Selected Security And Privacy Challenges of Implementing Distance Bounding Protocols

Kasper Rasmussen <kasper.rasmussen@cs.ox.ac.uk>

University of Oxford

April 15, 2018

Two problems

Claim

Implementing Distance Bounding correctly is “difficult”

Two of the reasons are that it is hard to make a DB protocol that

- does not leek the (relative) location of the users to others.
- complies with the rigours timing constraints necessary for a tight bound.

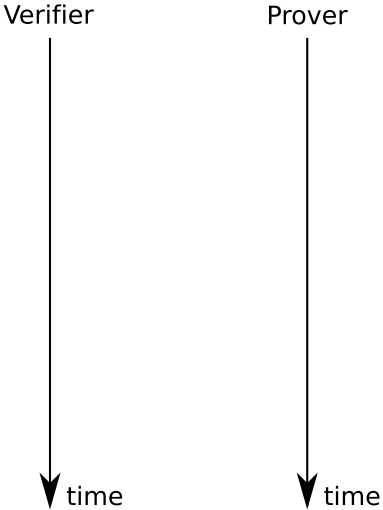
Simplified to

- Space
- Time

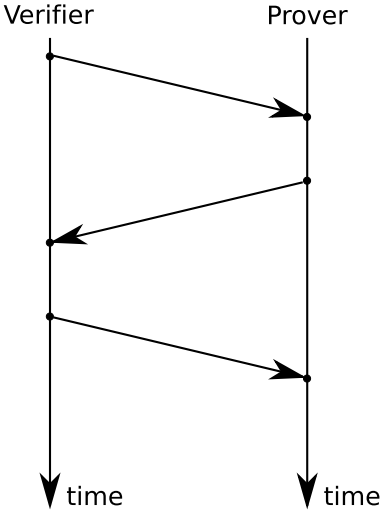
It turns out that the solution to these two problems might be the same.

