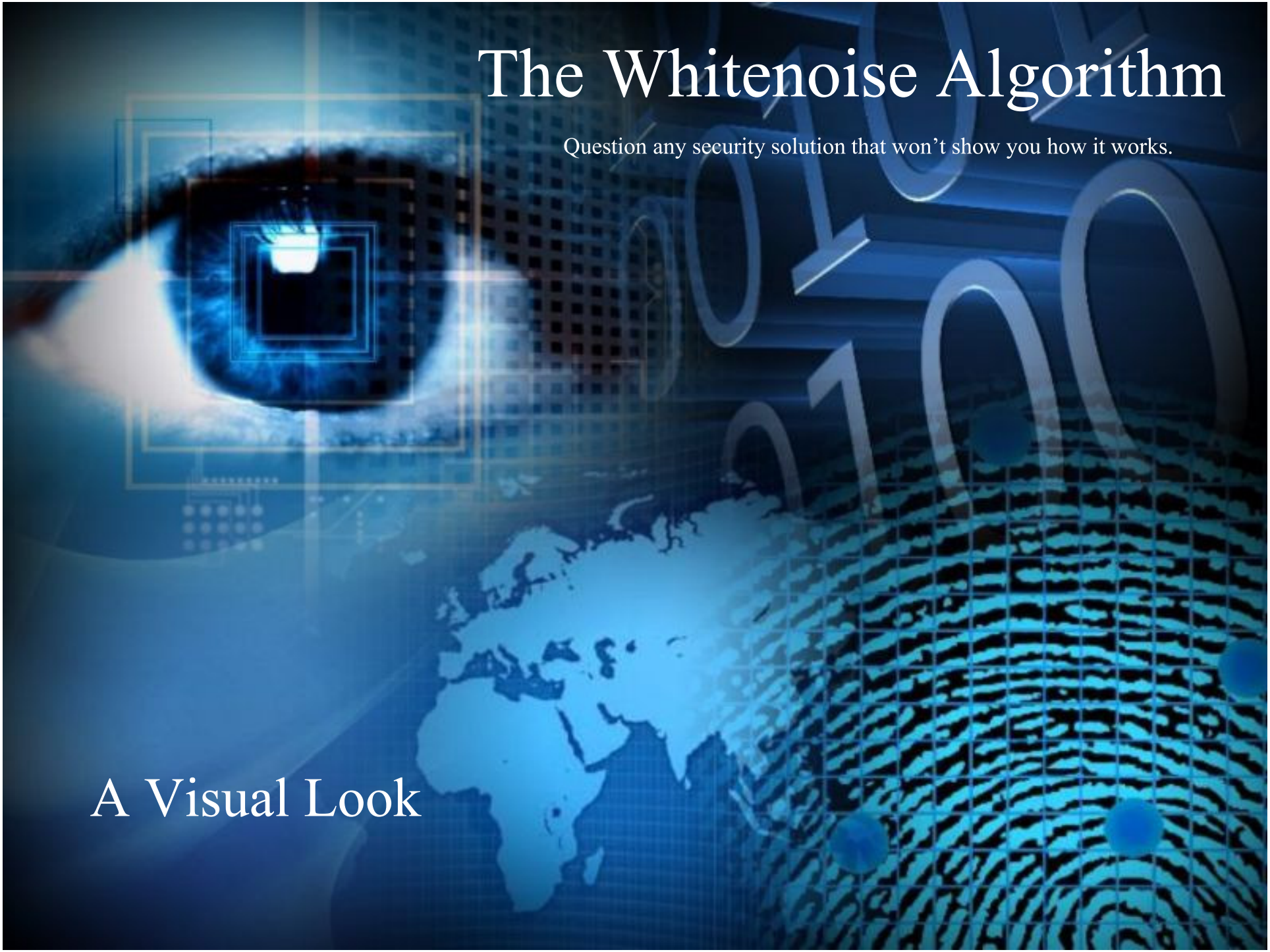
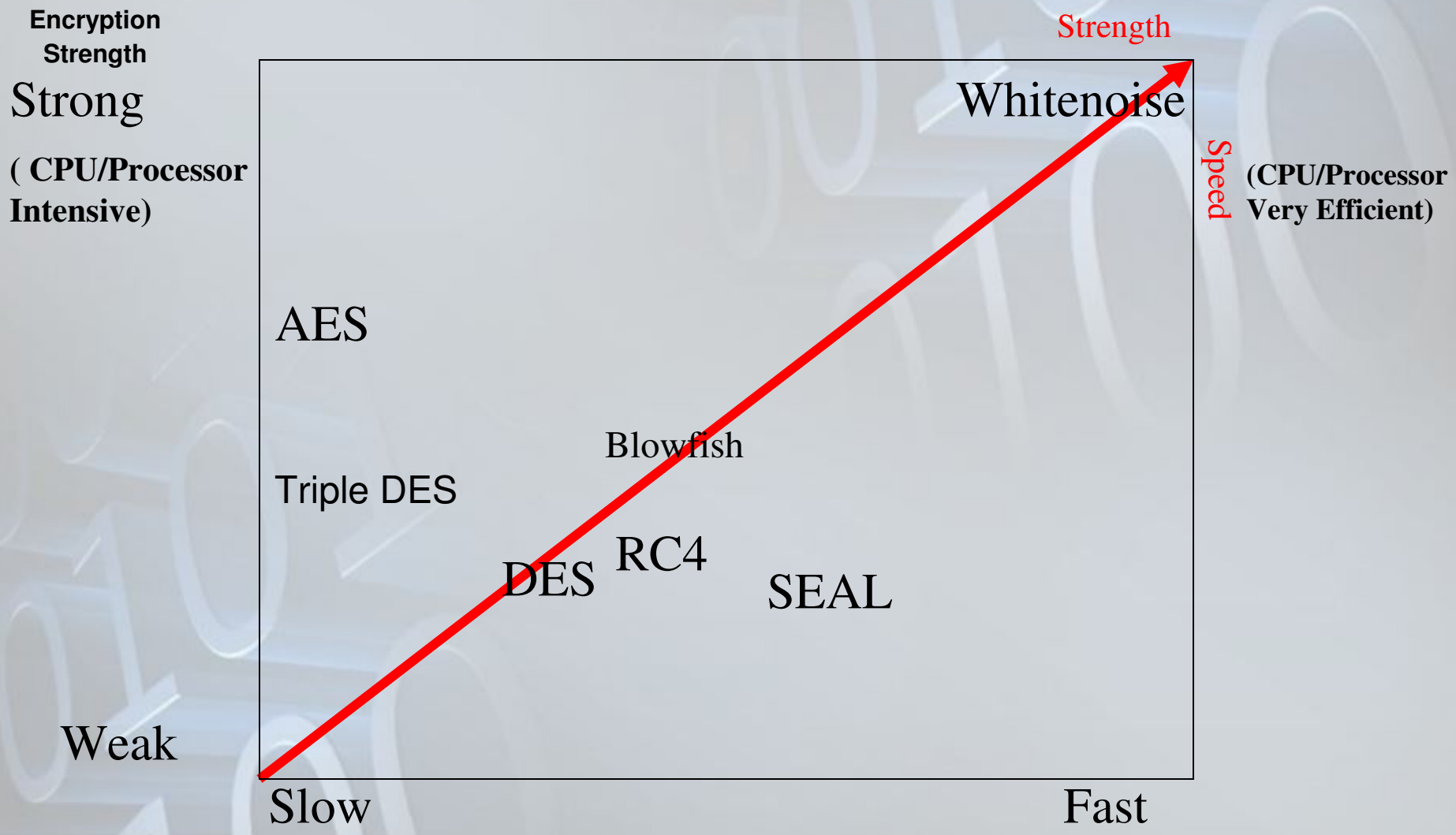

