2006-08-03

Status
Released Version

Revision
V10r00

Document File Name
Simple Pairing_WP

Document Owner
Core Specification Working Group

E-mail Address
radio-feedback@bluetooth.org

Bluetooth®
SPECIAL INTEREST GROUP

SIMPLE PAIRING WHITEPAPER

ABSTRACT: This whitepaper provides an overview of the cryptographic procedures and algorithms for the Simple Pairing feature in the Lisbon release of the Bluetooth Core Specification. The purpose of this whitepaper is to provide information to the security community for a peer review prior to the adoption of the Lisbon release.

Revision History		
Revision	**Date**	**Description**
V10r00	2006-08-03	Released version

Contributors	
Name	**Company**
Joel Linsky (Document Owner)	RF Micro Devices
Terry Bourk	RF Micro Devices
Ayse Findikli	Atheros
Robert Hulvey	Broadcom
Shawn Ding	Broadcom
Robin Heydon	CSR
Steven Singer	CSR
Simon Kingston	CSR
Steven Wenham	CSR
Dave Suvak	iAnywhere
Mattias Edlund	Infineon
Penny Chen	Intel
Selim Aissi	Intel
Peter Hauser	Microsoft
Josh Benaloh	Microsoft
Gideon Yuval	Microsoft
Yacov Yacobi	Microsoft
Joby Lafky	Microsoft
Dan Simon	Microsoft
David Roberts	Microsoft
Don Stanwyck	Microsoft
Kristin Lauter	Microsoft
Greg Muchnik	Motorola
Kanji Kerai	Nokia
Kaisa Nyberg	Nokia
N. Asokan	Nokia
Noel Lobo	Nokia
Philip Ginzboorg	Nokia
Dominique Everaere	Philips
Reinhard Meindl	Philips
Guido Bertoni	ST
Eran Reuveni	TI
Yoshimitsu Shimojo	Toshiba

CONTENTS

1 Purpose

This whitepaper describes the cryptographic functions, protocols and algorithms used in the Simple Pairing feature of the forthcoming Lisbon release of the Bluetooth Core Specification. This whitepaper is being provided to the security community prior to the adoption of the Lisbon release for the purpose of peer review.

Any feedback is appreciated and should be e-mailed to:

radio-feedback@bluetooth.org

Since the Simple Pairing feature is still under development, what is included in the adopted version of the Lisbon Specification may not be identical to the details provided in this whitepaper.

Note that this whitepaper does not discuss key management after the Simple Pairing procedures have completed.

2 Overview of Simplified Pairing

This whitepaper provides an overview of the association models, cryptographic functions, protocols and algorithms used in the Simple Pairing feature of the Lisbon release of the Bluetooth Core Specification. The primary goal of Simple Pairing is to simplify the pairing process from the point of view of the user. Secondary goals are to maintain or improve the security in Bluetooth wireless technology. Since high levels of security and ease-of-use are often at opposite ends of the spectrum in many technologies and products, much care has been taken by the Core Specification Working Group in the creation of the Simple Pairing proposal to maximize security while minimizing complexity from the end user's point of view.

Simple Pairing has two security goals: protection against passive eavesdropping and protection against man-in-the-middle attacks (active eavesdropping). It is also a goal of Simple Pairing to exceed the maximum security level provided by the use of a 16 character alphanumeric PIN with the pairing algorithm used in Bluetooth Core Specification 2.0 + EDR and earlier versions. It should be noted that many Bluetooth devices today use a 4-digit PIN or a fixed PIN of commonly known values significantly limiting the security on the link.

2.1 PASSIVE EAVESDROPPING PROTECTION

A strong link key coupled with a strong encryption algorithm is necessary to give the user protection against passive eavesdropping. The strength of the link key is based on the amount of entropy (or randomness) in its generation process which would not be known by an attacker. Using legacy pairing[1], the only source of entropy is the PIN which, in many use cases, is typically four digits either selected by the user or fixed for a given product. Therefore, if the pairing procedure and one authentication exchange is recorded, one can run exhaustive search to find the PIN in a very short amount of time on commonly available computing hardware.[2] With Simple Pairing, the recording attack becomes much harder as the attacker must solve a hard problem in public key cryptography in order to derive the link key from the recorded information. This protection is independent of the length of the passkey or other numeric values that the user must handle. Simple Pairing gives the same resistance against the recording and passive eavesdropping attacks even when the user is not required to do anything.

Simple Pairing uses Elliptic Curve Diffie-Hellman (ECDH) public key cryptography[3] as a means to thwart passive eavesdropping attacks. ECDH provides a very high degree of strength against passive eavesdropping attacks but it may be subject to man-in-the-middle attacks, which are much more difficult to perform in practice than the passive eavesdropping attack (see Section 2.2).

Using the Bluetooth Core Specification version 2.0 + EDR security protocols, a 16 character, numeric digit PIN achieves about 53 bits of entropy whereas a 16 character, case sensitive, alphanumeric PIN yields about 95 bits of entropy when the entire 62 character set is used ([0, … 9, 'A', … 'Z', 'a', … 'z']). Simple Pairing has approximately 95 bits of entropy using the FIPS approved P192 elliptic curve[4] which is at least as good as the entropy achieved in Bluetooth Core Specification version 2.0 + EDR using a 16 character, case sensitive, alphanumeric PIN. Simple Pairing, therefore, exceeds the requirements of the Bluetooth SIM Access Profile (SAP)[5] which has the most stringent security requirements of the Bluetooth profiles. ECDH cryptography was selected over standard Diffie-Hellman (often referred to as DH76[6]) since it is computationally less complex and less likely to exceed the low computational capacity in common Bluetooth Controllers.

---

[1] Bluetooth 2.0 + EDR and earlier
[2] http://www.eng.tau.ac.il/~yash/shaked-wool-mobisys05/index.html
[3] See: http://en.wikipedia.org/wiki/Elliptic_Curve_Diffie-Hellman for more details
[4] http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf
[5] SIM Access Profile: http://www.bluetooth.org/foundry/adopters/document/SAP_SPEC_V10/
[6] Diffie, W., and M.E. Hellman. ``New Directions in Cryptography." *IEEE Transactions on Information Theory, v. IT-22, n.*. (November 1976).

2.2 MAN-IN-THE-MIDDLE PROTECTION

A man-in-the-middle (MITM) attack[7] occurs when a user wants to connect two devices but instead of connecting directly with each other they unknowingly connect to a third (attacking) device that plays the role of the device they are attempting to pair with. The third device then relays information between the two devices giving the illusion that they are directly connected. The attacking device may eavesdrop on communication between the two devices (known as active eavesdropping) and is able to insert and modify information on the connection. In this type of attack, all of the information exchanged between the two devices is compromised and the attacker may inject commands and information into each of the devices thus potentially damaging the function of the devices. Devices falling victim to the attack are capable of communicating only when the attacker is present. If the attacker is not active or is out of the range, the two victim devices will not be able to communicate directly with each other and the user will notice it.

Simple Pairing protects the user from man-in-the-middle attacks with a goal of offering a 1 in 1,000,000 chance that a man-in-the-middle[8] could mount a successful attack (a probability that is considered low enough to meet FIPS 140-2[9] requirements for authentication). The strength of the man-in-the-middle protections was selected to minimize the user impact by using a six digit number for numerical comparison and Passkey entry. This level of protection was selected since, in most cases, users can be alerted to the potential presence of a man-in-the-middle attacker when the connection process fails as a result of a failed man-in-the-middle attack. While most users feel that, provided that they have not had their passkey compromised, a 4-digit passkey is sufficient for authentication (i.e. bank card PIN codes), the use of six digits was deemed to have little perceivable usability impact.

