Week 6

An Introduction to Cyber Security – CS 573

Instructor: Dr. Edward G. Amoroso
eamoroso@tag-cyber.com

Required Week Six Readings

1. "Why Cryptosystems Fail," Ross Anderson
https://www.cl.cam.ac.uk/~rja14/Papers/wcf.pdf

2. **Finish Reading "*From CIA to APT: An Introduction" to Cyber Security*, E. Amoroso & M. Amoroso**

Twitter: @hashtag_cyber
LinkedIn: Edward Amoroso

Week 6: Symmetric Cryptography

Reminder: What are the
Three Methods of Cryptanalysis?

Three Cryptanalytic Methods

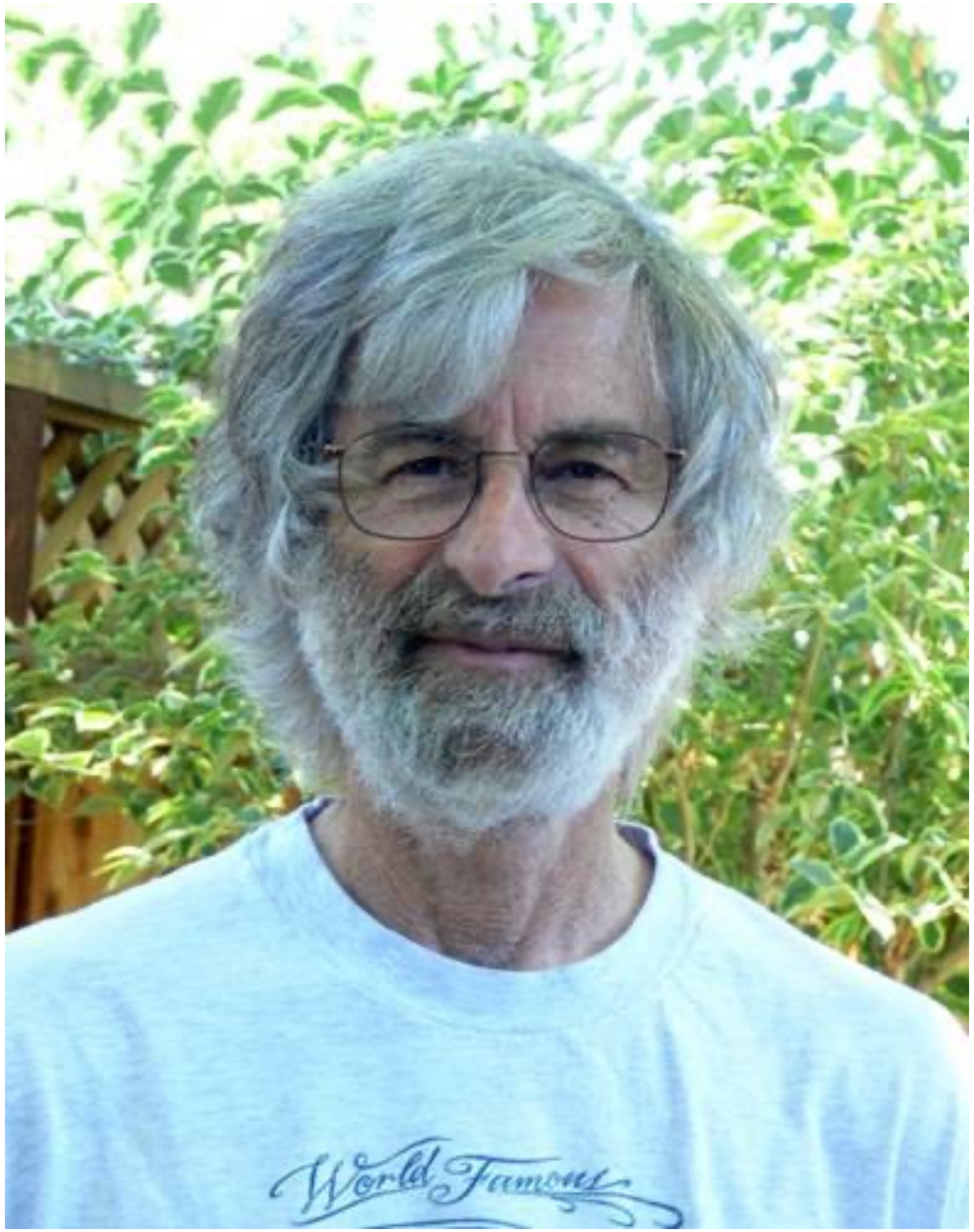
Ciphertext-Only: In this method of cryptanalysis, the attacker has access only to a collection of ciphertexts.

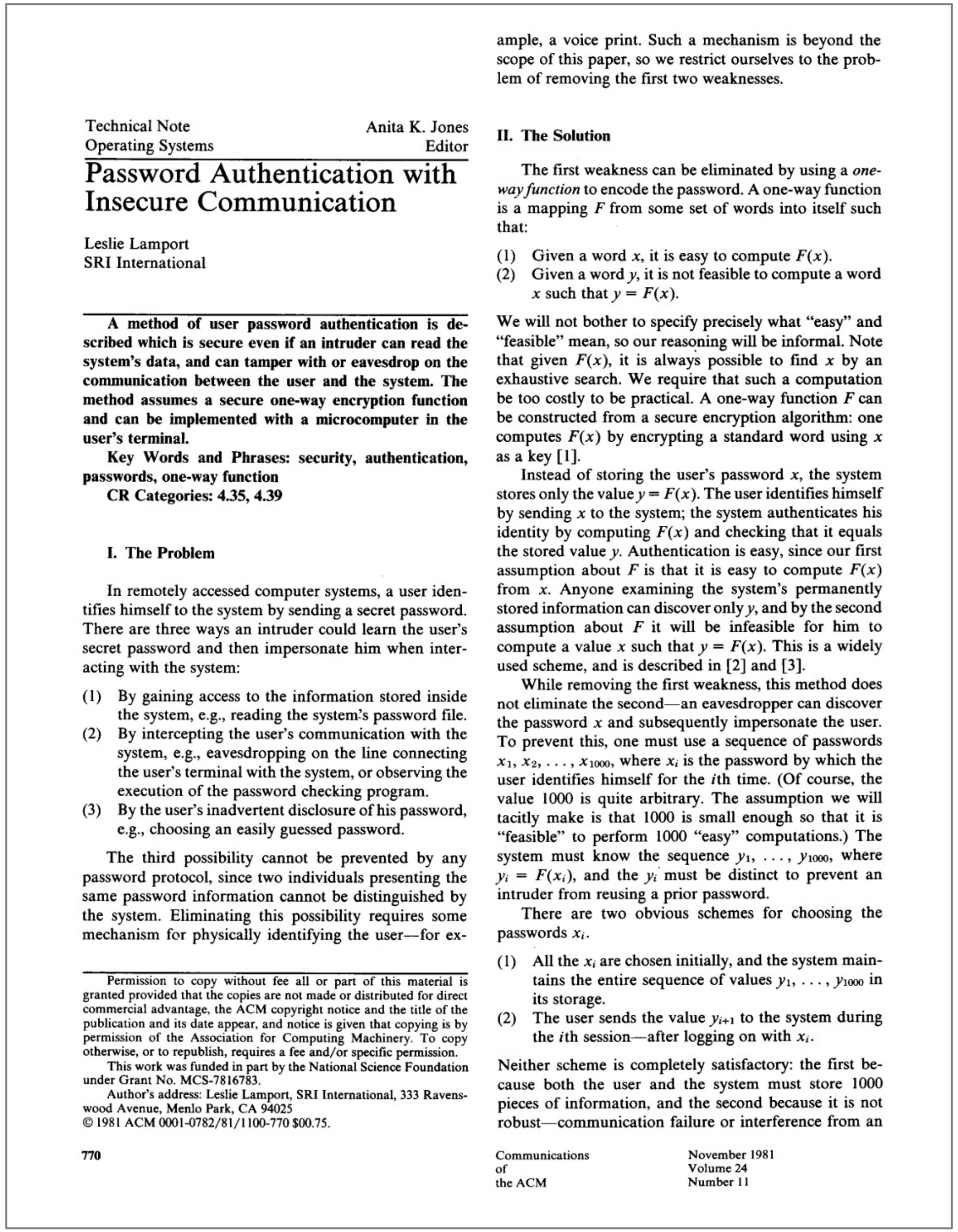
Known-Plaintext: In this method of cryptanalysis, the attacker has a set of ciphertexts to which they know the corresponding plaintext.

Chosen-Plaintext: In the codebook method, the attacker can obtain the ciphertexts (plaintexts) corresponding to an arbitrary set of plaintexts (ciphertexts) of their own choosing.

Are Authentication Protocols with No Challenge Values Always Ciphertext-Only?

Week 6





Technical Note
Operating Systems

Anita K. Jones
Editor

Password Authentication with Insecure Communication

Leslie Lamport
SRI International

A method of user password authentication is described which is secure even if an intruder can read the system's data, and can tamper with or eavesdrop on the communication between the user and the system. The method assumes a secure one-way encryption function and can be implemented with a microcomputer in the user's terminal.

Key Words and Phrases: security, authentication, passwords, one-way function

CR Categories: 4.35, 4.39

I. The Problem

In remotely accessed computer systems, a user identifies himself to the system by sending a secret password. There are three ways an intruder could learn the user's secret password and then impersonate him when interacting with the system:

(1) By gaining access to the information stored inside the system, e.g., reading the system's password file.
(2) By intercepting the user's communication with the system, e.g., eavesdropping on the line connecting the user's terminal with the system, or observing the execution of the password checking program.
(3) By the user's inadvertent disclosure of his password, e.g., choosing an easily guessed password.

The third possibility cannot be prevented by any password protocol, since two individuals presenting the same password information cannot be distinguished by the system. Eliminating this possibility requires some mechanism for physically identifying the user—for example, a voice print. Such a mechanism is beyond the scope of this paper, so we restrict ourselves to the problem of removing the first two weaknesses.

II. The Solution

The first weakness can be eliminated by using a *one-way function* to encode the password. A one-way function is a mapping F from some set of words into itself such that:

(1) Given a word x, it is easy to compute $F(x)$.
(2) Given a word y, it is not feasible to compute a word x such that $y = F(x)$.

We will not bother to specify precisely what "easy" and "feasible" mean, so our reasoning will be informal. Note that given $F(x)$, it is always possible to find x by an exhaustive search. We require that such a computation be too costly to be practical. A one-way function F can be constructed from a secure encryption algorithm: one computes $F(x)$ by encrypting a standard word using x as a key [1].

Instead of storing the user's password x, the system stores only the value $y = F(x)$. The user identifies himself by sending x to the system; the system authenticates his identity by computing $F(x)$ and checking that it equals the stored value y. Authentication is easy, since our first assumption about F is that it is easy to compute $F(x)$ from x. Anyone examining the system's permanently stored information can discover only y, and by the second assumption about F it will be infeasible for him to compute a value x such that $y = F(x)$. This is a widely used scheme, and is described in [2] and [3].

While removing the first weakness, this method does not eliminate the second—an eavesdropper can discover the password x and subsequently impersonate the user. To prevent this, one must use a sequence of passwords $x_1, x_2, \ldots, x_{1000}$, where x_i is the password by which the user identifies himself for the ith time. (Of course, the value 1000 is quite arbitrary. The assumption we will tacitly make is that 1000 is small enough so that it is "feasible" to perform 1000 "easy" computations.) The system must know the sequence $y_1, \ldots, y_{1000}$, where $y_i = F(x_i)$, and the y_i must be distinct to prevent an intruder from reusing a prior password.

There are two obvious schemes for choosing the passwords x_i.

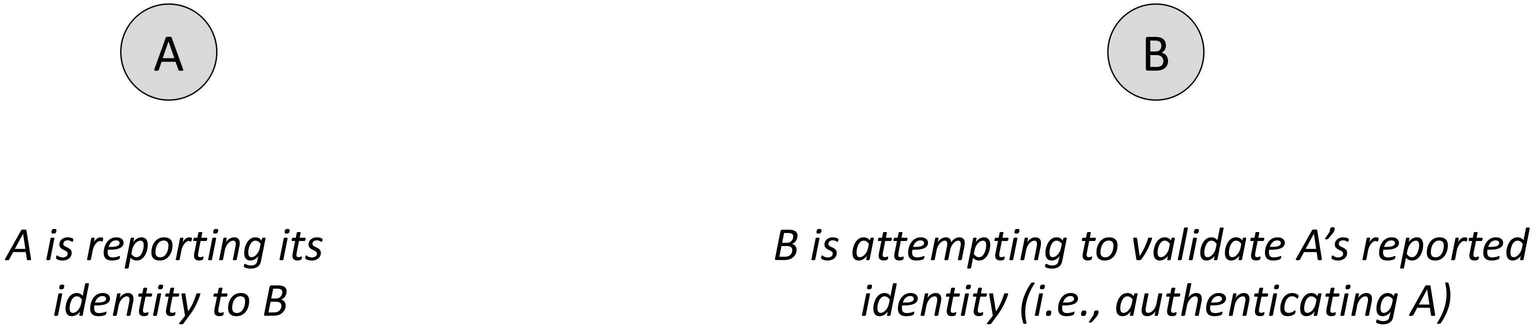
(1) All the x_i are chosen initially, and the system maintains the entire sequence of values $y_1, \ldots, y_{1000}$ in its storage.
(2) The user sends the value y_{i+1} to the system during the ith session—after logging on with x_i.

Neither scheme is completely satisfactory: the first because both the user and the system must store 1000 pieces of information, and the second because it is not robust—communication failure or interference from an

This work was funded in part by the National Science Foundation under Grant No. MCS-7816783.

Author's address: Leslie Lamport, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025
© 1981 ACM 0001-0782/81/1100-770 $00.75.

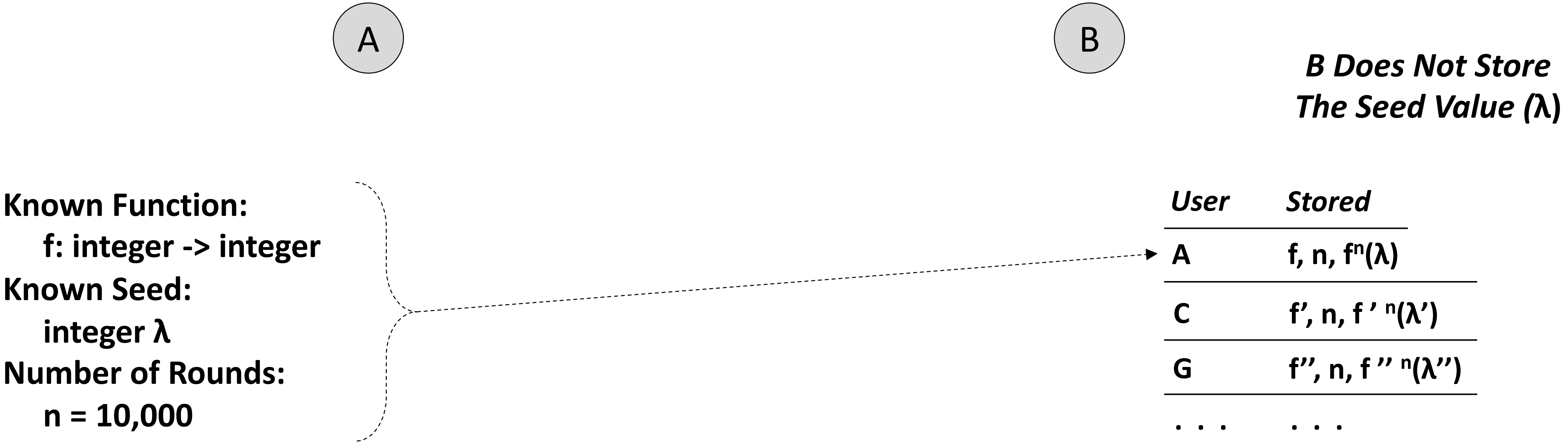
Lamport S/Key Protocol – Purpose



A is reporting its identity to B

B is attempting to validate A's reported identity (i.e., authenticating A)

Lamport S/Key Protocol – Set-Up

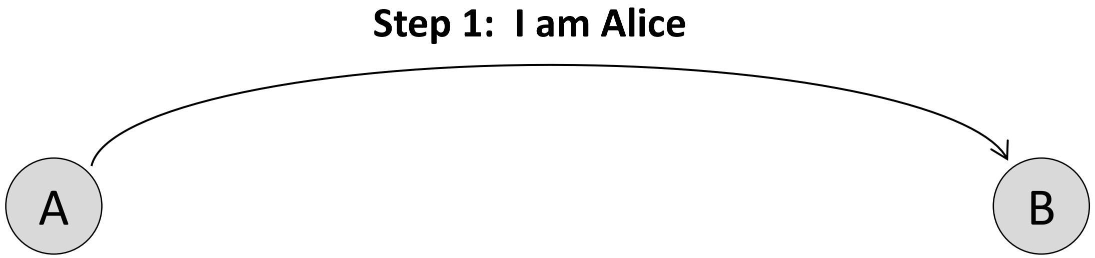


B Does Not Store The Seed Value (λ)

Known Function:
f: integer -> integer
Known Seed:
integer λ
Number of Rounds:
n = 10,000

User	*Stored*
A	**$f, n, f^{n}(\lambda)$**
C	**$f', n, f'^{\,n}(\lambda')$**
G	**$f'', n, f''^{\,n}(\lambda'')$**
. . .	**. . .**

Lamport S/Key Protocol



Step 1: I am Alice

Known Function:

f: integer -> integer

Known Seed:

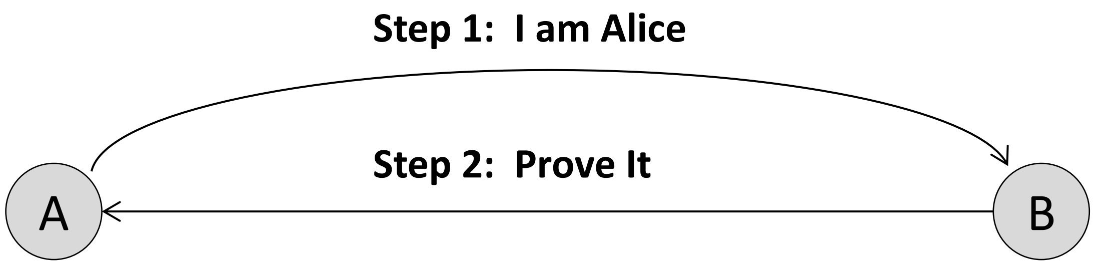
integer λ

Number of Rounds:

n = 10,000

User	*Stored*
A	**f, n, $f^n(\lambda)$**

Lamport S/Key Protocol



Step 1: I am Alice

Step 2: Prove It

Known Function:

f: integer -> integer

Known Seed:

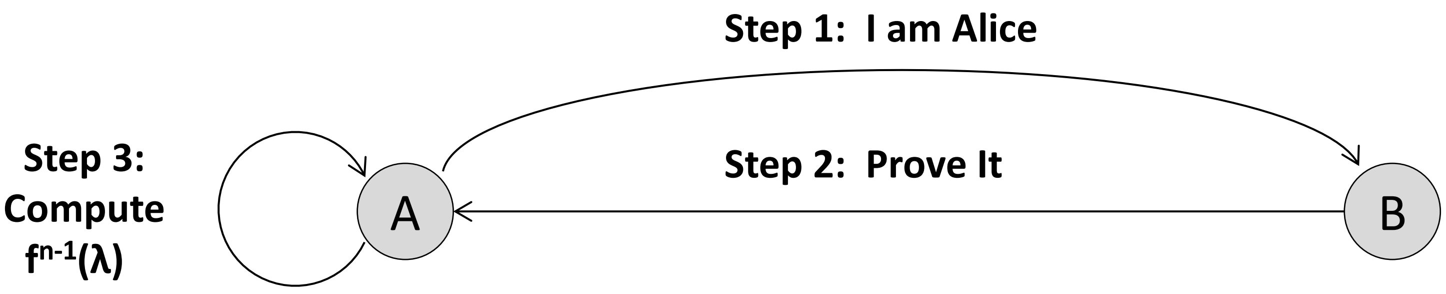
integer λ

Number of Rounds:

n = 10,000

User	***Stored***
A	**f, n, $f^n(\lambda)$**

Lamport S/Key Protocol

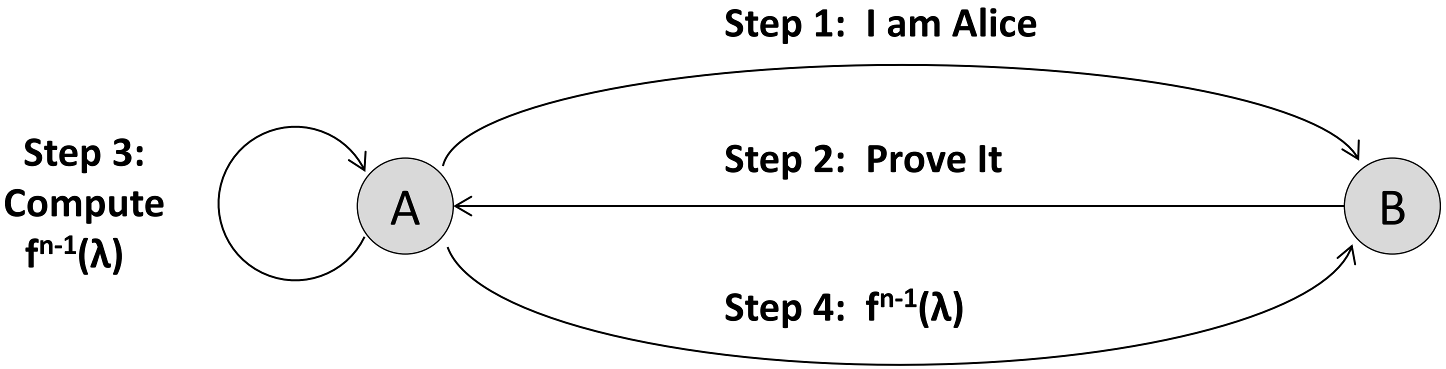


Known Function:
f: integer -> integer
Known Seed:
integer λ
Number of Rounds:
n = 10,000

User	*Stored*
A	**f, n, $f^n(\lambda)$**

Week 6

Lamport S/Key Protocol



Step 1: I am Alice

Step 2: Prove It

Step 3: Compute $f^{n-1}(\lambda)$

Step 4: $f^{n-1}(\lambda)$

Known Function:

f: integer -> integer

Known Seed:

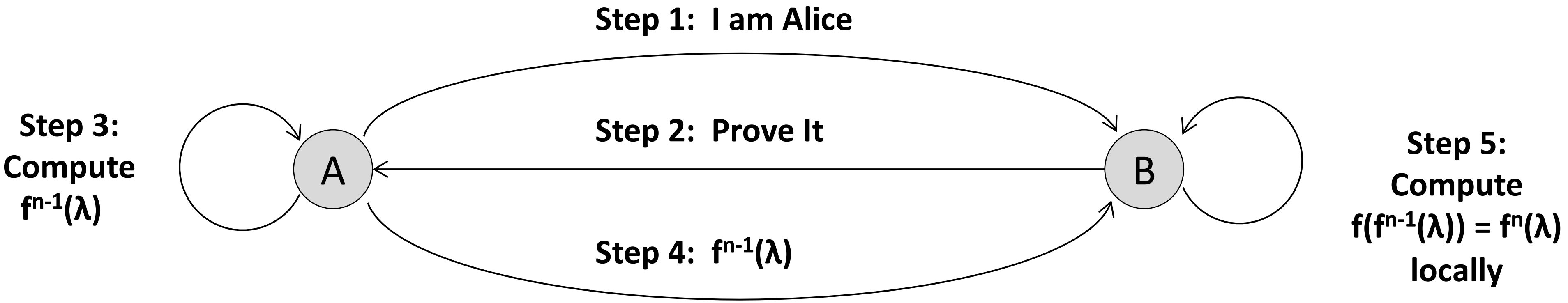
integer λ

Number of Rounds:

n = 10,000

User	*Stored*
A	**f, n, $f^{n}(\lambda)$**

Lamport S/Key Protocol



Step 1: I am Alice

Step 2: Prove It

Step 3: Compute $f^{n-1}(\lambda)$

Step 4: $f^{n-1}(\lambda)$

Step 5: Compute $f(f^{n-1}(\lambda)) = f^{n}(\lambda)$ locally

User	*Stored*
A	f, n, $f^{n}(\lambda)$

Known Function:

f: integer -> integer

Known Seed:

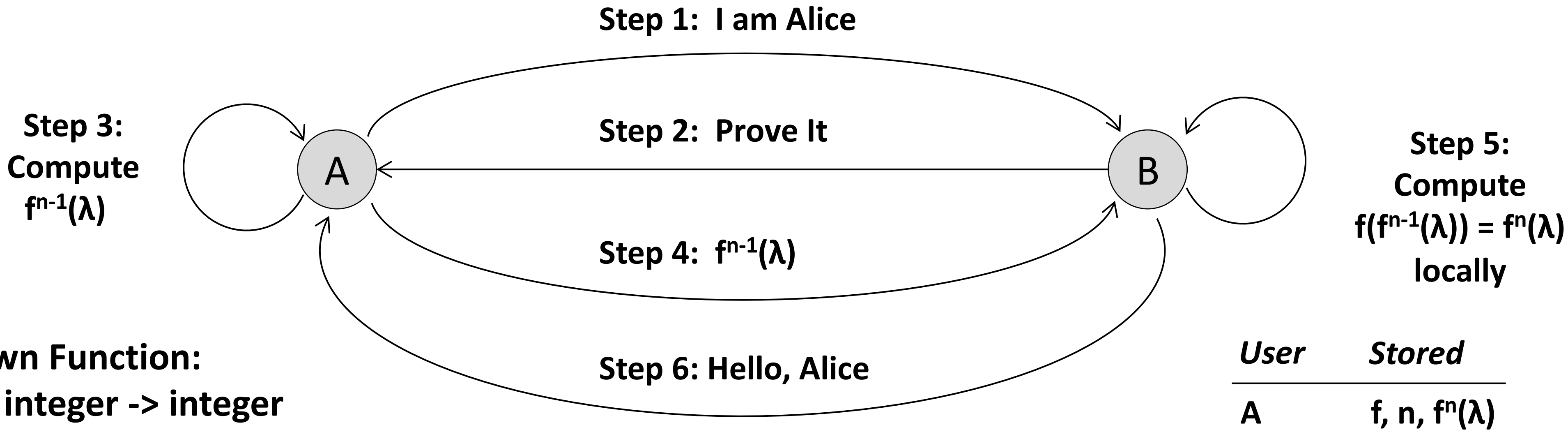
integer λ

Number of Rounds:

n = 10,000

Week 6

Lamport S/Key Protocol



Step 1: I am Alice

Step 2: Prove It

Step 3: Compute $f^{n-1}(\lambda)$

Step 4: $f^{n-1}(\lambda)$

Step 5: Compute $f(f^{n-1}(\lambda)) = f^{n}(\lambda)$ locally

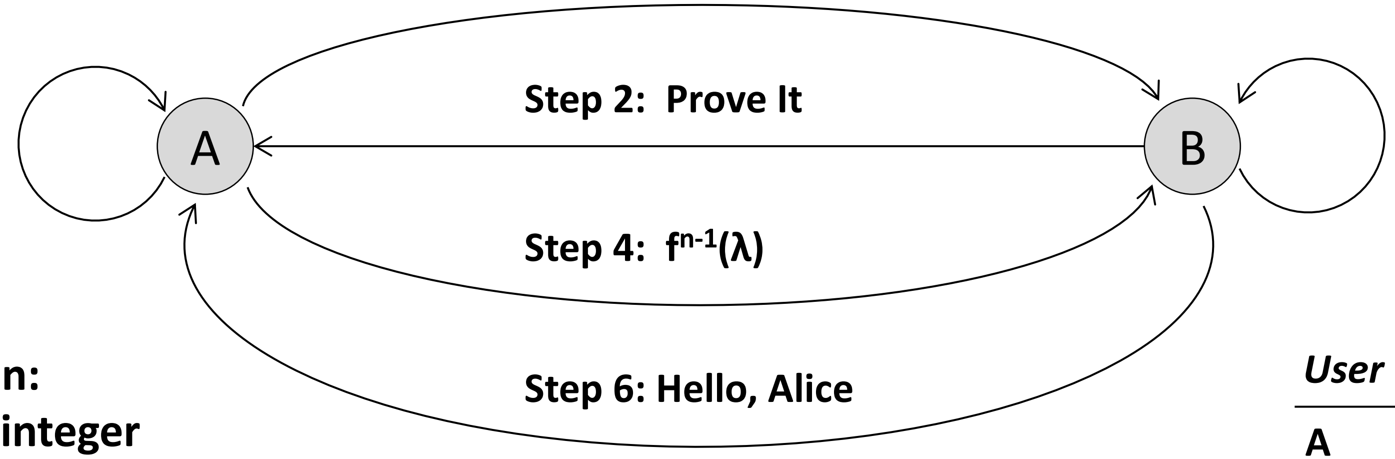
Step 6: Hello, Alice

Known Function:
f: integer -> integer
Known Seed:
integer λ
Number of Rounds:
n = 10,000

User	*Stored*
A	**f, n, $f^{n}(\lambda)$**

Week 6

Lamport S/Key Protocol



Step 1: I am Alice

Step 2: Prove It

Step 3: Compute $f^{n-1}(\lambda)$

Step 4: $f^{n-1}(\lambda)$

Step 5: Compute $f(f^{n-1}(\lambda)) = f^{n}(\lambda)$ locally

Step 6: Hello, Alice

Known Function:

f: integer -> integer

Known Seed:

integer λ

Number of Rounds:

n = 10,000

$f^{n}(\lambda)$ stored

User	*Stored*
A	f, n, $f^{n}(\lambda)$

Lamport S/Key Protocol

Known Function:

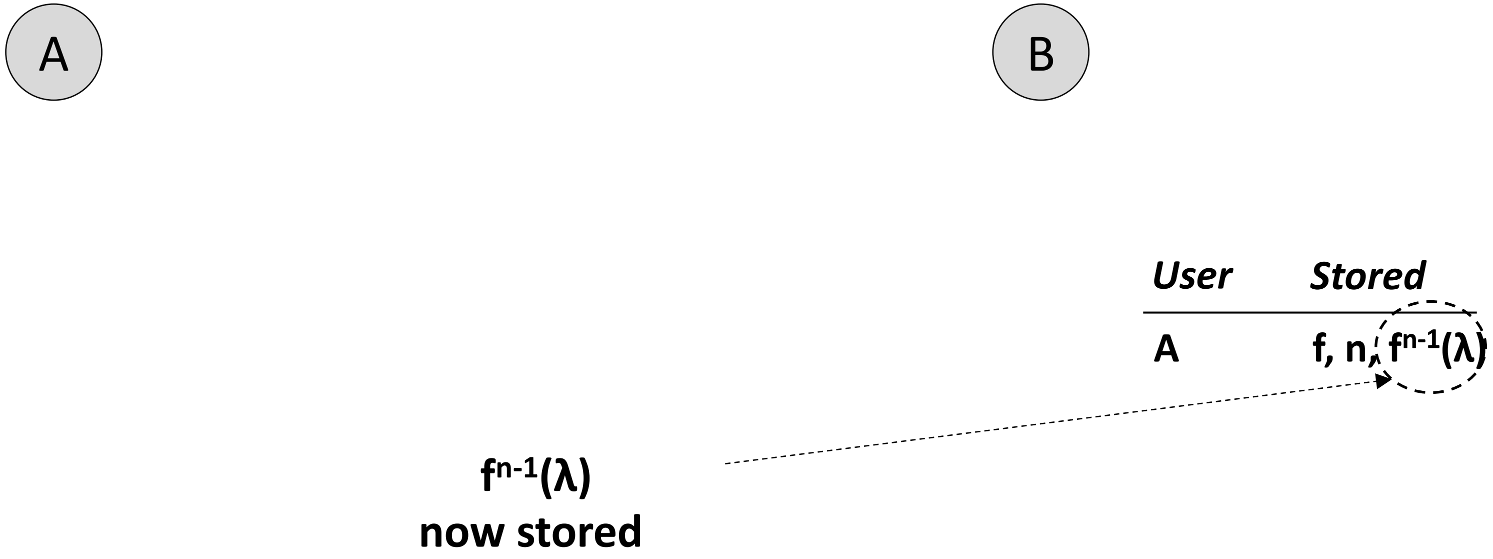
f: integer -> integer

Known Seed:

integer λ

Number of Rounds:

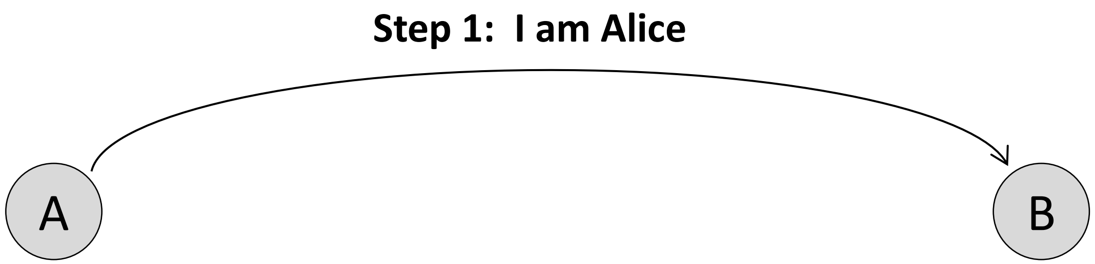
n-1 = 9,999



User	*Stored*
A	f, n, $f^{n-1}(\lambda)$

$f^{n-1}(\lambda)$
now stored

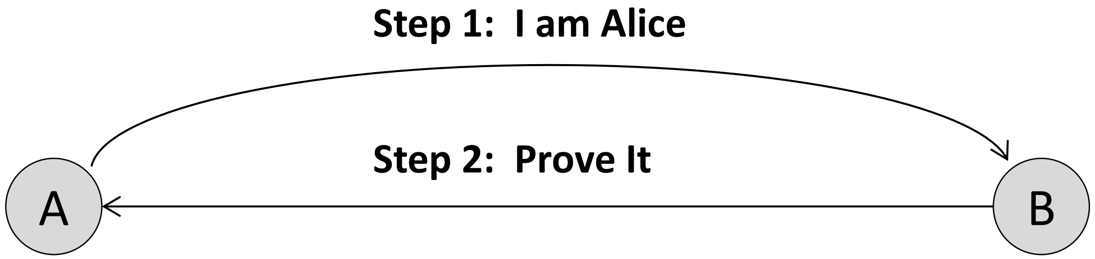
Lamport S/Key Protocol


Step 1: I am Alice

Known Function:
f: integer -> integer
Known Seed:
integer λ
Number of Rounds:
n-1 = 9,999

User	*Stored*
A	**f, n, $f^{n-1}(\lambda)$**

Lamport S/Key Protocol



Known Function:

f: integer -> integer

Known Seed:

integer λ

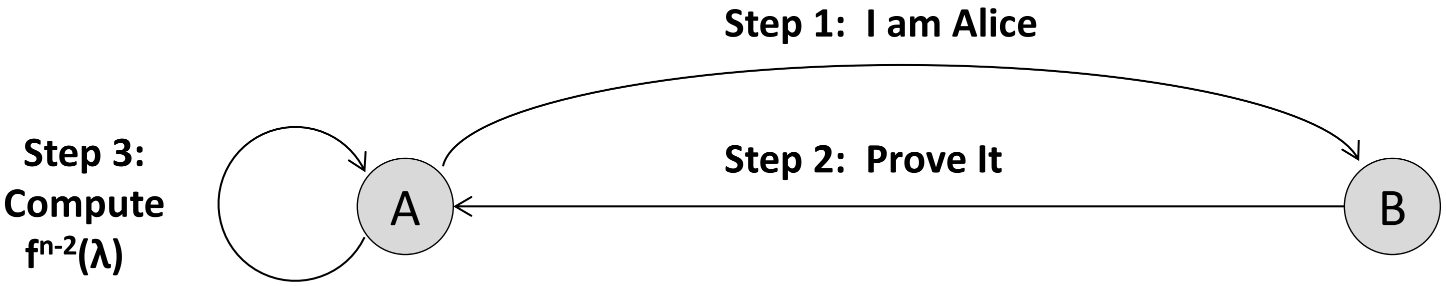
Number of Rounds:

n-1 = 9,999

User	*Stored*
A	**f, n, $f^{n-1}(\lambda)$**

Week 6

Lamport S/Key Protocol



Known Function:

f: integer -> integer

Known Seed:

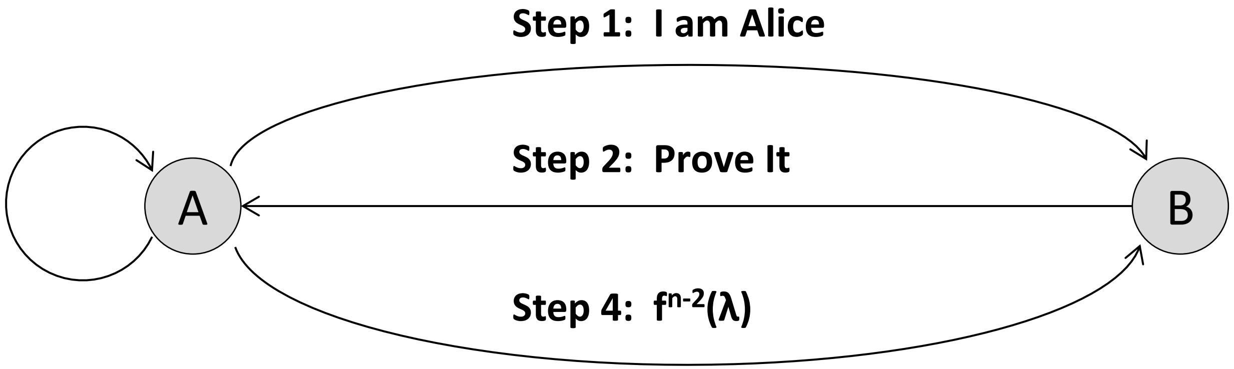
integer λ

Number of Rounds:

n-1 = 9,999

User	*Stored*
A	**f, n, $f^{n-1}(\lambda)$**

Lamport S/Key Protocol


Step 1: I am Alice
Step 2: Prove It
Step 3: Compute $f^{n-2}(\lambda)$
A
B
Step 4: $f^{n-2}(\lambda)$

Known Function:

f: integer -> integer

Known Seed:

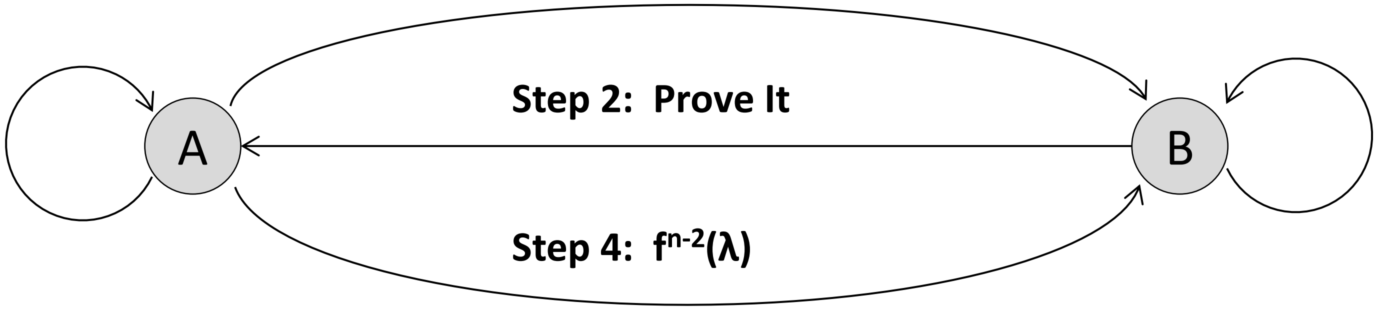
integer λ

Number of Rounds:

n-1 = 9,999

User	*Stored*
A	**f, n, $f^{n-1}(\lambda)$**

Lamport S/Key Protocol



Step 1: I am Alice

Step 2: Prove It

Step 3: Compute $f^{n-2}(\lambda)$

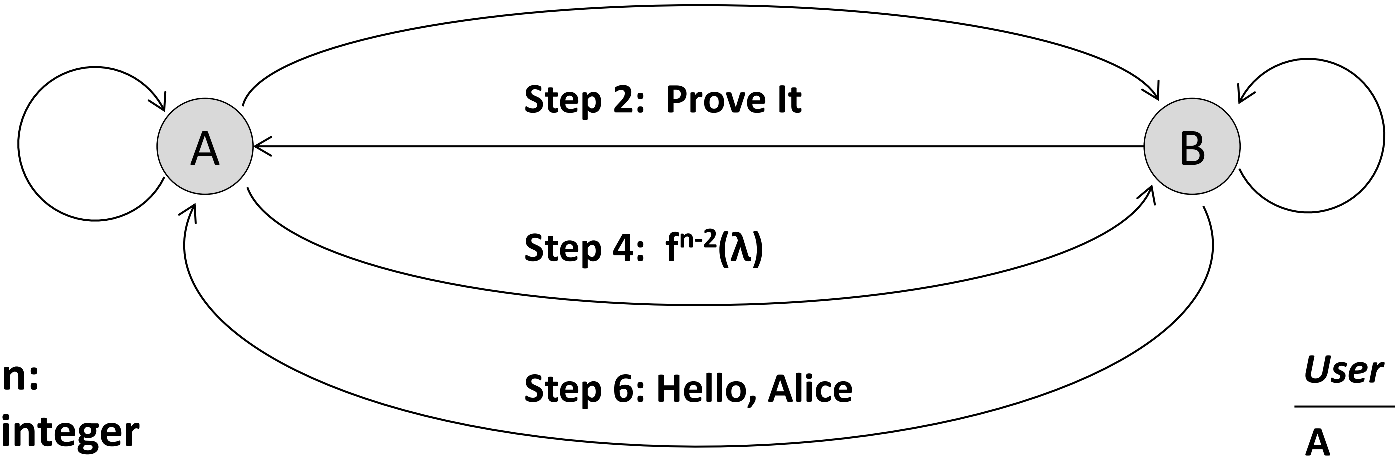
Step 4: $f^{n-2}(\lambda)$

Step 5: Compute $f(f^{n-2}(\lambda)) = f^{n-1}(\lambda)$ locally

Known Function:
  f: integer -> integer
Known Seed:
  integer λ
Number of Rounds:
  n-1 = 9,999

User	*Stored*
A	f, n, $f^{n-1}(\lambda)$

Lamport S/Key Protocol



Step 1: I am Alice

Step 2: Prove It

Step 3: Compute $f^{n-2}(\lambda)$

Step 4: $f^{n-2}(\lambda)$

Step 5: Compute $f(f^{n-2}(\lambda)) = f^{n-1}(\lambda)$ locally

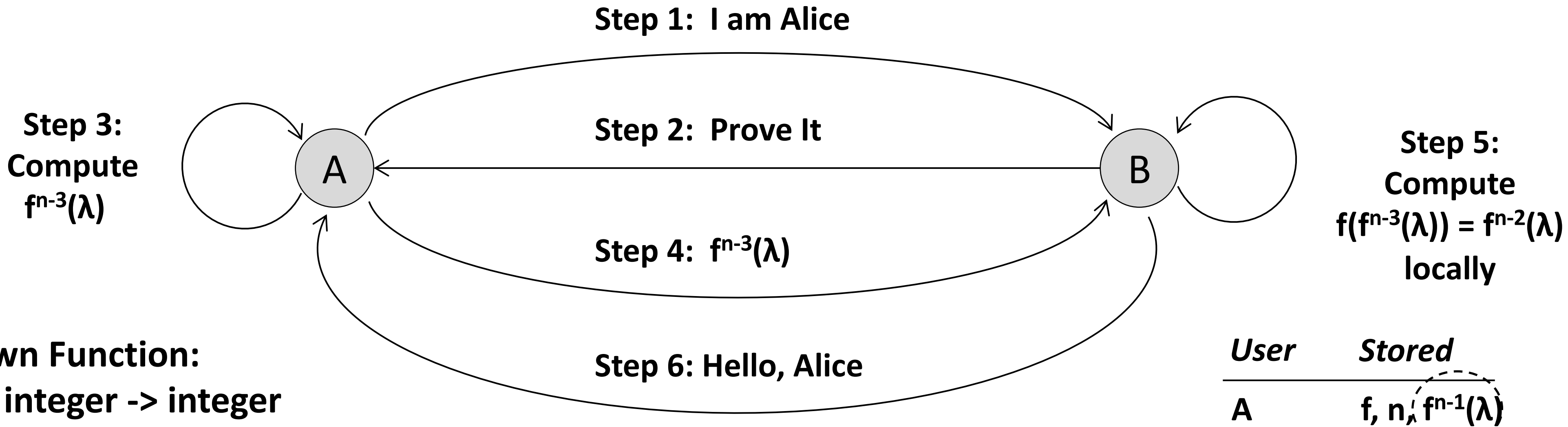
Step 6: Hello, Alice

Known Function:
 f: integer -> integer
Known Seed:
 integer λ
Number of Rounds:
 n-1 = 9,999

