Network Security

Dr. Xiqun Lu

College of Computer Science

Zhejiang University

Network Security

- Network security problems can be divided roughly into four closely intertwined areas:
  - Secrecy (机密性)
  - Authentication (身份认证)
  - Nonrepudiation (不可否认)
  - Integrity control (完整性)

- Except for physical layer security, nearly all network security is based on cryptographic principles.

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security (https)

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security (https)

Outlines

- Basic knowledge
  - Cryptography
    - Kerckhoff's principle
    - Substitution ciphers
    - Transposition ciphers
    - One-time pads
    - Quantum cryptograph
    - Two fundamental cryptographic principles
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys

Cryptography: Kerckhoff's Principle

- All algorithms must be public; only the keys are secret.

- The longer the key, the higher the work factor the cryptanalyst has to deal with.
  - The work factor for breaking the system by exhaustive search of the key space is *exponential* in the key length.
- Encryption methods can be divided into *two categories*: **substitution ciphers** and **transposition ciphers**.

Cryptography: Substitute ciphers

- In a substitution cipher, each letter or group of letters is replaced by another letter or group of letters to disguise it.
- **Caesar cipher** (attribute to Julius Caesar): for example, a becomes D, b becomes E, c becomes F, …, and z becomes C. so “attack” becomes DWWDFN.
  - A slight generalization of the Caesar cipher allows the ciphertext alphabet *to be shifted by k letters*, instead of always three. In this case, ***k* becomes a key** to the general method of circularly shifted alphabets.
  - The next improvement is to have each of the symbols in the plaintext

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

Cryptography: Substitute ciphers

- The basic attack takes advantage of the statistical properties of natural languages.
  - In English, for example, *e* is the most common letter, followed by *t*, *o*, *a*, *n*, *i*, etc. the most common two-letter combinations, or digrams, are *th*, *in*, *er*, *re*, and *an*. The most common three-letter combinations, or trigrams are *the*, *ing*, *and*, and *ion*.
  - For example, consider the following ciphertext from an accounting firm:

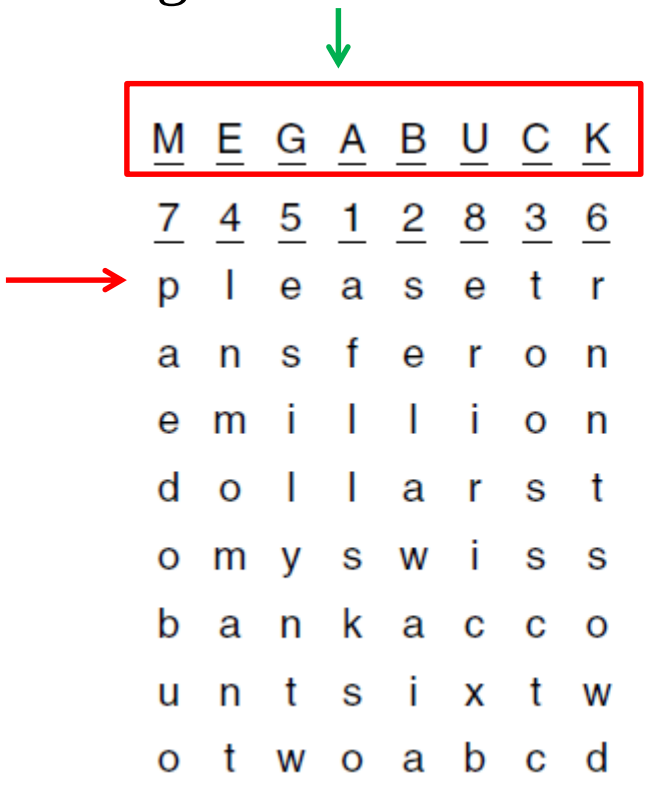
```
CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ
QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW
```

  - A likely word in a message from an accounting firm is "financial".

CTQTZ CQVUJ: C → *i*, T → *n*, Q → *a* 这里两个重复出现字母中间有4个其它字母的组合很多，但是如果按照"financial"，只有第一行最后两块符合要求。(Z → c)

Cryptography: Transposition Ciphers

- Substitution ciphers preserve the order of the plaintext symbols but disguise them.
- Transposition ciphers, in contrast, reorder the letters but do not disguise them.



The cipher is keyed by a word or phrase **not** containing any **repeated** letters.

Plaintext

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Ciphertext

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Cryptography: Transposition Ciphers

- 1) To break a transposition cipher, the cryptanalyst must first be aware that he is dealing with a transposition cipher.
  - By looking at the frequency of *E*, *T*, *O*, *A*, *N*, *I* etc., it is easy to see if they fit the normal pattern of plaintext.
- 2) The next step is to make the guess at the number of columns (the key length).
  - In many cases, a probable word or phrase may be guessed at from the context.
- 3) The remaining step is to order the columns.
  - When the number of columns k is small, each of the $k(k-1)$ column pairs can be examined in turn to see if its digram frequencies match those for English plaintext. The pair with the best match is assumed to be correctly positioned. Now each of the remaining columns is tentatively tried as the successor to this pair.

Cryptography: One-Time Pads*

- First choose a random bit string as the key. Then convert the plaintext into a bit string, for example, by using its ASCII representation. Finally, compute the XOR of these two strings, bit by bit.
- The resulting ciphertext cannot be broken because in a sufficiently large sample of ciphertext, each letter will occur equally often, as will every digram, every trigram, and so on.
  - The reason derives from information theory: there is simply no information in the message because all possible plaintexts of the given length are equally likely.

Message 1:	1001001	0100000	1101100	1101111	1110110	1100101	0100000	1111001	1101111	1110101	0101110
Pad 1:	1010010	1001011	1110010	1010101	1010010	1100011	0001011	0101010	1010111	1100110	0101011
Ciphertext:	0011011	1101011	0011110	0111010	0100100	0000110	0101011	1010011	0111000	0010011	0000101
Pad 2:	1011110	0000111	1101000	1010011	1010111	0100110	1000111	0111010	1001110	1110110	1110110
Plaintext 2:	1000101	1101100	1110110	1101001	1110011	0100000	1101100	1101001	1110110	1100101	1110011

Figure 8-4. The use of a one-time pad for encryption and the possibility of getting any possible plaintext from the ciphertext by the use of some other pad.

Cryptography: One-Time Pads*

- One-time pads are great in theory but have a number of **disadvantages** in practice.
  - The key cannot be memorized, so both sender and receiver must carry a written copy with them.
  - The total amount of data that can be transmitted is limited by the amount of key available.
  - Another problem is the sensitivity of the method to lost or inserted characters.

Cryptography: Quantum Cryptography*

- BB84 (Bennet and Brassard 1984)
- Quantum cryptograph is based on the fact that light comes in little packets called **photons**, which have some peculiar properties.
- Light can be polarized by being pass through a polarizing filter.
  - If a beam of light is passed through a polarizing filter, all the photons emerging from it will be polarized in the direction of the filter's axis. If the beam is now passed through a second polarizing filter, the intensity of the light emerging from the second filter is proportional to the square of the cosine of the angle between the axes. If the two axes are **perpendicular**, no photons get through.
- Bits sent one photon at a time are called **qubits**.

Cryptography: Quantum Cryptography*

- To generate a one-time pad, Alice needs two sets of polarizing filters: one is a rectilinear basis (one vertical filter and one horizontal filter), and the other is a diagonal basis.
  - In reality, Alice does not have four separate filters, but a crystal whose polarization can be switched electrically to any of the four allowed directions at great speed.
  - For each basis, Alice assigns one direction as 0 and the other as 1.
  - Now Alice picks a one-time pad, for example based on a random number generator (a complex subject all by itself). She transfer it bit by bit to Bob, choosing one of her two bases at random for each bit.
  - To send a bit, her photon gun emits one photon polarized appropriately for the basis she is using for that bit.
- Bob has the same equipment as Alice.
- The fact that Alice and Bob each have two bases available is essential to quantum cryptography.

Cryptography: Quantum Cryptography*

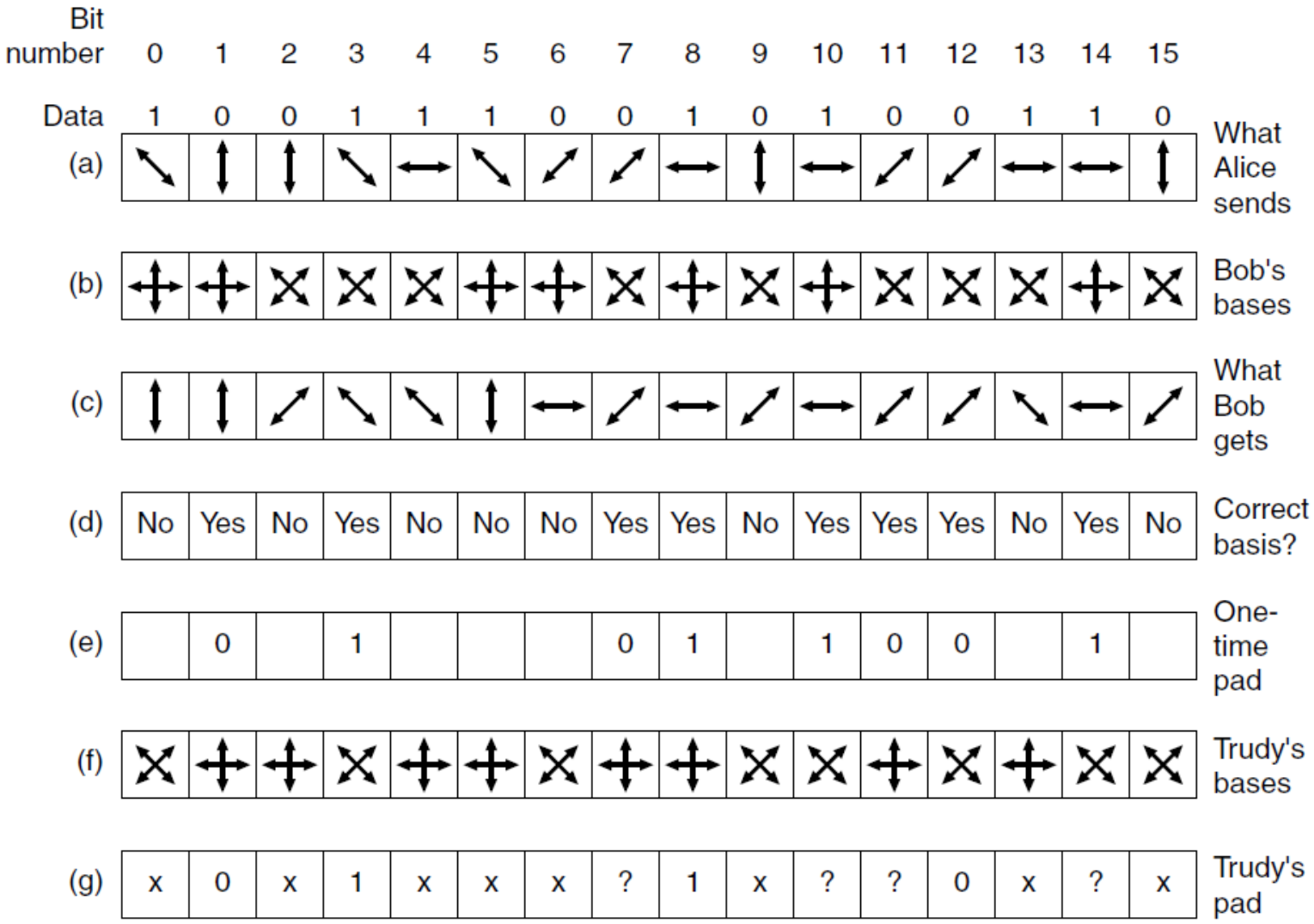


Figure 8-5. An example of quantum cryptography.

Bob does not know which bases to use, so he picks one **at random** for each arriving photon and just uses it. If he picks the incorrect basis, he gets a random bit because if a photon hits a filter polarized at 45 degrees to its own polarization, it randomly jumps to the polarization of the filter or a polarization perpendicular to the filter, with equal probability.

Cryptography: Quantum Cryptography*

- How does Bob find out which bases he got right and which he got wrong?
- He simply tells Alice which basis he used for each bit in plaintext and she tells him which are right and which are wrong in plaintext. From this information, both of them can build a bit string from the correct guesses.
- On the average, this bit string will be half the length of the original bit string, but since both parties know it, they can use it as a one-time pad. All Alice has to do is transmit a bit string slightly more than twice the desired length, and she and Bob will have a one-time pad of the desired length.

Cryptography: Two Fundamental Cryptographic Principles

- 1) All the encrypted messages must contain some **redundancy**.
- 2) Measures must be taken to ensure that each message received can be verified as being **fresh**.

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security

Symmetric-Key Algorithms

- In **symmetric-key** algorithms, they use **the same key** for encryption and decryption.
- Block ciphers take an *n*-bit block of plaintext as input and transform it using the key into an *n*-bit block of ciphertext.

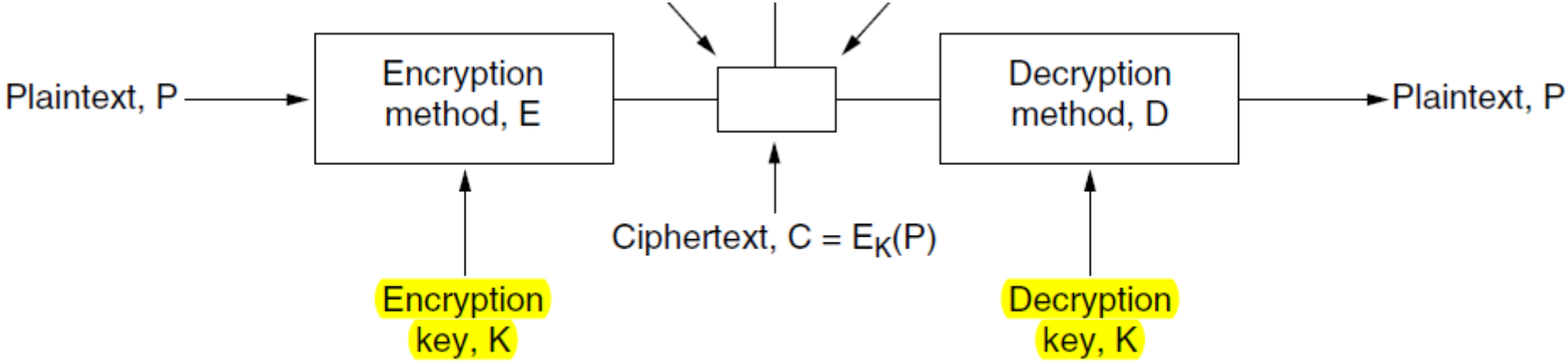


Figure 8-2. The encryption model (for a symmetric-key cipher).

Symmetric-Key Algorithms

- P-box: permutation
- S-box:
  - The 3-bit input selects one of the eight lines exiting from the first stage and sets it to 1; all the other lines are 0.
  - The 2nd stage is a P-box
  - The 3rd stage encodes the selected input line in binary again.

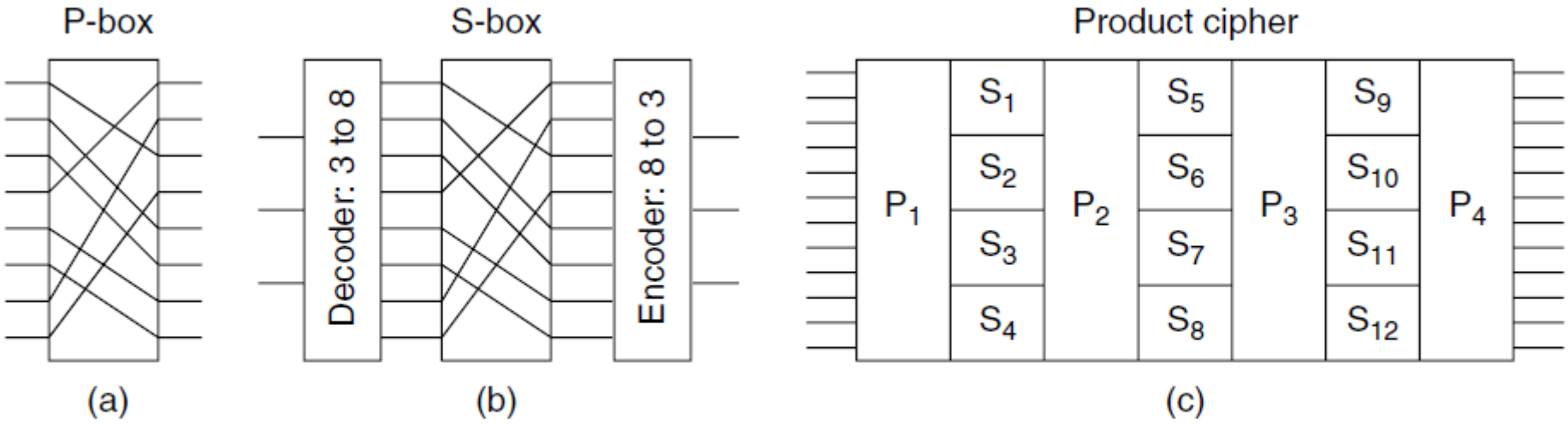
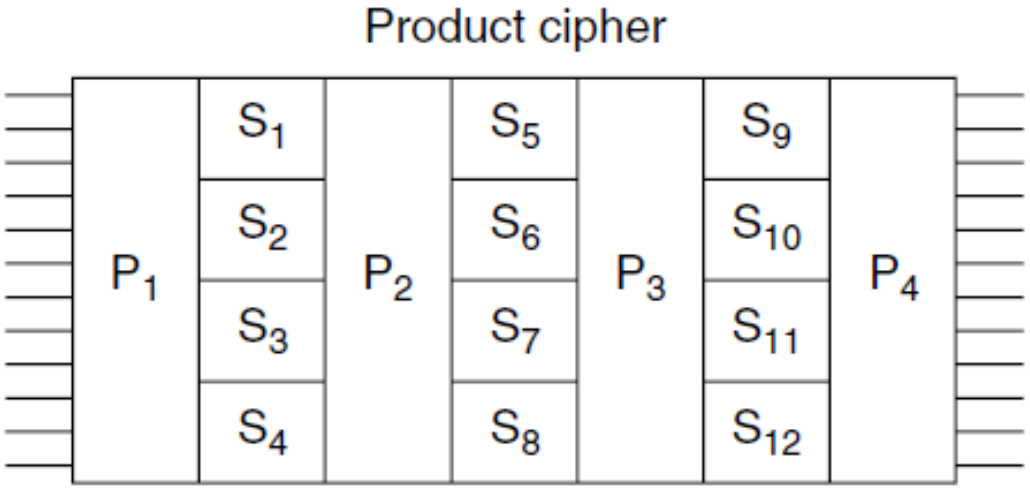


Figure 8-6. Basic elements of product ciphers. (a) P-box. (b) S-box. (c) Product.

Symmetric-Key Algorithms

- The real power of these basic elements only becomes apparent when we cascade a whole series of boxes to form a product cipher.
  - Typically, *k* is 64 to 256. A hardware implementation usually has at least 10 physical stages, instead of just 7 as in Fig.8-6 (c).



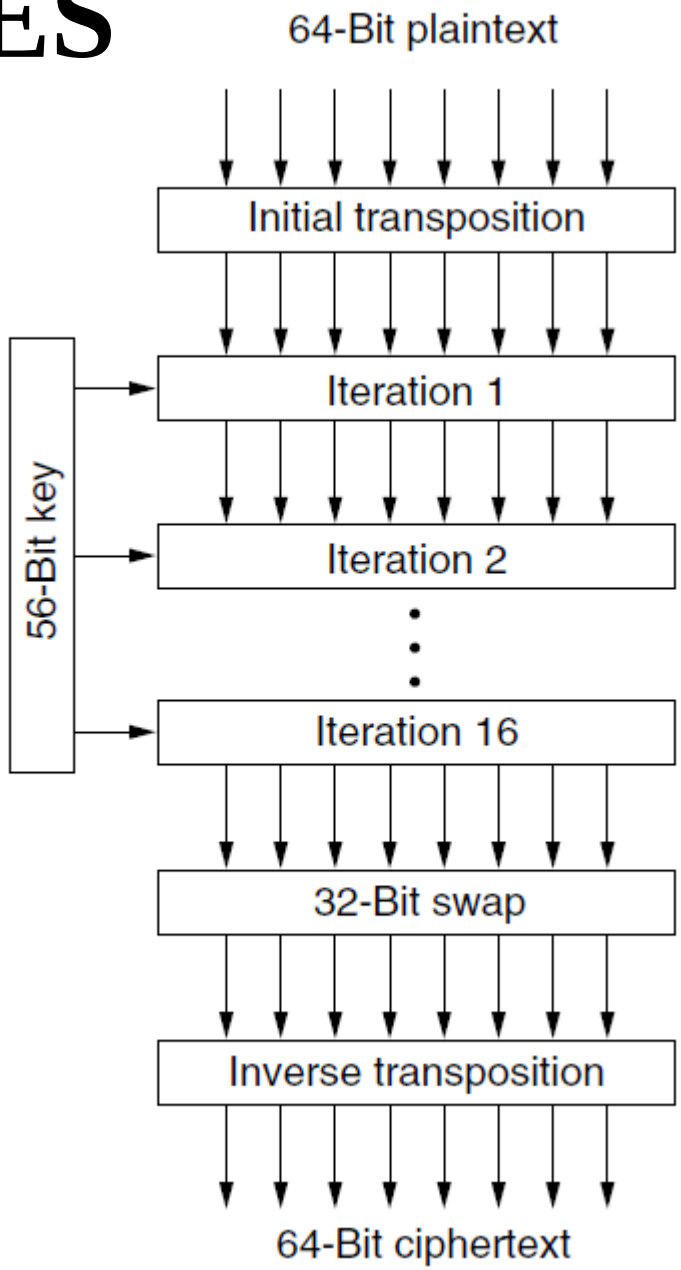
Note here each S is a S-box contains a Decoder, a P-box and an Encoder.

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
    - DES
    - Triple DES
    - Rijndael
  - Public-key algorithms
  - Digital signatures
  - Management of public keys

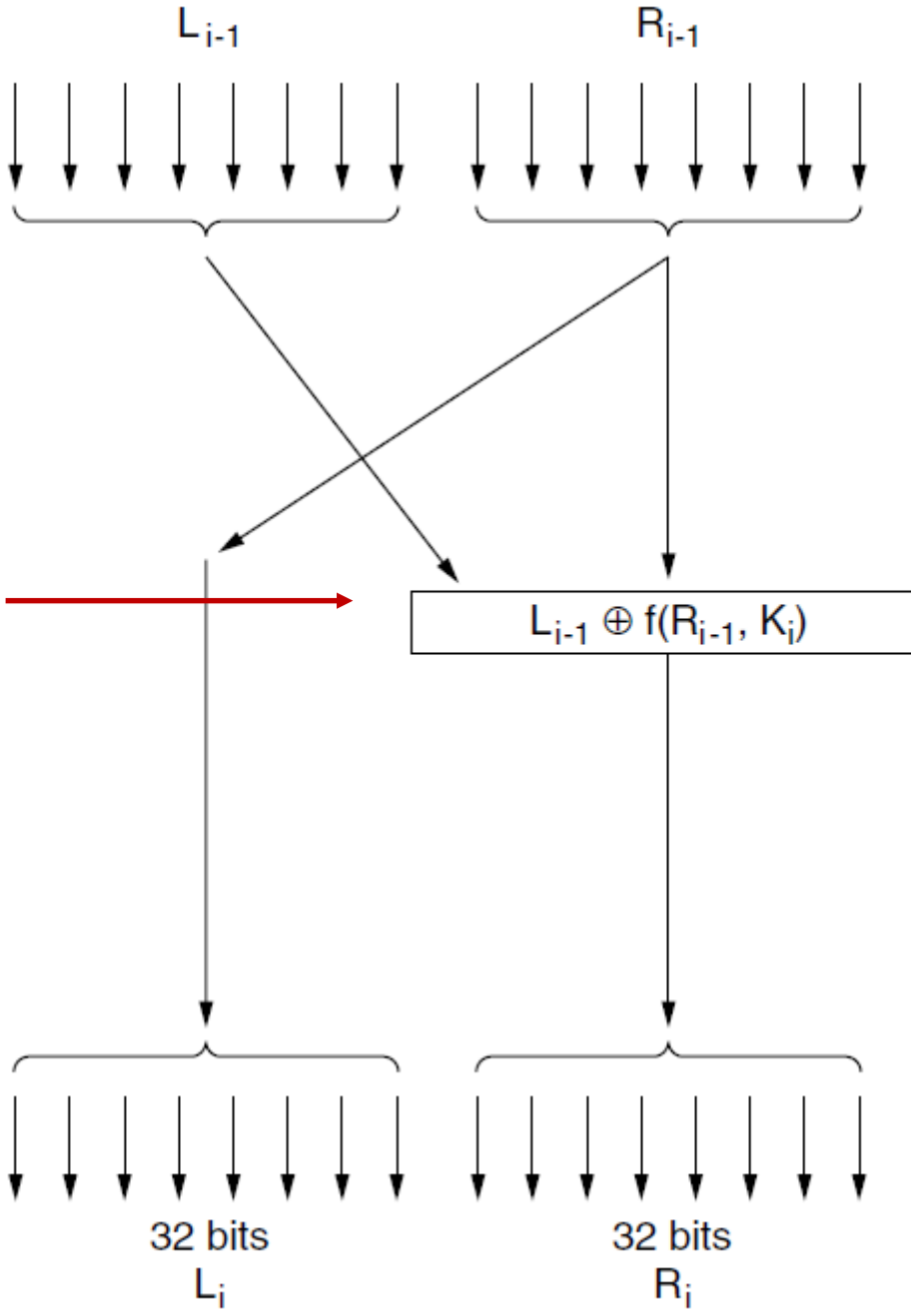
Symmetric-key algorithm: **DES**

- **DES** (Data Encryption Standard, Jan. 1977 for unclassified information)
  - Plaintext is encrypted in blocks of 64 bits, yielding 64 bits of ciphertext.
  - The algorithm, which is parameterized by **a 56-bit key**, has **19 distinct stages**.
  - The first stage is a key-independent transposition on the 64-bit plaintext.
  - The last stage is the exact inverse of this transposition.
  - The stage prior to the last one exchanges the leftmost 32 bits with the right-most 32 bits.
  - The remaining 16 stages are functionally identical but are parameterized by different functions of the key.
  - The algorithm has been designed to allow **decryption** to be done with the same key as encryption. The steps are just run in the reverse order.



DES

- The left output is simply a copy of the right input.
- The right output is the bitwise XOR of the left input and *a function* of the right input and the key for this stage K_i.
- The function consists of 4 steps:
  - 1) A 48-bit number, *E*, is constructed by expanding the 32-bit R_{i-1} according to a fixed transposition and duplication rule.
  - 2) *E* and K_i are XORed together.
  - 3) This output is then partitioned into 8 groups of 6 bits each, each of which is fed into a different S-box. Each of the 64 possible inputs to an S-box is mapped onto a 4-bit output
  - 4) these 8 × 4 bits are passed through a P-box.
- In each of the 16 iterations, a different key is used.



Detail of **One Iteration**

Symmetric-key algorithm: Triple DES

- A technique that is sometimes used to make DES stronger is called **whitening**.
  - It consists of XORing a random 64-bit key with each plaintext block before feeding it into DES and then XORing a second 64-bit key with the resulting ciphertext before transmitting it.
  - Whitening can easily be removed by running the reverse operations (if the receiver has the two whitening keys).
- **Triple DES** (IBM, 1979) — EDE (Encrypt Decrypt Encrypt)
  - The reason for encrypting, decrypting, and then encrypting again is backward compatibility with existing single-key DES systems.

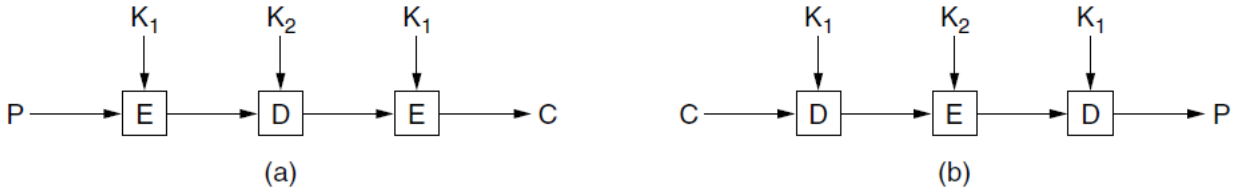


Figure 8-8. (a) Triple encryption using DES. (b) Decryption.

