

알기 쉬운

정보보호개론

3판

흥미로운 암호 기술의 세계

INFORMATION SECURITY and CRYPTOGRAPHY





INFORMATION SECURITY and CRYPTOGRAPHY

CHAPTER 14 PGP

Section 01 PGP 개요

Section 02 키 쌍의 작성

Section 03 암호화와 복호화

Section 04 디지털 서명 작성과 검증

Section 05 디지털 서명작성과 암호화 및 복호화한
디지털 서명 검증

Section 06 신뢰망

Section 01

PGP 개요

1.1 PGP

1.2 OpenPGP에 대해

1.3 GNU Privacy Guard

1.4 PGP 기능

1.1 PGP

- **PGP**(Pretty Good Privacy)
- 1990년경에 필립 짐머만(Philip Zimmermann)이 제작
- 현재도 전 세계에서 널리 사용되고 있는 암호 소프트웨어
- Windows, UNIX, Macintosh 등 기타 많은 플랫폼 상에서 작동
- 상업용 또는 프리 소프트웨어용으로 제작
 - 프리 소프트웨어
 - **OpenPGP**(RFC 4880) 사양에 따라 GNU가 작성한 **GnuPG**(GNU Privacy Guard)

1.2 OpenPGP에 대해

- OpenPGP
 - 암호문이나 디지털 서명 형식을 정한 규격
 - RFC1991: (1991년) PGP 메시지 형식을 기술
 - RFC2440
 - RFC4880: (2007년) RSA와 DSA를 지원
 - RFC5581
 - RFC6637: (2012년) 타원 곡선 암호(ECC)를 지원
 - Curve P-256, P-384, P-521이라는 3종류의 타원곡선을 사용하여 타원곡선 DSA(Elliptic Curve Digital Signature Algorithm: ECDH)라는 2개의 알고리즘을 지원

RFC6637

- 암호적으로 강력한 균형을 유지하기 위한 표 제시

– 예:

- 타원 곡선 암호로 256 비트를 선택한 경우 해시로서 256 비트를 선택하고 아직 대칭암호의 키 길이로서 128 비트를 선택

표 14-1 암호적인 강한 밸런스

타원곡선명	ECC	RSA	해시	대칭 암호
P-256	256 비트	3,072 비트	256 비트	128 비트
P-384	384 비트	7,680 비트	384 비트	192 비트
P-521	521 비트	15,360 비트	512 비트	256 비트

1.3 GNU Privacy Guard

- GNU Privacy Guard(GnuPG, GPG)
 - OpenPGP 규격에 따라 만들어진 암호화 소프트웨어
 - 암호화, 디지털 서명, 키 관리, S/MIME와 ssh 등을 지원
 - GNU PGL에 기초한 프리소프트웨어

GnuPG 버전

- GnuPG stable
 - 버전번호가 2.0.이기 때문에, OpenPGP, S/MIME, ssh도 지원
- GnuPG modern
 - 버전번호가 2.1.이기 때문에 stable에 더해서 타원 곡선 암호도 지원
 - GnuPG stable을 대체하는 새로운 버전
- GnuPG classic
 - 버전번호가 1.4.로 구 버전

1.4 PGP 기능

- 대칭암호
- 공개 키 암호
- 디지털 서명
- 일방향 해시 함수
- 인증서
- 압축
- 텍스트 데이터
- 큰 파일의 분할과 결합
- 키 고리 관리

대칭암호

- 대칭 암호에 의한 암호화와 복호화를 지원
- 대칭 암호 단독이용 가능
- 공개 키 암호와 조합한 하이브리드 암호 시스템의 일부로서 이용 가능
- 사용할 수 있는 대칭 암호 알고리즘
 - AES, IDEA, CAST, 삼중 DES, Blowfish, Twofish
- 블록 암호의 모드:
 - CFB 모드

공개 키 암호

- 공개 키 암호의 키 쌍 작성, 공개 키 암호에 의한 암호화와 복호화를 지원
- 하이브리드 암호 시스템을 이용
- 이용할 수 있는 공개 키 암호 알고리즘
 - RSA, ElGamal

디지털 서명

- 디지털 서명 작성과 검증을 지원
- 파일에 디지털 서명 첨부
- 파일과 서명 분리
- 이용할 수 있는 디지털 서명 알고리즘
 - RSA, DSA

일방향 해시 함수

- 일방향 해시 함수를 이용해서 메시지의 해시 값을 계산하고 표시
- 이용 가능한 일방향 해시 함수 알고리즘
 - MD5, SHA-1, RIPEMD-160

인증서

- OpenPGP에서 정해진 형식의 인증서와, X.509 호환용 인증서를 작성
- 공개 키의 취소 증명서(revocation certificate) 발행

압축

- 데이터의 압축과 확장
- 압축에 이용되는 형식
 - ZIP

텍스트 데이터 변환

- 2진 데이터와 텍스트 데이터의 상호 변환을 수행
- 2진 데이터를 텍스트 데이터(ASCII radix-64 형식)로 변환
 - radix-64 형식
 - 메일 등에서 자주 사용되는 base64 인코딩에 데이터에러 검출을 위해 검사합을 부가한 것

큰 파일의 분할과 결합

- 대 용량 파일을 복수의 파일로 분할하는 기능
- 역으로 복수의 파일을 결합해서 하나의 파일로 결합하는 기능

키 고리 관리

- 생성한 자신의 키 쌍이나, 입수한 공개 키를 관리하는 기능
 - 키 링(key ring) 혹은 키 고리
 - 키를 모아 놓은 파일

Section 02

키 쌍의 작성

- 암호화나 디지털 서명을 하려면, 우선 자신의 키 쌍 작성 필요

GnuPG 1.4.9를 이용한 키 쌍 작성

```
$ gpg2 --full-gen-key ← 키 쌍 생성 명령
gpg (GnuPG) 2.1.4; Copyright (c) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
  (1) RSA and RAS (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection? 1 ← 키 종류를 선택(여기서는 암호용으로 RSA, 디지털 서명용으로 RSA를 선택)
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048 ← 키 비트 수를 지정
Requested keysize is 2048 bits
Please specify how long the key should be valid.
  0 = key does not expire
  <n> = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0) 1y ← 키의 유효기간을 지정 (1년)
Key expires at Sat Jun  4 11:13:57: 2016 JST
Is this correct? (y/N) y ← 올바른지 확인

GnuPG needs to construct a user ID to identify your key.

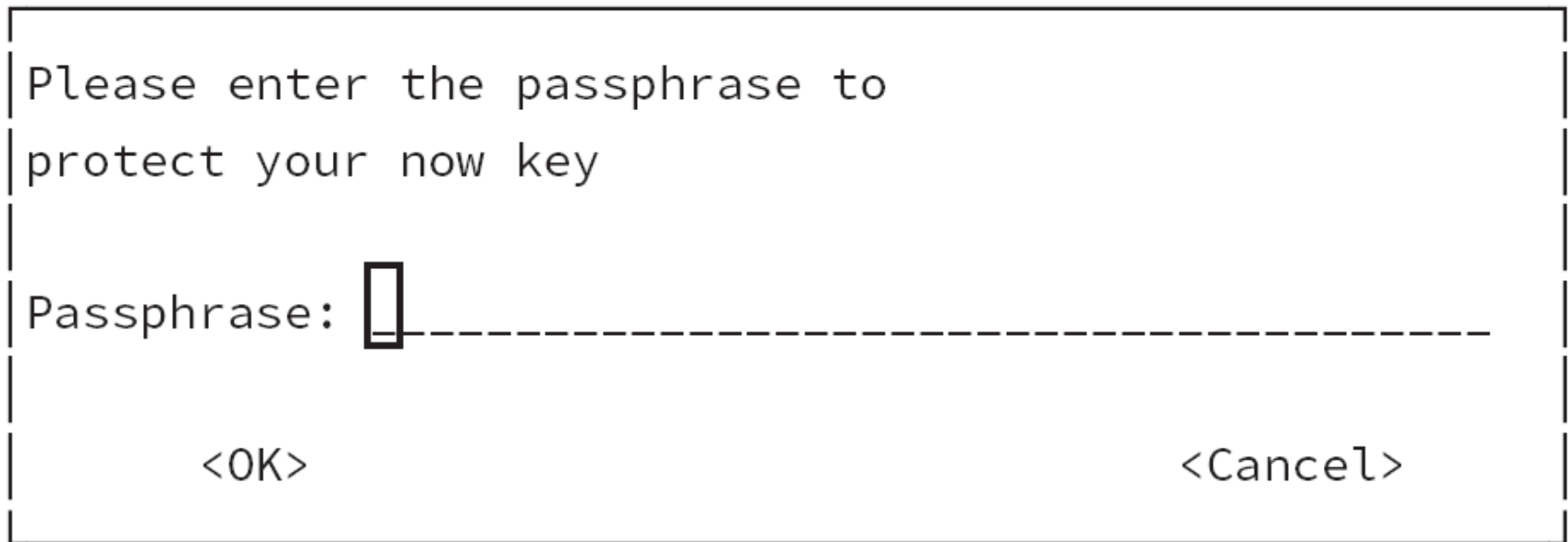
Real name: Alice ← 이름 입력
Email address: alice@example.com ← 메일 주소 입력
Comment: Alice Liddell ← 코멘트 입력
You selected this USER-ID:
  "Alice (Alice Liddell) alice@example.com"

Change (N)ame, (C)omment, (E)mail or (O)key/(Q)uit? 0 ← 이것으로 OK
(여기에 그림 13-2와 같이 패스 프레이즈 입력화면을 나타냄)
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /User/alice/.gnupg/trustdb.gpg: trustdb created ← 「신뢰의 망」 데이터베이스를 생성
gpg: key A57FF192 marked as ultimately trusted ← 자기 자신의 키 등으로 「완전 신뢰」로 취급

gpg: directory '/Users/alice/.gnupg/openpgp-revocs.d' created ← 무효 인증서를 저장할 장소를 생성
public and secret key created and signed

gpg: checking the trustdb
gpg: 3 marginal (s) needed, 1 complete (s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m 0f, 1u
gpg: next trustdb check due at 2016-06-04
pub  rsa2048/A57FF192 2015-06-05 [expires: 2016-06-04] ← "A57FF192"라는 키가 생성
     key fingerprint = D00A 09E6 OEE8 3680 0E04 2BEE 81AD 4C36 A57F F192
uid      [ultimate] Alice (Alice Lindell)alice@example.com
sub  rsa2048/32D4291D 2015-06-05 [expires: 2016-06-04]
```

