Network Security

Secure Channel Add-On

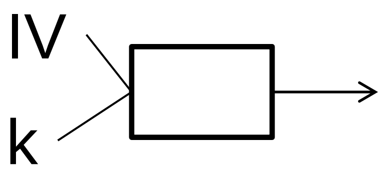
Overview

- Part I: The Secure Channel
- **Part II: Attacks against Secure Channel**
- Part III: Authenticated Encryption

Attacks against Secure Channel with Stream Ciphers

- Part I: The Secure Channel
- Part II: Attacks against Secure Channel
  - **Attacks against Secure Channel with Stream Cipher**
- Part III: Authenticated Encryption

- Re-use of Initialization Vector (IV)



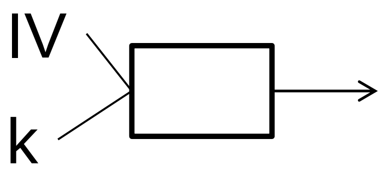
1 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1

xor

P1 = 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 1 0

C1 = 1 0 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1

Then some time later the same IV is used again:



1 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1

xor

P2 = 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1

C2 = 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 1 1 0

Re-use of Initialization Vector

- Re-use of Initialization Vector (IV) continued

C1 = 1 0 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1

C2 = 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 1 1 0

C1+C2 = 1 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1

= = = = = = = = = = = = … → P1+P2=C1+C2

P1+P2 = 1 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1

P1 = 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 1 0

P2 = 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1

- As we see from the example, the attacker can computer C1+C2 because he observes C1 and C2, but that means he knows also P1+P2.
- Known Plaintext (e.g. P1) → attacker can compute other plaintext
- Statistical properties of plaintext can be used if plaintext is not random-looking. That means if entropy of P1+P2 is low.

Attacks against Secure Channel with Stream Ciphers

- Part I: The Secure Channel
- Part II: Attacks against Secure Channel
  - **Padding Oracle Attack against bad combination of CBC mode and MAC**
- Part III: Authenticated Encryption

Guessing a secret (revisited)

- Passwords
  - N: size of alphabet (number of different characters)
  - L: length of password in characters
- Complexity of guessing a randomly-generated password / secret
  - The assumption is, we generate a password and then we test it.

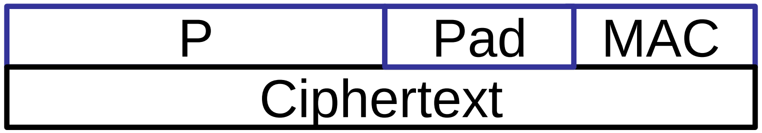
  → $O(N^L)$
- Complexity of guessing a randomly-generated password character by character
  - The assumption is that we can check each character individually for correctness.
  - For each character it is N/2 (avg) and N (worst case)
  - So, overall L*N/2 (avg)
- In the subsequent slides we will show an attack that reduces the decryption of a blockcipher in CBC mode to byte-wise decryption (under special assumptions).

MAC-then-Encrypt Issues

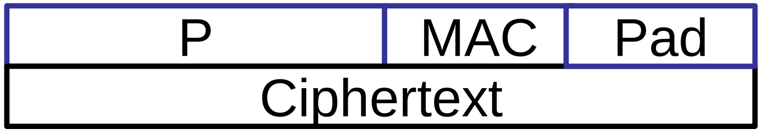
- Operation
  - P and MAC are encrypted and hidden in the ciphertext.
  - Receiver
    - Decrypts P
    - Decrypts MAC
    - Computes and checks MAC →MAC error or success
- Consequence
  - MAC does not protect the ciphertext.
  - Integrity check can only be done once everything is decrypted.
  - As a consequence, receiver will detect malicious messages at the end of the secure channel processing and not earlier.
  - But is that more than a performance issue? Well, yes.

MAC-then-Encode-then-Encrypt

- If we use a block cipher, we have to ensure that the message encoding fits to the blocksize of the cipher.



- Encode-then-MAC-then-Encrypt:
  - Format P so that with the MAC added the encryption sees the right size.
  - Needs that we know the size of the MAC and blocksize of cipher when generating P | Padding.



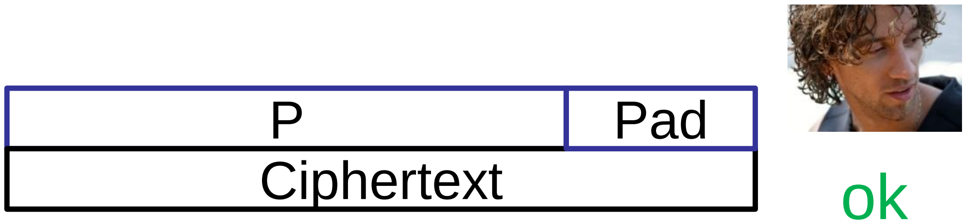
- MAC-then-Encode-then-Encrypt
  - Used in TLS/SSL
  - Here, we add the MAC first and then pad the P | MAC to the correct size.
  - How do we know what is padding and what not? Padding in TLS/SSL:
    - If size of padding is 1 byte, the padding is 1.
    - If size of padding is 2 bytes, the padding is 2 2.
    - If size of padding is 3 bytes, the padding is 3 3 3.
    - ….