Symmetric-key algorithm: Rijndael

- Rijndael (Oct. 2000, by Joan Daemen and Vincent Rijmen, two young Belgian cryptographers)
  - Based on Galois field theory
  - Like DES, Rijndael uses substitution and permutations, and it also use multiple rounds.
  - Unlike DES, all operations involve entire bytes, to allow for efficient implementation in both hardware and so software.
  - Rijndael encryption and decryption is now part of the instruction set for some microprocessors (e.g. Intel)

Symmetric-key algorithm: Rijndael

```
#define LENGTH 16                                   /* # bytes in data block or key */
#define NROWS 4                                     /* number of rows in state */
#define NCOLS 4                                     /* number of columns in state */
#define ROUNDS 10                                   /* number of iterations */
typedef unsigned char byte;                         /* unsigned 8-bit integer */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
  int r;                                            /* loop index */
  byte state[NROWS][NCOLS];                         /* current state */
  struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];    /* round keys */

  expand_key(key, rk);                              /* construct the round keys */
  copy_plaintext_to_state(state, plaintext);        /* init current state */
  xor_roundkey_into_state(state, rk[0]);            /* XOR key into state */

  for (r = 1; r <= ROUNDS; r++) {
      substitute(state);                            /* apply S-box to each byte */
      rotate_rows(state);                           /* rotate row i by i bytes */
      if (r < ROUNDS) mix_columns(state);           /* mix function */
      xor_roundkey_into_state(state, rk[r]);        /* XOR key into state */
  }
  copy_state_to_ciphertext(ciphertext, state);      /* return result */
}
```

Figure 8-9. An outline of Rijndael in C.

Symmetric-key algorithm: Rijndael

- During the calculation, the current state of the data is maintained in a byte array, **state**, whose size is *NROWS* × *NCOLS*.
- *The state array is initialized to the plaintext and modified by every step in the computation*. In some steps, byte-for-byte substitution is performed. In others, the bytes are permuted within the array. Other transformations are also used. At the end, the contents of the state are returned as the ciphertext.
- 1) expand_key(key, *rk*): The code starts out by expanding the key into 11 arrays of the same size as the *state*. They are stored in *rk*, which is *an array of structs*, each containing a state array. One of these will be used at the start of the calculation and the other 10 will be used during the 10 rounds, one per round.

```
struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];
```

Symmetric-key algorithm: Rijndael

- 2) copy_plaintext_to_state(state, plaintext): copy in column order.
- 3) xor_roundkey_into_state(state, rk[0]): *rk*[0] is XORed into state, byte for byte.

The calculation of the round keys from the encryption key is too complicated for us to get into here.

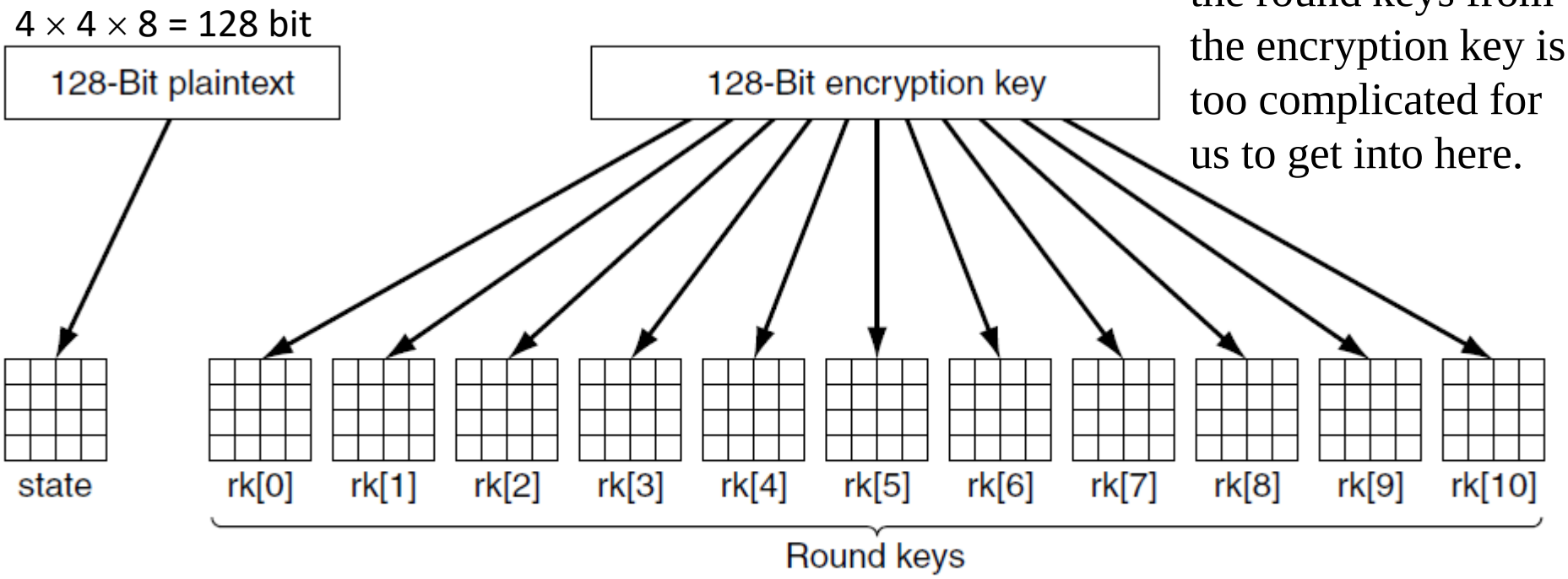


Figure 8-10. Creating the *state* and *rk* arrays.

Symmetric-key algorithm: Rijndael

- The loop executes 10 iterations: one per round, transforming state on each iteration. The contents of each round is produced in 4 steps:
  - Step 1 does a byte-for-byte substitution on state. Each byte in turn is used as an index into an S-box to replace its value by the contents of that S-box entry. ~ subsitute (state)
  - Step 2 rotates each of the four rows to the left. Row 0 is rotated by 0 bytes (i.e., not changed), row 1 is rotated 1 byte, row 2 is rotated by 2 bytes, and row 3 is rotated by 3 bytes. ~ rotate_rows (state)
  - Step 3 mixes up each column independently of the other ones. The mixing is done using matrix multiplication in which the new column is the product of the old column and a constant matrix, with the multiplication done using the finite Galois field, $GF(2^8)$. ~ mix_columns (state)
  - Step 4 XORs the key for this round into the state array for use in the next round.~ xor_roundkey_into_state (state, *rk*[r])

Other symmetric-key ciphers

Cipher	Author	Key length	Comments
DES	IBM	56 bits	Too weak to use now
RC4	Ronald Rivest	1–2048 bits	Caution: some keys are weak
RC5	Ronald Rivest	128–256 bits	Good, but patented
AES (Rijndael)	Daemen and Rijmen	128–256 bits	Best choice
Serpent	Anderson, Biham, Knudsen	128–256 bits	Very strong
Triple DES	IBM	168 bits	Good, but getting old
Twofish	Bruce Schneier	128–256 bits	Very strong; widely used

Figure 8-16. Some common symmetric-key cryptographic algorithms.

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security

Public-Key Algorithms

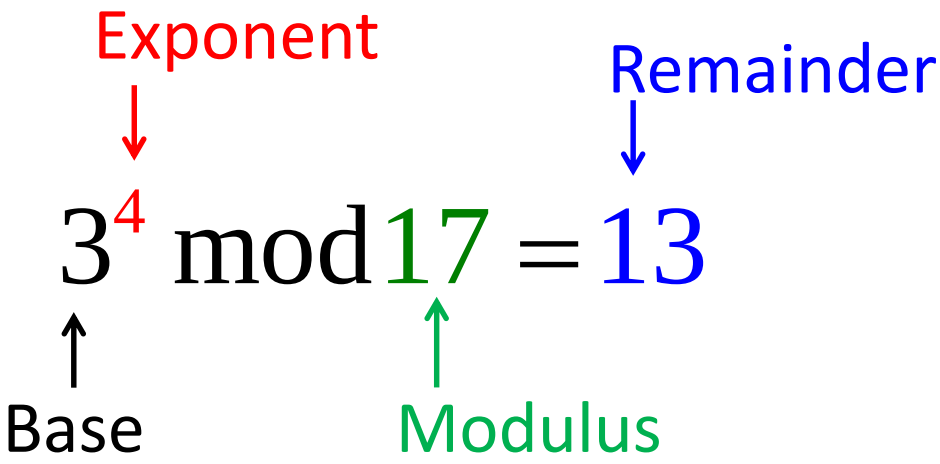
- Historically, *distributing the keys has always been the weakest link in most cryptosystems*. No matter how strong a cryptosystem was, if an intruder could steal the key, the system is worthless.
- In 1976, two researchers at Stanford University, **Diffie** and **Hellman** proposed a radically new kind of cryptosystem, one in which *the encryption and decryption keys were so different that the decryption key could not feasibly be derived from the encryption key*.
- **These requirements** can be stated simply as follows:
  - 1. D(E(*P*)) = *P*.
  - 2. It is exceedingly difficult to deduce D from E.
  - 3. E cannot be broken by a chosen plaintext attack.
- Public-key cryptography requires each user to have two keys: **a public key**, used by the entire world for encrypting message to be sent to that user, and **a private key**, which the user needs for decrypting messages.

Public-key algorithm: RSA

- In 1970, **James H. Ellis**, a British Engineering mathematician was working on an idea for **non secret encryption**. It is based on a simple yet clever concept: Alice can send *the open lock* to Bob, but keep the key secretly. — No keys are exchanged.
  - To split a key into an encryption key and a decryption key, and the decryption key performs the inverse or undo operation which was applied by the encryption key.
- The solution was found by another British mathematician encryptographer: **Clifford Christopher Cocks**. Cocks needed to construct a special kind of **one-way function**, called *the Trapdoor one-way function*. This is a function that is easy to compute in one direction yet difficult to reverse unless you have special information called Trapdoor.

RSA (I)

- Modular exponentiation (one-way function)



- Message → a number "*m*"

$$m^e \bmod N = c \xrightarrow{\text{easy}}$$

$$\xleftarrow{\textit{hard}} ?^e \bmod N = c$$

Here N is a random number

RSA (II)

- To reverse the encryption

$$m^e \bmod N = c$$

$$c^d \bmod N = m$$

$$\downarrow$$

$$\left(m^e\right)^d \bmod N = m$$

$$m^{ed} \bmod N = m$$

- Here e is the encryption key and d is the decryption key.

RSA (III)

- Therefore, we need a way to construct *e* and *d*, which makes it difficult for anyone else to find *d*.
- Now the question is how to find *d*? — **Euclidean's Prime Factorization**: Euclidean showed that every number has exactly one prime factorization.
  - For example: $30 = 5 \times 3 \times 2$
- As we all known, the multiplication of two primes is easy, but to find the prime factorization of a very large number is hard.
  - For example: to find the prime factorization for 437231?
  - If we try to get a computer to factor larger and larger numbers, there is **a runway effect**. This is a hard problem.

RSA (IV)

- 1) So factorization is what Cocks used to build the Trapdoor solution.
  - Alice generate two random **prime** numbers **with roughly the same size**: P_1 and P_2 (~ 150 digits) and multiply them together to get a composite number $N = P_1 \times P_2$ (~ 300 digits long).
  - She can give the number N to anyone else, but hide the factorization as a secret. They would have to have a computer running for years to find the solution.
- 2) Cocks needed to find a function which depends on knowing the factorization of N. Now he turned to **the Distribution of Prime Numbers** found by Leonardo Euler (Swiss mathematician) in 1760.

RSA (V)

- One important function Euler defined is called Φ (**Phi function**) — it measures the breakability of a number. Φ(*N*) outputs how many integers that are less than N that do **not** share any common factor with *N*.
  - For example Φ(8) = 4. (∵ 1, 2, 3, 4, 5, 6, 7), so calculating the Phi function is hard, except in one case: **The Phi of any prime number P is simply P – 1**. Φ(7) = 6. Φ(21377) = 21376. Therefore, Phi of any prime is easy to computer.
  - The Phi function is also **multiplicative**: $\Phi(A \times B) = \Phi(A) \times \Phi(B)$.
- If we know a number *N* is the product of two primes P_1 and P_2, then : $\Phi(N) = \Phi(P_1) \times \Phi(P_2) = (P_1 - 1) \times (P_2 - 1)$.
- We now have a trapdoor for solving Φ (Phi). If you know the factorization of *N*, then finding the Φ(*N*) is easy.

RSA (VI)

- 3) How to connect the Φ (Phi) function to modular exponentiation? **Euler's Theorem**: the relationship between the Phi function and modular exponentiation:

$$m^{\Phi(n)} = 1 \bmod n = 1$$

  - This means that you can pick any two numbers such that they do not share a common factor. For example: $m = 5$, $n = 8$, $5^{\Phi(8)} = 5^4 = 625 = 1 \bmod 8 = 1$ (625 mod 8 = 1).

$$1^k = 1$$

$$m^{k\Phi(n)} = 1 \bmod n$$

RSA (VII)

$$1 \times m = m \quad m^{k\Phi(n)} = 1 \bmod n = 1$$

$$m \times m^{k\Phi(n)} = m \bmod n$$

Note: m and n share no common factor

$$m^{k\Phi(n)+1} = m \bmod n$$

$$m^{e \times d} = m \bmod n$$

$$e \times d = k\Phi(n) + 1 \rightarrow d = \frac{k\Phi(n)+1}{e}$$

- We now have an equation for finding e times d which depends on $\Phi(n)$. It is easily to calculate d only if the factorization of n is known. Now d should be **the private key** of Alice.

RSA (VIII)

- This trick was independently *rediscovered* by a group at MIT (Rivest, Shamir, Adleman, 1978)
  - It has survived all attempts to break it for more than 30 years and is considered very strong.
  - Rivest, Shamir and Adleman were given **the 2002 ACM Turing Award**.
- The RSA method is based on some principles from **number theory**.
  - 1. Choose two large primes, p and q (typically 1024 bits)
  - 2. Compute $n = p \times q$ and $z = (p - 1) \times (q - 1)$.
  - 3. Choose a number relatively prime to z and call it d.
  - 4. Find e such that $e \times d = 1 \bmod z$.
  - "n" and "e" is the public key, and "d" is the private key.
- Its major disadvantage is that it requires keys of at least 1024 bits for good security.

RSA (IX)

- A trivial pedagogical example of how the RSA algorithm works: $p = 3$ and $q = 11$, giving $n = 33$ and $z = 20$. A suitable value for d is $d = 7$, since 7 and 20 have no common factors. With these choices, e can be found by solving the equation $7e = 1 \pmod{20}$, which yields $e = 3$.

Plaintext (P)			Ciphertext (C)		After decryption	
Symbolic	Numeric	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$	Symbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
Sender's computation				Receiver's computation		

Figure 8-17. An example of the RSA algorithm.

ECC

- Elliptic Curve Cryptography
  - 它基于**椭圆曲线离散对数**的数学难题，使用特定的曲线方程和基点生成公钥和私钥，子算法 ECDHE 用于秘钥交换，ECDSA 用于数字签名。
  - 目前常用的两个曲线是 P-256（secp256r1，在 OpenSSL 称为 Prime256v1）和 x25519。P-256 是 NIST（美国国家标准技术研究所）和 NSA（美国国家安全局）推荐使用的曲线，而x25519 被认为是最安全、最快速的曲线。
  - 比起 RSA，ECC 在安全强度和性能上都有明显的优势。160 位的 ECC 相当于 1024 位的 RSA，而 256 位的 ECC 则相当于 2048 位的 RSA。因为秘钥短，所以相应的计算量、消耗内存和带宽也就少，加解密的性能就会好，对于现在移动互联网非常有吸引力。

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security

Digital Signatures

- The **authenticity** of many legal, financial, and other documents is determined by the presence or absence of an authorized handwritten signature.
- Digital signatures allow digital documents to be signed in an unforgeable way.
- The basic requirements of digital signatures:
  - 1. The receiver can verify the claimed identity of the sender.
  - 2. The sender cannot later repudiate the contents of the message.
  - 3. The receiver cannot possibly have concocted the message himself.

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
    - Symmetric-key signatures
    - Public-key signatures
    - Message digests
    - The birthday attack
  - Management of public keys

Symmetric-Key Signatures

- To have a **central authority** (such as Big Brother)
- A and B are Alice and Bob's identities, respectively
- P is a signed plaintext sent by Alice to Bob.
- R_A (a random number chosen by Alice) and the timestamp *t* are used for *freshness*.
- K_A, K_B and K_{BB} are the secrete keys of Alice, Bob and BB, respectively.

- One potential problem with the signature protocol of Fig.8-18 is **the replay attack**. (R_A and *t* are used to minimize this problem)

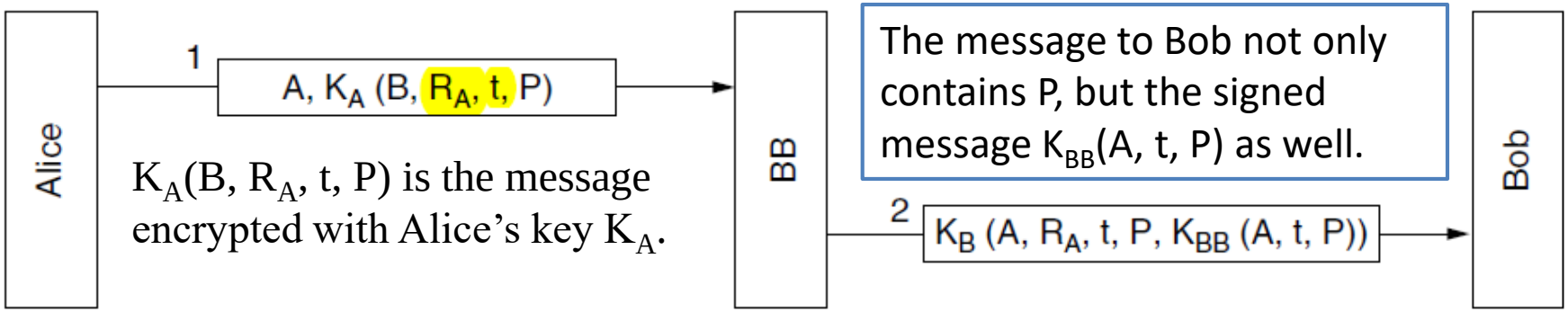


Figure 8-18. Digital signatures with Big Brother.

Replay Attack [2]

- A **replay attack** (also known as **playback attack**) is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed.
- This is carried out either by the originator or by an adversary who intercepts the data and re-transmits it, possibly as part of a *spoofing attack* by **IP packet substitution**. This is one of the lower-tier versions of a man-in-the-middle attack.
  - To minimize this problem, timestamps are used throughout.
  - To check all recent message to see if R_A is used in any of them

Public-Key Signatures (I)

- A structural problem with using symmetric-key cryptography for digital signatures is that everyone has to agree to *trust Big Brother*.
- Signing documents did not require a trusted authority.
- Assume that the public-key encryption and decryption algorithms have the property that $E(D(P)) = P$, in addition, to the usual property $D(E(P)) = P$.
  - **RSA** has this property. The de facto industry standard is the RSA algorithm. Many security products use it.
  - In principle, any public-key algorithm can be used for digital signatures.

Public-Key Signatures (II)

- Alice can send a signed plaintext message P, to Bob by transmitting $E_B(D_A(P))$. (D_A is Alice's private key, and E_B is Bob's public key.)
- When Bob receives the message, he transforms it using his private key, D_B, as usual, yielding $D_A(P)$. He stores this text in a safe place and then applies E_A (Alice's public key) to get the original plaintext.
  - Because Bob don't know what Alice's private key is ($D_A(P)$), the only way Bob could have acquired a message encrypted by it is if Alice did indeed send it.

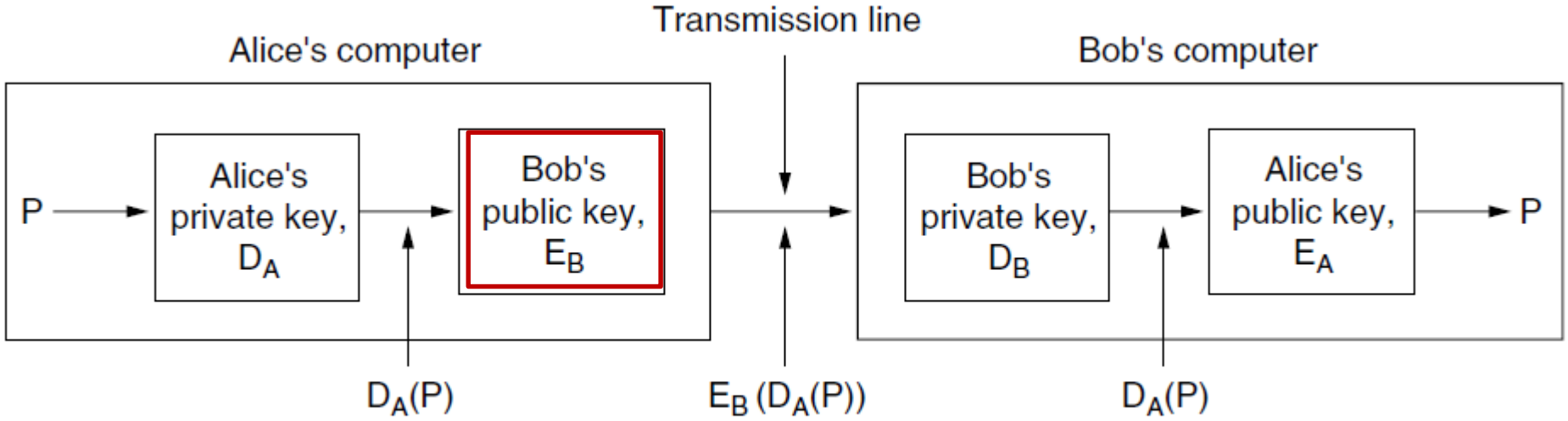


Figure 8-19. Digital signatures using public-key cryptography.

Message Digests (I)

- One criticism of signature methods is that they often couple two distinct functions: **authentication** and **secrecy**. Often authentication is needed but secrecy is not always needed.
- The following authentication scheme is based on the idea of **a one-way hash function** that takes an arbitrarily long piece of plaintext and from it computes a fixed-length bit string.
  - This scheme does **not** require encrypting the entire message.
- This hash function, MD, often called a **message digest**, has four important properties:
  - 1. Given *P*, it is easy to compute *MD*(*P*). — **efficiency**
  - 2. Given *MD*(*P*), it is effectively impossible to find *P*. — **confidential**
  - 3. Given *P*, no one can find *P′* such that *MD*(*P′*) = *MD*(*P*). — **uniqueness**
    - The hash should be at least 128 bits long, preferably more.
  - 4. A change to the input of even 1 bit produces a very different output. — **trustworthiness**
    - The hash must mangle the bits very thoroughly.

Message Digests (II)

- Computing a message digest from a piece of plaintext is **much faster** than encrypting that plaintext with a public-key algorithm, so message digests can be used to speed up digital signature algorithms.
- Message digests work in public-key cyptosystems, as shown in Fig.8-20.
  - Alice first computes the message digest of her plaintext. She then signs the message digest and sends both the signed digest and the plaintext to Bob.
  - If Trudy replaces *P* along the way, Bob will see this when he computes MD(*P*)

Alice's private key is (D_A(P))

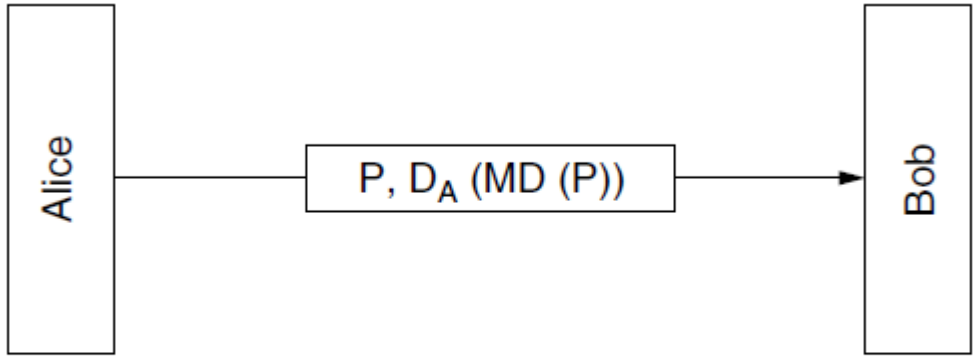


Figure 8-20. Digital signatures using message digests.

Message Digests (III)

- Message digests work in symmetric-key cyptosystems, as shown in Fig.8-18.
  - Instead of signing P with $K_{BB}(A, t, P)$, BB now computes the message digest by applying MD to P, yielding MD(P).

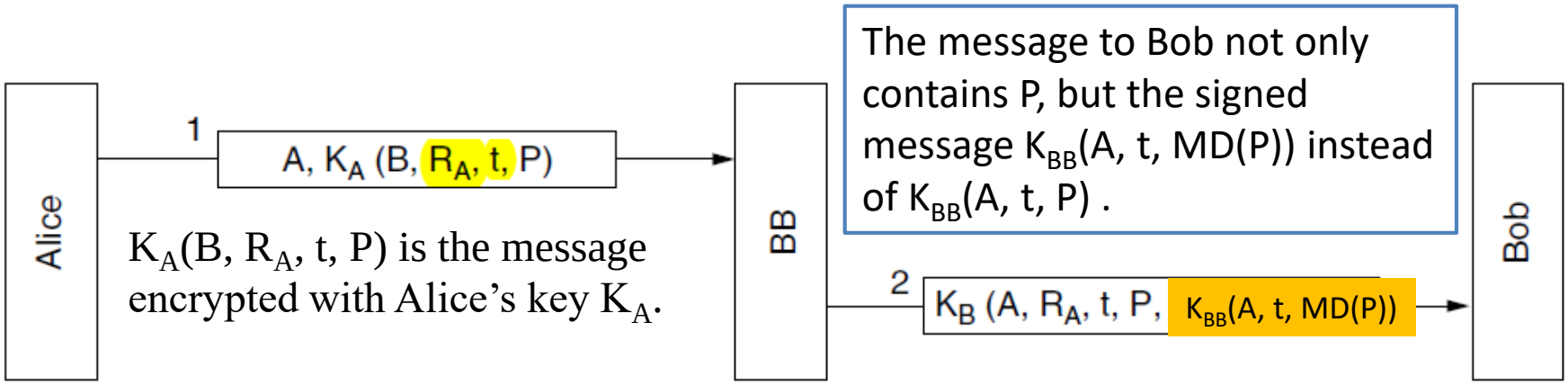


Figure 8-18. Digital signatures with Big Brother.

SHA-1 and SHA-2

- SHA-1 (Secure Hash Algorithm, **NIST**, 1993)
  - It operates by mangling bits in a sufficiently complicated way that every output bit is affected by every input bit.
  - It processes input data in 512-bit blocks, and it generates a 160-bit message digest.
- After receiving the message, Bob computes the SHA-1 hash himself and also applies Alice's public key to the signed hash to get the original hash, H. If the two agree, the message is considered valid.

