Ministerstvo financií
Slovenskej republiky

EFQM
Recognised for excellence
5 star - 2012

Asymetrické šifry

M. Stanek

KPMG
cutting through complexity™

Asymetrické šifry

Martin Stanek

Obsah

1. RSA
2. Schémy založené na probléme DLOG
3. Eliptické krivky

Idea

- Inštancia asymetrickej šifry: verejný kľúč, súkromný kľúč
- **Verejný kľúč** $\Rightarrow$ každý vie šifrovať
- **Súkromný kľúč** $\Rightarrow$ len vlastník vie dešifrovať

- Schéma asymetrickej šifry (trojica algoritmov):
  - generovanie kľúčov (vytvorenie inštancie)
  - šifrovanie
  - dešifrovanie
- Požadujeme:
  - Korektnosť – po dešifrovaní ŠT očakávame pôvodný OT
  - Efektívnosť – pre všetky algoritmy schémy
  - Bezpečnosť – CPA scenár pre útoky je minimum (viac neskôr)

Výhody/výzvy

- Nie je potrebné zabezpečiť dôvernosť verejných kľúčov
- Ako dôveryhodne distribuovať verejný kľúč (autentickosť)?
  - Osobne / protokolárne
  - Prostredníctvom dôveryhodnej tretej strany – PKI
- Kľúče obvykle platné a používané dlhšie ako symetrické
- Aj asymetrické kľúče je potrebné pravidelne meniť ...
  - Google Chrome ... 60 certifikátov (Trusted Root CA) z toho 16 exspiruje až v 2031 a ďalších 25 certifikátov po roku 2020
- Výkonnostné obmedzenia

Problémy pre asymetrické konštrukcie

- Faktorizácia veľkých čísel
  - RSA problém, QR problém, počítanie druhých odmocnín a pod.
- Diskrétny logaritmus v rôznych grupách
  - DLOG, rozhodovací a výpočtový DH problém, Gap DH problém, Twin DH problém a pod.
- Mriežky
  - SVP (Shortest vector problem), CVP (Closest vector problem), LWE (Learning with errors) a pod.
- Iné
  - SDP (Syndrome Decoding Problem) a pod.

Faktorizácia

- Úloha: vypočítať rozklad čísla n
  - zvyčajne $n = p \cdot q$ (súčin dvoch veľkých prvočísel)
- Najlepší všeobecný algoritmus: General Number Field Sieve (GNFS)
- Heuristická zložitosť:

$$\exp(((64/9)^{1/3} + o(1)) (\ln n)^{1/3} (\ln \ln n)^{2/3})$$

- Efektívnejšie algoritmy existujú pre niektoré voľby p, q
  - Napr. p, q blízko pri sebe, (p - 1) alebo (q - 1) bez veľkého prvočíselného faktora a pod.

Faktorizácia (2)

- Ekvivalentné (porovnateľné) dĺžky kľúčov

symetrický kľúč	faktorizácia *n*	
80	1024	1248
112	2048	2432
128	3072	3248
256	15360	15424
	NIST SP 800-57 part 1 rev. 3	ECRYPT II report

Porovnanie www.keylength.com

Diskrétny logaritmus

- Úloha: pre dané g a y vypočítať číslo x také, že $g^x = y$
  - g je generátor nejakej grupy G a „násobenie“ príslušná operácia
- Ľahký/ťažký problém v závislosti na G
- V kryptografii sú obvykle používané:
  - Multiplikatívna grupa (podgrupa) modulo prvočíslo, pričom operácia je násobenie v modulárnej aritmetike
  - Grupa bodov eliptickej krivky (operácia je sčítanie bodov krivky)
- Všeobecný generický algoritmus (pre ľubovoľnú konečnú cyklickú grupu): $O(n^{1/2})$
  - Baby-step giant-step, Pollard ρ
- Pre modulárnu aritmetiku: Number Field Sieve pre DLOG
  - Rovnaká zložitosť ako GNFS pre faktorizáciu

Diskrétny logaritmus (2)