Oracles and Side Channels

- In ancient times, people asked oracles for guidance.
- In computer science, oracles are functions that give as cheaply access to information that would otherwise hard to compute.
  - E.g. O(1) cost to ask specific NP-complete question → polynomial hierarchy
- In cryptography, an attacker can trigger some participant O in a protocol or communication to leak information that might or might not be useful.
  - Participant O may re-encrypt some message fragment
  - Participant O responds with an error message explaining what went wrong
  - Response time of participant O may indicate where error happened
  - Response time may leak information about key if processing time depends (enough) on which bits are set to 1.
    - More obvious for the computationally expensive public key algorithms, but implementations of symmetric ciphers have also been attacked.

Side Channels and Padding Oracles

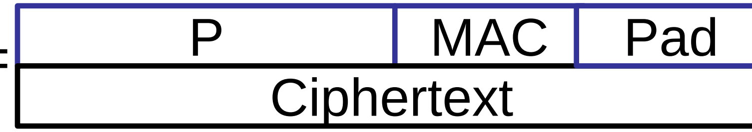
- Side Channel Attacks
  - A general class of attacks where the attacker gains information from aspects of the physical implementation of a cryptosystem.
  - Can be based on: Timing, Power Consumption, Radiation, …



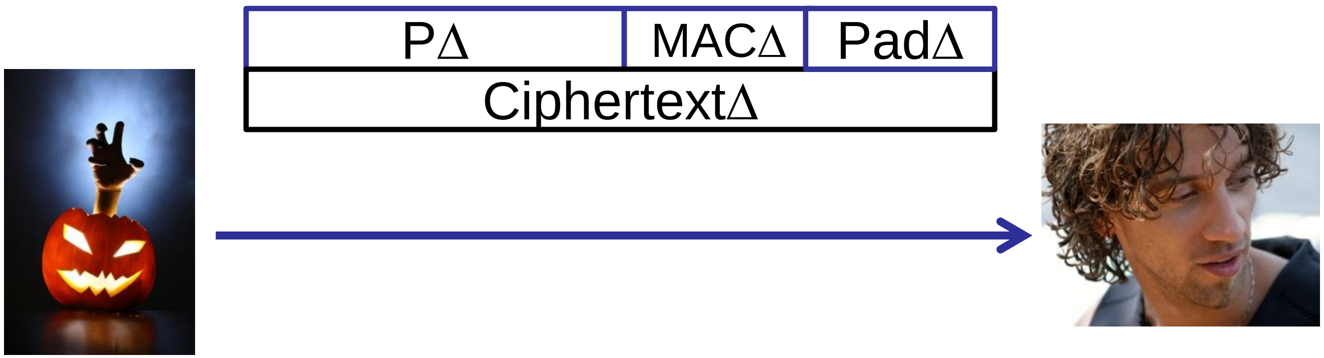
- Padding Oracle
  - The oracle tells the attacker if the padding in the message was correct.
  - This may be due to a *message with the information*.
  - It can also be due to *side channel like the response time*.

Concept of Padding Oracle Attack (against CBC)

