

Applied Cryptography

thgoebel@ethz.ch

ETH Zürich, FS 2021

This document is a **short** summary for the course *Applied Cryptography* at ETH Zurich. It is intended as a document for quick lookup, e.g. during revision, and as such does not replace reading the slides or a proper book.

We do not guarantee correctness or completeness, nor is this document endorsed by the lecturers. Feel free to point out any erratas.

Contents

| | | |
|----------|---|----------|
| 1 | Symmetric Cryptography | 3 |
| 1.1 | Block Ciphers | 3 |
| 1.2 | Symmetric Encryption | 5 |
| 1.3 | Attacks | 6 |
| 1.4 | Hash Functions | 7 |
| 1.5 | Message Authentication Codes MACs | 9 |
| 1.6 | Authenticated Encryption | 14 |

List of Figures

| | | |
|----|--|----|
| 1 | PRP game | 3 |
| 2 | ECB mode | 4 |
| 3 | CBC mode (left: encipher, right: decipher) | 4 |
| 4 | CTR mode | 5 |
| 5 | IND-CPA game | 5 |
| 6 | Padding oracle attack on CBC mode | 7 |
| 7 | Attack on predictable IVs in CBC mode | 7 |
| 8 | Merkle-Damgård transform | 8 |
| 9 | Davies-Meyer construction | 9 |
| 10 | Sponge construction: absorbing | 9 |
| 11 | Sponge construction: squeezing | 9 |
| 12 | MAC game | 10 |
| 13 | HMAC | 12 |
| 14 | Carter-Wegman MAC tag algorithm | 13 |
| 15 | INT-CTXT + INT-PTXT game | 14 |
| 16 | IND-CCA game | 15 |

Credits: images are generally taken from the lecture slides.

1 Symmetric Cryptography

One-time pad Plaintext p , key k such that $|p| = |k|$. Ciphertext $c = p \oplus k$.

If k u.a.r. and only used once then the OTP is **perfectly secure**, i.e. $\Pr[P = p | C = c] = \Pr[P = p]$.

Note: keys can re-occur (as a result of random sampling) but they must not be re-used (i.e. the adversary must not be aware that the same key is used).

Issues: same lengths, key distribution, single use.

1.1 Block Ciphers

Block cipher A block cipher with key length k and block size n consists of two efficiently computable permutations¹:

$$E : \{0, 1\}^k \times \{0, 1\}^n \mapsto \{0, 1\}^n \quad D : \{0, 1\}^k \times \{0, 1\}^n \mapsto \{0, 1\}^n$$

such that for all keys K D_K is the inverse of E_K (where we write E_K short for $E(K, \cdot)$).

Security notions Known plaintext attack, chosen plaintext attack, chosen ciphertext attack. Exhaustive key search on (P, C) pairs – no attack should be better, else we throw the cipher away.

Pseudo-randomness

- Adversary \mathcal{A} interacts either with block cipher (E_K, D_K) or a truly random permutation (Π, Π^{-1}) .
- A block cipher is called a **pseudo-random permutation PRP** if no efficient² \mathcal{A} can tell the difference between E_K and Π (no access to the inverse).
- A block cipher is called a **strong-PRP** if no efficient \mathcal{A} can tell the difference between (E_K, D_K) and (Π, Π^{-1}) .

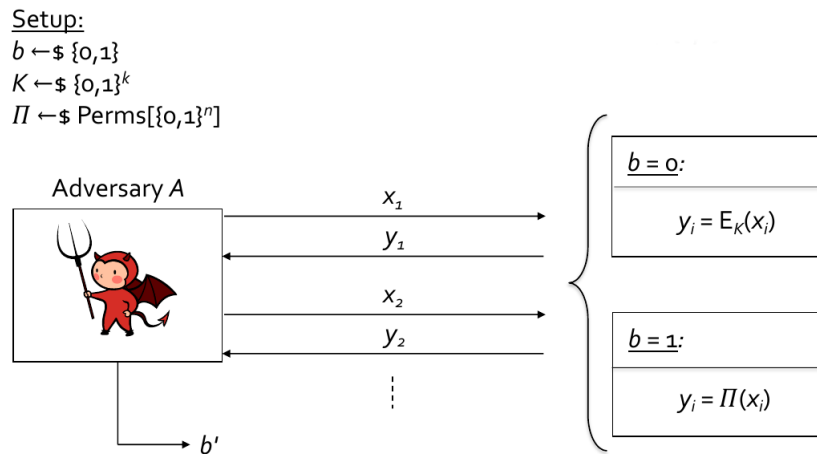


Figure 1: PRP game

¹Encipher and decipher

²Quantified by runtime + number of oracle queries.

The advantage is defined as:

$$\mathbf{Adv}_E^{PRP}(\mathcal{A}) = 2 \cdot \left| \Pr[\text{Game } \mathbf{PRP}(\mathcal{A}, E) \Rightarrow \text{true}] - \frac{1}{2} \right|$$

where the probability is over the randomness of b, K, Π, \mathcal{A} . Note that:

$$\Pr[\text{Game } \mathbf{PRP}(\mathcal{A}, E) \Rightarrow \text{true}] = \Pr[b' = b]$$

Also see the Advantage Rewriting Lemma (1.2).

