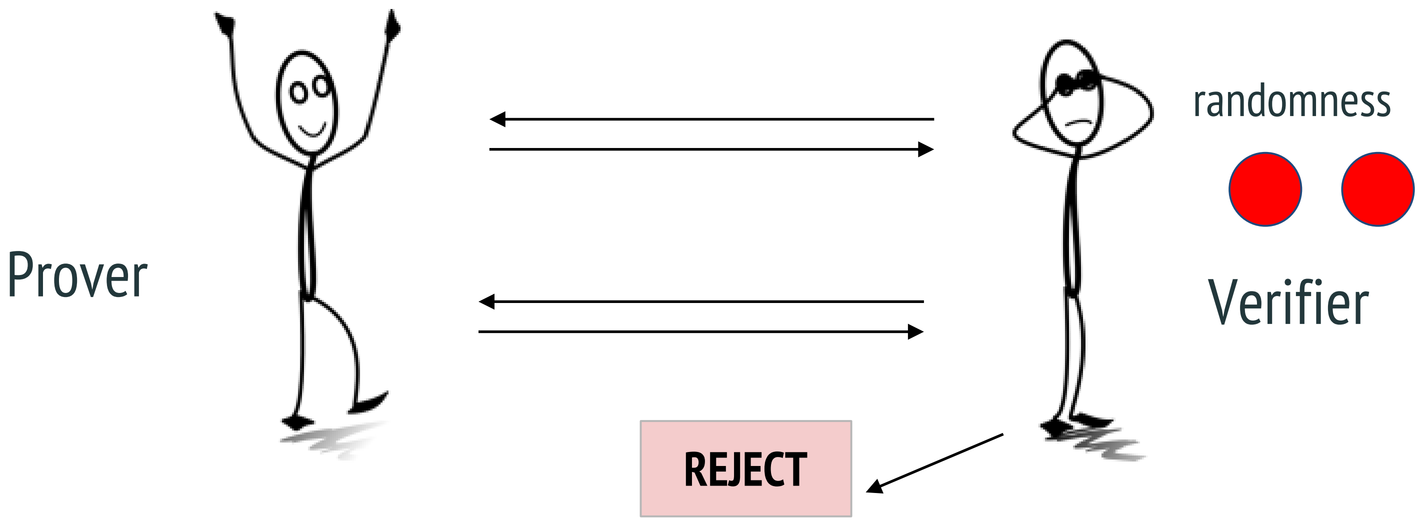
Verifier hides the balls behind his back and either interchanges them or not and then asks the prover to guess what he did.

Since they are different colored, Prover will be able to answer correctly every single time

After k correct guesses, Verifier accepts the proof

Simple Interactive Proof

Soundness: If they are of same color



Since they are same color Prover can only guess correctly with probability ½ that the balls have been flipped or not.

Verifier will reject Prover proof with high probability

Prover to guess what he did.

Going beyond NP:

Think of proving the knowledge of winning strategies in games:
Interactive proof: Just play the game several times!

But what if there is a secret

Prover asks Verifier to check her proof of the Riemann hypothesis.

Verifier copies it, publishes it, and wins a million dollars.

(plus tenure at Harvard!)

Or, consider the blockchain transaction verification computation, with only the hash values public. You need to prove that there exists a valid transaction that takes the pre-hash to post-hash

Enter Zero Knowledge

Zero Knowledge (ZK) property

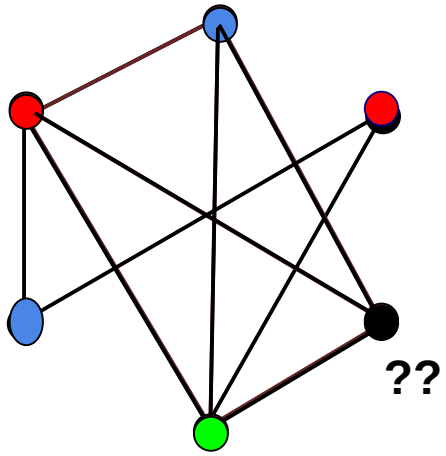
The verifier shouldn't gain **any** knowledge from the interaction

Example of a ZKP

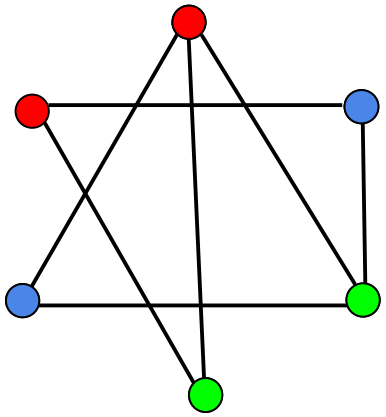
Is a given graph **3-Colorable**?

Graph 3-Coloring (G3C)

Label graph vertices using only 3 colors:
(**red**, **green**, **blue** here)



non-3-colorable graph



3-colorable graph

Why choose G3C?

G3C is NP-Complete

NP-COMPLETENESS THEORY

Let’s digress a bit!

POLYNOMIAL TIME REDUCIBILITY

A function $f: \Sigma^* \longrightarrow \Sigma^*$ is a ***polynomial time computable function*** if some polynomial time Turing machine M exists that halts with just $f(w)$ on its tape, when started on any input w.

Language A is ***polynomial time mapping reducible***,[1] or simply ***polynomial time reducible***, to language B, written $A \leq_{\mathrm{P}} B$, if a polynomial time computable function $f: \Sigma^* \longrightarrow \Sigma^*$ exists, where for every w,

$$w \in A \iff f(w) \in B.$$

The function f is called the ***polynomial time reduction*** of A to B.

DEFINITION OF NP-COMPLETENESS

A language B is ***NP-complete*** if it satisfies two conditions:

1. B is in NP, and
2. every A in NP is polynomial time reducible to B.

THE COOK–LEVIN THEOREM

SAT is NP-complete.

If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

PROOF We already know that C is in NP, so we must show that every A in NP is polynomial time reducible to C. Because B is NP-complete, every language in NP is polynomial time reducible to B, and B in turn is polynomial time reducible to C. Polynomial time reductions compose; that is, if A is polynomial time reducible to B and B is polynomial time reducible to C, then A is polynomial time reducible to C. Hence every language in NP is polynomial time reducible to C.

3SAT is NP-complete.

$$(a_1 \vee a_2 \vee \cdots \vee a_l),$$

we can replace it with the $l - 2$ clauses

$$(a_1 \vee a_2 \vee z_1) \wedge (\overline{z_1} \vee a_3 \vee z_2) \wedge (\overline{z_2} \vee a_4 \vee z_3) \wedge \cdots \wedge (\overline{z_{l-3}} \vee a_{l-1} \vee a_l).$$

3SAT is polynomial time reducible to *CLIQUE*.

where

$$3SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable 3cnf-formula}\}.$$

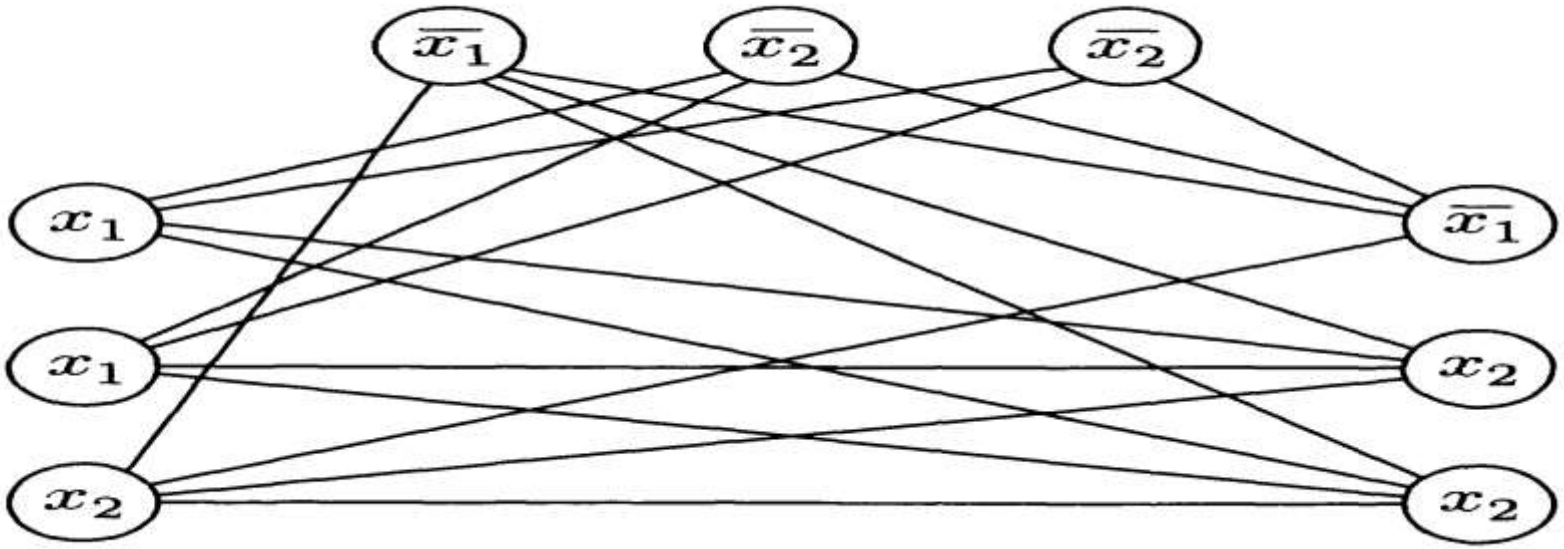
PROOF Let ϕ be a formula with k clauses such as

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \quad \cdots \quad \wedge (a_k \vee b_k \vee c_k).$$

The reduction f generates the string $\langle G, k \rangle$, where G is an undirected graph defined as follows.

