Autentizační kód zprávy

Andrew Kozlík

KA MFF UK

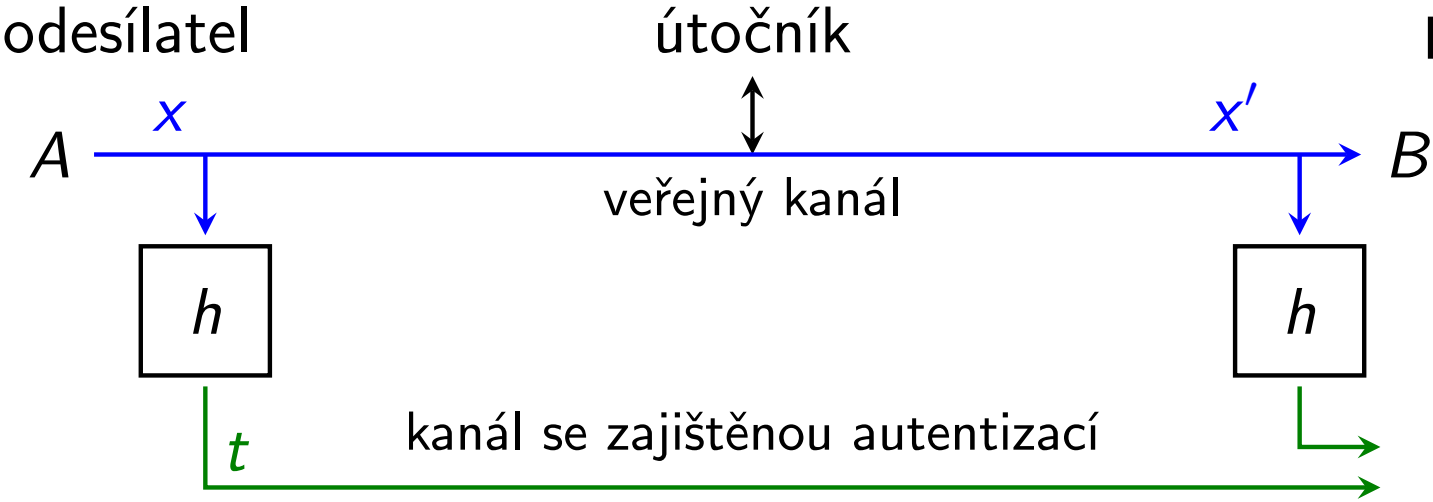
Autentizační kód zprávy

- Anglicky: message authentication code (MAC).
- MAC algoritmus je v podstatě hashovací funkce s klíčem:

$$\mathrm{MAC} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n, \quad \text{kde } k, n \in \mathbb{N}.$$

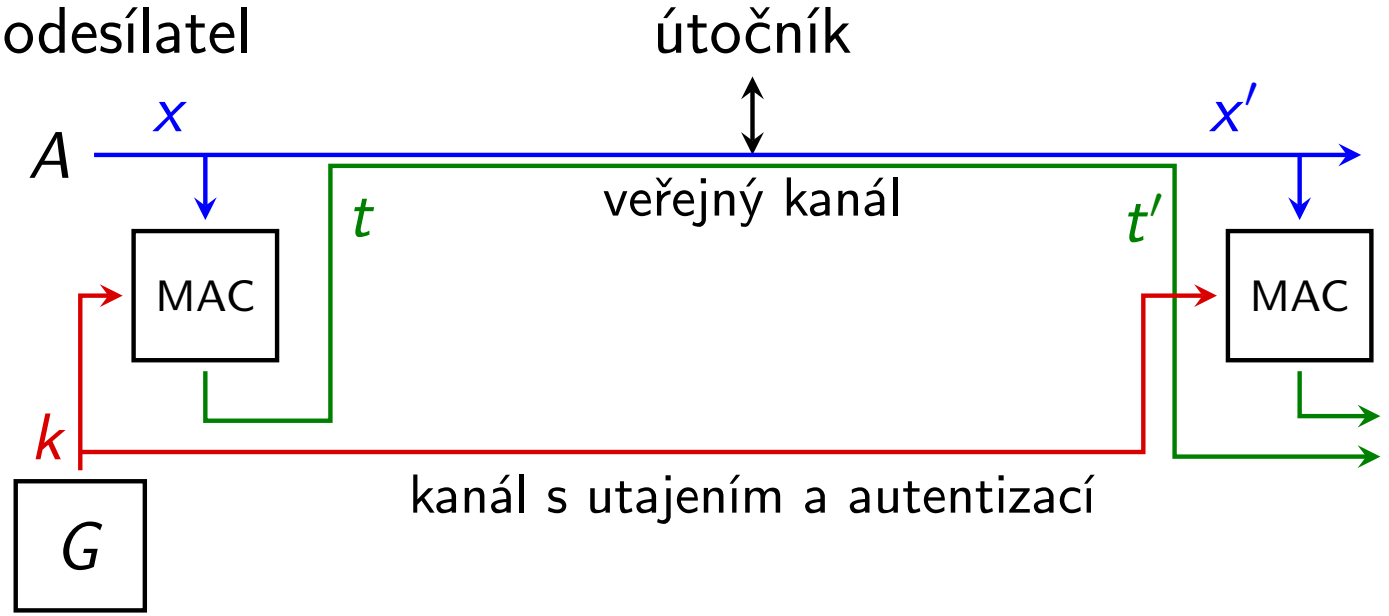
- MAC slouží k autentizaci původce zprávy.
- Implicitně zahrnuje i zajištění integrity zprávy.
- Obraz nazýváme „tag“ nebo prostě „MAC“.
- Tag můžeme připojit ke zprávě jako jakýsi „podpis“.
  Není to podpis, protože nezajišťuje nepopiratelnost!

