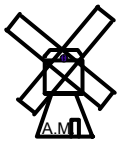
HCC Cryptography Presentation

HCC Presentatie
Oktober 2017

Ir. Andries Mulder
Mulder Training & Consultancy
the Netherlands
gsm. +31 651 71 40 00
tel. +31 544 37 40 37
email driesmulder@kpnmail.nl

A. Mulder
driesmulder@kpnmail.nl



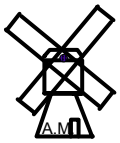
HCC Cryptography Presentation

Introduction of Andries Mulder

- Andries Mulder from the Netherlands
- Studied applied mathematics in the 70's
- Worked for IBM from 1978 - 2002
  - Systems Engineer working w. IBM S34, S36
  - Stepped into Pc's in early 80's
  - Moved into crypto in 89
    project requiring Pc skills and crypto
- Self employed since 2002

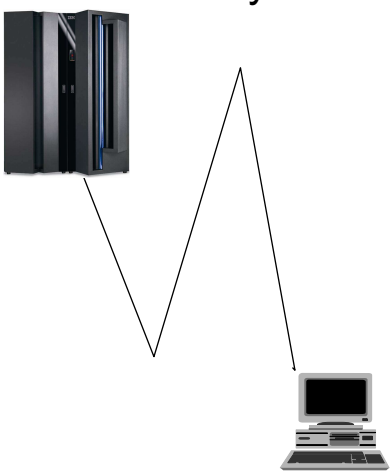
- I have been running classes on:
  - Intel Assembler programming
  - OS/2 internals
  - IBM Crypto
  - C-programming
  - Lotus 123
  - MFC programming
  - IBM UDX Toolkit

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Security Goals



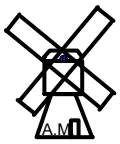
Authentication
(findout about honest
or dishonest user)

Non-Repudiation
(Make sure that sender
of a message cannot deny
that he sent this message)

Prevent Active Wiretapping
(modifying data)

Prevent Passive Wiretapping
(listening)

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Cryptographic Algorithms

- **In Cryptography we have two Mainstreams:**

- **Symmetric Algorithms**
  - ***DES + TDES***
  - ***AES***
  - ***Blowfish, Twofish***
  - ***RC4, RC5***
  - ***etc.***

Will be discussed right now:
Are called symmetric because

key for ENCIPHER = key for DECIPHER

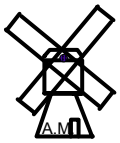
- **A-Symmetric Algorithms**
  - ***RSA***
  - ***Diffie-Helmann***
  - ***Elliptic Curve***
  - ***etc.***

Will be discussed in a separate presentation

Uses a Private and a Public key

In IBM products we use a mix of RSA, DES, AES and ECC

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

The DES Algorithm

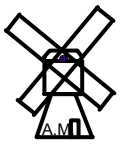
Developed by IBM in the seventies
Uses a 56 bits key
Basic functions:
- Encipher
- Decipher

Symmetric Algorithm

Encipherkey = Decipherkey

I am focussing a little bit on DES here as it is still widely used especially by banks. For new projects AES is the preferred symmetric algorithm. Nobody uses Single DES anymore but TDES instead

A. Mulder
driesmulder@kpnmail.nl



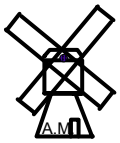
HCC Cryptography Presentation

The AES Algorithm

- AES is a newer Symmetric algorithm
- NIST approved in 2001
- Uses key lengths of 128, 192 or 256 bits
- Blocklength 16 bytes
- Basic functions: ENC DEC
- Invented in Belgium (Rijmen, Daemen)
- Effective as a Federal government standard on May 26, 2002
- NIST has published an AES_KAT file (Zip format)

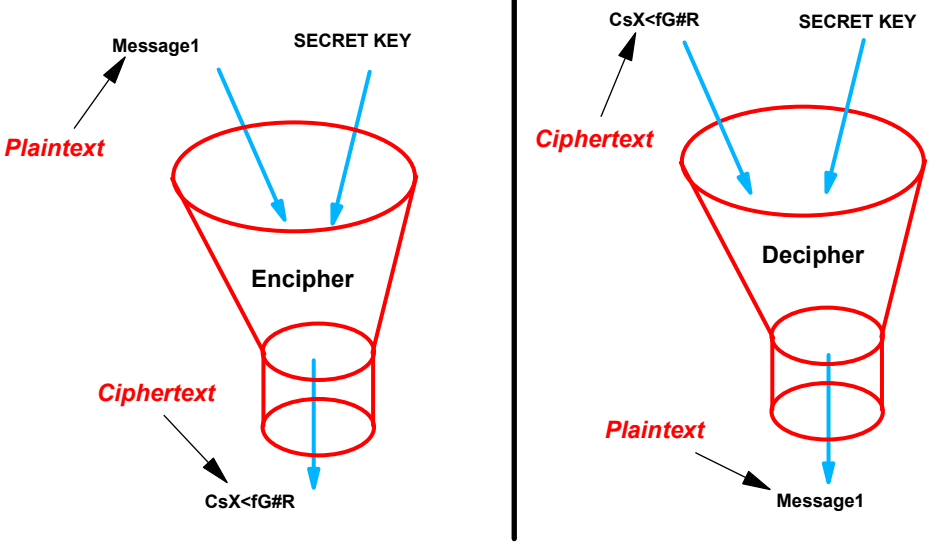
The usage of DES and AES do not have big differences except for other key-sizes and block-kengths. And ... Most important: AES offers better security and performance and is therefore the preferred symmetric algorithm for new projects.

A. Mulder
driesmulder@kpnmail.nl



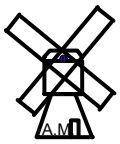
HCC Cryptography Presentation

The DES or AES Algorithm



Same Secret Key used for Encipher and Decipher

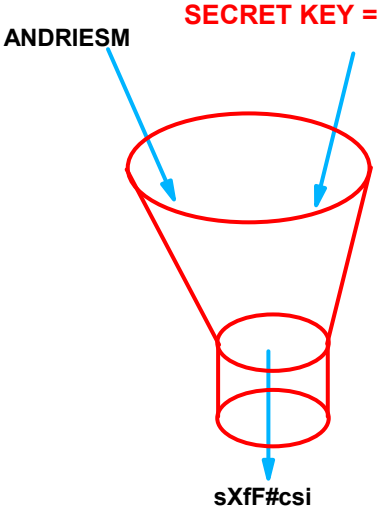
A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

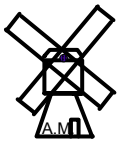
The DES Algorithm: Basic Facts for Single DES

SECRET KEY = 86 8A BC 51 BA B4 19 9D (in hex)



- In Single DES, the key has a length of 56 bits but parity bits are added
- Input = Plaintext / Cleartext
- Output = Ciphertext
- Plaintext = 8 bytes
- Ciphertext = 8 bytes
- Key Inp = Key Outp

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Properties of Symmetric algorithms (DES , AES)

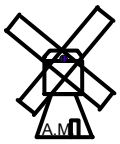
Entities involved when DES / AES operates:

Plaintext
Key
Ciphertext

all nn bytes blocks
datalen = 8 or 16, keylen = 8, 16, 24, 32

- Knowledge of Ciphertext does NOT give knowledge of plaintext
- Knowledge of both ciphertext and plaintext does NOT give knowledge of key
- Key used for ENC = Key used for DEC

A. Mulder
driesmulder@kpnmail.nl



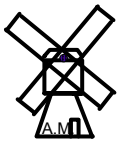
HCC Cryptography Presentation

Characteristics of DES, TDES and AES

	DES	TDES	AES
Key Length	56 bit	112, or 168	128 192 256
Block Length	8 byte	8 byte	16 byte
Strength	obsolete	acceptable	good
Speed	slow	slower	faster

- Single DES is obsolete now
- TDES uses mostly a Double length key
  TDES Masterkeys can be Triple length
- AES has a scalable key length (16, 24 or 32)
- DES, TDES and AES are all BLOCK-Ciphers

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Key Length and Algorithm strength

For all algorithms, the following holds true:

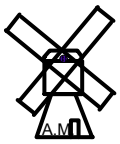
- The longer the key, the better the strength
- The longer the key the lower the Performance

The following (NSA) table shows equivalent key sizes (best guess)

Symm. Key Size	RSA, DH mod len	ECC key len
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

see www.nsa.gov/business/programs/elliptic_curve.shtml

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Secret and Non-Secret Algorithms

- **Published Algorithms**
  - Allow for independent strength evaluation and analysis
  - Allow normal distribution procedures
  - Allow and encourages widespread use
  - Encourage new applications
  - Make standardization easy to accomplish
- **Secret Algorithms**
  - Do not allow independent strength evaluation and analysis
  - Create distribution problems
  - Restricts use
  - Complicates standardization
  - No guarantee that it is a good algorithm !

