Decrypting Encryption in HDL Design and Verification

ALDEC Webinar

Created and Presented by
Jerry Kaczynski,
ALDEC Research Engineer

Rev. 1.2

Agenda

- Symmetric Ciphers
- Asymmetric Ciphers
- Practical Implementations
- Interoperable IP Delivery Solution
- Reference: Explanation of Cryptology Terms

Simple and Reliable:

SYMMETRIC CIPHERS

Symmetric Ciphers

- All ciphers in use until late 20th century have one thing in common: a ***secret key*** – number or phrase – that must be known to both ***sender and recipient*** of the message.
- Since both parties have to keep the key secret, those ciphers are known as ***symmetric ciphers*** or ***secret key ciphers***.

We use single frame around the key to signify that it must be secret

Block Ciphers

- One class of modern symmetric ciphers performs encryption on fixed-length chunks of data: we call them ***block ciphers***.
- Originally 64 bit (8 characters) block size was used, now 128 bit (16 characters) blocks are more popular.
- Plaintext is divided into the block-size chunks before encryption; last chunk is ***padded*** to full block size if needed.
- Each chunk is encrypted the same way (with the same key) by identical encryption units.
- Internal operations of encryption unit consist of several ***rounds*** of substitutions, transpositions and logical operations.
- Each round gets its own key derived from the secret key using ***key schedule*** algorithm.
- Outputs of encryption units are merged into ciphertext.

Popular Block Ciphers

- ***DES*** (***D***ata ***E***ncryption ***S***tandard) was announced in 1976 as a national standard in the USA and quickly gained worldwide popularity.
  - DES uses 64 bit block and 56 bit keys.
  - DES was broken in 22 hours in 1999, so it is no longer considered secure in critical applications.

- ***AES*** (***A***dvanced ***E***ncryption ***S***tandard), a DES successor, was announced in 2001 as a winner of 5 year long contest.
  - AES implements 128 bit block length.
  - Uses 3 strengths of keys: 128 bit, 192 bit and 256 bit.
  - All versions of AES are safe now, although 128 bit version may be broken in the nearest future.

- Other block ciphers worth mentioning: 3DES, IDEA, Blowfish.

Breaking Block Ciphers

- The most basic form of breaking any cipher if ***brute force attack***: testing all possible key values until the message is decrypted.
- For a cipher with *n*-bit key, brute force attack requires 2^n operations. It means that to break DES you must test $2^{56}=7.2\cdot10^{16}$ keys and breaking the strongest version of AES requires $2^{256}=1.16\cdot10^{77}$ operations.

- If you consider that one year has 31536000 seconds, testing 100 keys per second would require over 22 million years to break DES.
- Currently available RIVYERA **parallel** computer with 128 Spartan-6 chips can break DES in one day.
- That explains why DES is considered insecure, while AES looks pretty safe…

Advantages of Symmetric Ciphers

- Modern symmetric ciphers (block and stream) are frequently used due to several advantages:
  - Security (if you are using latest/updated versions).
  - Fast operation.
  - Easy implementation in software (e.g. *OpenSSL* library).
  - Efficient implementation in hardware (some may be patented, though).

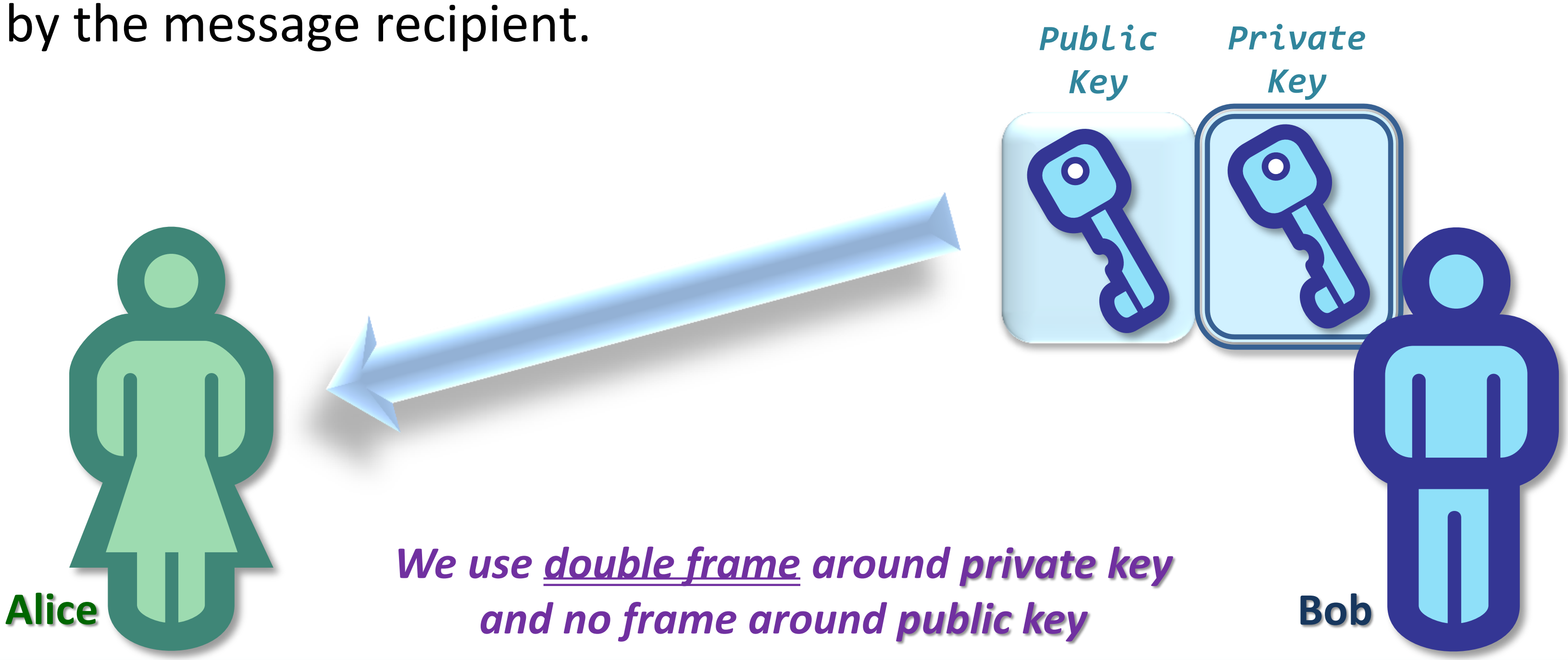
- There is one serious disadvantage of symmetric ciphers:
  - Managing *secret keys*.
    (Getting secret keys to all involved parties and keeping them secure from unauthorized access is a critical part of symmetric cipher cryptosystem.)

Very Sophisticated:

ASYMMETRIC CIPHERS

Asymmetric Ciphers

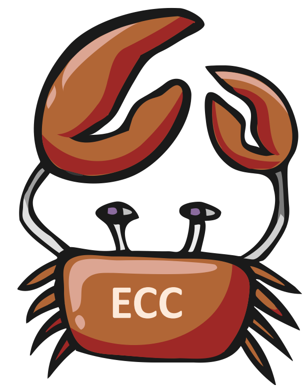
- New idea appeared in the 20th century: an ***asymmetric cipher*** with a ***public key*** (available to anybody) that allows message encryption, but not decryption.
- To decrypt a message, matching ***private key*** is needed; it is used only by the message recipient.



*We use <u>double frame</u> around **private key** and no frame around **public key***

Popular Asymmetric Ciphers

- Most popular asymmetric ciphers (a.k.a. public key algorithms) are based on the concept of a ***one-way function***, i.e. mathematical operation that cannot be (easily) inverted.



- ***ElGamal*** encryption algorithm is based on easy computation of raising to power and difficult computation of discrete logarithm over cyclic groups.
- ***RSA*** cipher (described in the following slides) utilizes easy multiplication and difficult factorization of large numbers.
- ***ECC*** (***E***lliptic ***C***urve ***C***ryptography) algorithms are based on difficulty of inverting elliptic curve point multiplication.

