UPPSALA
UNIVERSITET

Sense the Key:
Security Bootstrapping for Small Devices

Christian Rohner

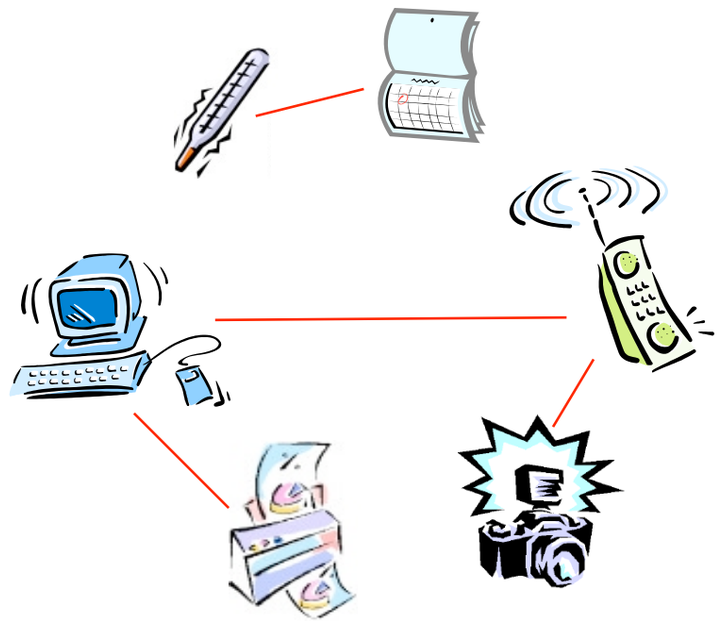
VINNOVA

Stiftelsen för Strategisk Forskning

Spontaneous Networking

- Variety of specialised devices
- Devices offer their functionality as services
- Spontaneous interaction

- Challenges
  - many devices, from big to small
  - no central services
  - individual users



Security Challenge

Security bootstrapping for networked devices:

- No pre-defined relations

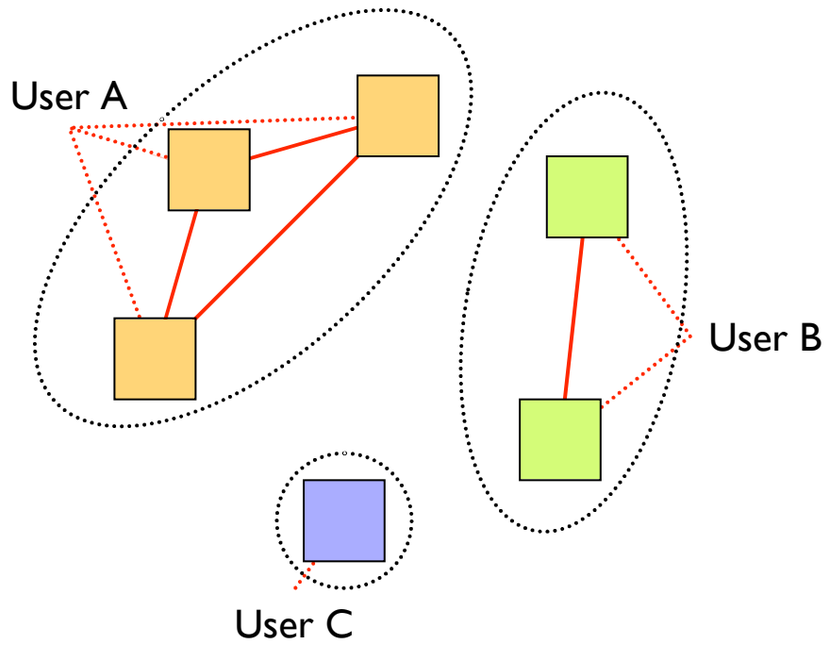
  no secure links, no trust relations, unknown devices

