Syncrosoft MCFACT™
Secure Data Processing Technology

Re-trust Sixth Quarterly Meeting, March 11, 2008
Villach, Austria

Wulf Harder, Atis Straujums,
{harder,straujums}@syncrosoft.com

Who are we?

Syncrosoft
crypto & security solutions

- European company, established in 1991
  - Offices in Latvia and Germany
  - Software copy-protection
  - Protection of intellectual property
- Wulf Harder
  - Chief scientist and founder
- Atis Straujums
  - Lead of MCFACT development

Syncrosoft
crypto & security solutions

Challenge of Re-trust

"How to ensure that a trusted code base is running on an untrusted machine at all times and that the original code functionality has not been modified prior to or during run-time?"

Syncrosoft
crypto & security solutions

Capabilities of MCFACT

- Execute encrypted code on encrypted data without decrypting either of them
- Ensure code and data integrity
- Execute several independent functionalities concurrently and inseparably

All these features are useful in reaching the goals of the Re-trust project.

Agenda for today

Syncrosoft
crypto & security solutions

- What is MCFACT? (30 minutes)
  - Finite automata
  - Encoders
  - Automata composition
- Applying MCFACT (10 minutes)
  - Example in C++
  - MCFACT design variants
  - Speed and size of protected code
- AES white-box implementation (10 minutes)
  - Resistance against attack by Olivier Billet et al.
  - Techniques usable in Re-trust
- Discussion (5 minutes)

What is MCFACT?

What is MCFACT?

Syncrosoft
crypto & security solutions

MCFACT [em see fakt'] **stands for:**

Multi-Channel Finite Automata Code Transformation

MCFACT is used to:

- Execute encrypted code on encrypted data without decrypting either of them
- Ensure code and data integrity
- Execute several independent functionalities concurrently and inseparably

What is MCFACT?

Syncrosoft
crypto & security solutions

MCFACT is comprised of the following concepts:

- Multi-channel finite automata
  - Operation automata
  - Encoder automata
- Composition
  - Sequential
  - Parallel
- Colours

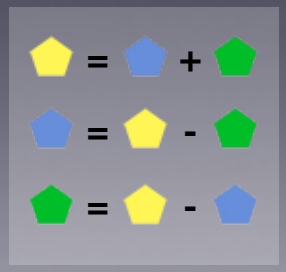
What is MCFACT?

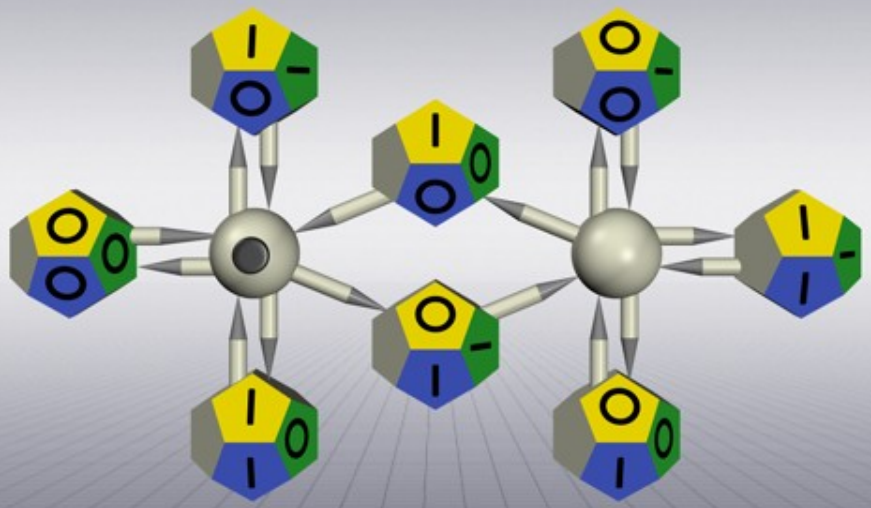
Syncrosoft
crypto & security solutions

Demonstration of execution of a multi-channel finite automaton

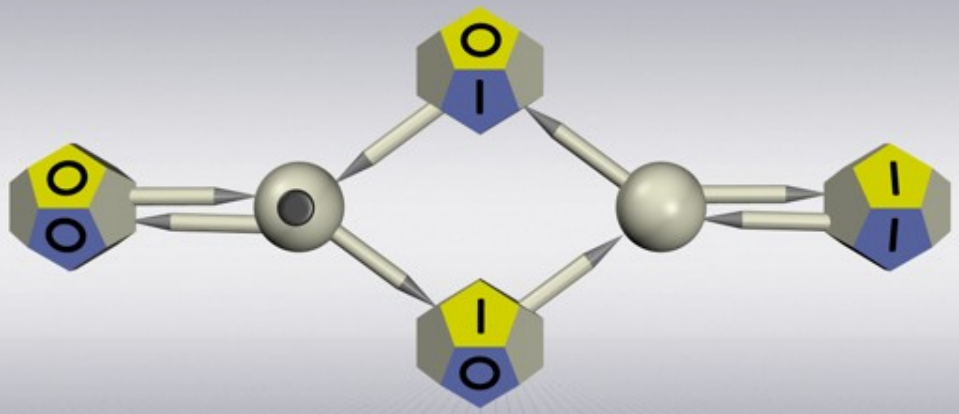
- Two operations are executed in parallel
- Two channels are read from, two are written to

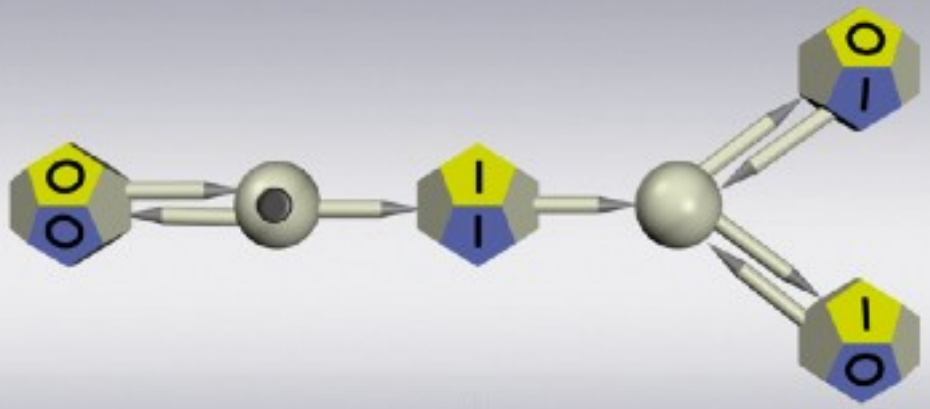
Go to demonstration





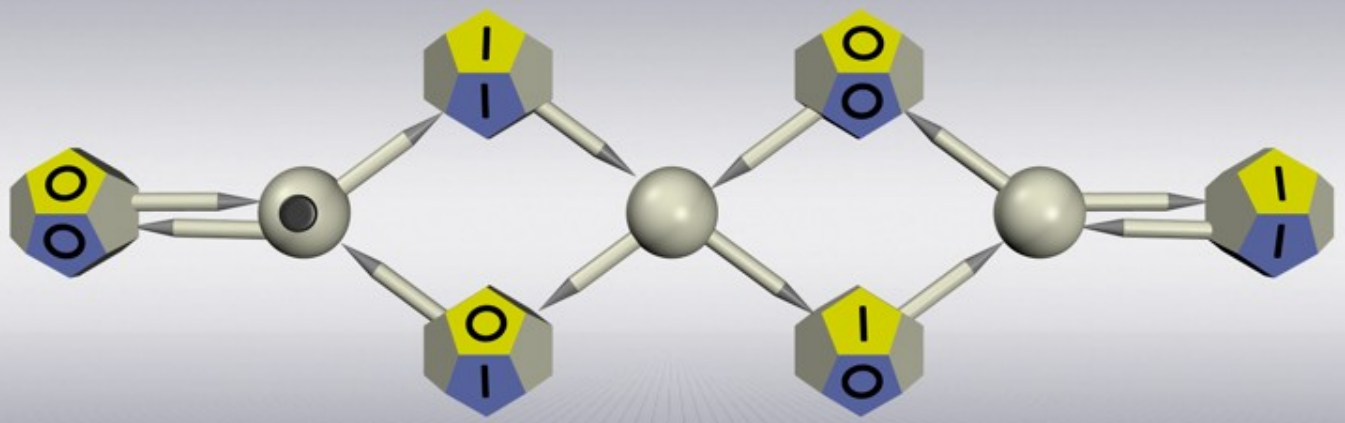
= x 2



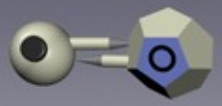


= x 3

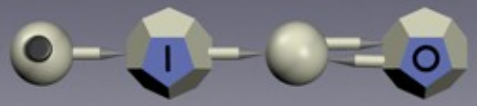
= / 3



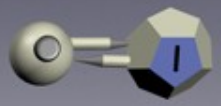
= 0



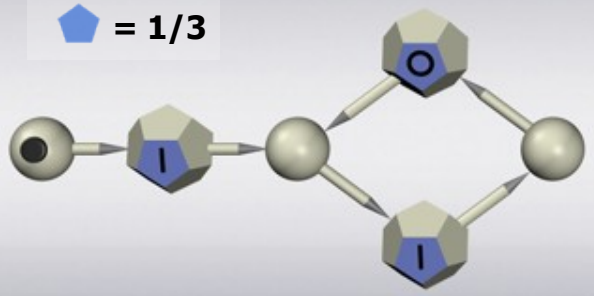
= 1



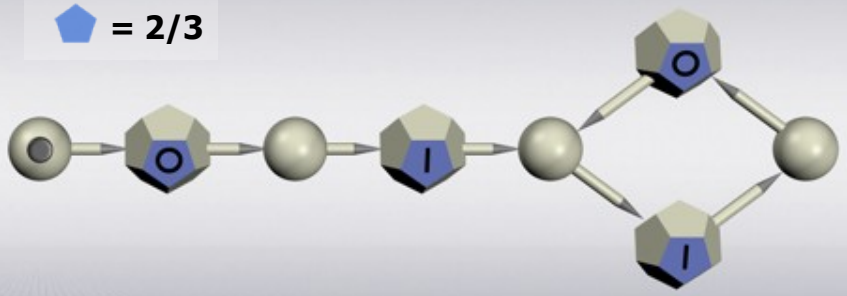
= -1



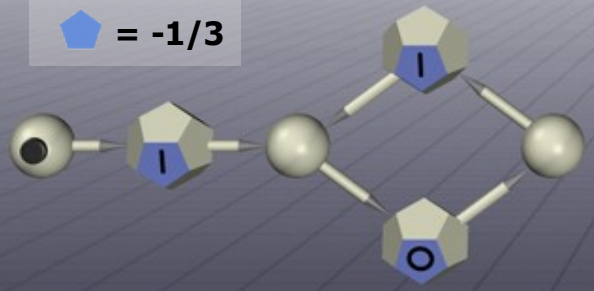
= 1/3



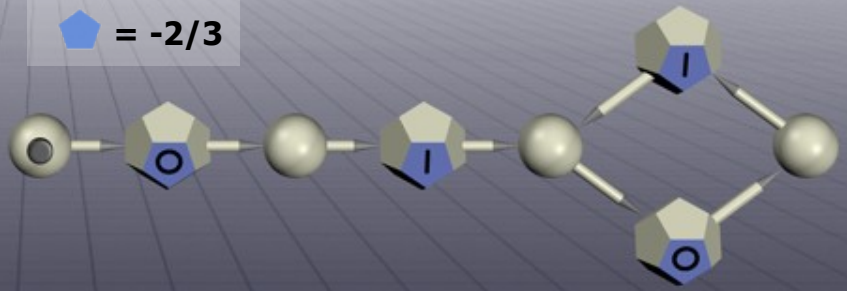
= 2/3

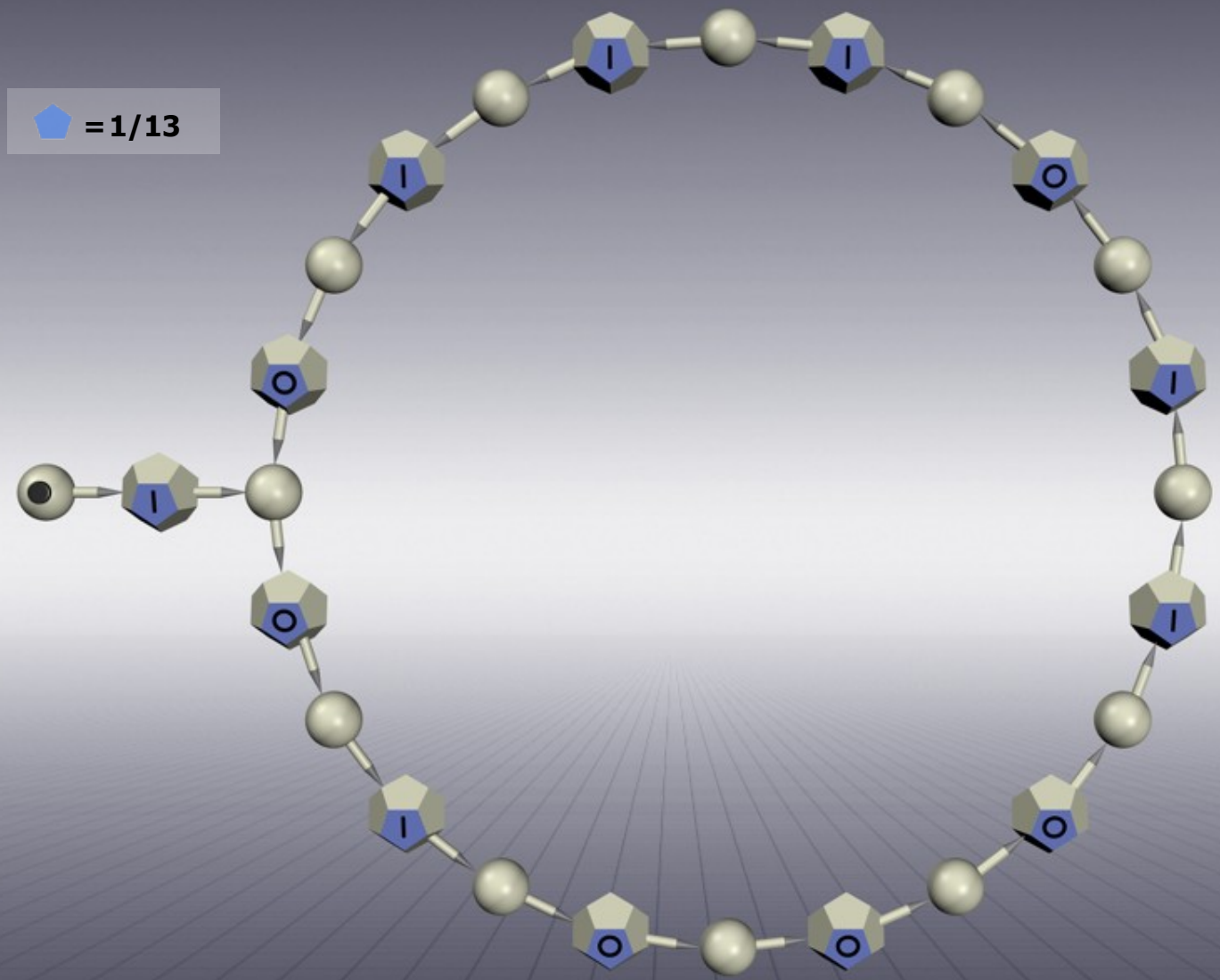


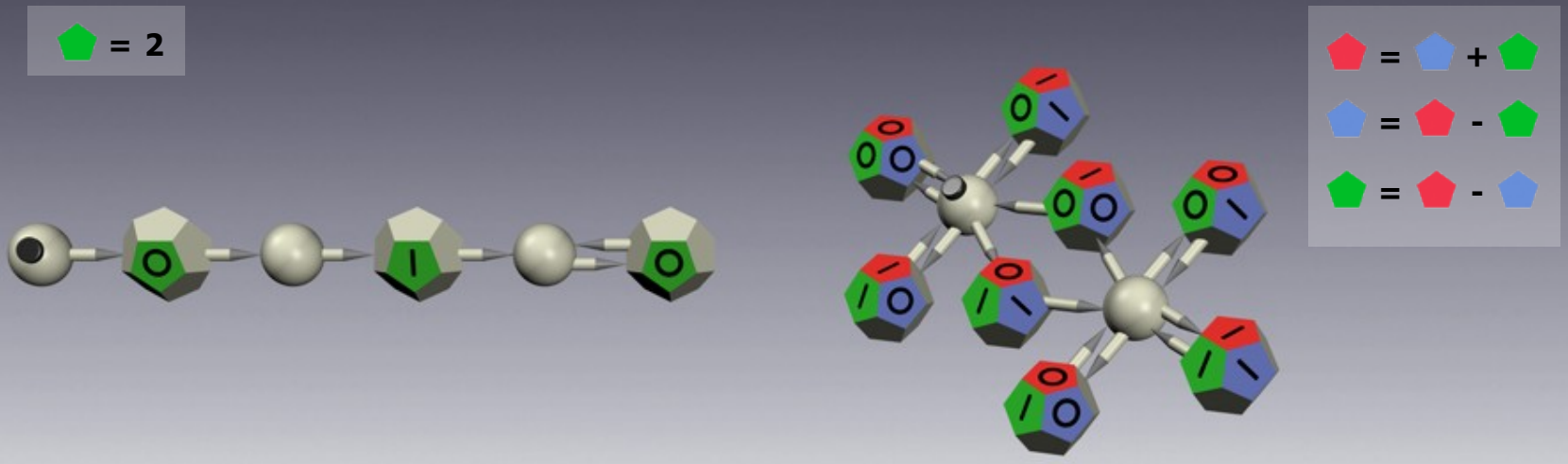
= -1/3



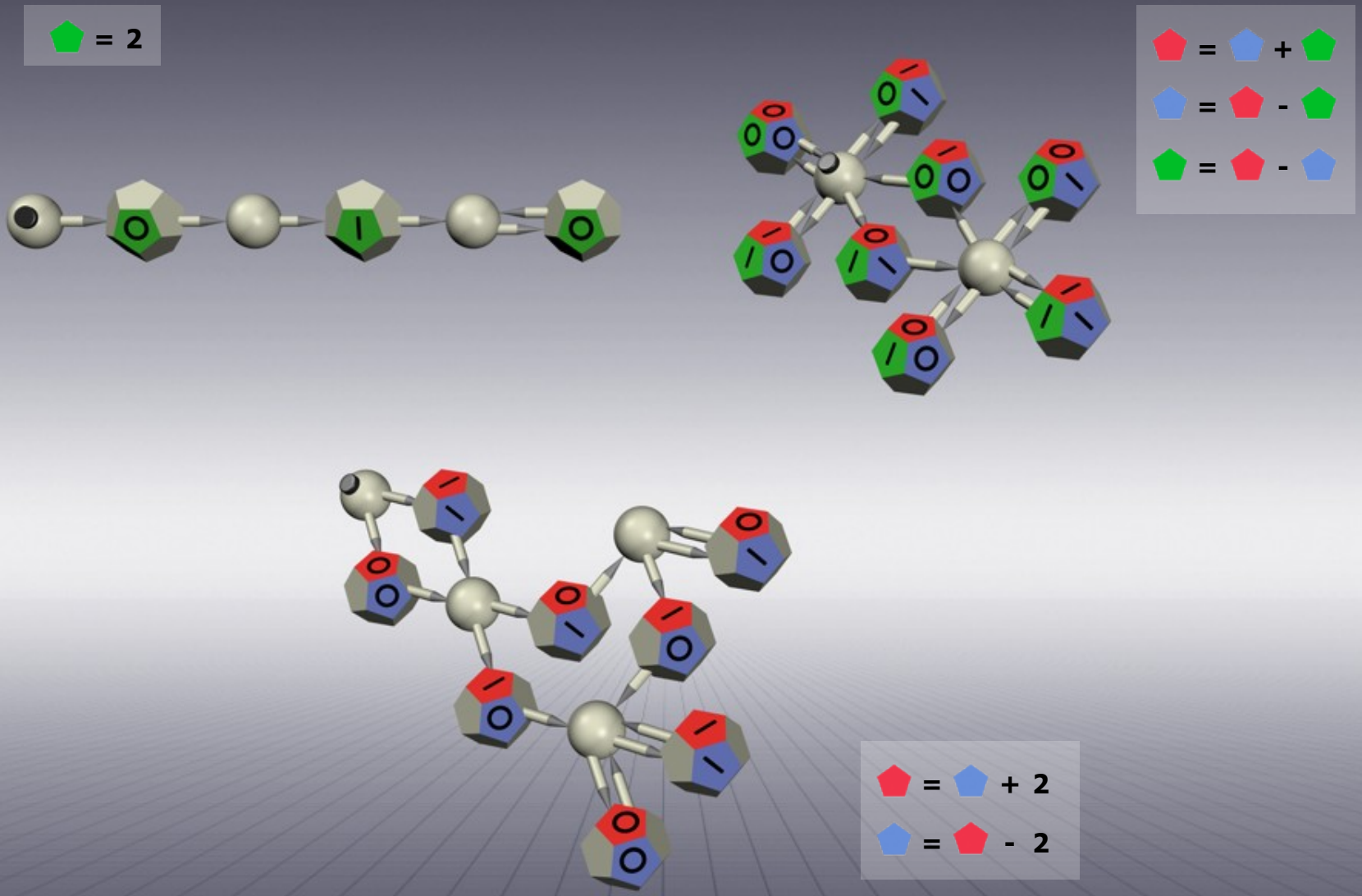
= -2/3



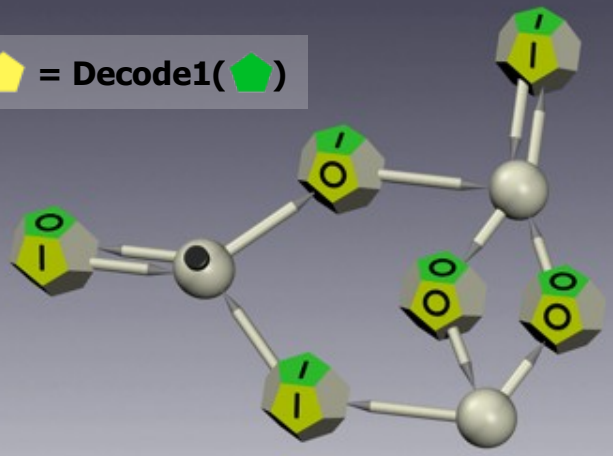

=1/13
I
I
O
I
I
O
I
O
I
O
O
O



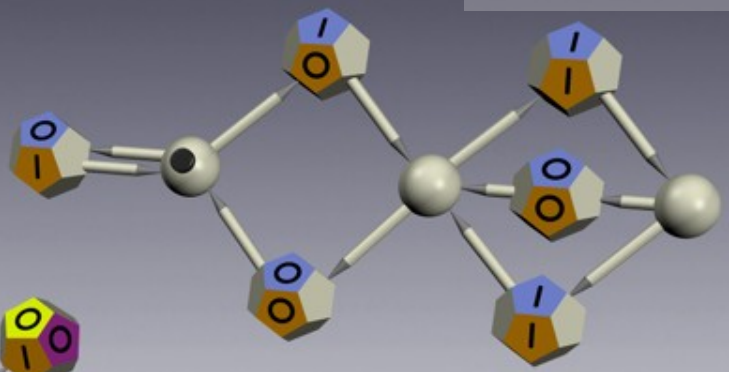
Composition example movie

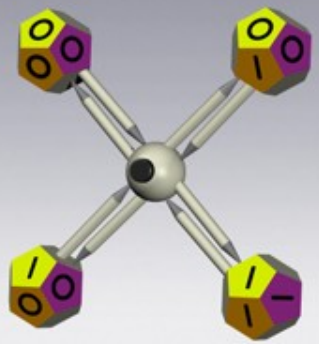

= 2
= +
= -
= -
= + 2
= - 2

= Decode1( )

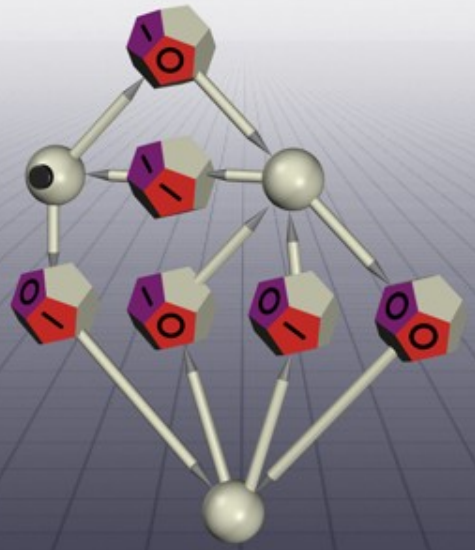


= Decode2( )





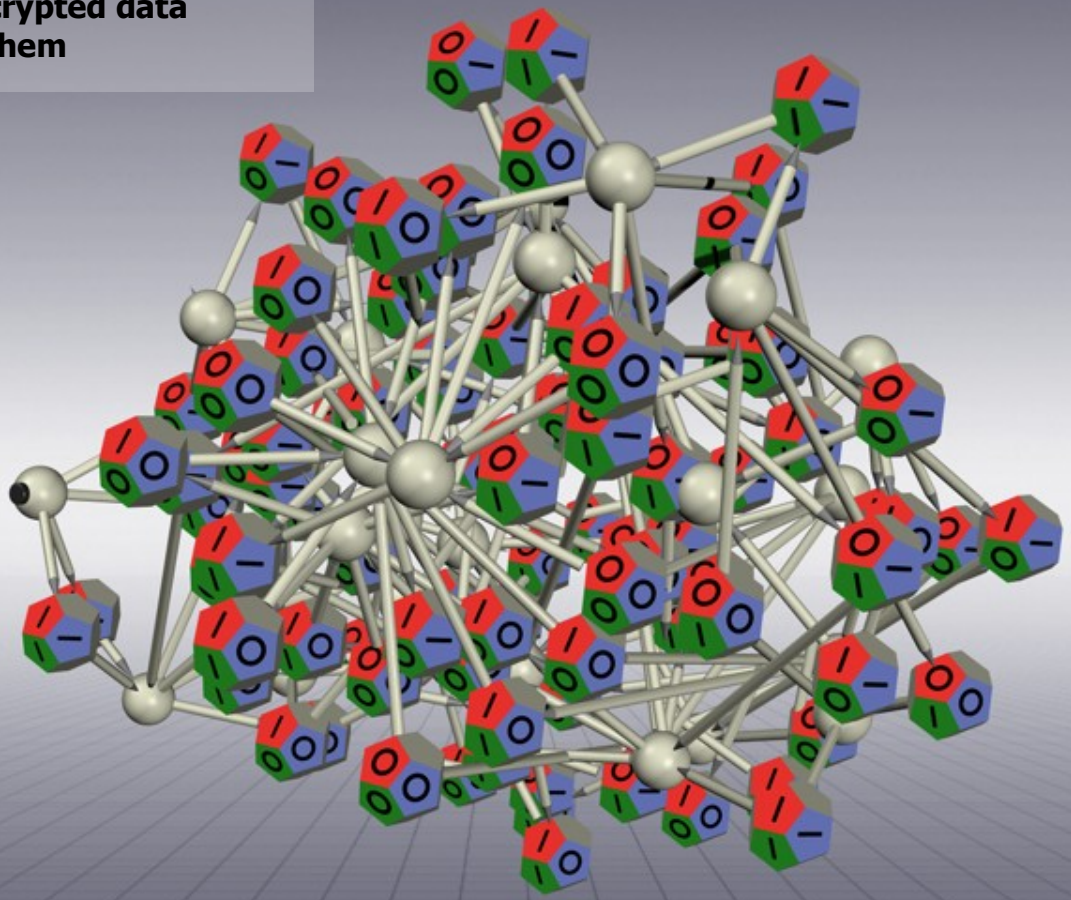
= &



= Encode( )

= Encode(Decode1( ) & Decode2( ))

Composed automaton that performs bitwise AND on encrypted data without revealing them



What is MCFACT?

Known decomposition algorithms

- Have time complexity between $O(N^2)$ and $O(N^3)$
- Decomposition is not unique
- N is the number of states
- In MCFACT, states are represented as a bit vector
- Security is adjusted by choosing the number of state bits

What is MCFACT?

Syncrosoft
crypto & security solutions

"Just as it is hard to factor the product of two large primes, it is also hard to factor the composition of two finite automata."

Bruce Schneier, Applied Cryptography, John Wiley & Sons, 1996.

Comment by Syncrosoft:

- The complexity class of factoring is unknown
- The complexity class of decomposition with regards to the number of states (N) is P
- The complexity class of decomposition with regards to the size of the bit vectors that represent states is unknown

What is MCFACT?

