Hitag 2 Hell – Brutally Optimizing Guess-and-Determine Attacks

Aram Verstegen and Roel Verdult and Wouter Bokslag

FactorIT B.V.

August 14th 2018
Overview

1. Introduction
2. LFSR-based stream ciphers
3. Hitag2
4. Guess-and-determine attacks
5. Performance optimization
6. Performance evaluation
7. Takeaways
8. End-to-end demonstration
Remote Keyless Entry

**UHF frame format**

- Contains 32-bit MAC (KS) on frame data
- KS is a 32-bit sample of Hitag2 keystream
- Determined by the public UID, secret 48-bit key and partly public 32-bit nonce
- 28-bit counter is incremented after each press
LFSR-based stream ciphers

- State register is initialized with secret key
- Round function shifts register by one bit
- *Feedback function* computes new bit to shift in, based on state

LFSR

Verstegen, Verdult, Bokslag (FactorIT)
LFSR-based stream ciphers

- State register is initialized with secret key
- Round function shifts register by one bit
- Feedback function computes new bit to shift in, based on state

Cryptanalysis

- Try to find state at any point in time
- Construct inverse round function, roll back to initial state
Hitag2

Hitag2 filter function

- Keystream generated by nonlinear filter function
- Consists of two layers of boolean operations
- Keystream bit value is determined by 20 state bits
Leakage of state information

- Keystream bit reveals information about cipher state bits
- $2^{20}$ candidates, but only $2^{19}$ generate observed bit
- Eliminate half the possibilities for each keystream bit
Guess-and-determine

Propagate guesses over multiple layers

- Several of the input bits for \( f \) can overlap with previous guesses
- Fewer guesses to make on the state for each subsequent keystream bit tested
- Still eliminate half the guesses in each layer
- Layer guesses can be expressed as bit masks
Guess-and-determine

Propagate guesses over multiple layers

- Several of the input bits for $f$ can overlap with previous guesses
- Fewer guesses to make on the state for each subsequent keystream bit tested
- Still eliminate half the guesses in each layer
- Layer guesses can be expressed as bit masks
Guess-and-determine

| Layer 0           | 47  | 46  | 45  | 44  | 43  | 42  | 41  | 40  | 39  | 38  | 37  | 36  | 35  | 34  | 33  | 32  | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|-------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|                   | 47  | 46  | 45  | 44  | 43  | 42  | 41  | 40  | 39  | 38  | 37  | 36  | 35  | 34  | 33  | 32  | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |

Propagate guesses over multiple *layers*

- Several of the input bits for $f$ can overlap with previous guesses
- Fewer guesses to make on the state for each subsequent keystream bit tested
- Still eliminate half the guesses in each layer
- Layer guesses can be expressed as bit masks
Guess-and-determine

Propagate guesses over multiple layers

- Several of the input bits for \( f \) can overlap with previous guesses
- Fewer guesses to make on the state for each subsequent keystream bit tested
- Still eliminate half the guesses in each layer
- Layer guesses can be expressed as bit masks
Guess-and-determine

<table>
<thead>
<tr>
<th>Layer</th>
<th>States</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Layer 3</th>
<th>Layer 4</th>
<th>Layer 5</th>
<th>Layer 6</th>
<th>Layer 7</th>
<th>Layer 8</th>
<th>Layer 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Full state is determined at layer 9

- $2^{39}$ states reach this point on average
- Test 23 more keystream bits to reduce to $\sim 2^{16}$
- Internal states can be rolled back to initial state with inverse LFSR
Performance optimization

Trivial optimizations

- Pre-compute filter function outputs through lookup tables
- Pre-compute state bit guesses
- Unroll 9 layers of recursion as 9 nested loops
Performance optimization

Trivial optimizations
- Pre-compute filter function outputs through lookup tables
- Pre-compute state bit guesses
- Unroll 9 layers of recursion as 9 nested loops

Advanced optimizations
- Only generate ideal guesses
- Pre-compute and cache the results for later computations in earlier ones
- Bit-slice the implementation
Performance optimization

Generating ideal guesses

- Use known state information and expected result to generate ideal lookup tables per layer
- Can test first 9 keystream bits without relying on $f$ entirely
- 23 MB tables too large to fit in cache – negative trade-off
Performance optimization

Memoization

- From layer 2 onwards, some subfilters for upcoming layers are entirely determined
- Keep *memos* of the evaluation of these subfilters (*memoization*)
Performance optimization

![Graph showing subfilters computed vs keystore bit test for naive and memoized to 12 methods.](image)
Performance optimization

LFSR view

Many LFSRs are evaluated

Transpose bits in registers

for (i = 0; i < 8, i++)

ans[i] = ((r[i] & 0b1000) >> 3) ^ r[i] & 1

Bit-slicing

- Parallelize to native register width
- Transpose multiple states over multiple $v$-bit registers
Performance evaluation

<table>
<thead>
<tr>
<th>Optimization type</th>
<th>Platform</th>
<th>Cores</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive BF (Immler 2012)</td>
<td>Tesla C2050</td>
<td>448 CUDA</td>
<td>660 minutes</td>
</tr>
<tr>
<td>32-way bit-sliced (Benadjila et al. 2017)</td>
<td>GTX780Ti</td>
<td>2880 CUDA</td>
<td>1080 minutes</td>
</tr>
<tr>
<td>32-way bit-sliced (Benadjila et al. 2017)</td>
<td>16 x Tesla K80</td>
<td>79872 CUDA</td>
<td>45 minutes</td>
</tr>
<tr>
<td>Naive guess-and-determine</td>
<td>i7 3700K</td>
<td>4 Intel</td>
<td>127 minutes</td>
</tr>
<tr>
<td>Avoiding impossible guesses</td>
<td>i7 3700K</td>
<td>4 Intel</td>
<td>96 minutes</td>
</tr>
<tr>
<td>256-way bit-sliced, memoized to depth 12</td>
<td>i7 3700K</td>
<td>4 Intel</td>
<td>59 minutes</td>
</tr>
<tr>
<td>32-way bit-sliced, memoized to depth 17</td>
<td>GTX1080Ti</td>
<td>3584 CUDA</td>
<td>1.2 minutes</td>
</tr>
</tbody>
</table>

Table: Performance evaluation of average running time compared to related work.

- Two previous publications implement naive brute force
- Guess-and-determine offers serious performance improvement
- Factor 500 performance improvement\(^a\)

\(^a\)Normalizing for cuda core count and gpu base clock speed
Takeaways

Guess-and-determine optimization techniques

- Conceptual (predetermine guesses, memoized)
- Implementational (unrolled recursion, lookups, bit-sliced)

Proof of concept on Hitag2 stream cipher

- Recovers internal state in roughly 1 minute
- Applicable in RKE as well as in immobilizer context
Demonstration

Verstegen, Verdult, Bokslag (FactorIT)

Hitag 2 hell

August 14th 2018 18 / 20
Keystream: 0x74136f48
Demonstration

$ ./hitag2_cracker 74136f48
[+] FactorIT HITAG2 state finder
[i] Using platform: NVIDIA CUDA
[+] Cracker started
[i] Running Hitag2 on 1 device (25.00%)
[i] Running Hitag2 on 1 device (50.00%)
[i] Running Hitag2 on 1 device (75.00%)
[+] Cracker finished
Writing outfile 74136f48.txt

$ head 74136f48.txt
09a0245fcb36
0880245f0481
0f000ef0b3e3
...
http://www.hitag2hell.com/