Horst Görtz Institute for IT-Security

Realtime A5/1 Attacks with Precomputation Tables

Martin Novotný, Andy Rupp
Ruhr University Bochum

Outline

Time-Memory Trade-off Tables

- Original Hellman Approach
- Distinguished points
- TMTO with multiple data
- Rainbow tables
- Thin-rainbow tables

Architecture of the A5/1 TMTO engine

Implementation results

Two extreme approaches

Brute force attack

Check all combinations of a key *K* online

- **time** $T = N = 2^k$
- memory $M = 1$

Table lookup

(For a given plaintext *P*)

All pairs key-ciphertext $\{K_i, C_i\}$ precomputed and stored (sorted by **C**)

Online phase: Look-up **C** in the table (and find *K*)

- time $T = 1$
- **memory** $M = N = 2^k$

Time-Memory Trade-Off (Hellman, 1981)

Compromises the above two extreme approaches

Precomputation phase: For a *given* plaintext ***P***:

- *precompute* (ideally all) pairs key-ciphertext $\{K_i, C_i\}$;
- *store* only ***some*** of them in the table.

Online phase:

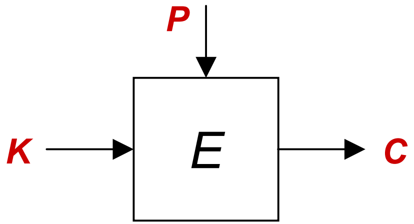
- Perform *some* computations;
- lookup the table and find the key ***K***.
  - time $T = N^{2/3}$
  - memory $M = N^{2/3}$

Precomputation (offline) phase

Horst Görtz Institute
for IT-Security

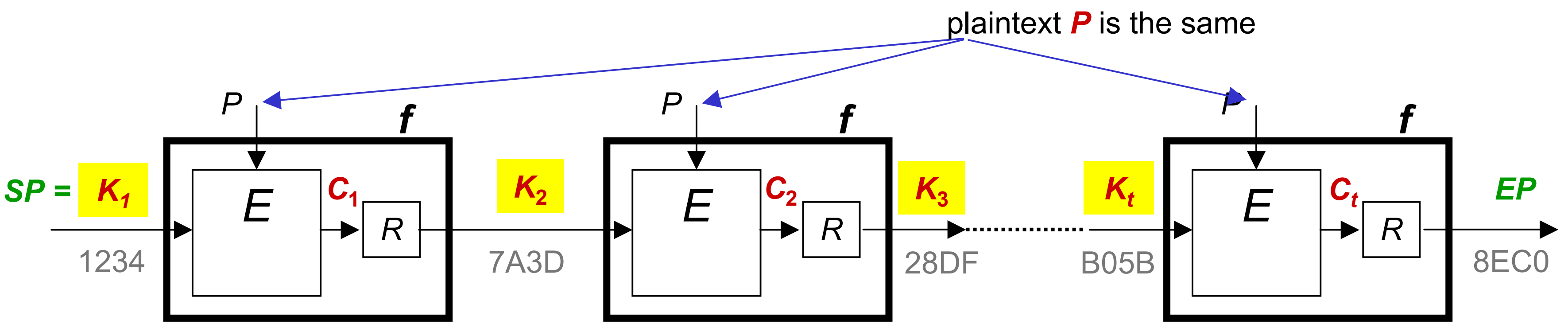
Idea: Encryption function ***E*** is a pseudo-random function

$$C = E_K(P)$$



Pairs {K_i, C_i} organized in chains

- C_i is used to create a key K_{i+1} for the next step
- ***E*** is pseudo-random ⇒ we perform a *pseudo-random walk* in the keyspace



R	–	reduction function	(DES: ***C*** has **64** bits, ***K*** has **56** bits)
f	–	step function	$f(x) = R(E_x(P))$

