ESC - Echternach - 2008

A Security Architecture for Body Area Networks

Marijke De Soete†, Michael Peeters†, Dave Singelée‡

† NXP Semiconductors - Leuven,
‡ University of Leuven – COSIC

ESC Echternach Crypto Seminar, 7th – 11th Jan, 2008

michael.peeters@nxp.com

Overview

- Part I – Security Architecture for Body Area Network

  Marijke De Soete (NXP), Michael Peeters (NXP), Dave Singelée (KUL COSIC)

- Part II – Looking Quickly for Trails…

  Guido Bertoni (ST), Joan Daemen (ST), Michael Peeters (NXP),
  Gilles Van Assche (ST)

Part I
Body Area Network
Introduction

Security Architecture for Body Area Networks

- Body Area Networks are interesting use case for symmetric crypto!
  - Tight constraints
    - (Ultra)-low power devices
    - Low-gate devices
  - Heterogeneous devices with different computation power
    - can be exploited in designs
  - Security requirements
    - Eg. health care requires *confidentiality*, *integrity*, and also *privacy*
- And because security in BAN is a **challenge!**

1st Use Case – Home Monitoring

Store & Forward

Patient

BAN Sensors

e.g. CoolFlux ECG

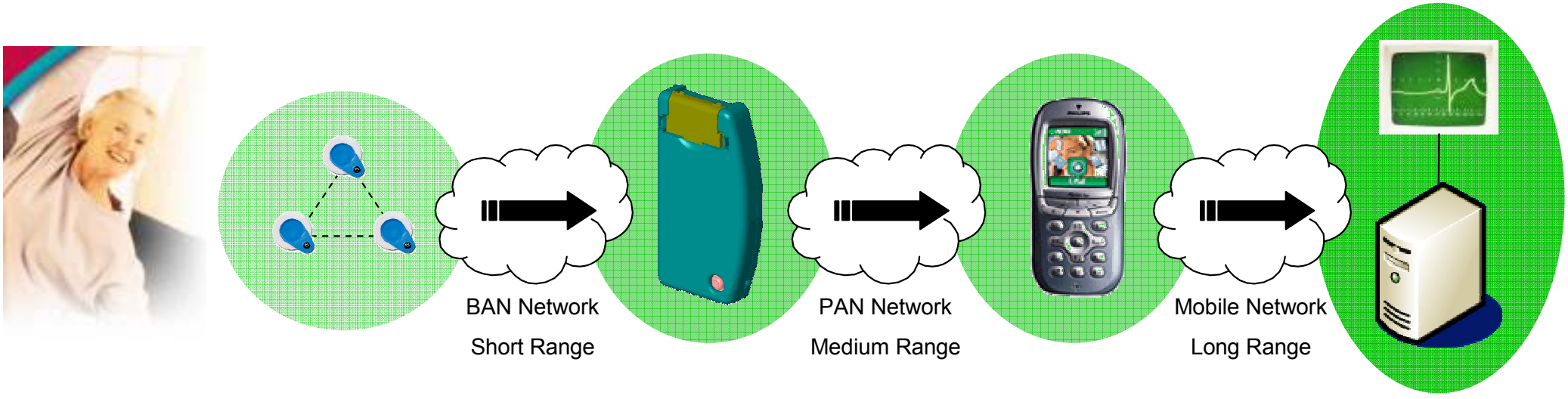
BAN-PAN Bridge

e.g. CF - BlueTooth

PAN-enabled GSM

e.g. BlueTooth GSM

Server



Ultra-Low Power

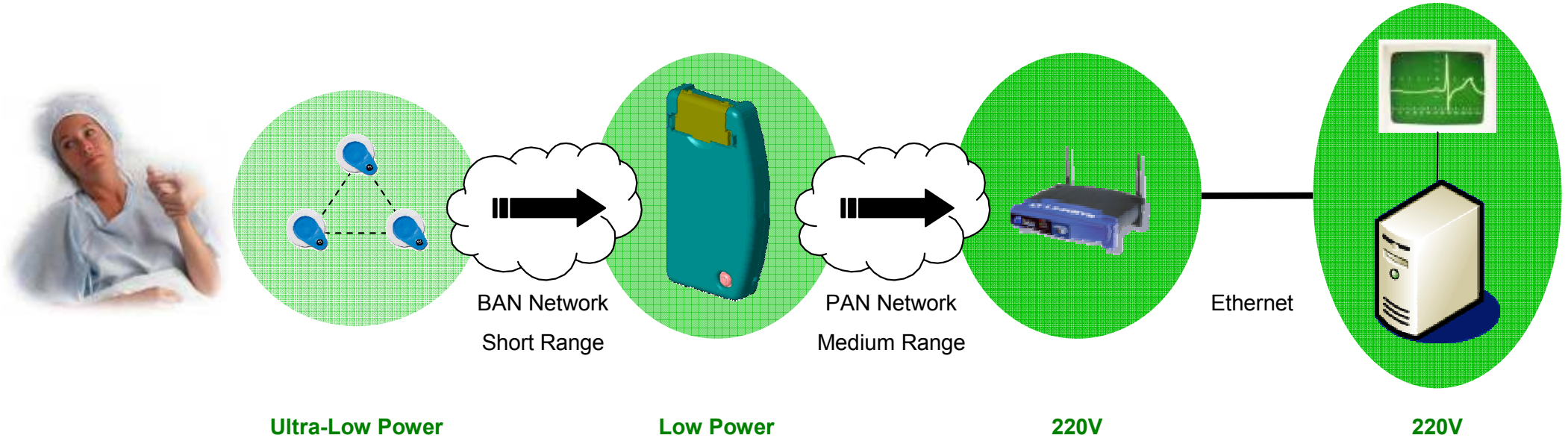
Low Power

Low Power

220V

2nd Use Case – Intra-Mural Monitoring

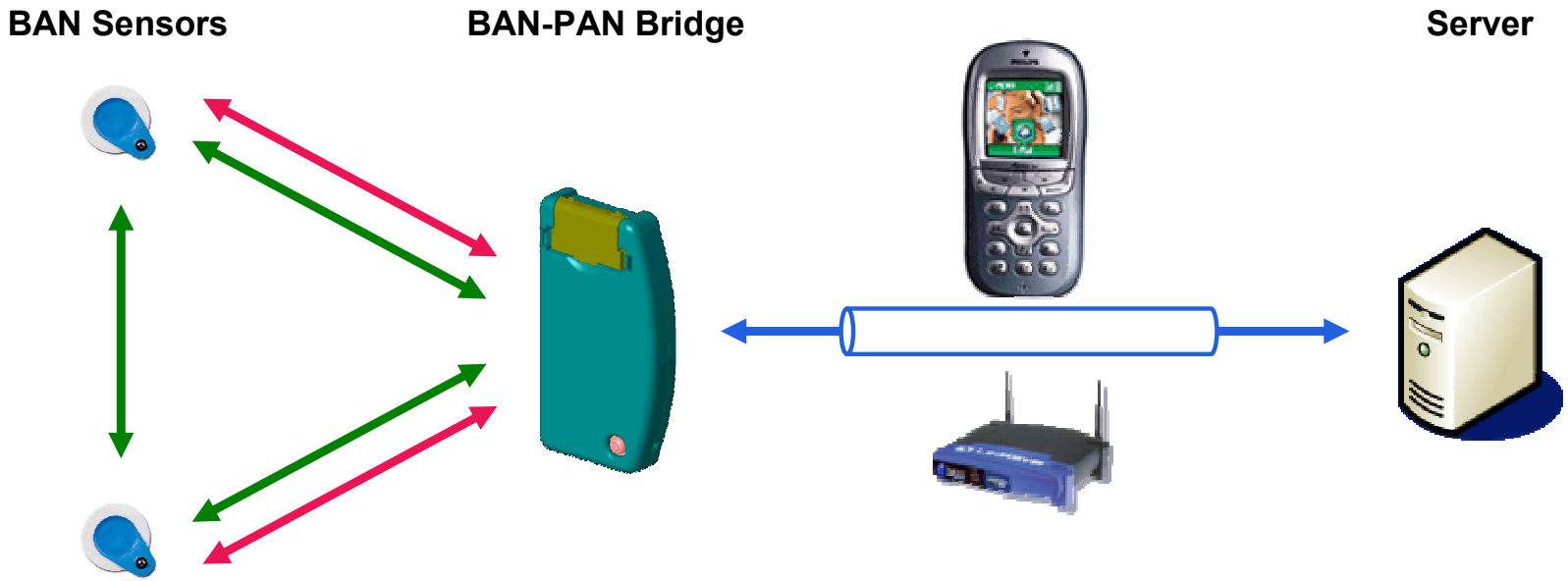
Real-Time



Threat Analysis

- For this talk we only focus on the BAN
- Most relevant
  - Message Eavesdropping / Injection / Modification
  - Privacy infringement / Tracking
  - ID Spoofing
- Not in scope
  - Side-channel attacks
    - Eg. attacks through another application running on the bridge, DPA, …
  - Denial-of-Service
  - Corruption of bridge
  - Corruption of sensors
    - … but need reaction mechanisms

Architecture Overview



- 3 communication channels
  - **Bridge – Server Channel**
    - Both for application data and management protocols.
    - Tunnel through AP or mobile…
  - **BAN Management Channel**
    - All BAN management protocols
  - **BAN Application Data Channel**
    - Application data (sync or async), application commands, …

High security

Ultra low power

Part I
Body Area Network

Bridge – Server Channel

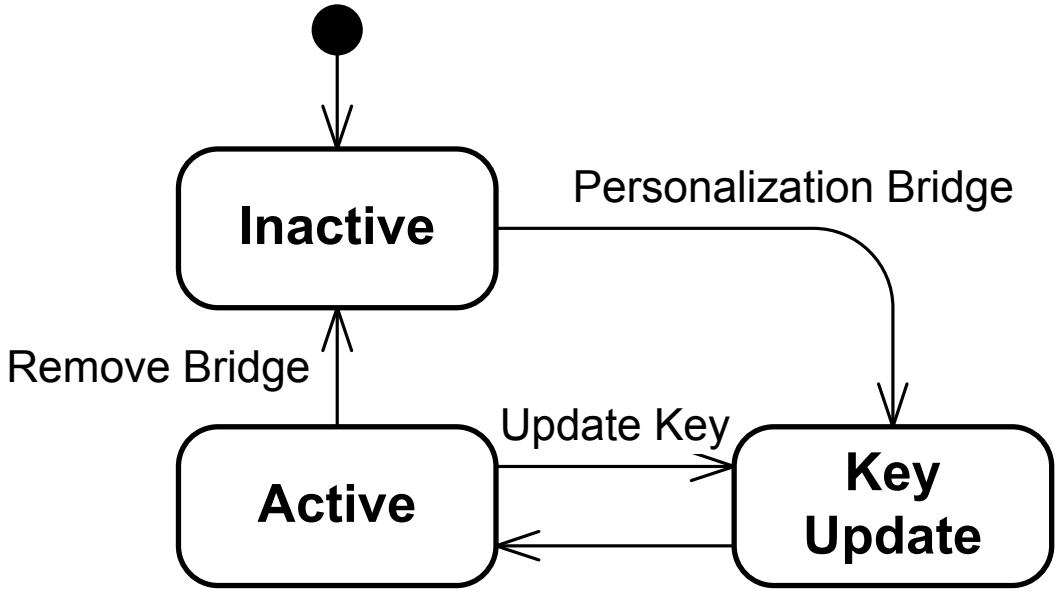
BAN Management Channel

BAN Application Channel

Bridge – Server Channel

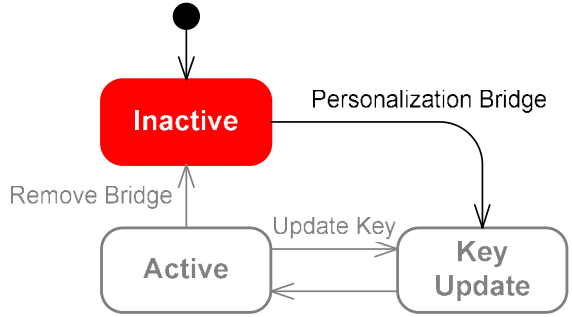
- Same channel used for exchanging BAN management and application messages

- Simple 3-state FSM

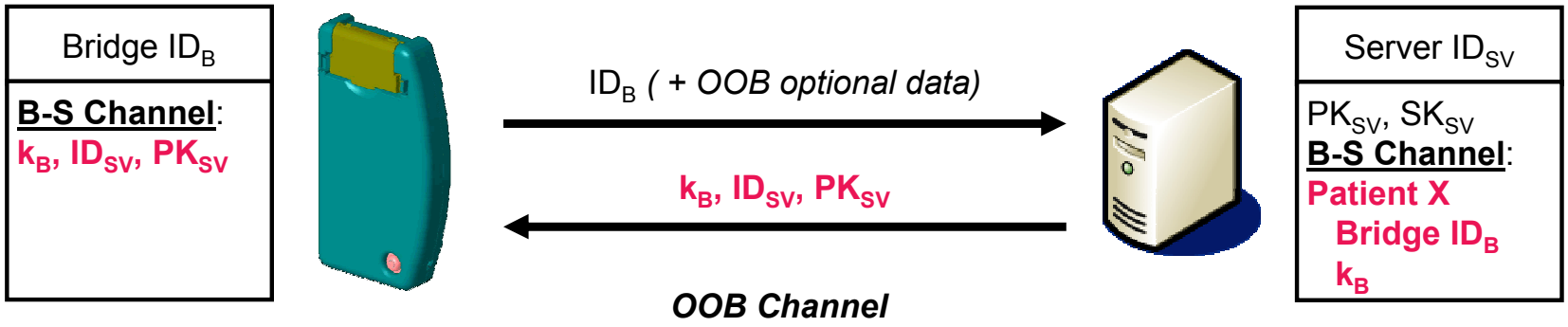


Bridge – Server Channel

Inactive State

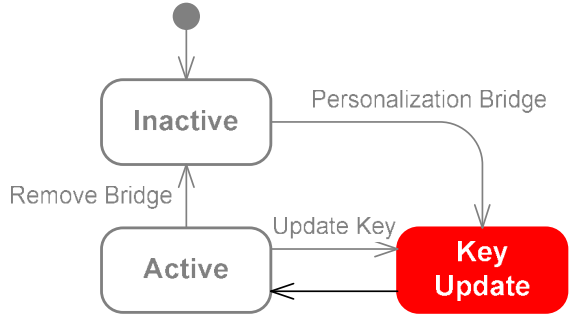


- Assumption
  - Server correctly & securely configured.
  - Bridge empty & inactive.
- Bridge waits until being assigned to a patient.
- When Bridge is assigned to a Patient, it is personalized (using OOB channel):
  - Bridge receives k_B & PK_{SV} to set up secure channel
  - Bridge tied to a given patient X.

