Comparison-Based Key Exchange and the Security of the Numeric Comparison Mode in Bluetooth v2.1

Yehuda Lindell*

January 6, 2009

Abstract

In this paper we study key exchange protocols in a model where the key exchange takes place between devices with limited displays that can be compared by a human user. If the devices display the same value then the human user is convinced that the key exchange terminated successfully and securely, and if they do not then the user knows that it came under attack. The main result of this paper is a rigorous proof that the numeric comparison mode for device pairing in Bluetooth version 2.1 is secure, under appropriate assumptions regarding the cryptographic functions used. Our proof is in the standard model and in particular does not model any of the functions as random oracles. In order to prove our main result, we present formal definitions for key exchange in this model and show our definition to be equivalent to a simpler definition. This is a useful result of independent interest that facilitates an easier security analysis of protocols in this model.

1 Introduction

A central problem in cryptography is that of enabling parties to communicate secretly and reliably in the presence of an adversary. This is often achieved by having the parties run a protocol for generating a mutual and secret session key. This session key can then be used for secure communication using known techniques (e.g., applying encryption and message authentication codes to all communication). Two important parameters to define regarding this problem relate to the strength of the adversary and the communication model and/or initial setup for the parties. The problem of session-key generation was initially studied by Diffie and Hellman [8] who considered a passive adversary that can eavesdrop on the communication of the parties, but cannot actively modify messages on the communication line. Thus, the parties are assumed to be connected by reliable, albeit non-private, channels. Many efficient and secure protocols are known for this scenario. In contrast, in this paper, we consider a far more powerful adversary who can modify and delete messages sent between the parties, as well as insert messages of its own choice. It is well known that in the presence of such a powerful adversary, it is impossible for the parties to generate a secret session key if they have no initial secrets and can only communicate over the adversarially controlled channel. This is due to the fact that the adversary can carry out a separate execution with each of the parties, where in each execution it impersonates the other. Since there is no initial secret (like a password or public-key infrastructure), there is nothing that prevents the adversary from succeeding in its impersonation.

*Bar-Ilan University, Israel. Email: `lindell@cs.biu.ac.il`. Most of this work was carried out for Aladdin Knowledge Systems Ltd.

The common solution to the above problem is to indeed introduce a shared secret, like a password, or to assume a public-key infrastructure. However, these solutions are not always possible nor always desired (a user cannot memorize a long private-key and short human-memorizable passwords are notoriously problematic). Another option is therefore to assume that the parties have an additional *authenticated communication channel* that cannot be tampered with by the adversary and can be used to send a short message [10, 15]. There are a number of ways that such a channel can be implemented in reality. In this paper, we consider the case that the parties running the key exchange protocol (or, more accurately, the *devices*) each have a screen upon which they can display a short (say, 6 digit) number. The human user then compares to make sure that both devices display the same number, and if they do, is convinced that the key exchange terminated securely. We remark that although this does not seem to be an authenticated communication channel, it is essentially equivalent to one. This is because one party can send a short message to the other party (using the insecure channel), and then they can both display the message on their screens. If the adversary modifies the message en route, then this will be detected by the human user who will reject the result. Thus, the screens can be used to communicate a single short number from one party to the other (for usability reasons, it is required that only a single value be displayed).

Our results. Our main result is a *rigorous proof of security* of the numeric comparison mode in the simple pairing protocol of Bluetooth version 2.1 [1]. The importance of this result is due to the popularity of Bluetooth, and the unfortunate historic fact that vulnerabilities have often been found in unproven key exchange protocols, sometimes many years after they were released. We stress that our analysis focuses solely on the numeric comparison mode and says nothing about the security of the entire standard (and in particular, nothing about the security regarding the interplay between the different modes and backward compatibility with version 2.0). We prove the security of the protocol in the standard model, by appropriately modeling the functions used in the Bluetooth protocol as standard cryptographic primitives. We stress that we do not model any of the functions as ideal primitives (like random oracles), although this would have made the proof of security much easier.

In order to prove our results, we present a formal definition of comparison-based key exchange that is based on the definitions of key exchange of [3, 4]. Our definition is similar in spirit to that of [15], except that we focus specifically on the problem of key exchange, whereas [15] considered a more general setting of message authentication. As is standard for definitions of security for key exchange protocols, we consider a complex setting where many different protocol instances are run concurrently. Since it is difficult to analyze the security of protocols in complex settings, we present an alternative definition that implies our main definition. The alternative definition is slightly more restrictive but seems to capture the way protocols typically work in this setting. This definition is easier to work with, and to demonstrate this further, we show that it is equivalent to a definition whereby only a single protocol execution takes place. We believe that this alternative definition and its equivalence to the simpler setting is of independent interest as it facilitates significantly easier proofs of security of protocols in this model.

Related work. The problem of secure key exchange has achieved a huge amount of attention, whether it be in the plain model with an eavesdropping adversary, or whether it considers an active adversary and assumes the existence of a full public-key infrastructure, shared high quality secrets or low quality passwords. The comparison-based model that we consider here was first studied in [11, 12, 10], with a more general treatment appearing in [15]. Tight bounds for achieving information-theoretic security in this model were shown in [14]. The MA-DH protocol of [13] has many similarities to the Bluetooth v2.1 numeric comparison protocol analyzed in this paper. Nev-

ertheless, it has significant differences, making it necessary to provide a separate security analysis and proof.

2 Comparison-Based Secure Key-Exchange – Definitions

2.1 Defining Security

Preliminaries. We denote the security parameter by n. A function $f : \mathsf{N} \to [0,1]$ is negligible if for every polynomial $p(\cdot)$ there exists an integer N such that for every $n > N$ it holds that $f(n) < 1/p(n)$. We denote an arbitrary negligible function by negl.

Background. In this section, we adapt the definition of secure key exchange of [3, 4] to our setting. Although the basic ideas are similar, there are a number of fundamental differences between this model and the classic model of key exchange. First and foremost, the parties do not only interact via regular communication channels. In particular, the parties are able to carry out a numeric comparison between two short numbers of length ℓ, and this can be used to prevent the adversary from carrying out a successful man-in-the-middle attack. We formally model the comparison as part of the protocol in the following simple way: each entity participating in a key exchange holds a local public "comparison variable" (the variable is public in the sense that the adversary can read its value whenever it wishes). The comparison variable can be set only once in any instance (i.e., it is $\mathsf{write\text{-}once}$ only); this rules out protocols that use multiple comparisons (arguably, such protocols have more limited use in practice). Another fundamental difference between this setting and the classic model of key exchange is that it is *not* enough for the adversary to learn the secret key that one of the parties obtains at the end of a protocol execution (it can always succeed in doing this by just interacting with the party). Rather, the adversary only succeeds if it manages to learn the secret key that a pair of parties obtain in an execution in which the *parties' comparison variables are equal*. A third difference is that there is no public-key infrastructure or secret setup information and thus all instances of the protocol are identical. This is in contrast to the shared secret setting where each pair of parties hold a shared secret key, and every protocol instance run by a party is initialized with the party's secret key. Despite this, the protocol is supposed to be secure in the presence of an active adversary, and not just an eavesdropping one.

We remark that in our setting here, it makes no sense to allow a single party to run many instances of the protocol concurrently. This is because each party has only one interface for displaying the comparison variable, and so more than one execution cannot be run at the same time. In addition, since there is no shared setup between different executions, allowing more than one execution would be equivalent in any case (when there is no shared setup, a number of executions by a single party is equivalent to a number of parties running a single execution each). Of course, the different parties running different executions may be running concurrently. We could additionally allow each party to run many executions sequentially, but this clearly makes no difference and thus for simplicity we just assume that each party runs one execution.

The definition. A protocol for secure key exchange assumes that there is a set of *principals* which are the parties (clients, servers or others) who will engage in the protocol. We denote by Π_i the instance of the protocol that is run by user P_i (recall that in contrast to [3, 4] each party runs one execution only). The adversary is given oracle access to these instances and may also control some of the instances itself. We remark that unlike the standard notion of an "oracle", in this model instances maintain state which is updated as the protocol progresses. In addition to information regarding the protocol execution, the state of an instance Π_i includes the following variables (initialized as *null*):

- sid_i: the *session identifier* of this particular instance;
- comp_i: the aforementioned write-once *comparison variable* of the instance; we denote the length of comp_i by ℓ;
- pid_i: the *partner identifier* which is the name of the principal P_j with whom P_i's comparison variable is compared (we note that pid_i can never equal i); in our setting here, it is always the case that if $\mathsf{pid}_i = j$ then $\mathsf{pid}_j = i$ because the human comparing the variables will always work in this way;[1]
- acc_i: a boolean variable set to true or false denoting whether Π_i accepts or rejects at the end of the execution.

Partnering. We say that two instances Π_i and Π_j are *partnered* if the following properties hold: (1) $\mathsf{pid}_i = j$ (and thus by our requirement $\mathsf{pid}_j = i$); and (2) $\mathsf{sid}_i = \mathsf{sid}_j \neq null$. The notion of partnering is important for defining security, as we will see.

The adversary model. The adversary is given total control of the external network (i.e., the network connecting clients to servers). In particular we assume that the adversary has the ability to not only listen to the messages exchanged by players, but also to interject messages of its choice and modify or delete messages sent by the parties.[2] The above-described adversarial power is modeled by giving the adversary oracle access to the instances of the protocol that are run by the principals. Notice that this means that the parties actually only communicate through the adversary. The oracles provided to the adversary are as follows:

- $\mathsf{Execute}(i, j)$: When this oracle is called, pid_i is set to j and pid_j is set to i, and then a complete protocol execution between instances Π_i and Π_j is run. The oracle-output is the protocol transcript (i.e., the complete series of messages exchanged by the instances throughout the execution). These oracle calls reflect the adversary's ability to passively eavesdrop on protocol executions. As we shall see, the adversary should learn nothing from such oracle calls. If an Init call has already been made including i or j, then $\mathsf{Execute}(i, j)$ is ignored.
- $\mathsf{Init}(i, j)$: This call initializes $\mathsf{pid}_i = j$ and $\mathsf{pid}_j = i$. If pid_i or pid_j is already set, then this call does nothing. In addition, it returns the first message that Π_i sends to Π_j in a protocol execution.
- $\mathsf{Send}(i, M)$: This call sends the message M to the instance Π_i. The output of the oracle is whatever message the instance Π_i would send after receiving the message M (given its current state). This oracle allows the adversary to carry out an active man-in-the-middle attack on the protocol executions.
- $\mathsf{Reveal}(i)$: This call outputs the secret key sk_i that instance Π_i outputs at the end of the protocol execution. This oracle allows the adversary to learn session keys from previous and concurrent executions, modeling improper exposure of past session keys and ensuring independence of different session keys in different executions.

