TU Graz

SCIENCE
PASSION
TECHNOLOGY

Cryptography 3: Encryption

Maria Eichlseder

Information Security – WT 2019/20

You Are Here

Crypto 1	Crypto 2	Crypto 3	Crypto 4
Introduction to InfoSec & Crypto	Symmetric Authentication	Symmetric Encryption	Asymmetric Cryptography
Terminology	Integrity	Confidentiality	Establishing communication
Security notions	Hash functions	AEAD (Auth. Encryption)	Key exchange
Keys, Kerckhoffs' principle	MACs (Message Authentication)	Symmetric primitives	Signatures
			Asymmetric primitives

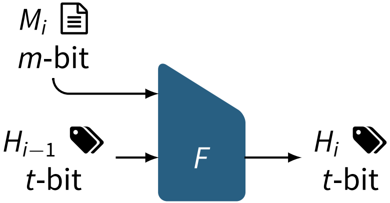
Recap of Last Week (1): Schemes for Message Authentication

Cryptographic schemes for message authentication compute a short, fixed-length Tag T from the Message M and (in some cases) a Key K.

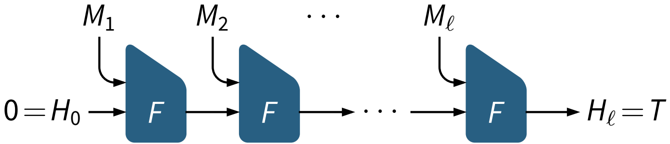
Hash Function $\mathcal{H}$	MAC $\mathcal{H}_{K_{AB}}$	Signature $\mathcal{S}_{K_A}$
Unkeyed	Symmetric Key K_{AB}	Asymmetric Key K_A
Anyone can compute T	A, B can compute T	A can compute T
Anyone can verify T	A, B can verify T	Anyone can verify T

Recap of Last Week (2): Merkle–Damgård Hashing

Primitive: Compression Function F (fixed-size inputs)



Mode: Merkle–Damgård (MD) Hash Function $\mathcal{H}(M) = T$ (variable-size inputs)



Outline

Confidentiality

Introduction

Confidentiality

Confidentiality of Data

Prevent unauthorized entities from learning information (messages, data) that authorized parties are communicating or processing.

There are several related, but different concepts:

- Anonymity: The users' identity is unknown, they are not identifiable within a certain set of users
- Privacy: The users are able to seclude themselves, or information about themselves, and thereby express themselves selectively. This often refers to sensitive personal information.

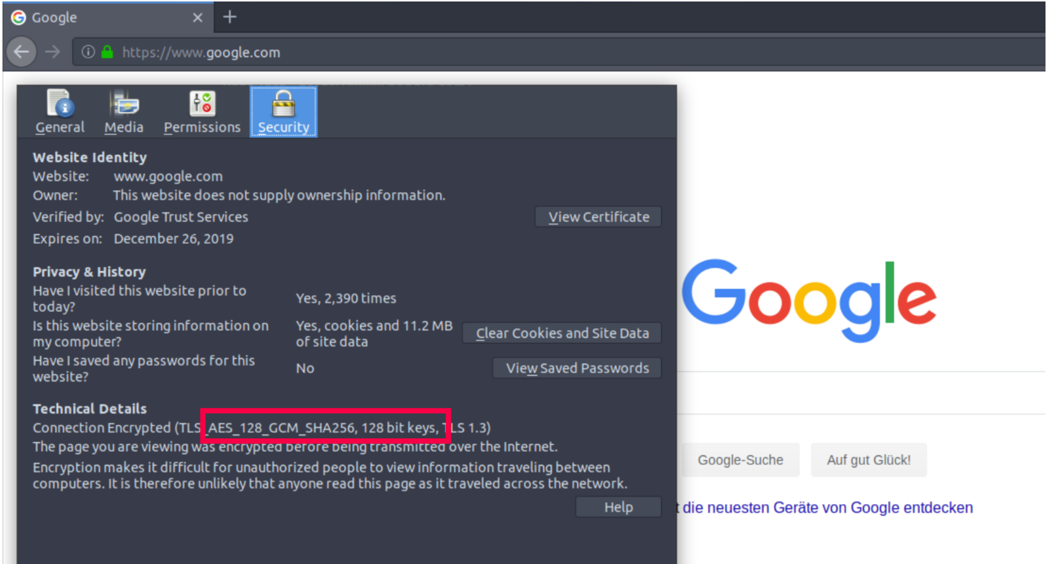
Cryptographic Schemes for Encryption

Encryption schemes transform a plaintext Message M of arbitrary length to a Ciphertext C of about the same length based on a Key K of fixed length.

Schemes may require additional inputs or produce an authentication Tag T.