패스 프레이즈 입력 화면



Please enter the passphrase to
protect your now key

Passphrase:

<OK> <Cancel>

그림 14-2 • 패스 프레이즈 입력 화면

공개 키의 예

\$ gpg2 --expert --armor A57FF192 ← “A5A57FF192”라는 키를 지정해서 표시

-----BEGIN PGP PUBLIC KEY BLOCK-----

Version: GnuPG v2

```
mQENBFVxBfABCADbmYB9Veg8+/PHXIqFIJ47m4DfEHcnMpR5pYLUgE/Jp1/UaPJjINkNtPYA1LD4
VvwrgXM8wyxQBua4dvqfFmgFxm8CED4VBCBHuZu/V+YQc/BjyaxV2zrswNYYvasZRYucX0ssUb3kQ
v1ITMabi7N1wXITsEUpZsnuLy1mZd+hAwd0mEbrWd+xpeeHa67mHucDC9HMs7X1rCb2ktAFnvfToE
ScsbU/DcI3uh54i1zbGnpzHx26+isTBK0nsU2bSrCG3r4oR1ShzJjftvhs12RzI7UWqSpb7poiD9b
126PymBjhtYsPlyckPHfysZ3b7SOSCpyvPoF1X2QCwjM0WzVABEBAdAG0KUFsaWNLICHbBGljZSBM
aWRKZWxsKSA8YWxpY2VAZXhhbXBsZS5jb20+iQBBMBCAAnBQJVCXwAhsDBQkb4TOABQsJCAcCBh
UICQoLAgQWAgMBAh4BAheAAAoJEIGtTDa1f/GSkSsH/0rahwa01H1Kt8rdxCrFoZLQ87tSi+oacL
/9UP/2ZA/dRRG68t/atoB22Po6C8pTR8CdIFJKv2s9hxIyGUEVbjijsiyk1zgiFCihHZu+qNBA3iW
NWU63H26TpML9F1abiWik/wYjBXC3ky46shHCZh6L2mhcfJP3waeHzpxV5Bxg9FTSLVfYqDF1y+
sSgD1q4TbrCfs2DQ+7EU4zd0MoP1NJaG9uqqLTqc4p6q9zdou6fjro10Py2mOWCU51FVWYqoyRK/
mU1BJ/Yaf/20s//ZtnX2IN3UUCijZQvUztbNHwxe21h/u7R+9tGgmwnTo66QULLkHnzVkIghM9Y7
p1rFtHzNS34c+SNc9tLMsfS+5AQ0EVXEF8AEIA02hzb1ISOIXsi3wwLIQaqshuGzPWB9qXGuHIAyT
4E32xMWT8oeWDyUp0fXQU1YZZbndWJmaydrwHHTdw2xffG5i4vrxCCLrvSt13tICImDVH05DnHBQ
BQyMEwBBJvEenZuBraWgXeG8c/AwQa1wY8awehfTiZMJsLYRyQToJTe1BNiv3TrMALHe34wQtn3
hq0aC/9WHLvP61PPQU2MIrG+FGd1DxgksQ0xgksQ0X5fCeVFTcmJQTtiGBQoWzKPr/GRZXBODer
BqBnnKwbJkKonum+RrNrh+sH9ej2VQhw2b4YRLQWPy5AYvCo8rsrp1s3LqkTz3Z1TNBD0QrdqNF
ODSuLTshYfdY23ddci2jH8sAEQEAAyKBJQYQAQgADwUCVXEF8AIBDAUJAeEzgAAKCRCBruW2pX/
xkivyB/91wDdcKnHtvU81v/Q3YY1PJMKKFD05TubmJ0qJVSskLKjAIocnbqiBoxu02BJzt76xuLJ
KWOn/24nkEak3tGT7sskto/zNbYrbsfETuUKMBH3fr+uHtTod9ebjE2i7cF3bNt0td0I9D7RyeaMS
mp2on2/dILjeeXFyJheFD6cIYh/7mrE2nQtn2+CtWmluzS+1G1iu5uyhNs3ysvs/zyxjuf9gxMrwk
zo29nrxrZTYMyOBBD2RCsMbvsSiS4gGUUVS4iit8hPG2j0tXDqVxC/CT82nr4sdPaFCJ018jt8wiNFG
2zSg3RBAHRfNveEajGewzr0sY+GtDfkfUruszLTcLinyPvsnzF6LvRa76bFs2qLPqA0eGbXDWIn2h
02wHBT5mjM
```

=0rVX

-----END PGP PUBLIC KEY BLOCK-----

Section 03

암호화와 복호화

3.1 암호화

3.2 복호화

3.1 암호화

- 세션 키 생성과 암호화
 - 1) 의사난수 생성기를 사용해서 세션 키를 생성
 - 2) 세션 키를 수신자 공개키로 암호화
- 메시지 압축과 암호화
 - 1) 메시지를 압축
 - 2) 압축한 메시지를 세션키로 대칭 암호화
 - 3) 암호화한 세션 키(절차(2))와, 암호화한 메시지(절차(4))를 결합
 - 4) 절차(5)의 결과를 텍스트 데이터로 변환
변환한 결과가 송신 데이터

