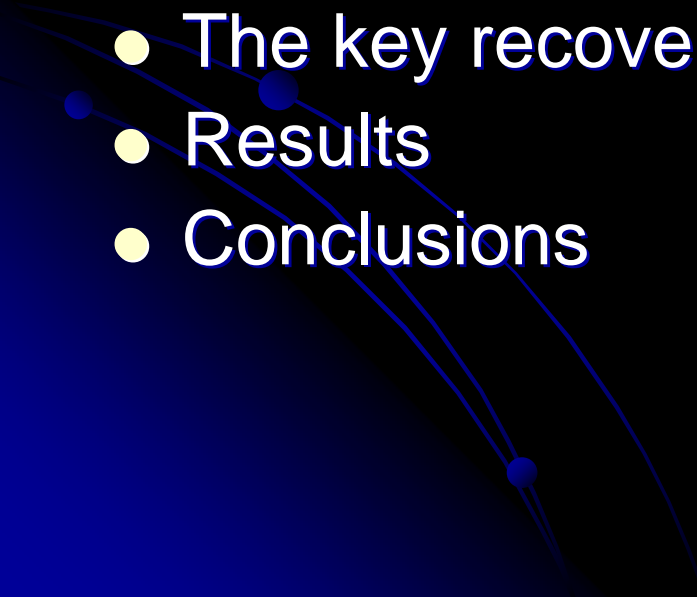


An Advanced Timing Attack Scheme on RSA


Rudolf Tóth, Zoltán Faigl, Máté Szalay and Sándor Imre

Department of Telecommunications
Budapest University of Technology and Economics
Email: toth.rudolph@gmail.com, zfaigl@mik.bme.hu
szalay@mlabdiat.hit.bme.hu and imre@hit.bme.hu

Outline

- Motivation
 - What are side-channel attacks?
 - Concept of the attack: a null hypothesis test
 - Behavior of the test function
 - The key recovery algorithm
 - Results
 - Conclusions
- 

Motivation

- Security mechanisms rely on crypto primitives
 - Cryptographic algorithms safe in theory, deceptive signs on implementation level:
 - can be exploited by side-channel attacks
 - Goal: establish a new key recovery scheme
 - supposed that the attacker gathered information that correlates with coding time
 - Proof-of-concept: key recovery of an RSA implementation known to be using Montgomery multiplication
- 

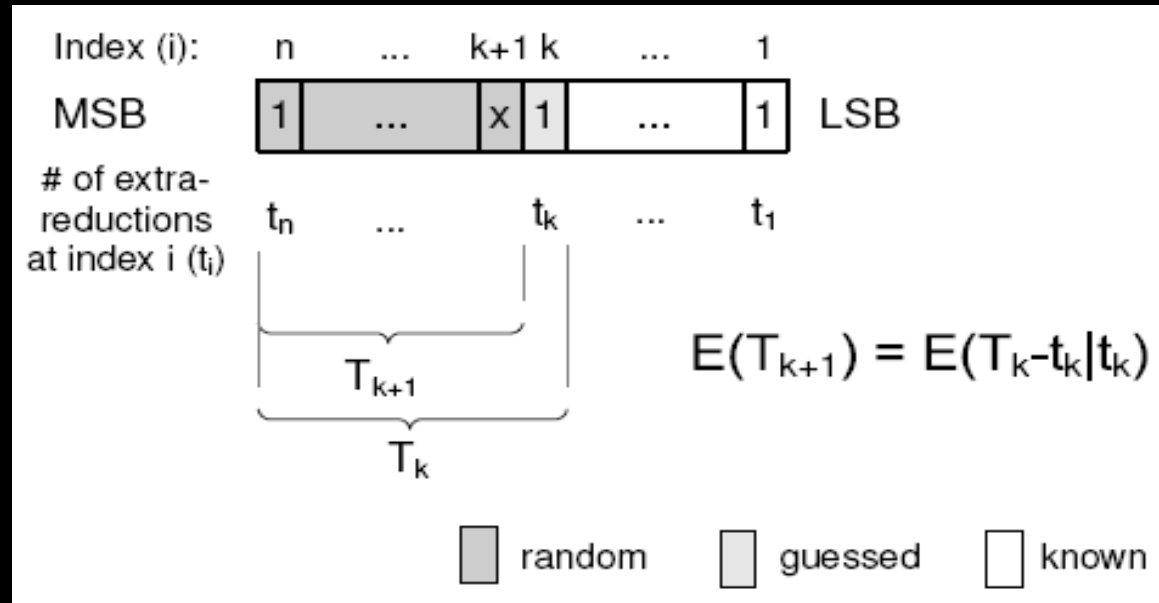
Side-channel attacks

- Exploit information leaked out on implementation-level, e.g.:
 - Error analysis (error, msg feedbacks)
 - Performance analysis (CPU load, power consumption)
 - **Timing attacks** (duration of operations)
- Attacker:
 - knowledge of implementation details (Kerckhoffs' principle)
 - collected samples of
<input messages, side-channel information >
→ Key recovery