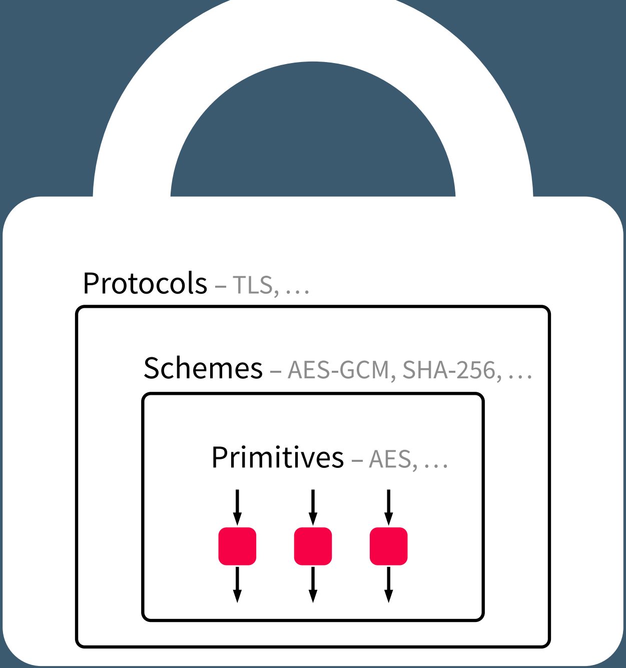
Encryption $\mathcal{E}_{K_{AB}}$	Auth. Enc. $\mathcal{AE}_{K_{AB}}$	Key Encapsulation
Symmetric Key K_{AB}	Symmetric Key K_{AB}	Asymmetric Keys
Confidentiality only	Confid. + Authenticity	Confidentiality
A, B can encrypt	A, B can encrypt + auth	Anyone can encrypt
A, B can decrypt	A, B can decrypt + verify	A can decrypt

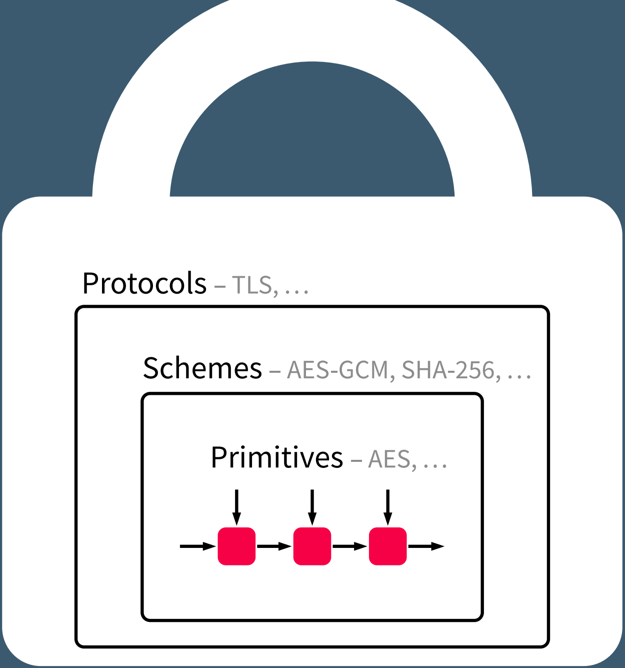
Examples (1): Secure Communication with HTTPS



Example (2): Disk Encryption with LUKS

```
meichlseder@x1tblme ~ % sudo cryptsetup luksDump /dev/nvme0n1p3
[sudo] password for meichlseder:
LUKS header information
Version:        2
Epoch:          4
Metadata area:  16384 [bytes]
Keyslots area:  16744448 [bytes]
UUID:           087c56a9-a282-42f2-8361-869ec488e61e
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          (no flags)

Data segments:
  0: crypt
        offset: 16777216 [bytes]
        length: (whole device)
        cipher: aes-xts-plain64
        sector: 512 [bytes]

Keyslots:
  0: luks2
        Key:        512 bits
        Priority:   normal
        Cipher:     aes-xts-plain64
        Cipher key: 512 bits
        PBKDF:      argon2i
        Time cost:  3
        Memory:     1048576
        Threads:    4
        Salt:       81 0d d7 18 01 e4 1d d9 6c 14 68 08 95 f5 f4 73
                    fc 8c 32 9a 4e 94 a0 aa 23 91 6b 2a 6d 66 51 13
        AF stripes: 4000
        AF hash:    sha256
```


Protocols – TLS, . . .
Schemes – AES-GCM, SHA-256, . . .
Primitives – AES, . . .

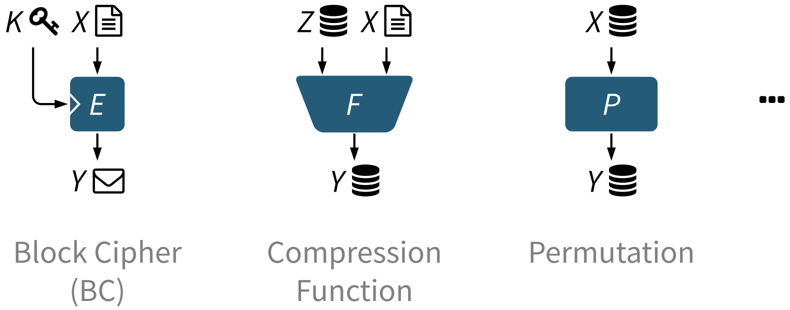


Protocols – TLS, . . .
Schemes – AES-GCM, SHA-256, . . .
Primitives – AES, . . .