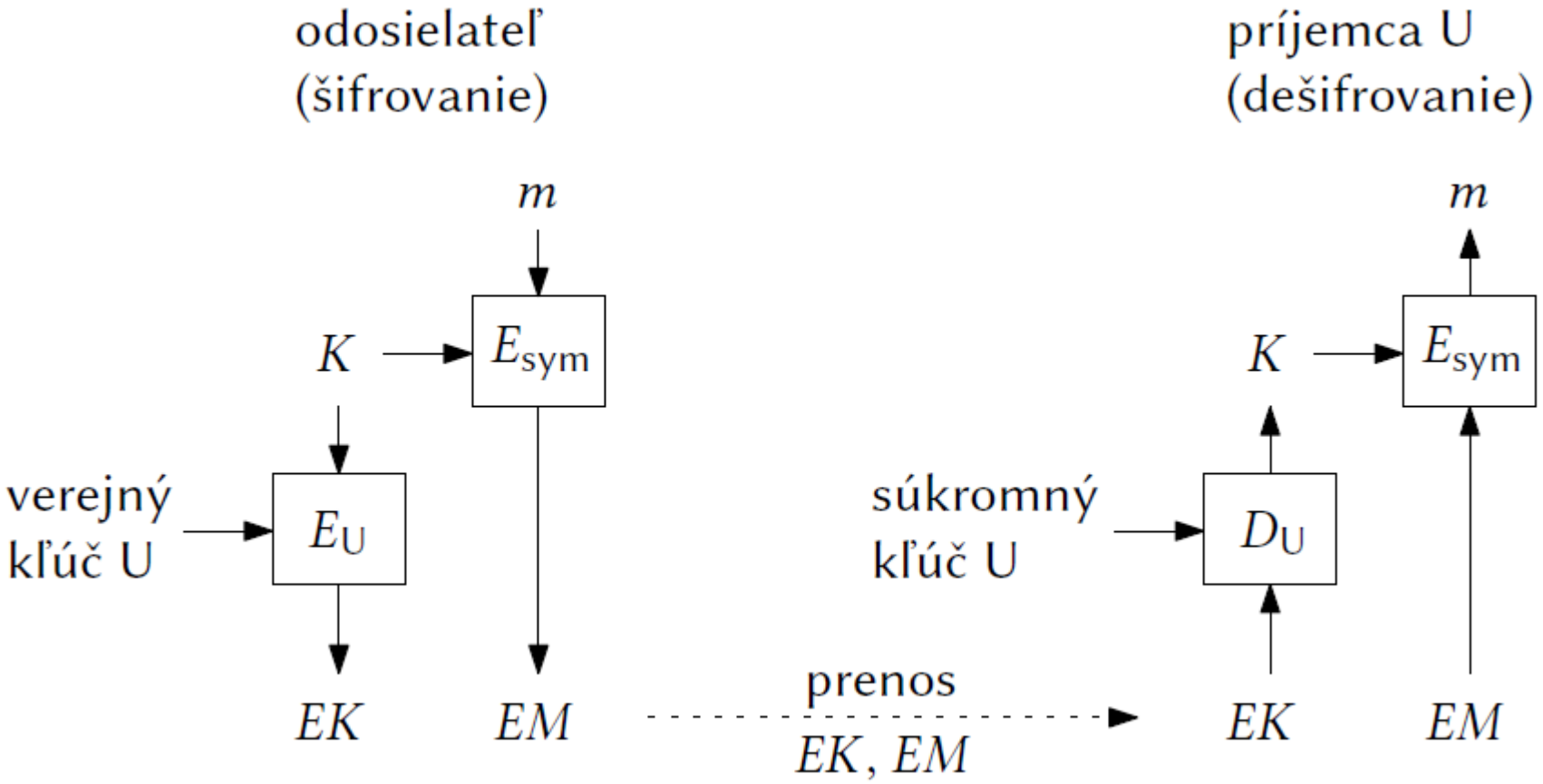
- Ekvivalentné (porovnateľné) dĺžky kľúčov

symetrický kľúč	mod (podgrupa)	el. krivky
80	1024 (160)	160
112	2048 (224)	224
128	3072 (256)	256
256	15360 (512)	512

NIST SP 800-57 part 1 rev. 3

Hybridné schémy

- Vzhľadom na výkonové charakteristiky – použitie asymetrického šifrovania najmä v hybridných schémach



RSA

- Rivest, Shamir, Adleman 1977, (Cocks 1973)
- Inicializácia:
  1. Rôzne prvočísla p, q; modulus $n = pq$
  2. Verejný exponent e nesúdeliteľný s $(p-1)(q-1)$
  3. Súkromný exponent d: $ed \equiv 1 \pmod{(p-1)(q-1)}$
- Verejný kľúč: (n, e)
- Verejná transformácia: $E(m) = m^e \bmod n$, pre $m \in Z_n$
- Súkromný kľúč: d
- Súkromná transformácia: $D(c) = c^d \bmod n$, pre $c \in Z_n$
- E a D sú navzájom inverzné permutácie množiny Z_n
  - Teda $D(E(x)) = x$, $E(D(x)) = x$