Concept of the key recovery

For each msg the attacker

- knows the total time
- guessed well the key bits from LSB to k-1
- can measure the coding time until bit k-1
- calculate the remaining time T_k (from bit k to n)



Recovery of next key bit:

- makes a guess of key bit k
- measures the coding time t_k in round k
- calculates the remaining time (from bit k+1 to n): $T_{k+1} = T_k - t_k$
- If the guess was good t_k is correct, T_k is correctly reduced

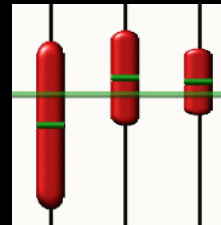
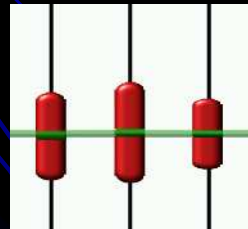
Null hypothesis: $E(T_{k+1}) = E(T_k - t_k | t_k \text{ is good}) \rightarrow$ ANOVA test (F values)

What is Analysis of Variance (ANOVA)?

- Statistical method to test if there is a significant difference between the means of groups
- Null hypothesis: group means are equal

$$E (T_{k+1} | t_k=0) = E (T_{k+1} | t_k=1) = E (T_{k+1} | t_k=2)$$

- F-value: indicator of the difference
 - F function tests ~ inter-group / intra-group variance
 - E.g. $F_1 < F_2$

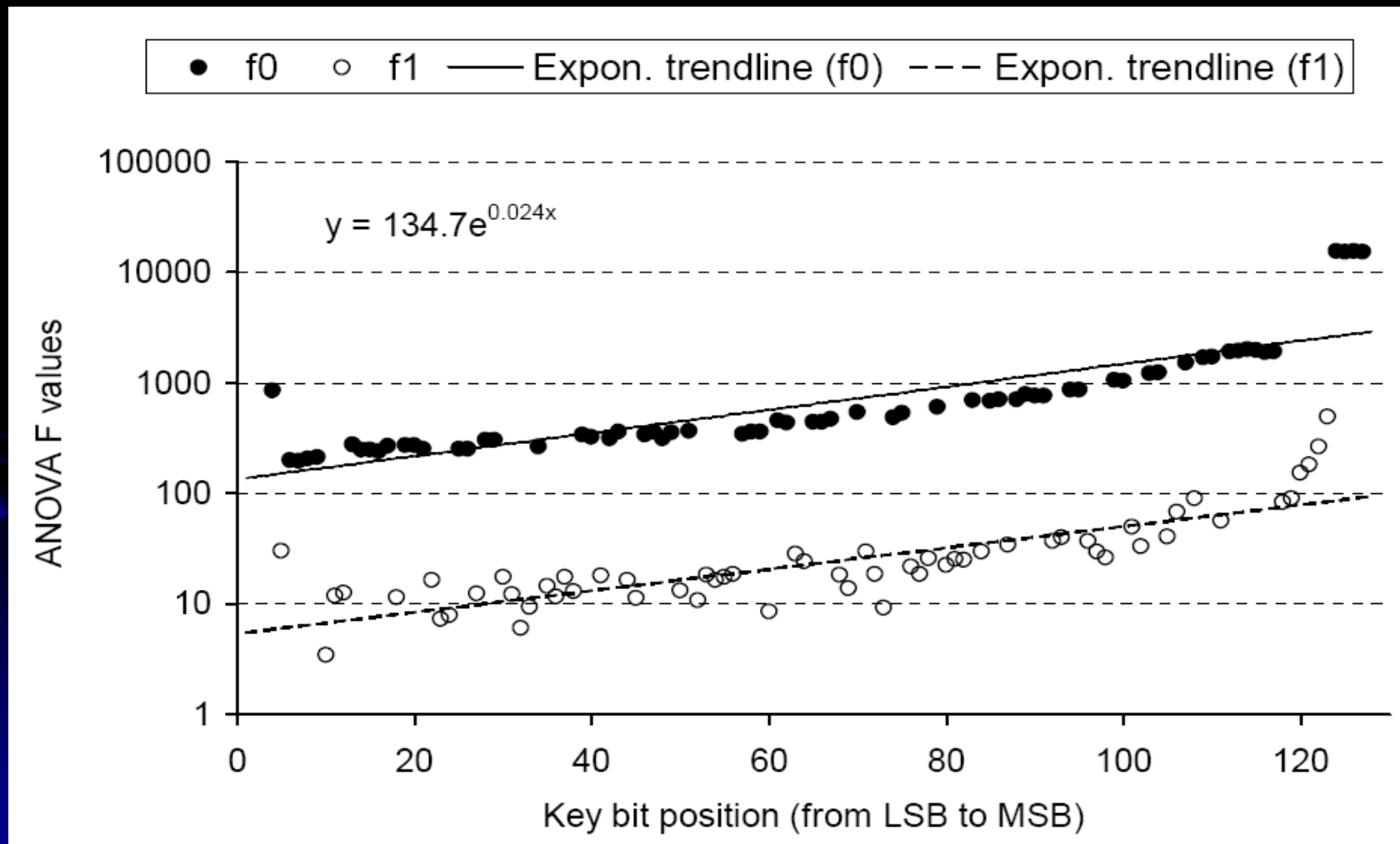


RSA with Montgomery multiplication

- Aim: test our concept
 - Can F function used as indicator? Analyze F with known keys
- We implemented RSA with Montgomery multiplication,
 - Measured the occurrences of events which are supposed to correlate with encryption time
 - For each key bit,
 - it happens 0 times, if the key bit is 0,
 - it happens 0, 1, or 2 times if the key bit is 1, depending on the key and the message that is encrypted