PGP 암호화

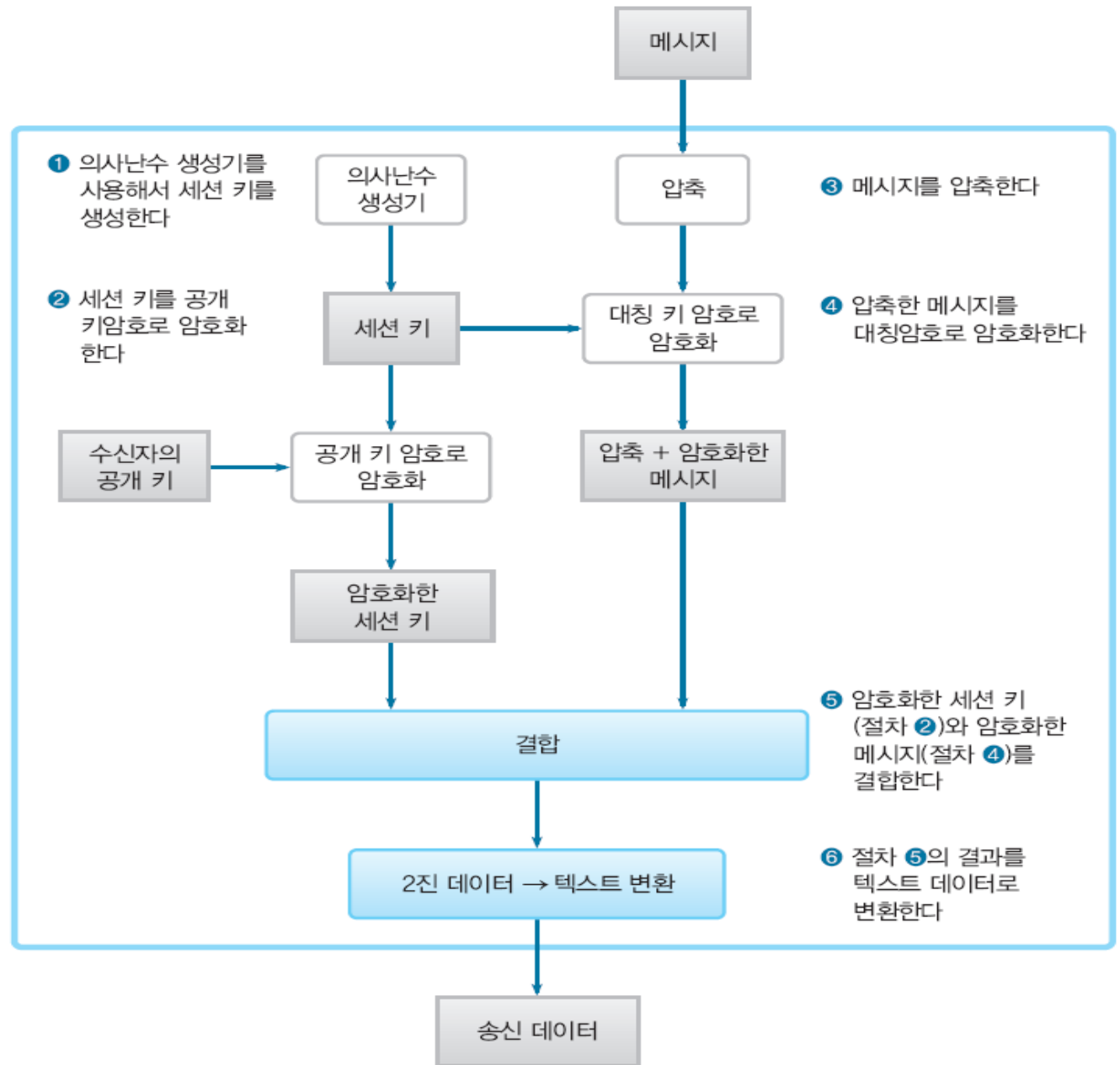


그림 14-4 • PGP에 의한 암호화

3.2 복호화

- 송신 데이터를 수신한 수신자가 원래의 메시지를 얻는 과정
 - 개인 키 복호화
 - 세션 키 복호화
 - 메시지의 복호화와 확장

개인키 복호화

- 1) 수신자는 복호화를 위한 패스 프레이즈를 입력
- 2) 패스 프레이즈의 해시 값을 취해 개인 키를 복호화하기 위한 키를 생성
- 3) 키 고리 안에 있는 암호화되어 있는 개인 키를 복호화

세션키 복호화

- 4) 수신 데이터(텍스트 데이터)를 이진 데이터로 변환
- 5) 2진 데이터를, 암호화되어 있는 세션 키와 압축+암호화되어 있는 메시지로 분해
- 6) 암호화되어 있는 세션 키를 공개 키 암호로 복호화
 - 절차(3)에서 생성한 수신자의 개인 키를 사용한다.

메시지 복호화와 확장

- 7) 절차(5)에서 얻은 압축+암호화되어 있는 메시지를 대칭 암호로 복호화
 - 이 때 절차(6)에서 생성한 세션 키를 사용
- 8) 절차(7)에서 얻은 압축 메시지를 확장
- 9) 이것으로 메시지 획득