RSA Cipher

- ***RSA****** is the most popular asymmetric cipher based on the following principles:
  - Finding two large, random prime numbers ***p*** and ***q*** and computing ***n=pq*** is easy.
  - Factorizing ***n*** is extremely expensive operation, so even if you reveal ***n***, recovering ***p*** and ***q*** is not feasible.
  - Selecting two more numbers ***d*** and ***e*** (related to ***p*** and ***q***) lets you create simple function for message encryption: m^e ***mod*** n and ciphertext decryption: c^d ***mod*** n.
  - ***Public key*** contains **modulus** ***n*** and **public exponent** ***e***.
  - ***Private key*** contains **modulus** ***n*** and **private exponent** ***d***.

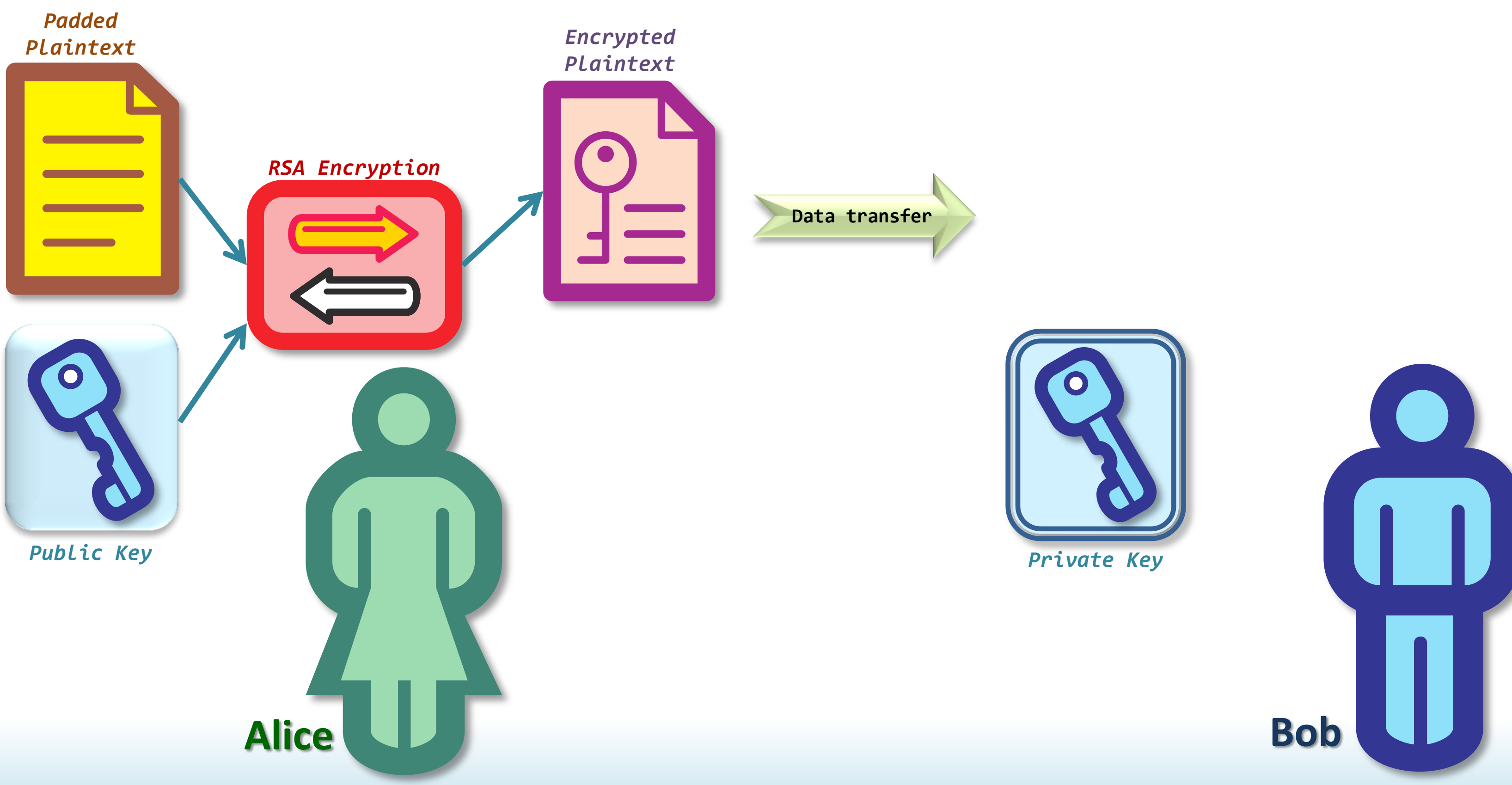
$$c = m^e \ mod \ n$$

- RSA was developed independently in the 1973-76 period by the UK intelligence agency and MIT* teams.
- RSA with keys shorter than 1024 bits is no longer considered secure; 2048 bit keys are recommended for long-term applications.

*RSA stands for **R**ivest, **S**hamir and **A**dleman – its MIT inventors.

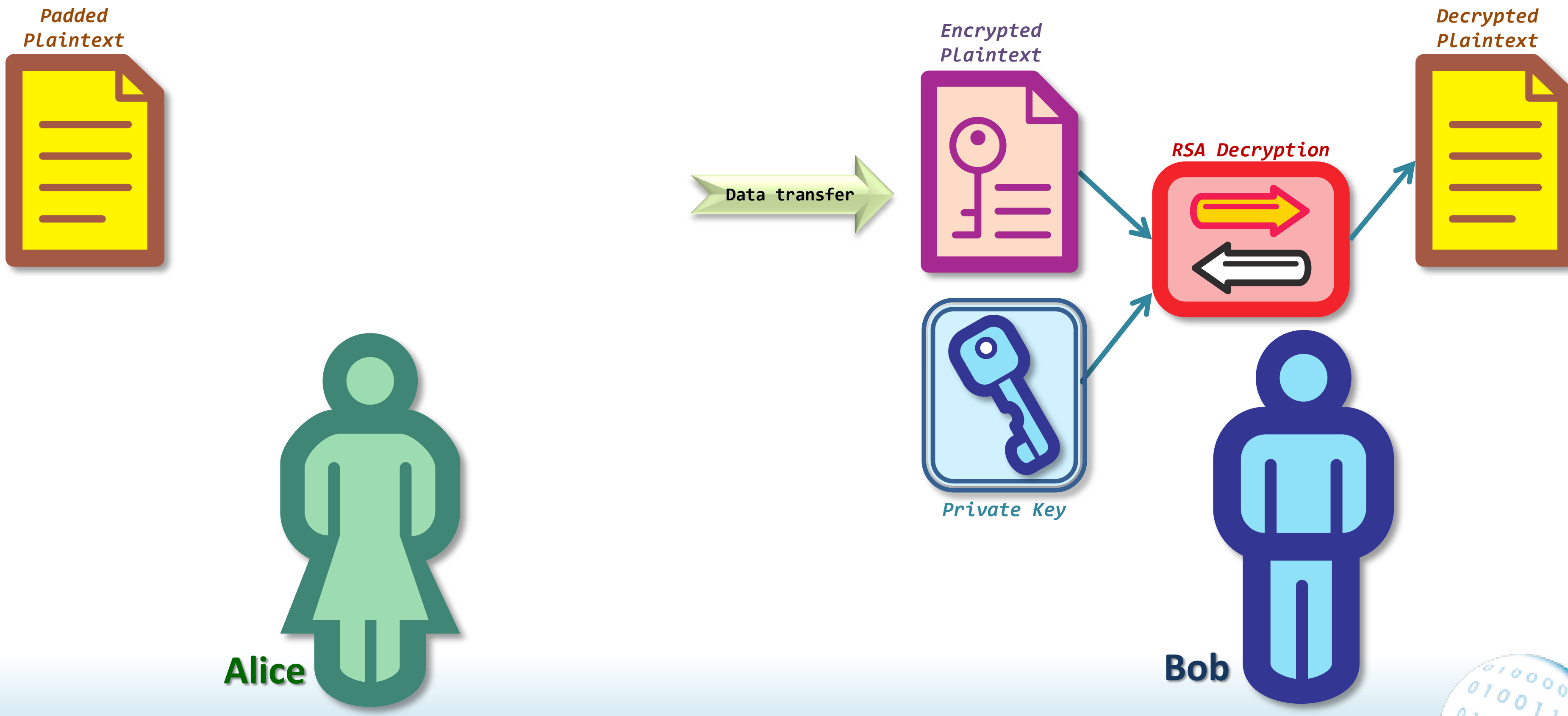
Sample RSA Cryptosystem

- In typical RSA cryptosystem sender grabs up-to-date public key of the recipient, encrypts and sends the message. Recipient uses private key to decrypt the message.



