Cryptography In The Real World

Ryan Darnell (He/Him)
Senior Software Security Engineer
Rockwell Automation

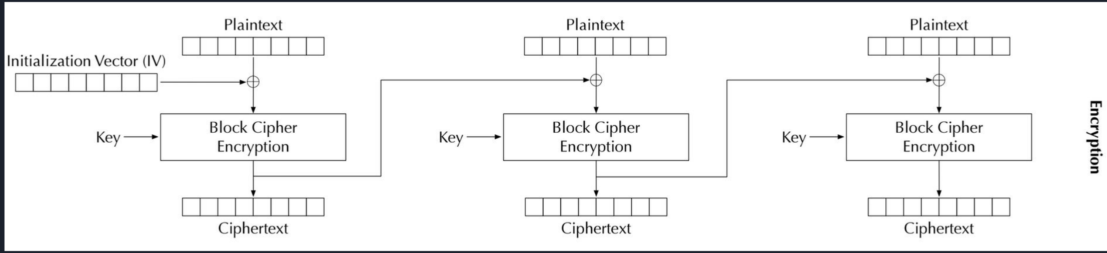
Brief Intro

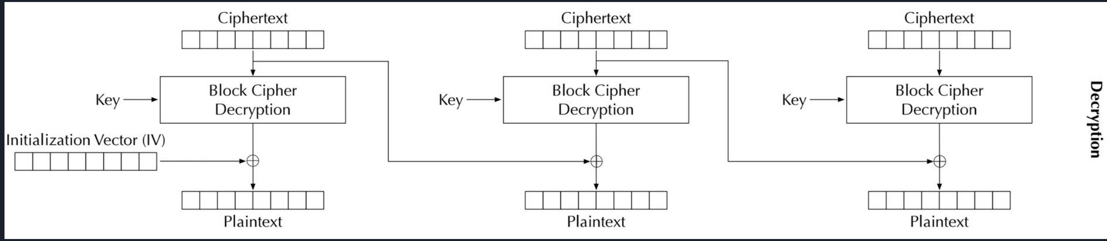
- Studied Architecture and Civil Engineering before switching to Computer Science
- Spring 2017 - Graduated from MSU
- Started working for Rockwell Automation in Ohio
  - Immediately was placed on the Software Security Team (Team Name: MI7)
- The MI7 Security Team
  - Design and Implement Security features within the products our team works on
  - Work with penetration testers to resolve any issues they find within the features we implement prior to release
  - Find and patch any existing security vulnerabilities discovered in the field
    - Be it our products directly or through software libraries we use
- Security Projects
  - Encryption/Decryption Mechanism
    - Designed the encryption mechanism to minimize data loss and support extensibility and maintainability over time
    - Designed the decryption mechanism to be self-recovering and simple for authenticated users to operate.
  - Authentication and Authorization
  - Certificate Management and Expiration Detection within a Cloud-Based Technology stack
- Disclaimer

Quick Review + Tips

Cipher Block Chaining (CBC)

- Confidentiality
- Block dependency
- Ensures that with unique IV’s, similar plaintext strings have different ciphertext
- Fixed Block Lengths
  - Padding bytes required
- Data Loss Potential:
  - 1 Byte corrupted = 1 Block + 1 Byte lost
  - 1 Byte lost = All data lost



