CTF на Физтехе

Занятие 7

План

- Асимметричное шифрование
  - Протокол Диффи-Хеллмана
  - RSA
  - Цифровая подпись
  - Сертификат открытого ключа
- Хеширование
  - MD5, SHA-1, …
  - Length-extension attack

Асимметричное шифрование

Асимметричное шифрование

- Для шифрования и расшифровки используются разные ключи
- Открытый ключ известен всем и используется для шифрования
- Закрытый ключ используется для расшифровки
- По открытому ключу нельзя (вычислительно сложно) восстановить закрытый

Теоремы Ферма и Эйлера

Fermat's Little Theorem

If p is a prime and a is any integer, then $a^p \equiv a \,(mod\ p)$.

Definition

The **Euler φ-Function** is defined on the set of positive integers as follows. For each positive integer n,
$\varphi(n)$ is the number of integers a satisfying $1 \leqslant a \leqslant n$ and $(a, n) = 1$.

Recall: The group of units of the ring $\mathbb{Z}_n$ of integers mod n is

$$\mathbb{Z}_n^* = \{[a]_n \mid 1 \leqslant a \leqslant n \text{ and } (a, n) = 1\},$$

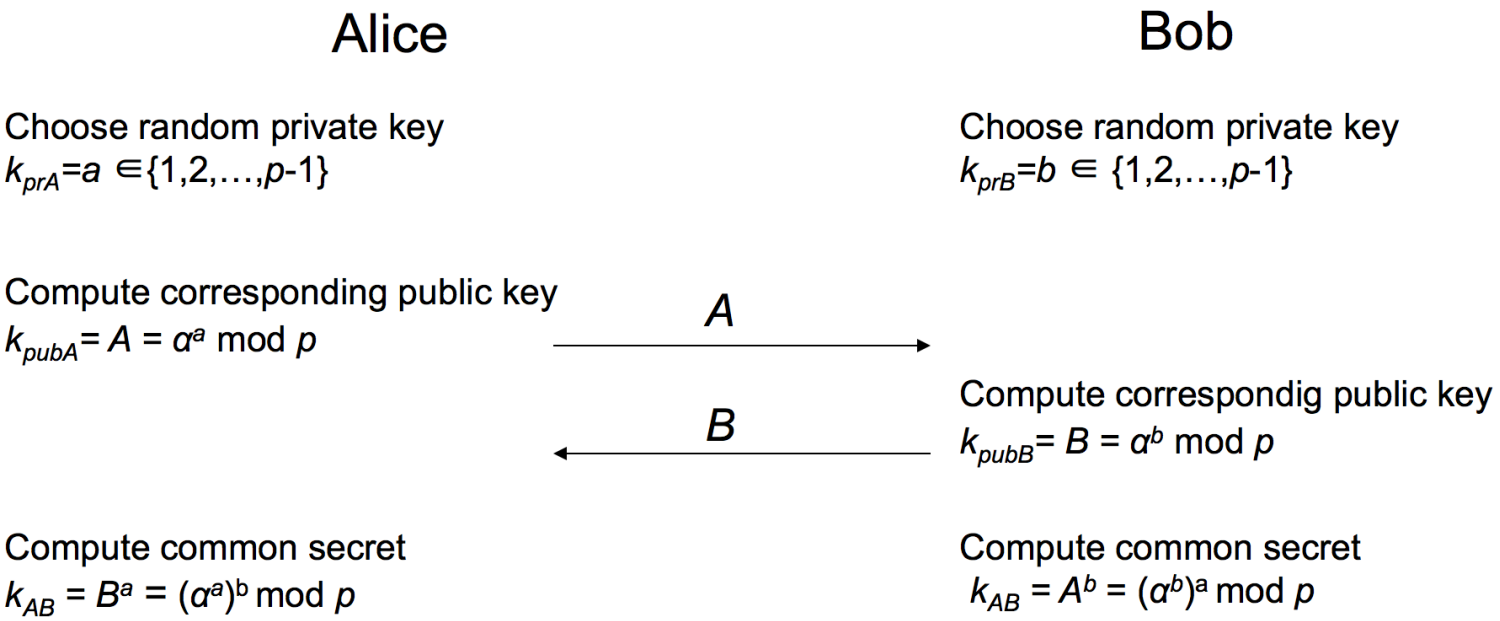
hence $|\mathbb{Z}_n^*| = \varphi(n)$.