User	*Stored*
A	f, n, $f^{n-1}(\lambda)$

Week 6

Lamport S/Key Protocol



Step 1: I am Alice

Step 2: Prove It

Step 3: Compute $f^{n-3}(\lambda)$

Step 4: $f^{n-3}(\lambda)$

Step 5: Compute $f(f^{n-3}(\lambda)) = f^{n-2}(\lambda)$ locally

Step 6: Hello, Alice

User	*Stored*
A	f, n, $f^{n-1}(\lambda)$

Known Function:

f: integer -> integer

Known Seed:

integer λ

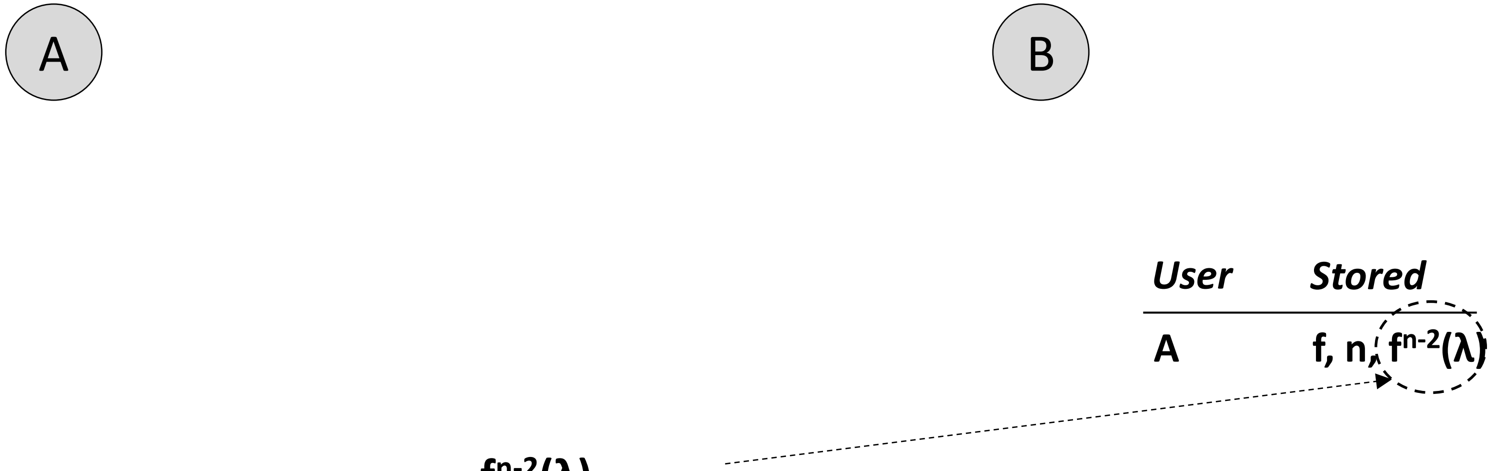
Number of Rounds:

n-1 = 9,998

$f^{n-1}(\lambda)$ stored

Week 6

Lamport S/Key Protocol



User	*Stored*
A	f, n, $f^{n-2}(\lambda)$

Known Function:
f: integer -> integer
Known Seed:
integer λ
Number of Rounds:
n-2 = 9,998

$f^{n-2}(\lambda)$
now stored
(decremented)

Lamport S/Key Protocol – Analysis

	Input	Output
Round 1	-	$f^n(\lambda)$

Lamport S/Key Protocol – Analysis

	Input	Output
Round 1	-	$f^n(\lambda)$
Round 2		$f^{n-1}(\lambda)$

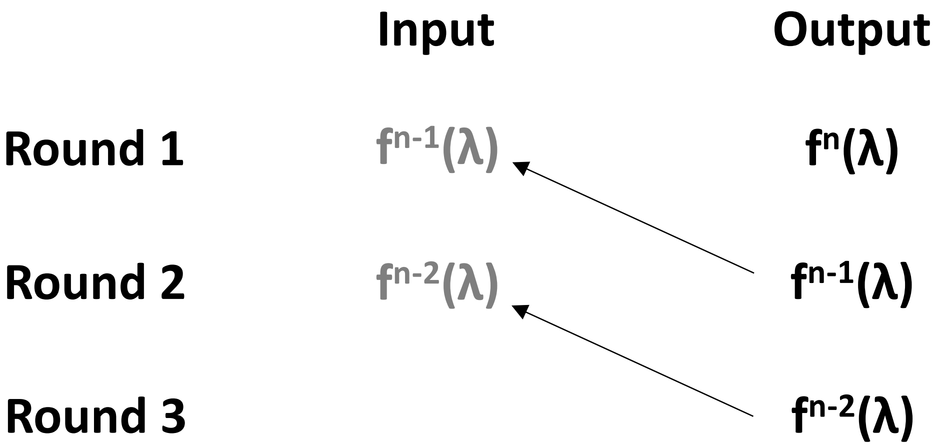
Note:
$f(f^{n-1}(\lambda)) = f^n(\lambda)$

Lamport S/Key Protocol – Analysis

	Input	Output
Round 1	$f^{n-1}(\lambda)$	$f^{n}(\lambda)$
Round 2		$f^{n-1}(\lambda)$

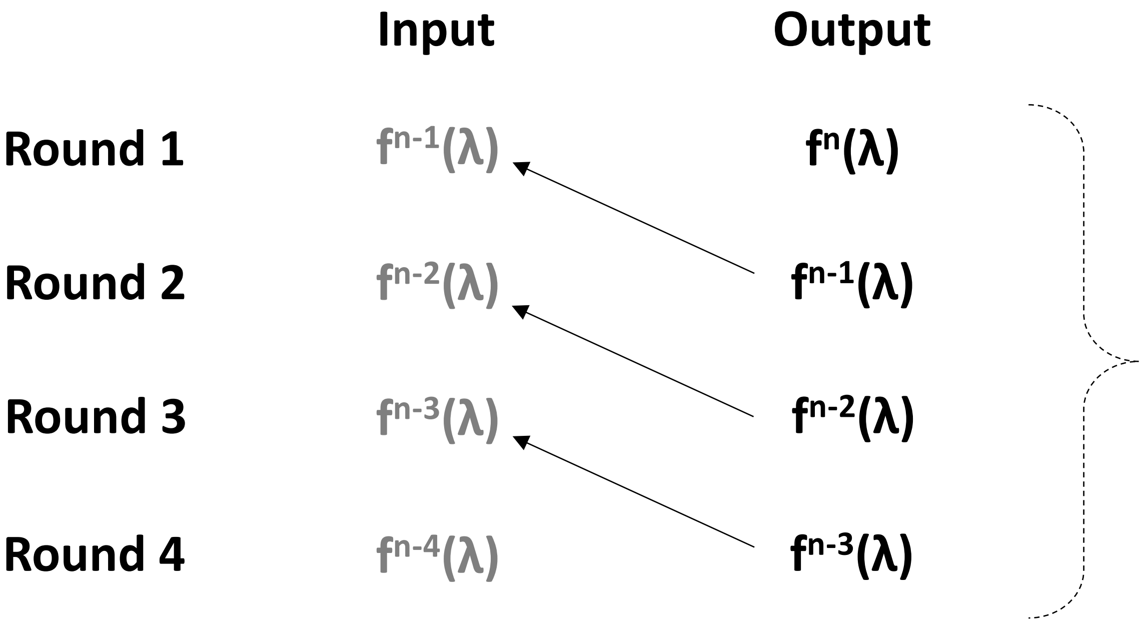
Week 6

Lamport S/Key Protocol – Analysis



	Input	Output
Round 1	$f^{n-1}(\lambda)$	$f^{n}(\lambda)$
Round 2	$f^{n-2}(\lambda)$	$f^{n-1}(\lambda)$
Round 3		$f^{n-2}(\lambda)$

Lamport S/Key Protocol – Analysis

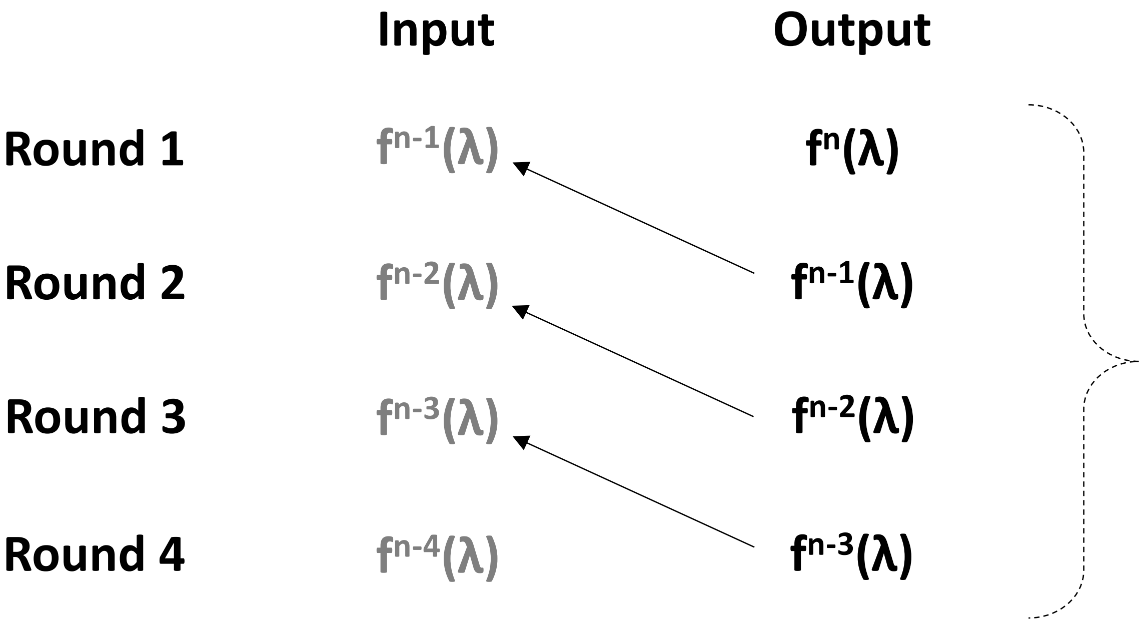


	Input	Output
Round 1	$f^{n-1}(\lambda)$	$f^{n}(\lambda)$
Round 2	$f^{n-2}(\lambda)$	$f^{n-1}(\lambda)$
Round 3	$f^{n-3}(\lambda)$	$f^{n-2}(\lambda)$
Round 4	$f^{n-4}(\lambda)$	$f^{n-3}(\lambda)$

By waiting for successive rounds, observer Eve can see the plaintext for the previous round

Week 6

Lamport S/Key Protocol – Analysis



	Input	Output
Round 1	$f^{n-1}(\lambda)$	$f^{n}(\lambda)$
Round 2	$f^{n-2}(\lambda)$	$f^{n-1}(\lambda)$
Round 3	$f^{n-3}(\lambda)$	$f^{n-2}(\lambda)$
Round 4	$f^{n-4}(\lambda)$	$f^{n-3}(\lambda)$

By waiting for successive rounds, observer Eve can see the plaintext for the previous round

Implies *Known Plaintext* Cryptanalysis

How Does Conventional Cryptography Work?

Definition: Cryptosystem

A cryptosystem is a five-tuple consisting of

- Encryption function E
- Decryption function D
- Set of plaintext elements P
- Set of ciphertext elements C
- Set of cryptographic keys K

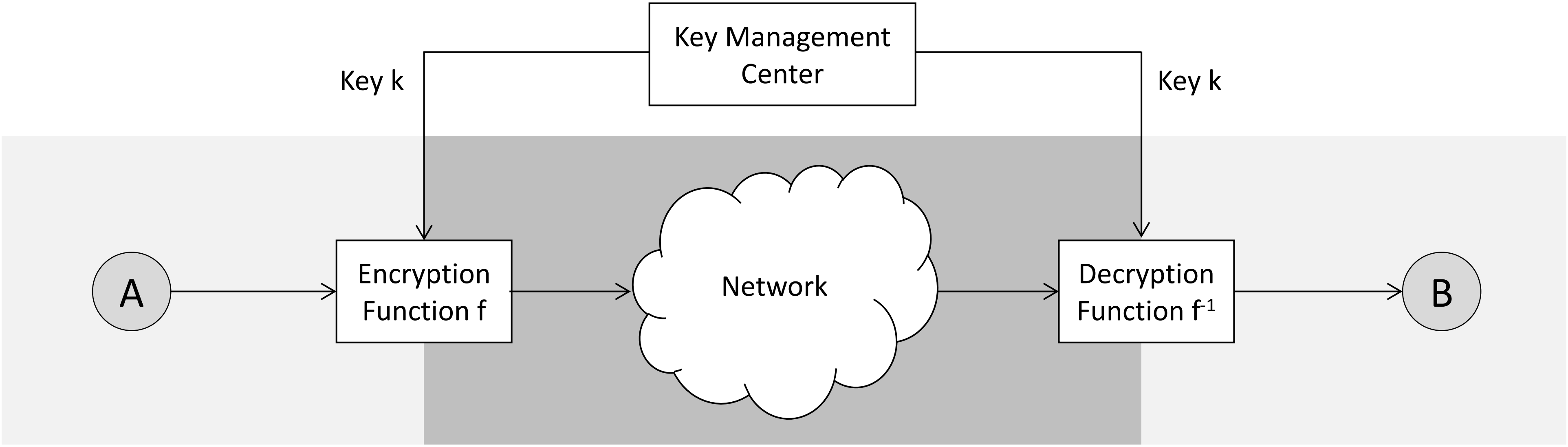
Definition: Cryptosystem

A cryptosystem is a five-tuple consisting of

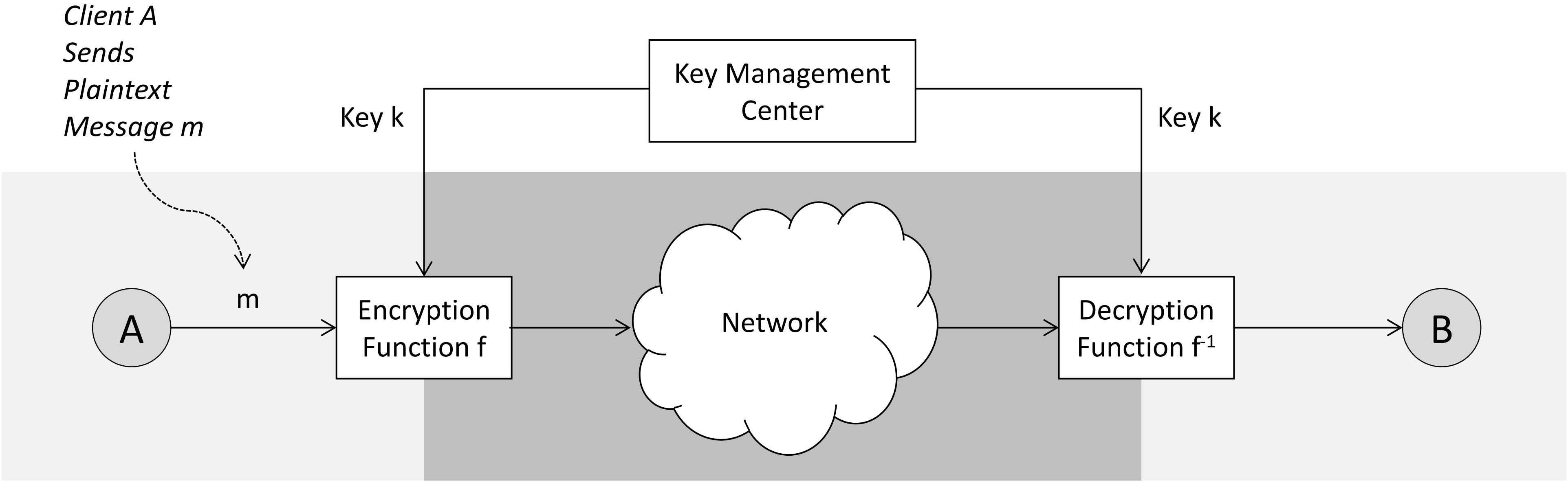
- Encryption function E
- Decryption function D
- Set of plaintext elements P
- Set of ciphertext elements C
- Set of cryptographic keys K

$E(p) = c$	$\{ p \} = c$
$D(c) = p$	$\{ c \} = p$
$D (E (p))$	$\{ \{ p \} \} = p$
$E_k(p) = c$	$\{ p \}_k = c$
$D_k(c) = p$	$\underset{\substack{\uparrow \\ \textit{Decrypt}}}{\{} \underset{\substack{\uparrow \\ \textit{Encrypt}}}{\{} p \}_k \}_k = p$