Zajištění autentizace pomocí hashovací funkce



Jestliže $h(x') = t$, usoudíme, že $x' = x$ a že x pochází od A.

Zajištění autentizace pomocí MACu



Jestliže $\mathrm{MAC}(k, x') = t'$, usoudíme, že $x' = x$ a že x pochází od A.

Typy útoků na MAC

- **Known message útok**
  Útočník zná jeden nebo více párů (x_i, t_i), kde $t_i = \mathrm{MAC}(k, x_i)$ a k je konstantní.

- **Chosen message útok**
  Útočník volí zprávy $x_1, \ldots, x_n$ a dozvídá se hodnoty $t_i = \mathrm{MAC}(k, x_i)$, kde k je konstantní.

- Cílem je vytvořit novou dvojici (x, t), kde $t = \mathrm{MAC}(k, x)$.

Šifra nezajišťuje ani integritu ani autentizaci

- Alice posílá do banky platební příkaz ve prospěch Boba.
- Příkaz má standardní formát

$$x = \textit{pořadové číslo} \parallel N_A \parallel N_B \parallel \textit{částka}$$

  a zašifruje se proudovou šifrou

$$y = x \oplus s,$$

  kde s je keystream, N_A a N_B jsou čísla účtů Alice a Boba.
- Cestou do banky je zašifrovaný příkaz zadržen Evou.
- Ta ho pozmění a pošle dál:

$$y' = y \oplus (0 \ldots 0 \parallel 0 \ldots 0 \parallel N_B \oplus N_E \parallel 0 \ldots 0).$$

- Po dešifrování y' v bance dostaneme

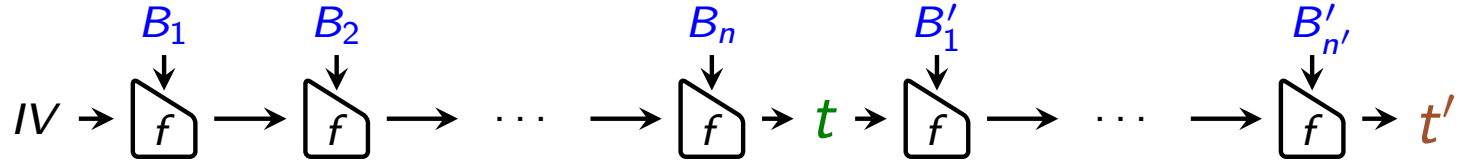
$$\textit{pořadové číslo} \parallel N_A \parallel N_E \parallel \textit{částka}.$$

Autentizované šifrování

- Encrypt-then-MAC
  - Máme klíče k_1 a k_2 (jeden může být odvozený z druhého).
  - Spočítáme $y = E(k_1, x)$ a $t = \mathrm{MAC}(k_2, y)$.
  - Pošleme (y, t).
- Existují operační režimy blokových šifer, které kombinují utajení a autentizaci.
  - Výstupem je šifrový text a tag.
  - Například: Galois Counter Mode (GCM).
- Mysleme i na obranu proti replay útokům:
  Do zprávy přidáme důkaz čerstvosti, např. pořadové číslo.

