15 Years of Broken Encrypted Emails

…and we’re still doing it wrong

Alfredo Pironti – IOActive
alfredo.pironti@ioactive.com

IMDEA - NEXTLEAP
3 Mar 2017

Agenda

- Intro to OpenPGP
- An efficient attack on signatures
  - And other well known attacks
- Application to encrypted emails
- Proposing a fix
- Future work and conclusion

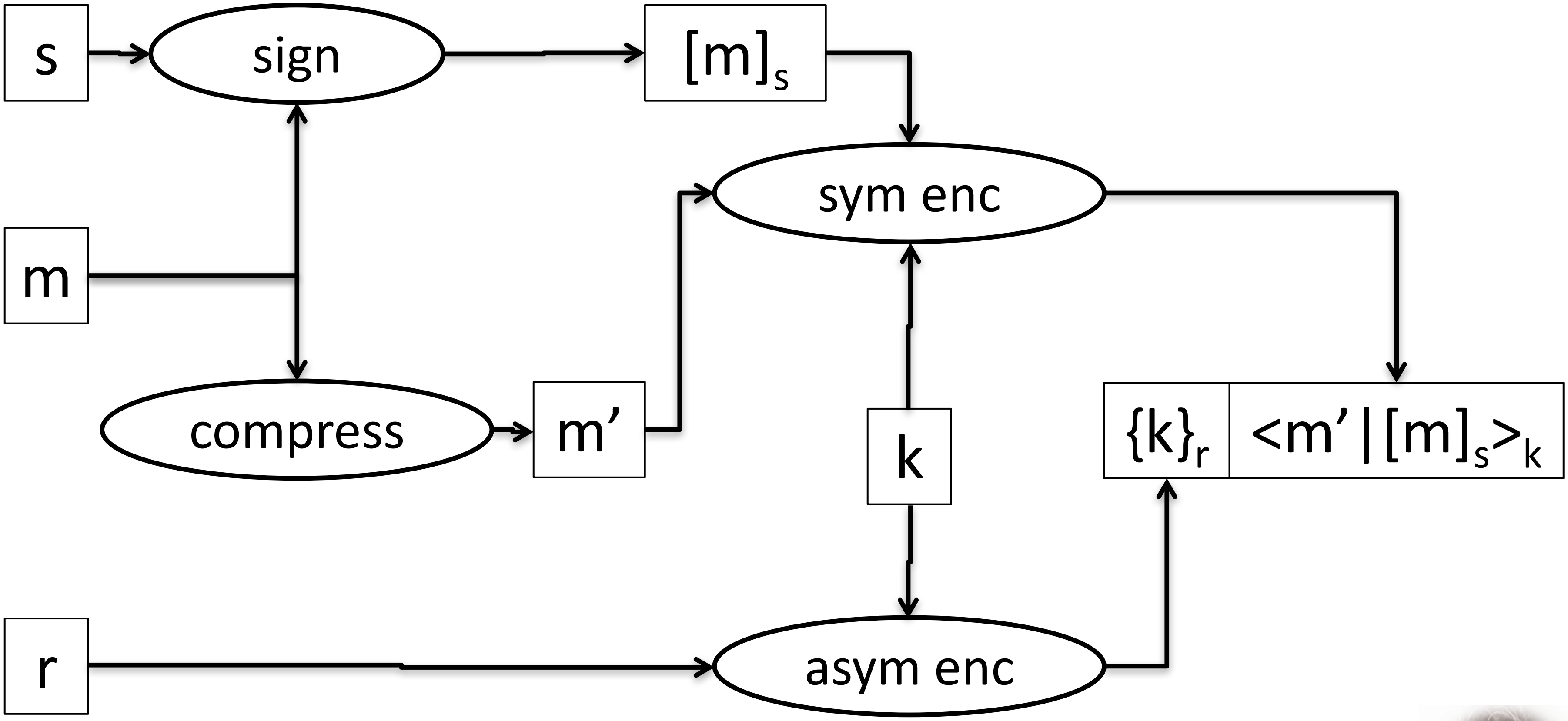
Intro to OpenPGP

IOActive
Hardware | Software | Wetware
SECURITY SERVICES

The OpenPGP Standard

- RFC 4880 (2007)
  - How to perform encryption
  - Encrypt; Sign; Sign & Encrypt
- RFC 3156 (2001)
  - How to use OpenPGP to encrypt email
- Widely used
  - Email, password managers, git…
- Design is about 20 years old