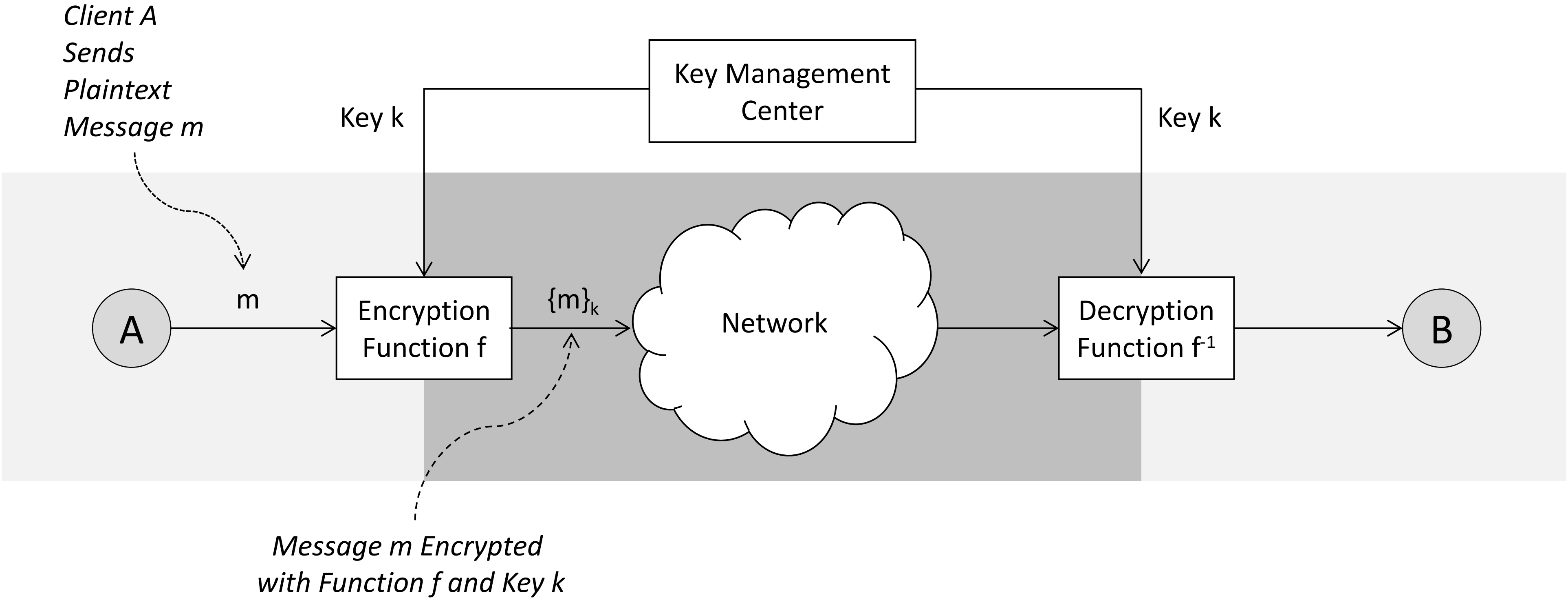
Conventional Encryption Schema



Conventional Encryption Schema



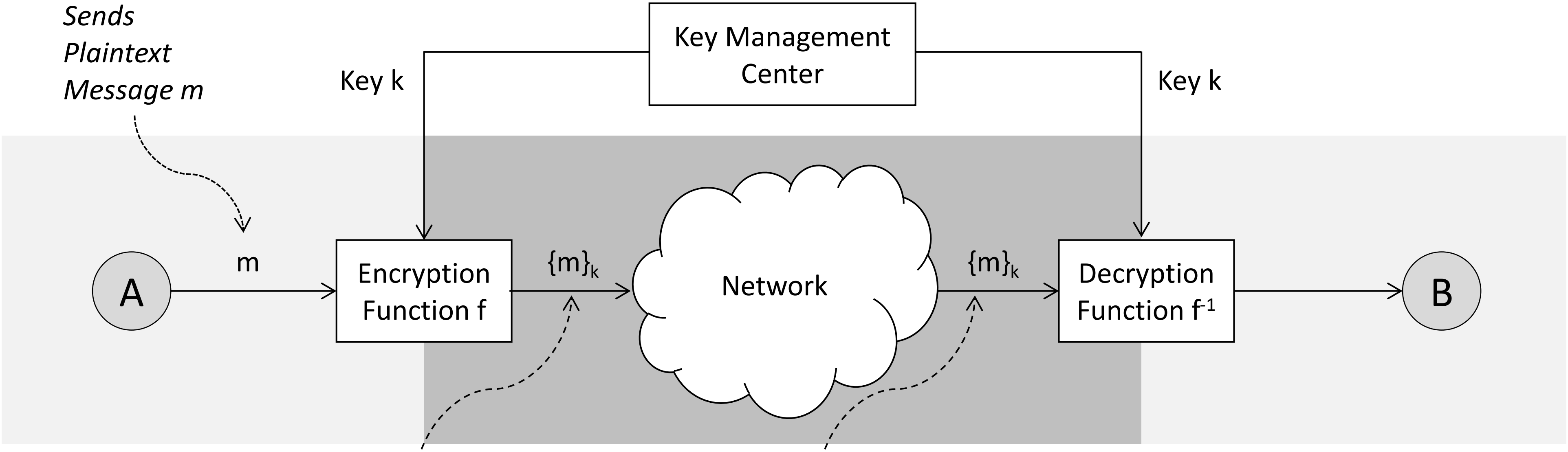
Conventional Encryption Schema



Week 6

Conventional Encryption Schema

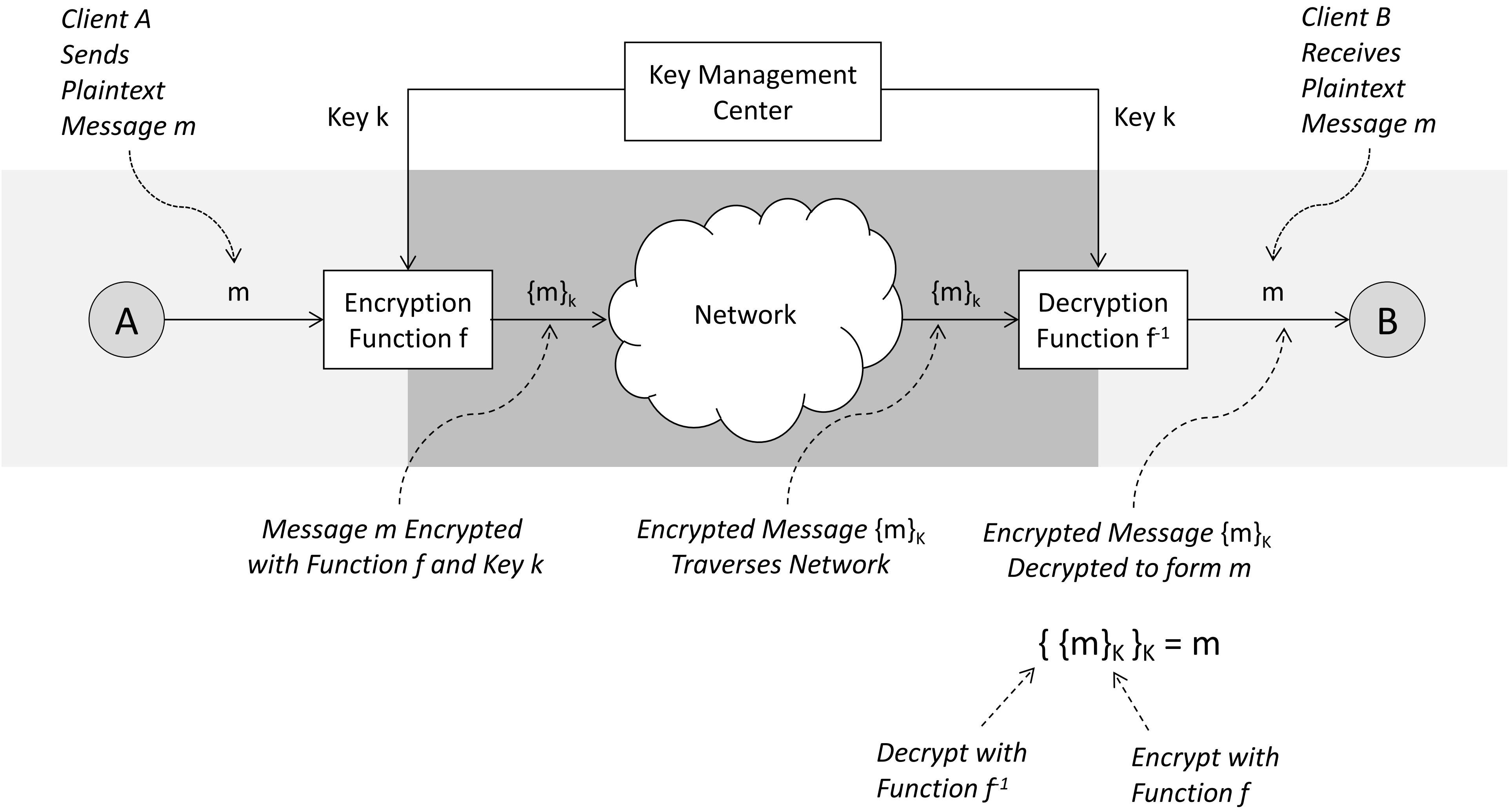
Client A Sends Plaintext Message m



Message m Encrypted with Function f and Key k

Encrypted Message $\{m\}_K$ *Traverses Network*

Conventional Encryption Schema



Client A Sends Plaintext Message m

Client B Receives Plaintext Message m

Message m Encrypted with Function f and Key k

Encrypted Message $\{m\}_K$ Traverses Network

Encrypted Message $\{m\}_K$ Decrypted to form m

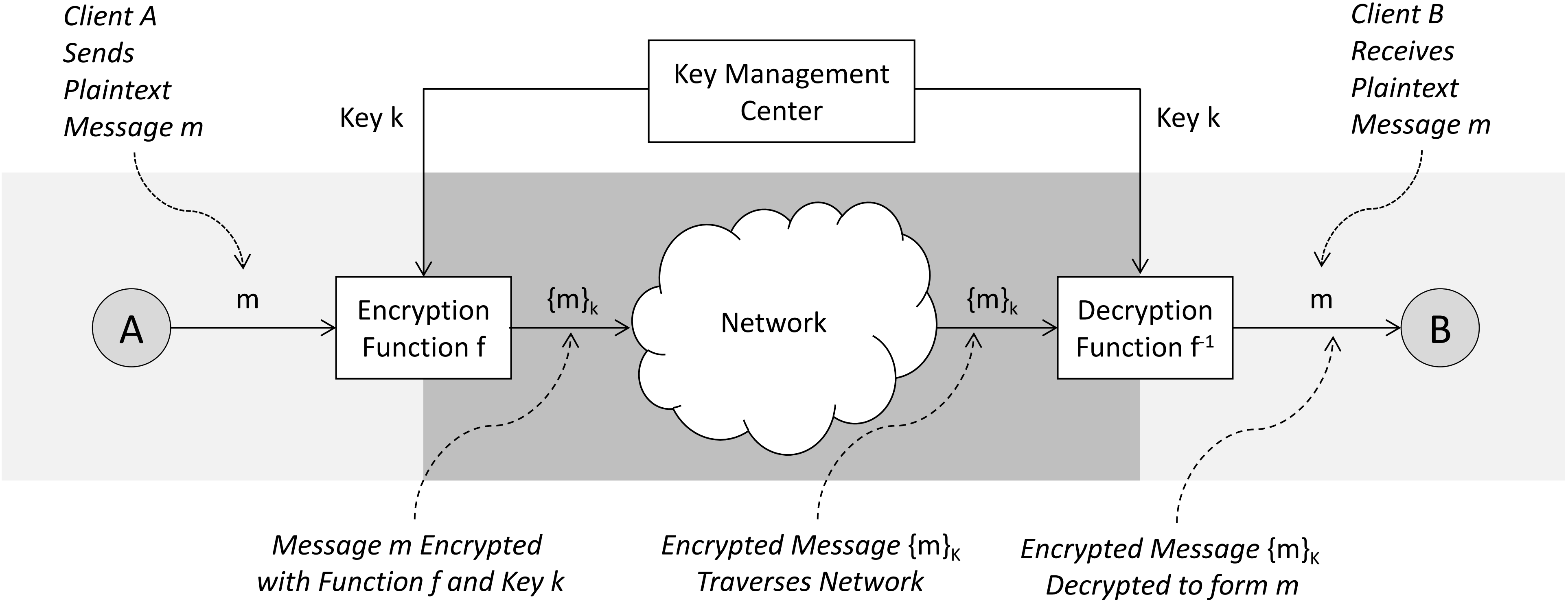
$$\{\, \{m\}_K \,\}_K = m$$

Decrypt with Function f^{-1}

Encrypt with Function f

Week 6

Conventional Encryption Schema



Two Important Security Properties:

1. Secrecy Between A and B
2. Authentication of A by B

$$\{\ \{m\}_K\ \}_K = m$$

Decrypt with Function f^{-1}

Encrypt with Function f

What is the Simplest Example Encryption Algorithm?

XOR Function

1 XOR 1 = 0
0 XOR 0 = 0

1 XOR 0 = 1
0 XOR 1 = 1

Conventional Encryption Algorithm – Simplest Example

Conventional Encryption Algorithm – Simplest Example

XOR Function

```
Plaintext Input     0 0 1 0  0 0 1 1
```

1 XOR 1 = 0
0 XOR 0 = 0

1 XOR 0 = 1
0 XOR 1 = 1

XOR Function

Plaintext Input	0 0 1 0 0 0 1 1
Key	1 1 1 0 1 1 1 0

1 XOR 1 = 0
0 XOR 0 = 0

1 XOR 0 = 1
0 XOR 1 = 1

Conventional Encryption Algorithm – Simplest Example

Conventional Encryption Algorithm – Simplest Example

XOR Function

Plaintext Input	0 0 1 0	0 0 1 1
Key	1 1 1 0	1 1 1 0
Ciphertext	1 1 0 0	1 1 0 1

1 XOR 1 = 0
0 XOR 0 = 0

1 XOR 0 = 1
0 XOR 1 = 1

XOR Function

Plaintext Input	0 0 1 0 0 0 1 1
Key	1 1 1 0 1 1 1 0
Ciphertext	1 1 0 0 1 1 0 1
Key	1 1 1 0 1 1 1 0

1 XOR 1 = 0
0 XOR 0 = 0

1 XOR 0 = 1
0 XOR 1 = 1

Conventional Encryption Algorithm – Simplest Example

Conventional Encryption Algorithm – Simplest Example

XOR Function

Plaintext Input	0 0 1 0 0 0 1 1
Key	1 1 1 0 1 1 1 0
Ciphertext	1 1 0 0 1 1 0 1
Key	1 1 1 0 1 1 1 0
Plaintext Output	0 0 1 0 0 0 1 1

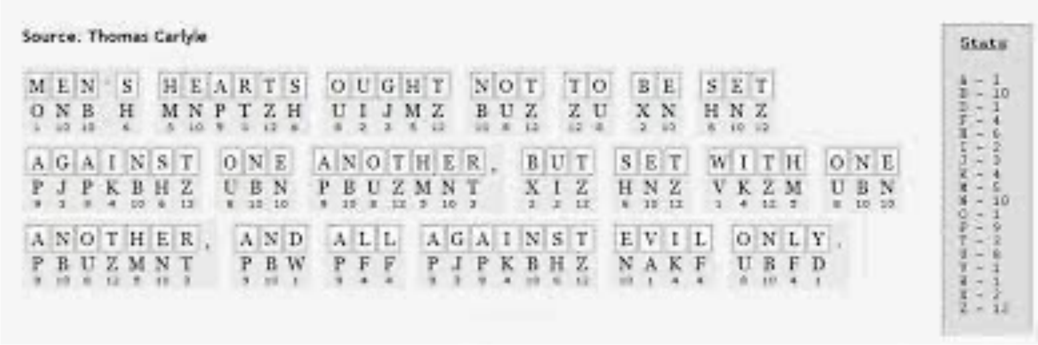
1 XOR 1 = 0
0 XOR 0 = 0

1 XOR 0 = 1
0 XOR 1 = 1

What are the Two Most Basic Design Strategies for Encryption Algorithms?

Week 6

Substitution Cipher – Replacement of one or more things with one or more things (Symmetric versus Asymmetric)



Source: Thomas Carlyle

```
MEN S  HEARTS  OUGHT  NOT  TO  BE  SET
ONB H  MNPTZH  UIJMZ  BUZ  ZU  XN  HNZ

AGAINST  ONE  ANOTHER,  BUT  SET  WITH  ONE
PJPKBHZ  UBN  PBUZMNT   XIZ  HNZ  VKZM  UBN

ANOTHER,  AND  ALL  AGAINST  EVIL  ONLY,
PBUZMNT   PBW  PFF  PJPKBHZ  NAKF  UBFD
```

Stats

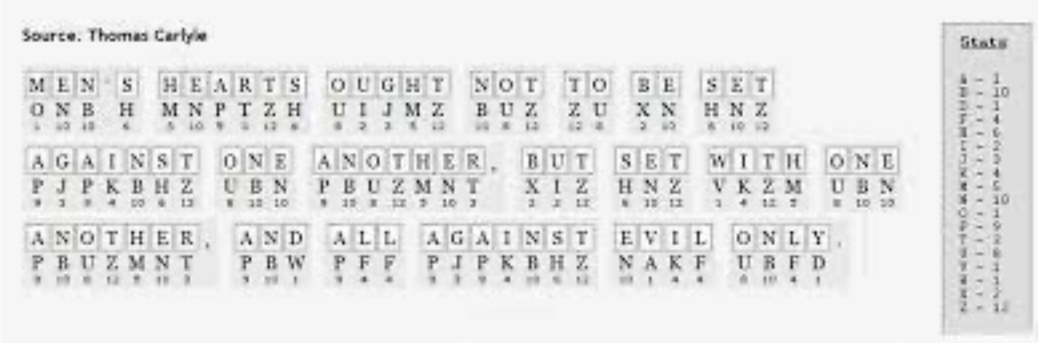
A - 1
B - 10
D - 1
F - 4
H - 6
I - 2
J - 3
K - 4
M - 5
N - 10
O - 1
P - 9
T - 3
U - 8
V - 1
W - 1
X - 2
Z - 12

Conventional Encryption Algorithm – Strategies

Week 6

Conventional Encryption Algorithm – Strategies

Substitution Cipher – Replacement of one or more things with one or more things (Symmetric versus Asymmetric)



Source: Thomas Carlyle

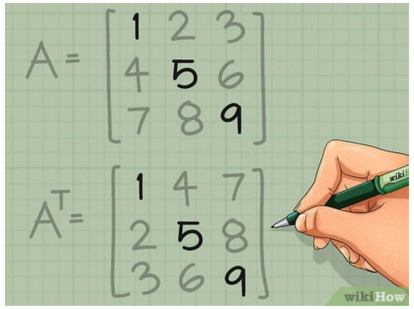
MEN'S HEARTS OUGHT NOT TO BE SET
ONB H MNPTZH UIJMZ BUZ ZU XN HNZ

AGAINST ONE ANOTHER, BUT SET WITH ONE
PJPKBHZ UBN PBUZMNT XIZ HNZ VKZM UBN

ANOTHER, AND ALL AGAINST EVIL ONLY.
PBUZMNT PBW PFF PJPKBHZ NAKF UBFD

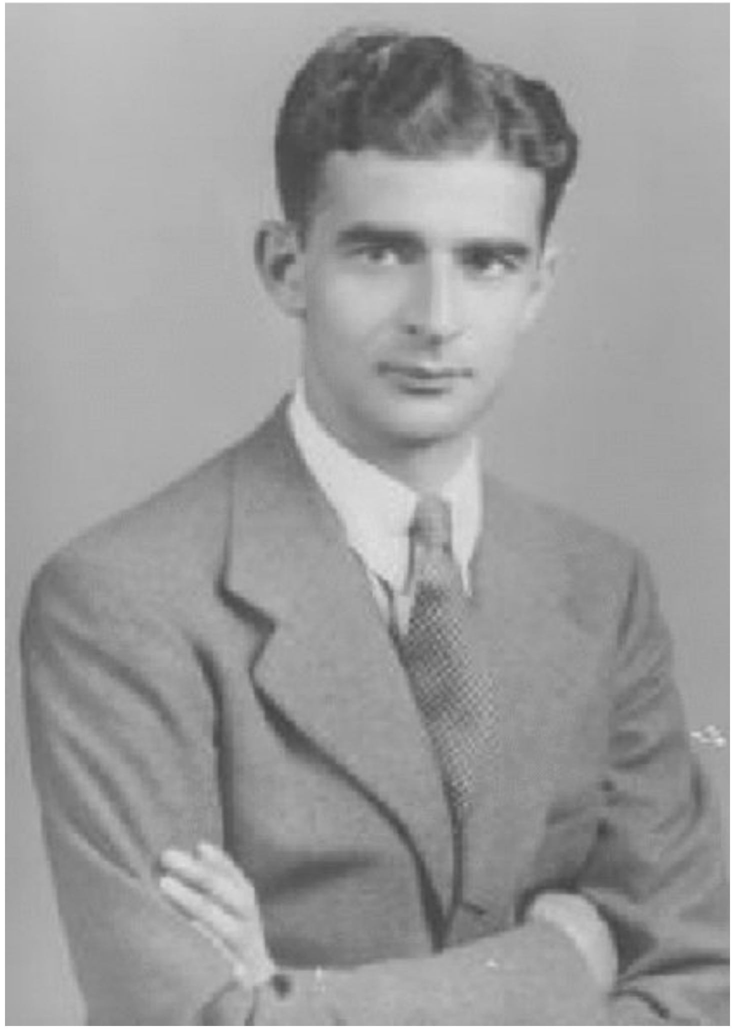
Stats

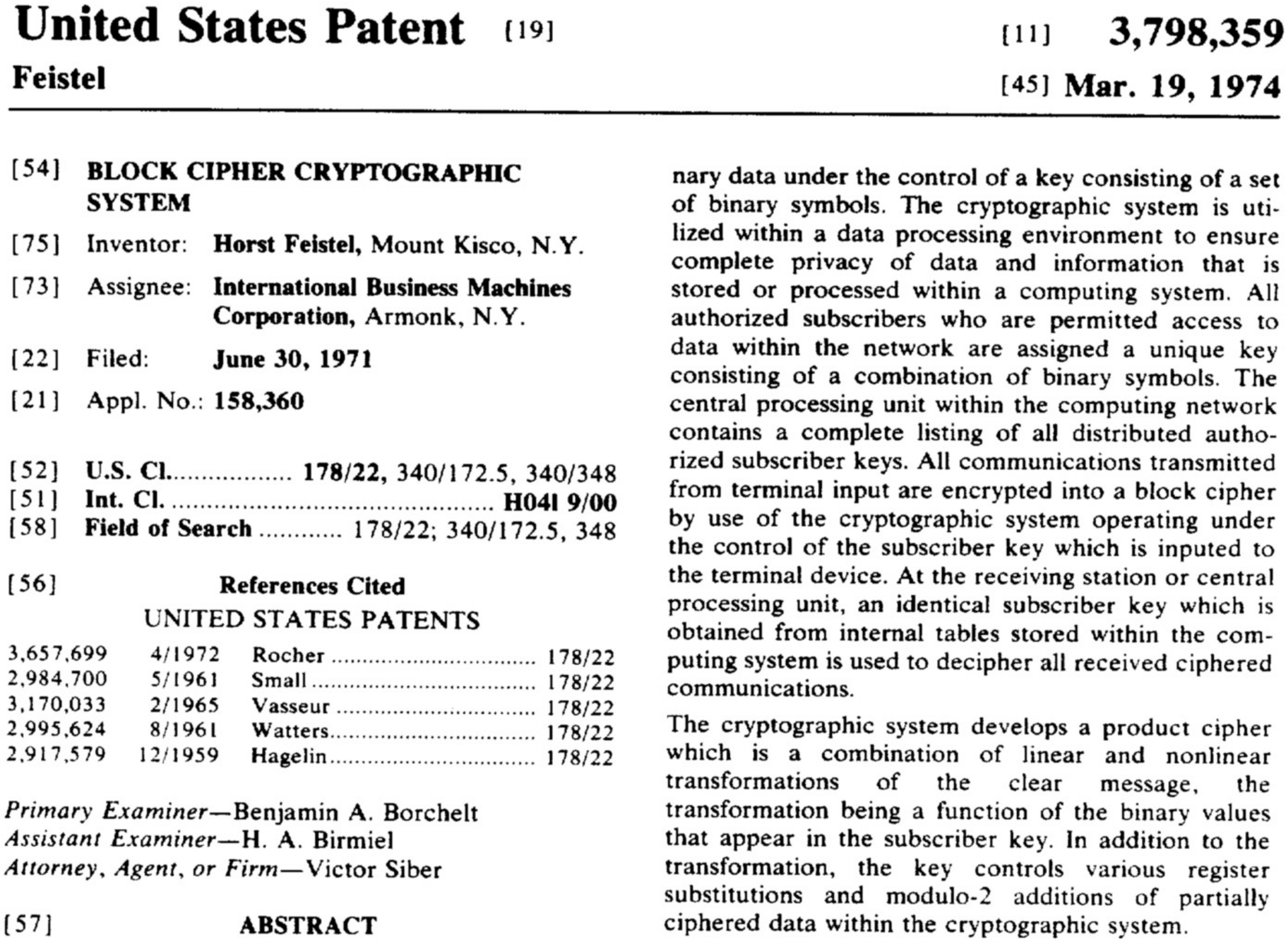
Transposition Cipher – Use of matrix arithmetic to represent and manipulate text (Linear Algebraic basis)



What is the Data Encryption Standard (DES)?
(How Did It Influence AES?)

Week 6





United States Patent [19]

Feistel

[11] **3,798,359**

[45] **Mar. 19, 1974**

[54] **BLOCK CIPHER CRYPTOGRAPHIC SYSTEM**

[75] Inventor: **Horst Feistel,** Mount Kisco, N.Y.

[73] Assignee: **International Business Machines Corporation,** Armonk, N.Y.

[22] Filed: **June 30, 1971**

[21] Appl. No.: **158,360**

[52] **U.S. Cl.**.................. **178/22,** 340/172.5, 340/348
[51] **Int. Cl.** **H04l 9/00**
[58] **Field of Search** 178/22; 340/172.5, 348

[56] **References Cited**

UNITED STATES PATENTS

3,657,699	4/1972	Rocher	178/22
2,984,700	5/1961	Small	178/22
3,170,033	2/1965	Vasseur	178/22
2,995,624	8/1961	Watters................................	178/22
2,917,579	12/1959	Hagelin................................	178/22

Primary Examiner—Benjamin A. Borchelt
Assistant Examiner—H. A. Birmiel
Attorney, Agent, or Firm—Victor Siber

[57] **ABSTRACT**

nary data under the control of a key consisting of a set of binary symbols. The cryptographic system is utilized within a data processing environment to ensure complete privacy of data and information that is stored or processed within a computing system. All authorized subscribers who are permitted access to data within the network are assigned a unique key consisting of a combination of binary symbols. The central processing unit within the computing network contains a complete listing of all distributed authorized subscriber keys. All communications transmitted from terminal input are encrypted into a block cipher by use of the cryptographic system operating under the control of the subscriber key which is inputed to the terminal device. At the receiving station or central processing unit, an identical subscriber key which is obtained from internal tables stored within the computing system is used to decipher all received ciphered communications.

The cryptographic system develops a product cipher which is a combination of linear and nonlinear transformations of the clear message, the transformation being a function of the binary values that appear in the subscriber key. In addition to the transformation, the key controls various register substitutions and modulo-2 additions of partially ciphered data within the cryptographic system.

