High-Speed Dating Privacy-Preserving Attribute Matching for RFID

Lejla Batina[1(✉)], Jens Hermans[2], Jaap-Henk Hoepman[1], and Anna Krasnova[1]

[1] Digital Security Group, Radboud University, Heyendaalseweg 135, 6525 AJ Nijmegen, The Netherlands
{lejla,jhh}@cs.ru.nl
anna@mechanical-mind.org
[2] KU Leuven and IMinds, Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
jens.hermans@esat.kuleuven.be

Abstract. This paper presents a new approach for RFID tag *attribute matching* problem. Unlike previous approaches, most notably the T-Match protocol, presented in [9], we do not need a central database server or any connectivity between readers. Furthermore, we do not need expensive homomorphic encryption or multiparty computation and we extend attribute matching to multiple attributes per tag; a feature that broadens the range of possible applications of the protocol. We achieve this increased flexibility and decreased complexity by moving some relatively cheap cryptographic computations to the tags. Specifically, one of the protocols presented in this paper only needs a (lightweight) hash function implemented on the tags. Two other protocols additionally need asymmetric encryption, which is feasible on more powerful tags that support elliptic-curve scalar multiplication.

Keywords: Attribute matching · RFID · Privacy · Unforgeabilitiy · Unlinkability

1 Introduction

RFID technology is predominantly used for identification and authentication of items and persons. In a typical setup the tag has some key which it uses in an identification or authentication protocol with the reader. Attribute-matching protocols on the other hand focus on determining whether two or more tags have a set of attributes that match a specific relationship. By using an attribute matching protocol one can also authenticate tags by simply matching it with a tag known to be genuine. Provided both tags share an attribute (or key) they will pass the validation.

Anna Krasnova – This research was conducted within the Privacy and Identity Lab (PI.lab, http://www.pilab.nl) and funded by SIDN.nl (http://www.sidn.nl/). Permanent ID of this document: bdc1cee0d2de9dab7248278472d954a5. Date: 2014.08.18.

N. Saxena and A.-R. Sadeghi (Eds.): RFIDSec 2014, LNCS 8651, pp. 19–35, 2014.
DOI: 10.1007/978-3-319-13066-8_2

As an important application for tag authentication by matching we envision preventing counterfeit products. A producer can provide a reference tag to the verifier, containing the same key as the genuine products. By matching product tags with the reference one can detect counterfeits, without the key ever leaving the tags which can be protected on hardware level. Such an approach has major advantages compared to classical authentication protocols. Symmetric key authentication protocols are very efficient but require storing the secret key on the reader. Authentication by matching combines this efficiency with a typical property of asymmetric protocols that do not require secret (private) keys on the verifier.

To illustrate our protocols we will use the example of a *speed-dating* party (or rather, with the protocols presented in this paper, a *high-speed-dating* party). The typical setup of a *speed-dating* party is that singles try to find a partner through many short meetings with many people. The goal of these short conversations is to find out whether the two people share interests and want to engage in a longer conversation or a date. *High-speed dating* replaces these short matchmaking conversations by scanning RFID tags as follows: The organizer of the party has collected all relevant attributes (hobbies, city of residence, kids and pets preferences, etc.) of all participants in advance. Every participant receives an RFID tag which stores his or her attributes. When two persons want to decide whether it is worth starting a conversation, they just have their tags scanned simultaneously by a reader, and the reader determines whether these persons have overlapping interests and wishes. Thus, the task of a reader is to detect the fact that two tags have matches in their interests, and output the number of these matches. No false positives are desirable, since false positive will steal time of participants. The obvious target group for such a party are "nerds and geeks", who typically have very serious concerns about their privacy. They do not want a reader or another person to learn anything about them, except for the fact whether they share interests with another person or not. Also, tags' unlinkability should be preserved, so nobody can trace tags. Last but not least, no attributes stored on tags shall be disclosed. This application may not sound like the most serious scenario, but it illustrates very well what the protocol does and what properties we expect from the protocol.

This paper presents three private-attribute-matching protocols (or *speed-dating protocols*). The first protocol uses only symmetric cryptography, the second only asymmetric cryptography, and the last a combination of both. None of the protocols requires readers to be connected to a central database; they are furthermore not required to have specific knowledge about the tags. The first two protocols provide matching for one attribute per tag. The symmetric protocol provides speed and efficiency at the cost of a lower privacy protection level. The asymmetric ones provides better privacy at the cost of a single asymmetric encryption step. The hybrid encryption protocol still uses a single encryption step with asymmetric encryption, and several with symmetric one, in order to support tags with several attributes. All protocols provide provable security against false positives and provable privacy protection.

The remainder of this paper is organized as follows. Section 2 presents the model of the system and the adversarial games. Section 3 introduces the light-weight symmetric protocol. Section 4 introduces the two remaining protocols using asymmetric primitives. Related work is discussed in Sect. 5.

2 Model and Notations

Let T be the set of all tags in the system. Each tag $t_i \in T$, $i \in \{1, 2, \ldots, n\}$ is supplied with attributes. Each attribute is a human-readable string of arbitrary length, the set of all attributes in the system is denoted as $C = \{a_1, \ldots, a_p\}$. Let the security parameter be λ, and the number of attributes and tags be polynomially bounded in λ. Each attribute in the system is related to a secret key stored on a tag in the following way. An issuer starts with an attribute set C. To setup the system S, the tag issuer generates a set of *keys* $K = \{k_1, \ldots, k_p\}$, which are each associated with an attribute. Each key k_j for $j \in \{1, \ldots, p\}$ is a λ-bit string.