The Whitenoise Algorithm
Question any security solution that won't show you how it works.
A Visual Look

Whitenoise Algorithm Attributes

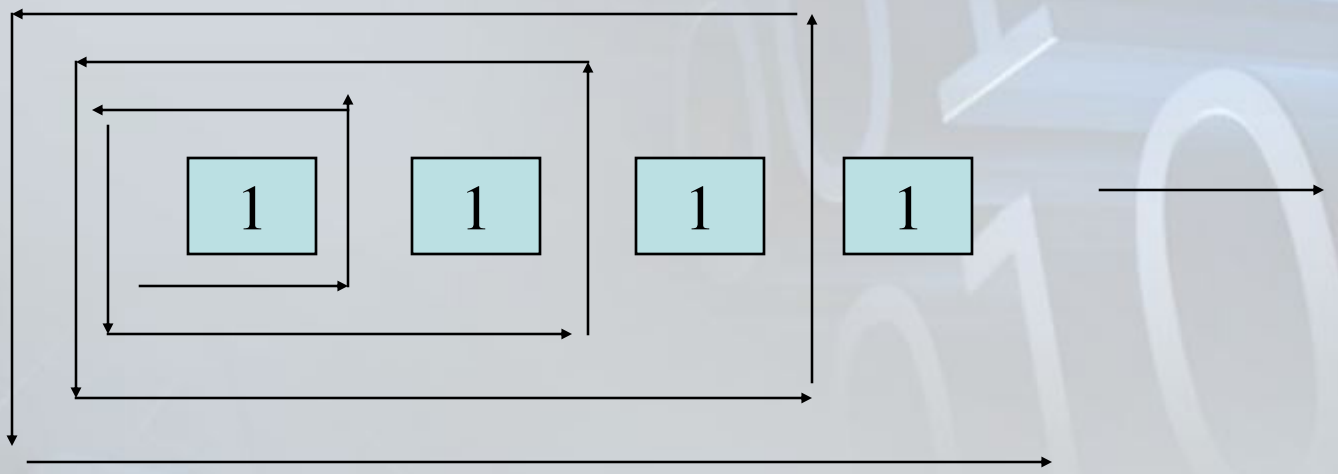
Keystream length exceeds the size of Data to be sent or stored; (Keys built from small amount of stored data); key segments are never re-used; Key stream Data never transmitted in an unsecured state

- ✓ Fast – 5 Clock Cycles per Byte (S/W) >2 Bytes / CC (H/W) –speed and strength are scalable
- ✓ Error Tolerant - bit and data independent
- ✓ Efficient - Low Processor Requirements – Lower cost devices
- ✓ Data Type Independent - Multimedia Support – Voice Data Video – Real Time streaming
- ✓ Manages Linear Offsets - Strong Identity & Digital Rights Management DIVA
- ✓ Receiver & Sender synchronized Keystream
- ✓ Scalable - Small Footprint ≤ 300k – Will run on 8 bit cpu

Whitenoise Algorithm Positioning

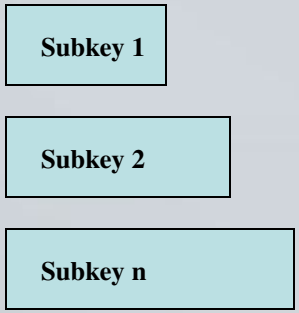


Traditional Ciphers



In traditional ciphers, processing is exponential, intensive [high overhead], and therefore slow. By analogy, bit one is encrypted, then you circle back and encrypt bit 1 and bit 2, then you circle back and encrypt bit 1 and 2 and 3 etc. Each bit encrypted is dependent on all the bits preceding it. If a bit is flipped the balance of the stream is corrupted.