Sample RSA Cryptosystem

- In typical RSA cryptosystem sender grabs up-to-date public key of the recipient, encrypts and sends the message. Recipient uses private key to decrypt the message.



Advantages of Asymmetric Ciphers

- The main advantages of asymmetric ciphers include:
  - Easy key distribution.
  - High security.
  - Versatility.

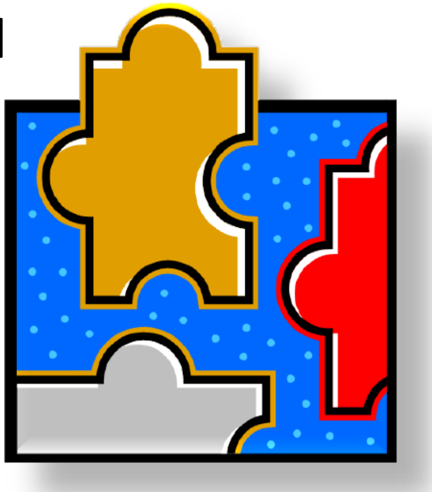
- The key disadvantage of asymmetric ciphers is:
  - Complex/slow operation.
    - In practical applications, only sending messages shorter than key length makes sense.
    - Proper padding of the message is required to maintain high security.

Where to Use Them:

PRACTICAL IMPLEMENTATIONS

Old Style: Source Obfuscation

- ***Source obfuscation*** is the method of replacing meaningful names in the IP source with something hard to read and intentionally confusing.
- Source obfuscation is an example of "*security through obscurity*", i.e. not really secure method of IP delivery.
- Source obfuscation was acceptable for end users and tool vendors, but not good for IP creators.
- Related ***tool-specific source encryption*** methods were more secure than obfuscation, but difficult to manage.



Old Style: Binary Delivery

- Some EDA tools have native, ***binary library formats***, usually enhanced with ***protection*** mechanisms that prevent debugging.
- IP can be delivered in ***tool-specific binary format***, but the only group of people happy with this solution are users of that specific tool. Popular IP vendor cannot be happy, because multiple tools must be supported…
- There were some ***'universal' binary formats***, but they created nightmares on the end-user side. FPGA users remember one solution from this group: after configuring model library for vendor "A" it stopped working in vendor "X" tools and reconfiguring it back to vendor "X" support disabled it in vendor "A" tools…

New Style: Cryptosystems

- Cryptosystem suitable for source encryption can be based on symmetric ciphers, asymmetric ciphers or be a hybrid of the two.
- All modern ciphers are secure if keys are long enough.
- ***Symmetric cipher cryptosystems*** are fast and easy to implement, but their key management is very difficult.
- ***Asymmetric cipher cryptosystems*** have easy key management, but speed that is too slow for reasonable length sources.
- ***Hybrid cryptosystems*** seem to inherit only the advantages of pure symmetric/asymmetric ones…

Cryptosystem:	Symmetric cipher	Asymmetric cipher	Hybrid
Speed:	High	Low	High*
Security:	High	High	High
Key Management:	Difficult	Easy	Easy
Flexibility:	Low	Low	High

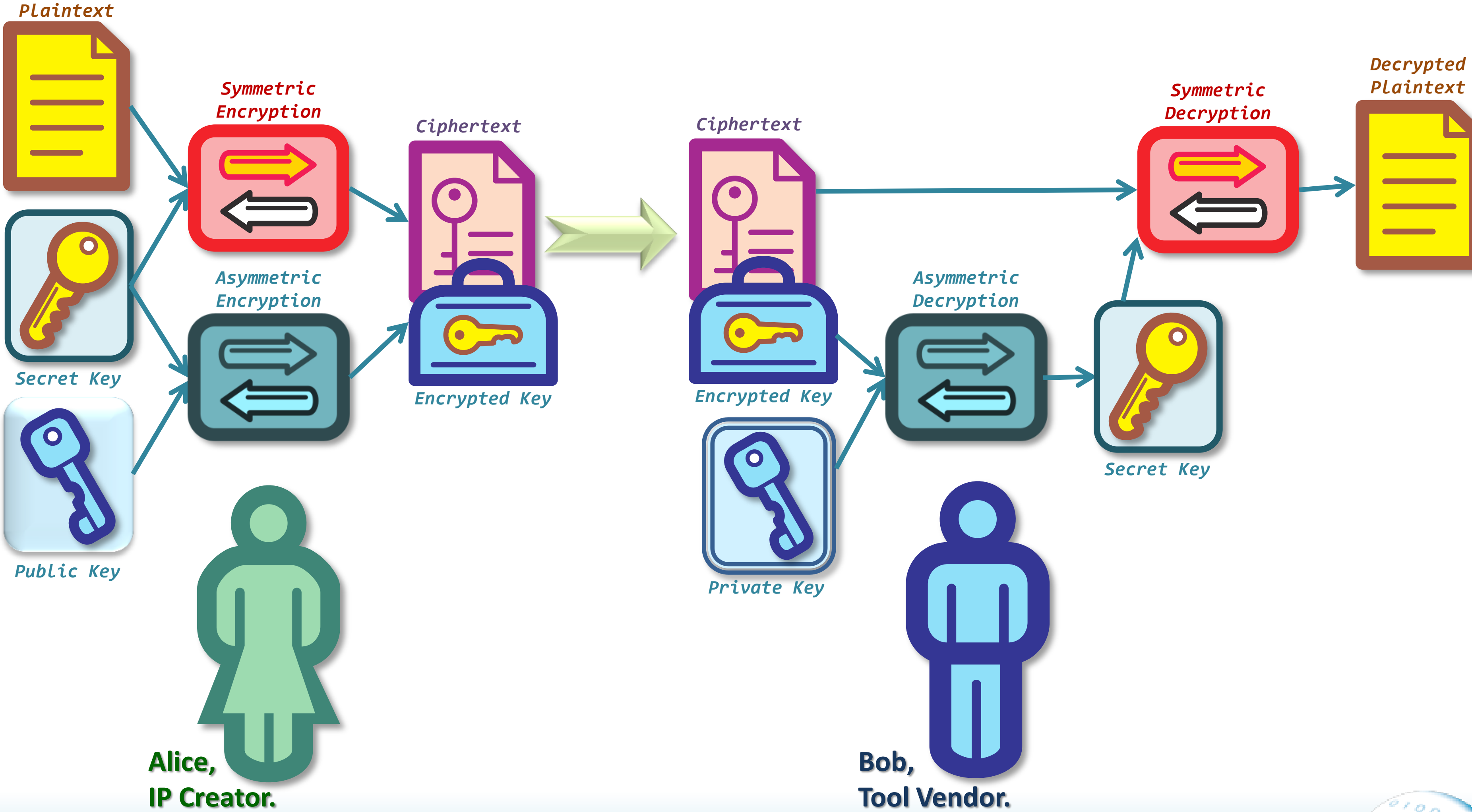
* assuming message length significantly larger than key length.

