Krakow Quantum Informatics Seminar (KQIS)
When the asymmetric cryptography will be outdated?

Paweł Topa, Ph.D.

June 16, 2020

Cryptology

Science concerned with data communication and storage in secure and usually secret form.

- Cryptography — secret writing (or other methods of hiding information)
- Cryptanalysis — reading secret messages (without knowledge of encryption key).

Past → Today → Future

- **Classic cryptography** — manipulation (substituting and permuting) of symbols (letters), codebooks.
- **Modern cryptography** — computers and mathematics.
- **Quantum cryptography** — quantum physics and quantum computers.
  - quantum cryptanalysis (Shor's algorithm),
  - key exchange methods,
  - detection of security breeches.
- **Postquantum cryptography** — algorithms resistant to quantum cryptanalysis

Public-key cryptography

- Symmetric cryptography:

Public-key cryptography

- Symmetric cryptography:
  Encryption and decryption uses the same key.

Public-key cryptography

- Symmetric cryptography:
  Encryption and decryption uses the same key.
- Public-key (asymmetric) cryptography:

Public-key cryptography

- Symmetric cryptography:
  Encryption and decryption uses the same key.
- Public-key (asymmetric) cryptography:
  Key for encryption (public) is different than key for decryption (private).

Public-key cryptography

- Symmetric cryptography:
  Encryption and decryption uses the same key.
- Public-key (asymmetric) cryptography:
  Key for encryption (public) is different than key for decryption (private).
- Encryption/decryption and digital signature.

Public-key cryptography

- Symmetric cryptography:
  Encryption and decryption uses the same key.
- Public-key (asymmetric) cryptography:
  Key for encryption (public) is different than key for decryption (private).
- Encryption/decryption and digital signature.
- Public key is used to encrypt and verify signature.

Public-key cryptography

- Symmetric cryptography:
  Encryption and decryption uses the same key.
- Public-key (asymmetric) cryptography:
  Key for encryption (public) is different than key for decryption (private).
- Encryption/decryption and digital signature.
- Public key is used to encrypt and verify signature.
- Private key is used to decrypt and to create digital signature.

History of public-key cryptography

1. Diffie-Hellman key exchange protocol: 1976
2. RSA algorithm: 1977
3. ElGamal cryptosystem: 1985
4. British Intelligence (GCHQ): 1970 – 1973 (classified for 25 years)

Key exchange Diffie-Hellman-Merkle protocol

- First asymmetric cryptography algorithm (1976).
- Invented by Martin Hellman and Whitfield Diffi, and independently by Ralph Merkle.
- Only key exchange.
- Security based on the difficulty of calculating discrete logarithms (Discrete Logarithm Problem).

How it works?

1. Alice and Bob select large prime number n and a number g such that $1 < g < n$.

.

How it works?

1. Alice and Bob select large prime number n and a number g such that $1 < g < n$.
2. Alice chooses large number a and send to Bob result of calculation $x = g^a \text{mod } n$.

.

How it works?

1. Alice and Bob select large prime number n and a number g such that $1 < g < n$.
2. Alice chooses large number a and send to Bob result of calculation $x = g^a \text{mod } n$.
3. Bob also chooses large number b and send Alice result of calculation $y = g^b \text{mod } n$.

.

How it works?

1. Alice and Bob select large prime number n and a number g such that $1 < g < n$.
2. Alice chooses large number a and send to Bob result of calculation $x = g^a \text{mod } n$.
3. Bob also chooses large number b and send Alice result of calculation $y = g^b \text{mod } n$.
4. Alice computes $k_A = y^a \text{mod } n$.

.

How it works?

1. Alice and Bob select large prime number n and a number g such that $1 < g < n$.
2. Alice chooses large number a and send to Bob result of calculation $x = g^a \text{mod } n$.
3. Bob also chooses large number b and send Alice result of calculation $y = g^b \text{mod } n$.
4. Alice computes $k_A = y^a \text{mod } n$.
5. Bob computes $k_B = x^b \text{mod } n$.

.

How it works?