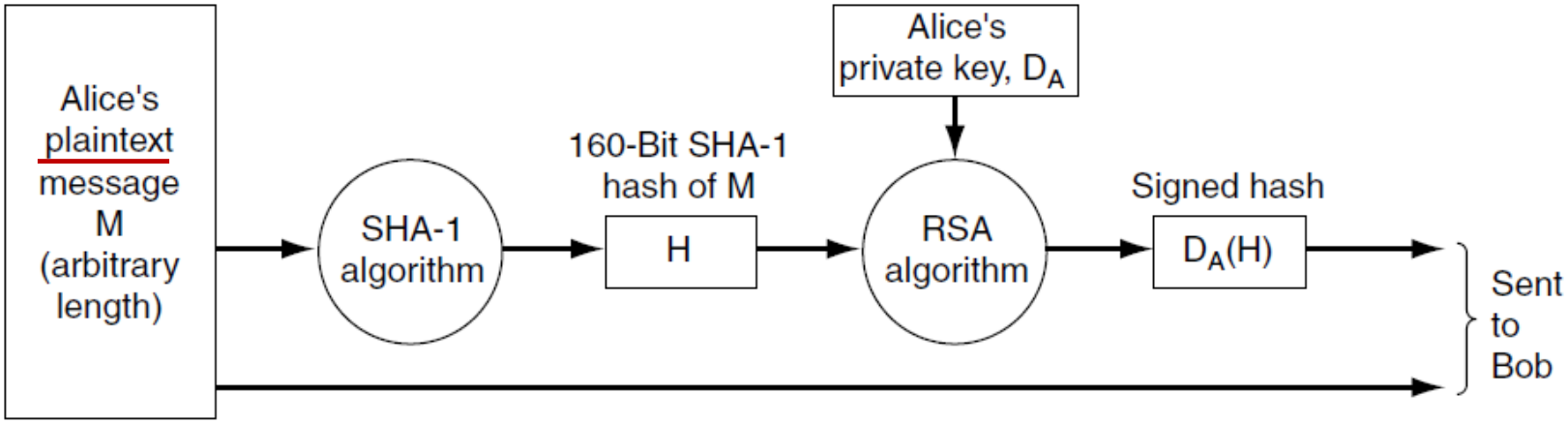


Figure 8-21. Use of SHA-1 and RSA for signing nonsecret messages.

SHA-1 (I)

- How SHA-1 works:
  - 1) It starts out by padding the message by adding a 1 bit to the end, followed by as many 0 bits as are necessary, but at least 64, to make the length a multiple of 512 bits. SHA-1 always pads the end of the message, no matter which endian machine is used (Big-endian or small-endian).

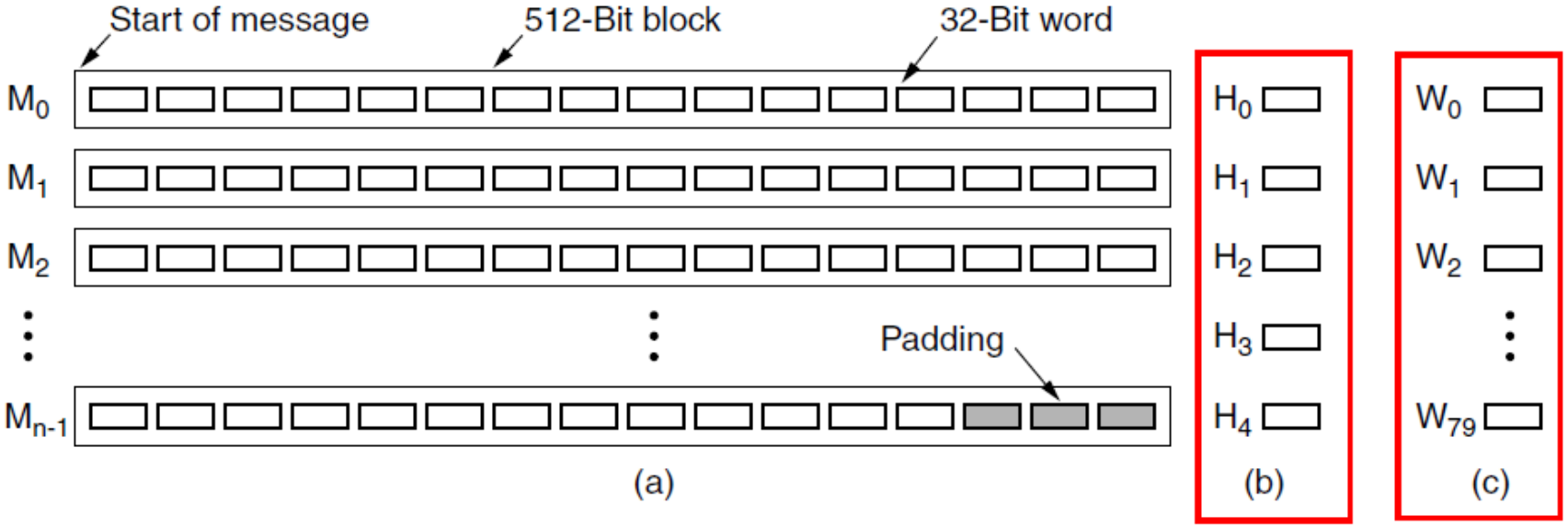


Figure 8-22. (a) A message padded out to a multiple of 512 bits. (b) The output variables. (c) The word array.

每个块block有16个words，每个word有32-bit。

SHA-1 (II)

- 2) Then a 64 bit number containing the message length before padding is ORed into the low-order 64 bits.
- 3) SHA-1 maintains five 32-bit variables H_0 through H_4, where the hash **accumulates**. They are initialized to constants specified in the standard.

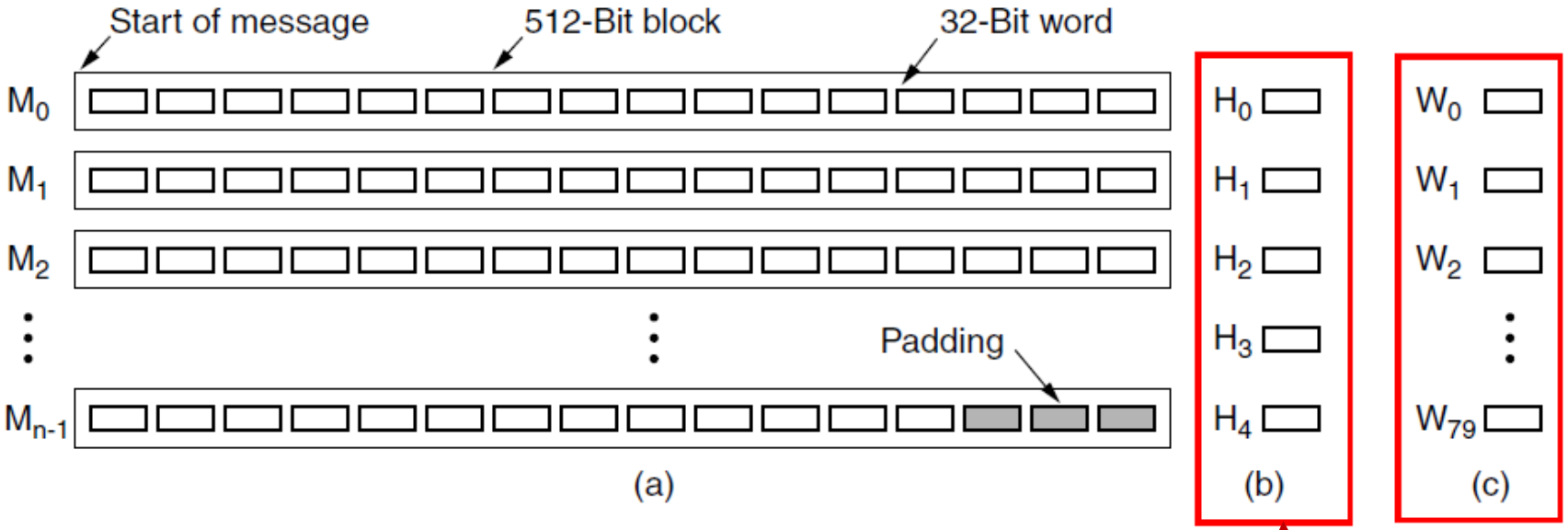


Figure 8-22. (a) A message padded out to a multiple of 512 bits. (b) The output variables. (c) The word array.

160bit output

SHA-1 (III)

- 4) Each of the blocks M_0 through M_{n-1} is processed in turn.
  - For the current block, the 16 words (each word is with 32-bit) are first copied into the start of an auxiliary 80-word array W (W_0, W_1, …, W_{15}). Then the other 64 words in W are filled in using the formula

$$W_i = S^1(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \quad (16 \le i \le 79)$$

where $S^b(W)$ represents the left circular rotation of the 32-bit word, W, by b bits.

  - Now 5 scratch variables, A through E, are initialized from H_0 through H_4, respectively (A = H_0, B = H_1, C = H_2, D = H_3, E = H_4).

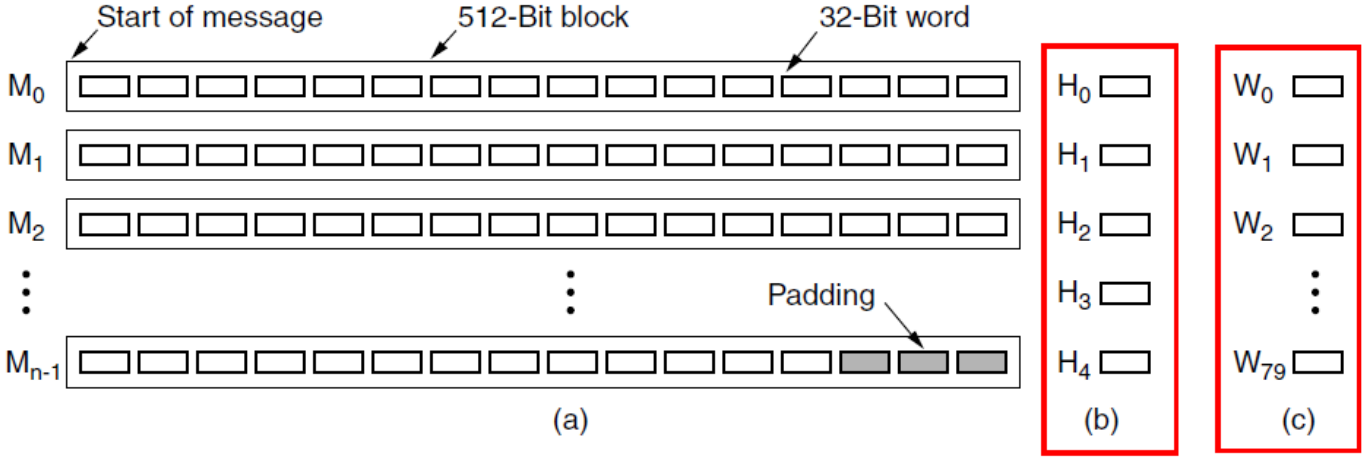


Figure 8-22. (a) A message padded out to a multiple of 512 bits. (b) The output variables. (c) The word array.

SHA-1 (IV)

The actual calculation can be expressed in pseudo-C as

```
for (i = 0; i < 80; i++) {
    temp = S^5(A) + f_i (B, C, D) + E + W_i + K_i;
    E = D;  D = C;  C = S^30(B);  B = A;  A = temp;
}
```

where the K_i constants are defined in the standard. The mixing functions f_i are defined as

$$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) \qquad (0 \le i \le 19)$$
$$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D \qquad (20 \le i \le 39)$$
$$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \qquad (40 \le i \le 59)$$
$$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D \qquad (60 \le i \le 79)$$

  - When all 80 iterations of the loop are completed, A through E are added to H_0 through H_4, respectively.

- Now that the first 512-bit block has been processed, the next one is started. The W array is reinitialized from the new block, but H is left as it was.
- When the last block has been finished, the 5 32-bit words in the H array are output as the 160-bit cryptographic hash.

SHA-1: Example (I) [4]

- 例子: 信息“abc”，’a’的ASCII码=97，’b’的ASCII码=98，’c’的ASCII码=99，
- 1) 将其转换为二进制位串之后为： 01100001 01100010 01100011。信息长度为24bit（这个长度值转换为16进制就为18）。
- 2)对转换后的位字符串进行补位操作：Sha-1算法标准规定，必须对消息摘要进行补位操作，即将输入的数据进行填充，使得数据长度对512求余的结果为448，填充比特位的最高位补一个1，其余的位补0，如果在补位之前已经满足对512取模余数为448，也要进行补位，在其后补一位1即可。总之，补位是至少补一位，最多补512位。

01100001 01100010 01100011 1 (后面补了423个0)，补位后的长度为448比特。
而后我们将补位操作后的信息摘要转换为十六进制
61626380 00000000 … 00000000 (有13个这样“00000000”)

SHA-1: Example (II) [4]

- 3) 附件长度值：在信息摘要后面附加64bit的信息，用来表示原始信息摘要的长度，在这步操作之后，信息报文便是512bit的倍数。

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018

- 4) 初始化缓存：一个160位MD缓冲区用以保存中间和最终散列函数的结果。它可以表示为5个32位的寄存器(H0,H1,H2,H3,H4)。初始化为：

H0 = 0x67452301
H1 = 0xEFCDAB89
H2 = 0x98BADCFE
H3 = 0x10325476
H4 = 0xC3D2E1F0

SHA-1: Example (III) [4]

- 5) 计算消息摘要：
  - 在计算报文之前我们还要做一些基本的工作，就是在我们计算过程中要用到的方法，或定义。
  - (1) 循环左移操作符$S^n(x)$, x是一个字，也就是32bit大小的变量，n是一个整数且$0 \le n \le 32$。
  - (2) 在程序中所要用到的常量，这一系列常量字k(0)、k(1)、...k(79)，将其以十六进制表示如下：

Kt = 0x5A827999 (0 <= t <= 19)
Kt = 0x6ED9EBA1 (20 <= t <= 39)
Kt = 0x8F1BBCDC (40 <= t <= 59)
Kt = 0xCA62C1D6 (60 <= t <= 79)

  - (3) 所要用到一系列函数

$$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) \quad (0 \le i \le 19)$$
$$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D \quad (20 \le i \le 39)$$
$$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \quad (40 \le i \le 59)$$
$$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D \quad (60 \le i \le 79)$$

SHA-1: Example (IV) [4]

- 5) 计算消息摘要：
  - (4) 计算：计算需要一个缓冲区，由5个32位的字组成，还需要一个80个32位字的缓冲区。第一个5个字的缓冲区被标识为A，B，C，D，E (A = H_0, B = H_1, C = H_2, D = H_3, E = H_4)。80个字的缓冲区被标识为W0, W1,..., W79

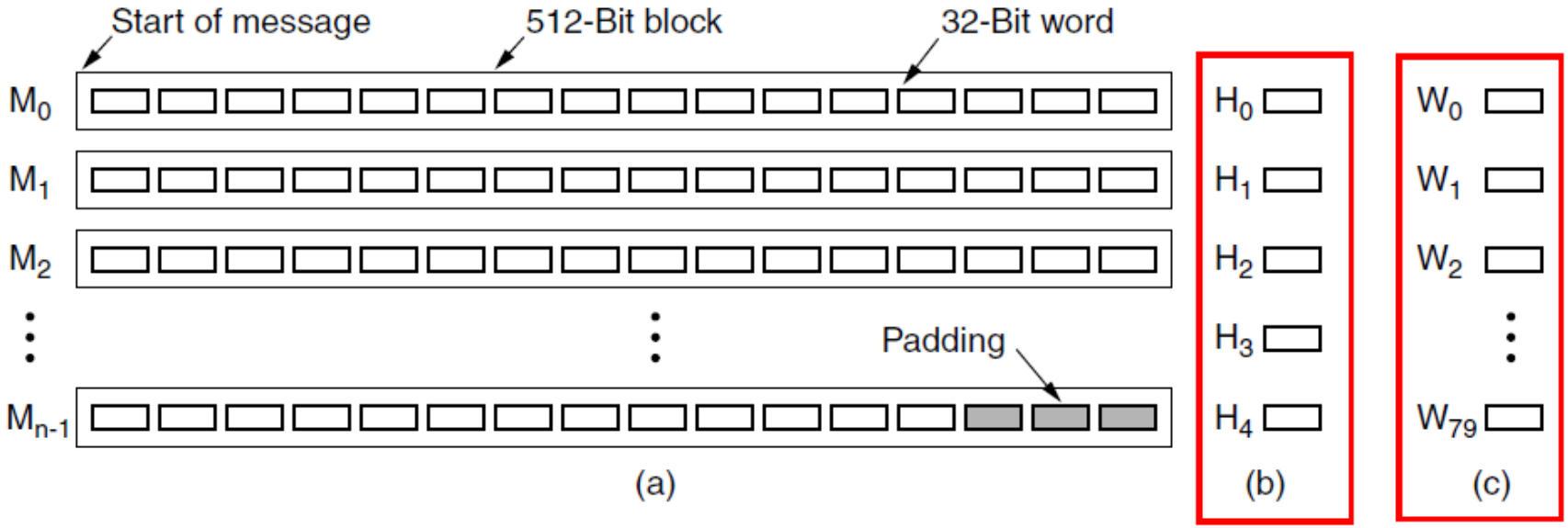


Figure 8-22. (a) A message padded out to a multiple of 512 bits. (b) The output variables. (c) The word array.

SHA-1: Example (V) [4]

- 5) 计算消息摘要：
- (4) 计算：另外还需要一个一个字的TEMP缓冲区。为了产生消息摘要，在第4部分中定义的16个字的数据块$M_0, M_1, ..., M_{n-1}$ 会依次进行处理，处理每个数据块M_i 包含80个步骤。
- 现在开始处理$M_0, M_1, ..., M_{n-1}$。为了处理 M_i，需要进行下面的步骤：

(1). 将 M_i 分成 16 个字 W0, W1, ... , W15, W0 是最左边的字
(2). 对于 t = 16 到 79 令 $W_t = S^1(W_{t-3}$ XOR W_{t-8} XOR W_{t-14} XOR $W_{t-16})$.
(3). 令 A = H0, B = H1, C = H2, D = H3, E = H4.
(4) 对于 t = 0 到 79，执行下面的循环

```
for (i = 0; i < 80; i++) {
    temp = S^5(A) + f_i (B, C, D) + E + W_i + K_i;
    E = D;  D = C;  C = S^30(B);  B = A;  A = temp;
}
```

(5). 令 H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.
在处理完所有的 Mn, 后，消息摘要是一个160位的字符串，以下面的顺序标识 H0 H1 H2 H3 H4.

SHA-2

- New version of SHA-1 have been developed that produce hashes of 224, 256, 384, and 512 bits. Collectively, these versions are called SHA-2.

- MD5 (Rivest, 1992)
  - The death knell is that different messages with the same hash.
  - No new systems should use it as part of their design.

The Birthday Attack*

- The idea for this attack comes from a traditional probability question: how many students do you need in a class before the probability of having two people with the same birthday exceeds 50%?
- Probability theory says it is just 23. With 23 students, we can form $(23 \times 22)/2 = 253$ different pairs, each of which as a probability of 1/365 of being a hit.
- Generally, there are $n(n-1)/2$ input pairs. If $n(n-1)/2 > k$ the chance of having a least one match is pretty good. Thus approximately, a match is likely for $n > (k)^{1/2}$.
- The birthday attack is that with two different plaintexts, but have the same message digests.

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security

Management of Public Keys (I)

- If Alice and Bob do not know each other, how do they get each other's public keys to start a communication process?
  - The obvious solution is to put your public key on your web page, but the following example says it does not work.

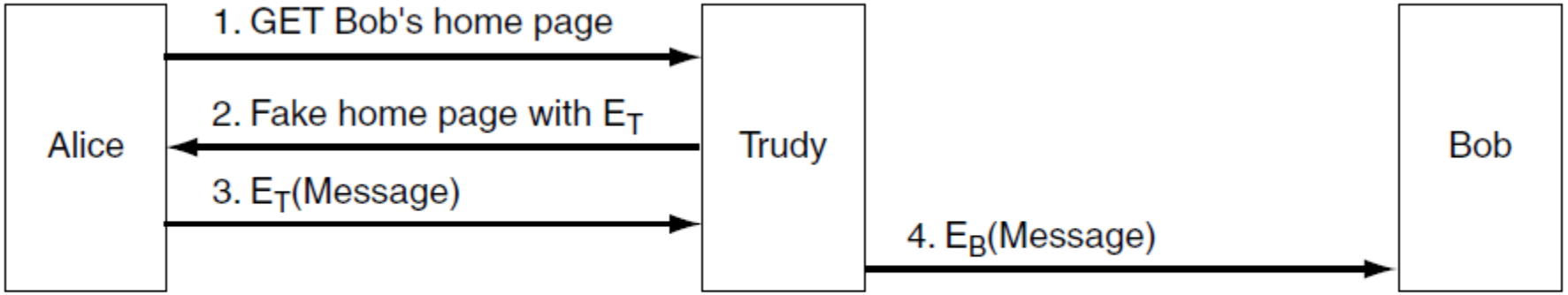


Figure 8-23. A way for Trudy to subvert public-key encryption.

Management of Public Keys (II)

- Some mechanism is needed to make sure that public keys can be exchanged securely.
  - CA (Certification Authority)
  - Standard X.509 (RFC5280, 格式标准而已)
  - Public key infrastructure

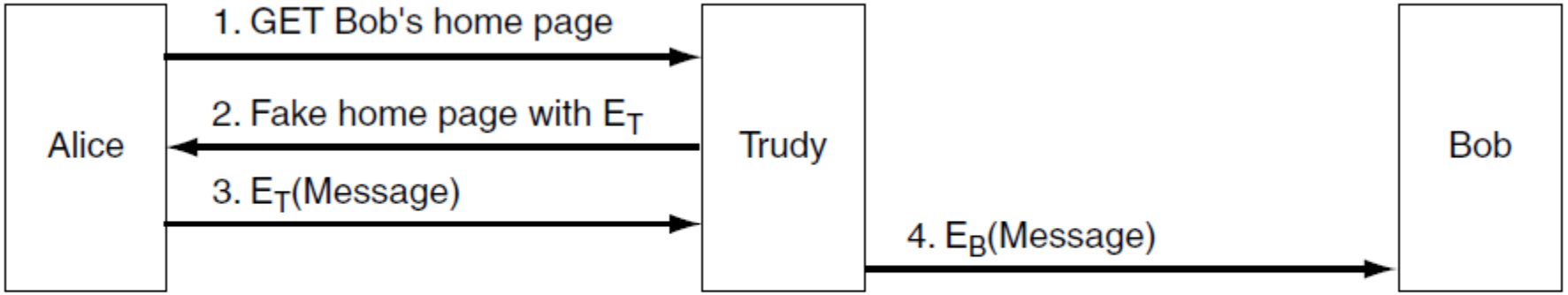
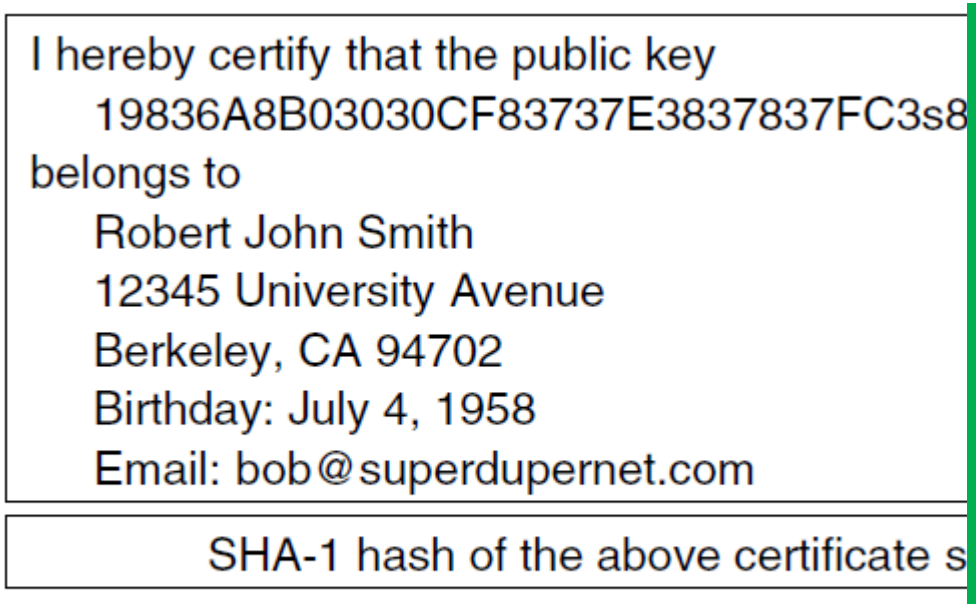


Figure 8-23. A way for Trudy to subvert public-key encryption.

Certificates

- An organization that certifies public keys is now called a **CA** (**Certification Authority**)
  - The fundamental job of a certificate is to build a public key to the name of a principal (individual, company, etc.)



If Trudy changed the certificate, then when Alice runs the SHA-1 algorithm on the certificate, she will get a hash that does not agree with the one she gets when she applies the CA's well known public key to the signature block. Since Trudy does not have the CA's private key, she has no way of generating a signature block that contains the hash of the modified Web page with her public key on it.

Figure 8-24. A possible certificate and its signed hash.

- The certificate and the signature block (the signed SHA-1 hash of the certificate, here the signed SHA-1 for integrity)

X.509: a standard for certificates*

- The primary fields in a certificate

Field	Meaning
Version	Which version of X.509
Serial number	This number plus the CA's name uniquely identifies the certificate
Signature algorithm	The algorithm used to sign the certificate
Issuer	X.500 name of the CA
Validity period	The starting and ending times of the validity period
Subject name	The entity whose key is being certified
Public key	The subject's public key and the ID of the algorithm using it
Issuer ID	An optional ID uniquely identifying the certificate's issuer
Subject ID	An optional ID uniquely identifying the certificate's subject
Extensions	Many extensions have been defined
Signature	The certificate's signature (signed by the CA's private key)

Figure 8-25. The basic fields of an X.509 certificate.

Public Key Infrastructures (I)

- Having a *single* CA to issue all the world's certificates obviously would not work.
  - It would collapse under the load and be a central point of failure as well.
- PKI (Public Key Infrastructure) has multiple components, including users, CAs, certificates, and directories.
  - What the PKI does is to provide a way of structuring these components and define standards for the various documents and protocols.
  - A particularly simple form of PKI is a hierarchy of CAs.

Public Key Infrastructures (II)

- The top level CA, the root, certifies second-level CAs, which we call RAs (Regional Authorities)
- When the root authorizes a new RA, it generates an X.509 certificate stating that it has approved the RA, includes the new RA's public key in it, signs it, and hands it to the RA. Similarly, when an RA approves a new CA, it produces and signs a certificate stating its approval and containing the CA's public key.

It is assumed that everyone knows the root's public key.