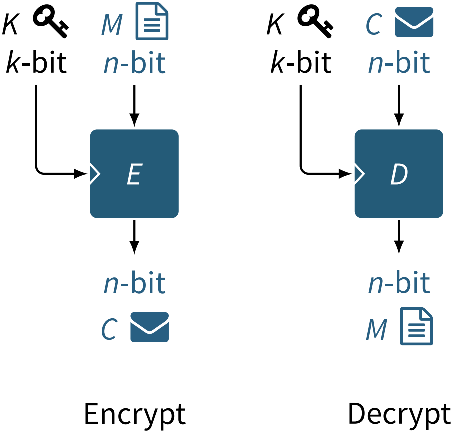
Symmetric Primitives

Secure Building Blocks

Symmetric Primitives



Block Ciphers – Key Space and Plaintext Space



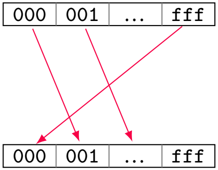
A block cipher is a family of permutations (bijective functions) E_K.

Each k-bit key K defines a permutation E_K that encrypts n-bit message blocks M to n-bit ciphertext blocks $C = E_K(M)$.

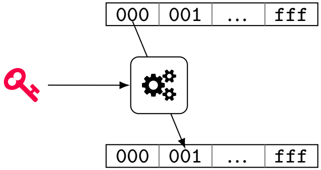
It also defines the inverse permutation $D_K = E_K^{-1}$ that maps C back to M.

- 2^n possible inputs/outputs M
- 2^k possible keys (mappings) K

Ideal Block Ciphers vs. Real Block Ciphers



Ideal

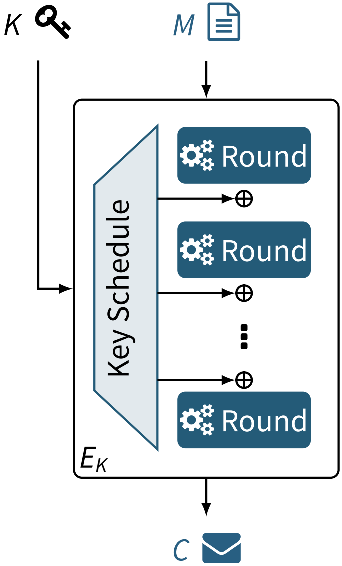


Real

A block cipher instance with a fixed key is a permutation that assigns some n-bit output C to each n-bit input M. It can be seen as a large, secret lookup table (codebook). In total, there are $(2^n)!$ possible instances – this is an astronomical number ($\gg 2^{256}$) already for $n = 6$.

- An ideal block cipher is a uniformly random selection of 2^k instances.
- A real block cipher must implement an algorithm to map M to C using K.
  An attacker must still be unable to recover M from only C, or K from M and C.

Making It Implementable – The Key-Alternating Construction

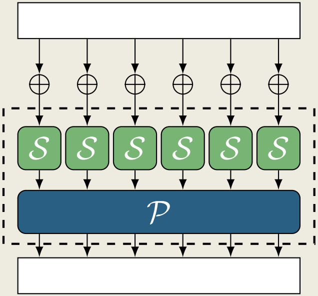


Two fundamental ideas:

1. Repeat simple circuit r times: the "round function"
   - Avoid huge lookup tables
2. Make the round circuit public but mix input with round key
   - Avoid key-dependent circuitry
   - Kerckhoffs' principle

How to Build an Invertible Round Function

SPN Round (incl. Key-XOR)



Decryption requires inverses of S-box and permutation circuit

- A popular approach is the Substitution-Permutation-Network (SPN).

  Its round function has 2 parts:

- S-box layer ($\mathcal{S}$): Decomposes state in small chunks, each chunk is substituted using a small lookup table (= "S-box")

- Linear layer ($\mathcal{P}$): Applies an invertible linear function, using bit permutation and/or XORs

The AES Competition (1997–2000)

- AES – Advanced Encryption Standard
- Goals: A block cipher to replace DES
  - The previous Data Encryption Standard (DES) was co-designed by NSA
  - Its security level was no longer adequate (small key, cryptanalysis)
- Organized by NIST (US Institute of Standards and Technology)
- Announced 1997, 15 submissions from 50 cryptographers
- Winner: `Rijndael/AES`, designed by Joan Daemen and Vincent Rijmen
  - Now used *everywhere* for secure encryption

