KRY Projekt 2: MAC za použití SHA-256 & Length extension attack

Cíl projektu

V rámci druhého projektu předmětu KRY budete implementovat naivní generátor MAC (Message Authentication Code) pro podpis zpráv za použití Vámi implementovaného hashovacího algoritmu a tajného klíče. Nástroj bude umožňovat generování MAC zprávy, ověření MAC zprávy a provedení útoku na rozšíření zprávy o libovolný text a přepočítání MAC bez znalosti tajného klíče.

1. Implementujte hashovací algoritmus SHA-256.
2. Implementujte mechanismus generování MAC (Message Authentication Code) pro vstupní zprávu a tajný klíč.
3. Implementujte mechanismus ověření MAC (Message Authentication Code) pro vstupní zprávu a tajný klíč.
4. Implementujte length extension attack na použití SHA-256 pro generování MAC.

Specifikace

Během implementace hašovacího algoritmu SHA-256 postupujte dle standardu NIST FIPS 180-4[1]. Pro ověření výsledků použijte nástroj `sha256sum`, který je dostupný na **referenčním stroji Merlin**. Pro kontrolu výsledků v konkrétních krocích algoritmu SHA-256 doporučuji použít nástroj `https://sha256algorithm.com/`.

Dalším krokem je implementace *naivního* generátoru MAC využitím dříve implementovaného SHA-256. MAC zaručuje dva bezpečnostní cíle – autentizaci (pomocí utajeného klíče) a integritu (pomocí ireverzibilní vlastnosti hashovací funkce). Vytvoření MAC probíhá následujícím způsobem:

$$MAC = SHA256(SECRET\_KEY + MSG)$$

Kontrola MAC probíhá za dostupnosti zprávy i tajného klíče a je prováděna porovnáním dodaného MAC a nově vypočítaného MAC ze vstupní zprávy a tajného klíče.

Poslední částí projektu je implementace nástroje pro provedení length extension útoku na popsané použití SHA-256 pro mechanismus MAC. Jako základ studijní literatury mohou posloužit tyto zdroje:

1 `http://dx.doi.org/10.6028/NIST.FIPS.180-4`

- `https://lord.io/length-extension-attacks/`
- `https://bostik.iki.fi/dc4420/size_t-does-matter--hash-length-extensions.pdf`
- `https://www.javacodegeeks.com/2012/07/hash-length-extension-attacks.html`

Pro ověření výsledků bude použita Python knihovna `length-extension-tool`[2]. V případě využití knihovny pro testovací účely při vývoji dejte pozor na vlastnosti tištění posloupnosti bytů v jazyce Python v souvislosti s požadovaným formátem padding u výsledné zprávy při provedení útoku.

Vaše aplikace čte vstupní zprávu z STDIN a musí podporovat následující rozhraní a s ním spojenou funkcionalitu:

- Program spuštěný bez parametrů – program tiskne dokumentaci použití na STDOUT a vrací hodnotu `1`.
- Program může být spuštěný s právě jedním z následujících přepínačů. Každý následující přepínač mění funkcionalitu aplikace:
  `-c` – Vypočte a tiskne SHA-256 checksum vstupní zprávy na STDOUT (zakončeno právě jedním `\n`).
  `-s` – Vypočte MAC, použitím implementované SHA-256, pro vstupní zprávu a tisne výsledek na STDOUT (zakončeno právě jedním `\n`). Musí být použito v kombinaci s `-k` parametrem.
  `-v` – Ověří MAC pro daný klíč a vstupní zprávu. Vrací `0` v případě validního MAC, jinak `1`. Musí být použito v kombinaci s `-k` a `-m` parametry.
  `-e` – Provede length extension útok na MAC a vstupní zprávu. Přepočítaný MAC a prodloužená zpráva jsou tištěny na STDOUT v tomto pořadí (každá položka zakončena právě jedním `\n`). Padding přidaný ke vstupní zprávě bude ve formátu posloupnosti escapovaných znaků `\xXX`, kde `XX` bude ASCII hodnota znaku v hexadecimální soustavě. Musí být použito v kombinaci s `-m`, `-n` a `-a` parametry.
- Program může vyžadovat další parametry vzhledem ke zvolené funkcionalitě. Tyto parametry specifikují dodatečné vstupní informace:
  `-k KEY` – Specifikuje tajný klíč pro výpočet MAC. Formát klíče očekávejte dle regulárního výrazu `^[A-Fa-f0-9]*$`.
  `-m CHS` – Specifikuje MAC vstupní zprávy pro jeho verifikace či provedení útoku.
  `-n NUM` – Specifikuje délku tajného klíče (nutnou pro provedení útoku).
  `-a MSG` – Specifikuje prodloužení vstupní zprávy pro provedení útoku. Formát rozšíření zprávy očekávejte dle regulárního výrazu `^[a-zA-Z0-9!#$%&'"()*+,\-.\/:;<>=?@[\]\\^_{}|~]*$`.

