Wireless Network Security

Broadcast Authentication

Srdjan Čapkun

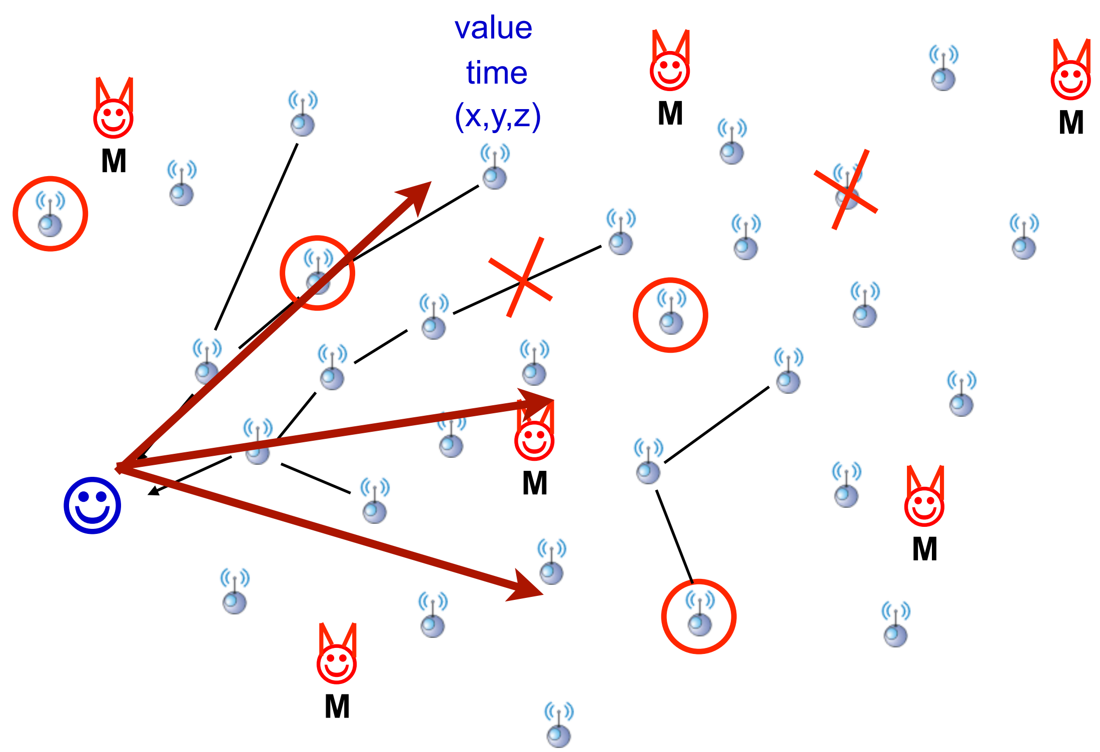
Broadcast Authentication

Tesla

Broadcast Authentication

Broadcast Message Authentication

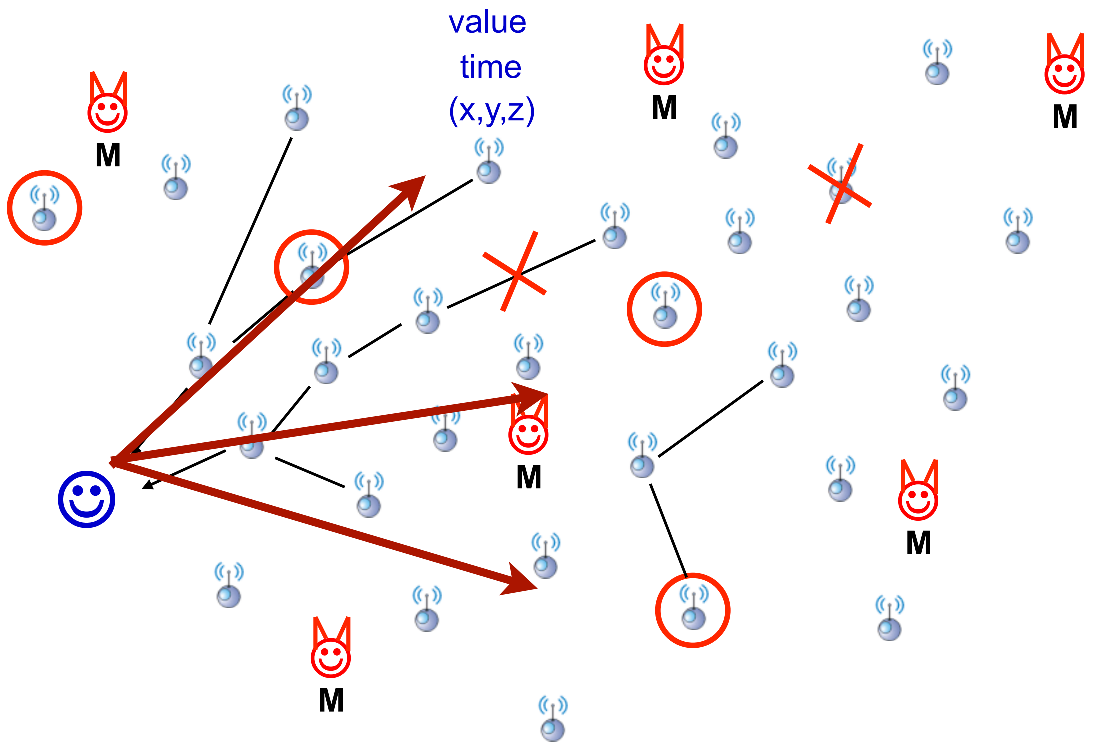
- One sender, a number of receivers *(possibly malicious and unknown to the sender)*.
- All receivers need to *verify the authenticity of the sender's messages*.



Broadcast Authentication

Broadcast Message Authentication

- One sender, a number of receivers *(possibly malicious and unknown to the sender)*.
- All receivers need to *verify the authenticity of the sender's messages*.

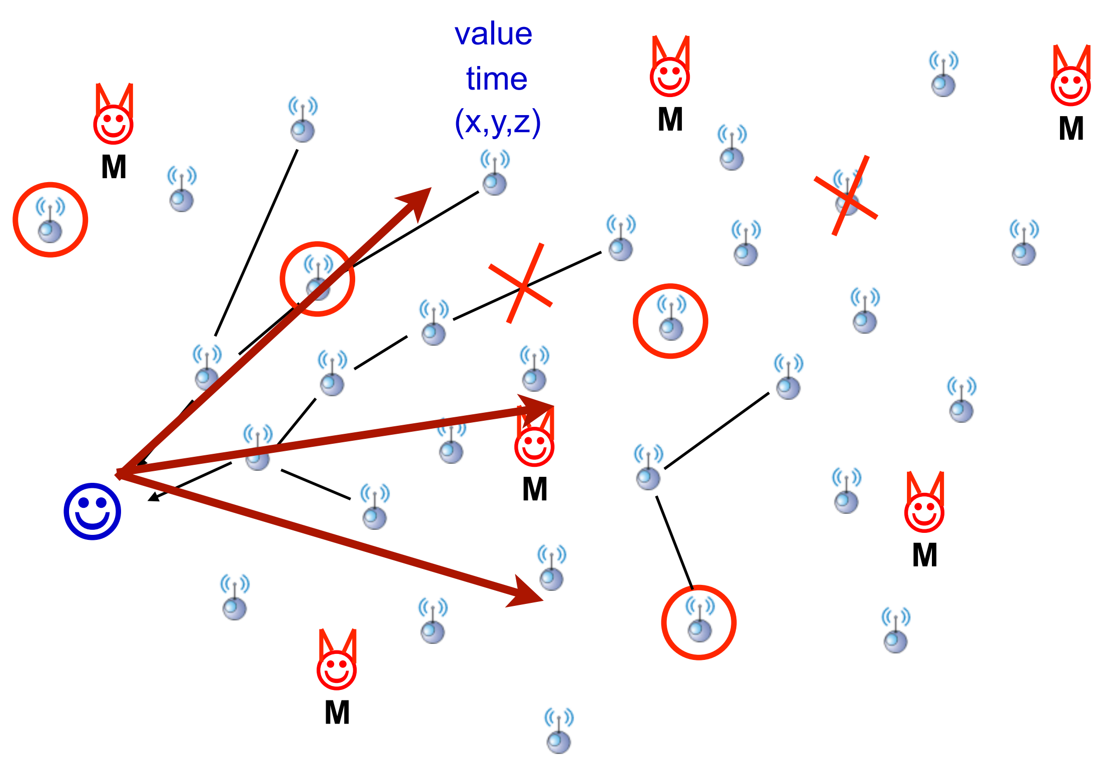


Any ideas how to solve this problem?