[7] http://en.wikipedia.org/wiki/Man_in_the_middle_attack
[8] The Core Specification Working Group has not been able to determine if there is a standard or known precedence for this probability in consumer devices.
[9] See http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf Section 4.3.3

3 Simple Pairing Association Models

There are four association models used in Simple Pairing. Throughout this document, the four association models are referred to as *Numeric Comparison*, *Just Works*, *Out Of Band*, and *Passkey Entry*. Each of these association models are described in more detail in the following sections.

3.1 NUMERIC COMPARISON

The *Numeric Comparison* association model is designed for scenarios where both devices are capable of displaying a six digit number and both are capable of having the user enter “yes” or “no”. A good example of this model is the cell phone / PC scenario.

The user is shown a six digit number (from “000000” to “999999”) on both displays and then asked whether the numbers are the same on both devices. If “yes” is entered on both devices, the pairing is successful.

The numeric comparison serves two purposes. First, since many devices do not have unique names, it provides confirmation to the user that the correct devices are connected with each other. Second, the numeric comparison provides protection against man-in-the-middle attacks (see Section 2.2).

Also note that if this model seems quite a bit like the PIN entry model used by Core Specifications 2.0 + EDR and earlier, there is a significant difference from a cryptographic point of view. In the *Numeric Comparison* association model, the six digit number is an artifact of the security algorithm and not an input to it, as is the case in the current security model. Knowing the displayed number is of no benefit in decrypting the encoded data exchanged between the two devices.

3.2 JUST WORKS

The *Just Works* association model is primarily designed for scenarios where at least one of the devices does not have a display capable of displaying a six digit number nor does it have a keyboard capable of entering six decimal digits. A good example of this model is the cell phone / mono headset scenario where most headsets do not have a display.

The *Just Works* association mode uses the Numeric Comparison protocol but the user is never shown a number and the application may simply ask the user to accept the connection (exact implementation is up to the end product manufacturer).

The *Just Works* association model provides the same protection as the Numeric Comparison association model against passive eavesdropping but offers no protection against man-in-the-middle attacks.

When compared against today’s experience of a headset with a four digit, fixed PIN, the security level of the *Just Works* association model is considerably higher since a high degree of protection against passive eavesdropping is realized.

3.3 OUT OF BAND

The *Out Of Band* (OOB) association model is primarily designed for scenarios where an OOB mechanism is used to both discover the devices as well as to exchange or transfer cryptographic numbers used in the pairing process. In order to be effective from a security point of view, the OOB channel should provide different properties in terms of security compared to the Bluetooth radio channel. The OOB channel should provide resistance to man in the middle attacks and/or privacy.

The user’s experience may differ depending on the out of band mechanism. As an example, with a Near Field Communication (NFC) solution, the user(s) will initially touch the two devices together. The user may then be asked if they

wish to pair with the other device and if “yes” is entered the pairing is successful. This is a single touch experience where the exchanged information is used in both devices. The information exchanged includes discovery information (such as the Bluetooth Device Address) as well as cryptographic information. One of the devices will use the received Bluetooth Device Address to establish a connection with the other device. The rest of the exchanged information is used during authentication.

Depending on the characteristics of the OOB mechanism, either a one-way or two-way authentication is performed.

The OOB protocol is selected only when the pairing process has been activated by previous OOB exchange of information and one (or both) of the device(s) advertises OOB in its IO capabilities response. The protocol uses the information which has been exchanged and simply asks the user to confirm connection.

The *Out of Band* association model supports any OOB mechanism where cryptographic information and Bluetooth Device Address can be exchanged. The *Out of Band* association model does not support a solution where the user has activated a connection using Bluetooth wireless technology and would like to use the OOB channel for authentication during a connection.

3.4 PASSKEY ENTRY

The *Passkey Entry* association model is primarily designed for scenarios where one device has input capability but does not have the capability to display six digits and the other device has output capabilities. A good example of this model is the PC and keyboard scenario.

The user is shown a six digit number (from “000000” to “999999”) on the device with a display. They are then asked to enter the number on the other device. If the value entered on the second device is correct, the pairing is successful.

3.5 ASSOCIATION MODEL OVERVIEW

The following diagram shows Simple Pairing from the point of view of the technology used for discovery and then the different association possibilities.

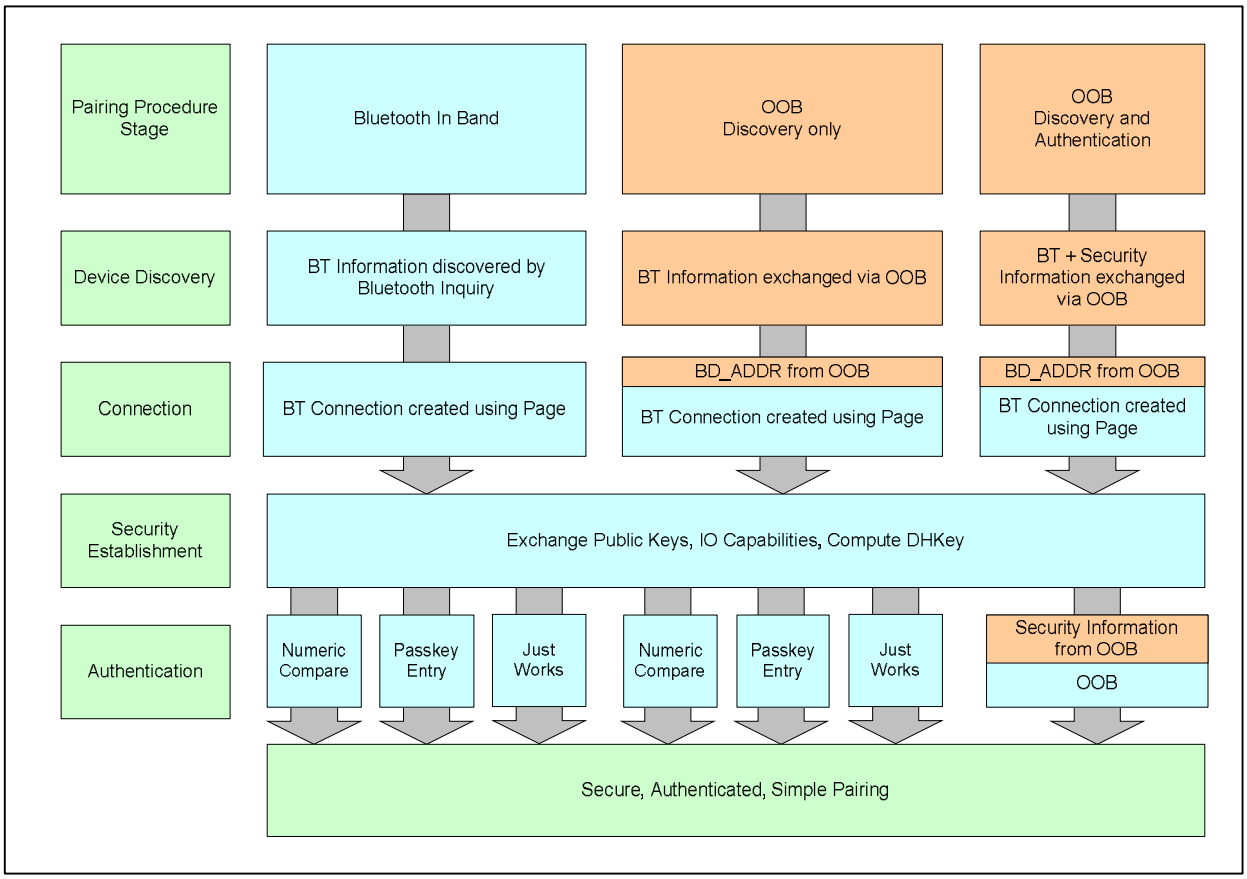

Figure 1 – Simple Pairing Association Models

4 Security Protocol

There are five phases of Simple Pairing:

- Phase 1: Public key exchange
- Phase 2: Authentication Stage 1
- Phase 3: Authentication Stage 2
- Phase 4: Link key calculation
- Phase 5: LMP Authentication and Encryption

Phases 1, 3, 4 and 5 are the same for all protocols whereas phase 2 (Authentication Stage 1) is different depending on the protocol used. Distributed through these five phases are 13 steps.