Constructing block ciphers In general: keyed round function that is repeated many times.

- Feistel cipher: halved blocks crossing back and forth, e.g. DES
- Substitution-permutation network: confusion + diffusion, e.g. AES

Electronic Code Book (ECB) mode Same plaintext always maps to the same ciphertext (deterministic). Thus serious leakage, don't use.

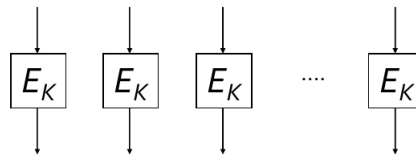


Figure 2: ECB mode

Cipher Block Chaining (CBC) mode Use u.a.r. IV/previous ciphertext block to randomise encryption.

A bit flip in C_i completely scrambles/randomises P_i and flips the same bit in P_{i+1} .

Caveats: non-random IV, padding oracle attack, ciphertext block collisions (after using the same key for $2^{n/2}$ blocks by the birthday bound).

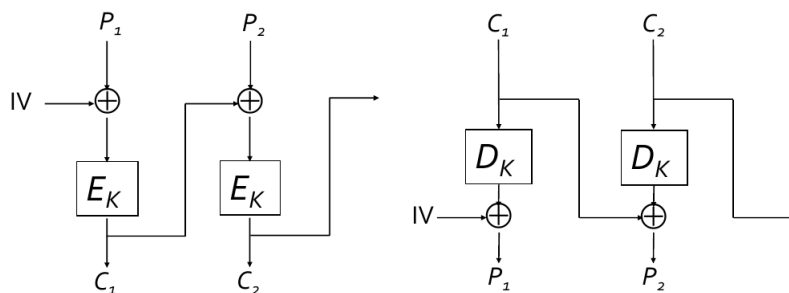


Figure 3: CBC mode (left: encipher, right: decipher)

Counter (CTR) mode Incrementing counter is encrypted with block cipher to produce a pseudo-random value to xor the plaintext block with.

Effectively a stream cipher producing OTP keys. E_K does not even need to be invertible. No padding needed, can just truncate the last block. A bit flip in C_i flips the same bit in P_i .

Caveats: counter must not repeat/wrap around (else xor of ciphertexts = xor of plaintexts).

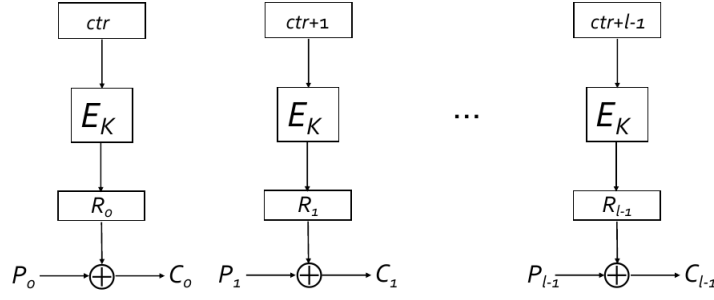


Figure 4: CTR mode

1.2 Symmetric Encryption

Symmetric Encryption Scheme is a triple $SE = (KGen, Enc, Dec)$. We have key space $\mathcal{K} = \{0,1\}^k$, message space $\mathcal{M} = \{0,1\}^{*3}$ and ciphertext space $\mathcal{C} = \{0,1\}^*$. For correctness, we have $Dec_K(Enc_K(m)) = m$.

IND-CPA Security Informally: computational version of perfect security – an efficient adversary cannot compute anything useful from a ciphertext (e.g. hide every bit of the plaintext). Equivalent to *semantic security*.

Formally: For any efficient adversary \mathcal{A} , given the encryption of one of two equal-length messages of its choice, \mathcal{A} is unable to distinguish which one of the two messages was encrypted.

In the security game, \mathcal{A} gets access to a *Left-or-Right encryption oracle*. The advantage of \mathcal{A} is:

$$\text{Adv}_{SE}^{\text{IND-CPA}}(\mathcal{A}) = 2 \cdot \left| \Pr[\text{Game IND-CPA}(\mathcal{A}, SE) \Rightarrow \text{true}] - \frac{1}{2} \right|$$

Notes: Deterministic schemes **cannot** be IND-CPA secure (why?⁴). CBC and CTR mode (if used properly) can be proven to be IND-CPA secure (assuming that Enc is a PRP-secure block cipher).

Caveats: No integrity. Says nothing about messages of non-equal length. No chosen ciphertexts.

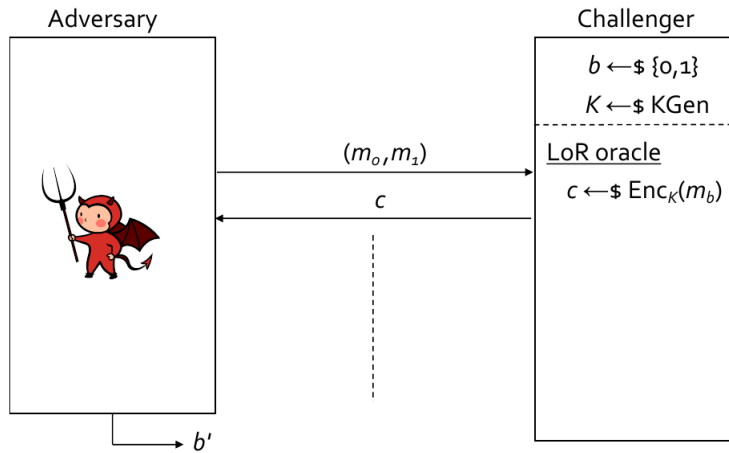


Figure 5: IND-CPA game

³In reality we might have a maximum plaintext length.