1. Alice and Bob select large prime number n and a number g such that $1 < g < n$.
2. Alice chooses large number a and send to Bob result of calculation $x = g^a \text{mod } n$.
3. Bob also chooses large number b and send Alice result of calculation $y = g^b \text{mod } n$.
4. Alice computes $k_A = y^a \text{mod } n$.
5. Bob computes $k_B = x^b \text{mod } n$.
6. Alice and Bob now share a secret (number)$k_A = k_B = g^{ab} \text{mod } n$.

.

How it works?

1. Alice and Bob select large prime number n and a number g such that $1 < g < n$.
2. Alice chooses large number a and send to Bob result of calculation $x = g^a \text{mod } n$.
3. Bob also chooses large number b and send Alice result of calculation $y = g^b \text{mod } n$.
4. Alice computes $k_A = y^a \text{mod } n$.
5. Bob computes $k_B = x^b \text{mod } n$.
6. Alice and Bob now share a secret (number)$k_A = k_B = g^{ab} \text{mod } n$.

Eve (eavesdropper) knows x, y, g, n

.

How it works?

1. Alice and Bob select large prime number n and a number g such that $1 < g < n$.
2. Alice chooses large number a and send to Bob result of calculation $x = g^a \text{mod } n$.
3. Bob also chooses large number b and send Alice result of calculation $y = g^b \text{mod } n$.
4. Alice computes $k_A = y^a \text{mod } n$.
5. Bob computes $k_B = x^b \text{mod } n$.
6. Alice and Bob now share a secret (number)$k_A = k_B = g^{ab} \text{mod } n$.

Eve (eavesdropper) knows x, y, g, n

.

How it works?

1. Alice and Bob select large prime number n and a number g such that $1 < g < n$.
2. Alice chooses large number a and send to Bob result of calculation $x = g^a \text{mod } n$.
3. Bob also chooses large number b and send Alice result of calculation $y = g^b \text{mod } n$.
4. Alice computes $k_A = y^a \text{mod } n$.
5. Bob computes $k_B = x^b \text{mod } n$.
6. Alice and Bob now share a secret (number)$k_A = k_B = g^{ab} \text{mod } n$.

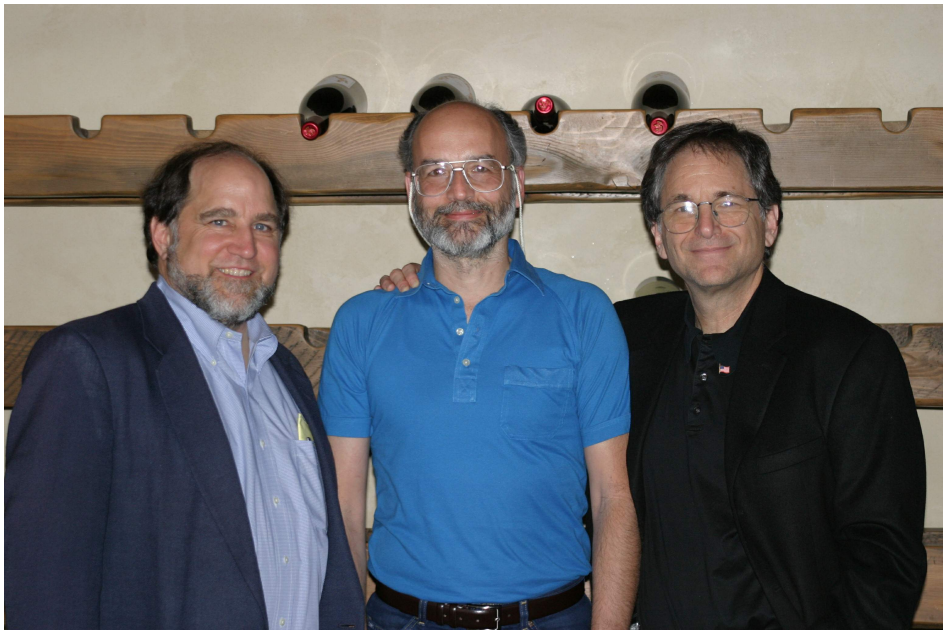
Eve (eavesdropper) knows x, y, g, $n \rightarrow$ In order to know a and b she must calculate discrete logarithm.

