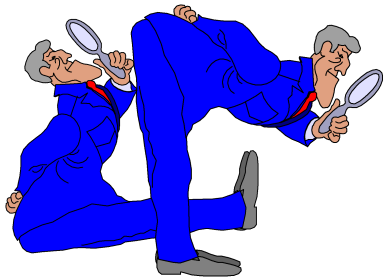
7. Asymetrické šifrovací systémy neboli systémy s veřejným klíčem RSA-algoritmus

Jan Paseka

Ústav matematiky a statistiky
Masarykova univerzita

23. listopadu 2023

O čem to bude



Úvod

Připomeňme dvě zásadní tvrzení z teorie čísel.

Úvod

Připomeňme dvě zásadní tvrzení z teorie čísel.

Věta 1.1

(Euler) *Nechť* $(c, m) = 1$. *Pak platí* $c^{\varphi(m)} = 1 \bmod m$.

Úvod

Připomeňme dvě zásadní tvrzení z teorie čísel.

Věta 1.1

(Euler) *Nechť* $(c, m) = 1$. *Pak platí* $c^{\varphi(m)} = 1 \bmod m$.

Věta 1.2

(Fermat) *Nechť p je prvočíslo* $(c, p) = 1$. *Pak platí* $c^{p-1} = 1 \bmod p$.

Postup při šifrování RSA-algoritmu I

1. Najděme dvě "velká" prvočísla p a q a položme $n = p \cdot q$.

Postup při šifrování RSA-algoritmu I

1. Najděme dvě "velká" prvočísla p a q a položme $n = p \cdot q$.
2. Najděme "velké a náhodné" přirozené číslo d tak, že je nesoudělné s číslem $(p-1) \cdot (q-1)$.

Postup při šifrování RSA-algoritmu I

1. Najděme dvě "velká" prvočísla p a q a položme $n = p \cdot q$.
2. Najděme "velké a náhodné" přirozené číslo d tak, že je nesoudělné s číslem $(p-1) \cdot (q-1)$.
3. Vypočtěme jediné přirozené číslo e ležící v oboru hodnot $1 \leq e \leq (p-1) \cdot (q-1)$ ze vztahu

$$e \cdot d = 1 \bmod (p-1) \cdot (q-1).$$

4. Zveřejněme veřejný klíč, který se skládá z dvojice přirozených čísel (e, n).

Postup při šifrování RSA-algoritmu I

1. Najděme dvě "velká" prvočísla p a q a položme $n = p \cdot q$.
2. Najděme "velké a náhodné" přirozené číslo d tak, že je nesoudělné s číslem $(p-1)\cdot(q-1)$.
3. Vypočtěme jediné přirozené číslo e ležící v oboru hodnot $1 \leq e \leq (p-1)\cdot(q-1)$ ze vztahu
$$e \cdot d = 1 \bmod (p-1)\cdot(q-1).$$
4. Zveřejněme veřejný klíč, který se skládá z dvojice přirozených čísel (e, n).
5. Reprezentujme zprávu M jako přirozené číslo z intervalu $\{1, \ldots, n\}$; rozdělme zprávu M do bloků, je-li příliš velká.

Postup při šifrování RSA-algoritmu I

1. Najděme dvě "velká" prvočísla p a q a položme $n = p \cdot q$.
2. Najděme "velké a náhodné" přirozené číslo d tak, že je nesoudělné s číslem $(p-1)\cdot(q-1)$.
3. Vypočtěme jediné přirozené číslo e ležící v oboru hodnot $1 \leq e \leq (p-1)\cdot(q-1)$ ze vztahu
$$e \cdot d = 1 \bmod (p-1)\cdot(q-1).$$
4. Zveřejněme veřejný klíč, který se skládá z dvojice přirozených čísel (e, n).
5. Reprezentujme zprávu M jako přirozené číslo z intervalu $\{1, \ldots, n\}$; rozdělme zprávu M do bloků, je-li příliš velká.
6. Zakódujme M do kryptogramu C dle předpisu
$$C = M^e \bmod n.$$
7. Dešifrujme pomocí soukromého klíče d a předpisu
$$D = C^d \bmod n.$$

Postup při šifrování RSA-algoritmu II

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

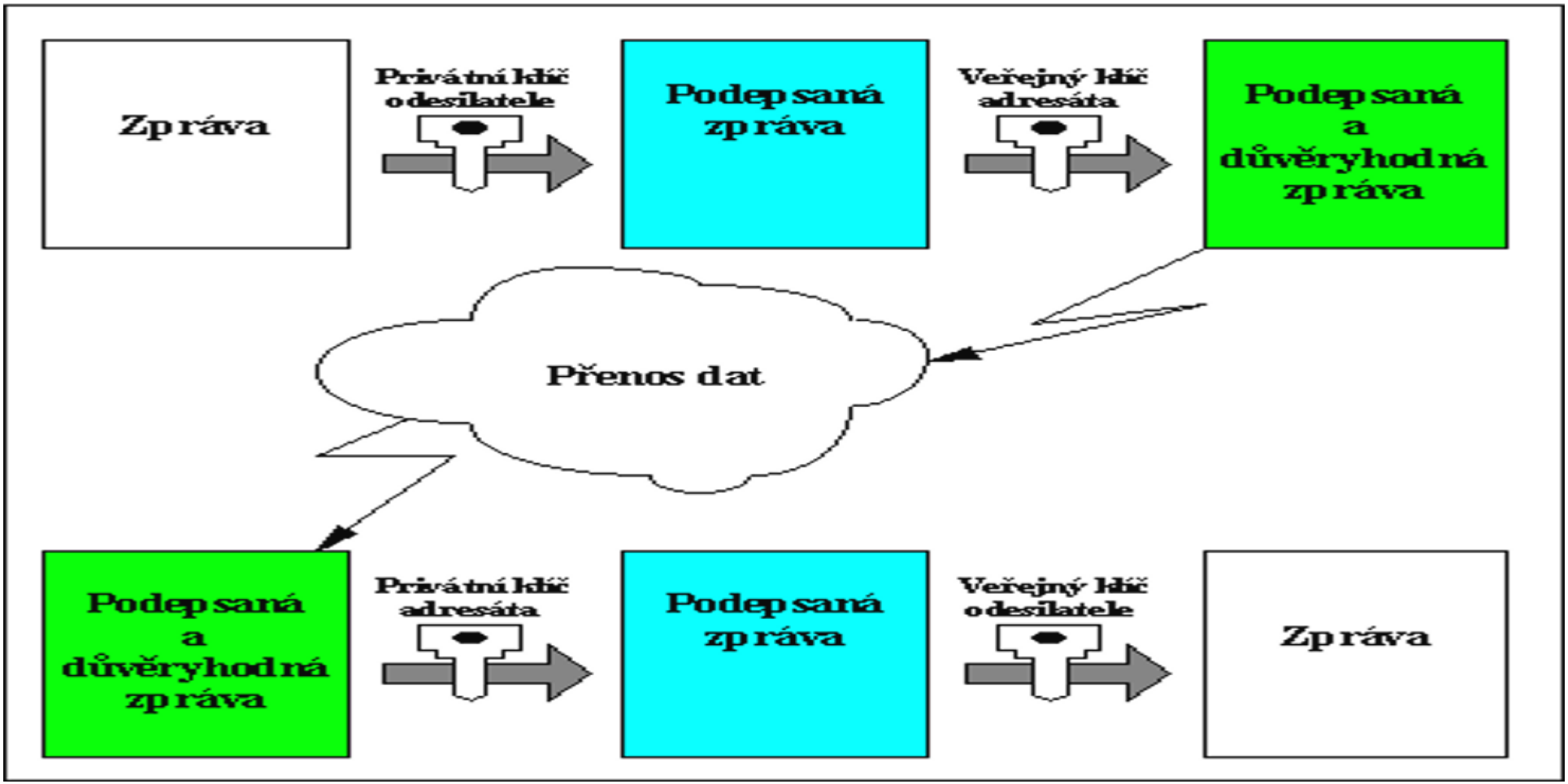
Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

