



# **Latest generation of crypto smart cards**

## **Capabilities and applications**

### **Talk + Demonstration**

**31 May 2003**

**Mr. Lin Yih**

- **Director, Digital Applied Research and Technology Pte Ltd**
- **Chair, Cards and Personal Identification Technical Committee**
- **Executive Consultant, Netrust Pte Ltd**
- **MIT Education Councilor**

**Email: [dartpl@singnet.com.sg](mailto:dartpl@singnet.com.sg)**

**Tel: 67787684, 67787969**



# Crypto Smart Card - Intro

- **Many types of smart card: simple memory (e.g. for pay phones), 8/16-bit CPU with software encryption (e.g. PS Card, MPCOS), 8/16-bit CPU with cryptographic hardware**
- **Cryptographic co-processor: DES, 3-DES, RSA, Elliptic Curve, AES (announced)**
- **Sometimes not all crypto hardware can be squeezed into one chip (read datasheets carefully)**



# Crypto Smart Card - Speed

- **DES by software: 50 msec (typical)**
- **DES by hardware: 31 usec**
- **3-DES by software: 150 msec (typical)**
- **3-DES by hardware: 49 usec**
- **RSA-1024 by software: not practical ☹**
- **RSA-1024 by hardware: 1 sec (no CRT, no turbo)**
- **ECC-160 by software: 2 sec (or more)**
- **ECC-160 by hardware: 1 sec (+/-)**
- **\* note: ECC has more diversity in implementation (here  $GF(2^m)$ )**



# Crypto Smart Card – Why?

- **Is PC encryption software faster? Or crypto smart card? Answer: PC is faster!**
- **So why not use PC? Answer: then “secrets” have to move in and out of smart card. Transaction can be wire-tapped.**
- **Transaction can be protected by encryption, but implies encryption secret key stored on both smart card and PC. Complication....**
- **Fact: PS Card (not crypto smart card) moves “secrets” in and out of smart card, protected by encryption.**



# Crypto Smart Card – Example

- **Infineon SLE66CX322P**
- **(\*\* Philips has its own FameX, FameXE, SmartMX series, other companies too)**
- **ACE (hardware registers for RSA-1024; 2048 by software)**
- **DDES-EC2 (DES, 3-DES, Cipher Block Chaining, ECC up to 192 bits)**
- **64KB ROM, 32KB EEPROM, 2048KB XRAM, 256 byte IRAM**
- **CURSE (current scrambler to handle DPA, SPA)**
- **RWS (random wait state to handle timing attacks)**



# Crypto Smart Card – 66CX322P

- **RNG (hardware noise generator)**
- **UART (for up to 115K baud I/O)**
- **PLL (internal clock multiplier, x4 external 3.58MHz, or max 15 MHz)**
- **MMU (memory management unit)**
- **MED (memory encryption unit)**
- **Active Shield (against external probing)**
- **Security Logic (low frequency sensor, high frequency filter, low/high voltage sensor)**
- **0.22 um CMOS process**
- **Note: DES-EC2 available for contactless (no RSA, yet)**
- **Note: there is 88CX720P, RISC based (not 8051) that runs up to 66MHz. But RISC needs faster MHz compared with CISC.**



# Crypto Smart Card – PKI

- **Crypto Smart Card for DES/3DES? Only Speed advantage. *Symmetric* key encryption means key must move in/out.**
- **Most useful for *Asymmetric* key crypto like RSA and ECC.**
- **Card generates *secret key* and *public key*. Then exports public key only. Secret key never leaves the card.**
- **! Catch: if you need *key escrow / recovery*, then must export secret key! So why use crypto smart card? Still useful because export only for key escrow, not every operation!**



# Crypto Smart Card — RSA-1024 key

- **Generate random primes P, Q (each about 512 bits)**
- **$N = P \times Q$  (N is public, 1024 bits)**
- **Generate random E (public, typically 32 bits)**
- **Compute D by Extended Euclidean algorithm where  $(E \times D) \bmod ((P-1) \times (Q-1)) = 1$**
- **D is secret, 1024 bits**
- **Destroy P, Q. Export E, N. Keep D**
- **Note: to prevent Pollard p-1 attack, should ensure (P-1) and (Q-1) have large prime factors**
- **Can use probabilistic prime testing to generate P, Q**