[1] This is in contrast to the standard setting of key exchange where P_1 may think that it's interacting with P_2 who in turn thinks that it's interacting with P_3.

[2] In principle, the adversary should also be given control over a subset of the oracles, modeling the case of an "inside attacker". However, in our setting, there are no initial secrets and thus this makes no difference.

- $\mathsf{Test}(i)$: This call is needed for the definition of security and does not model any real adversarial ability. The adversary is only allowed to query it once, and the output is either the private session key of Π_i, denoted sk_i, or a random key sk that is chosen independently of the protocol executions (each case happens with probability 1/2). The adversary's aim is to distinguish these two cases. We let b^i_{test} denote the bit chosen by $\mathsf{Test}(i)$ to determine whether to output sk_i or a random sk.

The security of key exchange protocols is composed of three components: non-triviality, correctness and privacy. We begin by stating the non-triviality requirement (this is different from the definition in [3, 4] because we also require that $\mathsf{comp}_i = \mathsf{comp}_j$ so that a human user will accept the result):

Non-triviality: If two instances Π_i and Π_j that hold each other's partner identifier communicate without adversarial interference (as in an $\mathsf{Execute}$ call), then Π_i and Π_j are partnered, $\mathsf{comp}_i = \mathsf{comp}_j$ and they both accept.

Correctness: If two *partnered* instances Π_i and Π_j accept (i.e., $\mathsf{acc}_i = \mathsf{acc}_j = 1$) and $\mathsf{comp}_i = \mathsf{comp}_j$, then they must both conclude with the same session key (i.e., $sk_i = sk_j$).

Privacy: We now define what it means for a protocol to be private. Intuitively, a protocol achieves privacy if the adversary cannot distinguish real session keys from random ones. (This then implies that the parties can use their generated session keys in order to establish secure channels; see [5] for more discussion on this issue.) Of course, the adversary can always correctly guess the bit in a $\mathsf{Test}(i)$ query if it queried $\mathsf{Reveal}(i)$ or $\mathsf{Reveal}(j)$ when Π_i and Π_j are partnered. Therefore, $\mathcal{A}$ is only said to have succeeded if these oracles were not queried. In addition, we are only interested in the case that $\mathcal{A}$ correctly guesses the key when $\mathsf{comp}_i = \mathsf{comp}_j$ and both instances accept. This is due to the fact that if $\mathsf{comp}_i \neq \mathsf{comp}_j$ then the human user will not accept the result, and if one of the instances does not accept then no session-key will be output by that instance. This yields the following definition of adversarial success. Formally, we say that an adversary $\mathcal{A}$ $\mathsf{succeeds}$ if the following conditions are *all* fulfilled:

1. $\mathcal{A}$ outputs b^i_{test}
2. $\mathsf{comp}_i = \mathsf{comp}_j$ and $\mathsf{acc}_i = \mathsf{acc}_j = \mathsf{true}$
3. If Π_i and Π_j are partnered then $\mathcal{A}$ did not query $\mathsf{Reveal}(i)$ or $\mathsf{Reveal}(j)$.

Now, the adversary's $\mathsf{advantage}$ is formally defined by:

$$\mathsf{Adv}(\mathcal{A}) = \left|2 \cdot \mathrm{Prob}[\mathcal{A}\ \mathsf{succeeds}\ ] - 1\right|.$$

We reiterate that an adversary is only considered to have succeeded if it correctly guesses the bit used by the $\mathsf{Test}(i)$ oracle, $\mathsf{comp}_i = \mathsf{comp}_j$, and the adversary did *not* query $\mathsf{Reveal}(i)$ or $\mathsf{Reveal}(j)$ when Π_i and Π_j are partnered. We stress that if $\mathsf{pid}_i = j$ but $\mathsf{sid}_i \neq \mathsf{sid}_j$, then Π_i and Π_j are not partnered and thus the adversary succeeds if $\mathsf{comp}_i = \mathsf{comp}_j$ and it correctly guesses the bit used by the $\mathsf{Test}(i)$ oracle, even if it queried $\mathsf{Reveal}(j)$.

An important observation here is that when there is no initial setup and only a short comparison channel of length ℓ is used, the adversary can gain an advantage of $2^{-\ell}$ for every pair of instances by just running two separate executions with two instances and hoping that their comparison variables will end up being equal. A protocol is therefore called private if it is limited to random success of this fashion. Notice that in $\mathsf{Execute}$ oracle calls, the adversary is passive and thus it should only have a negligible advantage in guessing the secret key of such an instance, irrespective of the value

of ℓ. We do not explicitly require this, but rather provide it with a $2^{-\ell}$ advantage only when it queries the Send oracle. This is reminiscent of the definition for password-based key exchange of [2]. In order to define this, we define Q_{send} to be the number of protocol instances without common partner identifiers to which the adversary made Send oracle queries. We stress that if $\mathcal{A}$ makes multiple Send queries to Π_i and to Π_j, and $\mathsf{pid}_i = j$, then this is counted as 1 in Q_{send}. Formally, a protocol is said to be $\mathsf{private}$ if the advantage of the adversary is at most negligibly more than $Q_{\text{send}}/2^{\ell}$, where ℓ is the length of the comparison variable. In summary,

Definition 2.1 (comparison-based key exchange): *A comparison-based key exchange protocol with a comparison variable of length $\ell \in \mathsf{N}$ is said to be* secure *if for every probabilistic polynomial-time adversary $\mathcal{A}$ that makes at most Q_{send} queries of type* Send *to different protocol instances without common partner identifiers, there exists a negligible function* negl *such that*

$$\mathsf{Adv}(\mathcal{A}) < \frac{Q_{\text{send}}}{2^{\ell}} + \mathsf{negl}(n).$$

Furthermore, the probability that the non-triviality or correctness requirement is violated is at most negligible in the security parameter n.

We note that the bound of $Q_{\text{send}}/2^{\ell}$ for $\mathcal{A}$'s advantage is optimal. Specifically, one can construct an adversary $\mathcal{A}$ who obtains this exact advantage by separately interacting with two protocol instances Π_i and Π_j for which $\mathsf{pid}_i = j$ and $\mathsf{pid}_j = i$. At the end of the execution, $\mathcal{A}$ will know both sk_i and sk_j and will succeed if $\mathsf{comp}_i = \mathsf{comp}_j$. If this does not hold, then $\mathcal{A}$ can just invoke an $\mathsf{Execute}$ oracle call for two other instances, query the Test oracle for one of those instances, and then just randomly guess the test result, succeeding with probability one half. The advantage of this adversary $\mathcal{A}$ is as follows. First, under the assumption that an honest protocol execution yields a uniformly distributed comparison variable, we have that $\mathsf{comp}_i = \mathsf{comp}_j$ with probability exactly $2^{-\ell}$. In this case, $\mathcal{A}$ succeeds with probability 1. Noting further that if $\mathsf{comp}_i \neq \mathsf{comp}_j$ then $\mathcal{A}$ succeeds with probability 1/2, we have:

$$\begin{aligned}
\Pr[\mathcal{A}\ \mathsf{succeeds}] &= \Pr[\mathcal{A}\ \mathsf{succeeds} \mid \mathsf{comp}_i = \mathsf{comp}_j] \cdot \Pr[\mathsf{comp}_i = \mathsf{comp}_j] \\
&\quad + \Pr[\mathcal{A}\ \mathsf{succeeds} \mid \mathsf{comp}_i \neq \mathsf{comp}_j] \cdot \Pr[\mathsf{comp}_i \neq \mathsf{comp}_j] \\
&= 1 \cdot \frac{1}{2^{\ell}} + \frac{1}{2} \cdot \left(1 - \frac{1}{2^{\ell}}\right) \\
&= \frac{1}{2^{\ell}} + \frac{1}{2} - \frac{1}{2^{\ell+1}} = \frac{1}{2} + \frac{1}{2^{\ell+1}}
\end{aligned}$$

implying that $\mathcal{A}$'s advantage is $1/2^{\ell}$. Noting finally that $Q_{\text{send}} = 1$ in this case, we have that $\mathcal{A}$ achieves the upper bound of $Q_{\text{send}}/2^{\ell}$ on the advantage as stated in Definition 2.1. The above argument holds for any value of Q_{send} (and not just the special case that $Q_{\text{send}} = 1$). In this case, $\mathcal{A}$ interacts separately with Q_{send} pairs and succeeds if for any of the pairs it holds that $\mathsf{comp}_i = \mathsf{comp}_j$ (or with probability 1/2 otherwise, as above). Since the probability that $\mathsf{comp}_i = \mathsf{comp}_j$ in at least one of the executions is $Q_{\text{send}}/2^{\ell}$ we have that $\mathcal{A}$ succeeds with probability $Q_{\text{send}}/2^{\ell} + \frac{1}{2} \cdot (1 - Q_{\text{send}}/2^{\ell})$. As above, this results in an advantage of $Q_{\text{send}}/2^{\ell}$, as required.