Precomputation (offline) phase

Horst Görtz Institute
for IT-Security

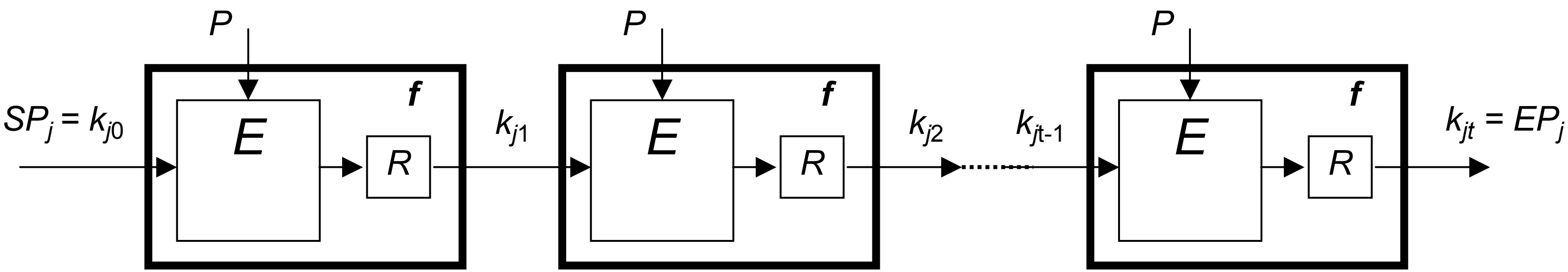
1234 $SP_1 = k_{10} \to^f k_{11} \to^f k_{12} \to^f \ldots \to^f k_{1t\text{-}1} \to^f k_{1t} = EP_1$ 8EC0

1235 $SP_2 = k_{20} \to^f k_{21} \to^f k_{22} \to^f \ldots \to^f k_{2t\text{-}1} \to^f k_{2t} = EP_2$ 2A1B

1236 $SP_3 = k_{30} \to^f k_{31} \to^f k_{32} \to^f \ldots \to^f k_{3t\text{-}1} \to^f k_{3t} = EP_3$ 4D3C

… … …

9999 $SP_m = k_{m0} \to^f k_{m1} \to^f k_{m2} \to^f \ldots \to^f k_{mt\text{-}1} \to^f k_{mt} = EP_m$ 02E3



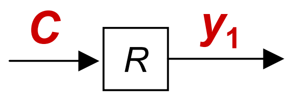
m chains with a fixed length ***t*** generated

Only pairs {SP_i, EP_i} stored (sorted by ***EP***) ⇒ reducing memory requirements

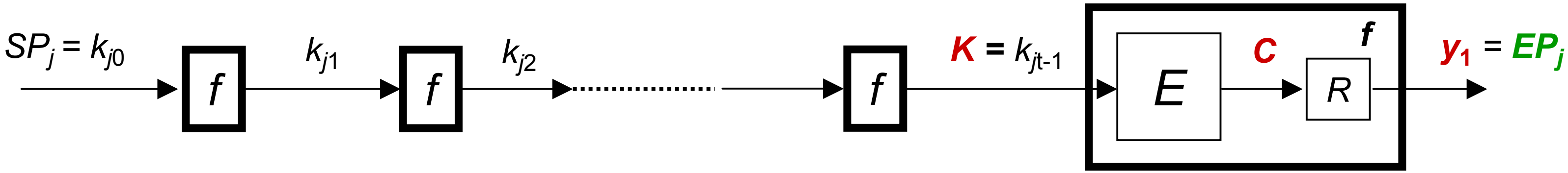
Online phase

Given $\boldsymbol{C}$.

1. Compute $\boldsymbol{y_1} = R(\boldsymbol{C})$;
   Lookup table if $\boldsymbol{y_1} = \boldsymbol{EP_j}$

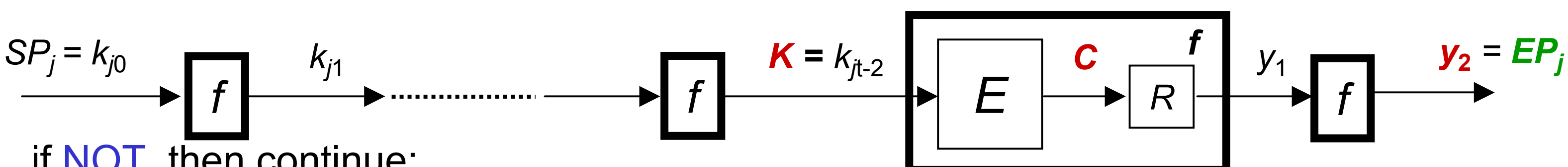


   1. if YES, then (potentially) $\boldsymbol{K} = \boldsymbol{k_{jt-1}} \Rightarrow$ compute $\boldsymbol{k_{jt-1}} = \overbrace{f(f(f(\ldots f(}^{(t-1)\times}\boldsymbol{SP_j}) \ldots)))$, stop