Broadcast Authentication

Broadcast Message Authentication

- One sender, a number of receivers *(possibly malicious and unknown to the sender)*.
- All receivers need to *verify the authenticity of the sender's messages*.



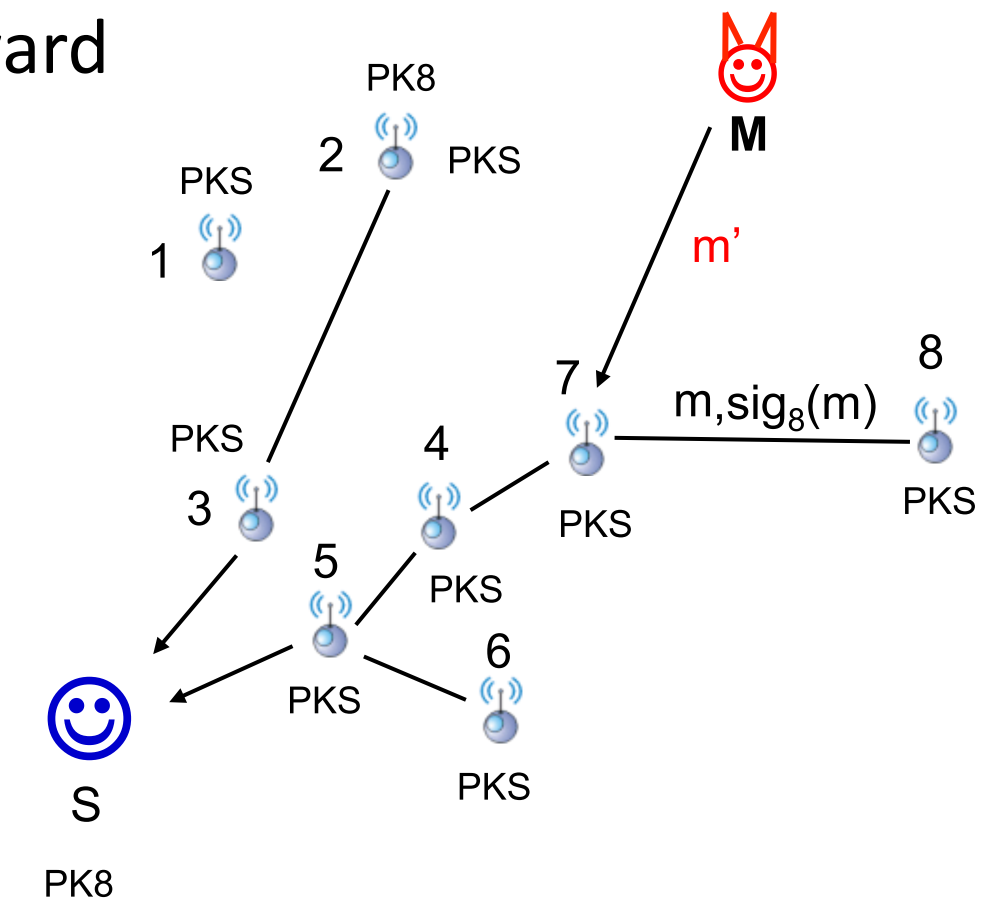
Any ideas how to solve this problem?

Efficiently?

Using Public-Key Cryptography for Broadcast Authentication

Using PK crypto in distributed networks is:

- simple
- effective
- enables broadcast authentication
- distribution of new keys and insertion of new nodes is straightforward
- ...
- ***expensive***



Resource-constrained Devices

Moteiv Tmote sky

8MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash)

250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver

Hardware link-layer encryption and authentication

Tinynode

8MHz Texas Instruments MSP430 microcontroller

868 MHz Xemics XE1205 multi channel wireless transceiver

RAM 10K bytes, Program Space 48K bytes, External Flash 512K bytes, Configuration Flash 256 bytes

Mica2, MicaZ, …

Example Costs of Crypto Operations (indicative)

Diffie-Hellman with 1,024-bit keys (Mica2)

- 54.1144 sec for key generation
- 1,250 B of SRAM
- 11,350 B of ROM
- 1.185 Joules (3.9897 x 108 cycles)

ECC with 163-bit keys (Mica2) by BBN (D. Malan)

- 34.390 sec for key generation
- 1,140 B of SRAM
- 34,342 B of ROM
- 0.82149 J (2.5289 x 108 cycles)

More ECC

- TinyECC takes ***12 to 16 seconds to verify a signature*** on MicaZ
- Sizzle from Sun, several seconds on Atmel chip

Symmetric-key computations: SKIPJACK blockcipher with 80-bit keys on Mica2

- *2,190 μsec for encrypt()*
- *3,049 μsec for computeMac()*

Broadcast Authentication without PK Crypto?

Can we enable broadcast authentication without PK crypto primitives?

Two approaches:

- Delayed Key Disclosure (Cheung, Tesla)
- Presence Awareness

Broadcast Authentication based on Delayed Key Disclosure

Main characteristics:

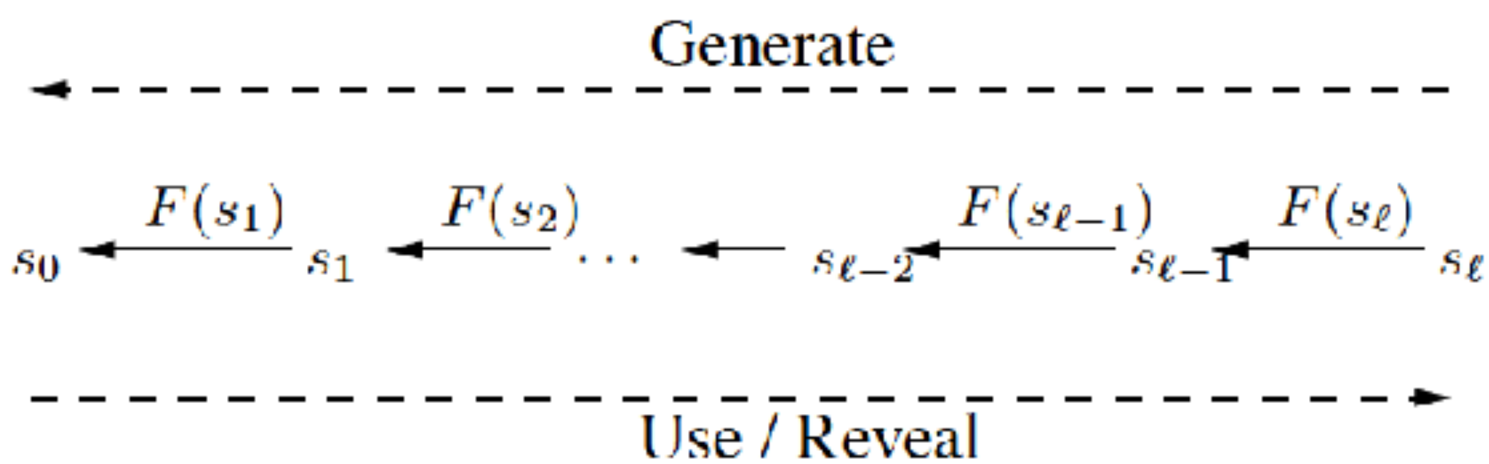
- Uses purely symmetric primitives (MACs)
- Asymmetry from delayed key disclosure
- Self-authenticating keys (one-way hash chains)
- Requires loose time synchronization

First proposal by Cheung in 97, follow-up proposal by Perrig in 2001 (named Tesla)

Tesla: http://sparrow.ece.cmu.edu/group/broadcast-authentication.html

Broadcast Authentication based on Delayed Key Disclosure (TESLA)

One-way chains:

