Secret Key Cryptography

Cryptography (lecture portion)

Andreas Unterweger

School of ITS
Salzburg UAS

Winter term 2023/24

Recap

- Traditional naming of actors
  - Alice
  - Bob
  - Eve
- Terms
  - Plaintext (further: chosen-plaintext attack)
  - Ciphertext (further: chosen-ciphertext attack)
  - Key
  - Encryption (function)
  - Decryption (function)
- Types of ciphers
  - Symmetric vs. asymmetric cryptography
  - Stream vs. block ciphers

Overview of secret key cryptography

- Symmetric cryptography: Key is kept secret/private
- Perfect security (Vernam) requires keys as long as the message

$\rightarrow$ Impractical (shorter keys desired)

$\rightarrow$ Practical solution: Approaches which are not perfectly secure, but only computationally secure (weaker security guarantee)

- Computational security example: Cipher can be broken with a probability of $< 10^{-30}$ in 200 years using the fastest available computer
- Computationally secure schemes rely on (difficulty) assumptions

- Focus topics:
  - Example of a symmetric cipher: The Advanced Encryption Standard
  - Key exchange
  - Cryptographic hashes (for integrity)

Overview of AES

- The Advanced Encryption Standard (AES)
  - Standardized as winner of a competition (2001)
  - Widely used and supported
  - Block cipher (128 bits) with a substitution-permutation network
  - Supports 128-, 192- or 256-bit keys
  - Known attacks on simplified variations of AES, but no better way known to break full AES than exhaustive/brute-force key search

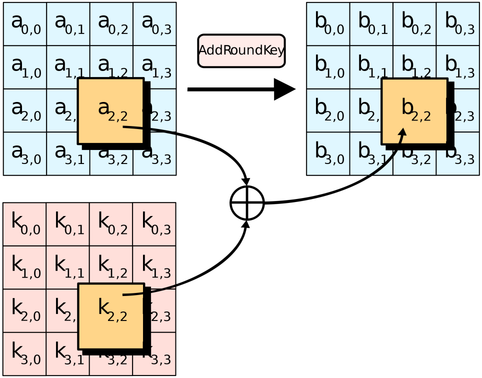
- Recall: $2^{128} \approx 10^{38}$ possible keys is **a lot**
  - Optimistic 10^{10} decryptions/s on a desktop computer: 10^{28} s $\approx 10^{20}$ a!
  - Optimistic 10^{20} decryptions/s on a computer cluster: 10^{18} s $\approx 10^{10}$ a!

The AES algorithm I

- High-level overview of encryption:
  - $4 \cdot 4$ byte state array initially equal to the input (block)
  - Multiple rounds which change the state array
  - Number of rounds depends on the key length (e.g., 14 for 256 bits)
  - Each round: 4 stages (substitution-permutation network)
  - Key influences state array in each round (without details)
  - Final round has slightly different stages (without details)
- Decryption inverts encryption steps (simplified)
- Messages which are not a multiple of the block size long need to be padded (without details)

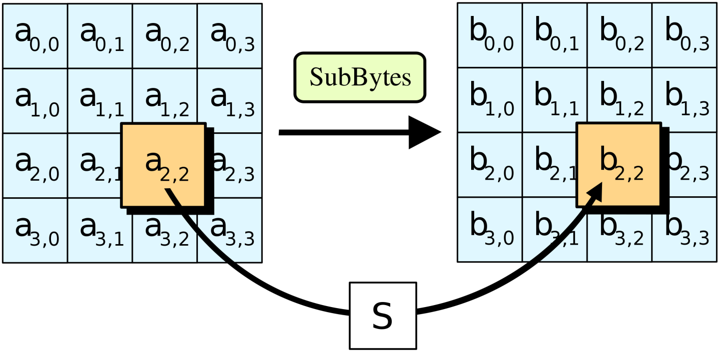
The AES algorithm II

- Stage 1: `AddRoundKey`
  - Derive 128-bit round key from key
  - XOR state array with round key



Source: Matt Crypto: AddRoundKey operation for AES. `https://commons.wikimedia.org/wiki/File:AES-AddRoundKey.svg#/media/File:AES-AddRoundKey.svg` (accessed on August 23, 2022), 2006.