Euler's Theorem

If n is a positive integer and a is any integer such that $(a, n) = 1$, then

$$a^{\varphi(n)} \equiv 1 \,(mod\ n).$$

Протокол Диффи-Хеллмана



Протокол Диффи-Хеллмана: пример

Domain parameters p=29, α=2

Alice		Bob
Choose random private key $k_{prA} = a = 5$		Choose random private key $k_{prB} = b = 12$
Compute corresponding public key $k_{pubA} = A = 2^5 = 3 \bmod 29$	A →	
	← B	Compute correspondig public key $k_{pubB} = B = 2^{12} = 7 \bmod 29$
Compute common secret $k_{AB} = B^a = 7^5 = 16 \bmod 29$		Compute common secret $k_{AB} = A^b = 3^{12} = 16 \bmod 29$

RSA

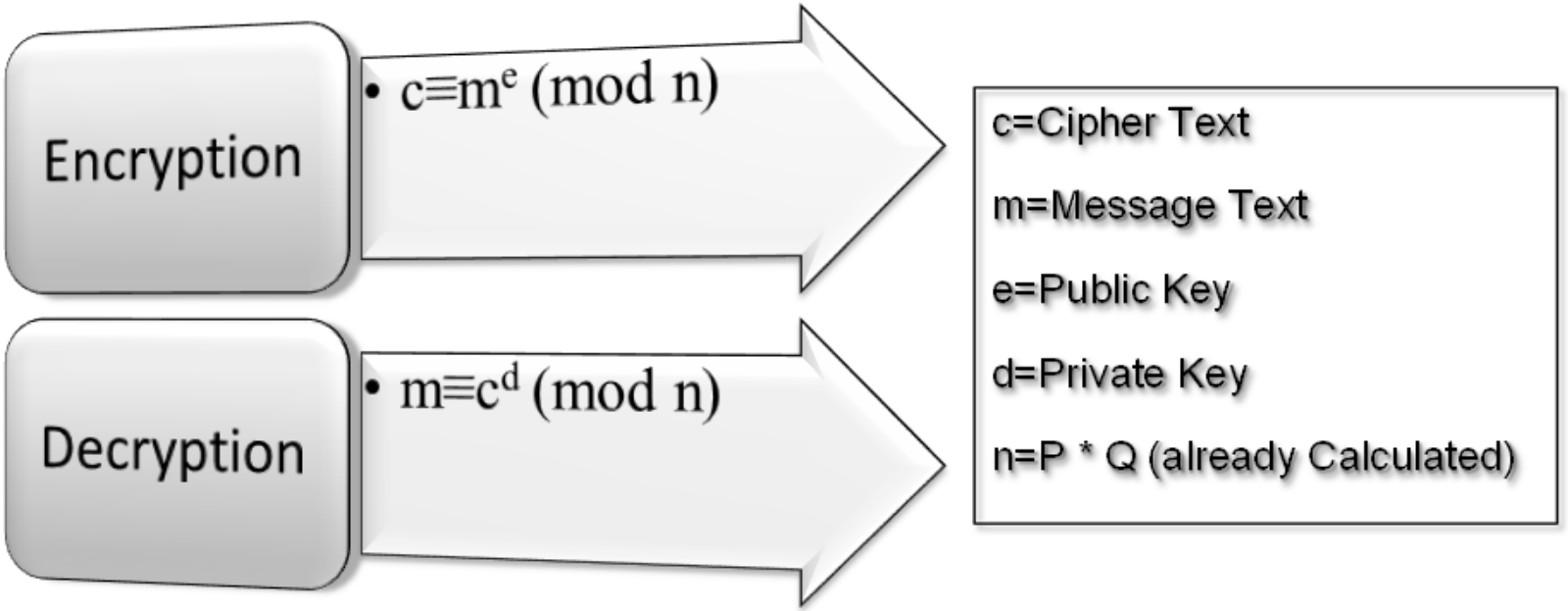
- Pick two large primes p and q
- Calculate n = pq
- Pick e such that it is relatively prime to phi(n) = (q-1)(p-1)
  - "Euler's Totient Function"
- $d \sim= e^{-1}$ mod phi(n)