- Attacker sees unknown ciphertext C =
  that was sent from Alice to Bob



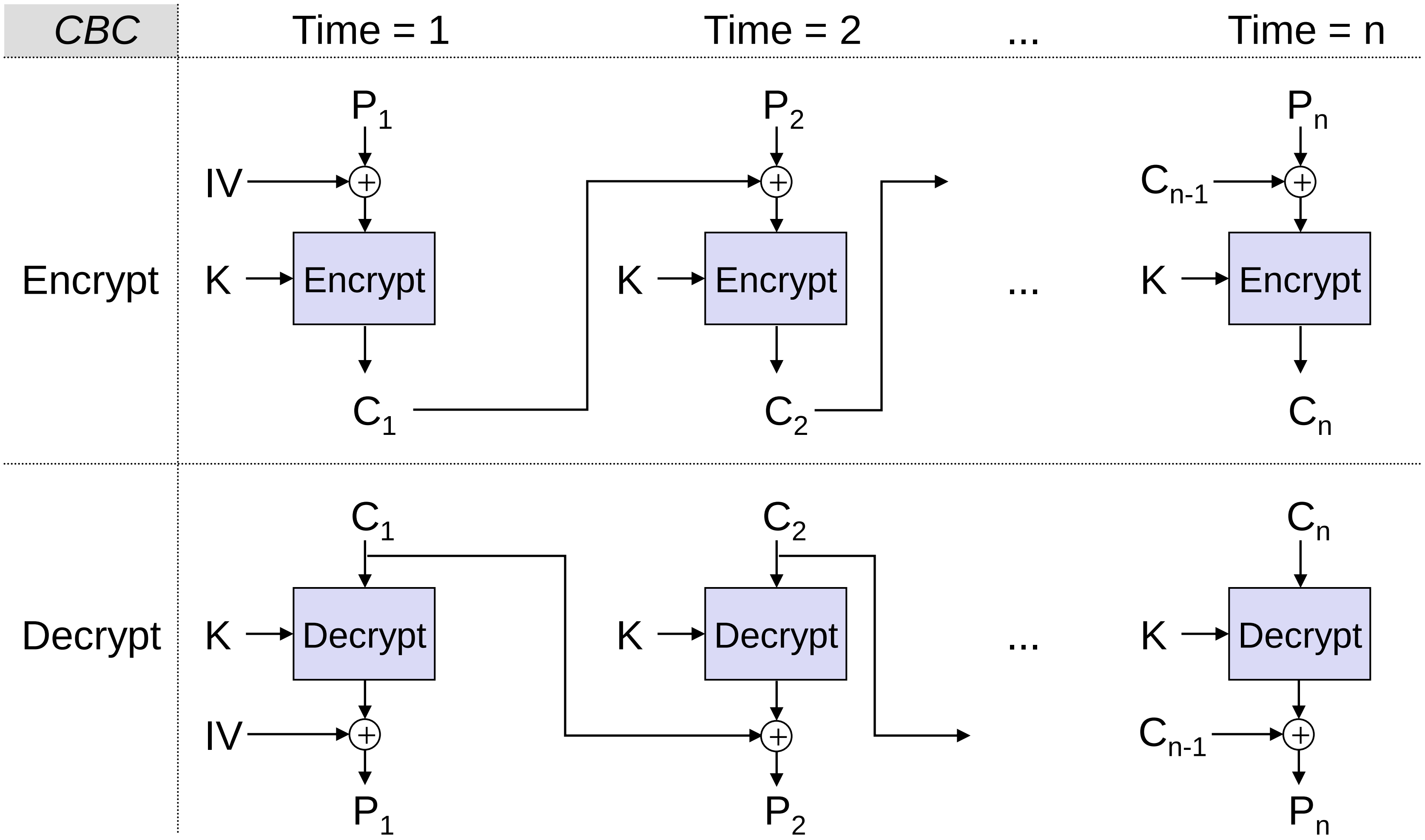
- To decrypt the ciphertext, the attacker modifies C and sends it to Bob.



- It is unlikely that the MAC and padding are correct. So, Bob will send an error back to Alice (and the attacker).
- In earlier versions of TLS, Bob sent back different error messages for padding errors and for MAC errors.

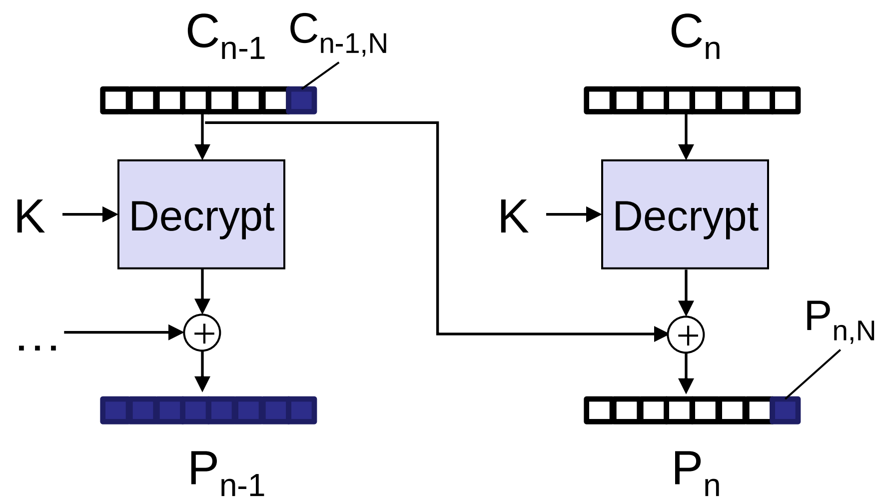
Padding Oracle Attack – CBC mode decryption (revisited)

- Encryption and Decryption in CBC mode



Padding Oracle Attack against CBC

- We have n blocks and N bytes per block. The attacker first wants to decrypt the last block C_n.
- In order to do so, he starts with the last byte $C_{n-1,N}$ of the block C_{n-1}. If he changes this byte (blue bytes are changed bytes)



