# Srovnání protokolů pro „Remotely Keyed Encryption“

**Petr Švenda**

xsvenda@fi.muni.cz

Faculty of Informatics
Masaryk University
Brno, Czech Republic

## Abstrakt

Protokoly pro vzdálené šifrování (Remotely Keyed Encryption, dále RKE) jsou určeny pro prostředí, ve kterém má výkonný, ale nedůvěryhodný hostitel přístup k dostatečně bezpečné čipové kartě, avšak s omezenou výkonností. Šifrování dat v tomto prostředí je standardně prováděno dvěma způsoby: 1. Čipová karta slouží pouze jako bezpečný nosič šifrovacího klíče a během šifrování je klíč nahrán do prostředí hostitele. 2. Veškerá data jsou šifrována přímo na kartě. Nevýhodou prvního řešení je přítomnost šifrovacího klíče v nedůvěryhodném prostředí hostitele, nevýhodou druhého pak většinou nedostatečná rychlost z důvodu omezeného výkonu a propustnosti komunikačního rozhranní karty. Protokoly RKE umožňují přenést většinu operací na stranu hostitele a zároveň ponechat šifrovací klíč pouze na čipové kartě. Přínosem příspěvku je ucelený přehled známých RKE protokolů, doplněný o výkonnostní srovnání na čipové kartě Gemplus GXPLite-Generic.

Klíčová slova: Remotely Keyed Encryption, smart card, untrusted host.

## Úvod

Protokoly pro vzdálené šifrování (Remotely Keyed Encryption, dále RKE) jsou určeny pro prostředí, ve kterém je výkonný, ale nedůvěryhodný hostitel s přístupem k dostatečně bezpečné čipové kartě, avšak s omezenou výkonností. Šifrování dat v tomto prostředí bylo standardně prováděno těmito následujícími způsoby:

1. Čipová karta je použita pouze jako úložiště šifrovacího klíče, který je v potřebném okamžiku načten do nedůvěryhodného prostředí hostitele a použit jako vstup šifrovacího algoritmu implementovaného v hostitelském prostředí. Při kompromitaci hostitelského prostředí může dojít i ke kompromitaci šifrovacího klíče, například s využitím systémových ladících nástrojů. Celkový výkon je závislý pouze na výkonu hostitele.
2. Čipová karta obsahuje šifrovací klíč a implementaci šifrovacího algoritmu. Hostitel zasílá vstupní data, čipová karta provede zašifrování resp. dešifrování a zpět vrací pouze zpracovaný výstup. Celkový výkon je závislý na rychlosti použité čipové karty a může být pro potřeby aplikace nedostatečný.

Cílem RKE protokolů je navrhnout postup, který vhodným způsobem přesune část operací z karty na nedůvěryhodného hostitele tak, aby šifrovací klíč(e) nebylo nutné pro dosažený požadovaného výkonu přesouvat do nedůvěryhodného prostředí hostitele. Navržené protokoly využívají jako základ standardních kryptografických primitiv typu bloková šifra, hashovací funkce a generátor náhodných čísel. Zjednodušeně pracují v následujících krocích (viz. **Obrázek** 1):

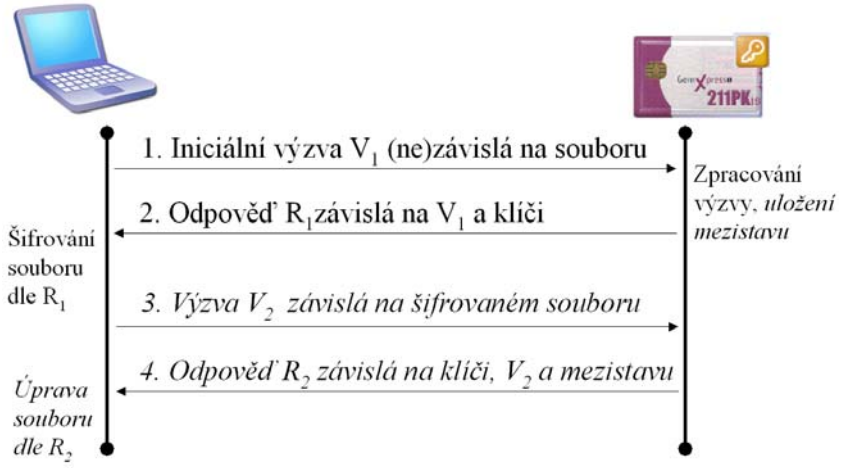
1. Hostitel na základě vstupních dat připraví výzvu(y) s fixní délkou (nezávislou na délce vstupní zprávy), kterou zašle kartě.
2. Karta na základě nesené tajné informace (typicky šifrovací klíč) výzvu zpracuje a zašle zpět odpověď fixní délky.
3. Hostitel na základě odpovědi provede na vstupními daty sérii operací. V některých případech je formulována ještě druhá výzva a karetní odpověď opět zpracována. Vstupní data jsou nyní zašifrovaná data a bez přístupu ke kartě by je nemělo být možné dešifrovat. Na základě získané výzvy by nemělo být možné získat jakoukoli informaci o tajné informaci nesené na kartě.

Z důvodu specifické konfigurace nelze použít běžný model útočníka. Jsou proto zavedeny upravené modely reflektující možnost přístupu útočníka k důvěryhodné a nedůvěryhodné části. **Základní model** útočníka zavedený v [Bl96] předpokládá, že útočník nemá přístup ke kartě a nemůže vytvářet vlastní výzvy pro kartu.

Útočník má plnou kontrolu nad operacemi vykonávanými v hostitelské části včetně použitých hodnot. Protokol je považován za bezpečný, pokud útočník není schopen bez využití karty dešifrovat zašifrovanou zprávu ani vytvořit korektní zašifrovanou zprávu. Později byl model rozšířen podmínkami pro **silný BFN model** [BFN98]. Silný BFN model oproti základnímu modelu navíc předpokládá útočníka, který na omezenou dobou získá možnost přístupu ke kartě a je schopen získávat zpracovaný výstup pro libovolnou korektní výzvu.

Protokol je bezpečný proti *inverznímu* útoku („inversion secure"), pokud útočník s přístupem k dešifrovacímu směru protokolu není schopen zašifrovat náhodně vybraný text a naopak. Protokol je považován za *pseudonáhodný*, pokud jím realizovaná bloková šifra splňuje vlastnost pseudonáhodnosti, tedy zašifrovaný text není rozlišitelný od náhodného řetězce. Protokol je bezpečný proti *podvrhnutému vstupu* („forgery secure"), pokud útočník s dočasným přístupem k kartě není schopen na základě libovolného počtu jím volených výzev zašifrovat/dešifrovat text různý od provedených výzev.