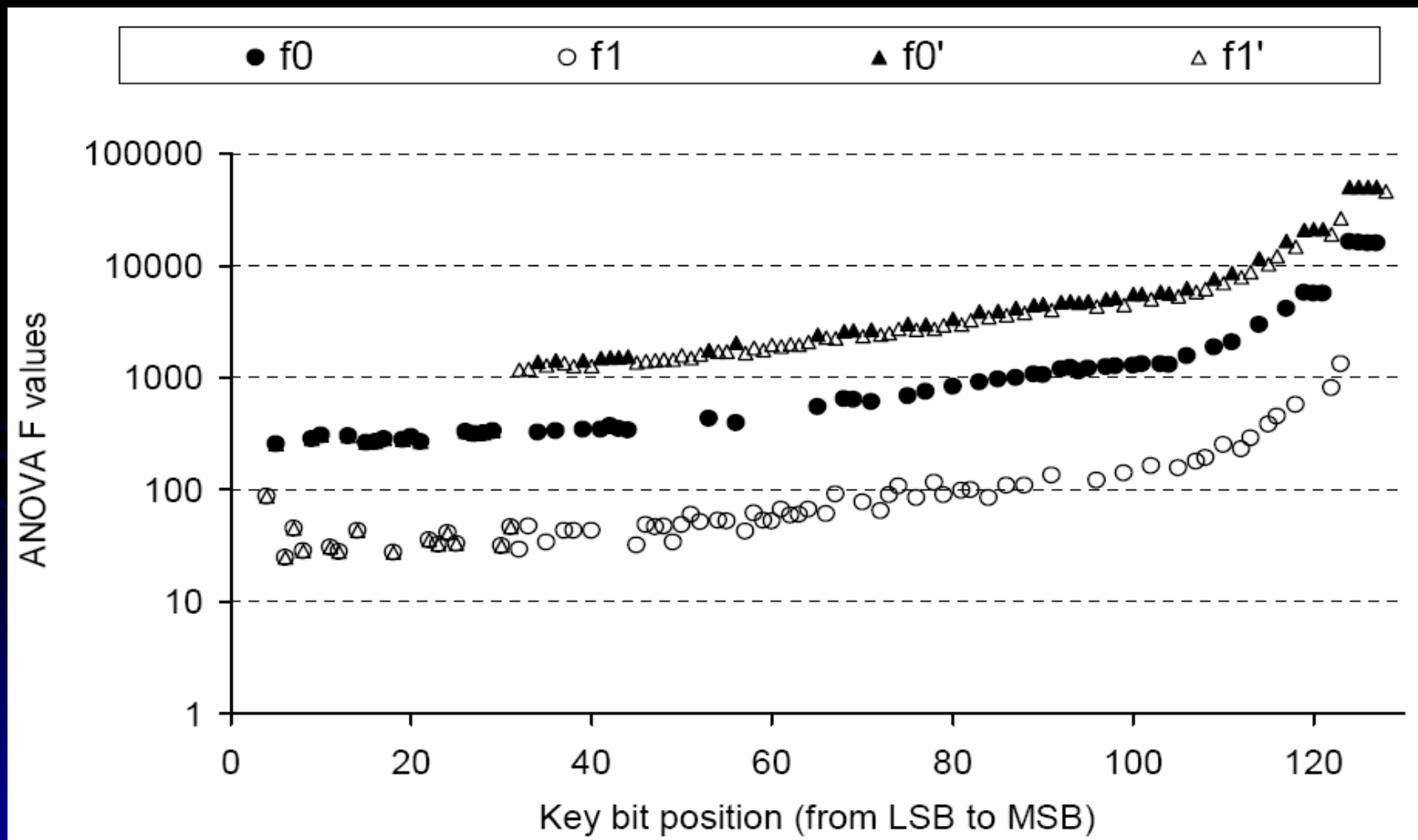
Analysis of F values when guessing key bits