Each tag stores a subset of K of cardinality at most m, where $1 \leq m \leq p$. For two tags t_i and t_j, which share an attribute a_i, the corresponding key for a_i is thus stored on both tags. For simplicity, further in the text we use the term keys and attributes interchangeably.

The goal of the protocol is to determine the number of attributes on two tags that match. Let the state S_i denote the set of attributes stored on a tag t_i. Note, that if tags t_i and t_j have the very same attributes assigned to them, their states are equal $S_i \equiv S_j$. This state is assigned by the issuer during the setup phase: a function $Setup(\lambda)$ is used to assign state S_i to a tag t_i, with $i \in \{1, 2, ..., n\}$, and generates keys for readers (if necessary). All secret values are generated taking the security parameter λ into account. Later, during the protocol run, a reader R simultaneously scans two tags t_i and t_j to obtain the result of the function $Match : T \times T \rightarrow \mathbf{N}$. This function computes the cardinality of the intersection of the states of two tags: $Match(t_i, t_j) = |S_i \cap S_j|$. Some applications do not require $Match$ to compute the cardinality of $S_i \cap S_j$, but only need to know whether this intersection is empty or not. For those applications we use $Match : T \times T \rightarrow \{0, 1\}$.

The function $Match$ must fulfill the following properties:

1. *Correctness:* In the absence of adversaries the output is correct.
2. *Unforgeability:* False positives are impossible, that is an adversary is unable to convince a reader that tags match on more attributes than they actually are.
3. *Unlinkability:* Neither a reader, nor an external adversary is able to decide, whether in two protocol runs the same tags participated twice or not.
4. *Confidentiality:* After a protocol run nobody can learn any of the attributes (corresponding keys) stored on a tag, unless possessing a valid key for an attribute. The amount of attributes stored on tags is computationally hard to derive from a protocol run, unless possessing all keys in the system.

The protocols presented in this paper do not protect against false negatives. Thus, they are useful for applications that do not require to prevent false negatives. To summarize, the system has the following functions:

1. *Setup*(λ): is used to generate a private key or a public/private key pair for a reader (if specified by the protocol) based on the security parameter λ. It assigns a state S_i to all tags $t_i \in T$. The state includes the set of secret keys $k_{i1}, k_{i2}, ..., k_{im}$ assigned to the tag and the publicly known information (e.g. public keys of readers).
2. *Match*(t_i, t_j): Is a protocol carried out by two tags t_i and t_j, and a reader R. The protocol is initiated by the reader. As a result of the protocol, the reader obtains the cardinality of the intersection of S_i and S_j.

2.1 Adversary Model

The security of our protocols relies on the secrecy of the keys stored on tags. We thus make the common assumption that those keys are stored in a secure way, and that computations involving those keys are implemented in a way that does not leak information about the keys (for example, through side channels). The adversary controls all the communication, pretends to be one of the valid tags, but does not perform relay attacks using a tag outside the proximity of the reader. A reader is assumed to behave according to the protocol. Thus, it is considered "honest but curious".

The type of an adversary A is specified by the actions he can perform. Let π be a protocol execution entity. The oracles below define the whole set of possible actions. An adversary gets an access to a subset if oracles depending on his type. Oracles distinguish between the left and the right message denoted as m_{left}, m_{right}. This notion is needed to distinguish communication with tag of the left and right side. The oracles are:

- *Launch*($m_{\text{left}}, m_{\text{right}}$) $\rightarrow \pi, m$: when this oracle is called, the reader starts a new protocol execution π by sending out the message m. The whole execution of the protocol can then be performed using oracles *SendReader* and *SendTag*. These two oracles can be used to simulate the *Execute* oracle from the model defined by Juels and Weis [16]
- *SendReader*($m_{\text{left}}, m_{\text{right}}, \pi$) $\rightarrow (m'_{\text{left}}, m'_{\text{right}})$: sends a message m to a reader from the left side (right side or both) in the context of protocol execution π. The output of the oracle is a response of the reader m' sent in any of the directions according to the description of the protocol.
- *SendTag*(m, t_i) $\rightarrow m'$: sends a message m to a tag t_i. The output of the oracle is a response of the tag m'.
- *Result*(π) $\rightarrow x$: outputs the result of function *Match* after the protocol execution π.
- *Corrupt*(t_i) $\rightarrow s_i$: returns the internal state of the tag, allowing an adversary to learn all secret keys stored on this tag.

The model and the privacy game presented in this section are inspired by work of Juels and Weis [16], Vaudenay [19] and Hermans et al. [14]. Unfortunately, all these models were designed with the classes of protocols in mind that are different from our protocol. All the models consider the scenario of communication between a reader and a single tag. We therefore adopt the classification by Vaudenay and modify the Juels and Weis game for unlinkability to fit the needs of matching protocols.

The classification by Vaudenay is faceted in two dimensions. An attacker who does not have access to the *Result* oracle is called *NARROW*. An attacker who does is called *WIDE*. An attacker who cannot corrupt tags is called *WEAK*. An attacker who is not allowed to perform any protocol interactions after he corrupted one tag is called *FORWARD*. An attacker without any restrictions regarding corruption of tags is called *STRONG*.