An alternative definition. In Section 2.2 below we present an alternative definition that is easier to work with. We prove that security under the alternative definition implies security under Definition 2.1 and thus it suffices to use the alternative definition. In addition, we prove that when considering the alternative definition, security in the concurrent setting with many protocols instances is equivalent to security in a one-time setting where an adversary interacts once with a protocol instance P_1 and once with a protocol instance P_2 (and where $\mathsf{pid}_1 = 2$ and $\mathsf{pid}_2 = 1$).

2.2 A Stronger Definition

We now present an alternative definition that is more restrictive, but as we will see, is useful for proving the security of protocols. Informally speaking, the alternative definition states that an adversary can succeed in one of two ways: either by guessing the key for an instance Π_i that is partnered with some other instance Π_j, *or* by somehow causing two accepting instances Π_i and Π_j that are not partnered to have the same comparison value, meaning that $\mathsf{comp}_i = \mathsf{comp}_j$. Recall that our original definition required the adversary to guess the key in the case that $\mathsf{comp}_i = \mathsf{comp}_j$ (and we did not care if Π_i and Π_j were partnered except to exclude the case where a Reveal query was made), whereas here the adversary succeeds immediately if $\mathsf{comp}_i = \mathsf{comp}_j$ without the need to learn the key. Another difference from the original definition is that we separately consider the adversary's success in each case. That is, we require that the adversary can succeed in guessing the key of partnered instances with probability at most $1/2 + \mathsf{negl}(n)$, and separately require that the adversary can succeed in having two unpartnered instances conclude with the same comparison value with probability at most $Q_{\text{send}}/2^\ell + \mathsf{negl}(n)$.

Formally, we define two events referring to the adversary's success:

1. An adversary $\mathcal{A}$ succeeds in a $\mathsf{guess\ attack}$, denoted $\mathsf{succ}_{\mathcal{A}}^{\text{guess}}$, if it outputs the correct b_{test}^i bit after querying $\mathsf{Test}(i)$ for an instance Π_i that is partnered with some other instance Π_j, and $\mathsf{Reveal}(i)$ or $\mathsf{Reveal}(j)$ were not queried.

2. An adversary $\mathcal{A}$ succeeds in a $\mathsf{comparison\ attack}$, denoted $\mathsf{succ}_{\mathcal{A}}^{\text{comp}}$, if there exist two accepting instances Π_i and Π_j with $\mathsf{pid}_i = j$ and $\mathsf{pid}_j = i$ that are not partnered and yet $\mathsf{comp}_i = \mathsf{comp}_j$.

We are now ready to define security:

Definition 2.2 (comparison-based key exchange – stronger definition): *A comparison-based key exchange protocol with a comparison variable of length $\ell \in \mathsf{N}$ is said to be* secure *if for every probabilistic polynomial-time adversary $\mathcal{A}$ that makes at most Q_{send} queries of type* Send *to different protocol instances without common partner identifiers,*

$$\Pr[\mathsf{succ}_{\mathcal{A}}^{\text{guess}}] < \frac{1}{2} + \mathsf{negl}(n) \qquad \text{and} \qquad \Pr[\mathsf{succ}_{\mathcal{A}}^{\text{comp}}] < \frac{Q_{\text{send}}}{2^\ell} + \mathsf{negl}(n).$$

Furthermore, the probability that the non-triviality or correctness requirement is violated is at most negligible in the security parameter n.

Observe that under the assumption that independent executions are partnered with only negligible probability, the strategy described after Definition 2.1 that achieves an advantage of $Q_{\text{send}}/2^\ell$ also achieves that same advantage under the stronger definition. Therefore, as above, the bound is optimal. (Note that if independent executions may be partnered with non-negligible probability, then the protocol cannot be secure by Definition 2.2 because an adversary just needs to run independent executions with Π_i and Π_j and then if they turn out to be partnered it knows the secret key with certainty and so it succeeds in a guess attack. This is in contrast to Definition 2.1 where such a strategy would not be considered successful unless it also holds that $\mathsf{comp}_i = \mathsf{comp}_j$.)

Before proceeding, we show that Definition 2.2 is no weaker than Definition 2.1. This demonstrates that it suffices to prove security under Definition 2.2. We have not succeeded in finding an example proving that Definition 2.2 is strictly stronger (i.e., that there exist protocols that are secure according to Definition 2.1 and not according to Definition 2.2), although we conjecture that this is the case. Nevertheless, as we will see below, Definition 2.2 is easier to work with, and it seems to cover the way natural protocols are designed in this model.

Claim 2.3 *Let* Π *be a protocol that is secure according to Definition* 2.2*. Then,* Π *is also secure according to Definition* 2.1*.*

Proof: Let $\mathcal{A}$ be an adversary. We begin by showing the connection between $\mathcal{A}$'s success by Definition 2.1 and the events $\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}}$ and $\mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$. Consider the event that $\mathcal{A}$ succeeded according to Definition 2.1. This means that $\mathcal{A}$ guessed b_{test}^{i} correctly and $\mathsf{comp}_i = \mathsf{comp}_j$, and if Π_i and Π_j are partnered then $\mathcal{A}$ did not query $\mathsf{Reveal}(i)$ or $\mathsf{Reveal}(j)$. Consider two cases:

1. *Case 1 –* Π_i *and* Π_j *are partnered:* in this case, the event $\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}}$ occurred;
2. *Case 2 –* Π_i *and* Π_j *are not partnered:* in this case, the event $\mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$ occurred (comp_i must equal comp_j and they must both accept because otherwise $\mathcal{A}$ would not have succeeded by Definition 2.1).

It therefore follows that if $\mathcal{A}$ succeeds by Definition 2.1, then at least one of the events $\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}}$ and $\mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$ must have occurred. That is, for every adversary $\mathcal{A}$,

$$\Pr[\mathcal{A}\ \mathsf{succeeds}] \leq \Pr[\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}} \vee \mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}] \tag{1}$$

Now, let $\mathcal{A}$ be any adversary. We construct an adversary $\mathcal{A}'$ that internally runs $\mathcal{A}$ and behaves in exactly the same way. The only difference is that if $\mathcal{A}'$ observes that $\mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$ has occurred, then it outputs a random bit b_{test} (instead of what $\mathcal{A}$ outputs); in contrast, if $\mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$ has not occurred then $\mathcal{A}'$ outputs the bit b_{test} that $\mathcal{A}$ outputs. Observe that for $\mathcal{A}'$ it holds that

$$\Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}} \wedge \mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}] = \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}} \mid \mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}] \cdot \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}] = \frac{1}{2} \cdot \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}]$$

because whenever $\mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}$ occurs, the probability of $\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}}$ occurring is exactly $1/2$ ($\mathcal{A}'$ outputs a random b_{test} in this case). Furthermore, if the event ($\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}} \vee \mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$) occurs then event ($\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}} \vee \mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}$) also occurs; in order to see this note that $\mathcal{A}'$ only behaves differently to $\mathcal{A}$ if $\mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$ already occurred. Thus, $\Pr[\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}} \vee \mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}] \leq \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}} \vee \mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}]$.

Combining the above with Eq. (1) we have:

$$\begin{aligned}
\Pr[\mathcal{A}\ \mathsf{succeeds}] &\leq \Pr[\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}} \vee \mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}] \\
&\leq \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}} \vee \mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}] \\
&= \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}}] + \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}] - \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}} \wedge \mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}] \\
&= \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}}] + \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}] - \frac{1}{2} \cdot \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}] \\
&= \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{guess}}] + \frac{1}{2} \cdot \Pr[\mathsf{succ}_{\mathcal{A}'}^{\mathrm{comp}}] \\
&\leq \frac{1}{2} + \mathsf{negl}(n) + \frac{1}{2}\left(\frac{Q_{\mathrm{send}}}{2^{\ell}} + \mathsf{negl}(n)\right) \\
&= \frac{1}{2} + \frac{Q_{\mathrm{send}}}{2^{\ell+1}} + \mathsf{negl}'(n)
\end{aligned}$$

where the second last inequality is by the assumption that Π is secure according to Definition 2.2. We conclude that $\Pr[\mathcal{A}\ \mathsf{succeeds}] \leq 1/2 + Q_{\mathrm{send}}/2^{\ell+1} + \mathsf{negl}(n)$ and so $\mathsf{Adv}(\mathcal{A}) \leq Q_{\mathrm{send}}/2^{\ell} + \mathsf{negl}(n)$ proving that Π is secure according to Definition 2.1, as required. ■

2.3 Single versus Multiple Executions

Definitions 2.1 and 2.2 are cast in a setting where many different protocol instances take place concurrently. This models the real-world scenario, and as such the definition takes this complex setting into account. However, this also makes it difficult to analyze security. It is therefore desirable to work in a more restricted setting where the adversary interacts *once* with two protocol instances Π_1 and Π_2 where $\mathsf{pid}_1 = 2$, using the Send and Test oracles (i.e., the Reveal and Execute oracles are removed). Needless to say, analyzing the security of protocols in such a restricted setting is much simpler. In this section, we show that the restricted and general settings are *equivalent* when considering Definition 2.2, under the assumption that two independent executions of the protocol conclude with the same comparison value with probability that is negligibly close to $2^{-\ell}$. (We do not know whether equivalence between the general and restricted settings also holds for Definition 2.1 but did not succeed in proving it.) Before continuing, we remark that since there are only two protocol instances, one of each party, we just denote them P_1 and P_2 (by the parties playing them) and not as "instances" (which are needed when many executions involving P_1 are run).

We remark that security in this setting is defined exactly as before, except that here Q_{send} is always fixed to 1. Despite this, for the sake of clarity, we repeat the definition.

Definition 2.4 (restricted setting): *A comparison-based key exchange protocol with a comparison variable of length $\ell \in \mathsf{N}$ is said to be* secure in the one-time setting *if for every probabilistic polynomial-time adversary $\mathcal{A}$ interacting with P_1 and P_2 where $\mathsf{pid}_1 = 2$ and $\mathsf{pid}_2 = 1$, it holds that*

$$\Pr[\mathsf{succ}_{\mathcal{A}}^{\text{guess}}] < \frac{1}{2} + \mathsf{negl}(n) \qquad \text{and} \qquad \Pr[\mathsf{succ}_{\mathcal{A}}^{\text{comp}}] < \frac{1}{2^{\ell}} + \mathsf{negl}(n).$$

Furthermore, the probability that the non-triviality or correctness requirement is violated is at most negligible in the security parameter n.