AES – State and Operations

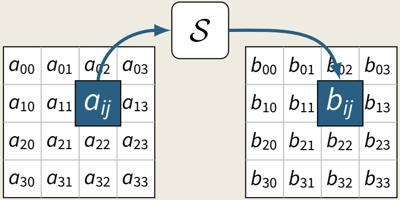
- Block size $n = 128$ bits, Key size $k \in \{128, 192, 256\}$ bits
- 3 Block ciphers named after their key size: AES-128, AES-192, AES-256
- The 16-byte input block $M = s_{00} \| s_{10} \| s_{20} \| s_{30} \| s_{01} \| \dots \| s_{33}$ is written as a 4×4 matrix of bytes, the $\{16, 24, 32\}$-byte key K as a $4 \times \{4, 6, 8\}$ matrix:

$$M = \begin{array}{|c|c|c|c|} \hline s_{00} & s_{01} & s_{02} & s_{03} \\ \hline s_{10} & s_{11} & s_{12} & s_{13} \\ \hline s_{20} & s_{21} & s_{22} & s_{23} \\ \hline s_{30} & s_{31} & s_{32} & s_{33} \\ \hline \end{array}, \qquad K = \begin{array}{|c|c|c|c|c|c|c|c|} \hline k_{00} & k_{01} & k_{02} & k_{03} & k_{04} & k_{05} & k_{06} & k_{07} \\ \hline k_{10} & k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} & k_{17} \\ \hline k_{20} & k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} & k_{27} \\ \hline k_{30} & k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36} & k_{37} \\ \hline \end{array}$$

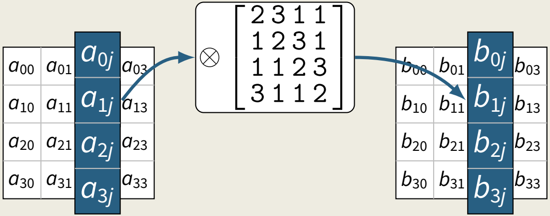
- The state is initialized to M and updated in 10 rounds (for AES-128) or 12 rounds (AES-192) or 14 rounds (AES-256). The last round is different.

AES Round Function – Overview

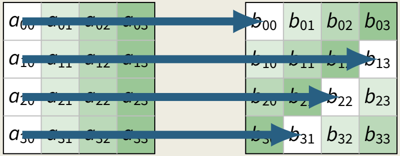
1 SubBytes (SB)



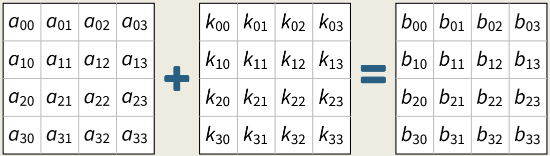
3 MixColumns (MC)



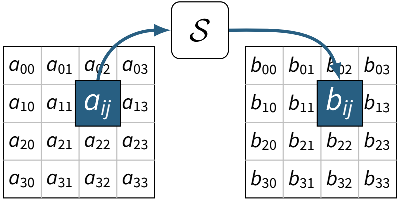
2 ShiftRows (SR)



4 AddRoundKey (AK)

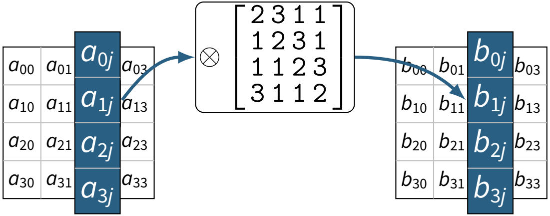


AES Round Function – SubBytes (SB)



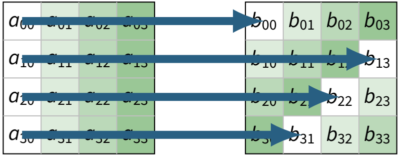
- S-box layer: $b_{ij} = \mathcal{S}[a_{ij}]$
- Each of the 16 state bytes a_{ij} is substituted using an 8-bit lookup table
  $\mathcal{S}[\texttt{0x00}] = \texttt{0x63}, \quad \mathcal{S}[\texttt{0x01}] = \texttt{0x7C}, \quad \mathcal{S}[\texttt{0x02}] = \texttt{0x77}, \quad \ldots, \quad \mathcal{S}[\texttt{0xFF}] = \texttt{0x16}$
- The S-box $\mathcal{S}$ has strong cryptanalytic properties to defend against attacks

AES Round Function – MixColumns (MC)



- Part of the linear layer: $(b_{0j}, b_{1j}, b_{2j}, b_{3j}) = M \cdot (a_{0j}, a_{1j}, a_{2j}, a_{3j})$
- Each column of the state is updated using a multiplication with a matrix M
  (this multiplication is over a "finite field", not normal integer multiplication!)
- If one byte at the input changes, all output bytes in the column will change
- This step is omitted in the last round

AES Round Function – `ShiftRows` (SR)



- Part of the linear layer: $b_{i,j} = a_{i,j+i\,\%\,4}$
- Each row i of the state is rotated to the left by i bytes
- The values of one column are shifted to four different columns