Correct key bit guesses using 100000 samples



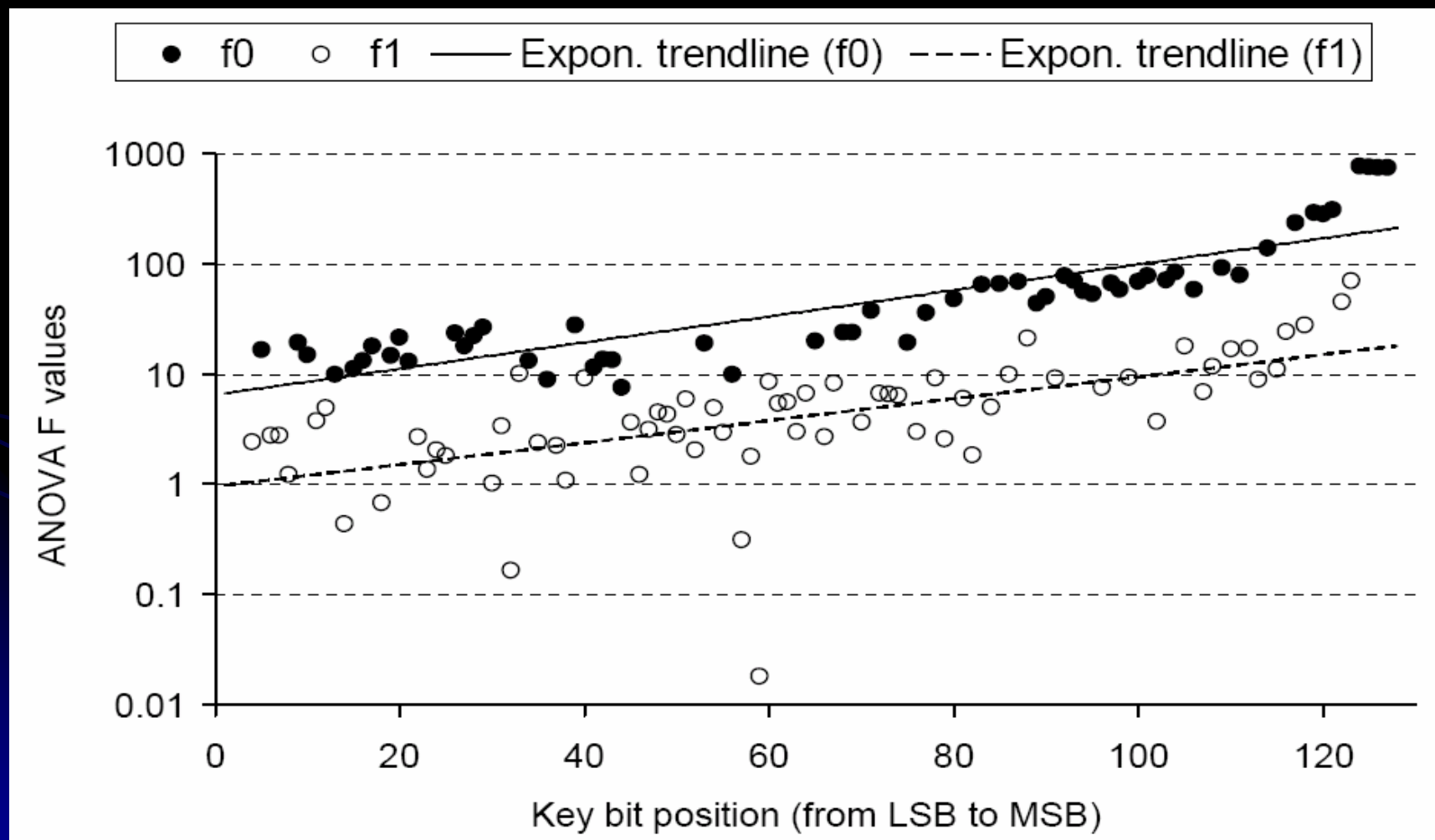
Analysis of F values when guessing key bits

Incorrect key bit guess at key bit position 30 using 100000 samples



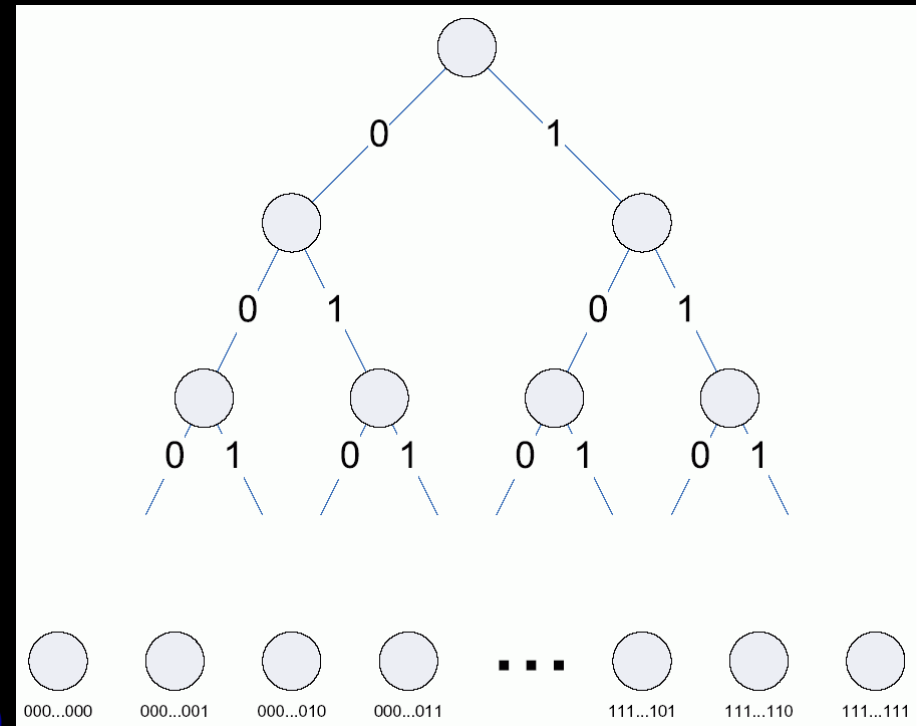
Analysis of F values when guessing key bits