   2. if NOT, then continue:

2. Compute $\boldsymbol{y_2} = f(\boldsymbol{y_1}) = f(R(\boldsymbol{C}))$;
   Lookup table if $\boldsymbol{y_2} = \boldsymbol{EP_j}$
   1. If YES then (potentially) $\boldsymbol{K} = \boldsymbol{k_{jt-2}} \Rightarrow$ compute $\boldsymbol{k_{jt-2}} = \overbrace{f(f(f(\ldots f(}^{(t-2)\times}\boldsymbol{SP_j}) \ldots)))$, stop



   2. if NOT, then continue:

3. Compute $\boldsymbol{y_3} = f(\boldsymbol{y_2}) = f(f(\mathbf{y1})) = f(f(R(\boldsymbol{C})))$;
   Lookup table if $\boldsymbol{y_3} = \boldsymbol{EP_j}$

Birthday paradox problem

Horst Görtz Institute for IT-Security

m chains of fixed length ***t*** generated

R is not bijective ⇒ some k_{ij} collide. Collisions yield in chain merges or in cycles in chains

Matrix stopping rule: Hellman proved that it is not worth to increase ***m*** or ***t*** beyond the point at which

$$m \cdot t^2 = N$$

(the coverage of keyspace does not increase too much)

He recommends to use ***r*** tables, each with different reduction function ***R***

Since also $N = m \times t \times r$, then $r = t$

Hellman recommends $m = t = r = N^{1/3}$

Hellman TMTO – Complexity

Precomputation phase

- Precomputation time $PT = m \times t \times r = N$
- Memory $M = m \times r = N^{2/3}$

Online phase

- Memory $M = N^{2/3}$
- Online time $T = t \times r = t^2 = N^{2/3}$
- Table accesses $TA = T = N^{2/3}$

Tradeoff curve

$$M^2 T = m^2 t^2 t^2 = m^2 t^4 = N^2$$

$$M^2 T = N^2$$

Distinguished points (Rivest, ????)

Horst Görtz Institute for IT-Security

Slight modification of original Hellman method

Goal: to reduce the number of table accesses ***TA*** (in Hellman ***TA*** = $N^{2/3}$)

Distinguished point is a point of a certain property (e.g. 20 most significant bits are equal to **0**).

000000000000000000000010101001101100101010010111110010110101

Distinguished points

Precomputation phase

- chains are generated until the *distinguished point* (DP) is reached
  - if the chain exceeds maximum length t_{max}, then it is discarded and the next chain is generated
  - the chain is also discarded if the DP has been reached, but the chain is too short t_{min} (to have better coverage)
- triples $\{SP_j, EP_j, l_j\}$ stored, sorted by EP (l_j is a length of the chain)

Online phase

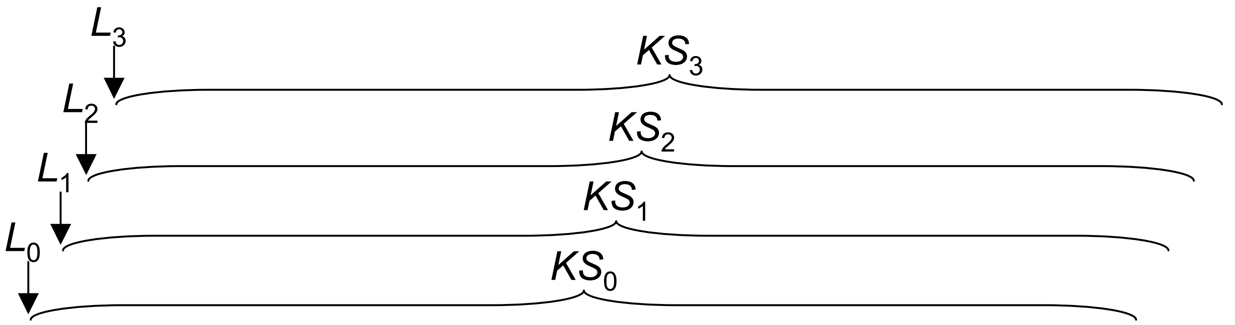
- compute $y_{i+1} = f(y_i)$ iteratively until the DP is reached (or the maximum length t_{max} is exceeded)
- lookup the table (only) if the distinguished point is reached

