CSE 3231
Computer Networks

Cryptography
part 1

William Allen, PhD

Spring 2022

Privacy of Network Traffic

- We have seen how applications use Transport-layer protocols to make connections and exchange data
- However, un-encrypted connections over shared media *can be monitored* by anyone with access to that shared media
  - as you saw in assignment # 2, some protocols like HTTP do not hide the user's data while in transit and it can be captured and monitored

Privacy of Network Traffic

- We will look at how cryptography works and how it can be applied to network connections to secure private data
  - First, we will have a quick overview of cryptography concepts
  - Later, we will look at how these concepts are applied in specific network protocols to protect data privacy and/or to determine if the packet's private data was modified

Network Security Goals

- Confidentiality – (secrecy or privacy) assure that assets can only be accessed by authorized individuals - i.e., *private data should be hidden*
- Integrity – assets can only be modified by authorized individuals and in authorized ways - i.e., *we need to know when it has been modified*
- Availability – assets can be accessed *when needed*, by anyone who is authorized to - i.e., *if we can't access our data, it is useless to us*

Cryptography

- “secret writing”
  - been used for 1000’s of years
  - cryptographers are constantly defeating old techniques and creating new ones
  - used properly, can ensure Confidentiality and/or Integrity, but not Availability
  - in reality, often misunderstood or misused, resulting in a failure to protect data
  - best modern cryptographic systems are based on very complex mathematics

Terms

- encode / decode – translate whole words or phrases at a time
- encipher / decipher – convert characters or symbols *individually*
- encrypt / decrypt – covers both methods
- cryptosystem – system that supports both encryption and decryption of data

More Terms

- plaintext – original message (sequence of characters or bytes), $P = \langle p_1, p_2, \ldots, p_n \rangle$
- ciphertext – encrypted version of plaintext, $C = \langle c_1, c_2, \ldots, c_n \rangle$
- encryption/decryption algorithms – steps to go between plaintext and ciphertext:

  $C = E(P)$ and $P = D(C)$, where E is encryption and D is decryption

Even More Terms

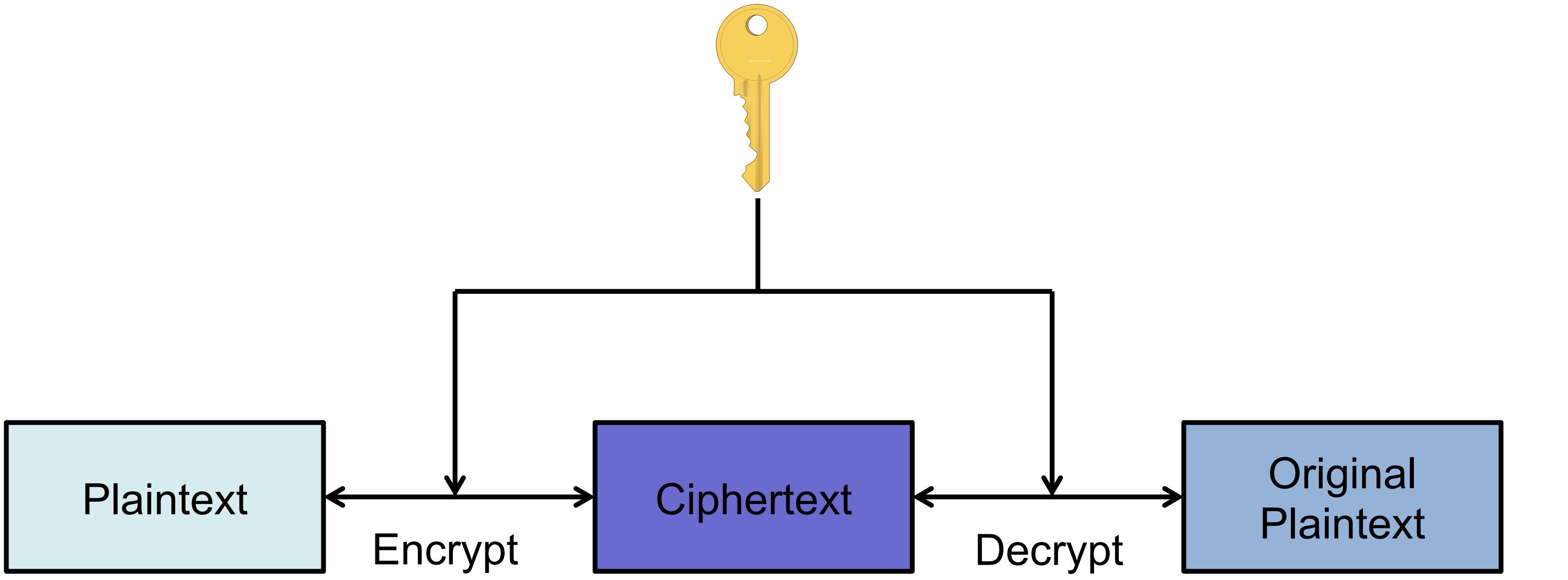
- key – information or device used to encrypt/decrypt plaintext (symbol K)
- symmetric encryption – the same key is used for both encryption and decryption

$$C = E(K, P) \qquad\qquad P = D(K, C)$$

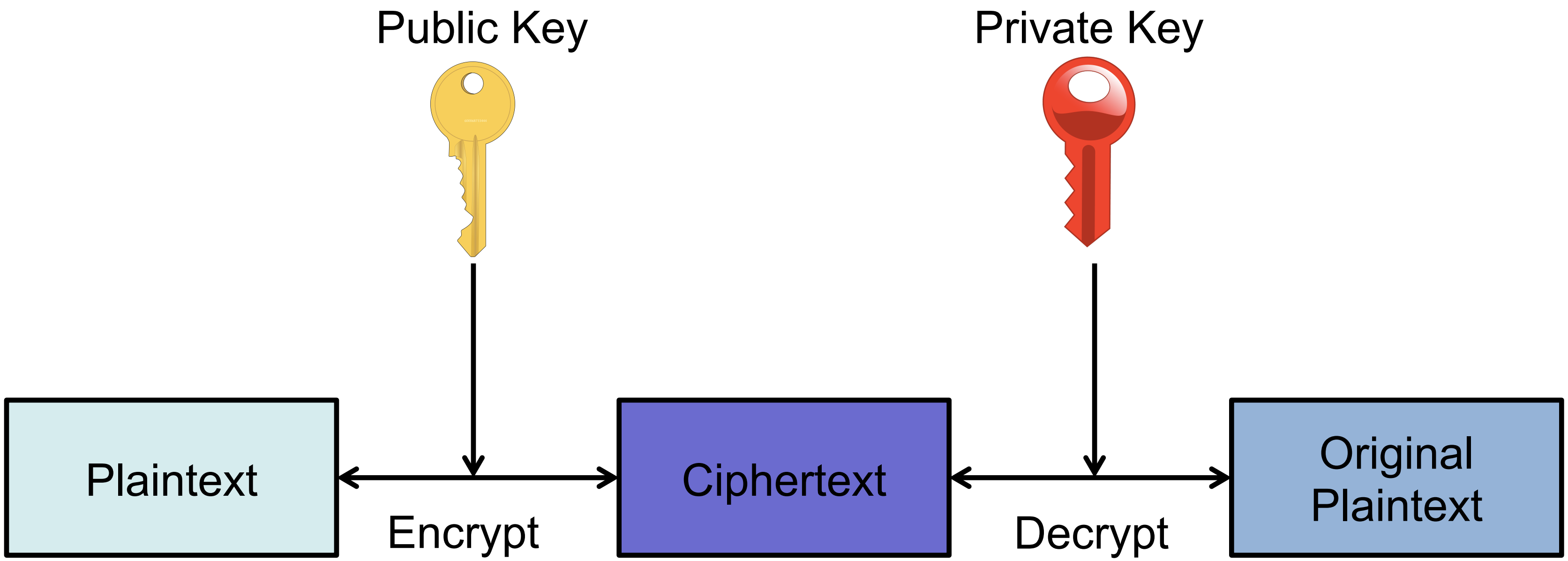
- asymmetric encryption – different keys are used for encryption and decryption

$$C = E(K_E, P) \qquad\qquad P = D(K_D, C)$$

Symmetric Cryptography



Asymmetric Cryptography



Still Even More Terms

- Breakable Encryption – can the plaintext be found without access to the key?
  - yes, it isn’t always necessary to have the key!
  - however, some algorithms are theoretically unbreakable or will take too long to break
- Cryptanalysis – analyzing messages with the purpose of breaking an encryption
  - techniques include: letter frequency analysis, mathematical analysis and brute force

The Most Important Principle of Cryptography

Security through obscurity doesn't work

– hidden algorithms will be discovered

– security is based entirely on secret keys

- Auguste Kerckhoffs (1883):

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

– keys should be reasonably easy to deliver and use

– it should not require multiple people to encrypt or decrypt a message

A Simple Substitution Example

Substitution Cipher:

The set of substitution characters form the encryption key

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
k	p	a	v	w	r	g	n	f	t	b	c	u	x	d	h	s	y	i	m	j	z	l	q	o	e

Encryption:

M	E	E	T		A	T		D	A	W	N

Substitute a character from the key
for each plaintext character

A Simple Substitution Example

Substitution Cipher:

What is the disadvantage of using this method?