Digitální podpis



RSA-podpisovací schéma I

Označme veřejný klíč uživatele I dvojici (e_I, n_I) a soukromý klíč d_I.

RSA-podpisovací schéma I

Označme veřejný klíč uživatele I dvojici (e_I, n_I) a soukromý klíč d_I.

V praxi je n_I obvykle voleno jako číslo, které je součin dvou náhodně zvolených asi 100-místných prvočísel (300-600 bitů), $n_I = p_I \cdot q_I$.

RSA-podpisovací schéma I

Označme veřejný klíč uživatele I dvojici (e_I, n_I) a soukromý klíč d_I.

V praxi je n_I obvykle voleno jako číslo, které je součin dvou náhodně zvolených asi 100-místných prvočísel (300-600 bitů), $n_I = p_I \cdot q_I$.

Prvočísla p_I a q_I jsou osobním tajemstvím uživatele I.

RSA-podpisovací schéma I

Označme veřejný klíč uživatele I dvojici (e_I, n_I) a soukromý klíč d_I.

V praxi je n_I obvykle voleno jako číslo, které je součin dvou náhodně zvolených asi 100-místných prvočísel (300-600 bitů), $n_I = p_I \cdot q_I$.

Prvočísla p_I a q_I jsou osobním tajemstvím uživatele I.

Dále si uživatel I zvolí tzv. šifrovací exponent e_I tak, aby byl nesoudělný s $\varphi(n_I)$.

RSA-podpisovací schéma I

Označme veřejný klíč uživatele I dvojici (e_I, n_I) a soukromý klíč d_I.

V praxi je n_I obvykle voleno jako číslo, které je součin dvou náhodně zvolených asi 100-místných prvočísel (300-600 bitů), $n_I = p_I \cdot q_I$.

Prvočísla p_I a q_I jsou osobním tajemstvím uživatele I.

Dále si uživatel I zvolí tzv. šifrovací exponent e_I tak, aby byl nesoudělný s $\varphi(n_I)$.

Zejména tedy platí, že $(e_I, (p_I - 1) \cdot (q_I - 1)) = 1$.

RSA-podpisovací schéma I

Označme veřejný klíč uživatele I dvojici (e_I, n_I) a soukromý klíč d_I.

V praxi je n_I obvykle voleno jako číslo, které je součin dvou náhodně zvolených asi 100-místných prvočísel (300-600 bitů), $n_I = p_I \cdot q_I$.

Prvočísla p_I a q_I jsou osobním tajemstvím uživatele I.

Dále si uživatel I zvolí tzv. šifrovací exponent e_I tak, aby byl nesoudělný s $\varphi(n_I)$.

Zejména tedy platí, že $(e_I, (p_I - 1) \cdot (q_I - 1)) = 1$.

Uživatel I najde číslo d_I tak, že splňuje $e_I \cdot d_I = 1 \bmod \varphi(n_I)$. Toto číslo je výše uvedenými hodnotami jednoznačně určeno.

RSA-podpisovací schéma II

Pak šifrovací předpis pro odesílatele A, který chce zaslat příjemci B podepsanou zprávu M následovně:

RSA-podpisovací schéma II

Pak šifrovací předpis pro odesílatele A, který chce zaslat příjemci B podepsanou zprávu M následovně:

1. Očíslujme po řadě písmena latinské abecedy A=01 B=02, . . . , Z=26. V praxi se používá desítkové vyjádření v ASCII kódu. Text, který chceme utajit, převedeme do číselné formy a rozdělíme na bloky stejné délky. Číselné vyjádření bloku B označíme M. Přitom požadujeme, aby $1 \leq M < n_A$.

RSA-podpisovací schéma II

Pak šifrovací předpis pro odesílatele A, který chce zaslat příjemci B podepsanou zprávu M následovně:

1. Očíslujme po řadě písmena latinské abecedy A=01 B=02, . . . , Z=26. V praxi se používá desítkové vyjádření v ASCII kódu. Text, který chceme utajit, převedeme do číselné formy a rozdělíme na bloky stejné délky. Číselné vyjádření bloku B označíme M. Přitom požadujeme, aby $1 \leq M < n_A$.
2. Odesílatel A vypočte podpis S jako

$$S = M^{d_A} (\mathrm{mod}\, n_A).$$

RSA-podpisovací schéma II

Pak šifrovací předpis pro odesílatele A, který chce zaslat příjemci B podepsanou zprávu M následovně:

1. Očíslujme po řadě písmena latinské abecedy A=01 B=02, ..., Z=26. V praxi se používá desítkové vyjádření v ASCII kódu. Text, který chceme utajit, převedeme do číselné formy a rozdělíme na bloky stejné délky. Číselné vyjádření bloku B označíme M. Přitom požadujeme, aby $1 \leq M < n_A$.
2. Odesílatel A vypočte podpis S jako
$$S = M^{d_A} (\mathrm{mod}\, n_A).$$
3. Pak A vypočte kryptogram
$$C = S^{e_B} (\mathrm{mod}\, n_B).$$

RSA-podpisovací schéma III

4. Po obdržení kryptogramu C vypočte B podpis

$$S = C^{d_B} (\mathrm{mod}\, n_B).$$

RSA-podpisovací schéma III

4. Po obdržení kryptogramu C vypočte B podpis

$$S = C^{d_B} (\mathrm{mod}\, n_B).$$

5. Dále B vypočte zprávu

$$M = S^{e_A} (\mathrm{mod}\, n_A).$$

Korektnost RSA-algoritmu I

Lemma 1.3

Pro všechna vhodně zvolená M platí

$$S = M^{d_A} (\mathrm{mod}\ n_A)$$
$$M = S^{e_A} (\mathrm{mod}\ n_A).$$

Korektnost RSA-algoritmu I

Lemma 1.3

Pro všechna vhodně zvolená M platí

$$S = M^{d_A} (\mathrm{mod}\ n_A)$$
$$M = S^{e_A} (\mathrm{mod}\ n_A).$$

Důkaz Můžeme bez újmy na obecnosti předpokládat, že $(M, n_A) > 1$.