- No central service

  no administrator, no trusted third party

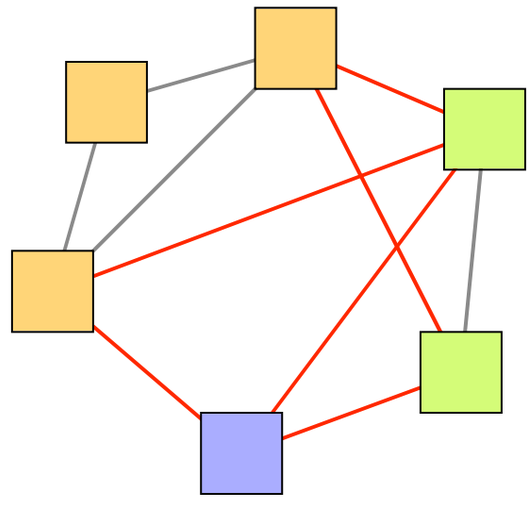
Ownership Model

- Transient association to User



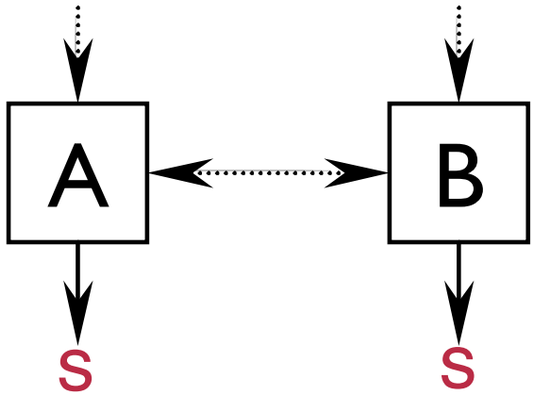
Security Policy

- Introduce other devices
- Assign rights to relations



Pairing

- Key exchange between two particular devices
- Notion of proximity



Diffie-Hellman

$$
\begin{array}{rcl}
A & & B \\
x \in [0..n] & & y \in [0..n] \\
u = g^x \bmod n & \longrightarrow & \\
 & \longleftarrow & v = g^y \bmod n \\
s = (g^y)^x \bmod n & \longrightarrow & s = (g^x)^y \bmod n
\end{array}
$$

... and many variations (be careful!)

Password-authenticated key exchange

- Bluetooth
- Provably secure algorithms [e.g., Katz et al.]