Advantages

- Table accesses $TA = r = N^{1/3}$ (c.f. $TA = t \times r = N^{2/3}$ in original Hellman)
- Chain loops are not possible

TMTO with multiple data (Biryukov & Shamir, 2000)

Important for stream ciphers: To reveal an internal state L_i having k bits we need only k bits of a keystream KS_i



0100111101101010110100101010010100010010100010011110110001

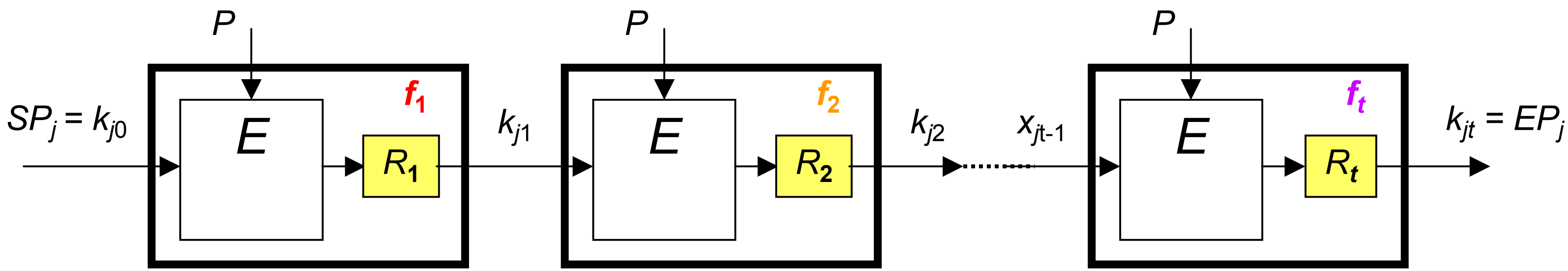
Having D data samples of the ciphertext C (or the keystream KS) we have D times more chances to find the key K (or the internal state L)

⇒ We calculate r/D tables only
- ⇒ we save the precomputation time PT and the memory M
- × online time T and the number of table access TA remain unchanged

Rainbow tables (Oechslin, 2003)

Idea: to use different reduction function R_i in each step of chain generation, hence the step functions are:

$$f_1 \quad f_2 \quad f_3 \quad \dots \quad f_{t-1} \quad f_t$$



Online phase:

– Compute $y_1 = R_t(C)$, compare with *EP*s, if no match, then
– Compute $y_2 = f_t(R_{t-1}(C))$, compare with *EP*s, if no match, then
– Compute $y_3 = f_t(f_{t-1}(R_{t-2}(C)))$, compare with *EP*s, if no match, then
– …

Rainbow tables

Just one table (or only several tables) generated,

- $m = N^{2/3}$ (t reduction functions used ⇒ the table can be t times longer),
- $t = N^{1/3}$

Advantages

- chain loops impossible
- point collisions lead to chain merges only if the equal points appear in the same position of the chain
- online time T about ½ of the online time of original Hellman (for single data)
- number of table accesses the same like for the Hellman+DP method (for single data)

Disadvantages

- Inferior to the Hellman+DP method in the case of multiple data ($D > 1$) (online time T and the number of table accesses TA are D-times greater)

Thin-rainbow tables

hgi
Horst Görtz Institute
for IT-Security

The way to cope with the rainbow tables when having multiple data

The sequence of ***S*** different reduction functions is applied ***k***-times periodically in order to create a chain:

$$f_1\ f_2\ f_3 \dots f_{S-1}\ f_S\ f_1\ f_2\ f_3 \dots f_{S-1}\ f_S\ \dots \dots\ f_1\ f_2\ f_3 \dots f_{S-1}\ f_S$$