Korektnost RSA-algoritmu I

Lemma 1.3

Pro všechna vhodně zvolená M platí

$$S = M^{d_A} (\mathrm{mod}\ n_A)$$
$$M = S^{e_A} (\mathrm{mod}\ n_A).$$

Důkaz Můžeme bez újmy na obecnosti předpokládat, že $(M, n_A) > 1$.

Dle předpokladu RSA-algoritmu $e_A \cdot d_A = (q_A - 1) \cdot c + 1$ pro vhodné číslo c.

Korektnost RSA-algoritmu I

Lemma 1.3

Pro všechna vhodně zvolená M platí

$$S = M^{d_A} (\mathrm{mod}\ n_A)$$
$$M = S^{e_A} (\mathrm{mod}\ n_A).$$

Důkaz Můžeme bez újmy na obecnosti předpokládat, že $(M, n_A) > 1$.

Dle předpokladu RSA-algoritmu $e_A \cdot d_A = (q_A - 1) \cdot c + 1$ pro vhodné číslo c.

Z Fermatovy věty máme

$$p_A^{e_A \cdot d_A - 1} = (p_A^{q_A - 1})^c = 1 \bmod q_A.$$

Korektnost RSA-algoritmu I

Lemma 1.3

Pro všechna vhodně zvolená M platí

$$S = M^{d_A} (\mathrm{mod}\ n_A)$$
$$M = S^{e_A} (\mathrm{mod}\ n_A).$$

Důkaz Můžeme bez újmy na obecnosti předpokládat, že $(M, n_A) > 1$.

Dle předpokladu RSA-algoritmu $e_A \cdot d_A = (q_A - 1) \cdot c + 1$ pro vhodné číslo c.

Z Fermatovy věty máme

$$p_A^{e_A \cdot d_A - 1} = (p_A^{q_A - 1})^c = 1 \bmod q_A.$$

Po vynásobení číslem p_A máme

$$p_A^{e_A \cdot d_A} = p_A \bmod p_A \cdot q_A.$$

Korektnost RSA-algoritmu II

Pokračování důkazu Lemmatu 1.3.

Je-li $(M, n_A) > 1$, je M dělitelné buď p_A nebo q_A. Nechť např. $M = a \cdot p_A$, $1 \leq a < q_A$. Pak

$$(M^{d_A})^{e_A} = M^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot {p_A}^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot p_A \bmod p_A \cdot q_A.$$

Korektnost RSA-algoritmu II

Pokračování důkazu Lemmatu 1.3.

Je-li $(M, n_A) > 1$, je M dělitelné buď p_A nebo q_A. Nechť např. $M = a \cdot p_A$, $1 \leq a < q_A$. Pak

$$(M^{d_A})^{e_A} = M^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot {p_A}^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot p_A \bmod p_A \cdot q_A.$$

Protože $(a, q_A) = 1$, máme dle Eulerovy věty

$$a^{e_A \cdot d_A} = a \bmod q_A.$$

Korektnost RSA-algoritmu II

Pokračování důkazu Lemmatu 1.3.

Je-li $(M, n_A) > 1$, je M dělitelné buď p_A nebo q_A. Nechť např. $M = a \cdot p_A$, $1 \leq a < q_A$. Pak

$$(M^{d_A})^{e_A} = M^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot {p_A}^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot p_A \bmod p_A \cdot q_A.$$

Protože $(a, q_A) = 1$, máme dle Eulerovy věty

$$a^{e_A \cdot d_A} = a \bmod q_A.$$

Po vynásobení číslem p_A máme

$$(M^{d_A})^{e_A} = p_A \cdot a^{e_A \cdot d_A} = p_A \cdot a = M \bmod p_A \cdot q_A.$$

Korektnost RSA-algoritmu II

Pokračování důkazu Lemmatu 1.3.

Je-li $(M, n_A) > 1$, je M dělitelné buď p_A nebo q_A. Nechť např. $M = a \cdot p_A$, $1 \leq a < q_A$. Pak

$$(M^{d_A})^{e_A} = M^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot {p_A}^{e_A \cdot d_A} = a^{e_A \cdot d_A} \cdot p_A \bmod p_A \cdot q_A.$$

Protože $(a, q_A) = 1$, máme dle Eulerovy věty

$$a^{e_A \cdot d_A} = a \bmod q_A.$$

Po vynásobení číslem p_A máme

$$(M^{d_A})^{e_A} = p_A \cdot a^{e_A \cdot d_A} = p_A \cdot a = M \bmod p_A \cdot q_A.$$

Pro $M = b \cdot q_A$, $1 \leq b < p_A$ se důkaz provede analogicky.
Pokud $(M, n_A) = 1$, pak z Eulerovy věty máme

$$(M^{d_A})^{e_A} = M^{d_A \cdot e_A} = M^{1 + k\varphi(n_A)} = M \bmod p_A \cdot q_A.$$

Korektnost RSA-algoritmu III

Poznamenejme, že odhlédneme-li od nutného požadavku na komutování šifrovací a dešifrovací funkce, musí být nutně podpis S vypočtený odesílatelem A v definičním oboru šifrovací procedury e_B.

Korektnost RSA-algoritmu III

Poznamenejme, že odhlédneme-li od nutného požadavku na komutování šifrovací a dešifrovací funkce, musí být nutně podpis S vypočtený odesílatelem A v definičním oboru šifrovací procedury e_B.

Tato poslední podmínka nemusí platit, když použitý systém je RSA; podpis S může být větší přirozené číslo, než je veřejný klíč n_B.

Korektnost RSA-algoritmu III

Poznamenejme, že odhlédneme-li od nutného požadavku na komutování šifrovací a dešifrovací funkce, musí být nutně podpis S vypočtený odesílatelem A v definičním oboru šifrovací procedury e_B.

Tato poslední podmínka nemusí platit, když použitý systém je RSA; podpis S může být větší přirozené číslo, než je veřejný klíč n_B.

Můžeme však zajistit platnost této podmínky tím, že přizpůsobíme velikost bloků naší zprávy tak, že výsledek padne do požadovaného definičního oboru.

Korektnost RSA-algoritmu IV

Příklad 1.4

Zvolme $(e_A, n_A) = (5, 35)$, $(e_B, n_B) = (3, 15)$, $M = 3$. *Pak* $d_A = 5$, $d_B = 3$.