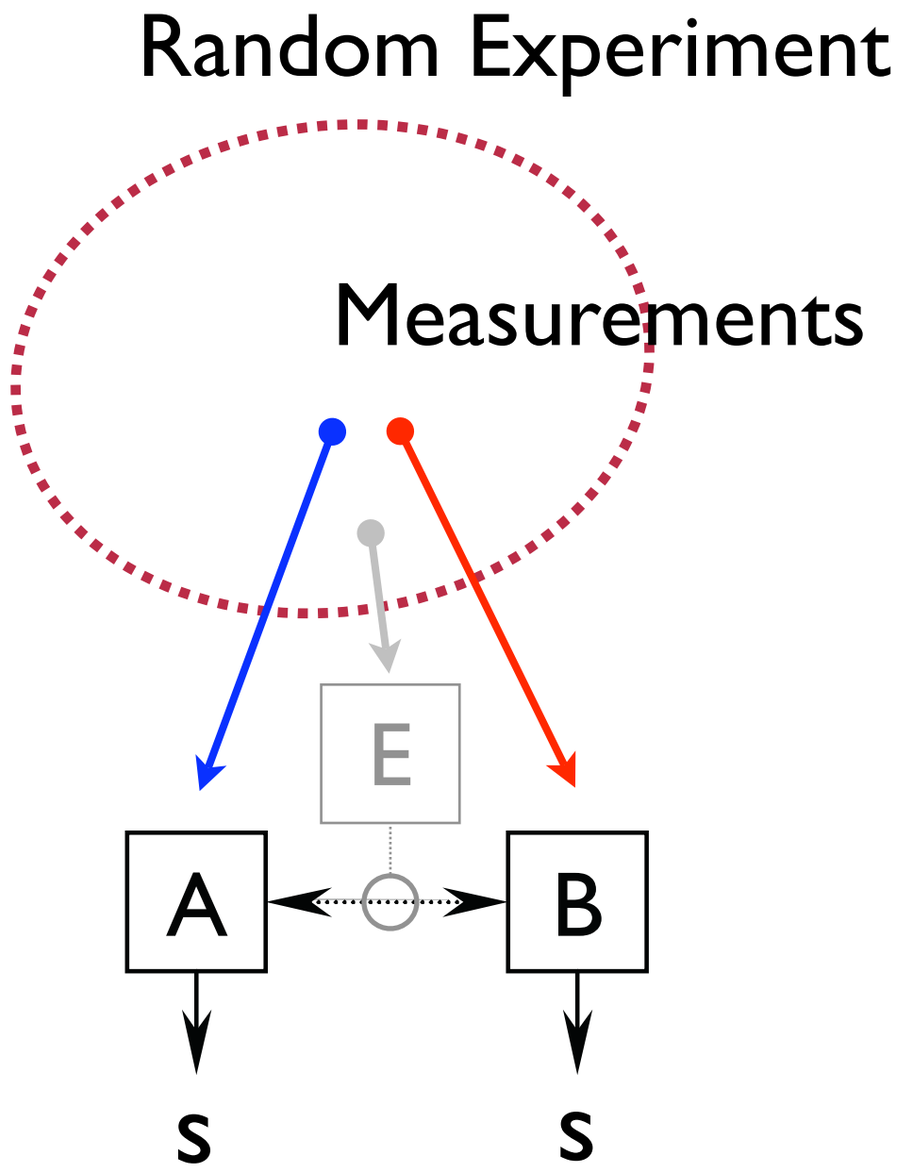
> Pairwise security associations

> Computational expensive

Sense the Key

- Sensors get ubiquitous:
  - Accelerators
  - Microphones
  - etc.

- Proximity
  - > similar measurement
- Adversary E
  - can do similar measurement
  - can listen to communication



Key Agreement by Public Discussion

[Maurer92]

Assumption: A,B,E cannot measure exact value.

1. Advantage distillation
   - remove the differences

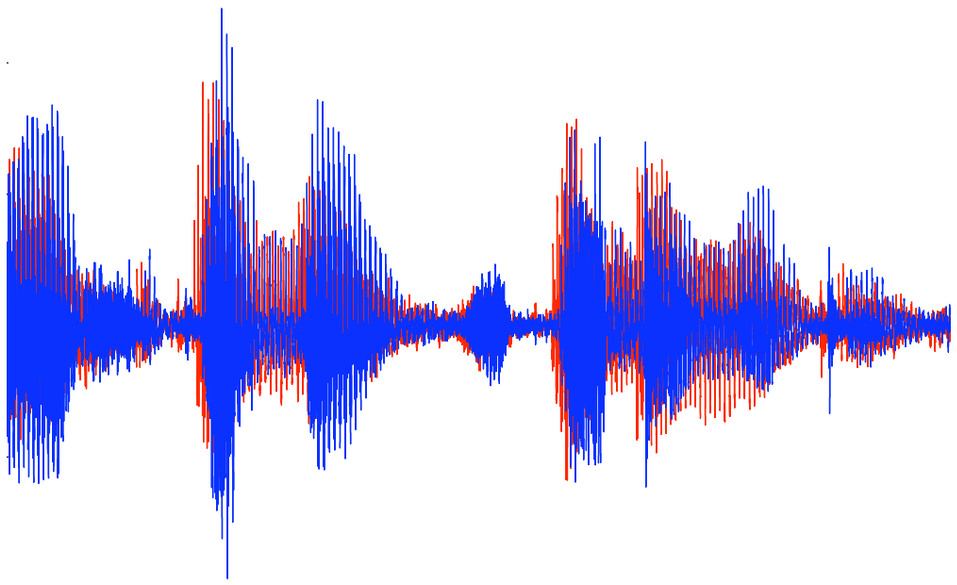
     exchange parity of bit pairs; erase one bit, keep the other if the parity matches.
2. Information reconciliation
   - correct the remaining differences

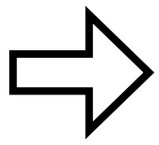
     binary search over larger parity blocks
3. Privacy amplification
   - make small differences big

     hash over the bit string

> Many random bits for a short key...