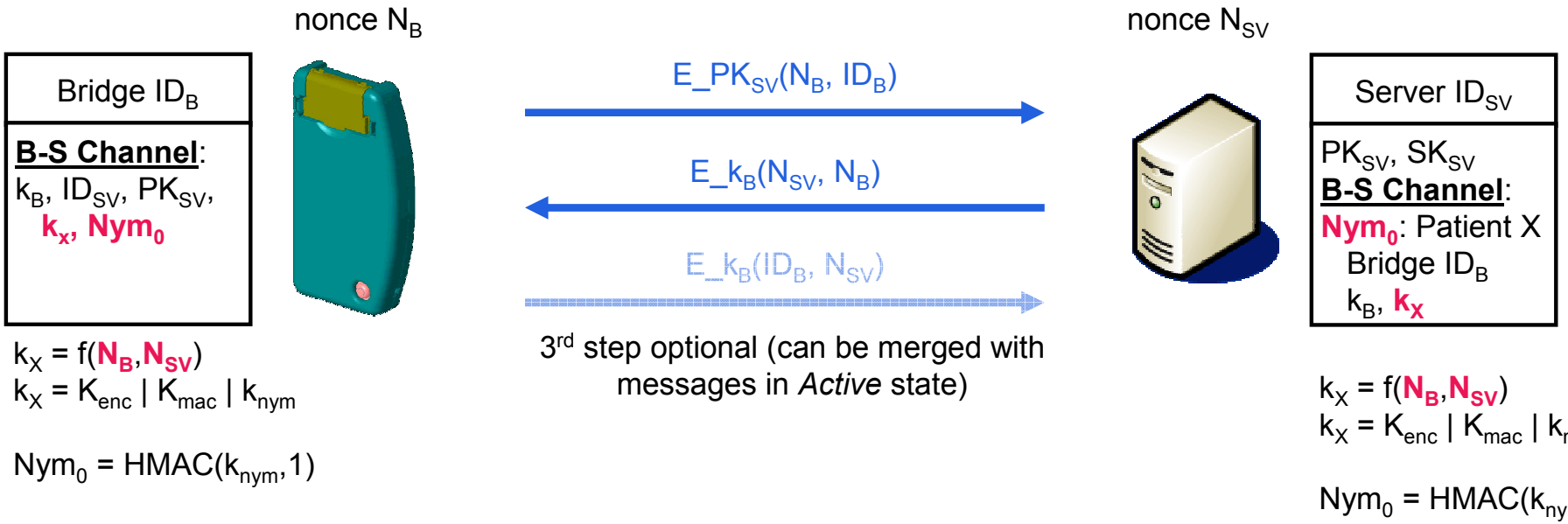


Bridge – Server Channel

Key Update State

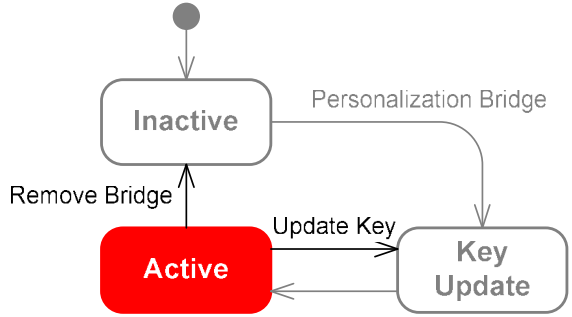


- Bridge first updates the channel keys before going to the Active state
- Purpose:
  - Update the key $k_X = k_{enc} \mid k_{mac} \mid k_{nym}$, allowing secure anonymous exchange
  - Note that this exchange doesn't disclose identifiable information

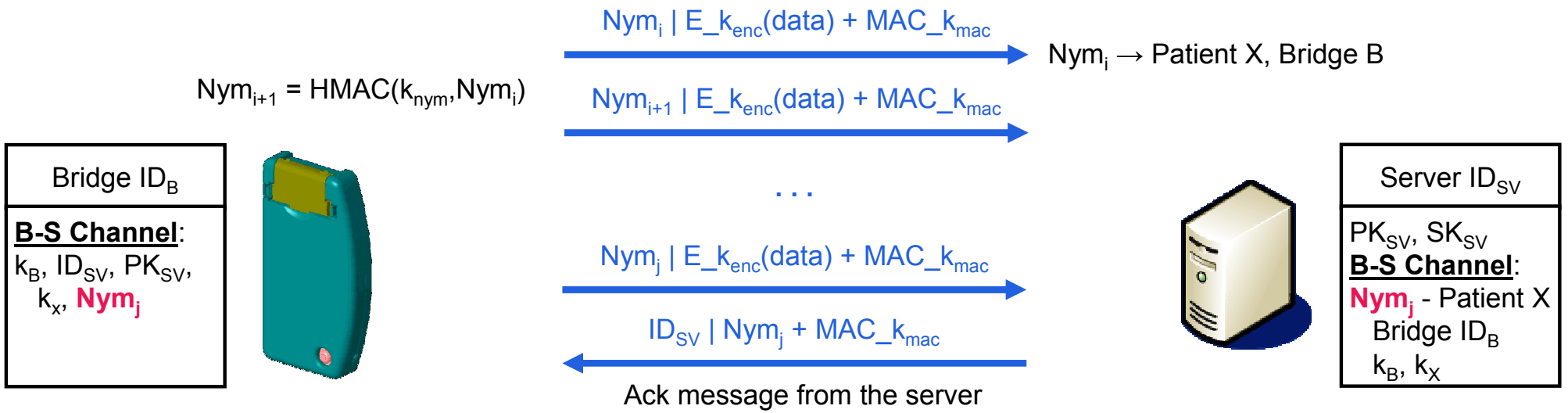


Bridge – Server Channel

Active State



- Data are sent in *asynchronous anonymous* messages
- Message prefixed with a pseudonym Nym_i that is updated at each message.
- Server may acknowledge reception at any time, triggering deletion of data in Bridge memory
- Mechanism to resent missing message (following last acknowledged message)



Part I
Body Area Network

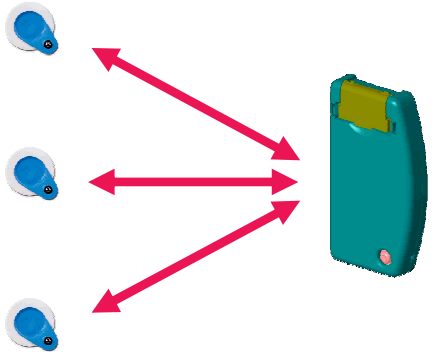
Bridge – Server Channel

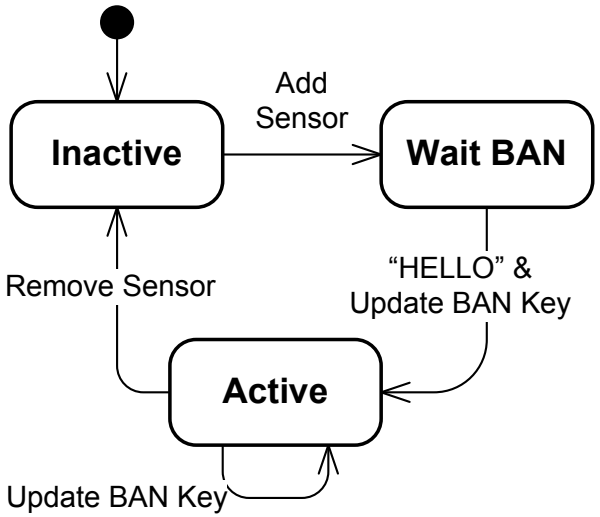
BAN Management Channel

BAN Application Channel

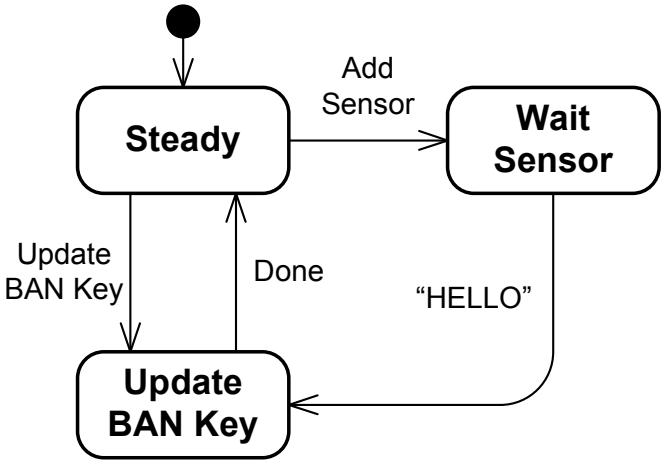
BAN Management Channel

- Main purpose: setup BAN key k_{BAN} that will be used in the **BAN Data Channel**
- **Star** topology around Bridge
- Bridge and sensor have related FSM





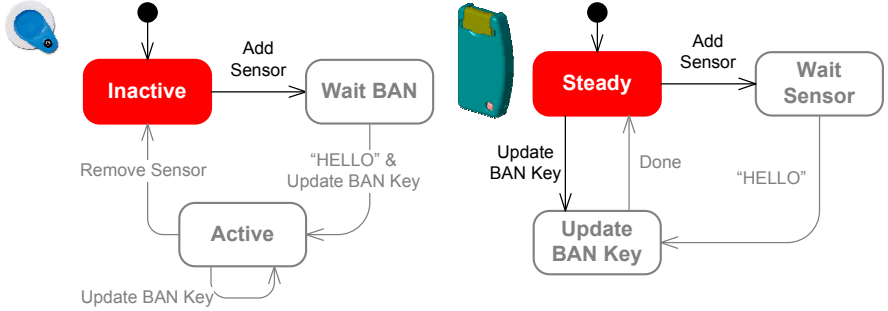
Sensor FSM



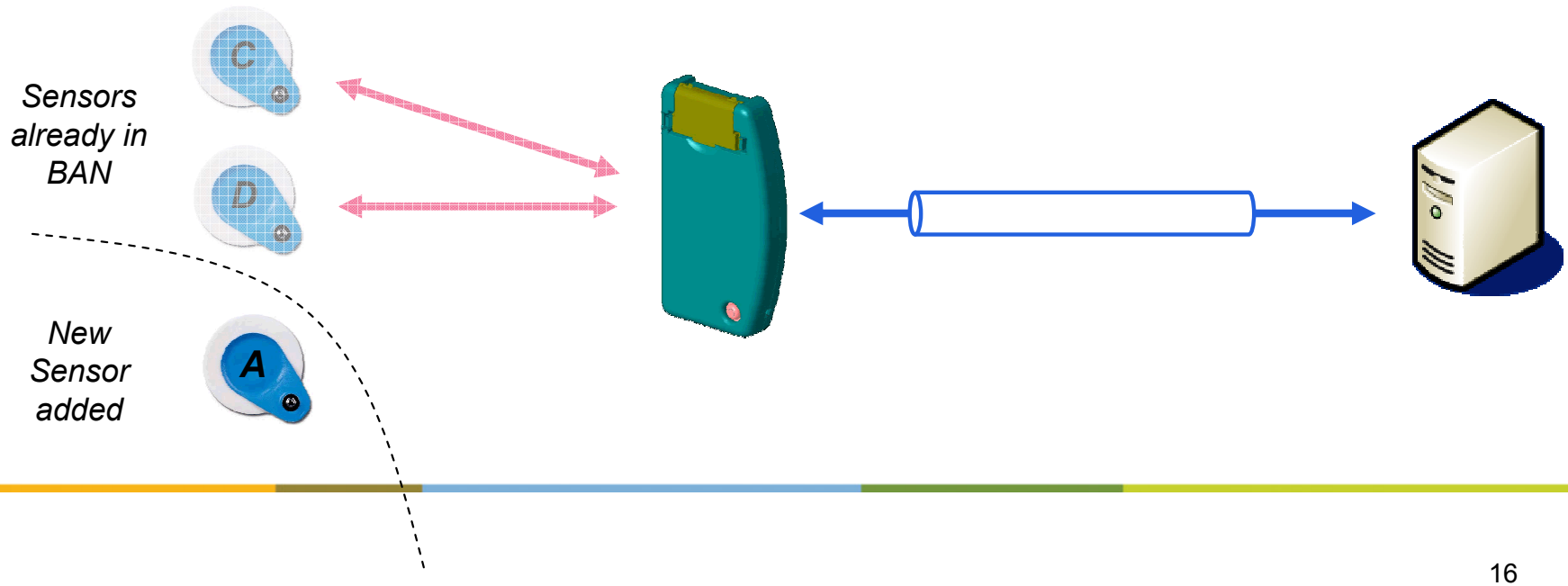
Bridge FSM

BAN Management Channel

Sensor Inactive / Bridge Steady (I)

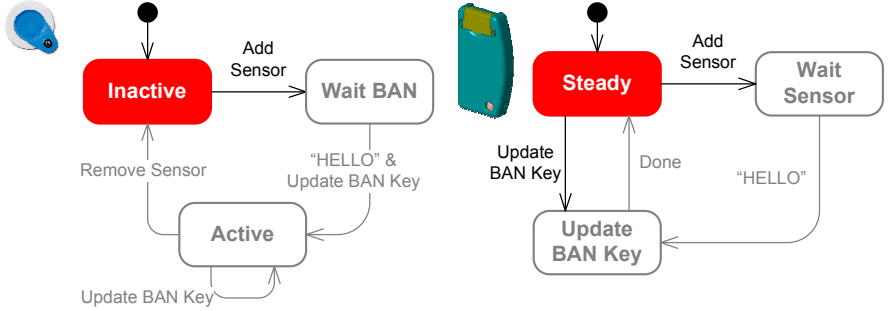


- Assumption
  - A channel between the Bridge and the Sensor exists.
- Bridge is in *Steady* state
  - Possibly already connected to some *Active* sensors.
- New sensor **A** is in *Inactive* state
  - Wait to be assigned to a patient BAN.

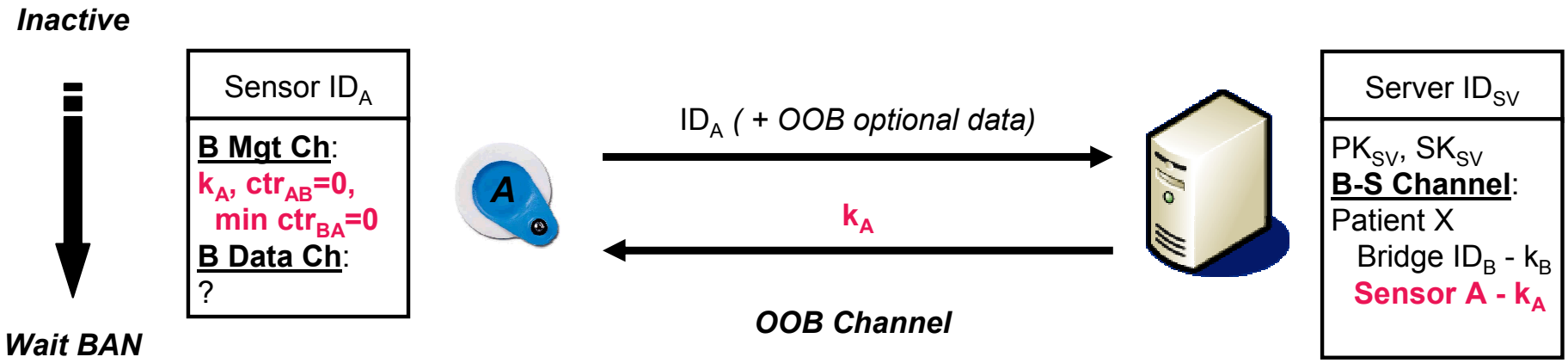


BAN Management Channel

Sensor Inactive / Bridge Steady (II)

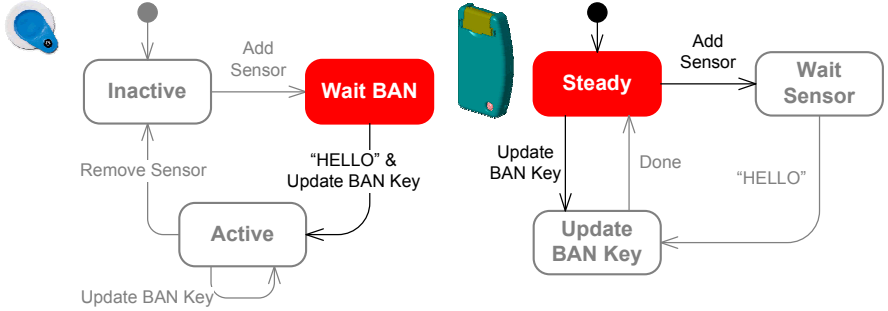


- Adding a sensor is triggered by the Server
- Use **Add Sensor** command
- Purpose
  - Establish a unique key k_A shared between Sensor and Bridge.
- Sensor
  - Personalized when added to a given patient (using OOB channel)
  - Sensor receives k_A, and set non-replay counters ctr_{AB} and min ctr_{BA} to 0.
  - Sensor tied to a given patient X's BAN.