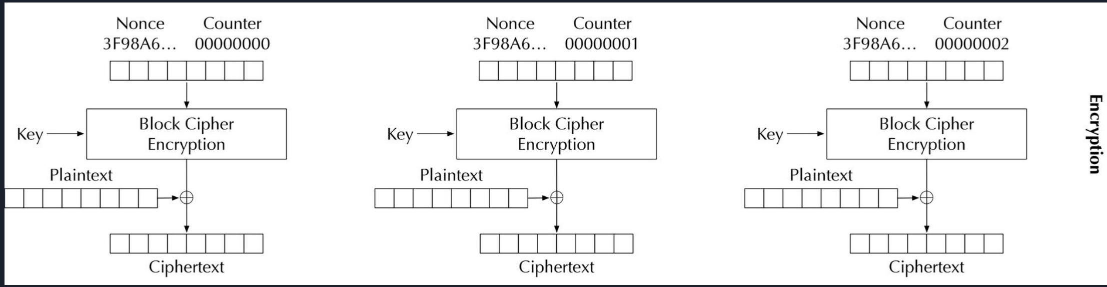


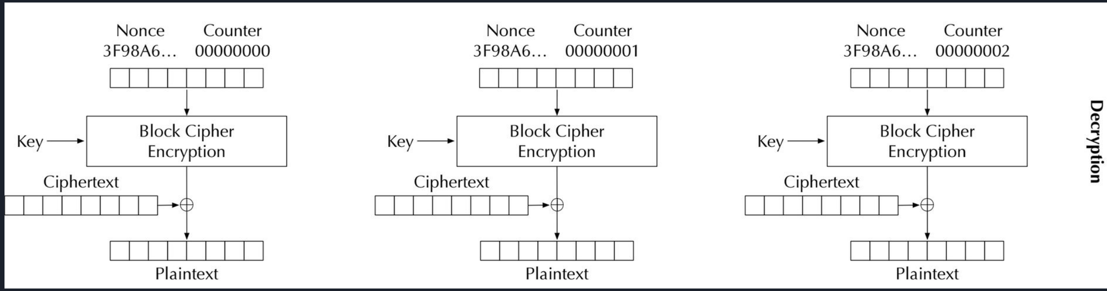
CyberChef Recipe

Counter (CTR)

- Confidentiality
- Ensures that with a unique nonce, they keystream xor'd with the plaintext will always be unique
- No padding required
  - Can be used as a stream cipher
- Data Loss Potential:
  - 1 Byte corrupted = 1 byte lost
  - 1 Byte Lost = N bytes lost
    - N = L - x | L: length of data; x: position of lost byte
    - IOW - all data following the lost byte is lost

CyberChef Recipe

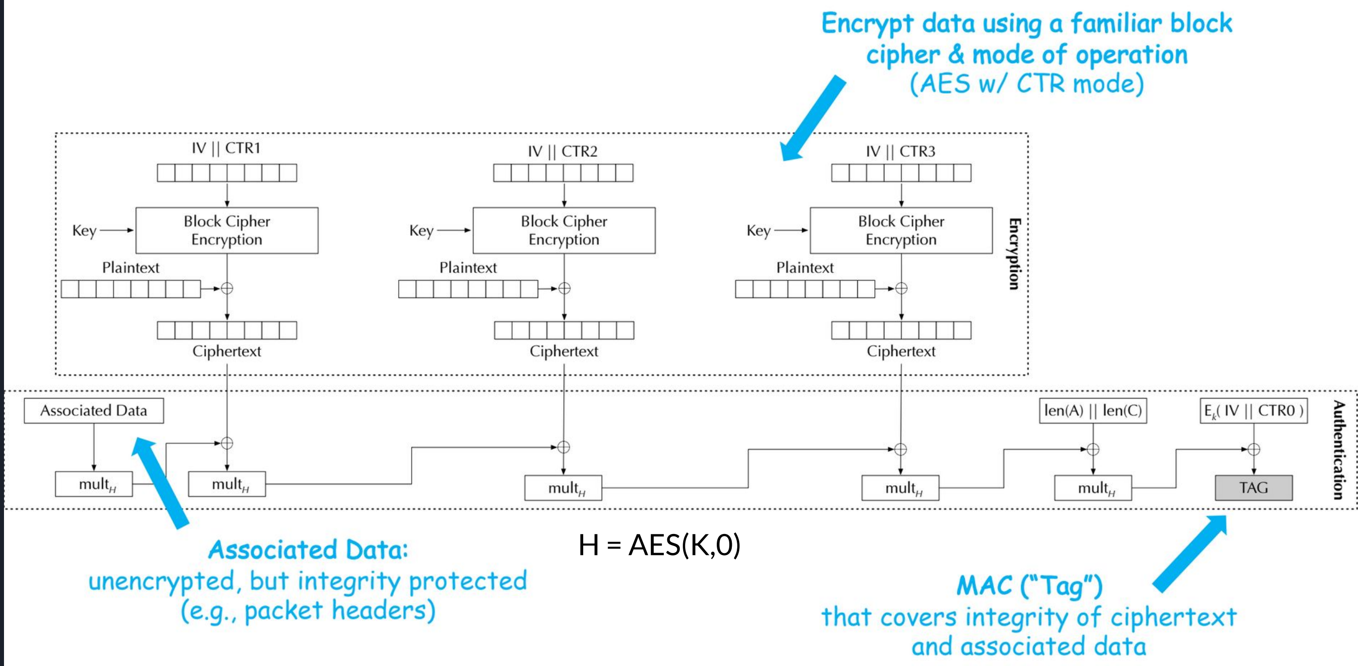




Galois/Counter Mode (GCM)

- Confidentiality + Authenticity (Integrity)
- Encrypt-Then-MAC
  - AES-CTR (start with Counter 1)
  - Message Authentication Codes
- No padding required
  - Can be used as a stream cipher
- Data Loss Potential:
  - 1 Byte modified = Ciphertext rejected
  - 1 Byte Lost = Ciphertext rejected