⁴The adversary can first query (m_0, m_0) to learn c_0 , then query (m_0, m_1) and thus win with high probability.

Advantage Rewriting Lemma Let b be a uniformly random bit and b' the output of some algorithm. Then:

$$\begin{aligned} 2 \left| \Pr[b' = b] - \frac{1}{2} \right| &= \left| \Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0] \right| \\ &= \left| \Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1] \right| \end{aligned}$$

Difference Lemma Let Z, W_1, W_2 be events. If

$$(W_1 \wedge \neg Z) \text{ occurs if and only if } (W_2 \wedge \neg Z) \text{ occurs}$$

then

$$\left| \Pr[W_2] - \Pr[W_1] \right| \leq \Pr[Z]$$

In practice: Z is a bad event that rarely happens, W_1, W_2 are when \mathcal{A} wins in security games G_1, G_2 . Useful for *game hopping* proofs.

PRP-PRF Switching Lemma Let E be a block cipher. Then for any algorithm \mathcal{A} making q queries:

$$\left| \text{Adv}_E^{\text{PRP}}(\mathcal{A}) - \text{Adv}_E^{\text{PRF}}(\mathcal{A}) \right| \leq \frac{q^2}{2^{n+1}}$$

1.3 Attacks

Padding Added before encryption to expand plaintext to a multiple of the block size, i.e.

$\text{pad}(\cdot) : \{0, 1\}^* \mapsto \{\{0, 1\}^n\}^*$. Must be expanding (why?) and efficiently computable. May be either randomised or deterministic.

E.g. TLS padding: appends $t + 1$ bytes of value t .

Problem: adversary can detect padding errors (explicit error messages, logs, timing differences).

Padding Oracle Given a ciphertext C , returns whether the padding of the decryption is valid or invalid. Leaks 1 bit of information. Kind of a chosen ciphertext attack, thus not covered by IND-CPA security! Main problem: no ciphertext integrity.

In practice: \mathcal{A} needs 128 oracle queries on average to recover one plaintext byte. Repeat for all bytes and all blocks for full plaintext recovery. Additional practical details to consider. For TLS attacks, see Lucky 13 and POODLE.

Predictable IVs Random IVs are not just theoretically relevant for IND-CPA security of CBC mode. Consider the following steps (see Figure 7):

1. \mathcal{A} queries LoR oracle with (P_0, P_1)
2. Oracle responds with $C = C_0 || C_1$ where $C_1 = E_K(P_b \oplus C_0)$
3. \mathcal{A} predicts next $IV = C'_0$
4. \mathcal{A} queries $P_0 \oplus C_0 \oplus C'_0$
5. $\implies b = 0 \iff P_b = P_0 \iff C_1 = C'_1 \implies$ breaks IND-CPA security

In practice: systems may use *IV chaining* (use the last ciphertext as the next IV, in order to avoid having to sample new randomness). E.g.: SSLv3, TLS 1.0, SSH in CBC mode. See also the BEAST attack on TLS.

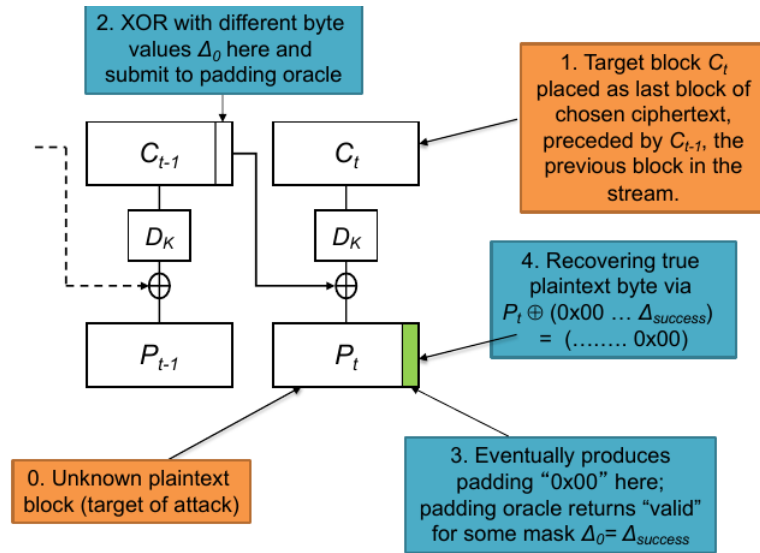


Figure 6: Padding oracle attack on CBC mode

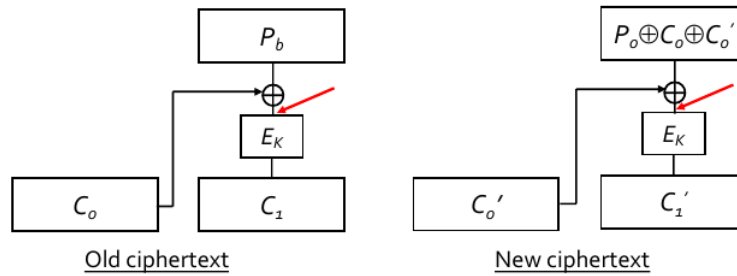


Figure 7: Attack on predictable IVs in CBC mode

1.4 Hash Functions

Cryptographic Hash Function An efficiently computable function $H : \{0, 1\}^* \mapsto \{0, 1\}^n$ that maps arbitrary-length inputs to fixed-length outputs ("message digests"). Not keyed!