Korektnost RSA-algoritmu IV

Příklad 1.4

Zvolme $(e_A, n_A) = (5, 35)$, $(e_B, n_B) = (3, 15)$, $M = 3$. *Pak* $d_A = 5$, $d_B = 3$.

Máme pak $S = 3^5 = 33 \ (mod 35)$, $C = 33^3 = 12 \ (mod 15)$.

Korektnost RSA-algoritmu IV

Příklad 1.4

Zvolme $(e_A, n_A) = (5, 35)$, $(e_B, n_B) = (3, 15)$, $M = 3$. *Pak* $d_A = 5$, $d_B = 3$.

Máme pak $S = 3^5 = 33 \ (mod 35)$, $C = 33^3 = 12 \ (mod 15)$.

B vypočte podpis $S' = 12^3 = 3 \ (mod 15)$ *a z něho zprávu* $M' = 3^5 = 33 \ (mod 35)$.

Korektnost RSA-algoritmu IV

Příklad 1.4

Zvolme $(e_A, n_A) = (5, 35)$, $(e_B, n_B) = (3, 15)$, $M = 3$. *Pak* $d_A = 5$, $d_B = 3$.

Máme pak $S = 3^5 = 33 \ (mod 35)$, $C = 33^3 = 12 \ (mod 15)$.

B vypočte podpis $S' = 12^3 = 3 \ (mod 15)$ *a z něho zprávu* $M' = 3^5 = 33 \ (mod 35)$.

Protože $3 \neq 33$, *není pro uživatele B splněna podmínka komutování šifrovací a dešifrovací funkce a M se nám zobrazí na zcela jinou zprávu* M'.

Korektnost RSA-algoritmu IV

Příklad 1.4

Zvolme $(e_A, n_A) = (5, 35)$, $(e_B, n_B) = (3, 15)$, $M = 3$. *Pak* $d_A = 5$, $d_B = 3$.

Máme pak $S = 3^5 = 33 \ (mod 35)$, $C = 33^3 = 12 \ (mod 15)$.

B vypočte podpis $S' = 12^3 = 3 \ (mod 15)$ *a z něho zprávu* $M' = 3^5 = 33 \ (mod 35)$.

Protože $3 \neq 33$, *není pro uživatele B splněna podmínka komutování šifrovací a dešifrovací funkce a M se nám zobrazí na zcela jinou zprávu* M'.

Pokud by však bylo $n_A < n_B$ *(zvolme např.* $(e_B, n_B) = (5, 35)$, $(e_A, n_A) = (3, 15)$, $M = 3$, $d_B = 5$, $d_A = 3$*), máme* $S = 3^3 = 12 \ (mod 15)$, $C = 12^5 = 17 \ (mod 35)$, $S' = 17^5 = 12 \ (mod 35)$, $M' = 12^3 = 3 \ (mod 15)$.

Korektnost RSA-algoritmu V

Rivest, Shamir a Adleman (1978) navrhli mnohem elegantnější řešení:

Je zvolena ***mezní hodnota*** h pro systém s veřejným klíčem (řekněme $h \sim 10^{199}$).

Korektnost RSA-algoritmu V

Rivest, Shamir a Adleman (1978) navrhli mnohem elegantnější řešení:
Je zvolena ***mezní hodnota*** h pro systém s veřejným klíčem (řekněme $h \sim 10^{199}$).

Každý uživatel pak má ***dvě*** dvojice veřejných klíčů, jednu pro zašifrování a druhou pro ověření podpisu.

Korektnost RSA-algoritmu V

Rivest, Shamir a Adleman (1978) navrhli mnohem elegantnější řešení:
Je zvolena ***mezní hodnota*** h pro systém s veřejným klíčem (řekněme $h \sim 10^{199}$).

Každý uživatel pak má ***dvě*** dvojice veřejných klíčů, jednu pro zašifrování a druhou pro ověření podpisu.

Označme je po řadě (e_l, n_l) a (f_l, m_l), kde l probíhá množinu uživatelů.

Korektnost RSA-algoritmu V

Rivest, Shamir a Adleman (1978) navrhli mnohem elegantnější řešení:
Je zvolena ***mezní hodnota*** h pro systém s veřejným klíčem (řekněme $h \sim 10^{199}$).

Každý uživatel pak má ***dvě*** dvojice veřejných klíčů, jednu pro zašifrování a druhou pro ověření podpisu.

Označme je po řadě (e_l, n_l) a (f_l, m_l), kde l probíhá množinu uživatelů.

Soukromý klíč odpovídající dvojici pro ověření podpisu budeme značit (d_l, g_l).

Korektnost RSA-algoritmu V

Rivest, Shamir a Adleman (1978) navrhli mnohem elegantnější řešení:
Je zvolena ***mezní hodnota*** h pro systém s veřejným klíčem (řekněme $h \sim 10^{199}$).

Každý uživatel pak má ***dvě*** dvojice veřejných klíčů, jednu pro zašifrování a druhou pro ověření podpisu.

Označme je po řadě (e_l, n_l) a (f_l, m_l), kde l probíhá množinu uživatelů.

Soukromý klíč odpovídající dvojici pro ověření podpisu budeme značit (d_l, g_l).

Zvolený předpis se řídí tím, že šifrovací modul n_l a podpisovací modul m_l by měly pro každého uživatele l splňovat

$$m_l < h < n_l.$$

Korektnost RSA-algoritmu VI

Počítáme pak následovně:

1. Odesílatel A vypočte podpis S jako

$$S = M^{g_A} (\mathrm{mod}\, m_A).$$

Korektnost RSA-algoritmu VI

Počítáme pak následovně:

1. Odesílatel A vypočte podpis S jako
$$S = M^{g_A} (\mathrm{mod}\, m_A).$$
2. Pak A vypočte kryptogram
$$C = S^{e_B} (\mathrm{mod}\, n_B).$$

Korektnost RSA-algoritmu VI

Počítáme pak následovně:

1. Odesílatel A vypočte podpis S jako
$$S = M^{g_A} (\mathrm{mod}\, m_A).$$
2. Pak A vypočte kryptogram
$$C = S^{e_B} (\mathrm{mod}\, n_B).$$
3. Po obdržení kryptogramu C vypočte B podpis
$$S = C^{d_B} (\mathrm{mod}\, n_B).$$

Korektnost RSA-algoritmu VI

Počítáme pak následovně:

1. Odesílatel A vypočte podpis S jako
$$S = M^{g_A} (\mathrm{mod}\, m_A).$$
2. Pak A vypočte kryptogram
$$C = S^{e_B} (\mathrm{mod}\, n_B).$$
3. Po obdržení kryptogramu C vypočte B podpis
$$S = C^{d_B} (\mathrm{mod}\, n_B).$$
4. Dále B vypočte zprávu
$$M = S^{f_A} (\mathrm{mod}\, m_A).$$

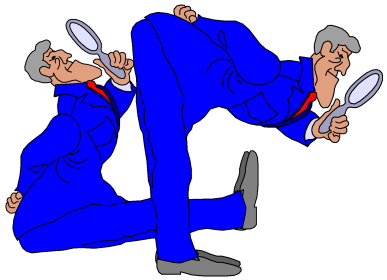
Korektnost RSA-algoritmu VII

Snadno se ověří, že tento systém opravdu pracuje a aby bylo zprávy možno podepsat a ověřit všemi uživateli systému, vše co potřebujeme, aby platilo

$$0 \leq \max \mathbf{M} \leq \min \{m_l : l \in \mathbf{U}\}, \tag{1.1}$$

kde **U** je množina uživatelů systému.