RSA Cryptosystem

- Rivest, Shamir, Adleman (1977 r.)

RSA Cryptosystem

- Rivest, Shamir, Adleman (2003 r.)



- Security based on the difficulty of integer factorization.

RSA — Key generation

Key generation

1. Choose two distinct large prime numbers p i q,

RSA — Key generation

Key generation

1. Choose two distinct large prime numbers p i q,
2. Choose an integer $e > 1$ and e is coprime with $(p-1)(q-1)$,

RSA — Key generation

Key generation

1. Choose two distinct large prime numbers p i q,
2. Choose an integer $e > 1$ and e is coprime with $(p-1)(q-1)$,
3. Compute $d = e^{-1} \mathrm{mod}(p-1)(q-1)$,

RSA — Key generation

Key generation

1. Choose two distinct large prime numbers p i q,
2. Choose an integer $e > 1$ and e is coprime with $(p-1)(q-1)$,
3. Compute $d = e^{-1} \mathrm{mod}(p-1)(q-1)$,
4. Compute $n = pq$.

RSA — Key generation

Key generation

1. Choose two distinct large prime numbers p i q,
2. Choose an integer $e > 1$ and e is coprime with $(p-1)(q-1)$,
3. Compute $d = e^{-1} \mathrm{mod}(p-1)(q-1)$,
4. Compute $n = pq$.

Public key: (n, e).

RSA — Key generation

Key generation

1. Choose two distinct large prime numbers p i q,
2. Choose an integer $e > 1$ and e is coprime with $(p-1)(q-1)$,
3. Compute $d = e^{-1} \mathrm{mod}(p-1)(q-1)$,
4. Compute $n = pq$.

Public key: (n, e).
Private key: (n, d).

Encryption with RSA

1. Message in divided into blocks of size not greater that key size (i.e. 2048 bits). Message is treated as integer number $m_i < n$.
2. Encryption:
   $c_i = m_i^e \mod n$
3. Decryption:
   $m_i = c_i^d \mod n$.
4. RSA encryption in this form is insecure — textbook RSA

Digital signature

Digital signature can be implemented using RSA cryptosystem
Author:

Digital signature

Digital signature can be implemented using RSA cryptosystem
Author:

1. Applies digest function to the message: $h_m = \text{hash}(m)$.

Digital signature

Digital signature can be implemented using RSA cryptosystem
Author:

1. Applies digest function to the message: $h_m = \text{hash}(m)$.
2. Signature is generated by encrypting digest with private key: $sig_m = h_m^d \text{mod } n$

Digital signature

Digital signature can be implemented using RSA cryptosystem
Author:

1. Applies digest function to the message: $h_m = \text{hash}(m)$.
2. Signature is generated by encrypting digest with private key: $sig_m = h_m^d \text{mod } n$
3. Send message and signature to receiver

Digital signature

Digital signature can be implemented using RSA cryptosystem
Author:

1. Applies digest function to the message: $h_m = \text{hash}(m)$.
2. Signature is generated by encrypting digest with private key: $sig_m = h_m^d \text{mod } n$
3. Send message and signature to receiver

Receiver:

Digital signature

Digital signature can be implemented using RSA cryptosystem
Author:

1. Applies digest function to the message: $h_m = \text{hash}(m)$.
2. Signature is generated by encrypting digest with private key: $sig_m = h_m^d \text{mod } n$
3. Send message and signature to receiver

Receiver:

1. Encrypts signature with public key: $sig_m^e = h_m^{ed}$

Digital signature

Digital signature can be implemented using RSA cryptosystem
Author:

1. Applies digest function to the message: $h_m = \text{hash}(m)$.
2. Signature is generated by encrypting digest with private key: $sig_m = h_m^d \text{mod } n$
3. Send message and signature to receiver

Receiver:

1. Encrypts signature with public key: $sig_m^e = h_m^{ed}$
2. Compute digest for received message.

Digital signature

Digital signature can be implemented using RSA cryptosystem
Author:

1. Applies digest function to the message: $h_m = \text{hash}(m)$.
2. Signature is generated by encrypting digest with private key: $sig_m = h_m^d \text{mod } n$
3. Send message and signature to receiver

Receiver:

1. Encrypts signature with public key: $sig_m^e = h_m^{ed}$
2. Compute digest for received message.
3. Compare $h_m^{ed} = h_m$ and verify the signature.