Intuitively, the one-time and multiple-execution settings are equivalent because there is no secret state that is used by the parties over different executions. For example, they have no long-term secret password, or public key infrastructure. This means that an attacker who succeeds in the multiple-execution setting can be simulated in the one-time setting by just internally simulating all of the executions except for one. We stress that such a strategy cannot be carried out in the password setting because it is not possible to simulate all of the executions of a given party (except one) without knowing its password. We now formally prove the equivalence:

Theorem 2.5 *Let Π be a two-party protocol for comparison-based key exchange such that two independent executions of Π conclude with the same comparison value with probability that is negligibly close to $2^{-\ell}$. Then, Π is secure as in Definition 2.2 if and only if it is secure in the one-time setting as in Definition 2.4.*

Proof: Non-triviality is immediate because the distribution over honestly-run executions is the same in the one-time and multiple execution settings. Thus, if non-triviality is violated in one setting with non-negligible probability it will also be violated in the other setting with non-negligible probability. Correctness is also not difficult to show. This is due to the fact that it must hold, except with negligible probability. Thus an adversary in a single execution setting can internally simulate multiple executions while running just one. The probability that correctness is violated on the single execution actually being run is a polynomial fraction of the probability in the multiple

execution setting. Thus, if the probability is negligible in the one-time setting, it must also be negligible in the multiple execution setting. (The formal details of how the internal executions are run can be derived from the more involved proof of security below.)

We now prove the equivalence with respect to security. It is immediate that if Π is secure under Definition 2.2 then it is secure under Definition 2.4 because in this case $Q_{\text{send}} \leq 1$. The more interesting direction states that security under Definition 2.4 implies security under Definition 2.2. Let Π be secure under Definition 2.4. We separately prove that for every $\mathcal{A}$, $\Pr[\mathsf{succ}_{\mathcal{A}}^{\text{guess}}] < \frac{1}{2} + \mathsf{negl}(n)$ and $\Pr[\mathsf{succ}_{\mathcal{A}}^{\text{comp}}] < Q_{\text{send}}/2^{\ell} + \mathsf{negl}(n)$ in the general setting.

We begin by proving that $\Pr[\mathsf{succ}_{\mathcal{A}}^{\text{guess}}] < \frac{1}{2} + \mathsf{negl}(n)$. Assume by contradiction that there exists an adversary $\mathcal{A}$ and a non-negligible function $\epsilon = \epsilon(n)$ such that $\Pr[\mathsf{succ}_{\mathcal{A}}^{\text{guess}}] = \frac{1}{2} + \epsilon$ in the general setting. We construct an adversary $\mathcal{A}'$ for the one-time setting that works as follows. Let t be an upper-bound on the number of protocol instances that $\mathcal{A}$ invokes in its attacks (we can always take t to be the running-time of $\mathcal{A}$). Then, adversary $\mathcal{A}'$ chooses a random $i \in \{1, \ldots, t\}$ and internally simulates for $\mathcal{A}$ all of the protocol instance pairs – answering all the Execute, Send and Reveal queries – except for the ith pair; this ith pair is run externally using P_1 and P_2 that $\mathcal{A}'$ interacts with (the ith pair is defined by the ith Init oracle call for which neither pid is already set). If $\mathcal{A}$ queries its Test oracle on the ith pair, then $\mathcal{A}'$ outputs the guess bit b_{test} that $\mathcal{A}$ outputs; otherwise $\mathcal{A}$ outputs a random $b_{\text{test}} \in_R \{0, 1\}$ for its guess. We now analyze the probability that the event $\mathsf{succ}_{\mathcal{A}'}^{\text{guess}}$ occurs in this one-time setting. We use the fact that $\mathcal{A}$ queries its Test oracle on the ith pair with probability exactly $1/t$:

$$\begin{aligned}
\Pr[\mathsf{succ}_{\mathcal{A}'}^{\text{guess}}] &= \frac{1}{t} \cdot \Pr[\mathsf{succ}_{\mathcal{A}}^{\text{guess}}] + \left(1 - \frac{1}{t}\right) \cdot \frac{1}{2} \\
&= \frac{1}{t} \cdot \left(\frac{1}{2} + \epsilon\right) + \left(1 - \frac{1}{t}\right) \cdot \frac{1}{2} \\
&= \frac{1}{2t} + \frac{\epsilon}{t} + \frac{1}{2} - \frac{1}{2t} \\
&= \frac{1}{2} + \frac{\epsilon}{t}.
\end{aligned}$$

Noting that t is a polynomial and thus if ϵ is non-negligible then so is ϵ/t, we have that $\mathcal{A}'$ succeeds in a guessing attack in the one-time setting, in contradiction to the security of Π by Definition 2.4. This proves that no $\mathcal{A}$ can succeed in a guessing attack in the general setting with probability that is non-negligibly greater than $1/2$.

We now proceed to prove that for every $\mathcal{A}$, $\Pr[\mathsf{succ}_{\mathcal{A}}^{\text{comp}}] < Q_{\text{send}}/2^{\ell} + \mathsf{negl}(n)$ in the general setting. Assume by contradiction that there exists an adversary $\mathcal{A}$ sending Q_{send} queries to the Send oracle and a non-negligible function $\epsilon = \epsilon(n)$ such that $\Pr[\mathsf{succ}_{\mathcal{A}}^{\text{comp}}] = Q_{\text{send}}/2^{\ell} + \epsilon$. Without loss of generality, we can assume that $\mathcal{A}$ always asks exactly Q_{send} queries to its Send oracle. We now use $\mathcal{A}$ to construct an adversary $\mathcal{A}'$ that contradicts the assumed security of Π in the one-time setting. The adversary $\mathcal{A}'$ works as follows:

1. $\mathcal{A}'$ chooses a uniformly distributed index $i \in_R \{1, \ldots, Q_{\text{send}}\}$

2. $\mathcal{A}'$ invokes $\mathcal{A}$ internally and internally simulates all the protocol instances, except for the ith pair invoked by $\mathcal{A}$). The ith pair of instances is run externally with P_1 and P_2 that $\mathcal{A}'$ interacts with. We remark that all of the Execute, Test, Send and Reveal queries for pairs other than the ith pair are all answered by $\mathcal{A}'$ internally. In contrast, the Test and Send and queries for the ith pair are forwarded by $\mathcal{A}'$ externally to its own oracles. $\mathcal{A}'$ halts after the ith pair concludes the execution; $\mathcal{A}'$ does not output anything because its aim is to succeed

in a comparison attack, not a guess attack. (If $\mathcal{A}$ makes a Reveal query on the ith pair then this is only meaningful after the ith pair has concluded. It therefore doesn't matter.)

We now analyze the probability that $\mathsf{succ}^{\mathrm{comp}}_{\mathcal{A}'}$ occurs. It is clear that the simulation by $\mathcal{A}'$ of the general setting for $\mathcal{A}$ is perfect. Furthermore, the probability that $\mathcal{A}$ succeeds in its comparison attack on the ith pair equals exactly $1/Q_{\mathrm{send}}$ times the probability that $\mathcal{A}$ succeeds in its comparison attack in general (this is because i is uniformly distributed). Thus, we conclude that

$$\Pr[\mathsf{succ}^{\mathrm{comp}}_{\mathcal{A}'}] = \frac{1}{Q_{\mathrm{send}}} \cdot \Pr[\mathsf{succ}^{\mathrm{comp}}_{\mathcal{A}}] = \frac{1}{Q_{\mathrm{send}}} \cdot \left(\frac{Q_{\mathrm{send}}}{2^{\ell}} + \epsilon \right) = \frac{1}{2^{\ell}} + \frac{\epsilon}{Q_{\mathrm{send}}}.$$

Now, if ϵ is non-negligible then so is $\epsilon' = \epsilon/Q_{\mathrm{send}}$. Therefore, $\mathcal{A}'$ succeeds in a comparison attack in the one-time setting with probability non-negligibly greater than $1/2^{\ell}$ in contradiction to the security of Π in the one-time setting. ■

Discussion – Definition 2.2 versus Definition 2.1. As we have seen, Definition 2.2 has a significant advantage in that it suffices to prove security in the one-time setting, as formulated in Definition 2.4. We argue that it has another advantage in that it is easier to provide a separate analysis of the two potential attacks than a single combined one. Thus, as we will see when we analyze protocols below, the introduction of Definition 2.2 and its proof of equivalence to Definition 2.4 constitute a significant contribution that greatly simplifies the typically difficult task of proving the security of key exchange protocols.

3 Bluetooth Pairing in Numeric Comparison Mode

In this section, we describe the Bluetooth pairing protocol in the numeric comparison mode. We also describe the cryptographic functions that are used by the protocol, and state the assumptions that are needed regarding each one in order to prove the security of the protocol. The Bluetooth specification refers to devices A and B; in order to be consistent with our definitional notations, we refer to parties P_1 and P_2 instead.

3.1 Cryptographic Tools and Functions

The numeric comparison mode in the Bluetooth simple pairing protocol uses the following tools:

- An Elliptic curve group in which it is assumed that the Decisional Diffie-Hellman problem is hard. We denote the group by $\mathcal{G}$, the generator by g, and the group order by q.
- A non-interactive computationally binding and *non-malleable* commitment scheme C. We denote a commitment to a string x using coins r by $C(x;r)$. The computational binding property means that it is infeasible for any polynomial-time adversary $\mathcal{A}$ to find x, r, x', r' where $x \neq x'$ but $C(x;r) = C(x';r')$. Informally speaking, non-malleability [9] means that given a commitment $c = C(x;r)$ it is infeasible for a polynomial-time adversary to generate a commitment c' so that later given (x,r) it can produce (x',r') such that $c' = C(x';r')$ and x, x' are related via a predetermined polynomial-time computable relation; this is typically called non-malleability with respect to opening [7]. Our formal definition can be found in Appendix A and is adapted from the definition in [6] with some minor changes.