Konstrukce MAC algoritmu z hashovací funkce

- **Nápad:**
$$\mathrm{MAC}(k, x) := h(k \parallel x).$$
- **Problém:**
Když h má Merkleovu-Damgårdovu konstrukci:
  - Útočník odposlechne (x, t), kde $t = h(k \parallel x)$.
  - Pro libovolné x' může spočítat $h(k \parallel \overline{x} \parallel x')$ bez znalosti k.
  - Stačí navázat na posloupnost volání kompresní funkce:

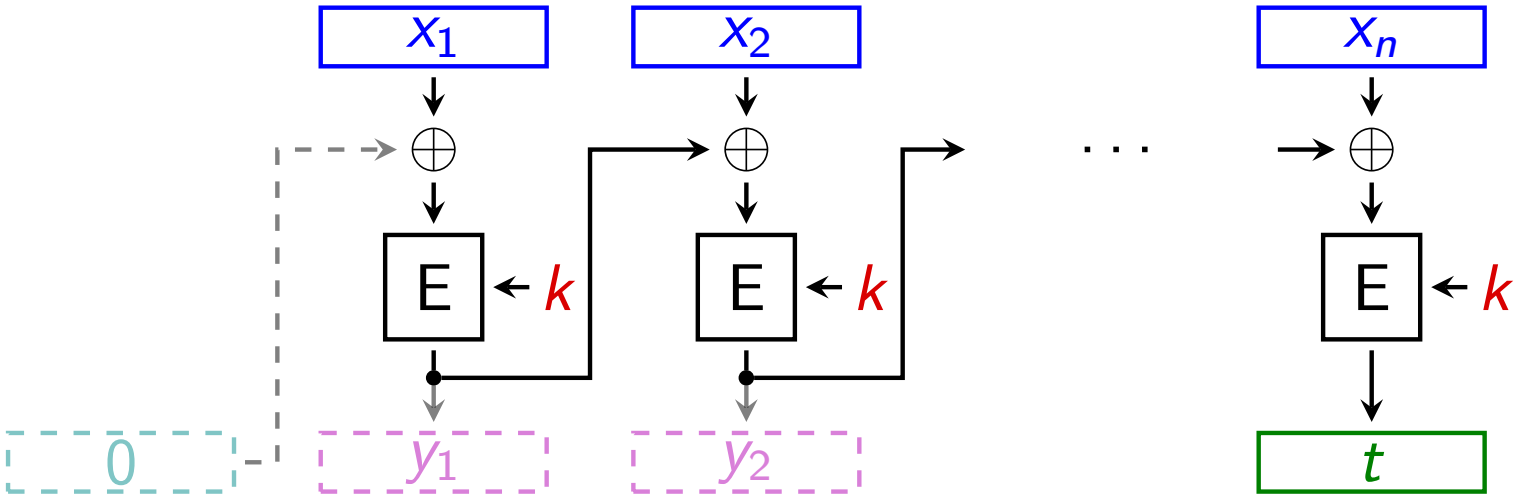


  - Tím získá tag t' pro zprávu $\overline{x} \parallel x'$, aniž by znal klíč k.
- **Řešení:**
$$\mathrm{HMAC}(k, x) := h\Big((k \oplus \mathrm{opad}) \Big\| h\big((k \oplus \mathrm{ipad}) \parallel x\big)\Big),$$
kde opad a ipad jsou standardem dané konstanty.