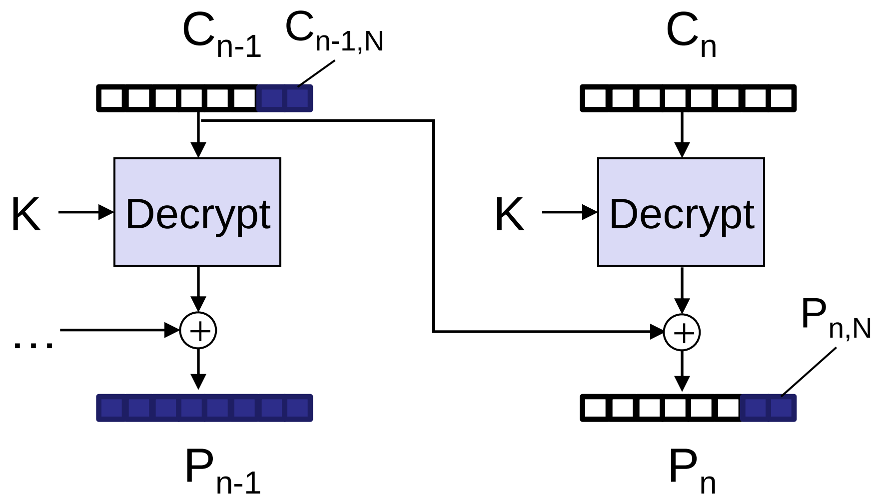
  - the MAC will most likely be invalid (chance 1 in 2^m for MAC length m)
  - the padding will be invalid unless $C_{n-1,N}$ xor $P_{n,N}$= 1 (chance 1 in 256)

➔ After testing the 256 values for $\mathbf{C_{n-1,N}}$ all of them produced padding errors except for one that matches $C_{n-1,N}$ xor $P_{n,N}$= 1.

➔ We know $P_{n,N}$. The original P is then Orig$P_{n,N}$= Orig$C_{n-1,N}$ xor $P_{n,N}$.

Padding Oracle Attack against CBC (2)

- Now, the byte $P_{n,N-1}$. For that we produce a padding of length 2.
- Since we know $P_{n,N}$ we can calculate $C_{n-1,N}$ so that $C_{n-1,N}$ xor $P_{n,N}$= 2
- Now, we have to find the $C_{n-1,N-1}$ that satisfies $C_{n-1,N-1}$ xor $P_{n,N-1}$= 2

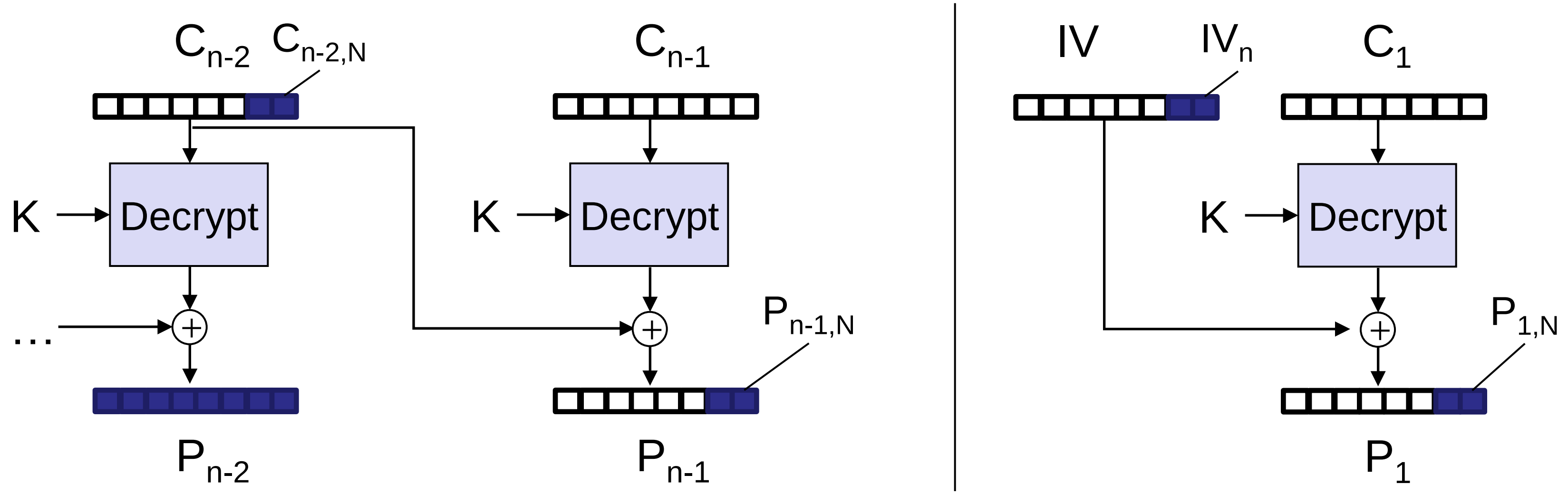


- With the same argument as before, we need to try up to 256 values, all values except for the correct one will generate a padding error. The correct one will produce a MAC error.

  ➔ We know $P_{n,N-1}$.

Padding Oracle Attack against CBC (3)

- To completely decrypt C_n we have to repeat the procedure until all bytes of the block are decrypted. In the figure with 8 bytes per block, the last padding we generate is 8 8 8 8 8 8 8 8.
- To decrypt C_{n-1} we can cut off C_n and repeat the same procedure with C_{n-1} as last block. For decrypting C_1 we can use the IV as ciphertext for the attack modifications.