PGP 복호화

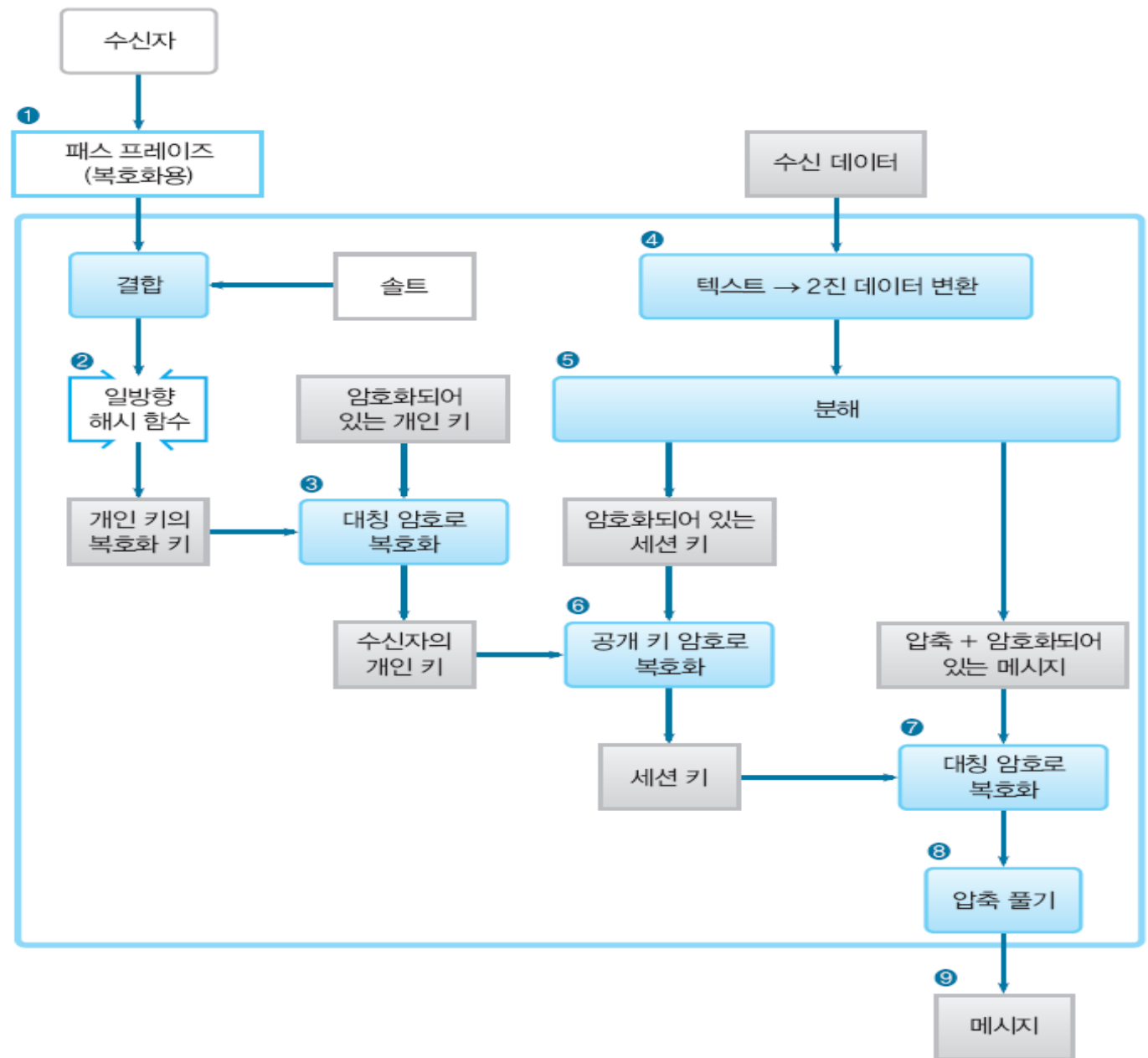


그림 14-5 • PGP에 의한 복호화

암호화와 복호화

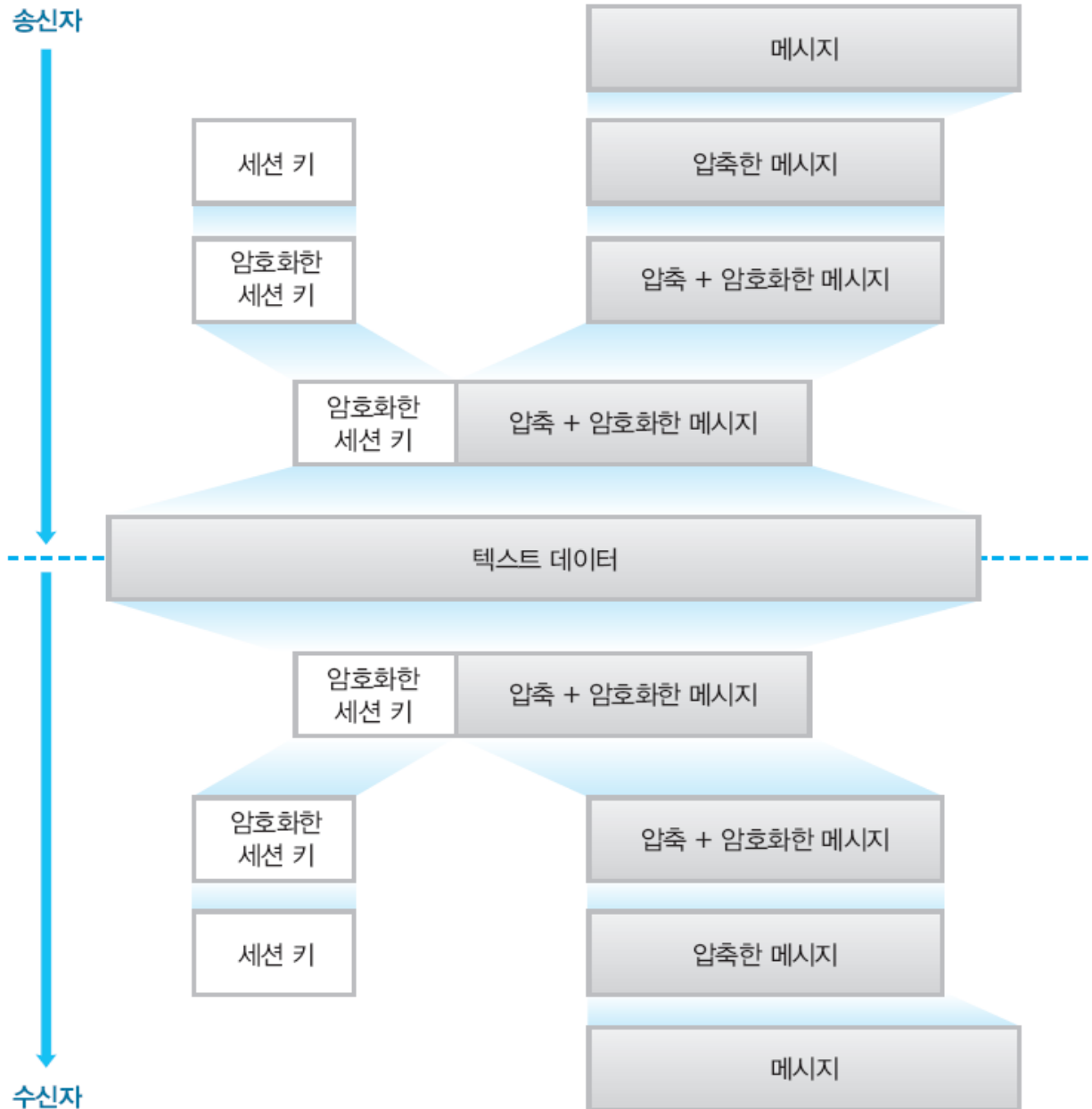


그림 14-6 • 암호화와 복호화

Quiz 1 압축과 암호화의 순서

그림 14-4를 보면 메시지의 압축을 행하고 나서 암호화를 하고 있다. 어째서 압축->암호화의 순으로 처리를 하고 있는 것일까?

Section 04

디지털 서명 작성과 검증

4.1 디지털 서명 작성

4.2 디지털 서명 검증

4.1 디지털 서명 작성

- 메시지와 그 메시지에 대한 서명을 결합해서 송신 데이터(텍스트 데이터)로 변환
- 송신 데이터를 텍스트 데이터로 할지 어떨지는 PGP를 사용할 때 선택
- 개인 키 복호화
- 디지털 서명 작성

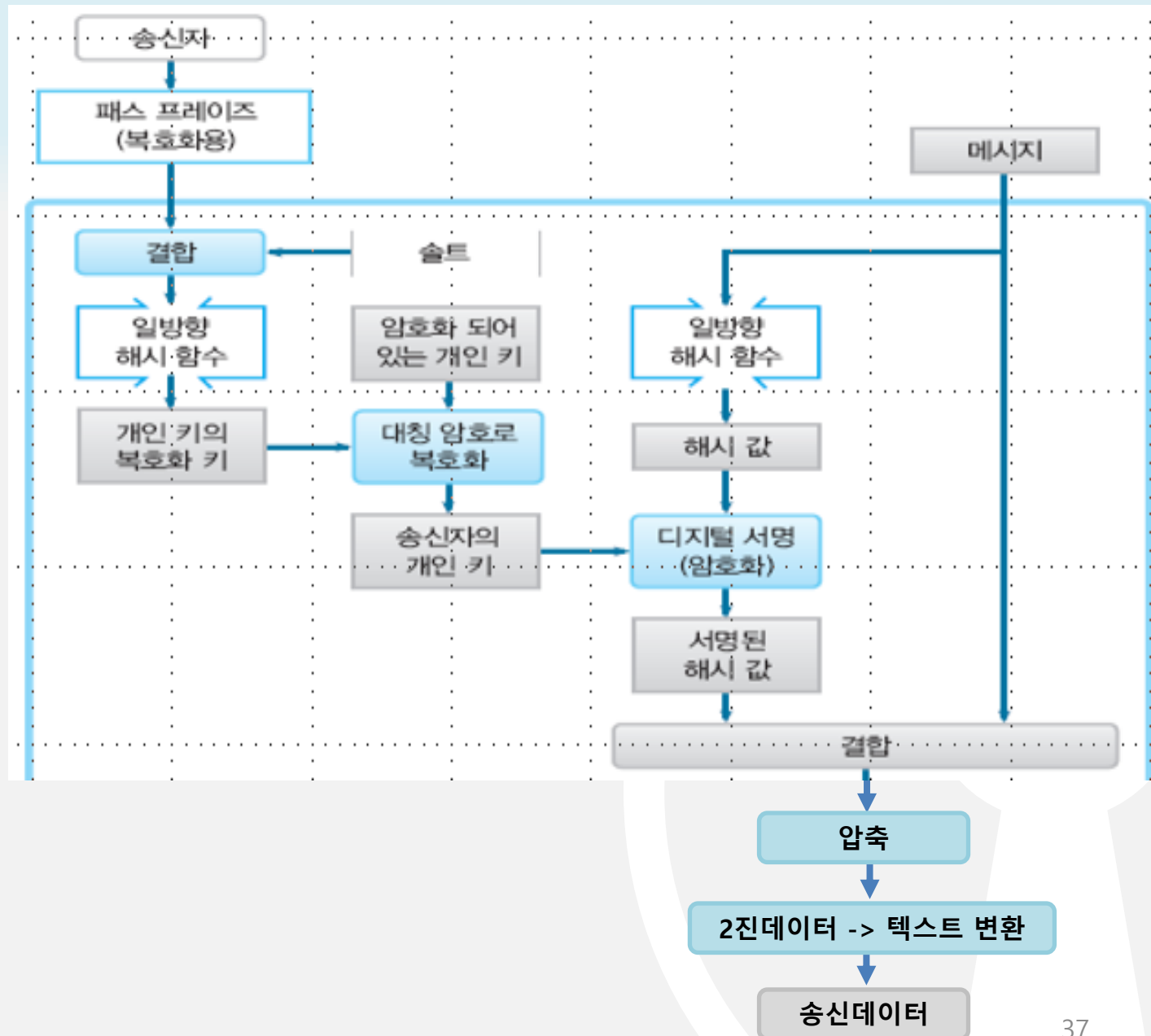
개인 키 복호화

- 1) 송신자는 서명을 위한 패스 프레이즈를 입력
- 2) 패스 프레이즈의 해시 값을 취해 개인 키를 복호화하기 위한 키를 생성
- 3) 키 고리 안에 있는 암호화되어 있는 개인 키를 복호화