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
k	p	a	v	w	r	g	n	f	t	b	c	u	x	d	h	s	y	i	m	j	z	l	q	o	e

Encryption:

You have to deliver the key to the person who decrypts the message.

M	E	E	T		A	T		D	A	W	N
u	w	w	m		k	m		v	k	l	x

Decryption:

u	w	w	m		k	m		v	k	l	x
M	E	E	T		A	T		D	A	W	N

Substitution and Transposition

- *Substitution* replaces characters in the plaintext with different characters
  - causes *confusion*, changing plaintext into a message that is not readable
    - but, the order of the characters is not changed
  - decrypt by reversing the replacements
- *Transposition* reorders characters
  - creates *diffusion*, relocating characters to break up the original order of the characters
    - but, characters are not changed, just their location
  - decrypt by reversing the relocations

A Simple Transposition Example

Example: Columnar Transposition

Assume that the key is 4

- write message in rows that are 4 columns wide and read down the columns

Message to encipher:

WE ARE DISCOVERED SEND HELP

A Simple Transposition Example

Plaintext:

WE ARE DISCOVERED SEND HELP

W	E	A	R
E	D	I	S
C	O	V	E
R	E	D	S
E	N	D	H
E	L	P	

guessing the number of columns works well for breaking a simple example like this

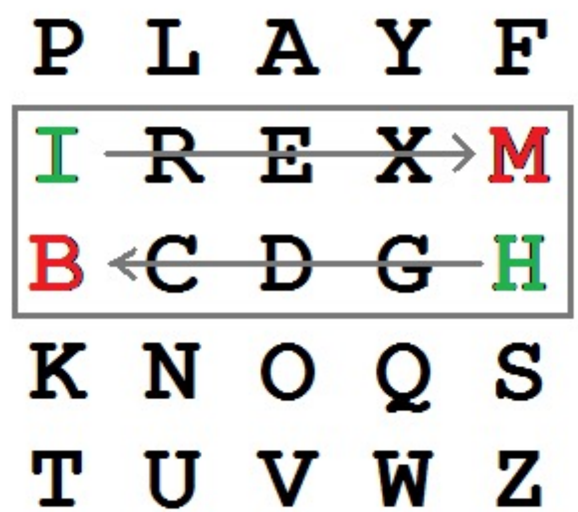
Ciphertext:

wecreeeedoenlaivddprsesh

Playfair Cipher

- Playfair Cipher – a more complex form of transposition cipher that was used by several governments in WW1 & WW2

- Playfair arranges letters in a grid and replaces plaintext with ciphertext based on their position in the grid: I→M, H→B



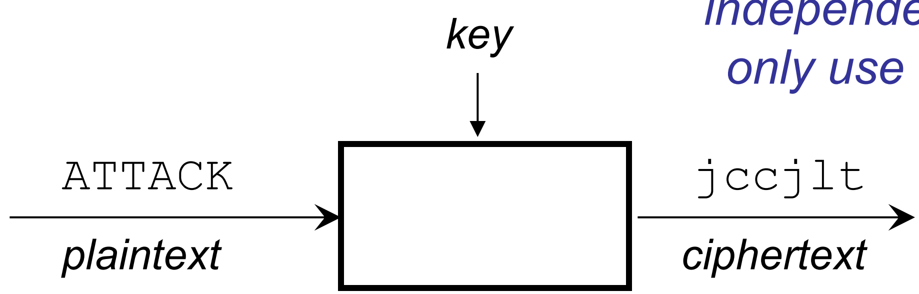
Stream vs. Block Ciphers

- Stream cipher – independently converts each character of plaintext into a character of ciphertext (i.e., substitution)
- Block cipher – converts a group of plaintext characters into a block of ciphertext
  - we may convert by grouping a number of bits instead of a block of characters
  - can use substitution and/or transposition

Stream vs. Block Ciphers

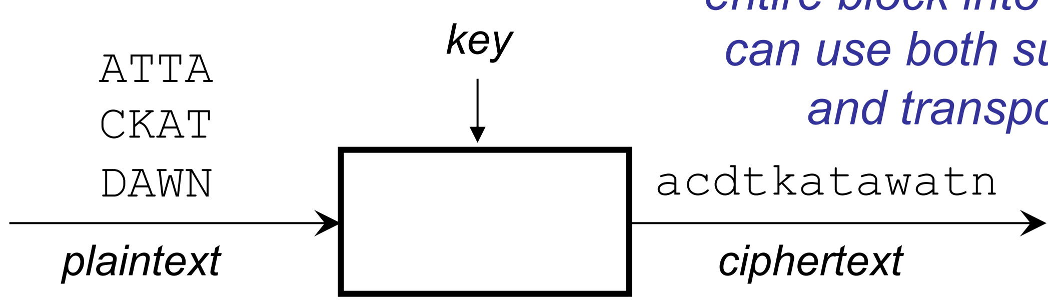
- Stream cipher:

converts each character independently, but can only use substitution



- Block cipher:

gathers characters into a block and converts the entire block into ciphertext, can use both substitution and transposition



Stream vs. Block Ciphers

Stream:

- Fast(er), errors are easier to recover from
- ✗ Doesn’t hide character frequencies (a common cryptanalysis tool)
- ✗ Attacker can easily insert or replace individual characters without affecting other characters

Block:

- May hide letter frequencies
- Can’t insert or change individual letters
- ✗ Slower & any errors can affect a whole block

Using Stream vs Block

- Data in a file can easily be encrypted either way since the file is static until used
- Data being transmitted over a network can be encrypted either way, depending on the method of transmission
  - real-time data easily be encrypted using a stream approach, but it is usually held in a buffer until a large enough block exists
  - files or email messages are already in blocks

“Real-time” Block Encryption

- If the data rate is low, the user won’t notice the additional processing time
  - many protocols already buffer data
- If the data transfer rate is high, we can still buffer the data in small, fixed-size blocks (e.g., 64 bits) and use block encryption
  - causes data to be sent in block-sized groups
  - but, if it is processed and transmitted fast enough, the user won’t notice the delay