Syncrosoft
crypto & security solutions

MCFACT uses similar principles as FAPKC

- Renji Tao, Shihua Chen, "Finite automata public key cryptosystem and digital signature", Computer Acta, Vol. 8, No. 6, 1985, pp. 401-409. (in Chinese)
- Public key cryptosystem based on finite automata
- Based on the decomposition problem

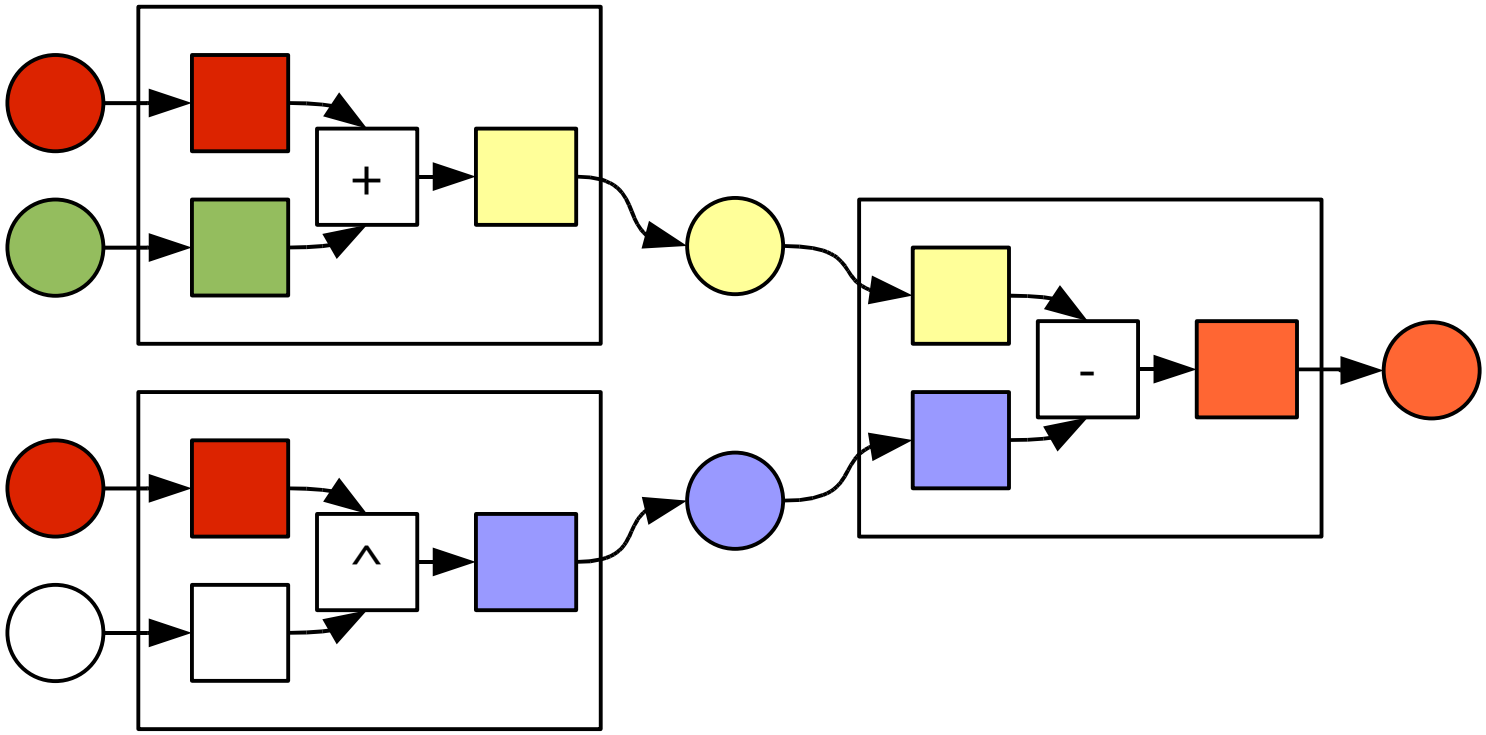
Syncrosoft
crypto & security solutions

FAPKC – broken but without decomposition

- Feng Bao, Yoshihide Igarashi, "Break Finite Automata Public Key Cryptosystem", ICALP 1995, pp. 147-158.
- Demonstration of how to break FAPKC by inverting the public key automaton
- No decomposition is used
- New version of FAPKC was proposed by original authors in 1995

Syncrosoft
crypto & security solutions

Colours – MCFACT way to ensure code integrity



Applying MCFACT

Applying MCFACT

Syncrosoft
crypto & security solutions

Current tools implement MCFACT as a source-code level protection

- Supported languages
  - C/C++
- Integration in development environments
  - Microsoft Visual Studio
- Protected code works on popular platforms/compilers
  - Microsoft Visual C/C++ (6, 7, 8)
  - GCC (3, 4)
  - Microsoft Windows
  - MacOS (PowerPC and Intel)
  - Linux (eLicenser support pending)

Applying MCFACT

Syncrosoft
crypto & security solutions

Example

- Easy to use system
- Protected code and data
- Code integrity
- Virtual Processor

Applying MCFACT

Syncrosoft
crypto & security solutions

```
//MCFACT_PROTECTED
unsigned int findInverse(unsigned int n)
{
    unsigned int test = 1;
    unsigned int result = 0;
    unsigned int mask = 1;
    while (test != 0)
    {
        if (mask & test)
        {
            result |= mask;
            //MCFACT_AUTHORIZED
            test -= n;
        }
        mask <<= 1;
        n <<= 1;
    }
    return result;
}
```

Applying MCFACT

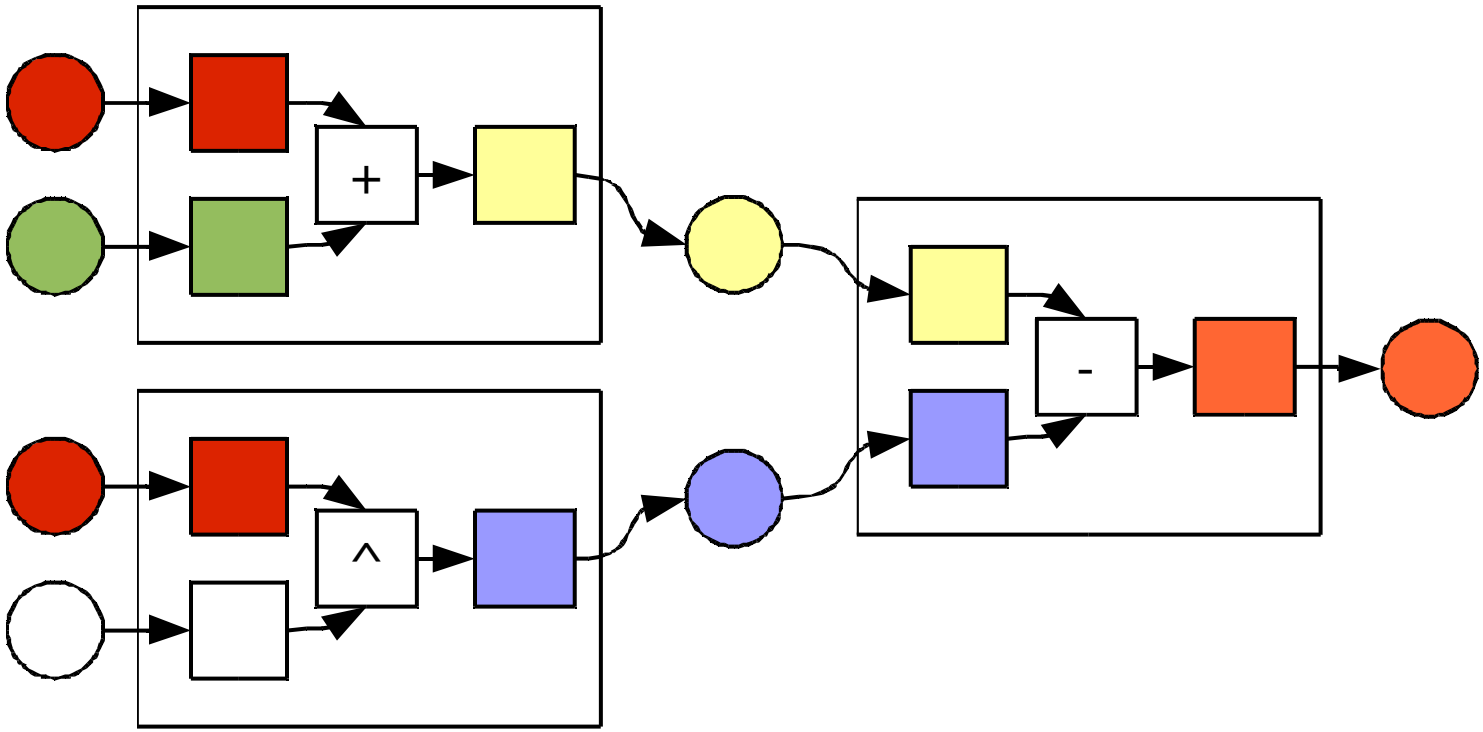
Syncrosoft
crypto & security solutions

```
unsigned int findInverse(unsigned int n)
{
  ...
  class _calculations_cpp_test_0 test=((_init0_0::_init0)());
  class _calculations_cpp_result_0 result=((_init1_0::_init1)());
  class _calculations_cpp_mask_0 mask=((_init2_0::_init2)());
  _cycle1:{
    signed char tmp8;
    ((tmp8)=((IsNotEqual)((test), (_calculations_cpp_c_0_0))));
    if ((tmp8)) {
      {
        unsigned int tmp7;
        ((And)((mask), (test), (tmp7)));
        if ((tmp7)) {
          ((Or)((result), (mask), (result)));
          ((Sub)((test), (n), (test)));
        }
        ((ShiftLeftOne)((mask), (mask)));
        ((n)<<=(1U));
      }
      goto _cycle1;
    }
  }
  {
    unsigned int tmp9;
    ((Copy)((result), (tmp9)));
    return(tmp9);
  }
}
```

```
//MCFACT_PROTECTED
unsigned int findInverse(unsigned int n)
{
    unsigned int test = 1;
    unsigned int result = 0;
    unsigned int mask = 1;
    while (test != 0)
    {
        if (mask & test)
        {
            result |= mask;
            //MCFACT_AUTHORIZED
            test -= n;
        }
        mask <<= 1;
        n <<= 1;
    }
    return result;
}
```

Syncrosoft
crypto & security solutions

Operations work on certain colours



Applying MCFACT

Syncrosoft
crypto & security solutions

```
void Or(
    const _calculations_cpp_result_0& arg0,
    const _calculations_cpp_mask_0& arg1,
    _calculations_cpp_result_0& arg2)
{
  static const short opCodeLarge[40] = {
    61, 14, 10, 3, 45, 22, 10, 5, 39, 34, 32, 53, 53, 54, 54, 53, 27, 1, 34, 32,
    32, 34, 27, 43, 34, 43, 54, 53, 54, 1, 39, 39, 1, 32, 43, 53, 53, 39, 39, 53
  };
  static const short opCodeSmall[40] = {
    16, 112, 96, 48, 16, 32, 0, 112, 32, 64, 48, 16, 80, 80, 16, 112, 48, 112, 32, 48,
    112, 112, 0, 112, 80, 96, 32, 64, 0, 0, 0, 112, 112, 32, 96, 16, 48, 96, 112, 64
  };
  MainFunction(
    arg0.value_skJ35dF4i2,
    arg1.value_skJ35dF4i2,
    arg2.value_skJ35dF4i2,
    opCodeSmall, opCodeLarge,
    0, 40, 40, 40
  );
}
```

Applying MCFACT

Syncrosoft
crypto & security solutions

```
void MainFunction (
    const unsigned char * var1, const unsigned char * var2, unsigned char * resultVar,
    const short * opC1, const short * opC2, int stwb, int varLength1, int varLength2,
    int varLength3)
{
  unsigned int currentState, currentState2, nextState, nextState2, res;
  int varLength = varLength1, i;
  static const int firstSteps[512] = {287810,1859014,1736134,1723842,...,1974800};

  currentState = currentState2 = 0;
  if (varLength2 < varLength) varLength = varLength2;
  if (varLength3 < varLength) varLength = varLength3;
  for (i = 0; i < varLength; i++) {
    nextState = first7table0Func()[((var1[i] & 3) << 2) | ((var2[i] & 3) << 0)
      | (opC1[i]) | ((currentState & 528482304) >> 16)];
    nextState ^= first7table1Func()[(currentState >> 16) & 127];
    nextState ^= first7table2Func()[(currentState >> 8) & 255];
    nextState ^= first7table3Func()[(currentState >> 0) & 255];
    currentState = nextState;
    res = (nextState & 3758096384u);
    nextState2 = second7table0Func()[(res >> 23) | (opC2[i]) | ((currentState2
      & 1966080) >> 8)];
    nextState2 ^= second7table1Func()[(currentState2 >> 12) & 31];
    nextState2 ^= second7table2Func()[(currentState2 >> 6) & 63];
    nextState2 ^= second7table3Func()[(currentState2 >> 0) & 63];
    resultVar[(i)?(i - stwb):0] = (nextState2 >> 21)& 3;
    currentState2 = nextState2;
    ...
  }
}
```

Syncrosoft
crypto & security solutions

Speed and size

- Speed and size depend on the security level
- For a given security, speed can be increased at the expense of size

Applying MCFACT

Syncrosoft
crypto & security solutions

Examples of the functionality that is protected with MCFACT:

- Memory allocation
- Custom GUI controls
- Pointer arithmetic
- File encryption
- Hashing algorithms

Applying MCFACT

Syncrosoft
crypto & security solutions

MCFACT is used by:

- Microsoft
- Yamaha
- Steinberg
- Vienna Symphonic Library
- Korg
- Eleco
- many more...

AES white-box implementation

Syncrosoft
crypto & security solutions

Attack by Olivier Billet et al.

- Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C.,

White-Box Cryptography and an AES Implementation, In Nyberg, K., Heys, H.M., eds.: Selected Areas in Cryptography – SAC 2002. Volume 2595 of Lecture Notes in Computer Science, Springer Verlag (2003) 250–270. http://citeseer.ist.psu.edu/736207.html

- Olivier Billet, Henri Gilbert, Charaf Ech-Chatbi,

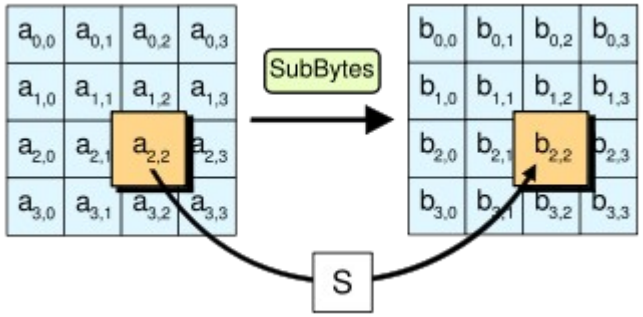
Cryptanalysis of a White Box AES Implementation, H. Handschuh and A. Hasan (Eds.): SAC 2004, LNCS 3357, pp. 227–240, Springer-Verlag Berlin Heidelberg 2005. http://bo.blackowl.org/research/paper/wbaes-cryptanalysis

AES white-box implementation

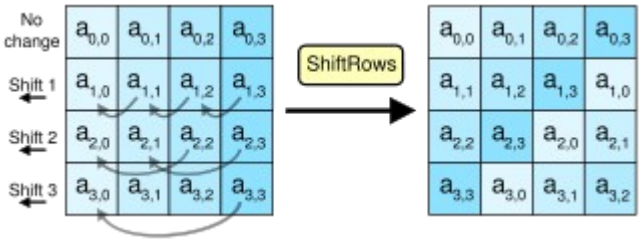
Syncrosoft
crypto & security solutions

AES algorithm

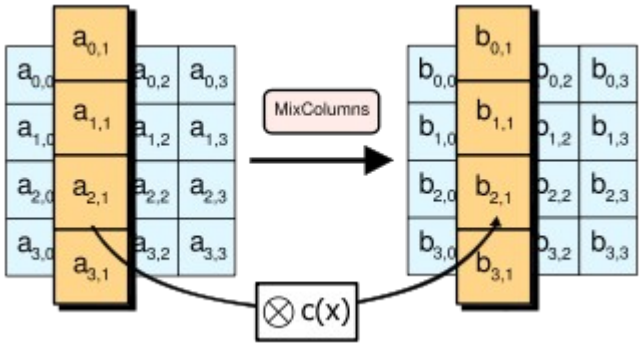
1)



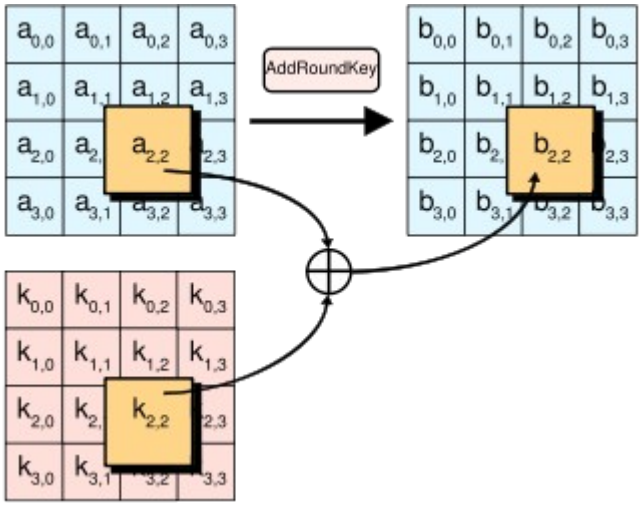
2)



3)



4)



Illustrations: http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Syncrosoft
crypto & security solutions

Attack by Olivier Billet et al.

The main condition for the attack to be successful is:

- It must be possible to compare if a certain byte has the same value for two AES runs that use different inputs.

Syncrosoft
crypto & security solutions

Solution using MCFACT

- Encrypted data throughout the algorithm
- Random component in every variable
- Code integrity based on colours
- Data integrity – hash of input that is checked parallel to useful operations

Syncrosoft
crypto & security solutions

Random component in every variable

- Each MCFACT-protected variable can contain several values (meaningful value, random, hash)
- Operations can change each value independently yet inseparably
- Random component is changed to achieve maximum data diversity (dependency on the history of operations)

AES white-box implementation

Syncrosoft
crypto & security solutions

Code integrity based on colours

- All loops are unrolled
- Each byte has own colour
- Only matching colours produce correct results
- SubBytes and MixColumns are implemented using calculations instead of table look-ups

Syncrosoft
crypto & security solutions

Data integrity due to hashing

- Hash of all inputs of algorithm is calculated
- Hash is stored inseparably together with every input
- Each operation checks if hashes of both arguments match
- If they don't, the calculation produces errors

Summary

Syncrosoft
crypto & security solutions

MCFACT:

- Executes encrypted code on encrypted data without decrypting either of them
- Ensures code and data integrity
- Executes several independent functionalities concurrently and inseparably

- Is used to solve challenges similar to those of the Re-trust project
- Is complementary to other trust-ensuring techniques (e.g. TrustedFlow, obfuscation, etc.)

Questions, please!

Supplemental slides:

- Finite automata or polynomials?
- MCFACT vs. obfuscation
- MCFACT approach to control flow integrity
- Attacks on MCFACT
- Other Syncrosoft projects

Thank you!

Wulf Harder
harder@syncrosoft.com

Atis Straujums
straujums@syncrosoft.com

Automata or polynomials?

Syncrosoft
crypto & security solutions

XOR of numbers 0..10 using polynomials:

$$\begin{aligned}\mathrm{XOR}(x,y) = {} & x+y+2xy+8x^3y+3x^7y+9x^9y+10x^{10}y+6x^2y^2+6x^3y^2+5x^4y^2+x^5y^2+\\ & 3x^6y^2+3x^7y^2+2x^8y^2+6x^9y^2+x^{10}y^2+8xy^3+6x^2y^3+7x^3y^3+5x^4y^3+\\ & 4x^5y^3+3x^6y^3+7x^7y^3+2x^8y^3+9x^9y^3+6x^{10}y^3+5x^2y^4+5x^3y^4+\\ & 6x^4y^4+10x^5y^4+8x^6y^4+8x^7y^4+9x^8y^4+5x^9y^4+10x^{10}y^4+x^2y^5+4x^3y^5+\\ & 10x^4y^5+7x^5y^5+6x^6y^5+4x^7y^5+4x^8y^5+2x^9y^5+x^{10}y^5+3x^2y^6+3x^3y^6+8x^4y^6+\\ & 6x^5y^6+7x^6y^6+7x^7y^6+x^8y^6+3x^9y^6+6x^{10}y^6+3xy^7+3x^2y^7+7x^3y^7+\\ & 8x^4y^7+4x^5y^7+7x^6y^7+7x^7y^7+x^8y^7+9x^9y^7+3x^{10}y^7+2x^2y^8+2x^3y^8+\\ & 9x^4y^8+4x^5y^8+x^6y^8+x^7y^8+8x^8y^8+2x^9y^8+4x^{10}y^8+9xy^9+6x^2y^9+\\ & 9x^3y^9+5x^4y^9+2x^5y^9+3x^6y^9+9x^7y^9+2x^8y^9+10x^9y^9+6x^{10}y^9+\\ & 10xy^{10}+x^2y^{10}+6x^3y^{10}+10x^4y^{10}+x^5y^{10}+6x^6y^{10}+3x^7y^{10}+\\ & 4x^8y^{10}+6x^9y^{10}+7x^{10}y^{10} \pmod{11}\end{aligned}$$

Automata or polynomials?

Syncrosoft
crypto & security solutions

Difficulty of some operations in each domain

		MCFACT	
		Easy	**Hard**
Polynomials	**Easy**	Add, Sub, Neg, bitwise Not	Mul
	Hard	bitwise Xor, And, Or	bitwise rotation

← Back to questions

MCFACT vs obfuscation

encrypt

1. To put into code or cipher.

2. *Computer Science* To alter (a file, for example) using a secret code so as to be unintelligible to unauthorized parties.

obfuscate

1. To make so confused or opaque as to be difficult to perceive or understand: "A great effort was made . . . to obscure or obfuscate the truth" Robert Conquest.

2. To render indistinct or dim; darken: The fog obfuscated the shore.

The American Heritage® Dictionary of the English Language, Fourth Edition.
Copyright© 2004, 2000 by Houghton Mifflin Company.

MCFACT vs obfuscation

Syncrosoft
crypto & security solutions

Security of MCFACT:

- Data are encrypted and never appear plain
- Code is encrypted and never appears plain
- Stream encryption using finite automata
- Decryption, processing and encryption are composed into single finite automaton
- Decomposition is a hard problem for large automata

← Back to questions

Control flow integrity

Syncrosoft
crypto & security solutions

MCFACT relies on repeated verification to ensure control flow:

- Condition is checked and branch is taken
- In each branch condition is checked again
  - Parallel composition is used to prevent tampering
- If repeated condition check fails, data get corrupted

← Back to questions

Attacks on MCFACT?

Syncrosoft
crypto & security solutions

MCFACT has never been broken!

- The last successful attack on the link between MCFACT and eLicenser has been published in June, 2005

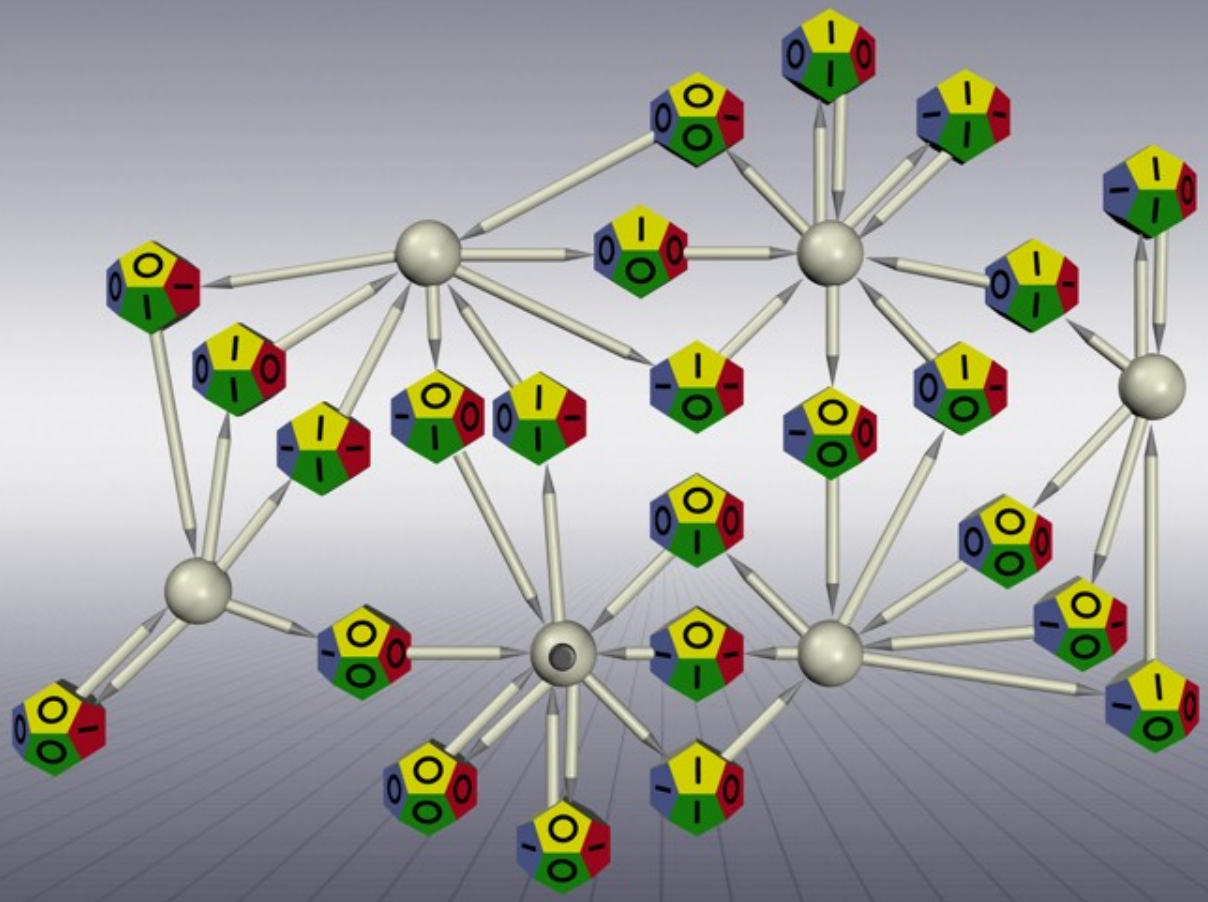
← Back to questions

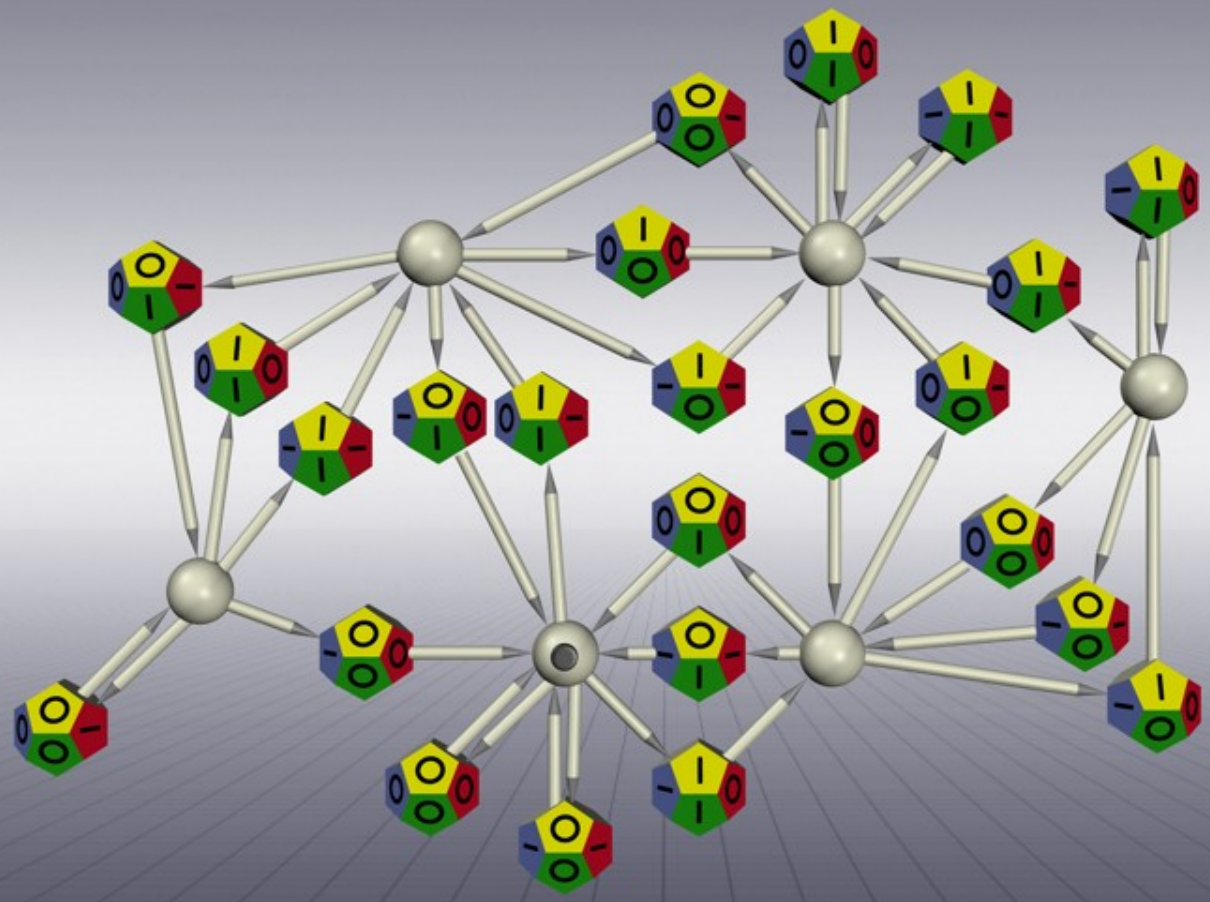
What else is Syncrosoft doing?