OpenPGP Sign & Encrypt

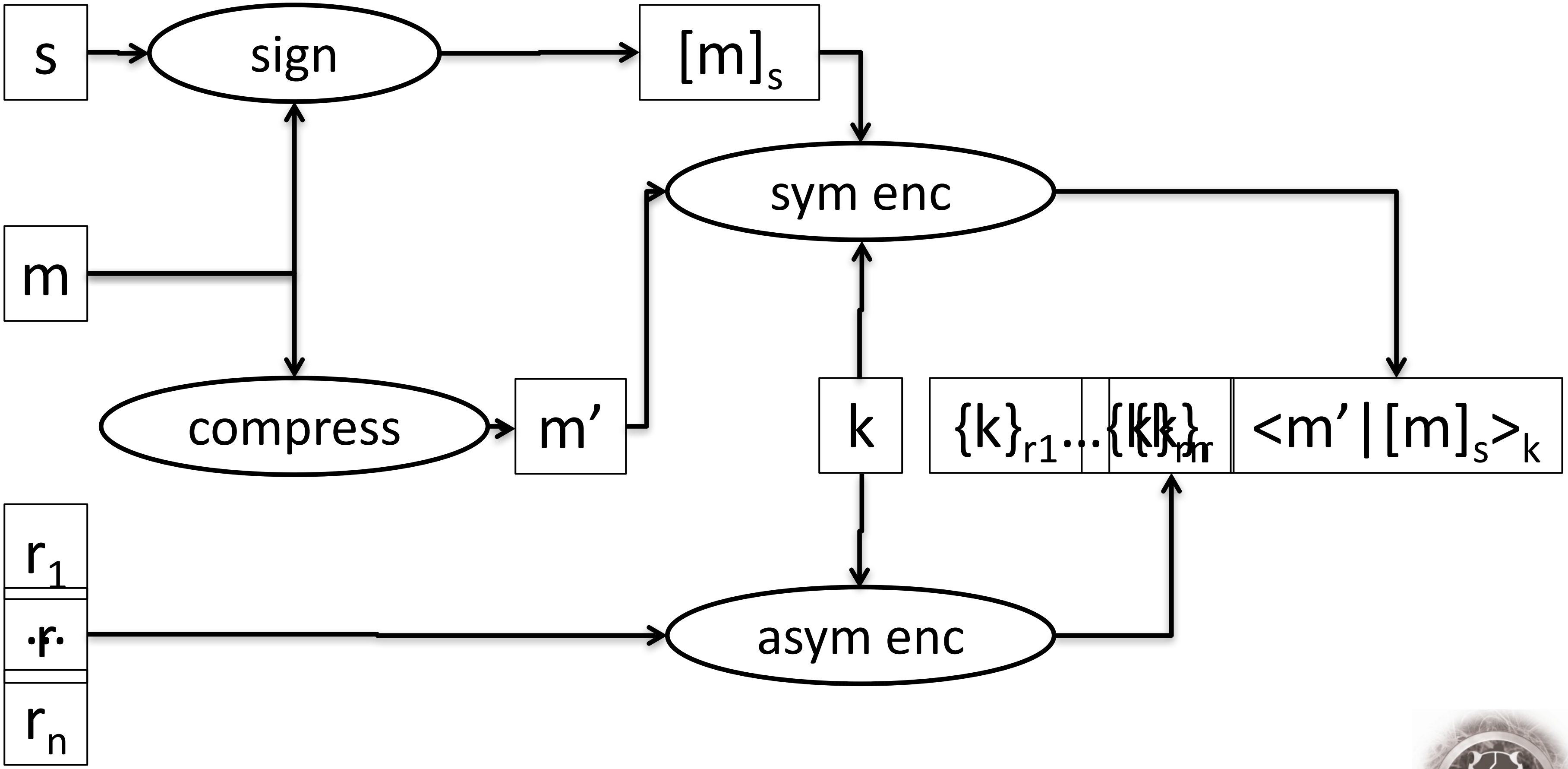


OpenPGP Sign & Encrypt

Properties:

- Probabilistic encryption
- Efficient for large messages
- Efficient for multiple recipients

Multiple Recipients



An Efficient Attack on Signatures and Other Well-Known Attacks

Surreptitious Forwarding [1]

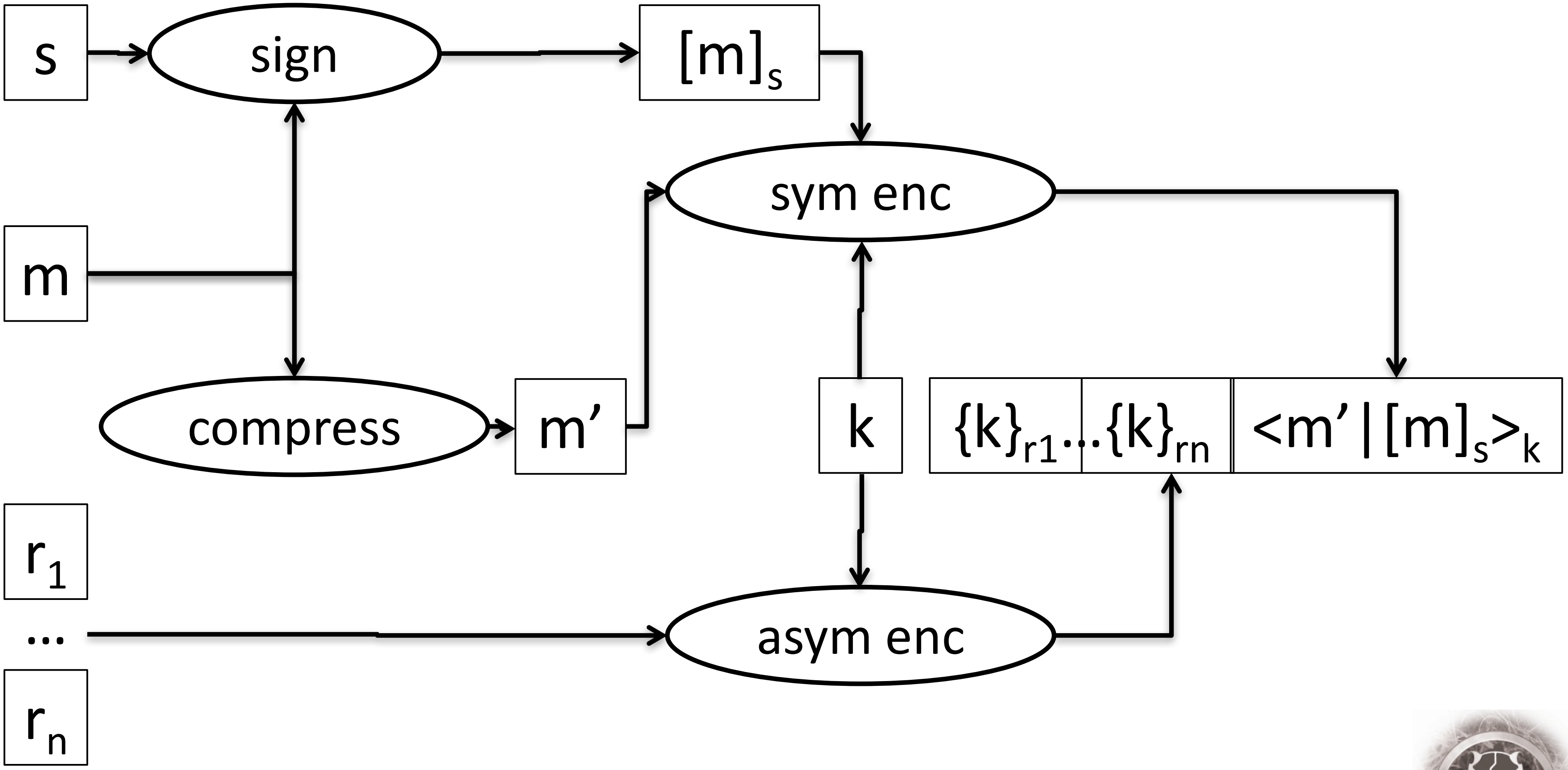
- $A \rightarrow B: \{ [\text{“I love you”}]_a \}_b$
- $B \rightarrow C: \{ [\text{“I love you”}]_a \}_c$

