Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers

Ivica Nikolić (joint work with Alex Biryukov)

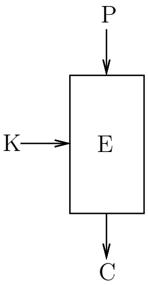
University of Luxembourg, Luxembourg

1 June 2010

Basics

Block cipher $E_K(P)$

- Input: Plaintext P and key K
- Output: Ciphertext C



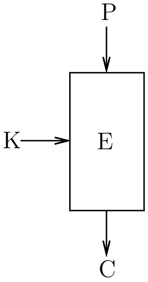
Basics

Attacker does not know the key.
He can fix:

- P and obtain C
- C and obtain P

and try to find:

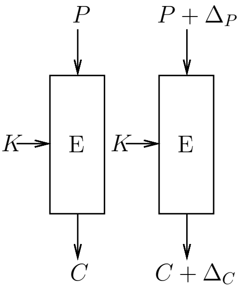
- Distinguisher
- Key recovery



Differential Attacks

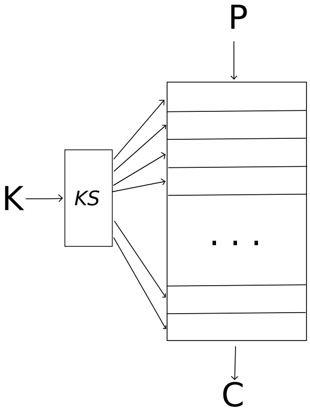
Differential analysis – the most popular form of attack. Find *specific* differences Δ_P, Δ_C s.t.:

$$\text{plaintexts } (P, P \oplus \Delta_P)$$
$$\downarrow$$
$$\text{ciphertexts } (C, C \oplus \Delta_C)$$



Differential Attacks

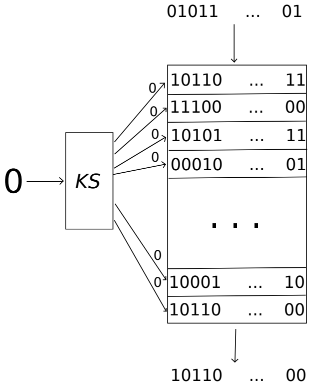
- Internally, a cipher has some number of rounds
- A key schedule from the master key produces round keys (subkeys)



Differential Attacks

Differential characteristic – round-by-round propagation of some initial difference

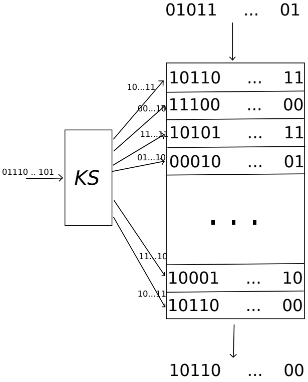
- Fixed-key differential characteristic - no difference in the key



Differential Attacks

Differential characteristic – round-by-round propagation of some initial difference

- *Related-key differential characteristic* - difference in the key as well



Recent Attacks on AES

- Related-key differential attack on AES-256 (Biryukov-Khovratovich-Nikolić , CRYPTO 2009)
- Related-key boomerang attacks on AES-192 and AES-256(Biryukov-Khovratovich, ASIACRYPT 2009)

Our Objectives

- Create a tool for automatic search of related-key differential characteristics in all versions of AES
- Extend the tool and apply it to other ciphers

Target ciphers

Byte-oriented block ciphers - All the transforms in the cipher are byte-oriented

- Big advantage: *compact representation of the state and the subkeys is applicable* $\Rightarrow$ the effective size can be reduced by a factor of 8 $\Rightarrow$ search becomes feasible (when there is low branching in the round transforms)
- Example: AES-128 has 128-bit state and 128-bit subkeys $\Rightarrow$ 16-bit state and 16-bit subkeys $\Rightarrow$ search space is 2^{32}

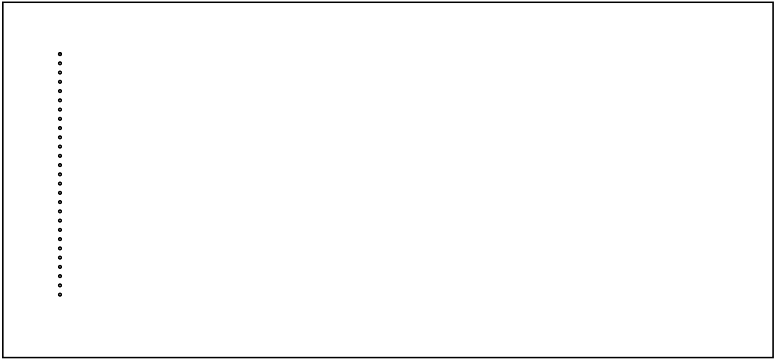
Matsui's approach to DES

We base our tool on Matsui's approach for search of the best fixed-key characteristic in DES

- Given the probabilities of the best $1, 2, \ldots, r-1$ round characteristics and some r-round characteristic it builds *t*he best r-round characteristic.
- Recursive; extend the characteristics only if its prob. $\times$ the prob. of the rest of the rounds is higher then the previous best prob. on all rounds.