The nodes in G are organized into k groups of three nodes each called the ***triples***, $t_1, \ldots, t_k$. Each triple corresponds to one of the clauses in ϕ, and each node in a triple corresponds to a literal in the associated clause. Label each node of G with its corresponding literal in ϕ.

The edges of G connect all but two types of pairs of nodes in G. No edge is present between nodes in the same triple and no edge is present between two nodes with contradictory labels, as in x_2 and $\overline{x_2}$. The following figure illustrates this construction when $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$.



Now we demonstrate why this construction works. We show that ϕ is satisfiable iff G has a k-clique.

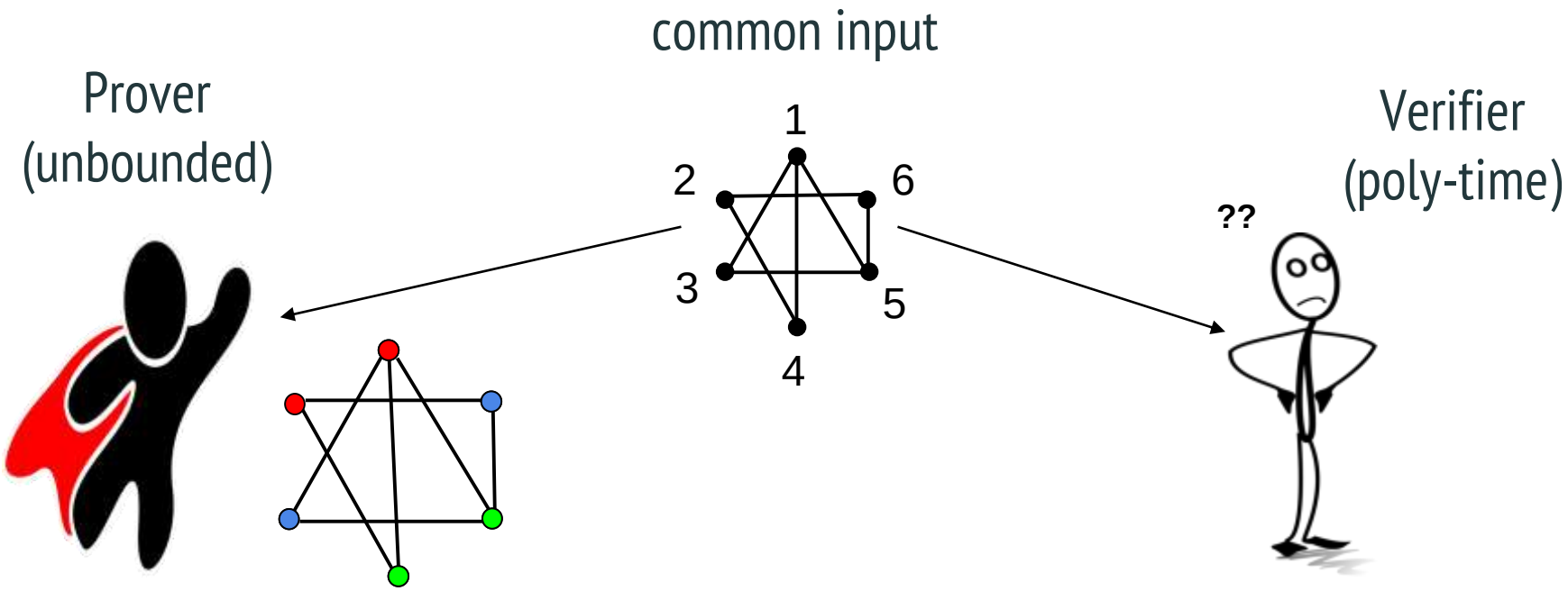
Suppose that ϕ has a satisfying assignment. In that satisfying assignment, at least one literal is true in every clause. In each triple of G, we select one node corresponding to a true literal in the satisfying assignment. If more than one literal is true in a particular clause, we choose one of the true literals arbitrarily.

The nodes just selected form a k-clique. The number of nodes selected is k, because we chose one for each of the k triples. Each pair of selected nodes is joined by an edge because no pair fits one of the exceptions described previously.

Similarly, the other direction can be proved.

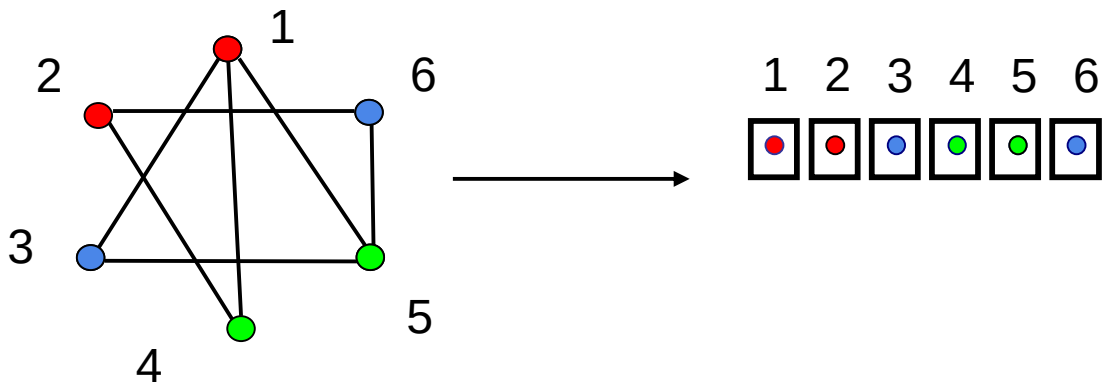
Back to Zero-Knowledge!