  The commitment scheme is instantiated as follows: in order to commit to a string x, r^a where r^a is uniformly distributed and half of the length of the key for HMAC-SHA256, choose a

random string r^b which is also half of the length of the key for HMAC-SHA256, compute HMAC-SHA256$_r(x)$ where $r = (r^a, r^b)$, and set the commitment value to be the 128 most significant bits of the result. We remark that it may appear to be more natural to use randomness that is the entire length of the HMAC key and then let x be the entire string that is committed to. Indeed, this would have been more natural. However, in the Bluetooth protocol, part of r must be considered to remain secret and we therefore take it to be part of the value being committed to. We remark that the computational binding of this commitment scheme follows directly from the assumption that it is hard to find a collision in SHA256. The assumption on non-malleability is less studied, but seems reasonable given the chaotic behavior of such functions. We remark also that it follows trivially from any random-oracle type assumption.

- A function $g : \{0,1\}^* \to \{0,1\}^\ell$ with the property that when any long-enough part of the input is uniformly distributed, then the output is close to uniform. We formalize this by allowing an adversary to choose two values α and β and then asking what the probability is that $g(\alpha, r) = \beta$ when $r \in_R \{0,1\}^{n/2}$ is uniformly distributed. We call this *computational 2-universal hashing*. In order to be consistent with the exact use of g in the protocol, we first introduce the following notation: For an arbitrary string α, we denote by $\alpha[r]$ the string derived by combining α and r in a predetermined way. (In our use, $\alpha[r]$ will either be the concatenation of r after α, or it involves parsing α into α_1 and α_2 where $|\alpha_2| = n/2$ and then setting $\alpha[r] = (\alpha_1, r, \alpha_2)$.) Then:

  Definition 3.1 *A function $g : \{0,1\}^* \to \{0,1\}^\ell$ is a* computational 2-universal hash function *if for every probabilistic polynomial-time machine $\mathcal{A}$ there exists a negligible function* negl *such that*

  $$\Pr_{(\alpha,\beta)\leftarrow\mathcal{A}(1^n);r\leftarrow\{0,1\}^{n/2}}[g(\alpha[r]) = \beta] < \frac{1}{2^\ell} + \mathsf{negl}(n)$$

  We stress that $r \in_R \{0,1\}^{n/2}$ is *uniformly distributed* and thus chosen independently of α and β output by $\mathcal{A}$. The function g in Bluetooth is defined by $g(x) = \text{SHA256}(x) \bmod 2^{32}$. It seems very reasonable to assume that SHA256 fulfills this property.

- A pseudorandom function F keyed with keys output from Diffie-Hellman key exchange over Elliptic curve groups. This is implemented using HMAC-SHA256 and taking the 128 most significant bits. Formally, we say that a function F is pseudorandom when keyed with $\mathcal{G}$ if it is a pseudorandom function when the key is a random element of $\mathcal{G}$. It is easy to show that if F is pseudorandom when keyed with $\mathcal{G}$ and the Decisional Diffie-Hellman (DDH) assumption holds in $\mathcal{G}$, then it is pseudorandom when keyed with the result of a Diffie-Hellman key exchange. This follows directly from DDH which states that the result of a Diffie-Hellman key exchange is indistinguishable from a random element in $\mathcal{G}$. For simplicity, we state this directly in the Definition below:

  Definition 3.2 *Let* $\mathsf{gen}(1^n)$ *be an algorithm that outputs the description of a group $\mathcal{G}$, its generator g, and its order q. A function ensemble $F = \{F_k\}$ is* pseudorandom when DDH-keyed with gen *if for every probabilistic polynomial-time distinguisher D there exists a negligible function* negl *such that*

  $$\left|\Pr\left[D^{F_{g^{ab}}}(1^n, g^a, g^b) = 1\right] - \Pr\left[D^H(1^n, g^a, g^b) = 1\right]\right| < \mathsf{negl}(n)$$

  where $(\mathcal{G}, g, q) \leftarrow \mathsf{gen}(1^n)$, a, b are randomly chosen in $\{1, \ldots, q\}$, and H is a truly random function ensemble.

As we have mentioned, any function ensemble that is pseudorandom when keyed with a random element from $\mathcal{G}$ is also pseudorandom when DDH-keyed with gen, under the assumption that the DDH assumption holds relative to gen. Note that a "standard" pseudorandom function receives a uniformly distributed bit string. Therefore, this does not necessarily suffice (a random element of $\mathcal{G}$ is not necessarily a uniformly distributed bit string).

3.2 The Protocol and Correctness

The Bluetooth simple pairing protocol in numeric comparison mode appears in Figure 1 and is denoted Π. It is easy to see that Protocol Π is non-trivial. The proof that Π fulfills correctness is also not difficult and appears after the protocol description.

Protocol Π

- **Pre-protocol exchange:** Parties P_1 and P_2 exchange party identifiers 1 and 2 (in Bluetooth, these are their respective Bluetooth addresses) as well as additional auxiliary information α_1 and α_2 (we will ignore the content of this information here). P_1 sets $\mathsf{pid}_1 = 2$, and P_2 sets $\mathsf{pid}_2 = 1$.
- **Phase 1 – Public-Key Exchange:**
  1. The initiating party P_1 generates a Diffie-Hellman value by choosing a random $a \in \{1, \ldots, q\}$ and computing $pk_1 = g^a$. P_1 sends pk_1 to P_2.
  2. Upon receiving pk_1 from P_1, party P_2 chooses a random $b \in \{1, \ldots, q\}$, computes $pk_2 = g^b$, and sends pk_2 to P_1.
  3. Party P_1 sets $\mathsf{sid}_1 = (pk_1, pk_2)$ and party P_2 sets $\mathsf{sid}_2 = (pk_1, pk_2)$.
- **Phase 2 – Authentication Stage 1:**
  1. P_2 chooses a random string $r_2 \in_R \{0,1\}^n$ and sends $c_2 = C(pk_2, pk_1, 0; r_2)$ to P_1.
  2. P_1 chooses a random string $r_1 \in_R \{0,1\}^n$ and sends r_1 to P_2.
  3. P_2 sends r_2 to P_1. Upon receiving r_2, party P_1 checks that $c_2 = C(pk_2, pk_1, 0; r_2)$, where c_2 is the value it received above and pk_1, pk_2 are as exchanged in phase 1.
  4. P_1 sets $\mathsf{comp}_1 = g(pk_1, pk_2, r_1, r_2)$ and P_2 sets $\mathsf{comp}_2 = g(pk_1, pk_2, r_1, r_2)$.
- **Phase 3 – Authentication Stage 2:**
  1. P_1 computes $k = (pk_2)^a$ and P_2 computes $k = (pk_1)^b$, where the computation is in the group $\mathcal{G}$.
  2. P_1 computes $e_1 = F_k(r_1, r_2, 0, \alpha_1, 1, 2)$ and sends e_1 to P_2. (Note that 1 and 2 here, and below, are the parties identifiers and not constants. Thus, they are actually the parties' Bluetooth addresses and more.)
  3. P_2 checks that $e_1 = F_k(r_1, r_2, 0, \alpha_2, 1, 2)$; if yes it sends P_1 the value $e_2 = F_k(r_1, r_2, 0, \alpha_1, 2, 1)$ and sets $\mathsf{acc}_2 = \mathsf{true}$; otherwise P_2 sets $\mathsf{acc}_2 = \mathsf{false}$ and aborts.
  4. P_1 checks that $e_2 = F_k(r_1, r_2, 0, \alpha_1, 2, 1)$; if yes it sets $\mathsf{acc}_1 = \mathsf{true}$, and if not it sets $\mathsf{acc}_1 = \mathsf{false}$ and aborts.
- **Phase 4 – Link-Key Calculation:**
  1. Party P_1 outputs $sk_1 = F_k(r_1, r_2, \beta, 2, 1)$ where β is a fixed string.
  2. Party P_2 outputs $sk_2 = F_k(r_1, r_2, \beta, 2, 1)$.

Figure 1: Bluetooth 2.1 Pairing – Numeric Comparison Mode

Claim 3.3 *Assume that F is a pseudorandom function when DDH-keyed with gen, as in Definition 3.2. Then, protocol Π meets the correctness requirement, except with negligible probability.*

Proof: We need to show that if P_1 and P_2 are partnered, they both accept and $\mathsf{comp}_i = \mathsf{comp}_j$, then $sk_1 = sk_2$ except with negligible probability. Now, if P_1 and P_2 are partnered, then this implies that $\mathsf{sid}_1 = \mathsf{sid}_2$ and so they both received each other's correct Diffie-Hellman values pk_1 and pk_2. Now, P_1 and P_2 accept if and only if the e_1 and e_2 values of phase 3 pass the checks. If both parties hold the same r_1 and r_2 values then they clearly accept, and also output the same session key $sk_1 = sk_2$ (this is due to the fact that the only other values involved in the computation of sk_1 and sk_2 are constants). Thus, correctness holds unless the following event occurs: parties P_1 and P_2 hold different values r_1 and r_2, and yet they both accept. (This is a problem because $sk_1 \neq sk_2$ with overwhelming probability and yet the parties are still partnered and accept.) Denote by r_1, r_2' the values held by P_1 and by r_1', r_2 the values held by P_2 (recall that P_1 chose r_1 but received r_2; we denote the fact that P_1 may not have received the same r_2 as that sent by P_2 by writing r_2'). It suffices to show that

$$\Pr[(r_1, r_2') \neq (r_1', r_2) \ \wedge \ F_k(r_1, r_2', 0, \alpha_2, 1, 2) = F_k(r_1', r_2, 0, \alpha_2, 1, 2)] < \mathsf{negl}(n)$$