How Hybrid Cryptosystem Works

- Hybrid cryptosystem is the best fit for source encryption.
- If ***Alice*** (IP creator) wants to send a message (IP) to ***Bob*** (Tool vendor):
  - Alice ***encrypts*** big ***message*** using ***symmetric cipher*** with ***random secret key*** (a.k.a. ***session key***).
  - Alice ***encrypts session key*** using ***asymmetric cipher*** and Bob's ***public key***.
  - Alice sends encrypted data and encrypted key to Bob
  - Bob ***decrypts session key*** using his private key.
  - Bob ***decrypts message*** using recovered secret key and discards the key.

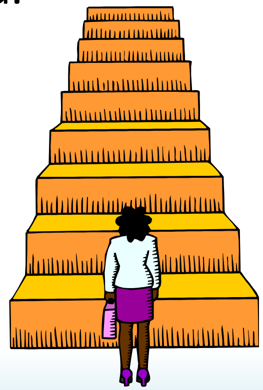
- Hybrid cryptosystem tries to have the best of both worlds:
  - Speed and security of established symmetric ciphers.
  - Easy key handling and security of asymmetric ciphers.

Simple Hybrid Cryptosystem



Challenges Facing Implementers

- IP creators and tool vendors have to agree on the reasonably small set of supported ciphers:
  - Symmetric – 3DES and AES for new applications, DES for legacy.
  - Asymmetric – RSA now; Elliptic Curve for future applications.
- HDL governing bodies have to agree on the way of embedding encrypted data in the code so that HDL compilers can process it.
- Encrypted data must be labeled properly:
  - Tools must be able to identify session key encrypted with their own public key.
  - Both symmetric and asymmetric algorithm used must be identified.
  - Author/owner of the encrypted IP should be easy to identify.



Benefits

- Once properly standardized and implemented, source encryption can be highly beneficial method of IP delivery:
  - One method working with all tools in a chain.
  - One encrypted source to maintain.
  - Any encryption tool can be used to encrypt for any vendor tool.
  - Secure encryption algorithms.
  - Low implementation cost: libraries of encryption and decryption procedures publicly available.

Encryption in HDL Sources:

INTEROPERABLE IP DELIVERY SOLUTION

History

- All currently used source encryption systems for HDLs originate from the same donation by Cadence.
- Verilog was the first HDL standard to get source encryption (Verilog-2005).
- VHDL added some much needed explanations (VHDL-2008).
- SystemVerilog clarified some issues present in other language implementations (SystemVerilog-2009).

Unfortunately, all those HDL implementations have some problems…

Common Features

- All HDLs use compatible set of ***encryption commands***.
- Encryption commands with encrypted data are placed directly in the HDL code, creating ***digital envelopes***.
- Encryption commands are recognized thanks to standardized prefix:

Language:	VHDL	Verilog/SystemVerilog
Prefix:	`` `protect ``	`` `pragma protect ``

- Encrypted data (HDL code and symmetric key) is encoded using Base64 to avoid accidental modifications in text editors.

HDL Code Before Encryption

```
--*************************************************************
--* Intellectual Property ©2011 ACME, Inc.                    *
--* VHDL Package: my_pack                                     *
--*************************************************************
package my_pack is
    function magic (arg : integer) return integer;
end package my_pack;
```

```
`protect begin
package body my_pack is
    function magic (arg : integer) return integer is
    begin
        report "Magic function was called!";
        return arg * arg + 777;
    end function magic;
end package body my_pack;
`protect end
```

Command marking start of code to be encrypted

Plaintext of HDL Code To Be Encrypted

Command marking end of code to be encrypted

HDL Code After Encryption

```
--*****************************************************
--* Intellectual Property ©2011 ACME, Inc.           *
--* VHDL Package: my_pack                             *
--*****************************************************
package my_pack is
      function magic (arg : integer) return integer;
end package my_pack;

`protect begin_protected
`protect version = 1
`protect author= "john.smith@acme.com"
`protect encrypt_agent= "Aldec protectip.pl, rev. 164098"
`protect key_keyowner= "Aldec", key_keyname= "ALDEC08_001", key_method= "rsa"
`protect encoding= (enctype="base64")
`protect key_block
CBbtjb/TnL8uQsfiqMNPicHtioL2Kxzi0mf+iRQn4Af1P9cO1HAEpDrSU7J2DdMItSXg9P/bHZ4A
2Vjx+4VyoEyboG+NeXaE7UV7djweCJ0bOPVASagQsuLAWWlj6UiqUIOVa3C1fLkr+0quEzKhcNVn
mbvs92U0oKCuyC4JwlQXwuKyl81x1pJxDySMrs74oFqNeoLt4ZHdKiy6yeNn9OiP0QMXzGR3wpe5
CypVsLF15SibeocemLNO7fUINw0ZgmaoirkVS/kqfo++3nIlWKvhzytWxxKurGzo2RQOZuApYO/q
4NN9wW8MY0ohT1YbNJivApWD/fMdYQ7VRgr/3g==
`protect data_keyowner= "ACME_Inc.", data_keyname= "Random"
`protect data_method= "aes256-cbc"
`protect encoding= (enctype="base64")
`protect data_block
pyB+WqJ2KgutN0E39HXls+8GUPF+YedbmZ1CV+u8bz5P8zlj+2FPzWXb9vswNElksxAojMA50VHx
tAN21ebA0gI9tqt4iC7UQkM2epiuvMEm6APC9a7q9XD8LeLqWILv1cqF57PFIiBGvj/EdpC8xwl6
iKV7QP+/Lmd0AV7BNtVEGpr+LpsAcVb/2f11KjU2xkOE3UgI9tki7RTUJ3sD9+OlL4yxljU5Vl9E
BoxCtbegynYyt2yeMsSEE4QtnEbzSgeAZDgr9hGx3QayYEdL2tLwDGV7Ce4BWB1D1Tk7KL4=
`protect end_protected
```

Public Key Commands

Encrypted, Encoded Session Key

Data Encryption Commands

Encrypted, Encoded HDL Code

Digital Envelopes?

```
--*************************************************************
--* Intellectual Property ©2011 ACME, Inc.                     *
--* VHDL Package: my_pack                                      *
--*************************************************************
package my_pack is
      function magic (arg : integer) return integer;
end package my_pack;

`protect begin_protected
`protect version = 1
`protect author= "john.smith@acme.com"
`protect encrypt_agent= "Aldec protectip.pl, rev. 164098"
`protect key_keyowner= "Aldec", key_keyname= "ALDEC08_001", key_method= "rsa"
`protect encoding= (enctype="base64")
`protect key_block
CBbtjb/TnL8uQsfiqMNPicHtioL2Kxzi0mf+iRQn4Af1P9cO1HAEpDrSU7J2DdMItSXg9P/bHZ4A
2Vjx+4VyoEyboG+NeXaE7UV7djweCJ0bOPVASagQsuLAWWlj6UiqUIOVa3C1fLkr+0quEzKhcNVn
mbvs92U0oKCuyC4JwlQXwuKyl81x1pJxDySMrs74oFqNeoLt4ZHdKiy6yeNn9OiP0QMXzGR3wpe5
CypVsLF15SibeocemLNO7fUINw0ZgmaoirkVS/kqfo++3nIlWKvhzytWxxKurGzo2RQOZuApYO/q
4NN9wW8MY0ohT1YbNJivApWD/fMdYQ7VRgr/3g==
`protect data_keyowner= "ACME_Inc.", data_keyname= "Random"
`protect data_method= "aes256-cbc"
`protect encoding= (enctype="base64")
`protect data_block
pyB+WqJ2KgutN0E39HXls+8GUPF+YedbmZ1CV+u8bz5P8zlj+2FPzWXb9vswNElksxAojMA50VHx
tAN21ebA0gI9tqt4iC7UQkM2epiuvMEm6APC9a7q9XD8LeLqWILv1cqF57PFIiBGvj/EdpC8xwl6
iKV7QP+/Lmd0AV7BNtVEGpr+LpsAcVb/2f11KjU2xkOE3UgI9tki7RTUJ3sD9+OlL4yxljU5Vl9E
BoxCtbegynYyt2yeMsSEE4QtnEbzSgeAZDgr9hGx3QayYEdL2tLwDGV7Ce4BWB1D1Tk7KL4=
`protect end_protected
```