O čem to bude



Prvočísla I

V uvedené verzi RSA-algoritmu vystupují veřejné parametry (e_I, n_I) a tajné parametry d_I, p_I a q_I spolu s číselným vyjádřením (části) zprávy M. Rozeberme si požadavky na jejich výběr.

- Při použití RSA-algoritmu každý účastník systému používá dvě (čtyři) cca. 100-místná prvočísla.

Prvočísla I

V uvedené verzi RSA-algoritmu vystupují veřejné parametry (e_I, n_I) a tajné parametry d_I, p_I a q_I spolu s číselným vyjádřením (části) zprávy M. Rozeberme si požadavky na jejich výběr.

- Při použití RSA-algoritmu každý účastník systému používá dvě (čtyři) cca. 100-místná prvočísla.

  Obvykle se dnes volí prvočísla o velikosti alespoň 1024 bitů, tj. čísla řádově kolem 2^{1024}. Modul n má tedy velikost 2048 bitů.

Prvočísla I

V uvedené verzi RSA-algoritmu vystupují veřejné parametry (e_I, n_I) a tajné parametry d_I, p_I a q_I spolu s číselným vyjádřením (části) zprávy M. Rozeberme si požadavky na jejich výběr.

- Při použití RSA-algoritmu každý účastník systému používá dvě (čtyři) cca. 100-místná prvočísla.

  Obvykle se dnes volí prvočísla o velikosti alespoň 1024 bitů, tj. čísla řádově kolem 2^{1024}. Modul n má tedy velikost 2048 bitů.

  Kolik jich máme k dispozici? Použitím prvočíselné funkce π, která udává počet prvočísel menších než dopředu zvolené číslo n a odhaduje se pomocí odhadu

  $$\pi(n) \doteq \frac{n}{\ln n}$$

  získáme přibližný počet prvočísel δ ležících v intervalu $[2^{1023}, 2^{1024}]$.

Prvočísla II

Počítejme:

$$\begin{aligned}\delta &= \pi(2^{1024}) - \pi(2^{1023}) \doteq \frac{2^{1024}}{\ln 2^{1024}} - \frac{2^{1023}}{\ln 2^{1023}} \\ &= \frac{2^{1024}}{1024 \cdot \ln 2} - \frac{2^{1023}}{1023 \cdot \ln 2} \doteq \frac{2^{1023}}{1024 \cdot \ln 2} \doteq 2^{1023}.\end{aligned}$$

Pravděpodobnost, že by si dva účastníci systému vybrali tutéž dvojici 100-místných prvočísel, je pak řádově 2^{-4092}.

Prvočísla III - Jak je nalézt?

- Dalším problémem je nalezení 100-místného náhodného prvočísla.

Prvočísla III - Jak je nalézt?

- Dalším problémem je nalezení 100-místného náhodného prvočísla.

  Nejprve pomocí generátoru pseudonáhodných čísel sestrojíme 100-místné náhodné číslo m.

Prvočísla III - Jak je nalézt?

- Dalším problémem je nalezení 100-místného náhodného prvočísla.

  Nejprve pomocí generátoru pseudonáhodných čísel sestrojíme 100-místné náhodné číslo m.

  V případě, že m bude sudé, nahradíme ho číslem $m + 1$. Pak nové číslo m otestujeme některým z testů na prvočíselnost.

Prvočísla III - Jak je nalézt?

- Dalším problémem je nalezení 100-místného náhodného prvočísla.

  Nejprve pomocí generátoru pseudonáhodných čísel sestrojíme 100-místné náhodné číslo m.

  V případě, že m bude sudé, nahradíme ho číslem $m + 1$. Pak nové číslo m otestujeme některým z testů na prvočíselnost.

  Pokud m nebude prvočíslo, vyzkoušíme číslo $m + 2$ a postup opakujeme až do té doby, než nenajdeme první prvočíslo větší než m.

Prvočísla III - Jak je nalézt?

- Dalším problémem je nalezení 100-místného náhodného prvočísla.

  Nejprve pomocí generátoru pseudonáhodných čísel sestrojíme 100-místné náhodné číslo m.

  V případě, že m bude sudé, nahradíme ho číslem $m + 1$. Pak nové číslo m otestujeme některým z testů na prvočíselnost.

  Pokud m nebude prvočíslo, vyzkoušíme číslo $m + 2$ a postup opakujeme až do té doby, než nenajdeme první prvočíslo větší než m.

  Lze ověřit, že počet pokusů nutných k nalezení prvočísla v okolí čísla m je logaritmickou funkcí čísla m.

Prvočísla IV - Jak je nalézt?

Jako příklad uvedeme prvočísla o délce 1024 bitů.

Prvočísla IV - Jak je nalézt?

Jako příklad uvedeme prvočísla o délce 1024 bitů.

Podle výše uvedeného odhadu je z 2^{1024} čísel této délky asi 2^{1013} prvočísel.

Prvočísla IV - Jak je nalézt?

Jako příklad uvedeme prvočísla o délce 1024 bitů.

Podle výše uvedeného odhadu je z 2^{1024} čísel této délky asi 2^{1013} prvočísel.

Pravděpodobnost, že náhodně vybrané číslo je prvočíslo, je přibližně 2^{-10}.

Prvočísla IV - Jak je nalézt?

Jako příklad uvedeme prvočísla o délce 1024 bitů.

Podle výše uvedeného odhadu je z 2^{1024} čísel této délky asi 2^{1013} prvočísel.

Pravděpodobnost, že náhodně vybrané číslo je prvočíslo, je přibližně 2^{-10}.

Pokud od počátku vyloučíme sudá čísla, zlepší se pravděpodobnost na 2^{-9}.

Prvočísla IV - Jak je nalézt?

Jako příklad uvedeme prvočísla o délce 1024 bitů.

Podle výše uvedeného odhadu je z 2^{1024} čísel této délky asi 2^{1013} prvočísel.

Pravděpodobnost, že náhodně vybrané číslo je prvočíslo, je přibližně 2^{-10}.

Pokud od počátku vyloučíme sudá čísla, zlepší se pravděpodobnost na 2^{-9}.