RSA – malý príklad

- $p = 13$, $q = 17$ ... $n = 13 \cdot 17 = 221$
- $(p-1)(q-1) = 12 \cdot 16 = 192$
- zvoľme $e = 7$ (nesúdeliteľné so 192)
- potom $d = 55$, lebo $7 \cdot 55 \equiv 1 \pmod{192}$
- verejný kľúč: (221, 7), súkromný kľúč: 55
- Príklady hodnôt E a D transformácií:

$$E(10) = 10^7 \bmod 221 = 192$$
$$D(192) = 192^{55} \bmod 221 = 10$$

$$E(99) = 99^7 \bmod 221 = 57$$
$$D(57) = 57^{55} \bmod 221 = 99$$

Na zamyslenie (1)

Použime RSA systém takto:

Šifrujme OT po bitoch (teda každý bit zvlášť), priamym použitím verejnej transformácie *E*.

Je takáto schéma bezpečná? Prečo?

RSA – implementačné poznámky

- Verejný exponent volený najčastejšie ako 65537
  - prvočíslo $\Rightarrow$ vysoká pravdepodobnosť nesúdeliteľnosti s $(p\text{-}1)(q\text{-}1)$
  - ľahké testovanie nesúdeliteľnosti už pri generovaní p, q
  - krátke a binárny zápis s malým počtom 1 $\Rightarrow$ rýchly výpočet E
  - príliš malý exponent (napr. $e = 3$) môže byť problematický z hľadiska bezpečnosti
- Súkromný exponent je jednoznačne určený
  - dĺžka d približne rovnaká ako dĺžka n
  - matematicky je možné najskôr zvoliť d a dopočítať e (krátke d je náchylné na útoky, pre $d < n^{0,292}$)
  - D výpočtovo náročnejšia ako E

    napr. RSA-2048: $E \sim 27496$ ops/s, $D \sim 857$ ops/s

RSA – implementačné poznámky 2

- *D* sa urýchľuje s využitím známej faktorizácie
  - prevypočítané hodnoty (nezávisia na vstupe):

    $d_p = d \bmod (p-1)$

    $d_q = d \bmod (q-1)$

    $w = q^{-1} \bmod p$

- Výpočet $D(c)$:

  $D(c) = m' + q \cdot (w \cdot ((c^{dp} \bmod p) - m') \bmod p)$

  kde $m' = c^{dq} \bmod q$

napriek 2 umocneniam efektívnejšie ako priame umocnenie s *d*

RSA – štruktúra súkromného kľúča

- openssl

  openssl genrsa -out rsa-private.key 2048

  openssl rsa -in rsa-private.key -text

  modulus: n

  publicExponent: e

  privateExponent: d

  prime1: p

  prime2: q

  exponent1: d_p

  exponent2: d_q

  coefficient: w

Bezpečnosť RSA

- RSA problém: daný verejný kľúč a ŠT treba vypočítať OT
- Faktorizácia $\Rightarrow$ RSA problém
- RSA problém $\Rightarrow ? \Rightarrow$ Faktorizácia (otvorený problém)

- Faktorizácia $\Leftrightarrow$ znalosť *d*

- Schopnosť počítať zo ŠT paritný bit OT alebo predikát *half* umožní dešifrovať ľubovoľný ŠT

Bezpečnosť „učebnicovej“ verzie

- Determinizmus (*E* a *D* sú deterministické)
  - Každému OT zodpovedá len jeden ŠT (možnosť testovať OT)
  - Problém s malým priestorom správ („áno“/„nie“, výplata a pod.)

- Poddajnosť šifrovania (malleability):

$$E(m_1) \cdot E(m_2) \bmod n = E(m_1 \cdot m_2 \bmod n)$$

  - teda vieme prenásobiť OT dvoma, jednou polovicou atď.