Security of RSA

1. Cryptanalyst wants to know the private exponent d.

Security of RSA

1. Cryptanalyst wants to know the private exponent d.
2. Easy to compute d when p i q are known.

Security of RSA

1. Cryptanalyst wants to know the private exponent d.
2. Easy to compute d when p i q are known.
3. Factorization of n is computationally not feasible.

Security of RSA

1. Cryptanalyst wants to know the private exponent d.
2. Easy to compute d when p i q are known.
3. Factorization of n is computationally not feasible.
4. Some method are usable when key components fulfill some conditions.

Security of RSA

1. Cryptanalyst wants to know the private exponent d.
2. Easy to compute d when p i q are known.
3. Factorization of n is computationally not feasible.
4. Some method are usable when key components fulfill some conditions.
5. Quantum computing and Shor algorithm.

Why the RSA is so important?

- Symmetric cryptography is secure but how to exchange keys?
- Public-key cryptography is extremely slow.
- Solution:
  1. public-key cryptography for key exchange/negotiation
  2. symmetric cryptography for data encryption/decryption.
- How to ensure that public key is correct?
  1. Web of Trust like in PGP
  2. Public Key Infrastructure

SSL/TLS protocol

- *Secured Socket Layer*

SSL/TLS protocol

- *Secured Socket Layer*
- *Netscape* 1994 (SSL 3.0, 1995)

SSL/TLS protocol

- *Secured Socket Layer*
- *Netscape* 1994 (SSL 3.0, 1995)
- TLS 1.0 *Transport Layer Security* 1996

SSL/TLS protocol

- *Secured Socket Layer*
- *Netscape* 1994 (SSL 3.0, 1995)
- TLS 1.0 *Transport Layer Security* 1996
- TLS 1.2, RFC 5246, Sierpień 2008.