Data Encryption Standard (DES)

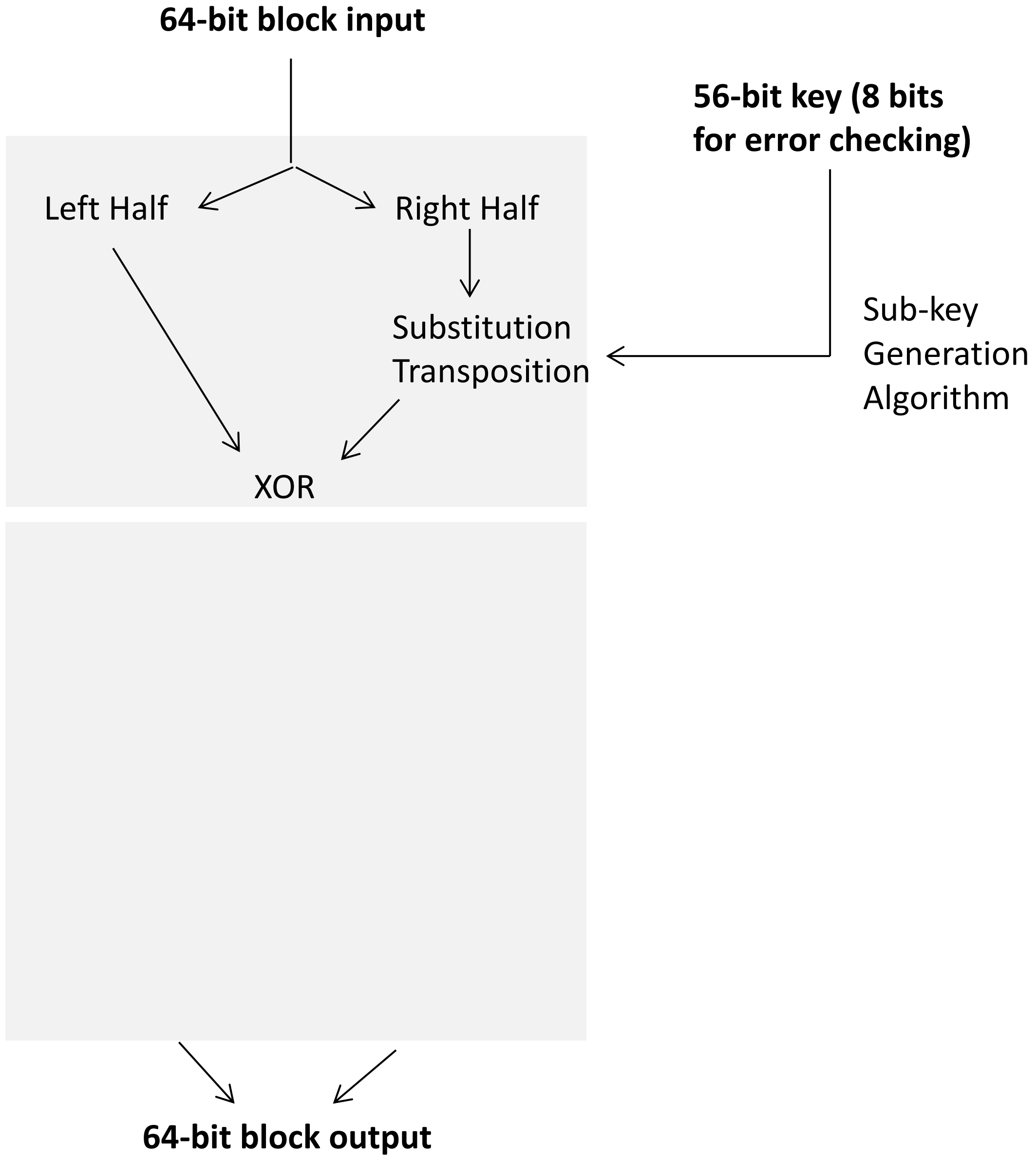
64-bit block input

**56-bit key (8 bits
for error checking)**

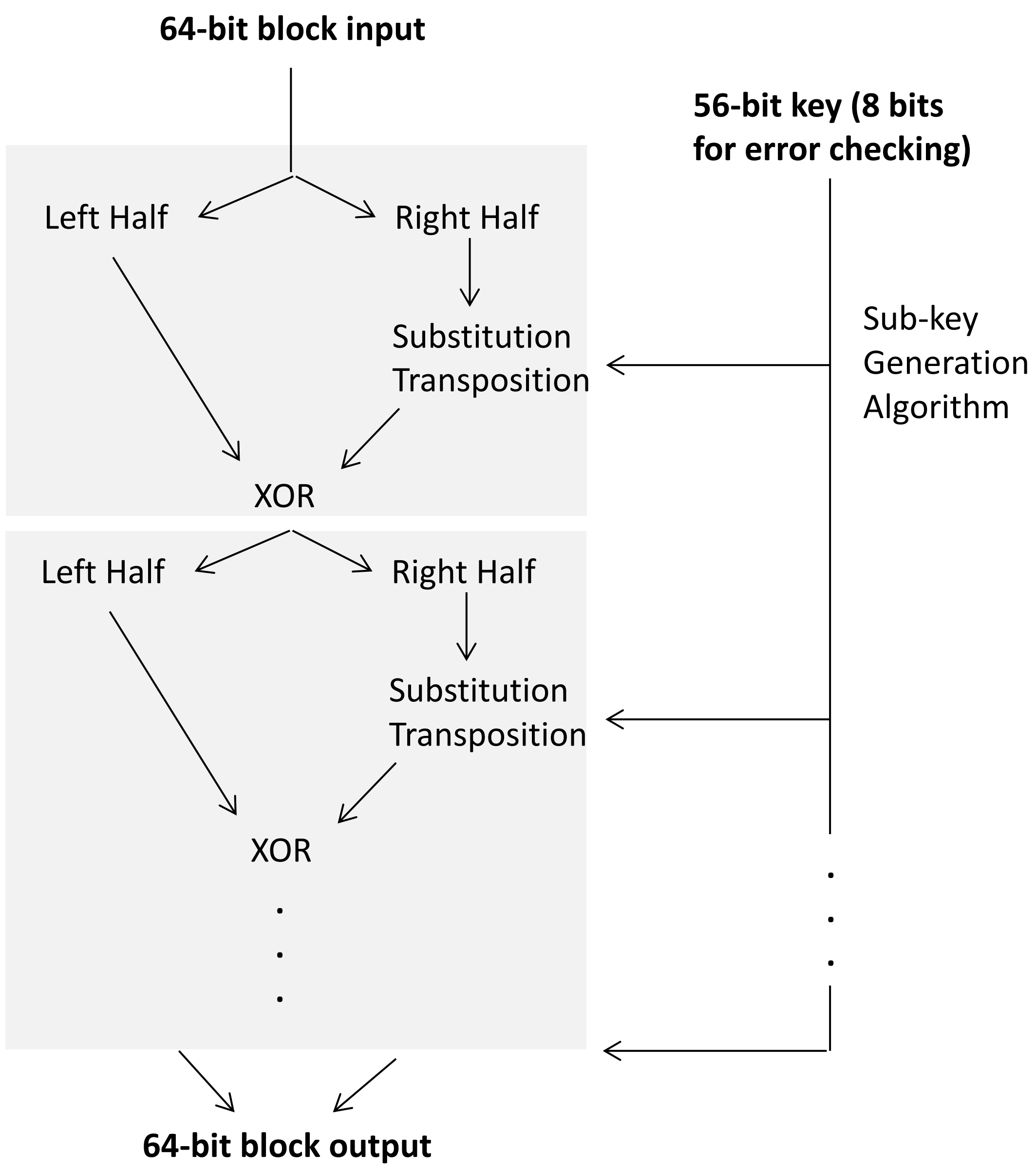
Original National Bureau of Standards (NBS) in Washington

64-bit block output

Data Encryption Standard (DES)

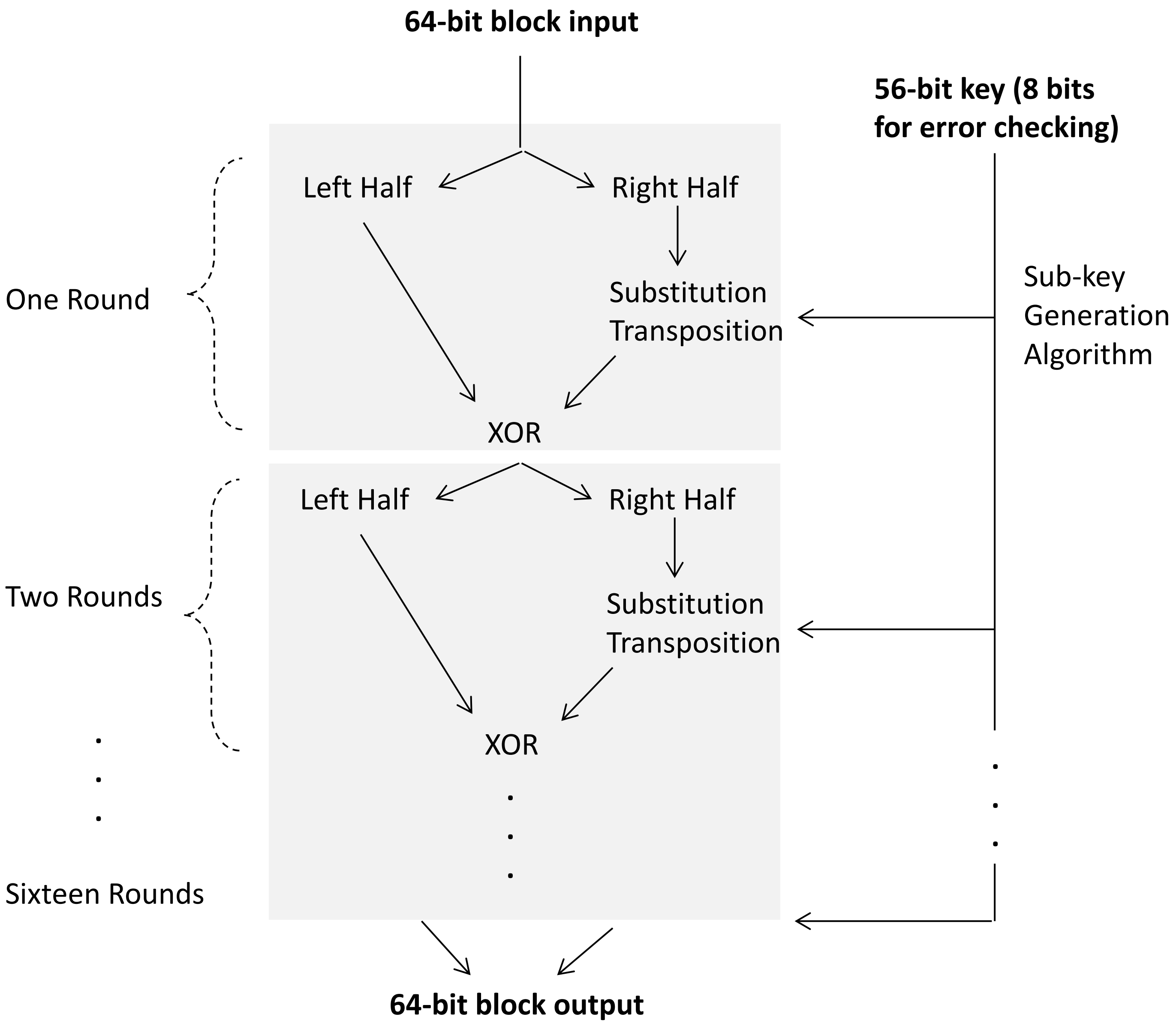

64-bit block input
56-bit key (8 bits for error checking)
Left Half
Right Half
Substitution Transposition
Sub-key Generation Algorithm
XOR
64-bit block output

Data Encryption Standard (DES)

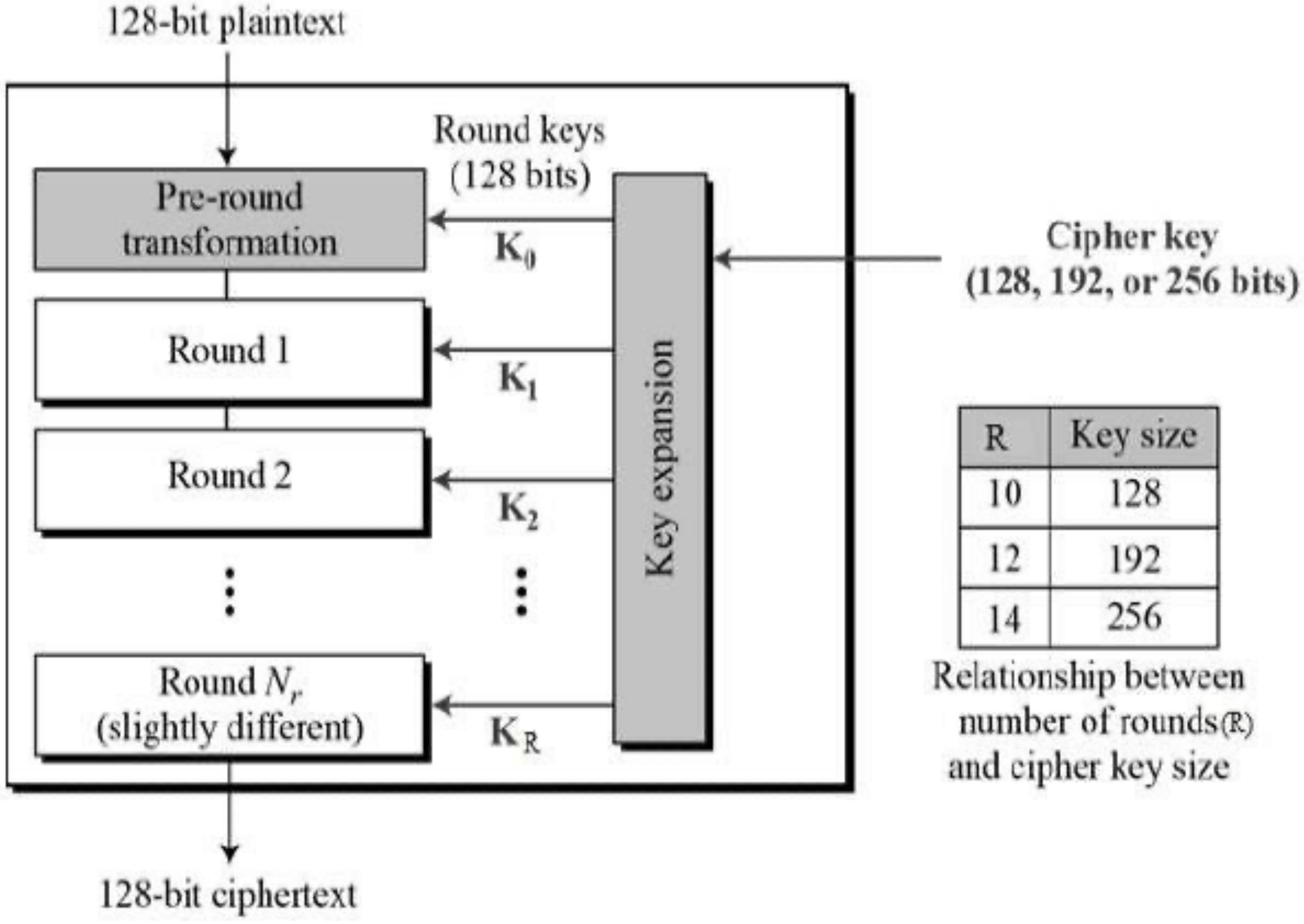

64-bit block input
56-bit key (8 bits for error checking)
Left Half
Right Half
Substitution Transposition
Sub-key Generation Algorithm
XOR
Left Half
Right Half
Substitution Transposition
XOR
. . .
. . .
64-bit block output

Week 6

Data Encryption Standard (DES)



Advanced Encryption Standard (AES)



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

What is Triple DES (3DES)?
(How Did It Solve Key Length Issues and 1DES Interoperability?)

Walt Tuchman – IBM
Circa 1981

Triple-DES

$\{ m \}_{K1}$	Single-DES	56 Bit Key

Triple-DES

$\{ m \}_{K1}$	Single-DES	56 Bit Key
$\{ \{ m \}_{K1} \}_{K2}$	Double-DES	112 Bit Key

Triple-DES

$\{ m \}_{K1}$	Single-DES	56 Bit Key
$\{ \{ m \}_{K1} \}_{K2}$	Double-DES	112 Bit Key
$\{ \{ \{ m \}_{K1} \}_{K2} \}_{K3}$	Triple-DES	168 Bit Key

Triple-DES

$\{ m \}_{K1}$	Single-DES	56 Bit Key
$\{ \{ m \}_{K1} \}_{K2}$	Double-DES	112 Bit Key
$\{ \{ \{ m \}_{K1} \}_{K2} \}_{K3}$	Triple-DES	168 Bit Key

Single-DES Mode: K1 = K2 ≠ K3

$$\{ \{ \{ m \overset{\overset{E}{\downarrow}}{\}}_{K1} \overset{\overset{D}{\downarrow}}{\}}_{K1} \overset{\overset{E}{\downarrow}}{\}}_{K3}$$

$$\{ \quad m \quad \}_{K3}$$

Triple-DES

$\{ m \}_{K1}$	Single-DES	56 Bit Key
$\{ \{ m \}_{K1} \}_{K2}$	Double-DES	112 Bit Key
$\{ \{ \{ m \}_{K1} \}_{K2} \}_{K3}$	Triple-DES	168 Bit Key

Single-DES Mode: $K1 = K2 \neq K3$

E D E

$$\{ \{ \{ m \overset{\downarrow}{\}}_{K1} \overset{\downarrow}{\}}_{K1} \overset{\downarrow}{\}}_{K3}$$

$$\{ \quad m \quad \}_{K3}$$

Triple-DES Mode: $K1 = K3 \neq K2$

Effective 112 bits

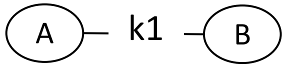
$$\underset{\uparrow}{\{}\, \underset{\uparrow}{\{}\, \underset{\uparrow}{\{} m \}_{K1} \}_{K2} \}_{K1}$$

E D E Mode

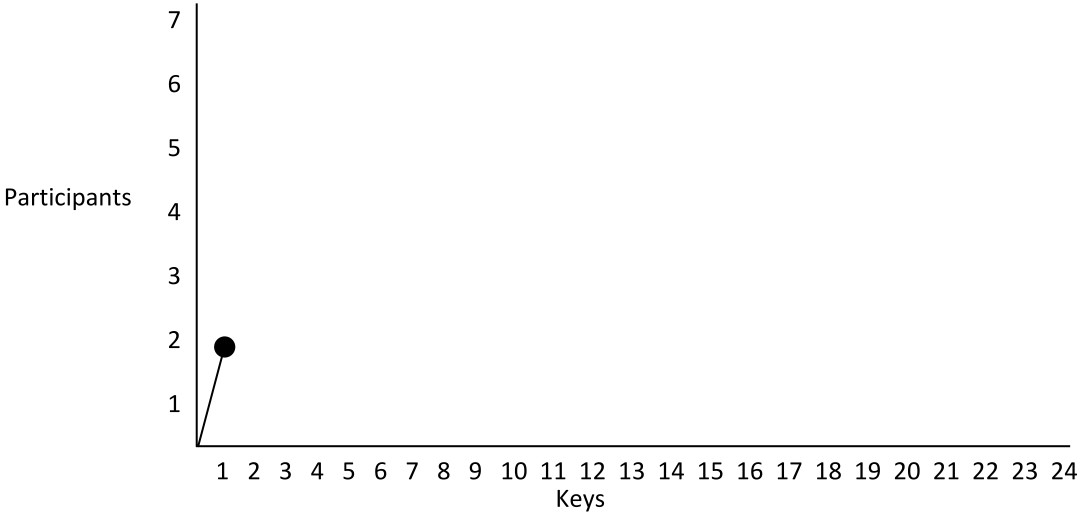
What is the Scaling Issue for Conventional Cryptography?