Final Remarks

- The attack was against CBC mode used in MAC-then-Encode-then-Encrypt mode.
  - Padding Oracle attack known long in cryptography.
  - Mode still used in SSL / TLS. Hacks have utilized that. However, defenses have been added.

- CBC with Encode-then-Encrypt-then-MAC does not have this vulnerability.
  - Because MAC check would fail first, process would be aborted, and padding problems would then not be leaked.

Overview

- ❑ Part I: The Secure Channel
- ❑ Part II: Attacks against Secure Channel
- ❑ **Part III: Authenticated Encryption**

Authenticated Encryption

- Observations and Thoughts
  - Encryption → go over the data with some encryption mode
  - Integrity and authentication → go over the data with some MAC mode
  - Usually, both is needed. → Two passes over the data.
  - Difficult to do right. → Why not simplify process by providing both with one API call.

- Authenticated Encryption (AE)
  - Block Cipher Mode that provides Confidentiality, Integrity, and Authenticity
    - Any combination (e.g. AES-CTR-SHA-1-HMAC) would fall into the category
  - Some modern authenticated encryption modes do not *combine an encryption mode with a MAC mode*, but they provide *both in one mode*.
    - Needs only one pass over the data.
  - Examples for AE modes are GCM (Galois/Counter Mode), OCB (Offset Codebook Mode), CCM (Counter with CBC-MAC).

Offset Codebook Mode (OCB)

- Offset Codebook Mode
  - Authenticated Encryption Mode
  - Proposed 2001 [OCB1]
  - Standardized May 2014 [RFC 7253]

  - Encryption
    - Inspired by ECB with block-dependent offsets (avoids ECB problems!)
  - Associated Data *A*
    - *A* is not encrypted but authenticated
    - For example: Unencrypted header data
  - MAC
    - Checksum = XOR over plaintext, length- and key-dependent variables
    - MAC = (Encryption of checksum with shared key k) XOR (hash(k,A))

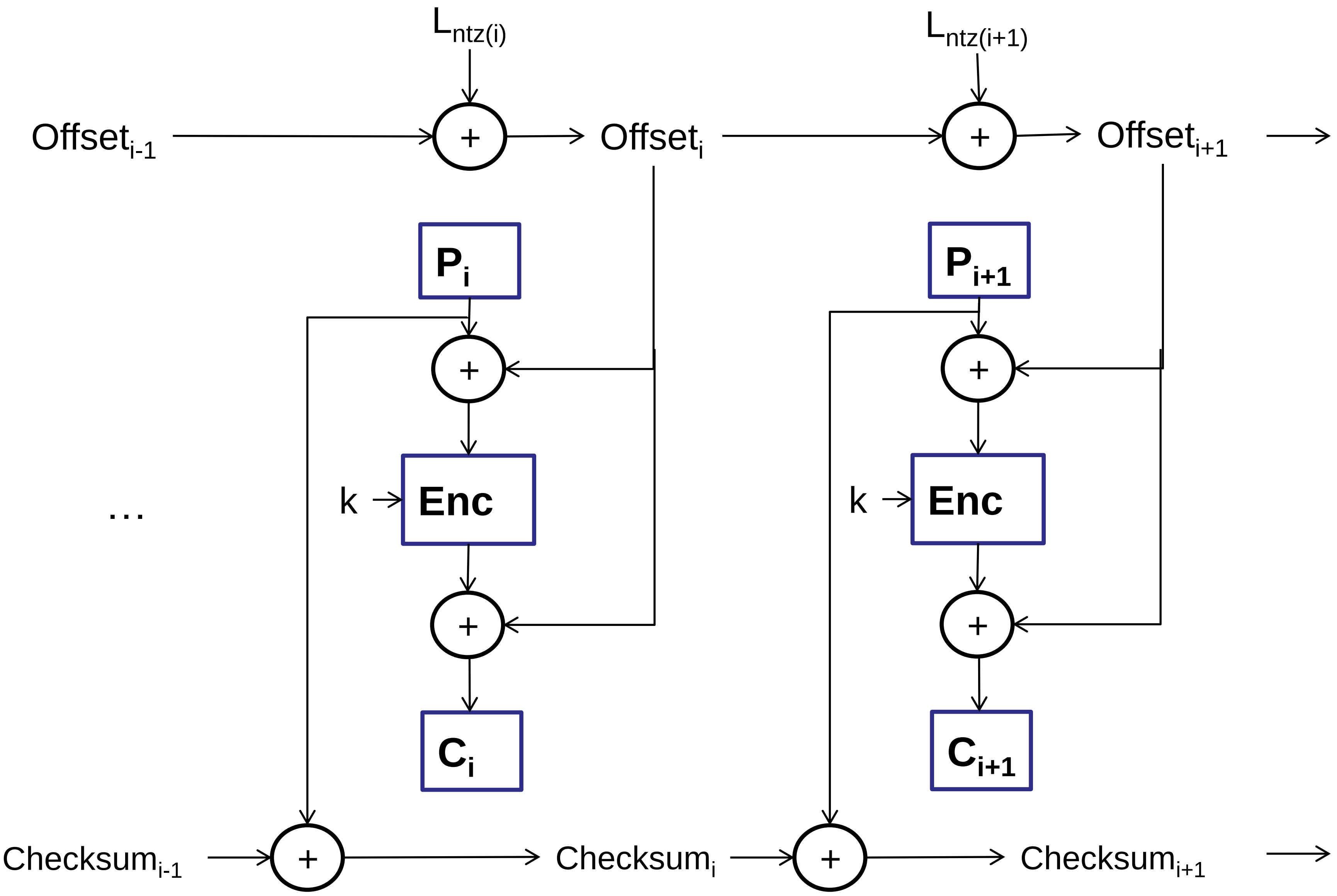
  - Requires only one key K for encryption and authentication
  - Requires a fresh nonce every time