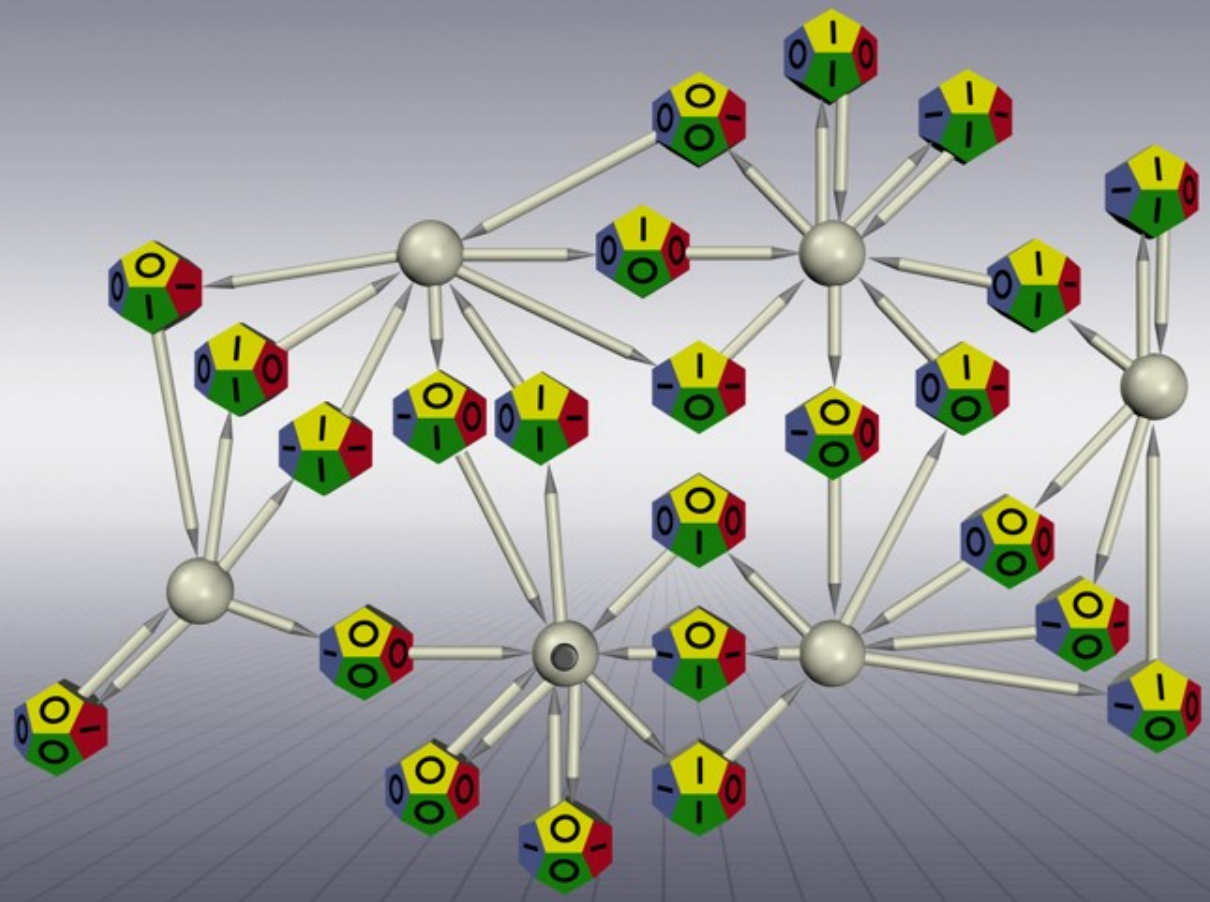
Main projects Syncrosoft is working on:

- Cryptography toolbox (ECC, AES, DES, TDES, SHA-1, SHA-256, RSA coming soon)
- Alternative protected domains
- 1:N symmetric encryption
- Analyser – cryptanalysis tool
- Software copy protection and licensing system, using dedicated hardware – eLicenser
- Digital content protection