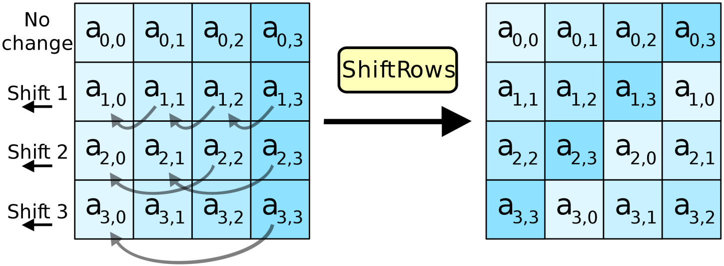
- Stage 2: `SubBytes`
  - Replace bytes based on a lookup table
  - Lookup table is fixed for all bytes and rounds



Source: Matt Crypto: SubBytes operation for AES. `https://commons.wikimedia.org/wiki/File:AES-SubBytes.svg#/media/File:AES-SubBytes.svg` (accessed on August 23, 2022), 2006.

The AES algorithm IV

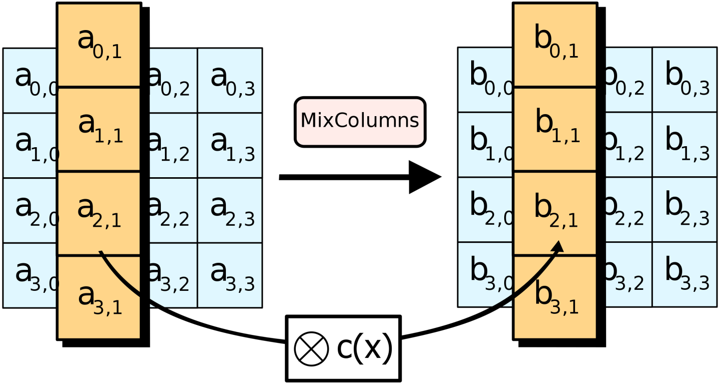
- Stage 3: `ShiftRows`
  - Cyclical shift of bytes in rows of state array
  - Different shift for each row



Source: Matt Crypto: ShiftRows operation for AES. `https://commons.wikimedia.org/wiki/File:AES-ShiftRows.svg#/media/File:AES-ShiftRows.svg` (accessed on August 23, 2022), 2006.

The AES algorithm V

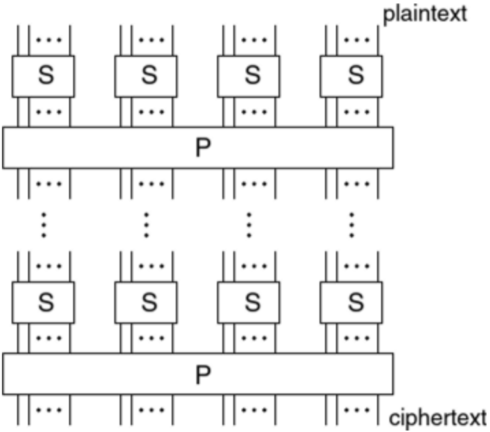
- Stage 4: `MixColumns`
  - Column-wise transformation (without details)
  - Achieves diffusion together with stage 3



Source: Matt Crypto: MixColumns operation for AES. `https://commons.wikimedia.org/wiki/File:AES-MixColumns.svg#/media/File:AES-MixColumns.svg` (accessed on August 23, 2022), 2006.

