A Practical Fault Attack on Square and Multiply

Jörn-Marc Schmidt Christoph Herbst

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria

{joern-marc.schmidt,christoph.herbst}@iaik.tugraz.at

Motivation

- Square and multiply is a common stragegy for implementing modular exponentiation
- Modular exponentiation is used in public key cryptography
- RSA is based on modular exponentiation
- Fault attack on RSA implementations without CRT

Other Modular Exponentiation Methods

- Left-to-right square and multiply
- Right-to-left square and multiply
- k-ary exponentiation
- Sliding window method
- Montgomery powering ladder

Square and Multiply

Function 1 Left-to-Right Square and Multiply Algorithm

Input: Message m, Exponent $e = (e_t, \ldots, e_0)_2$, Modulus N

$R = 1$
for $i = t$ downto 0 **do**
  $R = R \cdot R \mod N$
  if $e_i = 1$ **then**
    $R = R \cdot m \mod N$
  end if
end for
return R

Recent Work

Different attacks on square and multiply - assuming

- Bit flip
  - Dan Boneh et al. (1997)
  - Feng Bao et al. (1997)
  - Marc Joye et al. (1997)
- Safe errors
  - Sung-Ming Yen and Marc Joye (2000)
- Random fault in intermediate value
  - Michele Boreale (2006)

Our Fault Model

Manipulation of the program flow

- Skip instruction
- Not always successful
- Motivated by spike attacks

Skip a Squaring Operation

- Exponent $e = (e_t, \ldots, e_0)_2$, leading zeros are neglected
- $(t - k + 1)$-th square operation skipped
  $\Rightarrow Sig_k$, $k \in \{0, \ldots, t\}$
- $Sig_t = Sig$ as $R = 1$

$$Sig_k = \prod_{i=k+1}^{t} m^{e_i 2^{i-1}} \cdot \prod_{i=0}^{k} m^{e_i 2^i} \mod n.$$

Iterative Attack

For $k = 0$

$$Sig = \begin{cases} (Sig_0)^2 \mod n & \text{for} \quad e_0 = 0 \\ (Sig_0)^2 \cdot m^{-1} \mod n & \text{for} \quad e_0 = 1 \end{cases}$$

For $k \in \{1, \ldots, t-1\}$

$$Sig_k = \begin{cases} Sig_{k-1} & \text{for} \quad e_k = 0 \\ m^{2^{k-1}} \cdot Sig_{k-1} \mod n & \text{for} \quad e_k = 1 \end{cases}$$

Results in $(1, e_{t-1}, \ldots, e_0)$.

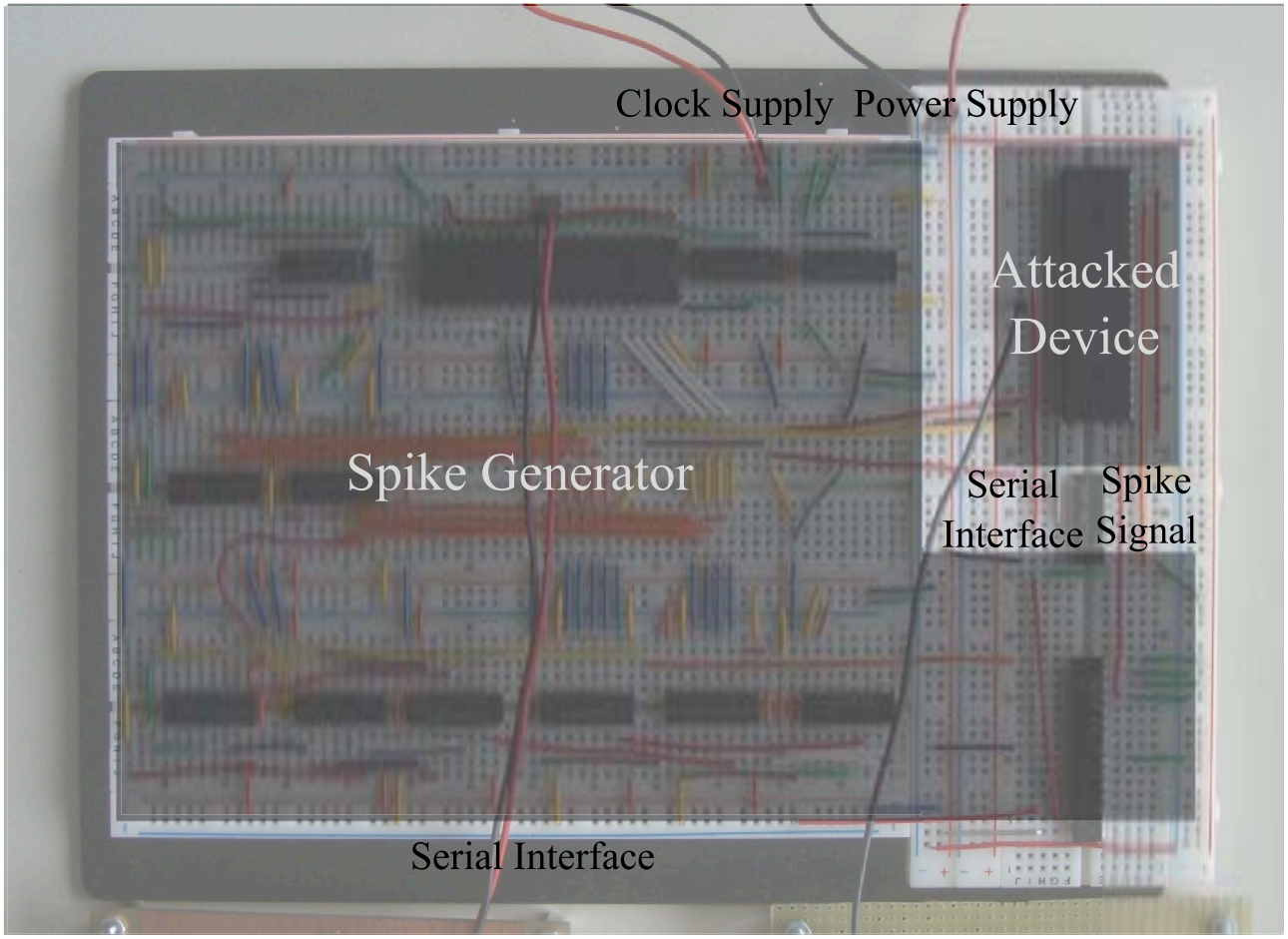
Device Under Test (DUT)