Location Privacy of Distance Bounding

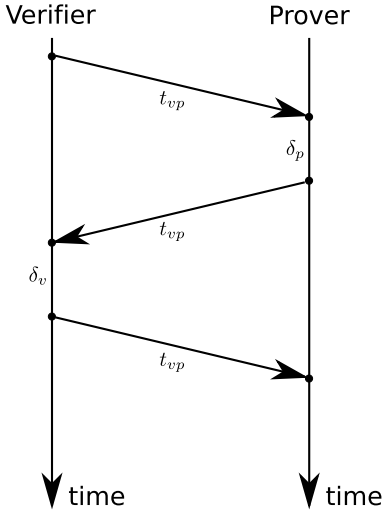
Distance leakage



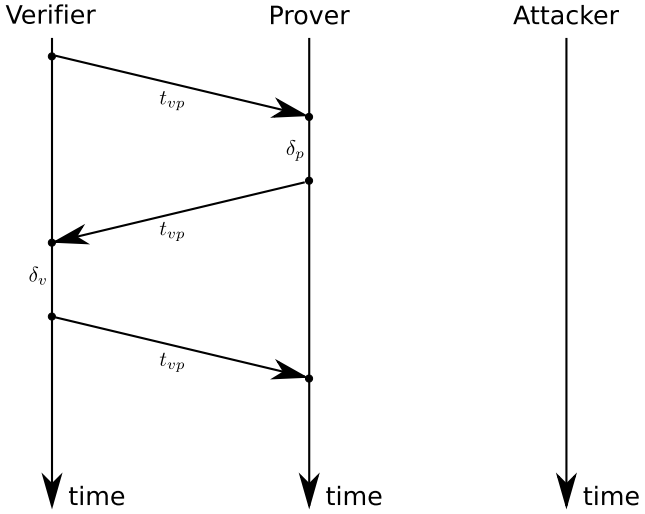
Distance leakage



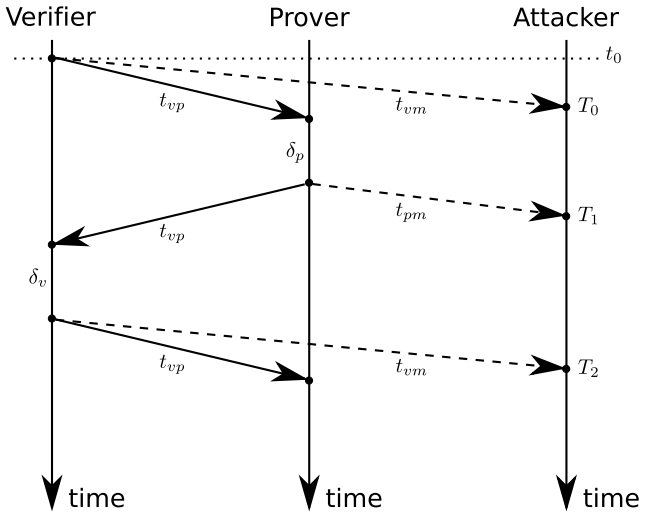
Distance leakage



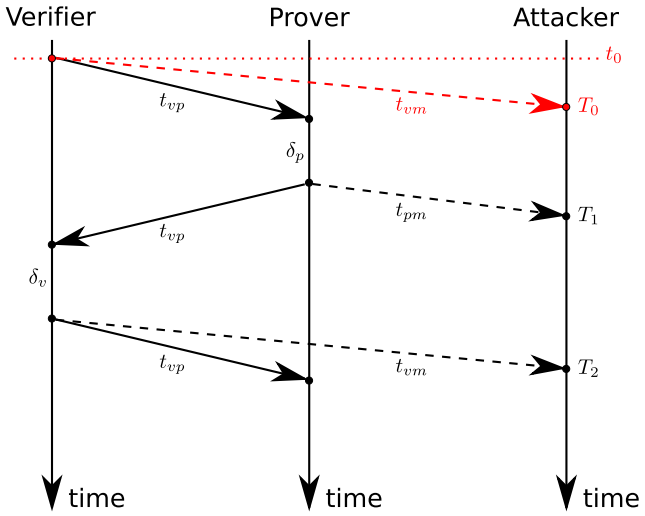
Distance leakage



Distance leakage

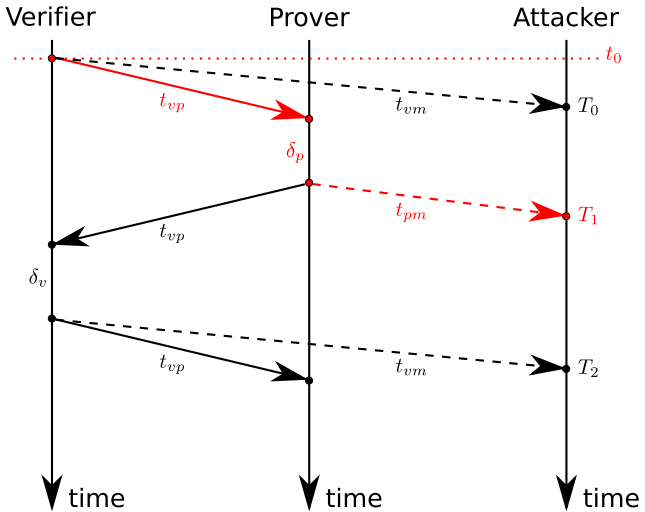


Distance leakage



$$T_0 = t_0 + t_{vm}$$

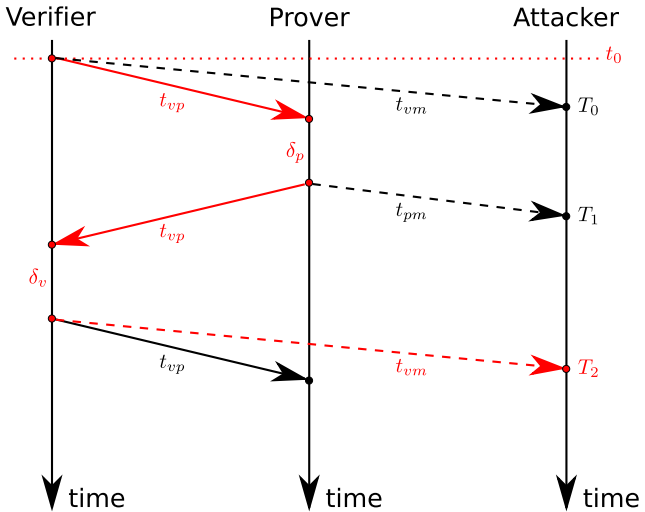
Distance leakage



$T_0 = t_0 + t_{vm}$

$T_1 = t_0 + t_{vp} + \delta_p + t_{pm}$

Distance leakage

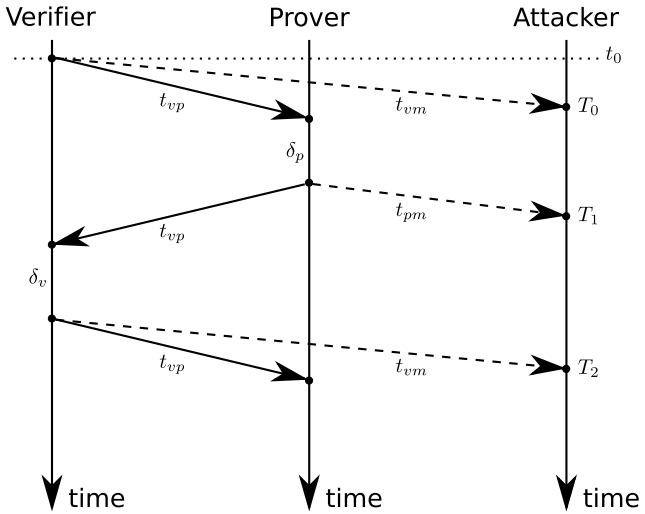


$$T_0 = t_0 + t_{vm}$$

$$T_1 = t_0 + t_{vp} + \delta_p + t_{pm}$$

$$T_2 = t_0 + 2t_{vp} + \delta_p + \delta_v + t_{vm}$$

Distance leakage

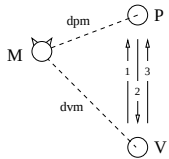


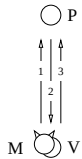
$$T_0 = t_0 + t_{vm}$$
$$T_1 = t_0 + t_{vp} + \delta_p + t_{pm}$$
$$T_2 = t_0 + 2t_{vp} + \delta_p + \delta_v + t_{vm}$$

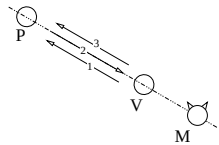
$$t_{vp} = \frac{(T_2 - T_0) - \delta_p - \delta_v}{2}$$

Distance leakage (8 different scenarios)

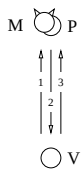
- Attacker needs to capture two messages

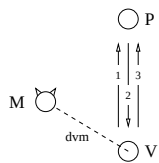






- Attacker needs to capture three messages

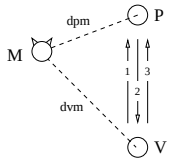


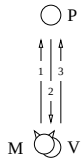


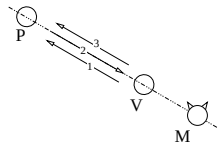
...

Distance leakage (8 different scenarios)

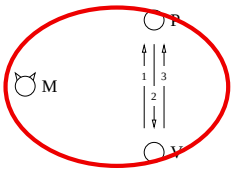
- Attacker needs to capture two messages

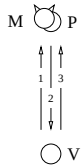


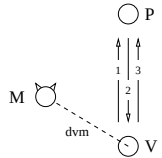




- Attacker needs to capture three messages

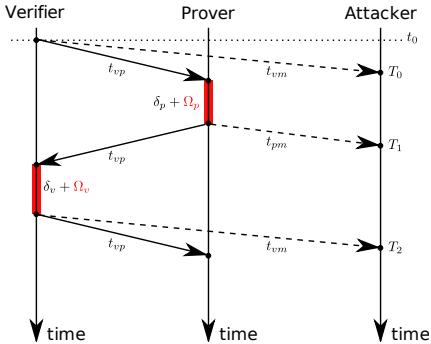






. . .

