Cryptography and Systems

CS-576 Systems Security

Instructor: Georgios Portokalidis

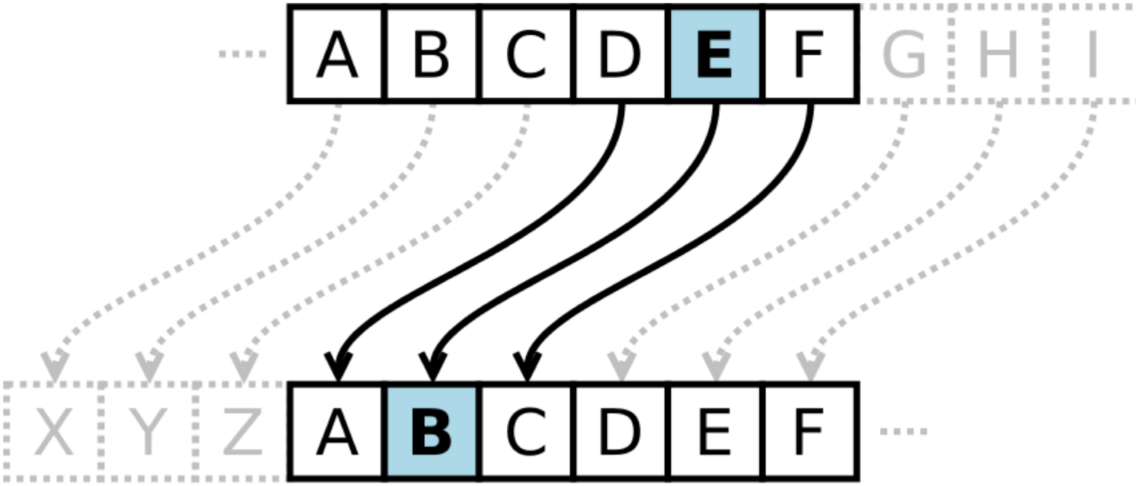
Spring 2018

History of Cryptography

Scytale

https://en.wikipedia.org/wiki/Scytale

Caesar Cipher



Shift by 3 and substitute

```
Plaintext:
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Ciphertext:
QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD
```

Caesar Cipher

$$E_n(x) = (x + n) \bmod 26, n = 3$$

```
Plaintext:
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Ciphertext:
QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD
```

Goals of Cryptography

Confidentiality

- Keep content secret from unauthorized entities

Integrity

- Protect content from unauthorized modification

Authentication

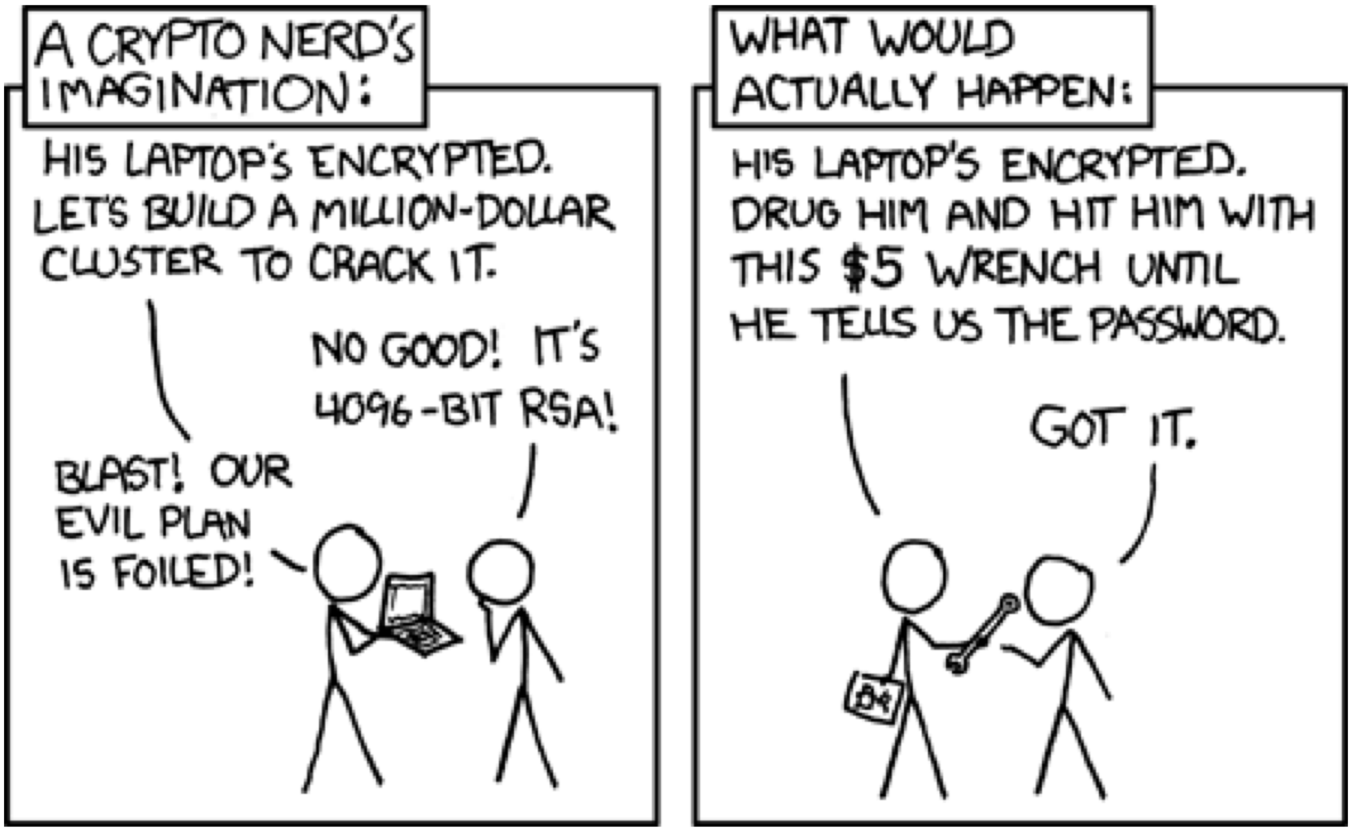
- Confirm the identity of communicating entities
- Confirm the identify of data author

Non-repudiation

- Prevent entities from denying previous commitments or actions

How to Break Crypto

Adi Shamir: “Crypto is typically bypassed, not penetrated



This Lecture

Symmetric encryption

Public-key encryption

Hashing and message authentication codes

Secure channels in practice

Public key authentication

TLS/SSL and attacks

Symmetric Encryption

Symmetric Encryption

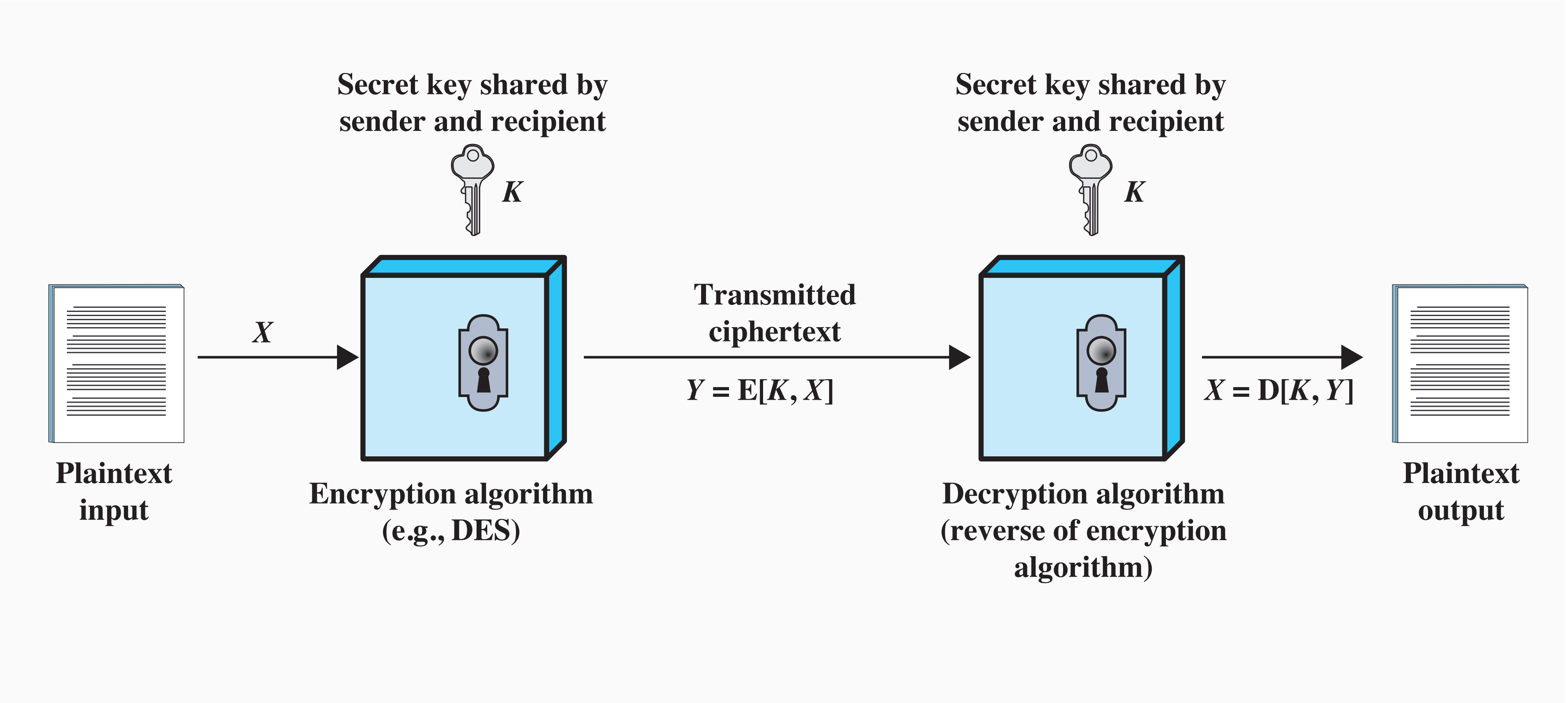
The universal technique for providing confidentiality for transmitted or stored data

Also referred to as conventional encryption or single-key/secret-key encryption

Two requirements for secure use:

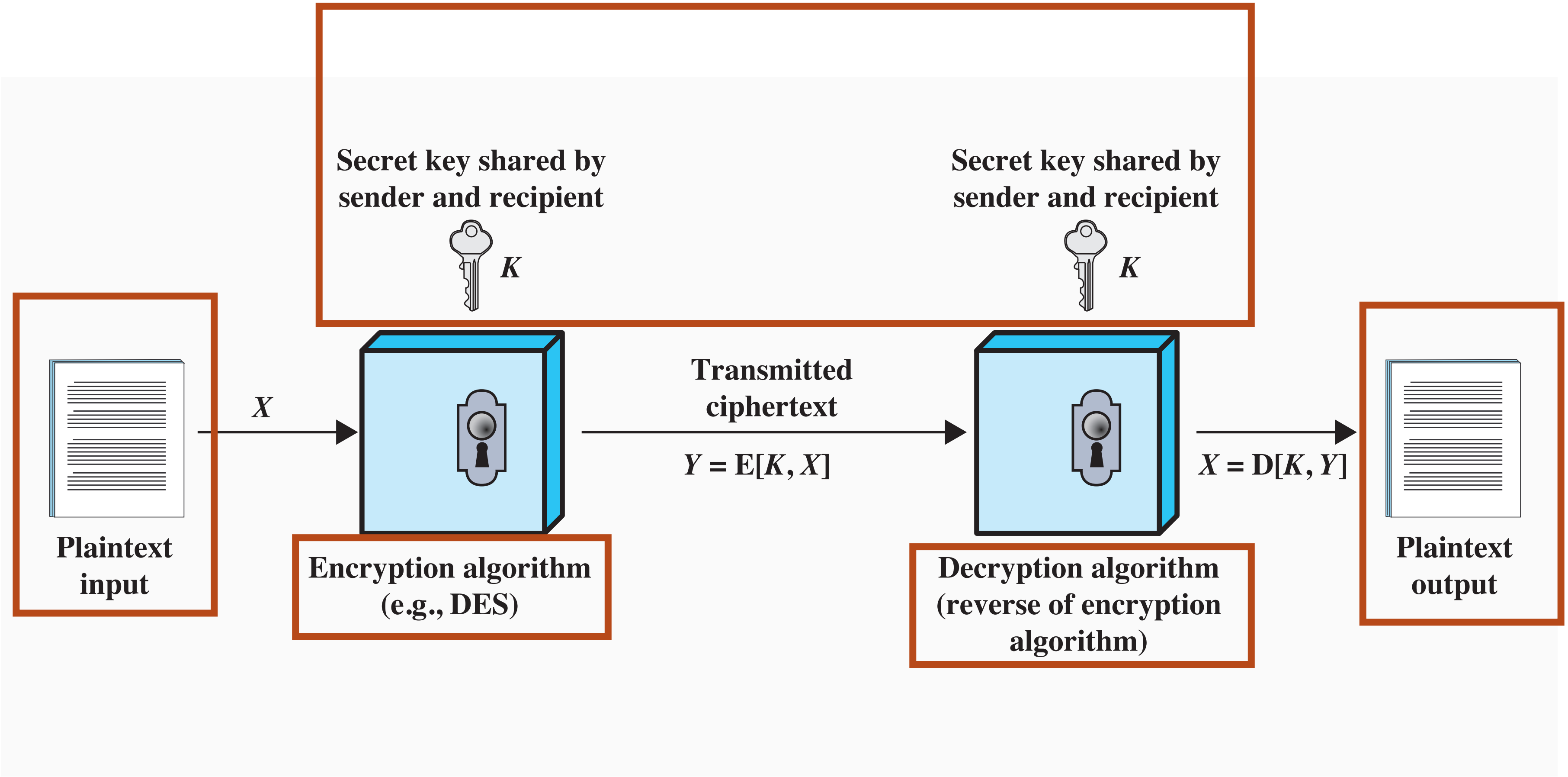
- A strong encryption algorithm
- Sender and receiver **must have obtained copies of the secret key in a secure fashion** and **must keep the key secure**

Overview



Secret key shared by
sender and recipient
K
Secret key shared by
sender and recipient
K
X
Transmitted
ciphertext
Y = E[K, X]
X = D[K, Y]
Plaintext
input
Encryption algorithm
(e.g., DES)
Decryption algorithm
(reverse of encryption
algorithm)
Plaintext
output

Terminology



Types of Ciphers

Block ciphers

Processes the input one block of elements at a time

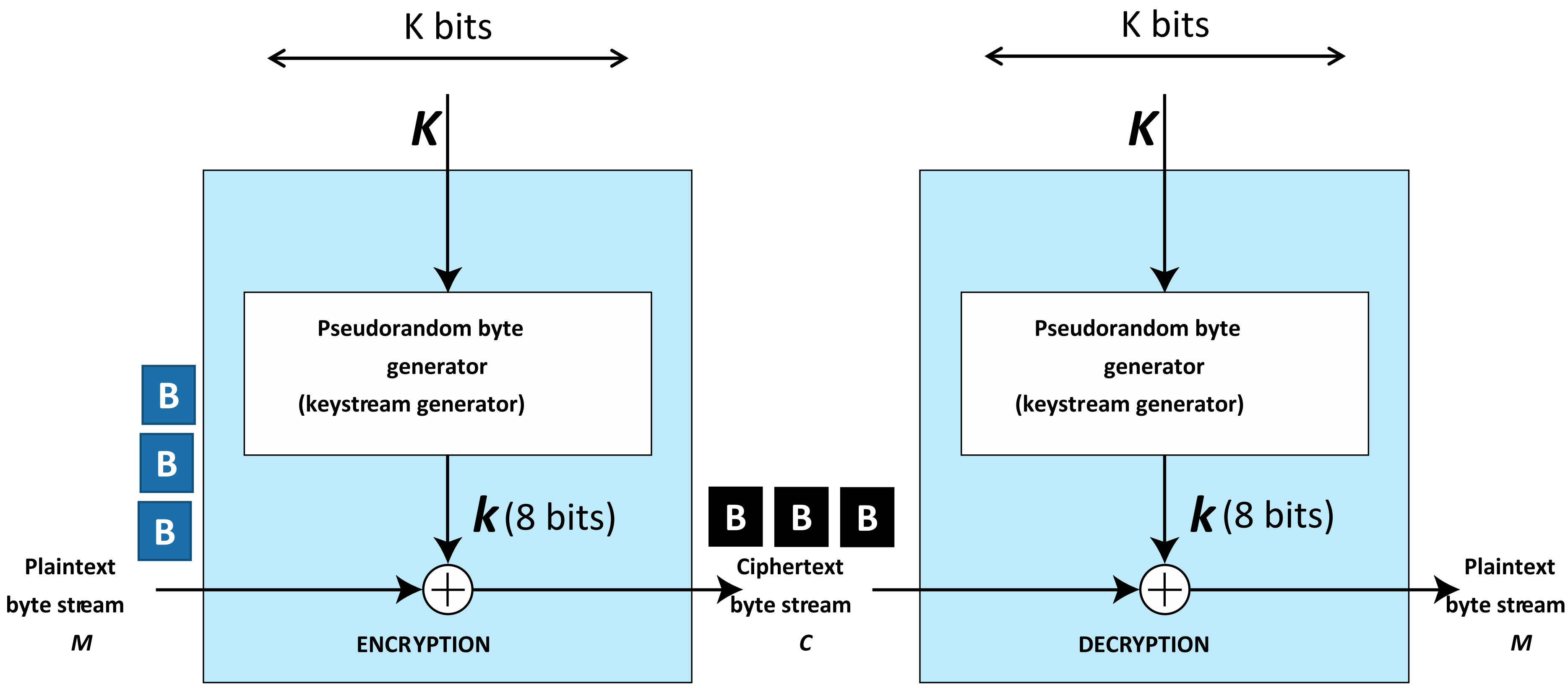
Produces an output block for each input block

Stream ciphers

Processes the input and produces output one element at a time

Requires unpredictable pseudorandom stream independent of the key

Stream Ciphers



Beware of Randomness

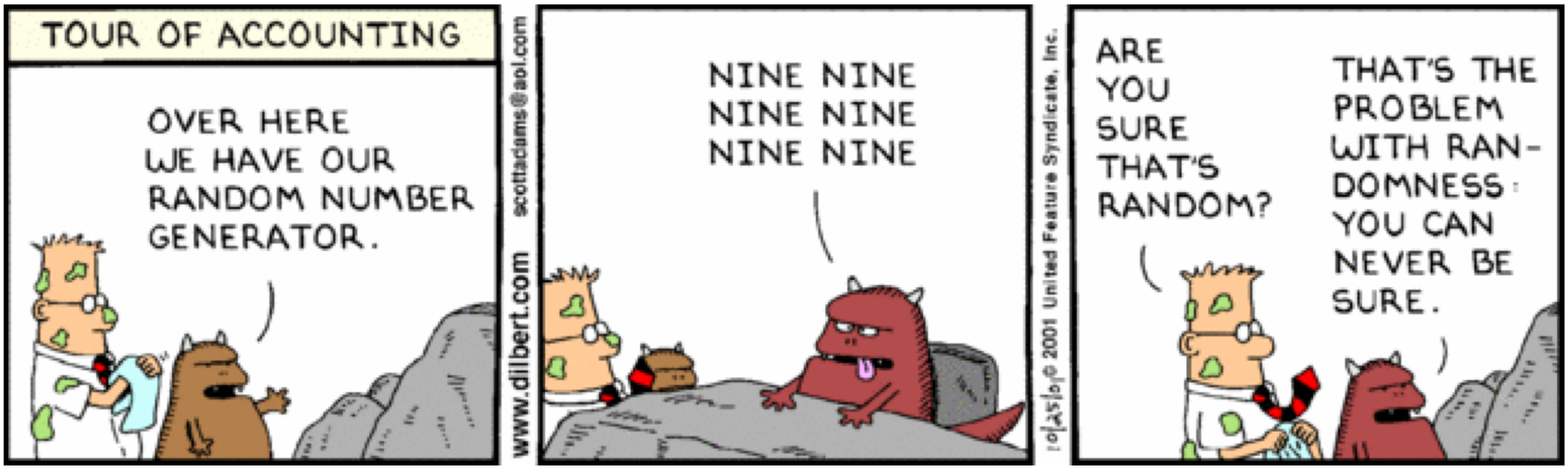
Cryptographic algorithms frequently require random numbers

A true random number generator (TRNG)

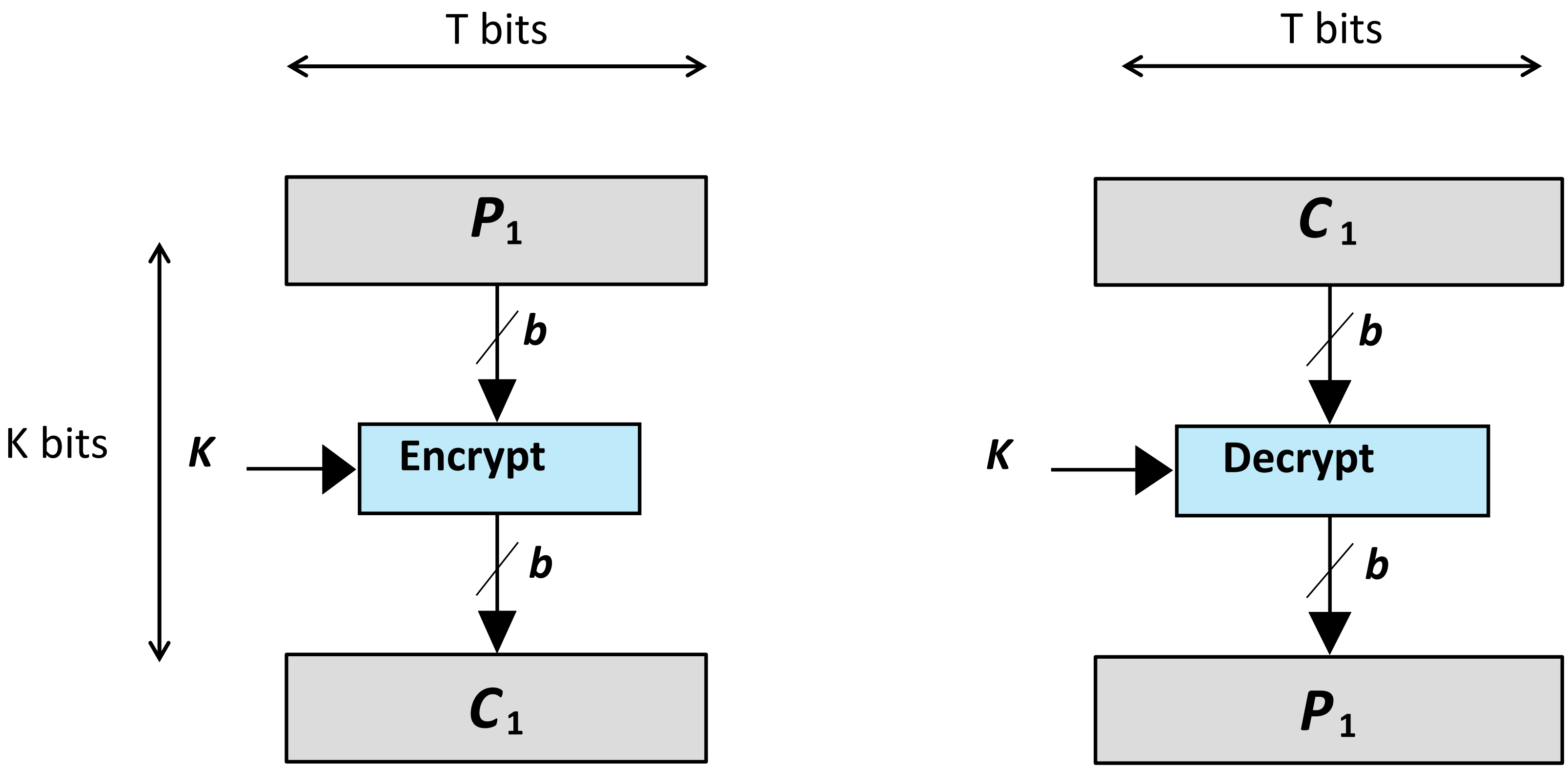
- Uses a nondeterministic source to produce randomness
- Most operate by measuring unpredictable natural processes
  - e.g., radiation, gas discharge, leaky capacitors
- Available on modern systems, but cannot provide high-volume of data

Pseudorandom numbers are

- Sequences produced that satisfy statistical randomness tests
- Likely to be predictable
- **Likely to be used by implementations**


TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
NINE NINE NINE NINE NINE NINE
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RAN-DOMNESS: YOU CAN NEVER BE SURE.
www.dilbert.com scottadams@aol.com
© 2001 United Feature Syndicate, Inc.