디지털 서명 작성

- 4) 일방향 해시 함수를 사용해서 메시지의 해시 값을 계산
- 5) 절차(4)에서 얻은 해시 값에 서명
- 6) 절차(5)에서 작성한 디지털 서명과 메시지를 결합
- 7) 절차(6)의 결과를 압축
- 8) 절차(7)의 결과를 텍스트 데이터로 변환
- 9) 절차(8)의 결과가 송신 데이터

PGP에 의한 디지털 서명 작성



4.2 디지털 서명 검증

- 작성한 송신 데이터를 수신한 수신자가 원래의 메시지를 얻어 디지털 서명을 검증
- 수신한 해시 값 복원
- 해시 값 비교

수신한 해시 값 복원

- 1) 수신 데이터(텍스트 데이터)를 2진 데이터로 변환
- 2) 압축되어 있는 데이터를 확장
- 3) 신장한 데이터를 서명되어 있는 해시 값과 메시지로 분해
- 4) 서명되어 있는 해시 값(암호화되어 있는 해시 값)을 송신자의 공개 키를 사용해서 복호화하고, 보내 온 해시 값을 복원

해시 값 비교

- 5) 절차(3)에서 분해한 메시지를 일방향 해시 함수에 부여하여 해시 값을 계산
- 6) 절차(4)에서 얻은 해시 값과 절차(5)에서 얻은 해시 값을 비교
- 7) 절차(6)의 결과가 같으면 디지털 서명 검증 성공, 같지 않으면 검증 실패
 - 이것이 디지털 서명의 검증 결과
- 8) 절차(3)에서 분해한 메시지가 송신자의 메시지