Konstrukce MAC algoritmu z blokové šifry

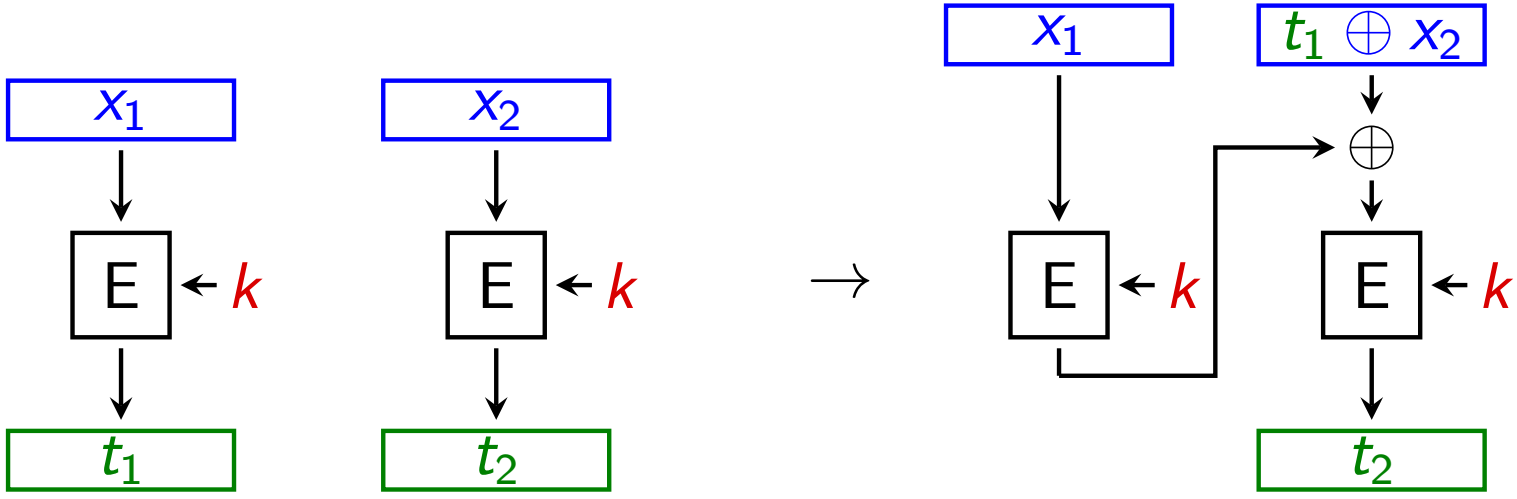
- **Nápad:** CBC-MAC
  - Zprávu vyplníme a rozdělíme na bloky $x_1, \ldots, x_n$.
  - „Zašifrujeme“ blokovou šifrou v CBC režimu, $IV = 0$.
  - Jako tag použijeme pouze poslední blok výstupu.

- Pokud zprávu zároveň šifrujeme, tak pro šifrování a CBC-MAC je obecně nutno použít různé klíče!



Konstrukce MAC algoritmu z blokové šifry

- **Problém:** Known message útok na CBC-MAC.
  - Útočník odposlechne (x_1, t_1) a (x_2, t_2), kde $t_i = \text{CBC-MAC}(k, x_i)$.
  - Pro jednoduchost předpokládejme, že délka x_i je rovna délce jednoho bloku šifry, potom $t_i = E(k, x_i)$.
  - Označme $x_3 := t_1 \oplus x_2$.
  - Potom $\text{CBC-MAC}(k, x_1 \parallel x_3) = E(k, E(k, x_1) \oplus x_3) = E(k, t_1 \oplus t_1 \oplus x_2) = t_2$.
  - Sestrojili jsme novou platnou dvojici $\big(x_1 \parallel (t_1 \oplus x_2),\ t_2\big)$.



Konstrukce MAC algoritmu z blokové šifry

- **Řešení:** EMAC
  Výsledek CBC-MACu navíc zašifrujeme jiným klíčem:

$$\text{EMAC}(k_1, k_2, x) := E(k_2, \text{CBC-MAC}(k_1, x))$$

- **Problém:** Chosen message útok
  - Podaří-li se útočníkovi narazit na kolizi (x_1, t) a (x_2, t), kde $t = \text{EMAC}(k_1, k_2, x_1) = \text{EMAC}(k_1, k_2, x_2)$, pak $\text{CBC-MAC}(k_1, x_1) = \text{CBC-MAC}(k_1, x_2)$.
  - Nechť x' je libovolný řetězec o délce jednoho bloku.
  - Uvažujeme-li chosen message útok, pak se útočník může dotázat na $\text{EMAC}(k_1, k_2, (x_1 \parallel x')) = t_3$.