However, this follows immediately from the assumption that F is a pseudorandom function when DDH-keyed with gen (equality would only occur with a truly random function with negligible probability). Formally, assume that there exists an adversary $\mathcal{A}$ that causes correctness to be violated with non-negligible probability. This implies that

$$\Pr[(r_1, r_2') \neq (r_1', r_2) \ \wedge \ F_k(r_1, r_2', 0, \alpha_2, 1, 2) = F_k(r_1', r_2, 0, \alpha_2, 1, 2)] = \epsilon(n)$$

for some non-negligible function ϵ. We construct a distinguisher D that works as follows (recall that by Definition 3.2, D receives g^a, g^b for input and is given oracle access to either $F_{g^{ab}}$ or a random function). Upon input g^a, g^b, D sets $pk_1 = g^a$ and $pk_2 = g^b$, invokes $\mathcal{A}$ and simulates an execution of Protocol Π in the one-time setting, running P_1 and P_2. When $\mathcal{A}$ sends a Send query to party P_1 causing it to reply with its message in the public-key exchange phase then D replies with pk_1. Likewise, when $\mathcal{A}$ sends an analogous query to P_2, D replies with pk_2. In contrast to the public-key exchange phase, D runs the authentication stage 1 phase as honest P_1 and P_2 would (note that the secret values a and b are not needed in this phase). Then, when D reaches phase 3 and it has to compute the e_1 and e_2 values, it uses its oracle. If $(r_1, r_2') \neq (r_1', r_2)$ then D computes e_1 and e_2 using its oracle and outputs 1 if and only if they are equal. If $(r_1, r_2') = (r_1', r_2)$ then D outputs a random bit. Now, when D's oracle is $F_{g^{ab}}$, the simulation for $\mathcal{A}$ by D is identical to a real execution. It therefore follows that:

$$\begin{aligned}
\Pr[D^{F_{g^{ab}}}(g^a, g^b) = 1] &= \Pr[D^{F_{g^{ab}}}(g^a, g^b) = 1 \ \wedge \ (r_1, r_2') \neq (r_1', r_2)] \\
&\quad + \Pr[D^{F_{g^{ab}}}(g^a, g^b) = 1 \ \wedge \ (r_1, r_2') = (r_1', r_2)] \\
&= \epsilon(n) + \frac{1}{2}
\end{aligned}$$

In contrast,

$$\begin{aligned}
\Pr[D^{H}(pk_1, pk_2) = 1] &= \Pr[D^{H}(pk_1, pk_2) = 1 \ \wedge \ (r_1, r_2') \neq (r_1', r_2)] \\
&\quad + \Pr[D^{F_k}(pk_1, pk_2) = 1 \ \wedge \ (r_1, r_2') = (r_1', r_2)] \\
&= \mathsf{negl}(n) + \frac{1}{2}
\end{aligned}$$

because a truly random function collides on two different values with only negligible probability. Therefore,

$$\left| \Pr[D^{F_k}(pk_1, pk_2) = 1] - \Pr[D^{H}(pk_1, pk_2) = 1] \right| = \epsilon(n) - \mathsf{negl}(n)$$

which is non-negligible, in contradiction to the assumption on the pseudorandomness of F when keyed with a Diffie-Hellman key. ■

4 The Proof of Security

We now prove that Π is a secure comparison-based key-exchange protocol. The structure of our proof demonstrates the usefulness of Definition 2.4 as a tool; a proof of security that works directly with Definition 2.1 would be much more complex.

Theorem 4.1 *Assume that the Decisional Diffie-Hellman assumption holds relative to* gen*, that* F *is a pseudorandom function when DDH-keyed with* gen*, that* C *is a computationally-binding non-malleable commitment scheme and that* g *is a computational 2-universal hash function. Then, Protocol* Π *is a secure comparison-based key-exchange protocol.*

Proof: We prove the security of Π in two stages. First, we prove that $\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}}$ occurs with probability at most negligibly greater than 1/2. Intuitively, this holds because if P_1 and P_2 are partnered, then this implies that they both have the same Diffie-Hellman values and so essentially have completed a Diffie-Hellman key exchange *undisturbed*, with the adversary only eavesdropping. This in turn implies that F_k is a pseudorandom function and thus the session keys that are output are pseudorandom. We then proceed to prove that $\mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$ occurs with probability at most negligibly greater than $2^{-\ell}$. This follows from the security of the commitment scheme C and the 2-universality of g. Specifically, phase 2 of the protocol can be viewed as a method of choosing two random strings r_1 and r_2 that are (computationally) independent of each other. In order for this to hold even if $\mathcal{A}$ carries out a man-in-the-middle attack, the commitment scheme C must be non-malleable (see Appendix A). This forces $\mathcal{A}$ to either just copy the commitment sent by P_2 or modify it, in which case it will contain an independent r_2 value. If $\mathcal{A}$ copies the commitment, then it will contain the parties' public keys. However, by the assumption that they are not partnered, these keys do not match with those that the parties received in the protocol. $\mathcal{A}$ must therefore modify the commitment, resulting in r_1 and r_2 being independent of each other. Once this is given, it is possible to apply the 2-universality of g stating that whichever is chosen last causes the comparison value to be almost uniformly distributed. We proceed now to the formal proof.

As stated, we prove the protocol using Definition 2.4, and apply Theorem 2.5 and Claim 2.3 to derive that the protocol is secure also under Definition 2.1. Note that in order to apply Theorem 2.5 we have to show that two independent executions of Π conclude with the same comparison value with probability that is negligibly close to $2^{-\ell}$. However, this follows immediately from the assumption that g is computationally 2-universal (and the fact that the r_1 and r_2 values are independently chosen in the two different executions).

We begin by proving that for every probabilistic polynomial-time $\mathcal{A}$ interacting with P_1 and P_2, it holds that

$$\Pr[\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}}] < \frac{1}{2} + \mathsf{negl}(n)$$

Recall that $\mathsf{succ}_{\mathcal{A}}^{\mathrm{guess}}$ occurs if $\mathcal{A}$ outputs the correct b_{test} value after querying $\mathsf{Test}(1)$ or $\mathsf{Test}(2)$ *and* P_1 is partnered with P_2. Now, by the protocol description, the session identifiers are defined to be (pk_1, pk_2) and thus if $\mathsf{sid}_1 = \mathsf{sid}_2$ it follows that P_1 and P_2 hold the same Diffie-Hellman values. Intuitively, this means that if $\mathcal{A}$ can guess the correct b_{test} value with non-negligible probability, then it can solve the DDH problem in $\mathcal{G}$ with non-negligible advantage. The formal reduction follows. Let $\mathcal{A}$ be a probabilistic polynomial-time adversary and let ϵ be a function such that

$\Pr[\mathsf{succ}^{\text{guess}}_{\mathcal{A}}] = \frac{1}{2} + \epsilon(n)$. We show that ϵ must be negligible by presenting a distinguisher D that solves the DDH problem in $\mathcal{G}$ with advantage ϵ. Distinguisher D receives (g^a, g^b, k) and attempts to determine if $k = g^{ab}$ or if $k \in_R \mathcal{G}$. D invokes $\mathcal{A}$ and when it sends a Send oracle query to which P_1 is supposed to reply with its public-key exchange message, then D replies with $pk_1 = g^a$. Likewise, when $\mathcal{A}$ sends an analogous message for P_2 then D replies with $pk_2 = g^b$. If $\mathcal{A}$ does not forward the same pk_1, pk_2 messages unmodified (and so P_1 and P_2 are not partnered) then D outputs a random bit and halts. Otherwise, it proceeds. (Note that D may need to proceed with the simulation before knowing if they are partnered. In this case, it assumes that they will be, and if it turns out to be incorrect it immediately outputs a random bit and halts.) Now, from this step on, D acts exactly like the honest P_1 and P_2 would. In particular, when D reaches the authentication stage 2 of the protocol, it uses the value k that it received in its input to compute e_1 and e_2. Likewise, it uses k to compute sk_1 and sk_2. Now, when $\mathcal{A}$ queries $\mathsf{Test}(1)$ or $\mathsf{Test}(2)$, D chooses a random $b \in_R \{0,1\}$ and replies with sk_1 (or sk_2 respectively) if $b = 0$ and with a random value $\tilde{s}k \in_R \{0,1\}^{|sk_1|}$ otherwise. Finally, D outputs 1 if and only if $\mathcal{A}$ outputs $b_{\text{test}} = b$.

If $k = g^{ab}$ then the simulation above by D is exactly what $\mathcal{A}$ would see in a real protocol execution. Therefore,

$$\Pr[D(g^a, g^b, g^{ab}) = 1] = \frac{1}{2} + \epsilon(n)$$

In contrast, when k is a random value, the simulation by D is "wrong". In particular, the e_1, e_2, sk_1, sk_2 values are computed using a random key k independent of pk_1, pk_2, instead of using g^{ab}. We would like to claim that $\mathcal{A}$ outputs $b_{\text{test}} = b$ with probability $1/2$ in this case, but this may not be true because k has been used to compute e_1, e_2 which are seen by $\mathcal{A}$. Thus, if $\mathcal{A}$ was not computationally bounded it could determine $b_{\text{test}} = b$. Nevertheless, we prove that if F is indeed a pseudorandom function, then $\mathcal{A}$ can output $b_{\text{test}} = b$ with probability at most $1/2 + \mathsf{negl}(n)$. Let δ be a function such that $\mathcal{A}$ outputs $b_{\text{test}} = b$ in this case with probability $1/2 + \delta(n)$. We first prove that δ is a negligible function. Specifically, we construct a distinguisher D_F who receives an oracle that is either the pseudorandom function F_k or a truly random function. D_F invokes $\mathcal{A}$ and works in the same way as D with the following differences. First, D_F generates random pk_1 and pk_2 values itself and uses them. Second, it computes e_1, e_2, sk_1, sk_2 using its *function oracle*. If D_F is given a random function oracle, then sk_1, sk_2 are completely random and independent of everything that $\mathcal{A}$ has seen so far. Thus, information-theoretically, $\mathcal{A}$ outputs $b_{\text{test}} = b$ with probability exactly $1/2$. In contrast, if D_F is given F_k as an oracle, then it generates exactly the same distribution as D when $k \in_R \mathcal{G}$ is a random value. It follows that in this case $\mathcal{A}$ outputs $b_{\text{test}} = b$ with probability $1/2 + \delta(n)$. This implies that

$$\left|\Pr[D_F^{F_k}(1^n) = 1] - \Pr[D_F^H(1^n) = 1]\right| = \delta(n)$$

and so δ must be a negligible function, by the assumption that F_k is a pseudorandom function. Combining the above, we have that

$$\left|\Pr[D(g^a, g^b, g^{ab}) = 1] - \Pr[D(g^a, g^b, k) = 1]\right| = \left|\frac{1}{2} + \epsilon(n) - \frac{1}{2} - \delta(n)\right| = |\epsilon(n) - \delta(n)|$$

and so ϵ must also be a negligible function, proving that $\mathsf{succ}^{\text{guess}}_{\mathcal{A}}$ occurs with probability that is at most negligibly greater than $1/2$, as required.