- Príliš krátky verejný exponent, napr. $e = 3$ a priamočiare šifrovanie symetrického kľúča, napr.:
  - K má 256 bitov, n má 2048 bitov
  - K^3 má 768 bitov, teda $K^3 < n$ ... ľahko dešifrovať

Bezpečnosť „učebnicovej“ verzie (2)

- TMTO pre krátky OT
  - K má t bitov a nech $K = K_1 \cdot K_2$
  - Vstup: $C = E(K)$
  - Predvypočítame tabuľku hodnôt $E(1), E(2), ..., E(2^{t/2})$
  - Testujeme, či $C \cdot (E(i))^{-1} \bmod n$ je v tabuľke pre $i = 1, ..., 2^{t/2}$
  - Úspech znamená nájdenie K_1 a K_2
  - Zložitosť $\sim 2^{t/2}$
- Malý exponent a polynomiálne závislé správy:
  - $m_2 = p(m_1)$, $c_1 = E(m_1)$, $c_2 = E(m_2)$
  - $(z - m_1) \mid z^e - c_1$ a $(z - m_1) \mid p(z)^e - c_2$
  - Výpočet $\gcd(z^e - c_1, p(z)^e - c_2)$ môže viesť k m_1 a následne m_2

Na zamyslenie (2)

Predpokladajme, že algoritmus *A* dokáže efektívne riešiť RSA problém (teda dešifrovať bez súkromného kľúča) pre jednu miliónтinu hodnôt zo Z_n. Je možné potom efektívne riešiť RSA problém pre všetky hodnoty zo Z_n?

RSA v praxi – výplňové schémy

- RSA sa má používať s výplňou (padding)
- Motivácia
  - Znáhodnenie šifrovania
  - Možnosť *dokázať* bezpečnosť schémy
- Používané výplne
  - PKCS#1 v1.5
  - OAEP
- Iné
  - OAEP+, REACT, SAEP, ...

PKCS#1 v1.5

- „not recommended for new applications“ (RFC 3447)
- Štruktúra výplne:

$$00 \,||\, 02 \,||\, PS \,||\, 00 \,||\, m$$

  - *m* – otvorený text
  - PS – reťazec pseudonáhodných nenulových bajtov dĺžky aspoň 8
  - diskusia o bezpečnosti výplne a rôzne doporučenia v RFC 3447
  - hlavný problém: s nezanedbateľnou pravdepodobnosťou možno generovať ŠT s korektnou výplňou

Bleichenbacherov útok

- CCA (útok s možnosťou voľby ŠT, adaptívne), 1998
- Predpoklad: vieme odlíšiť ŠT sa správnou/nesprávnou výplňou PKCS#1 v1.5
  - Napr. špecifická chybová hláška, timing útok a pod.

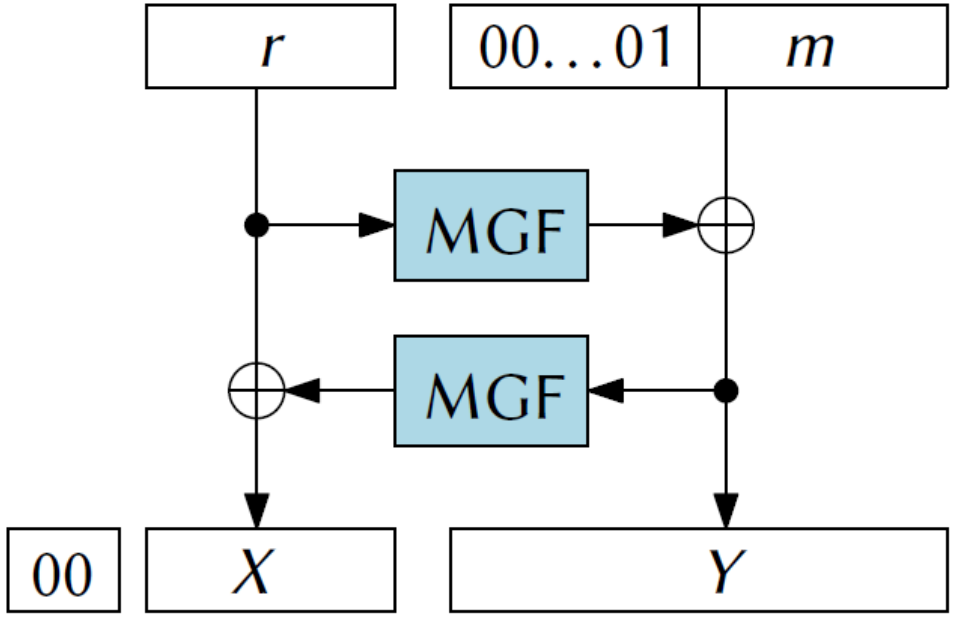
- Útok: výpočet $D(c)$ pre ľubovoľné c
- RSA 1024 ~ 2^{20} zvolených ŠT