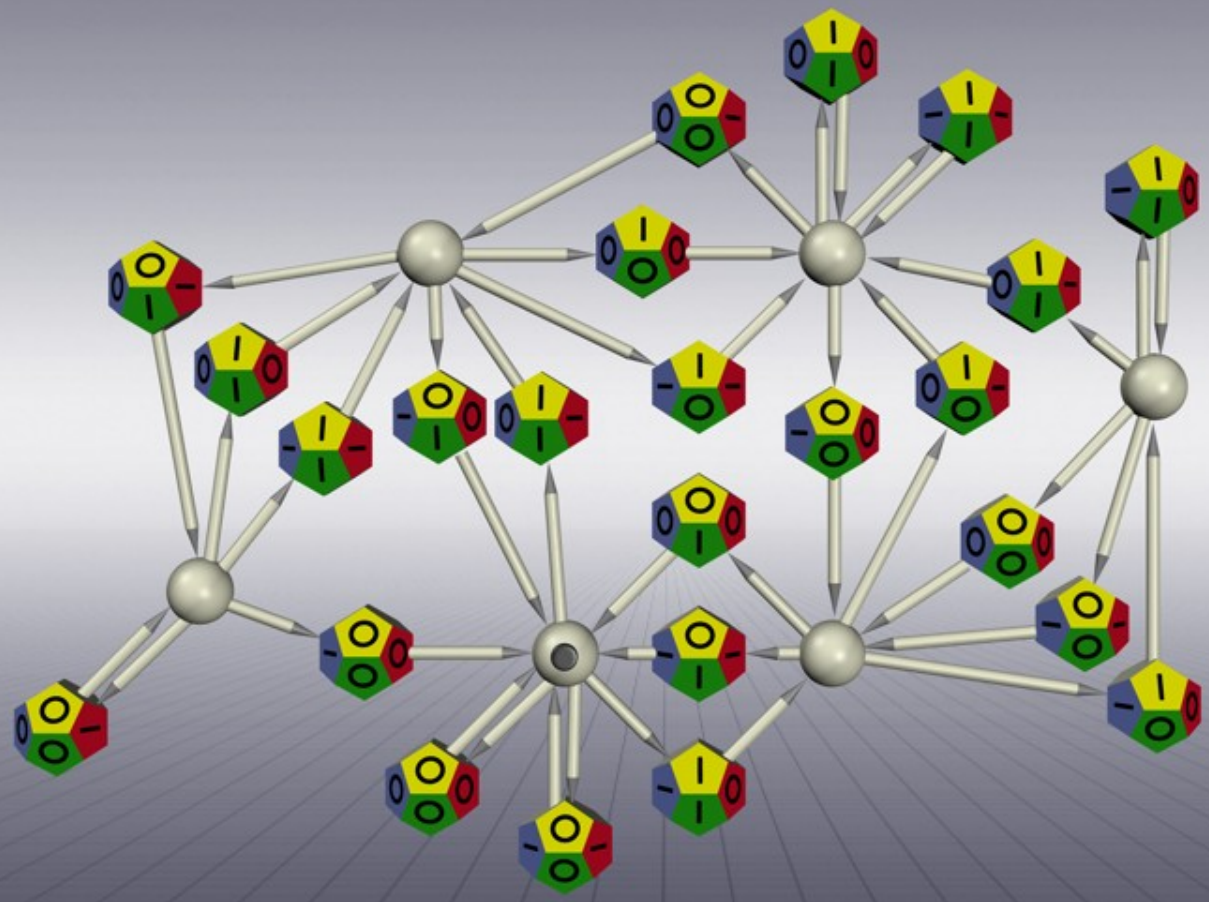
← Back to questions

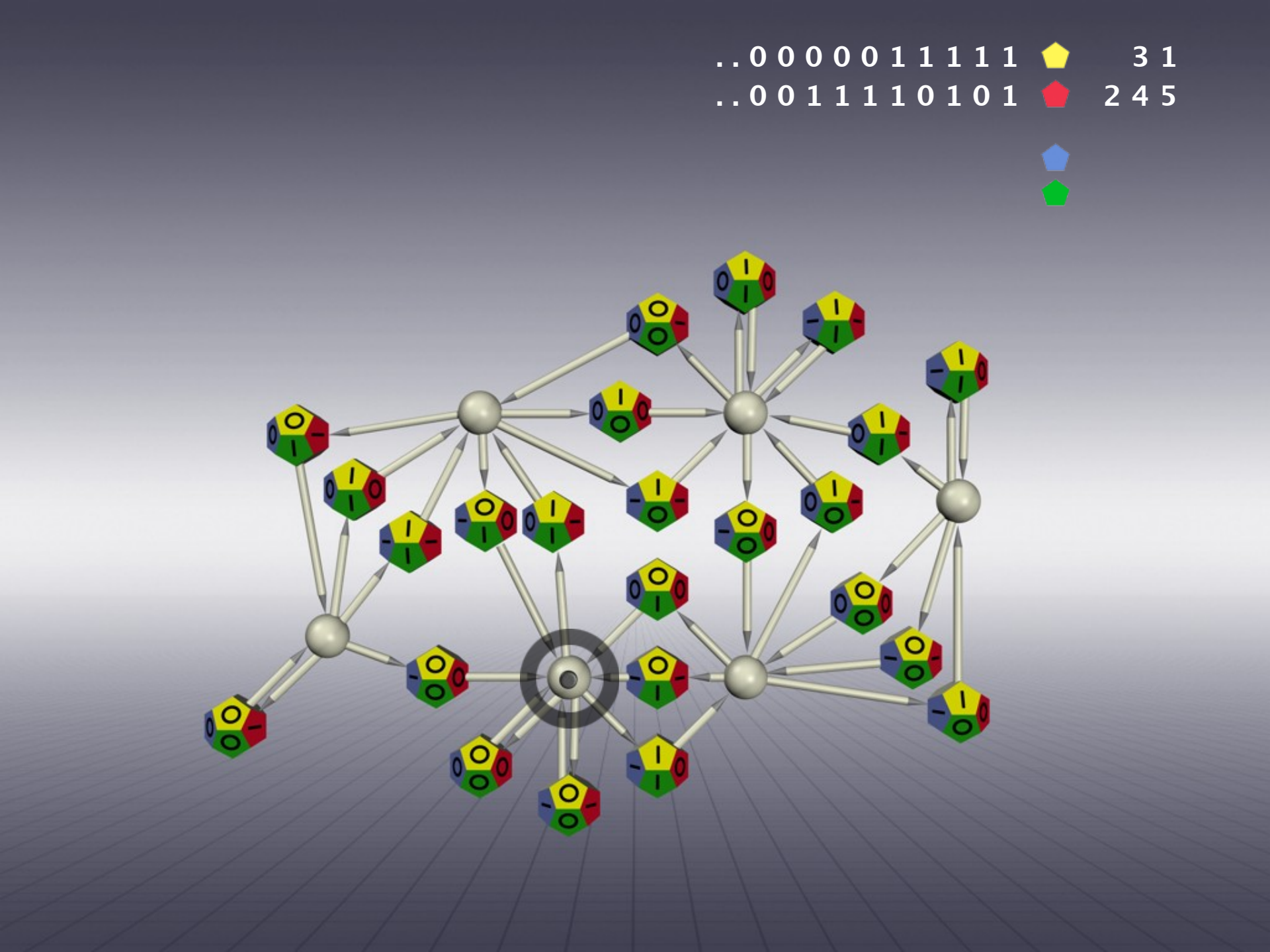


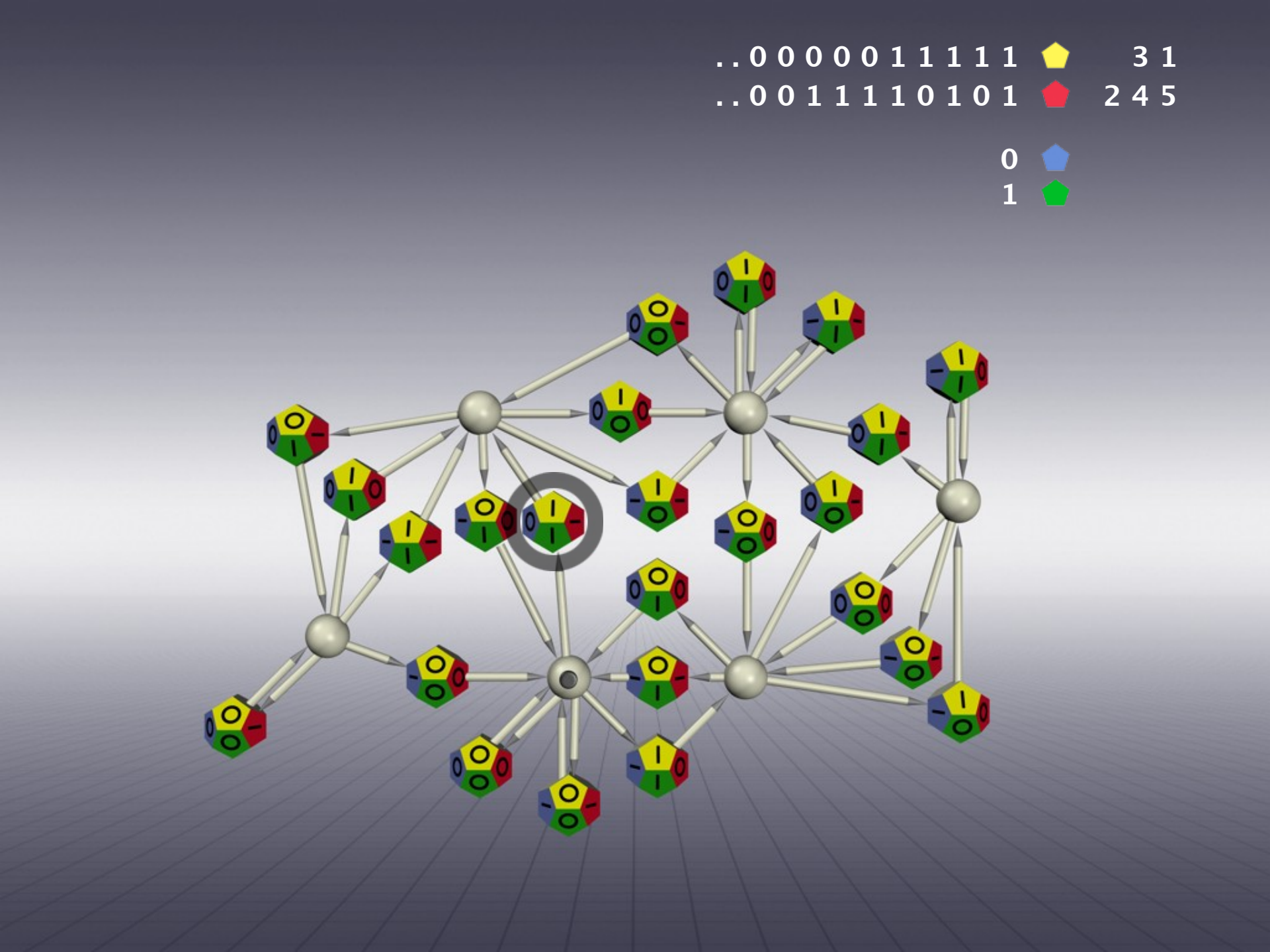


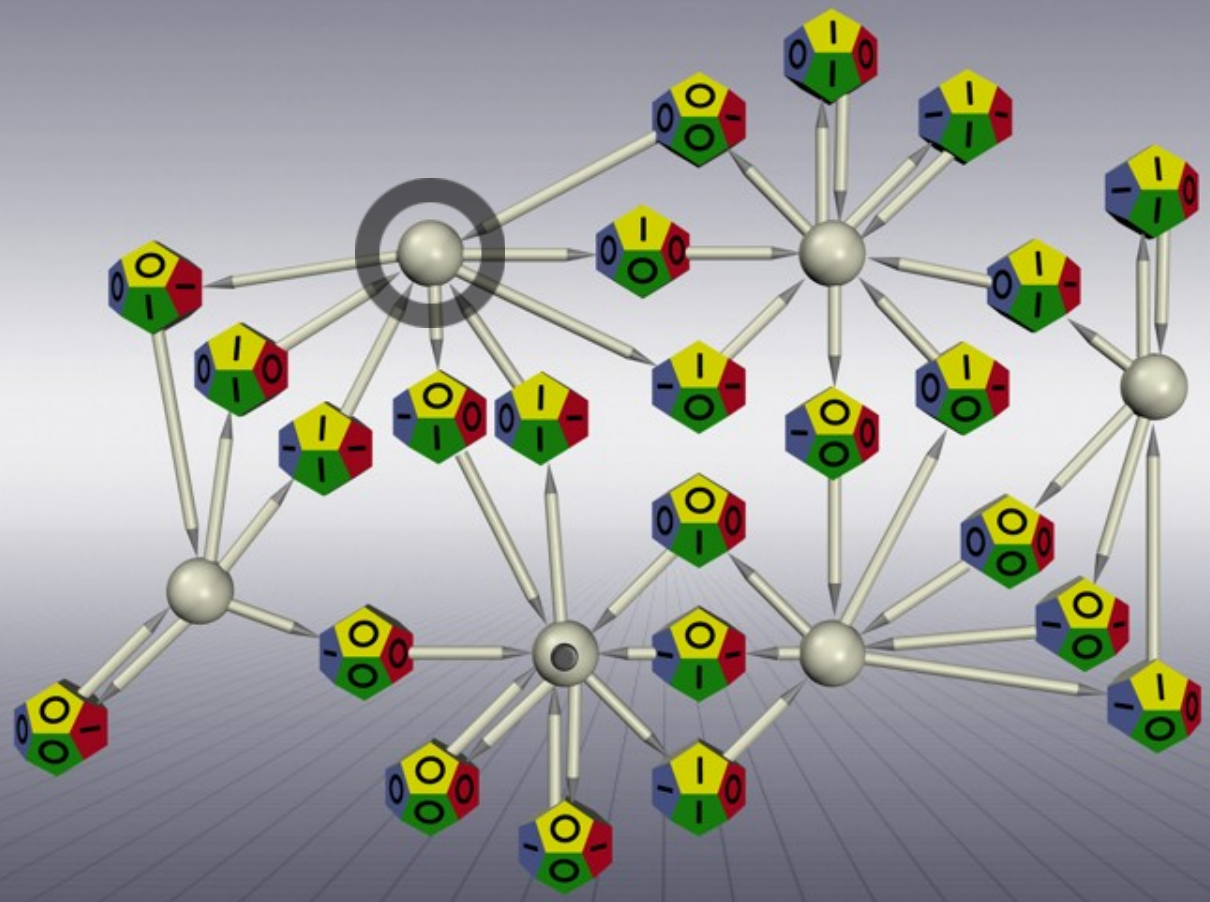

3 1
2 4 5

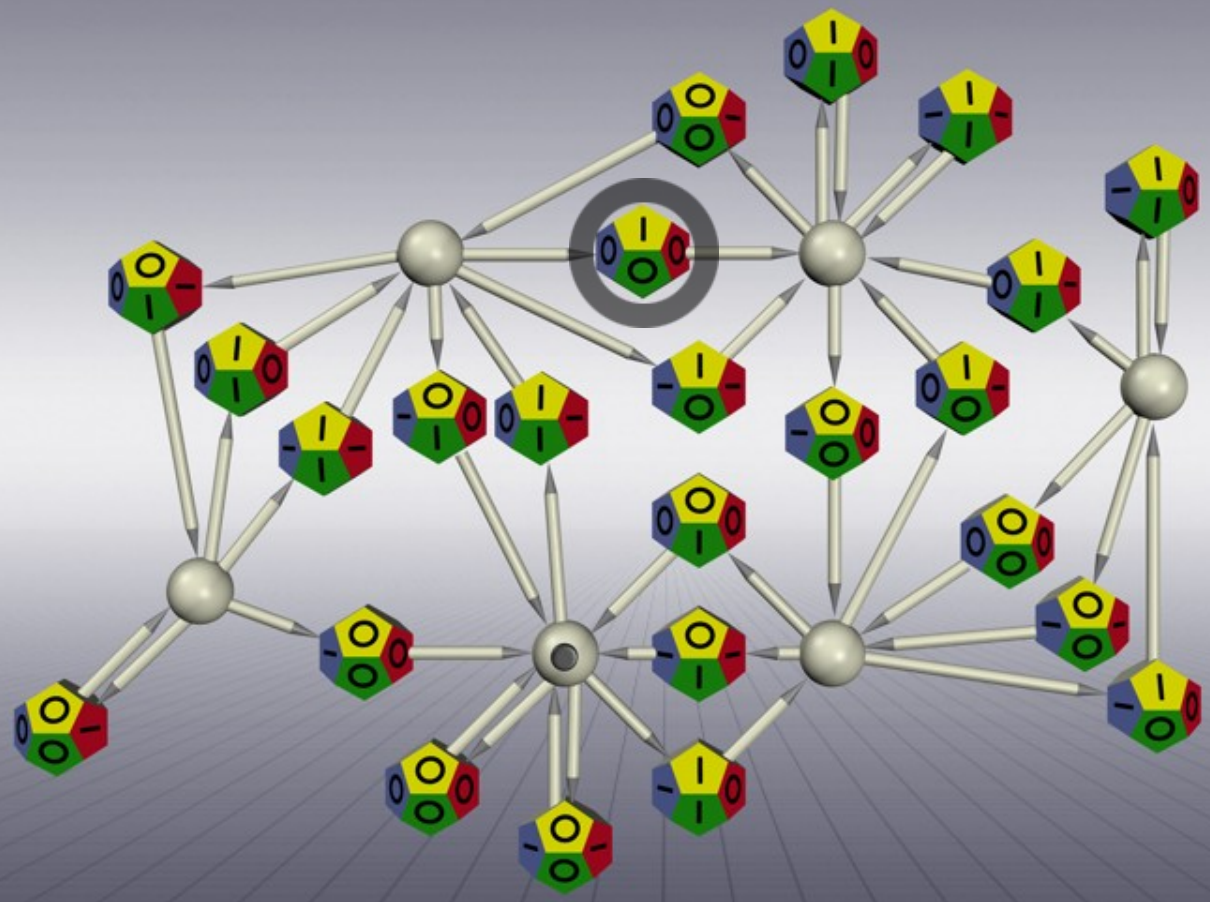
..0000011111 31
..0011110101 245

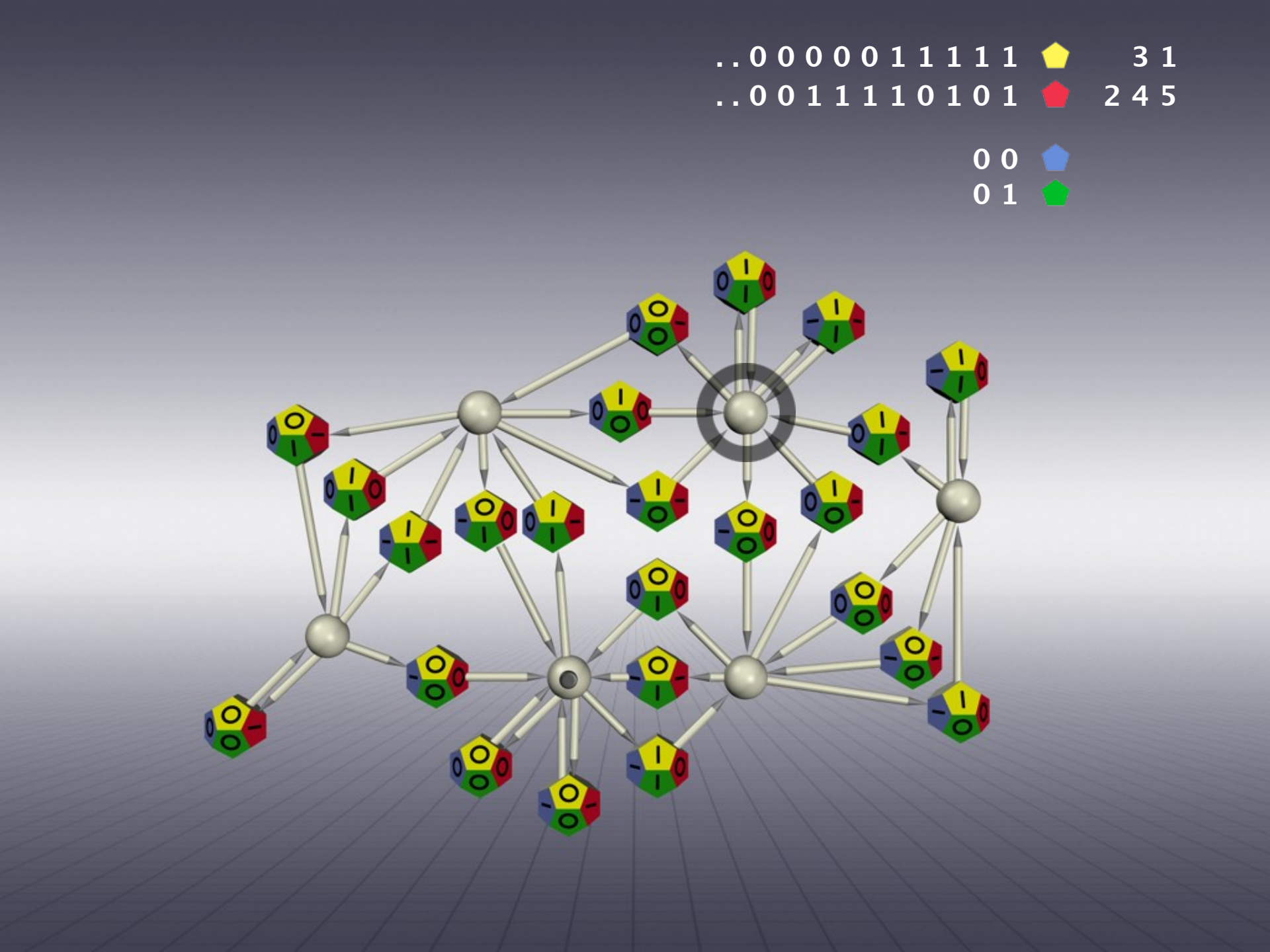


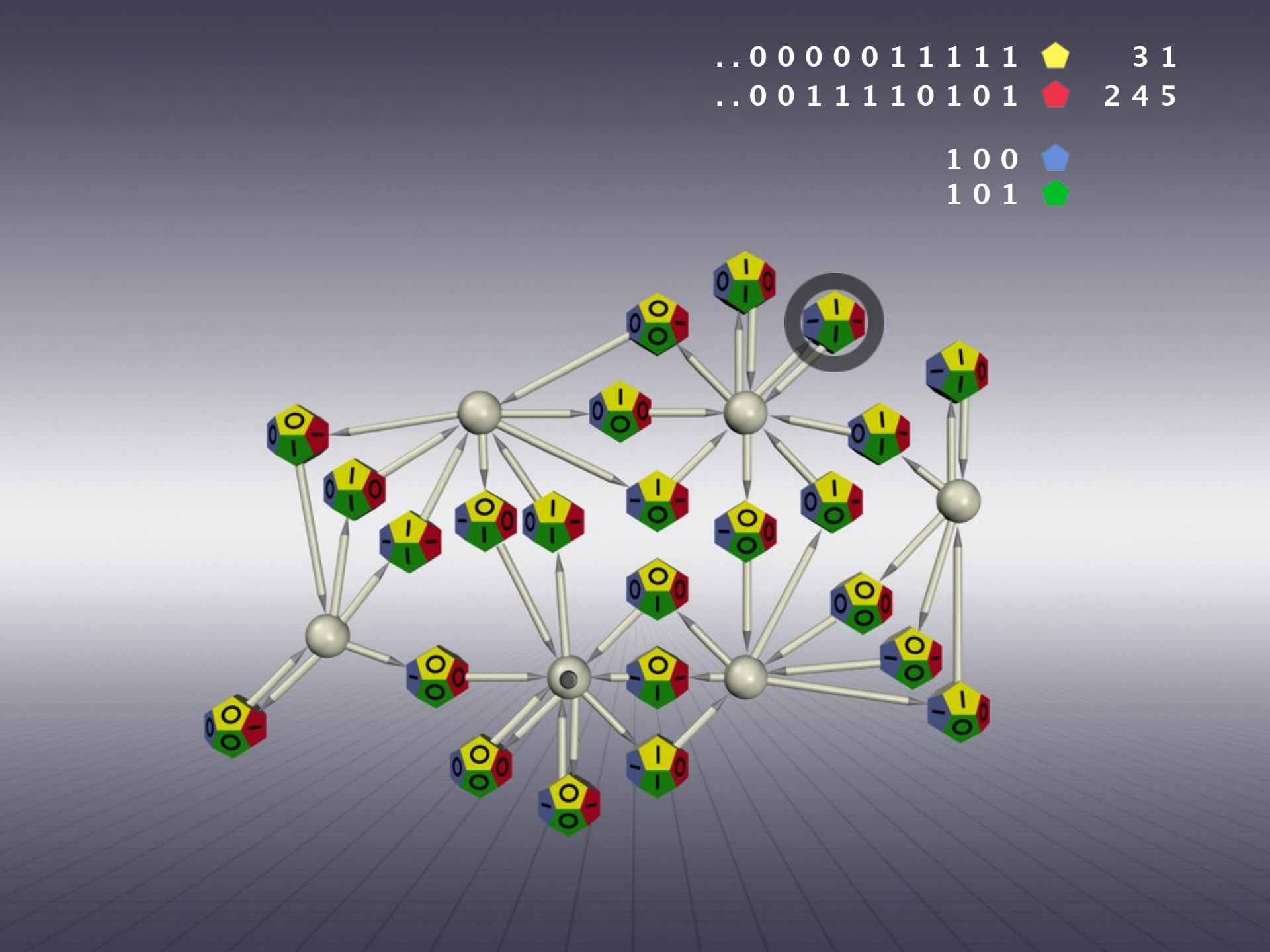

..0000011111 31
..0011110101 245

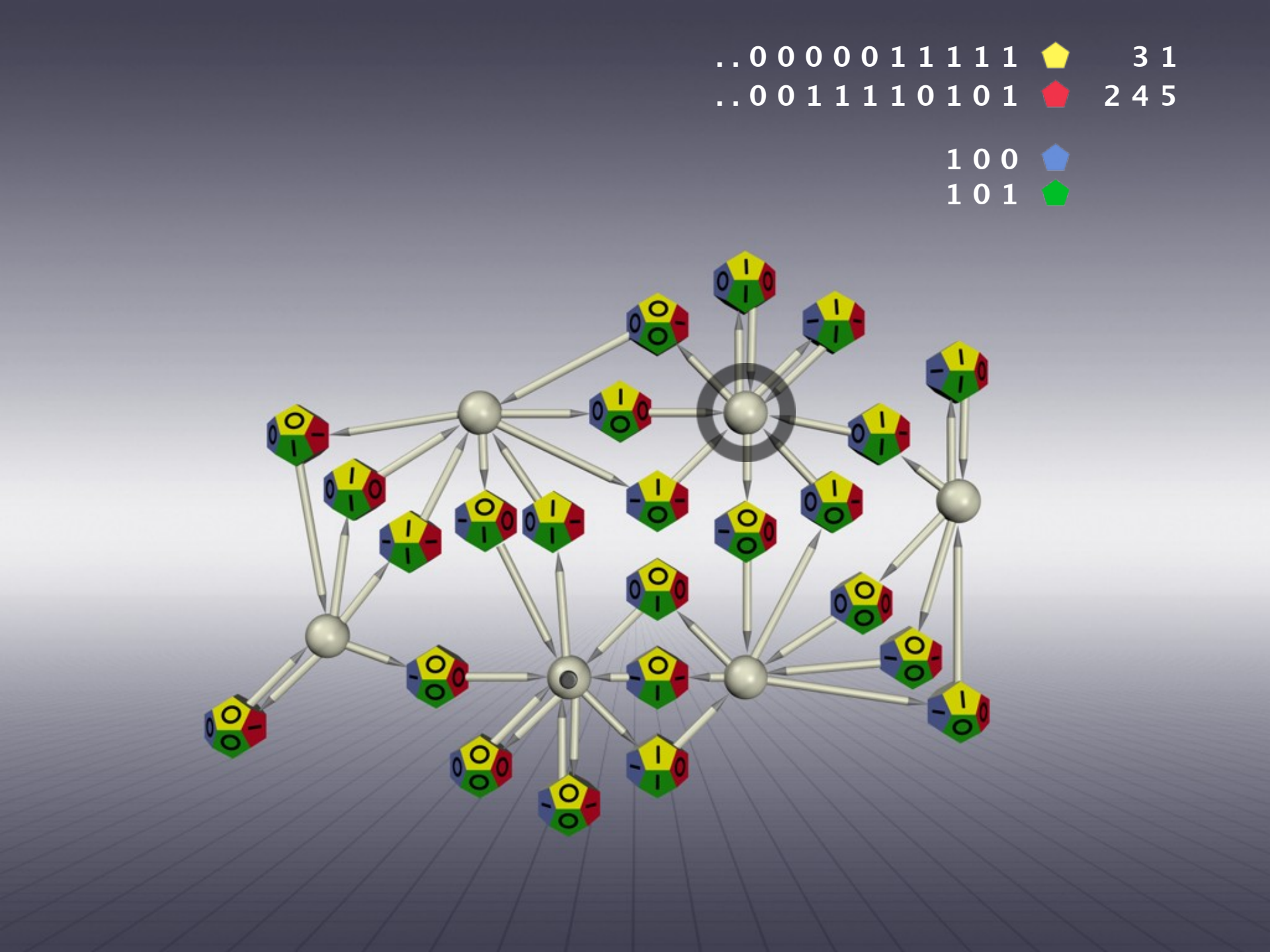

..0000011111 31
..0011110101 245
0
1

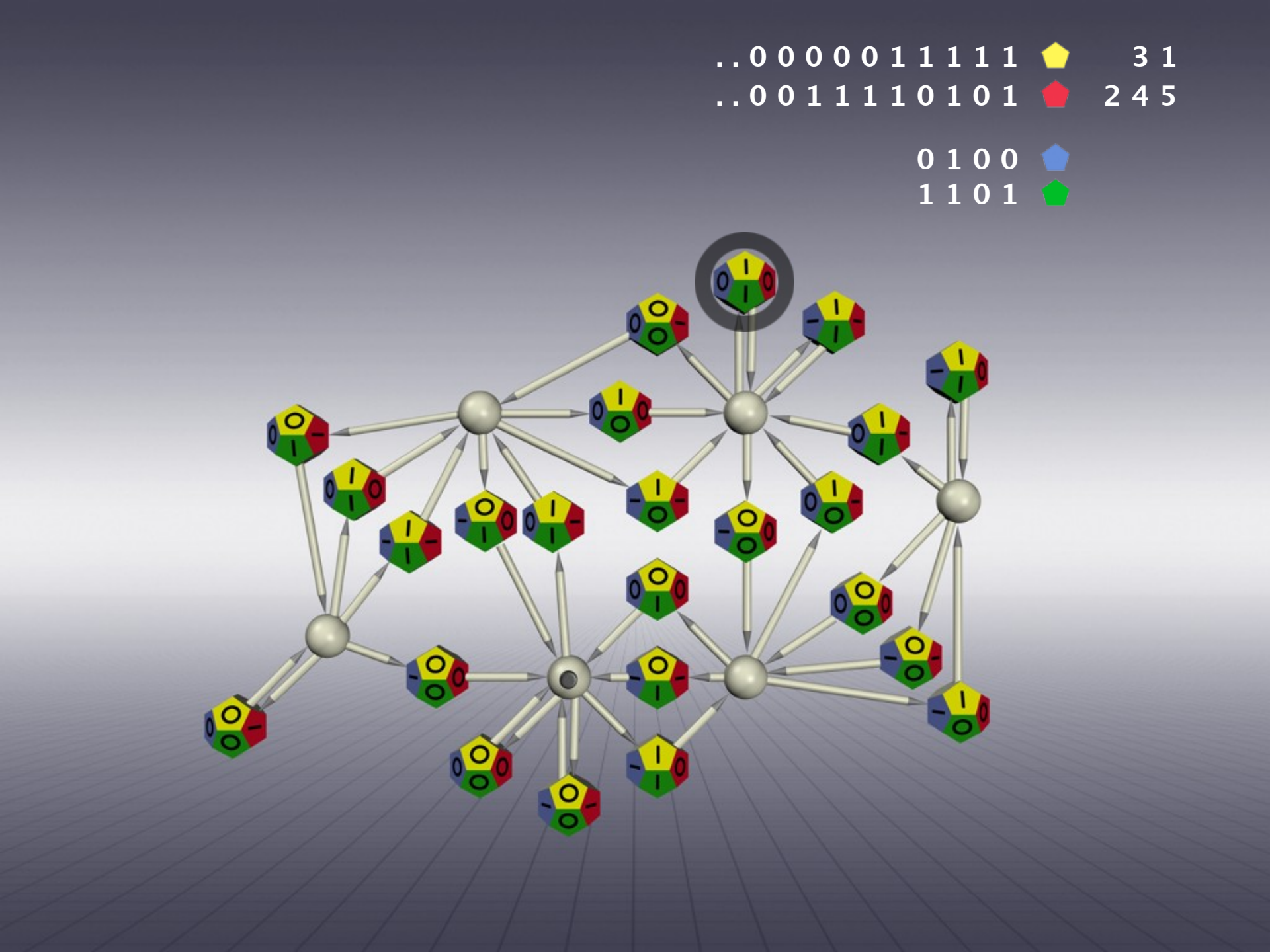

..0000011111 31
..0011110101 245
0
1

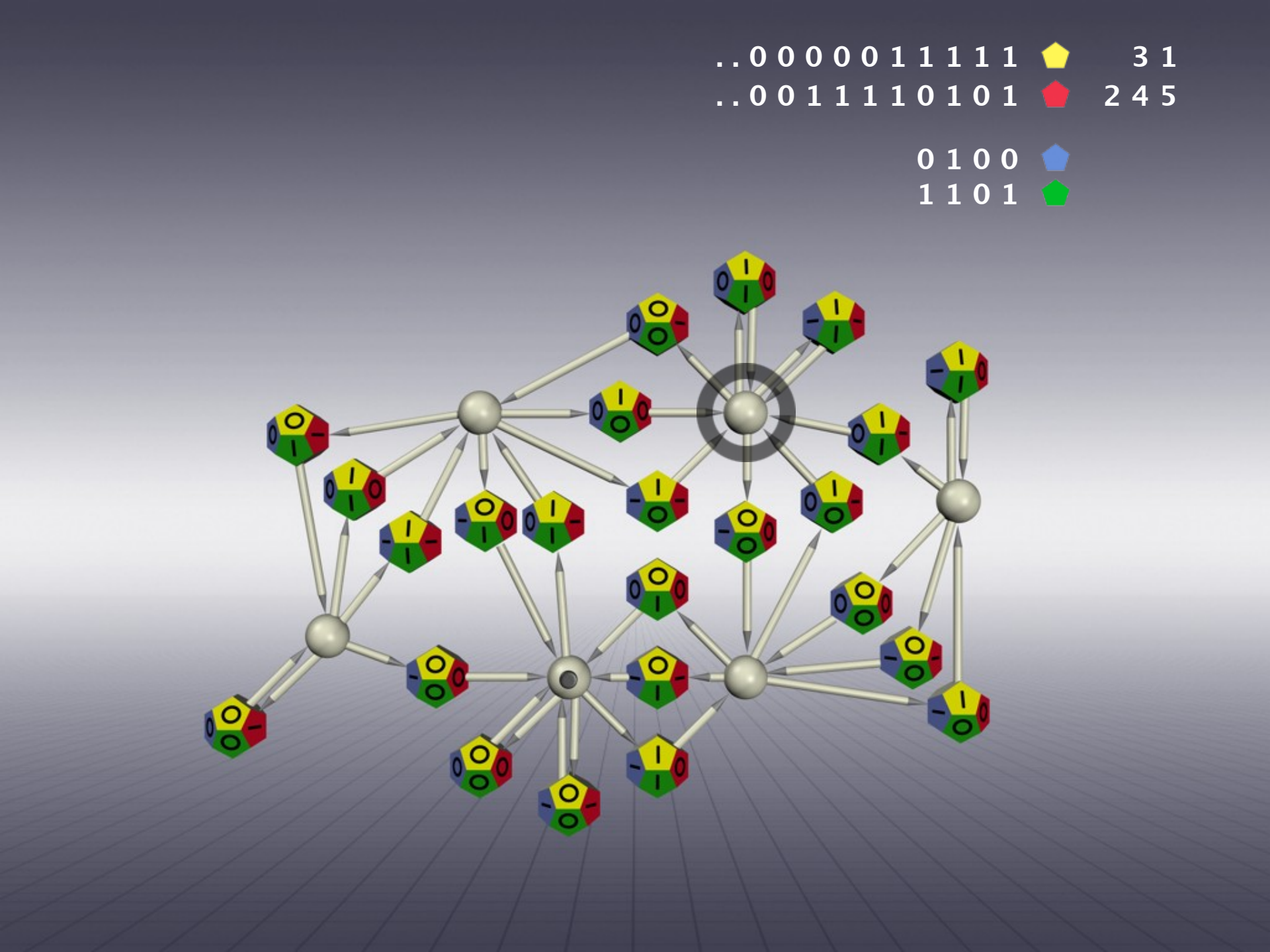

..0000011111 31
..0011110101 245
00
01

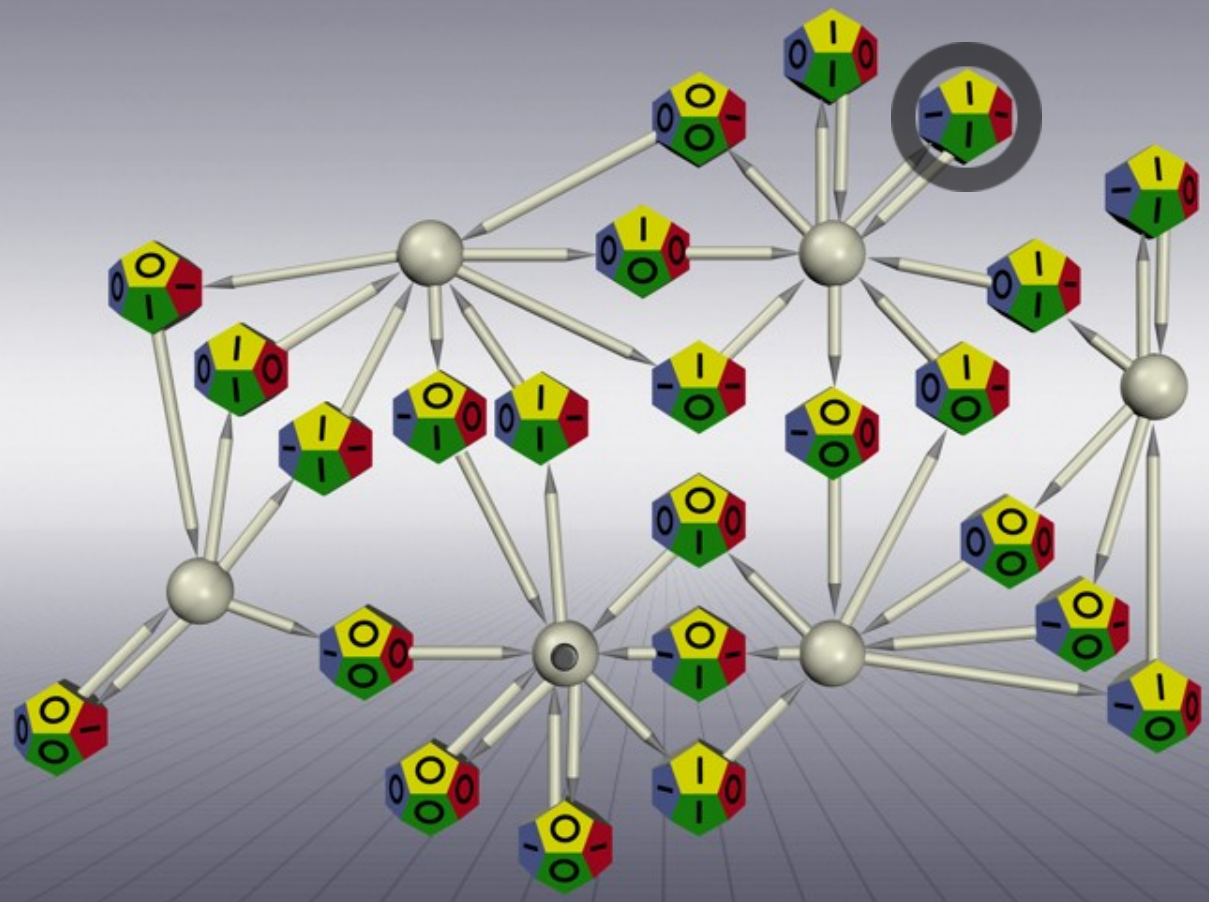

..0000011111 31
..0011110101 245
00
01


..0000011111 31
..0011110101 245
100
101


..0000011111 31
..0011110101 245
100
101

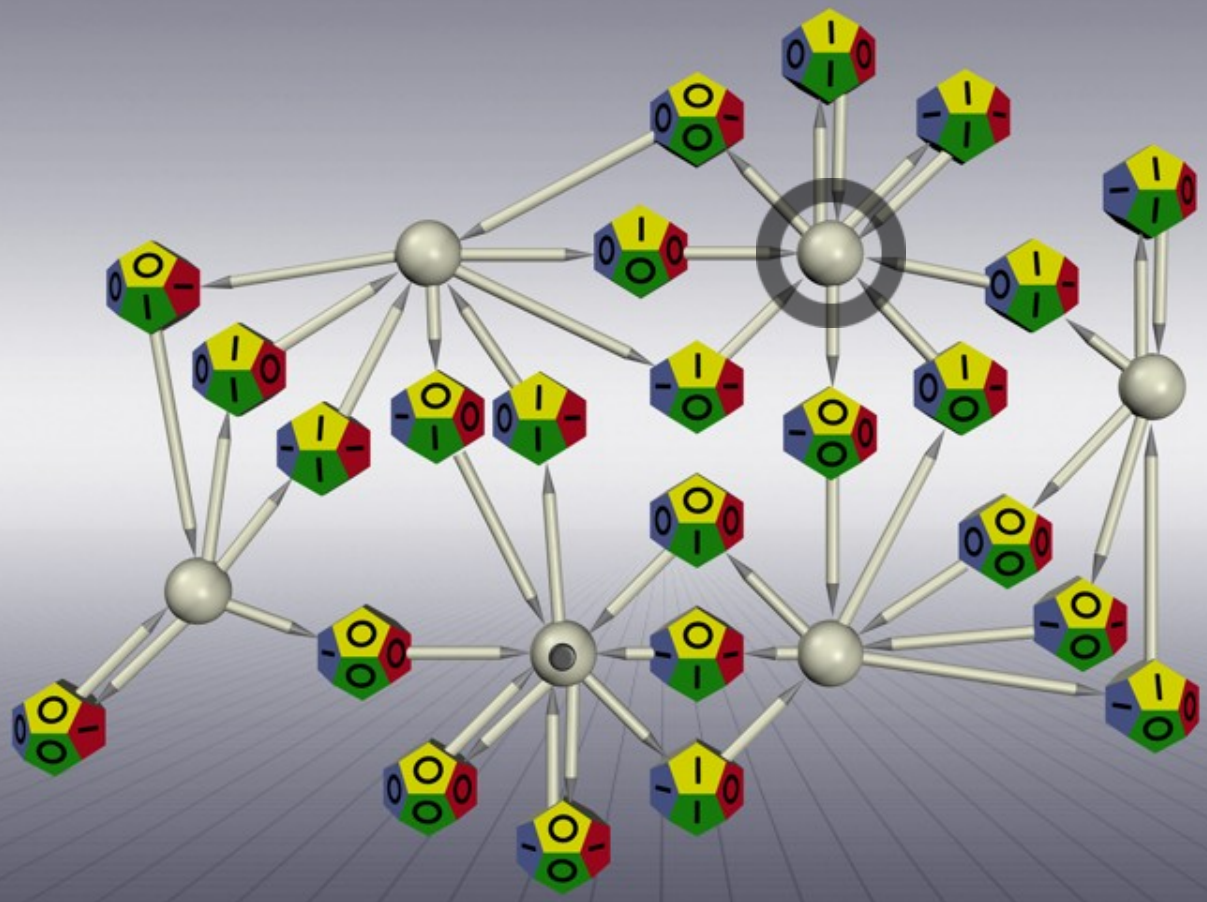