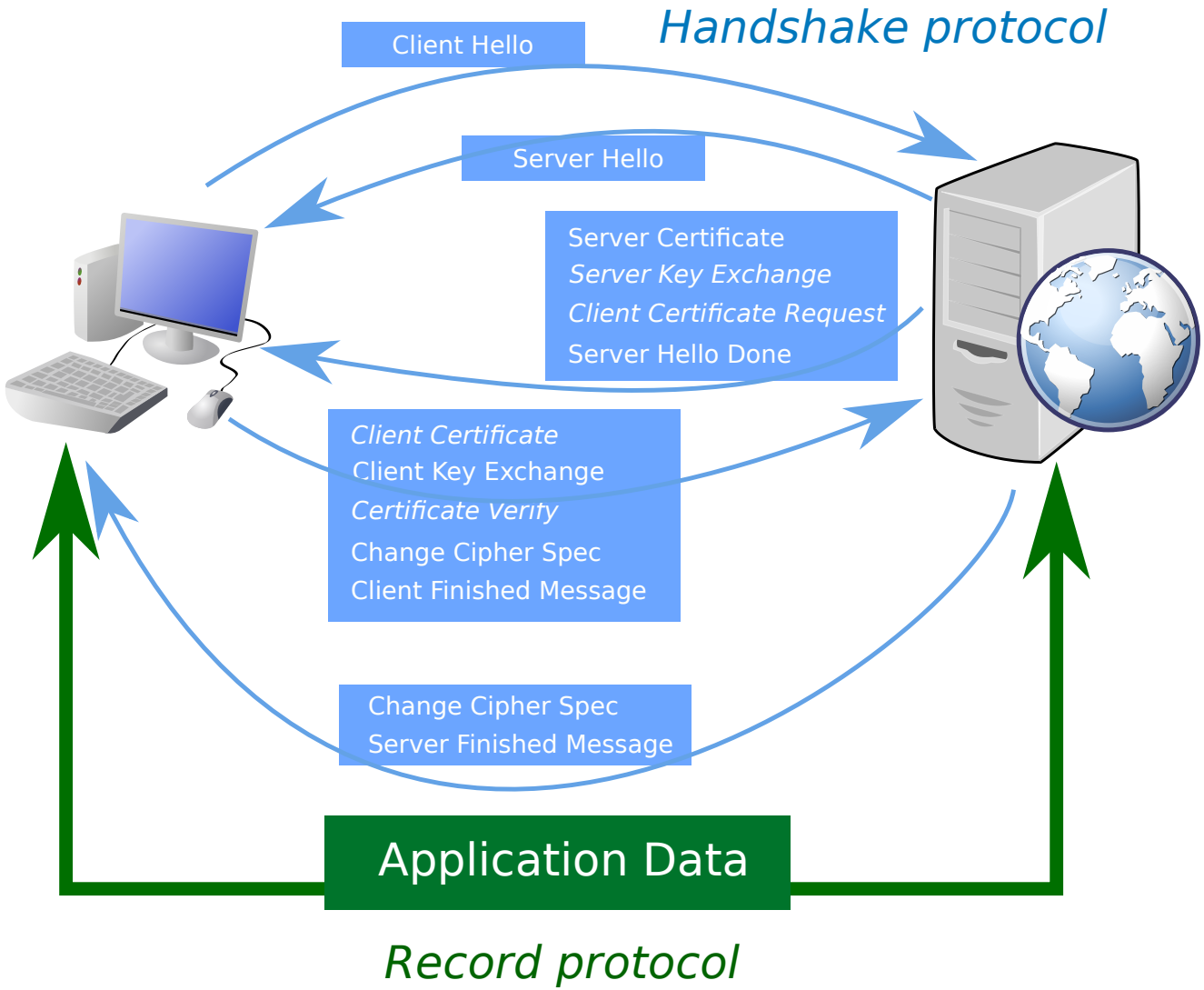
SSL/TLS protocol

- *Secured Socket Layer*
- *Netscape* 1994 (SSL 3.0, 1995)
- TLS 1.0 *Transport Layer Security* 1996
- TLS 1.2, RFC 5246, Sierpień 2008.
- TLS 1.3, 2018.

SSL/TLS protocol

- *Secured Socket Layer*
- *Netscape* 1994 (SSL 3.0, 1995)
- TLS 1.0 *Transport Layer Security* 1996
- TLS 1.2, RFC 5246, Sierpień 2008.
- TLS 1.3, 2018.
- OpenSSL, GnuTLS — open implementations

Handshake protocol w SSL/TLS



Client/Server Hello messages

Client Hello

- Highest supported TLS version.
- 32B string: 4B[client's time] + 28B[random number] — will be used to generate session key.
- Session ID — only if session is restarted.
- Supported cryptographic suite, np. TLS_ECDHE_RSA_WITH _AES_128_GCM_SHA256.

Server Hello

- $\Leftarrow$.
- $\Leftarrow$.
- Session ID:
  - new ID or
  - ID existing session or
  - null — not supported
- Cryptographic suite. Server chooses the strongest.

Cipher suites

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256.

ECDHE — *Elliptic curve Diffie–Hellman* for session key

RSA — for certificates and digital signature.

AES_128_GCM — AES encryption, 128b block, GCM (Galois/Counter Mode) cipher mode.

SHA256 — digest function.

Integer factorization

- Input date: integer N with d digits.

Integer factorization

- Input date: integer N with d digits.
- Task: find all prime factors of N.