RSA-OAEP

- OAEP (Optimal Asymmetric Encryption Padding)
- Odporúčané
  - PKCS #1 v2.1, RFC 3447
  - NIST SP 800-56B
  - ENISA: Algorithms, Key Sizes and Parameters Report – 2013 recommendations
- Dokázateľná bezpečnosť:
  - IND-CCA2 bezpečná schéma (neskôr)
  - model s náhodným orákulom (pre použité hašovacie funkcie)
  - zložitosť RSA problému
  - redukcia nie je „tesná“

RSA-OAEP (2)

- Zjednodušená verzia
- 2 kolá Feistelovskej siete
- *r* – náhodný reťazec
- MGF – mask generation function, skonštruovaná z hašovacej funkcie
- Výstup OAEP je vstupom do *E*



- Overovanie korektnosti výplne pri dešifrovaní
- Nemožné (efektívne) skonštruovať platný ŠT (t.j. prístup k dešifrovaciemu orákulu je zbytočný)

RSA-KEM

- KEM (Key Encapsulation Mechanism)
- Potreba výplne pri hybridných konštrukciách + dôkazy
- RSA-KEM
  - NIST, ENISA
  - Bezpečné v modeli s náhodným orákulom
- Zjednodušene:

  Symetrický kľúč K = KDF(R), kde R je náhodné zo Z_n

  Posielané: $E(R)$, $E_{sym}(K, M)$

  Dešifrovanie: získanie R, výpočet K a symetrické dešifrovanie
- KDF (Key derivation function)
  - Rôzne konštrukcie – obvykle využívajú hašovacie funkcie

Na zamyslenie (3)

Aký problém je s nasledujúcim kódom pre generovanie prvočísel pre RSA?

```
random.seed(int(time.time()))
rnd_state = random.getstate()

def gen_prime(length = 310):
    global rnd_state
    while True:
        random.setstate(rnd_state)
        a = int(''.join([chr(ord('0')+random.randint(0,9)) for _ in range(length)]))
        if a%2 == 0: a = a+1
        rnd_state = random.getstate()
        if is_prime(a):
            return a
```

Entropia času

- Predpokladáme rovnakú pravdepodobnosť inicializácie počas celého intervalu
- Entropia (v bitoch):

Interval	Granularita 1 sekunda	Granularita 1 milisekunda
1 hodina	11,8	21,8
1 deň	16,4	26,4
1 týždeň	19,2	29,2

Postranné kanály

- Získavanie informácií o súkromnom kľúči alebo OT podľa prejavov implementácie algoritmu
- Timing attack
  - Znalosť vstupu do D a času potrebného na jej výpočet – získanie d alebo faktorizácie
  - Ochrana: zaslepenie (blinding) $D(c) = D(c \cdot x^e) \cdot x^{-1}$
- Ďalšie postranné kanály:
  - Akustické „vyžarovanie“ – zvuky vydávané elektronikou počítača a ich závislosť na súkromnom kľúči (2013)
  - Elektrický potenciál na šasi počítača (2014)
  - atď.

Prax

- Lenstra a kol. (2012)
- 11,4 miliónov RSA modulov
  - X.509 certifikáty, PGP kľúče
- Z toho 26965 (vrátane 10 RSA-2048) sú zraniteľné
  - Spoločné práve jedno prvočíslo
  - Ľahké testovanie pomocou gcd
- Iné pozorovania
  - „Debianovské“ prvočísla
  - Spoločné *n*
  - Malé faktory *n*
  - Nekorektné *n* (napr. prvočíslo)

Prax (2)

- Bernstein a kol. (2013)
- Taiwanská národná databáza „Citizen Digital Certificate“
- Vládou vydávané čipové karty (FIPS 140-2 Level 2, CC EAL4+ pre protection profile BSI-PP-0002-2001)
- Celkovo 3,2 miliónov jedinečných RSA modulov
  - 103 modulov faktorizovaných pomocou gcd (spoločný netriviálny prvočíselný faktor)
  - Analýza „nenáhodnosti“ zistených prvočísel + trocha matematiky (a LLL algoritmus)
  - Celkovo 184 rôznych 1024-bit RSA modulov faktorizovaných
- cca. 10000 kariet bez testov RNG, no-FIPS mód