Applications: MACs, signatures, key derivation, commitments.

Random oracle model Model under which hash functions are assumed to produce outputs that are uniformly random distributed. I.e. a hash function could be modelled by a random oracle. Very strong assumption!

Birthday paradox When drawing elements at random from a set of size s then after \sqrt{s} trials we expect a collision with 39% probability. We quickly get to 99% with an additional constant factor.

Security goals Primary goals:

- *Pre-image resistance* (one-wayness): Given h , it is infeasible to find an m such that $H(m) = h$.
- *Second pre-image resistance*: Given m_1 , it is infeasible to find an m_2 such that $H(m_1) = H(m_2)$.
- *Collision resistance*: It is infeasible to find any $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$.

Secondary goals:

- *Near-collision resistance*: It is infeasible to find any $m_1 \neq m_2$ such that $H(m_1) \approx H(m_2)$ (e.g. hashes agree on most bits).
- *Partial pre-image resistance 1*: Given $H(m)$, it is infeasible to recover any partial information about m .
- *Partial pre-image resistance 2*: Given a target string t with $|t| = l$ it is infeasible to find an m such $H(m) = t||x$ (faster than with 2^l brute-force hash evaluations).

CR adversary:

Cannot quantify security over all efficient \mathcal{A} (collisions exist, \mathcal{A} can hardcode one). Instead, define “ (t, ε) -CR adversaries”, running in time t and with $\mathbf{Adv}_H^{CR}(\mathcal{A}) = \varepsilon$. Build reductions from there.

Relationships:

- Collision resistance \implies second pre-image resistance
- Maybe: Collision resistance \implies pre-image resistance – depending on how you define pre-image resistance (sampling from the domain or from the range?)

Generic attacks (in the ROM):

(Second) pre-image resistance: 2^n evaluations.

Collision resistance: $2^{n/2}$ evaluations! (by the birthday paradox)

\implies e.g. SHA-1 with 160-bit outputs only achieves 80-bit security

Merkle-Damgård iterated hashing Constructs a hash function H from a *compression function* $h : \{0, 1\}^k \times \{0, 1\}^n \mapsto \{0, 1\}^n$. Used e.g. in MD5, SHA-1, SHA-2.

Steps: pad message, split into k -bit chunks, repeatedly apply h to the chunks and IV/chaining values (see Figure 8).

Padding scheme:

$$m' = \text{pad}(m) = m || 10^t || [\text{len}(m)]_k$$

where $[\text{len}(m)]_k$ is the k -bit encoding of the message length⁵ and $0 \leq t < k$ is minimal.

Theorem: If h is collision resistant, then so is H .⁶

Length extension attack: Given $H(m)$ (but not m !) once can compute a valid hash $y = H(\text{pad}(m) || m'')$. This is problematic e.g. when construction MACs or KDFs from a hash function.

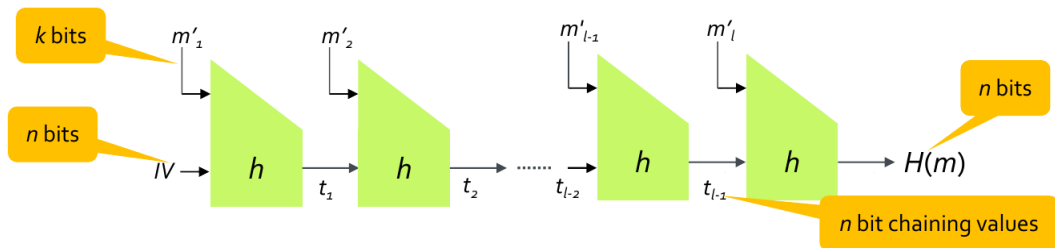


Figure 8: Merkle-Damgård transform

⁵This limits the maximum length of a message that can be hashed to $2^k - 1$.

⁶Note that the choice of padding scheme matters for the proof! Also note that the two IVs must be the same for a *full collision*. The much weaker form of two colliding messages for different IVs is called a *freestart collision*.

Constructing compression functions from block ciphers E.g. Davies-Meyer, Matyas-Meyer-Oseas, Miyaguchi-Preneel constructions. Use message as the key (need fast rekeying!) and (some variation of) chaining value as “plaintext” input.

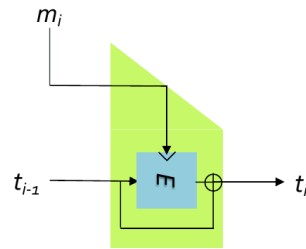


Figure 9: Davies-Meyer construction

Sponge construction Different design approach. Centered around 2 phases: absorbing + squeezing. Key ideas: giant bit permutation F , not inputting/outputting entire state. Variable length output.