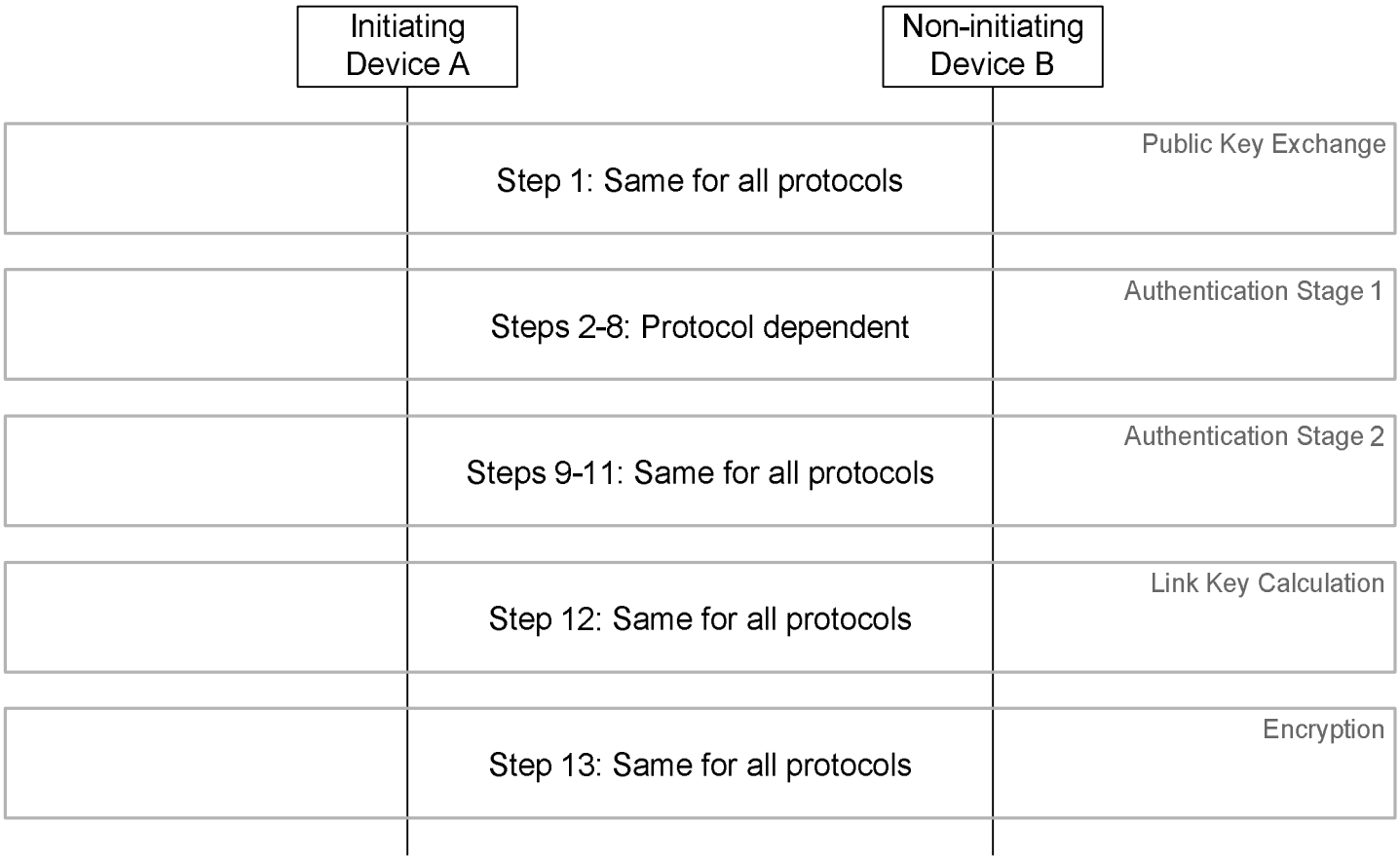

Figure 2- Simple Pairing Security Phases

The terminology used in the security sections is defined in Table 1:

Table 1 – Terminology

Term	Definition
Cx	Commitment value from device X
Cxi	i^{th} commitment value from device X. Only used in the passkey entry protocol
DHKey	Diffie-Hellman key
Ex	Check value from device X
f1( )	Used to generate the 128-bit commitment values Ca and Cb
f2( )	Used to compute the link key and possible other keys from the DHKey and random nonces
f3( )	Used to compute check values Ea and Eb in Authentication Stage 2
g( )	Used to compute numeric check values
IOcapA	IO capabilities of device A
IOcapB	IO capabilities of device B
LK	Link Key
Nx	Nonce (unique random value) from device X
Nxi	i^{th} nonce (unique random value) from device X. Only used in the passkey entry protocol
PKx	Public Key of device X
rx	Random value generated by device X
rxi	Bit i of the random value rx. Only used in the passkey entry protocol
SKx	Secret (Private) Key of device X
Vx	Confirmation value on device X. Only used in the numeric compare protocol.
X	BD_ADDR of device X

4.1 PHASE 1: PUBLIC KEY EXCHANGE

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (step 1). This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.

Pairing is initiated by the initiating device sending its public key to the receiving device (step 1a.). The responding device replies with its own public key (step 1b.) These public keys are not regarded as secret although they may identify the devices. Note that steps 1a and 1b are the same in all three protocols.

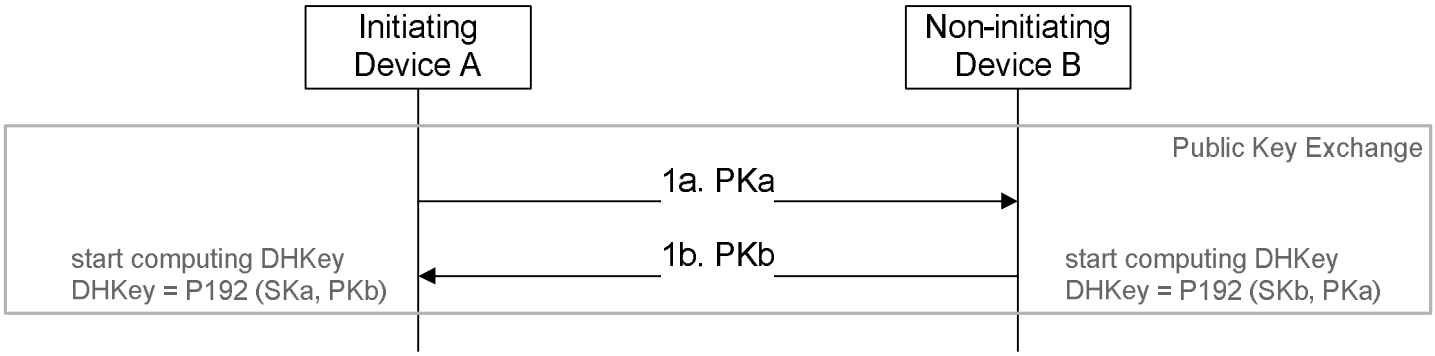

Figure 3 – Public Key Exchange Details

4.2 PHASE 2: AUTHENTICATION STAGE 1

Authentication Stage 1 has three different protocols: Numeric Comparison, Out-of-Band, and Passkey Entry. Note that there is not a Just Works protocol as the *Just Works* association model shares the Numeric Comparison protocol.