Requirements





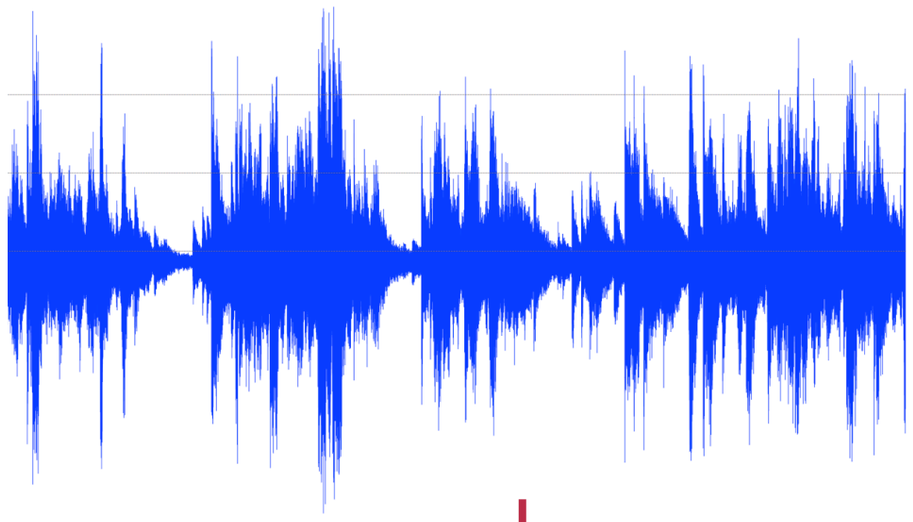
A: 1100101100010100

B: 1000101110010100

- Random bit string
- Similar
  - the more differences, the more bits are required...
- Large key space