E.g. used in SHA-3/Keccak: $\text{SHA-3}(m) = \text{out}_1 || \text{out}_2 || \text{out}_3 || \dots$

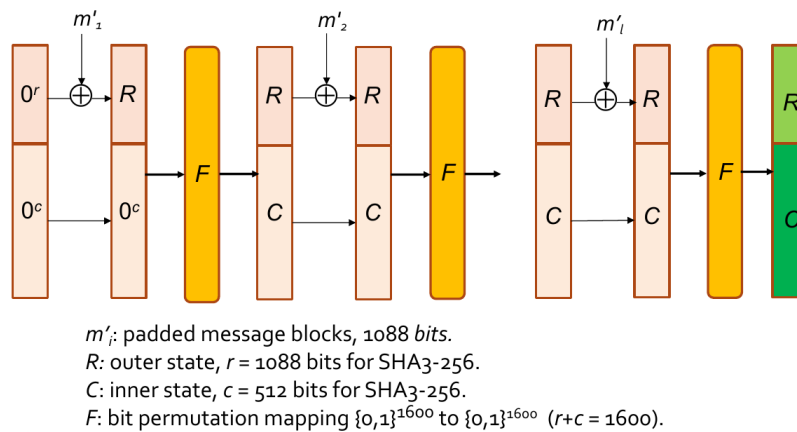


Figure 10: Sponge construction: absorbing

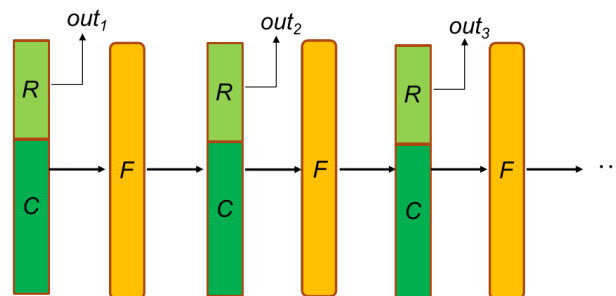


Figure 11: Sponge construction: squeezing

1.5 Message Authentication Codes MACs

Motivation Provide *integrity* and *data origin authentication*, i.e. no attacker should be able to modify or forge messages (unforgeability).

They do not provide confidentiality or secure against message deletion, replay, reordering, or reflection (if the key is used in both directions).

Definition (MAC scheme) A MAC scheme with key length k and tag length t consists of three efficient algorithms:

$$\begin{aligned} KGen &: \{\} \mapsto \{0, 1\}^k \\ Tag &: \{0, 1\}^k \times \{0, 1\}^* \mapsto \{0, 1\}^t \\ Vfy &: \{0, 1\}^k \times \{0, 1\}^* \times \{0, 1\}^t \mapsto \{0, 1\} \end{aligned}$$

For correctness⁷, we require that

$$\forall K, m. Vfy(K, M, \tau) = 1 \text{ where } \tau = Tag(K, m)$$

Vfy and Tag may be randomised. Note that if they are deterministic and Vfy internally (re)computes $Tag(K, m) \stackrel{?}{=} \tau$, then the scheme has *unique tags*.

MAC Security Informally: it is hard for an attacker to forge a valid message-tag pair (m', τ') .

Security game: \mathcal{A} gets access to a tag oracle and a verify oracle.

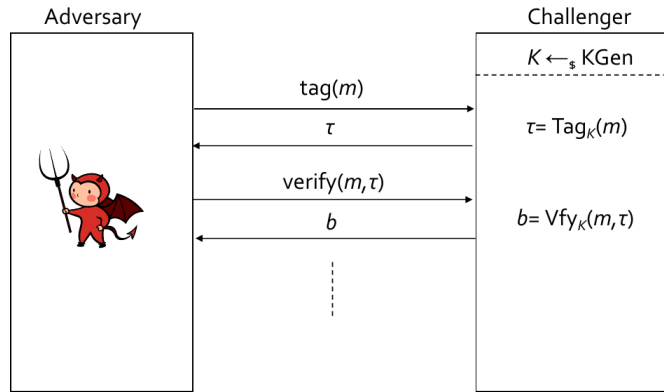


Figure 12: MAC game

Weak unforgeability under chosen message attack (WUF-CMA) \mathcal{A} wins if it submitted some (m^*, τ^*) to the verify oracle such that $Vfy(m^*, \tau^*) = 1$ and no query $Tag(m^*)$ was made.

Strong unforgeability under chosen message attack (SUF-CMA) \mathcal{A} wins if it submitted some (m^*, τ^*) to the verify oracle such that $Vfy(m^*, \tau^*) = 1$ and no query $Tag(m^*)$ with response τ^* was made.

Note that:

- A SUF-CMA adversary can also win by coming up with a new tag for an old message. We make the adversary “stronger” by relaxing the winning condition.
- $SUF-CMA \implies WUF-CMA$ (somewhat counter-intuitively)

⁷A functionality, not a security requirement.

- For deterministic MAC schemes that build Vfy from Tag , the two are equal (why?⁸).
- More formally: $(q_t, q_v, t, \varepsilon)$ -W/SUF-CMA security

Generic attacks: guess (m, τ) pairs, guess the key, few tag queries + exhaustive key search.