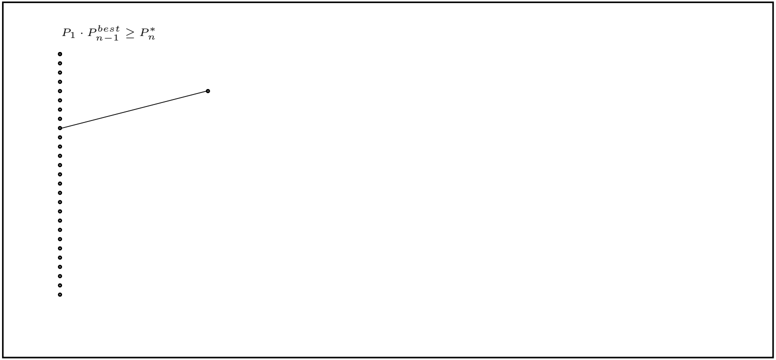
Matsui's approach

For each $r-1$ round char.: extend for one round, and check if $P_r \cdot P_{n-r}^{best} \geq P_n^*$ (if $P_n > P_n^*$ update P_n^*)



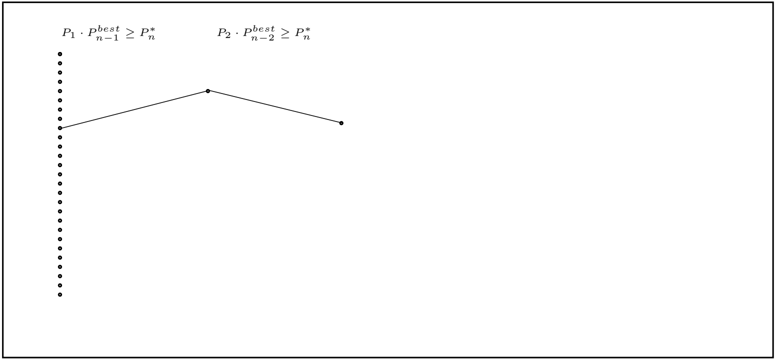
Matsui's approach

For each $r-1$ round char.: extend for one round, and check if $P_r \cdot P_{n-r}^{best} \geq P_n^*$ (if $P_n > P_n^*$ update P_n^*)



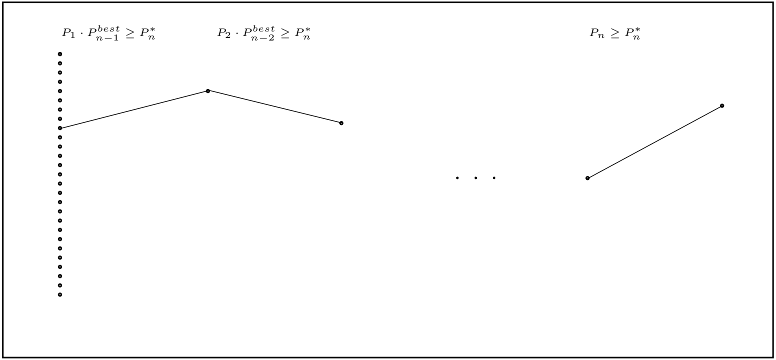
Matsui's approach

For each $r-1$ round char.: extend for one round, and check if $P_r \cdot P_{n-r}^{best} \geq P_n^*$ (if $P_n > P_n^*$ update P_n^*)



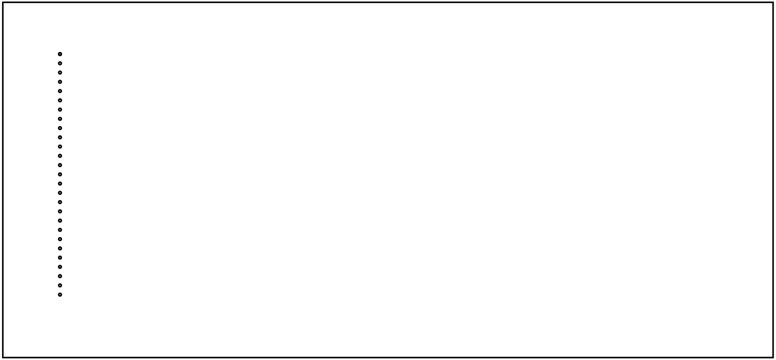
Matsui's approach

For each $r-1$ round char.: extend for one round, and check if $P_r \cdot P_{n-r}^{best} \geq P_n^*$ (if $P_n > P_n^*$ update P_n^*)