Blocking Ciphers



Block Ciphers - DES

Data Encryption Standard (DES)

- The most widely used encryption scheme in 1970-2000
- **Block size: 64 bits, key size: 56 bits**

Problems

- 56-bit key is too small
- Electronic Frontier Foundation (EFF) announced in July 1998 that it had broken a DES key in 56 hours

Block Ciphers - 3DES

Triple DES (3DES)

- Repeats basic DES algorithm three times using either two or three unique keys
- **Key size: 168 bits, block size: 64 bits**

Problems

- Algorithm is sluggish in software
- Small block size

Block Ciphers - AES

Advanced Encryption Standard (AES)

- A specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001
- A subset of the Rijndael cipher
- **Multiple key sizes: 128, 192 or 256 bits**
- **Block size: 128 bits**

Currently considered safe to use

Attacks

Brute force attacks

- Try all possible keys on some ciphertext until an intelligible translation into plaintext is obtained
- On average half of all possible keys must be tried to achieve success

Time Required to Brute-force

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years

Attacks

Brute force attacks

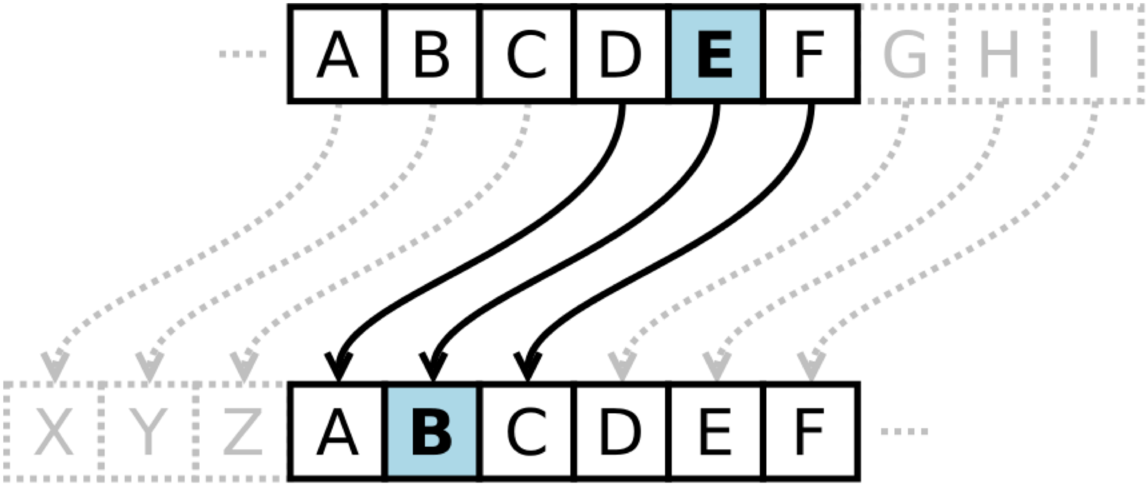
- Try all possible keys on some ciphertext until an intelligible translation into plaintext is obtained
- On average half of all possible keys must be tried to achieve success

Cryptanalytic attacks

- Exploit the characteristics of the algorithm and attempt to deduce **a specific plaintext** or the **key** being used
- Requires…
  - … knowledge of the general characteristics of the plaintext
  - … sample plaintext-ciphertext pairs

Attack Type	Information Known by Attacker
Ciphertext only	•Encryption algorithm •Ciphertext to be decoded
Known plaintext	•Encryption algorithm •Ciphertext to be decoded •One or more plaintext-ciphertext pairs formed with the secret key
Chosen plaintext	•Encryption algorithm •Ciphertext to be decoded •Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen ciphertext	•Encryption algorithm •Ciphertext to be decoded •Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen text	•Encryption algorithm •Ciphertext to be decoded •Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key •Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

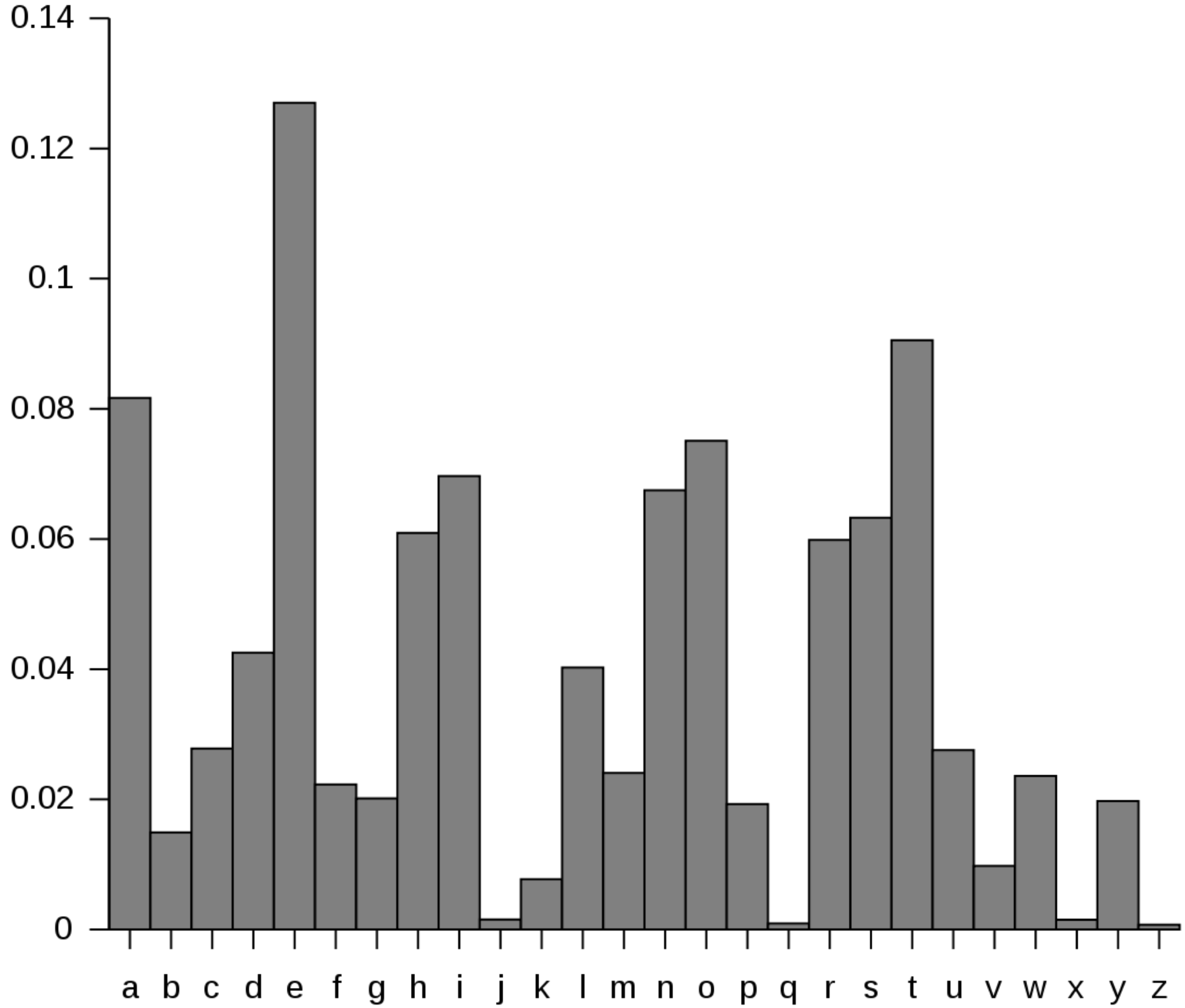
Attacking the Caesar Cipher

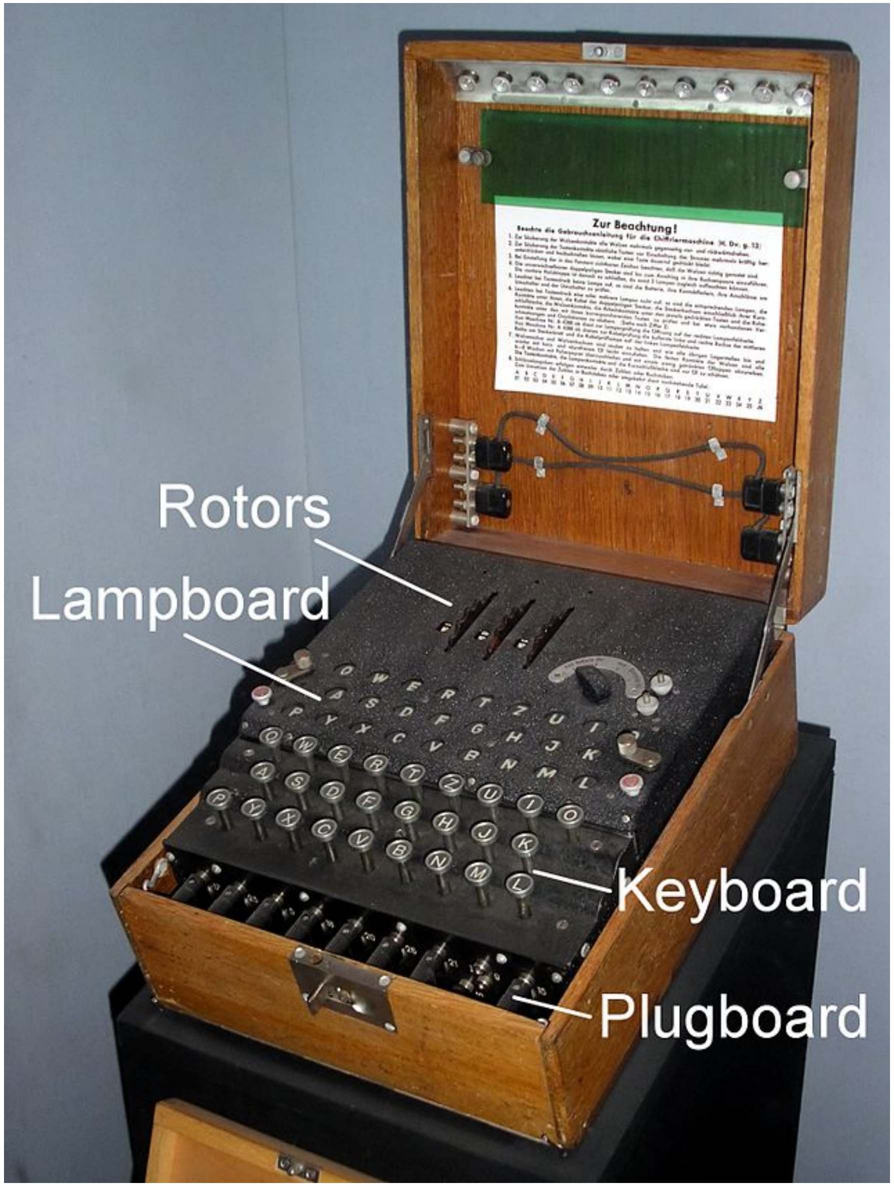


Shift by 3 and substitute

```
Plaintext:
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Ciphertext:
QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD
```

English Letter Frequency




Rotors
Lampboard
Keyboard
Plugboard

Modes of Operation

Direct use of block ciphers is not very useful

- Attackers can build a “code book” of plaintext/ciphertext equivalents
- Message-length needs to be multiple of cipher block size

Solution! Modes of operation

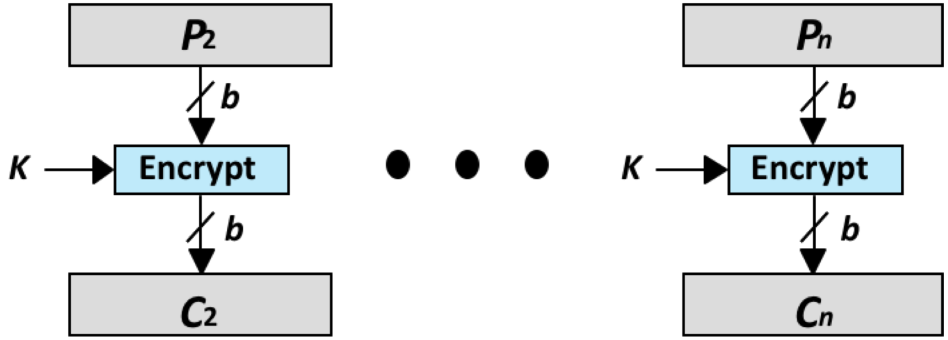
- Five standard modes

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

ECB Mode

In electronic codebook (ECB) mode each block of plaintext is encrypted using the same key

- Easy to parallelize



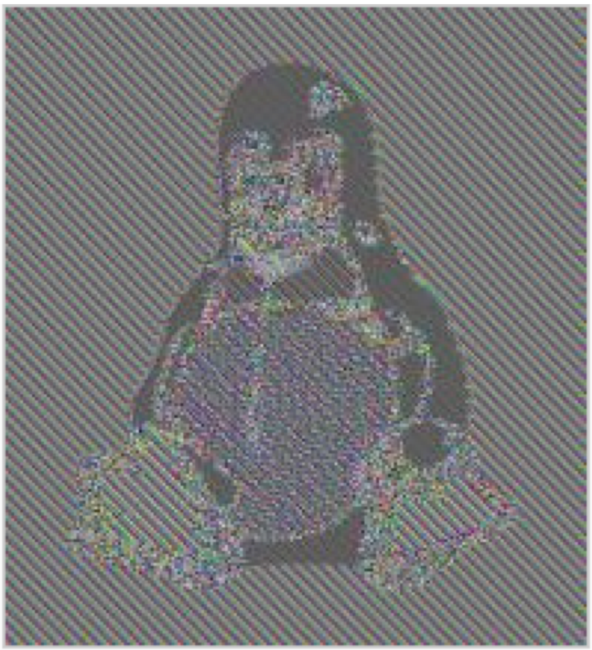
Problems

- Cryptanalysts may be able to exploit regularities in the plaintext (e.g., if $p_i == p_j$ then $c_i == c_j$)
- Data patterns may remain visible

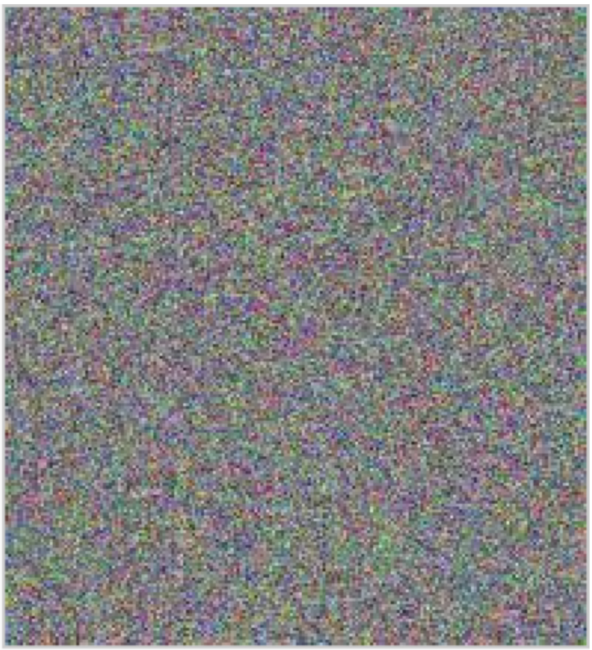
ECB Mode

ECB mode is not recommended

Original image



Encrypted using ECB mode

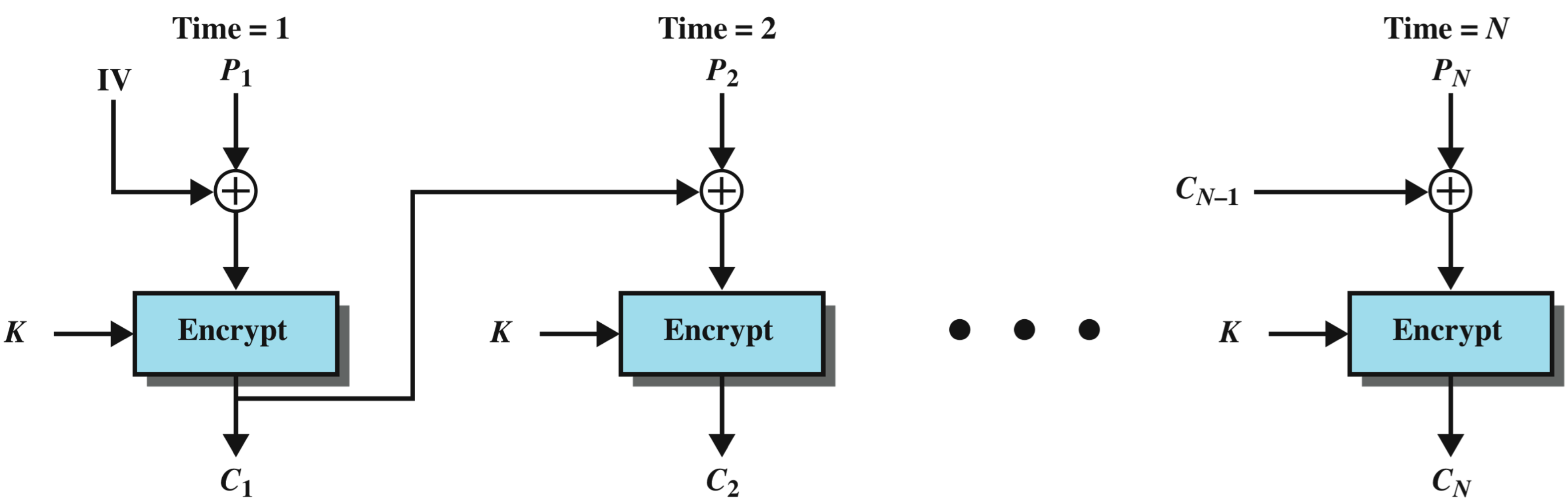


Modes other than ECB result in pseudo-randomness

CBC Mode

In Cipher Block Chaining mode the input is the XOR of the current plaintext block and the preceding ciphertext block

- **Initialization vector (IV)**
  - Must be random and must not be reused
- Not parallelizable

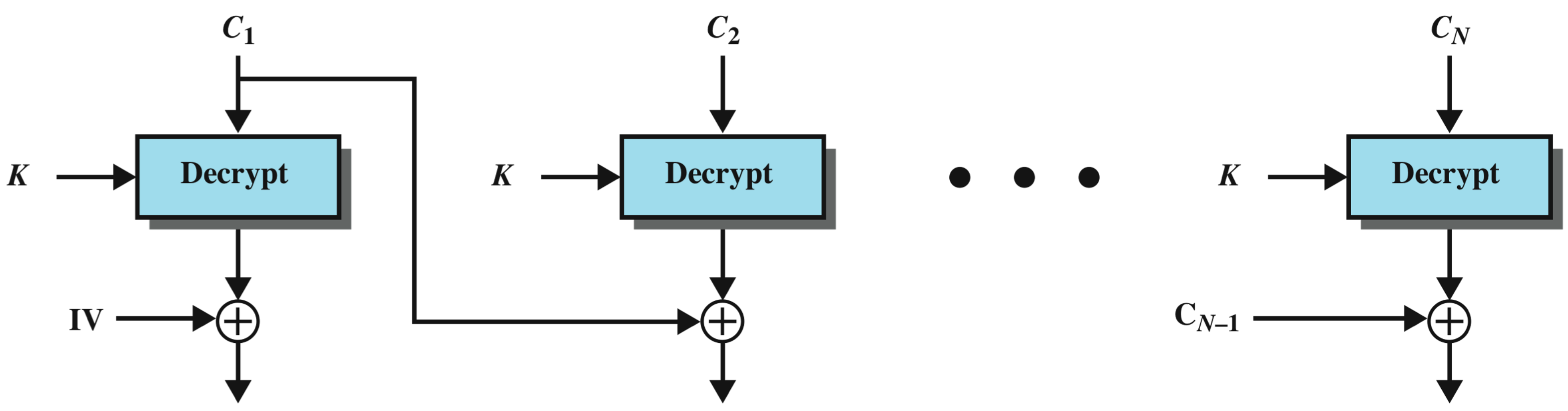


CBC Mode

During decryption the same IV must be used

- Can be transmitted with the message

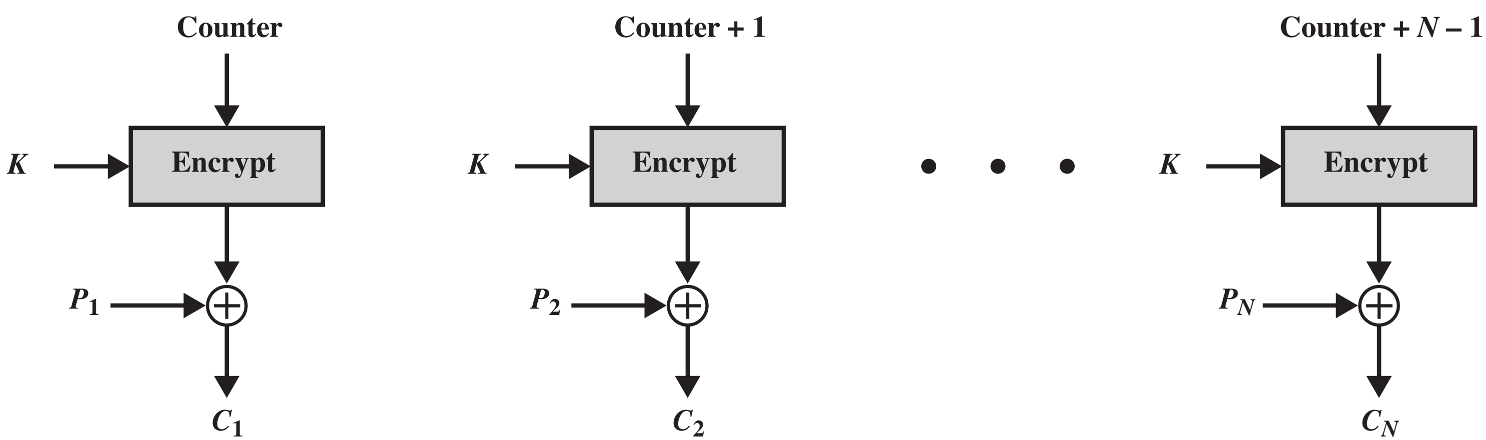
An error in a transmitted block also affects the following block but not subsequent ones