Máte tedy v průměru kolem 512 pokusů na nalezení prvočísla o délce 1024 bitů.

Prvočísla IV - Jak je nalézt?

Jako příklad uvedeme prvočísla o délce 1024 bitů.

Podle výše uvedeného odhadu je z 2^{1024} čísel této délky asi 2^{1013} prvočísel.

Pravděpodobnost, že náhodně vybrané číslo je prvočíslo, je přibližně 2^{-10}.

Pokud od počátku vyloučíme sudá čísla, zlepší se pravděpodobnost na 2^{-9}.

Máte tedy v průměru kolem 512 pokusů na nalezení prvočísla o délce 1024 bitů.

Jako další krok uvedeme příklad jednoduchého pravděpodobnostního algoritmu na zjištění prvočíselnosti čísla m.

Prvočísla V - Jak je nalézt?

Algoritmus na zjištění prvočíselnosti čísla m na k pokusů

BEGIN

READ (m, k);

Prvočísla V - Jak je nalézt?

Algoritmus na zjištění prvočíselnosti čísla m na k pokusů

```
BEGIN
    READ (m, k);
    FOR i := 1 TO k DO
        BEGIN
            a :=RANDOM(1, m − 1);
            b := (a**(m − 1) MOD m);
          IF b <> 1 THEN
                BEGIN
                  WRITE (m, "je složené číslo");
                  GO TO KONEC
                END
    END;
```

Prvočísla V - Jak je nalézt?

Algoritmus na zjištění prvočíselnosti čísla m na k pokusů

```
        BEGIN
            READ (m, k);
            FOR i := 1 TO k DO
                BEGIN
                    a :=RANDOM(1, m − 1);
                    b := (a**(m − 1) MOD m);
                IF b <> 1 THEN
                        BEGIN
                          WRITE (m, "je složené číslo");
                          GO TO KONEC
                        END
                END;
            WRITE (m, "je prvočíslo");
KONEC:  END.
```

Prvočísla VI - Jak je nalézt?

Funkce RANDOM vybírá pseudonáhodná celá čísla z určeného intervalu.

Prvočísla VI - Jak je nalézt?

Funkce RANDOM vybírá pseudonáhodná celá čísla z určeného intervalu.

Algoritmus na vstupu načte číslo m a číslo k a na výstupu obdržíme buď pravdivou odpověď, že m je složené číslo nebo odpověď, že se asi jedná o prvočíslo.

Prvočísla VI - Jak je nalézt?

Funkce RANDOM vybírá pseudonáhodná celá čísla z určeného intervalu.

Algoritmus na vstupu načte číslo m a číslo k a na výstupu obdržíme buď pravdivou odpověď, že m je složené číslo nebo odpověď, že se asi jedná o prvočíslo.

V případě, že je k dostatečně velké, je pravděpodobnost, že se nejedná o prvočíslo, v případě kladné odpovědi velmi malá.

Nevýhody RSA-systému I

V praxi máme několik nevýhod RSA-systému:

Nevýhody RSA-systému I

V praxi máme několik nevýhod RSA-systému:

(a) Odesílatel A může úmyslně "ztratit" svůj soukromý klíč tak, že, ačkoliv je uložen v "bance soukromých klíčů" před startem systému, jí odeslané zprávy se stanou neověřitelnými.

Nevýhody RSA-systému I

V praxi máme několik nevýhod RSA-systému:

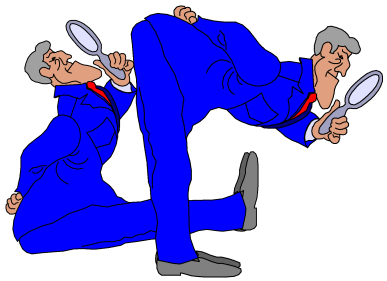
(a) Odesílatel A může úmyslně "ztratit" svůj soukromý klíč tak, že, ačkoliv je uložen v "bance soukromých klíčů" před startem systému, jí odeslané zprávy se stanou neověřitelnými.

(b) Odesílatel A může úmyslně vydat svůj soukromý klíč d_A a dovolit tak, aby všechny jí adresované zprávy byly řešitelné.

Nevýhody RSA-systému II

(c) Doba věnovaná šifrování, podepsání, dešifrování a prověření může být nepřiměřená. Totiž teprve nedávno byla nalezena rozumná implementace RSA-algoritmu a v současné době jsou na trhu RSA-čipy, které ale mají rychlost asi 10 Kbit/s. K dispozici jsou i speciální RSA-karty, které zvládnou 100 Kbit/s. Uvážíme-li však, že budoucí ISDN síťový standard elektronické pošty pracuje s 64Kbit/s a že se v půmyslu (lokální sítě apod.) pracuje s rychlostmi kolem 10 Mbit/s, vidíme, že nebude ještě dlouho možno použít RSA-algoritmus za účelem šifrování zpráv, nýbrž hlavně pro správu klíčů a elektronické podpisování. Při tvorbě elektronického podpisu se totiž nejdříve text zkomprimuje a podpisovací algoritmus se aplikuje na komprimát; není tedy nutno podpisovat velké soubory.

O čem to bude



Popis metody I

Jeden z prvních (1978) sytémů s veřejným klíčem byl vyvinut **Merklem** a **Hellmanem** a byl založen na tzv. ruksakovém problému.

Popis metody I

Jeden z prvních (1978) sytémů s veřejným klíčem byl vyvinut **Merklem** a **Hellmanem** a byl založen na tzv. ruksakovém problému.

Přesněji, jedná se o výpočetní problém známý jako ***PODMNOŽINOVÝ SOUČET*** definovaný následovně:

Popis metody I

Jeden z prvních (1978) sytémů s veřejným klíčem byl vyvinut **Merklem** a **Hellmanem** a byl založen na tzv. ruksakovém problému.

Přesněji, jedná se o výpočetní problém známý jako ***PODMNOŽINOVÝ SOUČET*** definovaný následovně:

Vstup:

Otázka:

Popis metody I

Jeden z prvních (1978) sytémů s veřejným klíčem byl vyvinut **Merklem** a **Hellmanem** a byl založen na tzv. ruksakovém problému.

Přesněji, jedná se o výpočetní problém známý jako ***PODMNOŽINOVÝ SOUČET*** definovaný následovně:

Vstup: Kladná reálná čísla $a_1, a_2, \ldots, a_n, t$

Otázka:

Popis metody I

Jeden z prvních (1978) sytémů s veřejným klíčem byl vyvinut **Merklem** a **Hellmanem** a byl založen na tzv. ruksakovém problému.