PGP에 의한 디지털 서명 검증

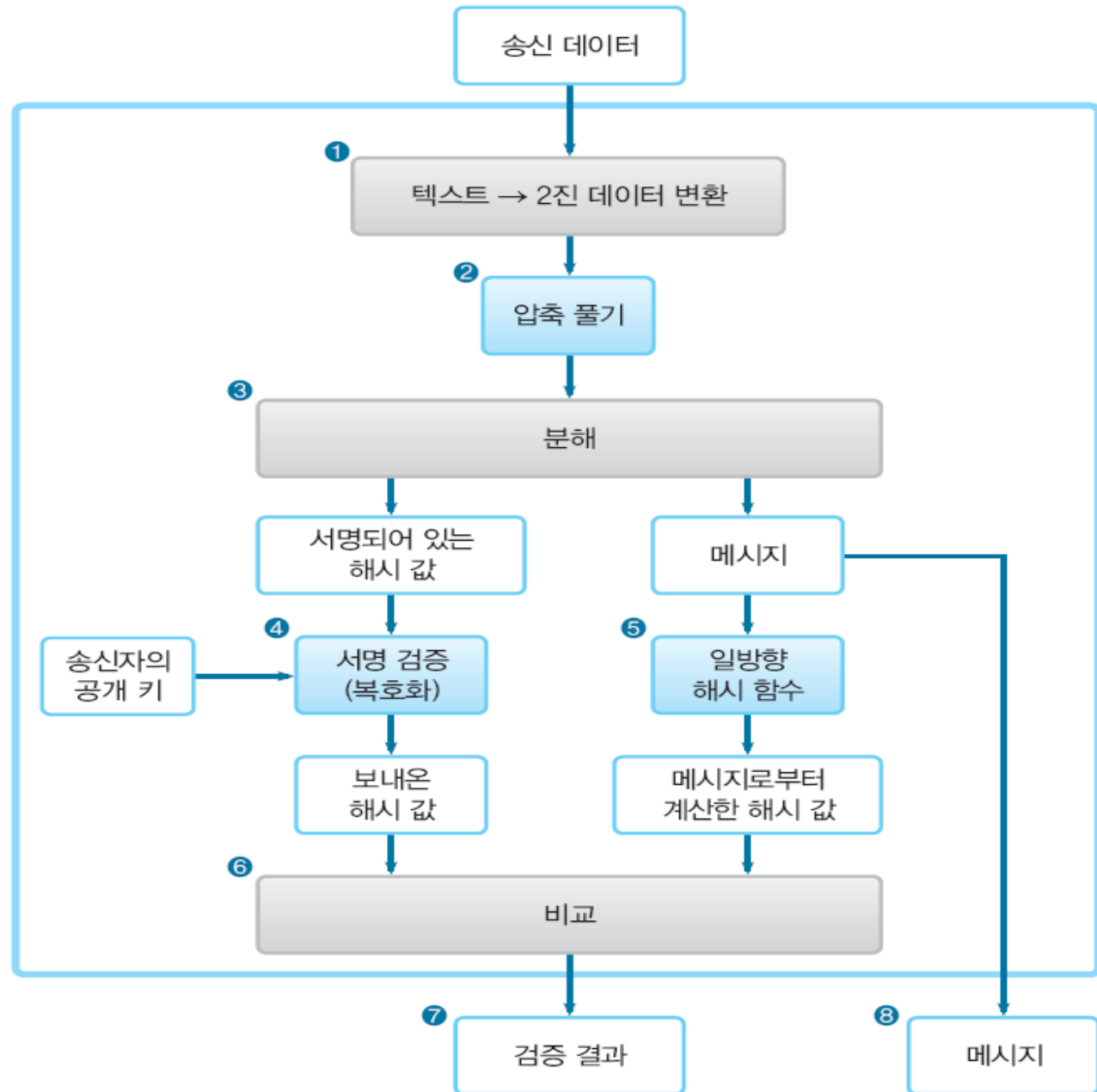


그림 14-8 • PGP에 의한 디지털 서명 검증

디지털 서명 작성과 검증

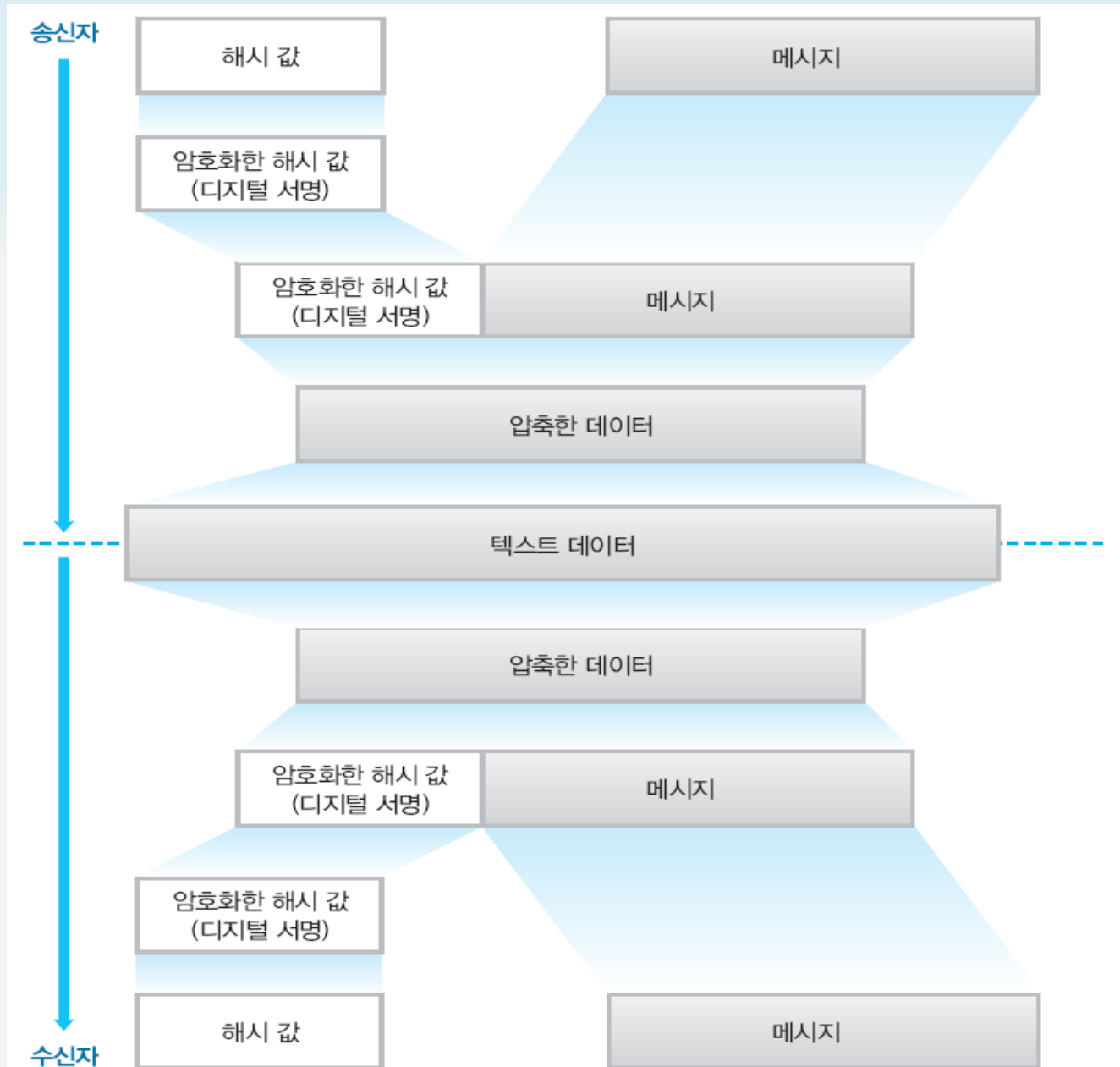


그림 14-9 • 디지털 서명 작성과 검증

Section 05

디지털 서명 작성과 암호화 및 복호화한 디지털 서명 검증

5.1 디지털 서명 작성과 암호화

5.2 복호화와 디지털 서명 검증

5.1 디지털 서명 작성과 암호화

- 디지털 서명 작성
 - 메시지에 대한 디지털 서명 작성
- 암호화
 - 암호화의 대상이 되는 것
 - 메시지
 - 디지털 서명과 메시지를 결합한 데이터