F values in case of correct key bit guesses using 5000 samples



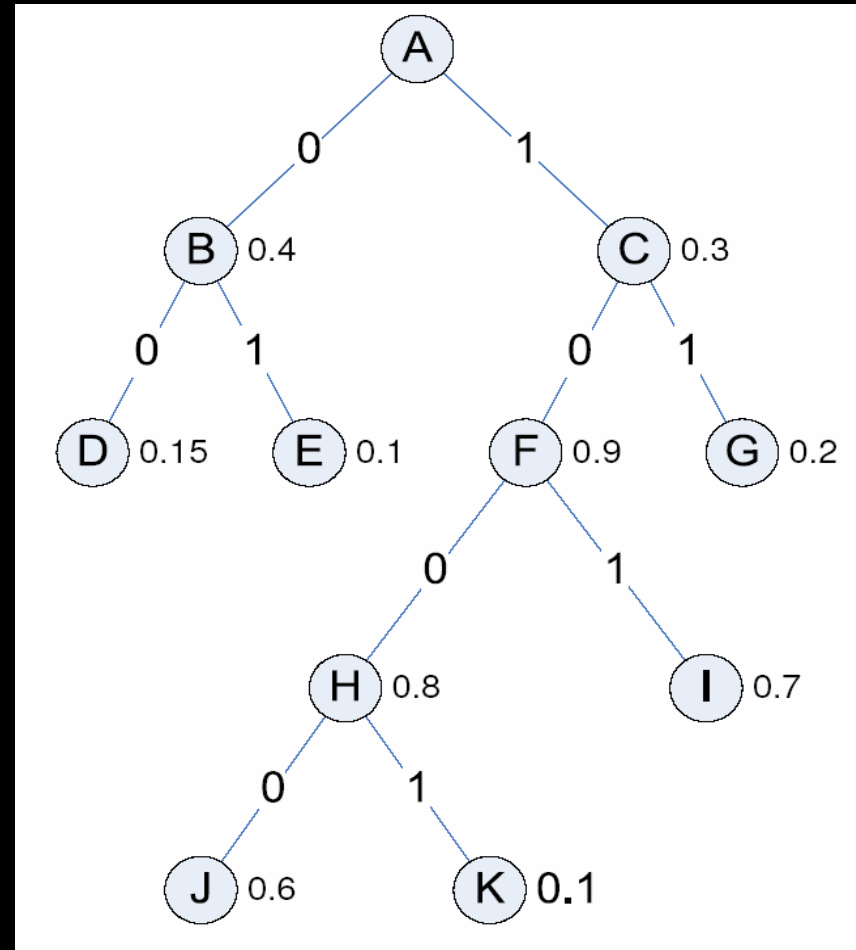
Key-tree concept

- Length of the key is n bits
- Binary tree of depth n
- Each node is a key prefix
- 2^n leaves = possible keys
- Each path is a key from LSB to MSB
- Goal: instead of exhaustive search among 2^n leaves, find the correct path by visiting as few nodes as possible:
 - Theoretical minimum:
visit n nodes, one path in the tree
 - Theoretical maximum:
visit all the nodes in the tree ($2^{n+1}-1$)
- Finding the good key = not to miss the good path
- If we are on a wrong path: how to backtrack?
- We store the key prefix candidates that have been visited, and always extend the most probable one
 - Selection is based on the “goodness” of the candidates