CyberChef Recipe



Encrypt data using a familiar block cipher & mode of operation (AES w/ CTR mode)

H = AES(K,0)

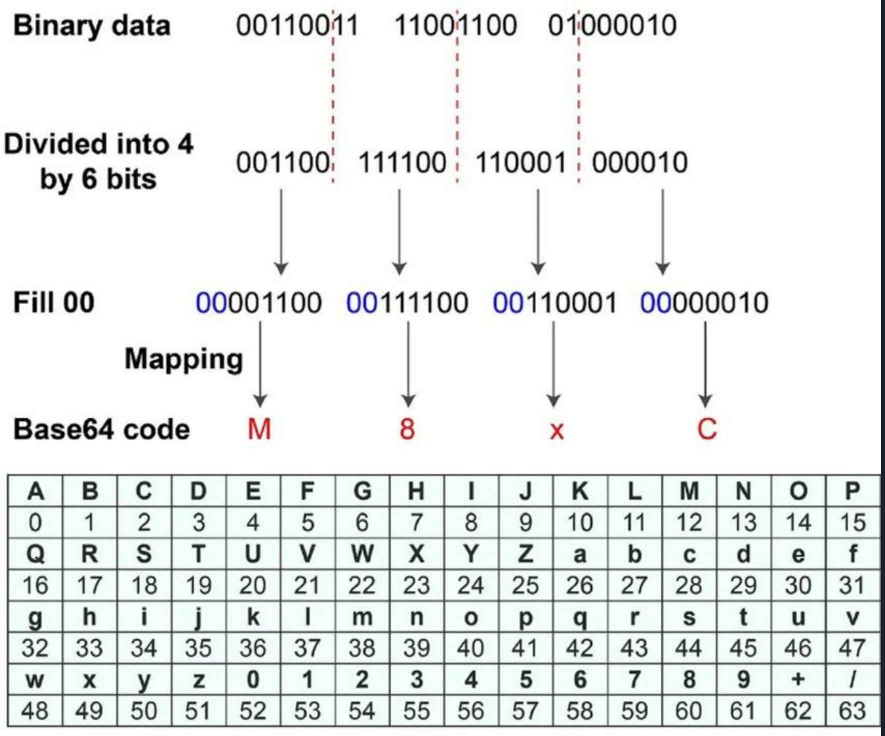
Associated Data: unencrypted, but integrity protected (e.g., packet headers)

MAC ("Tag") that covers integrity of ciphertext and associated data

Encoding Binary Data

- Base64 Encoding
  - Divides binary data into 6-bit groups
  - Alphabet:
    - A-Za-z0-9+/=

[CyberChef Recipe](#)



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
w	x	y	z	0	1	2	3	4	5	6	7	8	9	+	/
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Using Asymmetric and Symmetric Encryption Together

- Use Asymmetric encryption for securing symmetric keys and Initialization Vectors/Nonces
- Asymmetric keys will rotate (swapped out for a new set) over time
  - These keys might be updated monthly, yearly, between software versions, etc.
  - Always assume the asymmetric keys will change at some point and that the encryption/decryption mechanisms will need to support that
- Use Symmetric encryption to encrypt larger sets of data
- Always use a new symmetric key and IV/Nonce whenever possible.
  - Ensure that there's never a situation where the same key and IV/nonce are used more than once

Components of a Good Design

- Security (obviously)
  - Threat Model
- Extensibility
  - Can the design be easily modified and upgraded in the future?
  - Does the design support more than one version?
  - Backwards Compatibility
- Scalability
  - Can this tool be adopted elsewhere?
  - Can behavior be added on dynamically without requiring significant effort?
- Performance
  - Memory
  - Speed
  - Disk Usage (If applicable)
- Recoverability
  - E.g. How might the decryption mechanism or software recover if the ciphertext is corrupted?
- Cost to Implement
  - How long will this effort take?
  - Does it require rebuilding/refactoring currently implemented features? Why?

Design Strategy Tips

- Ask Questions!
- Ask MORE Questions!
  - Figure out as much of the Unknowns as you can
- Figure out how the tool will work around the feature you are meant to design
  - How might the design leverage what's already there?
  - What might need to change in order for your design to work?
- Narrow your focus
  - Focus on the things that can be easily solved
  - Slowly expand on the components within the problem that are more complex
  - Figure out how to piece together what's been solved
- Bounce ideas off your teammates
  - Chances are, you won't be able to think of everything and your teammates can see something you missed or ask you about a scenario/edge case you didn't think of
- Refine, Recycle and Reiterate

Design Problem Activity