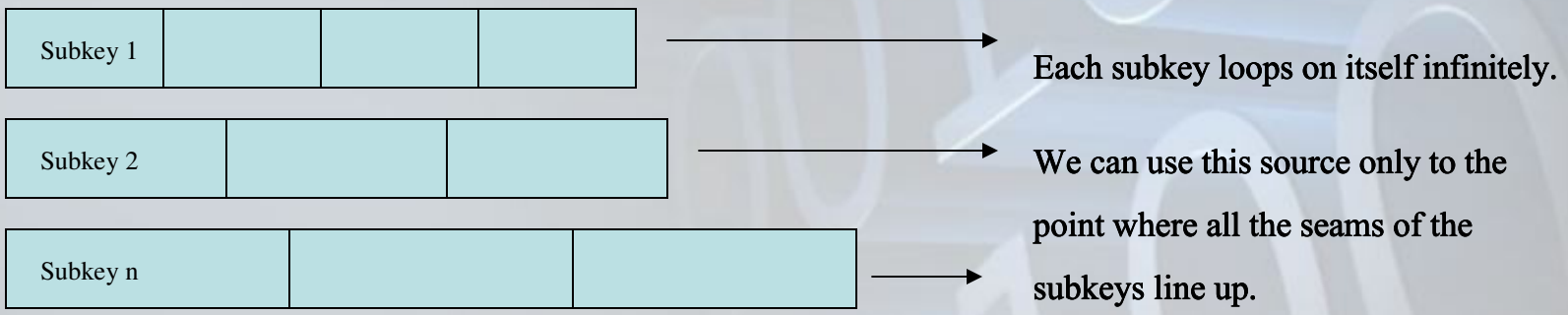
Creating a pseudo-random data source



Each subkey is a unique co-prime length. There can be a variable number of subkeys of variable lengths.

These subkey lengths give some of the internal key structure. This is very important because it is like the DNA of a WN Superkey. It is a very small amount of information that needs to be transmitted in order to create a WN key of extraordinary length. OTP's were difficult to pursue because one needed to transmit as much key information as the volume of data to be encrypted. A small packet of information can create an enormous key ie 20k of data can create a key hundreds of trillions of bytes long.

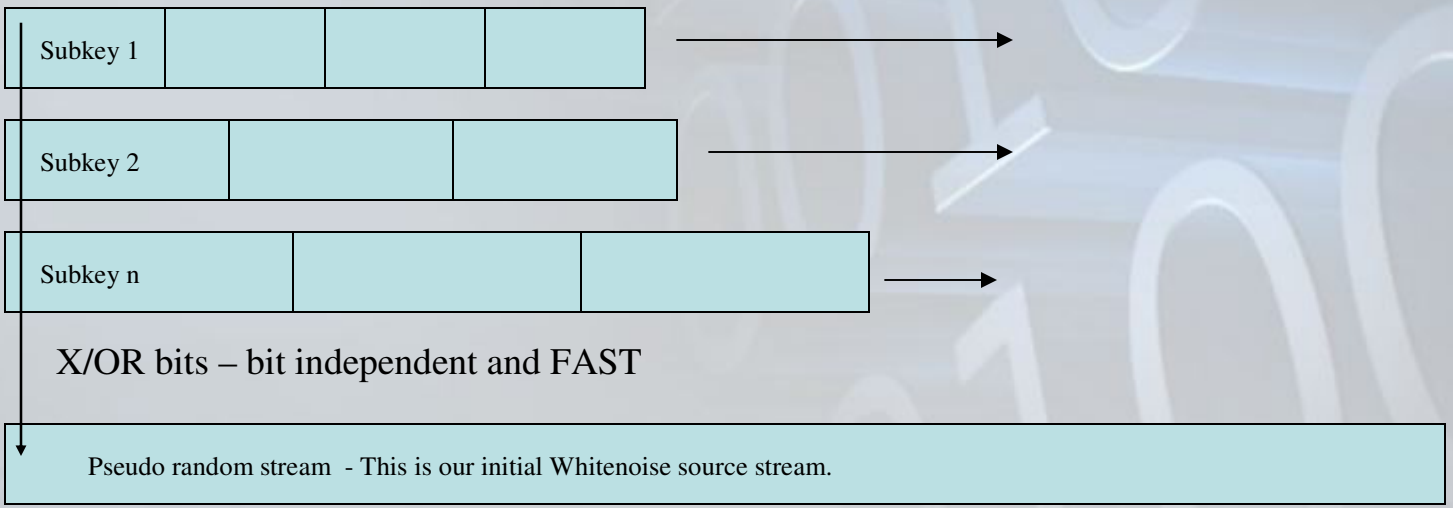
Creating an infinite array



Each subkey loops infinitely [horizontally] to create an endless source of data, or an infinite array.