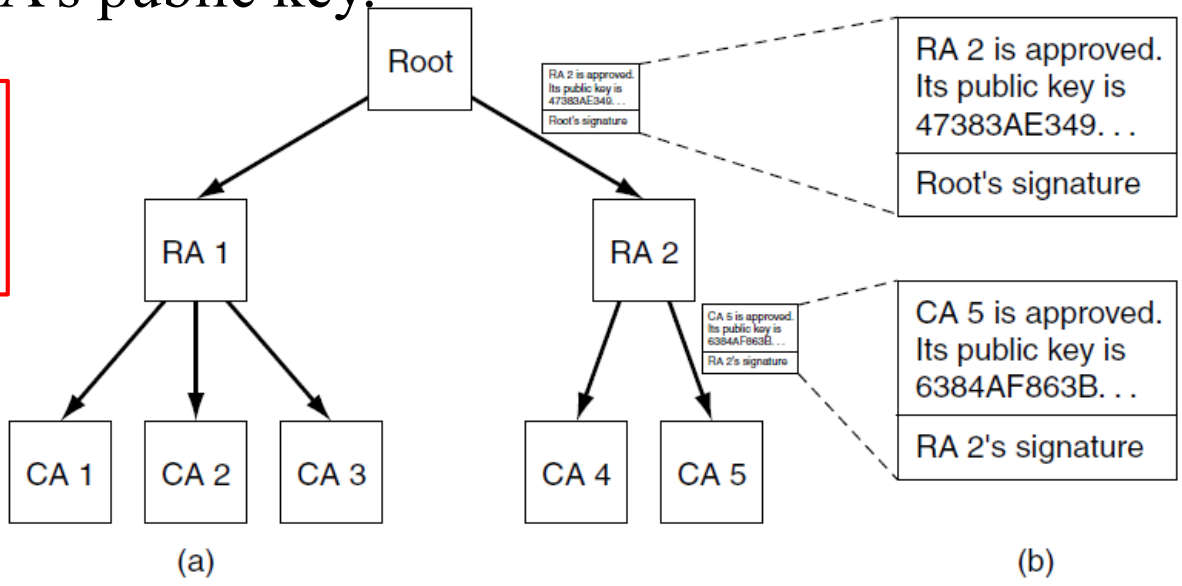
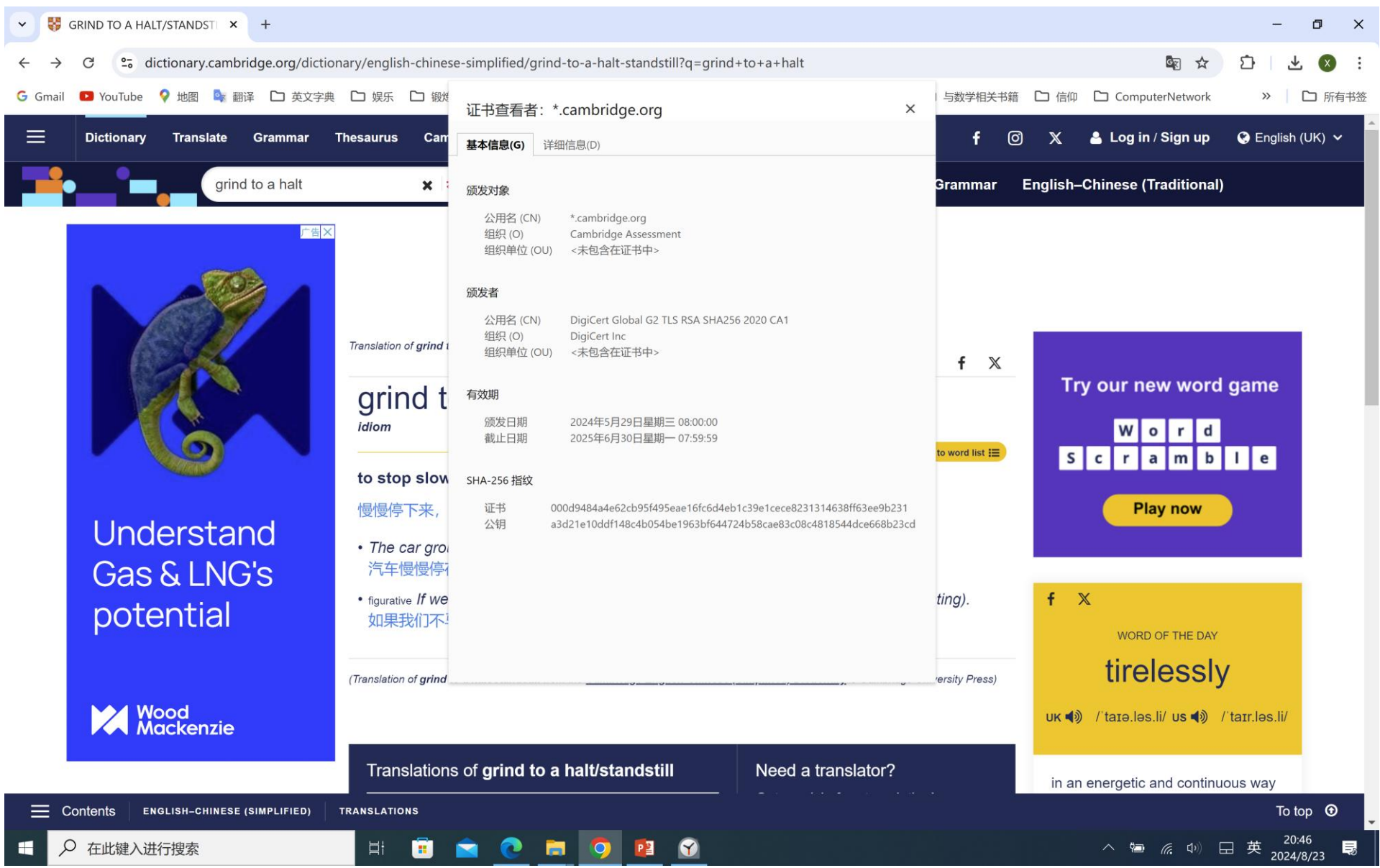


Figure 8-26. (a) A hierarchical PKI. (b) A chain of certificates.

Public Key Infrastructures (III)

- It is assumed that everyone knows the root's public key.
- A chain of certificates going back to the root is called **a chain of trust** or **a certification path**.
- Modern browsers come preloaded with the public keys for over 100 roots.
- Most browser allow users to inspect the root keys (usually in the form of certificates signed by the root) and delete any that seem shady.


GRIND TO A HALT/STANDST
dictionary.cambridge.org/dictionary/english-chinese-simplified/grind-to-a-halt-standstill?q=grind+to+a+halt
证书查看者：*.cambridge.org
基本信息(G)
详细信息(D)
颁发对象
公用名 (CN) *.cambridge.org
组织 (O) Cambridge Assessment
组织单位 (OU) <未包含在证书中>
颁发者
公用名 (CN) DigiCert Global G2 TLS RSA SHA256 2020 CA1
组织 (O) DigiCert Inc
组织单位 (OU) <未包含在证书中>
有效期
颁发日期 2024年5月29日星期三 08:00:00
截止日期 2025年6月30日星期一 07:59:59
SHA-256 指纹
证书 000d9484a4e62cb95f495eae16fc6d4eb1c39e1cece8231314638ff63ee9b231
公钥 a3d21e10ddf148c4b054be1963bf644724b58cae83c08c4818544dce668b23cd
Dictionary
Translate
Grammar
Thesaurus
Log in / Sign up
English (UK)
grind to a halt
English–Chinese (Traditional)
Understand Gas & LNG's potential
Wood Mackenzie
Try our new word game
Word Scramble
Play now
WORD OF THE DAY
tirelessly
in an energetic and continuous way
Translations of grind to a halt/standstill
Need a translator?
Contents
ENGLISH–CHINESE (SIMPLIFIED)
TRANSLATIONS
To top
在此键入进行搜索
20:46
2024/8/23

网页公钥的例子

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security

Outlines

- Applications
  - Communication security
    - IPsec
    - Firewalls
    - VPN
    - Wireless Security
  - Authentication protocols
  - Email security
  - Web security

IPsec (I)

- **IPsec** (**IP security**) described in RFCs 2401, 2402, 2406, and 2410, among others.
- The complete IPsec design is a framework for multiple services, algorithms, and granularities.
  - The reason for multiple services is that not everyone wants to pay the price for having all the services all the time, so the services are available a la carte (套餐).
    - The major services are secrecy, data integrity, and protection from replay attacks.
    - All of these are based on symmetric-key cryptography because high-performance is crucial.
  - The reason for multiple algorithms is that an algorithm that is now thought to be secure may be broken in the future. By making IPsec algorithm-in-dependent, the framework can survive even if some particular algorithm is later broken.
  - The reason for multiple granularities is to make it possible to protect a single TCP connection, all traffic between pair of hosts, or all traffic between a pair of secure routers, among other possibilities.

IPsec (II)

- IPsec is in the IP layer (the network layer), but it is connection oriented.
  - To have any security, a key must be established and used for some period of time.
- A “connection” in the context of IPsec is called an **SA** (**Security Association**).
  - An SA is a simplex connection between two endpoints and has a security identifier associated with it.
  - If secure traffic is needed in both directions, two security associations are required.
  - Security identifiers are carried in packets traveling on these secure connections and are used to look up keys and other relevant information when a secure packet arrives.

IPsec (III)

- IPsec has two principal parts:
  - The 1st part describes **two new headers** that can be added to packets to carry the security identifier, integrity control data, and other information.
    - The IPsec authentication header (Fig.8-27)
    - ESP (Encapsulating Security Payload) (Fig.8-28)
  - The other part, **ISAKMP** (**Internet Security Association and Key Management Protocol**), deals with establishing keys.
    - ISAKMP is a framework.
    - The main protocol for carrying out the work is **IKE** (**Internet Key Exchange**), RFC4306.

IPsec (IV)

- **IPsec** can be used in either of *two modes*:
  - In **transport mode**, the IPsec header is inserted just after the IP header. The protocol field in the IP header is changed to indicate that an IPsec header follows the normal IP header (before the TCP header).
    - The IPsec header contains security information, primarily the **SA identifier**, a new sequence number and possibly an integrity check of the payload.
  - In **tunnel mode**, the entire IP packet, header and all, is encapsulated in the body of a new IP packet with a completely new IP header.
    - Tunnel modes is useful when the tunnel ends at a location other than the final destination. In some cases, the end of the tunnel is *a security gateway machine*.
      - This is commonly the case for a VPN (Virtual Private Network).
      - In this mode, the security gateway encapsulates and decapsulates packets as they pass through the security gateway.
    - Tunnel mode is useful when a bundle of TCP connections is aggregated and handled as one encrypted stream because it prevents an intruder from seeing who is sending how many packets to whom.
      - Studying the flow patterns of packets, even if they are encrypted, is called **traffic analysis**. Tunnel mode provides a way to foil it to some extent.
    - The disadvantage of tunnel mode is that it adds an extra IP header, thus increasing packet size substantially. In contrast, transport mode does not affect packet size as much.

IPsec (V)

- 1. AH (Authentication Header) It provides **integrity checking** and **anitreplay security**, but no secrecy (i.e., no data encryption).
  - In IPv4, it is interposed between the IP header (including any options) and the TCP header.
  - In IPv6, it is just another extension header and is treated as such.
  - The payload may have to be padded out to some particular length for the authentication algorithm.
  - 1) The next header field is used to store the value that the IP Protocol field had before it was replaced with **51** to indicate that an AH header follows. In most cases, the code for **TCP** (**6**) will go here.

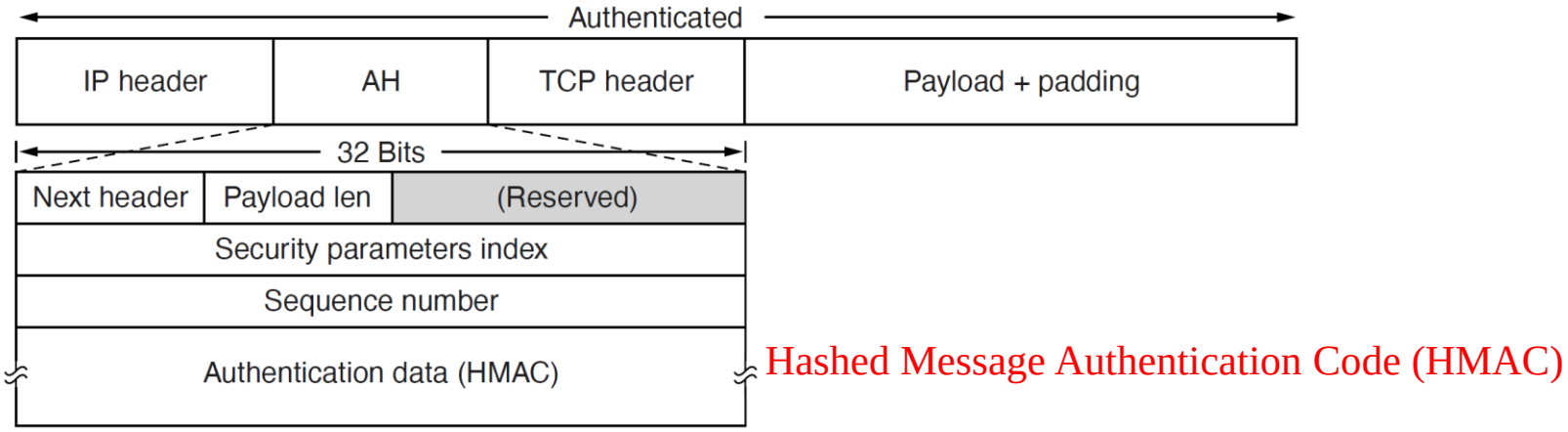
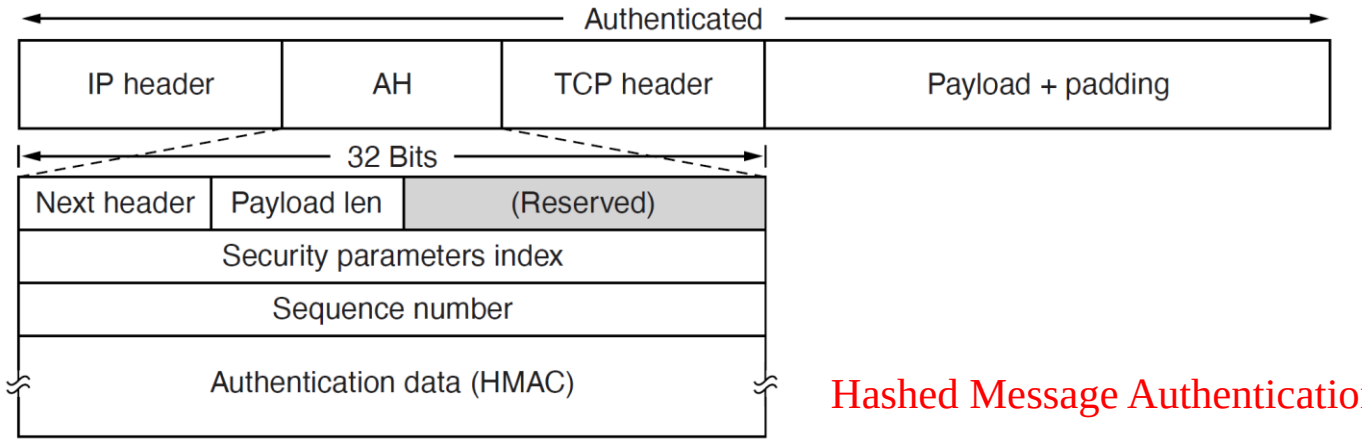


Figure 8-41. The IPsec authentication header in transport mode for IPv4.

IPsec (VI)

- 2) The **payload length** is the number of 32-bit words in the AH header minus 2.
- 3) The **security parameters index** is the connection identifier. It is inserted by the sender to indicate a particular record in the receiver's database. This record contains the shared key used on this connection and other information about the connection.
- 4) The **sequence number field** is used to number all the packets sent on an SA. Every packet gets a unique number, even retransmissions. The purpose of this field is to detect replay attacks. These sequence numbers may not wrap around. If all 2^{32} are exhausted, a new SA must be established to continue communication.



Hashed Message Authentication Code (HMAC)

Figure 8-41. The IPsec authentication header in transport mode for IPv4.

IPsec (VII)

- **5) Authentication data**, which is a variable-length field that contains *the payload's digital signature*.
  - When the SA is established, the two sides negotiate which signature algorithm they are going to use.
    - Normally, public-key cryptography is not used here because packets must be processed extremely rapidly and all known public-key algorithms are too slow.
  - Since IPsec is base on symmetric-key cryptography and the sender and the receiver negotiate a shared key before setting up an SA, the shared key is used in the signature computation.
  - AH handled only integrity checking, no data encryption.
    - The integrity check covers some of fields in the IP header, namely, those that do not change as the packet moves from router to router.
      - Not TTL but the IP source address

IPsec (VIII)

- **2. ESP** (Encapsulation Security Payload) Its use for both transport mode and tunnel mode. ESP handled only secrecy.
- The ESP header consists of two 32-bit words. They are **Security parameters index** and **Sequence number** fields that we saw in AH.
- Putting the HMAC at the end has an advantage in a hardware implementation: the HMAC can be calculated as the bits are going out over the network interface and appended to the end.
  - This is why Ethernet and other LANs have their CRCs in a trailer.
  - With AH, the packet has to be buffered and the signature computed before the packet can be sent, potentially reducing the number of packets/sec that can be sent.

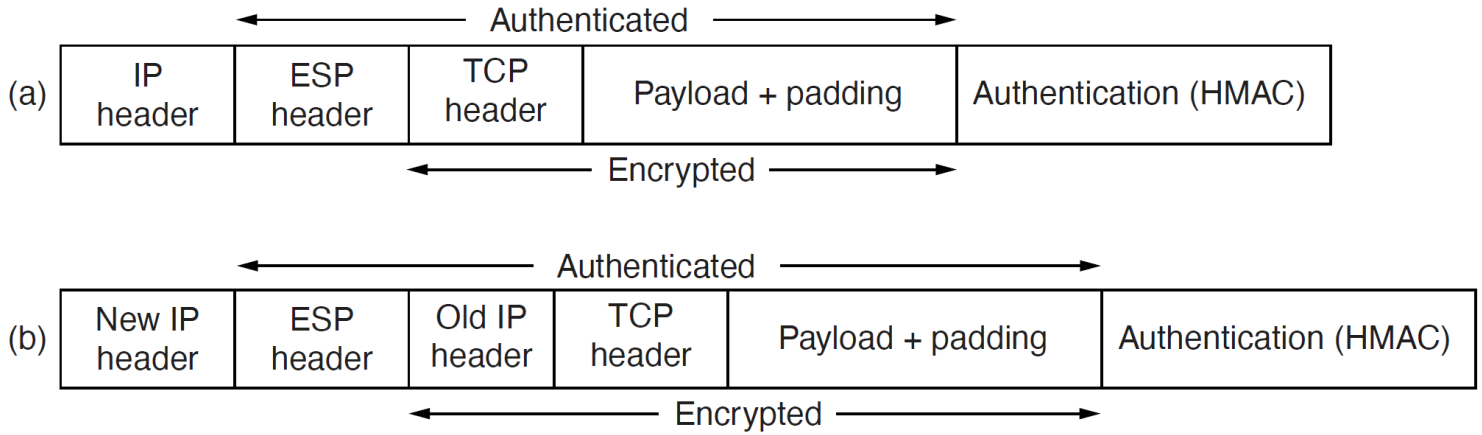


Figure 8-42. (a) ESP in transport mode. (b) ESP in tunnel mode.

Firewalls (I)

- The firewall acts as **a packet filter**.
  - It inspects each and every incoming and outgoing packet. Packets meeting some criterion described in rules formulated by the network administrator are forwarded normally. Those that failed the test are unceremoniously dropped.

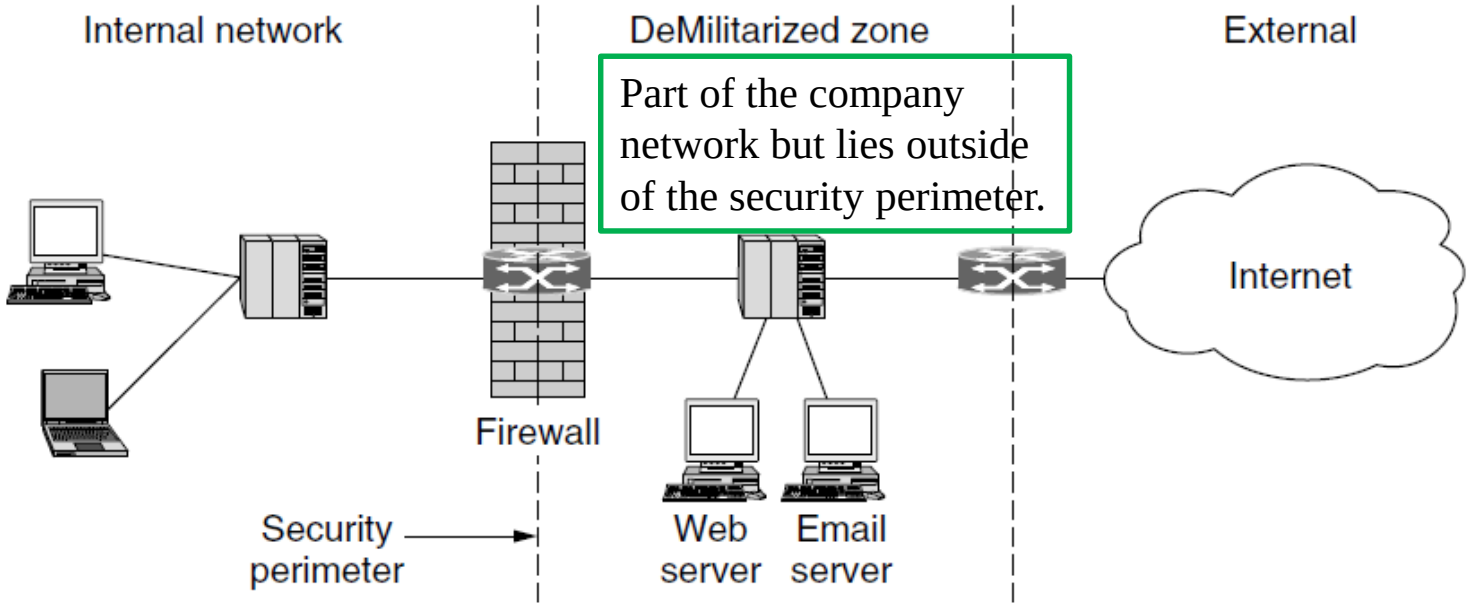


Figure 8-29. A firewall protecting an internal network.

Firewalls (II)

- The firewall acts as **a packet filter**, but violate the standard layering of protocols.
  - They are network layer devices, but they peek at the transport and application layers to do their filtering.

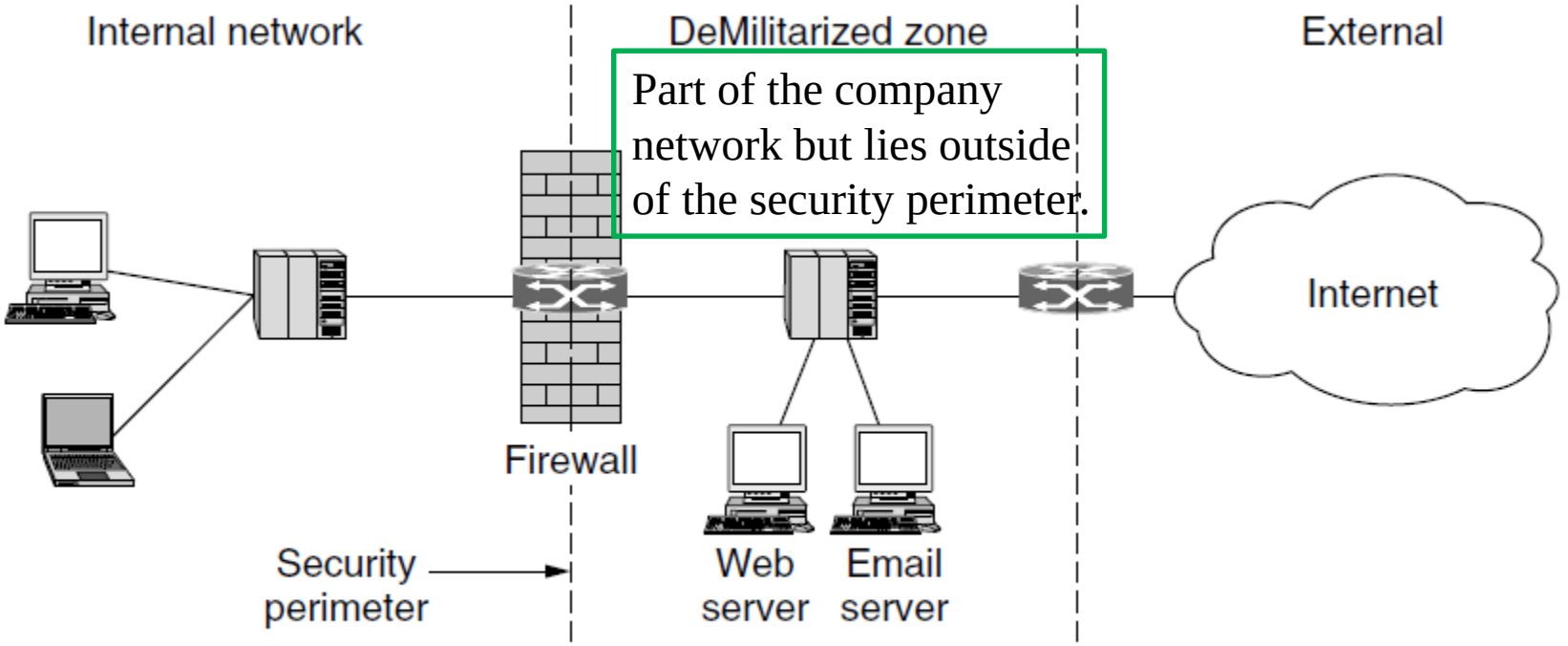


Figure 8-29. A firewall protecting an internal network.

Firewalls (III)

- Firewalls tend to *rely on standard port numbering conventions* to determine what kind of traffic is carried in a packet.
  - Some peer-to-peer applications select ports dynamically to avoid being easily spotted (and blocked).
  - Encryption with IPsec or other schemes hides higher-layer information from the firewall.
- Firewalls are often used in a layered defense.
  - For example, a firewall may guard the entrance to the internal network and each computer may also run its own firewall.
- The basic idea of a firewall is to prevent intruders from getting in and secret data from getting out. But
  - **DoS** (Denial of Service): Attacks in which the intruder's goal is to shut down the target rather than steal data. — *DNS DoS*
  - **DDoS** (Distributed Denial of Service) attack

Virtual Private Networks (VPN)

- A network build up from company computers and leased telephone lines is called a **private network**.
  - Expensive
- VPNs are **overlay networks** on top of public networks (i.e. the Internet) but with most of the properties of private networks.
  - To equip each office with a **firewall** and create **tunnels** through the Internet between all pairs of offices
    - flexible

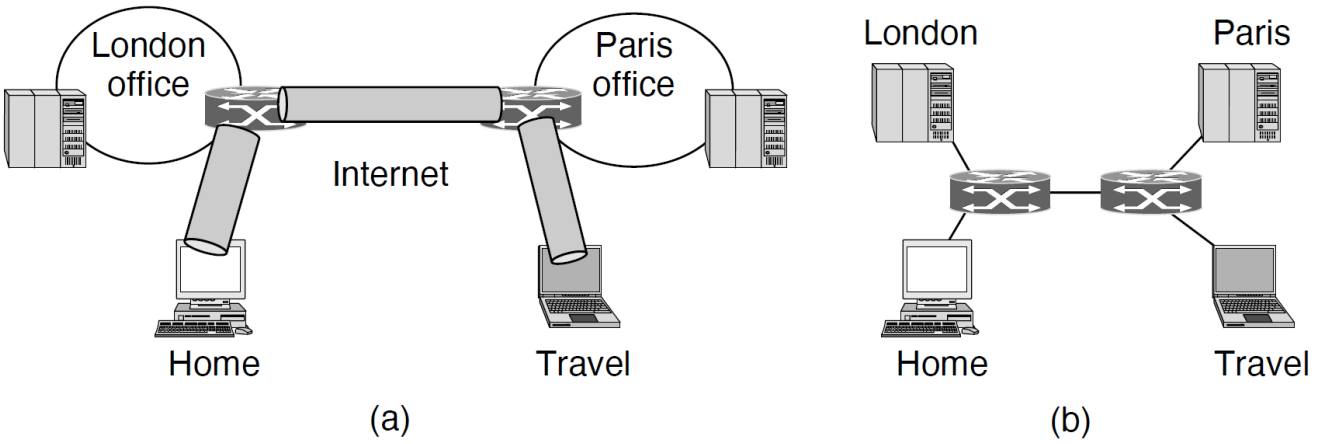


Figure 8-43. (a) A virtual private network. (b) Topology as seen from the inside.

Virtual Private Networks (VPN)

- When the system is brought up, each pair of firewalls has to negotiate the parameters of its SA (Security Association), including the services, modes, algorithms, and keys.
- If IPsec is used for tunneling, it is possible to aggregate all traffic between any two pairs of officers onto a single authenticated, encrypted SA, thus providing integrity control, secrecy, and even considerable immunity to traffic analysis.

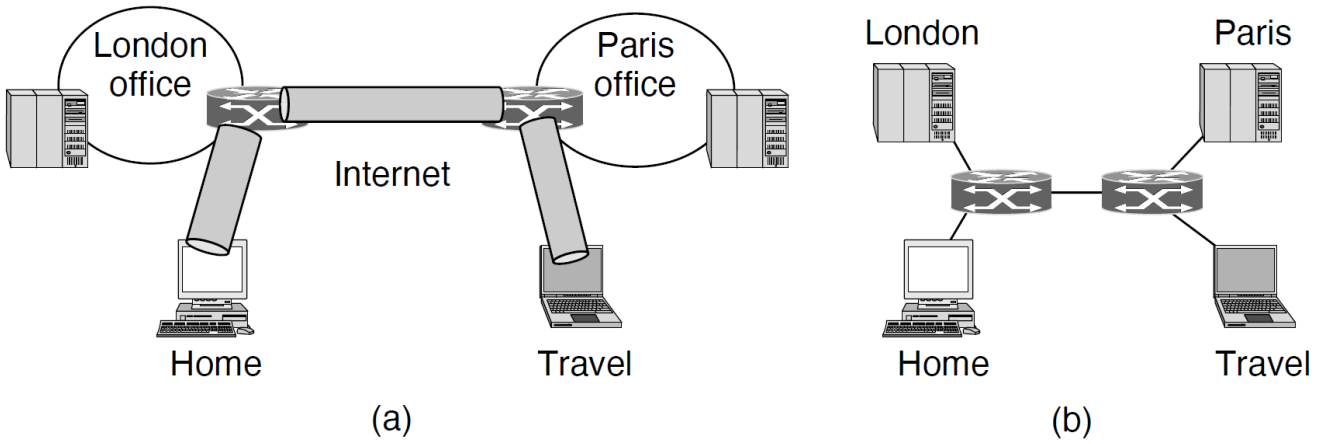


Figure 8-43. (a) A virtual private network. (b) Topology as seen from the inside.