Offset Codebook Mode

- Let *double* be multiplication by the variable in the OCB Galois Filed
- Variables depending on the key: $L_\star$, $L_\$$, L_0, L_1, L_2 …
  - $L_\star = Enc_K(0)$
  - $L_\$ = \text{double}(L_\star)$
  - $L_0 = \text{double}(L_\$)$
  - $L_i = \text{double}(L_{i-1})$
- Let *ntz* be number of trailing zeros (zero bits at the end)
- Usage of the L's
  - $L_\$$ → MAC
  - $L_\star$ → Last Block
  - $L_{ntz(i)}$ → intermediate blocks

- Note: $L_{\mathbf{ntz(i)}}$ is used
  - Only few L_i are needed (for a fixed K)
  - They can be pre-computed and stored in a **L**ookup table

Offset Codebook Mode (OCB)



OCB Initialization

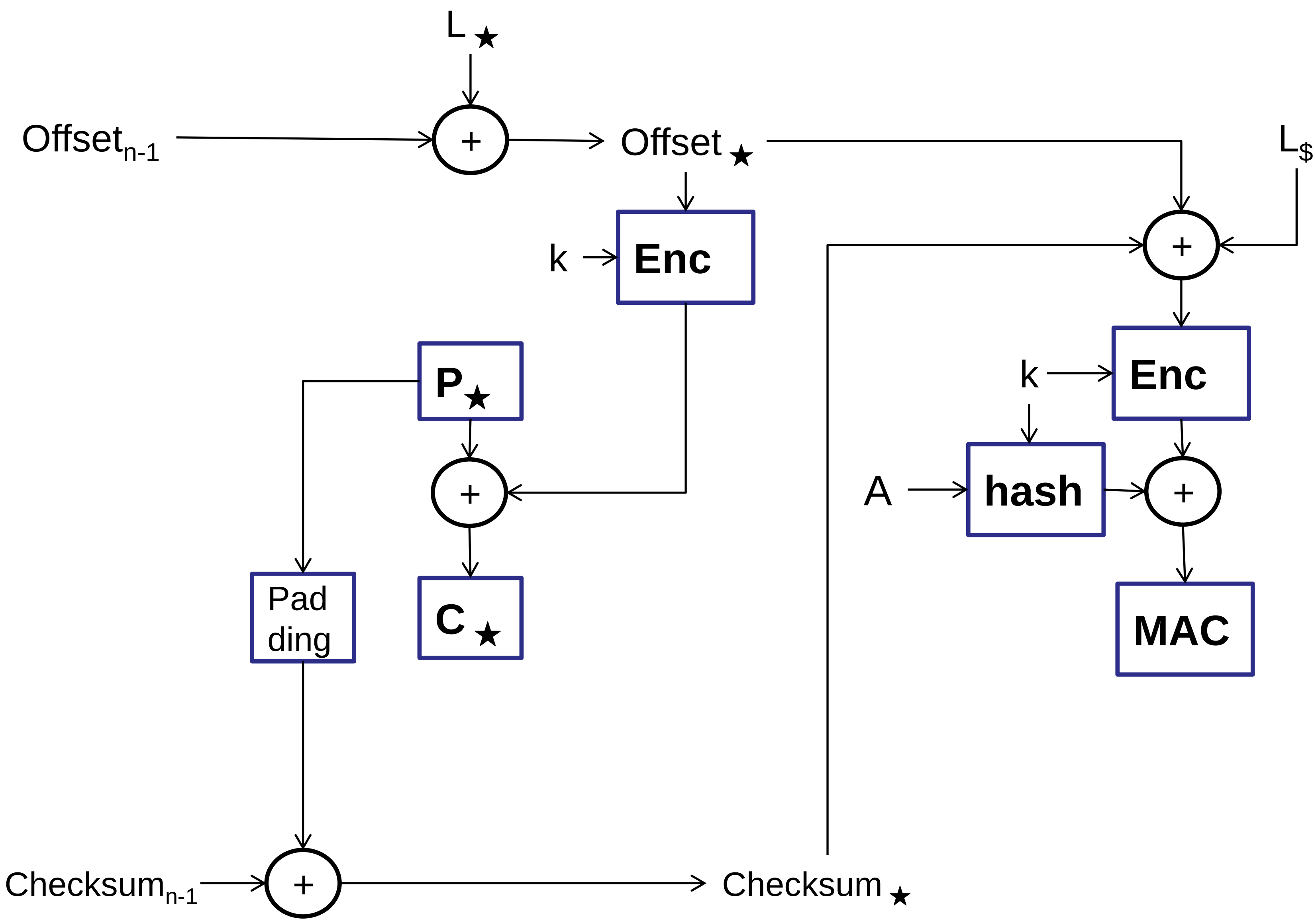
- $Offset_0$ depends on the key and the **nonce**
- "It is crucial that, as one encrypts, one does not repeat a nonce."