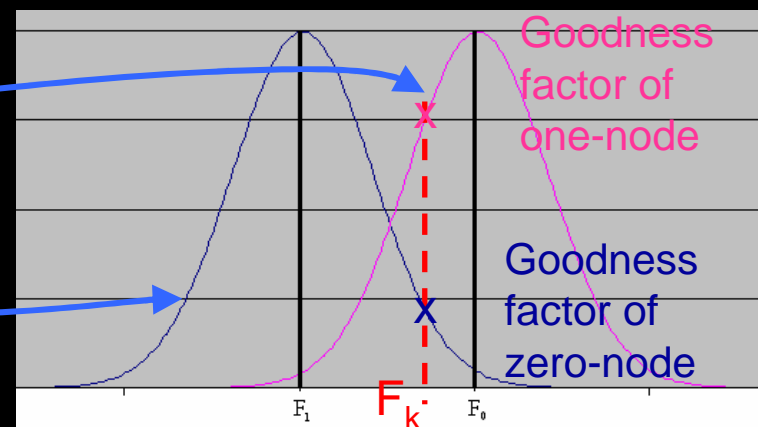
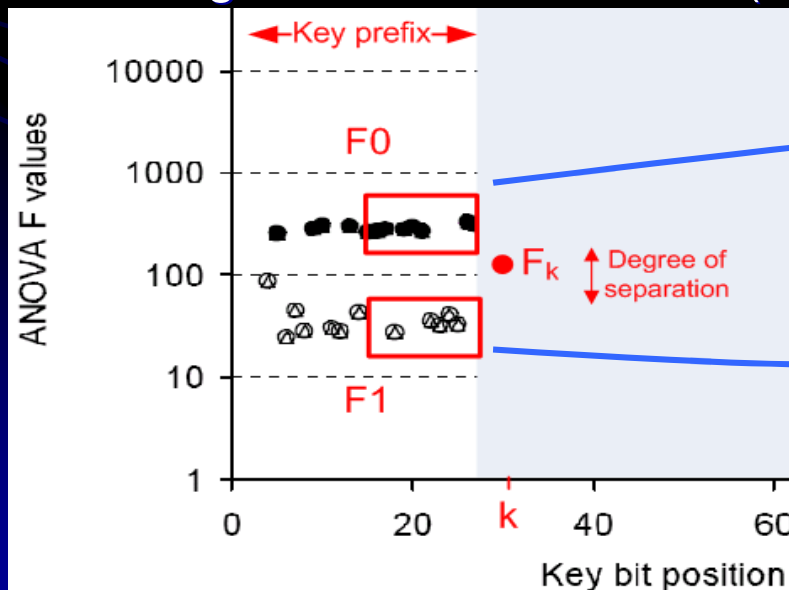
Key recovery algorithm

- Start from root (node A)
 - Guess $d_1 = 1 \rightarrow$ calculate F
 - Insert B, C leaves and calculate their goodness from F (0.4 and 0.3)
 - Update list of continuation nodes (candidate prefixes) = B, C
 - Select the best continuation node (B)
 - Guess $d_2 = 1 \rightarrow$ F
 - Insert D, E leaves, and calculate their goodness
 - Update list of continuation nodes: D, E, C
 - ...
- Initialization phase:
- Generate the full 10 bit depth tree
 - Goodness values for 2^{10} continuation nodes



Local goodness of a node

- Aim: normalized goodness measure of the guesses
- Goodness: after guessing key-bit k , we get F_k
 - A. How F_k fits in the group of previous F values of 0 or 1 bits?
Fit bell-shaped curves on previous F_0 and F_1 values
Value at F_k = goodness factor of zero-node and one-node
 - B. Degree of separation of previous F_0 and F_1 values (to detect incorrect guesses)
2-sample unpooled t-test: amount of separation bw the means of 2 samples
- Local goodness of a node is (A x B)



Cumulated goodness of a node

- Variance of local goodness values can cause unnecessary jumps between paths in the tree
- Cumulated goodness:
 - Decreases variance of local goodness values
 - Exponential forgetting of local goodness of previous nodes on the path:
 - x : goodness of the node, $x^{(1)}$ goodness of its parent, ...
 - λ : forgetting „speed“ [0,1] (0.5-0.6)