Virtual Private Networks (VPN)

- Firewalls, VPNs, and IPsec with ESP in tunnel mode are a natural combination and widely used in practice.
- Another approach is to have the ISP set up the VPN.
  - MPLS (MultiProtocol Label Switching, Chapter 5), paths for the VPN traffic can be set up across the ISP network between the company offices.

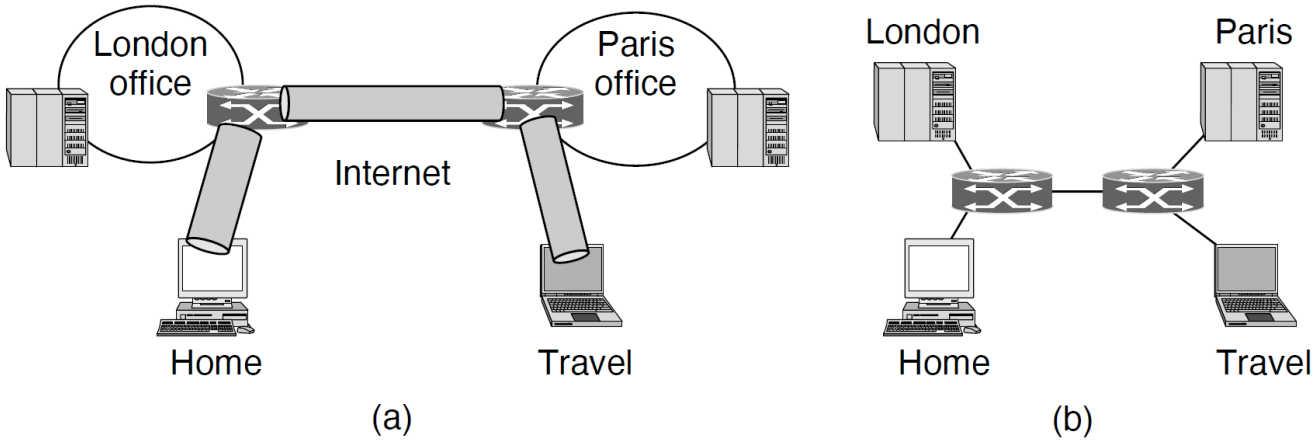


Figure 8-43. (a) A virtual private network. (b) Topology as seen from the inside.

Wireless Security

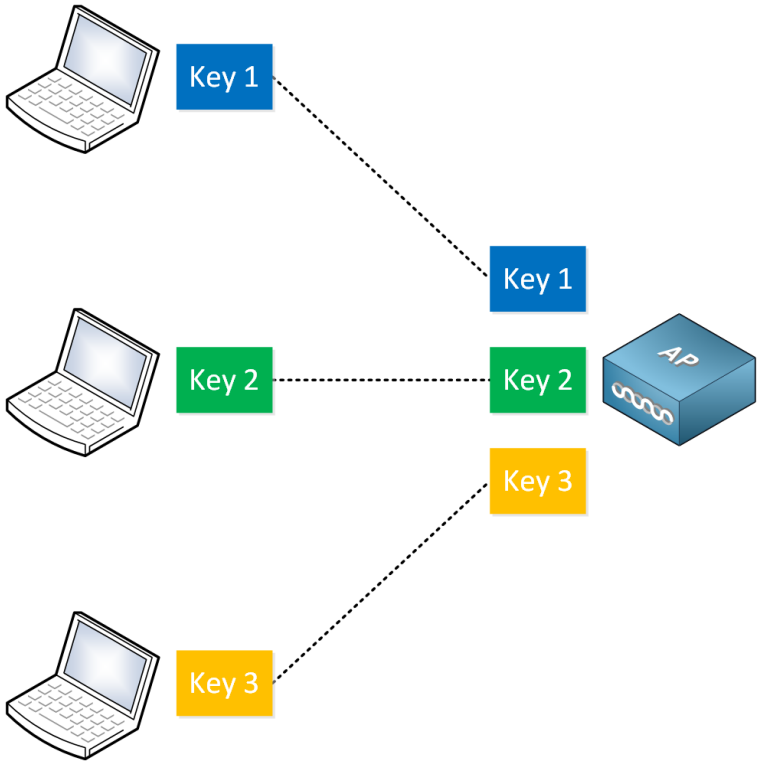
- Wireless security is important because we use wireless signals for transmission that go everywhere. Everyone can listen in, and you can’t detect when someone does.
- The first IEEE 802.11 standard had Wired Equivalent Privacy (WEP), which, as it turns out, was completely broken and had many vulnerabilities.
- The 802.11i amendment introduced a new security framework known as **Wi-Fi Protected Access** (**WPA**).

Wireless Security

- Instead of using a single key for encryption, WPA uses **a key hierarchy** with many keys for encryption and integrity checks of unicast and multicast/broadcast traffic.
- WPA uses a 4-way handshake for authentication and to create all required keys.
- With a wireless network, the wireless client is the supplicant, and the Access Point (AP) is the authenticator.
  - In wireless 802.11 terminology, a wireless client is officially called a station (STA).

Pairwise Keys

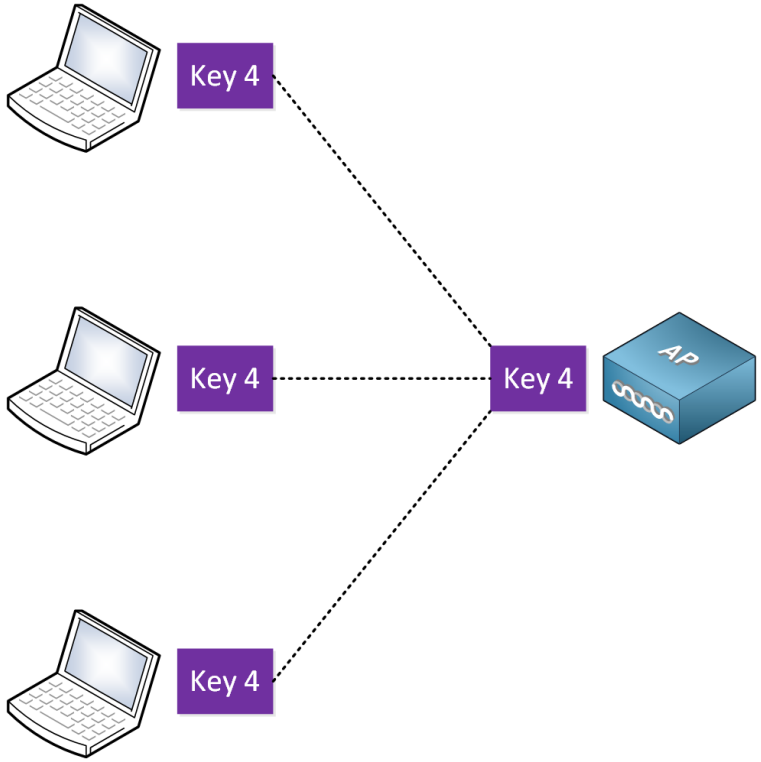
- An AP has many wireless clients. Unicast traffic between a wireless client and the AP has to be private. You don't want one client to be able to decrypt traffic between another wireless client and the AP. This is why you should have **different keys between each wireless client and AP**.



The AP has multiple pairwise keys, one for each associated wireless client.

Group Keys

- There is also broadcast and multicast traffic. All wireless clients should be able to encrypt and decrypt this traffic, so we need **a shared key**.

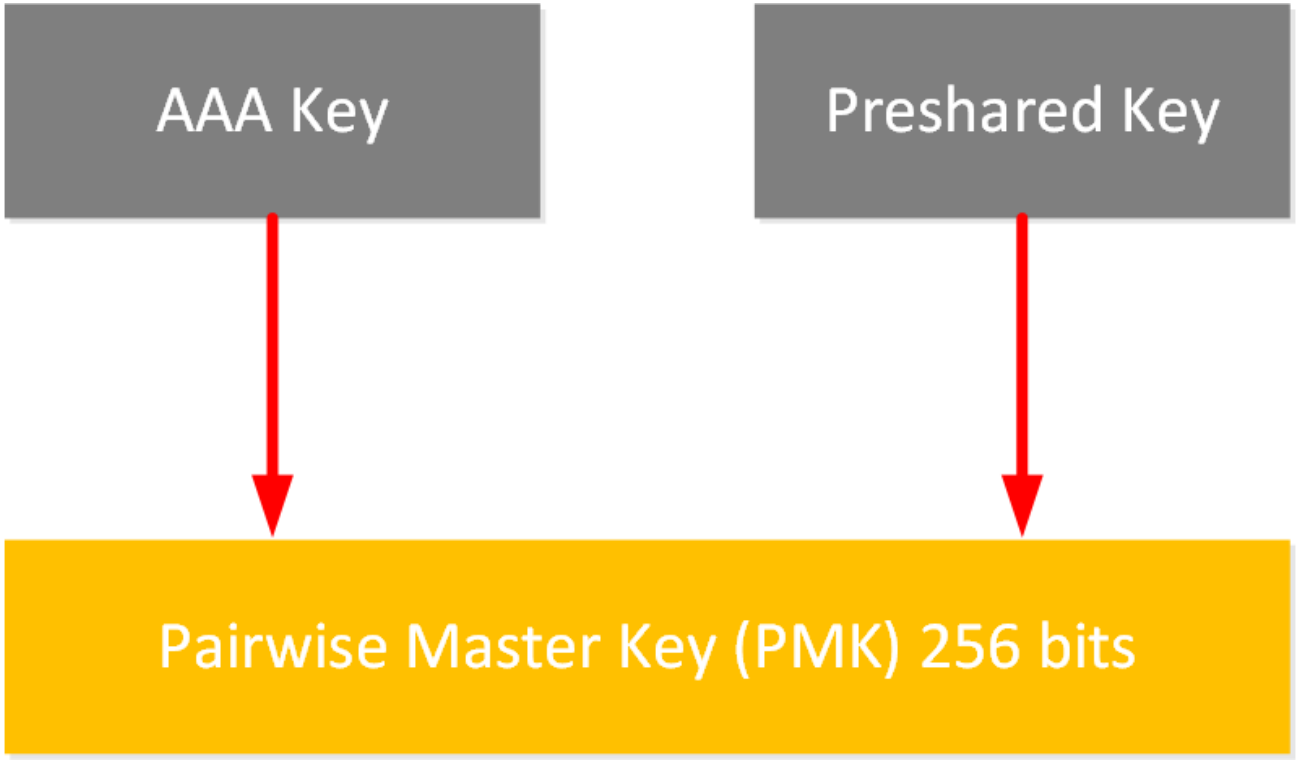


All associated wireless clients of the AP have the same key. We call this a **group key**.

Key Derivation

- Key derivation is a process used in cryptography to generate one or more cryptographic keys from one or more values, such as keys or a passphrase.
  - A Key Derivation Function (KDF) is a specific algorithm used for key derivation.

Pair-wise Master Key (PMK)

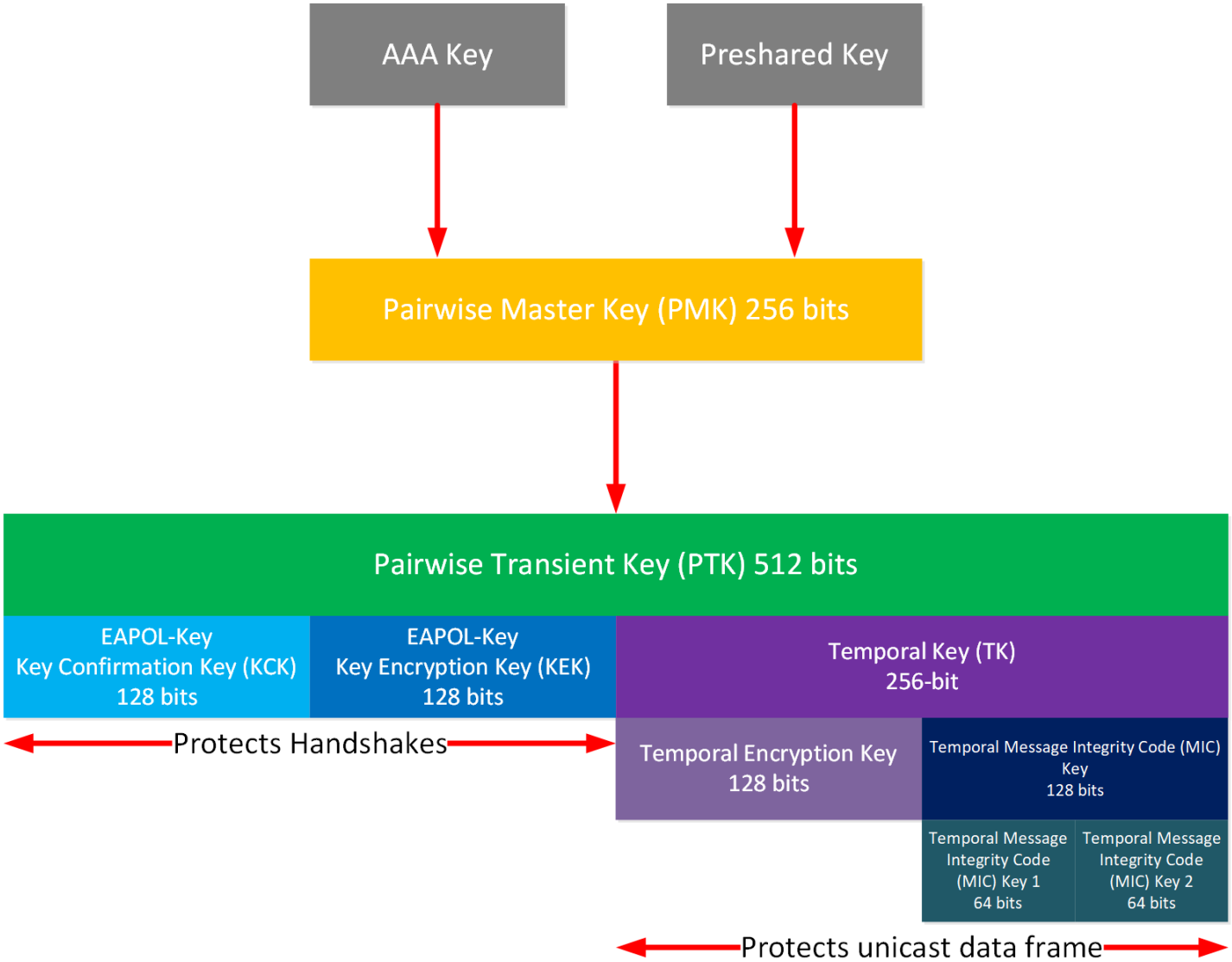


To generate pairwise master key can be a password (for personal network, but not scalable), or Authentication, Authorization, and Accounting (AAA) (for WPA-enterprise)

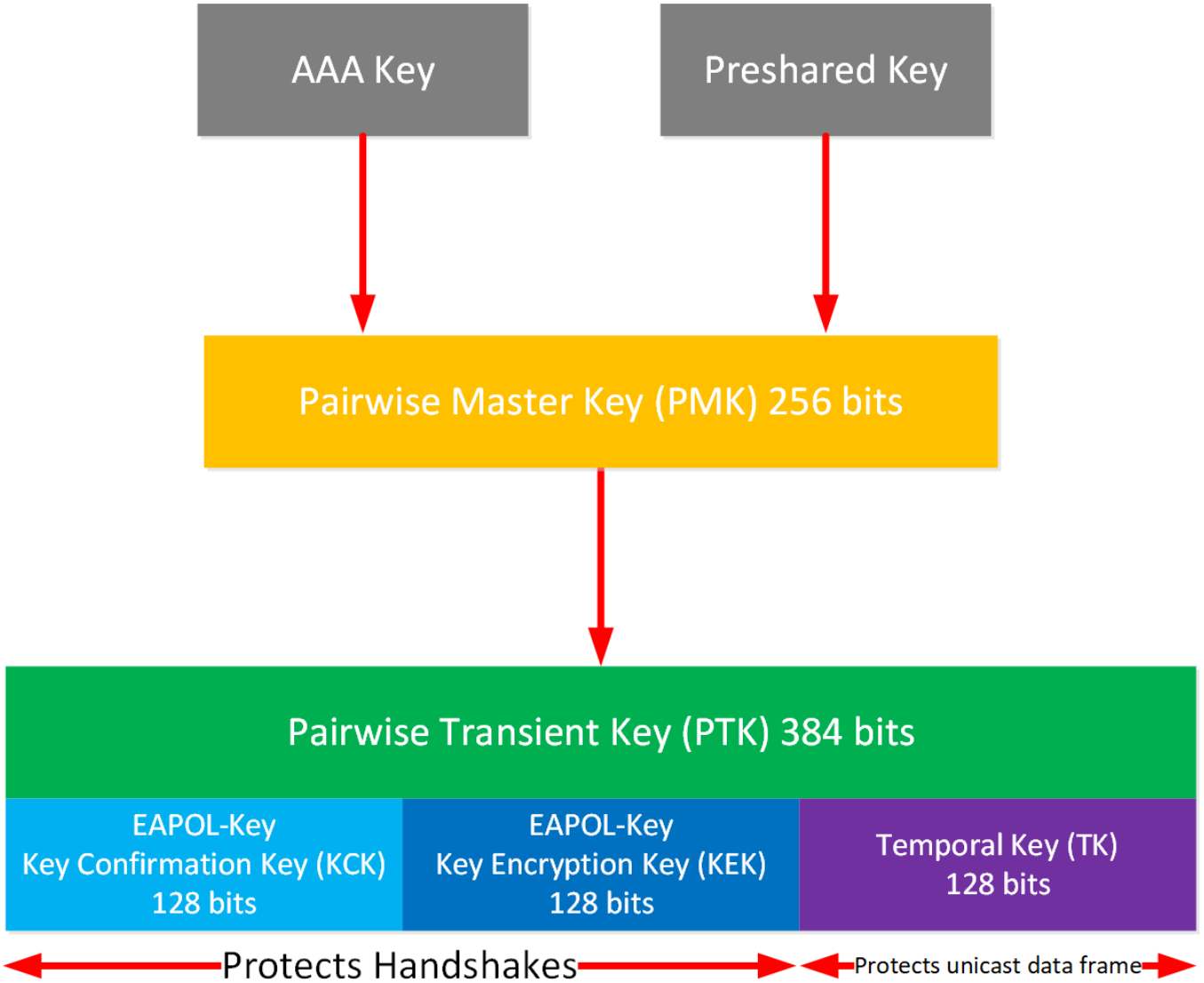
Pairwise Transient Key (PTK)

- The PTK is derived by combining these attributes
  - PMK
  - Nonce_{AP}
  - Nonce_{C}
  - AP MAC address
  - Client MAC address
- These attributes are inserted in a pseudo-random function (PRF); the output is the PTK.
- The PTK is used to protect two items:
  - The WPA 4-way handshake
  - User data
- The PTK is partitioned into multiple keys, and TKIP and AES-CCMP use different parts of the key differently.

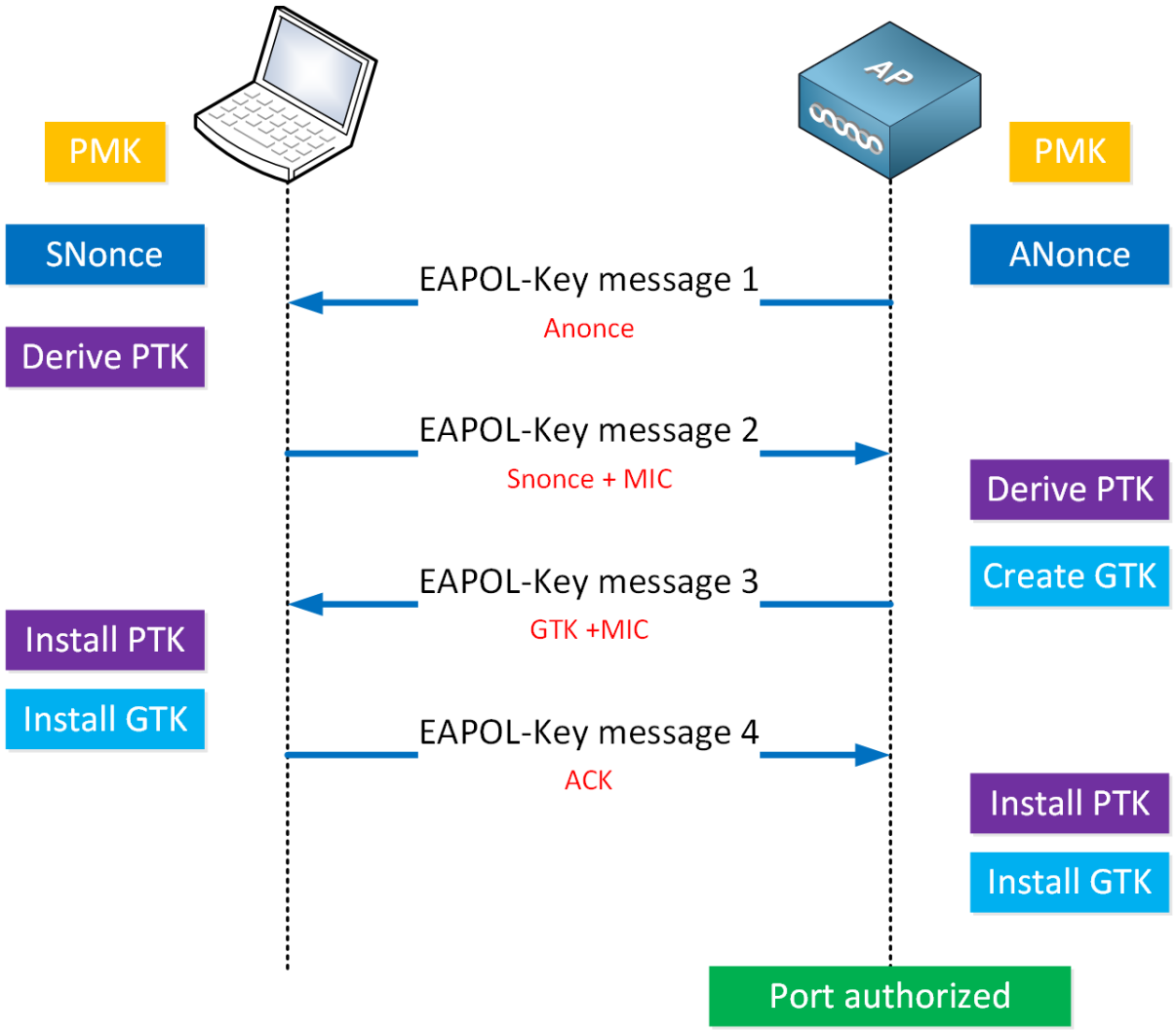
Temporal Key Integrity Protocol (TKIP)



AES-CCMP


AAA Key
Preshared Key
Pairwise Master Key (PMK) 256 bits
Pairwise Transient Key (PTK) 384 bits
EAPOL-Key
Key Confirmation Key (KCK)
128 bits
EAPOL-Key
Key Encryption Key (KEK)
128 bits
Temporal Key (TK)
128 bits
Protects Handshakes
Protects unicast data frame

802.11i Key Setup – 4-way Handshake



Once the 4-way handshake is complete, the wireless client and access point (AP) have a secure connection, and all traffic will be encrypted.

4-way Handshake (Message 1)

- The AP generates the $Nonce_{AP}$ value and sends message one to the client, which includes the $Nonce_{AP}$.
  - This message is unencrypted and has no integrity check.
  - If someone messes with this message, the handshake will fail and that's it.
- The client receives message one and derives the Pairwise Transient Key (PTK)
  - The client already had:
    - Pairwise Master Key (PMK)
    - $Nonce_{C}$
    - The client MAC address
  - The client received this in message one
    - $Nonce_{AP}$
    - The AP MAC address

4-way Handshake (Message 2)

- The client sends message two with $Nonce_C$ and a Message Integrity Check (MIC) value
  - This message is unencrypted, but it does have a MIC to prevent tampering.
  - We use the Key Confirmation Key (KCK) to calculate the MIC.

- The AP receives message two from the client and does this:
  - Derive the PTK (Pairwise Transient Key), which is possible because we now have the client $Nonce_C$.
  - Use KCK to generate the MIC.
  - Compare received MIC with calculated MIC.
    - If the MIC is the same, this proves that the client and AP both have the same PMK.

- Now we are at 50% of the 4-way handshake, and both sides have derived the PTK. We have not encrypted anything yet. The AP and client both have the keys, but we don’t install them yet.

4-way Handshake (Message 3)

- The AP sends message three that includes
  - Key installation request: this will ask the client to install the keys.
  - MIC
  - Group Transient Key (GTK)
- The client receives message three and does this:
  - Compare received MIC with calculated MIC:
    - If the MIC is the same, this proves that the client and AP both have the same PMK.

4-way Handshake (Message 4)

- The client installs the PTK and GTK and sends message four:
  - This message is unencrypted but includes a MIC.
  - This message includes an ACK and completes the 4-way handshake.
  - This indicates that the client will install the keys and use encryption from now on.
- The client sends an EAPOL-Key to confirm it has installed the keys.
- The AP receives message four and does this:
  - Compares received MIC with calculated MIC.
    - If the MICs are the same, install the PTK and GTK.

Outlines

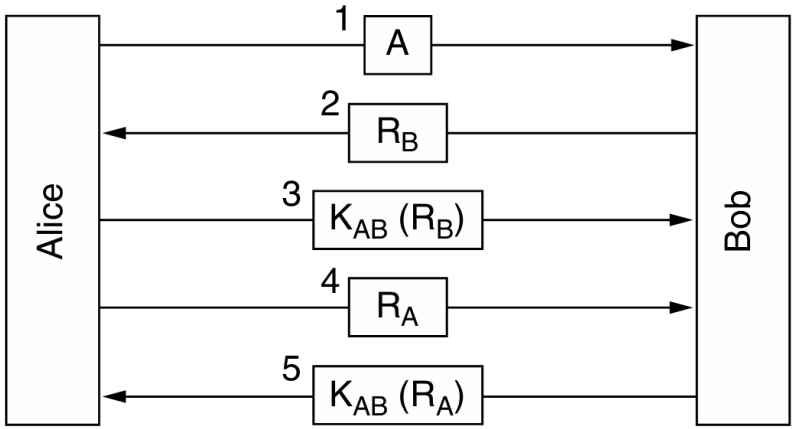
- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security

Authentication Protocols

- Authentication is the technique by which a process verifies that its communication partner is who it is supposed to be and not imposter.
  - **Authentication** (验证) vs. **Authorization** (授权)
  - Authentication deals with the question of whether you are actually communicating with a specific process.
  - Authorization is concerned with what the process is permitted to do.
  - For example, say a client process contacts a file server and says: “I am Scott’s process and I want to delete the file cookbook.old”.
    - Is this actually Scott’s process (authentication)?
    - Is Scott allowed to delete cookbook.old (authorization)?

Authentication based on a shared secret key (I)

- Assume that Alice and Bob already share a secret key K_{AB}.
- One party sends **a random number** to the other, who then transforms it in a special way and returns the result.
  - Such protocols are called **Challenge-response** protocols
  - This protocol contains five messages.



A, *B* are the identities of Alice and Bob.
R_i's are the challenges, where *i* identifies the challenger.
K_i's are keys, where *i* indicates the owner.
K_S is the session key.

Figure 8-32. Two-way authentication using a challenge-response protocol.

Authentication based on a shared secret key (II)

- A shorten two-way authentication protocol as illustrated in Fig 8-33.
  - Is this new protocol an improvement over the original one?
- Under certain circumstances, Trudy can defeat this protocol by using what is known as **a reflection attack**.
  - Trudy can break it if it is possible to **open multiple sessions** with Bob at once.