$\texttt{AES}$ Round Function – $\texttt{AddRoundKey}$ ($\texttt{AK}$)

Play with it here

$$\begin{array}{|c|c|c|c|}\hline a_{00} & a_{01} & a_{02} & a_{03} \\ \hline a_{10} & a_{11} & a_{12} & a_{13} \\ \hline a_{20} & a_{21} & a_{22} & a_{23} \\ \hline a_{30} & a_{31} & a_{32} & a_{33} \\ \hline \end{array} + \begin{array}{|c|c|c|c|}\hline k_{00} & k_{01} & k_{02} & k_{03} \\ \hline k_{10} & k_{11} & k_{12} & k_{13} \\ \hline k_{20} & k_{21} & k_{22} & k_{23} \\ \hline k_{30} & k_{31} & k_{32} & k_{33} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline b_{00} & b_{01} & b_{02} & b_{03} \\ \hline b_{10} & b_{11} & b_{12} & b_{13} \\ \hline b_{20} & b_{21} & b_{22} & b_{23} \\ \hline b_{30} & b_{31} & b_{32} & b_{33} \\ \hline \end{array}$$

- Key-alternating construction: $b_{ij} = a_{ij} \oplus k_{ij}^{(r)}$
- XOR the round key $k^{(r)}$ of round r to the state
- The round keys $k_{ij}^{(r)}$ are derived from the key K using the key schedule
  (details omitted – the key schedule uses similar operations to the round function)
- An additional `AddRoundKey` step happens before the first round

Symmetric Primitives – Conclusion

- Primitives are the foundation of security in symmetric cryptography
- Their security cannot be "proven", but only "analyzed"

- Symmetric primitives in TLS 1.3:
  - `AES`-$\{128, 256\}$ block cipher
  - `ChaCha20` stream cipher
  - `SHA`-$\{256, 512\}$ compression function
- All of these are expected to provide long-term security (also in a post-quantum world)

Encryption

Protecting Confidentiality

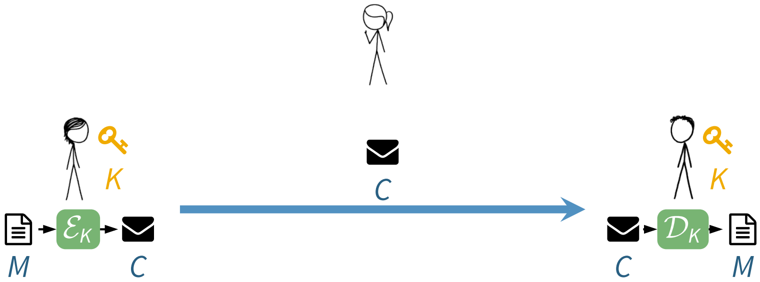
Encryption Schemes – Definition, First Attempt

An encryption scheme is a keyed function $\mathcal{E}_K$ that maps a k-bit key K and a message M of arbitrary length to a ciphertext C, together with its inverse decryption function $\mathcal{D}_K$, to protect the confidentiality of M:

$$\mathcal{E}_K : \mathbb{F}_2^k \times \mathbb{F}_2^* \to \mathbb{F}_2^*, \qquad \mathcal{E}_K(M) = C$$
$$\mathcal{D}_K : \mathbb{F}_2^k \times \mathbb{F}_2^* \to \mathbb{F}_2^*, \qquad \mathcal{D}_K(C) = M$$

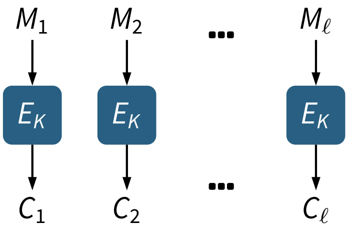
The challenge of protecting the confidentiality of M is transformed into protecting the confidentiality of K.

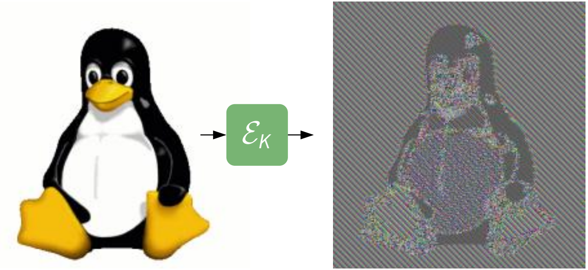
Encryption Schemes – Application, First Attempt



1. Alice computes $C = \mathcal{E}_K(M)$
2. Alice transmits C to Bob (over an insecure channel controlled by Eve)
3. Bob computes $M = \mathcal{D}_K(C)$

How NOT To Do It – The Electronic CodeBook mode (ECB)





Split M into blocks $M_1, M_2, \ldots, M_\ell$ and encrypt each block with block cipher E_K.