ZK Proof for Graph 3-Coloring

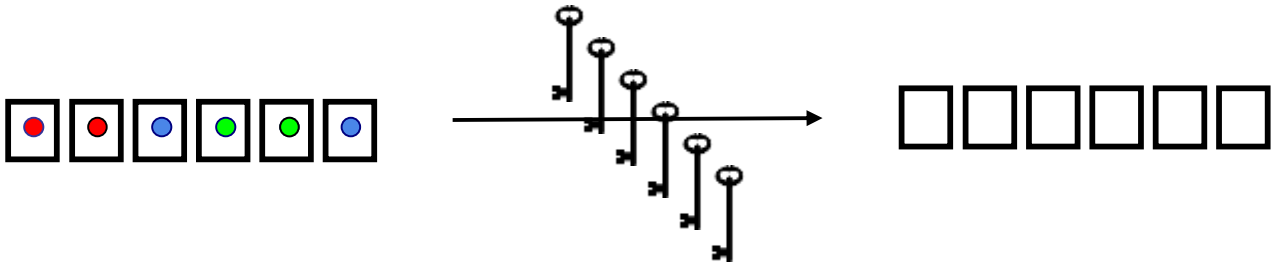


Prover side computation

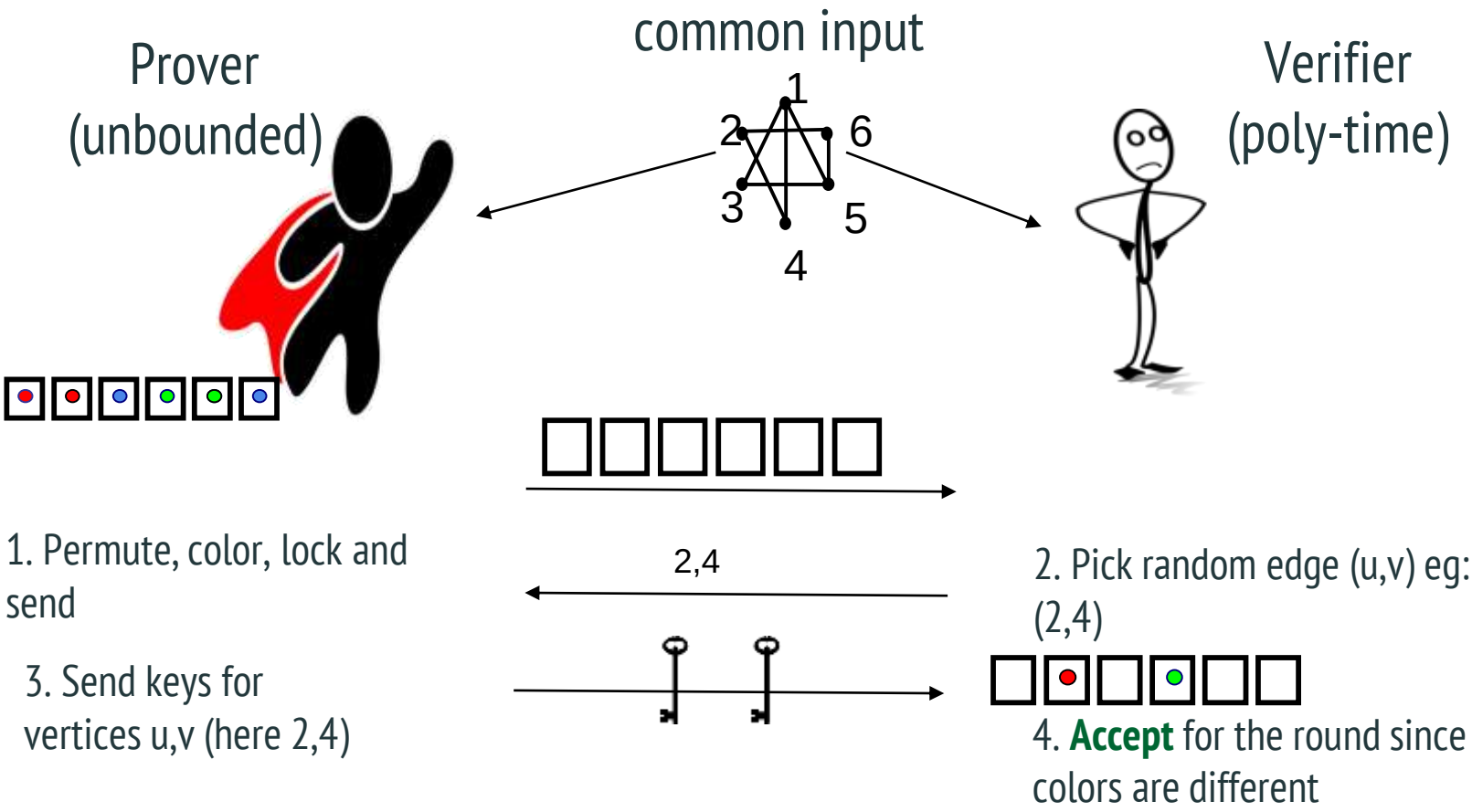
1. Select random permutation and color



2. Lock the vertices in individual boxes

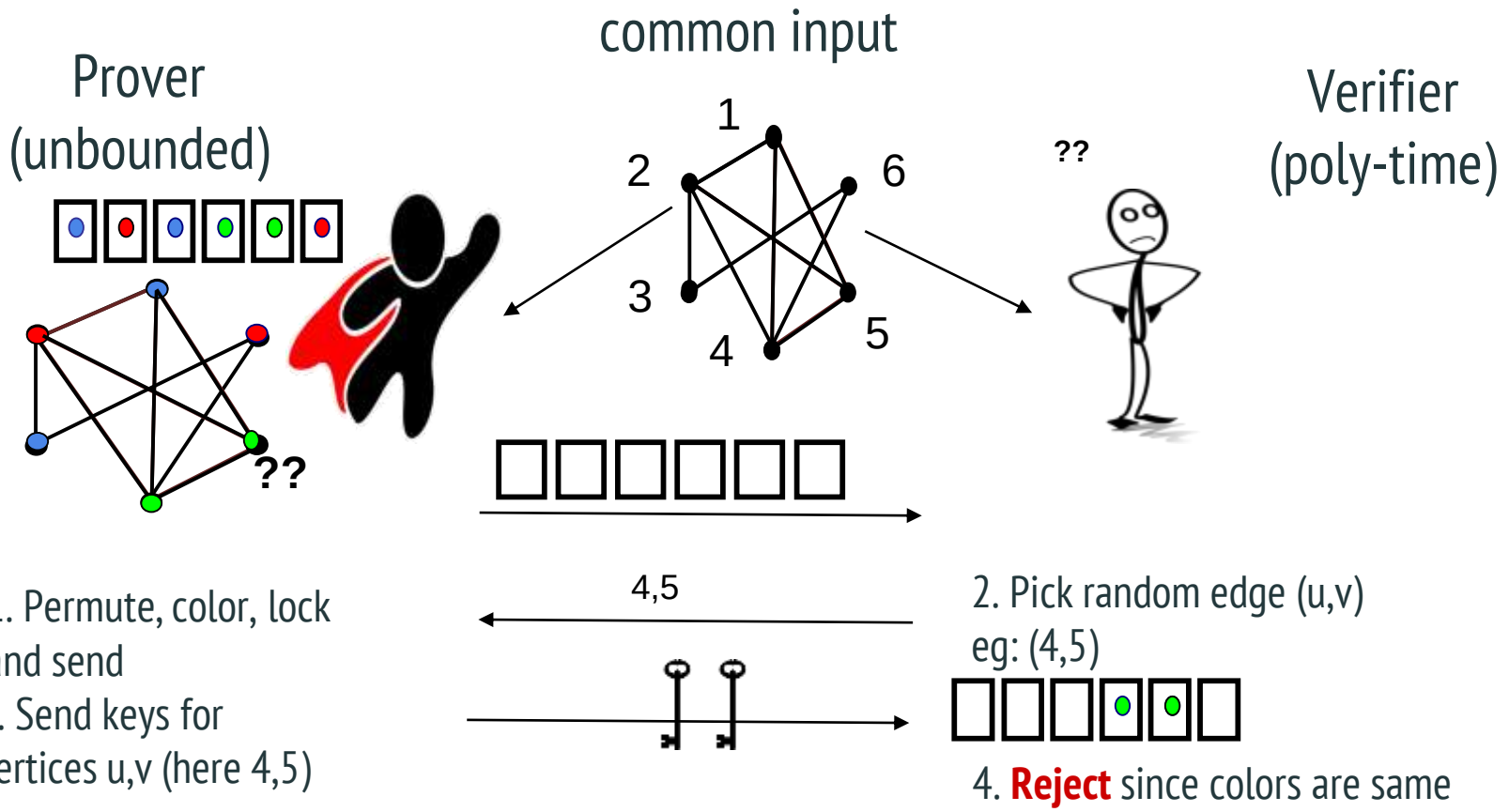


Completeness: Honest Prover



Completeness: Prover honest; accept after k rounds

Soundness: Dishonest Prover

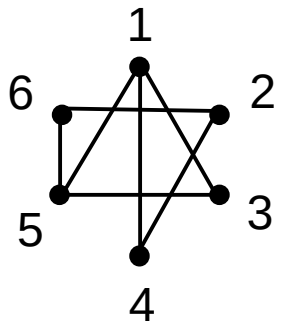


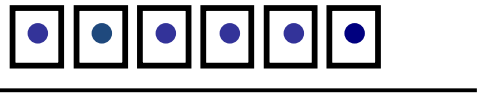
Soundness: Prover lies; reject in k^{th} round (high probability)

Formalizing Zero Knowledge

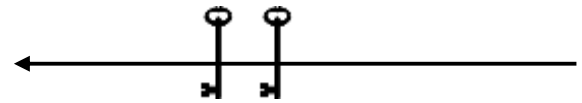
Verifier learns nothing if it could have "simulated" the interaction on its own.