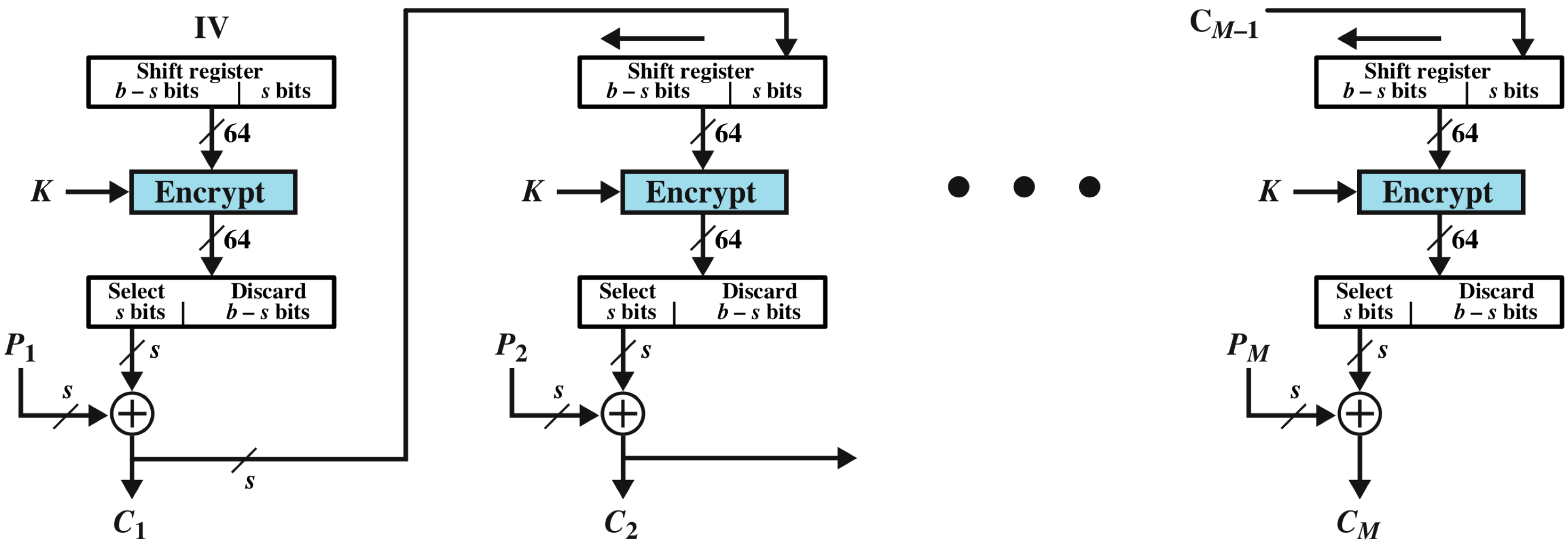
CTR Mode

Counter mode can be used to turn any blocking cipher to a stream cipher

- The counter is a combination of an integer (0..N-1) with an nonce (IV)
- Parallelizable!



Cipher Feedback (CFB)



(a) Encryption

Public-key Encryption

Public-Key Encryption

Publicly proposed by Diffie and Hellman in 1976

Based on mathematical functions

- …on the practical difficulty of factoring the product of two large prime numbers

Asymmetric

- Uses two separate keys a public and a private key
- Public key is made public for others to use

Multiple algorithms with different uses

- Establish a shared secret key
- Encrypt a message
- Digital signatures

Requirements for Public-Key Cryptosystems

Computationally easy …

- … to create key pairs
- … for sender knowing public key to encrypt messages
- … for receiver knowing private key to decrypt ciphertext

Computationally infeasible …

- … for opponent to determine private key from public key
- … for opponent to otherwise recover original message

Useful if either key can be used for each role

Symmetric vs Asymmetric

Which one is best?

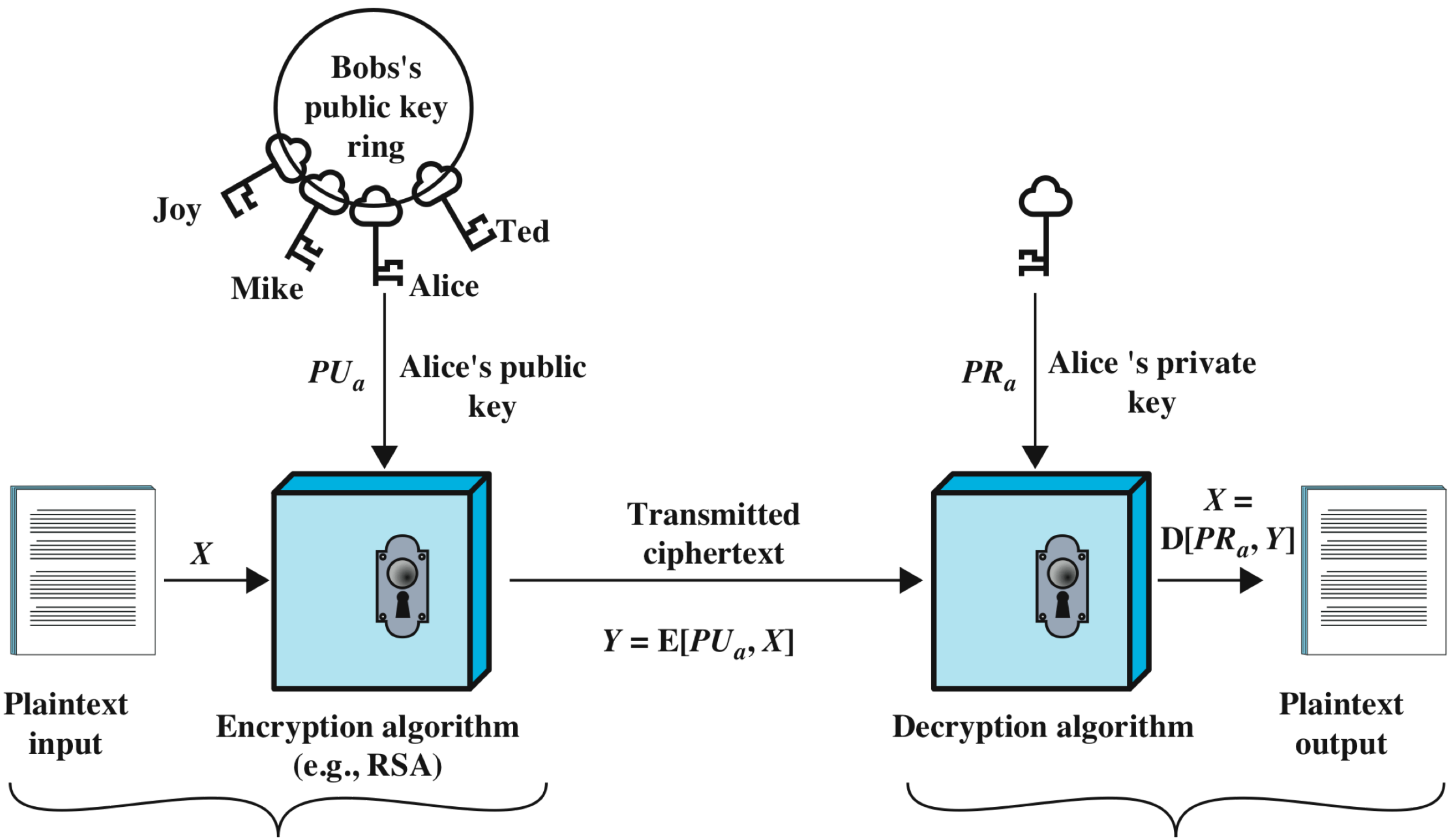
The strength of public-key cryptography depends more heavily on the length of the key

Intrinsically both offer similar guarantees against cryptanalysis

Public-key encryption is usually slower

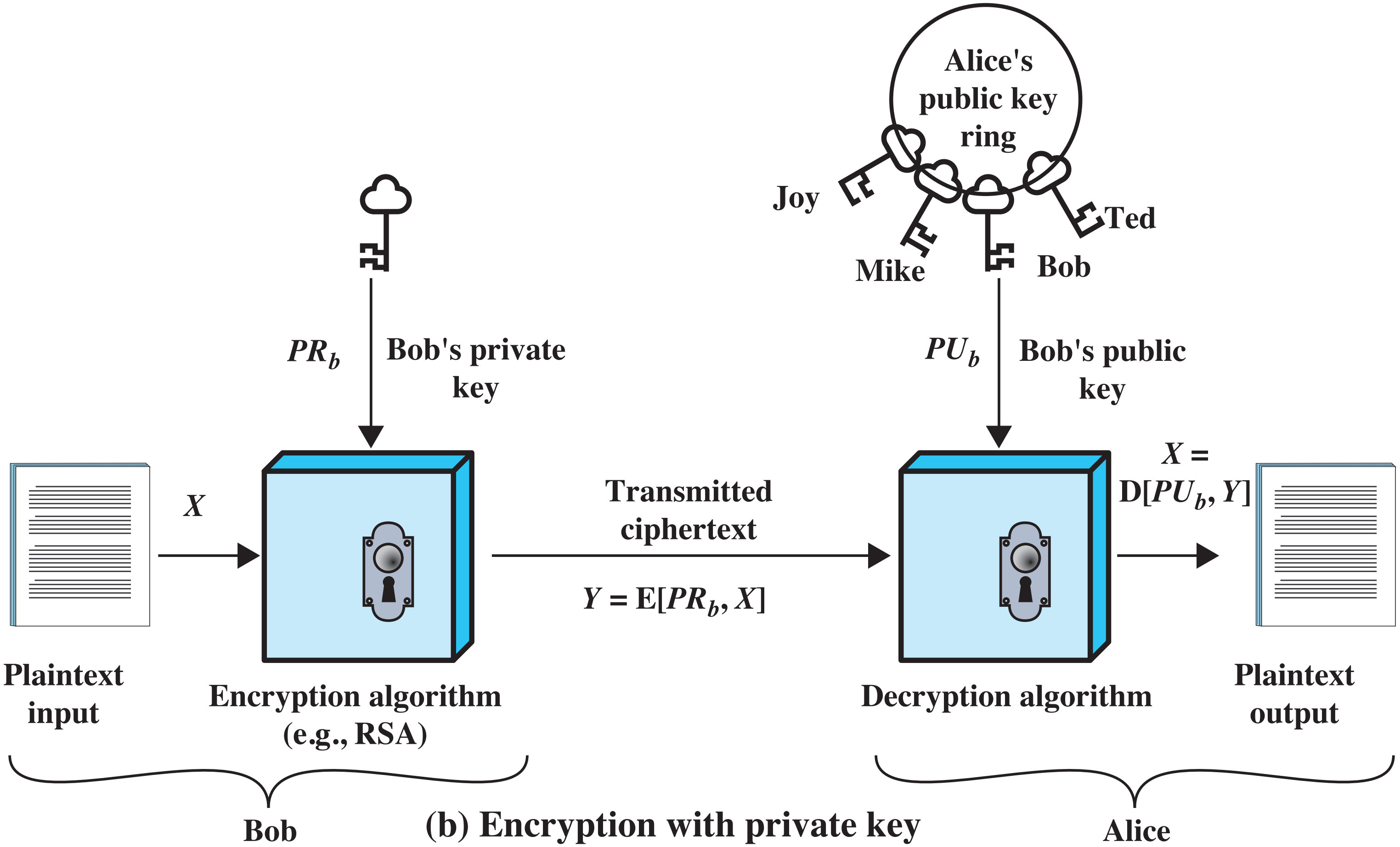
A shared key must be kept secret, similarly to the private key, but unlike the public key