Robust Bit String Generation

- Use the characteristics of a Signal



1100101100010100

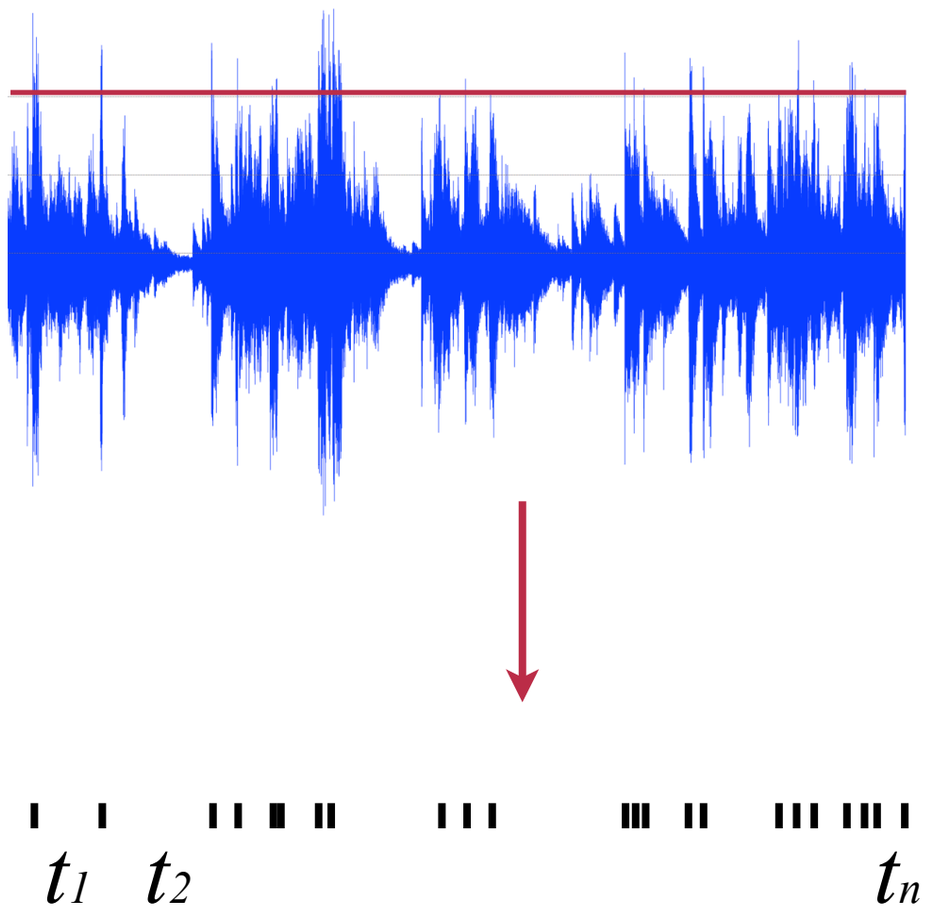
Transformation

- parameters *p1 .. pn*

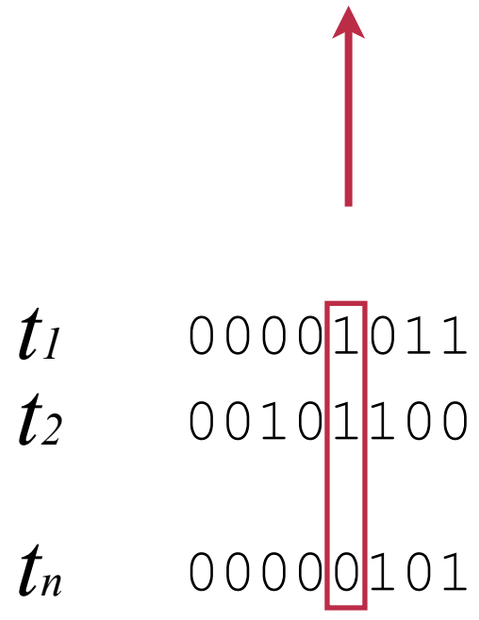
Bit representation

Robust Bit String Generation

- Audio example:
  - time intervals between peeks
    - what defines a peek? threshold, steepnes, etc.

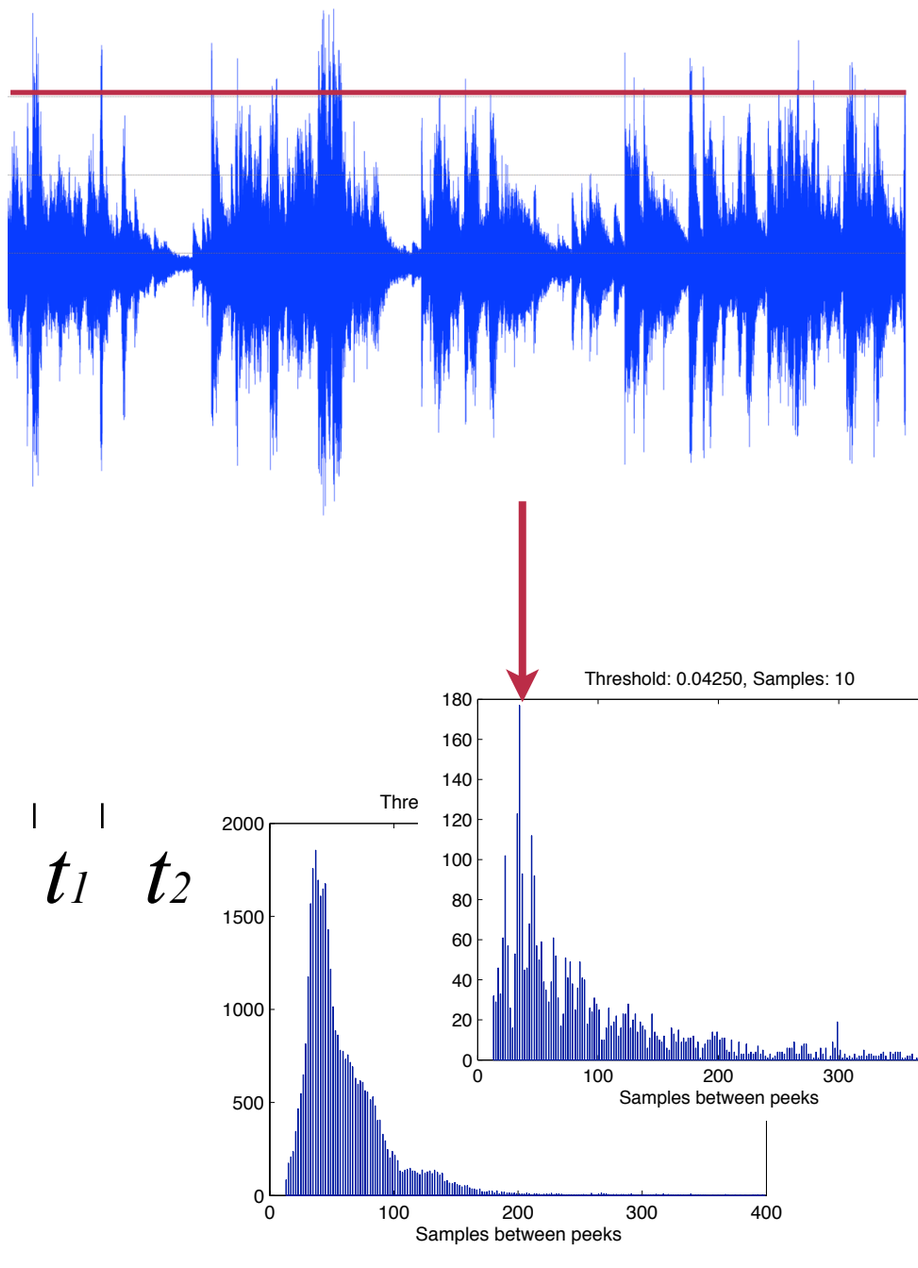


1100101100010100



Robust Bit String Generation

- Audio example:
  - time intervals between peeks (threshold)