Podmínka bezpečnosti proti *inverznímu* útoku a *podvrhnutému vstupu* je relevantní pouze v silném BFN modelu, neboť v základním modelu se předpokládá, že útočník nemá přístup ke kartě a nemá tedy přístup k zařízení realizující jeden směr protokolu (*inverzní* útok) ani nemůže zasílat omezenou dobu výzvy (útok *podvrhnutým vstupem*). Pokud je protokol bezpečný v silném BFN modelu, musí splňovat všechny tři výše uvedené podmínky. Pro bezpečnost v základním útočníkově modelu postačuje splnění podmínky *pseudonáhodnosti* (z výše uvedených).




**Obrázek 1: Diagram výměny zpráv během RKE protokolu. Kurzívou jsou vyznačeny procesy prováděné jen některými protokoly.**

# Popis schémat

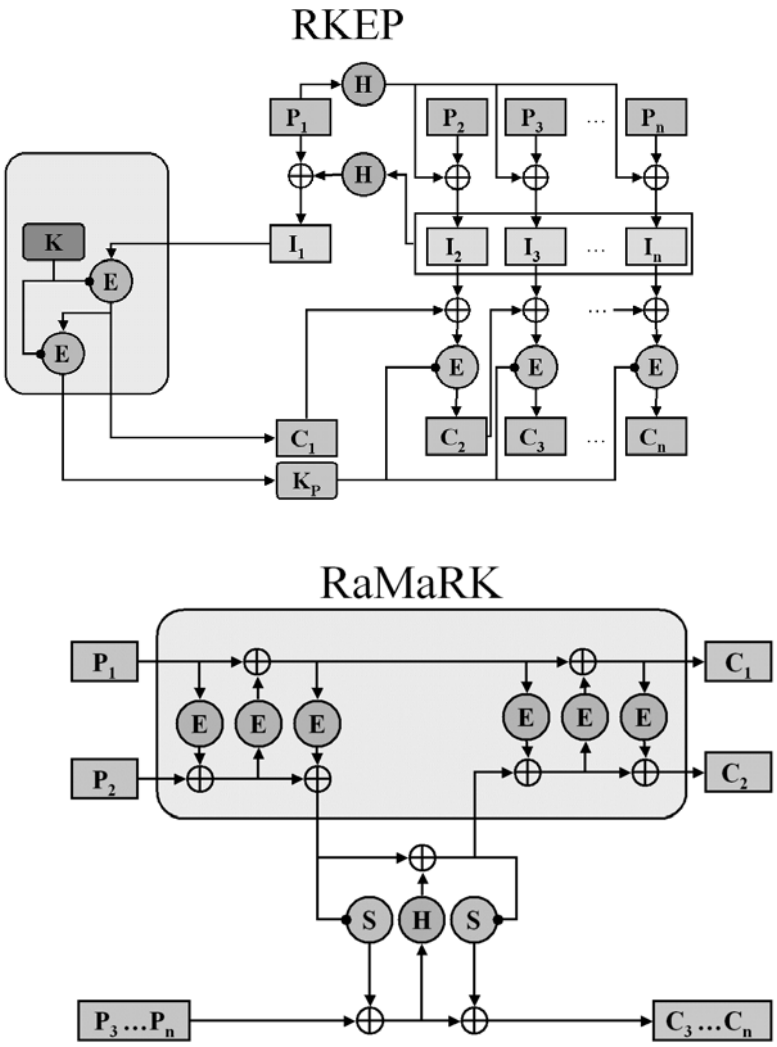
V dalším textu bude použito následující označení. $P_i$ resp. $C_i$ značí i-tý blok otevřeného resp. zašifrovaného textu. $I_i$ značí i-tý blok hodnoty průběžného výpočtu. H značí blokovou hashovací funkci s fixním výstupem (např. MD5, SHA-1). LPH značí hashovací funkci s délkou výstupu odpovídajícímu délce vstupu. E značí blokovou šifrovací funkci (např. DES, AES). Operace bitového exklusivního součtu je značena jako $\oplus$. Šifrovací klíč je značen písmenem K a $K_p$, s případným číselným indexem $K_i$. Generátor náhodných čísel je značen jako RND, náhodný řetězec jako R. Klíčovaná náhodná funkce je značena jako F. Funkce pro generování proudu klíčů je značena jako S.

## 1.1 RKEP

Remotely Keyed Encryption Protokol. První z publikovaných protokolů pro RKE [Bl96]. Popisuje vhodný scénář použití a zavádí odpovídající upravený model útočníka. Na hostitelské straně využívá šifrovací a hashovací funkci, na straně karty pouze šifrovací funkci. Šifrovací ani dešifrovací proces neumožňuje paralelizovat hostitelské a karetní operace. RKEP zachovává délku šifrované zprávy, což je výhodné především

pro krátké zprávy. Protokol vyměňuje jeden příkaz APDU (Application Protocol Data Unit)† komunikačního rozhranní ISO7816.

Protokol je zranitelný několika útoky publikovanými v [Lu97]. Útočník je schopen bez přístupu ke kartě zašifrovat text P, pokud je mu umožněno získat dvojice $<P_1, C_1>$ a $<P_2, C_2>$, kde útočník volí $P_1$ a $P_2$ vypadající jako náhodné řetězce různé od P. Útočník je schopen bez přístupu ke kartě dešifrovat text C, pokud je mu umožněno získat dvojice $<P_1, C_1>$ a $<P_2, C_2>$, kde útočník volí $C_1$ a $P_2$ vypadající jako náhodné řetězce a různé C. Protokol nelze považovat za bezpečný a neměl by být použit.




### 1.2 RaMaRK

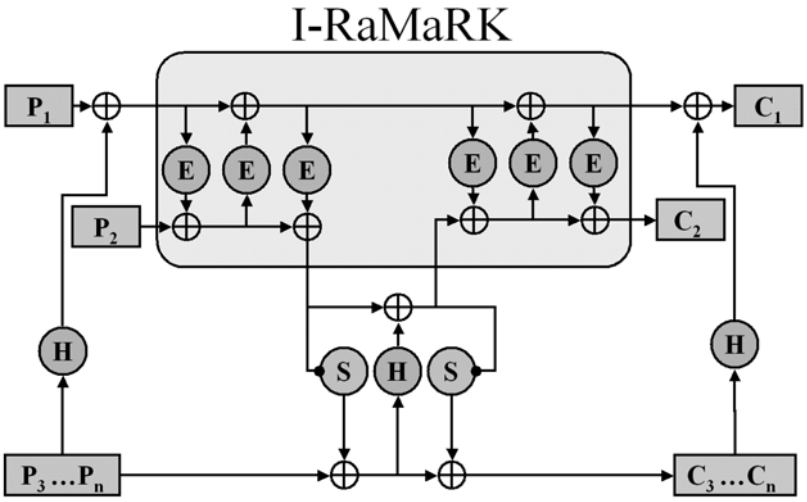
Random Mapping based Remotely Keyed protokol [Lu97]. Formálně zaveden útočníkův model bez přístupu ke kartě a zavedeny vlastnosti, které by měl splňovat bezpečný RKE protokol. Je poskytnut formální důkaz bezpečnosti tohoto protokolu.