Přesněji, jedná se o výpočetní problém známý jako ***PODMNOŽINOVÝ SOUČET*** definovaný následovně:

Vstup: Kladná reálná čísla $a_1, a_2, \ldots, a_n, t$

Otázka: Existuje podmnožina $J \subseteq \{1, \ldots, n\}$ tak, že

$$\sum_{i \in J} a_i = t?$$

Popis metody I

Jeden z prvních (1978) sytémů s veřejným klíčem byl vyvinut **Merklem** a **Hellmanem** a byl založen na tzv. ruksakovém problému.

Přesněji, jedná se o výpočetní problém známý jako ***PODMNOŽINOVÝ SOUČET*** definovaný následovně:

Vstup: Kladná reálná čísla $a_1, a_2, \ldots, a_n, t$

Otázka: Existuje podmnožina $J \subseteq \{1, \ldots, n\}$ tak, že

$$\sum_{i \in J} a_i = t?$$

Tento problém je jedním z klasických NP-úplných problémů.

Popis metody II - Zašifrování zprávy

1. Odesílaná zpráva je odeslána v binárním tvaru **m**.
2. Veřejné klíče tvoří soubor n-tic $(a_1, \ldots, a_n)$ kladných přirozených čísel.
3. Binární zpráva **m** je rozdělena do bloků a n znacích tak, že $\mathbf{m} = \mathbf{m_1} \ldots \mathbf{m_t}$, kde každé $\mathbf{m_j}$ je n-tice nul a jedniček.
4. Pro každé j, $1 \leq j \leq t$, položme

$$c_= \sum_{i=1}^{n} M_i \cdot a_i,$$

   kde $\mathbf{m_j} = (M_1, \ldots, M_n)$.
5. Přeneseme posloupnost (kryptogram) $c_1, \ldots, c_t$.

Popis metody III - Dešifrování zprávy

Zdánlivě se příjemce a každý, kdo zachytí kryptogram, zabývají tímže problémem; aby bylo možno rozluštit zprávu z posloupnosti $c_1, \ldots, c_t$ a veřejného klíče $(a_1, \ldots, a_n)$, musí vyřešit t různých NP-úplných problémů, každý pro jedno c_j.

Popis metody III - Dešifrování zprávy

Zdánlivě se příjemce a každý, kdo zachytí kryptogram, zabývají tímže problémem; aby bylo možno rozluštit zprávu z posloupnosti $c_1, \ldots, c_t$ a veřejného klíče $(a_1, \ldots, a_n)$, musí vyřešit t různých NP-úplných problémů, každý pro jedno c_j.

Merkle-Hellmanův systém je založen na skutečnosti, že ne všechny případy NP-úplných problémů jsou obtížně řešitelné.

Popis metody III - Dešifrování zprávy

Zdánlivě se příjemce a každý, kdo zachytí kryptogram, zabývají tímže problémem; aby bylo možno rozluštit zprávu z posloupnosti $c_1, \ldots, c_t$ a veřejného klíče $(a_1, \ldots, a_n)$, musí vyřešit t různých NP-úplných problémů, každý pro jedno c_j.

Merkle-Hellmanův systém je založen na skutečnosti, že ne všechny případy NP-úplných problémů jsou obtížně řešitelné.

Řekneme, že posloupnost $a_1, \ldots, a_n$ je ***superrostoucí***, jestliže pro všechna k, $1 \leq k \leq n$, platí

$$a_{k+1} > \sum_{i=1}^{k} a_i. \tag{3.1}$$

Popis metody IV - Dešifrování zprávy

Lemma 3.1

Existuje **rychlý algoritmus** *(v polynomiálním čase) pro vyřešení třídy problémů podmnožinového součtu pro superrostoucí posloupnosti.*

Popis metody IV - Dešifrování zprávy

Lemma 3.1

Existuje **rychlý algoritmus** *(v polynomiálním čase) pro vyřešení třídy problémů podmnožinového součtu pro superrostoucí posloupnosti.*

Důkaz. Předpokládejme, že posloupnost $a_1, \ldots, a_n$ je superrostoucí,. Potřebujeme reprezentovat vstup t jako součet vybrané podposloupnosti posloupnosti $a_1, \ldots, a_n$ nebo rozhodnout, že takovouto reprezentaci nelze najít.

Popis metody IV - Dešifrování zprávy

Lemma 3.1

Existuje **rychlý algoritmus** *(v polynomiálním čase) pro vyřešení třídy problémů podmnožinového součtu pro superrostoucí posloupnosti.*

Důkaz. Předpokládejme, že posloupnost $a_1, \ldots, a_n$ je superrostoucí,. Potřebujeme reprezentovat vstup t jako součet vybrané podposloupnosti posloupnosti $a_1, \ldots, a_n$ nebo rozhodnout, že takovouto reprezentaci nelze najít.
Položme $r = \max\{i : a_i \leq t\}$. Pak $t = a_r + s$, kde nyní potřebujeme najít reprezentaci čísla s jako součet vybrané podposloupnosti posloupnosti $a_1, \ldots, a_{r-1}$ nebo rozhodnout, že takovouto reprezentaci nelze najít. Opakování tohoto postupu nám pak dá naši reprezentaci pro t nebo objeví, že takovou reprezentaci není možno najít.

Základ Merkle-Hellmanova systému I

1. Typický uživatel A si vybere "snadnou" superrostoucí posloupnost přirozených čísel $e_1, \ldots, e_n$.

Základ Merkle-Hellmanova systému I

1. Typický uživatel A si vybere "snadnou" superrostoucí posloupnost přirozených čísel $e_1, \ldots, e_n$.
2. Uživatel si vybere dvojici "velkých" nesoudělných přirozených čísel w a N a transformuje pomocí ní vybranou superrostoucí posloupnost do "obtížné" posloupnosti $T(e_1), \ldots, T(e_n)$ podle předpisu

$$T(e_i) = w \cdot e_i \pmod{N}.$$

Základ Merkle-Hellmanova systému I

1. Typický uživatel A si vybere "snadnou" superrostoucí posloupnost přirozených čísel $e_1, \ldots, e_n$.
2. Uživatel si vybere dvojici "velkých" nesoudělných přirozených čísel w a N a transformuje pomocí ní vybranou superrostoucí posloupnost do "obtížné" posloupnosti $T(e_1), \ldots, T(e_n)$ podle předpisu

$$T(e_i) = w \cdot e_i \pmod{N}.$$

Transformovaný vektor $(T(e_1), \ldots, T(e_n))$ se stane ***veřejným klíčem*** uživatele A. Přitom by mělo být

$$N > e_1 + e_2 + \ldots + e_n.$$

Základ Merkle-Hellmanova systému II

Lemma 3.2

Buď c kryptogram odeslaný uživatelem A při použití obtížného veřejného klíče $T(e_1), \ldots, T(e_n)$ uživatele A a předpisu $T(e_i) = w \cdot e_i \pmod{N}$. Lehký kryptogram c' pak získáme z následujícího vzorce:

$$c' = w^{-1} \cdot c \pmod{N}.$$

Základ Merkle-Hellmanova systému II

Lemma 3.2

Buď c kryptogram odeslaný uživatelem A při použití obtížného veřejného klíče $T(e_1), \ldots, T(e_n)$ uživatele A a předpisu $T(e_i) = w \cdot e_i \pmod{N}$. Lehký kryptogram c' pak získáme z následujícího vzorce:

$$c' = w^{-1} \cdot c \pmod{N}.$$

Důkaz. Položme $a_i = T(e_i)$. Je-li tedy $M = (M_1 M_2 \ldots M_n)$ zpráva, pak máme

$$c = \sum_{i=1}^{n} M_i \cdot a_i.$$

Základ Merkle-Hellmanova systému II

Ale

$$c = \sum_{i=1}^{n} M_i \cdot a_i = \sum_{i=1}^{n} M_i \cdot w \cdot e_i + \sum_{i=1}^{n} M_i \cdot N \cdot d_i = \sum_{i=1}^{n} M_i \cdot w \cdot e_i \pmod{N}$$

pro vhodná přirozená čísla d_i.
Máme tedy

$$w^{-1} \cdot c = \sum_{i=1}^{n} M_i \cdot e_i \pmod{N}.$$

Výhody a nevýhody I

Zásadní výhodou je relativně vysoká rychlost šifrování odesílatelem. Skutečně, výpočet součtu je velmi rychlý. Viditelnou nevýhodou systému je jeho linearita. Skutečně, platí $E(x+y) = E(x) + E(y)$, kde $E(x) = x * a = \sum_{i=1}^{n} x_i a_i$ je operace zašifrování.

Výhody a nevýhody I

Zásadní výhodou je relativně vysoká rychlost šifrování odesílatelem. Skutečně, výpočet součtu je velmi rychlý. Viditelnou nevýhodou systému je jeho linearita. Skutečně, platí $E(x+y) = E(x) + E(y)$, kde $E(x) = x * a = \sum_{i=1}^{n} x_i a_i$ je operace zašifrování.

Navíc v nejnižším bitu součtu $E(x)$ se operace sčítání proměňuje na operaci XOR. Proto nejnižší bit součtu $E(x)$, což je dostupný šifrový text, je roven výsledku operace XOR těch bitů otevřeného textu tj. vektoru x, které stojí u lichých a_i. Bude-li například liché jen a_1 a a_2, dostaneme x_1 or x_2=nejnižší bit $E(x)$.

Výhody a nevýhody I

Zásadní výhodou je relativně vysoká rychlost šifrování odesílatelem. Skutečně, výpočet součtu je velmi rychlý. Viditelnou nevýhodou systému je jeho linearita. Skutečně, platí $E(x+y) = E(x) + E(y)$, kde $E(x) = x * a = \sum_{i=1}^{n} x_i a_i$ je operace zašifrování.

Navíc v nejnižším bitu součtu $E(x)$ se operace sčítání proměňuje na operaci XOR. Proto nejnižší bit součtu $E(x)$, což je dostupný šifrový text, je roven výsledku operace XOR těch bitů otevřeného textu tj. vektoru x, které stojí u lichých a_i. Bude-li například liché jen a_1 a a_2, dostaneme x_1 or x_2=nejnižší bit $E(x)$.

I když to není velká informace, ze šifrového textu by neměla "vyzařovat". Kvalitní šifrovací systémy nevydávají o otevřeném textu vůbec žádnou využitelnou informaci.

Výhody a nevýhody II

Merkle, jeden ze spoluautorů výše uvedeného šifrovacího systému, si byl jeho bezpečností tak jist, že na něj vsadil 100 USD. Bezpečností se pak zabývalo mnoho vědců.

Výhody a nevýhody II

Merkle, jeden ze spoluautorů výše uvedeného šifrovacího systému, si byl jeho bezpečností tak jist, že na něj vsadil 100 USD. Bezpečností se pak zabývalo mnoho vědců.

Herlestam učinil zkušenost, že poměrně často lze zjistit jeden bit otevřené zprávy.

Výhody a nevýhody II

Merkle, jeden ze spoluautorů výše uvedeného šifrovacího systému, si byl jeho bezpečností tak jist, že na něj vsadil 100 USD. Bezpečností se pak zabývalo mnoho vědců.

Herlestam učinil zkušenost, že poměrně často lze zjistit jeden bit otevřené zprávy.

Shamir ukázal, že je řešitelný tzv. kompaktní problém rance.

Výhody a nevýhody II

Merkle, jeden ze spoluautorů výše uvedeného šifrovacího systému, si byl jeho bezpečností tak jist, že na něj vsadil 100 USD. Bezpečností se pak zabývalo mnoho vědců.

Herlestam učinil zkušenost, že poměrně často lze zjistit jeden bit otevřené zprávy.

Shamir ukázal, že je řešitelný tzv. kompaktní problém rance.

S Zippelem pak dokázali řešitelnost Merkle-Hellmanova systému, jestliže luštitel bude znát tajný modul N. Připomeňme, že "problém celočíselného programování" je NP-úplný.

Výhody a nevýhody III

Pokrok pak nastal po Lenstrově objevu řešitelnosti problému celočíselného programování pro "zafixovaný počet proměnných" v polynomiálním čase.

Výhody a nevýhody III

Pokrok pak nastal po Lenstrově objevu řešitelnosti problému celočíselného programování pro "zafixovaný počet proměnných" v polynomiálním čase.

S jeho využitím pak Shamir popsal metodu řešení ruksakového problému, který se používá v kryptografii, v polynomiálním čase a tím vyhrál Merklovu sázku.

Výhody a nevýhody III

Pokrok pak nastal po Lenstrově objevu řešitelnosti problému celočíselného programování pro "zafixovaný počet proměnných" v polynomiálním čase.

S jeho využitím pak Shamir popsal metodu řešení ruksakového problému, který se používá v kryptografii, v polynomiálním čase a tím vyhrál Merklovu sázku.

Jeho algoritmus vyřeší "většinu" kryptografických ruksakových problémů. Merkle sice prohrál 100 USD, ale vsadil desetkrát tolik, že iterovaný problém nebude rozbit.

Výhody a nevýhody III

Pokrok pak nastal po Lenstrově objevu řešitelnosti problému celočíselného programování pro "zafixovaný počet proměnných" v polynomiálním čase.

S jeho využitím pak Shamir popsal metodu řešení ruksakového problému, který se používá v kryptografii, v polynomiálním čase a tím vyhrál Merklovu sázku.

Jeho algoritmus vyřeší "většinu" kryptografických ruksakových problémů. Merkle sice prohrál 100 USD, ale vsadil desetkrát tolik, že iterovaný problém nebude rozbit.

E. Brickel v létě 1984 oznámil, že je schopen rozluštit čtyřicetkrát iterovaný problém rance během jedné hodiny.