1100101100010100

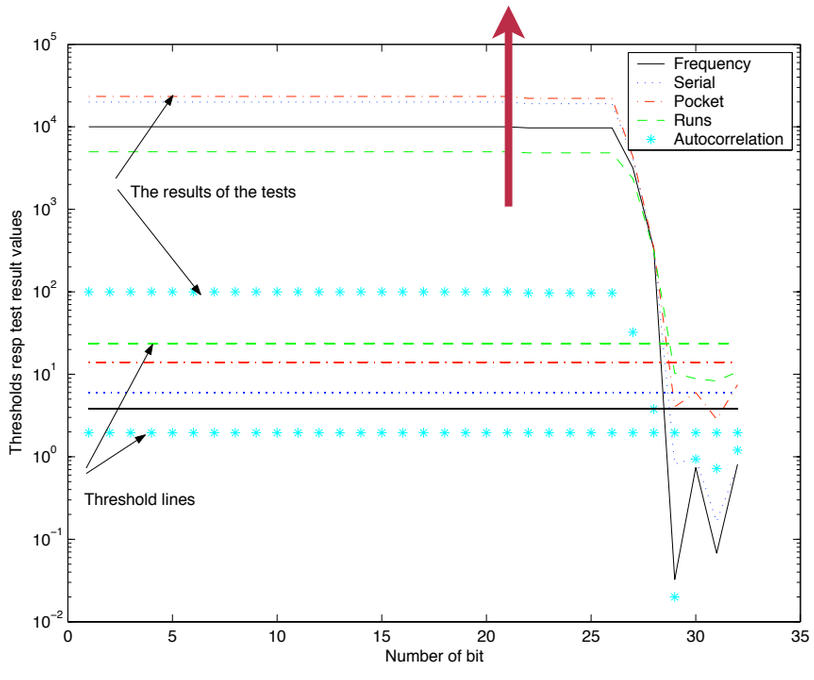
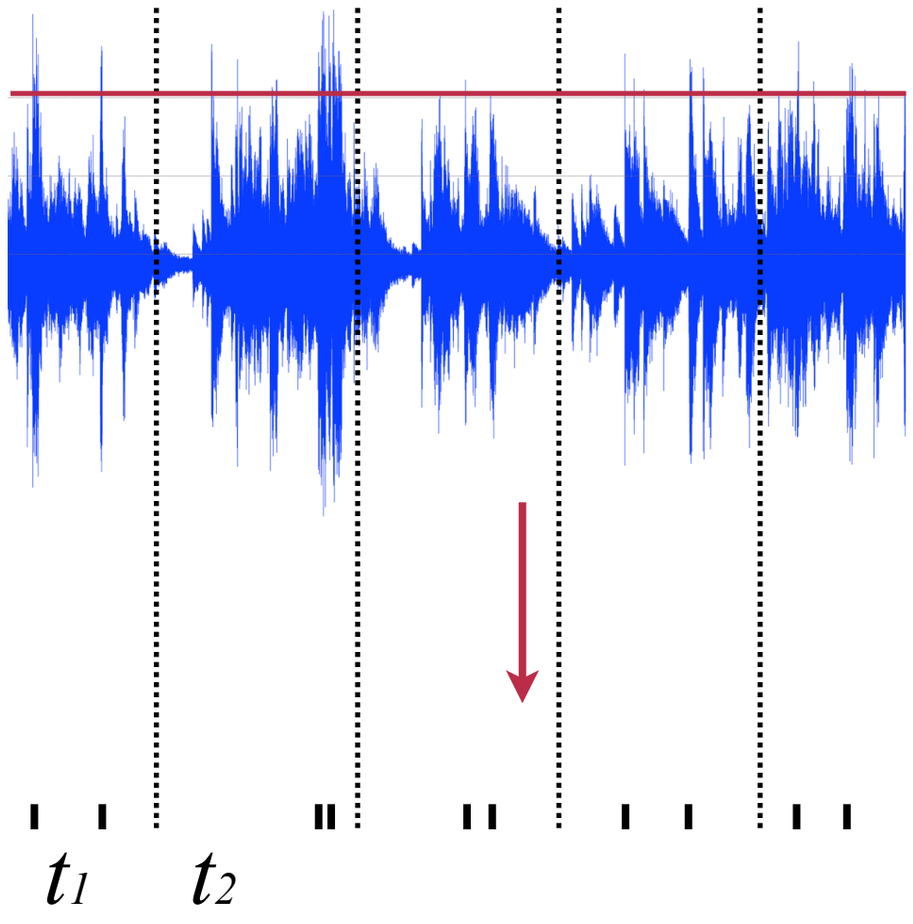