A. Mulder
driesmulder@kpnmail.nl

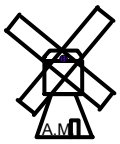
HCC Cryptography Presentation

A DES test set (KAT)

DES is highly standardized; Any implementation should give the same results on this test-set

Key	Input (plain)	Output (Cipher)
0000000000000000	0000000000000000	8CA64DE9C1B123A7
FFFFFFFFFFFFFFFF	FFFFFFFFFFFFFFFF	7359B2163E4EDC58
3000000000000000	1000000000000001	958E6E627A05557B
1111111111111111	1111111111111111	F40379AB9E0EC533
0123456789ABCDEF	1111111111111111	17668DFC7292532D
1111111111111111	0123456789ABCDEF	8A5AE1F81AB8F2DD
0000000000000000	0000000000000000	8CA64DE9C1B123A7
FEDCBA9876543210	0123456789ABCDEF	ED39D950FA74BCC4
7CA110454A1A6E57	01A1D6D039776742	690F5B0D9A26939B
0131D9619DC1376E	5CD54CA83DEF57DA	7A389D10354BD271
07A1133E4A0B2686	0248D43806F67172	868EBB51CAB4599A
3849674C2602319E	51454B582DDF440A	7178876E01F19B2A
04B915BA43FEB5B6	42FD443059577FA2	AF37FB421F8C4095
0113B970FD34F2CE	059B5E0851CF143A	86A560F10EC6D85B
0170F175468FB5E6	0756D8E0774761D2	0CD3DA020021DC09
43297FAD38E373FE	762514B829BF486A	EA676B2CB7DB2B7A
FFFFFFFFFFFFFFFF	0000000000000000	CAAAAF4DEAF1DBAE
0123456789ABCDEF	0000000000000000	D5D44FF720683D0D
FEDCBA9876543210	FFFFFFFFFFFFFFFF	2A2BB008DF97C2F2

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Protecting the PRIVACY of Data

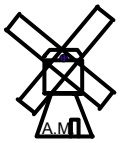
Classical application of Cryptography

Used in military environment

Make the data secret by using strong encipher

Requires a shared secret key between sender and receiver of a message

A. Mulder
driesmulder@kpnmail.nl



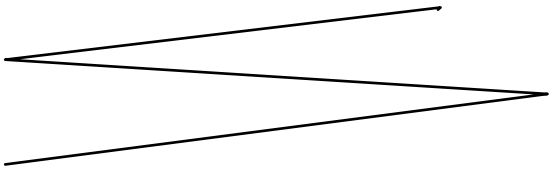
HCC Cryptography Presentation

Protection of the Integrity of Data

Use a Cryptographic Error Detection Method like a MAC

MESSAGE MAC GENERATE MAC

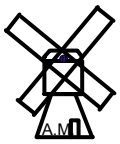
SEND MESSAGE + MAC to other node



MESSAGE MAC Do a MAC VERIFY

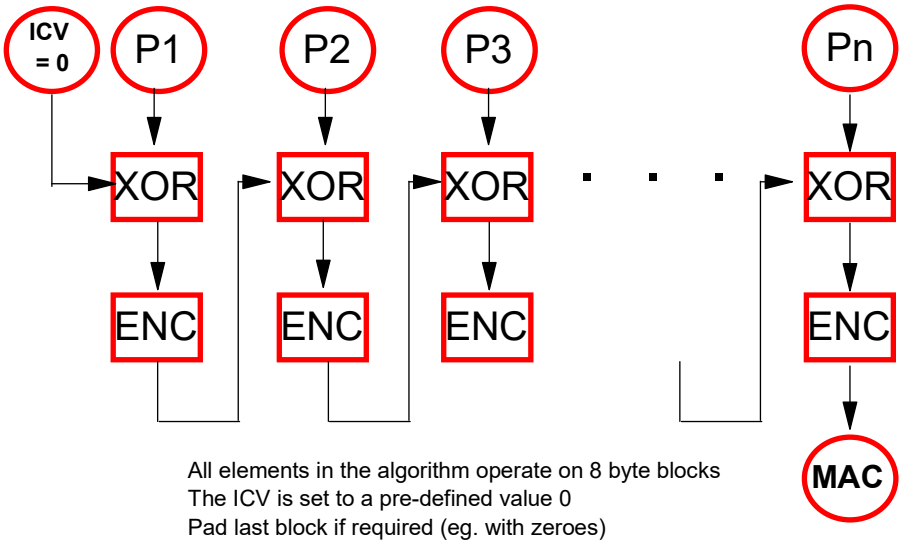
We can have TDES Mac's and AES MAC's

A. Mulder
driesmulder@kpnmail.nl



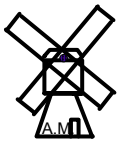
HCC Cryptography Presentation

The MAC algorithm



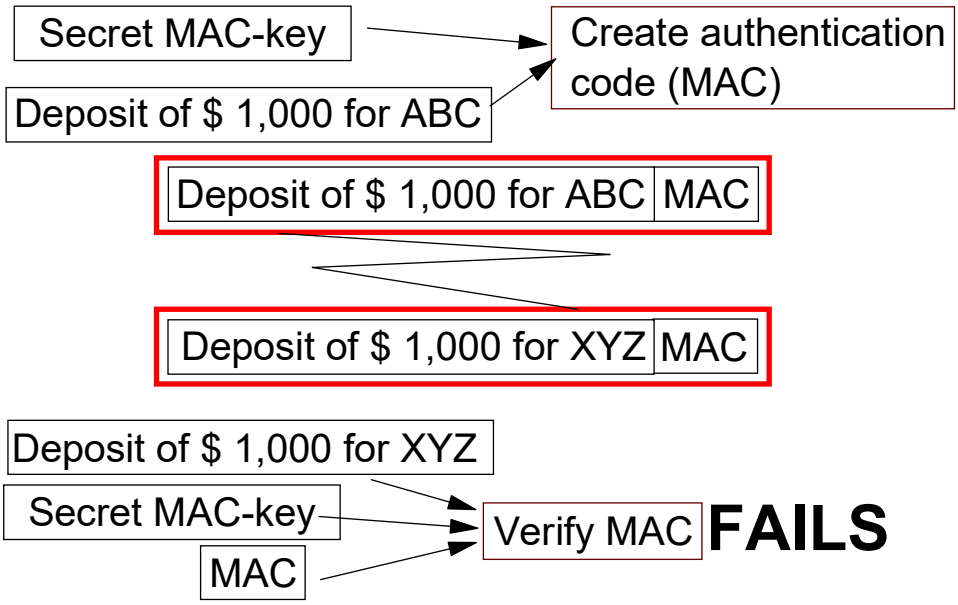
All elements in the algorithm operate on 8 byte blocks
The ICV is set to a pre-defined value 0
Pad last block if required (eg. with zeroes)