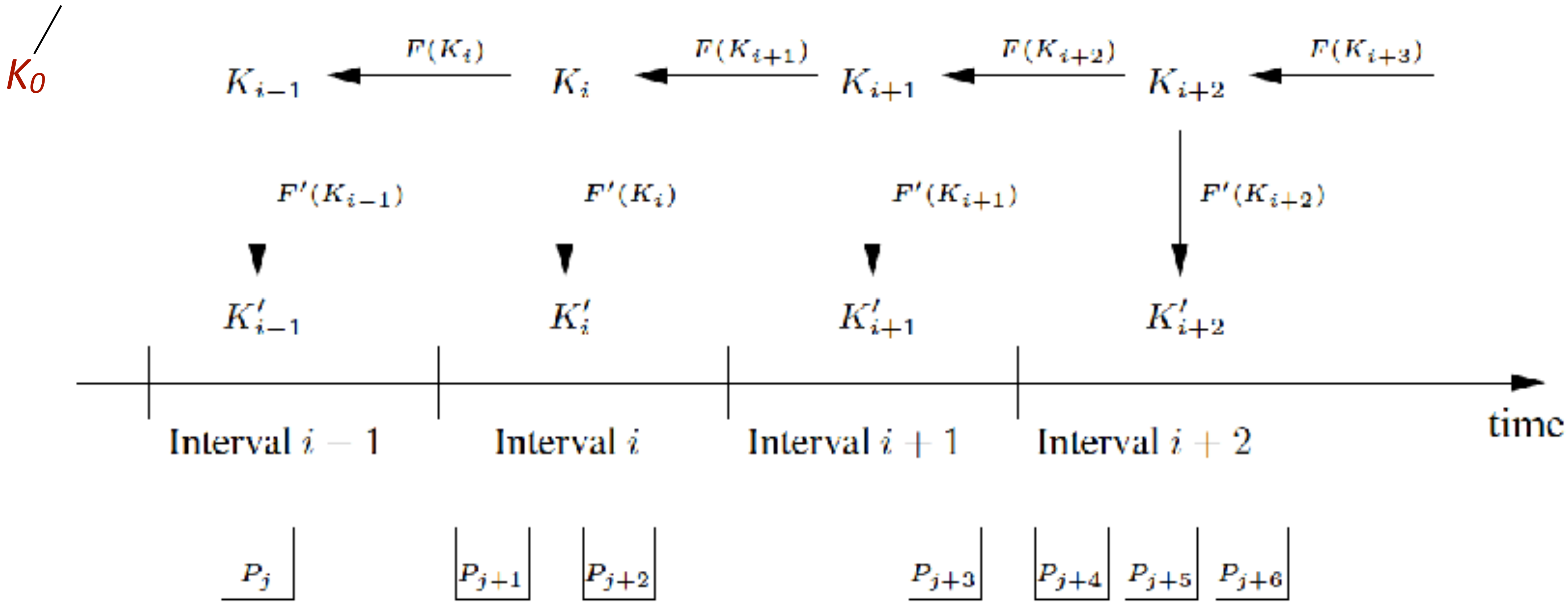


- s_ℓ is randomly chosen
- F(.) is a one-way (hash) function

- If an attacker knows s_i, it can easily generate s_{i-1}, (by applying F(.), but cannot generate s_{i+1}

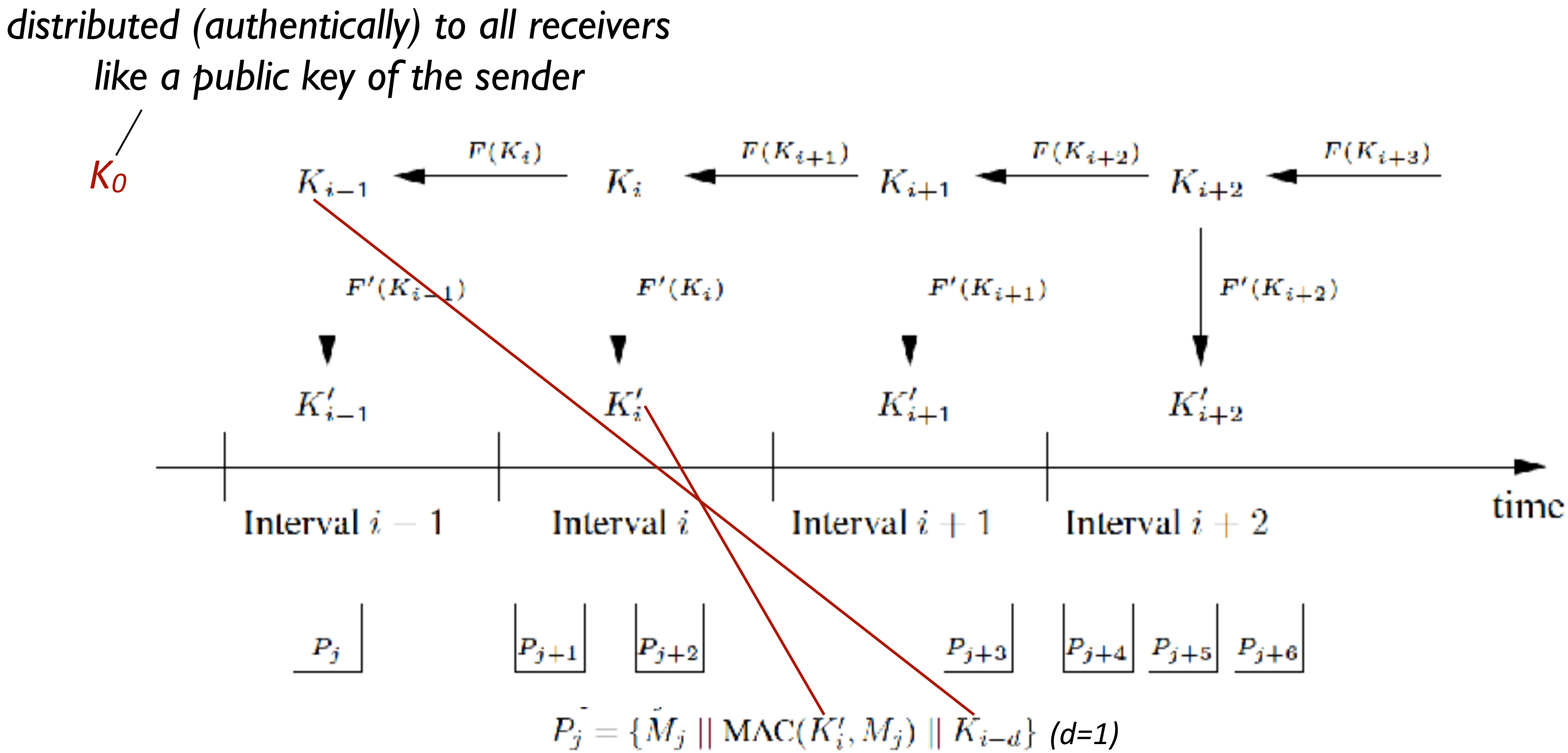
Broadcast Authentication based on Delayed Key Disclosure (TESLA)