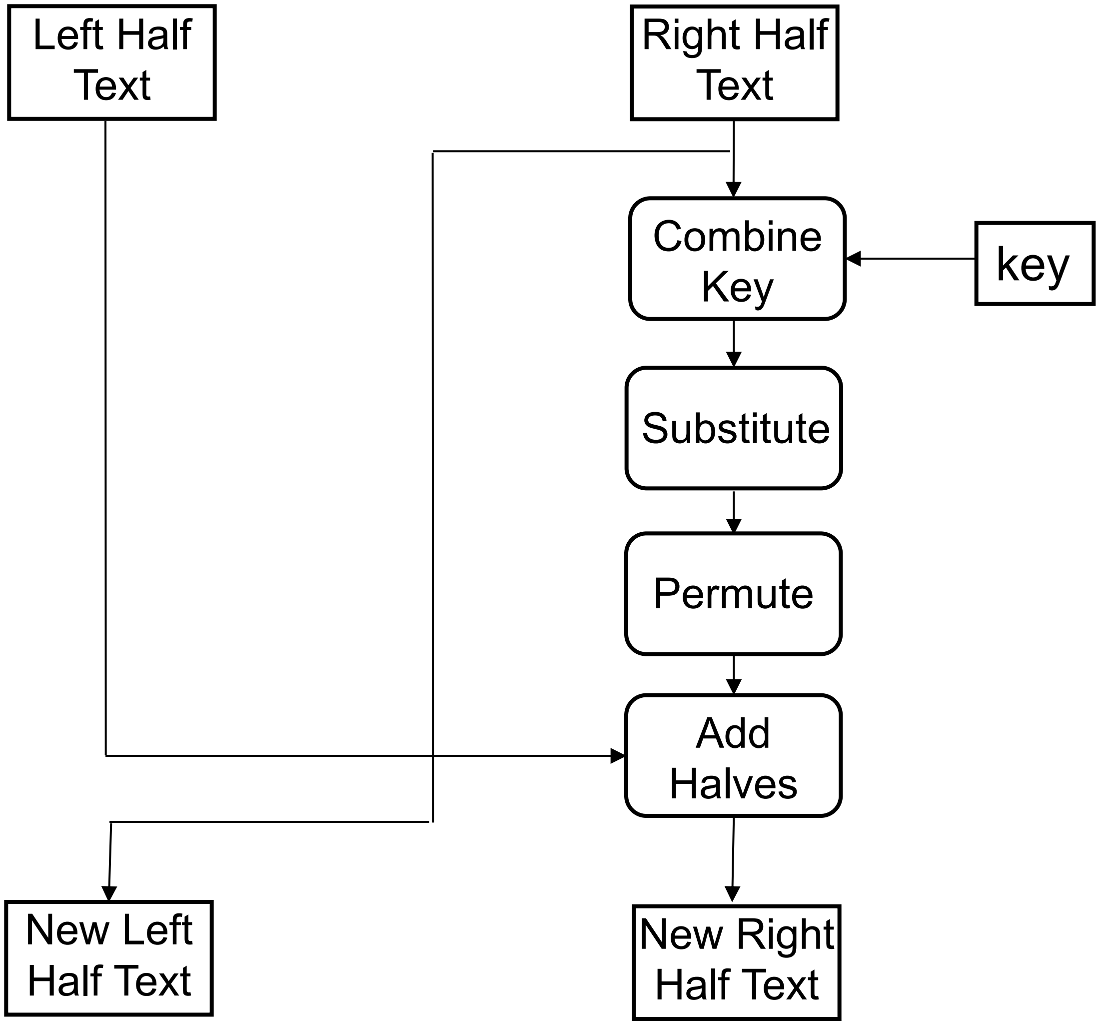
"Trustworthy" Encryption

- Trustworthy algorithms:
  - Are based on sound mathematics
  - Have been thoroughly checked by experts
  - Have stood the "test of time"
- Note: trustworthy algorithms *don't* guarantee secure systems
  - clearly, developing a new technique takes time and may still be broken in a few years
  - many systems fail due to poor implementation

Encryption Algorithms

- Data Encryption Standard (DES) (1977)
  – symmetric encryption
  – uses both substitution and transposition
  – uses normal math operations, no special HW
  – uses blocks of 64 bits and a 64 bit key, but only 56 bits of the key are used
    - originally considered strong enough
    - 56-bit key DES can now be broken in a few hours
  – minor flaws in design also reduce its security

DES Cycle (repeated 16 times)



DES Keys

- The subkey for each of the 16 rounds is a permutation of the key used in the previous round
- To decrypt, the keys are used in the reverse order as for encrypting, but the order of the DES cycle stays the same
- Can be implemented hardware/firmware for faster operation

DES Improved

- DES is designed for a 56-bit key
- Triple DES (TDES) uses 3 keys (168 bits)

$$C = E(K_3, E(K_2, E(K_1, P)))$$

  - improved security but, since it's not a new technique, it just takes more time to break
  - due to several well-known attacks, it is only rated as being equal to 80-bit security
    - meet-in-the-middle, known & chosen plaintext
  - TDES is widely used for financial transactions

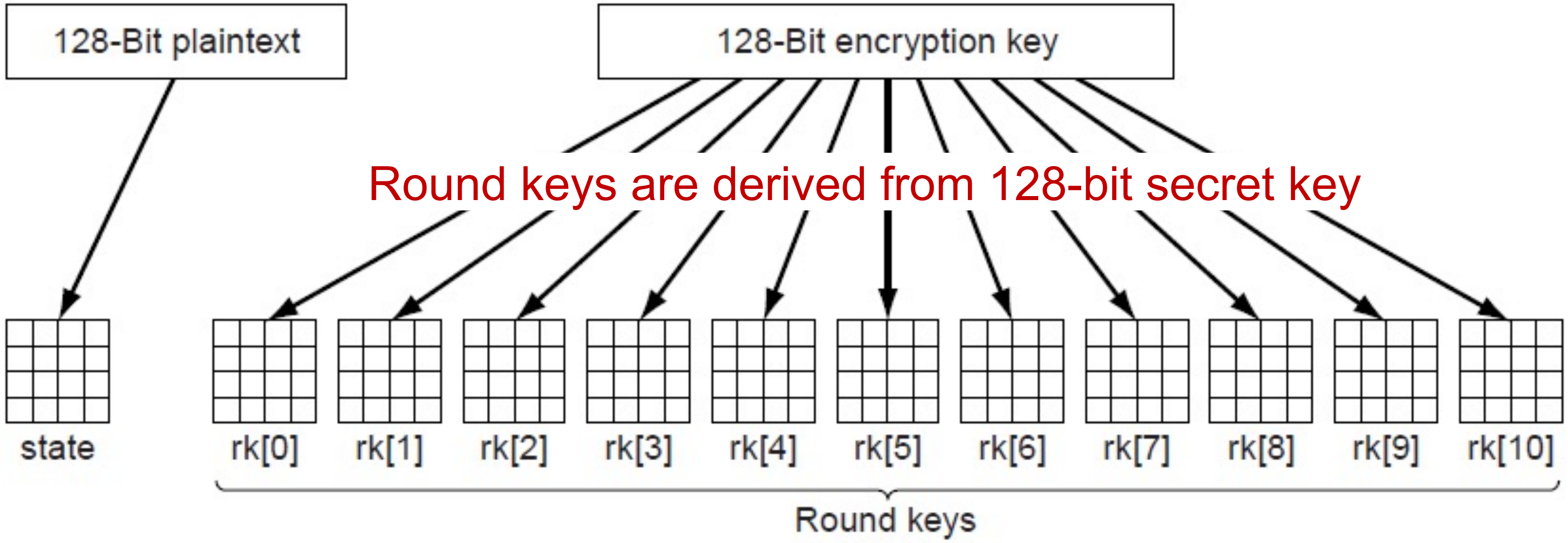
A New Approach: AES or Rijndael (Rijmen, Daemen)

- Advanced Encryption Standard (AES)
  - NIST competition to find a new approach for symmetric encryption
  - winner [Rijndael (*RINE dahl*) algorithm] made publicly available, widely tested and analyzed
  - can use 128, 192 or 256 bits for keys
  - runs 10, 12 or 14 cycles, each of which does:
    - byte substitution in 128-bit blocks (diffusion)
    - transposition (confusion) and column shifting
    - XORs with part of key (confusion)

Advanced Encryption Standard

For example, AES with 10 rounds and a 128-bit key uses a block size that is 128-bits

- Each round uses a key derived from the 128-bit key
- Each round has a mix of substitutions and rotations
- All steps are reversible to provide for decryption



AES Advantages

- Much faster to encrypt/decrypt than DES
- AES algorithm can be extended for longer keys and more cycles
- Tested for two years with no known bugs
  - Only effective attacks use side-channel techniques which require direct access to the machine doing encryption/decryption
  - U.S. Government accepted AES for encrypting *Secret* and *Top Secret* data

Public Key Encryption

- Asymmetric keys (one public, one private)

  Encrypt: $C = E(K_{public}, P)$

  Decrypt: $P = D(K_{private}, C)$

- Can provide confidentiality and integrity, but with poor performance compared to symmetric systems with a single key
  - can be 1000's of times slower for the same sized key

Using Public Key Encryption

- We will assume that each user has two keys, a private key $K_{private}$ that nobody else has access to and a public key K_{public} that is available to anyone
- There are two possible goals for users
  - send a message to someone that no one can read except for the intended recipient
  - receive a message that could only have come from the person who claimed to have sent it