Na případy nesprávného použití parametrů aplikace reaguje chybovým návratovým kódem

2 `https://github.com/viensea1106/hash-length-extension`

a hláškou na STDERR. Všechny SHA-256 checksum řetězce / MACs budou formátovány dle regulárního výrazu `^[A-Fa-f0-9]{64}$`.

Technické požadavky

- Projekt implementujte v jazyce C/C++.
- Je zakázáno používat jiné než standardní knihovny C/C++, s výjimkou knihovny GMP pro práci s velkými čísly.
- Projekt musí obsahovat soubor Makefile.
- Při spuštění příkazu `make` ve složce projektu se zkompiluje a sestaví binární soubor `./kry`.
- Projekt nesmí obsahovat žádné složky.
- Projekt musí být sestavený pomocí gcc se zaplými sanitizéry `address` a `memory` (přepínač `-fsanitize=NAME`).

Příklady

- ```
  ~> echo -ne "zprava" | ./kry -c
  d8305a064cd0f827df85ae5a7732bf25d578b746b8434871704e98cde3208ddf
  ~>
  ```

- ```
  ~> echo -ne "zprava" | ./kry -s -k heslo
  23158796a45a9392951d9a72dffd6a539b14a07832390b937b94a80ddb6dc18e
  ~>
  ```

- ```
  ~> echo -ne "zprava" | ./kry -v -k heslo -m 23158796a45a9392951d
  9a72dffd6a539b14a07832390b937b94a80ddb6dc18e
  ~> echo $?
  0
  ~>
  ~> echo -ne "message" | ./kry -v -k password -m 23158796a45a9392
  951d9a72dffd6a539b14a07832390b937b94a80ddb6dc18e
  ~> echo $?
  1
  ~>
  ```

- ```
  ~> echo -ne "zprava" | ./kry -e -n 5 -a ==message -m 23158796a45
  a9392951d9a72dffd6a539b14a07832390b937b94a80ddb6dc18e
  ```

```
a3b205a7ebb070c26910e1028322e99b35e846d5db399aae295082ddecf3edd3
zprava\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x58==message
~>
```

Odevzdávání a hodnocení

Projekt odevzdávejte do systému ve formátu `000000.zip`, kde *000000* nahradíte **Vaším osobním číslem**. Archív bude obsahovat zdrojové soubory, soubor Makefile a soubor README.md se stručnou dokumentací. V případě potřeby konzultace se obraťte na iplesko@fit.vut.cz.

- **Jakákoliv forma plagiátorství bude hodnocena 0 body a může vést k zahájení disciplinárního řízení!**
- Archív, který bude obsahovat složky = 0 bodů
- Archív, který nebude obsahovat Makefile = 0 bodů
- Projekt, který nepůjde přeložit přiloženým Makefile souborem na **referenčním serveru Merlin** = 0 bodů
- Použití AI asistentů (ChatGPT, Github Copilot, Gemini, ...) = 0 bodů
- Upozornění sanitizérů = strženy 2 body
- Chybějící dokumentace = strženy 2 body
- Projekt bude hodnocen převážně automatickými testy – dbejte na přesnou implementaci rozhraní!