Šifrovací i dešifrovací proces umožňuje částečně paralelizovat hostitelské a karetní operace, neboť jedna z APDU zpráv protokolu může být vykonána souběžně s prováděním jedné funkce S. RaMaRK zachovává délku šifrované zprávy a vyměňuje dva APDU příkazy.

Protokol je zranitelný útokem publikovaným v [BFN98] pod útočníkovým modelem s omezenou dobu přístupu ke kartě. Výsledek zašifrování je závislý pouze na prvních dvou blocích vstupu a každé dvě zprávy se stejným

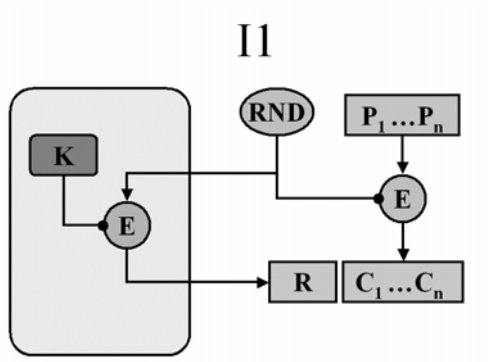
† Elementární datový paket obsahující krátkou hlavičku (6B) a volitelné datové pole obsahující vstupní nebo výstupní data pro čipovou kartu (cca 250B). Čipová karta nemůže sama iniciovat komunikaci, pouze odpovídá na přijatý APDU příkaz z PC.

počátkem budou zašifrovány stejně[†]. Upravenou verzí odolnou vůči tomuto útoku je I-RaMaRK, která by měla být použita namísto původního protokolu.




### 1.3 I-RaMaRK

Improved Random Mapping based Remotely Keyed protokol [BWL00]. Protokol odstraňuje zjištěné bezpečnostní nedostatky zavedením závislosti šifrování na celém vstupu pomocí hashovacích funkcí. Operace umístěné na kartě zůstávají beze změn. Přidané hashovací operace jsou provedeny pouze na hostitelské straně, proto by jejich vliv na výsledný výkon neměl být výrazný. Protokol je shodně s původním částečně paralelizovatelný. I-RaMaRK zachovává délku šifrované zprávy a vyměňuje dvě APDU.

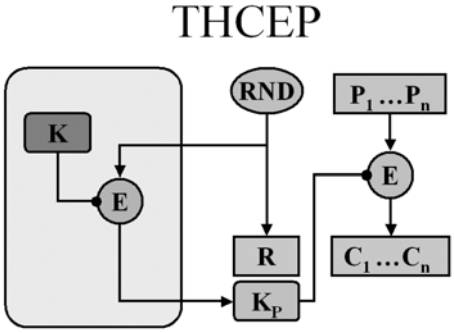



### 1.4 I1

Protokol I1 [BFN98]. Jedná se o jednoduchý protokol využívající generátor náhodných čísel pro vytvoření šifrovacího klíče, který je přiložen k výsledné zprávě ve tvaru zašifrovaném pomocí klíče uloženého na kartě. I1 prodlužuje šifrovanou zprávu o jeden blok a vyměňuje jeden APDU příkaz. Šifrovací proces umožňuje paralelizovat hostitelské a karetní operace, dešifrovací nikoli.

Protokol není bezpečný v silném BFN modelu. Výzva zpracovávaná kartou není závislá na obsahu šifrované zprávy, protokol tak umožňuje útočníkovi s omezenou dobou přístupu ke kartě generovat libovolné množství dvojic <R, $K_p$>, které může později použít pro vytvoření korektní zašifrované zprávy. I1 je proto určen pro scénáře užití, kde postačuje bezpečnostní model bez možnosti dočasného přístupu útočníka ke kartě.

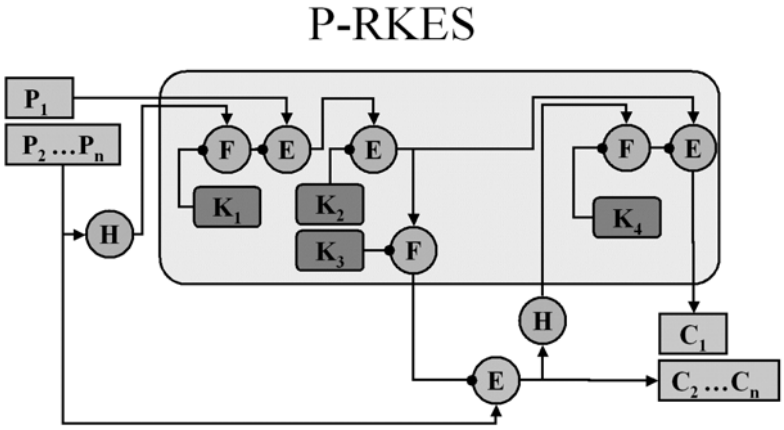
[†] Není splněna podmínka pseudonáhodnosti.




### 1.5 THCEP

Trivial Host Card Protocol [We00]. Jedná se o jednoduchý protokol využívající generátor náhodných čísel pro vytvoření výzvy, která je po zpracování kartou použita jako šifrovací klíč. Použitá výzva je přiložena k výsledné zprávě. THCEP prodlužuje šifrovanou zprávu o jeden blok a vyměňuje jeden APDU příkaz. Šifrovací ani dešifrovací proces není paralelizovatelný. Na rozdíl od I1 je zpracování výzvy kartou shodné pro šifrovací i dešifrovací proces protokolu a blokovou šifru na kartě lze nahradit např. HMAC konstrukcí nebo pouze šifrovacím algoritmem blokové šifry.