- $A \rightarrow B: \{ [\text{“sales plan”}]_a \}_b$
- $B \rightarrow C: \{ [\text{“sales plan”}]_a \}_c$

- $A \rightarrow B: \{ [\text{“I owe you 10K”}]_a \}_b$
- $B \rightarrow C: \{ [\text{“I owe you 10K”}]_a \}_c$

[1] Davis, D.: Defective sign & encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP and XML. In USENIX 2001

Efficient Surreptitious Forwarding



PoC tool available on demand

IOActive
Hardware | Software | Wetware
SECURITY SERVICES

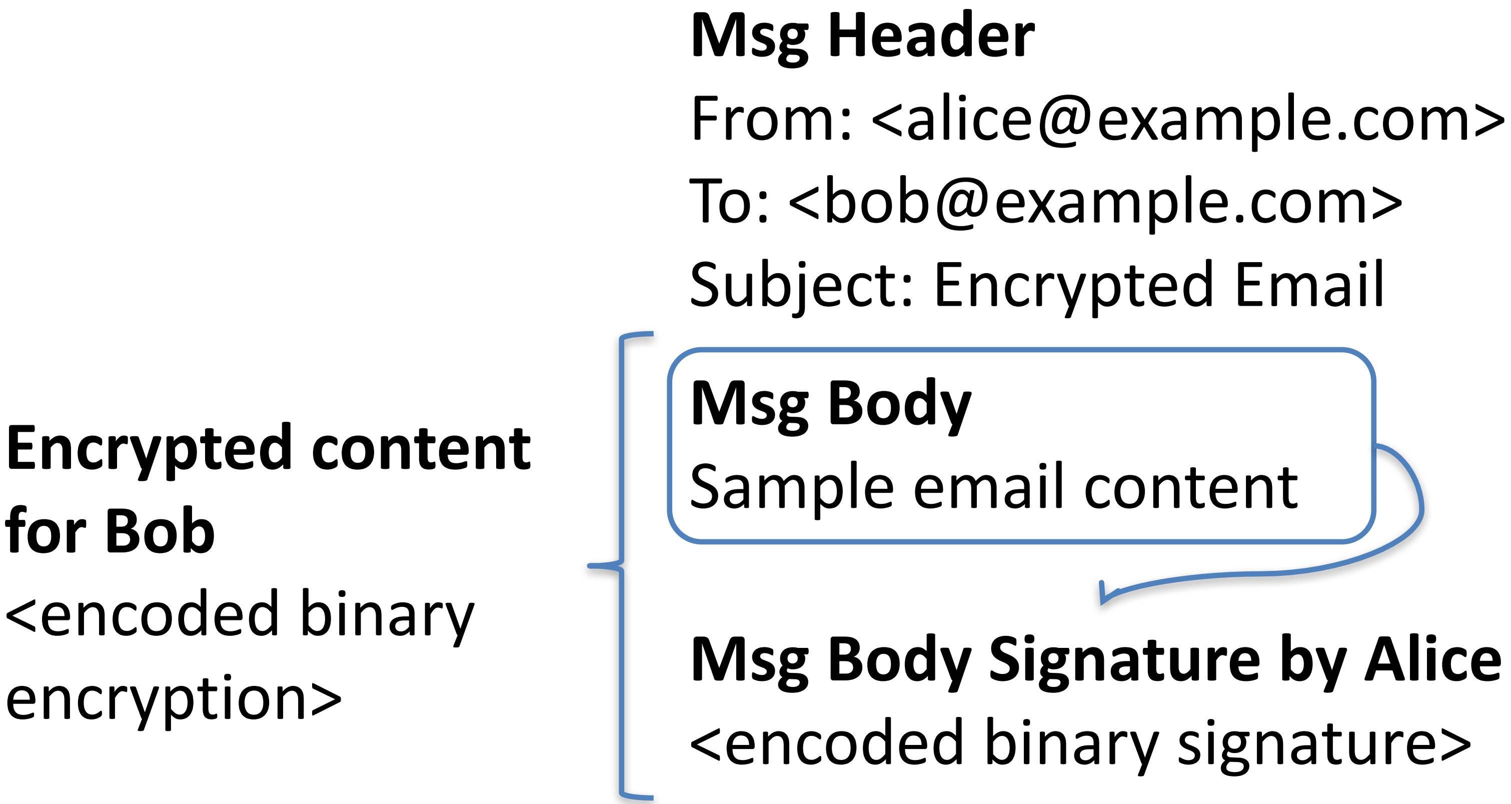
Message Compression

- Seriously?
  - “OpenPGP implementations *should* compress the message after applying signature but before encryption” – RFC 4880

- Remember CRIME attack on TLS?
  - Compression leaks information about entropy of plaintex

Application to Encrypted Emails

RFC 3156 – Email Sign & Encrypt



Msg Header
From: <alice@example.com>
To: <bob@example.com>
Subject: Encrypted Email

Encrypted content for Bob
<encoded binary encryption>

Msg Body
Sample email content

Msg Body Signature by Alice
<encoded binary signature>

Alternatively, use the OpenPGP Sign & Encrypt scheme

Tampering with Email Headers

- From:
  - Confidentiality traded for routing purposes
  - Could use pseudonyms
  - Should be signed
- To:
  - Confidentiality traded for routing purposes
  - Could use pseudonyms
  - No signature makes encryption pointless!
- Subject:
  - Not encrypted: strong contrast with user expectation
  - Hard to encrypt in a backward-compatible way
- Reply-To:
  - *Please, re-encrypt the whole thread with the attacker's key!*

Tampering with Reply-To: in Practice

- Sent several encrypted test reports to “secure@” of software vendors
- Added an attacker-controlled Reply-To: address
  - Avoiding the social engineering aspect: Reply-To: address totally different from sender’s
- Attacker got more than 50% responses
  - One informed him that the message was signed, but not encrypted
  - One replied to both, asking which address should be used
  - Some answers were not signed
- Caveats
  - Small sample: < 10 recipients
  - Test data did not look critical; no rise in attention