Chain length

$$t = S \times k \qquad \Rightarrow \qquad S = t/k$$

Thin-rainbow + DP (to reduce # table accesses ***TA***):

- DP criterion is checked after each f_S

$$f_1\ f_2\ f_3 \dots f_{S-1}\ f_S\ f_1\ f_2\ f_3 \dots f_{S-1}\ f_S\ \dots \dots\ f_1\ f_2\ f_3 \dots f_{S-1}\ f_S$$

- We store only chains for which $k_{min} < k < k_{max}$

Candidates for implementation (in case of multiple data, *D*>1)

1. Hellman + DP
2. Thin-rainbow + DP

Both have the same precomputation complexity

Both have comparable online time ***T*** and # table accesses ***TA***

Hellman+DP checks DP-criterion after each step-function ***f***

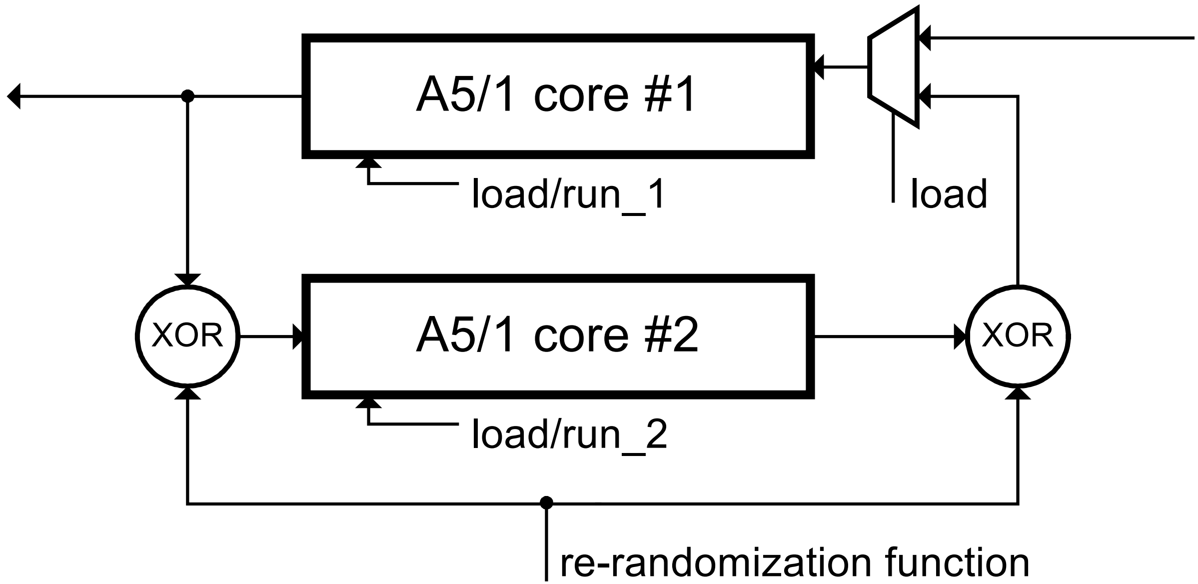
Thin-rainbow+DP checks DP-criterion after f_S only

⇒ We implemented Thin-rainbow+DP

- simpler HW, better time/area product
- $S \cong 2^{14}$; $k \cong D \cong 2^{6}$

A5/1 TMTO basic element

hgi
Horst Görtz Institute
for IT-Security



Calculates one chain

Two-stroke mode:

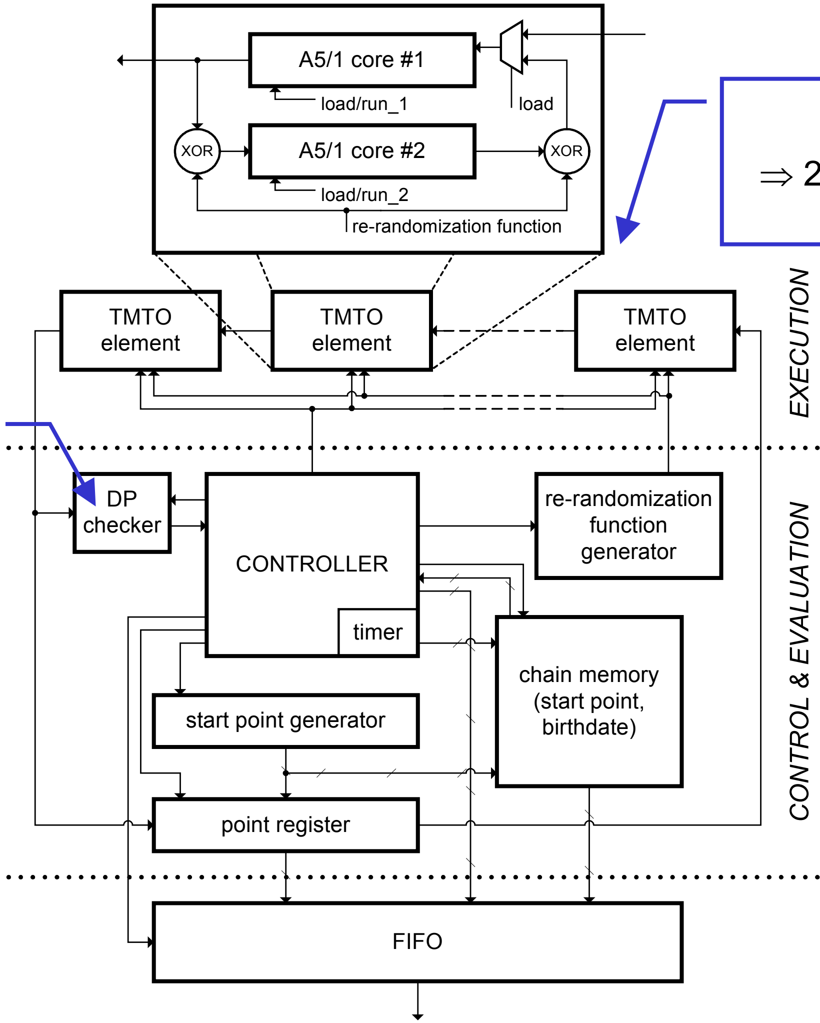
1. core #1 generates keystream, core #2 is loaded
2. core #2 generates keystream, core #1 is loaded

A5/1 TMTO engine

234 TMTO elements

⇒ 234 chains computed in parallel in Spartan 3-1000 FPGA

TMTO elements share the DP-checker



Implementation results

COPACOBANA is able to perform up to **2^{36}** step-functions f_i per second

- 234 TMTO elements/FPGA
- 120 FPGAs
- maximum frequency f_{max} = 156 MHz
- one step-function takes 64 clock cycles

$$234 \times 120 \times 156 \cdot 10^6 / 64 \;\cong\; 2^{36}$$

Parameter choices for *D*=64

Horst Görtz Institute for IT-Security

chains computed m	rainbow sequence S	DP criterion d [bits]	#seq. in chain k	precomp. time PT [days]	disk usage DU [TB]	online time OT [s]	table accesses TA	success ratio SR
2^{41}	2^{15}	5	$[2^3, 2^6]$	337.5	7.49	27.8	2^{21}	0.86
2^{39}	2^{15}	5	$[2^3, 2^7]$	95.4	3.25	36.3	2^{21}	0.67
2^{40}	**2^{14}**	**5**	**$[2^4, 2^7]$**	**95.4**	**4.85**	**10.9**	**2^{20}**	**0.63**
2^{40}	2^{14}	5	$[2^3, 2^6]$	84.4	7.04	7.0	2^{20}	0.60
2^{39}	2^{15}	5	$[2^3, 2^6]$	84.4	3.48	27.8	2^{21}	0.60
2^{40}	2^{14}	5	$[2^4, 2^6]$	84.4	5.06	8.5	2^{20}	0.55
2^{37}	2^{15}	6	$[2^4, 2^8]$	47.7	0.79	73.5	2^{21}	0.42