Plug-in: Substitution-permutation networks I

- Multiple rounds of invertible substitution and permutation
- Implement the confusion-diffusion paradigm



Source: Ebrary.net: DES. `https://ebrary.net/134519/computer_science/` (accessed on August 23, 2022), 2022.

Plug-in: Substitution-permutation networks II

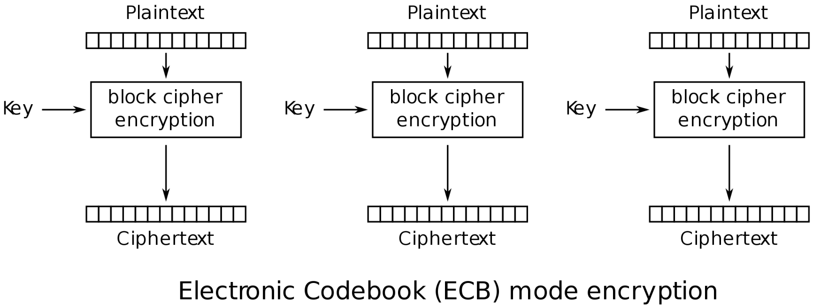
- Confusion-diffusion paradigm
  - Confusion: Permute all parts/bytes of a block separately
  - Diffusion: Reorder bits to propagate changes to all parts of the output
  - Repeated use of confusion and diffusion result in random-looking permutation overall (without details)

  → Avalanche effect: Small changes to the input result in large changes to the output (a single input bit should affect all output bits)

  → Changing one input bit is expected to flip 50% of the output bits
- Security depend on choices of substitutions and permutations as well as the number of rounds
- Networks/block ciphers by themselves are **not** secure against chosen-plaintext attacks (details in chapter 3 of Katz (2008))

Modes of operation: ECB

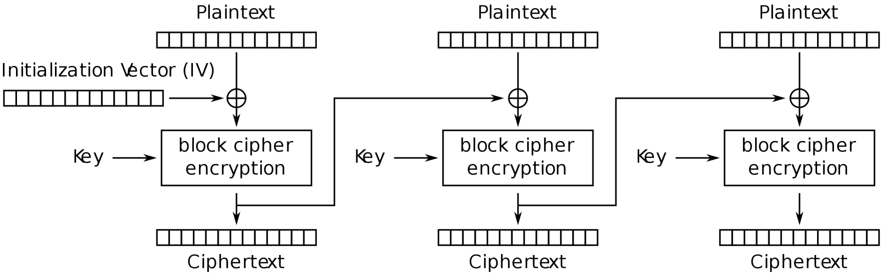
- ECB: Electronic code book mode
- Same plaintext always yields same ciphertext under same key → identical plaintext blocks repeat in ciphertext → **not** secure!



Source: WhiteTimberwolf: Encryption using the Electronic Code Block (ECB) mode. `https://commons.wikimedia.org/wiki/File:ECB_encryption.svg#/media/Datei:ECB_encryption.svg` (accessed on August 23, 2022), 2013.

Modes of operation: CBC I

- CBC: Cipher block chaining mode
- Ciphertexts from previous blocks affect ciphertext of current block
- Initialization vector (IV) must be chosen for first block

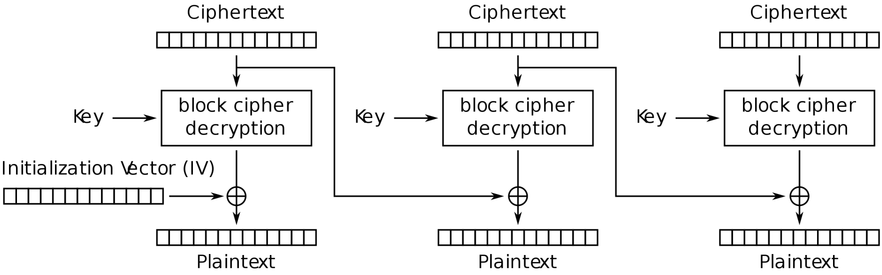


Cipher Block Chaining (CBC) mode encryption

Source: WhiteTimberwolf: Encryption using the Cipher Block Chaining (CBC) mode. `https://commons.wikimedia.org/wiki/File:CBC_encryption.svg#/media/File:CBC_encryption.svg` (accessed on August 23, 2022), 2013.