Chosen message útok na EMAC

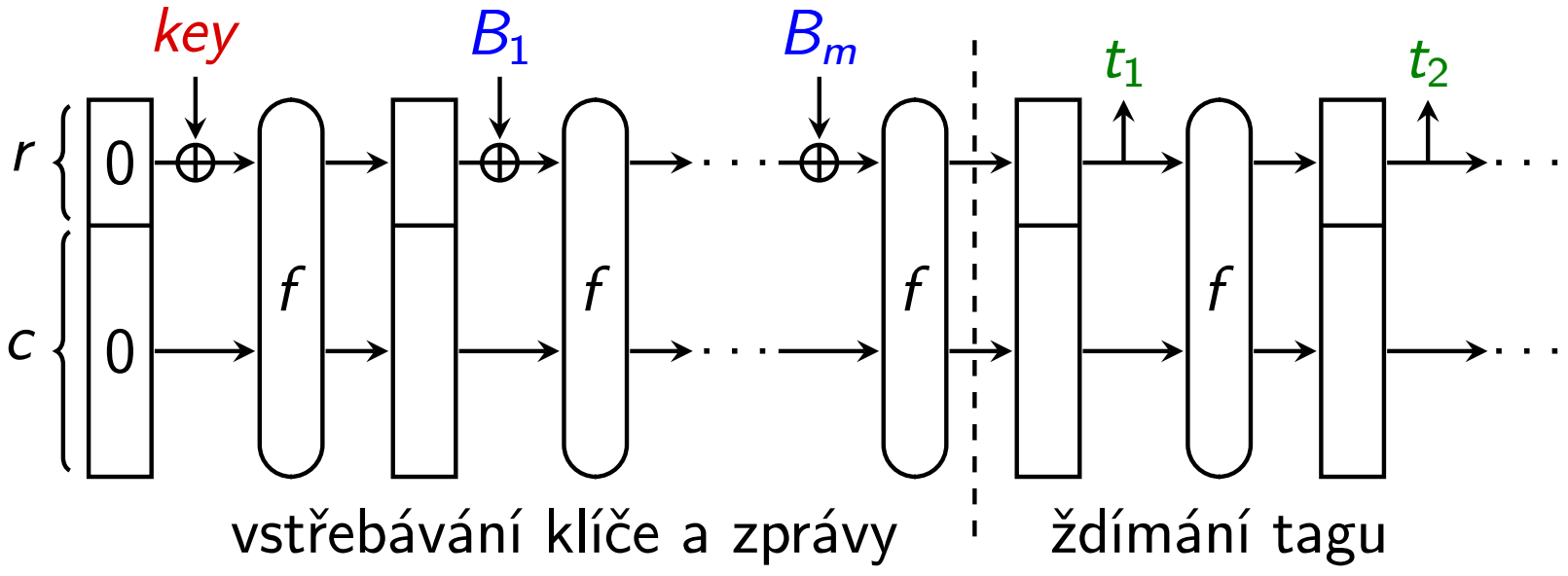
- Pokračování:
  - Nyní platí:

$$\begin{aligned}\text{EMAC}(k_1, k_2, (x_2 \parallel x')) &= E(k_2, \text{CBC-MAC}(k_1, (x_2 \parallel x'))) \\ &= E(k_2, E(k_1, \text{CBC-MAC}(k_1, x_2) \oplus x')) \\ &= E(k_2, E(k_1, \text{CBC-MAC}(k_1, x_1) \oplus x')) \\ &= E(k_2, \text{CBC-MAC}(k_1, (x_1 \parallel x'))) \\ &= \text{EMAC}(k_1, k_2, (x_1 \parallel x')) = t_3.\end{aligned}$$

  - Sestrojili jsme novou platnou dvojici $\big((x_2 \parallel x'), t_3\big)$, ačkoliv se útočník na tag pro $(x_2 \parallel x')$ nikdy nedotázal.
- **Řešení:**
  1. Omezíme počet zpráv, pro které se k_1 a k_2 použijí, aby kolize tagů byly málo pravděpodobné.
  2. Jako tag použijeme jen několik prvních bitů výstupu EMACu, aby útočník nemohl z tagu poznat kolize CBC-MACu.