distributed (authentically) to all receivers
like a public key of the sender



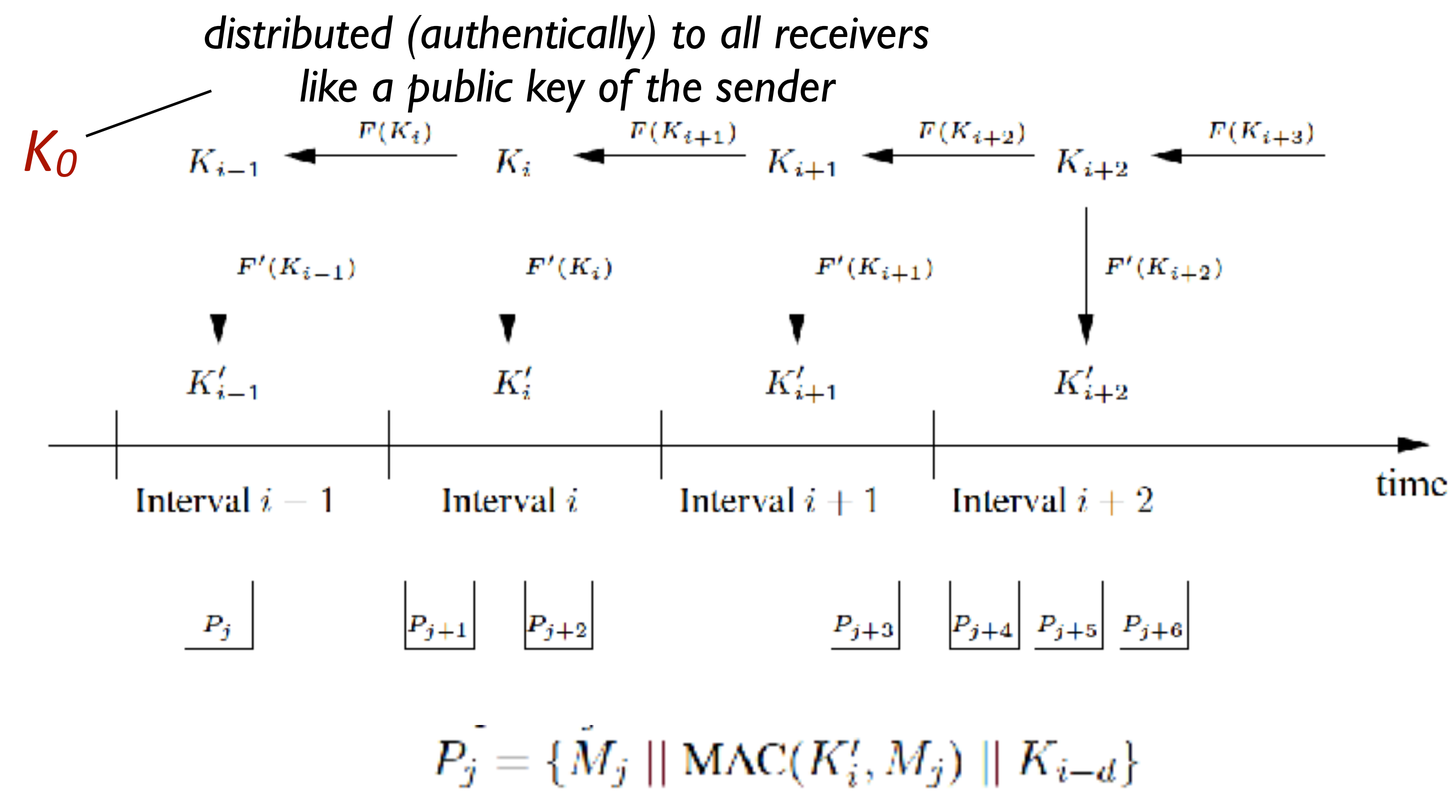
- Sender generates a key K_ℓ and keeps it confidential
- Generates K_0 and distributes it to all receivers

Broadcast Authentication based on Delayed Key Disclosure (TESLA)



- To transmit a message M_j, the sender MAC's M_j with the key of the current time interval (K_i')
- The key is used *ONLY WITHIN ITS INTERVAL*
- Each key is *explicitly disclosed in cleartext after the interval*

Broadcast Authentication based on Delayed Key Disclosure (TESLA)



Message Verification:

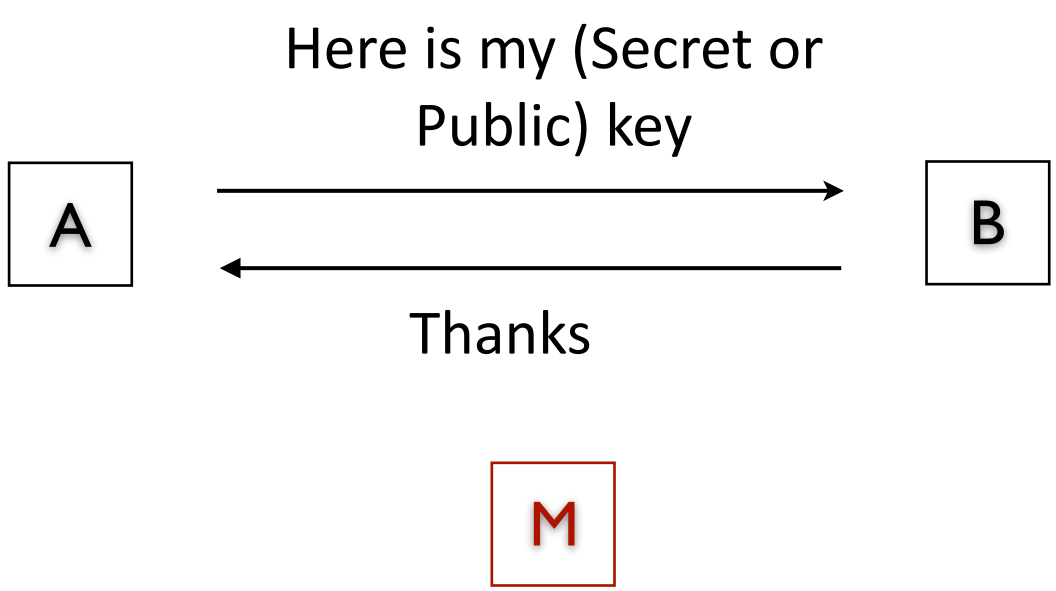
- Receive M_j
- Receive K_i
- Compute $K_i' = F'(K_i)$
- Verify MAC
- Verify that $F^n(K_i) = K_0$
- *Verify that the message was received within the key validity interval (before the key was disclosed)*

- The keys are authenticated using one-way hash chains
- The messages are authenticated using the keys
- If the key is used after the interval, the message is ignored

Wireless Device Pairing

Device Pairing: *Problem*

Given a pair of wireless devices, *how do they establish a secret key in the presence of an adversary* (passive or active – MITM attack) ?



Note: the devices have no preloaded keys / credentials (e.g., two mobile phones, a phone and a printer, ...)

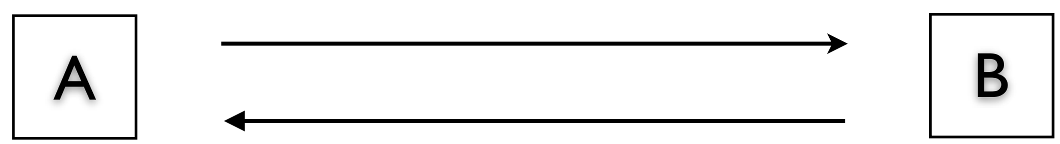
Device Pairing: *Diffie-Hellman Protocol*

DH protocol enables *secret key establishment by public communication*.

generate a

g^a mod p

generate b



g^b mod p

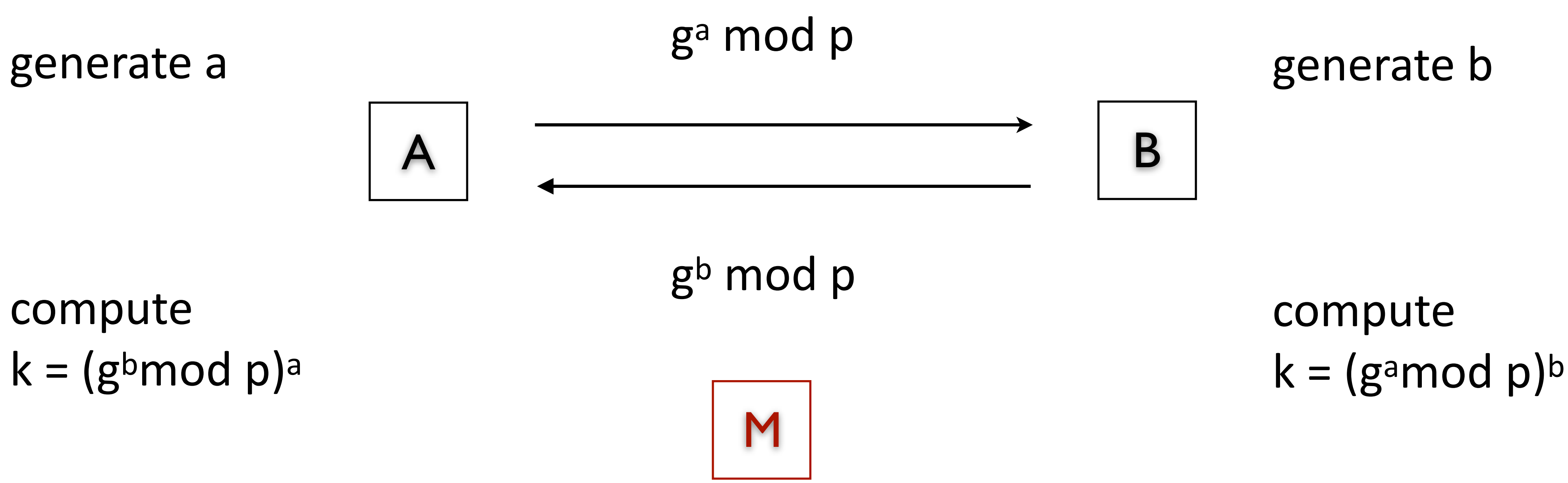
compute
$k = (g^b \bmod p)^a$

compute
$k = (g^a \bmod p)^b$

Given a prime *p*, a generator *g* of Z_p* and elements g^a *mod p* and g^b *mod p it is computationally difficult to find* g^{ab} *mod p.* Given g^x*mod p it is computationally difficuly to find x*.

