증빙 — 측정 스크립트 전문과 실행 출력 원문

본편 ③의 모든 숫자는 아래 스크립트의 출력입니다. 파이썬만 있으면 누구나 같은 표를 재현할 수 있습니다.

증빙 1. 실행 출력 원문 (python3 scrub_test.py)

손대지 않은 표준출력 전문입니다.

```
시험 문장 총 59건 (개인정보 포함 36건 / 정상 23건)

===== 운영본 v7 정규식 이식본 (서버 ai-chat 기준 · 2026-09-16 소스 대조 완료) =====
개인정보 포함 문장 36건 중 마스킹 성공 24건 / 실패 12건  → 탐지율 66.7%
정상 문장 23건 중 오탐 5건 → 오탐률 21.7%
-- 못 막은 유형 12건 --
   · [휴대폰/점] 번호 010.1234.5678 입니다
   · [휴대폰/국제] +82 10-1234-5678 로 연락 가능해요
   · [휴대폰/국제붙임] +821012345678 입니다
   · [휴대폰/82하이픈] 82-10-1234-5678 로 부탁드려요
   · [유선/무구분] 0212345678 로 걸면 되나요
   · [유선/점] 02.123.4567 입니다
   · [계좌번호] ○○은행 123456-01-234567 로 입금했어요
   · [계좌번호2] △△은행 110-123-456789 입금 확인해주세요
   · [한글숫자전화] 공일공 일이삼사 오육칠팔 입니다
   · [차량번호] 12가3456 차량인데 주차 되나요
   · [여권번호] 여권 M12345678 로 예약했어요
   · [주소] ○○구 △△아파트 209동 404호 인데 배달되나요
-- 오탐 5건 --
   · [정상문장/긴숫자] 주문번호 2026 0915 0001 0042 확인 부탁드려요  →  주문번호 [카드번호] 확인 부탁드려요
   · [정상문장/긴숫자] 예약번호 1234567890123456 로 조회되나요  →  예약번호 [카드번호] 로 조회되나요
   · [정상문장/긴숫자] 쿠폰코드 2024-0101-2345-6789 쓸 수 있나요  →  쿠폰코드 [카드번호] 쓸 수 있나요
   · [정상문장/긴숫자] 기프티콘 번호 3812 4455 9900 1234 사용 가능한가요  →  기프티콘 번호 [카드번호] 사용 가능한가요
   · [정상문장/긴숫자] 영수증 번호 20260916 1430 0007 확인해주세요  →  영수증 번호 [카드번호] 확인해주세요

===== 개선안 v8 (구분자 확장·국제표기·유선 무구분) =====
개인정보 포함 문장 36건 중 마스킹 성공 30건 / 실패 6건  → 탐지율 83.3%
정상 문장 23건 중 오탐 6건 → 오탐률 26.1%
-- 못 막은 유형 6건 --
   · [계좌번호] ○○은행 123456-01-234567 로 입금했어요
   · [계좌번호2] △△은행 110-123-456789 입금 확인해주세요
   · [한글숫자전화] 공일공 일이삼사 오육칠팔 입니다
   · [차량번호] 12가3456 차량인데 주차 되나요
   · [여권번호] 여권 M12345678 로 예약했어요
   · [주소] ○○구 △△아파트 209동 404호 인데 배달되나요
-- 오탐 6건 --
   · [정상문장/긴숫자] 주문번호 2026 0915 0001 0042 확인 부탁드려요  →  주문번호 [카드번호] 확인 부탁드려요
   · [정상문장/긴숫자] 예약번호 1234567890123456 로 조회되나요  →  예약번호 [카드번호] 로 조회되나요
   · [정상문장/긴숫자] 쿠폰코드 2024-0101-2345-6789 쓸 수 있나요  →  쿠폰코드 [카드번호] 쓸 수 있나요
   · [정상문장/긴숫자] 기프티콘 번호 3812 4455 9900 1234 사용 가능한가요  →  기프티콘 번호 [카드번호] 사용 가능한가요
   · [정상문장/긴숫자] 02 123 4567 이 번호 맞나요  →  [전화] 이 번호 맞나요
   · [정상문장/긴숫자] 영수증 번호 20260916 1430 0007 확인해주세요  →  영수증 번호 [카드번호] 확인해주세요

>>> 탐지율 66.7% → 83.3%  (오탐 5건 → 6건)

===== 개선안 v9 (v8 + 카드번호 룬 검증 · 문맥어 제외) =====
개인정보 포함 문장 36건 중 마스킹 성공 30건 / 실패 6건  → 탐지율 83.3%
정상 문장 23건 중 오탐 1건 → 오탐률 4.3%
-- 못 막은 유형 6건 --
   · [계좌번호] ○○은행 123456-01-234567 로 입금했어요
   · [계좌번호2] △△은행 110-123-456789 입금 확인해주세요
   · [한글숫자전화] 공일공 일이삼사 오육칠팔 입니다
   · [차량번호] 12가3456 차량인데 주차 되나요
   · [여권번호] 여권 M12345678 로 예약했어요
   · [주소] ○○구 △△아파트 209동 404호 인데 배달되나요
-- 오탐 1건 --
   · [정상문장/긴숫자] 02 123 4567 이 번호 맞나요  →  [전화] 이 번호 맞나요

>>> 탐지율 66.7% → 83.3% → 83.3%
>>> 오탐률 21.7% → 26.1% → 4.3%
```

증빙 2. 측정 스크립트 전문 — 시험 문장 59건 포함

v7은 서버 함수의 정규식 5개를 2026-09-16에 현재 소스와 한 줄씩 대조한 뒤 옮긴 것이고, v8·v9는 개선안입니다.