We now prove that for every probabilistic polynomial-time $\mathcal{A}$ interacting only with P_1 and P_2, it holds that

$$\Pr[\mathsf{succ}^{\text{comp}}_{\mathcal{A}}] < \frac{1}{2^\ell} + \mathsf{negl}(n)$$

Recall that $\mathsf{succ}^{\mathrm{comp}}_{\mathcal{A}}$ holds if P_1 and P_2 are *not* partnered, and yet $\mathsf{comp}_1 = \mathsf{comp}_2$. Since the session identifier in Π is defined to be the pair of public keys (pk_1, pk_2) exchanged in the first phase, we have that $\mathsf{succ}^{\mathrm{comp}}_{\mathcal{A}}$ can only hold if P_1 and P_2 hold different public keys. This occurs if at least one of the keys sent by an instance was not received as-is by the other instance, but was rather "modified" en route by $\mathcal{A}$.

We introduce the following notation that will be helpful in the proof below. If one instance sends a message α, then we denote by α' the message received by the other instance. Thus, the public key sent by P_1 is denoted pk_1 and the public key received by P_2 is denoted pk'_1. Using this notation, we have that P_1 and P_2 are not partnered if P_1 has $\mathsf{sid}_1 = (pk_1, pk'_2)$ and P_2 has $\mathsf{sid}_2 = (pk'_1, pk_2)$, and either $pk_1 \neq pk'_1$ or $pk_2 \neq pk'_2$ or both.

Now, the first authentication stage involves P_2 sending $c_2 = C(pk_2, pk'_1, 0; r_2)$ and P_1 receiving some c'_2. Then, P_1 sends r_1 and P_2 receives r'_1. Finally, P_2 returns r_2 and P_1 receives some string r'_2. Using the above notation, we have that $\mathsf{succ}^{\mathrm{comp}}_{\mathcal{A}}$ occurs if and only if

$$(pk_1, pk'_2) \neq (pk'_1, pk_2) \quad \text{and} \quad g(pk_1, pk'_2, r_1, r'_2) = g(pk'_1, pk_2, r'_1, r_2) \tag{2}$$

(Note that $\mathsf{comp}_1 = g(pk_1, pk'_2, r_1, r'_2)$ and $\mathsf{comp}_2 = g(pk'_1, pk_2, r'_1, r_2)$.) Without loss of generality, we assume that $\mathcal{A}$ always causes P_1 and P_2 to be not partnered (otherwise it always fails so this does not make any difference), and so $(pk_1, pk'_2) \neq (pk'_1, pk_2)$ always. We analyze the probability that Eq. (2) holds in two disjoint cases related to the possible schedulings of messages by $\mathcal{A}$:

1. *Case 1 – P_2 sends r_2 after P_1 has received c'_2:* The main difficulty in the proof here is due to the fact that it is theoretically possible that $\mathcal{A}$ can make c'_2 depend on c_2 (and likewise r'_2 can depend on r_2). Therefore, the inability of $\mathcal{A}$ to succeed depends on the *non-malleability* of the commitment scheme C; see Definition A.1 in Appendix A (familiarity with the exact definition is needed for the proof below). Let $\mathcal{A}$ be a probabilistic polynomial-time adversary. We prove that $\mathsf{succ}^{\mathrm{comp}}_{\mathcal{A}}$ occurs in this case with probability that is at most negligibly greater than $2^{-\ell}$. First, we show that there exists an adversary $\hat{\mathcal{A}}$, a relation $\hat{R}$ and a distribution $\hat{D}$ for the non-malleability experiment $\mathsf{Expt}^{\mathrm{real}}_{\hat{\mathcal{A}},\hat{R},\hat{D}}(1^n)$ such that

$$\Pr[\mathsf{Expt}^{\mathrm{real}}_{\hat{\mathcal{A}},\hat{R},\hat{D}}(1^n) = 1] = \Pr[\mathsf{succ}^{\mathrm{comp}}_{\mathcal{A}}] \tag{3}$$

   Adversary $\hat{\mathcal{A}}$ for the non-malleability experiment begins by invoking $\mathcal{A}$ (the adversary for the key exchange protocol) and emulating the parties P_1 and P_2 until the point that P_2 is supposed to send c_2. Note that at this point, the keys pk'_1 and pk_2 are fully defined. Then, $\hat{\mathcal{A}}$ outputs $z = (pk_2, pk'_1)$. The distribution $\hat{D}$ receives z, chooses a random $r^a_2 \in_R \{0,1\}^{n/2}$ and outputs $m_1 = (pk_2, pk'_1, 0, r^a_2)$. Adversary $\hat{\mathcal{A}}$ then receives com_1 (by the definition of C, com_1 is a commitment to m_1 using random coins r^b_2 of length $n/2$), and hands it to $\mathcal{A}$ as if it is the commitment c_2 sent by P_2 in the key exchange protocol. When $\mathcal{A}$ sends a commitment c'_2 to P_1, then $\hat{\mathcal{A}}$ defines this to be com_2 and outputs it. Following this, as defined in the non-malleability experiment, $\hat{\mathcal{A}}$ receives dec_1 which is the string $(pk_2, pk'_1, 0, r^a_2, r^b_2)$. $\hat{\mathcal{A}}$ defines $r_2 = (r^a_2, r^b_2)$ and hands it to $\mathcal{A}$ as if coming from P_2. Finally, when $\mathcal{A}$ wishes to send r'_2 to P_1, $\hat{\mathcal{A}}$ defines $\mathsf{dec}_2 = (pk'_2, pk_1, 0, r^{a\prime}_2, r^{b\prime}_2)$ and $\sigma = (pk_1, pk'_2, r_1, r'_1, r^b_2, r^{b\prime}_2)$ where these are the appropriate strings sent in the emulation carried out by $\hat{\mathcal{A}}$ ($\hat{\mathcal{A}}$ needs to include r^b_2 and $r^{b\prime}_2$ because these are not part of the messages m_1, m_2 but randomness used to generate the commitments). Finally, R outputs 1 if and only if $g(pk_1, pk'_2, r_1, r'_2) = g(pk'_1, pk_2, r'_1, r_2)$, where the values input to g are parsed from m_1, m_2 and σ. Eq. (3) follows from the observation that $\hat{\mathcal{A}}$'s emulation of an execution of Π for $\mathcal{A}$ is perfect, and from the fact that R outputs 1 if and only if $\mathsf{succ}^{\mathrm{comp}}_{\mathcal{A}}$ occurs.

Now, by the assumption that the commitment scheme C is non-malleable with respect to opening, we have that there exists an adversary $\hat{\mathcal{A}}'$ such that

$$\Pr[\mathsf{Expt}^{\text{real}}_{\hat{\mathcal{A}},\hat{R},\hat{D}}(1^n) = 1] < \Pr[\mathsf{Expt}^{\text{sim}}_{\hat{\mathcal{A}}',\hat{R},\hat{D}}(1^n) = 1] + \mathsf{negl}(n)$$

We don't know how $\hat{\mathcal{A}}'$ works, but we do know that it first outputs a string z and then a pair (σ, m_2). The output of the experiment is then equal to 1 if and only if $g(pk_1, pk_2', r_1, r_2') = g(pk_1', pk_2, r_1', r_2)$, where $pk_1, pk_2', pk_1', pk_2, r_1, r_1',\ r_2'$ are all derived from z, σ and m_2, and r_2^a is *uniformly distributed* and independent of all other values. We stress that r_2^a is random and independent since $r_2^a \in_R \{0,1\}^{n/2}$ is chosen randomly by $\hat{D}$ and not given to $\hat{\mathcal{A}}'$. (Note that we cannot say anything about r_2^b because this is chosen by $\hat{\mathcal{A}}'$ as part of σ.) We conclude this case by using the computational 2-universality of g. That is, letting $\beta = \mathsf{comp}_1$ (which is fully defined by z, σ and m_2) and $\alpha_1 = (pk_1', pk_2, r_1'), \alpha_2 = r_2^b$ (again, fully defined by z and σ), we have that

$$\Pr_{r_2^a \leftarrow \{0,1\}^{n/2}}[g(\alpha_1, r_2^a, \alpha_2) = \beta] < \frac{1}{2^\ell} + \mathsf{negl}(n).$$

Thus

$$\Pr[\mathsf{Expt}^{\text{sim}}_{\hat{\mathcal{A}}',\hat{R},\hat{D}}(1^n) = 1] < \frac{1}{2^\ell} + \mathsf{negl}(n),$$

implying that

$$\begin{aligned}\Pr[\mathsf{succ}^{\text{comp}}_{\mathcal{A}}] &= \Pr[\mathsf{Expt}^{\text{real}}_{\hat{\mathcal{A}},\hat{R},\hat{D}}(1^n) = 1] \\ &< \Pr[\mathsf{Expt}^{\text{sim}}_{\hat{\mathcal{A}}',\hat{R},\hat{D}}(1^n) = 1] + \mathsf{negl}(n) < \frac{1}{2^\ell} + \mathsf{negl}'(n)\end{aligned}$$

proving that the probability that $\mathsf{succ}^{\text{comp}}_{\mathcal{A}}$ is at most negligibly greater than $2^{-\ell}$, as required. (We remark that the above proof only works in the scheduling case where $\mathcal{A}$ sends c_2' to P_1 before receiving r_2 from P_2, because in the non-malleability experiment com_2 must be output by the adversary before it receives dec_1.)