Device Pairing: *Diffie-Hellman Protocol*

DH protocol enables *secret key establishment by public communication*.

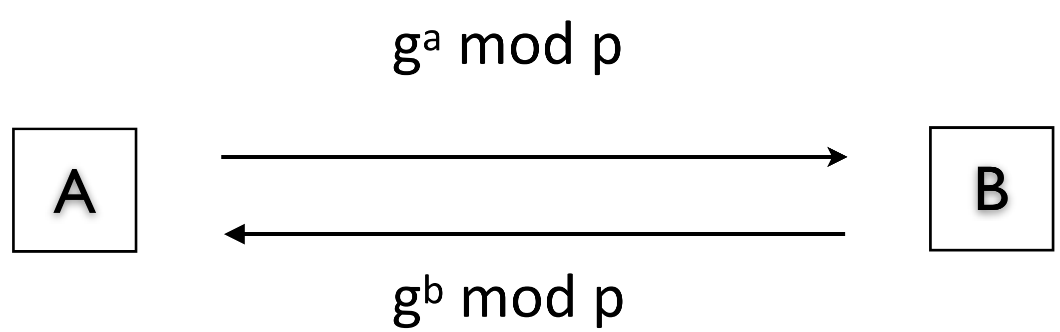


DH fully resists passive attackers (eavesdropping only).

DH is not secure against active attackers (MITM attacks).

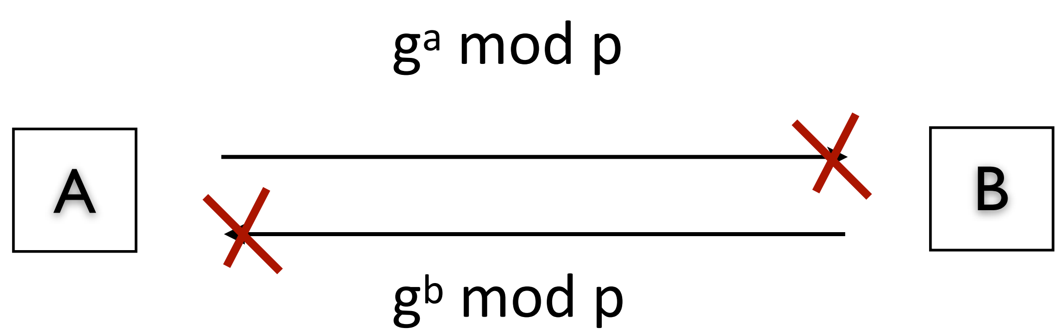
Device Pairing: *Diffie-Hellman Protocol*

DH is not secure against active attackers (MITM attacks).



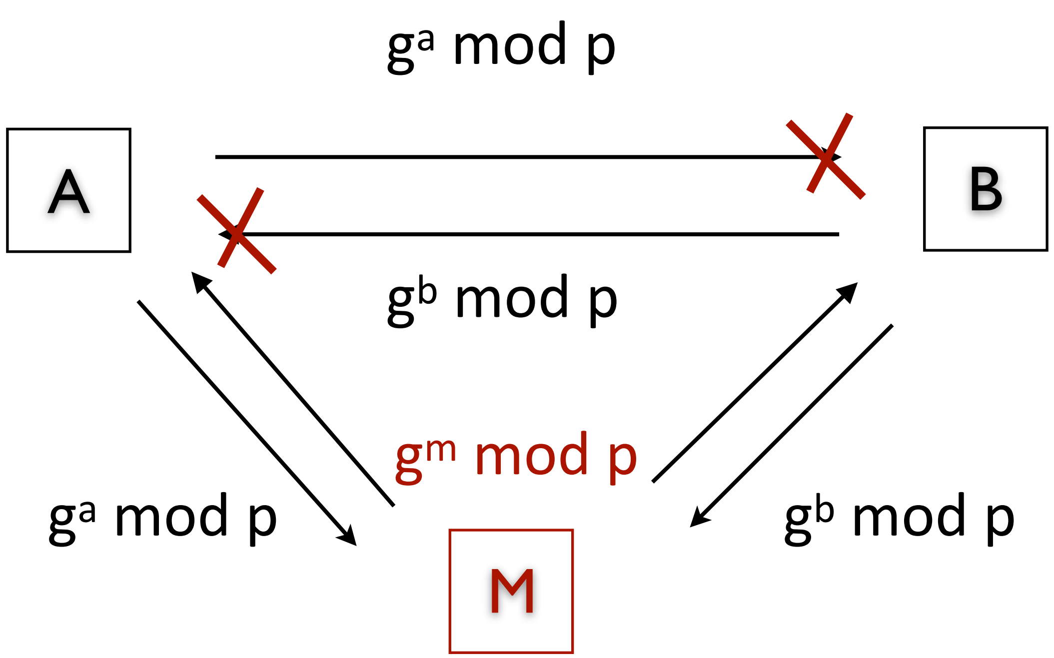
Device Pairing: *Diffie-Hellman Protocol*

DH is not secure against active attackers (MITM attacks).



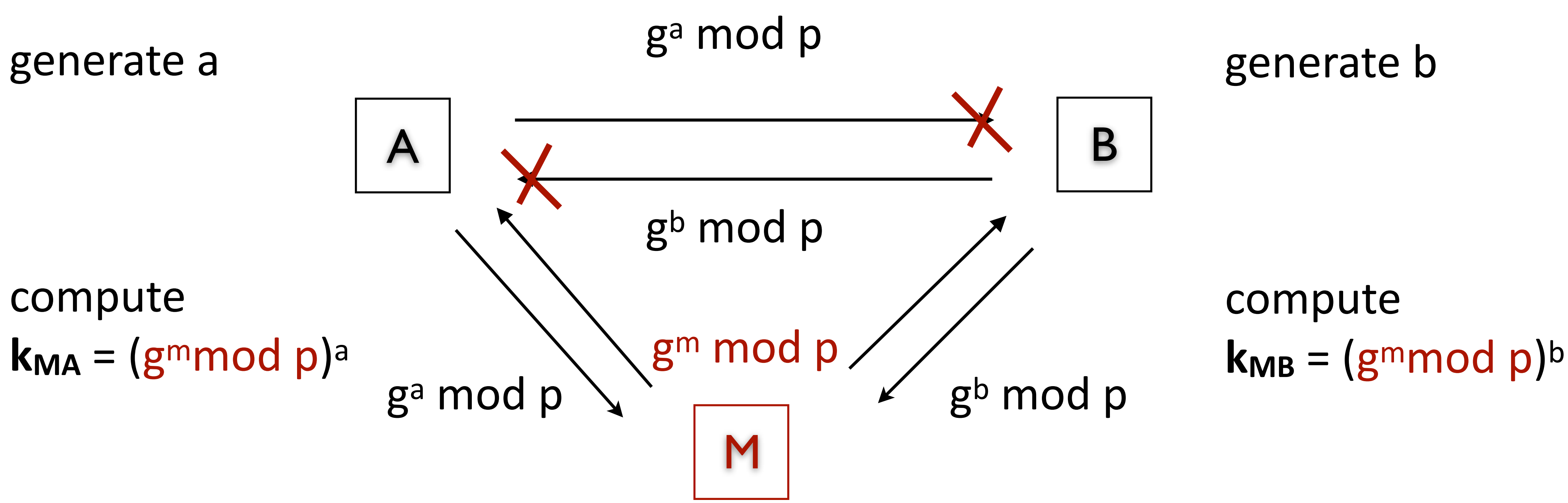
Device Pairing: *Diffie-Hellman Protocol*

DH is not secure against active attackers (MITM attacks).