Were the subkeys to operate only horizontally, then each subkey could be viewed as a Line Feed Shift Register. LFSRs have linearity characteristics that need to be addressed, because "untreated" they can be susceptible to algebraic attacks. However, this is well known and easily addressed. ATM's use LFSRs and they are delinearized to protect against attacks by using readily available Invertible Non-Linear Function (INLF) utilities. The WN key structure is NOT an LFSR as we will see.

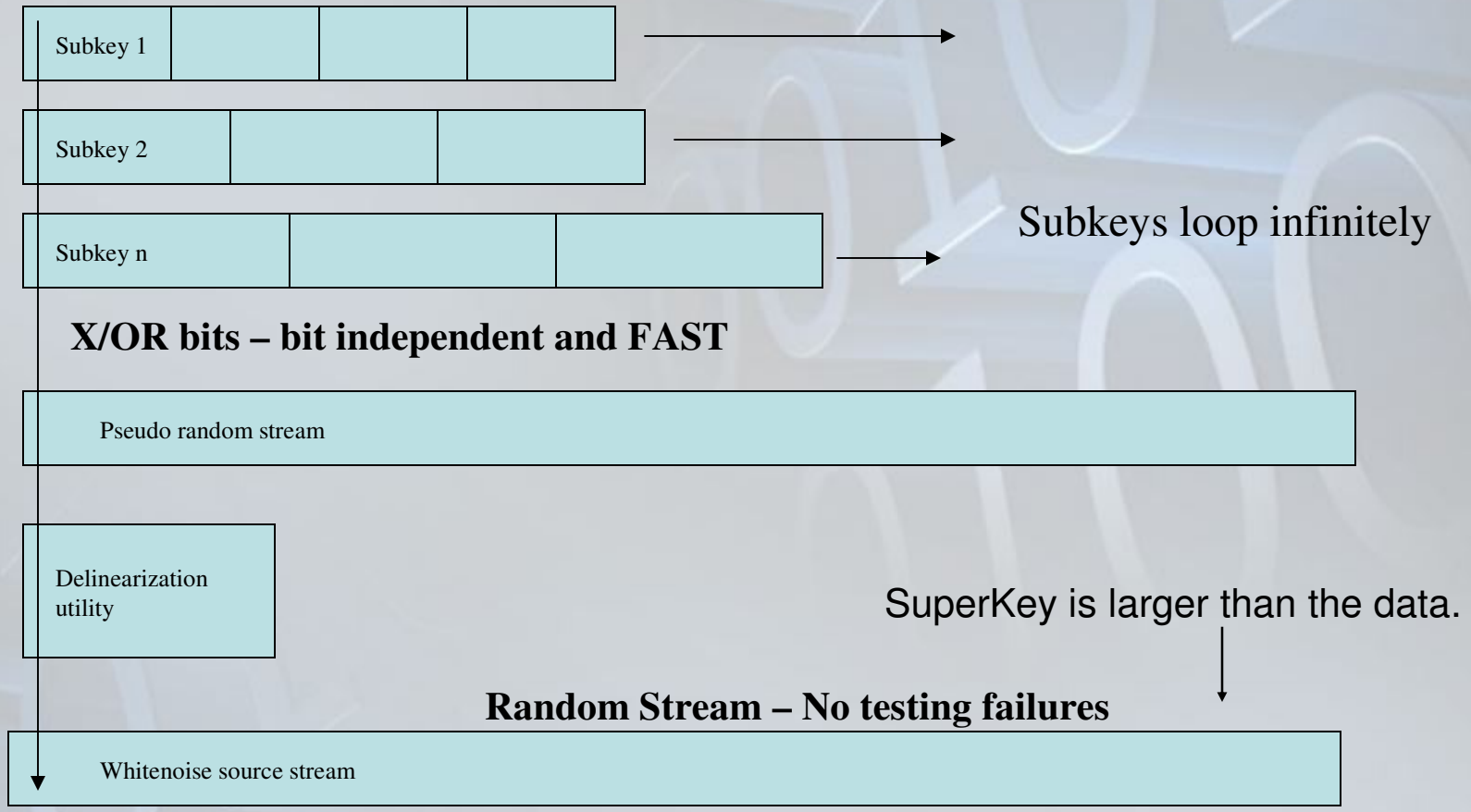
Creating the initial WN source stream – RNG



Whitenoise is NOT a Line Feed Shift Register. Each bit of each subkey is X OR'd with the corresponding bit of the next subkey in a vertical fashion. Because of this several things are occurring:

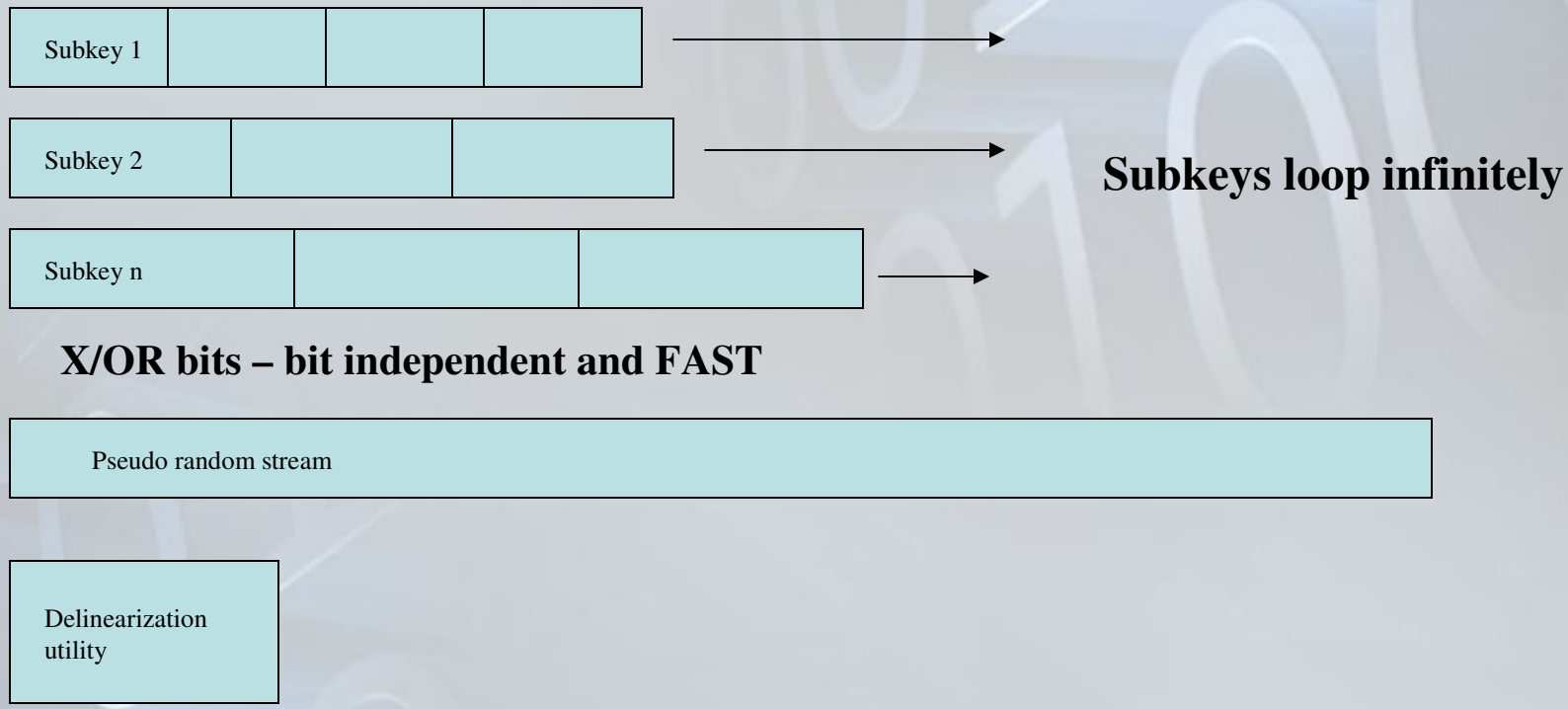
1. The structure is bit independent which means that any subsequent bits are not affected by the previous ones.
2. We are able to use the XOR function which is the fastest function available on a computer. The entropy, or randomness, increases as you add subkeys.
3. The resulting WN source stream is highly random, much better than most Random Number Generators, however, the resulting WN source stream still retains internal linearity characteristics that we want to remove.

The Whitenoise Cipher Stream



Each bit is X OR'd with the corresponding bit of the next subkey, increasing its entropy. This creates the first bit of our source stream which is then run through a Delinearization step and becomes the first bit of our Whitenoise Cipher Stream (cipher) which is completely delinearized and more random than any source recorded. There was not a single failure in randomness testing, not even anticipated statistical failures. No encryption algorithm has ever reached an acceptable level of randomness in a single round.

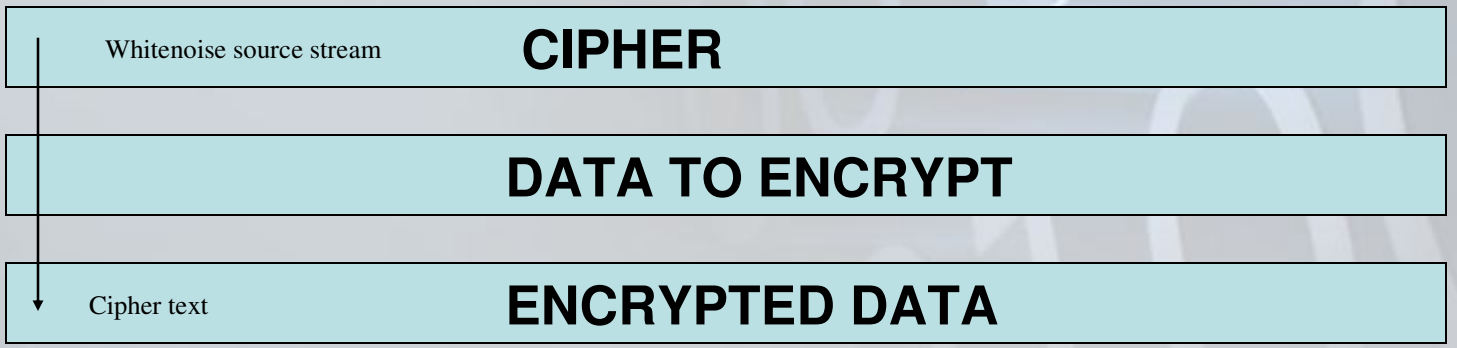
Removing any internal linearity characteristics from WN source stream



At this stage we are going to delinearize the WN source stream. This is done by using a many-to-one robust substitution array [$S_{65,536}$], using multiple byte draws etc.