```
# -*- coding: utf-8 -*-
"""
VI One — ai-chat v7 scrubPII 커버리지 실측
운영 코드(app/supabase/functions/ai-chat/index.ts, v7)의 정규식을 그대로 파이썬으로 옮겨
소상공인 매장에 실제로 올 법한 손님 문의 문장에 대해 개인정보 마스킹률을 측정한다.
실행: python3 scrub_test.py
"""
import re, json, sys

# ── 운영 v7 정규식 (index.ts 19~26행과 1:1 대응) ──────────────────────
V7 = [
    (re.compile(r'\d{6}[-\s]?[1-4]\d{6}'),                          '[주민번호]'),
    (re.compile(r'01[016789][-\s]?\d{3,4}[-\s]?\d{4}'),             '[전화]'),
    (re.compile(r'[\w.+-]+@[\w-]+\.[\w.-]+'),                       '[이메일]'),
    (re.compile(r'\b0\d{1,2}-\d{3,4}-\d{4}\b'),                     '[전화]'),
    (re.compile(r'\b\d{4}[-\s]?\d{4}[-\s]?\d{4}[-\s]?\d{4}\b'),    '[카드번호]'),
]
# ── v8 개선안: 구분자 확장(점)·국제표기·유선 무구분 추가 ──────────────
V8 = [
    (re.compile(r'\d{6}[-\s.]?[1-4]\d{6}'),                         '[주민번호]'),
    (re.compile(r'(?:\+?82[-\s.]?|0)1[016789][-\s.]?\d{3,4}[-\s.]?\d{4}'), '[전화]'),
    (re.compile(r'[\w.+-]+@[\w-]+\.[\w.-]+'),                       '[이메일]'),
    (re.compile(r'\b0\d{1,2}[-\s.]?\d{3,4}[-\s.]?\d{4}\b'),        '[전화]'),
    (re.compile(r'\b\d{4}[-\s.]?\d{4}[-\s.]?\d{4}[-\s.]?\d{4}\b'),'[카드번호]'),
]
def scrub(s, rules):
    for rx, tag in rules: s = rx.sub(tag, s)
    return s

# ── 시험 문장 ──────────────────────────────────────────────────────────
# pii=True  : 개인정보가 들어 있으므로 반드시 가려져야 하는 문장
# pii=False : 개인정보가 없으므로 절대 가려지면 안 되는 문장(오탐 측정용)
CASES = []
def C(kind, text, pii, must):  CASES.append(dict(kind=kind, text=text, pii=pii, must=must))

# 휴대폰 — 표준 표기
C('휴대폰/하이픈', '예약자 홍길동 010-1234-5678로 확인 부탁드려요', True, '010-1234-5678')
C('휴대폰/하이픈', '제 번호 010-9876-5432입니다 예약 변경해주세요', True, '010-9876-5432')
C('휴대폰/붙임',   '01012345678 이 번호로 예약했는데요', True, '01012345678')
C('휴대폰/공백',   '연락처는 010 2222 3333 입니다', True, '010 2222 3333')
C('휴대폰/011',    '011-234-5678로 연락주세요', True, '011-234-5678')
C('휴대폰/016',    '016-777-8888 입니다', True, '016-777-8888')
C('휴대폰/017',    '017-333-4444로 부탁해요', True, '017-333-4444')
C('휴대폰/018',    '018-555-6666 맞나요', True, '018-555-6666')
C('휴대폰/019',    '019-888-9999 로 문자 주세요', True, '019-888-9999')
C('휴대폰/3자리',  '010-123-4567 예약 확인이요', True, '010-123-4567')
# 휴대폰 — 실제로 자주 쓰이는 변형 (v7 미대응 예상)
C('휴대폰/점',     '번호 010.1234.5678 입니다', True, '010.1234.5678')
C('휴대폰/국제',   '+82 10-1234-5678 로 연락 가능해요', True, '10-1234-5678')
C('휴대폰/국제붙임','+821012345678 입니다', True, '1012345678')
C('휴대폰/82하이픈','82-10-1234-5678 로 부탁드려요', True, '10-1234-5678')
# 유선
C('유선/서울',     '가게 전화 02-123-4567 맞나요', True, '02-123-4567')
C('유선/경기',     '031-1234-5678 로 전화드렸어요', True, '031-1234-5678')
C('유선/070',      '070-1234-5678 이 번호 쓰시나요', True, '070-1234-5678')
C('유선/무구분',   '0212345678 로 걸면 되나요', True, '0212345678')
C('유선/점',       '02.123.4567 입니다', True, '02.123.4567')