The protocol is chosen based on the IO capabilities of the two devices.

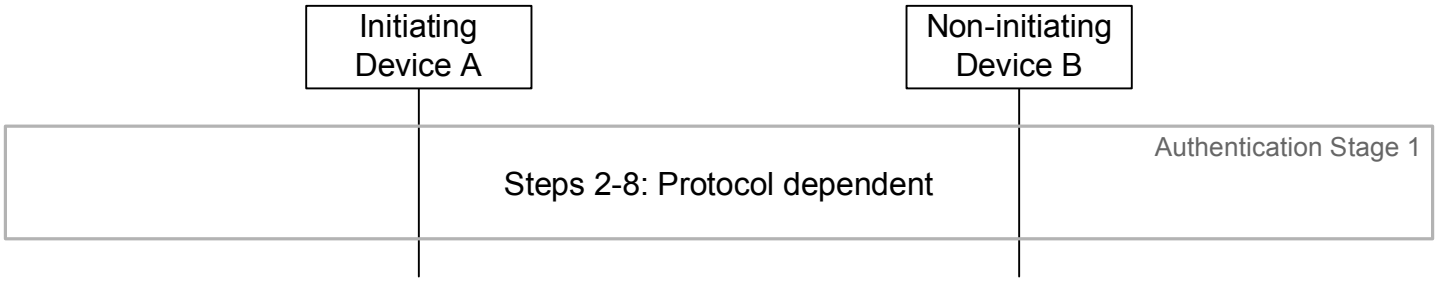

Figure 4 – Authentication Stage 1 (High Level)

The three protocols are described in the following sections.

4.2.1 AUTHENTICATION STAGE 1: NUMERIC COMPARISON PROTOCOL

The Numeric Comparison protocol provides limited protection against active “man-in-the-middle” attacks as an active man-in-the-middle will succeed with a probability of 0.000001 on each iteration of the protocol. Provided that there is no man-in-the-middle at the time the pairing is performed, the shared Link Key that is generated is computationally secure from even a passive eavesdropper that may have been present during the pairing.

The sequence diagram of Authentication Stage 1 for the Numeric Comparison protocol from the cryptographic point of view is shown below:

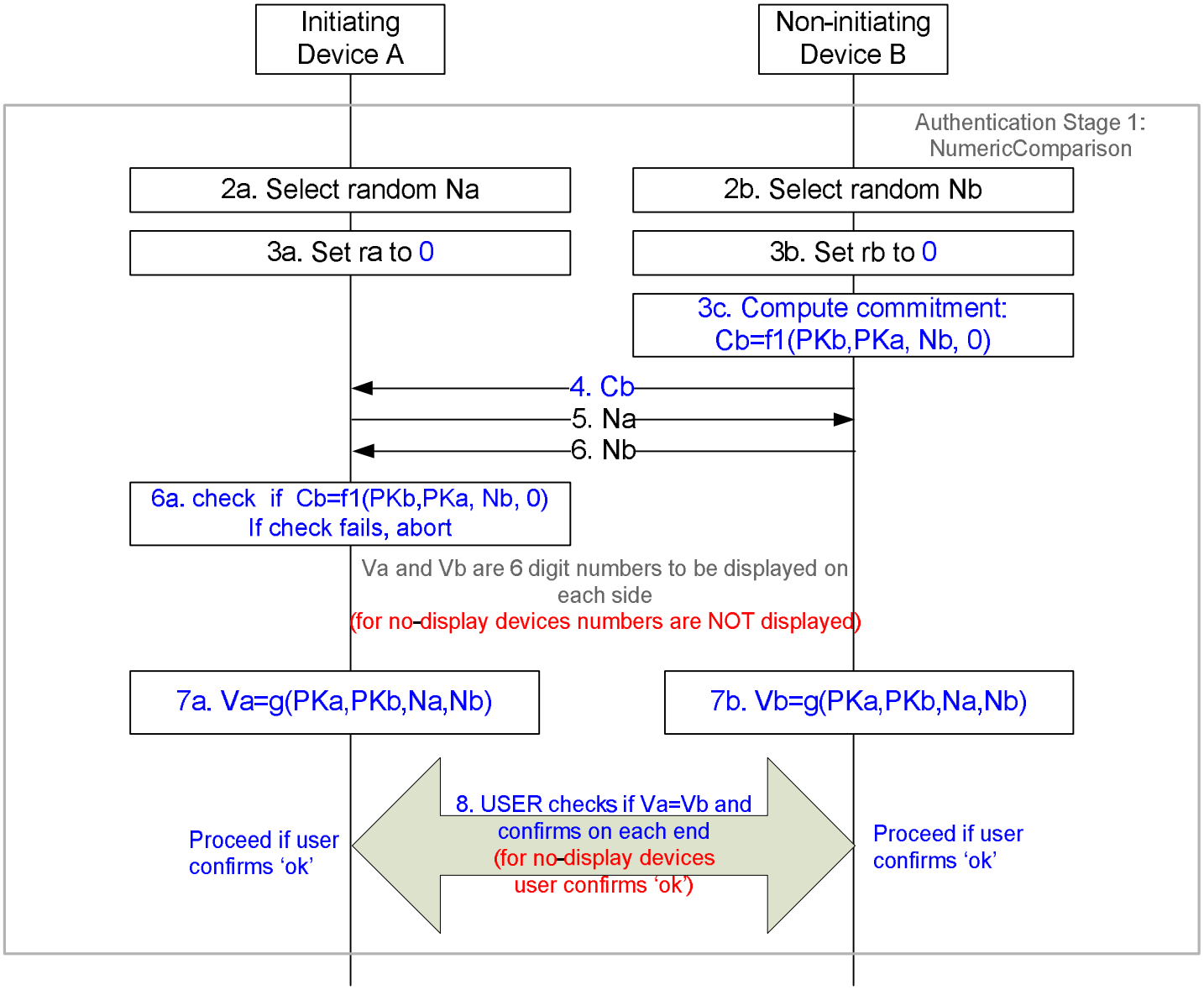

Figure 5 – Authentication Stage 1: Numeric Comparison Protocol Details

After the public keys have been exchanged, each device selects a pseudo-random 128-bit nonce[10] (step 2). This value is used to prevent replay attacks and must be freshly generated with each instantiation of the pairing protocol. This value should be generated directly from a physical source of randomness or with a good pseudo-random generator seeded with a random value from a physical source.

The responding device then computes a commitment to the two public keys that have been exchanged and its own nonce value (step 3c). This commitment is computed as a one-way function of these values and is transmitted to the initiating device (step 4). The commitment prevents an attacker from changing these values at a later time.