A. Mulder
driesmulder@kpnmail.nl

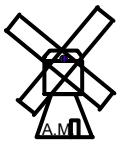


HCC Cryptography Presentation

MAC Protection (2)



A. Mulder
driesmulder@kpnmail.nl



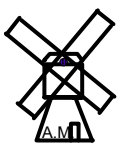
HCC Cryptography Presentation

Hashing algorithms

- Hashing is a One Way Function that can protect the integrity of data
- Hashing uses a non-secret key (build in algorithm) or no key
- Hashing makes a so called Hash which is a value of at least 128 bits.
- When the message is available, one can calculate the Hash;
- It is impossible to create a message that matches a given Hash
- The receiver of a message, re-calculates the Hash and compares to the one that was received
- Message and Hash are sent using separate transportation channels
- There are several Hashing algorithms

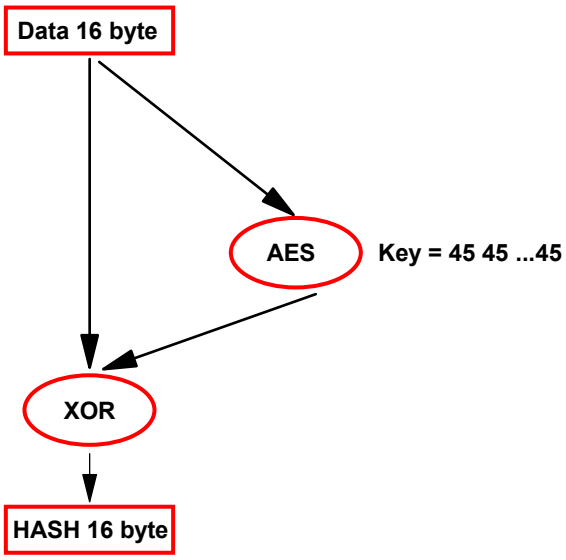
  SHA-1, SHA-2, SHA-3 , MD5 ..... (CSNBOWH verb)

A. Mulder
driesmulder@kpnmail.nl

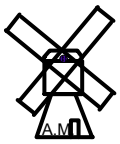


HCC Cryptography Presentation

An example (trivial) of a Hash algorithm



A. Mulder
driesmulder@kpnmail.nl

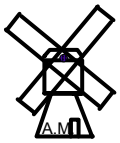


HCC Cryptography Presentation

Triple DES

- DES can be made stronger by doing it multiple times
- Triple DES uses a Double length key or a Triple length key
- Triple DES is much much stronger than Single DES
- Triple DES is used for Key Management purposes
  but also on Data (since 2000)
- DES and TDES are outdated now, but still not "broken"
  Because banks have large investments in ATM's they
  continue to use TDES.
- **For new projects, AES is preferred choice**

A. Mulder
driesmulder@kpnmail.nl



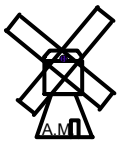
HCC Cryptography Presentation

Triple DES Encryption with a Triple length key

K1, K2 and K3 are each 56 bits and 8 parity bits

The 4765 uses this type of encryption to protect an Operational DES key by the **DES Masterkey** of the 4765 (local protection)

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Using the DES / AES Algorithm on other text-lengths

- DES operates on quantities of 8 bytes (AES on 16)
- Very often we will have input-text of which the length:
  - Is longer than the block-length
  - Is not a multiple of the block-length
- We can overcome this problems by:
  - Processing longer texts in blocks and just issue a sequence of calls to DES or AES The **chaining** of blocks can be done in several ways (ECB, CUSP, CBC etc)
  - Adding a few dummy bytes to a non-block-length text This is called **padding**

    Sender and recipient of encrypted data should agree on the procedure to do the chaining and about the padding (eg. X9.23)

A. Mulder
driesmulder@kpnmail.nl

HCC Cryptography Presentation

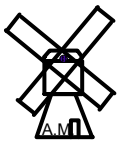
Cipher Block Chaining

To avoid repetetive output when enciphering repetetive input, DES and AES can be used in Cipher Block Chaining mode.

e.g. You encipher a plaintext with many blanks in it; You don't want the output to be vulnerable to a statistical attack (counting freq's)

CBC makes a statistical attack impossible, by taking the output of Encipher n as a part of the input for Encipher n+1

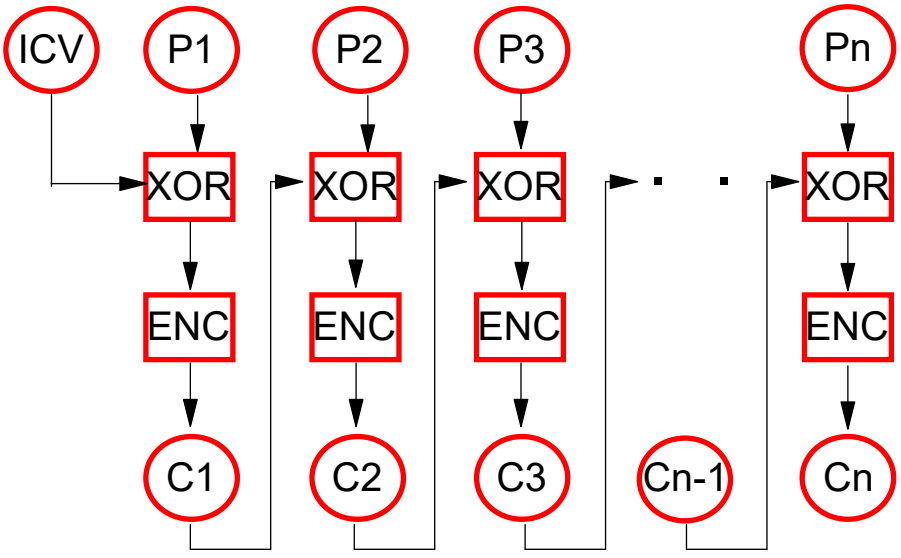
A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

CBC Encipher

ICV = Initial Chaining Vector

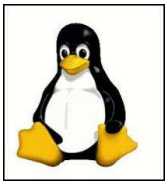


A. Mulder
driesmulder@kpnmail.nl

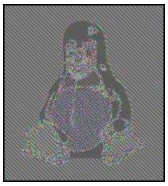
HCC Cryptography Presentation

Cipher Block Chaining effect

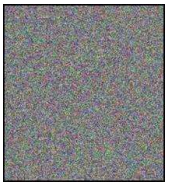
Clear picture



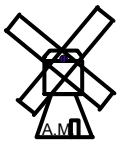
ECB Enciphered



CBC Enciphered



A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Tamper Resistant Hardware

In Cryptographic hardware, the system is designed in such a way that sensitive variables are protected by a hardware shield.