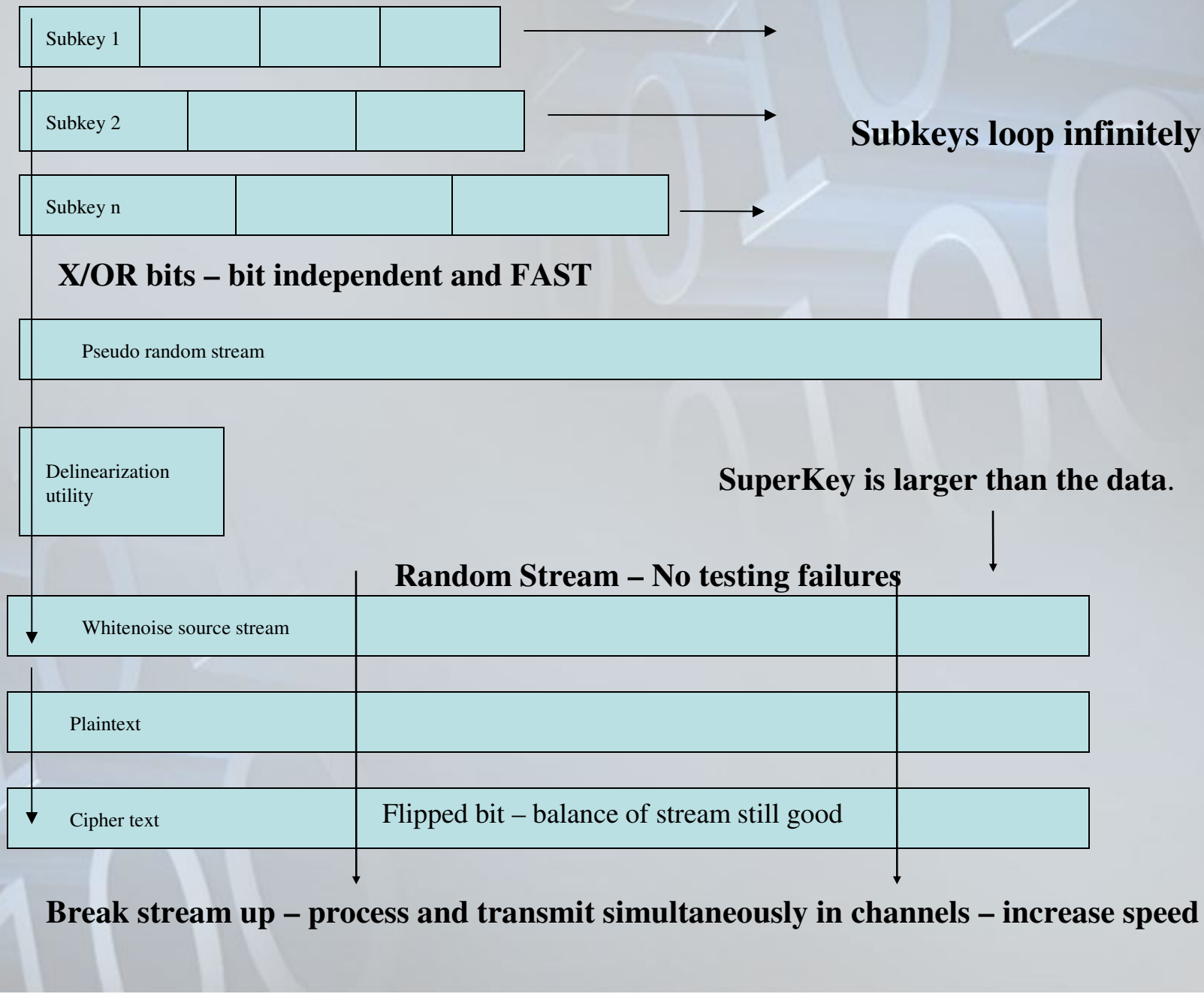
Encrypting Data

Random Stream – No testing failures



We can now safely encrypt data. Again moving in a vertical fashion, the first bit of the plaintext is X OR'd with the first bit of the cipher, starting at the cipher offset point, to create the first bit of encrypted data. This proceeds in this fashion until all the plaintext is encrypted and then stops (and the offset incremented for the next session). This structure is important as well because the Cipher Text is independent of the Plaintext and independent of the Cipher. This lends itself to very fast processing and very low overhead (no measurable latency for real time communications). This bit independence means that a "flipped" bit will not affect the balance of a transmission. It also means the the speed of WN is scalable to increase speeds with no end of speed increases in sight. This structure allows us to be able to break streams up into segments and process them and transmit them simultaneously in channels. This allows for caching and decryption and use in real time.