The initiating and responding devices then exchange their respective nonce values (steps 5 and 6) and the initiating device confirms the commitment (step 6a). A failure at this point indicates the presence of an attacker or other

[10] Nonce: a random number generated, used only once, and then discarded. See http://en.wikipedia.org/wiki/Nonce

transmission error and causes the protocol to abort. The protocol may be repeated with or without the generation of new public-private key pairs, but new nonces must be generated if the protocol is repeated.

Assuming that the commitment check succeeds, the two devices each compute 6-digit confirmation values that are displayed to the user on their respective devices (steps 7a. 7b, and 8). The user is expected to check that these 6-digit values match and to confirm if there is a match. If there is no match, the protocol aborts and, as before, new nonces must be generated if the protocol is to be repeated.

An active man-in-the-middle must inject its own key material into this process to have any effect other than denial-of-service. A simple man-in-the-middle attack will result in the two 6-digit display values being different with probability 0.999999. A more sophisticated attack may attempt to engineer the display values to match, but this is thwarted by the commitment sequence. If the attacker first exchanges nonces with the responding device, it must commit to the nonce that it will use with the initiating device before it sees the nonce from the initiating device. If the attacker first exchanges nonces with the initiating device, it must send a nonce to the responding device before seeing the nonce from the responding device. In each case, the attacker must commit to at least the second of its nonces before knowing the second nonce from the legitimate devices. It therefore cannot choose its own nonces in such a way as to cause the display values to match.

4.2.2 AUTHENTICATION STAGE 1: OUT OF BAND PROTOCOL

The Out-of-Band protocol is used when authentication information has been received by at least one of the devices and indicated in the OOB Authentication Data Present parameter in the LMP IO capability exchange sequence. The sequence diagram for Authentication Stage 1 for the Out of Band protocol from the cryptographic point of view is shown in the following diagram:

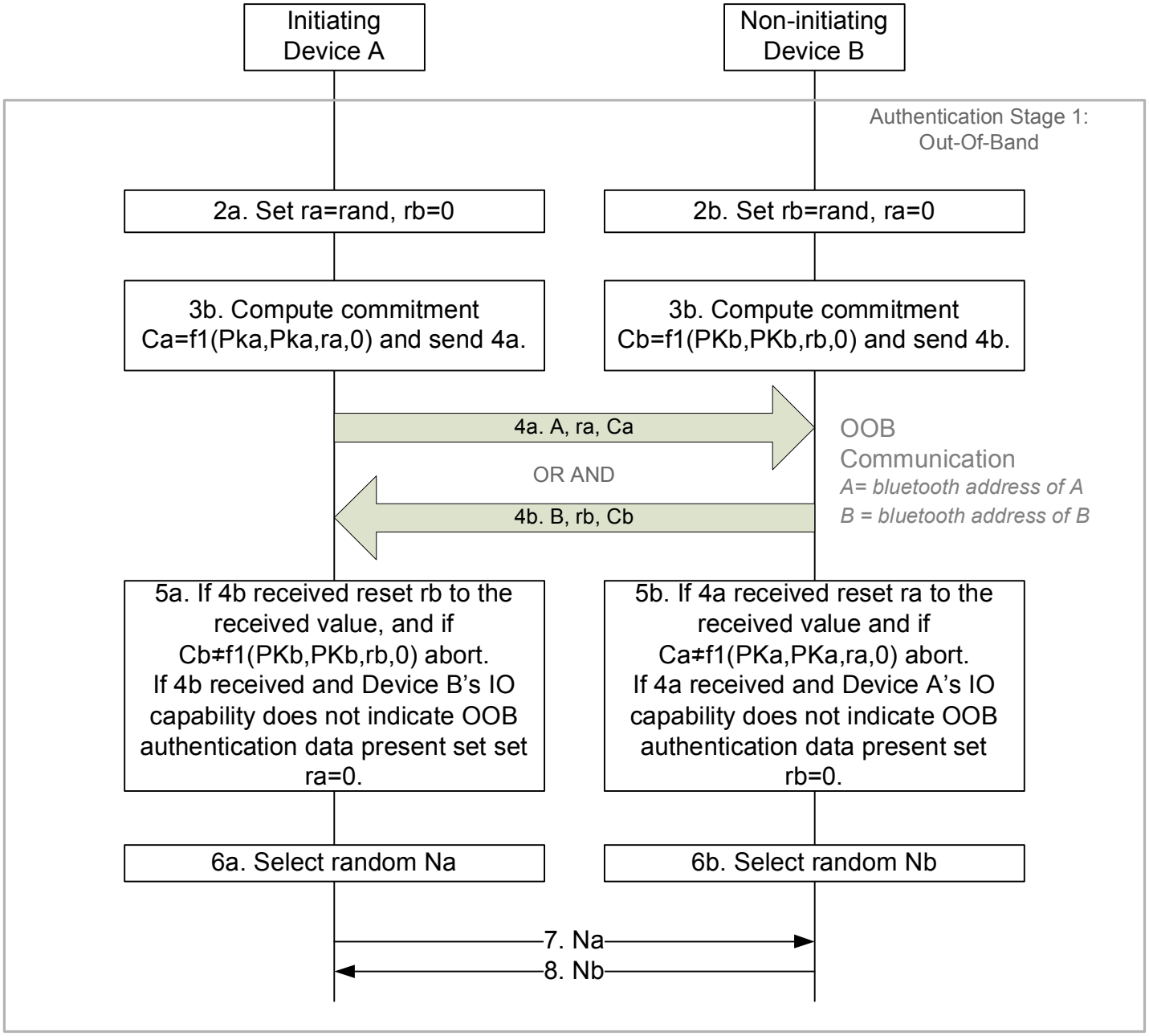

Figure 6 – Authentication Stage 1: Out of Band Details

Principle of operation: If both devices can transmit and/or receive data over an out-of-band channel, then mutual authentication will be based on the commitments of the public keys (Ca and Cb) exchanged OOB in Authentication stage 1. If OOB communication is possible only in one direction (e.g., one device can only be read as is the case of a device equipped with a passive NFC tag), then authentication of the device receiving the OOB communication will be based on that device knowing a random number r sent via OOB. In this case, r must be secret: r can be created afresh every time, or access to the device sending r must be restricted. If r is not sent by a device, it is assumed to be 0 by the device receiving the OOB information in step 4a or 4b.

If the OOB communication cannot be tampered with (i.e. immune to a man-in-the-middle attack), this association method is also not susceptible to MITM attack. Also, in the Out of Band protocol the size of authentication parameters (Ca, Cb and

ra, rb) need not be restricted by what the user can comfortably read or type. For that reason, the Out of Band protocol can be more secure than using the Numeric Comparison or Passkey Entry protocols. However, both devices need to have matching OOB interfaces.

Roles of A and B: The protocol is symmetric with respect to the roles of A and B. It does not require that device A always will initiate pairing and it automatically resolves asymmetry in the OOB communication, e.g. in the case when one of the devices has an NFC tag and can only transmit OOB. However, when computing the link key in step 12 (see Figure 9) both parties must input the parameters in the same order, otherwise each device will compute a different key. That order is as follows: device A's parameters are those of the piconet master and device B's parameters are those of the piconet slave.

Order of steps: The public key exchange must happen before the verification step 5. In the diagram the in-band public key exchange between the devices (step 1) is done before the OOB communication (step 4). But when the pairing is initiated by an OOB interface, public key exchange will happen after the OOB communication (step 1 will be between steps 4 and 5).