  [RFC 7253, § 5.1]
- Nonce *may* not be random, e.g. a counter works fine
- A new nonce for every authenticated encryption API call is needed!

- Details about the initialization:

  http://www.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm

OCB – Last Block and MAC



Offset Codebook Mode

- Question: XOR plaintext and then encrypt, that sounds like the weak MAC example from Chapter 2.2. Why is OCB more secure than the easy-to-break example?

- "OCB enjoys provable security: the mode of operation is secure assuming that the underlying blockcipher is secure. As with most modes of operation, security degrades as the number of blocks processed gets large" [RFC 7253]

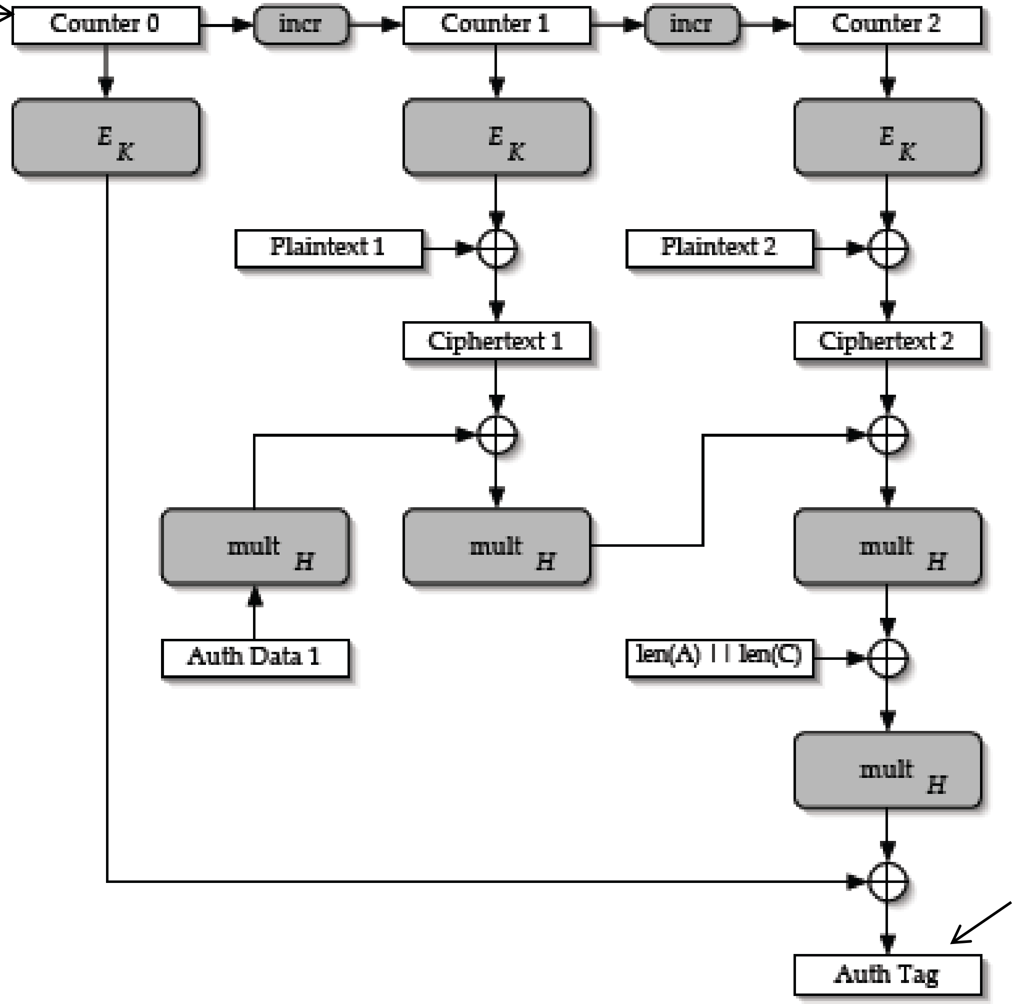
Galois/Counter Mode (GCM)

- Galois/Counter Mode (GCM)
  - Developed by John Viega and David A. McGrew
  - Standardized by NIST in 2007, IETF standards for cipher suites with AES-GCM for TLS (SSL) and IPSec exist.
  - Follows the Encrypt-then-MAC concept
  - Combines concept of Counter Mode for encryption with Galois Field Multiplication to compute MAC on the ciphertext
  - GF(2^128) based on polynomial x^128 + x^7 + x^2 + x+1

- Definitions
  - H is Enc(k,0)
  - Auth Data is data not to be encrypted. GCM generates check value by XOR and GF multiplication with H for each block.
  - For the MAC, this process continues on the ciphertext and a length field in the end.