Key Envelope

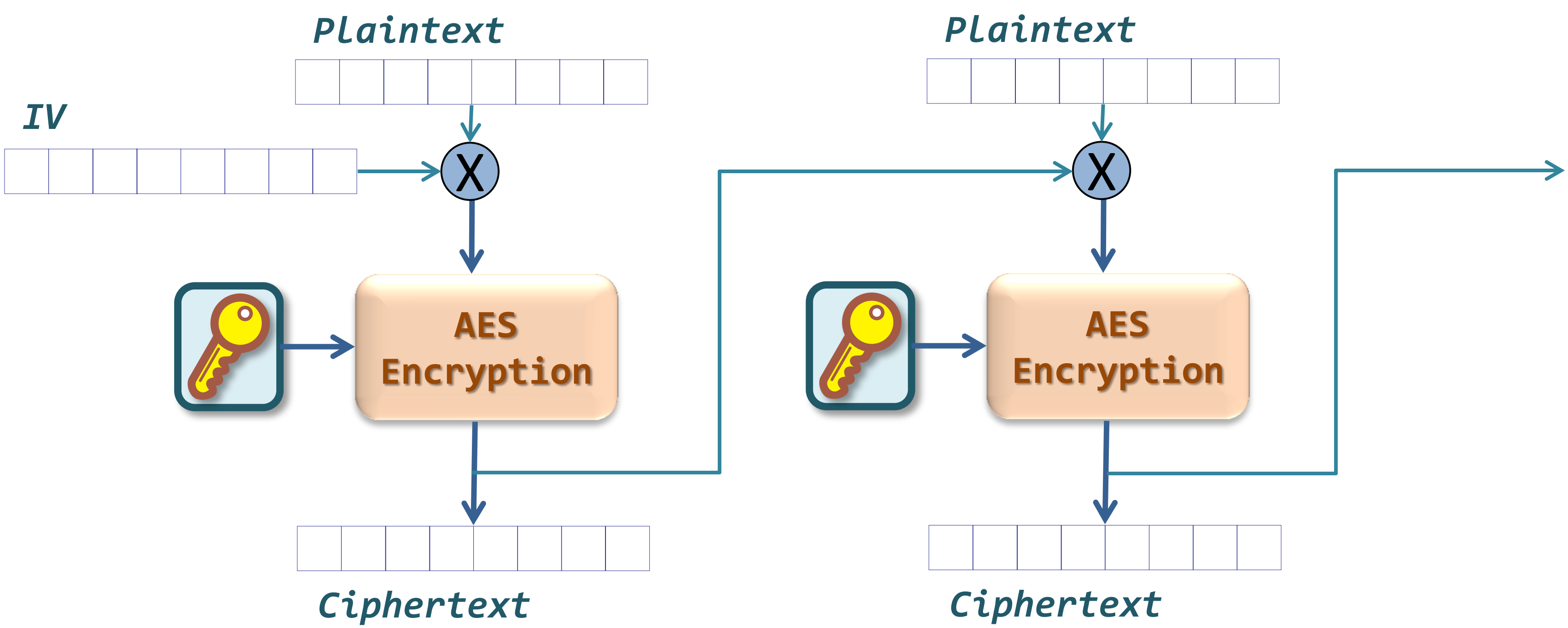
Data Envelope

Slight Imperfections

- Although working solutions based on Verilog, VHDL and SystemVerilog standard were available for some time, not everything looked perfect.
- Even by looking at the list of encryption commands in the LRMs careful readers could notice some discrepancies.
- Attempts to implement encryption in a tool chain quickly led to discovery of some well-hidden, but potentially dangerous issues:
  - Unspecified treatment of ***Initialization Vector*** in symmetric cipher encryption in CBC mode.
  - Unspecified ***padding*** in the RSA (asymmetric) encryption of the session key.
- All those issues make ***interoperability*** of IP encryption problematic…

IV Problem in CBC Mode

- In ***CBC*** mode of symmetric ciphers, the ***Initialization Vector*** (***IV***) is needed.
- It should be random, but does not have to be secret.
- Verilog-2005 and VHDL-2008 did not say how to handle IV.



*Check **Cryptology for Engineers** webinar for more explanations*

RSA Padding Problem

- Hybrid cryptosystem uses RSA cipher to encrypt session key (secret key needed by the symmetric cipher).
- While the longest keys used by symmetric ciphers are 256-bit, the shortest safe RSA keys are 1024-bit.
- It means that a lot of padding must be added to the session key to match the requirements of RSA.
- ***Padding with zeroes or spaces is not safe!***
- Padding selected arbitrarily by the encryption tool may differ from the padding expected by the simulator/synthesizer…

IEEE P1735

- To address IP protection related issues discovered in existing standards and to extend functionality of the system, IEEE created special group working on proposed standard 1735 (***IEEE P1735***).
- Many IP producers and consumers joined P1735 working group – ALDEC included.
- Although no public documents were released yet, the group created internal ***Version 1 Recommendations*** document addressing the issues mentioned in the previous slides.
- Tools that create/recognize **`version = 1`** encryption commands implement ***Version 1 Recommendations*** and should have no problems interacting with each other.

```
`protect begin_protected
`protect version = 1
```

Key Version 1 Recommendations

- IV required for symmetric cipher encryption should be generated randomly and appended in front of the encrypted data before encoding of the data block.
- Padding of session key data during RSA encryption should follow the rules specified in RSAES-PKCS1-v1_5 section of the IETF RFC 3447 document.
- Multiple key envelopes should be allowed in the protected source; a tool reading the source should use key envelope created with its own public key and ignore all remaining key envelopes.

Protection with 2 Key Envelopes

```
`protect begin_protected
`protect version = 1
`protect author= "john.smith@acme.com"
`protect encrypt_agent= "Aldec protectip.pl, rev. 164098"
`protect key_keyowner= "Aldec", key_keyname= "ALDEC08_001", key_method= "rsa"
`protect encoding= (enctype="base64")
`protect key_block
bTPJvH/W4ukdvVJCLPIuFRmsoYfhbhL8O7d9LDF4PPjQKZ+4gGNX9gY8UGmxUFzatIAWmieImp9X
Sex28pndznF6sziTXCifjqgKogj/2vQ9yPV8PoUtsJkpgYaD6m5sQUjPw1ay2IVkcfYMS+Xw3pyc
LDDQWXWXGFkPkIb5m6N1AxktscWMKomh2Ygg+39MSb0LbUEif2Z68mlB8VUMcgmcADL+MpB1tfQf
LLscFb9pMpKAXoPFPbfhaKx3NHwx99kKfRTdLZsNetFMSAbX4019Q4y9qyMkGssB5Y6tj2F0cN8p
Nmoju6yDdTjQh44Y6wxcso+w/AI/CYIykCSM+w==
`protect key_keyowner= "FooBar", key_keyname= "FB 003", key_method= "rsa"
`protect encoding= (enctype="base64")
`protect key_block
Wlx+uDk/bdbs0FXPuwIoBJABdT0WokpF0mUqepKxuiCj35cF1CyMXRGTxexwFnA1QFNgC7YbID7W
M1R2y+F9BopeQtb2w0EA/PRSzdLw3iGL4kj7A+JRI81h1jybmPFmwHkJaos6G+nFLQ1tbi+dQGuR
6i7DnkhGtjoWk6M6mck=
`protect data_keyowner= "ACME_Inc.", data_keyname= "Random"
`protect data_method= "aes128-cbc"
`protect encoding= (enctype="base64")
`protect data_block
6ly36sTXs9sZGj5PjVgnrdeAhi2/UmiHUrKtWTmQIdB5J5eKV+APeqhyuUCxQNHjYo36fIRonuld
Nj6vG18jWp3BMqEXr32GLlrbeY4w9uzvlhygrSZPqQ+PM/Dkqq4Sqk4QOk9s8+oDhUIs+QlyVWUo
5FOysqqFBMyagZQJgruH8fRiDIvBpsVv6vsa3qpk
`protect end_protected
```