Simulator for 3-Coloring





(2,6)

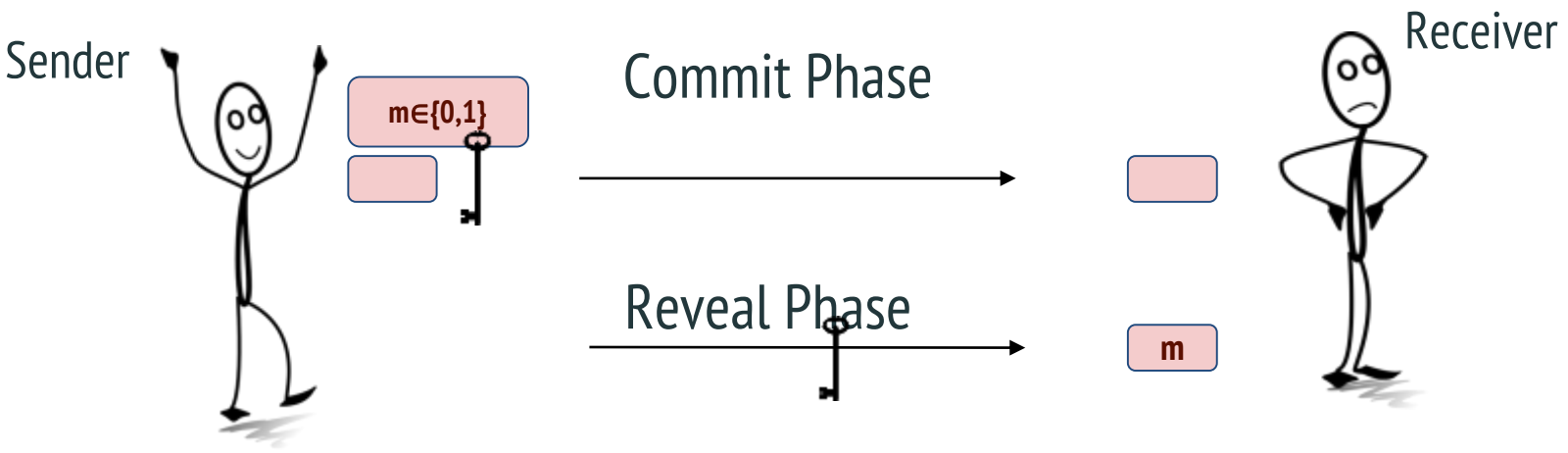


1. Randomly guess verifier's challenge, e.g. (2,6)
2. Prepare commitments.
3. Run verifier.
4. If guessed correctly, complete simulation. Else try again

How to Design the “Boxes”?

Crypto to the rescue!

Commitment Scheme



Prover commits a value by sending the key

Prover reveals value by sending the key

Commitment Properties

Binding: Sender can't change **m** after commit

Hiding: Receiver can’t see **m** before reveal

Using one-way functions

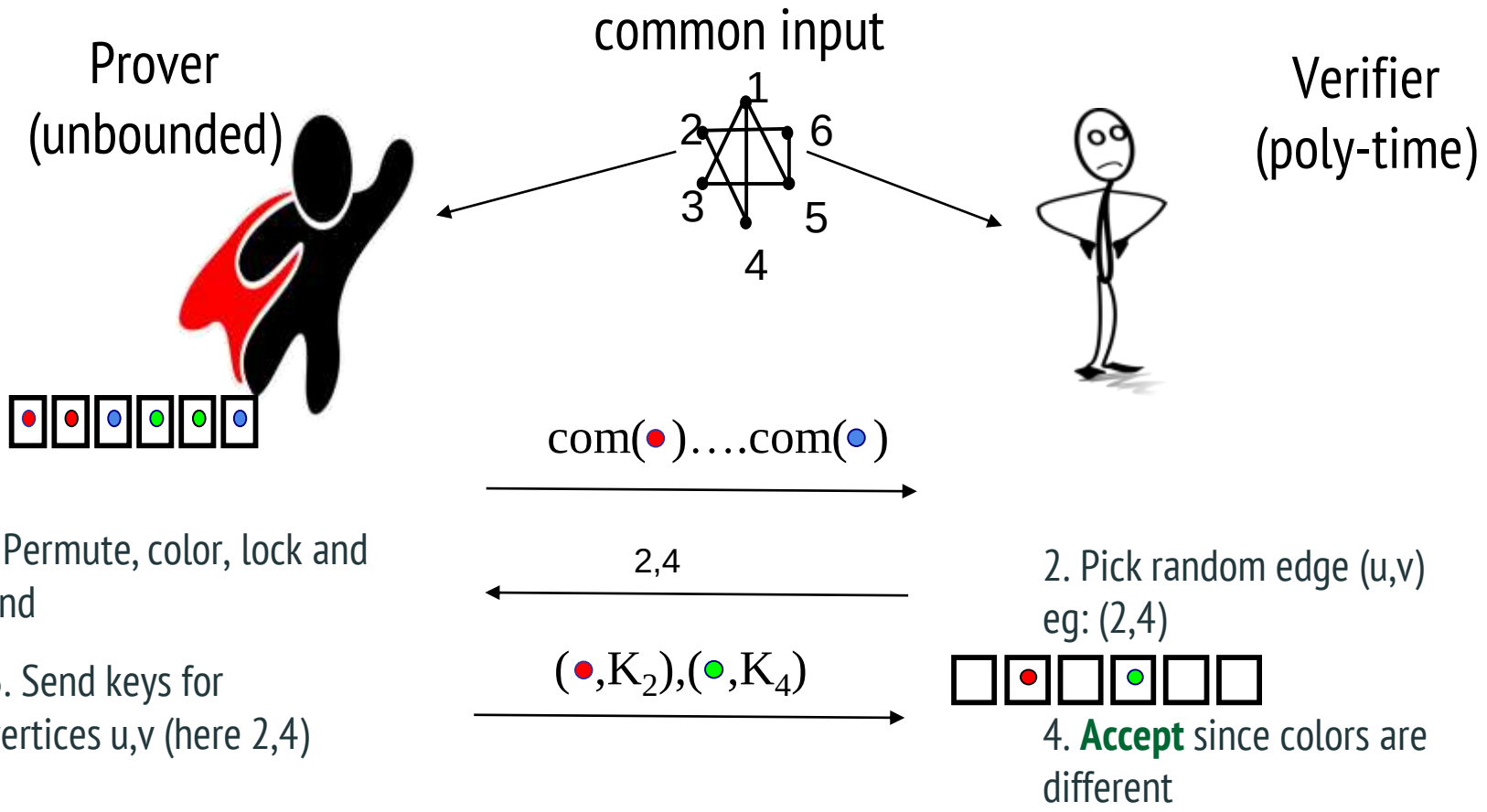
- $f: D \longrightarrow D$
  - bijective
  - One-way
    - Easy to compute
    - Hard to invert
- Commit Phase
  - Choose a random ***r***
    - Send $f(r)$ and $(h(r) \oplus m)$
- Reveal Phase
  - Send ***r*** and ***m***

Example:

$f(r) = g^r \bmod p$

h(r) = MSB(r)