  or

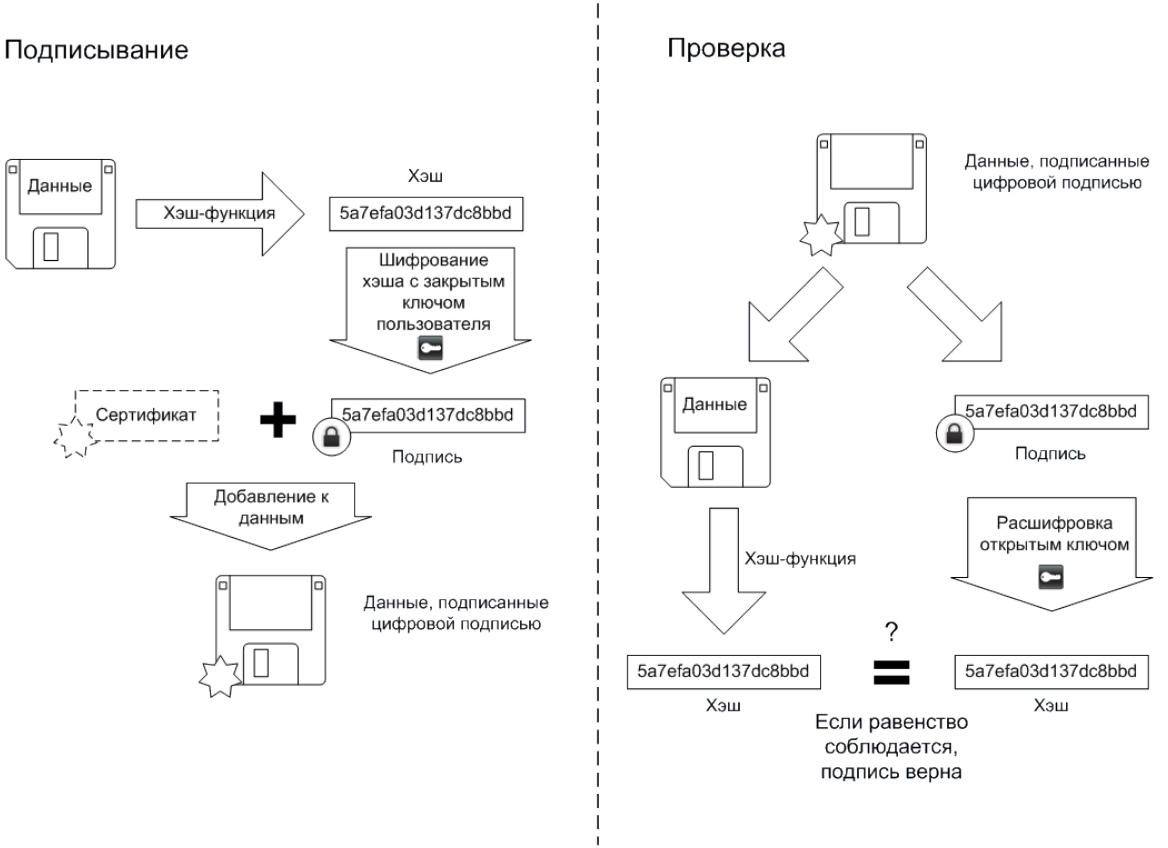
  de mod phi(n) = 1

1. p=3, q=11
2. n = 3*11 = 33
3. phi(n) = (2*10) = 20
4. e = 7 | GCD(20,7) = 1
   "Euclid's Algorithm"
5. d = 7-1 mod 20
   d = 7 mod 20 = 1
   d = 3