# Crypto Smart Card – RSA encrypt

- **Cipher = (Plain)<sup>E</sup> mod N**
- **Plain = (Cipher)<sup>D</sup> mod N**
- **Example: to encrypt a message for Recipient, use Recipient's public E and N. Only Recipient's secret D can decrypt.**
- **Note: Plain must be in chunks of 128 bytes (1024 bits = 128 bytes). Each chunk value must be smaller than N**
- **Note: memory size: N (128 bytes), E (4 bytes), D (128 bytes). Total = 260+ bytes (in a smart card)**
- **Note: since E is small (4 bytes), encryption is fast, but decryption is slow (relative)**



# Crypto Smart Card – RSA sign

- To sign data, first compute a *hash* or *message digest* (e.g. MD5) for the data
- MD5 produce 16 byte output; SHA-1 produce 20 bytes
- Signature = (Digest)<sup>D</sup> mod N
- Example: to sign a message, use Sender's secret D and public N. Use Sender's public E to decrypt and verify
- Summarize: to Sign and Encrypt an email, Sender needs own D (secret) and N (public) to sign, plus Recipient's E (public) and N (public) to encrypt
- Similarly, a complementary set for Verify and Decrypt
- Note: since D is large, signing is slow (relative) while verification is fast



# Crypto Smart Card – RSA CRT

- Chinese Remainder Theorem (CRT) method helps speed up  $D$  calculation. Typically 1/3 time.
- During Key Generation, compute  $D_p$ ,  $D_q$ , and  $U$
- $D_p = D \bmod (P-1)$
- $D_q = D \bmod (Q-1)$
- Find  $U$  where  $(P \times U) \bmod Q = 1$
- Then decryption can be done without  $D$ , using  $D_p$ ,  $D_q$ ,  $U$ ,  $P$ ,  $Q$  only (details omitted here)
- Note: if  $N = 1024$  bits, then  $D_p$ ,  $D_q$ ,  $U$  are all 512 bits.  
Total =  $N + E + D + P + Q + D_p + D_q + U = 580+$  bytes on smart card (actually  $D$  can be discarded)



# Crypto Smart Card — ECC vs RSA

- ECC details not covered by this talk
- ECC is good for signature, not for encryption. RSA is good for both.
- 160-bit ECC = 1024-bit RSA, about 220-bit ECC = 2048 (\* based on Menezes paper)
- ECC signature is more compact: 160-bits = 20 bytes; 1024-bit = 128 bytes
- ECC key size is also smaller
- Small size = short data transmission time
- Time is always important for *contactless* operations
- ECC more effort to standardized (must choose curve), but standards exists (ECDSA)



# Crypto Smart Card – CA?

- If 2 people can exchange public keys (i.e. E and N) face to face, with trust, then there is no need for a Certification Authority (CA)
- If not, alternative is to get your E, and N signed by a CA (CA's secret D and public N), then published the signed E and N on a *directory server* (complicated? 😊)
- Your signed E, D, name, ID, email address, etc. is your *digital certificate* (the data in a cert is *plain*)
- People will get your E and N from the directory server
- How do they know it is genuine? They verify the signature using the CA's *public* key!
- Where do they get CA's *public* key? From CA's web server, or pre-installed with O/S (e.g. Windows setup)



# Crypto Smart Card – Deployment

- “Modern” Standard Deployment Scenario: John has a crypto smart card, generated E, D, N on card. E, N can be read out from smart card.
- John goes to a CA (could be a Bank with CA capability), then CA reads out E, N and produces a digital certificate
- Digital Cert can be published. It can be stored on the smart card as a *write-once read-only* file (!)
- CA does not have John’s private key D (or  $D_p$ ,  $D_q$ , U, P, Q)
- To verify the digital cert, the terminal or PC must have the CA public key
- To prevent *illegal substitution* of the CA public key, the key can (should) be stored in a SAM (secure access module)
- Note: CA Public Key typically exists as a *self-signing cert*
- If SAM is a smart card, then let 2 smart cards talk direct 😊



# Crypto Smart Card – Caution

- **Data I/O for smart card is “slow”**
- **Example: a digital cert is 2048 bytes**
  - **To read out at 9600 baud: minimum 2 seconds (assuming one single big packet, unlikely)**
  - **To read out at 115200 baud: minimum 178 msec (“eternity” for contactless transit needs)**
- **Cannot feed a “photo file” into a smart card for it to sign (too much I/O time)**
- **MD5, SHA-1 currently done using software on smart card. Recommend using PC to compute instead**
- **Beware of excessive command / response overhead (ISO standard not much help here ☺ )**



# **Crypto Smart Card**

**Thank you**

**Open for Demo and Q&A**