Modes of operation: CBC II

- IV should be random and not be reused
- IV must be sent with the ciphertext for decryption
- Encryption and decryption cannot be parallelized



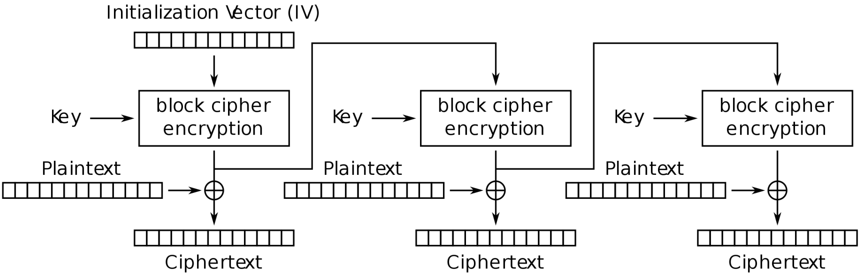
Cipher Block Chaining (CBC) mode decryption

Source: WhiteTimberwolf: Decryption using the Cipher Block Chaining (CBC) mode. https://commons.wikimedia.org/wiki/File:CBC_decryption.svg#/media/File:CBC_decryption.svg (accessed on August 23, 2022), 2013.

Modes of operation: OFB

- OFB: Output feedback mode
- Only IV is input repeatedly into the block cipher, not the plaintext

→ Plaintext is XOR-ed with the encrypted byte stream

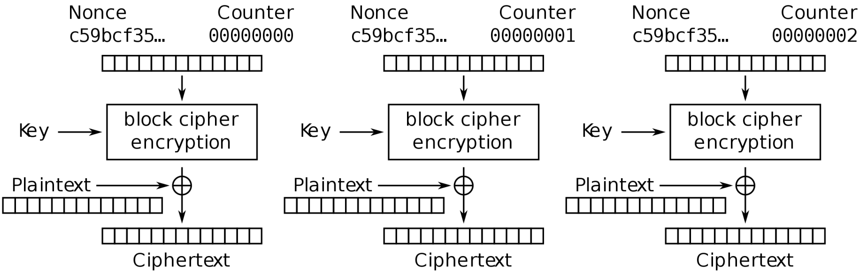


Output Feedback (OFB) mode encryption

Source: WhiteTimberwolf: Encryption using the Output Feedback (OFB) mode. https://commons.wikimedia.org/wiki/File:OFB_encryption.svg#/media/File:OFB_encryption.svg (accessed on August 23, 2022), 2013.

Modes of operation: CTR

- CTR: Counter mode (different variations)
- Like OFB, but with incrementing counter (starts at random number)
- Parallelizable (like OFB) and secure against chosen-plaintext attack



Counter (CTR) mode encryption

Source: WhiteTimberwolf: Encryption using the Counter (CTR) mode. `https://commons.wikimedia.org/wiki/File:CTR_encryption_2.svg#/media/File:CTR_encryption_2.svg` (accessed on August 23, 2022), 2013.