Príklady a početnosť zistených prvočísel

```
46, 0xc000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000002f9
7, 0xc92424922492924992494924492424922492924992494924492424922492924992494924492424922492924992494924492424922492924992494924492424e5
7, 0xc00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000101ff
6, 0xd24949244924249224929249924949244924249224929249924949244924249224929249924949244924249224929249924949244924249224929249924949d7
4, 0xf6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdbc1
4, 0xdb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6c6e23
4, 0xedb6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b867
3, 0xd08408424210210808428421210810848421421010840842421021080842842121081084842142101084084242102108084284212108108484214210108409 85
2, 0xe00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000f
2, 0xf5ad5ad6d6b56b5a5ad6ad6b6b5ab5adad6bd6b5b5ad5ad6d6b56b5a5ad6ad6b6b5ab5adad6bd6b5b5ad5ad6d6b56b5a5ad6ad6b6b5ab5adad6bd6b5b5ad5d39
2, 0xc28550a128500a14850aa14250a114280a144285a14228501428850a428550a128500a14850aa14250a114280a144285a14228501428850a428550a128500a6f
2, 0xfdefdef7f7bd7bdedef7ef7b7bdebdefef7bf7bdbdefdef7f7bd7bdedef7ef7b7bdebdefef7bf7bdbdefdef7f7bd7bdedef7ef7b7bdebdefef7bf7bdbdefe0b1
2, 0xd24949244924249224929249924949244924249224929249924949244924249224929249924949244924249224929249924949244924249224929249924 84a0f
2, 0xe94a94a5a529529494a54a525294294a4a52a529294a94a5a529529494a54a525294294a4a52a529294a94a5a529529494a54a525294294a4a52a529294b9af5
2, 0xdb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d6db66db6b6dbb6dbdb6ddb6d7015
2, 0xca52a529294a94a5a529529494a54a525294294a4a52a529294a94a5a529529494a54a525294294a4a52a529294a94a5a529529494a54a525294294a4a52a601
2, 0xc00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000002030b
2, 0xd8c68c636318318c8c63c631318c18c6c631631818c68c636318318c8c63c631318c18c6c631631818c68c636318318c8c63c631318c18c6c631631818c69107
2, 0xf18c18c6c631631818c68c636318318c8c63c631318c18c6c631631818c68c636318318c8c63c631318c18c6c631631818c68c636318318c8c63c631318c1907
2, 0xf7bd7bdedef7ef7b7bdebdefef7bf7bdbdefdef7f7bd7bdedef7ef7b7bdebdefef7bf7bdbdefdef7f7bd7bdedef7ef7b7bdebdefef7bf7bdbdefdef7f7bd8289
2, 0xc42142101084084242102108084284212108108484214210108408424210210808428421210810848421421010840842421021080842842121081084842143 69
2, 0xef7bf7bdbdefdef7f7bd7bdedef7ef7b7bdebdefef7bf7bdbdefdef7f7bd7bdedef7ef7b7bdebdefef7bf7bdbdefdef7f7bd7bdedef7ef7b7bdebdefef7bf969
```

Bezpečnosť asymetrického šifrovania

- Cieľ útočníka:
  IND – indistinguishability (neodlíšiteľnosť šifrových textov)
  - Iné možnosti: NM – non-malleability (nepoddajnosť)
- Možnosti útočníka: najsilnejší variant CCA2
- Útočník
  - vstup: verejný kľúč
  - musí byť efektívny (polynomiálny algoritmus)
  - zvolí m_0 a m_1 (dva rôzne OT)
  - útočník dostane $c = E(m_b)$ pre náhodne zvolený bit b
  - útočník je úspešný ak následne vypočíta správnu hodnotu b
  - prístup k dešifrovaciemu orákulu (kedykoľvek, s výnimkou otázky na c po jeho získaní)
- IND-CCA2 bezpečná schéma, ak neexistuje útočník s pravd. úspechu nezanedbateľne väčšou ako 1/2

Bezpečnosť asymetrického šifrovania (2)

- IND-CCA2 = NM-CCA2
  - „robustná“ definícia
- IND-CCA2 bezpečné schémy:
  - RSA-OAEP, predpoklady – zložitosť RSA problému, MGF ako náhodné orákulum
  - Cramer-Shoup, predpoklady – DDH a h.f. odolná voči kolíziám
  - a iné ...

Diskrétny logaritmus

- Úloha: pre dané g a y vypočítať číslo x také, že $g^x = y$
  - g je generátor nejakej grupy G a „násobenie“ príslušná operácia
- Ľahký/ťažký problém v závislosti na G
- Je jedno akú bázu/generátor g zvolíme
  - Nech h, g sú generátory
  - Ak vieme počítať DLOG pre h, tak vieme počítať aj pre g:

    $$h^a = g,\ h^b = y \Rightarrow g^{ba^{-1}} = y$$

  - Nemusí to byť jedno pre konkrétne kryptografické konštrukcie!