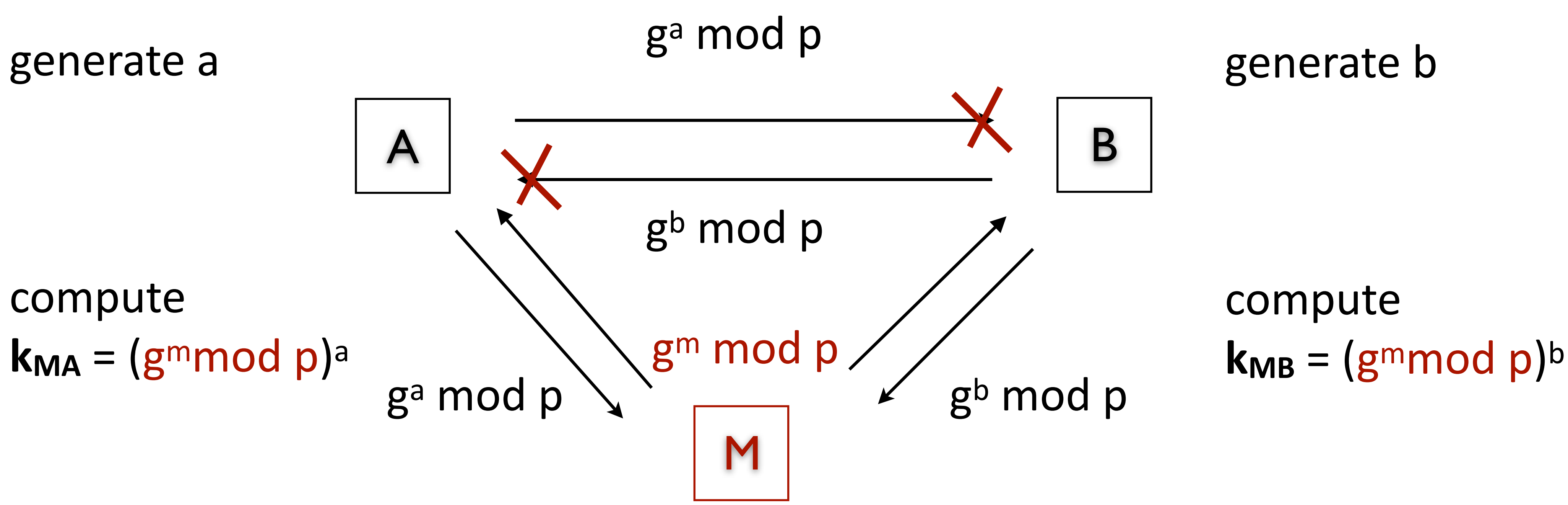
Device Pairing: *Diffie-Hellman Protocol*

DH is not secure against active attackers (MITM attacks).



Device Pairing: *Diffie-Hellman Protocol*

DH is not secure against active attackers (MITM attacks).



DH keys / contributions (g^a mod p and g^b mod p) therefore *need to be authenticated* or there has to be a procedure to *verify with whom the key was established*.

Device Pairing

Device Pairing can be built using

- Diffie-Hellman (i.e., using public-key crypto)
- Using symmetric key techniques *(under some special assumptions)*

Pairing is easy if the devices can verify each-other's certificates (they can then authenticate their DH keys/contributions by signatures).

Device Pairing: A Large Number of Proposals

- Resurrecting duckling (Stajano, Anderson), *physical contact*
- Balfanz et al. location-limited channel (e.g., *infrared link*)
- Asokan, Ginzboorg, *shared password*
- Jakobsson, Larsson, solutions to derive a strong key from a *shared weak key*
- Castellucia, Mutaf, *device signal indistinguishability*
- ... button presses, accelerometers, sound, *PIN entry (BT)*...

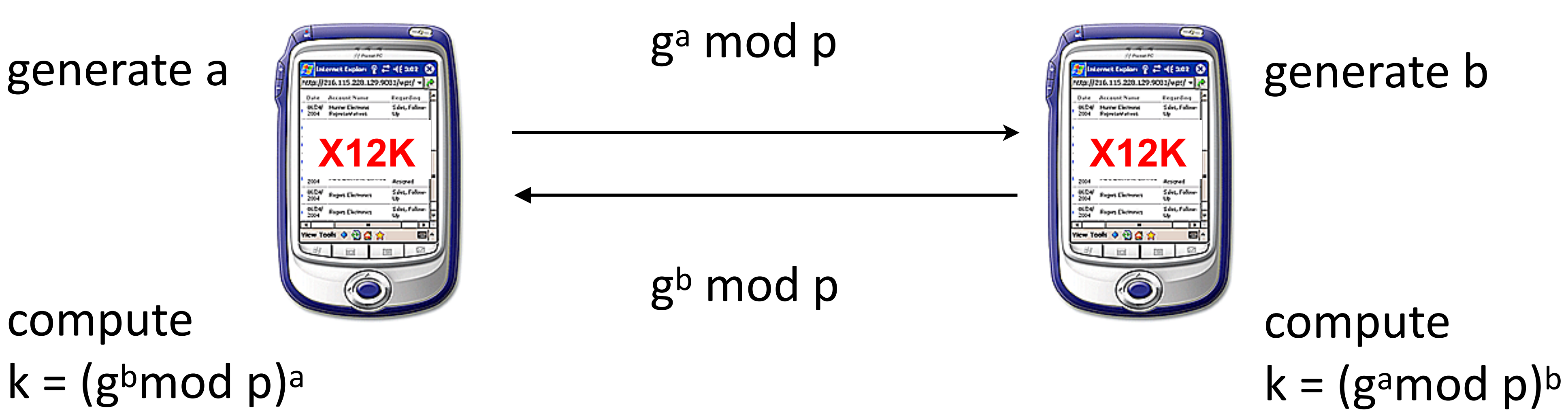
- Cagalj, Capkun, Hubaux, *distance bounding*
- Perrig and Song, *Public-key hash visualization*
- Gehrmann et al., *short string comparison*
- Cagalj, Capkun, Hubaux, *short string comparison*
- Dohrmann and Ellison, *short word comparison*
- ...

Device Pairing: Short String Comparison

Maher, 93, US patent, Gehrmann et al 01,03,04, (MANA I, II, III)

Steps:

- Establish key k using DH
- *Hash the key h(k) and display on both devices*
- Compare the displayed values (160 bits = 20 characters)



Device Pairing: Seeing is Believing

McCune et al. 05, Seeing is believing

Idea:

- Send the public key over an authentic channel (visual).

	A		**B**
1	$h_A \leftarrow Hash(PK_A)$		
2		$\xrightarrow{h_A}$ (visual)	
3		$\xrightarrow{PK_A}$ (other)	$h' \leftarrow Hash(PK_A)$
4			$if\, h' \neq h_A\ then\ abort$

Device Pairing: Loud and Clear

Goodrich et al. 05

Idea

Human-assisted string comparison using voice communication

Steps:

- A hashes its public key PK
- *h(PK) mapped to a recognizable sentence (public mapping)*
- sentence transmitted over the voice channel
- PK transmitted over the wireless channel
- B compares the maps the sentence to the hash of PK

Similar: on-line authentication

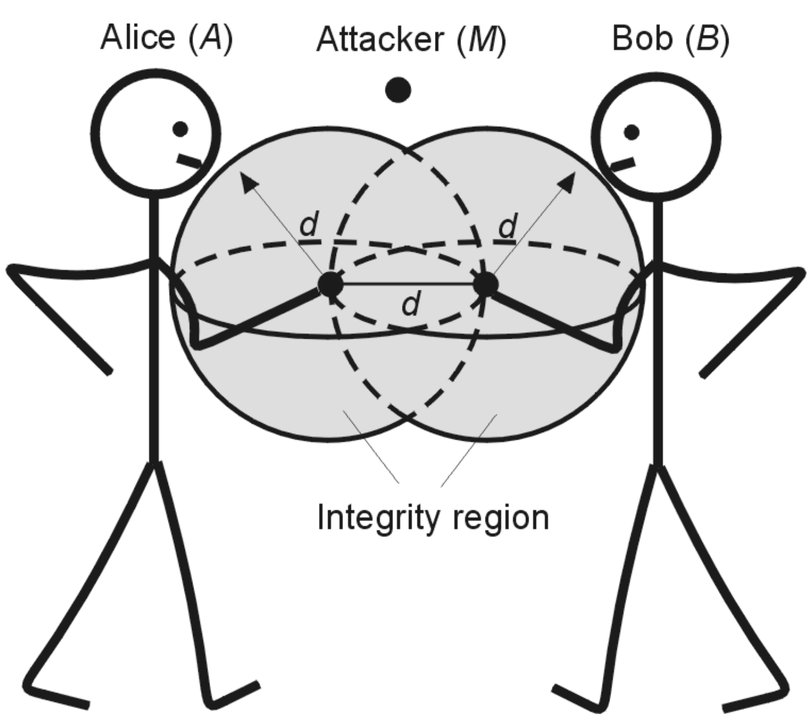
(e.g., for Secure VOIP applications) http://zfoneproject.com/

Device Pairing: Integrity Regions

Capkun, Cagalj 06

Idea:

- Establish key k using DH
- *Authenticate DH keys by physical proximity (distance bounding)*
- 'if the DH key comes from a close proximity it comes from a friend'



Device Pairing: Shake Them Up!