Two Possible Goals

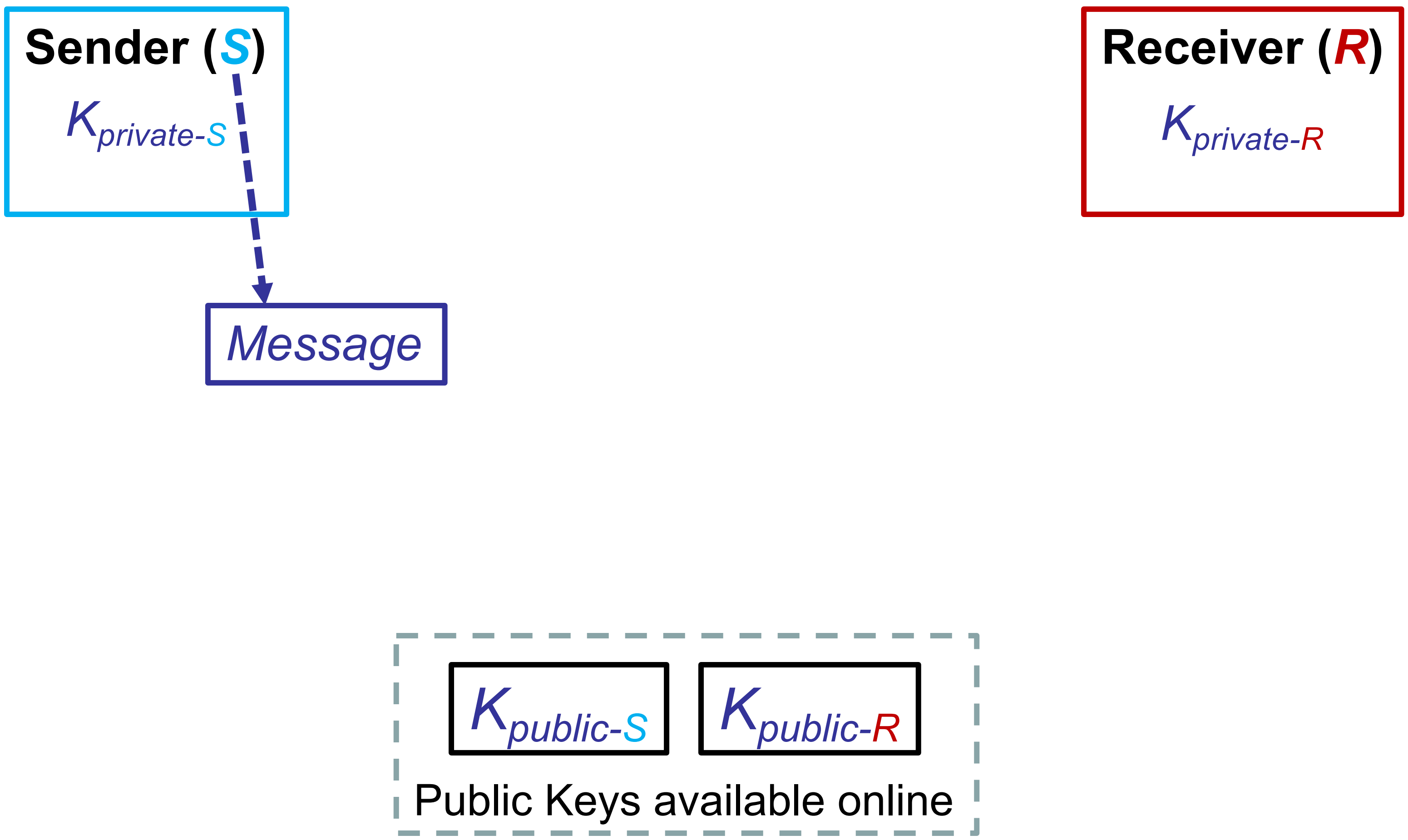
- Ensure that the *receiver is the only person who can read* the sender's message
  - The sender would use the receiver's public key and only the receiver has the private key that is required to decrypt the message
- Enable the receiver to trust that the sender is the only person who could send it
  - The sender uses their private key and only the sender's public key can be used to decrypt the message

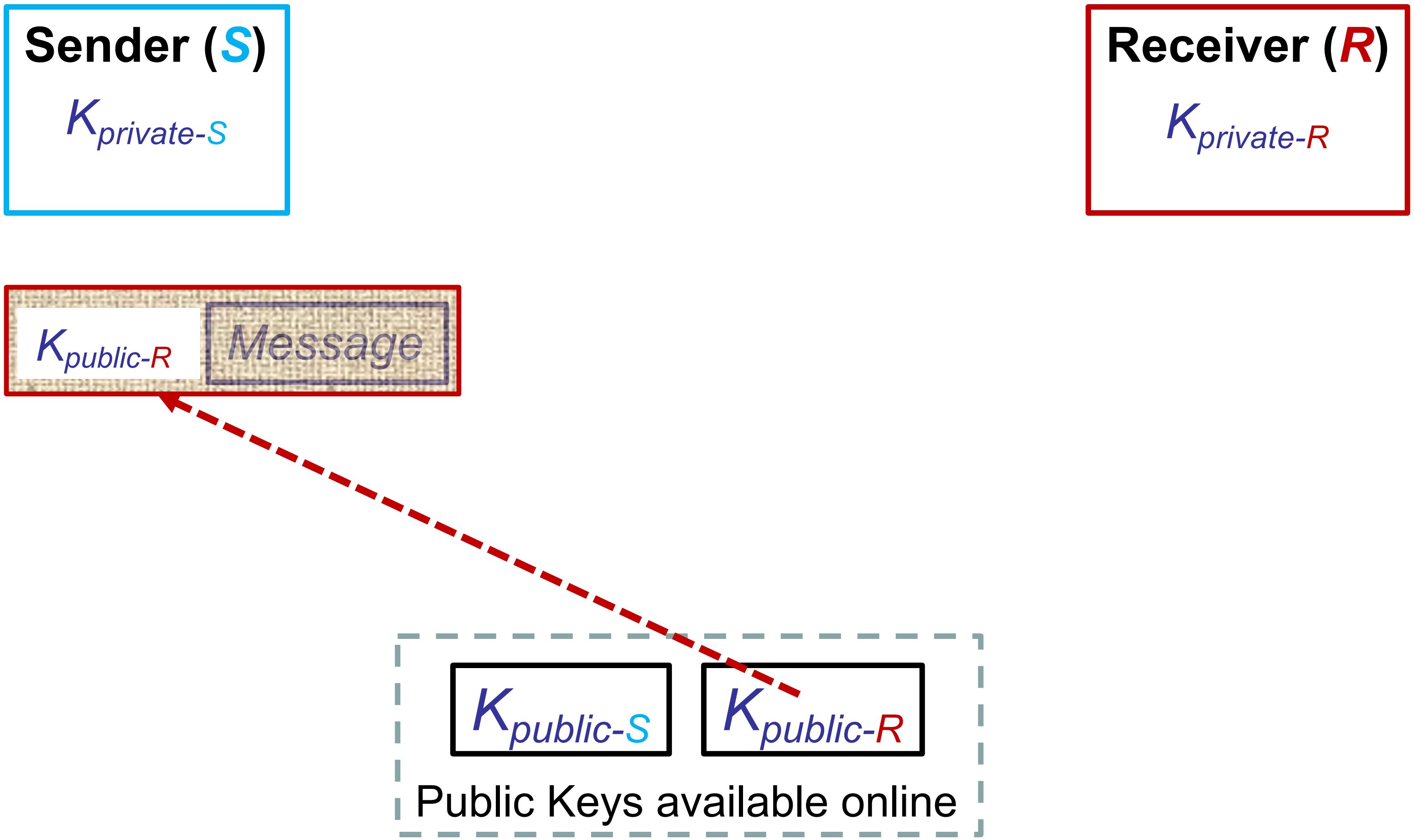
Example: Only Receiver can Read

Sender (S) sends message M to Receiver (R)