ALDEC Key Envelope

Other Tool Key Envelope

Public Key in HDL Source

- Encryption tools implementing ***Version 1 Recommendations*** should recognize public key specified in plaintext source, as shown in Verilog code sample below.
- Encryption tools provided by simulation tool vendors will usually add its own public key encryption automatically.
  (Example: If ALDEC encryption tool processes code listed below, both ALDEC and Acme key envelopes will be present in the output.)

```
`pragma protect key_keyowner = "Acme"
`pragma protect key_method = "rsa"
`pragma protect key_keyname = "ACME_KEY7_11"
`pragma protect key_public_key
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCh/TyTxv6yTpGxBRQ0RBuTNc05
gcwndTBEOgJVKinj9bNiUCLoFU3YTGa/L+M0pTfR/eetiIu1AnFg9Y4sKXYmaCjx
1/7hOB07hUK+vl0xNXk701/Q0wwoQfVsHctTbwRP8NGVKbqlP//QL+o1UC1FPixy
FZy6oMnRULLoBy0s8QIDAQAB
`pragma protect begin
// Verilog code to be encrypted
module top;
 . . .
```

Summary

- Modern cryptology has numerous safe and reliable applications.
- ***Secure IP delivery*** is the cryptology application that all hardware designers may encounter in verification tools.
- Standard for ***Interoperable IP Delivery*** is in the works (IEEE P1735).
- ALDEC participates in the development of this standard.

Conclusion

- Feel free to contact ALDEC if you need more info about EDA-related topics and our design creation and verification tools.

ALDEC Website:	http://www.aldec.com	
Telephone		**E-mail**
USA	+1-702-990-4400	sales@aldec.com
Europe:	+44 (1295) 201240	sales-eu@aldec.com
Japan:	+81-3-5312-1791	sales-jp@aldec.com
India:	+91-80-32551030	sales-sa@aldec.com
China:	+86-21-6875-20-30	info@aldec.com.cn
Taiwan:	+886-2-26599119 ext. 950	sales-tw@aldec.com
Israel:	+972-52-2573422	sales-il@aldec.com

Download:

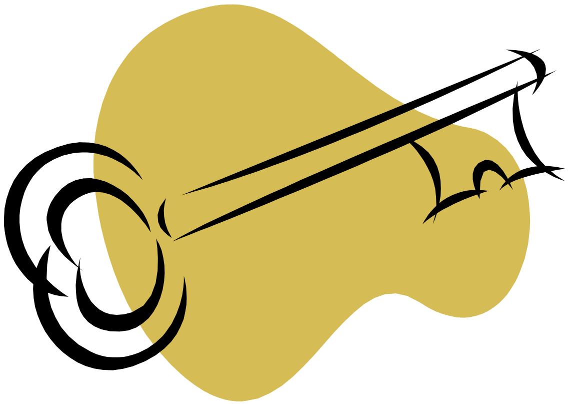
- ***Recorded webinars***
- ***White papers***
- ***Trial versions of our tools***

Extra:

EXPLANATION OF CRYPTOLOGY TERMS

Cryptology

- ***Cryptology*** combines Greek terms ***κρυπτός*** (*kryptos* = secret) and ***λόγος*** (*logos* = study) to describe science or study of hiding, securely transferring and recovering information.
- Cryptology can be divided into two closely related disciplines:
  - ***Cryptography*** – dealing with securing information,
  - ***Cryptanalysis*** – trying to break security.
- Cryptology finds many practical implementations in
  - banking,
  - electronic commerce,
  - telecommunication,
  - military and
  - *IP (Intellectual Property) protection*.



Plaintext

To be, or not to be- that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune
Or to take arms against a sea of troubles,
And by opposing end them. To die- to sleep-
No more; and by a sleep to say we end
The heartache, and the thousand natural shocks
That flesh is heir to. 'Tis a consummation
Devoutly to be wish'd. To die- to sleep.
To sleep- perchance to dream: ay, there's the rub!
For in that sleep of death what dreams may come
When we have shuffled off this mortal coil,
Must give us pause. There's the respect
That makes calamity of so long life.
For who would bear the whips and scorns of time,
Th' oppressor's wrong, the proud man's contumely,
The pangs of despis'd love, the law's delay,
The insolence of office, and the spurns
That patient merit of th' unworthy takes,
. . .

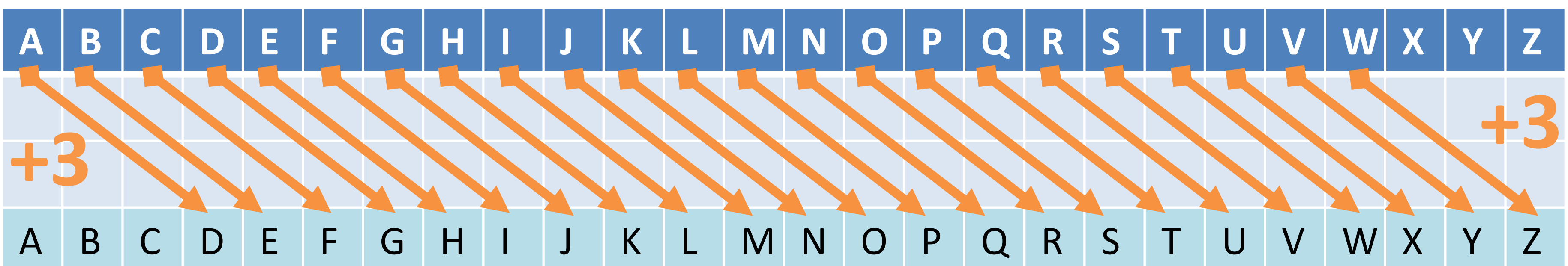
Hamlet 3/1

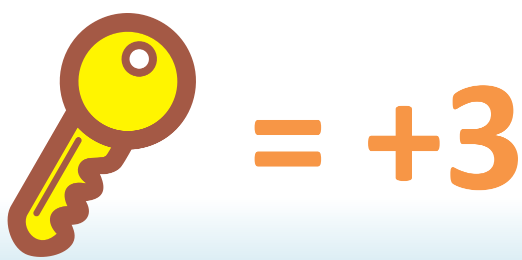
≡

- ***Plaintext*** is the document/message everybody can read and understand.
- We are using ***document icon*** to represent plaintext in diagrams.

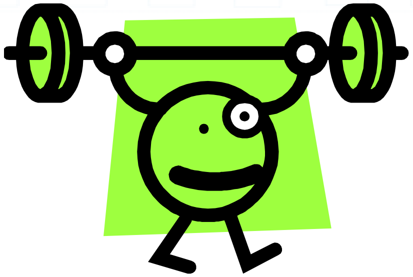
Cipher & Key

- ***Cipher*** is an ***algorithm*** that converts plaintext into something that cannot be read by uninitiated persons and later allows retrieval of the plaintext.
- ***Key*** is a value that personalizes cipher by modifying its algorithm.
- ***Caesar's cipher*** (one of the oldest known ciphers) shifts each letter in plaintext alphabet by given number of positions.
- The key in Caesar's cipher is the number of shifted positions (+3 in our diagram).





