

# Encrypted CCN with Public Keys

## Draft 1

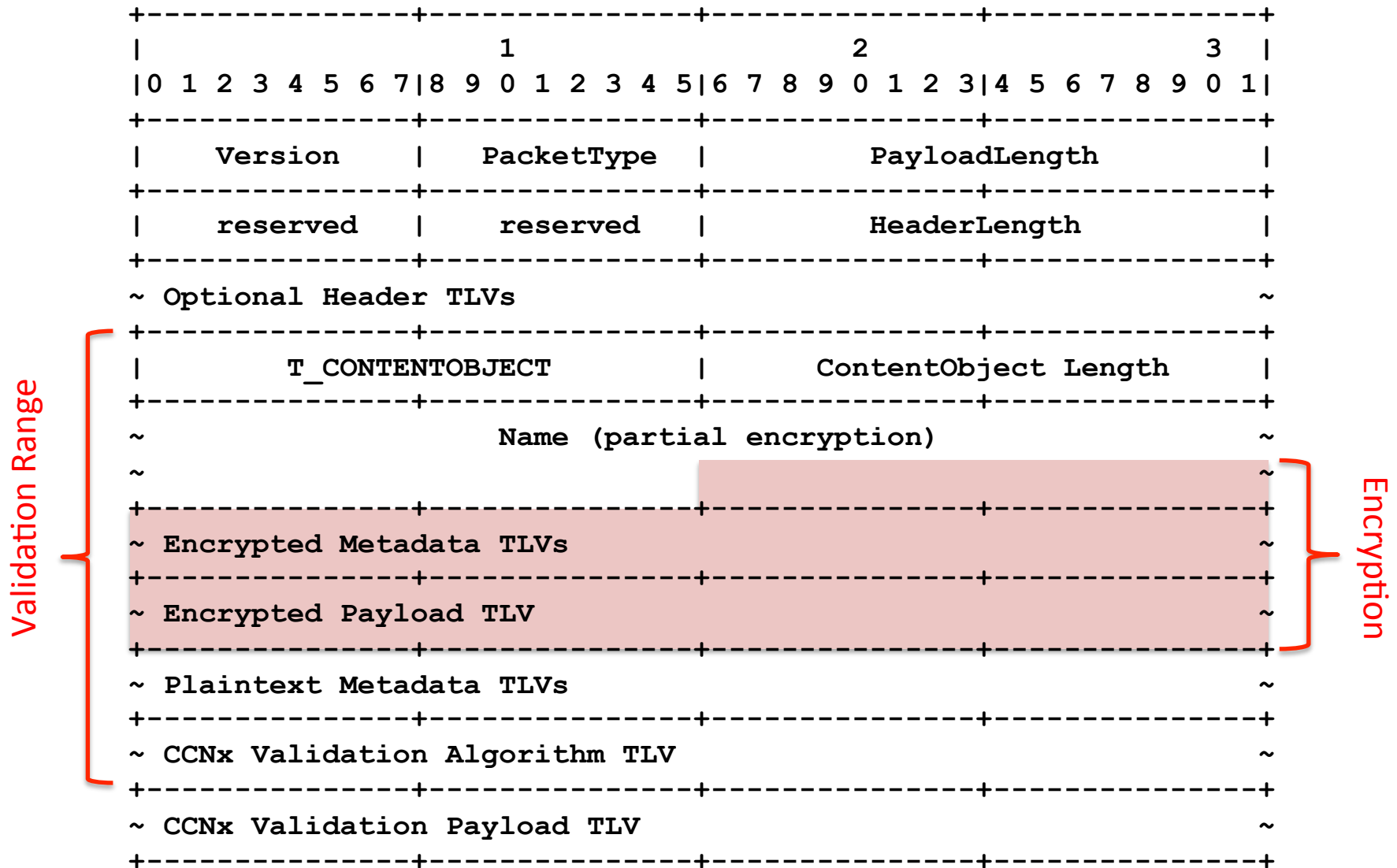
Marc Mosko

June 18, 2015

# Problem Area

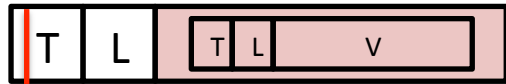
- Encrypt a message under another party's public key.
- Uses same basic format as “Encrypted CCN for Sessions”

# 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

# 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 SymmetricKeyld (4 bytes) which is used in CWC-AES-128 packets as Keyld.

Encrypted  
bit

1	1	348	0	0	0	8
// CONTENT OBJECT						
00000000	00000010	0	169			
// Name						
00000000	00000000	0	21			
00000000	00000001	0	3	a	b	c
00000000	00000001	0	3	d	e	f
00000000	00000001	0	3	g	h	i
// Expiry Time						
00000000	00000110	0	8			
expiry time (msec UTC)						
// Payload						
00000000	00000001	0	128			
128 bytes of payload						
// ValidationAlg						
00000000	00000011	0	135			
// T_RSA_SHA256_CWC_AES_128						
00000000	00010000	0	131			
00000000	00001001	0	32	keyid		
00000000	00010011	0	32	decryptor keyid		
00000000	00010001	0	11	11-byte nonce		
00000000	00010010	0	32	RSA encrypted 128-bit K		
00000000	00010100	0	4	Symmetric KeyID		
// ValidationPayload						
00000000	00000100	0	32			
32 bytes RSA signature						



# Example (ciphertext)

1	1	348	0	0	0	8
+-----+-----+-----+-----+-----+-----+-----+						
00000000	00000010	0	169	// CONTENT OBJECT		
+-----+-----+-----+-----+-----+-----+-----+						
00000000	00000000	0	21	// Name		
+-----+-----+-----+-----+-----+-----+-----+						
00000000	00000001	0	3	a	b	c
00000000	00000001	0	3	d	e	f
01000000	00000001	0	3			
+-----+-----+-----+-----+-----+-----+-----+						
00000000	00000110	0	8	// Expiry Time		
+-----+-----+-----+-----+-----+-----+-----+						
expiry time (msec UTC)						
+-----+-----+-----+-----+-----+-----+-----+						
01000000	00000001	0	128	// Payload		
+-----+-----+-----+-----+-----+-----+-----+						
~ 128 bytes of payload ~						
+-----+-----+-----+-----+-----+-----+-----+						
00000000	00000011	0	135	// ValidationAlg		
+-----+-----+-----+-----+-----+-----+-----+						
00000000	00010000	0	131	// T_RSA_SHA256_CWC_AES_128		
+-----+-----+-----+-----+-----+-----+-----+						
00000000	00001001	0	32	keyid		
00000000	00010011	0	32	decryptor keyid		
00000000	00010001	0	11	11-byte nonce		
00000000	00010010	0	32	RSA encrypted 128-bit K		
00000000	00010100	0	4	Symmetric KeyID		
+-----+-----+-----+-----+-----+-----+-----+						
00000000	00000100	0	32	// ValidationPayload		
+-----+-----+-----+-----+-----+-----+-----+						
~ 32 bytes RSA signature ~						
+-----+-----+-----+-----+-----+-----+-----+						

Authenticated

Authenticated

# 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