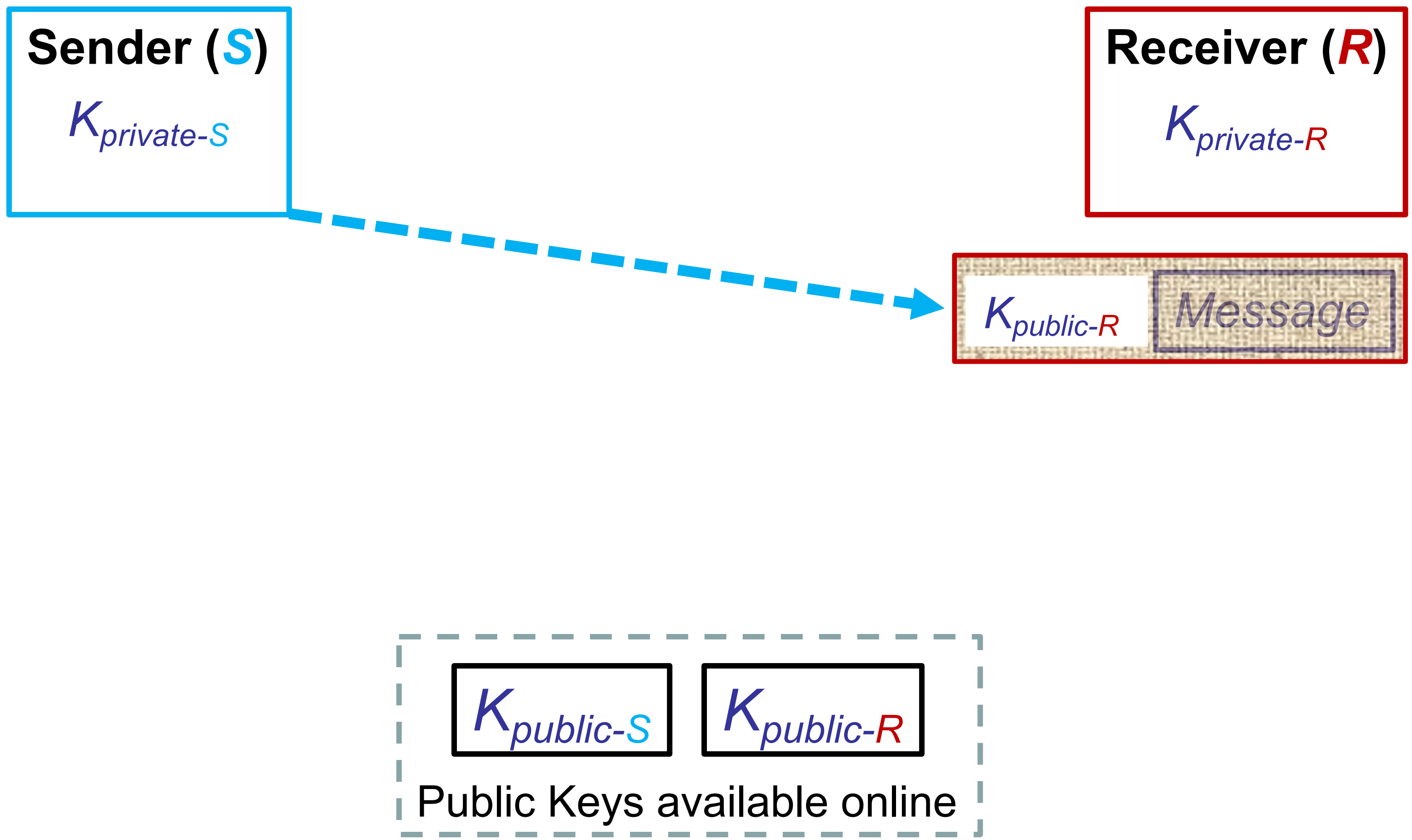
1. S encrypts message M using key $K_{public\text{-}R}$
2. S sends $E(K_{public\text{-}R}, M)$ to R
3. R decrypts $E(K_{public\text{-}R}, M)$ using $K_{private\text{-}R}$

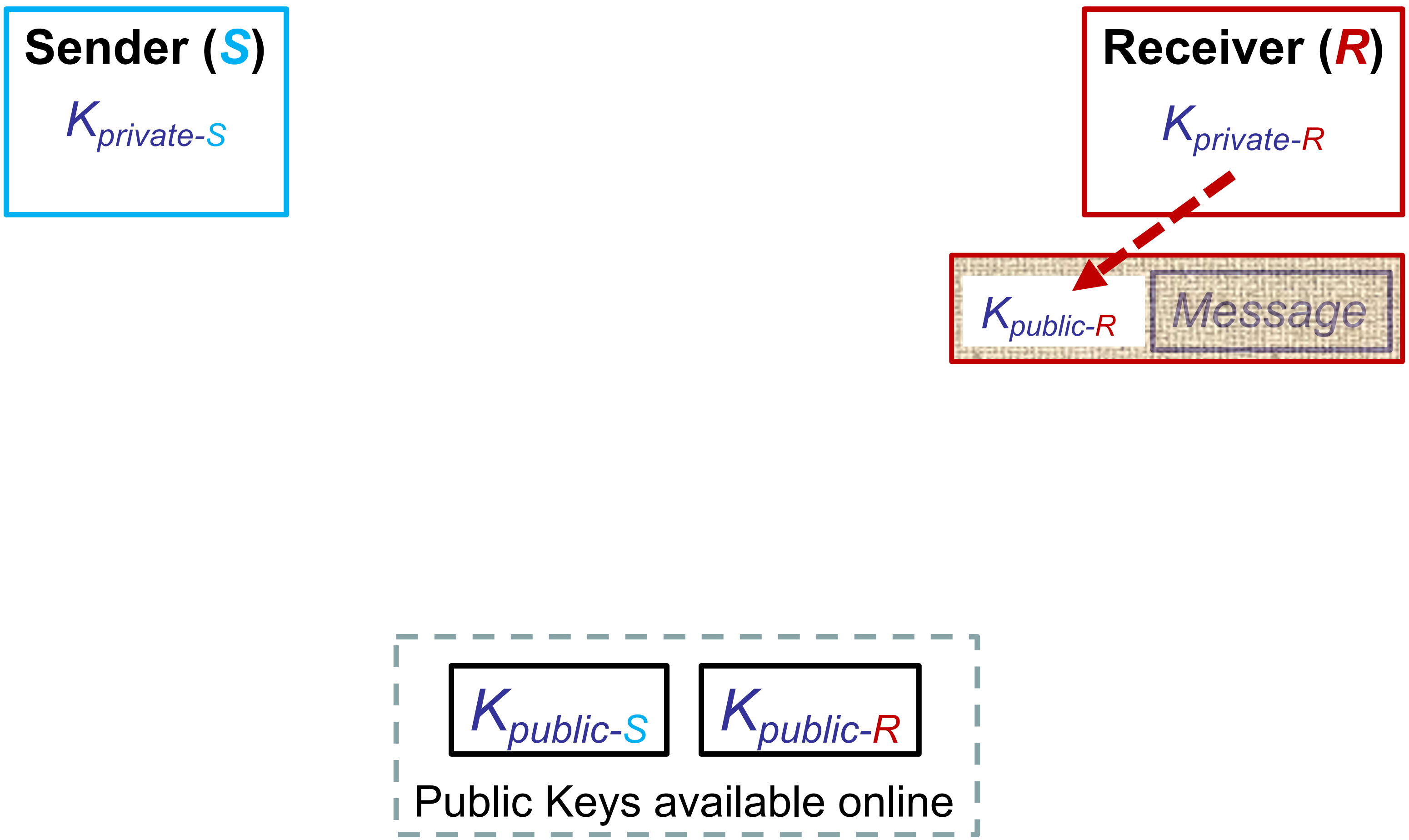
- Since only R has the key $K_{private\text{-}R}$, only R could decrypt the message that was encrypted by the key $K_{public\text{-}R}$
- Therefore, only R can read message M