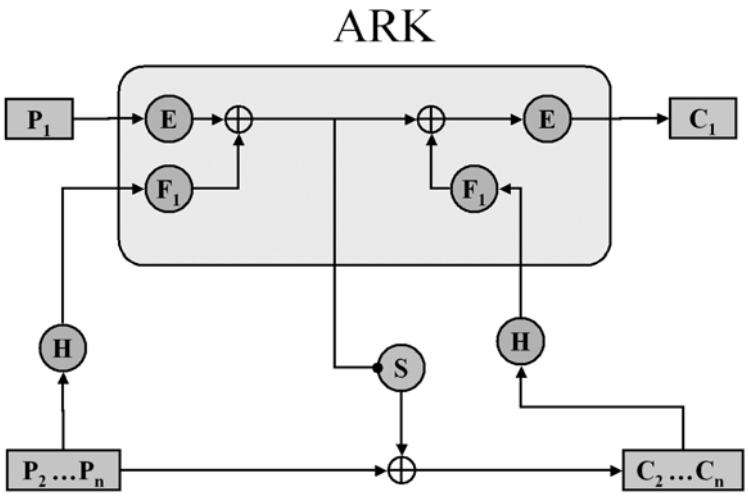
Protokol není bezpečný v silném BFN modelu, stejně jako I1. THCEP je proto určen pro scénáře užití, kde postačuje bezpečnostní model bez možnosti dočasného přístupu útočníka ke kartě.




### 1.6 P-RKES

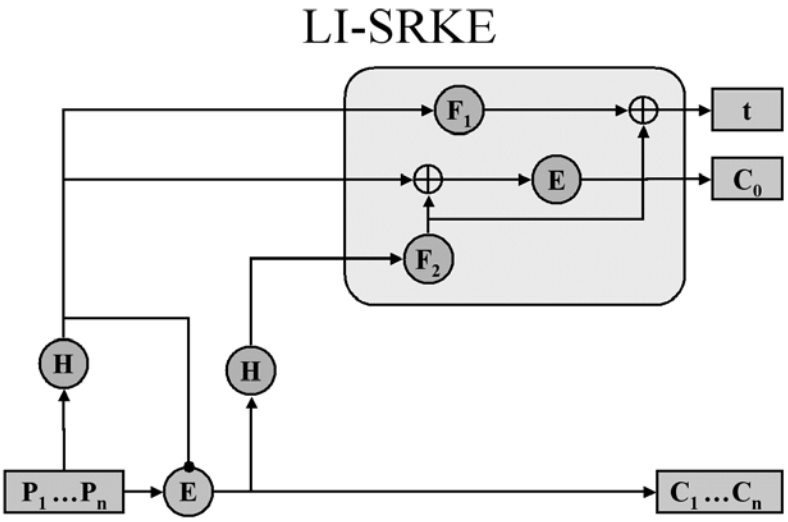
Length Preserving Remotely Keyed Encryption Scheme [BFN98]. Protokol odstraňuje bezpečností nedostatky protokolu RKES v základním bezpečnostním modelu. Je proveden formální důkaz bezpečnosti protokolu v silném BFN modelu. Na hostitelské straně využívá šifrovací a hashovací funkci, na straně karty šifrovací funkci E a klíčované náhodné funkce F[†]. Šifrovací algoritmus na kartě má klíč závislý na vstupních datech, je tedy nutné provádět přípravu klíče, což negativně ovlivňuje celkový výkon protokolu. Šifrovací ani dešifrovací proces není paralelizovatelný. P-RKES zachovává délku šifrované zprávy a vyměňuje dva APDU příkazy.

[†] Může být realizováno vhodnou blokovou šifrovací nebo hashovací funkcí.




### 1.7 ARK

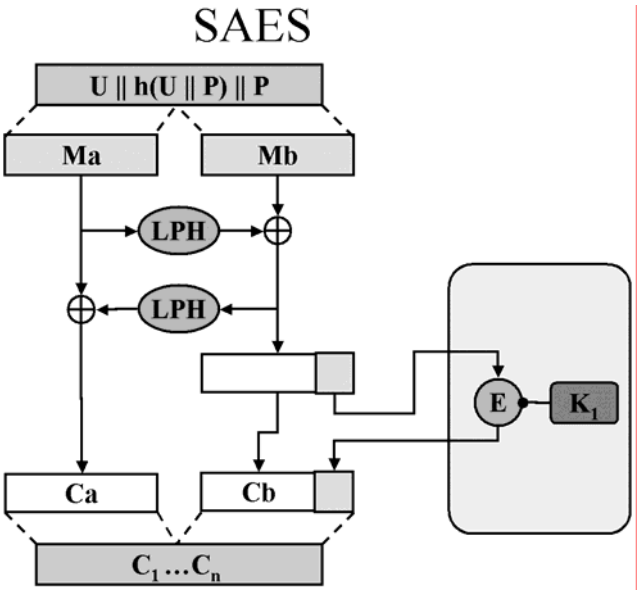
Accelerated Remotely Keyed Encryption protocol [Lu99]. Na hostitelské straně využívá hashovací funkci H a funkci S pro generování klíčového proudu, na straně karty šifrovací funkci E a klíčované náhodné funkce F. Je proveden formální důkaz bezpečnosti protokolu v silném BFN modelu. Šifrovací ani dešifrovací proces není paralelizovatelný. ARK zachovává délku šifrované zprávy a vyměňuje dva APDU příkazy.




### 1.8 LI-SRKE

Length Increasing Secure Remotely Keyed Encryption [SSR00]. Protokol je založen myšlence "All or Nothing Transformation" [Ri97] zajišťující, aby pro zisk libovolného bloku zašifrované zpráva bylo nutné dešifrovat všechny ostatní bloky. Na hostitelské straně využívá šifrovací funkci E a hashovací funkci H, na straně karty šifrovací funkci E a klíčované náhodné funkce F[†]. Šifrovací ani dešifrovací proces není paralelizovatelný. LI-SRKE prodlužuje délku šifrované zprávy o dva bloky a vyměňuje jeden APDU příkaz. Je proveden formální důkaz bezpečnosti protokolu v silném BFN modelu. Na rozdíl od protokolů P-RKES a ARK vyměňuje pouze jeden APDU příkaz, což je významné pro krátké zprávy.

[†] Může být realizováno vhodnou blokovou šifrovací funkcí.




### 1.9 SAES

Scramble All, Encrypt Small [JSY99]. Protokol využívá kryptograficky bezpečné hashovací funkce s výstupem stejné délky, jako vstupní zpráva (Length preserving hash function, dále LPH). Během hostitelských operací je vstupní zpráva nejprve doplněna náhodným řetězcem R a výstupem hashovací funkce s fixní délkou výstupu h aplikované na R a vstupní zprávu P. Výsledek je rozdělen na dvě poloviny a střídavě zpracován pomocí hashovací funkce LPH formou Feistelova schématu. Poslední blok druhé části je použit jako výzva a použit jako poslední blok šifrovaného výstupu. SAES prodlužuje délku šifrované zprávy o dva bloky[†] a vyměňuje jeden APDU příkaz. Šifrovací proces umožňuje částečně paralelizovat hostitelské a karetní operace při výpočtu druhé LPH funkce, dešifrovací proces paralelizovatelný není.