Integer factorization

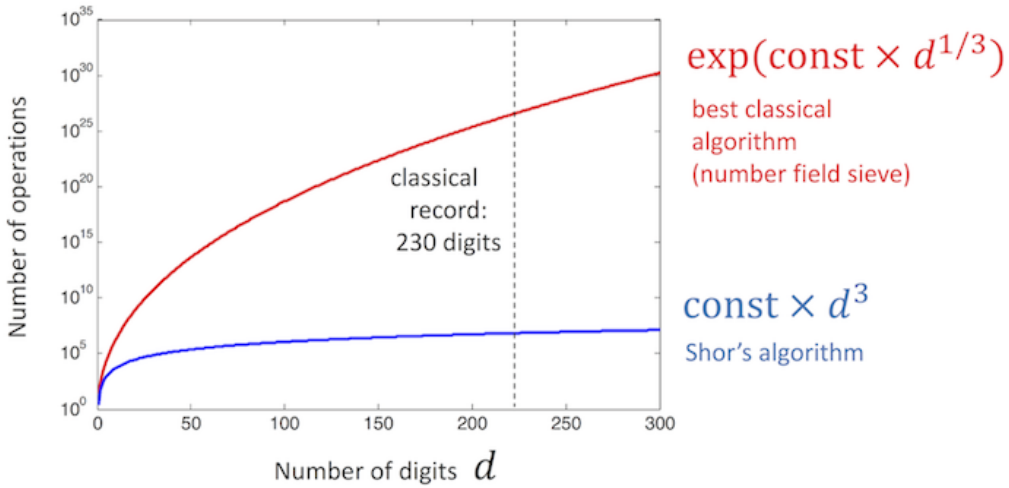
- Input date: integer N with d digits.
- Task: find all prime factors of N.
- Breaking RSA requires finding only two prime factors

Integer factorization

- Input date: integer N with d digits.
- Task: find all prime factors of N.
- Breaking RSA requires finding only two prime factors
- Best conventional algorithm has sub-exponential computational complexity: General Number Field Sieve (GNFS)

Integer factorization

- Input date: integer N with d digits.
- Task: find all prime factors of N.
- Breaking RSA requires finding only two prime factors
- Best conventional algorithm has sub-exponential computational complexity: General Number Field Sieve (GNFS)
- Shor's algorithm has polynomial complexity



Source: IBM Quantum Experience

Shor algorithm

Peter Shor, 1994

1. Classic (conventional) part: integer factorization to order-finding problem
2. Quantum part: solving order-finding problem

Problem of order-finding

- Modulo operation:

$$a \equiv b \pmod{N}$$

  two integers a, b are congruent modulo N if there is an integer k such that $a - b = kn$
- Let's $\mathbb{Z}_N = \{0, \ldots N-1\}$ be a set defined by operations modulo N
- Let's $\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N : GCD(a, N) = 1\}$ be the multiplicative group of integers modulo N (appropriate operations are defined)
- For $a \in \mathbb{Z}_N^*$ the *order of* a in $\mathbb{Z}_N^*$ is the smallest positive integer r such that:

$$a^r \equiv 1 \pmod{N}$$

Problem of order-finding: example

Let's $N = 15$ and $a = 7$: $7^x \bmod N$

- $7^1 \pmod{15} = 7$
- $7^2 \pmod{15} = 4$
- $7^3 \pmod{15} = 13$
- $7^4 \pmod{15} = 1$ (Eureka!)
- $7^5 \pmod{15} = 16807 \pmod{15} = 7$
- $7^6 \pmod{15} = 117649 \pmod{15} = 4$
- $7^7 \pmod{15} = 823543 \pmod{15} = 13$
- $7^8 \pmod{15} = 5764801 (\mod 15) = 1$ Once again!

then order of 7 in $\mathbb{Z}_{15}^*$ is $r = 4$:
Let's denote the above expression as function $f^7(x) = 7^x \ (mod\ 15)$. It is periodical: $f^7(x + 4) = f^7(x)$.

From integer factorization to order-finding problem

1. Let's assume that N has two prime factors p_1 i p_2: $N = p_1 \times p_2$

From integer factorization to order-finding problem

1. Let's assume that N has two prime factors p_1 i p_2: $N = p_1 \times p_2$
2. Pick a random integer a: $2 \geq a \geq N - 1$.

From integer factorization to order-finding problem

1. Let's assume that N has two prime factors p_1 i p_2: $N = p_1 \times p_2$
2. Pick a random integer a: $2 \geq a \geq N - 1$.
3. Let's assume that $GCD(N, a) = 1$.