Practical concerns

- Which key length is sufficient?
  - Depends until when the ciphertext should be secure
  - Depends on who you ask (professional recommendations)
  - Comparison at `https://www.keylength.com/en/compare/`
- AES and the presented modes do not provide
  - Integrity/authentication: Eve can change bits of the ciphertext without Bob noticing $\rightarrow$ GCM (Galois/counter mode, without details) or additional integrity checks $\rightarrow$ cryptographic hashes
  - Security against chosen-ciphertext attacks without additional measures
  - Security if they are used improperly, e.g., when IVs are reused

Key exchange

- Secret key cryptography requires a pre-shared key

→ How do Alice and Bob share a secret key?
  - Physically (meeting, mailing etc.)
  - Key distribution centers (requires trust)

  … (other solutions which are impractical for transient communication)
- For multiple communicating parties
  - Each pair of communicating parties needs their own key

  → Quadratic complexity (impractical for storage and management)
- Alternative: Diffie-Hellman key exchange

Plug-in: Modular arithmetic I

- For all $a, b, N \in \mathbb{N}\backslash\{0, 1\}$, $a \equiv b \pmod{N}$ if $a \mod N = b \mod N$
- a and b are congruent modulo N when their remainders upon division by N are equal, e.g., $15 \equiv 3 \pmod{12}$; $123 \equiv 35 \equiv 2 \pmod{11}$

Source: Time Clock Experts.com: Pyramid 13" Analog STD 12/24-Hr Clock Battery Operated (For 915MHz). `https://www.timeclockexperts.com/Pyramid-13-915MHz-9A13D-Battery-Operated-p/s9a3acgbxb.htm` (accessed on August 25, 2022), 2006.

Plug-in: Modular arithmetic II

- Adding in a modulus is like adding on a clock – examples: $11 + 3 \equiv 14 \equiv 2 \pmod{12}$; $123 + 35 \equiv 158 \equiv 4 \pmod{11}$
- Most standard rules of arithmetic still work in modular arithmetic:
  - Addition: If $x \equiv x' \pmod{N}$ and $y \equiv y' \pmod{N}$, then $x + y \equiv x' + y' \pmod{N}$
  - Subtraction (analogously)
  - Multiplication: If $x \equiv x' \pmod{N}$ and $y \equiv y' \pmod{N}$, then $x \cdot y \equiv x' \cdot y' \pmod{N}$
  - Division does **not** work in general
- Invertibility (not always possible):
  - Define a^{-1} such that $a \cdot a^{-1} \equiv 1 \pmod{N}$
  - Example: $a = 3, a^{-1} = 4, N = 11$; counter-example: $a = 3, N = 12$
  - Requires that $\gcd(a, N) = 1$ (details in Katz (2008))

Plug-in: Multiplicative groups I

- A multiplicative group is a set S which
  - is closed under multiplication (multiplying yields an element of the set)
  - has an identity (element) e such that $\forall s \in S : e \cdot s = s$
  - has an inverse for every element
- Counter-examples:
  - $\{2\}$ does not fulfill any of the criteria
  - $\mathbb{R}$: Zero is not invertible
  - $\{1, j, -j\}$ is not closed (e.g., $j \cdot j = -1$)
- Simple example: $\mathbb{R}\backslash\{0\}$ is a multiplicative group
  - $\mathbb{R}\backslash\{0\}$ is closed
  - The identity element is 1
  - Every element has an inverse: $e^{-1} = \frac{1}{e}$
- Multiplications may also be performed in a modulus

Plug-in: Multiplicative groups II

Multiplication tables of the groups $\mathbb{Z}_5^* = \{1, 2, 3, 4\}$ modulo 5 (left) and $\{3, 6, 9, 12\}$ modulo 5 (right):

×	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

×	3	6	9	12
3	9	3	12	6
6	3	6	9	12
9	12	9	6	3
12	6	12	3	9

Sources: Purwanto, Hidayah, I. N., and Hasanah, D.: Results and Problems on Constructing Multiplicative Groups in Modular Arithmetic. `http://fmipa.um.ac.id/wp-content/uploads/2019/10/MATEMATIKA_PURWANTO-Rev-14-23.pdf` (accessed on August 25, 2022), 2019.

[1] Purwanto, Hidayah, I. N., and Hasanah, D.: Results and Problems on Constructing Multiplicative Groups in Modular Arithmetic. `http://fmipa.um.ac.id/wp-content/uploads/2019/10/MATEMATIKA_PURWANTO-Rev-14-23.pdf` (accessed on August 25, 2022), 2019.