..0000011111 31
..0011110101 245
0100
1101


..0000011111 31
..0011110101 245
0100
1101


..0000011111 31
..0011110101 245
10100
11101

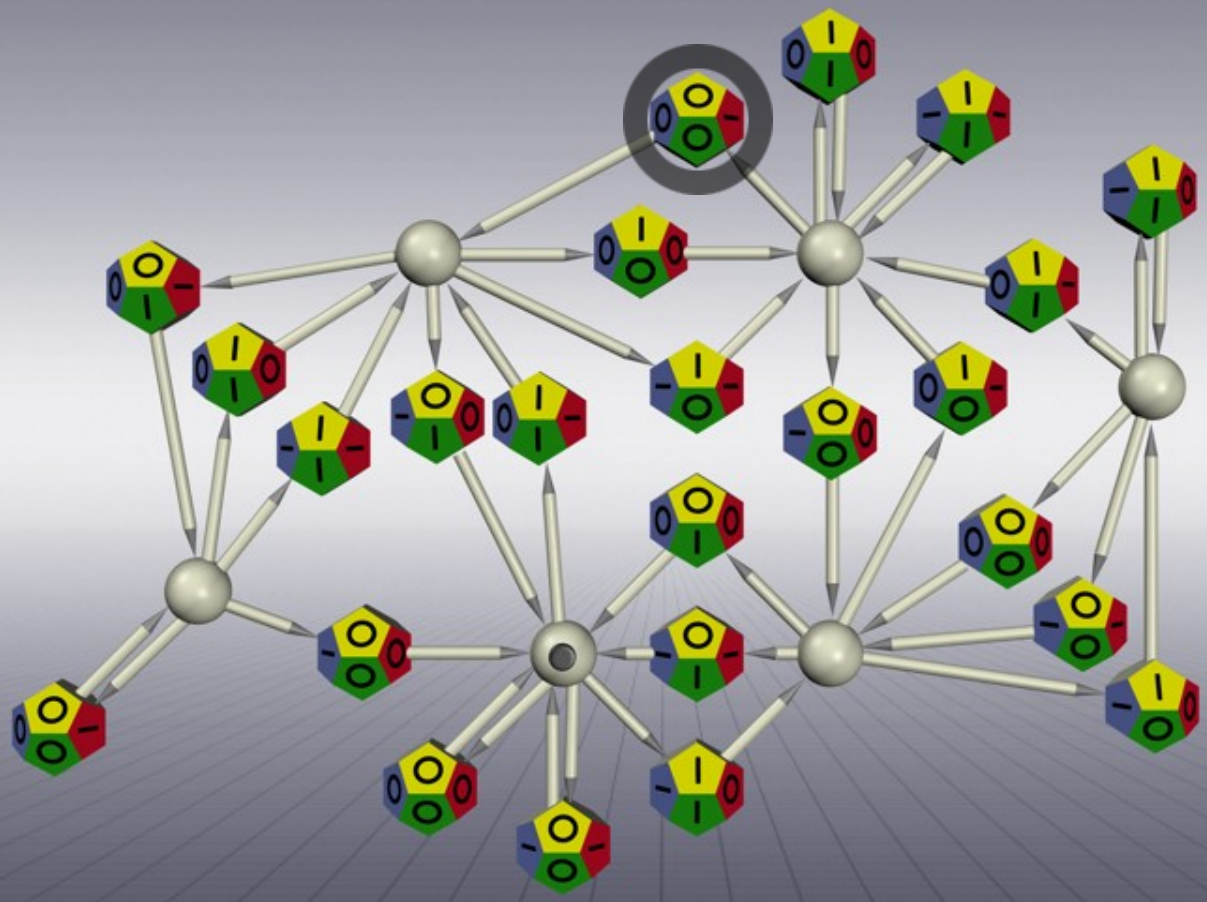
..0000011111 31
..0011110101 245

10100
11101



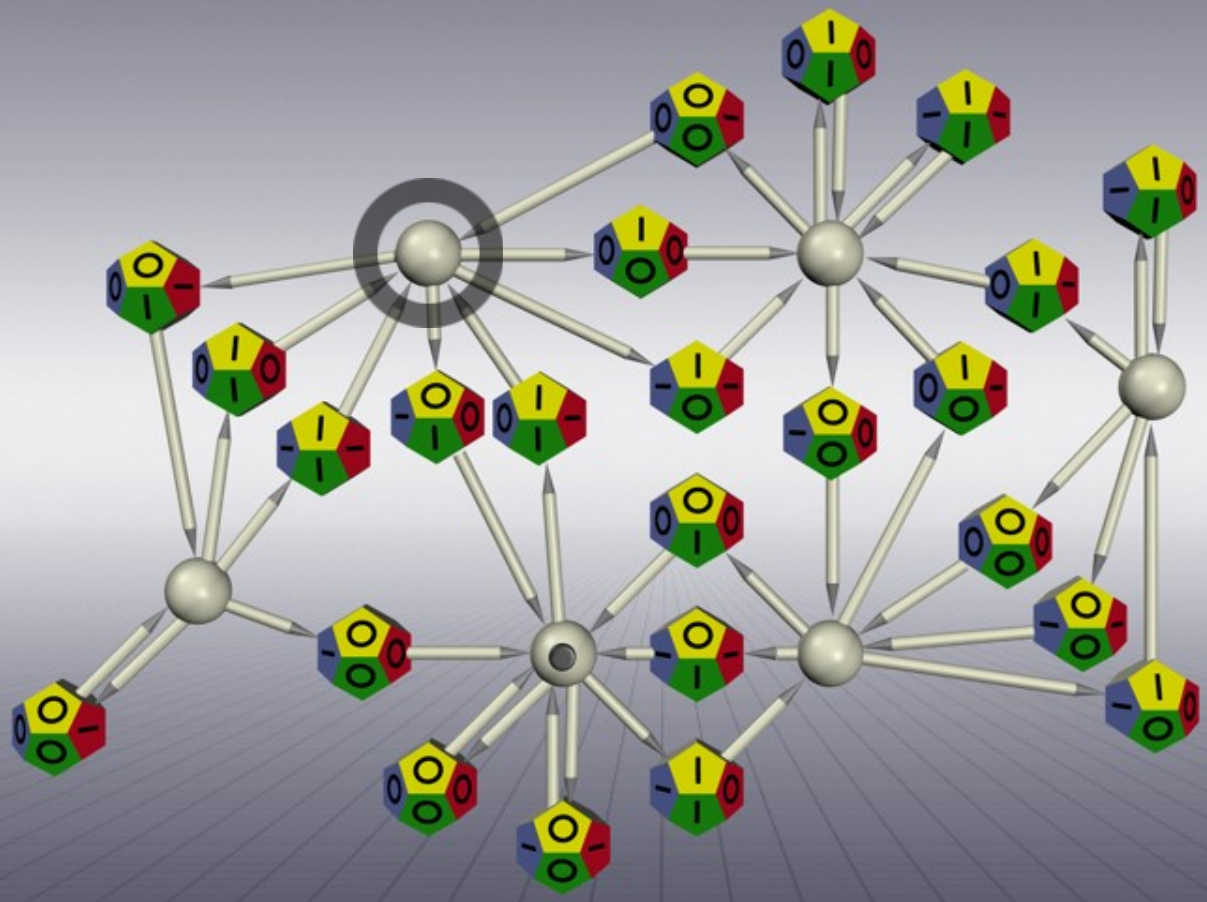
..0000011111 31
..0011110101 245

010100
011101



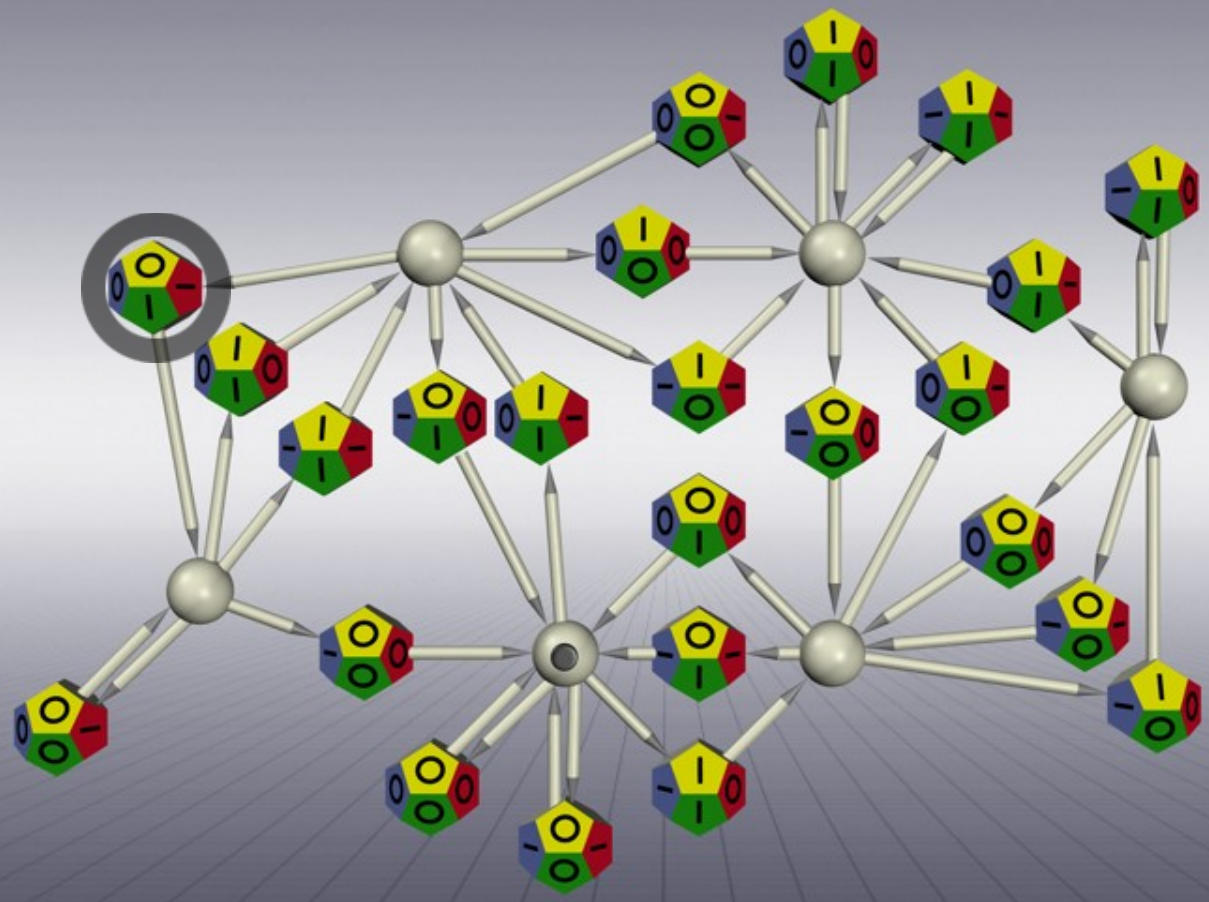
..0000011111 31
..0011110101 245

010100
011101



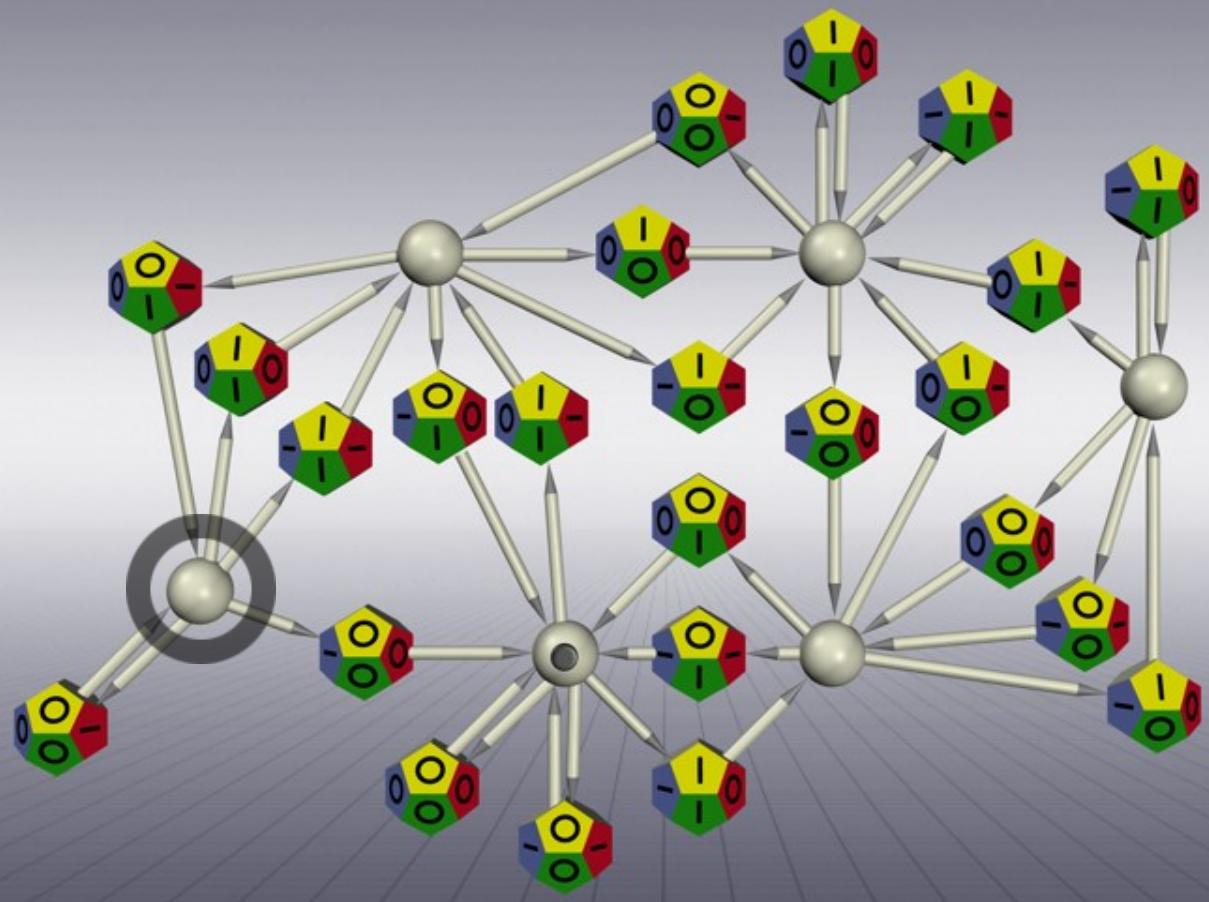
..0000011111 31
..0011110101 245

0010100
1011101



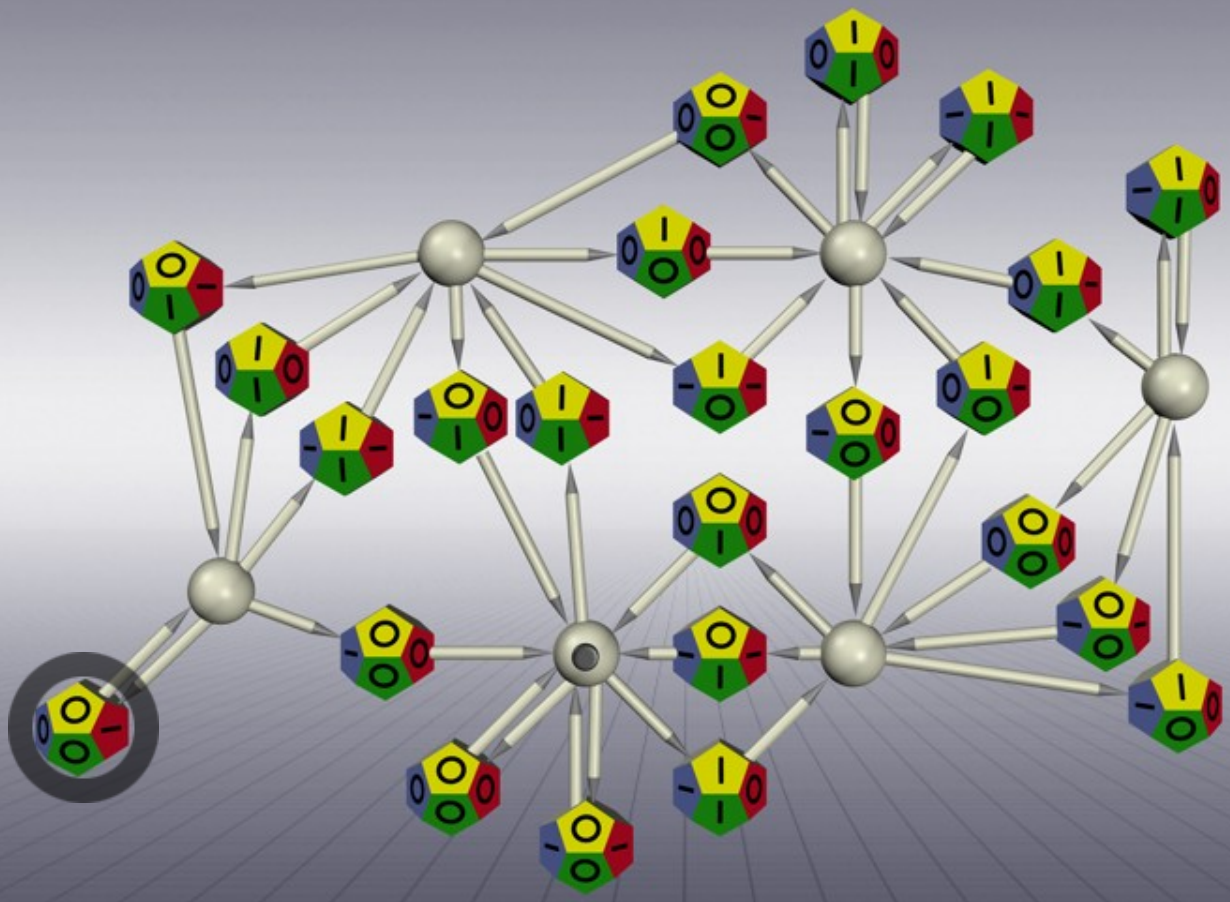
..0000011111 31
..0011110101 245

0010100
1011101



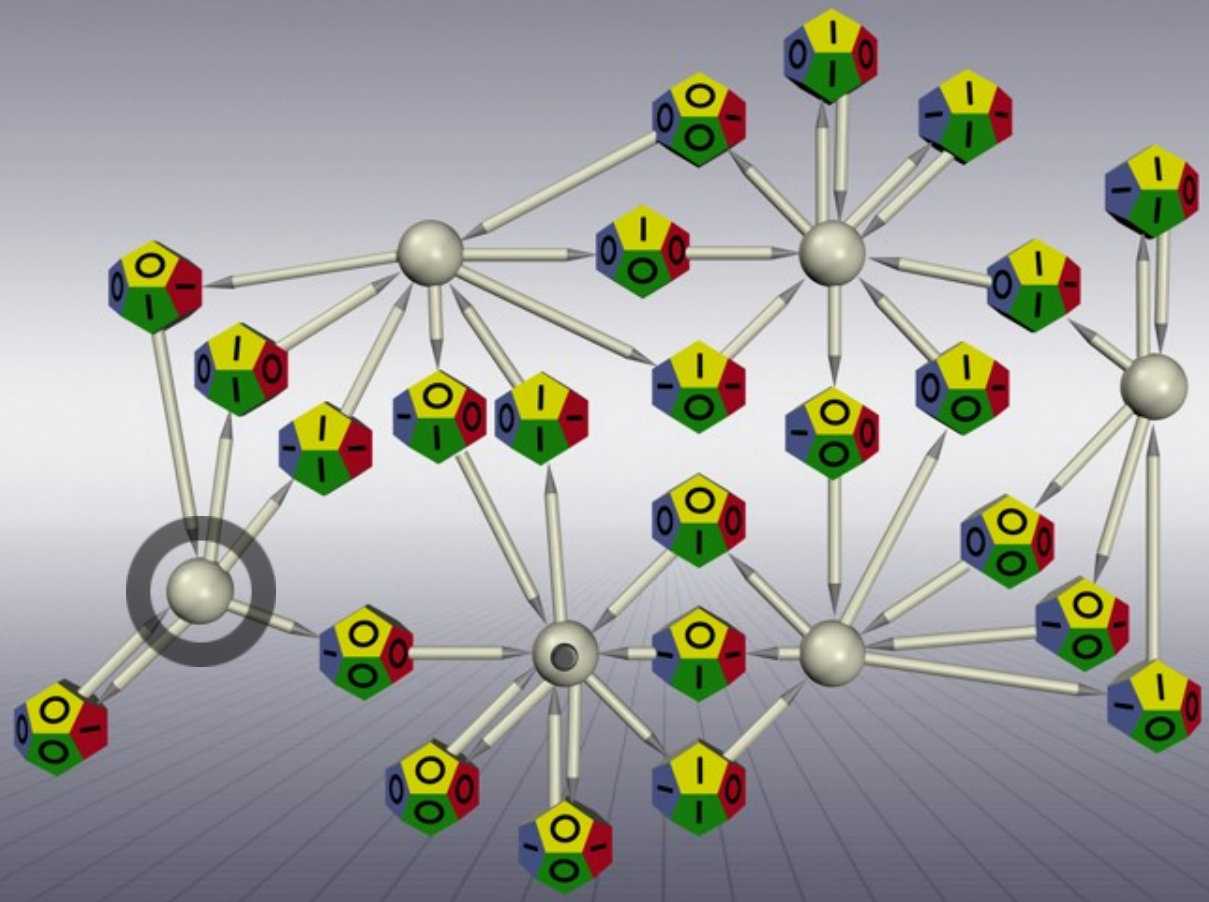
..0000011111 31
..0011110101 245

00010100
01011101



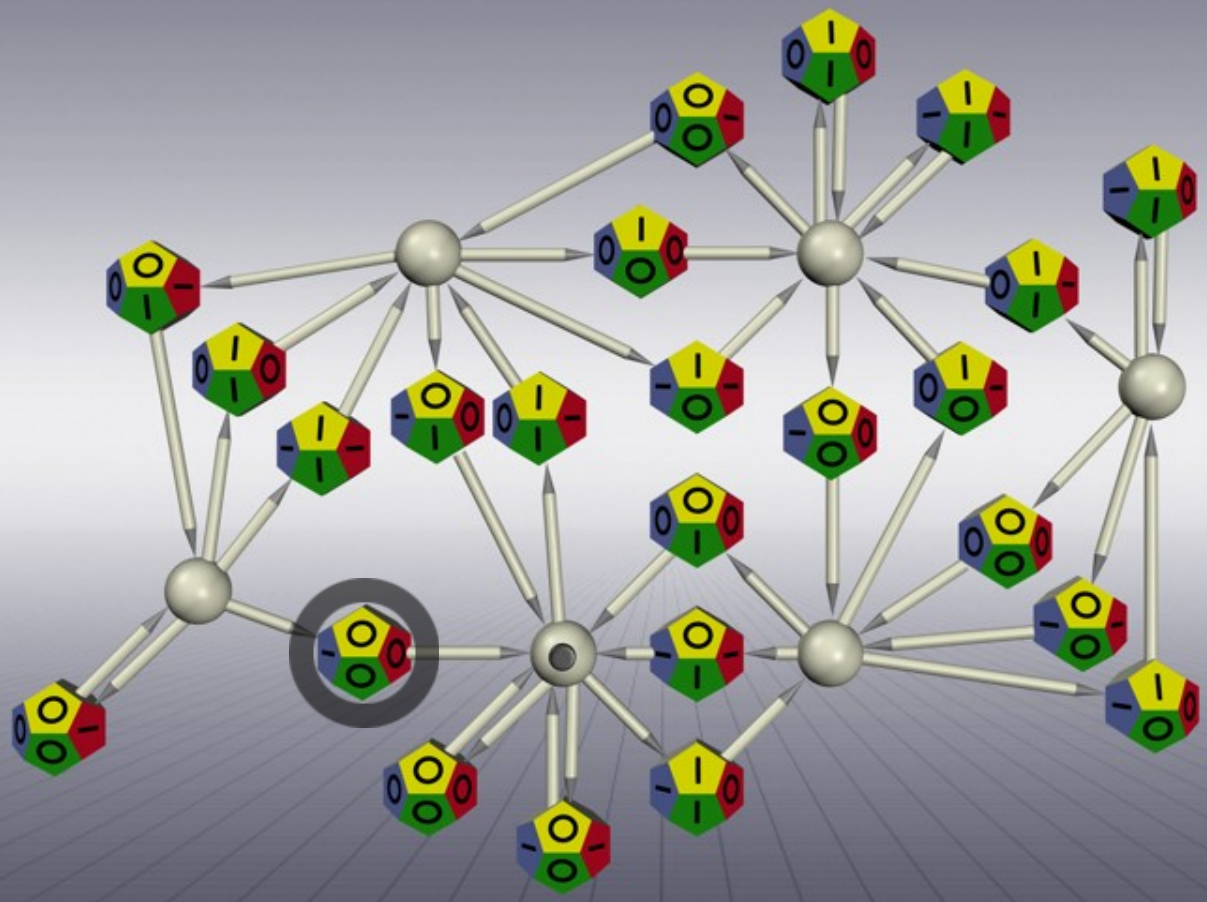
..0000011111 31
..0011110101 245

00010100
01011101



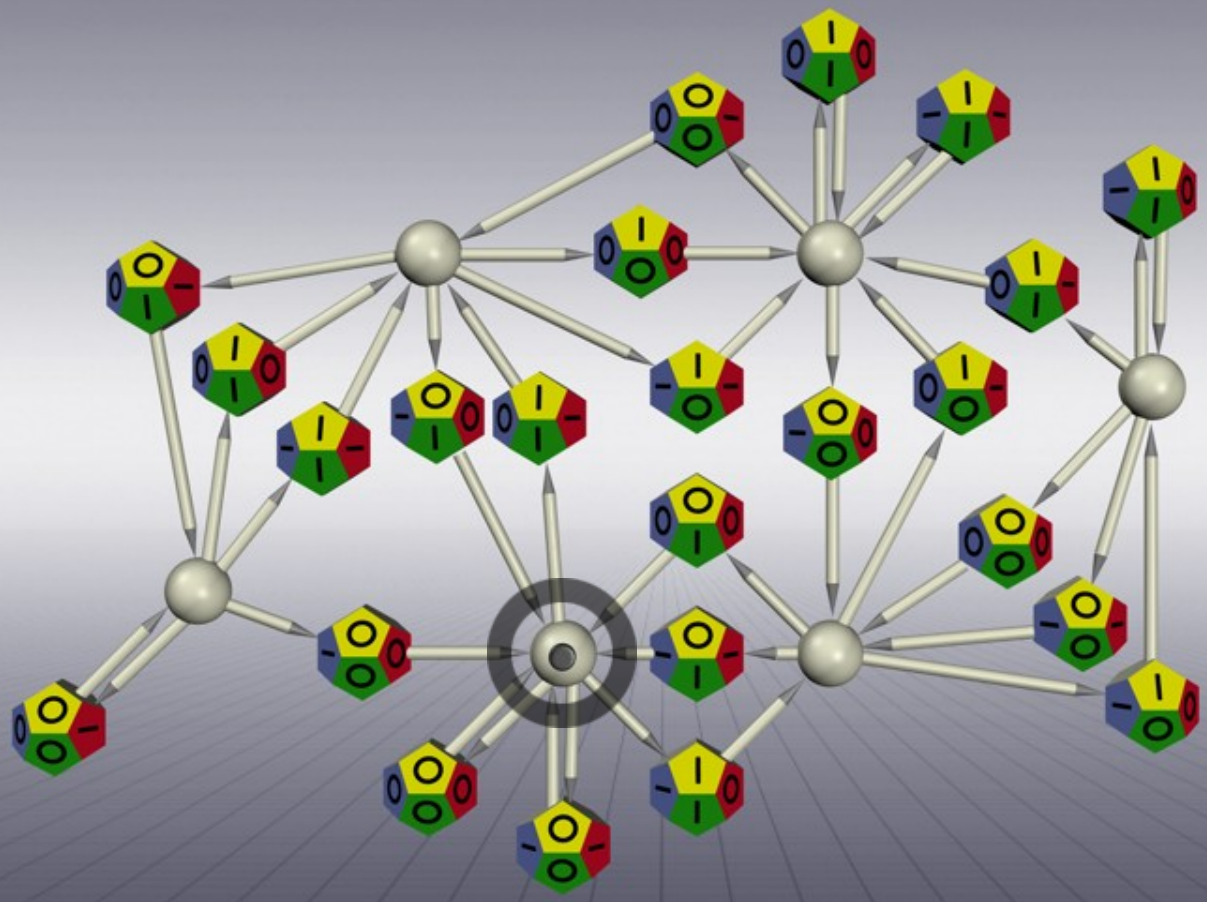