Je proveden formální důkaz bezpečnosti protokolu v modelu analogickém k silnému BFN modelu se zohlednění počáteční randomizace vstupních dat řetězcem R. Protokol poskytuje speciální ochranu proti útoku hrubou silou na hodnotu použitého klíče, neboť útočník je nucen pro ověření zvolené hodnoty klíče dešifrovat všechny bloky zašifrované zprávy, náročnost útoku tak roste lineárně s celkovým počtem bloků zprávy [Ri97].

## 2 Srovnání protokolů

### 2.1 Vlastnosti

Srovnání v tabulkách 1 a 2 zachycuje vlastnosti s následujícím významem:

- **Počet APDU** – celkový počet APDU příkazů, které je třeba vyměnit s kartou pro dokončení zašifrování jednoho souboru. Délka vstupních a výstupních bufferů je nejvýše dvojnásobek délky bloku dat použité blokové šifry (16B pro AES128).
- **Typ hostitelských operací** – typ operací, které jsou vykonány na hostitelské straně (šifr. – blokový šifrovací algoritmus, stream – algoritmus pro tvorbu proudu klíčů, hash – blokový hashovací algoritmus, rng – generátor náhodných čísel)
- **Počet šifrovacích operací** – počet volání šifrovací algoritmu na kartě.
- **Závislý klíč –** závislost klíče šifrovacího algoritmu na kartě na vstupních datech. Při závislém klíči je třeba provádět přípravu nového klíče pro každý soubor (key scheduling).
- **Paralel. šifrování/dešifrování –** udává možnost paralelního vykonávání hostitelských operací v době, kdy probíhají operace prováděné na kartě při šifrování nebo dešifrování souboru.
- **Prodloužení délky zprávy** - udává velikost prodloužení vstupních dat v blocích.

[†] Je závislé na délce použité hashovací funkce.

| | RKEP | RaMaRK | I-RaMaRK | P-RKES | ARK |
|---|---|---|---|---|---|
| **Základní model útočníka** | zranitelný[1] | bezpečný | bezpečný | bezpečný | bezpečný |
| **Silný BFN model útočníka** | zranitelný[1] | zranitelný[2] | bezpečný | bezpečný | bezpečný |
| **Počet APDU** | 1 | 2 | 2 | 2 | 2 |
| **Typ hostitelských operací** | šifr., hash | stream, hash | stream, hash | šifr. | stream, hash |
| **Počet šifrovacích operací** | 2 | 6 | 6 | 6 | 4 |
| **Závislý klíč** | ne | ne | ne | ano/2x | ne |
| **Paralel. šifrování** | ne | část. | část. | ne | ne |
| **Paralel. dešifrování** | ne | část. | část. | ne | ne |
| **Prodloužení délky zprávy** | ne | ne | ne | ne | ne |

**Tabulka 1. RKE protokoly zachovávající délku zprávy**

| | THCEP | I1 | LI-SRKE | SAES |
|---|---|---|---|---|
| **Základní model útočníka** | bezpečný | bezpečný | bezpečný | bezpečný |
| **Silný BFN model útočníka** | zranitelný[3] | zranitelný[4] | bezpečný | bezpečný |
| **Počet APDU** | 1 | 1 | 1 | 1 |
| **Typ hostitelských operací** | rng, šifr. | rng, šifr. | šifr. hash | rng, lp-hash |
| **Počet šifrovacích operací** | 1 | 1 | 3 | 1 |
| **Závislý klíč** | ne | ne | ne | ne |
| **Paralel. šifrování** | ne | ano | ne | část. |
| **Paralel. dešifrování** | ne | ne | ne | ne |
| **Prodloužení délky zprávy** | 1 blok | 1 blok | 2 bloky | 2 bloky |

**Tabulka 2. RKE protokoly prodlužující délku zprávy**

## 2.2 Výkonnostní charakteristiky

Výkonnostní testy byly provedeny na počítači s procesorem AMD Athlon 2500+ (Burton), 1GB operační paměti, MS Windows 2000. Karetní část je realizována pomocí kryptografické čipové karty Gemplus GXPLite-Generic připojené pomocí USB čtečky GemPC Twin. Zdrojové kódy jsou psány v jazyce C++ a JavaCard. Pro šifrování dat v hostitelské části byl použit algoritmus AES ve verzi se 128 bitovým klíčem v režimu CBC. Pro hashování dat v hostitelské části byl použit algoritmus MD5. Pro konzistenci srovnání protokolů byl algoritmus pro tvorbu proudů klíčů nahrazen šifrováním v režimu CBC. Na kartě byl z důvodu nedostupnosti čipové karty s implementací algoritmu AES použit algoritmus DES. Z důvodu rozdílné délky bloků AES a DES (128 ku 64 bitům) bylo na zašifrování jednoho bloku algoritmu AES použito dvou volání algoritmu DES aplikovaných v režimu CBC. Klíčované náhodné funkce byli realizovány pomocí algoritmu DES v režimu CBC.

---

[1] Útoky v [Lu97].

[2] Útok v [BFN98].

[3] Není bezpečný proti inverznímu útoku (“inversion secure”) ani podvrhnutému vstupu (“forgery secure”).

