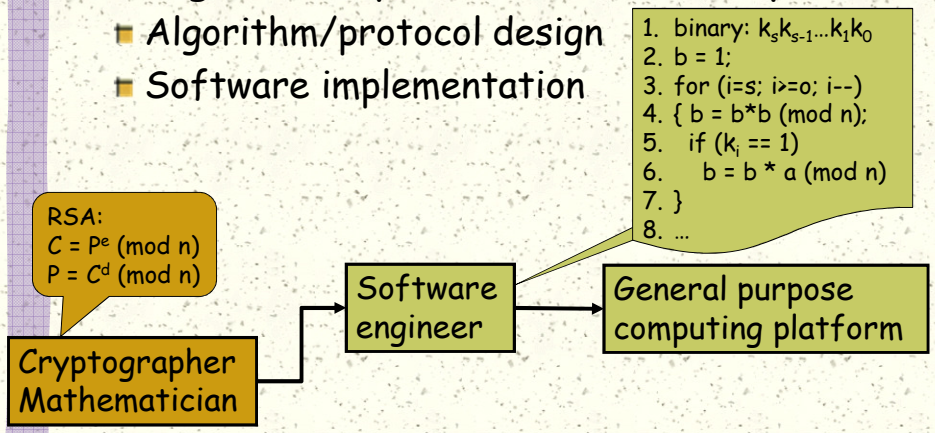
Side Channel Attacks -- Memory Vulnerabilities and Cache Attacks

Cybersecurity Specialization
-- Hardware Security

Development of a Cipher

- Design and implementation of a cipher
  - Algorithm/protocol design
  - Software implementation

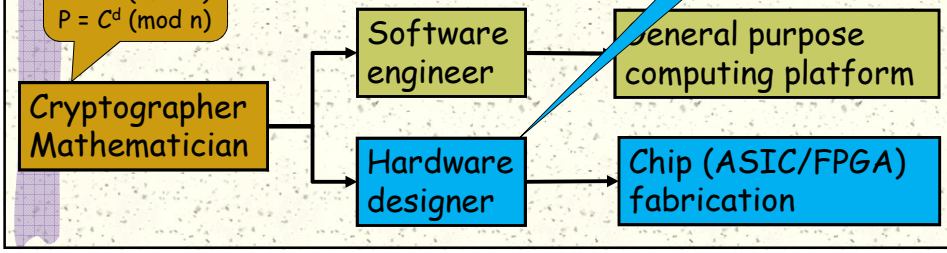


Development of a Cipher

- Design and implementation of a cipher
  - Algorithm/protocol design
  - Software implementation
  - Hardware implementation

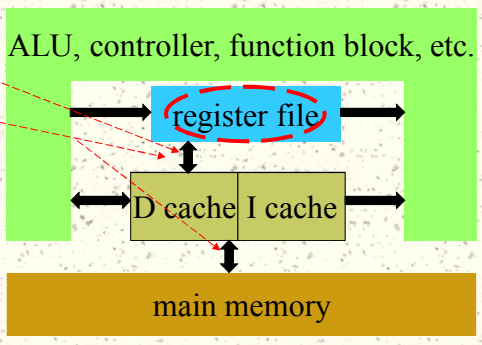
```
module A (clk, reset, in, out)
 input clk, reset, in;
 output out;
 reg out;
 reg [1:0] cState, nState;
 always @(in or cState) begin
   case (cState)
   ...
```

RSA:
$C = P^e \pmod n$
$P = C^d \pmod n$



Execution and Vulnerabilities

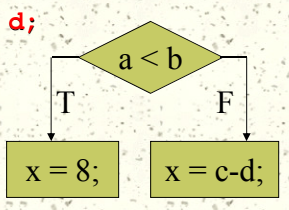
- Execution flow
  - Assume that secret data is stored in register file during execution
- Vulnerabilities
  - Memory load
  - Memory store
  - A&L operation
    - Ex.: x = x*y
  - Control flow
    - Ex.: if-else



Example: C → Assembly

- C:

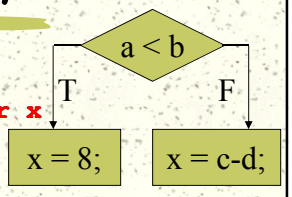
```
if (a < b) {x = 8;} else x = c - d;
```



- Assembly:

```
; compute and test condition
  ADR r4,a          ; get address for a
  LDR r0,[r4]       ; get value of a
  ADR r4,b          ; get address for b
  LDR r1,[r4]       ; get value for b
  CMP r0,r1         ; compare a < b
  BGE fblock        ; if a >= b, branch to false block
```

Example: C → Assembly




```
; true block
  MOV r0,#8         ; generate value for x
  ADR r4,x          ; get address for x
  STR r0,[r4]       ; store x
; false block
fblock ADR r4,c     ; get address for c
  LDR r0,[r4]       ; get value of c
  ADR r4,d          ; get address for d
  LDR r1,[r4]       ; get value for d
  SUB r0,r0,r1      ; compute c-d
  ADR r4,x          ; get address for x
  STR r0,[r4]       ; store value of x
...
```

Memory load

Memory store

```
    1101,1001
-)    10,0110

    1101,1011
-)        10
```

Control flow: 3 instructions vs. 7 instructions

Cache Attacks on Ciphers

- Cache attacks may happen when
  - Look-up tables are used in the cipher
  - Cache memories are used in the processor
- Cache attack methods
  - Trace-driven: adversary monitors the cache activity (Miss or Hit) for each memory access.
  - Time-driven: monitors the execution time of the encryption, large amount, can be done remotely.
  - Access-driven: the cache is shared by other processes until it is evicted.

Trace-Driven Attack: Example 1

- A DES implementation accesses 8 S-boxes per round, each S-box is implemented with a table. The usage of each S-box depends on the input and the key.
  - Cache miss/hit on each of the LUTs
    - First round (all cache miss): MMMM, MMMM
    - If second round: MHMM, HMMM
    - Collisions on the 2nd and 5th tables between the two rounds of memory accesses.

Trace-Driven Attack: Example 1

- The usage of each S-box depends on the input and the key.
- Attack
  - Use a random input for the first round
  - Select input for the second round to have an H on the target table/S-box
  - Filter out invalid keys (not causing the H)
  - Repeat till the key becomes unique
- The 56-bit DES key is revealed with 2^{10} inputs and a key search space of 2^{32}.

D. Page. Defending Against Cache-based Side-Channel Attacks. 2003.

Trace-Driven Attack: Example 2

- Attack based on Induced Cache Miss
  - Encrypt an input x
  - Invalidate a cache line occupied by the S-box/table
  - Encrypt the input x again and monitor cache miss/hit or power consumption (a miss implies that the invalidated cache line is accessed)

G. Bertoni, et al. AES Power Attack based on Induced Cache Miss and Countermeasure. 2005.

Timing Attacks on Ciphers

- Requirements for successful timing attacks
  - Execution time variation on some operations
  - The variation depends on the secret key
  - The execution time variation is measurable
  - The number of measurements depends on the amount of information from measurements
  - A synchronization signal for the start/completion of the operations
  - Design of the crypto-system is known

Paul Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. 1996.

Cache Attacks: Countermeasures

- Application level
  - No look-up tables
  - Small tables
  - Cache warming
  - Data-oblivious memory access pattern
  - Run time control
    - Constant
    - Random delay
- Hardware level
  - Non-cached memory access
  - Specialized cache design
  - Special instructions to the ISA
  - Prefetching