SAPHE - Simple Accelerometer based wireless Pairing with HEuristic trees

Bogdan Groza[1] and Rene Mayrhofer[2]

[1]Politehnica University of Timisoara, Romania,
[2]Upper Austria University of Applied Sciences

December 4, 2012

The problem: usable and secure device pairing

The problem: usable and secure device pairing

One approach: deriving keys from shared motion

Difficulty: estimating entropy of accelerometer data

Shaking devices is intuitive, vigorous, and varying

But can't predict variability

- Time windows with low activity
- Resulting time series parts will have low entropy
- Long feature vectors
  $\rightarrow$ low probability of match
- Short feature vectors
  $\rightarrow$ brute-force key search

Basic idea (the easy part) — Step 1/3

Commit random material to derive thresholds on each side: $\bar{t}_A \leftarrow \mathcal{T}(r_A)$, $\bar{t}_B \leftarrow \mathcal{T}(r_B)$ (from some PRNG)

Device (A)		Device (B)
	Commitment Stage	
1. $r_A \in_R 1^k$		
	$\xrightarrow{H(r_A)}$	
2.		$r_B \in_R 1^k$
	$\xleftarrow{H(r_B)}$	

Basic idea (the easy part) — Step 2/3

Compare accelerometer values to the threshold points (random)

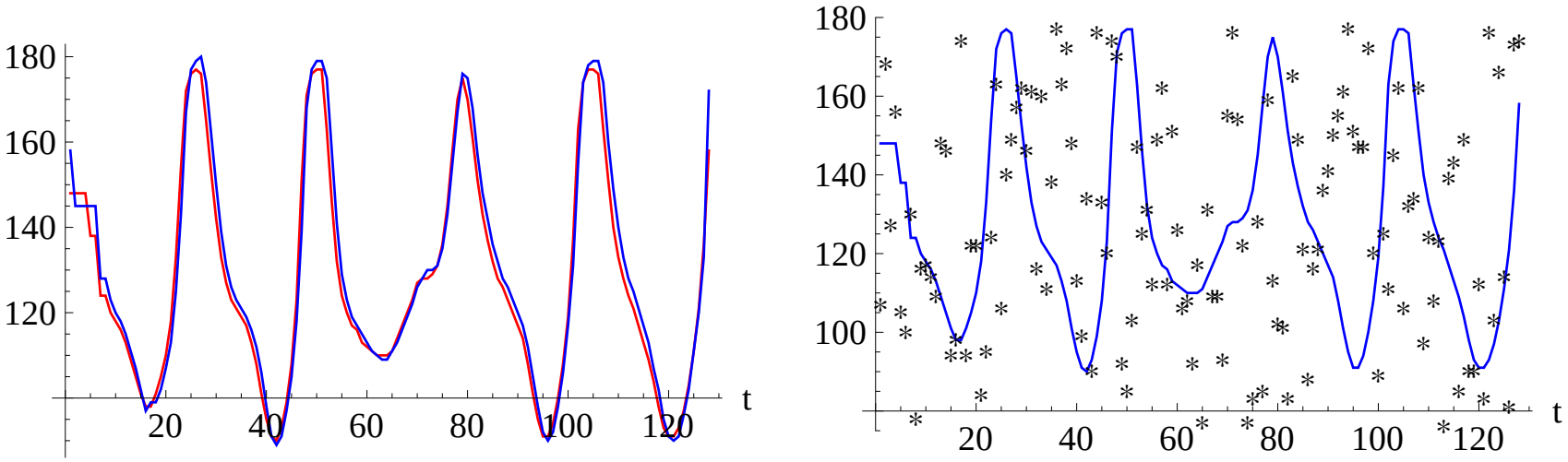

Figure: Data collected from two accelerometers shaken together (left) and the challenge threshold (right)

Result of all comparisons is used as key string (binary vector)

- How to find same string on the other side?