MACs from PRFs Construction of $MAC(F)$: Let $F : \{0, 1\}^k \times \{0, 1\}^n \mapsto \{0, 1\}^t$ be a PRF.

$$\begin{aligned} KGen() : & K \leftarrow \$ \{0, 1\}^k \\ Tag(K, m) : & \tau \leftarrow F(K, m) \\ Vfy(K, m, \tau) : & \tau' \leftarrow F(K, m); \text{ return } \tau' == \tau \end{aligned}$$

Theorem: If F is a PRF then $MAC(F)$ is a SUF-CMA.

Domain extension with CR hashing Motivation: build a MAC for a larger input domain \mathcal{X}' from a MAC with fixed domain \mathcal{X} (to handle more than just block-sized messages).

Construction of $HtMAC$: Let $MAC = (KGen, Tag, Vfy)$ be a MAC and let $H : \mathcal{X}' \mapsto \mathcal{X}$ be a hash function.

$$\begin{aligned} Tag'(K, m) : & Tag(K, H(m)) \\ Vfy'(K, m, \tau) : & Vfy(K, H(m), \tau) \end{aligned}$$

Theorem: If MAC is SUF-CMA secure and H is collision resistant then $HtMAC$ is SUF-CMA.

Hash-based MAC HMAC Turn (the often fast) hash functions into a keyed primitive using a two-key nest to build a PRF (and thus a MAC):

$$F_{nest}((K_1, K_2), m) = H(K_2 || H(K_1 || m))$$

Notes: Cannot simply prepend the key (length extension attacks). Also cannot simply append the key (offline collision attacks). In practice, derive two keys from one key using an inner and outer mask. Implementations often use an initialise-update-finalise interface (which can open timing attack vectors).

HMAC is gradually supplanted by faster designs based on universal hashing (see below), yet it is still used in key derivation scenarios (that use PRFness).

Nonce A number used once. Does neither need to be secret, nor does it need to be unpredictable. But must never ever repeat. Reasoning: it is easier to only use a number once than it is to get a good source of randomness.

Nonce-based MAC NMAC Like normal MAC, but Tag and Vfy now also take a nonce $N \in \mathcal{N}$. In the security game, we require that nonces are distinct (e.g. enforced by the tag oracle).

⁸When tags are unique then no adversary can come up with another tag for a previously queried message. However, consider the following deterministic scheme that is WUF-CMA but not SUF-CMA:

$$Tag'(K, m) = 0 || Tag(K, m) \quad Vfy'(K, m, b || \tau) = Vfy(K, m, \tau)$$

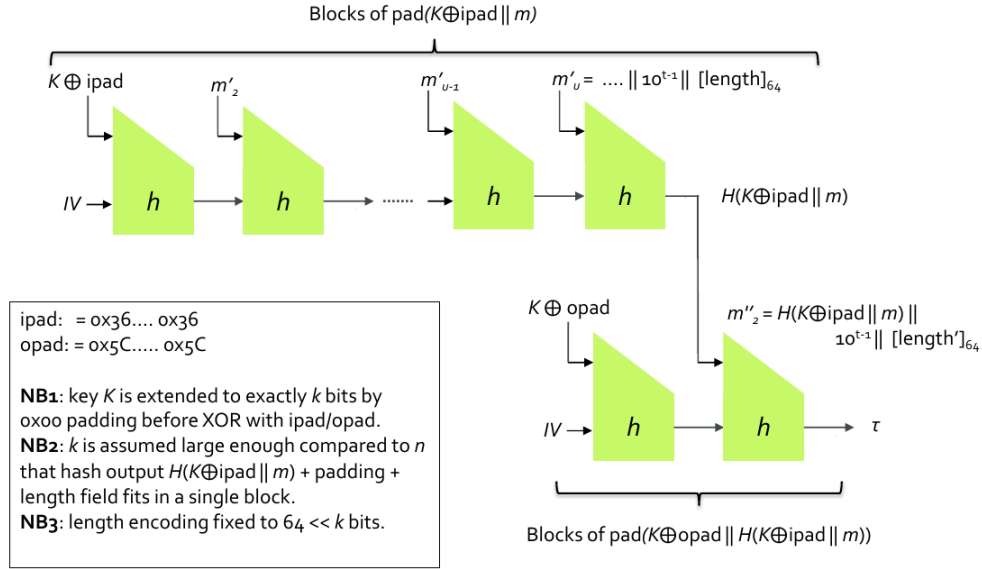


Figure 13: HMAC

Keyed hash function A keyed hash function H is a deterministic algorithm that takes a key and a message and produces a *digest* $t = H(K, m)$ over the digest space \mathcal{T} .

Universal hashing UHF Informally: selecting a hash function at random from a family of hash functions (the universal family).

Security game: Let H be a keyed hash function. Then:

1. Challenger sets $K \leftarrow \$ \{0, 1\}^k$
2. Adversary outputs distinct (m_0, m_1)

\mathcal{A} wins if $H(K, m_0) = H(K, m_1)$ and has advantage $\text{Adv}_H^{\text{UHF}}(\mathcal{A})$.

H is an ε -bounded universal hash function if $\text{Adv}_H^{\text{UHF}}(\mathcal{A}) \leq \varepsilon$ for all \mathcal{A} (even unbounded ones).