RSA

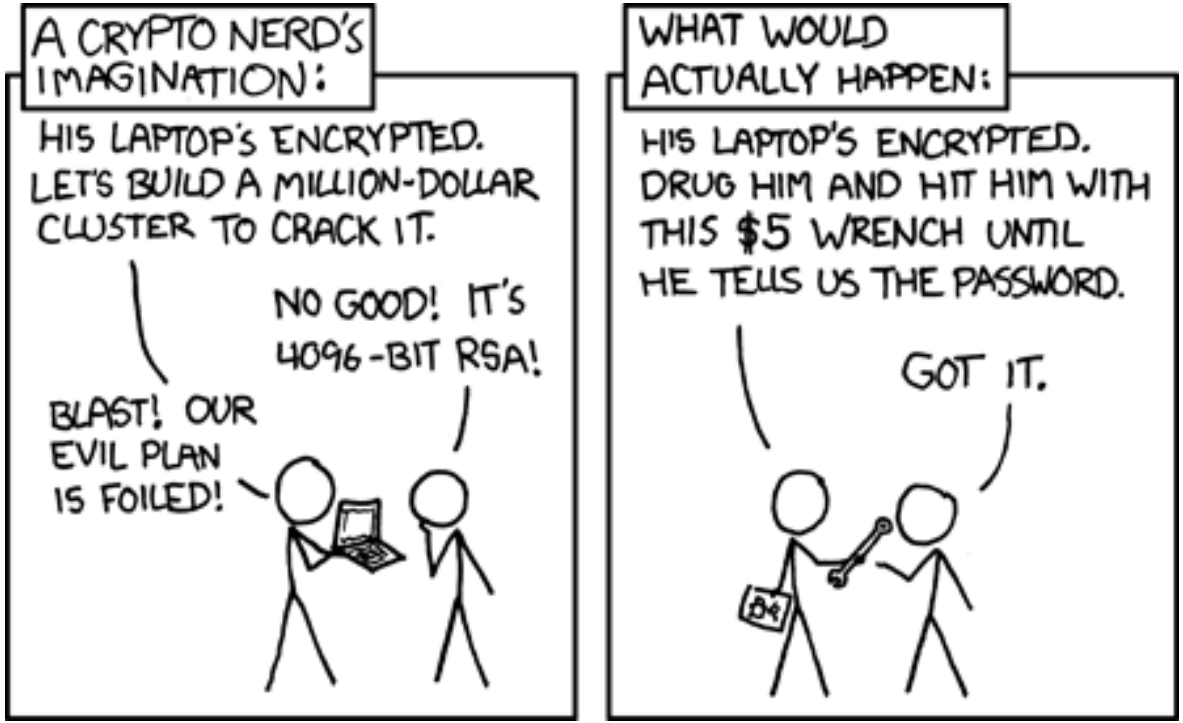

Encryption
• $c \equiv m^e \pmod{n}$
Decryption
• $m \equiv c^d \pmod{n}$
c=Cipher Text
m=Message Text
e=Public Key
d=Private Key
n=P * Q (already Calculated)

Электронная цифровая подпись



Сертификат открытого ключа

- https://upload.wikimedia.org/wikipedia/commons/9/96/Usage-of-Digital-Certificate.svg


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Хеширование

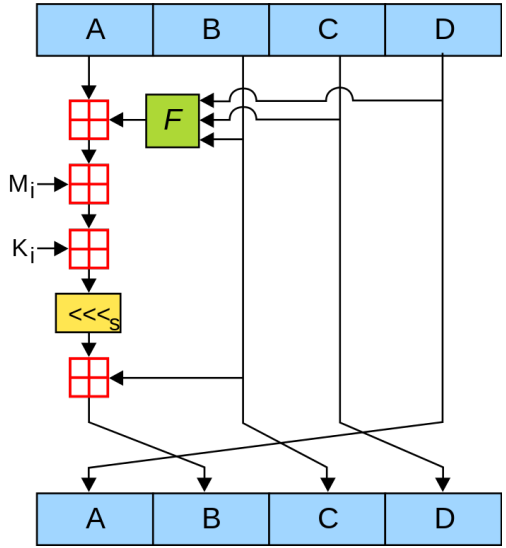
Криптографическая хеш-функция

- F(message) = hash
  - MD5(“Hello world!”) = 86fb269d190d2c85f6e0468ceca42a20
  - SHA-1(...) = d3486ae9136e7856bc42212385ea797094475802
- Применения
  - Цифровая подпись
  - Хранение паролей
  - Контрольная сумма
  - ...
- Возможны коллизии

MD5

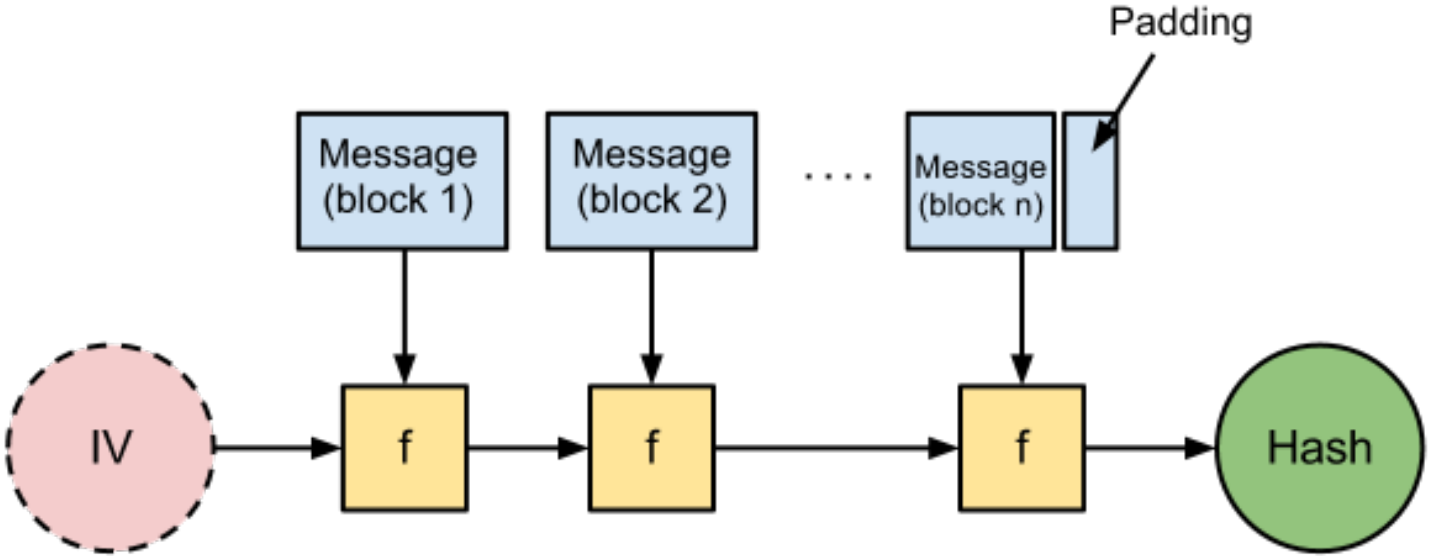
1. Выравнивание: к сообщению дописывают единичиный бит (байт 0x80), а затем необходимое число нулевых бит, чтобы новый размер сообщения был сравним с 448 по модулю 512.
2. В оставшиеся 64 бита дописывают 64-битное представление длины данных до выравнивания.
3. Инициализируется внутреннее 128-битное состояние.
4. Данные разбиваются на блоки по 512 бит, которые обрабатываются последовательно. В процессе обработки модифицируется внутреннее состояние.