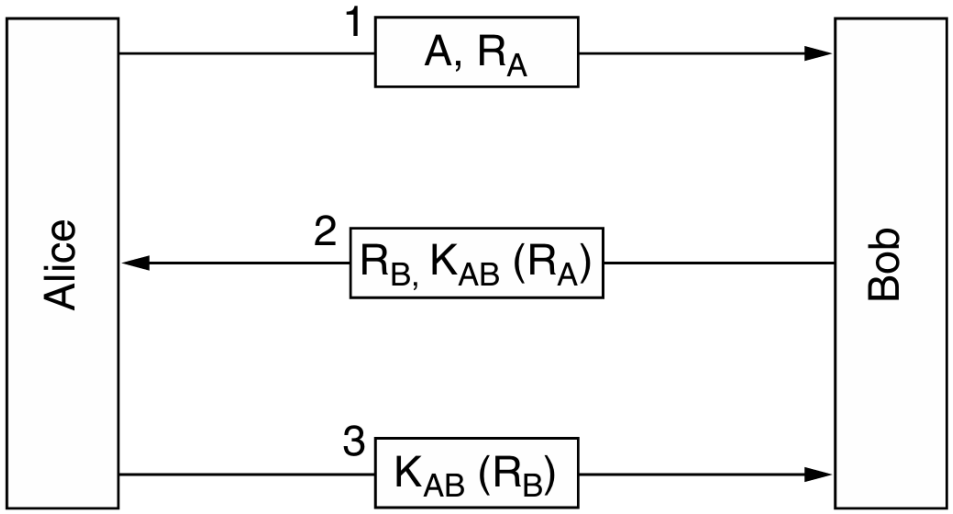


Figure 8-33. A shortened two-way authentication protocol.

Authentication based on a shared secret key (III)

- Trudy's **reflection attack** is shown in Fig. 8-34.
  - Trudy can open a second session with message 3, supplying the R_B taken from message 2 as her challenge.

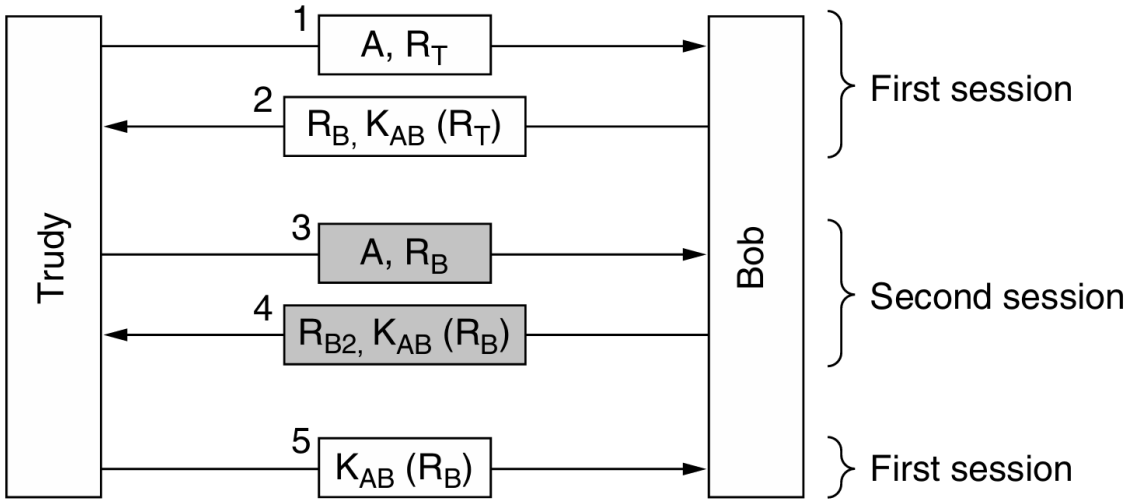


Figure 8-34. The reflection attack.

中间两个灰色交换信息中：Trudy用第一个session中R_B作为自己的随机数，然后Bob会用他与Alice共有的Secret Key: K_{AB}来加密R_B发给Trudy。然后message 5中Trudy就用$K_{AB}(R_B)$来回复Bob的message 2使Bob相信Trudy就是Alice。

Authentication based on a shared secret key (IV)

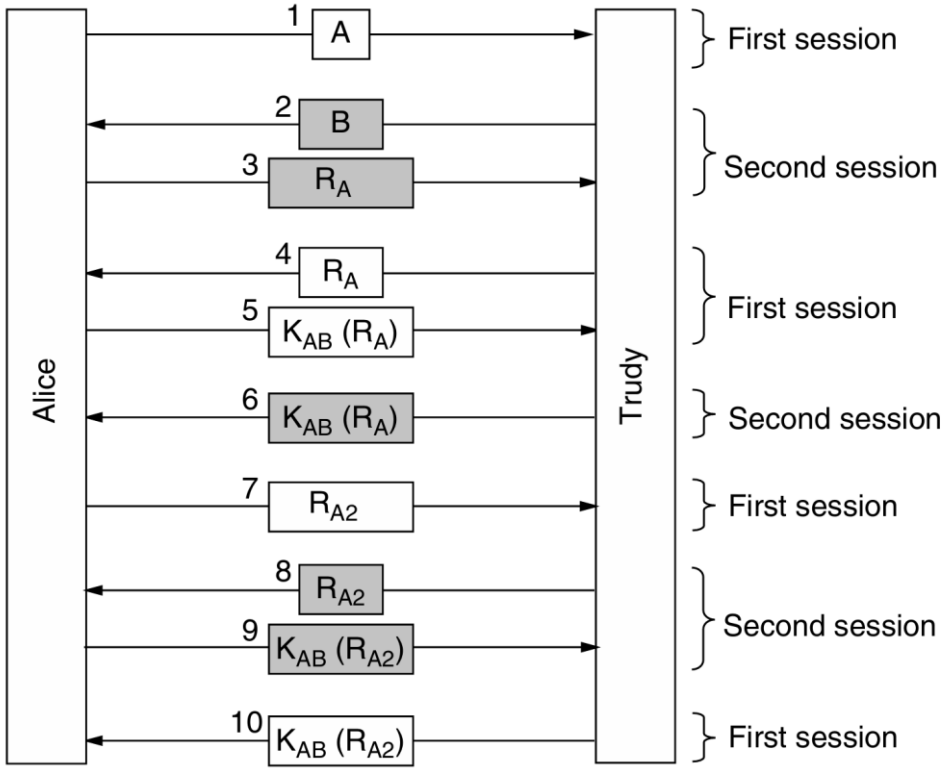
- The 4 general rules often help the designer avoid common pitfalls

1. Have the initiator prove who she is before the responder has to. This avoids Bob giving away valuable information before Trudy has to give any evidence of who she is.
2. Have the initiator and responder use different keys for proof, even if this means having two shared keys, K_{AB} and K'_{AB}.
3. Have the initiator and responder draw their challenges from different sets. For example, the initiator must use even numbers and the responder must use odd numbers.
4. Make the protocol resistant to attacks involving a second parallel session in which information obtained in one session is used in a different one.

If even one of these rules is violated, the protocol can frequently be broken.

Authentication based on a shared secret key (V)

- What would happen if Alice is a general-purpose computer that also accepted multiple sessions?
  - This time Trudy intercept the messages from Alice, claiming to be "Bob".



Trudy uses message 3 from Alice as the challenge to Alice (以子之矛攻子之盾)

Figure 8-35. A reflection attack on the protocol of Fig. 8-32.

Authentication based on a shared secret key (VI)

- The **HMAC** (Hashed Message Authentication Code) is formed by building a data structure consisting of Alice's nonce, Bob's nonce, their identities, and the shared secret key K_{AB}. This data structure is then **hashed** into the HMAC, for example, using SHA-1.
  - Trudy cannot subvert this protocol. Because she cannot force either party to encrypt or hash a value of her choice, as happened in Fig.8-34 and Fig.8-35.

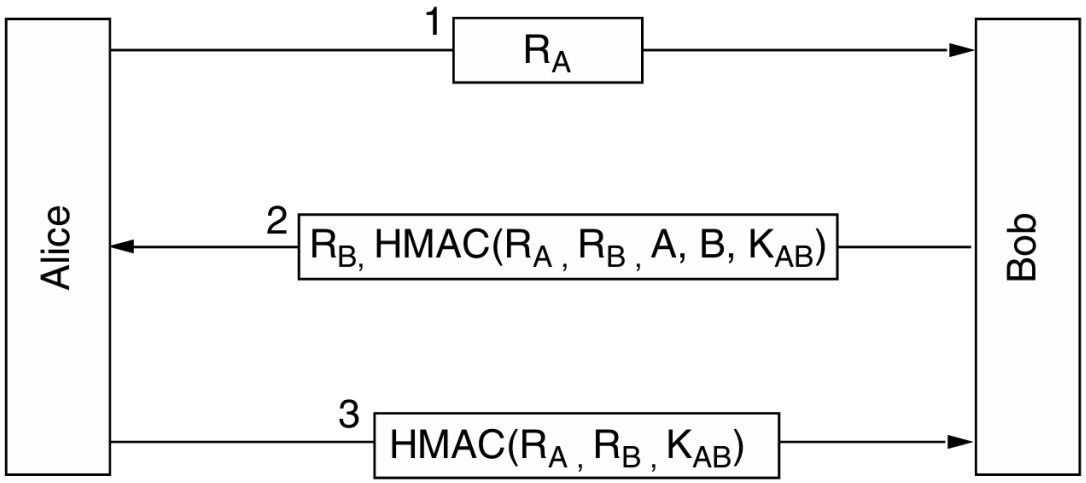


Figure 8-36. Authentication using HMACs.

Establishing a Shared Key: the Diffie-Hellman Key Exchange (I)

- The protocol that allows **strangers** to establish a shared secret key.
- The Diffie-Hellman Key exchange (Diffie and Hellman, 1976)
  - The two parties have to agree on two large numbers, n and g, where n is a prime, $(n - 1)/2$ is also a prime, and certain conditions apply to g.
  - One party (Alice) picks a large number x, and keeps it secret. Similarly, the other party (Bob) picks a large secret number y.
  - Alice initiates the key exchange protocol by sending Bob a message containing $(n, g, g^x \bmod n)$.
  - Bob responds by sending Alice a message containing $g^y \bmod n$.
  - Alice raises the number Bob sent her to the xth power modulo n to get $(g^y \bmod n)^x \bmod n$. Bob performs a similar operation to get $(g^x \bmod n)^y \bmod n$.
  - By **the laws of modular arithmetic**, both calculations yield $g^{xy} \bmod n$ (the shared key).

Establishing a Shared Key: the Diffie-Hellman Key Exchange (II)

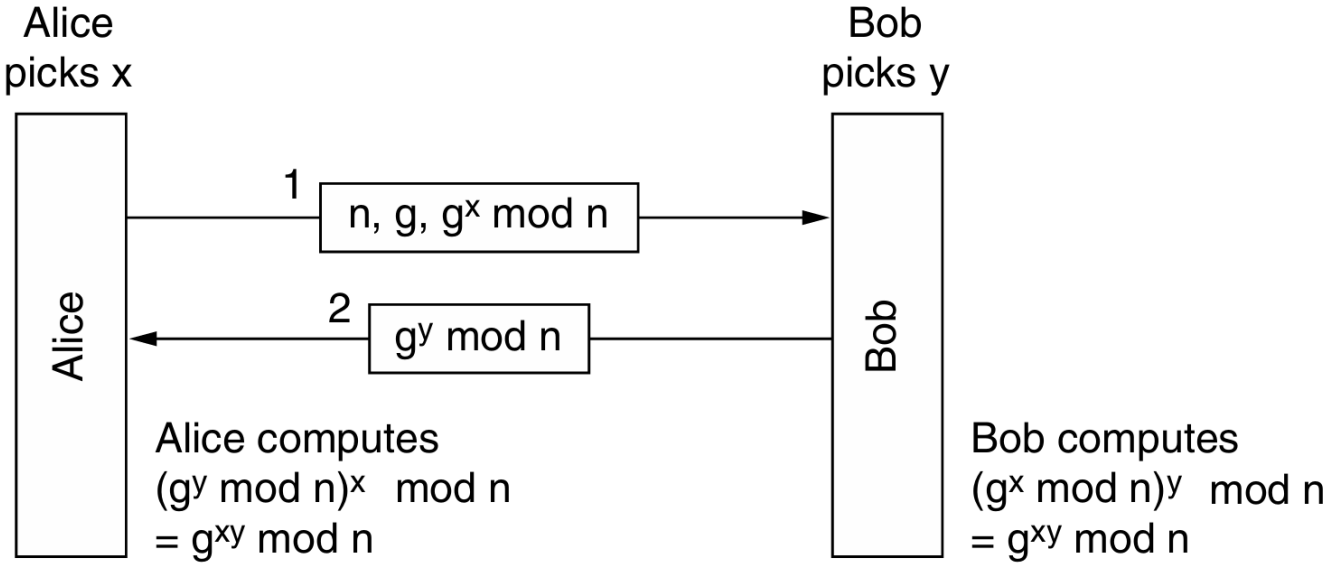


Figure 8-37. The Diffie-Hellman key exchange.

Of course, Trudy can see both messages, but given only g^x mod n, she cannot find x. No practical algorithm for computing discrete logarithm modulo a very large prime number is known.

Establishing a Shared Key: the Diffie-Hellman Key Exchange (III)

- Despite the elegance of the Diffie-Hellman algorithm, there is a problem: when Bob gets the triple (n, g, g^x mod n), how does he know it is from Alice and not from Trudy?
  - There is no way he can know. Unfortunately, Trudy can exploit this fact to deceive both Alice and Bob.
    - **The man-in-the-middle attack** (bucket brigade attack)

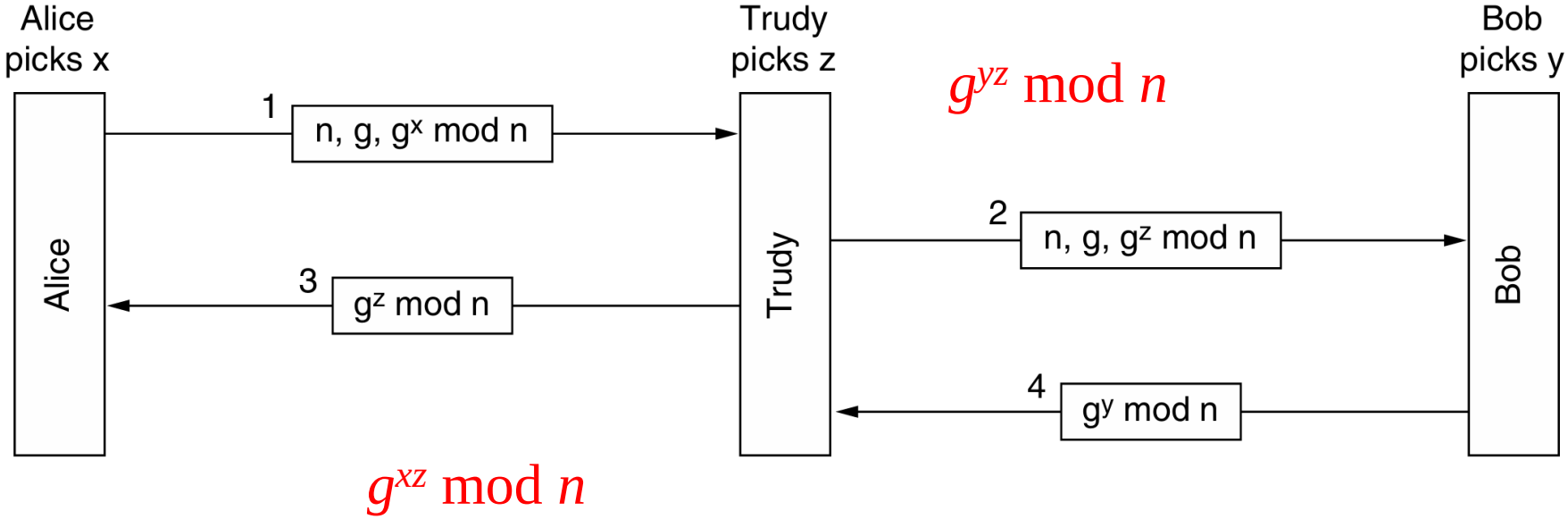
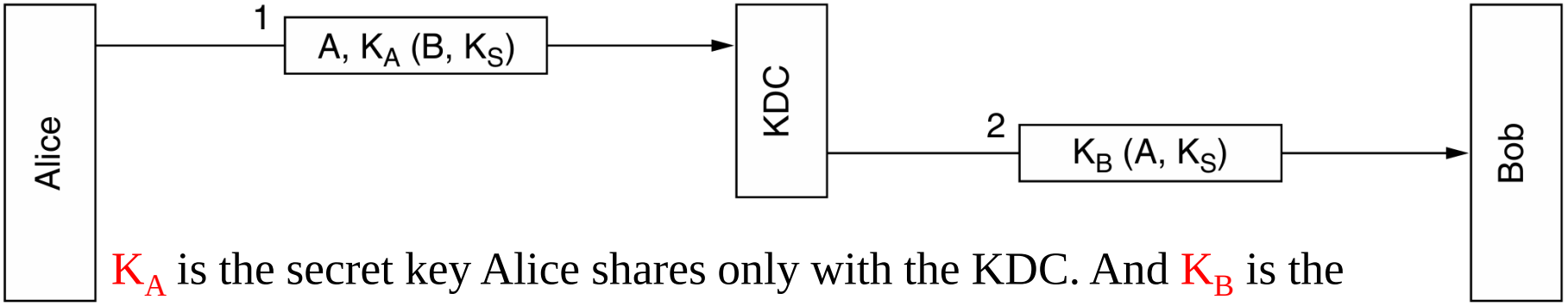


Figure 8-38. The man-in-the-middle attack.

Authentication Using a Key Distribution Center (I)

- For popular people, key management would become a real burden. A different approach is to introduce a trusted key distribution center.
  - In the protocol of Fig.8-39, each user has a single key shared with the KDC.
  - Authentication and session key management now go through the KDC.
  - But have the problem of **replay attack**.



K_A is the secret key Alice shares only with the KDC. And K_B is the secret key Bob shares only with the KDC. K_S is the session key that Alice want to use when she talk to Bob.

Figure 8-39. A first attempt at an authentication protocol using a KDC.

Solutions to the Replay Attack (I)

- Several solutions to the replay attack are possible.
  - To include a **timestamp** in each message.
    - The trouble with this approach is that clocks are never exactly synchronized over a network.
  - To put a **nonce** in each message
    - Each party should remember all previously nonces.

Solutions to the Replay Attack (II)

- The Needham-Schroeder authentication protocol (1978)
  - A multiway challenge-response protocol
  - By having each party both generate a challenge and respond to one, the possibility of any kind of replay attack is eliminate.

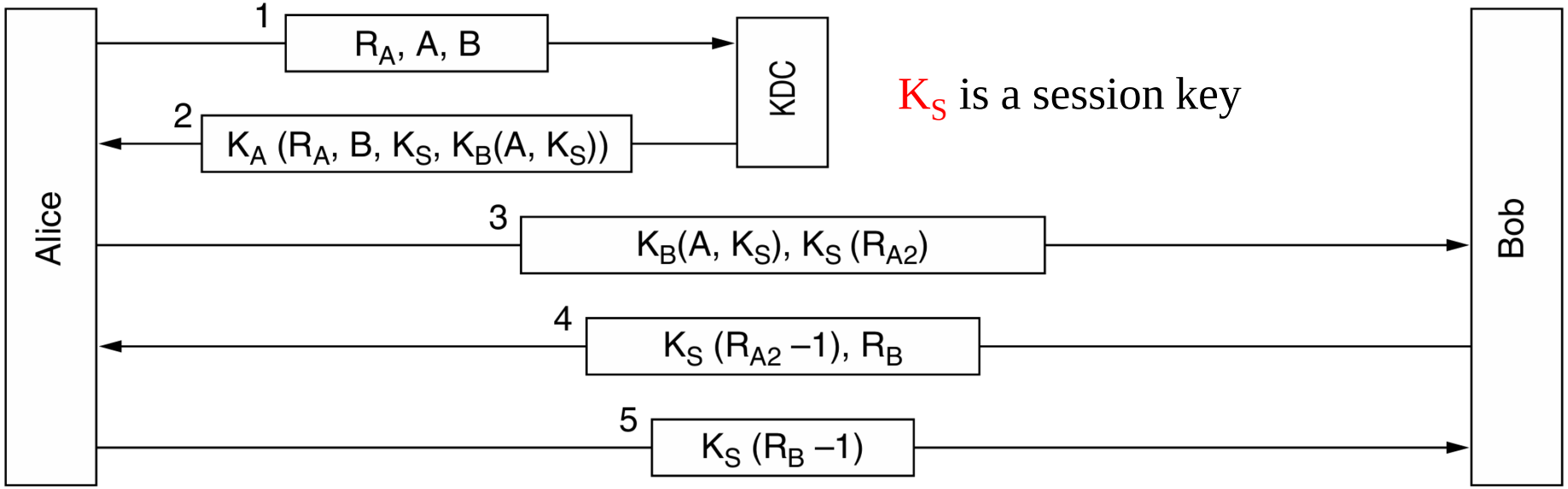


Figure 8-40. The Needham-Schroeder authentication protocol.

Solutions to the Replay Attack (III)

- The Needham-Schroeder authentication protocol (1978)
  - Message 1: The large random number, R_A, is used as nonce.
  - Message 2: The KDC sends back to Alice a message containing Alice random number, a session key, and a ticket that she can send to Bob. R_A is to assure Alice that message 2 is fresh, and not a reply. Bob's identity is enclosed in case Trudy replaces B in message 1 with her own identity. K_B is included inside the encrypted message to prevent Trudy from replacing it with something else.

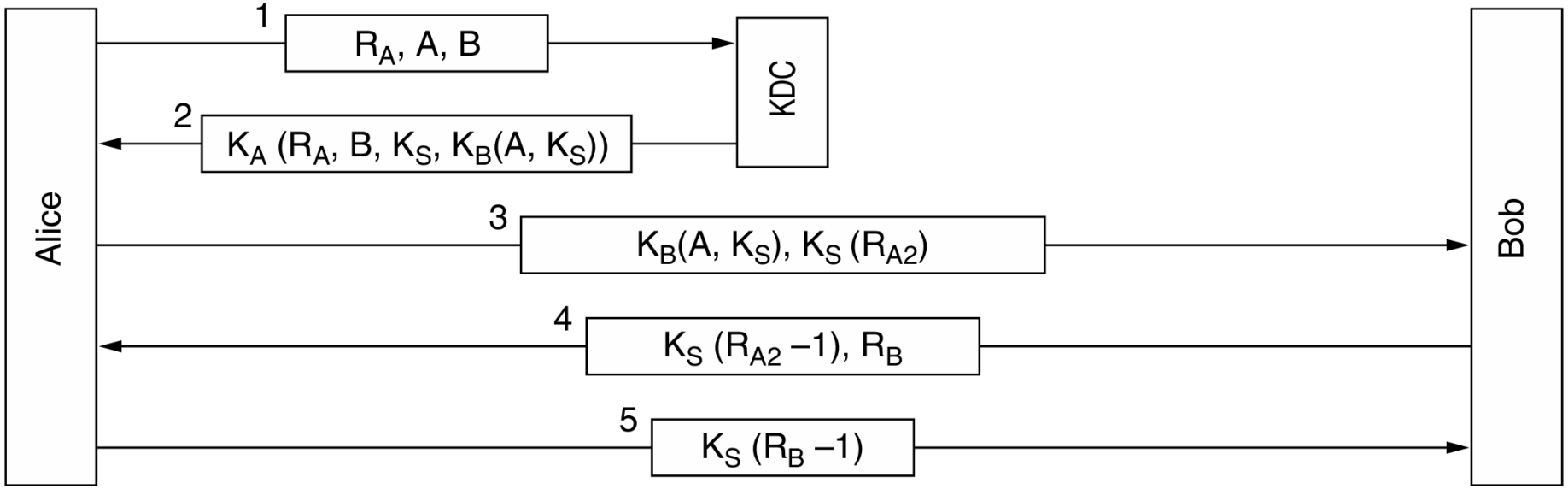


Figure 8-40. The Needham-Schroeder authentication protocol.

Solutions to the Replay Attack (IV)

- The Needham-Schroeder authentication protocol (1978)
  - Message 3: Alice now sends the ticket to Bob, along with a new random number, $R_{A2,}$ encrypted with the session key, K_S.
  - Message 4: Bob sends back $K_S(R_{A2} - 1)$ to prove Alice that she is talking to the real Bob.
    - Sending back $K_S(R_{A2})$ would not have worked, since Trudy could just have stolen it from message 3.

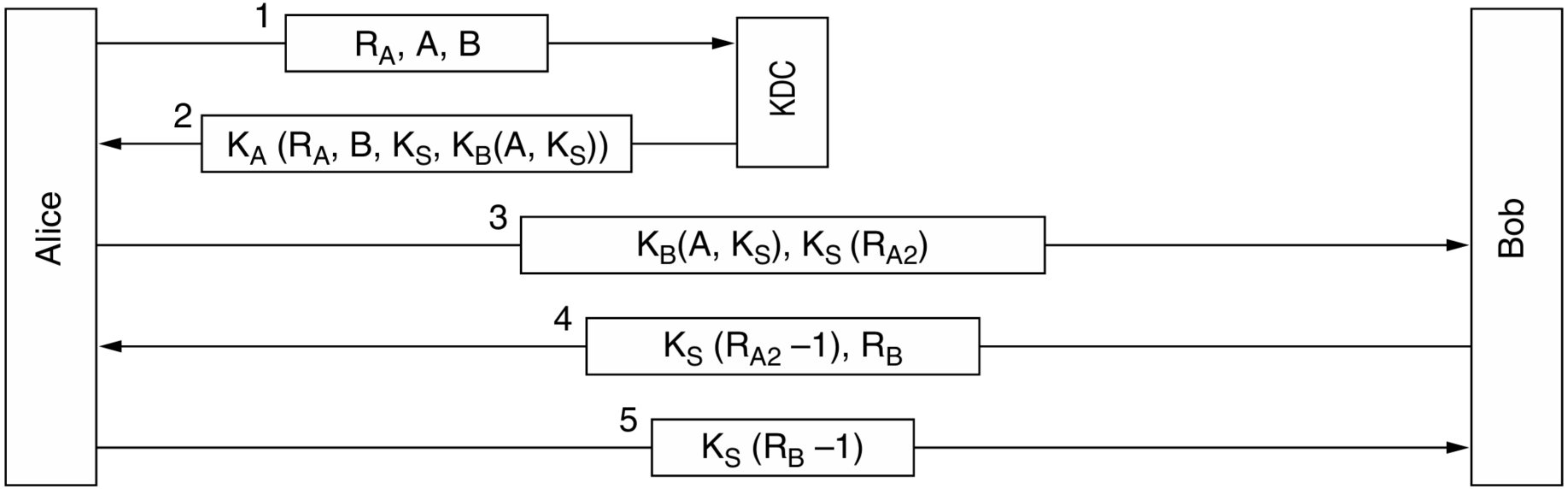


Figure 8-40. The Needham-Schroeder authentication protocol.

Solutions to the Replay Attack (V)

- The Needham-Schroeder authentication protocol (1978)
  - Message 5: is to convince Bob that it is indeed Alice he is talking to.
- This protocol does have a slight weakness. If Trudy ever manages to obtain an old session key in plaintext, she can initiate a compromised key and convince him that she is Alice.