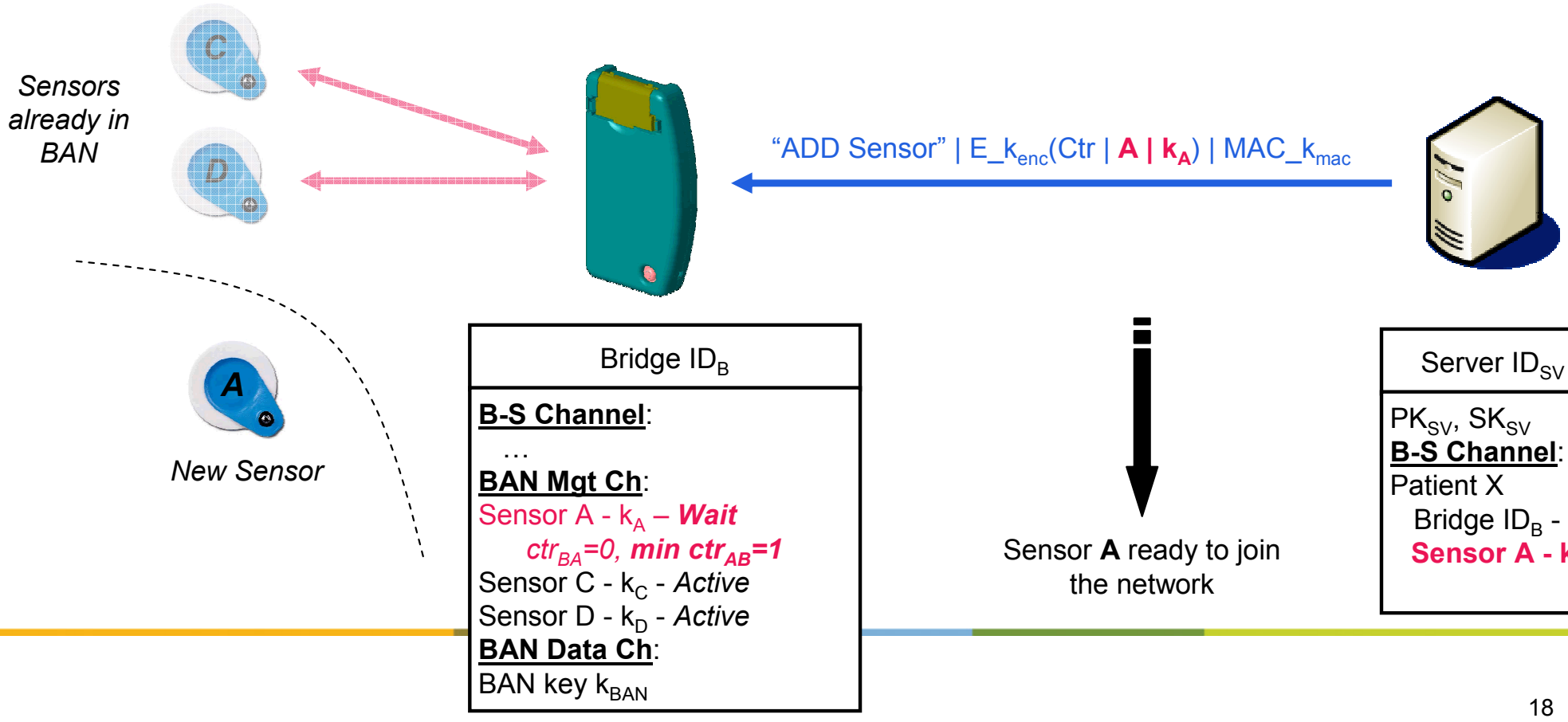


BAN Management Channel

S. Wait BAN / Bridge Steady (III)

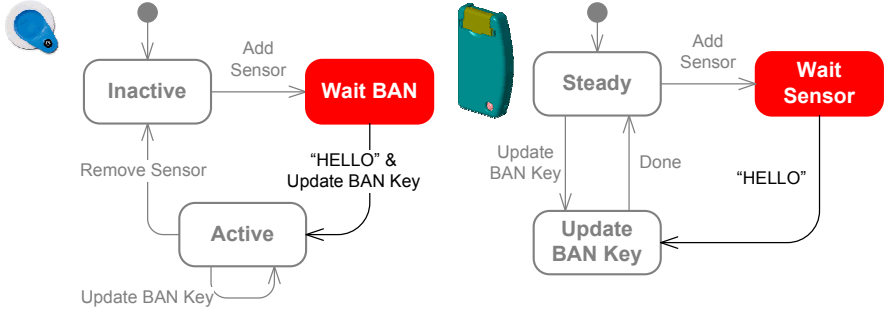


- Bridge
  - Server has personalized the new sensor.
  - Bridge is notified that a new Node will join the BAN
    - Using the available **Bridge – Server Channel**



BAN Management Channel

S. Wait BAN / B. Wait Sensor

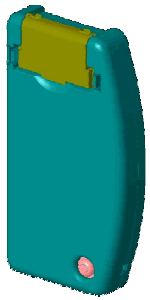


- New Sensor **A** is ready to be added to Patient BAN
- When Sensor **A** detects a BAN in range, it sends a "HELLO" message.
  - Anti-replay counter is incremented.

Sensor ID_A
B Mgt Ch: k_A, **ctr_{AB}=1**, min ctr_{BA}=0 **B Data Ch**: ?

New Sensor

HELLO | A | ctr_{AB} + MAC_$k_{A\text{-}mac}$

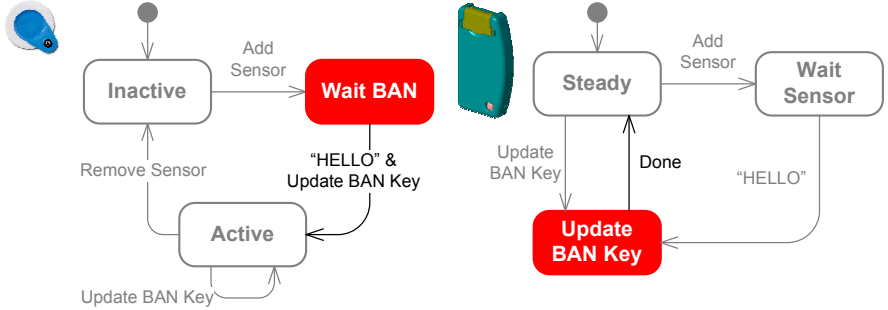


Bridge ID_B
B-S Channel: … **BAN Mgt Ch**: Sensor A - k_A – ***Update*** ***ctr_{BA}=0, min ctr_{AB}=1*** Sensor C - k_C – ***Update*** Sensor D - k_D – ***Update*** **BAN Data Ch**: BAN key k_{BAN}

- Bridge verifies freshness, and goes into *Update BAN Key* state
  - For forward security, a **new BAN key** is generated when a new node joins.

BAN Management Channel

S. Wait BAN / B. Update BAN Key



- Bridge sends an Update BAN Key message to all sensors not updated yet.
- For existing Sensors in *Active* state,
  - Sensor receives and uses the new BAN key **k'$_{BAN}$**.
- For the new Sensor **A** in *Wait BAN* state,
  - it will also assign a (very short) **localID$_A$**, for privacy purpose.
- After update, Sensor **A** becomes *Active*

Sensor ID$_A$
B Mgt Ch: k_A, **ctr$_{AB}$=2,** **min ctr$_{BA}$=1** **B Data Ch**: **localID$_A$**, **k'$_{BAN}$**

New Sensor

localID$_A$ | ctr$_{BA}$ | E_k$_{A\text{-}enc}$(k'$_{BAN}$) + MAC_k$_{A\text{-}mac}$

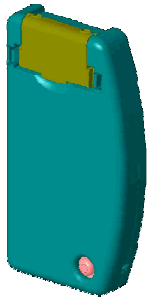
ctr$_{AB}$ + MAC_k$_{A\text{-}mac}$

Sensor ID$_i$
B Mgt Ch: k_i, **ctr$_{i\,B}$,** **min ctr$_{B\,i}$** **B Data Ch**: localID$_i$, **k'$_{BAN}$**

Active Sensors

Idem

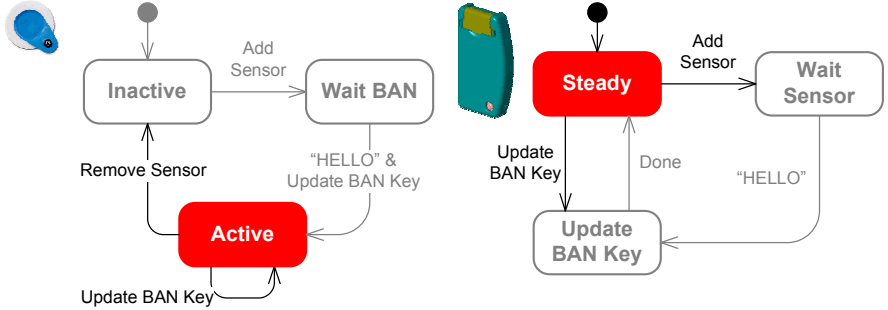
Idem



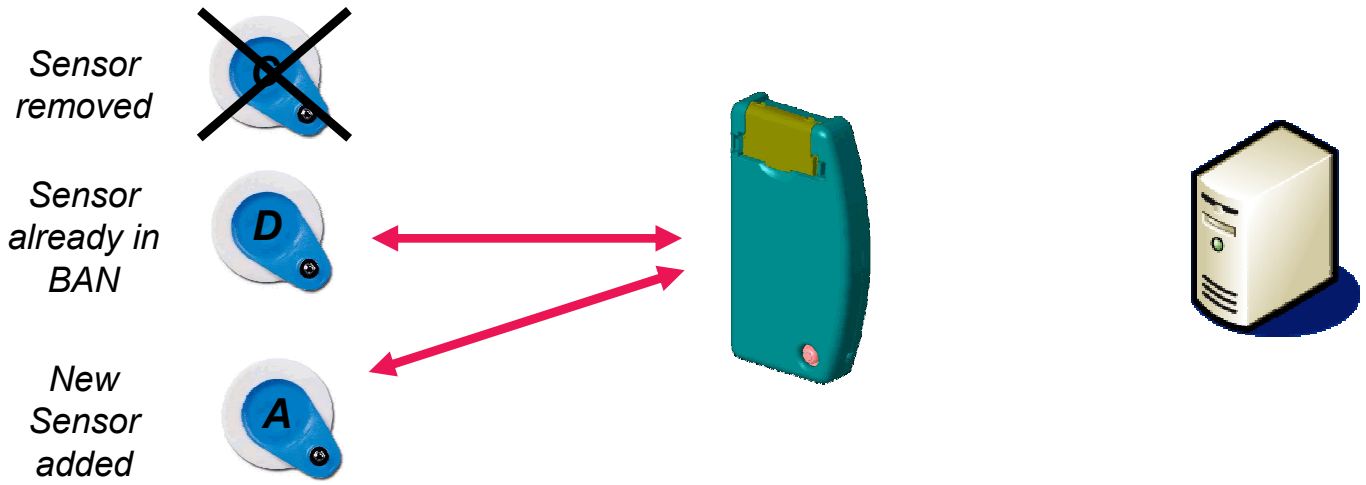
Bridge ID$_B$
B-S Channel: … **BAN Mgt Ch**: Sensor A - k_A – ***Active*** ***ctr$_{BA}$=1, min ctr$_{AB}$=2*** Sensor C - k_C – ***Active*** Sensor D - k_D – ***Active*** **BAN Data Ch**: New BAN key: **k'$_{BAN}$** Old BAN key: **k$_{BAN}$**

BAN Management Channel

Sensor Active / Bridge Steady



- New Sensor **A** is now *Active* in the BAN
- Sensor may use the **BAN Data Channel**
- Bridge may on regular basis **update** the BAN key
  - Depending on the strength of algorithms used in the **BAN Data Channel**.
- Sensor may also be **removed** from the BAN



Part I
Body Area Network

Bridge – Server Channel

BAN Management Channel

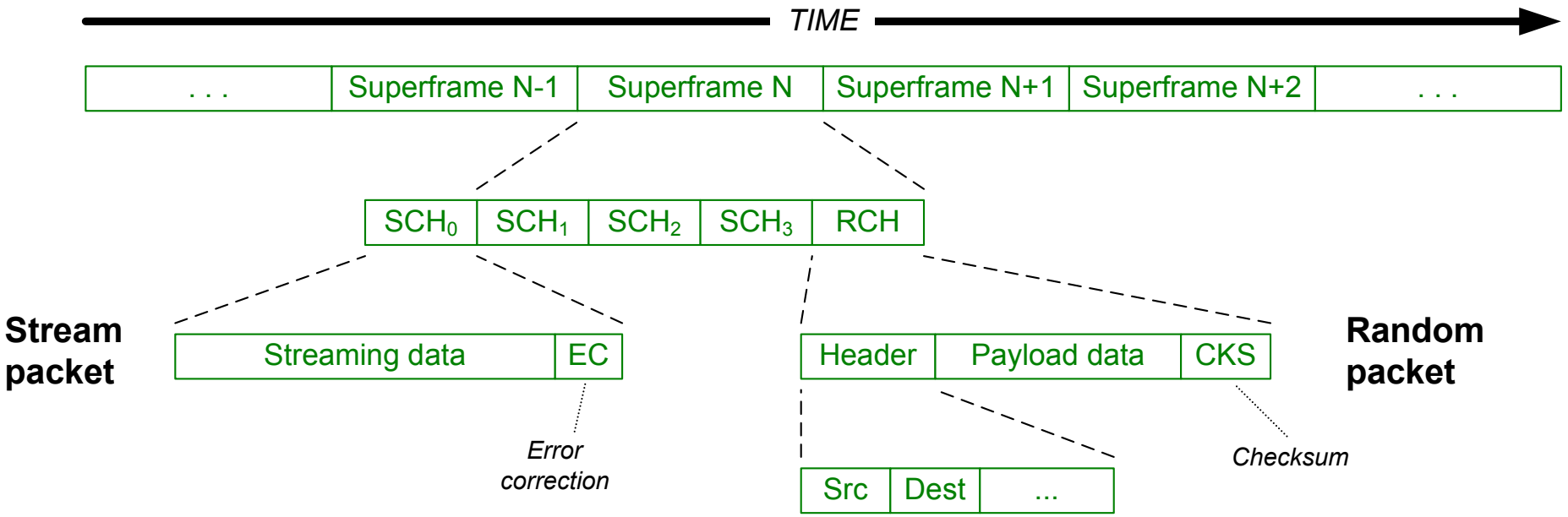
BAN Application Channel

Description

Analysis

BAN Application Channel

Radio layer



Securing the Radio MAC Layer

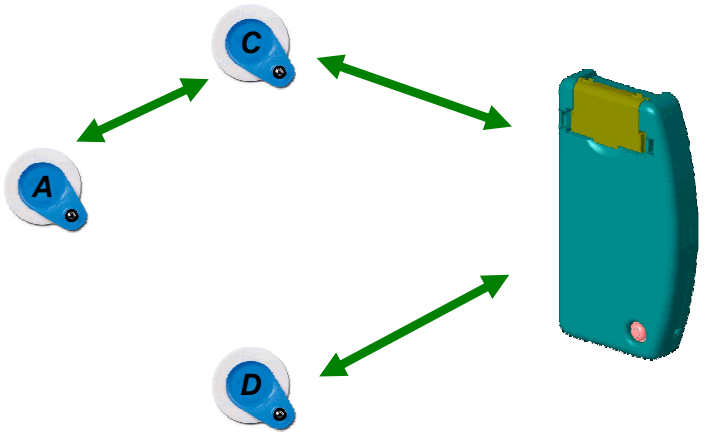
- We need **encryption** and **data authentication**
  - Design will be implemented in HW to decrease power consumption
- For encryption,
  - **Stream ciphers** are best candidates given constraints
    - Allow low power, low gate-count implementation
    - Allow to have very small impact on radio data path (simple xor).
  - All nodes share same stream cipher key $\mathbf{k_{BAN}}$
    - **(+)** Allow any BAN topology, including multi-hop
    - **(-)** All nodes must synchronize their stream cipher state

Encryption

Synchronization of the Stream Cipher state

- Easy Solution:
  - Sensors **consume the same # of random bits from the PRNG** for any encrypted channels **even if they are not target of the message**.
  - **(+)** Very easy
  - **(-)** power consumption overhead (small)
- In case of sync loss, sensors may request BAN key update from the Bridge

Packet	SCH_0	SCH_1	SCH_2	SCH_3	RCH
Length	128b	128b	128b	128b	max 64b
Sensor A		TX			RX/TX
Sensor C	RX	RX	TX		RX/TX
Sensor D	RX			TX	RX/TX
Bridge	TX		RX	RX	RX/TX

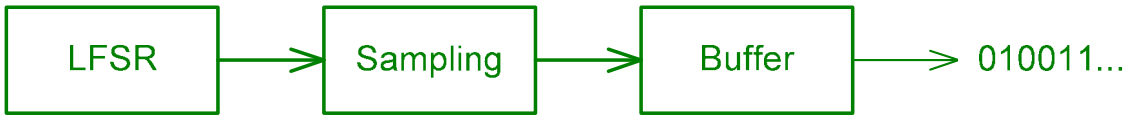


Example of channels allocation

Encryption

Algorithm selection

- The ***Self-Shrinking Generator*** (**SSGen**) is a good choice for now
  - Very cheap implementation



- **128-bit SSGen**
  - ~ 80-bit key security [zenner01]
  - Require key update after ~10^9 generated random bits [Mihaljević96]
- Weak point is that bit rate is not constant
  - Random bits must be buffered for synchronization with radio bits
  - No efficient fast forward
    - Would be helpful when sensor must discard random bit to stay synchronized with other sensors
- Keep an eye on the eStream candidates!