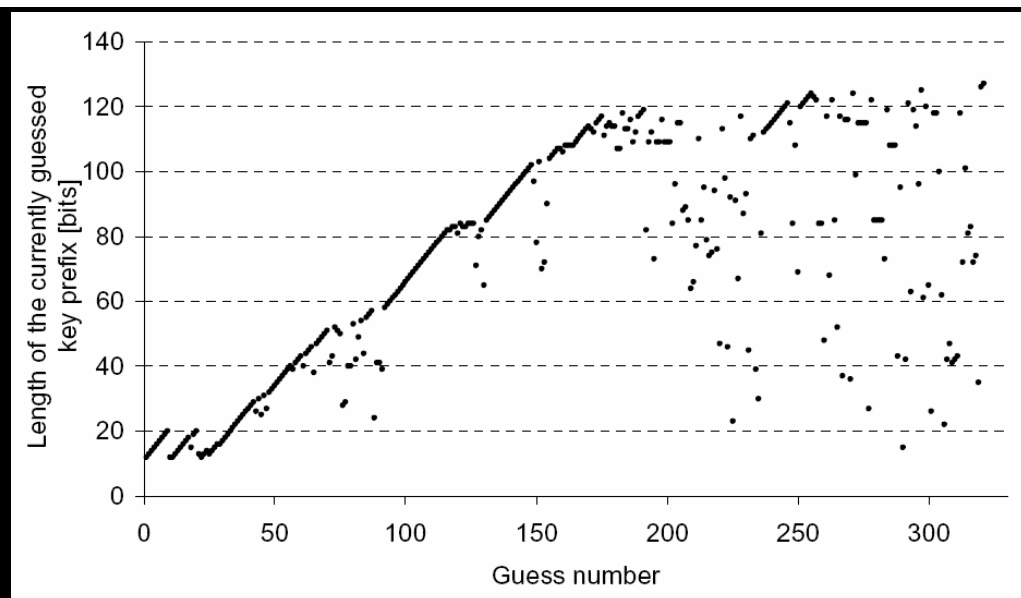
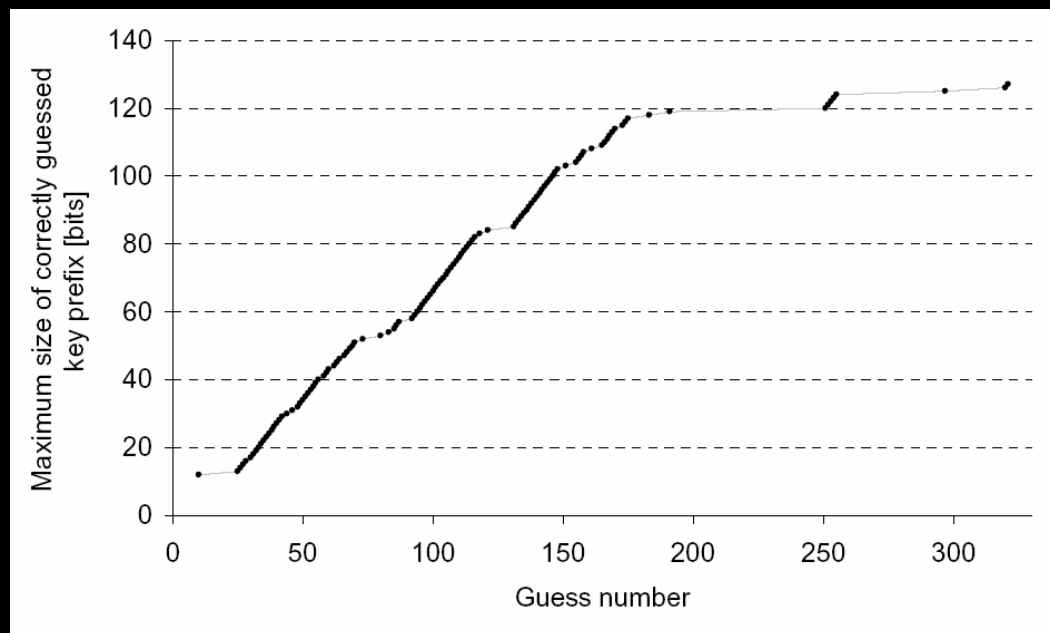
$$C = \lambda x + (1 - \lambda) \lambda x^{(1)} + (1 - \lambda) \lambda^2 x^{(2)} + \dots + (1 - \lambda) \lambda^n x^{(n)}$$

- Cumulated goodness values are used as the “goodness” of the nodes in the key-tree

Results

- 10.000 messages
- 67 hours
- 127 bit key
- 320 guesses
(60% of guesses recovered 90% of key)

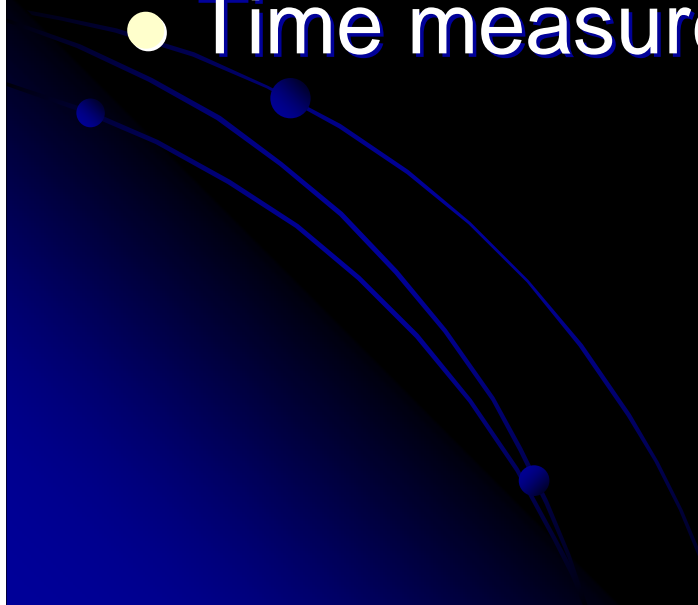
- Problematic issues
 - Last 8-10 bits → brute force
 - Long runs in the key → dynamic look-back



