Encrypted CCN with Public Keys
Draft 1

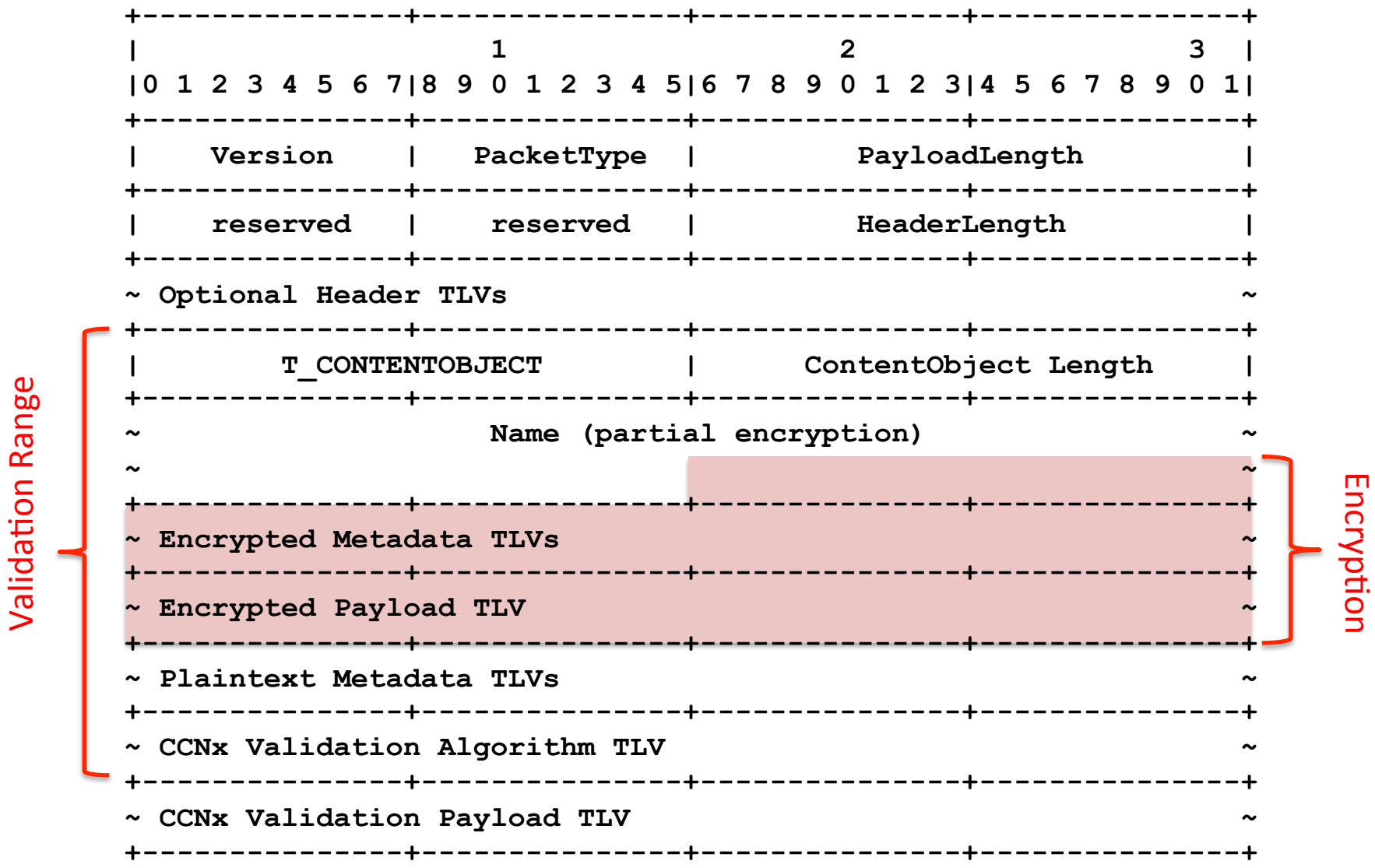
Marc Mosko

June 18, 2015

Problem Area

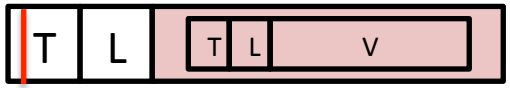
- Encrypt a message under another party's public key.
- Uses same basic format as "Encrypted CCN for Sessions"

CONTENT OBJECT



We want selective encryption

Implications of TLV encryption



3) Encrypt V, leave T+L plaintext, use "encrypted" bit to mark an encrypted V. Preferred Option.

- We can begin computing a block cipher from byte 0 of the CCNx message.
- Skip the XOR of all T and L fields.
- Only XOR Vs of Ts marked for encryption.
- Does not require re-ordering fields in CCNx message to have a "plaintext" and "ciphertext" sections.