2.2 Unforgeability

The goal of an adversary is to convince a reader that the number of matching attributes is larger than it actually is. We call a protocol *unforgeable* if it resists this attack. Let S be a system and A be an adversary.
Unforgeability is defined as the following game $\mathbf{Exp}^{forge}_{S,\,A}(\lambda)$:

Setup: Setup(λ) is used to initialize all readers and tags.
Learning: The adversary may perform calls to the available oracles on the given set of tags T. The set of available oracles depends on the adversary type. The strongest adversary gets access to: *Launch*, *SendReader*, *SendTag*, *Corrupt*. Let the union of the sets of all corrupted tags be C_t.
Challenge:

1. The adversary chooses a tag t_i, to which he did not call a *Corrupt* oracle.
2. The tag t_i is removed from the set T. The challenger returns t_i to the adversary.
3. The adversary is not allowed to modify any of the messages sent to or from the tag t_i. The adversary is simulating a tag t_j on the other side.

Result: The experiment outputs true if the reader outputs a value lager than $|S_i \cap (S_j \cup C_t)|$.

The advantage of adversary of winning the game is defined as:

$$Adv^{forge}_{S,\,A}(\lambda) = Pr[\mathbf{Exp}^{forge}_{S,\,A}(\lambda) = true]$$

We call the system unforgeable if a maximal advantage of all polynomial time adversaries is negligible in the security parameter λ. During the challenge phase an adversary can only passively eavesdrop messages exchanged between the challenge tag and a reader. The adversary has to simulate the other tag, which will be matched by the reader with the challenge tag.

The model above considers tag corruption by taking into account that the keys extracted from corrupted tags will trivially allow the adversary to increase the match count output. For the protocols presented in this paper only WEAK adversaries are considered for unforgeability, and hence $C_t = \emptyset$.

2.3 Unlinkability

The goal of the attacker A is to distinguish tags, thus breaking their unlinkability. In case an attacker is able to obtain the result of the protocol run, an attack on unlinkability becomes trivial as pointed out in [9]. It is sufficient to have one tag participate in two protocol runs with potentially different tags. By comparing the result (i.e. cardinality of the intersection) one can determine if that tag was matched against different tags or not. That implies that the *Result* oracle cannot be used by an attacker. The match protocol by its nature is giving away information about tags, namely the relationships between them. There are two approaches for designing the speed dating protocol to tackle this problem. One is to make sure that the protocol itself is not providing any evidence of the relationships between tags, except for the output of the reader. The other is to provide only a *Minimal* level of protection. In this minimal model an attacker is unable to recognize the same tag he was observing before, once he initiates a protocol with only this tag. To illustrate, assume an attacker was collecting interactions among tags on the speed dating party. After the party is over, he suddenly sees a person wearing an RFID tag from this party. An attacker triggers a protocol, having no other valid tag at hand. He should be unable to learn the identity of the tag even having all the old protocol run transcripts at hand.
Unlinkability of an attacker A in the system S is defined as the following game $\mathbf{Exp}^{link}_{S,\,A}(\lambda)$:

Setup: $Setup(1^\lambda)$ is used to initialize all readers and tags.
Learning: The adversary may perform calls to the available oracles on the given set of tags T. The set of available oracles depends on the adversary type: *Launch*, *SendReader*, *SendTag*, *Corrupt*.
Challenge:

1. The adversary chooses two tags t_i and t_j, to which he did not call a *Corrupt* oracle.
2. The challenger assigns $t_0^* = t_i$ and $t_1^* = t_j$. Both tags are removed from the set T.
3. Let $b \in_R \{0, 1\}$. The challenger returns t_b^* to the adversary.
4. The adversary is allowed to perform calls to the oracles: *Launch*, *SendReader*, *SendTag*, having tag t_b^* on one of the sides and any of the tags from the set T on the other.
5. The adversary outputs a guess bit b'.

Result: The experiment outputs true if the adversary correctly outputs $b' = b$.

Minimal privacy is achieved if an attacker during challenge phase gets only one tag t_b^* to communicate with and he has to simulate a tag on the other side. This game modification is used only against a *WEAK* adversary, if the adversary has an access to the *Result* oracle. Otherwise an attacker could simulate all the tags he corrupted during the learning phase. Thus there would be no difference between two games, since an attacker could use all the broken tags to let them

communicate with the challenge tag. This would allow an attacker to use the knowledge of topology of tag's relationships to win the game.

The advantage of adversary of winning the game is defined as:

$$Adv^{link}_{S,A}(\lambda) = |Pr[\mathbf{Exp}^{link}_{S,A}(\lambda) = true] - \frac{1}{2}|$$

We call the system unlinkable if the maximal advantage of all polynomial time adversaries is negligible in the security parameter λ.

2.4 Cryptographic Primitives

One of the important primitives used in speed-dating protocols is a pseudo-random function (PRF), which cannot efficiently be distinguished from a truly random function. In the game definition of a PRF, an adversary submits inputs to the PRF challenger. The challenger replies with either an output of the PRF, or an output of a truly random function. The adversary wins if he has a non-negligible advantage to distinguish these two possible outputs. Let a secure pseudo-random function be denoted as $Fun_k(\cdot)$, where k is a key of the PRF function.

Note that we can efficiently construct a PRF from a hash function through the Merkle-Damgård iteration [18] as described, for example, in [7, Sect. 6]. This approach is particularly interesting for speed-dating protocols, since two of the three versions are using hash function anyway. In a similar way, one can construct a PRF using keyed modes of lightweight sponge-based hash functions like Quark [2] to construct a PRF function. A crytographic hash function is denoted as $H(\cdot)$ further in the text.

It was proven by Bellare et al. [3,4] that any PRF is a secure message authentication code (MAC). This property is essential for our protocols. Let us introduce *secure MACs* in a form of brief game description, for details see [12]. During the MAC-unforgeability game, an adversary queries the challenger with distinct messages and obtains MACs for them. He can also submit several message-tag pairs to the verification oracle. An adversary wins if he succeeds with a non-negligible advantage in outputting a valid message-tag pair not previously requested from the challenger.