This simple mode has 2 major problems:

- Patterns: Two identical blocks M_i, M_j get encrypted to the same C_i, C_j
- Context: Two identical messages M, M' get encrypted to the same C, C'

Encryption Schemes – Definition, Revisited

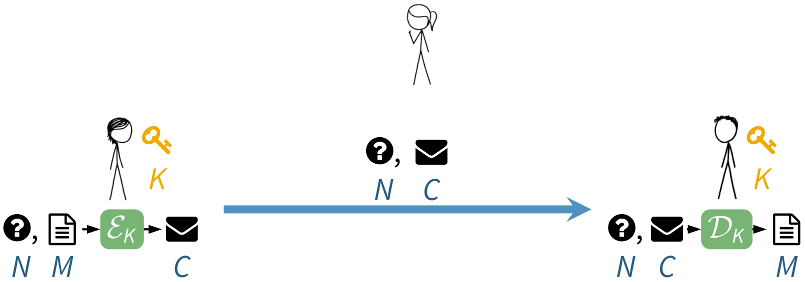
An encryption scheme is a keyed function $\mathcal{E}_K$ that maps a k-bit key K, n-bit nonce N, and a message M of arbitrary length to a ciphertext C, together with its inverse decryption function $\mathcal{D}_K$, to protect the confidentiality of M:

$$\begin{aligned} \mathcal{E}_K &: \mathbb{F}_2^k \times \mathbb{F}_2^n \times \mathbb{F}_2^* \to \mathbb{F}_2^*, & \mathcal{E}_K(N, M) &= C \\ \mathcal{D}_K &: \mathbb{F}_2^k \times \mathbb{F}_2^n \times \mathbb{F}_2^* \to \mathbb{F}_2^*, & \mathcal{D}_K(N, C) &= M \end{aligned}$$

The nonce (number used only once) makes sure that an adversary can't tell if two encrypted messages are the same!

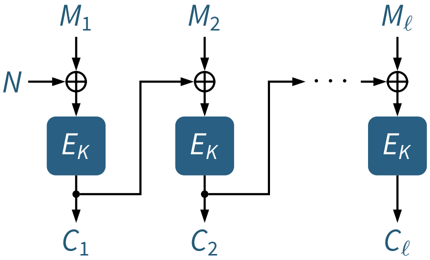
In practice, N can be randomly generated or (in some cases only!) a counter.

Encryption Schemes – Application, Revisited



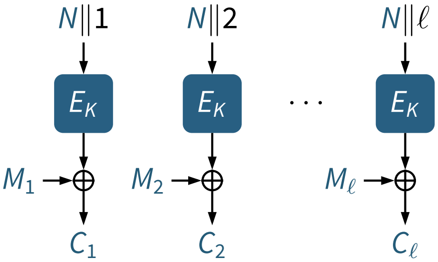
1. Alice computes $C = \mathcal{E}_K(N, M)$
2. Alice transmits N and C to Bob (over an insecure channel controlled by Eve)
3. Bob computes $M = \mathcal{D}_K(N, C)$

Cipher Block Chaining mode (CBC)



- **Goal:** C_i should depend on the "context", i.e., on blocks $M_1, \ldots, M_i$ and the nonce N.
- **Idea** similar to CBC-MAC: XOR $\oplus$ previous ciphertext block C_{i-1} (= chaining value) to msg block M_i, then encrypt with E_K
- **Idea**: Start with random (!) nonce N to hide repeated messages
- Must be combined with a suitable padding scheme for the message M.

CounTeR mode (CTR)



- **Goal**: C_i should depend on the "context", i.e., on block M_i, position i, and the nonce N.
- **Idea**: Create a streaming mode that produces a keystream depending on K, N and XOR it to M
- Nonce N can be random (unpredictable) or a counter (predictable), as long as it never repeats for the same K
- No padding needed, $\mathtt{len}(C) = \mathtt{len}(M)$

Encryption in Practice

- `CBC` and `CTR` provide only confidentiality, no authenticity
- There are VERY FEW applications that need pure (unauthenticated) encryption or where authenticated encryption doesn't fit.

  Example: some file system encryption schemes (no space for tags)
- Usually you instead want Authenticated Encryption!

Authenticated Encryption

Protecting Confidentiality and Authenticity

Authenticated Encryption – Goals

If your data is worth encrypting,
you almost certainly don't want it modified!

- Confidentiality
  as provided by block cipher modes $\mathcal{E}_K$

- Authenticity, integrity
  as provided by message authentication codes $\mathcal{H}_K$

> *it is very easy to accidentally combine secure encryption schemes with secure MACs and still get insecure authenticated encryption schemes*