Galois/Counter Mode (GCM)

Starts with IV, not with 0.



MAC

Image from Wikipedia, Author from NIST.

Galois Field Multiplication

- In a Galois Field we consider the bitstring to represent a polynomial.
  - E.g. 1011 = $x^3 + x + 1$
- As a consequence Galois Field Multiplication is based on polynomial multiplication modulus the polynomial of the field.

- Example: In GF(2^{128}) based on polynomial $g(x) = x^{128} + x^7 + x^2 + x+1$
  - $P(x) = x^{127}+x^7$
  - $Q(x) = x^5 + 1$
  - $P(x)*Q'(x) = x^{132} + x^{127} + x^{12} + x^7$
  - To compute the modulus, we have to compute a polynomial division $P(x)*Q(x)/g(x)$.
  - We can see that $x^4 * g(x)$ removes the x^{132}, so $P(x)*Q(x)-x^4*g(x) = x^{127} + x^{12} + x^{11} + x^7 + x^6 + x^5 + x^4$
  - Since this polynomial fits into the 128 bit, this is the remainder of the division, thus the result, in bits: 1000…01100011110000.

References

[Bell95] M. Bellare and P. Rogaway, Provably Secure Session Key Distribution - The Three Party Case, Proc. 27th STOC, 1995, pp 57--64

[Boyd03] Colin Boyd, Anish Mathuria, “Protocols for Authentication and Key Establishment”, Springer, 2003

[Bry88a] R. Bryant. *Designing an Authentication System: A Dialogue in Four Scenes.* Project Athena, Massachusetts Institute of Technology, Cambridge, USA, 1988.

[Diff92] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. Designs, Codes, and Cryptography, 1992

[Dol81a] D. Dolev, A.C. Yao. *On the security of public key protocols.* Proceedings of IEEE 22nd Annual Symposium on Foundations of Computer Science, pp. 350-357, 1981.

[Fer00] Niels Ferguson, Bruce Schneier, “A Cryptographic Evaluation of IPsec”. http://www.counterpane.com/ipsec.pdf 2000

[Fer03] Niels Ferguson, Bruce Schneier, „Practical Cryptography“, John Wiley & Sons, 2003

[Gar03] Jason Garman, “Kerberos. The Definitive Guide”, O'Reilly Media, 1st Edition, 2003

References

[Kau02a] C. Kaufman, R. Perlman, M. Speciner. *Network Security*. Prentice Hall, 2nd edition, 2002.

[Koh94a] J. Kohl, C. Neuman, T. T'so, *The Evolution of the Kerberos Authentication System.* In Distributed Open Systems, pages 78-94. IEEE Computer Society Press, 1994.

[Mao04a] W. Mao. *Modern Cryptography: Theory & Practice*. Hewlett-Packard Books, 2004.

[Nee78] R. Needham, M. Schroeder. *Using Encryption for Authentication in Large Networks of Computers.* Communications of the ACM, Vol. 21, No. 12, 1978.

[Woo92a] T.Y.C Woo, S.S. Lam. *Authentication for distributed systems.* Computer, 25(1):39-52, 1992.

[Lowe95] G. Lowe, „An Attack on the Needham-Schroeder Public-Key Authentication Protocol", *Information Processing Letters*, volume 56, number 3, pages 131- 133, 1995.

References

[OCB1] Rogaway, P., Bellare, M., Black, J., and T. Krovetz, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption", ACM Conference on Computer and Communications Security 2001 - CCS

[OCB] T.Krovetz, P. Rogaway, „The OCB Authenticated-Encryption Algorithm“ http://tools.ietf.org/html/draft-irtf-cfrg-ocb-03

[RFC 4106] The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)

[RFC 5288] AES Galois Counter Mode (GCM) Cipher Suites for TLS.

[RFC 7253] The OCB Authenticated-Encryption Algorithm

Additional references from the IETF

[RFC2560] M. Myers, et al., “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP”, June 1999

[RFC3961] K. Raeburn, “Encryption and Checksum Specifications for Kerberos 5”, February 2005

[RFC3962] K. Raeburn, “Advanced Encryption Standard (AES) Encryption for Kerberos 5”, February 2005

[RFC4757] K. Jaganathan, et al., “The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows ”, December 2006

[RFC4120] C. Neuman, et al., “The Kerberos Network Authentication Service (V5)”, July 2005

[RFC4537] L. Zhu, et al, “Kerberos Cryptosystem Negotiation Extension”, June 2006

[RFC5055] T. Freeman, et al, “Server-Based Certificate Validation Protocol (SCVP)”, December 2007