Add random delay between messages?



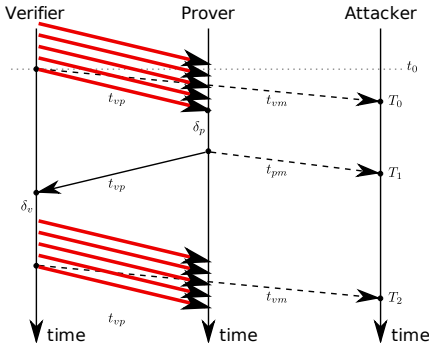
$$T_0 = t_0 + t_{vm}$$
$$T_1 = t_0 + t_{vp} + \delta_p + \Omega_p + t_{pm}$$
$$T_2 = t_0 + 2t_{vp} + \delta_p + \Omega_p + \delta_v + \Omega_v + t_{vm}$$

$$t_{vp} = \frac{(T_2 - T_0) - \delta_p - \delta_v - \Omega_p - \Omega_v}{2}$$

Doesn't work because

- In order for V to find the distance to P the delays must be known to V.
- To prevent P from shortening the distance at least one of the "random" delays must be zero.
- If the delay is zero we are back where we started.

Send multiple challenges?

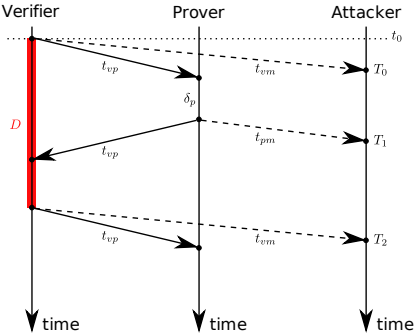


$$T_0 = t_0 + t_{vm}$$
$$T_1 = t_0 + t_{vp} + \delta_p + t_{pm}$$
$$T_2 = t_0 + 2t_{vp} + \delta_p + \delta_v + t_{vm}$$

Doesn't work because

- M can distinguish messages from V and P based on
  - Signal strength, Reception time, Signal fingerprinting.
- M can assume the last message from V triggered response from P.

Send challenges with a fixed interval?



$$T_0 = t_0 + t_{vm}$$
$$T_1 = t_0 + t_{vp} + \delta_p + t_{pm}$$
$$T_2 = t_0 + t_{vm} + D$$

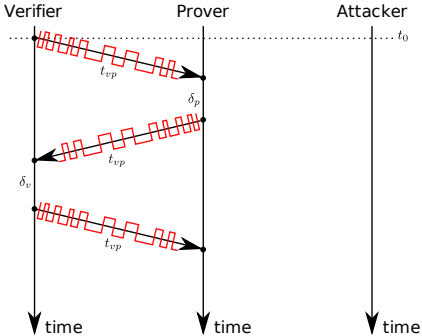
$$(T_2 - T_0) = D$$

- Prevents information leakage in scenarios where distance leakage requires three messages!

Doesn't (always) work because

- Sometimes you only need two messages, say, when M is close to V.

Hide the transmission of messages? (DSSS/FH)



$T_0 = ?$

$T_1 = ?$

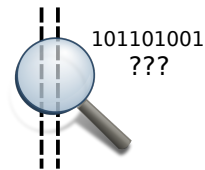
$T_2 = ?$

Doesn't work because

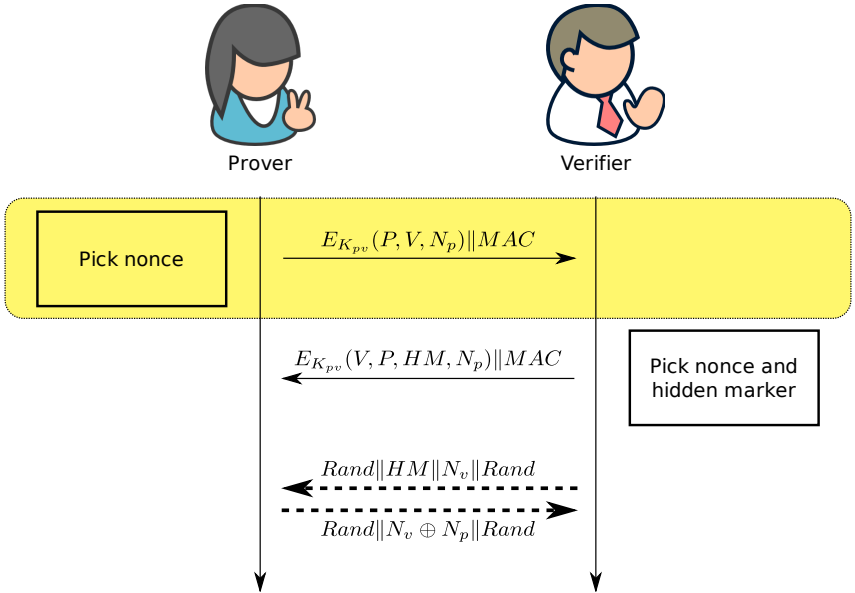
- With DSSS acquisition time is non-deterministic.
  - Rough synchronization to synchronize the receiver to within one chip.
  - Phase locked loop (PLL) performs fine grained synchronization.
- With FH it is trivial to find the messages with post processing.

Our Solution

- Assumes a shared key between the prover and verifier.
- Uses two continuous transmissions (streams) to hide the transmission times.
  - Attacker only sees continuous data.
  - Embedded within the stream is a hidden marker.
  - Following the *HM* is the challenge (nonce).
  - Verifier replies using his own stream.