Encryption with Public Key



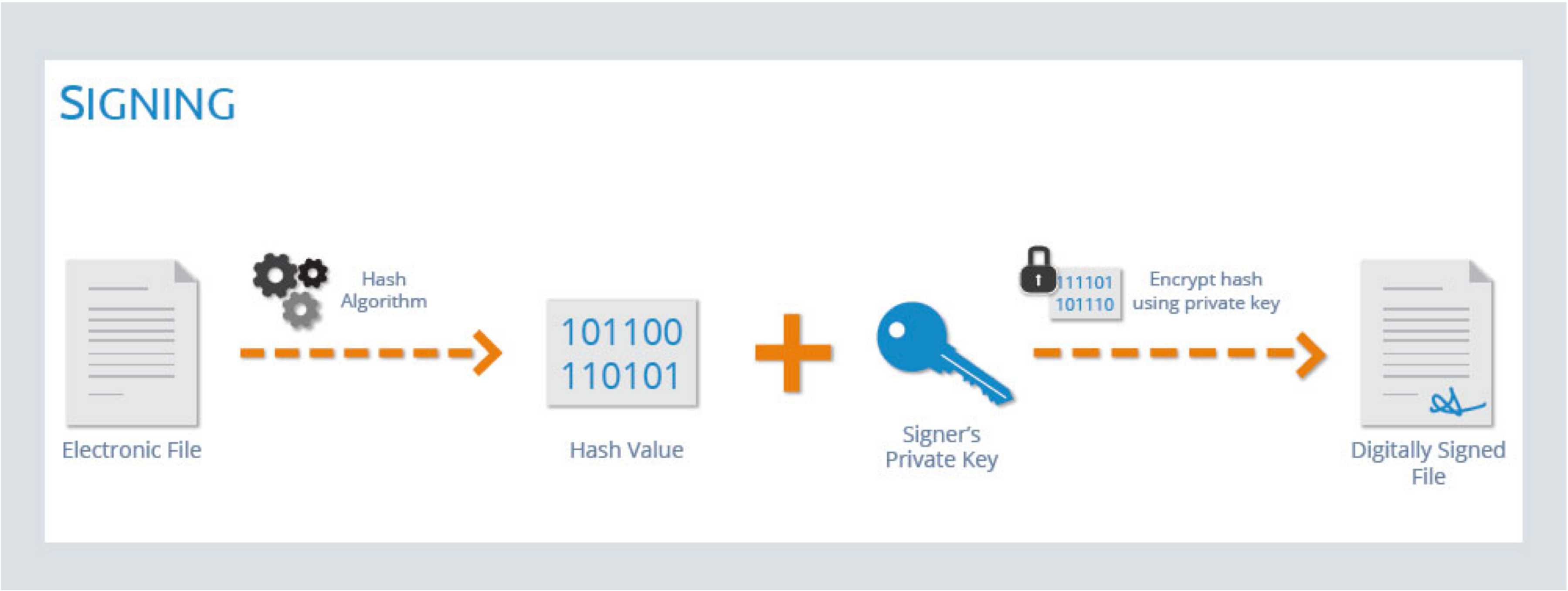
(a) Encryption with public key

Encryption with Private Key



(b) Encryption with private key

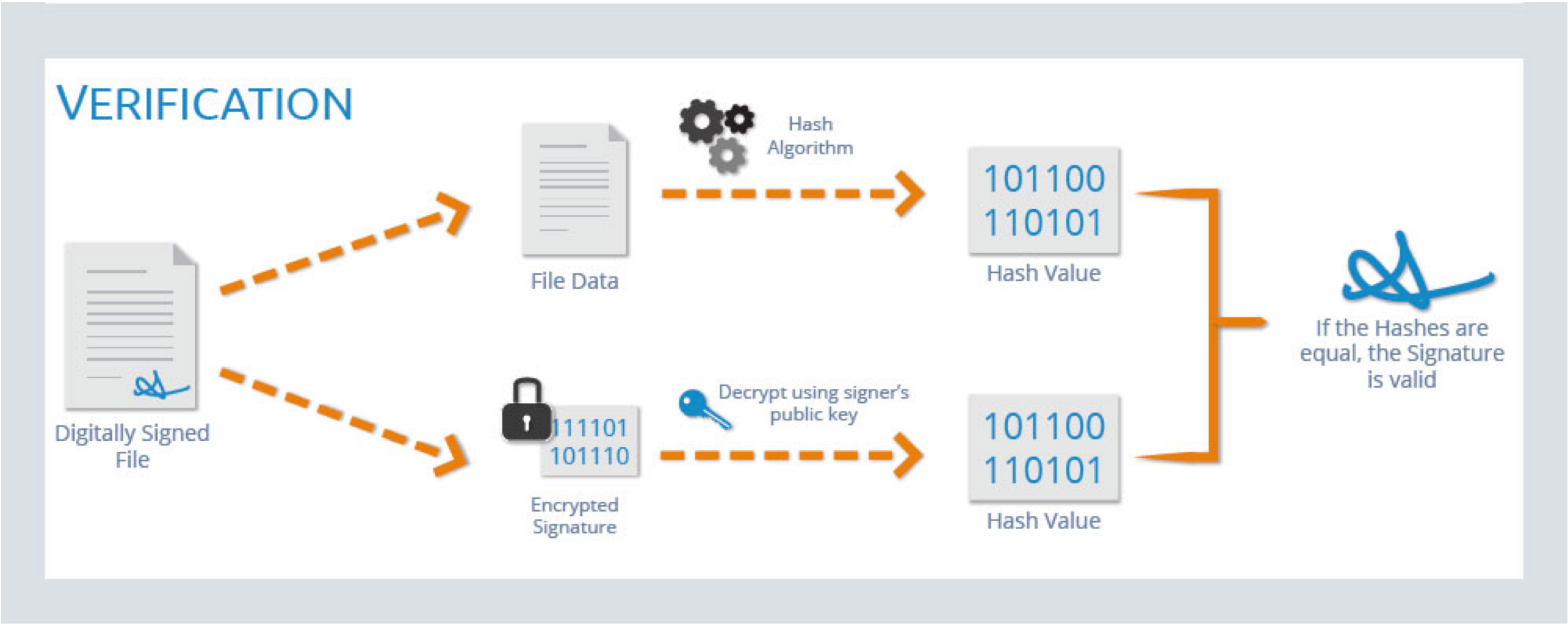
Digital Signing



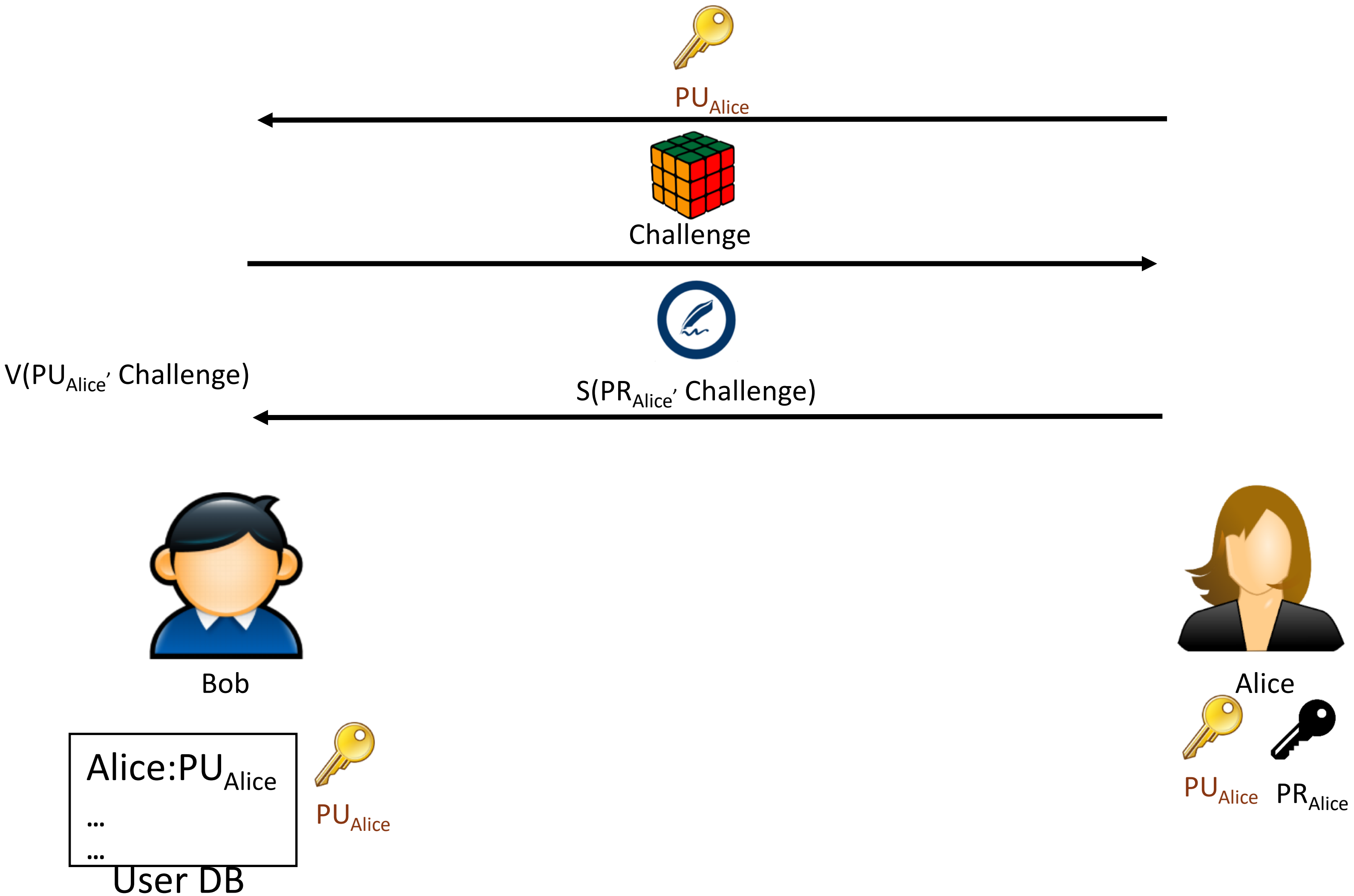
Digital Signing

Verify …

- … the author of data
- … the integrity of data



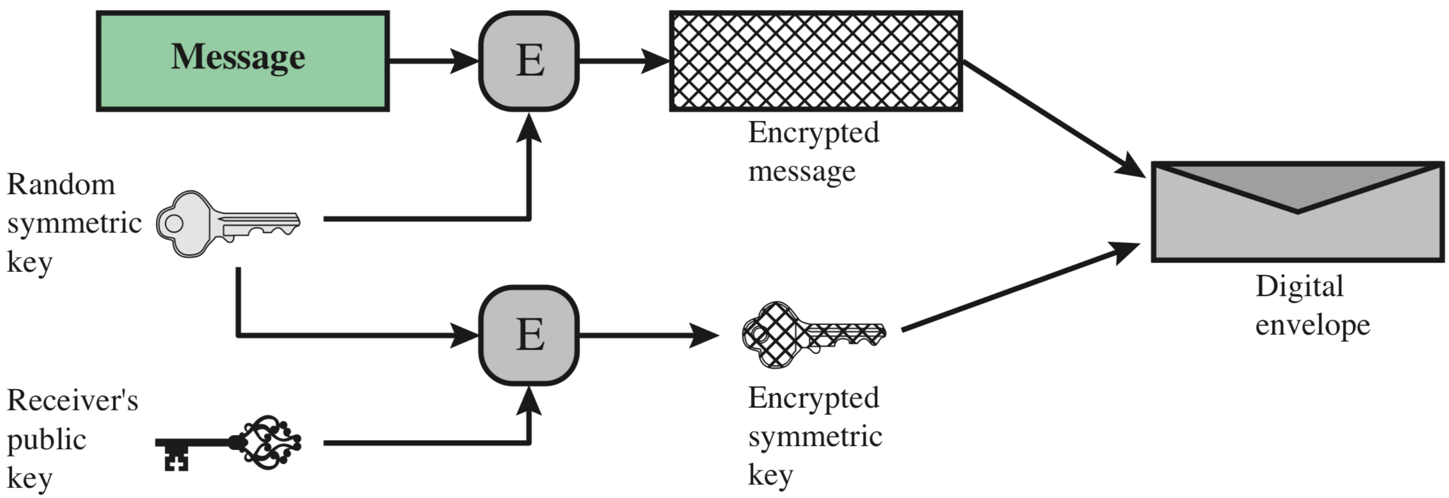
Authentication with Digital Signatures



Digital Envelopes

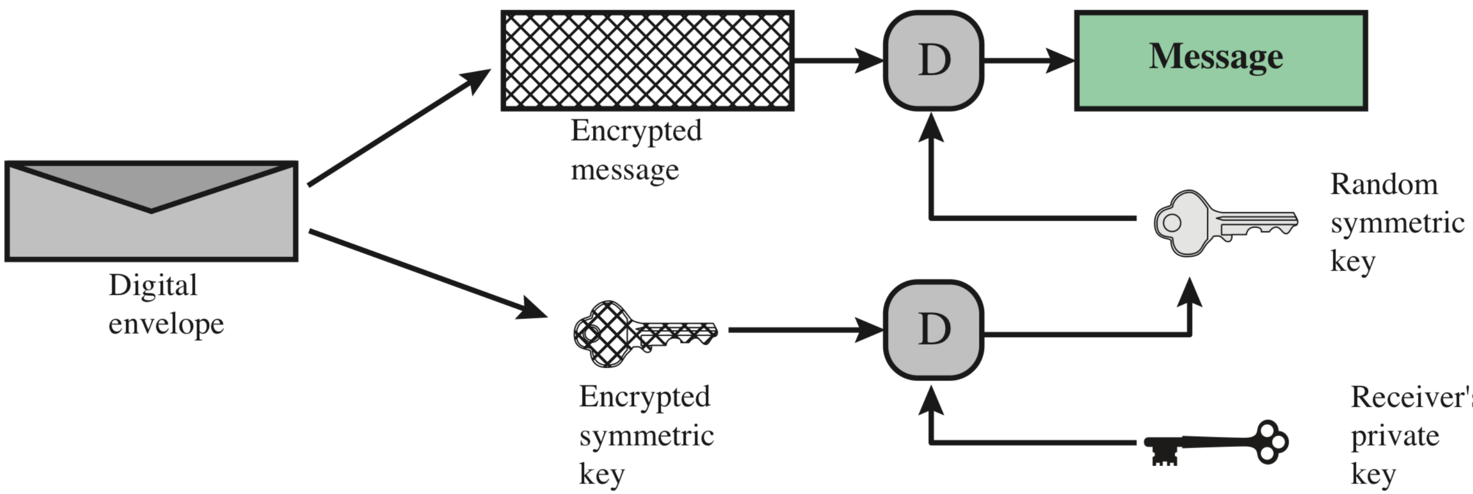
Use PK cryptography for encrypting a randomly generated symmetric key, which is used to encrypt a (large) message

- PK is only used to encrypt the key



Digital Envelopes

Opening an envelope



PK Encryption Algorithms

Diffie-Hellman key exchange algorithm

- Enables two users to securely reach agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages
- Limited to the exchange of the keys

RSA (Rivest, Shamir, Adleman)

- Developed in 1977
- Most widely accepted and implemented approach to public-key encryption

Elliptic curve cryptography (ECC)

- Security like RSA, but with much smaller keys

Comparison

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

RSA Security

Based on the assumption that factoring numbers is hard

Variable key length

- Largest, publicly known, factored RSA number is 768 bits
- It is generally believed that 1024-bit keys may have already been broken or will soon be
- 2048-bit keys are recommended as the minimum

Part of the Public Key Cryptography Standards (PKCS)

In practice used with digital envelopes

RSA Security

Brute force

- Trying all possible private keys
- Defense is to use a large key space, however this slows speed of execution

Mathematical attacks

- Several approaches, all equivalent in effort to factoring the product of two primes

Timing attacks

- Depend on the running time of the decryption algorithm
- Comes from a completely unexpected direction and is a ciphertext-only attack
- Countermeasures: constant exponentiation time, random delay, blinding

Chosen ciphertext attacks

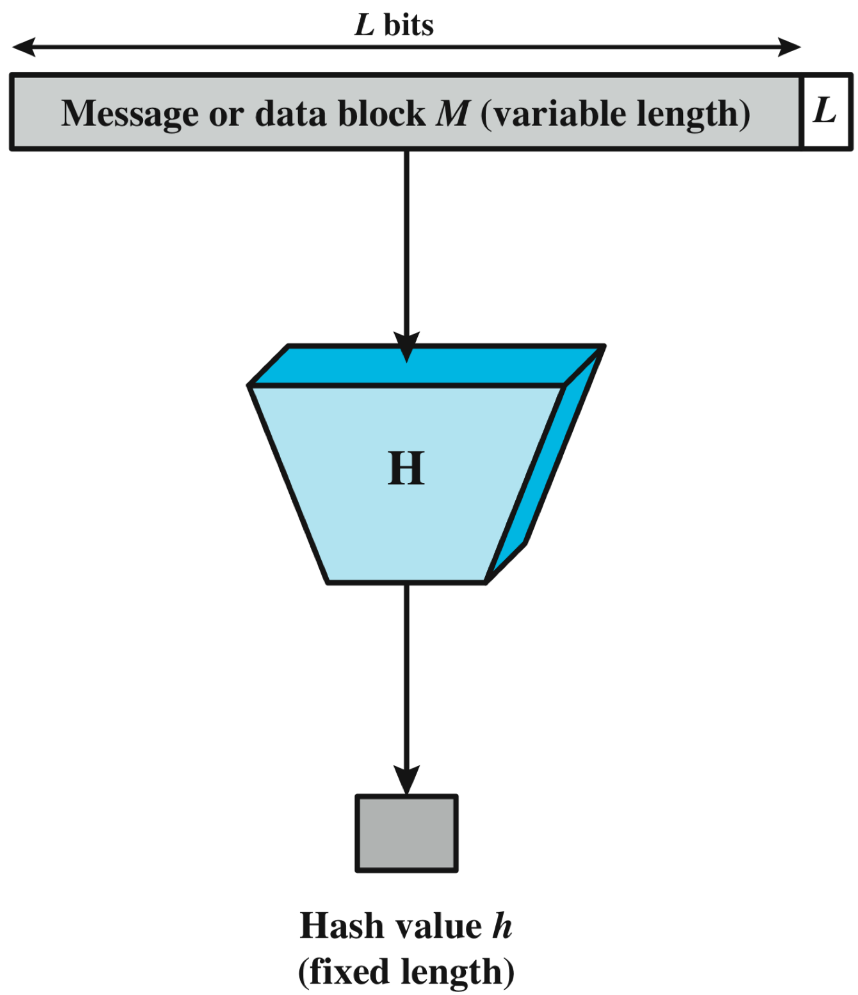
- Attack exploits properties of the RSA algorithm

Cryptographic Key Recommendation

Year	Symmetric	Factoring (modulus) (1)	(2)	Discrete Logarithm Key	Group	Elliptic Curve	Hash
2015	82	1613	1248	145	1613	154	163
2016	83	1664	1312	146	1664	155	165
2017	**83**	**1717**	**1344**	**147**	**1717**	**157**	**166**
2018	84	1771	1376	149	1771	158	168
2019	85	1825	1440	150	1825	160	169

https://www.keylength.com/en/1/

Hashing and Message Authentication Codes


L bits
Message or data block M (variable length)
L
H
Hash value h
(fixed length)

Hash Function Properties

Can be applied to a block of data of any size

Produces a fixed-length output

Security properties:

- One-way or pre-image resistant: computationally infeasible to find x such that H(x) = h
- Given x and H(X), it is computationally infeasible to find y ≠ x such that H(y) = H(x)
- Collision resistant or strong collision resistance: computationally infeasible to find any pair (x,y) such that H(x) = H(y)

Simple Hash Function

Split input in blocks of n bits

$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$

	Bit 1	Bit 2	• • •	Bit n
Block 1	b_{11}	b_{21}		b_{n1}
Block 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
Block m	b_{1m}	b_{2m}		b_{nm}
Hash code	C_1	C_2		C_n

Secure Hash Algorithm (SHA)

SHA was originally developed by NIST

- Published as FIPS 180 in 1993
- Revised in 1995 as SHA-1
- Produces 160-bit hash values

SHA-2 adds 3 additional versions of SHA

- SHA-256, SHA-384, SHA-512 with 256/384/512-bit hash values
- Same basic structure as SHA-1 but greater security

SHA Comparison

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

Notes: 1. All sizes are measured in bits.

2. Security refers to the fact that a birthday attack on a message digest of size *n* produces a collision with a work factor of approximately $2^{n/2}$.

SHA-3

SHA-1 considered insecure and has been phased out for SHA-2

SHA-2 shares same structure and mathematical operations as its predecessors and causes concern

Due to the time required to replace SHA-2 should it become vulnerable, NIST announced in 2007 a competition to produce SHA-3

SHA-3, a subset of the cryptographic primitive family **Keccak**

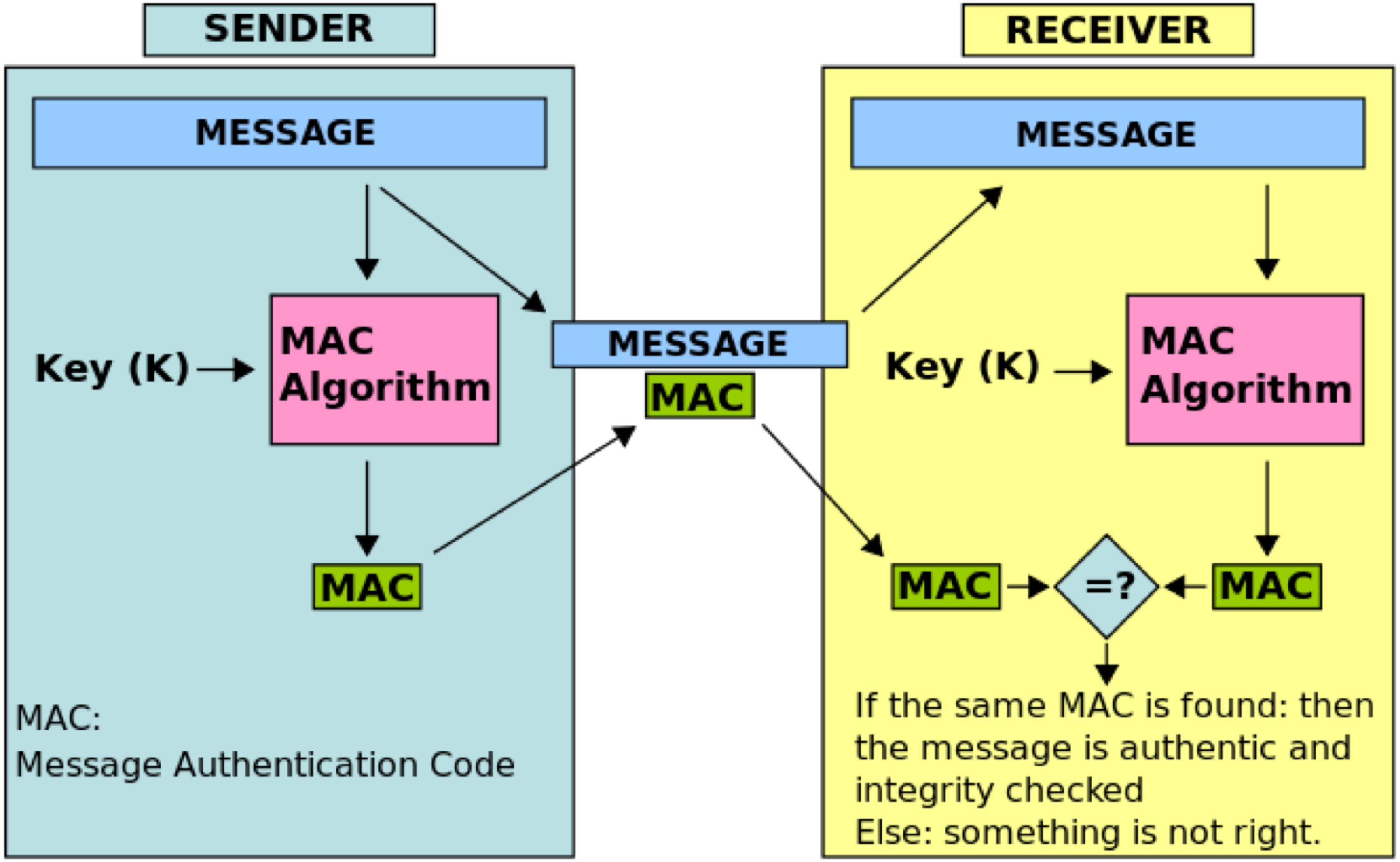
- Better security (resist attacks against SHA-2)
- Appropriate for fast implementation in hardware

Comparison from Wikipedia

Algorithm and variant		Output size (bits)	Block size (bits)	Max message size (bits)	Security (bits)	Example Performance (MiB/s)[12]
MD5 (as reference)		128	512	$2^{64} - 1$	<64 (collisions found)	335
SHA-0		160	512	$2^{64} - 1$	<80 (collisions found)	-
SHA-1		160	512	$2^{64} - 1$	<80 (theoretical attack[13] in 2^{61})	192
SHA-2	SHA-224 SHA-256	224 256	512	$2^{64} - 1$	112 128	139
	SHA-384 SHA-512 SHA-512/224 SHA-512/256	384 512 224 256	1024	$2^{128} - 1$	192 256 112 128	154
SHA-3	SHA3-224 SHA3-256 SHA3-384 SHA3-512 SHAKE128 SHAKE256	224 256 384 512 d (arbitrary) d (arbitrary)	1152 1088 832 576 1344 1088	∞	112 128 192 256 min(d/2, 128) min(d/2, 256)	

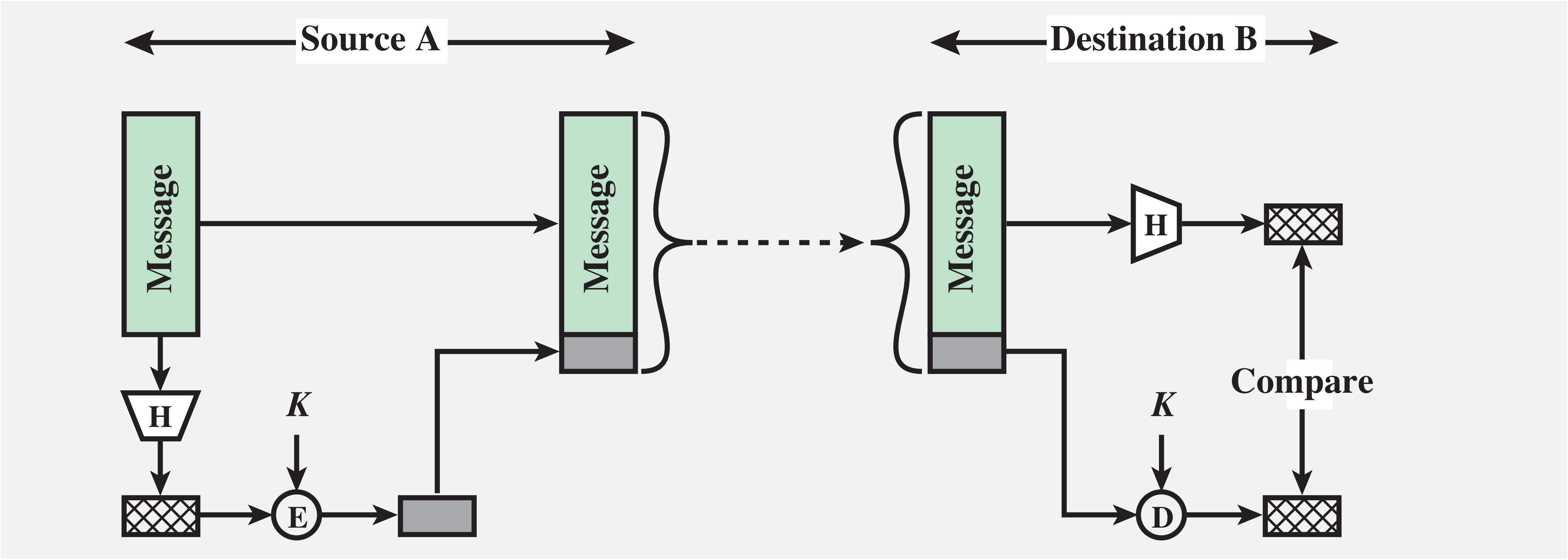
Message Authentication Code

Verify message integrity and authenticity



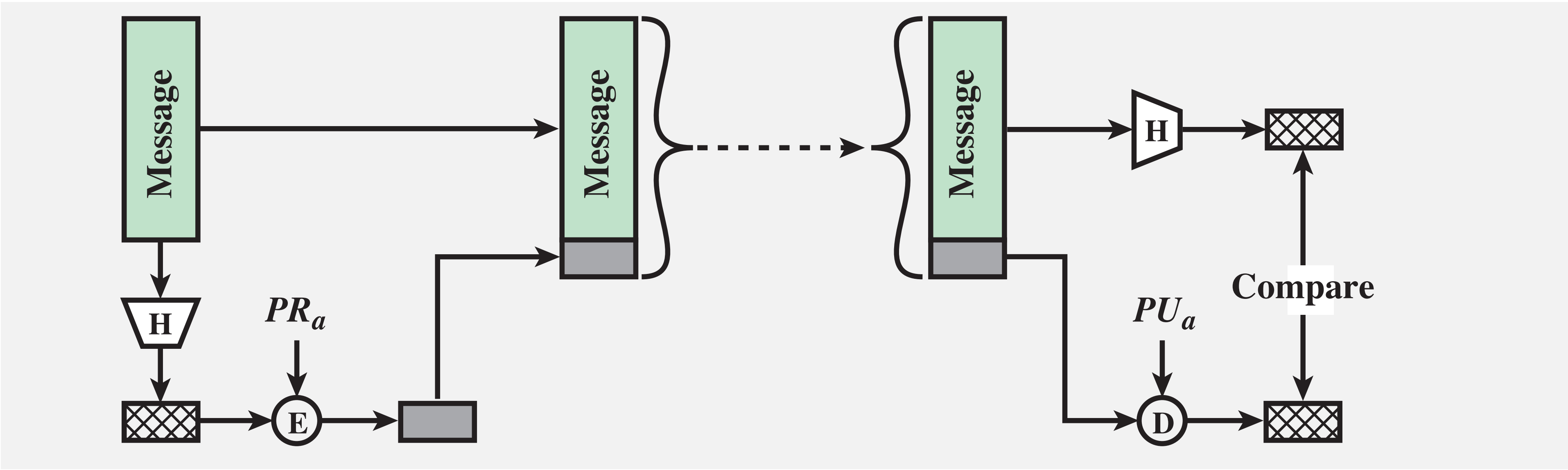
MAC with Symmetric Encryption

Encrypt hash of message using shared secret key, verify by decrypting with the same key