Tamper Resistant Module →

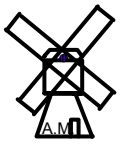
Physical Security

Assure secrecy and integrity of:
- Algorithm
- Crypto variables
- Crypto functions

No secret crypto variable exists outside crypto hardware

Tamper Resistant Hardware gives excellent protection against insider-attacks
You can USE a key, but you can not REVEAL it

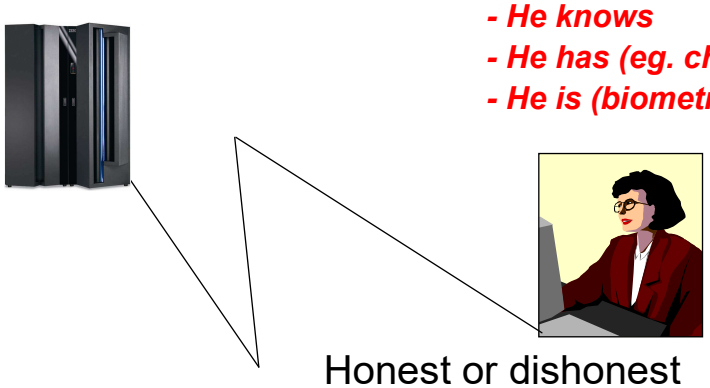
A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Authentication (1)

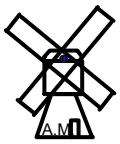
Cryptography can be used to give proof that someone is really who he says to be.
The user is authenticated based on something:



- He knows
- He has (eg. chipcard)
- He is (biometrics)

Honest or dishonest user

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Key Management

- Whole set of functions to:
  - Generate keys
  - Enter keys
  - Backup/Restore keys
  - Revoke keys
  - Distribute keys
  - Verify keys
- This must be done securely
- Additional requirements:
  - function separation
    eg. MAC / MACVER or ENC / DEC
  - dual control

A. Mulder
driesmulder@kpnmail.nl

HCC Cryptography Presentation

Key wrapping

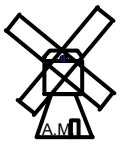
Keys do (almost) never exist in clear format,
but are always enciphered using another key.

Keks are keys that are used to encrypt other keys

- Key-Encrypting Key (KEK)
- DES KEK's are Double Length
  or Triple length (128 bits or 192 bits)
  AES KEK's can be 128, 192 or 256 bits

- Examples of KEK's
  - Masterkey
  - EXporter key
  - IMporter key

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

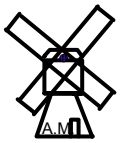
Masterkey Concept

- A key that is stored outside protecting hardware
  is always enciphered (3x) using another key
- For that purpose we keep one special key
  inside our hardware,
  that is called the **MASTERKEY**
- The Masterkey itself is stored inside protecting hardware
  in a special place called Masterkey-Register
- The Masterkey KM protects other keys that are stored
  locally on our system(s)
- A key that is encrypted using KM is ready for immediate
  use and is therefore named "Operational"
- An operational key on system A is denoted:

$$E_{KM_A}(KD)$$

- Now we can have up to 4 different Masterkeys
  SYM-MK, ASYM-MK, AES-MK, and APKA-MK

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

KEY Generation

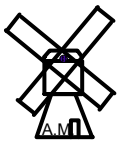
The generation of a Symmetric key (DES / AES) basically consists of the construction of a "real" random number (ODD parity for DES keys)

Most Cryptosystems do have a RandomNumberGenerate facility that generates (pseudo) random numbers

There are also systems, where the key is derived from a user chosen password or passphrase (derived keys)

It is also possible to derive a key from a known value and from another key. (smartcards derive a key based on S/n and 1 other key)

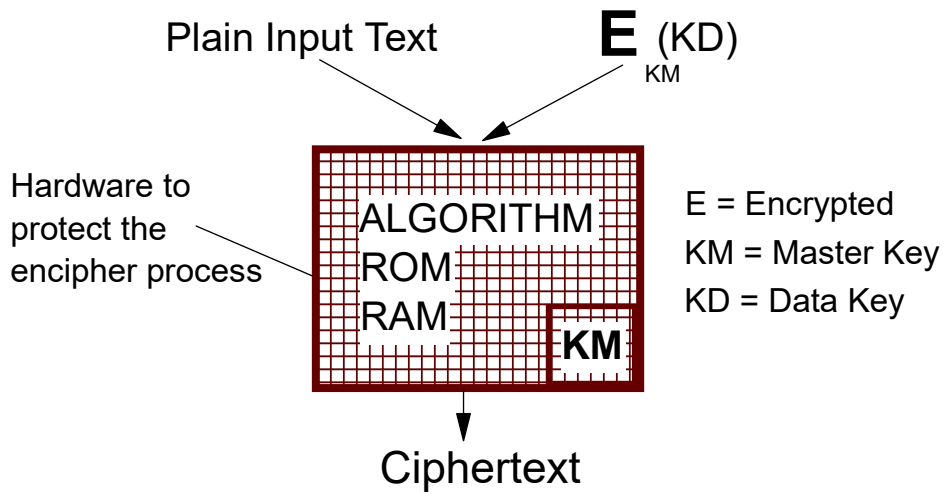
A. Mulder
driesmulder@kpnmail.nl



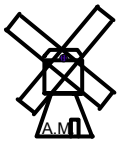
HCC Cryptography Presentation

Hiding Keys (2)

Triple Encryption/Decryption of a key when performing encipher



A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

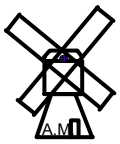
Distribution of Symmetric keys

In most symmetric cryptographic systems we do have a "chicken and egg" problem:

- We want to distribute keys, but we don't have a protected path to transport the keys
- Once the first key is distributed, we can transport next key-generations using the old keys.

This problem has a very elegant solution that uses the RSA or ECC algorithm, but if we use a SYMM-only system, we must protect the very first key transport by non-cryptographical measures.

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

The Key Part Import Operation

How to build a TDES key from 4 different random-numbers ?

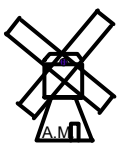
Most systems (like CCA) support an instruction like Key_Part_Import, which reads in keyparts and enciphers them using the KM

$KEY = E_{KM} ((RN1 + RN2) XOR (RN3 + RN4))$

Note: In key management matters, enciphering is done triple

For AES, it is similar (len = 16, 24, 32 bytes)

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Key Export Operation

Many crypto implementation do have an instruction like:

Key export

Source of key export is: $E_{KM}(KD)$

What is done: (inside hardware)

triple decipher from Masterkey

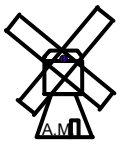
triple encipher using KEK

Result is: $E_{KEK}(KD)$

So the transformation is as follows: $E_{KM}(KD) \longrightarrow E_{KEK}(KD)$

For AES, a similar process exists.

A. Mulder
driesmulder@kpnmail.nl



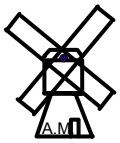
HCC Cryptography Presentation

Keytest instruction

- In some situations you want to have a check if a key still has the same value
- However, You don't want to ask for its value or to registrate its value (because of security)
- But if you could record some (randomly chosen) input value and some value resulting from a cryptographic operation that used the key (under investigation), these recorded values could help you in finding out

A "Fingerprint" of the key that does not expose the key-value

A. Mulder
driesmulder@kpnmail.nl

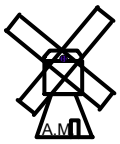


HCC Cryptography Presentation

Keytokens

- All CCA keys are "enveloped" in a Token
- In general, you don't use clear keys, but you use **Wrapped** keys in a Keytoken
- A wrapped key in a token is always enciphered using another key (KM or KEK)
- At this encryption, also **key attributes** are included
- When the key is to be used, it is recovered Inside the CF, which is "**secure hardware**"
- The key attributes give protection against mis-use
  - **Control Vector** (CV) for DES keys
  - **Associated Data** for AES, RSA, ECC keys

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

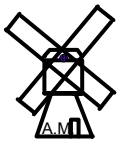
A description of the IBM 4767 PCIe Cryptographic Coprocessor

IBM 4767

Secure Co-processor

See...
http://www.ibm.com/security/cryptocards

A. Mulder
driesmulder@kpnmail.nl



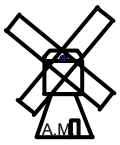
HCC Cryptography Presentation

What is a Secure Cryptographic Coprocessor?