AES-128 Block Cipher (similar to CWC-AES-128)

- Used between systems with public key cryptography.
- The secret key is encrypted with a public key.
- Typically, this would only be done in the first chunk and subsequent chunks would use CCW-AES-128.

Proposed Solution

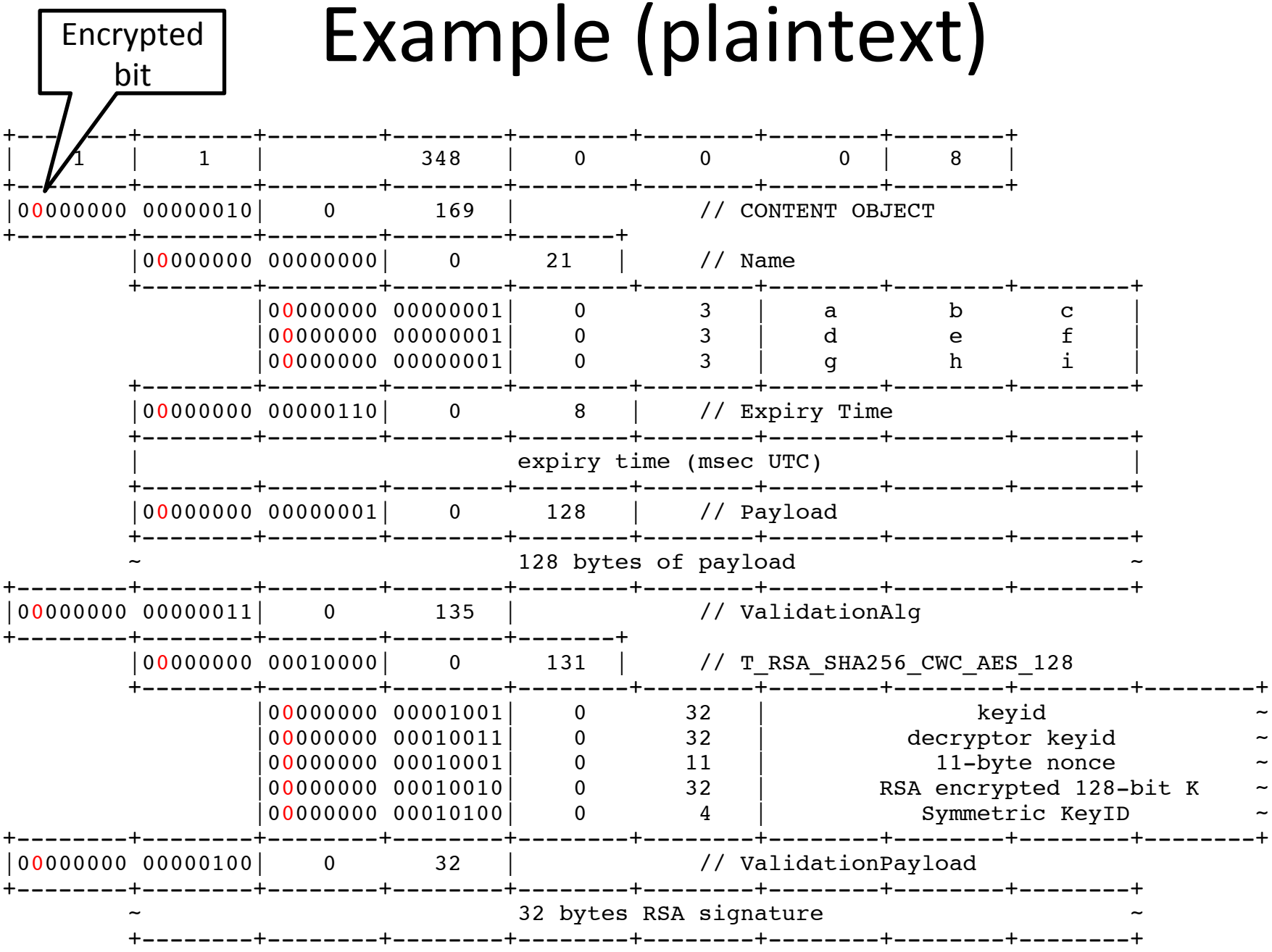
- Use same cypher as CWC-AES-128
  - Generate a random key K and nonce N
  - Encrypt CCNx message as per Session proposal
  - ValidationAlg is T_RSA_SHA256_CWC_AES_128
    - Has Nonce is plaintext
    - Has K encrypted under recipients public key (new field)
    - Has KeyId of publisher (as normal)
    - Has DecryptionKeyId of receiver (new field)
    - Has 4-byte SymmetricKeyId to identify key later (new field)
  - ValidationPayload
    - Is normal signed SHA256 of publisher