Castelluccia, Mutaf 05

Problem:

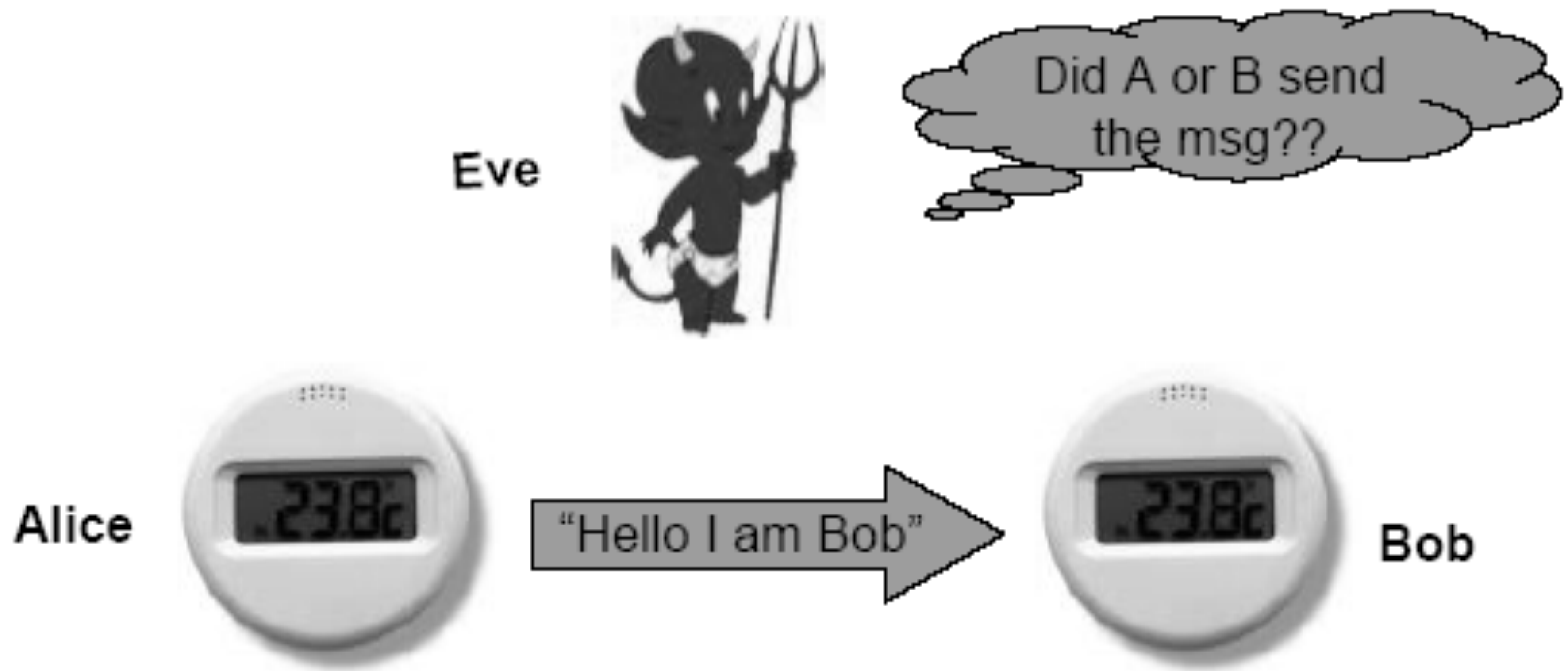
- Resource-constrained devices need to establish keys
- DH (PK crypto) is not an option (too expensive)

Idea:

- Rely on the fact that the attacker does not know which device transmits at which time ...

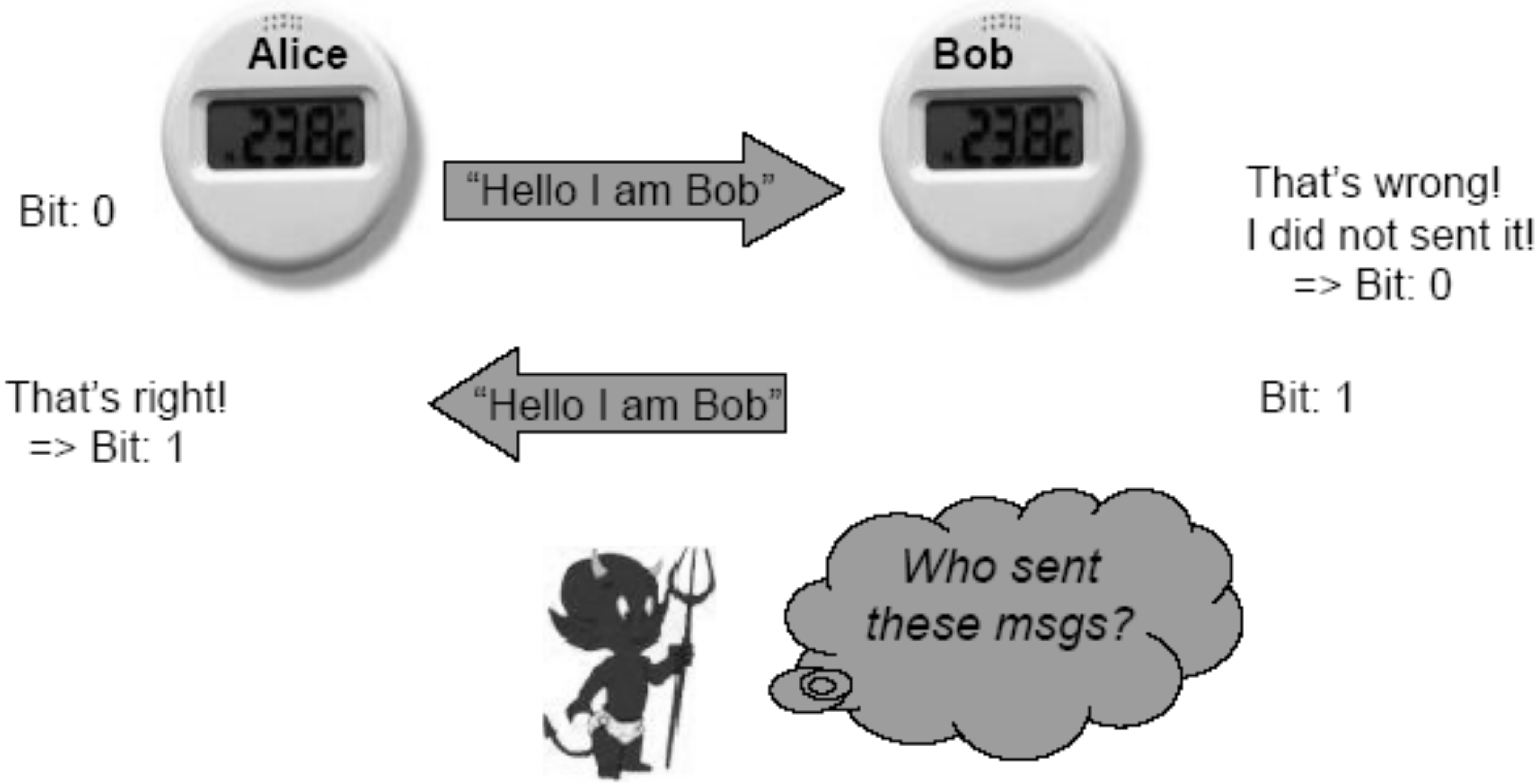
Device Pairing: Shake Them Up!

- Let's assume that Alice (A) and Bob (B) communicate over a wireless *anonymous* broadcast channel
  - Eve can read the exchanged packets
  - ...but can not identify the source of the packets.