A general-purpose computing environment that withstands physical attacks (and logical attacks)

- It runs the programs it's supposed to
- It runs them unmolested
- One can (remotely) tell the difference between the real card and a clever impersonator
- An attacker might carry out destructive analysis of one or more devices
- *Not just a fast crypto box*

A. Mulder
driesmulder@kpnmail.nl



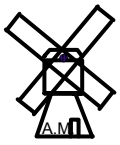
HCC Cryptography Presentation

Hardware Protection in the 4767

- Tamper resistant metal/wiring shield is the 1st barrier built in
- High/Low temperature detection
- Intrusion detection
- High/Low voltage detection
- DPA prevention (DPA = Differential Power Analysis)

Very hard physically to tamper

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

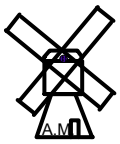
Enabled Target Applications

- High speed bulk encryption/decryption of ...
  - Digital content of movies and interactive TV
  - Confidential video conferences
  - Patient health records to/from insurance companies, hospitals, doctors offices
  - Telecommuting

Secure application in an un-trusted environment

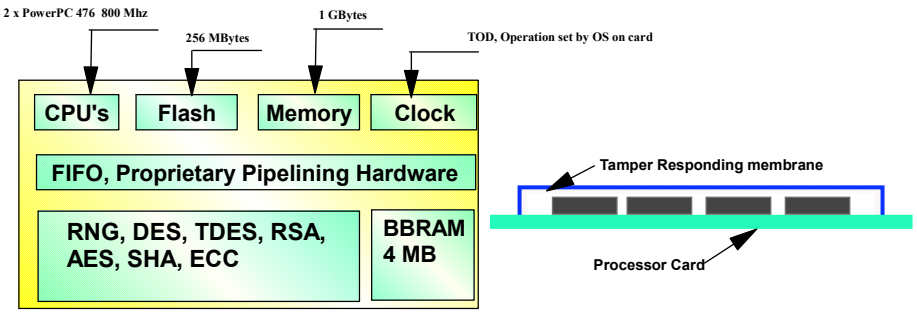
- e-commerce (secure signature)
- Metering
- CA's
- Electronic secure trading (secure time stamps)
- Post Office postage meters
- Internet gambling
- Home banking
  - **ATM+ PIN**

A. Mulder
driesmulder@kpnmail.nl

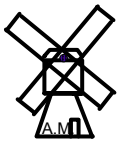


HCC Cryptography Presentation

4767 hardware

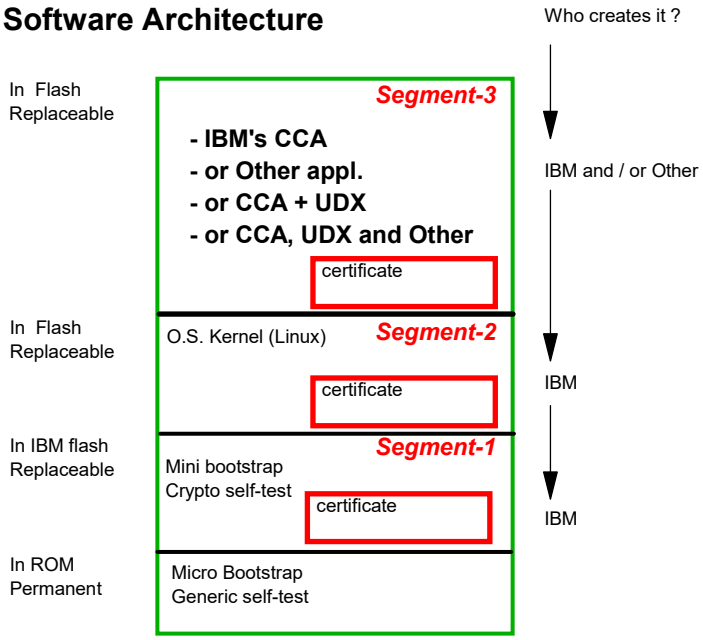


A. Mulder
driesmulder@kpnmail.nl

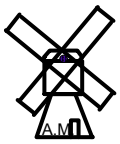


HCC Cryptography Presentation

Layered Software Architecture



A. Mulder
driesmulder@kpnmail.nl



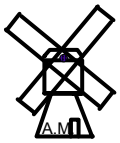
HCC Cryptography Presentation

Access Control

- The 4767 has a built in Access Control System that is designed to run in a multi-tasking environment
- Different sessions can have different access rights
- A session obtains it access rights by logging on
- Access rights are based on Roles and Profiles
  - A Role defines access rights and authorizations
  - A Profile points to a specific Role
  - A user will normally logon to a Profile
  - If Nobody has logged on , a DEFAULT is valid
  - All Roles and Profiles are customizable

User → Logon → Profile → Role → Access Rights

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Triple DES performance of 4767

TDES encryption (performance very dependant from blocksize)

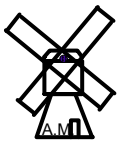
Block Size	Calls / Sec	Byte / Sec
8	5200	41.600
128	5200	666.000
1.024	5080	5.190.000
8.192	3350	27.489.900
65.536	680	45.000.000
524.288	95	50.000.000
4.194.304 (4MB)	12	50.600.000

What does it tell ?

Card does about a 5200 calls per second, for easy operations. (eg. 8 bytes of TDES) So 1 call takes about 0,2 milli-seconds. Max troughput ± 50 MB/sec

These are figures measured at Host side from a real single threaded program (tstperf1.c) Note: Multi-threading will dramatically increase total troughput

A. Mulder
driesmulder@kpnmail.nl



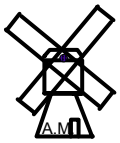
HCC Cryptography Presentation

RSA performance of 4767

Operation	Calls / Second Exp = 65537	Calls / Second Exp Random	Calls / Second Exp = 3
RSA Keygen 1024 ModExp	30	7	12
RSA Keygen 1024 CRT	26	8	10
RSA Keygen 2048 CRT	8	1	3
RSA Keygen 4096 CRT	0.9	XXXX	0.9
RSA Priv 1024 ModExp	1160	1146	1157
RSA Priv 1024 CRT	2460	2460	2518
RSA Priv 2048 CRT	1040	1036	1040
RSA Priv 4096 CRT	205	XXXX	206
RSA Pub 1024 ModExp	3319	1103	3325
RSA Pub 1024 CRT	3325	109430	3290
RSA Pub 2048 CRT	2740	214	2900
RSA Pub 4096 CRT	1776	XXXX	2083

These are figures measured at Host side from a real single threaded program.

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

AES performance of 4767 (256 bits enciphered key)

AES encryption (performance very dependant from blocksize)

Block Size	Calls / Sec	Bytes / Sec
16	5000	80.000
128	4950	634.000
1.024	4900	5.019.600
8.192	3344	27.340.000
65.536	717	47.000.000
524.288	100	52.000.000
4.194.304 (= 4 MB)	12	52.000.000

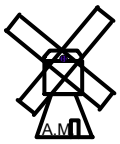
What does it tell ?

Card does about a 5000 calls per second, for easy operations. (eg. 16 bytes of AES) So 1 call takes about 0,2 milli-seconds.