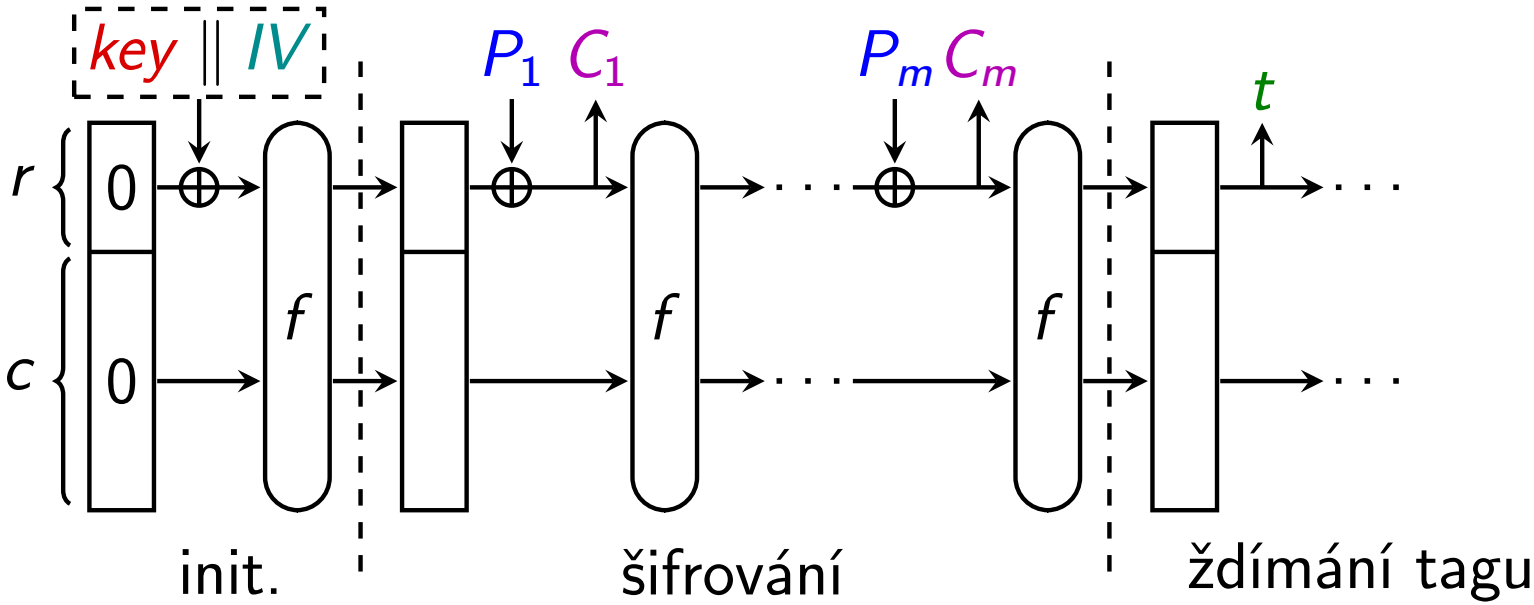
Konstrukce MAC algoritmu z houby

- Hashovací funkci, která je založena na houbovité konstrukci, lze použít jako MAC napřímo: $h(\mathit{key} \parallel \mathit{message})$.
  - Merkleova-Damgårdova konstrukce:
    Výstupem je celý vnitřní stav.
  - Houbovitá konstrukce:
    Na výstupu chybí minimálně c bitů stavu houby.
- Na tomto principu funguje KMAC (KECCAK Message Authentication Code algorithm).