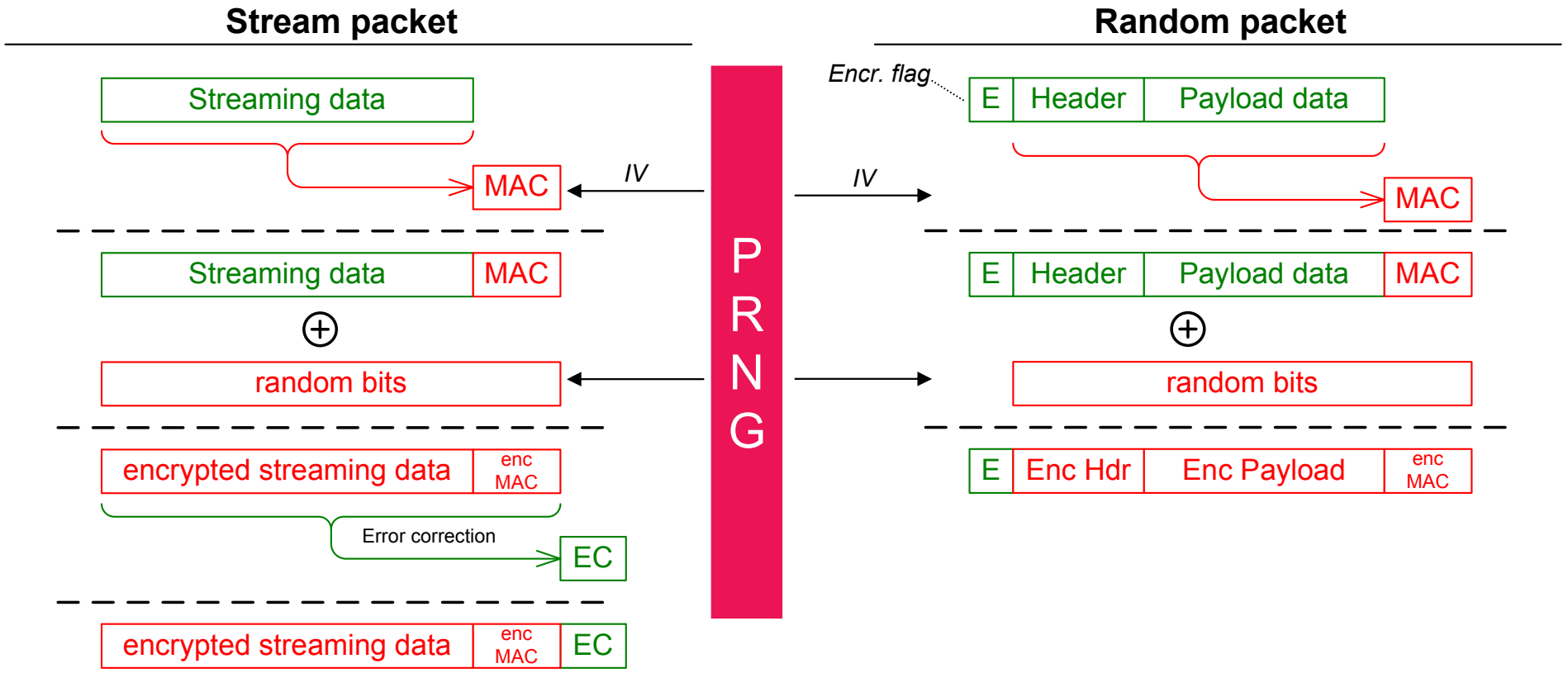
Message Authentication

- Principle
  - Leverage on what we have already
    - i.e. our 80-bit key equiv. stream cipher
  - Communication is real-time (only 1 try)
  - We can’t ask too much

Message Authentication

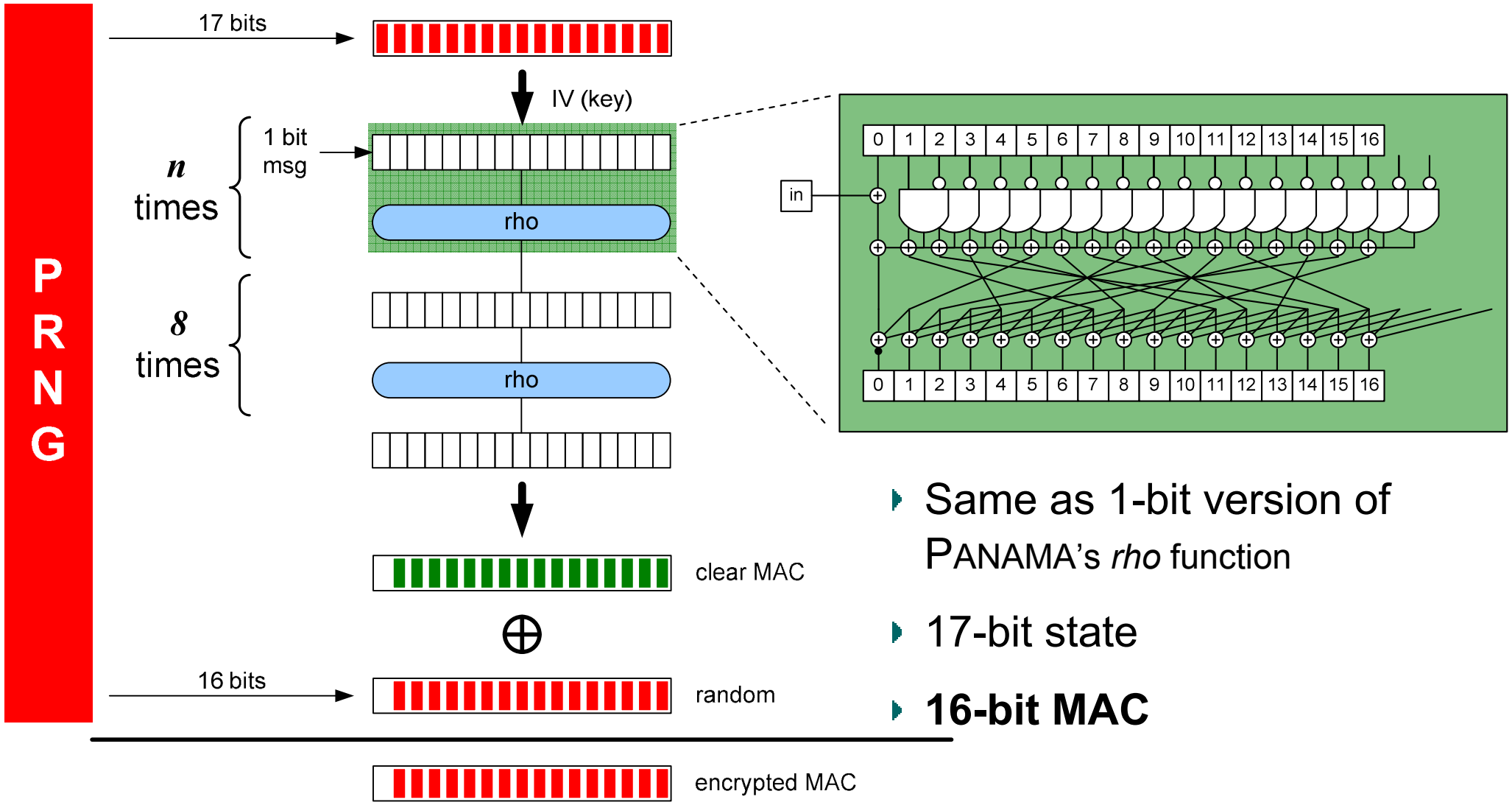
The MACing scheme

- MAC is seeded with the PRNG
- MAC is encrypted
  - Only encrypted messages can be authenticated



Message Authentication

Selection of the Algorithm



- Same as 1-bit version of PANAMA's *rho* function
- 17-bit state
- **16-bit MAC**

Stronger MACs

- Can we extend easily the scheme to get stronger MAC?
- **Extended MAC**
  - Idea: *Use a larger MAC block*
    - Use **2-bit/word version** of the PANAMA rho's function → 32-bit MAC
    - Along with **2-bit input / round**
    - A few more blank rounds
  - **(+)** Same power efficiency (except for blank rounds)
  - **(–)** 2x gate-count.
- **Repeated MAC**
  - Idea: *Send the same message twice, and both MAC must be correct.*
  - **(+) Exactly same HW**
  - **(+)** Very easy
  - **(+)** Same gate count
  - **(–)** Less power effective
    - **(+)** Leave the choice to application layer (eg. for sensitive messages)

Part I
Body Area Network

Bridge – Server Channel

BAN Management Channel

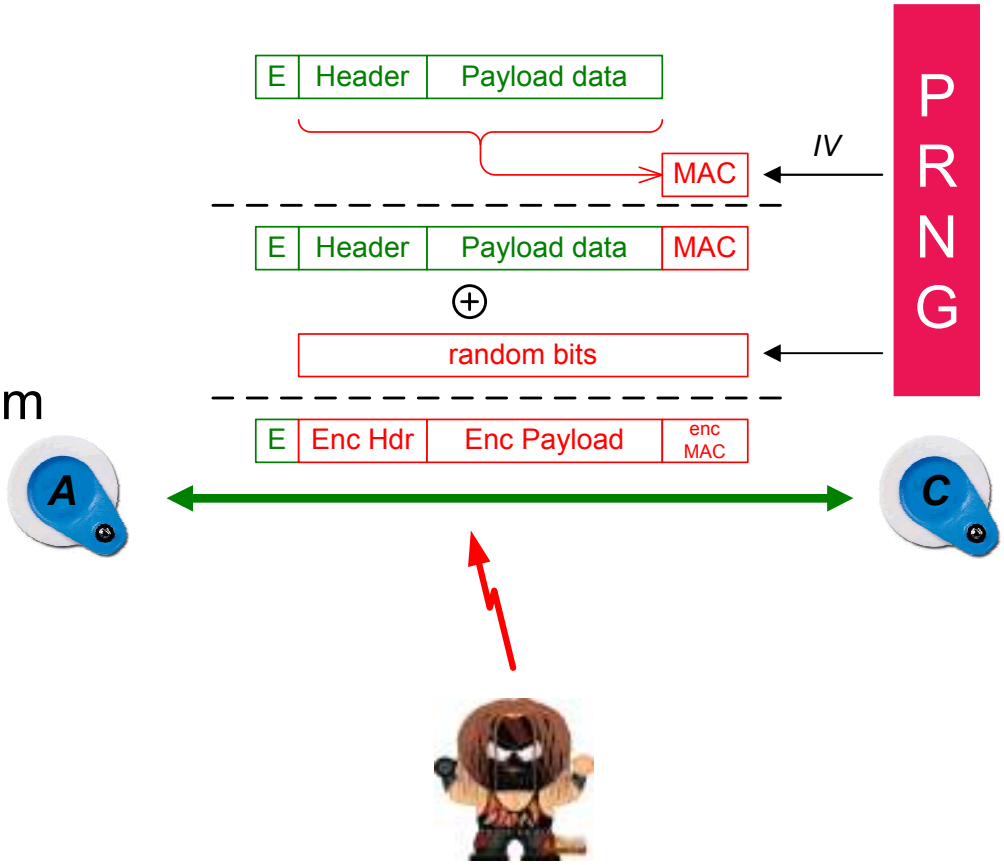
BAN Application Channel

Description

Analysis

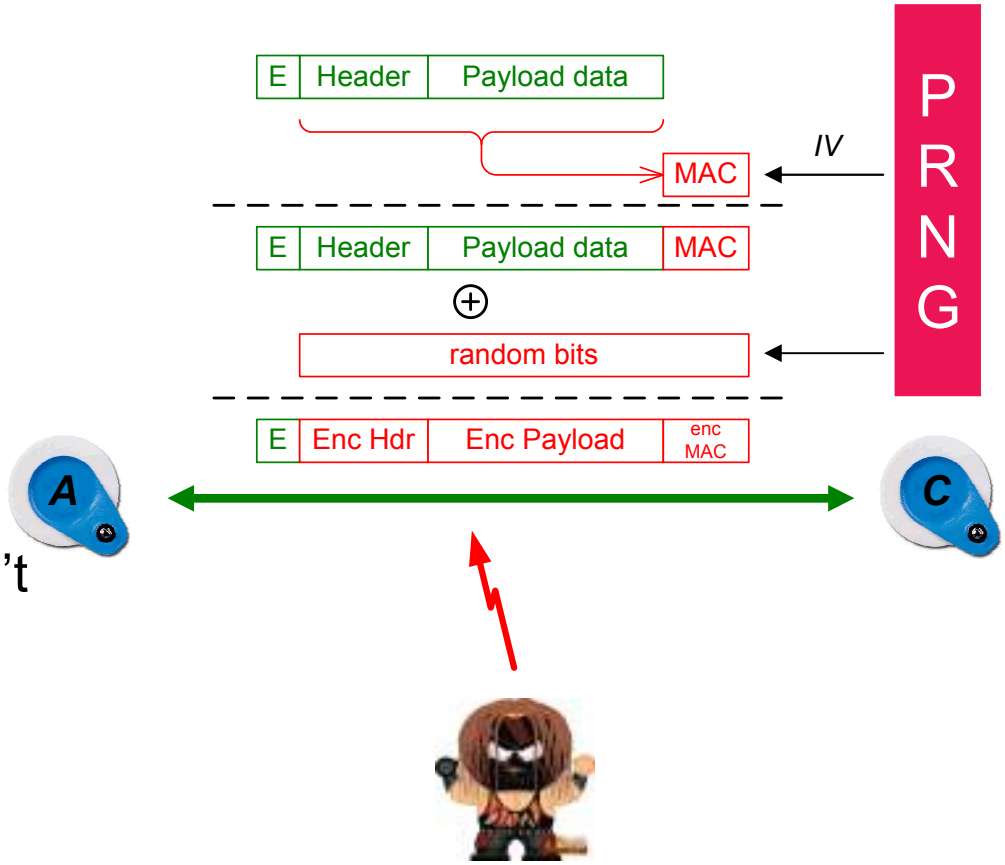
Security Analysis

- Assumption
  - No corruption/interference with sensors
  - Random bits sync with. radio stream
  - → No replay
- MAC block is invertible
  - Could reverse easily blank round
  - Can't be done because MAC encrypted
- Messages and MAC encrypted
  - Attacker can't inject new messages (except by chance)



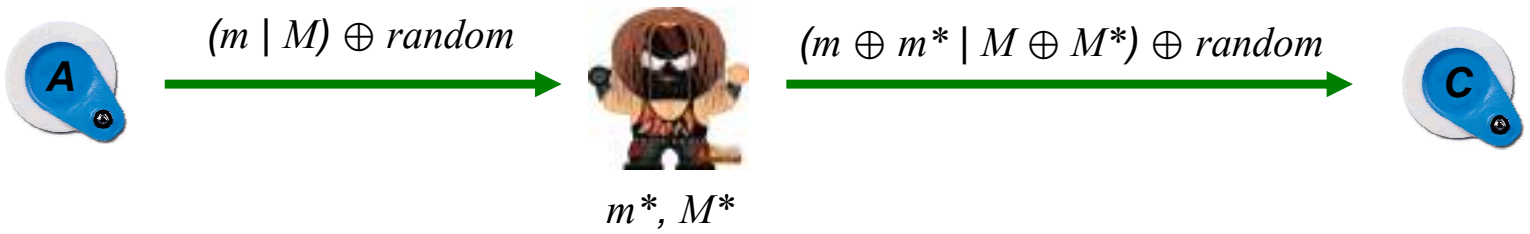
Security Analysis

- MAC key recovery attack?
  - Irrelevant
  - MAC IV (=key) only used once,
  - MAC encrypted anyway
- Fixed points?
  - Not possible because attacker can't control input to the MAC function
  - Messages length is limited
  - MAC is encrypted



The Attack (I)

- Attacker **intercepts** encrypted message and MAC
- Inject **differences** m^* and M^* in message and MAC to get a new message of **same length** that is accepted by the BAN sensors with **high probability**.