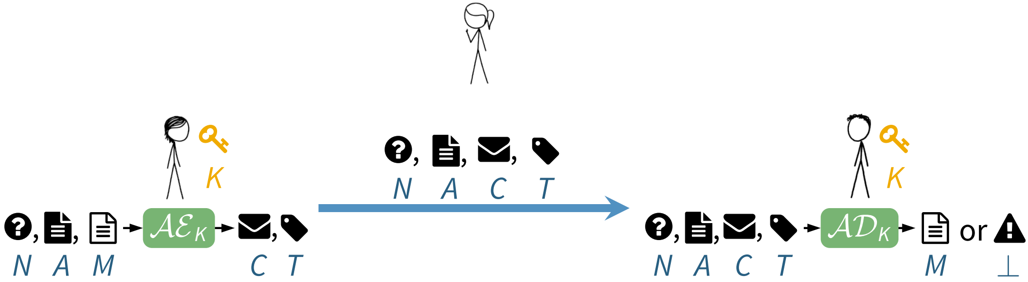
Authenticated Encryption

An Authenticated Encryption scheme is a keyed function $\mathcal{AE}_K$ that maps a k-bit key K, n-bit nonce N, and a message M of arbitrary length to a ciphertext C with attached tag T to protect both confidentiality and authenticity of M. Its inverse verified decryption function $\mathcal{AD}_K$ returns either the message M or, on invalid ciphertexts, an error $\perp$.

$$\begin{aligned} \mathcal{AE}_K &: \mathbb{F}_2^k \times \mathbb{F}_2^n \times \mathbb{F}_2^* &&\to \mathbb{F}_2^* \times \mathbb{F}_2^t, && \mathcal{AE}_K(N, M) = C, T \\ \mathcal{AD}_K &: \mathbb{F}_2^k \times \mathbb{F}_2^n \times \mathbb{F}_2^* \times \mathbb{F}_2^t &&\to \mathbb{F}_2^* \cup \{\perp\}, && \mathcal{AD}_K(N, C, T) = M \end{aligned}$$

Modern AE schemes are usually AEAD (Authenticated Encryption with Associated Data) schemes and accept an additional input A of arbitrary length and protects its authenticity, but not confidentiality (e.g., metadata, addressing) – so it can also be used as a MAC.

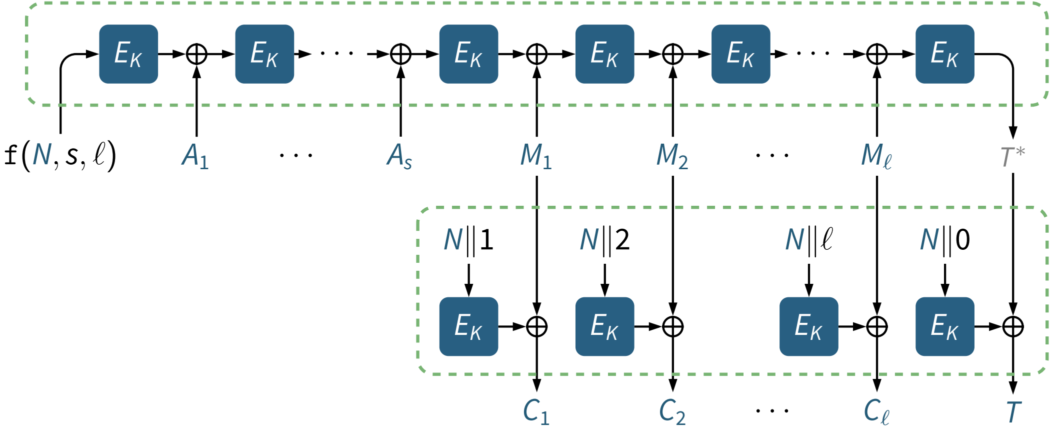
Authenticated Encryption



Important:

- $\mathcal{AE}_K$: Nonce N must never repeat for the same K; a counter is usually ok
- $\mathcal{AD}_K$: (Parts of) Message M must never be released before verifying T

Example: CCM Mode – CTR encryption with CBC-MAC authentication



$A_1, \ldots, A_s$ and $M_1, \ldots, M_\ell$ are the blocks of the padded A and M.
$\mathtt{f}(N, s, \ell)$ encodes various parameters in one block (details here).

Popular Authenticated Ciphers

In TLS 1.3:

- `AES-GCM` (the TLS default), with `AES-`$\{128, 256\}$
- `AES-CCM`
- `ChaCha20-Poly1305` (not based on `AES`, uses `ChaCha20` stream cipher)

The Key

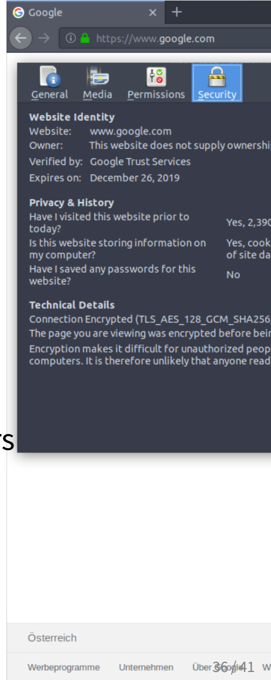
Where Do Keys Come From? (1)

In most cases, they are generated randomly

- Short-term keys distributed with key exchange, or
- Long-term keys (e.g., in 2-factor-authentication devices)

High-quality random numbers are essential

- DO use Cryptographically Secure Random Number Generators (CSPRNGs) like `/dev/urandom`, `OpenSSL` library, ...
- DON'T use "cheap" random numbers, low-quality seeds, PRNGs without guarantees like `std::random_device()`, non-secure updates like Mersenne Twister, ...