Both encryption schemes used in the speed-dating protocol are required to have the *IND-CPA* property. Let a symmetric encryption scheme be denoted as $symENC = (G', E', D')$ and an asymmetric encryption scheme as $pkENC = (G, E, D)$. Let us briefly sketch the game for this property. During the learning phase an adversary gets to query an encryption oracle, which answers an adversary with ciphertexts of received plaintext messages. During the challenge phase the adversary submits several pairs of distinct non-repeating messages (m_0, m_1). Depending on the initial decision, the challenger answers with a ciphertext of one of the messages, either m_0 or m_1. The adversary succeeds if he has a non-negligible advantage to distinguish these two possible outputs.

3 One-Key Symmetric Speed Dating

The protocol presented in this section prevents false positives and provides minimal privacy. False positives occur when a tag does not possess any attribute matching with attributes on the other tag. And yet it manages to make a *Match* function output that tags have a match. Minimal privacy is the privacy protection achieved against an adversary that can read output of the protocol runs and thus build a topology of tag's relationships. The protocol only requires a few calls to a (lightweight) hash functions.

3.1 Single Attribute per Tag

We start with the setting that each user has a single attribute. Assume tags named (for convenience) by their owners, namely Alice and Bob. Tags Alice and Bob possess respectively keys k_A, $k_B \in K$. These keys are representing attributes of tags. A reader scans both tags to figure out whether their attributes match or not. Figure 1 depicts the protocol.

The general idea of the protocol is the following:

1. *Commit phase.* Tags generate random numbers and exchange commitments with each other. These commitments later on help a reader to identify replies of tags and prevent cheating. The exchange is happening with the help of the reader.
2. *Check phase.* Tags create pseudo-random values from the challenges by feeding them to a PRF function *Fun*. These values are exchanged with the help of a reader. Tags perform a check of the values generated by the other tag using their secret keys. After this phase tags know whether they have an equal key or not.

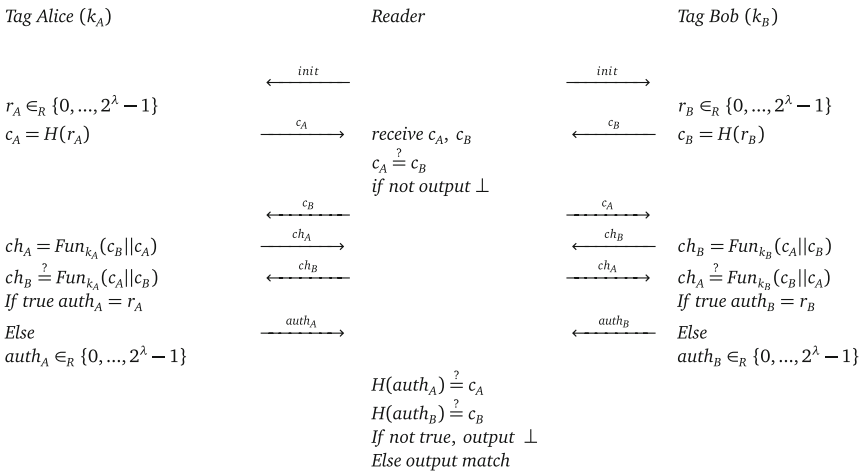

Fig. 1. One-key symmetric speed dating protocol

3. *Match phase.* If there is a match, tags open their commitments towards the reader, which determines the result of the protocol.

Commit phase

1. The tags generate random values and calculate commitments to these values. Alice generates: $r_A \in_R \{0, ..., 2^\lambda - 1\}$, $c_A = H(r_A)$. Bob generates: $r_B \in_R \{0, ..., 2^\lambda - 1\}$, $c_B = H(r_B)$.
2. The tags send the commitments c_A, c_B to a reader.
3. The reader checks if $c_A = c_B$. If so, the protocol run is terminated with output $\perp$. The reader forwards c_B to Alice and c_A to Bob.
4. Each tag concatenates the commitments and puts the value it generated on the last position. Alice, for example, obtains $c_B||c_A$.
5. Each tag computes the PRF function *Fun* using their group keys and send it to the reader. Alice computes: $ch_A = Fun_{k_A}(c_B||c_A)$. Bob computes: $ch_B = Fun_{k_B}(c_A||c_B)$.
6. The reader forwards ch_A to Bob, ch_B to Alice.

Check phase

1. Each tag checks the received commit value. Alice checks $ch_B \stackrel{?}{=} Fun_{k_A}(c_A||c_B)$. Bob checks $ch_A \stackrel{?}{=} Fun_{k_B}(c_B||c_A)$.
2. If the equality holds, Alice computes: $auth_A = r_A$. Else, she sends the response with $auth_A$ filled with a random value. Similarly, Bob computes: $auth_B = r_B$ if equality holds. Else, he sends the response with $auth_B$ filled with a random value.
3. The tags send $auth_A$ and $auth_B$ to the reader.

Match phase

1. The reader checks that $H(auth_A) \stackrel{?}{=} c_A$ and $H(auth_B) \stackrel{?}{=} c_B$. If any of these two values are false, the reader outputs $\perp$, otherwise it outputs a match.