Max troughput ± 52 MB/sec

These are figures measured at Host side from a real single threaded program
Note: Multi-threading will dramatically increase total troughput (see next slide)

A. Mulder
driesmulder@kpnmail.nl



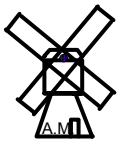
HCC Cryptography Presentation

Public Key Concepts

PKA --> Public Key Algorithm

RSA
ECC

A. Mulder
driesmulder@kpnmail.nl

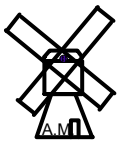


HCC Cryptography Presentation

The RSA Algorithm

- Asymmetric algorithm
- Two keys instead of one (Key - PAIR)
  - Private key (sometimes denoted PR)
  - Public key (sometimes denoted PU)
- RSA = Rivest/Shamir/Adleman
- Sometimes called PKA = Public Key Algorithm

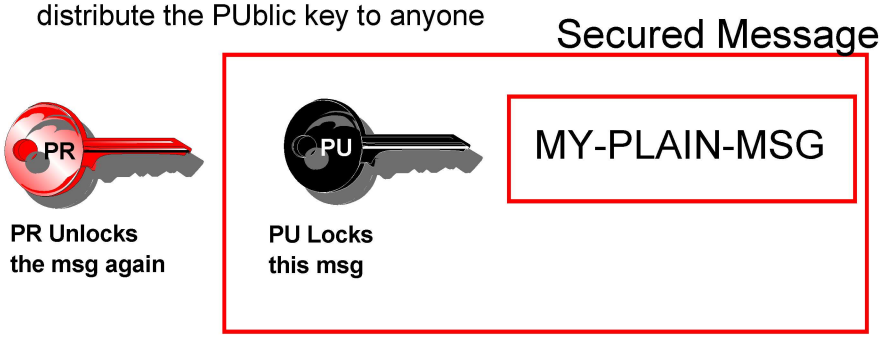
A. Mulder
driesmulder@kpnmail.nl



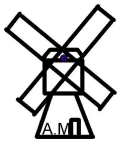
HCC Cryptography Presentation

Concepts of RSA (2)

- Encrypt/Decrypt (= applying PU or applying PR) is commutative which means that the sequence in which you apply PR or PU does not make any difference.
- The system generates PR/PU as a PAIR
- Knowledge of PU does NOT give knowledge of PR, so you can distribute the PUblic key to anyone

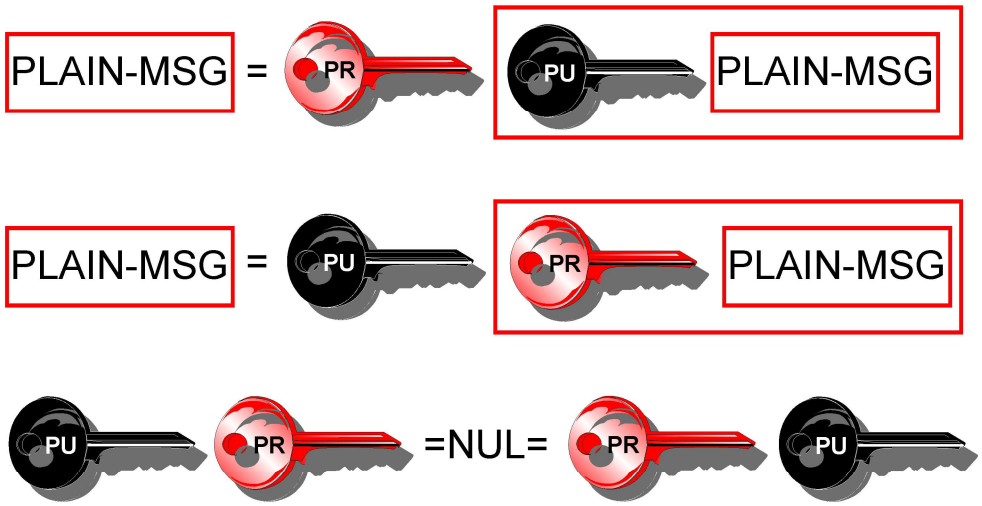


A. Mulder
driesmulder@kpnmail.nl

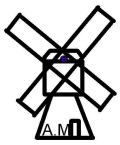


HCC Cryptography Presentation

Do - Undo behaviour of RSA



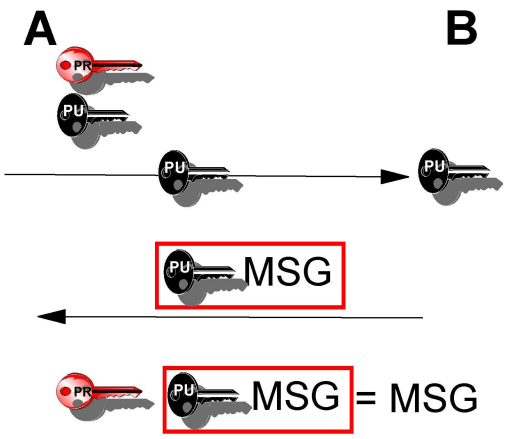
A. Mulder
driesmulder@kpnmail.nl



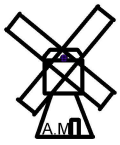
HCC Cryptography Presentation

Basic Usage of RSA

- Two nodes A and B
- A generates PR/PU pair
- A sends PU to his "friend" B
- Now B can send A a message that is locked with A's PU
- Only A can Unlock this msg, using his PR
- Lock = Encrypt Unlock = Decrypt



A. Mulder
driesmulder@kpnmail.nl



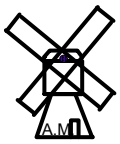
HCC Cryptography Presentation

Problems in of the Concept

- How can B be sure that he really got A's public key?
- An adversary could easily replace A's PU by his own PU and B would not notice
- Speed: RSA is too slow to be used for data encryption

Therefore we use the concept in a slightly modified way

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Application Areas of RSA

- Initial SYMM key distribution
- Digital Signatures

In most cases we will use
hybrid Symmetric / RSA equipment

A. Mulder
driesmulder@kpnmail.nl

HCC Cryptography Presentation

Digital Signatures (1)

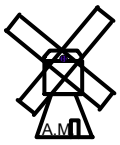
We need a mechanism :

- That can replace an old fashioned signature on a contract or on a message
- That can be added electronically to a digital message

What should it prove ?

- That the Message is proven to come from a certain node
- That the Message has not been modified on its way from origin to its destination
- That the origin has really send this message (no denial or non-repudiation)

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Digital Signatures (2)

The IBM CCA products offer a Digital Signature mechanism that:

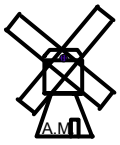
- **Satisfies all the requirements on previous sheet**
- **That is ISO 9796 complient**

A Digital Signature prevents:

- Manipulation of the message, by an outsider or by the receiver

- The sender of the message to deny that he sent this message

A. Mulder
driesmulder@kpnmail.nl



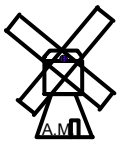
HCC Cryptography Presentation

Generating a Digital Signature

- Node A generates a PR/PU pair
- Node A distributes PU to everyone with whom he communicates
- Node A creates a message M
- Node A compresses M to a shorter (128 or 160 bits) value, using a one-way-process eg. MDC, MD5 or SHA-1)
  The output of this process is called "HASH" or "DIGEST"
- Node A decrypts the hash, by applying his PRivate key on the hash
- The DECRYPTED value is called "Digital Signature"

Note: Only A can produce this Digital Signature
because only A has got knowledge of PR

A. Mulder
driesmulder@kpnmail.nl



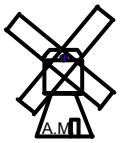
HCC Cryptography Presentation

Preserving Integrity of PU

In both Digital Signatures and Initial DES key distribution it is very important that we can trust the PU's that are used

- If someone can manage to replace a PU being sent from A to B, by his own PU he can attack both protocols
- A receiver of a PU cannot determine if the PU he received is the original unmodified PU from the originator
- Therefore we MUST take special measures to preserve the integrity of the PU
- Most of the complexity of PKA systems is in all the measures that are taken to protect the PUblic keys.