Week 6

Conventional Encryption Scaling Issue

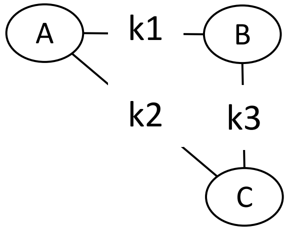


2 participants – 1 shared key



Week 6

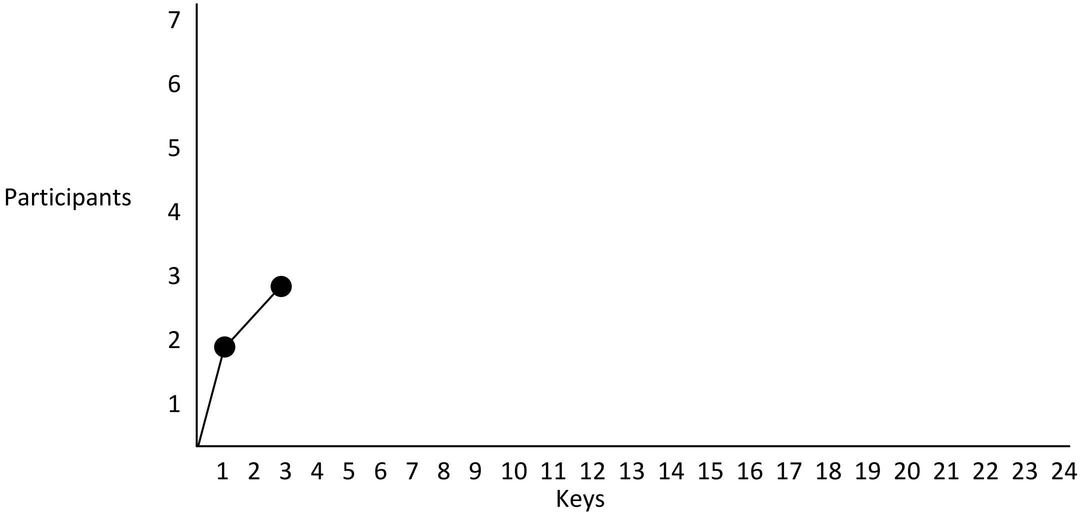
Conventional Encryption Scaling Issue



2 participants – 1 shared key

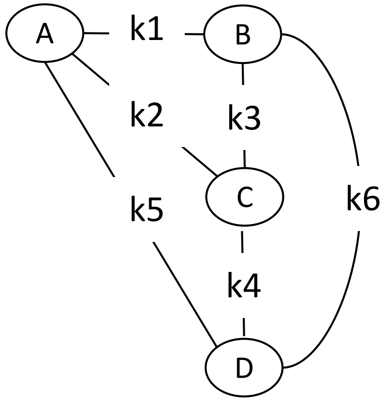
3 participants – 3 shared keys

Added participant 1
Added new keys 2



Week 6

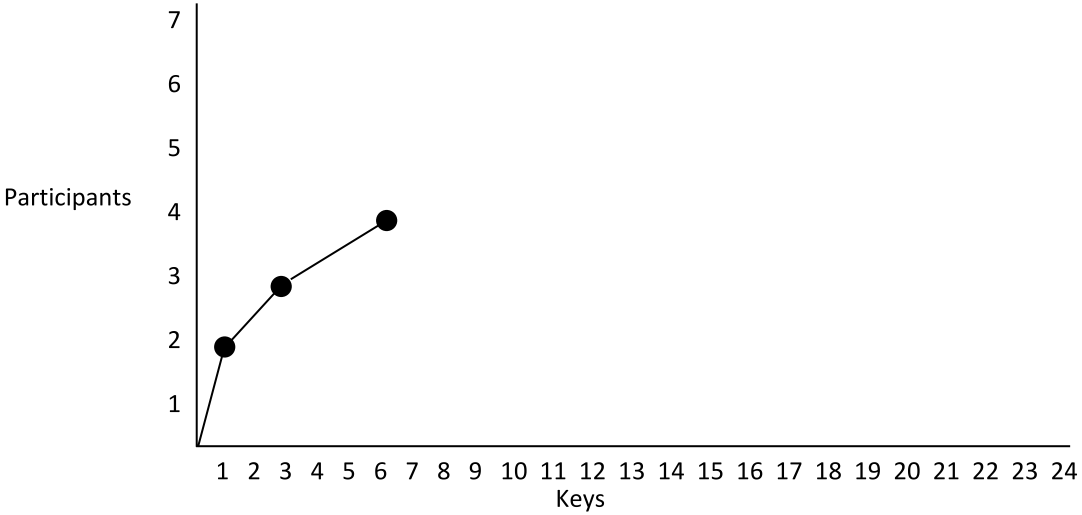
Conventional Encryption Scaling Issue



2 participants – 1 shared key

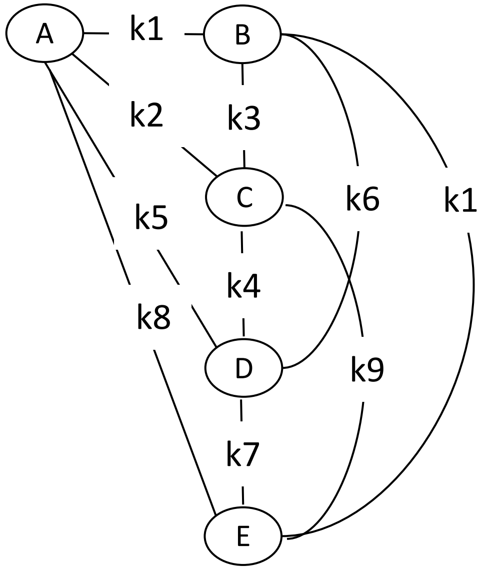
3 participants – 3 shared keys

4 participants – 6 shared keys



Added participant	1
Added new keys	3

Conventional Encryption Scaling Issue

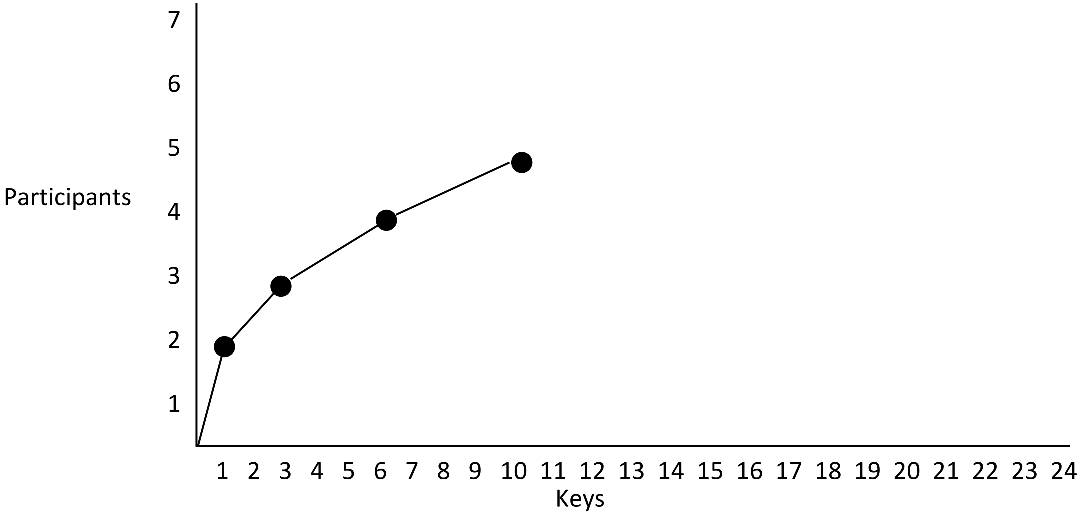


2 participants – 1 shared key

3 participants – 3 shared keys

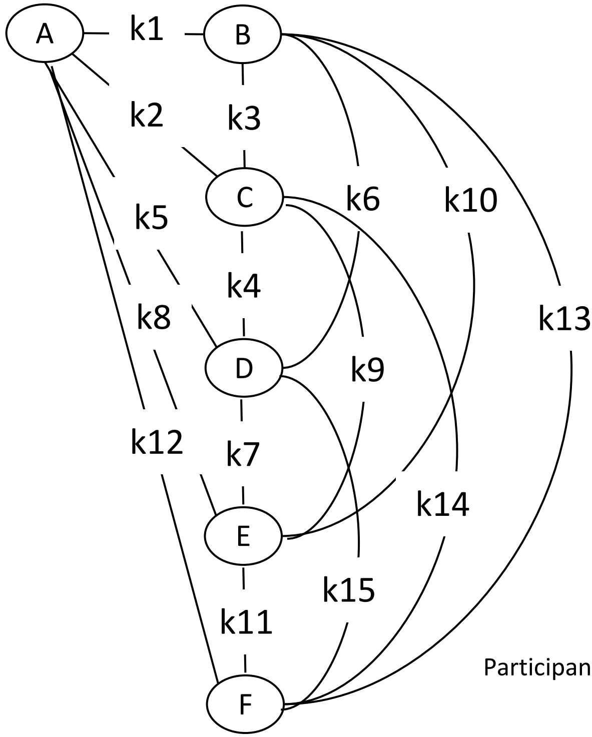
4 participants – 6 shared keys

5 participants – 10 shared keys



Added participant	1
Added new keys	4

Conventional Encryption Scaling Issue



2 participants – 1 shared key

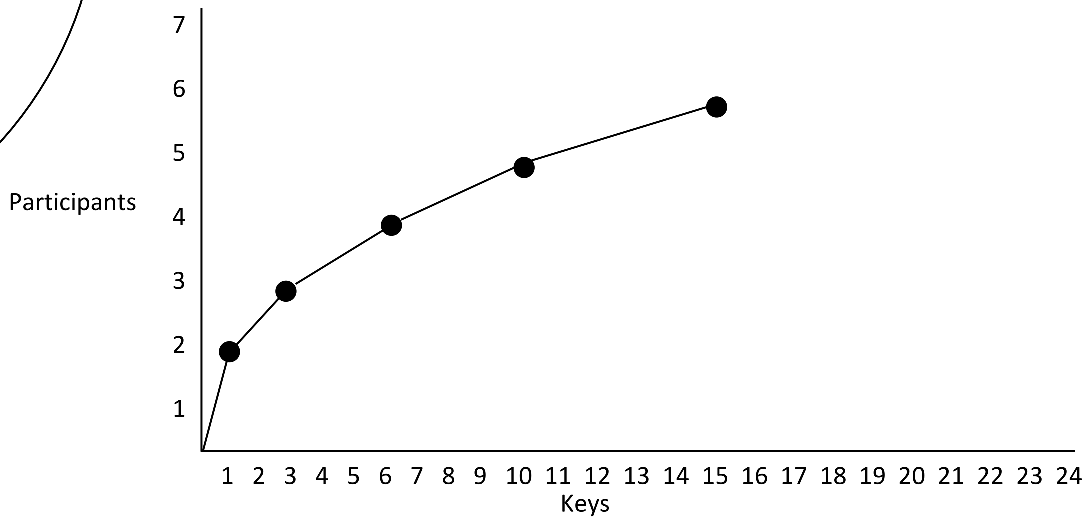
3 participants – 3 shared keys

4 participants – 6 shared keys

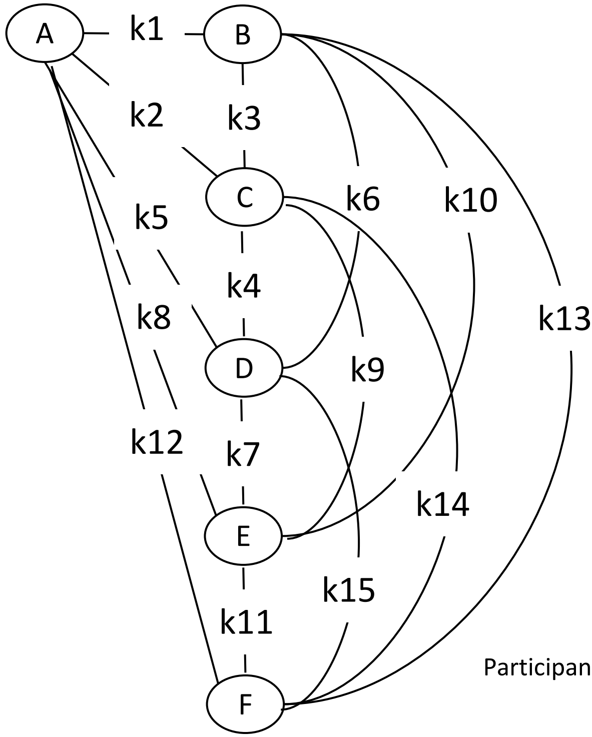
5 participants – 10 shared keys

6 participants – 15 shared keys

Added participant 1
Added new keys 5



Conventional Encryption Scaling Issue



2 participants – 1 shared key

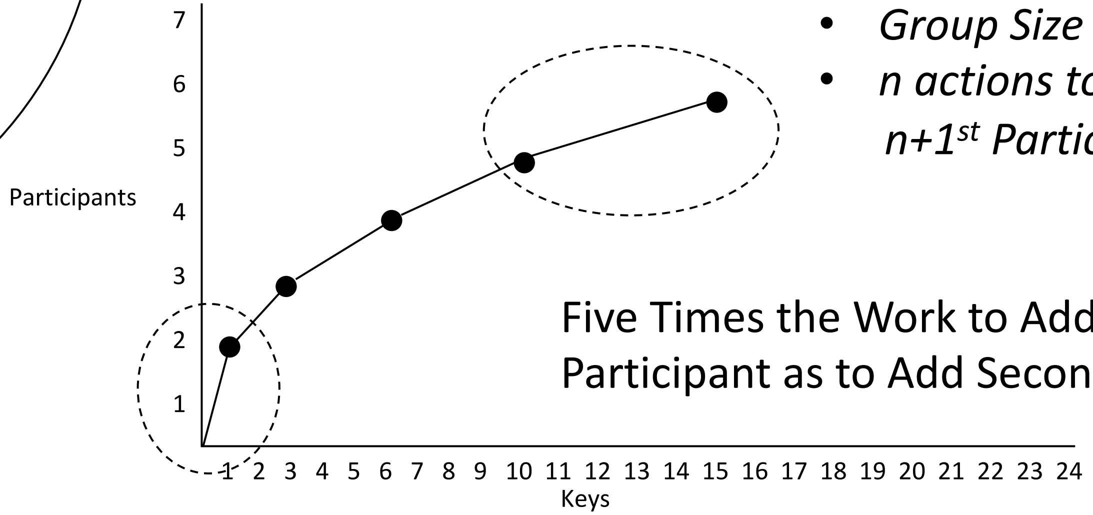
3 participants – 3 shared keys

4 participants – 6 shared keys

5 participants – 10 shared keys

6 participants – 15 shared keys

Added participant 1
Added new keys 5

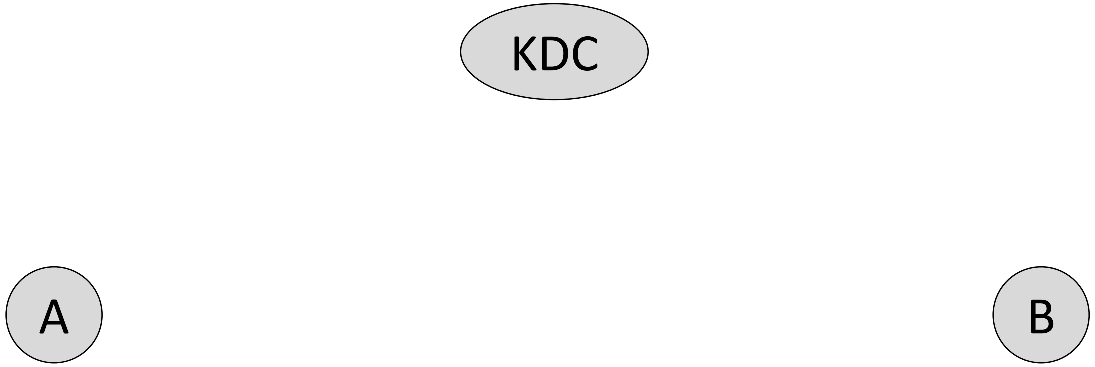


- *Group Size = n*
- *n actions to add n+1st Participant*

Five Times the Work to Add Sixth Participant as to Add Second

What are the Key Security Properties of Conventional Cryptography?

Conventional Cryptography

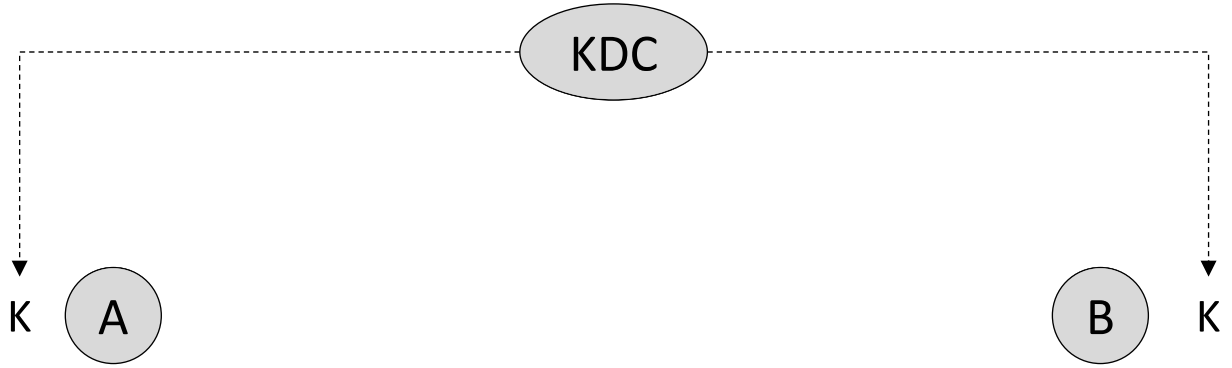


KDC

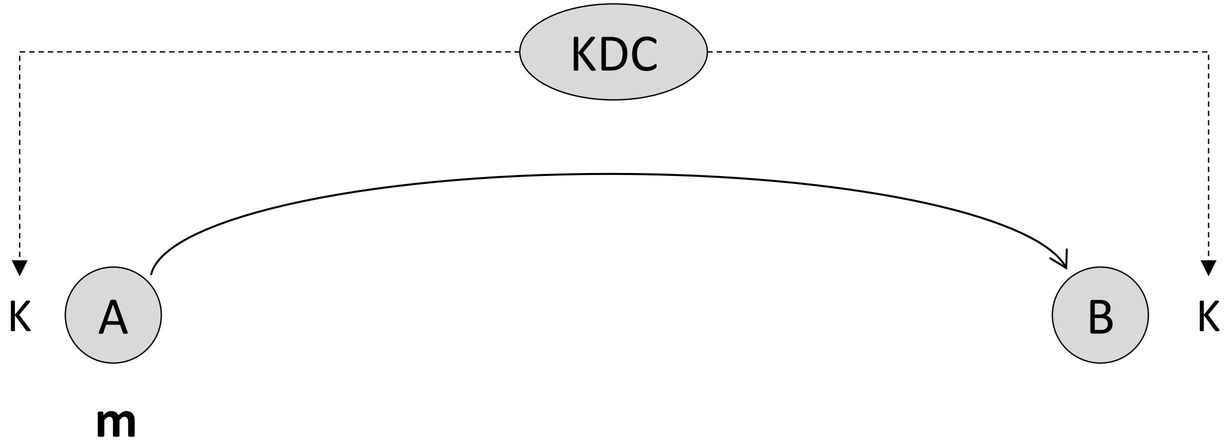
A

B

Conventional Cryptography

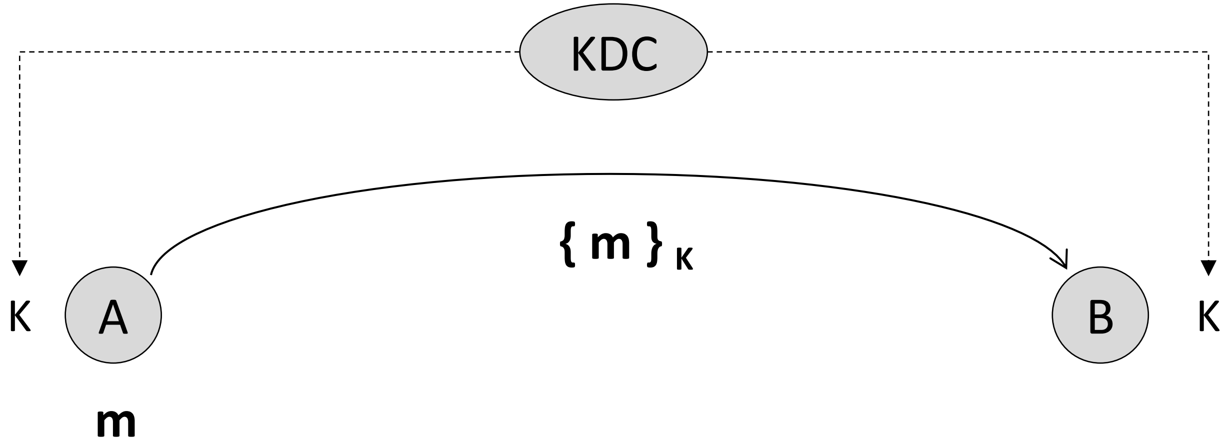


Conventional Cryptography



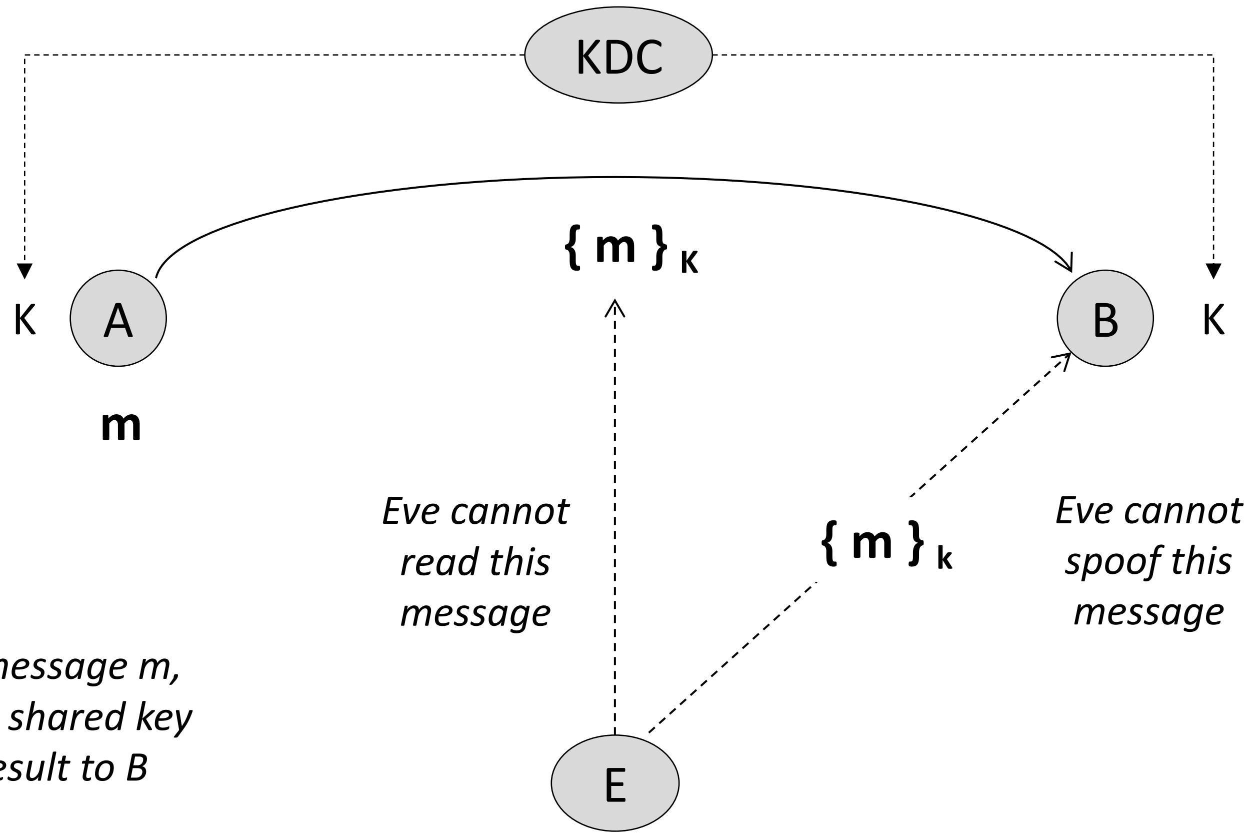
Alice creates message m . . .

Conventional Cryptography



Alice creates message m, encrypts using shared key k, and sends result to B

Conventional Cryptography



KDC

K

A

B

K

$\{ m \}_K$

m

Eve cannot read this message

$\{ m \}_k$

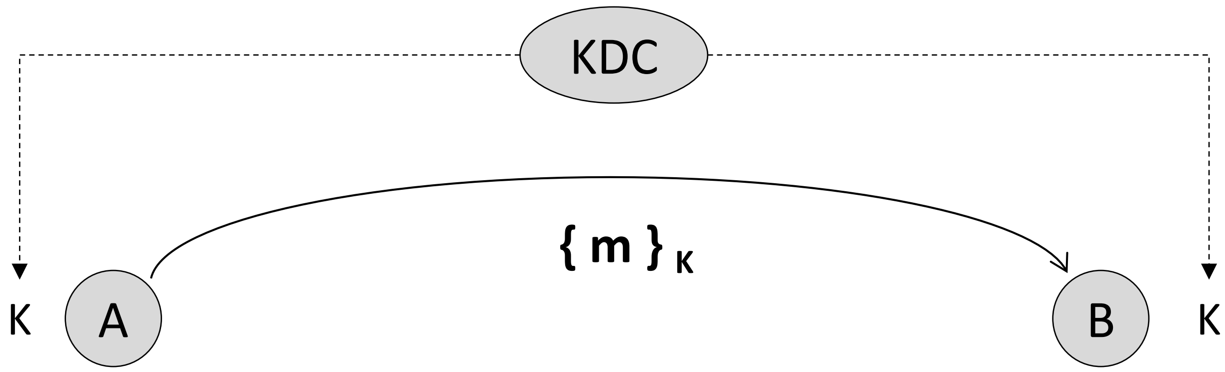
Eve cannot spoof this message

E

Does not have K

Alice creates message m, encrypts using shared key k, and sends result to B

Conventional Cryptography



KDC

$\{ m \}_K$

K A

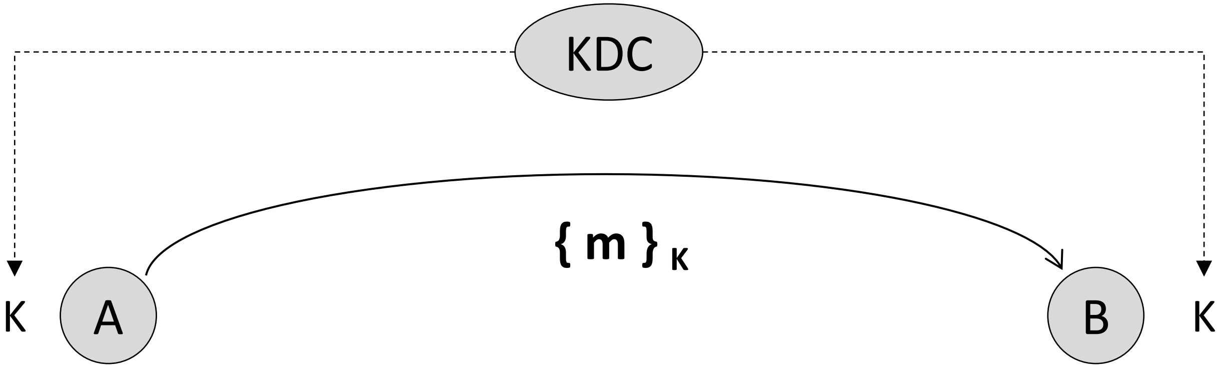
B K

m

$\{ \{ m \}_K \}_k = m$

Bob receives encrypted message, and decrypts using shared key k, and obtains message m

Conventional Cryptography



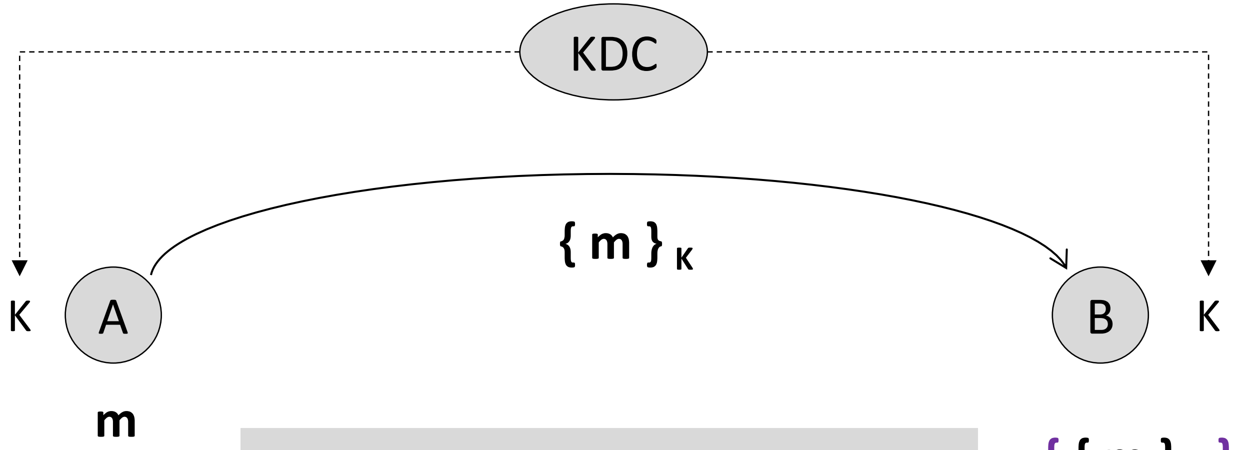
m

Secrecy Between A and B? **YES**
Authentication of A by B? **YES**

$\{ \{ m \}_K \}_k = m$

Week 6

Conventional Cryptography



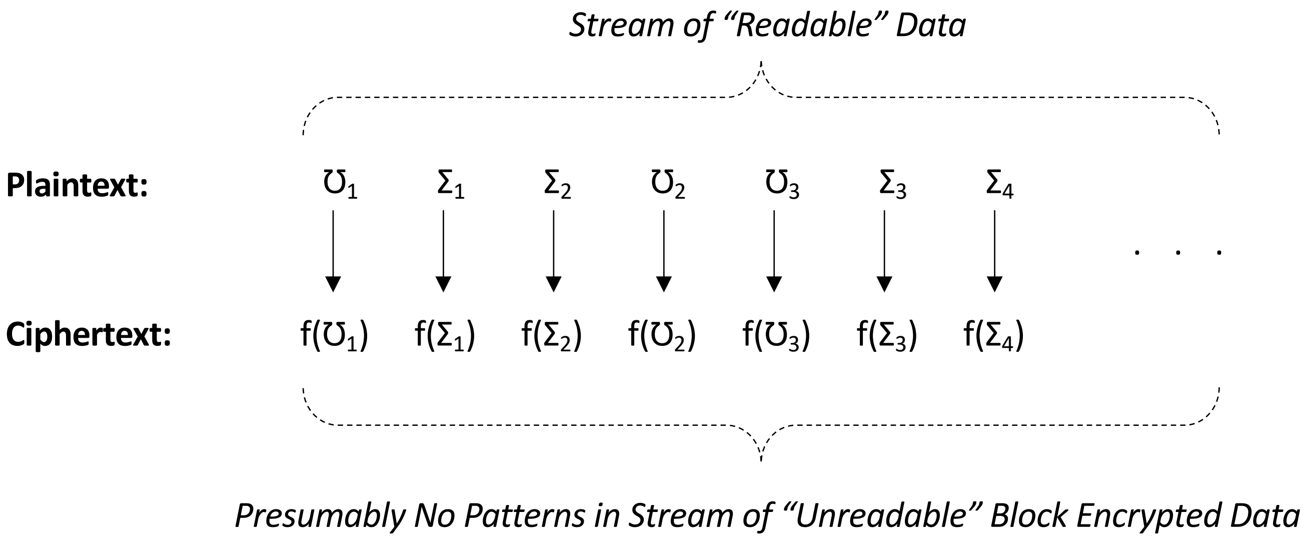
$\{\{ m \}_K\}_k = m$

Secrecy Between A and B? **YES**
Authentication of A by B? **YES**

Does this approach scale? **NO**

How Does Block Chaining Work?