Theorem 1. *If Fun is a PRF, then* $Adv^{forge}_{S,A}(\lambda)$ *of a WEAK adversary, that runs in polynomial time, to win the unforgeability game is negligible in the random oracle model.*

Proof. Since the protocol is symmetrical, we consider the protocol from the perspective of tag Alice without loss of generality. Since any PRF is a secure MAC [3,4], we can consider *Fun* to be a secure MAC function. Assume an adversary A can break the protocol unforgeability. We show that there exists an adversary A', that can then win the MAC unforgeability game using adversary A as an oracle.

The hash function is modeled as a random oracle *RO*. The adversary A' is interacting with a MAC game challenger possessing a key k_{ch}. Adversary A' is simulating the unforgeability game for an adversary A by answering all requests

an adversary A makes to oracles with a small exception. One particular tag t_i (or more) however is simulated with a help of the MAC challenger. This tag possesses a key k_{ch}. Let us call the corresponding attribute a_{ch}. This tags is simulated to A by A' in the following way:

1. First *SendTag*: A' outputs $c_A = RO(r_A)$ according to the protocol.
2. Second *SendTag*: A' provides $c_B||c_A$ as an input to the MAC challenger, and returns it's output as ch_A.
3. Third *SendTag*: A' validates the input value ch_b. If ch_b was generated by the MAC challenger, then A' knows the values should match (this can be double-checked with the help of the MAC verification oracle). Otherwise A' directly knows what the result should be, as he is simulating the rest of the tags in the system. A' returns an output, as specified by the protocol.

During the learning phase of the adversary A, A' gets to see different tuples of messages and corresponding tags: $m = (c_A||c_B), t = MAC_{k_i}(c_A||c_B)$, MAC value generated by a challenger. Since r_A and hence c_A is selected randomly, there will be no repeating values with overwhelming probability. Additionally, since H is a cryptographic hash function, A cannot win the unforgeability game by finding preimages of commitments c_A with overwhelming probability.

Assume A selects a tag with key k_{ch} as challenge tag. Assume the challenge tag is Bob and the adversary takes the role of Alice. Note that communication between the reader and the challenge tag is performed directly by the challenger. If A wins the unforgeability game, it has to produce a valid ch_A (otherwise r_B will not be sent to the reader and hence validation will fail). This ch_A will be the MAC of an input $c_A||c_B$ that was not used previously in the game, since c_B is fresh. Hence, ch_A can be forwarded to the MAC challenger to win the MAC unforgeability game.

The probability that the adversary selects the tag t_i is at least $\frac{1}{n}$. If the non-negligible advantage of A to win the game is ϵ, and the cardinality of the set of tags is n, then the advantage of A' is $\geq \frac{\epsilon}{n}$. This value is non-negligible, since n is polynomially bounded in a security parameter λ (see Sect. 2).

Theorem 2. *If Fun is a PRF, then* $Adv^{link}_{S,A}(\lambda)$ *of a WEAK adversary, that runs in polynomial time, to win the minimal unlinkability game is negligible in the random oracle model.*

Proof. Since the protocol is symmetrical, we consider the protocol from the perspective of tag Alice without loss of generality. We are going to show that an attacker is unable to distinguish any of the challenge tags from a simulator that is returning random values, and, thus, is unable to distinguish challenge tags themselves.

The hash function is modeled as a random oracle RO. We now simulate the *SendTag* oracle as follows to an adversary, explaining each step of the protocol:

1. A tag generates fresh pseudo-random values r_A and returns $RO(r_A)$.
2. Upon receiving c_B, the simulated tag returns a random value as ch_A.

3. Upon receiving ch_B, the simulated tag outputs a random value. Since the adversary is playing the minimal game, there is no other valid tag to create the proper ch_B value. Next, an attacker cannot forge a value output by *Fun*, as proven by Theorem 1. Thus, ch_B can never be accepted as valid.

The above simulated tag is indistinguishable from a real tag. First, the random ch_A is indistinguishable from $Fun_{k_A}(c_A||c_B)$ since *Fun* is a PRF, r_A is selected randomly and c_A is never repeated as an output of *RO*. Second, when replying to ch_B we can rely on the soundness of the protocol, as proven by Theorem 1. This ensures that it is impossible for an adversary to forge ch_B. So either it was sent by a tag (and hence a match will be found) or it was forged and should be rejected by the tag. Finally, since the game is minimal, an attacker does not have any possibility to distinguish between the real tag and the simulator by matching them with other tags and comparing the output.

Assume the challenge bit $b = 0$. Tag t_0 is indistinguishable from a simulated tag. The same argument applies to challenge bit $b = 1$. Hence, an attacker cannot distinguish between two tags.

4 Match Protocol for Asymmetric Encryption

The protocols presented in this section prevent false positives and provide a higher privacy level under the following requirement. An adversary must be unable to obtain the output of the protocol runs and, thus, build a topology of tag's relationships. The cost of the higher privacy level is the usage of public-key encryption.

4.1 One-Key Asymmetric Speed Dating Protocol

The advantage of the protocol in this section is that it protects confidentiality of the exchanged messages. Thanks to that, when an adversary corrupts tags and obtains their secret keys, it will not help him to succeed in identifying tags. This also implies that any external observer is unable to learn the result of the protocol from the exchanged messages. Also, it is easily expandable to handle the case of tags storing multiple attributes.

As in the previous section, assume each user can possess a single attribute. Tag Alice and Bob posses respectively group keys k_A, $k_B \in K$. In the protocol, an asymmetric encryption system $pkENC$ is used. A reader holds a public-private key pair (pk, sk), all tags are supplied with the public key pk of a reader.