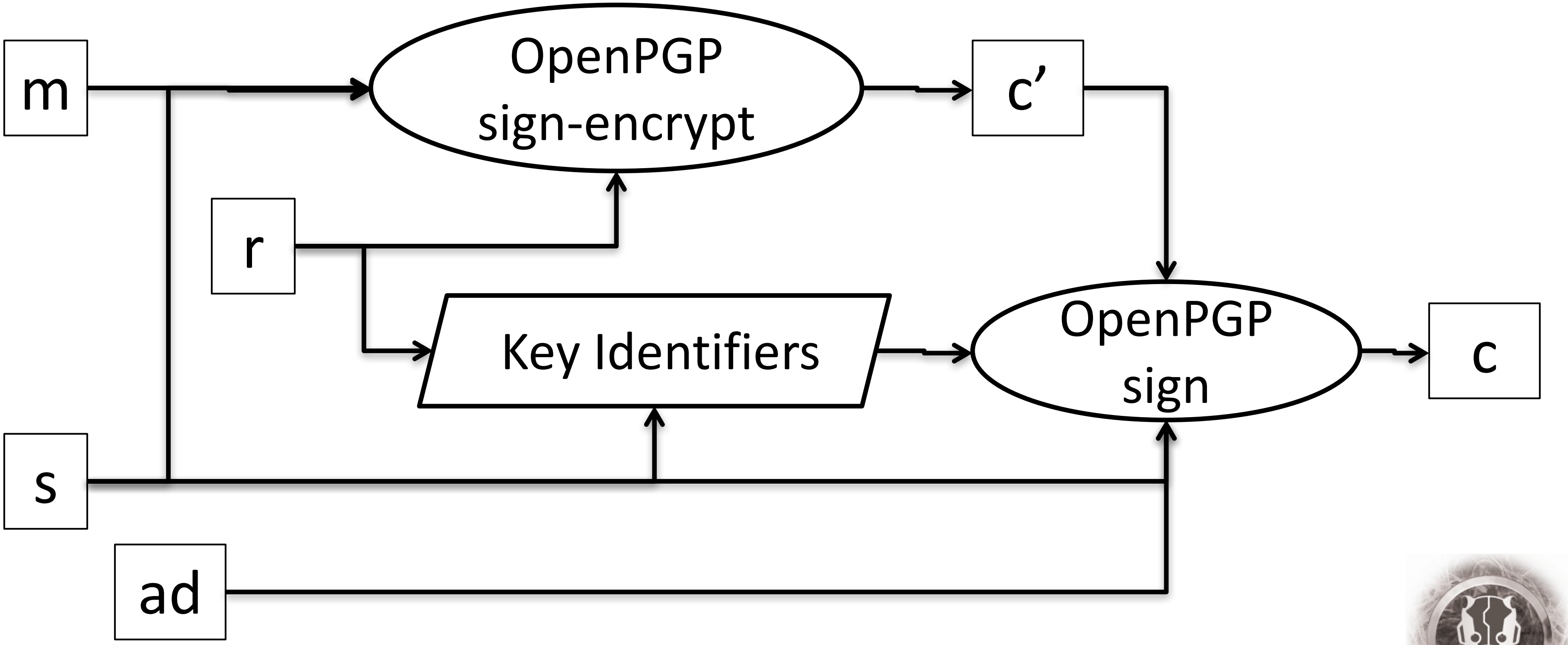
Proposing a Fix

AEAD for OpenPGP

- Authenticated Encryption with Additional Data
  - Additional data are signed, but not encrypted
  - Examples in the symmetric world: AES-GCM
  - Email headers are AD