Conventional Block Cryptography



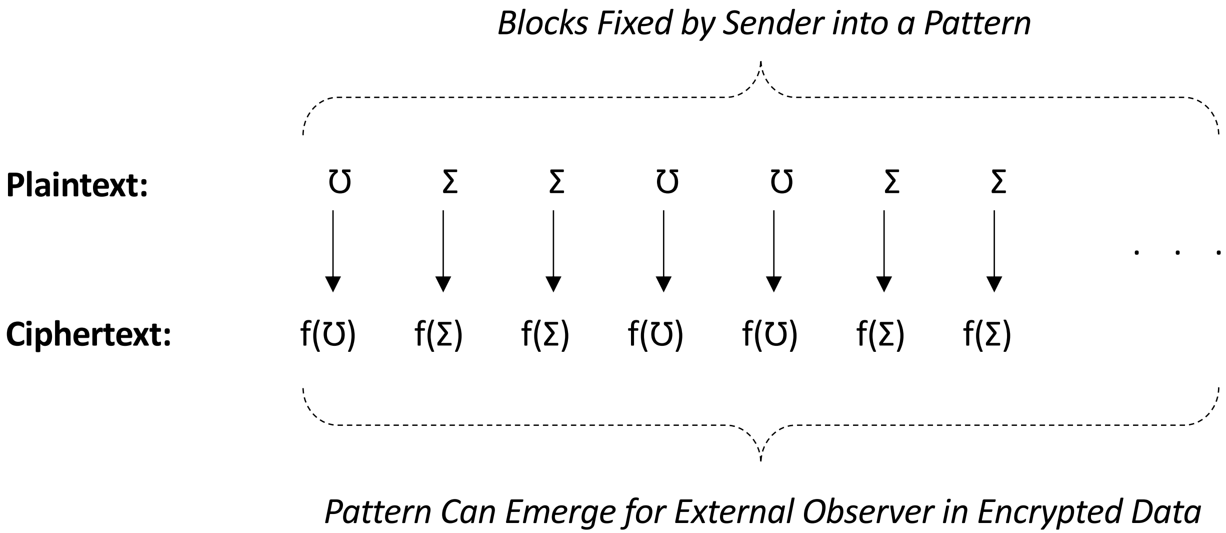
Stream of "Readable" Data

Plaintext: $\mho_1$ Σ_1 Σ_2 $\mho_2$ $\mho_3$ Σ_3 Σ_4 . . .

Ciphertext: $f(\mho_1)$ $f(\Sigma_1)$ $f(\Sigma_2)$ $f(\mho_2)$ $f(\mho_3)$ $f(\Sigma_3)$ $f(\Sigma_4)$

Presumably No Patterns in Stream of "Unreadable" Block Encrypted Data

Conventional Block Cryptography – Covert Channel



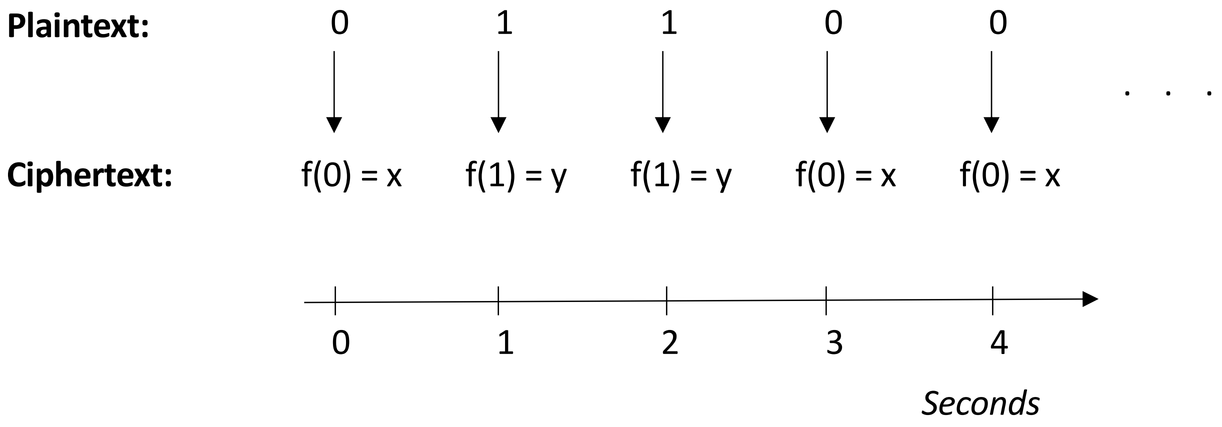
Blocks Fixed by Sender into a Pattern

Plaintext: Ʊ Σ Σ Ʊ Ʊ Σ Σ . . .

Ciphertext: f(Ʊ) f(Σ) f(Σ) f(Ʊ) f(Ʊ) f(Σ) f(Σ)

Pattern Can Emerge for External Observer in Encrypted Data

Conventional Block Cryptography – 1 bps Channel



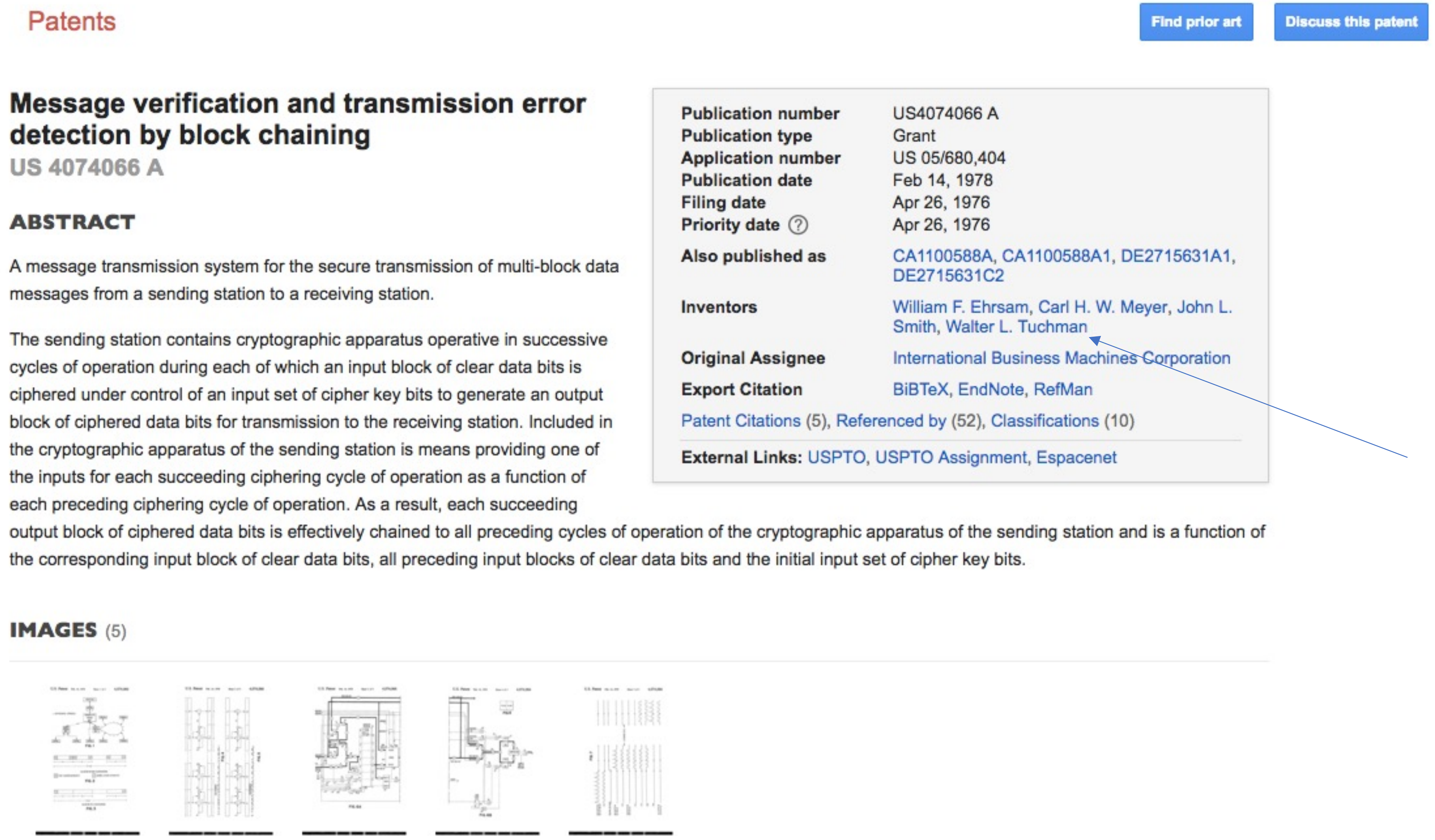
Plaintext: 0 1 1 0 0 . . .

Ciphertext: f(0) = x f(1) = y f(1) = y f(0) = x f(0) = x

0 1 2 3 4

Seconds

Block Chain Mode Cryptography – Circa 1976 at IBM



Message verification and transmission error detection by block chaining

US 4074066 A

Publication number	US4074066 A
Publication type	Grant
Application number	US 05/680,404
Publication date	Feb 14, 1978
Filing date	Apr 26, 1976
Priority date ?	Apr 26, 1976
Also published as	CA1100588A, CA1100588A1, DE2715631A1, DE2715631C2
Inventors	William F. Ehrsam, Carl H. W. Meyer, John L. Smith, Walter L. Tuchman
Original Assignee	International Business Machines Corporation
Export Citation	BiBTeX, EndNote, RefMan

Patent Citations (5), Referenced by (52), Classifications (10)

External Links: USPTO, USPTO Assignment, Espacenet

ABSTRACT

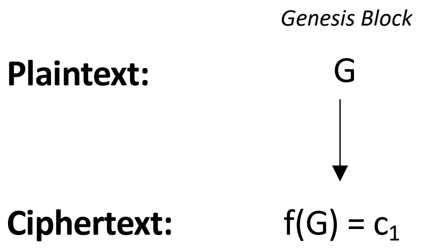
A message transmission system for the secure transmission of multi-block data messages from a sending station to a receiving station.

The sending station contains cryptographic apparatus operative in successive cycles of operation during each of which an input block of clear data bits is ciphered under control of an input set of cipher key bits to generate an output block of ciphered data bits for transmission to the receiving station. Included in the cryptographic apparatus of the sending station is means providing one of the inputs for each succeeding ciphering cycle of operation as a function of each preceding ciphering cycle of operation. As a result, each succeeding output block of ciphered data bits is effectively chained to all preceding cycles of operation of the cryptographic apparatus of the sending station and is a function of the corresponding input block of clear data bits, all preceding input blocks of clear data bits and the initial input set of cipher key bits.

IMAGES (5)

Week 6

Block Chain Mode Cryptography

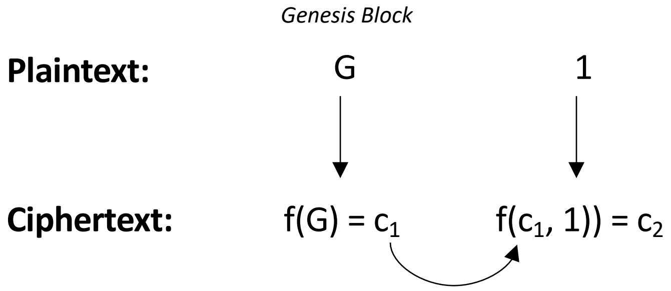


Genesis Block

Plaintext: G

Ciphertext: $f(G) = c_1$

Block Chain Mode Cryptography

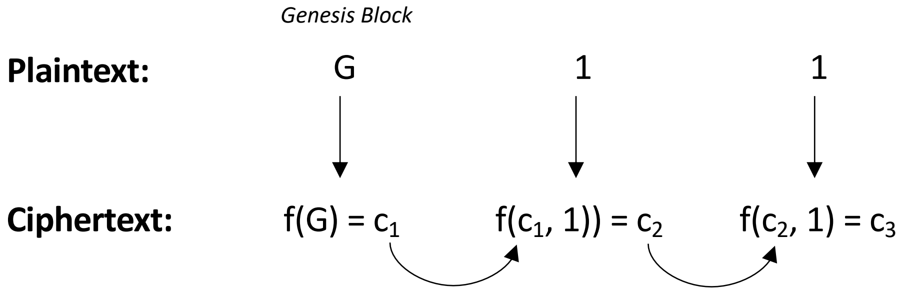


Genesis Block

Plaintext: G 1

Ciphertext: $f(G) = c_1$ $f(c_1, 1)) = c_2$

Block Chain Mode Cryptography

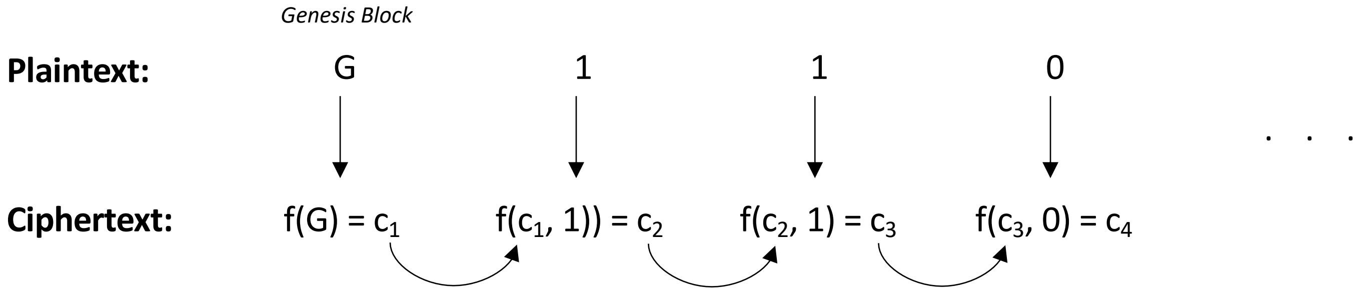


Genesis Block

Plaintext: G 1 1

Ciphertext: $f(G) = c_1$ $f(c_1, 1)) = c_2$ $f(c_2, 1) = c_3$

Block Chain Mode Cryptography

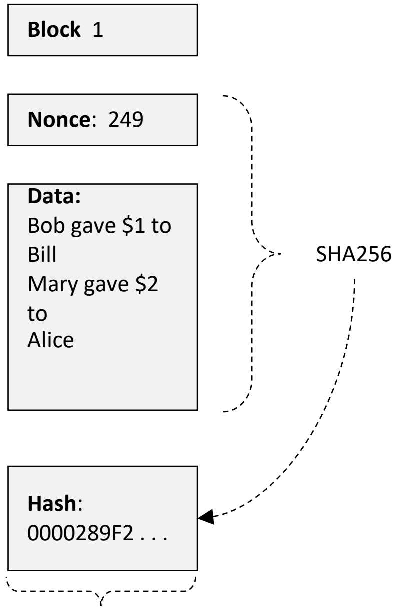


Genesis Block

Plaintext: G 1 1 0 . . .

Ciphertext: $f(G) = c_1$ $f(c_1, 1)) = c_2$ $f(c_2, 1) = c_3$ $f(c_3, 0) = c_4$

Modern Block Chain Usage



Block 1

Nonce: 249

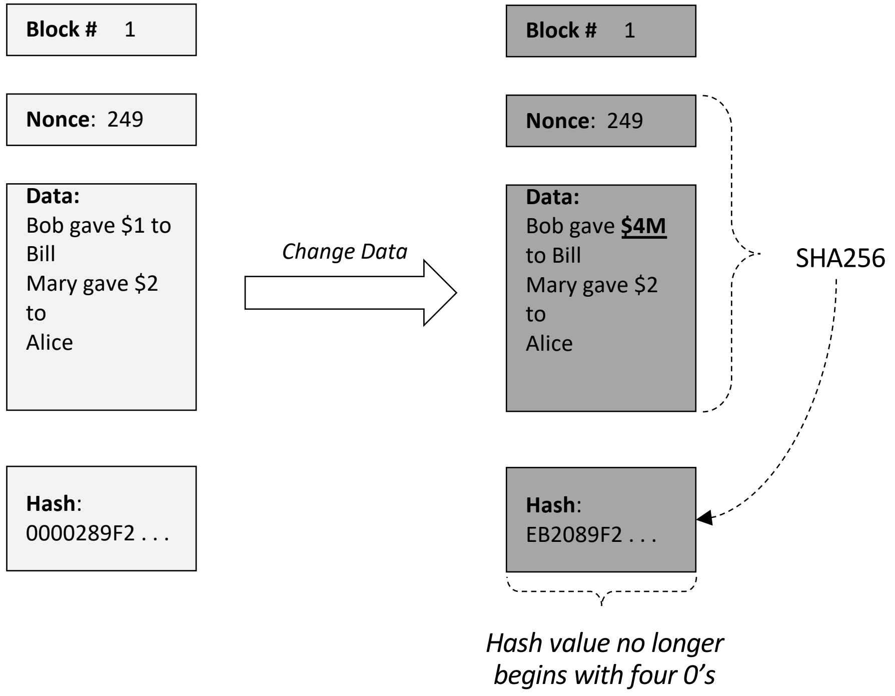
Data:
Bob gave $1 to
Bill
Mary gave $2
to
Alice

SHA256

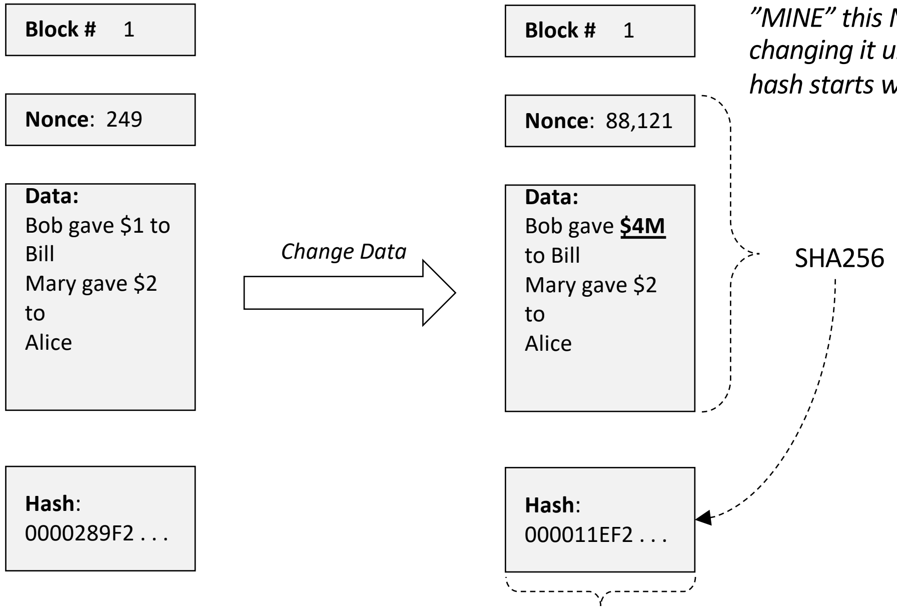
Hash:
0000289F2 . . .

Hash value happens to begin with four 0's

Modern Block Chain Usage



Modern Block Chain Usage



Block # 1

Nonce: 249

Data:
Bob gave $1 to Bill
Mary gave $2 to Alice

Hash:
0000289F2 . . .

Change Data

Block # 1

Nonce: 88,121

Data:
Bob gave **$4M** to Bill
Mary gave $2 to Alice

Hash:
000011EF2 . . .