MAC with Public-Key Encryption

Encrypt hash of message using private key, verify using public key of sender



Digital Signatures

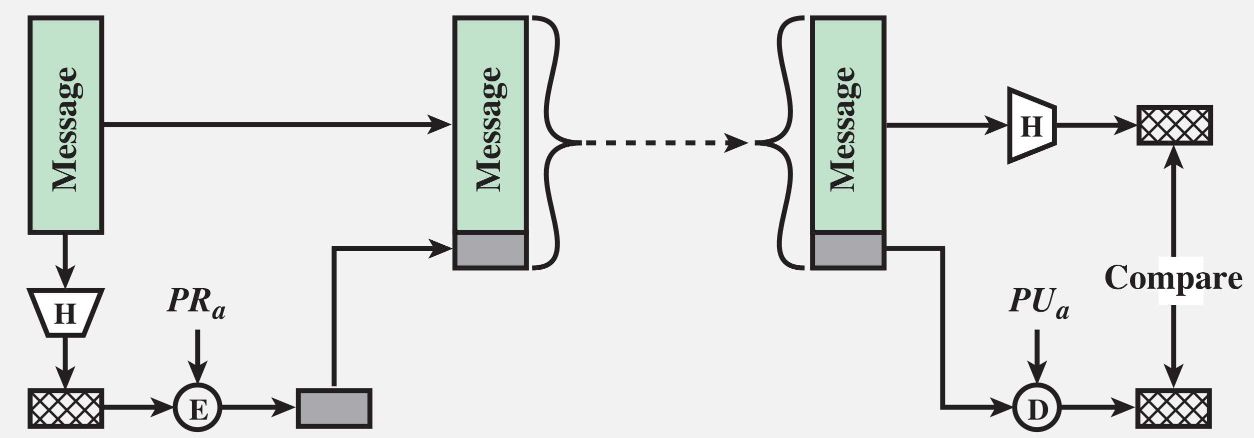
Similar to MAC using public-key cryptography

Used for authenticating both source and data integrity

Created by encrypting hash code with private key

Does not provide confidentiality

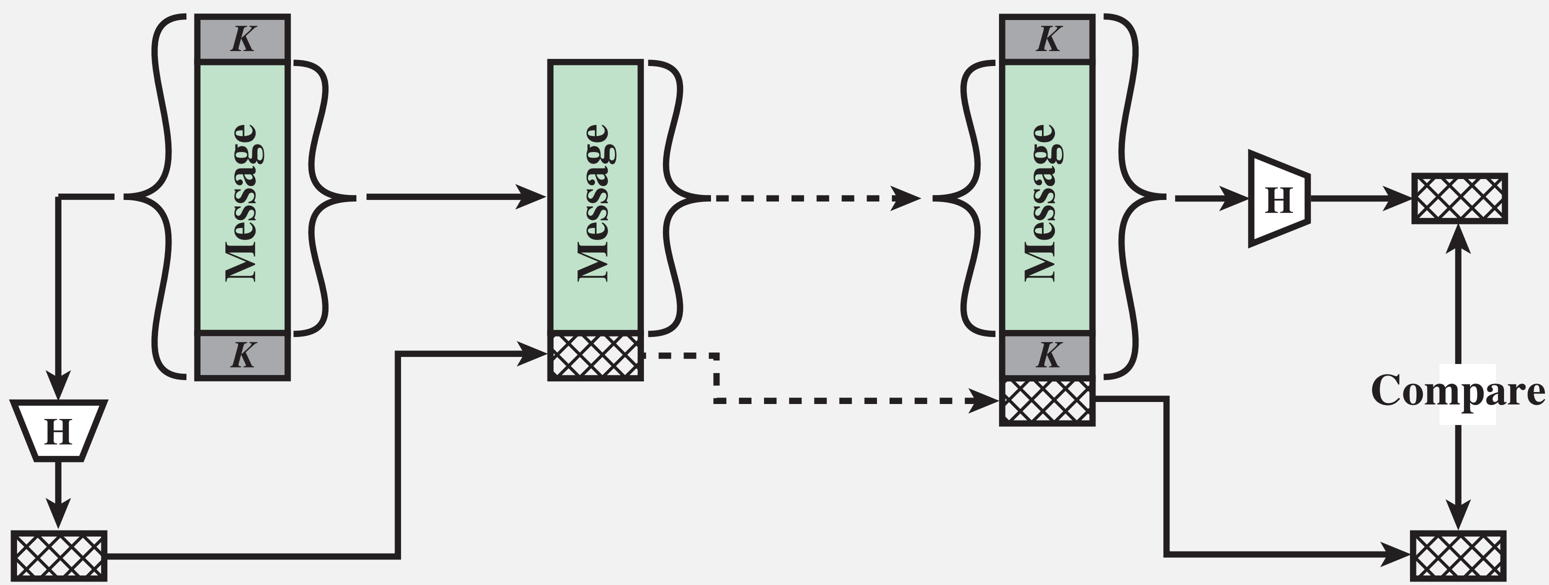
- Message is safe from alteration but not eavesdropping



(b) Using public-key encryption

MAC with Secret Value

Prefix and suffix message using nonce and hash the result, verify using the reverse



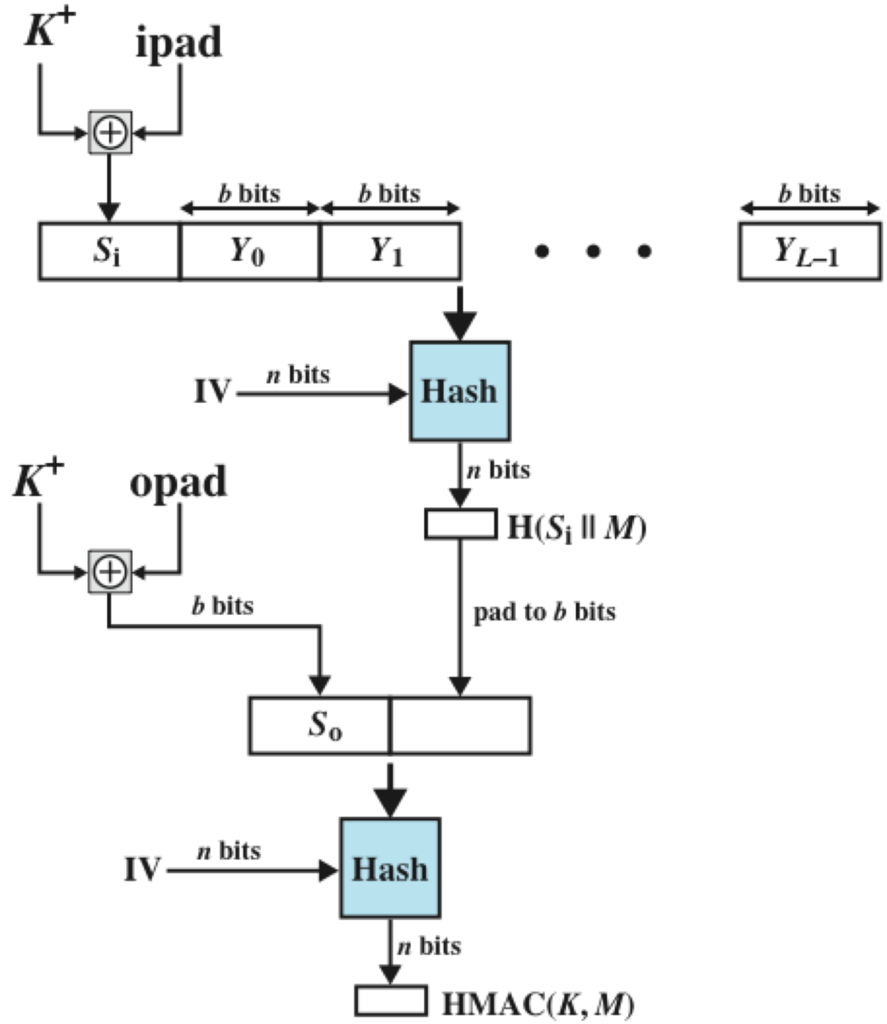
Hashed MAC (HMAC) Standard

A MAC using a secret key that enables the use of available hash functions without modifications

To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required

To use and handle keys in a simple way

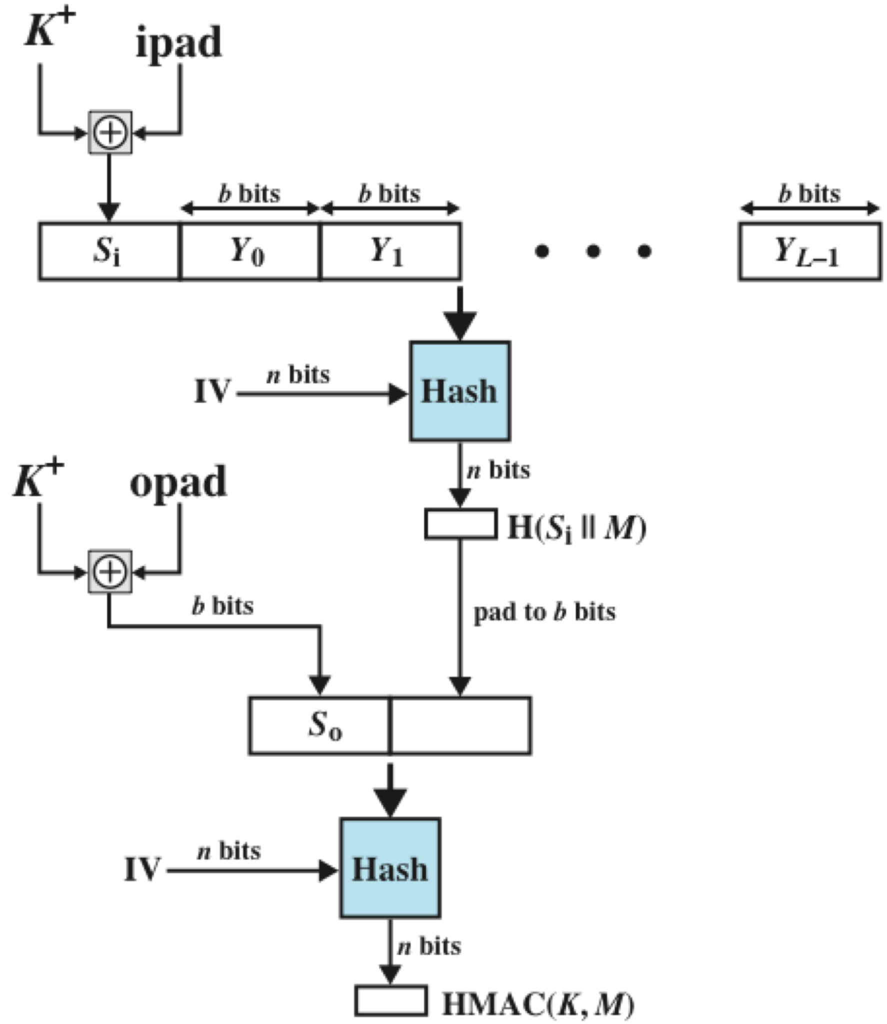
- **K+** is K padded with zeros on the left so that the result is b bits in length
- **ipad** is a pad value of 36 hex repeated to fill block
- **opad** is a pad value of 5C hex repeated to fill block
- **M** is the message input to HMAC (including any padding)
- **IV** Initialization vector (if hash function requires one)



$$HMAC(K,M) = Hash[(K^{+} \text{ XOR } opad) \,||\, Hash[(K^{+} \text{ XOR } ipad) \,||\, M)]]$$

- Note that the XOR with ipad results in flipping one-half of the bits of *K* .

- Similarly, the XOR with opad results in flipping one-half of the bits of *K , but a* different set of bits. In effect, by passing S_i *and* S_o *through the hash algorithm, we* have pseudorandomly generated two keys from *K* .



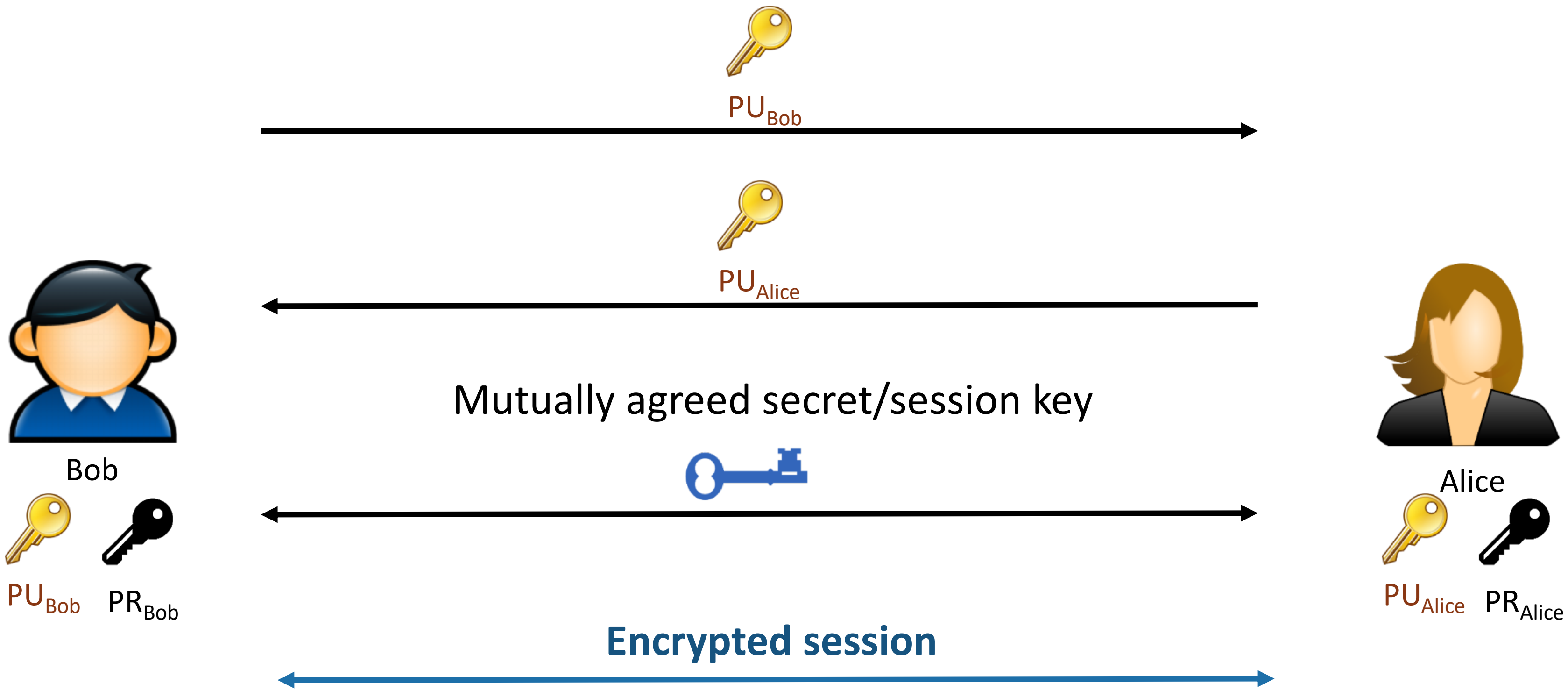
$$HMAC(K,M) = Hash[(K^+ \text{ XOR opad}) \,||\, Hash[(K^+ \text{ XOR ipad}) \,||\, M)]]$$

Hashes vs MACs vs Signatures

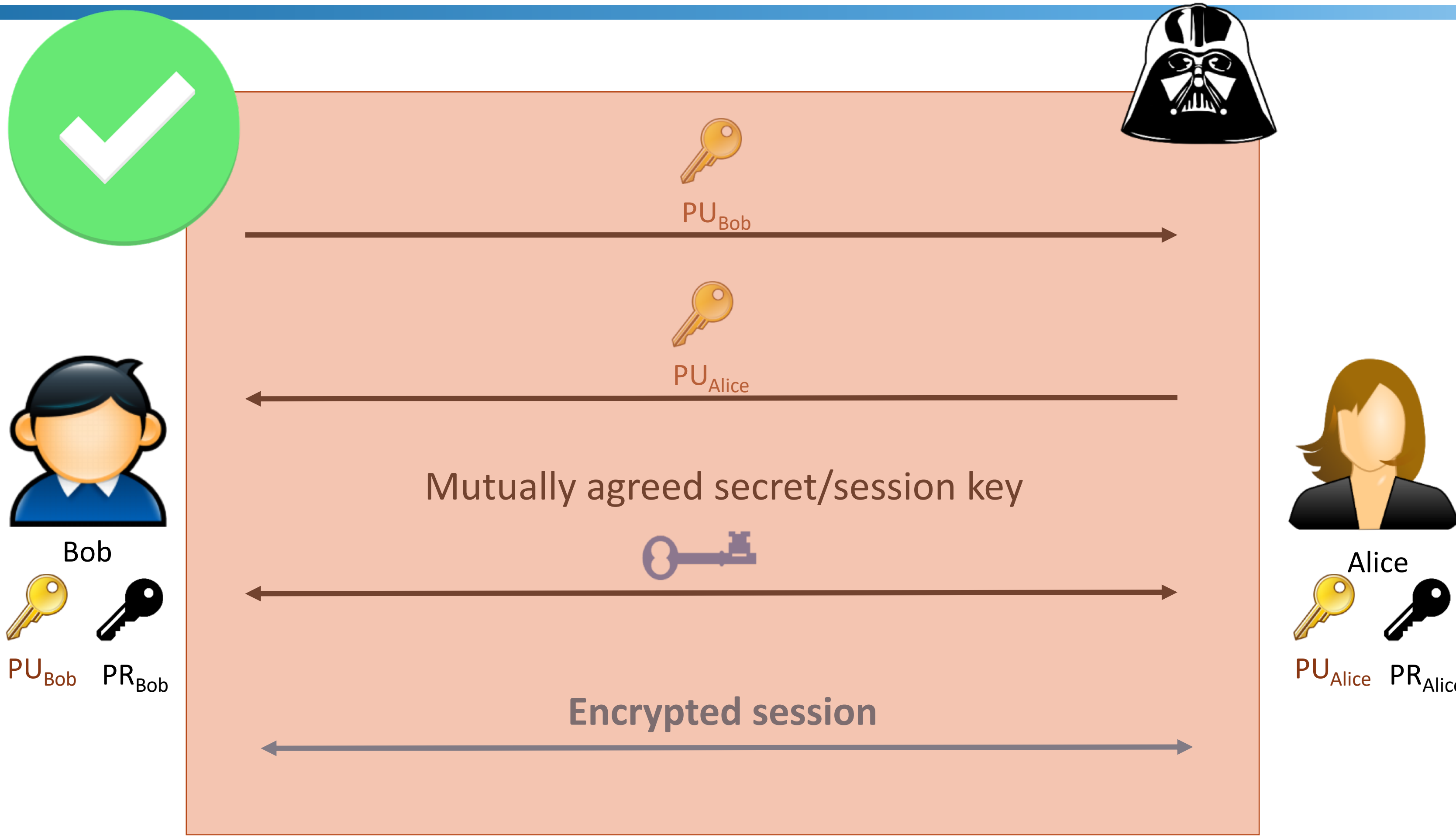
	Hash	MAC	Signature
Integrity	✔	✔	✔
Authentication		✔	✔
Non-repudiation			✔
Keys	None	Symmetric	Asymmetric