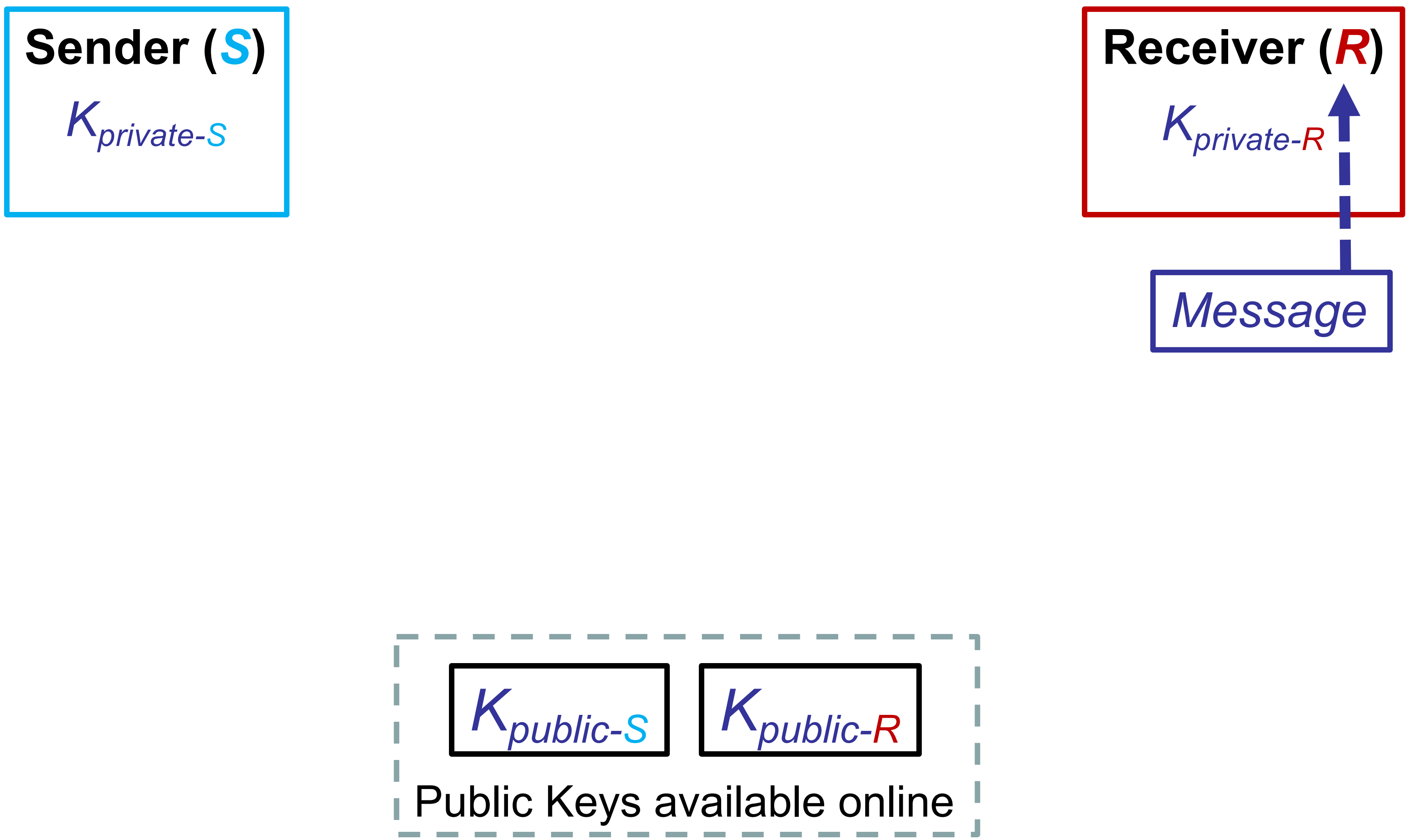

Sender (S)
$K_{private\text{-}S}$
Message
Receiver (R)
$K_{private\text{-}R}$
$K_{public\text{-}S}$
$K_{public\text{-}R}$
Public Keys available online



Sender (S)
$K_{private\text{-}S}$
Receiver (R)
$K_{private\text{-}R}$
$K_{public\text{-}R}$
Message
$K_{public\text{-}S}$
$K_{public\text{-}R}$
Public Keys available online


Sender (S)
$K_{private\text{-}S}$
Receiver (R)
$K_{private\text{-}R}$
$K_{public\text{-}R}$
Message
$K_{public\text{-}S}$
$K_{public\text{-}R}$
Public Keys available online


Sender (S)
K private-S
Receiver (R)
K private-R
K public-R
Message
K public-S
K public-R
Public Keys available online

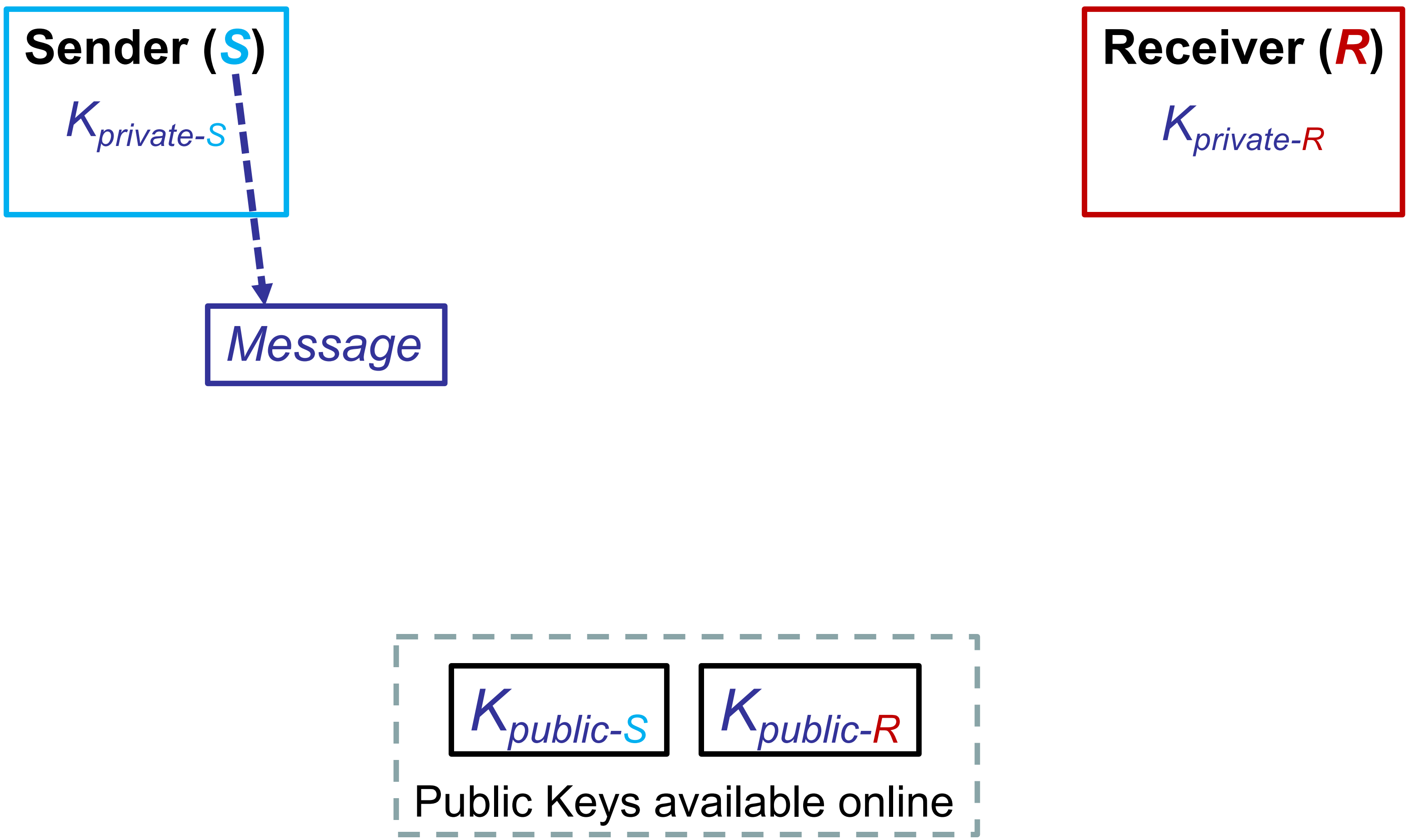

Sender (S)
$K_{private\text{-}S}$
Receiver (R)
$K_{private\text{-}R}$
Message
$K_{public\text{-}S}$
$K_{public\text{-}R}$
Public Keys available online