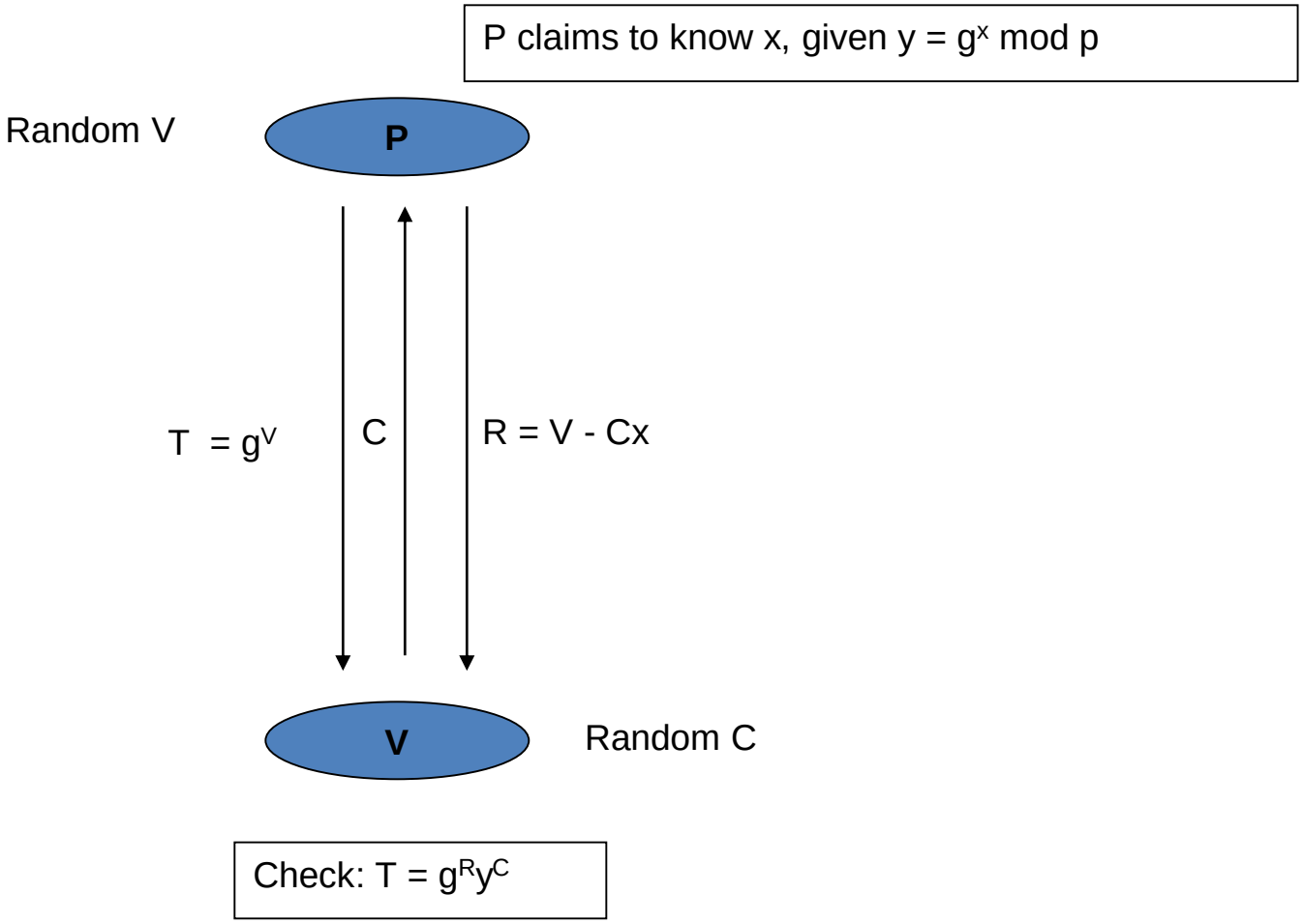
3-Coloring with commitment



ZKP for Discrete Logarithm Problem

Why study DLP?

Interactive ZKP for DLP



Completeness: $g^R y^C = g^{(V-Cx)} g^{xC} = g^V = T$

Quick Recap & Next-Up

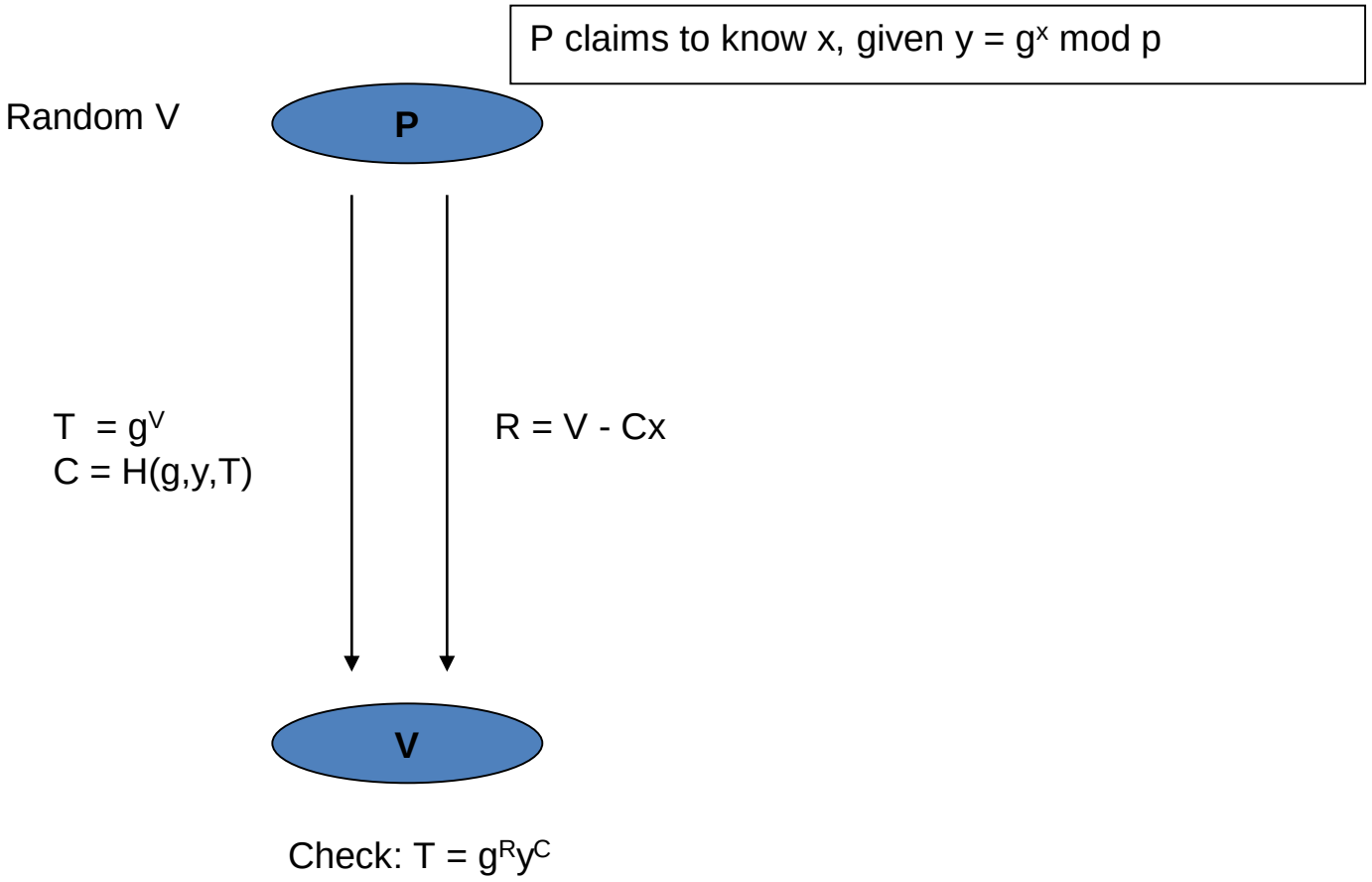
- Interactive Proofs
  - How to convince others (that you know something)?
- Zero-Knowledge Proofs (ZKP)
  - Proofs that reveal nothing (about what you know)

NEXT …

- Non-interactive ZKP
  - Using the Fiat-Shamir Heuristic
- Succinctness
  - Can the proof that you know something be shorter than (or even *independent* of) the length of what you know?
- ZK-SNARKS
  - Using Quadratic Arithmetic Programs & Bilinear Pairings

The Fiat-Shamir Heuristic

Non-Interactive ZKP for DLP: Fiat-Shamir Heuristic

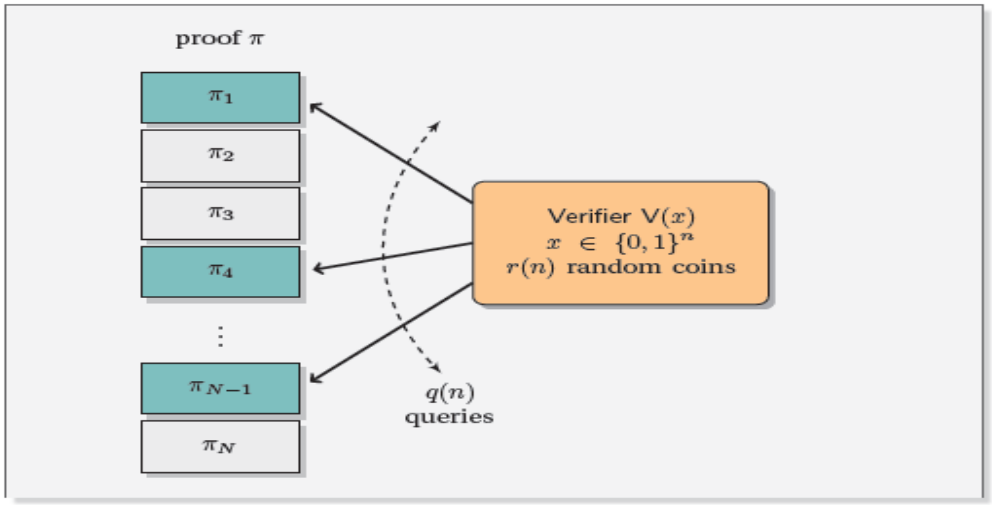


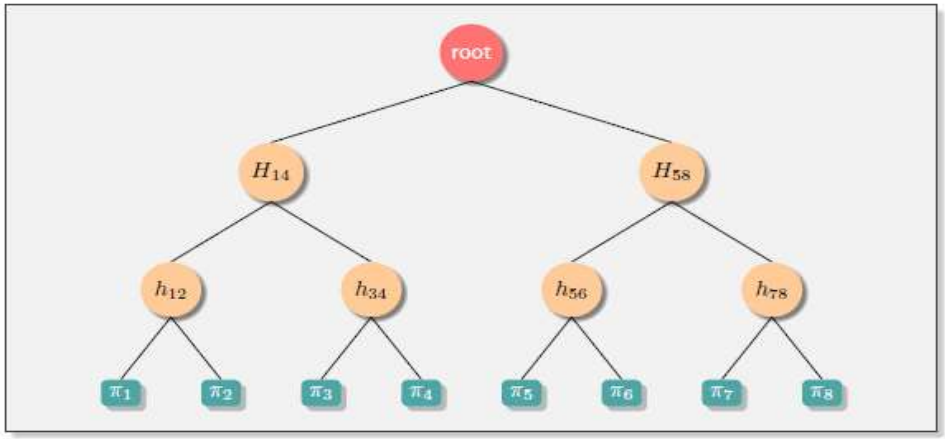
SCHNORR SIGNATURES

What about succinctness?