From integer factorization to order-finding problem

1. Let's assume that N has two prime factors p_1 i p_2: $N = p_1 \times p_2$
2. Pick a random integer a: $2 \geq a \geq N - 1$.
3. Let's assume that $GCD(N, a) = 1$.
4. Let's r is period of $f^a(x) = a^x \bmod N$ **magically calculated**

From integer factorization to order-finding problem

1. Let's assume that N has two prime factors p_1 i p_2: $N = p_1 \times p_2$
2. Pick a random integer a: $2 \geq a \geq N - 1$.
3. Let's assume that $GCD(N, a) = 1$.
4. Let's r is period of $f^a(x) = a^x \bmod N$ **magically calculated**
5. If r is odd go back to step 2.

From integer factorization order-finding problem t

1. $a^r \equiv 1 \pmod{N}$

From integer factorization order-finding problem t

1. $a^r \equiv 1 \pmod{N}$
2. Let's reformulate: $a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$

From integer factorization order-finding problem t

1. $a^r \equiv 1 \pmod{N}$
2. Let's reformulate: $a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$
3. We note that $a^{\frac{r}{2}} - 1$ is not multiply of N ($\frac{r}{2}$ is not a period)

From integer factorization order-finding problem t

1. $a^r \equiv 1 \pmod{N}$
2. Let's reformulate: $a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$
3. We note that $a^{\frac{r}{2}} - 1$ is not multiply of N ($\frac{r}{2}$ is not a period)
4. Let's assume that $a^{\frac{r}{2}} + 1$ is not multiply of N too.

From integer factorization order-finding problem t

1. $a^r \equiv 1 \pmod{N}$
2. Let's reformulate: $a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$
3. We note that $a^{\frac{r}{2}} - 1$ is not multiply of N ($\frac{r}{2}$ is not a period)
4. Let's assume that $a^{\frac{r}{2}} + 1$ is not multiply of N too.
5. $a^{\frac{r}{2}} \pm 1$ are not multiply of N but they product is.

From integer factorization order-finding problem t

1. $a^r \equiv 1 \pmod{N}$
2. Let's reformulate: $a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$
3. We note that $a^{\frac{r}{2}} - 1$ is not multiply of N ($\frac{r}{2}$ is not a period)
4. Let's assume that $a^{\frac{r}{2}} + 1$ is not multiply of N too.
5. $a^{\frac{r}{2}} \pm 1$ are not multiply of N but they product is.
6. Prime factors: $p_1(p_2) = GCD(N, a^{\frac{r}{2}} \pm 1)$

Shor algorithm

1. Fourier Transform: retrieve from periodic signal all frequencies
2. Quantum Fourier Transform (QFT) does the same

Shor's algorithm in practice

Table 5: *Quantum factorization records*

Number	# of factors	# of qubits needed	Algorithm	Year implemented	Implemented without prior knowledge of solution
15	2	8	Shor	2001 [2]	×
	2	8	Shor	2007 [3]	×
	2	8	Shor	2007 [3]	×
	2	8	Shor	2009 [5]	×
	2	8	Shor	2012 [6]	×
21	2	10	Shor	2012 [7]	×
143	2	4	minimization	2012 [1]	✓
56153	2	4	minimization	2012 [1]	✓
291311	2	6	minimization	not yet	✓
175	3	3	minimization	not yet	✓

What about other public-key cryptosystems?

- Diffie-Hellman protocol, ElGamal and DSA: *Peter W. Shor, Polynomial-Time Algorithms for Prime Factorization and* ***Discrete Logarithms*** *on a Quantum Computer*
- Elliptic-Curve Cryptography (ECC), ECDH (Elliptic Curve Diffie-Hellman), ECDSA (Elliptic Curve Digital Signature Algorithm): *Quantum Resource Estimates for Computing* ***Elliptic Curve Discrete Logarithms****, Shor's discrete logarithm quantum algorithm for* ***elliptic curves***

Future of cryptography

- Crypto-agility
- Post-quantum cryptography:
  - lattice-based cryptography
  - hash-based cryptography
  - correction codes-based cryptography
  - multivariate cryptography