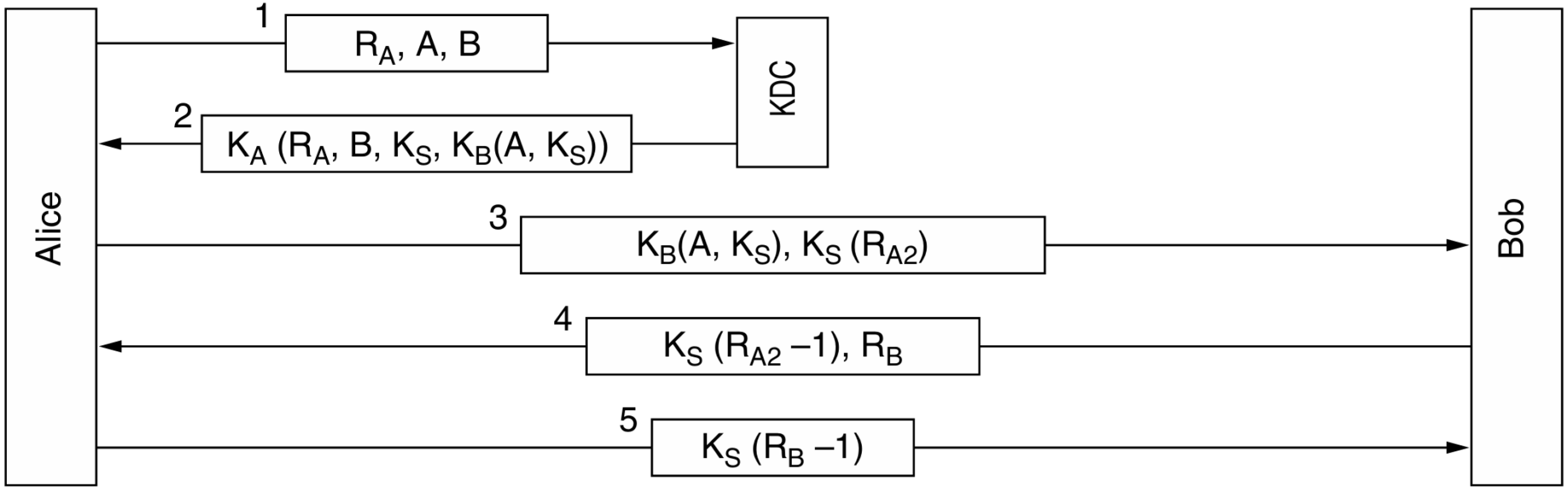


Figure 8-40. The Needham-Schroeder authentication protocol.

Solutions to the Replay Attack (VI)

- In the Otway-Rees protocol, Alice starts out by generating a pair of random numbers: R, which will be used as a common identifier, and R_A, which Alice will use to challenge Bob.
- When Bob gets this message, he construct a new message from the encrypted part of Alice's message and an analogous one of his own. Both the parts encrypted with K_A and K_B identify Alice and Bob, contain the common identifier, and contain a challenge.

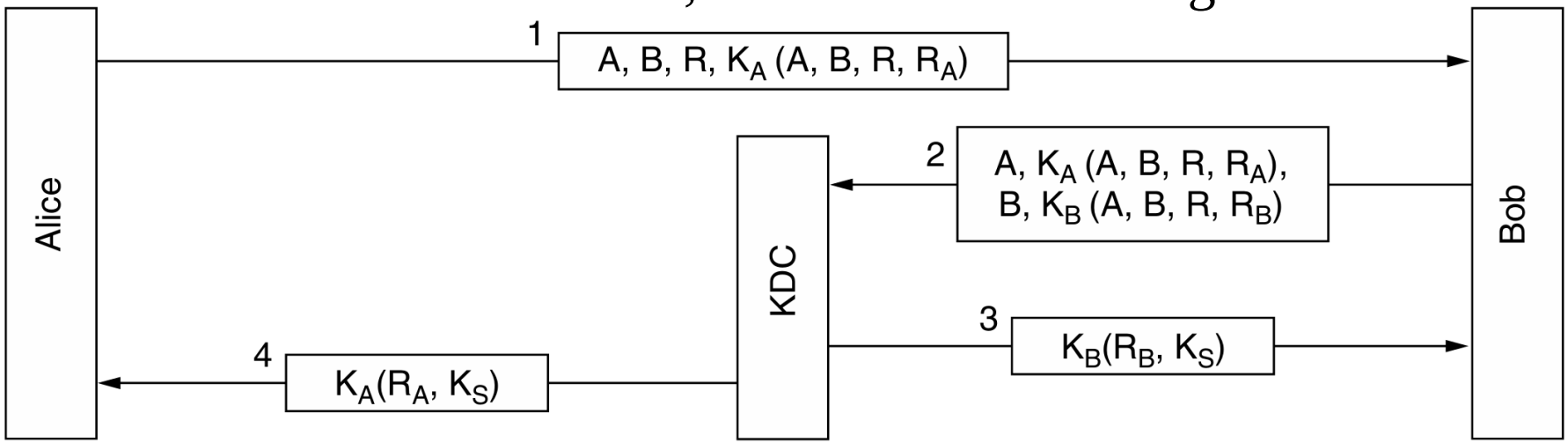


Figure 8-41. The Otway-Rees authentication protocol (slightly simplified).

Solutions to the Replay Attack (VII)

- When the KDC checks to see if the R in both parts is the same, the KDC believes that the request message from Bob is valid. It then generates a session key and encrypts its twice, once for Alice and once for Bob. Each message contains the receiver's random number, as proof that the KDC, and not Trudy, generated the message.
- At this point, both Alice and Bob can use the session key to start communication.

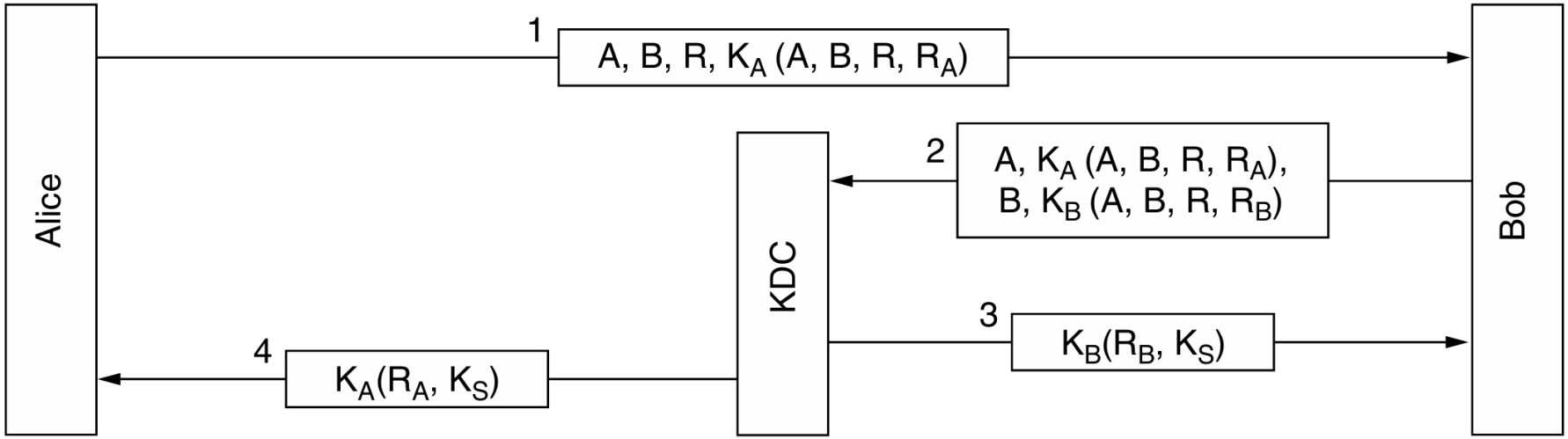


Figure 8-41. The Otway-Rees authentication protocol (slightly simplified).

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security

Email Security (I)

- Secure email system PGP (Pretty Good Privacy, 1991, Phil Zimmermann)
  - Released in 1991, PGP is a complete email security package that provides privacy, authentication, digital signatures, and compression, all in an easy-to-use form.
  - PGP encrypts data by using a block cipher called **IDEA** (International Data Encryption Algorithm), which uses 128-bits keys.
  - Key management uses **RSA** and data integrity uses **MD5**.
  - It is like a preprocessor that takes plaintext as input and produces signed cipher text in base64 as output. This output can then be email.

Email Security (II)

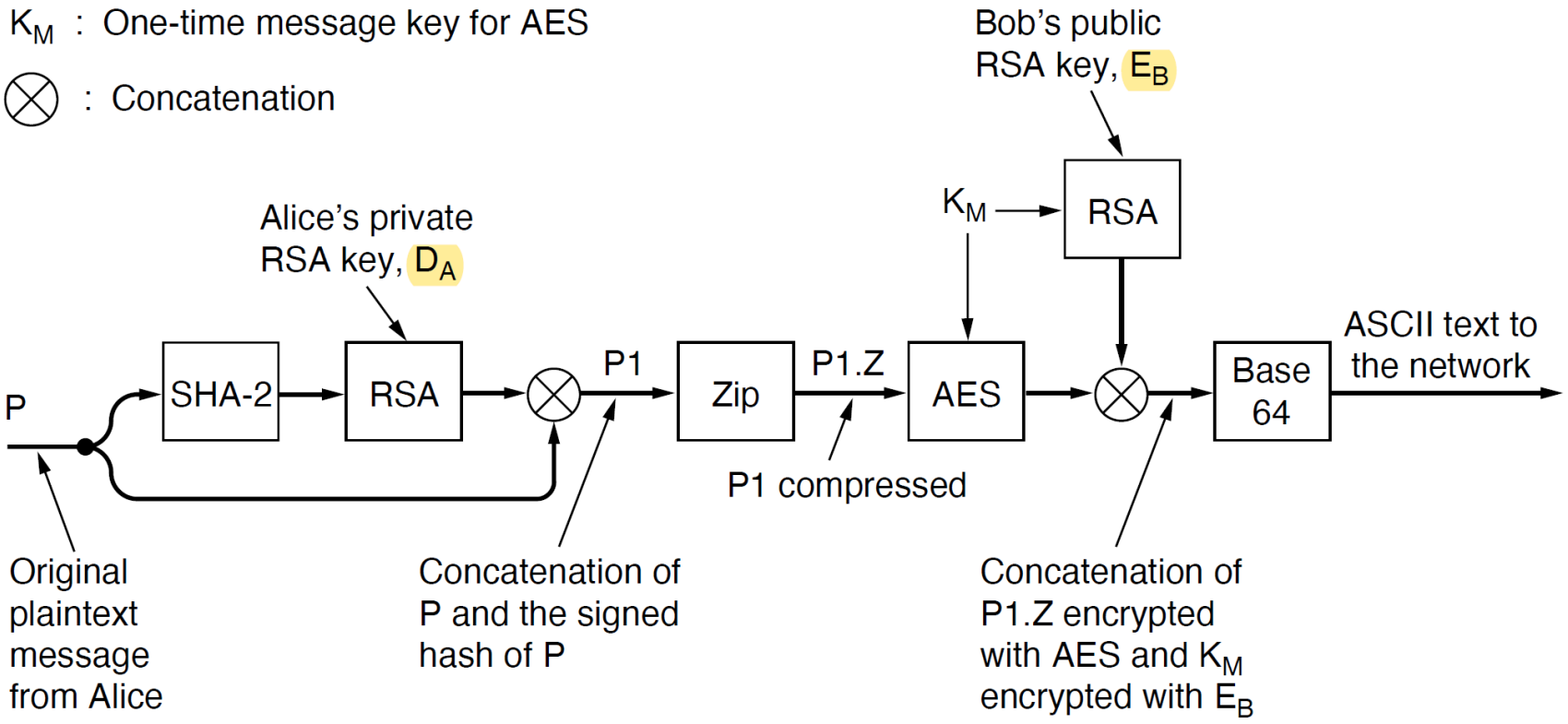


Figure 8-45. PGP in operation for sending a message.

1) Alice first hashes the original plaintext message P using **SHA-2**.
2) Then encrypts the resulting hash using her **private RSA key, D_A**.
3) The encrypted hash and the original message are **concatenated** into a single message P1, and compressed using the ZIP program (The Ziv-Lempel algorithm, 1977).

Email Security (III)

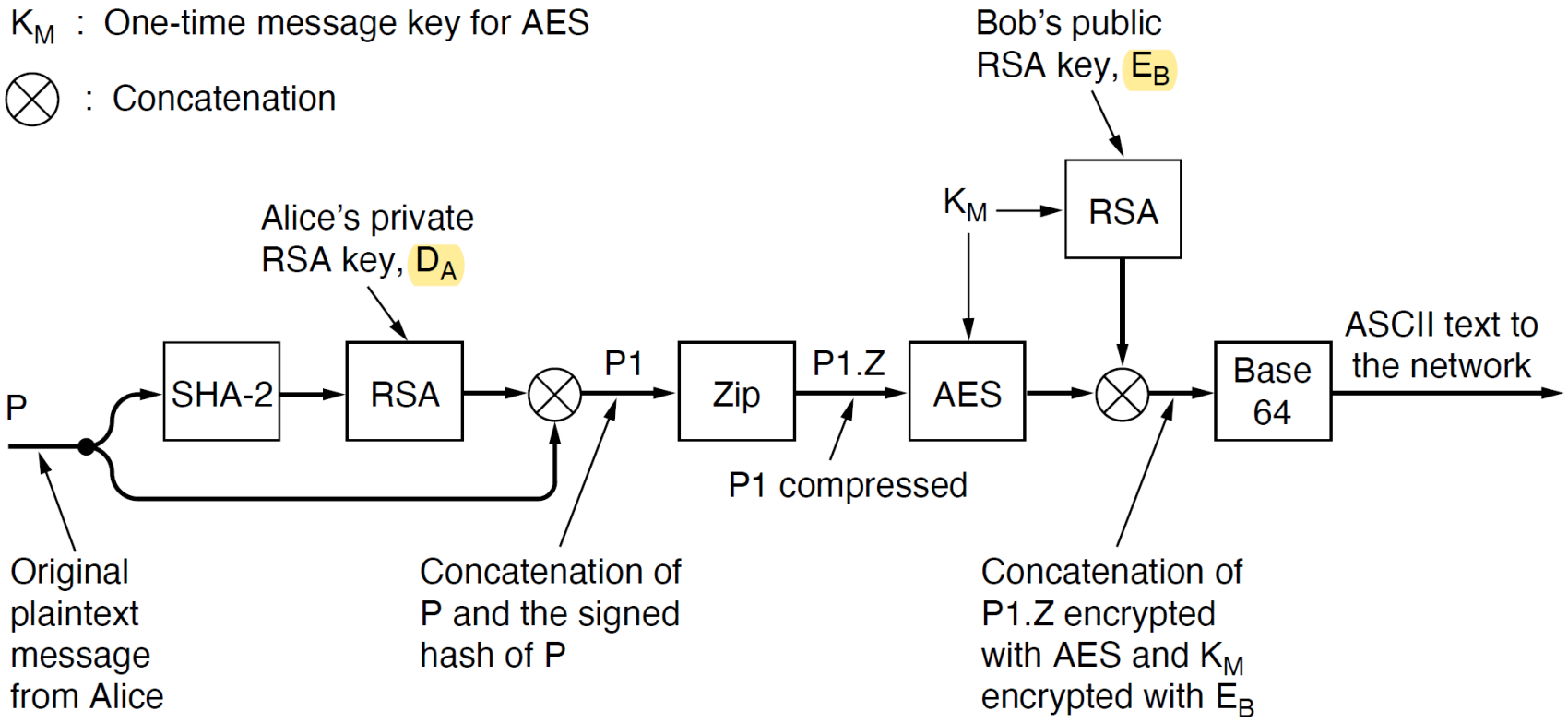


Figure 8-45. PGP in operation for sending a message.

4) PGP prompts Alice for some random input. Both the content and the typing speed are used to generated a 256-bit AES message key, K_M.

5) The K_M is used to encrypted P1.Z with IDEA in cipher feedback mode. In addition, K_M is encrypted with **Bob's public key**, $\mathbf{E_B}$. These two components are concatenated and converted to base64.

Email Security (IV)

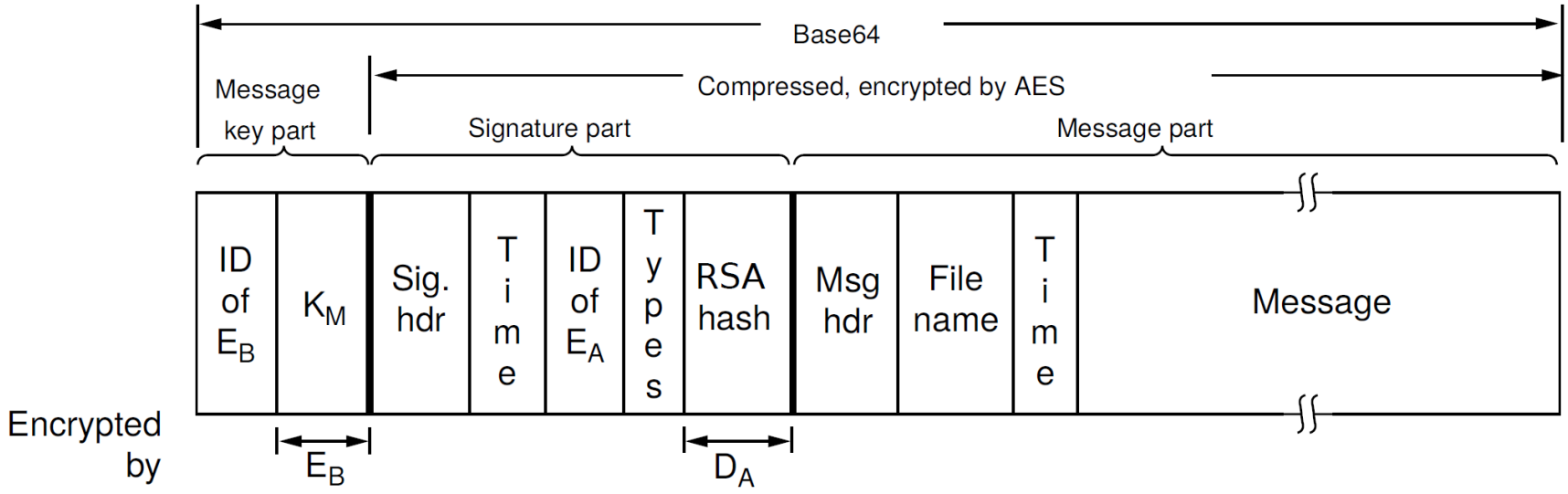


Figure 8-46. A PGP message.

In PGP, RSA is only used for two small computations. The heavy-duty encryption is done by AES, which is orders of magnitudes faster than RSA.
The message has three parts, containing the IDEA key, the signature, and the message respectively.

Outlines

- Basic knowledge
  - Cryptography
  - Symmetric-key algorithms
  - Public-key algorithms
  - Digital signatures
  - Management of public keys
- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security (https)

Outlines

- Applications
  - Communication security
  - Authentication protocols
  - Email security
  - Web security (https)
    - DNSsec
    - SSL — the Secure Sockets Layer
    - TLS — the Transport Layer Security Protocol

DNS Spoofing

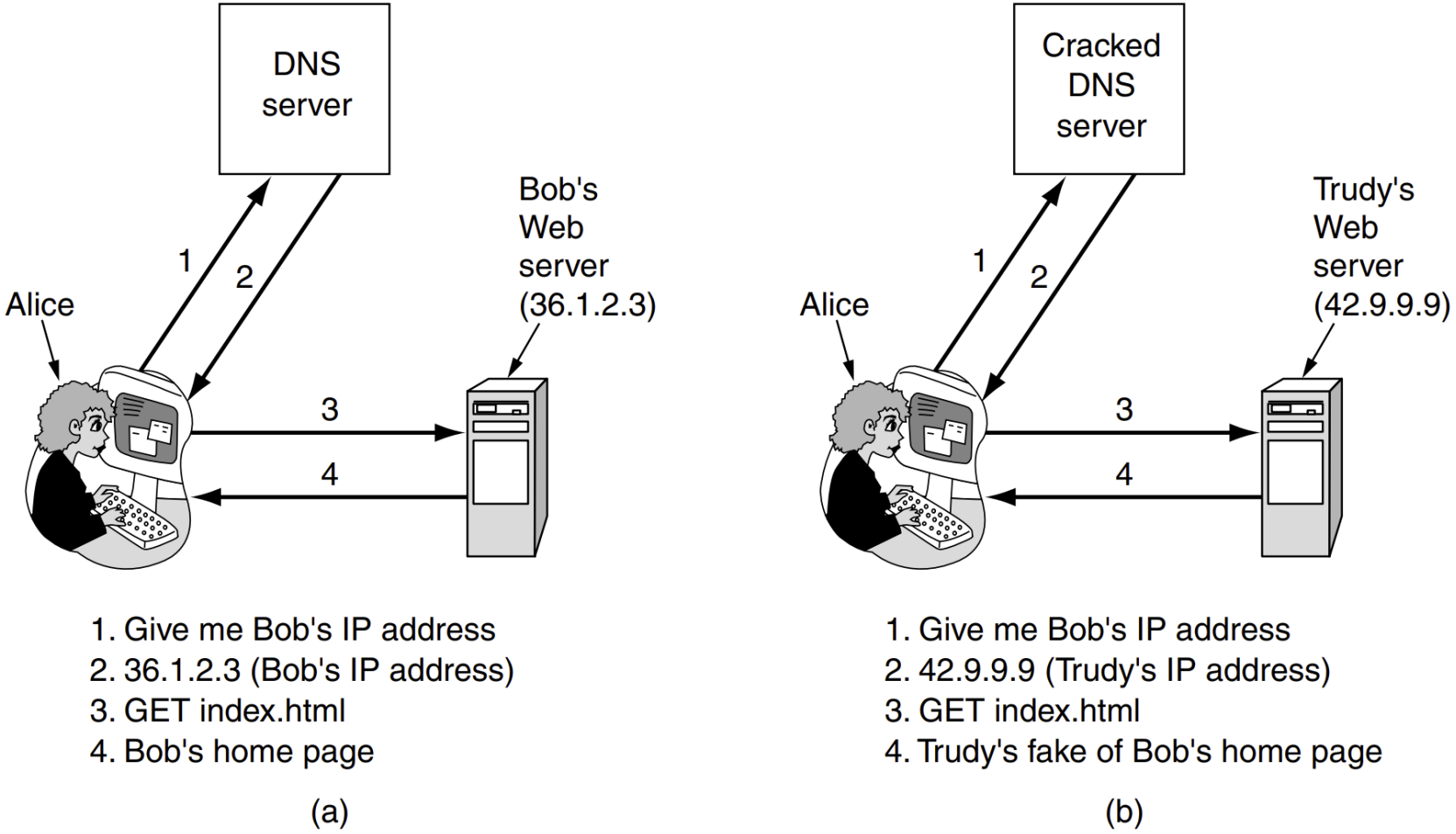
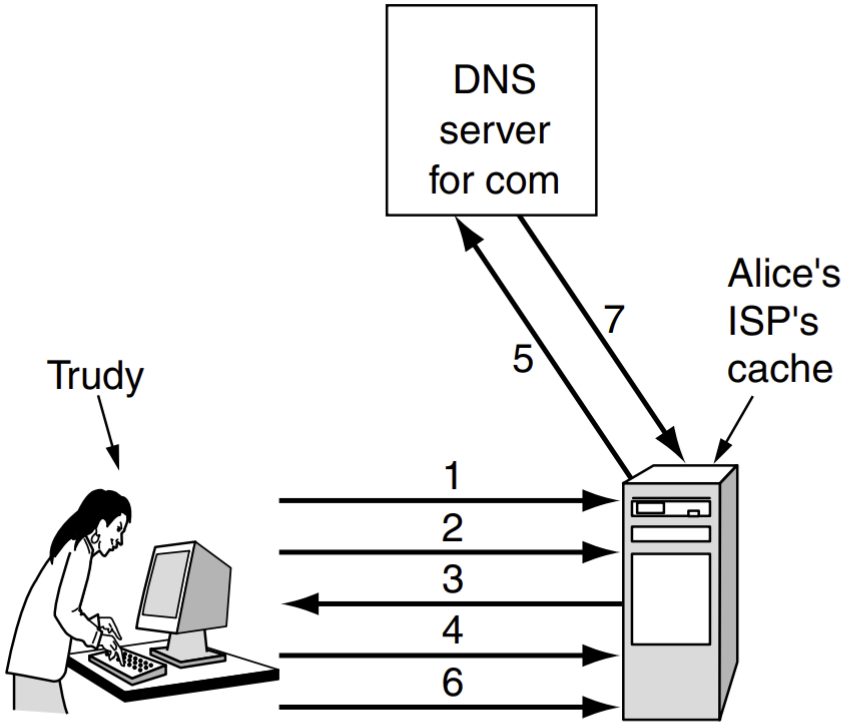


Figure 8-46. (a) Normal situation. (b) An attack based on breaking into a DNS server and modifying Bob's record.

DNS Spoofing — man-in-the-middle attack



1. Look up foobar.trudy-the-intruder.com
   (to force it into the ISP's cache)
2. Look up www.trudy-the-intruder.com
   (to get the ISP's next sequence number)
3. Request for www.trudy-the-intruder.com
   (Carrying the ISP's next sequence number, n)
4. Quick like a bunny, look up bob.com
   (to force the ISP to query the com server in step 5)
5. Legitimate query for bob.com with seq = n+1
6. Trudy's forged answer: Bob is 42.9.9.9, seq = n+1
7. Real answer (rejected, too late)

Figure 8-47. How Trudy spoofs Alice's ISP.

DNSSEC (DNS security)

- Presented in RFC2535, it is based public-key cryptograph.
  - Every DNS zone has a public/private key pair. All information sent by a DNS server is signed with the originating zone's private key, so the receiver can verify its authenticity.
- DNSSEC offers three fundamental services
  - 1. Proof of where the data originated
  - 2. Public key distribution
  - 3. Transaction and request authentication

DNSSEC (DNS security)

- DNS records are grouped into sets called RRSets (Resource Record Sets) with all the records having the same name, class and type.
- Each RRSets is cryptographically hashed (e.g. using SHA-1)
- The hash is signed by the zones' private key (e.g. using RSA)
- The unit of transmission to clients is the signed RRSets.

Security of DNS (II)

- Extensions and enhancements to DNS queries
  - 1) **EDNS0 CS (Extended DNS Client Subnet** or **EDNS Client Subnet** option): a client's local recursive resolver passes **the IP address subnet of stub resolver** to the authoritative name server.
    - Example: Google and Cloudflare both operate public DNS resolvers (8.8.8.8 and 1.1.1.1 respectively) [6]. If a client is configured to use one of these local recursive resolvers, then the authoritative name server does not learn much useful information from the IP address of the recursive resolver.
  - Knowing this information can typically allow an authoritative DNS server to perform a more effective mapping to a nearby copy of a replicated service.

Security of DNS (III)

- Extensions and enhancements to DNS queries
  - 2) **0x20 Encoding**: a local recursive server would toggle the case on each QNAME.
    - Previously, the names in DNS queries are not case sensitive.
    - e.g., ZJu.EdU instead of zju.edu (随机化大小写验证技术，该技术会匹配请求包和响应包的大小写，如果大小写不匹配就会丢弃响应包。)

Security of DNS (IV)

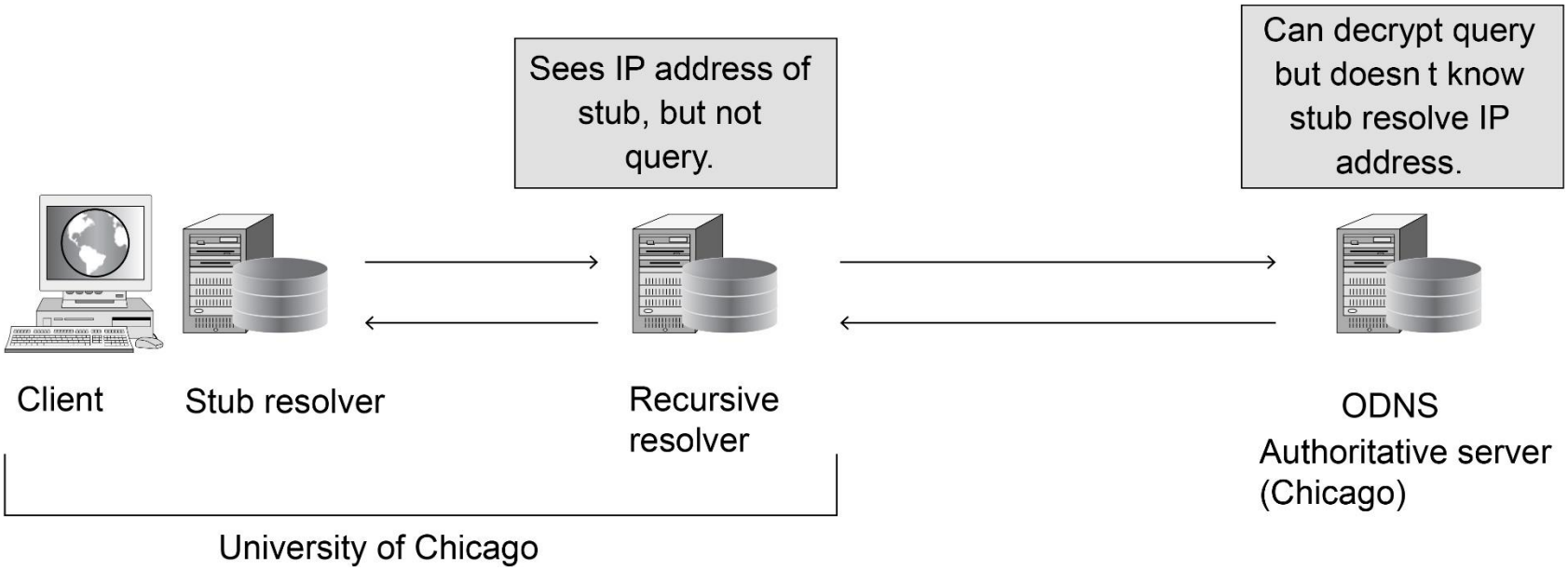
- When performing a series of iterative lookups, a recursive DNS server resolver might send the entire QNAME to the sequence of authoritative name servers returning the responses.
  - A privacy risk.
  - **QNAME minimization** (RFC7816) not the entire full qualified domain name (FQDN)

Security of DNS (V)

- Originally DNS maps names to IP addresses, another common use for DNS queries is to look up domains in a **DNSBL** (DNS-based blacklist)
  - The lists that are commonly maintained to keep track of IP addresses associated with spammers (垃圾广告) and malware (恶意软件).
- Ranging from cache poisoning to distributed denial-of-service (DDoS) attacks, has resulted in an increasing trend towards the use of **TCP** as the transport protocol for DNS.
  - DNS-over-TLS (**DoT**) and DNS-over-HTTPS (**DoH**)
  - Privacy risk

Security of DNS — Oblivious DNS

- **Oblivious DNS** (2019)



The stub resolver encrypts the original query to the local recursive resolver, which in turn sends the encrypted query to an authoritative name server that can decrypt and resolve the query, but does not know the identity or IP address of the stub resolver that initiated the query.