# 주민등록번호
C('주민번호/하이픈','주민번호 900101-1234567 필요하신가요', True, '900101-1234567')
C('주민번호/붙임',  '8503151111119 이거 맞나요', True, '8503151111119')
C('주민번호/공백',  '020304 3456789 입니다', True, '020304 3456789')
# 이메일
C('이메일',        '영수증 hong.gildong@example.com 으로 보내주세요', True, 'hong.gildong@example.com')
C('이메일/플러스', 'test+shop@example.com 로 부탁해요', True, 'test+shop@example.com')
C('이메일/기업',   'user_name@example.co.kr 입니다', True, 'user_name@example.co.kr')
# 카드번호
C('카드/하이픈',   '카드 4242-4242-4242-4242 로 결제했어요', True, '4242-4242-4242-4242')
C('카드/붙임',     '5500005555555559 결제건 확인 부탁', True, '5500005555555559')
C('카드/공백',     '4111 1111 1111 1111 로 계산했습니다', True, '4111 1111 1111 1111')
```

```
# 복합
C('복합/전화+이메일','010-1111-2222 이고 메일은 a@b.com 이에요', True, '010-1111-2222')
C('복합/주민+전화', '901231-1234567 이고 010-3333-4444 입니다', True, '901231-1234567')
# v7·v8 모두 못 막을 것으로 예상되는 유형 (정직하게 기록)
C('계좌번호',      '○○은행 123456-01-234567 로 입금했어요', True, '123456-01-234567')
C('계좌번호2', '△△은행 110-123-456789 입금 확인해주세요', True, '110-123-456789')
C('한글숫자전화',  '공일공 일이삼사 오육칠팔 입니다', True, '공일공 일이삼사 오육칠팔')
C('차량번호',      '12가3456 차량인데 주차 되나요', True, '12가3456')
C('여권번호',      '여권 M12345678 로 예약했어요', True, 'M12345678')
C('주소',          '○○구 △△아파트 209동 404호 인데 배달되나요', True, '209동 404호')

# 개인정보 없음 — 오탐(false positive) 측정용
for t in [
    '주차 되나요',
    '영업시간 몇 시까지예요',
    '커트 얼마예요',
    '내일 오후 3시에 2명 예약 가능한가요',
    '2026-09-22 예약 가능할까요',
    '가격이 15,000원 맞나요',
    '남자 커트 20000원인가요',
    '주말에도 여나요 토요일 10시부터 8시까지인가요',
    '반려동물 데려가도 되나요',
    '아이 의자 있나요',
    '사업자등록번호 123-45-67890 세금계산서 발행되나요',
    '1인 2메뉴 인가요',
    '3층이라고 하셨는데 엘리베이터 있나요',
    '예약금 10000원 입금하면 되나요',
    '스탬프 10개 모으면 1회 무료인가요',
]: C('정상문장', t, False, None)


# 오탐 적대 시험 — 예약·고객관리 매장에 실제로 오는 '개인정보 아닌 긴 숫자'
for t in [
    '주문번호 2026 0915 0001 0042 확인 부탁드려요',
    '예약번호 1234567890123456 로 조회되나요',
    '쿠폰코드 2024-0101-2345-6789 쓸 수 있나요',
    '기프티콘 번호 3812 4455 9900 1234 사용 가능한가요',
    '사업자등록번호 012-34-56789 로 계산서 부탁드려요',
    '02 123 4567 이 번호 맞나요',
    '계좌 마지막 네자리 4231 이요',
    '영수증 번호 20260916 1430 0007 확인해주세요',
]: C('정상문장/긴숫자', t, False, None)

# ── 실행 ──────────────────────────────────────────────────────────
def run(rules, label):
    hit=miss=fp=0; missed=[]; fps=[]
    for c in CASES:
        out = scrub(c['text'], rules)
        changed = out != c['text']
        if c['pii']:
            if changed and c['must'] not in out: hit+=1
            else: miss+=1; missed.append((c['kind'], c['text']))
        else:
            if changed: fp+=1; fps.append((c['kind'], c['text'], out))
    n_pii=sum(1 for c in CASES if c['pii']); n_cl=len(CASES)-n_pii
    print(f"\n===== {label} =====")
    print(f"개인정보 포함 문장 {n_pii}건 중 마스킹 성공 {hit}건 / 실패 {miss}건  → 탐지율 {hit/n_pii*100:.1f}%")
    print(f"정상 문장 {n_cl}건 중 오탐 {fp}건 → 오탐률 {fp/n_cl*100:.1f}%")