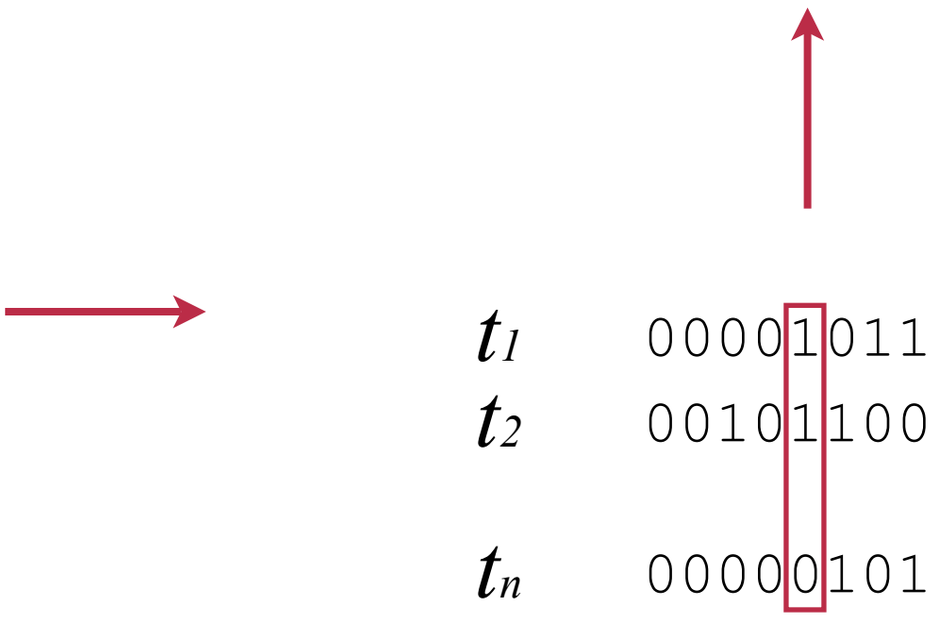
Figure 3.5: The values and the thresholds for significance level 0.05 (logarithmic y-axis)

Robust Bit String Generation

- Audio example:
  - time intervals between peeks
  - *n* highest peeks per time interval (fixed)