Our Protocol



Our Protocol

Prover

Verifier

Pick nonce

$E_{K_{pv}}(P, V, N_p) \| MAC$

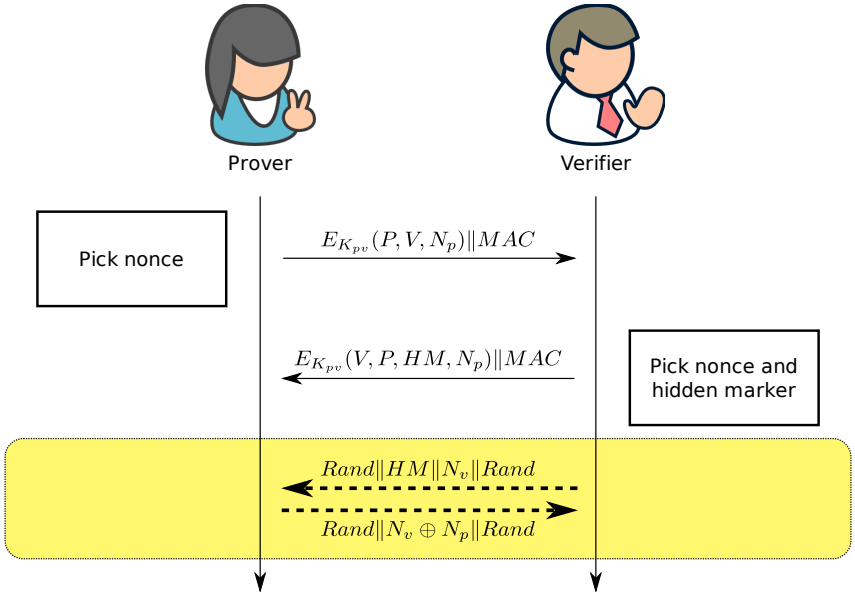
$E_{K_{pv}}(V, P, HM, N_p) \| MAC$

Pick nonce and hidden marker

$Rand \| HM \| N_v \| Rand$

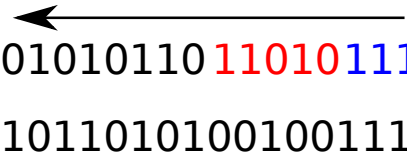
$Rand \| N_v \oplus N_p \| Rand$

Our Protocol



An example

Prover



Verifier

An example

Prover

11101010100011101110011001

01010100100111101101100

Verifier

An example

Prover

01010100110011101110011001

10101010100100111101110

Verifier

An example

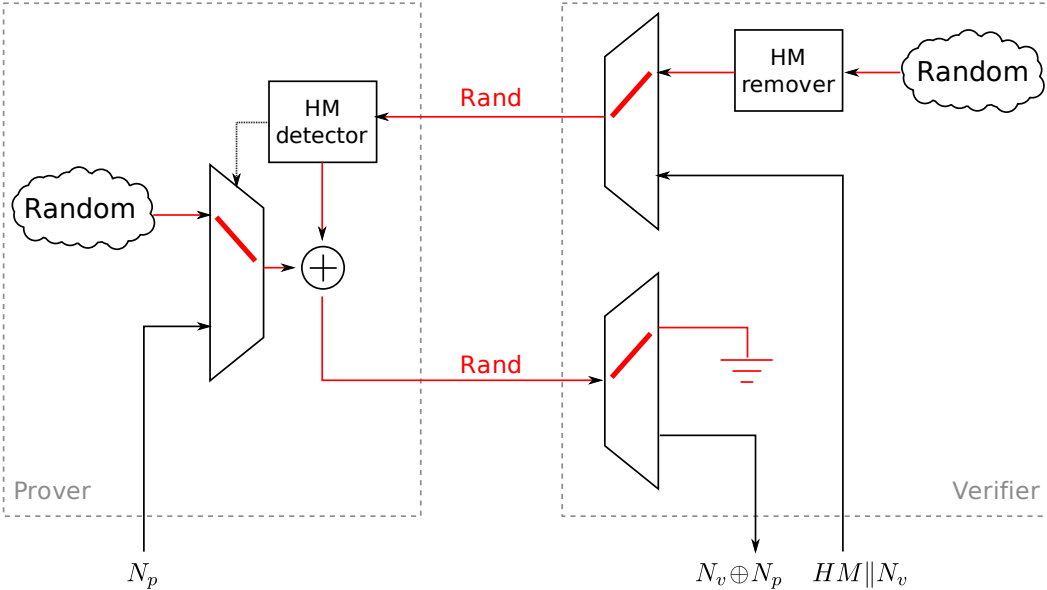
Prover

1101110011010000111011

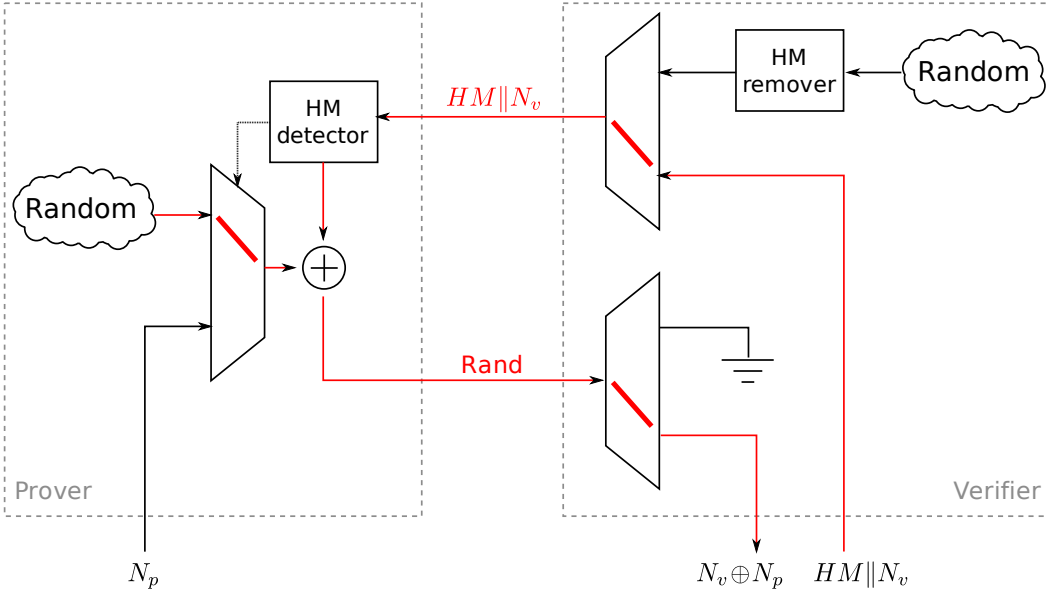
0101011011010100101101

Verifier

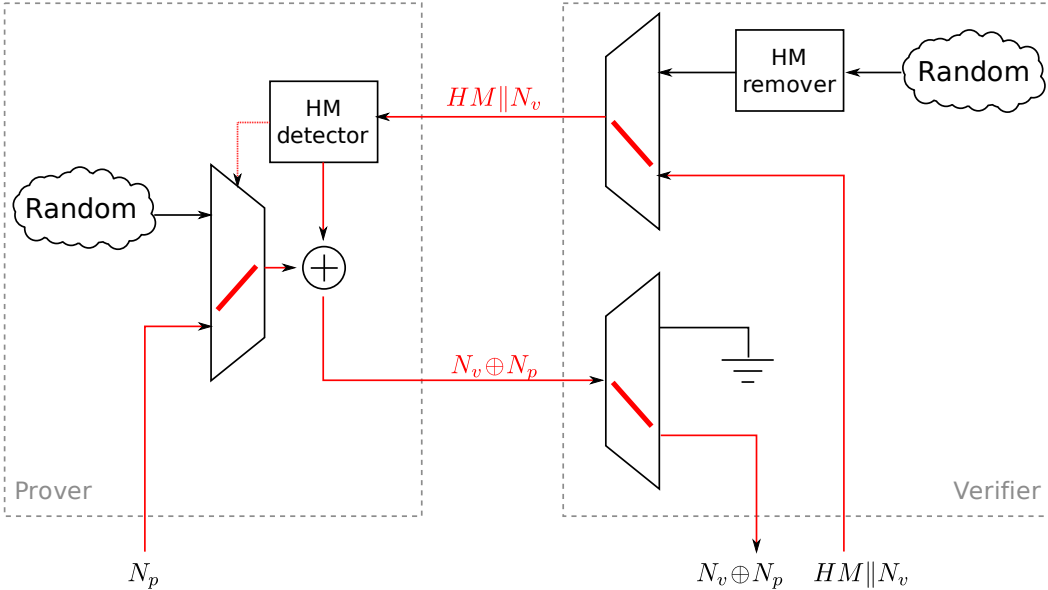