    if missed:
        print(f"-- 못 막은 유형 {len(missed)}건 --")
        for k,t in missed: print(f"   · [{k}] {t}")
    if fps:
        print(f"-- 오탐 {len(fps)}건 --")
        for k,t,o in fps: print(f"   · [{k}] {t}  →  {o}")
    return hit, miss, fp, n_pii, n_cl

print(f"시험 문장 총 {len(CASES)}건 "
      f"(개인정보 포함 {sum(1 for c in CASES if c['pii'])}건 / 정상 {sum(1 for c in CASES if not c['pii'])}건)")
a=run(V7,'운영본 v7 정규식 이식본 (서버 ai-chat 기준 · 2026-09-16 소스 대조 완료)')
b=run(V8,'개선안 v8 (구분자 확장·국제표기·유선 무구분)')
print(f"\n>>> 탐지율 {a[0]/a[3]*100:.1f}% → {b[0]/b[3]*100:.1f}%  (오탐 {a[2]}건 → {b[2]}건)")

# ── v9 개선안: v8 + 카드번호 룬(Luhn) 검증 + 문맥어 제외 ──────────
def luhn_ok(num: str) -> bool:
```

```
    d = [int(c) for c in num if c.isdigit()]
    if len(d) != 16: return False
    tot = 0
    for i, n in enumerate(reversed(d)):
        if i % 2 == 1:
            n *= 2
            if n > 9: n -= 9
        tot += n
    return tot % 10 == 0

CARD_RX = re.compile(r'\b\d{4}[-\s.]?\d{4}[-\s.]?\d{4}[-\s.]?\d{4}\b')
CTX_RX  = re.compile(r'(주문|예약|쿠폰|기프티콘|영수증|송장|운송장)\s*(번호|코드)?\s*$')

def scrub_v9(s: str) -> str:
    for rx, tag in V8[:4]:          # 주민번호·휴대폰·이메일·유선은 v8 그대로
        s = rx.sub(tag, s)
    def card_sub(m):
        head = s[:m.start()]
        if CTX_RX.search(head): return m.group(0)      # 앞에 주문/예약/쿠폰 등이 오면 두기
        if not luhn_ok(m.group(0)): return m.group(0)  # 룬 검사 실패면 카드가 아님
        return '[카드번호]'
    return CARD_RX.sub(card_sub, s)

def run_v9(label):
    hit=miss=fp=0; missed=[]; fps=[]
    for c in CASES:
        out = scrub_v9(c['text']); changed = out != c['text']
        if c['pii']:
            if changed and c['must'] not in out: hit+=1
            else: miss+=1; missed.append((c['kind'], c['text']))
        else:
            if changed: fp+=1; fps.append((c['kind'], c['text'], out))
    n_pii=sum(1 for c in CASES if c['pii']); n_cl=len(CASES)-n_pii
    print(f"\n===== {label} =====")
    print(f"개인정보 포함 문장 {n_pii}건 중 마스킹 성공 {hit}건 / 실패 {miss}건  → 탐지율 {hit/n_pii*100:.1f}%")
    print(f"정상 문장 {n_cl}건 중 오탐 {fp}건 → 오탐률 {fp/n_cl*100:.1f}%")
    if missed:
        print(f"-- 못 막은 유형 {len(missed)}건 --")
        for k,t in missed: print(f"   · [{k}] {t}")
    if fps:
        print(f"-- 오탐 {len(fps)}건 --")
        for k,t,o in fps: print(f"   · [{k}] {t}  →  {o}")
    return hit, miss, fp, n_pii, n_cl
c=run_v9('개선안 v9 (v8 + 카드번호 룬 검증 · 문맥어 제외)')
print(f"\n>>> 탐지율 {a[0]/a[3]*100:.1f}% → {b[0]/b[3]*100:.1f}% → {c[0]/c[3]*100:.1f}%")
print(f">>> 오탐률 {a[2]/a[4]*100:.1f}% → {b[2]/b[4]*100:.1f}% → {c[2]/c[4]*100:.1f}%")
```

시험 문장의 전화번호·주민등록번호·사업자등록번호·계좌번호·여권번호·차량번호·이메일·이름·주소는 모두 형식만 맞춰 지어낸 값입니다. 주민등록번호는 공식 검증식을 통과하지 않는 값이고, 이메일은 배달되지 않는 예약 도메인(example.com)이며, 카드번호는 결제사 공개 시험용 번호입니다.