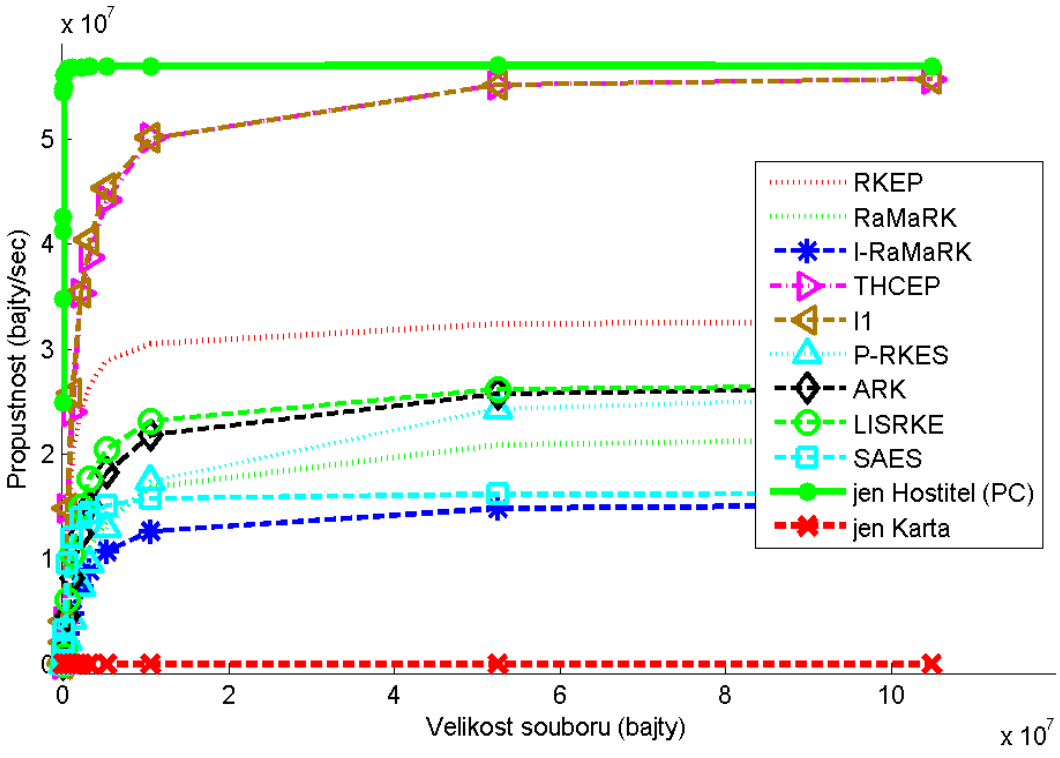
[4] Není bezpečný proti útoku podvrhnutým vstupem (“forgery secure”).

Propustnost jednotlivých protokolů byla určena průměrem z 30 opakovaných měření pro každou zkoumanou délku vstupních dat, provedených po jednom počátečním volání pro odstranění nedeterministických efektů spojených s prvním volání (inicializace systémových struktur). Před počátkem měření je vytvořeno spojení s čipovou kartou a vybrán aktivní applet, vyměňují se tedy pouze APDU příkazy bezprostředně související s protokolem. Příprava použitých klíčů na hostitelské straně je součástí měření a nebyla prováděna předem. Šifrovací klíče na karetní straně jsou připraveny předem, pokud nejsou závislé na vstupních datech. Paralelizace karetní a hostitelské části nebyla záměrně provedena ani u protokolů, které toto umožňují pro odstranění vlivů souvisejícími s tímto procesem[†]. Diskuze k dopadům paralelizace je uvedena u shrnutí pro středně velké soubory.

**Obrázek** 2 zachycuje celkové přehled propustnosti jednotlivých protokolů pro samostatné zprávy s délkami 32 bajtů až 100 megabajtů. Tučnou čárou je zvýrazněna propustnost při použití šifrování pouze na hostitelské straně a při šifrování pouze na čipové kartě. Počáteční, téměř lineární závislost (viz podrobněji **Obrázek** 3) je způsobena výrazně převyšující dobou zpracování na kartě oproti době zpracování v hostitelské části. Doba zpracování na kartě je nezávislá na celkové délce šifrované zprávy a představuje tak fixní výkonnostní penalizaci výraznou především pro krátké zprávy.

Výsledky pro krátké zprávy mezi 32 bajty až 8kB zachycuje **Obrázek** 3. Je zde patrný výrazný výkonnostní odstup protokolů RKEP, THCEP, I1 a SAES (v grafu splývají) s jedním APDU příkazem oproti protokolům vyžadujícím výměnu dvou APDU příkazů. Protokol LI-SRKE s jedním APDU příkazem je penalizován dvojnásobnou aplikací hashovací funkce na vstupní data. Nižší výkon protokolu P-RKES oproti I-RaMaRK je způsoben nutností provádět na kartě přípravu klíče závislého na šifrovaných datech.

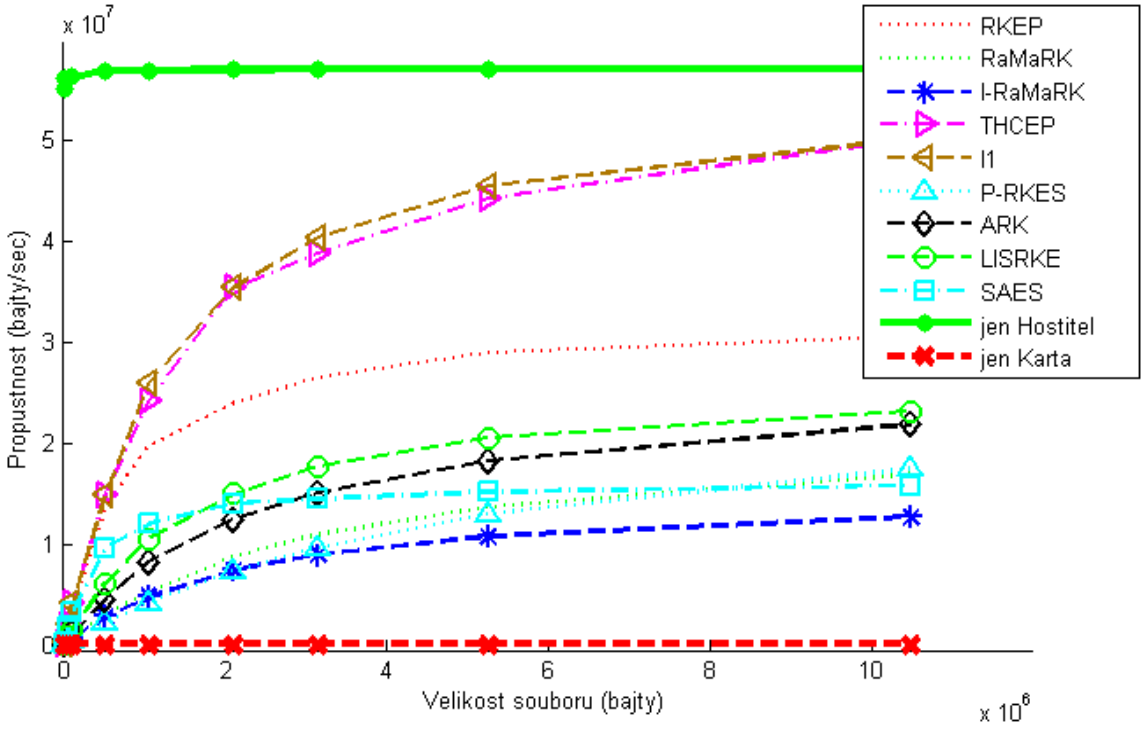
**Obrázek** 4 zachycuje oblast přechodu délky zpráv, u kterých přestává být určujícím faktorem pro propustnost rychlost karty a začíná být zanedbatelná vzhledem k času vyžadovaném hostitelskou částí. Při zvyšování výkonu karty se bude oblast přechodu přesouvat směrem k menším délkám zpráv, při zvyšování výkonu hostitelské části naopak. Je vhodné upozornit, že na výkon karty má významný vliv rychlost komunikačního rozhranní, nejen rychlost implementace šifrovacího algoritmu. Na použité čipové kartě Gemplus GXPLite-Generic pro rychlé protokoly typu THCEP a I1 tvořil podíl času věnovanému provedení šifrovacího algoritmu na kartě jen 25 %, zbylá část připadá na komunikační režii.