Succinctness from PCP Theorem and Merkle Trees

The PCP Theorem states that proofs for any NP language can be encoded in such a way that their validity can be verified by only reading a constant number of bits, and with an error probability that is upper bounded by a constant.





ZK-SNARKs from QAPs

First Main Ingredient in ZK-SNARKs

Encoding as a problem of polynomial divisibility, namely Quadratic Arithmetic Programs (QAPs)

The program that is to be checked is compiled into a quadratic equation of polynomials: $t(x)h(x) = w(x)v(x)$, where the equality holds if and only if the program is computed correctly. The prover wants to convince the verifier that this equality holds.

Second Main Ingredient in ZK-SNARKs

Succinctness by evaluation of polynomials at some random abscissa value

The verifier chooses a secret evaluation point s to reduce the problem from multiplying polynomials and verifying polynomial function equality to simple multiplication and equality check on numbers: $t(s)h(s) = w(s)v(s)$

Third Main Ingredient in ZK-SNARKs

Bilinear Pairing based cryptography for homomorphic computation/verification in encrypted domain, using a common reference string (CRS)

An encoding/encryption function E is used that has some homomorphic properties (but is not fully homomorphic, something that is not yet practical). This allows the prover to compute $E(t(s))$, $E(h(s))$, $E(w(s))$, $E(v(s))$ without knowing s, she only knows $E(s)$ and some other helpful encrypted values.

Last Main Ingredient in ZK-SNARKs

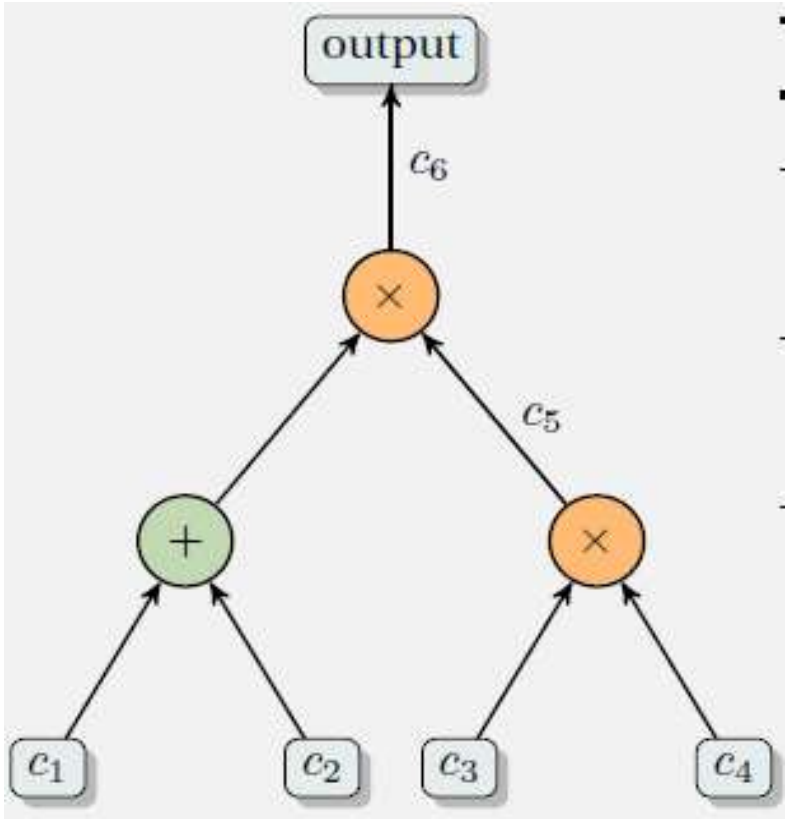
Achieving Zero-Knowledge by random blinding

The prover obfuscates the values $E(t(s))$, $E(h(s))$, $E(w(s))$, $E(v(s))$ by multiplying with a number so that the verifier can still check their correct structure without knowing the actual encoded values.

Encoding arithmetic circuits as QAPs

Definition ! (QAP). *A Quadratic Arithmetic Program Q over the field $\mathbb{F}$ contains three sets of $m+1$ polynomials $\mathcal{V} = \{v_i(x)\}$, $\mathcal{W} = \{w_i(x)\}$ and $\mathcal{Y} = \{y_i(x)\}$, $i \in \{0, 1 \ldots m\}$ and a target polynomial $t(x)$. Suppose F is an arithmetic function that takes as input n elements of $\mathbb{F}$ and outputs n' elements, for a total of $N = n + n'$ I/O elements. Then, $(c_1, \ldots, c_N) \in \mathbb{F}^N$ is a valid assignment of F's inputs and outputs, if and only if there exist coefficients $(c_{N+1}, \ldots, c_m)$ such that $t(x)$ divides $p(x)$, where:*

$$p(x) \coloneqq \left(v_0(x) + \sum_{i=1}^{m} c_i v_i(x)\right) \cdot \left(w_0(x) + \sum_{i=1}^{m} c_i w_i(x)\right) - \left(y_0(x) + \sum_{i=1}^{m} c_i y_i(x)\right).$$



Roots	Polynomials in QAP $(V, W, Y, t(x))$		
Gates	Left inputs	Right inputs	Outputs
r_5	$v_3(r_5) = 1$ $v_i(r_5) = 0,$ $i \neq 3$	$w_4(r_5) = 1$ $w_i(r_5) = 0,$ $i \neq 4$	$y_5(r_5) = 1$ $y_i(r_5) = 0,$ $i \neq 5$
r_6	$v_1(r_6) = v_2(r_6) = 1$ $v_i(r_6) = 0,$ $i \neq 1, 2$	$w_5(r_6) = 1$ $w_i(r_6) = 0,$ $i \neq 5$	$y_6(r_6) = 1$ $y_i(r_6) = 0,$ $i \neq 6$

$$t(x) = (x - r_5)(x - r_6)$$

$$p(x) \coloneqq \left(v_0(x) + \sum_{i=1}^{m} c_i v_i(x)\right) \cdot \left(w_0(x) + \sum_{i=1}^{m} c_i w_i(x)\right) - \left(y_0(x) + \sum_{i=1}^{m} c_i y_i(x)\right)$$

there must exist some polynomial $h(x)$ such that $h(x)t(x) = p(x)$.

BILINEAR GROUPS

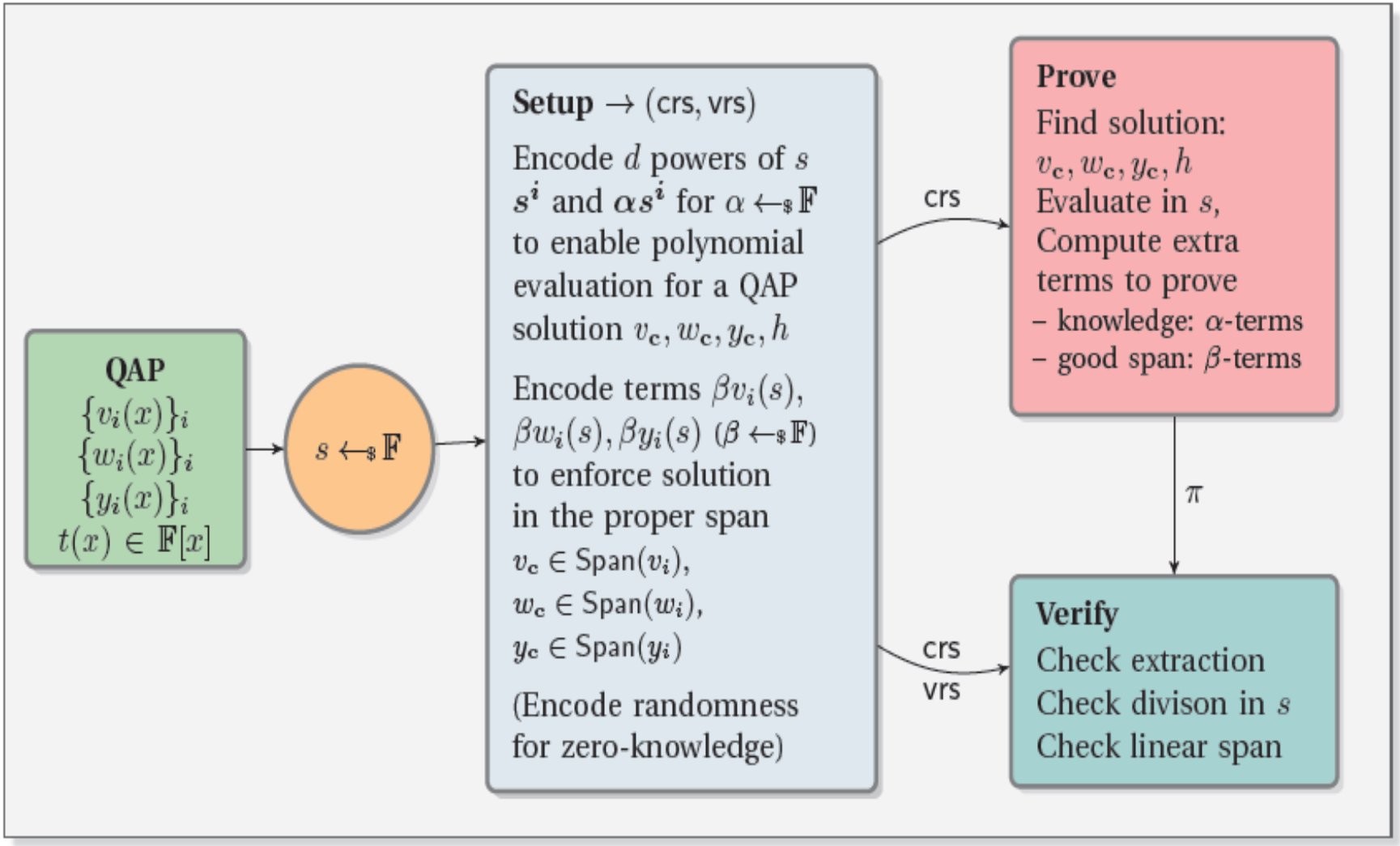
Bilinear Group $\mathsf{gk} := (p, \mathbb{G}, \mathbb{G}_T, \mathsf{e})$