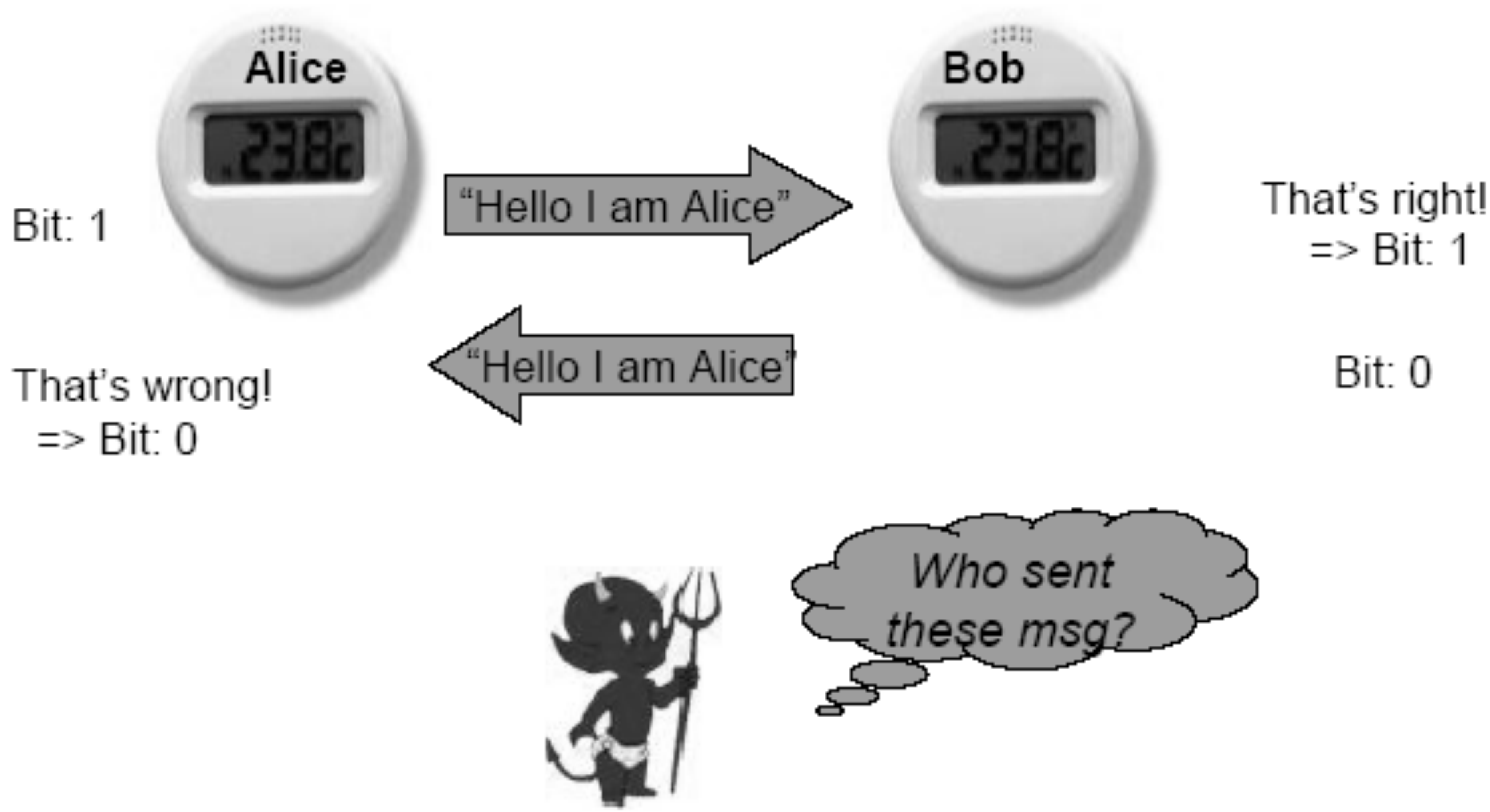
Device Pairing: Shake Them Up!

- Alice and Bob can then use the following algorithm:



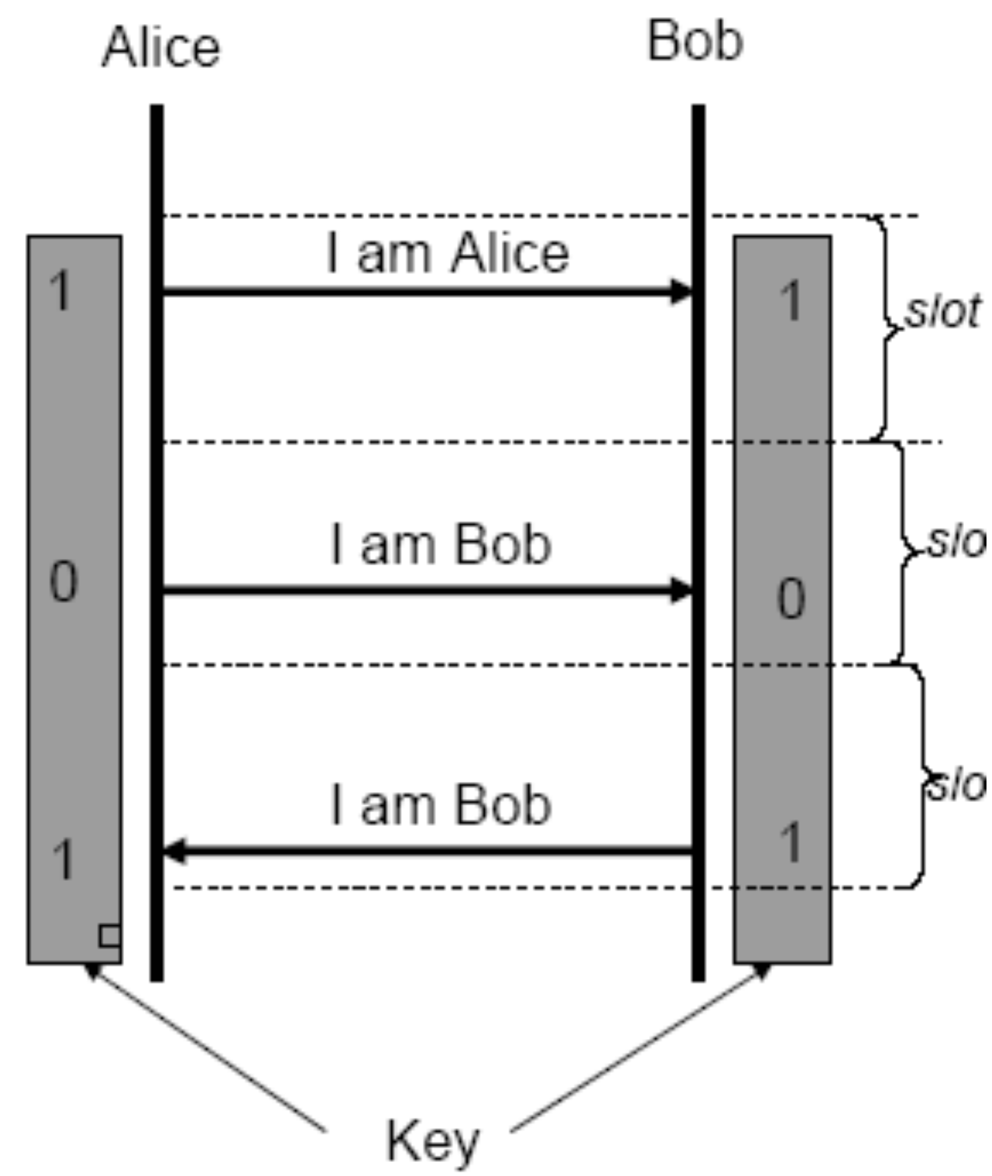
Device Pairing: Shake Them Up!

- Of course the protocol is symmetrical i.e. Alice can also send the bit "1" and Bob the bit "0"



Device Pairing: Shake Them Up!

- Divide the time in N slots.
- In each slot, either A or B sends a message
- Transmission order is random
  ➔ Eve can not group the messages and retrieve the key…



Device Pairing: Shake Them Up!

Idea:

- Device indistinguishability

Some issues

- Synchronization (done through shaking ?)
- Signal fingerprinting (power, frequency, ...) need to be addressed before using this approach

Device Pairing: Conclusion

DH can be protected against MITM attacks without previously established keys/certificates

- physical contact
- device indistinguishability (anonymity)
- string comparison (voice communication)
- image comparison (hash visualization)
- distance bounding (physical presence verification)

The string length is a security parameter that can be modified and adjusted for each particular application.

- We can do it without PK (Shake, Accelerometers, ..)

Device Pairing: Protocol issues

DH can be protected against MITM attacks without previously established keys/certificates

- physical contact
- device indistinguishability (anonymity)
- string comparison (voice communication)
- image comparison (hash visualization)
- distance bounding (physical presence verification)

The string length is a security parameter that can be modified and adjusted for each particular application.

- We can do it without PK (Shake, Accelerometers, ..)