A schematic view



A schematic view



A schematic view



Properties of our solution

- Original distance bounding properties are preserved.
  - Provides a distance bound.
  - Prover can not shorten distance.
- An attacker can not initiate a “rapid message exchange” and get the distance that way.
- A passive attacker can not obtain the distance based on the communication between V and P.

What about this processing function?

(is xor really the right choice)

Processing Function Speed

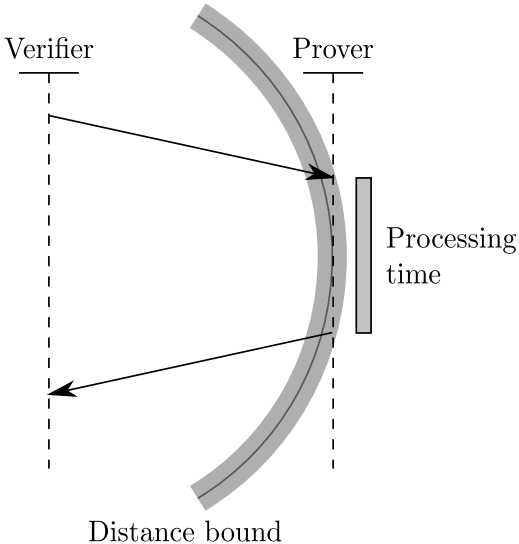
- The verifyer computes the distance to the prover as

$$d = \frac{t_2 - t_1 - \delta_p}{2} \cdot c$$

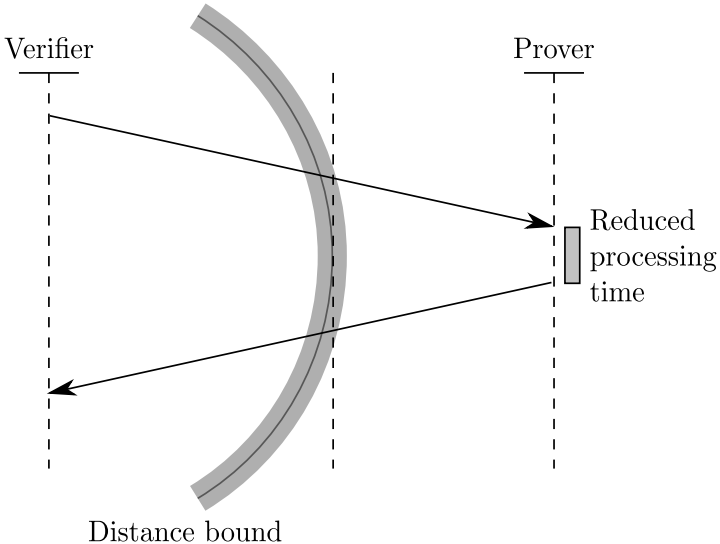
- δ_p must be a public value.