Values of ra and rb: Since the direction of the peer's OOB interface cannot be verified before the OOB communication takes place, a device will always generate and if possible transmit through its OOB interface a random number r to the peer. Each device applies the following rules locally to set the values of its own r and the value of the peer's r:

1. Initially, r of the device is set to a random number and r of the peer is set to 0 (step 2).
2. If a device has received OOB, it sets the peer's r value to what was sent by the peer (Step 5).
3. If the remote device does not indicate that it has received OOB authentication data, it sets its own r value to 0 (Step 5)

These rules ensure that when entering Authentication Stage 2, both A and B have the same values for ra and rb if OOB communication took place.

4.2.2.1 NFC AS AN OUT OF BAND MECHANISM

Near Field Communication (NFC)[11] devices may support a variety of different modes of operation including different data rates (106kbps, 212kbps and 424kbps) and different modes of operation (active and passive). In addition, some NFC devices have the ability to initiate (initiator / reader mode) and accept connections (tag / target mode) whereas other devices only have the capability to accept connections.

An OOB-IO NFC device has a mechanism allowing it to communicate with the Bluetooth Controller as well as transmit and receive data from another NFC device. An OOB-O NFC device cannot communicate with the Bluetooth Controller and can only transmit data.

There are three scenarios that are practical for NFC as an out of band mechanism:

Scenario	Device A	Device B
1	OOB-IO NFC device	OOB-O NFC device
2	OOB-O NFC device	OOB-IO NFC device
3	OOB-IO NFC device	OOB-IO NFC device

Since one device must be a reader in each case, the OOB-O / OOB-O (tag / tag) case doesn't exist.

[11] http://en.wikipedia.org/wiki/Near_Field_Communication

4.2.3 AUTHENTICATION STAGE 1: PASSKEY ENTRY PROTOCOL

The sequence diagram for Authentication Stage 1 for the Passkey Entry protocol from the cryptographic point of view is shown below:

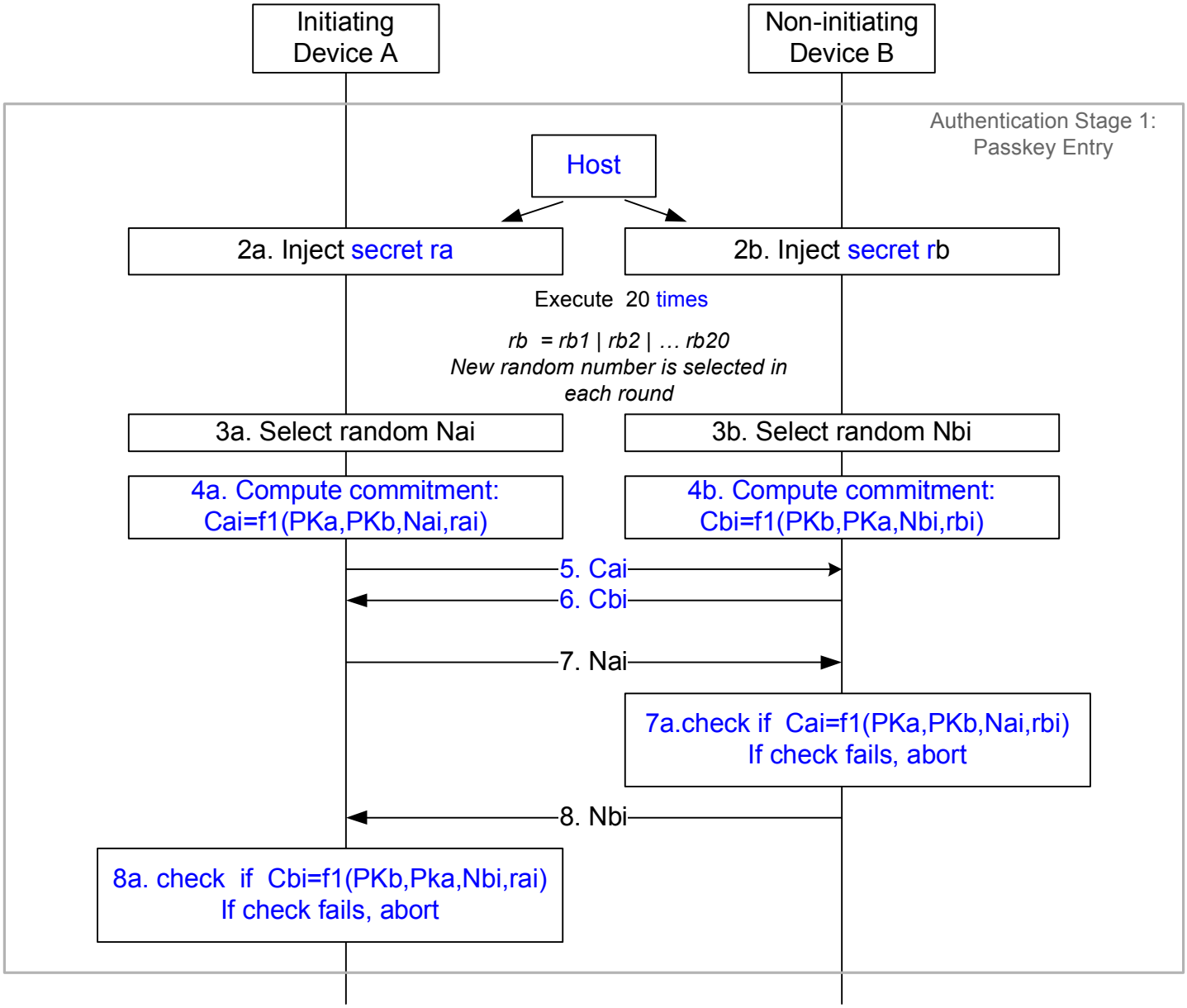

Figure 7 – Authentication Stage 1: Passkey Entry Details

The user inputs an identical Passkey into both devices. Alternately, the Passkey may be generated and displayed on one device, and the user then inputs it into the other (step 2). This short shared key will be the basis of the mutual authentication of the devices. Steps 3 through 8 are repeated k times for a k-bit Passkey --e.g., k=20 for a 6-digit Passkey (999999=0xF423F).

In Steps 3-8, each side commits to each bit of the Passkey, using a long nonce (128 bits), and sending the hash of the nonce, the bit of the Passkey, and both public keys to the other party. The parties then take turns revealing their commitments until the entire Passkey has been mutually disclosed. The first party to reveal a commitment for a given bit of the Passkey effectively reveals that bit of the Passkey in the process, but the other party then has to reveal the

corresponding commitment to show the same bit value for that bit of the Passkey, or else the first party will then abort the protocol, after which no more bits of the Passkey are revealed.

This "gradual disclosure" prevents leakage of more than 1 bit of un-guessed Passkey information in the case of a man-in-the-middle attack. A MITM attacker with only partial knowledge of the Passkey will only receive one incorrectly-guessed bit of the Passkey before the protocol fails. Hence, a MITM attacker who engages first one side, then the other will only gain an advantage of at most two bits over a simple brute-force guesser which succeeds with probability 0.000001.

The long nonce is included in the commitment hash to make it difficult to brute-force even after the protocol has failed. The public Diffie-Hellman values are included to tie the Passkey protocol to the original ECDH key exchange, to prevent a MITM from substituting the attacker's public key on both sides of the ECDH exchange in standard MITM fashion.