**Obrázek 2: Propustnost protokolů 32B-100MB (B/s).**

[†] Start a synchronizace vláken atp.

**Obrázek 3: Propustnost protokolů 32B-8kB (B/s).**




**Obrázek 4: Propustnost protokolů 8kB-10MB (B/s).**

### 2.3 Výkonnostní srovnání

Výsledky závisí na použitém hardwaru na straně čipové karty i hostitelské části (PC). Konkrétní hodnoty velikosti malých, středních a velkých souborů je nutné určit na základě testů pro použitý hardware. Celkově lze výsledky shrnout takto:

- Pro *malé soubory (<100kB)* jsou zanedbatelné operace hostitelské části. Výkonnostní výhodu mají protokoly s jedinou zprávou z důvodu velkého časového podílu karty na zpracování zprávy. Protokoly s jednou zprávou navíc umožňují snadno implementovat zisk více odpovědí v rámci jednoho APDU pro souběžné zpracování více nezávislých souborů pro úsporu na komunikační režii. Výhodné jsou protokoly zachovávající délku zprávy a protokoly, které nevyžadují časově náročnou přípravu klíče na kartě (*key schedule*).
- Pro *středně velké soubory (100kB – 2MB)* je výhodná možnost souběžného vykonávání karetní a hostitelské části, neboť vliv obou časté na celkovém výsledku je výrazný. Tuto možnost poskytují protokoly I-RaMaRK, I1 a SAES (poslední dva jen pro šifrování). Výhodné jsou protokoly s jednou zprávou obdobně jako pro malé soubory. Vhodnost použití paralelizace lze teoreticky odhadnout podle následujícího vzorce, který udává podmínky pro velikost souboru tak, aby dosaženo alespoň *X %* zrychlení: *(čas_karet_oper_sec * čas_host_oper_sec) / velikost_souboru > (X / 100)*
- Pro *velké soubory (>> 2MB)* jsou zanedbatelné operace karty. Pro zrychlení celkové propustnosti lze paralelizovat průběh hostitelských operací, např. pomocí CTR módu [CTR] pro šifrování[†].

[†] Nesouvisí přímo s RKE protokoly, lze provést i pro ostatní velikosti souborů. Pro velké soubory však bude mít největší přínos.

# Závěr

Zpráva poskytuje přehled publikovaných RKE protokolů (RKEP, RaMaRK, I-RaMaRK, I1, THCEP, P-RKES, ARK, LI-SRKE, SAES), srovnává jejich významné návrhové charakteristiky, výkon při použití pro šifrování vybrané délek vstupních dat.

Volba vhodného RKE protokolu je závislá na předpokládanému scénáři použití, velikosti zpracovávaných dat a očekávanému útočníkově modelu. Protokoly RKEP a RaMaRK by neměly být použity z důvodu existujících útoků. Protokoly THCEP a I1 poskytují největší propustnost, zároveň by však měly být použity pouze v prostředí, kde je nepravděpodobný přístup útočníka ke kartě a není vyžadována bezpečnost v rámci silného BFN modelu. Pro silný BFN model se jako nejvhodnější jeví protokol SAES pro velikosti souborů do cca 2MB a protokoly LI-SRKE a ARK pro soubory větší než 2MB. Konkrétní přechodová hodnota závisí na rychlosti použité čipové karty a výkonu hostitelského počítače. Významným faktorem určujícím výkon karty je rychlost komunikačního rozhranní.

Z důvodu nedostupnosti vhodné čipové karty s implementací algoritmu AES byl na karetní straně využit algoritmus DES. Dle srovnání rychlostí obou algoritmů na shodné hardwarové platformě [GCh01, SM03] lze předpokládat zrychlení karetních operací, významným faktorem však stále zůstane rychlost komunikačního rozhranní. Největší přínos by měl být pro protokoly s vyšším počtem šifrovacích operací prováděných na kartě (ARK, I-RaMaRK a P-RKES).

# Literatura


[Bl96] Matt Blaze, High-Bandwith Encryption with Low-Bandwith Smartcards, Proceedings of the Fast Software Encryption Workshop, LNCS 1039, Springer, Berlin, s. 33-40, 1996.

[BFN98] Matt Blaze, Joan Feigenbaum, Moni Naor: A Formal Treatment of Remotely Keyed Encryption, Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms, s. 868-869, 1999.

[BWL00] Baastian Bakker, Rudigger Weis, Stefan Lucks: How to Ring A Swan, Adding Tamper Resistant Authentication to Linux IPSec, SANE 2000 http://www.cryptolabs.org/vpn/BakkerWeisLucksIPSecSANE2000.ps.gz (10/2005).

[CTR] Block cipher mode of Operation, Counter mode. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf. (10/2005)

[GCh01] Kris Gaj, Pawel Chodowiec: Fast Implementation and Fair Comparison of the Final Candidates for AES Using FPGA, Springer LNCS 2020, Berlin, 2001, s. 84-99.

[JSY99] Markus Jakobsson, Julien P. Stern, Moti Yung: Scramble All, Encrypt Small, FSE'99, LNCS 1636, s. 95-111, 1999.

[Lu97] Stefan Lucks: On the Security of Remotely Keyed Encryption, Proceedings of Fast Software Encryption Workshop 1997, LNCS 1267, s. 219 – 229, 1997.

[Lu99] Stefan Lucks: Accelerated Remotely Keyed Encryption, Fast Software Encyption '99, LNCS 1636, 1999.

[SM03] Akashi Satoh, Sumio Morioka: Hardware-Focused Performance Comparison for the Standard Block Ciphers AES, Camellia, and Triple-DES. Springer LNCS 2851, Berlin, 2003, s. 252-266.

[We00] Rudigger Weis: A Trivial Host Card Protocol. http://www.cryptolabs.org/thc/cryptolabstrthc.ps.gz (10/2005)

[Ri97] Ronald L. Rivest: All-Or-Nothing Encryption and The Package Transform, in Proceedings of the Fast Software Encryption Workshop, LNCS 1267, s. 210-218, 1997.

[SSR00] Sang Uk Shin, Weon Shin, Kyung Hyune Rhee: All-or-Nothing Transform and Remotely Keyed Encryption Protocols, PKC 2000, LNCS 1751, s. 178-195, 2000.