A malicious prover can potentially cheat by as much as $d_{error} = \frac{\delta_p \cdot c}{2}$

Processing Function Attack

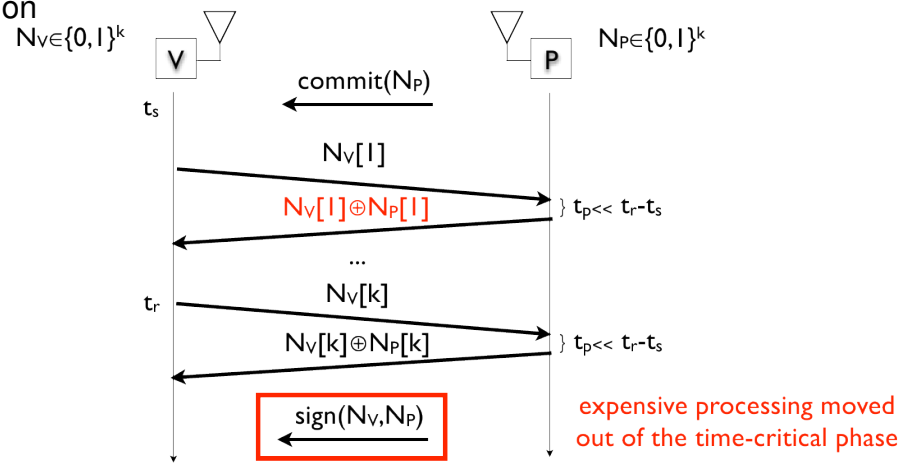


Processing Function Attack



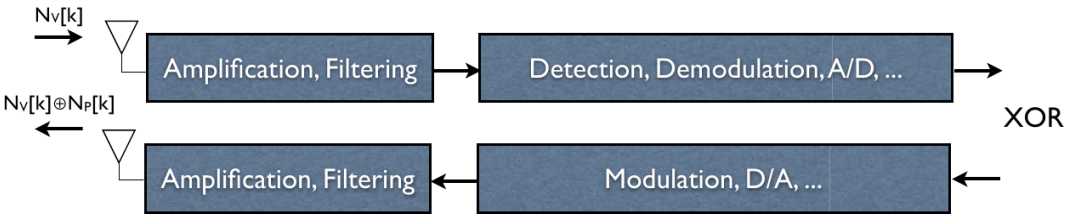
Processing Function Choices

- sign(), MAC(), hash(), enc(). Slow!
- XOR
- Selection



XOR and Selection

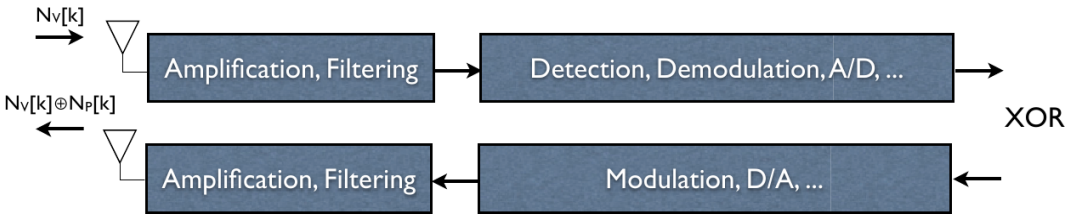
XOR and Selection are still too slow for DB.



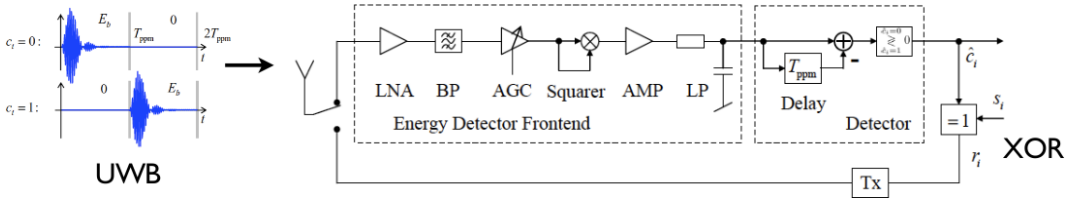
Long symbol lengths are problematic.

XOR and Selection

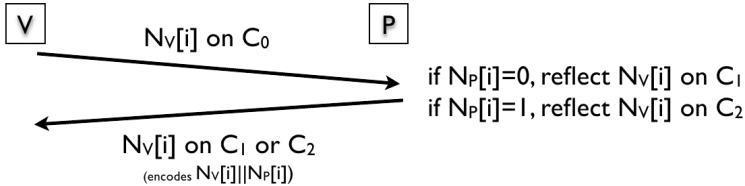
XOR and Selection are still too slow for DB.



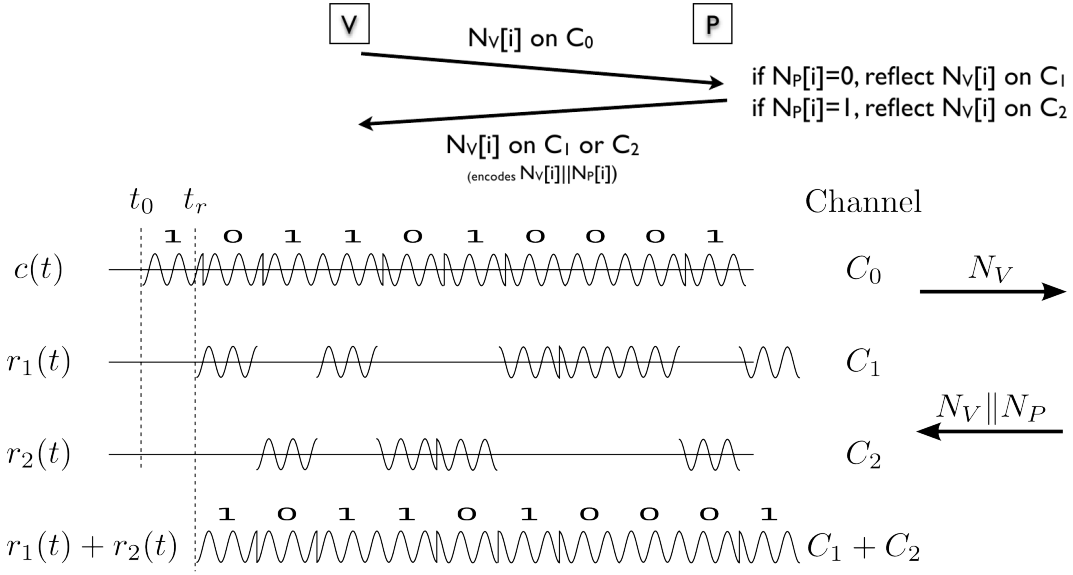
Long symbol lengths are problematic.