Basic idea (the easy part) — Step 3/3

Matching is achieved whenever the values ($v_{A,i}$ and $v_{B,i}$) are on the same side of the threshold, i.e., $sgn(v_{A,i} - t_i) = sgn(v_{B,i} - t_i)$

Probability of this event is :

$$\Pr\left[v_{A,i} > t_i \wedge v_{B,i} > t_i\right] + \Pr\left[v_{A,i} \leq t_i \wedge v_{B,i} \leq t_i\right]$$

Key observation: matching probability increases with distance between the values

Failure to match is upper bounded by distance (which fortunately drops exponentially - left side figure):

$$\Pr\left[v_{A,i} > t_i \wedge v_{B,i} \leq t_i\right] \leq \Pr\left[\Delta(v_{A,i}, v_{B,i}) > \Delta(v_{A,i}, t_i)\right]$$

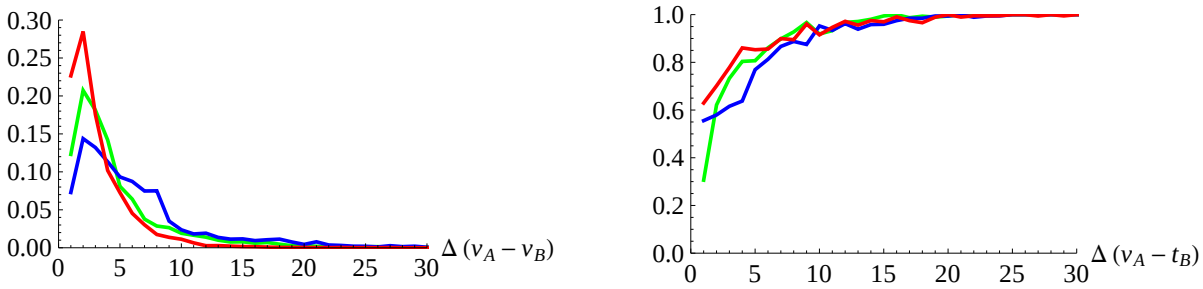

Figure: Drift (left) and success (right) probabilities (WiiMotes)

First improvement: perform heuristic search for a match (depth limited by the bitlength of the key)

Rather than exhaustive search, explore nodes in the search tree in the order of their distance to the threshold (higher distance means lower failure probability ⇒ place higher distance nodes toward the root of the search tree)

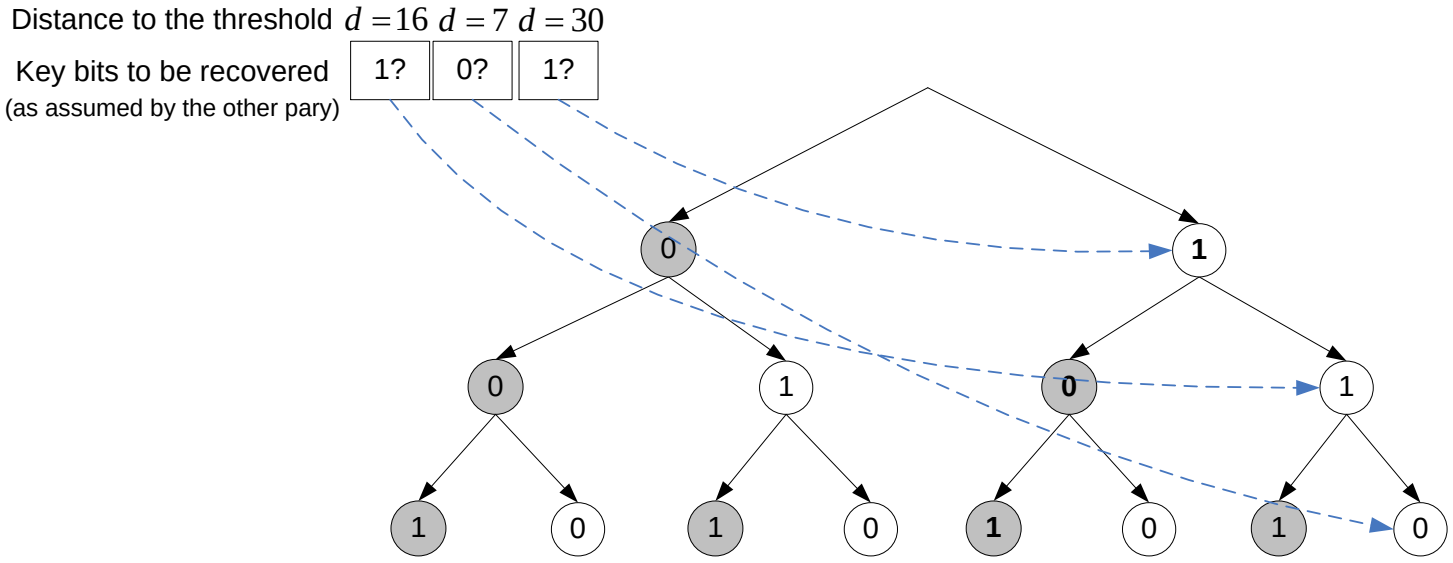

Figure: Heuristic search example (for 3 bits)

Disadvantage: an adversary can do the same

Same search strategy can be done by an adversary on virtually all key extraction procedures from related work (based on the time domain)

Hard to conclude how efficient would this be in the absence of experimental data (high speed camera recordings, image processing techniques, pattern recognition, etc.)