ElGamalova schéma

- Kedysi štandardný algoritmus v GPG
  - stále jedna z možností pre asymetrické šifrovanie
- Všeobecný popis – g generátor grupy G, $|G| = q$
- Inicializácia:
  1. Zvolíme náhodné $x \in \{1, 2, ..., q - 1\}$
  2. $y = g^x$

  Verejný kľúč: y, g, q, G (hodnoty g, q, G môžu byť spoločné)

  Súkromný kľúč: x
- Šifrovanie: OT $m \in G$
  - Šifrový text $(r, s) = (g^k, y^k \cdot m)$, kde k je náhodne zvolené zo Z_q
- Dešifrovanie: ŠT (r, s)
  - $s / r^x = y^k \cdot m / r^x = m$

Na zamyslenie (4)

Aké dôsledky majú nasledujúce slabiny pri generovaní *k* na bezpečnosť ElGamalovej schémy:

1. Predikovateľné *k*?
2. Znovupoužitie rovnakého *k*?

ElGamalova schéma – poznámky

- Obvykle v $(Z_p, \cdot)$
- Znáhodnené šifrovanie
  - 1 OT $\rightarrow$ exponenciálne veľa ŠT
- ŠT dlhší ako OT
- Poddajnosť schémy
  - znásobenie dvoch ŠT, prinásobenie k druhej časti ŠT
- Dešifrovanie bez súkromného kľúča $\Leftrightarrow$ riešenie CDH
- CDH – výpočtový Diffieho-Hellmanov problém:
  pre dané g^a, g^b vypočítať g^{ab}
  - nie je ťažší ako problém diskrétneho logaritmu

ElGamalova schéma – poznámky (2)

- Ako kódovať správu (napr. reťazec bitov) ako prvok z G?
- Variant schémy s použitím vhodnej hašovacej funkcie:

  šifrovanie: $(r, s) = (g^k, H(y^k) \oplus m)$

  dešifrovanie: $s \oplus H(r^x)$
- Schéma je stále poddajná, bezpečnosť závisí na CDH a vlastnostiach H

Eliptické krivky

- Obvykle sa začína s eliptickými krivkami nad **R**
- Weierstrassova rovnica ($a, b \in \mathbf{R}$):

  $$y^2 = x^3 + ax + b$$

- Nesingulárne krivky, teda $4a^3 + 27b^2 \neq 0$
- Nesingulárne $\sim x^3 + ax + b$ nemá viacnásobné korene
- Geometrický tvar eliptických kriviek
- Body na eliptickej krivke:

  $$E = \{(x, y) \mid y^2 = x^3 + ax + b\} \cup \{0\}$$

  kde 0 je neutrálny prvok („bod v nekonečne“)

Eliptické krivky – sčítanie bodov

- Na množine E vieme definovať operáciu, ktorú označíme ako „sčítanie“
- Geometrická interpretácia sčítania
- Označenie: $P = (x_P, y_P)$, $P^* = (x_P, -y_P)$
- Sčítanie:
  - $P + P^* = 0$
  - $P + P = R = (x_R, y_R)$ také, že PR^* je dotyčnica el. krivky v P
  - $P + Q = R = (x_R, y_R)$ také, že body P, Q, R^* ležia na priamke
- Takto definované sčítanie má všetky potrebné vlastnosti
- **R** je nepraktické pre kryptografické použitie

Eliptické krivky nad konečným poľom

- Napr. GF(p) = (Z_p, +, ·) pre prvočíslo $p > 3$
  - Aj iné konečné polia môžu byť použité, napr. GF(2^n), rôzne formy el. kriviek, rôzne podmienky a formuly pre sčítanie

$$E = \{(x, y) \mid y^2 = x^3 + ax + b \bmod p\} \cup \{0\}$$

pre $a, b \in Z_p$ spĺňajúce $4a^3 + 27b^2 \neq 0 \pmod p$

- Sčítanie (formuly) „fungujú“ aj teraz (mod p)
- (E,+) je opäť grupa
- Žiadna zmysluplná geometrická interpretácia

Eliptické krivky nad konečným poľom (2)