This version of the protocol can be obtained from the symmetric one (Sect. 3) in two steps. The first step is encrypting messages sent between a reader and each tag. To ensure encryptions differ, tags append the random number they generate to the *Fun* value before encrypting. The second is to provide the same inputs to the *Fun* function on both tags, since there is no need to produce different outputs of the PRF function. The reason is that messages appended with unique randomness are sent encrypted. This way the protocol is protected

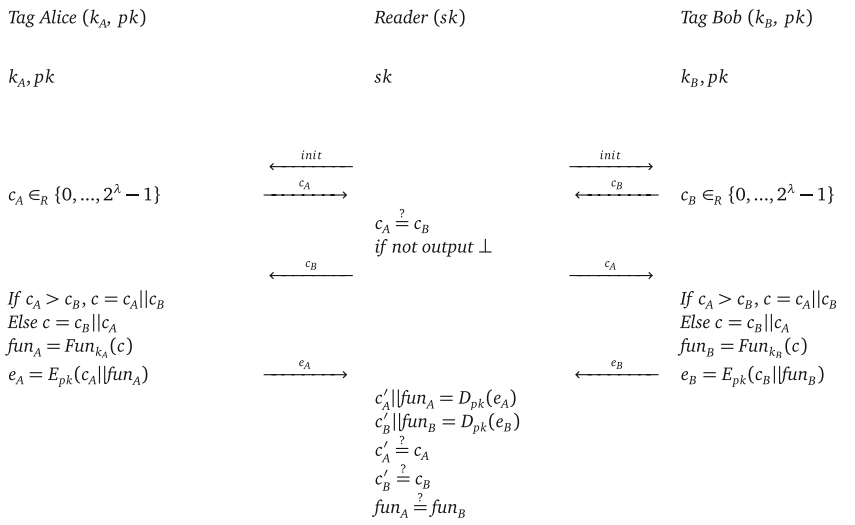

Fig. 2. One-key asymmetric speed dating protocol

from trivial replay attacks, and the need for commitments c_A, c_B and their openings is eliminated. Figure 2 illustrates the protocol.
Speed dating protocol:

1. Tags generate random values and send c_A, c_B to the reader. Alice generates: $c_A \in_R \{0,...,2^\lambda - 1\}$. Bob generates: $c_B \in_R \{0,...,2^\lambda - 1\}$
2. Reader checks if $c_A = c_B$, then protocol run is terminated with output $\perp$. Otherwise it exchanges random numbers between tags.
3. Tags sort random numbers. Assume, that $c_A > c_B$. Alice learns she has to put c_A in the beginning. Bob learns he has to append c_B to the end.
4. Tags compute *Fun* values using their group keys. Alice computes: $fun_A = Fun_{k_A}(c_A||c_B)$. Bob computes: $fun_B = Fun_{k_B}(c_A||c_B)$
5. Tags send *Fun* values $E_{pk}(c_A||fun_A)$ and $E_{pk}(c_B||fun_B)$ over secure channel.
6. Reader decrypts received values using his secret key sk.
7. Reader checks if there are appended c_A and c_B in the decrypted messages.
8. Reader checks if $fun_A \stackrel{?}{=} fun_B$. If true reader outputs *Match*. Otherwise it outputs $\perp$.

Theorem 3. *If Fun is a PRF, then* $Adv^{forge}_{S,A}(\lambda)$ *of a WEAK adversary, that runs in polynomial time, to win the unforgeability game is negligible in the random oracle model.*

The proof is omitted because of space limitations, and as it is very similar to the proof of the Theorem 1.

Theorem 4. *If the encryption scheme pkENC is IND-CPA secure, then* $Adv^{link}_{S,A}(\lambda)$ *of a (non-minimal) NARROW-STRONG adversary, that runs in polynomial time, to win the unlinkability game is negligible in the random oracle model.*

Proof. Since the protocol is symmetrical, we consider the protocol from perspective of tag Alice without loss of generality. The goal of the proof is to show that if an adversary A can break unlinkability of the protocol, the adversary A' can win IND-CPA game. The adversary A' interacts with the IND-CPA game challenger, possessing a key pair (pk, sk). The adversary A' simulates the unlinkability game for an adversary A by answering all requests A makes to oracles. The reader key pair in the simulation is the key pair of the IND-CPA game challenger.

During the learning phase A' queries the encryption oracle to obtain $E_{pk}(fun_A)$, which he then forwards to A. During the challenge phase A' creates messages for both challenge tags fun_{A0} and fun_{A1}. A' then submits both messages to the IND-CPA challenger. The received ciphertext $E_{pk}(fun_{Ab})$ is forwarded to A. If A can break unlinkability of the protocol, A' will be able to distinguish which message was encrypted.

A NARROW-STRONG adversary cannot get the result of the protocol run, but he can corrupt tags. Corrupting tags will provide an adversary with secret keys of tags. However, it does not help him in distinguishing tags and their output. This holds for the simple reason, all the information related to tags is transferred encrypted using the asymmetric IND-CPA encryption scheme *Enc*. This implies, encrypted messages do not provide an adversary any useful information.

Theorem 5. *If Fun is a PRF, then* $Adv^{link}_{S,A}(\lambda)$ *of a polynomial-time adversary that possesses the private key of a valid reader to win the minimal unlinkability game is negligible in the random oracle model.*

The proof is omitted because of space limitations, and as it is very similar to the proof of Theorem 2.

4.2 Many-Keys Asymmetric Speed Dating Protocol

Assume each user can possess at most m attributes. Tag Alice and Bob posses respectively group keys $s_A = \{k_A[i] \in K \cup \{\perp\} | i \in \{1, \ldots, m\}\}$, $s_B = \{k_B[i] \in K | i \in \{1, \ldots, m\}\}$. All tags are supplied with the public key pk and all readers possess the private key sk.