[1] http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/cwc/cwc-spec.pdf

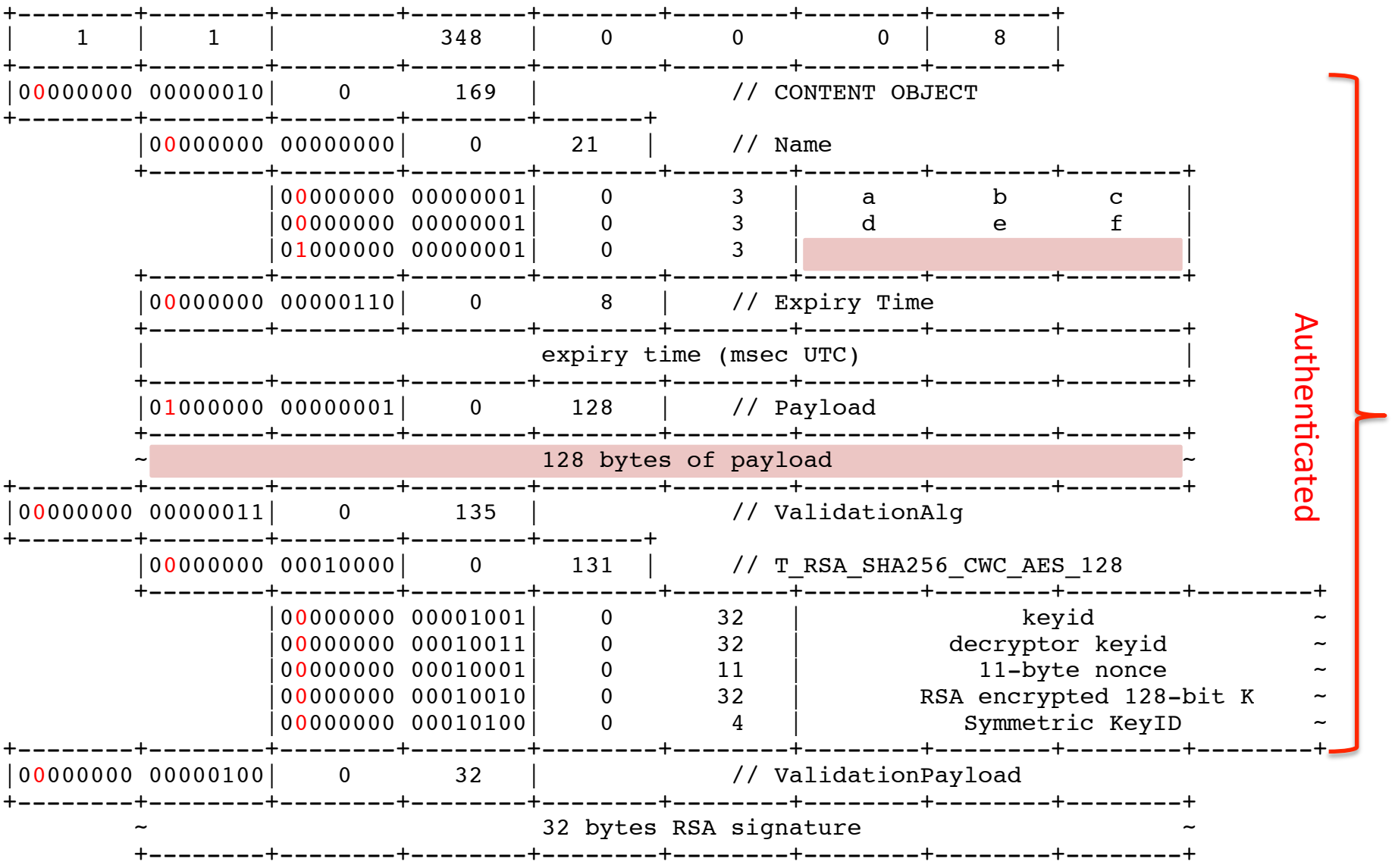
Use in Chunked content

- Intersperse RSA chunks
  - First chunk setups up key K_0
  - May also carry payload encrypted under K_0
  - Every so often (e.g. every M megabytes) can rotate key by using an RSA chunk with K_i that applies to subsequent chunks
  - RSA chunks identify SymmetricKeyId (4 bytes) which is used in CWC-AES-128 packets as KeyId.

Example (plaintext)



Example (ciphertext)



Conclusion

- Uses same encryption as session format for CCNx message
- Uses RSA in Validation section
  - Encryption to pass random key
  - Signing for authentication
  - Passes DecryptionKeyID to identify remote party
- Allows off-line operation without live key exchange
  - For example, could be in first chunk of content