Private Connections in Practice

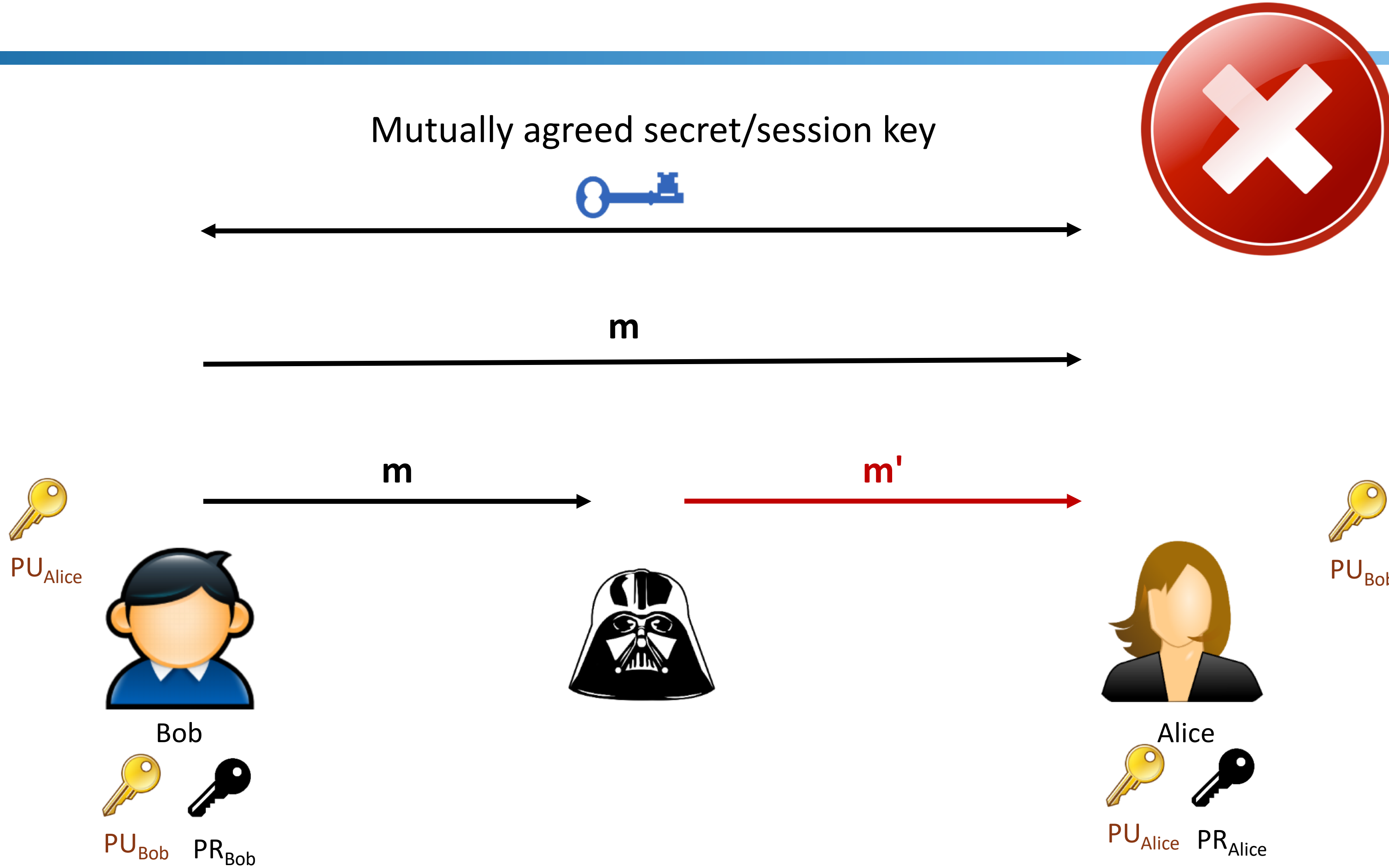
Encrypted Connections



Passive Attacker



Active Attacker



Encrypt and MAC

Encrypted data need to protected with MAC against active adversaries

MAC-and-Encrypt $\quad E(P) \,||\, M(P)$

- No integrity of the ciphertext

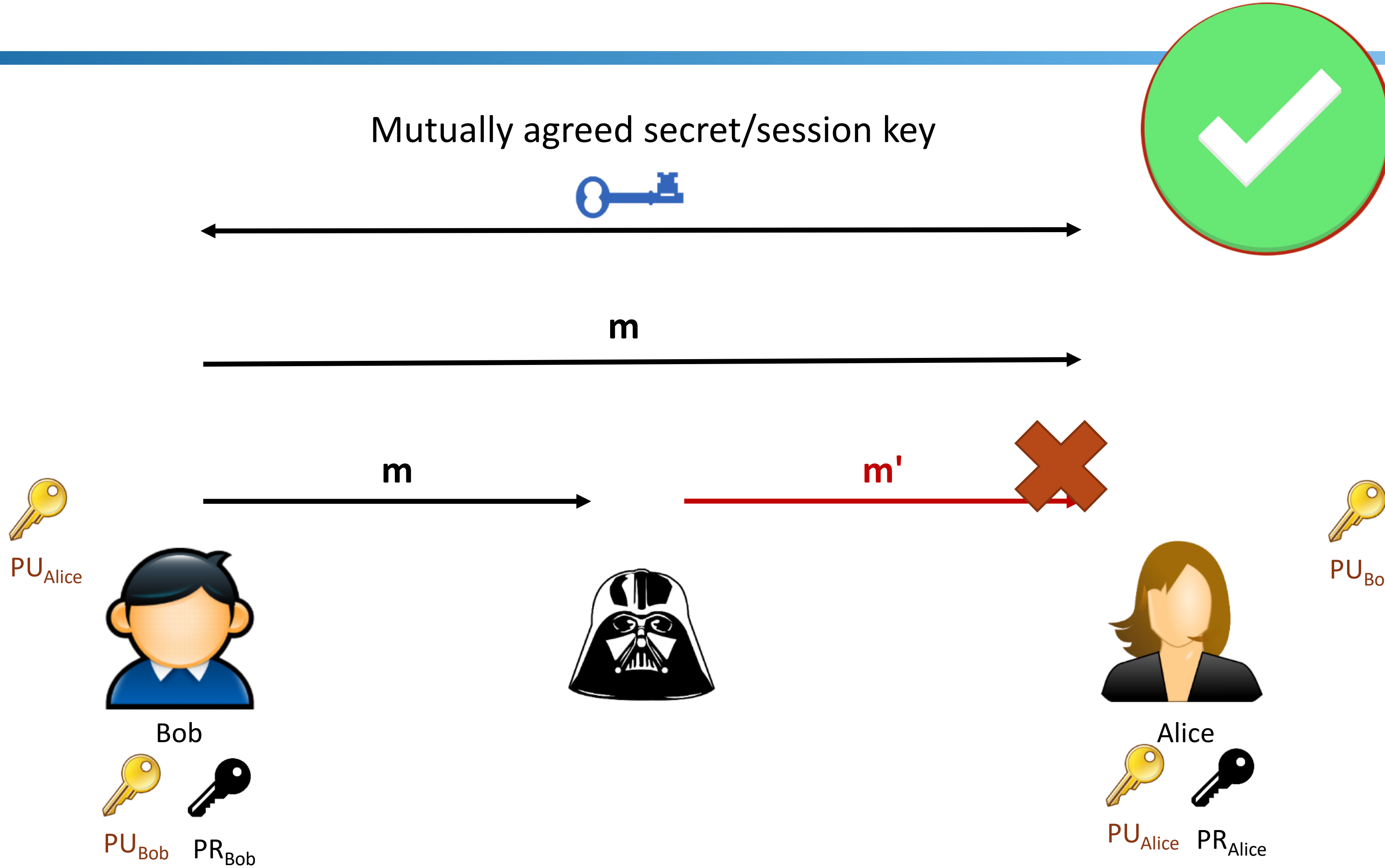
MAC-then-Encrypt $\quad E(P \,||\, M(P))$

- No integrity of the ciphertext

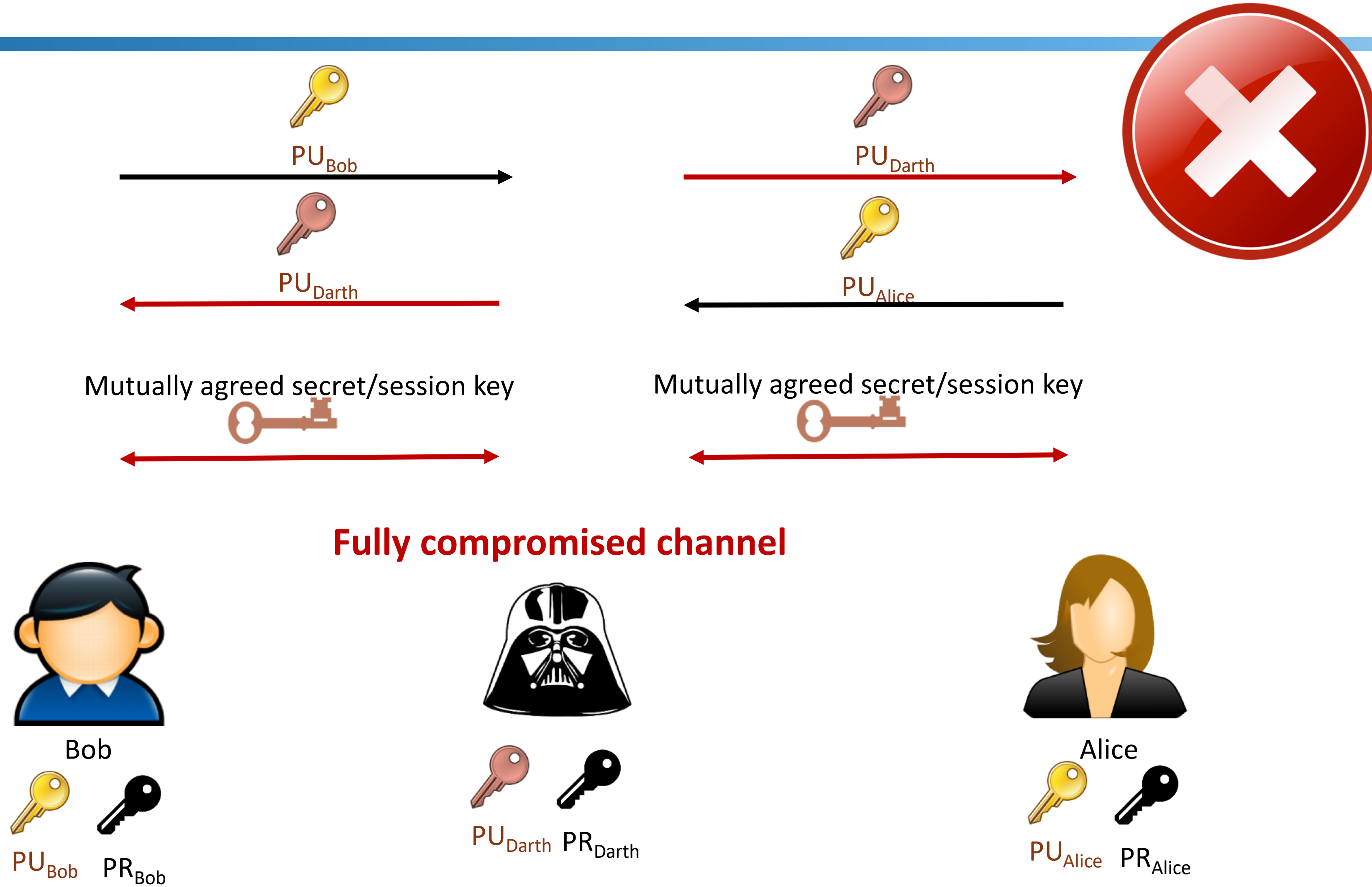
Encrypt-then-MAC $\quad E(P) \,||\, M(E(P))$

- **The right option**

Active Attacker



Man-in-the-middle (MITM)



Types of Adversaries/Attacks

Passive – does not affect system resources

- Can intercept messages but not modify

Active – attempt to alter system resources or affect their operation

- Can intercept, re-order, and alter messages

Authentication of Public-keys

Public-Key Authenticity

PK encryption requires that parties can establish the authenticity of public keys

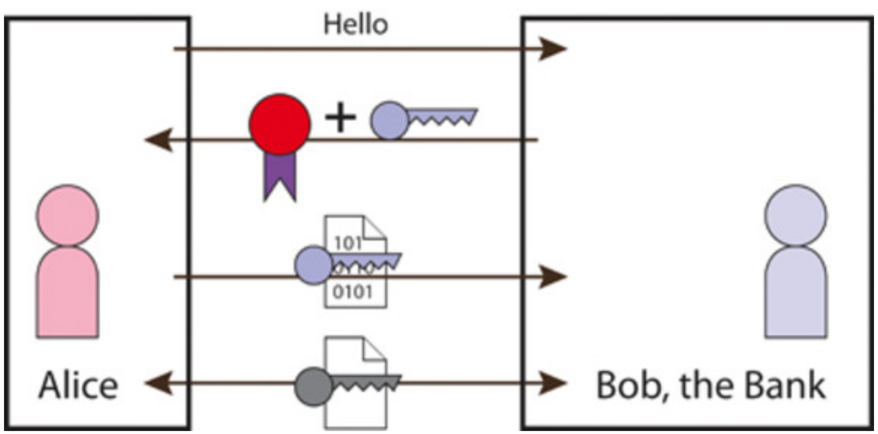
Some ways to accomplish this:

- Trust on first use (TOFU)
- Web of Trust
- **Public-key infrastructure (PKI)**

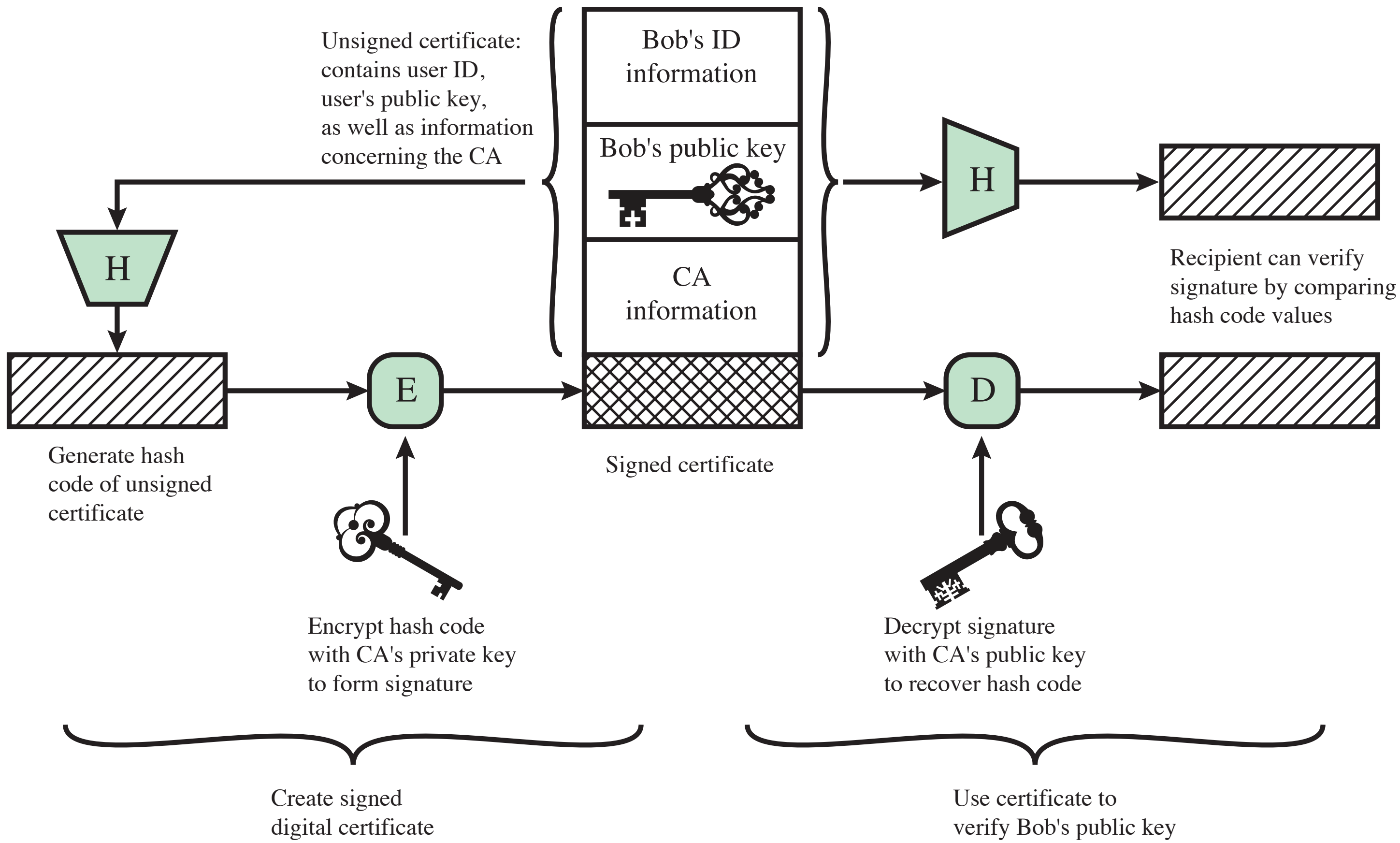
Certificates

Certificates are essentially signed public keys

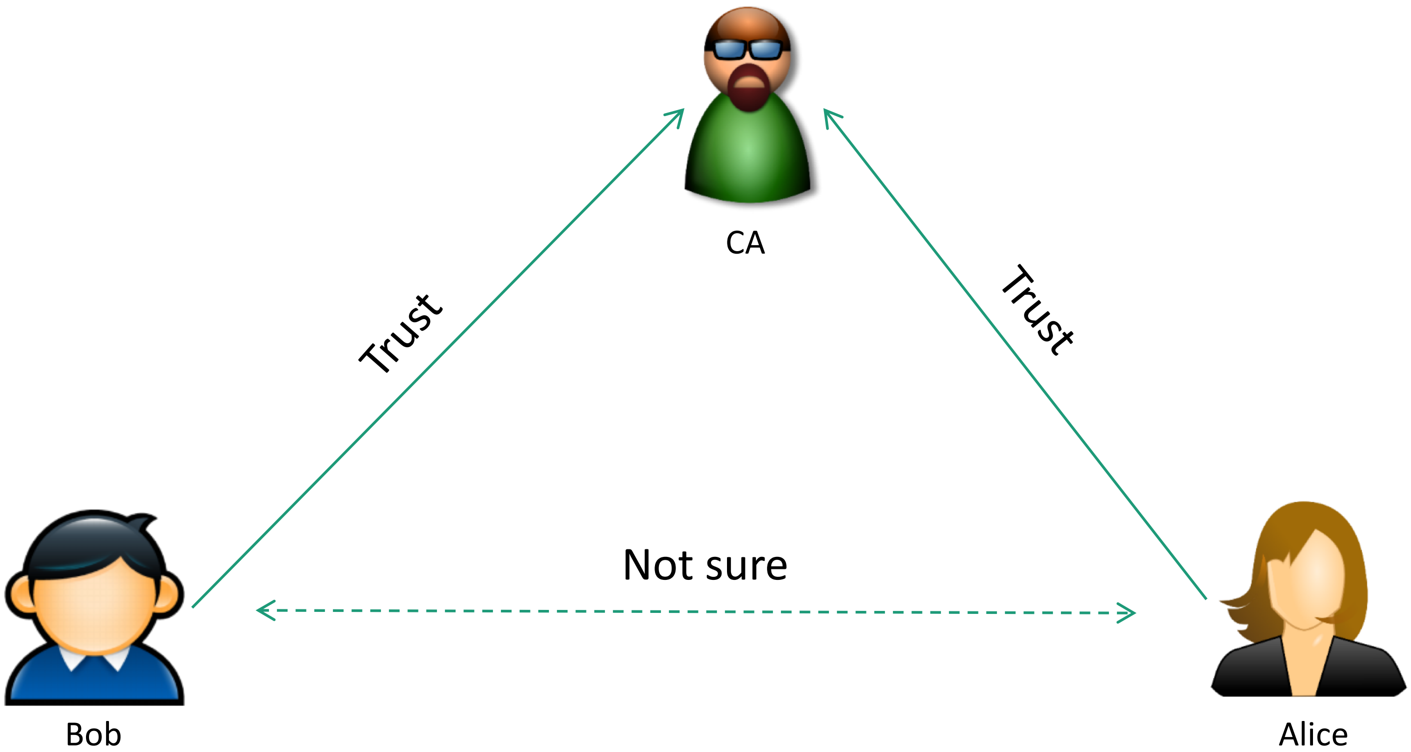
- Signed with the private key of a **certificate authority**

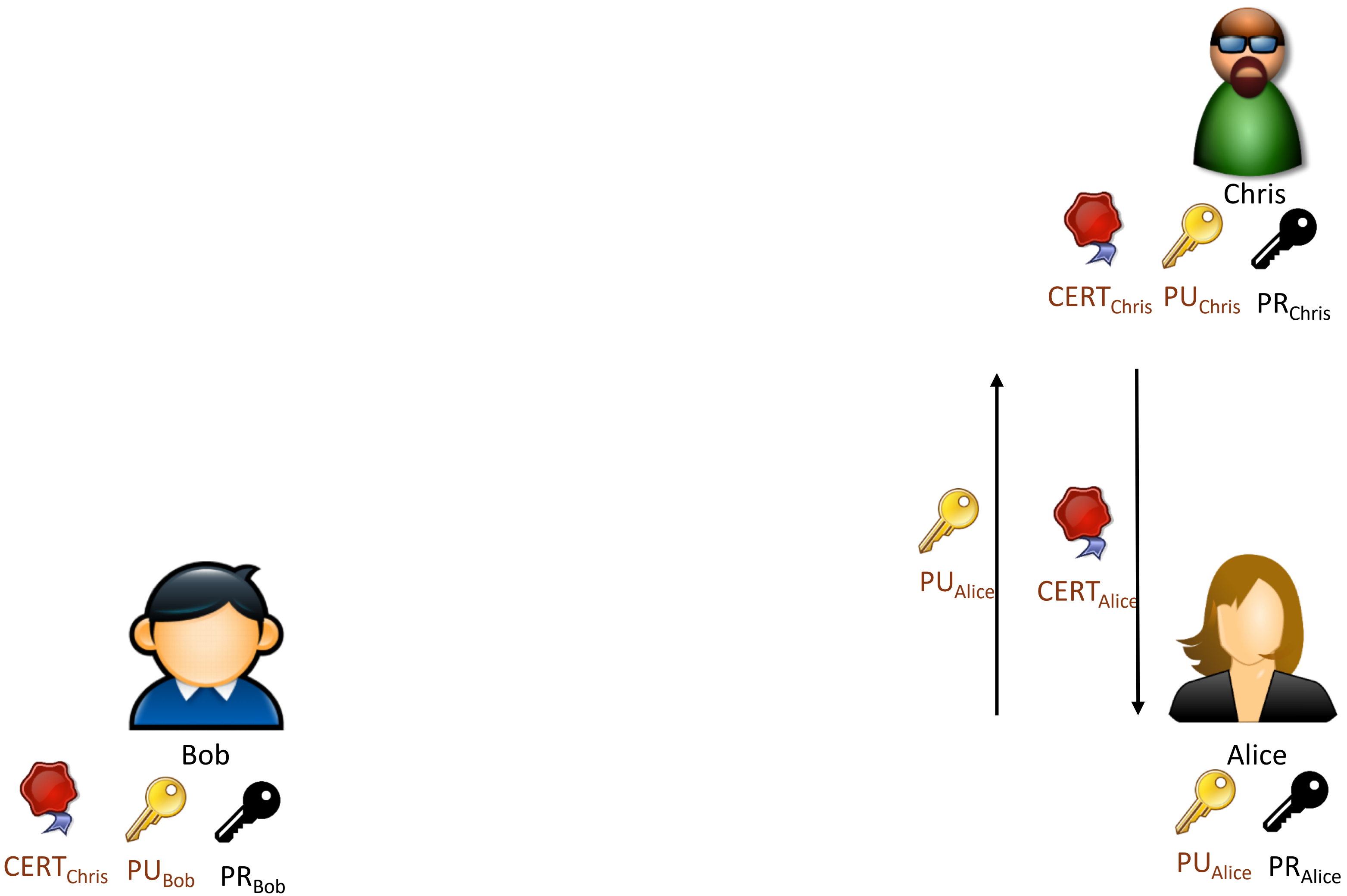


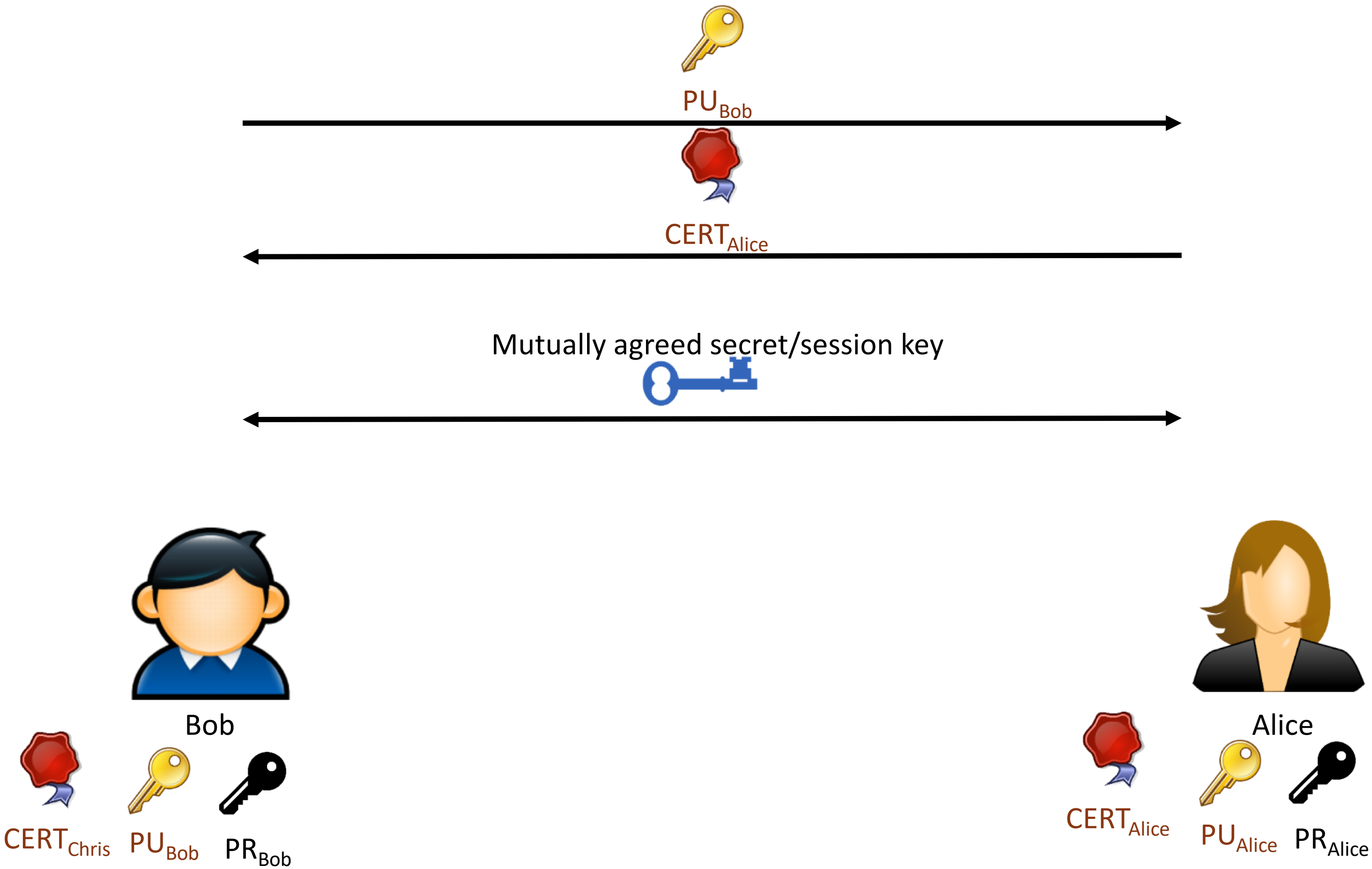
Certificates



Trusted Certificate Authorities




Chris
CERT_Chris
PU_Chris
PR_Chris
PU_Alice
CERT_Alice
Bob
CERT_Chris
PU_Bob
PR_Bob
Alice
PU_Alice
PR_Alice