- Attacker doesn't know the clear value of the message m or the M
  - m^* and M^* are independent from m and M.

The Attack (II)

- Which m^* and M^* to use?
  - For all possible input message length,
  - For all possible input message differences m_i^*,
  - For all possible MAC IV,
  - Compute all possible MAC differences M_i^{j*} and their probabilities p_i^j
- Most efficient attack is to use (m_i^*, M_i^{j*}) with the highest probability p_i^j

Differential m_0^*	*MAC difference* M_0^{0*}	*with probability* p_0^0
	MAC difference M_0^{1*}	*with probability* p_0^1
	…	…
Differential m_1^*	*MAC difference* M_1^{0*}	*with probability* p_1^0
	MAC difference M_1^{1*}	*with probability* p_1^1
	…	…
Differential …	…	…

⇒ Say that $\boldsymbol{p_0^1} = max\ p_i^j$

Best attack:

inject difference (m_0^*,M_0^{1*}) with probability of success p_0^1

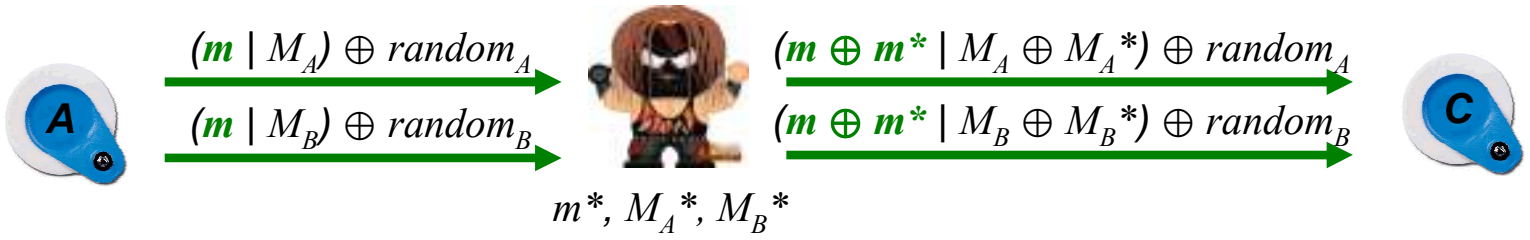
(for any intercepted message)

The Attack (III)

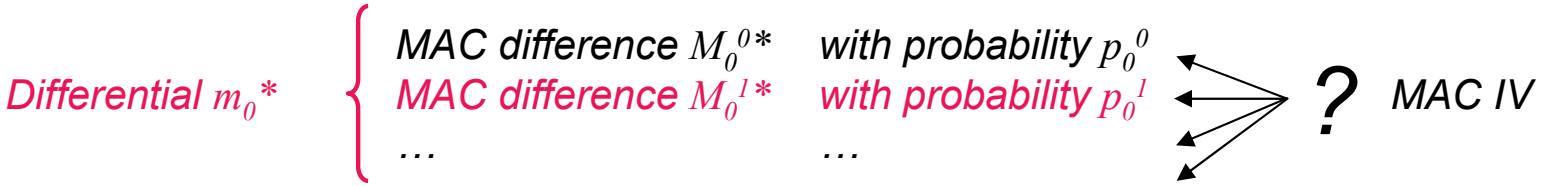
- Since MAC state is 17-bit, we have for a given differential m_i*,
  - $\sum_j p_i^j = 1$
  - Number of possible MAC difference M_i^j* < 2^{17}
- So clearly we have $max\ p_i^j > 2^{-17}$
  - i.e. a differential with “high” probability exists
- This eats up some bits in the resistance claim we can achieve
  - $2^{16} \rightarrow 2^{17-x}$
  - if random permutation, x ~ 5 → 2^{12}
- Let’s check the Repeated MAC…

Attacking the Repeated MAC

- Repeated MAC: Same message m sent twice



- Attacker can't exploit the fact that same message MACed twice
  - Because MAC IV in the 2nd message is indep. from the one in 1st message

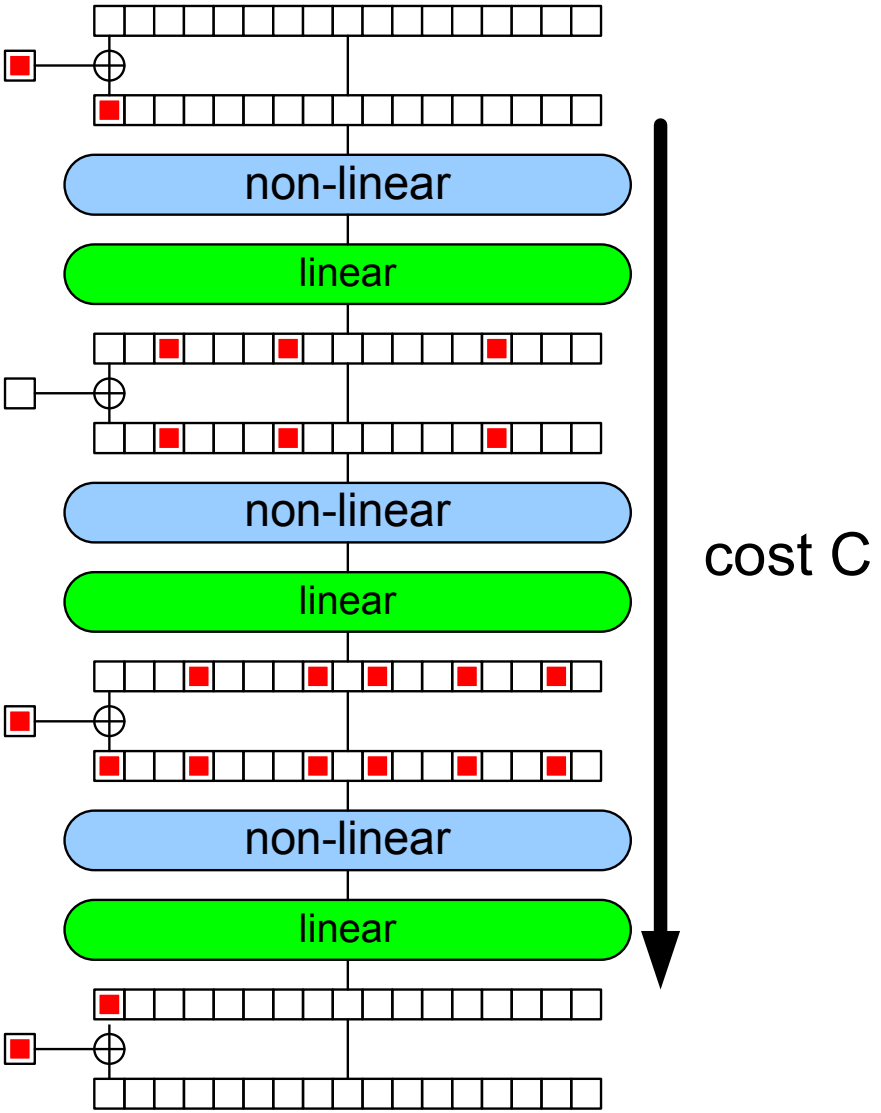


- Best attack
  - We reuse the same differential (m_i^*, M_i^{j*}) with highest probably, and inject the same difference in both messages (ie. $M_A^* = M_B^*$)
  - **Probability of success is $(p_i^j)^2$**

Part II
Looking Quickly for Trails…

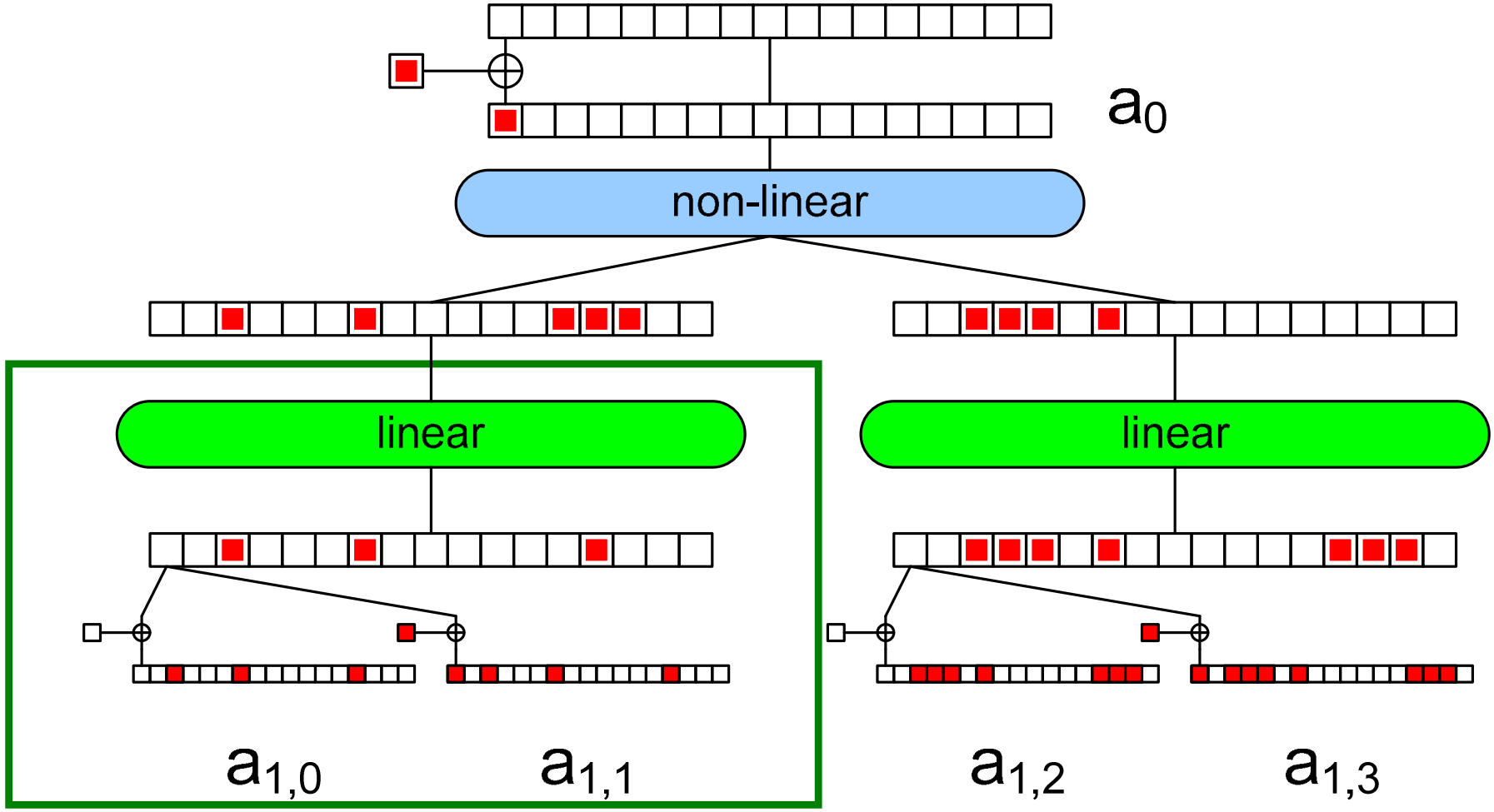
Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche

Looking for Trails…



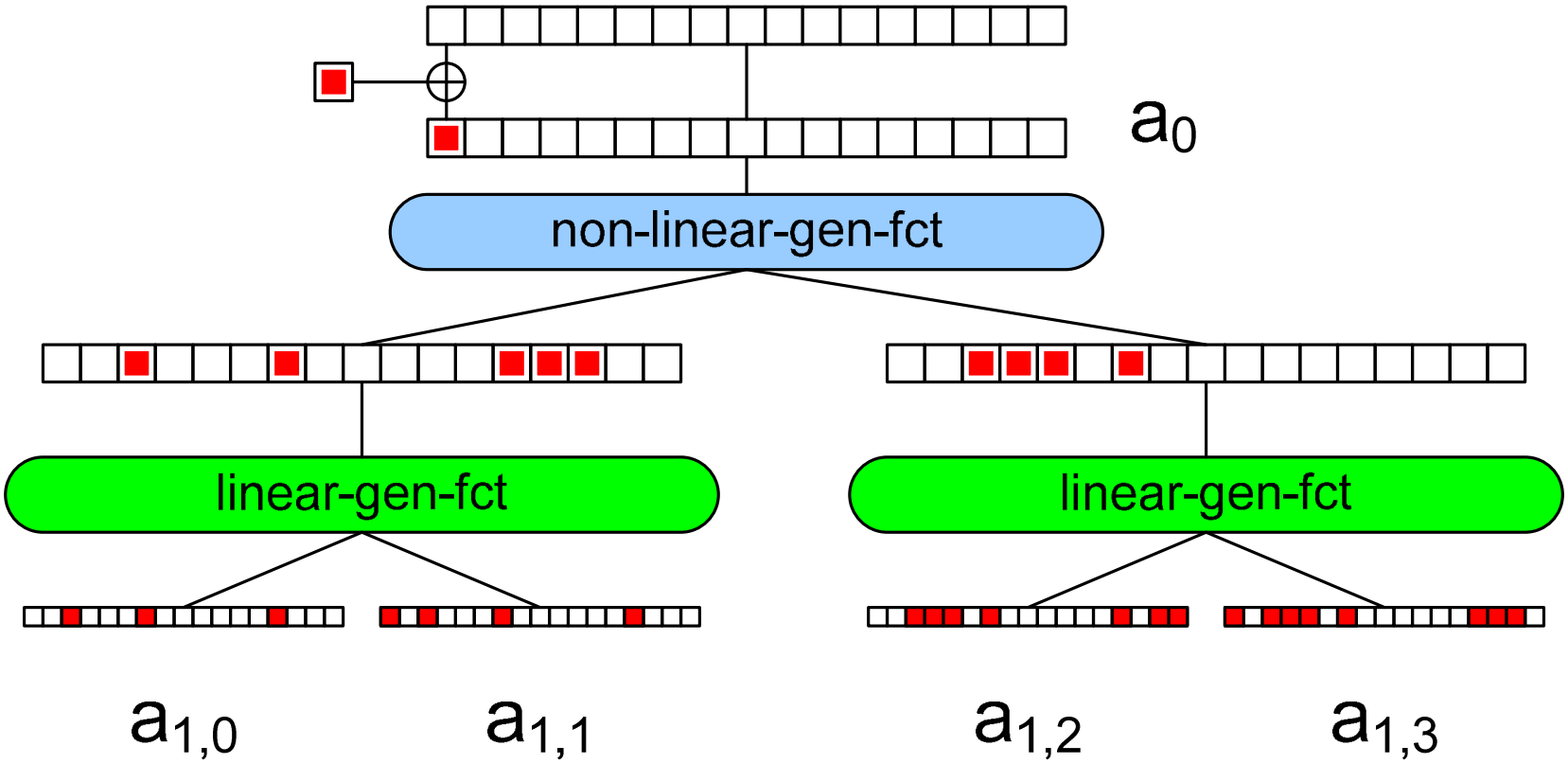
- Say we want to find a ***collision trail*** in the Panama *rho* function
- Here 3-round (dummy) example trail.
- Given this trail, attacker can mount an attack with cost C
- We want to find the trail with lower cost C