- p is a λ-bit prime
- $\mathbb{G}, \mathbb{G}_T$ are cyclic groups of order p
- $\mathsf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map:

$$\forall a, b \in \mathbb{Z}_p : \mathsf{e}(g^a, g^b) = \mathsf{e}(g, g)^{ab}$$

- if $\langle g \rangle = \mathbb{G}$, then $\langle \mathsf{e}(g, g) \rangle = \mathbb{G}_T$

Roadmap for SNARKs



Extra terms computed by the Prover

$$H := \mathsf{Enc}(h(s)), \qquad \widehat{H} := \mathsf{Enc}(\alpha h(s)),$$
$$V_{\mathsf{mid}} := \mathsf{Enc}(v_{\mathsf{mid}}(s)), \qquad \widehat{V}_{\mathsf{mid}} := \mathsf{Enc}\left(\alpha v_{\mathsf{mid}}(s)\right),$$
$$W := \mathsf{Enc}(w_{\mathsf{c}}(s)), \qquad \widehat{W} := \mathsf{Enc}\left(\alpha w_{\mathsf{c}}(s)\right),$$
$$Y := \mathsf{Enc}(y_{\mathsf{c}}(s)), \qquad \widehat{Y} := \mathsf{Enc}\left(\alpha y_{\mathsf{c}}(s)\right),$$
$$B := \mathsf{Enc}\left(\beta_v v_{\mathsf{c}}(s) + \beta_w w_{\mathsf{c}}(s) + \beta_y y_{\mathsf{c}}(s)\right).$$

Verification

Extractability terms. $\hat{H} \stackrel{?}{=} \alpha H, \quad \hat{V}_{\mathsf{mid}} \stackrel{?}{=} \alpha V_{\mathsf{mid}}, \quad \hat{W} \stackrel{?}{=} \alpha W, \quad \hat{Y} \stackrel{?}{=} \alpha Y.$

Divisibility check. $H \cdot T \stackrel{?}{=} V \cdot W - Y$ where $T = \mathsf{Enc}(t(s))$, $V := \mathsf{Enc}(v_{\mathsf{c}}(s))$ and can be computed using crs. This corresponds to the polynomial division constraint.

Linear span check. $B \stackrel{?}{=} \beta_v V + \beta_w W + \beta_y Y$. This check makes sure that the polynomials $v_{\mathsf{c}}(x), w_{\mathsf{c}}(x)$ and $y_{\mathsf{c}}(x)$ are indeed linear combinations of the initial set of polynomials $\{v_i\}_i, \{w_i\}_i, \{y_i\}_i$.

SNARKs from QAP

Gen$(1^\lambda, \mathsf{C})$

$\mathsf{gk} := (p, \mathbb{G}, \mathbb{G}_T, \mathsf{e})$

$s, \alpha, \beta_v, \beta_w, \beta_y \leftarrow_\$ \mathbb{Z}_q$

$Q := \left(\{v_i, w_i, y_i\}_{i\in[m]}, t\right)$

$\mathcal{I}_{\mathsf{mid}} = \{N+1, \ldots m\}$

$\mathsf{crs} := \Big(Q, \mathsf{gk},$

$\{g^{s^i}, g^{\alpha s^i}\}_{i=0}^{d}$

$g^{\beta_v}, \{g^{\beta_v v_i(s)}\}_{i\in\mathcal{I}_{\mathsf{mid}}}$

$g^{\beta_w}, \{g^{\beta_w w_i(s)}\}_{i\in[m]}$

$g^{\beta_y}, \{g^{\beta_y y_i(s)}\}_{i\in[m]}\Big)$

return crs

Prove(crs, u, w)

$u := (c_1, \ldots, c_N)$

$w := (\{c_i\}_{i\in\mathcal{I}_{\mathsf{mid}}})$

$v_{\mathsf{mid}} := \sum_{i\in\mathcal{I}_{\mathsf{mid}}} c_i v_i(x)$

$H := g^{h(s)}, \quad \widehat{H} := g^{\alpha h(s)}$

$V_{\mathsf{mid}} := g^{v_{\mathsf{mid}}(s)}, \widehat{V}_{\mathsf{mid}} := g^{\alpha v_{\mathsf{mid}}(s)}$

$W := g^{w_{\mathsf{c}}(s)}, \quad \widehat{W} := g^{\alpha w_{\mathsf{c}}(s)}$

$Y := g^{y_{\mathsf{c}}(s)}, \quad \widehat{Y} := g^{\alpha y_{\mathsf{c}}(s)}$

$B := g^{\beta_v v_{\mathsf{c}}(s) + \beta_w w_{\mathsf{c}}(s) + \beta_y y_{\mathsf{c}}(s)}$

$\pi := (H, \widehat{H}, V_{\mathsf{mid}}, \widehat{V}_{\mathsf{mid}}, W, \widehat{W}, Y, \widehat{Y}, B)$

Ver(crs, u, π)

Extractability check:

$\mathsf{e}(H, g^\alpha) = \mathsf{e}(g, g^{\widehat{H}})$

$\mathsf{e}(V_{\mathsf{mid}}, g^\alpha) = \mathsf{e}(g, g^{\widehat{V}_{\mathsf{mid}}})$

$\mathsf{e}(W, g^\alpha) = \mathsf{e}(g, g^{\widehat{W}})$

$\mathsf{e}(Y, g^\alpha) = \mathsf{e}(g, g^{\widehat{Y}})$

Divisibility check:

$\mathsf{e}(H, g^{t(s)}) = \mathsf{e}(V, W)/\mathsf{e}(Y, g)$

Linear span check:

$\mathsf{e}(B, g) = \mathsf{e}(V, g^{\beta_v}) \cdot \mathsf{e}(W, g^{\beta_w}) \cdot \mathsf{e}(Y, g^{\beta_y})$

Adding ZERO-KNOWLEDGE

Note that if the polynomial t(x) divides V(x)W(x) – Y(x) then, t(x) also divides the following for randomly chosen d_v,d_w and d_y:

$$[V(x) + d_v t(x)] [W(x) + d_w t(x)] - [Y(x) + d_y t(x)]$$

prover just uses some random values $\delta_v, \delta_w, \delta_y \in \mathbb{F}$:

$$
\begin{aligned}
v'_{\mathsf{mid}}(x) &:= v_{\mathsf{mid}}(x) + \delta_v t(x), \\
w'_{\mathsf{c}}(x) &:= w_{\mathsf{c}}(x) + \delta_w t(x), \\
y'_{\mathsf{c}}(x) &:= y_{\mathsf{c}}(x) + \delta_y t(x), \\
h'(x) &= h(x) + \delta_v w_{\mathsf{c}}(x) + \delta_w v_{\mathsf{c}}(x) + \delta_v \delta_w t^2(x) - \delta_y t(x).
\end{aligned}
$$

SUMMARY

- Interactive Proofs
  - Enables us go beyond NP (if NP ≠ PSPACE)
- Zero-Knowledge Proofs (ZKP)
  - For all NP (eg. G3C)
- Non-interactive ZKP for DLP
  - Using the Fiat-Shamir Heuristic
- ZK-SNARKs
  - Zero-Knowledge Succinct Non-Interactive Arguments of knowledge

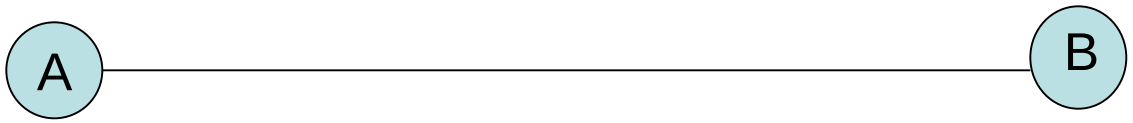
Thank you

Any questions?