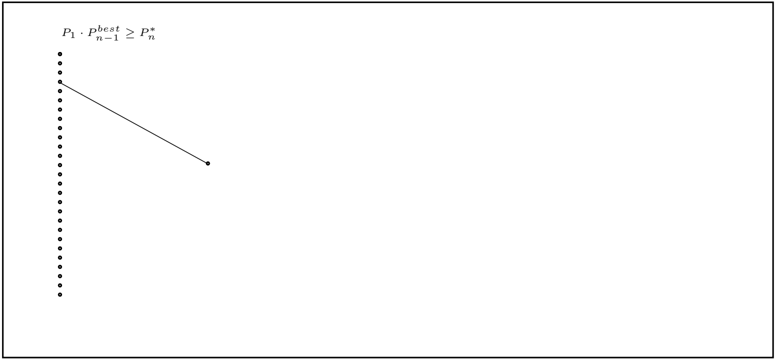
Matsui's approach

For each $r-1$ round char.: extend for one round, and check if $P_r \cdot P_{n-r}^{best} \geq P_n^*$ (if $P_n > P_n^*$ update P_n^*)



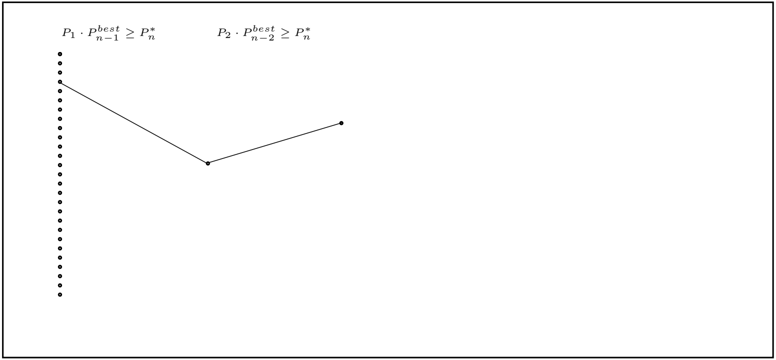
Matsui's approach

For each $r-1$ round char.: extend for one round, and check if $P_r \cdot P_{n-r}^{best} \geq P_n^*$ (if $P_n > P_n^*$ update P_n^*)



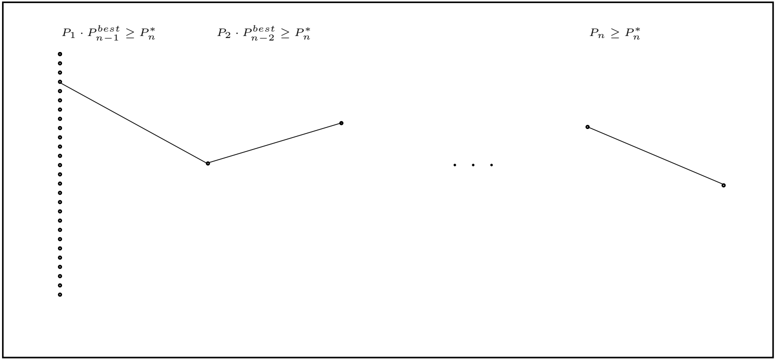
Matsui's approach

For each $r-1$ round char.: extend for one round, and check if $P_r \cdot P_{n-r}^{best} \geq P_n^*$ (if $P_n > P_n^*$ update P_n^*)



Matsui's approach

For each $r-1$ round char.: extend for one round, and check if $P_r \cdot P_{n-r}^{best} \geq P_n^*$ (if $P_n > P_n^*$ update P_n^*)



Matsui's approach: Pros and cons

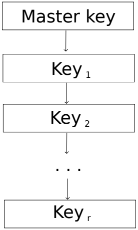
- Computation complexity cannot be predicted - it depends on how "good" the round-reduced characteristics are (worst case, it is exponential)
- Requires negligible memory
- When too many one-round characteristics, the search becomes infeasible

We introduce modifications in the tool to overcome the last obstacle

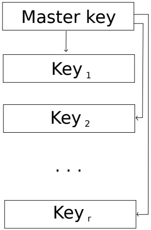
Variants of the tool

Depending on the key schedule, different variants are interesting:

Subkeys consecutively obtained one from another



Subkeys obtained from the master key



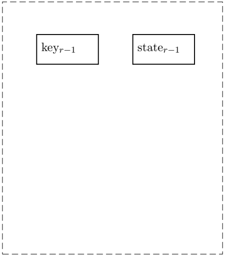
Variants of the tool

Depending on the degree of branching in the key schedule (for a fixed difference), consecutive key schedule can be divided into:

- KS with low branching
- KS with high branching

Variant 1 - low branching, consecutive subkeys

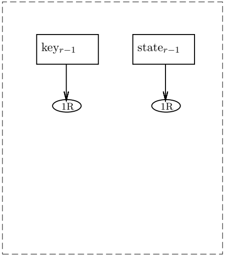
Apply straightforward Matsui's approach



One round characteristic

Variant 1 - low branching, consecutive subkeys

Apply straightforward Matsui's approach



One round characteristic

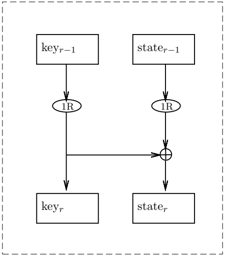
- Go 1R in subkey and state
-

Variant 1 - low branching, consecutive subkeys

Apply straightforward Matsui's approach

One round characteristic

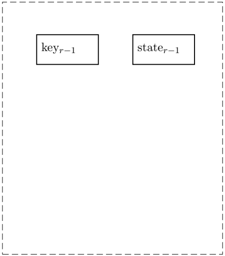
- Go 1R in subkey and state
- XOR the subkey to the state



Variant 2 - high branching, consecutive subkeys

Apply Matsui's approach, but change how one-round characteristics are produced

One round characteristic

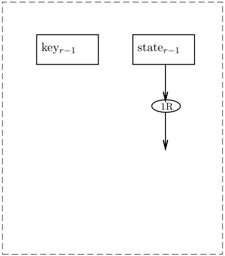


Variant 2 - high branching, consecutive subkeys

Apply Matsui's approach, but change how one-round characteristics are produced

One round characteristic

- Go 1R in state
-
-
-

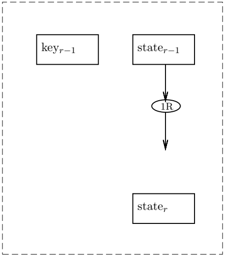


Variant 2 - high branching, consecutive subkeys

Apply Matsui's approach, but change how one-round characteristics are produced

One round characteristic

- Go 1R in state
- Take $state_r$
-
-

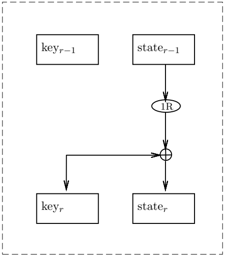


Variant 2 - high branching, consecutive subkeys

Apply Matsui's approach, but change how one-round characteristics are produced

One round characteristic

- Go 1R in state
- Take state_r
- Produce subkey
-

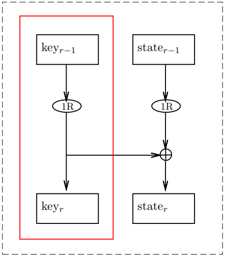


Variant 2 - high branching, consecutive subkeys

Apply Matsui's approach, but change how one-round characteristics are produced

One round characteristic

- Go 1R in state
- Take $state_r$
- Produce subkey
- Check if the subkey is good



Variant 3 - non-consecutive subkeys

Apply Matsui's approach for fixed subkeys

1. Fix the master key difference, and obtain all possible subkeys differentials
2. For each characteristics in the state, apply the variant 1 assuming the subkey characteristics are already fixed

AES

- AES has high branching in the key schedule due to XORs $\Rightarrow$ variant 2 is used
- Results:

Cipher	Attack	Rounds	Workload
AES-128	Differential	5	$\geq 2^{6 \cdot 17}$
	Boomerang	7	2^{97}
AES-192	Differential	11	$\geq 2^{6 \cdot 31}$
	Boomerang[a]	12	2^{169}
AES-256	Differential[b]	14	2^{131}

[a] The attack was improved

[b] The characteristic was confirmed to be optimal

Camellia

The key schedule is not byte-oriented, we attack a modified version with changed rotational amounts

- Camellia has non-invertible key schedule $\Rightarrow$ variant 3 is used

Results:

- Differential characteristic on 8 rounds (out of 18)
- Chosen-key attack on all 18 rounds

Khazad

- Khazad has high branching in the key schedule due to XORs $\Rightarrow$ variant 2 is used

Results:

- Differential characteristic on 7 rounds (out of 8)
- Boomerang attack on 7 rounds – lower complexity
- Chosen-key attack on all 8 rounds

1 Block Ciphers

2 The tool

3 Applications

4 Conclusion

Conclusion

- Presented a tool for automatic search of related-key differential characteristics in byte-oriented ciphers
- The best characteristics in AES, byte-Camellia and Khazad were found
- The tool can be used to prove to resistance of ciphers to RK attacks

Future Research

- Apply a similar tool to other byte-oriented primitives (hash functions with bigger internal state)
- Apply the tool to ciphers with a small non-byte oriented part (such as the original version of Camellia)
- Find a similar tool for word-oriented primitives