At the end of this stage, Na is set to Na20 and Nb is set to Nb20 for use in Authentication Stage 2.

4.3 PHASE 3: AUTHENTICATION STAGE 2

The second stage of authentication then confirms that both devices have successfully completed the exchange. This stage is identical in all three protocols.

Each device computes a new confirmation value that includes the previously exchanged values and the newly derived shared key (step 9). The initiating device then transmits its confirmation value which is checked by the responding device (step 10). If this check fails, it indicates that the initiating device has not confirmed the pairing, and the protocol must be aborted. The responding device then transmits its confirmation value which is checked by the initiating device (step 11). A failure indicates that the responding device has not confirmed the pairing and the protocol should abort.

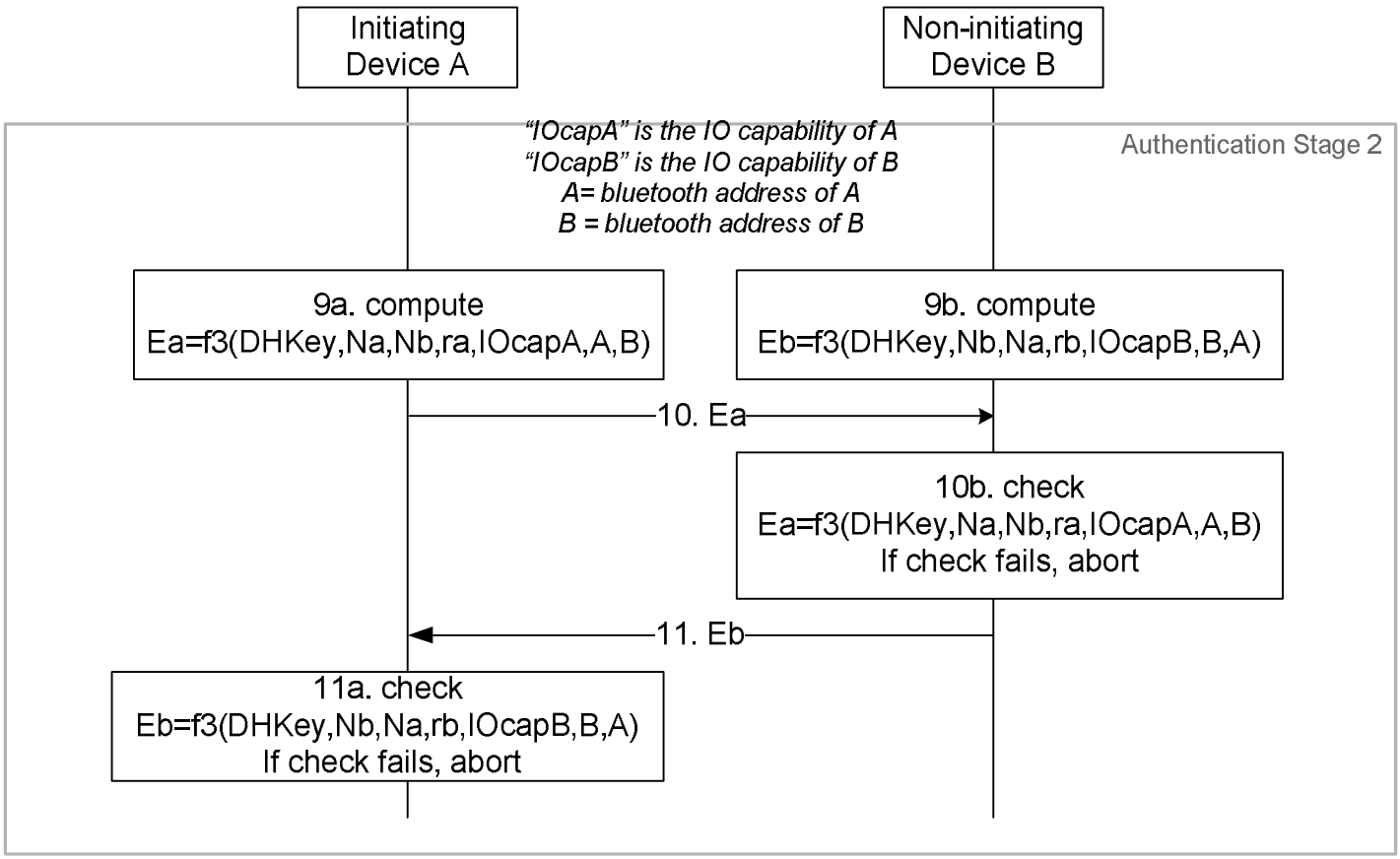

Figure 8 – Authentication Stage 2

4.4 PHASE 4: LINK KEY CALCULATION

Once both sides have confirmed the pairing, a link key is computed from the derived shared key and the publicly exchanged data (step 12). The nonces ensure the freshness of the key even if long-term ECDH values are used by both sides. This link key is used to maintain the pairing.

When computing the link key both parties shall input the parameters in same order to ensure that both devices compute the same key: device A's parameters are those of the piconet master and device B's parameters are those of the piconet slave.

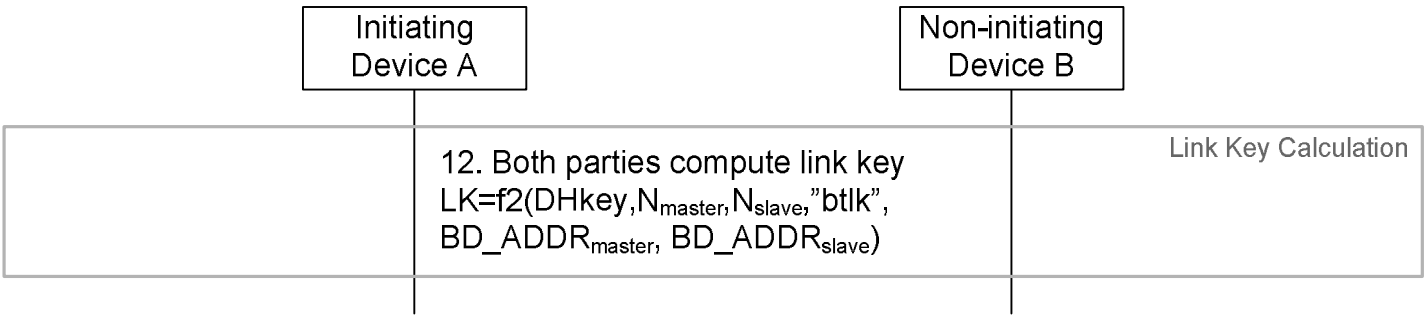

Figure 9 – Link Key Calculation

4.5 PHASE 5: LMP AUTHENTICATION AND ENCRYPTION

The final phase in Simple Pairing is generating the encryption keys. This is the same as the final steps in legacy pairing.

5 Cryptographic Functions

5.1 ELLIPTIC CURVE DEFINITION

Simple Pairing uses the FIPS P-192 curve.[12] Elliptic curves[13] are specified by p, a, b and are of the form

$$E: y^2 = x^3 + ax + b \pmod p$$

For each value of b a unique curve can be developed. In NIST P-192:

a = mod (-3, p)
b is defined and its method of generation can be verified by using SHA-1[14] (with a given seed s and using $b^2c = -27 \pmod p$)

The following parameters are given:

- The prime modulus p, order r, base point x-coordinate Gx, base point y-coordinate Gy
- The integers p and r are given in decimal form; bit strings and field elements are given in hex

  - *p* = `6277101735386680763835789423207666416083908700390324961279`
  - *r* = `6277101735386680763835789423176059013767194773182842284081`
  - *b* = `64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1`
  - *Gx* = `188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012`
  - *Gy* = `07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811`

The function P192( ) is defined as follows. Given an integer u, 0 < u < r, and a point V on the curve E, the value P192(u,V) is computed as the x-coordinate of the u-th multiple uV of the point V.