Going Through 1 round



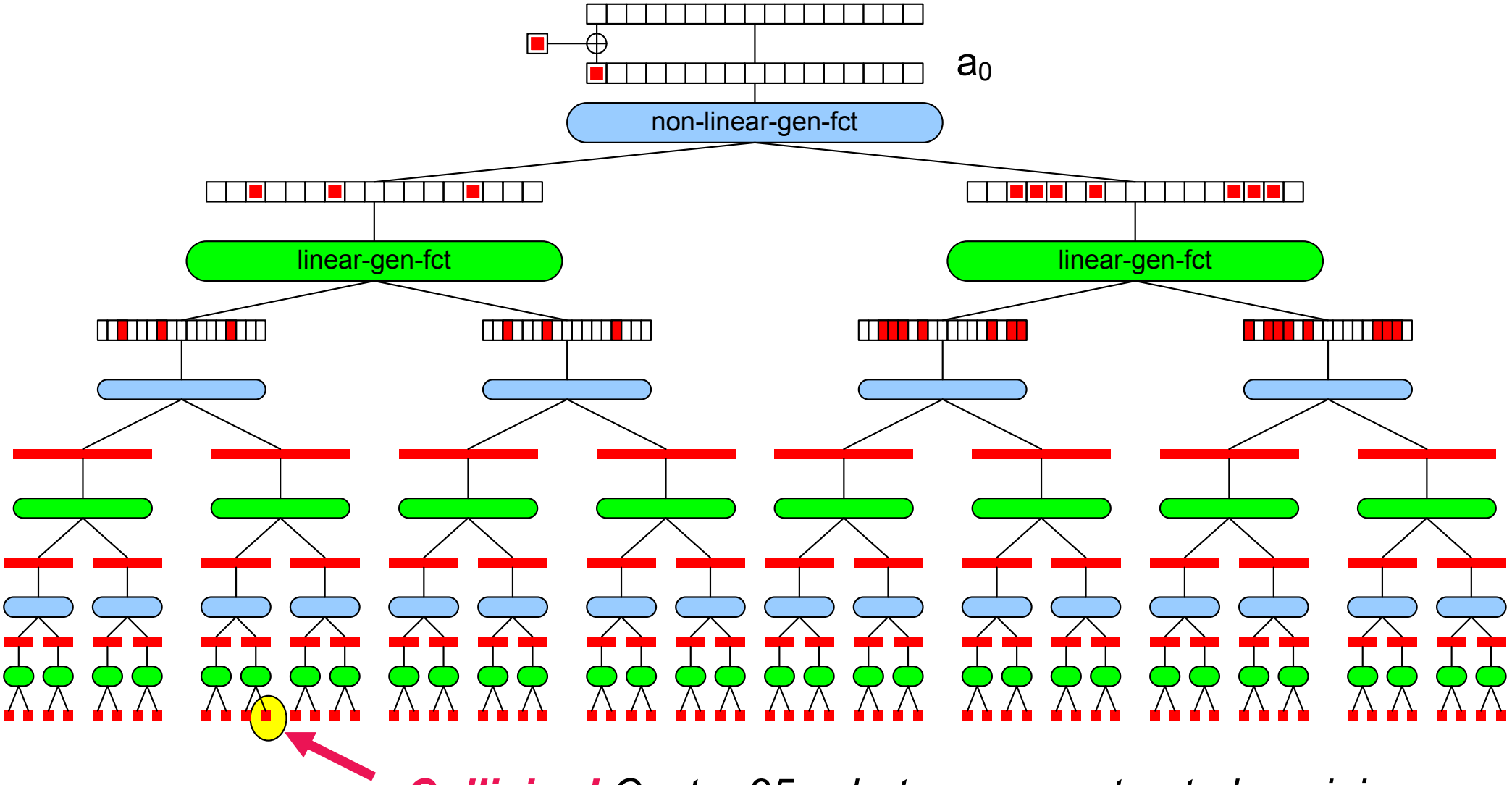
We can merge this step

Tree node Generation Functions



Tree Search

Depth-first search

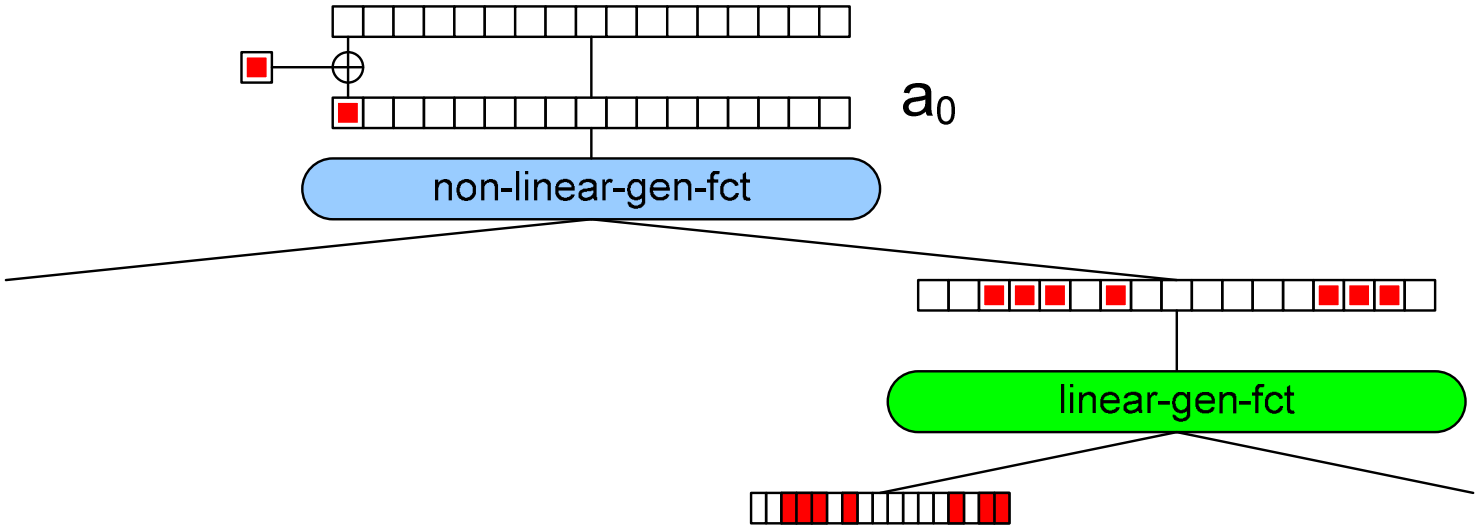


Collision! *Cost = 25… but no guarantee to be minimum*

Tree Search

Cost-first search

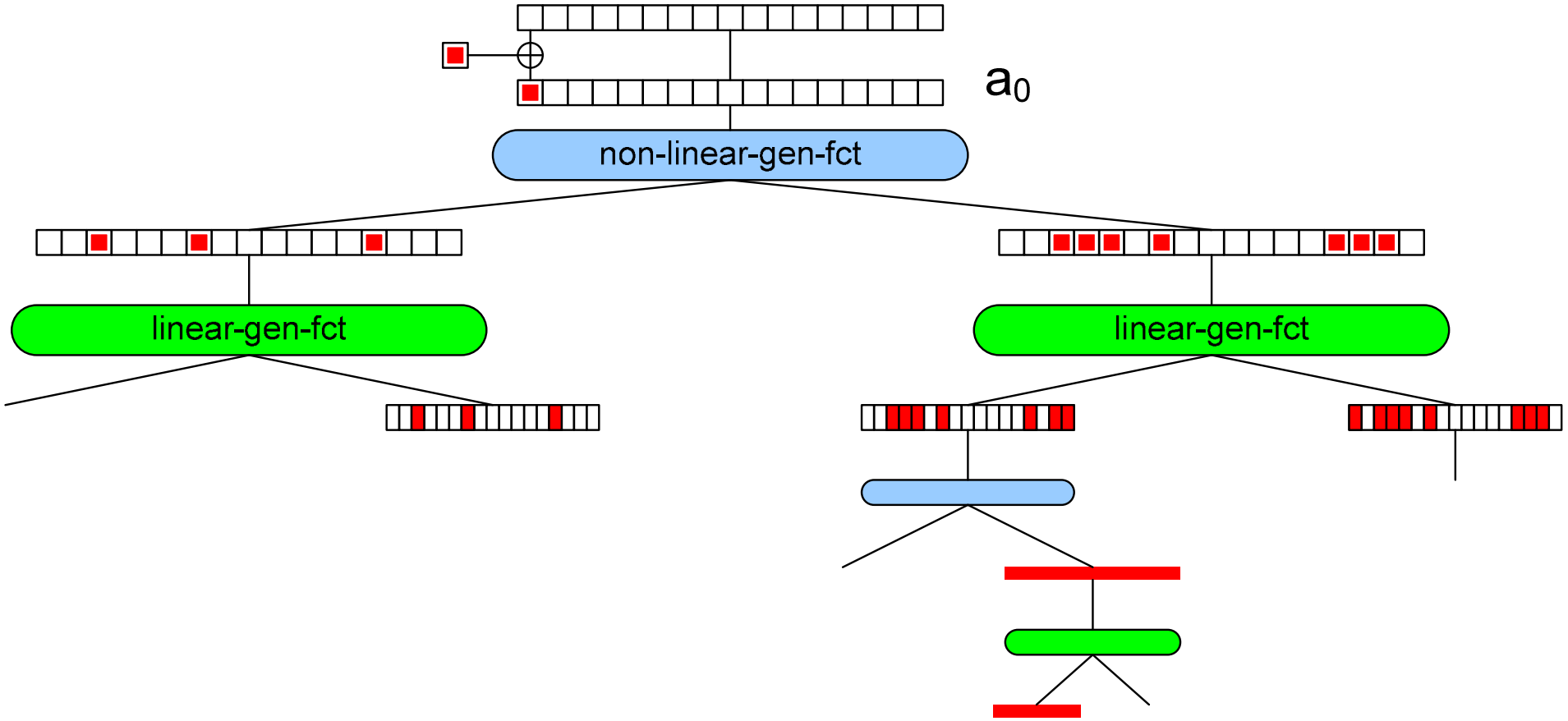
*Look for **cost ≤ 5***



Tree Search

Cost-first search

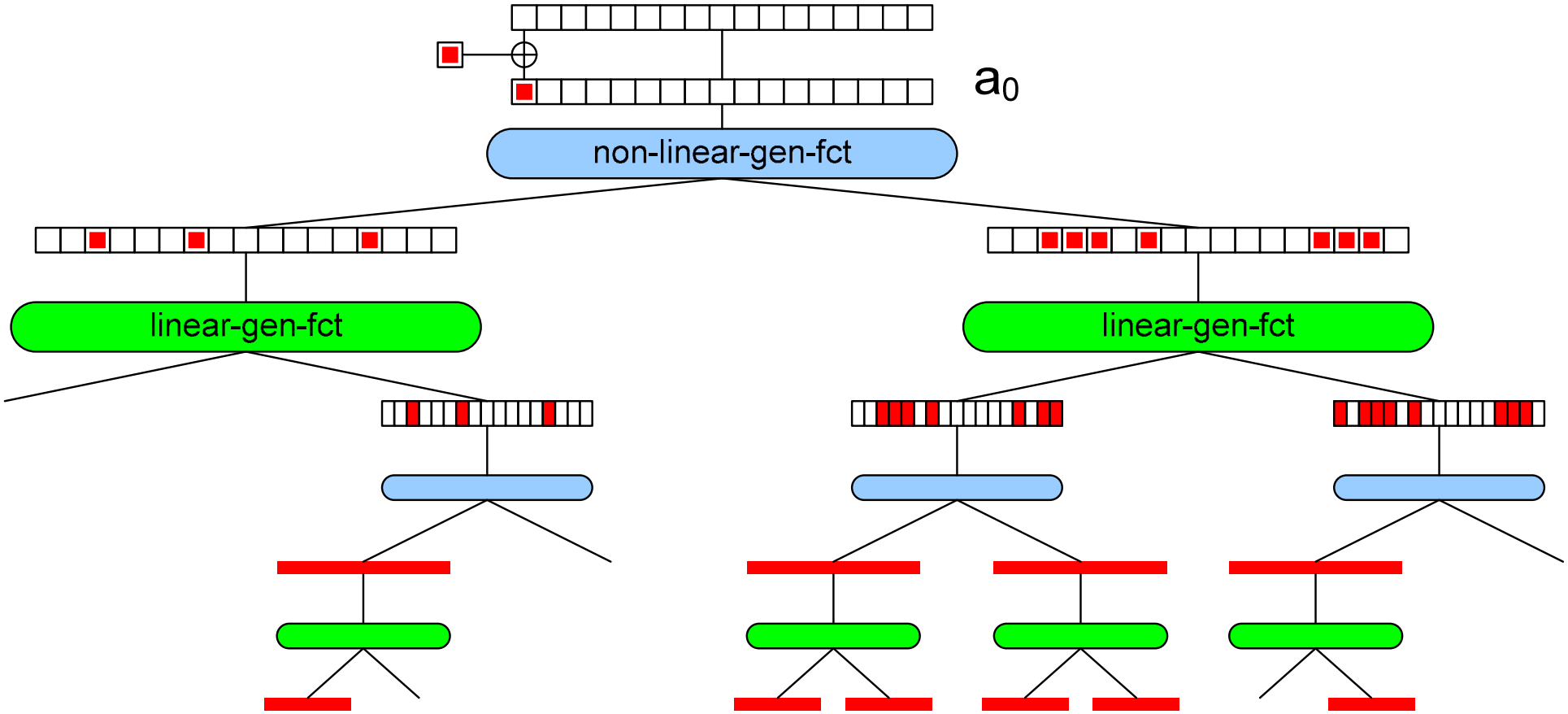
*Look for **cost ≤ 10***



Tree Search

Cost-first search

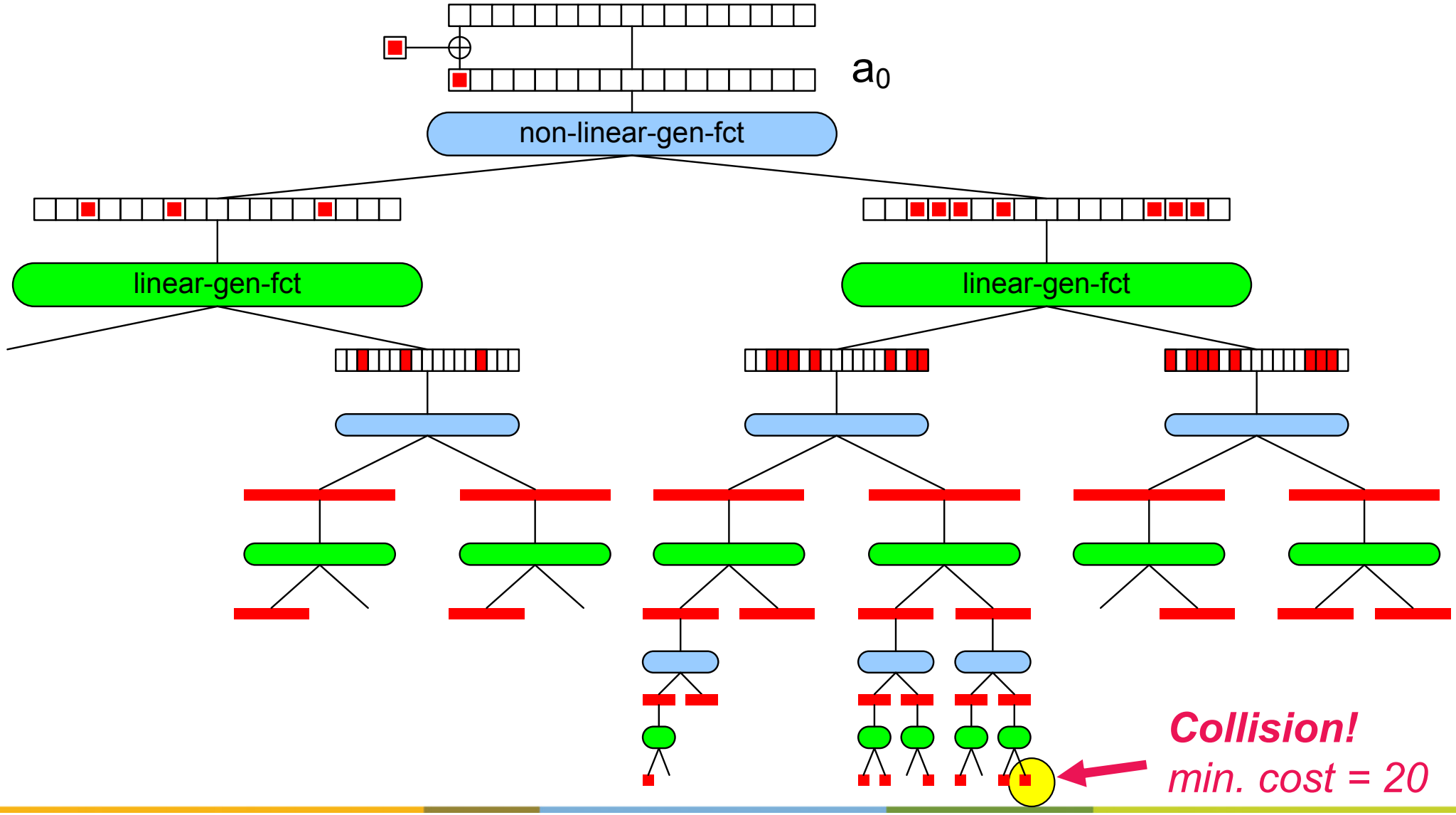
*Look for **cost ≤ 15***



Tree Search

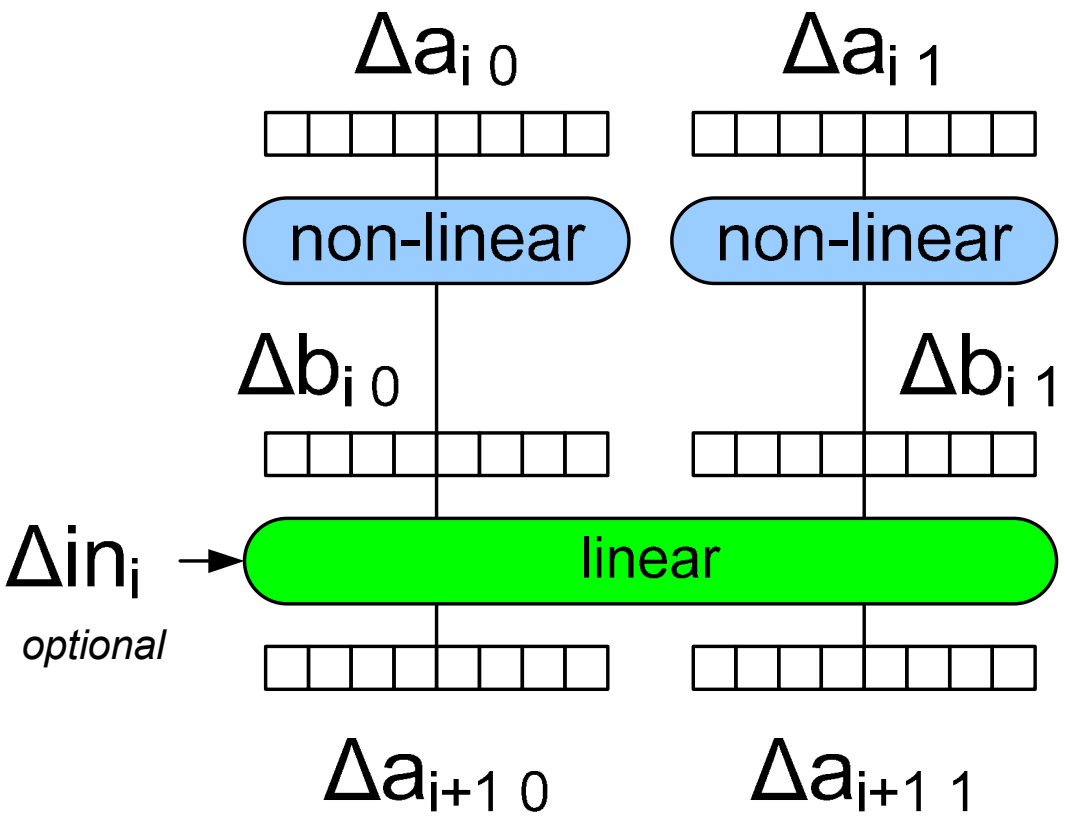
Cost-first search

*Look for **cost ≤ 20***



Time-Efficient Cost-First Search

Optimizing 1-round



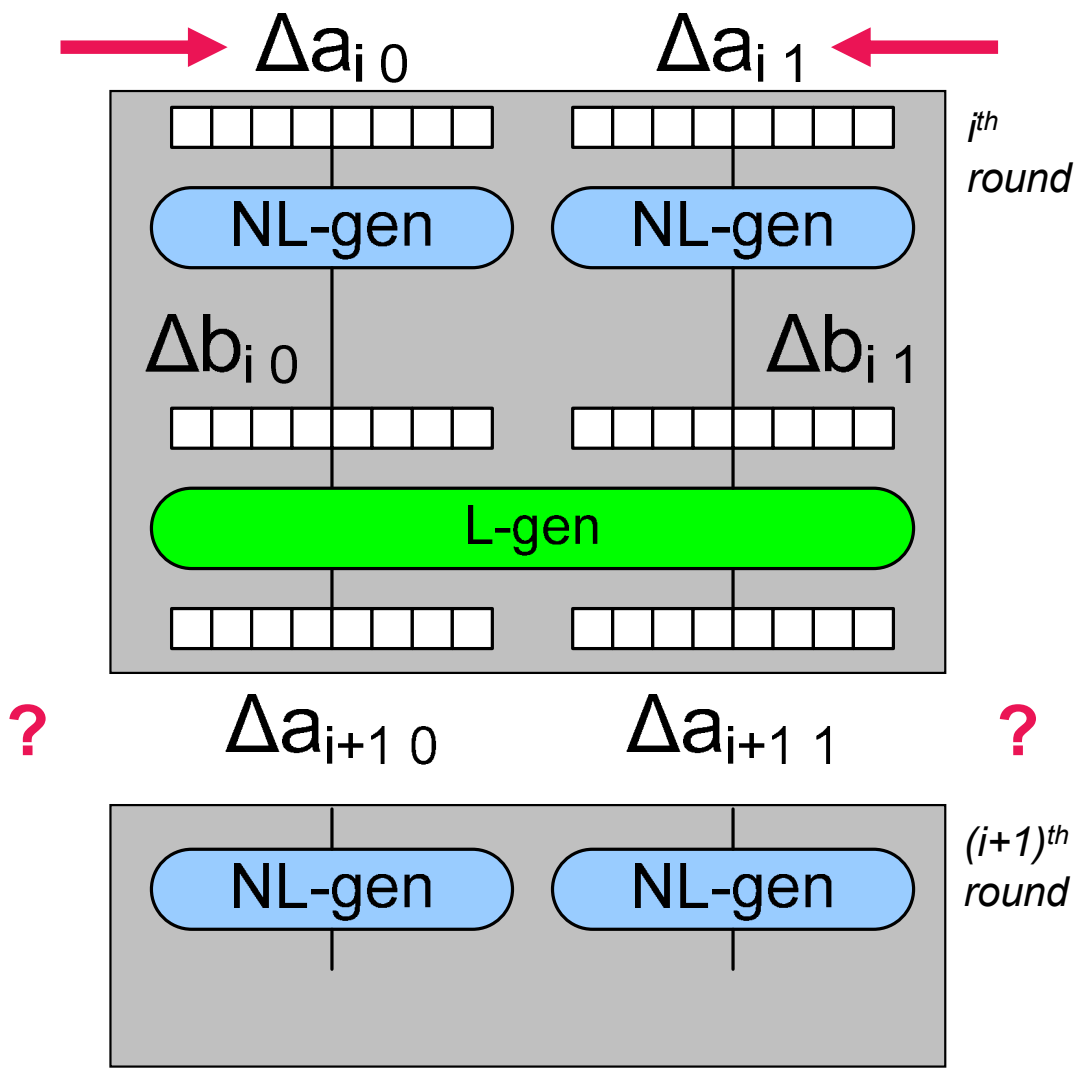