Certainly, more advantageous than exhaustive search

Second improvement: hash the heuristic tree

Hash the key bits in the order of their distances, i.e., the key is extracting taking the order into account $k_B \leftarrow \mathcal{E}^{-1}(\overline{v}_B, \overline{t}_B, \overline{o}_B)$ and $\overline{o}_B$ is sent to the other party (A in this case)

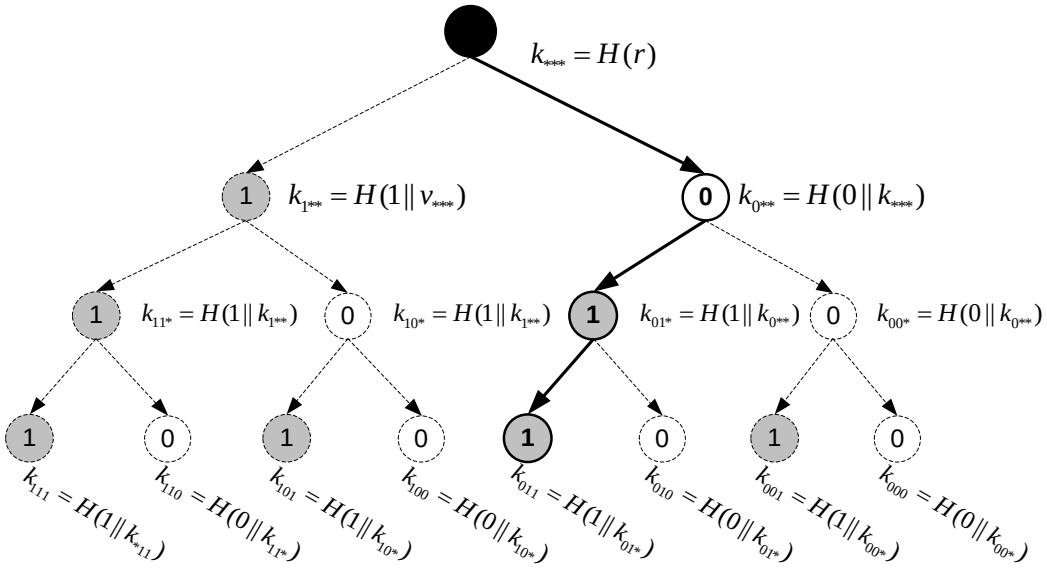

Figure: Hashed heuristic tree

Advantage: the heuristic-search attack is no longer that efficient since the order in the tree is fixed by the other party (you have to re-hash the entire path in the tree)

Efficiency analysis

The average solving time is the sum of probabilities on each level $\times\ 2^i$, i.e., (note that as i increases the p_{d-i} decreases due to the heuristic ordering in the search tree $\Rightarrow$ efficiency)

$$T = d + 2^d - 1 - \sum_{i=0}^{d-1} 2^i \cdot p_{d-i}$$

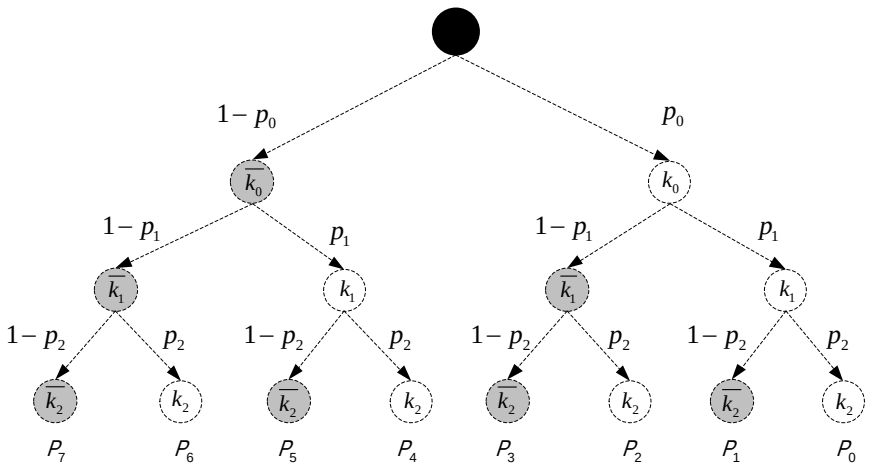

Figure: Heuristic search tree

Advantage: good and fast matching with recent WiiMotes devices

Success rate of over 75% for extracting keys of 64 bits with less than 1024 hashes (cheaper than public-key primitives)