Example: Only Sender could Send

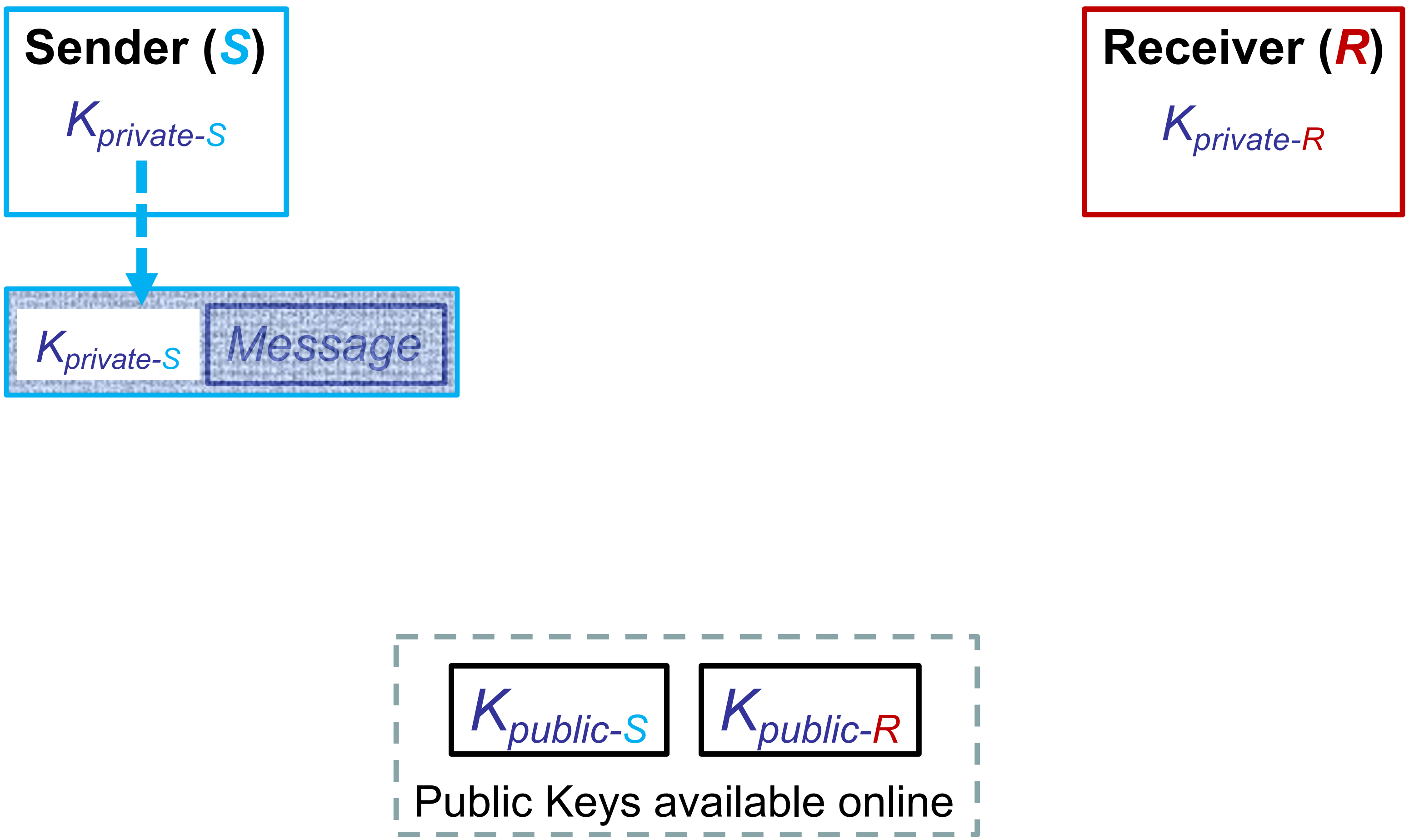
Sender (S) sends message M to Receiver (R)

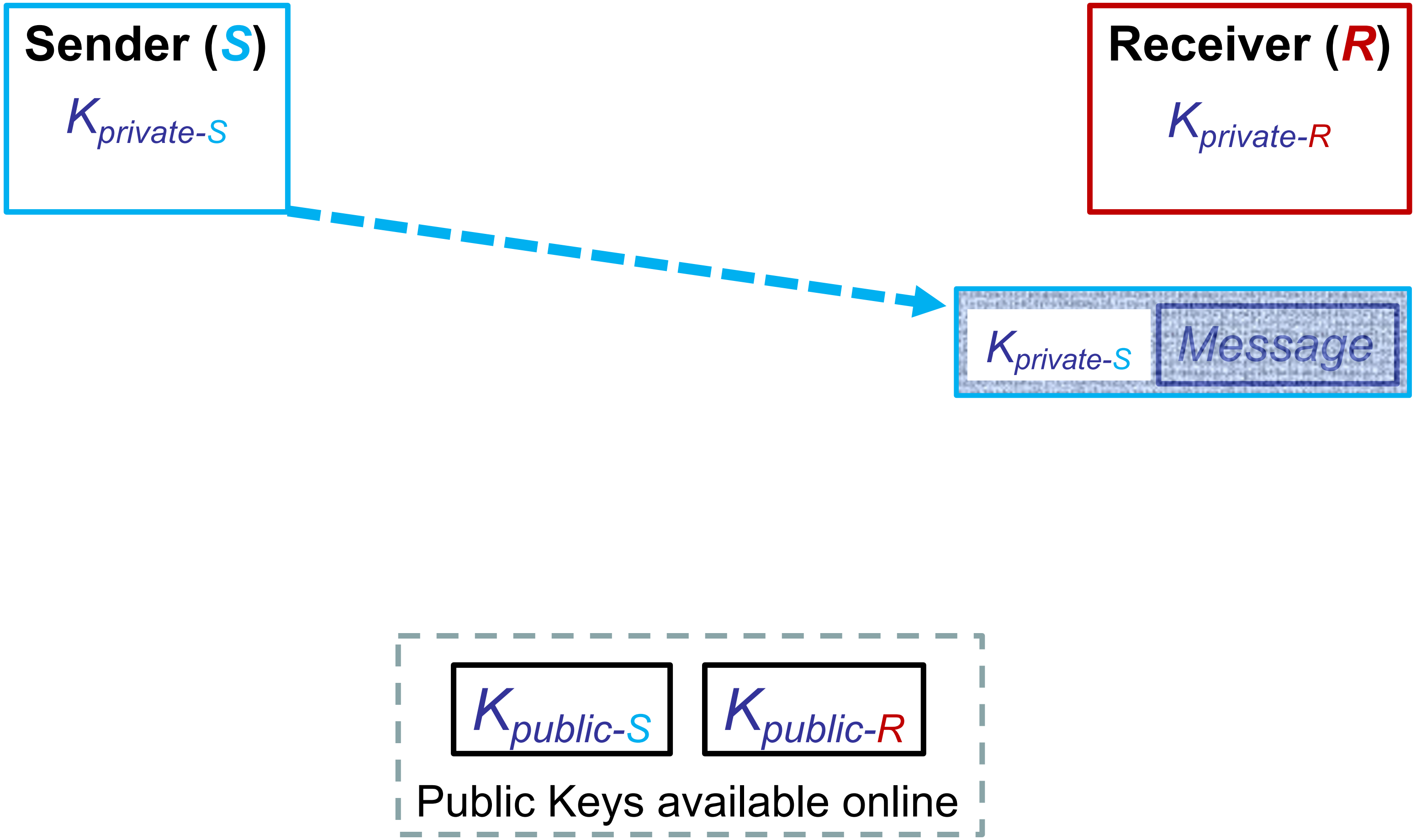
1. S encrypts message M using key $K_{private\text{-}S}$
2. S sends $E(K_{private\text{-}S}, M)$ to R
3. R decrypts $E(K_{private\text{-}S}, M)$ using $K_{public\text{-}S}$

- Since only S has the key $K_{private\text{-}S}$, only S could have created an encrypted message that would be decrypted by the key $K_{public\text{-}S}$
- Therefore, M must have come from S