PU_{Bob}
$CERT_{Alice}$
Mutually agreed secret/session key
Bob
$CERT_{Chris}$
PU_{Bob}
PR_{Bob}
Alice
$CERT_{Alice}$
PU_{Alice}
PR_{Alice}

Certificate Chains

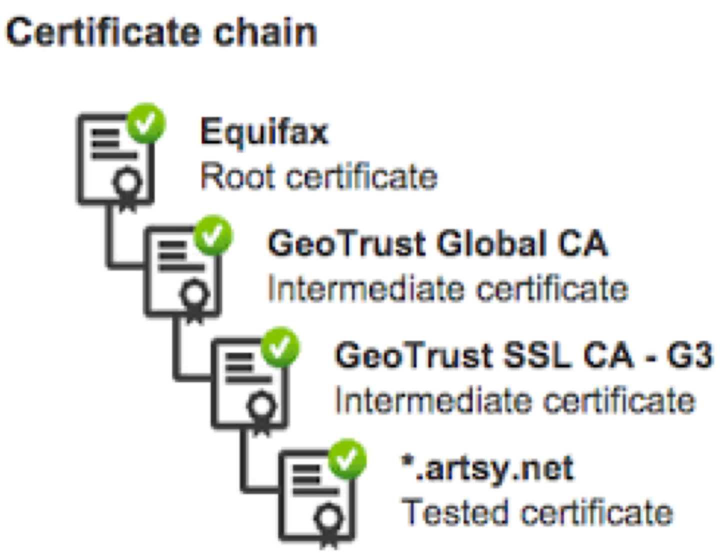
Trust anchors: Systems are preconfigured with a list of trusted certificates

- System-wide or application-based store
- More can be added: self-signed, organization certificates, MiTM certificates, etc.

Server provides a chain of certificates

Any CA can sign certificates for any domain

- The system is as secure as the weakest CA



Problems with CAs

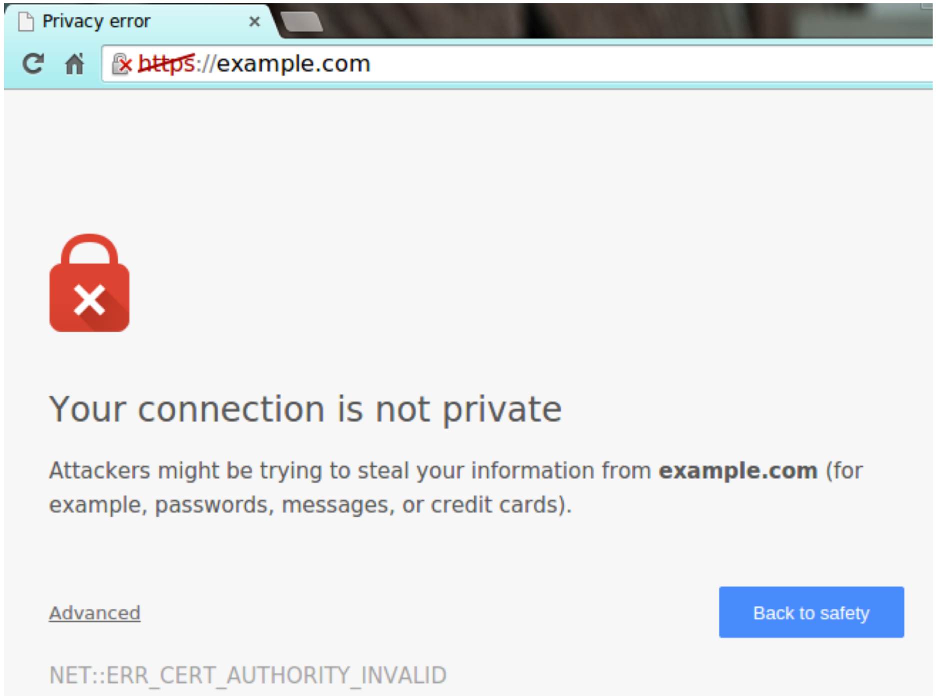
CAs are businesses doing this for profit

- Certificates are expensive Self-signed certs cost nothing

Despite the warnings users tend to keep going

Now you can a cert for free

- https://letsencrypt.org/



Problems with CAs

CAs issuing invalid certs

Google Security Blog

The latest news and insights from Google on security and safety on the Internet

Chrome's Plan to Distrust Symantec Certificates

September 11, 2017

Posted by Devon O'Brien, Ryan Sleevi, Andrew Whalley, Chrome Security

This post is a broader announcement of plans already finalized on the blink-dev mailing list.

Update, 1/31/18: Post was updated to further clarify 13 month validity limitations

Problems with CAs

Misplaced “CA” keys

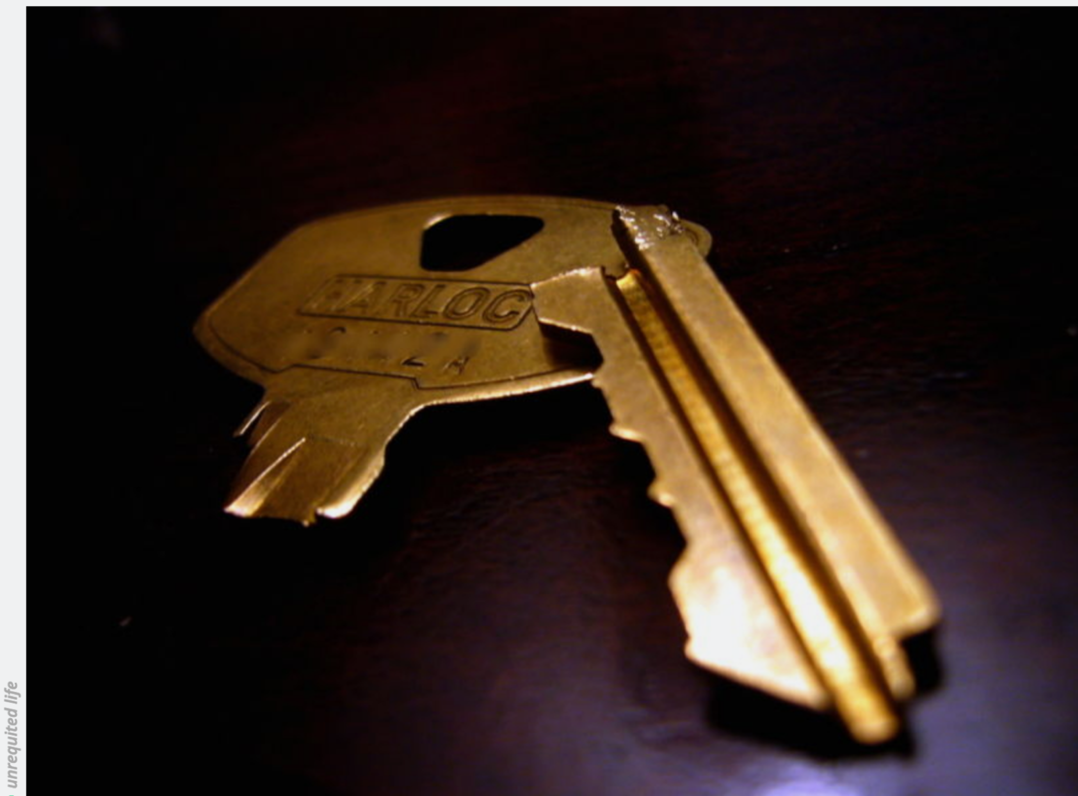
ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORU

DUST UP —

23,000 HTTPS certificates axed after CEO emails private keys

Flap that goes public renews troubling questions about issuance of certificates.

DAN GOODIN - 3/1/2018, 8:36 AM



unrequited life

Enlarge

Problems with CAs

Why is this root cert in my browser?

ars TECHNICA — BIZ & IT | TECH | SCIENCE | POLICY | CARS | GAMING & CULTURE | FOR

NOTHING "DISHONEST"? —

Turkish government agency spoofed Google certificate “accidentally”

CA mistakenly gave Ankara's transit authority even more authority.

SEAN GALLAGHER - 1/4/2013, 3:44 PM

Microsoft has released a security advisory concerning a fraudulent digital certificate for all Google domains apparently created by the Turkish government. The certificate, which was created by a subsidiary Certificate Authority issued to the transportation directorate of the city government of Ankara, could have been used to intercept SSL traffic as part of a "man in the middle" attack to spoof Google's encryption certificate and decrypt secure Web sessions to Google Plus and GMail.

According to a statement from the Turkish certificate authority Turktrust, the organization mistakenly issued two organizations subsidiary CA certificates in 2011—created during testing of Turktrust's certificate production system—instead of the standard SSL certificates they were supposed to receive. Subsidiary CA certificates give the holder the ability to issue SSL certificates with the original CA's authority.

TLS/SSL and Attacks

TLS

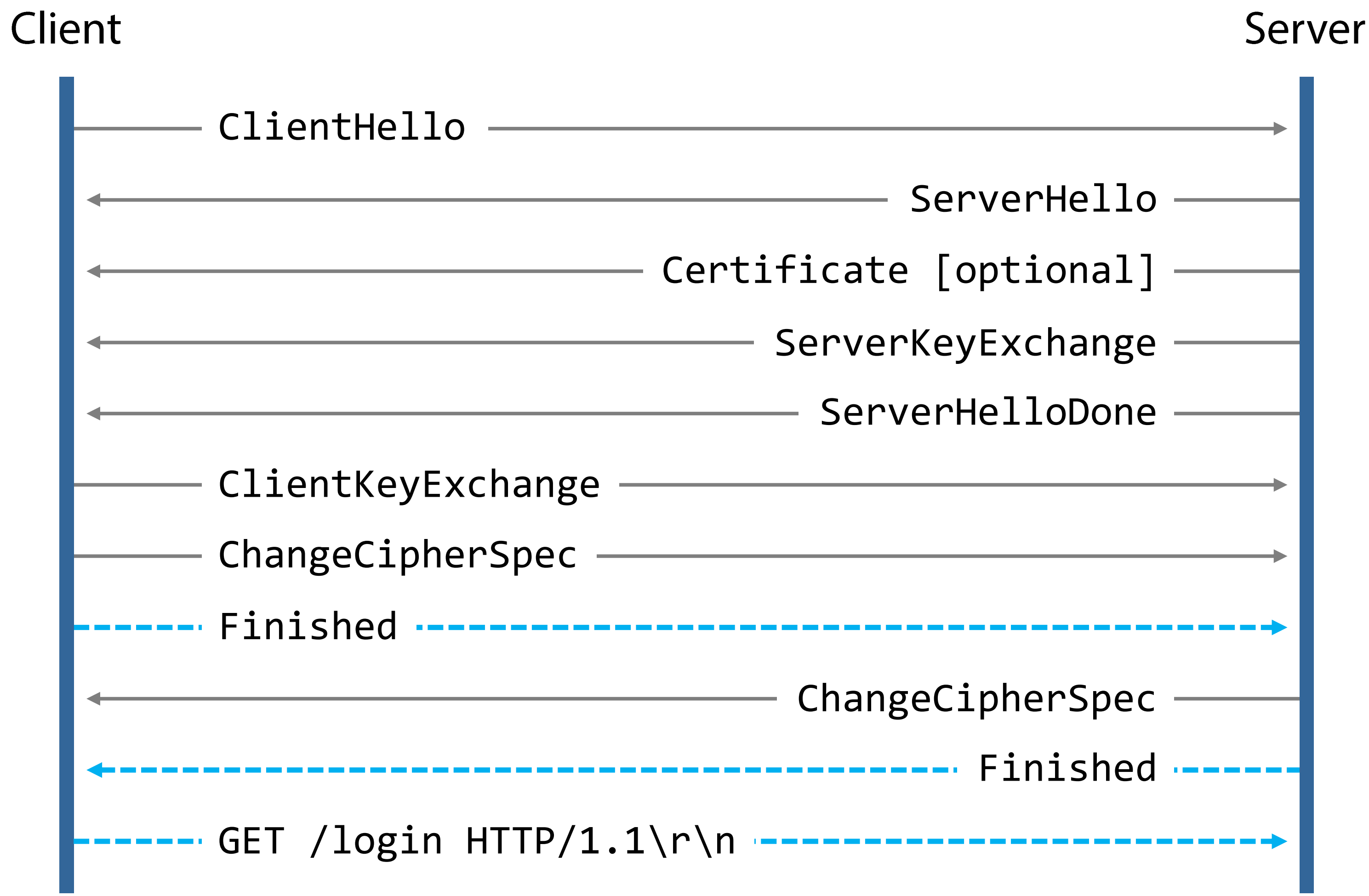
Transport Layer Security (TLS) is the most widely used protocol for secure communications over TCP

Succeeds the Secure Socket Layer (SSL)

- Plagued by various security issues

Used in HTTPS, IMAPS, SMTP, etc.

TLS Handshake



TLS Protocols

Handshake

- Negotiate sessions keys
- Authenticate server and (optionally) client

Record

- Exchange messages encrypted and MACed with established session key
- Compression before encryption
  - **Don't do it**
- Extensible sub-protocols
  - For example, **change the cipher suit used**

Downgrade Attacks

Goal: force the use of a weak cipher suite

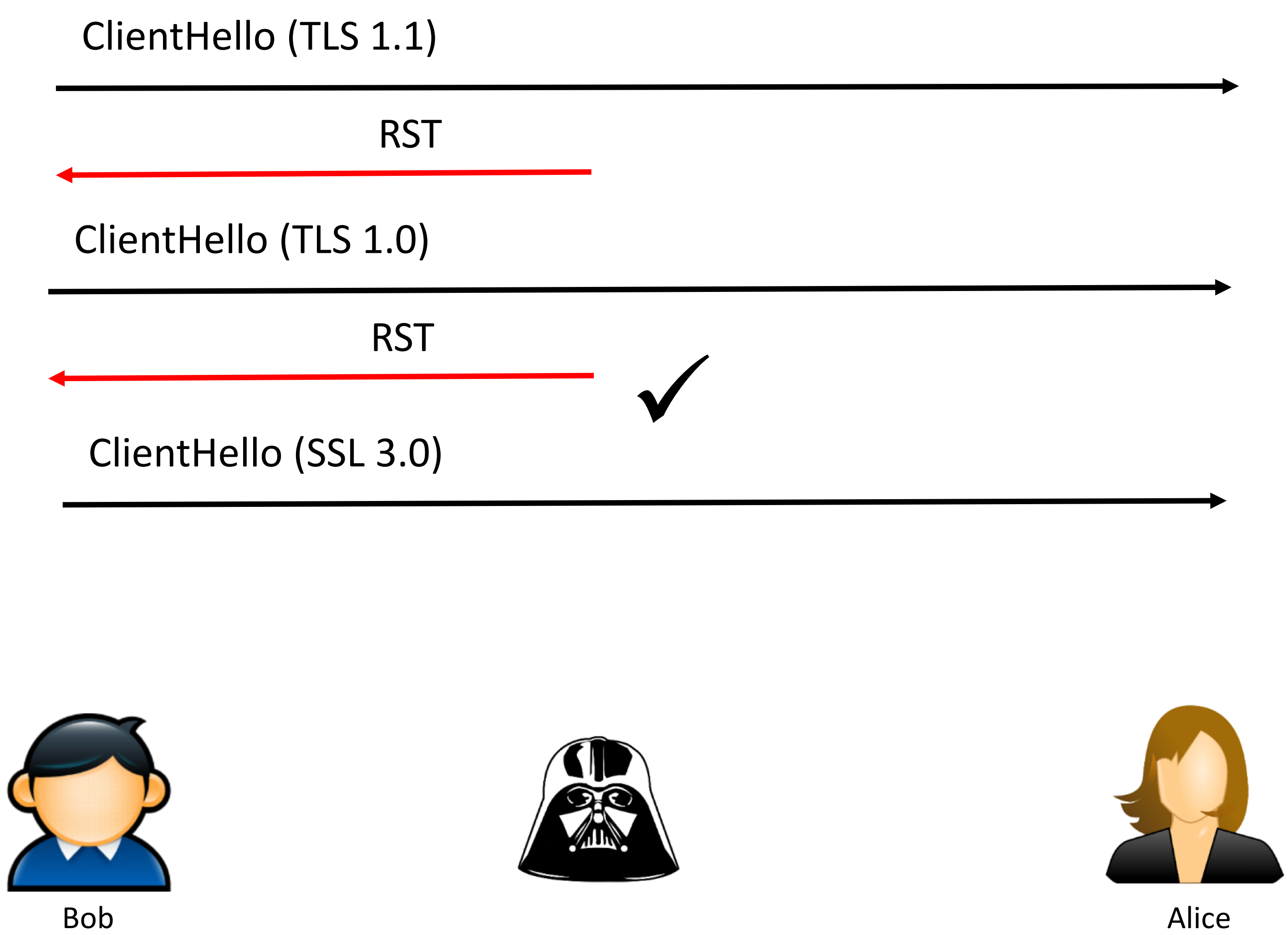
Possible because browsers voluntarily downgrade the protocol upon handshake failure

- For interoperability reasons
- Due to server bugs
- Due to protocol weaknesses

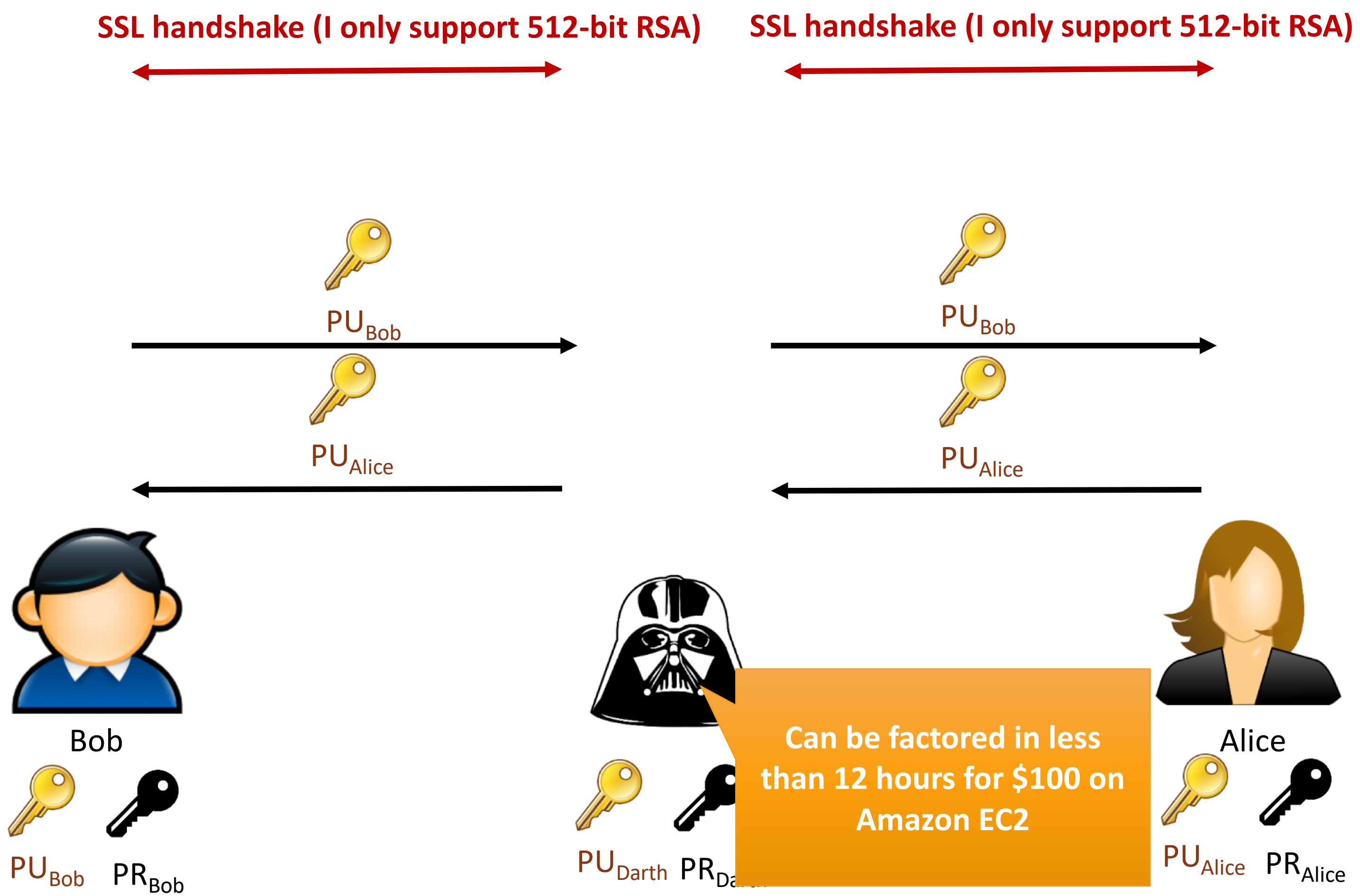
Methods:

- Close connections until retry with lower SSL/TLS version
- Modify list of supported ciphers sent from the client

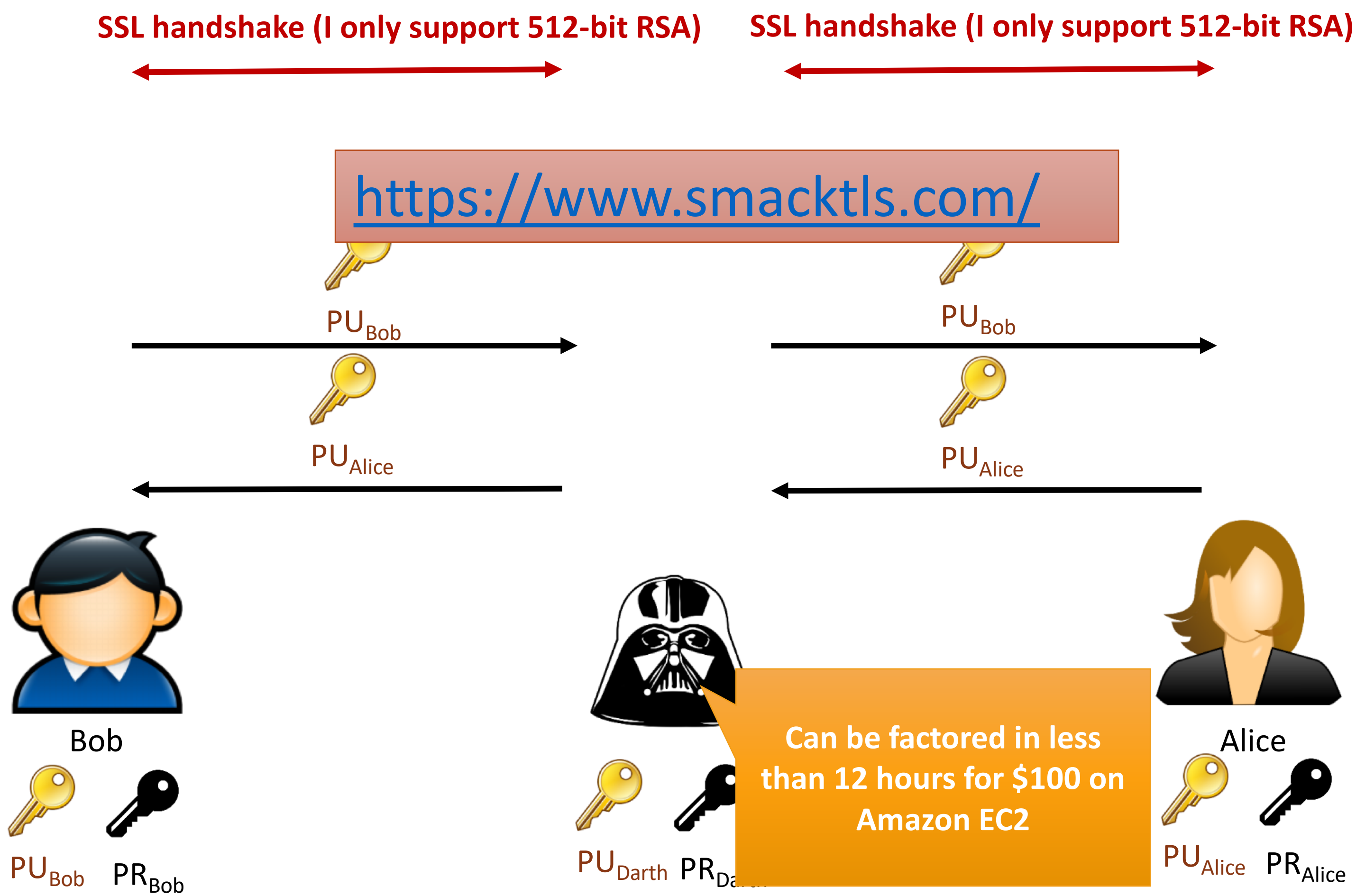
Downgrading TLS Connection



Downgrade Cipher Suite



Downgrade Cipher Suite



SSL Stripping


```
Location: http://…                Location: https://…
 <a href="http://…">               <a href="https://…">
<form action="http://…">         <form action="https://…">
```

HSTS

HTTP Strict Transport Security protects against SSL stripping and other attacks

- Convert any insecure links to https
- Treat all errors as fatal

Implemented through an HTTP header

- `Strict-Transport-Security: max-age=31536000`

You may need to safely load the site once

- Trust-on-first use

Browsers now also do HSTS-preloading

Other Mitigations

HTTP Public Key Pinning

https://en.wikipedia.org/wiki/HTTP_Public_Key_Pinning

Online Certificate Status Protocol

https://en.wikipedia.org/wiki/Online_Certificate_Status_Protocol

Apple Fail (https://gotofail.com/)

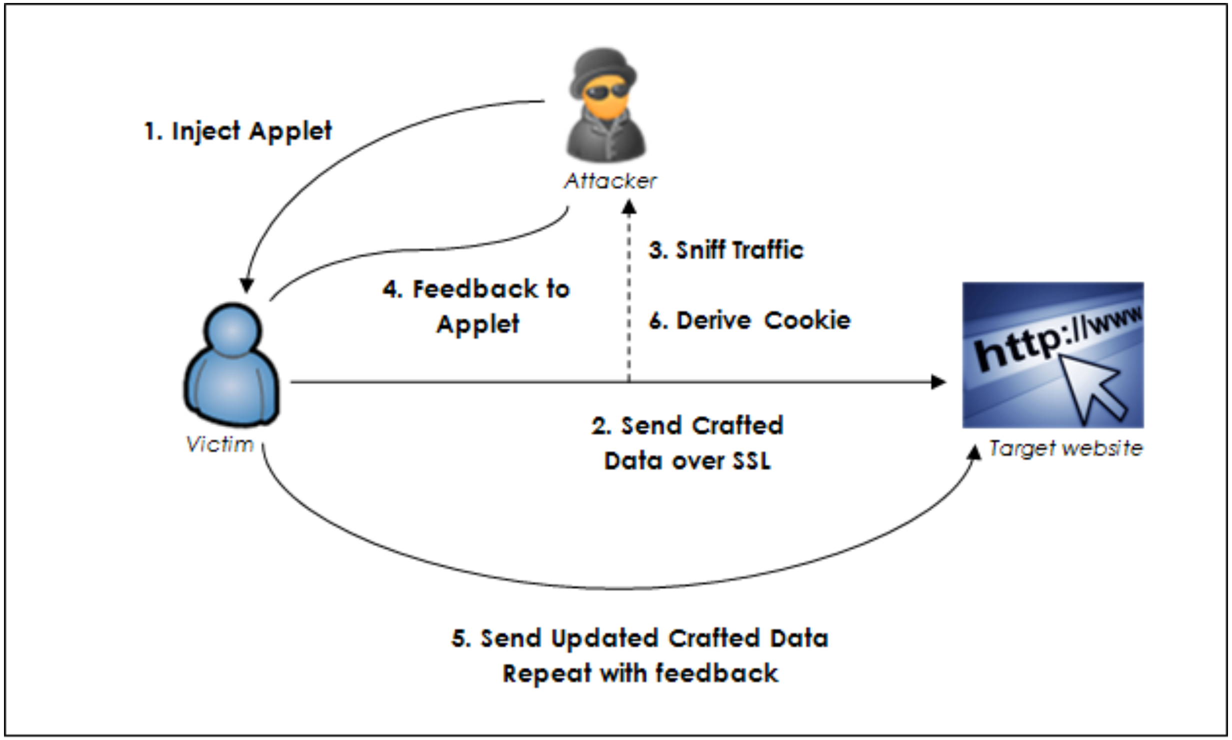
```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
    OSStatus        err;
    SSLBuffer       hashOut, hashCtx, clientRandom, serverRandom;
    uint8_t         hashes[SSL_SHA1_DIGEST_LEN + SSL_MD5_DIGEST_LEN];
    SSLBuffer       signedHashes;
    uint8_t         *dataToSign;
    size_t          dataToSignLen;

    ...
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

    err = sslRawVerify(ctx,
                       ctx->peerPubKey,
                       dataToSign,                /* plaintext */
                       dataToSignLen,             /* plaintext length */
                       signature,
                       signatureLen);
    if(err) {
        sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
        goto fail;
    }
```

CRIME Attack

Leverage compression to leak HTTP cookies

Need to be able to inject a script in a webpage

Issue multiple requests to target website to brute force cookie



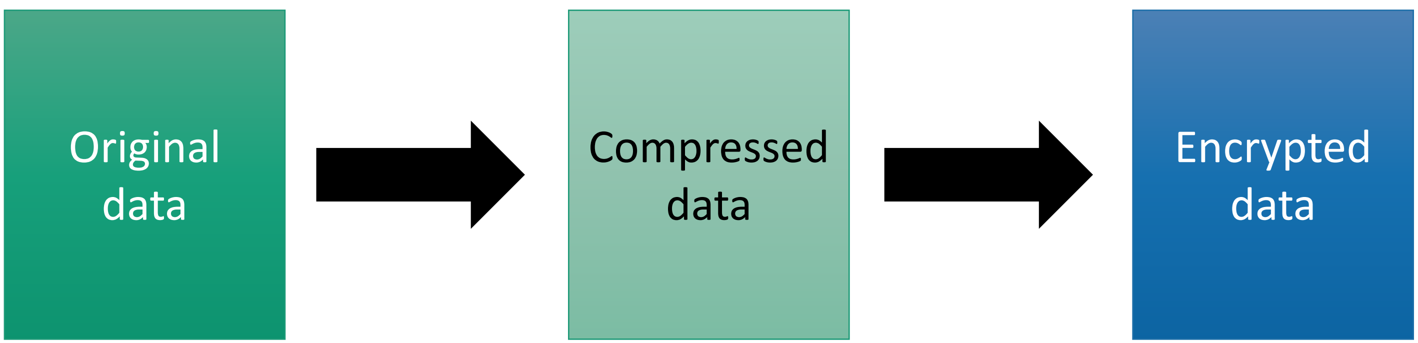
Compression

Header sent with every request

POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

POST data

Slkgloirskjdal3irjlndfsdnvlsidjsdp91jnflijdsf;9jas;ofdas;dqlnds



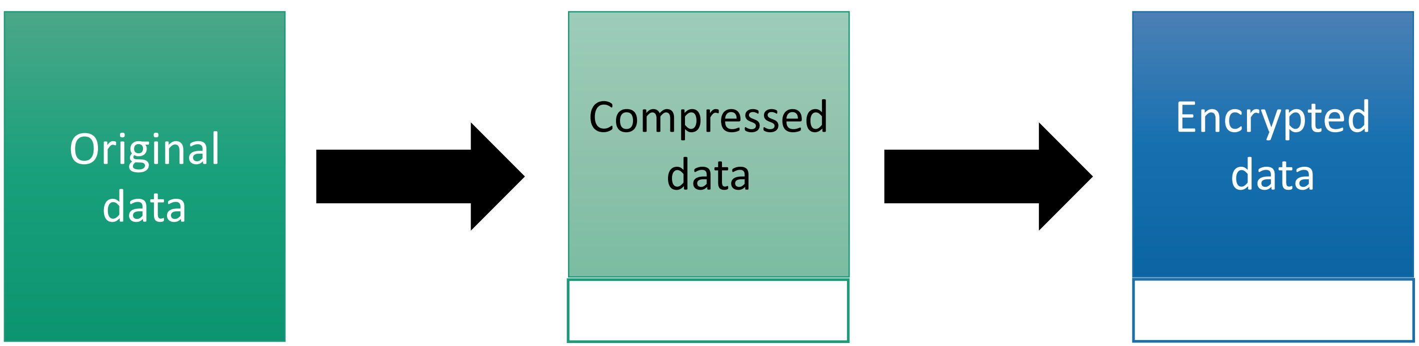
Compression

Header sent with every request

POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

POST data

Cookie: sessionid=a



Saved transmission bandwidth due to compression

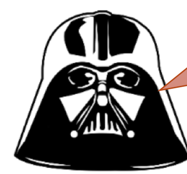
Compression

Header sent with every request

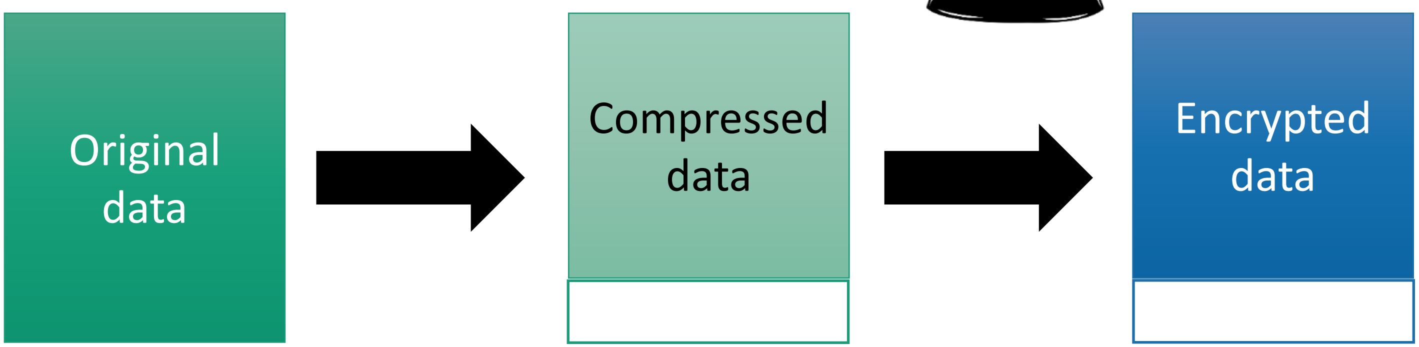
POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

POST data

Cookie: sessionid=d

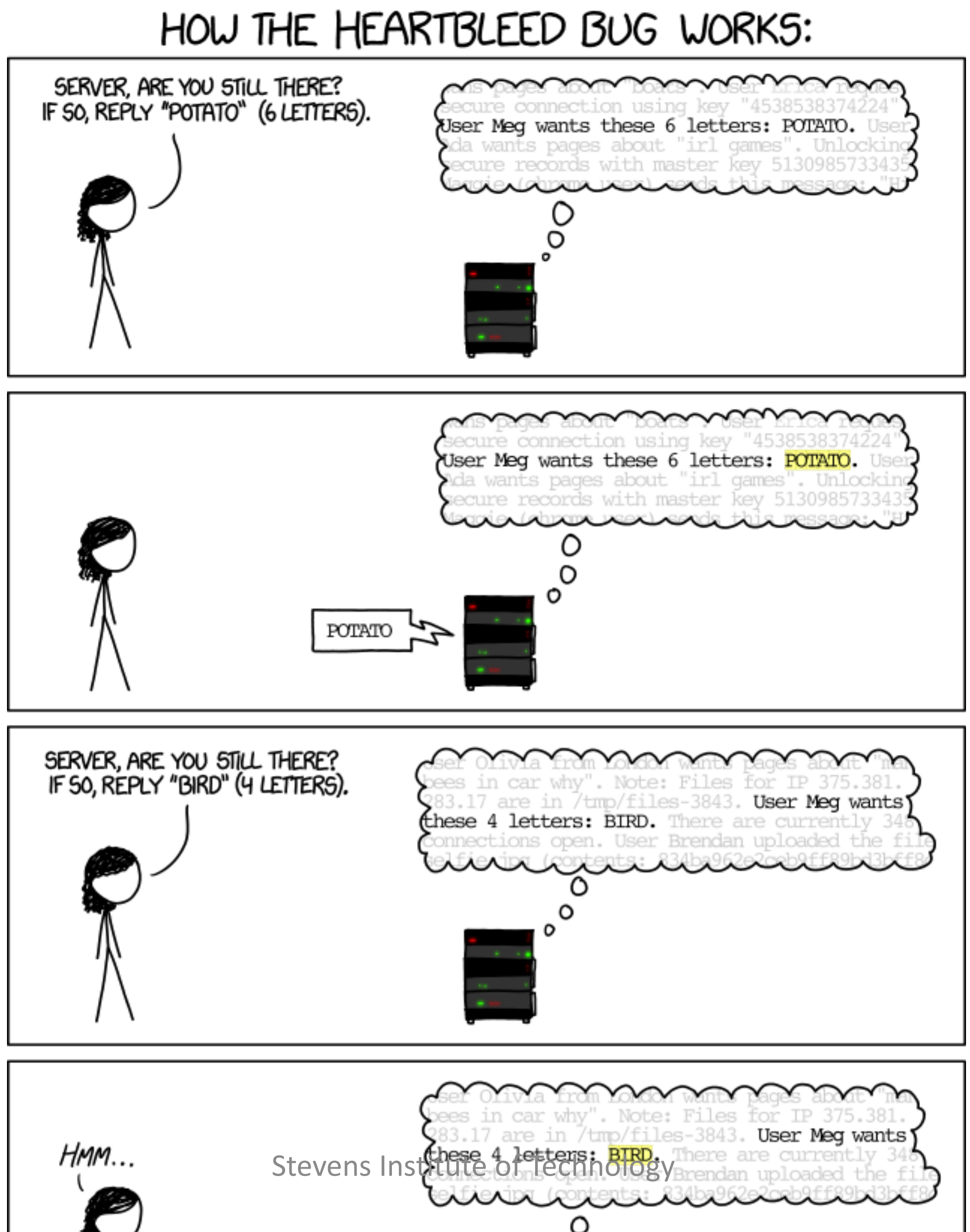


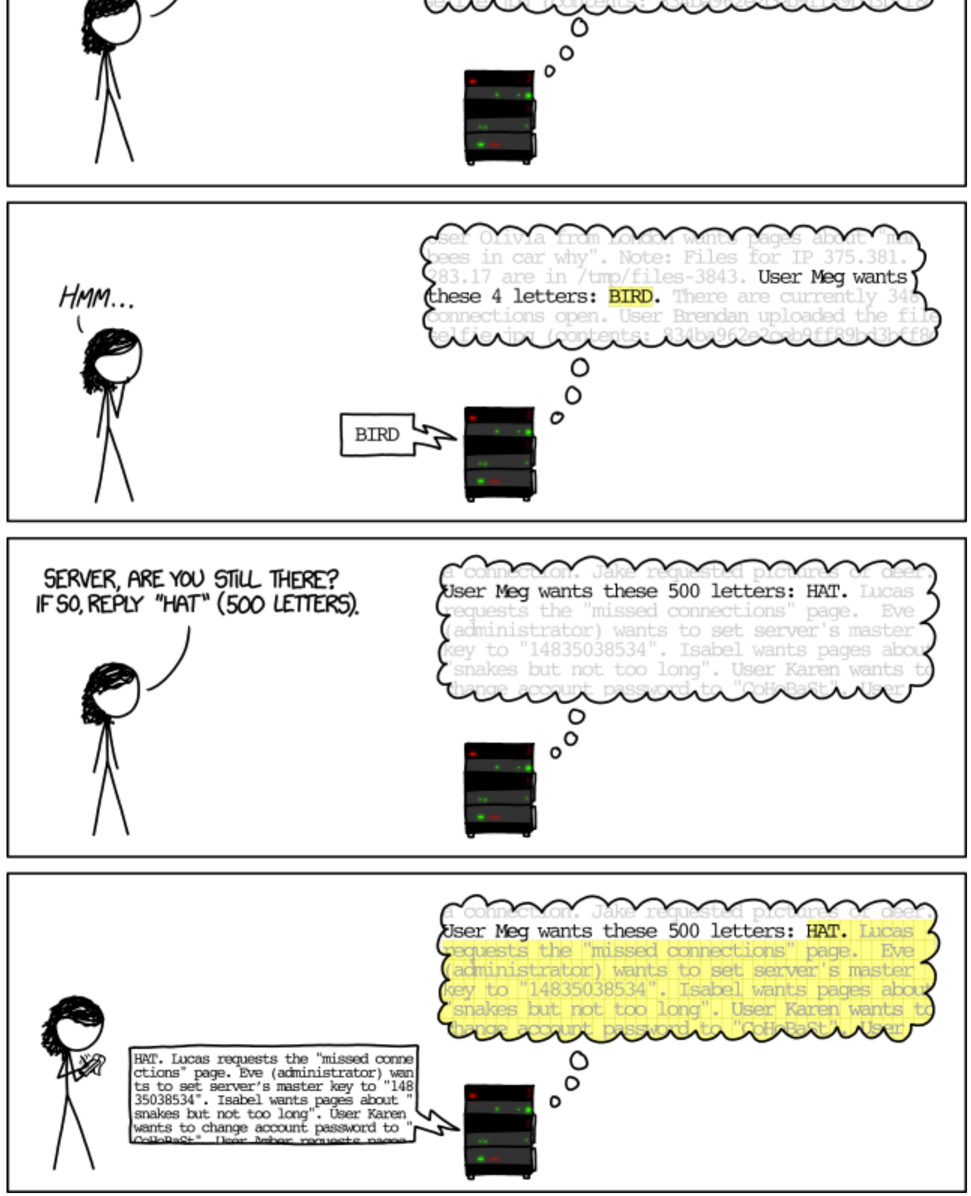
Observing the amount of data transmitted tells me when I get a match in the POST data



Saved transmission bandwidth due to compression

Heartbleed


HOW THE HEARTBLEED BUG WORKS:
SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
User Meg wants these 6 letters: POTATO.
User Meg wants these 6 letters: POTATO.
POTATO
SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
User Meg wants these 4 letters: BIRD.
HMM...
User Meg wants these 4 letters: BIRD.


HMM...
User Meg wants these 4 letters: BIRD.
BIRD
SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to
User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "

Reading

TLS - https://hpbn.co/transport-layer-security-tls/

TLS attacks - https://mitls.org/pages/attacks/

Analysis of the HTTPS Certificate Ecosystem

http://conferences.sigcomm.org/imc/2013/papers/imc257-durumericAemb.pdf