Plug-in: Multiplicative groups III

- Exponentiation for groups (analogous to "regular" exponentiation)
  - Defined as repeated multiplication: $g^m := \underbrace{g \cdot g \cdot \cdots \cdot g}_{m \text{ times}}$
  - $g^0 := 1$ (where 1 needs to be an element of the group)
  - Familiar rules apply, e.g., $(g^a)^b = g^{a \cdot b}$
- Special groups $\mathbb{Z}_n^* := \{1, 2, \cdots, n-1\}$ with multiplication modulo $n \in \mathbb{P}$ (restricted for now, general definition later)
  - n **must** be prime (otherwise some elements are not invertible)
  - Exponentiating to base $g \in \mathbb{Z}_n^*$ (cyclically) generates subsets/subgroups
  - Example: $g = 3$ for $\mathbb{Z}_5^*$ generates $\{1, 3, 4, 2\} = \mathbb{Z}_n^*$
  - Smaller example: $g = 4$ for $\mathbb{Z}_5^*$ generates the subgroup $\{1, 4\}$

Note: For proofs of the above claims see Katz (2021)

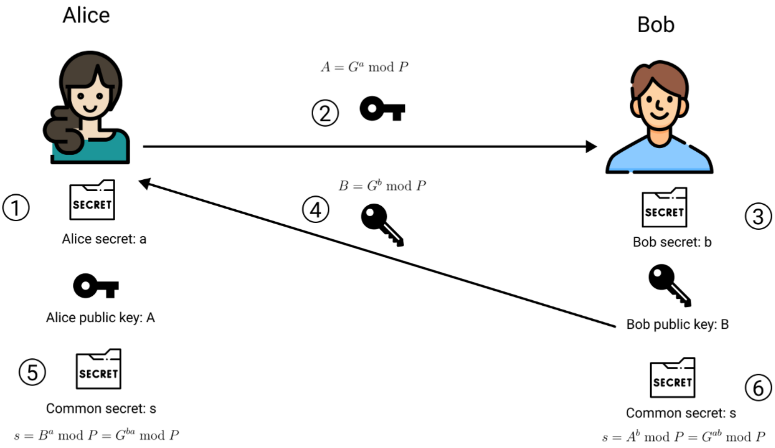
Diffie-Hellman key exchange I

- Discrete logarithm (analogous to "regular" logarithm)
  - Inverse operation of exponentiation in a modulus
  - Definition: $log_g(h) = x$ if $g^x \equiv h \mod n$ in $\mathbb{Z}_n^*$
  - Example: $log_3(4) \equiv 2 \mod 5$
  - For some g in cyclic groups, log_g is easy to compute
  - For some g, log_g is **believed** to be hard to compute

- Diffie-Hellman assumption (computational Diffie-Hellman problem):
  - Task: Given $X := g^x$ and $Y := g^y$ with known generator g and modulus n, determine $g^{x \cdot y} = (g^x)^y = (g^y)^x$
  - If log_g is easy to compute for g in $\mathbb{Z}_n^*$, $g^{x \cdot y}$ is easy to compute as $X^{log_g(Y)} = (g^x)^y = g^{x \cdot y}$
  - **If** log_g is hard to compute for g in $\mathbb{Z}_n^*$, $g^{x \cdot y}$ is hard to compute

→ Use this assumption to build a key exchange protocol

Diffie-Hellman key exchange II



Source: Kosolov, P.: Diffie-Hellman Key Exchange via REST. `https://medium.com/@razumovsky_r/diffie-hellman-key-exchange-via-rest-b7a91c9df7b1` (accessed on August 25, 2022), 2022.