5.2 복호화와 디지털 서명 검증

- 복호화
 - 복호화에서 얻어지는 것
 - 메시지
 - 디지털 서명과 메시지를 결합한 데이터
- 디지털 서명 검증

PGP에 의한 디지털 서명 작성과 암호화

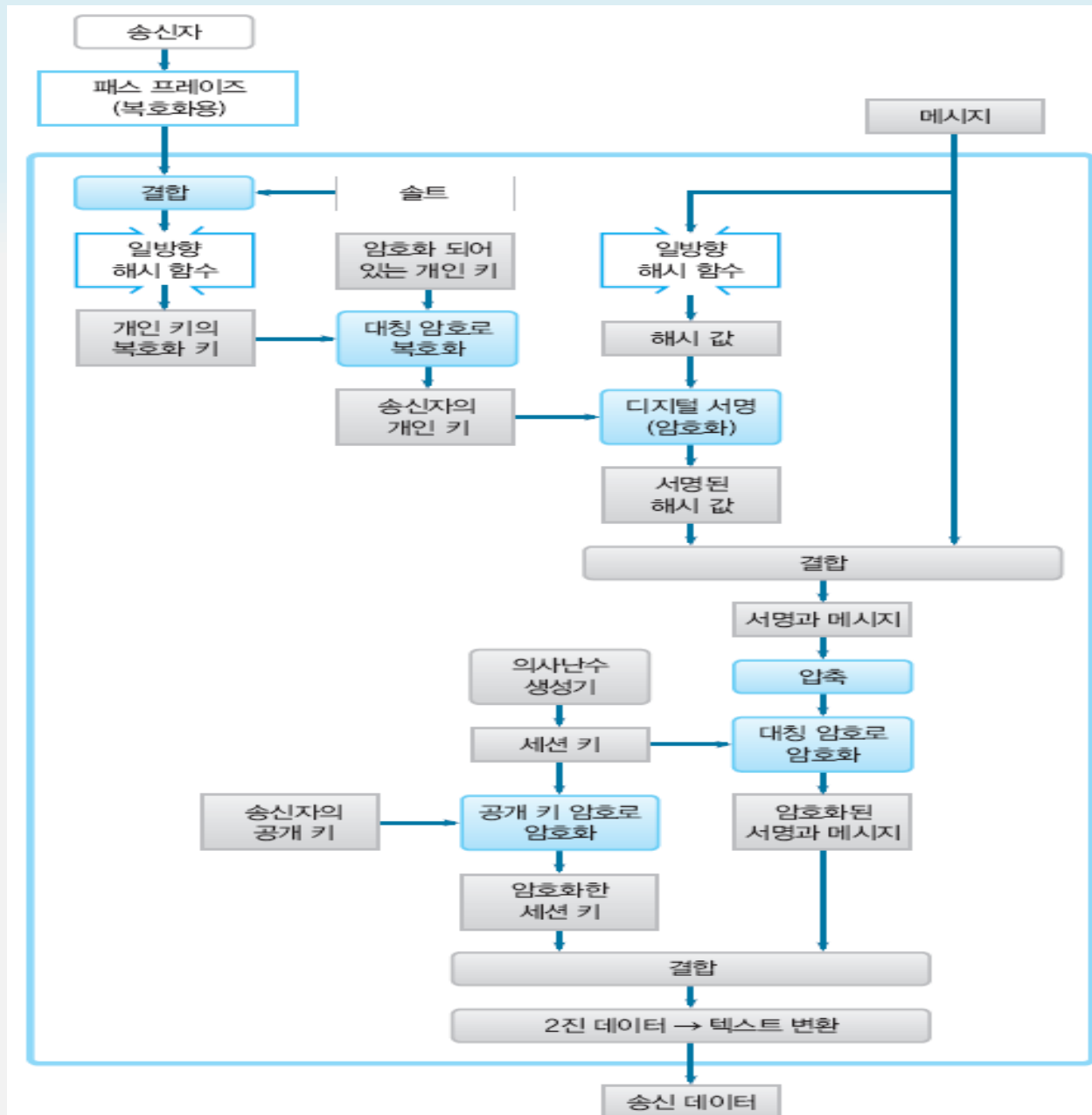


그림 14-10 • PGP에 의한 디지털 서명 작성과 암호화

PGP에 의한 복호화와 디지털 서명 검증

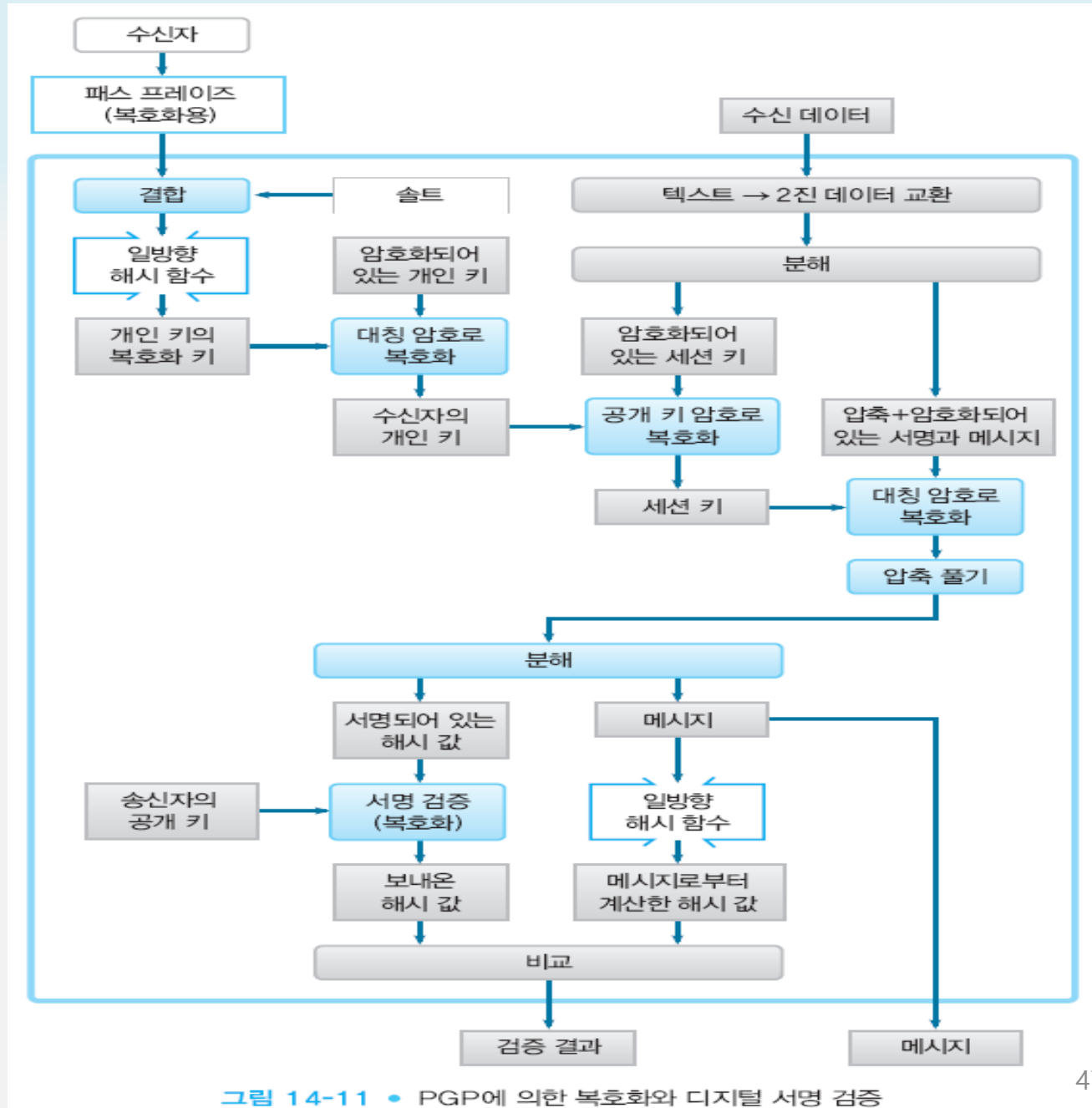
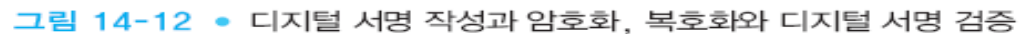


그림 14-11 • PGP에 의한 복호화와 디지털 서명 검증



Section 06

신뢰망

6.1 공개 키의 정당성

6.2 경우1: 자기 자신의 디지털 서명에 의해 확인한다

6.3 경우2: 자신이 항상 신뢰하고 있는 사람의 디지털 서명에 의해 확인한다

6.4 경우3: 자신이 부분적으로 신뢰하고 있는 사람들의 디지털 서명에 의해 확인한다

6.5 공개 키의 정당성과 소유자 신뢰는 별개

6.6 소유자 신뢰 값은 개인적인 것

6.1 공개 키의 정당성

• 공개 키의 정당성

- 입수한 공개 키가 정말로 자신이 생각하고 있는 인물의 공개키인지 확인 필요
- 중간자 공격의 피해 가능
- 확인 방법
 - 인증서
- PGP는 인증기관을 이용하지 않는다
- PGP에서는 **신뢰 망**(web of trust) 방법을 이용

신뢰 망(web of trust)

- PGP 사용자가 서로의 공개 키에 대해 서로 디지털 서명을 하는 방법
- 인증기관을 설치하지 않고 개인끼리 신뢰를 확립
- 어느 키를 신용할지를 자신이 결정

PGP 신뢰 망 구성

- **경우 1** : 자기 자신의 디지털 서명으로 확인
- **경우 2** : 자신이 항상 신뢰하고 있는 사람의 디지털 서명으로 확인
- **경우 3** : 자신이 부분적으로 신뢰하고 있는 사람들의 디지털 서명으로 확인

6.2 경우1: 자기 자신의 디지털 서명에 의해 확인한다

- 앨리스는 밥으로부터 직접 밥의 공개키를 입수
- 앨리스는 밥의 공개 키에 자신의 개인 키로 직접 디지털 서명 실시
- **의미:** 앨리스 자신이 가지고 있는 밥의 공개키는 정당한 공개 키임을 주장

밥의 서명이 붙은 메시지를 PGP로 검증하기

- 1) 밥의 디지털 서명 검증을 위해 PGP는 앨리스의 공개 키 고리에서 밥의 공개 키를 검색
- 2) 앨리스의 공개 키 고리에는 밥의 공개 키가 존재
- 3) PGP는 밥의 공개 키에 앨리스의 디지털 서명이 붙어 있는 것을 확인
- 4) 앨리스의 디지털 서명 검증을 위해 PGP는 앨리스의 공개 키 고리에서 앨리스 자신의 공개 키를 검색
- 5) 앨리스의 공개 키 고리에는 앨리스 자신의 공개 키가 존재
- 6) PGP는 앨리스의 공개 키를 사용해서 밥의 공개 키에 서명된 앨리스의 디지털 서명을 검증
 - 검증에 성공하면 이것은 정당한 밥의 공개 키라고 확신
- 7) PGP는 정당한 밥의 공개 키를 사용해서 밥의 메일에 서명된 밥의 디지털 서명을 검증

Quiz 2 공개키의 정당성 확인 수단

앨리스에게는 자주 이야기하는 엘마라는 친구가 있다. 어느 날, 엘마한테 메일이 와서 『이것이 내 공개키야』라고 하는 메시지와 공개키가 첨부되어 있었다.

이 메일은 적극적 공격자 맬로리에 의해 조작되어 있을지도 모른다고 앨리스는 생각했다. 그래서 앨리스는 엘마에게 전화를 걸어, 보내 온 공개키가 엘마 본인이 것인지 알아보려고 했다.

그런데 보내 온 공개키 파일이 너무 커서 1바이트씩 전화로 읽어서 체크하는 것이 힘들다. 어떻게 하면 좋을까?

6.3 경우2: 자신이 항상 신뢰하고 있는 사람의 디지털 서명으로 확인

- 앨리스의 공개 키 고리 안에는 앨리스의 디지털 서명이 붙은 트렌트의 공개 키가 존재
- 앨리스는 트렌트를 **심리적으로 신뢰**한다
 - “트렌트가 디지털 서명한 공개 키는 반드시 정당한 것임에 틀림없다”라고 믿는다
 - 믿는 정도는 앨리스가 결정한다
 - 100%~0% 사이(?)

신뢰 정도

- 소유자 신뢰(owner trust):
 - 트랜트에 대한 앨리스의 신뢰도를 표현(예: 항상 신뢰: fully trust)하고 그 정보에 앨리스가 디지털 서명을 한다
 - 캐롤의 공개키를 얻었는데 트랜트의 서명이 붙어 있다면 캐롤의 공개키를 정당한 것으로 판단한다
 - 트랜트에 대한 신뢰도가 항상 신뢰이기 때문
 - 앨리스는 트랜트를 소개자로서 신뢰하기 때문에 캐롤의 공개키를 정당한 것으로 받아들인다

소유자 신뢰(owner trust) 값

표 14-2 소유자 신뢰(owner trust) 값

Ultimately trusted	완전히 신뢰한다(개인 키를 가지고 있는 본인이다)
Fully trusted	항상 신뢰한다
Marginally trusted	부분적으로 신뢰한다
Never trust this key	신뢰하지 않는다
Not enough information	미지의 키이다
No ownertrust assigned	미설정

6.4 경우3: 자신이 부분적으로 신뢰하는 사람들의 디지털 서명으로 확인

- 여러 명의 소유자 신뢰 합계로 공개 키의 정당성을 확인하는 방법
- 앨리스는 딥과 프래드를 부분적으로 신뢰 (trust marginally)한다
- 이 두 사람이 모두 디지털 서명한 제 3자의 공개키는 정당한 것으로 받아들인다
 - 딥과 프래드에 대한 소유자 신뢰 값을 더해서 결정

신뢰 값은 어떻게 결정?