This protocol can be obtained from the one-key version by changing from using only asymmetric encryption scheme to using a hybrid one. An asymmetric encryption is used to securely transfer key material. A hash function is applied to it as a key derivation function. The result is used as a key for a semantically secure encryption system.

The next change is due to the necessity to hide the amount of attributes a tag has. Tags are generating $f_A[i]$ and $f_B[i]$ values using their keys, $i \in \{1, \ldots, m\}$. Whenever a tag has less then m attributes, the f values are filled with randomness. Tags perform random permutations on f values before they are sent to a reader. This is done in order to conceal the order of attributes, otherwise this could expose sensitive information about tags to the reader.

Upon obtaining and decrypting all of the $f_A[i]$ and $f_B[i]$ values from two tags, a reader starts by sorting both sets descending. After that a reader can easily

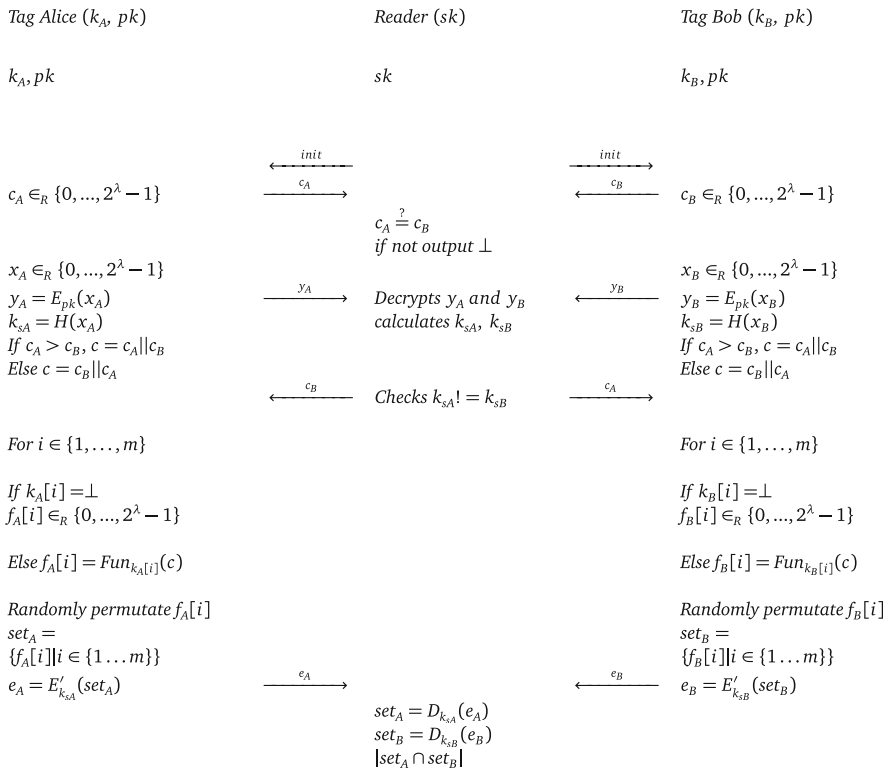

Fig. 3. Many-keys asymmetric speed dating protocol

compute an intersection of two sets *Int* and outputs its cardinality $|Int|$. Figure 3 depicts the protocol.

Theorem 6. *If Fun is a PRF and H is a cryptographic hash function, then* $Adv^{forge}_{S,A}(\lambda)$ *of a WEAK adversary, that runs in polynomial time, to win the unforgeability game is negligible in the random oracle model.*

The proof is omitted because of space limitations as it is very similar to the proof of Theorem 1.

Theorem 7. *If the encryption schemes pkENC and symENC used are IND-CPA secure, and H is a cryptographic hash function, then* $Adv^{link}_{S,A}(\lambda)$ *of a NARROW-STRONG adversary, that runs in polynomial time, to win the unlinkability game is negligible in the random oracle model.*

The proof is omitted because of space limitations.

Theorem 8. *If Fun is a PRF, then* $Adv^{link}_{S,A}(\lambda)$ *of a polynomial-time adversary that possesses the private key of a valid reader to win the minimal unlinkability game is negligible in the random oracle model.*

The proof is omitted because of space limitations.

5 Related Work

In 2012, Elkhiyaoui, Blass, and Molva presented a protocol that allows an RFID reader to determine whether two tags store some attributes that jointly fulfill a boolean constraint, without violating the privacy of the tags [9]. They motivate their protocol by considering the transportation of chemicals where safety regulations prohibit the joint transportation of chemicals that might react with each other. By equipping each container with a tag and scanning for certain boolean constraint describing reactive combinations the reader can check if the transportation is safe. Elkhiyaoui *et al.* also focus extensively on privacy of their protocol, although this is problematic for their specific application: legal regulations for the transport of dangerous goods require a clear labeling which voids any of the privacy a tag might offer [10].

The speed-dating protocols described in this paper achieve the same goal with a different trade-offs between privacy and efficiency. Tags in our protocol are more costly, since they require to be able to perform calculations. The cost of calculations on tags is fairly low when symmetric encryption is used. Asymmetric encryption on tags is more expensive, but feasible to be implemented in both secure and efficient way, as numerous studies demonstrate in theory and practice [5,11,13,17]. As observed by other researches, asymmetric primitives will provide more secure systems [8,19]. Currently, many studies proposed protocols for which asymmetric encryption schemes are essential, [1,6,8] to mention a few. One of these protocols named "Yoking-Proofs" [15] is similar to speed-dating in the sense that it also considers two simultaneously scanned tags. What is different is the goal of the protocol: they provide a prove of the fact that two particular tags have been scanned simultaneously.