The private keys shall be between 1 and r/2, where r is the Order of the Abelian Group on the elliptic curve (e.g. between 1 and $2^{192}/2$).

5.2 CRYPTOGRAPHIC FUNCTION DEFINITIONS

In addition to computing the Elliptic Curve Diffie-Hellman key, the Numeric Comparison, Out-of-Band and Passkey Entry protocols require four cryptographic functions. These functions are known as *f1*, *g*, *f2* and *f3*.

f1 is used to generate the 128-bit commitment values Ca and Cb
g is used to compute numeric check values
f2 is used to computer the link key and possible other keys from the DHKey and random nonces
f3 is used to compute check values Ea and Eb in Authentication Stage 2

The basic building block for these functions is SHA-256.[15].

[12] http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf
[13] See http://en.wikipedia.org/wiki/Elliptic_curve for more details
[14] See http://en.wikipedia.org/wiki/SHA-1 for more details
[15] See http://en.wikipedia.org/wiki/SHA-1

5.2.1 THE SIMPLE PAIRING COMMITMENT FUNCTION *F1*

The commitments are computed with function *f1*. The definition of the Simple Pairing commitment function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-256$_X$ with 128-bit key X.

The inputs to the Simple Pairing function *f1* are:

U is 192 bits
V is 192 bits
X is 128 bits
Z is 8 bits

Z is zero (i.e. 8 bits of zeros) for Numeric Comparison and OOB protocol. In the Passkey protocol the most significant bit of Z is set equal to one and the least significant bit is made up from one bit of the passkey e.g. if passkey is '1' then Z = 0x81 and if the passkey is '0' then Z = 0x80.

The output of the Simple Pairing *f1* function is:

$$f1(U, V, X, Z) = \text{HMAC-SHA-256}_X (U \,||\, V \,||\, Z) / 2^{128}$$

The inputs to *f1* are different depending on the different protocols.

Table 2 – Inputs to f1 for the Different Protocols

Numeric Comparison	Out-Of-Band	Passkey Entry
Ca = *f1*(PKax, PKbx, Na, 0) Cb = *f1*(PKbx, PKax, Nb, 0)	Ca = *f1*(PKax, PKax, Ra, 0) Cb = *f1*(PKbx, PKbx, Rb, 0)	Cai = *f1*(PKax, PKbx, Nai, rai) Cbi = *f1*(PKbx, PKax, Nbi, rbi)

where PKax denotes the x-coordinate of the public key PKa of A. Similarly, PKbx denotes the x-coordinate of the public key PKb of B. Nai is the nonce value of ith round. For each round Nai value is a new 128 bit number. Similarly, rai is one bit value of the passkey expanded to 8 bits (e.g. either 0x80 or 0x81).

5.2.2 THE SIMPLE PAIRING NUMERIC VERIFICATION FUNCTION *G*

The Simple Pairing *g* function is defined as follows:

The inputs to the Simple Pairing function *g* are:

U is 192 bits
V is 192 bits
X is 128 bits
Y is 128 bits

The output of the Simple Pairing *g* function is:

$$g(U, V, X, Y) = \text{SHA-256}(U \,||\, V \,||\, X \,||\, Y) \bmod 2^{32}$$

The numeric verification value is taken as six least significant digits of the 32-bit integer *g*(PKax, PKbx, Na, Nb) where PKax denotes the x-coordinate of the public key PKa of A and PKbx denotes the x-coordinate of the public key PKb of B.

Output of SHA-256 is truncated to 32 bits by taking the least significant 32 bits of the output of SHA-256. This value is converted to decimal numeric value. The checksum used for numeric comparison is the least significant six digits.

$$\text{Compare Value} = g(U, V, X, Y) \bmod 10^6$$

E.g. if output = 0x 01 2e b7 2a then decimal value = 19838762 and the checksum used for numeric comparison is 838762.

5.2.3 THE SIMPLE PAIRING KEY DERIVATION FUNCTION *F2*

The definition of the Simple Pairing key derivation function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-256$_W$ with 192-bit key W.

The inputs to the Simple Pairing function *f2* are:

W is 192 bits
N_1 is 128 bits
N_2 is 128 bits
keyID is 32 bits
A_1 is 48 bits
A_2 is 48 bits

The string “btlk” is mapped into keyID using extended ASCII[16] as follows:

keyID[0] = `0110 1011 (lsb)`
keyID[1] = `0110 1100`
keyID[2] = `0111 0100`
keyID[3] = `0110 0010`

`KeyID = 0x62746c6b`

The output of the Simple Pairing *f2* function is:

$f2(W, N_1, N_2, KeyID, A_1, A_2) = HMAC\text{-}SHA\text{-}256_W (N_1 || N_2 || KeyID || A_1 || A_2) / 2^{128}$

The output of *f2* is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256.

The link key is then calculated as:

LK = *f2*(DHKey, N_master, N_slave, “btlk”, BD_ADDR_master, BD_ADDR_slave)

[16] http://en.wikipedia.org/wiki/ISO_8859-1

5.2.4 THE SIMPLE PAIRING CHECK FUNCTION *F3*

The definition of the Simple Pairing *f3* check function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-256$_W$ with 192-bit key W.

The inputs to the Simple Pairing function *f3* are:

W is 192 bits
N_1 is 128 bits
N_2 is 128 bits
R is 128 bits
IOcap is 16 bits
A_1 is 48 bits
A_2 is 48 bits

IOcap is two octets with the most significant octet as the LMP Out-of-Band Authentication Data and the least significant octet as the LMP IO capability.

The output of the Simple Pairing *f3* function is:

$$f3(W, N_1, N_2, R, IOcap, A_1, A_2) = HMAC\text{-}SHA\text{-}256_W (N_1 || N_2 || R || IOcap || A_1 || A_2) / 2^{128}$$

The output of *f3* is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256. The check values are computed with function *f3*. The inputs to *f3* are different depending on the different protocols:

Table 3 – Inputs to f3 for the Different Protocols

Numeric Comparison	Out-Of-Band	Passkey Entry
Ea = *f3*(DHKey, Na, Nb, 0, IOcapA, A, B) Eb = *f3*(DHKey, Nb, Na, 0, IOcapB, B, A)	Ea = *f3*(DHKey, Na, Nb, ra, IOcapA, A, B) Eb = *f3*(DHKey, Nb, Na, rb, IOcapB, B, A)	Ea = *f3*(DHKey, Na20, Nb20, ra, IOcapA, A, B) Eb = *f3*(DHKey, Nb20, Na20, rb, IOcapB, B, A)

DHKey denotes the the shared secret Diffie-Hellman Key computed as P192(SKa,PKb) by A and as P192(SKb,PKa) by B. Adata denotes the capability data of A and Bdata denotes the capability data of B. In Passkey Entry, the data ra and rb are 6-digit passkey values which are expressed as a 128-bit integer. E.g. if the 6-digit value of ra is 131313 then R = 0x 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 f1. The input A is the BD_ADDR of device A and the input B is the BD_ADDR of device B.