Equivalent definition: H is an ε -UHF if $\forall m_0, m_1 . \Pr[H(K, m_0) = H(K, m_1)] \leq \varepsilon$, where the probability is over the random choice of K .

Universal hash functions from polynomials Let \mathcal{F} be a finite field (e.g. integers mod p or $GF(2^n)$) and let $\mathcal{K} = \mathcal{T} = \mathcal{F}$ and $\mathcal{M} = \mathcal{F}^{\leq l}$. Then we define:

$$H_{\text{poly}}(K, (a_1, \dots, a_v)) = K^v + a_1 K^{v-1} + a_2 K^{v-2} + \dots + a_{v-1} K + a_v$$

That is, H_{poly} is always a degree v polynomial (even if the message vector a is zero). It can efficiently be evaluated using finite field operations and Horner's rule.

Theorem: H_{poly} is an ε -UHF for $\varepsilon = \frac{l}{|\mathcal{F}|}$.

Difference unpredictable hashing DUHF Security game: Let H be a keyed hash function and let the digest space \mathcal{T} have a group operation $+$ (with inverse $-$)⁹. Then:

1. Challenger sets $K \leftarrow \$ \{0, 1\}^k$
2. Adversary outputs distinct $m_0, m_1 \in \mathcal{M}$ and $\delta \in \mathcal{T}$.

\mathcal{A} wins if $H(K, m_0) - H(K, m_1) = \delta$ and has advantage $\mathbf{Adv}_H^{DUHF}(\mathcal{A})$.

H is an ε -bounded distance unpredictable hash function (ε -DUHF) if $\mathbf{Adv}_H^{DUHF}(\mathcal{A}) \leq \varepsilon$ for all \mathcal{A} (even unbounded ones).

XOR universal hashing Special case of DUHF where we have XOR as the group operation. We define H_{xpoly} as follows:

$$\begin{aligned} H_{xpoly}(K, (a_1, \dots, a_v)) &= K^{v+1} + a_1 K^v + a_2 K^{v-1} + \dots + a_{v-1} K^2 + a_v K \\ &= K \cdot H_{poly}(K, (a_1, \dots, a_v)) \end{aligned}$$

Theorem: H_{xpoly} is an ε -DUHF for $\varepsilon = \frac{l+1}{|\mathcal{F}|}$.

Carter-Wegman MACs Let H be an ε -DUHF and let F be a PRF. Then we define $\text{CW-MAC}(F, H)$ as follows:

$$\begin{aligned} KGen() : & (K_1, K_2) \leftarrow \$ \mathcal{K}_H \times \mathcal{K}_F \\ Tag((K_1, K_2), N, m) : & \tau \leftarrow H(K_1, m) + F(K_2, N) \\ Vfy((K_1, K_2), N, m, \tau) : & \tau' \leftarrow Tag((K_1, K_2), N, m); \text{ return } \tau' == \tau \end{aligned}$$

Theorem: If H is an ε -DUHF and F a PRF then $\text{CW-MAC}(F, H)$ is SUF-CMA secure.

In practice: e.g. GMAC (used in AES-GCM) instantiates F with AES and H with H_{xpoly} over $\mathcal{F} = GF(2^{128})$.

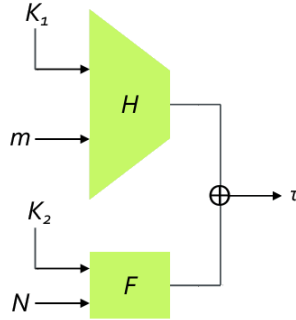


Figure 14: Carter-Wegman MAC tag algorithm

⁹E.g. addition mod n in \mathbb{Z}_n or XOR in $\{0, 1\}^n$.

1.6 Authenticated Encryption

Integrity of ciphertexts INT-CTXT Informally: adversary cannot forge new ciphertexts.

Security game: \mathcal{A} can submit any m to an encryption oracle to obtain $c = \text{Enc}_K(m)$. At the end \mathcal{A} submits one c^* to a *try* oracle.

\mathcal{A} wins if 1) c^* is distinct from all previously seen c and 2) c^* decrypts to a $m^* \neq \perp$.

\mathcal{A} has advantage $\text{Adv}_{SE}^{\text{INT-CTXT}}(\mathcal{A})$.

Integrity of plaintexts INT-PTXT Informally: adversary cannot force a new plaintext to be accepted by the receiver.

Same as INT-CTXT but with an additional winning requirement:

\mathcal{A} wins if ... 3) m^* is distinct from all previous queries m .