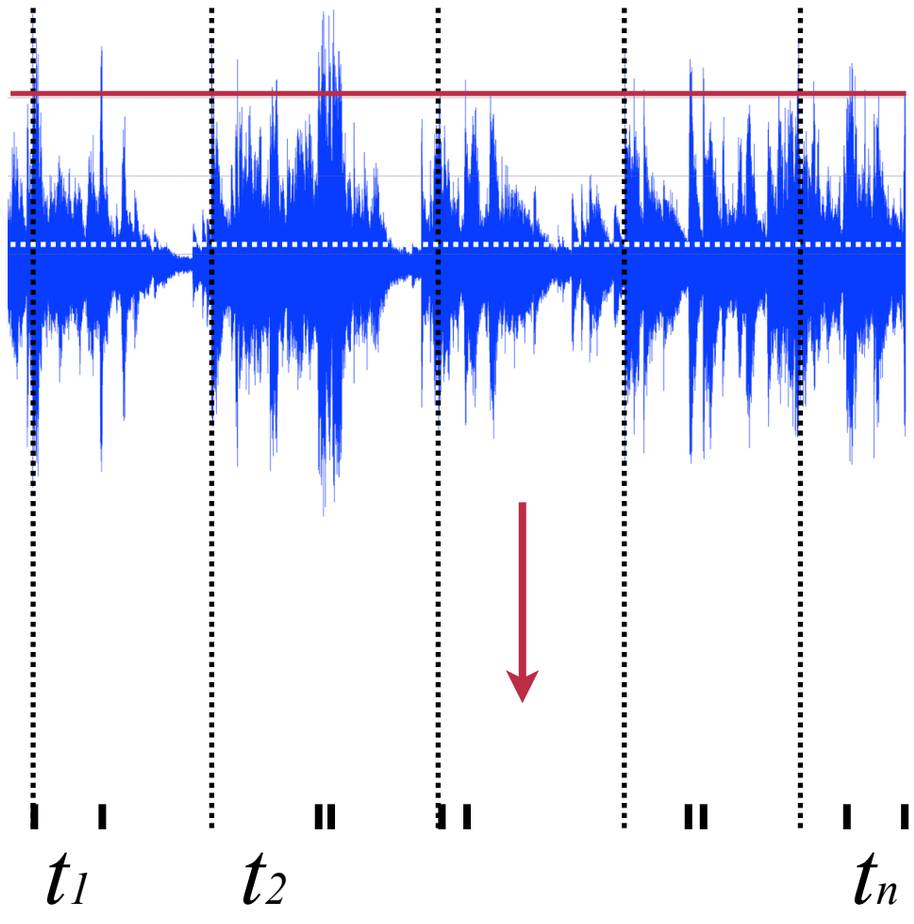
11001011100101100



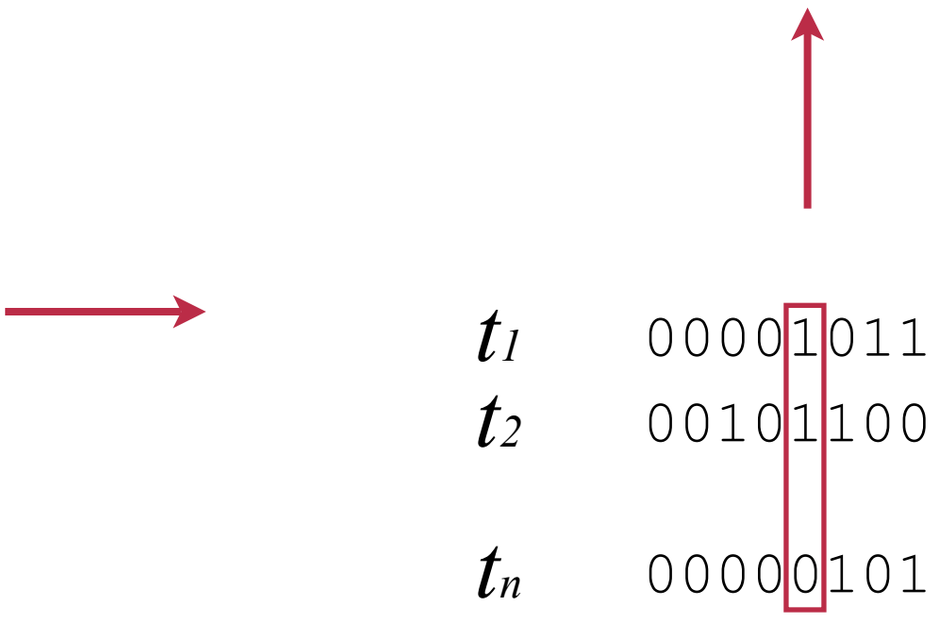
> feedback, synchronisation...

Robust Bit String Generation

- Audio example:
  - time intervals between peeks
  - *n* highest peeks per time interval (adaptive)



1100101100010100



> self-synchronisation?

Robust Bit String Generation

- The sensitivity of a sensor (variance) defines the time it takes to collect sufficient bits.
- Too much sensitivity introduces large measurement “errors”, that is, it is hard to get similar bit strings.
- Significant signal fluctuations simplify synchronisation.

Conclusion

- Challenging engineering task
  - robustness!
- Takes long time to get sufficient bits.
- Computational less expensive, but more communication than conventional approaches.

- Future Work:
  - robustness, other sensors.
  - rigourous analysis on computation and communication requirements.

- Acknowledgement: Anna Otto, Fredrik Bjurefors, Mats Björkman, Per Gunningberg