Diffie-Hellman key exchange III

- Steps (G and P are assumed to be publicly known):
  1. Alice generates a random group element a (out of scope)
  2. Alice sends $A = G^a \mod P$ to Bob
  3. Bob generates a random group element b (out of scope)
  4. Bob sends $B = G^b \mod P$ to Alice
  5. Alice computes the shared secret/key $s = B^a = (G^b)^a = G^{a \cdot b}$
  6. Bob computes the shared secret/key $s = A^b = (G^a)^b = G^{a \cdot b}$
- Eve can see $A = G^a$ and $B = G^b$, but cannot compute $s = G^{a \cdot b}$ from this information alone (Diffie-Hellman assumption)
- There may be other ways to compute $s \rightarrow$ stronger security definition through decisional Diffie-Hellman problem (out of scope)
- A man in the middle could still intercept communication, so additional protections are needed (without details)

Practical concerns

- Forward secrecy
  - If an attacker finds the shared secret somehow, he can only read the current, but not previous or future conversations between Alice and Bob
  - Requires that Alice and Bob generate/exchange new keys every time they communicate (also called Diffie-Hellman Ephemeral)
- How to choose good generators and groups
  - For $\mathbb{Z}^*_{n \in \mathbb{P}}$, the discrete logarithm is easy to compute in some cases
  - → Use "safe" primes and certain large subgroups (out of scope)
  - Additional checks for vulnerable g and n (out of scope)
- How to choose the size of the modulus n
  - Symmetric key sizes **cannot** be compared to (sub)group sizes
  - Recommendations at `https://www.keylength.com/en/compare/`

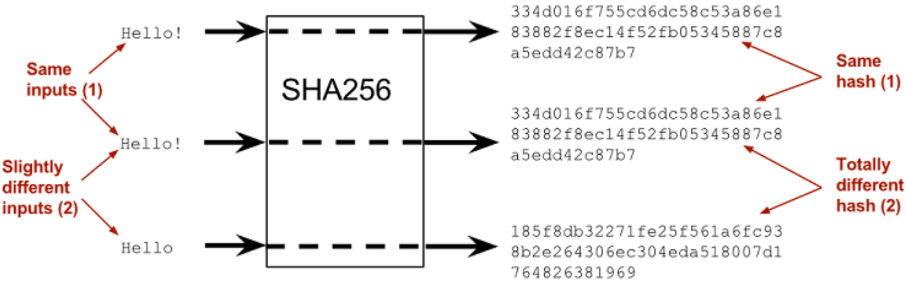
[2] IBM Corporation: Variants of Diffie-Hellman.
`https://www.ibm.com/docs/en/zvse/6.2?topic=SSB27H_6.2.0/fa2ti_openssl_variants_of_diffie_hellman.html`
(accessed on August 25, 2022), 2021.

Overview of cryptographic hashes I

- A hash function $H(m)$
  - creates a fingerprint/digest/hash of a given message m
  - maps a variable-sized input to a fixed-sized output
  - is "one-way", i.e., computationally infeasible to invert
- Applications
  - Message integrity: Detect tampering
  - Digital signatures (later): Sign hash instead of full message
  - Pseudo-random number generation (out of scope)
- Example hash function: Secure Hash Algorithm 2

Overview of cryptographic hashes II

- The same input always gives the same output, i.e., $H(m_1) = H(m_2)$ if $m_1 = m_2$
- Different inputs ideally give different outputs (more later)



Source: Manning Publications: Cryptographic Hashes and Bitcoin.
`https://freecontent.manning.com/cryptographic-hashes-and-bitcoin/` (accessed on August 23, 2022), 2017.