INT-CTXT \implies INT-PTXT¹⁰

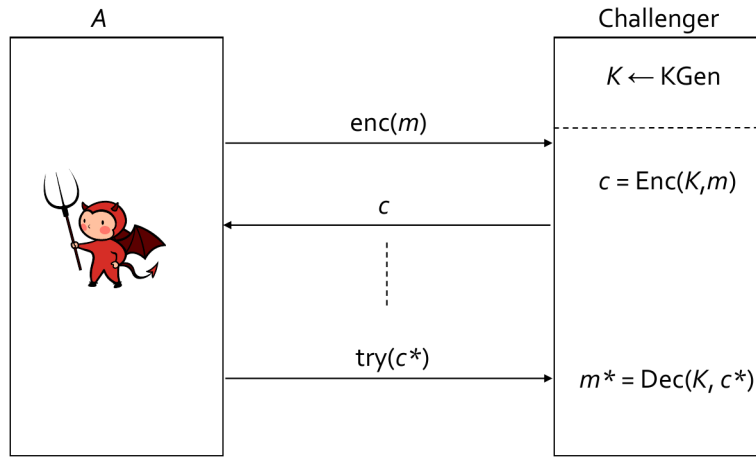


Figure 15: INT-CTXT + INT-PTXT game

IND-CCA Security (Indistinguishable under chosen ciphertext attack)

Same as IND-CPA (LoR oracle), with an additional decryption oracle.¹¹

Authenticated Encryption AE An SE is said to be *AE secure* if it is IND-CPA secure and INT-CTXT secure.

AE \implies IND-CCA¹²

Encrypt-and-MAC E&M Compute $c \leftarrow \text{Enc}_{KE}(m), \tau \leftarrow \text{Tag}_{KM}(m)$ and output $c' = c || \tau$.

NOT secure in general: If MAC is deterministic (e.g. a PRF) then trivial IND-CPA attack applies.

E.g. in SSH (in a stateful variant).

¹⁰Since due to SE correctness a new plaintext implies a new ciphertext.

¹¹Trivially, the adversary is not allowed to submit ciphertexts received from the LoR oracle to the decryption oracle.

¹²Informal proof by case distinction: Either at some point the IND-CCA adversary queries a c that does not decrypt to \perp (breaking INT-CTXT) or never, in which case the decryption oracle is useless (breaking IND-CPA).

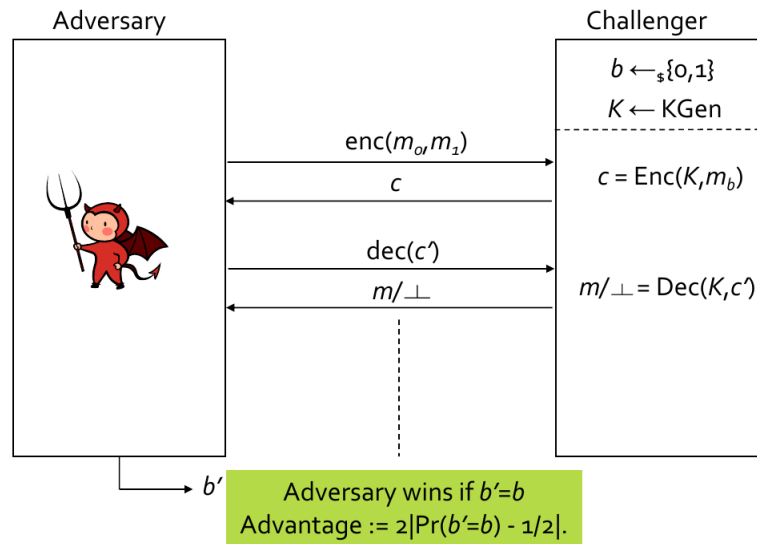


Figure 16: IND-CCA game

Mac-then-Encrypt MtE Compute $\tau \leftarrow \text{Tag}_{KM}(m)$ and output $c \leftarrow \text{Enc}_{KE}(m||\tau)$.

Can be made secure under specific instantiations, but unsafe in general – AVOID!

E.g. in TLS prior to 1.3.

Encrypt-then-Mac EtM Compute $c \leftarrow \text{Enc}_{KE}(m)$, $\tau \leftarrow \text{Tag}_{KM}(c)$ and output $c' = c||\tau$.

AE secure under the assumption that SE is IND-CPA and MAC is SUF-CMA.

E.g. in IPSec ESP.

AE with Associated Data AEAD Informally: integrity-protect some data, provide confidentiality for the rest. E.g. AES-GCM, ChaCha20-Poly1305.

Both Enc and Dec also take the associated data AD as an additional argument – but the ciphertext does not “contain” AD (instead AD is sent alongside, like the nonce N).

IND-CPA game: adversary sends $(N, AD, (m_0, m_1))$ to LoR oracle.

INT-CTXT game: adversary sends (N, AD, m) to encryption oracle and (N^*, AD^*, c^*) to the try oracle.

In practice: if nonces implicitly use a counter, then the adversary cannot reorder or delete messages (and of course can't insert either due to INT-CTXT).

EtM for AEAD:

$$c \leftarrow \text{Enc}_{KE}(m), \tau \leftarrow \text{Tag}_{KM}(\text{len}(AD)||AD||c); \text{ output } c' = c||\tau$$

Note that tagging over the length is required to avoid mis-parsing between AD and c . Enc and Tag may be nonce-based, e.g. AES-GCM.

AES-GCM (Galois Counter Mode) EtM with AES in CTR mode and a CW-MAC (H_{xpoly} over $GF(2^{128})$). Single key, MAC key is derived: $K_M = \text{AES}(K, 0^{128})$. Nonces are usually 96 bit and reuse results in catastrophic failure (MAC key recovery), thus not *nonce-misuse resistant*. Counters for SE are of the form $N||ctr$. Almost as fast as CTR mode, since UHF in MAC is fast.