- Основная часть алгоритма обрабатывает 128-битные блоки данных (рисунок справа).
- https://en.wikipedia.org/wiki/MD5#Pseudocode



- A, B, C, D - блоки по 32 бита
- F - нелинейная функция
- Mi - 32-битный блок сообщения
- Ki - 32-битная константа

Процесс вычисления



Сравнение хеш-функций

Comparison of SHA functions

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations	Security (bits)	Example Performance[31] (MiB/s)
MD5 (as reference)		128	128 (4 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or	<64 (collisions found)	335
SHA-0		160	160 (5 × 32)	512	$2^{64} - 1$	80	And, Xor, Rot, Add (mod 2^{32}), Or	<80 (collisions found)	-
SHA-1		160	160 (5 × 32)	512	$2^{64} - 1$	80		<80 (theoretical attack[32] in 2^{61})	192
SHA-2	*SHA-224*	224	256 (8 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112	139
	SHA-256	256						128	
	SHA-384	384	512 (8 × 64)	1024	$2^{128} - 1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192	154
	SHA-512	512						256	
	SHA-512/224	224						112	
	SHA-512/256	256						128	
SHA-3	*SHA3-224*	224	1600 (5 × 5 × 64)	1152	Unlimited	24	And, Xor, Rot, Not	112	-
	SHA3-256	256		1088				128	
	SHA3-384	384		832				192	
	SHA3-512	512		576				256	
	SHAKE128	*d* (arbitrary)		1344				min (*d*/2, 128)	-
	SHAKE256	*d* (arbitrary)		1088				min (*d*/2, 256)	

Length extension attack

Original Data: count=10&lat=37.351&user_id=1&long=-119.827&waffle=eggo

Original Signature: 6d5f807e23db210bc254a28be2d6759a0f5f5d99

Desired New Data: count=10&lat=37.351&user_id=1&long=-119.827

&waffle=eggo&waffle=liege

signature = hash(secret + data)

Length extension attack

New Data: count=10&lat=37.351&user_id=1&long=-119.827
&waffle=eggo\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\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\x02\x28&waffle=liege

New Signature: 37c9dcaef9a370feb2daf97211a1bbb273812753

Соль

- salt = случайная строка
- Вместо hash(password) хранить hash(salt + password)
- Затрудняет восстановление пароля

Password cracking tools

- hashcat: http://hashcat.net/oclhashcat/
- John the Ripper: http://www.openwall.com/john/

python + bash

```
>>> import hashlib
>>> m = hashlib.md5()
>>> m.update('Hello world!')
>>> m.hexdigest()
'86fb269d190d2c85f6e0468ceca42a20'
```

```
>>> import hashlib
>>> m = hashlib.sha1()
>>> m.update('Hello world!')
>>> m.hexdigest()
'd3486ae9136e7856bc42212385ea797094475802'
```

```
$ echo -n "Hello world!" | md5sum
86fb269d190d2c85f6e0468ceca42a20  -
$ echo -n "Hello world!" | sha1sum
d3486ae9136e7856bc42212385ea797094475802  -
```

Вопросы?