## Příloha A – Formální zápis jednotlivých protokolů

Pro zápis šifrovacího směru RKE protokolů je použito následující označení. $P_i$ resp. $C_i$ značí i-tý blok otevřeného resp. zašifrovaného textu. $P_1$, …, $P_n$ značí bloky 1 až n zprávy P. $I_i$ značí i-tý blok hodnoty průběžného výpočtu. H značí blokovou hashovací funkci s fixním výstupem (např. MD5, SHA-1). LPH značí hashovací funkci s délkou výstupu odpovídajícímu délce vstupu. Příkladem takové funkce může být $LPH(X_{1,\ldots,n}) = \aleph_1(X_1) \parallel \aleph_2(X_2) \parallel \ldots \parallel \aleph_{n/(l-1)}(X_{n/(l-1)})$; $\aleph_i(x) = H(tag \parallel i \parallel H(x))$. E značí blokovou šifrovací funkci (např. DES, AES). Operace bitového exklusivního součtu je značena jako ⊕. Šifrovací klíč je značen písmenem K a $K_p$, s případným číselným indexem $K_i$. Generátor náhodných čísel je značen jako RND, náhodný řetězec jako R. Klíčovaná náhodná funkce je značena jako F. Funkce pro generování proudu klíčů je značena jako S.

**RKEP C = [$C_1$, $C_2$, …, $C_n$]**

1. PC: $I_i = P_i \oplus H(P_1)$, $i \in \{2, \ldots, n\}$; $I_1 = P_1 \oplus H(I_2, \ldots, I_n)$;
2. PC-> SC: ($I_1$)
3. SC: $C_1 = E_K(I_1)$; $K_P = E_K(C_1)$;
4. SC->PC: ($C_1$, $K_P$)
5. PC: $C_2 = E_{KP}(I_2 \oplus C_1)$; …; $C_n = E_{KP}(I_n \oplus C_{n-1})$;

**RaMaRK C = [$C_1$, $C_2$, $C_{3,\ldots,n}$]**

1. PC->SC: ($P_1$, $P_2$)
2. SC: $U = E_{K1}(P_1) \oplus P_2$; $T = E_{K2}(U) \oplus P_1$; $X = E_{K3}(T) \oplus U$; ulož T;
3. SC->PC: (X)
4. PC: $I = S(X) \oplus R$, $Y = H(I)$; $C_{3,\ldots,n} = S(Z) \oplus I$;
5. PC->SC: (Z)
6. SC: $V = E_{K4}(T) \oplus Z$; $C_1 = E_{K5}(V) \oplus T$; $C_2 = E_{K6}(A) \oplus V$;
7. SC->PC: ($C_1$, $C_2$);

**I-RaMaRK C = [$C_1$, $C_2$, $C_{3,\ldots,n}$]**

1. PC: $P_1 = P_1 \oplus H(P_{3,\ldots,n})$;
2. PC->SC: ($P_1$, $P_2$)
3. SC: $U = E_{K1}(P_1) \oplus P_2$; $T = E_{K2}(U) \oplus P_1$; $X = E_{K3}(T) \oplus U$; ulož T;
4. SC->PC: (X)
5. PC: $I = S(X) \oplus R$, $Y = H(I)$; $C_{3,\ldots,n} = S(Z) \oplus I$;
6. PC->SC: (Z)
7. SC: $V = E_{K4}(T) \oplus Z$; $C_1 = E_{K5}(V) \oplus T$; $C_2 = E_{K6}(A) \oplus V$;
8. SC->PC: ($C_1$, $C_2$);
9. PC: $C1 = C1 \oplus H(C_{3,\ldots,n})$;

**I1 C = [R, $C_{1,\ldots,n}$]**

1. PC: $T = RNG()$; $C_{1,\ldots,n} = E_T(P_{1,\ldots,n})$;
2. PC->SC: (T)
3. SC: $R = E_K(T)$;
4. SC->PC: (R)

**THCEP C = [R, $C_{1,...,n}$]**

1. PC: R = RNG();
2. PC->SC: (R)
3. SC: $K_P = E_K(R)$;
4. SC->PC: $C_{1,...,n} = E_{KP}(P_{1,...,n})$;

**P-RKES C = [$C_1$, $C_{2,...,n}$]**

1. PC: $X = H(P_{2,...,n})$;
2. PC->SC: ($P_1$, X)
3. SC: $Ka = F_{K1}(X)$; $W = E_{Ka}(P_1)$; $Y = E_{K2}(W)$; $S = F_{K3}(Y)$; ulož Y;
4. SC->PC: (S)
5. PC: $C_{2,...,n} = E_S(P_{2,...,n})$; $Z = H(C_{2,...,n})$;
6. PC->SC: (Z)
7. SC: $K_b = F_{K4}(Z)$; $C_1 = E_{Kb}(Y)$;
8. SC->PC: ($C_1$)

**ARK C = [$C_1$, $C_{2,...,n}$]**

1. PC: $X = H(P_{2,...,n})$;
2. PC->SC: ($P_1$, X)
3. SC: $W = F_{K1}(X)$; $Y = E_{K2}(P_1) \oplus W$; ulož Y;
4. SC->PC: (Y)
5. PC: $C_{2,...,n} = S(Y) \oplus (P_{2,...,n})$; $Z = H(C_{2,...,n})$;
6. PC->SC: (Z)
7. SC: $A = F_{K2}(Z) \oplus Y$; $C_1 = E_{K3}(A)$;
8. SC->PC: ($C_1$)

**LI-SRKE C = [t, $C_0$, $C_{1,...,n}$]**

1. PC: $K = H(P_{1,...,n})$; $C_{1,...,n} = E_K(P_{1,...,n})$; $X = H(C_{1,...,n})$;
2. PC->SC: (K, X)
3. SC: $W = F_{K1}(K)$; $Y = F_{K2}(X)$; $C_0 = E_{K3}(K \oplus Y)$; $t = W \oplus Y$;
4. SC->PC: (t, $C_0$)

**SAES C = [Ca || Cb‘]**

1. PC: U = RNG(); $X = H(U, P_{1,...,n})$; Ma = první_polovina(U || X || $P_{1,...,n}$); Mb = druhá_polovina(U || X || $P_{1,...,n}$); Cb = Mb ⊕ LPH(Ma); Ca = Ma ⊕ LPH(Cb); R = posledních_k_bitů(Cb);
2. PC->SC: (R)
3. SC: $K = E_{K1}(R)$;
4. SC->PC: (K)
5. PC: q = velikost(K); nahraď posledních q bitů z Cb za K => Cb‘;