The advantage of using more costly tags is that the infrastructure of readers for our speed-dating protocols is more flexible and robust, because readers do not have to be connected to a central database. Additionally, unlike in the T-Match protocol presented in [9], readers do not need to perform any homomorphic encryption operations, or expensive multi-party computations.

The protocol described in Sect. 4.2 furthermore extends the protocol model to allow multiple attributes per tag. This makes speed-dating a suitable solution for a broader set of applications.

6 Conclusion

Three protocols to privately match attributes on RFID tags were presented in this paper. None of them requires a centralized system with readers constantly connected to a central database. Neither do readers require any knowledge about attributes stored on tags. This makes the system flexible and easy to use for several parties. The first protocol protects privacy of users by only utilizing symmetric encryption, which makes it extremely lightweight. This comes at a cost of a slightly lower protection level, than the one provided by the other two protocols. Just one step of asymmetric encryption that is required by both of them, quite noticeably changes anonymity protection.

There is a restriction in all of the presented protocols. All possible applications are limited to the ones, which are sensitive to false positives. That is, the protocol protects against matching tags with no matching attributes. These applications should not be sensitive to false negatives. The interesting future work is to see how protocol can be improved to add detection of false negatives.

Acknowledgements. We would like to express our very great appreciation to Andreas Hülsing for his valuable support on this project. His constructive suggestions and willingness to spend his time so generously is very much appreciated. We would also like to thank Ari Juels for very fruitful discussions.

References

1. Alpár, G., Batina, L., Lueks, W.: Designated attribute-based proofs for RFID applications. In: Hoepman, J.-H., Verbauwhede, I. (eds.) RFIDSec 2012. LNCS, vol. 7739, pp. 59–75. Springer, Heidelberg (2013)
2. Aumasson, J.-P., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: a lightweight hash. J. Cryptology **26**(2), 313–339 (2013)
3. Bellare, M., Goldreich, O., Mityagin, A.: The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive, Report 2004/309 (2004). http://eprint.iacr.org/2004/309/
4. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. **61**(3), 362–399 (2000)
5. Braun, M., Hess, E., Meyer, B.: Using elliptic curves on RFID tags. IJCSNS Int. J. Comput. Sci. Netw. Secur. **8**(2), 1–9 (2008)
6. Bringer, J., Chabanne, H., Icart, T.: Cryptanalysis of EC-RAC, a RFID identification protocol. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 149–161. Springer, Heidelberg (2008)
7. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS - a practical forward secure signature scheme based on minimal security assumptions. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 117–129. Springer, Heidelberg (2011)
8. van Deursen, T., Radomirović, S.: Insider attacks and privacy of RFID protocols. In: Petkova-Nikova, S., Pashalidis, A., Pernul, G. (eds.) EuroPKI 2011. LNCS, vol. 7163, pp. 91–105. Springer, Heidelberg (2012)
9. Elkhiyaoui, K., Blass, E.-O., Molva, R.: T-Match: privacy-preserving item matching for storage-only RFID tags. In: Hoepman, J.-H., Verbauwhede, I. (eds.) RFIDSec 2012. LNCS, vol. 7739, pp. 76–95. Springer, Heidelberg (2013)
10. Council of the European Union European Parliament. Regulation (EC) No 1272/2008 of the European Parliament and of the Council of 16 December 2008 on classification, labelling and packaging of substances and mixtures, amending and repealing Directives 67/548/EEC and 1999/45/EC, and amending Regulation (EC) No 1907/2006 (2008). http://new.eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32008R1272&rid=3
11. Fürbass, F., Wolkerstorfer, J.: ECC processor with low die size for RFID applications. In: IEEE International Symposium on Circuits and Systems, ISCAS 2007, pp. 1835–1838 (2007)
12. Goldwasser, S., Bellare, M.: Lecture notes on cryptography (2008). http://cseweb.ucsd.edu/users/mihir/papers/gb.pdf

13. Hein, D., Wolkerstorfer, J., Felber, N.: ECC is ready for RFID – a proof in silicon. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 401–413. Springer, Heidelberg (2009)
14. Hermans, J., Pashalidis, A., Vercauteren, F., Preneel, B.: A new RFID privacy model. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 568–587. Springer, Heidelberg (2011)
15. Juels, A.: "yoking-proofs" for RFID tags. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops 2004, pp. 138–143 (2004)
16. Juels, A., Weis, S.A.: Defining strong privacy for RFID. In: Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops '07, pp. 342–347 (2007). http://www.emc.com/emc-plus/rsa-labs/staff/bios/ajuels/publications/rfid_privacy/rfidprivacy.pdf
17. Lee, Y.K., Sakiyama, K., Batina, L., Verbauwhede, I.: Elliptic-curve-based security processor for RFID. IEEE Trans. Comput. **57**(11), 1514–1527 (2008)
18. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
19. Vaudenay, S.: On privacy models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)

http://www.springer.com/978-3-319-13065-1

Radio Frequency Identification: Security and Privacy Issues
10th International Workshop, RFIDSec 2014, Oxford, UK,
July 21-23, 2014, Revised Selected Papers
Saxena, N.; Sadeghi, A.-R. (Eds.)
2014, VIII, 215 p. 62 illus., Softcover
ISBN: 978-3-319-13065-1