Challenge Reflection with Channel Selection (CRCS)

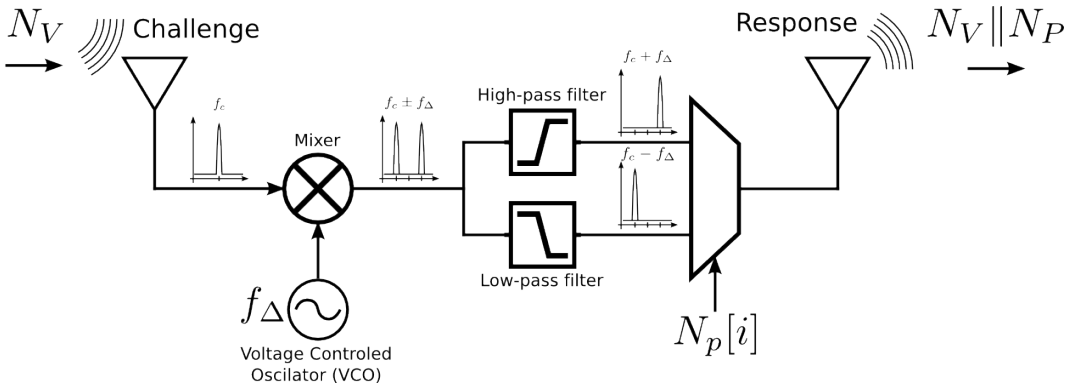


Challenge Reflection with Channel Selection (CRCS)



Implementation of CRCS

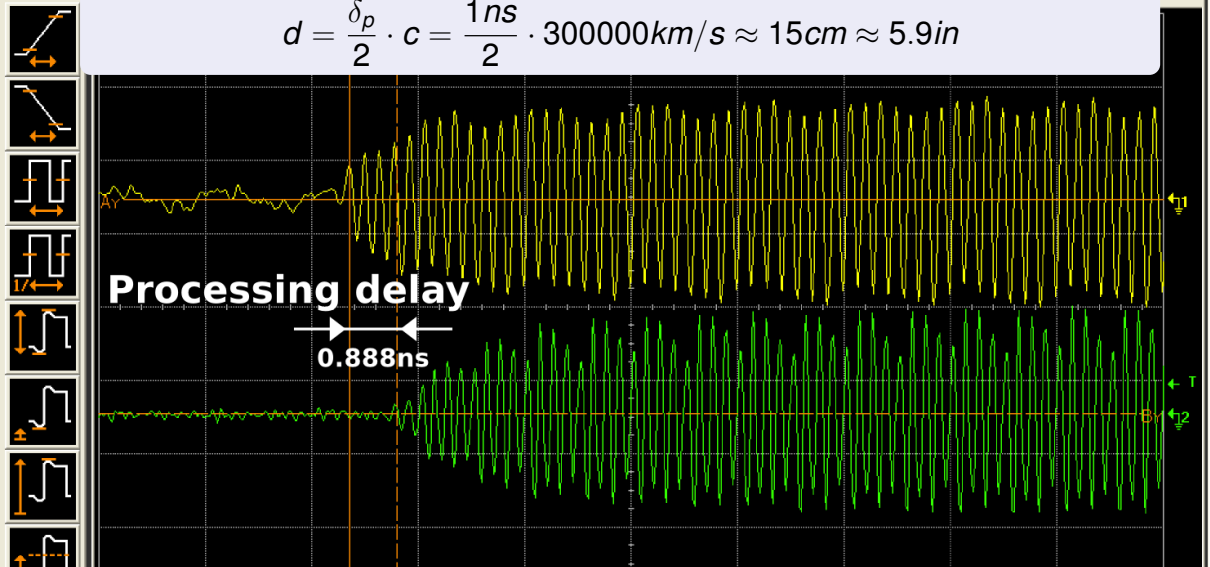
- CRCS enables receive + processing + send in $t_p < 1ns$



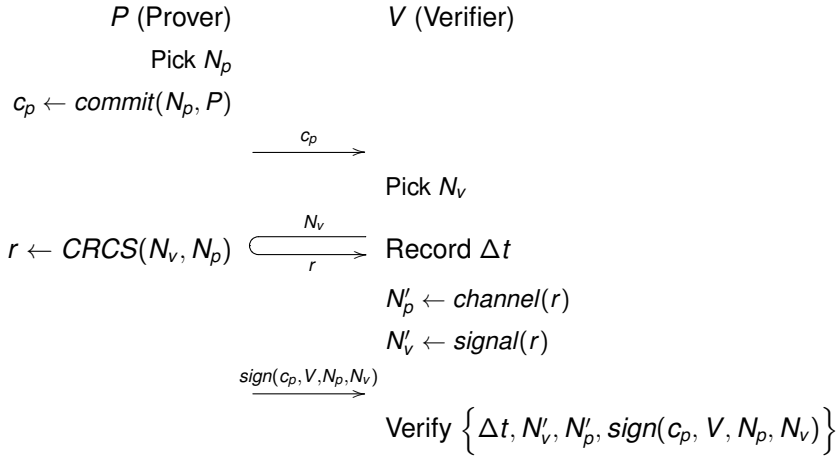

File Control Setup Trigger Measure Analyze Utilities Help
21 Jan 2010 4:54 PM
Acquisition is stopped.
40.0 GSa/s 1.00 kpts
8 GHz
On
500 mV/
On
200 mV/
On
On
Processing delay
0.888ns

Corresponding to a maximum window for the attacker to cheat of:

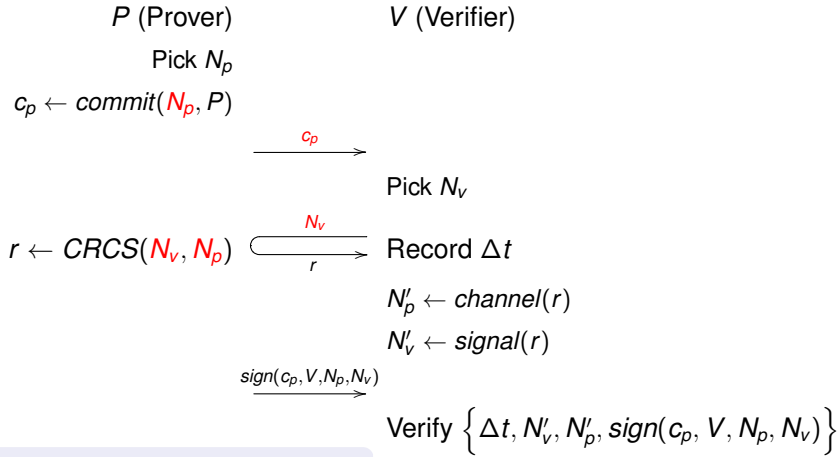
$$d = \frac{\delta_p}{2} \cdot c = \frac{1ns}{2} \cdot 300000km/s \approx 15cm \approx 5.9in$$



Distance Bounding Protocol Using CRCS

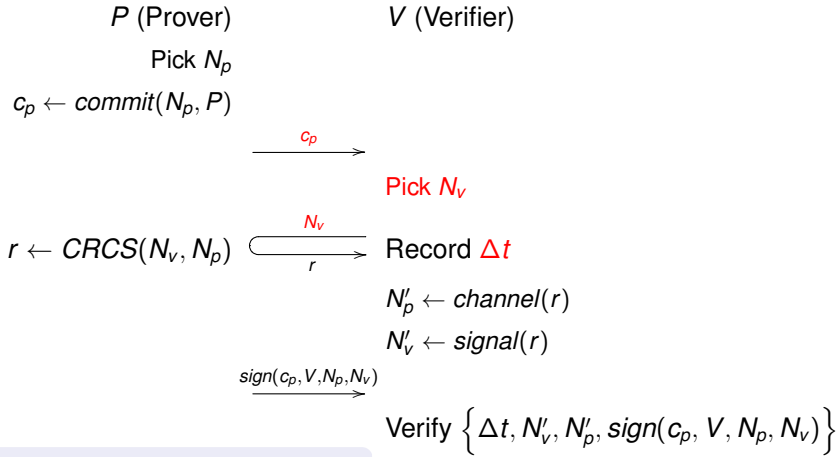


Distance Bounding Protocol Using CRCS



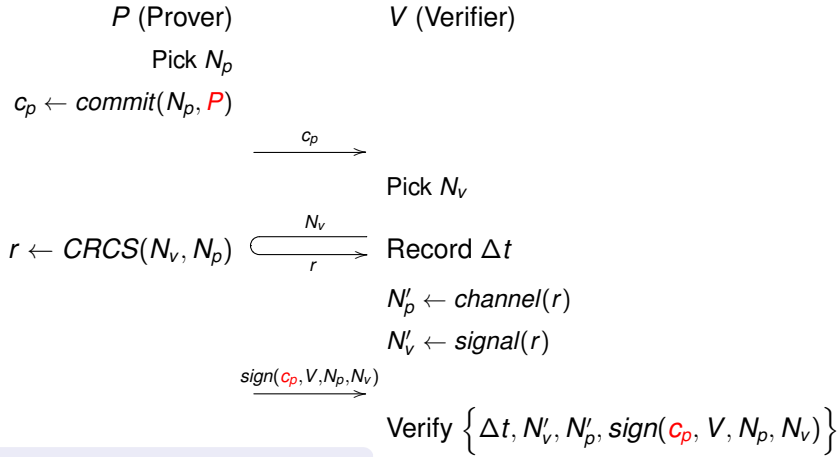
Secure against Distance Fraud

Distance Bounding Protocol Using CRCS



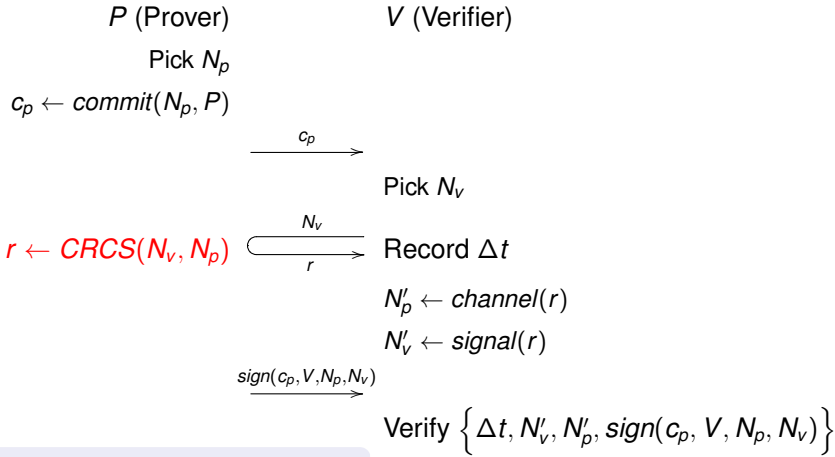
Secure against Mafia Fraud

Distance Bounding Protocol Using CRCS



Secure against Distance Hijacking

Distance Bounding Protocol Using CRCS



Processing delay close to zero.

Steam Based DB

“Steam Based” DB looks promising.

- The answer to both the privacy issue and the processing/response time issue is a stream based protocol.
- Requires full duplex communication.

Thank you for your attention

kasper.rasmussen@cs.ox.ac.uk
`http://www.cs.ox.ac.uk/people/kasper.rasmussen/`