- Round function with several S-boxes.
- Assumption

$$\min [\ \textit{cost increase}\ ] \Leftrightarrow \min \left[f(\Delta a_{i\,0}) + f(\Delta a_{i\,1}) \right]$$

$$f: \{0,1\}^n \rightarrow \mathbb{N}$$

- We need to optimize when no collision found
  - ie. $\Delta a_{i+1\,0} \neq 0$ or $\Delta a_{i+1\,1} \neq 0$

Going Through 1 Round in the Tree



We visit this node

Can generate nodes with cost C ($\leq C_{max}$)

Generate only if must be visited later on

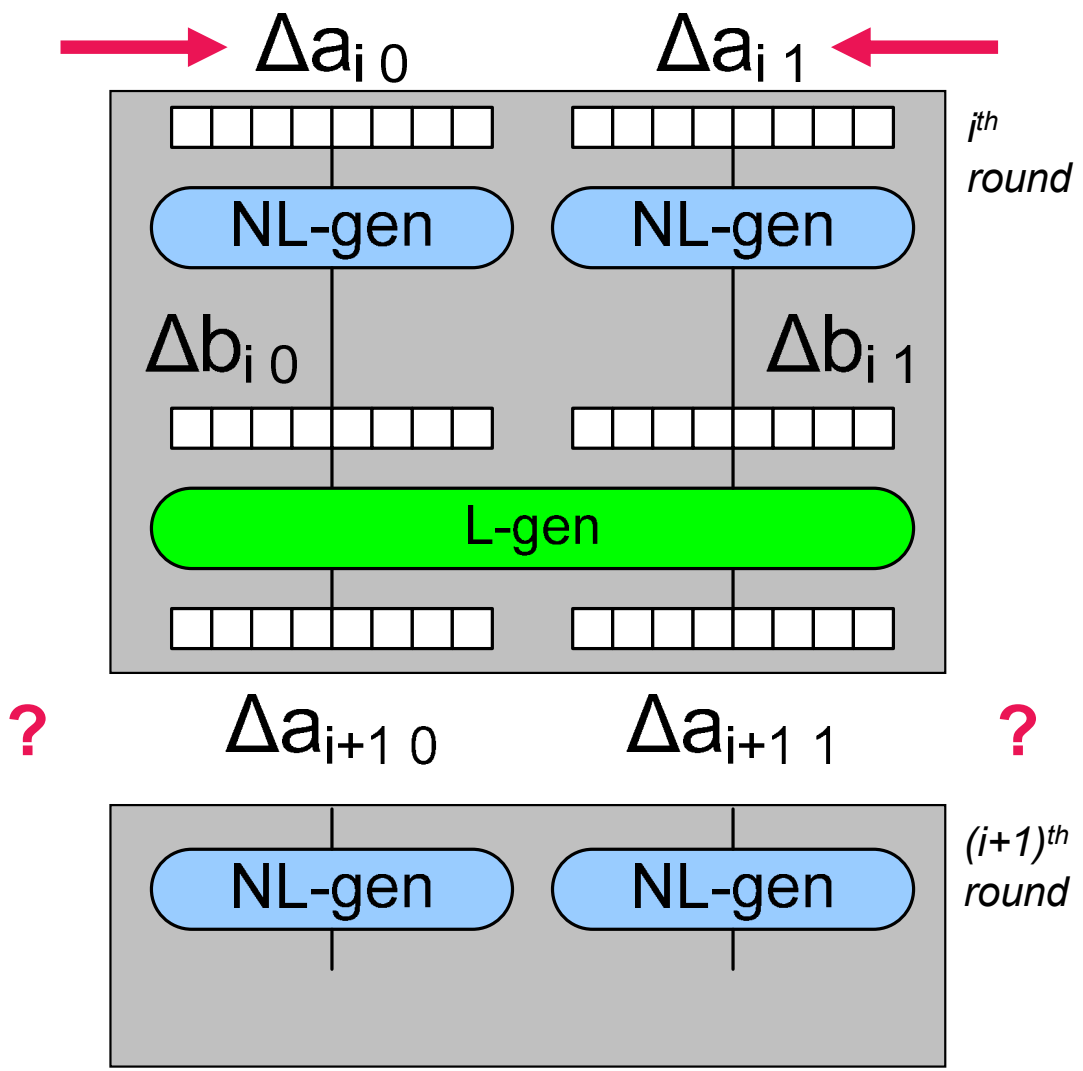
$\Leftrightarrow$

$\Delta Cost_{i+1} \leq C - C_{max}$

$\Leftrightarrow$

$f(\Delta a_{i+1\,0}) + f(\Delta a_{i+1\,1}) \leq C^{st}$

Going Through 1 Round in the Tree



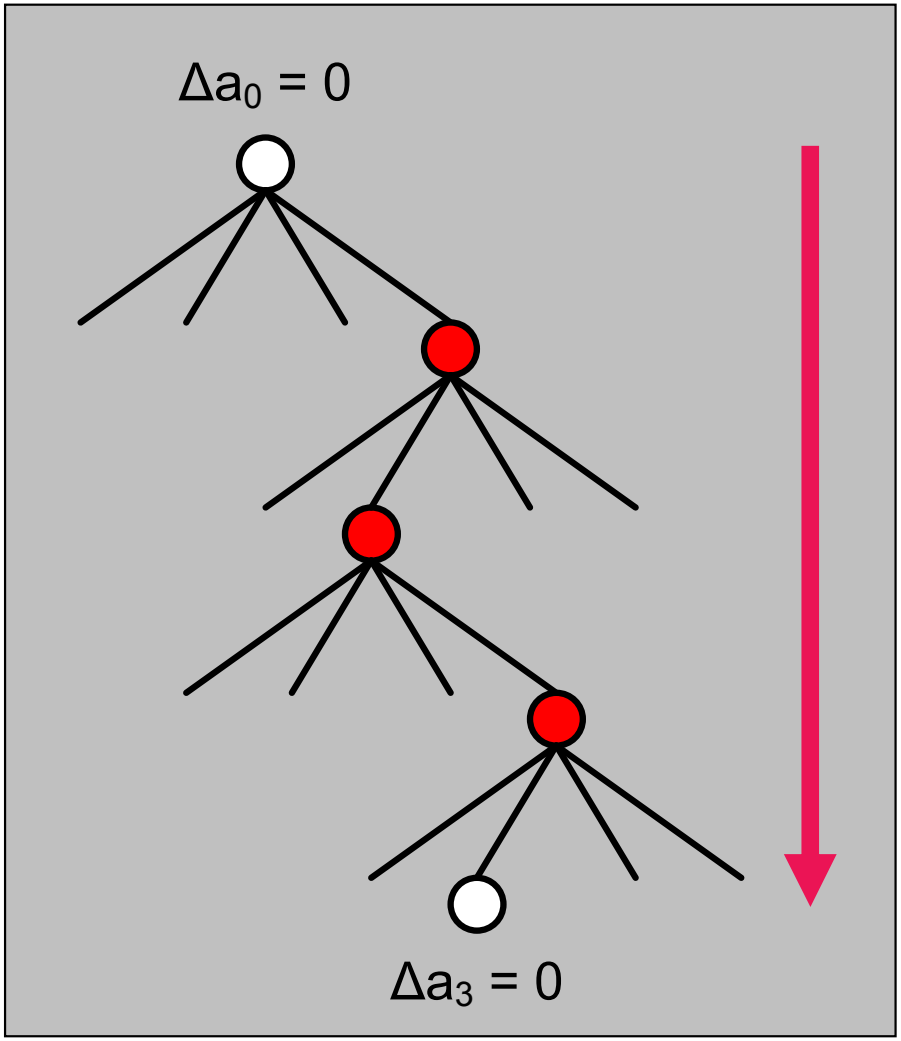
Problem:
Generate only Δa_{i+1} that satisfy