- Grupa musí byť dostatočne veľká
- Hasseho veta $||E| - p - 1| \leq 2\,p^{1/2}$
- Počítanie presného počtu bodov:
  - Schoofov algoritmus alebo zlepšená jeho verzia Schoofov-Elkiesov-Atkinov algoritmus
- Poznámka: na jednoznačnú reprezentáciu bodu $P = (x_P, y_P)$ stačí hodnota x_P a „znamienko“ y_P
- Pri generovaní/konštrukcii hľadáme el. krivku kde $|E|$ má veľký prvočíselný faktor (príp. je samotné $|E|$ prvočíslo)

Príklad krivky (1): NIST P-256

- Predpokladaná bezpečnosť: „128 bitov“
- Prvočíslo $p = 2^{256} – 2^{224} + 2^{192} + 2^{96} – 1$
- Krivka:

$$y^2 = x^3 – 3x + 41058363725152142129326129780047268409114441015993725554835256314039467401291$$

- Prvočíslený počet bodov:

$$115792089210356248762697446949407573529969552241357603424222590610685120443 69$$

- Používané takmer všade
- „pošramotená“ reputácia kriviek po problémoch s Dual_EC_DRBG

Príklad krivky (2): Curve25519

- Prvočíslo: $p = 2^{255} - 19$
- Krivka:

  $$y^2 = x^3 + 486662\,x^2 + x$$

- Počet bodov $8 \cdot p_1$ pre prvočíslo

  $$p_1 = 2^{252} + 27742317777372353535851937790883648493$$

- Montgomeryho krivka
  - iné formuly pre sčítanie
  - efektívnejšie počítanie
  - možné prepísať do Weierstrassovej formy
- Neštandardná krivka
- Používaná v rôznych aplikáciách (OpenSSH, iOS, TextSecure, atď.)

Diskrétny logaritmus na el. krivkách

- (E, +) – grupa bodov eliptickej krivky
- Bod $P \in E$
- $kP = P + P + ... + P$ (k krát), pre celé číslo $k \geq 0$

- DLOG: pre daný bod kP vypočítať k
- CDH: pre dané body aP a bP vypočítať $(ab)P$

EC verzia ElGamalovej schémy

- (E,+) – grupa bodov eliptickej krivky
- $G \in E$ – generátor podgrupy E, ord(G) = q (prvočíslo)
- Súkromné kľúče: náhodné $x \in Z_q$
- Verejný kľúč: $Y = xG$
- Šifrovanie $M \in E$:

$$(R, S) = (kG, kY + M) \text{ pre náhodné } k \in Z_q$$

- Dešifrovanie $(R, S) \in E \times E$:

$$S - xR = (kY + M) - xR = (kx)G + M - (kx)G = M$$

- „prepísanie“ schémy do (E, +)
- Reprezentácia OT ako bodu eliptickej krivky

ECIES-KEM (1)

- EC Integrated Encryption Scheme
- príklad KEM konštrukcie založenej na el. krivkách
  - Doporučené ENISA, súčasť ISO/IEC 18033-2

- Zjednodušený popis
- (E, +), $G \in E$, ord(G) = q (prvočíslo)
- Súkromný a verejný kľúč: $x \in Z_q$ a $Y = xG$

ECIES-KEM (2)

- Šifrovanie (výstupom je kľúč (OT) a šifrový text):
  1. $C = kG$ pre náhodné $k \in Z_q$
  2. K = KDF(C || $[kY]_{\mathbf{x}}$) kde $[?]_{\mathbf{x}}$ označuje x-ovú súradnicu bodu
  3. Výstup: (K, C)

- Dešifrovanie (vstup: C)
  1. $Q = xC$ (= $(xk)G = k(xG) = kY$)
  2. Výstup: K = KDF(C || $[kY]_{\mathbf{x}}$)

- Bezpečnosť: Gap-ECDH v modeli s náhodným orákulom
  - Gap-ECDH: riešiť CDH ak máme k dispozícii orákulum pre DDH

ISO/IEC 18033-2 Asymmetric ciphers

- Hybridné schémy založené na ElGamalovej schéme:
  - ECIES-HC, PSEC-HC, ACE-HC
- Schémy založené na RSA:
  - RSA-HC (RSA-KEM)
  - RSAES (RSA-OAEP)

- HIME(R):
  - Schéma založená na probléme faktorizácie
  - Navrhnutá spol. Hitachi
  - $N = p^2q$
  - Šifrovanie: OAEP a $E(m) = m^2 \bmod N$

Ďakujem za pozornosť