Cipher Strength



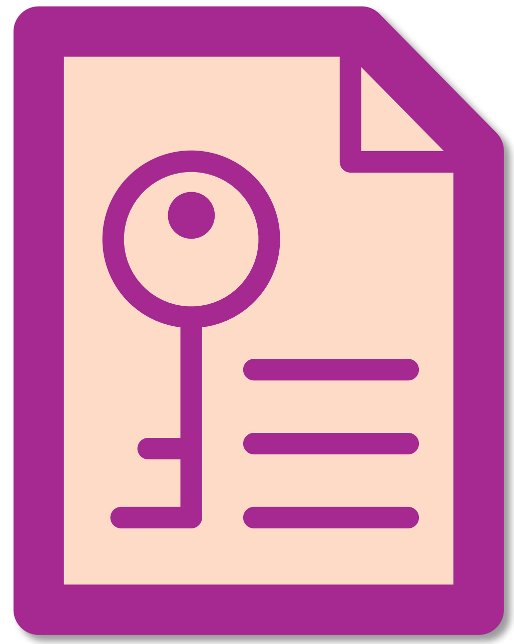
- The stronger the cipher, the more difficult it is to break it for some attacker.
- ***Strength*** of the cipher is measured by ***complexity of the algorithm*** and ***size of the key***.
- The more complex the cipher algorithm, the more difficult it is to devise quick method of reversing it.
  (Caesar's cipher is so simple that anybody can break it now.)
- The longer the key, the more difficult it is to guess it by ***trial and error*** approach, a.k.a. ***brute force attack***.
  (Caesar's key is 5-bit number – 1 of 26 possible in our example – so guessing it without computer takes no more than an hour.)

Ciphertext

Wr eh, ru qrw wr eh- wkdw lv wkh txhvwlrq:
Zkhwkhu 'wlv qreohu lq wkh plqg wr vxiihu
Wkh volqjv dqg duurzv ri rxwudjhrxv iruwxqh
Ru wr wdnh dupv djdlqvw d vhd ri wurxeohv,
Dqg eb rssrvlqj hqg wkhp. Wr glh- wr vohhs-
Qr pruh; dqg eb d vohhs wr vdb zh hqg
Wkh khduwdfkh, dqg wkh wkrxvdqg qdwxudo vkrfnv
Wkdw iohvk lv khlu wr. 'Wlv d frqvxppdwlrq
Ghyrxwob wr eh zlvk'g. Wr glh- wr vohhs.
Wr vohhs- shufkdqfh wr guhdp: db, wkhuh'v wkh uxe!
Iru lq wkdw vohhs ri ghdwk zkdw guhdpv pdb frph
Zkhq zh kdyh vkxiiohg rii wklv pruwdo frlo,
Pxvw jlyh xv sdxvh. Wkhuh'v wkh uhvshfw
Wkdw pdnhv fdodplwb ri vr orqj olih.
Iru zkr zrxog ehdu wkh zklsv dqg vfruqv ri wlph,
Wk' rssuhvvru'v zurqj, wkh surxg pdq'v frqwxphob,
Wkh sdqjv ri ghvslv'g oryh, wkh odz'v ghodb,
Wkh lqvrohqfh ri riilfh, dqg wkh vsxuqv
Wkdw sdwlhqw phulw ri wk' xqzruwkb wdnhv,
. . .

Hamlet 3/1 with Caesar's cipher & +3 key

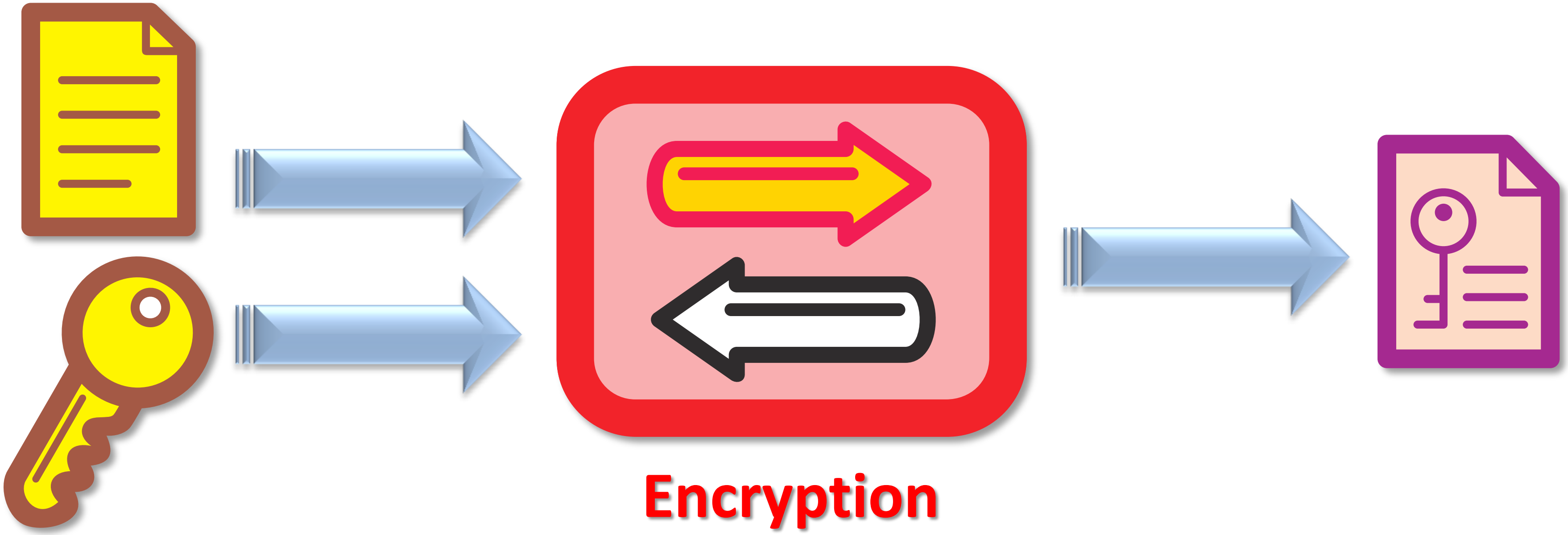
≡



- ***Ciphertext*** is the document/message ***encrypted*** using some cipher and readable only to those who have the key.
- We are using ***document with key*** icon to represent ciphertext in diagrams.