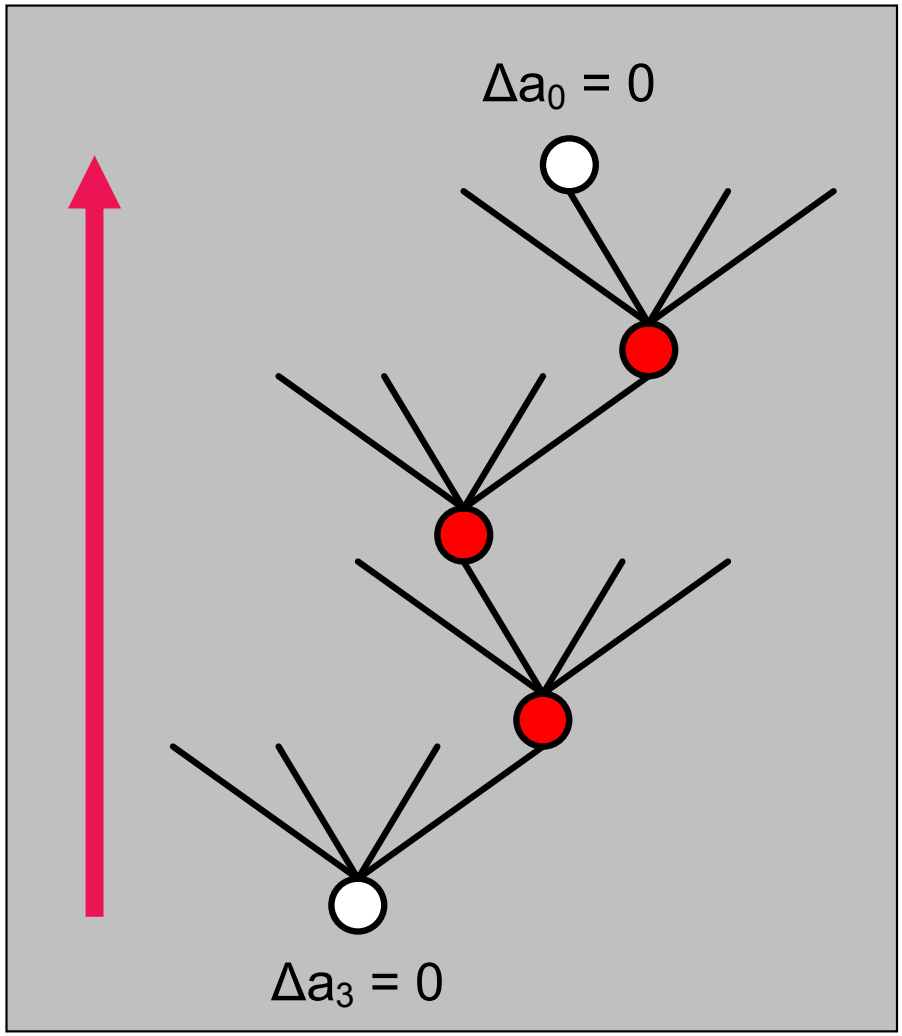
$$f(\Delta a_{i+1\,0}) + f(\Delta a_{i+1\,1}) \leq C^{st}$$

Solution:
1) Generate all of them
2) Discard those that don't satisfy the condition

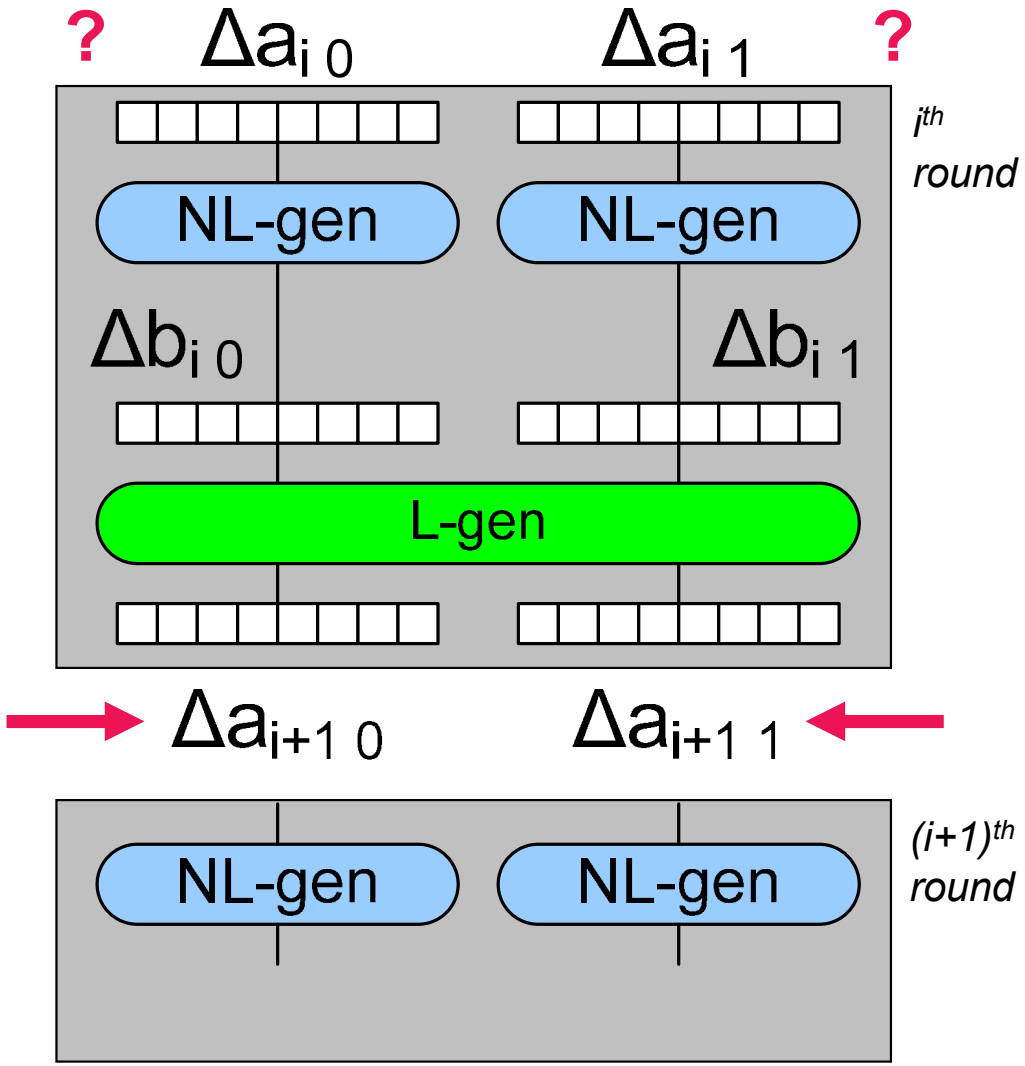
- "Stupid"
- Inefficient
- But only solution (?)

Walking the Tree Backwards





Walking the Tree Backwards



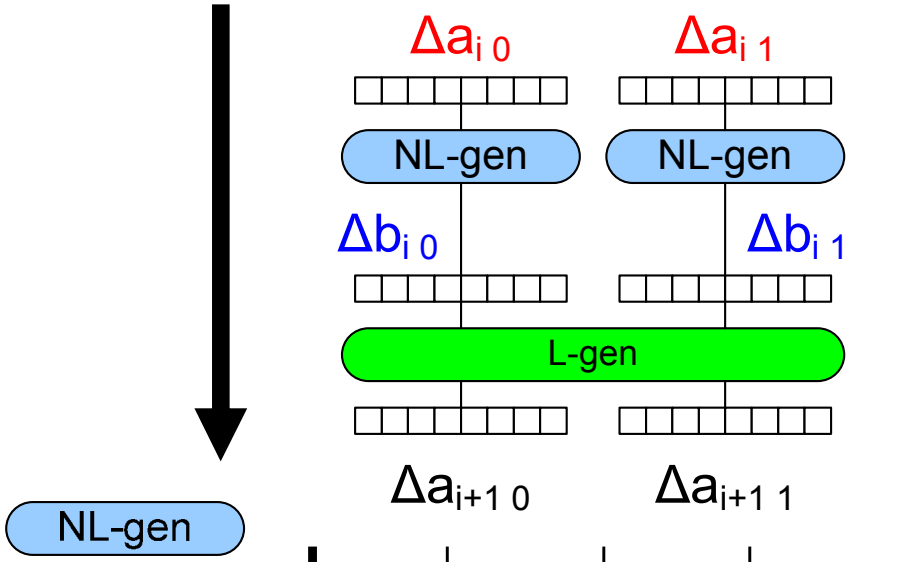
Problem:
Generate only Δa_i that satisfy

$$f(\Delta a_{i\,0}) + f(\Delta a_{i\,1}) \leq C^{st}$$

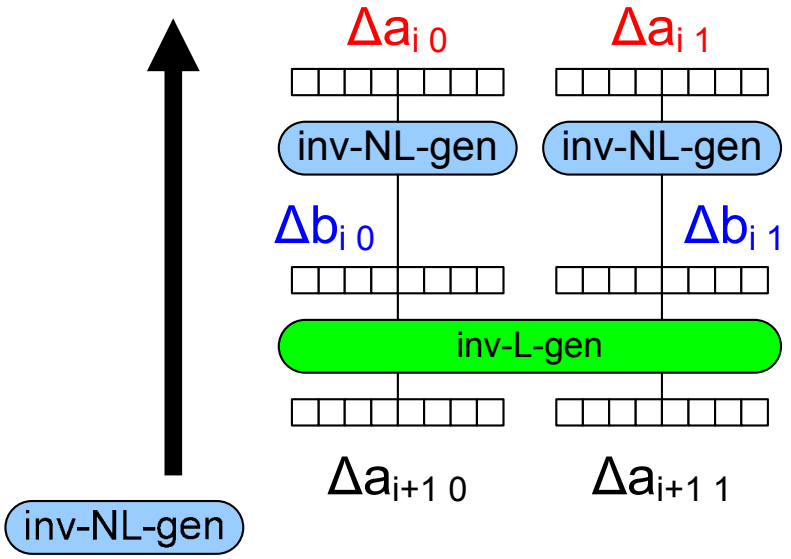
How?

By inverting the state generation functions in the tree

Inverting the Tree Node Generation Function

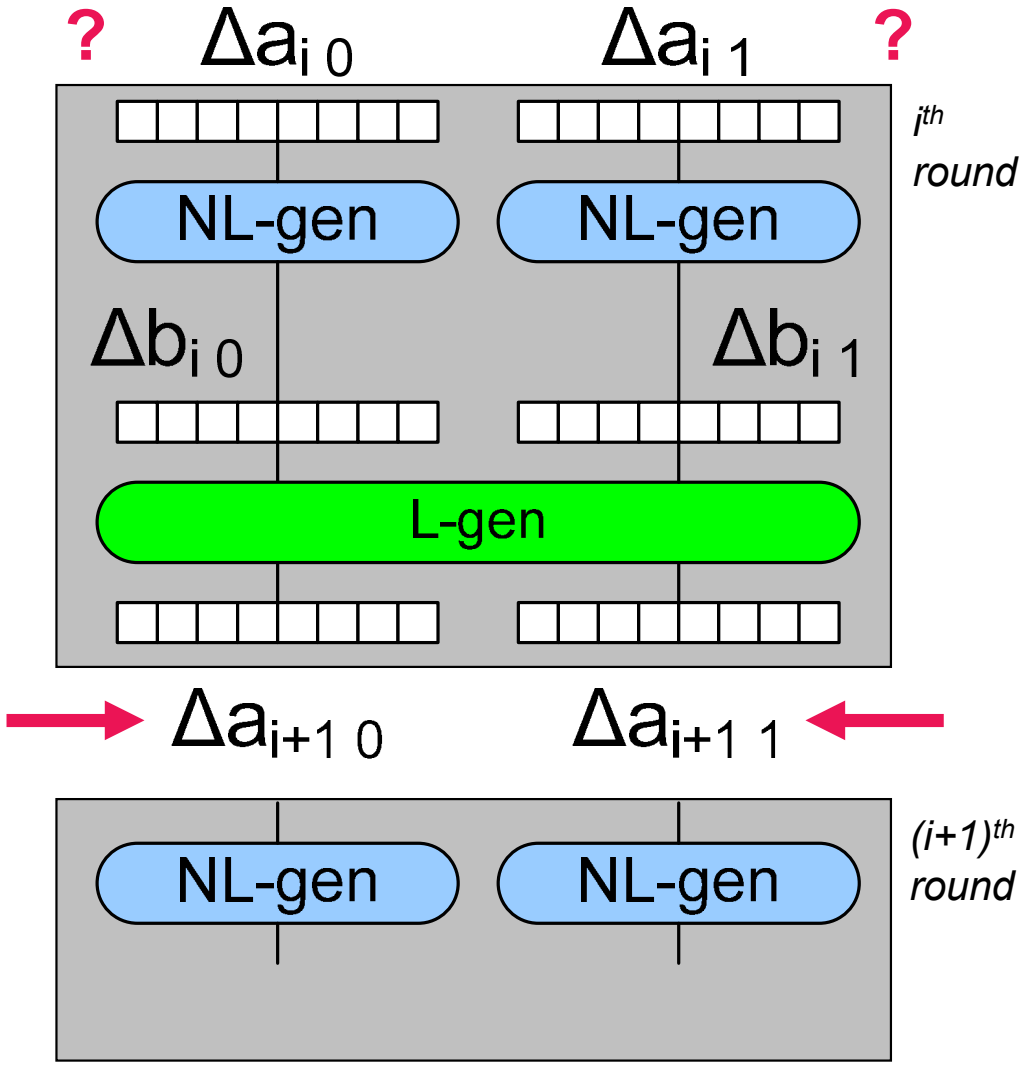


$f(\Delta a)$	0	*1*	2	3
…	∅	{**Δb**}	∅	∅
Δa	∅	∅	∅	{**Δb**}
…	∅	∅	{**Δb**}	∅



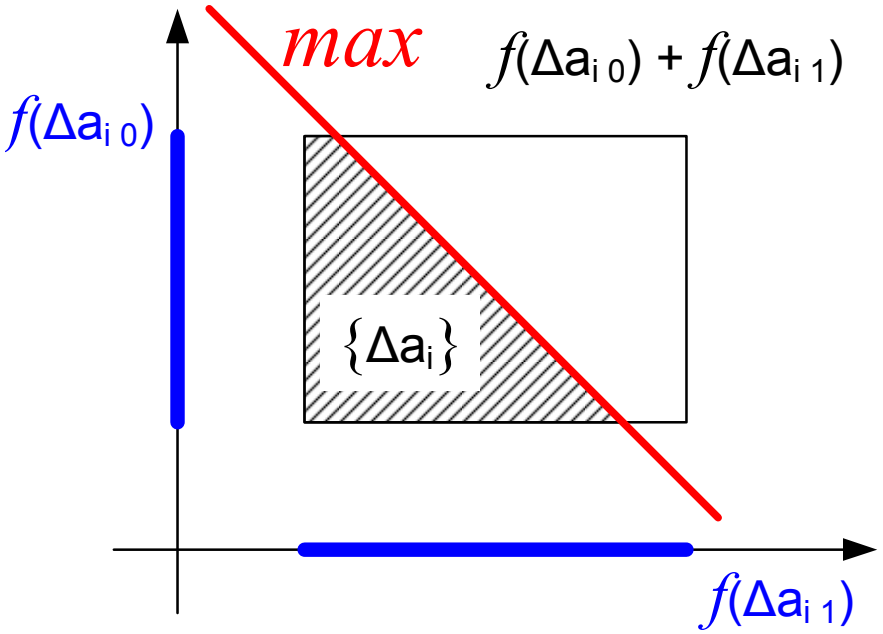
$f(\Delta a)$	*0*	1	2	3
…	∅	{**Δa**}	{**Δa**}	{**Δa**}
Δb	{**Δa**}	{**Δa**}	{**Δa**}	∅
…	{**Δa**}	{**Δa**}	∅	∅

Walking the Tree Backwards



inv-NL-gen	$f(\Delta a)$			
…	∅	{Δa}	{Δa}	{Δa}
Δb	{Δa}	{Δa}	{Δa}	∅
…	{Δa}	{Δa}	∅	∅

1) $\Delta a_{i+1} \Rightarrow \Delta b_i$
2) Divide & Conquer $\Rightarrow \Delta a_i$



Speed-up of 100x and more

Conclusions

Conclusions (I)

Architecture for Body Area Network

- Architecture for Body Area Network
- HW Link layer security
  - Low power & low gates
  - Encryption
    - 128-bit SSGen (2^{80} key security)
  - Authentication
    - 16-bit HW MAC (2^{16-x} resistance)
    - or 32-bit HW MAC (2^{32-x} resistance)
  - x to be refined (incl. eg. known plaintext attack)
- **Repeated MAC** mode to increase resistance
  - More resistance with exact same HW
  - At the discretion of the application
    - Eg. Use standard MAC for streaming, but repeated MAC for command data.

Conclusions (II)

Looking for Trails

- Faster cost-first search tree search for collision trails (100x and more)
- Based on
  - Assumption:

$$\min [\ \textit{cost increase}\ ] \Leftrightarrow \min [\ \sum_j f(\Delta a_{i\,j})\ ]$$

  - Walking in the tree backwards
  - Divide & Conquer
- Possible extensions
  - Meet-in-the-Middle?
  - Extend to more general assumptions?

$$\min [\ \textit{cost increase}\ ] \Leftrightarrow \min [\ \sum_j f(\Delta a_{i\,j,}\ \Delta b_{i\,j})\ ]$$