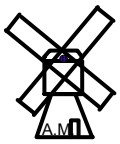
A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

the RSA Algorithm

A. Mulder
driesmulder@kpnmail.nl

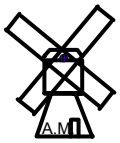


HCC Cryptography Presentation

RSA Encryption/Decryption

- **The RSA algorithm works with NUMBERS instead of working with DATA**
- **Of course numbers can represent data and vice versa, but you might have to convert data to numbers at encryption and numbers to data at decryption**
- **The algorithm is based on mathematical difficulties of specific math problems. (eg. it is difficult to factorize a big number in prime components)**

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Public Key Cryptography / Key Generation (1)

- Generate two BIG Prime numbers P and Q
  In our example we take small numbers just to
  have an easy calculation, but in real life you take ± 500 bits numbers.

  Example: P = 5
  Q = 11

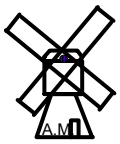
- Multiply the Prime Numbers; This gives the MODULUS N

  P x Q = 55 =====> N = 55

- The Private and the Public key are now of the form: MODULUS and EXPONENT
  where the Public Key exponent is Public and the Private key exponent is kept secret

Public key = N E = 55 E
Private key = N D = 55 D

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Public Key Cryptography / Key Generation (2)

- Calculate FI (P - 1) * (Q - 1) Example: 4 * 10 = 40

- Select the Public Key Exponent E as a ODD number.
  The chosen number must be relatively Prime to (P - 1) * (Q - 1)

  Example: E = 3

- Now calculate the secret exponent D for the Private key, by calculating D

  D should satisfy: 1 = (E * D) MOD ((P-1) (Q - 1))

  so that 1 = (E * D) MOD 40

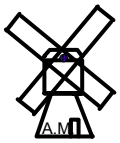
  27 satisfies this, because 1 =(3 * 27) MOD 40

  So the Private Secret Exponent becomes 27

Public key = N E = 55 3
Private key = N D = 55 27

Note: because in real life P and Q are chosen as very big numbers (eg. 1024 bits) it is mathematically very hard to calculate P and Q from a given modulus (eg. from the Public key value)

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

RSA Encryption (using the Public key)

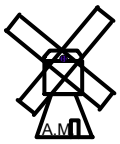
- We first convert our input data to NUMBERS
  In our example we take "ANDRIES" as an example plaintext

  ANDRIES can be converted to: 1 14 4 18 9 5 19

- To Encipher, we take the following steps:
  - Raise number to power of E
  - divide by Modulus
  - Resulting Remainder of division is ciphertext

1^3	= 1	Remainder of 1 / 55 is 1	==== > Result is 1
14^3	= 2744	Remainder of 2744 / 55 is 49	==== > Result is 49
4^3	= 64	Remainder of 64 / 55 is 9	==== > Result is 9
18^3	= 5832	Remainder of 5832 / 55 is 2	==== > Result is 2
9^3	= 729	Remainder of 729 / 55 is 14	==== > Result is 14
5^3	= 125	Remainder of 125 / 55 is 15	==== > Result is 15
19^3	= 6859	Remainder of 6859 / 55 is 39	==== > Result is 39

A. Mulder
driesmulder@kpnmail.nl



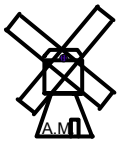
HCC Cryptography Presentation

RSA Decryption (using the Private key)

- To Decipher, we take the following steps:
  - Raise number to power of D (Private key Exponent)
  - divide by Modulus
  - Resulting Remainder of division is recovered plaintext
- We must convert the numbers then backwards to data

1^{27} = 1	Remainder of 1 / 55 is 1	==== > Result is 1, which translates in "A"
49^{27} = 43181145673964365640352930977077280875522488 49	Remainder of 43181145673964365640352930977077280875522488 49 / 55 is 14	==== > Result is 14, which translates in "N"
9^{27} = 581497370030400596903901 69	Remainder of 581497370030400596903901 69 / 55 is 4	==== > Result is 4, which translates in "D"
2^{27} = 134217728	Remainder of 134217728 / 55 is 18	==== > Result is 18, which translates in "R"
14^{27} = 881976397794628113044498441 8304	Remainder of 881976397794628113044498441 8304 / 55 is 9	==== > Result is 9, which translates in "I"
15^{27} = 568151286615952849388122558 59375	Remainder of 568151286615952849388122558 59375 / 55 is 5	==== > Result is 5, which translates in "E"
39^{27} = 909377887614652551975371341130628025063 9479	Remainder of 909377887614652551975371341130628025063 9479 / 55 is 19	==== > Result is 19, which translates in "S"

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

An second example (Key Generation)

- Example: P = 47, Q = 71
- To get the MODULUS N, multiply P and Q --------> N = 3337

  (P - 1) * (q - 1) = 3220
- Select the Public Key Exponent E In our Example we select E = 3
- Now calculate D such that 1 = (D * 3) MOD 3220 -------> D = 2147

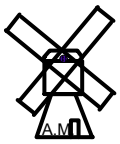
  2147 * 3 = 6441 and 6441 / 3220 = 2 with remainder 1

Public key = N E = 3337 3

Private key = N D = 3337 2147

Note: Extended Euclidean Algorithm

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

A few questions

- Why are Public Exponents 3, 5, 17 , 257 or 65537 attractive ?
- Why is a Random Public Exponent slower ?
- What will happen when I use EXP3 and I have a very short plain-text ?
- How would you do the padding ?
  - Random
  - ZERO-PAD
  - Other

A. Mulder
driesmulder@kpnmail.nl

HCC Cryptography Presentation

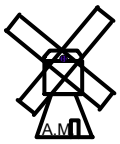
Chinese Remainder Theorem

- Invented by chinese mathematician sun tzu
- Same result, but calculated in another way
- Almost 2 times as fast for most Private key operations
- Requires the Private key to be pre-determided for CRT
  (key has other format and different key-material)

  CRT Key stores p,q, dp, dq and U
  ME key stores n, e and d

- You need to make a choice before you generate the key pair
- Converting a CCA key from ME to CRT or from CRT to ME
  requires a UDX
  - look at Wikipedia and search Chinese_remainder_theorem

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

ECC Diffie Hellman Key Exchange

A. Mulder
driesmulder@kpnmail.nl

HCC Cryptography Presentation

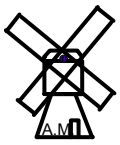
The Basics (very simplified)

- Elliptic Curve is a newer variant of A-symmetric Cryptography
- A-symmetric cryptography uses "trapdoor" functions
- The trapdoor function in ECC is stronger than the RSA trapdoor function
  RSA's trapdoor is factorizing a product of 2 big primes)
- As a result, ECC is stronger than RSA (at equal key size) or
  ECC can use shorter keys
- At same strength, ECC should perform better than RSA
  (see table on next page)

- Diffie Hellman is a protocol for key-exchange
- DH was traditionally done with big primes (like RSA)

- Now (since CCA rel 4.3) we have ECC-DH Elliptic Curve Diffie Hellman in CCA

A. Mulder
driesmulder@kpnmail.nl



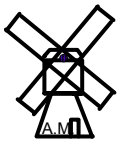
HCC Cryptography Presentation

Equivalent key strength

Symmetric Key Length	Equivalent RSA or DH Key Length	ECC Key Length
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

source: http://www.nsa.gov/business/programs/elliptic_curve.shtml
The case for Elliptic curve

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

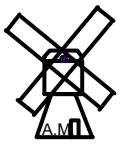
"Normal" DH vs ECC DH
Relative computation cost

Security level (bits)	ECC
80	3 : 1
112	6 : 1
128	10 : 1
192	32 : 1
256	64 : 1

The stronger the key that you want to exchange,
the bigger the advantage of using ECC-DH protocol.

source: **http://www.nsa.gov/business/programs/elliptic_curve.shtml**
The case for Elliptic curve

A. Mulder
driesmulder@kpnmail.nl

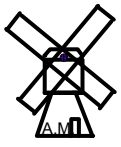


HCC Cryptography Presentation

PIN Processing

Ir. Andries Mulder
Mulder Training & Consultancy
the Netherlands
gsm. +31 651 71 40 00
tel. +31 544 37 40 37
email driesmulder@kpnmail.nl

A. Mulder
driesmulder@kpnmail.nl



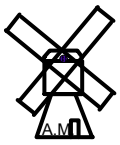
HCC Cryptography Presentation

PIN Processing

PIN Verification

- A bank client withdraws Cash out of ATM
- Easy to understand transaction
- Secrecy of PIN is obvious

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

An example:

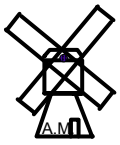
- A bank's customer complains about a PIN withdrawal
- ***"I didn't do that cash withdrawal"***

How can the bank prove that its not their fault ?