OBLIVIOUS TRANSFER (OT)
And
SECURE TWO PARTY COMPUTATION

Oblivious Transfer (OT)

Can A learn only the bit $\mathbf{b_i}$ without revealing ‘i’ to B?



Index i

Bit-Array: $b_0, b_1, \ldots b_{n-1}$

Why is Perfect OT Impossible?

- B’s view of any protocol needs to be identical for index i=0 as well as i=1 (assume n=2)
- Therefore, B can simulate its view even without interacting with A!
- Similar case with A too
- The protocol cannot begin to send “useful” information at any stage in either direction!

A Cryptographic Solution to OT

OT Protocol

Assume f to be a trapdoor one-way permutation, and let B have the trapdoor

- Step 1: A chooses a random bit-array X = $[r_0,\ldots,r_{n-1}]$ and replaces r_i with $f(r_i)$ and sends this array X to B

OT Protocol (Contd.)

- Step 2a: B computes f^{-1} of every element in array X to obtain array $Y = [f^{-1}(r_0), \ldots, r_i, \ldots, f^{-1}(r_{n-1})]$

- Step 2b: B obtains a bit-array Z by computing element-wise XOR of the array H of hardcore predicates of Y, i.e., h(Y[i]) and the input array $[b_0, \ldots, b_{n-1}]$. He then sends the array Z to A.

OT Protocol (Contd.)

- Step 3: On receipt of the array Z, A recovers the bit b_i

$$b_i = Z[i] \text{ XOR } h(r_i)$$

Founding Private Computing on OT

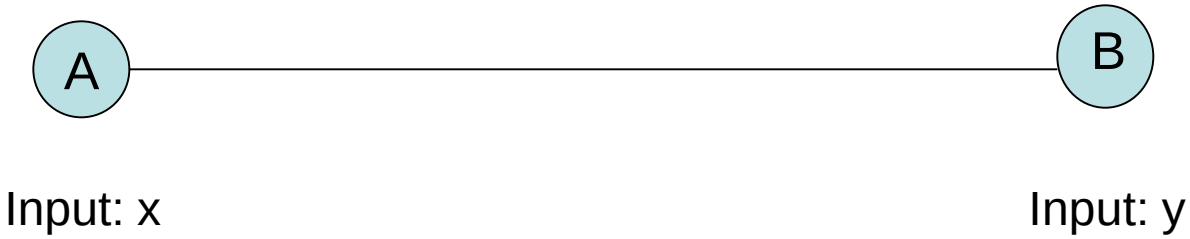
For us, privacy means …

- The Setting
  - Two players collaborating towards a common computational goal (via a protocol)
  - Each player has some local input to the protocol
  - Each player receives his designated output
- Privacy
  - No more information about the local inputs must be revealed (during the protocol execution) than what is revealed by the output

A “Textbook” Example

- **Yao’s Millionaire problem**
  - **Two rich people wish to find out who between them is richer without revealing their respective assets’ values to each other**

Privately Computing f(x,y)



f() is an arbitrary bi-variate Boolean function

Direct Solution Via OT

Privately Computing f(x,y) (Contd.)

Input: x

Input: y

Array **F = [f(0,y), f(1,y), …, f(n-1,y)]**
for all n possibilities of x

Execute OT for A to (privately) obtain the value f(x,y)

The Challenge?

Number of possible values for x grows exponentially in the size of x.

Solution

- **Represent f() as a circuit of smaller functions with small inputs.**

- **Famous Solution: Use an AND and XOR circuit (of fan-in 2, that is, just 4 possible inputs) that computes f().**

Computing f(x,y) (Contd.)

- Step 1a: Express the function by a Boolean circuit using only AND and XOR gates (fan-in 2 and fan-out 1)

- Step 1b : Each player has a variable associated with each wire (connecting two gates) in the circuit

Computing f(x,y) (Contd.)

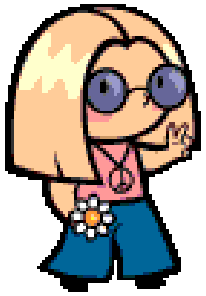
Initializing input variables

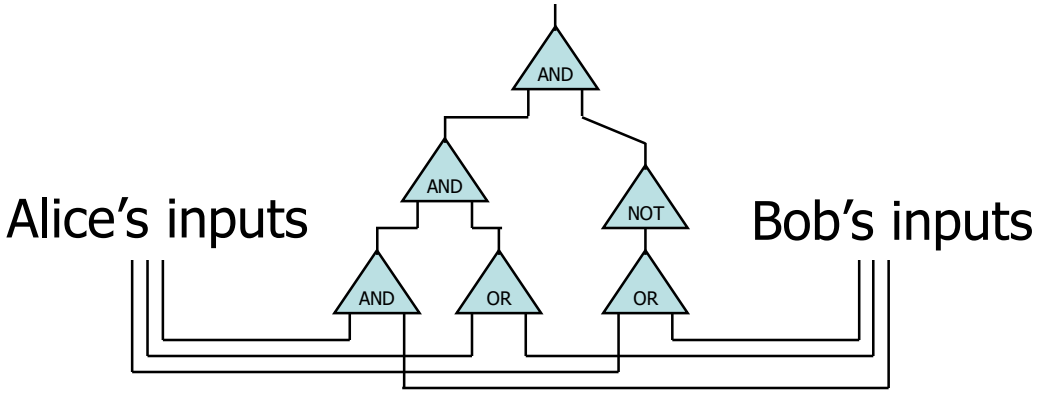
- Step 2a: For all input wires carrying y, A stores a random bit y_1 and sends it to B who stores the bit $y_2 = y \text{ xor } y_1$ for that wire

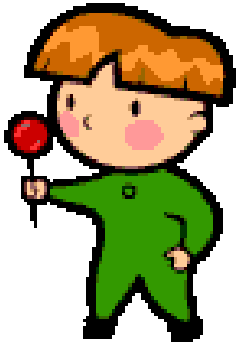
- Step 2b: For all input wires carrying x, A stores a random bit y_1 and sends it to B who stores the bit $y_2 = y \text{ xor } y_1$ for that wire

Protocol Description

- f : polynomial time functionality
- Input : (x, y) Output : f(x, y)
- First, convert the function into a Boolean circuit C.







- Next evaluate <u>one gate</u> securely.
  - Later, generalize to the entire circuit.

Computing f(x,y) (Contd.)

Evaluating XOR gates

- For each xor gate for which the input variables are known to players, each player does the following
  - the output variable is just the xor of the two input variables (this computation is easily done locally by the players)

Computing f(x,y) (Contd.)

Evaluating AND gates

- For each AND gate for which the input variables are known to players, say a_1,a_2 for player A (with output wire c_1) and b_1, b_2 for player B (with output wire c_2):
  - Player A assigns a random bit to c_1
  - Goal: To privately compute c_2 s.t.

$(c_1 \text{ xor } c_2) = (a_1 \text{ xor } b_1) \text{ AND } (a_2 \text{ xor } b_2)$

Evaluating AND gates (Contd.)

- Player A creates a four-sized bit array as follows:
  - $b_0 = c_1$ xor (a_1 AND a_2)
  - $b_1 = c_1$ xor (a_1 AND a_2^c)
  - $b_2 = c_1$ xor (a_1^c AND a_2)
  - $b_3 = c_1$ xor (a_1^c AND a_2^c)

Evaluating AND gates (Contd.)

- Player B chooses the index i as follows and does an OT with A, the output of which is assigned to the variable c_2

- i = 0 if b_1=0 and b_2=0
- i = 1 if b_1=0 and b_2=1
- i = 2 if b_1=1 and b_2=0
- i = 3 if b_1=1 and b_2=1

Securely Computing Any Function in GF(2)

- Any circuit can thus be securely evaluated gate-by-gate till the output wire is reached;

- The variables associated with output wires from the two players can be XORed to obtain f(x,y)

THANK YOU

Questions?