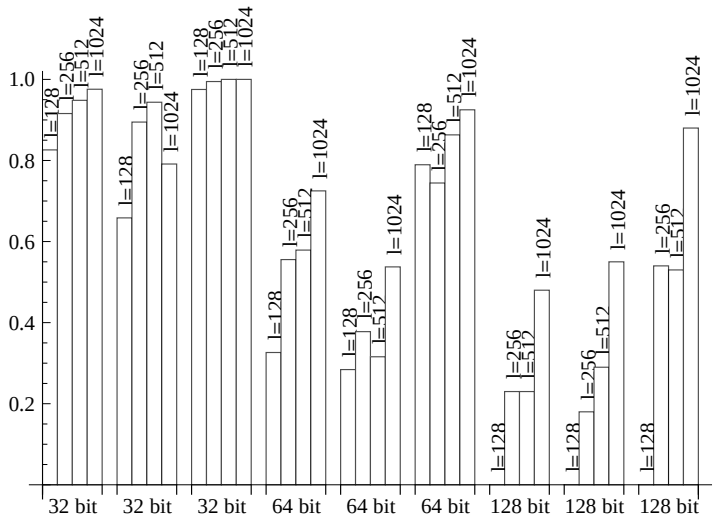

Figure: Success rate for 32, 64, and 128 bit keys with search depth limited to 128, 256, 512 and 1024 hashes

Disadvantage: less efficient on home-made sensors

With the data sets from Open-UAT (after careful data filtering, see figure) success rate is only 20%-40% (more accurate sensors needed)

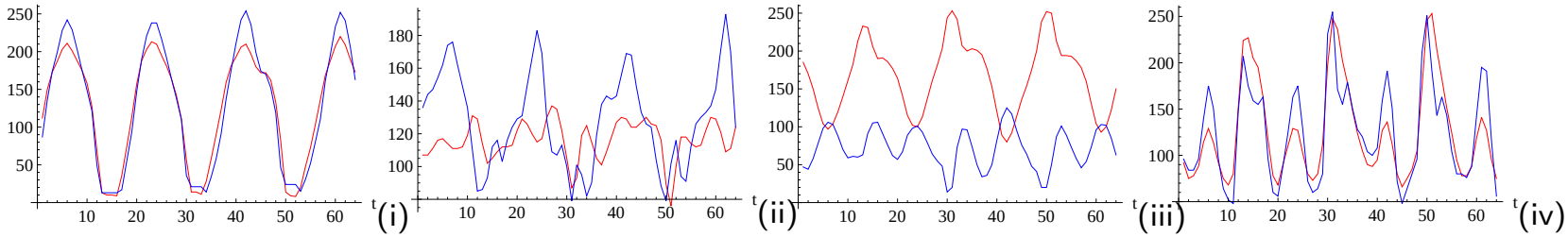

Figure: Example of data collected on the three axes (i), (ii), (iii) and the result after preprocessing (iv)

Comparison to related approaches

The proposal has higher security than some of the previous approaches and is flexible, e.g., the heuristic function (distance) can be changed (yet, failure rate is also higher in case of lower quality sensors)

Protocol	Advantages	Disadvantages	Fix
SAPHE	clear security bound, only symm. functions, flexible heuristic	higher failure rates	high quality sensors
ShaCK	only symm. functions, high success probability	low entropy vectors insecure, less clear security bound	increasing vector size (invariantly leads to higher failure rates)
ShaVe	high success probability	expensive public-key cryptography	N/A
Fuzzy-crypto	reduced failure rate, only symm. functions	special care at choosing system parameters	N/A

Table: A comparison between different methods

Conclusion and future directions

Fast and flexible solution by performing heuristic search

Increased security with the hashed heuristic tree

Further improvements on the heuristic function to speed up computations (e.g., fewer hashes)

Deriving formal proofs of security (either by means of security reductions or automatic analysis) for spontaneous authentication protocols (hard to model attacks: guessing, resource-exhaustion, etc.)