- By proving that no single employee can get a customer's PIN value
- By using certified cryptographic hardware
- By having strict procedures in place
  - To protect keys (that protect PIN's)
  - To protect PIN's

In a dispute in court, they should be able to win the case

A. Mulder
driesmulder@kpnmail.nl



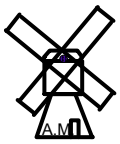
HCC Cryptography Presentation

How is the PIN usually derived ?

PINs are calculated as follows.

- **Take the last five significant digits of the account number, and prefix them by eleven digits of validation data.**
- **These are often the first eleven digits of the account number. They could also be a function of the card issue date.**
- **In any case, the resulting sixteen digit value is input to an encryption algorithm (TDES)**
- **The result is then Decimalised (mapping on 0,1,...9 digits) Result of decimalising is the "Natural PIN"**

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

How is the PIN usually derived ? Example

Account number PAN: 4506602100091715
Last 5 digits: 91715
Validation data: 88070123456
Concatenate Val.data and last 5 Digits of PAN
Data input to DES algorithm: 8807012345691715
PIN key input to DES algorithm: FEFEFEFEFEFEFEFE
Output of DES algorithm: A2CE126C69AEC82D
Take first 4 digits (nibbles) and decimalise

First four digits decimalised --> 0224 A=(1)0 2=2 C=(1)2 E=(1)4
Natural PIN = 0224 (question: Is this decimalisation OK ?)

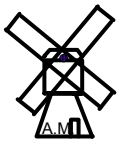
Natural PIN = 0224
Offset: = 6565
Customer PIN: = 6789

Note: Offset is only used if the customer has a selectable PIN

Decimalisation table (example)

0 -> 0
1 -> 1
2 -> 2
3 -> 3
4 -> 4
5 -> 5
6 -> 6
7 -> 7
8 -> 8
9 -> 9
A -> 0
B -> 1
C -> 2
D -> 3
E -> 4
F -> 5

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Decimalisation of the PIN

Example: The raw PIN data is **DC 88**
Now we assume here in this example the following decimalisation table:

decimalisation table = 6028 0786 0808 3644
0123 4567 89AB CDEF

On offset 0 we have a 6	On offset 1 we have a 0
On offset 2 we have a 2	On offset 3 we have a 8
On offset 4 we have a 0	On offset 5 we have a 7
On offset 6 we have a 8	On offset 7 we have a 6
On offset 8 we have a 0	On offset 9 we have a 8
On offset 10 (A) we have a 0	On offset 11 (B) we have a 8
On offset 12 (C) we have a 3	On offset 13 (D) we have a 6
On offset 14 (E) we have a 4	On offset 15 (F) we have a 4

DC 88 now translates as follows to a natural PIN

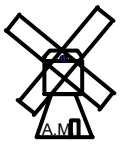
D ---> 6
C ---> 3
8 ---> 0
8 ---> 0

So the PIN becomes 6300

Decimalisation table used here

0 -> 6
1 -> 0
2 -> 2
3 -> 8
4 -> 0
5 -> 7
6 -> 8
7 -> 6
8 -> 0
9 -> 8
A -> 0
B -> 8
C -> 3
D -> 6
E -> 4
F -> 4

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

PIN blocks example of an **ISO0** PIN block:

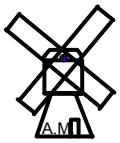
```
PAN = 33 61 44 97 ØØ ØØ ØØ ØØ  (8 bytes)
PIN = 12 34  ---> PIN length = 4

Interm. PIN Bl. = Ø4 12 34 FF FF FF FF FF
PAN Block       = ØØ ØØ 33 61 44 97 ØØ ØØ
Clear PIN block = Ø4 12 Ø7 9E BB 68 FF FF
```

The PIN block is always encrypted

$$E_K(PB)$$

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Verification of the PIN

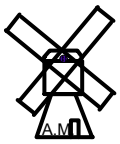
- PIN calculation method

  After receiving the encrypted trial PIN block the system calculates the correct PIN, based on PAN and other info (dec. table etc.) , decrypts the trial PIN block and compares the two clear PIN values

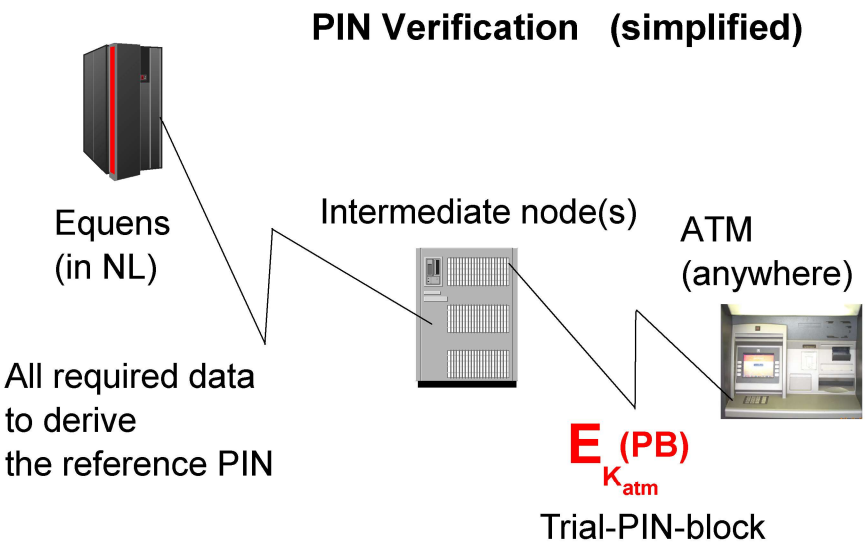
- PIN Database method (use reference PIN)

  After receiving the encrypted trial PIN block the system retrieves the encrypted reference PIN block from a database and decrypts both PIN blocks and compares the two values

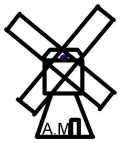
A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation



A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

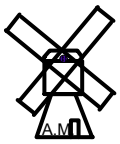
PIN Verification (simplified)

INSIDE HSM at bank-side:

- Decrypt trial PIN block $E_{K_{atm}}^{(PB)}$
- Derive / Calculate reference PIN
- Compare decrypted trial PIN and reference PIN
- Return YES or NO to caller

No clear PIN ever shows up

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Example HSM PINVER function (simplified)

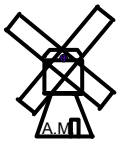
Encrypted_PIN_Verify (PIN_encr_key,
PIN_ver_key,
PIN_info,
PAN_info,
Encrypted_trial_PIN_block
Result)

To decrypt trial PIN

To derive reference PIN

The function checks the correct key type

A. Mulder
driesmulder@kpnmail.nl



HCC Cryptography Presentation

Key Separation

What would happen if PIN_DECIPHER_key1 could be used in Decipher function ?

BIG TROUBLE CAN OCCUR

BECAUSE $E_{K_{atm}}(PB) \longrightarrow PB$

Therefore we MUST use key separation, so that a PIN-key never can exist as a decipher-key

A. Mulder
driesmulder@kpnmail.nl