- AVR microcontroller
- Straight forward left-to-right square and multiply
- Montgomery for modulo multiplication
- Spikes in the power supply

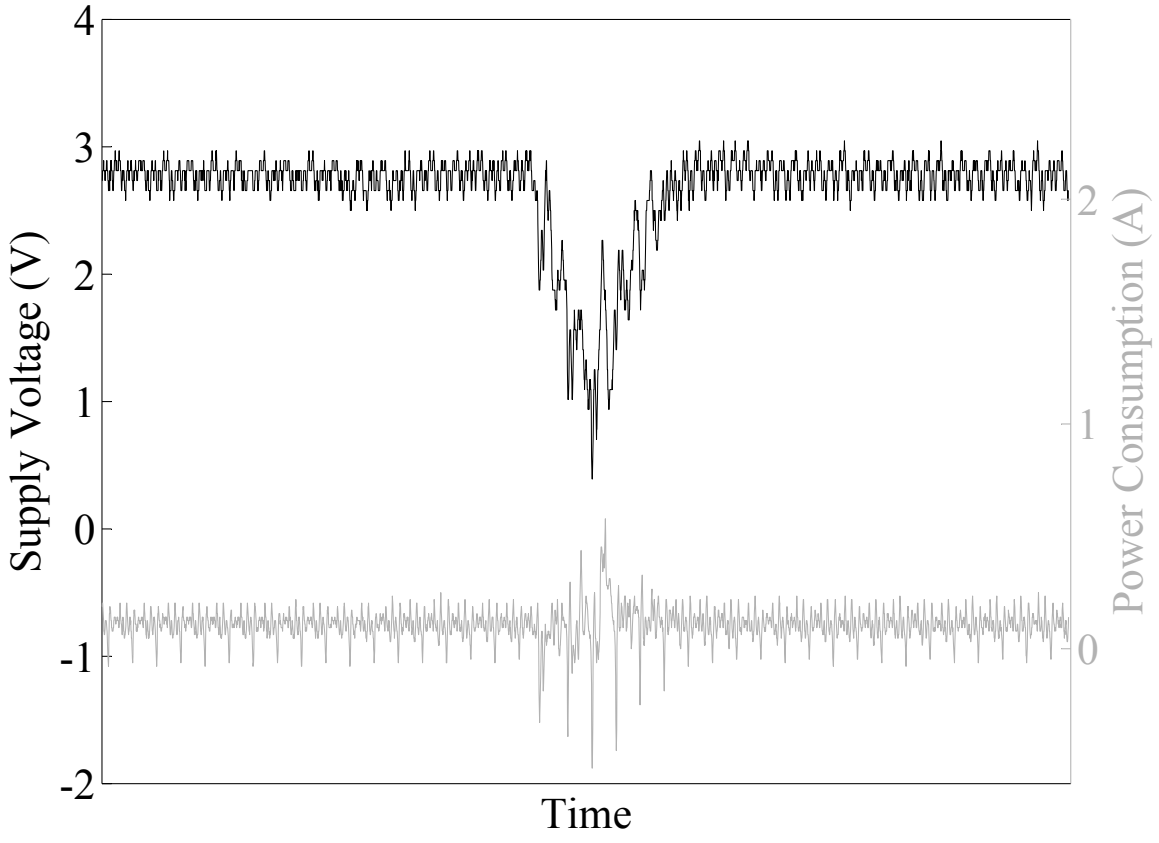
Spike Generation

- Circuit Board for DUT and Spike generation (low cost)
- Controlled by PC over serial interface
- Spike offset precision 0.5 clock cycles
- Spike length 0.5-5 clock cycles

Setup for the performed spike attack



Spike (black) and power consumption (gray)



Searching the right spike parameters

- *Right* spike positions and length unknown
- Calculate expected values for $e_k = 0$ and $e_k = 1$
- Sweep over the whole computation starting from the end
- If e_k found, calculate expected values for e_{k+1}

Problems

- Fine sweep may lead to double detections
- Store precomputed values indicating a 1 as long as 0 e detected
- Compare all following results to these values and repair detected exponent if match found
- Another Solution: Use power trace to guess positions
  $\Rightarrow$ requires more knowledge and equipment
- Afterwards add a 1 to the detected exponent
- Test result by calculating a signature

Attack in the presence of DPA Countermeasures

- Square and Always Multiply
- Message Blinding
- Exponent Blinding
- Further Countermeasures

Outlook

- Mount attack on ECC double and add
- Attack Montgomery powering ladder in modified fault model
- Investigate existing countermeasures in more detail

Conclusion

- We presented a new attack on square and multiply
- Based on program flow manipulation
- Possible to check whether or not fault injection was successful
- Practical implementation at low cost

Thank you for your attention. Questions?

Jörn-Marc Schmidt Christoph Herbst

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria

{joern-marc.schmidt,christoph.herbst}@iaik.tugraz.at