- 신뢰 값은 앨리스 스스로 결정하는 값이다
- 전적으로 사회적 관계 속에서 결정되는 값이다
- 따라서 매우 주관적일 수 있다

6.5 공개 키의 정당성과 소유자 신뢰는 별개

- 「공개 키는 정당한가」라는 문제와, 「소유자를 신뢰하는가」라고 하는 문제는 별개
- 앨리스는 밥에게 받은 공개 키를 정당한 것이라고 생각한다.
 - 왜냐하면 직접 건네받은 것이기 때문
- 하지만 앨리스는 디지털 서명에 관한 밥의 판단력은 신뢰하지 않을 수 있다
- 따라서 밥에 대한 소유자 신뢰값은 신뢰하지 않음 (do Not trust)로 설정한다
 - 나중에 밥이 서명한 잉게의 공개키를 입수하면 잉게의 공개키를 신뢰하지 않는다

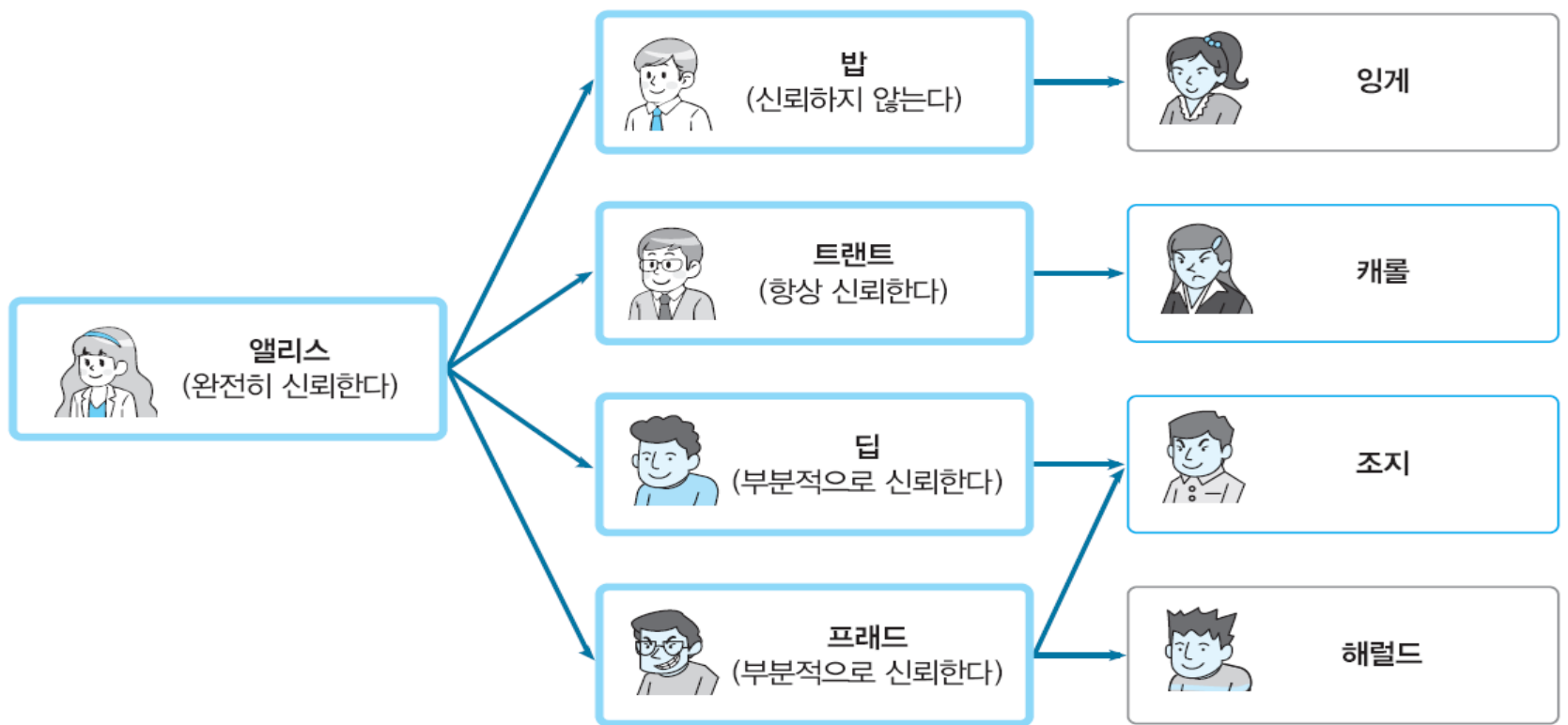
6.6 소유자 신뢰 값은 개인적인 것

- 엘리스는 엘리스 자신의 공개 키에 디지털 서명을 하고 있다 (소유자 신뢰 값은 「완전히 신뢰한다」).
- 엘리스는 밥의 공개 키에 디지털 서명을 하고 있다 (소유자 신뢰 값은 「신뢰하지 않는다」).
- 엘리스는 트랜트의 공개 키에 디지털 서명을 하고 있다 (소유자 신뢰 값은 「항상 신뢰한다」).
- 엘리스는 딥의 공개 키에 디지털 서명을 하고 있다 (소유자 신뢰 값은 「부분적으로 신뢰한다」).
- 엘리스는 프래드의 공개 키에 디지털 서명을 하고 있다 (소유자 신뢰 값은 「부분적으로 신뢰한다」).
- 트랜트는 캐롤의 공개 키에 디지털 서명을 하고 있다.
- 딥은 조지의 공개 키에 디지털 서명을 하고 있다.
- 프래드는 조지의 공개 키에 디지털 서명을 하고 있다.
- 프래드는 해럴드의 공개 키에 디지털 서명을 하고 있다.
- 밥은 잉게의 공개 키에 디지털 서명을 하고 있다.

공개키에 대한 정당성

- 엘리스. 밥. 트랜트. 딥. 프래드. 캐롤. 조지의 공개 키를 정당하다고 판단
- 해럴드와 잉게의 공개 키는 정당하다고 판단하지 않는다
- 「어느 키의 소유자를 얼마만큼 신뢰할지」는 매우 사적으로 결정
- 엘리스는 누구를 어느 정도 신뢰할지 전적으로 자신이 설정한다

앨리스의 「신뢰 망」



앨리스가 직접 서명한 정당한 키



정당한 키라고 판단할 수 있는 키



정당한 키라고 판단할 수 없는 키



A는 B의 공개 키에 서명하고 있다

그림 14-13 • 앨리스의 「신뢰 망」

Quiz 3 많은 디지털 서명

- 앨리스가 제리라고 하는 사람의 공개키를 입수했더니, 그 공개 키에는 10개의 정당한 서명이 붙어 있었다. 이 공개키는 정당하다고 판단할 수 있는가?

Quiz 4 PGP의 기초 지식

- 다음 문장 중 바른것에는 O, 잘못된 것에는 X를 표시하십시오.
 - (1) PGP에서는 데이터를 압축해서 암호화한다.
 - (2) PGP사용자는 <http://www.pgp.com/>에 설치되어 있는 인증기관을 이용한다.
 - (3) PGP를 사용하면 자신의 공개 키와 개인 키 쌍을 만들 수 있다.
 - (4) PGP에서 암호화를 할 때에는 대칭 암호 알고리즘은 사용하지 않고, 공개 키 암호 알고리즘을 사용한다.
 - (5) PGP에서 디지털 서명 검증을 할 때에는 패스 프레이즈를 입력할 필요가 있다.
 - (6) PGP에서 디지털 서명 작성과 암호화를 할 때에는 송신자는 패스 프레이즈를 입력할 필요가 있다.