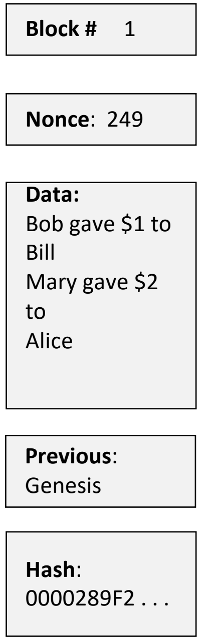
SHA256

”MINE” this Nonce repeatedly, changing it until the resulting hash starts with four 0’s

Hash value now begins with four 0’s

Week 6

Modern Block Chain Usage



Block # 1

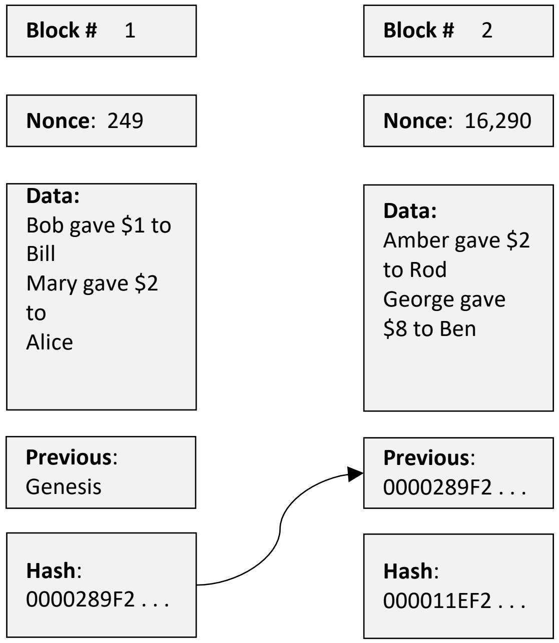
Nonce: 249

Data:
Bob gave $1 to
Bill
Mary gave $2
to
Alice

Previous:
Genesis

Hash:
0000289F2 . . .

Modern Block Chain Usage



Block # 1

Nonce: 249

Data:
Bob gave $1 to Bill
Mary gave $2 to
Alice

Previous:
Genesis

Hash:
0000289F2 . . .

Block # 2

Nonce: 16,290

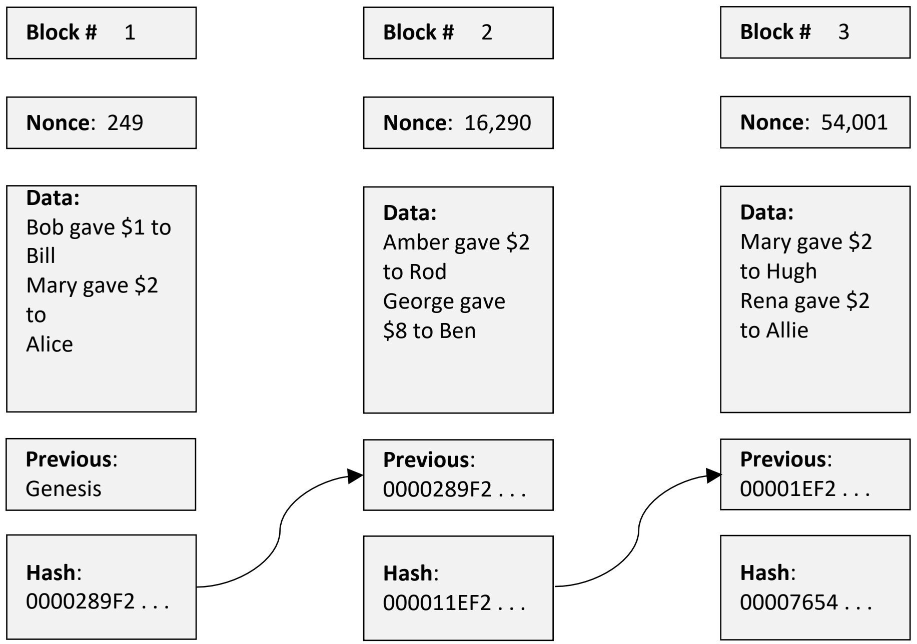
Data:
Amber gave $2 to Rod
George gave $8 to Ben

Previous:
0000289F2 . . .

Hash:
000011EF2 . . .

Week 6

Modern Block Chain Usage



Block # 1

Nonce: 249

Data:
Bob gave $1 to Bill
Mary gave $2 to Alice

Previous: Genesis

Hash: 0000289F2 . . .

Block # 2

Nonce: 16,290

Data:
Amber gave $2 to Rod
George gave $8 to Ben

Previous: 0000289F2 . . .

Hash: 000011EF2 . . .

Block # 3

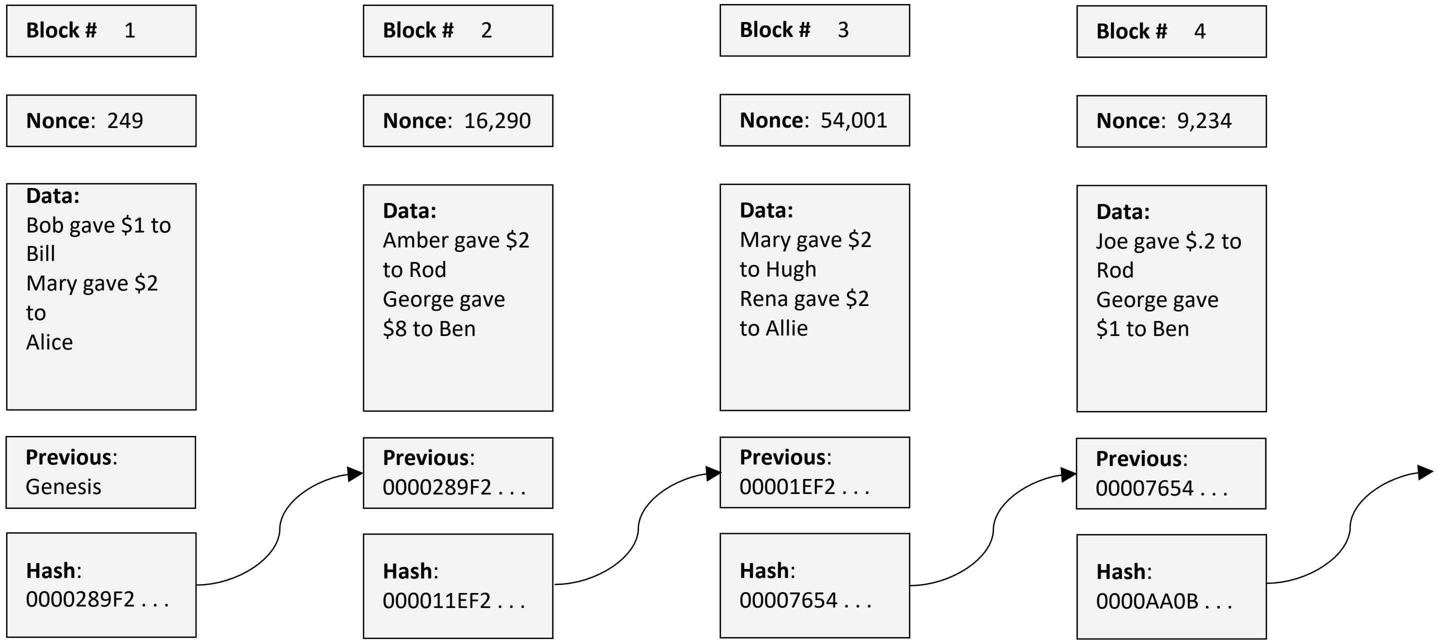
Nonce: 54,001

Data:
Mary gave $2 to Hugh
Rena gave $2 to Allie

Previous: 00001EF2 . . .

Hash: 00007654 . . .

Modern Block Chain Usage



Block # 1

Nonce: 249

Data:
Bob gave $1 to
Bill
Mary gave $2
to
Alice

Previous:
Genesis

Hash:
0000289F2 . . .

Block # 2

Nonce: 16,290

Data:
Amber gave $2
to Rod
George gave
$8 to Ben

Previous:
0000289F2 . . .

Hash:
000011EF2 . . .

Block # 3

Nonce: 54,001

Data:
Mary gave $2
to Hugh
Rena gave $2
to Allie

Previous:
00001EF2 . . .

Hash:
00007654 . . .

Block # 4

Nonce: 9,234

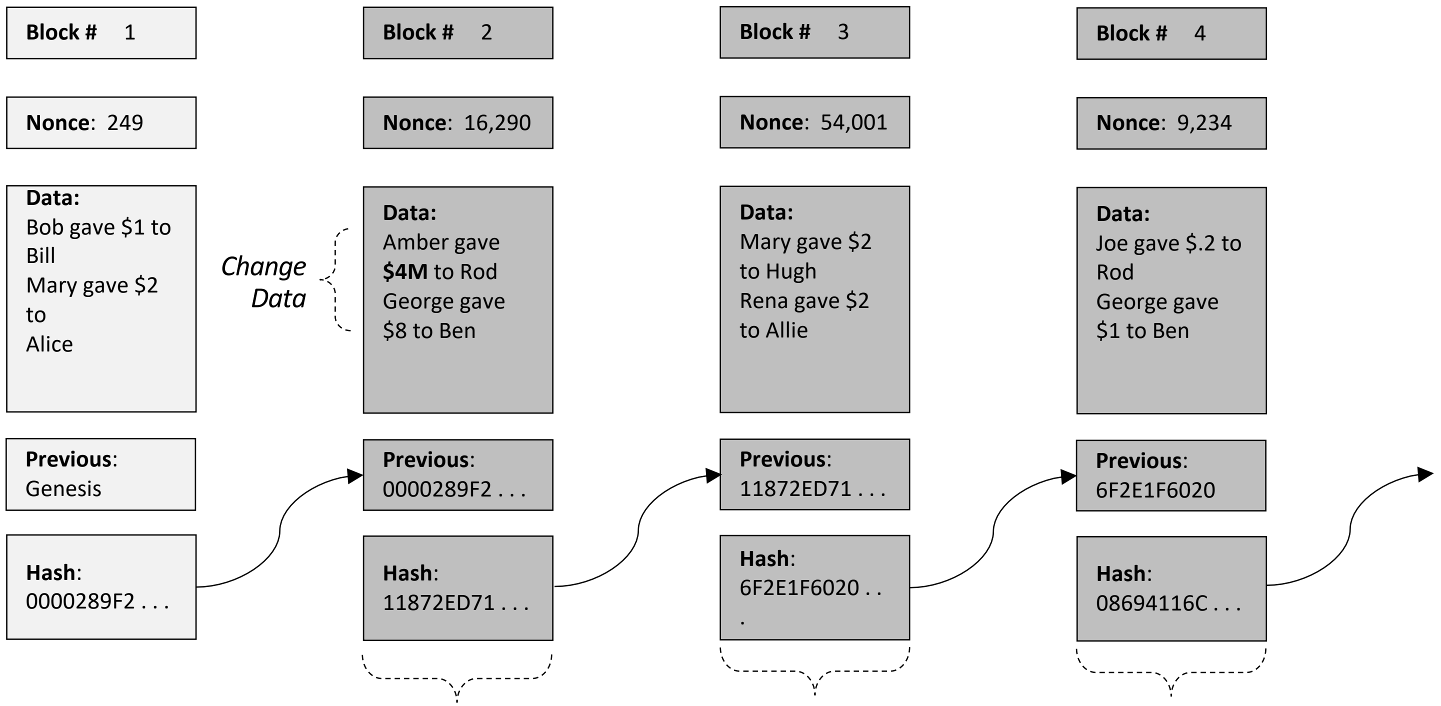
Data:
Joe gave $.2 to
Rod
George gave
$1 to Ben

Previous:
00007654 . . .

Hash:
0000AA0B . . .

Week 6

Modern Block Chain Usage



Block # 1

Nonce: 249

Data:
Bob gave $1 to Bill
Mary gave $2 to Alice

Previous:
Genesis

Hash:
0000289F2 . . .

Change Data

Block # 2

Nonce: 16,290

Data:
Amber gave **$4M** to Rod
George gave $8 to Ben

Previous:
0000289F2 . . .

Hash:
11872ED71 . . .

Block # 3

Nonce: 54,001

Data:
Mary gave $2 to Hugh
Rena gave $2 to Allie

Previous:
11872ED71 . . .

Hash:
6F2E1F6020 . . .

Block # 4

Nonce: 9,234

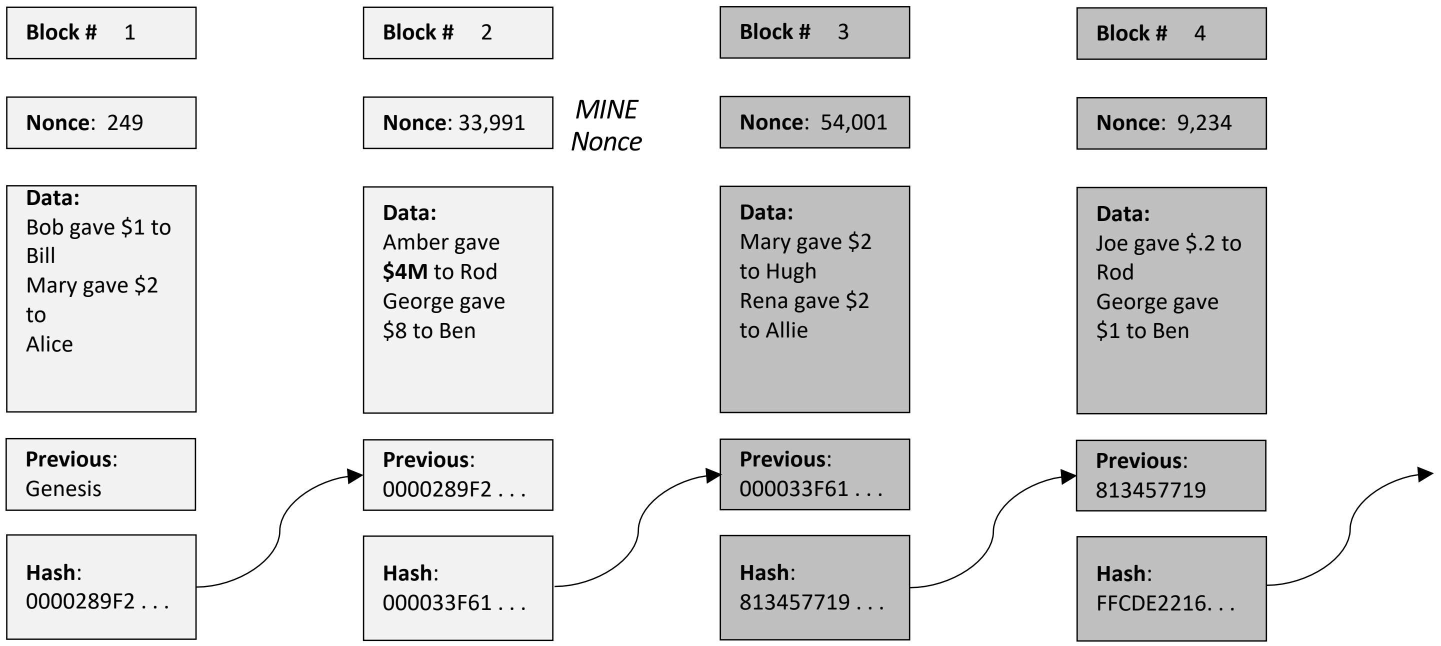
Data:
Joe gave $.2 to Rod
George gave $1 to Ben

Previous:
6F2E1F6020

Hash:
08694116C . . .

Messes Up Hashes for all Subsequent Blocks (Lose Leading 4 Zero Property)

Modern Block Chain Usage



Block # 1

Nonce: 249

Data:
Bob gave $1 to Bill
Mary gave $2 to
Alice

Previous:
Genesis

Hash:
0000289F2 . . .

Block # 2

Nonce: 33,991

MINE Nonce

Data:
Amber gave **$4M** to Rod
George gave $8 to Ben

Previous:
0000289F2 . . .

Hash:
000033F61 . . .

Block # 3

Nonce: 54,001

Data:
Mary gave $2 to Hugh
Rena gave $2 to Allie

Previous:
000033F61 . . .

Hash:
813457719 . . .

Block # 4

Nonce: 9,234

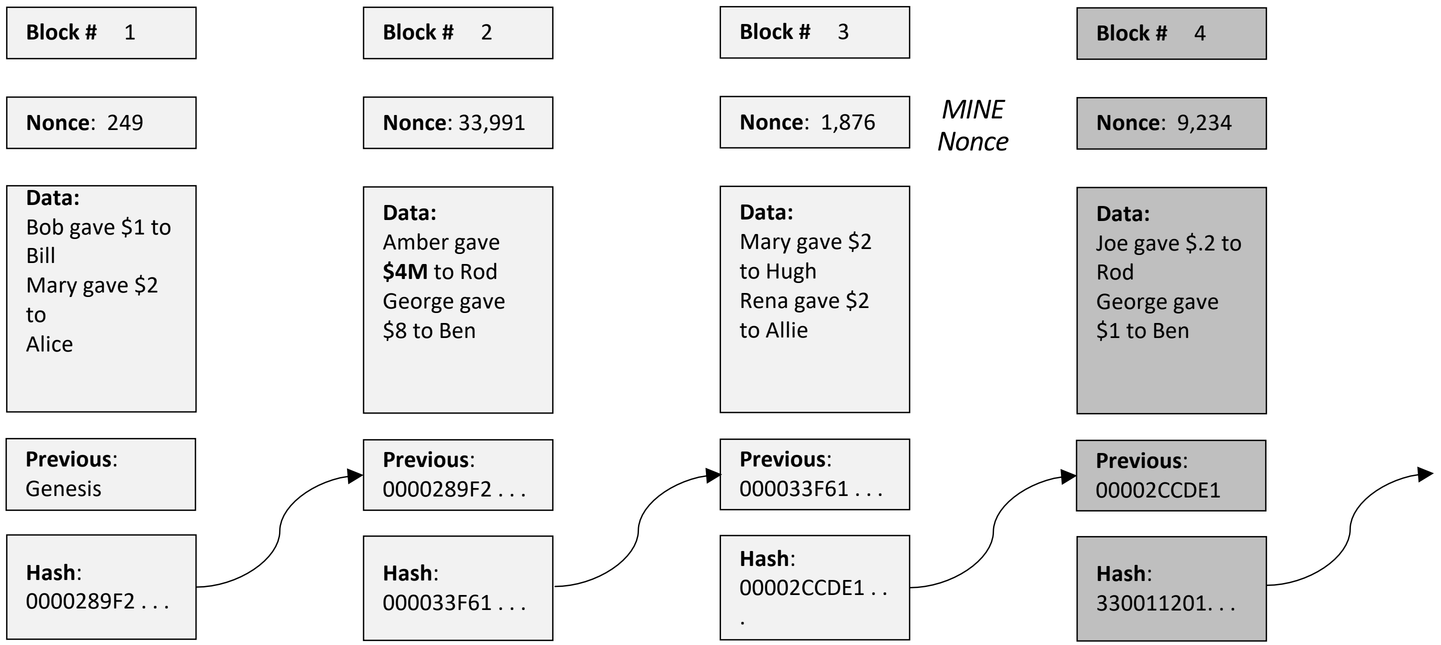
Data:
Joe gave $.2 to Rod
George gave $1 to Ben

Previous:
813457719

Hash:
FFCDE2216. . .

Week 6

Modern Block Chain Usage



Block # 1

Nonce: 249

Data:
Bob gave $1 to Bill
Mary gave $2 to Alice

Previous: Genesis

Hash: 0000289F2 . . .

Block # 2

Nonce: 33,991

Data:
Amber gave **$4M** to Rod
George gave $8 to Ben

Previous: 0000289F2 . . .

Hash: 000033F61 . . .

Block # 3

Nonce: 1,876

MINE Nonce

Data:
Mary gave $2 to Hugh
Rena gave $2 to Allie

Previous: 000033F61 . . .

Hash: 00002CCDE1 . . .

Block # 4

Nonce: 9,234

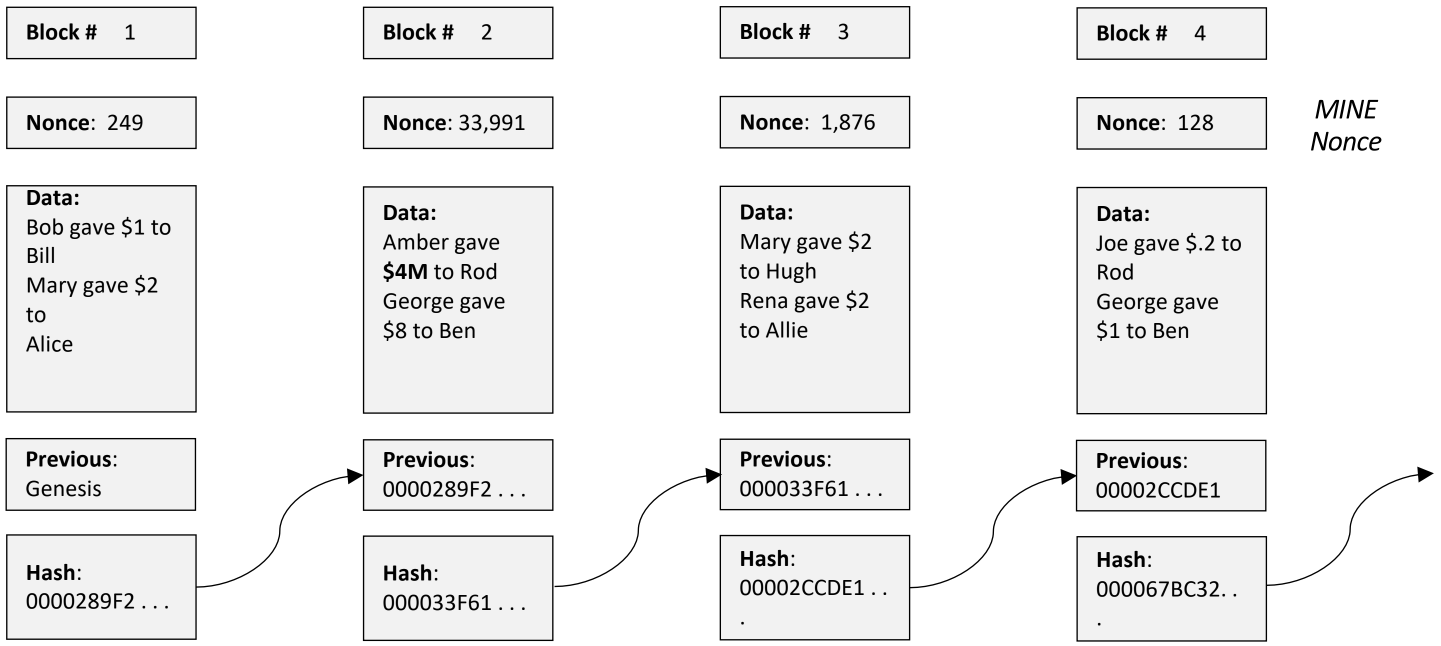
Data:
Joe gave $.2 to Rod
George gave $1 to Ben

Previous: 00002CCDE1

Hash: 330011201. . .

Week 6

Modern Block Chain Usage



Block # 1

Nonce: 249

Data:
Bob gave $1 to
Bill
Mary gave $2
to
Alice

Previous:
Genesis

Hash:
0000289F2 . . .

Block # 2

Nonce: 33,991

Data:
Amber gave
$4M to Rod
George gave
$8 to Ben

Previous:
0000289F2 . . .

Hash:
000033F61 . . .

Block # 3

Nonce: 1,876

Data:
Mary gave $2
to Hugh
Rena gave $2
to Allie

Previous:
000033F61 . . .

Hash:
00002CCDE1 . .
.

Block # 4

Nonce: 128

MINE
Nonce

Data:
Joe gave $.2 to
Rod
George gave
$1 to Ben

Previous:
00002CCDE1

Hash:
000067BC32. .
.

Week 6