..0000011111 31
..0011110101 245

1000101000
0010111010



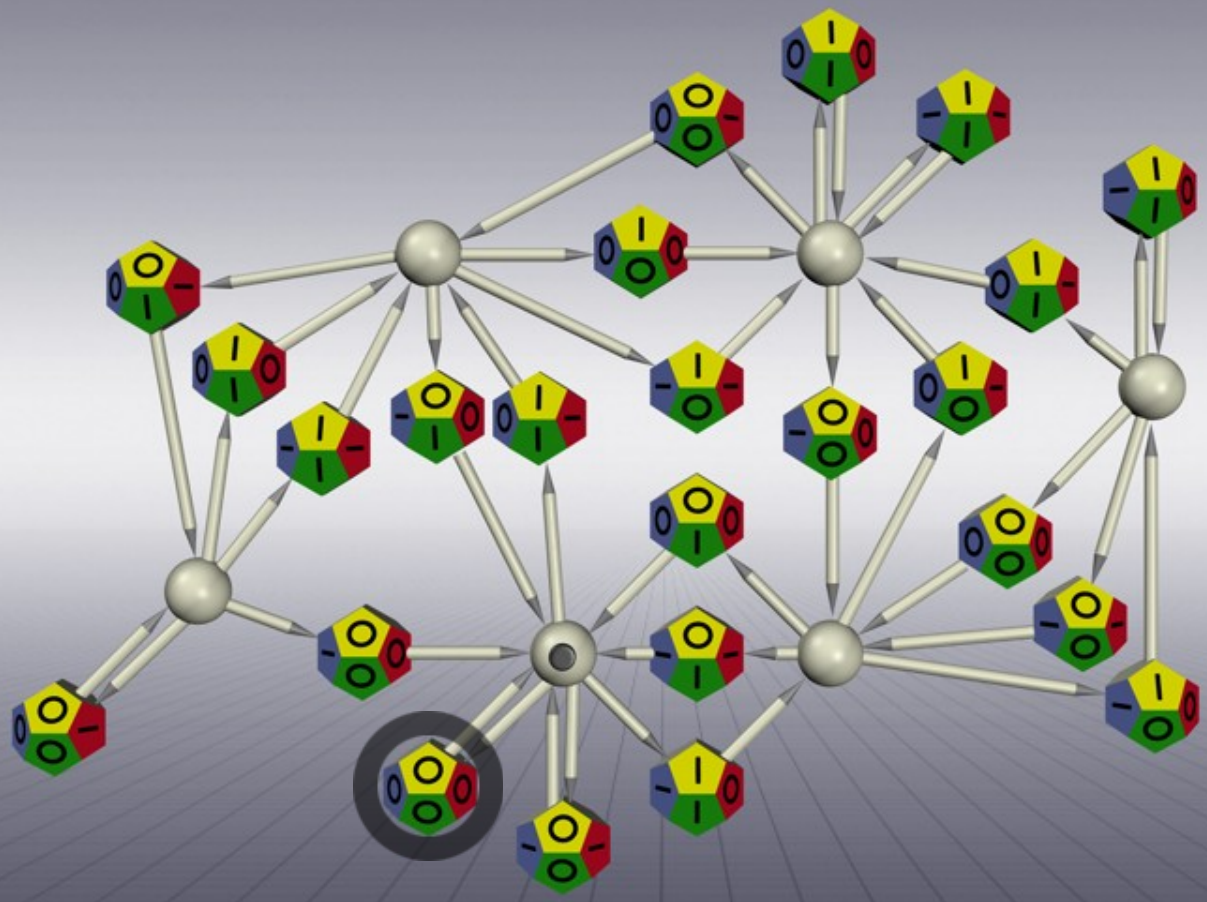
..0000011111 31
..0011110101 245

100010100
001011101



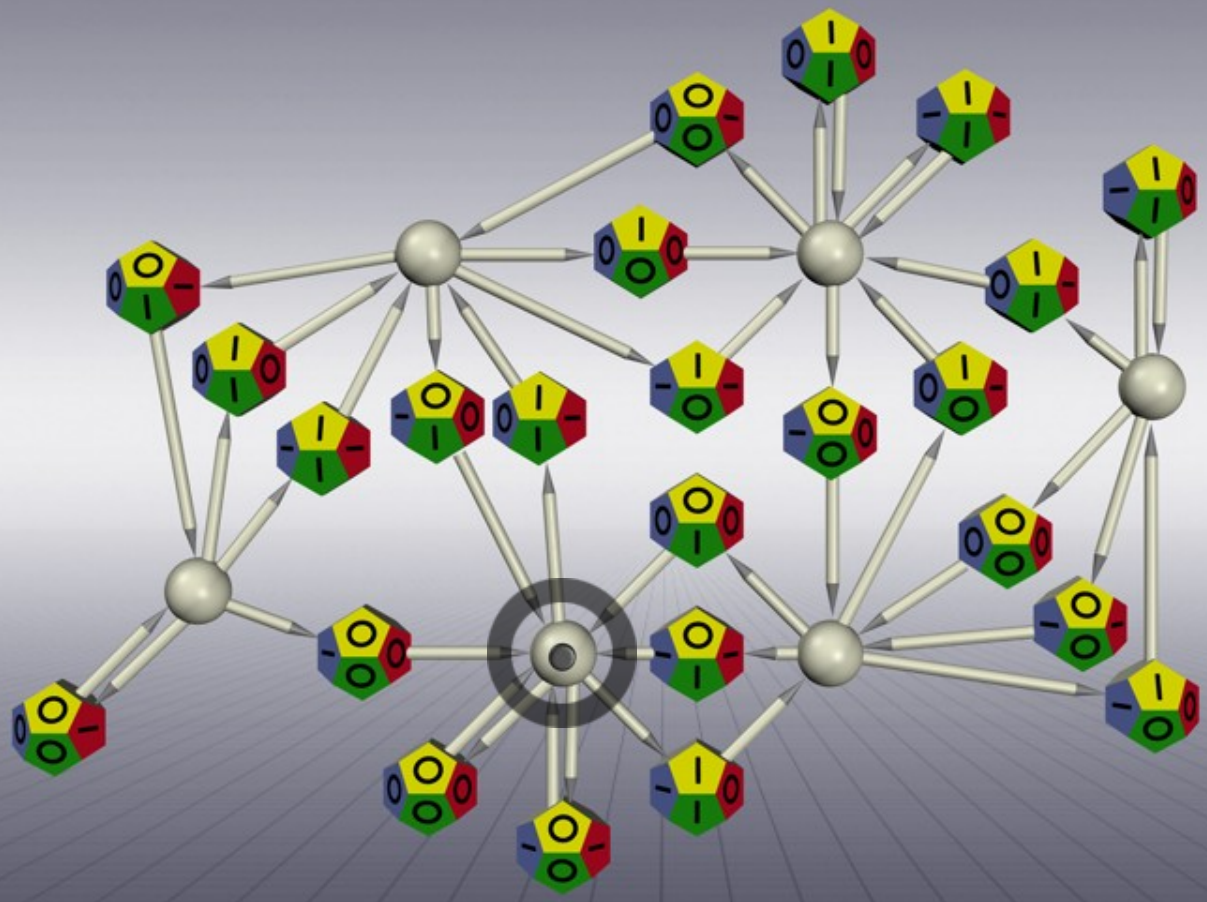
..0000011111 31
..0011110101 245

0100010100
0001011101



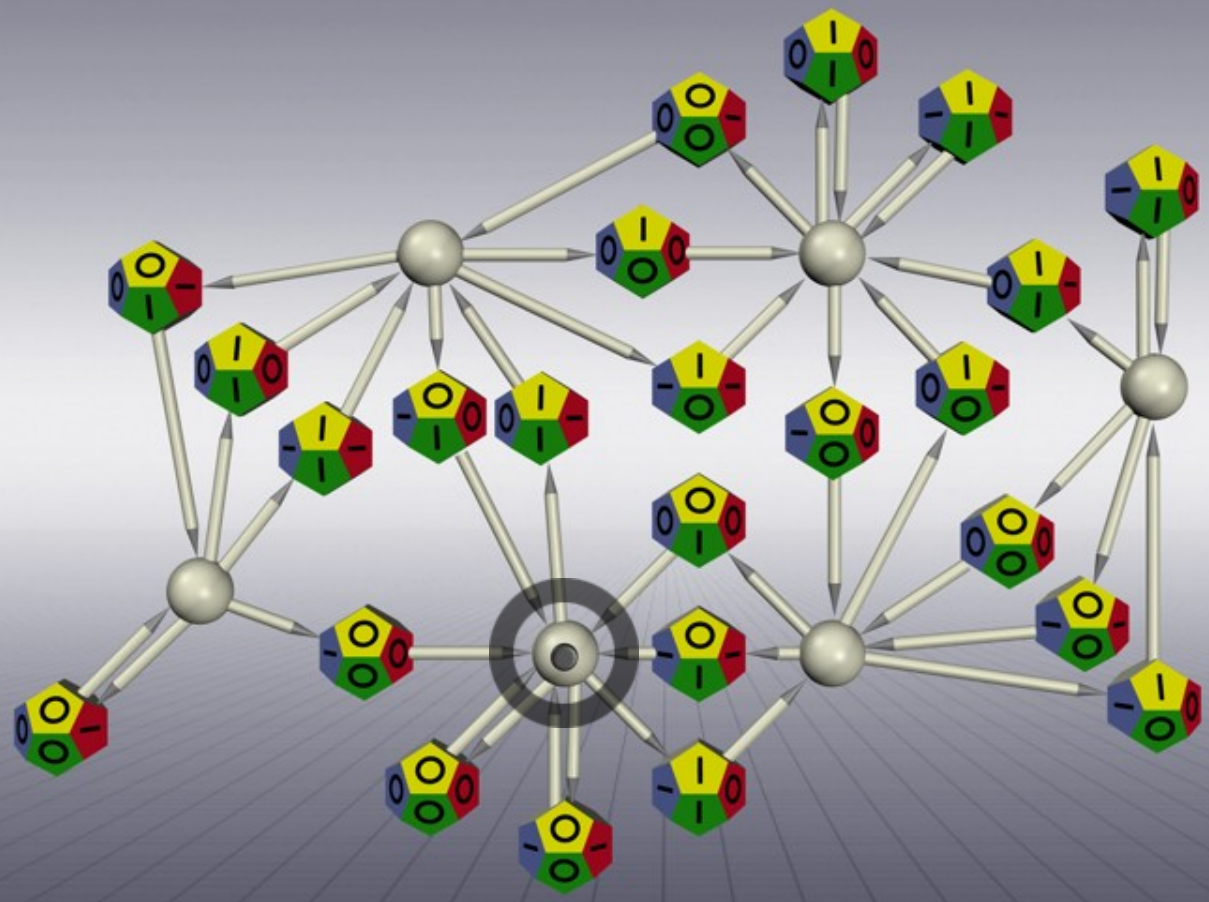
..0000011111 31
..0011110101 245

0100010100
0001011101



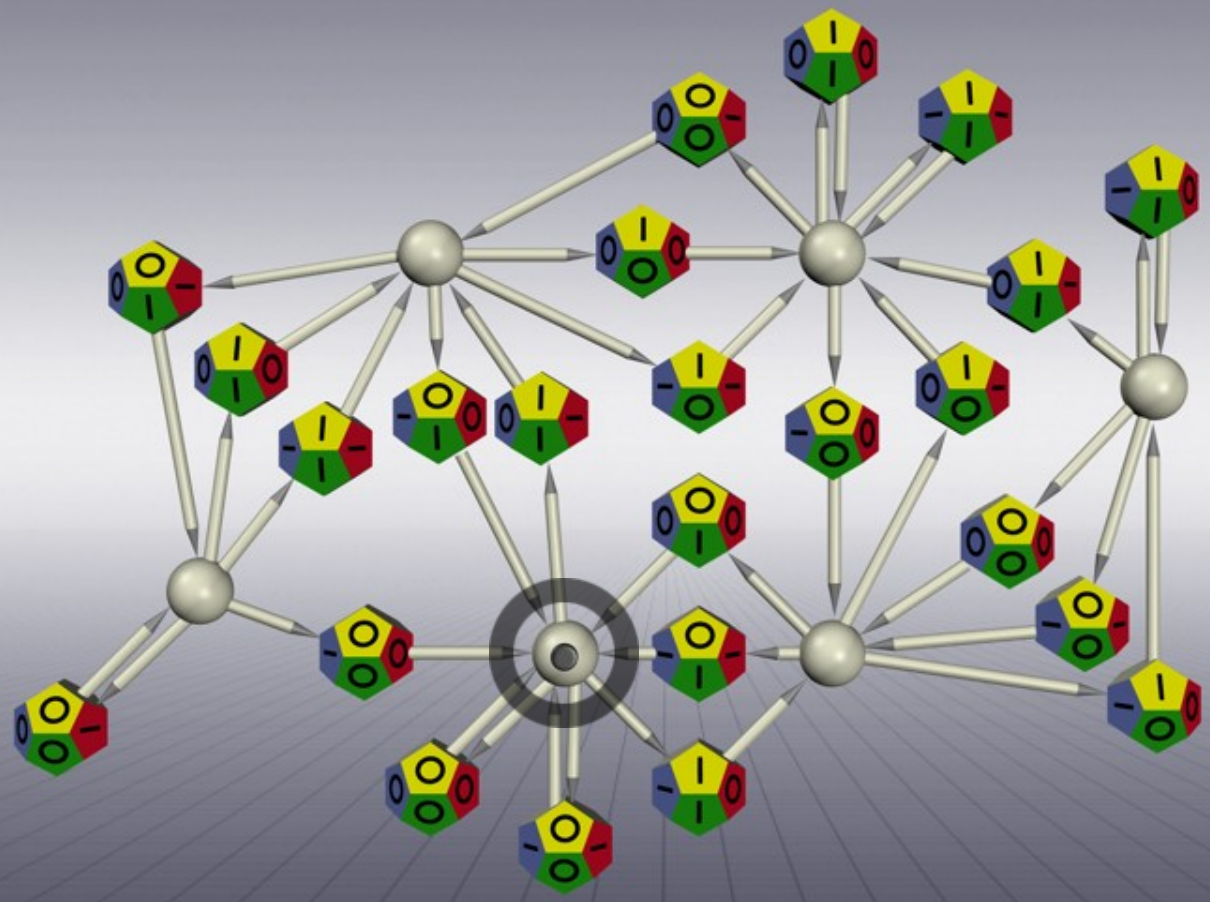
..0000011111 31
..0011110101 245

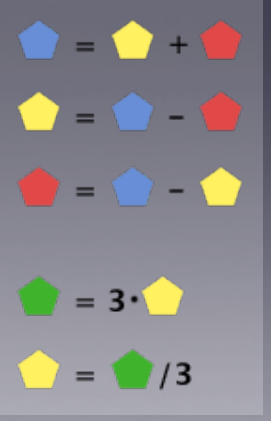
..0100010100
..0001011101



..0000011111 31
..0011110101 245

..0100010100 276
..0001011101 93

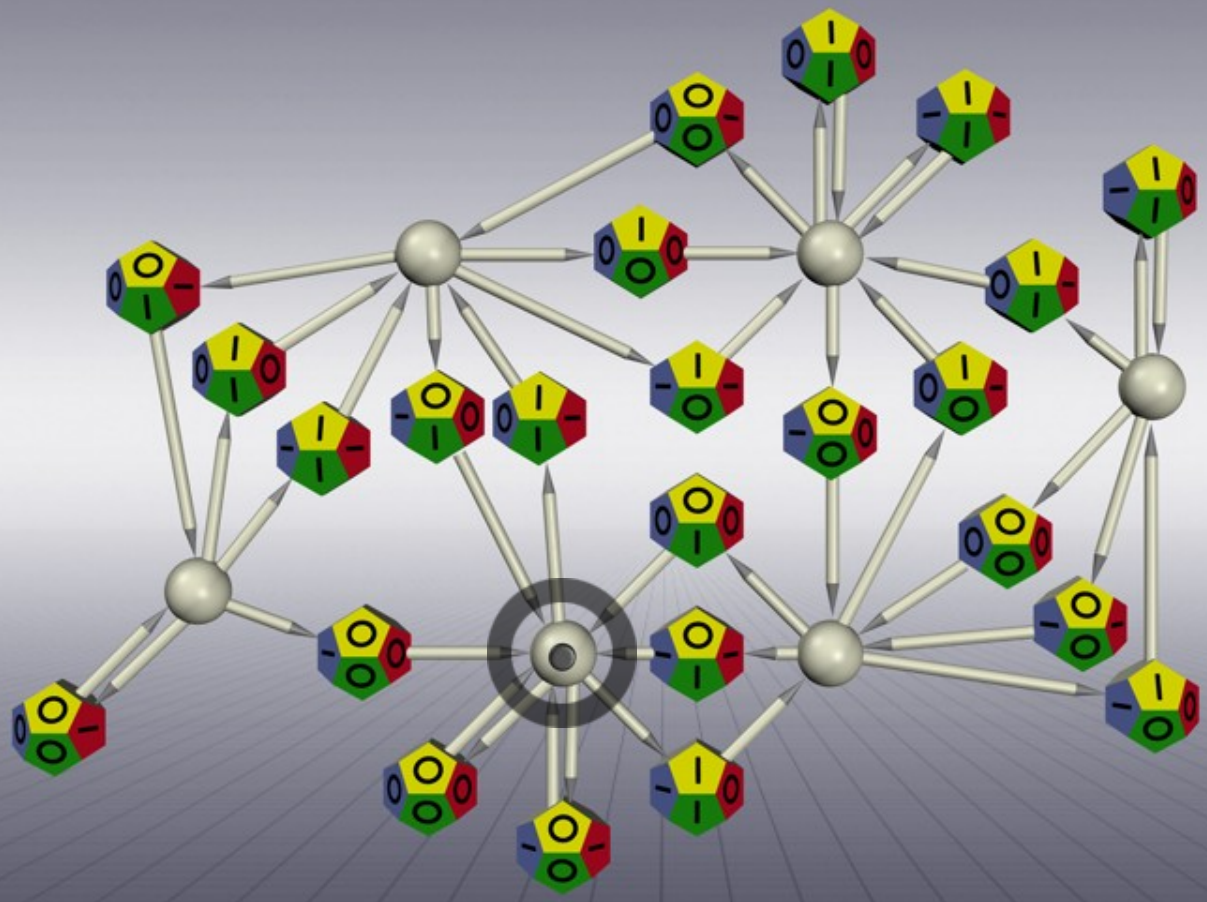




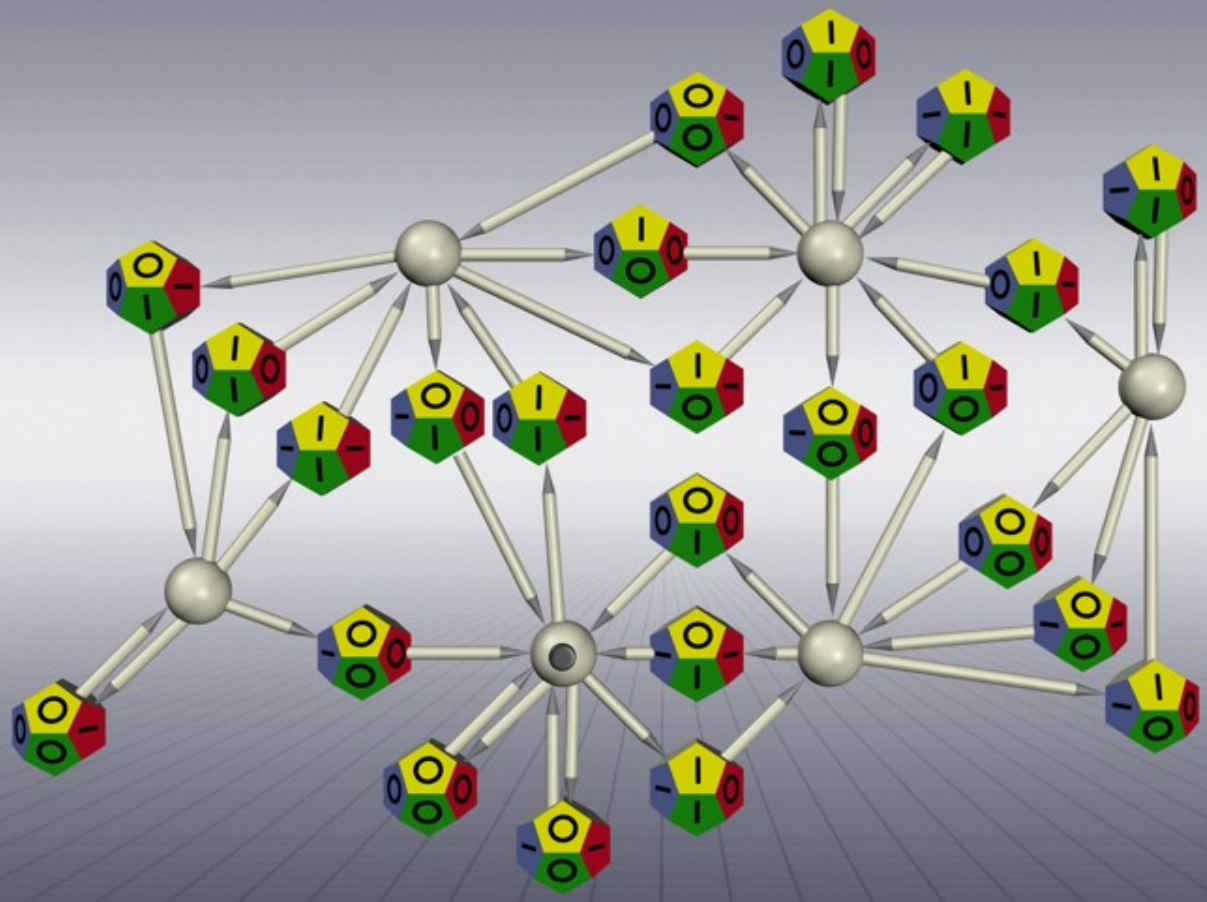
276 = 31 + 245

93 = 3 * 31

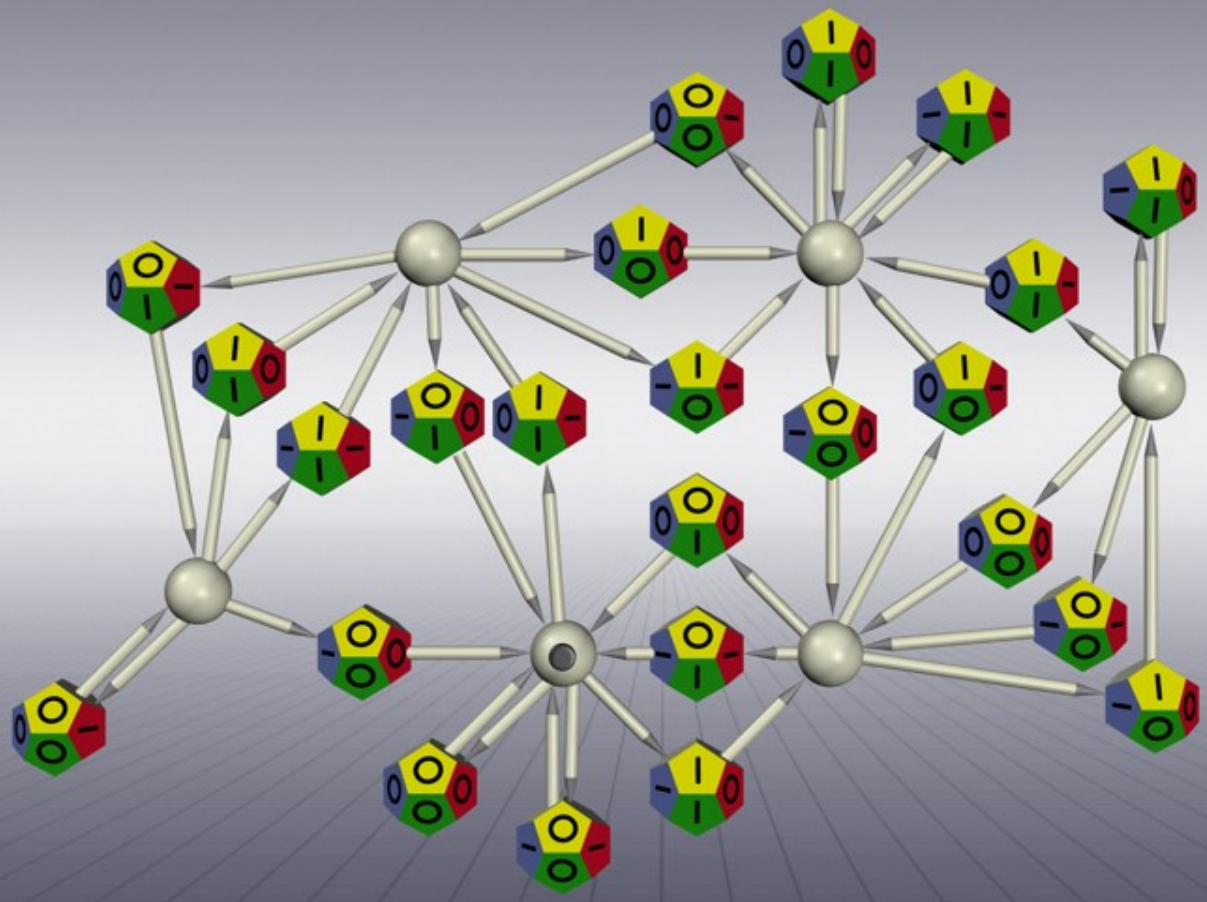
```
..0000011111   31
..0011110101  245

..0100010100  276
..0001011101   93
```