2. *Case 2 – P_2 sends r_2 before P_1 has received c_2':* Observe that phase 2 involves P_2 sending c_2, P_1 sending r_1 and then P_2 replying with r_2. Thus, in this case, $\mathcal{A}$ effectively runs the executions with P_1 and P_2 *sequentially*. That is, $\mathcal{A}$ concludes phase 2 with P_2 before beginning phase 2 with P_1. Intuitively, in this case, $\mathsf{succ}^{\text{comp}}_{\mathcal{A}}$ can only occur with probability $2^{-\ell}$ because comp_2 is *fixed* before r_1 is chosen by P_1. Thus, the computational 2-universality of g suffices to show that $\mathsf{comp}_1 = \mathsf{comp}_2$ with probability at most negligibly greater than $2^{-\ell}$. More formally, let β be the comp_2 value of P_2. By this scheduling case, this is fixed before P_1 receives c_2' and so, in particular, before it chooses r_1. However, if $\mathcal{A}$ can choose r_2' after receiving r_1 from P_1, then the property of g no longer holds (recall that α and β must be independent of r). Intuitively this is not a problem due to the computational binding property of C.

   Formally, let $\mathcal{A}$ be an adversary for the key exchange protocol; we assume that $\mathcal{A}$ always sends a valid r_2' to P_1 (otherwise P_1 rejects). We construct α and β as required for g as follows. Invoke $\mathcal{A}$ and emulate an execution with P_1 and P_2 until the end of phase 2 with P_1. Since phase 2 has finished, the strings c_2' and r_2' are fully defined, as are $pk_1, pk_1', pk_2, pk_2', r_2, r_1'$ (recall that phase 2 with P_2 concluded before it even started with P_1). These values therefore define α and β as follows: $\alpha = (pk_1, pk_2', r_2')$ and $\beta = \mathsf{comp}_2 = g(pk_1', pk_2, r_1', r_2)$. Now, we argue that

$$\Pr[\mathsf{succ}^{\text{comp}}_{\mathcal{A}}] < \Pr_{r \leftarrow \{0,1\}^n}[g(\alpha, r) = \beta] + \mathsf{negl}(n) \tag{4}$$

In order to see that this holds, after $\mathcal{A}$ sends r_2' at the end of phase 2 (in the above procedure for determining α and β), rewind $\mathcal{A}$ to the point before r_1 is sent by P_1. Then, replace it with the random string r in Eq. (4). The value r_1 sent by P_1 in the process of determining α and β is identically distributed to the value r from Eq. (4). Now, there are two possibilities: $\mathcal{A}$ sends the same r_2' as when determining α and β, or $\mathcal{A}$ sends a different r_2'. In the first case, we have that $\mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$ occurs *if and only if* $g(\alpha, r) = \beta$. In the second case, we have that $\mathcal{A}$ can be used to contradict the binding property of C (the formal reduction of this fact is straightforward and thus omitted). Thus, this case can occur with at most negligible probability. Eq. (4) therefore follows. By the security of g, we have that $\mathsf{succ}_{\mathcal{A}}^{\mathrm{comp}}$ occurs with probability at most negligibly greater than $2^{-\ell} + \mathsf{negl}(n)$, as required.

This completes the proof of security. ■

References

[1] *Specification of the Bluetooth system.* Covered Core Package version 2.1 + EDR. 26 July 2007.

[2] M. Bellare, D. Pointcheval and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Eurocrypt 2000*, Springer-Verlag (LNCS 1807), pp.139–155, 2000.

[3] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO'93*, Springer-Verlag (LNCS 773), pages 232–249, 1994.

[4] M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: the Three Party Case. In the 27*th STOC*, pages 57–66, 1995.

[5] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Eurocrypt 2001*, Springer-Verlag (LNCS 2045), pages 453–474, 2001.

[6] G. Di Crescenzo, J. Katz, R. Ostrovsky and A. Smith. Efficient and Non-interactive Non-malleable Commitment. In *EUROCRYPT 2001*, Springer-Verlag (LNCS 2045), pages 40–59, 2001.

[7] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. In 30*th STOC,* pages 141–150, 1998.

[8] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22, pages 644–654, 1976.

[9] D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing,* 30(2):391–437, 2000.

[10] C. Gehrmann, C. Mitchell and K. Nyberg. Manual Authentication for Wireless Devices. *RSA Cryptobytes*, vol. 7, pages 29–37, 2004.

[11] J.H. Hoepman. The Ephemeral Pairing Problem. In *Financial Cryptography*, Springer-Verlag (LNCS 3110), pages 212–226, 2004.

[12] J.H. Hoepman. Ephemeral Pairing on Anonymous Networks. In the 2*nd Conference on Security in Pervasive Computing*, Springer-Verlag (LNCS 3450), pages 101–116, 2005.

[13] S. Laur and K. Nyberg. Efficient Mutual Data Authentication Using Manually Authenticated Strings. In *CANS 2006*, Springer-Verlag (LNCS 4301), pages 90–107, 2006.

[14] M. Naor, G. Segev and A. Smith. Tight Bounds for Unconditional Authentication Protocols in the Manual Channel and Shared Key Models. In *CRYPTO 2006*, Springer Verlag (LNCS 4117), pages 214–231, 2006.

[15] S. Vaudenay. Secure Communications over Insecure Channels Based on Short Authenticated Strings. In *CRYPTO 2005*, Springer-Verlag (LNCS 3621), pages 309–326, 2005.

A Non-Malleable Commitments – Definition

Informally speaking, a commitment scheme is non-malleable if given a commitment c it is computationally hard to generate a commitment c' that is "related" by some predefined relation R. When considering computationally binding commitments, it is possible that c' can actually be a commitment to any value. Therefore, it is not clear what it means that the value committed to in c' is related to the value committed to in c. This problem is solved by defining the notion *with respect to opening* [7]. This means that given a decommitment for c to some value x, it is hard for the adversary (who generated c' after being given c) to generate a decommitment for c' to some x' so that x' is related to x. Of course, the probability of success depends on the relation (some are "easier" than others). Therefore, the requirement is that it is possible to generate a related commitment c' given c with the same probability as it is possible to generate a related x' without even being given x. This is formalized by defining two experiments: a real experiment in which the adversary is given c, and a simulation experiment where the adversary just outputs a message and hopes that it's related. Our formal definition is adapted from [6] with two minor changes. First, we allow the adversary $\mathcal{A}$ to provide input to the distribution machine that generates the value to be committed to. Second, we allow the adversary to output state information which is used by the relation. Both of these changes do not seem to make it particularly easier for the adversary, but they make the definition much more useful for proving the security of protocols which rely on non malleability. The experiments relate to a probabilistic polynomial-time adversary $\mathcal{A}$, a polynomial-time computable relation R and a probabilistic polynomial-time samplable distribution D. We also denote the committer/sender algorithm by P_1 and the receiver algorithm by P_2 (the receiver takes for input a commitment string and a decommitment value and output a string that represents the value that was committed to). The experiments are defined as follows:

Experiment $\mathsf{Expt}^{\text{real}}_{\mathcal{A},R,D}(1^n)$**:**

1. $z \leftarrow \mathcal{A}(1^n)$
2. $m_1 \leftarrow D(1^n, z)$
3. $(\mathsf{com}_1, \mathsf{dec}_1) \leftarrow P_1(m_1)$
4. $\mathsf{com}_2 \leftarrow \mathcal{A}(1^n, \mathsf{com}_1)$
5. $(\sigma, \mathsf{dec}_2) \leftarrow \mathcal{A}(1^n, \mathsf{com}_1, \mathsf{dec}_1)$
6. $m_2 \leftarrow P_2(\mathsf{com}_2, \mathsf{dec}_2)$
7. Output 1 if and only if $\mathsf{com}_1 \neq \mathsf{com}_2$ and $R(\sigma, m_1, m_2) = 1$

Experiment $\mathsf{Expt}^{\mathrm{sim}}_{\mathcal{A}',R,D}(1^n)$**:**

1. $z \leftarrow \mathcal{A}'(1^n)$
2. $m_1 \leftarrow D(1^n, z)$
3. $(\sigma, m_2) \leftarrow \mathcal{A}'(1^n)$
4. Output 1 if and only if $R(\sigma, m_1, m_2) = 1$

We now define security by stating that for every $\mathcal{A}$ in the real experiment there exists an $\mathcal{A}'$ who succeeds with almost the same probability in the simulation experiment. We allow the machine $\mathcal{A}'$ to know the distribution machine D and relation R (unlike [6]); this suffices for our proof of security and is possibly a weaker requirement.

Definition A.1 *A non-interactive commitment scheme C with sender/receiver algorithms (P_1, P_2) is* non-malleable with respect to opening *if for every probabilistic polynomial-time $\mathcal{A}$, every probabilistic polynomial-time samplable distribution D and every polynomial-time computable ternary relation R, there exists a probabilistic polynomial-time $\mathcal{A}'$ and a negligible function* negl *such that:*

$$\Pr\left[\mathsf{Expt}^{\mathrm{real}}_{\mathcal{A},R,D}(1^n) = 1\right] < \Pr\left[\mathsf{Expt}^{\mathrm{sim}}_{\mathcal{A}',R,D}(1^n) = 1\right] + \mathsf{negl}(n)$$