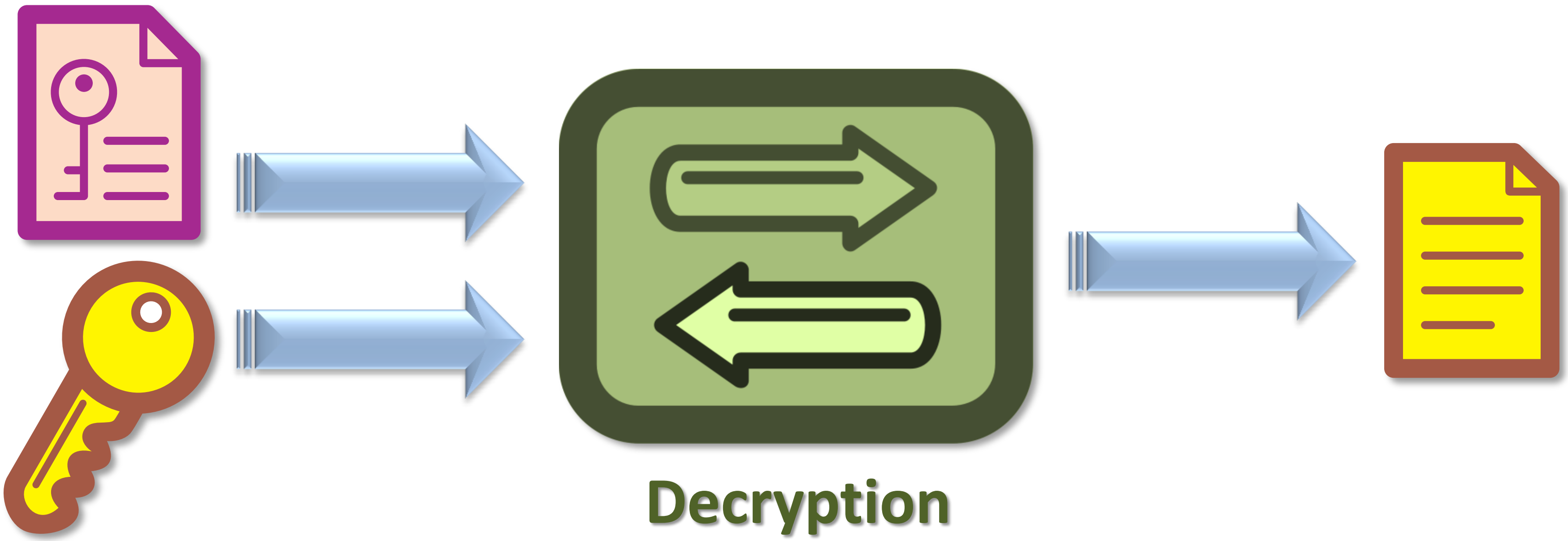
Encryption

- ***Encryption*** is the process of applying selected cipher and key to the plaintext in order to ***obtain ciphertext*** (encrypted message).



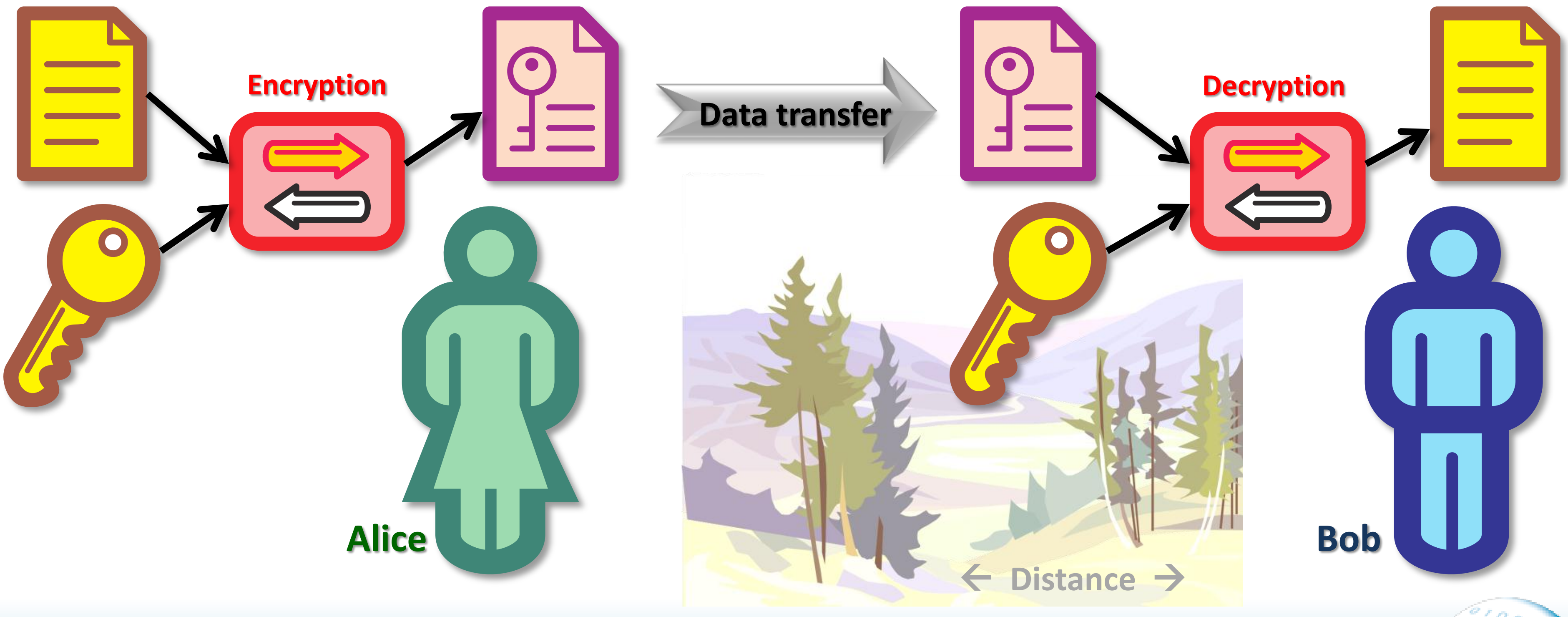
Decryption

- ***Decryption*** is the process of applying known key and cipher (in reverse) to the ciphertext in order to ***recover plaintext*** (original message).



Cryptosystem

- ***Cryptosystem*** is a complete system encompassing all people, procedures, tools, ciphers, keys, and transmission channels involved in a secure data transfer.



Randomness

- ***Randomness*** is a highly desired quality in many cryptology applications.
- ***Randomness*** can be defined in many different ways, but the shortest description says that it is the ***lack of predictability***.
- Randomness can show truly in sequences of numbers if:
  - There is ***no bias*** (all possible numbers show with equal frequency).
  - There are ***no repeatable patterns***.

- Almost all ***random number generators*** are really pseudo-random, but some approach the ideal closely while others are highly predictable.
- Popular random number generators (**`rand()`** function in C, **`Random`** class in Java) are so bad that they should never be used in cryptology applications.
- If true random number generator is not available, always use ***cryptographically secure pseudo-random number generators*** (CSPRNG).

Binary Data in Text Channels

- Sometimes encrypted information must be transferred via channels prepared for text data (e.g. e-mail, IP embedded in source code).
- All modern ciphers produce pure binary output, which can fool text tools into things like jumping to new page or ending transmission.
- The first 32 symbols of the ASCII code are the culprits here: they are known as unprintable ***control characters (codes)***:

Hex	Name	Hex	Name	Hex	Name	Hex	Name
00	NUL	08	BS	10	DLE	18	CAN
01	SOH	09	HT	11	DC1	19	EM
02	STX	0A	LF	12	DC2	1A	SUB
03	ETX	0B	VT	13	DC3	1B	ESC
04	EOT	0C	FF	14	DC4	1C	FS
05	ENQ	0D	CR	15	NAK	1D	GS
06	ACK	0E	SO	16	SYN	1E	RS
07	BEL	0F	SI	17	ETB	1F	US

Benefits of Encoding

- The method of solving 'binary in text' problem is called ***encoding***: for each 3 bytes of un-encoded message, 4 printable characters are generated.
- Character set in the encoding output is selected so that it looks the same no matter which text tools open it/ what computing platform is used.
- Once popular ***UUencoding*** was now replaced with ***Base64*** encoding.

Plaintext	Encoding can be very useful!
AES ciphertext (hex)	66 EF 21 AC 78 C0 65 97 FD 65 3F 66 C6 A4 A8 82 76 43 03 97 AA 0C C4 63 3F FA EB BE 7F 0E BF 54
Base64 encoded ciphertext	Zu8hrHjAZZf9ZT9mxqSognZDA5eqDMRjP/rrvn8Ov1Q=

Documentation Sources

Standards and Other Cryptology Documents:

DES official description: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
AES official description: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
RSA standard page: http://www.rsa.com/rsalabs/node.asp?id=2125
Cryptographic Toolkit: http://csrc.nist.gov/groups/ST/toolkit/index.html

P1735 Website:

Working Group official website contains some public data, but many private documents require login: http://www.eda.org/twiki/bin/view.cgi/P1735/WebHome

NOTES:

- *Wikipedia is a surprisingly reliable source of additional data about cryptology.*
- *Visit cryptool.org if you want to experiment with cryptology safely.*