SSL — The Secure Sockets Layer

- Some applications, such as purchasing merchandise by credit card, online banking, and electronic stock trading, demands for **secure connections**.
- SSL builds a secure connection between two sockets, including
  - 1. Parameter negotiation between client and server.
  - 2. Authentication of the server by the client
  - 3. Secret communication
  - 4. Data integrity protection

The Position of SSL in the Protocol Stack

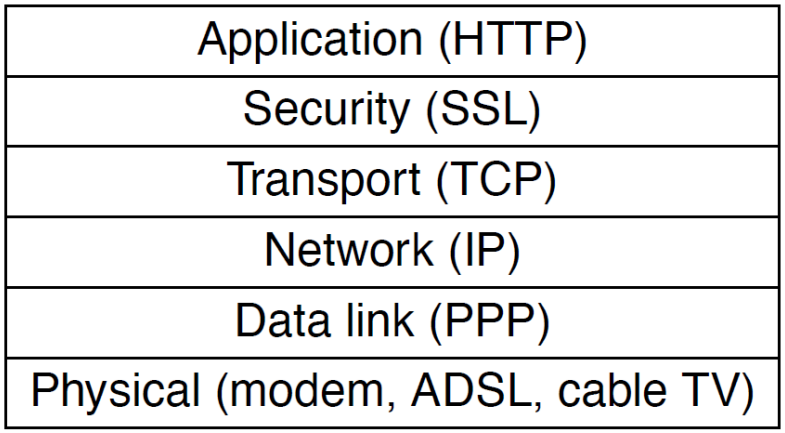


Figure 8-48. Layers (and protocols) for a home user browsing with SSL.

Once the secure connection has been established, SSL's main job is handling compression and encryption.
When HTTP is used over SSL, it is called **HTTPS** (**Secure HTTP**), even though it is the standard HTTP protocol. Sometimes it is available at a new port (**443**) instead of port 80. *SSL is not restricted to Web browsers*.

The SSL Protocol: Establishment

- SSL consists of two subprotocols, one for establishing a secure connection (SSL version 3) and one for using it (TLS, Transport Layer Security).

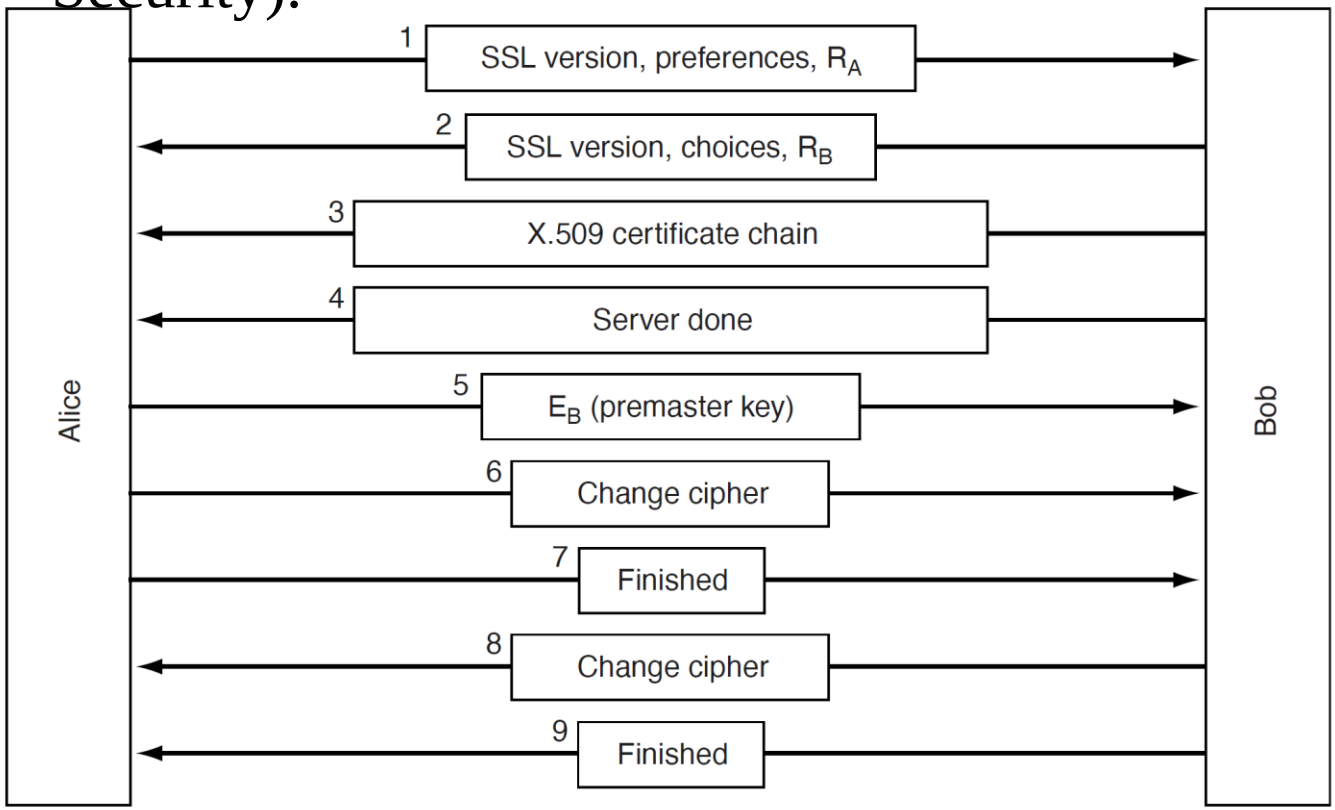


Figure 8-49. A simplified version of the SSL connection establishment subprotocol.

In message 3, he sends a certificate containing his public key.
In message 5, Alice responds by choosing a random 384-bit premaster key and sending it Bob encrypted with his public key.
The actual session key used for encrypting data is derived from the premaster key combined with both nonces in a complex way.

The SSL Protocol: Transmission

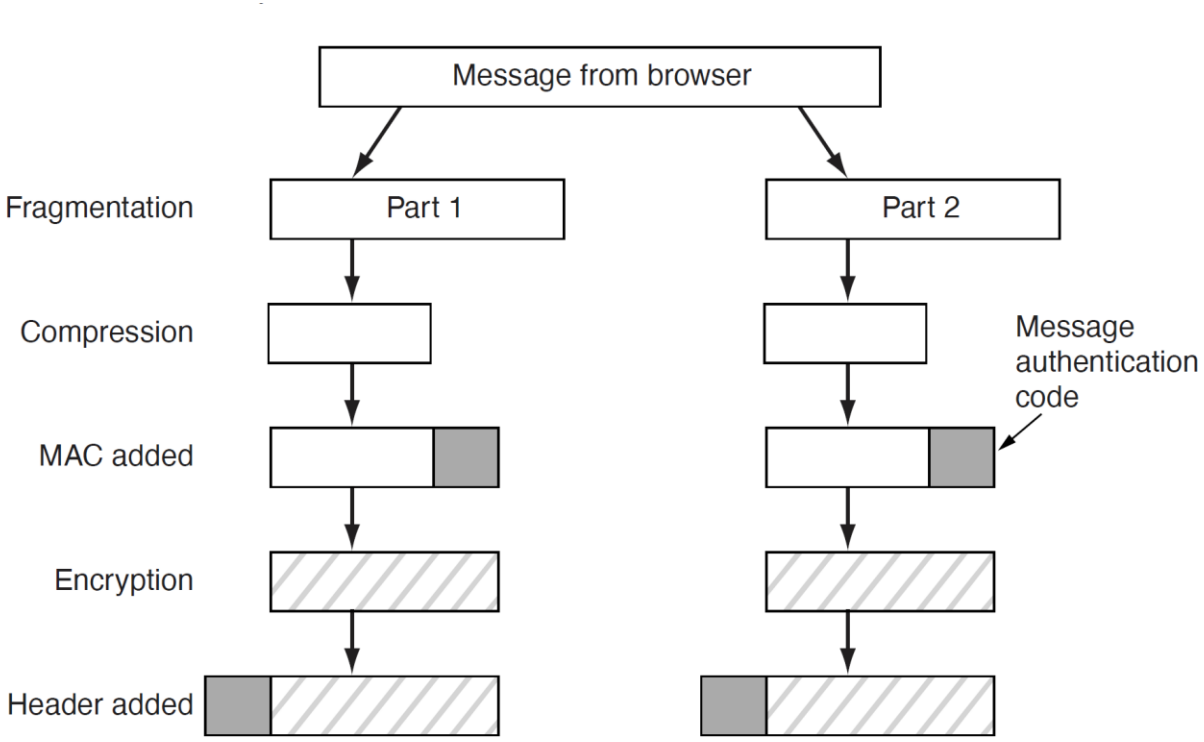
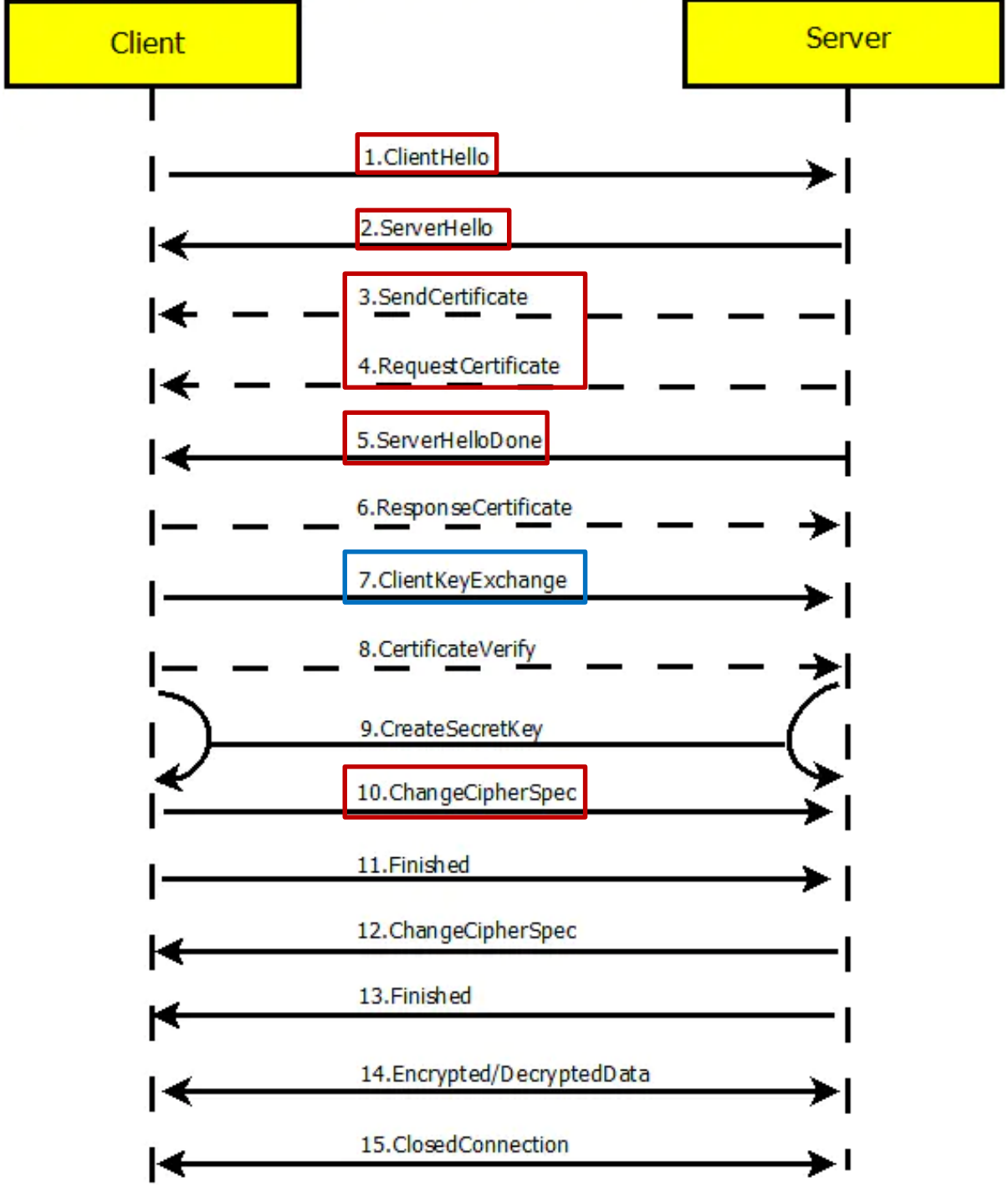
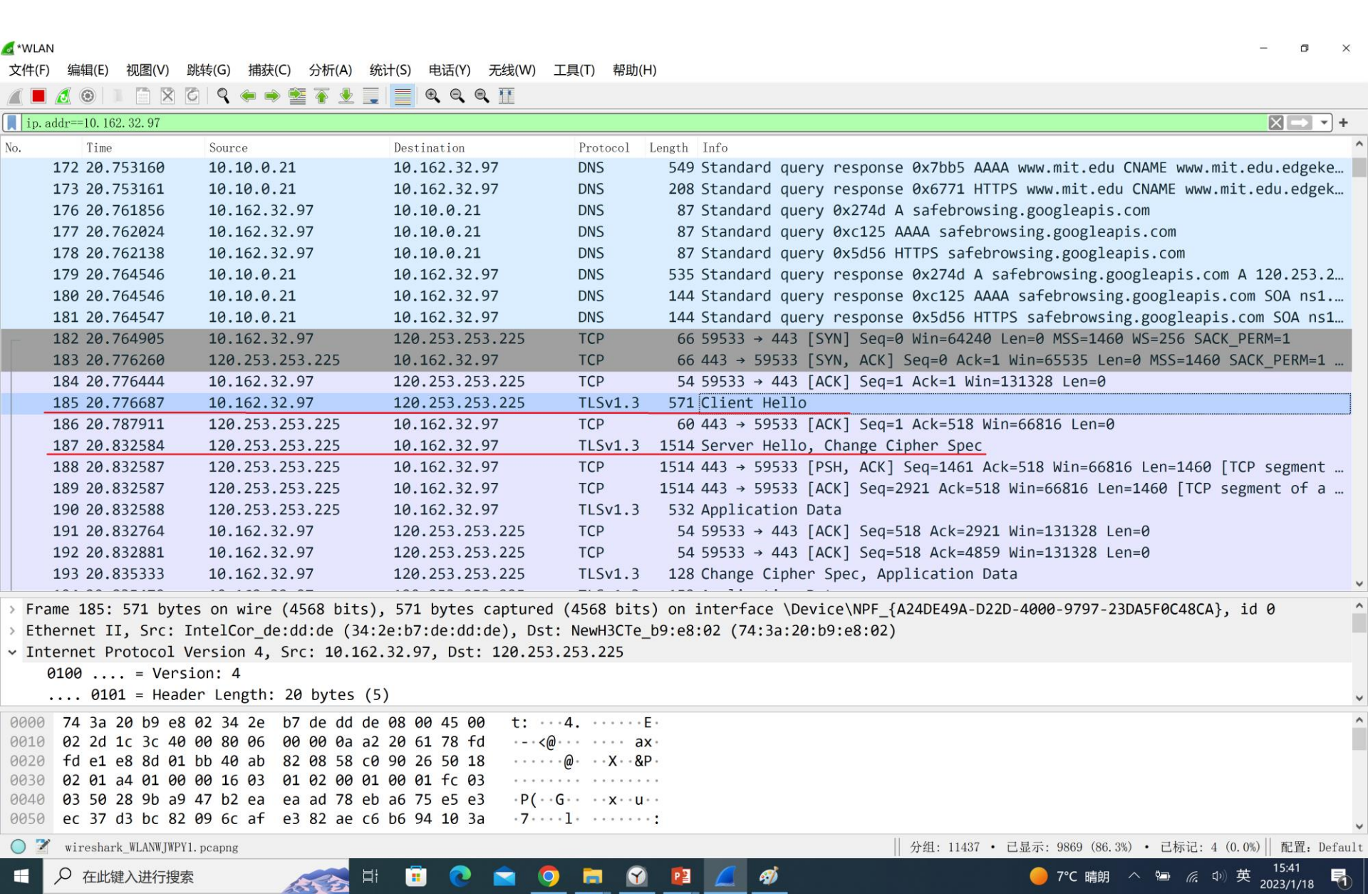


Figure 8-50. Data transmission using SSL.

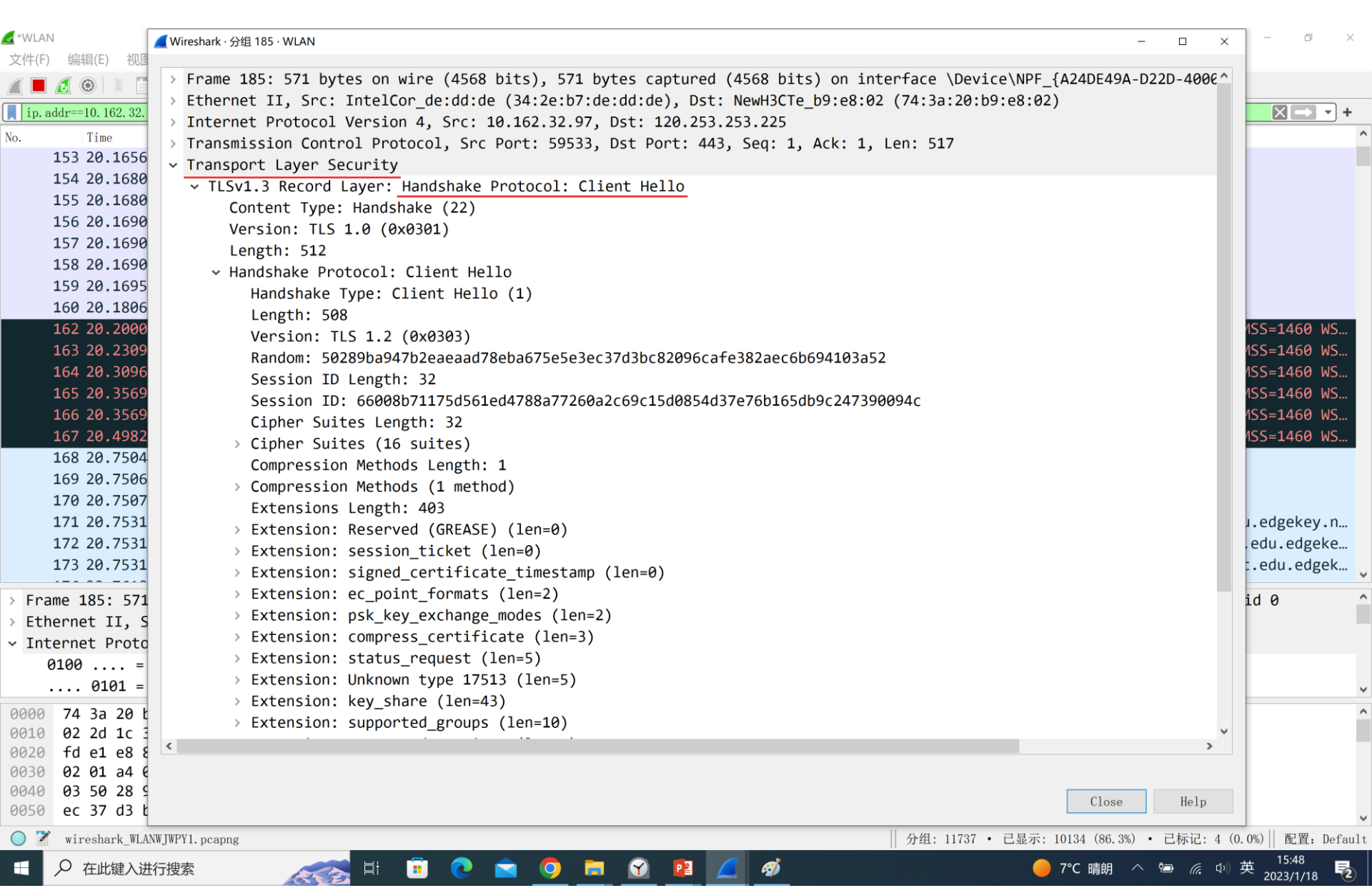
Messages from the browser are first broken into units of up to 16 KB. If compression is enabled, each unit is then separately compressed. A secret key derived from the two nonces and premaster key is concatenated with the compressed text and the result is hashed with the agreed-on hashing algorithm. This hash is appended to each fragment as the MAC. The compressed fragment plus MAC is then encrypted with the agree-on symmetric encrypted algorithm. Finally, a header is added.

[4]

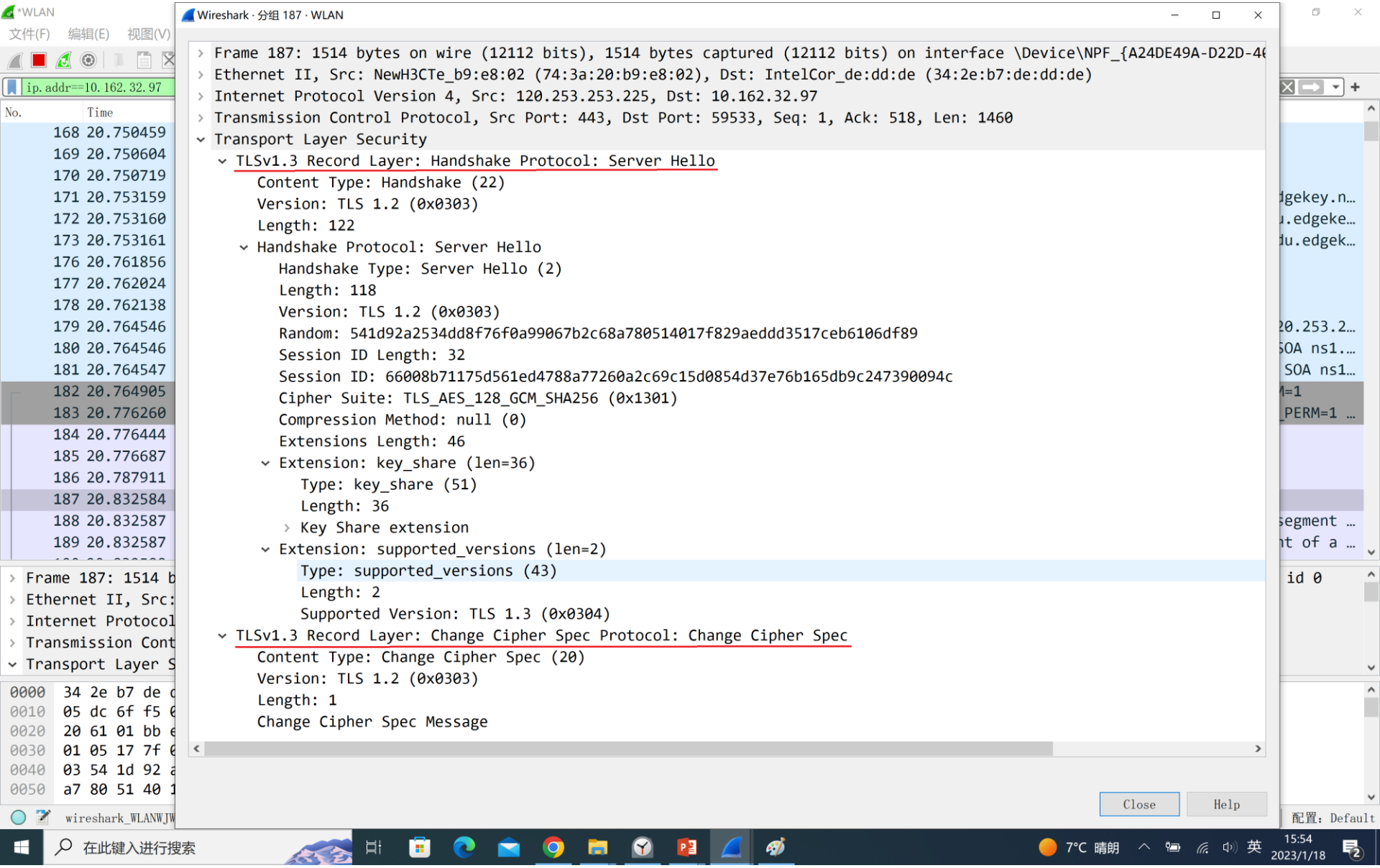

Client
Server
1.ClientHello
2.ServerHello
3.SendCertificate
4.RequestCertificate
5.ServerHelloDone
6.ResponseCertificate
7.ClientKeyExchange
8.CertificateVerify
9.CreateSecretKey
10.ChangeCipherSpec
11.Finished
12.ChangeCipherSpec
13.Finished
14.Encrypted/DecryptedData
15.ClosedConnection



注意这里的TLS1.3 “Client Hello”数据包和“Server Hello, Change Cipher Spec”数据包



注意这里“Client Hello”有很多扩展项。



“Server Hello”数据包里包含了“Change Cipher Spec”

Main Points (I)

- Kerckhoff's principle: **All algorithms must be public; only the keys are secret**.
  - The longer the key, the higher the work factor the cryptanalyst has to deal with.
- Encryption methods can be divided into *two categories*: **substitution ciphers** and **transposition ciphers**.
- In **symmetric-key** algorithms, they use **the same key** for encryption and decryption.
  - DES
  - Triple DES
  - AES (Rijndael)
  - RC4
  - RC5

Main Points (II)

- Public-key cryptography requires each user to have two keys: **a public key**, used by the entire world for encrypting message to be sent to that user, and **a private key**, which the user needs for decrypting messages.
  - RSA
- Digital signatures
  - Symmetric-key signatures
  - Public-key signatures
  - Message digests
    - SHA-1 or MD5
  - The birthday attack

Main Points (III)

- Authentication protocols
  - reflection attack
  - 1) The Diffie-Hellman Key Exchange (shared key)
    - The man-in-the-middle attack
    - Replay attack
      - The solution to replay attack: time stamp and nonce
      - The Needham-Schroeder authentication protocol (1978)
      - The Otway-Rees protocol
  - 2) Kerberos: all clocks are fairly well synchronized, three servers (authentication server, ticket grant server, server)
  - 3) Public-key
- Firewall
  - They are network layer devices, but they peek at the transport and application layers to do their filtering.

Main Points (IV)

- IPsec
- VPN
- Wireless security
- Email security (PGP)
- Web Security
  - DNSsec
  - SSL/TLS (HTTPs, port: 443)

References

- [1] A.S. Tanenbaum, and D.J. Wetherall, Computer Networks, 6th Edition, Prentice Hall, 2021.
- [2] https://www.openpgp.org/
- [3] https://philzimmermann.com/EN/background/index.html
- [4] https://www.jianshu.com/p/1fc7130eb2c2
- [5] https://www.jianshu.com/p/2574aa671888 (TLS一个简短说明)
- [6] https://www.openssl.net.cn/
- [7] https://github.com/openssl/openssl