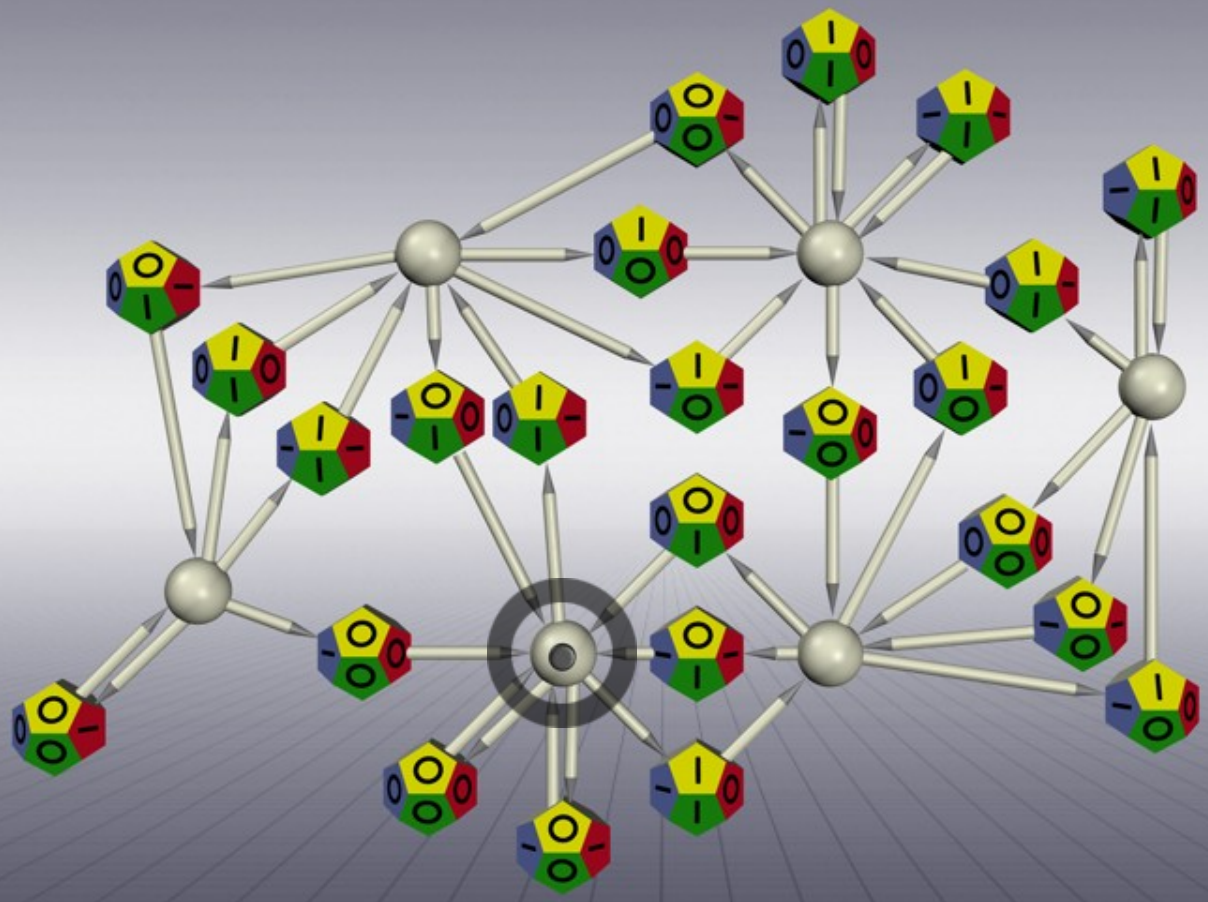
..01000010100 276



..0100010100 276

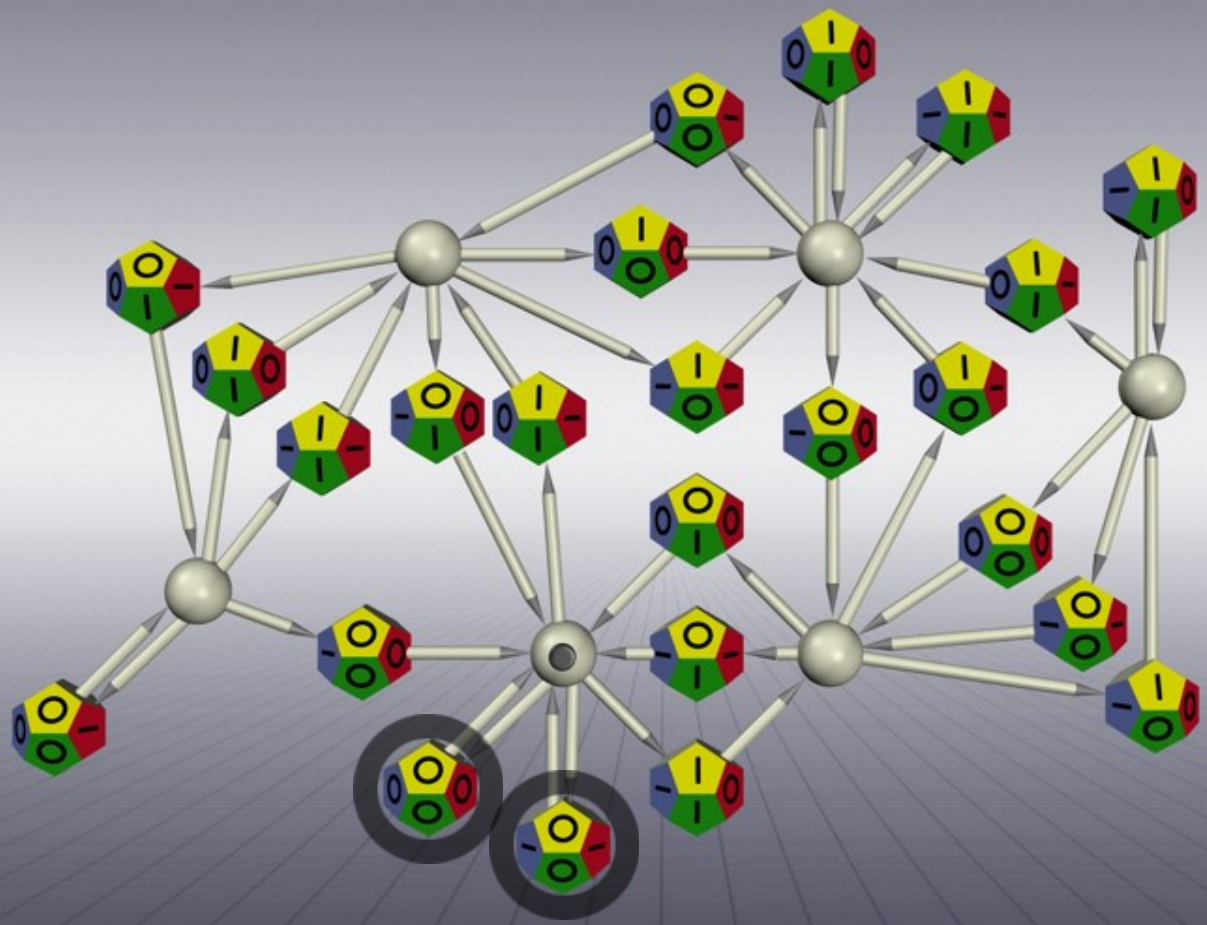


..0100010100 276



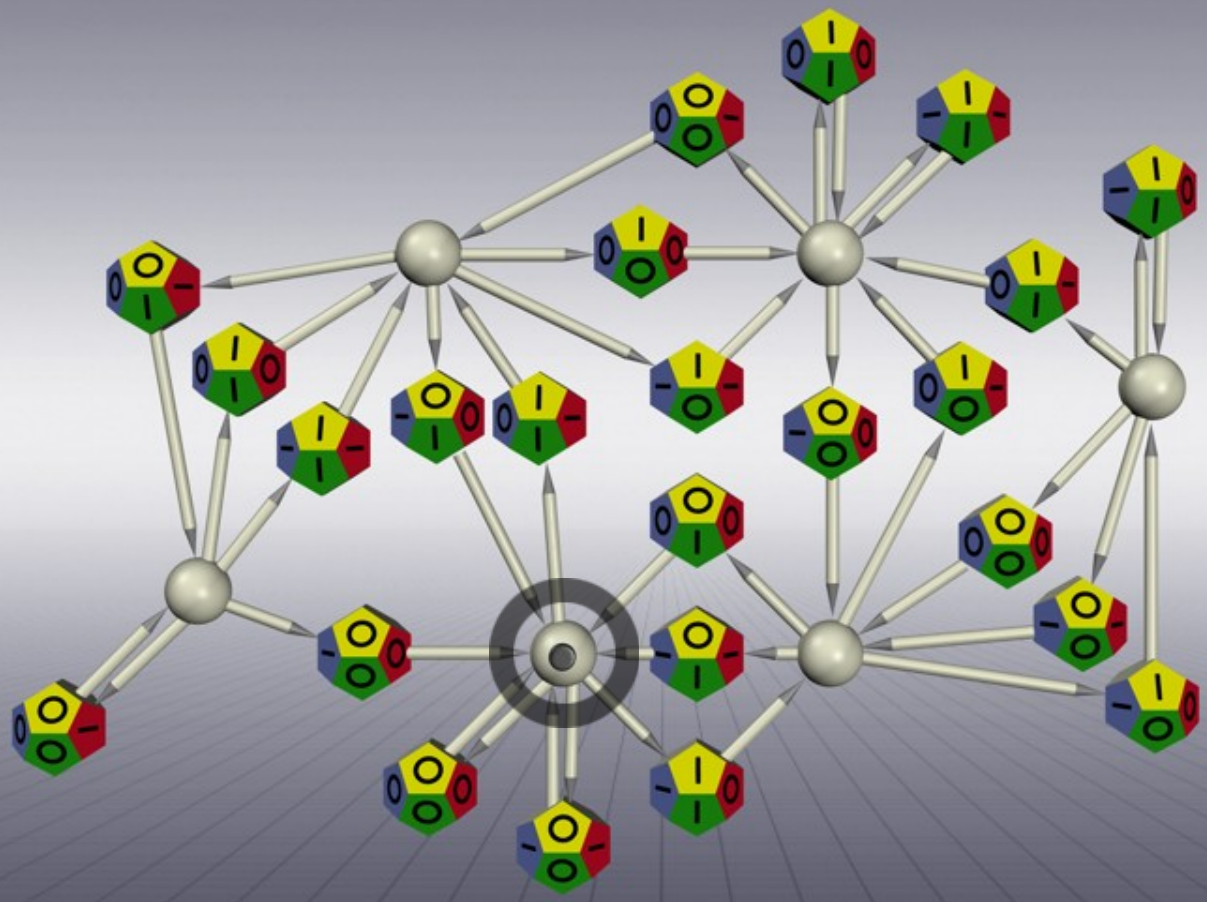
0

..0100010100 276



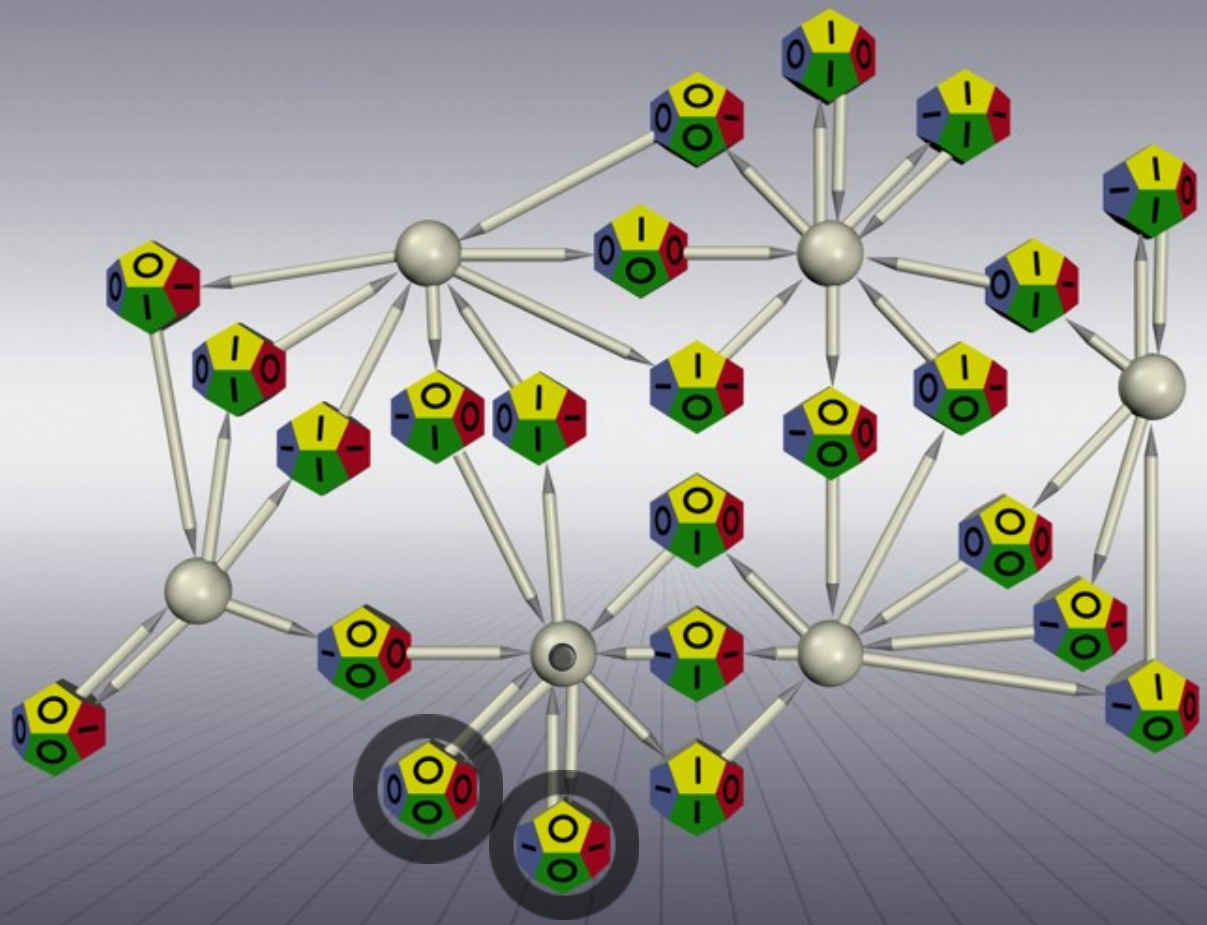
0

..0100010100 276



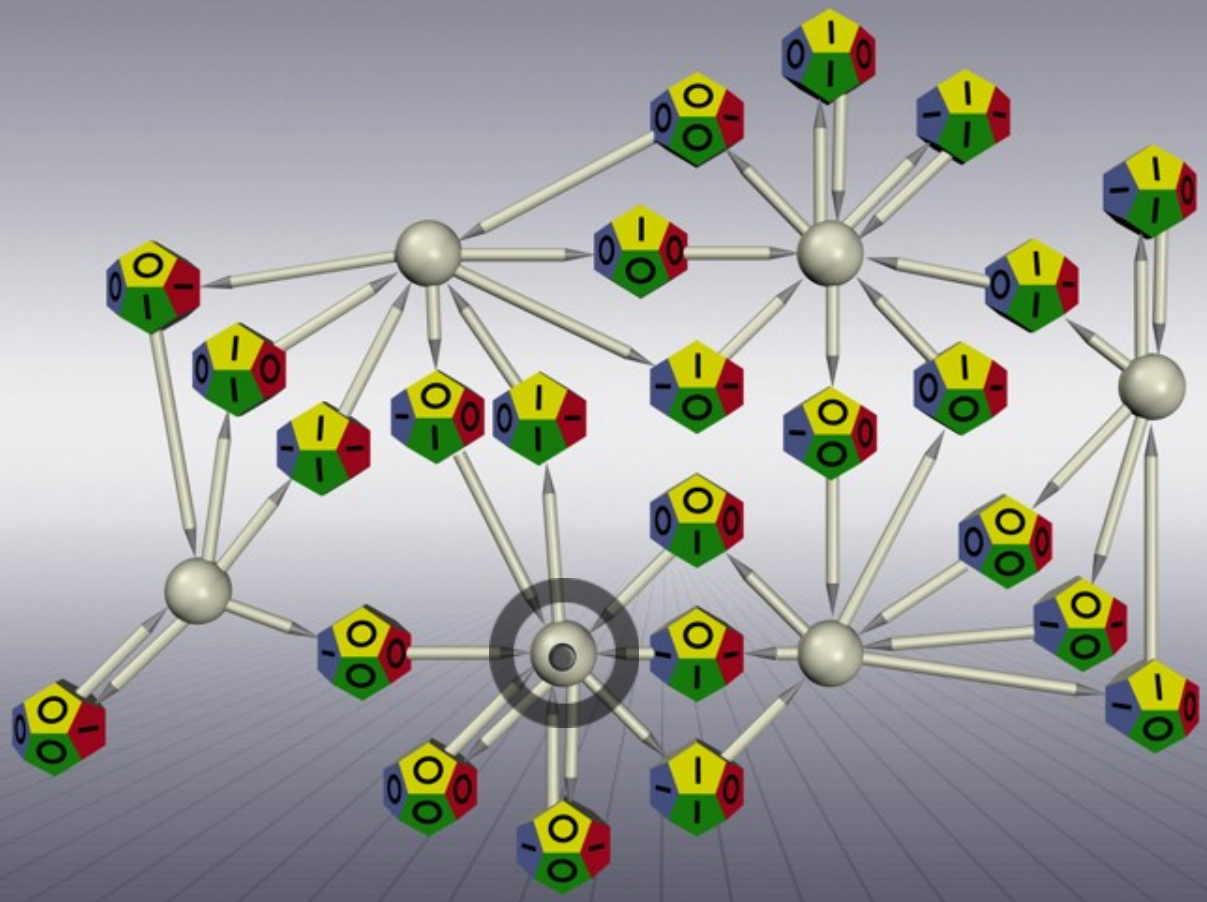
00

..0100010100 276



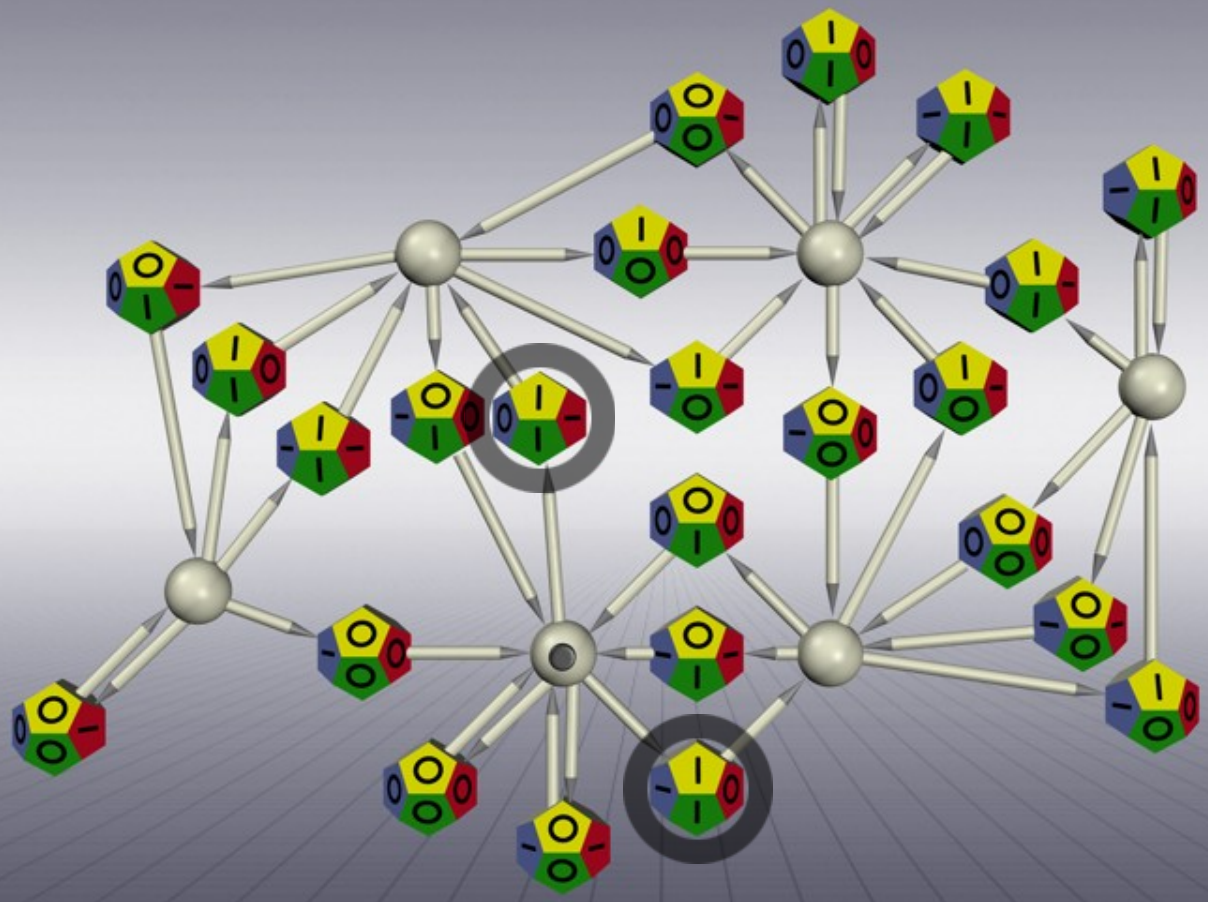
00

..0100010100 276



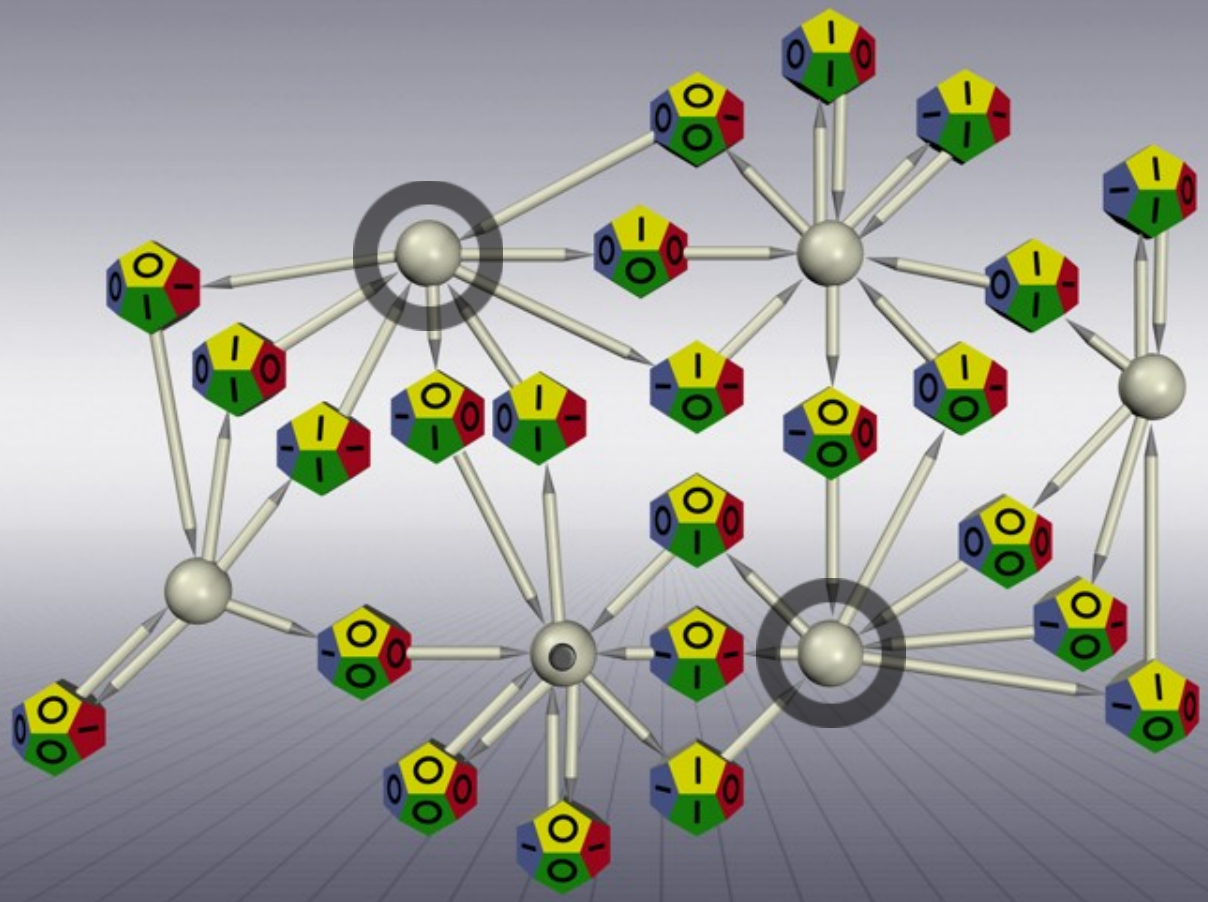
100

..0100010100 276



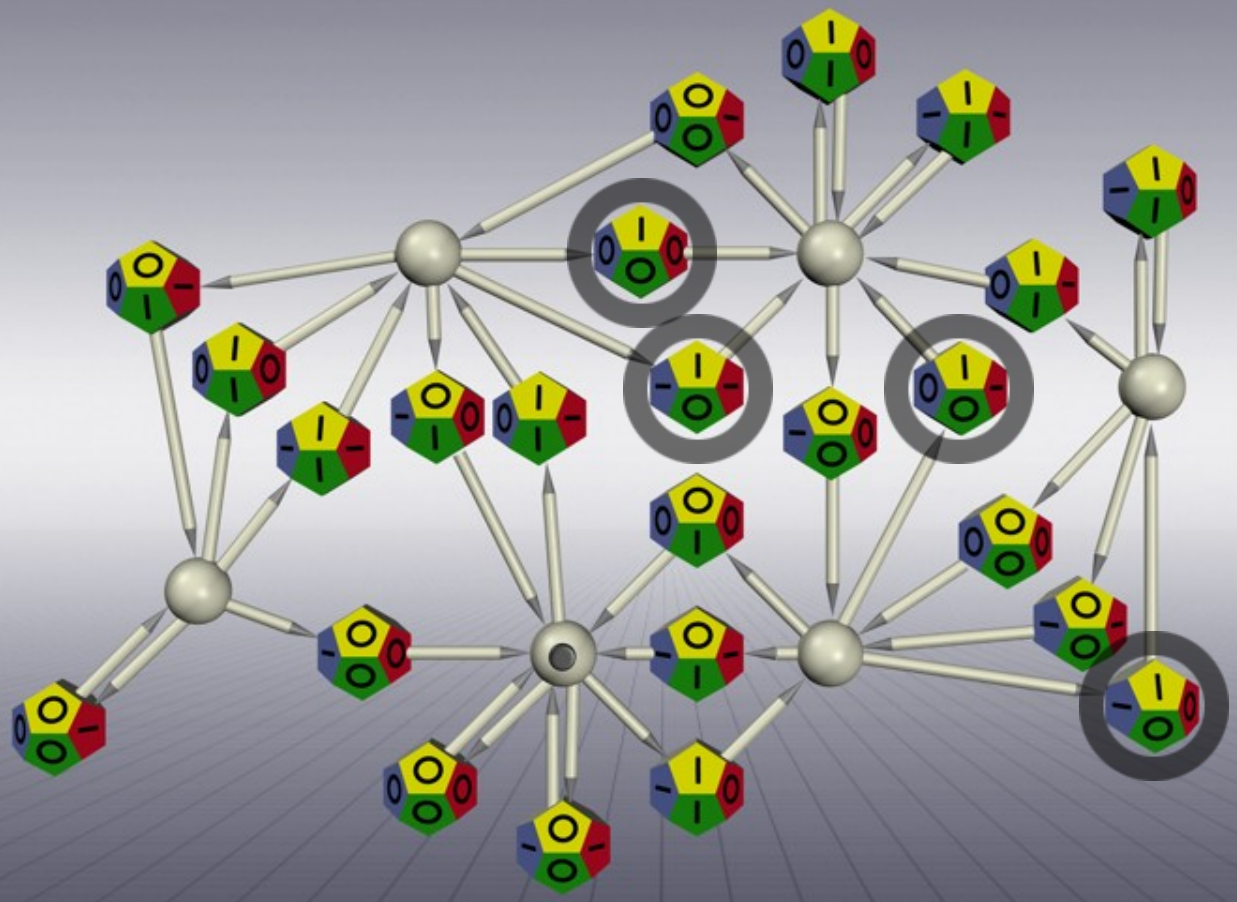
100

..0100010100 276



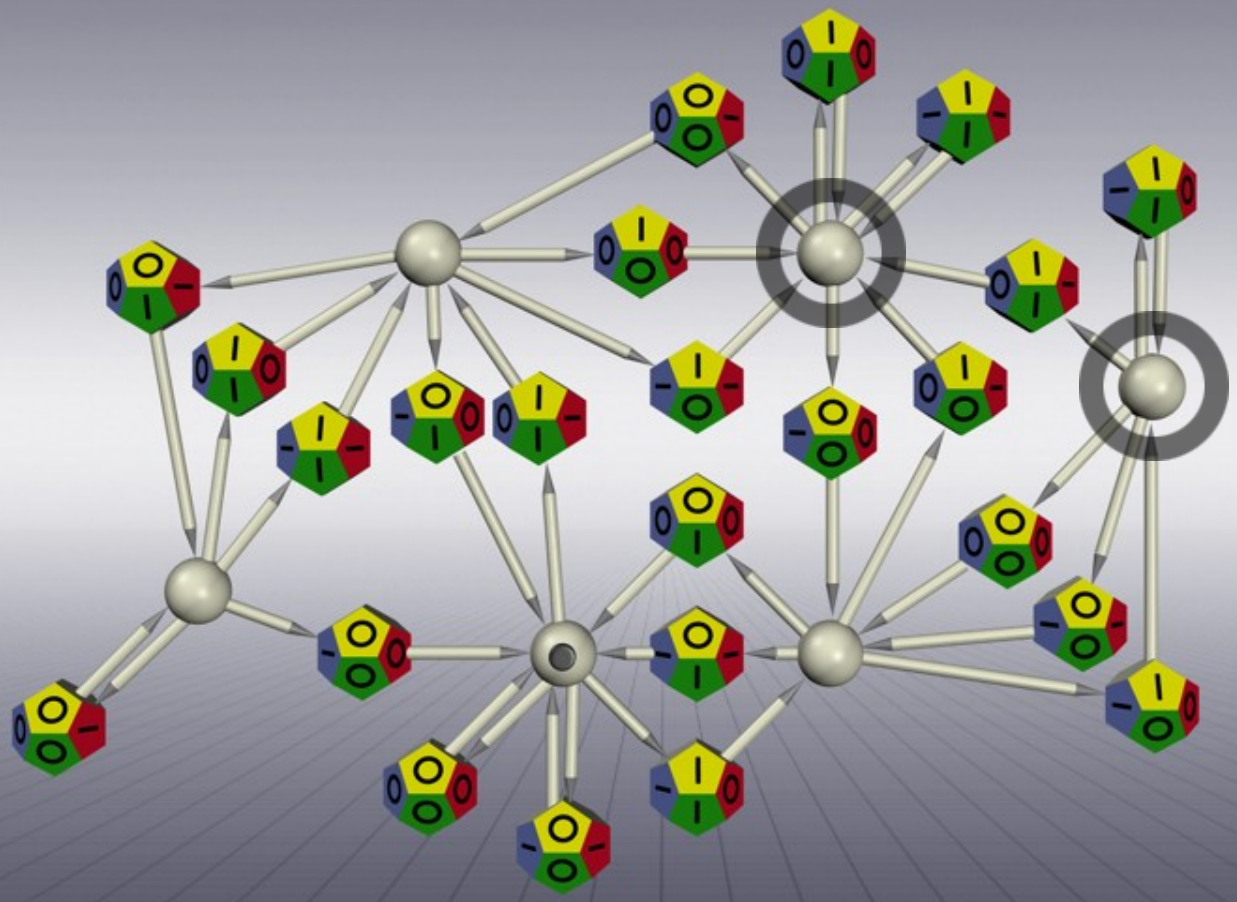
1100

..0100010100 276



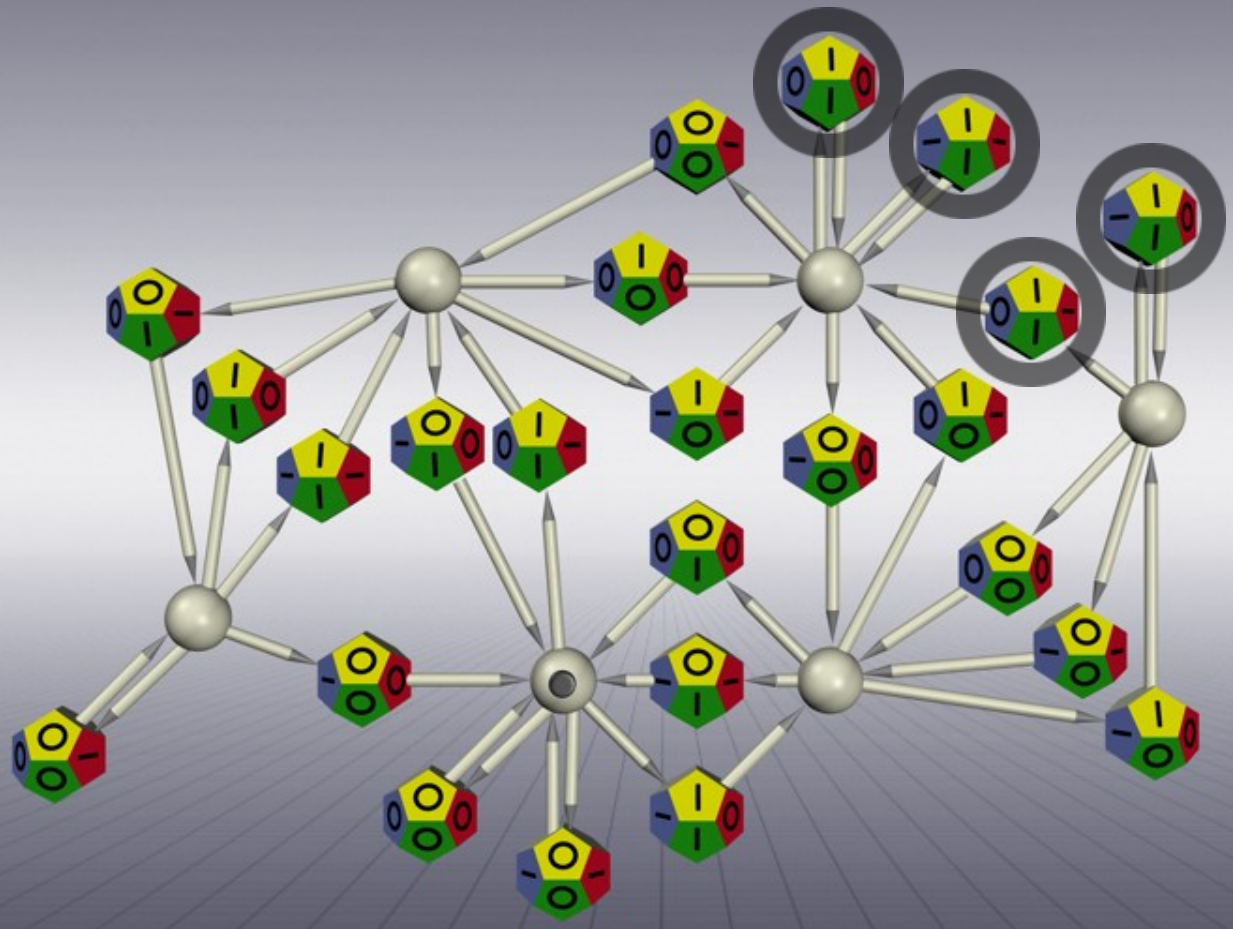
1100

..0100010100 276



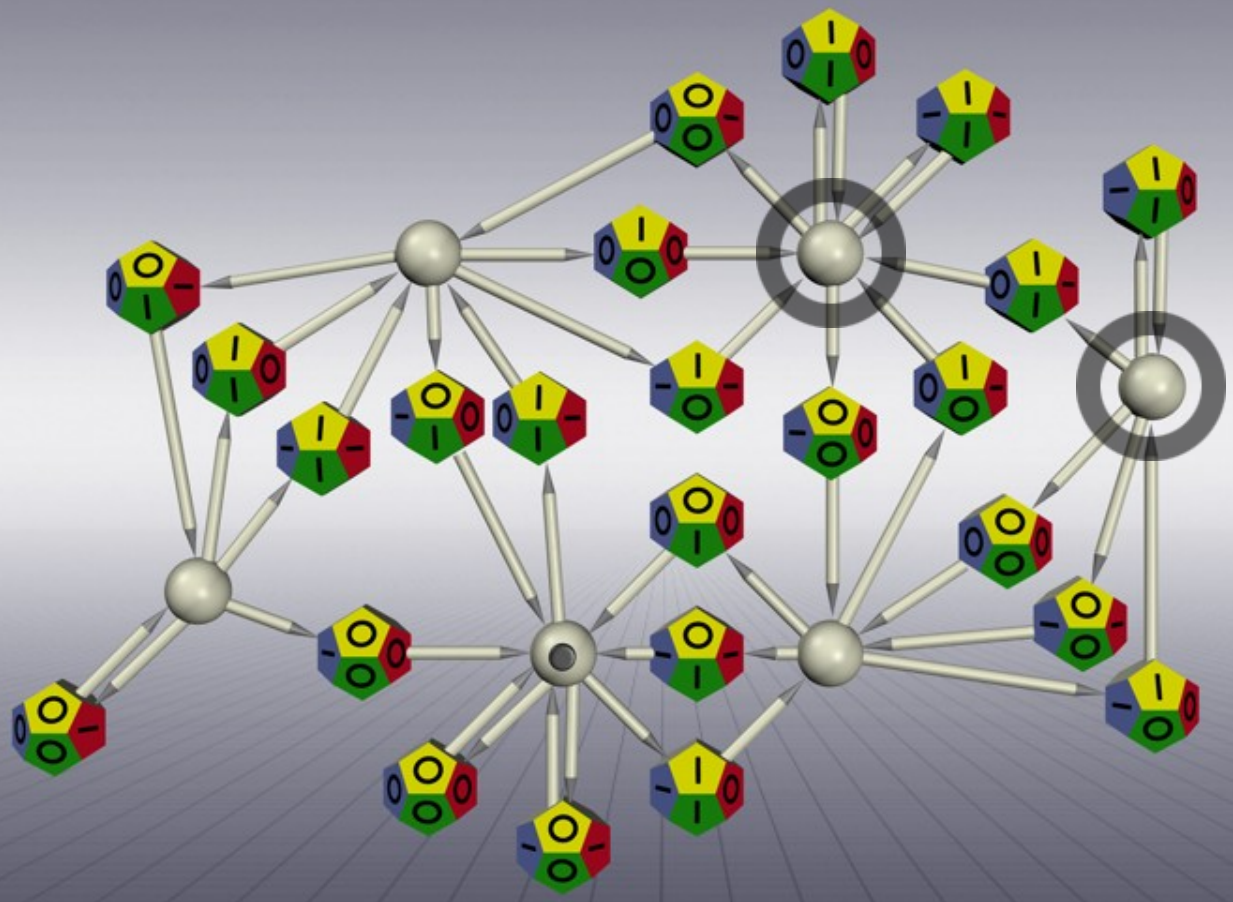
11100

..0100010100 276



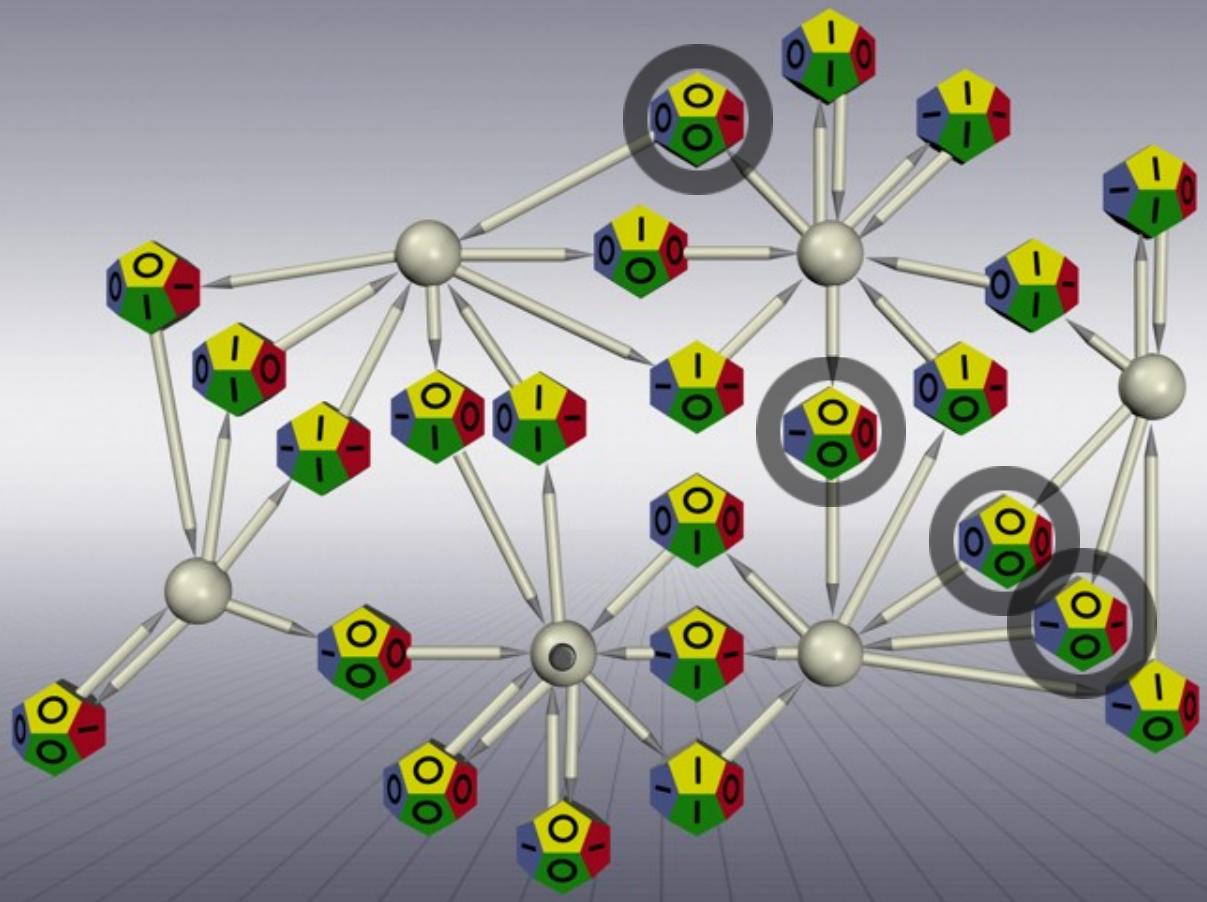
11100

..0100010100 276



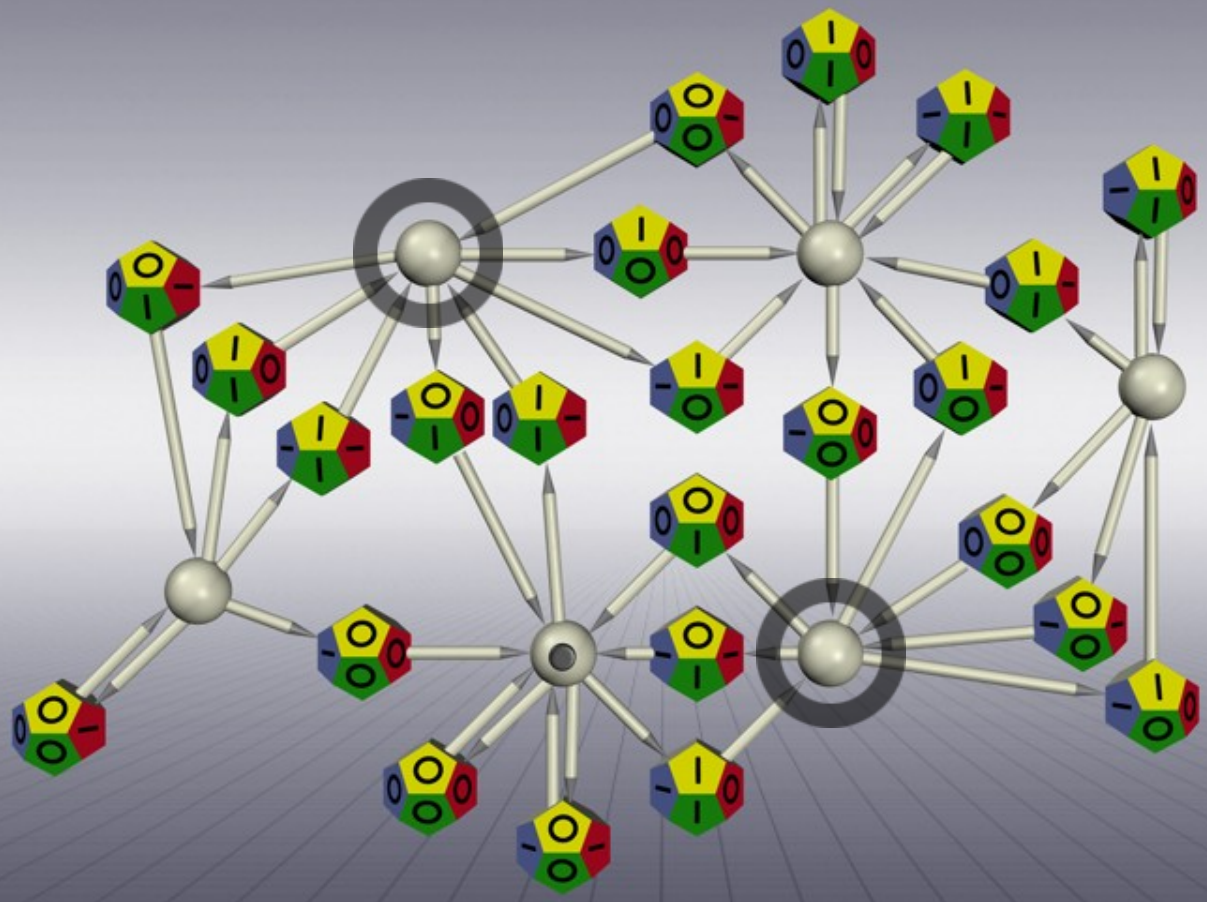
011100