An OpenPGP-compatible Scheme

- `Enc(s,r,m,ad)`
  - Sign-encrypt-sign

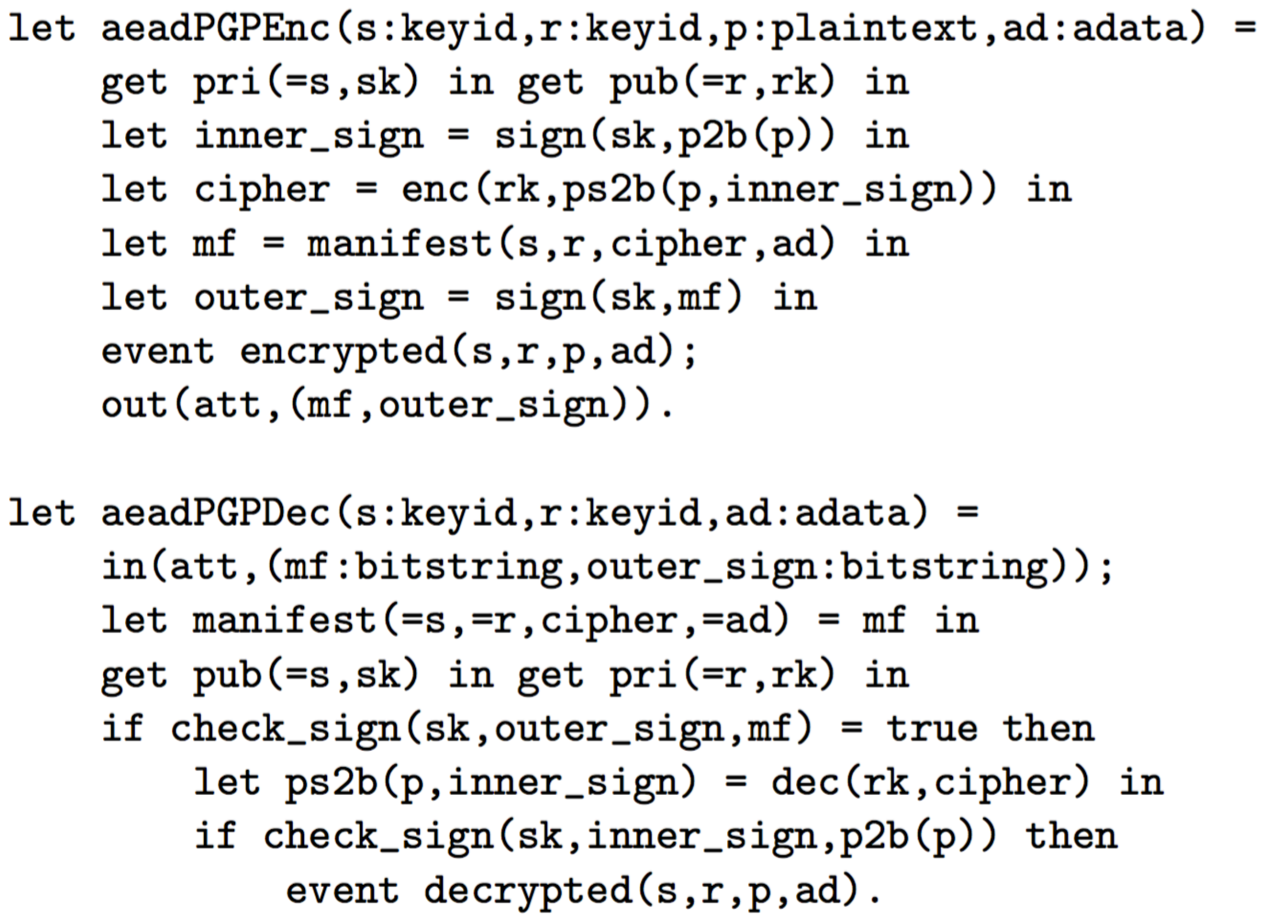


Details and Properties

- On decryption, inner and outer signature keys must match
- Generalization of Sign-Encrypt-Sign scheme proposed by Davis [1]
  - Accounts for AD
  - Fits into the OpenPGP standard
- Compression is disabled
- Preserves probabilistic encryption
- Provides CTXT-INT

Formal Verification

- ProVerif, symbolic model



```
let aeadPGPEnc(s:keyid,r:keyid,p:plaintext,ad:adata) =
    get pri(=s,sk) in get pub(=r,rk) in
    let inner_sign = sign(sk,p2b(p)) in
    let cipher = enc(rk,ps2b(p,inner_sign)) in
    let mf = manifest(s,r,cipher,ad) in
    let outer_sign = sign(sk,mf) in
    event encrypted(s,r,p,ad);
    out(att,(mf,outer_sign)).

let aeadPGPDec(s:keyid,r:keyid,ad:adata) =
    in(att,(mf:bitstring,outer_sign:bitstring));
    let manifest(=s,=r,cipher,=ad) = mf in
    get pub(=s,sk) in get pri(=r,rk) in
    if check_sign(sk,outer_sign,mf) = true then
        let ps2b(p,inner_sign) = dec(rk,cipher) in
        if check_sign(sk,inner_sign,p2b(p)) then
            event decrypted(s,r,p,ad).
```

Application to Emails

- Headers are AD
  - Must agree on signed headers order, or use extra header
  - Watch out for outer signature stripping (don’t allow legacy email encryption)

Future Work and Conclusion

End-to-End Email Encryption

- Extension for in-browser email encryption
- From the docs:
  - Implements RFC 4880
  - Headers unencrypted (nor signed?)
  - RFC 3156 not *currently* supported

  - Uses elliptic curves
  - Centralized key distribution with transparency
  - Not yet ready for general use

Conclusion

- Mismatch between user expectations and cryptographic properties
- Relying on dated standards with known design flaws
- Practical attacks are possible
- AEAD with backward compatibility is possible
- New momentum in secure email

Thank you!

Questions?