One-way functions

- A one-way function
  - is easy to compute
  - is hard to invert
  - relies on the existence of a problem which is easy to compute one way, but hard to compute the other way around
- Example: Factoring as a one-way function
  - Given arbitrary and large $p, q \in \mathbb{P}$, multiply and output $N = p \cdot q$
  - Inverse problem: given an arbitrary and large N which is the product of two primes, factor p and q such that $p \cdot q = N$
  - **Assumes** the factoring assumption is true (multiplying is computationally "easy", but factoring is computationally "hard")
- One-way functions alone are insufficient to build robust hash functions
- Practical hash functions do not base their security on provable reduction to one-way functions, but on heuristics

Secure Hash Algorithm 2 I

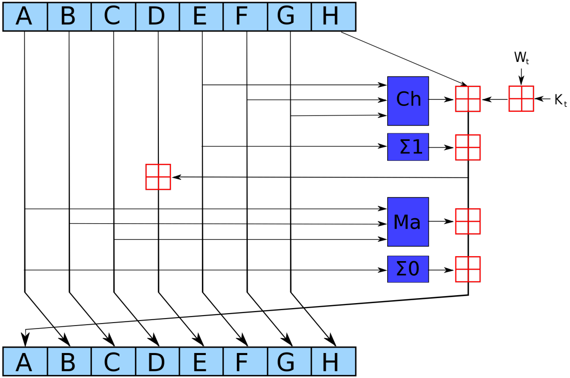
- Secure Hash Algorithm 2 (SHA-2)
  - Standardized (2001) after SHA-1 weaknesses
  - Widely used and supported
  - Supports 224-, 256-, 384- and 512-bit digests (for recommended digest sizes see `https://www.keylength.com/en/compare/`)
  - Known attacks on simplified variations of SHA-2, but no better way known to break full SHA-2 than generic attacks affecting all cryptographic hashes (later)

- Example: SHA-512 (SHA-2 with 512-bit digests)
  - 80 rounds of updating 32-bit internal state variables `A-H` with 64-bit words W based on the input message (simplified)
  - Automatic padding for messages whose length is not an integer multiple of 64 bits

[3] Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In: FSE 2012: Fast Software Encryption, 2012.

Secure Hash Algorithm 2 II

- Note: The red plus denotes 64-bit addition, the blue function blocks perform logical and bit-shift operations detailed in the image source



Source: kockmeyer: A schematic that shows the SHA-2 algorithm. https://commons.wikimedia.org/wiki/File:SHA-2.svg#/media/File:SHA-2.svg (accessed on August 23, 2022), 2007.

[3] Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In: FSE 2012: Fast Software Encryption, 2012.

Attacks on cryptographic hashes – Introduction

- Basic definitions:
  - Collision: Two inputs (messages) which give the same output (hash): $H(m_1) = H(m_2)$ if $m_1 \neq m_2$
  - Collision resistance: It is computationally infeasible to find collisions
- Collisions **must** happen when allowing arbitrarily-sized inputs with a fixed-sized output (pigeon-hole principle)

$\rightarrow$ Weaker security levels in practice:
  - Second-preimage resistance: Given a message m, it is infeasible to find an $m' \neq m$ such that $H(m') = H(m)$
  - Preimage resistance: Given a hash $h = H(m)$, it is infeasible to find an m' such that $H(m') = h$

Attacks on cryptographic hashes – Overview

- Brute-force attack ((second-)preimage attack)
  - Try to find the preimage m or another m' such that $H(m) = H(m') =: h$
  - Generate messages $m_1, m_2, \ldots$
  - For n-bit hashes, the probability of any m_i being hashed to h is $\frac{1}{2^n}$

  $\rightarrow$ 2^n attempts necessary
- Birthday attack (collision attack)
  - Try to find two preimages $m \neq m'$ such that $H(m) = H(m')$
  - Generate **distinct** messages $m_1, m_2, \ldots$ **uniformly at random**
  - For n-bit hashes, the probability of any two m_i and m_j $(i \neq j)$ being hashed to the same output is determined by the Birthday problem

  $\rightarrow$ $\approx 2^{\frac{n}{2}}$ attempts necessary
- Length extensions (out of scope)
- Partial-message attack (out of scope)

...

Thank you for your attention!

Questions?