Whitenoise Architecture



A quick look at the multiplicity

3

5

7

11

13

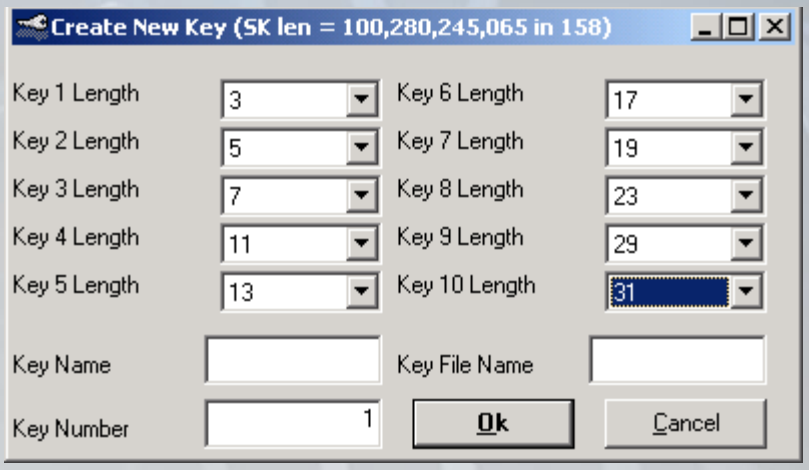
17

19

23

27

29

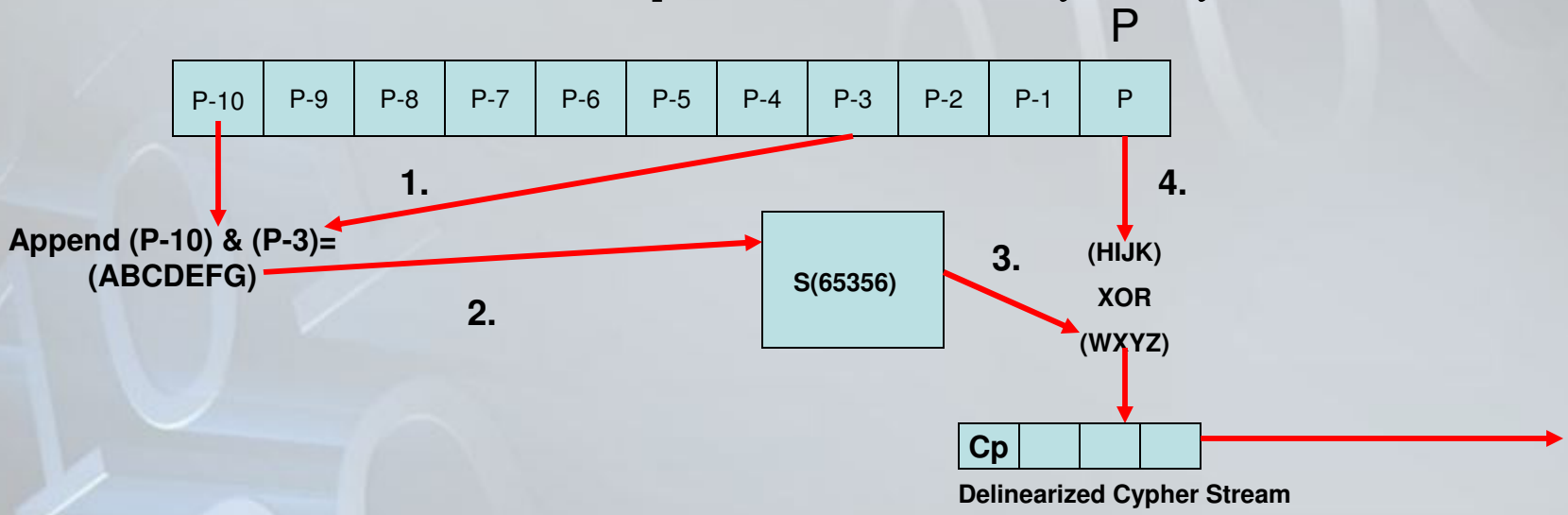


If we multiply the lengths of the subkeys, we see that using 10 subkeys and the smallest primes would result in a key 110,280,245,065 bytes long. We only need to transmit 158 bytes of internal key information (not including offsets) in order to recreate this key.

The bit strength of the cipher is calculated by adding the key stream byte lengths and multiplying by 8 bits per byte.

Whitenoise Delinearization

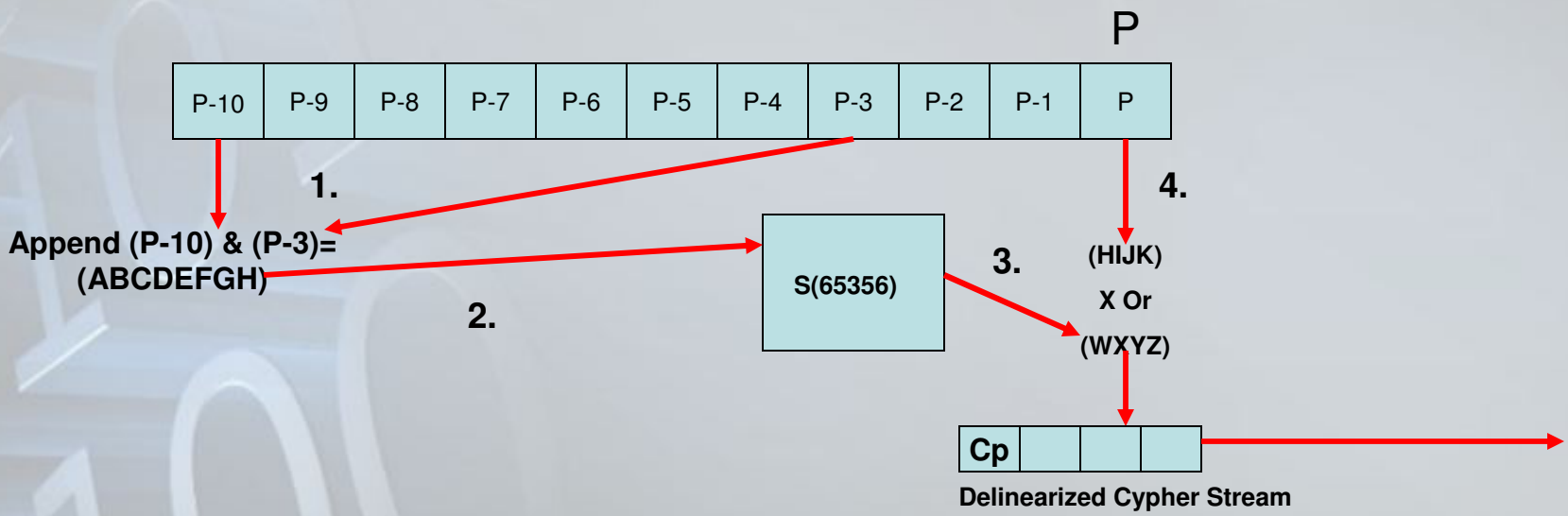
- ✓ Each box represents one byte
- ✓ P is the byte addressed by our offset point (the first byte in this example)
- ✓ The address P-3 is a co-prime distance of three bytes away from P.
- ✓ The address P-10 is an additional co-prime distance of seven bytes away from P-3