Konstrukce autentizované šifry z houby

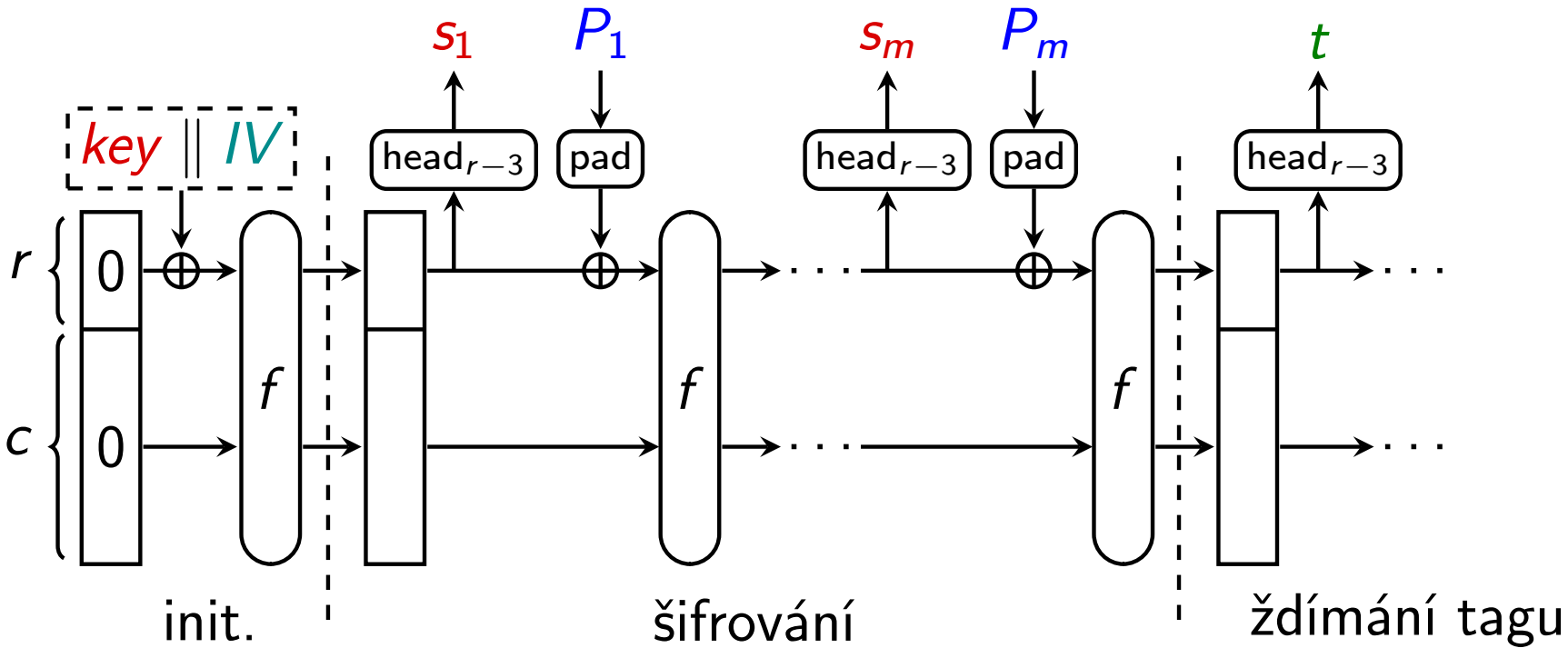
- Zjednodušený princip:



- **Problém:** known plaintext attack, kterým se zkrátí zašifrovaná zpráva a zfalšuje tag.
  - Útočník zachytí $(IV, C_1 \| \cdots \| C_m, t)$ a zná P_m.
  - Útočník odešle $(IV, C_1 \| \cdots \| C_{m-1}, t')$, kde $t' = P_m \oplus C_m$.
  - Po dešifrování příjemce získá $P_1 \| \cdots \| P_{m-1}$ a tag t' se příjemci jeví jako validní.

Autentizovaná šifra z duplexní konstrukce

- Otevřený text se rozdělí na bloky $P_1, \ldots, P_{m-1}$ délky $r-3$ bitů a blok P_m délky nejvýše $r-3$ bitů.
- K bloku P_m se připojí bitová výplň tvaru 010^*1 a k ostatním bitová výplň 110^*1, čili 111, aby všechny bloky měly délku r.
- Průběžně čteme vnější část stavu houby a přidáváme OT. Výsledkem je proud hesla $s_1, \ldots, s_m$ a tag t.



K čemu je výplň v duplexní konstrukci?

- První bit výplně je tzv. *frame bit*, který je $b_f = 0$ v posledním bloku a $b_f = 1$ v ostatních.
- Frame bit brání útočníkovi ve vytvoření validního tagu pro zkrácenou zprávu.
- Jako výplň lze použít $b_f 10^* 1$ nebo $b_f 10^*$. První má výhodu, že brání padělku MACu mezi konstrukcemi s různým ratem.
- Existují i jiná řešení, která nevyužívají frame bit, např. přimíchají znovu klíč na konci šifrování (šifra Ascon).

Příklad využití MACu: TOTP

TOTP: Time-Based One-Time Password Algorithm (RFC 6238)

- K je předem domluvený klíč, obvykle sdílený přes QR kód.
- C je počet vteřin od 1.1.1970 děleno 30 zaokrouhleno dolů.
- Truncate: vybere se určitých 31 bitů z MACu, vyjádří se jako celé číslo a vrátí se zbytek po dělení 10^6.
- Uživatel spočte Truncate(HMAC-SHA1(K, C)) a použije ho jako jednorázový přístupový kód.