Random Number Generators – 2 Types

Nondeterministic hardware source

A hardware random number generator (HRNG) or true random number generator (TRNG) is a device that generates random numbers from a physical process (such as quantum effects or other microscopic effects), rather than by means of an algorithm.

Deterministic pseudorandomness

A pseudorandom number generator (PRNG) or deterministic random bit generator (DRBG) is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers. This sequence is not truly random, but completely determined by an initial value, the seed. A PRNG can be built from cryptographic schemes like stream ciphers or hash functions.

Where Do Keys Come From? (2)

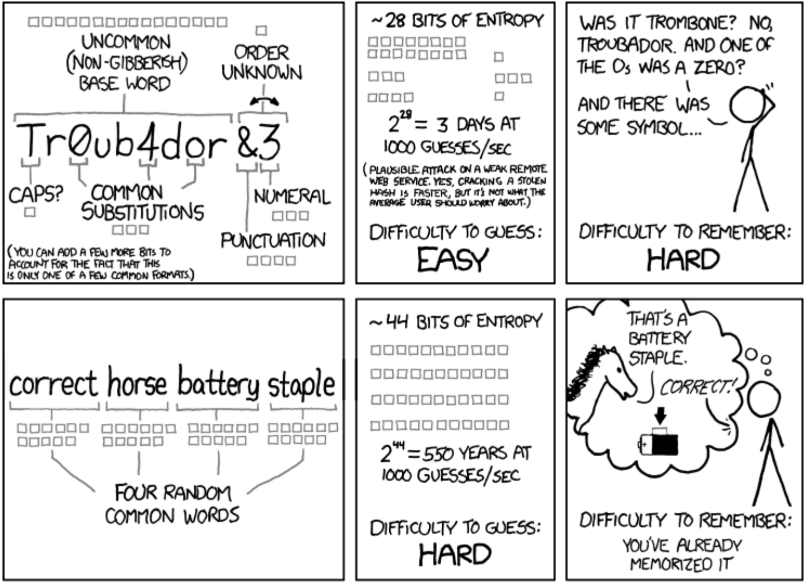
- Sometimes, they are derived from other secrets like passwords
- Password-Based Key Derivation Functions (PBKDFs), similar to (password) hash functions
- The quality of the key depends on the quality of the password

```
meichlseder@x1tblme ~ % sudo cryptsetup lu
[sudo] password for meichlseder:
LUKS header information
Version:        2
Epoch:          4
Metadata area:  16384 [bytes]
Keyslots area:  16744448 [bytes]
UUID:           087c56a9-a282-42f2-8361-8
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          (no flags)

Data segments:
  0: crypt
        offset: 16777216 [bytes]
        length: (whole device)
        cipher: aes-xts-plain64
        sector: 512 [bytes]

Keyslots:
  0: luks2
        Key:        512 bits
        Priority:   normal
        Cipher:     aes-xts-plain64
        Cipher key: 512 bits
        PBKDF:      argon2i
        Time cost:  5
        Memory:     1048576
        Threads:    4
        Salt:       81 0d d7 18 01 e4 1d
                    fc 8c 32 9a 4e 94 a0
        AF stripes: 4000
        AF hash:    sha256
        Area offset:32768 [bytes]
        Area length:258048 [bytes]
        Digest ID:  0
  1: luks2
        Key:        512 bits
        Priority:   normal
        Cipher:     aes-xts-plain64
        Cipher key: 512 bits
        PBKDF:      argon2i
        Time cost:  5
        Memory:     1048576
        Threads:    4
        Salt:       f9 df 21 39 44 07 d3
                    d1 28 2b 2c 48 5f 8f
        AF stripes: 4000
        AF hash:    sha256
        Area offset:290816 [bytes]
        Area length:258048 [bytes]
        Digest ID:  0
Tokens:
```

Entropy

Entropy is a measure for the “amount of randomness” of a random variable. It depends on the distribution of the random variable.

- A 128-bit string where each bit is independently and uniformly randomly selected has an entropy of 128 bits.
- A 128-bit string that is selected to be either $\texttt{00...0}$ or $\texttt{11...1}$ has an entropy of 1 bit.
- A password chosen uniformly at random from a list of 10 000 words has an entropy of $\log_2(10\,000) \approx 13.29$ bits.
- It is not possible to “measure” the entropy of a particular value, such as a specific password or key.



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Conclusion

Conclusion

- Symmetric schemes protect data confidentiality and/or integrity
- Their security builds on secure primitives by using a secure mode
- Confidentiality can be protected with
  - Encryption (⚠ no authenticity)
  - Authenticated Encryption
  - Asymmetric encryption, key encapsulation (next lecture)

Questions