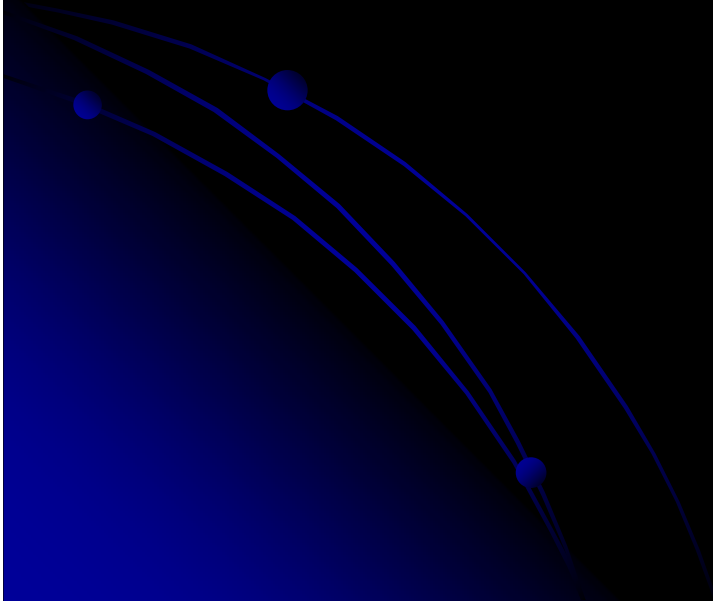
Concluding remarks

- Timing attack scheme
 - New concept on key-trees and goodness assignment
 - It examines a very small part of the key-space
 - Guess one bit at a time, but instead of keeping track of only one key prefix and progressing linearly, we store several key-prefix candidates at the same time and extend the most probable one
- Proof-of-concept attack against RSA with Montgomery multiplication
 - New concept on the exploitation of extra-reduction steps using ANOVA test

Future work

- Ameliorate performance at last key bits
 - Ameliorate goodness calculation in case of long runs of same value in the key
 - Apply our concept on other cryptographic algorithm implementations
 - Time measurement, information gathering
- 

Thank you for your attention!
Any questions?



RSA with Montgomery multiplication

- RSA: $C = P^d \pmod{N}$
 - (d, N) is the private key, (e, N) is the public key
 - Attacker aims to recover d based on a number of P s and their coding time
- An implementation of RSA
 - Progresses bit-by-bit : the coding time sums up
 - 2 multiplications (line 4 and 6), line 6 is known

Algorithm 1 RSA using modulo multiplication.

```
1:  $C := 1$ 
2: for  $i = 1$  to length of  $d$  do
3:   if  $d_i$  is 1 then
4:      $C := CP \pmod{N}$ 
5:   end if
6:    $P := PP \pmod{N}$ 
7: end for
```

RSA with Montgomery multiplication

- RSA with Montgomery multiplication
- Advantages: multiplication replaced by addition and shift operations

```
Algorithm 2 Montgomery Algorithm (MM):  
 $Z \equiv \text{MM}(X, Y) \equiv XYr^{-1} \pmod{N}$   
1:  $Z := 0$   
2: for  $i = 0$  to  $n - 1$  do  
3:   if  $X_i$  is 1 then  
4:      $Z = Z + Y$   
5:   end if  
6:   if  $Z$  is an odd number then  
7:      $Z = Z + N$   
8:   end if  
9:   Shift right the binary form of  $Z$  with one position  
10: end for  
11: if  $Z \geq N$  then  
12:    $Z = Z - N$  // This is the extra-reduction step.  
13: end if
```

- One multiplication can be performed by two Montgomery operations:
 $X = A \cdot B \pmod{N}$ is equivalent with $A' = \text{MM}(A, r^2)$, $X = \text{MM}(A', B)$
- One multiplication contains inherently 0, 1, or 2 extra-reduction steps

Guessing key bits

- For each key bit from LSB to MSB
 - Let guess key bit k
 - Measure t_k , and calculate T_{k+1} for each P
 - Organize T_{k+1} values into three groups based on $t_k=0|1|2$
 - Test with ANOVA whether group means are equal
 - we get F_k value for key bit k
- F_k value \sim correctness of the guess ?

Challenges in using F values for key recovery

- In case of correct key prefix the F values are well separated into two groups:
 - „Low F” if the guess was correct
 - „High F” if the guess was incorrect
- Quick feedback for incorrect bits in the key prefix
 - Next 2-8 guesses
- But
 - more than exponential increase of values
 - initial phase: high variation, not easy to separate
 - end phase: sharp increase in both trend lines
 - depends on key, key bit position, number of samples
 - long runs of 0 or 1 make difficult to indicate correctness

Key recovery algorithm

- Attacker knows
 - RSA with Montgomery multiplication
 - Public key (e, N)
 - Coding time (in extra-reduction steps) for a large number of messages P
- OFFLINE key recovery (our exceptions)
 - Progresses bit-by-bit from LSB to MSB
 - Keeps track of many key prefixes indicating their “goodness” (introduced later)
 - Choose at each step the most probable one
 - Finds the correct key with 100% probability

RSA with Montgomery multiplication

- Multiplications are implemented using the Montgomery algorithm
 - Line 4 inherently contains 0, 1, or 2 extra-reduction steps: $t_k=0|1|2$
 - No extra-reductions can not be calculated, just measured with a guessed key prefix
- Attacker has knowledge on the total number of extra-reduction steps in line 4

Algorithm 1 RSA using modulo multiplication.

```
1:  $C := 1$ 
2: for  $i = 1$  to length of  $d$  do
3:   if  $d_i$  is 1 then
4:      $C := CP \bmod N$ 
5:   end if
6:    $P := PP \bmod N$ 
7: end for
```