..0100010100 276



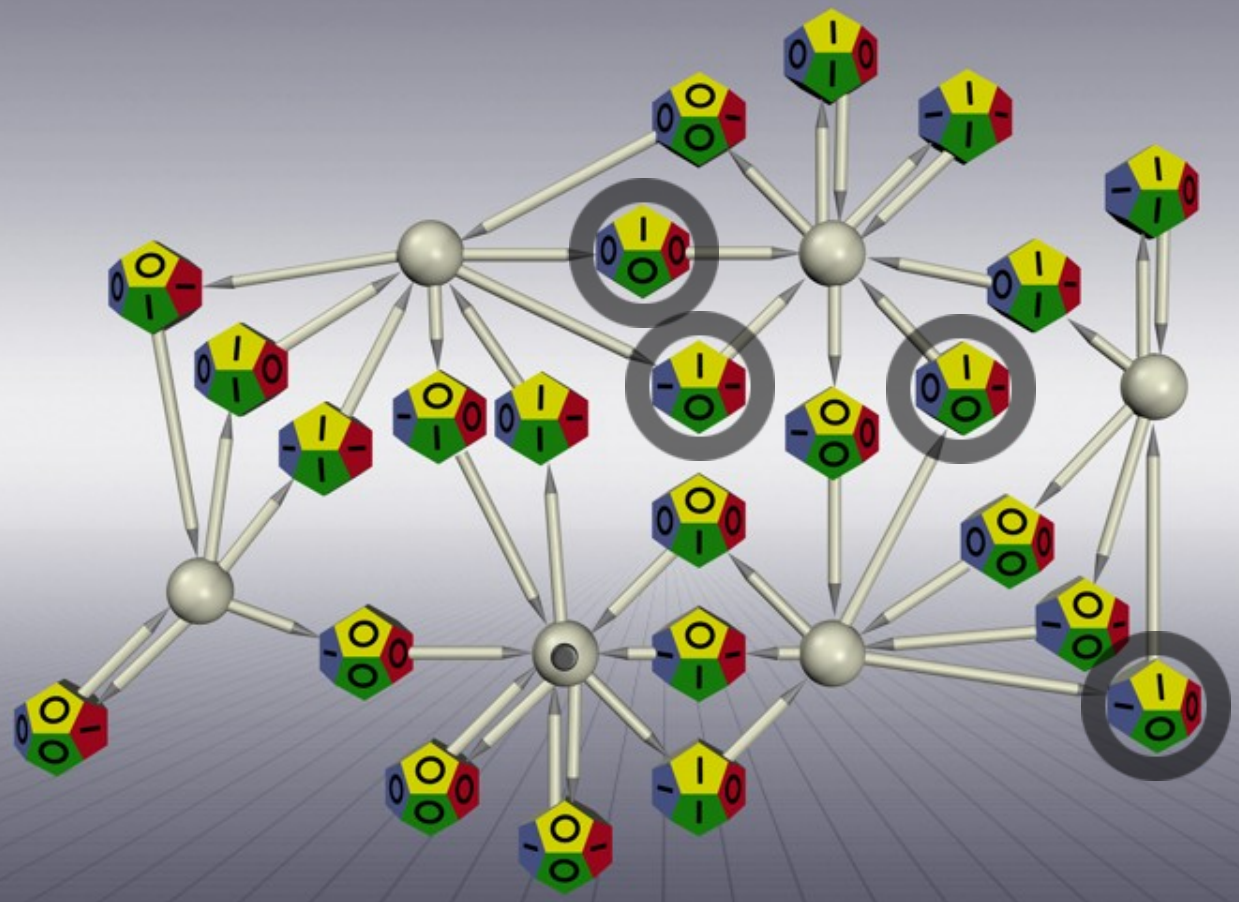
011100

..0100010100 276



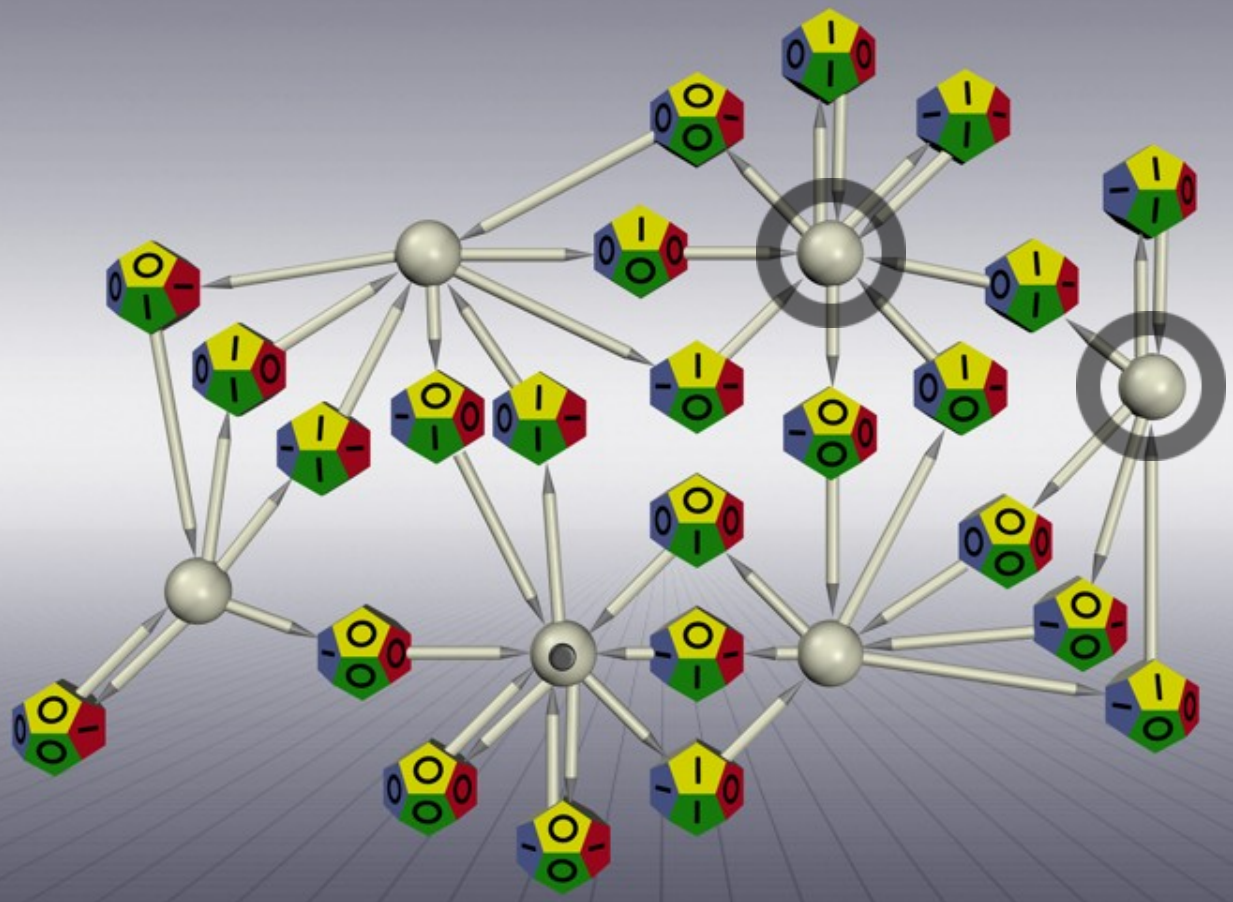
1011100

..0100010100 276



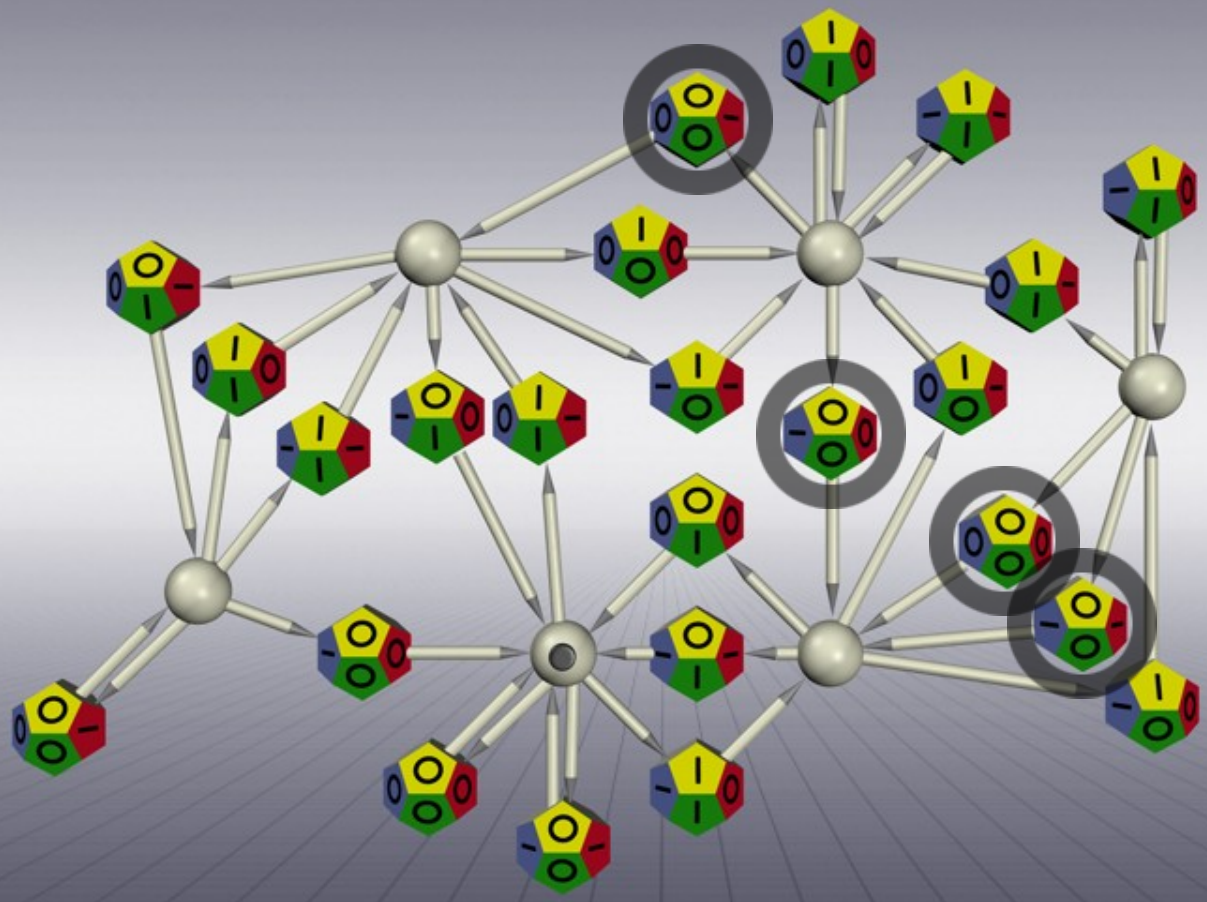
1011100

..0100010100 276



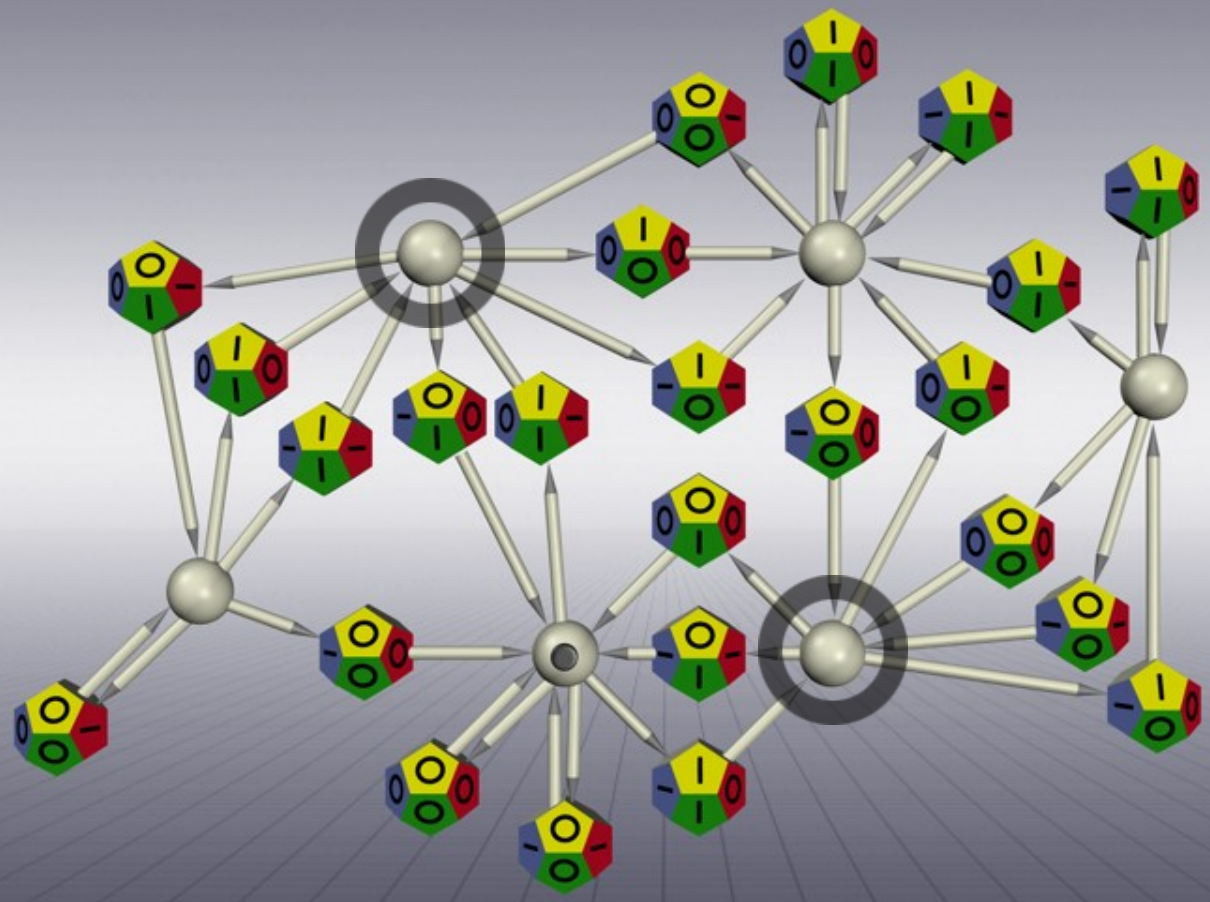
01011100

..0100010100 276



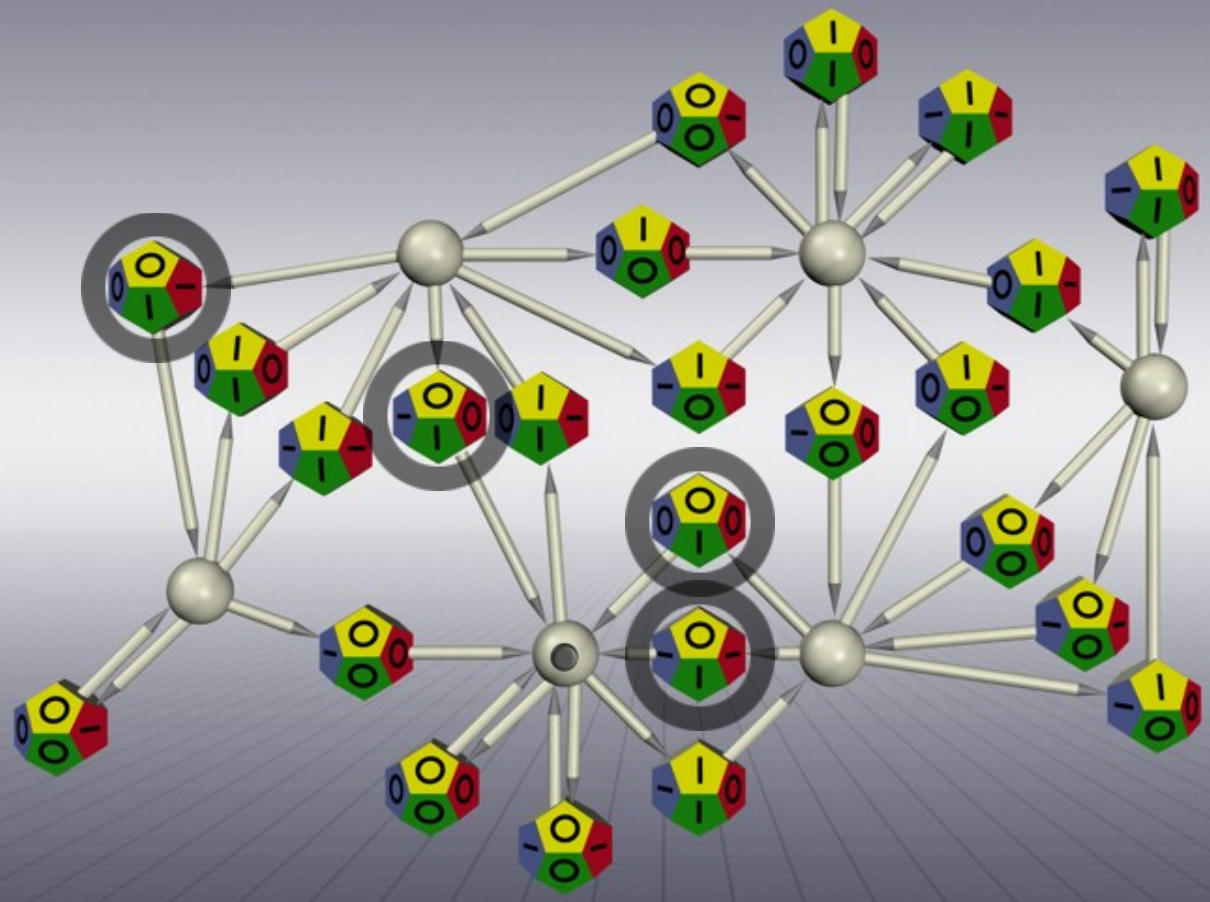
01011100

..0100010100 276



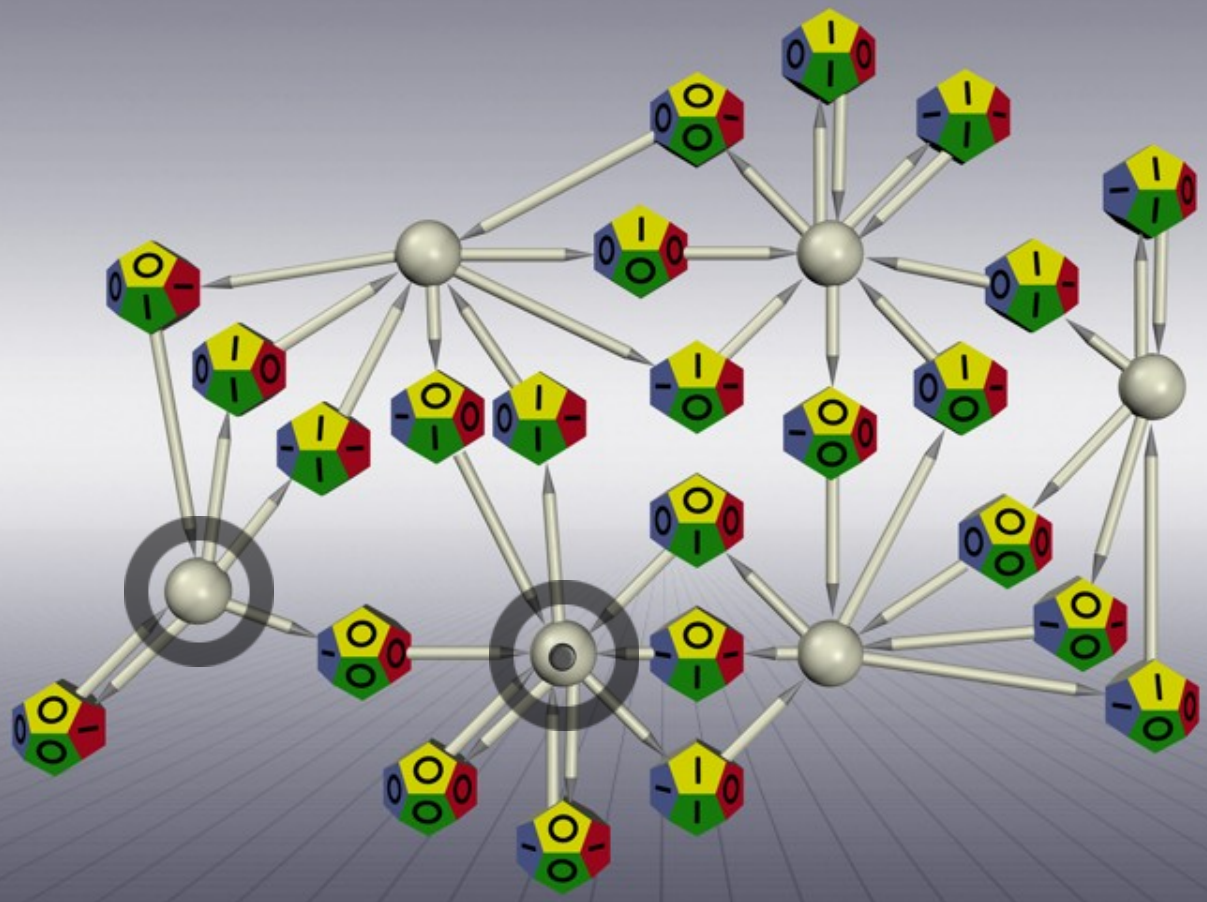
0 0 1 0 1 1 1 0 0

..0 1 0 0 0 1 0 1 0 0 276



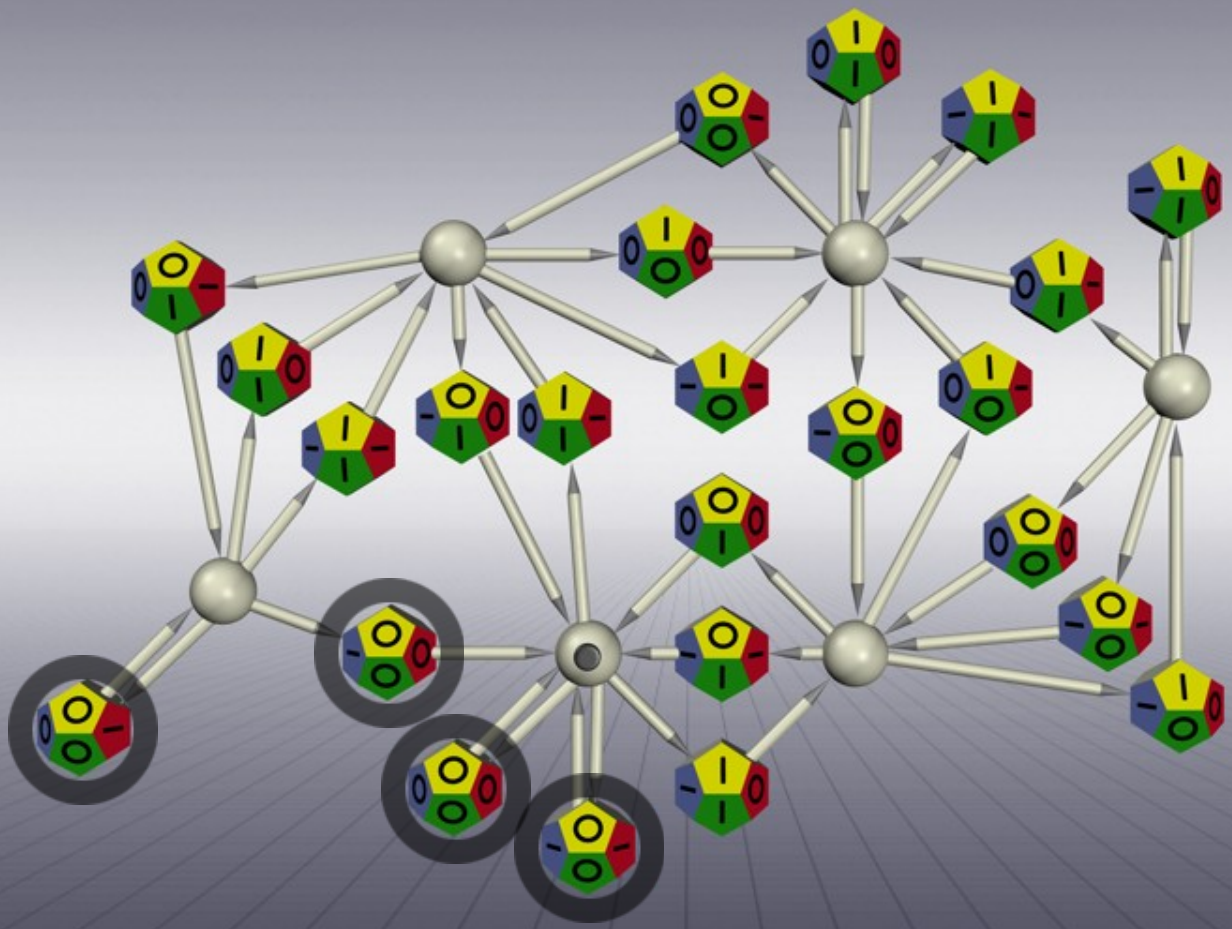
0 0 1 0 1 1 1 0 0

..0 1 0 0 0 1 0 1 0 0 276



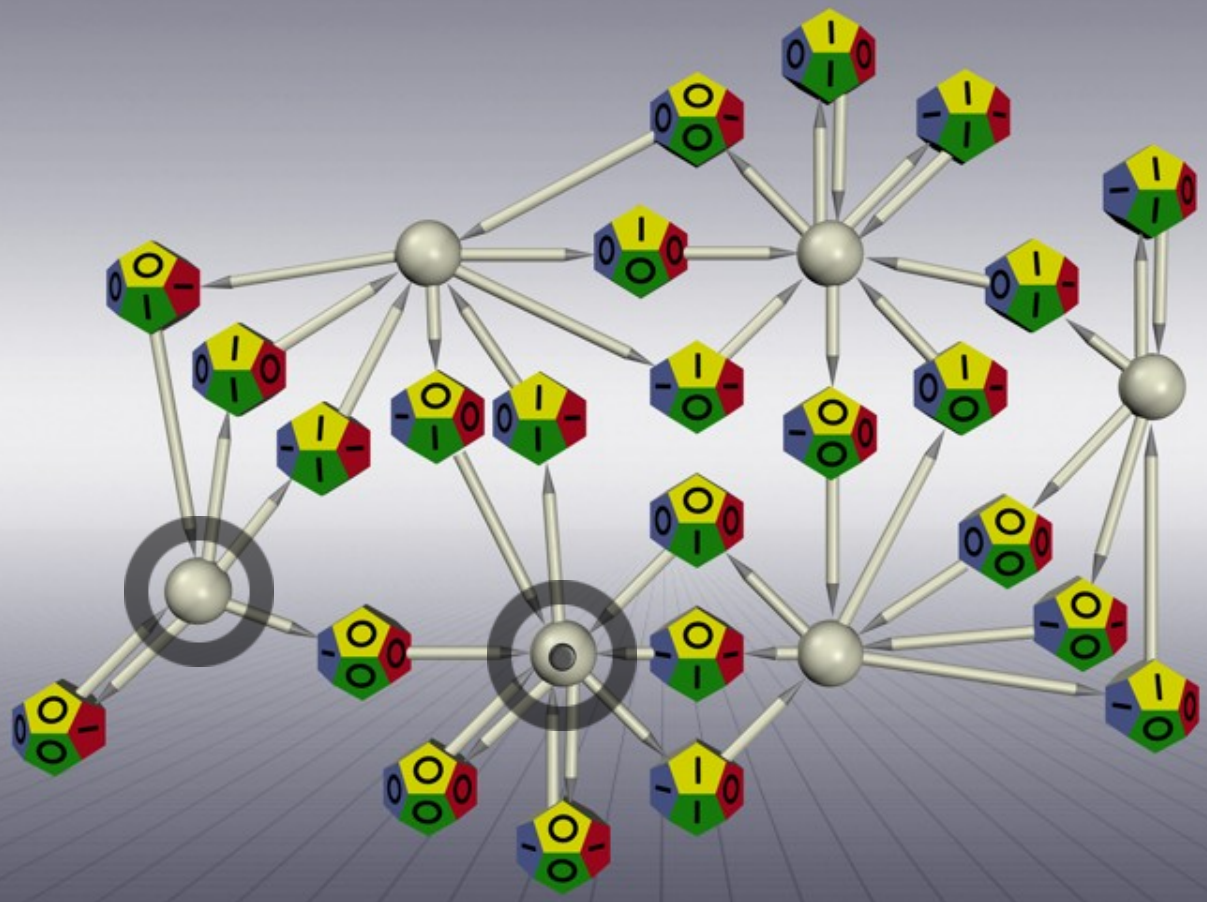
0 0 0 1 0 1 1 1 0 0

. . 0 1 0 0 0 1 0 1 0 0 276



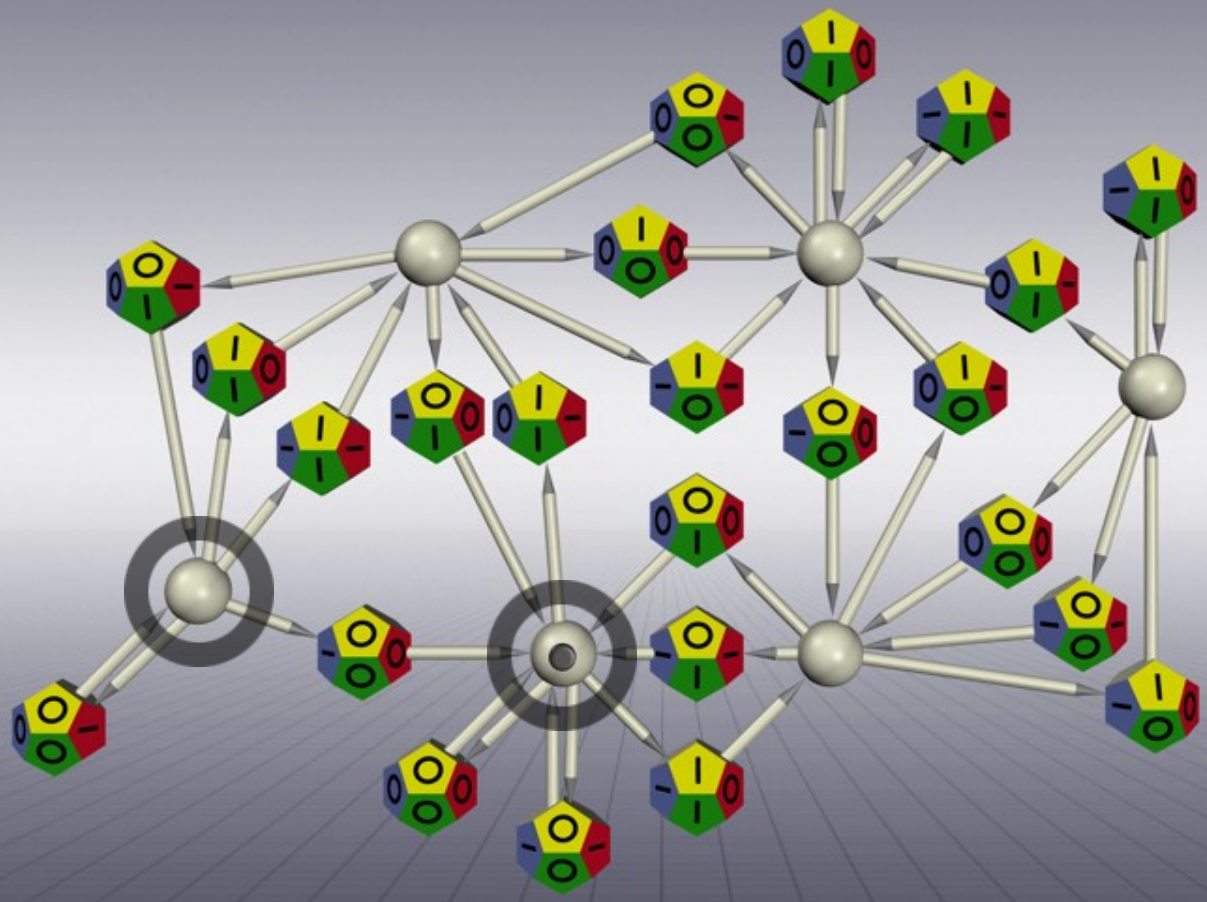
0001011100

..0100010100 276



..0001011100

..0100010100 276



= +

= −

= −

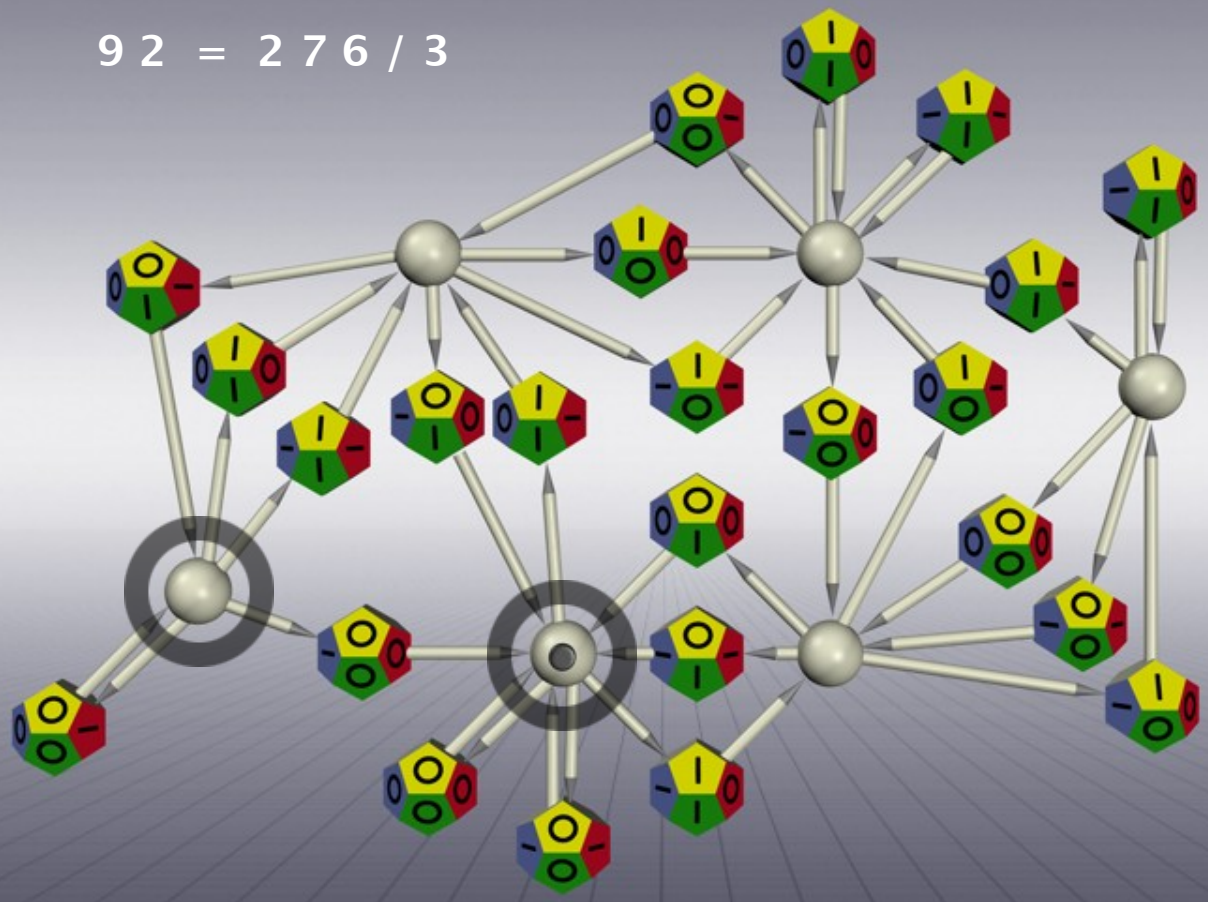
= 3·

= /3

..0001011100 92

..0100010100 276

92 = 276 / 3



Back