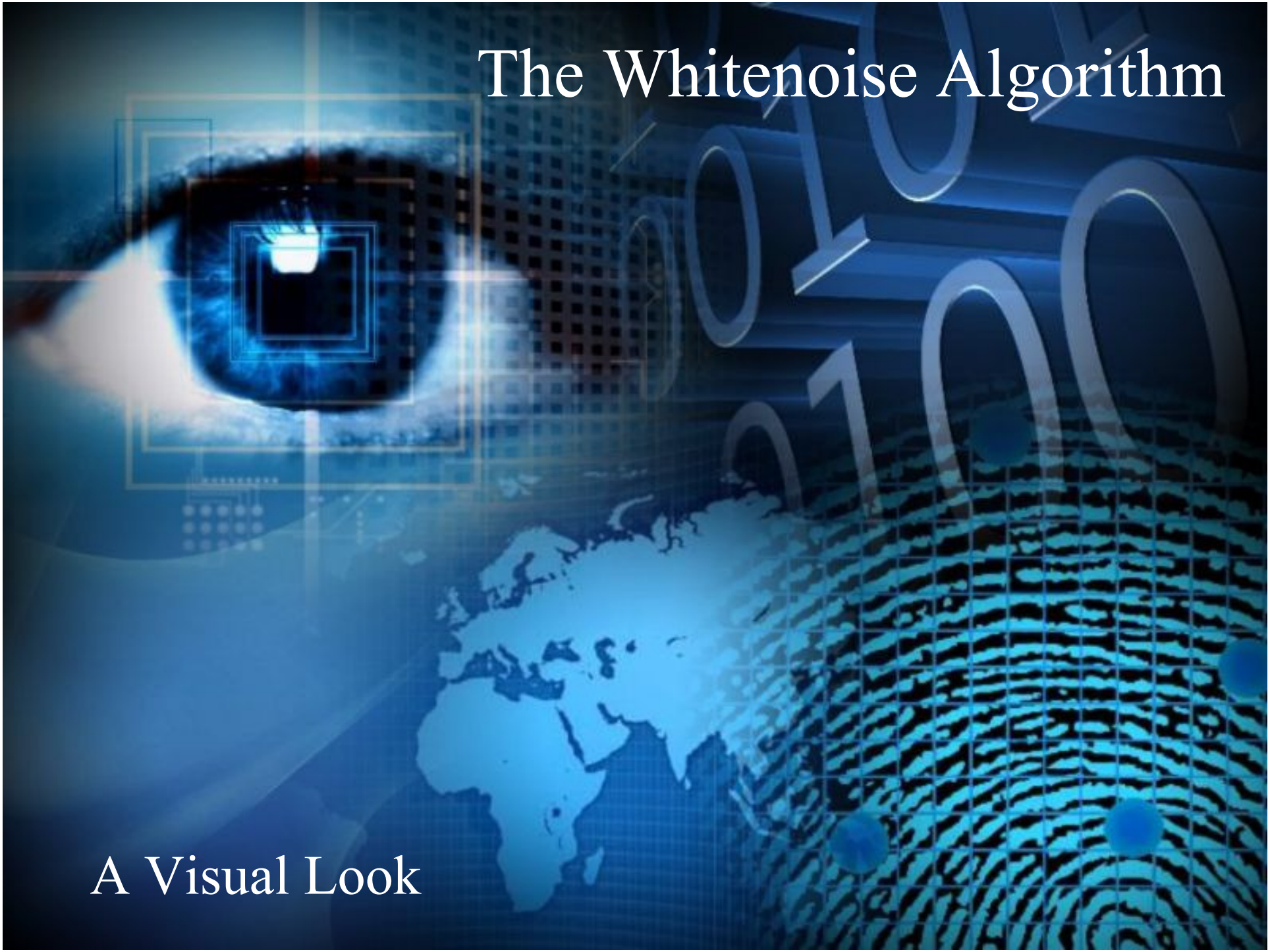


Note: The author of http://eprint.iacr.org/2003/250 has been recognized on page 8 of the revision http://eprint.iacr.org/2003/249.

Whitenoise Delinearization (Cont'd)

1. Reaching backwards on the pseudo-random stream, grab the addresses for the 2 bytes represented by P-10 and P-3. Note that both these values are co-prime distances away from P. The value for P-10 is ABCD and the value for P-3 is EFGH. Append the 2 byte addresses together, with P-10 being the higher order bit values, to create a 16 bit value ABCDEFGH.
2. Push this value into the S(65356).
3. One byte emerges the S(65356). Note: 2 bytes enter-1byte exits-one way function
4. XOR the emerged byte with the bit value of the current byte P.
5. This becomes the first completely delinearized byte of our cipher stream.




The Whitenoise Algorithm
A Visual Look