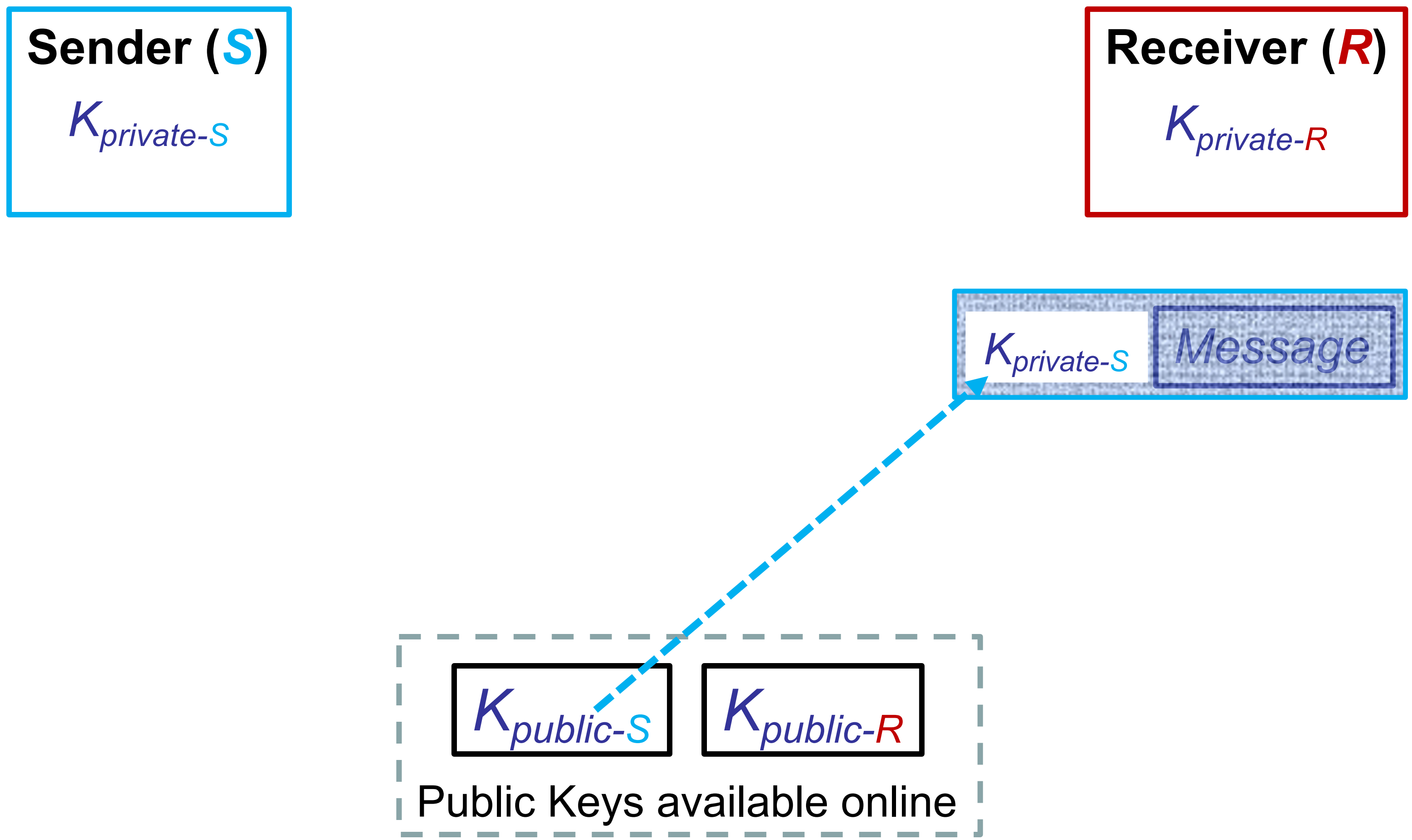


Sender (S)
$K_{private\text{-}S}$
Receiver (R)
$K_{private\text{-}R}$
Message
$K_{public\text{-}S}$
$K_{public\text{-}R}$
Public Keys available online

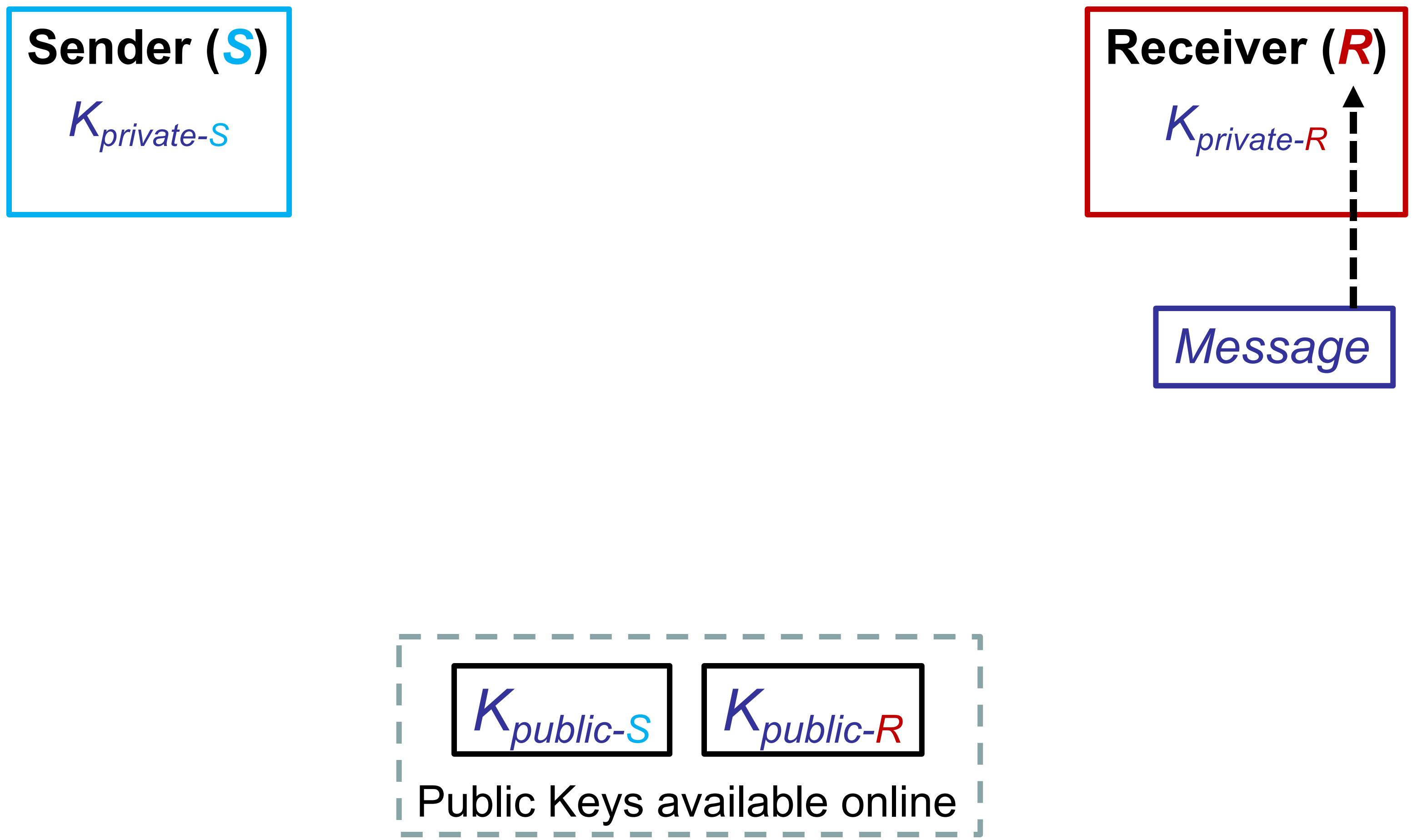

Sender (S)
$K_{private\text{-}S}$
$K_{private\text{-}S}$
Message
Receiver (R)
$K_{private\text{-}R}$
$K_{public\text{-}S}$
$K_{public\text{-}R}$
Public Keys available online



Sender (S)
$K_{private\text{-}S}$
Receiver (R)
$K_{private\text{-}R}$
$K_{private\text{-}S}$
Message
$K_{public\text{-}S}$
$K_{public\text{-}R}$
Public Keys available online



Sender (S)
$K_{private\text{-}S}$
Receiver (R)
$K_{private\text{-}R}$
$K_{private\text{-}S}$
Message
$K_{public\text{-}S}$
$K_{public\text{-}R}$
Public Keys available online



Sender (S)
$K_{private\text{-}S}$
Receiver (R)
$K_{private\text{-}R}$
Message
$K_{public\text{-}S}$
$K_{public\text{-}R}$
Public Keys available online